Escuela Politecnica Superior de Jaen

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

MONITORIZACIÓN DE UNA ESTACIÓN METEOROLÓGICA

Alumno: Ana Villena Alguacil

Tutor: Prof. D. Elisabet Estévez Estévez
Dpto: Ingeniería Electrónica y Automática

Febrero, 2017

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Doña Elisabet Estévez Estévez , tutor del Proyecto Fin de Carrera titulado: Monitorización de una estación meteorológica, que presenta Ana Villena Alguacil, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Febrero de 2017

El alumno:

Ana Villena Alguacil

La tutora:

Elisabet Estévez Estévez

Agradecimientos

A mis padres,
sin su apoyo y paciencia
llegar hasta aquí habría sido imposible.
A todos mis amigos y compañeros.
A Elisabez Estévez, tutora de este trabajo,
por su tiempo, colaboración
y por ser un ejemplo de dedicación y eficacia
a lo largo de estos años.
A Mª José Moreno,
por ayudarme a elegir el camino que quería,
y no el que los demás esperaban que tomase.

Tabla de contenido

Tabla de ilustraciones

Tabla de diagramas

Listado de tablas

1. MOTIVACIÓN

El objetivo de este Trabajo Final de Grado es dar un paso más después de los cuatro años de grado, no sólo buscando una aplicación a todo lo aprendido; sino dando continuidad al estudio para conocer qué tecnologías están en desarrollo y así buscar aplicaciones que supongan una mejora en este campo y en la vida cotidiana.

Por este motivo, la decisión final fue centrar el trabajo en *El Internet de las Cosas*. Este es un concepto ambiguo, que está en auge y hace referencia al uso de diferentes tecnologías de recopilación de datos con el objetivo que lograr la comunicación tanto entre diversos objetos cotidianos como entre estos y sus usuarios.

Es precisamente esta comunicación lo que hace interesante esta estación meteorológica ya que permite conocer los valores de las magnitudes que son objeto de medida, así como configurar una serie de parámetros, sin necesidad de encontrarse en el mismo punto que el dispositivo.

Este trabajo es una primera maqueta que ha supuesto una oportudidad para conocer un campo que está creciendo rápidamente dentro del mundo de la tecnología e investigar cuál es su alcance tanto actual como futuro. Como proyecto, supone una pequeña base para futuros proyectos más ambiciosos que aprovechen las innumerables soluciones ofrecidas por *The Internet of Things* o *IoT,* como se hará referencia a este concepto en el desarrollo de este documento.

2. OBJETIVOS

El objetivo fundamental de este proyecto es la monitorización en tiempo real, utilizando un servicio de nube y una aplicación de *Android,* de las magnitudes atmosféricas medidas por una estación meteorológica basada en *Arduino.*

Para alcanzar este objetivo es necesario cumplir con una serie de objetivos secundarios:

- Diseño hardware del dispositivo. Este incluye la recopilación de las variables atmosféricas que van a ser objeto de medida, la selección de sus componentes, entre los que se encuentran los sensores y el microcontrolador; y el posterior montaje de estos elementos.
- Diseño del software que gestionará el funcionamiento del microcontrolador. Este algoritmo definirá las funciones necesarias para la adquisición de los datos de los sensores y su posterior procesamiento, la creación de una comunicación del microcontrolador con el servicio *Cloud* y la aplicación de *Android.*

  Además, este código debe incluir la gestión de un sistema de alarma que proporcione al dispositivo seguridad frente a errores de funcionamiento o eventos no deseados.
- Comparación de las prestaciones de todos los servicios de nube ofertados en *Internet* para seleccionar el más adecuado, aquel que permita comunicarse con el microcontrolador elegido, visualizar los datos en tiempo real y almacenar dicha información.
- Diseño de una aplicación de *Android* gracias a la cual sea posible establecer una comunicación bidireccional con el microcontrolador, es decir, que permita recibir datos y mostrarlos en un dispositivo móvil y también, enviar información al microcontrolador para la configuración de una serie de preferencias.

Además, como función adicional, el último de los objetivos marcados para este proyecto es el diseño de un controlador para una de las variables más significativas. Dicho controlador contará con dos versiones, On-Off y PID.

3. ESCENARIO GENERAL

Esta maqueta puede dividirse en tres unidades: el microcontrolador, el servicio de nube y la aplicación de Android. Cada una de ellas lleva a cabo una serie de tareas necesarias para el funcionamiento conjunto de la estación meteorológica.

En primer lugar, se encuentra el microcontrolador donde se procesan las magnitudes analógicas enviadas desde los sensores. Utilizando estos datos se gestiona un sistema de alarmas y un control ON-OFF de una de las variables. Desde aquí también se gestiona la comunicación con las otras dos unicades, Bluetooth con Android y WiFi con la nube.

A continuación, aparece el servicio de nube que,una vez configurada, permitirá la monitorización en tiempo real y almacenamiento de las variables medidas y el estado de las alarmas.

Por último, la aplicación Android ofrece la posibilidad de configurar una serie de parámetros necesarios para el funcionamiento del sistema además de poder visualizar las variables leídas por el microcontrolador.

Figura 1. Escenario general

4. DISEÑO HARDWARE

Este apartado está dedicado a la descripción del diseño del dispositivo físico, es decir, a la unidad fundamental de la estación. Este está formado por un microcontrolador y una serie de sensores y dispositivos secundarios tal y como puede observarse en Figura 2

El diseño inicial se realizó utilizando una protoboard, Figura 3, y sobre éste se fueron añadiendo y modificando elementos hasta alcanzar los objetivos marcados. Una vez alcanzado este punto se llevó a cabo el diseño y fabricación de la PCB.

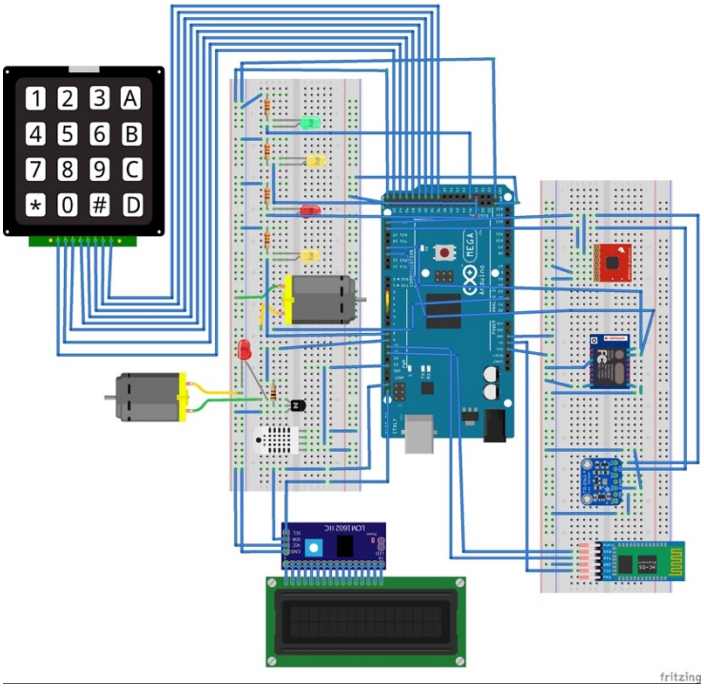

Figura 2. Esquema del diseño

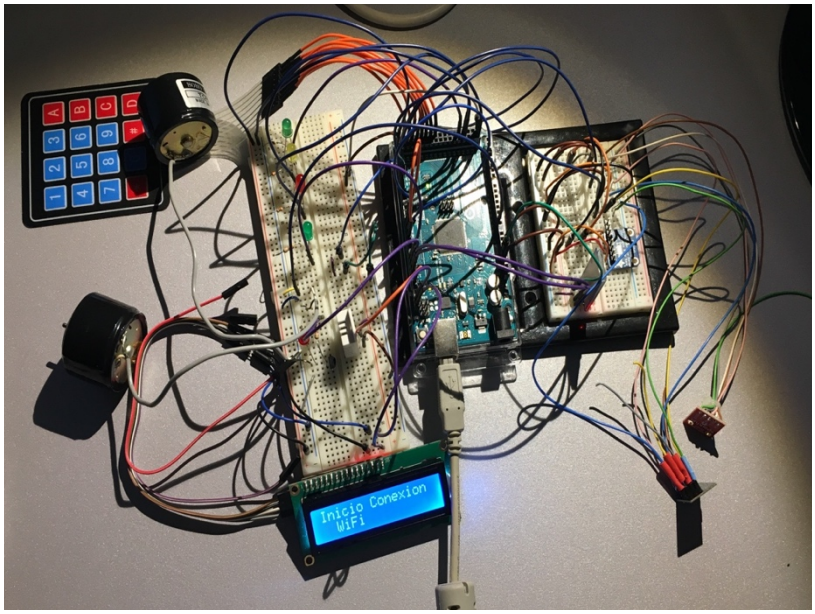

Figura 3. Diseño inicial

Para la fabricación de la PCB se llevaron a cabo una serie de instrucciones que se detallan a continuación:

1. Sobre una placa virgen de cobre dibujar el esquema del dibujo utilizando un rotulador permanente Figura 4

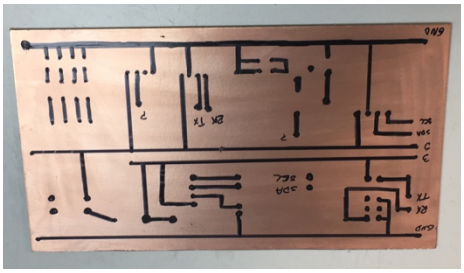

Figura 4. Dibujo de las pistas

2. Preparar una mezcla sobre la que se sumergirá dicha placa, Figura 5, con el objetivo de eliminar todo el cobre excepto el que se encuentra bajo las pistas dibujadas con rotulador. Esta mezcla estará formada por 125 ml de ácido clorhídroco y 250 ml agua oxigenada.

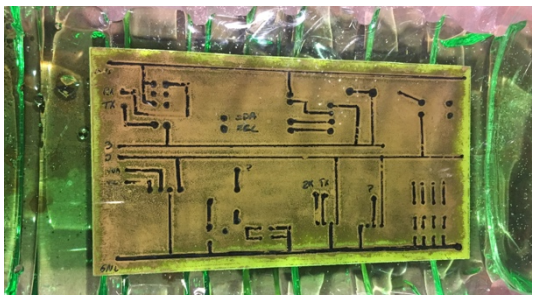

Figura 5. Placa sumergida en la mezcla

3. Una vez que el cobre ha desaparecido enjuagar la placa con agua para detener la acción del ácido. Tras esto, utilizar hay que utilizar acetona para eliminar las marcas de rotulador. En Figura 6 puede verse el resultado de este proceso

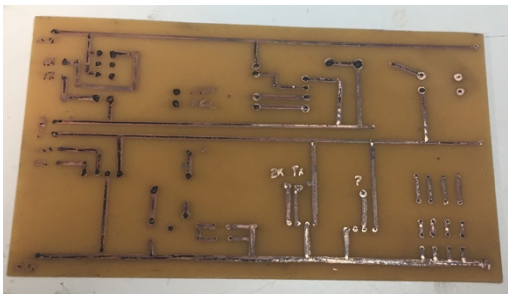

Figura 6. Pistas de cobre

4. Finalmente, para colocar los componentes, será necesario taladrar unos orificios donde colocarlos y fijarlos a estos utilizando un soldador e hilo de estaño como puede verse en la Figura 7.

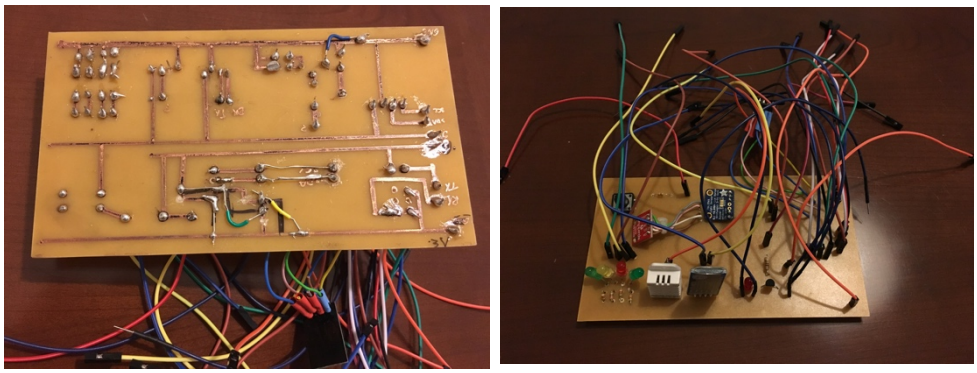

Figura 7. Placa con componentes soldados

Por último se colocan todos los componentes que forman la estación para obtener la maqueta final, mostrada en la Figura 8.

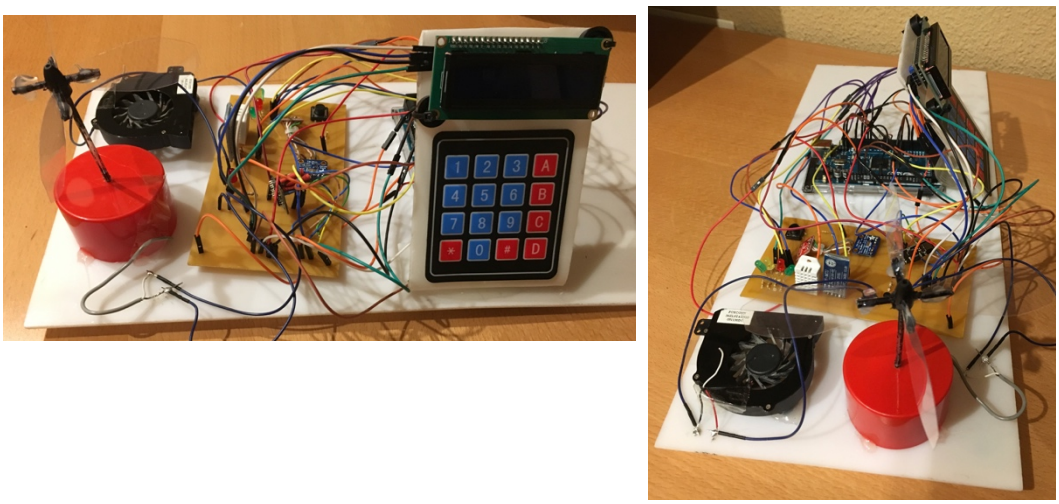

Figura 8. Maqueta final

4.1. Placa Arduino

El microcontrolador elegido es Arduino [1] ya que es una compañía de hardware libre, que es programado utilizando un entorno de desarrollo (IDE) gratuito, para la que es posible encontrar una amplia gama de productos de bajo coste y prestaciones suficientes para un proyecto de estas características.

A pesar de que existe una amplia variedad de placas que podrían haberse utilizado se realizó una comparación entre las características de dos de ellas *Uno* y *Mega*, ver en Tabla 1.

Tabla 1. Características de las placas Arduino

Característica	Uno	Mega
Microcontrolador	ATmega328P	ATmega2560
Pins I/O digitales	14	54
Pins entradas analógicas	6	16
Puertos Serial	1	4
Flash Memory	32 KB	256 KB
SRAM	2 KB	8 KB
EEPROM	1 KB	4 KB
Clock Speed	16 MHz	16 MHz
Interrupciones	2	6
Dimensiones(mm)	68.6 x 53.4	101.52 x 53.3

En general, es evidente que las prestaciones de la placa *Mega* son superiores a las de *Uno,* aunque las de esta última son suficientes para alcanzar los objetivos del proyecto. Sin embargo, en algunas características concretas, *Uno* no cumple con los requisitos mínimos necesarios.

En primer lugar, *Mega* cuenta con un mayor número de pines de entradas y salidas digitales, que serán necesarias para la conexión de los sensores y dispositivos auxiliares, así como para el desarrollo de funciones del programa. Además, permite utilizar hasta cuatro Puertos Serial con los que se llevan a cabo las comunicaciones inalámbricas con las otras dos estaciones del proyecto, además de con un PC. Por último, la placa *Uno* no ofrece interrupciones por hardware suficientes para llevar a cabo la gestión de paro y el funcionamiento de algunos dispositivos auxiliares.

Por lo tanto, tras definir unos objetivos y comparar ambas placa, la elegida es la *Mega.*

4.2. Sensores y dispositivos auxiliares

Además de por el microcontrolador el dispositivo está formado por una serie de sensores, que proporcionan los datos de cada una de las variables atmosféricas elegidas; y unos dispositivos auxiliares gracias a los cuales es posible la comunicación con el microcontrolador y la visualización de las variables medidas sin necesidad de recurrir a un PC o dispositivo móvil.

4.2.1. TSL2561

Este es el sensor utilizado para medir la luminosidad convirtiendo la intensidad de la luz en una seña digital. Su característica más destacada es que combina en un circuito integrado CMOS un fotodiodo de banda ancha, que permite medir el espectro visible y el infrarrojo, y un fotodiodo infrarrojo; lo que le permite proporcionar una respuesta precisa con una resolución de 16 bits. Cuenta con siete pines de conexión, tal y como puede observarse en Figura 9, de los cuales únicamente se utilizarán cuatro para su alimentación y comunicación con el microcontrolador. Para alimentarlo, se utilizará la salida de 3.3 V de la placa Arduino. En cuanto a la comunicación con el microcontrolador, se lleva a cabo utilizando una de las tres direcciones disponibles, 0x39 0x29 ó 0x49, del bus de datos I2C (*Inter-Integrated Circuit*), es decir, los pines SDA (*Serial Data*) y SCL (*Serial Clock*). Gracias a una librería de Arduino específica para el sensor es posible, además de configurar parámetros como la dirección y la frecuencia de muestreo, obtener la lectura directamente en lux siempre que esta esté dentro del rango permitido entre 0.1 y 40000 lux.

Figura 9. Sensor TSL2561

4.2.2. DHT22

Este sensor, Figura 10, permite medir la humedad relativa y temperatura en grados Centígrados o Fahrenheit gracias a un sensor de humedad capacitivo y un termistor. Para alimentarlo se utiliza la salida de 5 V del Arduino mientras que para su comunicación, la salida digital. Este sensor permite medir la humedad dentro de un rango entre 0 y 100 % con una precisión de ±2 %, mientras que el rango de la temperatura está entre -40 y 80 ºC con una precisión de ±0.5 º; con una frecuencia de lectura de dos veces por segundo, Como sucedía en el caso anterior, el sensor cuenta con una librería específica que permite la configuración de parámetros y la obtención de las variables medidas.

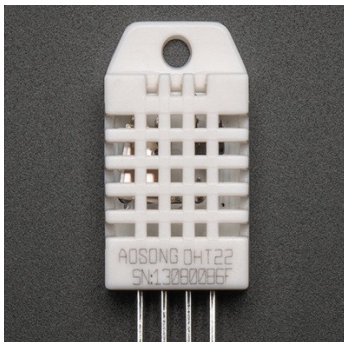

Figura 10. Sensor DHT22

4.2.3. BMP 180

Este sensor,mostrado en Figura 11 y basado en tecnología piezoresistiva, permite medir presión atmosférica y altitud respecto al nivel del mar, aunque en este caso se utilizará únicamente para la primera lectura. Además de por el sensor piezoresistivo, está formado por un convertidor analógico-digital y una unidad de control con memoria EEPROM, donde están almacenados 176 bit de datos calibrados, para compensar el valor del offset. Al igual que el TSL256, irá conectado a los pines *SDA y SCL* ya que para la comunicación se utiliza la dirección *0x77* del bus I2C. Dos de los tres pines restantes se utilizarán para la alimentación del dispositivo, para la que se requieren 3.3V. En ese caso su rango de lectura está entre 300 y 1100 hPa y tiene una precisión de 0.02 hPa.

Figura 11. Sensor BMP180

4.2.4. Sensor de velocidad del viento

En este caso no se ha utilizado un anemómetro comercial sino que he diseñado uno utilizando un motor DC y anemómetro casero, Figura 12. El motor va conectado a un pin analógico y cuando se mueve el eje de este, generando voltaje, esta entrada analógica tomará un valor entre 0 y 1023.

Para construir el anemómetro se han utilizado tubos de plástico de pequeño diámetro. El de mayor longitud va conectado al eje del motor y será su movimiento el que genere voltaje. Sobre un soporte colocado al final de dicho tubo se taladrán otros dos tubos del mismo diámetro pero de menor longitud. En cada uno de los cuatro extremos de estos tubos se colocarán las aspas que serán movidas por el viento. Por último, en el interior de un cilindro, se coloca el motor y con ayuda de un taladro se hacen dos orificios: uno para el eje principal y otro para la salida del cableado.

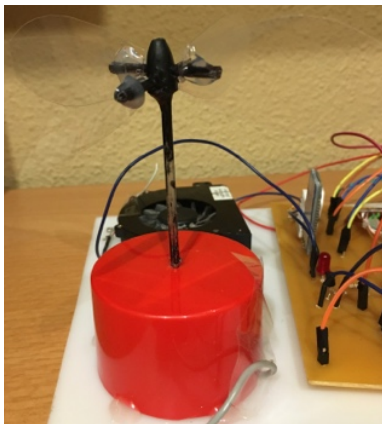

Figura 12. Anemómetro

4.2.5. Teclado matricial 4x4

Es un conjunto de botones con símbolos alfanuméricos conectados en filas y columnas, Figura 13. De esta forma se requiere un menor número de pines digitales para su lectura, ya que sólo es necesario uno para cada una de las filas y columnas; en lugar de uno para cada botón.

Figura 13. Teclado matricial

4.2.6. Pantalla LCD 16x2

Este display está formado por dos filas con 16 caracteres cada una de ellas y en él se mostrarán los mensajes enviados desde el microcontrolador. Los displays convencionales necesitan doce pines digitales para su funcionamiento pero en ese caso, gracias al módulo LCM1602, Figura 14, es posible su conexión únicamente con cuatro pines. Dos de ellos se conectarán a los pins *SDA y* SCL,utilizando la dirección 0x3F del bus I2C para la comunicación; y los dos restantes irán conectados a la salida de 5V del Arduino y a masa. Este módulo además cuenta con un potenciómetro que permite variar el nivel de contraste del display.

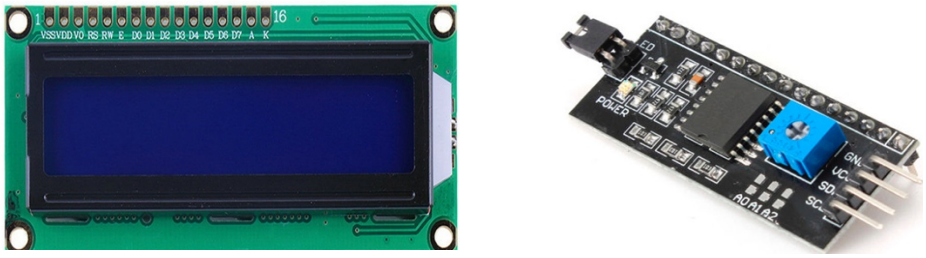

Figura 14. Pantalla LCD y módulo LCM1602

4.2.7. Diodos LEDs

El dispositivo cuenta con cinco diodos LEDs, conectados a distintos pines digitales, que funcionan como avisos luminosos para el sistema de alarma, el control ON-OFF y la comunicación con la app de Android.

6.2 Módulos de comunicación

Para alcanzar los objetivos marcados es necesario establecer dos comunicaciones inalámbricas. La primera, vía WiFi, comunica la unidad principal y la nube; mientras que la segunda comunica la unidad y la aplicación mediante Bluetooth. Los módulos utilizados son los siguientes:

4.3.1. HC-06. Módulo Bluetooth

Permite la creación de un enlace Bluetooth con el microcontrolador utilizando uno de sus Puertos Serie. Aunque podemos encontrar el módulo HC-05 con prestaciones superiores, puede ser configurado en modo maestro o modo esclavo, no ha sido utilizado ya que en este caso el dispositivo no tiene por qué iniciar la comunicación por lo que es suficiente con el funcionamiento en modo esclavo. Su conexión se realiza mediante cuatro pines, Figura 15, dos de ellos para la alimentación y la conexión a tierra; y los dos restantes para la conexión con el Puerto Serie. El módulo funciona con un voltaje entre 3.6 V y 6 V por lo que se conectará a la salida de 5 V del microcontrolador mientras que los pines TXD y RXD se conectarán de forma cruzada con los del Arduino, es decir, TXD-RXD y RXD-TXD,

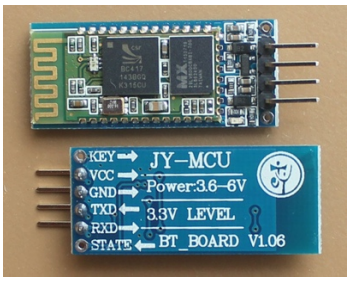

Figura 15. Módulo HC-06

Como paso previo a su instalación en la estación es necesario configurar una serie de valores: nombre, clave para establecer el enlace y velocidad, en bits por segundo, que debe coincidir con la velocidad del Puerto Serial. Esta configuración se

lleva a cabo conectando únicamente el módulo al microcontrolador y enviándole, desde el *IDE Arduino [1]*, los comandos AT que aparecen en Tabla 2.

Tabla 2. Comandos AT de configuración del módulo HC-06.

Comando	Función	Valor
AT	Verificar conexión	-
AT + RESET	Resetear el módulo	-
AT+NAME=*Añadir nombre*	6	ESTACION
AT+PSWD= *Añadir clave*	1	1234
AT+UART= *Valor entre 0 y 9*	Configurar una velocidad entre 1200 y 230400 bps	8 (=115200bps)

4.3.2. ESP8266

Para la comunicación WiFi existían varias opciones en el mercado. En primer lugar, la *Yùn Shield de Arduino,* que es una placa adicional que permite que el Arduino se conecte a Internet vía Wifi o Ethernet. Sin embago, existen módulos más asequibles que también cumplen con las necesidades del proyecto. La segunda opción, tambien de la plataforma Arduino, es la placa *MKR1000* que permite la conexión WiFi sin necesidad de añadir una placa adicional aunque, al igual que sudecía con la placa *UNO*, algunas de sus prestaciones no alcanzan los requisitos mínimos. Por lo que finalmente, el módulo elegido para establecer esta comunicación es el *ESP8266.*

Este, es un módulo de transmisión UART-WiFI que puede tanto actuar como *host* como descargar las funciones WiFi de una red cercana. Además, ofrece la posibilidad de ser integrado con sensores y otra serie de dispositivos específicos gracias a sus pines GPIOs y también, establecer una conexión con un micorocontrolador.

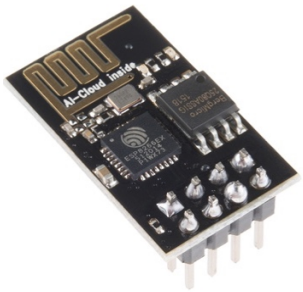

Figura 16. Módulo ESP8266

En este caso concreto, actuará como un adaptador WiFi que permita a la placa Arduino conectarse a un punto de acceso cercano. Para la comunicación entre la placa y el módulo se utilizará una conexión UART, mediante uno de los Puerto Serial.

Para su conexión con el microcontrolador serán necesarios cinco de los ocho pins mostrados en Figura 16, ya que los GPIOs no se utilizarán. Al igual que sucedía con el módulo Bluetooth, los pins TXD y RXD se conectarán de forma cruzada con los pins de Arduino; los pins VCC y CU_PD irán conectados a la salida de 3.3 V para garantizar la alimentación del módulo. El pin restante, GND, irá conectado a masa.

En cuanto a su configuración también se utilizan comandos AT, al igual que en el HC-06, que serán detallados en Tabla 3. Sin embargo, en este caso, esta no es previa a la instalación del módulo sino que se realizará directamente desde el programa principal de Arduino para el funcionamiento de la estación.

5. DISEÑO SOFTWARE

Este apartado describe, en primer lugar, el programa desarrollado para el correcto funcionamiento de la placa Arduino logrando así, que ésta realice las funciones necesarias para alcanzar los objetivos marcados. Además, se describirán las otras dos unidades que forman la estación, explicando su configuración, modo de desarrollo e interacción con el resto de unidades.

Para facilitar la comprensión de este apartado se incluirá, además de la explicación teórica, un ejemplo práctico. Dicho ejemplo, consiste en colocar el dispositivo descrito anteriormente en una habitación. Donde una persona fije los parámetros de la manera siguiente:

- Luminosidad inicial: 10 lux.
- Luminosidad final: 35 lux.
- Humedad relativa: 31,30 %
- Temperatura inicial: 22,90 ºC
- Temperatura final: 26 ºC
- Velocidad del viento: 20 %
- Presión: 965 hPa.

5.1 Arduino

Antes de comenzar es necesario conocer el entorno de desarrollo Figura 17, *IDE [1]* en inglés y como será nombrado partir de ahora, que permitirá la programación del microcontrolador así como el lenguaje utilizado para tal fin. Ambos están basados en *Processing [10],* un lenguaje de programación y entorno de desarrollo libre basado en el lenguaje *Java.*

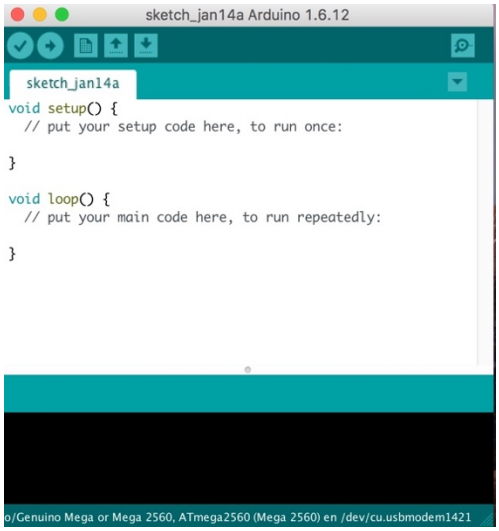

Figura 17. IDE Arduino

La configuración previa necesaria para establecer la comunicación entre el *IDE* y la placa es sencilla, únicamente es necesario seleccionar el modelo de placa que se va a utilizar y el puerto USB del PC donde será conectado. Todo ello se realiza desde la pestaña de *Herramientas,* como puede verse en la Figura 18.

Figura 18. Configuración de la placa

Una de las ventajas de este software es que ofrece una serie códigos ya desarrollados que pueden ser útiles a modo de ejemplo para todos aquellos usuarios que se inicien en la programación de este tipo de microcontroladores. Para acceder a ellos, hay que seleccionar la pestaña *Ejemplos* dentro de *Archivo.*

En cuanto al lenguaje de programación empleado, es similar a *C++* aunque acepta ciertas funciones en lenguaje *C.* Los algoritmos para *Arduino*, como puede verse en la Figura 17, se dividen en dos funciones: *setup y loop.* La primera de ellas, donde se incluyen todas las funciones de configuración, se ejecutará una única vez al inicio del algoritmo; mientras que la segunda, es un bucle en el que se desarrollan las funciones que forman el programa principal y se repetirá de forma infinita siempre que la placa esté alimentada.

Finalmente, el monitor de Puerto Serie, mediante el que es posible comunicarse con el microcontrolador. En esta pantalla se mostrarán los mensajes enviados desde el Arduino y a la vez, podrán enviarse mensajes a este utilizando el teclado del PC.

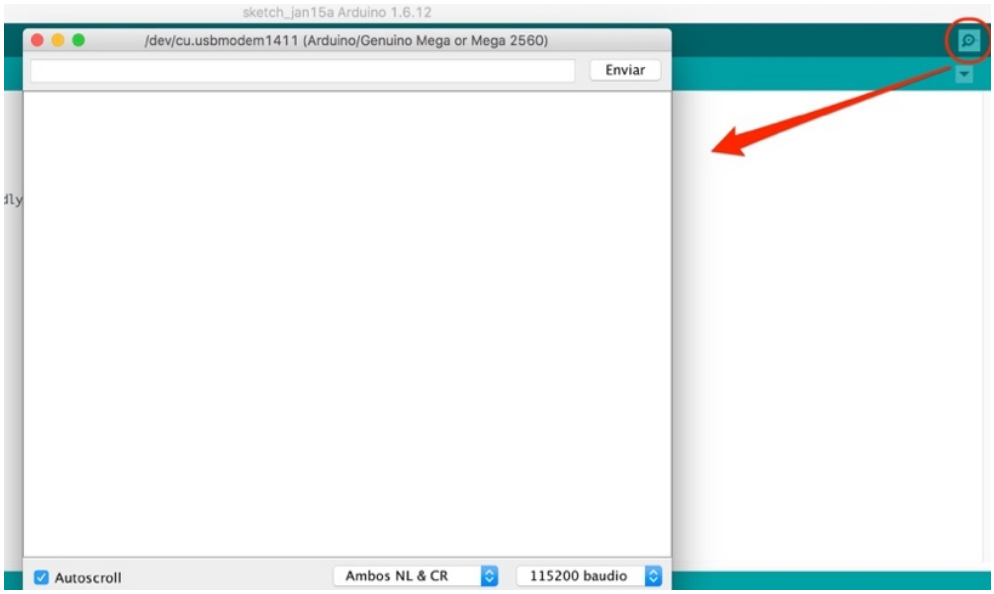

Figura 19. Monitor del Puerto Serie.

En la Figura 19 es posible observar como acceder a dicho monitor así como sus posibles configuraciones de ajuste de línea y la velocidad en baudios, bits por segundo.

5.1.1 Programa principal estación

En primer lugar, antes de entrar en la función *setup,* es necesario incluir todas las librerías que van a utilizarse para poder usar las funciones ya desarrolladas que permitan realizar de forma sencilla ciertos procedimientos. En este caso habrá que

incluir las librerías de cada uno de los sensores, de los dispositivos auxiliares, de la memoria EEPROM, del bus de comunicación I2C y una librería que permita crear Puertos Serial mediante Software.

Tras esto, hay que definir una serie de variables, objetos y estructuras que aparecerán a lo largo del algoritmo y que cumplirán funciones distintas. Algunos ejemplos serían los *flags* utilizandos en las alarmas, los objetos para cada tipo de sensor para poder utilizar sus corresondientes librerías o la estructura donde se han englobado todos las las variables atmosféricas.

El término *objeto* procede de la programación orientada a objetos, como es *C++* o *Java,* y puede definirse como una instancia de una *clase*, siendo esta una estructura en la que quedan definidos los atributos y operaciones disponibles para un objeto. En ese caso, ha sido necesario definir un objeto para cada uno de los sensores y dispostivios auxiliares con el fin de utilizar la funciones propias de su clase.

Una estructura es un tipo de dato que permite almacenar un conjunto de datos de distinto tipo. La ventaja de este tipo de dato es que permite hacer referencia al conjunto de los datos que almacena como un único dato o a cada uno de estos de forma individual. En este caso, se utilizará para almacenar los valores de las magnitudes atmosféricas y las variables del controlador.

Finalmente se declaran todas las funciones que han sido desarrolladas para ser llamadas desde el *loop* y que serán definidas más adelante. Se indicará el nombre de cada función junto con sus parámetros de entrada y de salida, en el caso de que existan.

Función Setup

En esta función, desarrollada en el Diagrama 1, se realizarán todas las configuraciónes necesarias para el correcto funcionamiento de los dispositivos durante el programa principal. En primer lugar se iniciarán, con una velocidad de 115200 baudios, los tres Puertos Serial que permitirán la comunicación con un PC, si el microcontrolador está conectado mediante un USB; y las comunicaciones inalámbricas vía Wifi y Bluetooth. Para los dos primeros casos se utilizaron los Puertos 0 y 2 mientras que, en el caso del Bluetooth se optó por una solución alternativa.

En primer lugar, se llevaron a cabo numerosas pruebas conectando el HC-06 a todos los Puertos Serial y tras estas, se llegó a la conclusion de que el módulo era capaz de enviar información a través de cualquiera de ellos pero no de recibirla. Como consecuencia, se creó un Puerto Serial mediante software utilizando la librería citada anteriormente y dos pines digitales del microcontrolador,evitando así los problemas mencionados.

A continuación se configuran como entradas o salidas cada uno de los pines digitales que van a intervenir en el programa. Los LEDs serán salidas y los pines asociados a las interrupciones serán entradas. Además, aparecen una serie de variables llamadas *flags*, configuradas como salidas, cuyos pines irán físicamente conectados a los pines de las interrupciones. El objetivo de esto es crear interrupciones internas, es decir, activadas vía software ya que el Arduino no permite este tipo de interrupciones. Cuando en el transcurso del programa aparece un evento que debe provocar una interrupción se activa su flag asociado por lo que aparecerá un nivel alto en su salida y así, simultáneamente, un nivel alto en la entrada asociada a interrupción que provocará una llamada a la función definida.

Posteriormente, se inicializan los sensores, el display LCD y las funciones asociadas a las interrupciones, *ISR* procedente del inglés *Interrupt Service Routine.* En este caso serán necesarias tres interrupciones, *INT0 INT1 e INT5,* que serán definidas más adelante.

Finalmente, se llamará a cuatro funciones que tendrán los siguientes objetivos:

- Resetear la memoria EEPROM, en caso de que el usuario lo solicitase utilizando el teclado matricial.
- Configurar valores necesarios para el funcionamiento del programa como son los valores máximos que puede tomar cada una de las magnitudes medidas, que más adelante utilizará el sistema de alarma; y el *setpoint* de una de las magnitudes y error máximo, necesarios para el funcionamiento del controlador. Estos valores podrán ser configurados utilizando el teclado matricial o la aplicación Android. Tras esto, los valores serán almacenados en la memoria EEPROM, evitando

de este modo que tengan que ser configurados siempre que se inicie el sistema.

- Mostrar estos valores en el monitor de Puerto Serie si el Arduino se encuentra conectado a un PC.
- Enviar al módulo ESP8266 los comandos necesarios para su configuración inicial, que serán explicados en apartados posteriores.

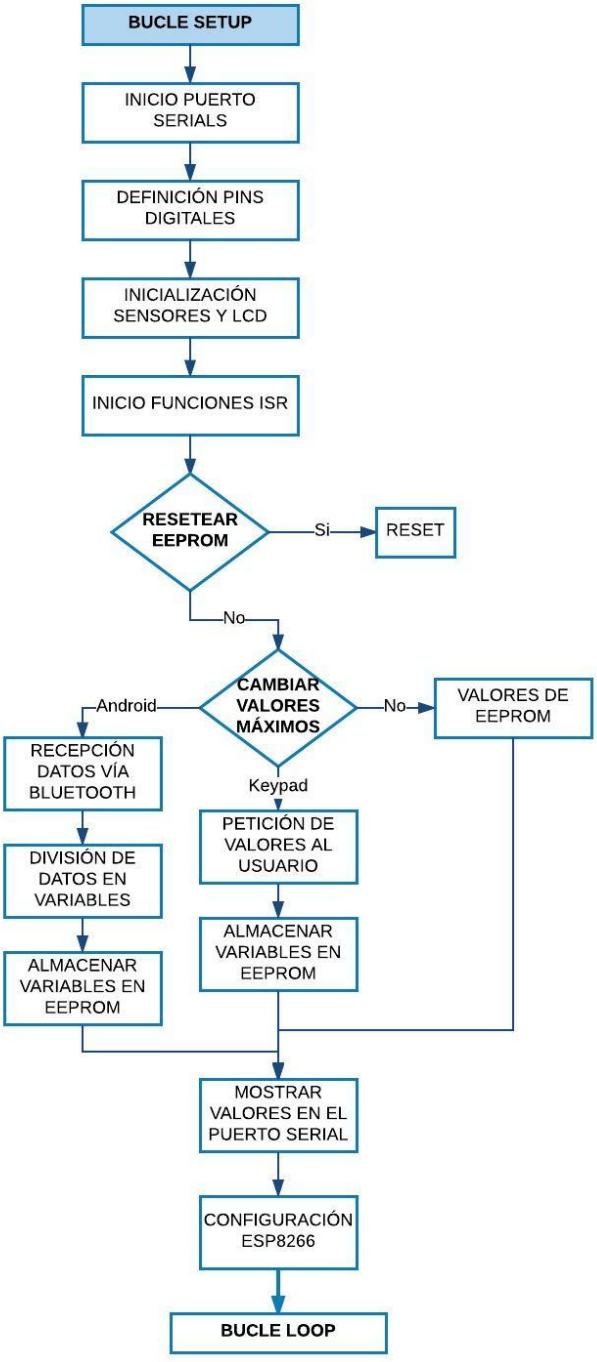

Diagrama 1. Funcionamiento del setup

.

En este inicio del programa es necesaria la intervención del usuario que, siguiendo con el ejemplo descrito anteriormente, realizará las siguientes acciones:

a. Habilitar el reset de la memoria EEPROM, eliminando así la configuración de un usuario anterior.
b. Definir los valores máximos de las variables y configurar el controlador utilizando cualquiera de los dos métodos descritos. En este caso, han sido configurados con los siguientes valores:

   - Luminosidad máxima: 30 lux.
   - Humedad relativa máxima: 40 %.
   - Temperatura máxima: 35 ºC.
   - Velocidad del viento máxima: 20 %.
   - Presión atmosférica máxima: 990 hPa.
   - Setpoint: 23 ºC.
   - Error máximo del controlador: 1 ºC.

Función Loop

Este bucle se repetirá cíclicamente, siempre que la placa de Arduino esté alimentada, por lo que desde aquí se llamará a todas las funciones que realizan las tareas necesarias para alcanzar los objetivos de la estación. De forma general, su funcionamiento se reduce a recorrer la secuencia de tareas mostradas en el Diagrama 2. Cada tarea será repetida con una frecuencia de aproximadamente 14 segundos.

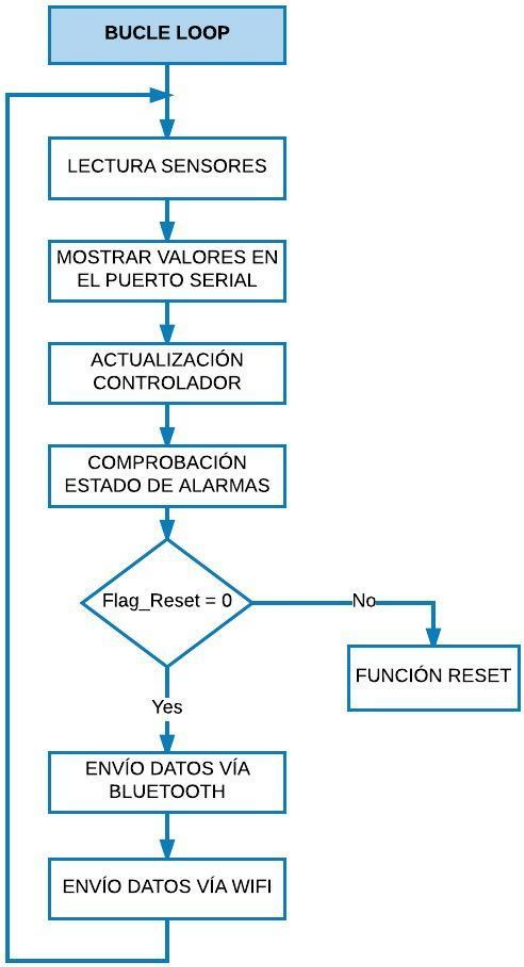

Diagrama 2. Funcionamiento del loop

Además, se muestra en la LCD el valor de la magnitud seleccionada por el usuario tras comprobar en numerosas ocasiones a lo largo del código el estado de una variable interna que determina la magnitud elegida. Este procedimiento será detallado más adelante.

Como será explicado a continuación con más detalle, se estudiarán dos ciclos de este bucle para el ejemplo definido anteriormente. Durante el primero todo transcurrirá sin incidentes mientras que en el segundo, sí se activará alguna de las funciones adicionales, como la alarma. Como consecuencia de esto, en ningún momento se activará el *flag* que fuerza el bloqueo del sistema; sin embargo, si se ampliase el estudio a un mayor número de ciclos y permaneciese la alarma activa, el desarrollo del *loop* sería interrumpido tras llamar a la función de reset.

Tras el *loop* aparece el código de todas las funciones que han sido desarrolladas para lograr correcto funcionamiento de todas las tareas que aparecen el en diagrama.

Lectura de los sensores

Para este procedimiento, Diagrama 3, se necesitará el pin digital donde está conectado el sensor y un valor entre 0 y 4 que determinará qué función será utilizada para obtener la magnitud deseada, ya que cada sensor tiene su propia función que viene desarrollada en su correspondiente librería. Para el caso del anemómetro no existe código desarrollado previamente, por lo que es necesario crear un algoritmo. El objetivo de este será hacer una conversión entre dos rangos de valores. El primero comprende los valores entre 0 y 1023, que es el rango de valores que devuelve una salida analógica de *Arduino;* y el segundo, indicará el valor de un porcentaje, es decir, un valor entre 0 y 100. Al no conocer las especificaciones el motor que constituye el anemómetro no es posible obtener la velocidad del viento por lo que se mostrará un porcentaje aproximado.

Esta función devolverá siempre una variable de tipo *float*, es decir, con un valor numérico con decimales; y será llamada una vez por cada una de las magnitudes.

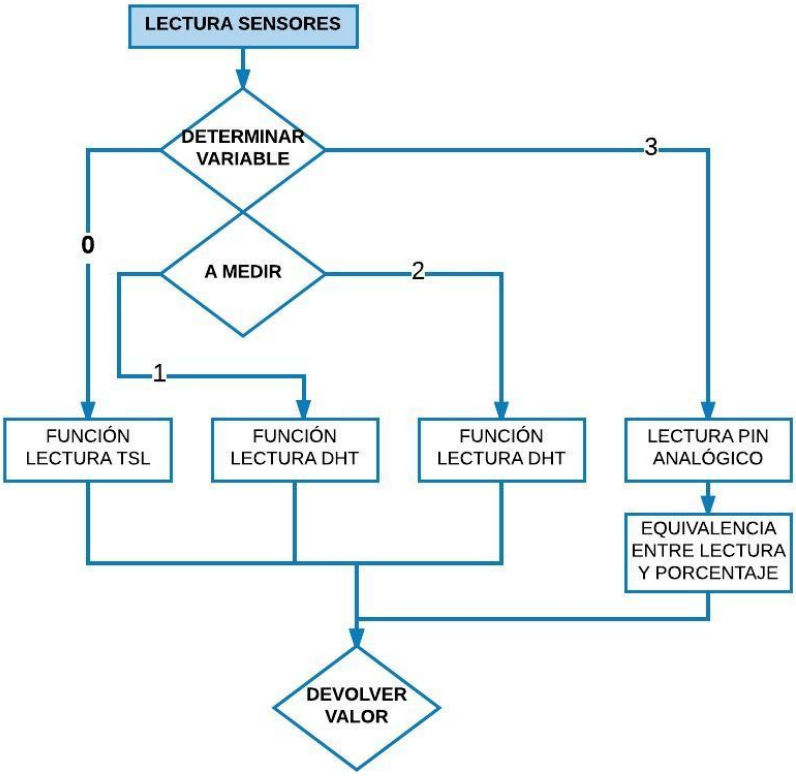

Diagrama 3. Función de lectura de los sensores

En el caso concreto descrito y suponiendo que se evalúan dos ciclos del *loop,* se realiza una lectura de cada uno de los sensores y se almacena este valor en la variable correspondiente. Tal y como fue definido, los valores de ambas lecturas serán los mismos en tres de las magnitudes mientras que se obtendrá un resultado diferente para la luminosidad y la temperatura.

Comprobación del estado de las alarmas

En primer lugar, es necesario saber que han sido configuradas dos tipos de alarmas: la primera permite detectar si el funcionamiento de alguno de los sensores es incorrecto y la segunda, avisará al usuario de que se ha alcanzado el valor configurado como máximo en alguna de las variables. En ambas situaciones, en el caso de que el estado de alarma persista, el sistema será bloqueado automáticamente y será necesario hacer *reset* en el microcontrolador para volver a poner el sistema en funcionamiento.

El procedimiento, tal y como aparece en el Diagrama 4, comienza comparando el valor de cada una de las magnitudes con un rango de valores entre cero y el valor máximo configurado. En el caso de ser un valor inferior a este rango se considerará que existe un error de lectura, ya que ninguna de las variables puede ser negativa; por lo que se activará el LED correspondiente, el *flag* que permite activar la interrupción, tal y como se explicó anteriormente; y se enviará un aviso a la nube mediante WiFi. Si, por el contrario, el valor es superior al rango se considera que el sistema ha entrado en estado de alarma. El proceso que se realiza es el mismo que en el caso anterior pero actuando sobre su LED y flag correspondientes. Evidentemente, si el valor se encuentra dentro del rango los dos LEDs se mantendrán apagados y los dos *flags* en nivel bajo.

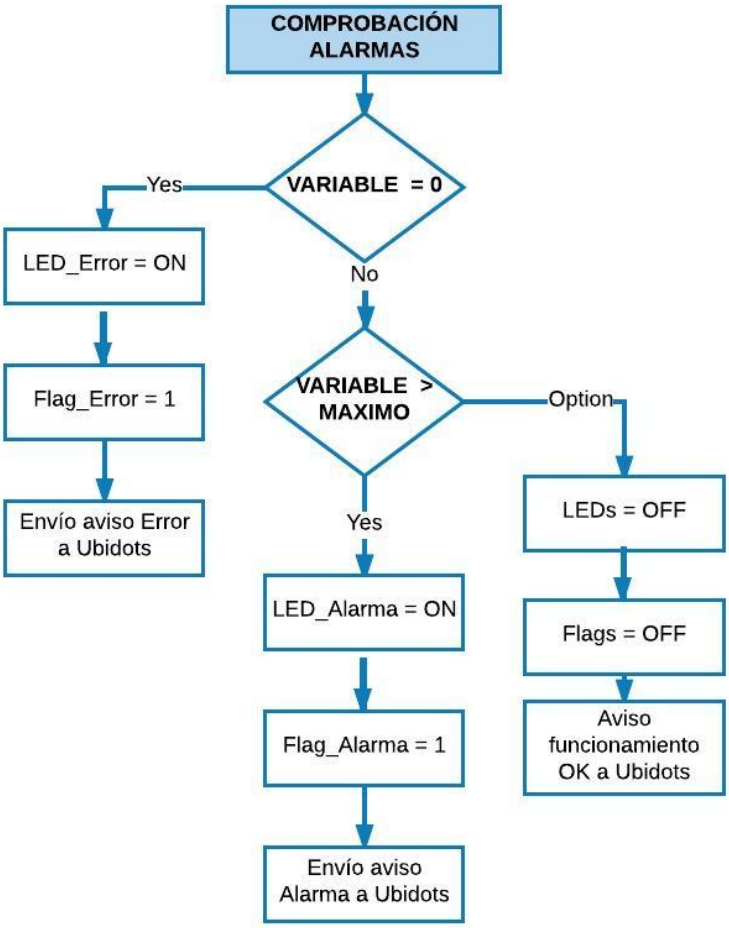

Diagrama 4. Función para comprobar alarmas

Finalmente, se comprueba si el *flag* que activa el bloqueo del dispositivo se encuentra activo y se actúa en consecuencia.

Diagrama 5. Función para detener el dispositivo

Según lo descrito, el primer ciclo del bucle *loop* del ejemplo transcurre con normalidad ya que ninguno de los valores supera los valores máximos. Sin embargo, tras la segunda lectura, la luminosidad tiene un valor de 35 lux por lo que se ejecutarán las acciones correspondientes a la alarma ya que la luminosidad máxima era de 30 lux. En el caso de que la luminosidad no disminuyese en ciclos posteriores, el sistema optaría por bloquearse y solicitar su reinicio.

Actualización del controlador

El controlador elegido es el ON-OFF, es decir, un tipo de controlador que sólo puede cambiar entre dos valores. La salida del interruptor será un nivel alto si el error del sistema es mayor que el error máximo permitido y un nivel bajo, en caso contrario. Este error será la diferencia entre el valor de referencia y el valor actual de la variable que se desea controlar.

En este caso, la variable controlada será la temperatura y los valores de referencia, definido como *setpoint,* y error máximo han sido configurados por el ususario en el *setup.* Tal y como muestra el Diagrama 6, al inicio de la función se calcula el error actual del sistema según la definición anterior. A continuación, se comprueba si el error calculado es mayor que el error máximo. En el caso de que este valor sea mayor, en valor absoluto, que el error máximo; el motor, que actúa como ventilador y permite disminuir la temperatura medida, se encenderá y a la vez, se activará un *flag* que informará a las demás unidades de su estado. En caso contrario, se desactivará el motor y el *flag.*

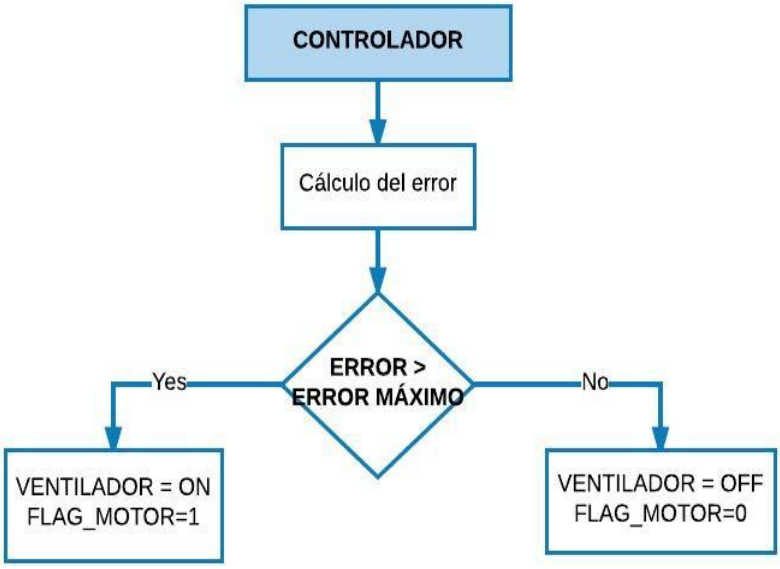

Diagrama 6. Función del controlador On – Off

En el caso práctico, en el primer ciclo del programa se obtiene un error de 0.1 ºC ya que la temperatura es de 22,9 ºC y el *setpoint es 23 ºC.* Teniendo en cuenta que el error máximo es de 1 ºC, no es necesario el funcionamiento del ventilador. Sin embargo, este si será necesario en el segundo ciclo para disminuir la temperatura como consecuencia de obtener un error, con una temperatura de 26 ºC, superior al máximo.

El segundo tipo de controlador que iba a formar parte del dispositivo es el PID [11], es decir, controlador proporcional-integral-derivativo y se define con la siguiente ecuación:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt} \qquad (1)$$

$u = variable\ de\ control$

$e = y_{setpoint} - y = error$

$K_p = Acción\ Proporcional = \ Constante\ de\ proporcionalidad.$

$T_i = Tiempo\ Integral.$

$T_d = tiempo\ derivativo$

La ventaja de este tipo de controlador es que, al aplicar simultáneamente las tres acciones, a lo largo del proceso se obtendrán las ventajas de las tres por lo que se obtiene un controlador de mayor precisión. De este modo, la acción proporcional aumentará la ganancia a la vez que disminuye el tiempo de respuesta aunque, en ese proceso se crea un error estático. Este error será eliminado utilizando la acción integral, aunque simultáneamente creará una oscilacion en la respuesta. Por último, la acción derivativa es una acción anticipativa que suaviza los cambios bruscos de la señal de control que, de este modo, eliminará las oscilaciones creadas por la acción anterior. Resumiendo, gracias al controlador PID es posible obtener un sistema con una respuesta más rápida y sin error estático.

El primer paso antes de diseñar un algoritmo para *Arduino* que ejecute las funciones de este tipo de controlador es obtener la función de transferencia que defina el sistema. Para ello se utilizará *Matlab*, concretamente la herramienta *ident;* la cual, *grosso modo,* devuelve esta función de transferencia si se le proporcionan unos vectores de entrada y salida obtenidos del funcionamiento del sistema.

Conectando únicamente el sensor de temperatura al microcontrolador y variando prograsivamente la temperatura del ambiente se han obtenido los vectores de datos mostrados en la Figura 20.

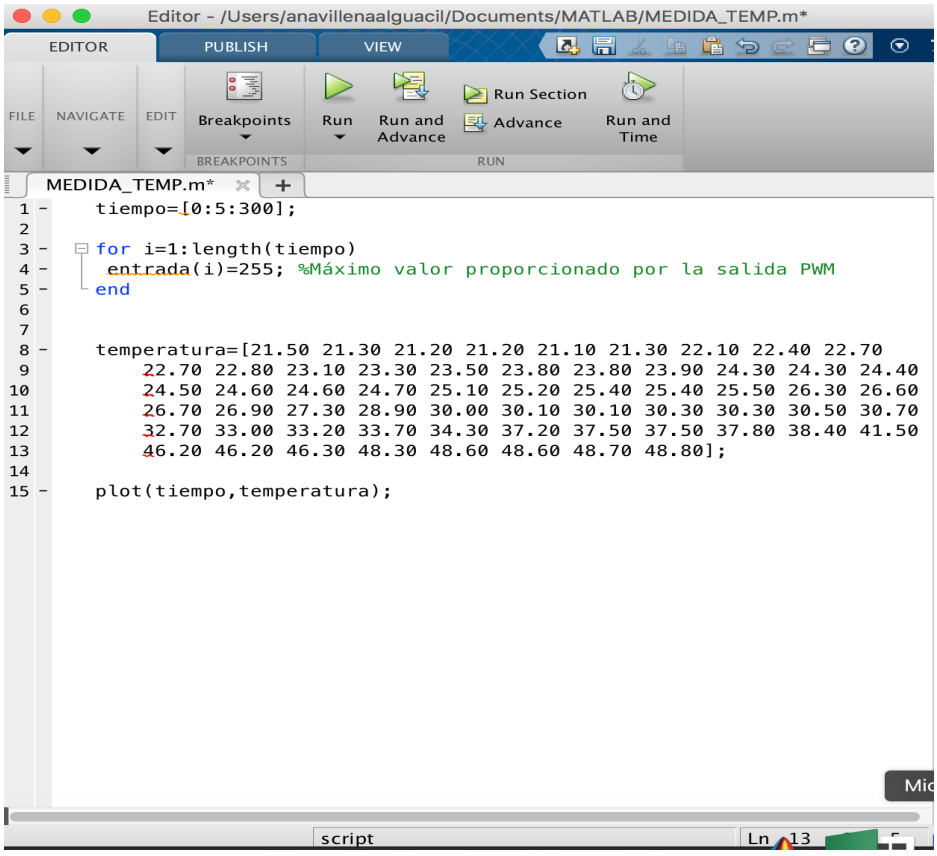

Figura 20. Vectores de datos

La frecuencia de muestreo es de 5 segundos durante un tiempo total de 300 segundos por lo que se obtiene un vector con 61 valores de temperatura. El vector *entrada* hace referencia al control del motor que, al realizarse mediante un puerto PWM, su valor máximo será de 255. En la Figura 21 se muestra el gráfico obtenido con estos valores.

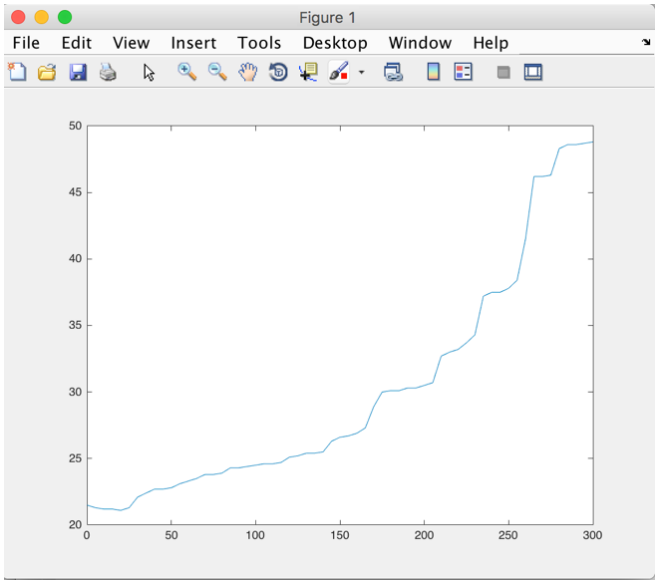

Figura 21. Gráfica tiempo-temperatura

Tras esto, desde el *Workspace,* se abrirá la herramienta *ident* por lo que aparecerá una ventana, Figura 22, desde la que importar los datos. Para ello hay que seleccionar *Time domain data* y definir *entrada* como *Input* y *temperatura* como *Output; y* cambiar *Sample Time* o el tiempo de muestreo a 5 segundos

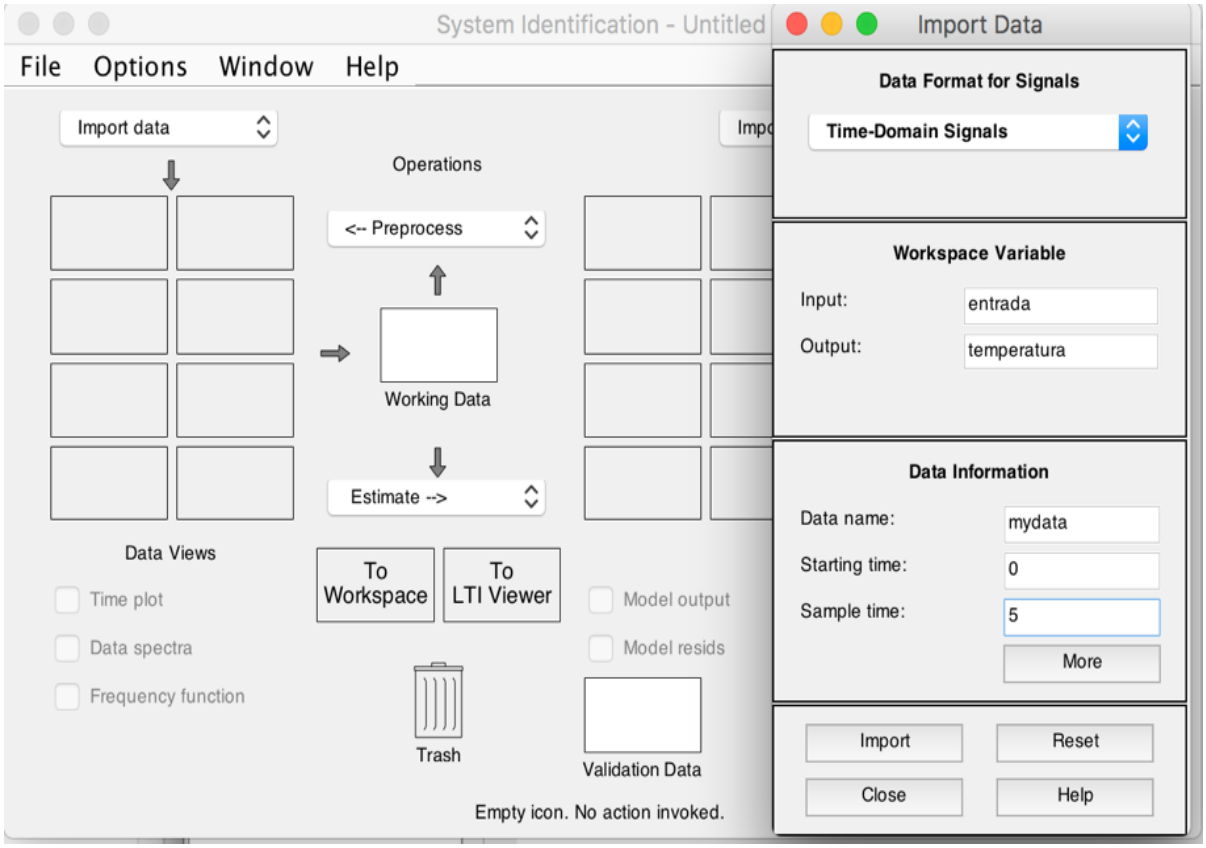

Figura 22. Pantalla inicio herramienta ident

El siguiente paso es estimar el modelo del proceso para lo que hay que abrir el desplegable *Estimate,* pulsar *Process Models* y finalmente *Estimate.* En la parte derecha, *Model Views,* aparecerá el modelo que habrá que arrastrar hasta *Workspace*.

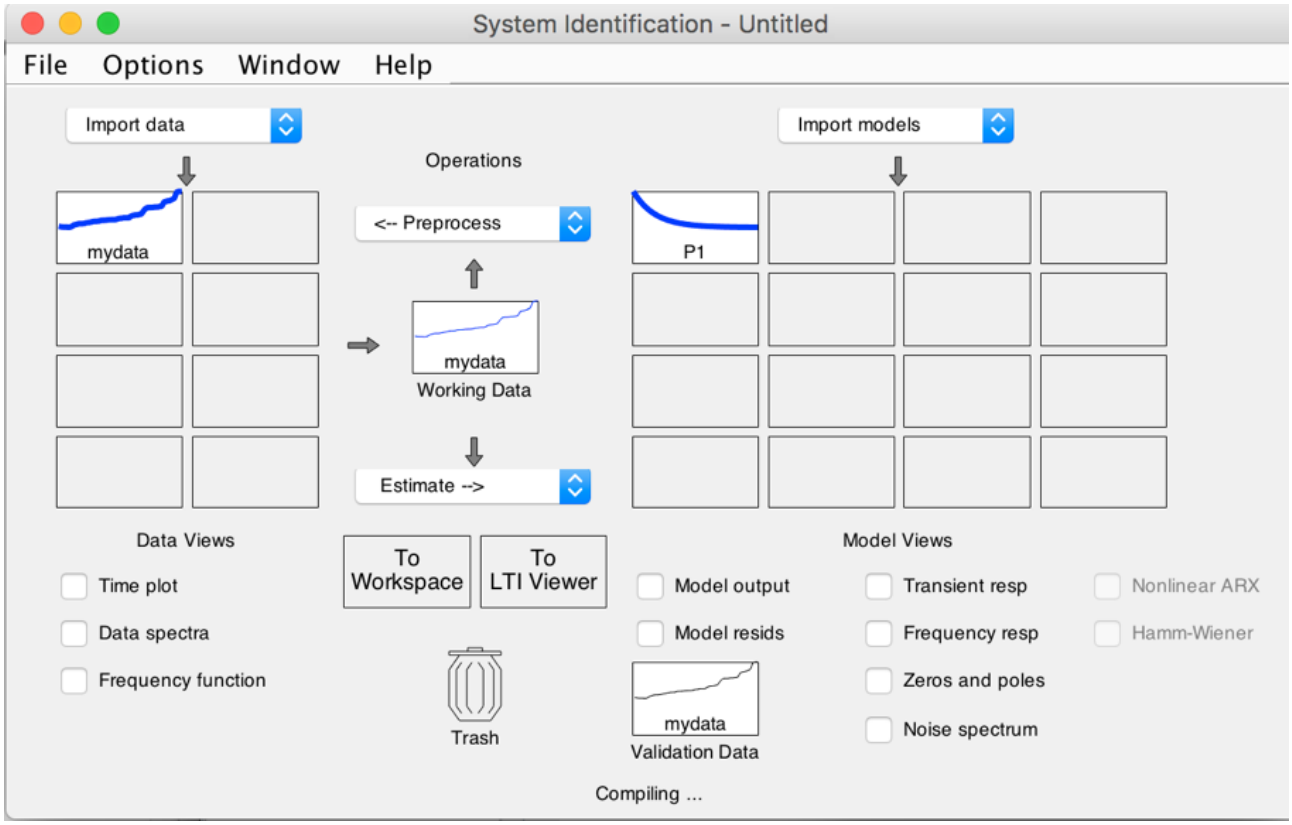

Figura 23. Importación al Workspace

Una vez de vuelta en el *Workspace,* haciendo una llamada al modelo *P1,* aparecerá la función de transferencia del sistema y las características del modelo, Figura 24.

```
>> ident
>> P1

P1 =
Process model with transfer function:
             Kp
  G(s) = ----------
          1+Tp1*s

        Kp = 0.63756
       Tp1 = 1391.7

Name: P1
Parameterization:
    'P1'
   Number of free coefficients: 2
   Use "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using PROCEST on time domain data "mydata".
Fit to estimation data: 58.29%
FPE: 13.75, MSE: 12.46
```

Figura 24. Características del modelo

El siguiente paso sería el diseño del controlador PID utilizando el *lugar de las raices* mediante la herramienta *rltool*, sin embargo, el proceso se detuvo aquí ya que, como puede observarse en la imagen superior el porcentaje de estimación es de un 58,29%. Es un valor insuficiente considerando que al ejecutar este proceso con toda la estación en funcionamiento el tiempo de muestreo ascendería hasta unos 15 segundos por lo que la estimación alcanzada con la función de transferencia sería menor y como consecuencia, el controlador PID no cumpliría sus funciones de un modo correcto.

Envío de datos vía Bluetooth

Esta función consiste, como aparece en Diagrama 7, en concatenar en un único mensaje todas las variables que van a ser enviadas separadas por un carácter que permita, en la siguiente unidad de la estación, conocer cuál es el inicio y el fin de cada uno de estos datos. En este caso se enviarán una variable por cada magnitud medida, el valor del *setpoint* para el fucionamiento del controlador y tres *flags* que permitirán informar a la aplicación del estado de las alarmas y del ventilador. Tras esta concatenación únicamente es necesario enviar el mensaje a través del Puerto Serial correspondiente que en este caso es el que fue desarrollado por software.

Diagrama 7. Función envío Bluetooth

En el ejemplo, los mensajes concatenados y envíados a través del Puerto Serial en los dos ciclos estudiados son:

10;31,3;22,90;7;965;23;0;0;0;

35;31,3;**26**;7;965;23;**1**;0;0;

La diferencia entre ambos, además de los valores de las variables,es el envío de un nivel alto en el flag que informa de la activación de la alarma.

Envío de datos vía WiFi

El primer paso es la configuración del módulo ESP8266 que se realizará, como se mencionó anteriormente, con una función que será llamada desde el *setup.* En esta, únicamente se enviarán a través del Puerto Serial 2 los comandos de la Tabla 3 necesarios para modificar ciertos parámetros del módulo.

Tabla 3. Comandos AT de configuración del módulo ESP8266.

Comando	Función	Valor
AT	Verificar conexión	-
AT + RST	Resetear el módulo	-
liAT + CWMODE= *Valor entre 0 y 3*	Modo de funcionamiento del módulo	3
AT + CWJAP="*Nombre punto acceso*","*Clave de acceso*"	Conectar el módulo a un punto de acceso cercano	A elección del usuario.
AT + CIPMUX= 0 ó 1	Habilitar conexiones simultáneas	1
AT + CIPSTART=0,"TCP","dirección ó IP", nº de puerto	Iniciar comunicación TCP/IP la dirección deseada.	Dirección:things.ubidots.com Puerto: 8080

Una vez que el módulo está configurado, conectado al punto de acceso deseado y se ha establecido la comunicación con la plataforma IoT pueden enviarse datos a esta desde el *Arduino.* Este envío puede realizarse utilizando las funciones de una

librería desarrollada para la comunicación entre el módulo ESP8266 y la plataforma IoT. Sin embargo, esta es únicamente válida cuando el módulo está conectado a *Arduino Uno*, por lo que no puede utilizarse en este caso y ha sido necesario implementar una función que desarrolle esta tarea, Diagrama 8.

Para ello, en primer lugar hay que tener en cuenta qué instrucción AT se utiliza para enviar datos con este módulo:

AT+CIPSEND=0,N

siendo N el tamaño del mensaje que se desea enviar

Este comando solicita el envío de un número determinado de caracteres, por eso es necesario especificar el tamaño del mensaje. Como consecuencia de esto, antes de ejecutar la solicitud y el posterior envío se implementa un algorítmo que calcula el tamaño del mensaje y lo almacena en una variable. Posteriormente, se envía el siguiente mensaje a través del Puerto Serial:

POST /api/v1.6/variables/ID/values http/1.1

Host: things.ubidots.com

User-Agent: Arduino-ESP8266/1.0

X-Auth-Token: TOKEN

Content-Type: application/jsos

Content-Length: tamaño del mensaje

{“value”: valor}

Todo lo relacionado con la plataforma, incluyendo la comunicación con el microcontrolador, será el objeto del siguiente apartado; por lo que será entonces cuando se explique en detalle este fragmento de código.

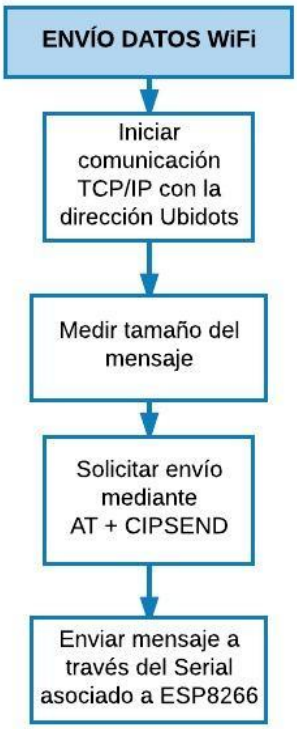

Diagrama 8. Función envío WiFi

Para el caso descrito, los valores enviados a la plataforma en cada ciclo son los mismos que los enviados vía Bluetooth; siendo la única diferencia que en lugar de enviar un mensaje con todos los valores, será necesario hacer una llamada a la función para cada una de las variables.

Interrupciones

Las interrupciones son ejectudadas cuando se dispara un evento concreto, en este momento, la ejecución del programa principal se interrumpe para llevar a cabo una función especial definida como rutina de servicio de interrupción a la que se hará referencia como *ISR,* procedente del inglés *Interrupt Service Routine.* Una vez que se han ejecutado las tareas definidas en la ISR, el proceso vuelve al punto del programa principal donde se había detenido.

Existen tres tipos de eventos que pueden provocar una interrupción:

- Un evento hardware,es decir, un evento externo.
- Un temporizador o evento programado
- Un evento software, es decir, definido en el algoritmo del programa.

Sin embargo, Arduino sólo soporta los dos primeros tipos. Como paso previo a su desarrollo, estas *ISR* deben ser activadas en el *setup* y pueden ser configuradas con condiciones de disparo distintas:

- *LOW.* La interrupción se dispara cuando el estado del pin asociado es bajo.
- *CHANGE.* La interrupción se dispara cuando hay un cambio en el estado del pin.
- *RISING.* La interrupción se dispara en el flanco de subida.
- *FALLING*. La interrupción se dispara en el flanco de bajada.

En el caso concreto de la estación meteorológica, se necesitan interrupciones activadas vía software pero Arduino solo soporta los dos primeros tipos. La solución aplicada, como se explicó en el apartado de setup, consiste en definir unas variables llamadas *flags* que serán activados cuando la interrupción deba ejectuarse. Estos flags están definidos como salidas digitales del Arduino y están conectadas físicamente a los pines digitales a los que están asociadas las interrupciones por hardware. Como consecuencia,cuando uno de los flag pase a nivel alto, el pin asociado a la interrupción también lo hará y así, se ejecutará la ISR.

En el algoritmo de la estación se han configurado tres interrupciones, dos de las cuales se encangan de la gestión de las alarmas mientras que la otra permite modificar el mensaje mostrado en el display LCD.

El funcionamiento de las dos primeras es el mismo, únicamente cambia el mensaje mostrado en la LCD y el *flag*, mediante el que se informa al resto de unidades de que la interrupción se ha disparado. La primera de ellas, *INT0,* está asociada al pin digital 2 y se ejecuta cuando el valor de una de las variables supera su valor máximo; la otra, *INT1*, se activa mediante el pin digital 3 y está asociada a un error de lectura en alguno de los sensores. Al ser ejecutadas, ambas mostrarán un mensaje de alerta e incrementarán un contador, ya que no será hasta el tercer mensaje de alerta cuando el sistema se bloquee y se solicite su *reset;* tal y como muestra el Diagrama 9.

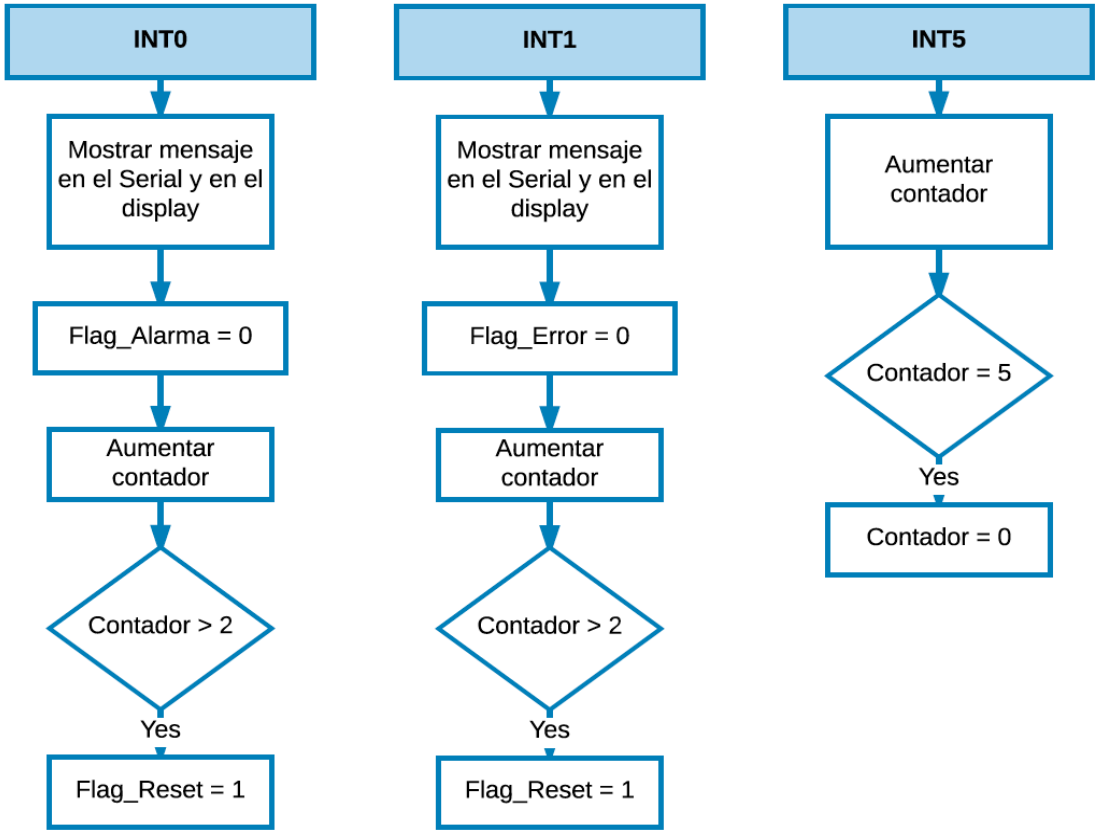

Diagrama 9. Funciones asociadas a las interrupciones

La interrupción encargada de la gestión del display, *INT5,* sí es una interrupción hardware ya que será el usuario, mediante un pulsador conectado al pin digital 18, el que solicite su ejecución. En su código únicamente aparece un contador que será incremetado con cada pulsación y reseteado al alcanzar el valor 5, el número de variables diferentes que es posible mostrar. Este valor será el parámetro de entrada de una función ejecutada de forma repetida en el *loop* que, de este modo, seleccionará que variable debe ser mostrada. Este proceso sólo se interrupmirá si alguno de los *flags* de alarma se encuentre activo ya que, en ese caso, el display mostrará los mensajes de error.

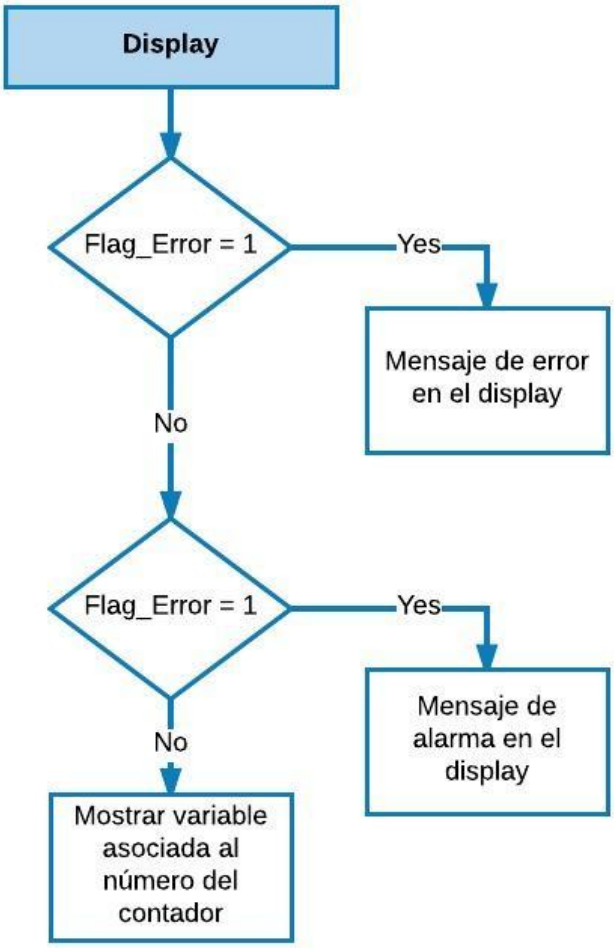

Diagrama 10. Función cambiar mensaje display

Este procedimiento, Diagrama 10, se tomó como solución obligada ya que, tras varias pruebas, no se consiguió modificar el estado del display desde el código de la propia *ISR*.

Haciendo referencia al ejemplo, durante la primera ejecución del *loop* las funciones asociadas a *INT0 e INT1* no intervienen ya que no se ha activado el flag asociado a la alarma ni se ha mencionando en ningún momento que exista un fallo en alguno de los sensores. La situación en el segundo ciclo es distinta ya que el flag de la alarma si ha sido activado y esto hace que se ejecuten las acciones definidas en la función asociada a *INT0.* El contador de esta función se aumentará hasta que su valor sea superior a 2, momento en el que se activará el flag que fuerza el bloqueo del dispositivo.

En cuanto a la función asociada a la *INT5,* ésta podrá ser llamada en ambos ciclos tantas veces como el usuario desee cambiar el mensaje mostrado en el display. Sin embargo, durante la segunda ejecución, y debido al estado de alarma, el mensaje mostrado está predefinido y es independiente del valor que tenga el contador de la función en cada momento; por lo tanto, no tiene sentido el uso del pulsador del display si el sistema se encuentra con la alarma o el error de lectura activos.

5.2 Ubidots: plataforma IoT

La segunda unidad de la estación meteorológica es el servicio de nube que, en este caso y por motivos que serán explicados a lo largo de este apartado, será una plataforma IoT llamada Ubidots[2]. Esta plataforma permite establecer una comunicación con múltiples dispositivos, incluidos los microcontroladores; y de este modo, recibir una serie de datos con el objetivo de ser monitorizados en tiempo real y almacenados al mismo tiempo.

5.2.1 Internet de las cosas (IoT)

El Internet de las Cosas [6][7][8] es un concepto que procede del inglés *The Internet of Things* y al que desde este momento se hará referencia como *IoT.* La primera definición de este concepto es “la interconexión digital de todo tipo de objetos cotidianos a través de la Red”.

El objetivo de esta tecnología que está creciendo rápidamente es que cualquier objeto de uso diario, no únicamente los dispositivos electrónicos, cuenten con una dirección *IP* mediante la cual puedan ser identificados y conectados a Internet. Gracias a esto, estos objetos serían capaces de enviar y recibir información.

Figura 25. Representación IoT

Para lograr el funcionamiento de esta tecnología es necesario una amplia gama de microcontroladores, sensores y módulos de comunicación que han sido diseñados para tal fin. Estos serán los encargados de medir las magnitudes, procesarlas para convertirlas en datos y posteriormente enviarlas para que sea posible su análisis y gestión.

El segundo requerimiento para poner en funcionamiento el *IoT* es un software que permita el análisis de los datos que son enviados a la Red. Es en este punto donde surgen el concepto de *Cloud Computing [14] [15]* o Computación en la Nube, que engloba un conjunto de servicios en Internet que permiten la comunicación con dispositivos conectados a este logrando así la recepción y el envió de información. Además, este grupo de servicios permiten el almacenamiento y el análisis de dichos datos y por supuesto, cuentan con una *interface* que permite a los usuarios visualizar el estado de las variables y estudiar los resultados obtenidos del análisis de los datos. Este servicio de Nube, como también es llamado, es el objeto de este apartado por lo que sus prestaciones y funcionamiento serán definidos en secciones posteriores.

Por último, es importante conocer cuál es el alcance de esta tecnología y tras lograr su introducción en los hogares, como por ejemplo en el control de iluminación o del sistema de calefacción; el siguiente objetivo es establecer el concepto de *Smart Cities* o Ciudades Inteligentes. En estas, gracias a esta interconexión digital, será posible la medida y gestión automática, sin intervención humana, de numerosas magnitudes, como la energía consumida; o dispositivos, como los semáforos; que forman parte del día a día de una ciudad.

La implantación de esta tecnología, tanto en uso doméstico como a gran escala, supondría una mejora en la eficiencia de todo tipo de procesos lo que llevaría a un ahorro energético y por tanto económico tanto a nivel del individuo como global.

5.2.2 Análisis y selección de plataformas IoT.

Al ser una tecnología en auge son muchos los servicios de nube que pueden encontrarse siendo muchos de pago, otros gratuitos y otros “mixtos”. Estos últimos ofrecen una amplia gama de prestaciones gratuita y además, de forma adicional y tras abonar una cuota cuentan con servicios de nivel más avanzado. En este caso, se han comparado las prestaciones de tres de las plataformas agrupadas en el grupo de “mixtas” para encontrar aquella que se adapte más a las necesidades actuales del proyecto sin limitar una posible ampliación de servicios en el futuro.

Para alcanzar los objetivos de la estación meteorológia esta plataforma debe ser capaz, como mínimo, de recibir la información de los sensores desde *Arduino* a través

del módulo *ESP8266,* representarla en tiempo real para que el usuario pueda interpretarla y por último, almacenar todos los datos recibidos.

La primera opción fue *ArduinoCloud* [12] ya que al pertenecer a la misma plataforma la conexión utilizando el *IDE Arduino* sería directa y más sencilla, sin embargo únicamente puede comunicarse con el Arduino utilizando uno de los siguientes módulos: *Yùn Shield, MKR1000 o WIFI SHIELD 101.* Por lo tanto, al no cumplir uno de los requisitos mínimos, está opción fue rechazada sin necesidad de conocer sus características y servicios.

Las otras dos opciones valoradas fueron *Thingspeak [13]* y *Ubidots [2].* Su funcionamiento es bastante similar ya que ambas están basadas en una *API REST*, concepto que será desarrollado posteriormente, por lo que a la hora de ser configuradas no existe ninguna diferencia destacable.

La primera diferencia encontrada entre ambas es que *Ubidots* ofrece más posibilidades a la hora de representar los datos, apartado en el que *Thingspeak* está más limitado. Esta característica no es fundamental pero sí permite crear un entorno gráfico más atractivo y funcional, ya que es posible adaptar la representación al tipo de magnitud que se desea mostrar.

Por otro lado, la característica más destacada de *ThinkSpeak,* es que ofrece un enlace directo con *Matlab* donde es posible llevar a cabo un análisis más exhaustivo de las variables. Sin embargo, este servicio no entraba dentro de las necesidades del proyecto por lo que la plataforma no sería aprovechada en su totalidad.

Por último, entre las funciones adicionales de *Ubidots* se encuentra la creación de eventos, entre los que se incluyen los avisos vía SMS o email. Esta característica sí resulta interesente y supondría una mejora en el diseño inicial de la estación ya que permitiría al usuario ser informado en caso de funcionamiento anómalo en el dispositivo.

Teniendo las prestaciones de una plataforma y otra, se optó por *Ubidots* ya que sus características resultaban más útiles, observando el planteamiento de este proyecto concreto, que las ofrecidas por *ThingSpeak.*

5.2.3 Estructura y configuración de *Ubidots*.

Ubidots [2] se define como un servicio en la nube que permite almacenar e interpretar información de sensores en tiempo real, haciendo posible la creación de aplicaciones para el Internet de las Cosas de una manera fácil, rápida y divertida.

El primer paso para utilizar *Ubidots* es completar un registro gratuito el cual proporcionará un usuario y contraseña, Figura 26, que permitirá la creación y modificación de tantos proyectos como el usuario necesite, siempre teniendo en cuenta la limitación de las prestaciones ofrecidas al no adquirir la versión de pago.

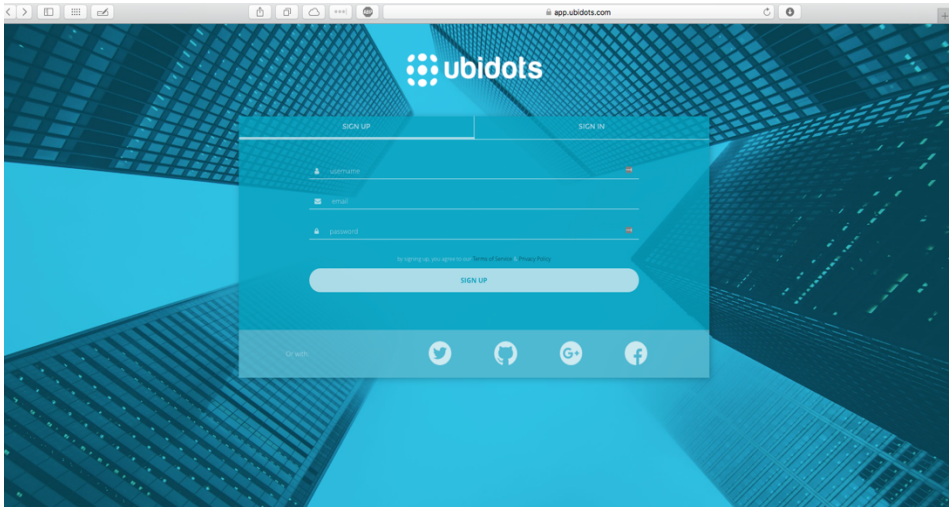

Figura 26. Inicio Ubidots

Una vez dentro del sistema aparecerán tres pestallas que permitirán acceder a las distintas ventanas que forman la nube:

- *Dashboards* o escritorios. Monitorización en tiempo real de las variables cuyos datos están siendo enviados a la nube.
- *Sources* o fuentes. Configuración de sistemas, incluyendo las variables que lo forman.
- *Events* o eventos. Creación de alertas y avisos al usuario.

A continuación, se establecerán los pasos necesarios para configurar la plataforma con el objetivo de alcanzar los objetivos marcados. De forma simultánea, se describirá la nube diseñada para la estación meteorológica.

Sources

El primer paso para poner el funcionamiento la nube es la configuración de variables y para ello hay que crear una *source* general que las englobe. Para ello únicamente hay que darle un nombre, *Estación* en este caso concreto, y pueden añadirse algunas etiquetas que permitan clasificarlos en el caso de existir numerosos proyectos. Además, como puede verse en Figura 27, se muestra el tiempo transcurrido desde la última vez que la fuente estuvo activa.

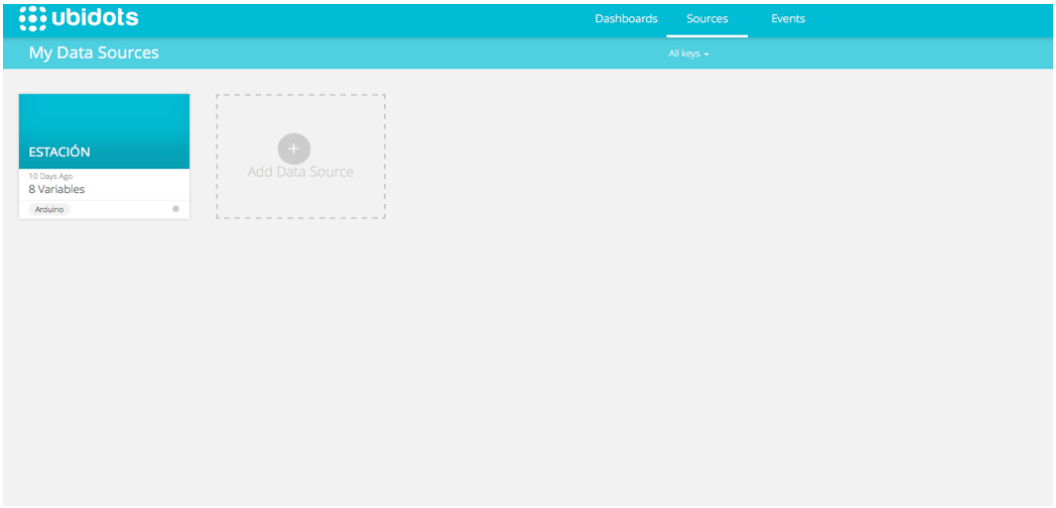

Figura 27. Pantalla de Sources

Haciendo clic en *Estación* se accede a una nueva pantalla, Figura 28, donde aparecen todas las variables que ya han sido creadas. Al igual que con la *Source,* cada una de ellas muestra su nombre, último dato recibido y tiempo transcurrido desde que este se recibió. Es posible añadir variables nuevas, para lo que sólo es necesario darle un nombre que las identifique.

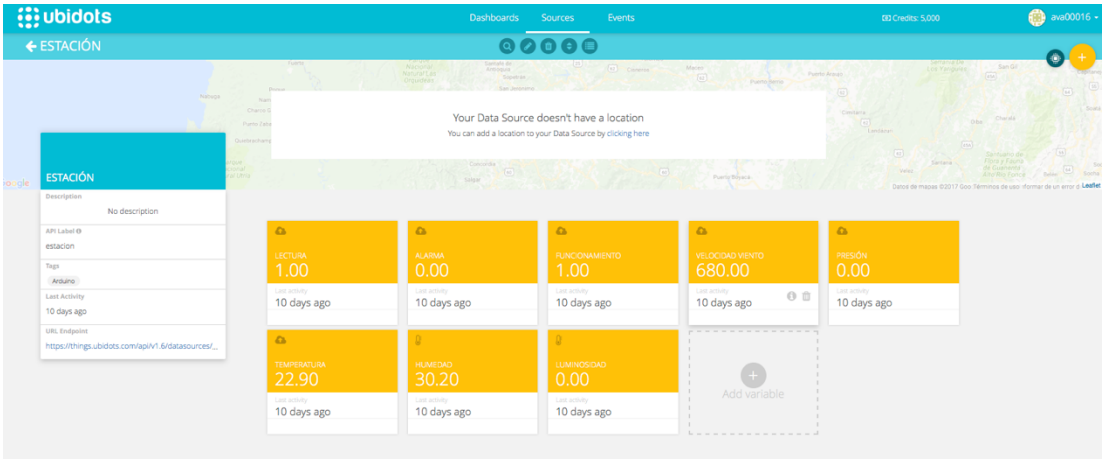

Figura 28. Pantalla de creación de variables

Entrando en cualquiera de las variables aparerá una pantalla en la que es posible visulizar un histórico con los datos de esta. En primer lugar aparece un gráfico en el que es posible configurar la fecha, o rango de fechas, que el usuario desea visualizar; así como el tipo de dato: individuales, media, mínimo, máximo o sumatorio. En la parte inferior, hay una tabla con todos los datos almacenados de esta variable, ordenados de forma cronológica.

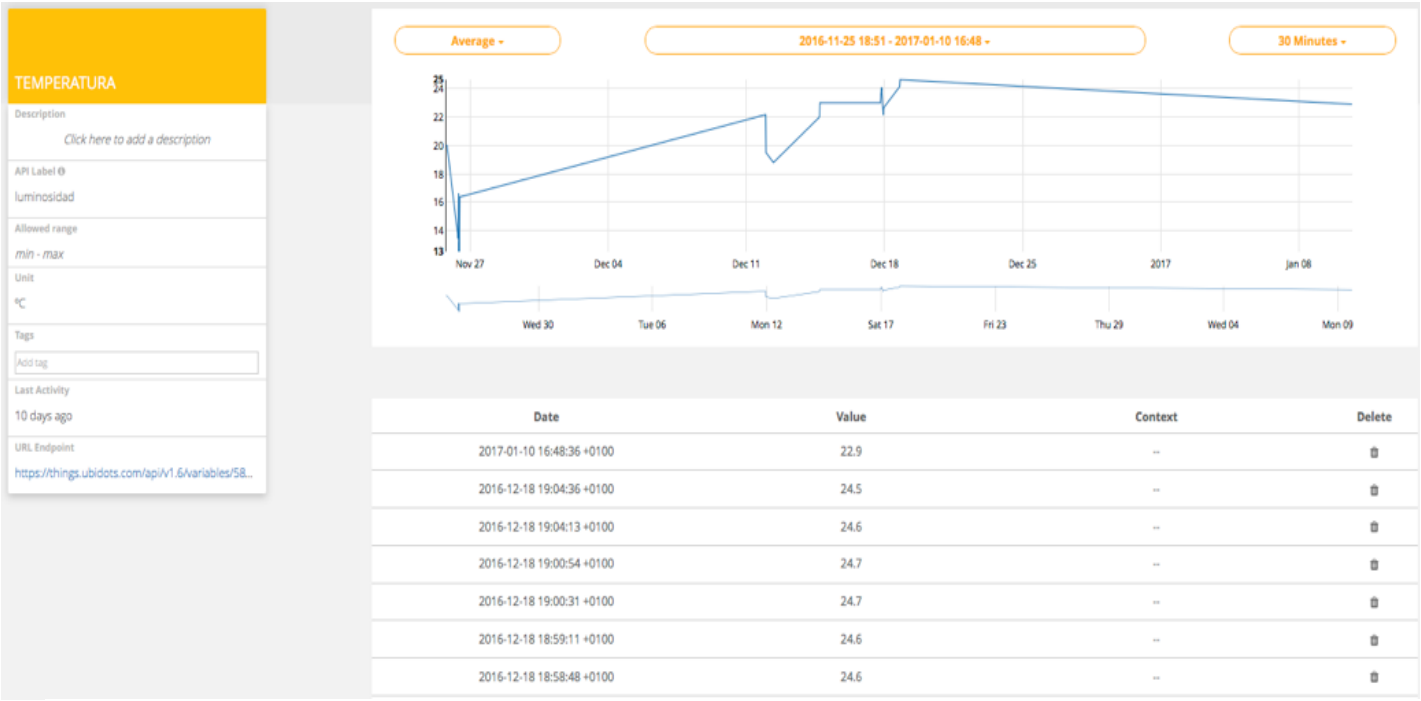

Figura 29. Datos de una variable concreta

Estos valores pueden ser exportados en formato .csv y enviados a una dirección de correo deseada pulsando en uno de los botones de la parte superior de la pantalla.

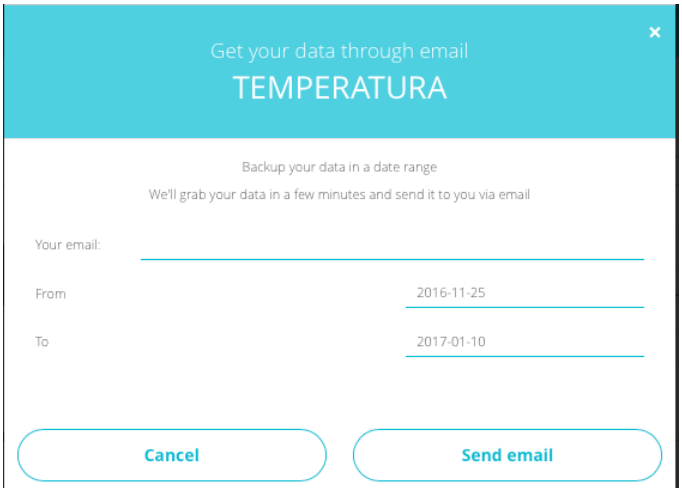

Figura 30. Solicitud de envío archivo CSV

Por último, en el lado izquierdo de la Figura 29 puede verse un resumen de la variable; donde puede añadirse una descripción, una etiqueta, la unidad en la que se mide la variable y definir un rango de valores máximos y mínimos de la variable.

Dashboard

En esta sección de la plataforma es posible crear tantos escritorios como se desee para visualizar la evolución de las variables en tiempo real mediante distintos *widgets* que pueden ser elegidos por el usuario en función de sus necesidades o preferencias. Entre estos *widges* se encuentran:

- *Chart* o gráfico. Permite representar los datos de una o varias variables en un histograma o en un gráfico utilizando tanto puntos como líneas.
- *Metric* o afirmación. Muestra el valor medio, máximo, mínimo, suma o último valor dentro de un periodo de tiempo acompañado de una oración que define a que se corresponde dicho valor. Por ejemplo, “la temperatura máxima la semana pasada fue de 34 ºC”.
- *Map* o Mapa. Permite mostrar la ubicación del dispositivo, en el caso de que este cuente con un GPS.
- *Table* o Tabla. Permite colocar una tabla que muestre el último dato recibido para una variable o el histórico de datos de dicha variable.
- *Indicator* o Indicador. Muestra el valor de una variable de una forma gráfica, sin mostrar el valor numérico. Permite seleccionar entre un indicador de dos posiciones o uno analógico que muestra los valores que se encuentran dentro de un rango predefinido.
- *Control* o Controlador. Permite enviar valores desde la plataforma al microcontrolador. Las dos opciones dentro de este apartado son interruptor de dos posiciones, On-Off, o un regulador que permite variar el valor dentro de un rango.

En la Figura 31 es posible observar las opciones mencionadas anteriormente y el proceso que hay que seguir para crear el *widget* deseado.

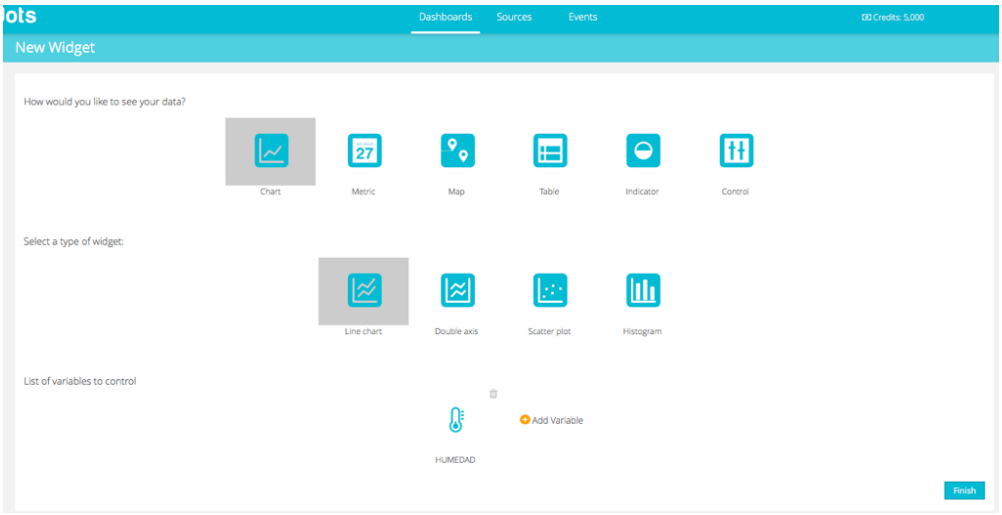

Figura 31. Ejemplo de creación de un nuevo widget

En este caso concreto se han creado tres escritorios o *dashboard* gracias a los que será posible visualizar las variables meteorológicas y además, controlar el estado del dispositivo, es decir, saber si se encuentra en funcionamiento o si alguna de las alarmas ha sido activada.

En primer lugar, el escritorio “ESTADO ESTACIÓN” donde aparecen tres indicadores de dos posiciones, uno para conocer si esta está encendida y otros dos para el sistema de alarmas.

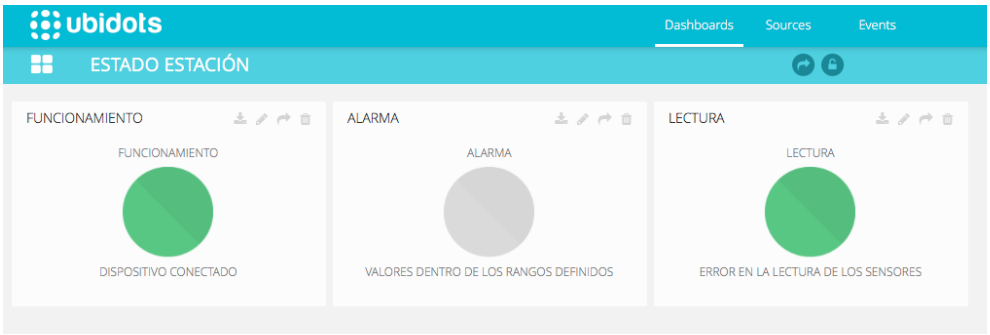

Figura 32. Dashboards Estado Estación

Estos LEDs ,mostrados en Figura 32, varían del mismo modo en el que lo hacen los *flags* mencionados en el apartado anterior y, cada uno de ellos, está acompañado de un mensaje aclaratorio tanto en la posición *On*, cuando el LED es de color verde; como en la de *Off,* color gris.

Volviendo al ejemplo concreto, el LED de funcionamiento aparecerá activo durante los dos ciclos estudiados y solo se desactivará si el dispositivo entrase en estado de bloqueo. En cuanto a los otros dos LEDs, el de lectura no se activará y el de alarma se encenderá tras el segundo envío de datos desde el microcontrolador.

El segundo escritorio, “GRÁFICO VARIABLES” está formado por cinco gráficos, uno para cada una de las variables, que se irán actualizando con cada nuevo valor que reciba la plataforma desde el microcontrolador. Este escritorio, Figura 33, será de especial utilidad para comprobar que el funcionamiento del controlador ON-OFF es correcto ya que permitirá controlar que los valores siempre se encuentran dentro del rango formado por el *Setpoint* y el error máximo.

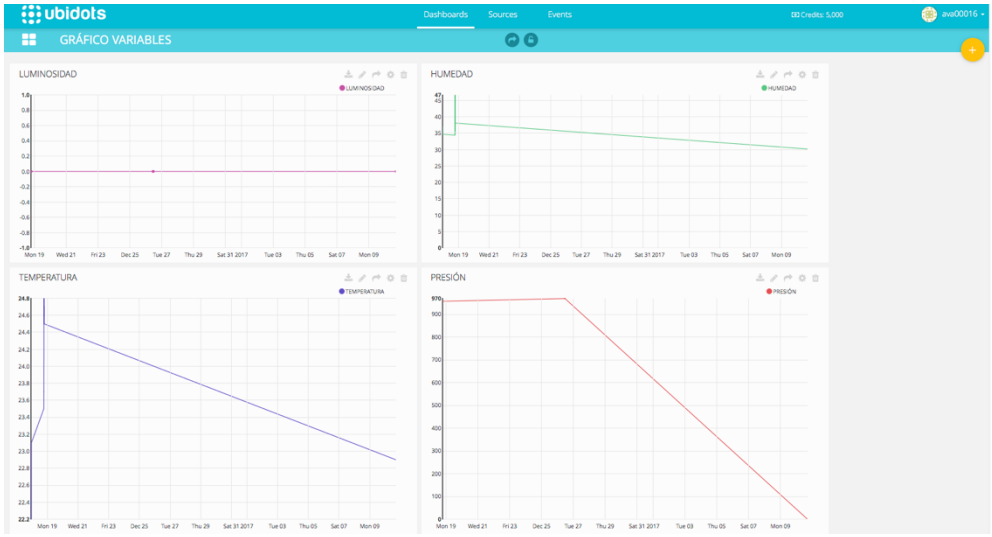

Figura 33. Dashboards Gráfico Variables

Estos gráficos pueden ser configurados según las preferencias del usuario, tal y como aparece en Figura 34, ya que permiten la modificación de los ejes, el tipo de gráfico, el número de datos mostrados y el color del gráfico.

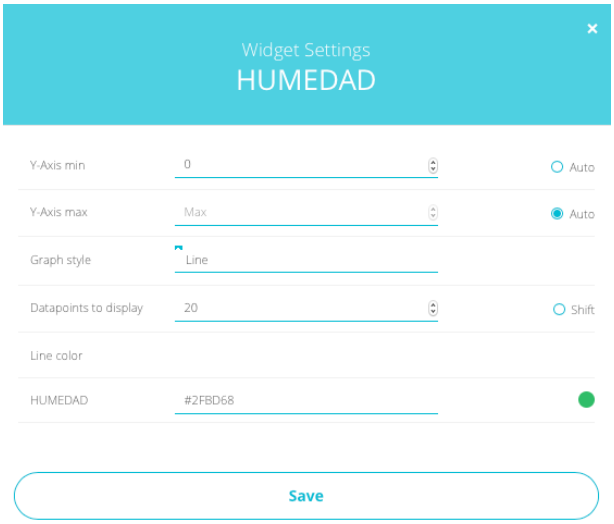

Figura 34. Ejemplo de configuración gráfico

Por último, el escritorio "HISTÓRICO VARIABLES" muestra una serie de tablas donde aparecen los diez últimos datos recibidos, acompañados de la fecha y hora en la que tuvo lugar dicha recepción. De este escritorio, Figura 35, es posible obtener la frecuencia con la que se actualizan cada una de las variables, que es de aproximadamente 20 segundos.

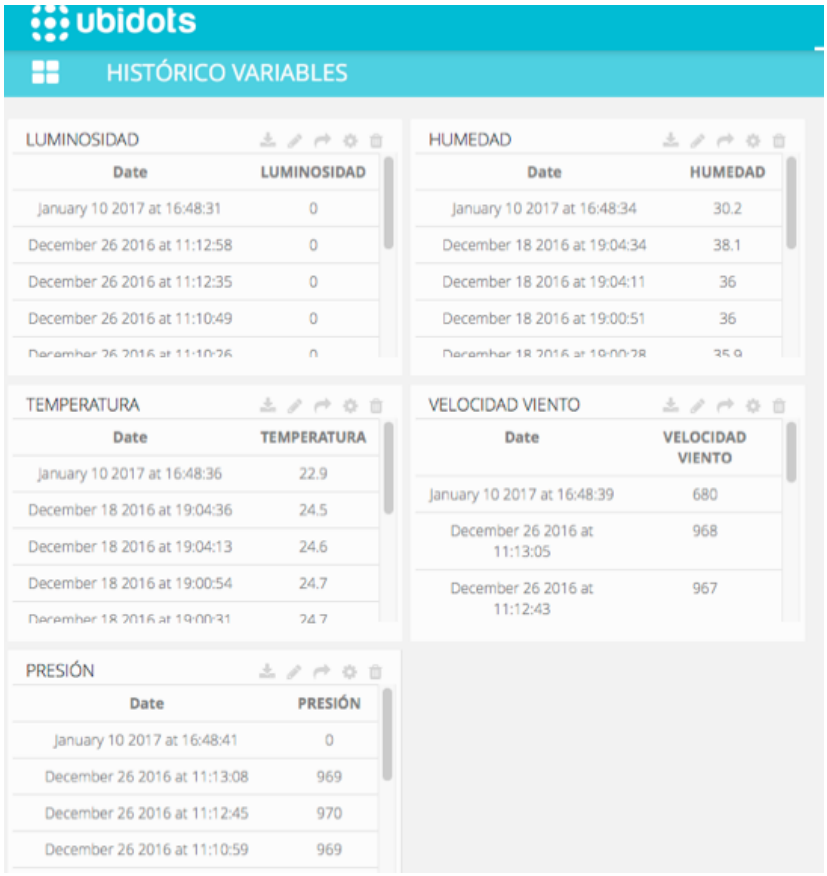

Figura 35. Dashboard Histórico variables

Todo lo descrito anteriormente, al igual que el apartado de *Events* que será desarrollado a continuación, es de acceso privado por lo que sólo aquella persona que posea el usuario y la contraseña correctos podrá acceder a toda la configuración. Sin embargo, es posible crear un acceso público a los *dashboards* para su visualización. Para obtener dichos enlaces únicamente hay que acceder a la opción *Share* o compartir dentro de los escritorios. Para este sistema, los enlaces públicos son los siguientes:

- Estado de la estación:

https://app.ubidots.com/ubi/public/getdashboard/page/nV4cye330BLkRss-9QC42xPYd9k

- Gráfico de las variables:

https://app.ubidots.com/ubi/public/getdashboard/page/dnZ0W5-NxMZJ3CmPELgDOeqrdOs

- Histórico de datos:

https://app.ubidots.com/ubi/public/getdashboard/page/j93O-_u4WdwwM9hLSNZ38yD6DqI

Events

Esta sección de *Ubidots* supone una de las grandes ventajas de esta plataforma ya que permite al usuario crear eventos predeterminados y recibir un aviso en el caso de que alguno de estos sucediese. En esta pantalla, Figura 36, se muestran las características de los eventos ya creados, es decir; la variable implicada, la condición para generar el aviso y el modo en el que se realizará el aviso.

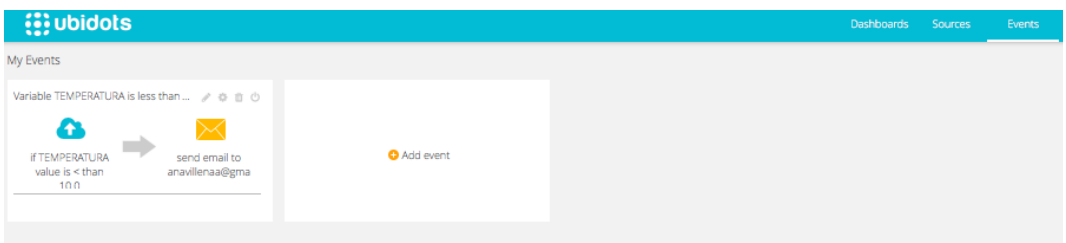

Figura 36. Pantalla inicial Events

Esta herramienta hubiese supuesto una ventaja a la hora de configurar el sistema de alarma de la estación sin embargo, no es posible comparar una variable con otra, como sería el valor máximo permitido enviado desde el microcontrolador; por lo que ésta opción tuvo que ser descartada.

Una vez elegida la variable y la condición del evento es necesario configurar la acción que se llevará a cabo, siendo posible elegir entre enviar un e-mail, un sms, un mensaje de *Telegram,* gestionar un *Webhook,* por ejemplo hacer una publicación en una *web;* o modificar el valor de otra variable.

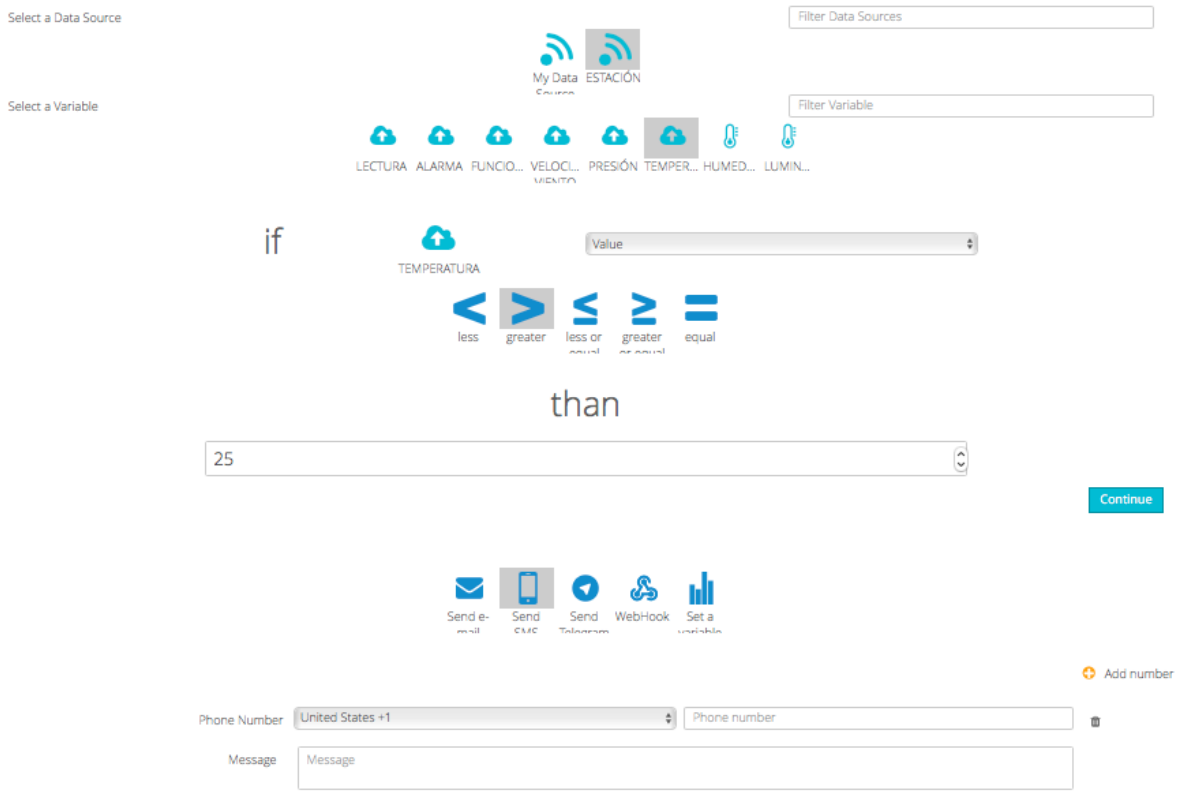

Figura 37. Creación de un evento

En el caso concreto que se ha desarrollado y siguiendo los pasos de la Figura 37, se podrían crear tantos eventos como fuese necesarios; incluso uno para cada una de las variables. De este modo, y teniendo en cuenta que es necesario configurar manualmente desde la plataforma los valores con los que se va a comparar cada variable para generar el aviso, es posible mantener al usuario informado del estado de la estación en todo momento. Si se configurase un evento para recibir un SMS

cuando la temperatura superase los 25 ºC, el resultado sería el mostrado en Figura 38.

Figura 38. SMS recibido de Ubidots

5.2.4 Comunicación con el microcontrolador

Como primer paso de este apartado es necesario conocer el concepto *API REST [9]. API* significa *Application Programming Interface,* traducido a Interfaz de Programación de Aplicaciones; y puede definirse como un conjunto de comandos, funciones y protocolos que permiten crear programas específicos para ciertos sistemas operativos. Su ventaja principal es que no es necesario implementar un código nuevo cada vez que se quiera establecer una comunicación entre varios componentes *software,* sino que es posible utilizar estas funciones prediseñadas.

Por otro lado, *REST* son las siglas de *Representational State Transfer* o Transferencia de Estado Representacional, y hace referencia a un tipo de arquitectura de desarrollo web basado en el protocolo HTTP, es decir, el que utilizan los navegadores para comunicarse con las páginas web. Este permite crear servicios y aplicaciones que puedan ser usadas por cualquier dispositivo que acepte este protocolo, por lo que se ha convertido en la opción más utilizada para crear *API*s.

Por lo tanto, *Ubidots es una API,* gracias a la cual es posible la interacción con la plataforma*,* que ha sido creada utilizando la estructura *REST*. *En* este caso, únicamente será necesario utilizar cuatro métodos del protocolo HTTP para establecer una comunicación con la plataforma:

- *GET*. Utilizado para obtener datos de la plataforma.
- *POST.* Utilizado para enviar información a la plataforma.
- *PUT.* Utilizado para modificar la información de la plataforma.
- *DELETE.* Utilizado para eliminar información presente en la plataforma.

Además de estos comandos es necesario conocer el formato que será utilizado para enviar los datos, siendo los más comunes *XML, eXtensible Markup Language;* y *JSOS, JavaScript Object Notation.* La *API* de *Ubidots* utiliza el segundo debido a su simplicidad por lo que será este el que se explique con más detalle.

JSOS puede definirse en español como Notación de Objetos de JavaScript; y es un formato de intercambio de datos basado en un subconjunto del lenguaje de programación *JavaScript* que es independiente del lenguaje utilizado,es decir, puede ser leído por cualquiera de ellos. La ventaja de este hecho es que puede utilizarse para intercambiar información entre distintas tecnologías. Un ejemplo de estructura de datos con este formato sería el siguiente:

```
{
“value”:1000, “context”: { “estado”: “encendido”}
}
```

El siguiente paso es conocer cómo interactuar con la *API* de *Ubidots[3]* para lo que se necesita un *Token* y el *ID* de la variable a la que quiere enviarse los datos. El *Token* es una clave de identificación que permitirá al dispositvo, en este caso *Arduino,* ser autentificado por *Ubidots;* mientras que el *ID* es un identificador que está asociado a una variable y permite al microcontrolador interactuar con esta.

Ambos valores se obtienen desde la plataforma, el primero puede encontrarse accediendo a *My Profile* y a continuación a *API Keys,* Figura 39. En esta sección puede encontrarse la opción de crear un nuevo *Token,* ya que pueden existir tantos como fuesen necesarios; sin embargo, en el caso de la estación ya aparece uno creado, que coincide con el que está incluido en el código de *Arduino.*

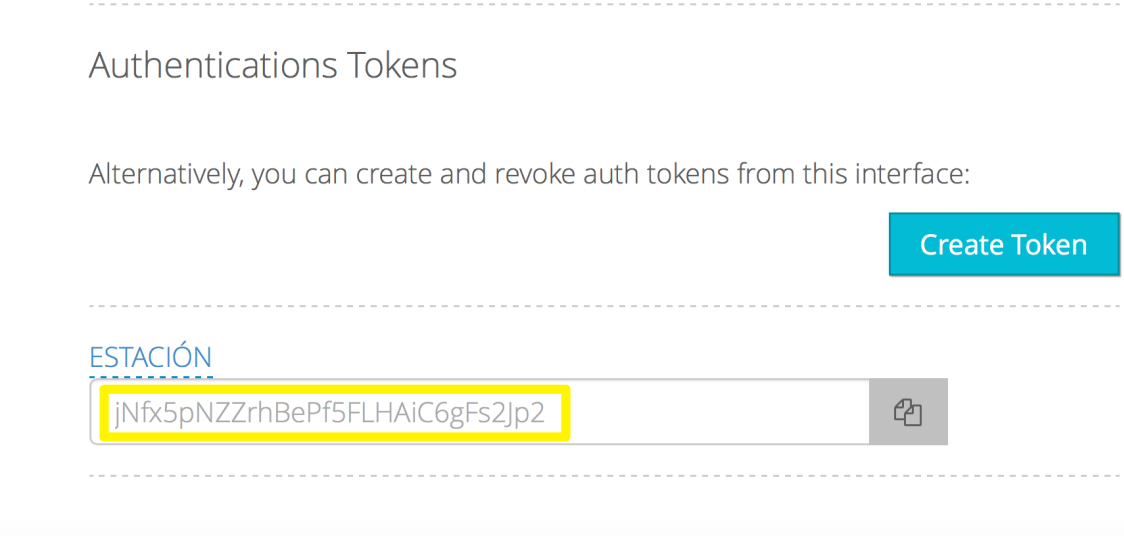

Figura 39. Obtención del Token

En cuanto al *ID,* puede encontrarse en la pantalla *Sources* pulsando en el icono de información de la variable deseada, Figura 40. Al igual que sucedía con el *Token*, los *ID* de cada una de las variables que forman este sistema pueden encontrarse en el código de *Arduino.*

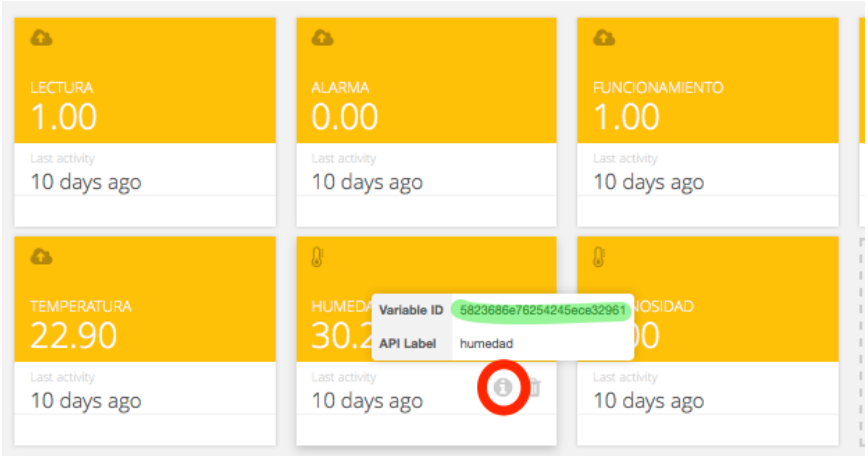

Figura 40. Obtención del número ID

Como fue descrito en el apartado de *Arduino*, ha sido necesario crear un algoritmo que permitiese enviar información a la plataforma ya que la librería que proporciona *Ubidots* no es válida para la placa *Mega.* La estructura enviada por *Arduino* es la siguiente:

POST /api/v1.6/variables/**ID**/values http/1.1

Host: things.ubidots.com

User-Agent: Arduino-ESP8266/1.0

X-Auth-Token: **TOKEN**

Content-Type: application/jsos

Content-Length: **tamaño del mensaje**

{“value”: **valor**}

El objetivo de esta función es enviar la lectura obtenida por los sensores, que aparece como **valor**, a su variable de la plataforma cuyo identificador es **ID.** Para acceder a la plataforma, se utiliza el método *POST* y su dirección *http://things.ubidots.com/api/v1.6*. Por último, es necesario proporcionarle a la plataforma el *Token, el host* o *anfitrión,* el dispositivo utilizado y el tamaño del mensaje que se desea enviar.

5.3 Aplicación Android

Este capítulo está dedicado a explicar en detalle la última unidad que forma el proyecto, la app de *Android.* Gracias a esta, es posible acceder al estado de las variables de la estación mediante un dispositivo móvil y sin necesidad de utilizar conexión a Internet. A su vez, y a diferencia de lo que sucedía con *Ubidots*, ha sido desarrollada para que la comunicación sea bidireccional, es decir, también será posible la configuración de una serie de parámetros del sistema sin utilizar los elementos auxiliares conectados a *Arduino.*

5.3.1 My App Inventor

My App Inventor [4] es un entorno de desarrollo de software que permite diseñar aplicaciones compatibles con el sistema operativo *Android.* Este entorno fue creado por *Google* en 2010 con el objetivo de acercar la creación de aplicaciones personalizadas a personas sin amplios conocimientos sobre programación. Sus principales ventajas son que se trata de un entorno gratuino, ya que solo es necesario tener una cuenta de *Google* para acceder; y su simplicidad, ya que la creación de la aplicación está basada en un sistema de bloques, cada uno de los cuales realiza una función.

Evidentemente, las prestaciones que pueden ofrecer las aplicaciones diseñadas de este modo son limitadas sin embargo, estas cubrían con creces las necesidades del proyecto. Para este trabajo hubiese sido posible utilizar cualquiera de las múltiples aplicaciones ya diseñadas que permiten una comunicación directa con *Arduino* utilizando un módulo *Bluetooth* y además, disponen de múltiples opciones de configuración para ajustar la apariencia de esta a las necesidades de cada caso. Sin embargo, se optó por *My App Inventor* por ser una alternativa sencilla rápida y distinta a lo utilizado hasta este momento que permite crear una aplicación exclusiva para la estación meteorológica.

5.3.2 Descripción de la interface

Una vez que se ha introducido el usuario y contraseña es necesario crear un proyecto nuevo, al que hay que asignarte un nombre. Tras esto nos encontraremos la pantalla principal, Figura 41, que puede ser dividida en varias secciones.

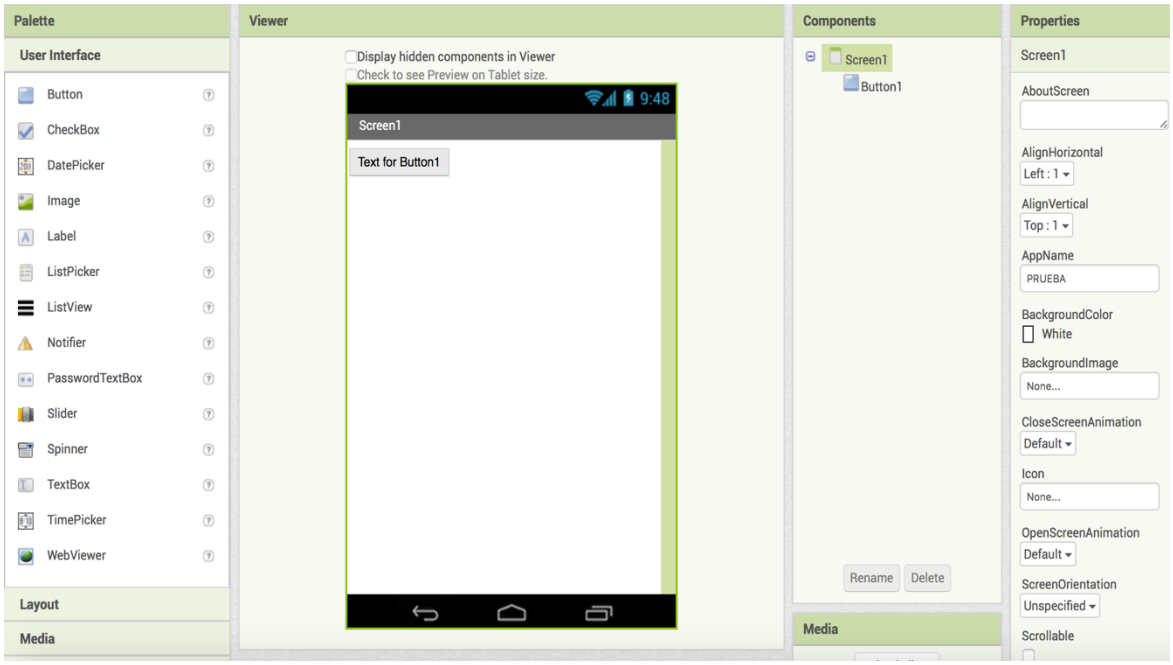

Figura 41. Pantalla Design

En primer lugar, ocupando la parte central, se encuentra el visor donde se muestra la vista actual de la pantalla que se está diseñando. Sobre esta pantalla se añadirán los elementos que darán forma a la aplicación, estos pueden pertenecer a distintas categorías dependiendo de su función. En la parte superior se encuentran la opciones de añadir una nueva pantalla y de cambiar de una pantalla a otra entre las existentes.

Observando la parte izquierda de la Figura 41 puede verse una paleta con una serie de pestañas dentro de las cuales se encuentran todos los elementos que pueden ser añadidos:

- Interfaz del usuario. Elementos que permiten la comunicación con el usuario mostrando datos, como una etiqueta; o recibiendo datos introducidos por este, como un desplegable o un campo de texto.
- *Layout.* Se muestran distintas opciones para configurar la disposición de la pantalla.

- Media. Desde esta sección pueden añadirse la cámara, el grabador de sonidos, el reconocimiento de voz, etc…
- Dibujos y animaciones. Permite añadir a la pantalla todo tipo de imágenes y animaciones colocados sobre un lienzo.
- Sensores. Permite obtener los datos proporcionados por funciones propias de los dispositivos móviles como el podómetro o la ubicación.
- Social. Elementos con los que comunicarse con otros usuarios, como publicar un mensaje en *Twitter* o enviar un *SMS.*
- Almacenamiento. Componentes que permiten guardar información, como la opción de crear un archivo.
- Conectividad. Desde esta sección se incluyen los elementos que permiten la conexión tanto mediante *Bluetooth* como *WiFi.*
- LEGO. Funciones específicas para *LEGO MINDSTORMS.*
- Experimental. Funciones que se encuentran aún en desarrollo.

Todos estos elementos tienen características propias, como son el tamaño, el color, la forma o el texto mostrado; que pueden ser modificadas a elección del usuario desde una sección situada a la derecha de la pantalla principal. En la Figura 42 pueden verse a modo de ejemplo las características de un elemento concreto, el botón, que aparece a la derecha de la pantalla principal.

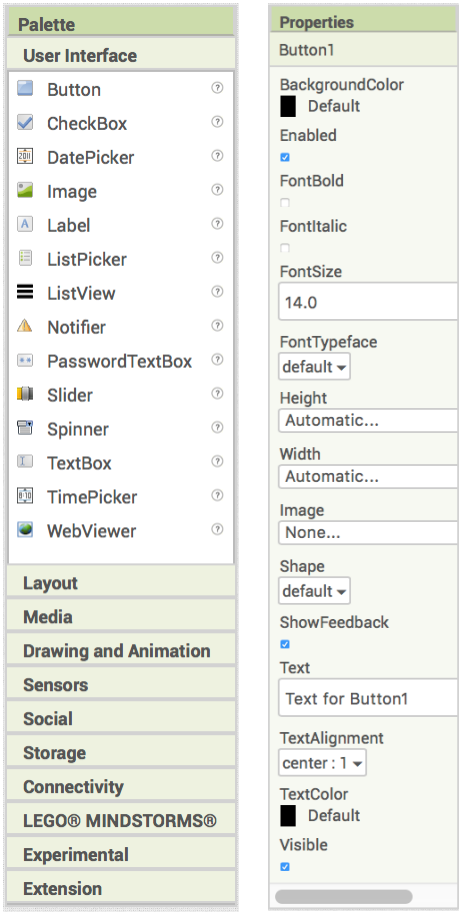

Figura 42. Paleta y propiedades

Tras completar el diseño de la pantalla, añadiendo todos los elementos que se consideren necesarios y modificando sus características para obtener el resultado deseado, el siguiente paso es asignarle a cada uno de estos una función. Esto se lleva a cabo mediante el enlace de bloques *[5]* en la pantalla *Blocks.* Para acceder a ella sólo hay que pulsar en la parte superior derecha de la pantalla inicial.

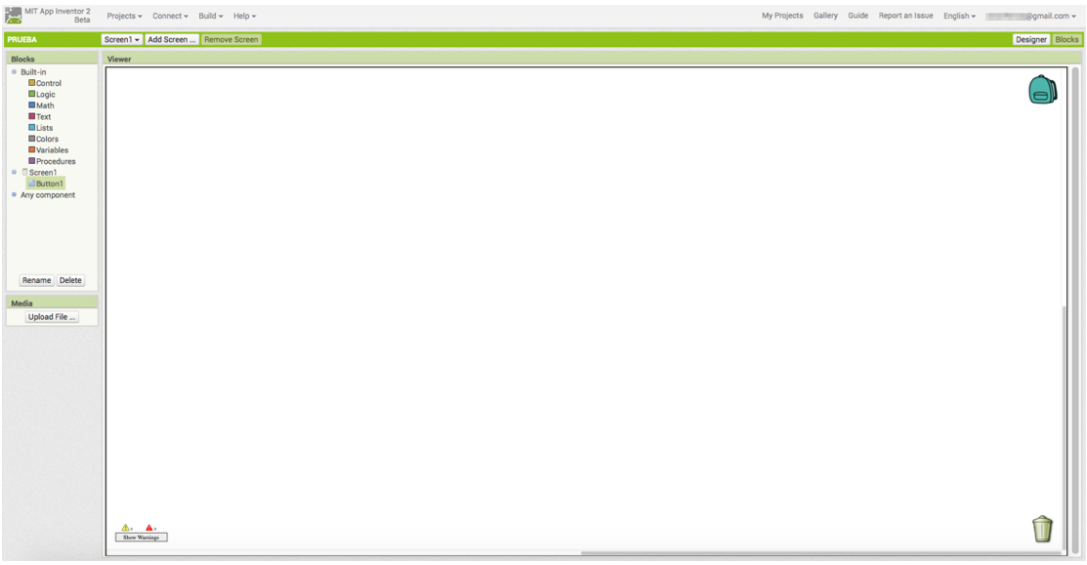

Figura 43. Pantalla Blocks

Como puede observarse en la parte central de la Figura 43 aparece un amplio lienzo donde se irán colocando todos los bloques que den forma al funcionamiento de la aplicación. Sobre este, se encuentran un icono de una mochila y otro de una papelera y permiten almacenar conjuntos de bloques ya creados; o eliminar dichos grupos. Además, en la esquina inferior izquierda aparecen dos tipos de avisos; uno de los cuales puede definirse como una advertencia, y no evita el funciomiento de la aplicación; mientras que el otro si puede definirse como una alerta, ya que se activa cuando un elemento de la programación impide el normal funcionamiento de toda ésta.

En la parte izquierda de la pantalla se encuentran todos los tipos de bloques que inicialmente puden divirse entre los de tipo *Built-in,* o integrados; y los propios de cada elemento añadido a la pantalla. En el primer grupo cuenta con las siguientes opciones:

- *Control*. Bloques utilizados como base para la configuración del funcionamiento del programa, aquí se encontrarán funciones como *if* o *while.*
- *Logic.* Aparecen bloques para trabajar con funciones *OR o AND* además de las variables *true* y *false* que permiten establecer condiciones.
- *Math.* Aquí se encuentran todo tipo de operaciones matemáticas.

- *Text.* Funciones básicas para trabajar con *Strings* o cadenas de caracteres como por ejemplo, concatenar, buscar un carácter, dividir la cadena, etc...
- *List.* Funciones relacionadas con las de listas de datos, es decir, *arrays.* Entre estas se encuentran creación, añadir elementos, buscar un elemento,etc…
- *Colours.* Bloques con distintos colores que, añadiéndolos a otras funciones, permiten cambiar la configuración de los elementos de la pantalla de la aplicación durante su fucionamiento.
- *Variables.* Permiten la creación de variables locales y globales; y tomar y guardar datos en estas.
- *Procedures* o funciones. Incluyen los bloques para la creación de una función y para su llamada desde cualquier punto del programa.

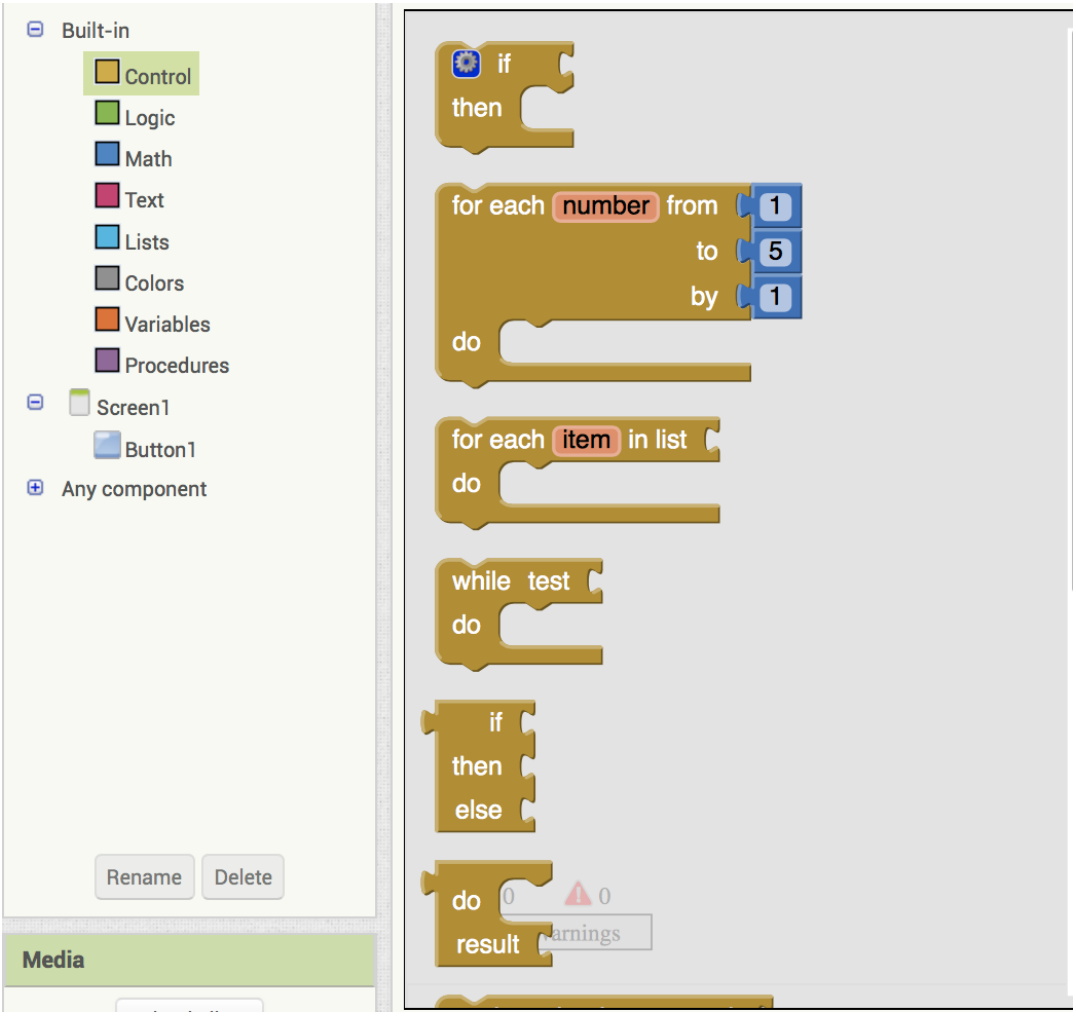

Figura 44. Bloques de control

En cuanto a los bloques asociados a cada elemento, en este caso solo aparece el botón que fue añadido a la pantalla, estos permitirán la gestión de funciones específicas de dicho elemento. En caso concreto del botón permitirá, por ejemplo, definir cómo actuará el programa cuando se pulse dicho botón. En la Figura 45 aparece una pequeña muestra de la programación de la aplicación.

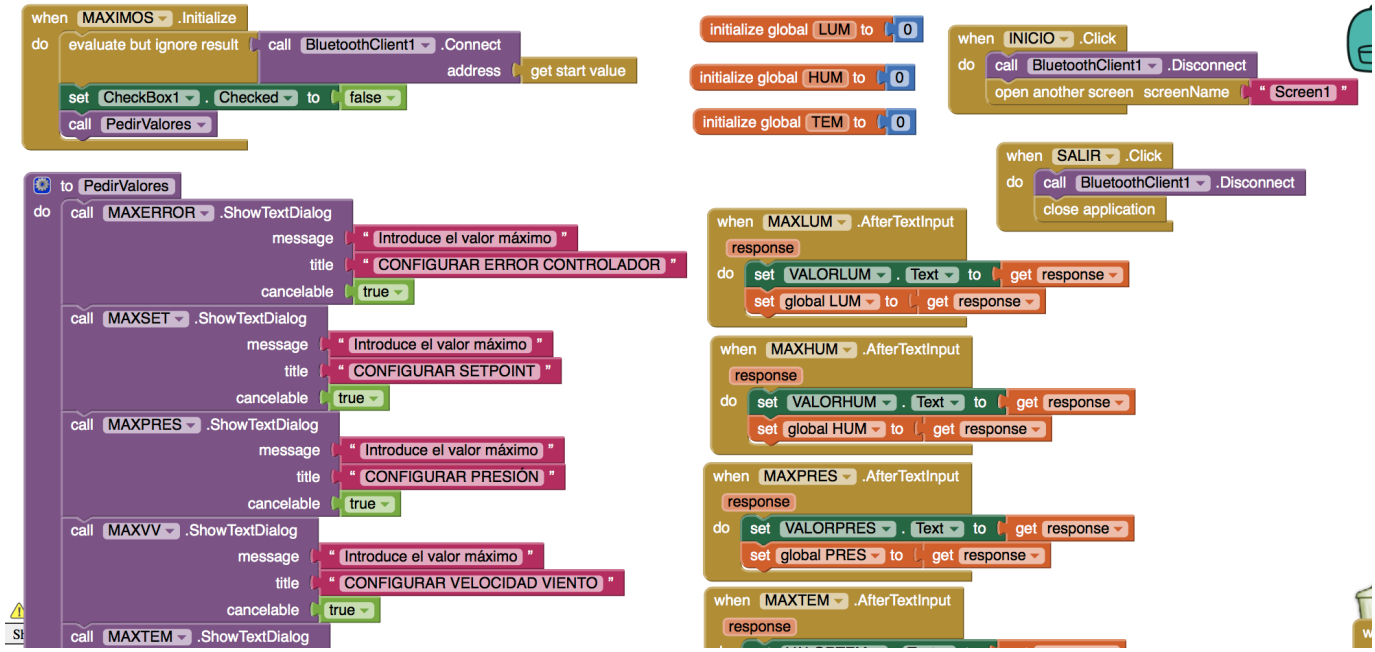

Figura 45. Ejemplo de programación por enlace de bloques

Por último, para concluir con esta sección, es necesario conocer los procedimientos disponibles para instalar la aplicación en un dispositivo móvil. El primero de ellos consiste en escanear un código *QR* desde el dispositivo móvil que, tras abrir una página web en el navegador, comenzará a descargar la aplicación. La otra opción es descargar directamente el archivo *.apk* e instalarlo en el dispositivo.

A lo largo del desarrollo del proyecto se han utilizado los dos métodos: el primero, debido a su mayor rápidez, ha sido empleado durante el periodo de pruebas; y al final de este, al obtener la aplicación deseada, se ha creado el archivo *.apk.*

5.3.3 Funcionamiento

Esta sección se dedica a explicar con detalle la aplicación diseñada para el proyecto. Como será desarrollado a continuación, esta aplicación puede dividirse en

cuatro pantallas, una de las cuales es la principal desde la que se accede al resto; y las otras permiten la ejecución de todas las opciones ofertadas por la aplicación.

5.3.3.1. Pantalla inicial

Esta primera pantalla, Figura 46, cuenta con cinco botones cada uno de los cuales tiene una función muy concreta y sencilla de ejecutar.

Figura 46. Pantalla inicial

En primer lugar, es necesario enlazar el dispositivo móvil con el módulo HC-06 para lo que es preciso pulsar en “CONECTAR BLUETOOTH”. Tras esto, aparecerá un desplegable, Figura 47, con todos los dispositivos *Bluetooth* cercanos con los que es posible conectarse donde habrá que elegir “La Estación”

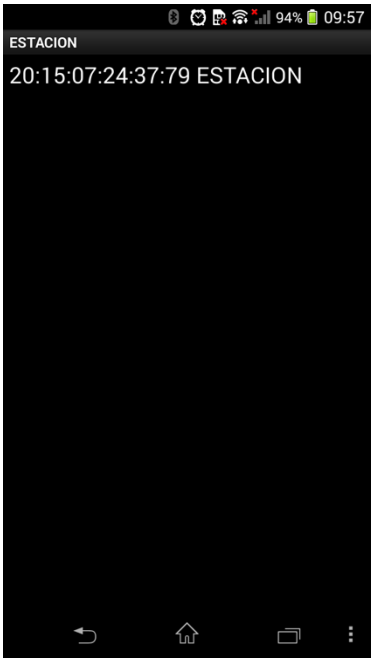

Figura 47. Selector enlace Bluetooth

Una vez que se ha ejecutado este paso obligado existen tres opciones distintas:

- Configurar máximos. Guía al usuario a una pantalla desde la que es posible configurar los valores máximos de las magnitudes atmosféricas, además de aquellas que regulan el controlador.
- Visualización de variables. Muestra una pantalla en la que se monitorizan todas la variables siendo posible conocer su estado sin utilizar el dispositivo físico de la estación.
- Ubidots. Esta pantalla permite acceder a la vista pública de los *Dashboards* de *Ubidots.*

Para lograrlo, únicamente es necesario utilizar el bloque que permite ejecutar una tarea tras pulsar el botón, esta tarea sería abrir otra pantalla.

Finalmente, puede encontrarse un botón de “SALIR” cuya única función es deshabilitar el enlace *Bluetooth* y cerrar la aplicación.

5.3.3.2. Pantalla configuración máximos

En esta pantalla es posible configurar los 7 valores máximos necesarios para el funcionamiento del programa en el microcontrolador. Para lograrlo se han programado

una serie de notificadores en los que se pedirá dicho valor al usuario en el momento en el que se acceda a esta pantalla, tal como muestra la Figura 48.

Figura 48. Petición valores

Tras la recepción de cada uno de los valores, estos se introducen en su respectivo campo de texto, como puede observarse en la Figura 49*;* además, se concatenarán en una cadena de caracteres, colocando un carácter especial entre un valor y el siguiente para delimitarlos, con el objetivo de ser enviados posteriormente al microcontrolador. Para esta concatenación se utiliza el bloque *join* que permite unir una serie de caracteres formando un único mensaje. Para que la recepción de estos datos en el *Arduino* tenga éxito es necesario esperar a que la ejecución del programa se encuentre en el momento oportuno, por lo que es necesario solicitar el envío utilizando el botón destinado a tal fin. Tras esta solicitud la aplicación esperará, utilizando un bloque *while,* hasta que se produzca la recepción desde *Arduino* de una señal predefinida que automáticamente ejecute la llamada a la función diseñada para enviar por *Bluetooth* la cadena que contiene los valores configurados. Será posible conocer el momento exacto en el que se ejecuta dicho envío ya que aparecerá un *tick* en el *checkbox* situado en la parte inferior de la pantalla.

Figura 49. Pantalla de inicio

Finalmente, en la parte inferior se encuentran dos botones. El primero de ellos devuelve al usuario a la pantalla de inicio, permitiéndole así cambiar de pantalla; mientras que el segundo tiene el objetivo de cerrar la aplicación. Estos dos botones también formarán parte de las dos pantallas restantes por lo que, al no existir diferencia alguna entre ellos, no serán mencionados en los apartados siguientes.

Haciendo referencia al ejemplo, sería desde esta pantalla desde que la que el usuario tendría que configurar los valores máximos, en el caso de no querer utilizar el *keypad,* que se definieron en los apartados anteriores. Es necesario recordar que si existiese un error de comunicación entre ambas unidades, el dispositivo móvil y el microcontrolador, el sistema tomaría automáticamente los valores máximos almacenados en la memoria EEPROM.

5.3.3.3. Pantalla visualización variables

Desde esta será posible controlar el estado de la estación ya que muestra los valores de las variables y cuenta con cuatro imágenes que permiten conocer el estado de las alarmas y del controlador.

Tal y como fue descrito en el apartado que hace referencia al algoritmo de *Arduino*, tras obtener las lecturas de los sensores y configurar los *flags* que definen el estado de ciertas variables; estas se concatenan en una cadena de caracteres,

colocando un carácter entre ellas; y dicha cadena se envía a través del HC-06. Esta pantalla será la encargada de gestionar este mensaje para que el usuario pueda conocer el estado de la estación sin ningún inconveniente.

En primer lugar, se ha configurado un reloj que con una frecuencia de 1s hace una llamada a la función que realiza las siguientes funciones:

a. Comprobar, utilizando los bloques *if* y *BytesAvailableToReceive,* si existe algun *byte* disponible para recibir del cliente *Bluetooth.*
b. En caso afirmativo, almacenar toda la cadena de caracteres recibida en una variable previamente creada.
c. Crear una lista de nueve elementos y, utilizando el bloque *spit,* asignar a cada uno de estos elemenos el valor correspondiente a una variable.
d. Asociar cada uno de los elemenos de la lista a uno de los cuadros de texto, para obtener el resultado de la Figura 50.

Figura 50. Pantalla de visualización de variables

Para controlar si el dispositivo se encuentra en funcionamiento, alguna de las alarmas está activa o el motor asociado al controlador ha sido activado; se han incluido tres imágenes como componentes no visibles que únicamente aparecerán en la pantalla en el caso de darse unas determinadas condiciones.

-  Aparecerá en el caso de que se exista un error de lectura, es decir, de que se haya recibido un estado alto en el *flag* correspondiente.
-  Será visible cuando el sistema se encuentre en estado de alerta, es decir, si alguno de los valores máximos ha sido superado.
-  . Esta imagen aparecerá siempre que el motor que actúa como ventilador esté en funcionamiento.

Además de estas imágenes, han sido configurados tres notificadores que mostrarán un mensaje al usuario para informar de su activación. Estos notificadores, mostrados en la Figura 51, únicamente aparecerán en los flancos de subida, es decir, cuando haya una transición de nivel bajo a nivel alto en alguno de los indicadores. Aunque estos no se muestren tantas veces como el microcontrolador envíe un nivel alto, los iconos de las alarmas y del estado del ventilador permanecerán visibles a lo largo de este periodo.

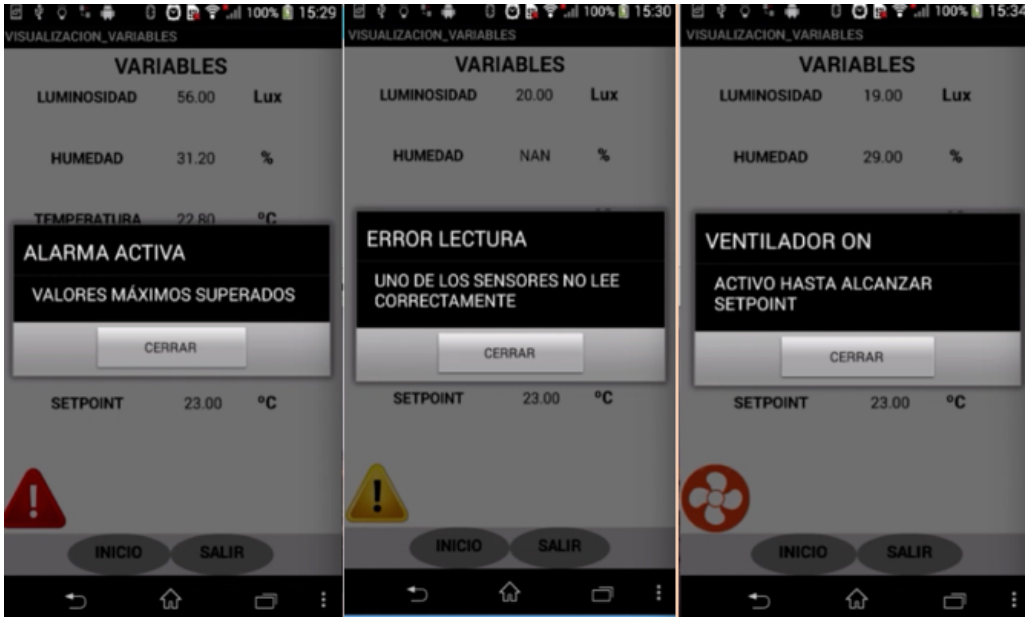

Figura 51. Notificadores

Para lograrlo, se han utilizado variables internas creadas previamente, como un *flag* que se utilizará para diferenciar entre los flancos de subida y los estados altos de los indicadores; bloques *if* y bloques que permiten gestionar las tareas a realizar tanto por las imágenes como por los notificadores. Este procedimiento es el descrito en el Diagrama 11.

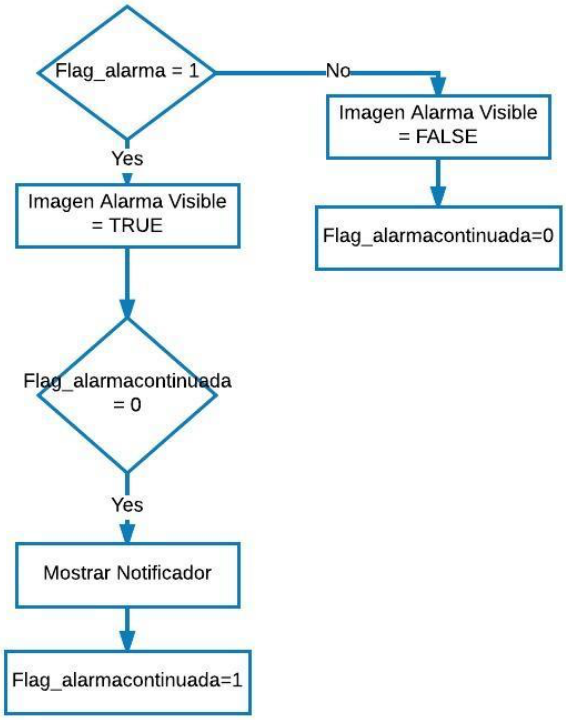

Diagrama 11. Gestion alarmas en la App

5.3.3.4. Pantalla plataforma *Ubidots*

Esta última pantalla cuenta con un visor Web que ha sido configurado para conectarse automáticamente a uno de los tres escritorios, el elegido por el usuario, de la plataforma *Ubidots.*

Para ello, al acceder a esta pantalla aparecerá un notificador que preguntará al usuario qué escritorio desea visualizar. Gracias a un *if encadenado* es posible hacer hacer visible el visor web elegido y ocultar los dos restantes.

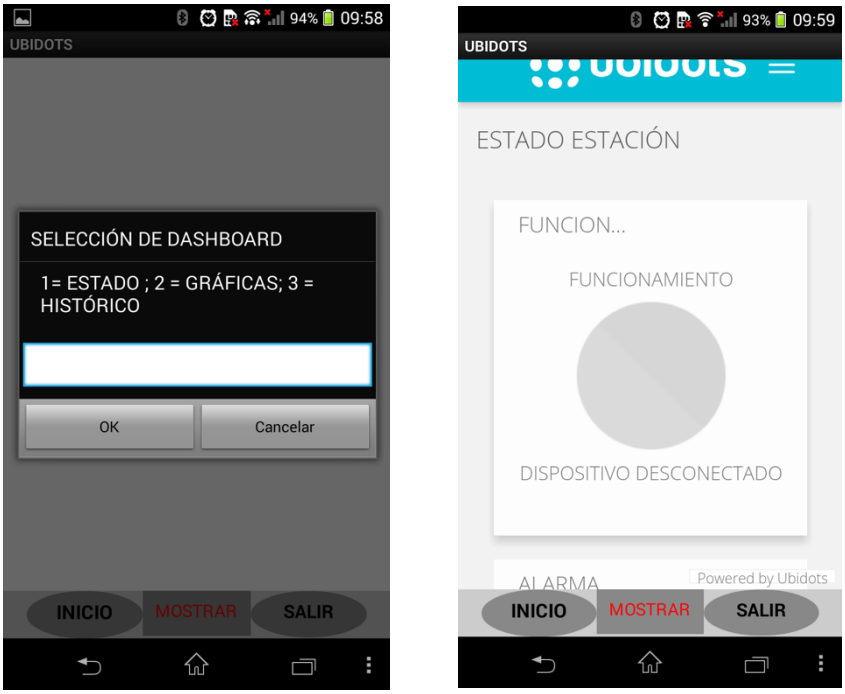

Figura 52. Pantalla visor web

Gracias a este pantalla, Figura 52, es posible acceder, si el dispositivo móvil cuenta con conexión WiFi, al histórico de variables o a los gráficos y así conocer todos los valores almacenados y no únicamente a los instantáneos.

6. PRESUPUESTO

En este capítulo se describen los costes económicos de todos los dispositivos y materiales necesarios para la fabricación de la maqueta. Además, también se dedica una sección a una estimación de las horas necesarias para el desarrollo del trabajo.

6.1 Descomposición de precios

A continuación, serán detallados los precios de cada uno de los elementos descritos en el apartado DISEÑO HARDWARE y de materiales adicionales necesarios para el montaje, como pueden ser soportes o cables.

Código	Resumen	Cantidad	Precio	Total
1.01	**Ud. Arduino Mega**			
	Arduino Mega 2560 Rev 3	1	35,00€	35,00€
1.02	**Ud. Sensor Luminosidad**			
	Adafruit TSL2561	1	15,80€	15,80€
1.03	**Ud. Sensor Humedad y Temp.**			
	Adafruit DHT22.	1	7,02€	7,02€
1.04	**Ud. Sensor Presión atmosférica**			
	Sparkfun BMP180.	1	19,20€	19,20€
1.05	**Ud. Motor**			
	Motor DC de 5V.	1	1,50€	1,50€
1.06	**Ud. Ventilador**			
	Conectado a un motor DC de 5V	1	2,20€	2,20€

Código	Resumen	Cantidad	Precio	Total
1.07	**Ud. Teclado**			
	Teclado matricial 4x4.	1	4,97€	4,97€
1.08	**Ud. Display**			
	KooKie Display LCD 16x2 con módulo LCM 1202	1	6,85€	6,85€
1.09	**Ud. Diodo LED**			
	Diodos de color Rojo (633nm) Verde(570nm) y Amarillo(595nm)	5	0,20€	1,00€
1.10	**Ud. Módulo Bluetooth**			
	SunFounder HC – 06.	1	9,99€	9,99€
1.11	**Ud. Módulo WiFi**			
	Sunfounder ESP8266.	1	7,99€	7,99€
1.12	**Ud. Base**			
	Placa de metacrilato con dimensiones 30x30 cm	1	3,00€	3,00€
1.13	**Ud. Placa cobre**			
	Placa de cobre virgen con dimensiones 20x10 cm para fabricacion de PCB	1	2,50€	2,50€
1.14	**Ud. Cable de puente macho a hembra**			
	Pack 40 cables Arduino de 20 cm	1	1,13€	1,13€
1.15	**Ud. Cable de puente macho a macho**			
	Pack 20 Arduino de 10 cm	1	0,90€	0,90€
1.16	**Ud. Pulsador**	1	0,80€	0,80€

1.17	**Ud. Resistencia 330 Ω**	5	0,15€	0,75€
1.18	**Ud. Resistencia 1 kΩ**	1	0,15€	0,15€
1.19	**Ud. Transistor PNP**	1	0,20€	0,20€
			TOTAL	**120,95 €**

6.2 Estimación horas empleadas

Para hacer el cálculo aproximado de las horas que se han invertido en el desarrollo del trabajo se ha dividido el proceso en cinco fases:

- Diseño hardware. Incluye la selección de todos los elementos que forman el dispositivo.
- Diseño software. Programación y configuración de las tres unidades de la estación, además del estudio de las diversas opciones en el caso de la Nube.
- Montaje. Se refiere a la fabricación de la PCB, del anemómetro y la colocación de todos los elementos sobre la base elegida para obtener la maqueta final.
- Fase de pruebas. Incluye tanto las pruebas que se han realizado con el dispositivo terminado como aquellas realizadas entre fases a lo largo del desarrollo.
- Realización de la memoria.

Código	**Resumen**	**Cantidad**
2.01	**h. Diseño hardware**	15
2.02	**h. Diseño software**	80
2.03	**h. Montaje**	10
2.04	**h. Fase de pruebas**	60
2.05	**h. Realizacion de la memoria**	60
	TOTAL	**225 h.**

6.3 Precio final.

El presupuesto material del proyecto asciente, incluyendo el IVA, a CIENTO VEINTE euros con NOVENTA Y CINCO céntimos (120,95€).

7. CONCLUSIONES

Se ha conseguido llevar a cabo el diseño y la puesta en marcha de una maqueta de estación meteorológica que no solo permite la medida de una serie de variables atmosféricas sino que ha sido adaptada a conceptos tecnólogicos actuales, como el *IoT,* dándole así al usuario la oportunidad de gestionar y controlar el dispositivo sin necesidad de tener acceso físico a este mismo.

No ha sido un proceso sencillo ya que durante su desarrollo han ido apareciendo una serie de problemas que no siempre han sido resueltos con facilidad. Sin duda, las mayores dificultades se han presentado a la hora de establecer las comunicaciones entre las tres unidades vía *Bluetooth* y *Wifi*; y han aparecido como consecuencia de utilizar una placa *Arduino* distinta de la recomendada o de algunos desajustes en el envío o recepción de caracteres a través de los *Puertos Serial* ya que su velocidad teórica no siempre se corresponde con la real.

También es necesario tener en cuenta que, aunque durante todo el desarrollo se ha llevado a cabo una interacción simultánea con las tres unidades, es posible utilizarlas de forma independiente siempre y cuando el microcontrolador se encuentre en funcionamiento. Esto quiere decir que el seguimiento del programa mediante el monitor del *IDE* de Arduino no es obligatorio ya que el microcontrolador puede controlarse con el *keypad* y el display, y es posible alimentar el *Arduino* con una batería sin tener que recurrir a la alimentación por *USB.* Además, evidentemente, el proceso de recepción y almacenamiento de los datos en *Ubidots* es continuo e independiente de que los escritorios estén activos o no.

Como última ventaja de esta estación, destacar que no se trata de un proyecto totalmente terminado, es decir, está abierto a numerosas ampliaciones y mejoras en cualquiera de las tres estaciones que lo forman. Por un lado, tanto *Ubidots* como la *App* cuentan con una serie de prestaciones adicionales que pueden agregarse para cubrir futuras necesidades del usuario. Del mismo modo, otra futura función podría ser la incorporación de una bombilla LED que, gracias a un controlador y al sensor de luminosidad, mantuviese constante la iluminación del espacio en el que se encuentre el dispositivo.

Para finalizar, destacar los beneficios de este trabajo en el ámbito personal, y más concretamente en el académico, ya que gracias a este ha sido posible adquirir una serie de conocimientos que no habían sido presentados en el grado, como puede ser la creación de una aplicación para el sistema *Android;* además de la ampliación otros, entre los que se incluye la programación de microcontroladores de la gama *Arduino.*

Además, este trabajo ha supuesto un primer contacto con el mundo real por dos motivos principales. En primer lugar, por ser una muestra de que no siempre los cálculos teóricos son totalmente válidos y de que en la mayoría de las ocasiones es necesario utilizar el ingenio para encontrar ese pequeño detalle que hace que el resultado teórico y el objetivo real deseado, concuerden. En segundo lugar, por suponer una oportunidad tanto para conocer los avances tecnológicos en los que se está trabajando hoy en día, como el *IoT,* como para familiarizarse con ellos y aportar un pequeño grano de arena para que estos nuevos proyectos sigan desarrollándose y mejorando la vida cotidiana de las personas.

8. REFERENCIAS

[1] Arduino (2017) Online - https://www.arduino.cc/en/Reference/HomePage
[2] Ubidots (2017) Online- https://ubidots.com
[3] Comunicación con el API Ubidots (2017) Online - https://ubidots.com/docs/es/
[4] Documentación My App Inventor (2017) Online - http://appinventor.mit.edu/explore/library
[5] Tutoriales My App Inventor (2017) Online - https://sites.google.com/site/aprendeappinventor/documentacion
[6] Internet de las cosas (I) (2016) Online - http://www.domodesk.com/a-fondo-que-es-el-internet-de-las-cosas
[7] Internet de las cosas (II) (2016) Online - http://www.ticbeat.com/tecnologias/que-es-el-internet-de-las-cosas/
[8] Internet de las cosas (III) (2016) Online - https://www.xataka.com/internet-of-things/las-3-tecnologias-clave-para-el-internet-de-las-cosas
[9] Documentación API REST(2016) Online - https://developer.wordpress.org/rest-api/reference/
[10] Processing (2017) Online - https://processing.org
[11] PID Arduino (2016) Online - http://brettbeauregard.com/blog/wp-content/uploads/2012/07/Gu%C3%ADa-de-uso-PID-para-Arduino.pdf
[12] Arduino Cloud (2017) Online - https://cloud.arduino.cc/cloud
[13] Thingspeak (2017) Online - https://thingspeak.com/
[14] Cloud Computing (2016) Online - https://debitoor.es/glosario/definicion-cloud-computing
[15] Cloud Computing (2014) Online - http://www.ticbeat.com/cloud/que-es-cloud-computing-definicion-concepto-para-neofitos/