UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DISEÑO DE UN SISTEMA DE CAPTURA Y PROCESAMIENTO DE SEÑALES

AUTOR: SERGIO VILLANUEVA MARTINEZ

TUTOR: ANGEL RODAS JORDA

COTUTOR:

Curso Académico: 2014-15

Diseño de un sistema de captura y procesamiento de señales

RESUMEN

En el presente trabajo se llevará a cabo el diseño de un sistema de adquisición de datos mediante un microcontrolador Arduino. Se efectuarán dos diseños mediante los software Matlab y Labview compatibles entre sí. Ambos programas ofrecen control de entradas y salidas además de herramientas de filtrado de señales analógicas, con opción de guardado en ficheros de las señales para procesamientos posteriores. Concretamente en Labview será posible devolver la señal procesada por un canal a tiempo real. Además se estudiarán las posibilidades del software Processing como alternativa de programación a los diseños efectuados previamente.

Para ello se dispondrá de una serie de sensores analógicos y digitales, algunos de los cuales necesitarán un calibrado previo. Dichos sensores podrán ser utilizados en cualquier programa diseñado durante el proyecto para corroborar el buen funcionamiento de éstos y ser capaces de interactuar directamente con el entorno.

A continuación se buscará ampliar el alcance del proyecto mejorando la conectividad entre Arduino y el computador u otros dispositivos. Para ello se emplearán un módulo de Bluetooth y tarjeta Ethernet que podrán ser conectados sobre el microcontrolador, logrando finalmente establecer una conexión inalámbrica con otras máquinas.

Diseño de un sistema de captura y procesamiento de señales

ÍNDICE

1- INTRODUCCIÓN

1.1- OBJETIVO

El objetivo principal de este trabajo consiste en la programación de varias aplicaciones por ordenador que permitan muestrear señales a través de ciertos sensores mediante un microcontrolador Arduino. Una vez adquiridas, estas señales podrán ser procesadas en la aplicación diseñada. Se entiende por procesamiento de señales el realizar ciertas operaciones matemáticas sobre uno o varios canales con el objetivo de lograr una señal más nítida y de mayor utilidad que la inicial. Entre estas operaciones destacan los filtrados, amplificación y atenuación de señales y composición de dos señales procedentes de diferentes canales, además de cifrado y compresión de la información.

Por otra parte, no debemos olvidarnos de la importancia de la actuación y control sobre el entorno industrial, por lo que la aplicación diseñada dispondrá de unos métodos de actuación simples mediante señales eléctricas analógicas y digitales.

Para cumplir este objetivo ha sido necesaria en primera instancia la conexión Serial por cable USB entre el ordenador y microcontrolador. Debido a esto y a unas longitudes limitadas del cable se ha decidido que es necesario mejorar la forma de conexión entre dispositivos. La solución será emplear un módulo Bluetooth y una tarjeta Ethernet que permitirá conectar dos o más dispositivos de forma inalámbrica. De esta forma, cualquier aplicación buscada integrando estos módulos hará que todos nuestros diseños tengan una instalación más dinámica.

En resumen, los cableados y conexiones básicas a lo largo del desarrollo del proyecto son los mostrados en la siguiente figura. Destacar que las líneas discontinuas representan conexiones inalámbricas entre dispositivos. Se explicará cada conexión con mayor detalle en su correspondiente momento, pero por ahora es suficiente para dar una idea del alcance de este proyecto.

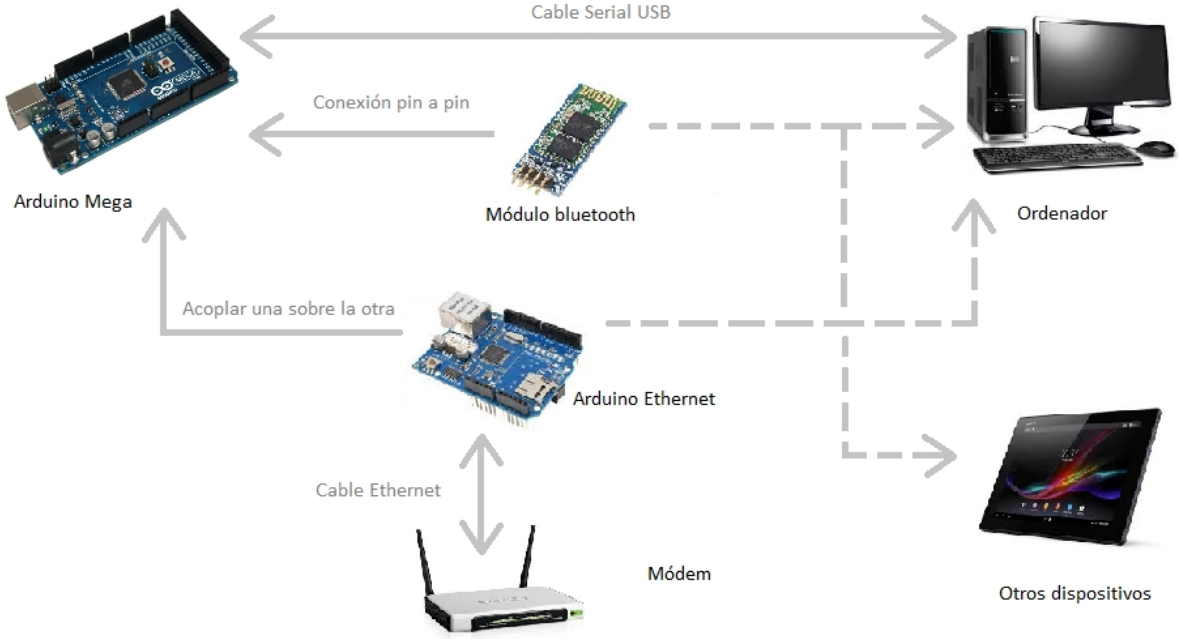


Figura 1: Conexiones básicas del proyecto.

1.2- MATERIALES Y MÉTODOS

Para efectuar este proyecto se han adquirido ciertos componentes electrónicos necesarios para la realización de los ensayos, así como software informático empleado para diseñar los programas. A continuación se presentará cada uno de los materiales utilizados, destacando alguna de sus características más relevantes.

1.2.1- Componentes básicos de Arduino

Muchos de los materiales de este apartado ya han sido brevemente introducidos, pero ahora deben ser presentados formalmente. En primer lugar, el microcontrolador que se empleará durante todo el proyecto es el Arduino MEGA 2560. Entre todos los microcontroladores Arduino se ha escogido éste ya que dispone de más memoria de programación, mayor memoria RAM y mayor número de pines tanto analógicos como digitales, además de una entrada para fuente de alimentación externa. La conectaremos al ordenador a través de un cable USB.

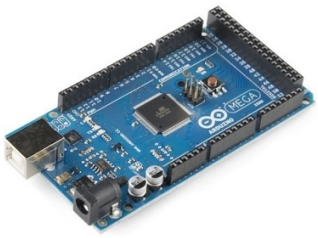


Figura 2: Modelo de ATMEGA 2560.

En segundo lugar Tenemos el Arduino Ethernet Shield REV3 con POE, tarjeta que permite la conexión por cable Ethernet al módem o al propio ordenador. Se ha elegido este modelo por el hecho de ser compatible con Arduino MEGA. Una de las características más destacadas es la posibilidad de conectar una tarjeta SD en ella y el poder manipular ficheros en ella. Además, incorpora la tecnología POE (Power over Ethernet), que permite alimentar eléctricamente ambos módulos a través del cable Ethernet.

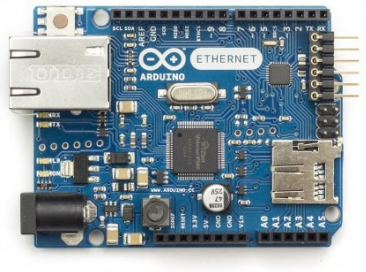


Figura 3: Arduino Ethernet Shield REV3 con POE.

El siguiente componente es el módulo Bluetooth. Los dos módulos más empleados son el HC-06 y el HC-05. Se ha escogido trabajar con el HC-06 ya que, aunque el hardware es el mismo, este módulo dispone de 4 pines, lo que facilita el empleo frente a su homólogo de 6 pines. De estos 4 pines, 2 se utilizarán para la lectura y transmisión de datos, mientras que los otros dos alimentarán de tensión a nuestro dispositivo.

Figura 4: Vista anterior y posterior del módulo HC-06

1.2.2- Componentes electrónicos

Los siguientes componentes son una parte importante del proyecto, pues gracias a ellos podemos realizar el montaje de los circuitos. Entre ellos se encuentras los sensores a utilizar, como también controles, indicadores y otros componentes como cables y resistencias que permiten conectar fácilmente todas las partes del circuito con Arduino.

Todos los montajes serán realizados sobre una protoboard o placa de pruebas, que facilitará en montaje de las conexiones entre componentes, así como su desmonte.

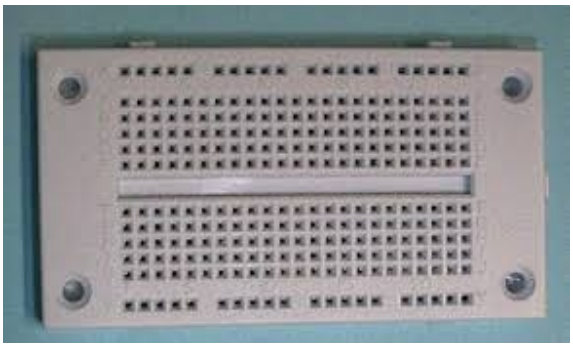


Figura 5: Protoboard empleada.

Entre los componentes de uso general que se dispondrán en el proyecto destacan:

- Juego de resistencias de los siguientes valores:
  - 2 x 120 Ω
  - 1 x 1 KΩ
  - 1 x 4.7 KΩ
  - 3 x 22 KΩ
  - 1 x 39 KΩ
- Potenciómetro o resistencia variable, cuyo valor máximo es 22 kΩ.
- Indicadores LED de colores rojo, verde y amarillo. Tensión de trabajo 2V y corriente entre 20-50 mA.
- 22 cables eléctricos de 18 cm de longitud.
- Final de carrera, empleado como actuador digital.
- Fotorresistencia LDR, sensor de luz.
- Termistor NTC, sensor de temperatura.
- Sensor digital de temperatura/humedad DHT11.

En relación a estos tres últimos sensores, tanto la LDR como la NTC han necesitado un calibrado previo para conocer las ecuaciones y curvas que rigen su comportamiento, mientras que el DHT11 nos ofrece las medidas directamente a través de una serie de pulsos digitales cada cierto periodo de tiempo. Se detallará más información sobre ellos en su correspondiente apartado de la memoria.

1.2.3- Hardware y Software informático

El software informático es la herramienta básica de programación para este proyecto, por eso se dedicará cierto tiempo a su aprendizaje para finalmente poder diseñar la aplicación de procesamiento y captura de señales. Para ello, se utilizará en primer lugar el Arduino IDE, entorno de programación que nos permitirá cargar los programas en el microcontrolador. Como complemento a éste , se utilizará Processing un lenguaje de programación basado en Java cuya interfaz es idéntica a la de Arduino IDE. Processing se podrá comunicar con el microcontrolador a través de un puerto serie (ya sea por cable USB o a través de Bluetooth) mientras que el usuario controla el programa mediante una interfaz gráfica muy completa fácilmente diseñable.

A continuación se empleará dos programas de mayor uso en el ámbito ingenieril que son Matlab y Labview. El objetivo con ambos será similar: diseñar una aplicación completa de captura y procesamiento. Para ello, se cargará en el microcontrolador un programa ofrecido por los fabricantes. (también conocidos como Firmatas) que simplifica la comunicación con estas aplicaciones. Se dedicarán varios apartados a estos programas más adelante.

Además, se recurrirá a un gestor de dispositivos de Bluetooth para controlar las comunicaciones por Bluetooth desde el PC. Para ello se empleará la aplicación TOSHIBA Bluetooth Stack 8.00.12, que nos permitirá abrir un puerto serie virtual para establecer conexión inalámbrica con Arduino. Para ello será necesario un adaptador USB Bluetooth conectado al PC, es nuestro caso será el LM054 de LM Technologies.

Finalmente se probará a conectar el Arduino con un dispositivo Android, concretamente una Tablet. A través de la aplicación de Android Bluetooth Terminal, podremos recibir datos de Arduino en forma de líneas de texto, así como controlar el microcontrolador mediante líneas de comandos que deberán ser programadas.

1.3. PLANIFICACIÓN Y TOMA DE DECISIONES EN EL PROYECTO

Al inicio del proyecto se planteó que se emplearían principalmente tres tipos de software de programación entre los descritos en el punto anterior, que son Processing, Matlab, y Labview, aparte de Arduino IDE, indispensable para la carga de programas en el microcontrolador. Tras haber tomado contacto con los materiales y el Arduino Mega, puesto que es la primera vez que trataba con este microcontrolador, se decidió comenzar con el desarrollo de la aplicación de captura y procesamiento en Matlab. A continuación, se pasó a programar una aplicación con Labview, con la intención de que no sólo ofreciera las mismas herramientas ya diseñadas en Matlab, sino que además fuera una mejora con más opciones de procesamiento.

Hecho esto, se decidió cambiar el plan inicial, dedicando un menor esfuerzo a la parte de programación en Processing, con el que se realizaron aplicaciones de captura más simples que las anteriores para finalmente ser presentado como una alternativa con la que mejorar el conjunto del proyecto en un futuro.

Llegado a este punto, se disponía de suficiente tiempo como para ampliar el proyecto en dos direcciones: La adquisición de más sensores que un simple potenciómetro para probar las aplicaciones diseñadas, enunciados en el punto anterior, y la mejora de la conectividad mediante tecnología inalámbrica, es decir, utilizando el módulo Bluetooth y la tarjeta Ethernet adquiridos. El estudio de este material y su inclusión en el contexto del proyecto ocupó el resto de tiempo de trabajo, por tanto, aquí concluye la fase de elaboración del proyecto y se inicia la redacción de esta memoria.

1.4- MOTIVACIÓN Y ASIGNATURAS RELACIONADAS DIRECTAMENTE CON EL PROYECTO

Mis motivaciones personales a la hora de elección un proyecto fueron mi pasión por la electrónica y programación y las ganas de aprender desde cero cómo utilizar y programar un microcontrolador Arduino. Todo esto, unido al interés del proyecto sobre estos temas hizo que decidiera escogerlo entre las otras ofertas.

Por otra parte, no hay que olvidar la importancia que tienen los las señales eléctricas en muchos ámbitos, tanto en la vida cotidiana como en la industria o en la investigación. Las señales de audio de micrófonos y altavoces, el control de automatismos en la industria o instrumentación sanitaria como los electrocardiogramas son algunos de los ejemplos de estas señales Es por esto que el hecho de poder adquirirlas, procesarlas y devolverlas al entorno para ser usadas adquiere gran importancia.

Además, este proyecto pone a prueba los conocimientos aprendidos en varias asignaturas. Las nociones de programación que se dieron en Informática han sido de gran ayuda a lo largo de todo el trabajo. Los conceptos de entrada y salida impartidos en Tecnología Informática Industrial han sido fundamentales para controlar el flujo de comunicaciones entre dispositivos sin que interfieran unos con otros, así como la conversión analógica-digital propia del microcontrolador para poder interpretar sus mediciones.

Respecto al filtrado de señales, han sido fundamentales los conocimientos teóricos de Sistemas Electrónicos, aunque concretamente para el filtrado digital de señales se han tomado como referencia los distintos tipos de filtros diseñados en la optativa Laboratorio de Automatización y Control. Por último, en relación a la instrumentación de sensores han sido muy importantes los conceptos teóricos y prácticos impartidos en Tecnología Electrónica para acondicionar las señales de nuestros sensores a las entradas analógicas del microcontrolador.

Una vez presentadas mis motivaciones personales como estudiante y los conocimientos aprendidos durante el grado que han tenido una aplicación directa sobre este estudio, estamos en disposición de comenzar el desarrollo del proyecto.

2- HERRAMIENTAS DE DESARROLLO

2.1- ESPECIFICACIONES Y GENERALIDADES DEL MICROCONTROLADOR

Vamos a presentar en detalle la herramienta esencial del proyecto, que es el microcontrolador Arduino MEGA 2560. Dispone de 16 canales analógicos preparados únicamente para recibir entradas de señales, así como 54 canales digitales de entrada y salida. Sin embargo, hay que destacar que algunos de estos pines tienen un uso reservado para otros fines, como es el caso de los pines digitales 0 y 1, destinados a la comunicación serie mediante cable USB. Si fuera necesario establecer una comunicación serie con más de un dispositivo, existen otras parejas de pines que pueden cumplir con este fin (los pines 18-19 forman el Serial1, el 16-17 el Serial2, y el 14-15 son el Serial3). Es necesario comentar que existe una librería en Arduino IDE, *Software Serial* que permite escoger 2 pines digitales cualquiera para realizar esta función.

Por otro lado, los pines digitales del 2 al 13, pueden ser usados como salidas de PWM, es decir ofrecer como salida un valor de tensión entre 0 y 5 V, mediante una señal que está activa a 5V durante un tiempo para luego volver a 0V, controlando el tiempo de que permanece la señal en estado alto durante un tiempo de ciclo determinado (de 0% a 100%).

En cuanto a la manera de alimentar eléctricamente a la placa, existen 3 métodos para poder hacerlo. El primero y más sencillo es la conexión a través del cable USB directamente al ordenador. El segundo es mediante una fuente de alimentación externa que es posible conectar desde la entrada adecuada (marcada con el número 5 en la figura 6 inferior). El último método es a través de un cable que esté a 5V conectado en el pin Vin (el primero de la izquierda del grupo 6 de la figura 6).

Una vez alimentado Arduino, tenemos unos pines especiales en el grupo 6 de la figura. Estos son dos pines de puesta a tierra, etiquetados como GND, además de otros dos pines de 5V y 3.3V, que pueden emplearse para alimentar circuitos exteriores. Nosotros los usaremos para esto último.

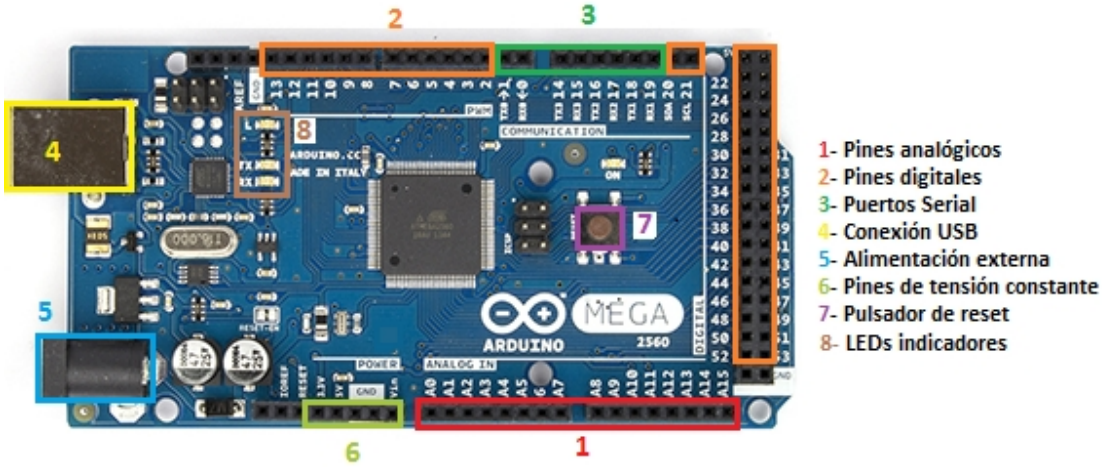


Figura 6: Detalle del Arduino MEGA 2560.

Cuando vayamos a conectarlo una fuente de alimentación externa, debemos tener en cuenta que este voltaje no supere los 20V ni sea inferir a 6V. El fabricante recomienda valores entre 7 -12 V. En este caso se empleará una fuente de alimentación de 9V cuando sea necesario.

La placa también tiene un botón de reset, que al pulsarlo hará que el programa que tenga cargado en su interior vuelva a comenzar desde la primera línea de código.

El microcontrolador, además, tiene tres leds indicadores. Los dos inferiores (TX y RX) corresponden a la información digital que viaja a través del puerto serie formado por los pines 0 y 1. El led superior (marcado como L) es un led de prueba que conecta con el pin digital 13, y puede usarse tanto para ver el valor que toma la entrada conectada a él, como para verificar que la salida ordenada por el programa cargado es realmente la deseada.

Cuando se decide emplear los pines digitales como salidas o entradas debe tenerse en cuenta que la corriente continua que circula por estos pines es de unos 40 mA, lo cual limita las aplicaciones a baja potencia y sería necesaria una etapa de amplificación si la aplicación requiriera potencias e intensidades mayores.

Por último, el Arduino MEGA 2560 tiene 256 KB de memoria flash en la que se guardan los programas cargados. También incluye una memoria SRAM de 8KB y una EEPROM de 4KB, la cual puede ser escrita y leída mediante funciones de la librería *EEPROM*.

2.2- PROGRAMACIÓN CON ARDUINO IDE Y PROCESSING

Es hora de programar el microcontrolador. Para ello conectaremos el Arduino MEGA al ordenador con un cable USB. Esta será la conexión estándar con la que trabajaremos durante la primera parte del desarrollo del proyecto.

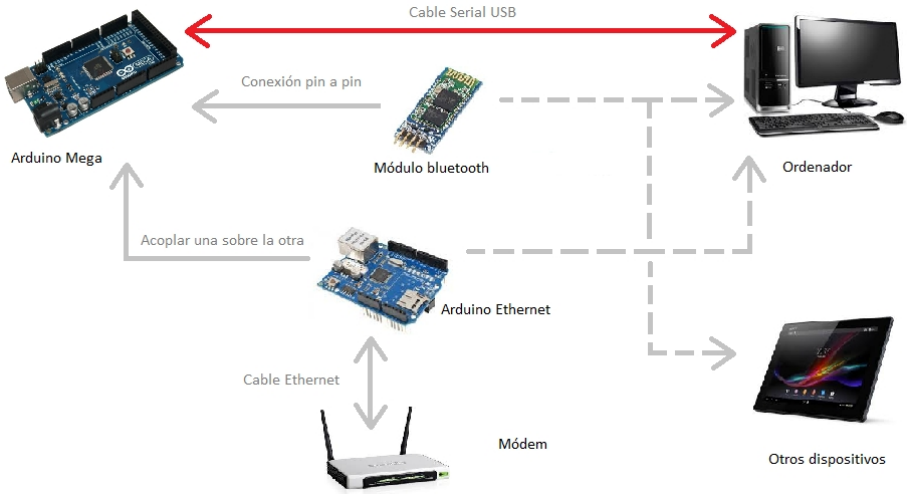


Figura 7: Conexión Arduino-PC mediante cable USB

El software que se usará para programar el microcontrolador es Arduino IDE, que está basado en el lenguaje C. Todo programa en Arduino IDE tiene dos partes claramente diferenciadas. La primera, el *void setup*, donde configuramos detalles como la velocidad de transmisión de la información (o baudios), la inicialización de los puertos serie que se vayan a emplear o el modo de empleo de un pin digital concreto (es decir, si dicho pin será empleado como entrada o salida de datos). El *void setup* solamente se ejecutará una vez al inicio del programa.

La segunda parte del programa, el *void loop*, es un bucle que ejecuta el código en su interior una y otra vez. En él se programan las instrucciones que queremos realizar periódicamente, como la lectura o escritura de datos sobre los pines digitales y analógicos deseados. Puesto que es un bucle, se debe controlar el periodo de ejecución del mismo con alguna instrucción. Esta es *delay (milis)* siendo milis el tiempo en milisegundos que el programa esperará hasta pasar a la siguiente línea (o volver a repetir el bucle si esa instrucción era la última línea del void *loop*).

Cabe mencionar que en las primeras líneas del programa se pueden declarar librerías y variables que serán usadas tanto en el *void setup* como en el *void loop*. Algunas de estas librerías que usaremos más adelante son *DHT.h* y *Ethernet.h*.

Para programar en Arduino es necesario conocer cuatro funciones de control que son muy empleadas en todos los programas:

1- *digitalWrite (pin, estado)*: esta función escribe en el pin digital deseado (cualquier número distinto de 0 y 1 presente en la placa) el estado binario indicado (*LOW, 0 Voltios o HIGH, 5 Voltios*).

2- *int dato = digitalRead (pin)*: procede a la lectura del pin digital elegido y devuelve en la variable dato un valor de 0 (voltaje del pin menor de 2.5V) o 1 (voltaje del pin mayor de 2.5V) dependiendo del estado de dicho pin.

3- i*nt dato = analogRead (pin*): lee el pin analógico deseado (número entre 0 y 15) y devuelve en dato su tensión de 0 a 5 voltios convertido en un valor entre 0-1023. Esto es debido a un convertidor analógico digital de 10 bits, que segmenta los 5 voltios en 2^{10} divisiones. Posteriormente, se podrá obtener la tensión digital con una función de conversión:

$$Tensión\ de\ entrada\ pin\ analógico\ (V) = \frac{dato\ x\ 5}{1023}$$

4- *analogWrite (pin, ciclo)*: esta función puede resultar confusa, pues aunque su nombre haga referencia a analógico, lo que realmente hace es generar sobre el pin digital deseado una señal PWM. Es decir escribirá un valor de tensión entre 0 y 5 V dependiendo del valor de *ciclo*, tomando ésta un valor entre (0-255). La tensión de salida seguirá por tanto la siguiente función de conversión:

$$Tensión\ de\ salida\ por\ pin\ digital\ (V) = \frac{ciclo\ x\ 5}{255}$$

Veamos un ejemplo de programa. En este ejemplo, se inicializa en primer lugar el puerto Serie por defecto a una tasa de velocidad de 9600 baudios. A continuación, se leerá periódicamente el canal analógico 1 (dato tomará valores entre 0-1023), cuyo valor dividido entre 4 será escrito en el puerto serie, puesto que en este puerto deben escribirse valores numéricos enteros entre 0-255 (más adelante veremos que también se pueden escribir y leer cadenas de caracteres), Esta medida podrá ser leída en el PC por la consola de Arduino u otros programas como Processing.

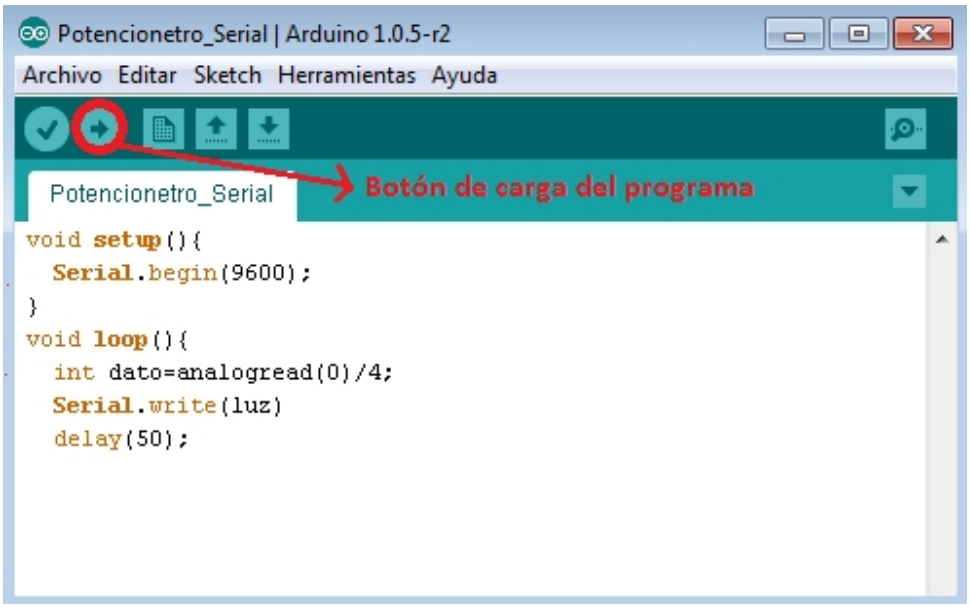


Figura 8: Ejemplo de programa en Arduino IDE

Una vez acabado el programa, es la hora de cargarlo sobre el microcontrolador. Para ello en el menú Herramientas >> Tarjeta, se elige la opción Arduino Mega 2560. A continuación, se elige el puerto serie en el que está conectado la placa en Herramientas >> Puerto Serial. En este proyecto se empleará el puerto "COM7". Finalmente, se pulsa sobre el botón Cargar marcado en la figura 8 superior.

Ahora que sabemos cómo cargar programas y enviar y recibir datos es hora de recibirlos o dar instrucciones con una buena interfaz gráfica. Para ello la primera opción será el empleo del software Processing.

Processing es un lenguaje de programación basado en Java que permite al usuario crear una ventana gráfica que contenga imágenes, movimientos e interacciones de los objetos con el teclado y diferentes movimientos del ratón partiendo de una única ventana en blanco. En este proyecto se empleará esta herramienta como método de visualización de datos o mecanismo de control de las acciones que el microcontrolador vaya a realizar.

Ya que Processing tiene numerosas librerías y hay muchos manuales que explican su programación, solamente se explicará las partes básicas que componen un programa de Processing, sin incidir en la explicación de las funciones que se han empleado. Las primeras líneas de código están reservadas para declarar las librerías y variables necesarias. Se utilizará la librería la librería *Processing.serial* para abrir la comunicación del puerto COM7 compartido con Arduino MEGA 2560.

A continuación se encuentra el *void setup*, que es, como en caso de Arduino IDE, una función que se ejecuta una sola vez al inicio del programa y sirve para inicializar el tamaño de la ventana de trabajo, el puerto serie o ficheros de escritura, entre otros.

Seguidamente, el *void draw* es el equivalente al *void loop* de Arduino IDE. Es un bucle de ejecución periódica que contiene las instrucciones de creación de la ventana gráfica. Para posicionar los objetos sobre ella, es necesario saber que la ventana posee un sistema de coordenadas bidimensional de tamaño personalizable en el *void setup*, siendo la coordenada (0,0) la esquina superior izquierda.

Por último, se pueden llamar a otras funciones especiales ya implementadas que permiten detectar cuando se pulsa cualquier botón del teclado, o detectar los clics del ratón, incluso el arrastre del ratón con el botón presionado. El programa interrumpirá el *void draw* para entrar en una de estas funciones si se cumple la condición de entrada a la función, por ejemplo pulsar la barra espaciadora del teclado o hacer clic derecho sobre una coordenada de la ventana.

Finalmente, existen 2 funciones realmente útiles que pueden emplearse en cualquier línea del programa, que devuelven un valor entero de las coordenadas del ratón sobre la ventana. Estas son *MouseX()* y *MouseY()*, y las emplearemos a menudo para crear botones y reconocerlos si se clica el ratón estando situado sobre ellos.

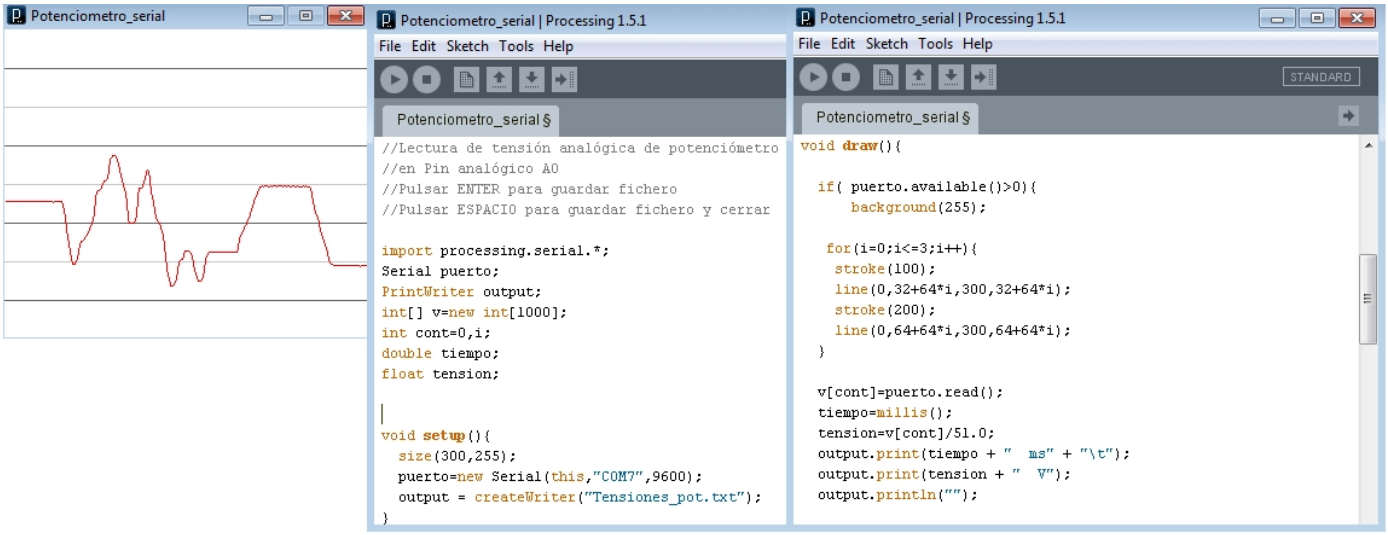


Figura 9: Ejemplo de programa en Processing. De izquierda a derecha: Ventana de trabajo, Void setup y void draw

El ejemplo mostrado en la Figura 9 lee en tiempo real el valor del puerto serie enviado por el microcontrolador (siempre que haya una nueva medida disponible, de lo contrario no se ejecutará ningún tratamiento posterior), la guarda en un vector *v* y grafica estos valores en tiempo real en la ventana de trabajo. Además permite guardar los datos en un fichero de texto con un formato de dos columnas, la primera será el tiempo de adquisición de la medida y la segunda será la propia medida. Comentar que esté programa sólo funcionará si sobre la placa Arduino se ha cargado el programa de la Figura 8.

Por último, hay dos métodos para comunicar Processing con Arduino. El primero es el del ejemplo anterior, que es cargando un programa de creación propia sobre el microcontrolador y ejecutando Processing al mismo tiempo. Con este método se debe tener precaución con el flujo de datos a través del puerto serie, pues ambos programas lo utilizan y habrá que tener un trato adecuado de interrupciones cuando tanto Processing como Arduino MEGA 2560 escriban y reciban datos simultáneamente para que no haya interferencias.

El segundo método es algo diferente a lo que hemos visto hasta ahora. La idea es cargar sobre el microcontrolador un programa existente de alta complejidad, llamado *StandardFirmata*, que está incluido en los ejemplos de Arduino IDE. De esta manera, se puede programar desde Processing con las funciones generales de control de pines (*analogRead, digitalWrite*) empleando la librería *cc.arduino*. De esta manera no tenemos control directo sobre la comunicación por el puerto Serie, pero simplifica enormemente la programación y permite realizar programas más complejos sin la preocupación de las interferencias en la comunicación. Por esto mismo, este método de programación por Firmatas será empleado seguidamente tanto con Matlab y Arduino.

2.3- CONEXIÓN CON MATLAB Y LABVIEW

Llegó la hora de integrar el microcontrolador con dos softwares informáticos frecuentemente empleados en ingeniería, como son Matlab y Labview. A diferencia del apartado anterior, éste no se centrará en explicar estas aplicaciones a nivel de programación, sino que será enfocado a la explicación de cómo lograr enlazar el Arduino MEGA 2560 con estos dos programas, qué herramientas vamos a emplear de cada uno de ellos y los detalles que se tendrán en cuenta cuando se desarrolle la aplicación final de procesamiento de señales. .

Matlab es un entorno de programación con un alto nivel de cálculo numérico que utiliza un lenguaje de programación característico, el lenguaje M. Además de la implementación de programas tradicionales y herramientas de simulación, dispone de opciones para la creación de Interfaces Gráficas de Usuario (de ahora en adelante llamadas GUI), con las que se puede crear interfaces con botones, gráficas e indicadores muy personalizables que responderán a las instrucciones de un programa principal.

Al igual que en Processing, existen dos formas de conectar Arduino con Matlab. El primero de ellos es abriendo por comandos, en un programa de extensión *.m* o en un script el puerto serie a emplear, *puerto_serial =serial('COM7')*, elegir la velocidad en baudios adecuada, *puerto_serial.baudRate = 9600* y tratar el puerto serie como si se tratara de un fichero de texto, con su correspondiente *fopen (puerto_serial)* para abrirlo y *fclose (puerto_serial)* para cerrarlo. Durante el enlace, se puede transmitir información con *fwrite (puerto_serial, dato)* o recibirla mediante *dato=fscanf (puerto_serial, '%d')*, siendo *dato* un número entero entre 0 y 255, teniendo cargado en el microcontrolador el programa adecuado.

La segunda de las opciones es cargar en Arduino la Firmata para Matlab, llamada *adiosrv* encontrada en una subcarpeta de ArduinoIO, fácilmente descargable de internet. A continuación se ejecutará un archivo .m desde Matlab llamado *install_arduino*, que también está incluido en subcarpetas de ArduinoIO. Finalmente, con la siguiente instrucción *Arduino=arduino ('COM7')*. De esta manera, podremos usar desde Matlab los comandos generales propios de Arduino IDE precedidos de la clase Arduino de la siguiente forma: *Arduino.analogWrite(pin, valor)* o *Arduino.digitalWrite(pin, valor)*.

Una vez elegido el método de comunicación que se va a emplear es hora de programar las instrucciones de acción sobre la placa. Se elegirá la creación de una GUI porque de esta manera se podrá controlar y visualizar el estado de los pines de una manera muy intuitiva, además de otras utilidades que veremos más adelante. Toda GUI está compuesta por dos archivos: una figura gráfica .fig que contiene la disposición de los objetos sobre una cuadrícula y un archivo .m que incluye las funciones de creación de dichos objetos y las funciones de llamada o Callbacks para cada objeto que se activarán cuando el usuario interactúe con el elemento.

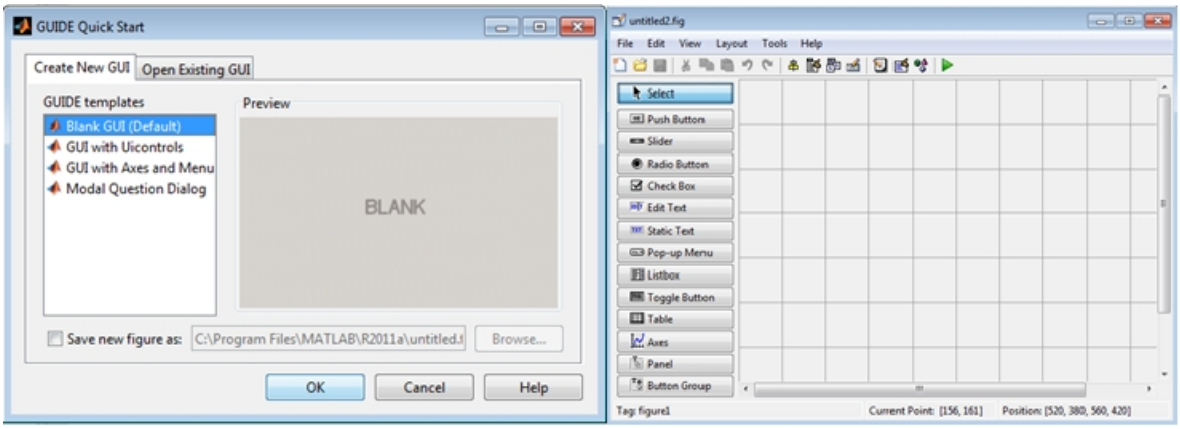


Figura 10: Ventana de inicio de creación de un GUI (izquierda) e Interfaz de diseño gráfico del GUI (derecha)

Puesto que la GUI a diseñar enviará y recibirá información sobre el estado de varios pines simultáneamente, se escogerá la comunicación por Firmata con Arduino para que las instrucciones de transmisión de datos estén todas reguladas por una única aplicación que será Matlab.

Una vez resuelta la comunicación entre Arduino y Matlab, se procede a explicar cómo conectar el microcontrolador con Labview. El único método posible de comunicación entre Labview y Arduino MEGA es mediante una Firmata presente en los paquetes descargables de Labview,. Para ello se descargará el *LIFA (Labview Interface for Arduino)* que se encuentra entre todas las extensiones de Labview en la aplicación *JKI VI Package Manager*, instalada con el software de Labview, como se puede ver en la figura 11.

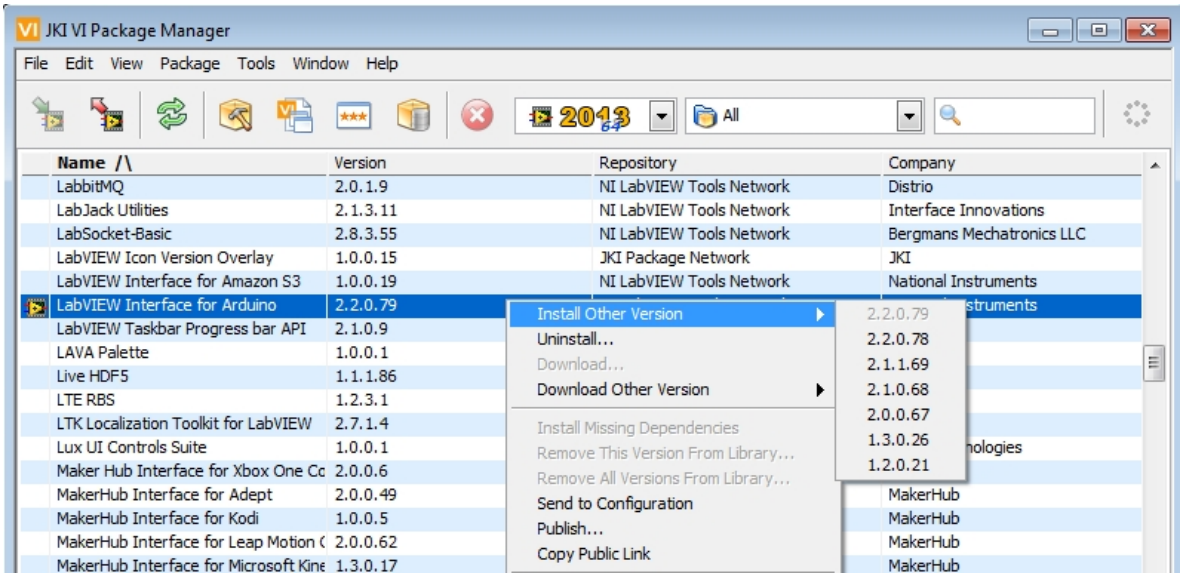


Figura 11: Descarga de LIFA (Labview Interface for Arduino)

Al descargar la última versión, tendremos disponible en el siguiente directorio *LabVIEW 2013\ vi.lib\LabVIEW Interface for Arduino\ Firmware* la Firmata que cargaremos sobre el microcontrolador con el nombre de *LIFA_Base*.

Llegado a este punto tenemos todas las herramientas preparadas para empezar a programar. Labview utiliza el lenguaje gráfico de programación G basado en la conexión de entidades con entradas y salidas, operaciones y bucles todos ellos cableados correctamente sobre un diagrama de bloques. A parte del diagrama de bloques, sobre el que se realiza toda la programación, está el panel frontal donde se ubicarán todos los controles e indicadores requeridos para la aplicación, como botones, leds, controles numéricos o gráficas.

Si se echa un vistazo a la paleta de funciones en el diagrama de bloques, aparece un último grupo nuevo llamado *Arduino*. En su interior están los objetos necesarios para poder dar instrucciones al microcontrolador. Además, al desplegar el subgrupo *Low level* se apreciarán las entidades homólogas a las funciones básicas de Arduino IDE de control de los pines. Podemos ver en detalle el resto de objetos de este grupo en la figura 12.

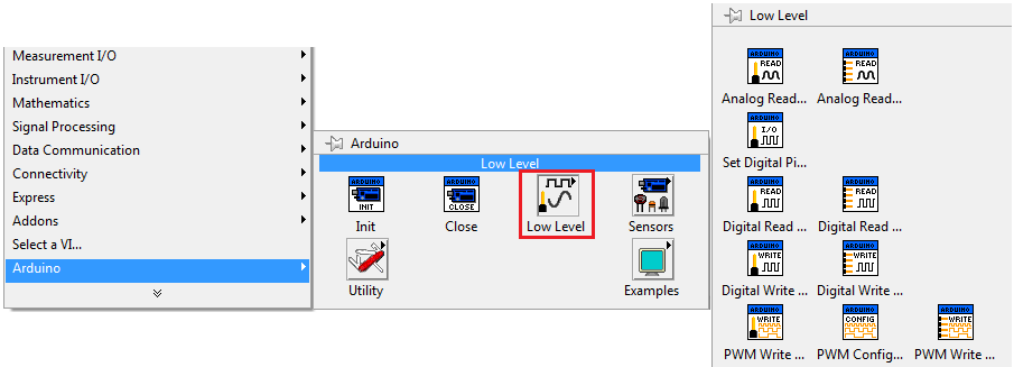


Figura 12: Nueva herramienta de Arduino en el diagrama de bloques

La conexión de estos objetos debe ser en serie, uno detrás de otro, con la información viajando de izquierda a derecha en el diagrama de bloques. En primer lugar se situará el objeto *Init*, que necesita ciertos parámetros de entrada para reconocer al microcontrolador, como son el puerto serie en el que está alojado, los velocidad de transmisión (cuyo valor debe estar en 115200 baudios, debido a los requerimientos de la Firmata LIFA_Base) y el modelo del microcontrolador, en este caso "Mega 2560".

A continuación vienen los elementos de control de pines del subgrupo *Low Level*. Se pueden conectar tantos como se desee en serie, ejecutándose las instrucciones paso a paso de izquierda a derecha. Cada uno de estos elementos ofrece unos parámetros propios de entrada y salida, como el AnalogRead de la figura 13, que lee el canal analógico 0 y envía su valor (número real entre 0 y 5 voltios) a un indicador analógico.

Finalmente, cerramos la conexión con el bloque *Close*. Además, todos estos objetos están conectados por dos cables por la parte superior e inferior. El superior, de color rosa, es el recurso de Arduino mientras que el inferior, verde, es el detector de errores en la comunicación del que se puede sacar un breve informe como salida del bloque *Close*.

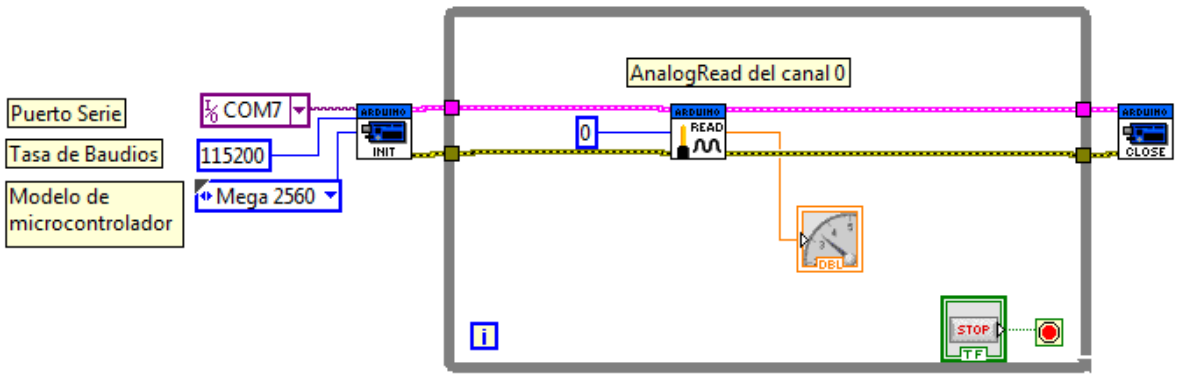


Figura 13: Ejemplo de programación en Labview

Finalmente, combinando estos objetos introducidos con unas nociones básicas de programación en Labview es posible desarrollar una aplicación de captura y procesamiento bastante completa. Éste será el objetivo final que se buscará lograr en Labview desarrollado en el apartado 4.4.

3- INSTRUMENTACION DE SENSORES Y RECOGIDA DE DATOS

Antes de presentar el núcleo central del proyecto, que son los programas de captura y procesamiento de señales, es necesario comprender cómo se comportan los sensores que se emplearán en ellos. Los sensores analógicos, que son el LDR y el NTC, requieren de una resistencia de acondicionamiento, que deberá ser elegida adecuadamente para cada uno de ellos. En cuanto al sensor digital DHT11, destacar que tienen un funcionamiento completamente distinto a los otros dos, cuyo funcionamiento será explicado una vez consultado su manual.

3.1- SENSOR LDR DE LUZ

El primer sensor que se ha empleado es un sensor de luz como es el LDR, cuyas siglas significan *Light Dependent Resistor* o Resistencia dependiente de la luz. Está formado por un material semiconductor que cuando más intensa es la luz que incide sobre él, hay un número mayor de electrones que reciben la suficiente energía para saltar a la banda de conducción de semiconductor. Esto genera huecos en la banda de valencia que permiten el paso de una corriente mayor, por tanto la resistencia del sensor disminuye.

En resumen, la luz y la resistencia del LDR siguen una relación inversa de tipo logarítmica. Puesto que en un principio no conocemos la expresión matemática que los relaciona, era necesario calibrar el sensor utilizando el montaje de la figura 15. Como el valor máximo resistivo de este LDR es 22 KΩ, se ha decidido conectar en serie con una resistencia de 4.7 KΩ y tomar la medida del punto de conexión entre ambas. El cable negro de la figura está conectado a tierra, 0 V y el cable rojo suministra una tensión de 5V.

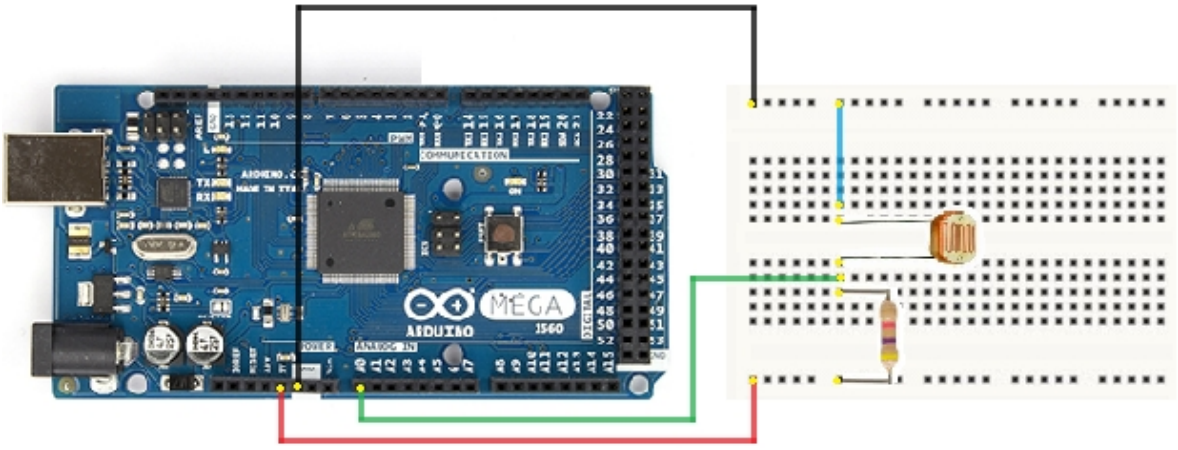


Figura 15: Montaje del sensor LDR

De esta manera, haciendo incidir diferentes focos de luz (luz ambiente, una lámpara o una linterna) sobre el sensor y conociendo los valores típicos de estas fuentes emisoras en luxes, hemos medido la tensión entre la LDR y la resistencia fija en el pin 0 de Arduino. Haciendo las conversiones adecuadas de medida analógica a tensión, y de tensión a resistencia se ha podido interpolar una curva de calibrado que cumple el comportamiento del sensor, cuya curva está representada en la figura 16 y sus expresiones matemáticas son:

$$Resistencia\_LDR(\Omega) = \frac{tensión\_pin_0\ (V)\ x\ 4700}{5\ V - \ tensión\_pin_0\ (V)}$$

$$Luz\ (Lux) = 181494\ x\ Resistencia\_LDR^{-0.77}$$

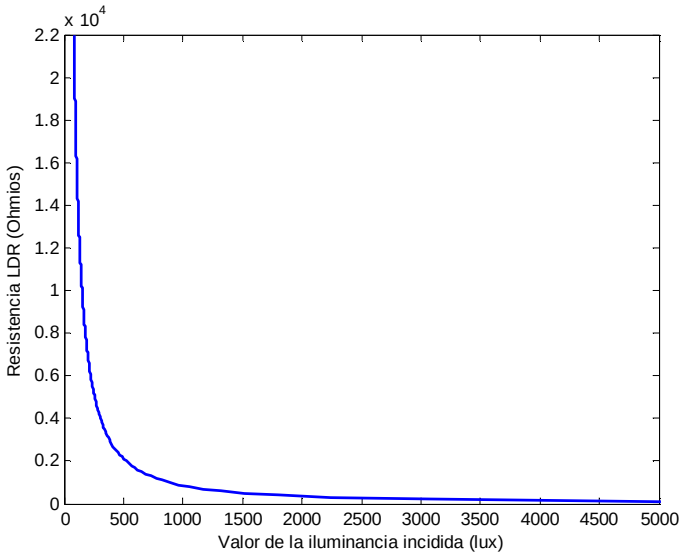


Figura 16: Curva de comportamiento del LDR (iluminancia frente a resistencia)

Con la LDR calibrada, se ha decidido crear un pequeño programa en Processing para verificar si la calibración realizada es válida o no. El resultado de esta prueba se puede ver en la figura 17, donde se puede afirmar que cualitativamente la relación obtenida sí es válida. Cuantitativamente, los datos de luz devueltos por Processing tienen un orden de magnitud coherente con los datos usados para la calibración.

Un último apunte de interés es el tiempo de respuesta de este sensor ante un cambio brusco de luz marcados por las rectar verticales marrones de la figura 16. Se puede apreciar que es un tiempo de respuesta bastante rápido, cuyos valores oscilan entre 500ms y 150ms desde el cambio de fuente luminosa hasta que la señal se estabiliza.

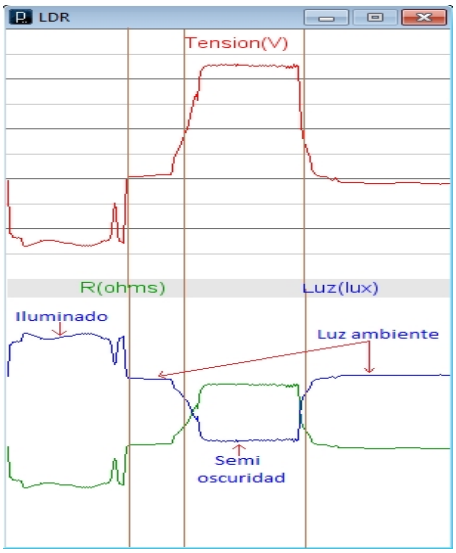


Figura 17: Captura de medidas del LDR con Processing durante cambios de iluminancia.

3.2- TERMISTOR NTC, SENSOR DE TEMPERATURA

Los termistores son sensores que varían su resistencia ante cambios en la temperatura ambiente. Hay dos tipos de termistores, los PTC (*Coeficiente Positivo de Temperatura*) que aumentan la resistencia cuando sube la temperatura y los NTC (*Coeficiente Negativo de Temperatura*) que reducen su resistencia al subir la temperatura. Ambos tienen un funcionamiento y tecnología similar, por lo que es indiferente cuál emplear para esta aplicación y se ha adquirido un termistor NTC.

Como el LDR, el NTC está constituido por un semiconductor que al aumentar la temperatura disminuye su resistividad, por tanto su resistencia óhmica. Este sensor se calibrará y se tomarán sus medidas con el siguiente montaje. Se conectará el termistor, cuya resistencia es de 65 KΩ a temperatura ambiente en serie con una resistencia de 39 KΩ. El NTC irá además conectado a 0V (cable negro) mientras que la resistencia auxiliar se pondrá a 5V con el cable rojo. La medida de tensión del nodo intermedio se muestreará a través del pin analógico 1, como muestra la figura 18.

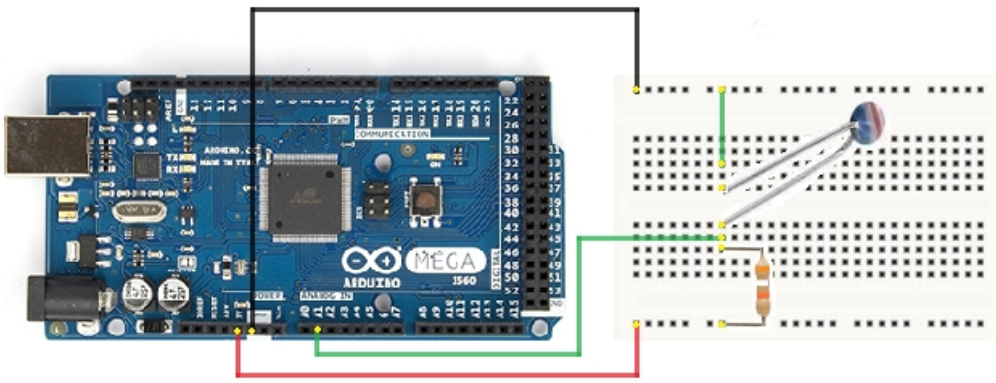


Figura 18: Montaje del sensor NTC

La calibración ha seguido un proceso ligeramente distinto, pues la expresión teórica del NTC viene dada por dos parámetros, su resistencia a la temperatura de referencia de 25ºC (o 298 K) que se ha medido y es 65 KΩ y otro parámetro desconocido que es la temperatura característica del material de la NTC. Por tanto las medidas de resistencia hechas a una temperatura conocida se han empleado en la obtención de dicho parámetro, que resultó ser de 3000 Kelvin. Con esto se pudo determinar la expresión que relaciona la temperatura T en Kelvin y resistencia que es:

$$Resistencia\_NTC(\Omega) = 65118 \text{ x } e^{-\frac{3000}{298}} \, x \, e^{\frac{3000}{T(K)}}$$

Finalmente con la relación entre la tensión medida en el pin analógico 1 y la resistencia de la NTC en ese momento se ha podido verificar si los valores de temperatura medidos eran adecuados. Las expresiones utilizadas para ello son las siguientes:

$$Resistencia\_NTC(\Omega) = \frac{tensión\_pin_1\,(V)\; x\; 39000}{5\,V - \; tensión\_pin_1\,(V)}$$

$$T(ºC) = \frac{3000}{\log\left(\frac{Resistencia_{NTC}(\Omega)}{2.7644}\right)} - 273$$

Y la curva de comportamiento resultante para este sensor es la representada en la figura 19. Comentar que las mediciones realizadas han dado valores adecuados de temperaturas en el rango de 20ºC a 40ºC y sería necesario realizar más mediciones con el fin validar su calibración para el resto de temperaturas.

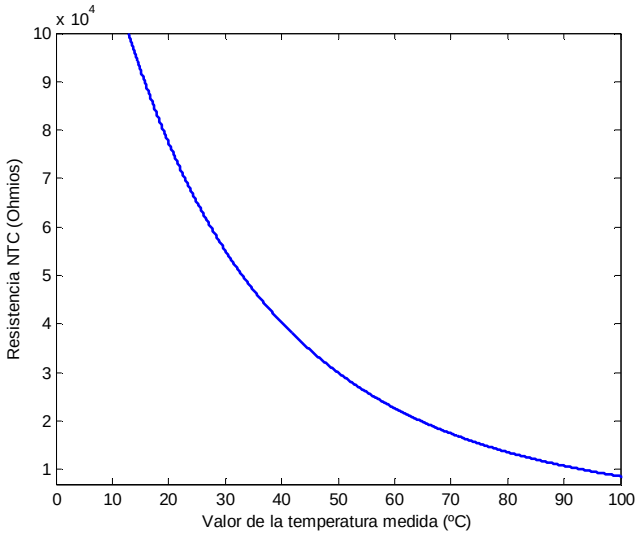


Figura 19: Curva de comportamiento del NTC (temperatura frente a resistencia)

3.3- SENSOR DIGITAL DHT11 DE TEMPERATURA/HUMEDAD

El último sensor que se utilizará es el sensor digital de humedad y temperatura DHT11. En primer lugar decir que es un sensor digital, cuya única transmisión por un pin digital son series binarias de unos y ceros ordenadas con un patrón conocido de forma que las podamos descifrar al recibirlas.

Los datos enviados por el sensor consisten en una serie de 40 bits, interpretables en paquetes de 8 bits. Los 8 primeros representan la parte entera de la medida de humedad, los 8 siguientes la parte decimal de la humedad y los dos siguientes grupos de 8 bits son la parte entera y decimal de la temperatura respectivamente. Los últimos 8 bits son la verificación de si la transmisión ha sido correcta, enviando los últimos 8 bits de la suma de los 4 grupos anteriores (parte entera y decimal de la humedad y temperatura).

Se debe aclarar que cada conjunto de 8 bits representa un número entero en sistema binario (representables 256 números, aunque sólo se enviaran datos del 0 al 100).

El proceso de comunicación a través del pin digital es el siguiente. En primer lugar, el microcontrolador pone dicho pin en estado bajo de tensión y pasado un tiempo breve vuelve a ponerlo en estado alto. En este momento el DHT11 comienza la transmisión, poniendo la señal a 0V y pasado un tiempo a 5V para que el microcontrolador esté preparado para la recepción de datos. En envío de bits es de la siguiente manera. Para empezar la transmisión de un bit, sea un 0 o un 1, el DHT11 pondrá la señal a 0 voltios durante 50 microsegundos. A continuación, pondrá la señal a nivel alto un tiempo cuya duración dependerá del dato que vaya a enviar, es decir, 70 microsegundos si se envía un 1 y 27 microsegundos si el envío es un 0. Entonces la señal vuelve a estado bajo otros 50 microsegundos y se procede al envío del siguiente bit. Este complejo proceso de envió de información se puede ver en la figura 20.

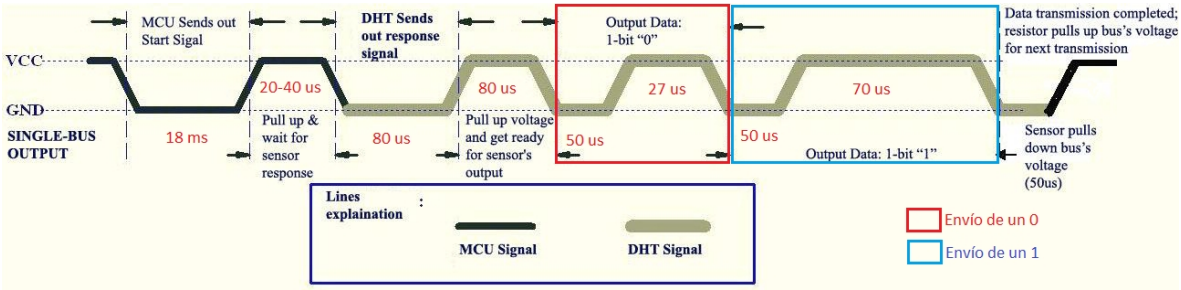


Figura 20: Proceso de transmisión de bits del DTH11 con el MCU (unidad microcontroladora) y duración de cada etapa

Una vez comprendido la manera en que se va a recibir la información, es hora de explicar cómo se conectará el sensor al Arduino MEGA 2560 y cómo adquiriremos la medida. El montaje de adquisición es el mostrado en la figura 21, conectando el pin VCC del DHT11 a 5V y el GND a ground (0V). El pin central se conectará al canal digital 2, ya que el 0 y 1 son ocupados por la comunicación del puerto serie.

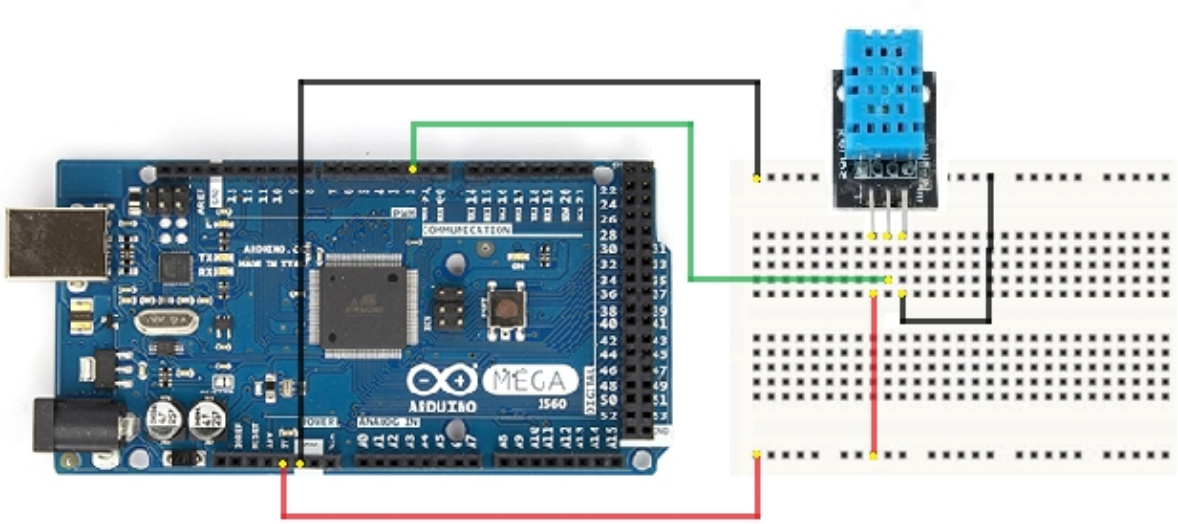


Figura 21: Montaje del sensor DHT11

La programación de la adquisición de la señal se hará con Arduino IDE. Hay 2 métodos para capturar la información. El primero y más complejo es programar nosotros mismos una serie de instrucciones que envíen la petición de medida y luego programar una serie de lecturas con un control minucioso del tiempo entre cada lectura para recibir los 40 bits de datos del DHT11, para decodificarlos a continuación y obtener las medidas de humedad y temperatura.

El segundo método, más simple, es incluir la librería *DHT.h* en Arduino IDE y emplear funciones simples que nos devolverán directamente las medidas de temperatura y humedad en forma de número real. Para ello, una vez descargada la librería y guardada en su carpeta correspondiente, se programará, se creará una constante diciendo el tipo de sensor que utilizaremos *#define DHTTYPE DHT11*, así como una variable de tipo dht con *DHT dht(DHTPIN, DHTTYPE)*, siendo DHTPIN el pin digital donde estará conectado. El sensor será inicializado dentro del *void setup* con *dht.begin()*.

Hecho esto, ya se podrán usar las funciones de medida del sensor en cualquier parte del programa con las funciones *dht.readHumidity()* y *dht.readTemperature()*, que devolverán valores de humedad, que devolverán valores reales de humedad en el rango de 20% a 90% y temperaturas entre 0-50ºC, que son los rangos de operación del sensor DHT11.

3.4- FILTRADO DE SEÑALES

Todas las medidas de los sensores anteriores presentan una ligera variación a lo largo del tiempo sin llegar a estabilizarse en un valor constante en ningún momento de la medición. Es por esto que es necesario el empleo de alguna herramienta como el filtrado digital que permita obtener señales más uniformes sin perder la calidad y la magnitud de dicha señal.

Los filtros digitales son una herramienta de procesamiento matemática que tiene como entrada una señal analógica o digital discreta en el tiempo y tiene como salida el resultado de aplicar un determinado filtrado a la señal original. Existen grandes diferencias entre los filtros analógicos tradicionales y los filtros digitales que se emplearán en este proyecto. En cuanto a su implementación, los analógicos se constituyen a base de componentes electrónicos como condensadores y amplificadores operacionales, mientras que los digitales pueden implementarse tanto en software (aplicación) como en hardware (soporte físico). Otra de las diferencias es el ancho de banda de frecuencias en el que usar el filtro, definido únicamente por el periodo de muestreo para el digital y limitado por el tipo de amplificadores operacionales para el analógico, además de una potencia de cálculo mayor para los filtros digitales.

El proceso de filtrado consiste en el tratamiento de la señal de entrada en su espectro de frecuencias con el fin de atenuar algunas frecuencias que no contienen información relevante de la señal. Para ello, la operación a realizar por el filtro sería la aplicación de una ecuación diferencial sobre la entrada del filtro, pero como las señales que se emplearán son discretas se utilizará una ecuación en diferencias que consiga el mismo efecto que la ecuación diferencial homóloga.

Dicha ecuación en diferencias del filtro viene determinada en función de unos parámetros o coeficientes del filtro y de las entradas y salidas del filtro en los instantes anteriores, cuya expresión general es:

$$y_k = \sum_{i=0}^{k} b_i \cdot x_{k-i} - \sum_{i=1}^{k} a_i \cdot y_{k-i}$$

Siendo y_i y x_i la salida y entrada del filtro en el instante "i" respectivamente, y los coeficientes b_i y a_i son los coeficientes del filtro que definen la función de transferencia del filtro o relación entre la entrada y salida. El orden del filtro viene determinado por el número de parámetros empleados para la entrada o para la salida. A mayor orden del filtro, mayor será la caída de atenuación de la señal para las frecuencias establecidas.

Los filtros que siguen esta ecuación y sus parámetros permanecen constantes durante el proceso de filtrado son conocidos como filtros digitales Lineales e Invariantes en el Tiempo (abreviado como LTI). Se puede decir que estos filtros son sistemas LTI, ya que tienen las propiedades de linealidad entre su salida y entrada e invariancia de la salida filtrada independientemente del instante de tiempo en el que la señal ha sido procesada.

Además, se deben distinguir entre dos familias de filtros discretos en relación a la ecuación anterior, que es la respuesta del sistema ante la entrada de un impulso. Si la respuesta es un pulso finito, estamos ante un filtro FIR (*Finite Impulse Response*), cuya característica es que la salida del filtro no depende de las salidas de instantes anteriores, por lo cual sus coeficientes a_i=0. Por el contrario, si la respuesta es infinita con una atenuación prolongada, el filtro empleado será del tipo IIR (In*finite Impulse Response*), cuyas salidas sí dependen de las salidas anteriores, por lo tanto su ecuación tendrá la forma completa que se ha presentado arriba.

De estas dos familias se emplearán filtros IIR en la programación de Matlab, programando nosotros mismos la ecuación del filtro con la ayuda de las funciones de Matlab para calcular sus parámetros. En Labview se emplearán los diagramas de bloques de los filtros FIR, cuya salida no depende de las salidas anteriores, por tanto estos filtros no necesitará realimentación.

Existen 4 tipos de filtros comúnmente usados. Los filtros paso bajo, que permiten pasar las frecuencias por debajo de una frecuencia de corte *(Wc)* en Hercios, atenuando las frecuencias por encima de ella (mayor atenuación cuanto más alejada esté de la frecuencia de corte). Por el contrario, los filtros paso alto atenúan las frecuencias por debajo de la frecuencia de corte. Los filtros paso banda tienen dos frecuencias de corte, una mayor que la otra y, como su nombre indica, deja pasar el espectro en frecuencias contenido entre esas dos frecuencias, atenuando la amplitud del resto. Y finalmente los filtros rechaza banda tienen también dos frecuencias de corte y eliminan las frecuencias intermedias a estas dos frecuencias de corte. La representación de estos filtros se realiza en un diagrama de Bode, donde en el eje de abscisas se disponen las frecuencias en escala logarítmica y en el eje de ordenadas se marcan los decibelios atenuados. Estos 4 filtros y sus diagramas de Bode están representados en la figura 22.

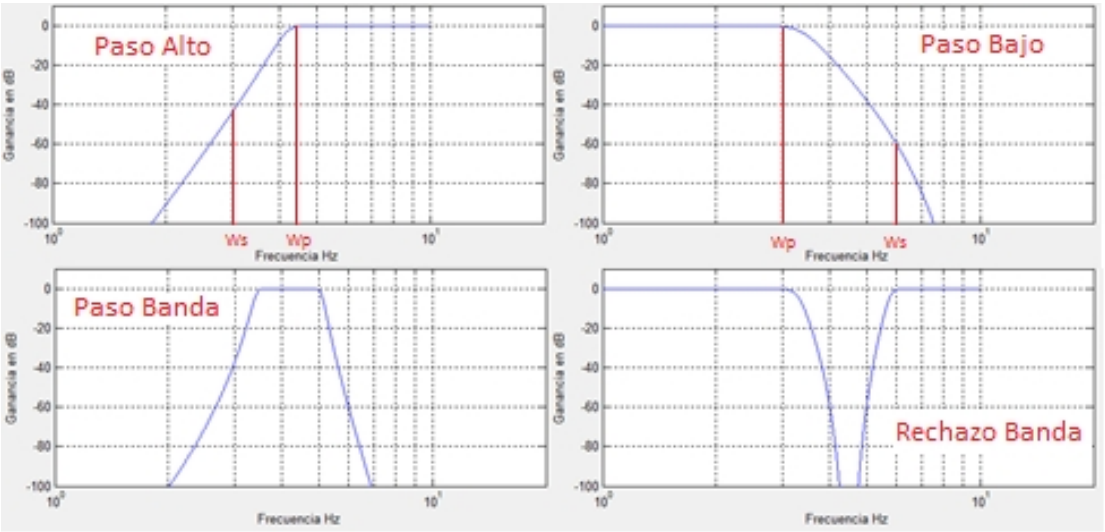


Figura 22: Tipos de filtros y sus diagramas de Bode

Por otro lado, hay infinidad de modelos de filtro que pueden aplicar cualquiera de estos tipos de filtro. Entre todos los existentes, se emplearán los filtros elípticos, Butterwoff y Chebycheff por el hecho de haber sido estudiados previamente en las asignaturas ya comentadas y ser filtros con una gran aplicación. Cada uno de ellos tiene unas características que lo diferencia de los otros dos y los hace más viables para unas u otras aplicaciones.

En cuanto a los filtros elípticos y Chebycheff, destacan por ofrecer una gran caída en decibelios en un intervalo de frecuencias muy estrecho, con el inconveniente de presentar un leve rizado del diagrama de bode antes y después de la caída para los filtros elípticos, y sólo en una de las bandas para los filtros Chebycheff, dependiendo del tipo de filtro Chebycheff (tipo 1, con rizado en la banda no atenuada y tipo 2, con el rizado en la banda de rezado).

En cambio, los filtros Butterwoff no presentan rizado alguno, siendo su respuesta mucho más plana alrededor de la frecuencia de corte, pero tiene la desventaja de que el orden del filtro crece enormemente si se busca una caída de decibelios brusca, con unas frecuencias de paso y de rechazo muy próximas.

Estos tres modelos requieren de dos parámetros que definen cada frecuencia de corte: la frecuencia de paso (*Wp*, ver figura 23) que es la frecuencia desde/a partir de la cual la señal deberá permanecer sin cambios, y la frecuencia de rechazo (Ws, ver figura 23), o frecuencia desde/a partir se deberá garantizar un cierto nivel de atenuación. Además, a la hora de diseñar estos filtros en Matlab, tenemos la opción de elegir el nivel de atenuación máximo permitido en la banda de paso (*Rp*) y el nivel de atenuación mínimo aceptado en la banda de rechazo (Rs), ambos el decibelios.

Para la aplicación que requiere el proyecto solamente se usarán los filtros paso bajo, ya que los datos monitorizados suelen tener una parte importante de señal continua y constante, que es de frecuencia 0 o muy próxima, y todo aquello que supere la frecuencia de corte establecida será considerado ruido a eliminar. En la figura 23 se presentan los modelos de filtro finalmente empleados.

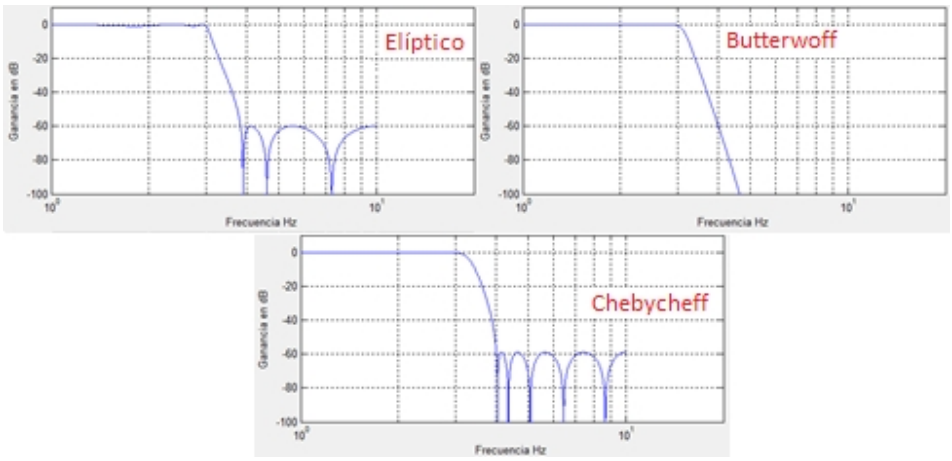


Figura 23: Modelos de filtros paso bajo empleados

A continuación se presenta la implementación de algunos de estos filtros en los programas de Matlab y Labview presentados en el apartado 4 de la memoria.

En la figura 24 se han adjuntado dos fragmentos del código de Matlab. Las dos líneas superiores son el cálculo del orden del filtro y sus coeficientes en relación a las frecuencias de paso y de rechazo elegidas y las atenuaciones máximas y mínimas permitidas. En cuanto a las siguientes líneas, muestran la implementación del filtrado, concretamente un filtrado online de orden 2 (la expresión será análoga para otro orden de filtro). En el caso de un filtrado offline, Matlab permite utilizar la función *Y=filter(b,a,X)*, donde Y e X son los vectores de salida y entrada respectivamente.

```
[N,Wn]=buttord(Wp/(f/2), Ws/(f/2), Rp, Rs); % N, orden del filtro
[b,a]=butter(N,Wn); % a,b, vectores de coeficientes
……..
……..
% después de tomar la medida i, en el caso de un filtro de orden 2:
if N==2
   if i==1     Vel_filt(i)=b(1)*Vel(i);
   elseif i==2 Vel_filt(i)=b(1)*Vel(i)+ b(2)*Vel(i-1)- a(2)*Vel_filt(i-1);
   else
   Vel_filt(i)=b(1)*Vel(i)+ b(2)*Vel(i-1) + b(3)*Vel(i-2) – a(2)*Vel_filt(i-1)
   - a(3)*Vel_filt(i-2);
   end
end % Vel es la señal de entrada, y Vel_filt es la salida del filtro
```

Figura 24: Diseño del filtro y filtrado online en Matlab.

En relación al filtrado en Labview, el diagrama de bloques resultante en el caso de un filtrado online sería el mostrado en la figura 25. Se ha programado un selector de casos según el modelo de filtro (Chebycheff en este ejemplo), que incluye el bloque de filtrado cuyas entradas son el vector de entrada, las frecuencias de corte y el tipo de filtro. La salida es una variable de tipo vector, de la que tomaremos únicamente el último valor para graficarla junto a los filtrados anteriores. Para el filtrado offline el diagrama es muy similar, sólo que el vector de entradas no procede de la lectura de Arduino, sino de un fichero de datos.

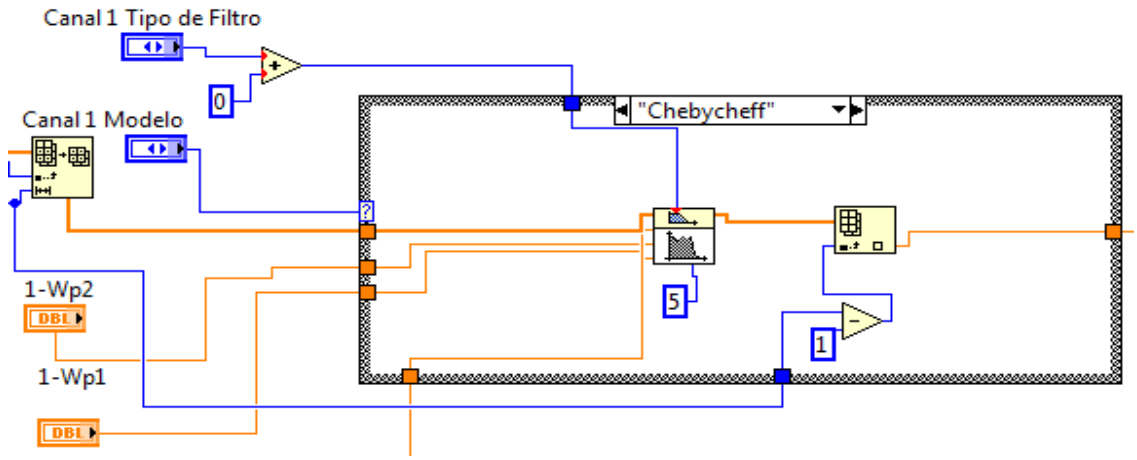


Figura 25: Filtrado online en Labview

4- DISEÑO DEL ENTORNO DE CAPTURA Y PROCESAMIENTO DE SEÑALES

4.1- INTRODUCCIÓN A LA CAPTURA Y PROCESAMIENTO DE SEÑALES

El tema de desarrollo de este proyecto no es algo innovador ni revolucionario. Hace miles de años que el hombre toma mediciones de los fenómenos que ocurren a su alrededor y los registraban sobre papiros o materiales similares, tanto en la investigación como en la medicina o arquitectura. Incluso era posible un procesamiento de señales manual antes del auge tecnológico y la generalización del uso de los computadores, ya que lo único necesario es la realización de operaciones y cálculos matemáticos sobre una serie de datos. Es más, se conoce que se utilizaron técnicas rudimentarias de procesamiento digital de señales siglos atrás para realizar predicciones sobre el movimiento de cuerpos celestes y las mareas, entre otros.

Los primeros tratamientos de señales modernos se realizaron con la ayuda de circuitos electrónicos. Y con la llegada de los microcomputadores, estas técnicas de procesamiento se extendieron a muchas aplicaciones tecnológicas como el radar, sónar, electrocardiogramas, televisión o reconocimientos de voz. Muchas de estas aplicaciones necesitan que el sistema opere en tiempo real, es decir, que los datos de salida sean calculados a la misma velocidad de entrada de datos, hecho que ha sido posible con un aumento progresivo de la velocidad de cálculo e integración de los microprocesadores.

Debido a la diversidad de aplicaciones que tiene la captura y procesamiento de señales se han desarrollado una infinidad de softwares que cumplen con alguno de estos dos propósitos, adaptándose a las necesidades de aquellos que demandan su uso.

Por todo esto se puede afirmar sobre este proyecto que, a pesar de que todo lo que se programe, desarrolle y se diseñe en los puntos siguientes de la memoria ya está inventado, es de gran importancia el haber encontrado una manera diferente y personal de hacer las cosas, adaptando los programas diseñados a las necesidades que se han creído convenientes. De esta manera, habremos logrado diseñar un conjunto de programas propios, que como programadores, conoceremos en detalle y podremos emplearlos en las aplicaciones concretas para las que han sido pensados.

4.2- NECESIDADES DE LA PLATAFORMA A DISEÑAR

Una de las principales necesidades que las aplicaciones deben satisfacer es el poder tener control de lectura y escritura sobre los pines analógicos y digitales de manera muy intuitiva, pudiendo además controlar el tiempo de muestreo entre dos medidas consecutivas. Respecto a los pines digitales su estado será visualizado en todo momento desde unos leds asociados a cada pin. Se dedicará mayor esfuerzo al análisis de las señales analógicas, pudiendo escoger uno o varios pines analógicos y ver tanto su valor en tiempo real como su gráfica representada por sus valores anteriores.

Estos datos analógicos podrán recibir distintos tipos de procesamiento. El primero de ellos, su filtrado, que se podrá realizar mediante dos métodos: el filtrado online, donde los datos serán filtrados al mismo tiempo que son adquiridos y el filtrado offline, que será realizado a partir de una señal anteriormente adquirida cuyos datos estarán guardados en un fichero de texto con un formato estándar. Se podrá elegir además el modelo de filtro (elíptico, Butterwoff o Chebycheff) y el tipo de filtro (paso alto, paso bajo, paso banda, rechazo banda), además de las frecuencias de corte de dicho filtro, por lo que el usuario deberá tener cierto conocimiento teórico sobre elegir estos parámetros. Si este no fuera el caso, se recomendarán para cada aplicación unos parámetros de filtro de aplicación general.

Por otra parte, se dará la opción de operar con los datos recibidos dos canales analógicos, con operaciones como sumar, restar o multiplicar sus valores en cada instante de tiempo. También se programarán otras herramientas como la amplificación o atenuación de cada uno de estos dos canales multiplicando cada valor por un factor de amplificación, o el reducir la componente continua de la señal sumándole o restándole un parámetro manipulable. Esto será útil en aplicaciones que requieran adaptar la señal de entrada a ciertos valores de tensión y posteriormente enviarla por un pin PWM del microcontrolador después de ser procesada, lista para ser usada en la aplicación que lo requiera. Otro ejemplo sería reducir o amplificar el volumen de una señal de audio adquirida por micrófono para adecuar la señal a un nivel de decibelios más adecuado.

Finalmente se requiere de una herramienta para guardar las adquisiciones realizadas y poder emplearlas en otro momento o que sean utilizadas por otros programas que no se hayan utilizado para adquirirla. Para ello se ha decidido emplear el guardado en ficheros de texto, donde los datos estarán guardados en dos columnas. La primera de ellas indicará en tiempo transcurrido desde el inicio de la adquisición. La segunda columna será el valor de tensión acotado entre 0 y 5 V de la señal recibida por un pin analógico. Estos ficheros, una vez adquiridos, podrán recibir el procesamiento anterior, y el resultado de estos filtrados será guardado en otro fichero distinto. Por supuesto, el nombre del fichero donde se guardan los datos podrá ser editado desde la propia aplicación.

Una vez explicadas las necesidades que deben cubrir los programas diseñados, es hora de presentar finalmente las distintas aplicaciones desarrolladas, que entre todas cumplen con los objetivos propuestos.

4.3- DISEÑO CON LA INTERFAZ GRÁFICA DE MATLAB

Como se explicó en la programación de Matlab, el diseño de esta aplicación se basará en una GUI (o interfaz gráfica de usuario) con botones, leds indicadores y gráficas donde visualizar la información. Para ello ha sido necesario el diseño de la interfaz en una figura, cuyo aspecto se puede ver en la figura 26 y la programación de un archivo *.m* que contenga las funciones de creación de cada uno de los objetos y sus llamadas, es decir, funciones que se activarán únicamente cuando el usuario interactúe con el ratón o teclado con dicho objeto. Sobre este archivo .m se escribirán la mayor parte de las instrucciones para controlar, capturar y procesar los datos recibidos de Arduino, conectado a través del cable USB con el ordenador y comunicándose con él mediante el puerto Serie "COM7".

Figura 26: Aspecto de la interfaz del programa

Además se necesitarán cuatro funciones auxiliares que serán llamadas varias veces desde distintas líneas del programa principal. Estas *son lectura_digital*, que será llamada periódicamente para realizar la operación escogida sobre un pin digital, y actualizará cada vez los leds indicadores del estado de los pines digitales. Las dos siguientes, *param_bode* y *dibujar_bode*, calculan los coeficientes y el orden del filtro según las opciones escogidas en el módulo de filtrado, además, la segunda de ellas dibuja el diagrama de bode del filtro una vez es presionado el botón "Ver bode". La última función auxiliar es *filtro_online*, que calculará para el instante de tiempo actual, la salida del filtrado de la señal analógica del pin elegido, teniendo en cuenta los valores anteriores de entradas y salidas del filtro.

Vamos a explicar las distintas opciones y botones que tiene la GUI diseñada. Empezando por la parte de arriba a la izquierda, tenemos el botón naranja de "RUN", que al presionarlo iniciará un bucle de comunicación entre Arduino y Matlab en el que en cada iteración se enviará o recibirá una medida digita y una medida analógica. Este bucle seguirá activo hasta que se pulse el botón verde "detener".

A su izquierda se encuentra el cuadro para modificar el tiempo de muestreo. Se ha programado que su valor mínimo sea 50 milisegundos, ya que el tiempo de ejecución del bucle de captura cuando las opciones de procesamiento estén ejecutándose está entre los 40 y 50 milisegundos. De lo contrario, si el usuario intenta introducir un valor menor, el tiempo será sustituido por el valor mínimo programado que es 50 ms.

A su izquierda está el Panel de acciones, se encuentra el botón de "Read/Write". Este botón se deberá pulsar a continuación de "RUN" si queremos iniciar la lectura y escritura de datos con Arduino. De lo contrario, no será necesario activarlo en caso de que se quiera únicamente usar el módulo de filtrado (par diseño de filtros y filtrado online).

Bajamos al grupo "ModoWrite". Aquí se puede controlar la escritura sobre cualquier pin digital entre el 2 y el 13 (ya que estos son los pines más comunes de empleo). Para ello, se activa una de las dos opciones disponibles, "Digital" si deseamos escribir un valor de tensión de 0 o 5 V o "PWM" si lo que se busca es poner cierto pin a una tensión intermedia entre 0 y 5 V. A continuación se elige el pin a escribir con el menú despegable "PIN" adecuado, uno para digital y otro par PWM. Y finalmente se elige qué escribir, si el pin a 0 V con "LOW", a 5 V con "HIGH" o un valor intermedio PWM con la barra deslizante que va desde 0 (o 0V) a 255 (o 5V).

A la derecha de éste está el "ModoRead" de lectura digital y analógica. En la parte digital, todos los pines del 2 al 13 son leídos automáticamente, y unos leds indican el estado de cada pin. El color rojo indica que el pin está en estado bajo y el color verde, que está en estado alto. Cualquier color intermedio indica que en ese pin está siendo escribida una señal PWM, cuya tonalidad variará entre verde y rojo dependiendo de si está más cerca de 0V (rojo) o de 5V (verde).

En cuanto a la parte Analógica del "ModoRead", dispone de un menú desplegable de elección de pin y, a su derecha, un cuadro de texto gris donde se ve la tensión de dicho pin, medida que se refresca periódicamente con el programa. El botón de "Reanudar gráfica" iniciará la adquisición de la señal analógica del canal elegido y será representada en la gráfica superior derecha. Además, los datos adquiridos desde que el botón fue pulsado serán guardados en el fichero *.txt* cuyo nombre puede ser editado en el cuadro blanco datos1 (en ese caso, el fichero creado será datos.txt). El formato de este fichero ya ha sido explicado: dos columnas, la primera, vector de tiempos y la segunda, tensiones del pin muestreado.

El resultado de una adquisición sobre el pin analógico 0 puede verse en la figura 27. Para detener la adquisición se pulsará sobre "Detener gráfica", que es el mismo botón que "Reanudar gráfica" pero con la etiqueta actualizada.

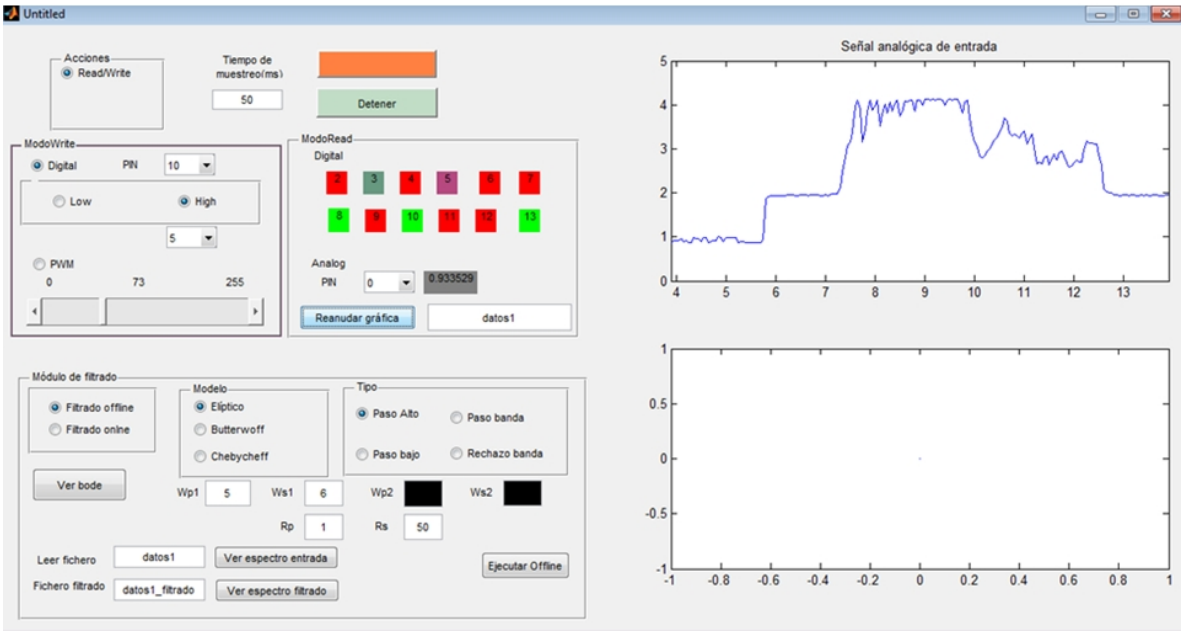


Figura 27: Captura de una señal analógica

A continuación, se procederá la explicación del módulo de filtrado. En la interfaz se pueden ver todas las opciones, modos y tipos de filtrado de los que se hablaron en el apartado *3.4- Filtrado de señales*. Se escogerá uno de los dos filtrados realizables (offline y online), un modelo y un tipo de filtrado. A continuación se procederá a la elección de los parámetros del filtro. Estos son la frecuencia de paso "Wp1" y frecuencia de rechazo "Ws1" para los filtros paso bajo y paso alto, como también "Wp2" y "Ws2", que son las frecuencias de paso y de rechazo superiores si escogemos un filtrado paso banda o rechazo banda (se recuerda que estos dos filtros requieren de dos frecuencias de corte, por eso hacen falta los parámetros que definen la frecuencia de corte inferior (Wp1 y Ws1) y la frecuencia de corte superior (Wp2 y Ws2). La magnitud de estas frecuencias escogidas son los Hercios.

Los otros dos parámetros, "Rp" y "Rs" son los límites de atenuación máximo y mínimo permitidos para las bandas de paso y de rechazo, respectivamente, ambos con unidades de decibelios.

Pero antes de elegir los parámetros debemos conocer el espectro en frecuencias de la señal a muestrear. Para ello, se debe realizar el desarrollo en serie de Fourier de la señal a filtrar y representar las frecuencias obtenidas del análisis con su amplitud correspondiente en un diagrama amplitud frente a frecuencia. En este diagrama el usuario debe ser capaz de reconocer que intervalo de frecuencias es el de interés y de acuerdo con esto elegir los parámetros del filtro que elimine el resto de información. Este espectro en frecuencias se puede ver en la figura 28.

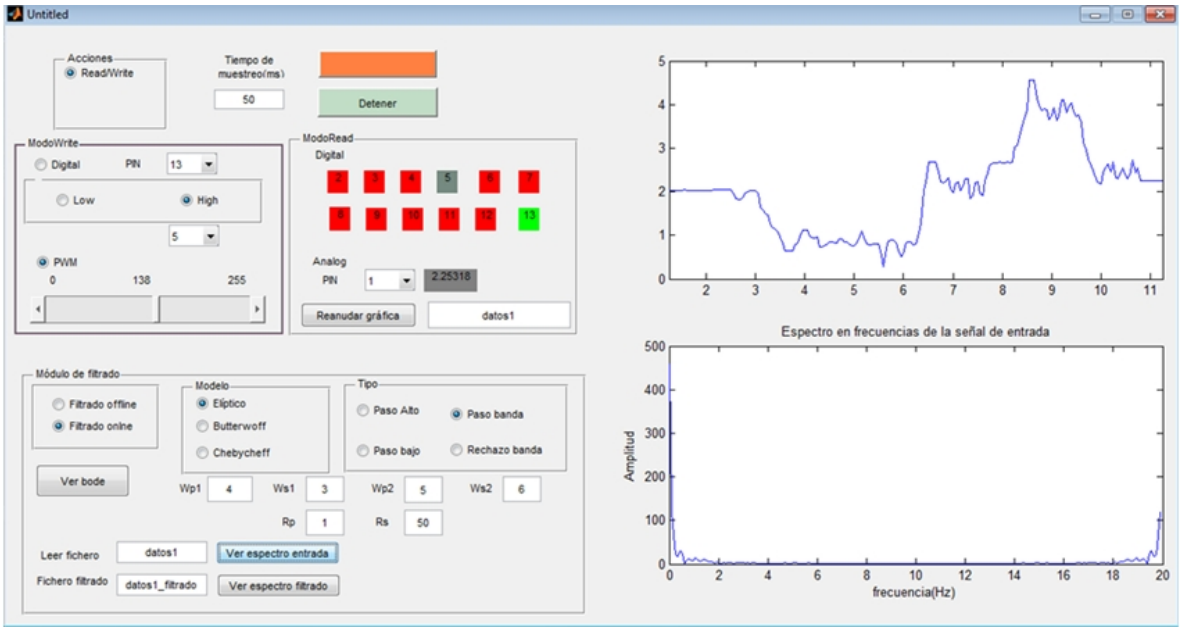


Figura 28: Análisis del espectro en frecuencias de la señal

Para la obtención de dicho espectro, debemos escoger en primer lugar el fichero que contiene la señal filtrada. Para ello se han preparado dos direcciones de ficheros editables en la parte inferior izquierda nombrados datos1 (que leerá datos1.txt) y datos1_filtrado. Esta segunda dirección de fichero tiene dos funciones: servir cono fichero de escritura para la señal filtrada, tanto online como offline (en un fichero llamado datos1_filtrado, siguiendo el ejemplo) y ser el fichero que guarda la señal de la que podemos ver el espectro. Para esto último, se presiona el botón “Ver espectro entrada” (o “Ver espectro filtrado”, una vez se haya creado el fichero). Con esto, podremos ver el espectro en frecuencias de la señal elegida en la gráfica inferior derecha (Figura 28). Claramente se aprecia que toda la información de interés está a frecuencias bajas y en alguna frecuencia entre 1 y 2 Hercios la información deja de ser relevante.

Es por eso que, como regla general, se diseñara un filtro paso bajo, con frecuencia de paso Wp1 entre 0.75-1.5 Hz y frecuencia de rechazo Ws1 entre 1-2 Hz. Recordar que Wp1 debe ser menor que Ws1 para que el filtro esté bien diseñado (Wp1<Ws1). Finalmente, se elegirán parámetros de atenuación de Rp entre 1-5 dB y valores de Rs entre 25-50 dB.

Hecho esto, se pulsará “Ver Bode” para verificar si el filtro diseñado tiene las características escogidas, como se puede ver en la figura 29 en la gráfica inferior derecha.

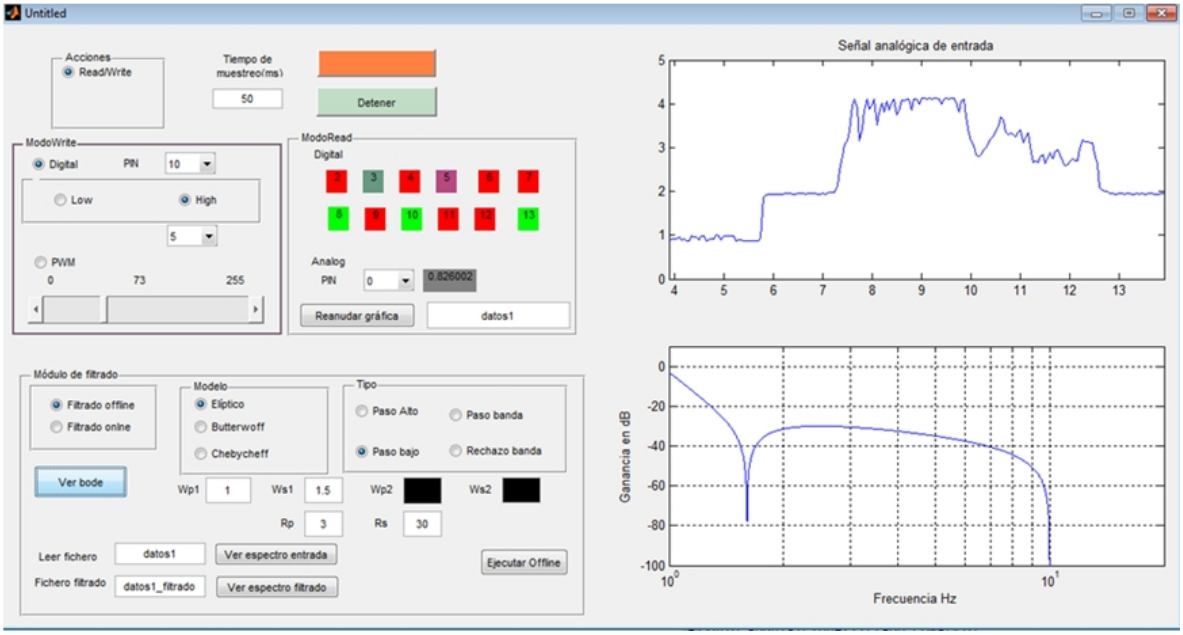


Figura 29: Elección del filtro y sus parámetros

Una vez se tiene el diseño del filtro, se puede proceder al filtrado de la señal. En primer lugar, se explicará el filtrado offline, pues que llegado a este punto ya tenemos un fichero de datos llamado "datos1.txt". Para ello, se escribe en el cuadro de texto "Leer fichero" el nombre del fichero que contiene la señal a filtrar, y en "Fichero filtrado" en nombre del fichero que contendrá la salida del filtro. A continuación, se pulsa sobre el botón "Ejecutar Offline", y el resultado de esta operación se visualizará en la gráfica inferior derecha, resultado que se puede ver en la figura 30.

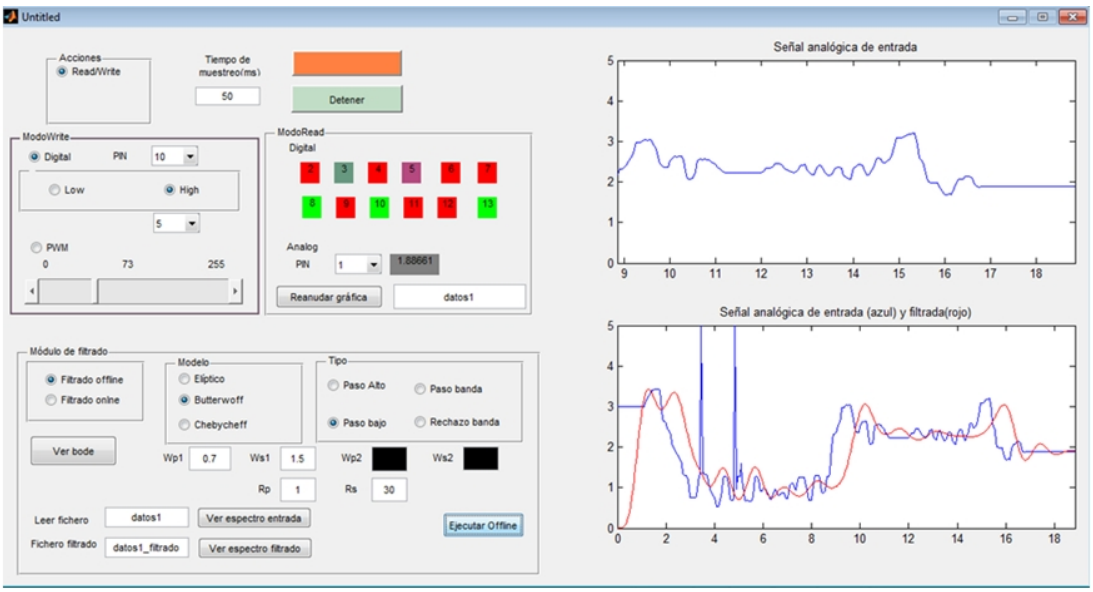


Figura 30: Filtrado offline

En la figura 30, se distingue la señal original en azul, con diversos ruidos a lo largo del tiempo, y la señal filtrada en rojo. A simple vista, es un buen filtrado ya que ha anulado completamente la mayoría de ruidos a excepción de los picos azules que aparecen alrededor de los 4 segundos (el eje X de esta gráfica representa ahora segundos de muestreo) que han sido fuertemente atenuados.

En general, se aprecia un ligero retraso en el tiempo de la señal filtrada respecto a la original. Esto es debido a la naturaleza del filtro, que hace que exista un desfase entre la salida y entrada, lo que origina un retraso de la señal filtrada.

Finalmente, se explicará el filtrado online con esta aplicación. Para ello, se escoge la opción "Filtrado online" y, con los parámetros del filtro escogidos (si se realiza algún cambio se debe pulsar en "Ver Bode" para actualizar los cambios) se pulsa el botón "Reanudar gráfica" en el grupo ModoRead. En este momento comienza la adquisición de la señal en la gráfica superior en azul, mientras que la señal va siendo filtrada en tiempo real en la gráfica inferior en rojo, a medida que los datos van siendo adquiridos. El resultado se puede apreciar en la figura 31, apreciándose un buen resultado en el filtrado logrado.

Los datos adquiridos y filtrados serán guardados también en ficheros de nombres editables en el cuadro a la derecha de "Reanudar/Detener gráfica" y más abajo en el campo de "Fichero filtrado".

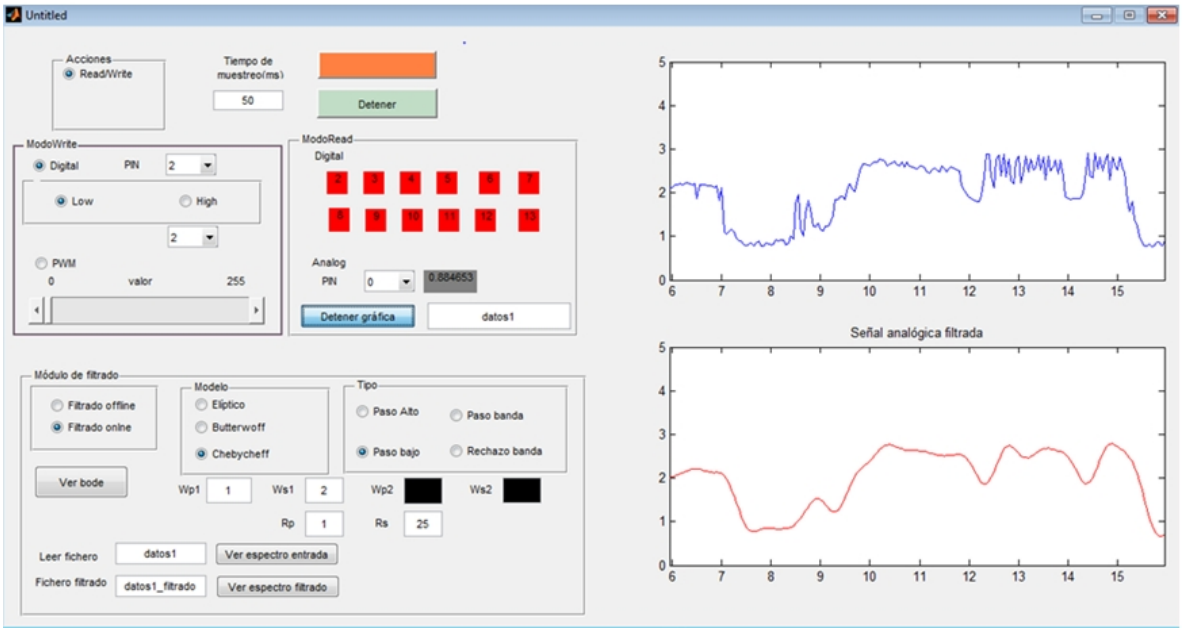


Figura 31: filtrado online

Cuando se desee cerrar la aplicación se debe pulsar el botón verde "Detener", que cerrará la comunicación entre Arduino y Matlab a través del puerto Serial. Hasta aquí llega la explicación de todas las operaciones realizables con esta GUI de Matlab. Ahora se complementará con el diagrama de la figura 32, que describe su funcionamiento y los detalles de implementación del código.

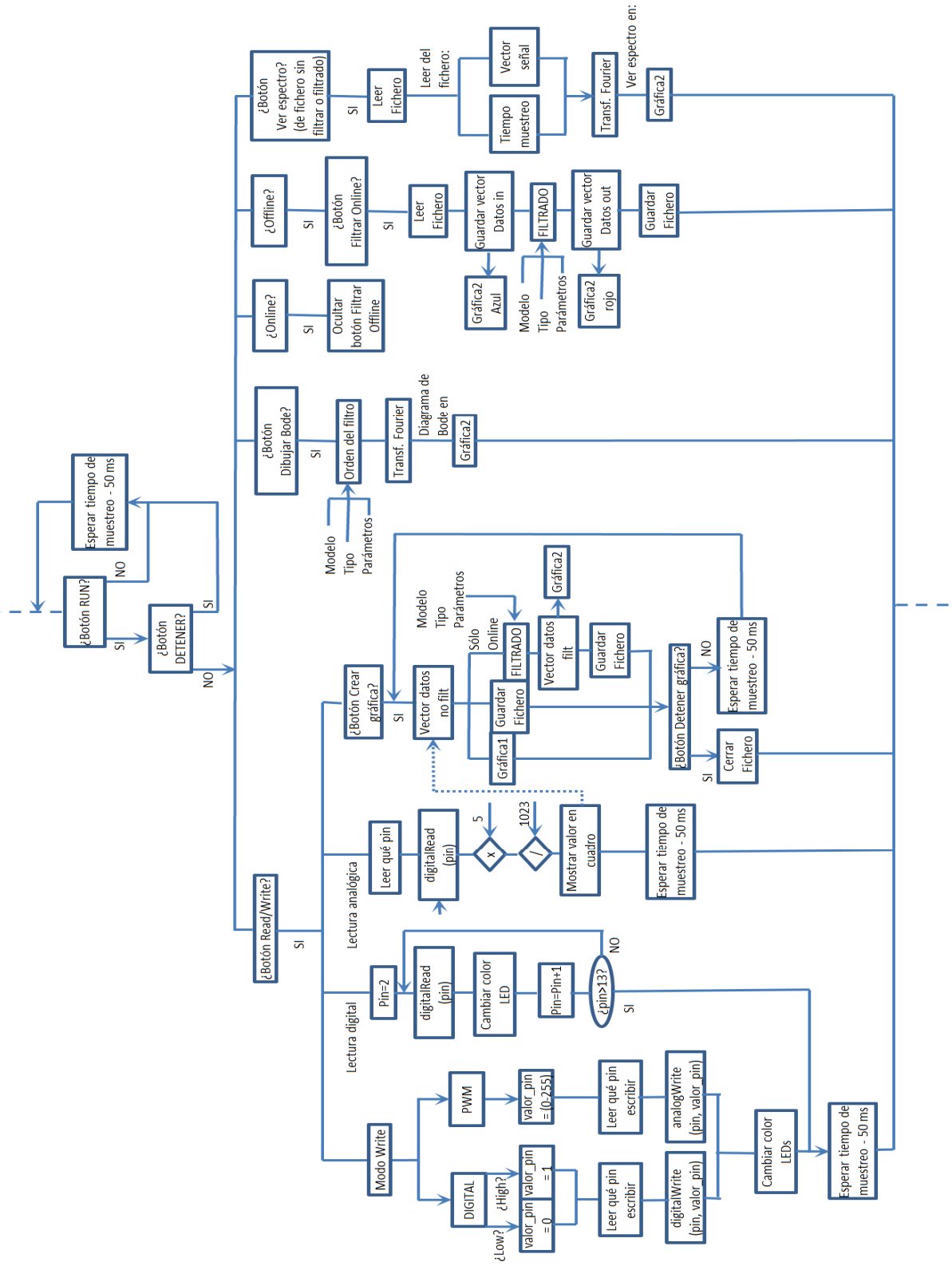


Figura 32: Diagrama de flujo de la programación realizada en Matlab.

Para complementar este diagrama de flujo del código del programa es necesario hacer referencia de algunas partes del diagrama a su respectivo código de programación. Se empieza analizando el Modo Write, que realiza la escritura sobre pines digitales del valor elegido por el usuario, cuyo diagrama parcial es el siguiente:

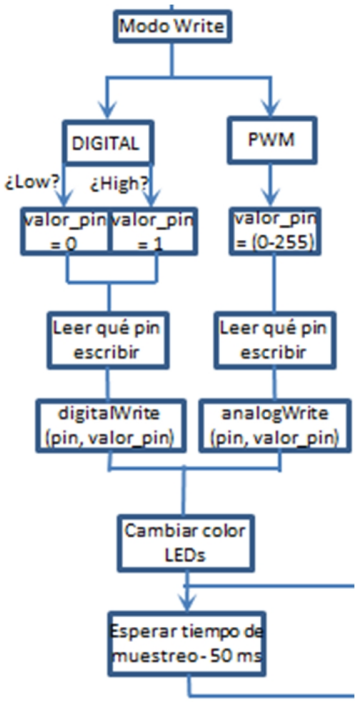


Figura 33: Diagrama parcial de Matlab: Modo Write.

El fragmento de código que realiza esta operación es el de la figura 34. Las instrucciones más importantes del código están explicadas con comentarios en verde después de cada instrucción. Básicamente, si el usuario escoge el Modo Write, se detectará si lo que desea escribir es un valor digital (0 o 1) o una PWM sobre el pin elegido, escribiendo este valor el dicho pin y finalmente esperar a que transcurra el tiempo de muestreo fijado y actualizar los Leds indicadores del estado de los pines del microcontrolador.

```
if (get(handles.Write,'Value')==1)

 if (get(handles.Digital,'Value')==1) % si se escribe un valor digital
        if (get(handles.Low,'Value')==1) handles.valorpin=0;
        end
        if (get(handles.High,'Value')==1) handles.valorpin=1;
        end                                % guardar el valor a escribir
    handles.num2=get(handles.selecion,'Value')+1; % guardar el número de pin
    handles.a.pinMode(handles.num2,'output');
    handles.a.digitalWrite(handles.num2,handles.valorpin);
                                % escribir valor digital sobre el pin
    handles.pinpwm(handles.num2)=0;
    end
```

```
    if (get(handles.Analog,'Value')==1) % si se escribe un valor PWM
    handles.valorpin=floor(get(handles.barra,'Value'));
                                    % guardar el valor a escribir
    handles.num2=get(handles.selecion3,'Value')+1; % guardar el número de pin
    handles.a.pinMode(handles.num2,'output');
    handles.a.analogWrite(handles.num2,handles.valorpin);
                                     % escribir PWM sobre el pin
        if(handles.valorpin==255 || handles.valorpin==0)
            handles.pinpwm(handles.num2)=0;
        else
            handles.pinpwm(handles.num2)=1;
            handles.valor_pinpwm(handles.num2)=handles.valorpin;
        end            % guardar el estado del pin para encendido de los leds

    end

tiempo=max(handles.temp-0.05,0.001);
pause(tiempo); % esperar tiempo de muestreo menos 50 milisegundos
lectura_digital(handles); % función de actualizado de leds
```

Figura 34: Código de programación de Matlab: Modo Write.

El segundo y último diagrama parcial de Matlab que se analizará es el del filtrado offline, que se encuentra en la figura 35. La idea básica del código es realizar la lectura de un fichero de datos elegido por el usuario, guardarlos temporalmente mediante un tratamiento de vectores y filtrar dicha señal según el modelo, tipo de filtro y parámetros elegidos. El resultado este filtrado será guardado en otro fichero y tanto la entrada como salida del filtro serán representadas en distintas gráficas.

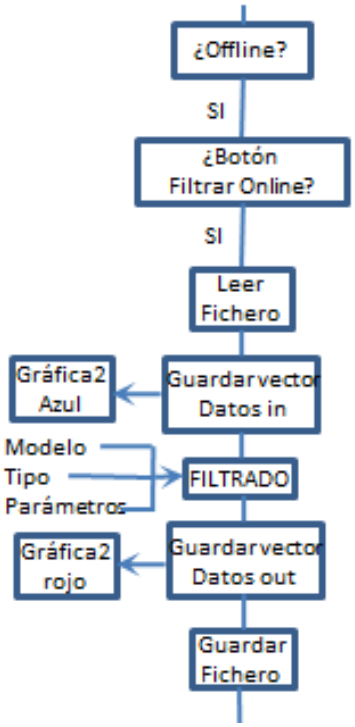


Figura 35: Diagrama parcial de Matlab: Filtrado Offline.

Comentar que *handles* es la estructura de datos que utiliza por defecto la GUI para guardar el estado de todos sus objetos y otras variables. El código de programación que realiza las acciones del filtrado offline es el siguiente, con las explicaciones de las instrucciones más relevantes en verde.

```
[N,b,a]=param_bode(handles); % función propia, calcula los parámetros del
                             filtro
clear handles.datos_in;    % limpiar vectores de entrada/salida del filtro
clear handles.datos_out;
nombre=get(handles.leer1,'String');
nombre=strcat(nombre,'.txt');
handles.datos_in=load(nombre); % leer fichero de entrada de datos

handles.datos_out=handles.datos_in(:,1); % 1ª columna, vector de tiempos.

handles.datos_out(:,2) = filter(b,a,handles.datos_in(:,2));
                               % 2ª columna, vector de datos ya Filtrados

 nombre2=get(handles.leer2,'String');
 nombre2=strcat(nombre2,'.txt');
 fid=fopen(nombre2,'w');
lon=length(handles.datos_out(:,1));
for i=1:1:lon
fprintf(fid,'%d \t %d \n',handles.datos_out(i,1),handles.datos_out(i,2));
end          % guardar datos filtrados en fichero, con el formato estándar
fclose(fid); % cerrar fichero
hold off
plot(handles.axes2,handles.datos_in(:,1),handles.datos_in(:,2),'b');
hold on
plot(handles.axes2,handles.datos_out(:,1),handles.datos_out(:,2),'r');
axis(handles.axes2,[0 handles.datos_out(lon,1) 0 5]);
title(handles.axes2,'Señal analógica de entrada (azul) y filtrada(rojo)');
             % Gráficas de las dos señales,sin filtrar y filtrada.
```

Figura 36: Diagrama parcial de Matlab: Filtrado Offline.

4.4- PROPUESTA DE PROCESAMIENTO CON LABVIEW:

La aplicación desarrollada con Labview permitirá realizar operaciones similares a la GUI de Matlab, pero además ofrece una serie de mejoras en cuanto a herramientas de procesamiento. Para la programación destaca el empleo de diversas estructuras como bucles while, elección de casos *case* o estructuras por secuencia de bloques (donde una serie de instrucciones no empezará hasta que la anterior haya concluido) y variables locales que serán ocultas bajo una pestaña "Auxiliares" no accesibles al usuario, que hay servido de gran ayuda a la hora de controlar las distintas línea de ejecución del programa y el flujo de datos.

En cuanto al panel frontal, que es lo único que verá el usuario, se distingue una pestaña de controles en la parte superior, como se puede ver en la figura 30. En esta barra de controles se encuentra un botón de "STOP" general que detendrá por completo la ejecución, un campo editable de periodo de muestreo (que nuevamente debe ser superior a 50 ms debido a los tiempos de ejecución de los bucles de adquisición) y una serie de palancas que permiten al usuario acceder a las distintas pestañas de la aplicación. Por último, hay dos botones, "Iniciar/Detener captura ONLINE", que iniciará la adquisición y guardado en los ficheros correspondientes y "Iniciar filtrado ONLINE", que ejecutará el filtrado de señales guardadas en ficheros, volcando el resultado en un fichero diferente.

Las palancas intermedias de la barra de controles sirven para moverse entre las pestañas. La pestaña 1 "Digital" (figura 37) será accesible si la palanca de la izquierda está en la posición "Digital". Si dicha palanca la movemos a "Analog", se cambiará de pestaña dependiendo de la posición de la segunda palanca. Si la palanca intermedia está en "Control analógico", la ventana cambiará a la pestaña 2 "Analógico Control" (figura 38). Por el contrario, si la palanca intermedia marca "Operación analógica", la ventana pasará a la pestaña 3 llamada "Analógica Operación" (figura 41).

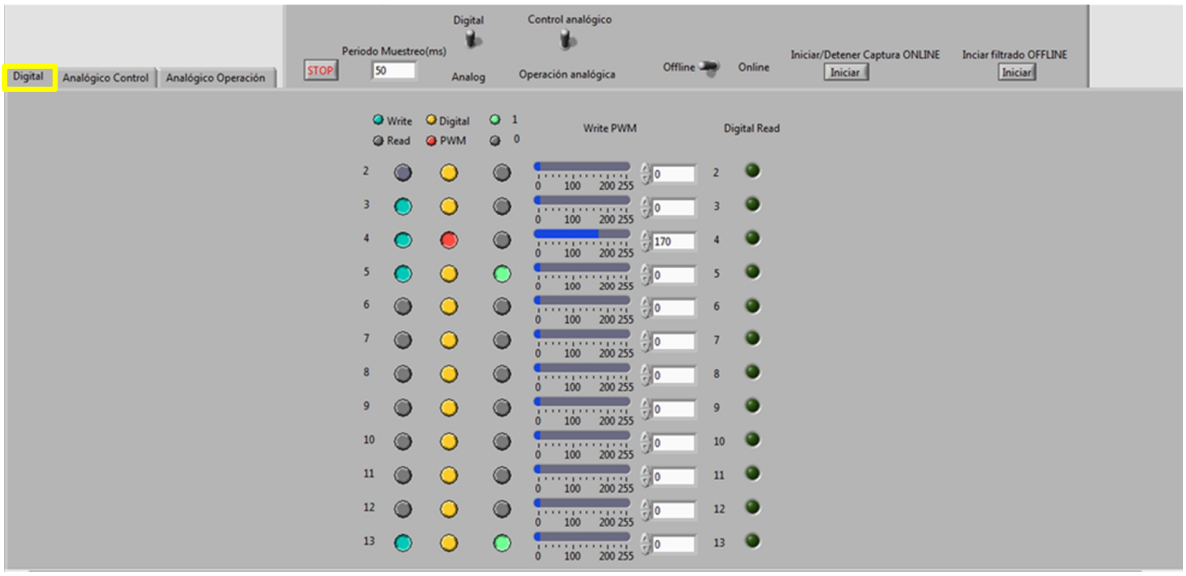


Figura 37: Pestaña 1 - Control y lectura digital

Vamos a explicar primero las opciones de la pestaña “Digital”, que se puede ver en la figura 37 de la página superior. En la parte izquierda hay 3 columnas de pulsadores donde cada fila está etiquetada con un número (que hace referencia al pin que controlará esa fila de botones), a la derecha de cada fila hay unas barras deslizantes con el valor numérico que marcan a su derecha, y que servirán para elegir el valor de PWM que se escribirá sobre cada pin. Por último, hay una columna de leds digitales que indicarán el estado del pin correspondiente, si esa es la acción de control elegida. El led permanecerá apagado si su pin está a menos de 2.5 V y se encenderá si su pin está a una tensión superior a 2.5 V.

Las tres columnas de botones de la izquierda son los que controlan las acciones sobre los pines digitales. Se ha adjuntado una leyenda sobre la función de cada columna en la parte superior. La primera columna controla si la acción a realizar sobre ese pin es de escritura (botón azul) o de lectura (botón gris). Si la opción de escritura es elegida, la segunda columna elige si la acción a realizar es la escritura de un valor digital (botón amarillo) o de una PWM (botón rojo). Además, la tercera columna de leds elige qué valor digital escribir cuando esté elegida la opción de escritura digital, de manera que el pin correspondiente será puesto a 0 V si se escoge un 0 (botón gris) o será puesto a 5 V si se elige un 1 (botón verde). Por último, el palor de PWM escrito si está opción es activada se controlara con la barra deslizante al igual que en Matlab.

En cuanto a la parte analógica, figura 38, hay dos pestañas para adquirir señales y procesarlas. La primera de ellas, “Analógico Control”, es donde se definen los parámetros del filtro y los nombres de los ficheros donde se guardarán y cargarán las señales. Se disponen de dos canales para leer simultáneamente dos pines analógicos distintos (elegibles en la tercera pestaña). La configuración del filtro y ficheros del canal 1 se realiza en el lado izquierdo, donde se elige la dirección y el nombre del fichero donde guardar los datos sin filtrar y filtrados. Se elige el tipo de filtro y el modelo de filtro con los botones correspondientes y se escogen las frecuencias de corte. En este caso Wp1 se corresponde a la frecuencia de corte 1 y Wp2 es la frecuencia de corte 2, solamente empleada en los filtros pasa banda y rechazo banda. Se elegirá en general un paso bajo con un valor de Wp1 entre 0.75 y 1.5 Hz.

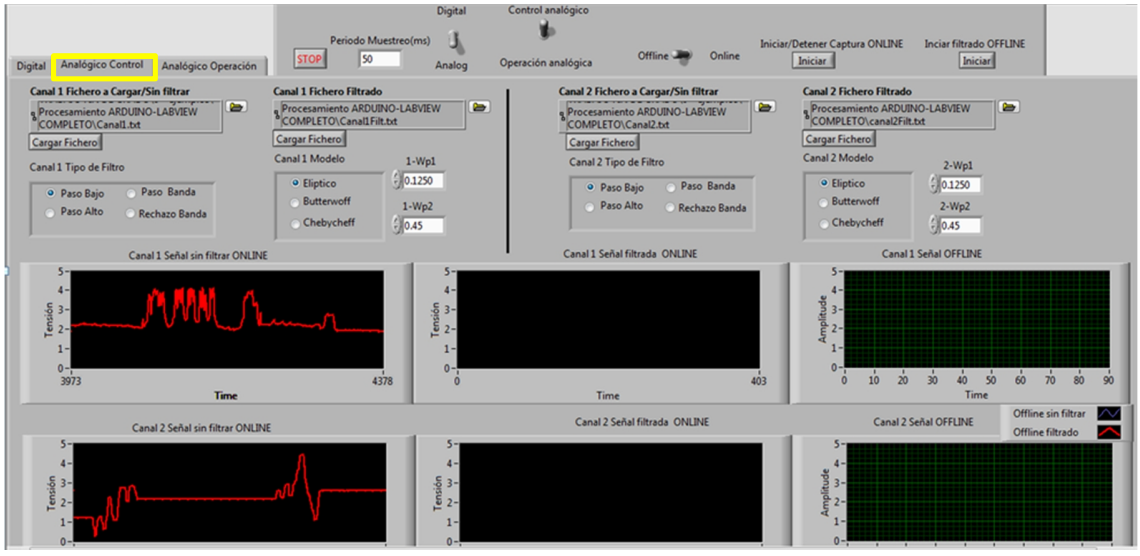


Figura 38: Pestaña 2 - Control analógico

En la parte derecha de la pestaña de Control analógico tenemos las opciones y parámetros homólogos que serán aplicados al canal 2. De las seis gráficas inferiores, las tres superiores serán usadas para el canal 1, mientras que las tres inferiores visualizarán datos del canal 2. La primera columna de gráficas, cuyo nombre es "sin filtrar ONLINE" estará continuamente adquiriendo datos solamente por estar en esta pestaña. Es la señal no filtrada.

Si activamos en la barra de control superior la palanca "ONLINE", empezará el filtrado de las señales, que serán visibles en la segunda columna de gráficas "Señal filtrada ONLINE", pero no estamos guardados los datos en ficheros. Para ello, hay que pulsar el botón de control de "Iniciar/Detener captura ONLINE". En este momento comenzará a escribirse sobre los 4 ficheros, otros de datos no filtrados y otros dos para los datos ya filtrados, 2 de cada tipo porque estamos procesando dos canales analógicos. Si se quisieran guardas datos sin llegar a filtrar la señal únicamente habrá que mover la tercera palanca de control a "Offline" e iniciar la captura de la misma manera. Para detener la captura en cualquiera de los casos se volverá a presionar el botón "Iniciar/Detener Captura ONLINE". Los resultados del filtrado Online se pueden ver en la figura 39.

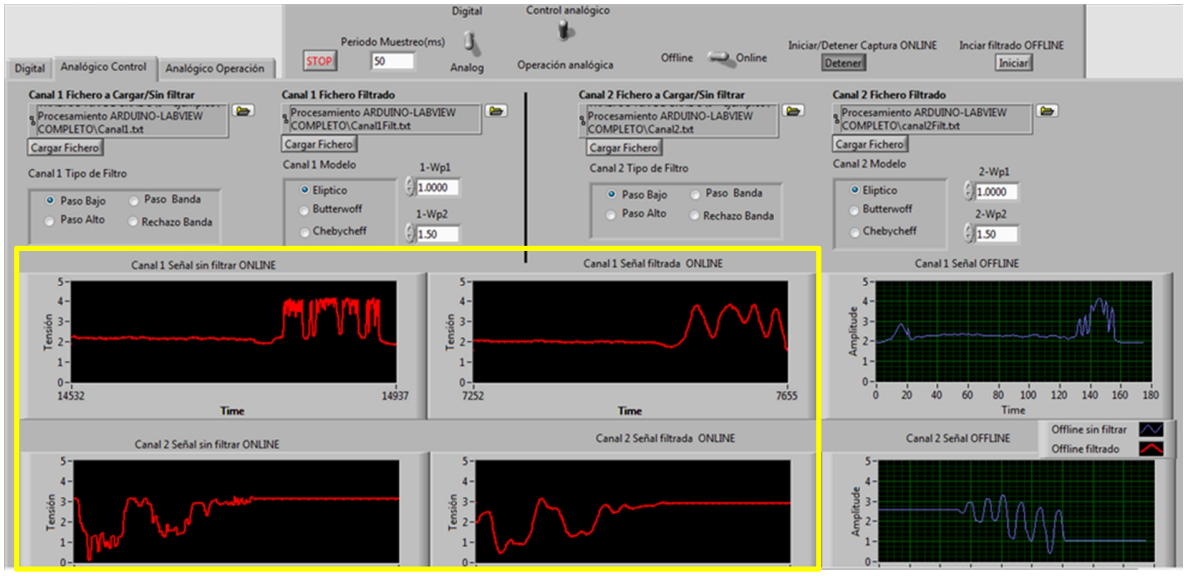


Figura 39: Filtrado online y captura de ficheros

Las últimas dos gráficas de la derecha son utilizadas para visualizar las señales filtradas y sin filtrar cuando se emplea el método offline. Para cargar cada señal desde fichero se pulsa sobre el botón "Cargar fichero" que se encuentra justo debajo de la dirección de fichero correspondiente. Las señales sin filtrar serán de color azul, mientras que las filtradas serán de color rojo. Una vez cargadas, el filtrado offline comenzara cuando el botón de control "Iniciar filtrado OFFLINE" sea pulsado. Las gráficas filtrada y sin filtrar se superpondrán en este par de gráficas. El resultado de esta operación se puede visualizar en la figura 40.

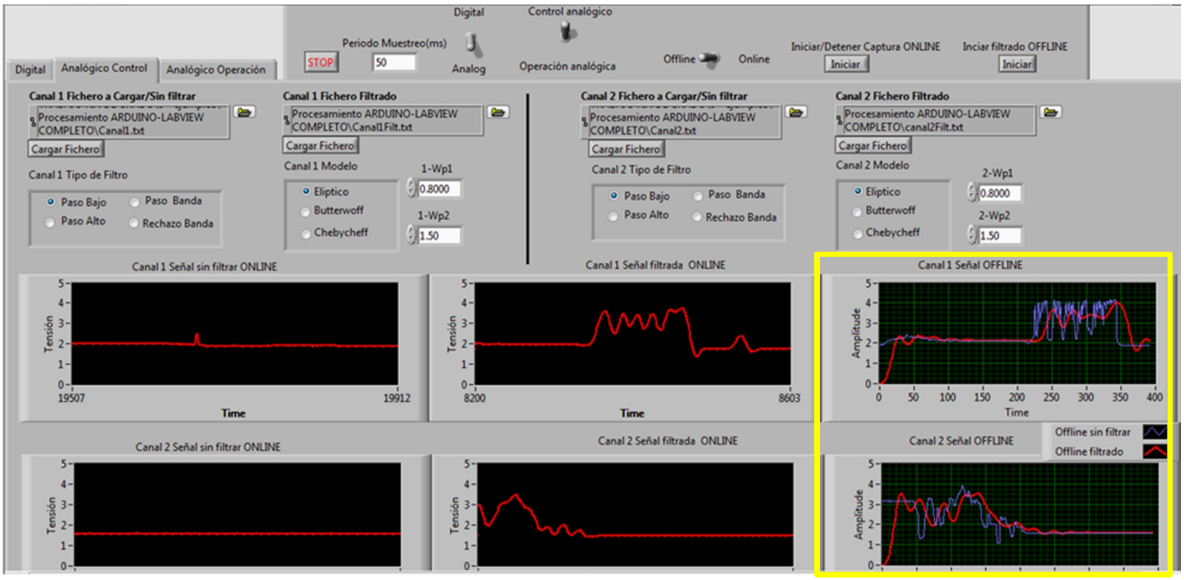


Figura 40: Filtrado offline

Por último la tercera y última pestaña es donde se encuentra la mayor diferencia entre esta aplicación y la implementada en Matlab. En la pestaña 3 “Analógica Operación”, a la que accedemos a través de la palanca de “Operación Analógica” vemos un diagrama de bloques de todas las etapas de procesamiento que se llevan a cabo para los canales 1 y 2, como muestra la figura 41.

De izquierda a derecha, se encuentra el selector de pin de Arduino que contiene la señal analógica a capturar. A continuación, se le pueden añadir 2 ruidos artificiales a cada canal, el primero de ellos es el “Ruido blanco”, que es un ruido aleatorio de una determinada Amplitud y de media 0. El segundo de los ruidos es una senoidal, de amplitud y frecuencia (Hz) personalizables.

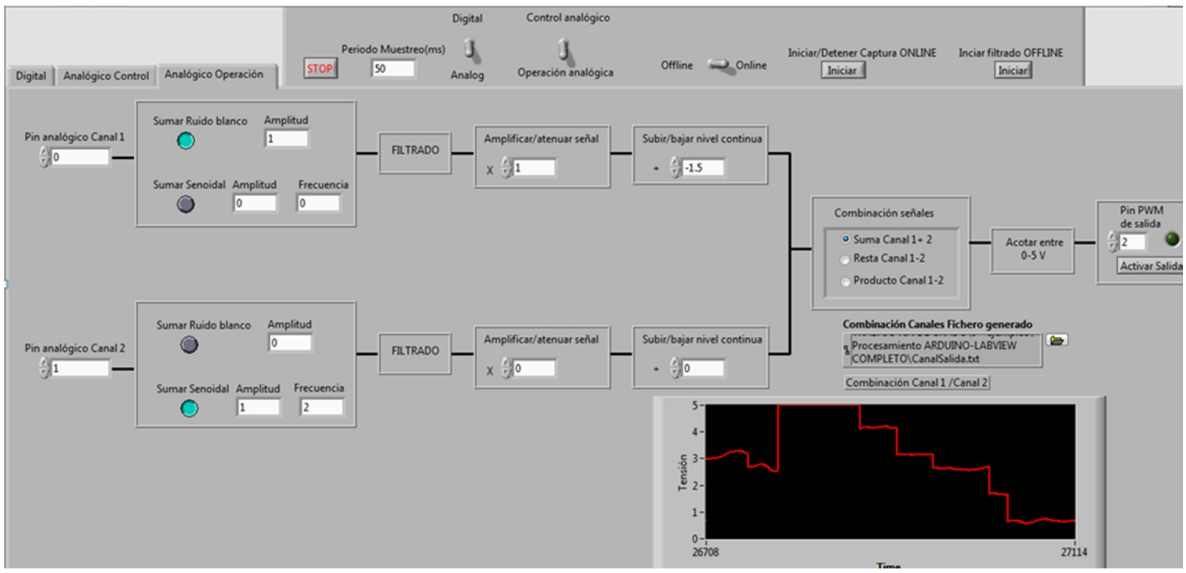


Figura 41: Pestaña 3 - Operaciones analógicas

Estos dos ruidos nos permiten comprobar si los filtros funcionan como el usuario desea, pues es posible añadir una senoidal de una determinada frecuencia y se puede verificar si esta componente senoidal añadida es eliminada o no dependiendo del filtro diseñado. Para añadir uno o los dos ruidos a cada canal sólo hay que presionar el botón de nombre “Sumar Ruido Blanco” o Sumar Senoidal”.

La siguiente operación que recibe la señal es el filtrado, programado anteriormente en la pestaña 2 “Control Analógico”, que puede realizarse o no dependiendo si la tercera palanca está en la posición “Online” (entonces si se filtrará) o “Offline” (por lo que la señal pasará a la siguiente operación sin filtrar).

Las dos siguientes operaciones son de multiplicar y sumar una constante a los valores adquirido en cada instante de tiempo. Ya se ha explicado anteriormente que esta operación resulta extremadamente útil si se necesita adecuar el valor de tensión a una aplicación concreta, sin modificar el perfil de la señal. Para ignorar esta operación, se debe poner como parámetro de amplificación/atenuación un 1, y un 0 en el parámetro de Subir/bajar el nivel de continua.

Ahora llega otra novedad de Labview, que es la combinación en el tiempo de dos canales analógicos ya procesados. En el bloque de “Combinación de señales” podemos escoger si sumar, restar o multiplicar para cada instante de tiempo los valores de ambos canales. El problema de esta operación es que una vez realizada, los valores de tensión pueden estar por encima de los 5V o por debajo de los 0V, por lo cual deberemos emplear el siguiente bloque que acotará la señal entre los valores máximos y mínimos con los que trabaja Arduino (entre 0 y 5 Voltios).

Una vez realizado esto en el penúltimo bloque, se tendrá una señal única señal procesada que podrá ser enviada a Arduino a través de un pin digital PWM. Para ello, se escogerá el pin de salida y se decidirá si queremos tener disponible la señal en dicho pin con el botón “Activar Salida”. Hecho esto, se activará el led de la derecha de este bloque como indicador de que estamos enviando la señal. Si se desea desactivar esta salid, solo se necesita pulsar nuevamente el botón “Activar Salida”, ahora etiquetado como “Desactivar Salida”.

Además se visualizará esta señal a lo largo del tiempo en la gráfica llamada “Combinación Cana1/Canal2 y se dispondrá de esta señal en un fichero de nombre editable en esta misma pestaña.

Las aplicaciones de esto último son inmensas, pues sería posible realimentar el microcontrolador Arduino con esta señal y realizar algún tipo de control automático. Incluso se podría implementar algún controlador PID simple, haciendo una analogía entre las funciones de transferencia del controlador y de los filtros empleados y comparando sus parámetros.

Con esto concluye la explicación sobre el funcionamiento de esta aplicación. Ahora se presentarán los detalles de la programación realizada en Labview en un diagrama de flujo, que incluye las funciones y operaciones más destacadas del diagrama de bloques construido. Este diagrama de flujo se muestra en la figura 42:

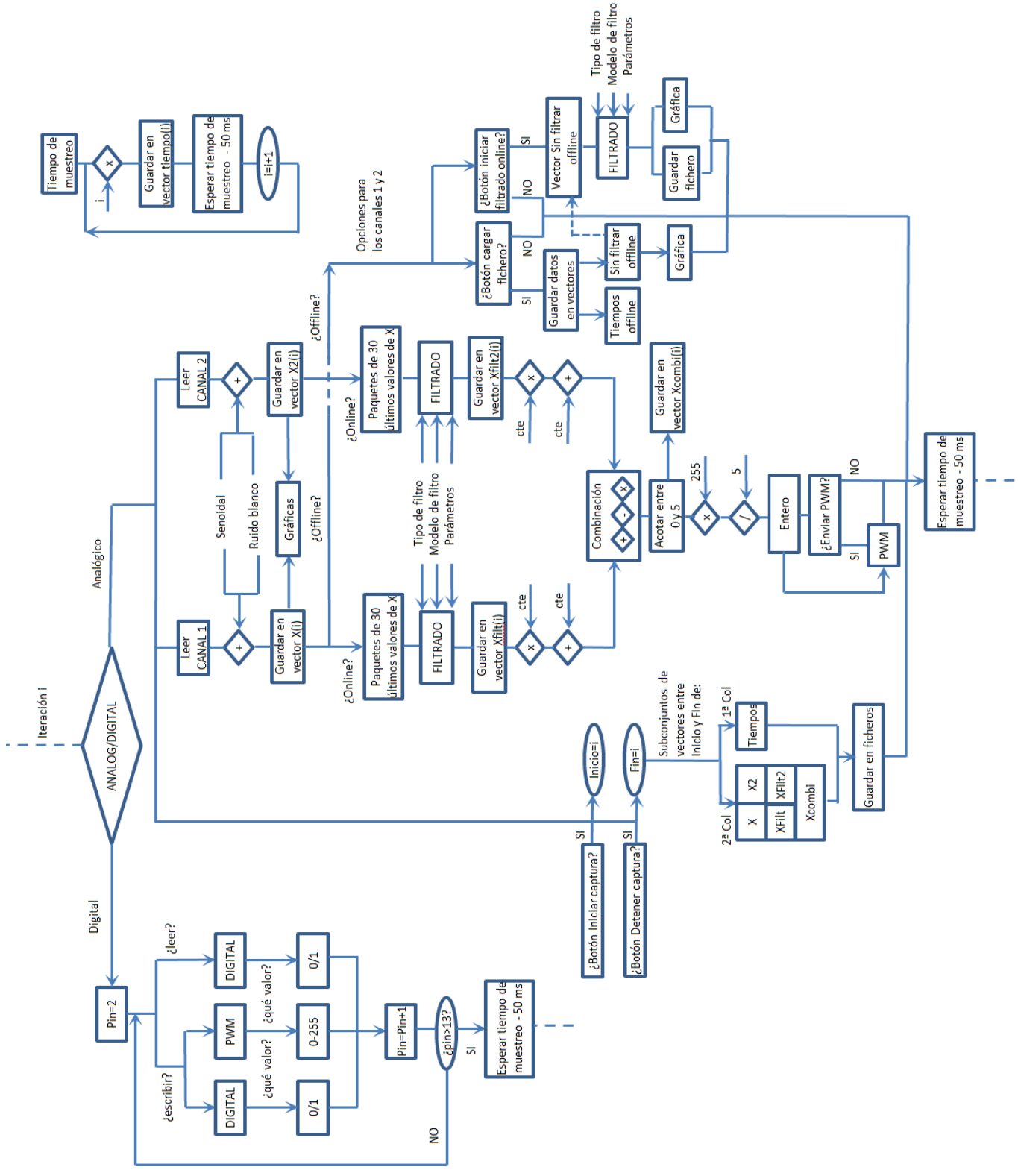


Figura 42: Diagrama de flujo de la programación en Labview

Una vez visto el diagrama de flujo de la aplicación de Labview, se escogerán algunos fragmentos de él para analizar con más detalle y asociar a las partes del código en diagrama de bloques correspondiente.

En primer lugar, se ha decidido ampliar la herramienta que más diferencia la aplicación de Labview frente a la desarrollada en Matlab, que es el procesamiento paralelo de dos canales, su combinación y posterior envío por un pin PWM. Los diagramas parciales que realizan esta operación son los de la figura 43. En la parte izquierda, se ve el flujo de la información del programa, mientras que la figura de la derecha muestra el código en diagrama de bloques que ejecuta todas estas acciones. La información de cada canal se suma y multiplica por un parámetro modificable, para luego llegar al módulo de combinación con su selector de operación (suma, resta o producto). La salida es acotada y transformada a valores de PWM enteros entre 0 y 255 para finalmente ser enviada a un pin digital del microcontrolador.

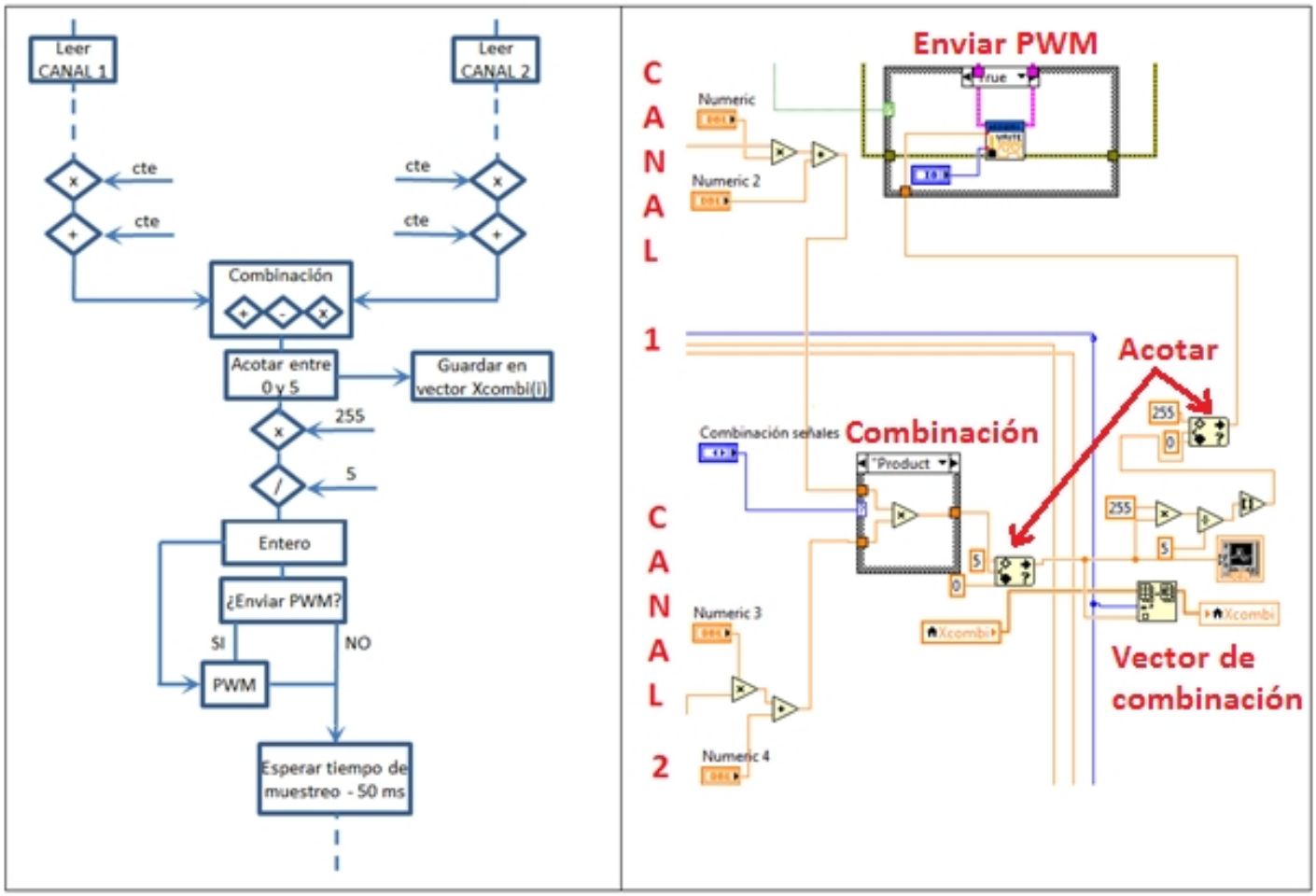


Figura 43: Diagrama parcial de Labview. Combinación de señales.

En segundo y último lugar, se presenta en la figura 44 la programación del filtrado offline incluyendo sus dos vertientes, que son la carga de ficheros de datos y el filtrado offline propiamente dicho. La figura superior derecha realiza la lectura del fichero guardando sus datos en dos vectores, uno por cada columna del fichero, además de obtener su periodo de muestreo, dato que será un parámetro de entrada del filtro offline.

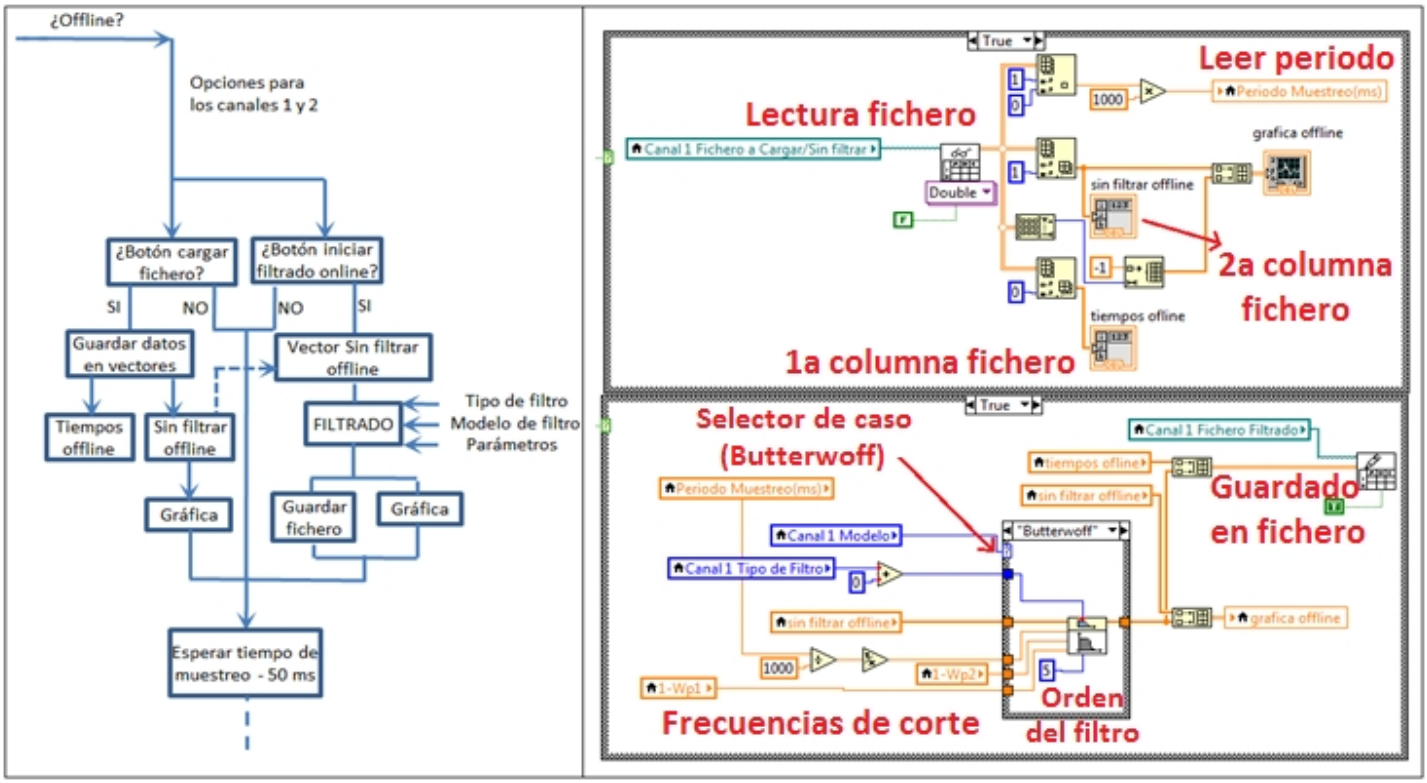


Figura 44: Diagrama parcial de Labview. Filtrado offline.

La figura inferior derecha implementa el filtrado offline. El módulo de filtrado central requiere de la entrada de la seña como vector, las frecuencias de corte y la frecuencia de muestreo empleada, además del tipo de filtro y un selector de modelo de filtro. La salida es directamente el vector filtrado, por lo que se guardará directamente en fichero junto a la columna de tiempos leía del fichero inicial. Nótese las diferencias de este módulo de filtrado offline con el de filtrado online de Labview presentado en la figura 25 del apartado 3.4.

Vistos un par de ejemplos de la programación en Labview, sólo queda recordar que el formato de ficheros que emplea tanto Labview como Matlab es el mismo y son perfectamente compatibles los datos recogidos y analizados con cada uno de los programas. Esto permite poder emplear las dos aplicaciones en paralelo de manera que se utilice de cada una aquellas herramientas que se consideren más útiles.

4.5-PROCESSING, UNA ALTERNATIVA FRENTE A LOS DISEÑOS ANTERIORES

Una vez vistas todas las funciones y herramientas presentes en las aplicaciones de Labview y Matlab, se ha dejado de lado una parte muy importante que es la programación del microcontrolador con Arduino IDE. Se recuerda que tanto Labview y Matlab funcionaban a través de Firmatas y esta parte ya estaba solventada. Por ello, se buscará realizar una aplicación mucho más simple que las anteriores pero donde la programación se centrará básicamente en la comunicación a través del puerto Serie y tengamos libertad absoluta sobre el programa alojado en el microcontrolador. El control de instrucciones será realizado desde Processing mediante la creación de una interfaz gráfica, mostrada en la figura 45,

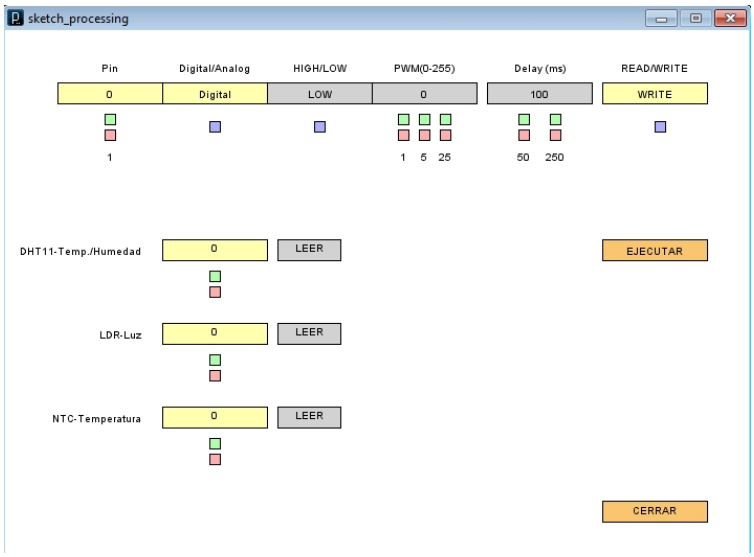


Figura 45: Interfaz de captura y control con Processing

Esta interfaz será controlada toda a través de botones. En la parte superior están los campos de “Pin” a controlar, “Digital/Analog” para decidir si se desea controlar un pin digital o analógico, “HIGH/LOW” para escribir el estado de un pin digital “PWM” para escribir un valor PWM comprendido entre 0 y 255, “Delay” o tiempo de espera entre medidas y “READ/WRITE” para escoger qué operación realizar entre escritura o lectura.

Es posible cambiar la selección de estas opciones con los botones cuadrados que tienen debajo de ellos. Los campos que sólo tengas 2 opciones posibles se podrán alternar con los botones azules, cambiando a su valor opuesto al presionarlo (cambio de Write a Read, o de HIGH a LOW). Los valores numéricos como el pin, la PWM o el tiempo de espera dispondrán de dos tipos de botones. Los verdes, que incrementan el valor de dicho campo un cierto número de unidades enteras (visible debajo del botón) y los botones rojos que disminuyen su valor en las unidades que representa dicho botón. Se programará el hecho de que el número de pin no sea inferior a 0 o que la PWM quede acotada entre 0 y 255 para que no haya problemas posteriores.

Siguiendo el mismo método de selección de pines, se crearán campos para realizar las medidas de los sensores ya presentados. Para ello se creará un campo que contenga el número de pin donde esté instalado cada sensor, con sus botones de incremento y decremento de cada pin, así como un botón adicional a su derecha con la etiqueta LEER (NO LEER una vez sea pulsado).

Finalmente, se incluirán dos botones naranjas de control de la aplicación. El primero de ellos, "EJECUTAR", enviará a través del puerto serie una cadena de parámetros de control que permite al microcontrolador trabajar como receptor o emisor de datos según se desee. El botón de "CERRAR" cierra la comunicación del puerto serie por la parte del ordenador, cerrando además la ventana gráfica de la aplicación.

Centrándonos en la cadena que se envía tras pulsar "EJECUTAR", ésta compuesta por la información de los siguientes campos, en forma de valores enteros: "READ/WRITE" (Se enviará un 0 para Read y un 1 para Write), "Pin", "Digital/Analog" (un 0 representa una petición digital y un 1 una operación sobre un pin analógico), "PWM", el estado de los botones de "LEER" de los sensores (enviando un 1 si se desea realizar la lectura y un 0 si no), los tres pines de los sensores, y el tiempo de espera "Delay".

Cuando esta cadena se envíe por el puerto Serie, el microcontrolador la recibirá y guardará respetando el orden de llegada en unas variables de control creadas para recibir estos datos. Una vez recibidos, el microcontrolador realizará las instrucciones programadas, ejecutándose únicamente aquellas que se han pedido mediante una serie de comparaciones sucesivas con las variables de control. Hecho esto, se procederá a la escritura de los pines digitales una única vez si esta ha sido la orden o se enviarán periódicamente las medidas pedidas a través del puerto Serie de vuelta a Processing. Para enviar las medidas se enviarán los valores acotados entre 0 y 255 (pues estos son los valores que el puerto Serie permite enviar en forma numérica) siguiendo el siguiente orden: lectura del pin del campo "PIN", dos lecturas del sensor DHT11 de humedad y temperatura, y las lecturas de los pines de la LDR y NTC.

Finalmente, se visualizarán las medidas en sus correspondientes cuadros después de realizar los cálculos de conversión pertinentes para cada una de las medidas. El resultado es el que se muestra en la figura 46.

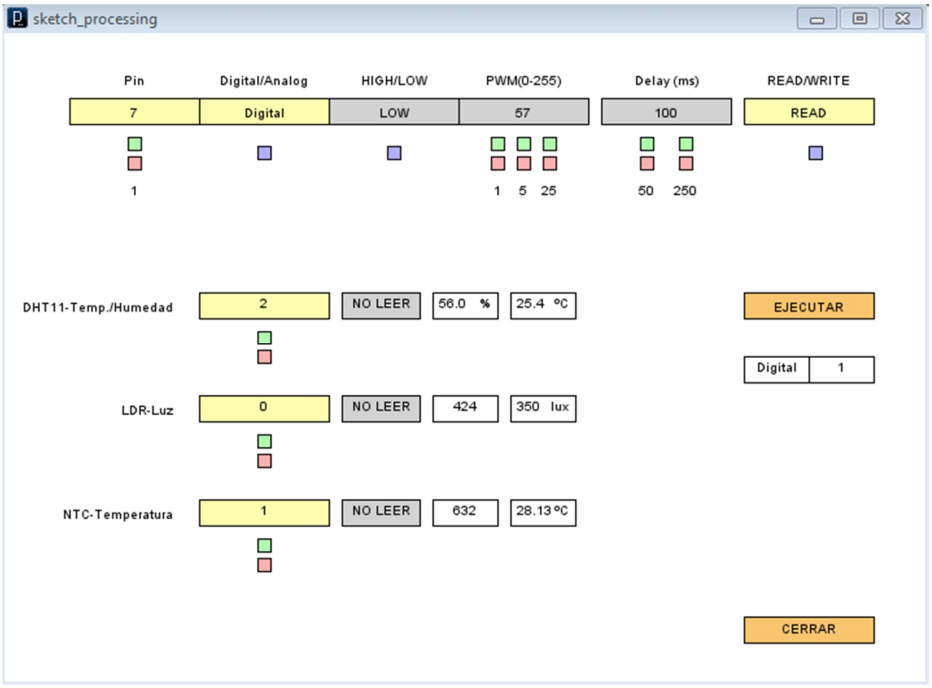


Figura 46: Lectura de medidas de los sensores.

Una vez visto el funcionamiento de esta aplicación, se va a mostrar la parte más importante del código desarrollado, que es la interacción entre el programa cargado en Arduino y Processing. Para evitar interferencias, en todo momento uno de los dos debe ser el que envíe datos mientras el otro debe estar preparado para recibirlas. Además, para cada uno de los casos, la cadena de datos enviada debe contener un número de datos constante, que serán leídos al completo por el otro dispositivo. Esto se conseguirá cuando se ejecuten los códigos de la tabla siguiente simultáneamente.

ACCIÓN	Ejecutar en ARDUINO	Ejecutar en PROCESSING

Processing envía órdenes
 	
Arduino recibe cadena de datos
	`if(Serial.available()>0){`
	`  rw=Serial.read();`
	`  pin=Serial.read();`
	`  ad=Serial.read();`
	`  digital=Serial.read();`
	`  pwm=Serial.read();`
	`  sensor1=Serial.read();`
	`  sensor2=Serial.read();`
	`  sensor3=Serial.read();`
	`  pin1=Serial.read();`
	`  pin2=Serial.read();`
	`  pin3=Serial.read();`
	`while(Serial.available()>0){`
	`  espera2+=Serial.read();}`
	`  espera=espera2;`
	`  espera2=0;}`
		`puerto.write(rw);  rw_act=rw;`
		`puerto.write(pin);`
		`puerto.write(ad);  ad_act=ad;`
		`puerto.write(digital);`
		`puerto.write(pwm);`
		`puerto.write(sensor1);`
		`puerto.write(sensor2);`
		`puerto.write(sensor3);`
		`puerto.write(pin1);`
		`puerto.write(pin2);`
		`puerto.write(pin3);`
		
		`sensor1_act=sensor1;`
		`sensor2_act=sensor2;`
		`sensor3_act=sensor3;`

Acción	Arduino	Processing

Arduino envía lecturas
 	
Processing recibe cadena de datos
	`if(rw==1){`
	`  Serial.write(dato);`
	`}`
	
	`if(sensor1==1){`
	`  Serial.write(h);`
	`  Serial.write(t);`
	`}`
	
	`if(sensor2==1){`
	`  dato2=analogRead(pin2)/4;`
	`  Serial.write(dato2);`
	`}`
	
	`if(sensor3==1){`
	`  dato3=analogRead(pin3)/4;`
	`  Serial.write(dato3);`
	`}`
		`while (puerto.available()>0){`
		`if (ad_act==0 && rw_act==1){`
		`  lectura_pin=puerto.read();`
		`}`
		`if (ad_act==1 && rw_act==1){`
		`  lectura_pin1=puerto.read()*4;`
		`}`
		`if (sensor1_act==1){`
		`  h=puerto.read();`
		`  t=puerto.read();`
		`}`
		`if (sensor2_act==1){`
		`  lectura_pin2=puerto.read()*4;`
		`}`
		`if (sensor3_act==1){`
		`  lectura_pin3=puerto.read()*4;`
		`}`
		`}`

Tabla 1: Instrucciones de comunicación entre Arduino y Processing

Con este ejemplo se pretende presentar Processing como un método viable de diseño de una aplicación de captura y procesamiento de señales, siendo posible conseguir una comunicación estable y sin interferencias en el puerto Serie. Además, Processing dispone de librerías de filtrado que simplificaría la programación del procesamiento, aunque de no ser así se podrían programar las ecuaciones del filtro como se hizo en uno de los programas auxiliares de Matlab.

Sin embargo, el verdadero potencial de Processing se encuentra en la potente programación gráfica que tiene y la libertad del programador para realizar cualquier diseño que tenga en cuenta los movimientos, clics y arrastres del ratón. Por ello se ha pensado la idea teórica de la creación de una aplicación que a su vez permita el propio diseño de captura y procesamiento de señales, basado en la programación en diagrama de bloques de Labview,

La idea sería la siguiente. Tener un menú de objetos que desempeñen una función en el procesamiento, como puede ser la captura de una señal analógica o distintos tipos de filtros con su correspondiente icono. A continuación, estos objetos podrían ser arrastrados y colocados por el usuario en un espacio de trabajo reservado para disponer estos objetos de manera ordenadas y conectarlos posteriormente con cables (o líneas). Finalmente, se emplearía algún objeto de visualización creado para graficar los resultados. Todo esto controlado por una barra de menús desplegables con campos como Archivo, Editar o Herramientas que controlen el inicio y fin de la adquisición entre otras cosas.

Esta idea es extremadamente pretenciosa y debido al gran esfuerzo y trabajo que supondría desarrollarla, se dejará pendiente como trabajo futuro o ampliación de este proyecto.

4.6- COMPARATIVA ENTRE SOFTWARES EMPLEADOS

Llegado a este punto del proyecto que ya se han realizado tres propuestas de captura y procesamiento de señales con distintos softwares, estamos en disposición de analizar los puntos fuertes e inconvenientes de cada uno de ellos que sea han ido encontrando durante el transcurso de la programación.

Comenzando con las GUI de Matlab, un aspecto importante es que la creación de objetos es muy intuitiva, ya que una vez dispuestos los objetos sobre la ventana, el programa crea automáticamente las llamadas de los objetos es un programa *.m*. Además, es posible modificar cualquier propiedad de estos objetos mediante comandos, como es caso de los leds digitales que varían su color gradualmente según el nivel de PWM escrito en los pines. O en caso de las gráficas, que pueden ser empleadas para representar varios tipos de información diferente variando sin ningún problema las escalas y rangos de los ejes o las leyendas de estos. Todo esto, sumado a que puede conectarse con Arduino tanto por Firmata como con comandos a través del puerto serie hace de Matlab un software muy completo para esta aplicación. Como gran inconveniente, destacar la imposibilidad de seguir varios hilos de ejecución, no pudiéndose ejecutar más de un bucle en paralelo al mismo tiempo.

Siguiendo con Labview, destacar su particular lenguaje de programación, que debe dominarse para poder desarrollar una aplicación con él. Un lenguaje de programación que con sus diagramas de bloques sí permite la ejecución en paralelo de varios bucles. Además, dispone de una gran cantidad de objetos como indicadores, controles y aspectos visuales que permiten un amplio abanico de posibilidades de diseño del panel frontal del programa. Un punto negativo a este software es que el único modo de comunicación con Arduino es mediante Firmata, por lo que no se podrá aprovechar todo el potencial del microcontrolador complementándolo con Arduino IDE, así como será muy difícil recibir medidas de sensores digitales como el DHT11.

En cuanto a Processing, es un software que requiere de la programación completa de todos sus objetos y de la mayoría de sus funciones, es por esto que tiene una de las interfaces gráficas más personalizables. Además se puede conectar con Arduino tanto por Firmata como por control directo del puerto Serie, siendo este último el más potente de los dos métodos como se ha dejado ver en el punto anterior. Una desventaja de programar en Processing es nuevamente la imposibilidad de tener a la vez más de un hilo de ejecución del programa, lo que dificultará la programación de las aplicaciones que así lo requieran.

En conclusión, queda claro que cualquiera de estos tres software puede emplearse para desarrollar una aplicación de captura y procesamiento que cumpla con las necesidades enunciadas en el inicio de este apartado. Por lo tanto, depende del programador escoger la opción que prefiera según la libertad que quiera tener a la hora de programar, siendo Labview el más cerrado y restrictivo y Processing el que ofrece más libertad y el más editable en cuando a interfaz y programación.

5- MEJORANDO LA CONECTIVIDAD

Hasta este momento toda la comunicación establecida entre Arduino MEGA 2560 y el ordenador ha sido a través de Cable USB y los puertos Serie del microcontrolador y del PC. Este método no es el único existente para lograr esa comunicación, ya que se pueden lograr conexiones inalámbricas a través de Bluetooth o Internet, y de esta manera, aumentar la distancia de separación entre el microcontrolador y el ordenador (u otros dispositivos) y evitar el empleo de cables físicos que enlacen ambos dispositivos.

5.1- EMPLEO DEL MÓDULO BLUETOOTH HC-06

El módulo Bluetooth HC-06 es una de las soluciones más útiles y económicas para lograr una conexión vía Bluetooth con un microcontrolador como Arduino. El esquema de conexión general entre los dispositivos implicados es el mostrado en la figura 47. El módulo se conectará mediante cableado pin a pin con Arduino MEGA 2560, con lo que el microcontrolador podrá recibir y enviar información a través de él una vez que el HC-06 esté vinculado con otro dispositivo que emplee tecnología Bluetooth. Con esto, se podrá establecer una comunicación inalámbrica de una distancia entre 5 y 10 entre el módulo y el ordenador o Tablet.

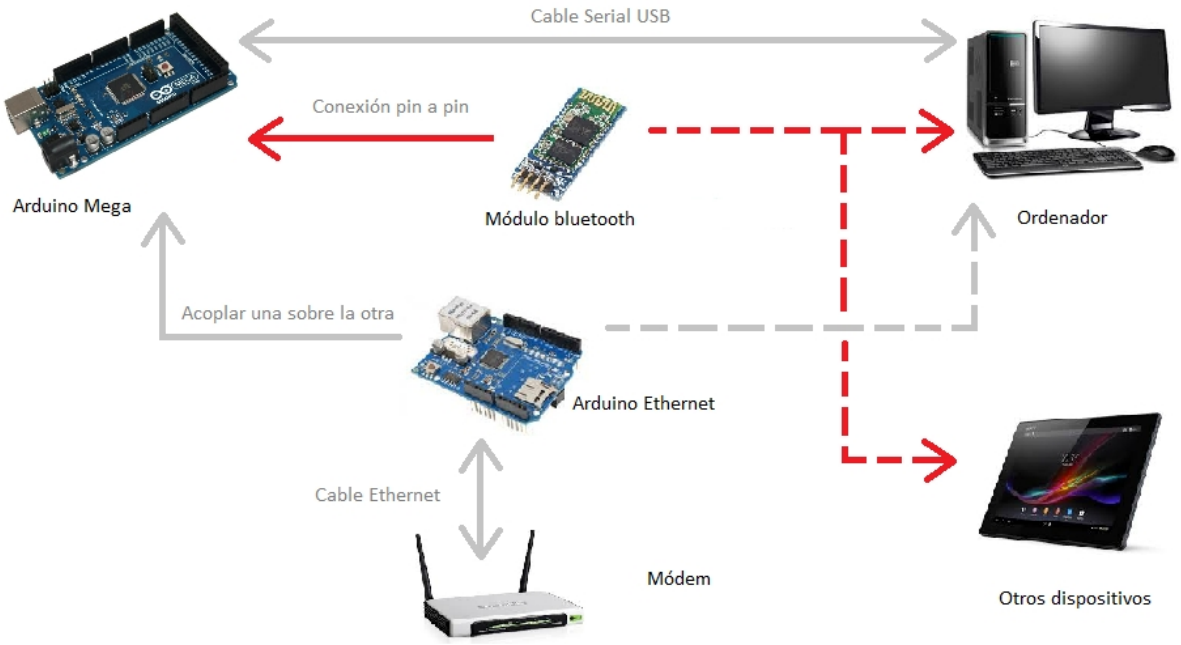


Figura 47: Conexión Arduino-PC / Arduino- Tablet vía Bluetooth

El primer aspecto que se tratará es cómo cablear esté modulo Bluetooth. El HC-06 dispone de 4 pines, de los cuales los dos primeros de la izquierda sirven para alimentar el módulo. Como muestra la figura 48, el primero de ellos, etiquetado como *VCC*, se pone a una tensión de tensión de 5V (cable rojo) y a su derecha, el pin *GND* se conecta a 0V (cable negro). Los dos siguientes pines, nombrados como *TXD* y *RXD*, permiten enviar y recibir información a través de un puerto Serial (formado por una pareja de pines del microcontrolador) y deben ser conectados con los correspondientes pines *TX_* y *RX_* del microcontrolador de manera cruzada. Es decir, si la comunicación se realiza por el puerto Serial 3, las conexiones a realizar serían enlazar el *TXD* del módulo con el *RX3* (pin 15, cable verde) de Arduino y el *RXD* con el *TX3* (pin 14, cable marrón) de la placa Arduino.

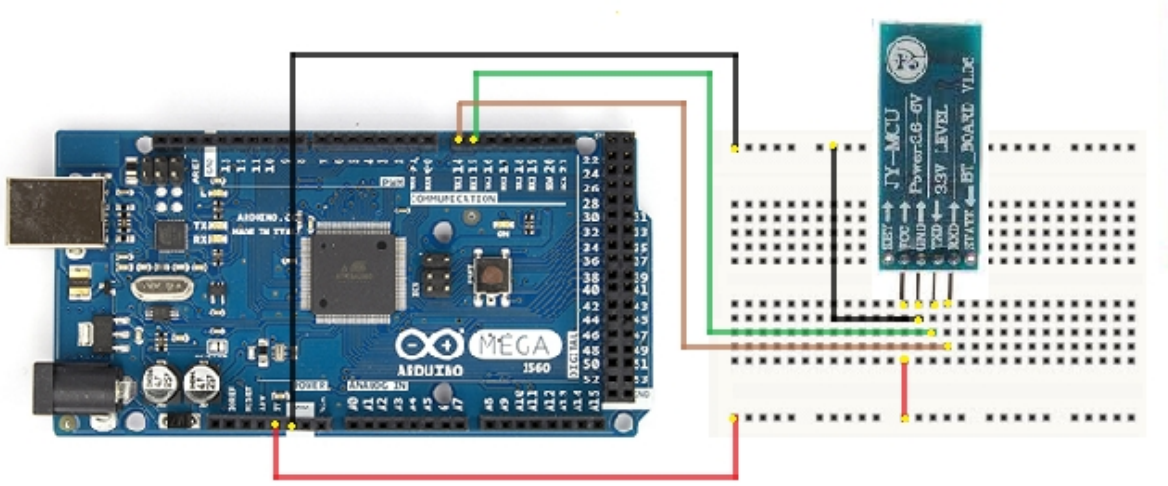


Figura 48: Montaje del módulo Bluetooth HC-06

Aunque en este trabajo se ha decidido emplear el puerto Serial 3, se puede usar cualquiera de los puertos que el microcontrolador tiene preparado, siempre que se cumpla la conexión cruzada de pines a la hora de conectarlo. Aunque hay una particularidad si se decide usar el puerto Serial 0, formado por los pines digitales 0 y 1, ya que es usado para cargar los programas sobre Arduino y durante esta operación el módulo Bluetooth no debe estar conectado, o habrá interferencias sobre este puerto.

Una vez realizadas las conexiones anteriores el siguiente paso será la configuración del módulo, para ello el módulo permite recibir ciertas instrucciones por el puerto Serie al que esté conectado en forma de líneas de texto. Estos son los llamados comandos AT del HC-06, que permiten configurar ciertos parámetros del módulo como su nombre de búsqueda, la tasa de baudios de comunicación o su clave de sincronización con otros dispositivos, y los que usaremos son los siguientes:

- AT+NAMEHC-06, que renombrará el dispositivo como HC-06.
- AT+BAUD4, que pondrá al módulo a funcionar a 9600 baudios (el número 4 corresponde a 9600).
- AT+PIN1234, que modificará la clave de sincronización a 1234.

Para usarlos se deberá crear un programa en Arduino IDE que escriba en el puerto Serial 3 estos comandos con la instrucción *Serial3.print ("comando AT")*. Se ejecutan todos los comandos una sola vez, y el HC-06 quedará finalmente configurado.

Hecho esto es hora de configurar el adaptador Bluetooth de nuestro ordenador. Para ello, conectaremos el adaptador LM054 a un puerto USB e instalaremos un gestor de Bluetooth para crear un puerto Serial virtual, llamado TOSHIBA Bluetooth Stack 8.00.12. Tras instalar esta aplicación y los Drivers del adaptador Bluetooth se podrá buscar el módulo HC-06 desde la consola de Toshiba, se seleccionará y se vinculará tras introducir la clave anteriormente elegida (1234). En los detalles de la conexión establecida se verá a través de que puerto virtual del PC se establecerá la comunicación, como muestra la figura 49.

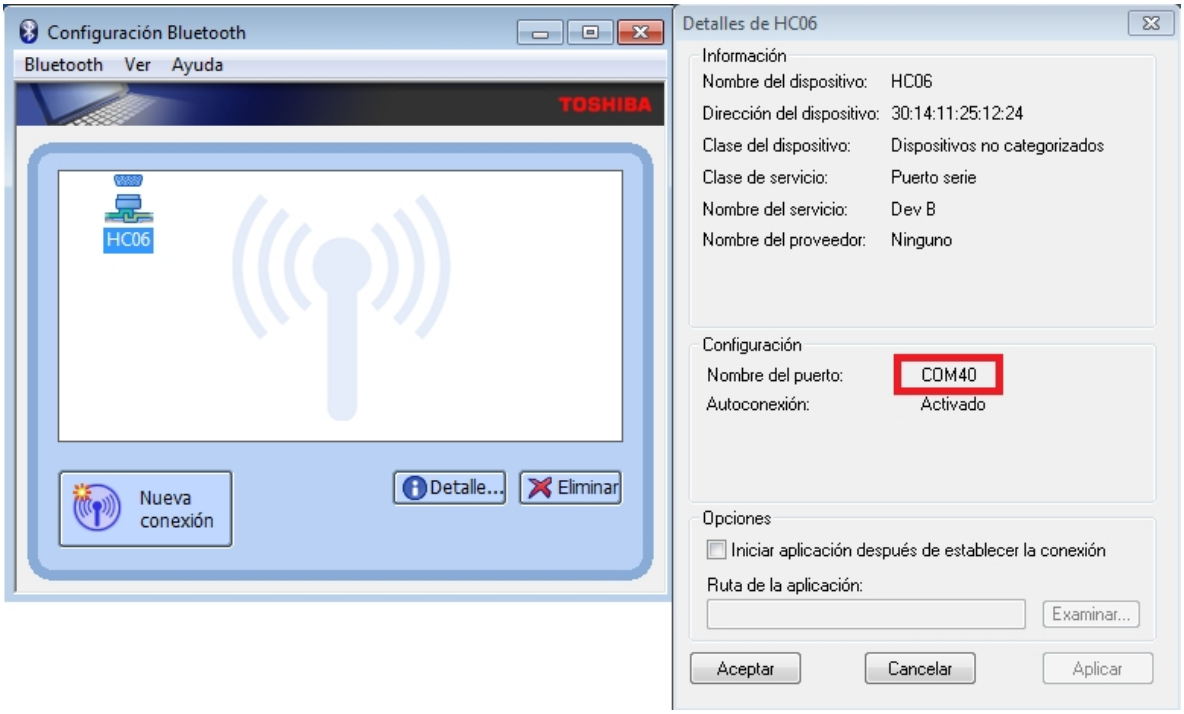


Figura 49: Detalle del gestor de Toshiba y el puerto virtual creado.

Una vez estén vinculados El HC-06 y el ordenador, todos los programas creados desde Processing podrán funcionar a través de la comunicación Bluetooth con Arduino. Para ello, sólo habrá que realizar dos pequeños cambios, que son el puerto abierto desde Processing (ya no será el *COM7*, sino el *COM40*), y las instrucciones de los programas correspondientes de Arduino IDE relativos al puerto serie utilizado (ahora emplearemos el Serial3 en lugar del Serial por defecto).

Todo esto no sería posible sin sustituir la alimentación del microcontrolador a través del cable USB por una fuente de alimentación de 9 voltios y 2 amperios. Esta fuente de alimentación será necesaria para poner en marcha las aplicaciones desarrolladas en los siguientes apartados.

Como ya se ha logrado la conexión inalámbrica entre Arduino MEGA y Processing, se intentó conseguir un resultado similar con los programas diseñados en Matlab y Labview. Pero el principal problema que nos encontramos es que para su funcionamiento necesitan una Firmata cargada en el microcontrolador, que está preparada para trabajar desde el puerto Serial por defecto (el del cable USB). Para solucionar esto, se modificaron las Firmatas para que trabaran por el puerto Serial 3, que es donde está conectado el HC-06, y sustituir el puerto COM7 declarado en Matlab y Labview por el nuevo COM40. Hecho esto, se probó a ejecutar los programas, sin embargo dieron problemas en el reconocimiento del puerto COM40, por lo que se concluyó que se necesitaban cambios adicionales en los programas para lograr su correcto funcionamiento con la tecnología Bluetooth, tarea que se dejará pendiente para trabajos posteriores.

Una vez establecida la conexión con el ordenador, se buscará conectar Arduino con un dispositivo Android a través de Bluetooth. Para ello, se utilizará una aplicación que puede enviar y recibir datos de otros dispositivos llamada Bluetooth Terminal. Su interfaz es una consola de comandos donde es posible enviar líneas de texto y visualizar todo tipo de información. Su conexión con el módulo HC-06 es muy simple, pues sólo se debe seleccionar éste entre los dispositivos encontrados e introducir su clave (1234).

En la Figura 50 se presenta el aspecto que tiene este programa, donde las líneas en rojo son las enviadas por comandos y las líneas azules son las recibidas desde Arduino, en el que se cargará un programa que sea capaz de descifrar e interpretar las instrucciones enviadas.

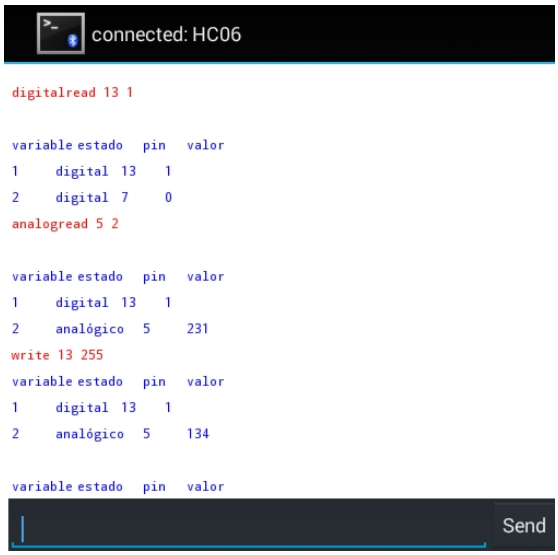


Figura 50: Captura de Bluetooth Terminal de Android.

La idea con esta aplicación es poder tener control sobre las órdenes ejecutadas por el microcontrolador y visualizar información procedente de Arduino. En el punto 5.4 se procederá a programar una aplicación en Arduino IDE controlada totalmente desde Bluetooth Terminal donde se emplee además la tarjeta Arduino Ethernet Shield.

5.2- USO DE ARDUINO ETHERNET SHIELD Y TARJETA SD

Arduino Ethernet Shield (el modelo REV3 con POE) es una tarjeta preparada para ser integrada en Arduino MEGA 2560 y poder transmitir y recibir datos en una red local de Ethernet. Para ello, se acoplará la tarjeta Ethernet encima de Arduino MEGA haciendo coincidir los pines correspondientes y esta tarjeta se conectará al módem a través de un cable Ethernet. Las conexiones generales a realizar se muestran en la figura 51.

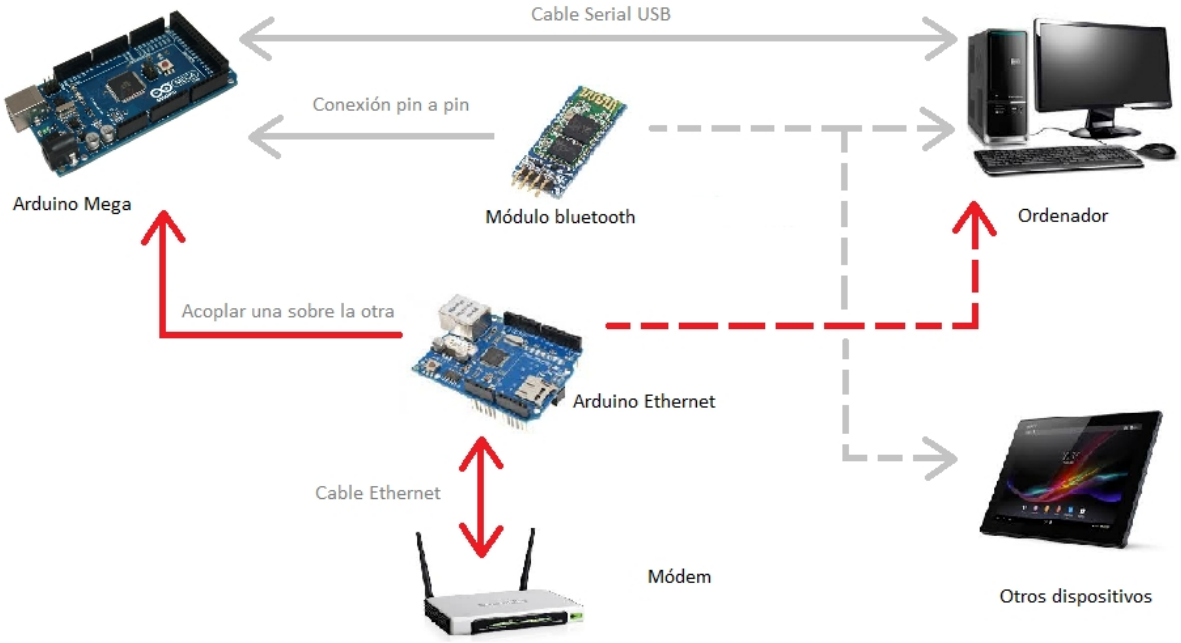


Figura 51: Conexión Arduino-PC mediante conexión inalámbrica Ethernet

Las aplicaciones que tiene Arduino Ethernet Shield son muy numerosas, como la creación de un servidor web, adoptar el rol de un cliente web o un cliente Telnet, un protocolo de comunicación remota entre máquinas. Además, dispone de una ranura de tarjetas SD, que podrá ser empleada para guardar ficheros de datos con el mismo formato que los ficheros empleados en las aplicaciones de Matlab y Labview.

La programación se realizará desde Arduino IDE, ya que dispone de la librería *Ethernet.h*, que contiene funciones con las que se puede controlar Arduino Ethernet Shield de manera sencilla. Hay varios ejemplos de programas de utilización de esta tarjeta en la página oficial de Arduino, que se tomarán de referencia para conocer el uso de esta librería y adecuarlos a las necesidades propias.

Arduino Ethernet Shield tiene un código de identificación que lo distingue del resto de dispositivos conectados a la red local y que deben ser definidos en el programa, que es la MAC, formada por 12 números hexadecimales. Además, se debe se debe elegir una dirección IP en la que transmitir y recibir la información, que debe un número no muy alejado de la propia dirección del módem (192,168,0,1), por eso se empleará la IP 192,168,0,15.

Hecho esto, en el *void setup* se inicializará tanto los puertos Serial empleados como la tarjeta Ethernet con la MAC y la IP elegida. En caso de dejar vacío el campo de la IP, se le asignará automáticamente una IP sin uso próxima a la del módem. En el *void loop* se ejecutarán las instrucciones de peticiones de comunicación entre servidor y cliente. Por ejemplo, la tarjeta Ethernet tomará en papel de servidor, y el cliente será el navegador en el que se busque la IP de la tarjeta, definida anteriormente. Siempre que haya un cliente conectado, Arduino enviará los datos programados en un formato de página *html*, que serán refrescados cada cierto intervalo de tiempo, que para esta aplicación será de 5 segundos. Para que esto funcione, el ordenador desde donde se lance el cliente deberá estar conectado al mismo módem que Arduino.

En el ejemplo anterior, el sentido de la comunicación es unidireccional desde Arduino al navegador del ordenador, por tanto, si se quiere trabajar en sentido inverso, es necesario hacer de Arduino un cliente web que reciba los datos de un servidor alojado en una página como Google, cargando en el microcontrolador el ejemplo adecuado modificado de acuerdo a nuestras necesidades.

Sin embargo, con el método de servidor/cliente web no es posible integrar el empleo de Ethernet con las aplicaciones desarrolladas en Processing, incluso en Matlab o Labview. Para ello, se necesita una comunicación bidireccional, que se podría conseguir mediante el empleo de la comunicación Telnet. Para ello, se debería ejecutar un servidor Telnet desde un programa de Processing y, con alguna aplicación de consola para gestionar comunicaciones Telnet como HyperTerminal o TeraTerm, poder enviar y recibir datos desde el ordenador a Arduino. Esto último no se ha logrado que funcione correctamente, por lo tanto se dejará como tarea pendiente para trabajos futuros.

Lo que sí tiene una aplicación sencilla y directamente relacionada con los diseños de Matlab y Labview, es usar la tarjeta Ethernet como almacenamiento de datos en ficheros de texto gracias a la ranura de Ethernet Shield que permite insertar tarjetas SD. El manejo de ficheros como su apertura, cierre, escritura o borrado de algún fichero puede ser gestionado cómodamente desde Arduino gracias a la librería *SD*.h. Para esto no es necesario que la tarjeta Ethernet esté conectada al módem, aunque si lo estuviera se puede combinar el empleo de ficheros como la visualización de datos en un servidor web, aplicación que ha sido programada y explicada en el siguiente apartado.

5.3- CAPTURA DE SEÑALES Y VISUALIZACIÓN EN UN SERVIDOR WEB

Una vez vistos diferentes usos de Arduino con las comunicaciones por Bluetooth y Ethernet, es hora de desarrollar una aplicación que tenga relación con el tema principal del proyecto. El objetivo de esta aplicación es combinar las tecnologías inalámbricas ya vistas para controlar la captura de señales mediante una serie de comandos desde Bluetooth Terminal, instalado en una Tablet Android. Estas instrucciones permitirán leer o escribir datos sobre cualquier pin digital o analógico de Arduino, así como la lectura de los sensores DHT11, LDR y NTC. Además se podrán guardar las lecturas sobre ficheros que tendrán un formato compatible para un procesamiento posterior con Labview o Matlab, y la visualización de estas medidas en un servidor web a tiempo real, con un periodo de refresco de 5 segundos.

La programación se ha realizado toda en Arduino IDE. Este programa tiene como base la manipulación de 5 variables de datos (variable 1,2,3,4,5), que contendrán la lectura de un tipo de dato medido en cierto pin del microcontrolador. Cada una de estas variables puede contener un dato de los siguientes tipos: lectura digital, lectura analógica, medida del LDR en luxes, medida del NTC en ºC, y medidas del DHT11 de humedad en tanto por cien y temperatura en ºC. Y cada una de estas cinco variables se podrá controlar para escribir sus datos periódicamente en ficheros *txt* alojados en la tarjeta SD, nombrados como *datos1.txt* para los datos de la variable 1, *datos2.txt* para almacenar la variable 2, y así sucesivamente.

Los comandos que se pueden emplear en el Bluetooth Terminal de Android son los mostrados en la siguiente tabla:

Número	Comando	Ejemplo(s)	Explicación
1	delay, tiempo	*delay 100*	Control sobre el tiempo de muestreo del microcontrolador, puesto a 100 milisegundos.
2	write, número de pin, 0/1/pwm	*write 5 1* *write 5 100*	Escribe sobre el pin digital 5 el valor de 1 (5V) o 0 (0V). En el caso de que el tercer campo sea un valor entre 2 y 255, lo que se escribirá sobre el pin 5 es una pwm de parámetro 100.
3	digitalread pin variable	*digitalread 7 2*	Guarda en la variable 2 del programa (recordar que hay 5 variables disponibles) la lectura periódica del pin digital 7.
4	analogread pin variable	*analogread 5 1*	Guarda en la variable 1 la lectura del pin analógico 5, cuyos valores están comprendidos entre 0 (0V) y 1023 (5V).
5	ldr pin variable	*ldr 0 2*	Guarda en la variable 2 el valor en lux medido por la LDR, conectada en el pin 0 de la manera que se explicó en el apartado 3.1 de la memoria.
6	ntc pin variable	*ntc 1 5*	Almacena en la variable 5 la medida de temperatura del NTC conectado en el pin 1 de la manera descrita en el punto 3.2

Número	Comando	Ejemplo(s)	Explicación
7	dht_h pin variable	*dht_h 0 4*	Guarda en la variable 4 la medida de humedad del DHT, conectado en el pin digital 0 como se detalla en el apartado 3.3
8	dht_t pin variable	*dht_t 0 4*	Almacena en la variable 4 el valor de temperatura devuelto por el DHT, conectado en el pin digital 0.
9	abrir_fichero variable	*abrir_fichero 1*	Comienza a guardar los datos de la variable 1 en el fichero *datos1.txt*.
10	abrir_todos	*abrir_todos*	Comienza el guardado de las 5 variables definidas en sus ficheros correspondientes.
11	cerrar_fichero variable	*cerrar_fichero 2*	Detiene el guardado en fichero de la variable 2.
12	cerrar_todos	*cerrar_todos*	Concluye las operaciones de guardado en todos los ficheros y los cierra.
13	eliminar_fichero variable	*eliminar_fichero 3*	Elimina de la memoria de la tarjeta SD el fichero *datos3.txt*.
14	eliminar_todos	*eliminar_todos*	Elimina todos los ficheros datos.txt de la memoria SD, que serán creados nuevamente al usar los comandos de apertura 9 y 10.

Tabla 2: Envío de comandos y recepción de datos con Bluetooth Terminal

Los comandos enviados llegan a Arduino como cadena de caracteres por el puerto Serial 3 donde está conectado el módulo Bluetooth HC-06. Esta cadena será comparada carácter a carácter para determinar a qué instrucción hace referencia. Además, Arduino devuelve las medidas de las variables del 1 al 5 definidas hasta el momento en la consola de Bluetooth Terminal, véase la Figura 52.

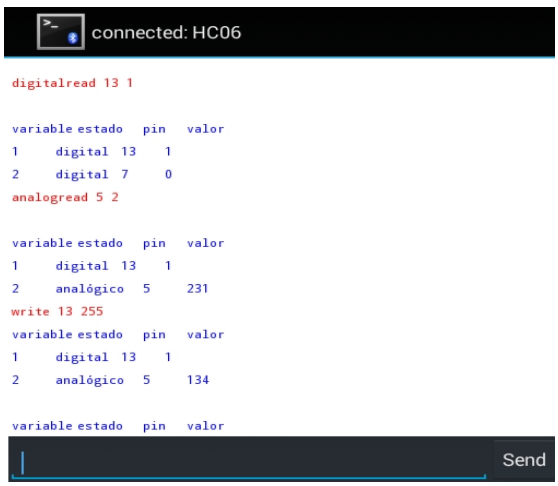


Figura 52: Envío de comandos y recepción de datos con Bluetooth Terminal

Finalmente, tras realizar dichas instrucciones, se enviarán las medidas de las variables mediante Ethernet en un formato *html* a un servidor Web cuya IP es 192,168,0,15, donde se visualizarán en una tabla información como los valores de las cinco variables, el tipo de dato que contienen y los pines a los que están asociados. El resultado visible en el navegador es el mostrado en la figura 53.

Variable	Tipo	Pin	Valor
1	Analogico	5	843
2	Digital	7	1
3	Ldr(lux)	0	538
4	Dht humedad(%)	2	54
5	Ntc temperatura(oC)	1	26.47

Figura 53: Visualización del Servidor Web en un navegador.

Con esta pequeña aplicación se puede concluir el estudio de las tecnologías inalámbricas aplicadas a Arduino y a la adquisición de señales. En este programa de Arduino IDE no se ha incluido ninguna herramienta de procesamiento, por tanto se debe emplear como complemento a los diseños de captura y procesamiento de señales realizados en Matlab y Labview, que son el bloque principal del proyecto y cumplen por si solos la principales el principal objetivo planteado en el inicio de esta memoria.

6- CONCLUSIONES

6.1- CONCLUSIONES Y ANÁLISIS

Una vez estudiados y analizados todos los puntos de interés que se plantearon al inicio de este proyecto, se puede concluir que se han cumplido todos los objetivos propuestos, pues se han logrado diseñar dos aplicaciones completas de captura, procesamiento y guardado de señales, que además permiten el control de salidas y entradas digitales de una manera muy simple. Y a pesar de que estas aplicaciones de Matlab y Labview funcionen mediante la comunicación con el microcontrolador a través de un cable USB, se ha presentado una mejora de conectividad con la tecnología Bluetooth y Ethernet.

Dicha mejora de conectividad inalámbrica no ha sido posible aplicarla directamente sobre los diseños de captura y procesamiento, pero se han probado sobre ejemplos más sencillos y se han dado las directrices a seguir para lograr que estos diseños tengan una comunicación sin cables con Arduino.

Para alcanzar este objetivo ha sido necesario profundizar en los lenguajes de programación que utilizan los softwares Matlab, Labview y Arduino IDE, y conocer las especificaciones principales y el funcionamiento del hardware y sensores empleados, explicado a lo largo de los apartados 2 y 3. Hecho esto, se ha considerado que estábamos en disposición de abordar el tema principal del trabajo y adecuar las aplicaciones a programar a las necesidades planteadas. Estas necesidades están basadas en la interacción con el entorno a través de la lectura, captura de datos y posterior actuación con señales eléctricas que pueden ser obtenidas a partir del procesamiento de los datos capturados.

Por otra parte, se debe tener en cuenta que los programas desarrollados no tienen una aplicación concreta para la que pueden ser usados, ni sirven únicamente para capturar un tipo de señal. Su finalidad es, por lo tanto, la de servir de herramienta a aquellas personas que requieran de una plataforma de captura y procesamiento de señales, señales que independientemente de su naturaleza (acústicas, magnitudes de fuerza) puedan ser transformadas a una señal eléctrica que guarde una relación conocida con la medida original. Dicha señal deberá ser amplificada o atenuada para llevarla a unos niveles adecuados de tensión y que pueda ser usada como entrada al microcontrolador.

Una vez reconocido el propósito general de la aplicación, se va a incidir es aquellos puntos susceptibles de mejora que debido a las limitaciones de tiempo no ha sido posible implementar de manera satisfactoria, y por lo tanto serán objeto de estudio de trabajos futuros.

6.2- TRABAJOS FUTUROS

Durante el desarrollo de la memoria se han presentado algunos de los puntos que se han decidido incluir en esta sección de trabajos futuros para mejorar los resultados obtenidos en este proyecto, la mayoría de ellos relacionados con el tema de las conexiones inalámbricas.

En primer lugar, se recuerda que tanto las aplicaciones de Labview y Matlab podrían conectarse con Arduino por Bluetooth o Ethernet, para lo que habría que manipular adecuadamente las Firmatas que utilizan y dichas aplicaciones para que trabajen por un puerto Serial virtual a través de una de las tecnologías inalámbricas. Concretamente, para Ethernet sería conveniente profundizar en el protocolo de comunicación Telnet, que permitiría la comunicación bidireccional entre dos dispositivos con la ayuda de una terminal de conexión Telnet como *TeraTerm* y un servidor Telnet alojado en Processing.

Además, durante el trabajo se han visto escasas utilidades a la tarjeta Arduino Ethernet Shield, por lo tanto convendría investigar en detalle otros de sus usos presentados en el manual de la tarjeta, y analizar si se pudieran emplear junto a los diseños de Labview y Matlab y si tendría sentido o no integrarlos con ellos.

Sobre las aplicaciones de captura y procesamiento programadas, es evidente que se podrían ampliar para que no sólo lean la tensión de los sensores, sino que además, elegido el tipo de sensor analógico, nos devuelva junto a la tensión de su pin el valor de la medida en las unidades correspondientes (de temperatura, luz, o cualquier otra). y en relación al sensor digital DHT11, que no puede ser medido por estas aplicaciones, se debe intentar la programación de la lectura controlando los intervalos de tiempo de transmisión de los bits, evitando de esta manera el uso de la librería *Dht.h* de Arduino IDE, o bien modificar las Firmatas para que puedan transmitir las medidas de humedad y temperatura devueltas por las funciones de esta librería.

Por último, las herramientas de procesamiento de señales incluidas en las aplicaciones como filtrado, combinación de señales, amplificación o manipulación de los niveles de continua son algunas herramientas de procesamiento que se han considerado de mayor utilidad. Aunque también sería de gran interés que estas aplicaciones dispusieran de otros tipos de procesamiento digital, como la codificación y decodificación de las señales recibidas, o la compresión de la información en caso de adquirir datos de gran tamaño, entre otros.

7- PRESUPUESTO

Sistema de captura y procesamiento de señales

DESCRIPCIÓN…….....	MEDICIÓN ..	PRECIO ..	IMPORTE
1 HARDWARE			
1.1 Arduino Mega 2560	1	47.19	47.19
1.2 Arduino Ethernet Shield REV3 con POE	1	57.96	57.96
1.3 Módulo HC 06	1	20.57	20.57
1.4 Adaptador Bluetooth LM054	1	22.28	22.28
1.5 Ordenador y Tablet Android	1	0*	0*
		Total	**148.00 €**
2 COMPONENTES ELECTRÓNICOS:			
2.1 Juego de resistencias, pulsador, cables y LEDs	1	0*	0*
2.2 Fotorresistencia LDR	1	0.54	0.54
2.3 Termistor NTC	1	0.44	0.44
2.4 Sensor DHT11	1	6.78	6.78
		Total	**7.76 €**
3 COSTES DE PROGRAMACIÓN			
3.1 Programación en Matlab	30h	7.00	210.00
3.2 Programación en Labview	40h	7.00	280.00
3.3 Programación en Arduino IDE	20h	7.00	140.00
		Total	**630.00 €**
4 OTROS			
4.1 Fuente de alimentación Arduino	1	12.20	12.20
		Total	**12.20 €**

PRESUPUESTO TOTAL …….. **797.96 €**

** El coste de estos materiales es nulo porque fueron adquiridos años antes del inicio del proyecto*

8- BIBLIOGRAFÍA / WEBGRAFÍA

- Manuales básicos de Arduino:

Web oficial de Arduino: http://www.arduino.cc/en/Guide/HomePage
Manual de instalación de drivers y funciones:
http://www.jcarazo.com/tmp/Arduino_user_manual_es.pdf
http://arduinobot.pbworks.com/f/Manual+Programacion+Arduino.pdf

- Processing:

Página oficial de Processing: descarga, librerías y funciones: https://processing.org/
Comunicación Arduino-Processing: https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing
http://diymakers.es/arduino-processing-primeros-pasos/
Comunicación Arduino-Processing mediante *Standard Firmata*:
http://www.marlonj.com/blog/2012/05/controlando-tarjeta-arduino-directamente-desde-processing/
https://www.youtube.com/watch?v=8OHs1VqIzVU
https://www.youtube.com/watch?v=QjvVPpjGWF8
Pdf de prácticas Arduino- Processing: http://myslide.es/documents/practicas-arduino-processingpdf.html

- Matlab:

Comunicación Serial: https://geekytheory.com/matlab-arduino-serial-port-communication/
Comunicación por Firmata: http://www.academia.edu/9958859/Enlace_Arduino_Matlab
Programación de GUIs: http://www.dspace.espol.edu.ec/bitstream/123456789/10740/19/%255D_MATLAB_GUIDE.pdf
Compilar una GUI en un ejecutable .exe: https://www.youtube.com/watch?v=rUARDvYdedI

- Labview:

Link de descarga de Labview: http://www.fiuxy.com/programas-gratis/3231121-labview-2013-full-32bits-64bits-mega.html
Programación de Labview para Arduino paso a paso:
http://es.slideshare.net/AlbertoSanchez6/arduino-lab-view

- Sensor DHT11:

Manual DHT11: http://www.micropik.com/PDF/dht11.pdf
Lectura desde Arduino IDE con o sin librería: http://tallerarduino.com/2012/12/24/sensor-dht11-humedad-y-temperatura-con-arduino/

- Filtrado:

Filtrado digital de señales:
http://www.elai.upm.es/webantigua/spain/Publicaciones/pub01/intro_procsdig.pdf
Características y diferencias entre modelos de filtros:
https://es.wikipedia.org/wiki/Filtro_de_Cauer
https://es.wikipedia.org/wiki/Filtro_de_Chebyshev
https://es.wikipedia.org/wiki/Filtro_de_Butterworth
Códigos de programas de la optativa Laboratorio de Automatización y Control 2015 como apoyo.

- Módulo Bluetooth HC-06:

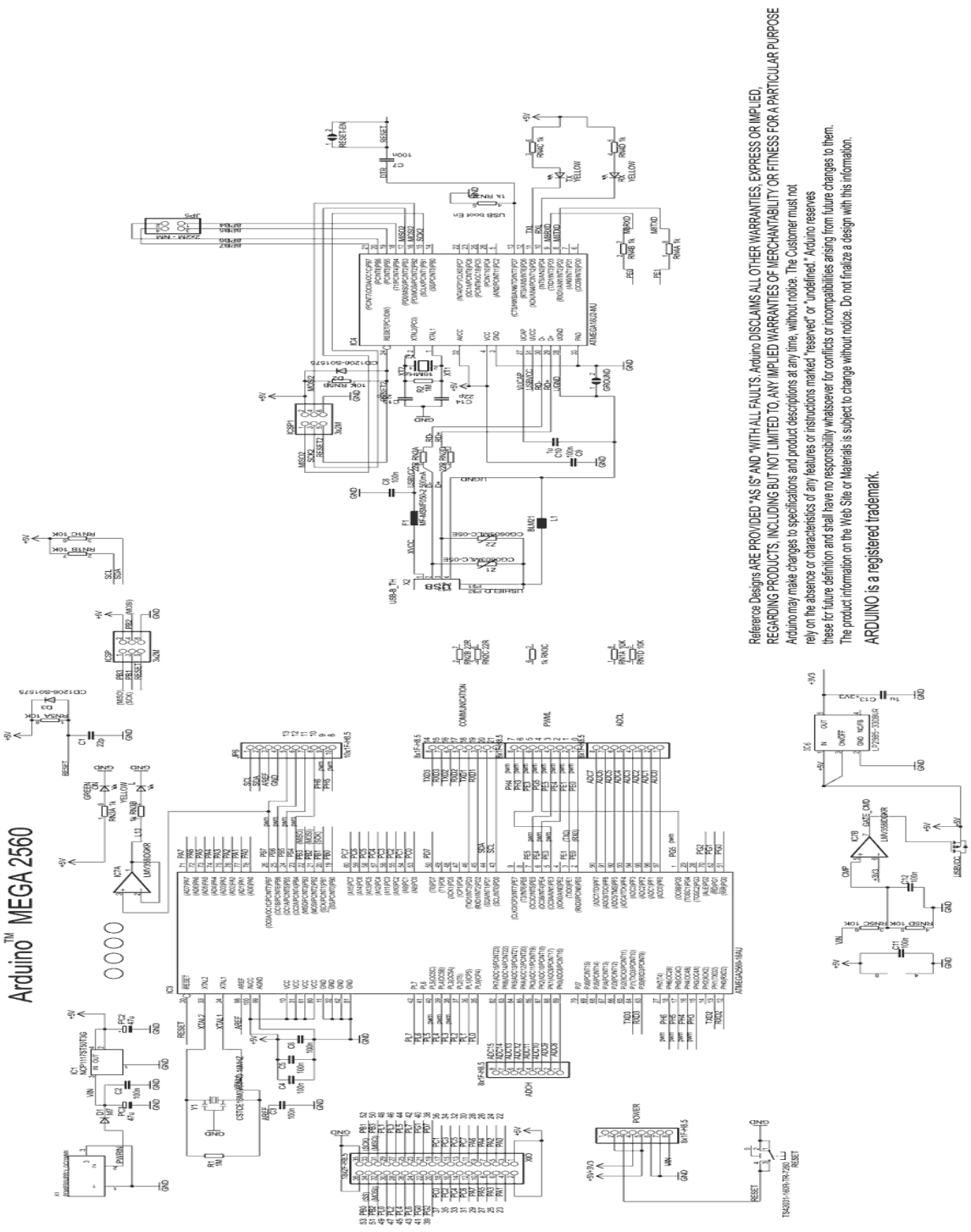
Manual modelos HC: http://www.tec.reutlingen-university.de/uploads/media/DatenblattHC-05_BT-Modul.pdf
Configuración con comandos AT y conexión con y sin librería Software Serial:
http://saber.patagoniatecnology.com/modulo-bluetooth-hc-06-bluetooth-arduino-slave-hc06-esclavo-iot/
http://diymakers.es/arduino-bluetooth/

- Arduino Ethernet Shield:

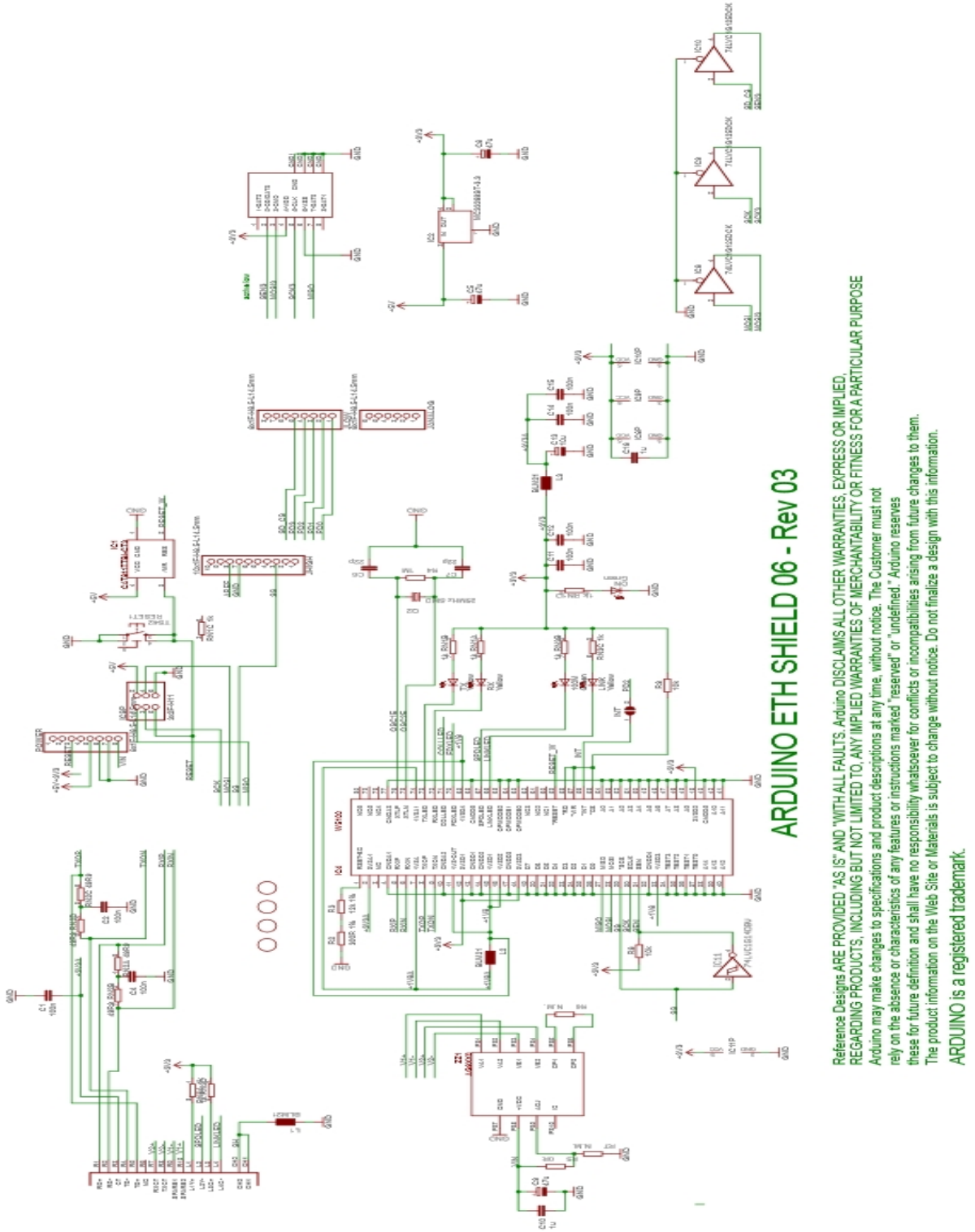
Manual y aplicaciones:
http://unicarlos.com/_ARDUINO/Arduino%20+%20Ethernet%20Shield%20%281%29.pdf
Configuración Módem y programación de un Cliente Web:
http://www.educachip.com/arduino-ethernet-shield/
http://www.tuelectronica.es/tutoriales/arduino/arduino-ethernet-shield.html
Uso de la tarjeta SD:
http://www.educachip.com/como-leer-y-escribir-datos-en-la-tarjeta-sd-de-arduino/

9-ANEJOS DE LA MEMORIA

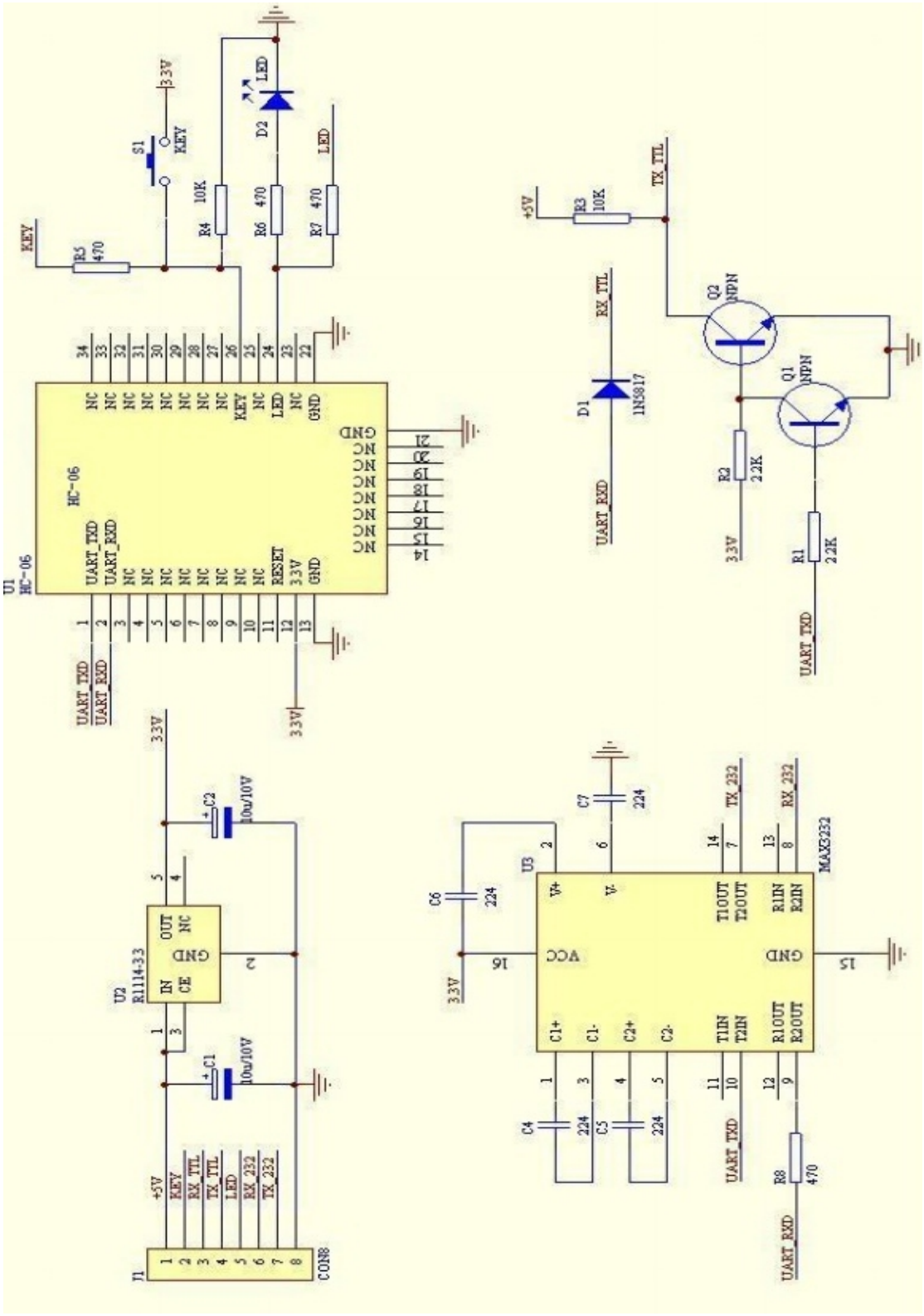
9.1- DATASHEET DE ARDUINO MEGA 2560:



9.2- DATASHEET DE ARDUINO ETHERNET SHIELD REV3:



9.3- DATASHEET DEL MÓDULO BLUETOOTH HC-06:



9.4- ESPECIFICACIONES DEL SENSOR DHT11:

Overview:

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Detailed Specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

6. Electrical Characteristics

VDD=5V, T = 25℃ (unless otherwise stated)

	Conditions	Minimum	Typical	Maximum
Power Supply	DC	3V	5V	5.5V
Current Supply	Measuring	0.5mA		2.5mA
	Average	0.2mA		1mA
	Standby	100uA		150uA
Sampling period	Second	1		