UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

"Implementación de una estación meteorológica con Arduino"

TRABAJO FINAL DE GRADO

Autor/a:
Jose Escribano Vega

Tutor/a:
María Asunción Pérez Pascual

GANDIA, 2016

RESUMEN

El presente trabajo expone la creación de una estación meteorológica basada en tecnología Arduino, cuyos datos obtenidos mediante sensores de temperatura, humedad y presión serán transmitidos desde Arduino a una aplicación Android utilizando la tecnología Bluetooth. Para llevar a cabo el proyecto se han integrado diferentes campos de programación, como punto base de este se encuentra la programación realizada mediante la IDE de Arduino, soportada por una librería implementada en el lenguaje C++. Para la creación de la aplicación se hace uso de un lenguaje adaptativo de creación de aplicaciones Android basado en programación de bloques.

PALABRAS CLAVE

Programación, meteorología, comunicación, Android, Arduino.

ABSTRACT

This work presents the creation of a weather station based on Arduino's technology, whose data obtained through temperature sensors, humidity and pressure will be transmitted from Arduino to an Android aplication using Bluetooth technology. To carry out the project, different programming languages have been integrated, as its base point, we find the programming made through the IDE of Arduino, supported by a bookshop installed in the C++ language. To create the application, we use an adaptive language of creation of Android applications based on programming block.

KEYWORDS
Programming, meteorology, communication, Android, Arduino.

Tabla de contenido

Tabla

Tabla de ilustraciones

1. INTRODUCCIÓN

En la actualidad la demanda de proyectos basados en microcontroladores de hardware libre se ha consolidado tanto a nivel de aprendizaje como profesional. La sencillez tanto de implementación como de diseño de estos dispositivos permite al usuario realizar proyectos con rapidez y eficacia, dejando atrás lenguajes de bajo nivel los cuales dificultaban la programación del microcontrolador. La expansión creciente de este mercado ha permitido que el usuario pueda llevar a cabo cualquier tipo de proyecto. Ya que la gama de microcontroladores cada vez es mayor y más potente, así como la multitud de sensores y componentes electrónicos adaptados a éstos.

Como bien es sabido los teléfonos inteligentes (smartphones) irrumpen en nuestra vida diaria desde hace unos años atrás. Dispositivos que ya no son un simple sistema de comunicación si no que es una herramienta de trabajo muy completa, asemejándose a la funcionalidad de un pequeño ordenador. La competencia de este mercado es encarnizada lo que ha provocado que la calidad sea cada vez mayor ofreciendo numerosas opciones tanto de dispositivo como de sistema operativo. El sistema operativo más utilizado a día de hoy es Android, la clave de su éxito está en *Google Play Store,* plataforma que ofrece numerosas aplicaciones gratuitas. Además, permite al usuario crear sus propias aplicaciones con la posibilidad de poder lanzarlas al público.

En este proyecto se juntan dos tecnologías nombradas con anterioridad. Por una parte, se ha utilizado como microcontrolador de hardware libre *Arduino UNO* usando el entorno de desarrollo propio de Arduino. Por otra parte, se ha elegido el sistema operativo Android para la creación de la aplicación donde serán visibles los datos, para realizar esta aplicación se utilizará App inventor 2. Con ello Arduino captará la información meteorológica con diferentes sensores y comunicará esta información a la aplicación. Como herramienta de comunicación entre ambas tecnologías se utiliza un módulo Bluetooth.

1.1. Objetivos

El **objetivo principal** de este proyecto es captar parámetros meteorológicos mediante sensores conectados a Arduino y mostrar el resultado en un dispositivo de telefonía móvil.

Objetivos segundarios que se pretenden conseguir en este proyecto:

I. Montaje del sistema.
II. Obtener información meteorológica de los sensores, mostrando el resultado en pantalla (monitor serie Arduino).
III. Creación de la aplicación Android.
IV. Establecer comunicación Bluetooth.
V. Obtener y mostrar los datos en la aplicación.

El **objetivo personal** que quiero cumplir al realizar este proyecto, es tener un mejor conocimiento sobre la tecnología de microprocesadores de hardware libre, en este caso Arduino y cómo adaptar este tipo de sistema para utilizarlo con dispositivos móviles.

1.2. Metodología

Tras la elección y aprobación del trabajo que aquí se expone. Se analizó las necesidades globales del sistema y la consiguiente elección de los dispositivos. Posteriormente se realizó un diseño que facilitó la comprensión del montaje del sistema. Tras el montaje se realizaron diversos algoritmos que a posteriori ayudaron a la implementación del código. Teniendo clara la estructura, la programación se realizó en pequeños fragmentos que posteriormente se han juntado en el código general. Por último, se realizaron numerosas pruebas que afianzaran la fiabilidad del proyecto.

1.3. Etapas

A continuación, se mostrará las etapas distribuidas en el tiempo que se han llevado a cabo para la realización del proyecto.
Cada cuadro indica una semana del mes, siendo el primer cuadro la primera semana del mes.

Tabla 1: Calendario de etapas

Feb.	Marzo	Abril	Mayo	Junio	Julio	Agosto
	2	3,4	6	8	9	10
1	2,3	3,4	6	8	7	11
1	2,3	4	6	9	9,10	
1	3	5	8	9	9,10	

1. **Búsqueda de documentación y bibliografía sobre el proyecto.** Búsqueda de información que ayude a afrontar a decidir los primeros pasos a seguir del proyecto. Decisiones como, que entornos de programación es más adecuado utilizar o planificación de las tareas a realizar.
2. **Aprendizaje de los conceptos básicos de Arduino.** Se estudia la base de programación de Arduino y el entorno de desarrollo del mismo. Para ello se hace uso de la página oficial y se realizan numerosas pruebas con módulos tinkerkit que posee la universidad.
3. **Estudio del lenguaje de programación C++.** Es necesario el estudio de este lenguaje ya que facilita la implementación de una librería externa a Arduino.
4. **Elección de los dispositivos.** Se realiza un estudio sobre que sensores y módulos son los más adecuados y económicos para cubrir las necesidades del proyecto.
5. **Encargo de los dispositivos.** Envío de la lista de componentes necesarios. El encargo se llevó a cabo a manos de mi tutora de proyecto.
6. **Diseño de montaje.** Se realizan unos planos indicando las conexiones entre arduino y los sensores.
7. **Montaje.** Conexiones, soldaduras, etc.
8. **Algoritmos de implementación.** Algoritmos de los diferentes fragmentos de código que se implementaran.
9. **Implementación del código de Arduino y librería.** Implementación del código necesario para la recepción de datos de los sensores en Arduino y la transmisión Bluetooth. Junto al código base de Arduino, se implementa una librería(C++), de esta forma se evita un código demasiado amplio en Arduino.
10. **Implementación y diseño de la aplicación mediante App Inventor 2.** Diseño e implementación (basada en bloques) de la aplicación de Android.
11. **Testeo del sistema.** Comprobar si el sistema funciona correctamente.

1.3. Estructura de la memoria

El contenido de la memoria se ha estructurado de la siguiente manera: en el capítulo 2 se presenta el material utilizado para implementar el sistema, se describe tanto el hardware como el software empleado. En el capítulo 3 se muestra el diseño y montaje del sistema, mostrando

los planos de conexión. En el capítulo 4 y 5 se presentan los algoritmos de los programas realizados y los códigos implementados, tanto para el control del Arduino como la aplicación en Android. El capítulo 6 muestra los resultados de la aplicación y, por último, en el capítulo 7 se presentan las conclusiones del trabajo.

2. MATERIAL

En los próximos apartados se describen las características e información de interés de los dispositivos utilizados en el proyecto, así como los entornos de programación utilizados.

2.1. Hardware

Características e información tanto del microcontrolador, como de los sensores que captarán los parámetros meteorológicos y transmitirán esta información.

2.1.1. Arduino UNO

Arduino dispone de una gama muy amplia de microcontroladores, no obstante, para este proyecto se ha decidido adquirir Arduino UNO. Arduino UNO es uno de los microcontroladores más conocidos del mercado debido a su equilibrio de sencillez y prestaciones, lo convierte en la herramienta perfecta para el aprendizaje.

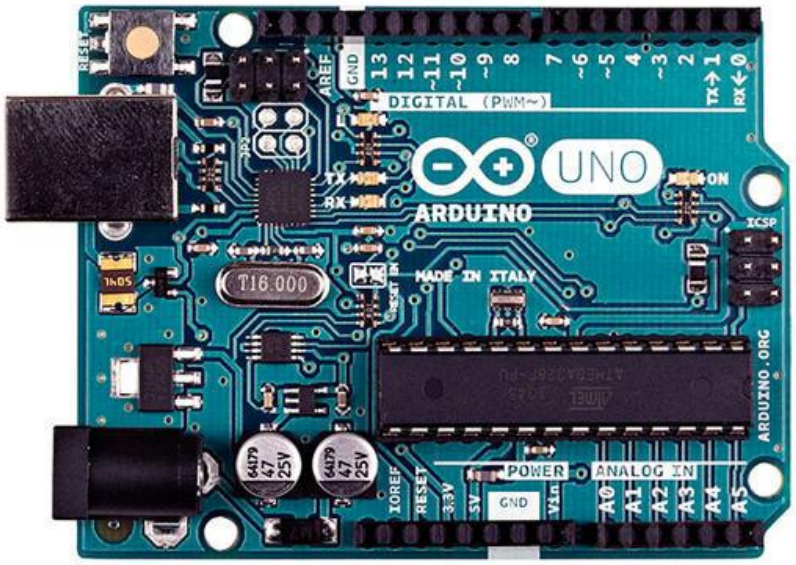


Ilustración 1: Vista frontal de Arduino UNO

Especificaciones técnicas**:**

Tabla 2: Especificaciones técnicas Arduino.

Microcontrolador	ATmega328P
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Pines de entrada/salida digitales	14 (de los cuales 6 proporcionan salida PWM)
Pines entrada/salida PWM digitales	6
Pines de entrada analógica	6
Corriente continua para Pin entrada/salida	20 mA
Corriente continua para Pin 3.3V	50 mA
Memoria flash	32 KB (ATmega328P) de los cuales 0,5 KB utilizado por el gestor de arranque
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)

Velocidad de reloj	16 MHz
Longitud	68,6 mm
Anchura	53,4 mm
Peso	25 g

Arduino UNO se encarga de llevar a cabo todo el procesamiento del proyecto, transmitiendo y recibiendo información de los módulos conectados. Para ello Arduino Uno está compuesto de 3 partes fundamentales:

- Unidad de procesamiento o **CPU**: encargada de ejecutar cada instrucción implementada en la IDE.
- Sección de **memoria:** encargada de almacenar las instrucciones y los datos de entrada y de salida.
- **Pines** de entrada y salida: ranuras que permiten a Arduino UNO conectarse con módulos exteriores. Algunos de estos pines son los encargados de ofrecer la tensión al mismo módulo para que éste obtenga energía.

Para más información consultar fichero “Ficheros anexos/ArduinoUNO”

2.1.2. Módulos

En este apartado se describe las características generales y funcionalidad de los sensores que se encargan de recibir los parámetros meteorológicos. También se explicará el módulo de comunicación que permite la comunicación entre la placa Arduino y el dispositivo Android.

2.1.2.1. *Sensor de humedad (HH10D)*

Sensor encargado de detectar la humedad relativa. Compuesto por un sensor capacitivo, un convertidor capacitivo CMOS y una EEPROM interna la cual se encarga de almacenar los valores de calibración del dispositivo.

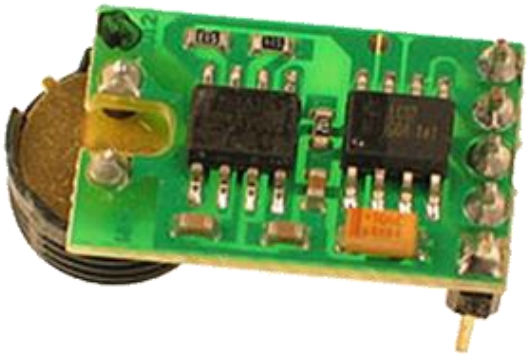


Ilustración 2: Sensor de humedad (HH10D)

Especificaciones técnicas:

Interfaz de comunicación	I2C
Alimentación	2.7V - 3.3V
Consumo	150µA
Rango de temperatura de funcionamiento	-10ºC a 60ºC
Precisión	+-3%
Rango de medida	1% a 99%

Pines:

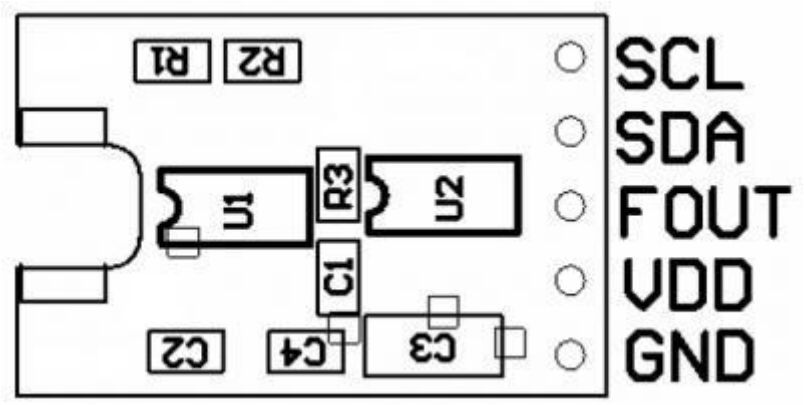


Ilustración 3: Esquema de pines del sensor de humedad HH10D

- **SCL:** pin de comunicación (interfaz I2C), marca la señal de reloj para que los datos sean recibidos de manera sincronizada.
- **SDA:** pin de comunicación (interfaz I2C), encargado de transmitir los datos.
- **FOUT:** pin donde se obtiene el valor de humedad.
- **VDD:** pin de alimentación.
- **GND:** masa, elemento de protección del circuito que evita perturbaciones que puedan afectar al funcionamiento del dispositivo.

Se eligió este dispositivo debido a la facilidad de su funcionamiento en Arduino y la cantidad de información que hay al respecto en la red.

2.1.2.2 Sensor de presión atmosférica y temperatura(Bmp180)

Sensor encargado de detectar los parámetros meteorológicos de presión atmosférica y temperatura. Bmp180 gracias a su sensor de presión barométrica ofrece un rango de medida bastante amplio de 300 a 1100 hPa con un rango de error muy bajo de tan solo 0.03hPa. Este tipo de sensores han sido diseñados para recibir información de temperatura y altitud con bastante precisión.

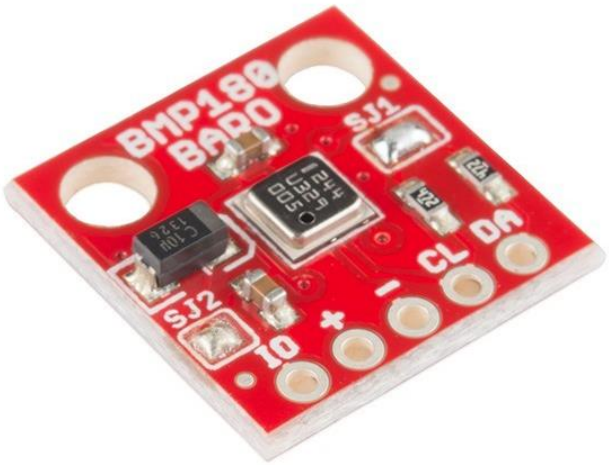


Ilustración 4: Sensor de presión atmosférica y temperatura(Bmp180)

Especificaciones Técnicas:

Interfaz de comunicación	I2C
Alimentación	1.8 - 3.6Vdc
Consumo	3-32µA
Rango de temperatura de funcionamiento	-40ºC a 60ºC (operacional) 0ºC a 45ºC (máx.)

Rango de medida (humedad)	300 a 1100 hPa
Precisión (humedad)	0.03hPa
Rango de medida (temperatura)	-40ºC – 85ºC

Pines:

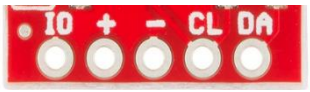


Ilustración 5: Pines del sensor de presión atmosférica (bmp180)

- **IO:** encargado de la tensión del sensor. Este conector se deja desconectado a menos que se conecte un microprocesador de tensión reducida.
- **+:** alimentación.
- **-:** GND, encargado de la protección electrónica del circuito.
- **CL:** señal de reloj (I2c)
- **DA:** datos (I2c)

Se escogió este dispositivo por razones similares al anterior sensor. Aparte este sensor incluye funcionalidades extra de interés en el proyecto como la toma de temperatura.

2.1.2.3 Bluetooth (HC-06)

Módulo encargado de la comunicación periférica con el dispositivo Android, en este caso un Smartphone.

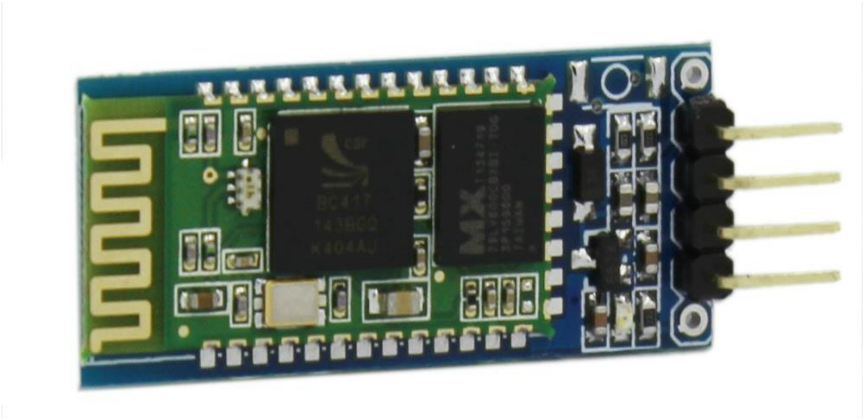


Ilustración 6: Módulo de comunicación Bluetooth (HC-06)

Especificaciones Técnicas:

Voltaje de alimentación	3.3Vdc – 6Vdc
Voltaje de operación	3.3Vdc
Velocidad de comunicación (baudios)	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
Corriente de operación	< 40mA
Corriente en estado de stand by	<1 mA

Este modelo dispone de cuatro **pines** que se indicaran a continuación:

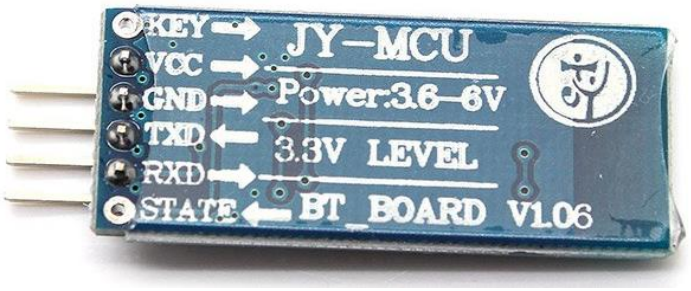


Ilustración 7: Pines bluetooth (HC-06)

- **VCC:** pin de alimentación de la placa.
- **GND:** masa.
- **TXD:** pin de comunicación, encargado de transmitir la información al dispositivo conectado.
- **RXD:** pin de comunicación, encargado de recibir información del dispositivo conectado.

Este módulo solo funciona en modo esclavo, es decir que solamente puede conectarse a un Master y recibir información de este. Un Master puede ser un ordenador, smartphone, etc. Si el módulo Bluetooth tuviera la posibilidad de realizar la funcionalidad de master como es el caso de HC-05, permitiría generar conexiones hacia otros dispositivos arbitrando las transferencias de información.

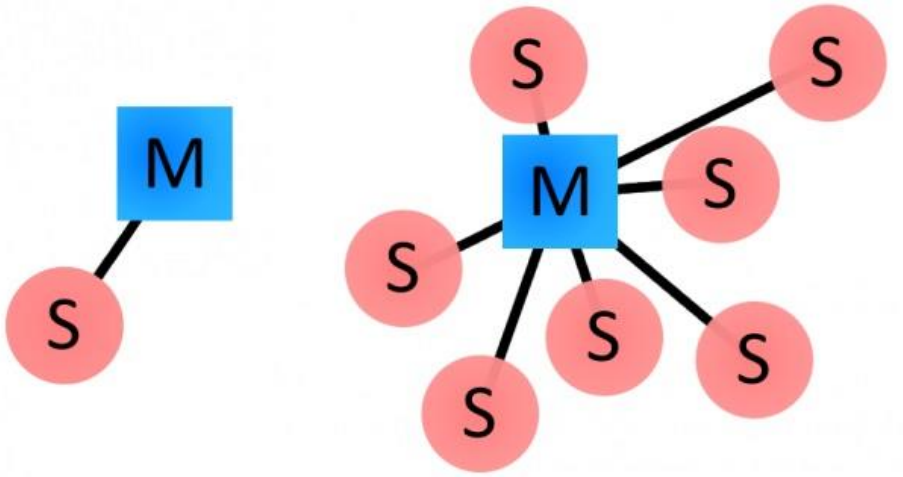


Ilustración 8: Esquema Master-Esclavo (modulo Bluetooth)

Este dispositivo permite que se configure características tales como: Velocidad de comunicación(baudrate), nombre, clave, etc. Esta configuración se puede llevar a cabo en el monitor serie de Arduino introduciendo comandos AT.

Con todo lo nombrado en las anteriores líneas se puede concluir que este módulo es más que adecuado para poder llevar a cabo el proyecto. Factores como información sobre éste, facilidad de conexión con Arduino y su precio ayudaron a la elección de este módulo Bluetooth.

2.2 Entorno de programación y software.

A continuación, se describe las características principales a destacar de estos entornos, así como de sus principales ventajas.

2.2.1 Arduino IDE

Entorno de programación de Arduino disponible en la página oficial [1]. Disponible para GNU/Linux, Mac OS y Windows, software totalmente gratuito. Basado en un lenguaje de programación inspirado en el lenguaje *Processing,* ofrece un lenguaje muy familiar a otros

manteniendo patrones de programación, ya sea creación de variables, bucles, bloques lógicos, etc.

Esta IDE cuenta con diversas herramientas que facilitan la ejecución, creación y testeo de los propios programas. A continuación, se mostrará la **interfaz** y las partes que lo componen:

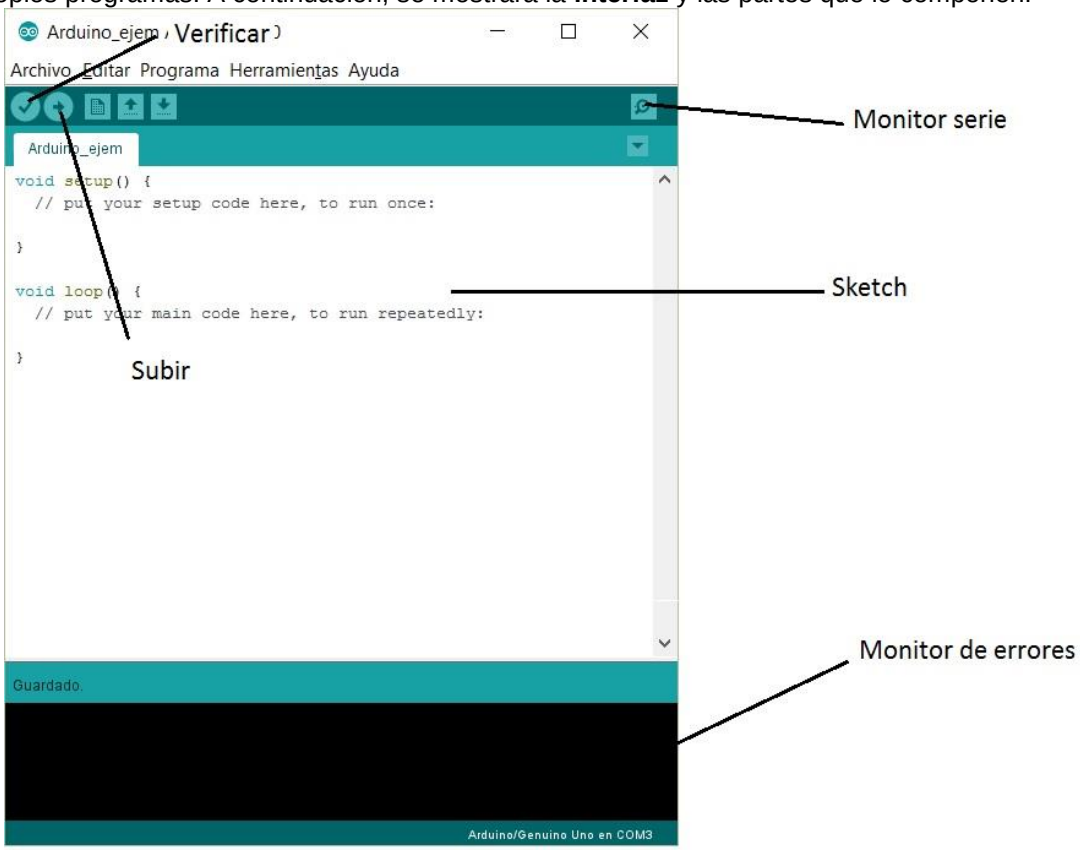


Ilustración 9: Interfaz IDE de Arduino

- **Verificar:** compilador, comprueba si hay algún error de código tanto del propio sketch como de una librería importada.
- **Subir:** compila y carga el programa en la placa Arduino.
- **Monitor serie:** permite observar la comunicación del USB conectado de Arduino a nuestro ordenador. En él se puede verificar si el programa funciona correctamente observando resultados del programa o notificaciones.
- **Sketch:** editor de texto de la IDE.
- **Monitor de errores:** muestra los errores que se han producido al ejecutar el compilador. Indica en que fichero y línea se ha producido el error, así como una pequeña descripción del error.

El código del programa se puede extender importando librerías propias de la IDE o externas mediante un formato de compresión ZIP. El propio entorno trae instaladas una serie de librerías por defecto de gran utilidad.

2.2.2 Notepad++

Notepad++ [16] es un editor de texto orientado a la programación que soporta un número extenso de lenguajes. Garantiza una velocidad de ejecución alta y un menor tamaño de fichero, debido a que está escrito en C++ y utiliza directamente la API de WIn32.

Ilustración 10: logo de Notepad++

En este entorno se implementa el código de la librería que aportara funciones al fichero de Arduino, de esta forma se evita un exceso de código en el fichero principal.

2.2.3 App Inventor 2

App Inventor 2 [8] es un innovador entorno de desarrollo para aplicaciones Android que tiene como objetivo simplificar la forma de programar aplicaciones. Para ello se ha sustituido el lenguaje complejo basado en texto por un lenguaje basado en bloques. De esta manera la tarea de programación pasa a ser visual, consiguiendo realizar proyectos en un menor tiempo.

Ilustración 11: Logo de App Inventor 2

La conexión con el dispositivo Smartphone es muy sencilla y adaptativa, permite una conexión mediante Wifi, Usb o ejecutar un emulador.

La **interfaz** está compuesta en dos partes, la parte que se lleva a cabo el diseño gráfico de la aplicación y la parte donde se lleva a cabo la programación basada en bloques.

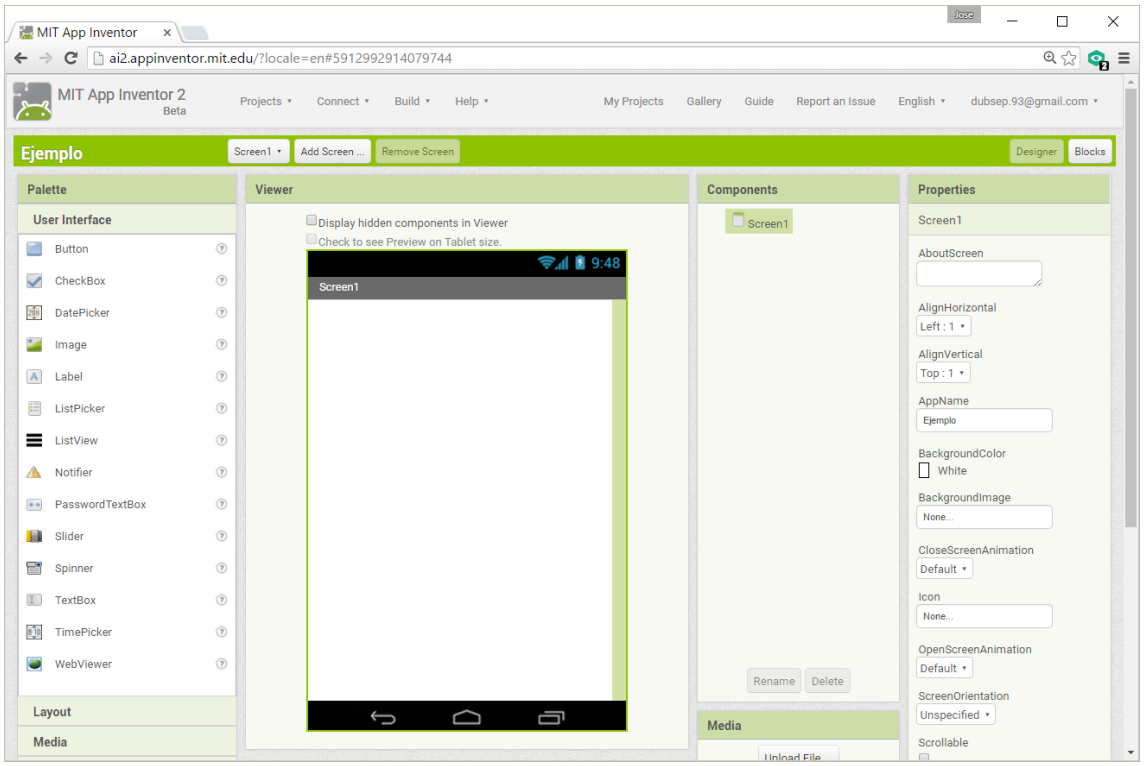


Ilustración 12: Interfaz gráfica de App Inventor 2.

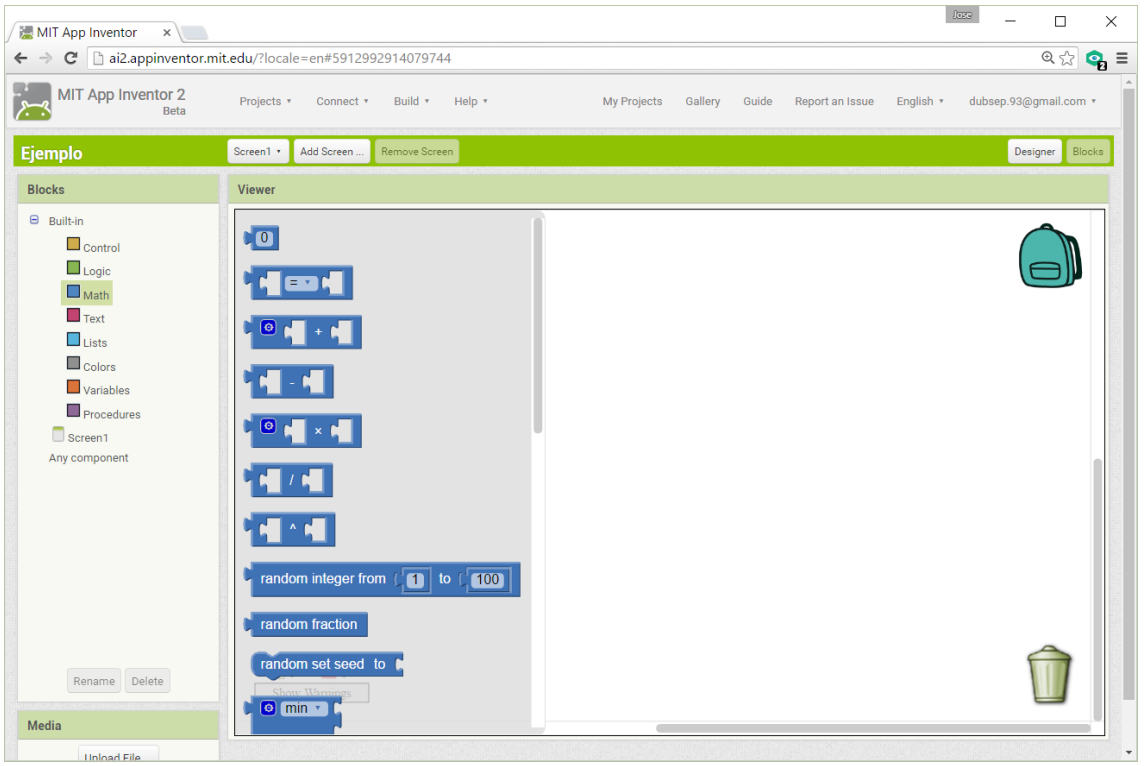


Ilustración 13: Interfaz de programación de App Inventor 2.

Debido a que este entorno de desarrollo tiene un módulo de bloques ya implementado para la conexión Bluetooth, ha hecho que sea la herramienta perfecta para llevar el desarrollo de la aplicación.

3. DISEÑO Y MONTAJE

En este apartado se muestra el proceso de montaje que se ha llevado a cabo en el proyecto. Para ello se explica el bus serie de datos I2c y se muestra los planos de conexión y su funcionamiento.

3.1 Bus I2c

Antes de mostrar los planos de conexión y como se comunican los módulos con Arduino UNO es necesario explicar el bus de comunicación serie I2c.

El bus de comunicación serie I2c, es un protocolo de comunicación digital de dispositivos electrónicos. Este bus está marcado por una serie de características que definen este protocolo, en los siguientes puntos se define la idea básica de este:

- **SDA** (Serial Data) y **SCL** (Serial Clock), compuesto por dos líneas de control, una dedicada a la transmisión de datos(SDA) y otra al reloj asíncrono, que indica cuando se debe leer los datos (SCL). A parte de estas líneas de control es necesario la línea de masa (GND) y la línea de alimentación (Vcc).
- **Dirección exclusiva,** cada dispositivo dispone de una dirección propia de 7 bits. Por lo tanto, se puede conectar un número total de 128 dispositivos (2^7).
- Al menos uno de los dispositivos tiene que ser el **Master,** encargado de controlar la señal de reloj. Debido a que el Master es el que controla el reloj no es necesario un control estricto de la velocidad de este.

El protocolo I2c sigue una serie de pasos a la hora de transmitir y leer información. En las siguientes ilustraciones se muestra el proceso de comunicación entre el Master y los esclavos.

Para información extra consultar fichero “Ficheros anexos/ProtocoloI2c”.

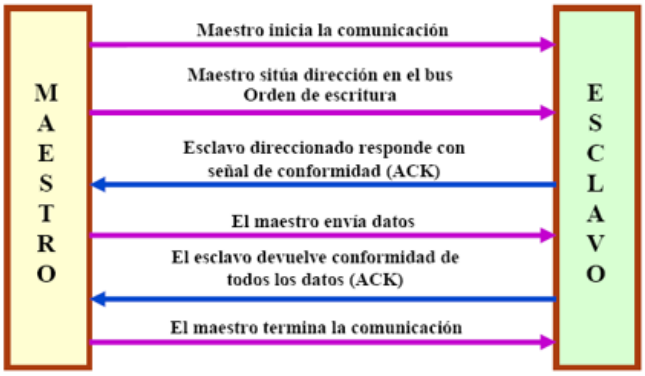


Ilustración 14: Diagrama de protcolo I2c, transmisión de datos Master-Slave

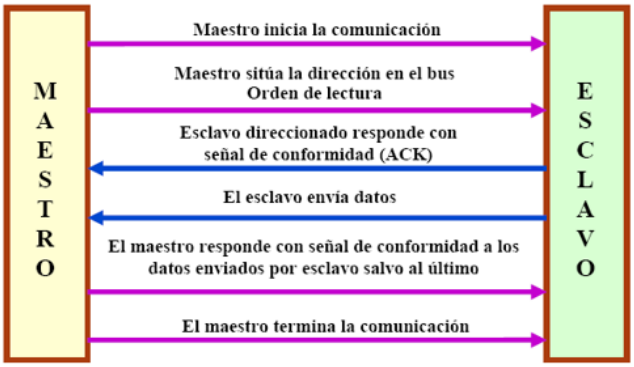


Ilustración 15: Diagrama de protocolo I2c, lectura de datos Master-Slave

El esquema que se expone a continuación, muestra la conexión I2c del proyecto.

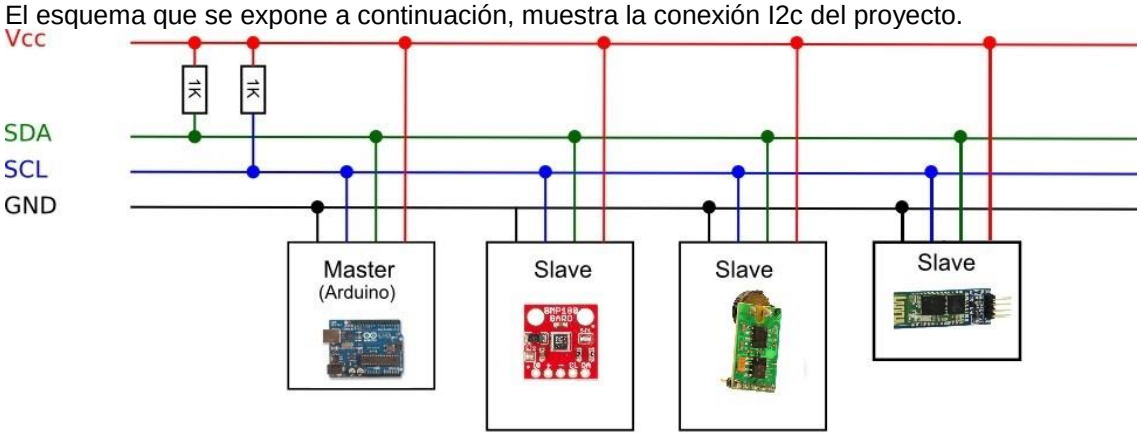


Ilustración 16: Esquema conexión I2c

Como muestra el esquema y siguiendo las especificaciones del protocolo, es recomendable añadir resistencias pull-up, aunque es posible conectar los pines SDA y SCL directamente. En el caso de este proyecto debido a que no es necesario se conecta directamente, ya que los propios dispositivos disponen de estas resistencias.

Arduino UNO soporta esta conexión de fábrica e indica en las especificaciones que los pines I2c tienen que ser conectados en los pines analógicos A4 (SDA) Y A5 (SCL).

3.2. Planos de conexión y funcionamiento

En los sub-apartados que preceden se muestra y explican los planos de cada módulo y del sistema general. Para la creación de planos se utiliza el software de creación de diagramas electrónicos Fritzing [18]. Debido a que los módulos no se encuentran incluidos en las librerías de Fritzing se añade a posteriori, mediante un editor de imagen.

Para facilitar la comprensión y tener una referencia de la conexión del circuito los cables, serán de un cierto color según la conexión. El patrón de colores se mantiene en todos los planos, de esta forma si un color tiene el mismo significado en diversos planos se facilita la comprensión del sistema completo.

Tabla 3: Patrón de colores de los cables.

	VCC (alimentación)
	GND(Masa)

	SCL(señal de reloj)
	SDA(Transmisión de datos)

3.2.1 Sensor de Humedad HH10D

Sensor de humedad HH10D, como se explicó este sensor utiliza el bus de comunicación I2c y es alimentado con 3.3V. En las siguientes ilustraciones se muestra el esquema de montaje de este dispositivo a Arduino y se realiza una explicación del sistema.

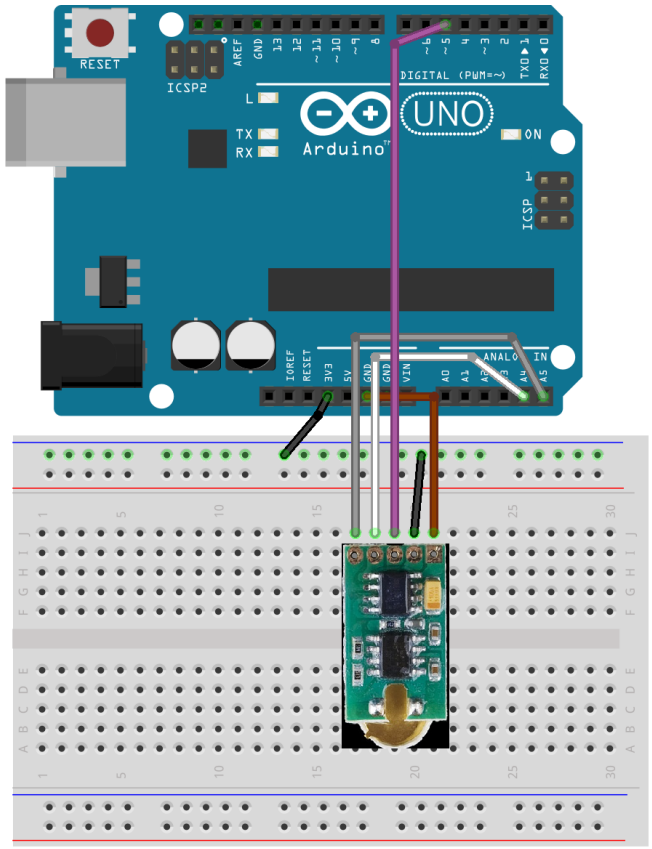


Ilustración 17: Plano de conexión HH10d-Arduino

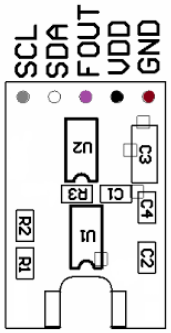


Ilustración 18: Plano de conexión HH10D, identificación de colores con los pines

Teniendo en cuenta los planos anteriores, a continuación, se explicará el significado de cada conexión correspondiente a cada pin.

Comenzare explicando el sistema de alimentación y control del circuito (conexión realizada con el cable negro y el cable marrón). El pin GND de HH10D es conectado mediante el **cable marrón** a unos de los pines GND de Arduino. Con esta conexión se consigue establecer un sistema de control de perturbaciones que pueda afectar al funcionamiento del mismo. Mientras que con el **cable negro** se conecta el pin VDD al pin 3,3V de Arduino, mediante esta conexión se consigue que HH10D obtenga energía para su funcionamiento (conexión de alimentación).

Para obtener el valor de humedad es necesaria la conexión establecida con el **cable morado.** Conexión que conecta el pin FOUT de HH10D con el pin digital 5. Por esta se emite una señal cuadrada de frecuencia variable la cual varia su frecuencia en relación a la humedad que capta en cada instante. Por lo tanto, es necesario saber el valor de esta frecuencia para obtener el valor de la humedad relativa.

La conexión blanca y gris establecen la conexión del bus I2c. La señal de clock corresponde al **cable gris**, uniendo entre si el pin SCL y el pin analógico 5 de Arduino. Mientras que la transmisión de datos se realiza por la conexión del cable blanco que une el pin SDA con el pin analógico 4 de Arduino.

3.2.2 Sensor de temperatura y presión Bmp180

Sensor del cual obtenemos temperatura y presión. En este apartado seré más breve ya que 4 de estas conexiones son similares al anterior.

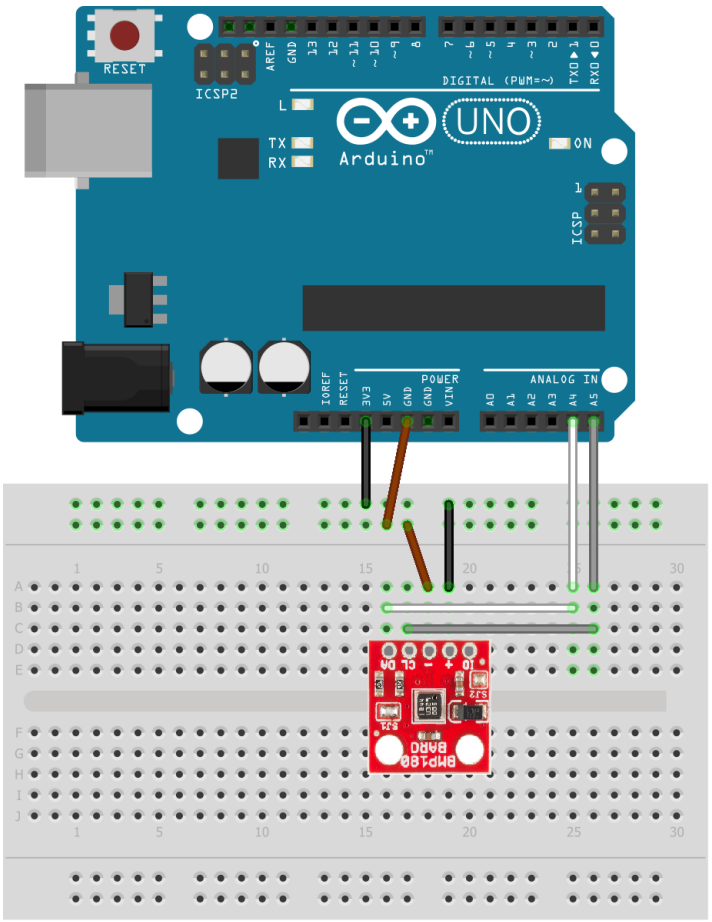


Ilustración 19: Plano de conexión Bmp180-Arduino

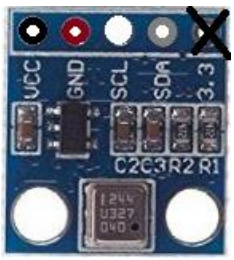


Ilustración 20: Plano de conexión bmp180, identificación de colores con los pines

Bmp 180 también transmite la información mediante el bus I2c, al igual que el anterior caso se conectan el cable gris (pin de datos, SDA) y blanco (pin de datos, SLC) a los pines analógico 4 y 5 de Arduino.

En cuanto a masa y la alimentación son conectados a Arduino mediante el cable marrón y negro a los pines de Arduino GND y 3,3V.

3.3.3 Módulo de comunicación Bluetooth (HC-06)

Módulo de comunicación bluetooth HC-06, en los siguientes párrafos se muestra el plano de conexiones junto a una explicación.

Antes de comenzar explicando el montaje que se ha llevado a cabo de este módulo se comentara brevemente en que se basa la tecnología Bluetooth. La tecnología inalámbrica Bluetooth es una tecnología basada en ondas de radio de corto alcance que funcionan a una frecuencia de 2.4Ghz. Permite comunicaciones de audio y datos entre diferentes dispositivos a una distancia que comúnmente ronda los 10 metros, aunque se puede llegar a 100 metros dependiendo de la potencia del dispositivo.

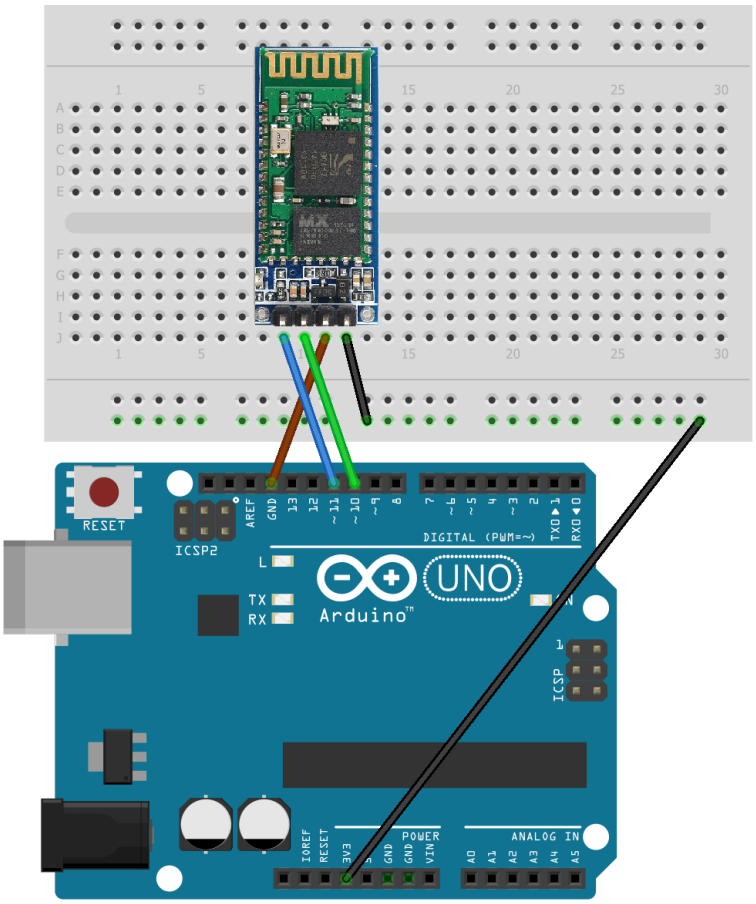


Ilustración 21: Plano de conexión HC-06-Arduino

Ilustración 22: Plano de conexión HC-06, identificación de colores con los pines

Al igual que en los anteriores módulos la alimentación y la masa se conecta de forma similar. En este caso el pin GND se conecta al pin digital de Arduino UNO, debido a que los pines del apartado de potencia ya están ocupados por los dos anteriores sensores.

En cuanto a los pines RXD y TXD, son los encargados de la transferencia de datos. La conexión se realizada mediante el **cable verde,** encargada de transmitir la información procedente de Arduino UNO, en este caso el pin digital 10 a HC-06. Mientras que la conexión establecida con el **cable azul** se encarga de comunicar la información procedente del dispositivo periférico (en este caso un Smartphone) al pin digital número 11 de Arduino UNO.

El módulo HC-06 informa mediante un led si el dispositivo está vinculado o no a otro dispositivo. Si el led parpadea quiere decir que no hay ningún dispositivo conectado, mientras que si se mantiene una luz continua quiere decir que se ha establecido la conexión con algún dispositivo.

3.3.4 Plano completo

En este apartado se muestra el plano general del sistema, también se incluirá una imagen en planta del sistema físico. Este plano se ha diseñado con el mismo orden de conexiones que el proyecto físico y una estructura lo más semejante posible.

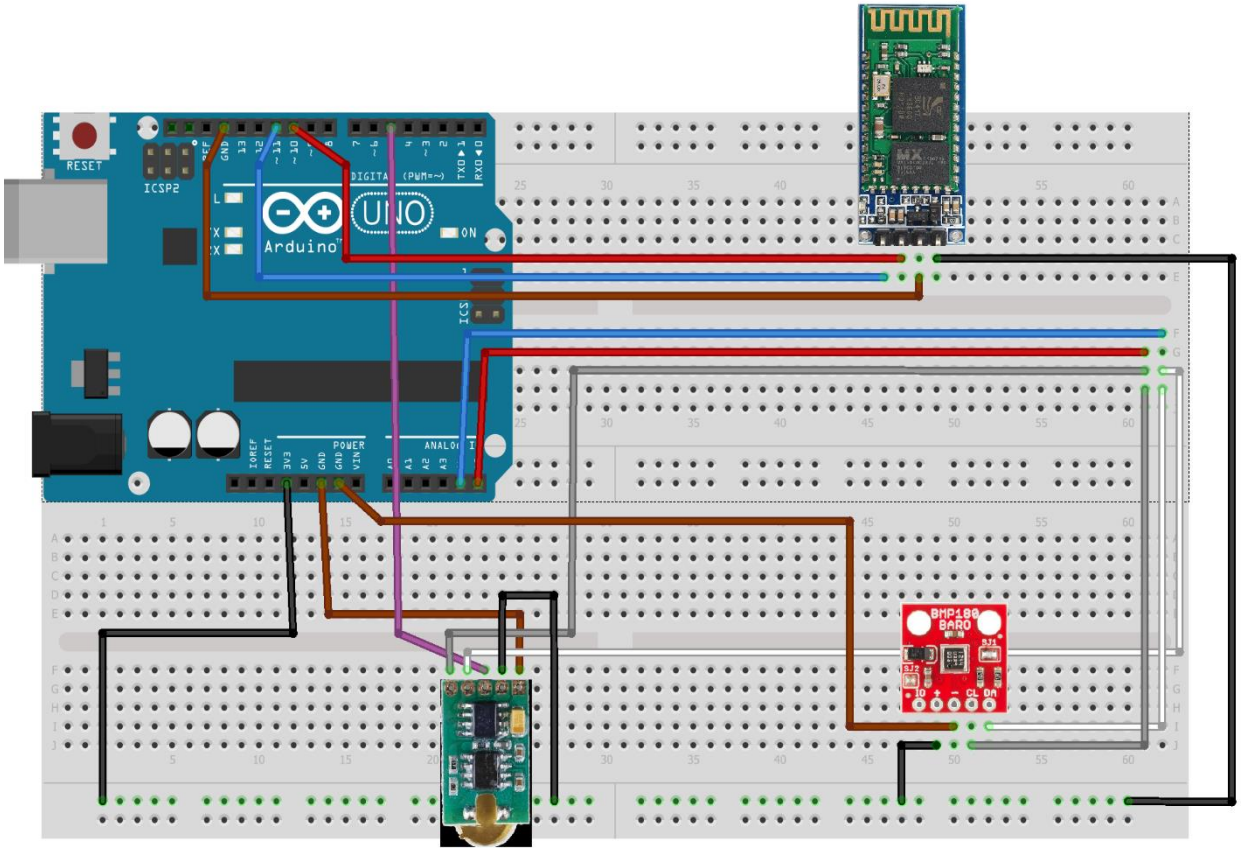


Ilustración 23: Plano de montaje completo.

Debido a que los tres módulos comparten conexiones es necesario llevar ciertas conexiones de Arduino a ranuras auxiliares de la Protoboard y conectar las diferentes ramas de esta a los módulos.

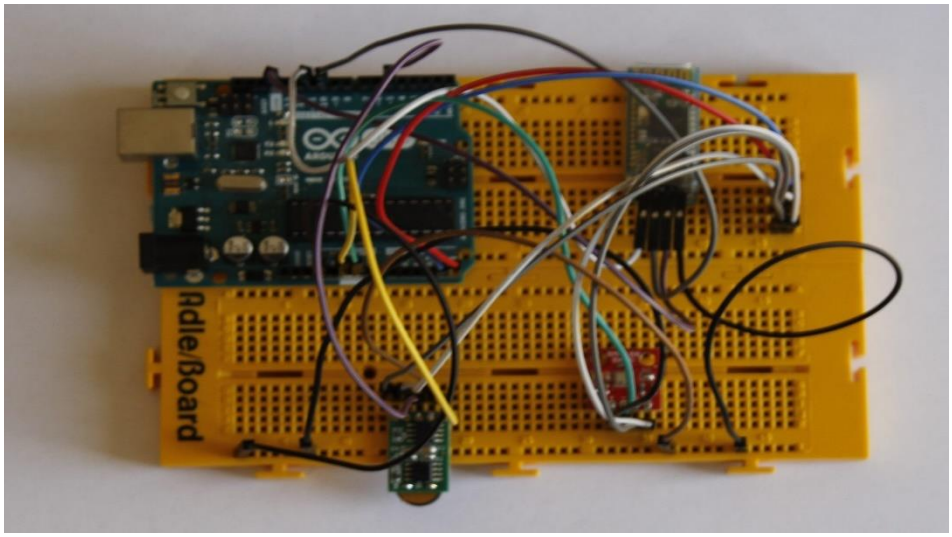


Ilustración 24: Vista de planta del montaje físico

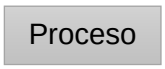
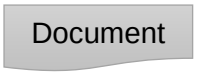
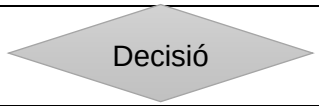
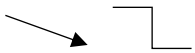
Como comentario decir que la placa Arduino UNO fue pegada a la protoboard mediante una masilla y los módulos conectadas a esta mediante tiras de pines macho con un paso de 2.54mm.

4.ALGORITMOS

En este apartado se presenta los algoritmos que han ayudado a la implementación del código del proyecto, para la realización de los mismos se diseñan diagramas de flujos usando la herramienta yEd Graph Editor [20]. Debido a que hay diversos algoritmos y no siguen un orden lógico se han fragmentado en 3 partes: Librería, Arduino y App Invenor 2.

Antes de nada, se realiza una breve explicación de los diagramas de flujos y sus símbolos. El objetivo de un algoritmo es representar mediante un diseño la tarea computacional que resuelve un cierto problema, para llevar a cabo esta representación se utiliza los diagramas de flujo. Un diagrama de flujo es una representación gráfica de un algoritmo que utiliza un conjunto de símbolos estandarizados por organizaciones tales como ANSI (American National Institute) e ISO (International Standard Organization). En la siguiente tabla se muestra los símbolos que se utilizan en los diagramas de flujo junto a una explicación.

Tabla 4: Simbología, diagrama de flujo

	Indica el inicio o término del diagrama.
	Indica una instrucción que debe realizar el computador.
	Ingreso y salida de datos.
	Salida de datos(escritura).
	Proceso cuyo diagrama de flujo se encuentra en otro lugar.
	Indica operaciones lógicas o de comparación entre datos.
	Elementos de conexión e indicación del orden de la ejecución de las instrucciones del diagrama.
	Conector que enlaza dos partes de un mismo diagrama.
	Conector que enlaza dos partes de dos diagramas representados en diferentes páginas.

Debido a que el número de diagramas es elevado y algunos de estos están conectados entre sí mediante el símbolo de unión entre diagramas  , se hará uso de una tabla indicando este tipo de conexiones para facilitar la comprensión.

Para una mayor brevedad de escritura nos referiremos al sensor de temperatura y presión con Bmp180, al sensor de humedad HH10D y al módulo Bluetooth HC-06. Esta nomenclatura se mantendrá tanto en esta tabla como en algún punto de este apartado. En la tabla se indicará a la izquierda la letra o número que tiene asignado el diagrama en cuestión y a la derecha el nombre del sensor junto al método, programa o datos llevado a cabo en el diagrama (separado con una _). Para distinguir bien si el símbolo apunta de la librería a la misma librería, código de Arduino a la librería o App Invetor 2 a Arduino se asignará un símbolo y un color a cada uno.

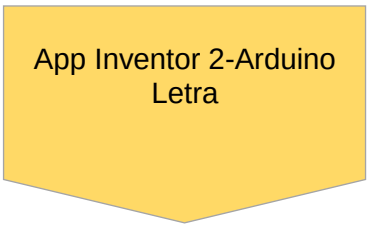
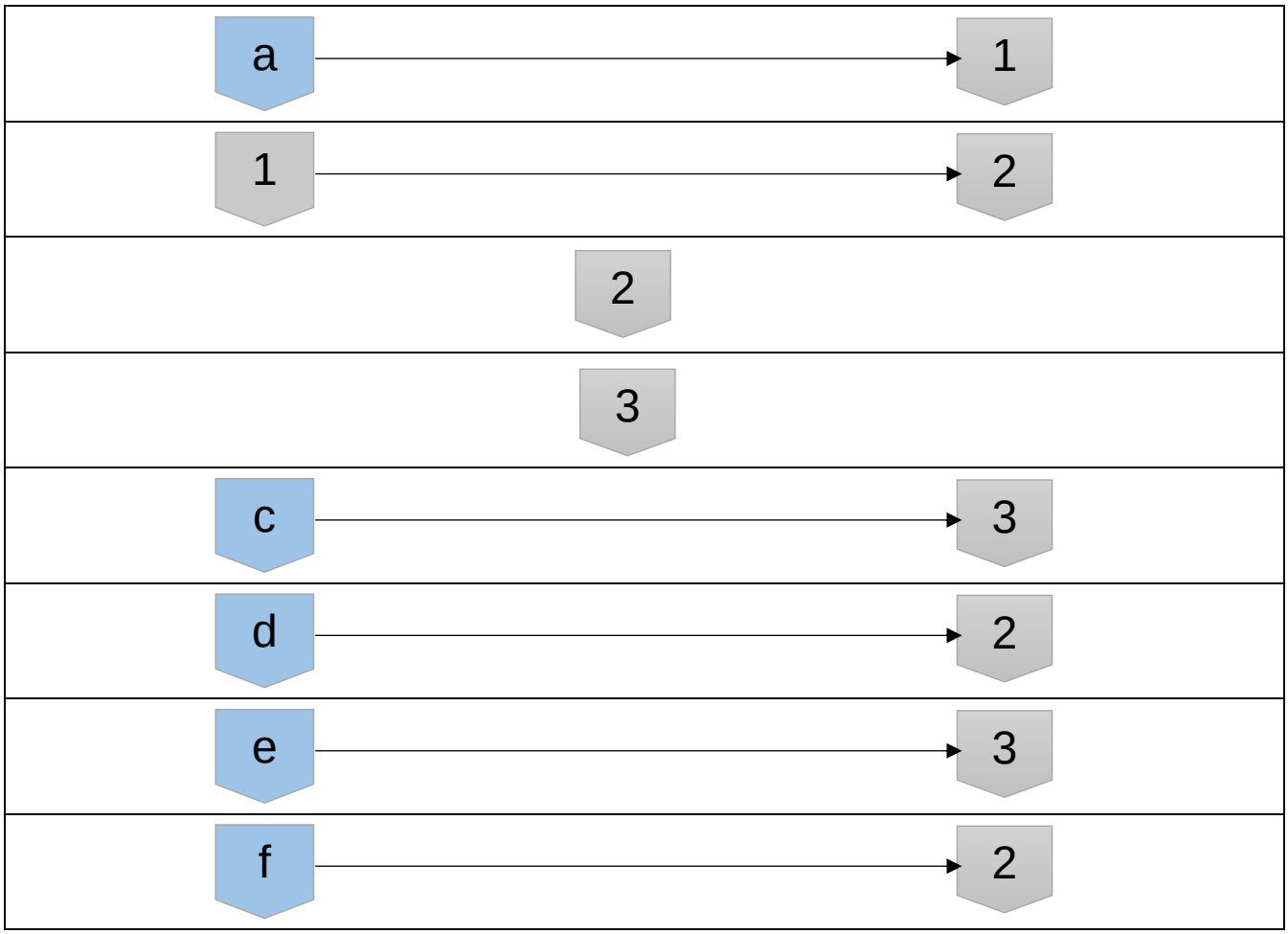
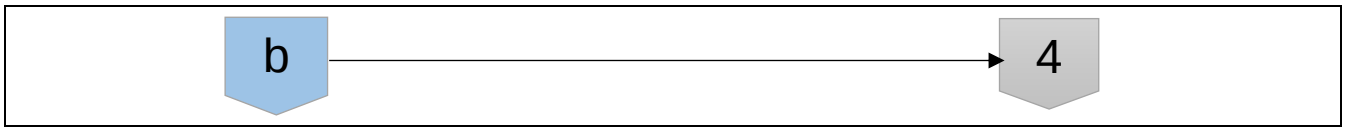


Tabla 5: Tabla de conexión entre diagramas

a	Bmp180_begin	2	Bmp180_readBytes
1	Bmp180_readInt	3	Bmp180_writeBytes
1	Bmp180_readUInt	c	Bmp180_startTemperature
d	Bmp180_getTemperature	e	Bmp180_startPressure
f	Bmp180_getPressure	b	HH10D_begin
4	HH10D_i2cReabBytes	g	Bmp180_HumedadRead
h	AppInventor_ Envió de caracteres (t,h y p)		

Tabla 6: Tabla de conexión entre diagramas, especificado





4.1 Algoritmos de librería.

La librería tiene como principal función hacer que el programa principal tenga un menor código posible utilizando clases y métodos de esta. En las siguientes ilustraciones se mostrarán los diagramas de flujos de los métodos que componen esta librería junto a una pequeña explicación de la función del mismo. Debido a que el número de métodos es muy amplio se ha decidido usar un anexo con algunos de los métodos que no se han considerado principales.

Consultar fichero “Ficheros anexos/AlgoritmosLibrería”, en los siguientes párrafos nos referiremos con anexo a este en concreto.

Método **Bmp180_begin**: el propósito de este método es poner en marcha el sensor de temperatura y presión (Bmp180). Para ello arranca la librería Wire de arduino (bus I2c) y recibe los datos de calibración mediante los métodos readInit y readUnit que leen un entero con signo y sin signo. Como se ha dicho anteriormente estos métodos están explicados y expuestos en el anexo antes citado. Por último, si los datos de calibración son obtenidos correctamente se devuelve un 1 si en caso contrario ha surgido algún error devuelve un 0.

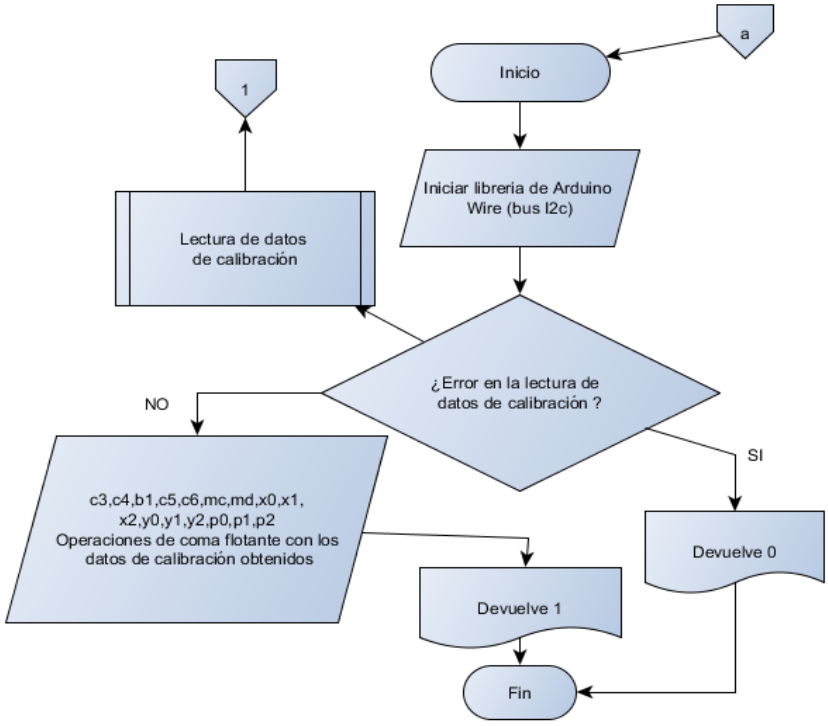


Ilustración 25: Diagrama de flujo, Bmp180_begin

Método **Bmp180_startTemperature:** encargado de inicializar la lectura de datos de temperatura. Para ello necesita comunicar al sensor que quiere tomar los datos mediante una serie de bytes especificados en el datasheet. El método encargado de escribir estos bytes en el dispositivo es writeBytes (explicado en el anexo). Si se ha producido algún error al escribir los bytes devuelve un 0, mientras que si el proceso se ha completado con éxito de vuelve un retardo de 5 ms.

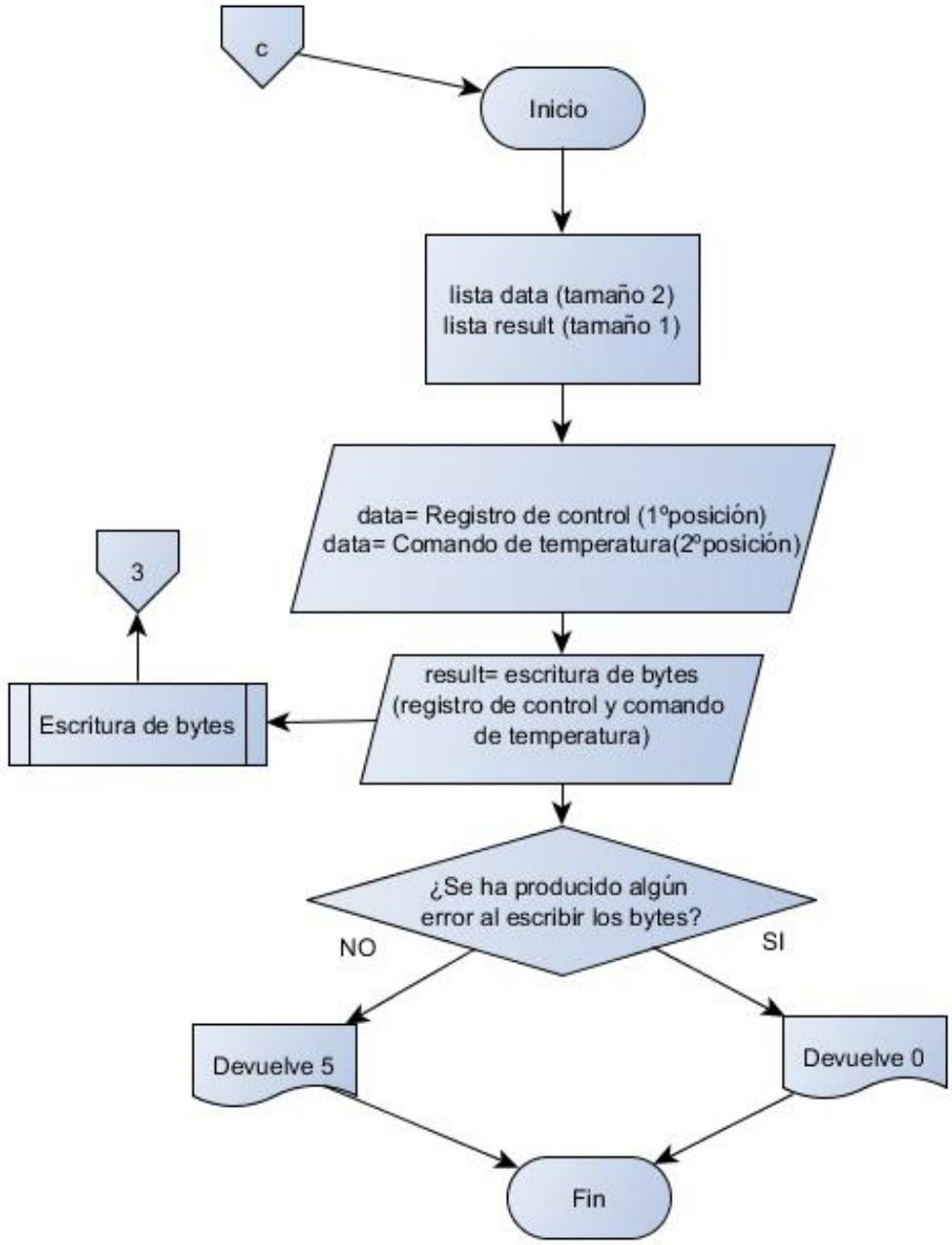


Ilustración 26: Diagrama de flujo, Bmp180_startTemperature

Método **Bmp180_getTemperature**: la función de este método es devolver la temperatura (ºC) capturada por el sensor. Mediante el registro *RESULT* indicado en el datasheet y el método readBytes explicado en el anexo, se indica al dispositivo que se quiere obtener dicha medida.

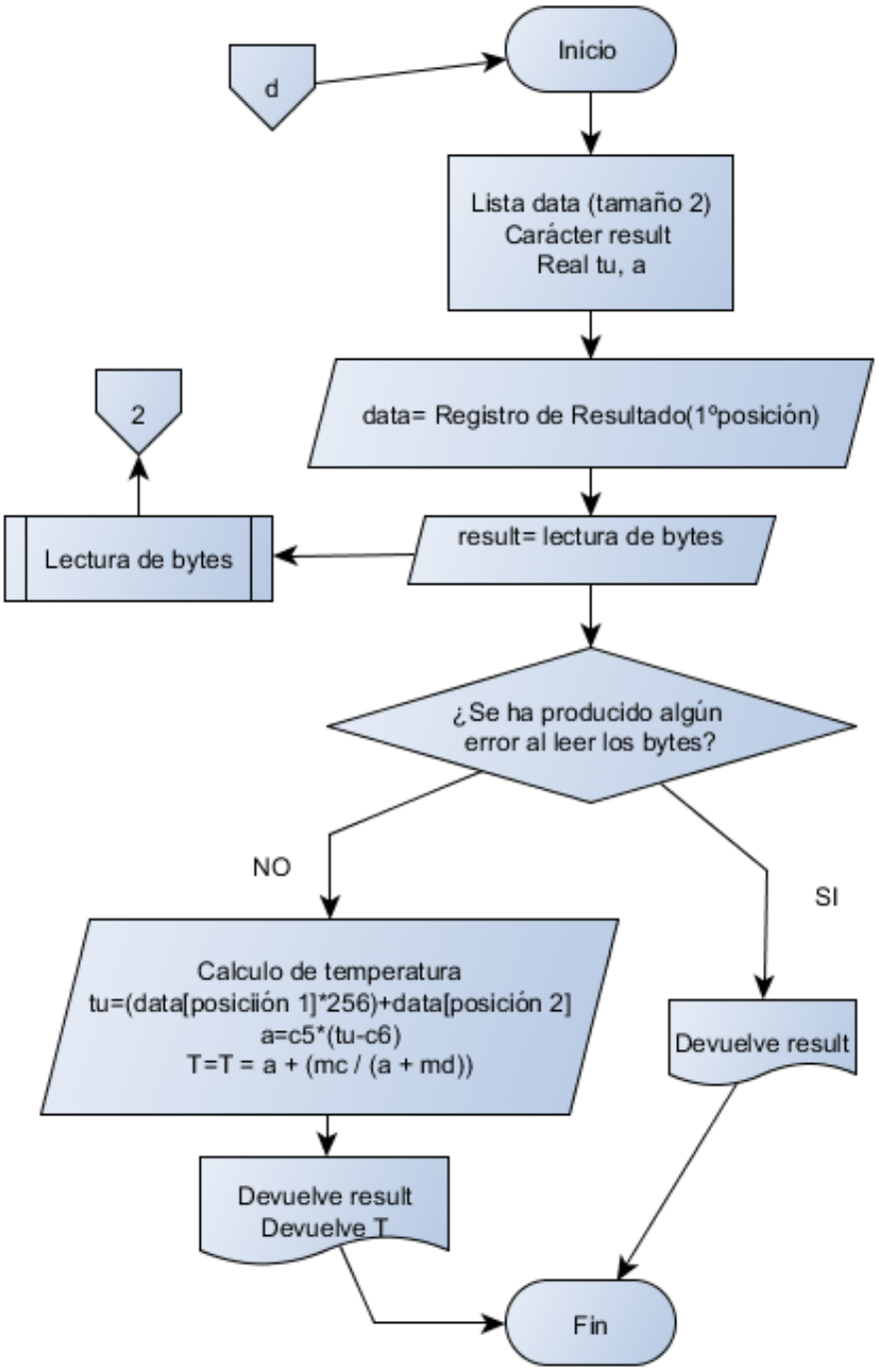


Ilustración 27: Diagrama de flujo, Bmp180_getTemperature

Método **Bmp180_startPressure:** este método tiene la función de arrancar la lectura de datos del sensor de presión. En éste se puede indicar el valor de muestreo que se le quiere asignar a la lectura devolviendo un tiempo de espera en ms en función de este valor. Según el tipo de sobre muestreo que le indiquemos se escribe un comando de 1 byte en el dispositivo indicando esta operación. Al igual que en startTemperature se hace uso del método writeBytes (explicado en el anexo). Si la escritura se ha ejecutado correctamente devuelve el tiempo en ms que debe esperar para la toma de medidas, mientras que si la escritura del byte ha sido fallida devuelve un 0.

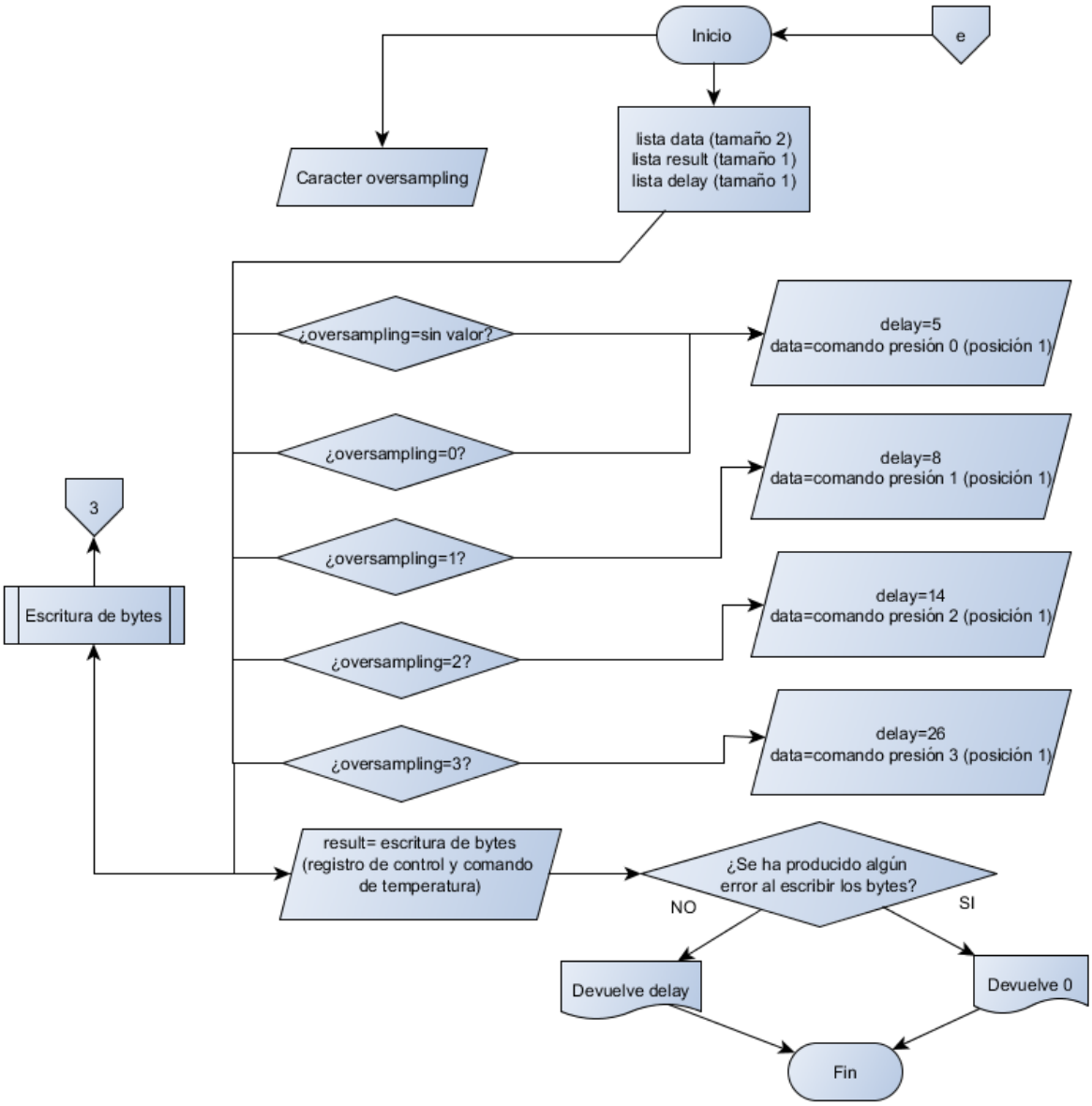


Ilustración 28: Diagrama de flujo, Bmp180_startPressure

Método **Bmp180_getPressure:** Método encargado de devolver el valor de presión (mb). Para ello igual que en el caso de temperatura, envía el comando de registro *RESULT* para la obtención del valor. Para leer los bytes de información se utiliza el método readBytes especificado en el anexo.

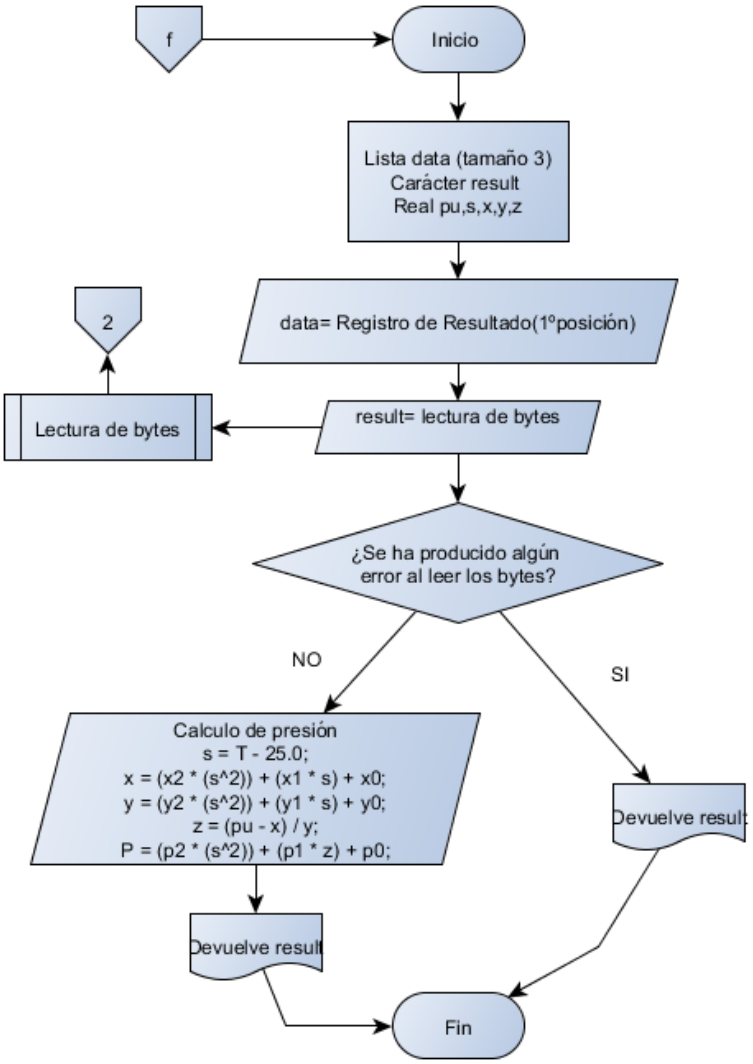


Ilustración 29: Diagrama de flujo, Bmp180_getPressure

Método **HH10D_begin:** Puesta en marcha del sensor de humedad HH10D, en este método se lee dos datos necesarios para el cálculo de la humedad relativa y se define el intervalo de lectura de la frecuencia (librería FreqCount de Arduino). Los datos leídos desde la memoria EEPROM son la sensibilidad y el offset, para poder comunicarse con el sensor se utiliza el método i2c_Read2Bytes expuesto en el siguiente método.

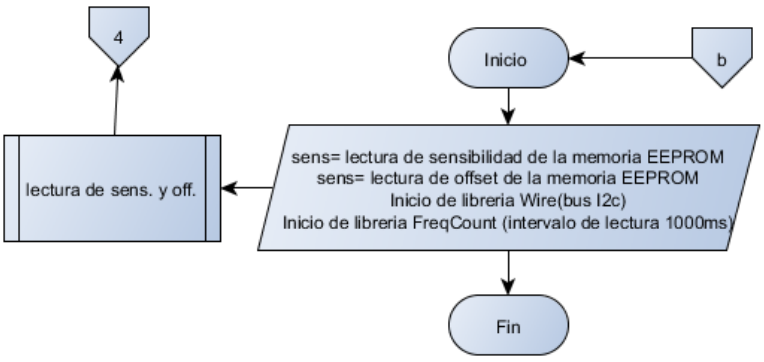


Ilustración 30: Diagrama de flujo, HH10D_begin

Método **HH10D_ i2c_Read2Bytes:** Método encargado de la comunicación bus i2c, Arduino nos facilita el proceso con una librería llamada *Wire*. El proceso que se lleva a cabo para establecer esta comunicación se basa en tres puntos fundamentales:

1. Establecimiento de transmisión de datos con el dispositivo.
2. Solicitud y disponibilidad de datos con el dispositivo.
3. Lectura de datos.

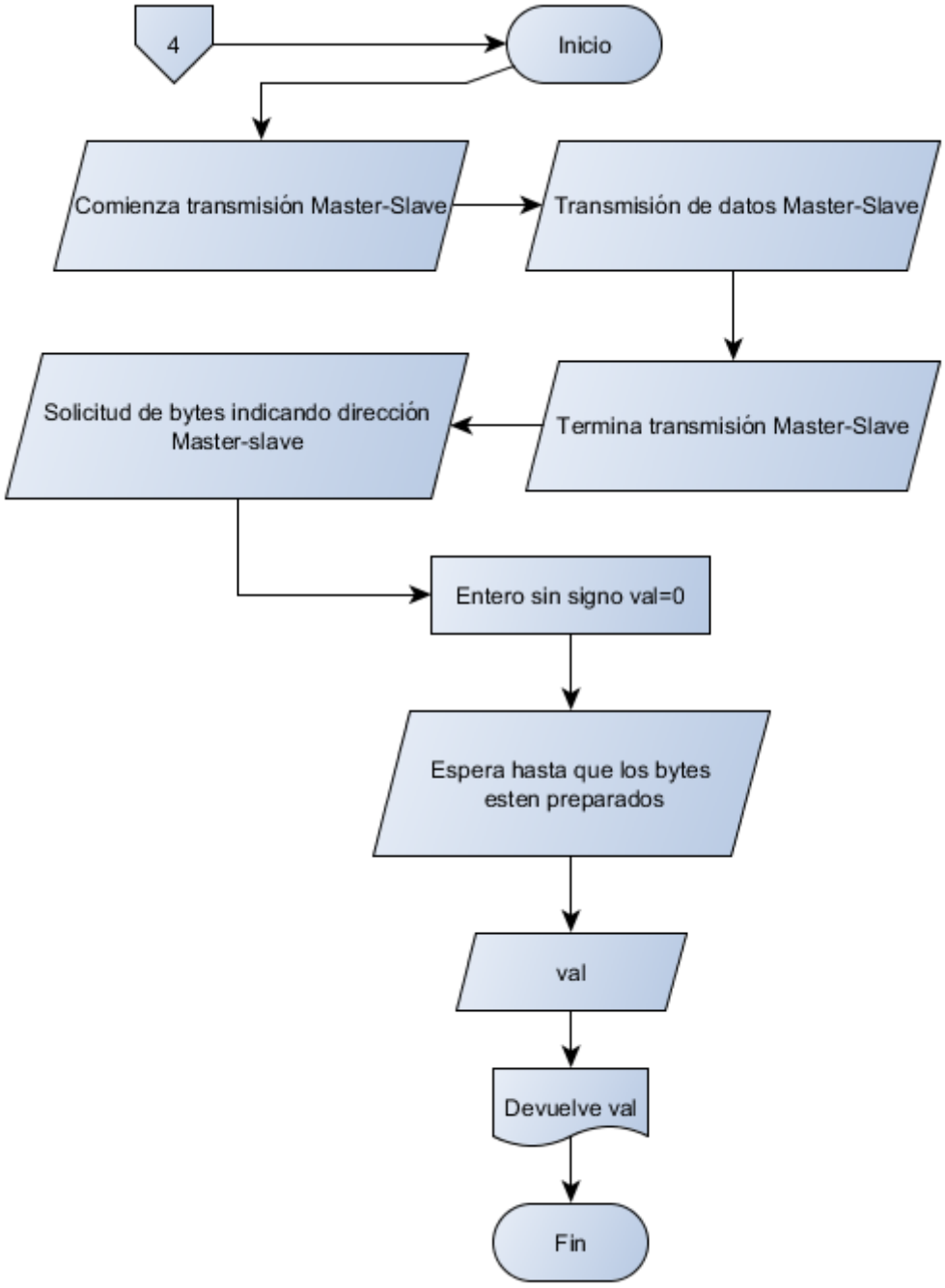


Ilustración 31: Diagrama de flujo, HH10D_i2creab2Bytes

Al terminar el proceso devuelve el valor(entero) de los bytes obtenidos.

Método **HH10D_humedadRead:** método encargado de la lectura de los datos de humedad relativa. Para realizar el cálculo de esta se necesita los valores anteriormente obtenidos en el método begin (sensibilidad y offset), junto al valor de frecuencia que se obtiene con los métodos de la librería *FreqCount* de Arduino. Finalmente, si la lectura de frecuencia está disponible, se calcula la humedad relativa y se devuelve este dato. En cambio, si la frecuencia no está disponible devuelve un 0.

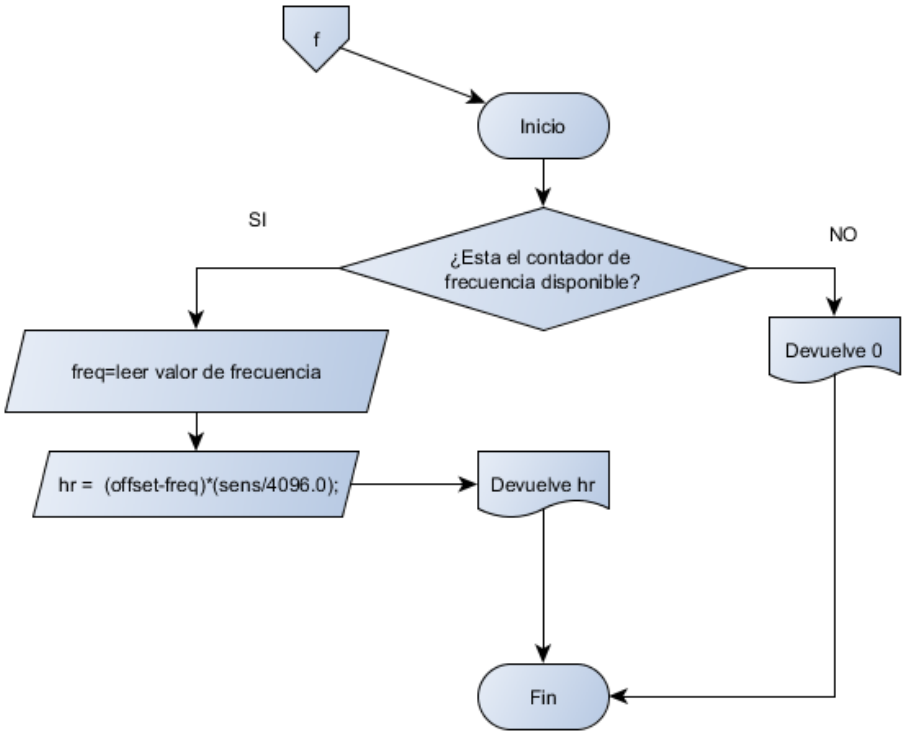


Ilustración 32: Diagrama de flujo, HH10D_humedadRead

4.2 Algoritmo de Arduino.

Algoritmo principal del proyecto, este es el encargado de la puesta en marcha del sistema encargando de inicializar, recibir y enviar datos a los dispositivos conectados.

El algoritmo está compuesto por tres partes diferenciadas:

1. **Inicialización:** en esta parte se inicializarán e importaran librerías, se declaran las clases provenientes de la librería y se pondrán en marcha los dispositivos.
2. **Toma de datos:** en la toma de datos se reciben y muestran (monitor serie de Arduino, explicado en el apartado de implementación) los datos provenientes de los sensores. Para ello se utilizarán los métodos de recepción de datos de la librería antes expuesta.
3. **Comunicación Bluetooth:** Comunicación con el módulo bluetooth. Transmisión de datos a la aplicación generado con app Inventor 2 (los datos serán visibles en el Smartphone).

A continuación, se mostrará el diagrama de flujo general de Arduino. Debido a que es grande y ciertas partes pueden ser poco visibles. Se ha fragmentado en 5 partes que se expondrán por separado. A continuación, se muestra la tabla indicando cual es el identificativo de cada apartado.

Tabla 7: Diagrama de flujo de Arduino, Fragmentación por colores.

	Inicialización
	Toma de medidas de temperatura
	Toma de medidas de presión
	Toma de medidas de humedad
	Comunicación Bluetooth

Para poder observar los apartados por separado junto a la explicación consultar el fichero “Ficheros anexos/Diagrama de flujo de Arduino”

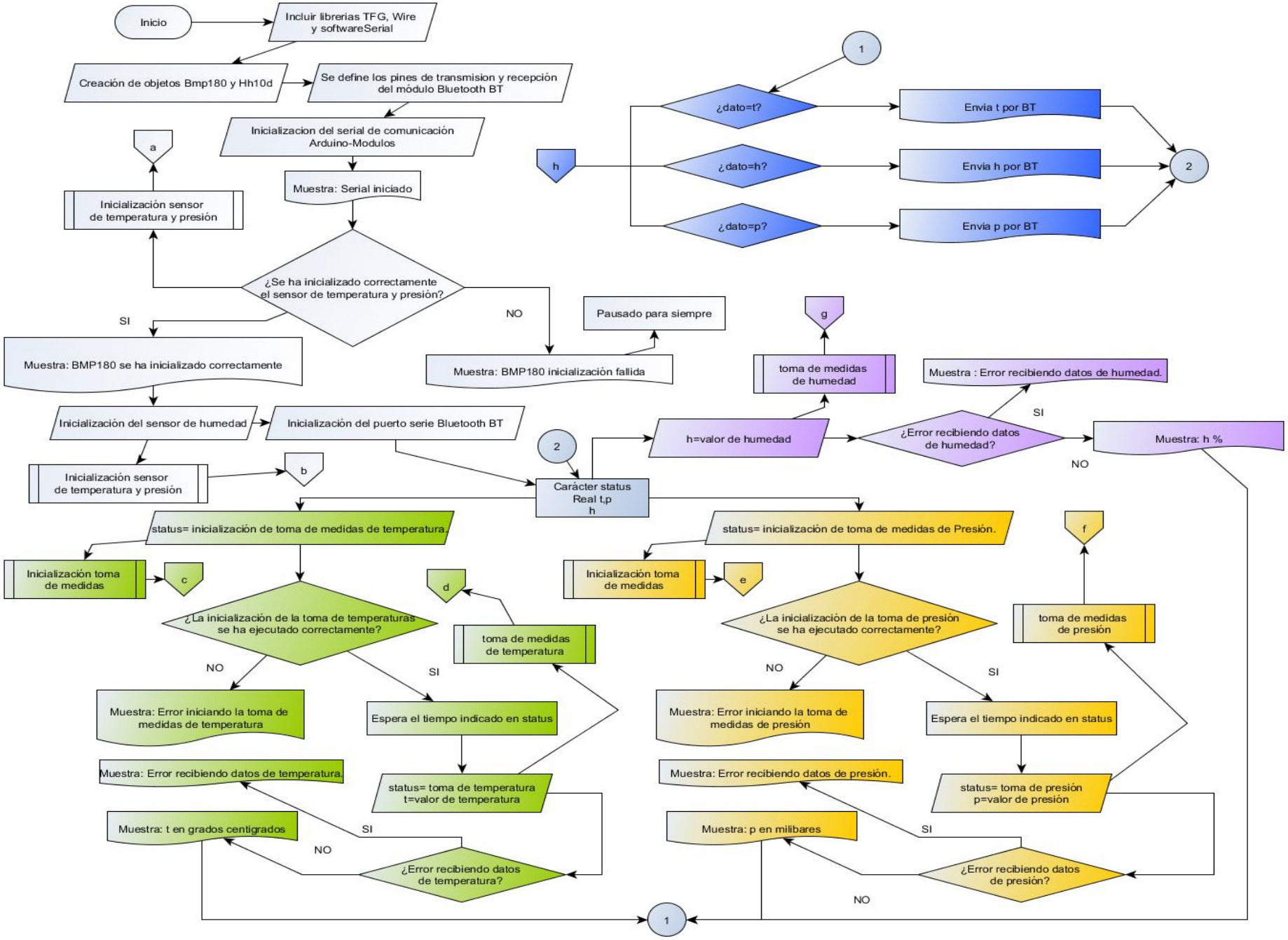


Ilustración 33: Diagrama de flujo, Arduino

Como comentario decir que el diagrama no contiene un fin si no que es un loop (explicado en siguiente apartado) constante a partir de la toma de medidas, es decir, existe un loop en los apartados: verde, amarillo, violeta y azul.

4.2 Algoritmo de App inventor 2

Encargado de recibir la información del módulo Bluetooth y representar los datos en un dispositivo Android, en este caso un Smartphone. Principalmente este diagrama se encarga de dos cosas: de conectar o desconectar la conexión con el módulo Bluetooth (HC-06) y de la recepción de datos controlado por un reloj. Debido a que se pretende recibir numerosos datos simultáneamente es necesario un control para que los datos no se mezclen entre sí, para ello se utiliza cambios de estado. Por ejemplo, si la temperatura tiene un estado actual de 0 y se quiere recibir ahora el valor de presión el valor de estado cambia a 1 una vez recibido el dato de temperatura y se comprueba que el estado es ahora 1. Mediante el diagrama de flujo expuesto a continuación se observará de una forma más clara el objetivo de este algoritmo.

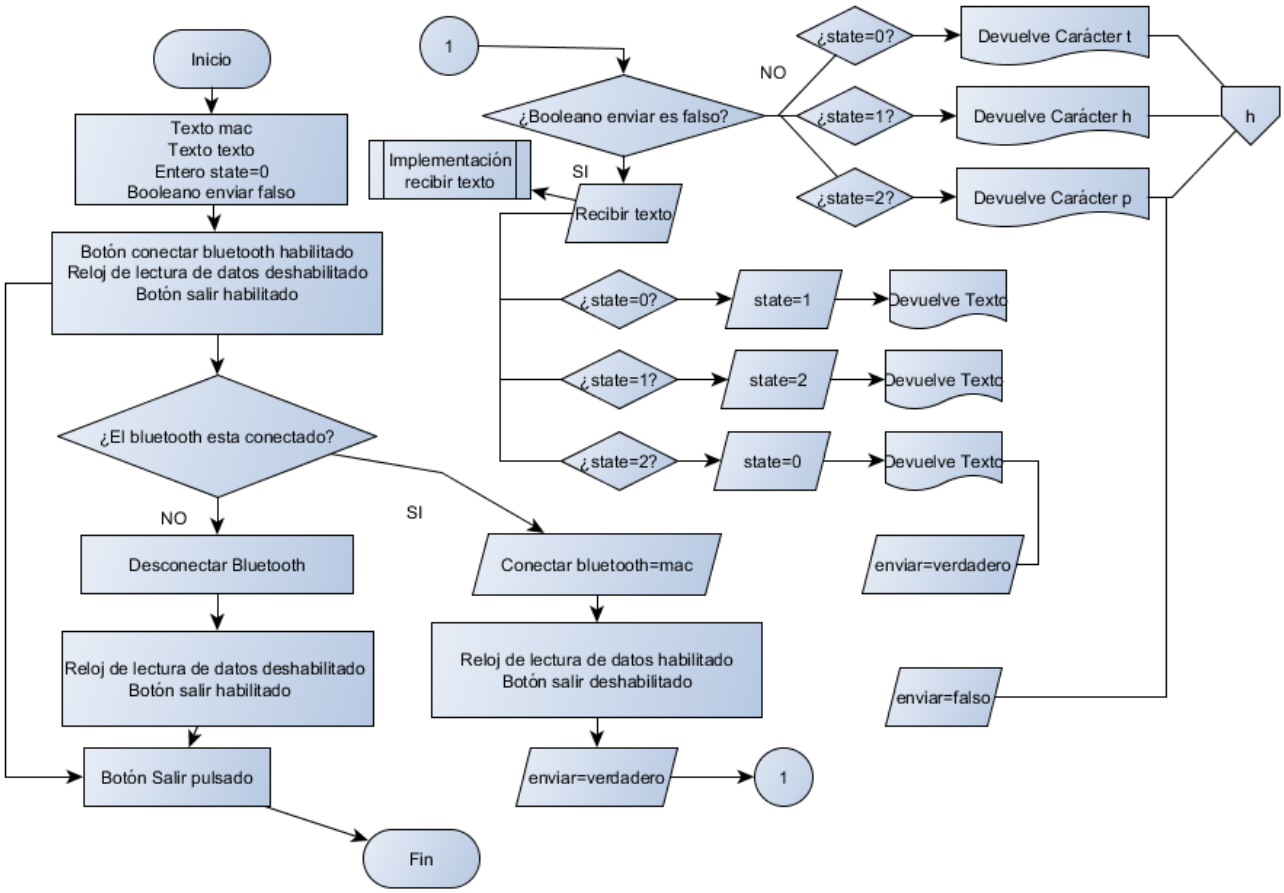


Ilustración 34: Diagrama de flujo, App Inventor 2

5. IMPLEMENTACIÓN

Una vez se tiene la parte del diseño (algoritmos) completado, se procede a la implementación del código basándonos en este. Este apartado es el encargado de mostrar dicha implementación junto a la estructura y creación e inicialización del proyecto en cada plataforma. Siguiendo el orden del anterior apartado se explica en primer punto el código de la librería, en segundo el de Arduino y por último el código de App Inventor 2. Antes de comenzar explicando los siguientes puntos, es necesario indicar cuál es la estructura general del proyecto en cuanto a directorios y las librerías que se han utilizado.

Los ficheros que incluyen el código comentado están disponibles en el fichero "Ficheros anexos/TFG.h,TFG.cpp y Weather_station.ino"

Estructura de directorios del proyecto**,** separado en 2 partes principales una compuesta por Arduino junto a la librería y otra compuesta por el fichero App Inventor 2 contenido en la nube.

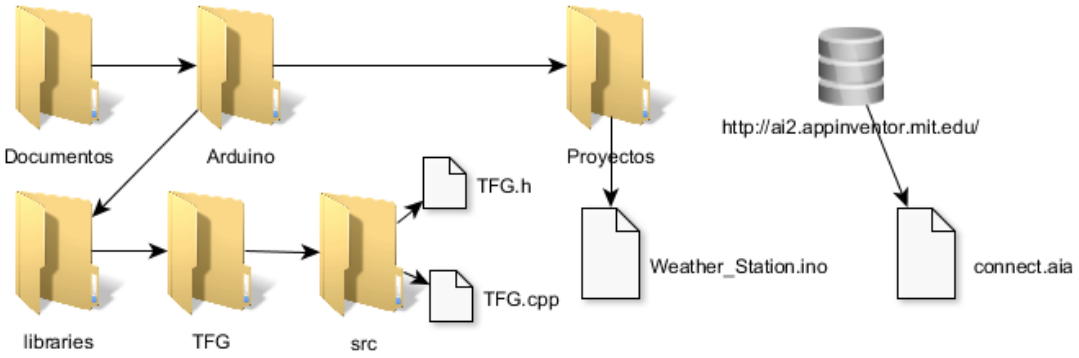


Ilustración 35: Directorio de documentos del proyecto

En programación es muy común el uso de **librerías** debido a que muchos de las soluciones que buscamos en nuestro proyecto ya han sido soportadas por alguna corporación o particular. De esta forma ahorramos tiempo y espacio de código. El principal objetivo de una librería es ahorrar líneas de código en el documento principal o *main* para ello utilizamos clases y métodos de estas. En este proyecto se han utilizado tanto librerías ya implementadas por Arduino como una propia llamada TFG. A continuación, se muestra en una tabla las librerías utilizadas junto a una pequeña explicación.

Tabla 8: Librerías utilizadas en el proyecto

Wire	Librería de Arduino, permite la comunicación I2c con dispositivos.
stdio	Librería de C, es una biblioteca estándar que define tres tipos de variable, macros y diversas funciones para realizar la entrada y salida de datos.
math	Librería de C++, contiene numerosas funciones matemáticas que podemos hacer uso de ellas.
FreqCount	Librería encargada de medir la frecuencia de la señal contando el número de impulsos en un rango de tiempo.
SoftwareSerial	Librería de Arduino, permite la comunicación serial en los pines digitales de Arduino.
TFG	Librería propia del proyecto que incluye los métodos de los sensores utilizados en el fichero .ino de Arduino.

5.1 Código de Librería.

En este apartado se muestra y explica las partes fundamentales del código constituido por la librería *TFG.* Esta librería se creó con el objetivo de ahorrar código en el fichero principal de Arduino *WeatherStation.ino.* Antes de comentar el código es necesario explicar cómo se crea este tipo de ficheros basados en lenguaje c++ y la estructura junto a un diagrama de bloques (basado en lenguaje UML) indicando

cuales son los métodos que conforman las clases, sus entradas y salidas. Para el desarrollo del diagrama se utiliza yED Graph Editor [20].

Para más información sobre lenguaje UML y observar el diseño UML de la librería consultar "Anexos/diseñoUML"

Como comentario decir que se eligió realizar una librería en c++ debido a que la información sobre cómo llevar a cabo la programación de los sensores estaba en este lenguaje.

Para **crear una librería de c++** es necesario generar dos tipos de ficheros dentro de una carpeta (llamada TFG en este caso), el fichero .h y el fichero .cpp.

- **Fichero .h:** Archivo de encabezado los cuales contienen solamente las declaraciones tanto de métodos, constantes, variables y clases. También pueden incluirse las librerías importadas, aunque en mi caso preferí añadirlo en el fichero que se explica a continuación.
- **Fichero .cpp:** Archivo de implementación, en él se implementa el código para las funciones declaradas en el archivo .h.

A continuación, se muestra el código utilizado en la librería TFG para crear este tipo de estructura. Tanto el nombre del archivo .h como el archivo .cpp deben tener el mismo nombre que la carpeta donde han sido creados.

Archivo .h, **TFG.h:**

```
#ifndef TFG_h
#define TFG_h
```

Ilustración 36: Código librería, inicio fichero header

```
#endif
```

Ilustración 37: Código librería, cierre fichero header

Entre estas dos sentencias se encuentra la declaración de clases, métodos, etc.

Archivo .h, **TFG.cpp:**

```
#include <TFG.h>
```

Ilustración 38: Código librería, importando header fichero de implementación

Se importa el archivo header, de esta forma se dispone de variables y declaraciones instanciadas por este.

Una vez presentado los puntos anteriores pasamos a explicar las partes principales del código de la librería TFG. Básicamente esta librería se basa en tres puntos fundamentales:

1. **Inicialización** de los dispositivos.
2. **Lectura y escritura** de datos con los dispositivos
3. **Captura y operación** de datos meteorológicos.

1. Inicialización: Para ello ambos dispositivos cuentan con un método begin. La primera instrucción que realizan ambos dispositivos es la inicialización de la librería Wire.

```
Wire.begin();
```

Ilustración 39: Código librería, inicialización de la librería Wire

Librería que como se comentó anteriormente permite la comunicación I2c.

El siguiente paso es la calibración del dispositivo, la recepción de los valores que permiten el cálculo del dato meteorológico a obtener. Ambos utilizan métodos privados para obtener dichos valores, estos métodos son ligeramente distintos entre ambos dispositivos, pero el objetivo es el mismo (consultar ficheros de código comentado para más información).

```
if (readInt(0xAA,AC1) &&
    readInt(0xAC,AC2) &&
    readInt(0xAE,AC3) &&
    readUInt(0xB0,AC4) &&
    readUInt(0xB2,AC5) &&
    readUInt(0xB4,AC6) &&
    readInt(0xB6,VB1) &&
    readInt(0xB8,VB2) &&
    readInt(0xBA,MB) &&
    readInt(0xBC,MC) &&
    readInt(0xBE,MD))
{
```

Ilustración 40: Código librería, calibración_Bmp180

```
// Leer la sensibilidad desde la EEPROM
sens = i2c_Read2Bytes( HH10D_ADDR, 10);
// Leer el offset desde la EEPROM
offset = i2c_Read2Bytes( HH10D_ADDR, 12);
// indicados en los datos de las expecifiaciones
```

Ilustración 41: Código librería, calibración_HH10D

2. Lectura y escritura: Obtención de los datos mediante métodos de la librería Wire. Los métodos que utilizan estas instrucciones son I2c_Read2Bytes para HH10D y readbytes y writebytes para bmp180 que estos a la vez son usados por readInt y readUint (especificado en el apartado de algoritmos.).

```
// comienza la transmisión con el dispositivo esclavo dada la  dirección de este como parámetro.
Wire.beginTransmission(BMP180_ADDR);
//transmite los datos al dispositivo esclavo en cuestión.
Wire.write(values[0]);
// Termina transimisión con el esclavo. Puede recibir multiples resultados del 0 a 4, el 0 indica que
//  la operación a concluido de forma correcta el resto de números indican un error.
_error = Wire.endTransmission();
if (_error == 0)
{
   // Utilizado por el Master(Arduino) para solicitar los bytes a Bmp180. Los bytes pueden ser recuperados
   // mediante los métodos read() y available().
   Wire.requestFrom(BMP180_ADDR,length);
   //avaliable() indica el número de bytes disponibles para poder ser leidos mediante read().
   while(Wire.available() != length) ; // espera hasta que los bytes estan preparados
   for(x=0;x<length;x++)
   {
      //lee un byte Mater-slave o slave-Master
      values[x] = Wire.read();
   }
```

Ilustración 42: Código librería, lectura de bytes de Bmp180

Para obtener los datos de interés, mediante esta serie de métodos y una dirección o comando se le indica al dispositivo cual es el valor que se quiere obtener.

3. Captura y operación: Mediante los métodos explicados anteriormente se captura el valor proveniente del sensor y con este se realizan los cálculos pertinentes para obtener el valor de humedad relativa, presión y temperatura. En el caso del cálculo de humedad relativa es diferente ya que se necesita el valor de frecuencia. Para la obtención de este dato se utilizan métodos de la librería FreqCount.

```
float HH10D::humedadRead()
{
    if (FreqCount.available())
  {
    freq = FreqCount.read();  // Leer el valor de la frecuencia

    /* Calcular la Humedad Relativa */
    float hr =  (offset-freq)*(sens/4096.0);

    return (hr);
  }
  else{return (0);}
}
```

Ilustración 43: Código librería, lectura y cálculo de datos de humedad.

Como se observa una vez está disponible el valor de frecuencia, se obtiene el valor junto a las constantes de sensibilidad y offset obtenidos con anterioridad se procede al cálculo de la humedad relativa.

5.2 Código de Arduino.

Código principal del sistema, este código es el que finalmente manda a la placa Arduino Uno a realizar las operaciones principales para que el sistema funcione correctamente (main en programación). Antes de explicar el código se explica cómo crear un proyecto en Arduino, importar librerías a este y su estructura general.

Para que nuestro Arduino pueda funcionar, el primer paso a realizar es la creación de un programa conocido como “sketch”. Una vez creado se implementa el código, se compila y vuelca en la memoria del microcontrolador.

Para **crear un proyecto** basta con iniciar la IDE de Arduino y clicar en la pestaña archivo y nuevo. Una vez creado el “sketch” en la pantalla principal se muestra:

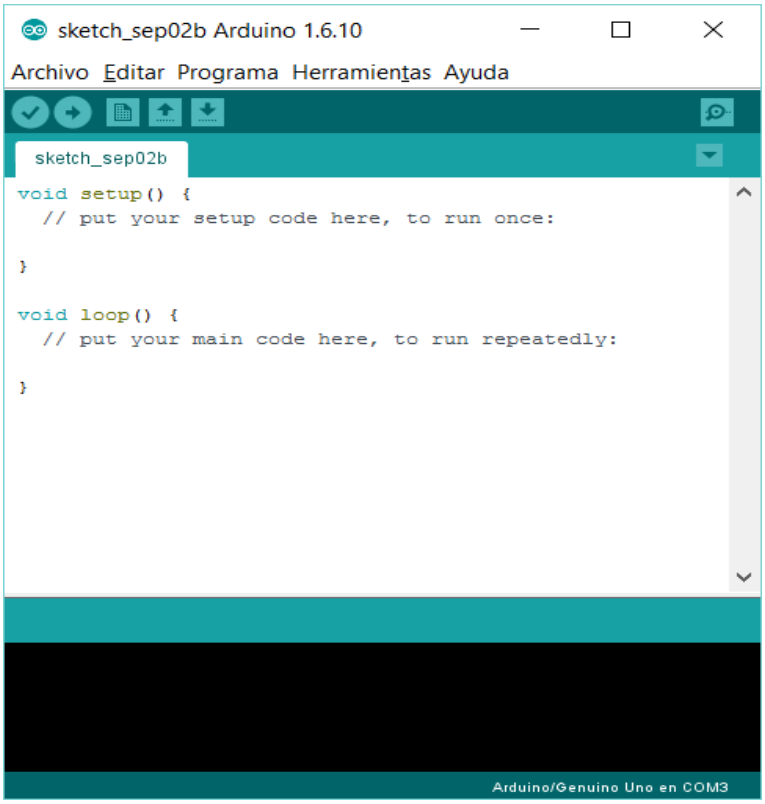


Ilustración 44: código Arduino, sketch

En ella observamos que se muestra por defecto un comando setup y un comando loop. El setup tiene como función inicializar variables, pines, librerías, etc. Es decir, es un apartado de configuración el cual solo se ejecuta una vez en el momento de reinicio o carga del programa a Arduino. Mientras que loop como su nombre indica es usado para controlar mediante instrucciones activamente Arduino.

El código de nuestro sketch los explicaremos en una serie de fases ya expuestas en el apartado de algoritmo. Se recomienda observar el fichero con el código completo y comentado en "código/weatherStation.ino"

1. **Inicialización:** al comenzar el programa se inicializan e importan las librerías, así como la creación de objetos que apuntan a las clases pertinentes de estas.
2. **Toma de datos meteorológicos:** haciendo uso de los objetos de las clases y los métodos explicado con anterioridad se obtiene el valor de los datos meteorológicos.
3. **Comunicación Bluetooth:** transmisión de datos mediante el serial de comunicación Bluetooth.

1.Inicialización: primero se importan las librerías necesarias.

```
// Libreria que incluye los métodos y las clases necesarias para obtener los valores de la estación.
#include <TFG.h>
#include <Wire.h>
#include <SoftwareSerial.h>   // Incluimos la librería  SoftwareSerial
```

Ilustración 45: código Arduino, importación de librerías

A continuación, se crean los objetos que apuntan a las clases de librería.

```
// Creación de objeto Bpm180, el cual apunta a la clase SFE_BPM180. Y Creación de objeto HH10D, el cual apunta a la clase HH10D
SFE_BMP180 Bmp180;
HH10D Hh10d;
SoftwareSerial BT(10,11); // Definimos los pines RX y TX del Arduino conectados al Bluetooth, determinando el serial de comunicación
```

Ilustración 46: código Arduino, creación de objetos

Por último, se inicializan las librerías.

```
void setup()
{
  /*Inizializacion del serial de comunicacion, 9600 bps (velocidad con la que se va comunicar).Se abre
   * un puerto llamado serial entre los modulos y arduino. De esta forma se transmiten los datos de unos a otros.
   * TX-RX
   */
    Serial.begin(9600);
    Serial.println("Serial Iniciado"); // Indicamos en el monitor que la inicialización de la comunicación serial se ha establecido

    // Inicialización del sensor (importante para obtener los valores decalibración del dispositivo)
    // Se comprueba si los datos de calibraciónse se han recuperado correctamente.
      if (Bmp180.begin())
    Serial.println("BMP180 se ha inicializado correctamente");
    else


    // Error,Problabemente sea un erro en la conexión entre el módulo y arduino.
    Serial.println("BMP180 inicialización fallida\n\n");
    while(1); // Pause forever.

    }

    Hh10d.begin();
    BT.begin(9600);       // Inicializamos el puerto serie BT que hemos creado
}
```

Ilustración 47: código Arduino, inicialización de librerias.

2.Toma de datos meteorológicos: debido a que la toma de medida de presión y temperatura tiene un código similar, se expondrá el código de temperatura y de humedad relativa.

```
//Comienzo de medida de temperatura:
// Si la llamada al método startTemperature ha ido correctamente, el número obtenido es en ms.
// Si la llamada ha sido fallida, se obtiene un 0.
status = Bmp180.startTemperature();
if (status != 0)
{
  // Espera el número en ms obtenido anteriormente, y obtiene la medida
  delay(status);

  // Se obtiene el valor de temperatura:
  // La medida es guardada en la variable t antes declarada.
  // La función devuelve 1 si la operación ha ido correctamente, 0 si falla.

  status = Bmp180.getTemperature(t);
  if (status != 0)
  {
    // Muestra el resultado obtenido.
    Serial.print(">Temperatura: ");
    Serial.print(t,2);
    Serial.print(" grados centigrados , ");
    Serial.print((9.0/5.0)*t+32.0,2);
    Serial.println(" grados fahrenheit ");
```

Ilustración 48: código Arduino, obtención de temperatura

```
h=Hh10d.humedadRead();

      if(h!=0)
      {
      Serial.print(">Humedad Relativa: ");
      Serial.print(h);
      Serial.println("%");
```

Ilustración 49: código Arduino, obtención de humedad relativa.

Como se observa hay una condición if. Si el valor obtenido es 0 quiere decir que ha habido algún problema en recibir los datos y se mostrara el siguiente mensaje:

```
    }
    else Serial.println("error toma de medidas de humedad\n");
  }
  else Serial.println("error recibiendo datos de presión\n");
```

Ilustración 50: código Arduino, error al recibir los datos.

3. Comunicación Bluetooth: esta parte del código es la encargada de enviar datos al módulo Bluetooth y este a la aplicación generada por app Inventor 2. La Aplicación cuando quiere recibir un dato meteorológico lo indica mediante un carácter y este al recibir dicho identificativo envía el dato correspondiente por el puerto serie.

```
char dato=BT.read(); //Lee el dato del puerto serie bluetooth antes asignado.
```

Ilustración 51: código Arduino, lectura del puerto serie Bluetooth

```
switch(dato)
 {
   //Envia datos de temperatura.
   case 't':
   BT.print(t);//Envía el dato por el puerto serie
   BT.println("°C");
   break;

   //Envia datos de Humedad
   case 'h':
   BT.print(h);//Envía el dato por el puerto serie
   BT.println("%");
   break;

   //Envía datos de presión atmosférica
   case 'p':
   BT.print(p);//Envía el dato por el puerto serie
   BT.println("mb");
   break;

 }
```

Ilustración 52: código Arduino, envió de datos por el puerto serie Bluetooth

5.3 Código de App Inventor 2

Código basado en bloques que tiene como objetivo crear una aplicación para Android donde se pueda controlar la comunicación Bluetooth y visualizar los parámetros meteorológicos. El código basado en bloques está separado en dos partes, la pantalla de inicialización y la pantalla principal.

Para inicializar un proyecto en App Inventor 2 nos dirigimos a la pestaña *projects* dentro de esta clicamos en *start new Project* y nos genera una interfaz que está compuesta por una parte gráfica y una de bloques. Ambas partes están relacionadas ya que en la parte gráfica representamos que objetos queremos que nuestra aplicación contenga y en la parte de bloques de programación damos una funcionalidad a estos objetos. Las siguientes ilustraciones muestran el código utilizado para la pantalla de inicialización y la pantalla principal.

Debido a que mostrar todos los bloques puede ser muy extenso se pondrán los restantes en un anexo “Ficheros anexos/códigoAppInventor2”

Pantalla de inicialización: Como su nombre indica es la pantalla de inicialización de la aplicación. En esta se muestra durante un período el título de la aplicación el emblema de la universidad y el logo de Arduino.

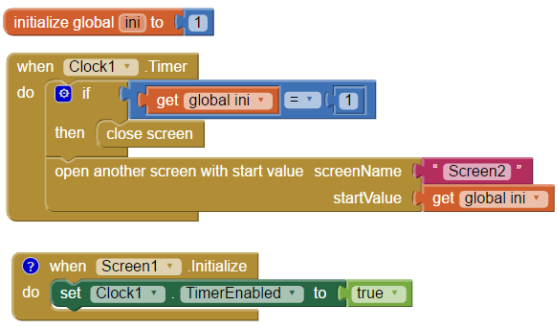


Ilustración 53: Código App Inventor 2, Pantalla inicialización.

El código básicamente consiste en cambiar a la pantalla principal cuando el intervalo de tiempo del reloj se cumpla.

Pantalla Principal: Encargada de la conexión Bluetooth y recepción de los datos meteorológicos. Primero se **declaran** variables y se i**nicializa** la pantalla.

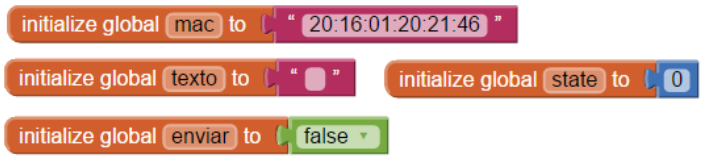


Ilustración 54: Código App Inventor 2, Pantalla principal (variables)

Estas variables son utilizadas para el control y recepción de datos.

- Variable **mac**: Es la dirección mac del Bluetooth la cual permitirá a posteriori la conexión de la aplicación con el dispositivo.
- Variable **texto:** Es la encargada de almacenar el contenido que recibe de Arduino en este caso los datos meteorológicos.
- Variable **enviar:** Variable de control que servirá para indicar cuando se envían datos a Arduino y cuando no.
- Variable **state:** Variable necesaria para el control simultaneo de datos de Arduino. En las siguientes líneas se observará su funcionamiento.

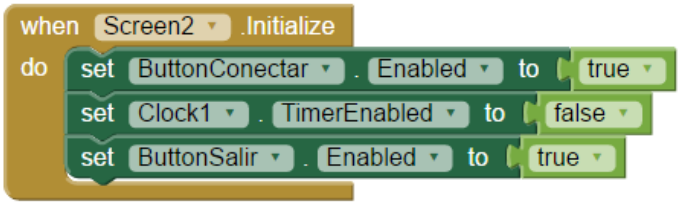


Ilustración 55: Código App Inventor 2, Pantalla principal (inicialización de pantalla)

La pantalla principal es iniciada, tanto el botón conectar como salir están habilitados de ser usados, mientras que el reloj encargado del control de la recepción de datos esta deshabilitado.

Una vez se han declarado e inicializado las variables, se procede a la **conexión Bluetooth**. Los bloques correspondientes al botón desconectar se muestran en el anexo.

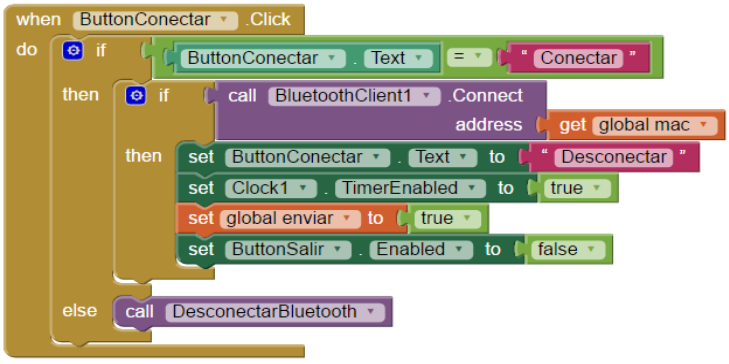


Ilustración 56: Código App Inventor 2, Pantalla principal (botón conectar)

Tras pulsar el botón conectar, la aplicación se conecta al módulo bluetooth mediante la variable mac antes declarada y se inicializa el reloj iniciando la recepción de datos.

A continuación, se muestra el bloque principal de la aplicación encargado de la lectura de datos.

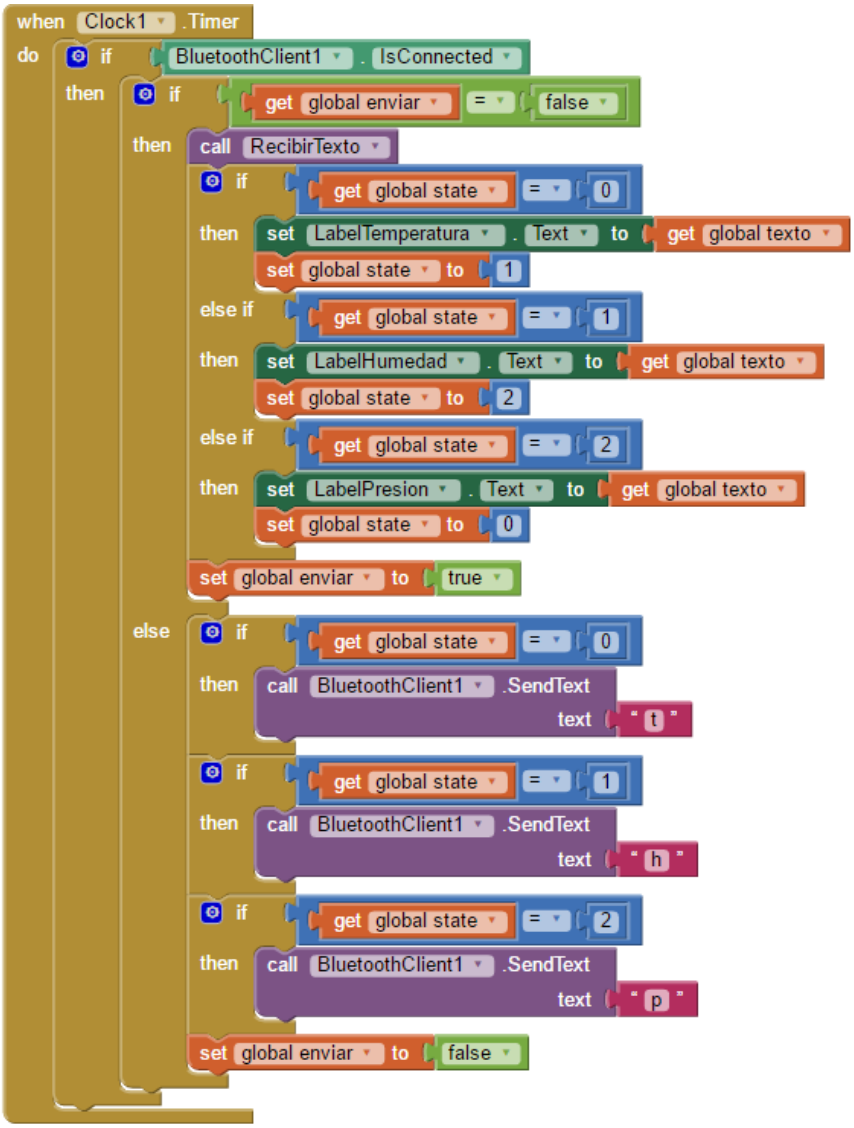


Ilustración 57: Código App Inventor 2, Pantalla principal (lectura de datos)

Este bloque se divide en dos funciones: recibir los datos metereologicos y enviar el carácter identificativo a Arduino para la recepción de los mismos. En primera instancia se comprueba si el Bluetooth está conectado, una vez se demuestra que este está conectado, se comprueba el estado de la variable enviar, debido a que la variable enviar es verdadera por defecto, se envía los caracteres identificativos a Arduino. En primera instancia se envía el carácter “t” debido a que la variable state fue definida con un 0. Una vez el carácter ha sido enviado el estado de enviar pasa a ser falso y en este punto el sistema es cuando recibe y guarda el texto de Arduino (función explicada en el anexo) y muestra el dato de temperatura. Al mostrar el dato de temperatura el estado de la variable state cambia a 1, pasando a obtener el valor de humedad y así continuamente.

6. RESULTADOS

En este apartado se muestra el resultado final del proyecto, separado principalmente en dos partes:

1. Comprobación de funcionamiento del sistema por el monitor serie de Arduino.
2. Resultados finales de la aplicación *Weather Station* generada por App Inventor 2.

En el **primer punto** se comprueba el funcionamiento de los sensores mediante mensajes de notificación en el monitor serie. Antes de mostrar los resultados obtenidos, hay que seguir una serie de pasos para poder visualizarlos, los pasos son los siguientes:

1. Conectar el pueto usb *COM 3 (Arduino/Genuino UNO*) de la placa Arduino al ordenador.
2. Comprobar que la IDE de Arduino reconoce dicho puerto, para ello nos dirigimos a la pestaña /herramientas/PuertoSerie y comprobamos que el puerto se corresponde con el nombre antes indicado. También comprobamos si la placa es la correcta para ello seleccionamos /herramientas/ placa y comprobamos si se corresponde al nombre “Arduino/Genuino UNO”.
3. Por último, pulsamos el botón del monitor serie o mediante los comandos Ctrl+Máyus+M y comprobamos si los resultados son correctos.

A continuación, se muestra los resultados obtenidos en el monitor serie y se compara con datos meteorológicos de la zona mediante la aplicación del tiempo de Windows 10.
Datos obtenidos y mostrados en el **monitor serie**:

```
>Temperatura: 30.44 grados centigrados , 86.80 grados fahrenheit
>Presion absoluta: 1016.33 mb, 30.02 inHg
>Humedad Relativa: 60.09%

>Temperatura: 30.46 grados centigrados , 86.83 grados fahrenheit
>Presion absoluta: 1016.37 mb, 30.02 inHg
>Humedad Relativa: 60.48%

>Temperatura: 30.47 grados centigrados , 86.84 grados fahrenheit
>Presion absoluta: 1016.42 mb, 30.02 inHg
>Humedad Relativa: 60.19%

>Temperatura: 30.47 grados centigrados , 86.84 grados fahrenheit
>Presion absoluta: 1016.44 mb, 30.02 inHg
>Humedad Relativa: 59.32%
```

Ilustración 58: Resultados, datos meteorológicos mostrados en el monitor serie

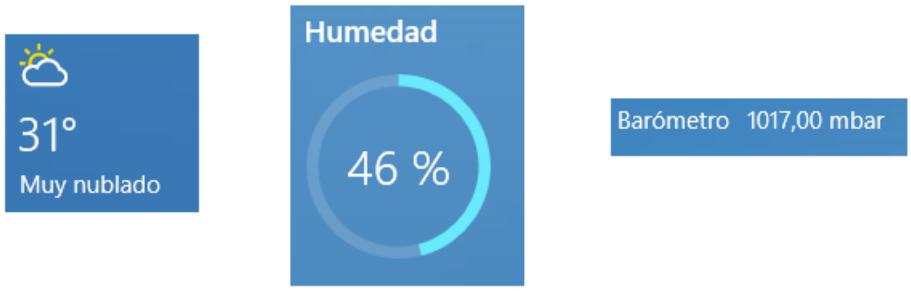


Ilustración 59: Resultados, datos meteorológicos de la aplicación de tiempo de windows 10

El resultado de humedad difiere bastante debido a que las medidas son tomadas desde mi casa que esta junto a la playa.

En el **segundo punto** se muestra la aplicación y los resultados provenientes de Arduino mediante el modulo Bluetooth. Antes de mostrar el resultado se indica las diferentes formas de conectar App Inventor 2 con Android.

App inventor 2 permite que durante la modificación del código y diseño de la aplicación se pueda observar los resultados simultáneamente en nuestro Smartphone. A continuación, se muestra las tres opciones que App Inventor 2 ofrece para realizar esta acción:

1. **AI Companion:** es necesario bajarse la aplicación disponible en app store *MIT AI2 Companion* [26]. Y mediante esta conectarse a la web haciendo uso del wifi. Esta es la opción elegida para el proyecto.
2. **Emulador:** emula un dispositivo Android en nuestro ordenador.
3. **USB:** mediante la aplicación de Windows 10 *aiStarter* [27] y poniendo el USB en modo depurar se puede conectar el dispositivo Andorid a la web.

Una vez el proyecto está finalizado se puede exportar como fichero de tipo. apk e instalar en el dispositivo Android.

A continuación, se muestra la aplicación y los resultados obtenidos en el mismo instante que los resultados anteriormente mostrados.

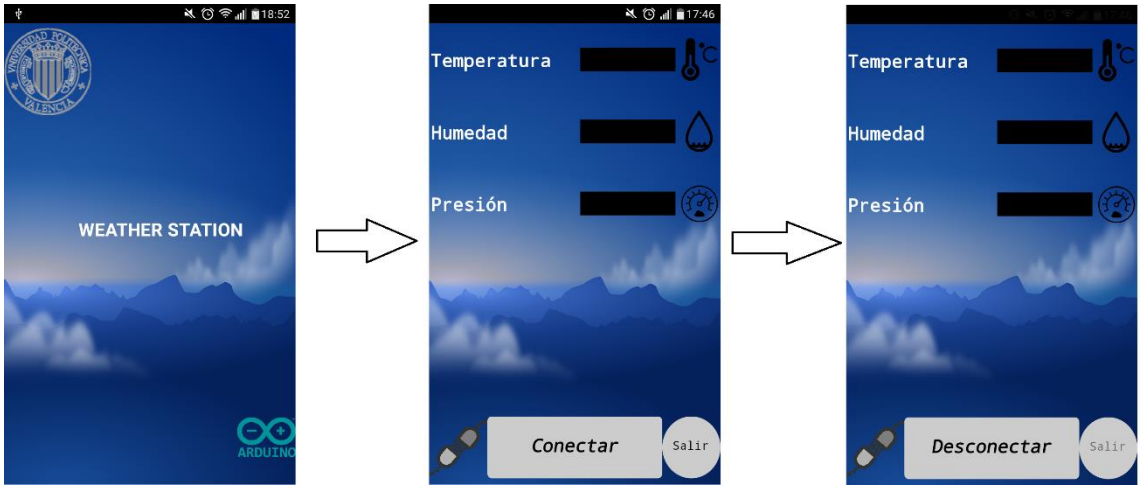


Ilustración 60 :Resultados, inicio de aplicación y conexión al Bluetooth

En estas imágenes se muestra la pantalla de inicialización que pasada 5 segundos pasa a la pantalla principal, en esta se pulsa el botón conectar y se establece la conexión Bluetooth.

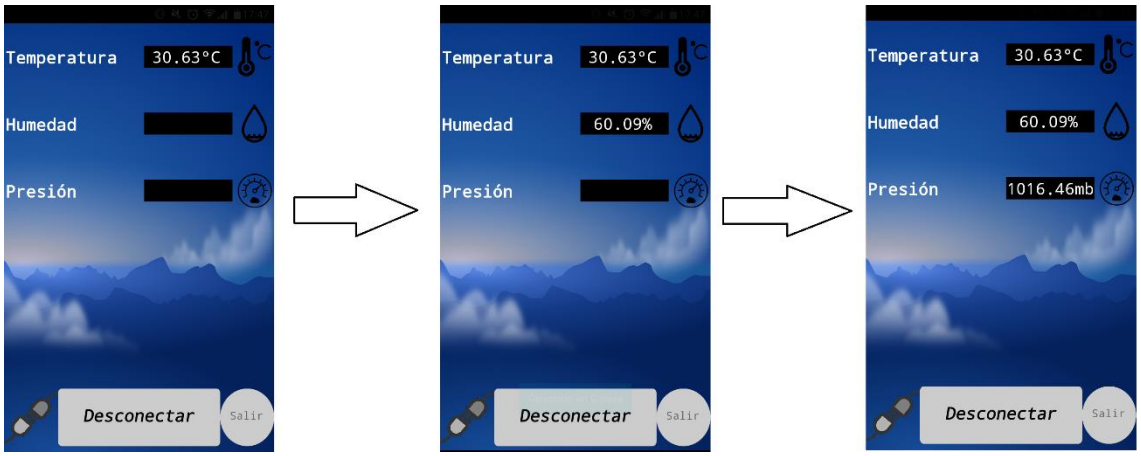


Ilustración 61: Resultados, datos meteorológicos mostrados en la aplicación.

Tras la espera de unos segundos se recibe el primer dato meteorológico, la temperatura. Tras recibir el primero consecutivamente se reciben el dato de humedad y presión. Los datos se actualizan constantemente cada 5 segundos.

7. CONCLUSIÓN

En líneas generales los objetivos del proyecto han sido cumplidos, se ha conseguido que nuestra aplicación reciba datos captados por Arduino y se muestren en un dispositivo móvil Android.

Este proyecto se cogió en buena parte por aprender nuevos lenguajes de programación y reforzar mi campo en este tipo de microcontroladores. Se puede decir que el objetivo se ha cumplido, he aprendido a programar desde cero el lenguaje Arduino y C++ apoyándome en un proyecto real. Todo esto me ayudara en un futuro a realizar este tipo de proyectos con una cierta base y experiencia.

A pesar de haber sufrido diversos contratiempos a causa de la falta de conocimiento y errores de código he de concluir que ha sido un trabajo que ha mejorado mi metodología a la hora de abarcar este tipo de proyectos.

En conclusión, la realización del Trabajo de final de grado ha sido una experiencia positiva que me ha permitido aprender y mejorar como profesional.

Bibliografía

[1] "Arduino - Home", *Arduino.cc*. [Online]. https://www.arduino.cc/.

[2] "C++ Tutorial | SoloLearn", *Sololearn.com*. [Online]. http://www.sololearn.com/Course/CPlusPlus/.

[3] L. Llamas, "Tutoriales de Arduino", *Luis Llamas*. [Online]. http://www.luisllamas.es/tutoriales-de-arduino/.

[4] S. Tushev, "Interfacing HH10D humidity sensor with Arduino - Simon Tushev Website", *Tushev.org*. [Online]. https://tushev.org/articles/arduino/6/interfacing-hh10d-with-arduino.

[5] "BMP180 Barometric Pressure Sensor Hookup - learn.sparkfun.com", *Learn.sparkfun.com*. [Online]. https://learn.sparkfun.com/tutorials/bmp180-barometric-pressure-sensor-hookup-.

[6] "Módulo BlueTooth HC-06 | Tutoriales Arduino", *Prometec.net*, 2016. [Online]. http://www.prometec.net/bt-hc06/.

[7] "Configuración del módulo bluetooth HC-06 usando comandos AT", *Naylampmechatronics.com*. [Online]. http://www.naylampmechatronics.com/blog/15_Configuraci%C3%B3n--del-m%C3%B3dulo-bluetooth-HC-06-usa.html.

[8] "Mit App Inventor", *Ai2.appinventor.mit.edu*. [Online]. http://ai2.appinventor.mit.edu.

[9] "Arduino Frequency Counter Library", *Interface.khm.de*. [Online]. http://interface.khm.de/index.php/lab/interfaces-advanced/arduino-frequency-counter-library/.

[10] "C Standard Library Reference Tutorial", *www.tutorialspoint.com*. [Online]. http://www.tutorialspoint.com/c_standard_library/.

[11] "Navegando entre pantallas con App Inventor - Aprender a programar apps", *Aprender a programar apps*, 2013. [Online]. http://aprenderaprogramarapps.es/2013/05/29/navegando-entre-pantallas-con-app-inventor/.

[12] "Enviar y recibir datos con módulo Bluetooth para Arduino", *Zygzax | Proyectos y tutoriales, electrónica y diseño 3D*, 2013. [Online]. http://zygzax.com/enviar-y-recibir-datos-con-modulo-bluetooth-para-arduino/.

[13] "Proyecto Arduino-Android: Temperatura y Humedad (app Inventor 2)", *Arduinoamuete.blogspot.com.es*. [Online]. http://arduinoamuete.blogspot.com.es/2014/04/proyecto-arduino-android-temperaruta-y.html.

[14] "Basic Bluetooth communications using App Inventor | App Inventor 2 – Learn to Code!",*Appinventor.pevest.com*. [Online]. http://appinventor.pevest.com/?p=520.

[15] "Historia de Arduino y su nacimiento", *BotScience - imagina, diseña, contruye...*, 2012. [Online]. https://botscience.wordpress.com/2012/06/05/historia-de-arduino-y-su-nacimiento/.

[16] "Notepad++ Home", *Notepad-plus-plus.org*. [Online]. https://notepad-plus-plus.org/.

[17] "El bus I2C | Tutoriales Arduino", *Prometec.net*. [Online]. http://www.prometec.net/bus-i2c/.

[18] "Fritzing", *Fritzing.org*. [Online]. http://fritzing.org/download/?donation=0.

[19] "Comunicación I2C y SPI", *Es.slideshare.net*. [Online]. http://es.slideshare.net/JonathanRuizdeGaribay/09bcomunicacin-i2-cyspi-9769471.

[20] "yEd Graph Editor", *yWorks, the diagramming company*. [Online]. https://www.yworks.com/products/yed.

[21] "Diseño de algoritmos en la programación de computadoras - Monografias.com", *Monografias.com*. [Online]. http://www.monografias.com/trabajos94/diseno-algoritmos-programacion-computadoras/diseno-algoritmos-programacion-computadoras.shtml.

[22] "Archivos de cabecera en C++", *Computación Gráfica FI*. [Online]. https://computaciongrafica.wordpress.com/2008/03/29/archivos-de-cabecera-en-c/.

[23] "Diagrama de Flujo ¿Què es y Como se Hace?", *Taringa.net*. [Online]. http://www.taringa.net/post/info/5099216/Diagrama-de-Flujo-Qu-es-y-Como-se-Hace.html.

[24] "Diagrama de flujo, símbolos", *2.bp.blogspot.com*. [Online]. http://2.bp.blogspot.com/-hdYnJEjl8EE/Vm3JSNKaVCI/AAAAAAAADwY/soxUsZRU2HQ/s1600/Diagrama%2Bde%2Bflujos.jpg.

[25] "Diagramas de clases de UML: Referencia", *Msdn.microsoft.com*. [Online]. https://msdn.microsoft.com/es-es/library/dd409437.aspx.

[26] "MIT AI2 Companion", *Play.google.com*. [Online]. https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3&hl=es.

[27] "Ai2 Starter 2.6 Download", *Software Informer*. [Online]. http://ai2-starter.software.informer.com/2.6/.

[28] *BMP 180 datasheet*, 2nd ed. 2013.

[29] *HH10D datasheet*, 2nd ed. 2010.