Escuela Politécnica Superior de Jaén

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN
ESCUELA POLITÉCNICA SUPERIOR DE JAÉN

Trabajo Fin de Grado

DATALOGGER CON ENLACE BLUETOOTH A SISTEMA ANDROID

Alumno: Juan Miguel Bejarano Bueno

Tutor: Prof. D. Luis Miguel Nieto Nieto
Dpto: Ingeniería Electrónica y Automática
Area: Tecnología Electrónica

Septiembre, 2015

Universidad de Jaén

Escuela Politécnica Superior de Jaén
Departamento de Electrónica y Automática

Don LUIS MIGUEL NIETO NIETO, tutor del Trabajo Fin de Grado titulado: DATALOGGER CON ENLACE BLUETOOTH A ANDROID, que presenta JUAN MIGUEL BEJARANO BUENO, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, SEPTIEMBRE de 2015

El alumno: El tutor:

JUAN MIGUEL BEJARANO BUENO LUIS MIGUEL NIETO NIETO

RESUMEN

Este Trabajo de Fin de Grado consiste en el desarrollo de dos sistemas. En primer lugar, se dispondrá de un sistema de registro de datos por eventos o datalogger. El sistema está basado en un microcontrolador Microchip y dispondrá de una interfaz de comunicaciones Bluetooth. Este dispositivo será capaz de realizar mediciones con una duración mínima configurable. En segundo lugar y de manera simultánea, se ha diseñado una aplicación Android capaz de gestionar la comunicación con el dispositivo de una manera gráfica. El sistema se ha diseñado siempre teniendo en cuenta el ahorro económico, dispondrá de una alimentación autónoma y el mantenimiento será prácticamente nulo. Se pretende así obtener un datalogger funcional, económico e intuitivo para el manejo del mismo por cualquier tipo de usuario.

ABSTRACT

This Bachelor Thesis consists on the development of two systems. First of all, the development of a device which offer data events registration, called datalogger. This system is based on a Microchip microcontroller with a Bluetooth communication interface. This device will be able to measure events with a minimal configurable duration. Secondly and simultaneously, an Android application has been designed. This application makes possible a management of communication in a graphic way. The system has been dessigned taking into account an economical saving, it will have an autonomus supply with almost no maintenance. In this manner, it is intended to provide a functional, economical and intuitive datalogger which can be used by all types of users.

ÍNDICE

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

CAPÍTULO 1. *Introducción.*

1. Introducción

El presente proyecto ha sido realizado como Trabajo de Fin de Grado de la titulación de Grado en Ingeniería Electrónica Industrial. Pertenece al grupo de Proyectos de Fin de Grado propuestos al departamento de Electrónica y Automática y ha sido supervisado por el tutor del mismo D. Luis Miguel Nieto Nieto.

Este proyecto aborda el desarrollo de un sistema de registro de datos por eventos o datalogger. Cuenta con un sistema basado en un microcontrolador Microchip, encargado de comunicarse con periféricos para obtener datos y transmitirlos mediante una interfaz de comunicación Bluetooth. Por lo tanto, el sistema requerirá una aplicación ejecutandose en un sistema Android que recibe los datos registrados vía Bluetooth y que permite enviar al datalogger información de configuración.

1.1. Descripción de los capítulos

El presente TFG se divide en siete capítulos que serán descritos a continuación:

- CAPÍTULO 1. *Introducción:* constituye el capítulo actual, se introduce de manera general el proyecto y se da una visión sobre los temas tratados en cada capítulo.
- CAPÍTULO 2. *Objetivos del proyecto:* conforma los objetivos que se persiguen con este proyecto, así como una descripción más detallada de las características y funcionalidades del mismo.
- CAPÍTULO 3. *Antecedentes:* se aborda el tema desde un punto de vista más general donde se incluye nuestro proyecto, los sistemas embebidos. Se describirá en qué consisten este tipo de sistemas, se dará una visión muy general sobre la estructura de los mismos y se expondrán algunas de las posibles soluciones que podrían barajarse en la actualidad.
- CAPÍTULO 4. *Soluciones adoptadas:* se describen en éste capitulo las soluciones que se han adoptado para el desarrollo de este trabajo justificando la elección del microcontrolador, así como la de los componentes implicados. Se detalla también el software empleado y la función que ha desempeñado en el proyecto.

- CAPÍTULO 5. *Descripción del proyecto:*Este capítulo conforma el pilar de este Trabajo de Fin de Grado, el mismo es detallado partiendo de una forma general hasta concretar de manera específica con todo lo relacionado con el desarrollo del sistema. Consta de dos partes bien diferenciadas, la primera es la relativa a la configuración del hardware y la segunda aborda los temas de la programación del software. Esta segunda parte engloba a su vez dos partes, el desarrollo del programa para nuestro microcontrolador y el desarrollo de la aplicación Android para el control y configuración del dispositivo.
- CAPÍTULO 6. *Conclusiones y Presupuesto:* para concluir este proyecto, se presentará un presupuesto, se comparará las características de nuestro datalogger con algunos productos del mercado actual y se expondrán las conclusiones.

Se podrá encontrar a cotinuación otros documentos de interés como son:

- Bibliografía: en este documento se exponen fuentes bibliográficas en las que se basa este trabajo.
- Anexos:

  ANEXO I. Planos.

  ANEXO II. Manual de usuario.

  ANEXO III. Dataloggers comerciales.

- Además de estos documentos, se dispone de un CD donde se incluyen: proyecto de simulación en Proteus, código de programa del microcontrolador PIC, el proyecto de aplicación Android con un manual de usuario para el uso de la misma, una copia de la memoria en formato digital y la hoja de características de los componentes.

CAPÍTULO 2. *Objetivos del proyecto.*

2. Objetivos del proyecto

Los objetivos del proyecto que se persiguen son los siguientes:

- El desarrollo de un sistema hardware (en nuestro caso será nuestro datalogger), el cual debe disponer de las siguientes funcionalidades:

  - Registro de eventos de corte de alimentación en la red (fecha, hora, minuto del corte y duración en minutos), que tengan una duración mínima configurable.

    Para probar la funcionalidad de nuestro datalogger, nuestro sistema dispondrá de registros de eventos de temperatura con un sensor de Microchip encargado de proporcionar los datos correspondientes junto con una Real Time Clock que nos proporcione los datos de fecha, hora y minuto.

  - Alimentación autónoma.

    Nuestro sistema dispondrá de una alimentación autónoma con baterías que funcionen a 5 V para alimentar todo el sistema. Además, se incluye una "Backup battery" o una pila de botón de 3 V encargada de mantener la configuración de nuestros dispositivos en caso de que nuestra batería de 5 V se agote. El funcionamiento de la pila es similar al empleado en una placa base de un PC.

  - Interfaz Bluetooth para comunicaciones.

    La comunicación, que anteriormente se realizaba mediante cableado con RS232 y que tan importante es para este tipo de proyectos, se ha llevado a cabo sin cables, gracias a la debida incorporación de un módulo Bluetooth configurado para la recepción y envío de datos.

- El desarrollo de una aplicación para un sistema basado en Android con las siguientes funcionalidades:

  - Envío de parámetros de configuración al datalogger (fecha actual, duración mínima de un evento para su registro, etc.).

    Nuestra aplicación dispone de un menú para el envío de datos (fecha, hora, minutos y segundos), así como el envío de la configuración de tiempo de registro de eventos.

  - Recepción de datos de estado y datos registrados. Creación de un archivo de datos.

    La recepción y envío de datos se realizará a través del Bluetooth de nuestro dispositivo Android. Sin embargo, se dará al usuario la opción del registro de datos de dos formas; la primera mediante la escritura en un archivo en una tarjeta SD; y la segunda, mediante el modo de lectura en tiempo real, que generará automáticamente un archivo en el almacenamiento interno de nuestro dispositivo Android.

CAPÍTULO 3. *Antecedentes.*

3. Antecedentes

La fuerte presión de los mercados emergentes así como el desarrollo y el alto nivel de competitividad en el mundo actual dan lugar a la aparición de los llamados Sistemas Embebidos. Su gran aplicabilidad a cualquier sector industrial, hace que su desarrollo sea un objetivo para la diferenciación de muchas empresas.

3.1. Qué es un Sistema Embebido

Un sistema embebido es un sistema computacional cuyo hardware y software están diseñados específicamente para realizar las labores de operación, control e instrumentación definida de una manera eficiente. (Heath, 2003)

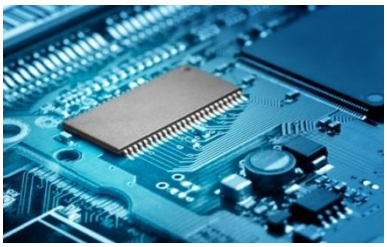


Imagen 1. Ejemplo de sistema embebido.

Los Sistemas Embebidos de los que disponemos en la actualidad se encuentran de igual manera en aspectos tan cotidianos como vehículos de transporte, como en procesos críticos de producción industrial.

Gozan de una alta capacidad de cálculo para realizar tareas complejas como el cálculo de la FFT[1] con aplicación a filtros digitales, análisis de señales, o cálculos matemáticos complejos para tomar decisiones en tiempo real. Estos sistemas aportan valor añadido a los productos y, cada vez más, son los responsables de las mejoras introducidas en términos de innovación y competitividad.

Estos sistemas electrónicos, como su propio nombre indica, se encuentran integrados dentro del dispositivo sobre el que actúan, con lo cual deben ser eficientes en cuanto a la energía, al tamaño de código, al peso y al costo. (UNED.Departamento de Ingeniería Eléctrica). Otras características básicas de los sistemas embebidos son las siguientes:

- La confiabilidad, es decir que el sistema trabaje correctamente.
- La seguridad informática: consiste en disponer de una comunicación confidencial y autentificada.
- Están dedicados a ciertas aplicaciones concretas.
- Cuenta con interfaces de usuario.

[1] **FFT** es la abreviatura usual (del inglés Fast Fourier Transform) de un eficiente algoritmo que permite calcular la transformada de Fourier discreta (DFT) y su inversa.

3.2. Estructura de un Sistema Embebido

Un sistema embebido se encuentra principalmente compuesto por:

- Microprocesador o Microcontrolador: Cumplen la función de CPU. Los microcontroladores a diferencia de los microprocesadores, disponen en su interior una memoria de programa y de datos, esto abarata costes y espacio, aumentando así la cantidad de entradas y salidas disponibles.
- Comunicación adecuada: Es de vital importancia la comunicación en este tipo de dispositivos electrónicos. El sistema tiene que ser capaz de comunicarse mediante interfaces estándar. Nos centraremos en los relacionados con nuestro proyecto.

  - I^2C: Este bus esta formado por dos lineas conocidas como SDA y SCL. Con este tipo de bus es posible la interconexión de varios dispositivos entre sí. Para el buen funcionamiento, precisa de una resistencia de Pull-Up o de polarización, que suelen tomar valores de 1 kΩ, 2.2 kΩ, 4.7 kΩ o 10 kΩ en ambas líneas, dependiendo de factores como los dispositivos conectados o la velocidad del bus. El papel que desempeñan los dispositivos puede ser: maestro o esclavo. El maestro es el encargado de generar una señal de reloj por SCL que se usará para sincronizar todos los dispositivos. La línea de datos, o SDA, sera la encargada de transmitir la información propiamente dicha. Cada esclavo debe tener configurada su dirección de esclavo, esto es, una variable de 8 bits donde los 7 primeros son un número en binario y el último bit indica si es una lectura o escritura lo que quiere hacer el maestro. (Coquet)

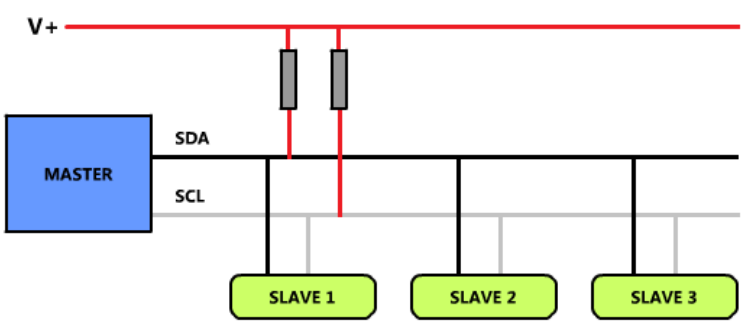


Imagen 2. Esquema de comunicación I^2C.

- UART o "Universal Asynchronous Receiver/Transmitter": es una interfaz de comunicación serie muy sencilla, útil y común. Está compuesto por tres lineas fundamentales: TX, RX y GND. La conexión entre dispositivos es cruzada; de esta forma, el pin de transmisión TX de un dispositivo está conectado al pin de recepción RX del otro dispositivo, y viceversa. De esta manera, la comunicación existe de manera unidireccional; es decir, un extremo transmite y otro extremo recibe en un solo cable. La unidad mínima para transferir son bytes de 8 bits. El envío de cada byte se realiza bit a bit precedido por un bit de inicio, seguido por un bit de paridad opcional y uno o dos bits de parada. (Martín)

  La velocidad de transmisión se mide en bits por segundo (bps) o baudios. Esta unidad será empleada a lo largo del proyecto para configuración de transmisión del Bluetooth. Las velocidades de transmisión más comunes son 9600, 19200, 38400, 115200 , etc.

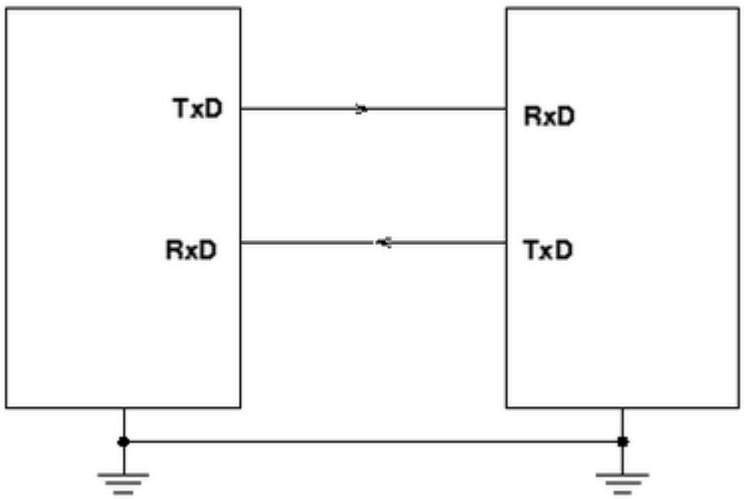


Imagen 3. Comunicación serial asíncrona

- Actuadores: Posibles elementos que el sistema ha de controlar con sus salidas
- E/S analógicas y digitales.
- Módulo de reloj: Encargado de generar las señales de reloj, puede ser tanto interno del uC o externo (cristal de cuarzo o circuito RC).
- Otros: Conversores AC/DC, filtros, etc.

3.3. Microcontroladores PIC

Los microcontroladores PIC (Peripheral Interface Controllers), fueron diseñados por la actual Microchip Technology Inc. Estos MCUs (Micro Controller Units) estan basados en una arquitectura Harvad tipo RISC[2] con un diseño original pensado para ser usado con una CPU de 16 bits, las malas prestaciones de entrada y salida fueron las causantes de la aparición del PIC de 8 bits en 1975, con mejoras en rendimiento.

Imagen 4. Logo Microchip.

Existe una gran diversidad de microcontroladores (desde 4 a 32 bits), es cierto que las prestaciones de los microcontroladores de 16 y 32 bits son superiores a las de 8 bits, aunque éstos son más que suficientes en la mayoría de los casos para realizar las tareas a las que estan destinados. La capacidad de los MCUs de 4 bits en muchos casos insuficiente, y los de 16 o 32, exceptuando aplicaciones especiales que los requieran, excesivamente potentes en la mayoría de los casos, suponiendo asímismo un coste más elevado.

Imagen 5. Ejemplo de Patillaje de un PIC (16F)

[2] Del inglés "**R**educed **I**nstruction **S**et **C**omputer", es un tipo de diseño en la CPU que cuenta con la ejecución de instrucciones de tamaño fijo, y solo las instrucciones de carga y almacenamiento acceden a la memoria de datos.

3.4. Gama de PICs (8 bits)

Entre los PICs de 8 bits podemos diferenciar claramente cuatro familias. Se han elegido las carácterísticas que han sido decisivas en este proyecto, aunque no se han puesto todas, sí las mas importantes. A continuación se expondrá una tabla comparativa con las principales características de cada familia de una forma general y gráfica.

Familia	Precio ($)	M. Progr (Kbytes)	RAM (bytes)	Pines	UART	SPI	I^2C	USB
10F	0.35	0.37-0.9	16-64	6	X	*[3]	*	
12F	0.65	0.75-7	25-256	8	X	*	*	
16F	1-2	3.5-56	128-4k	17-64	X	X	X	*
18F	1-5	4-128	256-4k	18-100	X	X	X	X

Tabla 1. Comparativa de familias de PIC

[3] * Como norma general no disponen de esta característica, aunque pueden existir PICs específicos de la familia que la tengan.

3.5. Otras plataformas

3.5.1. ARDUINO

Imagen 6. Logo Arduino.

Arduino es una herramienta que surgió en Italia en el año 2005 para el desarrollo de productos y proyectos electrónicos en código abierto. Su facilidad en el uso de una tarjeta hardware y de un entorno de programación gratuito para el control de numerosas magnitudes físicas de nuestro alrededor, hace que esta herramienta esté dirigida a todo tipo de usuarios, desde programadores experimentados hasta diseñadores y entusiastas que no hayan programado nunca. (Arduino Website)

Existe una amplia gama de productos Arduino, sin embargo, uno de los más utilizados es el Arduino UNO.

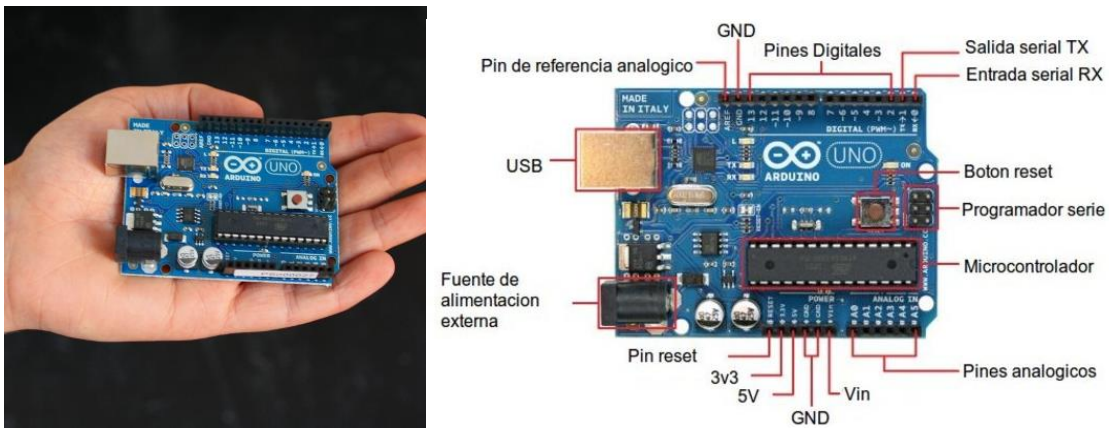


Imagen 7. Arduino UNO. Pines y Puertos.

3.5.2.Raspberry Pi

Raspberry Pi, es una placa de desarrollo que se comporta como un mini PC, pues cumple muchas de las funciones que haría un PC convencional. Es capaz de trabajar con hojas de cálculo, procesar texto y ejecutar juegos. Además de destas funcionalidades, esta placa consta de un puerto HDMI para reproducir video en alta calidad. (Raspberry Pi Website)

Imagen 8. Logo Raspberry Pi.

Aunque este proyecto fue ideado en 2006, no fue lanzado al mercado hasta 2012. Fue desarrollado en la Universidad de Cambridge con la idea de fomentar la enseñanza computacional en los niños.

La placa dispone de varios puertos y entradas, USB, Ethernet y HDMI. Estos puertos permiten conectar Raspberry Pi a otros dispositivos, teclados, ratones y pantallas.

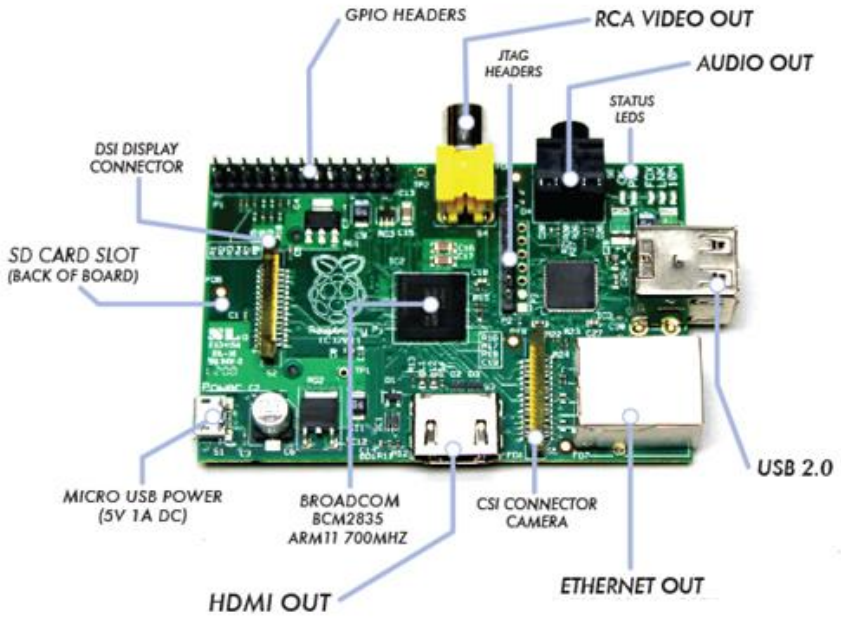


Imagen 9. Puertos de Raspberry Pi.

Basado en un System on Chip Broadcom BCM2835, dispone de un procesador ARM a 700 MHz, una GPU Broadcom VideoCore IV a 250 MHz y hasta 512 Mb de memoria RAM. Es posible instalar sistema operativos libres a través de una tarjeta SD.

Existen cinco modelos, RPI A, RPI A+, RPI B, RPI B+, RPI 2. Las características y el precio varían muy poco entre modelos A, al igual que sucede con modelos B. La versión más reciente y completa es la Raspberry Pi 2.

3.5.3. Otros fabricantes de microcontroladores

El mercado de los microcontroladores es muy amplio, existen fabricantes que desarrollan microcontroladores muy específicos para aplicaciones muy especializadas. A continuación se mencionan algunos de estos. (UPV)

Fabricante	Descripción
National Semiconductor	National Semiconductor: Microcontroladores CompactRISC y COP8. Fabrica un excelente microcontrolador de muy poco consumo con disponibilidad de modelos que incluyen Bluetooth, USB y CAN.
ZILOG	ZILOG: Pioneros del Z80, enfocan su trabajo hacia el Z8 Encore Flash y eZ80.
ANALOG DEVICES	ANALOG DEVICES: Fabricante de DSP's de 16 y 32 bits incluyendo herramientas de desarrollo. Amplia variedad de componentes analógicos de precisión.
CYGNAL	CYGNAL: Familia de Microcontroladores rápidos basados en 8051 incluyendo conversores A/D y sistemas de depurado.
DALLAS SEMICONDUCTOR	DALLAS SEMICONDUCTOR: Microcontroladores Flash rápidos pasados en 8051 añadiendo una bateria para matener los datos en la SRAM. Extensas herramientas de desarrollo.
TEXAS INSTRUMENTS	TEXAS INSTRUMENTS: Líder mundial en DSP's. Produce también microcontroladores de 16 bits de bajo consumo, componentes analógicos y productos de telecomunicaciones.

Tabla 2. Tabla de otros fabricantes de microcontroladores.

CAPÍTULO 4. *Soluciones Adoptadas.*

4. Soluciones Adoptadas

4.1. Por qué elegir PIC

La amplia gama de microcontroladores en el mercado actual hace que la elección de un microcontrolador u otro no sea algo tan evidente. Con lo cual se ha de establecer unos criterios de elección acordes con el proyecto que se quiera tratar. Los criterios de elección fueron los siguientes:

- Documentación
- Fiabilidad
- Precio
- Experiencia previa

La experiencia previa es un factor muy importante ya que si se tiene un conocimiento sobre un tipo de microcontrolador específico, normalmente estos suelen mantener la filosofía de trabajo en cuanto a lo que el conjunto de instrucciones, direccionamiento y registros respecta en las siguientes versiones. Esto supone un ahorro de tiempo y posibles problemas que aparecen cuando se trabaja con un microcontrolador del que no se tiene conocimiento.

La documentación existente es clave cuando no se es experto con algún tipo de microcontrolador, en este caso Microchip cuenta con una cantidad muy elevada de documentación al igual que Arduino, a pesar de su relativa reciente incorporación en el mercado.

Las herramientas de desarrollo juegan un rol muy importante a la hora de decantarse por un microcontrolador ya que puede suponer una ayuda inestimable en el desarrollo del proyecto. Existen fabricantes que ofrencen entornos de desarrollo de forma completamente gratuita como política para inclinarse por el uso de sus microcontroladores (AVR Studio de Atmel, MPLAB de Microchip o Eclipse de Texas Instruments).

El precio del microcontrolador, evidentemente es muy decisivo en la fabricación de este tipo de sistemas electrónicos low-cost en los que se busca una competitividad máxima.

Para este proyecto se barajaron dos posibles soluciones: Arduino y PIC. De acuerdo con los criterios elegidos en el proyecto, ambos poseen herramientas de desarrollo gratuitas así como una amplia gama de documentación y textos bibliográficos.

A partir de este punto, cabe diferenciar que PIC es un microcontrolador y Arduino es una placa de desarrollo basada en los microcontroladores Atmega de Atmel, esto tiene repercusión directa en la compacidad del producto final. Por otro lado, Arduino cuenta con una cantidad inmensa de librerías, que permiten controlar infinidad de dispositivos con una complejidad mucho menor que la que nos ofrece PIC en la mayoría de los casos, y cuenta con una curva de aprendizaje mucho más rápida. Sin embargo, esto también puede llegar a ser contraproducente, ya que goza de menor libertad al ser una placa universal con una estructura limitada.

La principal diferencia radica en el precio y capacidad de selección; mientras Arduino cuenta con alrededor de 20 tipos de placas de desarrollo, Microchip dispone de alrededor de 400 microcontroladores, sólo en la familia de 8 bits. Esto da lugar a que tengamos la capacidad de elegir el microcontrolador correcto, sin pagar más por algunas funciones que encarecen el producto, y no van a ser utilizadas.

Es por esto por lo que se eligió finalmente PIC de Microchip, más concretamente el PIC18F2455, del que se dará una visión general a continuación, con las características adecuadas para nuestro proyecto.

4.2. Hardware

4.2.1.PIC 18F2455

El PIC 18F2455 es un microcontrolador ideal por su baja potencia (en torno a los nanoWatios) y por la posibilidad de conectarse con tres puertos serie: FS-USB(12 Mbit/s), I²C y SPI (hasta 10 Mbit/s) y EUSART.

A parte de estas características que hacen posible la comunicación que se deseaba para este trabajo, una de las razones por la que este microcontrolador es adecuado, es porque dispone de 2 Kbytes de RAM. En un principio se utilizó un PIC de la familia 16F con una memoria RAM de 368 bytes; ésts se tuvo que sustituir por este microcontrolador, con una

memoria superior. La razón de este cambio se debe a que la escritura de datos con SPI se lleva a cabo mediante un sistema de archivos o filesystem FAT. Este requiere de un buffer de de datos de 512 bytes, que superaba la RAM máxima del microcontrolador anterior. Aun así, una microcontrolador que disponga de una RAM de 512 bytes sería insuficiente de igual forma ya que a parte del buffer, el microcontrolador hará uso de la RAM también para la gestión de varibles y otros procesos.

De esta manera, se dispondrá de un PIC con memoria suficiente, posibilidad de comunicación en todos los protocolos requeridos y con un patillaje más que suficiente para gestionar las entradas y salidas.(Véase Datasheet de PIC 18F2455)

Nombre del Parámetro	Valor
Tipo de Memoria de Programa	Flash
Memoria de programa (KB)	24
Velocidad de CPU (MIPS)	12
RAM	2,048
EEPROM (bytes)	256
Periféricos de comunicación digital	1-UART, 1-A/E/USART, 1-SPI, 1- I^2C 1-MSSP(SPI/ I^2C)
Periféricos/Comparadores/PWM	2 CCP
Timers	1 x 8-bit, 3 x 16-bit
ADC	10 canales, 10-bit
Comparadores	2
USB	1, USB 2.0
Rango de Temperaturas (C)	Desde -40 hasta 85
Rango de Operación de Voltaje (V)	Desde 2 hasta 5.5
Nº de Pines	28

Tabla 3. Tabla de características PIC 18F2455.

4.2.2.Módulo Bluetooth HC-06

Considerado una de las opciones más economicas para establecer una comunicación no física entre dos dispositivos, el módulo Bluetooth HC-06 es un dispositivo muy fácil de obtener, económico y sencillo de utilizar.

Imagen 10. Módulo Bluetooth HC-06.

Algunos de los principales motivos por los que el módulo HC-06 se consideró acertado para el proyecto son los siguientes:

- Reducido tamaño. Hace de este módulo una fácil integración en sistemas embebidos.
- Buena comunicación (transmisión y recepción de datos) a una distancia considerable.
- Funcionamiento de bajo consumo de corriente (en torno a 8-10 mA durante la transmisión/recepción de datos) y estado de espera o “Idle” cuando no está conectado a otro dispositivo.
- Su rango de funcionamiento entre 3.6V y 6V lo hacen ideal para trabajar con la tensión de alimentación de prácticamente cualquier microcontrolador.

A pesar de su amplia utilización en el mundo de la comunicación bluetooth low cost, el módulo presenta una desventaja que hace a muchos usuarios decantarse por otro modelo: la escasa información al respecto en la web. Sin embargo la información del hardware ha sido encontrada y llevada a cabo sin problema. (Véase Datasheet HC-06)

4.2.3.RTC PCF8563

La Real-Time Clock (RTC) PCF8563 es una agenda calendario basada en CMOS optimizada para un consumo mínimo. Cuenta con una salida de reloj programable así como una interrupción, que se activará en un tiempo configurable. La RTC, utiliza un bus bidireccional de comunicaciones I^2C, alcanzando una velocidad máxima de transmisión de 400 kbit/s. (Véase Datasheet PCF8563)

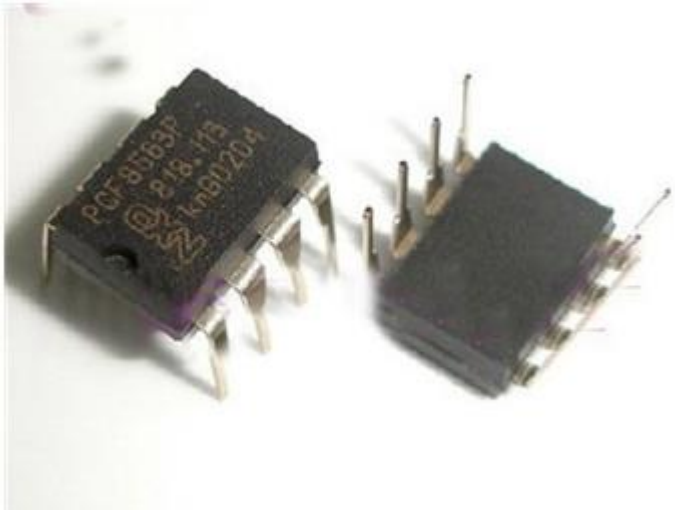


Imagen 11. Real Time Clock PCF8563P

Algunas de las características que hacen esta RTC apropiada para el proyecto son:

- Dispone de un sistema capaz de proveer datos de año, mes, día, día de la semana, horas, minutos y segundos basados en un cristal de cuarzo de 32.768 kHz.
- Umbral de operación del reloj entre 1.0V y 5.5V
- Corriente de backup de 0.25 µA a 3.0V
- Bus I^2C a 400 kHz entre 1.8V y 5.5V
- Alarma y funciones de Timer
- Integra un condensador interno para el oscilador
- Direcciones de esclavo I^2C de lectura (A3h) y escritura (A2h)
- Pin de Interrupción

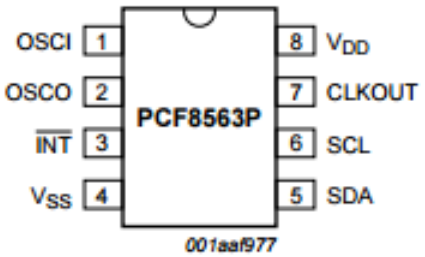


Imagen 12. Esquema de pines RTC 8563.

4.2.4.Microchip MCP9800

MCP9800 es un sensor de temperatura de alta precisión de Microchip Technology que opera entre temperaturas de -55 ºC y +125 ºC. Se consideró apropiado este sensor principalmente por la capacidad de conversión directa de la temperatura en una palabra digital. Con esto, nos ahorraremos la conversión analógica digital y la comunicación podrá ser establecida mediante I^2C.

El sensor proporciona una precisión de hasta 12 bits de resolución con un error de ±1 ºC (máximo) desde -10 ºC a +85 ºC, la cual es más que suficiente para nuestro propósito.

La familia MCP9800 ofrece registros programables por el usuario para proporcionar flexibilidad en aplicaciones de detección de temperatura. Los ajustes de registro permiten una resolución de medición de temperatura de 9 y 12 bits seleccionable por el usuario, una configuración del modo de apagado para el ahorro de energía, así como la especificación de salida de alerta de temperatura y límites de la histéresis.

Es posible generar una alerta cuando la temperatura alcance un límite previamente programado. Esta señal de alerta puede ser configurada de tal manera que la polaridad de salida de un comparador sea activo a nivel bajo o activo a nivel alto.

Este sensor tiene una interfaz serial compatible I^2C , permitiendo que se controlen hasta ocho dispositivos en un solo bus serial. Estas características permiten que el MCP9800 sea ideal para aplicaciones sofisticadas de control de la temperatura de varias zonas.(Véase Datasheet MCP9800)

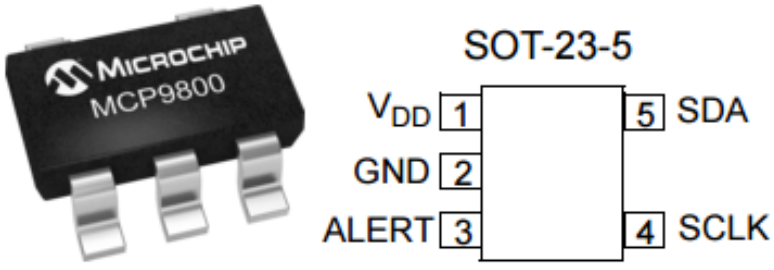


Imagen 13. Sensor de Temperatura MCP9800 en encapsulado SOT-23-5.

4.2.5. PicKit 3

PICkit 3 es un dispositivo que permite la depuración y la programación de microcontroladores PIC y dsPIC utilizando el entorno de desarrollo integrado (MPLAB), desarrollado por Microchip. PICkit 3 se conecta al PC mediante una interfaz USB F-S. El conector utiliza dos pines I/O y la línea de reset para implementar la depuración in-circuit y la programación In-Circuit Serial ™. (Microchip Website)

Imagen 14. PICkit 3 Programmer/Debugger.

Algunas de las características son:

- FS-USB (12 Mbits/s)
- Ejecución en tiempo real
- MPLAB IDE compatible
- Firmware actualizable desde Internet o PC
- Soporta bajo voltaje (2.0 V a 6.0 V)
- LEDs indicadores (power, busy, error)
- Lectura y escritura de la memoria de los microcontroladores
- Borrado del espacio de memoria con verificación
- Capacidad de congelamiento de periféricos en punto de interrupción (Breakpoint)
- Programación de hasta 512K Flash con la característica Programmer-to-Go

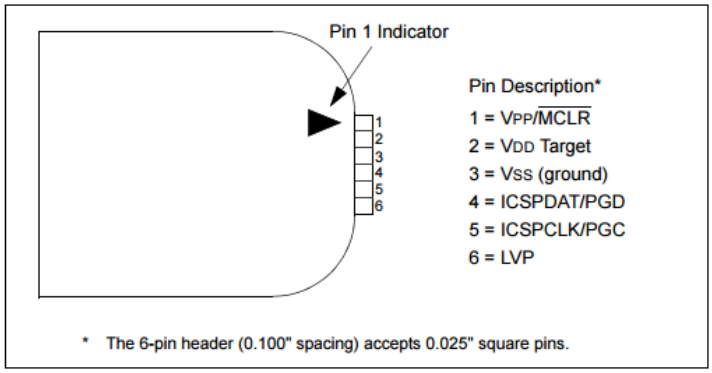


Imagen 15. Descripción de pines.

4.3. Software de desarrollo del sistema

A continuación, se detallará el software empleado tanto para el soporte físico del datalogger como para la aplicación de control en Android.

4.3.1. Compiladores PIC C (PCW) y MikroC (Mikroelektronika)

PCW/CCS es un entorno de desarrollo integrado/compilador que cuenta con una interfaz gráfica que ayuda a usuarios de distintos niveles de experiencia a realizar tareas desde la más básica, hasta las mas compleja.

Imagen 16. Logo CCS.

A continuación se muestra la interfaz del entorno y parte del código de nuestro proyecto, esto se debe a que el programa empezó a desarrollarse con este compilador pero no llegó a finalizarse con el mismo.

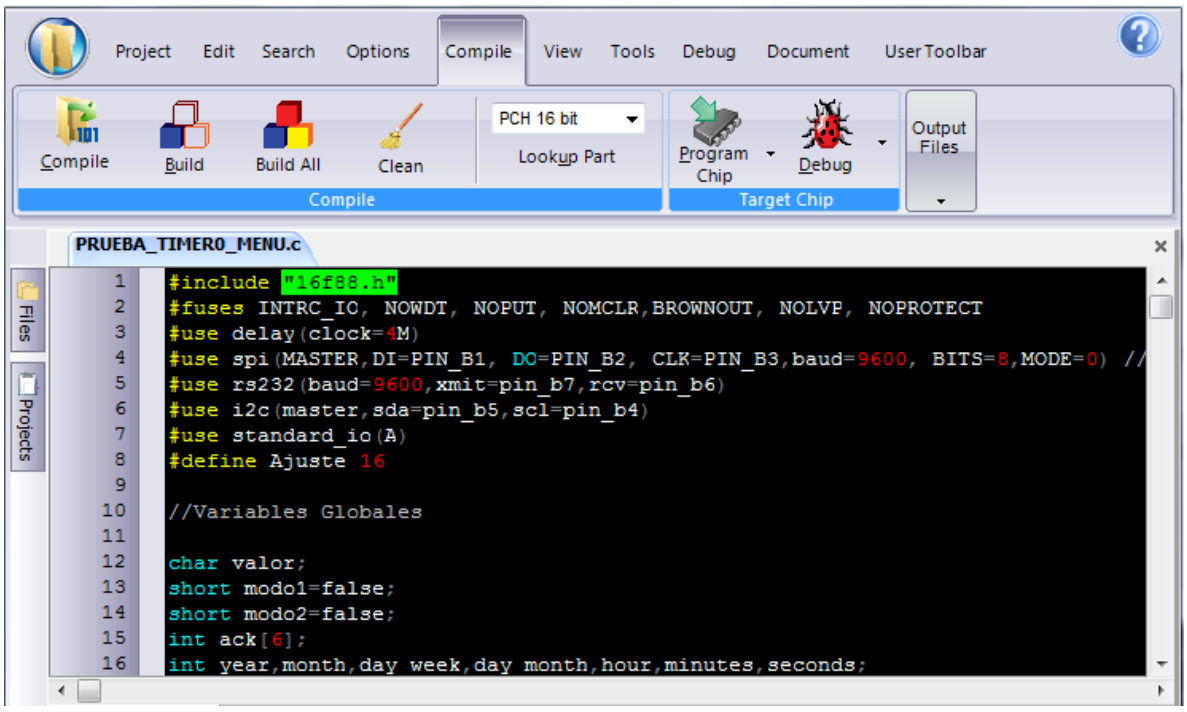


Imagen 17. Entorno de desarrollo PCW.

Se probaron distintos compiladores hasta dar con uno, cuya programación fuera cómoda y funcional. Surgieron además numerosos problemas relacionados con la utilización de librerías para el uso de un sistema de archivos FAT basado en la escritura de datos a través de SPI. Aparecieron errores en librerías, que llamaban a funciones de otras librerías relacionadas

con el sistema de archivos FAT y CCS. Dichos errores, que arrojaba el compilador en niveles muy profundos de las librerías y a un alto nivel de programación, hiceron muy difícil la adaptación a CCS.

Sin embargo, MikroC cuenta con funciones propias y específicas para el manejo de tarjetas SD además de las pertinentes para establecer las comunicaciones necesarias.

Imagen 18. Logo MikroC.

Exceptuando el inconveniente de la migración de código de un compilador a otro, que no fue una tarea breve, este compilador ofrecía una programación mucho mas cómoda y eficaz gracias a una muy buena organización de la documentación.

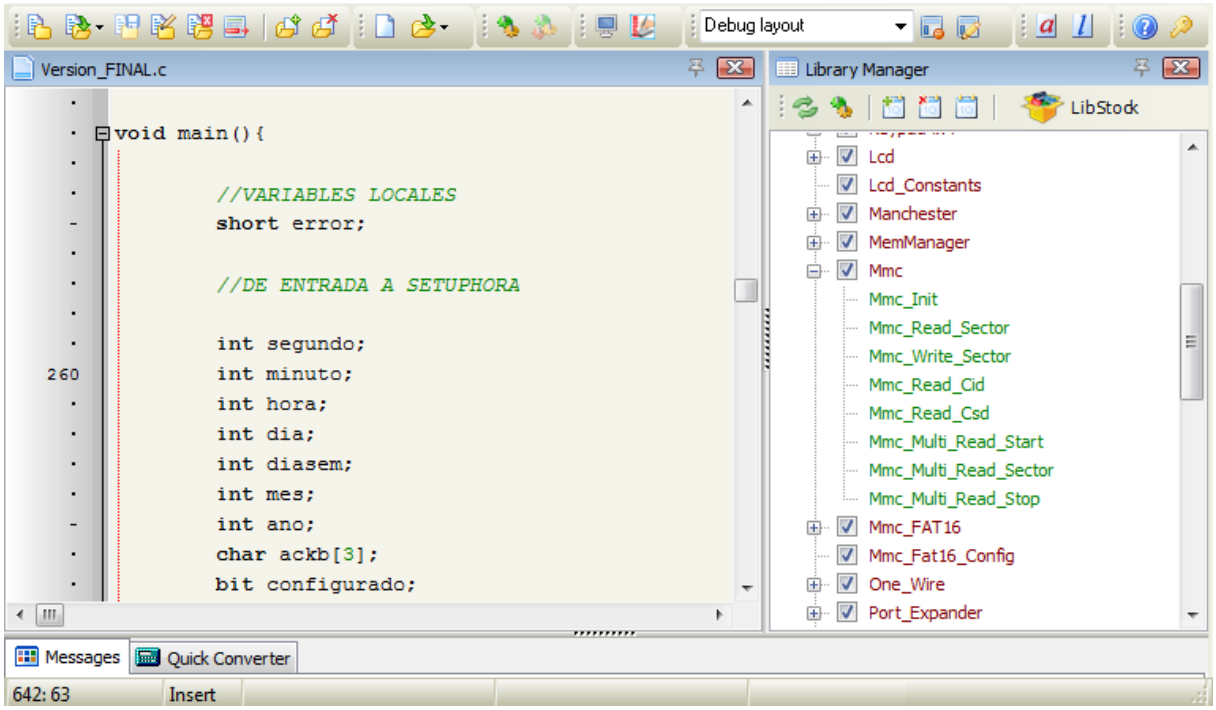


Imagen 19. Entorno de desarrollo MikroC.

4.3.2.Proteus VSM

Proteus VSM es un entorno de diseño electrónico de Labcenter Electronics que brinda la posibilidad de simular códigos generados en hexadecimal (aunque también es compatible con otros archivos) de alto y bajo nivel con la simulación de SPI-CE. Proteus se subdivide en tres potentes entornos de trabajo como son el ISIS, para el diseño gráfico, VSM (Virtual System Modelling) para simulación, y ARES: para diseño de PCB. (Labcenter Website) (Breijo, 2009)

Imagen 20. Ventajas que ofrece Proteus VSM.

4.3.3. Hyperterminal

Hyperterminal es un cliente para hacer conexiones telnet por medio de los puertos serie con dispositivos externos. Estos dispositivos pueden variar e incluyen opciones tales como equipos de radio comunicación, robots, instrumentos utilizados para mediciones científicas y equipos de red.

Este software fue utilizado junto con VSPE en un principio, aunque posteriormente se empleó el terminal que proporciona Proteus VSM.

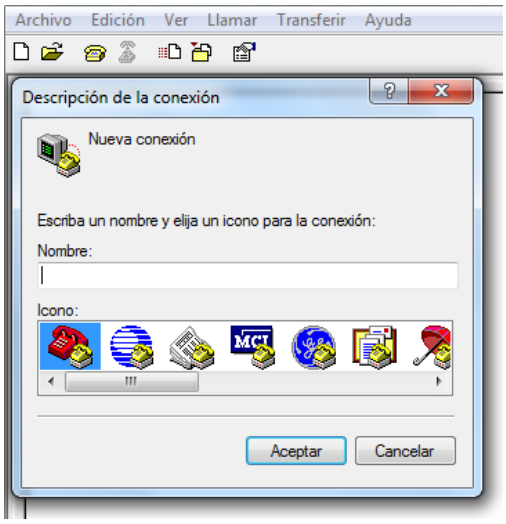


Imagen 21. HyperTerminal de Windows.

4.3.4. VSPE

VSPE es un software diseñado para ayudar a testear aplicaciones que usen puerto serie. Este programa es capaz de crear varios dispositivos virtuales para transmitir o recibir datos. A diferencia de los puertos series físicos, se crean enlaces virtuales que permiten la comunicación con programas de simulación como en nuestro caso Proteus ISIS. (Eterlogic Website)

En nuestro caso, se ha utilizado la opción 'Pair' para enlazar el puerto COM de Proteus con el que se usará en HyperTerminal. Se configurará la conexión a 9600 baudios y no se seleccionará la opción de Paridad.

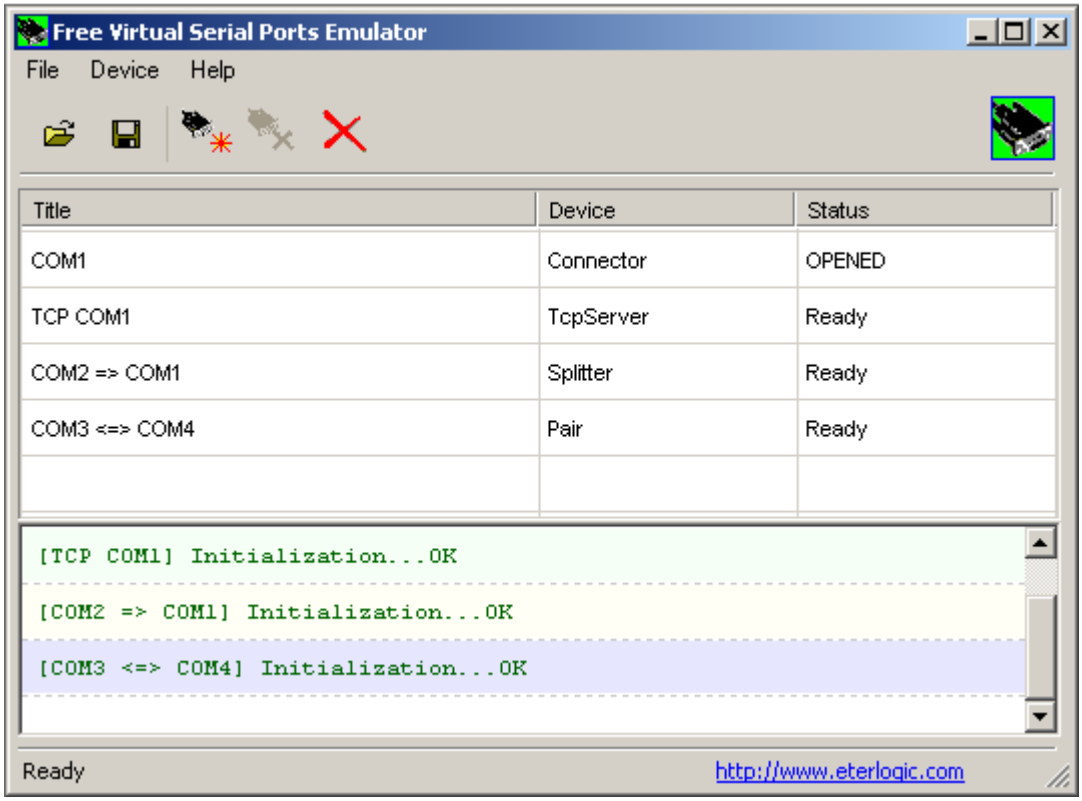


Imagen 22. Interfaz de Virtual Serial Ports Emulator

4.3.5. MPLAB IPE

Imagen 23. Entorno de programación de PIC.

El entorno de programación integrada MPLAB (IPE) es un software que ofrece una interfaz simple para programar nuestro microcontrolador. Dispone de un entono de programación seguro para programar cualquier PIC.

El entorno de programación opera en dos modos:

- Production Mode: Es el modo por defecto que ofrece este software, cuenta con todas las funciones necesarias para programar nuestro microcontrolador.

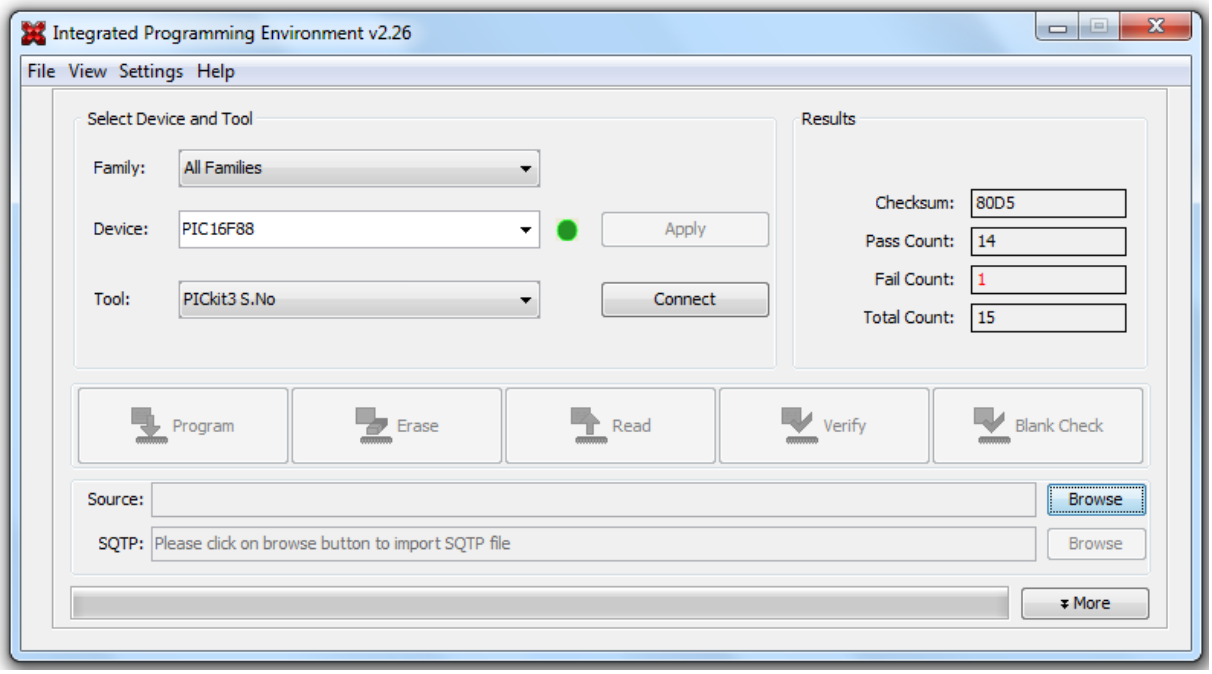


Imagen 24. Interfaz de programación MPLAB IPE.

- Advanced Mode: Es el modo avanzado del programa. Se accede mediante la contraseña por defecto ‘microchip’ y dispone de opciones como por ejemplo la alimentación del microcontrolador a través del programador, en nuestro caso, Pickit 3. Para nuestro propósito dentro de la opción ‘Power’ en el apartado de opciones de ICSP se seleccionará ‘Power Target Circuit from Tool’, para alimentar nuestro PIC a 5 V.

4.4. Software de desarrollo para aplicación Android

4.4.1. Processing, Java Developer Kit y Android Developer Kit

Java es uno de los lenguajes de programación más utilizados con una capacidad muy elevada para el desarrollo de programas para cualquier plataforma.

Imagen 25. Logo Java.

Para la elaboración de la aplicación en Android que nos permita la comunicación con nuestro datalogger se ha utilizado Processing.

Imagen 26. Logo Processing.

Este lenguaje de programación y entorno de desarrollo de código abierto está basado en Java, esto permite heredar todas sus funcionalidades convirtiéndose así en una herramienta muy útil y potente para el desarrollo de cualquier tipo de proyectos.

Existe un modo en Processing que permite el desarrollo de aplicaciones Android, instalando previamente Java Development Kit y Android Developer Kit.

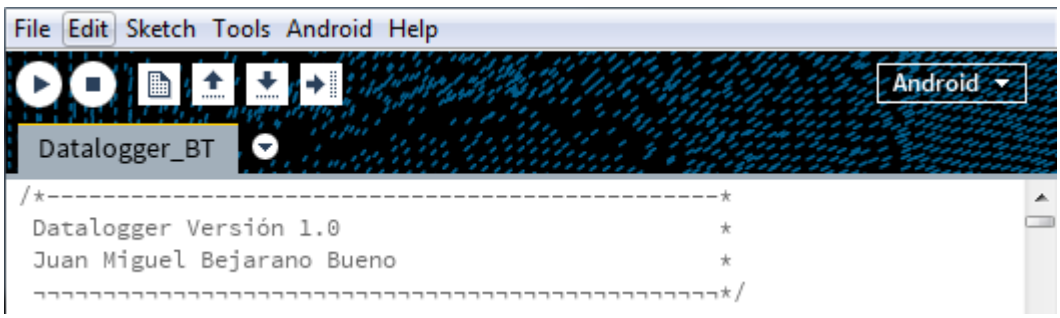


Imagen 27. Desarrollo de la app Android desde Processing

Para ello se ha descargado de la página oficial el JDK de Java 8 U40, es decir, la versión 8, update 40. Una vez llegado a este punto, es preciso distinguir entre JDK y JRE. JDK es el kit que el programador necesita para poder desarrollar aplicaciones en Java, incluye el JRE (Java Runtime Environment) que contiene la máquina virtual Java, y es necesario para poder ejecutar aplicaciones creadas en este lenguaje.

Java SE Development Kit 8u51

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux x86	146.9 MB	jdk-8u51-linux-i586.rpm
Linux x86	166.95 MB	jdk-8u51-linux-i586.tar.gz
Linux x64	145.19 MB	jdk-8u51-linux-x64.rpm
Linux x64	165.25 MB	jdk-8u51-linux-x64.tar.gz
Mac OS X x64	222.09 MB	jdk-8u51-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.36 MB	jdk-8u51-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	98.8 MB	jdk-8u51-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	139.79 MB	jdk-8u51-solaris-x64.tar.Z
Solaris x64	96.45 MB	jdk-8u51-solaris-x64.tar.gz
Windows x86	176.02 MB	jdk-8u51-windows-i586.exe
Windows x64	180.51 MB	jdk-8u51-windows-x64.exe

Imagen 28. Disponibilidad de JDK para plataformas (Versión 8, Update51).

De igual forma, es necesario la incorporación de Android SDK, este kit de desarrollador puede conseguirse en la web oficial de desarrolladores Android. Este paquete, disponible para Windows Mac OS y Linux, está comprendido por un debugger, un simulador de terminales, documentación, tutoriales y ejemplos de código. Cuenta con un entorno de desarrollo integral oficial (Android Studio), además el programador podrá utilizar comandos en un terminal como es el caso de nuestro proyecto.

Para ello es necesario la instalación de los paquetes JDK y Apache Ant, con ambos será posible crear y depurar aplicaciones, así como llevar el control de los dispositivos Android conectados.

Para la conexión y depuración de nuestro programa, en primer lugar conectaremos nuestro terminal en modo depuración en los ajustes del teléfono. Una vez configurado este modo e instalados los drivers de google para USB que proporciona el paquete SDK, un mensaje aparecerá cada vez que conectemos nuestro terminal al PC por USB.

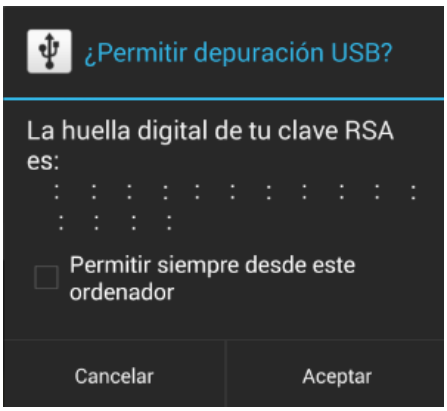


Imagen 29. Conexión USB para comunicación con terminal de Processing.

Una vez conectado el terminal, Processing lo reconocerá y aparecerá en la lista de dispositivos vinculados.

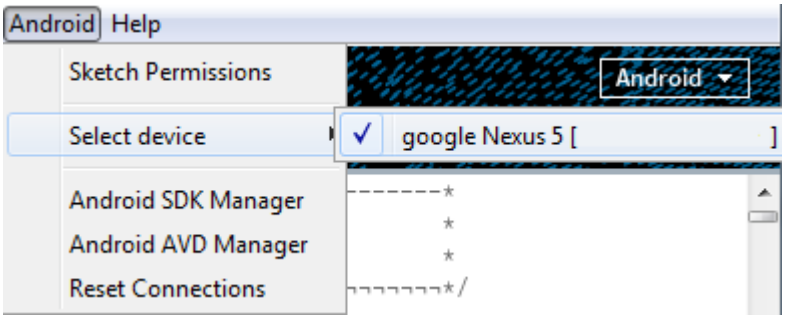


Imagen 30. Selección de dispositivo desde Processing.

Realizada correctamente la conexión se podrá monitorizar el estado del Bluetooth del dispositivo Android en el terminal de Processing tal y como se muestra a continuación.

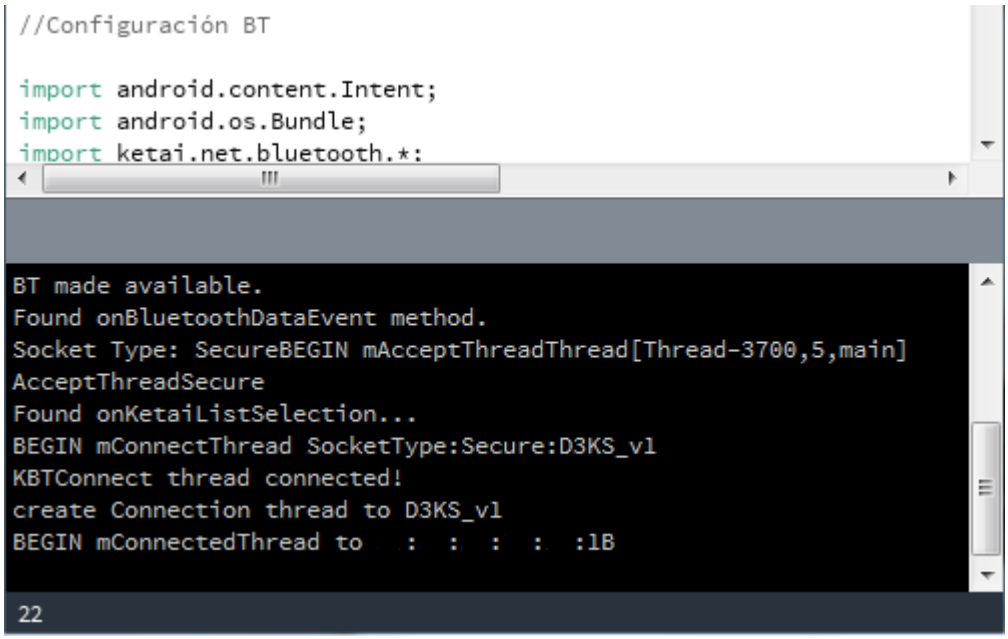


Imagen 31. Conexión del dispositivo con terminal de Processing.

Como se puede observar, el Bluetooth esta disponible, obtenemos el mensaje de inicio de conexión: "BEGIN mConnectThread SocketType:Secure:D3KS_v1". Este mensaje aparece porque el bluetooth del dispositivo ha sido reconocido y se intentará conectar. Posteriormente se consigue la conexión y se muestra la MAC de nuestro dispositivo Android.

CAPÍTULO 5. *Descripción del Proyecto.*

5. Descripción del Proyecto

A continuación se detalla el funcionamiento de nuestro datalogger de manera general, para ir profundizando en aspectos más específicos del mismo. En la siguiente imagen se expone el funcionamiento del sistema.

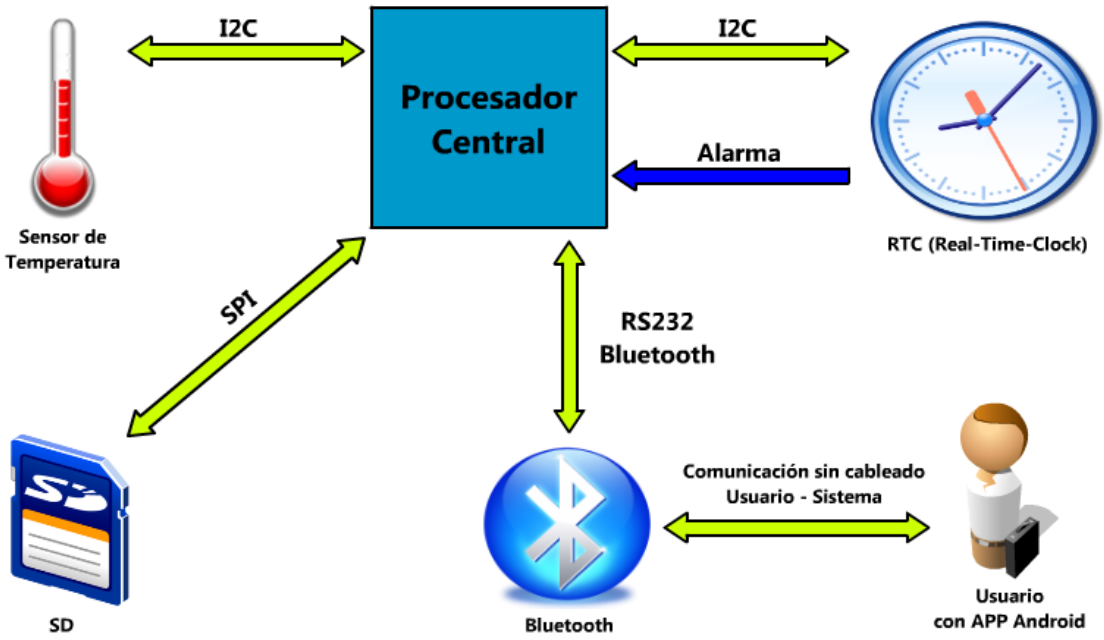


Imagen 32. Diagrama de bloques del sistema.

Como podemos observar, el núcleo del sistema es el procesador central. En nuestro caso este procesador viene a ser un microcontrolador, más concretamente un PIC 18F2455. Éste es el encargado de gestionar todos los recursos del sistema y de establecer las comunicaciones pertinentes para el correcto funcionamiento, que será detallado posteriormente.

Por otro lado, se dispone de un sensor de temperatura, más concretamente, el MCP 9800 de Microchip, encargado de recopilar datos, y transmitirlos de manera digital al procesador central mediante el bus I^2C.

De manera análoga, una RTC (Real-Time-Clock) se encargará de recopilar los datos del tiempo actual, la cual se ha programado con anterioridad. Esta RTC dispone de un sistema de alarma, que es capaz de activar una interrupción que despierta a nuestro sistema de un modo sleep de bajo consumo para realizar las tareas de lectura o programación del sistema. Tanto la puesta en hora, como consulta de datos se realiza mediante I^2C nuevamente.

Para hacer posible una interacción con el usuario, se dispone de un módulo Bluetooth (HC-06), que se comunicará con el dispositivo Android de dos posibles maneras. La primera, de cara a visualizar el funcionamiento del programa y para posibles "debugs" en líneas de futuro, a través de un terminal. Y la segunda enfocada a un uso nivel usuario, que es la aplicación desarrollada en Android.

El sistema hasta ahora, es capaz de ofrecernos datos en tiempo real de temperatura; sin embargo, estos datos han de ser recopilados de manera que puedan ser usados en un futuro. Es por ello por lo que se ofrece al usuario dos posibilidades. La primera se basa en la escritura de datos en una memoria de almacenamiento externo Secure Digital (SD) mediante el protocolo de comunicaciones SPI. La segunda sin emabargo, esta basada en la escritura de datos en la memoria interna de nuestro dispositivo Android, según los datos transmitidos por Bluetooth, mediante un protocolo de comunicación serie RS232.

Una vez descrita de forma general la línea de funcionamiento del sistema, se procederá de manera más específica en aspectos de hardware y software.

5.1. Aspectos de Hardware

El funcionamiento del microcontrolador y el del sistema en general se lleva a cabo mediante una alimentación de 5 V, aunque es preciso aclarar que para un correcto funcionamiento se han realizado adaptaciones de nivel en los siguientes casos:

- En el módulo bluetooth se ha incorporado un divisor de tensión en la patilla RX para acondicionar la señal a 3.3 voltios, sin embargo no es necesario en la patilla TX ya que el microcontrolador ya reconoce los 3.3 voltios como estado alto.

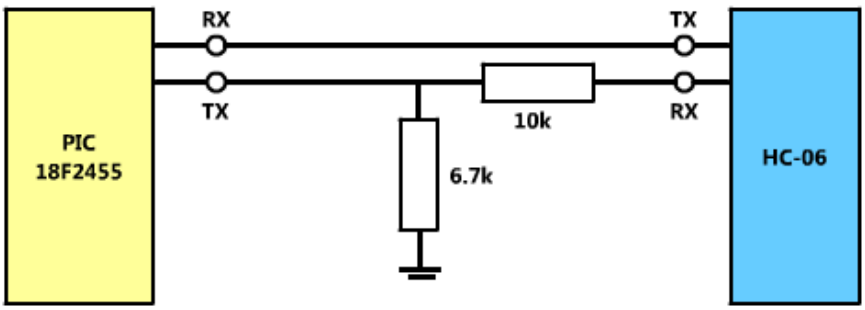


Imagen 33. Adaptación de nivel para módulo HC-06.

- El adaptador de SD del que se ha dispuesto en el proyecto incluye un regulador de tensión a 3.3 V con lo cual no es ningun problema la alimentación a 5 V. Sin embargo, la salida de los pines del PIC es también a 5 V, esto supone un problema para nuestra SD ya que este voltaje podría dañarla seriamente.

  Algunos adaptadores disponen de lo que se conoce como "lever shifter" o adaptador de nivel. En nuestro caso, no disponemos deeste adaptador en nuestra placa. Es por ello por lo que se ha pensado la utilización de un buffer hexadecimal CMOS no inversor, más concretamente el 4050BE de Texas Instruments (Véase Datasheet de Buffer Hexadecimal 4050BE). El circuito integrado es alimentado con 3.3 V de tal forma que los 5 V que llegan a la entrada del buffer salen a 3.3 V en la patilla contigua.

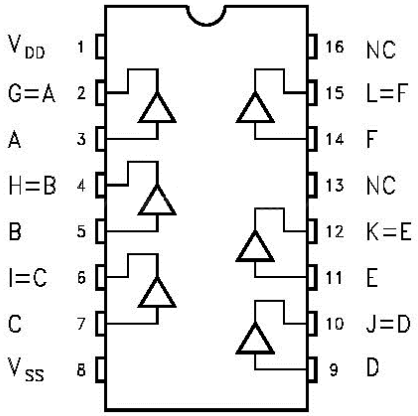


Imagen 34. Circuito Integrado CP4050BE.

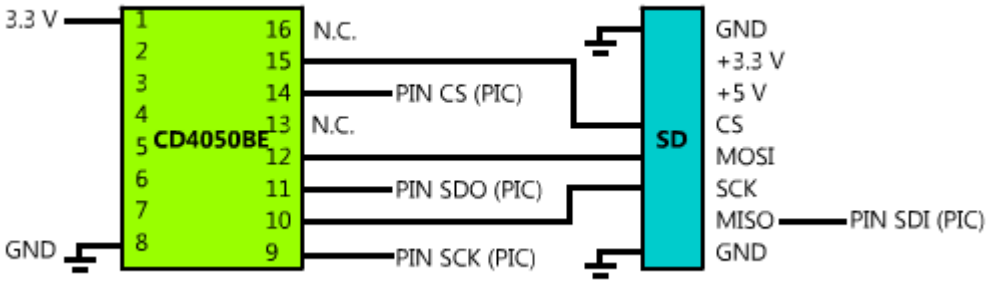


Imagen 35. Adaptación de nivel para módulo SD.

- Para que la RTC mantenga la hora es necesaria alimentarla constantemente. Se ha utilizado una alimentación con una pila de 3 V externa de manera que funcione tanto cuando el circuito esté alimentado como cuando no. Para ello se han empleado dos diodos 1N4148 (Véase Datasheet 1N4148) según queda representado en el siguiente esquema.

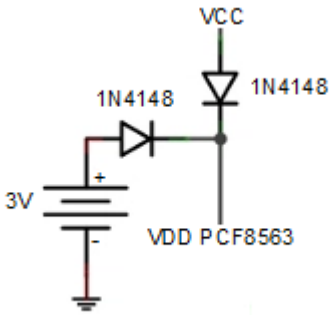


Imagen 36. Diodos para "Backup Battery".

- Por otro lado, debemos tener en cuenta la corriente que queremos que circule por los LEDs que nos permiten seguir el funcioamiento del programa de manera visual. Se ha fijado una corriente de funcionamiento en torno a 15 mA, con lo cual mediante un cálculo simple a partir de la ley de Ohm:

$$R = \frac{V}{I} = \frac{5}{0.015} = 333.33 \approx 332\ \Omega$$

Se obtiene una resistencia de 332 Ω (normalizada) para cada LED.

Las resistencias de Pull-up o resistencias de polarización juegan un papel muy importante para llevar a cabo las comunicaciones I^2C. Se han empleado por ello resistencias de Pull-up con un valor estándar de 10 kΩ, aunque 4.7 kΩ es el valor que se recomienda para velocidades menores de 100 kbps, solo se utilizan dos dispositivos en modo “Slave” con lo cual para nuestro proposito es un valor adecuado.

Para el correcto funcionamiento del microntrolador, se ha dotado de un cristal de cuarzo externo de 8 MHz entre los pines 9 y 10 de nuestro PIC, de manera análoga la RTC dispone de otro cristal de cuarzo, pero en este caso 32.768 kHz tal y como aparece en el datasheet de la misma. (Véase Datasheet PCF8563).

5.2. Aspectos de Software

El diseño del programa se ha llevado a cabo con el software de Mikroelektronika MikroC PRO for PIC. Este programa permite generar de manera automática, a través de una interfaz al inicio del programa, los parámetros y registros en hexadecimal imprescindibles para el correcto funcionamiento del microcontrolador.

5.2.1. Configuración de fusibles y registros

Entre los distintos "fuses" que oferta nuestro microcontrolador, podemos encontrar en primer lugar PPL Prescaler selection, la función de este fuse es la división de la frecuencia de entrada al oscilador OS1 y OS2 en distintas escalas según el cristal de cuarzo empleado. Esto se consigue mediante la configuración de PLLDIV, el cual funciona como un multiplexor. En nuestro caso se ha seleccionado la división por dos ya que se utiliza, como comentamos anteriormente, un cristal de cuarzo externo de 8MHz. De este modo tendremos 4 MHz a la salida del multiplexor.

La selección del Postscaler, sin embargo se encuentra predeterminada a "Primary Oscillator Src: /1, 96 Mhz PLL Src: /2" y no se modificará el valor. El clock del USB usado sólo para el modo "Full-Speed" esta directamente determinado por el bloque del oscilador primario sin "postscale".

Indicar de manera correcta al compilador cual es la selección del oscilador que queremos hacer, es uno de los aspectos mas importantes en la programación de nuestro microcontrolador. Una de las opciones más utilizadas en este tipo de proyectos es la implementación del oscilador interno mediante la selección de la opción: "Internal Oscilator, port function on RA6, EC used in USB" con esta opción configuraremos el PIC para trabajar con el oscilador interno y al mismo tiempo usar el pin RA6 como puerto de entrada/salida. La selección de la frecuencia viene determinada por el registro OSCCON.

REGISTER 2-2: OSCCON: OSCILLATOR CONTROL REGISTER

R/W-0	R/W-1	R/W-0	R/W-0	R[(1)]	R-0	R/W-0	R/W-0
IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **IDLEN:** Idle Enable bit
1 = Device enters Idle mode on SLEEP instruction
0 = Device enters Sleep mode on SLEEP instruction

bit 6-4 **IRCF2:IRCF0:** Internal Oscillator Frequency Select bits
111 = 8 MHz (INTOSC drives clock directly)
110 = 4 MHz
101 = 2 MHz
100 = 1 MHz[(3)]
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 31 kHz (from either INTOSC/256 or INTRC directly)[(2)]

bit 3 **OSTS:** Oscillator Start-up Time-out Status bit[(1)]
1 = Oscillator Start-up Timer time-out has expired; primary oscillator is running
0 = Oscillator Start-up Timer time-out is running; primary oscillator is not ready

bit 2 **IOFS:** INTOSC Frequency Stable bit
1 = INTOSC frequency is stable
0 = INTOSC frequency is not stable

bit 1-0 **SCS1:SCS0:** System Clock Select bits
1x = Internal oscillator
01 = Timer1 oscillator
00 = Primary oscillator

Note 1: Depends on the state of the IESO Configuration bit.
2: Source selected by the INTSRC bit (OSCTUNE<7>), see text.
3: Default output frequency of INTOSC on Reset.

Imagen 37. Registro OSCCON de PIC 18F2455

Sin embargo, para nuestro propósito se ha seleccionado "HS Oscillator" pues como ya se ha mencionado anteriormente nuestro sistema funciona con un cristal de cuarzo externo de 8 MHz. La razón por la que se ha seleccionado "HS Oscillator" en lugar de "XT Oscillator" es que realmente ambas opciones estan referidas a un oscilador externo, sin embargo HS Oscillator (High-Speed Oscillator) se utiliza para cristales con una frecuencia superior a 4 MHz. Para frecuencias menores a 4 MHz se deberá seleccionar "XT Oscillator".

Las opciones de Fail-Safe Clock Monitor, Internal/External Oscillator Switchover y Power-Up timer se encuentras deshabilitadas para nuestro propósito, sin embargo el Brown-out Reset, encargado de reiniciar el microcontrolador en caso de una caída en el voltaje (que podría causar valores de salida indeseados si el microcontrolador sigue funcionando a un voltaje mínimo) , se encuentra habilitado.

De manera análoga, se ha deshabilitado el regulador de voltaje USB, así como el Watchdog Timer. Este es el encargado de resetear el microcontrolador tras un periodo de tiempo determinado. El funcionamiento del mismo es similar a la interrupción provocada por un desbordamiento del timer, sin embargo en este caso, se produciría un Reset del microcontrolador. Tanto el bit MUX CCP2 como el pin de reset MCLR y la programación a bajo voltaje se han deshabilitado, el puerto B configurado como entradas analógicas o digitales también ha sido desactivado, pues no se trabajará con ninguna señal analógica en este proyecto.

Entre las distintas opciones que se ofertan para la protección de la memoria, no se ha protegido ningun bloque de código para la escritura o la lectura.

Con esta configuración se generarán unas líneas de configuración en hexadecimal que albergarán la información necesaria para el correcto funcionamiento del microcontrolador.

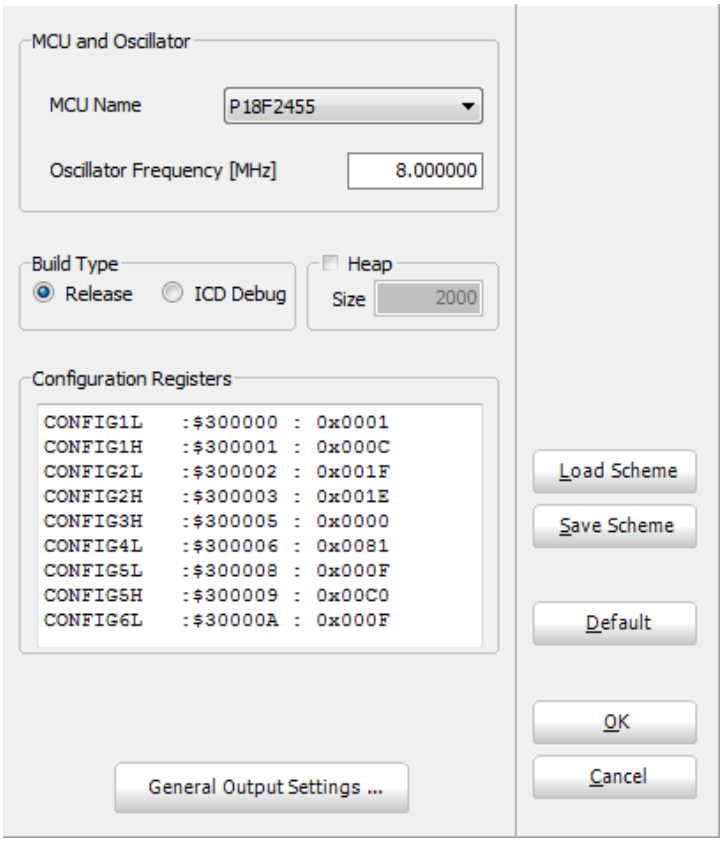


Imagen 38. Código de registros de configuración en hexadecimal.

Al inicio del programa, se han modificado los registros de configuración para: utilizar el puerto A como entradas y salidas digitales, desactivar los comparadores y habilitar las interrupciones.

```
//Configuracion como salidas en Puerto A

        ADCON1 |= 0x0F;

//Comparadores desactivados

        CMCON  |= 7;

//Puerto A de salida e inicializado a "0" Puerto B configurado como entrada en pin 2 (Interrupción)

        TRISA=0;
        TRISB=TRISB | 0b00000100;
        PORTA=0;

//Interrupciones

        INTCON.GIEH=1;          //Habilita las interrupciones globales de prioridad alta
        INTCON.GIEL=0;          //Deshabilita las interrupciones globales de prioridad baja
        INTCON.RBPU=1;          //Resistencias de Pull-Up internas desactivadas
        INTCON.PEIE=1;          //Interrupciones Externas de periféricos Habilitadas
        INTCON2.INTEDG2=0;      //Interrupciones en INT2 activas por flanco de bajada
        INTCON3.INT2IP=1;       //Interrupcion en INT2 considerada de alta prioridad
        INTCON3.INT2IF=0;       //Flag a 0
        INTCON3.INT2IE=1;       //Habilita la interrupción externa de INT2
        RCON.IPEN=1;            //Habilita prioridades en interrupciones
```

5.2.2. Gestión de comunicaciones.

Como ya se ha mencionado anteriormente, el proyecto requiere de tres tipos de comunicación, estos son: I^2C, SPI y RS232. Nuestro microcontrolador PIC dispone un patillaje adecuado para cada comunicación, sin embargo no existe la posibilidad de implementar estas tres comunicaciones de manera simultánea por hardware.

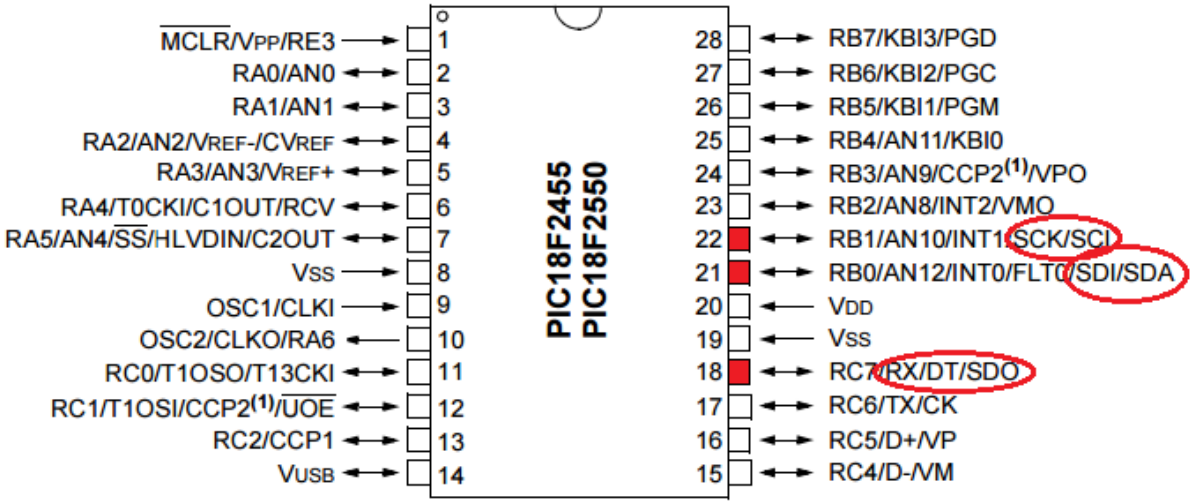


Imagen 39. Pines que entran en conflicto en comunicaciones por hardware I^2C,SPI y UART

Como se puede observar, es posible la implementación simultanea de I^2C y UART, sin embargo cuando entra en juego la comunicación SPI, utiliza los mismos pines que I^2C y UART en el caso de los pines 18, 21 y 22.

Es por ello por lo que se implementará la comunicación UART e I^2C por software y SPI por hardware. La elección de pines ha sido llevada a cabo de tal manera que estos no interfieran en otros pines con funciones especiales.

De este modo, para la UART se han seleccionado los pines RC1 como RX y RC2 como TX del puerto C, configurado a una velocidad de transmisión de 9600 baudios.

```
//INICIALIZACIÓN DE LA UART

Soft_UART_Init(&PORTC,1,2,9600,0);
```

Para el I^2C, se han empleado bits del puerto B. Para establecer la comunicación I^2C por software basta con definir los bits de manera que SCL esté en el pin RB3 y SDA en el pin RB4, así como la dirección.

```
//CONFIGURACION I2C

    sbit Soft_I2C_Scl at RB3_bit;
    sbit Soft_I2C_Sda at RB4_bit;

    sbit Soft_I2C_Scl_Direction at TRISB3_bit;
    sbit Soft_I2C_Sda_Direction at TRISB4_bit;
```

La implementación de SPI por hardware se llevará a cabo utilizando los pines correspondientes según la hoja de características. Como sabemos, la comunicación SPI dispone de la entrada de datos del "Slave" al microcontrolador por el pin RB0 habilitado como SDI, la señal de reloj o SCK situada en el pin RB1 y la salida de datos hacia el "Slave" por el pin RC7 hablitado como SDO.

Para ello se utiliza la librería de MikroC que permite la inicialización del mismo, sin embargo se utilizará la librería avanzada que permite la configuración deseada. Así pues tendremos una división de la frecuencia de oscilación en 64 (disminuyendo así la velocidad de cara a la correcta inicialización de la tarjeta SD), establecemos un muestreo en la mitad del intervalo, el estado de espera del clock en nivel bajo, y un flanco de transmisión ascendente.

```
//INICIALIZACION SPI

    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV64,
    _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW,
    _SPI_LOW_2_HIGH);
```

La inicialización y escritura de la tarjeta SD, se llevará a cabo definiendo en primer lugar los registros de función especial de Chip Select (asignado en el pin C0 del puerto C) así como las variables necesarias para el funcionamiento de las librerías.

```
//SD Chip Select

    sfr sbit Mmc_Chip_Select at LATC0_bit;
    sfr sbit Mmc_Chip_Select_Direction at TRISC0_bit;

// Variables para librerías MMC

    unsigned char SectorData[512];          //Buffer para la Lectura/escritura de SD
    unsigned char data_for_registers[16];  //Buffer para los registros CID y CSD

//Inicialización de la SD

    do{
    error = MMC_Init();
    Soft_UART_Write_Text("SD NO ENCONTRADA!\r\n");
    delay_ms(1000);
    }while(error==1);
```

Una vez inicializada la tarjeta SD procederemos a inicializar el sistema de archivos FAT para la gestión de nuestro archivo de texto donde se guardará toda la información recopilada.

```
//Inicialización FAT

     MMC_Fat_Init();

     SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV4,
     _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW,
     _SPI_LOW_2_HIGH);
```

Como ya se indicó anteriormente, la manera correcta de inicialización de las tarjetas es a una velocidad lenta. Una vez inicializada correctamente la misma y el sistema de archivos FAT, se aumentará la velocidad del SPI cambiando de una frecuencia de oscilación entre 64 a una frecuencia de oscilación dividida entre 4.

```
//Creación de archivo y escritura de la tabla

    Mmc_Fat_Assign(&filename,0xA0);
    Mmc_Fat_Append();
    Mmc_Fat_Write("Grados  Décimas  Año  Mes  Día  Hora  Minuto
    Segundo\r\n",60);
```

Se indica que será un archivo (0x20), y si no existe dicho archivo (0x80), con la máscara 0xA0 se creará y se le asignará el nombre ya definido en nuestro programa con la variable "filename".

Bit	Máscara	Descripción
0	0x01	Sólo Lectura
1	0x02	Oculto
2	0x04	Sistema
3	0x08	Etiqueta de Volumen
4	0x10	Subdirectorio
5	0x20	Archivo
6	0x40	Dispositivo (sólo uso interno)
7	0x80	Flag de creación de archivo. Si el archivo no existe, se creará uno nuevo con el nombre especificado en "filename".

Tabla 4. Tabla de atributos de MikroC para creación de archivos

5.2.3.Funciones Auxiliares.

Para evitar tareas repetitivas se han creado una serie de funciones auxiliares que facilitan tanto la escritura, como la lectura del programa.

En primer lugar, dado que la codificación de la fecha y hora de la RTC se basa en código BCD, se han creado funciones muy sencillas para agilizar la gestión de la misma introduciendo los datos en decimal. De este modo, la misma se encarguará de realizar la conversión automática, evitando de esto modo posible errores.

```
//Conversion de BCD a Decimal

    int bcdToDec(int valorbcd)
    {
    return ((valorbcd / 16) * 10 + valorbcd % 16);
    }


//Conversion de BCD a Decimal

    int decToBcd(int valordec)
    {
    return (valordec / 10 * 16 + valordec % 10);
    }
```

Aunque el tipo de datos que se emplee a la hora de realizar la programación de un microntrolador pueda parecer uno de los aspectos más sencillos, una mala gestión de los mismos puede suponer un problema bastante importante si no se presta atención en cada momento al tipo de dato con el que se trabaja. Es por ello por lo que tanto en la programación del microcontrolador como en la de la aplicación Android (en mayor medida) se han empleado funciones de conversión de datos.

```
//Conversión de datos tipo char a int

    int charToInt(char numero)
    {
    int num;
    num=numero-'0';
    return num;
    }
```

Esta función surge debido a la necesidad de comunicarse a través de la UART del microcontrolador con la aplicación Android en cuestión a través del módulo Bluetooth. La comunicación se lleva a cabo mediante el envío y recepción de caractéres (datos tipo char) como veremos a continuación.

Imagen 40. Tipos de datos en la comunicación Bluetooth.

Por esta razón se ha diseñado un algoritmo para la captura de datos de fecha y hora. Una vez que se entra en la parte del programa encargada de capturar los datos, el mismo permanecerá en estado de espera hasta recibir un valor para almacenarlo en la variable "valor". Es necesario incluir la variable "error" en la función Soft_UART_Read(), esta variable es un flag que almacenará un valor. Dicho valor será 0 si el dato se ha recibido correctamente, 1 si ha habido algún tipo de error y 255 si se ha abortado por el usuario llamando a la función Soft_UART_Break.

Una vez introducido algún carácter, se procederá a evaluarlo:

Si el valor es distinto al carácter empleado para iniciar la comunicación, en nuestro caso será '>', ni es el encargado de salir del caso que está ejecutando el programa ('e'), pedirá de nuevo dato hasta que alguno de estos sea seleccionado.

Si el valor es el empleado para salir del caso actual ('e') es pulsado, volverá al menú principal.

Esto resulta en dos posibles opciones para el usuario. Estas son el inicio de la comunicación, o bien salir del caso actual. Para entender algo mejor el funcionamiento en la captura de datos se expondrá un esquema explicativo.

Imagen 41. Algoritmo para la captura de datos.

La comunicación se iniciará cuando se reciba por el puerto serie el caracter '>', una vez llegado a este punto espera el primer dato. Cuando este es recibido se almacena en un array de caracteres de 7 posiciones 'datos1'. Este primer dato sera almacenado en la posición 1, y justo después se incrementará la posición del mismo para pasar al siguiente dato cuando se reciba entre dato y dato un '*' que será el separador que estableceremos.

En un principio se diseñó dicho algoritmo para mandar y recibir datos de dos dígitos ya que con cualquier número superior al 9 sería necesario. Sin embargo, el rango de números que requiere la puesta de fecha y hora, oscila entre 0 y 60, con lo cual se pensó en mandar sólo un dato como carácter y posteriormente convertirlo a entero. Esto sería posible puesto que para la conversión de carácter a entero empleamos la función anteriormente nombrada (charToInt) que resta el '0' (48 en ASCII) al carácter de entrada obteniendo así un rango de operación desde el 0 hasta el 207. De esta manera, se lograría una simplificación en el algoritmo y en el programa en general.

```
//Algoritmo para captura de datos en la puesta en hora

        do{
        valor=Soft_UART_Read(error);
        if ((valor=='>') && (valor!='e'))
        {
            Soft_UART_Write_Text("Inicia comunicacion:\r\n");
            do
            {
                dato1[pos]=Soft_UART_Read(error);
                pos++;
                Soft_UART_Write_Text("Dato: OK");
                valor=Soft_UART_Read(error);

                if (valor=='*')
                {
                    Soft_UART_Write_Text("Dato: RECIBIDO\r\n");
                }
```

```
            else if ((valor=='#') || (valor=='e'))
            {
                cambio='#';
            }

        }while (cambio!= '#');

        //Instrucciones del programa
    }

    ...

    else if (valor=='e')
    {
        //Gestion de LEDS
        valor='\0';
    }
}while(exitA==0);
```

En esta parte del código podemos observar que para realizar un seguimiento cuando se inicia el programa desde un terminal se ha empleado la función Soft_UART_Write_Text(). Ésta función, que es incorporada en la librería de la UART por hardware, no es incorporada en la librería de la UART por software. Es cierto que se incorpora Soft_UART_Write(), sin embargo, ésta función es capaz de escribir sólo un carácter. Es por ello por lo que ha surgido la necesidad de crear una función capaz de escribir texto a partir de la misma.

```
//Función de envío de cadenas de texto por UART


void Soft_UART_Write_Text(char *text)
{
int length;
int x;
int pos;
length=strlen(text);
```

```
for (x=0;x<length;x++)
{
    if (text[pos]!=0x00)
    {
        Soft_UART_Write(text[pos]);
        pos++;
     }
}
pos=0;
length=0;
}
```

Esta función tiene como parámetro de entrada una cadena de texto cuyo tamaño será medido con la función strlen(). Esta cadena será analizada posición por posición y se imprimirá carácter a carácter si la posición del vector no contiene el carácter nulo (0x00).

5.2.4.Funcionamiento del programa.

Una vez descritas las inicializaciones de las comunicaciones y las funciones auxiliares utilizadas, se procederá a describir el funcionamiento general del programa.

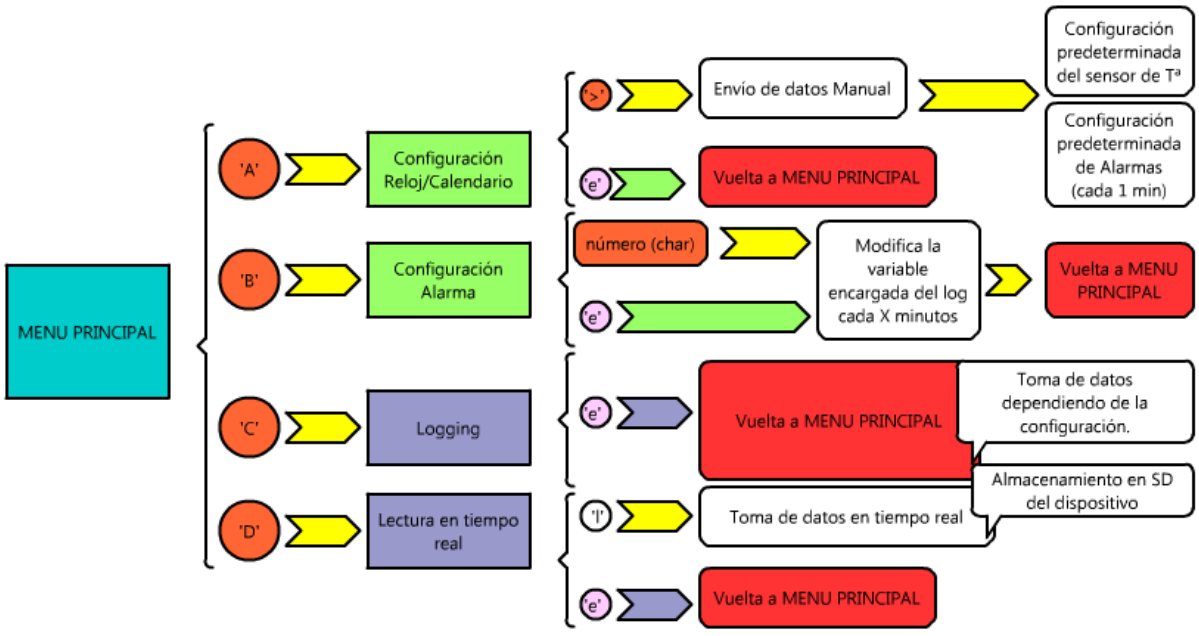


Imagen 42. Esquema de funcionamiento del programa.

El objetivo principal de este diagrama es transmitir de manera gráfica y a simple vista las posibles opciones que se barajan con nuestro programa. Para una información más detallada al respecto se deberá consultar el código de programa. (Véase Código Fuente PIC, incluido en CD)

Como hemos podido observar, el programa ha sido diseñado como máquina de estados. Esta máquina de estados consta de 5 casos: un menú principal, un menú para la configuración de una RTC, un menú de configuración de alarmas, una opción para establecer el modo en bajo consumo realizando registros según la alarma previamente programada y una opción de lectura en tiempo real.

Inicializadas todas las comunicaciones antes descritas, se entrará en un bucle de funcionamiento infinito. Partiendo del menú principal, una lectura del puerto serie del carácter 'A' nos hará entrar en el menú de configuración del reloj/calendario. La puesta de fecha y hora se llevará a cabo con el procedimiento ya descrito en el capítulo 5.2.3 "Funciones Auxiliares".

Una vez establecidas la fecha y hora deseada el programa, se encargará automáticamente de llamar a la función SetupHora(). Esta función consta de 7 parámetros de entrada (segundo, minuto, hora, día semana, día mes, mes y año) recogidos previamente por el código encargado de recopilar los datos. Esta función se encargará de iniciar el protocolo de comunicaciones I^2C. Una vez iniciada la comunicación y establecido el modo de escritura de agenda, se mandarán los datos a las direcciones de la RTC correspondientes.

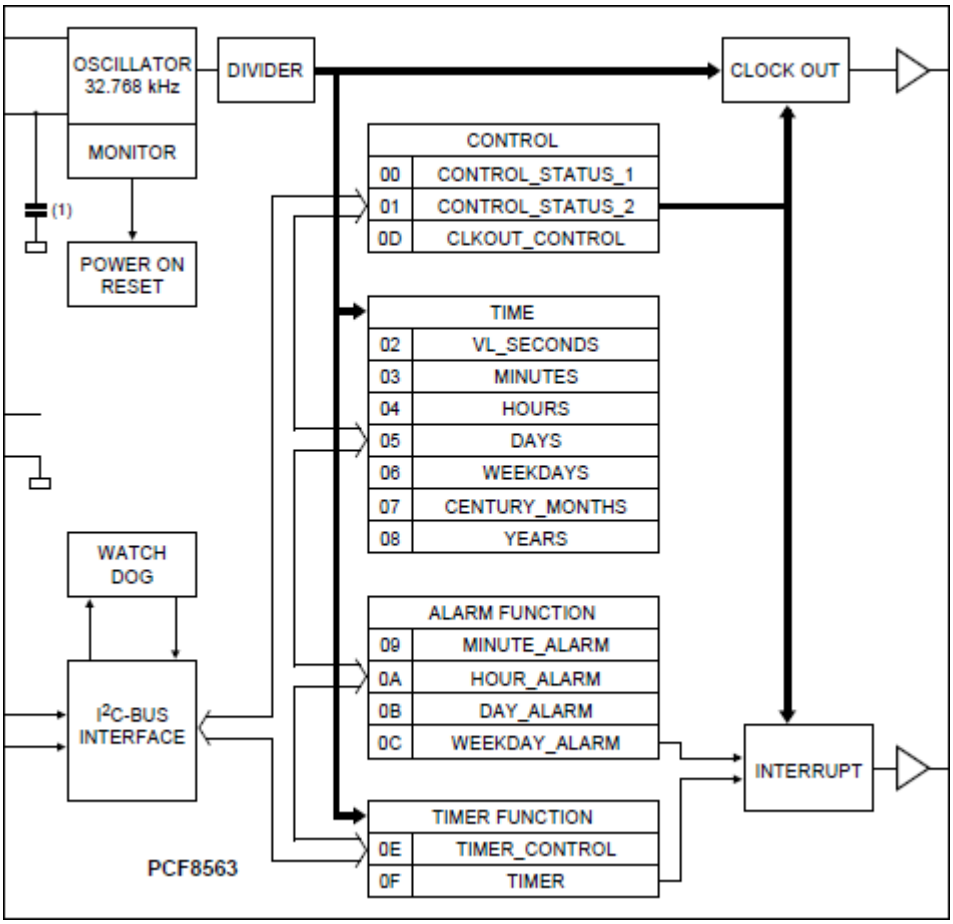


Imagen 43. Registros de RTC 8563.

Como ya se ha explicado anteriormente, y según indica el fabricante, la codificación de los parámetros de esta RTC es la BCD, con lo cual este momento es cuando el uso de las funciones de conversión de datos bcdToDec() y decToBcd() es de vital interés.

Para comprobar el correcto funcionamiento en el envío de datos, se ha definido un array de datos ack[] que nos permitirá saber si éste ha sido correctamente enviado o no. Para ello, esta variable tomará el valor de 0 si ha sido correctamente enviado, y un 1 si ha habido algun tipo de colisión en la escritura de datos con cualquier otro 'Slave'. Para garantizar que los datos son mandados de una forma correcta se ha empleado un ciclo do-while que se ejecutará hasta que todos los acks tomen el valor 0. Se puede observar que aunque el primer dato que se recoge es el dato del año y el último el valor de segundo, para la puesta en hora es justo al contrario. Así pues, justo cuando se recoge el dato de segundo, este es escrito inmediatamente para minimizar al máximo un posible error en la puesta en hora.

Si el programa se está ejecutando desde un terminal se recibirá por el puerto serie una notificiación (cadena de texto) de la correcta puesta en hora.

De forma seguida se configurará el sensor de temperatura. Se establecerá un valor standard de sensibilidad de 0.25º C modificando los bits 5-6 del registro de configuración correspondiente, dejando las demás opciones con un valor predeterminado.

REGISTER 5-3: CONFIGURATION REGISTER (CONFIG) – ADDRESS <0000 0001>b

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
One-Shot	Resolution		Fault Queue		ALERT Polarity	COMP/INT	Shutdown
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **ONE-SHOT** bit
1 = Enabled
0 = Disabled (Power-up default)

bit 5-6 ΣΔ **ADC RESOLUTION** bits
00 = 9 bit or 0.5°C (Power-up default)
01 = 10 bit or 0.25°C
10 = 11 bit or 0.125°C
11 = 12 bit or 0.0625°C

bit 3-4 **FAULT QUEUE** bits
00 = 1 (Power-up default)
01 = 2
10 = 4
11 = 6

bit 2 **ALERT POLARITY** bit
1 = Active-high
0 = Active-low (Power-up default)

bit 1 **COMP/INT** bit
1 = Interrupt mode
0 = Comparator mode (Power-up default)

bit 0 **SHUTDOWN** bit
1 = Enable
0 = Disable (Power-up default)

Imagen 44. Registro de Configuración de resolución.

Se vuelve a emplear un bucle do-while, que comprueba los acks a 0 para asegurar el correcto envío de los datos. Sin embargo, al completarse la acción se activará un flag (configurado) para evitar que se vuelva a entrar en esa parte del código si ya se ha puesto en hora la agenda con anterioridad. Esto agilizará el código, y evitará la sobreescritura de algo ya configurado.

De la misma manera que se configura el sensor con unos valores iniciales predeterminados, se llamará a una función ConfigAgenda(), para establecer unos valores de alarma iniciales que podrán ser modificados en un futuro.

Esta funcion se encargará de leer mediante I^2C el estado actual de la agenda. Una vez leído el dato de minutos, puesto que es la base de tiempos elegida, se almacenará en una variable m_test que será incrementada en una unidad y convertida a BCD preparada para ser escrita a continuación. Por lo tanto habilitaremos sólo y exclusivamente la alarma global cuando el flag de minutos llegue al siguiente minuto. De esta forma nos garantizaremos que el sistema de alarmas se pone en funcionamiento al minuto siguiente. Esto no significa que se va a realizar la medición oportuna en un minuto, sino que se entrará en la rutina de atención a la interrupción y se activará el flag que hará posible la medición de datos y reestablecimiento de flags cuando el programa se encuentre en el modo Logging (de bajo consumo). El funcionamiento de este modo será explicado más adelante.

Hasta aquí el funcionamiento del modo de configuración del reloj/calendario, a continuación se detalla el funcionamiento de la configuración de alarmas. Esta opción es algo mas sencilla que la de configuración de fecha y hora, esto se debe a que sólo modificamos la variable global 'in_minutos' que será usada para avisar al datalogger cada cuánto tiempo queremos que el mismo realice tareas de lectura de datos y escritura.

Una vez descritas las opciones de configuración de la agenda se pasará a detallar las opciones de registro de datos.

La opción Logging surge debido a la necesidad de poner el dispositivo en un modo bajo consumo mientras se mantenía en espera para tomar nuevos datos. De esta manera, ahorraríamos mucha energía y por ende se aumentaría el tiempo de duración de las baterías.

Sin embargo, esto complica algo mas la programación, ya que es imprescindible el uso de interrupciones que 'despierten' a nuestro dispositivo de este modo 'sleep'. Llegado a este punto, el problema aparece cuando el compilador de MikroC no permite la ejecución de ningun tipo de función que aparezca en el código del programa principal. Esto limita de una manera considerable, pues obliga a la activación de un flag cuando se entra a la interrupción y a una posterior ejecución del código deseado que tendrá que ser ejecutado en la funcion principal o main().

Una vez que se activa el flag en la rutina de atención a la interrupción, se recopilan los datos actuales de fecha y hora, se escriben por SPI, se muestran en la UART, y se consulta la variable local que antes fue configurada en la opción de configuración de alarmas

(in_minutos). Esta variable que puede variar desde 1 minuto hasta más de 1 mes. La información es recibida en minutos con lo cual es analizada por una parte del código que se encarga de distinguir en minutos, horas, días e incluso meses. Cabe destacar que es capaz de actuar de la manera apropiada según el mes en el que se encuentre y detectar si el año actual es bisiesto o no.

Una vez recibidos los datos se actualizará nuevamente los nuevos flags y se entrará en modo 'sleep'.

La lectura de datos en tiempo real es la última opción, en esta se ofrece al usuario realizar una lectura puntual cuando el mismo lo requiera. La principal ventaja que ofrece esta opción es un registro inmediato de los datos, sin que exista cableado ni contacto físico, en la memoria interna del dispositivo Android en un archivo de texto (.txt). Es por tanto un método de registro bastante rápido y efectivo.

Imagen 45. Esquema de consulta puntual con la opción Lectura en Tiempo Real.

El manejo del datalogger con la aplicación Android es mucho más gráfico e intuitivo. Se omiten aspectos como el de establecer de manera manual la comunicación de captura de datos, la busqueda del carácter ASCII correspondiente para el envío de fecha o el envío de caracteres como comandos para entrar en los distintos menus entre otros. Aunque la apliación Android limita algo más que el programa que usemos como terminal, a nivel de usuario se facilita el funcionamiento mucho más y se evitaría a la vez que el mismo introdujera datos no deseados, que supondrían un funcionamiento erróneo del programa. Para obtener información sobre el uso de la aplicación consultar el documento Manual de Usuario.

5.3. Desarrollo de la APP Android

El desarrollo de la aplicación Android surge con la necesidad de creación de un entorno dónde el manejo y configuración del dispositivo sea mucho mas llevadero, de tal manera que pueda ser usado por cualquier tipo de usuario aunque no disponga de conocimientos en programación.

5.3.1. Diseño de los recursos

Todos los recursos necesarios del programa han sido diseñados cuidadosamente con un programa de edición de imagen hasta dar el aspecto deseado. En primer lugar se crearon los recursos para interaccionar con el programa, es decir botones principalmente, y posteriormente los distintos menús con sus distintas funciones.

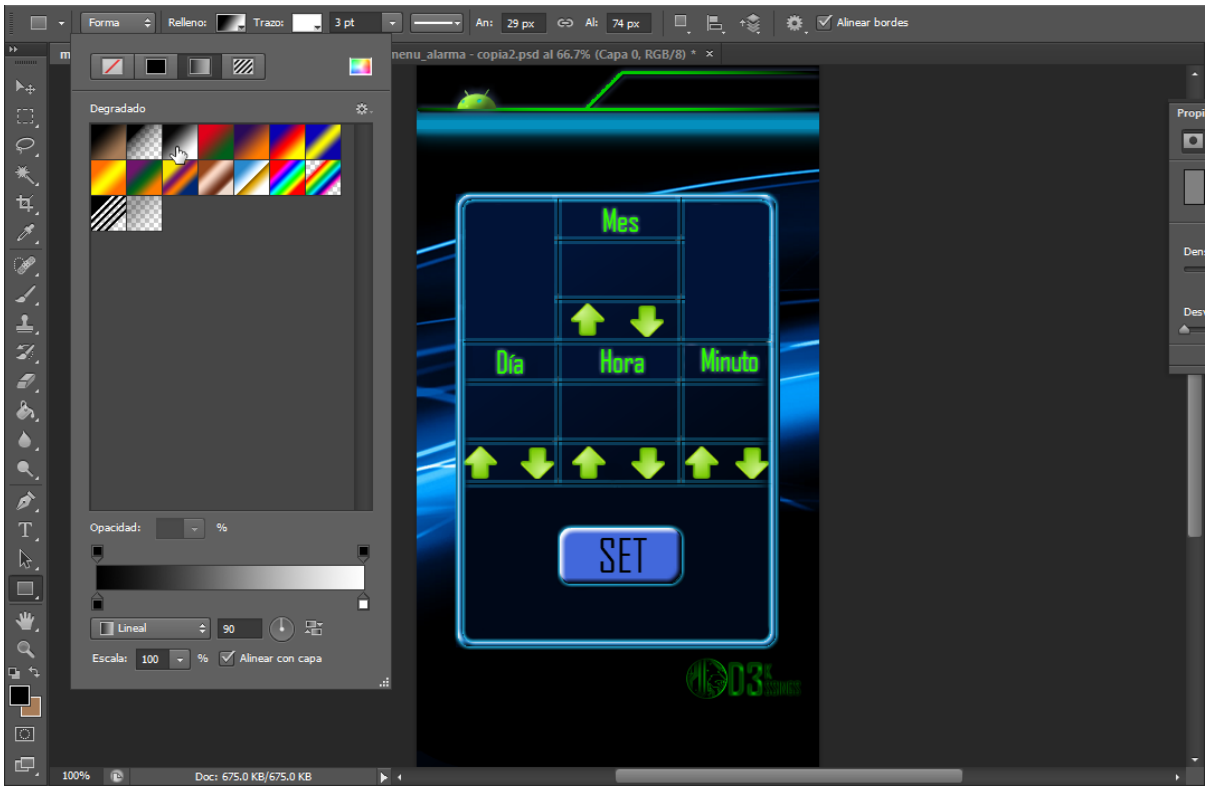


Imagen 46. Diseño gráfico de la APP Android

Las imágenes se pueden encontrar en la carpeta ‘data’ de nuestro proyecto, que es el directorio donde el programa buscará el recurso cuando sea necesario.

5.3.2.Librerías

La aplicación ha sido diseñada con Processing como ya se había comentado anteriormente. Sin embargo, el uso de librerías ha sido imprescindible por el grado de complejidad que supone establecer la comunicación de nuestro dispositivo Android con otro dispositivo sin las mismas.

Para ello, se ha empleado la clase KetaiBluetooth de la librería Ketai. KetaiBluetooth adminisitra las conexiones Bluetooth de los dispositivos Android. Aunque para nuestro propósito utilizaremos una comunicación entre dos dispositivos, dispone de la opción para la conexión de hasta tres conexiones Bluetooth simultáneas. La referencia de esta clase con la descripción de las funciones puede encontrarse en su web oficial. (Ketai)

La inclusión de la misma en nuestro proyecto se lleva a cabo añadiendo el archivo Ketai.jar en la carpeta 'code' del mismo.

5.3.3. Programa Principal

La aplicación en Android ha sido diseñada al igual que el programa del datalogger mediante máquinas de estado.

Imagen 47. Diagrama de estados del programa.

Sin embargo, antes de introducirnos en la explicación de sus posibles estados, se procederá a explicar las funciones auxiliares que se han empleado para el correcto funcionamiento.

En primer lugar se definirán las variables de configuración del Bluetooth y se importaran las librerías necesarias.

```
//Configuración BT

    import android.content.Intent;
    import android.os.Bundle;
    import ketai.net.bluetooth.*;
    import ketai.ui.*;
    import ketai.net.*;

    PFont fontMy;
    boolean bReleased = true;        //Evitará envío permanente cuando el
                                      dedo toca la pantalla

    KetaiBluetooth bt;
    boolean isConfiguring = true;   //Variable que indica que se esta configurando
    String info = "";
    KetaiList klist;
    ArrayList devicesDiscovered = new ArrayList();

//Fin configuración BT
```

Como se puede observar, se define la variable ‘bReleased’ para evitar el envío permanente de datos, indicando al programa cuando se esta pulsando o no el touchpad de nuestro dispositivo Android. De forma similar se emplea otra variable booleana ‘isConfiguring’ que indicará a la funcion draw(), encargada de graficar en pantalla, que se está configurando el Bluetooth. La variable ‘info’ se define como cadena de texto, en ella se almacenará toda la información recibida del datalogger al dispositivo Android. Posteriormente, la función draw() creará una lista al iniciar el programa con los dispositivos vinculados para establecer conexión.

```
//Creación de lista de dispositivos disponibles Bluetooth

    if (isConfiguring==true)
    {
        ArrayList names;
        background(78, 93, 75);
        klist =new KetaiList(this, bt.getPairedDeviceNames());
        isConfiguring = false;
    }
```

A parte de la definición de variables, se emplean distintas funciones de envío y recepción de datos. (Véase Código Fuente APP Android, incluido en CD)

Processing no incorpora una función delay como tal, con lo cual se ha creado como se muestra a continuación empleando la función sleep().

```
//Función Delay

    void myDelay(int ms)
    {
    try{
            Thread.sleep(ms);
        }catch(Exception e) { }
    }
```

Tal y como comentabamos en el caso del datalogger, la gestión de datos puede convertirse en un problema si no se trabaja con los datos que corresponden en cada momento. Para nuestro caso, la función de conversión de enteros a bytes está impuesta por la librería encargada de la transmisión de datos vía Bluetooth, esta librería sólo admite datos tipo byte[], es por esto por lo que la creación de una función de conversión es vital.

```
//Función para conversión de enteros a bytes

byte[] intToABytes(int i)
{
    byte[] result = new byte[4];
    result[0] = (byte) (i >> 24);
    result[1] = (byte) (i >> 16);
    result[2] = (byte) (i >> 8);
    result[3] = (byte) (i /*>> 0*/);
    return result;
}
```

De igual forma, será necesario la creación de una función send_char() que se encargue de convertir los datos tipo char en bytes. La función se encarga de transformar un carácter en un array de caracteres de una posición. A partir de este se creará un array de bytes permitido por la función 'bt.broadcast()'.

```
void send_char(char dato)
{
    char[] cadena= {dato};
    byte[] envio=new String(cadena).getBytes();
    bt.broadcast(envio);
    output.println(envio);
    myDelay(50);
}
```

Los datos recopilados con la función de lectura en tiempo real se recopilarán en un archivo de texto. Este archivo de texto será creado en la raiz de nuestro dispositivo (en este caso se ha empleado un Nexus 5 que dispone de una tarjeta SD interna).

```
//Escritura de archivos

PrintWriter output;
output = createWriter("/sdcard/Log.txt"); //Dirección de archivo donde se
escribirá con el nombre "Log.txt"
```

A lo largo del programa se podrá observar que se trabaja a nivel de píxel es por ello por lo que se crea en la función setup() un espacio de trabajo de 1080x1920 píxeles adecuado para la resolución de nuestro dispositivo. De cualquier modo, se ha creado un factor de escala con vistas a futuras modificaciones de tal manera que las variables estan referenciadas a una variable global ‘fact’ siendo posible un aumento o reducción de las imágenes de una forma rápida y eficaz.

Como se comentó con anterioridad, el diseño ha seguido la línea de programación del microcontrolador, es por ello por lo que se cuenta con los mismos estados. El código de fuente completo se puede encontrar en el CD que se incluye con el Trabajo de Fin de Grado. En éste, se podrá observar con más detalle aspectos como la gestión de la botonería encargada del control del programa, así como la gestión de imágenes a nivel de pixel entre otros.

CAPÍTULO 6. *Conclusiones y Presupuesto.*

6. Conclusiones y presupuesto

6.1. Presupuesto del proyecto

Los materiales utilizados en este proyecto se han presupuestado en base a la web de distribuidores de componentes electrónicos RS España y Farnell España. Ambas web ofrecen todo el material necesario para el desarrollo del trabajo.

Una vez barajadas las posibles opciones de compra, se ha elegido la opción más ajustada en cada caso.

6.1.1. Precios Unitarios

Código	Descripción	Unidad	Precio €
P01	Led Indicativo 826-379 (RS)	Ud.	0.26
P02	Resistencia 332 Ω 487-5347 (RS)	Ud.	0.696
P03	Resistencia 10 kΩ 148-736 (RS)	Ud.	0.043
P04	Resistencia 6.7 kΩ 754-6853 (RS)	Ud.	0.388
P05	Condensador 22 nF 653-0131 (RS)	Ud.	0.206
P06	Condensador 12 pF 831-2875 (RS)	Ud.	0.5
P07	Cristal de Cuarzo de 8 MHz 226-1718 (RS)	Ud.	1.11
P08	Cristal de Cuarzo de 32.768 kHz 547-6985 (RS)	Ud.	0.324
P09	Bateria de botón 3 V 597-201 (RS)	Ud.	2.04
P10	4xPila AAA Recargable 900mAh 812-442 (Farnell)	Ud.	4.185
P11	Reloj PCF8563 2064559 (Farnell)	Ud.	2.09
P12	Sensor Digital MCP9800 1439485 (Farnell)	Ud.	1.18
P13	Módulo Bluetooth HC-06 79484	Ud.	13.95
P14	Adaptador SD LC Studio	Ud.	5
P15	PIC 18F2455 623-0623 (RS)	Ud.	4.7
P16	Buffer Hexadecimal no inversor 4050BE 526-543 (RS)	Ud.	0.376
P17	Zócalo para pila de 3V 430-681 (RS)	Ud.	2.18
P18	Placa de baquelita virgen 100 x 160 mm 1267751 (Farnell)	Ud.	3.54
P19	Pulsador SPUN191000 (Farnell)	Ud.	0.534
MO01	Operario de Montaje	h	10
MO02	Soldador Cualificado	h	16
MQ01	Máquina Taladradora	h	3
MQ02	Insoladora	h	4

6.1.2. Precio Descompuesto

Ud. DATALOGGER V1.0

Ud. Datalogger con enlace Bluetooth a Aplicación Android

Código	Descripción	Unidad	Cantidad	Precio €	Subtotal
P01	Led Indicativo	Ud.	5	0.26	1.3
P01	Resistencia 332 Ω	Ud.	5	0.696	3.48
P02	Resistencia 10 kΩ	Ud.	5	0.043	0.215
P03	Resistencia 6.7 kΩ	Ud.	1	0.388	0.388
P04	Condensador 22 nF	Ud.	1	0.206	0.206
P05	Condensador 12 pF	Ud.	2	0.5	1
P06	Cristal de Cuarzo de 8 MHz	Ud.	1	1.11	1.11
P07	Cristal de Cuarzo de 32.768 kHz	Ud.	1	0.324	0.324
P08	Bateria de botón 3 V	Ud.	1	2.04	2.04
P09	4xPila AAA Recargable 900mAh	Ud.	1	4.185	4.185
P10	Reloj PCF8563	Ud.	1	2.09	2.09
P11	Sensor Digital MCP9800	Ud.	1	1.18	1.18
P12	Módulo Bluetooth HC-06	Ud.	1	13.95	13.95
P13	Adaptador SD LC Studio	Ud.	1	5	5
P14	PIC 18F2455	Ud.	1	4.7	4.7
P15	Buffer Hexadecimal no inversor 4050BE	Ud.	1	0.376	0.376
P16	Zócalo para pila de 3V	Ud.	1	2.18	2.18
P17	Placa de baquelita virgen 100 x 160 mm	Ud.	1	3.54	3.54
P18	Pulsador	Ud.	1	0.534	0.534
P19	Operario de Montaje	h	0.5	10	5
MO02	Soldador Cualificado	h	0.25	16	4
MQ01	Máquina Taladradora	h	0.2	5	1
MQ02	Insoladora	h	0.2	4	0.8
	Coste indirecto	%	3.00	2.35	1.76

Total……60.36 €

6.1.3. Resumen del Presupuesto

TOTAL DATALOGGER V 1.0	60.36 €
TOTAL DE PRESUPUESTO DE EJECUCION MATERIAL	**60.36 €**
16% GASTOS INDIRECTOS	**9.66 €**
6% BENEFICIO INDUSTRIAL	**3.62 €**
TOTAL (SIN I.V.A)	**73.63 €**
21% I.V.A	**15.46 €**
PRECIO (CON I.V.A)	**89.10 €**

Asciende el Presupuesto a la mencionada cantidad de OCHENTA Y NUEVE con DIEZ CÉNTIMOS.

6.2. Conclusiones y líneas futuras

Existen numerosos dataloggers comerciales como el que se ha desarrollado en este proyecto. Se ha extraído algunos ejemplos encontrados en distribuidores muy reconocidos como RS (Véase Anexo III).

Teniendo en cuenta las características de los mismos, se comparará por lo tanto características como el sensor empleado, la base de tiempos para el ciclo de medida, la precisión del sensor, la memoria de almacenamiento y la disponibilidad de software.

Datalogger con enlace a Bluetooth basado en PIC	Sensor: Temperatura
	Ciclo de medida: 1min hasta 1 mes
	Rango de Tª: -10º C hasta +85ºC
	Precisión: ±1ºC (Máximo)
	Memoria: Tarjeta SD de hasta 4 GB
	Software: Sí

Tabla 5. Características de Datalogger basado en PIC.

Tal y como podemos observar, nuestro sistema de adquisición de datos posee unas características muy parecidas a la de los sistemas comerciales. Si comparamos, podemos observar que aunque la precisión y el rango de operación de los sensores de temperatura que ofertan los dataloggers comerciales pueden ser algo mas elevados, nuestro datalogger esta muy cerca de lo que éstos nos ofrecen.

Los ciclos de medida son ajustables, estableciendo la base de tiempos mínima en 1 minuto. Será programable hasta un tiempo superior a un mes, sin embargo se ha establecido 1 mes como límite. Este valor es muy superior a lo que el mercado nos ofrece, además este ciclo podría ampliarse en un futuro con el desarrollo de una nueva versión que establezca el segundo como base de tiempos (tal y como se explicará en el apartado de líneas futuras).

El el almacenamiento de datos se lleva a cabo mediante una una tarjeta de almacenamiento de hasta 4GB, cantidad suficiente para albergar miles de registros. Se ha establecido el límite de 4GB ya que el testeo se ha llevado a cabo con una tarjeta SD, sin embargo la programación permite, según el compilador, el uso de SDHC. Este hecho haría posible un almacenamiento mucho mayor si fuera necesario.

Así pues, contaremos con un sistema de adquisición de datos muy versatil, ya que a parte de medir datos de temperatura, tambien es capaz de medir una gran cantidad de magnitudes físicas añadiendo los sensores correspondientes. Dado que el mercado actual es cada vez más competitivo y los dataloggers actuales incluyen en su mayoría un software para la gestión del hardware, se ha desarrollado de igual manera un software que permite la configuración de parametros así como la toma de datos en un momento preciso.

Por lo tanto, si se necesita realizar mediciones con una precisión no demasiado exigente con unas opciones simulares a las que se ofertan en el mercado, nuestro datalogger basado en PIC cumple estas necesidades.

Como sugerencia para **futuras líneas de trabajo** se propone:

- Una programación para aumentar el ciclo de medida de nuestro Datalogger actual, de manera que mediante el uso de timers internos o bien con la salida de reloj de la RTC, permita establecer una base de tiempos para el registro en intevalos de hasta 1 segundo.
- Una ampliación para el uso de USB como método opcional de almacenamiento de datos.

BIBLIOGRAFÍA.

BIBLIOGRAFÍA

(s.f.). Recuperado el 25 de Julio de 2015, de Labcenter Website: http://www.labcenter.com/index.cfm

Arduino Website. (s.f.). Recuperado el 24 de Julio de 2015, de https://www.arduino.cc/en/Guide/Introduction

Breijo, E. G. (2009). *Compilador C CCS y simulador Proteus para microcontroladores PIC 2ª Edición.* Marcombo.

Coquet, E. (s.f.). *Comunidad de electrónicos*. Obtenido de http://www.comunidadelectronicos.com/articulos/i2c.htm

Eterlogic Website. (s.f.). Recuperado el 25 de Julio de 2015, de http://www.eterlogic.com/Products.VSPE.html

Heath, S. (2003). *Embedded systems design. EDN series for design engineers (en inglés) (2 edición).* Newnes.

Ketai. (s.f.). *Ketai.org*. Recuperado el 20 de Agosto de 2015, de http://ketai.org/reference/netbluetooth/ketai-bluetooth/

Martín, F. J. (s.f.). *Universidad de Málaga*. Recuperado el 20 de Agosto de 2015, de http://www.el.uma.es/marin/Practica4_UART.pdf

Microchip Website. (s.f.). Recuperado el 25 de Julio de 2015, de http://www.microchip.com/Developmenttools/ProductDetails.aspx?PartNO=PG164130

Raspberry Pi Website. (s.f.). Recuperado el 24 de Julio de 2015, de https://www.raspberrypi.org/help/faqs/#introWhatIs

UNED.Departamento de Ingeniería Eléctrica, E. y. (s.f.). MASTER DEGREE: Industrial Systems Engineering.APARTADO 1.

UPV, E. P. (s.f.). *Comparativa de microcontroladores actuales.* Recuperado el 25 de Agosto de 2015, de http://server-die.alc.upv.es/asignaturas/lsed/2002-03/Micros/downloads/trabajo.pdf

ANEXOS.

ANEXO I. *Planos.*

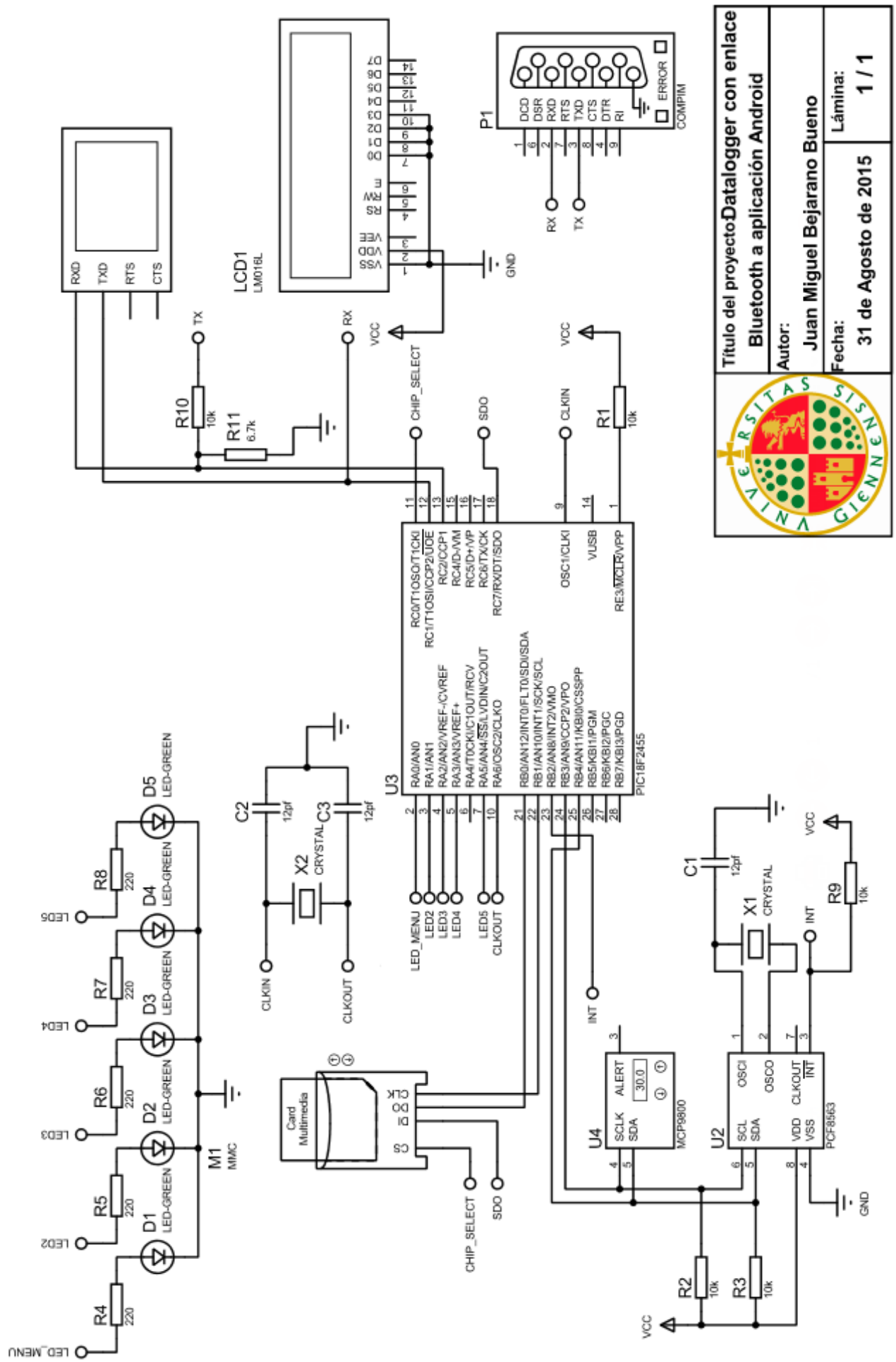
Anexo I. Planos.

1. Esquema con Proteus

A continuación se expodrá el esquema realizado con Proteus ISIS dónde se detallará el circuito.

Notas:

- Se incluirá además una adaptación de nivel para la tarjeta SD detallado en el apartado 2 de este mismo Anexo.
- Se ha incluido una pantalla LCD como línea de futuro por si se realiza con un microcontrolador como por ejemplo el 4550 que dispone de un puerto D.


DATALOGGER CON ENLACE BLUETOOTH A APLICACION ANDROID
Título del proyectoDatalogger con enlace Bluetooth a aplicación Android
Autor: Juan Miguel Bejarano Bueno
Fecha: 31 de Agosto de 2015
Lámina: 1 / 1
LCD1
LM016L
P1
COMPIM
U3
PIC18F2455
U4
MCP9800
U2
PCF8563
M1
MMC
X1
CRYSTAL
X2
CRYSTAL
C1
12pf
C2
12pf
C3
12pf
R1
10k
R2
10k
R3
10k
R4
220
R5
220
R6
220
R7
220
R8
220
R9
10k
R10
10k
R11
6.7k
D1
LED-GREEN
D2
LED-GREEN
D3
LED-GREEN
D4
LED-GREEN
D5
LED-GREEN
VCC
GND
TX
RX
CHIP_SELECT
SDO
CLKIN
CLKOUT
INT
LED_MENU
LED2
LED3
LED4
LED5

2. Adaptación de nivel de tarjeta SD

Para la simulación en Proteus ISIS es necesario conectar los pines del microcontrolador directamente al lector SD. Esto no debería realizarse en el montaje físico, ya que los voltajes que proporcionan los pines de salida del microcontrolador podrían dañar seriamente la tarjeta SD. Es por ello por lo que se indica el esquema de montaje a continuación empleando un buffer hexadecimal no inversor.

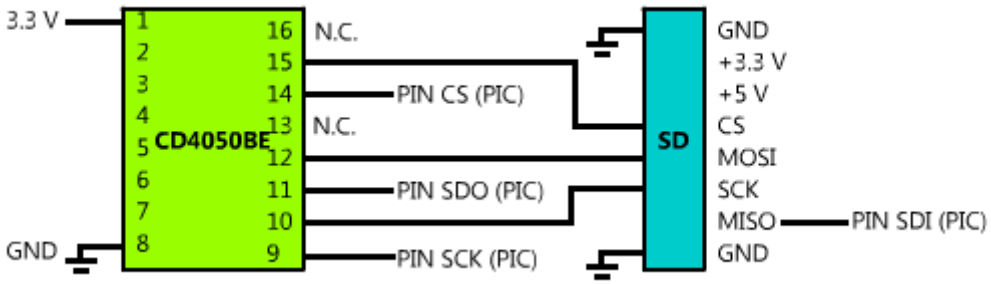


Figura 1.Adaptación de nivel para SD.

ANEXO II. *Manual de Usuario.*

CONTENIDO

ÍNDICE DE IMÁGENES

1. Introducción.

En el siguiente manual se expondrá al usuario los pasos a seguir para el correcto funcionamiento y puesta en marcha del datalogger.

2. Preparación del Datalogger.

En primer lugar, en el nuestro hardware pulsaremos el botón RESET (Alimentación) para asegurarnos que el programa se inicializa correctamente. Con esto borraremos la posible configuración que pudiera tener con anterioridad.

3. Funcionamiento del Programa.

Inicializaremos el programa "Datalogger_BT" desde el menú del dispositivo Android. Una ventana emergente pedirá la activación del Bluetooth en el caso de que no se tenga activado previamente. Una vez aceptado, una nueva ventana emergente dentro del programa nos mostrará los distintos dispositivos a los que se puede conectar nuestro dispositivo Android. (Figura 1.)

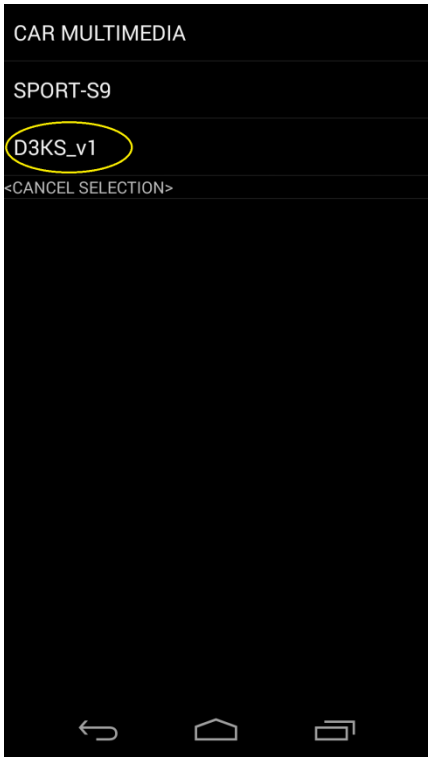


Figura 1. Selección de Dispositivo.

El nombre de nuestro dispositivo **(D3KS_v1)** deberá aparecer en esta lista.[1]

Una vez seleccionado el dispositivo "D3KS_v1", el LED del sistema que parpadea debe cambiar a un estado estable de encendido.

Llegados a este punto aparecerá una pantalla inicial con el nombre de la aplicación, deberemos pulsar en el logo verde tal como aparece en la Figura 2.

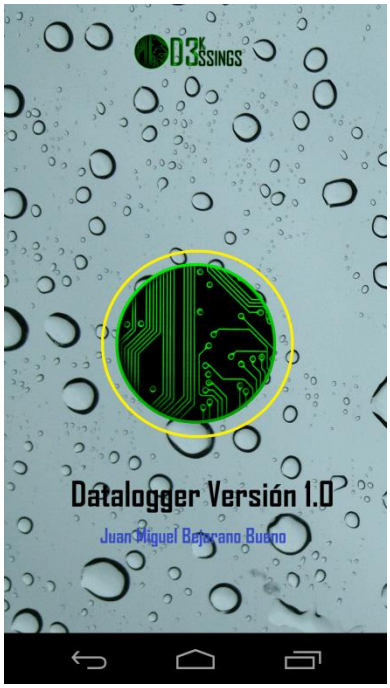


Figura 2. Inicio del Programa.

[1] Es importante que antes se haya vinculado desde el dispositivo Android.

Aparecerá en este punto el Menú Principal. (Figura 3.)

Figura 3. Menú Principal.

1. **Configuración Reloj/Calendario**: Este botón brinda la posibilidad de establecer la fecha y poner en hora el sistema.
2. **Configuración de Alarma**: Hace posible la programación del Datalogger para la toma de datos cada cierto tiempo.
3. **LOGGING / Lectura en tiempo real**: Llegado a este punto se nos ofrecen dos opciones:

    - Entrada en **modo bajo consumo** del Datalogger, que registrará los datos según su programación.
    - **Lecturas puntuales** de forma asíncrona y en tiempo real que serán grabadas en un archivo de texto guardado en el la raíz del dispositivo Android.

Situándonos en el menú principal, se tendrá que seguir por orden (muy importante que se realice por orden, ya que si no es así, podría no funcionar) los siguientes pasos:

Pulsar en "**Configuración de Reloj/Calendario**" (Punto 1, Figura 3). Una vez abierto el Menú de configuración pasaremos a la puesta en hora de la RTC mediante el uso de las flechas verdes.

Figura 4. Configuración de Reloj/Calendario.

Teniendo la fecha y hora deseada, se ha de pulsar el botón "**SET**" para poner la RTC en hora y el botón "**RESET**" si se quieren resetear los parámetros de la misma.[2] Una vez realizada la puesta en hora se ha de volver al menú principal pulsando "**Salir**".

[2] Es conveniente asegurarse que se ha pulsado correctamente para que la puesta en hora sea correcta.

Una vez en el Menú Principal. Pulsar en "**Configuración de Alarmas**" (Punto 2, Figura 3). La configuración de alarmas se realizará de una forma similar a la puesta en hora.

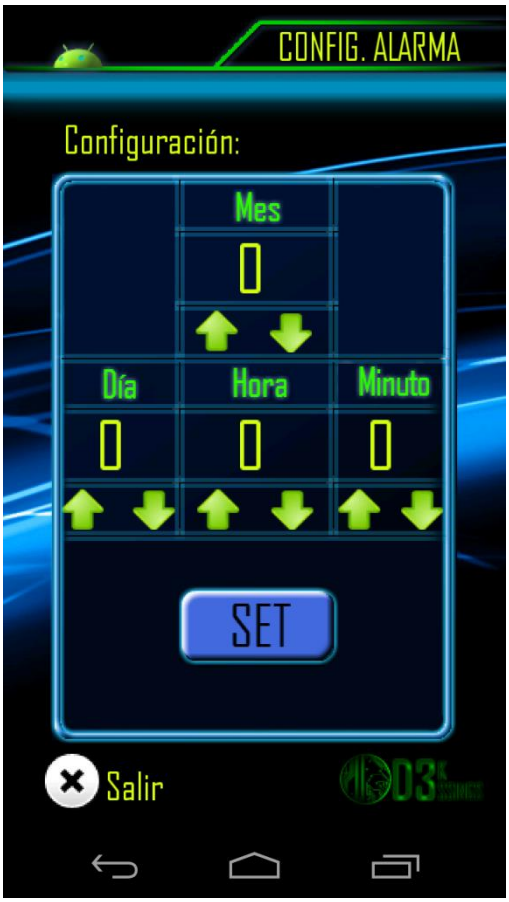


Figura 5. Configuración de Alarma.

Seleccionado el tiempo deseado para la toma de datos cada cierto tiempo, se pulsará el botón "**SET**" para enviar dichos datos al datalogger. Pulsar "**Salir**" para salir de este menú.

Llegados a este punto, se ofrece al usuario dos posibilidades:

- Si se desea poner el datalogger en modo de bajo consumo y dejar que este tome datos según la programación, pulsar el botón "**LOGGING**".

Figura 6. Datalogger en modo bajo consumo.

En caso de querer realizar una modificación de la configuración se deberá de proceder de la siguiente manera:

1. Pulsar el botón de INT del Datalogger.
2. Entrar en el menú "**LOGGING**" si no se está dentro y pulsar "**STOP LOG**".
3. Pulsamos "**Salir**" para volver al menú principal y realizar las modificaciones pertinentes en configuración siguiendo los pasos antes descritos.

- Si se desea realizar una Lectura puntual en tiempo real, pulsar "**Lectura en Tiempo Real**".

Figura 7. Modo Lectura en tiempo real.

Las lecturas puntuales serán realizadas cada vez que se pulse el botón "**Leer Ahora**" y serán guardadas en un archivo llamado "**Log.txt**" en la raíz del dispositivo.[3]

Nota: Este programa ha sido probado en un terminal LG NEXUS 5 con versión de Android 4.4.4.

[3] Es necesario dejar creado un archivo "Log.txt" con dicho nombre manualmente en la raíz.

ANEXO III. *Características de dataloggers comerciales.*

Este anexo incluye varios ejemplos de características de dataloggers en la actualidad a partir de una referencia muy reconocida como es RS España.

Rotronic Instruments TL-1D	Sensor: Temperatura y Humedad
	Ciclo de medida: 30 seg hasta 24h
	Rango de Tª: -20º C hasta +70ºC
	Precisión: ±0.3ºC
	Memoria: 32000
	Software: Sí
Corintech WIFI-TH	Sensor: Temperatura y Humedad
	Ciclo de medida: 10 seg hasta 12h
	Rango de Tª: -20º C hasta +60ºC
	Precisión: ±0.3ºC (Máximo)
	Memoria: 250000 Mediciones
	Software: Sí
Testo 184 T	Sensor: Temperatura
	Ciclo de medida: 1min hasta 24h
	Rango de Tª: -35º C hasta +70ºC
	Precisión: ±0.5ºC (Máximo)
	Memoria: 16.000
	Software: Sí

Tabla 1. Tabla de características de dataloggers comerciales