UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de un sistema de adquisición de datos relacionados con el consumo eléctrico basado en IOT

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Juan Manuel Cuartas Zabala

Tutor: Lenin Guillermo Lemus Zúñiga

2015/2016

Resumen

El coste de la energía eléctrica sigue incrementando día a día, por lo que las facturas relacionadas con el consumo eléctrico de las empresas se ven incrementadas. Esta alza de la electricidad es responsable de que una cantidad importante de Pymes sean inviables. Por otro lado, las TIC presentan una herramienta que sirva para monitorizar en tiempo real el consumo eléctrico de dichas Pymes. A partir de esta monitorización se pueden proponer mejoras que ayuden al ahorro energético. El objetivo del presente trabajo es utilizar herramientas de Internet de las Cosas (IoT) para almacenar la información referente al consumo eléctrico y a mostrarlo de forma eficiente para que los altos cargos de las Pymes puedan tomar medidas que fomente el ahorro energético. En concreto se utilizará un coreógrafo de procesos diseñado en el Instituto ITACA como núcleo del proyecto y se utilizarán como parte novedosa tarjetas Raspberry con el S.O. Windows 10 IoT. La metodología a seguir es la siguiente:

1) Se realizará un estudio de mercado referente a monitores de red.

2) Se realizará una selección del monitor de red apropiado a nuestras necesidades de monitorización.

3) En una Raspberry se instalará y configurar el SO Windows 10 IoT para que se comunique con el monitor de red seleccionado, si es necesario se implementarán el o los protocolos de red necesarios para realizar dicha comunicación.

4) En un servidor se pondrá en marcha el coreógrafo para que se encargue de recibir la información de la Raspberry y proceda a almacenar, procesar y generar alarmas en base a los datos monitorizados.

5) Los datos se almacenarán en BBDD.

6) Utilizando herramientas de inteligencia de negocio, se mostrarán resúmenes de la información recibida/almacenada.

Palabras clave: Sistema de monitorización basado en IoT, coreografía de procesos, visualización de datos, Windows, Raspberry PI 2, Pymes.

Abstract

The cost of electricity continues to increase day by day, so the invoices related to the electricity consumption of companies increases daily. This increase of electricity is responsible for a significant number of SMEs are impractical. On the otherhand, ICTs have a too that serves to monitor real-time power consumption of these SMEs. From this monitoring can be proposed improvements that help to save energy. The aim of this work is to use tools Internet of Things (IoT) for storing information relating to power consumption and efficiently show to senior officials of SMEs can take steps to promote energy saving. Specifically designed a choreographer process in the ITACA Institute as the core of the Project will be used and will be used as part Raspberry new cards with the operation system Windows 10 IoT. The methodology followed is:

1) A market study be conducted relating network monitors.

2) A selection of appropriate network monitor will be used to our needs monitoring.

3) In a Raspberry will be installed and configured the OS Windows 10 IoT to communicate with the selected network monitor, if necessary or required network protocols be implemented for such communication.

4) On a server will start the choreographer to be responsible for receiving information from the Raspberry and proceed to store, process and generate alarms based on the monitored data.

5) Data will be stored in databases.

6) Using business intelligence tools are summaries of information received / stored Display.

Keywords: Monitoring system based on IoT, process choreography, data visualization, Windows, Raspberry PI 2, SMEs.

Índice de abreviaturas

RTU (*Remote Terminal Unit*), Unidad terminal remota.

PLC (*Programmable Logic Controller*), Controlador lógico programable.

TTL (*Transitor-Transistor Logic*), Lógica de transistor a transistor.

FTDI (*Future Technology Devices International*), Futuro dispositivo de Tecnología Internacional.

IDE (*Integrated Development Environment*), Entorno de desarrollo integrado.

P2P (*Peer-to-Peer*), Red entre pares.

SBC (*Single Board Computer*), Ordenador de placa única.

Half-Duplex (*Half-Duplex*), Semidúplex.

OS (*Operation System*), Sistema Operativos.

OSMC (*Open Source Media Center*), Centro de Medio de Código Abierto.

CRUD (*Create, Read, Update, Delete*), Crear, Leer, Actualizar y Borrar.

XAML (*Extensible Application Markup Language*), Extensible Application Markup Language.

WPF (*Windows Presentation Foundation*), Windows Presentation Foundation.

ARM (*Advanced RISC Machine*), *Advanced RISC Machine*.

IoT (*Internet of Things*), Internet de las cosas.

UAP (*Universal App Platform),* Plataforma de Aplicación Universal.

UART (*Universal Asynchronous Receiver-Transmitter*), Transmisor-Receptor asincrono universal.

GPIO (*General Purpose Input/Output*), Entrada/Salida de propósitogeneral.

ACID (*Atomicity, Consistency, Isolation and Durability*), Atomicidad, Consistencia, Aislamiento y Durabilidad.

BBDD (*Data Base*), Base de Datos.

LE (*Little Endian*), Little Endian.

BE (*Big Endian*), Big Endian.

EBIS (*Enterprise Business Intelligence Suite*), Suites Empresariales de Inteligencia de Negocios.

BI (*Bussiness Intelligence*), Inteligencia de Negocios.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

En el siguiente trabajo se expone el desarrollo de una aplicación para la RPI2, es un acrónimo de Raspberry PI 2, en la cual se instalará una versión de Windows IOT para almacenar toda la información obtenida en la nube.

En este trabajo se desarrollarán dos aplicaciones, una que actuará como servidor y la otra como cliente. En la aplicación servidor, se ejecutará un servicio de coreografía la cual le indicará las órdenes necesarias a la RPI2 para que esta empiece a funcionar de forma autónoma. En la RPI2 correrá otra aplicación más sencilla con otro servicio de coreografía donde recibirá las órdenes del servidor y a su vez se encargará de la comunicación con el analizador de red para obtener los valores leídos en ella.

1.1. Contexto y motivación

Hace un tiempo me llamó mucho la atención poder realizar un proyecto en el cual se mezclen diferentes tecnologías, como es en este caso, donde tenemos una parte software y otra parte hardware.

La parte de software es un tema que cada día que pasa me gusta más y más, esto hace que me resulte fácil de entender, tanto los lenguajes utilizados hoy en día inclusive solventar los posibles errores que surgen a la hora de hacer algún programa.

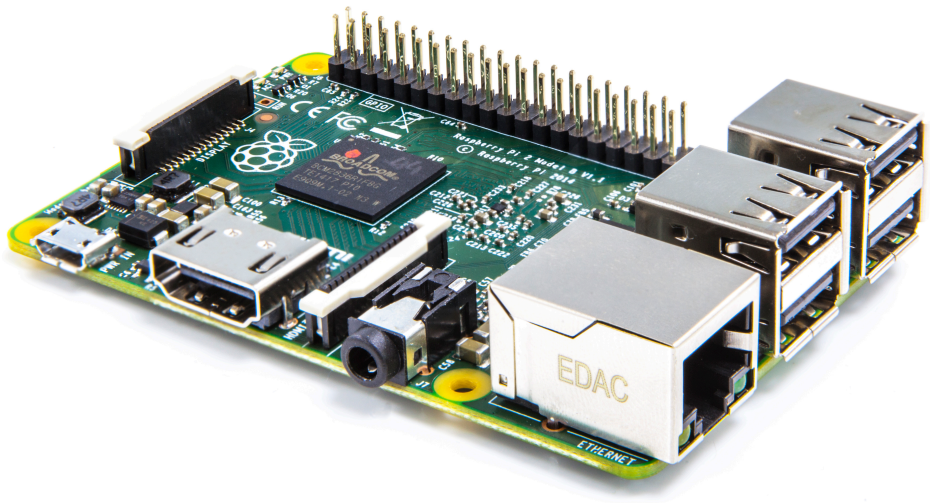
En cuanto a la parte, hardware, para mí era todo un reto, porque apenas he interactuado con esta parte de la informática, esto ha hecho que me resulte de gran interés realizar este proyecto y poder combinarlo con la parte software.

Dentro del tipo de proyecto elegido, la principal motivación fue la de trabajar con una RPI2, un mini ordenador con unas grandes características las cuales son ideales de explotar, como su programación y la forma en la que había que hacer encajar todas las piezas. Esto ha hecho que me resulte algo muy atractivo aparte de tener cierto grado de dificultad.

1.2. Hardware

Este proyecto se ha realizado con la interacción de tres componentes hardware. Los cuales vamos a describir a continuación:

Por una parte tenemos la *RPI2 Model B+ [1].*



SoC	Broadcom BCM2835
CPU	ARM11 ARMv6 700 MHz
Overclocking	Sí, hasta velocidad Turbo; 1000 MHz ARM, 500 MHz core, 600 MHz SDRAM, 6 overvolt. de forma segura
GPU	Broadcom VideoCore IV 250 MHz. OpenGL ES 2.0
RAM	1 GB LPDDR2 SDRAM 450 MHz
USB 2.0	4
Salidas de vídeo	HDMI 1.4 @ 1920x1200 píxeles
Almacenamiento	microSD
Ethernet	Sí, 10/100 Mbps
Tamaño	85,60x56,5 mm
Peso	45 g
Consumo	5v, 900mA, aunque depende de la carga de trabajo de los 4 cores

Figura 1: Modelo RPI2.

En las páginas posteriores explicaremos el motivo por el cual se ha seleccionado esta *single board computer* o *SBC.*

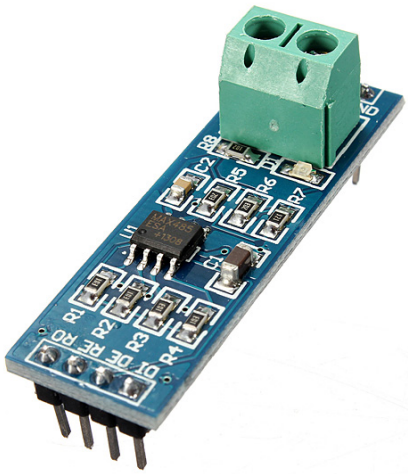
El segundo componente hardware que usamos en este proyecto es el *Circutor CVM-Mini [2].*

Medición de corriente .../5 ó .../1 A
Formato carril DIN de tan solo 3 módulos
Montaje en panel 72 x 72 mm con frontal adaptador (M5ZZF1)
Comunicación RS-485 (Modbus-RTU)
Dispone de dos salidas de transistor
Con tecnología ITF: protección de aislamiento galvánica, según tipo
Selección de parámetros a visualizar
Selección de página por defecto
Alimentación universal para tipo Plus
Precintable

Figura 2: Modelo Circutor CVM-Mini.

Este dispositivo será el encargado de leer el consumo que se lleve a cabo en una nave industrial, casa, etc. y para poder acceder a estos datos necesitamos comunicarnos a él mediante el protocolo Modbus sobre una conexión *RS-485*.

Para llevar a cabo esta conexión RS-485 hemos decidido usar un módulo de Arduino llamado *RS-485 Module for Arduino (MAX485) [3].*



Voltaje de trabajo	5v
Tamaño	44 x 14 mm
Chip	MAX485
Se pueden conectar múltiples unidades a la misma red RS-485.	

Figura 3: Modelo RS-485.

Gracias a este pequeño hardware podremos comunicarnos con el CVM Mini, pedirle los registros que queremos leer y recibir el contenido de estos registros.

Aunque la comunicación es *Half-Duplex* tenemos que esperar un par de ms desde que enviamos la petición y así dejar el canal libre para recibir la respuesta.

1.3. Sistemas operativos

La SBC RPI2 soporta un gran número de sistemas operativos [4], desde SO oficiales que se pueden descargar en la página oficial de Raspberry y las no oficiales, estas se pueden descargar desde sus páginas web.

La RPI2 puede usar sistemas operativos para uso de PC/Servidor o como OSMC para ejecutar contenido multimedia.

Estas serían algunas de las distribuciones que se pueden usar en la RPI2:

Oficiales:

- RASPBIAN Debian Jessie.
- RASPBIAN Lite Debian Jessie.
- PIDORA FEDORA REMIX.
- OSMC.
- UBUNTU MATE.

No Oficiales:

- ARCH LINUX.
- OPENSUSE.
- MINIBIAN.
- MINIBIAN-WIFI.
- UBUNTU 16.04.
- WINDOWS 10 IOT.
- CHROMIUM OS.

En definitiva, la RPI2 contiene una gran cantidad de sistemas operativos que se pueden instalar en ella, dependiendo de las necesidades que busquemos nos puede servir uno más que otro.

La versión que se va a usar es este proyecto es la versión de Windows IoT [5], esto es debido a que esta versión es la más novedosa y es algo nuevo a desarrollar.

1.4. Lenguajes

La RPI2 soporta muchos tipos de lenguajes, especialmente, estos dependerán del sistema operativo que esté instalado aparte de la funcionalidad que queremos obtener. El más usado se puede decir que es *Python*. Otros lenguajes también soportados son *TinyBasic*, *C*, *Perl* y *Ruby* por ejemplo.

En nuestro caso al usar una distribución de *Windows IOT*, el proyecto será realizado completamente en *C#*.

Como se ha mencionado antes, el sistema de coreografía también está hecho en *C#* por lo que esto nos ahorra tiempo en investigar métodos para integrarlo con otro tipo de lenguaje que no sea *C#*.

Por otra parte, se hará uso de una pequeña base de datos (*SQLite)* con la que almacenaremos los valores que vayamos recibiendo del *CVM Mini,* y para ello necesitamos realizar las funciones básicas de toda base de dato *CRUD*.

1.5. Objetivos

En el ámbito personal, los objetivos perseguidos son los de poder aplicar los conocimientos aprendidos durante mi estancia en la escuela técnica superior de informática (ETSINF) haciendo uso de documentación, investigación y adquiriendo conocimientos sobre sistemas embebidos, además del uso de la coreografía para la comunicación entre las diferentes aplicaciones.

Adicionalmente, la implementación de las aplicaciones requería el uso de librerías especificas desarrolladas en Windows 10, ya que la RPI2 llevara instalado la versión de Windows IoT y para poder desarrollar todo este conjunto se ha hecho uso del Visual Studio 2015.

El desarrollo de este proyecto en el que se han unido diferentes partes tanto hardware como software, ha hecho que se tengan que superar ciertos problemas a la hora de su desarrollo, pero el hecho de superarlos satisfactoriamente hace me enorgullezca por la formación recibida en este centro.

2. Conceptos Previos

Con la finalidad de facilitar la comprensión al usuario se usará una serie de términos o siglas que se irán describiendo en el presente documento, las siglas usadas serán definidas al inicio de este documento, en el apartado *Índice de Abreviaturas*.

2.1. Introducción al software empleado

Mayoritariamente el código está realizado en C#, pero en el diseño de la interfaz tanto la parte del servidor como del cliente, se hace uso del lenguaje *XAML*.

En este apartado vamos a explicar las principales características de los lenguajes que intervienen en el proyecto:

- *C# [6]*: Es un lenguaje de programación orientado a objetos, desarrollado y estandarizado por Microsoft como parte de su plataforma *.NET*. Su sintaxis deriva de C/C++, pero la parte de objetos es muy similar a la de Java, aunque posee mejoras derivadas de otros lenguajes.

  C# es auto contenido, esto quiere decir que elimina muchos elementos añadidos por otros lenguajes para facilitar su uso y comprensión.

  Todos los tipos base que posee, provienen de la clase base *Sistem.Object*, por lo que heredarán todos los miembros definidos de esta clase.

  También posee otras herramientas, como la recolección de basura, esto quiere decir que no hace falta incluir instrucciones de destrucción de objetos, ya que lo hace automáticamente.

- *XAML [7]*: Es un lenguaje declarativo basado en *XML*, esto surgió porque con *Windows Form* no era fácil separar la capa de presentación del código, por lo que al crear un nuevo formulario se estaba definiendo en la zona de código.

  Ahora con *WPF* se soluciona el problema que se generaba con *Windows Form*. Cuando se diseña una ventana nueva esta ya no va acoplaba al código por lo que son totalmente independientes, la interfaz se serializa en *tags XAML y* cuando se compila la aplicación los ficheros *XAML* se transforman en fichero *BAML* que estos a su vez se incrustan como un recurso en un ensamblado DLL o EXE.

- *SQLite [9]*: Base de datos muy ligera con la que podremos trabajar sin conexión en el caso de que la red se caiga o se corte la conexión a internet por cualquier otro motivo, además su configuración es realmente sencilla.

El motor de la base de datos es una biblioteca de procesos que no proporciona un servidor como tal, sino un motor transaccional, el gran impacto que ha tenido esta base de datos en el desarrollo de aplicaciones móviles se debe a su portabilidad y su pequeño tamaño.

Es una BBDD fiable, no tiene dependencias y ocupa poca memoria en ejecución.

2.2. Introducción al hardware empleado

Como se ha visto en el punto 1.2, se ha descrito el hardware que será usado en el desarrollo de este proyecto. Pero en este apartado añadiremos los elementos que se usan para conectar el hardware usado.

Tan solo nos hará falta un total de 6 cables *Macho-Hembra Protoboard*.

Figura 4: Cable Macho-Hembra.

3. Windows IoT

Windows IoT es una versión reducida de Windows 10 que esta optimizada para pequeños dispositivos, ya sea una RPI2, MinnowBoard MAX o Arrow DragonBoard 410c.

3.1. ¿Qué es Windows IoT?

Windows 10 IoT nos brinda la posibilidad de hacer grandes cosas en nuestros dispositivos, como interfaces de usuario, búsquedas, almacenamiento online o servicios de nube.

Dado que es una versión limitada de Windows 10 no cuenta con muchas opciones, ya que no las necesita. Se trata de una edición para arquitecturas de 32 bits en concreto para las *ARMv7,* esta versión no cuenta por ejemplo con una versión de escritorio y tampoco podremos ejecutar aplicaciones o juegos como se hace en su versión mayor.

Figura 5: Versiones de Windows IoT.

Windows IoT es una plataforma pensada para desarrolladores, donde ellos serán los encargados de darle vida a esta versión.

Las aplicaciones desarrolladas son llamadas UAP [10], donde se podrán ejecutar en los distintos dispositivos Windows, ya sea una RPI2, un teléfono móvil con Windows Phone, Xbox, Tablets y más. A nivel de sistema operativo es fácil de mantener, además de tener una API fiable y unificada.

Figura 6: Plataformas Windows.

Cabe destacar que Windows 10 IoT no se limita solo a las aplicaciones UAP, también se pueden ejecutar aplicaciones nativas de *Win32*, pero a la hora de ejecutarlas se harán de forma remota a través de PowerShell.

Actualmente tiene algunas limitaciones, como la compatibilidad con algunos dispositivos, ya que los drivers que lleva de serie son muy básicos y esto hace que muchos dispositivos o sensores no funcionen, por lo que en algunas ocasiones se tendrán que buscar alternativas o desarrollar tú mismo los drivers necesarios para que funcionen.

Una ventaja a destacar es su precio, es totalmente gratuito y se puede descargar desde la página oficial de Windows donde te indicarán los pasos y requisitos necesarios para proceder a su instalación.

3.2. Puesta en marcha

Una vez realizado los pasos del Anexo I, ya tendremos en nuestra SD la versión de Windows IoT para la RPI2, por lo que ahora vamos a proceder con la puesta en marcha del sistema y para ellos deberemos de seguir los siguientes pasos:

- Insertaremos la SD en RPI2.
- Conectaremos el cable HDMI al Monitor/Televisor.
- Podremos el cable de red.
- Y por último conectaremos la RPI2 a la luz.

En la página web nos recomiendan que se conecte la RPI2 vía Ethernet, dado que es más fiable que un acceso vía Wifi.

Al realizar los pasos previos iremos a la aplicación del DashBoard y seleccionaremos la pestaña ***Mis Dispositivos.***

En ella nos saldrá el dispositivo que hemos configurado e instalado en el paso previo. Aquí veremos el dispositivo y lo podremos volver a configurar, renombrarlo o cambiarle la contraseña de nuevo.

Tenemos dos formas de acceder, una visual y otra en modo terminal, Para usar el modo visual basta con pulsar el botón derecho sobre el dispositivo y seleccionar ***Abrir en Device Portal*** y para acceder en modo terminal seleccionaríamos iniciar ***PowerShell***, tanto en un modo como en otro, podremos realizar las mismas acciones.

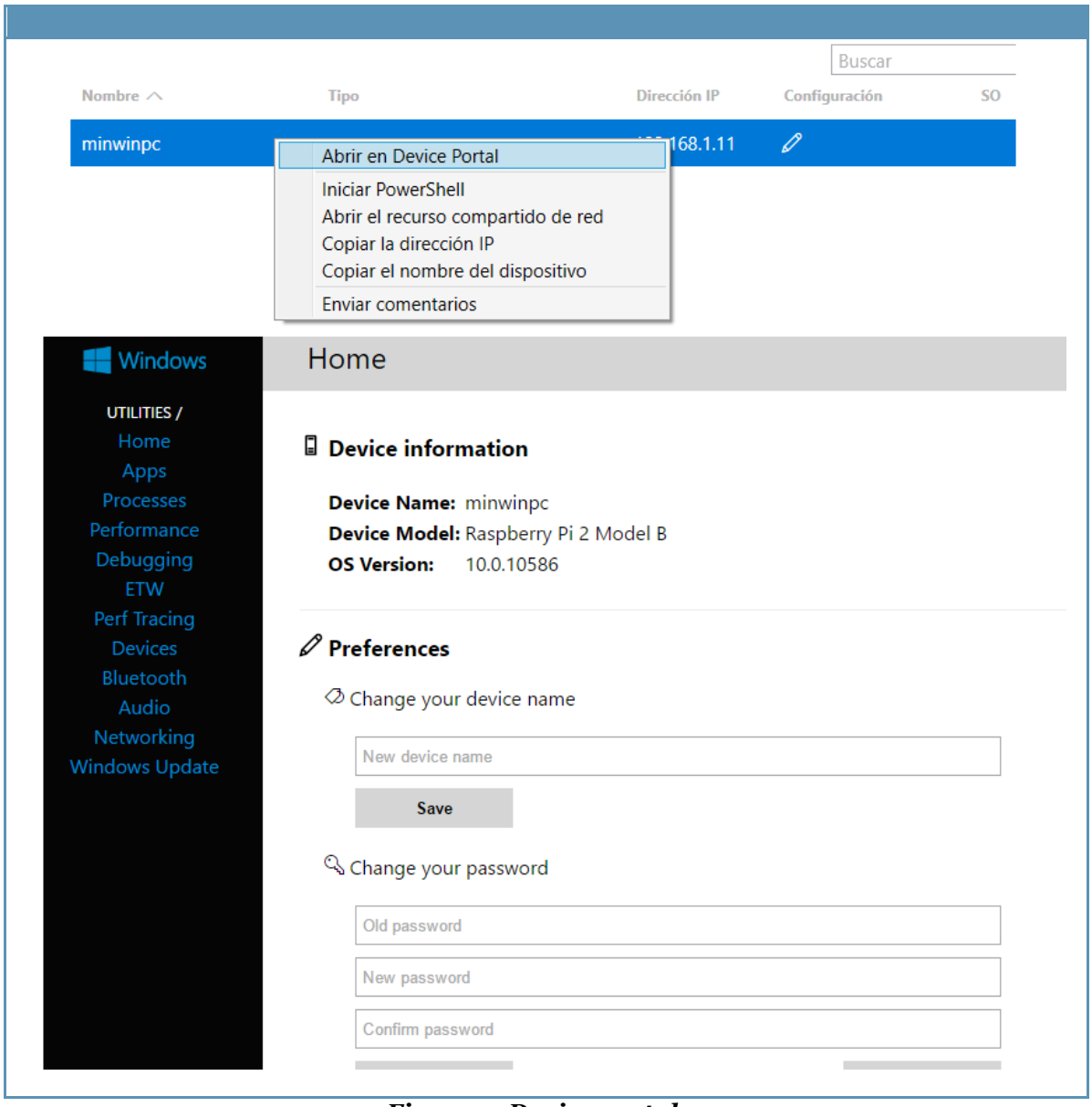


Figura 7: Device portal.

Con esto tendremos la RPI2 lista para instalar las aplicaciones que tengamos desarrolladas.

3.3. Programa Desarrollado Interfaz

Para este proyecto se han realizado dos tipos de interfaz:

- La primera interfaz está desarrollada para la RPI2, esta interfaz la veremos con detalle en el Anexo III, donde está explicado al detalle las partes que la componen. Esta se encargará de comunicarse con el analizador de red para la toma de datos y su posterior almacenamiento.
- La segunda aplicación se instalará en un PC Windows, la cual será la encargada de iniciar la aplicación que se ejecuta en la RPI2.

Las dos aplicaciones están realizadas íntegramente en C#, no hay mezcla de lenguajes ni nada por el estilo, se decidió hacerlo así, dado que en Windows IoT se pueden realizar aplicaciones en C#, además de ser un código fácil y legible de entender.

4. Raspberry PI

Raspberry PI es un ordenador de placa reducida (SBC) de bajo coste, desarrollado en Reino Unido por la fundación Raspberry PI con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Hasta la época se han desarrollado diferentes modelos de esta placa, desde el primer modelo que llevaba de serie 256 MB de RAM, hasta las actuales que llevan 1 GB y 4 puertos USB.

4.1. Historia

En 2006 [11] surgieron los primeros diseños de Raspberry PI y estos se basan en el micro controlador Atmel ATmega644, tanto sus esquemas como el diseño del circuito impreso están disponibles para su descarga pública.

La idea de este dispositivo era la de acercar más a los niños a que aprendieran a usar, manipular este tipo de dispositivos y de ahí surgió la idea de crear la fundación Raspberry PI como una asociación caritativa regulada por la comisión de caridad de Inglaterra y galés.

El objetivo principal es que los niños puedan llegar a entender el funcionamiento básico del ordenador de forma divertida y sean ellos mismos lo que desarrollen y amplíen sus dispositivos.

Eben Upton fue el precursor de esta idea y gracias a este gran dispositivo está actualmente en constante desarrollo.

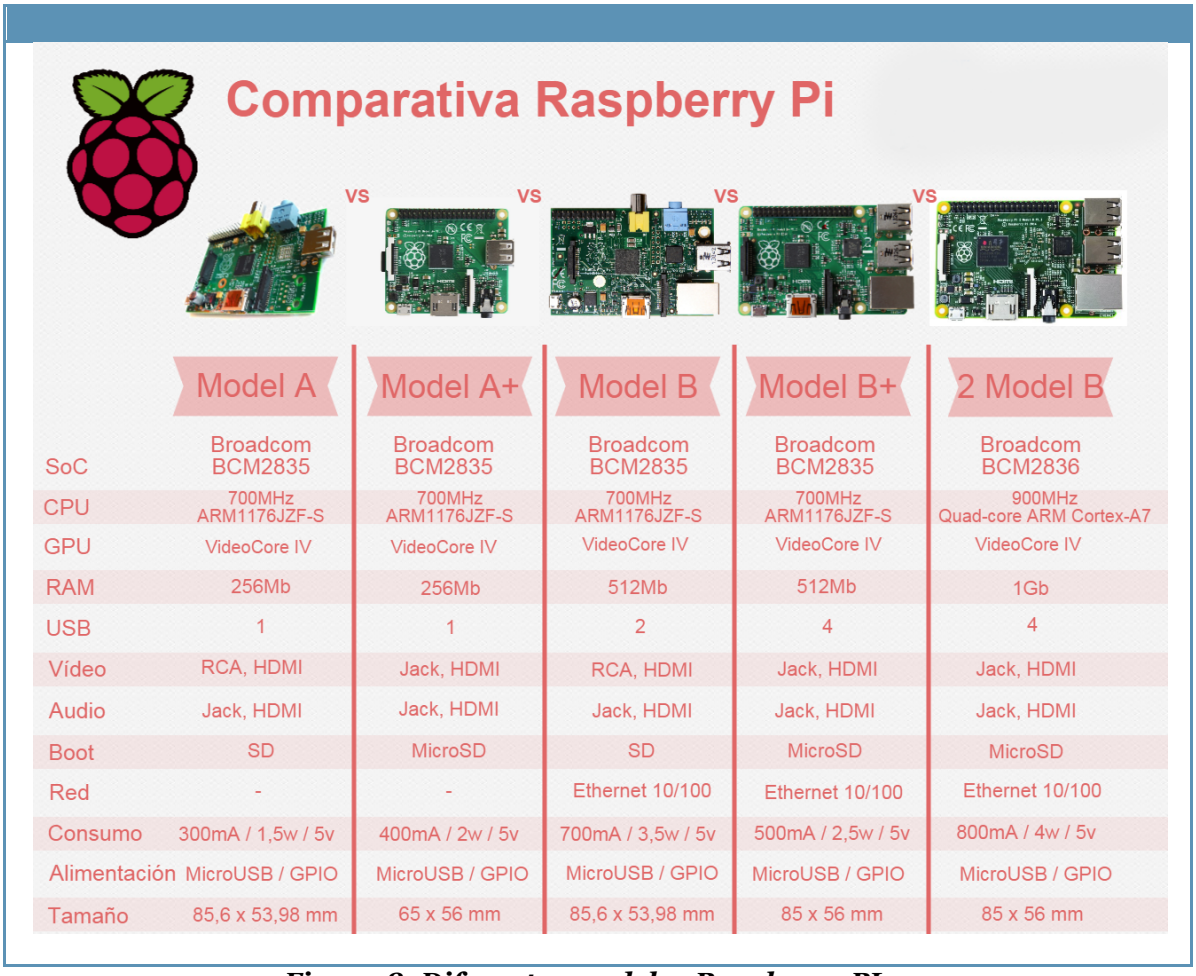
La fundación también da soporte técnico y desde su página web principal están disponibles para su descarga multitud de software compatible con este pequeño dispositivo tal y como se ha visto en el ***punto 1.3.***

El primer prototipo de este peculiar aparato era en forma de USB y en uno de los extremos tenía el puerto HDMI y en el otro el puerto USB.

En los años posteriores los modelos fueron cambiando, el modelo A fue desarrollado a mediados de verano de 2011 y de esta versión se fabricaron unas cincuenta placas mientras que el modelo B que fue lanzado en invierno de 2011 donde se ensamblaron unas veinticinco placas.

El primer lote de estas placas se fabricó en Taiwán China, dado que los costes de fabricación y la rapidez en lo que lo hacían eran menor que fabricarlos en Reino Unido. Con este ahorro pudieron invertir más en investigación y desarrollo.

En los seis meses siguientes se habrían vendido sobre las 500.000 unidades de esta diminuta placa, esto era un gran comienzo para esta fundación. En enero del 2015 ya se han vendido sobre las 5 millones de unidades vendidas y al parecer sigue creciendo al introducir y mejorar nuevos componentes y modelos al mercado.



	Model A	Model A+	Model B	Model B+	2 Model B
SoC	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836
CPU	700MHz ARM1176JZF-S	700MHz ARM1176JZF-S	700MHz ARM1176JZF-S	700MHz ARM1176JZF-S	900MHz Quad-core ARM Cortex-A7
GPU	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV
RAM	256Mb	256Mb	512Mb	512Mb	1Gb
USB	1	1	2	4	4
Vídeo	RCA, HDMI	Jack, HDMI	RCA, HDMI	Jack, HDMI	Jack, HDMI
Audio	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI
Boot	SD	MicroSD	SD	MicroSD	MicroSD
Red	-	-	Ethernet 10/100	Ethernet 10/100	Ethernet 10/100
Consumo	300mA / 1,5w / 5v	400mA / 2w / 5v	700mA / 3,5w / 5v	500mA / 2,5w / 5v	800mA / 4w / 5v
Alimentación	MicroUSB / GPIO	MicroUSB / GPIO	MicroUSB / GPIO	MicroUSB / GPIO	MicroUSB / GPIO
Tamaño	85,6 x 53,98 mm	65 x 56 mm	85,6 x 53,98 mm	85 x 56 mm	85 x 56 mm

Figura 8: Diferentes modelos Raspberry PI.

La figura anterior nos enseña algunos de los modelos que han sido desarrollados, aunque en la imagen no sale el último modelo que se ha desarrollado, este es la Raspberry PI 3, este modelo cuenta con un procesador hasta diez veces más rápido que la Raspberry original y un 50 % más rápido que el modelo 2, su procesador funciona a 1,2 GHz y es de 64 bits. La memoria RAM se mantiene en 1GB, además la nueva versión viene con Wifi y Bluetooth de serie.

4.2. Tipo de conexiones

La RPI2. Por ejemplo, cuenta con un gran número de conexiones y no tiene nada que ver con el primer modelo.

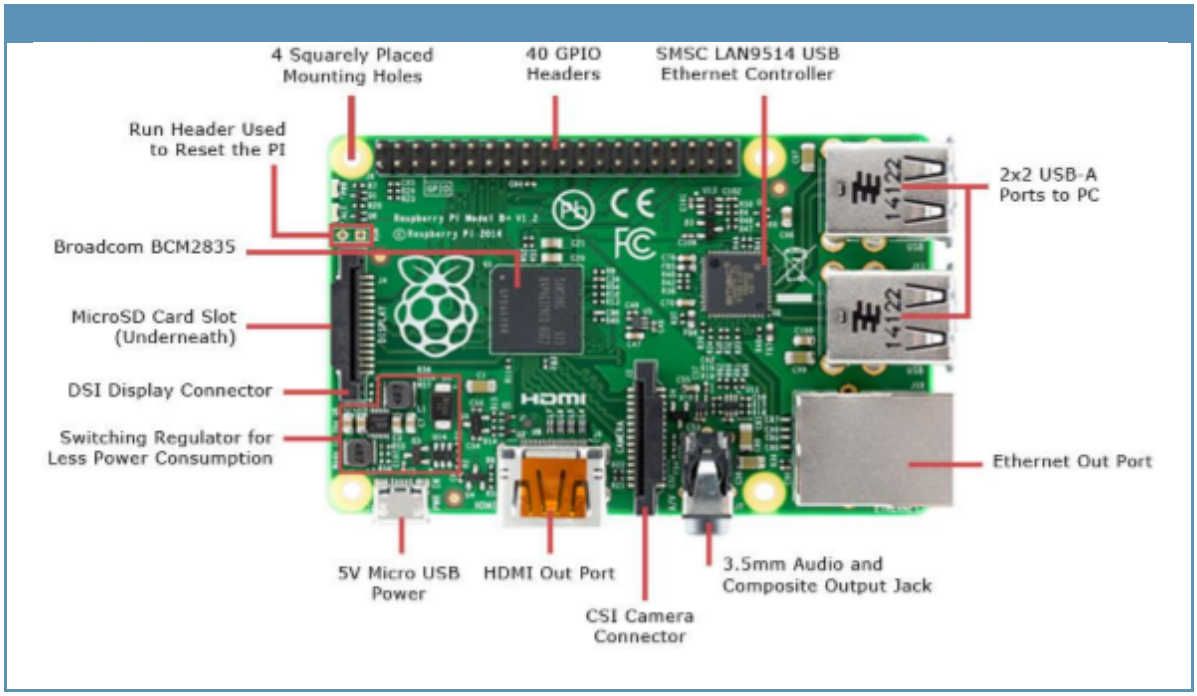


Figura 9: Conexiones Raspberry PI.

Se pueden distinguir diferentes tipos de conexiones, pero nosotros vamos a hacer más hincapié en los pines GPIO de entrada y salida, para poder utilizar los puertos UART. Así podremos realizar la comunicación serie para comunicarnos con el CVM Mini.

Ahora vamos a describir los puertos que se han usado en el siguiente proyecto, se ha tenido que usar diferentes puertos para poder realizar la comunicación con el chip *Max485*. Para ello en la siguiente tabla indicaremos los puertos usados:

Nombre en RPI2	Nombre en Chip	Puerto
VCC	VCC	GPIO3
GND	GND	GPIO6
TX	DI	GPIO14
DE	DE	GPIO23
RE	RE	GPIO24
RX	RO	GPIO15

Tabla 1: Relación de las conexiones entre RPI2 y Chip.

En la tabla anterior tenemos:

- los puertos GPIO 14 y 15, estos serán los encargados de realizar la comunicación en serie tanto TX como RX.
- los puertos GPIO 23 y 24, serán los encargados de permitir que la comunicación se realice en modo seguro, dado que la comunicación es en modo Half-Duplex no podemos transmitir y recibir al mismo tiempo.
- Los puertos GPIO 1 y 6, se encarga de suministrar energía al chip, como el chip funciona a 5v debemos de conectarlos al GPIO 1 y el GND al 6.

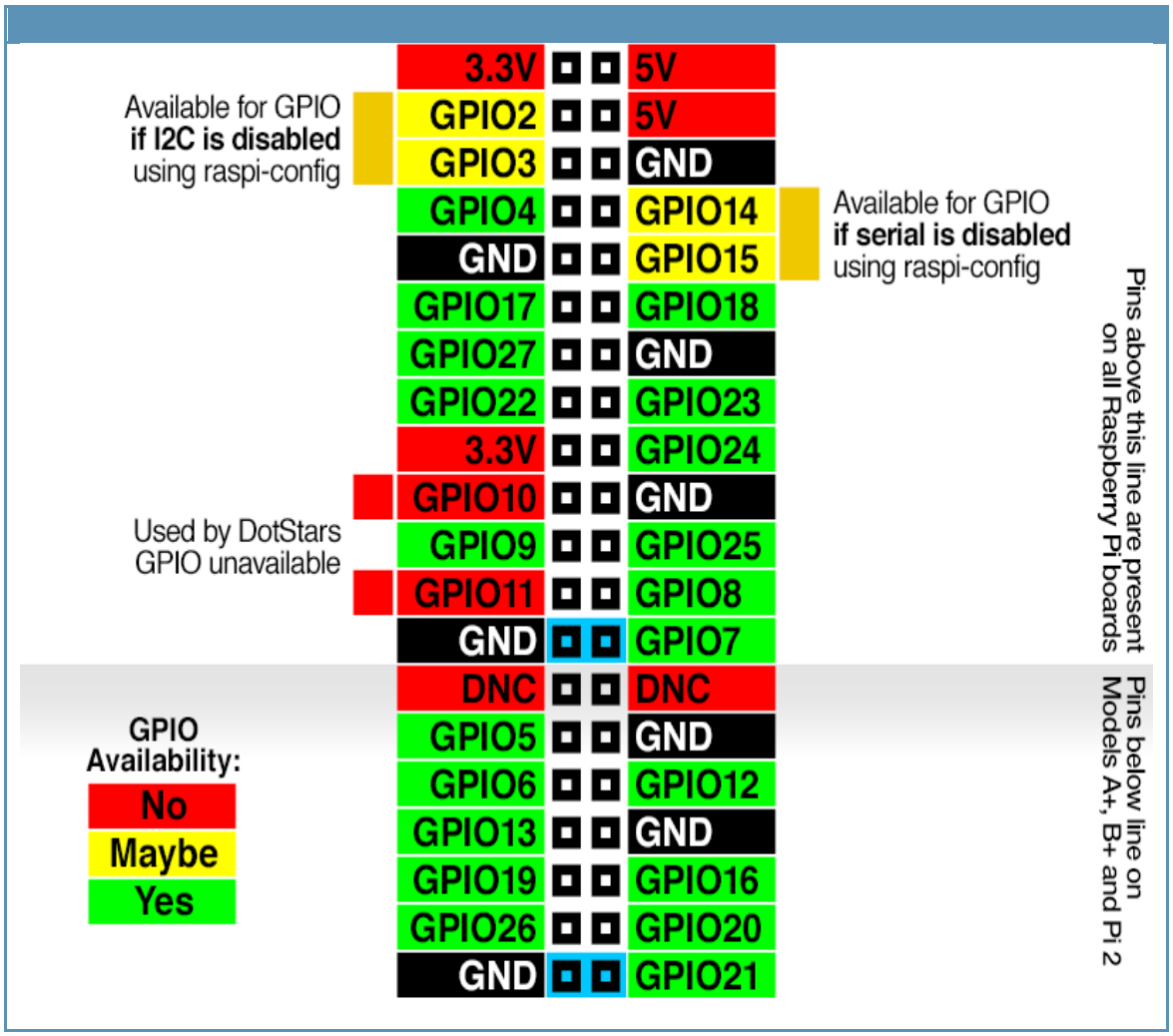


Figura 10: Conexiones GPIO en la RPI2.

Debemos de asegurarnos que el modo de funcionamiento de las señales GPIO 23 y 24 nunca tengan los valores 1 y 0, ya que es un estado inconsistente y pondría el canal en modo Full-Duplex, cosa que en nuestro caso no queremos, si se da este caso, no sabríamos si los valores que recibimos del CVM Mini son los correctos por lo que tenemos que evitar pasar por ese estado.

En el siguiente diagrama de estado, explicaremos los casos o estados por los que pasaremos en la aplicación.

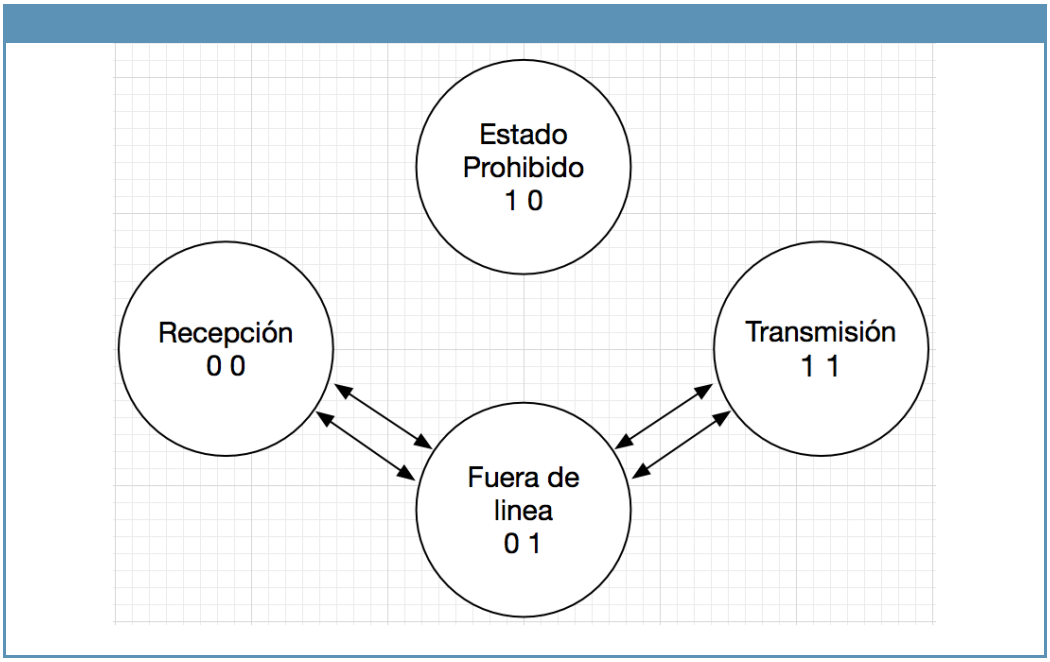


Figura 11: Diagrama de estados.

Se puede apreciar que los valores que han de tomar los pines GPIO 23 y 24. En ningún caso el pin GPIO 23 debe tomar el valor 1 y el ping GPIO 24 debe de tomar el valor 0.

Estados	GPIO23	GPIO24
Recepción	**0**	**0**
F. de Línea	**0**	**1**
Prohibido	**1**	**0**
Transmisión	**1**	**1**

Tabla 2: Estados del sistema.

El ***estado inicial*** de la aplicación es el estado de ***recepción***.

5. Analizadores de Red

Para poder realizar este proyecto se ha realizado un análisis de mercado sobre los analizadores de red, primero necesitamos saber las especificaciones que nos ofrecen, el precio y algunos campos más que ya iremos describiendo a lo largo de este apartado.

Nosotros buscábamos un analizador capaz de analizar redes trifásicas, para que nos indique el consumo de energía eléctrica en las diferentes líneas que estén conectadas. Por otra parte, también necesitábamos que el producto seleccionado se pueda comunicar con la RPI2 a través del protocolo RS-485 [12].

El resultado del análisis lo iremos resumiendo en los siguientes puntos.

5.1. Modelos existentes

En la actualidad podemos encontrar una gran multitud de analizadores de red, ya sean portables, fijos, de redes monofásicas o trifásicas, etc.

Como hemos mencionado anteriormente, nosotros buscábamos un analizador con carril DIN, para posicionarlo en el cuadro de luz junto con la RPI2 y una carcasa diseñada para acoplarla al carril DIN. Por esta razón los analizadores de redes eléctricas portables no nos interesan, aparte de que tiene ciertas ventajas y es más fácil de transportar, pero estos no cumplen o no usan la comunicación RS-485.

5.1.1. Modelos portables

Por una parte tenemos los modelos portables, estos son los que más abundan en el mercado, ya que son fáciles de transportar y tan solo basta con enganchar los clips a los cables de luz para empezar a tomar medidas.

Tenemos por ejemplo un dispositivo de la marca ***Fluke*** llamado ***Serie 430*** [13], este analizador portable nos permite analizar redes trifásicas, además permite medir todos los parámetros del sistema eléctrico, como tensión, corriente, potencia, consumo (energía), desequilibrio, flicker, armónicos e ínter armónico.

Posee un registrador que nos permite detallar los datos que necesitáramos, además de permitirnos leer los valores máximos, mínimos y promedios de hasta 100 parámetros distintos en las tres fases y el neutro. Estas serían algunas de sus características más importantes, este analizador es muy completo y con un gran soporte técnico, pero también tiene un precio muy elevado, el cual para nuestro pequeño proyecto es muy descabellado, su precio estimado es de unos ***6500 €*** dependiendo del proveedor donde se adquiera.

Figura 12: Analizador de red, Fluke 435.

El siguiente componente portable, es el modelo ***VEGA78*** *[14]* de la compañía ***HT Instruments***, este modelo prácticamente compite con el de ***Fluke*** y nos permite realizar muchas acciones al igual que su competidor.

El analizador ***VEGA78*** permite leer hasta un máximo de 251 parámetros distintos, guardarlos en la pequeña memoria que lleva incorporada o añadir una memoria USB y guardar los datos allí, este dispositivo también se puede conectar al PC a través de la interfaz USB, permitiendo compartir los datos capturados para su posterior análisis.

En cuanto a sus características generales, estas son muy similares a las del Fluke, por lo que lo único destacable es su precio respecto al otro, este modelo lo podemos conseguir en el marcado sobre los 2500 €, pero añadiendo los gastos de envió, ya que lo envían desde EE.UU., su precio final rondaría los ***3000 €***.

En caso de que un analizador de redes eléctricas portables fuera lo que necesitáramos para nuestro proyecto, elegiríamos este sin lugar a dudas, ya que nos ofrece las mismas características que el ***Fluke*** pero a un precio mucho menor, además de que el soporte técnico es tan completo como el de ***Fluke***.

Figura 13: Analizador de red, HT Instruments VEGA78.

5.1.2. Modelos Carril DIN

En este apartado estudiaremos algunos de los analizadores de redes que hay en el mercado.

Las características que estamos buscando principalmente son:

- Permita comunicación RS-485.
- Analice redes trifásicas.
- Instalación en carril DIN.
- Precio reducido.
- Soporte Técnico.

El analizador que cumpla o se acerque más a las características que estamos buscando, será el que usemos en nuestro proyecto. Por lo general los tres analizadores nos permiten hacer el mismo tipo de mediciones, leen los valores máximos, mínimos y nos generan alarmas cuando se cumplen determinadas acciones que hayamos configurado previamente.

Vamos a empezar a describir el primero de los tres posibles candidatos. Este analizador de red es el modelo ***Schneider PM3250*** [15], esta compañía posee más analizadores de este tipo, pero este modelo es el único el cual lleva el puerto ***RS-485***.

El analizador ***PM3250*** tiene las siguientes características:

- nos permite la supervisión de varios parámetros eléctricos como I, In, U, V, PQS, E, PF, Hz.
- Permite un registro del consumo de energía (Diario, Semanal, Mensual).
- Comunicación Modbus.
- Gestión de hasta 4 tarifas.
- Alarmas con marcas de tiempo.
- Demanda de potencia/corriente, demanda pico.
- Mínimos/Máximos.

Como vemos en sus características este analizador nos permite realizar muchas mediciones en sus diferentes líneas y al llevar la comunicación Modbus nos permitirá acceder a esos valores sin problemas.

Por otra parte, tenemos su precio, el precio de este dispositivo ronda los 280 €, adquiriéndolo fuera de España, por lo que hay que sumar gastos de envió y esto puede llegar hasta los ***320 €.***

Figura 14: Analizador de red, Schneider PM3250.

El segundo Analizador que vamos a describir es el ***Panasonic KW9M*** [16] y sus características son:

- Pantalla LCD Grande y nítida.
- Permite la monitorización de corrientes en torno a 1 mA.
- Capaz de medir hasta tres fases simultáneamente.
- Equipado con E/S para generar alarmas, eventos y/o control.
- Funciones de almacenamiento de datos.
- Posibilidad de Data Logger.
- Conexión directa a transformadores de corriente genéricos tipo 1A/5ª.
- Comunicación Modbus.

Panasonic tiene dos versiones de este modelo, el modelo estándar y el avanzado, el avanzado permite la supervisión de la potencia demandada y la calidad de potencia.

El precio de este analizador ronda los 285 € pero si le sumamos los gastos de envió, puede llegar a costar sobre los ***310 €*** dependiendo de la compañía con la que se adquiera.

Figura 15: Analizador de red, Panasonic KW9M.

Y por último vamos a mostrar algunas de las características del analizador de red ***Circutor CVM Mini*** [17].

El ***Circutor CVM Mini*** es muy completo y más sencillo que los anteriores, haciendo que su instalación sea más fácil que las otras, ya que su tamaño es más reducido que los anteriores.

Sus principales características son:

- Display LCD retro-iluminado.
- Comunicación RS-485 (Modbus RTU).
- Sonda de temperatura en el interior del dispositivo.
- Valores instantáneos, Máximos, mínimos de cada parámetro.
- Función medidora de energía.
- Medición en verdadero valor eficaz.

El precio del ***Circutor CVM Mini*** ronda los ***400 €,*** es el más elevado de los tres pero eso no quiere decir que sea mucho mejor que los otros.

Figura 16: Analizador de red, Circutor CVM Mini.

5.2. Modelo elegido

Une vez que hemos visto los tres analizadores y sus casi parecidas características, tendremos que ver cuál de los tres es el mejor o más adecuado para realizar nuestro proyecto, en la siguiente tabla veremos cuáles de los tres cumple con las características que necesitamos.

Analizadores de redes			
	Schneider PM3250	***Panasonic KW9M***	***Circutor CVM Mini***
Comunicación RS-485	Si	Si	Si
Análisis de redes trifásicas	4, 3-trifásicas, 1-neutro	4, 3-trifásicas, 1-neutro	4, 3-trifásicas, 1-neutro
Carril DIN	Si	Si	Si
Precio	320€	310€	400€
Soporte Técnico	bueno	bueno	Muy bueno

Tabla 3: Características de los analizadores de redes.

En la tabla anterior el analizador de Panasonic es el más barato de los tres, pero su soporte técnico no es tan bueno como el de Circutor.

En nuestro caso el dispositivo que vamos a elegir para realizar el presente trabajo es el dispositivo ***Circutor CVM Mini***, aunque tiene un precio mayor que los otros dos nos conviene, porque, aparte de la comunicación ***RS-485***, podemos usar el protocolo **Modbus** (**RTU**) para comunicarnos con él y obtener los valores que necesitemos. Lo cual nos facilita la tarea a la hora de realizar la aplicación para la RPI2.

6. Circutor CVM-Mini

En la introducción de este proyecto se ha comentado algo sobre este dispositivo, ahora vamos a continuar explicando un poco más su funcionamiento además de los registros de los que hace uso.

Este analizador de red mide, calcula y visualiza los principales parámetros de redes industriales trifásicas equilibradas o desequilibradas.

La media de los valores se realiza en verdadero valor eficaz, mediante tres entradas de tensión alterna, neutra, y tres entradas de corriente. Para las medidas exteriores se toman de los transformadores de medida exteriores.

En la siguiente figura veremos algunas de las conexiones que se podrían realizar en un cuadro eléctrico.

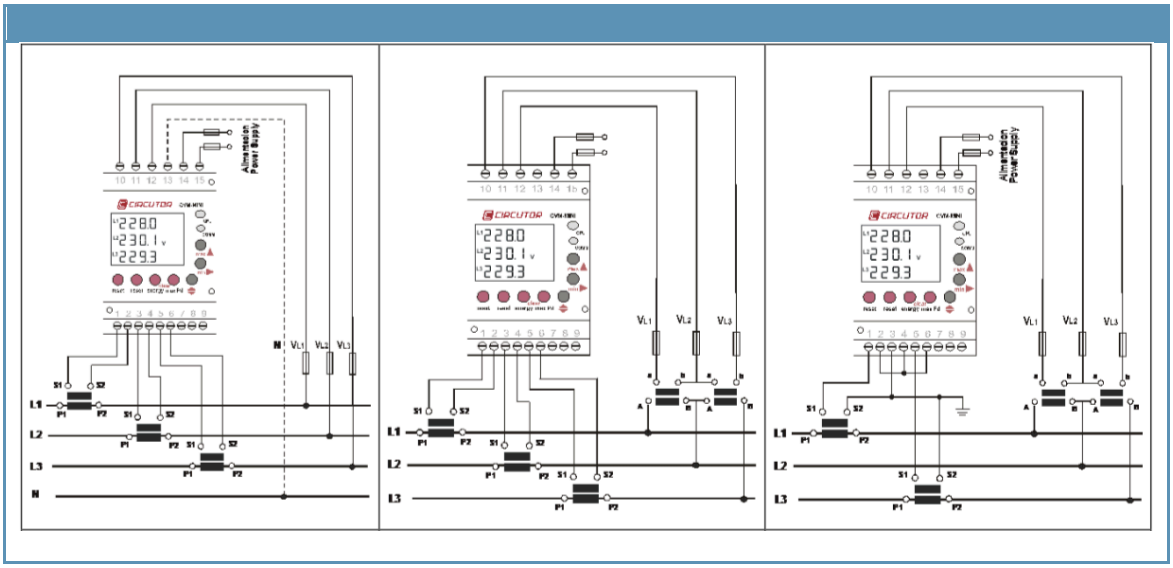


Figura 17: Esquemas de conexión.

- En la parte más izquierda de la imagen vemos una conexión tipo:
  - 4 hilos / 3 hilos – baja tensión.
- En la imagen del centro vemos:
  - 2 transformadores de tensión – 3 transformadores de corriente.
- en la imagen de la derecha vemos:
  - 2 transformadores de tensión – 2 transformadores de corriente.

El analizador CVM Mini lleva por defecto una configuración de fábrica por lo que no tenemos por qué preocuparnos la configuración del dispositivo.

Por otra parte, la para el funcionamiento a través de la comunicación Modbus se ha configurado el dispositivo para que funcione de la siguiente manera:

- Id del periférico: 0x01.
- Velocidad: 19200 bps.
- Paridad: No.
- Bits de datos: 8.
- Bits de parada: 1.

Configurando esto en el dispositivo podemos comunicarnos con la RPI2 para la obtención de datos.

Además, en el programa cuando se crea el puerto serie por el cual se va a iniciar la comunicación tenemos que inicializar los parámetros de la conexión con los datos arriba indicados, el resultado quedaría así:

```
serialPort.BaudRate = 19200;
serialPort.Parity = SerialParity.None;
serialPort.StopBits = SerialStopBitCount.One;
serialPort.DataBits = 8;
serialPort.Handshake = SerialHandshake.None;
```

Figura 18: Configuración Serial en Programa.

Toda la comunicación que va desde la RPI2 hasta el CVM Mini y viceversa se realiza mediante el protocolo ***Modbus RTU***, el cual dedicaremos un apartado para explicar los pasos realizados en el presente proyecto para su funcionamiento, porque hasta la fecha no hay una versión de este protocolo para la versión de ***Windows IoT***.

6.1. Protocolo Modbus RTU

Se ha mencionado que el proceso de comunicación corre a cargo del protocolo Modbus RTU [18] y en Windows IoT no hay una versión de este protocolo, por lo que hemos tenido que hacer una versión mini de Modbus para poder realizar las comunicaciones con la RPI2, a continuación explicaremos el proceso y los pasos que se han seguido para la creación del Frame que será enviado al CVM Mini.

El Frame de petición que tenemos que crear para poder comunicarnos con el CVM Mini ha de tener este formato.

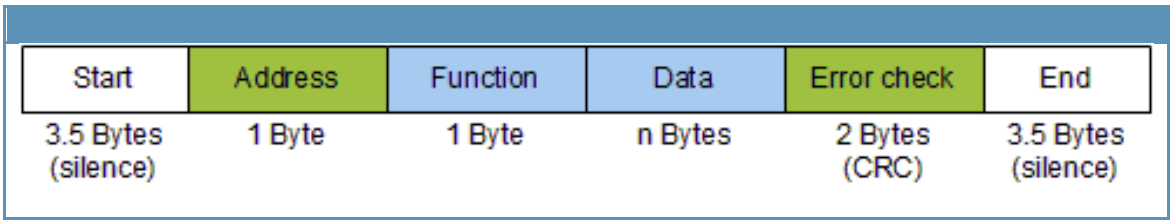


Figura 19: Frame Modbus RTU.

En la imagen de arriba podemos apreciar cómo se estructura este Frame de petición, los campos que los componen son:

- ***Start y End***: la comunicación es Half-Duplex por lo que tenemos que enviar estos dos campos, para evitar que haya interferencias en él envió del mensaje.
- ***Address***: Dirección del dispositivo al que queremos enviar la petición, en nuestro caso es **0x01**.
- ***Function***: el protocolo Modbus tiene multitud de funciones, las cuales cada una de ellas permite realizar una función diferente, en nuestro caso la función que vamos a utilizar es la función **0x03** (Leer N registros).
- ***Data***: está compuesto por dos bytes y son:
  - Byte 1: En este byte le indicaremos el registro inicial desde el cual se quiere leer.
  - Byte 2: Cuantos registros se quieren leer desde el registro indicado
- ***CRC***: El CRC se genera a partir de los valores anteriores.

El Frame de respuesta es prácticamente igual, salvo el campo Data, que multiplica por 2 los datos recibidos del CVM Mini, por ejemplo:

Cuando se hace una petición al CVM Mini se le indica que desde el registro 80 por ejemplo lea 4 registros, la respuesta de este es:

- ***Address***: 0x01.
- ***Function***: 0x03.
- ***N.º de bytes de datos*** (4*2): 8.
- ***CRC.***

Envía el doble de datos porque cada dato está formado por la parte alta y la parte baja en hexadecimal.

6.3. Como se genera el Frame en código

Ahora veremos los pasos necesarios que se han seguido en código para generar el Frame de petición, y como no el Frame de respuesta, todos los valores pasan de un entero de 16 bits a bytes para su posterior envió al CVM Mini.

Para empezar tenemos un Array global donde se irán rellenando las diferentes partes que lo componen, como se ha visto en el punto anterior el Frame está compuesto por: la dirección del dispositivo, la función que se quiere usar, el campo datos y el CRC.

Estos pasos son los que se han seguido. Para añadir la dirección del dispositivo y la función se ha realizado lo de la imagen. Se da por entendido que el Array ha sido previamente inicializado a un tamaño de 8 bytes.

```
/// <summary>
/// calcula la cabecera del array
/// pone el id del dispositivo al que se va a conectar
/// y el codigo de la funcion que se va a usar.
/// </summary>
private void añadeCabeceraIDCode()
{
    bArraySend[0] = ClassConstantes.IDSlave.ID;
    bArraySend[1] = ClassConstantes.CodigoFuncion.Code;
}
```

Figura 20: Id y Función del Frame.

En la imagen, la posición [0] del Array introduciremos el valor 0x01 que hace referencia al id del dispositivo y en la posición [1] introduciremos el valor 0x03.

En el siguiente paso tenemos que componer el campo Datos, recordar que este campo está formado por dos bytes, y uno de ellos indica el registro desde el que se quiere indicar la lectura y el otro byte indica cuantos se quieren leer. Por lo que a la hora de formar este campo tenemos el siguiente código.

Para generar el byte del registro inicial hacemos lo siguiente:

```
/// <summary>
/// Pasa de un entero de 16 bits a un array de bytes.
/// Pasa el valor del registro a un array de bytes
/// Se hace un array reverse para pasar de LE y BE
/// </summary>
/// <param name="i16Registro">Entero de 16 bits</param>
private void calculaInt16ToByteArrayRegistro(Int16 i16Registro)
{
    int iCont = 2;
    byte[] bRegistros = new byte[2];
    bRegistros = ConvertIntToByteArray(i16Registro);
    Array.Reverse(bRegistros);

    foreach (byte b in bRegistros)
    {
        bArraySend[iCont] = b;
        iCont++;
    }
}
```

Figura 21: Generar byte de REG inicial.

En la imagen anterior se puede apreciar que la función recibe como parámetro un entero de 16 bits, dentro de la función generamos otro Array de bytes con dos posiciones, esto es porque tenemos que pasar el Array que se genere al formato *Big Endian*. El parámetro que se recibe como argumento lo pasamos a un Array de bytes con la clase *BitConverter*, después hacemos un *Array.Revserse* para par de LE a BE y por último pasamos el contenido del Array a la posición [2] del Array que se va a enviar al CVM Mini.

La función que pasa el entero de 16 bits a un Array de bytes la siguiente:

```
/// <summary>
/// funcion que pasa de un entero de 16 bits a un array de bytes
/// </summary>
/// <param name="I16">entero de 16 bits</param>
/// <returns>byte[]</returns>
public byte[] ConvertIntToByteArray(Int16 I16)
{
    return BitConverter.GetBytes(I16);
}
```

Figura 22: Pasar un tipo int a un array de bytes.

Por ahora tenemos las posiciones del Array 0, 1, 2, 3. La posición [4] del Array indica la cantidad de registros que se quieren leer y se hace de la misma forma que la figura 26, por lo que no es necesario repetir el mismo código, salvo que a la hora de guardar el Array generado le tenemos que decir que lo ponga en la posición [4].

Bien, con esto ya tenemos las 6 primeras posiciones del Array completadas, ahora solo nos falta generar el CRC y tendríamos el Frame al completo.

Para generar el CRC tenemos que pasarle como argumento a su función un Array de tan solo 6 posiciones por lo que el Array que hemos generado antes tenemos que copiarlo a otro, dado que este tiene 8 posiciones.

El código resultante de la operación es el siguiente:

```
/// <summary>
/// Calcula el CRC con el array que se le pasa
/// como argumento.
/// </summary>
private void calculaCRC()
{
    int iCont = 6;
    byte[] crc;

    crc = ClaseCRC.CalculateCrc(bData);
    //Array.Reverse(crc);

    foreach (byte b in crc)
    {
        bArraySend[iCont] = b;
        iCont++;
    }
}
```

Figura 23: Cálculo del CRC.

El Array *bData* que se le pasa a la *ClaseCRC* es el resultado de copiar las 6 posiciones del Array que se va a enviar. Cuando el cálculo ha terminado el resultado se guarda en las posiciones 6 y 7 del Array.

Con esto ya tendríamos el Array al completo por lo que tan solo bastaría con enviárselo al CVM Mini para que nos devuelva el un Array con la respuesta.

El cálculo del CRC lo realiza la siguiente función:

```
/// <summary>
/// Calculate Cyclical Redundancy Check
/// </summary>
/// <param name="data">The data used in CRC</param>
/// <returns>CRC value</returns>
public static byte[] CalculateCrc(byte[] data)
{
    if (data == null)
        throw new ArgumentNullException("data");

    ushort crc = ushort.MaxValue;

    foreach (byte b in data)
    {
        byte tableIndex = (byte)(crc ^ b);
        crc >>= 8;
        crc ^= crcTable[tableIndex];
    }

    return BitConverter.GetBytes(crc);
}
```

Figura 24: Función que calcula el CRC.

Cabe destacar que la *ClaseCRC* es una clase simplificada de la que se utiliza en el Modbus original. El CRC que se calcula en la versión original es mucho más completo que la desarrollada en este proyecto.

7. Dispositivo Auxiliar

Como se ha menciona en la introducción de este proyecto, se ha hablado de un elemento muy importante que nos permite la comunicación con el CVM Mini, el cual vamos a proceder a explicar.

7.1. Chip RS-485 for Arduino

Este pequeño chip es usado porque el CVM Mini funciona con el protocolo Modbus y como se puede implementar sobre redes de comunicación RS-485, nos facilita la tarea a la hora de realizar el montaje del mismo.

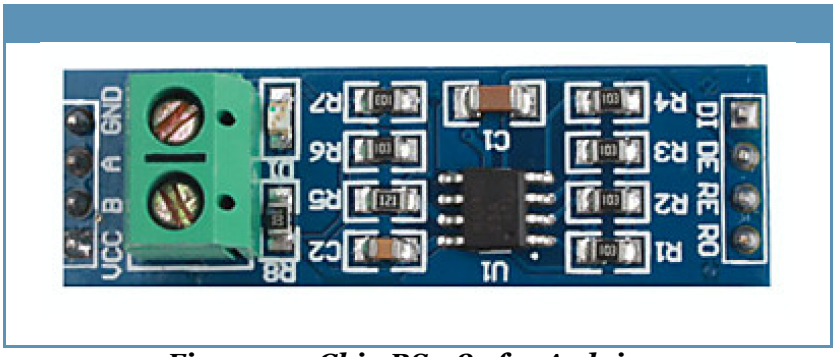


Figura 25: Chip RS-485 for Arduino.

Como vemos en la imagen, el chip tiene 8 conexiones, pero no es preciso que estén conectadas todas como es en nuestro caso.

Los pines que se conectarán son los siguientes:

- VCC – Este chip funciona con 5v por lo que se conectará al GPIO 3 de la RPI2.
- GND - Este pin ira conectado en el GPI 6.
- DI (Data in) – Se conectará en el GPIO 14.
- DE (Data Enable) – Conectado en el GPIO 23.
- RE (Receive Enable) – Conectado en el GPIO 24.
- RO (Receive Out) –Conectado en el GPIO 15.
- A y B son los pares RS-485 que irán conectados en el CVM Mini.

8. Comunicación entre los elementos

Vamos a detallar como se han conectado los dispositivos usados para este proyecto, es muy importante que estos estén bien conectados, dado que si hay algún error a la hora de conectarlos no funcionara bien, por ejemplo a la hora de enviar una petición de lectura de registros, el CVM Mini no acepte el Frame que se le haya llegado y rechace esta petición o por el contrario que le llegue la petición y en canal no esté libre y se produzca una colisión y se pierda la información.

En caso de que algún error de transmisión ocurra, tendríamos que desconectarnos del dispositivo y realizar una nueva conexión en serie para la petición de registros. Pero en el peor de los casos que se tenga que reiniciar la RPI2, por lo que la información que se tuviera en la base de datos se perdería y se tendría que volver a empezar de cero.

En la imagen siguiente veremos cómo se han conectado los distintos dispositivos.

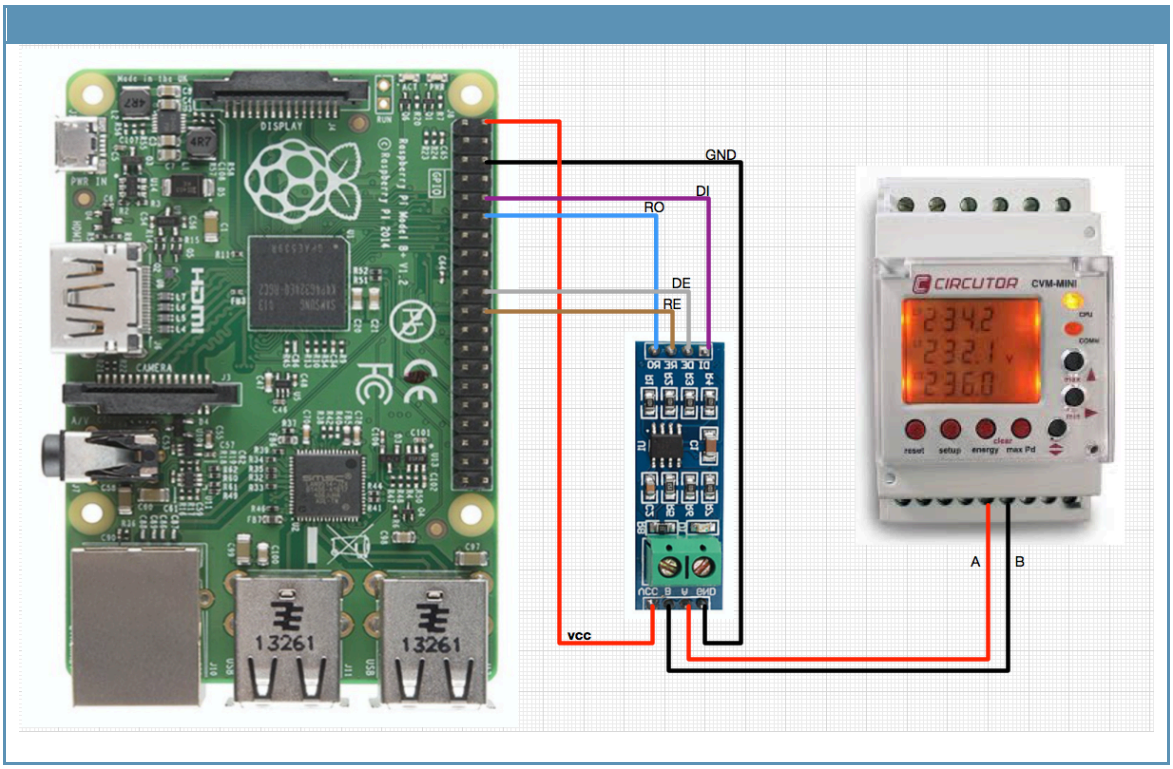


Figura 26: Conexión de dispositivos.

Se puede apreciar cómo están conectados los diferentes dispositivos que forman este proyecto, por una parte tenemos el CVM Mini del cual se conectan los pares A y B del RS-485 al chip de Arduino. El chip de Arduino está conectado a la RPI2, de ahí sacará el VCC necesario para funcionar el GND y las cuatro conexiones restantes para su correcto funcionamiento. En esta imagen se han omitido las conexiones del CVM Mini a la red eléctrica además de las conexiones usadas por la RPI2.

9. Coreografía de Procesos

El sistema de coreografía [19] es un concepto importado de la teoría de la automatización de procesos. Esto quiere decir que un servicio de coreografía es una descripción externa u observable de las interacciones P2P que existen sobre ese servicio.

La interacción entre varias parejas de procesos se puede ver como un solo proceso dado a su alta integración. Además, en la coreografía no existe un sistema central o coordinador, es un sistema totalmente distribuido.

Se podría decir que tenemos dos sistemas o formas de dirigir los procesos, por una parte tenemos la orquestación y por la otra la coreografía.

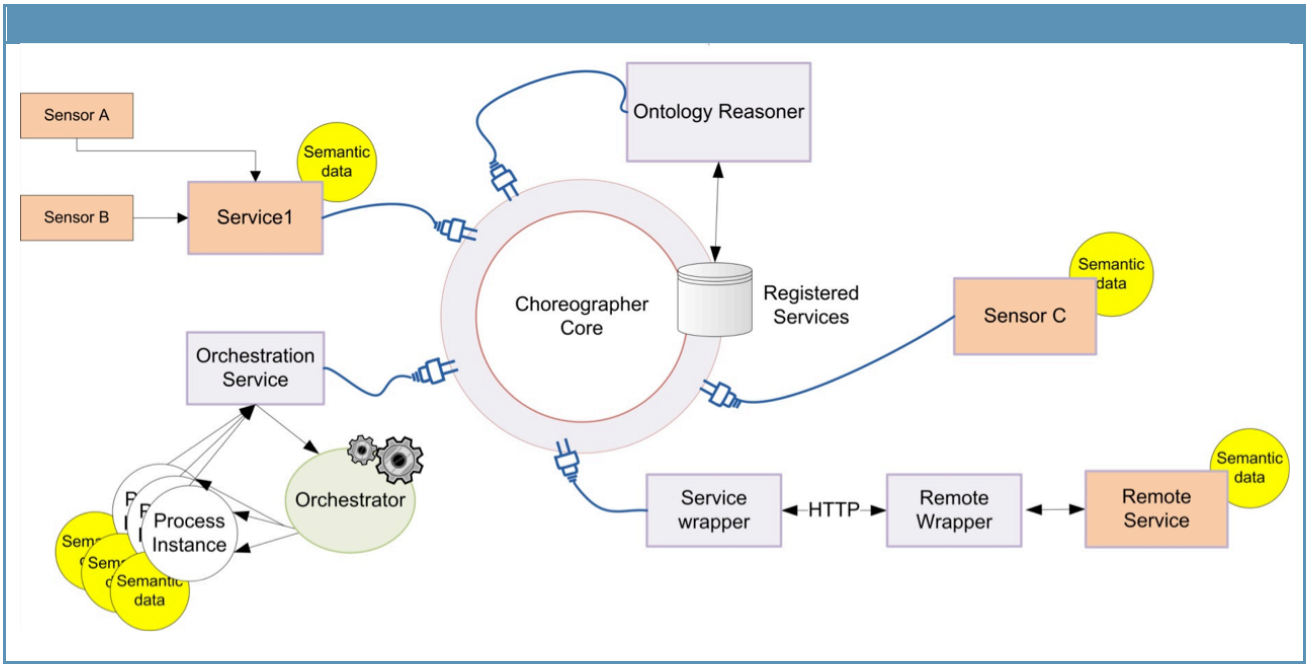


Figura 27: Ejemplo de un sistema de coreografía.

9.1. Coreografía

En la coreografía se proporciona un enfoque diferente, estos pueden tener una parte interactiva, sistemas basados en agentes y sistemas basados en eventos. Las reglas son creadas para determinar el comportamiento de cada proceso de forma individual permitiendo que multitud de procesos trabajen de forma independiente y se puedan comunicar entre ellos enviándose mensajes de peticiones, administración, errores, etc.

Para poder que estos procesos se comuniquen han de tener una dirección o id, para poder hacer un direccionamiento lógico, por norma general en este proyecto el sistema se coreografía que se ha usado hay sido desarrollado por el instituto ***ITACA*** y la forma en la que se ha diseñado para que la comunicación entre procesos funcione, es la siguiente:

Las direcciones entre servicios siguen la expresión regular:

- [id](.[id])*
    - Ejemplo – Dotnet.corografo.servicio1.

Podemos usar la multidifusión y para ello basta con añadir '*' a algún grupo de servicios o enviar solo '*':

- *: se enviaría a todos los procesos.
- *.coreógrafo: Envía a todos los servicios cuyo segundo identificador sea '*coreógrafo*'.

9.2. Orquestación

La orquestación es el que más se suele usar en la composición de servicios y procesos de negocio. Con la orquestación se define la secuencia de pasos que ha de seguir un proceso, por lo que las reglas ya están definidas en el sistema central y este se encarga de generar el proceso con estas reglas. Esto es en caso de procesos simples, para procesos más complejos se han de describir con alguna herramienta visual, para crear un modelo visual de la secuencia y luego generar el código necesario para que ejecute esa secuencia.

Los procesos expertos que son descritos con herramientas visuales también se les conocen como protocolos sin programación y estos son conocidos a su vez como ***WorkFlows***, estos son ejecutados vía ***WorkFlows Engines***.

Un ***WorkFlow*** es una automatización de un proceso de negocio durante una parte mientras algún documentos, información o tarea es pasado de un participante a otro de acuerdo con el conjunto de reglas procedurales.

Otras características de la orquestación son:

- Se define un solo control maestro de todos los aspectos del proceso.
- Soporta una vista gráfica de la secuencia.
- Se mapea fácilmente con SOA.
- Más simple de iniciar pero en los procesos complejos es difícil de hacerlo escalar.
- Representa el estado de la práctica y es soportado por la mayoría de las herramientas.

La popularidad de este enfoque es comprensible, pero no soporta todos los tipos de procesos.

9.3. Mensaje de coreografía

En el instituto ITACA han desarrollado para la plataforma .NET los mensajes de coreografía que se han usado en el presente proyecto. Un mensaje de coreografía está compuesto por la siguiente información:

- Identificador único de mensaje.
- Id de origen.
- Id de destino.
- Tipo de protocolo.
- Nombre del método: *ExecuteProcess*.
- Nombre de los parámetros.
- Tipo del parámetro.
- Valor del parámetro.

Cuando se envía un mensaje de coreografía, el objeto de respuesta es devuelto dentro de un método con el nombre ***[NombreMetodoOriginal] Response*** y el parámetro como ***[NombreMetodoOriginal] Result***.

Los protocolos o tipo de mensaje pueden ser de diferentes tipos como:

- Request (Petición de ejecución de un método).
- Inform (Respuesta a un Request).
- Event (Evento producido).
- Failure (Mensaje de error).
- Admin (Mensaje de administración).
- NoSpecified (Otros).

Los tipos de datos que se permiten en el coreógrafo son:

- String.
- Byte[].
- Int.
- Double.
- Float.
- Boolean.
- DateTime.

En la siguiente imagen se puede apreciar cómo está compuesto un mensaje de coreografía en .NET.

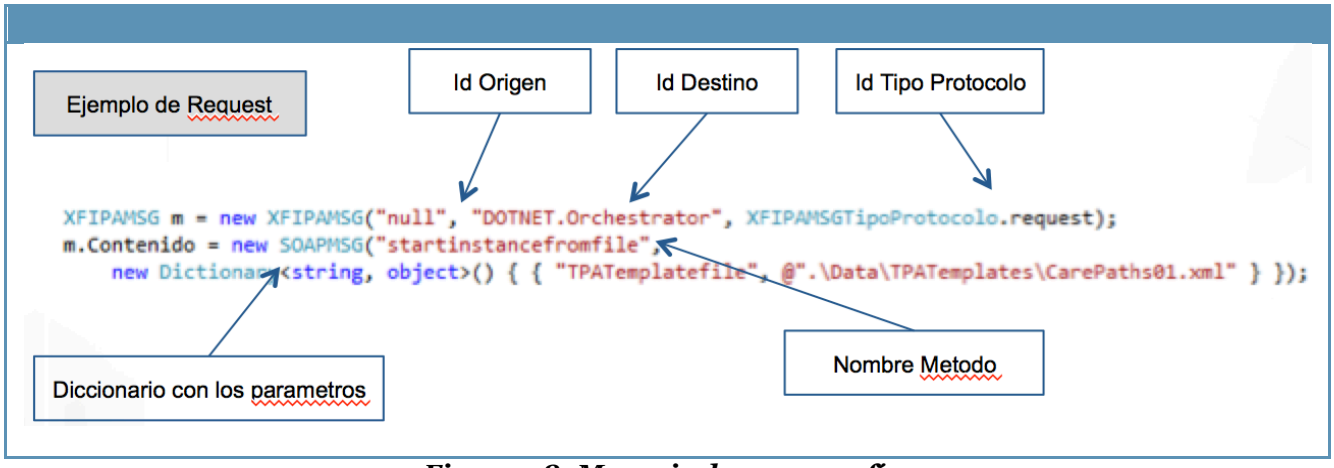


Figura 28: Mensaje de coreografía.

Para generar un servicio o generar un proceso se ha de cumplimentar la interfaz *IServicio* y cuando se quiere añadir un servicio se ha de hacer a través del coreógrafo con el método *AddService (IServicio)*. Si el coreógrafo está en marcha, este inicializará los servicios automáticamente.

Si tenemos un objeto que no cumpla con la interfaz *IServicio* podemos utilizar un objeto *WrapperService* para utilizarlo en el coreógrafo, los métodos públicos del objeto serán accedidos por *reflection* utilizando protocolos ***request***, la respuesta de realiza mediante protocolos ***inform*** y devuelve el resultado de la función.

10. Sistema Gestor de Base de Datos

En el presente proyecto hemos usado una pequeña pero potente base de datos para almacenar la información que vamos recogiendo a través del CVM Mini, como Windows IoT está pensado para trabajar en la nube nos viene bien que tengamos esta base de datos para la recopilación de información, los posibles futuros trabajos tratarían sobre como manipular esta información almacenada.

Para ello hemos instalado la versión de SQLite, dado que es la que más desarrollada esta para esta versión de Windows.

10.1. SQLite

SQLite es un sistema de gestión de base de datos relacional compatible con ***ACID***, el tamaño de esta base de datos es pequeña comparada con otro tipo de bases de datos, además la biblioteca está escrita en el lenguaje de programación C.

A diferencia de otros sistemas de gestión de base de datos Cliente-Servidor. El motor de SQLite es auto contenido esto quiere decir que SQLite se enlaza con el mismo programa pasando a ser parte integral del mismo, esto permite que las llamadas simples a subrutinas y funciones reduzcan la latencia de acceso a la base de datos porque estas llamadas son más eficientes que la llamada entre procesos.

El conjunto de la base de datos (Tablas, índices, etc.) está guardado en un solo fichero en la maquina host y esto se consigue bloqueando el fichero siempre que se inicie una transacción.

10.2. Proyecto.sqlite

La base de datos es de tipo SQLite y ahora pasaremos a mencionar las columnas, claves primarias que componen la tabla donde se guardará la información que se irá recibiendo.

Hemos de mencionar que a la hora de crear la tabla se quería diseñar para que varias de sus columnas fueran claves primarias pero se ha encontrado un error en la que no se permite la creación de más de una clave primaria y nos daba una excepción en la ejecución del programa, por lo que la solución a la que hemos llegado es de incluir más columnas y así evitar que se repitan valores, dado que estos serán únicos.

Los campos que se han usado para describir sus columnas son cuatro, estos son:

- ***Código del índice***: Clave primaria que se autoincrementa.
- ***Código del Registro***: Numero del registro leído.
- ***Código del valor***: Valor del registro leído.
- ***Fecha***: fecha en la que se introduce en la BBDD.

```
public class ClassStore
{
    [PrimaryKey, AutoIncrement]
    public int Cod_Indice { get; set; }
    public int Cod_Registro { get; set; }
    public int Cod_Valor    { get; set; }
    public DateTime Fecha   { get; set; }

    public override string ToString()
    {
        return string.Format("Cod_Indice={0}, Cod_Registro={1}, Cod_valor={2}, fecha={3}",
            Cod_Indice,
            Cod_Registro,
            Cod_Valor,
            Fecha);
    }
}
```

Figura 29: Tabla ClassStore.

En la imagen de arriba, el modelo será llamada ***ClassStore*** que posteriormente será traducido a su correspondiente tabla en SQLite y contendrá las cuatro columnas ya mencionadas, además se ha sobrecargado el método ***ToString()*** para mostrar la información almacenada.

10.3. ClassSQLite

Vamos a explicar los métodos que se encargarán tanto de crear, insertar, consultar los valores que tengamos en la base de datos.

Para poder realizar las acciones ***CRUD*** hemos creado la clase ***ClassSQLite*** donde contendrán todos los métodos necesarios para el correcto funcionamiento de la base de datos.

Para la creación de la base de datos, tenemos que combinar la ruta donde se quiere almacenar la base de datos en el dispositivo junto con el nombre del fichero y posteriormente, crear la conexión diciéndole el tipo de plataforma que se va a usar, en la siguiente imagen lo podremos apreciar mejor.

```
public ClassSQLite()
{
    try
    {
        path = Path.Combine(Windows.Storage.ApplicationData.Current.LocalFolder.Path, "Proyecto.sqlite");
        oCnn = new SQLiteConnection(new SQLite.Net.Platform.WinRT.SQLitePlatformWinRT(), path);
    }
    catch{ }
}
```

Figura 30: Creación de la base de datos.

Una vez que se ha creado la base de datos, pasaremos a crear el modelo de la información que será almacenaremos con SQLite.

Y para ello hacemos uso de la siguiente función:

```
public bool Start()
{
    try
    {
        oCnn.CreateTable<ClassStore>();
        return true;
    }
    catch
    {
        return false;
    }
}
```

Figura 31: Crear modelo ClassStore.

A la hora de introducir los datos en la base de datos hemos creado una función que recibe como parámetros tres valores estos son: *Código de registro, Valor del registro y la fecha* en la que se introduce.

El *Código índice* no hace falta introducirlo porque al definir el modelo ***ClassStore*** le hemos indicado que sea auto-incrementado, esto hará SQLite le asigne el valor correspondiente sin que se repite.

```
public bool insertClassStore(int iCodRegistro, int iValor, DateTime dtFecha)
{
    try
    {
        oCnn.Insert(new ClassStore()
        {
            Cod_Registro = iCodRegistro,
            Cod_Valor = iValor,
            Fecha = dtFecha
        });
        return true;
    }catch
    {
        return false;
    }
}
```

Figura 32: Insersión en el modelo ClassStore.

Esta no es la única forma en la que se puede introducir valores en la base de datos, el método ***Insert*** permite que se puede introducir por ejemplo una clase ***ClassStore*** ya creada sin necesidad de pasarle los valores por parámetros o incluso pasarle una lista de ***ClassStore***, la cual los introducirá sin problema.

También se han añadido otro métodos más por ejemplo en el de consultar todos los valores introducidos en la base de datos o el de mostrar el último valor que se ha introducido en la BBDD.

```
public IEnumerable<ClassStore> QueryAllStore()
{
    return from s in oCnn.Table<ClassStore>()
           orderby s.Cod_Indice
           select s;
}
```

Figura 33: Obtener Todos los valores introducidos.

Por último, indicaremos que hay un problema [20] con *SQLite*. Si en algún caso el dispositivo RPI2 se apagará o la conexión con la base de datos se suspendiera por algún motivo no se podría abrir la base de datos que hubiera posteriormente por lo que tendríamos que coger el fichero y llevarlo a otro lugar para su posterior tratamiento. En *SQLite* solo se permite crear una base de datos por ahora no se puede abrir una que ya tengamos hecha, pero esto solo ocurre en Windows IoT.

11. Herramientas de inteligencia de negocios

Las herramientas de inteligencia de negocios, son usadas para acceder a los datos de los negocios o corporaciones y que tengan su información almacenada en cualquier base de datos. Con esto se puede conseguir reportes, análisis, visualizaciones y alertas a los diferentes usuarios a los que este destinado este tipo de herramientas.

Actualmente en el mercado las herramientas de inteligencia de negocios se encuentran en dos segmentos, estas son: *Suites de Business Intelligence Empresarial* (*EBIS*) y por plataformas *BI*.

Las EBIS suelen ser usadas cuando hay muchos usuarios de diferentes niveles de habilidad técnica y cada uno de ellos tendrá unos reportes y vistas diferentes, estos serían menos analíticos. Por otra parte, las plataformas BI se usan cuando hay una necesidad de analizar aplicaciones complejas con una gran cantidad de cálculos.

La idea de incluir la inteligencia de negocios en este proyecto era la de usar los datos que se han ido almacenando en la BBDD e ir haciendo análisis de los mismos, con ellos conseguiríamos por ejemplo hacer un estudio del lugar de trabajo y ver cuál es el periodo de tiempo donde más se consume energía o la hora donde menos se consume, etc.

Al realizar este estudio se podrían ver alternativas o mejoras en el sistema para llegar a reducir ese consumo, para eso los informes generados contarían con la información necesaria indicando al representante de la pyme o empresa si esa toma de decisiones puede ser la adecuada para ellos o no.

12. Conclusiones

Es este apartado se expondrán a modo de resumen los objetivos que se han visto alcanzados durante el desarrollo del proyecto, así como los problemas más importantes que surgieron a lo largo del trabajo.

12.1. Contratiempos y problemas sufridos

El principal problema fue que, a medida que avanzaba el trabajo los problemas de conectividad entre los distintos tipos de hardware no funcionaban como se esperaban en Windows IoT, por lo que se tuvo que investigar en detalle los fallos para llegar a la solución deseada, por lo que se dedicó mucho tiempo al aprendizaje de los diferentes tipos de hardware.

En relación con las pruebas, se precisaba el uso del protocolo Modbus para la comunicación con el CVM Mini, por lo que aquí también se dedicó tiempo al desarrollo de esta parte de código para crear una versión reducida de Modbus. Por otra parte, en los laboratorios no se disponía de todo el material necesario para el desarrollo del proyecto, por lo que se hizo uso de un ordenador personal para el desarrollo del mismo y así tener control absoluto sobre la máquina.

En el diseño inicial no estaba contemplado usar el sistema de coreografía, por lo que se tuvo que ir al instituto ITACA, dedicar tiempo al aprendizaje y entendimiento del sistema de coreografía, y gracias a la ayuda de los responsables de este proyecto, y a las tutorías personalizadas se pudo integrar este sistema en el proyecto.

12.2. Consideraciones finales

Cabe destacar que, la realización de este proyecto no fue fácil y que sin la ayuda tanto de mi tutor como la de otro profesorado, no se hubiera podido llegar a finalizar con éxito este proyecto.

Este proyecto me ha dado una base que complementa mi formación universitaria, tanto a nivel profesional como a nivel personal, dado que ha sido un gran reto poder enfrentarme a un proyecto de esta envergadura y que implique diferentes conceptos.

13. Futuros trabajos

Existen infinidad de posibles futuros trabajos gracias a las prestaciones tanto de la RPI2 como a la de Windows IoT, pero como siempre, las versiones más modernas ofrecen mejores prestaciones y servicios como es en el caso de la Raspberry PI3, que cuenta con un chip Wifi integrado y no haría falta introducir un hardware de terceras personas, dado que a veces o no se puede o cuesta mucho de integrar en el sistema.

En la toma de datos se podría optimizar la velocidad con la que se lee los registros del CVM Mini. Con esto, se quiere decir que se podría realizar una versión mejorada de Modbus, ya que para Windows IoT no hay una versión oficial y la que hemos presentado en nuestro proyecto es una versión muy básica realizada por nosotros.

Otro posible de trabajo, sería la implementación de un sistema experto o la mejora de la inteligencia de negocio permitiendo el análisis de los datos almacenados y que aprenda de ellos, por ejemplo, a la hora de actualizar o renovar componentes de las pymes ya sean bombillas de bajo consumo o cualquier aparato electrónico que consuma electricidad. Con esto se puede llegar a conseguir que se realicen informes sobre la toma de decisiones indicando si fuera necesario, si esos cambios valdrían la pena o no al cabo del tiempo.

Este sistema se podría encargar de ofrecer un informe al responsable de la pyme indicando el tiempo que le llevaría a la pyme amortizar el dinero invertido en esas mejoras, indicando el precio del consumo de esos aparatos respecto el valor invertido y mostrando si es viable introducir esa mejora en el edificio o no.

El sistema de coreografía se podría mejorar para que este pueda funcionar con una infinidad de sistemas Raspberry PI, para eso habría que generar un orquestador que se encargará de la toma de decisiones, además de recibir y enviar las órdenes o datos necesarios para su funcionamiento.

14. Bibliografía

[1] *Raspberry PI*, 2012, [consulta: 22 Febrero 2016, 18:00]. Disponible en: https://www.Raspberrypi.org/.

[2]*Analizadores de redes Serie CVM-Mini*, 2015, [consulta: 10 Marzo 2016, 16:00]. Disponible en: http://docs.circutor.com/docs/M98174001-01.pdf.

[3]*Chip Rs485 Module,* 2016, [consulta: 15 Abril 2016, 11:30]. Disponible en: http://arduino-info.wikispaces.com/RS485-Modules.

[4]*Sistemas operativos para Raspberry PI*, 2016 [10 Agosto 2016, 10:30]. Disponible en: https://Raspberryparatorpes.net/Raspberry-pi-sistemas-operativos/.

[5]*Windows IoT,* 2015, [consulta: 29 Enero 2016, 10:00]. Disponible en: https://developer.microsoft.com/es-es/windows/iot.

[6]*C#*, 2016, [consulta: 6 agosto 2016, 9:30]. Disponible en: https://msdn.microsoft.com/es-es/library/kx37x362.aspx.

[7]*XAML*, 2016, [consulta: 30 Enero 2016, 17:30]. Disponible en: https://msdn.microsoft.com/es-es/library/cc295302.aspx.

[8]*Tutorial sobre WPF*, 2011, [consulta: 6 Febrero 2016, 19:00]. Disponible en: http://www.wpftutorial.net/GettingStarted.html.

[9]*Guía de SQLite*, 2015, [consulta: 28 Abril 2016, 11:45]. Disponible en: http://blog.chrisbriggsy.com/Using-SQLITE-in-Windows-10-IoT-Core-Insider-Preview/.

[10]*Introducción UAP, 2015*, [consulta: 10 Abril 2016, 15:30].Disponible en: https://msdn.microsoft.com/en-us/magazine/dn973012.aspx.

[11]*Historia sobre la Raspberry PI*, 2013, [consulta: 12 Julio 2016, 16:30]. Disponible en: http://histinf.blogs.upv.es/2013/12/18/Raspberry-pi/.

[12]*Todo sobre Rs-485*, 2016, [consulta: 25 Junio 2016, 10:00]. Disponible en: https://en.wikipedia.org/wiki/RS-485.

[13]*Producto Fluke 435*, 2016, [consulta: 20 Agosto 2016, 16:00]. Disponible en: http://www.fluke.com/fluke/eses/medidores-de-calidad-de-la-energia-electrica/registradores-de-calidad-electrica/fluke-435-series-ii.htm?pid=73939.

[14]*Producto Vega78*, 2016, [consulta: 21 Agosto 2016, 10:30]. Disponible en: http://www.ht-instruments.com/en/products/power-quality-analyzers/touch-screen/vega78/.

[15]*Manual de usuario PM3250*, 2016, [consulta: 23 Agosto 2016, 16:00]. Disponible en: http://azzo.com.au/wp-content/uploads/2015/05/PM3000-User-Manual.pdf.

[16]*Manual de usuario Panasonic KW9M*, 2016, [consulta: 25 Agosto 2016, 10:00]. Disponible en: https://www.panasonic-electric-works.com/pew/es/downloads/ds_63070_0008_es_kw9m.pdf.

[17]*¿Que es Modbus?*, 2016, [consulta: 5 Agosto, 2016, 11:00]. Disponible en: http://tecdigitaldelbajio.com/blog/27-modbus-parte-iii-que-es-el-modbus.html.

[18]*Información sobre el sistema de coreografía*, 2016, [consulta: 25 Febrero, 2016, 16:00]. Disponible en: http://www.sabien.upv.es/lach/topics-tagged/choreographer/.

[19]*Solución al error de SQLite*, 2015, [Consulta 28 Agosto 2016, 19:00]. Disponible en: http://igrali.com/2015/04/10/composite-primary-keys-in-sqlite-net/.

[20]*Guía de instalación Windows IoT*, 2016, [consulta: 20 Abril 2016, 20:00]. Disponible en: https://developer.microsoft.com/es-es/windows/iot/getstarted.

[21]*Mi primera aplicación en Windows IoT*, 2015, [consulta: 24 Febrero 2016, 16:00]. Disponible en: http://aminespinoza.com/mi-primera-aplicacion-en-windows-10-iot/.

15. Anexo I. Instalación Windows IoT

Para proceder con la instalación del Windows 10 IoT nos hará falta los siguientes componentes:

- RPI2.
- Una SD de al menos 8 GB de clase 10 o mejor.
- Un PC con Windows 10.

Al tener los componentes preparados procederemos con la instalación.

Para empezar tendremos que ir a la siguiente página web: Windows 10 IoT – Getting Started [21].

En la página nos indicará las opciones que se pueden seleccionar para descargar e instalar la versión de Windows que necesitemos:

Paso 1: Seleccionaremos el dispositivo al que queremos instalar la versión de Windows 10 IoT, en nuestro caso la RPI2.

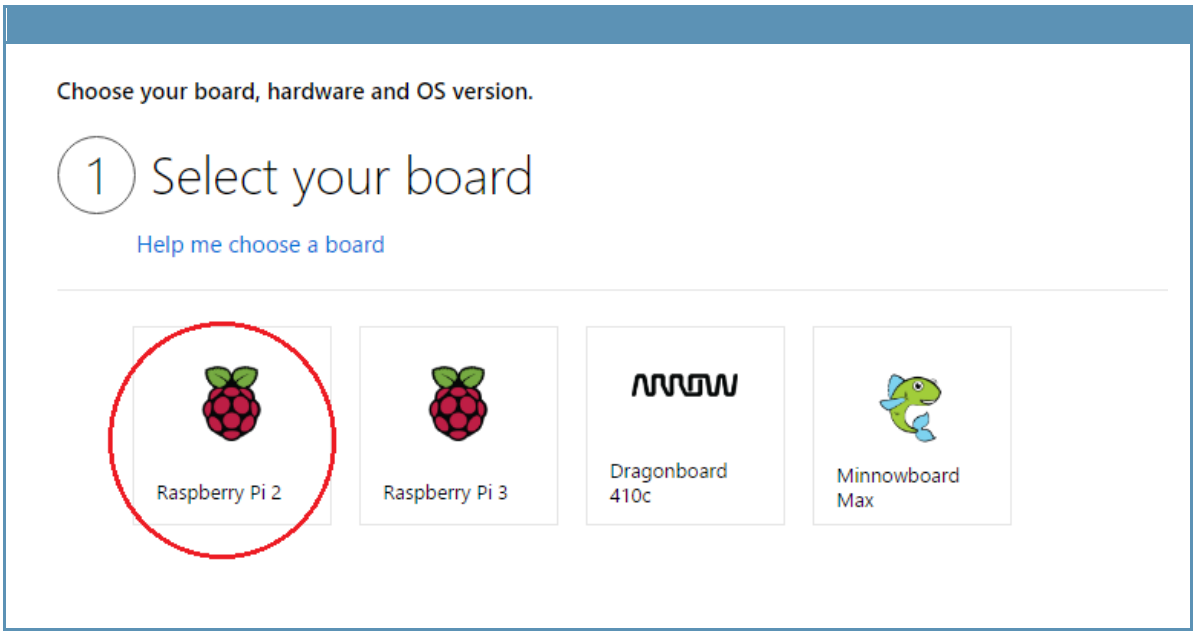


Figura 34: Selección de dispositivo.

Paso 2: En este paso debemos seleccionar *Flashear* en nuestra SD, aquí se instalará la versión de Windows.

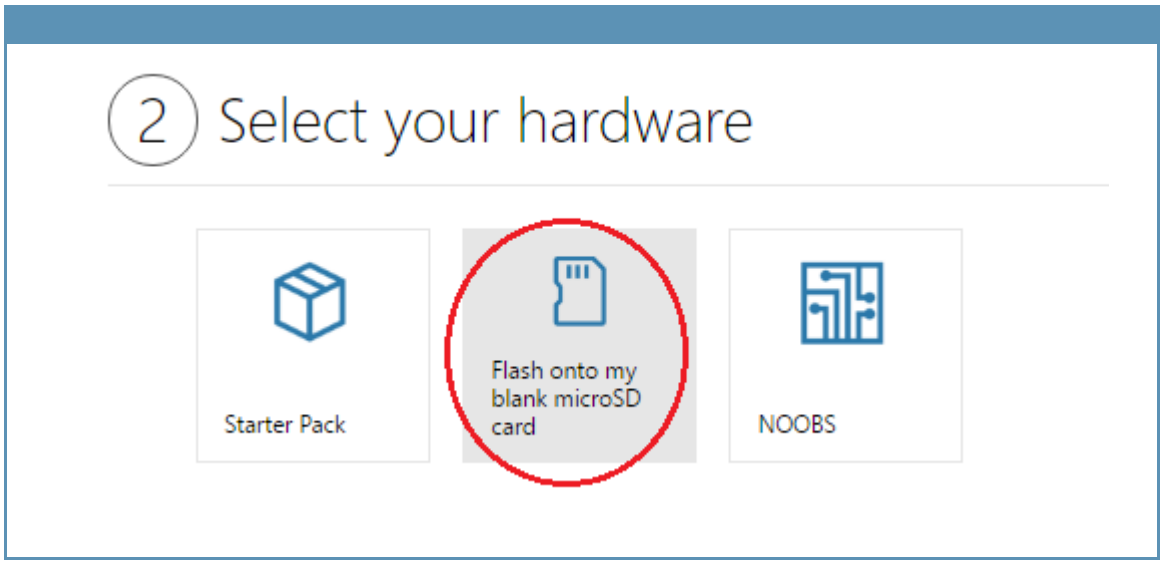


Figura 35: Método de instalación.

Paso 3: Aquí seleccionaremos la versión de Windows que se desea instalar y por último pulsamos el botón *Next*.

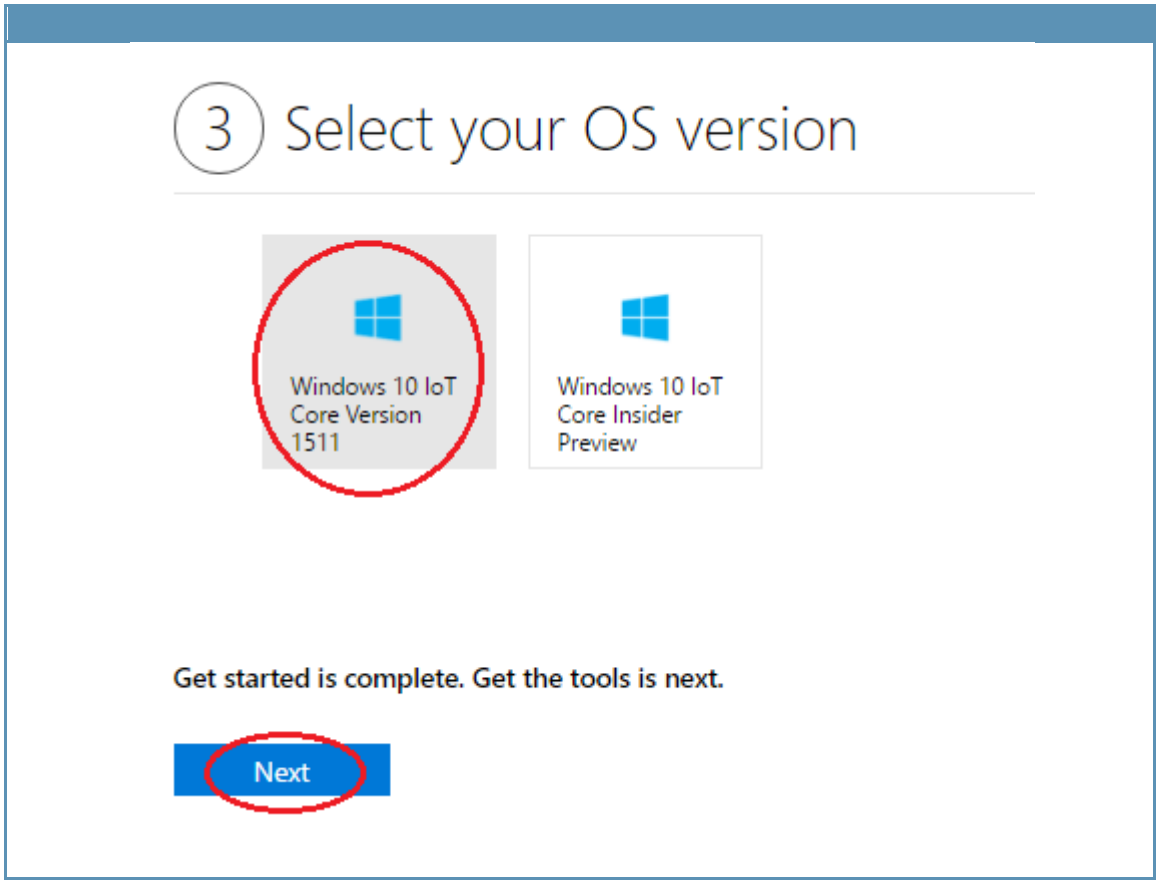


Figura 36: Versión de Windows IoT.

Cuando se pulse el botón ***Next*** nos llevará a otra página donde nos indicará, que para poder instalar la versión de Windows IoT nos hace falta tener una versión Windows 10 (Versión 10.0.10240 o mejor), esto es debido por compatibilidad a la hora de la instalación, evitando el ***error 87*** de Windows.

En nuestro siguiente paso deberemos de descargar el DashBoard para completar la instalación.

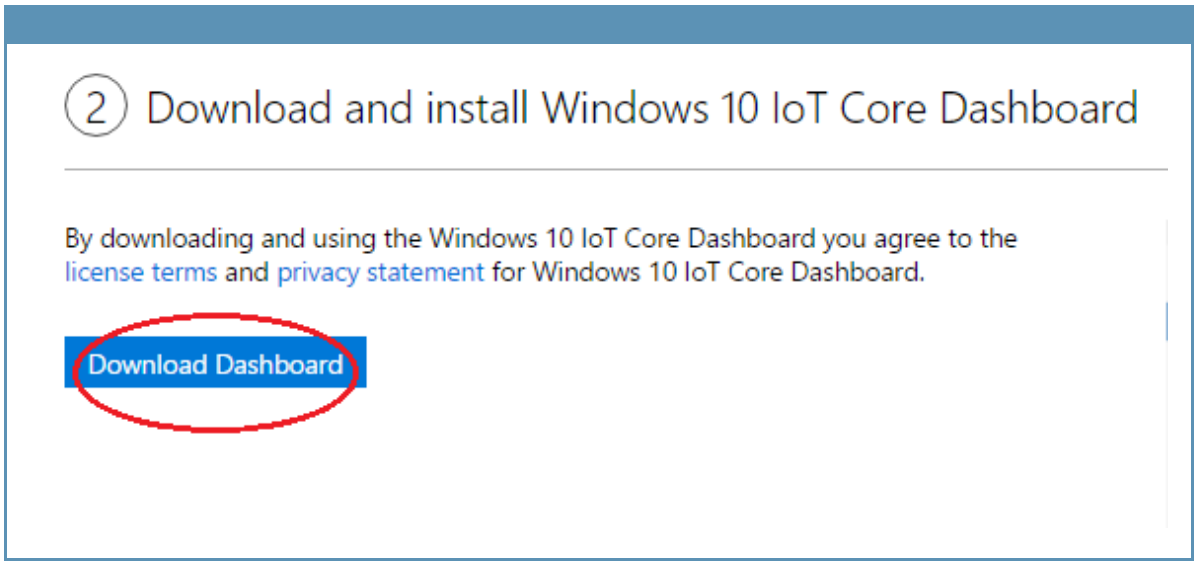


Figura 37: Descarga del DashBoard.

Cuando se haya terminado de descargar la aplicación, pasaremos a instalarla, aceptamos los términos de licencia y privacidad.

Pasamos a la siguiente página pulsando el botón ***Next,*** donde continuaremos con la instalación de Windows IoT.

Los pasos que deberemos de seguir son los siguientes:

1. Abrimos el Windows IoT Core DashBoard.
2. Pulsamos en *Configurar un nuevo dispositivo*.
3. Seleccionamos el dispositivo al que queremos instalar la versión de Windows, en nuestro caso RPI2 & 3.
4. Seleccionamos la versión de Windows IoT.
5. Escogemos la tarjeta SD donde se quiere instalar.

En la nueva versión del DashBoard, han añadido las siguientes opciones, como cambiarle el nombre a los dispositivos, añadirle una nueva contraseña y si nuestro dispositivo fuera una Raspberry PI 3 con Wifi, añadiríamos la conexión Wifi.

Al haber seleccionado los pasos previos pulsamos el botón de ***Download and Install.***

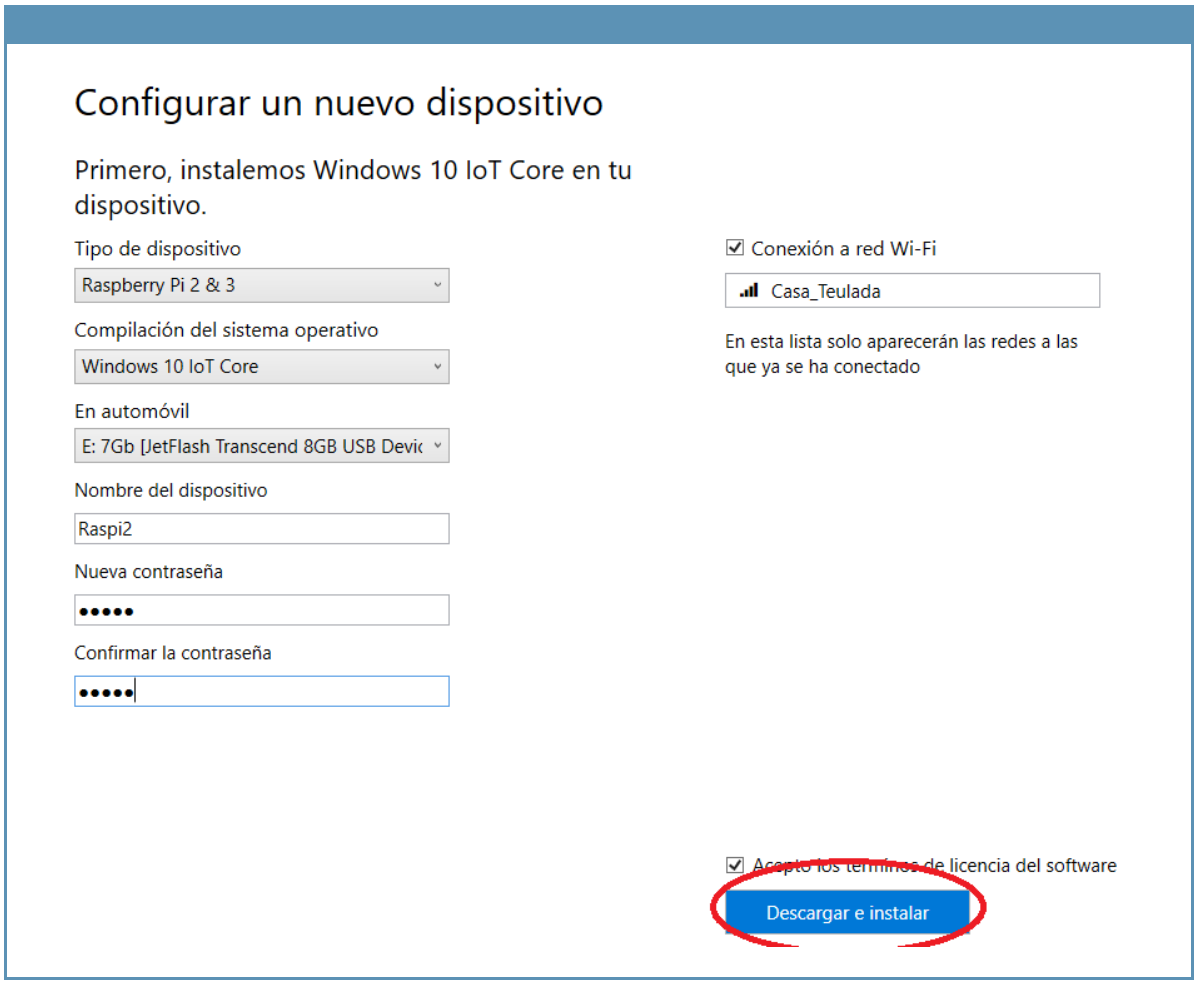


Figura 38: Instalación Windows IoT en RPI2.

16. Anexo II. Despliega tu primera App para Windows IoT (¡Hola Mundo!)

Una vez que se haya configurado el dispositivo solo basta con crear una nueva aplicación de Windows 10 con el Visual Studio 2015 [22].

Para realizar la creación de esta *App*, seguiremos los siguientes pasos:

- Abrimos el Visual Studio 2015, y creamos un nuevo proyecto de tipo ***Windows Universal***.
- Seleccionamos ***Blank App*** y le damos un nombre (**¡Hola Mundo!**).

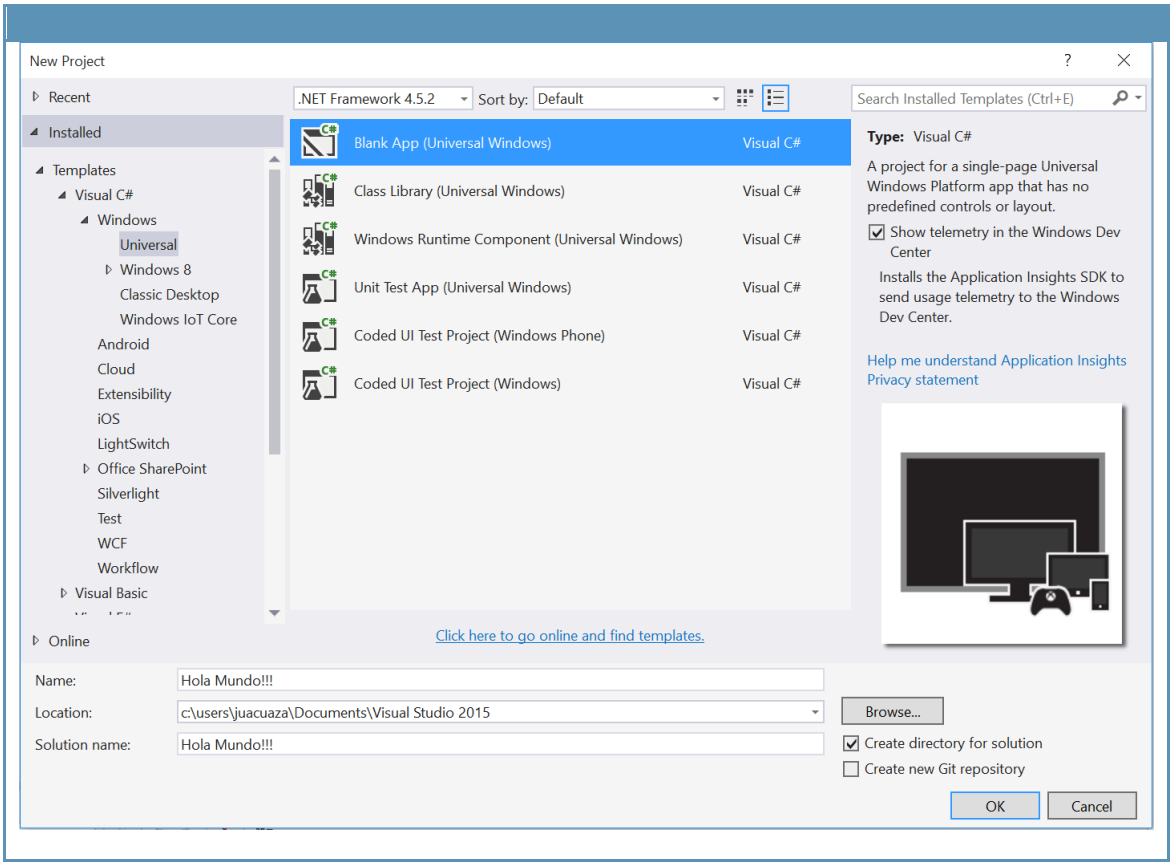


Figura 39: Creación App Windows IoT.

Pulsamos el botón OK y nos creará el proyecto. En la siguiente ventana nos enseñará el proyecto creado donde veremos lo siguiente.

- En el explorador de soluciones abrimos el archivo ***MainPage.xaml*** y añadimos estos tres componentes:
  - Un Contenedor (StackPanel).
  - Un Botón.
  - Y una caja de Texto.

Quedando el código resultante así:

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <StackPanel VerticalAlignment="Center"
                HorizontalAlignment="Center">
        <Button x:Name="btnControl"
                Content="Click me"
                Height="36"
                Width="250"
                Margin="10"
                Click="btnControl_Click"/>
        <TextBox
            x:Name="txtHola"
            Height="23"
            Margin="10"
            TextWrapping="Wrap"/>
    </StackPanel>
</Grid>
```

Figura 40: Código interfaz.

- Ahora iremos el fichero ***MainPage.xaml.cs*** e introduciremos el código del evento del botón.

```
private void btnControl_Click(object sender, RoutedEventArgs e)
{
    btnControl.Content = txtHola.Text;
    txtHola.Text = string.Empty;
}
```

Figura 41: Código del evento.

Con esto ya tenemos nuestra aplicación lista para ser desplegada, pero antes tendremos que hacer algunos pasos extras:

- Primero tendremos que ir a referencias y añadir una nueva extensión para poder instalar la *App* en la Raspberry PI.

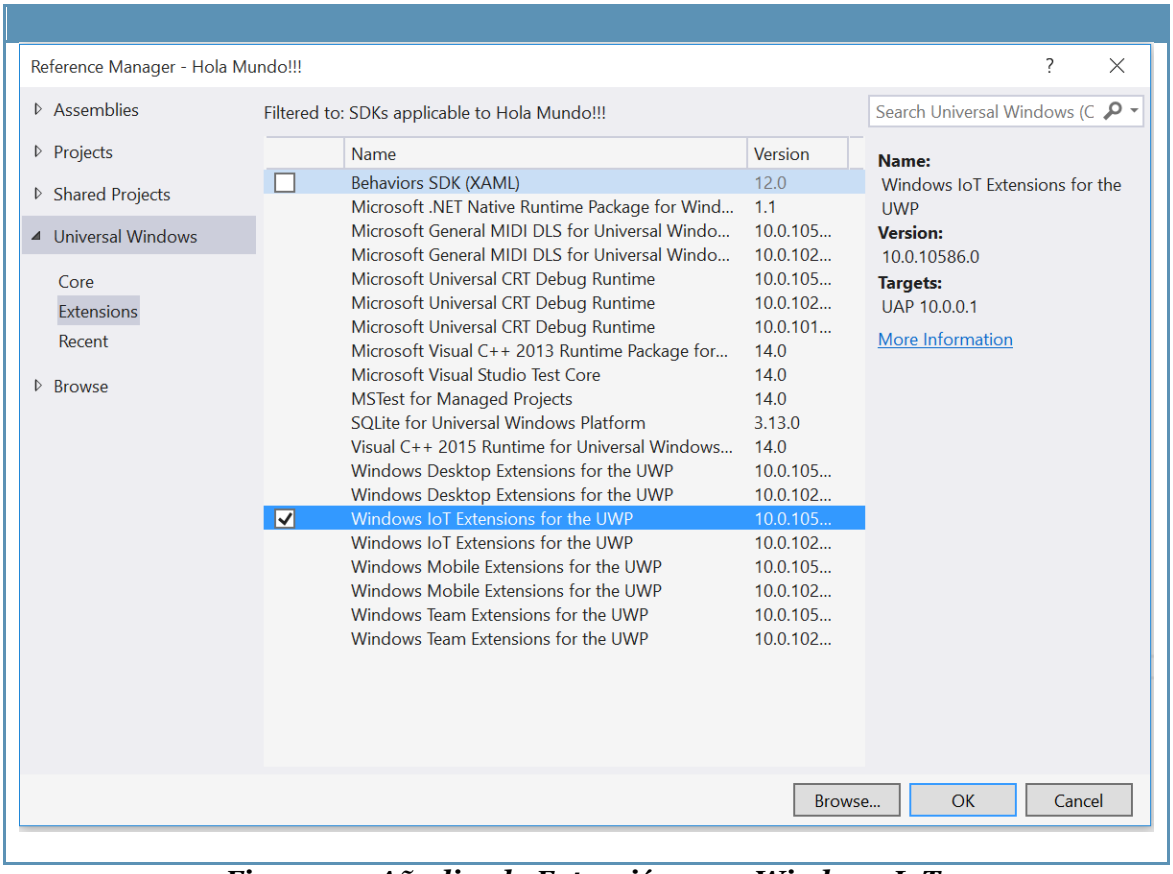


Figura 42: Añadiendo Extensión para Windows IoT.

- Segundo tendremos que seleccionar la arquitectura ***ARM*** y en el desplegable seleccionamos ***Remote Machine,*** Aquí es donde le indicaremos al Visual Studio donde tiene que efectuar el despliegue de la aplicación.
- Le tendremos que indicar la dirección ***IP*** del dispositivo y muy importante el tipo de autenticación que por defecto o por normal general ha de ser ***Universal (Unencrypte Protocol).***

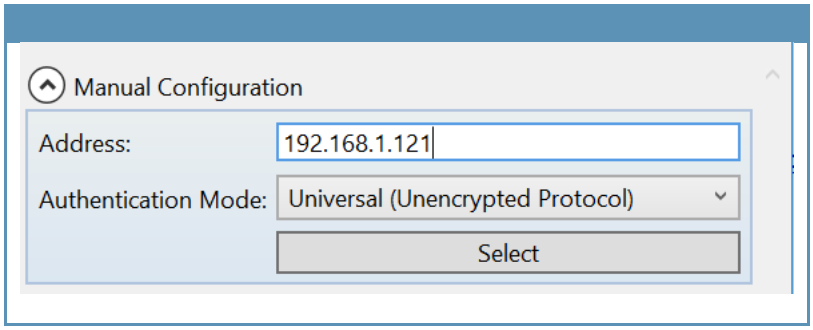


Figura 43: Selección de dispositivo.

Ahora tendremos que esperar a que el proceso de depuración termine y que empiece el despliegue de la aplicación. Dependiendo del tamaño de la aplicación esto nos puede llevar algo de tiempo.

17. Anexo III. Interfaz de la Raspberry Pi 2

En este anexo veremos la interfaz de la RPI2 y como está distribuida, explicando cada una de sus partes y qué función realiza cada una dentro de la aplicación.

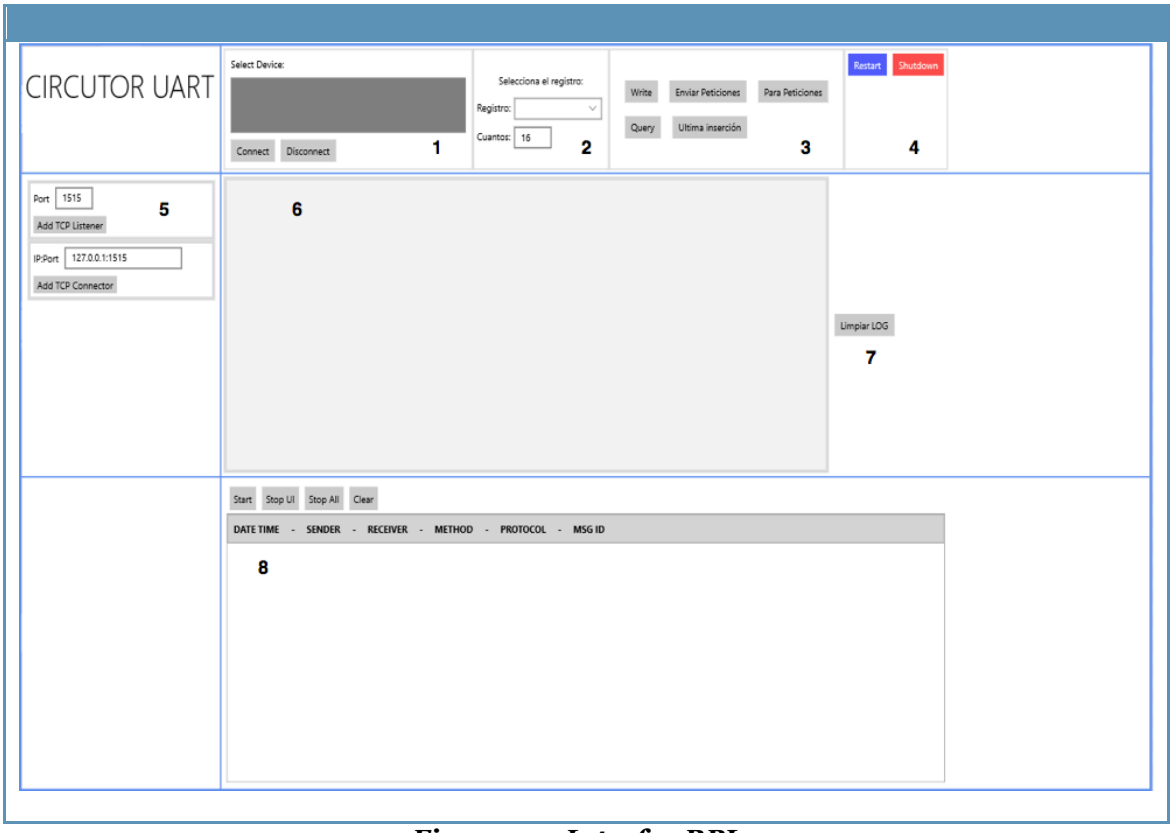


Figura 44: Interfaz RPI2.

Como podemos apreciar en la imagen, la interfaz la hemos divido en 8 secciones de las cuales, cada una de ellas hace una función diferente y que vamos a proceder a explicarlas:

- ***Sección 1***: en esta sección veremos los dispositivos a los cuales nos podremos conectar. Cuando todo el sistema está en funcionamiento se podrá ver el nombre del chip de Arduino en esta ventana. Pulsando en el botón ***Connect***, nos conectaremos al dispositivo que este seleccionado y pulsando en el botón ***Disconnect*** se desconectará del que se haya seleccionado previamente.

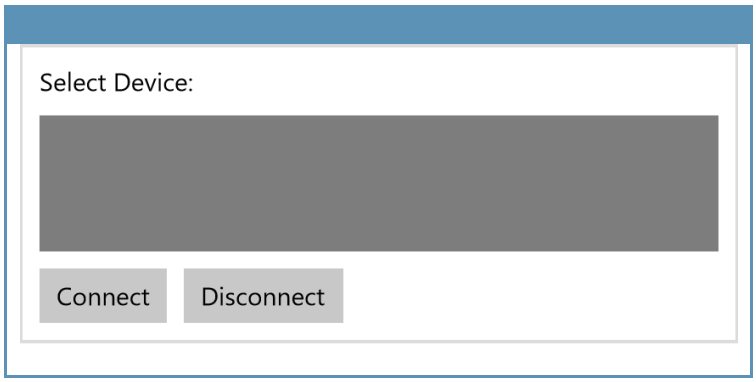


Figura 45: Sección 1 – Conectar con dispositivo.

- ***Sección 2***: En esta parte podremos seleccionar algunos registros al azar que se han añadido al ***ComboBox*** y podremos indicar también la cantidad de registros que se quieren leer rellenando la casilla ***Cuantos***.

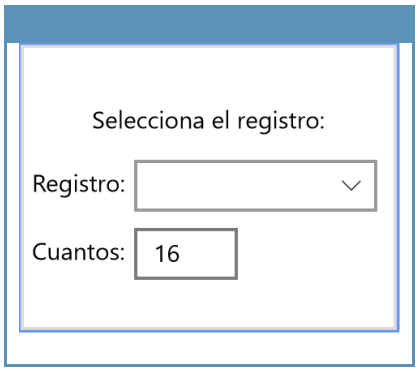


Figura 46: Sección 2 – Elección de Registro.

- ***Sección 3***: En esta sección podemos ver que está compuesta por varios botones, y cada uno de ellos realiza una acción diferente y son:
  - ***Write***: Envía la petición de lectura de registros con los datos indicados en la sección 2 hacia el ***CVM Mini.***
  - ***Enviar Peticiones***: Este botón hace que las peticiones se hagan de forma automática, es decir, empezará a pedir la lectura de los registros desde el 0 hasta le ultimo de todos, cuando se haya llegado al final volverá a empezar de nuevo, en caso de que hubiera un error de petición el programa guardará la última petición enviada y al cabo de unos segundos lo volverá a intentar, hasta que le llegue algún valor legible.
  - ***Para Peticiones***: Si se están enviando las peticiones de forma automática, este botón pararía todas las peticiones enviadas.
  - ***Query***: Al pulsar este botón mostraría todo el contenido de la base de datos y lo enseñaría en el log de la aplicación.
  - ***Última inserción***: con esto mostraríamos el último valor insertado en la base de datos.

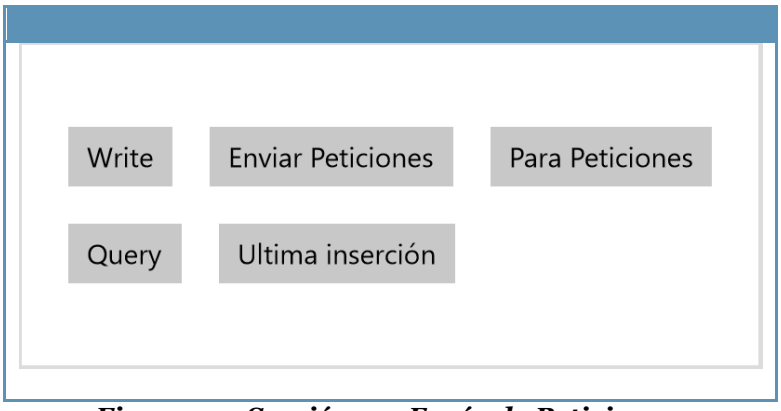


Figura 47: Sección 3 – Envío de Peticiones.

- ***Sección 4:*** Con los dos botones que forman esta sección conseguiremos que la RPI2 pueda Reiniciarse o Apagarse.

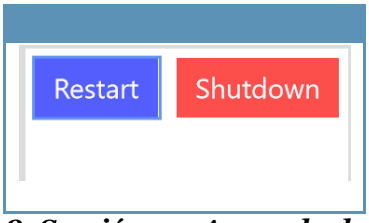


Figura 48: Sección 4 – Apagado de la RPI2.

- ***Sección 5***: En este apartado debemos de indicar dos cosas:

  1. Cuando se quiere recibir las peticiones necesarias para el funcionamiento del coreógrafo, se ha de crear un ***Listener*** para su funcionamiento, esto no es más que escuchar por el puerto que se le ha asignado al *Listener*.

  2. Cuando indicamos la dirección *IP:Puerto* le estamos indicando al coreógrafo a qué lugar se tiene que conectar para enviar las peticiones necesarias para su funcionamiento. El nombre que recibe esta función es el ***Connector***.

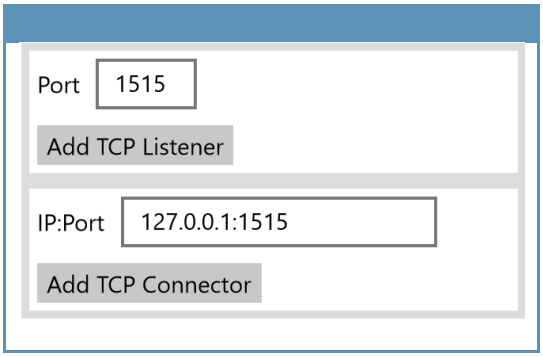


Figura 49: Sección 5 – Asignación de Conectores.

- ***Sección 6 y 7***: estas dos secciones están compuestas por el Log de la aplicación y el botón que borra el contenido del log, por normal general todos los eventos producidos por la aplicación se muestran en el Log, salvo los mensajes de coreografía, dado que estos tienen su propio Log.

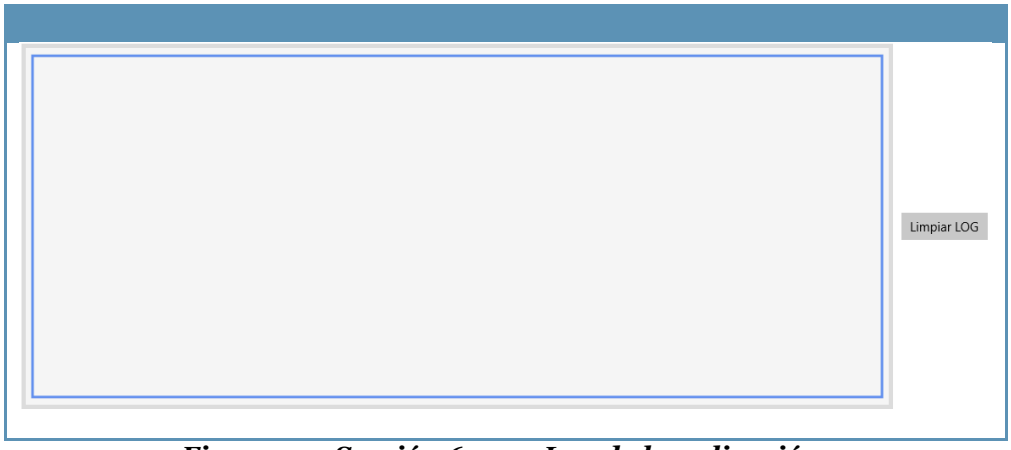


Figura 50: Sección 6 y 7 – Log de la aplicación.

- ***Sección 8***: En la última sección tenemos el Log del coreógrafo, en esta subventana nos llegará la información de cualquier evento que sea producido por algún mensaje de coreografía, aquí veremos el emisor y el receptor del mensaje, además del contenido del mensaje.

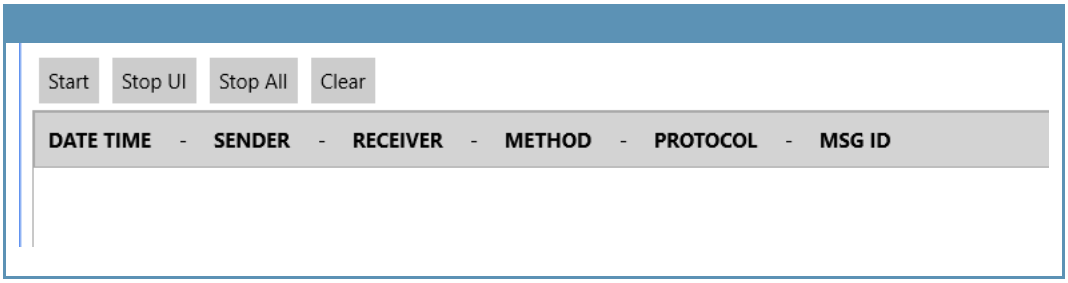


Figura 51: Sección 8 – Log de Coreográfia.