UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Escuela Técnica Superior de Ingeniería del Diseño

Bachelor's Degree in Industrial Electronics and Automation Engineering

Final Year Engineering Project

QT SET-UP FOR TEXAS INSTRUMENTS AM335X ARM CORTEX-A8 USING LINUX: APPLICATION FOR AN INDUSTRIAL SCALE

Documents:

Author:
Pablo Cholbi Alenda

Supervisor:
Dr. Àngel Perles Ivars

May 2014

Dedicated to my parents,
for their love, endless support and encouragement.

Acknowledgements

I am grateful and would like to express my sincere gratitude to my supervisor Dr. Àngel Perles Ivars for his invaluable guidance, continuous encouragement and constant support as a teacher and as a supervisor, which made this project possible. Without his advice and assistance it would have been a lot tougher to complete.

I would also like to thank Dr. Francisco José Gimeno Sales for taking me as an intern during my third and fourth year of my undergraduate degree. I found that the practical experience I obtained at the Laboratory of Renewable Energies to be very valuable and eye-opening.

My sincere thanks go to all lecturers and members of the staff at the ETSID, UPV, who helped me in many ways and made my educational journey at UPV unforgettable. These four years have not only provided my with an excellent education but also a broader and inter-disciplinary view on science, engineering and the world. I will remember fondly this period as I face this new chapter in life as an engineer.

Lastly, I would like to thank any person who contributed to my final year project directly or indirectly. I would like to acknowledge that all comments and suggestions were crucial for the successful completion of this project.

Abstract

In recent years, there has been a large increase of ARM based consumer products. This surge has been motivated by the efficiency and low cost of these processors, which make them appropriate for low power devices and applications that are not CPU intensive. This work examines the viability of implementing a graphical touchscreen application for an industrial scale on a Texas Instruments AM335x ARM Cortex-A development board as opposed to x86 based systems.

This work also intends to serve as a general reference for implementing cross-platform applications with the Qt framework targeted to an embedded Linux system running on an AM335x based hardware.

Specifically, the application to be developed is an intuitive graphical touchscreen application that interacts with a Microgram IE21 industrial scale from Microgram Instruments Española S.A. over RS-232 and a SQLite relational database for bulk product commerce by weight. Product and sales information is retrieved and stored in the database.

Keywords: embedded Linux, ARM Cortex-A, Qt, industrial scale.

Resumen

En los últimos años, ha habido un gran aumento de los productos de consumo basado en ARM. Este incremento ha sido motivado por la eficiencia y el bajo costo de estos procesadores, que los hacen adecuados para dispositivos de baja potencia y aplicaciones que no hacen un uso intensivo de la CPU. Este trabajo examina la viabilidad de implementar una aplicación con interfaz gráfica táctil para una báfscula industrial en una placa de desarrollo Texas Instruments AM335x ARM Cortex-A a diferencia de los sistemas basados en x86.

Este trabajo también tiene la intención de servir como una referencia general para la implementación de aplicaciones multiplataforma con las bibliotecas Qt para sistemas Linux embebido que se ejecuta en un hardware basado en AM335x.

En concreto, la aplicación a desarrollar es una aplicación con una interfaz gráfica táctil intuitiva que interactua con una báscula industrial Microgram EI21 de Microgram Instrumentos Española SA a través de RS-232 y una base de datos relacional SQLite para el comercio de productos a granel al peso. La información sobre el producto y las ventas se recupera y se almacena en la base de datos.

Palabras Clave: Linux embebido, ARM Cortex-A, Qt, báscula industrial.

Resum

En els últims anys, hi ha hagut un gran augment dels productes de consum basat en ARM. Aquest increment ha estat motivat per l'eficiència i el baix cost d'aquests processadors, que els fan adequats per a dispositius de baixa potència i aplicacions que no fan un us intensiu de la CPU. Aquest treball examina la viabilitat d'implementar una aplicació amb interfície gràfica tàctil per a una bàscula industrial en una placa de desenvolupament Texas Instruments AM335x ARM Cortex-A a diferència dels sistemes basats en x86.

Aquest treball també té la intenció de servir com una referència general per a la implementació d'aplicacions multiplataforma amb el marc de treball Qt per a sistemes Linux encastat que s'executa en un maquinari basat en AM335x.

En concret, l'aplicació a desenvolupar és una aplicació amb interfície gràfica tàctil intuïtiva que interactua amb una bàscula industrial Microgram EI21 de Microgram Instruments Española SA mitjançant RS-232 i una base de dades relacional SQLite per al comerç de productes a granel al pes. La informació sobre el producte i les vendes es recupera i s'emmagatzema a la base de dades.

Paraules Clau: Linux encastat, ARM Cortex-A, Qt, bàscula industrial.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Escuela Técnica Superior de Ingeniería del Diseño

Bachelor's Degree in Industrial Electronics and Automation Engineering

Final Year Engineering Project

QT SET-UP FOR TEXAS INSTRUMENTS AM335X ARM CORTEX-A8 USING LINUX: APPLICATION FOR AN INDUSTRIAL SCALE

1 - Report

Author:
Pablo Cholbi Alenda

Supervisor:
Dr. Àngel Perles Ivars

May 2014

Table of Contents

Continued...

Illustration Index

1. Background

In recent years there has been an increase in the number of smartphones and tablets. This change has not only brought new types of applications and programming philosophy to meet the new needs of the population, but it has also pushed the development of embedded processors. Users demand relatively high computational power with a low energy consumption maintain a long battery life.

ARM processors now dominate the mobile phone space due to their lower power requirements with 98% of mobile phones having an ARM processor [Roberts-Hoffman 2009], and is expanding its market in set-top boxes ans smart TVs. ARM processors dominate these markets because they are more efficient than their x86 counterparts and have a very competitive price.

It should also be noted that, at the moment, ARM processors cannot achieve the performance levels of some x86 systems, which still have the majority of the market share in laptops, desktops and servers.

The following graph, extracted from [Ou 2012], compares the power efficiency with CPU utilization of a ARM A9 cluster and an Intel dual core x86 workstation for web server application at different file sizes.

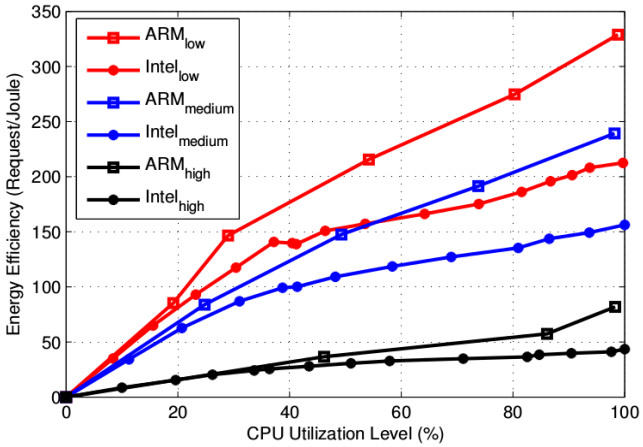


Figure 1-1: Energy-efficiency vs. CPU Utilization Level [Ou 2012]

With CPU utilization higher than 20%, the ARM A9 processor presents an advantage in efficiency over Intel processor.

The following graph, extracted from [Blem 2013], compares power consumption with performance of 2 ARM processors and 2 Intel x86 processors.

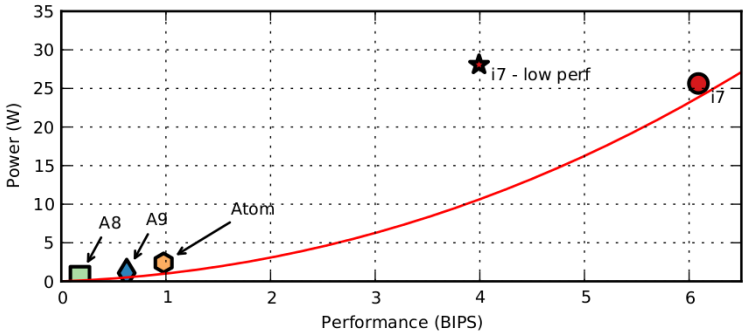


Figure 1-2: Power Performance Trade-offs [Blem 2013]

Regardless of ISA or energy-efficiency, high-performance processors require more power than lower- performance processors. They follow well established cubic power/performance trade-offs [Blem 2013].

The previous graphs suggest that if the task a computer must perform is not CPU intensive and/or can be subdivided into smaller tasks to run concurrently, it is advantageous, from the point of view of efficiency, to use an ARM based distributed system or cluster.

This hypothesis has been analysed in various studies in recent years. [Ou 2012] and [Vidal 2012] both conclude that ARM based clusters for server applications (web and web, file and database respectively) is a viable and power efficient solution. Likewise, [Abdurachmanov 2014] concludes that an ARM based cluster is a viable and power efficient solution for scientific computation at the CERN.

The results of the aforementioned studies suggest that the use of ARM processors in industrial computers could also be advantageous. It is worth noting that ARM processors are not only efficient, but there are cost-effective solutions on the market. A good example of this is the BeagleBone Black, a 45$ community-supported development platform with 512MB of RAM, TI AM3359 1.0GHz processor and a power consumption <2W.

The disadvantage that ARM based systems present versus x86 industrial PCs is binary incompatibility, x86 systems are well established and have been the standard for many years. Much software has been developed for these systems, most of which can not be easily ported to the ARM architecture due to copyright reasons and/or framework incompatibility.

By developing cross-platform applications, it is possible to use ARM based systems as industrial computes and benefit from the advantages of this platform over traditional x86 systems while developing applications that can easily be ported to said systems. Therefore, this work does not only present the implementation of an application for an industrial scale for an ARM development board, but it is also intended to serve as reference on how to develop cross-platform application for ARM embedded Linux systems that x86 computers can also benefit from.

By developing cross platform applications, the hardware and OS range of possibilities becomes larger and a more optimal solution can be found for a given problem.

2. Factors to Consider

2.1 Initial Specifications

- Touchscreen graphical open source application for an industrial scale using the Qt framework.
- Texas Instruments Sitara AM335x processor running Linux.
- System shall receive the output of a Microgram IE21 industrial scale from Microgram Instruments Española S.A. industrial scale over RS-232.
- Simple and intuitive software, such that the users are able to deduce how to use the application without any prior knowledge or help.
- The scale is intended to weigh fruit and vegetables, but the database should be generic such that it can be used for commerce of other goods by weight. The image, name, index, price and accumulated sales shall be stored in a relational database.
- The Application must automatically update the GUI when a product is added, removed or updated in the database.
- The system shall be network connected and have a fixed IP address assigned for easy maintenance and database access.
- Critical errors and non-critical errors should be handled differently. The application should exit immediately when a critical error is encountered. A warning system is allowed for non-critical errors.
- The application shall launch on system boot.
- The application shall have a system shutdown and system restart under the advanced options menu.
- The application shall be as modular as possible for simplicity and source code reuse.
- A low-cost and low-power solution should be favoured as the adopted solution.
- The application shall be cross-platform, developed under a framework which allows for the code to be easily compiled to run on different processor architectures and operational system.

2.2 Needs Study

The system is requires to have a touchscreen graphical user interface. The hardware platform should either come with such screen or have the possibility of adding one via off-the-shelf products via additional I/O such as GPIOs and/or HDMI + USB or any combination of these or any other valid media interface (VGA, DVI, etc.). In any case, the hardware platform must have the I/O capability for the selected screen. It is worth noting that the touchscreen can be a generic screen or one made specifically for the hardware platform. An example of the later are the many third party BeagleBone LCD “capes” for the Texas Instruments BeagleBone Black.

The system is also required to interface with an industrial scale over RS-232. Modern embedded systems are likely to not have a RS-232 ±15V DB-9 (or DB-25) port present, therefore, a USB-to-serial adapter shall be used in this project. This implies that the hardware platform must have at least 1 available USB (host) port. As the data transfer rates of RS-232 is much smaller than that of the USB protocol, a USB 1.1 or USB 2.0 port is sufficient. A USB 3.0 or USB 3.1 port is not required.

As the system is required to have a static IP address for easy database management and system maintenance, it is implied that the hardware platform must have at least 1 Ethernet Port. The purpose of this port is to establish an SSH or SFTP. For the expected loads, high data transfer rate is not required and therefore, a Gigagit Ethernet port is not required. A 100Mb/s Ethernet port is sufficient.

The software shall be designed to be cross-platform from the standpoint of the operating system and the hardware architecture. The application shall have a “native appearance” on all the platforms it is executed.

Finally, for the development of the application, the use of open source tools, libraries and compilers shall be favoured over closed source alternatives as they, amongst other advantages, improves the maintainability of the software and it grants independence from the supplier. It is worth noting that the use of open source tools in the development and documentation of the project are also favoured because of the client's and the author's philosophy.

3. Alternative Solutions

3.1 Alternatives for the Hardware

In this section, some of the most popular single-board ARM computers that are Linux capable shall be compared.

	Raspberry Pi B	A10-OLinuXino	BeagleBone Black	ODROID-U3	Cubieboard 2
Processor	ARM11 700MHz	Cortex-A8 1GHz	Cortex-A8 1GHz	Cortex-A9 Quad 1.7GHz	Cortex-A7 Dual 1GHz
GPU	Video Core IV	Mali-400	SGX530	Mali-400	Mali-400
RAM	512MB	512MB	512MB	2GB	1GB
Flash	0MB	0MB	2GB	0MB	4GB
USB	2	2 host + 1 OTG	1 host + 1 OTG	3 host + 1 OTG	2 host + 1 OTG
Ethernet	1×10/100M	1×10/100M	1×10/100M	1×10/100M	1×10/100M
WiFi	✗	✗	✗	✗	✗
Bluetooth	✗	✗	✗	✗	✗
SATA	✗	✓	✗	✗	✓
GPIO	17	160	65	Expansion	96
SPI	✓	✓	✓	Expansion	✓
I2C	✓	✓	✓	Expansion	✓
Price	$35 USD	30€ (≈ $41 USD)	$45 USD	$59 USD	$65 USD
Comment	Large Community		Large Community		

ODROID-U3 offers the best value, but it is clear that with its specifications and lack of stock GPIOs it is intended as a lightweight workstation. Nevertheless, it can be considered for industrial computer applications.

On the other hand, the ODROID-U3 is overpowered for the application described in this work. The BeagleBone Black or the A10-OLinuXino are better suited for the intended application in this sense.

Out of the two previously mentioned development boards, the BeagleBone Black is considered the best solution for the intended application. Despite the BeagleBone Black having slightly less features than the ODROID-U3, it has a larger community and more support, which should not be overlooked when choosing a hardware platform as it can reduce development times and ease operating system and package support.

It should be noted that the BeagleBone Black is considered the best hardware solution for this project, the application shall be develop on a Texas Instruments TMDSSK3358 AM335x Starter Kit, as this board was physically available for use at the Universitat Politècnica de València at the time of writing.

The Texas Instruments TMDSSK3358 AM335x Starter Kit has more features overall and has a higher price, but the BeagleBone Black and the TMDSSK3358 AM335x Starter Kit both have an AM335x processor, and therefore, the procedures to develop applications and the resulting binaries are the same in both platforms.

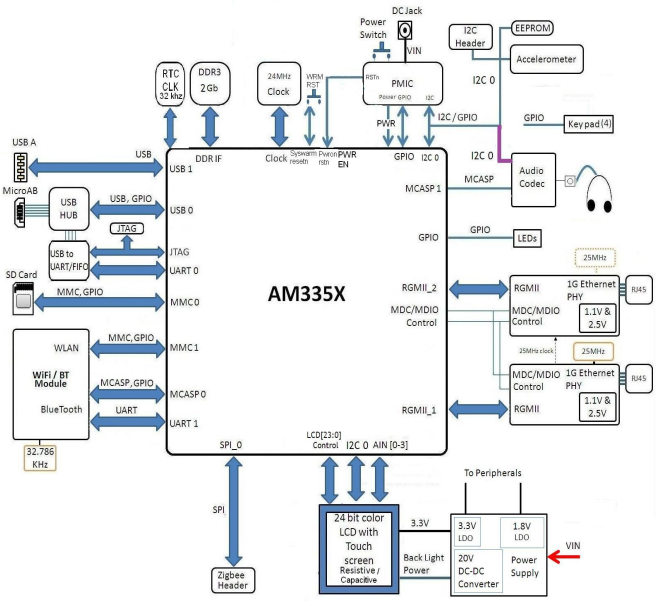


Figure 3.1-1: Major Functional Blocks of the TMDSSK3358

3.2 Alternatives for the Programming Language

There are a lot options for selecting a programming language, but this section will focus on comparing three of the most popular and predominant languages used to develop cross-platform applications.

3.2.1 Java

Pros	Cons
• Simple and powerful. • Very popular.	• Slower than low level languages. • Less efficient than native compiled languages. • Many vulnerabilities. • Requires a virtual machine.

Java applications run on top of virtual machines, which makes them slower and much more resource hungry than native compiled languages. Aside from efficiency, there are many security risks involved in using Java's virtual environment. Because of the aforementioned reasons, Java has not been considered an appropriate solution for this project, although it is a valuable option.

The rise of the android operating system in recent years is a good example of the potential of Java for the industry as the are a wide rage of hardware architectures running Android and they all use the same package system and binaries, which may simplify the maintainability and distribution of applications.

3.2.2 Python

Pros	**Cons**
• Simple. • Very popular.	• Slower than low level languages. • Less efficient than native compiled languages. • Prone to errors at development stage.

Python is currently a very popular language that has a large community, but it was designed to be used as a scripting language, which inherently makes it much slower than compiled languages. Also, being an interpreted language, many common error, which would otherwise be detected at compile time, are not detected until the application is run and debugged, this makes development times longer for large applications. Because of the aforementioned reasons, like Java, Python has not been considered an appropriate solution for this project, but is also a valuable option.

3.2.3 C++

Pros	**Cons**
• Very efficient. • Many libraries.	• Applications compiled for a specific architecture and operating system. • Complex syntax

C++ is widely accepted in the field of programming, and given the amount of documentation and the efficiency of the resulting applications, C++ is a good alternative with which to develop the application described in this work. There are interesting frameworks on top of C++ such as GTK, wxWidgets, Qt, etc. that may be used for the development of the application.

With the C++ Qt framework it is possible to develop cross-platform applications that can easily compiled to target various architectures and operating systems without modification of the source code. Specifically, there is a branch of the framework intended for Embedded Linux. This branch allows to create graphical applications that use a virtual frame buffer. The application itself provides the frame buffer, and a Qt application for embedded Linux is already running, other applications will connect to the first application's frame buffer, This saves on memory usage and does not require a fully fledged X11 server to be running on the target system.

3.3 Alternatives for the SQL Based Database Engine

There are a lot good options for selecting a database engine, but this section will focus on comparing three of the most popular and predominant database engines.

3.3.1 MySQL / MariaDB

Pros

- Scalable.
- Good performance features.
- Supports user permission.
- Good for client-server applications.
- Good with multi-master replication.

Cons

- Complex set-up.
- Requires server.

MySQL is popular in medium-to-high size systems. It has a large community and there are many open source project developed with this database system, which makes it relatively easy to set up and borrow code from these applications to reduce development time. On the other hand, set-up more complex than other database system and there are limitation when using it in large systems or when data accuracy is extremely important. It is intended for systems larger than the application to be developed. Because of the aforementioned reasons, MySQL is not considered and appropriate solution for this project.

It is worth noting that MySQL and MariaDB are not the same database engine, but they are closely related, as they both have the same initial author. MariaDB is as a community-developed fork of MySQL.

The goal for Maria-DB is to be a drop-in replacement for MySQL – with more features and better performance. MariaDB is based on the corresponding version of MySQL, if one exists. For example, MariaDB 5.1.53 is based on MySQL 5.1.53, with some added bug fixes, additional storage engines, new features, and performance improvements. [Bartholomew 2013]

3.3.2 PostgreSQL

Pros

- Scalable.
- Many performance features.
- Supports user permissions.
- Conforms with SQL standard
- Conforms with ACID standard
- Easily portable to Oracle SQL.

Cons

- Complex set-up.
- Requires server.
- Smaller community.

PostgreSQL is slower than other database engines such as MySQL, but it is much more feature-rich. It also conforms with the SQL standard and is specially appropriate if there is a possibility of later moving to Oracle DB or many other closed source database systems, as for the most part, code which runs PostgreSQL will run on these systems. On the other hand, it is harder to set up, there is less documentation and it has a smaller community. It is intended for systems larger than the application to be developed. Because of the aforementioned reasons, PostgreSQL is not considered and appropriate solution for this project.

3.3.3 SQLite

Pros	**Cons**
• Fast set-up.	• Not many performance features.
• Rapid development.	• Slow write operations.
• Good for embedding.	• Does not scale well.
• Single file database.	• Does not have user management.
• Easy backup.	• Not appropriate for large databases.
• Many supported programming languages.	• Not appropriate for high concurrency,
• Low overhead.	
• Fast read operations.	

SQLite was not designed to compete with MySQL, PostgreSQL or Oracle DB. Although it can be used, for example, on websites that do not have a very high traffic ($<10^6$ hits/day), but it excels in embedding small databases into application. Due to it's efficiency, ease of use and low maintenance, this server-less single-file database seems the best alternative with which to develop the application described in this work, as it is one of the situations it was designed for. Moreover, due to the characteristics of the applications to be developed, none of the disadvantages of this database file suppose a limitation.

According to [Allen 2010], *"One advantage of having a database server inside your program is that no network configuration or administration is required. [...] no firewalls or address resolution to worry about, and no time wasted on managing intricate permissions and privileges. Both client and server run together in the same process. This reduces overhead related to network calls, simplifies database administration, and makes it easier to deploy your application. Everything you need is compiled right into your program"*.

4. Development Environment Set-up

This section is a reference on how to set up a development environment to develop and debug Qt applications targeted to AM335x processors. Texas Instruments (TI) provides an SDK with everything necessary to develop Qt applications for this target.

4.1. Host Computer

Development must be done on a x86 Linux computer. TI strongly recommends to install the SDK and develop on an Ubuntu LTS release as this is the environment that was used to develop and test the SDK, but please note that this is a suggestion and it does not prevent the SDK from installing on other Linux distributions.

4.1.1 Installing TI SDK

First, the latest SDK from TI must be downloaded. At the time of writing, the newest version was 6.00.00.00.

```
user@debian:~$ cd Downloads/
user@debian:~/Downloads$ wget
  downloads.ti.com/sitara_linux/esd/AM335xSDK/latest/exports//ti
  -sdk-am335x-evm-06.00.00.00-Linux-x86-Install.bin
```

Once the download is complete, the binary must be made executable.

```
user@debian:~/Downloads$ chmod +x ti-sdk-am335x-evm-06.00.00.00-
  Linux-x86-Install.bin
```

Now a directory, owned by the current user, shall be created for the installation of the SDK.

```
user@debian:~/Downloads$ sudo mkdir /opt/ti-sdk-am335x-evm-
  06.00.00.00/
user@debian:~/Downloads$ sudo chown $USER /opt/ti-sdk-am335x-
  evm-06.00.00.00/
```

Finally the installation process can be started.

```
user@debian:~/Downloads$ ./ti-sdk-am335x-evm-06.00.00.00-Linux-
  x86-Install.bin
```

Note that when the asked for the installation directory, the previously created directory was selected.

4.1.2 Installing Qt IDE

First, the latest Qt Installer must be downloaded. At the time of writing, the newest version was Qt 5.2.0 with Qt creator 3.0.0.

```
user@debian:~$ cd ~/Downloads
user@debian:~/Downloads$ wget download.qt-
  project.org/official_releases/qt/5.2/5.2.0/qt-linux-
  opensource-5.2.0-x86_64-offline.run
```

Once the download is complete, the binary must be made executable.

```
user@debian:~/Downloads$ chmod +x qt-linux-opensource-5.2.0-
  x86_64-offline.run
```

Now a directory, owned by the current user, shall be created for the installation of the IDE.

```
user@debian:~/Downloads$ sudo mkdir /opt/Qt-5.2.0/
user@debian:~/Downloads$ sudo chown $USER /opt/Qt-5.2.0/
```

Finally the installation process can be started.

```
user@debian:~/Downloads$ ./qt-linux-opensource-5.2.0-x86_64-
  offline.run
```

Note that when the asked for the installation directory, the previously created directory was selected.

To be able to develop with the SDK, there are shell variables that must be set. TI provides an environment set-up file to automate this task. To ease this task further, a start-up script that sources the environment set-up file before launching Qt creator can be used.

```
user@debian:~$ nano /opt/Qt-5.2.0/launch-sitara.sh
```

The content of the script being:

```
#!/bin/bash
source /opt/ti-sdk-am335x-evm-06.00.00.00/linux-devkit/environment-setup
/opt/Qt-5.2.0/Tools/QtCreator/bin/qtcreator
exit 0
```

To make the script executable:

```
user@debian:~$ chmod +x /opt/Qt-5.2.0/launch-sitara.sh
```

From now on, when starting Qt creator to develop for an AM335x target, the following command shall be executed.

```
user@debian:~$ /opt/Qt-5.2.0/launch-sitara.sh
```

4.1.3 Configuring Qt kit for AM335x

The following section is a commented extract, with slight modifications of part of the Texas Instruments Document: *AMSDK Linux User's Guide*, [TI-Wiki-1].

4.1.3.1 Configuring qmake

- Click on the **Tools** → **Options** from the top menubar.
- On the left side vertical menubar click **Build & Run.**
- Click the **Qt Versions** tab under Build & Run.
- Click **Add...** on the right.
- Navigate to */opt/ti-sdk-am335x-evm-06.00.00.00/linux-devkit/sysroots/i686-arago-linux/usr/bin*.
- Select *qmake* then click on **Open.**
- Double click on **Version Name** and give the Qt Version a descriptive name such as “Qt 4.8.3 Sitara”**.**
- Click **Apply** to save your changes.

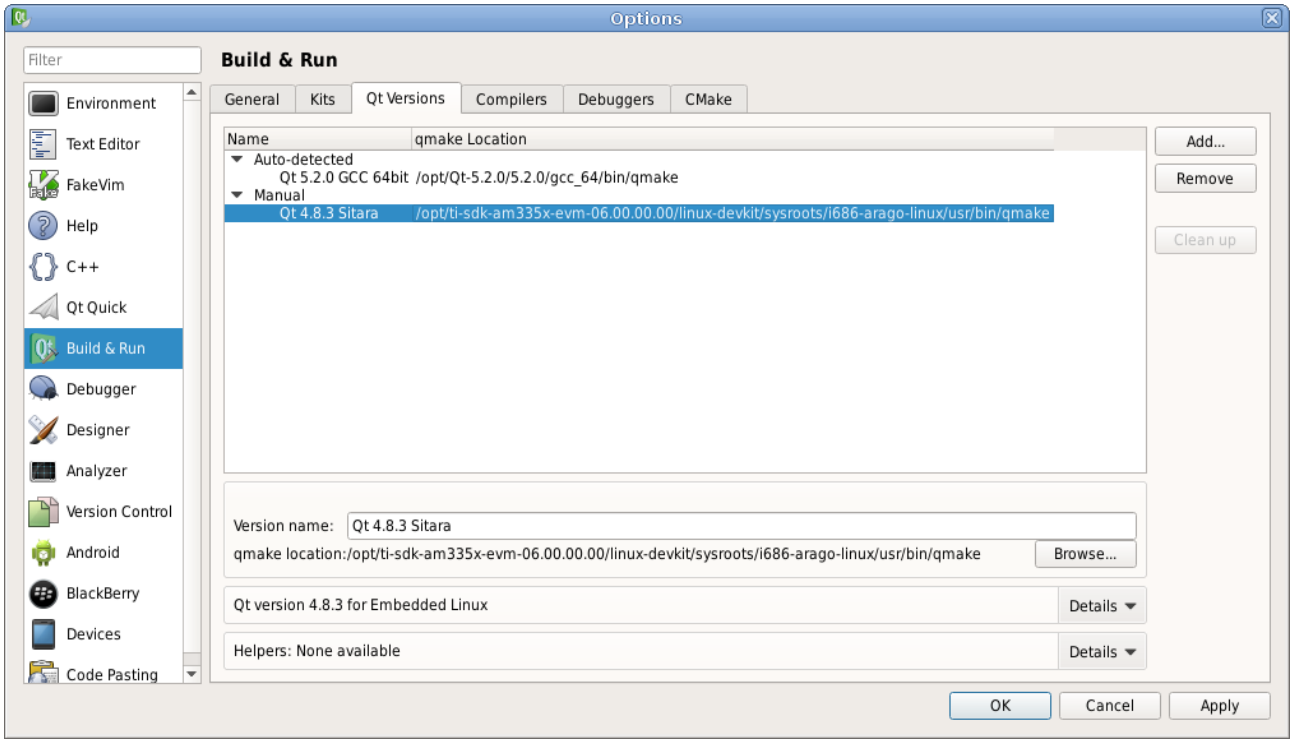


Figure 4.1-1: Configuring qmake

4.1.3.2 Configuring Compiler

- Click on the **Tools** → **Options** from the top menubar.
- On the left side vertical menubar click **Build & Run.**
- Click the **Compilers** tab under Build & Run.
- Click **Add** in the top right and add a **GCC**.
- Change the name to *arm-arago-GCC*. This can be done by modifying the “Name” field.
- To set the Compiler Path select **Browse**.
- Navigate to */opt/ti-sdk-am335x-evm-06.00.00.00/linux-devkit/sysroots/i686-arago-linux/usr/bin*.
- Select *arm-linux-gnueabihf-gcc* and click on **Open**.
- Click **Apply** to save your changes.

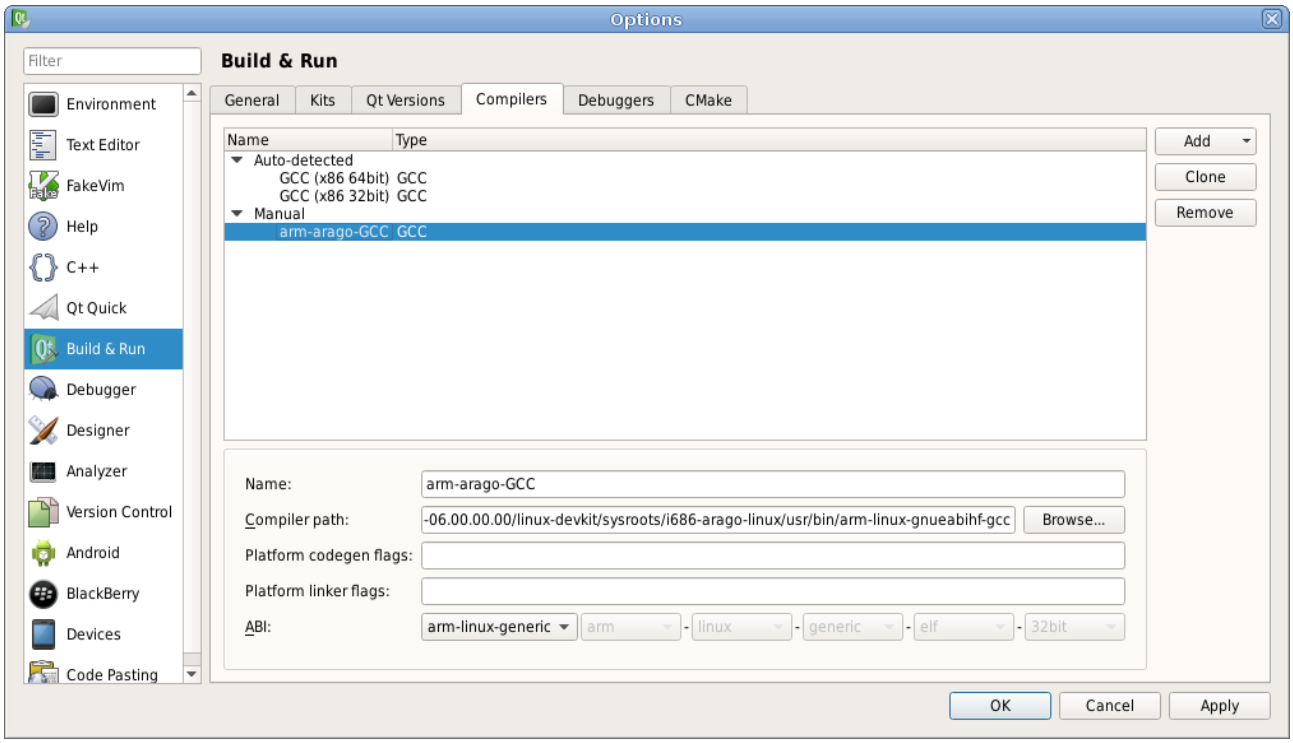


Figure 4.1-2: Configuring Compiler

4.1.3.3 Configuring Debugger

- Click on the **Tools** → **Options** from the top menubar.
- On the left side vertical menubar click **Build & Run.**
- Click the **Debugger** tab under Build & Run**.**
- Click **Add**.
- Change the name to *arm-arago-GDB*. This can be done by modifying the “Name” field.
- To set the Debugger Path select **Browse**.
- Navigate to */opt/ti-sdk-am335x-evm-06.00.00.00/linux-devkit/sysroots/i686-arago-linux/usr/bin*.
- Select *arm-linux-gnueabihf-gdb* and click on **Open**.
- Click **Apply** to save your changes.

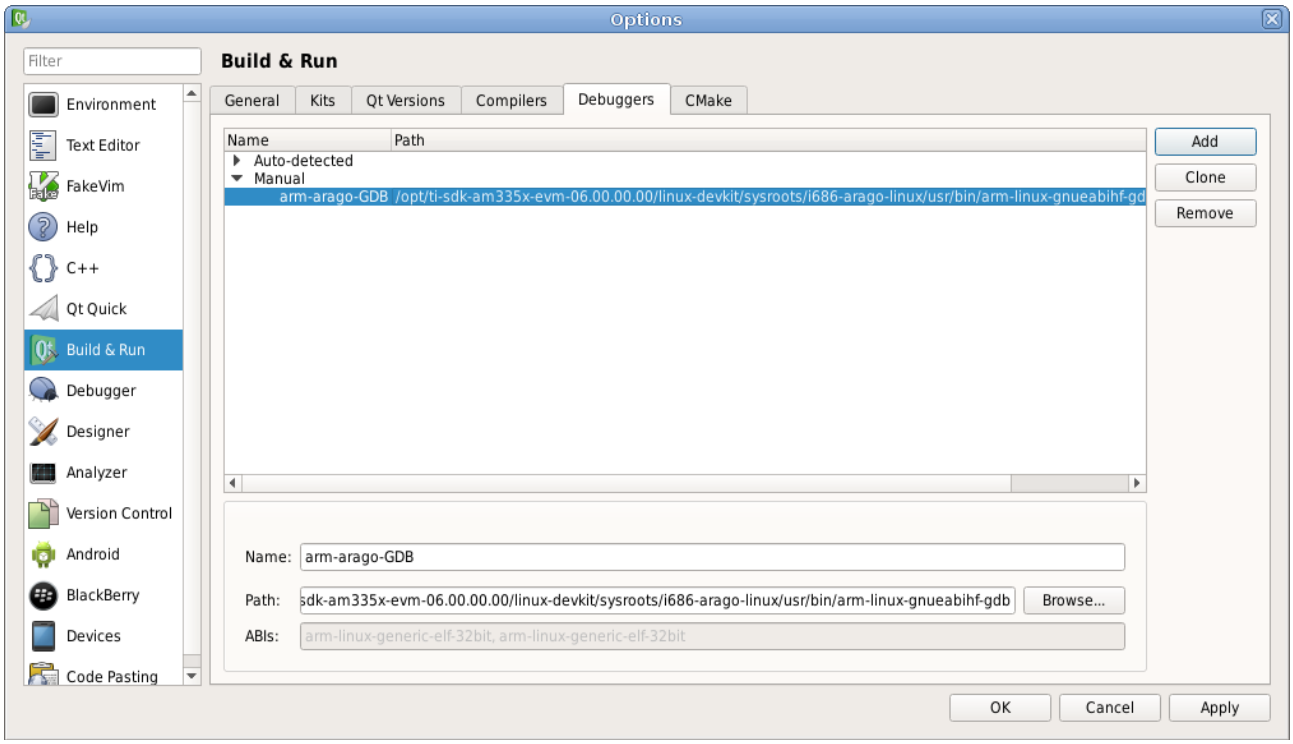


Figure 4.1-3: Configuring Debugger

4.1.3.4 Set-up Target

- Click on the **Tools** → **Options** from the top menubar.
- On the left side vertical menubar click **Devices.**
- Click **Add...** in the top right and select **Generic Linux device** and click on **Start Wizard**
- Change the name to *AM335x EVM*. This can be done by modifying the “Name” field.
- Type in the IP address of the embedded Linux device.
- Type in the password of the user.
- Type in the user (on the target) for SSH connection.
- Click **Next** and click **Finish**.
- Qt creator will attempt to connect to the embedded Linux device to check if correctly configured.

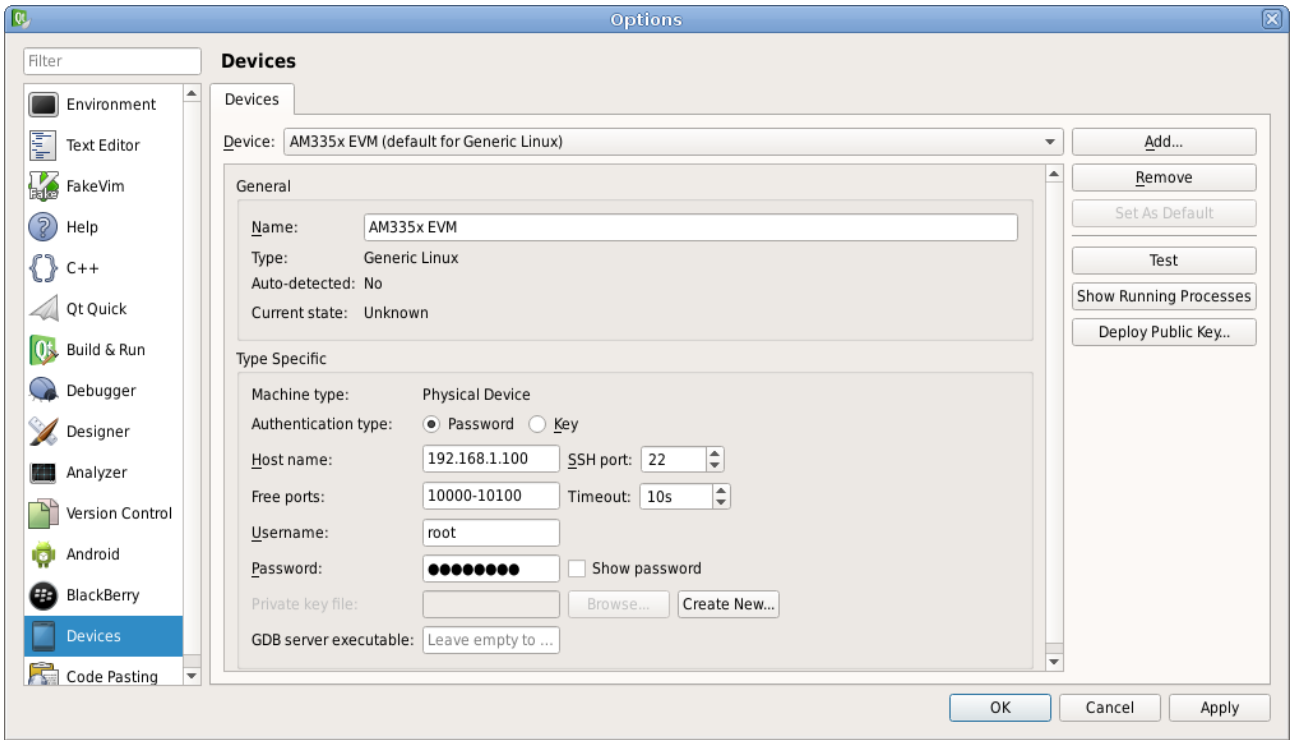


Figure 4.1-4: Configuring Device

4.1.3.5 Kit Configuration

- Click on the **Tools** → **Options** from the top menubar.
- On the left side vertical menubar click **Build & Run.**
- Click the **Kits** tab under Build & Run**.**
- Click **Add**.
- Change the name to *Sitara Qt 4.8.3*. This can be done by modifying the “Name” field.
- In the rest of the fields, select the previously configured components.
- To set the Sysroot Path select **Browse**.
- Navigate to */opt/ti-sdk-am335x-evm-06.00.00.00/linux-devkit/sysroots/i686-arago-linux.*
- Click **Choose**.
- Click **Apply** to save your changes.

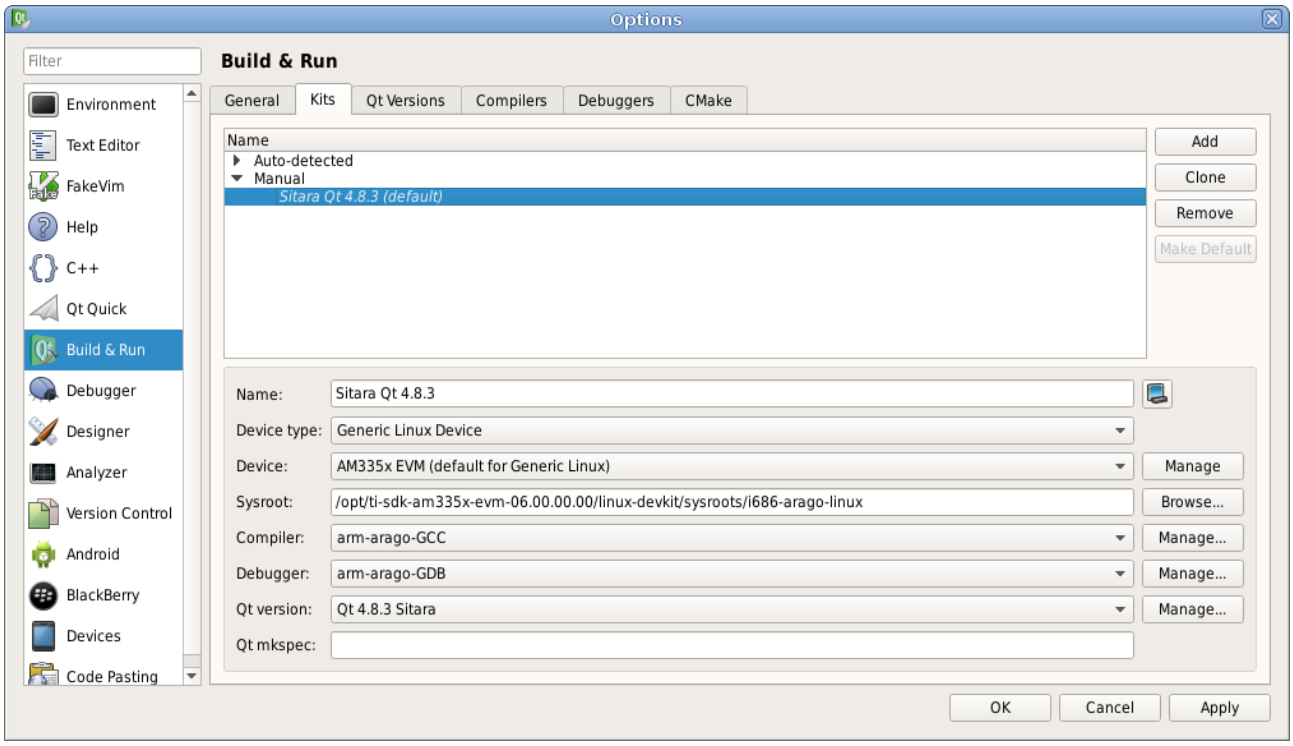


Figure 4.1-5: Kit Configuration

4.2. Target System

This section focuses on creating and configuring a bootable image for the target system such that the target system is set-up and ready to be developed on.

The board used in this work is a Texas Instruments TMDSSK3358 AM335x Starter Kit. But the set-up and configuration described in this section is valid for other TI systems based around the AM335x processor family.

4.2.1 Creating SD Card Image

TI provides a straightforward script with which to create SD card images within the SDK. To create a bootable SD card, insert a mini SD card of at least 2GB of capacity into a card reader on the host computer and execute the following commands.

```
user@debian:~$ cd /opt/ti-sdk-am335x-evm-06.00.00.00/bin/
user@debian:/opt/ti-sdk-am335x-evm-06.00.00.00/bin$ ./create-
  sdcard.sh
```

The script allows for various combinations, but the following options are suggested in this work given set-up.

- Select SD card drive.
- Select the 2 partition scheme.
- Yes, continue.
- Install pre-built images from SDK

Finally, before removing the mini SD card, the following command should be executed to force changed blocks to disk and update the super block.

```
user@debian:/opt/ti-sdk-am335x-evm-06.00.00.00/bin$ sudo sync
```

4.2.2 Initial Configuration

Before powering up the device, it is necessary to insert the mini SD card and connect a network cable to a router or switch on the same network as the host computer.

If using the AM335x Starter Kit, the network cable should be connected to the RJ45 jack farthest from the female micro-USB jack. This, on the TI image created previously, is interface eth0, which is enabled on start-up. The other RJ45 jack is interface eth1, which is neither configured nor enabled on start-up and therefore is not useful at the moment.

Once boot-up is complete, the TI Matrix-GUI should be visible on screen.

4.2.2.1 Login

To check the IP address of the device tap on **Settings** →**Network Settings**. The IP address should be listed under the eth0 interface section. For example, the IP address could be 192.168.1.33.

Once the IP address of the device is know, it is possible to start a SSH session from the host computer. By default, the root user does not have a password. When asked for the password, just hit return to log on.

```
user@debian:~$ ssh 192.168.1.33 -l root
root@192.168.1.33's password:
root@am335x-evm:~#
```

For security reasons, it is advisable to set a password for the root user.

```
root@am335x-evm:~# passwd
```

4.2.2.2 Disabling the Default Matrix-GUI Application

This section describes how to disable the TI Matrix-GUI from starting on boot if it is not needed or undesired.

First it is necessary to determine which is the default runlevel, which varies among different Linux distributions.

```
root@am335x-evm:~# runlevel
```

The previous command returns the current runlevel, which is assumed to be the default on boot. In this case, the default runlevel seems to be runlevel 5.

To list the services which are started (or stopped) on runlevel 5, the following command is executed.

```
root@am335x-evm:~# ls -l /etc/rc5.d
```

From the previous command it is possible to determine that S97matrix-gui-2.0 starts the Matrix-GUI service.

Finally, to disable the starting of this service on boot on the default runlevel.

```
root@am335x-evm:~# cd /etc/rc5.d
root@am335x-evm:/etc/rc5.d# mv S97matrix-gui-2.0 K97matrix-gui-
  2.0
```

4.2.2.3 Static IP Address

This section describes how to assign the device a static IP address to make is easily accessible and identifiable over a network.

Before modifying the network interface configuration file, it is advisable to make a backup.

```
root@am335x-evm:~# cp /etc/network/interfaces
  /etc/network/interfaces.bak
```

Now it is safe configure the network interfaces.

```
root@am335x-evm:~# vi /etc/network/interfaces
```

The section that configures the eth0 interface can be replaced with:

```
auto eth0
iface eth0 inet static
    address 192.168.1.100
    gateway 192.168.1.1
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
```

Note that with this configuration the device is assigned IP address 192.168.1.100.

4.2.2.4 Directory for Binaries

This section describes the creation of a dedicated directory for third party binary files and the inclusion of such directory in the binary search path.

The following command is executed to create the directory */usr/local/bin*, into which user developed applications shall be copied.

```
root@am335x-evm:~# mkdir -p /usr/local/bin
```

To add this directory automatically to the *$PATH* variable at login if the directory exist, a profile file is created.

```
root@am335x-evm:~# vi ~/.profile
```

The content of the file being:

```
#f [ -f ~/.bashrc ]; then
# . ~/.bashrc
#fi

# set PATH so it includes third party bin if it exists
if [ -d "/usr/local/bin" ] ; then
    PATH="$PATH:/usr/local/bin"
fi

#mesg n
```

4.2.2.5 USB-to-Serial Adapter Support

Please, note that this section is included for the TMDSSK3358 AM335x Starter Kit. The BeagleBone Black comes with support for USB-to-serial adapters out-of-the-box.

The application to be developed shall communicate with an industrial scale that offers an RS-232 output with a USB-to-serial adapter. However, the kernel currently installed on the target system does not have the drivers for these devices. It is necessary to compile a kernel with USB-to-serial adapter support.

There are many adapters on the market with different chips. In this project, the Prolific PL2303 chip is used as it is the integrated circuit used inside the USB-to-serial adapter used in the development of this project. Therefore, this section is centred around obtaining support for this device, but similar steps must be taken with other devices.

The following section is a commented extract, with slight modifications of part of the Texas

Instruments Document: *AMSDK Linux User's Guide*, [TI-Wiki-2].

In order to build the Linux kernel you will need a cross compiler installed on your system which can generate object code for the ARM core in your Sitara device. To add the compiler to the path, the following command is executed.

```
user@debian:~$ export PATH="/opt/ti-sdk-am335x-evm-
  06.00.00.00/linux-devkit/sysroots/i686-arago-linux/usr/bin:
  $PATH"
user@debian:~$ cd /opt/ti-sdk-am335x-evm-06.00.00.00/board-
  support/linux-3.2.0-psp04.06.00.11
```

Prior to compiling the Linux kernel it is often a good idea to make sure that the kernel sources are clean and that there are no remnants left over from a previous build. The command to clean the kernel is:

```
user@debian:~$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
  mrproper
```

Before compiling the Linux kernel it needs to be configured to select what components will become part of the kernel image, which components will be build as dynamic modules, and which components will be left out all together. This is done using the Linux kernel configuration system.

It is often easiest to start with a base default configuration and then customize it for you use case if needed. In the Linux kernel a command of the form:

```
user@debian:/opt/ti-sdk-am335x-evm-06.00.00.00/board-
  support/linux-3.2.0-psp04.06.00.11$ make ARCH=arm
  CROSS_COMPILE=arm-linux-gnueabihf- tisdk_am335x-evm_defconfig
```

After the configuration step has run the full configuration file is saved to the root of the kernel tree as *.config*. Any further configuration changes are based on this file until it is clean-up up by doing a kernel clean as mentioned above.

Now, the only modification that need to be done to the configuration file is to set the *CONFIG_USB_SERIAL* option.

```
user@debian:/opt/ti-sdk-am335x-evm-06.00.00.00/board-
  support/linux-3.2.0-psp04.06.00.11$ sed -i 's/#
  CONFIG_USB_SERIAL is not set/CONFIG_USB_SERIAL=y/g' .config
```

To compile the kernel, the following command is executed.

```
user@debian:/opt/ti-sdk-am335x-evm-06.00.00.00/board-
  support/linux-3.2.0-psp04.06.00.11$ make ARCH=arm
  CROSS_COMPILE=arm-linux-gnueabihf- uImage
```

At this point, before the compilation begins, as the USB-to-serial flag was set, the *make* executable will ask whether or not to enable support for a variety of devices. For the purpose of this project, only the *USB_SERIAL_PL2303* is enabled.

Finally it is necessary to install the kernel. Installation is done by moving the kernel image to the location where it is read on the target system. With the following sequence, a backup of the current kernel is made, the new kernel is transferred over to the target system and the target system is then

rebooted.

```
root@am335x-evm:~# cp /media/mmcblk0p1/uImage
  /media/mmcblk0p1/uImage.bak
```

```
user@debian:/opt/ti-sdk-am335x-evm-06.00.00.00/board-
  support/linux-3.2.0-psp04.06.00.11$ scp arch/arm/boot/uImage
  root@192.168.1.100:/media/mmcblk0p1/uImage
```

```
root@am335x-evm:/boot# shutdown -r now
```

If everything has gone well, the system will come up as normal.

To check if the new kernel is running, the following command can be executed. The date in the output of the command is the compilation date of the kernel that is currently running.

```
root@am335x-evm:~# uname -a
```

At this point, if the USB-to-serial adapter is connected then the following command is executed, the output of the command should inform that the adapter is now attached to *ttyUSBx*, where "x" is an integer.

```
root@am335x-evm:~# dmesg
```

4.2.2.6 Touchscreen Calibration and Test

TI provides utilities to calibrate and test the touchscreen with the system image.

```
root@am335x-evm:~# ts_calibrate
root@am335x-evm:~# ts_test
```

The *ts_calibrate* command prints the screen resolution. This will be useful to create applications of exactly the screen size. In this case, the touchscreen has a resolution 480×272 pixels.

4.3. Application Template

To create a basic project for the target system, the following steps have been followed.:

- Click on the **File → New File or Project** from the top menubar.
- At the Top right corner select **Embedded Linux Template**.
- Then select **QT Widget Application** from the centre list.
- Click on **Choose**.
- Name the project **Template-Sitara**.
- Click **Next**.
- Select the previously configured **Sitara Kit**.
- Click **Next**.
- In the Class Information window, select **QWidget Base Class**.

- Click **Next**.
- Leave version control to **<none>**.
- Click **Finish**.

With the project created, only some minor configuration remains to be done.

The size of the widget should match the screen size of the device. To do this, inside the UI editor, using the property editor, edit the following:

QWidget → Geometry → Width → 480 and **QWidget → Geometry → Height → 272**.

For testing purposes, a label with the text "*Hello World!*" was added to the widget.

As the Matrix-GUI may not be running, the application cannot connect to a running virtual frame buffer and the Qt embedded Linux applications must start one if this is the case. On the left side of the window, click on the **Projects** tab than select the **Run&Build** tab and finally the **Run** tab. In the **Arguments** field, "*-qws*" must be added.

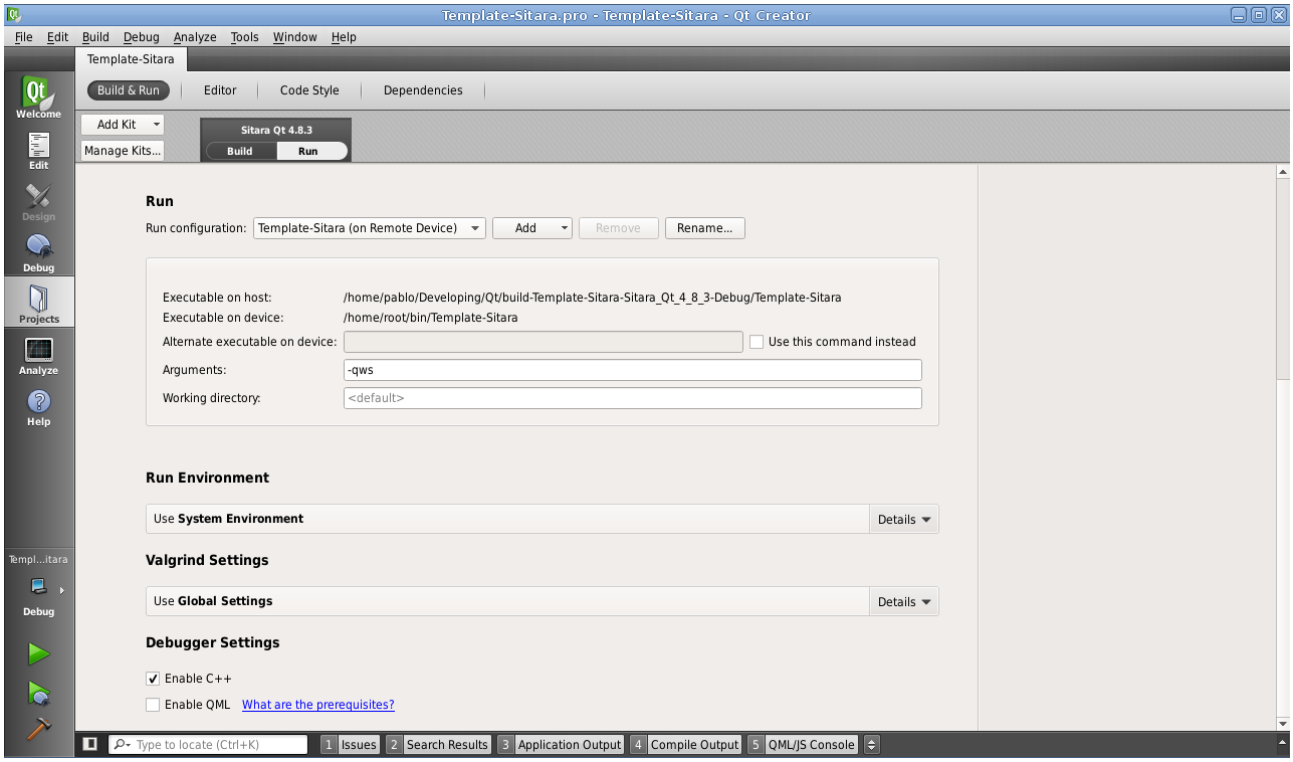


Figure 4.3-1: Application Arguments

Finally, the following lines should be added to the project file to Specify that the binaries should be copied over to the directory */usr/local/bin* on the target.

```
target.path += /usr/local/bin
INSTALLS += target
```

The template is now ready to be build, deployed and debugged.

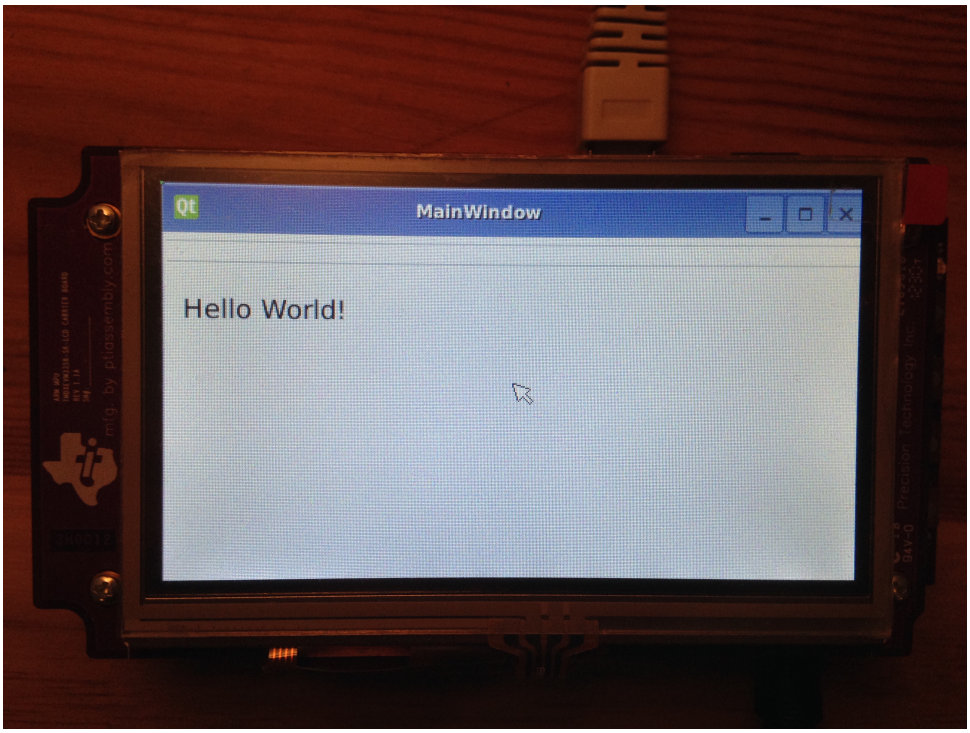


Figure 4.3-2: Test Application

5. Detailed Description of the Adopted Solution

5.1. Application

In this section, a detailed description of the adopted solution for the application to be developed shall be exposed.

5.1.1 External Relations

Following is a schematic diagram representing the input and output of data between the application and the different hardware, files and interfaces with which it must interact.

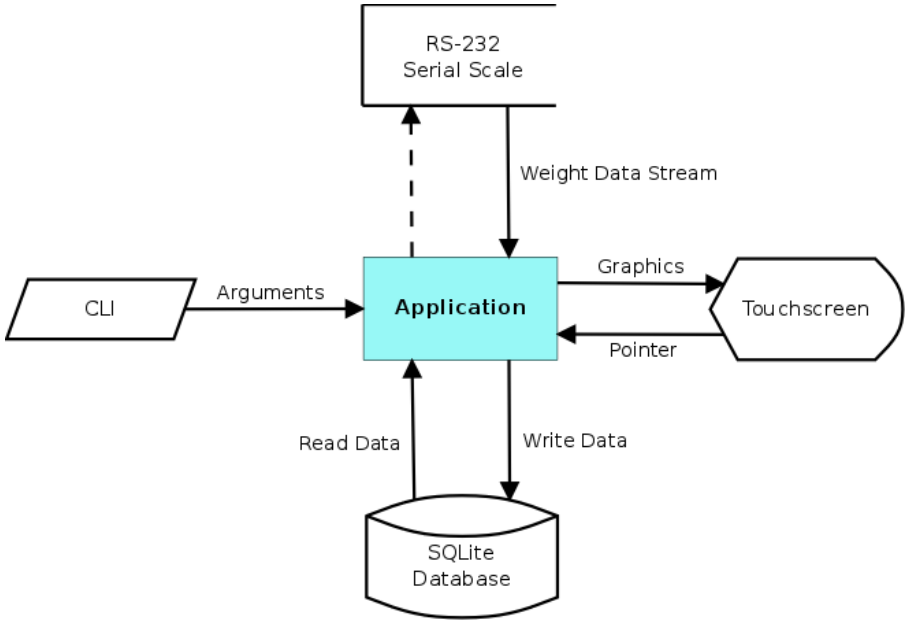


Figure 5.1-1: Application Interface

From the diagram above and with and general idea of how the application must interact with the different devices, files and interfaces, it is possible to determine that the following errors might be encountered during execution time:

- Argument error: failed to parse command line argument. An invalid command line argument was passed.
- Database error: failed to open/close database. The specified database could not be found.
- Query error: query failed to execute successfully. Query is malformed or database is corrupt.
- Serial error: serial connection could not be established. Specified serial port could not be opened.
- Scale error: scale is sending unexpected values. Scale Calibration needed.

It is possible to distinguish that “argument error” and “database error” are “malignant errors” and if encountered at run time, the application should close immediately and output an error message through standard error.

- Database error is a malignant error because the application should never start if the database file cannot be found, as no product information could be read or written. This would render the application useless as no transaction could be performed successfully.
- Argument error is a malignant error because the application should not start if an argument could not be interpreted. Otherwise, the user might be under the impression that the application is executing as specified through the command line interface, when in reality the invalid arguments are ignored.

On the other hand, “query error”, “serial error” and “scale error” are “benign errors” because if encountered during execution time, the error could be corrected without stopping the application. For these errors, a warning windows shall open here the user shall need to acknowledged the error before having the application can continue to execute.

If these errors are encountered, however, the application shall restrict the operations the user can perform with regard the affected modules or subsystems. For example: if USB-to-serial converter is not connected to the system, when the application attempts to open the port, the warning window shall be displayed, but even if the warning is acknowledged, the user shall not be allowed to register a sale.

Finally, to simplify the processing of these five errors, an enumerated data type and an a class shall be defined.

The enumerated data type, called errorCode, shall contain the five errors and a “success”. This data type shall be used as a return type to some functions to signal if they were executed without error or if an error occurred.

The benign error acknowledgement window shall be implemented in a class and the set-up function shall take as input an errorCode enumerated data type. Depending on the value of the input, different messages shall be displayed on-screen.

5.1.2 Application Structure

The structure of the application in terms of the classes that compose it is described in the diagram below. Note that the diagram only represents the classes that where created for this project and not the classes that C++ or any other third party entities provide.

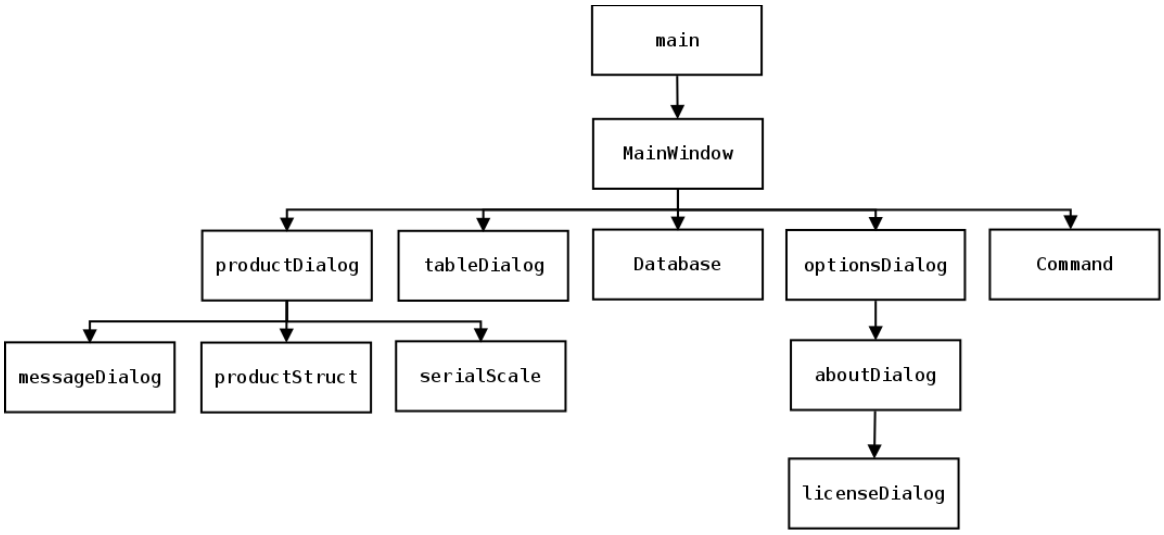


Figure 5.1-2: Application Structure

● **main**: The main function is not a class. It is the top level function whose purpose is to initiate application wide parameters, create and execute a MainWindow class and restart the application if MainWindow returns from execution a restart code. The main function starts the whole

● **MainWindow**: This class encapsulates the methods and variables necessary to display the main product selection window. MainWindows is main UI. It displays buttons with the icon, ID number and name of the products, in groups of 6, and the buttons necessary to navigate forward and backwards through the product list. It also provides buttons to execute the advanced options dialog and the accumulated sales dialog.

● **tableDialog**: This class encapsulates the methods and variables necessary to display the accumulated sales of all products in the database. The tableDialog class displays a scrollable table with all the products on the database and the current accumulated sales in the predefined mass unit (kg by default). From this class it is also possible to clear all the accumulated sales.

● **Database**: This class encapsulates the methods and variables necessary to manage the SQLite database. The Database class provides high-level, specialized functions that are needed to operate with the product database (open, close, read and write).

● **Command**: This class encapsulates the methods and variables necessary to parse the command line arguments that are supported. The Command class parses an array of command line arguments and updates parameters accordingly.

● **optionsDialog**: This class encapsulates the methods and variables necessary to display and execute configuration options. The options supported are: application restart, application close, system restart and system shutdown. The optionsDialog class also provides the button to execute the aboutDialog.

● **aboutDialog**: This class displays "about" information of the application. The information is static and defined in the UI file. The aboutDialog class also provides the button to execute the licenseDialog.

● **licenseDialog**: This class displays the license information of the application. The information is static and defined in the UI file.

● **productDialog**: This class encapsulates the methods and variables necessary to display product weight and information. Once selected the product, the data stream from the industrial scale is processed to read the weight and a product dialog is displayed with all the product information and the monetary value of the product being weighted.

● **messageDialog**: This class encapsulates the methods and variables necessary to display error and information messages. If a benign error is encountered, this dialog warns the user and asks for acknowledgement before continuing.

● **serialScale**: This class encapsulates the methods and variables necessary to manage the communication with the industrial scale. The serialScale class manages the serial communication between the industrial scale and the application. Communication is carried put over RS-232. The string processing carried out in this class is intended to work with the Microgram ie21 from Microgram Instruments Española, S.A. String processing functions must be modified if other hardware is to be used.

● **productStruct**: Product data structure. This data structure encapsulated variables to be used in the productDialog class.

5.1.3 Application Flowcharts

To better explain how the application functions, a flowchart and a screenshot for each dialog and the Main function is described in this section.

It should be noted that not all the source code shall be reflected in this section as there are modules such as database module, serial module or command module which provide important functions for the correct execution of the application but can not be described in terms of a flowchart by themselves. There are also header files such as errorcode.h and productstruct.h whose only purpose is to provide the implementation of an enumerated and structured data types respectively.

5.1.3.1 Main

The main function is composed solely of the C++ source code file main.cpp.

The flow diagram for this module is as follows:

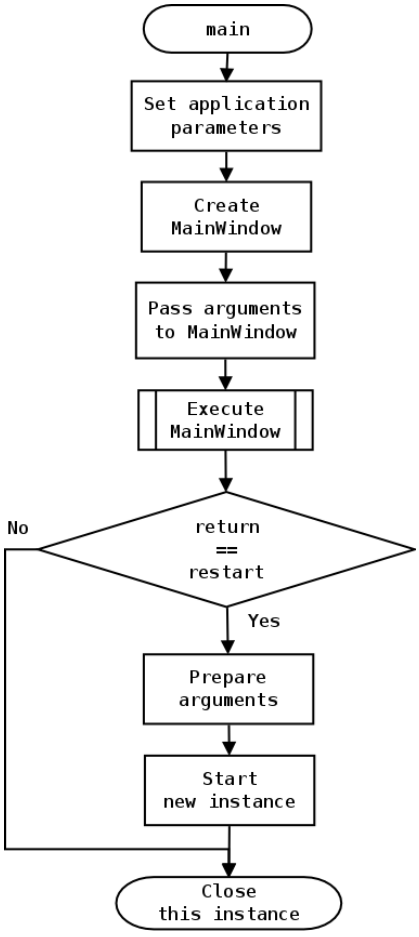


Figure 5.1-3: main Flowchart

5.1.3.2 Main Window

The MainWindow module is composed of the C++ source code file mainwindow.cpp, the C header file mainwindow.h and the Qt form mainwindow.ui.

The flow diagram for this module is as follows:

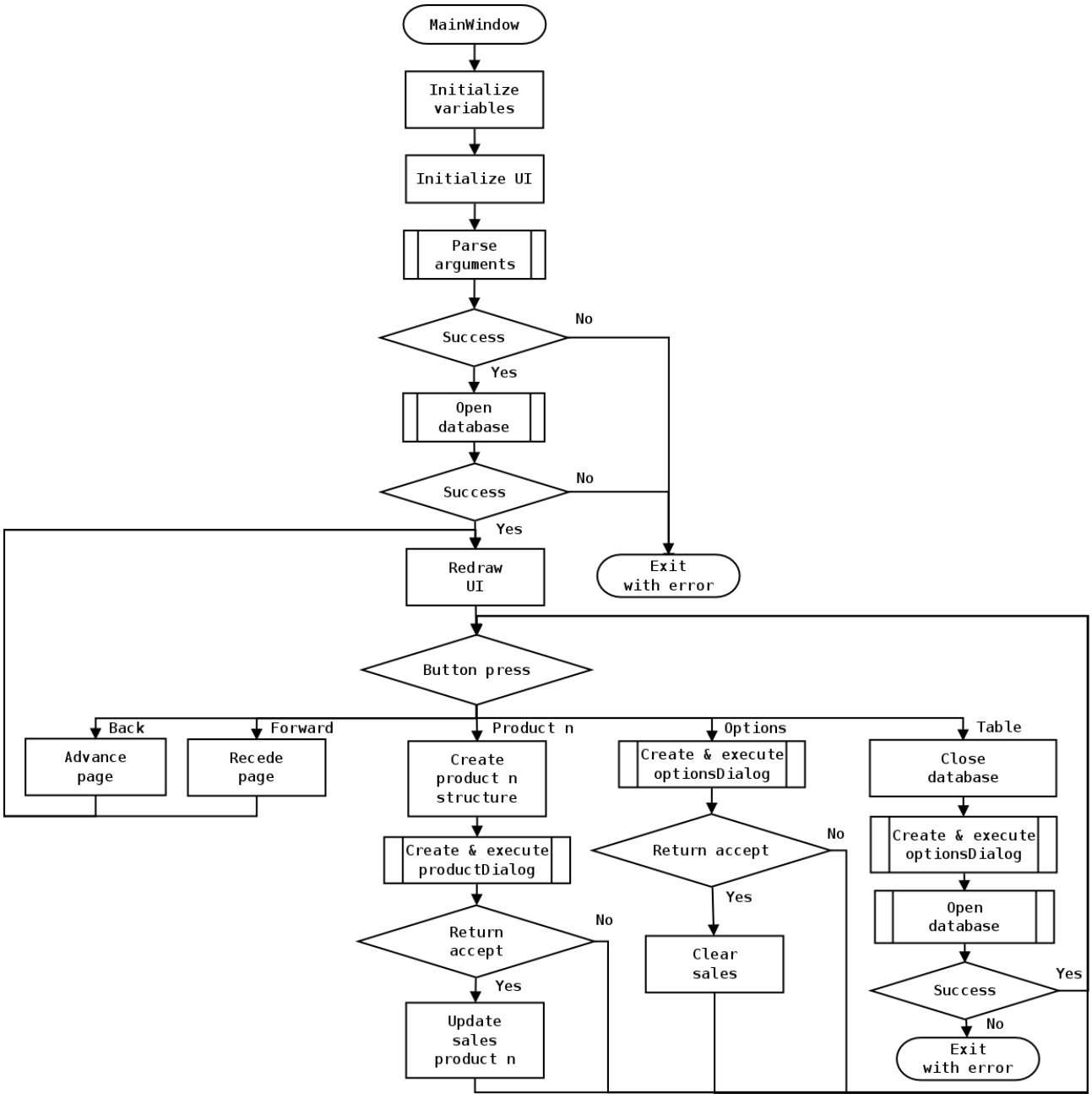


Figure 5.1-4: MainWindow Flowchart

The Qt form mainwindow.ui provides the GUI for this module, a screenshot of which is provided below.

Figure 5.1-5: MainWindow UI

5.1.3.3 Product Dialog

The productDialog module is composed of the C++ source code file productdialog.cpp, the C header file productdialog.h and the Qt form productdialog.ui.

The flow diagram for this module is as follows:

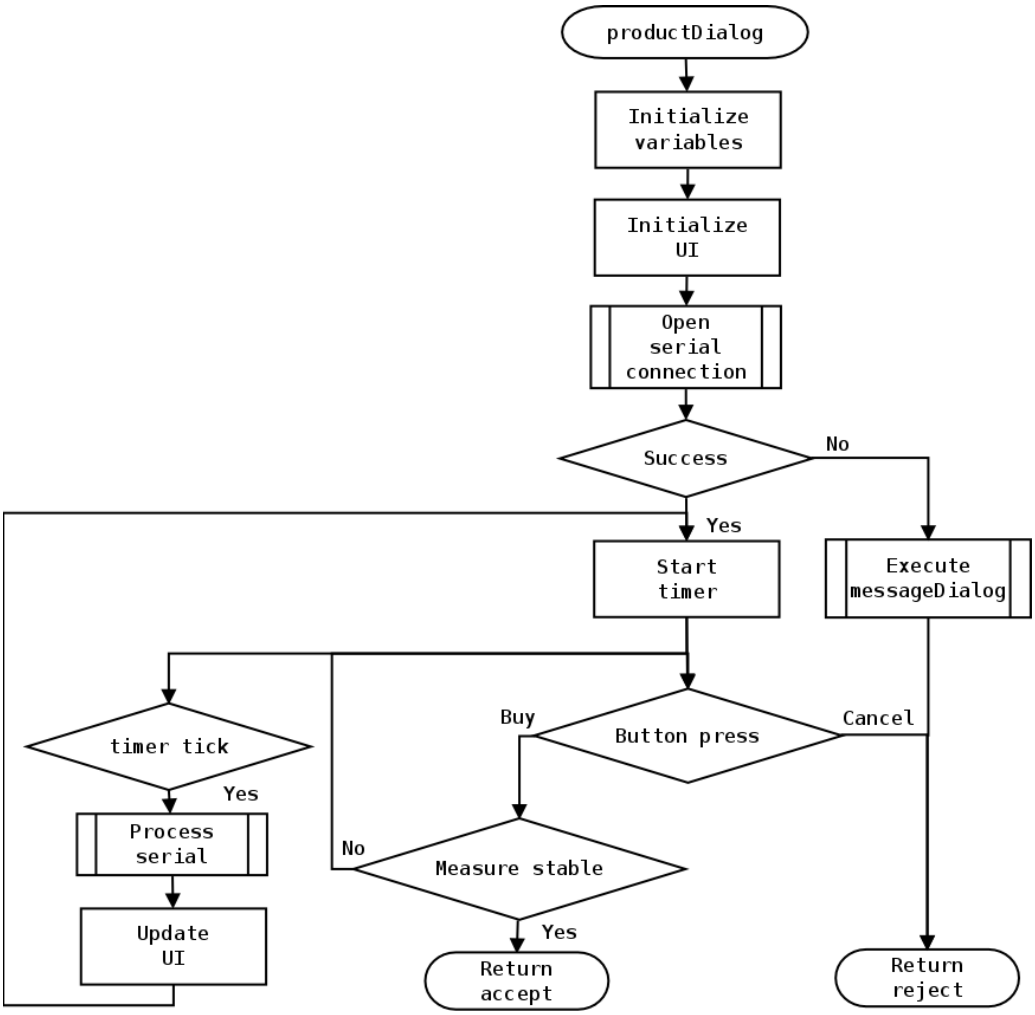


Figure 5.1-6: productDialog Flowchart

The Qt form productdialog.ui provides the GUI for this module, a screenshot of which is provided below.

Figure 5.1-7: productDialog UI

5.1.3.4 Message Dialog

The messageDialog module is composed of the C++ source code file messagedialog.cpp, the C header file messagedialog.h and the Qt form messagedialog.ui.

The flow diagram for this module is as follows:

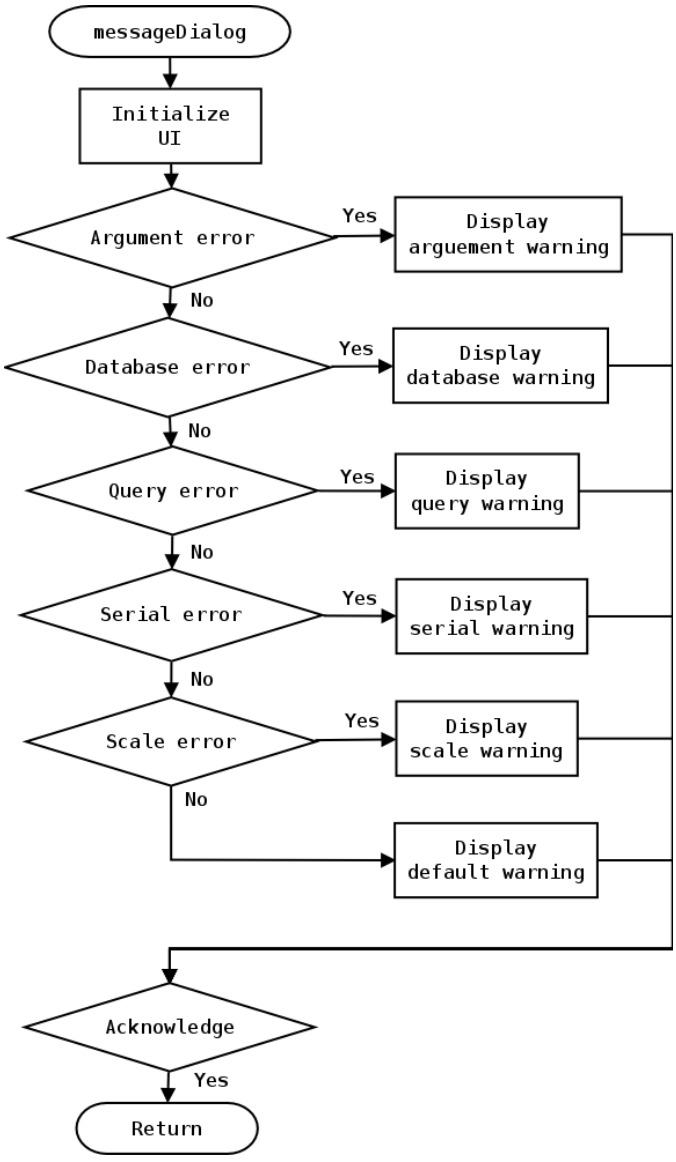


Figure 5.1-8: messageDialog Flowchart

The Qt form messagedialog.ui provides the GUI for this module, a screenshot of which is provided below. Note that this screenshot is warning of a serial communication error, but other errors are also reported.

Figure 5.1-9: messageDialog UI

5.1.3.5 Table Dialog

The tableDialog module is composed of the C++ source code file tabledialog.cpp, the C header file tabledialog.h and the Qt form tabledialog.ui.

The flow diagram for this module is as follows:

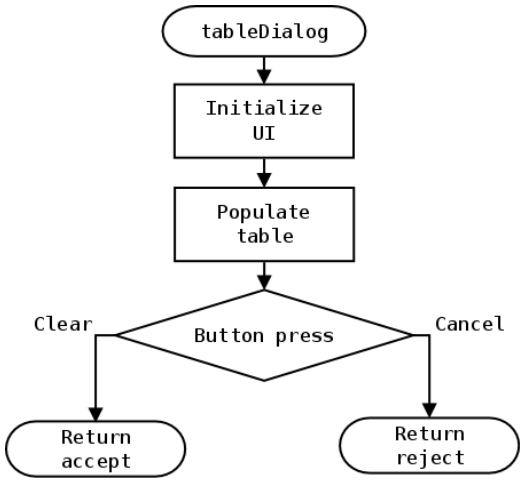


Figure 5.1-10: tableDialog Flowchart

The Qt form tabledialog.ui provides the GUI for this module, a screenshot of which is provided below.

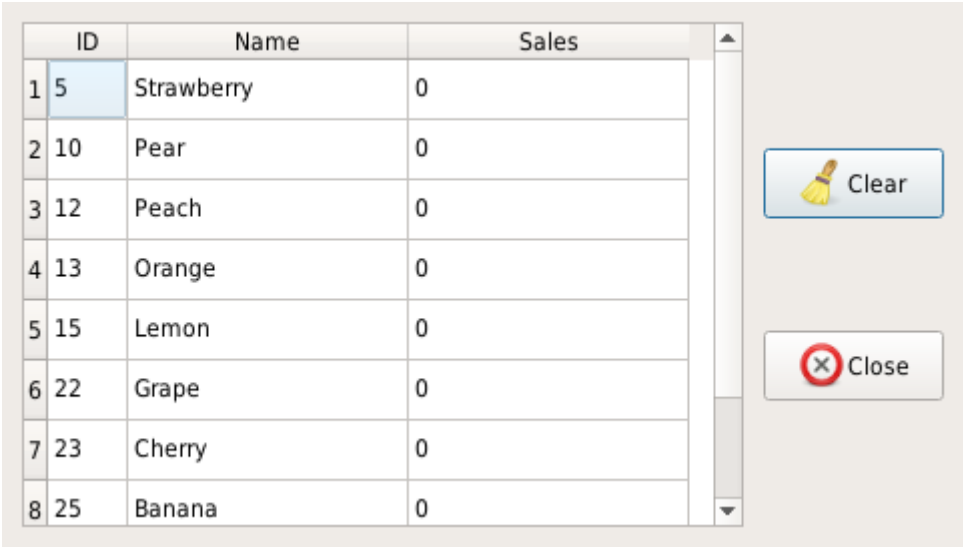


Figure 5.1-11: tableDialog UI

5.1.3.6 Options Dialog

The optionsDialog module is composed of the C++ source code file optionsdialog.cpp, the C header file optionsdialog.h and the Qt form optionsdialog.ui.

The flow diagram for this module is as follows:

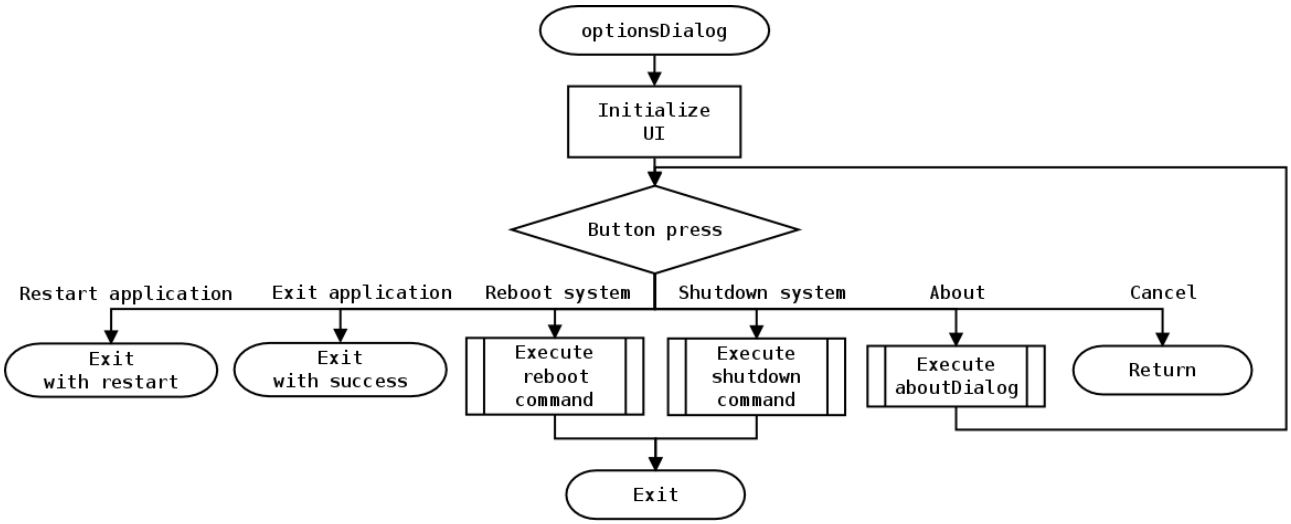


Figure 5.1-12: optionsDialog Flowchart

The Qt form optionsdialog.ui provides the GUI for this module, a screenshot of which is provided below.

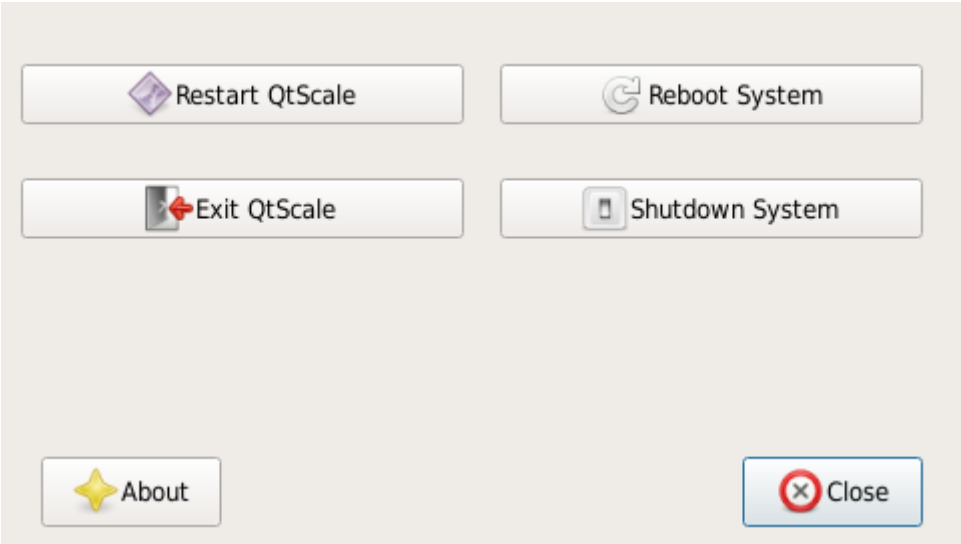


Figure 5.1-13: optionsDialog UI

5.1.3.7 About Dialog

The aboutDialog module is composed of the C++ source code file aboutdialog.cpp, the C header file aboutdialog.h and the Qt form aboutdialog.ui.

The flow diagram for this module is as follows:

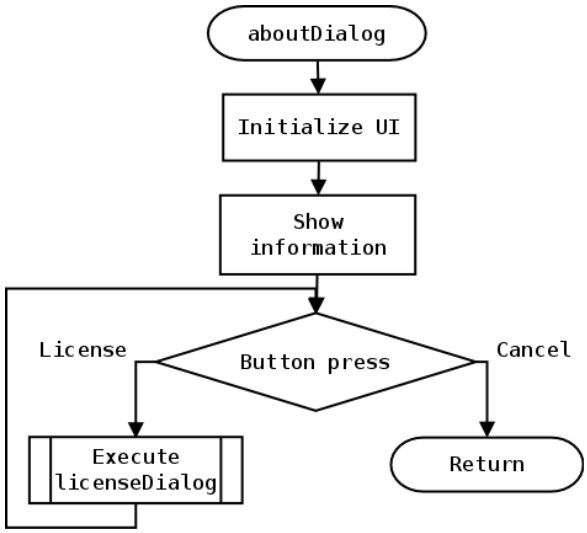


Figure 5.1-14: aboutDialog Flowchart

The Qt form aboutdialog.ui provides the GUI for this module, a screenshot of which is provided below.

Figure 5.1-15: aboutDialog UI

5.1.3.8 License Dialog

The licenseDialog module is composed of the C++ source code file licensedialog.cpp, the C header file licensedialog.h and the Qt form licensedialog.ui.

The flow diagram for this module is as follows:

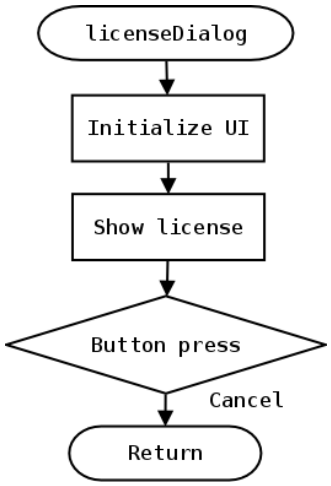


Figure 5.1-16: licenseDialog Flowchart

The Qt form licensedialog.ui provides the GUI for this module, a screenshot of which is provided below.

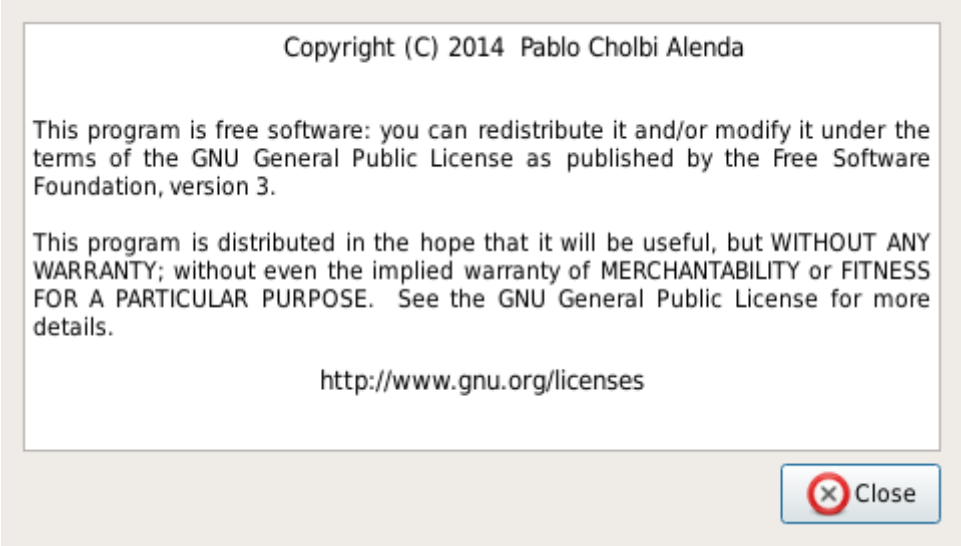


Figure 5.1-17: licenseDialog UI

5.1.4 Text Fonts for Internationalization

Taking into account the current level of globalization, which is expected to increase even more in the coming years, the application being developed should have internationalization support. As one of the goals of this project is to create a portable multi-platform (OS and architecture) application, internationalization can be seen as an extension of portability: it will give the application to be executed not only across a variety of operations systems and architectures, but also in a variety of languages and locales.

Qt has good support for application internationalization built-in, but in order to represent non-ASCII characters on-screen, the appropriate fonts must be present on the target system. The target system by default only comes with fonts for the English language.

This section details the process of adding fonts for the correct representation of alphabets and writing systems of languages other than the default for the target system. Specifically, Japanese fonts shall be added but the process can be extrapolated for other fonts.

First of all, it is necessary to download the TrueType files for the desired fonts. The latest Takao Japanese font family shall be downloaded, as it is a free and open source TrueType font as it is suitable for both display and printing and it is a popular font on the Debian and Ubuntu distributions.

```
user@debian:~$ cd Downloads/
user@debian:~/Downloads$ wget https://launchpad.net/takao-
  fonts/003.02/003.02.01/+download/takao-fonts-ttf-
  003.02.01.tar.gz
user@debian:~/Downloads$ tar -zxvf takao-fonts-ttf-
  003.02.01.tar.gz
user@debian:~/Downloads$ cd takao-fonts-ttf-003.02.01/
```

Now the fonts shall be transferred over to the target system.

```
user@debian:~/Downloads/takao-fonts-ttf-003.02.01$ scp *.ttf
  root@192.168.1.100:/usr/lib/fonts/
```

With this, the target system now has support for representing Japanese text.

Finally, some clean-up is required on the host computer.

```
user@debian:~/Downloads/takao-fonts-ttf-003.02.01$ cd ..
user@debian:~/Downloads$ rm -rf takao-fonts-ttf-003.02.01
user@debian:~/Downloads$ rm takao-fonts-ttf-003.02.01.tar.gz
```

Figure 5.1-18: Application Using Japanese Database

5.1.5 Application Start on Boot

One of the requirements is that the application must start on system boot or restart by default. The starting and stopping of programs and services on Unix and Unix-like systems is handled by the Init Scripts. [Debian-wiki] is taken as reference to create a Linux Standard Base (LSB) Init script to start the application being developed on all runlevels except shutdown, reboot and single-user mode. By making the script LSB compliant it makes it portable across many Unix and Unix-like distributions.

First, it is necessary to create the script on the target system.

```
root@am335x-evm:~# vi /etc/init.d/QtSerial-init.sh
```

The content of the script being:

```
#! /bin/sh
# /etc/init.d/QtScale-init.sh

### BEGIN INIT INFO
# Provides:          QtScale
# Required-Start:    $local_fs $syslog
# Required-Stop:     $local_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start QtScale at boot time.
# Description:       Start QtScale with "-qws" option
### END INIT INFO

case "$1" in
  start)
  echo "Starting QtScale"
    /usr/local/bin/QtScale -qws --path=/home/root/.QtScale/
    ;;
  stop)
```

```
    echo "Stopping QtScale"
    killall QtScale
    ;;
  *)
    echo "Usage: /etc/init.d/QtScale-init.sh {start|stop}"
    exit 1
    ;;
esac

exit 0
```

This would be a generic script. But as the application is being developed on the Texas Instruments TMDSSK3358 AM335x Starter Kit, further processing is required for the application to behave correctly: the touchscreen pointer needs to be configured. Taking as reference */etc/init.d/matrix-gui-2.0* Which is the Matrix-GUI application initialization script, the following changed are made to the basic script.

```
#! /bin/sh
# /etc/init.d/QtScale-init.sh

### BEGIN INIT INFO
# Provides:          QtScale
# Required-Start:    $local_fs $syslog
# Required-Stop:     $local_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start QtScale at boot time.
# Description:       Start QtScale with "-qws" option
### END INIT INFO

export TSLIB_TSDEVICE=/dev/input/touchscreen0
export QWS_MOUSE_PROTO=Auto
tsfile=/etc/pointercal

case "$1" in
  start)
    # ARM9 devices get a lot of alignment trap errors with the current
    # version of Qt (4.7.2) that we use.  The printing of these messages
    # is causing a severe slowdown with QtScale and other Qt applications
    # that QtScale launches.  The root cause is under investigation and an
    # issue is being filed in the Qt JIRA tracker.  For now using the
    # following command will do a software fixup of the alignment trap errors
    # in the kernel.  This should have no impact on cortex-A8 devices.
    echo 2 > /proc/cpu/alignment

    # Do not try to calibrate the touchscreen if it doesn't exist.
    if [ -e /dev/input/touchscreen0 ]
    then
        export QWS_MOUSE_PROTO=Tslib:/dev/input/touchscreen0
        # Check if the SD card is mounted and the first partition is
        # vfat.  If so let's write the pointercal file there so that if
        # someone messes up calibration they can just delete the file from
        # any system and reboot the board.
        mount | grep /media/mmcblk0p1 | grep vfat > /dev/null 2>&1
        if [ "$?" = "0" ]
        then
            tsfile=/media/mmcblk0p1/pointercal
            export TSLIB_CALIBFILE=$tsfile
```

```
        fi

        if [ ! -f $tsfile ] ; then
            echo -n "Calibrating touchscreen (first time only)"
            ts_calibrate
            echo "."
            # If we create a pointercal file and
            # it was not in /etc/pointercal
            # let's copy it there as well if it does not already exist.
            if [ ! -f /etc/pointercal -a -f $tsfile ]
            then
                cp $tsfile /etc/pointercal
            fi
        fi
    fi

    echo "Starting QtScale"
    /usr/local/bin/QtScale -qws --path=/home/root/.QtScale/
    ;;
  stop)
    echo "Stopping QtScale"
    killall QtScale
    ;;
  *)
    echo "Usage: /etc/init.d/QtScale-init.sh {start|stop}"
    exit 1
    ;;
esac

exit 0
```

After saving and exiting *vi*, the script must be made executable.

```
root@am335x-evm:~# chmod 755 /etc/init.d/QtScale-init.sh
```

To register the script, the following command is executed. Note that the “99” is to try to set-back the execution of the script as much as possible during boot.

```
root@am335x-evm:~# update-rc.d QtScale-init.sh defaults 99
```

Now the application will start automatically on boot. If for some reason it is necessary to stop the script from executing, the following command should be executed.

```
root@am335x-evm:~# update-rc.d -f QtScale-init.sh remove
```

5.2. SQLite Database

5.2.1 Database Design

The specifications demand for a database that has at least the following fields: image, name, index, price and accumulated sales. This information that the client has provided narrows down the adopted solution for the database design, but nevertheless, it is good practice to follow the standard design progress.

Determine the Purpose of the Database

The database is to be used as a local, single-file database for an embedded application when it is not expected to have more than one instance reading and/or writing in the database file. The database is intended to store product information and accumulated sales values.

Find and Organize the Information Required

As stated earlier, the data to be stored in the database has already been defined, but not the data type. In this section, each data point will be given a data type according to the nature of the data.

Data	Index	Name	Image	Price	Accumulated
Data type	Integer	Text	Blob	Real	Real

Divide the Information into Tables

At this point is is possible to extract that in this database, there is only one entity: “Product”. All that data that need to be included in the database are attributes of the “Product” entity (index, name, image, price and accumulated sales). Therefore, only one table is necessary in this database. It should be noted that this refers to the tables that arise from entity and entity relationships, more tables might be necessary when normalizing the database.

Turn Information Items into Columns

As there is only one entity and one table, all the attributed of the entity become columns in the table. In other words: the database has one table called “Products” with 5 columns named “Index”, “Name”, “Image”, “Price” and “Accumulated”.

Specify Primary Keys

Each table must have a primary key. The Product table's primary key is the index, as one might assume from the name. It is the primary key because the index is a unique integer that identified each product.

Set-up the Table Relationships

As there is only one table, there are no relationships between tables. The data stored in the Product table is unique and different for each product. This implies that the entity and all its attributes have one-to-one relationship. Please note that even though the price or accumulated sales of various entities can be the same at a given point, there is no point in creating a relationship from these attributes and separate tables (one-to-many relationship).

Refine the Design

No further refinement or simplifications can be made the database.

Apply the normalization rules

The designed database is so limited and simple that it does not require normalization as every cell contains a single value (atomic), every non-key column is fully dependent on the primary key and non-key columns are of each other. Another way to see this is that the database is so simple that it is already normalized to third form normalization (at least). The database is both OLTP (transactional) and OLAP (analytical) at the same time, or neither, depending how it is viewed.

In any case, the designed database can not be refined, simplified or normalized any further with the current specifications, and therefore it is taken as the adopted solution.

Products	
Index	int
Name	text
Image	blob
Price	float
Accumulated	float

Figure 5.2-1: Database Schema

5.2.2 Database Creation

This section describes the process of creating the previously designed database. As an example, a database that contains 9 fruits (products) shall be created with the following data.

Index	Name	Image	Price	Accumulated
5	Strawberry	(PNG blob)	2.50	0.0
10	Pear	(PNG blob)	1.30	0.0
12	Peach	(PNG blob)	1.00	0.0
13	Orange	(PNG blob)	0.70	0.0
15	Lemon	(PNG blob)	1.75	0.0

22	Grape	(PNG blob)	3.90	0.0
23	Cherry	(PNG blob)	3.70	0.0
25	Banana	(PNG blob)	1.40	0.0
37	Apple	(PNG blob)	1.60	0.0

The first step is to create the database file with the Product table that has been previously designed. The database shall be named QtScale.sqlite.

```
user@debian:~$ sqlite3 QtScale.sqlite "CREATE TABLE Products(ID
  integer primary key,Name text,Icon blob,Price real,Accumulated
  real);"
```

Now it is possible to insert the data. One command is required for every product record, therefore 9 commands must be executed. The images are inserted as a blob by doing a hexadecimal dump. This method has been proven to work with the Qt SQL module as images can be retrieved correctly.

```
user@debian:~$ echo "INSERT INTO
  Products(ID,Name,Icon,Price,Accumulated)
  values(5,'Strawberry',X'`hexdump -ve '1/1 "%.2x"'
  Icons/Strawberry.png`',2.50,0.0);" | sqlite3 QtScale.sqlite

user@debian:~$ echo "INSERT INTO
  Products(ID,Name,Icon,Price,Accumulated)
  values(10,'Pear',X'`hexdump -ve '1/1 "%.2x"'
  Icons/Pear.png`',1.30,0.0);" | sqlite3 QtScale.sqlite

user@debian:~$ echo "INSERT INTO
  Products(ID,Name,Icon,Price,Accumulated)
  values(12,'Peach',X'`hexdump -ve '1/1 "%.2x"'
  Icons/Peach.png`',1.05,0.0);" | sqlite3 QtScale.sqlite

user@debian:~$ echo "INSERT INTO
  Products(ID,Name,Icon,Price,Accumulated)
  values(13,'Orange',X'`hexdump -ve '1/1 "%.2x"'
  Icons/Orange.png`',0.70,0.0);" | sqlite3 QtScale.sqlite

user@debian:~$ echo "INSERT INTO
  Products(ID,Name,Icon,Price,Accumulated)
  values(15,'Lemon',X'`hexdump -ve '1/1 "%.2x"'
  Icons/Lemon.png`',1.75,0.0);" | sqlite3 QtScale.sqlite

user@debian:~$ echo "INSERT INTO
  Products(ID,Name,Icon,Price,Accumulated)
  values(22,'Grape',X'`hexdump -ve '1/1 "%.2x"'
  Icons/Grape.png`',3.95,0.0);" | sqlite3 QtScale.sqlite

user@debian:~$ echo "INSERT INTO
  Products(ID,Name,Icon,Price,Accumulated)
```

```
  values(23,'Cherry',X'`hexdump -ve '1/1 "%.2x"'
  Icons/Cherry.png`',3.70,0.0);" | sqlite3 QtScale.sqlite

user@debian:~$ echo "INSERT INTO
  Products(ID,Name,Icon,Price,Accumulated)
  values(25,'Banana',X'`hexdump -ve '1/1 "%.2x"'
  Icons/Banana.png`',1.40,0.0);" | sqlite3 QtScale.sqlite

user@debian:~$ echo "INSERT INTO
  Products(ID,Name,Icon,Price,Accumulated)
  values(37,'Apple',X'`hexdump -ve '1/1 "%.2x"'
  Icons/Apple.png`',1.60,0.0);" | sqlite3 QtScale.sqlite
```

With this, the database has been created and all the data has been inserted. This database would be a valid database ready to be used by the application to be developed.

Please note that this method assumes that there is a subdirectory in the current working directory called “Icons” that contains a PNG image for every product that has the same name as the product it represents followed by a “.png” extension. For a clearer explanation, the content of the subdirectory for this example is as follows.

```
user@debian:~$ ls Icons/

  Apple.png  Banana.png  Cherry.png  Grape.png  Lemon.png
  Orange.png  Peach.png  Pear.png  Strawberry.png
```

5.2.3 Database Management

For the management of the database, various options are available but for simplicity of use a GUI application is recommended. A popular standalone, open source, cross-platform GUI application for this purpose is “SQLite Database Browser”. SQLite Database Browser can be obtained from <http://sourceforge.net/projects/sqlitebrowser/>. However, there are many other applications[1] that will perform the required task successfully.

This application is intuitive and easy to use even for novice users and it allows to browse and modify databases without SQL queries. Although it does support SQL query input for the users who prefer it.

Please note that this section refers to the management and updating of existing database records. New records are recommended to be inserted with the aforementioned method as it is the only one where the image insertion has been tested to work successfully.

1 Another popular application to manage SQLite databases come in the form of a Firefox add-on with very similar functionality and UI as the aforementioned application. The add-on can be obtained from <https://addons.mozilla.org/en-US/firefox/addon/sqlite-manager/>

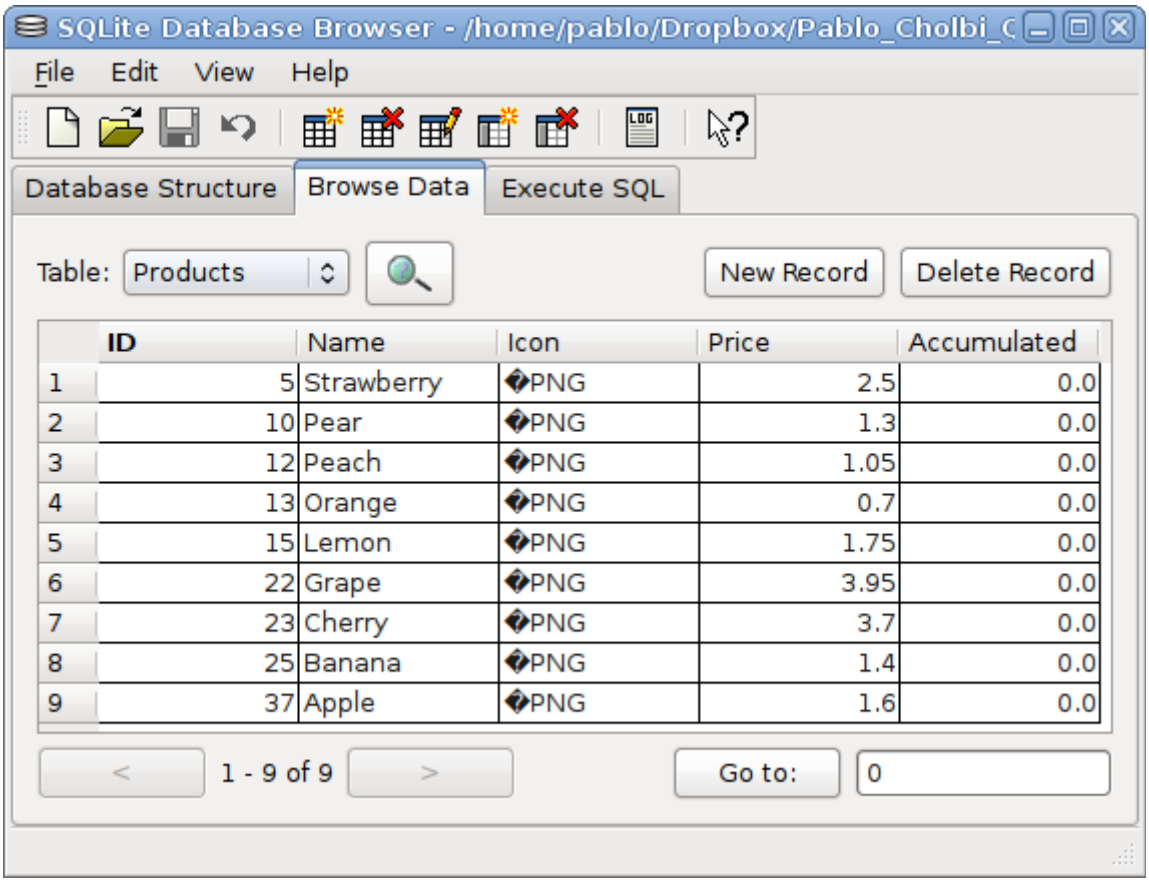


Figure 5.2-2: SQLite Database Browser

5.3. QtSerialPort

QtSerialPort was introduced as an internal module in Qt 5 as the substitute of QextSerialPort. QtSerialPort is now the default module in Qt to handle serial connections and it is being developed by the same team that worked on QextSerialPort. The QextSerialPort project has been abandoned.

In this project, Qt 4.8 libraries are being used as this is what is supplied in the Texas instruments SDK but it is worth noting that the QtSerialPort source code supports Qt 4, which means that the module can be configured, compiled and used in Qt 4.

QtSerialPort shall be used in this project as it is officially supported by Qt, no external third party library is required, it is more feature rich than QextSerialPort and, most importantly, because it eases cross-platform compatibility and compatibility with current and future Qt releases.

This section details the process of adding the QtSerialPort module to the current set-up such that it can be used in the application being developed. The detailed process is based on the instructions given in [Qt-Wiki].

First, it is necessary to download the source code in the TI SDK directory and set up the build environment.

```
user@debian:~$ source /opt/ti-sdk-am335x-evm-06.00.00.00/linux-
  devkit/environment-setup
[linux-devkit]:~> cd /opt/ti-sdk-am335x-evm-06.00.00.00/
```

```
[linux-devkit]:/opt/ti-sdk-am335x-evm-06.00.00.00> git clone
  git://gitorious.org/qt/qtserialport.git
[linux-devkit]:/opt/ti-sdk-am335x-evm-06.00.00.00> mv
  qtserialport/ qtserialport-src/
[linux-devkit]:/opt/ti-sdk-am335x-evm-06.00.00.00> mkdir
  qtserialport-build
[linux-devkit]:/opt/ti-sdk-am335x-evm-06.00.00.00> cd
  qtserialport-build/
```

Now it is possible to proceed and compile and install the shared library.

```
[linux-devkit]:/opt/ti-sdk-am335x-evm-06.00.00.00/qtserialport-
  build> qmake ../qtserialport-src/qtserialport.pro
[linux-devkit]:/opt/ti-sdk-am335x-evm-06.00.00.00/qtserialport-
  build> make
[linux-devkit]:/opt/ti-sdk-am335x-evm-06.00.00.00/qtserialport-
  build> make install
```

Now the library is present within the TI SDK but it has to be transferred to the target system.

```
[linux-devkit]:/opt/ti-sdk-am335x-evm-06.00.00.00/qtserialport-
  build> scp src/serialport/libQtSerialPortE.so.1.0.0
  root@192.168.1.100:/usr/lib/
```

Finally, some symbolic links must be created on the target system so that applications can resolve the dynamic library dependency correctly.

```
root@am335x-evm:~# ln -s /usr/lib/libQtSerialPortE.so.1.0.0
  /usr/lib/libQtSerialPortE.so
root@am335x-evm:~# ln -s /usr/lib/libQtSerialPortE.so.1.0.0
  /usr/lib/libQtSerialPortE.so.1
root@am335x-evm:~# ln -s /usr/lib/libQtSerialPortE.so.1.0.0
  /usr/lib/libQtSerialPortE.so.1.0
```

Now it is possible to compile binaries that depend on QtSerialPort on the host computer and executed on the target system.

To use QtSerialPort in a Qt 4 project, the following lines must be added to the project file.

```
CONFIG += serialport
```

To use QtSerialPort in a Qt 5 project, the following lines must be added to the project file.

```
QT += serialport
```

And the following lines must be added to the source code files that use functions, structures or classes from QtSerialPort.

```
#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QserialPortInfo>
```

6. Conclusions and Future Work

Taking into account that in recent years there has been an increase in the number of devices that use an ARM Cortex-A processor, because they are a cost-effective and power efficient solution for applications where the task to be performed is not CPU intensive and that their performance continues to improve year by year. It is possible to conclude that an increase in the number of industrial computers running on ARM processors with embedded Linux will be seen in the coming years due to its advantages for low power applications.

This project highlights that with the Qt C++ framework it is possible to easily develop cross-platform applications in C++ for embedded Linux that can be ported to industrial computers running on more traditional x86 processors. This fulfils part of the project requirements, which were to have this report serve as reference as a general reference for developing industrial applications for the TI AM335x processor family running embedded Linux. Having a wider range of hardware options is always an advantage.

In the developing of the application for the industrial scale, a greater understanding of SQL databases and industrial communications has been attained as well as the deepening of the knowledge of embedded Linux systems. The project, as a whole, has also served as valuable experience in the development of industrial projects in the fields of industrial computer science and industrial electronics.

Overall, it is possible to conclude that all the requirements and objectives set at the start of the project have been fulfilled and valuable academic and professional experience has been acquired in the process of developing

Going beyond the specification of the project, 3 upgrades are proposed as future work that would improve the flexibility, functionality and robustness of the application:

- **Add printing functionality** to the system such that when a sale is complete, a ticket is printed with information such as: the name of the product, the weight of the product, the total price, the date and time of the transaction and a bar code (and or QR code) to be later scanned by the cashier. This upgrade provides greater functionality to the project but it would be challenging to configure, compile and set up CUPS on the target system. Making this feature cross-platform might also be an issue.
- **Upgrade to the database** (and the application) and add an attribute to the product entity which would indicate if the product is in stock. This could simplify the database entertainment as the product record would not have to be removed when the product is out of stock to then be reinserted when it is back in stock. This would translate to less maintenance required to be done through the command line.
- **User defined serial port** during execution time instead of having it defined at compile time. The user should be able to select which serial port to use through a command line interface or an option window. In practice, the solution that would make more sense it to have only a command line interface for this function as it is an advanced setting that the average user should not need to be concerned with. A command line argument option, similar to the one implemented to specify the database location, could be implemented to set the serial port at the application start. Much like the database location option, this serial port option would have a default value (*/dev/ttyUSB0*) in case the user does not explicitly pass the argument.

7. Bibliography

[Abdurachmanov 2014] D. Abdurachmanov, K. Arya, J. Bendavid, T. Boccali, G. Cooperman, A. Dotti, P. Elmer, G. Eulisse, F. Giacomini, C. Jones, M. Manzali and S. Muzaffar: "Explorations of the Viability of ARM and Xeon Phi for Physics Processing" Arxiv 2014.

[Allen 2010] G. Allen, M. Owens: "The Definitive Guide to SQLite" Apress 2010.

[Bartholomew 2013] D. Bartholomew: "MariaDB vs. MySQL" ADMIN Magazine 2013, Retrieved from http://www.admin-magazine.com/Articles/MariaDB-vs.-MySQL

[Blem 2013] E. Blem, J. Menon and K. Sankaralingam: "A Detailed Analysis of Contemporary ARM and x86 Architectures" University of Wisconsin 2013.

[Debian-Wiki] Debian Wiki: "LSBInitScripts", Retrieved from https://wiki.debian.org/LSBInitScripts (Last edit: 2014-02-07) (as of 2014-04-11, 17:41 GMT).

[Ou 2012] Z. Ou, B. Pang, Y. Deng, J. Nurminen and A. Ylä-jääski: "Energy- and Cost-Efficiency Analysis of ARM-Based Clusters" IEEE Computer Society 2012.

[Qt-Wiki] Qt Project Wiki: "QtSerialPort", Retrieved from http://qt-project.org/wiki/QtSerialPort (Last edit: 2014-01-17) (as of 2014-04-11, 12:09 GMT).

[Roberts-Hoffman 2009] K. Roberts-Hoffman and P. Hegde: "ARM Cortex-A8 vs. Intel Atom: Architectural and Benchmark Comparisons" University of Texas at Dallas 2009.

[TI-Wiki-1] Texas Instruments Wiki: "Sitara Linux Training: Hands on with QT", Retrieved from http://processors.wiki.ti.com/index.php/Sitara_Linux_Training:_Hands_on_with_QT (as of 2014-03-10, 10:21 GMT).

[TI-Wiki-2] Texas Instruments Wiki: "AMSDK Linux User's Guide", Retrieved from http://processors.wiki.ti.com/index.php/AMSDK_Linux_User's_Guide (as of 2014-03-10, 14:50 GMT).

[Vidal 2012] R. Vidal Aroca and L. Garcia Gonçalves: "Towards Green Data Centers: A Comparison of x86 and ARM Architectures Power Efficiency" Academic Press 2012.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Escuela Técnica Superior de Ingeniería del Diseño

Bachelor's Degree in Industrial Electronics and Automation Engineering

Final Year Engineering Project

QT SET-UP FOR TEXAS INSTRUMENTS AM335X ARM CORTEX-A8 USING LINUX: APPLICATION FOR AN INDUSTRIAL SCALE

2 - Specifications

Author:
Pablo Cholbi Alenda

Supervisor:
Dr. Àngel Perles Ivars

May 2014

Table of Contents

1. Optional Conditions

1.1. Programming Work

The developer agrees to complete the project as outlined in the report, and deliver on the date agreed with the customer, previously reviewed.

1.2 Poor Performance and Modifications

The customer has the responsibility to review the operation of the supplied package and communicate the aspects that, according to him/her, are not well designed, extending such liability to those existing but undetected defects.

The changes that the client sees fit will be performed in any case with a corresponding increase in price of the project as they are deemed unnecessary changes for the normal operation of the system and made for the client's particular preferences. Expanding on this last point, the developer agrees to alter the content of the program provided that the changes do not impair the performance of the application. In any case, the rates to be applied for of amendments in the project, shall be the same as the rest of the project, following the standards published in the Spanish Official Bulletin of the State (B.O.E.). In the unlikely event that the detected anomalies are due to faulty programming, the developer agrees to make all changes it deemed appropriate to remedy the issue, without this originating any cost to the client.

The client may also choose to have a third party modify the software package. This is permitted as long as the modifications are done in accordance with the software license (GNU GPL V3).

1.3 Final Delivery of the Project

The developer shall complete, test and deliver the project by the date previously negotiated with the client.

2. Economical Conditions

2.1. Pricing Structure

The prices charged for the project are in line with the general provisions agreed with respect the economic activity under focus. Thus, the rates used for labour force correspond to an Industrial Engineer performing programming tasks.

2.2. Improvements

Extensions or improvements that the customer wishes to make to the program will suppose an increase in the total cost of the project, in accordance with the amount of time needed and the rates previously stipulated.

2.3 Price Review

The time that can elapse between the writing of the contract of this project and its delivery and acceptance by the customer may be significant. If the elapsed time is considered in the tables provided by the Association of Inspectors and Industrial Engineers of Valencia (COPITI),a revision of the project cost shall be performed at the delivery of the project. The formulas and tables used for that purpose shall be extracted from the Spanish Official Bulletin of the State (B.O.E.) of the relevant year.

3. Functional Conditions

3.1. Initial Functional Specifications

- Touchscreen graphical user interface application for an industrial scale.
- Texas Instruments Sitara AM335x processor running Linux.
- System shall receive the output of a Microgram IE21 industrial scale from Microgram Instruments Española S.A. industrial scale over RS-232.
- Simple and intuitive software, such that the users are able to deduce how to use the application without any prior knowledge or help.
- The scale is intended to weigh fruit and vegetables, but the database should be generic such that it can be used for commerce of other goods by weight. The image, name, index, price and accumulated sales shall be stored in a relational database.
- The Application must automatically update the GUI when a product is added, removed or updated in the database.
- The system shall be network connected and have a fixed IP address assigned for easy maintenance and database access.
- Critical errors and non-critical errors should be handled differently. The application should exit immediately when a critical error is encountered. A warning system is allowed for non-critical errors.
- The application shall launch on system boot.
- The application shall have a system shutdown and system restart under the advanced options menu.
- The application shall be as modular as possible for simplicity and source code reuse.
- A low-cost and low-power solution should be favoured as the adopted solution.
- The application shall be cross-platform, developed under a framework which allows for the code to be easily compiled to run on different processor architectures and operational system.

3.2. Software Limitations

For the compilation of the source code Qt libraries 4 or greater are required. The building of the project depends on the availability of a cross-compiler for the target architecture and the Qt libraries for the target system.

3.3. Hardware Limitations

The project being developed is intended to be cross-platform. A test shall be made to test at least compatibility with x86 and x86-64 based Linux systems, as well as Texas Instruments AM335x ARM Cortex-A, but the software is only guaranteed to work on the Texas Instruments AM335x processor family.

4. Software Package Conditions

4.1. Conditions of Package Content

The package consists of a CD, which contains the project files, documentation and the source code of the application. The source code is written in C++ and a Qt C++ environment is necessary to compile it.

4.2. Conditions of Package Maintenance

The software package does not requires a review by a technician, except for changes in the configuration of the hardware and/or software. In accordance with the license under which the software is distributed, with the appropriate citation, a technician appointed by the client may modify the source code. The client is responsible for all the cost derived from the modification of the software.

4.3. Conditions of Maintenance of the Optical Support

The conditions detailed here are usual guidelines for archival of optical media:

- The CD shall be kept away form strong magnetic fields.
- The CD shall not be exposed to temperatures outside the range between 10ºC and 60ºC.
- The bottom of the CD shall not be exposed to contact with the user's hands or exposed to surfaces other than the CD reader, to prevent accidental data loss.

5. Conditions of Warranty

There is no warranty for the program, to the extent permitted by applicable law except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. the entire risk as to the quality and performance of the program is with you. should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

However, within two years from the delivery of the project, if the original project files are lost or damaged, the developer shall provide a copy of the project on optical media. The shipping costs of the new CD shall be billed to the client and therefore, the developer shall negotiate the shipping method and cost with the client a forehand.

The developer is not responsible in any circumstances for any damages that may arise from the improper use of the software package, both material damage and damage to third parties.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Escuela Técnica Superior de Ingeniería del Diseño

Bachelor's Degree in Industrial Electronics and Automation Engineering

Final Year Engineering Project

QT SET-UP FOR TEXAS INSTRUMENTS AM335X ARM CORTEX-A8 USING LINUX: APPLICATION FOR AN INDUSTRIAL SCALE

3 - Budget

Author:
Pablo Cholbi Alenda

Supervisor:
Dr. Àngel Perles Ivars

May 2014

Table of Contents

Factors to Consider

The calculations for the budget of this project haven been performed under the assumption that the designer is an independent engineer.

However, the number of hours devoted to the completion of a project of this nature and the price paid per hour of labour is different if the project was carried out by an engineering student or an experienced professional engineer. The number of hours invested by a student will be higher than the number of hours invested by a professional engineer, but the price per hour of labour is lower for a student than that of a professional engineer. The calculation of time was made assuming that the designer is a professional engineer with experience in the sector, reducing the time spent on design and construction, but charging a superior price per hour rate.

As this is a final year engineering project, neither VAT nor profit have been considered in the final cost of the project.

1. Material Cost

First, the total cost of the material used in the project shall be calculated. In the elaboration of this section, the following assumptions were made:

- The end user has a host computer (Windows, Mac OS X or Linux) with which to administrate the target system and the database.
- The end user has a Microgram IE21 industrial scale from Microgram Instruments Española S.A.
- The end user has an an Ethernet network with or without Internet access.

As stated in the report, when studying the alternative hardware solutions, the BeagleBone Black was found to be the best option for this project. However, this project has been developed on a TI AM335x Starter Kit because it was available and it has the same processor. The TI AM335x Starter Kit is more feature rich than the BeagleBone Black, but it also more expensive.

This project does not utilize any of these “additional” features, and it is perfectly implementable on the BeagleBone Black. Therefore, for the calculation of the final cost of the project, shall take into account the cost of developing the project with a BeagleBone Black. However, the cost of the TI AM335x Starter Kit is also provided below as additional information or as an optional upgrade for future expansion.

Concept	Units	Cost (€)	Quantity	Total
BeagleBoard BeagleBone Black	ea.	35	1	35
Beaglebone Black LCD Cape 4.3"	ea.	45	1	45
		Total Cost		80.00

Concept	Units	Cost (€)	Quantity	Total
AM335x Starter Kit	ea.	150	1	150
		Total Cost		150.00

Total cost of materials = 80.00€

2. Labour Cost

2.1. Bare Labour Cost

As stated above, these calculations were made under the assumption that the designer is an independent engineer with experience. The hourly rate has been set at 35 €/hour.

Concept	Units	Cost (€)	Quantity	Total
Viability study	h	35	10	350
Software design	h	35	25	875
Software implementation	h	35	80	2800
Testing	h	35	15	525
Documentation	h	35	40	1400
		Total Cost		5950.00

The total cost of this section for this project is **5950 €**.

The total amount of hours is 170 h. Assuming an 8 h workday and 22 workdays a month, 170 h constitutes:

$$\frac{170\ h}{8\ h \cdot 22\ days} = 0.966\ months$$

2.2. Costs Derived From Self-Employed Work Under Spanish Legislation

The following characteristics were considered for the self-employed engineer:

Age	22
Months of Contributions	12
Spanish National Classification of Economic Activities	6201 - Computer Programming Activities
Contingency Basis	1400 €/month

Concept	Units	Cost (€)	Quantity	Total
Tax Contribution	% / month	1400	26.5	371
Coverage for accidents at work, occupational disease and cessation of activity	month	41.3	1	41.3
		subtotal Cost		412.30
		Total Cost		398.28

The cost derived from self-employed work under Spanish legislation with the aforementioned characteristics is 412.30 €/month. This project constitutes 0.966 months of work and therefore, the total cost of this section for this project is **398.28 €**.

2.3. Total Cost of Labour

Concept	Cost (€)
Bare Cost of Labour	5950
Cost derived from self-employment	398.28
Total Cost	6348.28

Total cost of labour = 6348.28€

3. General Expenses

As discussed above, the budget of the project is carried out assuming that the designer is an independent engineer. This entails some additional cost that must be taken into account for the final calculation of the cost of this project.

Concept	Units	Cost (€)	Quantity	Total
Office/Laboratory Rental (25m²)	month	350	0.966	338.10
Electrical supply	month	45	0.966	43.47
Water supply	month	10	0.966	9.66
Municipal Garbage Recollection	month	10	0.966	9.66
Phone Line and Internet Access	month	61.25	0.966	59.17
		Total Cost		460.06

Total cost of general expenses = 460.06€

4. Amortization

4.1. General Amortization

Concept	Period (years)	Cost (€)	Quantity (months)	Total
Furniture	10	2200	0.966	17.71
Banking Fees	1	80	0.966	77.28
		Total Cost		94.99

The total cost of this section for this project is **94.99 €**.

4.2. Hardware Amortization

Concept	Period (years)	Cost (€)	Quantity (months)	Total
Asus A55VD-SX049V Laptop	5	579	0.966	9.32
Logitech Mouse	2	15.99	0.966	0.64
Asus RT-N66U Router	5	99	0.966	1.59
		Total Cost		11.56

The total cost of this section for this project is **11.56 €**.

4.3. Software Amortization

Open source software was used throughout the whole project and, therefore, no software amortization is needed..

4.3. Total Cost of Amortization

Concept	Cost (€)
General Amortization	94.99
Hardware Amortization	11.56
Total Cost	106.55

Total cost of amortization = 106.55€

5. Total Cost

Concept	Cost (€)
Material Cost	80.00
Labour Cost	6348.28
General Expenses	460.06
Amortization	106.55
Total Cost	**6994.89**

The global cost of the project is **six thousand nine hundred ninety-four euro and eighty-nine cent**.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Escuela Técnica Superior de Ingeniería del Diseño

Bachelor's Degree in Industrial Electronics and Automation Engineering

Final Year Engineering Project

QT SET-UP FOR TEXAS INSTRUMENTS AM335X ARM CORTEX-A8 USING LINUX: APPLICATION FOR AN INDUSTRIAL SCALE

4 – User Manual

Author:
Pablo Cholbi Alenda

Supervisor:
Dr. Àngel Perles Ivars

May 2014

Table of Contents

Illustration Index

1. Requirements

The end user's host computer, used for the installation and management of the application, may run any of the following operational systems:

The minimum recommended requirements are listed below:

Management

- 128 MB of RAM
- 100 MB of free disk space
- 1 Ethernet port
- 1 Ethernet network or switch

2. Installation and Management

2.1. Application Installation

This section is a walkthrough of the installation process of the application and the application database. The process of updating the application binary file or the application database is the same as there is no installer or package manager involved, it is a manual process.

This section focuses on the installation of a pre-compiled binary of the application and the transfer of an exiting database to the target system.

The first step is to make sure that the required directories exist on the target system. To do this, log in as root user on the target system form the host computer with SSH. if the project report was followed, the default IP address of the target system is 192.168.1.100.

```
user@host:~$ ssh root@192.168.1.100
```

Once logged in, the following commands shall create the directory for the application binary file and the database if they do not already exist. Please note that these commands create the default directories for these files according the project report; which are */usr/local/bin* for the application binary file and */home/root/.QtScale* for the database file.

```
root@target:~$ if [[ ! -e /usr/local/bin ]]; then mkdir
  /usr/local/bin; fi
root@target:~$ if [[ ! -e /home/root/.QtScale ]]; then mkdir
  /home/root/.QtScale; fi
```

After creating the directories, it is possible to close the SSH session as the rest of the process shall be performed from the host computer.

```
root@target:~$ exit
```

To install the application binary file, navigate to the directory where the binary file is located on the host computer and execute the following command. This command will transfer the binary to */usr/local/bin.*

```
user@host:~$ scp QtScale root@192.168.1.100:/usr/local/bin
```

To transfer the application database file, navigate to the directory where the database file is located on the host computer and execute the following command. This command will transfer the database to */home/root/.QtScale.*

```
user@host:~$ scp QtScale.sqlite
  root@192.168.1.100:/home/root/.QtScale
```

With this, the installation or update of the application binary and/or the database is complete. However, that does not mean that the system, as a whole, behaves exactly as the specifications demand. For further configuration, please consult the project report.

As an alternative to the previous process, if the file manager on the host computer supports it, it is possible to start and SFTP session with the target system and manually create and transfer the files through a graphical file manager.

To do this, the file manager must be opened and the following line typed into the address bar:

```
sftp://root@192.168.1.100
```

2.2. Database Management

This section refers to the management and updating of existing database records. New records are recommended to be inserted according the method described in the project report as it is the only one where the image insertion has been tested to work successfully.

For the management of the database, various options are available but for simplicity of use a GUI application is recommended. A popular standalone, open source, cross-platform GUI application for this purpose is “SQLite Database Browser”. SQLite Database Browser can be obtained from <http://sourceforge.net/projects/sqlitebrowser/>. However, there are many other applications that will perform the required task successfully.

This application is intuitive and easy to use even for novice users and it allows to browse and modify databases without SQL queries. Although it does support SQL query input for the users who prefer it.

Please note that the database should never be modified while in use by the application.

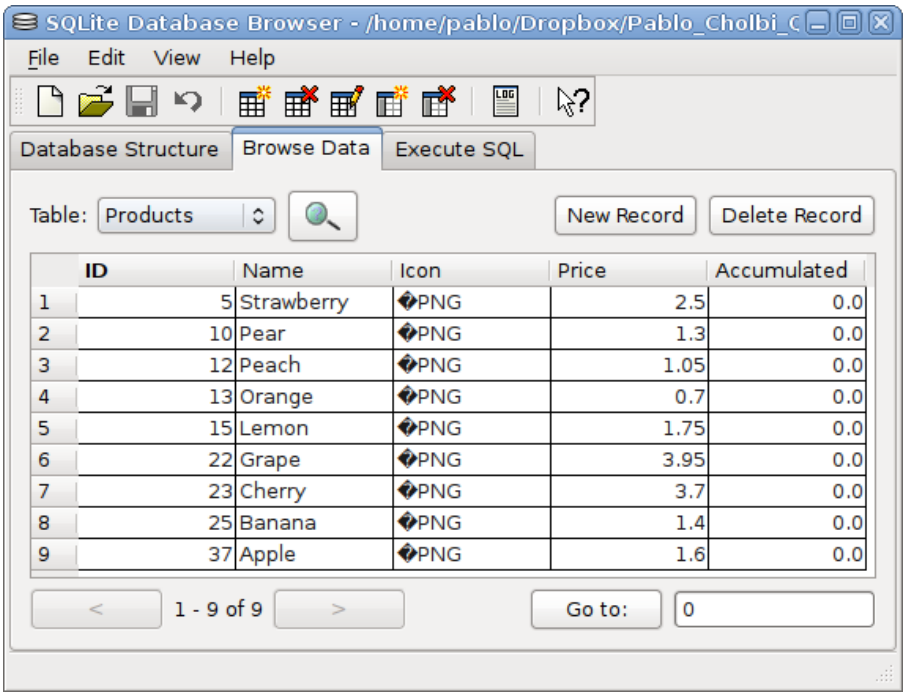


Figure 2.2-1: SQLite Database Browser

3. Application Interface

3.1. Command Line Interface

When starting the application, QtScale accepts some command line arguments to define the execution of the application. A list and a short description of the supported arguments is provided bellow.

--help	This arguments writes to STDOUT a help message similar to this table and exits successfully.
--version	This argument writes to STDOUT the version number of the QtScale application and exits successfully.
--path=<path>	This command sets the path to the database file to <path>. The default is path is *~/.QtScale*. The application supports the use of "~".
-qws	This argument start the Qt Windowing Server. This makes the application run with its own windowing server. This argument is important in embedded Linux systems as the application will not start if this argument is not passed to the application. This is an internal Qt argument and it is parsed before QtScale starts.

Please note that although the location of the database may be set trough command line arguments, the name of the database cannot be set and it it by default *QtScale.sqlite*. The name of the database that the application expects can only be changed at compile time.

If an invalid command is passed to the application, it exits immediately and writes to STDERR an error message.

3.2. Main Window

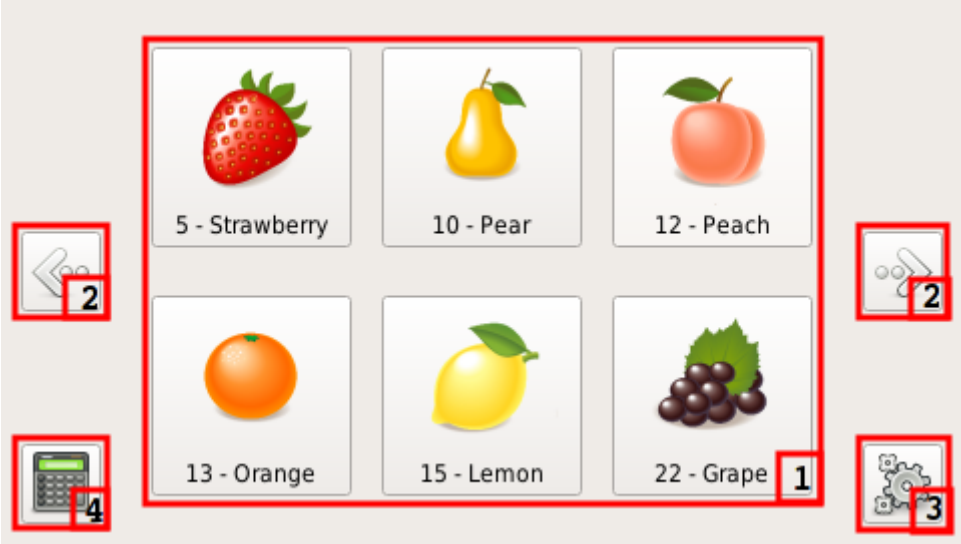


Figure 3.2-1: Main Window Interface

The Main Window, as its name indicates, is the primary window in the application, from which all other windows can be accessed (directly or indirectly). Product selection takes place in this window.

The different parts that constitute this window are explains below:

- **1** **Product selection:** This part of the window displays up to 6 product selection buttons with the product ID, product name and product image. Clicking on a product button lead to the corresponding product dialog to complete a sale. A set of 6 products constitutes a product “page”.Products are ordered by ID number. More information on the product dialog in the dedicated dialog subsection (section 3.3).
- **2** **Forward and back buttons:** By clicking these buttons, the product selection area (1) displays the next or previous set of 6 product (also know as product page). The buttons are hidden if it is not possible to advance or recede through the product list. This happens when the beginning or the end of the product list is reached.
- **3** **Options:** This button launches the options dialog. More information on this in the dedicated dialog subsection (section 3.6).
- **4** **Accumulated sales:** This button launches the accumulated sales dialog. More information on this in the dedicated dialog subsection (section 3.5).

3.3. Product Dialog

Figure 3.3-1: Product Dialog Interface

Once a product has been selected, this dialog is displayed, the weight is processed from the industrial scale and information of product weight, cost and price of the current sale is displayed. The user has the option to process the sale or cancel the transaction.

The different parts that constitute this dialog are explains below:

- **1** **Product image:** This part of the window displays the image associated to the product.
- **2** **Product information:** This part of the windows displays information of the product such as product ID, product name, product cost, weight of the product of the scale and price of the current transaction.
- **3** **Sale button:** This button accepts the current transaction, updates the accumulated sales of the product in the database and return to the main window. A sale can only be performed if the weight on the scale is grater than 0.00 kg and stable.
- **4** **Cancel button:** This button exist from the current dialog and returns to the main window without making the sale.

3.4. Message Dialog

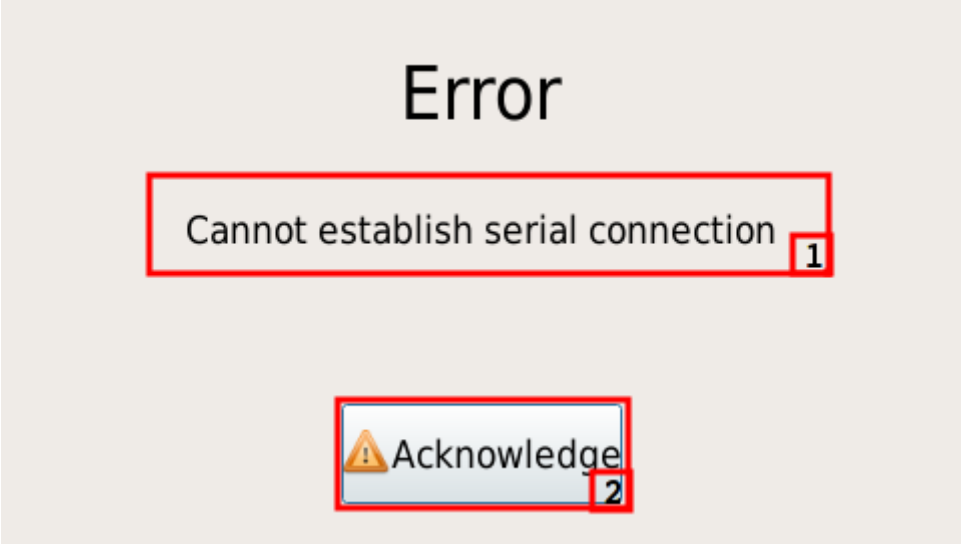


Figure 3.4-1: Message Dialog Interface

This dialog is displayed when an error en encountered and it must be acknowledged by the user before continuing. This dialog displays a warning in function of what error has been encountered.

The different parts that constitute this dialog are explains below:

- **1** **Warning message:** This part of the dialog displays the warning message associated with the error which has occurred.
- **2** **Acknowledgement button:** This button must be clicked by the user to acknowledge that an error has occurred.

3.5. Accumulated Sales Dialog

Figure 3.5-1: Accumulated Sales Dialog Interface

This dialog displays a scrollable table with all the accumulated sales for each product in units of mass (kg by default. Accumulated sales may also be cleared from this dialog.

The different parts that constitute this dialog are explains below:

- **1** **Product table:** This part of the window displays a scrollable table representing th product ID, the product name and the accumulated sales of all the products in the database in units of mass. Products are ordered by ID number.
- **2** **Clear Button:** This button resets all the accumulated sales of all the product in the database to 0.00.
- **3** **Close button:** This button exist from the current dialog and returns to the main window without clearing the accumulated sales.

3.6. Options Dialog

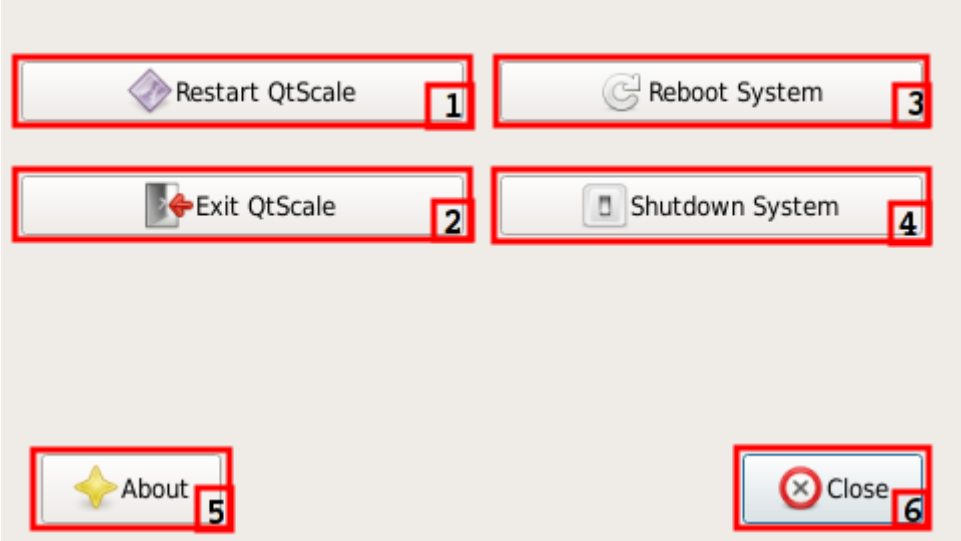


Figure 3.6-1: Options Dialog Interface

This dialog gives the user various options on the restarting and stoping of the application and the target system amongst others.

The different parts that constitute this dialog are explains below:

- **1** **Restart application:** This button restarts the application with the same arguments as the current instance was called. If the current instance was started remotely, clicking this button will free the shell.
- **2** **Exit application:** This button closes the current instance of QtScale.
- **3** **Restart system:** This button closes the current instance of QtScale and restarts the target system.
- **4** **Shutdown system:** This closes the current instance of QtScale and shuts down the target system.
- **5** **About:** This button launches the about dialog. More information on this in the dedicated dialog subsection (section 3.7).
- **6** **Close button:** This button exist from the current dialog and returns to the main window.

3.7. About Dialog

Figure 3.7-1: About Dialog Interface

This dialog displays information on the QtScale application.

The different parts that constitute this dialog are explains below:

- **1** **Product table:** This part of the window displays a scrollable table representing th product ID, the product name and the accumulated sales of all the products in the database in units of mass. Products are ordered by ID number.
- **2** **License:** This button launches the license dialog. More information on this in the dedicated dialog subsection (section 3.8).
- **3** **Close button:** This button exist from the current dialog and returns to the options dialog.

3.8. License Dialog

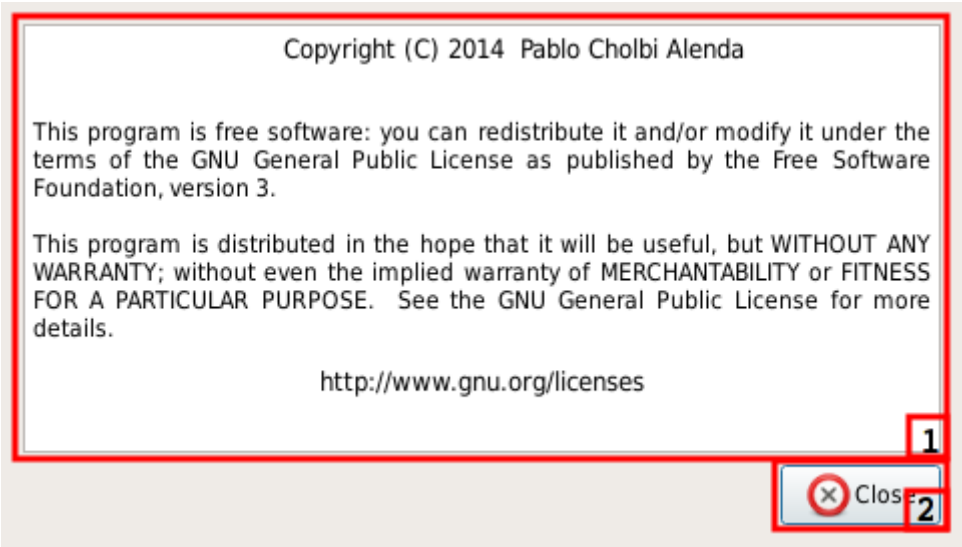


Figure 3.8-1: License Dialog Interface

This dialog displays the author, year and licence of the QtScale application.

The different parts that constitute this dialog are explains below:

- **1** **License information:** This part of the dialog displays information on the licence under which QtScale is distributed, the author and the year.
- **2** **Close button:** This button exist from the current dialog and returns to the about dialog.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Escuela Técnica Superior de Ingeniería del Diseño

Bachelor's Degree in Industrial Electronics and Automation Engineering

Final Year Engineering Project

QT SET-UP FOR TEXAS INSTRUMENTS AM335X ARM CORTEX-A8 USING LINUX: APPLICATION FOR AN INDUSTRIAL SCALE

Appendix 1 - Source Code Documentation

Author:
Pablo Cholbi Alenda

Supervisor:
Dr. Àngel Perles Ivars

May 2014

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

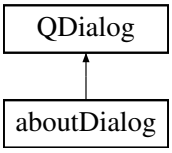
Class Documentation

4.1 aboutDialog Class Reference

this class displays "about" information of the application. The information is static and defined in the UI file.

`#include <aboutdialog.h>`

Inheritance diagram for aboutDialog:



Public Member Functions

- aboutDialog (QWidget *parent=0)

  Class constructor of aboutDialog object.
- ~aboutDialog ()

  Class destructor of aboutDialog object.

4.1.1 Detailed Description

this class displays "about" information of the application. The information is static and defined in the UI file.

Definition at line 42 of file aboutdialog.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 aboutDialog::aboutDialog (QWidget * *parent* = `0`) `[explicit]`

Class constructor of aboutDialog object.

This function creates an instance of aboutDialog class and initializes the widget.

Definition at line 39 of file aboutdialog.cpp.

4.1.2.2 aboutDialog::~aboutDialog ( )

Class destructor of aboutDialog object.

This function destroys an instance of aboutDialog class.

Definition at line 61 of file aboutdialog.cpp.

The documentation for this class was generated from the following files:

- /home/pablo/Developing/Qt/QtScale/src/aboutdialog.h
- /home/pablo/Developing/Qt/QtScale/src/aboutdialog.cpp

4.2 Command Class Reference

this class encapsulates the methods and variables necesary to parse the commandline arguments that are supported.

`#include <commandline.h>`

Public Member Functions

- Command ()

  Class constructor of Command object.
- errorCode parseArguments (QStringList argumentList)

  Parse incoming arguments.
- QString databasePath ()

  Return database location.

Protected Attributes

- QString basePath

  Database name.
- QString database

  Database path.

4.2.1 Detailed Description

this class encapsulates the methods and variables necesary to parse the commandline arguments that are supported.

Definition at line 39 of file commandline.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Command::Command ( )

Class constructor of Command object.

This function creates an instance of Command class and initializes the class variables.

Definition at line 45 of file commandline.cpp.

4.2.3 Member Function Documentation

4.2.3.1 QString Command::databasePath ( )

Return database location.

This function returns as a QString the path to the QtScale database file such that other modules can interact with it.

Returns

The path of the database.

Definition at line 152 of file commandline.cpp.

4.2.3.2 errorCode Command::parseArguments (QStringList *argumentList*)

Parse incoming arguments.

This function parses an incoming vector of arguments.

If an unexpected argument is found, the function returns argumentError error code.

Otherwise, it returns SUCCESS.

```
Supported arguments:
--help        Show help message
--version     Display application version information
--path=<dir>  Set database location
```

It should be noted that argument "-qws" is an internal Qt argument and is parsed before this function is executed.

Parameters

argumentList	Argument vector.

Returns

Error code, being SUCCESS if all arguments were parsed correctly or argumentError otherwise.

Definition at line 74 of file commandline.cpp.

The documentation for this class was generated from the following files:

- /home/pablo/Developing/Qt/QtScale/src/commandline.h
- /home/pablo/Developing/Qt/QtScale/src/commandline.cpp

4.3 Database Class Reference

this class encapsulates the methods and variables necessary to manage the SQLite database.

```
#include <database.h>
```

Public Member Functions

- Database ()

  Class constructor of Database.
- void setDatabase (QString database)

  Set database path.

- errorCode openDatabase ()

  Open database.
- void closeDatabase ()

  Close database.
- QList< int > list ()

  List database records.
- QIcon icon (int ID)

  Get product icon.
- QString name (int ID)

  Get product name.
- float price (int ID)

  Get product price.
- void updateAccum (float weight, int ID)

  Get product price.
- void clearAccum ()

  Reset accumulated sales.

Protected Attributes

- QSqlDatabase db

  Database object.

4.3.1 Detailed Description

this class encapsulates the methods and variables necessary to manage the SQLite database.

The database file must have a table named "Products" with the following fields:

```
-->Products Table:
+--------------------------+-------------+-------------+--------------+--------------------+
| ID (integer primary key) | Name (text) | Icon (blob) | Price (real) | Accumulated (real) |
+==========================+=============+=============+==============+====================+
| ID 1                     | Product 1   | PNG 1       | Price 1      | Accum 1            |
+--------------------------+-------------+-------------+--------------+--------------------+
| ID 2                     | Product 2   | PNG 2       | Price 2      | Accum 2            |
+--------------------------+-------------+-------------+--------------+--------------------+
| ...                      | ...         | ...         | ...          | ...                |
+--------------------------+-------------+-------------+--------------+--------------------+
```

For more information, please consult the project report.

Definition at line 58 of file database.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Database::Database ( )

Class constructor of Database.

This function creates an instance of Databaseand initializes the class member.

Definition at line 47 of file database.cpp.

4.3.3 Member Function Documentation

4.3.3.1 void Database::clearAccum ( )

Reset accumulated sales.

This function updates the accumulated sales of all products in the database and sets them to 0.

This function does not take parameters as it relies on other Database class functions.

Definition at line 213 of file database.cpp.

4.3.3.2 void Database::closeDatabase ( )

Close database.

This function closes the database.

Definition at line 93 of file database.cpp.

4.3.3.3 QIcon Database::icon (int *ID*)

Get product icon.

This function returns as a QIcon the icon for the specified product ID in the database.

Parameters

ID	Product index number (ID field in database).

Returns

Icon of specified product.

Definition at line 129 of file database.cpp.

4.3.3.4 QList< int > Database::list ( )

List database records.

This function returns a QList object with all the ID numbers of the products in the database in ascending order.

Returns

List of product IDs.

Definition at line 106 of file database.cpp.

4.3.3.5 QString Database::name (int *ID*)

Get product name.

This function returns the name of the specified product ID in the database.

Parameters

ID	Product index number (ID field in database).

Returns

Name of specified product.

Definition at line 151 of file database.cpp.

4.3.3.6 errorCode Database::openDatabase ( )

Open database.

This function opens the database to be able to interact with it.

If the SQLite database failed to open, the function returns databaseError error code.

Otherwise, it returns SUCCESS.

Returns

Error code, being SUCCESS if the database was opened or databaseError otherwise.

Definition at line 75 of file database.cpp.

4.3.3.7 float Database::price (int *ID*)

Get product price.

This function returns the Price of the specified product ID in the database.

Parameters

ID	Product index number (ID field in database).

Returns

Price of specified product.

Definition at line 171 of file database.cpp.

4.3.3.8 void Database::setDatabase (QString *database*)

Set database path.

This function sets the path the SQLite database.

Parameters

database	Path to SQLite database.

Definition at line 59 of file database.cpp.

4.3.3.9 void Database::updateAccum (float *weight,* int *ID*)

Get product price.

This function updates the accumulated sales in the product database record.

Parameters

weight	Sale to add to accumulated sales.
ID	Product index number (ID field in database).

Definition at line 189 of file database.cpp.

The documentation for this class was generated from the following files:

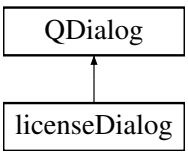
- /home/pablo/Developing/Qt/QtScale/src/database.h
- /home/pablo/Developing/Qt/QtScale/src/database.cpp

4.4 licenseDialog Class Reference

this class displays the license information of the application. The information is static and defined in the UI file.

`#include <licensedialog.h>`

Inheritance diagram for licenseDialog:



Public Member Functions

- licenseDialog (QWidget *parent=0)

  Class constructor of licenseDialog object.
- ~licenseDialog ()

  Class destructor of licenseDialog object.

4.4.1 Detailed Description

this class displays the license information of the application. The information is static and defined in the UI file.

Definition at line 40 of file licensedialog.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 licenseDialog::licenseDialog (QWidget * *parent* = 0) [explicit]

Class constructor of licenseDialog object.

This function creates an instance of licenseDialog class and initializes the widget.

Definition at line 38 of file licensedialog.cpp.

4.4.2.2 licenseDialog::~licenseDialog ( )

Class destructor of licenseDialog object.

This function destroys an instance of licenseDialog class.

Definition at line 58 of file licensedialog.cpp.

The documentation for this class was generated from the following files:

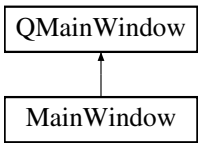
- /home/pablo/Developing/Qt/QtScale/src/licensedialog.h
- /home/pablo/Developing/Qt/QtScale/src/licensedialog.cpp

4.5 MainWindow Class Reference

this class encapsulates the methods and variables necesary to display the main product selection window.

`#include <mainwindow.h>`

Inheritance diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget *parent=0)

  Class constructor of MainWindow object.
- ~MainWindow ()

  Class destructor of MainWindow object.
- void setup (QApplication *a)

  Setup main window.
- void redraw ()

  Redraw main window.

Protected Attributes

- int page

  Current page being displayed.
- Database db

  Product database file.
- QList< int > vector

  ID numbers of the products.
- productStruct item

  Structure with product information.

4.5.1 Detailed Description

this class encapsulates the methods and variables necesary to display the main product selection window.

Definition at line 49 of file mainwindow.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 MainWindow::MainWindow (QWidget * *parent* = 0) [explicit]

Class constructor of MainWindow object.

This function creates an instance of MainWindow class, initializes the class variables and the widget.

Definition at line 51 of file mainwindow.cpp.

4.5.2.2 MainWindow::~MainWindow ( )

Class destructor of MainWindow object.

This function destroys an instance of MainWindow class.

Definition at line 79 of file mainwindow.cpp.

4.5.3 Member Function Documentation

4.5.3.1 void MainWindow::redraw ( )

Redraw main window.

This function updates the main window interface according to the information recorded on the database and the current "page" that is being viewed. Up to 6 products can be viewed at a time.

Definition at line 123 of file mainwindow.cpp.

4.5.3.2 void MainWindow::setup (QApplication * *a*)

Setup main window.

This function attempts to parse incoming command line arguments and open the database specified by these arguments (or default). Application exits with fail status if any of these actions fail.

This function must be called after the class object has been created and before the dialog is executed.

Parameters

*a	Pointer to QApplication object.

Definition at line 96 of file mainwindow.cpp.

The documentation for this class was generated from the following files:

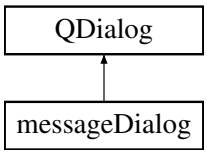
- /home/pablo/Developing/Qt/QtScale/src/mainwindow.h
- /home/pablo/Developing/Qt/QtScale/src/mainwindow.cpp

4.6 messageDialog Class Reference

this class encapsulates the methods and variables necesary to display error and information messages.

`#include <messagedialog.h>`

Inheritance diagram for messageDialog:



Public Member Functions

- messageDialog (QWidget *parent=0)

  Class constructor of messageDialog object.
- ~messageDialog ()

  Class destructor of messageDialog object.
- void errorMessage (errorCode error)

  Show error message.
- void genericMessage (QString msg)

  Show generic message.

4.6.1 Detailed Description

this class encapsulates the methods and variables necesary to display error and information messages.

Definition at line 42 of file messagedialog.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 messageDialog::messageDialog (QWidget * *parent* = 0) [explicit]

Class constructor of messageDialog object.

This function creates an instance of messageDialog class and initializes the widget.

Definition at line 42 of file messagedialog.cpp.

4.6.2.2 messageDialog::~messageDialog ( )

Class destructor of messageDialog object.

This function destroys an instance of messageDialog class.

Definition at line 58 of file messagedialog.cpp.

4.6.3 Member Function Documentation

4.6.3.1 void messageDialog::errorMessage (errorCode *error*)

Show error message.

This function displays an error message for the defined errors in errorCode.

Parameters

error	Error to me informed of.

Definition at line 70 of file messagedialog.cpp.

4.6.3.2 void messageDialog::genericMessage (QString *msg*)

Show generic message.

This function displays a generic message on screen.

Parameters

msg	Message to be displayed.

Definition at line 112 of file messagedialog.cpp.

The documentation for this class was generated from the following files:

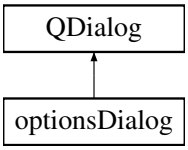
- /home/pablo/Developing/Qt/QtScale/src/messagedialog.h
- /home/pablo/Developing/Qt/QtScale/src/messagedialog.cpp

4.7 optionsDialog Class Reference

this class encapsulates the methods and variables necesary to display and execute configuration options.

`#include <optionsdialog.h>`

Inheritance diagram for optionsDialog:



Public Member Functions

- optionsDialog (QWidget *parent=0)

  Class constructor of optionDialog object.
- ~optionsDialog ()

  Class destructor of optionsDialog object.

4.7.1 Detailed Description

this class encapsulates the methods and variables necesary to display and execute configuration options.

Definition at line 42 of file optionsdialog.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 optionsDialog::optionsDialog (QWidget * *parent* = 0) [explicit]

Class constructor of optionDialog object.

This function creates an instance of optionsDialog class and initializes the widget.

Definition at line 41 of file optionsdialog.cpp.

4.7.2.2 optionsDialog::~optionsDialog ( )

Class destructor of optionsDialog object.

This function destroys an instance of optionsDialog class.

Definition at line 70 of file optionsdialog.cpp.

The documentation for this class was generated from the following files:

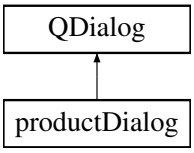
- /home/pablo/Developing/Qt/QtScale/src/optionsdialog.h
- /home/pablo/Developing/Qt/QtScale/src/optionsdialog.cpp

4.8 productDialog Class Reference

this class encapsulates the methods and variables necessary to display product weight and information.

`#include <productdialog.h>`

Inheritance diagram for productDialog:



Public Member Functions

- productDialog (QWidget *parent=0)

  Class constructor of productDialog object.
- ~productDialog ()

  Class destructor of productDialog object.
- bool setup (productStruct *item)

  Product dialog.

4.8.1 Detailed Description

this class encapsulates the methods and variables necessary to display product weight and information.

Definition at line 45 of file productdialog.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 productDialog::productDialog (QWidget * *parent* = 0) [explicit]

Class constructor of productDialog object.

This function creates an instance of productDialog class and initializes the widget.

Definition at line 50 of file productdialog.cpp.

4.8.2.2 productDialog::~productDialog ( )

Class destructor of productDialog object.

This function destroys an instance of productDialog class.

Definition at line 72 of file productdialog.cpp.

4.8.3 Member Function Documentation

4.8.3.1 bool productDialog::setup (productStruct * *item*)

Product dialog.

This function initializes the widget elements according to the productStruct and prepares the serial connections to weigh the product.

This function must be called after the class object has been created and the dialog should only be executed if the setup exits successfully.

Parameters

∗item	Pointer to productStruct object.

Returns

TRUE is setup exited successfully or FALSE otherwise.

Definition at line 90 of file productdialog.cpp.

The documentation for this class was generated from the following files:

- /home/pablo/Developing/Qt/QtScale/src/productdialog.h
- /home/pablo/Developing/Qt/QtScale/src/productdialog.cpp

4.9 productStruct Struct Reference

Product data structure.

`#include <productstruct.h>`

Public Attributes

- QString name

  Name of product.
- QIcon icon

  Icon of product.
- int id

  Id of product.
- float price

  Product price per unit of weight.
- float weight

  Weight received from the scale.

4.9.1 Detailed Description

Product data structure.

This data structure encapsulated variables to be used in the product dialog.

Definition at line 42 of file productstruct.h.

The documentation for this struct was generated from the following file:

- /home/pablo/Developing/Qt/QtScale/src/productstruct.h

4.10 serialScale Class Reference

this class encapsulates the methods and variables necessary to manage the communication with the industrial scale.

`#include <serialscale.h>`

Public Member Functions

- serialScale ()

  Class constructor of serialScale object.
- ~serialScale ()

  Class destructor of serialScale object.
- errorCode openConnection ()

  Establish serial connection with the industrial scale.
- void closeConnection ()

  Closes the serial port.
- void sendData (const char *data, int length)

  Send data over serial.
- int receiveData (const char *data, int maxData)

  Receive data over serial.
- void update ()

  Try to get new measurement.
- bool buffScan (char *buff, int numData, int *msgEnd)

  Scan buffer for measurements.
- void process (char *buff, int size)

  Process possible data stream.
- float getWeight ()

  Get weight from scale.
- bool isStable ()

  Get stability of scale measurement.
- void loopbackTest ()

  Test serial port.

Protected Attributes

- bool port

  Serial port found.
- bool connection

  Connection established.
- bool stable

  Stability of weight.
- bool signal

  Show warning.
- char buffer [bufferSize]

  Serial reception buffer.
- int totalData

  Position in buffer.
- double weight

  Weight of product.

4.10.1 Detailed Description

this class encapsulates the methods and variables necessary to manage the communication with the industrial scale.

This class manages the serial communication between the industrial scale and the application. Communication is carried put over RS-232.

The string processing carried out in this class is intended to work with the Microgram ie21 from Microgram Instruments Española, S.A.

String processing functions must be modified if other hardware is to be used.

The data stream encoding for this scale is shown here:

```
  +--------------> Data stream start character "s"
  |     +--------> <space>
  |     |  +-----> sign of weight (<space> or "-")
  |     |  |  +--> 9 characters for the weight
  |     |  |  |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|S |D |  |- |  |  |  |1 |2 |3 |. |4 |  |  |k |g |  |\r|\n|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     |                                      |  |  |  |  |
     +---> 2 characters for state    "k" <--+  |  |  |  |
(#1) +-> <space> <space>             "g" <-----+  |  |  |
(#2) +-> "D" <space>             <space> <--------+  |  |
(#3) +-> "I" "+"                    0x0D <-----------+  |
(#4) +-> "I" "-"                    0x0A <--------------+

(#1) - Stable weight
(#2) - Unstable weight
(#3) - Out-of-Range (scale overloaded)
(#4) - Out-of-Range (scale underloaded)
```

Definition at line 77 of file serialscale.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 serialScale::serialScale ( )

Class constructor of serialScale object.

This function creates an instance of serialScale class and initializes the class variables.

Definition at line 43 of file serialscale.cpp.

4.10.2.2 serialScale::∼serialScale ( )

Class destructor of serialScale object.

This function destroys an instance of serialScale class.

Definition at line 57 of file serialscale.cpp.

4.10.3 Member Function Documentation

4.10.3.1 bool serialScale::buffScan (char ∗ *buff,* int *numData,* int ∗ *msgEnd*)

Scan buffer for measurements.

This is the function searched for the start and end characters in the buffer. If they are found, the possible array is passed on for further processing.

This function depend on process().

This function is intended to work with the Microgram ie21 industrial scale from Microgram Instruments Española, S.A. For more information on the ie21 data stream coding please consult the serialScale class documentation and the annex which includes the manufacturers datasheet.

Returns

TRUE is a message is found and FALSE otherwise.

Definition at line 265 of file serialscale.cpp.

4.10.3.2 void serialScale::closeConnection ( )

Closes the serial port.

Closes the serial port if it is open.

This function is part of the "low level" functions of this class.

Definition at line 131 of file serialscale.cpp.

4.10.3.3 float serialScale::getWeight ( )

Get weight from scale.

Returns

Weight in kilograms.

Definition at line 399 of file serialscale.cpp.

4.10.3.4 bool serialScale::isStable ( )

Get stability of scale measurement.

Returns

Stability of most recent measurement.

Definition at line 409 of file serialscale.cpp.

4.10.3.5 void serialScale::loopbackTest ( )

Test serial port.

This functions can be used to test the serial port on the system by loopback. If the industrial scale is not connected to the system and the system's serial port pin RX is connected the to its TX pin, all data sent is redirected as input data. This allows to simulate data stream transmission from the scale.

Currently, there are 3 examples streams already defined for quick testing and debugging.

```
RX <--->TX
```

This function is intended to work with the Microgram ie21 industrial scale from Microgram Instruments Española, S.A. For more information on the ie21 data stream coding please consult the serialScale class documentation and the annex which includes the manufacturers datasheet.

Definition at line 436 of file serialscale.cpp.

4.10.3.6 errorCode serialScale::openConnection ( )

Establish serial connection with the industrial scale.

This function creates, configures and opens a QSerialPort connection to communicate with the industrial scale and receive information on the weight of the products.

Connection parameters configured to work with the Microgram ie21 industrial scale from Microgram Instruments Española, S.A.

```
Serial port:  /dev/ttyUSB0
Baud rate:    9600
Data bits:    8 bits
Stop bits:    1 bit
Parity:       no parity
Flow control: no flow control
```

This function is part of the "low level" functions of this class.

Returns

Error code, being SUCCESS if connection was established or serialError otherwise.

Definition at line 87 of file serialscale.cpp.

4.10.3.7 void serialScale::process (char * *buff,* int *size*)

Process possible data stream.

This function takes a possible data stream and processes it to either discard it or extract from it the weight and stability values that the scale is sending over RS-232.

This function is intended to work with the Microgram ie21 industrial scale from Microgram Instruments Española, S.A. For more information on the ie21 data stream coding please consult the serialScale class documentation and the annex which includes the manufacturers datasheet.

Definition at line 309 of file serialscale.cpp.

4.10.3.8 int serialScale::receiveData (const char * *data,* int *maxData*)

Receive data over serial.

This functions is non-blocking.

This function is part of the "low level" functions of this class.

Parameters

**data*	Pointer to char vector to which to write data.
maxData	Maximum number of data bytes to receive.

Returns

Number of Bytes received.

Definition at line 168 of file serialscale.cpp.

4.10.3.9 void serialScale::sendData (const char * *data,* int *length*)

Send data over serial.

This functions is non-blocking.

This function is part of the "low level" functions of this class.

Parameters

∗data	Pointer to char vector from which to read data.
length	Length in Bytes of data to be sent.

Definition at line 149 of file serialscale.cpp.

4.10.3.10 void serialScale::update ( )

Try to get new measurement.

This is the function to be called to attempt to retrieve and process a new weight measurement sent by the scale over RS-232.

When the function is called, data Bytes received since the last call are added to the buffer. Then the buffer is evaluated for valid data streams. If a valid stream is found and processed, it is removed from the buffer.

This function is intended to work with the Microgram ie21 industrial scale from Microgram Instruments Española, S.A. For more information on the ie21 data stream coding please consult the serialScale class documentation and the annex which includes the manufacturers datasheet.

This function depens on buffScan() and process().

Definition at line 210 of file serialscale.cpp.

The documentation for this class was generated from the following files:

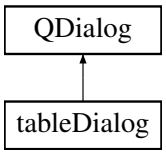
- /home/pablo/Developing/Qt/QtScale/src/serialscale.h
- /home/pablo/Developing/Qt/QtScale/src/serialscale.cpp

4.11 tableDialog Class Reference

this class encapsulates the methods and variables necessary to display the accumulated sales of all products in the database.

`#include <tabledialog.h>`

Inheritance diagram for tableDialog:



Public Member Functions

- tableDialog (QWidget ∗parent=0)

  Class constructor of tabletDialog object.
- ∼tableDialog ()

  Class destructor of tableDialog object.

4.11.1 Detailed Description

this class encapsulates the methods and variables necessary to display the accumulated sales of all products in the database.

Definition at line 43 of file tabledialog.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 tableDialog::tableDialog (QWidget * *parent* = 0) [explicit]

Class constructor of tabletDialog object.

This function creates an instance of productDialog class and initializes the widget.

The table displays the Id, product name and accumulated sales for all products in the database.

This function must be called after the class object has been created and before the dialog is executed.

Definition at line 47 of file tabledialog.cpp.

4.11.2.2 tableDialog::~tableDialog ( )

Class destructor of tableDialog object.

This function destroys an instance of tableDialog class.

Definition at line 84 of file tabledialog.cpp.

The documentation for this class was generated from the following files:

- /home/pablo/Developing/Qt/QtScale/src/tabledialog.h
- /home/pablo/Developing/Qt/QtScale/src/tabledialog.cpp

Chapter 5

File Documentation

5.1 /home/pablo/Developing/Qt/QtScale/src/aboutdialog.cpp File Reference

Class implementation of aboutDialog.

```
#include "aboutdialog.h"
#include "ui_aboutdialog.h"
#include "licensedialog.h"
```

5.1.1 Detailed Description

Class implementation of aboutDialog. This class displays information about the application.

Date

March, 2014

Author

Pablo Cholbi

Definition in file aboutdialog.cpp.

5.2 /home/pablo/Developing/Qt/QtScale/src/aboutdialog.h File Reference

Class definition of aboutDialog.

```
#include <QDialog>
#include "licensedialog.h"
```

Classes

- class aboutDialog

  this class displays "about" information of the application. The information is static and defined in the UI file.

5.2.1 Detailed Description

Class definition of aboutDialog.

Date

March, 2014

Author

Pablo Cholbi

Definition in file aboutdialog.h.

5.3 /home/pablo/Developing/Qt/QtScale/src/commandline.cpp File Reference

Class implementation of Command.

```
#include <stdio.h>
#include <cstdlib>
#include <QDir>
#include <QString>
#include <QStringList>
#include "commandline.h"
#include "errorcode.h"
```

5.3.1 Detailed Description

Class implementation of Command. This class contains the methods and members to be able to evaluate the supported command line arguments. The argument list is easily expandable.

Date

March, 2014

Author

Pablo Cholbi

Definition in file commandline.cpp.

5.4 /home/pablo/Developing/Qt/QtScale/src/commandline.h File Reference

Class definition of Command.

```
#include <QString>
#include <QStringList>
#include "errorcode.h"
```

Classes

- class Command

  this class encapsulates the methods and variables necesary to parse the commandline arguments that are supported.

5.4.1 Detailed Description

Class definition of Command.

Date

March, 2014

Author

Pablo Cholbi

Definition in file commandline.h.

5.5 /home/pablo/Developing/Qt/QtScale/src/database.cpp File Reference

Class implementation of Database.

```
#include <stdio.h>
#include <QIcon>
#include <QList>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlQueryModel>
#include <QVariant>
#include "database.h"
#include "errorcode.h"
```

5.5.1 Detailed Description

Class implementation of Database. This class contains the methods and members that enable interaction with SQLite databases.

Date

March, 2014

Author

Pablo Cholbi

Definition in file database.cpp.

5.6 /home/pablo/Developing/Qt/QtScale/src/database.h File Reference

Class definition of Database.

```
#include <QList>
#include <QIcon>
#include <QString>
#include <QSqlDatabase>
#include <QSqlQueryModel>
#include "errorcode.h"
```

Classes

- class Database

  this class encapsulates the methods and variables necessary to manage the SQLite database.

5.6.1 Detailed Description

Class definition of Database.

Date

March, 2014

Author

Pablo Cholbi

Definition in file database.h.

5.7 /home/pablo/Developing/Qt/QtScale/src/errorcode.h File Reference

Definition of errorCode data type.

Enumerations

- enum errorCode {
  SUCCESS, argumentError, databaseError, queryError,
  serialError, scaleError }

  Error code signaling data type.

5.7.1 Detailed Description

Definition of errorCode data type. This file contains the implementation of the errorCode enumerated data type.

Date

March, 2014

Author

Pablo Cholbi

Definition in file errorcode.h.

5.7.2 Enumeration Type Documentation

5.7.2.1 enum errorCode

Error code signaling data type.

This data type is used as a return in various functions to signal if the function exited successfully or if an error occurred. If an error occurs, errorCode also gives information on the error type, such that it can be handled accordingly and/or warning can be displayed/printed.

Enumerator

SUCCESS function exited successfully

argumentError failed to parse command line argument

databaseError failed to open/close database

queryError query failed to execute successfully

serialError serial connection could not be established

scaleError scale is sending unexpected values. Calibration needed.

Definition at line 40 of file errorcode.h.

5.8 /home/pablo/Developing/Qt/QtScale/src/licensedialog.cpp File Reference

Class implementation of licenseDialog.

```
#include "licensedialog.h"
#include "ui_licensedialog.h"
```

5.8.1 Detailed Description

Class implementation of licenseDialog. This class displays the license of the application.

Date

March, 2014

Author

Pablo Cholbi

Definition in file licensedialog.cpp.

5.9 /home/pablo/Developing/Qt/QtScale/src/licensedialog.h File Reference

Class definition of licenseDialog.

```
#include <QDialog>
```

Classes

- class licenseDialog

  this class displays the license information of the application. The information is static and defined in the UI file.

5.9.1 Detailed Description

Class definition of licenseDialog.

Date

March, 2014

Author

Pablo Cholbi

Definition in file licensedialog.h.

5.10 /home/pablo/Developing/Qt/QtScale/src/main.cpp File Reference

Start QtScale application.

```
#include <QApplication>
#include <QStringList>
#include <QProcess>
#include "mainwindow.h"
```

Functions

- int main (int argc, char ∗argv[])

  Start application.

5.10.1 Detailed Description

Start QtScale application.

Date

March, 2014

Author

Pablo Cholbi

Definition in file main.cpp.

5.10.2 Function Documentation

5.10.2.1 int main (int *argc*, char ∗ *argv[]*)

Start application.

Parameters

argc	Argument count.
argv	Argument vector.

This function starts the application, creates the main window and handles the the restart of the application if requested by a child.

Returns

Exit status of the application.

Definition at line 43 of file main.cpp.

5.11 /home/pablo/Developing/Qt/QtScale/src/mainwindow.cpp File Reference

Class implementation of MainWindow.

```
#include <QApplication>
#include <QString>
#include <QStringList>
#include <QList>
#include <cstdlib>
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "errorcode.h"
#include "commandline.h"
#include "database.h"
#include "optionsdialog.h"
#include "productdialog.h"
#include "tabledialog.h"
#include "productstruct.h"
```

5.11.1 Detailed Description

Class implementation of MainWindow. This class displays the main product selection window.

Date

March, 2014

Author

Pablo Cholbi

Definition in file mainwindow.cpp.

5.12 /home/pablo/Developing/Qt/QtScale/src/mainwindow.h File Reference

Class definition of MainWindow.

```
#include <QMainWindow>
#include <QApplication>
#include <QList>
#include "commandline.h"
#include "database.h"
#include "optionsdialog.h"
#include "productdialog.h"
#include "tabledialog.h"
#include "productstruct.h"
```

Classes

- class MainWindow

  this class encapsulates the methods and variables necesary to display the main product selection window.

5.12.1 Detailed Description

Class definition of MainWindow.

Date

March, 2014

Author

Pablo Cholbi

Definition in file mainwindow.h.

5.13 /home/pablo/Developing/Qt/QtScale/src/messagedialog.cpp File Reference

Class implementation of messageDialog.

```
#include <stdio.h>
#include <QString>
#include "messagedialog.h"
#include "ui_messagedialog.h"
#include "errorcode.h"
```

5.13.1 Detailed Description

Class implementation of messageDialog. This class displays error and information messages.

Date

March, 2014

Author

Pablo Cholbi

Definition in file messagedialog.cpp.

5.14 /home/pablo/Developing/Qt/QtScale/src/messagedialog.h File Reference

MessageDialog class definition.

```
#include <QDialog>
#include "errorcode.h"
```

Classes

- class messageDialog

  this class encapsulates the methods and variables necesary to display error and information messages.

5.14.1 Detailed Description

MessageDialog class definition.

Date

March, 2014

Author

Pablo Cholbi

Definition in file messagedialog.h.

5.15 /home/pablo/Developing/Qt/QtScale/src/optionsdialog.cpp File Reference

Class implementation of optionsDialog.

```
#include <QProcess>
#include "optionsdialog.h"
#include "ui_optionsdialog.h"
#include "aboutdialog.h"
```

5.15.1 Detailed Description

Class implementation of optionsDialog. This class displays and executes configuration options.

Date

March, 2014

Author

Pablo Cholbi

Definition in file optionsdialog.cpp.

5.16 /home/pablo/Developing/Qt/QtScale/src/optionsdialog.h File Reference

Class definition of optionsDialog.

```
#include <QDialog>
#include "aboutdialog.h"
```

Classes

- class optionsDialog

  this class encapsulates the methods and variables necesary to display and execute configuration options.

5.16.1 Detailed Description

Class definition of optionsDialog.

Date

March, 2014

Author

Pablo Cholbi

Definition in file optionsdialog.h.

5.17 /home/pablo/Developing/Qt/QtScale/src/productdialog.cpp File Reference

Class implementation of productDialog.

```
#include <QTimer>
#include <QIcon>
#include <QString>
#include "productdialog.h"
#include "ui_productdialog.h"
#include "productstruct.h"
#include "serialscale.h"
#include "messagedialog.h"
#include "errorcode.h"
```

Variables

- QString currencyUnit = QString::fromUtf8("€")

  Symbol for currency unit.
- QString massUnit = QString("kg")

  Symbol for weight unit.

5.17.1 Detailed Description

Class implementation of productDialog. This class displays product information and weighs the product.

Date

March, 2014

Author

Pablo Cholbi

Definition in file productdialog.cpp.

5.18 /home/pablo/Developing/Qt/QtScale/src/productdialog.h File Reference

Class definition of productDialog.

```
#include <QDialog>
#include <QTimer>
#include "productstruct.h"
#include "serialscale.h"
```

Classes

- class productDialog

  this class encapsulates the methods and variables necessary to display product weight and information.

5.18.1 Detailed Description

Class definition of productDialog.

Date

March, 2014

Author

Pablo Cholbi

Definition in file productdialog.h.

5.19 /home/pablo/Developing/Qt/QtScale/src/productstruct.h File Reference

Data structure type productStruct.

```
#include <QIcon>
#include <QString>
#include "messagedialog.h"
```

Classes

- struct productStruct

  Product data structure.

5.19.1 Detailed Description

Data structure type productStruct. This file contains the implementation of the productStruct data structure.

Date

March, 2014

Author

Pablo Cholbi

Definition in file productstruct.h.

5.20 /home/pablo/Developing/Qt/QtScale/src/serialscale.cpp File Reference

Class implementation of serialScale.

```
#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>
#include <string.h>
#include <stdio.h>
#include "serialscale.h"
#include "errorcode.h"
```

5.20.1 Detailed Description

Class implementation of serialScale. This class manages the serial communication with the industrial scale.

Date

March, 2014

Author

Pablo Cholbi

Definition in file serialscale.cpp.

5.21 /home/pablo/Developing/Qt/QtScale/src/serialscale.h File Reference

Class definition of serialScale.

```
#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>
#include "errorcode.h"
#include "messagedialog.h"
```

Classes

- class serialScale

  this class encapsulates the methods and variables necessary to manage the communication with the industrial scale.

Macros

- #define bufferSize 2048

  Size of serial reception buffer.

5.21.1 Detailed Description

Class definition of serialScale.

Date

March, 2014

Author

Pablo Cholbi

Definition in file serialscale.h.

5.22 /home/pablo/Developing/Qt/QtScale/src/tabledialog.cpp File Reference

Class implementation of tableDialog.

```
#include <QSqlQueryModel>
#include "tabledialog.h"
#include "ui_tabledialog.h"
```

5.22.1 Detailed Description

Class implementation of tableDialog. This class displays a table showing the accumulated sales of the products in the database.

Date

March, 2014

Author

Pablo Cholbi

Definition in file tabledialog.cpp.

5.23 /home/pablo/Developing/Qt/QtScale/src/tabledialog.h File Reference

Class definition of tableDialog.

```
#include <QDialog>
#include <QSqlQueryModel>
#include "database.h"
```

Classes

- class tableDialog

  this class encapsulates the methods and variables necessary to display the accumulated sales of all products in the database.

5.23.1 Detailed Description

Class definition of tableDialog.

Date

March, 2014

Author

Pablo Cholbi

Definition in file tabledialog.h.

Index

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Escuela Técnica Superior de Ingeniería del Diseño

Bachelor's Degree in Industrial Electronics and Automation Engineering

Final Year Engineering Project

QT SET-UP FOR TEXAS INSTRUMENTS AM335X ARM CORTEX-A8 USING LINUX: APPLICATION FOR AN INDUSTRIAL SCALE

Appendix 2 - Source Code

Author:
Pablo Cholbi Alenda

Supervisor:
Dr. Àngel Perles Ivars

May 2014

Contents

1 aboutdialog.cpp

```
/**
*@file   aboutdialog.cpp
*@brief  Class implementation of aboutDialog.
*
* This class displays information about the application.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include "aboutdialog.h"
#include "ui_aboutdialog.h"
#include "licensedialog.h"

/**
  @brief Class constructor of aboutDialog object.

  This function creates an instance of aboutDialog class
  and initializes the widget.
*/
aboutDialog::aboutDialog(QWidget *parent):QDialog(parent),
    ui(new Ui::aboutDialog)
{
    ui->setupUi(this);

    // set to fullscreen and frameless
    this->setWindowFlags(Qt::FramelessWindowHint);
    this->setFixedHeight(272);
    this->setFixedWidth(480);

    // add icons to buttons
    ui->licenseButton->setIcon(QIcon(":/Buttons/Buttons/certificate.png"));
    ui->licenseButton->setIconSize(QSize(24,24));
    ui->closeButton->setIcon(QIcon(":/Buttons/Buttons/cancel.png"));
    ui->closeButton->setIconSize(QSize(24,24));
}
```

```
/**
  @brief Class destructor of aboutDialog object.

  This function destroys an instance of aboutDialog class.
*/
aboutDialog::~aboutDialog()
{
    delete ui;
}

/**
  @brief open licenseDialog.

  This slot opens the a "license" dialog.
*/
void aboutDialog::on_licenseButton_clicked()
{
    license = new licenseDialog();
    license->exec();
    delete license;
}

/**
  @brief exit aboutDialog.

  This slot closes destroys current "about" dialog.
*/
void aboutDialog::on_closeButton_clicked()
{
    this->close();
}

/*** end of file ************************************************************/
```

2 aboutdialog.h

```
/**
*@file   aboutdialog.h
*@brief  Class definition of aboutDialog.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef ABOUTDIALOG_H
#define ABOUTDIALOG_H

#include <QDialog>

#include "licensedialog.h"

namespace Ui
{
    class aboutDialog;
}

/**
  \brief this class displays "about" information of the application.
  The information is static and defined in the UI file.
*/
class aboutDialog : public QDialog
{
    Q_OBJECT

    public:
        explicit aboutDialog(QWidget *parent = 0);
        ~aboutDialog();

    private slots:
        void on_licenseButton_clicked();
        void on_closeButton_clicked();

    private:
        Ui::aboutDialog *ui;
```

```
        licenseDialog *license;
};

#endif // ABOUTDIALOG_H

/*** end of file ************************************************************/
```

3 aboutdialog.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>aboutDialog</class>
 <widget class="QDialog" name="aboutDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>480</width>
    <height>272</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Dialog</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout">
   <item>
    <widget class="QLabel" name="label_about">
     <property name="text">
      <string notr="true">&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
           style=&quot; font-size:16pt; font-weight:600;&quot;&gt;QtScale 1.00&lt
          ;/span&gt;&lt;/p&gt;&lt;p&gt;&lt;span style=&quot; font-size:11pt;&quot
          ;&gt;GUI application for bulk product commerce by weight&lt;br/&gt;
          using an industrial RS-232 scale.&lt;/span&gt;&lt;/p&gt;&lt;p&gt;&lt;
          span style=&quot; font-size:11pt;&quot;&gt;This application is intended
           to work with the Microgram ie21 scale&lt;br/&gt;from Microgram
          Instruments Espa ola , S.A.&lt;/span&gt;&lt;/p&gt;&lt;p&gt;&lt;span
          style=&quot; font-size:11pt;&quot;&gt;Pablo Cholbi Alenda&lt;/span&gt;&
          lt;/p&gt;&lt;p&gt;&lt;span style=&quot; font-size:11pt;&quot;&gt;
          Copyright    2014&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</
          string>
     </property>
     <property name="alignment">
      <set>Qt::AlignCenter</set>
     </property>
     <property name="openExternalLinks">
      <bool>true</bool>
     </property>
     <property name="textInteractionFlags">
      <set>Qt::TextBrowserInteraction</set>
     </property>
    </widget>
   </item>
   <item>
    <layout class="QHBoxLayout" name="horizontalLayout">
     <item>
      <widget class="QPushButton" name="licenseButton">
       <property name="text">
        <string>License</string>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QPushButton" name="closeButton">
       <property name="text">
```

```
        <string>Close</string>
       </property>
      </widget>
     </item>
    </layout>
   </item>
  </layout>
 </widget>
 <resources/>
 <connections>
  <connection>
   <sender>closeButton</sender>
   <signal>clicked()</signal>
   <receiver>aboutDialog</receiver>
   <slot>close()</slot>
   <hints>
    <hint type="sourcelabel">
     <x>213</x>
     <y>241</y>
    </hint>
    <hint type="destinationlabel">
     <x>144</x>
     <y>134</y>
    </hint>
   </hints>
  </connection>
 </connections>
</ui>
```

4 commandline.cpp

```
/**
*@file   commandline.cpp
*@brief  Class implementation of Command.
*
* This class contains the methods and members to be able to evaluate
* the supported command line arguments. The argument list is easily expandable.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <stdio.h>
#include <cstdlib>
#include <QDir>
#include <QString>
#include <QStringList>

#include "commandline.h"
#include "errorcode.h"

/**
  @brief Class constructor of Command object.

  This function creates an instance of Command class and initializes
  the class variables.
*/
Command::Command()
{
    basePath="~/.QtScale/";     // Default database name
    database="QtScale.sqlite";  // Default database path
}

/**
  @brief Parse incoming arguments.

  This function parses an incoming vector of arguments.<br>
  If an unexpected argument is found, the function returns
```

```
  argumentError error code.<br>
  Otherwise, it returns SUCCESS.

  \verbatim
  Supported arguments:
  --help        Show help message
  --version     Display application version information
  --path=<dir>  Set database location
  \endverbatim

  It should be noted that argument "-qws" is an internal Qt argument
  and is parsed before this function is executed.

  @param argumentList Argument vector.

  @returns Error code, being SUCCESS if all arguments were parsed correctly
  or argumentError otherwise.
*/
errorCode Command::parseArguments(QStringList argumentList)
{
    // arguments for this app
    QRegExp rxArgHelp("--help");
    QRegExp rxArgVersion("--version");
    QRegExp rxArgPath("--path=(.*)");
    QRegExp rxArgQWS("-qws");

    for (int i=1;i<argumentList.size();i++)
    {
        // help requested
        if (rxArgHelp.exactMatch(argumentList.at(i)))
        {
            fprintf(stdout,"Syntax: %s [--help] [--version] [--path=<dir>] [-qws
                ]\n"
                    ,argumentList.at(0).toUtf8().data());
            fprintf(stdout,"  --Help         : Show this message\n");
            fprintf(stdout,"  --version      : Print version information and exit\
                n");
            fprintf(stdout,"  --path=<dir>   : Path to database (default = ~/.
                QtScale)\n");
            fprintf(stdout,"                   Database must be named %s\n",
                database.toUtf8().data());
            fprintf(stdout,"  -qws           : Start application with Qt Windowing
                 Server\n\n");
            exit(EXIT_SUCCESS);
        }

        // record version
        else if (rxArgVersion.exactMatch(argumentList.at(i)))
        {
            fprintf(stdout,"%.2f\n", appVersion); // format and print version
            exit(EXIT_SUCCESS);
        }

        // path
        else if (rxArgPath.exactMatch(argumentList.at(i)))
        {
            // change database path from default
```

```
            basePath = rxArgPath.cap(1);
        }

        // Qt Windowing Server
        else if (rxArgQWS.exactMatch(argumentList.at(i)))
        {
            // It should be noted that argument "-qws" is an internal Qt
                argument
            // and is parsed before this function is executed.
            // This is a dummy "else if" case such that application restart
            // does not fail on non-embedded-Linux systems.
        }

        // unexpected argument
        else
        {
            fprintf(stderr,"Unknown␣command␣line␣argument,␣please␣use␣--help\n")
                ;
            return argumentError; // return error
        }
    }

    // expand "~/" to user's home directory if needed
    if (basePath.startsWith("~/"))
    {
        QDir dir;
        basePath.replace("~/",dir.homePath()+"/");
    }

    // add slash to end of database path string if needed
    if (!basePath.endsWith('/'))
    {
            basePath.append('/');
    }

    return SUCCESS;
}

/**
  @brief Return database location.

  This function returns as a QString the path to the QtScale database
  file such that other modules can interact with it.

  @returns The path of the database.
*/
QString Command::databasePath()
{
    return basePath + database;
}

/*** end of file ************************************************************/
```

5 commandline.h

```
/**
*@file   commandline.h
*@brief  Class definition of Command.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef COMMANDLINE_H
#define COMMANDLINE_H

#include <QString>
#include <QStringList>

#include "errorcode.h"


/**
  \brief this class encapsulates the methods and variables necesary to
  parse the commandline arguments that are supported.
*/
class Command
{
    public:
        Command();
        errorCode parseArguments(QStringList argumentList);
        QString databasePath();

    protected:
        QString basePath;   ///< Database name
        QString database;   ///< Database path
};

#endif // COMMANDLINE_H

/*** end of file ***************************************************************/
```

6 database.cpp

```
/**
*@file   database.cpp
*@brief  Class implementation of Database.
*
* This class contains the methods and members that enable
* interaction with SQLite databases.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <stdio.h>
#include <QIcon>
#include <QList>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlQueryModel>
#include <QVariant>

#include "database.h"
#include "errorcode.h"

/**
  @brief Class constructor of Database.

  This function creates an instance of Databaseand initializes
  the class member.
*/
Database::Database()
{
    db = QSqlDatabase::addDatabase("QSQLITE");
}

/**
  @brief Set database path.

  This function sets the path the SQLite database.
```

```
  @param database Path to SQLite database.
*/
void Database::setDatabase(QString database)
{
    db.setDatabaseName(database);
}

/**
  @brief Open database.

  This function opens the database to be able to interact with it.<br>
  If the SQLite database failed to open, the function returns
  databaseError error code.<br>
  Otherwise, it returns SUCCESS.

  @returns Error code, being SUCCESS if the database was opened or
  databaseError otherwise.
*/
errorCode Database::openDatabase()
{
    if (db.open())
    {
        return SUCCESS;
    }
    else
    {
        fprintf(stderr,"Cannot open database requested\n");
        return databaseError;
    }
}

/**
  @brief Close database.

  This function closes the database.
*/
void Database::closeDatabase()
{
    db.close();
}

/**
  @brief List database records.

  This function returns a QList object with all the ID numbers of the products
  in the database in ascending order.

  @returns List of product IDs.
*/
QList<int> Database::list()
{
    QList<int> vector;
    QSqlQuery query("SELECT ID FROM Products");

    while (query.next())
    {
```

```
        vector.append(query.value(0).toInt());
    }

    return vector;
}

/**
  @brief Get product icon.

  This function returns as a QIcon the icon for the specified
  product ID in the database.

  @param ID Product index number (ID field in database).

  @returns Icon of specified product.
*/
QIcon Database::icon(int ID)
{
    QSqlQuery query("SELECT␣Icon␣FROM␣Products␣WHERE␣ID=" + QString::number(ID))
        ;
    query.next();

    QByteArray array = query.value(0).toByteArray();
    QPixmap pixmap = QPixmap();
    pixmap.loadFromData(array);

    return QIcon(pixmap);
}

/**
  @brief Get product name.

  This function returns the name of the specified
  product ID in the database.

  @param ID Product index number (ID field in database).

  @returns Name of specified product.
*/
QString Database::name(int ID)
{
    QSqlQuery query("SELECT␣Name␣FROM␣Products␣WHERE␣ID=" + QString::number(ID))
        ;
    query.next();

    QString name = query.value(0).toString();

    return name;
}

/**
  @brief Get product price.

  This function returns the Price of the specified
  product ID in the database.

  @param ID Product index number (ID field in database).
```

```
  @returns Price of specified product.
*/
float Database::price(int ID)
{
    QSqlQuery query("SELECT␣Price␣FROM␣Products␣WHERE␣ID=" + QString::number(ID)
        );
    query.next();

    float price = query.value(0).toFloat();

    return price;
}

/**
  @brief Get product price.

  This function updates the accumulated sales in the product database record.

  @param weight Sale to add to accumulated sales.
  @param ID Product index number (ID field in database).
*/
void Database::updateAccum(float weight, int ID)
{
    QSqlQuery query;

    query.exec("SELECT␣Accumulated␣FROM␣Products␣WHERE␣ID=" +
               QString::number(ID));;
    query.next();
    float accum = query.value(0).toFloat();
    accum = accum + weight;

    query.exec("UPDATE␣Products␣SET␣Accumulated=" +
          QString::number(accum) + "␣WHERE␣ID=" + QString::number(ID));
    query.next();
}

/**
  @brief Reset accumulated sales.

  This function updates the accumulated sales of
  all products in the database and sets them to 0.

  This function does not take parameters as it relies on
  other Database class functions.
*/
void Database::clearAccum()
{
    QSqlQuery query;
    QList<int> index = list();

    for (int i = 0; i < index.size(); i++)
    {
        query.exec("UPDATE␣Products␣SET␣Accumulated=0.0␣WHERE␣ID="
                   + QString::number(index.at(i)));
        query.next();
    }
```

```
}

/*** end of file ************************************************************/
```

7 database.h

```
/**
*@file   database.h
*@brief  Class definition of Database.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef DATABASE_H
#define DATABASE_H

#include <QList>
#include <QIcon>
#include <QString>
#include <QSqlDatabase>
#include <QSqlQueryModel>

#include "errorcode.h"

/**
  \brief this class encapsulates the methods and variables necessary to
  manage the SQLite database.

  The database file must have a table named "Products" with the following fields
      :

  \verbatim
  -->Products Table:
  +---------------------------+-------------+-------------+--------------+--------------------+

  | ID (integer primary key) | Name (text) | Icon (blob) | Price (real) |
      Accumulated (real) |
  +===========================+=============+=============+==============+====================+

  | ID 1                      | Product 1   | PNG 1       | Price 1      | Accum
      1            |
  +---------------------------+-------------+-------------+--------------+--------------------+
```

```
  | ID 2                        | Product 2    | PNG 2        | Price 2       | Accum
     2             |
  +---------------------------+-------------+-------------+--------------+--------------------+

  | ...                         | ...          | ...          | ...           | ...
                      |
  +---------------------------+-------------+-------------+--------------+--------------------+

  \endverbatim

  For more information, please consult the project report.
*/
class Database
{
    public:
        Database();
        void setDatabase(QString database);
        errorCode openDatabase();
        void closeDatabase();
        QList<int> list();
        QIcon icon(int ID);
        QString name(int ID);
        float price(int ID);
        void updateAccum(float weight, int ID);
        void clearAccum();

protected:
        QSqlDatabase db;    ///< Database object
};


#endif // DATABASE_H

/*** end of file **************************************************************/
```

8 errorcode.h

```
/**
*@file   errorcode.h
*@brief  Definition of errorCode data type.
*
* This file contains the implementation of the errorCode enumerated data type.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef ERRORCODE_H
#define ERRORCODE_H

/**
  @brief Error code signaling data type

  This data type is used as a return in various functions to signal if the
  function exited successfully or if an error occurred. If an error occurs,
  errorCode also gives information on the error type, such that it can be
      handled
  accordingly and/or warning can be displayed/printed.
*/
enum errorCode
{
    SUCCESS,        ///< function exited successfully
    argumentError,  ///< failed to parse command line argument
    databaseError,  ///< failed to open/close database
    queryError,     ///< query failed to execute successfully
    serialError,    ///< serial connection could not be established
    scaleError      ///< scale is sending unexpected values. Calibration needed.
};

#endif // ERRORCODE_H

/*** end of file ****************************************************************/
```

9 licensedialog.cpp

```
/**
*@file   licensedialog.cpp
*@brief  Class implementation of licenseDialog.
*
* This class displays the license of the application.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include "licensedialog.h"
#include "ui_licensedialog.h"

/**
  @brief Class constructor of licenseDialog object.

  This function creates an instance of licenseDialog class
  and initializes the widget.
*/
licenseDialog::licenseDialog(QWidget *parent):QDialog(parent),
    ui(new Ui::licenseDialog)
{
    ui->setupUi(this);

    // set to fullscreen and frameless
    this->setWindowFlags(Qt::FramelessWindowHint);
    this->setFixedHeight(272);
    this->setFixedWidth(480);

    // add icon to button
    ui->closeButton->setIcon(QIcon(":/Buttons/Buttons/cancel.png"));
    ui->closeButton->setIconSize(QSize(24,24));
}

/**
  @brief Class destructor of licenseDialog object.
```

```
  This function destroys an instance of licenseDialog class.
*/
licenseDialog::~licenseDialog()
{
    delete ui;
}

/**
  @brief exit licenseDialog.

  This slot closes destroys current "license" dialog.
*/
void licenseDialog::on_closeButton_clicked()
{
    this->close();
}

/*** end of file ************************************************************/
```

10 licensedialog.h

```
/**
*@file   licensedialog.h
*@brief  Class definition of licenseDialog.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef LICENSEDIALOG_H
#define LICENSEDIALOG_H

#include <QDialog>

namespace Ui
{
    class licenseDialog;
}

/**
  \brief this class displays the license information of the application.
  The information is static and defined in the UI file.
*/
class licenseDialog : public QDialog
{
    Q_OBJECT

    public:
        explicit licenseDialog(QWidget *parent = 0);
        ~licenseDialog();

    private slots:
        void on_closeButton_clicked();

    private:
        Ui::licenseDialog *ui;
};

#endif // LICENSEDIALOG_H
```

```
/*** end of file ************************************************************/
```

11 licensedialog.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>licenseDialog</class>
 <widget class="QDialog" name="licenseDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>480</width>
    <height>272</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Dialog</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout">
   <item>
    <widget class="QTextEdit" name="licenseText">
     <property name="readOnly">
      <bool>true</bool>
     </property>
     <property name="html">
      <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot; &
          quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot
    ; /&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Sans␣Serif'; font-
    size:9pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p align=&quot;center&quot; style=&quot; margin-top:12px; margin-bottom:12px;
     margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot
    ;&gt;&lt;span style=&quot; font-family:'MS␣Shell␣Dlg␣2'; font-size:10pt;&quot
    ;&gt;        &lt;/span&gt;&lt;span style=&quot; font-size:10pt;&quot;&gt;
    Copyright (C) 2014  Pablo Cholbi Alenda&lt;/span&gt;&lt;/p&gt;
&lt;p align=&quot;center&quot; style=&quot;-qt-paragraph-type:empty; margin-top
    :0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent
    :0; text-indent:0px; font-size:10pt;&quot;&gt;&lt;br /&gt;&lt;/p&gt;
&lt;p align=&quot;justify&quot; style=&quot; margin-top:0px; margin-bottom:0px;
    margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot
    ;&gt;&lt;span style=&quot; font-size:10pt;&quot;&gt;This program is free
    software: you can redistribute it and/or modify it under the terms of the GNU
     General Public License as published by the Free Software Foundation, version
     3.&lt;/span&gt;&lt;/p&gt;
&lt;p align=&quot;justify&quot; style=&quot;-qt-paragraph-type:empty; margin-top
    :0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent
    :0; text-indent:0px; font-size:10pt;&quot;&gt;&lt;br /&gt;&lt;/p&gt;
&lt;p align=&quot;justify&quot; style=&quot; margin-top:0px; margin-bottom:0px;
    margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot
    ;&gt;&lt;span style=&quot; font-size:10pt;&quot;&gt;This program is
    distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
    PARTICULAR PURPOSE.  See the GNU General Public License for more details.&lt
    ;/span&gt;&lt;/p&gt;
&lt;p align=&quot;center&quot; style=&quot;-qt-paragraph-type:empty; margin-top
    :12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-
```

```
    indent:0; text-indent:0px; font-family:'MS␣Shell␣Dlg␣2'; font-size:10pt;&quot
    ;&gt;&lt;br /&gt;&lt;/p&gt;
&lt;p align=&quot;center&quot; style=&quot; margin-top:12px; margin-bottom:12px;
     margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot
    ;&gt;&lt;span style=&quot; font-family:'MS␣Shell␣Dlg␣2'; font-size:10pt;&quot
    ;&gt;http://www.gnu.org/licenses&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html
    &gt;</string>
     </property>
    </widget>
   </item>
   <item alignment="Qt::AlignRight">
    <widget class="QPushButton" name="closeButton">
     <property name="text">
      <string>Close</string>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
 <resources/>
 <connections>
  <connection>
   <sender>closeButton</sender>
   <signal>clicked()</signal>
   <receiver>licenseDialog</receiver>
   <slot>close()</slot>
   <hints>
    <hint type="sourcelabel">
     <x>328</x>
     <y>197</y>
    </hint>
    <hint type="destinationlabel">
     <x>189</x>
     <y>109</y>
    </hint>
   </hints>
  </connection>
 </connections>
</ui>
```

12 main.cpp

```
/**
*@file   main.cpp
*@brief  Start QtScale application.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <QApplication>
#include <QStringList>
#include <QProcess>

#include "mainwindow.h"

/**
  @brief Start application.

  @param argc Argument count.
  @param argv Argument vector.

  This function starts the application, creates the main window and
  handles the the restart of the application if requested by a child.

  @returns Exit status of the application.
*/
int main(int argc, char *argv[])
{
    QApplication Scale(argc, argv);
    Scale.setOverrideCursor(Qt::BlankCursor);    // Hide cursor

    QCoreApplication::setApplicationName("QtScale");
    QCoreApplication::setApplicationVersion(QString::number(appVersion,'f',2));

    MainWindow w;
    w.setup(&Scale);
    w.show();

    int exitCode = Scale.exec();
```

```
    if (exitCode == RESTART)
    {
        QString app = qApp->arguments()[0];
        QStringList arg = qApp->arguments();

        // remove program form argument list and add "-qws"
        arg.replace(0,"-qws");

        // start new instance and exit
        QProcess::startDetached(app, arg);
        return 0;
    }
    return exitCode;
}

/*** end of file **************************************************************/
```

13 mainwindow.cpp

```
/**
*@file   mainwindow.cpp
*@brief  Class implementation of MainWindow.
*
* This class displays the main product selection window.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <QApplication>
#include <QString>
#include <QStringList>
#include <QList>
#include <cstdlib>

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "errorcode.h"
#include "commandline.h"
#include "database.h"
#include "optionsdialog.h"
#include "productdialog.h"
#include "tabledialog.h"
#include "productstruct.h"

/**
  @brief Class constructor of MainWindow object.

  This function creates an instance of MainWindow class,
  initializes the class variables and the widget.
*/
MainWindow::MainWindow(QWidget *parent):QMainWindow(parent),ui(new Ui::
    MainWindow)
{
    ui->setupUi(this);
```

```
    // set to fullscreen and frameless
    this->setWindowFlags(Qt::FramelessWindowHint);
    this->setFixedHeight(272);
    this->setFixedWidth(480);

    // initialize variables
    page = 0;

    // add icons to buttons
    ui->backButton->setIcon(QIcon(":/Buttons/Buttons/previous.png"));
    ui->backButton->setIconSize(QSize(32,32));
    ui->forwardButton->setIcon(QIcon(":/Buttons/Buttons/next.png"));
    ui->forwardButton->setIconSize(QSize(32,32));
    ui->optionsButton->setIcon(QIcon(":/Buttons/Buttons/options.png"));
    ui->optionsButton->setIconSize(QSize(32,32));
    ui->tableButton->setIcon(QIcon(":/Buttons/Buttons/calculator.png"));
    ui->tableButton->setIconSize(QSize(32,32));
}

/**
  @brief Class destructor of MainWindow object.

  This function destroys an instance of MainWindow class.
*/
MainWindow::~MainWindow()
{
    delete ui;
}

/**
  @brief Setup main window.

  This function attempts to parse incoming command line arguments and
  open the database specified by these arguments (or default).
  Application exits with fail status if any of these actions fail.

  This function must be called after the class object has been created
  and before the dialog is executed.

  @param *a Pointer to QApplication object.
*/
void MainWindow::setup(QApplication *a)
{
    // parse arguments
    Command cmd;
    if (cmd.parseArguments(a->arguments()) != SUCCESS)
    {
        exit(EXIT_FAILURE);
    }

    // open database
    db.setDatabase(cmd.databasePath());
    if (db.openDatabase() != SUCCESS)
    {
        exit(EXIT_FAILURE);
    }
```

```
    vector = db.list();
    redraw();
}

/**
  @brief Redraw main window.

  This function updates the main window interface according to the information
  recorded on the database and the current "page" that is being viewed. Up to
  6 products can be viewed at a time.
*/
void MainWindow::redraw()
{
    // show or hide "back button"
    if (page == 0)
    {
        ui->backButton->setVisible(false);
    }
    else
    {
        ui->backButton->setVisible(true);
    }

    // show or hide "forward button"
    if (vector.size() < (page + 1)*6)
    {
        ui->forwardButton->setVisible(false);
    }
    else
    {
        ui->forwardButton->setVisible(true);
    }

    // set up toolButtons 1
    if((page*6 + 0) < vector.size())
    {
        ui->toolButton_1->setVisible(true);

        int ID = vector.at(page*6 + 0);
        ui->toolButton_1->setIcon(db.icon(ID));
        ui->toolButton_1->setIconSize(QSize(75, 75));
        ui->toolButton_1->setToolButtonStyle(Qt::ToolButtonTextUnderIcon);
        ui->toolButton_1->setText(QString::number(ID) + "␣-␣" + db.name(ID));
    }
    else
    {
        ui->toolButton_1->setVisible(false);
    }

    // set up toolButtons 2
    if((page*6 + 1) < vector.size())
    {
        ui->toolButton_2->setVisible(true);

        int ID = vector.at(page*6 + 1);
        ui->toolButton_2->setIcon(db.icon(ID));
        ui->toolButton_2->setIconSize(QSize(75, 75));
```

```
    ui->toolButton_2->setToolButtonStyle(Qt::ToolButtonTextUnderIcon);
    ui->toolButton_2->setText(QString::number(ID) + "␣-␣" + db.name(ID));
}
else
{
    ui->toolButton_2->setVisible(false);
}

// set up toolButtons 3
if((page*6 + 2) < vector.size())
{
    ui->toolButton_3->setVisible(true);

    int ID = vector.at(page*6 + 2);
    ui->toolButton_3->setIcon(db.icon(ID));
    ui->toolButton_3->setIconSize(QSize(75, 75));
    ui->toolButton_3->setToolButtonStyle(Qt::ToolButtonTextUnderIcon);
    ui->toolButton_3->setText(QString::number(ID) + "␣-␣" + db.name(ID));
}
else
{
    ui->toolButton_3->setVisible(false);
}

// set up toolButtons 4
if((page*6 + 3) < vector.size())
{
    ui->toolButton_4->setVisible(true);

    int ID = vector.at(page*6 + 3);
    ui->toolButton_4->setIcon(db.icon(ID));
    ui->toolButton_4->setIconSize(QSize(75, 75));
    ui->toolButton_4->setToolButtonStyle(Qt::ToolButtonTextUnderIcon);
    ui->toolButton_4->setText(QString::number(ID) + "␣-␣" + db.name(ID));
}
else
{
    ui->toolButton_4->setVisible(false);
}

// set up toolButtons 5
if((page*6 + 4) < vector.size())
{
    ui->toolButton_5->setVisible(true);

    int ID = vector.at(page*6 + 4);
    ui->toolButton_5->setIcon(db.icon(ID));
    ui->toolButton_5->setIconSize(QSize(75, 75));
    ui->toolButton_5->setToolButtonStyle(Qt::ToolButtonTextUnderIcon);
    ui->toolButton_5->setText(QString::number(ID) + "␣-␣" + db.name(ID));
}
else
{
    ui->toolButton_5->setVisible(false);
}

// set up toolButtons 6
```

```
    if((page*6 + 5) < vector.size())
    {
        ui->toolButton_6->setVisible(true);

        int ID = vector.at(page*6 + 5);
        ui->toolButton_6->setIcon(db.icon(ID));
        ui->toolButton_6->setIconSize(QSize(75, 75));
        ui->toolButton_6->setToolButtonStyle(Qt::ToolButtonTextUnderIcon);
        ui->toolButton_6->setText(QString::number(ID) + "␣-␣" + db.name(ID));
    }
    else
    {
        ui->toolButton_6->setVisible(false);
    }
}

/**
  @brief Open product page for product in button 1.

  This slot closes extracts product information from the database
  and starts a productDialog to weigh the product and complete a sale.
  Once the productDialog exits, the accumulated sales of the selected
  product is updated in the database.
*/
void MainWindow::on_toolButton_1_clicked()
{
    item.weight = 0.0;

    int ID = vector.at(page*6 + 0);
    item.name = db.name(ID);
    item.icon = db.icon(ID);
    item.price = db.price(ID);
    item.id = ID;

    product = new productDialog();
    if (product->setup(&item))
    {
        if (product->exec())
        {
            db.updateAccum(item.weight, ID);
        }
    }
    delete product;
}

/**
  @brief Open product page for product in button 2.

  This slot closes extracts product information from the database
  and starts a productDialog to weigh the product and complete a sale.
  Once the productDialog exits, the accumulated sales of the selected
  product is updated in the database.
*/
void MainWindow::on_toolButton_2_clicked()
{
    item.weight = 0.0;
```

```
    int ID = vector.at(page*6 + 1);
    item.name = db.name(ID);
    item.icon = db.icon(ID);
    item.price = db.price(ID);
    item.id = ID;

    product = new productDialog();
    if (product->setup(&item))
    {
        if (product->exec())
        {
            db.updateAccum(item.weight, ID);
        }
    }
    delete product;
}

/**
  @brief Open product page for product in button 3.

  This slot closes extracts product information from the database
  and starts a productDialog to weigh the product and complete a sale.
  Once the productDialog exits, the accumulated sales of the selected
  product is updated in the database.
*/
void MainWindow::on_toolButton_3_clicked()
{
    item.weight = 0.0;

    int ID = vector.at(page*6 + 2);
    item.name = db.name(ID);
    item.icon = db.icon(ID);
    item.price = db.price(ID);
    item.id = ID;

    product = new productDialog();
    if (product->setup(&item))
    {
        if (product->exec())
        {
            db.updateAccum(item.weight, ID);
        }
    }
    delete product;
}

/**
  @brief Open product page for product in button 4.

  This slot closes extracts product information from the database
  and starts a productDialog to weigh the product and complete a sale.
  Once the productDialog exits, the accumulated sales of the selected
  product is updated in the database.
*/
void MainWindow::on_toolButton_4_clicked()
{
    item.weight = 0.0;
```

```
    int ID = vector.at(page*6 + 3);
    item.name = db.name(ID);
    item.icon = db.icon(ID);
    item.price = db.price(ID);
    item.id = ID;

    product = new productDialog();
    if (product->setup(&item))
    {
        if (product->exec())
        {
            db.updateAccum(item.weight, ID);
        }
    }
    delete product;
}

/**
  @brief Open product page for product in button 5.

  This slot closes extracts product information from the database
  and starts a productDialog to weigh the product and complete a sale.
  Once the productDialog exits, the accumulated sales of the selected
  product is updated in the database.
*/
void MainWindow::on_toolButton_5_clicked()
{
    item.weight = 0.0;

    int ID = vector.at(page*6 + 4);
    item.name = db.name(ID);
    item.icon = db.icon(ID);
    item.price = db.price(ID);
    item.id = ID;

    product = new productDialog();
    if (product->setup(&item))
    {
        if (product->exec())
        {
            db.updateAccum(item.weight, ID);
        }
    }
    delete product;
}

/**
  @brief Open product page for product in button 6.

  This slot closes extracts product information from the database
  and starts a productDialog to weigh the product and complete a sale.
  Once the productDialog exits, the accumulated sales of the selected
  product is updated in the database.
*/
void MainWindow::on_toolButton_6_clicked()
{
```

```
    item.weight = 0.0;

    int ID = vector.at(page*6 + 5);
    item.name = db.name(ID);
    item.icon = db.icon(ID);
    item.price = db.price(ID);
    item.id = ID;

    product = new productDialog();
    if (product->setup(&item))
    {
        if (product->exec())
        {
            db.updateAccum(item.weight, ID);
        }
    }
    delete product;
}

/**
  @brief Go to previous page.

  This slot changes the current page and forces the GUI to update.
*/
void MainWindow::on_backButton_clicked()
{
    page--;
    redraw();
}

/**
  @brief Go to next page.

  This slot changes the current page and forces the GUI to update.
*/
void MainWindow::on_forwardButton_clicked()
{
    page++;
    redraw();
}

/**
  @brief Go to options dialog.

  This slot closes the database and creates an option dialog.<br>
  If the user exits from the options dialog,
  the database is attempted to be reopened.
*/
void MainWindow::on_optionsButton_clicked()
{
    db.closeDatabase();
    options = new optionsDialog();
    options->exec();
    if (db.openDatabase() != SUCCESS)
    {
        exit(EXIT_FAILURE);
    }
```

```
    delete options;
}

/**
  @brief Go to table dialog.

  This slot creates a tableDialog and displays the accumulated sales of
  all products in the database.

  Depending on the return value of the dialog,
  the accumulates sales of all products in the database are cleared.
*/
void MainWindow::on_tableButton_clicked()
{
    table = new tableDialog();
    // show dialog and clear accumulated sales if requested
    if (table->exec())
    {
        db.clearAccum();
    }
    delete table;
}

/*** end of file ************************************************************/
```

14 mainwindow.h

```
/**
*@file   mainwindow.h
*@brief  Class definition of MainWindow.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QApplication>
#include <QList>

#include "commandline.h"
#include "database.h"
#include "optionsdialog.h"
#include "productdialog.h"
#include "tabledialog.h"
#include "productstruct.h"

namespace Ui
{
    class MainWindow;
}

/**
  \brief this class encapsulates the methods and variables necesary to
  display the main product selection window.
*/
class MainWindow : public QMainWindow
{
    Q_OBJECT

    public:
        explicit MainWindow(QWidget *parent = 0);
        ~MainWindow();
```

```
        void setup(QApplication *a);
        void redraw();

    protected:
        int page;           ///< Current page being displayed
        Database db;        ///< Product database file
        QList<int> vector;  ///< ID numbers of the products
        productStruct item; ///< Structure with product information

    private slots:
        void on_toolButton_1_clicked();
        void on_toolButton_2_clicked();
        void on_toolButton_3_clicked();
        void on_toolButton_4_clicked();
        void on_toolButton_5_clicked();
        void on_toolButton_6_clicked();
        void on_backButton_clicked();
        void on_forwardButton_clicked();
        void on_optionsButton_clicked();
        void on_tableButton_clicked();

    private:
        Ui::MainWindow *ui;
        optionsDialog *options;
        productDialog *product;
        tableDialog *table;
};

#endif // MAINWINDOW_H

/*** end of file ************************************************************/
```

15 mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>480</width>
    <height>272</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <widget class="QPushButton" name="optionsButton">
    <property name="geometry">
     <rect>
      <x>430</x>
      <y>222</y>
      <width>40</width>
      <height>40</height>
     </rect>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QPushButton" name="backButton">
    <property name="geometry">
     <rect>
      <x>10</x>
      <y>116</y>
      <width>40</width>
      <height>40</height>
     </rect>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QPushButton" name="forwardButton">
    <property name="geometry">
     <rect>
      <x>430</x>
      <y>116</y>
      <width>40</width>
      <height>40</height>
     </rect>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
```

```
<widget class="QToolButton" name="toolButton_3">
 <property name="geometry">
  <rect>
   <x>305</x>
   <y>24</y>
   <width>100</width>
   <height>100</height>
  </rect>
 </property>
 <property name="text">
  <string/>
 </property>
</widget>
<widget class="QToolButton" name="toolButton_1">
 <property name="geometry">
  <rect>
   <x>75</x>
   <y>24</y>
   <width>100</width>
   <height>100</height>
  </rect>
 </property>
 <property name="text">
  <string/>
 </property>
</widget>
<widget class="QToolButton" name="toolButton_5">
 <property name="geometry">
  <rect>
   <x>190</x>
   <y>148</y>
   <width>100</width>
   <height>100</height>
  </rect>
 </property>
 <property name="text">
  <string/>
 </property>
</widget>
<widget class="QToolButton" name="toolButton_2">
 <property name="geometry">
  <rect>
   <x>190</x>
   <y>24</y>
   <width>100</width>
   <height>100</height>
  </rect>
 </property>
 <property name="text">
  <string/>
 </property>
</widget>
<widget class="QToolButton" name="toolButton_4">
 <property name="geometry">
  <rect>
   <x>75</x>
   <y>148</y>
```

```
      <width>100</width>
      <height>100</height>
     </rect>
    </property>
    <property name="text">
     <string/>
    </property>
    <property name="checkable">
     <bool>false</bool>
    </property>
   </widget>
   <widget class="QToolButton" name="toolButton_6">
    <property name="geometry">
     <rect>
      <x>305</x>
      <y>148</y>
      <width>100</width>
      <height>100</height>
     </rect>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QPushButton" name="tableButton">
    <property name="geometry">
     <rect>
      <x>10</x>
      <y>222</y>
      <width>40</width>
      <height>40</height>
     </rect>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
  </widget>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```

16 messagedialog.cpp

```
/**
*@file   messagedialog.cpp
*@brief  Class implementation of messageDialog.
*
* This class displays error and information messages.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <stdio.h>
#include <QString>

#include "messagedialog.h"
#include "ui_messagedialog.h"
#include "errorcode.h"

/**
  @brief Class constructor of messageDialog object.

  This function creates an instance of messageDialog class
  and initializes the widget.
*/
messageDialog::messageDialog(QWidget *parent):QDialog(parent),
    ui(new Ui::messageDialog)
{
    ui->setupUi(this);

    // set to fullscreen and cursor-less
    this->setWindowFlags(Qt::FramelessWindowHint);
    this->setFixedHeight(272);
    this->setFixedWidth(480);
}

/**
  @brief Class destructor of messageDialog object.
```

```
  This function destroys an instance of messageDialog class.
*/
messageDialog::~messageDialog()
{
    delete ui;
}

/**
  @brief Show error message.

  This function displays an error message for the defined errors in errorCode.

  @param error Error to me informed of.
*/
void messageDialog::errorMessage(errorCode error)
{
    ui->Title->setText("Error");
    switch (error)
    {
        case argumentError:
            ui->Message->setText("Unknown command line argument");
            fprintf(stderr,"Error: Unknown command line argument\n");
            break;
        case databaseError:
            ui->Message->setText("Cannot open requested database");
            fprintf(stderr,"Error: Cannot open requested database\n");
            break;
        case queryError:
            ui->Message->setText("Cannot open requested database");
            fprintf(stderr,"Error: Cannot open requested database\n");
            break;
        case serialError:
            ui->Message->setText("Cannot establish serial connection");
            fprintf(stderr,"Error: Cannot establish serial connection\n");
            break;
        case scaleError:
            ui->Message->setText("Scale needs calibration");
            fprintf(stderr,"Error: Scale needs calibration\n");
            break;
        default:
            ui->Message->setText("An unexpected error occurred");
            fprintf(stderr,"Error: An unexpected error occurred\n");
            break;
    }
    ui->pushButton->setText("Acknowledge");
    ui->pushButton->setIcon(QIcon(":/Buttons/Buttons/warning.png"));
    ui->pushButton->setIconSize(QSize(24,24));
}

/**
  @brief Show generic message.

  This function displays a generic message on screen.

  @param msg Message to be displayed.
*/
void messageDialog::genericMessage(QString msg)
```

```
{
    ui->Title->setText("Information");
    ui->Message->setText(msg);
    ui->pushButton->setText("Acknowledge");
    ui->pushButton->setIcon(QIcon(":/Buttons/Buttons/warning.png"));
    ui->pushButton->setIconSize(QSize(24,24));
}

/**
  @brief Acknowledge message.

  This slot closes destroys current "message" dialog.
*/
void messageDialog::on_pushButton_clicked()
{
    this->close();
}

/*** end of file ****************************************************************/
```

17 messagedialog.h

```
/**
*@file   messagedialog.h
*@brief  MessageDialog class definition.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef MESSAGEDIALOG_H
#define MESSAGEDIALOG_H

#include <QDialog>

#include "errorcode.h"

namespace Ui
{
    class messageDialog;
}

/**
  \brief this class encapsulates the methods and variables necesary to
  display error and information messages.
*/
class messageDialog : public QDialog
{
    Q_OBJECT

    public:
        explicit messageDialog(QWidget *parent = 0);
        ~messageDialog();
        void errorMessage(errorCode error);
        void genericMessage(QString msg);

    private slots:
        void on_pushButton_clicked();

    private:
```

```
        Ui::messageDialog *ui;
};

#endif // MESSAGEDIALOG_H

/*** end of file ************************************************************/
```

18 messagedialog.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>messageDialog</class>
 <widget class="QDialog" name="messageDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>480</width>
    <height>272</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Form</string>
  </property>
  <widget class="QPushButton" name="pushButton">
   <property name="geometry">
    <rect>
     <x>170</x>
     <y>202</y>
     <width>140</width>
     <height>50</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <pointsize>14</pointsize>
    </font>
   </property>
   <property name="text">
    <string/>
   </property>
  </widget>
  <widget class="QLabel" name="Title">
   <property name="geometry">
    <rect>
     <x>160</x>
     <y>30</y>
     <width>160</width>
     <height>35</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <pointsize>28</pointsize>
    </font>
   </property>
   <property name="locale">
    <locale language="English" country="UnitedKingdom"/>
   </property>
   <property name="text">
    <string>Title</string>
   </property>
   <property name="alignment">
    <set>Qt::AlignCenter</set>
```

```
   </property>
  </widget>
  <widget class="QLabel" name="Message">
   <property name="geometry">
    <rect>
     <x>15</x>
     <y>100</y>
     <width>450</width>
     <height>30</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <pointsize>14</pointsize>
    </font>
   </property>
   <property name="locale">
    <locale language="English" country="UnitedKingdom"/>
   </property>
   <property name="text">
    <string>Message</string>
   </property>
   <property name="alignment">
    <set>Qt::AlignCenter</set>
   </property>
  </widget>
 </widget>
 <resources/>
 <connections/>
</ui>
```

19 optionsdialog.cpp

```
/**
*@file   optionsdialog.cpp
*@brief  Class implementation of optionsDialog.
*
* This class displays and executes configuration options.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <QProcess>

#include "optionsdialog.h"
#include "ui_optionsdialog.h"
#include "aboutdialog.h"

/**
  @brief Class constructor of optionDialog object.

  This function creates an instance of optionsDialog class
  and initializes the widget.
*/
optionsDialog::optionsDialog(QWidget *parent):QDialog(parent),
    ui(new Ui::optionsDialog)
{
    ui->setupUi(this);

    // set to fullscreen and frameless
    this->setWindowFlags(Qt::FramelessWindowHint);
    this->setFixedHeight(272);
    this->setFixedWidth(480);

    // add icons to buttons
    ui->restartButton->setIcon(QIcon(":/Buttons/Buttons/executable.png"));
    ui->restartButton->setIconSize(QSize(24,24));
    ui->rebootButton->setIcon(QIcon(":/Buttons/Buttons/refresh.png"));
    ui->rebootButton->setIconSize(QSize(24,24));
```

```
    ui->shutdownButton->setIcon(QIcon(":/Buttons/Buttons/shutdown.png"));
    ui->shutdownButton->setIconSize(QSize(24,24));
    ui->exitButton->setIcon(QIcon(":/Buttons/Buttons/exit.png"));
    ui->exitButton->setIconSize(QSize(24,24));
    ui->aboutButton->setIcon(QIcon(":/Buttons/Buttons/about.png"));
    ui->aboutButton->setIconSize(QSize(24,24));
    ui->closeButton->setIcon(QIcon(":/Buttons/Buttons/cancel.png"));
    ui->closeButton->setIconSize(QSize(24,24));
}
/**
  @brief Class destructor of optionsDialog object.

  This function destroys an instance of optionsDialog class.
*/
optionsDialog::~optionsDialog()
{
    delete ui;
}

/**
  @brief Restart application.

  This slot makes QApplication exit with "restart" code such that the main
  function can handle it.
*/
void optionsDialog::on_restartButton_clicked()
{
    qApp->exit(RESTART);
}

/**
  @brief Exit application.

  This slot makes QApplication exit with with exit code "0", or
  "successful termination".
*/
void optionsDialog::on_exitButton_clicked()
{
    qApp->quit();
}

/**
  @brief Reboot system.

  This slot makes starts an external process to reboot the system.

  \verbatim
  shutdown -r now
  \endverbatim
*/
void optionsDialog::on_rebootButton_clicked()
{
    QProcess::execute("shutdown -r now");
}

/**
  @brief Shutdown system.
```

```
  This slot makes starts an external process to halt the system.

  \verbatim
  shutdown -h now
  \endverbatim
*/
void optionsDialog::on_shutdownButton_clicked()
{
    QProcess::execute("shutdown -h now");
}

/**
  @brief open aboutDialog.

  This slot opens the an "About" dialog.
*/
void optionsDialog::on_aboutButton_clicked()
{
    about = new aboutDialog();
    about->exec();
    delete about;
}

/**
  @brief exit optionsDialog.

  This slot closes destroys current "options" dialog.
*/
void optionsDialog::on_closeButton_clicked()
{
    this->close();
}

/*** end of file ************************************************************/
```

20 optionsdialog.h

```
/**
*@file   optionsdialog.h
*@brief  Class definition of optionsDialog.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef OPTIONSDIALOG_H
#define OPTIONSDIALOG_H

#include <QDialog>

#include "aboutdialog.h"

namespace Ui
{
    class optionsDialog;
}

/**
  \brief this class encapsulates the methods and variables necesary to
  display and execute configuration options.
*/
class optionsDialog : public QDialog
{
    Q_OBJECT

    public:
        explicit optionsDialog(QWidget *parent = 0);
        ~optionsDialog();

    private slots:
        void on_restartButton_clicked();
        void on_exitButton_clicked();
        void on_rebootButton_clicked();
        void on_shutdownButton_clicked();
        void on_aboutButton_clicked();
```

```
        void on_closeButton_clicked();

    private:
        Ui::optionsDialog *ui;
        aboutDialog *about;
};

#endif // OPTIONSDIALOG_H

/*** end of file ************************************************************/
```

21 optionsdialog.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>optionsDialog</class>
 <widget class="QDialog" name="optionsDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>480</width>
    <height>272</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Dialog</string>
  </property>
  <widget class="QPushButton" name="closeButton">
   <property name="geometry">
    <rect>
     <x>370</x>
     <y>227</y>
     <width>90</width>
     <height>35</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <pointsize>10</pointsize>
    </font>
   </property>
   <property name="text">
    <string>Close</string>
   </property>
  </widget>
  <widget class="QWidget" name="verticalLayoutWidget">
   <property name="geometry">
    <rect>
     <x>249</x>
     <y>10</y>
     <width>211</width>
     <height>131</height>
    </rect>
   </property>
   <layout class="QVBoxLayout" name="verticalLayout_2">
    <item>
     <widget class="QPushButton" name="rebootButton">
      <property name="font">
       <font>
        <pointsize>10</pointsize>
       </font>
      </property>
      <property name="layoutDirection">
       <enum>Qt::LeftToRight</enum>
      </property>
      <property name="text">
       <string>Reboot System</string>
```

```
    </property>
   </widget>
  </item>
  <item>
   <widget class="QPushButton" name="shutdownButton">
    <property name="font">
     <font>
      <pointsize>10</pointsize>
     </font>
    </property>
    <property name="layoutDirection">
     <enum>Qt::LeftToRight</enum>
    </property>
    <property name="text">
     <string>Shutdown System</string>
    </property>
   </widget>
  </item>
 </layout>
</widget>
<widget class="QWidget" name="verticalLayoutWidget_2">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>10</y>
   <width>221</width>
   <height>131</height>
  </rect>
 </property>
 <layout class="QVBoxLayout" name="verticalLayout_1">
  <item>
   <widget class="QPushButton" name="restartButton">
    <property name="font">
     <font>
      <pointsize>10</pointsize>
     </font>
    </property>
    <property name="layoutDirection">
     <enum>Qt::LeftToRight</enum>
    </property>
    <property name="text">
     <string>Restart QtScale</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QPushButton" name="exitButton">
    <property name="font">
     <font>
      <pointsize>10</pointsize>
     </font>
    </property>
    <property name="layoutDirection">
     <enum>Qt::LeftToRight</enum>
    </property>
    <property name="text">
     <string>Exit QtScale</string>
```

```
      </property>
     </widget>
    </item>
   </layout>
  </widget>
  <widget class="QPushButton" name="aboutButton">
   <property name="geometry">
    <rect>
     <x>20</x>
     <y>227</y>
     <width>90</width>
     <height>35</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <pointsize>10</pointsize>
    </font>
   </property>
   <property name="text">
    <string>About</string>
   </property>
  </widget>
 </widget>
 <resources/>
 <connections/>
</ui>
```

22 productdialog.cpp

```
/**
*@file   productdialog.cpp
*@brief  Class implementation of productDialog.
*
* This class displays product information and weighs the product.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <QTimer>
#include <QIcon>
#include <QString>

#include "productdialog.h"
#include "ui_productdialog.h"
#include "productstruct.h"
#include "serialscale.h"
#include "messagedialog.h"
#include "errorcode.h"

QString currencyUnit = QString::fromUtf8("   "); ///< Symbol for currency unit
QString massUnit = QString("kg");               ///< Symbol for weight unit


/**
  @brief Class constructor of productDialog object.

  This function creates an instance of productDialog class
  and initializes the widget.
*/
productDialog::productDialog(QWidget *parent):QDialog(parent),
    ui(new Ui::productDialog)
{
    ui->setupUi(this);

    // set to fullscreen and frameless
```

```
    this->setWindowFlags(Qt::FramelessWindowHint);
    this->setFixedHeight(272);
    this->setFixedWidth(480);

    // add icons to buttons
    ui->buyButton->setIcon(QIcon(":/Buttons/Buttons/emblem-default.png"));
    ui->buyButton->setIconSize(QSize(32,32));
    ui->cancelButton->setIcon(QIcon(":/Buttons/Buttons/cancel.png"));
    ui->cancelButton->setIconSize(QSize(32,32));
}

/**
  @brief Class destructor of productDialog object.

  This function destroys an instance of productDialog class.
*/
productDialog::~productDialog()
{
    delete serial;
    delete ui;
}

/**
  @brief Product dialog.

  This function initializes the widget elements according to the productStruct
  and prepares the serial connections to weigh the product.

  This function must be called after the class object has been created
  and the dialog should only be executed if the setup exits successfully.

  @param *item Pointer to productStruct object.
  @returns TRUE is setup exited successfully or FALSE otherwise.
*/
bool productDialog::setup(productStruct *item)
{
    product = item;

    // set product icon
    ui->iconLabel->setPixmap(product->icon.pixmap(150, 150,
                                                  QIcon::Normal, QIcon::On));

    //set ID, name and price labels
    ui->idLabel->setText(QString::number(product->id));
    ui->nameLabel->setText(product->name);
    ui->priceLabel->setText(QString::number(product->price,'f', 2)
                            + "␣" + currencyUnit + "/" + massUnit);

    // set initial value for remaining labels
    ui->weightLabel->setText("0.000␣" + massUnit);
    ui->Cost->setText("0.00␣" + currencyUnit);

    serial = new serialScale();
    errorCode error = serial->openConnection();
    if (error != SUCCESS)
    {
        message = new messageDialog();
```

```
        message->errorMessage(error);
        message->exec();
        delete message;
        return false;
    }

    timer = new QTimer(this);
    connect(timer, SIGNAL(timeout()), this, SLOT(update()));
    timer->start(200);
    return true;
}

/**
  @brief Update values.

  This function makes the serial object evaluate incoming data and
  updates the weight and cost values accordingly.
*/
void productDialog::update()
{
    serial->update();
    product->weight = serial->getWeight();

    ui->weightLabel->setText(QString::number(product->weight, 'f', 3)
                             + "␣" + massUnit);
    ui->Cost->setText(QString::number(product->weight*product->price, 'f', 2)
                      + "␣" + currencyUnit);

    // Loopback test
        /* Attention: please read documentation */
        /* Do NOT uncomment if unsure of what you are doing */
    //serial->loopbackTest();
}

/**
  @brief Complete sale.

  This slot makes the productDialog execution exits with value "true"
  and closes the dialog.

  Another module must then process the return value.
*/
void productDialog::on_buyButton_clicked()
{
    // if weight is stable, update weight and close
    if (serial->isStable())
    {
        product->weight = serial->getWeight();
        delete timer;
        this->accept();
        this->close();
    }
}

/**
  @brief Cancel.
```

```
  This slot makes the productDialog execution exits with value "false"
  and closes the dialog.

  Another module must then process the return value.
*/
void productDialog::on_cancelButton_clicked()
{
    delete timer;
    this->reject();
    this->close();
}

/*** end of file ***************************************************************/
```

23 productdialog.h

```
/**
*@file   productdialog.h
*@brief  Class definition of productDialog.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef PRODUCTDIALOG_H
#define PRODUCTDIALOG_H

#include <QDialog>
#include <QTimer>

#include "productstruct.h"
#include "serialscale.h"
#include "productstruct.h"

namespace Ui
{
    class productDialog;
}

/**
  \brief this class encapsulates the methods and variables necessary to
  display product weight and information.
*/
class productDialog : public QDialog
{
    Q_OBJECT

    public:
        explicit productDialog(QWidget *parent = 0);
        ~productDialog();
        bool setup(productStruct *item);

    private slots:
            void on_buyButton_clicked();
```

```
            void on_cancelButton_clicked();
            void update();

    private:
            Ui::productDialog *ui;
            serialScale *serial;
            productStruct *product;
            messageDialog *message;
            QTimer *timer;
};

#endif // PRODUCTDIALOG_H

/*** end of file ************************************************************/
```

24 productdialog.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>productDialog</class>
 <widget class="QDialog" name="productDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>480</width>
    <height>272</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Dialog</string>
  </property>
  <widget class="QLabel" name="iconLabel">
   <property name="geometry">
    <rect>
     <x>20</x>
     <y>20</y>
     <width>150</width>
     <height>150</height>
    </rect>
   </property>
   <property name="text">
    <string/>
   </property>
  </widget>
  <widget class="QPushButton" name="buyButton">
   <property name="geometry">
    <rect>
     <x>360</x>
     <y>202</y>
     <width>100</width>
     <height>50</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <pointsize>12</pointsize>
    </font>
   </property>
   <property name="text">
    <string>Buy</string>
   </property>
  </widget>
  <widget class="QPushButton" name="cancelButton">
   <property name="geometry">
    <rect>
     <x>20</x>
     <y>202</y>
     <width>100</width>
     <height>50</height>
    </rect>
   </property>
```

```
 <property name="font">
  <font>
   <pointsize>12</pointsize>
  </font>
 </property>
 <property name="text">
  <string>Cancel</string>
 </property>
</widget>
<widget class="QLabel" name="idTextLabel">
 <property name="geometry">
  <rect>
   <x>260</x>
   <y>20</y>
   <width>90</width>
   <height>30</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <family>Sans Serif</family>
   <pointsize>12</pointsize>
   <weight>50</weight>
   <bold>false</bold>
  </font>
 </property>
 <property name="text">
  <string>Product ID:</string>
 </property>
</widget>
<widget class="QLabel" name="nameLabel">
 <property name="geometry">
  <rect>
   <x>260</x>
   <y>50</y>
   <width>200</width>
   <height>30</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <pointsize>18</pointsize>
  </font>
 </property>
 <property name="text">
  <string>name</string>
 </property>
</widget>
<widget class="QLabel" name="priceLabel">
 <property name="geometry">
  <rect>
   <x>310</x>
   <y>80</y>
   <width>150</width>
   <height>30</height>
  </rect>
 </property>
```

```
 <property name="font">
  <font>
   <pointsize>12</pointsize>
  </font>
 </property>
 <property name="text">
  <string>price</string>
 </property>
</widget>
<widget class="QLabel" name="idLabel">
 <property name="geometry">
  <rect>
   <x>350</x>
   <y>20</y>
   <width>110</width>
   <height>30</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <pointsize>12</pointsize>
  </font>
 </property>
 <property name="text">
  <string>id</string>
 </property>
</widget>
<widget class="QLabel" name="weightLabel">
 <property name="geometry">
  <rect>
   <x>310</x>
   <y>110</y>
   <width>150</width>
   <height>30</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <pointsize>12</pointsize>
  </font>
 </property>
 <property name="text">
  <string>weight</string>
 </property>
</widget>
<widget class="QLabel" name="Cost">
 <property name="geometry">
  <rect>
   <x>260</x>
   <y>150</y>
   <width>200</width>
   <height>40</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <pointsize>24</pointsize>
```

```
    </font>
   </property>
   <property name="text">
    <string>purchase cost</string>
   </property>
  </widget>
 </widget>
 <resources/>
 <connections/>
</ui>
```

25 productstruct.h

```
/**
*@file   productstruct.h
*@brief  Data structure type productStruct.
*
* This file contains the implementation of the productStruct data structure.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef PRODUCTSTRUCT_H
#define PRODUCTSTRUCT_H

#include <QIcon>
#include <QString>

#include "messagedialog.h"

/**
  @brief Product data structure

  This data structure encapsulated variables to be used in the product dialog.
*/
struct productStruct
{
   QString  name;   ///< Name of product.
   QIcon    icon;   ///< Icon of product.
   int      id;     ///< Id of product.
   float    price;  ///< Product price per unit of weight.
   float    weight; ///< Weight received from the scale.
};

#endif // PRODUCTSTRUCT_H

/*** end of file ************************************************************/
```

26 QtScale.pro

```
#-------------------------------------------------#
#                                                 #
# Project created by Pablo Cholbi 2014-02-27      #
#                                                 #
#-------------------------------------------------#

#-------------------------------------------------------------------------------#
#   This file is part of QtScale.                                               #
#                                                                               #
#   QtScale is free software: you can redistribute it and/or modify             #
#   it under the terms of the GNU General Public License as published by        #
#   the Free Software Foundation, version 3.                                    #
#                                                                               #
#   QtScale is distributed in the hope that it will be useful,                  #
#   but WITHOUT ANY WARRANTY; without even the implied warranty of              #
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               #
#   GNU General Public License for more details.                                #
#                                                                               #
#   Copyright (C) 2014  Pablo Cholbi Alenda.                                    #
#                                                                               #
#   You should have received a copy of the GNU General Public License           #
#   along with QtScale.  If not, see <http://www.gnu.org/licenses/>.            #
#-------------------------------------------------------------------------------#

CONFIG += serialport

DEFINES += appVersion=1.00 \
           RESTART=255

QT += \
    core \
    gui \
    sql

greaterThan(QT_MAJOR_VERSION, 4):QT += widgets

TARGET = QtScale
TEMPLATE = app

target.path += /usr/local/bin
INSTALLS += target

SOURCES += \
    main.cpp\
    commandline.cpp \
    database.cpp \
    mainwindow.cpp \
    messagedialog.cpp \
    optionsdialog.cpp \
    productdialog.cpp \
    tabledialog.cpp \
    serialscale.cpp \
    licensedialog.cpp \
    aboutdialog.cpp
```

```
HEADERS  += \
    commandline.h \
    errorcode.h \
    database.h \
    mainwindow.h \
    messagedialog.h \
    optionsdialog.h \
    productdialog.h \
    productstruct.h \
    tabledialog.h \
    serialscale.h \
    licensedialog.h \
    aboutdialog.h

FORMS    += \
    mainwindow.ui \
    messagedialog.ui \
    optionsdialog.ui \
    productdialog.ui \
    tabledialog.ui \
    licensedialog.ui \
    aboutdialog.ui

RESOURCES += \
    Resources.qrc

#--- end of file ----------------------------------------------------------#
```

27 Resources.qrc

```
<RCC>
    <qresource prefix="/Buttons">
        <file>Buttons/exit.png</file>
        <file>Buttons/next.png</file>
        <file>Buttons/options.png</file>
        <file>Buttons/refresh.png</file>
        <file>Buttons/shutdown.png</file>
        <file>Buttons/warning.png</file>
        <file>Buttons/cancel.png</file>
        <file>Buttons/previous.png</file>
        <file>Buttons/executable.png</file>
        <file>Buttons/calculator.png</file>
        <file>Buttons/clear.png</file>
        <file>Buttons/emblem-default.png</file>
        <file>Buttons/about.png</file>
        <file>Buttons/certificate.png</file>
    </qresource>
</RCC>
```

28 serialscale.cpp

```
/**
*@file   serialscale.cpp
*@brief  Class implementation of serialScale.
*
* This class manages the serial communication with the industrial scale.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>
#include <string.h>
#include <stdio.h>

#include "serialscale.h"
#include "errorcode.h"

/**
  @brief Class constructor of serialScale object.

  This function creates an instance of serialScale class
  and initializes the class variables.
*/
serialScale::serialScale()
{
    totalData = 0;
    port = false;
    connection = false;
    stable = false;
    signal = true;
}

/**
  @brief Class destructor of serialScale object.

  This function destroys an instance of serialScale class.
```

```
*/
serialScale::~serialScale()
{
    closeConnection();
    delete serial;
}

/**
  @brief Establish serial connection with the industrial scale.

  This function creates, configures and opens a QSerialPort connection
  to communicate with the industrial scale and receive information
  on the weight of the products.

  Connection parameters configured to work with the Microgram ie21
  industrial scale from Microgram Instruments Espa ola , S.A.

  \verbatim
  Serial port:  /dev/ttyUSB0
  Baud rate:    9600
  Data bits:    8 bits
  Stop bits:    1 bit
  Parity:       no parity
  Flow control: no flow control
  \endverbatim

  This function is part of the "low level" functions of this class.

  @returns Error code, being SUCCESS if connection was established
  or serialError otherwise.
*/
errorCode serialScale::openConnection()
{
    // create port if not already created
    if (!port)
    {
        serial = new QSerialPort();
        port = true;
    }

    // close connection if open
    if (connection)
    {
        closeConnection();
        connection = false;
    }

    // set up connection
    serial->setPortName("/dev/ttyUSB0");
    serial->setBaudRate(QSerialPort::Baud9600);
    serial->setDataBits(QSerialPort::Data8);
    serial->setStopBits(QSerialPort::OneStop);
    serial->setParity(QSerialPort::NoParity);
    serial->setFlowControl(QSerialPort::NoFlowControl);

    // attempt to open connection
    if (serial->open(QIODevice::ReadWrite))
```

```
    {
        connection = true;
        return SUCCESS;
    }
    else
    {
        connection = false;
        return serialError;
    }
}

/**
  \brief Closes the serial port.

  Closes the serial port if it is open.

  This function is part of the "low level" functions of this class.
*/
void serialScale::closeConnection()
{
    if (connection)
    {
        serial->close();
    }
}

/**
  @brief Send data over serial.

  This functions is non-blocking.

  This function is part of the "low level" functions of this class.

  @param *data Pointer to char vector from which to read data.
  @param length Length in Bytes of data to be sent.
*/
void serialScale::sendData(const char *data, int length)
{
    if (connection)
    {
        serial->write(data, length);
    }
}

/**
  @brief Receive data over serial.

  This functions is non-blocking.

  This function is part of the "low level" functions of this class.

  @param *data Pointer to char vector to which to write data.
  @param maxData Maximum number of data bytes to receive.
  @returns Number of Bytes received.
*/
int serialScale::receiveData(const char *data, int maxData)
{
```

```
    // exit if no connection has been established
    if (!connection)
    {
        return 0;
    }

    int bytes = serial->bytesAvailable();

    // read available data
    if (bytes > 0)
    {
        if (bytes > maxData)
        {
            bytes = maxData;
        }
        serial->read((char *)data, bytes);
    }

    return(bytes);
}

/**
  @brief Try to get new measurement.

  This is the function to be called to attempt to retrieve and
  process a new weight measurement sent by the scale over RS-232.

  When the function is called, data Bytes received since the
  last call are added to the buffer. Then the buffer is evaluated for
  valid data streams. If a valid stream is found and processed,
  it is removed from the buffer.

  This function is intended to work with the Microgram ie21
  industrial scale from Microgram Instruments Espa ola, S.A.
  For more information on the ie21 data stream coding please
  consult the serialScale class documentation and the annex which
  includes the manufacturers datasheet.

  This function depens on buffScan() and process().
*/
void serialScale::update()
{
    int newData = receiveData(buffer+totalData, bufferSize-(totalData-1));

    // reset values on update
    weight = 0.0;
    stable = false;

    // process new data
    if (newData > 0)
    {
        int msgEnd;
        bool msgFound;
        totalData = totalData + newData;

        do
        {
```

```
            // attempt to find message
            msgFound = buffScan(buffer, totalData, &msgEnd);

            if (msgFound)
            {
                // remove message from buffer
                memmove(buffer,buffer+msgEnd+1,totalData-(msgEnd+1));
                totalData = totalData-(msgEnd+1);
            }
        } while (msgFound);

        // reset buffer if almost full
             /* this should only happen in connections */
             /* with significant data losses */
        if (totalData > 0.90*bufferSize)
        {
            totalData = 0;
        }
    }
}

/**
  @brief Scan buffer for measurements.

  This is the function searched for the start and end characters
  in the buffer. If they are found, the possible array is passed
  on for further processing.

  This function depend on process().

  This function is intended to work with the Microgram ie21
  industrial scale from Microgram Instruments Espa ola, S.A.
  For more information on the ie21 data stream coding please
  consult the serialScale class documentation and the annex which
  includes the manufacturers datasheet.

  @returns TRUE is a message is found and FALSE otherwise.
*/
bool serialScale::buffScan(char *buff, int numData, int *msgEnd)
{
    bool msgFound = false;

    // attempt to find message string ending
    for (int i=0; i < numData; i++)
    {
        if (buff[i-1]=='\x0D' && buff[i]=='\x0A')
        {
            msgFound = true;
            *msgEnd = i;
            break;
        }
    }

    // if found find begining of message
    if (msgFound)
    {
        for (int i=(*msgEnd); i >= 0; i--)
```

```
        {
            if (buff[i]=='S')
            {
                // process message
                process(buff+i,(*msgEnd)-i+1);
                break;
            }
        }
    }
    return (msgFound);
}

/**
  @brief Process possible data stream.

  This function takes a possible data stream and processes it to either
  discard it or extract from it the weight and stability values that the
  scale is sending over RS-232.

  This function is intended to work with the Microgram ie21
  industrial scale from Microgram Instruments Espa ola , S.A.
  For more information on the ie21 data stream coding please
  consult the serialScale class documentation and the annex which
  includes the manufacturers datasheet.
*/
void serialScale::process(char *buff, int size)
{
    // check size
    if (size < 18)
    {
        return;
    }

    // check start of message
    if (strncmp((const char*)buff,"S",1)==0)
    {
        const char *p;
        p = (const char *)(buff+1);

        switch (*p)
        {
            int sign;

            // stable measure
            case'␣':
                p = (const char *)(buff+3);

                if(*p=='-') sign = -1;
                else sign = 1;
                p++; // jump "minus" sign

                // extract weight
                sscanf(p,"%lf",&weight);
                weight=weight*sign;
                stable = true;

                // in reality negative weights would not be processed
```

```
                if (weight < 0.0)
                {
                    stable = false; // buying not allowed

                    // only show message once
                    if (signal)
                    {
                        message = new messageDialog();
                        message->errorMessage(scaleError);
                        message->exec();
                        delete message;
                        signal = false;
                    }
                }
                break;

            // unstable measure
            case 'D':
                p = (const char *)(buff+3);

                if(*p=='-') sign = -1;
                else sign = 1;
                p++; // jump "minus" sign

                // extract weight
                sscanf(p,"%lf",&weight);
                weight=weight*sign;
                stable = false;
                break;

            /*
            case 'I':
                if(*(p+1)=='+')
                {
                    // scale overloaded
                }

                else
                {
                    //scale underload
                }
                break;
            */

            // disregard unstable and/or out-of-range values
            default:
                weight = 0.0;
                stable = false;
                break;
        }
    }
}

/**
  @brief Get weight from scale.

  @returns Weight in kilograms.
```

```
*/
float serialScale::getWeight()
{
    return float(weight);
}

/**
  @brief Get stability of scale measurement.

  @returns Stability of most recent measurement.
*/
bool serialScale::isStable()
{
    return stable;
}

/**
  @brief Test serial port.

  This functions can be used to test the serial port on the system by loopback.
  If the industrial scale is not connected to the system and the system's
  serial port pin RX is connected the to its TX pin, all data sent is
  redirected as input data. This allows to simulate data stream transmission
  from the scale.

  Currently, there are 3 examples streams already defined for quick
  testing and debugging.

  \verbatim
  RX <--->TX
  \endverbatim

  This function is intended to work with the Microgram ie21
  industrial scale from Microgram Instruments Espa ola , S.A.
  For more information on the ie21 data stream coding please
  consult the serialScale class documentation and the annex which
  includes the manufacturers datasheet.
*/
void serialScale::loopbackTest()
{
    char dummy[20] = "S␣␣␣000000.50␣kg␣"; // 0.5 kg - stable
    //char dummy[20] = "SD  000001.23 kg "; // 1.23 kg - unstable
    //char dummy[20] = "S  -000000.10 kg "; // -0.1 kg - stable

    dummy[17] = '\x0D';
    dummy[18] = '\x0A';
    sendData(dummy, sizeof(dummy));
}

/*** end of file **************************************************************/
```

29 serialscale.h

```
/**
*@file   serialscale.h
*@brief  Class definition of serialScale.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef SERIALSCALE_H
#define SERIALSCALE_H

#define bufferSize 2048 ///< Size of serial reception buffer

#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>

#include "errorcode.h"
#include "messagedialog.h"

/**
  \brief this class encapsulates the methods and variables necessary to
  manage the communication with the industrial scale.

  This class manages the serial communication between the industrial scale
  and the application. Communication is carried put over RS-232.

  The string processing carried out in this class is intended to work with the
  Microgram ie21 from Microgram Instruments Espa ola , S.A.

  String processing functions must be modified if other hardware is to be used.

  The data stream encoding for this scale is shown here:

  \verbatim
```

```
    +--------------> Data stream start character "s"
    |      +--------> <space>
    |      |  +-----> sign of weight (<space> or "-")
    |      |  |  +--> 9 characters for the weight
    |      |  |  |
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
  |S |D |  |- |  |  |  |1 |2 |3 |. |4 |  |  |k |g |  |\r|\n|
  +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
       |                                       |  |  |  |  |
       +---> 2 characters for state     "k" <--+  |  |  |  |
  (#1) +-> <space> <space>              "g" <-----+  |  |  |
  (#2) +-> "D" <space>              <space> <--------+  |  |
  (#3) +-> "I" "+"                     0x0D <-----------+  |
  (#4) +-> "I" "-"                     0x0A <--------------+

  (#1) - Stable weight
  (#2) - Unstable weight
  (#3) - Out-of-Range (scale overloaded)
  (#4) - Out-of-Range (scale underloaded)
  \endverbatim
*/
class serialScale
{
    public:
        serialScale();
        ~serialScale();
        errorCode openConnection();
        void closeConnection();
        void sendData(const char *data, int length);
        int receiveData(const char *data, int maxData);
        void update();
        bool buffScan(char *buff, int numData, int *msgEnd);
        void process(char *buff, int size);
        float getWeight();
        bool isStable();
        void loopbackTest();

    protected:
        bool port;                 ///< Serial port found
        bool connection;           ///< Connection established
        bool stable;               ///< Stability of weight
        bool signal;               ///< Show warning
        char buffer[bufferSize]; ///< Serial reception buffer
        int totalData;             ///< Position in buffer
        double weight;             ///< Weight of product

    private:
        QSerialPort *serial;
        messageDialog *message;
};

#endif // SERIALSCALE_H

/*** end of file ***************************************************************/
```

30 tabledialog.cpp

```
/**
*@file   tabledialog.cpp
*@brief  Class implementation of tableDialog.
*
* This class displays a table showing the accumulated sales of
* the products in the database.
*
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <QSqlQueryModel>

#include "tabledialog.h"
#include "ui_tabledialog.h"

/**
  @brief Class constructor of tabletDialog object.

  This function creates an instance of productDialog class
  and initializes the widget.

  The table displays the Id, product name and accumulated sales
  for all products in the database.

  This function must be called after the class object has been created
  and before the dialog is executed.
*/
tableDialog::tableDialog(QWidget *parent):QDialog(parent),
    ui(new Ui::tableDialog)
{
    ui->setupUi(this);

    // set to fullscreen and frameless
    this->setWindowFlags(Qt::FramelessWindowHint);
    this->setFixedHeight(272);
    this->setFixedWidth(480);
```

```
    // add icons to buttons
    ui->closeButton->setIconSize(QSize(24,24));
    ui->closeButton->setIcon(QIcon(":/Buttons/Buttons/cancel.png"));
    ui->clearButton->setIconSize(QSize(24,24));
    ui->clearButton->setIcon(QIcon(":/Buttons/Buttons/clear.png"));

    // populate table
    model = new QSqlQueryModel;
    model->setQuery("SELECT␣ID,Name,␣Accumulated␣FROM␣Products");
    model->setHeaderData(0, Qt::Horizontal, "ID");
    model->setHeaderData(1, Qt::Horizontal, "Name");
    model->setHeaderData(2, Qt::Horizontal, "Sales");
    ui->tableView->setModel(model);

    // set size of table columns
    ui->tableView->setColumnWidth(0, 40);
    ui->tableView->setColumnWidth(1, 140);
    ui->tableView->setColumnWidth(2, 140);

    ui->tableView->sortByColumn(0, Qt::AscendingOrder);
}

/**
  @brief Class destructor of tableDialog object.

  This function destroys an instance of tableDialog class.
*/
tableDialog::~tableDialog()
{
    delete model;
    delete ui;
}

/**
  @brief Clear sales.

  This slot makes the tableDialog execution exits with value "true" and closes
      the dialog.

  Another module must then process the return value.
*/
void tableDialog::on_clearButton_clicked()
{
    this->accept();
    this->close();
}

/**
  @brief Cancel.

  This slot makes the tableDialog execution exits with value "false" and closes
      the dialog.

  Another module must then process the return value.
*/
void tableDialog::on_closeButton_clicked()
```

```
{
    this->reject();
    this->close();
}

/*** end of file ************************************************************/
```

31 tabledialog.h

```
/**
*@file   tabledialog.h
*@brief  Class definition of tableDialog.
*@date   March, 2014
*@author Pablo Cholbi
*/

/*
    This file is part of QtScale.

    QtScale is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, version 3.

    QtScale is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    Copyright (C) 2014  Pablo Cholbi Alenda.

    You should have received a copy of the GNU General Public License
    along with QtScale.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef TABLEDIALOG_H
#define TABLEDIALOG_H

#include <QDialog>
#include <QSqlQueryModel>

#include "database.h"

namespace Ui
{
    class tableDialog;
}

/**
  \brief this class encapsulates the methods and variables necessary to
  display the accumulated sales of all products in the database.
*/
class tableDialog : public QDialog
{
    Q_OBJECT

    public:
        explicit tableDialog(QWidget *parent = 0);
        ~tableDialog();

    private slots:
        void on_clearButton_clicked();
        void on_closeButton_clicked();

    private:
```

```
        Ui::tableDialog *ui;
        QSqlQueryModel *model;
};

#endif // TABLEDIALOG_H

/*** end of file ************************************************************/
```

32 tabledialog.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>tableDialog</class>
 <widget class="QDialog" name="tableDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>480</width>
    <height>272</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Dialog</string>
  </property>
  <widget class="QTableView" name="tableView">
   <property name="geometry">
    <rect>
     <x>10</x>
     <y>10</y>
     <width>360</width>
     <height>252</height>
    </rect>
   </property>
  </widget>
  <widget class="QPushButton" name="clearButton">
   <property name="geometry">
    <rect>
     <x>380</x>
     <y>73</y>
     <width>90</width>
     <height>35</height>
    </rect>
   </property>
   <property name="text">
    <string>Clear</string>
   </property>
  </widget>
  <widget class="QPushButton" name="closeButton">
   <property name="geometry">
    <rect>
     <x>380</x>
     <y>164</y>
     <width>90</width>
     <height>35</height>
    </rect>
   </property>
   <property name="text">
    <string>Close</string>
   </property>
  </widget>
 </widget>
 <resources/>
 <connections/>
</ui>
```

Annex 1 – Documentation Licence

Author:
Creative Commons

The documentation of this project is licensed under the Creative Commons Attribution-ShareAlike 4.0 International Public License . The terms and conditions for use, reproduction and distribution of documentation subject to this license are the following:

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-ShareAlike 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a) **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b) **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c) **BY-SA Compatible License** means a license listed at creativecommons.org/compatiblelicenses, approved by Creative Commons as essentially the equivalent of this Public License.

d) **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

e) **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

f) **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

g) **License Elements** means the license attributes listed in the name of a Creative Commons Public License. The License Elements of this Public License are Attribution and ShareAlike.

h) **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

i) **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

j) **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

k) **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

l) **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

m) **You** means the individual or entity exercising the Licensed Rights under this Public License. Your has a corresponding meaning.

Section 2 – Scope.

a) **License grant.**

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

   A) reproduce and Share the Licensed Material, in whole or in part; and

   B) produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

   A) Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

   B) Additional offer from the Licensor – Adapted Material. Every recipient of Adapted Material from You automatically receives an offer from the Licensor to exercise the Licensed Rights in the Adapted Material under the conditions of the Adapter's License You apply.

   C) No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others

designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b) **Other rights.**

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.
2. Patent and trademark rights are not licensed under this Public License.
3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a) **Attribution**.

1. If You Share the Licensed Material (including in modified form), You must:
    A) retain the following if it is supplied by the Licensor with the Licensed Material:
        i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);
        ii. a copyright notice;
        iii. a notice that refers to this Public License;
        iv. a notice that refers to the disclaimer of warranties;
        v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
    B) indicate if You modified the Licensed Material and retain an indication of any previous modifications; and
    C) indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.
2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.
3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

b) **ShareAlike**.

In addition to the conditions in Section 3(a), if You Share Adapted Material You produce, the following conditions also apply.

1. The Adapter’s License You apply must be a Creative Commons license with the same License Elements, this version or later, or a BY-SA Compatible License.

2. You must include the text of, or the URI or hyperlink to, the Adapter's License You apply. You may satisfy this condition in any reasonable manner based on the medium, means, and context in which You Share Adapted Material.
3. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, Adapted Material that restrict exercise of the rights granted under the Adapter's License You apply.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a) for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b) if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material, including for purposes of Section 3(b); and

c) You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a) Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b) To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c) The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a) This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b) Where Your right to use the Licensed Material has terminated under Section 6(a), it

reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or
2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c) For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d) Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a) The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b) Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a) For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.

b) To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c) No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d) Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Annex 2 – Software License

Author:
Free Software Foundation

The software developed in this project is licensed under the GNU General Public License Version 3. The terms and conditions for use, reproduction and distribution of software subject to this license are the following:

PREAMBLE

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program-to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed

to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program,

in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you

remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.)You may place additional permissions on material, added by you to a covered work, for which you have or can give

appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under

section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either

(1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have

the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

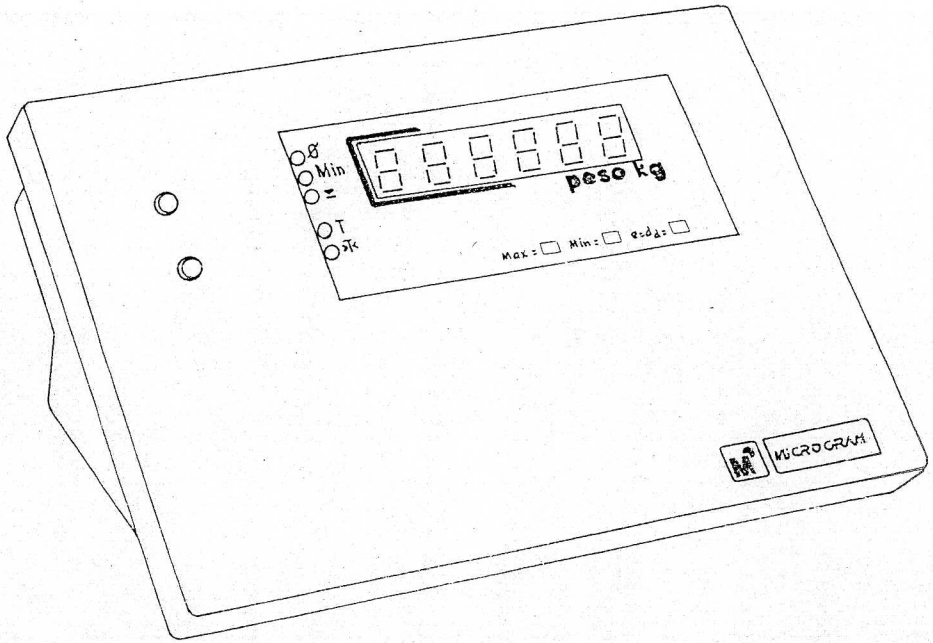
If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

Annex 3 - IE 21 Scale Manual

Author:
Microgram Instruments Española S.A.

SERVICIO TECNICO → 96 366 9199

FAX → 96 366 5000



IE 21
manual del usuario

1. CONFIGURACIÓN DEL VISOR

El visor de peso está configurado por 6 dígitos numéricos de 7 segmentos cada uno y de 5 leds que indican: el cero centrado, la pesada mínima, el peso estable, la tara y la tara bloqueada.

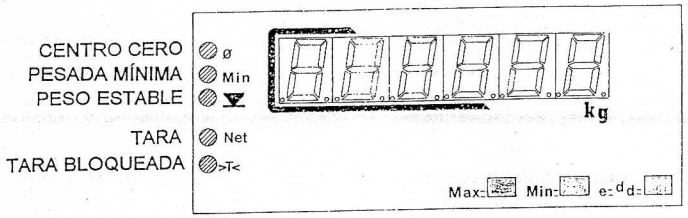


FIGURA 1

2. PUESTA EN MARCHA DEL INSTRUMENTO.

Encender el instrumento pulsando el interruptor de red situado en la parte posterior del instrumento, así se activa un proceso de test, tanto del softward como de los elementos físicos del sistema (sistema analógico y sistema digital); el visor se ilumina completamente para verificar todos los segmentos del display y se oscurece sucesivamente su visualización.

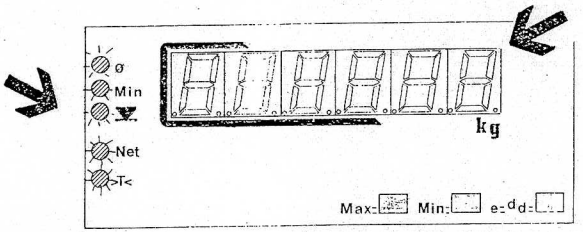


FIGURA 2

Al finalizar el test aparecerá en el visor el valor " 0 " (seguido del número de decimales programados en el instrumento). En el caso de no haber ningún decimal programado se visualizará la imagen representada en la figura 3 y el operador podrá iniciar la operación de pesar.

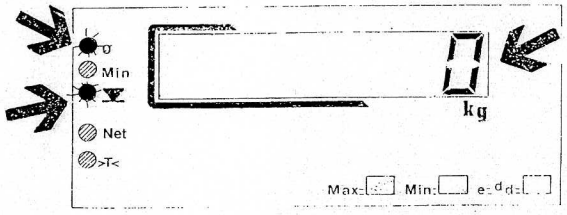


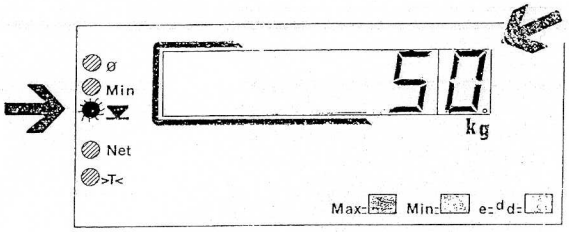
FIGURA 3

3. OPERACIÓN DE PESAR.

3.1 pesar simplemente.

El sistema visualiza el peso del objeto colocado en la plataforma, indicando sus dígitos el valor del peso.

Por ejemplo: colocar sobre la plataforma un peso de 50 kg, en el visor aparecerá 50 que indicará el peso en kg del objeto, cuando el peso sea estable se iluminará el led ⧗



FIGUR4

Cuando se retire el objeto de la plataforma el visor quedará tal como se ve en la figura 3.

3.2 Tara semiautomática.

Es posible efectuar una operación de tara, como tara substractiva del peso del objeto depositado sobre la plataforma.
Pulsando la tecla de tara, el instrumento espera a que el peso sea estable ante de restar el peso existente sobre la plataforma.
Es posible efectuar taras sucesivas.

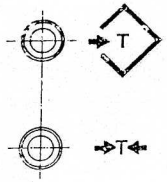
3.3 Bloqueo de la tara.

Pulsando la tecla de bloqueo se mantendrá fijo el valor de la última tara realizada sobre la plataforma.

4. TECLADO.

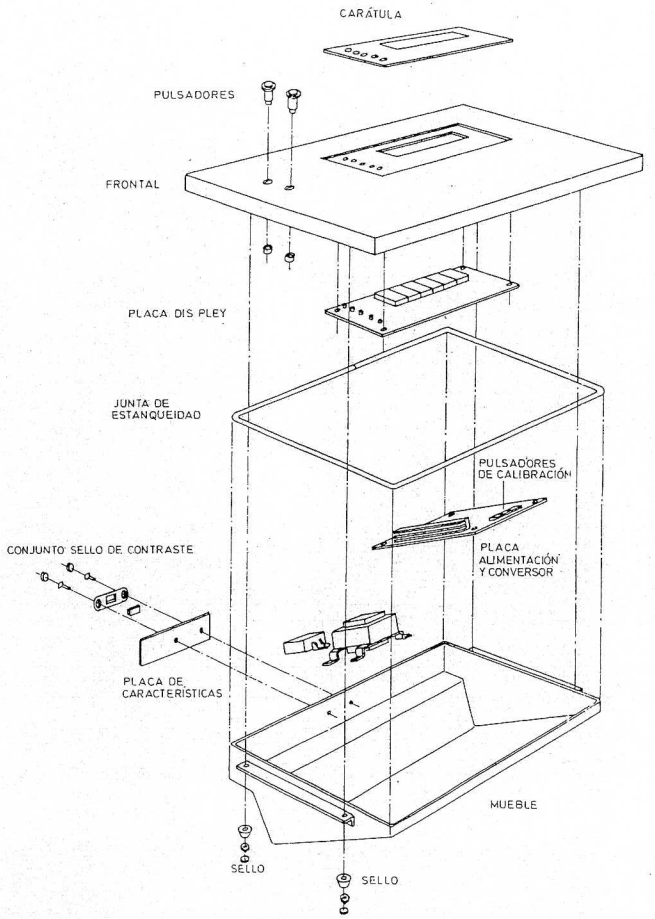
El teclado consiste en dos pulsadores que representan las funciones de tara y de bloqueo de la tara.

Los pulsadores se identifican con los siguientes símbolos:



Cuando se pulsa esta tecla, el instrumento espera que el peso depositado sobre la plataforma sea estable, antes de introducir el peso como tar.

Al pulsar esta tecla, el instrumento bloquea / desbloquea el peso obtenido al pulsar la tecla de tara.
Si no existe ninguna tara la función no se activa.



PROTOCOLO DE
COMUNICACIONES

5.- TRANSMISIÓN DE DATOS POR EL CANAL SERIE.

5.1.- CARACTERÍSTICAS GENERALES.

La transmisión es ASCII con 7 bits de datos, paridad programable (impar y 1 stop bit / No paridad 2 stop bits), así la longitud del byte de transmisión es de 10 Bytes siempre.

Velocidad de transmisión programable (Hardware) a 9600 o 1200 Baudios.

La codificación del Peso, Dirección, Fracción, Número de ceros y Flags, se efectúa codificando cada uno de los Nibbles de "0" a "F" que en ASCII corresponden a los valores hexadecimales de 30H a 39H y de 41H a 46H.

Cada equipo se identifica con una dirección de "0" a "F" (0 a 15 decimal) y sólo responde si coincide su dirección con la de demanda de datos. Dicha dirección se obtiene mediante puentes que se realizan en la CPU.

El equipo puede transmitir por petición o por propia iniciativa, cada 100 mseg o al finalizar la integración. Cuando arranca el equipo, un puente en la CPU determina el modo de funcionamiento. Durante el funcionamiento del equipo, el paso de un modo a otro se puede realizar por la recepción del comando "XOFF" (-"DC3"- 19 decimal)(paso a transmisión por petición) o del comando "XON" (-"DC1"- 17 decimal) (transmisión automática).

El equipo, sale de fábrica con las siguientes características de transmisión.

- **Direccionado en ADRESS 0 (30H)**
- ~~1200~~ **Baudios** 9600
- **Paridad impar**
- **7 bits de datos**
- **2 bits de stop**
- ~~**Transmisión por Petición**~~

5.2.- FORMATO DE LA TRANSMISIÓN DE DATOS.

Se transmite el siguiente paquete de datos, automáticamente o por petición (según párrafo anterior):

CÓDIGO	DESCRIPCIÓN	NOMENCLATURA	VALOR/ES
STX	inicio de trama		02H
ADRESS	dirección		0 ... F
PESO	Valor del peso	NIBBLE 0 -LSB	0 ... F
		NIBBLE 1	0 ... F
		NIBBLE 2	0 ... F
		NIBBLE 3 -MSB	0 ... F
FLAGS	Flags de Información	FLAG 0	0 ... F
		FLAG 1	0 ... F
		FLAG 2	0 ... F
		FLAG 3	0 ... F
FRACCIÓN	Valor de div.		1,2,5
CEROS	núm de ceros		0,1,2,3
ETX	Fin de trama		03H

Por ejemplo el paquete de datos a recibir (en Hexadecimal podría ser): (valores en Hexadecimal).

02	30	32	43	31	30	31	34	30	30	31	30	03

.<addr
.<peso
.<flags
.<fracción
.<ceros

El peso real será:
2C10 ..> 01C2H ..> 450D ->

$$peso = \frac{Valor}{3} \times (fracción \times 10^{ceros}) - \frac{450}{3} \times 1 = 150\ kg$$

Significa que transmite el equipo nº 0 un peso de 150 kg.

SIGNIFICADO DE LOS BITS		
FLAG 0	BIT 0 (LSB)	Peso Estable
	BIT 1	Error control RAM
	BIT 2	Error control ROM
	BIT 3 (MSB)	Peso Bajo Cero
FLAG 1	BIT 0 (LSB)	Peso fuera Escala
	BIT 1	Señal Excesiva
	BIT 2	Entrada prop. gen.
	BIT 3 (MSB)	Error crtl EEPROM
FLAG2	BIT 0 (LSB)	Conv. en Control
	BIT 1	Busca cero
	BIT 2	Error escr. EEPROM
	BIT 3 (MSB)	Señal bajo mínimo
FLAG3	BIT 0 (LSB)	PROGRAMACIÓN
	BIT 1	Cero Encontrado
	BIT 2	Inicilialización
	BIT 3 (MSB)	Busca Ganancia

En el ejemplo anterior nos resultaría:

-Peso Estable
-Entrada propósito general a 1

5.3.- TRANSMISIÓN POR PETICIÓN.

El paquete de datos a transmitir al equipo para que envíe el peso y los flags mencionados en el apartado anterior cuando este funciona en modo no automático debe ser:

CÓDIGO	VALORES POSIBLES
STX	02H
ADDRESS	30H .. 39H,41H .. 46H
ENQ	05H
ETX	03H

Conexión IE21 – Ordenador

IE 21 (canon DB-9)			Ordenador (canon DB-9)	
(RxD)	2	———	3	(TxD)
(TxD)	3	———	2	(RxD)
(GND)	5	———	5	(GND)
(RTS)	7	———	8	(CTS)
(CTS)	8	———	7	(RTS)
			4	
			6	

IE 21 (canon DB-9)			Ordenador (canon DB-25)	
(RxD)	2	———	2	(TxD)
(TxD)	3	———	3	(RxD)
(GND)	5	———	7	(GND)
(RTS)	7	———	5	(CTS)
(CTS)	8	———	4	(RTS)
			6	
			20	

Generalitat de Catalunya
Departament d'Indústria i Energia
Direcció General de Seguretat Industrial

Av. Diagonal, 405, bis
08008 Barcelona
Telèfon (93) 484 93 00
Telefax (93) 484 93 20

CERTIFICADO DE ENSAYO

Número E-95-02.C2

INDICADOR MICROGRAM MODELO IE 21

Emitido por : Dirección General de Seguridad Industrial de la Generalidad de Cataluña
(organismo notificado número 0315)
Avinguda de la Diagonal, 405 bis
E-08008 BARCELONA

En aplicación de: Párrafo 8.1 de la norma europea "Aspectos metrológicos de los instrumentos de pesaje de funcionamiento no automático" EN 45501:1992(+AC:1993). La fracción de error aplicada p_i en referencia al párrafo 3.5.4 de esta norma es 0,5.

Emitido para : MICROGRAM INSTRUMENTS ESPAÑOLA, S.A.
Travesia Industrial, 159
E-08902 L'HOSPITALET DE LLOBREGAT
ESPAÑA

Referente a : el modelo de un **dispositivo indicador**, ensayado como parte de un instrumento de pesaje de funcionamiento no automático.
fabricante: MICROGRAM INSTRUMENTS ESPAÑOLA, S.A.
modelo: IE 21.

Características : Adecuado para un instrumento de pesaje de funcionamiento no automático con las siguientes características: graduado, electrónico,monorango, monoescalón, con indicación digital de peso.

Clase de precisión (III),
número de escalones de verificación n ≤ 3000

Min 20 e
T -Max

Las características principales figuran en el anexo descriptivo adjunto que forma parte integrante del certificado de ensayo.
El modelo está descrito en la documentación técnica presentada, identificada con el número 8/95.
El resumen de los ensayos implicados se encuentra en el anexo descriptivo.

EL DIRECTOR GENERAL DE SEGURIDAD INDUSTRIAL

Albert Sabala i Duran

Barcelona, 18 de setiembre de 1995.

MICROGRAM INSTRUMENTS ESPAÑOLA, S.A.

Polígono Industrial FONTSANTA
Pasaje Mossota,16 - 08970 SANT JOAN DESPI (BARCELONA)
Tel. 93 477 30 40 - Fax 93 480 80 22 - Tel. S.A.T. 93 477 31 20
e-mail:microgram@retemail.es

CE

DECLARACIÓN DE CONFORMIDAD
DECLARATION OF CONFORMITY
DÉCLARATION DU CONFORMITÉ

El instrumento de pesaje a funcionamiento no automático de clase
The non-automatic weighing instrument of class
L´instrument de pesage à fonctionnement non automatique de classe

III

FABRICANTE: *Manufacturer:* Fabricant:	**MICROGRAM INSTRUMENTS ESPAÑOLA, S.A.**
TIPO: *Type:* Type:	**BÁSCULA MONOCÉLULA CON IE-21/B-31**
Nº del certificado de aprobación CE del tipo: *Nº of the EC type-approval certificate:* Nº du certificat d'approbation CE de type:	**I-98-010**

corresponde al modelo descrito en el certificado de aprobación CE del tipo, según las exigencias de la Directiva del Consejo 90/384/CEE y sucesivas modificaciones y a las exigencias de las directivas CE siguientes:
corresponds to the production model described in the EC type-approval certificate and to the requirements of the Council Directive 90/384/EEC as amended and to the requirements of the following EC directives:
correspond au modèle dècrit dans le certificat d'approbation CE de type, aux exigences de la Directive 90/384/CEE modifiée et aux exigences des directives CE suivantes:

- Compatibilidad electromagnética — 89/336 CEE
  Electromagnetic compatibility
  Compatibilité électromagnétique
- Seguridad eléctrica — 73/23 CEE
  Electrical security
  Sécurité électrique

Esta declaración es solo válida con el Certificado de Conformidad emitido por un Organismo Notificado.
This declaration is only valid with a Certificate of Conformity by a notified body.
Cette dèclarátion est valide seulement avec une attestation de conformité d´un organisme notifié.

Fdo. DIRECCIÓN GENERAL *Signature:* Signature:	FECHA: *Date:* Date: 22 MAYO 2003	Sello MICROGRAM INSTRUMENTS ESPAÑOLA, S.A. Polígono Industrial Fontsanta - Pasaje Mossota, 16 Tel. (93) 477 30 40 - Fax (93) 480 80 22 08970 SANT JOAN DESPI (Barcelona)