Desarrollo arquitectura de control de robots móviles basada en radiofrecuencia y su aplicación al robot Moway

Proyecto Fin de Carrera

Titulación: Ingeniería Superior Informática
Autor: Sami Racho Ferrandis
Director de Proyecto: Martín Mellado Arteche
01/09/2011

TABLA DE CONTENIDOS

Índice de Ilustraciones

1 INTRODUCCIÓN

Es indudable que la robótica está cada vez más extendida en la sociedad y que sus aplicaciones se cuentan por miles. Desde el ámbito industrial hasta el médico, su avance se ha disparado durante las últimas décadas.

Lo más interesante desde el punto de vista educativo, es que ese avance en las tecnologías relacionadas con la robótica se ha traducido en una mayor accesibilidad.

Un claro ejemplo es la gama de robots Lego Mindstorms NXT, concebidos como un juguete, pero que ofrecen muchísimas posibilidades para desarrollar aplicaciones que bien podrían ser el punto de inicio de otras más complejas, que podrían utilizarse en sistemas más avanzados.

Esto resulta una clara ventaja de cara a la investigación, puesto que con un presupuesto reducido se pueden realizar experimentos y desarrollar tanto software como hardware con grandes posibilidades de aplicarse a la práctica.

Siguiendo la estela de los avances en robótica durante las últimas décadas, podemos observar que las mejoras en la parte mecánica de los robots, no son el principal impedimento para conseguir que estos realicen más funciones y de forma más rápida y eficiente.

En la inteligencia artificial, disciplina íntimamente ligada a la robótica, es donde se encuentran más problemas a la hora de crear robots más “inteligentes”. Con inteligentes nos referimos a robots más capaces para realizar las tareas a las que han sido encomendados de forma eficaz.

El poder realizar análisis, simulaciones, resolución de algoritmos, reconocimientos de formas y trayectorias etc. abrirá sin lugar a dudas el camino para las siguientes generaciones de robots. De ahí la importancia de estos desarrollos.

2 OBJETIVOS

El objetivo de este proyecto es ofrecer las herramientas necesarias para que un usuario pueda comunicarse con un robot en un lenguaje de programación de alto nivel e interactuar con sus sensores.

Todos los esfuerzos están destinados a abstraer lo máximo el hardware del robot, para que el investigador no se vea obligado a conocer los detalles de su implementación y la comunicación se realice de la forma más natural posible. Para ello se ha desarrollado una librería que será la que actúe de intermediaria entre el robot y la aplicación que se realice y que se describirá detalladamente en las siguientes secciones.

Podemos clasificar los puntos que se abarcarán en el siguiente orden:

1. Elaboración del software necesario para comunicarse de forma inalámbrica. Para ello, basándose en el sistema operativo Windows, se debe implementar un sistema capaz de escribir y leer bits en un dispositivo usb de radio frecuencia.

2. Partiendo del punto anterior, se debe desarrollar una capa adicional para codificar y descodificar la información utilizando bits, de manera que los dispositivos puedan entenderse.

3. Por un lado el PC debe ser capaz de descodificar los datos enviados por el robot y también de enviar los comandos con un formato correcto para que puedan ser ejecutados por el microcontrolador del robot.

4. Se debe desarrollar una capa adicional, que permita interactuar con todos los sensores y motores del robot de la forma más natural posible, ocultando todos los procesos que se realizan a bajo nivel tanto en al pc como en el robot y

proporcionando comandos sencillos para enviar órdenes, y pedirle información sobre sus sensores al robot.

5. Todo ello debe encapsularse en una librería de enlace dinámico dll, en el lenguaje de programación C++, para que pueda ser agregada de forma sencilla a proyectos nuevos, proporcionando la funcionalidad requerida, sin afectar al resto del proyecto.

6. Por último, debe escribirse una documentación detallada, donde pueda consultarse el funcionamiento de la librería, el modo de uso de cada comando, los valores que devuelve etc.

En resumen, lo que se perseguirá, es crear una interfaz de comunicación inalámbrica bidireccional robot-PC con fines docentes, con la que los alumnos sean capaces de trabajar con muy pocos conocimientos de programación, e ignorando el funcionamiento interno de los componentes del hardware del robot. A continuación se adjunta un esquema de la librería acabada y las capas que la constituirán:

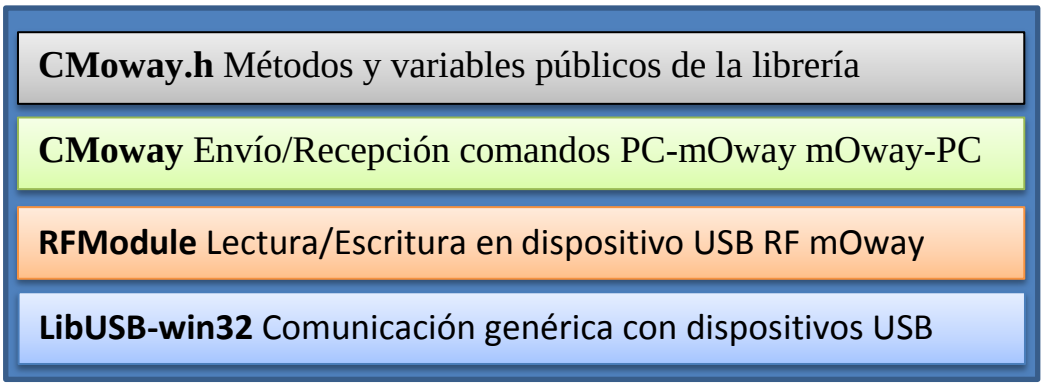

IMAGEN 1 ESQUEMA DE LA LIBRERÍA Y SUS CAPAS

3 EQUIPAMIENTO

3.1 EL ROBOT MOWAY.

El robot mOway es un minirobot autónomo y programable, destinado principalmente a entornos docentes y de investigación. Sus principales características que se detallarán en las secciones siguientes son:

- Sensores y motores para interactuar con el mundo real.
- Posibilidad de ampliar su hardware fácilmente gracias a un bus de expansión destinado a tal efecto.

3.1.1 DISEÑO EXTERNO:

Como puede apreciarse en las siguientes imágenes, el minirobot es bastante pequeño y manejable, y destaca su forma, carente de ángulos, lo que facilita que no se quede bloqueado al toparse con paredes, obstáculos e.t.c

IMAGEN 2 ASPECTO EXTERNO DEL ROBOT MOWAY

3.1.2 HARDWARE:

mOway está compuesto por los siguientes elementos.

- Procesador.
- Dos motores independientes
- Batería recargable
- Sensores

En el siguiente esquema se puede observar la situación de cada uno de estos componentes.

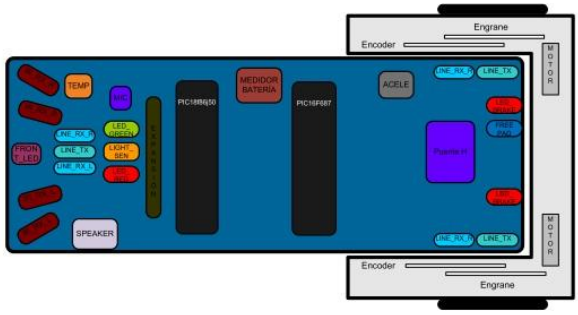

IMAGEN 3 ESQUEMA DE LOS COMPONENTES DEL MOWAY

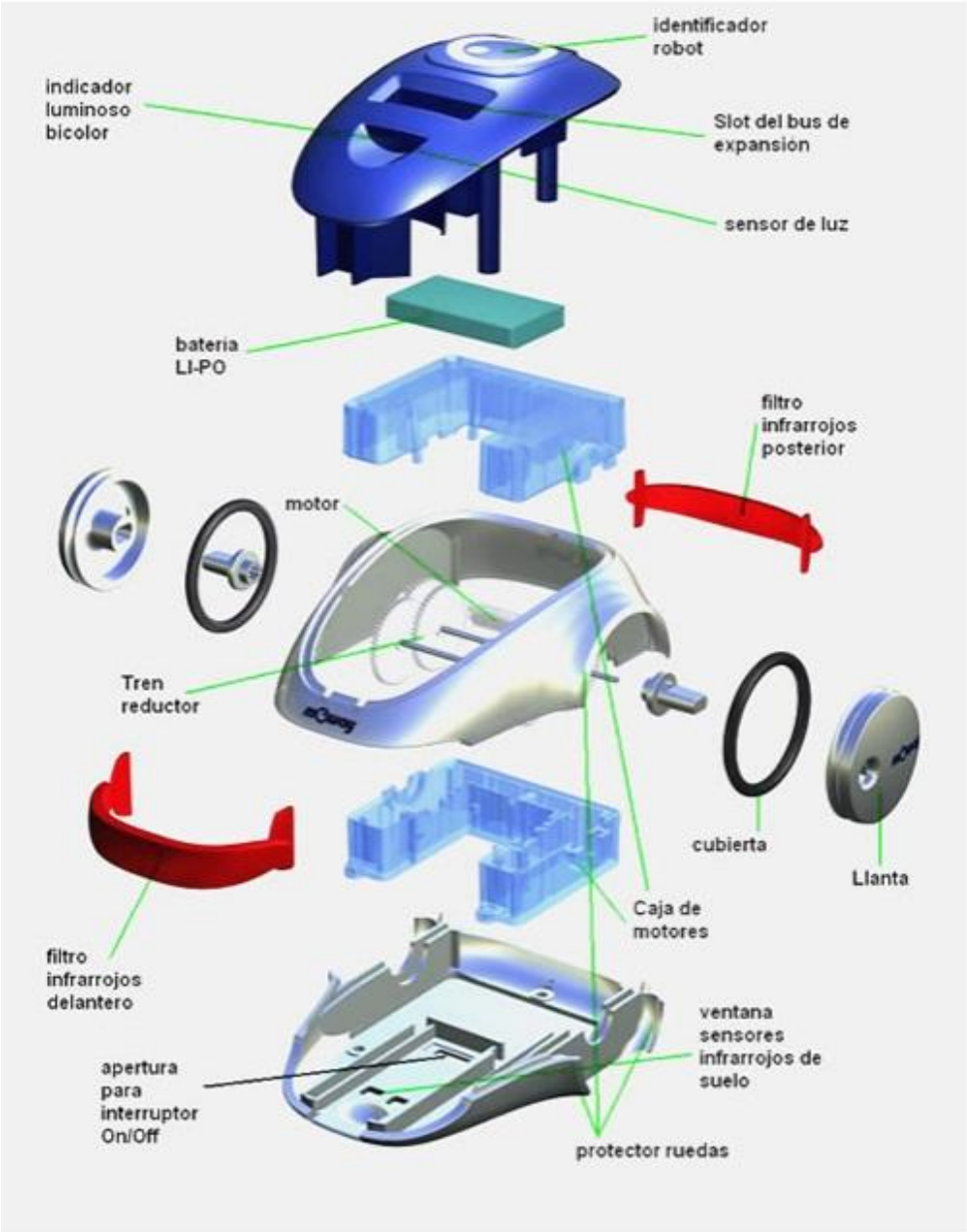

IMAGEN 4 VISTA DESGLOSADA HARDWARE MOWAY

3.1.2.1 PROCESADOR.

El mOway tiene un microcontrolador modelo PIC18F86J50 a 4Mhz.

En el siguiente enlace se puede encontrar la documentación completa del microcontrolador.

http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en027175

Los programas que se desarrollen para el robot se cargan en este microcontrolador. En nuestro caso, se diseñó un programa para que el robot pudiera enviar y recibir comandos a través de su interfaz inalámbrica.

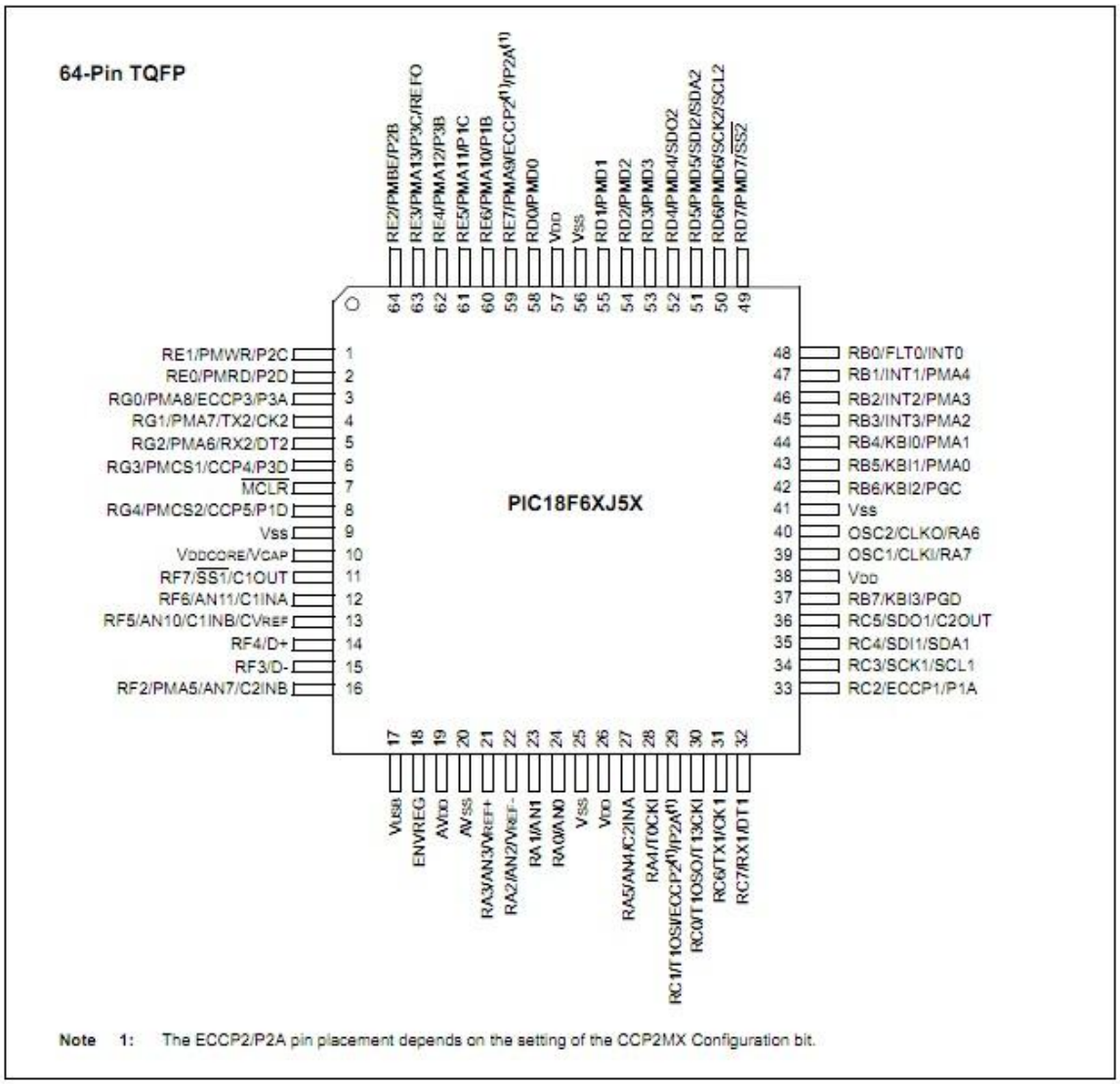

IMAGEN 5 DIAGRAMA DEL MICROCONTROLADOR

3.1.2.2 MOTORES.

mOway dispone de dos motores totalmente independientes que le permite desplazarse.

Estos motores constan de dos partes, una mecánica que se encarga de imprimir el movimiento al robot y otra electrónica encargada de la velocidad de los motores.

El mOway girará hacia un lado u otro cuando las velocidades de ambos motores sean distintas.

El servo-motor ofrece las siguientes funcionalidades.

1. Control de la velocidad de cada motor por separado.
2. Control de la distancia recorrida.
3. Cuentakilómetros
4. Control del ángulo de rotación del mOway
5. Control del tiempo de cada comando con procesión de 100ms.

3.1.2.3 SENSORES.

mOway cuenta con los siguientes sensores:

- Cuatro detectores de obstáculos.
- Dos sensores de línea.
- Sensor de luz.
- Bus de expansión.
- Cuatro tipos de LEDs
- Sensor de temperatura.
- Altavoz.
- Micrófono.
- Acelerómetro.
- Nivel de batería.

IMAGEN 6 PLACA ELECTRÓNICA MOWAY

3.1.2.3.1 DETECTORES DE OBSTÁCULOS.

mOway posee cuatro detectores de obstáculos independientes.

Los detectores están compuestos por dos fuentes de luz infrarroja KPA3010-F3C de Knightbright.

Enlace a su hoja de especificaciones:
http://www.kingbright.com/product_main_1.php?product01=20050504115352&product02=20050518061248&level=20050519091608&lang=Korea

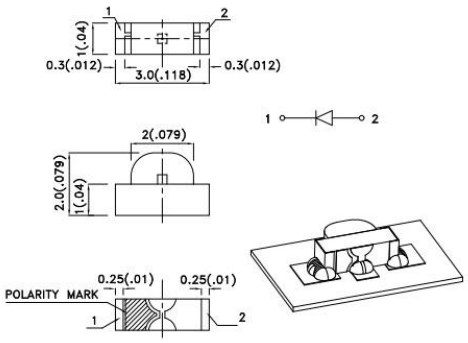

IMAGEN 7 ESQUEMA KPA3010-F3C

También posee cuatro receptores PT100F0MP de Sharp.

Están colocados en los extremos delanteros de la carcasa del robot. El modo de funcionamiento es el siguiente. El emisor de luz genera un pulso, si éste es reflejado en un obstáculo y captado por el receptor, se considera que el robot tiene algo delante de él.

3.1.2.3.2 SENSORES DE LÍNEA.

En la parte inferior del mOway se encuentran dos sensores que funcionan de manera parecida a los detectores de obstáculos. funcionan también con luz infrarroja para detectar el tono del suelo en el lugar donde se encuentre el robot. Para ello hace uso del sensor CNY70 de Vishay, capaz de distinguir entre diferentes tonos.

Hoja de especificaciones:

http://www.vishay.com/optical-sensors/list/product-83751/

IMAGEN 8 SENSOR VISHAY CNY70

Si el robot se encuentra sobre una superficie clara, se reflejará casi toda la luz infrarroja.

3.1.2.3.3 SENSOR DE LUZ

Este sensor permite captar la intensidad de luz ambiental. Está ubicado en un pequeño hueco en la parte delantera del robot. Está compuesto por un sensor APDS-9002 de Avago Technologies.

Enlace a su hoja de especificaciones:

http://www.avagotech.com/docs/5989-3051EN

IMAGEN 9 SENSOR APDS-9002

Este sensor está conectado a un puerto analógico del del microcontrolador, lo que nos permite detectar las variaciones de intensidad de la luz captada.

3.1.2.3.4 Bus de Expansión

Este bus permite conectar cirtuitos para ampliar el hardware del robot. En la siguiente tabla podemos encontrar la relación de conexiónes de que dispone.

Pin Expa	I/O	PIC
Pin1	O	Vcc 3.3v
Pin2	O	GND
Pin3	I/O /PMD3/AN12/P3C /C2INC	RH4
Pin4	I/O/PMA5/AN7/C2INB	RF2
Pin5	I/O /SCK1/SCL1	RC3
Pin6	I/O /SDO1/C2OUT	RC5
Pin7	I/O /SDI1/SDA1	RC4
Pin8	I/O/INT	RB0

3.1.2.3.5 LEDs

mOway posee los siguientes LEDs:

- LED Frontal de color blanco.
- LED bicolor (verde/rojo) superior.
- LED de freno (rojo)

3.1.2.3.6 Sensor de temperatura

La temperatura se mide mediante un transmisor modelo NTC de Murata. Su resistencia eléctrica disminuye a medida que la temperatura aumenta.

Enlace al catálogo de semiconductores de detección de temperatura Murata:

http://www.murata.com/products/catalog/pdf/r44e.pdf

3.1.2.3.7 Altavoz

mOway es capaz de emitir tonos a través de un buzzer CMT-1102 de CUI INC. Los tonos pueden estar entre los 250Hz hasta los 5,6kHz.

IMAGEN 10 BUZZER CMT-1102

3.1.2.3.8 MICRÓFONO

mOway monta un micrófono modelo CMC-5042PF-AC de CUI INC.

Permite detectar sonidos desde 100HZ hasta 20KHZ.

Enlace a hoja de especificaciones:

http://products.cui.com/adtemplate.asp?invky=925120

IMAGEN 11 MICRÓFONO CMC-5042PF-AC

3.1.2.3.9 ACELERÓMETRO.

El acelerómetro permite medir la aceleración y las fuerzas inducidas por la gravedad. Utiliza un acelerómetro MMA7455L de Freescale Semiconductor.

Enlace a su hoja de especificaciones:

http://www.freescale.com/files/sensors/doc/data_sheet/MMA7455L.pdf

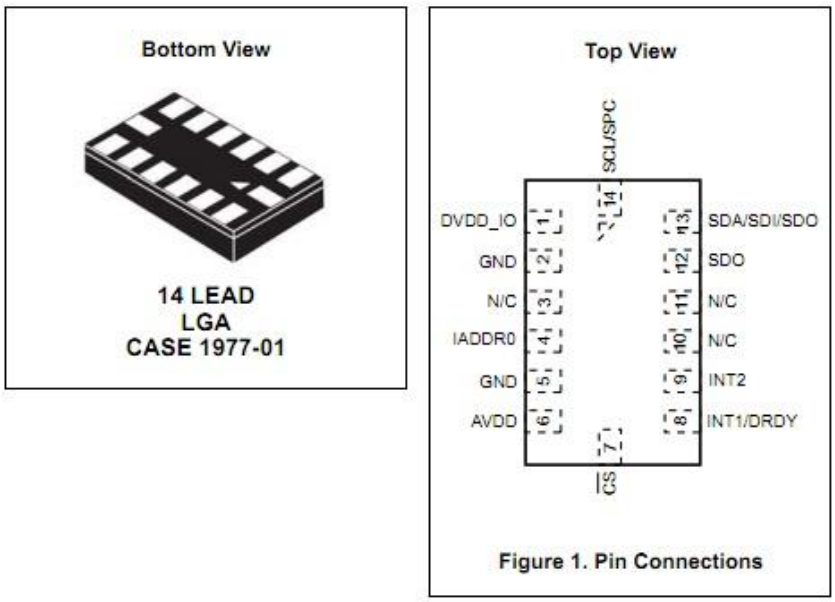

IMAGEN 12 ACELERÓMETRO MMA7455L

3.1.2.3.10 BATERÍA

mOway posee una batería LiPo recargable. Es posible medir la cantidad de batería restante ya que va conectada a una entrada analógica del microcontrolador.

3.1.3 SOFTWARE:

mOway viene acompañado de algunas aplicaciones para poder realizar programas para él y cargarlos a traves de su puerto USB.

Desde la página http://www.mOway-robot.com es possible descargarse el mOway Pack que ofrece los siguientes elementos:

- Manual de usuario del robot mOway.
- Guía de introducción.
- Software mOway GUI.

El software mOway GUI nos permite generar programas para el robot, de manera visual, muy útil como introducción y para uso en entornos docentes donde los alumnos no posean conocimientos de programación.

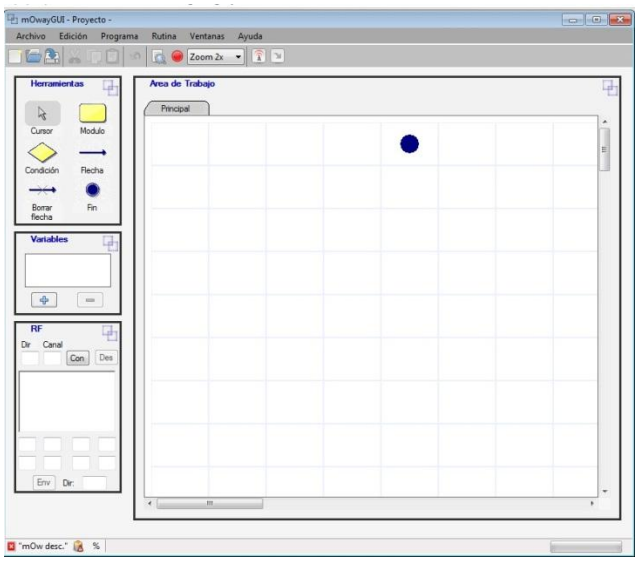

IMAGEN 13 MOWAY GUI

A través de mOway GUI es posible cargar software para el robot compilado con algún programa externo, siempre que el formato sea el adecuado para su microcontrolador.

Más adelante, en las secciones que describen cómo se realizó el proyecto, se indica paso a paso cómo instalar un entorno de desarrollo para el mOway.

4 ETAPAS DE DESARROLLO

Este proyecto comenzó como una práctica de la asignatura de Robótica, cuyo objetivo era desarrollar una librería que permitiera la comunicación inalámbrica entre un ordenador y un robot MOway por medio de un programa escrito en C++.

Para su realización, contábamos con la documentación del robot y una librería de comunicación escrita en C#.

La primera aproximación fue intentar recompilar la librería para que pudiera ser reutilizada, pero encontramos problemas insalvables, porque no poseíamos el código fuente completo, por lo que se decidió implementar una librería nueva desde cero.

El proyecto se dividió en tres partes:

1. Envío y recepción de bits entre el robot y el transmisor inalámbrico.
2. Firmware para el robot.
3. Librería con las funciones para enviar y recibir comandos.

4.1 DESARROLLO INICIAL.

4.1.1 ENVÍO Y RECEPCIÓN DE BITS ENTRE EL ROBOT Y EL TRANSMISOR INALÁMBRICO:

En la primera parte el objetivo era desarrollar el software necesario para poder leer y escribir datos en el transmisor inalámbrico USB que acompañaba al robot.

Para ello se hizo uso de una librería Open Source llamada LibUsb http://www.libusb.org/

Libusb permite un acceso uniforme a los dispositivos usb desde diferentes sistemas operativos. Puesto que el proyecto tenía como objetivo el sistema operativo Windows, y

el lenguaje de programación Visual C++ se utilizó libusb-win32. Se puede consultar toda la documentación relativa a este proyecto en su página Web:

http://sourceforge.net/apps/trac/libusb-win32/wiki

4.1.2 FIRMWARE DEL ROBOT:

Fue necesario el desarrollo de un firmware para que el robot interactuara con los distintos comandos codificados que se le enviaban de forma inalámbrica. Una versión inicial fue desarrollada en C++ con un compilador especial destinado al procesador que incluye el robot, más adelante se explica detalladamente tanto, la configuración del entorno de desarrollo para el firmware, así como de su código.

4.1.3 LIBRERÍA EN C++

El objetivo era ofrecer una capa de abstracción entre la parte 1 y la parte 2 del desarrollo. Para ello se creó el código necesario para que el envío y la recepción de datos entre el robot y un programa escrito en C++ se hicieran con funciones sencillas de utilizar. A continuación se incluye el informe de esta implementación.

WORK REPORT

on the development of a Programming Interface for the MOway Robot

Júlia Biró

julia.biro@gmail.com

Sami Racho Ferrandis

maesesami@gmail.com

Summary

The team developed a programming interface in the form of a linkable C++ class that can be used to program a MOway Robot. The interface was designed to serve teaching purposes. The interface can be used with other tools to remotely control and receive sensor information from a robot unit.

Keywords: MOway, robot, programming interface, dll

1. Assignment Goal, structure of assigment solution

The MOway Robot is a programmable mobile robot designed for teachning purposes. The group's goal was to develop a porgramming interface that is most apt for future assignments, that can be used to remotely control a MOway Robot unit.

The organization of this report is the following: in section 2.1, we will explain the structure of the solution, and mark out what part of this was completed by the team formed by the authors. In section 2.2 we will describe the programming interface. In Section 2.3 we will analyze the process of the work and the problems encountered. In section 2.4 we will give instructions how to use the programming interface.

2. Assignment solution

2.1 Structure of the solution

The solution of the assingment uses 4 independent layers, that were developed by separate teams (table 1). The first layer is the firmware of the MOway Robot, that provides a basic interface for commanding the robot and taking measurements from its sensors. The second layer is a communication layer, that ensures raw data communication between a comupter and a Radio Frequency unit that is attachable to the robot, using a USB radio transmitter. The third layer provides an interface for the raw data communication, processes the commands and the received data, and stores error information. The fourth layer provides a C++ programming interface that can be used from a Visual C++ program to command the robot. The team carried out the construction of the 4th layer, and participated in the testing and debugging of all 4 layers.

Table 1: Solution layers

	Layer name	**developer**
4	Programming Interface (CMOway)	S. Ferrandis, J. Biró
3	Communicaion interface (RFModule)	Marcin Nikliborc
2	Raw cumminication (libusb.lib)	
1	MOway firmeare	Peter Hanger

2.2 The programming interface

The programming interface was designed to be used in future class assignments, therefor it had to be able to provide a certain way of control, notably there was an epmhasis on controlling the movement of the robot by contorlling the motors of its two wheels.

To adapt to the future purpose of the interface, we studied a past class assigment using a similar robot and the programming interface providied to it. We designed the new interface to be as similar to the old one as possible. This sometimes meant that some more user-friendly capabilities of the MOway were made unaccessible from this interface.

The interface consists of one single C++ class that is providied as a dinamically linkeable library (dll).The interfaces functions can be classified into the following groups: connection handling methods, motor and LED commands, sensor readings, error handling. We are going to summarize the functions shortly, for exact deatils please consult the full documentation. All functions return a boolean value: true on successfull completion and false otherwise.

2.2.1 Connection handling methods

`bool ConnectMOway(int robotId)`: connects to the mOway robot identified by the number robotId. This id is set in the firmware, and indicated by a sticker on each robot.

`bool DisconnectMOway()` disconnects from robot.

Naturally any the control and sensor reading methods ony work, if the connection to the robot is successfully established.

2.2.2 Motor and LED commands

The inetrface defines enumerations to be used for certain parameters of the methods discussed in sections 2.2.2 and 2.2.3. These enums both improve the legibilitiy of programs and protect against incorrect parametrization.

The most important steering function is

`bool SetSpeed(int speed_motor_left, int speed_motor_right, direction dir_left=FORWARD, direction dir_right=FORWARD, int duration_left=0, int duration_right=0)`, it sets the speed and

direction of each motor, and it allows to set the duration while the motors should be moving (by defult it is until further command). For the direction parameters, the enumeration `enum direction {BACKWARD, FORWARD}` should be used; the speed should be given on a scale of 0-100..

The function `bool GoStraight (int speed, direction dir=FORWARD, int duration=0)` is a simplified version of the previous command, with the same values applying to both wheels. The function `bool MotorStop()` stops the motor of both wheels, therefoe making the mOway stop.

The function `bool ChangeLEDState(CMOway::leds led_number, CMOway::led_action action_number)` serves to turn on and off the different LEDs of the robot. For the parameters the enumerations `enum leds {LED_FRONT, LED_BRAKE, LED_TOP_GREEN, LED_TOP_RED}` and `enum led_action {ON, OFF}` should be used.

2.2.3. Sensor and motor reading

The sensor and reading commands serve to provide information about the environment and inner status of the robot. With their help it is possible to design behaviours that react to the changes of the surroundings of the robot. Since the reading functions too return a boolean value to indicate the success or failure of the reading process, the rad numeric vaules are written into variables that should be passed as parameters.

The firstgroup of the reading functions provide information on the motors. The method `bool ReadSpeed(int *speed_motor_left, int *speed_motor_right)` reads and writes the speed of each motor into the parameter varables.

The function `bool ReadMotorDistances(int* left, int* right)` reads the distance covered by each wheel in mms. The counter can be reset by `bool MotorDistanceReset()`. The function `bool CMOway::ReadMotorKM(int* KM)` returns the distance covered by the robot in kilometers, the counter can be reset by

`bool MotorKmReset()`. The function `bool ReadMotorTime(int* time)` reads the time the motors have been active, the counter resettable by `bool MotorTimeReset()`. For fine steering, it is possible to read the rotation of each wheel in degrees with the function `bool ReadWheelRotation(int* left, int* right)`.

The next group of functions make it possible the use the sensors of the robot, to measure environmental vales. The MOway Robot has a set of proximity sensors built in, their values can be read by the function `bool ReadProximitySensors(int* left, int *center_left, int* center_right, int* right)`. The amibent light level (on a scale 1-255) and temperature (in degrees Celsius) can be read by te functions `bool ReadAmbientLightSensor(int* lightsensor)`. and `bool ReadTemp(int* temp)`.

The acceleration in one axis can be read by `bool ReadAccelerator(int* result, CMOway::axis eje)`, the enumeration `enum axis {X, Y, Z}` should be used for the second parameter. The return value indicates a point on an linear scale between -2G and +2G.

And finally, the battery charge level can be read by `bool ReadBatteryStatus(int* battery)`.

2.2.4 Error handling

The interface collects all error messages accumulated, which can be obtained by the method `std::string GetErrors()`. The collection can be reset to empty by the function `void ResetErrors()`. The error messages indicate initialization errors, connection errors and parameter errors.

2.3 Work process and problems

The first step of the work was designing the solution structure. We decided that the best option to provide the interface would be a dynamically linked C++ library that can be used in a Visual C++ environment. For that we investigated how such libraries should be constructed and used, especially in the targeted programming environment,and we created test projects to see if this structure works porperly. The following step was to define the interface, deciding what commands should be supported, what features should be hidden. For that we studied both the MOway Robots manual and the interface used in a previous teaching assignment. After that we implemented the functions, using the 3rd layer to send commands to the mOway. A typical implementation consists of the following steps:

1. constructing a message of appropriate form using the received function
2. sending the message
3. recieveing the mOways answer
4. in case of reading functions, writing the read values into the reading parameters
5. error handling.

In the last phase we tested and debugged the solution.. For this, we created small projects and started to conduct experiments, testing the communication and the functions. During this phase we investigated and overcame the following problems (among others): the proper settings of the programming environment Visual Studio for using a precomplied dll in a project; a bug in the 3rd layer that resulted in problems when reading values from the robot; a compilation setting that induced unsuspected delays and caused seemingly inexplicable behaviour. That last one was the hardest problem to overcome, because it was caused by the porgramming environment and had nothing to do with our implementation.

2.4 Use of the solution

In order to use the library, you have to:

1. Enter into VS2008.
2. Create a new C++ Windows Form Project.
3. Go to Project properties (Alt+F7).
4. In General tab, change Common Language Compatibility to “Common Language Compatible /clr”

5. Copy inside the project folder CMOway.lib, CMOway.h, RFModule.h, usb.h and CMOway.dll
6. Right click on solution explorer, click on “Add existing file” and add CMOway.lib, CMOway.h, RFModule.h, usb.h, and CMOway.dll to the project.
7. Finally, add #include “CMOway.h” on the file you want to call the library.
   To test your program, after compliation:

   3. Plug in the usb transmitter
   4. Turn on the MOway Robot
   5. Run your program

   Please make sure that your solution is compiled in "Release" mode, because programs complied in "Debug" mode may have unexpected delays

   in them.

Acknowledgements

Martin Mellado, Peter Hanger, Marcin Nikliborc.

References

[1] MOway User manual

[2] Khapera assignment

4.2 ELABORACIÓN DEL PROYECTO

A partir de una versión bastante inmadura de la librería, comenzó el desarrollo de la versión completa, añadiendo el manejo de todos los sensores disponibles en el robot y depurando su funcionamiento.

Dividiremos el proceso de creación del proyecto en las siguientes subsecciones:

- Preparación del entorno de desarrollo.
- Programación y funcionalidad del Firmware instalado en el robot
- Programación y funcionalidad de la librería en C++
- Desarrollo de pruebas y resultado.
- Modificación de práctica destinada a estudiantes para el uso de la librería.
- Problemas encontrados y consejos de utilización de la librería.

4.2.1.1 PREPARACIÓN DEL ENTORNO DE DESARROLLO.

En esta sección se detallarán los pasos necesarios para tener un entorno de desarrollo funcional, tanto para compilar programas para el microcontrolador PIC18F86J50, como para utilizar la librería CMOway en un proyecto en Visual C++.

4.2.1.1.1 REQUISITOS DE HARDWARE.

Para poder trabajar necesitaremos los siguientes elementos:

- Un mOway v2.
- Un Módulo de expansión RF.
- Un módulo USB RF.
- Un cable USB – Mini USB
- Un PC con el sistema operativo Windows XP, Vista o 7 tanto de 32 bits como de 64 bits.

IMAGEN 14 MOWAY V2

IMAGEN 15 MÓDULO USB RF

IMAGEN 16 MÓDULOS EXPANSIÓN RF

IMAGEN 17 CABLE USB - MINI USB

4.2.1.1.2 Entorno de desarrollo para mOway

4.2.1.1.2.1 Paso 1: Instalación mOway Pack V2.2

El primer paso es instalar el software mOway Pack v2.2 que puede descargarse desde la página web del fabricante:

http://www.mOway-robot.com/index.php?option=com_content&task=blogcategory&id=53&Itemid=204

Este software lo necesitaremos tanto por las librerías adicionales que le acompañaban, como por los drivers del módulo RF USB. Ya que sin ellos nos será imposible comunicarnos de forma inalámbrica con el robot. Si la instalación se ha realizado correctamente, al conectar el modulo USB RF, Windows debería instalar los drivers automáticamente. En caso contrario habría que reinstalar el software asegurándose de ejecutarlo con permisos de Administrador local del equipo.

4.2.1.1.2.2 Paso 2: Instalación MPLAB IDE

Para generar programas para el robot mOway, haremos uso de la aplicación MPLAB IDE v8.56 que puede descargarse desde el siguiente enlace:

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002

Una vez instalado procederemos a su configuración.

Haciendo uso del Project Wizard de MPLAB, la configuración es sencilla.

Dentro del programa nos vamos al Menú Project > Project Wizard.

Seleccionamos el microcontrolador PIC18F86J50

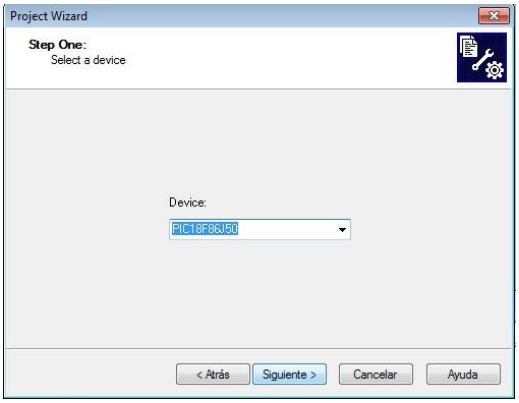

IMAGEN 18 PASO 1 WIZARD MPLAB

Configuramos las opciones de la siguiente manera:

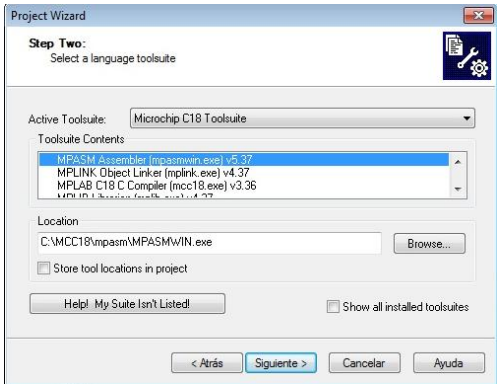

IMAGEN 19 PASO 2 PROJECT WIZARD MPLAB

Una vez generado el proyecto solo nos falta un último paso, se deben agregar los archivos .h del robot mOway que agregan la funcionalidad tanto para utilizar el módulo de expansión RF como los sensores. Para ello deberemos copiar los archivos lib_mot_mOway.h, lib_rf2gh4.h y lib_sen_mOway.h dentro de la carpeta Header Files de nuestro proyecto. Estos archivos se encuentran dentro de la carpeta de instalación del mOway Pack. La ruta por defecto es C:\Archivos de Programa\mOwayPack\Robot mOway\Librerías\C18

Dentro de nuestro archivo main.c incluiremos las referencias a los archivos .h

```
#include "p18f86j50.h"
#include "lib_rf2gh4.h"
```

```
#include "lib_mot_mOway.h"
#include "lib_sen_mOway.h"
```

Una vez configurado, el proyecto debería verse como en la siguiente imagen

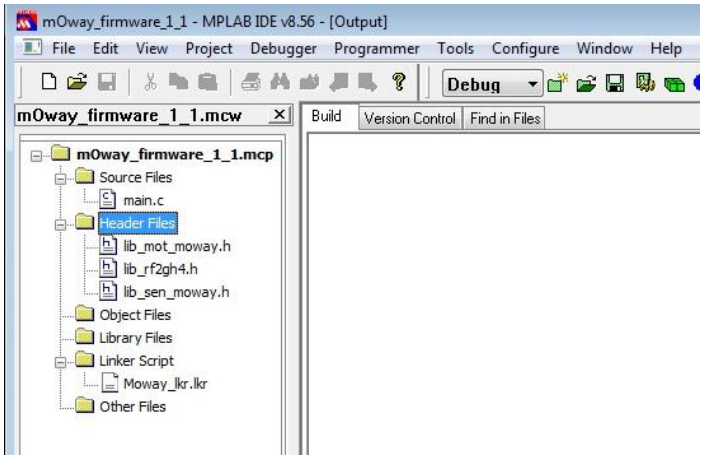

IMAGEN 20 PROYECTO MPLAB

4.2.1.1.2.3 PASO 3: DESCARGAR PROGRAMAS EN MOWAY.

Cuando compilemos un programa, nos generará un archivo .hex que será el que deberemos cargar en el robot a través del cable USB, mediante el programa mOway GUI.

Para ello conectamos el robot mOway con el cable USB al PC, y desde la aplicación mOway GUI, vamos al menú Archivo > Importar/Grabar

Seleccionamos el archivo. Hex y esperamos a que el programa se vuelque en la memoria del robot.

4.2.1.1.3 ENTORNO DE DESARROLLO EN VISUAL STUDIO 2008

Lo único que debemos hacer es crear un proyecto C++ según nuestros requisitos. Si deseamos utilizar formularios crearemos un proyecto Windows Forms, y sino una aplicación de consola Win32.

4.2.1.2 PROGRAMACIÓN Y FUNCIONALIDAD DEL FIRMARE INSTALADO EN EL ROBOT

El objetivo de esta parte del desarrollo, era conseguir un programa que permitiera al mOway establecer un canal de escucha para el módulo RF, y que fuera capaz tanto de transmitir la información de sus sensores, como de recibir órdenes y ejecutarlas.

En cada trama se pueden transmitir hasta 8 bytes. Lo que se intentó es aprovechar al máximo la ocupación de estos bits para evitar retardos tanto como fuera posible.

Un ejemplo de esto es fácil de observar en la configuración de los motores. En la versión inicial de la librería, si se quería establecer la velocidad de los motores de ambas ruedas, se debían enviar dos órdenes desde el PC al mOway. El problema de esto es que al enviar dos comandos, la velocidad de los motores no se establecía al mismo tiempo, y provocaba que el robot se desplazara levemente hacia un lado antes de iniciar el movimiento.

Siempre que ha sido posible, se ha intentado minimizar el impacto de los retardos sobre el funcionamiento del robot. Puesto que no es un programa demasiado largo, adjunto su código comentado para que se pueda ver qué funciones realiza.

Básicamente establece un canal para el módulo RF e inicia un bucle infinito esperando órdenes. Cuando recibe una orden la decodifica, y si es válida, ejecuta el comando asociado.

Para consultar el código comentado detallamente debe consultarse el anexo 1 . Exactamente el documento de nombre “main.c”

4.2.1.3 PROGRAMACIÓN Y FUNCIONALIDAD DE LA LIBRERÍA EN C++

El objetivo de la librería es ejercer de puente entre el programa cargado en el robot mOway y el módulo USB RF.

De este modo se pueden enviar comandos al robot comp por ejemplo:

mOwayRobot->MotorStop()

De esta forma se simplifica de manera evidente la comunicación.

A continuación se adjunta la tabla con todas las funciones disponibles en la librería.

Nombre	Descripción	Parámetros
bool ConnectMOway(int robotId)	Función para conectar con el robot	int robotid: Identificador del robot
void SetSendDelay(int delay);	Función para establecer el retardo en milisegundos antes de enviar un comando	Int delay: Retardo en milisegundos, por defecto 100
bool DisconnectMOway(bool disableRF=false);	Función para desconectar el robot	bool disableRF. Indica si queremos desconectar el m□o rf del mOway. Si se desconecta el rf del mOway habrá apagarlo y encenderlo otra vez para que pueda volver a recibir órdenes.
std::string GetErrors();	Devuelve un string con detalles sobre el error	
void ResetErrors();	Vacia el string de errores	
bool SetSpeaker (int freq, speaker_mode mode, int time=0);	Función para definir el Funciónamiento del altavoz	inf freq: Frecuencia del sonido. Valores posibles 0-255. Valor Frecuencia 0 0,0000000 10 5681,8181818 50 1225,4901961 100 618,8118812 150 413,9072848 200 310,9452736 250 249,0039841 255 244,1406250 enum speaker_mode. Valores posibles: SPEAKER_OFF, SPEAKER_ON, SPEAKER_TIME. (SPEAKER_TIME indica que se emitirá un sonido durante el tiempo especificado en la variable time) int time. Indica el tiempo que estará sonando (solo Funcióna si mode = SPEAKER_TIME). Valores posibles 0-255.(1 = 100ms)

bool SetSpeed(int speed_motor_left, int speed_motor_right, direction dir_left=FORWARD, direction dir_right=FORWARD, int duration_left=0, int duration_right=0);	Función para definir la velocidad del robot y la duración del movimiento	int speed_motor_left: velocidad del motor izquierdo en %. Valores posibles de 0-100 int speed_motor_right: velocidad del motor derecho en %. Valores posibles de 0-100 direction dir_left: sentido del motor izquierdo. Valores posibles enum {BACKWARD, FORWARD} direction dir_right: sentido del motor derecho. Valores posibles enum {BACKWARD, FORWARD} int duration_right: Duracion del movimiento del motor derecho(1 = 100ms). Por defecto 0, movimiento infinito. int duration_left: Duracion del movimiento del motor izquierdo(1 = 100ms). Por defecto 0, movimiento infinito.
bool SetSpeedDistance(int speed_motor_left, int speed_motor_right, direction dir_left, direction dir_right, int distance_right, int distance_left);	Función para definir la velocidad del robot y la distancia del movimiento	int speed_motor_left: velocidad del motor izquierdo en %. Valores posibles de 0-100 int speed_motor_right: velocidad del motor derecho en %. Valores posibles de 0-100 direction dir_left: sentido del motor izquierdo. Valores posibles enum {BACKWARD, FORWARD} direction dir_right: sentido del motor derecho. Valores posibles

		enum {BACKWARD, FORWARD} int distance_right: Distancia a recorrer por la rueda izquierda en mm. Valores posibles (0-255) int distance_left: Distancia a recorrer por la rueda derecha en mm. Valores posibles (0-255)
bool GoStraight (int speed, direction dir=FORWARD, int duration=0);	Función para hacer que el robot se mueva en línea recta	int speed velocidad de los motores de ambas ruedas en %. Valores posibles de 0-100 direction dir: sentido del movimiento de ambas ruedas. Valores posibles enum {BACKWARD, FORWARD} int duration: Duracion del movimiento de ambos motores (1 = 100ms). Por defecto 0, movimiento infinito.
bool GoStraightDistance (int speed, direction dir, int distance);	Función para hacer que el robot se mueva en línea recta una distancia concreta	int speed velocidad de los motores de ambas ruedas en %. Valores posibles de 0-100 direction dir: sentido del movimiento de ambas ruedas. Valores posibles enum {BACKWARD, FORWARD} int distance. Distancia a recorrer en mm. Valores posibles (0-255)
bool MotorStop();	Función para parar el movimiento del robot	
bool ChangeLEDState(CMOway::leds led_number, CMOway::led_action action_number);	Función para cambiar el estado de los leds	enum led led_number: El led. Valores posibles LED_FRONT, LED_BRAKE, LED_TOP_RED, LED_TOP_GREEN

		enum led_action: Para apagar o encender el led. Valores posibles ON,OFF
bool ReadSpeed(int *speed_motor_left, int *speed_motor_right);	Obtiene la velocidad de las ruedas en %. Valores posibles de 0-100	
bool ReadProximitySensors(int* left, int *center_left, int* center_right, int* right, bool digital=false);	Obtiene la lectura de los sensores de proximidad	En modo analógico obtiene valores entre 0 y 255 (0 no hay ningún obstáculo, 255 el sensor está pegado al obstáculo) En modo digital obtiene valores entre 0 y 1. (0 no hay obstáculo 1 el sensor está pegado al obstáculo) int left: Valor del sensor izquierdo. Valores posibles (0-255) int center_left: Valor del sensor centro-izquierdo. Valores posibles (0-255) int center_right: Valor del sensor centro-derecho. Valores posibles (0-255) int right: Valor del sensor derecho. Valores posibles (0-255) bool digital: Si se pone a true, se obtendrá el valor digital de cada sensor (devuelve 0 si no se detecta obstáculo y 1 en caso contrario). Si se pone a false se obtendrán valor entre 0-255 para cada sensor. Por defecto es false

bool ReadLineSensors(int* left, int* right, bool digital = false);	Obtiene la lectura de los sensores de línea	En modo analógico obtiene valores entre 0 y 255 (0 no hay ninguna línea, 255 el sensor está pegado a la la línea) En modo digital obtiene valores entre 0 y 1. (0 no hay ninguna línea 1 el sensor está pegado a la línea) int left: Valor del sensor izquierdo. Valores posibles (0-255) Valor del sensor derecho. Valores posibles (0-255) bool digital: Si se pone a true, se obtendrá el valor digital de cada sensor (devuelve 0 si no se detecta línea y 1 en caso contrario). Si se pone a false se obtendrá un valor entre 0-255 para cada sensor. Por defecto es false
bool ReadAmbientLightSensor(int* lightsensor);	Devuelve la cantidad de luz detectada en %. Valores posibles 0-100 (0 oscuridad)	
bool ReadBatteryStatus(int* battery);	Devuelve el estado de la bateren %. Valores posibles 0-100	
bool CMOway::ReadMic(int* temp, bool digital=false);	Función para leer el ruido ambiental	int* temp: El ruido detectado por el mOway. Valores posibles 0-255 bool digital: Indica si queremos obtener el valor digital del sensor de ruido. 0 no se detecta nada 1 se detecta ruido. Valor por defecto false. Devuelve true si se recibe confirmación del mensaje por parte

		del mOway
bool ReadTemp(int* temp);	Devuelve la temperatura en C. Error estimado +-5C	
bool MotorDistanceReset();	Pone a 0 el contador de distancia	
bool MotorKmReset();	Pone a 0 el cuentakilómetros	
bool MotorTimeReset();	pone a cero el tiempo	
bool ReadMotorDistances(int* left, int* right);	Obtiene la distancia recorrida por las ruedas en mm. Valores posibles 0-255	
bool ReadMotorTime(int* time);	Obtiene el tiempo transcurrido 100ms/bit. Valores posibles 0-255	
bool CMOway::ReadMotorKM(int* KM);	Obtiene el cuenta kilometros 1mm/bit. Valores posibles 0-512	
bool ReadWheelRotation(int* left, int* right);	Obtiene la rotacion de las ruedas. Valores posibles de 0-360	
bool ReadAccelerator(int* result, CMOway::axis eje);	Aceleración sufrida por el robot en los ejes x, y ,z. Valores de 0 a 255 (0 -2G, 255 2G) +-0.0156G/bit	

4.2.1.4 PRUEBAS

4.2.1.4.1 DESARROLLO DE LAS PRUEBAS

Para la realización de las pruebas se contó con la ayuda de varios alumnos que realizaron en base a una memoria que realicé, con el principal objetivo de intentar averiguar el comportamiento de la comunicación RF y la estabilidad de la librería de comunicación.

Cabe mencionar, que la solución CMoway en formato Visual Studio 2008, tiene incluido un proyecto llamado CheckDll, que permite comprobar la funcionalidad de la librería sin necesidad de crear un proyecto desde cero.

El enunciado de la práctica era el siguiente:

Pruebas de funcionamiento de la librería RF CMoway

Consideraciones.

El software moway-gui debe estar instalado en el equipo para poder utilizar el pen rf-usb.

Recordad que Moway-Gui debe estar cerrado cuando se hagan las pruebas porque de lo contrario el pen rf-usb estará bloqueado.

El robot pierde mucha precisión cuando su batería está por debajo del 50%

La comunicación rf es poco tolerante a obstáculos. Se debe intentar mantener una línea de visión limpia entre el pen rf-usb y el moway.

Los sensores de proximidad no funcionan correctamente sobre ciertas superficies (sobre todo oscuras) y por su disposición, no detectan obstáculos si el robot se aproxima a ellos en diagonal. Esto son limitaciones de diseño del robot que hay que tener en cuenta a la hora de evaluar los resultados de las pruebas.

Para los chequeos del micrófono se puede utilizar está página web, que genera ruido blanco: http://www.simplynoise.com/classic/

Chequeo físico del robot.

Durante la realización de las pruebas se deberá anotar cualquier anomalía física y reproducible del robot. (Que se desplace levemente hacia un lado, que no se encienda algún led, que algún sensor nunca de un valor correcto e.t.c)

Se rellenará una tabla con el siguiente formato.

Descripción del problema	Condiciones en las que ocurre

Chequeo de las funciones:

Se deben chequear las funciones de la librería CMoway para comunicarse con el robot. Para ello se probarán una a una con al menos **tres combinaciones de parámetros aleatorios** en caso de que la función tenga parámetros (pero dentro de los rangos válidos).

Se rellenará una tabla con la siguiente información:

Nombre de la función	Valor devuelto a 1m/2m		Datos devueltos a 1m/2m		Resultado robot 1m/2m	

Siendo:

Nombre de la función: Nombre de la función con los parámetros utilizados.

Valor devuelto a 1m/2m: Valor true o false devuelto por la función a 1m y 2m de distancia entre el pen rf-usb y el robot.

Datos devueltos a 1m/2m: En algunas ocasiones las funciones devuelven los datos de algunos sensores. Aquí se apuntaran sus valores.

Resultado robot 1m/2m: Aquí se indicará si el robot efectivamente ha realizado la tarea o no.

Así por ejemplo, si probamos la función `ConnectMoway` a 1m de distancia, la función nos devuelve true y el moway ha conectado, la tabla quedará de la siguiente forma:

Nombre de la función	Valor devuelto a 1m/2m		Datos devueltos a 1m/2m		Resultado robot 1m/2m	
ConnectMoway()	true		-		correcto	

Para chequear algunas funciones necesitaremos realizar alguna acción previa, por ejemplo al leer la velocidad sería recomendable que el moway estuviera en movimiento.

Funciones a chequear:

1. bool ConnectMoway(int robotId);
2. bool DisconnectMoway();
3. std::string GetErrors();
4. void ResetErrors();
5. bool SetSpeaker (int freq, speaker_mode mode, int time=0);
6. bool SetSpeed(int speed_motor_left, int speed_motor_right, direction dir_left=FORWARD, direction dir_right=FORWARD, int duration_left=0, int duration_right=0);
7. bool SetSpeedDistance(int speed_motor_left, int speed_motor_right, direction dir_left, direction dir_right, int distance_right, int distance_left);
8. bool GoStraight (int speed, direction dir=FORWARD, int duration=0);
9. bool GoStraightDistance (int speed, direction dir, int distance);
10. bool MotorStop();
11. bool ChangeLEDState(CMoway::leds led_number, CMoway::led_action action_number);
12. bool ReadSpeed(int *speed_motor_left, int *speed_motor_right);
13. bool ReadProximitySensors(int* left, int *center_left, int* center_right, int* right, bool digital=false);
14. bool ReadLineSensors(int* left, int* right, bool digital = false);
15. bool ReadAmbientLightSensor(int* lightsensor);
16. bool ReadBatteryStatus(int* battery);
17. bool CMoway::ReadMic(int* temp, bool digital=false);
18. bool ReadTemp(int* temp);
19. bool MotorDistanceReset();
20. bool MotorKmReset();
21. bool MotorTimeReset();
22. bool ReadMotorDistances(int* left, int* right);
23. bool ReadMotorTime(int* time);
24. bool CMoway::ReadMotorKM(int* KM);
25. bool ReadAccelerator(int* result, CMoway::axis eje);

Chequeo de retardos:

El objetivo de estas pruebas es intentar localizar los problemas que surgen cuando se le envían muchas órdenes seguidas al moway.

Para ello deberán realizarse algoritmos sencillos y repetitivos (con una duración no superior a 5 min), observar el comportamiento del robot, y en caso de que deje de responder, intentar acotar los fragmentos de código involucrados. Para ello sería recomendable que el programa de prueba fuera escribiendo en un archivo de texto las funciones que está ejecutando junto con una marca de tiempo, ya que si se utiliza el debugger la ejecución del programa es paso a paso y es más difícil reproducir los comportamientos anómalos.

En la línea 236 del archivo RFModule.cpp hay un `Sleep (100)`, que introduce un retardo al enviar una orden al robot, para evitar que se solapen comandos. Al realizar los algoritmos,

sería recomendable modificar este valor y comprobar si se obtienen mejores o peores resultados.

También se debe tener en cuenta la exposición a interferencias por redes WIFI. Si es posible, se deben ejecutar los algoritmos en zonas con más y con menos interferencias y comparar los resultados.

El objetivo de esta prueba no es resolver algún problema complejo, sino comprobar la estabilidad de la comunicación RF.

Ejemplos de algoritmos sencillos:

- Encender y apagar los leds en una secuencia 100 veces.
- Moverse, detectar si existen obstáculos y en ese caso girar y continuar.
- Emitir pitidos si se detecta mucha luz.

Se deberán apuntar los resultados obtenidos, así como sugerencias para mejorar su funcionamiento

4.2.1.4.2 RESULTADOS

Los resultados de las pruebas se pueden consultar en los anexos 3 y 4

4.2.1.5 MODIFICACIÓN DE PRÁCTICA DESTINADA A ESTUDIANTES PARA EL USO DE LA LIBRERÍA.

En base a la librería creada se construyó una práctica para los alumnos de la asignatura de Robótica, a continuación adjuntamos la última versión de la práctica con las modificaciones marcadas en amarillo, para reflejar las últimas funcionalidades agregadas a ésta.

El documento con las modificaciones pertinentes se puede encontrar en el Anexo 5

5 CONSEJOS DE UTILIZACIÓN DE LA LIBRERÍA

Prácticamente todos los métodos devuelven true si se han completado con éxito y false en caso contrario. Es muy recomendable comprobar estos valores, porque en caso contrario, si el envío de algún comando no es correcto, el resto de la ejecución del programa se vería comprometida.

Algo tan sencillo como un bucle que se repita X veces hasta que se tenga confirmación de que un comando se ha ejecutado con éxito, puede resolver muchos problemas.

Es importante que el nivel de batería esté alto, sino pueden producirse problemas en la respuesta del robot.

Siempre que sea posible, es aconsejable utilizar el robot con el WIFI desactivado y en lugares con pocas interferencias.

Debido al modo de funcionamiento de los detectores de proximidad, se deben usar como obstáculos, objetos preferentemente blancos, ya que así los sensores son más precisos.

Si el robot se acerca en diagonal a algún objeto, seguramente no detectara el obstáculo debido a que el haz de luz infrarroja saldrá rebotado hacia otra dirección. Esto es una limitación debida a la disposición de los sensores en el robot.

6 CONCLUSIONES

Ha resultado satisfactorio llevar a cabo este proyecto, y más aun sabiendo que tendrá una aplicación práctica inmediata y no se quedará simplemente acumulando polvo en los archivos. Esto me hace pensar que seguirá vivo y que incluso podrá ser retomado para futuras ampliaciones. Creo que es todo un acierto por parte de los profesores de robótica el intentar que sus alumnos tengan acceso a los robots y puedan trabajar con ellos, ya que motiva bastante más trabajar con modelos físicos y no quedarse solo en simulaciones.

En mi primera aproximación a la robótica, nuestro objetivo era crear un algoritmo para resolver un laberinto. Todo se hizo mediante simulaciones por ordenador, y nuestros supuestos robots no eran más que un punto en la pantalla. Esta primera toma de contacto me resultó frustrante y muy aburrida. Tanto que no volví a interesarme por la robótica hasta que tuve la oportunidad de hacer pequeñas pruebas con minirobots en la UPV.

El interactuar con un robot es infinitamente más inmersivo, incluso cuando se comenten errores, que utilizando una simulación que tan solo muestra un error por pantalla. Así que en este proyecto he buscado intentar facilitar esa interacción, para que los usuarios de la librería CMoway puedan comunicarse de forma trnasparente con el robot aún con conocimientos reducidos en programación.

La parte del proyecto que más dura, ha sido la relacionada con el transporte de las tramas de datos. Muchos problemas venían por el solapamiento de instrucciones, y el debugging era complicado, ya que al poner puntos de ruptura se pausaba la ejecución y los errores no se producían.

Otro gran problema con la librería de comunicación RF, es que el módulo USB RF suministrado por el fabricante, trabaja en la misma frecuencia que las extendidas redes WIFI, por lo que en entornos congestionados se producen muchas interferencias y a veces el mOway deja de responder. La solución a esto es complicada, ya que la única alternativa que ofrece el fabricante es un módulo de expansión WIFI, cuyo funcionamiento es completamente distinto, por lo que para adaptar la librería habría que reescribir gran parte del código. La solución más sencilla aunque no muy elegante, pasa por no alejar mucho el emisor del robot.

Una lectura muy interesante se encuentra en el Proyecto de fin de carrera de Carmen Coll Cámara, una estudiante de la universidad de Elche. En él, se dedican varias páginas a los errores en la odometría del mOway. Sus conclusiones son que los errores de precisión del robot son demasiado aleatorios como para encontrar un patrón, opinión que comparto.

Enlace al su proyecto:

http://coolab.umh.es/mOway_web/PFC_CarmenCollCamara.html

7 BIBLIOGRAFÍA

Web mOway-robot:

http://www.mOway-robot.com/

Documentación del robot mOway, Manual de usuario y material de ejemplo C18 incluído con el robot.

MPLAB IDE Manual

Avoiding DLL Hell

http://msdn.microsoft.com/en-us/magazine/bb985026.aspx

Creating Visual C++ Dll

http://logix4u.net/Programming/vc++/A_Tutorial_on_creating_DLLs_with_VC++.html

Creating Dinamic link libraries

http://msdn.microsoft.com/en-us/library/1ez7dh12.aspx

Libusb documentation

http://sourceforge.net/apps/trac/libusb-win32/wiki/libusbwin32_documentation

8 RELACIÓN DE DOCUMENTOS ANEXOS

Nombre del documento	Autor	Descripción
Anexo 1 Firmware mOway.zip	Sami Racho Ferrandis. Versión inicial de Peter Hanger	Proyecto de MPLAB con el programa creado para el robot mOway
Anexo 2 Solución VS2008 librería CMoway	Sami Racho Ferrandis. En la versión inicial colaboraron: • Julia Biro • Marcin Nikliborc • Slawomir Moriak	Solución VS2008 con la librería CMoway lista para ser compilada. También incluye un proyecto llamado CheckDll para hacer pruebas con la librería implementada.
Anexo 3 Resultado Pruebas Funcionamiento.pdf	Laura Arnal Benedicto	Resultados obtenidos al realizar las batería de pruebas.
Anexo 4 Resultado Pruebas Funcionamiento.pdf	Carmen Arbonara de la Asunción	Resultados obtenidos al realizar las batería de pruebas.
Anexo 5 Programación de un robot móvil mod.doc	Martin Mellado Arteche	Práctica realizada en base a la librería CMoway