UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Gandia

Domotización de un puesto de trabajo usando el microcontrolador ESP32

Trabajo Fin de Grado

Grado en Ingeniería de Sistemas de Telecomunicación, Sonido e Imagen

AUTOR/A: Madariaga Revuelta, Andres

Tutor/a: Pérez Pascual, Mª Asunción

CURSO ACADÉMICO: 2021/2022

RESUMEN

El propósito de este TFG es la domotización de un puesto de trabajo mediante microcontroladores ESP32 y diferentes sensores conectados a estos, en concreto se controlará la entrada mediante un sistema RFID en la que el usuario se identificará mediante una tarjeta y si es el usuario registrado se levantará un servo que permite la entrada, se monitorizarán la temperatura y la humedad del interior del despacho, se podrá controlar la ventilación y se integrarán estos sistemas para poder usarlos también mediante el asistente de Google. Mediante una aplicación Android también se podrán observar ciertos parámetros. La base de la programación será realizada mediante la IDE de Arduino, usando el lenguaje de programación C++. Para la aplicación se hará uso de un lenguaje adaptativo basado en programación de bloques.

PALABRAS CLAVE

Domótica, Programación, Sensor, Android, Microcontrolador.

ABSTRACT

The purpose of this TFG is the domotization of a workstation using ESP32 microcontrollers and different sensors connected to these. The entrance will be controlled by an RFID system in which the user will be identified by a card and if it is a registered user a servo will be raised to allow entry, the temperature and humidity inside the room will be monitored, cooling fan can be controlled too, these systems will be integrated to also use them through the Google Assistant, and through an Android application all of these parameters can also be observed.

As a base point, the programming will be done through the Arduino IDE, using the C++ programming language. An adaptative language based on block programming will be using for the application.

KEYWORDS

Domotics, Programming, Sensor, Android, Microcontroller.

Tabla de contenido

TABLA DE ILUSTRACIONES

1. INTRODUCCIÓN

La sociedad avanza hacia una automatización de procesos y una monitorización constante de cualquier aspecto del día a día. Los datos comienzan a ser moneda de cambio y a adquirir un valor importante, tanto económico como para desarrollar mejoras en las actividades más cotidianas.

El simple hecho de monitorizar la temperatura ambiente que hay en un puesto de trabajo puede ayudar a observar si un aumento o descenso de temperatura fuera de ciertos márgenes conlleva un descenso de la producción del trabajador que se encuentre en el mismo, y tomar medidas en consecuencia, todo mediante la automatización que se encuentra cada día más presente en todos los ámbitos.

En este proyecto se propone la domotización y monitorización de un puesto de trabajo usando el microcontrolador ESP32, el cual brinda un gran abanico de posibilidades debido a su potencia y conectividad. En el puesto de trabajo se contará con un control de acceso mediante identificación por radiofrecuencia conectado a un servomotor que acciona una cerradura, que a su vez podrá activarse mediante instrucción dictada a un altavoz bluetooth con el asistente de Google.

También se contará con un segundo microcontrolador ESP32 que monitorizará la temperatura y humedad interior y almacenará los datos en la plataforma Thingspeak, que actuará como servidor Web, un led rojo que se iluminará al sobrepasar los 25ºC, y un ventilador que se podrá activar mediante un interruptor o bien mediante una instrucción al asistente de Google para hacer descender la temperatura.

Finalmente, el proyecto contará con una aplicación móvil para sistema operativo Android, la cual se desarrollará mediante el entorno APP Inventor 2, con la cual se podrán gestionar los datos tanto de accesos como las mediciones meteorológicas del puesto de trabajo, estos datos se recibirán en el dispositivo móvil conectándose mediante Wifi a dos servidores web diferentes, y de la misma forma se enviarán desde el microcontrolador hacia los mismos.

1.1 OBJETIVOS

El **objetivo primario** de este proyecto es lograr la automatización de ciertos procesos dentro de un puesto de trabajo además de la monitorización de datos meteorológicos y de accesos, con la posibilidad de consultar estos datos desde cualquier lugar mediante la aplicación móvil.

Como **objetivos secundarios**, a continuación, se enumeran los diferentes objetivos que se tratan de conseguir:

1. Diseño y montaje de un sistema funcional y eficiente.
2. Envío de los datos desde el microcontrolador hacia el servidor Web que los almacena.
3. Implementación del uso del asistente de Google para la realización de ciertas acciones.
4. Desarrollo de la aplicación móvil de consulta de datos.
5. Obtención de los datos del servidor Web para mostrarlos en la aplicación.

El **objetivo personal** que persigo realizando este proyecto, es el de profundizar en el conocimiento de un campo de la tecnología como es la domótica que cada día tiene un peso más importante en la sociedad, y usando un microcontrolador de hardware libre como es el ESP32 y el editor de código de Arduino, comprenderlo desde un bajo nivel y adaptarlo al uso con dispositivos móviles o asistentes de voz.

1.2 METODOLOGÍA

Una vez se decidió que este fuera el proyecto final a realizar, se estudió el amplio mercado de sensores y actuadores disponibles y compatibles con el microcontrolador, finalmente se realizó la selección de dispositivos que se comentará detalladamente más adelante dentro de la memoria del proyecto, una vez seleccionados, se realizan diferentes esquemas de conexión para asimilar de una forma más sencilla el montaje del sistema, con esto claro, se procedió a desarrollar el código usando diferentes librerías de la IDE de Arduino para ir realizando pequeños fragmentos de código para su posterior anexión en los dos ficheros generales, finalmente, se llevó a cabo un periodo de pruebas para comprobar el funcionamiento solvente del sistema.

1.3 ETAPAS SEGUIDAS DURANTE EL PROYECTO

1. **Conceptos básicos y bibliografía sobre el proyecto.**

- Búsqueda de información sobre los diferentes dispositivos que se pueden usar en el proyecto, estudio de las posibilidades y elección de la óptima para disponer de posibilidades tanto para la realización del proyecto como para poder mejorarlo en un futuro.

2. **Estudio del entorno y lenguaje de programación elegidos.**

- Se busca y consulta la documentación de la página oficial además de la realización de pequeños tutoriales para lograr profundizar conceptos básicos para el proyecto.

3. **Selección de los sensores a utilizar.**

- Selección de los sensores y actuadores óptimos para crear un ecosistema de trabajo eficiente con el microcontrolador elegido.

4. **Bocetos sobre el montaje.**

- Elaboración de esquemas y planos de conexión entre el microcontrolador ESP32 y los sensores para estudiar las diferentes posibilidades.

5. **Montaje de los circuitos.**

- Montaje del diseño final seleccionado sobre una protoboard para poder realizar pruebas y realizar modificaciones.

6. **Selección de librerías y diseño de funciones a utilizar.**

- Estudio y selección de las librerías a utilizar, así como selección de servidores Web para almacenar y tratar los datos meteorológicos y de lecturas del sensor rfid, diseño de ciertas funciones para enviar datos hacia estos servicios.

7. **Implementación del código y primeras pruebas.**

- Implementación del código necesario para realizar todas las funciones deseadas en el proyecto, en primer lugar, las mediciones meteorológicas y la activación del ventilador mediante un relé que a su vez depende de un botón, movimiento del servomotor tras la lectura de una tarjeta rfid, posteriormente envío de datos a servidores Web y finalmente implementación del uso del asistente de Google para realizar ciertas acciones.

8. **Implementación y diseño de la interfaz de la aplicación móvil.**

- Diseño de la interfaz de la aplicación para dispositivos Android usando el entorno de desarrollo App Inventor 2 e implementación del código usando programación basada en bloques.

9. **Testeo general del sistema.**

- Diferentes pruebas para forzar el sistema a posibles fallos y dar solución a los mismos para así lograr un funcionamiento solvente.

1.4 ESTRUCTURA DE LA MEMORIA

El contenido de la memoria está estructurado de forma que se comienza hablando sobre los materiales usados para la realización del proyecto (punto 2), tanto a nivel de hardware como a nivel de software, a continuación, en el punto 3 se describe el diseño utilizado a través de diferentes diagramas de conexión y su montaje real con imágenes de los circuitos finales.

En el punto 4 se habla sobre el código implementado para la puesta en funcionamiento del proyecto, tanto de librerías existentes como de funciones desarrolladas expresamente para este proyecto.

En el punto 5 se trata el testeo realizado del sistema, así como de diferentes problemas encontrados y la búsqueda de las soluciones pertinentes, mostrando después el funcionamiento final del mismo.

Finalmente, en el punto 6 y último, se desarrollan las conclusiones tras la finalización del proyecto y las posibles mejoras que se podrían aplicar.

2. MATERIALES

A continuación, se procede a describir las características y funcionalidades de los diferentes dispositivos con los que se ha constituido el proyecto, tanto a nivel de hardware como a nivel de software.

2.1 HARDWARE

El Hardware es la parte del proyecto representada por los componentes físicos y tangibles del mismo, que, en este caso, son el microcontrolador y los sensores seleccionados.

2.1.1. Espressif ESP32

El microcontrolador ESP32 es un Soc (System on a Chip) desarrollado por la empresa Espressif el cual se considera uno de los mejores microcontroladores del mercado para su uso en IoT debido a sus capacidades de conexión tanto de Bluetooth como de Wifi a un bajo coste energético y económico, conjunto que lo convierte en una gran elección para proyectos de esta índole más personal de introducción al mundo de la domótica, consiguiendo ciertas características de conectividad extra

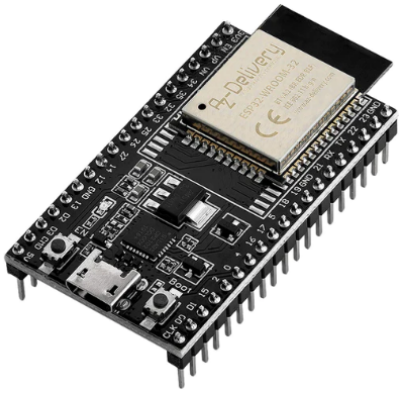


Ilustración 1. Vista de un microcontrolador ESP32

Este microcontrolador se encargará, en cada uno de los dos circuitos diseñados para el proyecto, de procesar la información que emitan los diferentes sensores conectados a él y transmitiendo la necesaria hacia los mismos además de enviarla mediante Wifi a los diferentes servidores Web que se encargarán de almacenarla.

Para lograr el objetivo anterior se hará uso de los 3 módulos principales que lo componen:

· **CPU** o unidad central de procesamiento: compuesta de 2 núcleos de procesamiento que trabajan a una frecuencia variable de entre 80 (MHz) y 240 MHz, ejecutará las instrucciones que se encuentren en la implementación del proyecto.

· **Memoria flash**: con una capacidad de 4 MB, la cual se encargará de mantener las instrucciones implementadas en el código incluso después de quitar la alimentación del microcontrolador, su límite está en unos 100.000 ciclos de escritura.

· **Conjunto de pines de entrada y salida**: estos pines están agrupados según su propósito, algunos son convertidores Digital – Analógico o viceversa, de entrada, de entrada/salida… en el fichero anexo ESP32 se trata en profundidad cada uno de los grupos y su propósito, en este proyecto se usarán estos pines para realizar la interconexión de los sensores con el microcontrolador.

2.1.2. Sensores y módulos

A continuación, se tratarán los diferentes sensores y módulos que se han utilizado para realizar las diferentes funciones del proyecto.

2.1.2.1. Sensor de temperatura y humedad relativa – DHT11

Sensor encargado de la medición de la temperatura y humedad relativa del interior de la sala, en su interior está compuesto de un sensor de humedad de dos electrodos y un sustrato que absorbe el vapor del aire y lo retiene, liberando los iones que aumentan la conductividad entre ambos electrodos, el cambio de resistencia entre ambos electrodos es inversamente proporcional a la humedad relativa.

También cuenta con un termistor NTC cuya resistencia decrece al aumentar la temperatura, finalmente cuenta con un microcontrolador de 8 bits que convierte los datos analógicos en digitales y genera una señal que contiene los valores relativos de humedad, temperatura y un byte de chequeo.

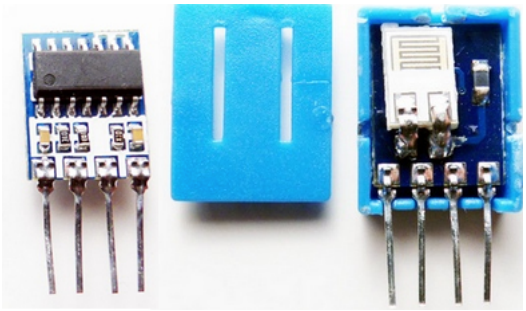


Ilustración 2. Exterior e interior del sensor DHT11

Los diferentes pines de las dos variantes que hay en el mercado de este sensor y el propósito de cada uno de ellos son:

VCC: pin de alimentación de 5V.

GND: pin de masa, su propósito es el de proteger el circuito integrado para evitar perturbaciones en la alimentación que afecten a su funcionamiento.

NC o UN: Not connected o Not used, es un pin sin uso, al aire.

SERIAL DATA: Es el pin encargado de transmitir las mediciones de temperatura y humedad relativa a través de comunicación serial, que es el mismo protocolo que usan algunos de los pines del microcontrolador ESP32.

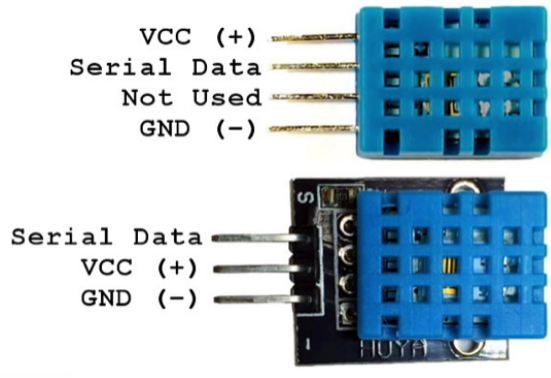


Ilustración 3. Esquema de pines del sensor DHT11

Las especificaciones técnicas de este sensor están disponibles en su datasheet [4].

Este sensor fue elegido para el proyecto debido a que no se necesitaban unas mediciones con un rango de error muy pequeño y su sencilla implementación y funcionamiento.

2.1.2.2. Módulo de relé

Este módulo es necesario para el accionamiento del ventilador que se encargará de rebajar la temperatura del interior del habitáculo, siendo un dispositivo electromagnético, su función es la de un interruptor que es controlado con un circuito eléctrico, el cual, a través de una bobina y un electroimán acciona el contacto que permite el paso de la alimentación hacia el ventilador para ponerlo en funcionamiento.

Exteriormente se puede observar una placa con un relé soldado a un conjunto de resistencias, transistores, diodos y unos terminales con tornillos.

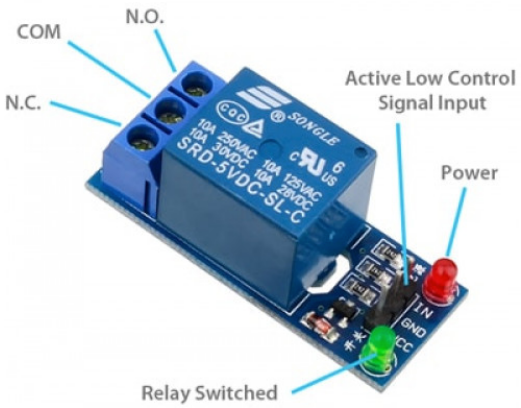


Ilustración 4. Exterior de un módulo de relé

Se puede observar en los terminales de la izquierda que están señalados cada uno de ellos con unas siglas:

N.O o normalmente abierto: cuando el relé se activa, la pletina del común pasa del normalmente cerrado a normalmente abierto, dejando pasar la tensión que se tenga en el común, en este terminal se conectará la alimentación del ventilador.

COM: contacto común, donde se conectará a la alimentación externa del ventilador, que es la que se quiere controlar.

N.C o normalmente cerrado: cuando el relé está en reposo, este terminal está conectado al terminal común, por lo que, cuando el relé no se acciona, tiene la misma tensión que se aplica al terminal común, y cuando se activa el relé, este terminal deja de suministrar tensión.

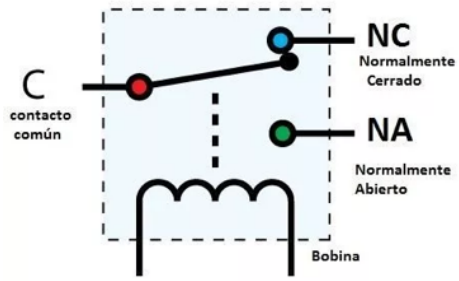


Ilustración 5. Esquema interno de un relé

También será necesario alimentar el relé con 3.3V y masa procedentes del microcontrolador, finalmente se cuenta con un pin **IN** que servirá para activar o desactivar el relé.

Las especificaciones técnicas se pueden consultar en su datasheet [9].

2.1.2.3. Servomotor

Un servomotor es un motor de corriente continua que, gracias a un mecanismo reductor, puede controlar de un modo muy preciso su posición, en este caso de 0º (lo que se considerará cerrado) hasta 180º (que se considerará abierto), mediante un potenciómetro que ejecutará las instrucciones tras la interpretación de los datos recibidos por parte del circuito integrado.

Cuenta con un alto nivel de par, que es el momento de fuerza que se ejerce sobre el eje, en contraposición, la velocidad de movimiento no es tan elevada como en otros motores de corriente continua.

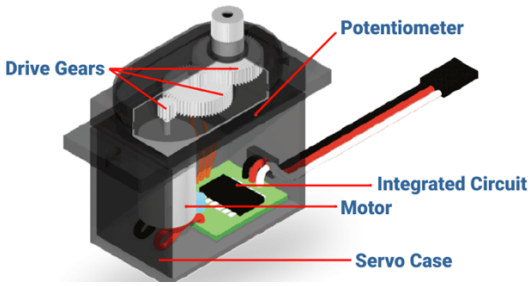


Ilustración 6.Interior de un servomotor

En cuanto a los pines de conexión para este dispositivo, son los siguientes tres:

Control Input o PWM: Pin de control por el cual se recibe la señal pwm que indicará la posición del servomotor, en este caso entre 0º y 180º.

Power o VCC: Pin de alimentación de 5v.

Ground o GND: Pin de masa, usado para protección de alteraciones en la alimentación.

Las especificaciones técnicas se pueden consultar en su datasheet [12].

2.1.2.4. Módulo RFID RC522

Un lector RFID (Identificación por radiofrecuencia) es un dispositivo que se encarga de leer a corta distancia etiquetas (en este caso tarjetas o llaveros) y mediante un sistema de modulación y demodulación obtener el número de serie identificativo de esa etiqueta, formado por 5 valores hexadecimales que se usarán para realizar operaciones en función de lo deseado.

En este caso, este módulo será usado para comprobar si alguna de las etiquetas que se aproximen al lector son alguna de las registradas en el sistema (2) obteniendo su número de serie, y si esto es correcto accionar un servomotor que actuará a modo de cerradura.

Ilustración 7. Módulo de lectura RFID RC522 junto a etiquetas

Para conectar este sensor con el microcontrolador, se usará el protocolo de comunicación SPI (Serial Peripheral Interface), que usa comunicación síncrona, es decir, que tanto el sensor como el microcontrolador comparten una señal de reloj que hace la función de sincronizar la lectura y escritura de instrucciones y mensajes para evitar los errores en la medida de lo posible.

Ambos dispositivos conectados al bus actúan como transmisor y receptor por lo que la comunicación serial es bidireccional.

El uso de este protocolo de comunicación implica que cualquiera de los dos dispositivos puede actuar con rol de esclavo o de maestro.

Un esclavo es un dispositivo que sólo puede conectarse a un único maestro y recibir información de este, mientras que un maestro puede enviar solicitudes de información y conectarse a varios esclavos simultáneamente o de forma individual, siendo el único que puede comenzar la comunicación [26].

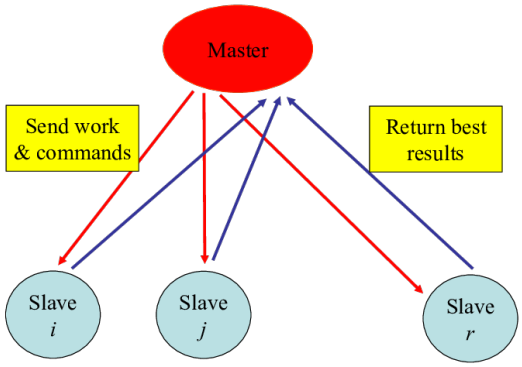


Ilustración 8. Esquema comunicación Maestro-Esclavo

La comunicación, en este caso, entre el microcontrolador ESP32 y el Módulo RC522 se lleva a cabo mediante 4 líneas lógicas:

CLK: Señal de reloj que se encarga de la sincronización de ambos dispositivos.

MOSI: Señal encargada de trasladar los bits desde el maestro hacia el esclavo.

MISO: Señal encargada de trasladar los bits desde el esclavo hacia el maestro.

SS: Señal encargada de seleccionar uno entre los esclavos disponibles y habilitarlo para transmitir información.

Las especificaciones técnicas se pueden consultar su datasheet [13].

2.1.2.5. Otros componentes

2.1.2.5.1. Diodos LED

Un diodo es un componente electrónico formado por la unión de dos semiconductores que únicamente permite el paso de corriente en un sentido, en el cual se comporta como un interruptor cerrado.

Un diodo LED, además de permitir este paso de corriente, emite luz en este mismo sentido, es decir, cuando se le proporciona cierto valor de tensión, lo que también se conoce como que el diodo está polarizado.

En este proyecto, los diodos LED se han usado para encenderse ante ciertas circunstancias o para señalar que se están realizando ciertas acciones, para hacer más visual al usuario final lo que está sucediendo en los dispositivos.

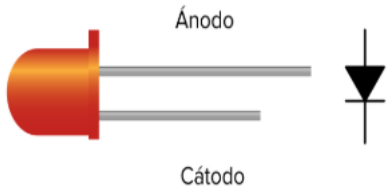


Ilustración 9. Diodo LED

El ánodo o patilla larga será conectada a la alimentación, que en este caso será uno de los pines del microcontrolador mediante una resistencia 330Ω debido a que el voltaje que proporciona (5V) es superior al voltaje de funcionamiento del LED, el cátodo o patilla corta se conectará a masa, se pueden comprobar sus especificaciones en su datasheet [5].

2.1.2.5.2. Ventilador 5V

Se usará un pequeño ventilador de 60x60mm alimentado con 5V de forma externa para hacer descender la temperatura medida por el sensor, se busca simular un sistema de ventilación más grande a pequeña escala para demostrar las posibilidades de este proyecto.

Para el control y la activación de este, se usará el módulo de relé, como se ha comentado con anterioridad, sus especificaciones técnicas se pueden consultar en su datasheet [11].

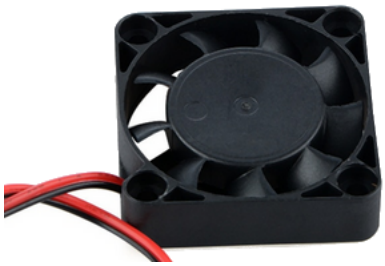


Ilustración 10. Ventilador 5v

2.1.2.5.3. Condensador electrolítico

Para prevenir un problema habitual en los microcontroladores ESP32 al tratar de cargar el código implementado desde el entorno de desarrollo, se coloca un condensador electrolítico de capacidad de 10 µF como el de la imagen, haciendo un puente entre los pines **EN** (Enable) y **GND** (Masa) haciendo así que el bootloader o modo de carga se active automáticamente en el microcontrolador [8].

Ilustración 11. Condensador electrolítico

Esto se debe a que el entorno elegido para desarrollar e implementar el código del proyecto, envía una señal al microcontrolador indicándole que va a cargar nuevo código en la memoria flash de este, por lo que solicita que el pin **EN** se encuentre a un nivel bajo, esto se consigue oprimiendo el botón **BOOT** del microcontrolador aunque a veces en algunas unidades este botón no mantiene el pin en el valor deseado el tiempo suficiente, por lo que, colocando el condensador, se consigue que las señales enviadas desde el microcontrolador hacia el pin **EN** para ponerlo de nuevo a alto nivel se atenúen y así poder cargar el código desde el entorno de desarrollo en el microcontrolador.

2.2 SOFTWARE

El software es la parte del proyecto representada por el entorno de desarrollo y el lenguaje de programación utilizado para implementar en el microcontrolador y los sensores anteriores las funciones deseadas.

2.2.1. Arduino IDE

Arduino IDE es un entorno de programación [1] gratuito y multiplataforma compuesto por un editor de código, un compilador, un debugger o depurador de código y una interfaz gráfica, además de las herramientas necesarias para poder cargar en el microcontrolador deseado el código implementado.

Admite como lenguajes de programación en su entorno C y C++, con el matiz de que usa una reglamentación estructural del código diferente.

Para implementar ciertas funciones de código o el uso de ciertos sensores, se debe hacer uso de librerías externas, que contienen pequeños ficheros de configuración para adaptar el entorno a esas funciones o sensores.

El aspecto de la IDE de Arduino es el que se puede observar en la ilustración de debajo.

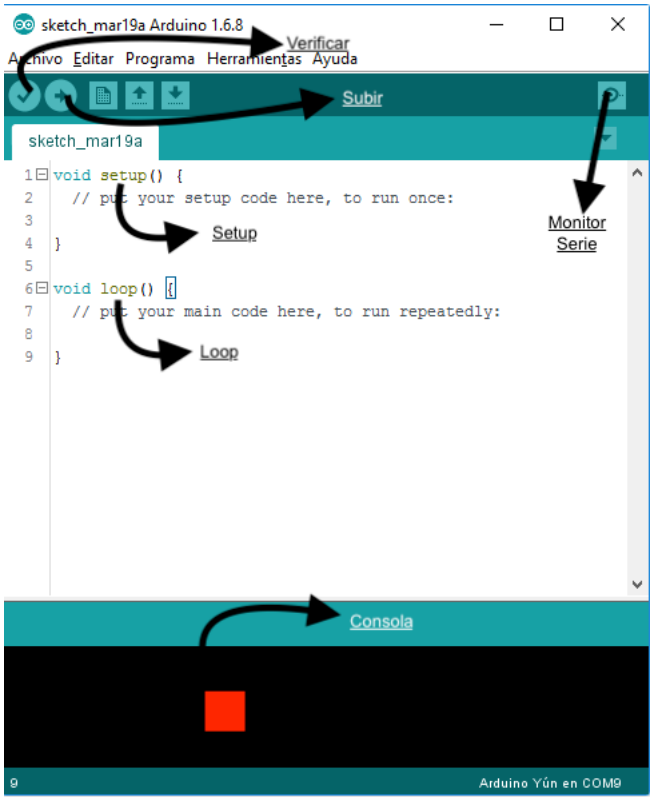


Ilustración 12. Vista de la interfaz de Arduino IDE

Se puede ver que en la ilustración se han señalado ciertas partes de la interfaz, son las siguientes:

Setup: es la parte del código que se ejecuta una única vez.

Loop: es la parte del código que se ejecuta en bucle indefinidamente hasta que se produzca un reseteo o desactivación del microcontrolador.

Monitor serie: es una consola que muestra los resultados de la comunicación serial entre el microcontrolador y el entorno de desarrollo, se le da un uso de comprobación de resultados u observación de pruebas.

Verificar: es el compilador del programa, revisa la sintaxis del código y devuelve los fallos encontrados a través de la consola.

Subir: realiza la función de la verificación (compilar el código) y lo carga en el microcontrolador conectado.

Consola: muestra información de lo que está sucediendo al realizar ciertas acciones en el programa, ya sean errores detectados en la compilación, con su identificación y descripción de este, así como mensajes de carga o porcentajes al cargar el código en el microcontrolador.

2.2.2. App Inventor 2

App Inventor [23] es un entorno de programación desarrollado por el Instituto de Tecnología de Massachussets (MIT), el objetivo con el que fue creado es el de simplificar el proceso de desarrollo de aplicaciones móviles, tanto para Android como para IOS.

Su principal característica es el uso del lenguaje basado en bloques, cambiando así el proceso de desarrollo a un entorno mucho más visual que el tradicional a través del texto.

Ilustración 13. Ejemplo de bloques de APP Inventor

La interfaz del entorno se compone de dos partes principalmente, que son las siguientes:

Diseño gráfico de la aplicación: donde se crearán los diferentes elementos que serán o no visibles en pantalla y que serán utilizados después para integrarlos en el funcionamiento de la aplicación.

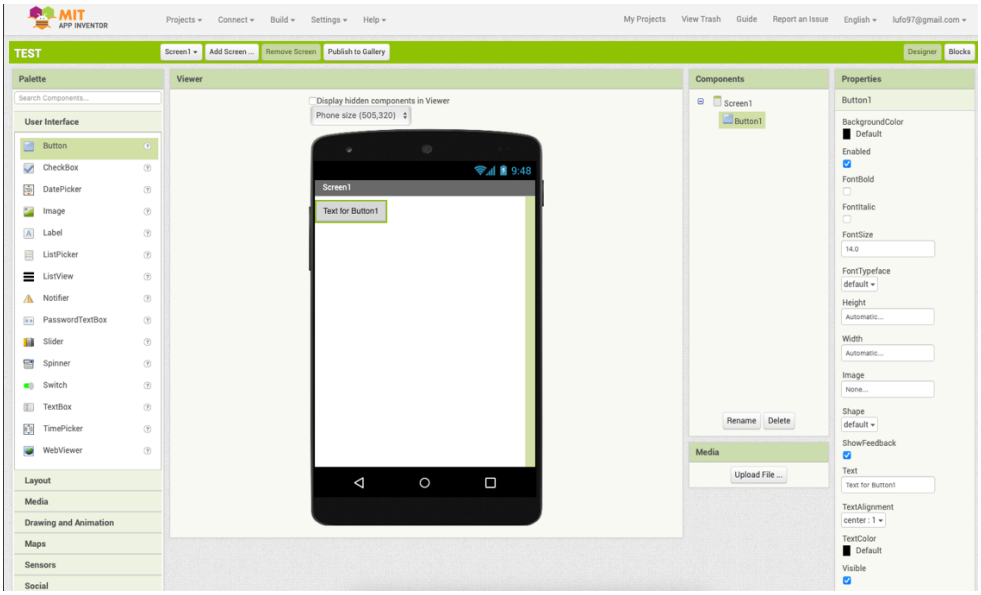


Ilustración 14. Interfaz gráfica de App Inventor 2

Programación de bloques de la aplicación: donde se aplicará la lógica necesaria a los elementos introducidos en el diseño de esta.

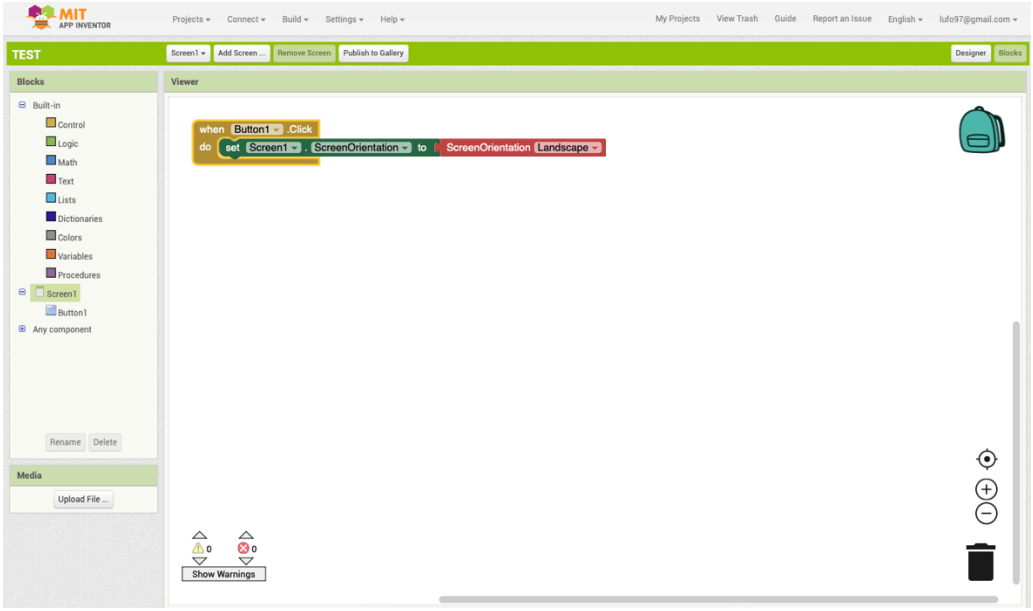


Ilustración 15. Interfaz de programación de bloques de App Inventor 2

Para poder testear la aplicación desarrollada, App Inventor 2 proporciona 3 vías, mediante una aplicación llamada AI Companion, la cual escaneará un código QR y accederá a los archivos mediante un servidor local, mediante un emulador descargable en el PC o pasando los archivos necesarios para la ejecución de la aplicación mediante conexión USB directa al dispositivo.

2.2.3. Notepad ++

Notepad++ es un editor de texto de código libre que da soporte a la programación en la mayoría de los lenguajes existentes.

Su formato es semejante al de un bloc de notas, está escrito en C++ y, al usar la API de Windows, su uso es fluido y consigue mayor compresión en el tamaño de los programas desarrollados.

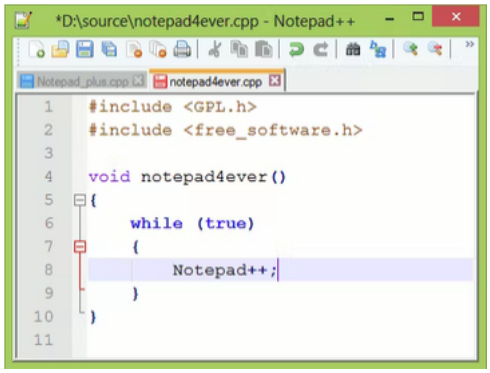


Ilustración 16. Interfaz de Notepad++

En este proyecto ha sido usado para escribir los scripts de PHP que ejecutan ciertas acciones hacia la base de datos alojadas en el servidor Web, así como para escribir los ficheros de configuración .h que serán utilizados en Arduino IDE.

3. DISEÑO Y MONTAJE DE LOS CIRCUITOS

A continuación, se va a tratar el diseño y proceso de montaje de los circuitos que unen los dos microcontroladores con los sensores y módulos mencionados anteriormente, explicando su funcionamiento y protocolo de comunicación [2].

3.1. Circuito de control de acceso

El propósito de este circuito es el de accionar el servomotor cuando se aproxima una etiqueta RFID con una identificación conocida por el sistema, simulando la apertura de la puerta del puesto de trabajo, iluminando también un diodo LED de color verde para indicar al usuario de forma visual que la identificación aproximada está registrada.

Cuando se aproxima una etiqueta RFID desconocida, se iluminará un diodo LED en color rojo indicando al usuario que se trata de una identidad no registrada.

Finalmente, cuando se pronuncian las palabras que activan la "skill" del asistente de Google "Ok Google, activa la puerta", se encenderá un diodo LED de color azul que indicará al usuario que se ha recibido correctamente la instrucción y se accionará el servomotor permitiendo el acceso.

Se puede observar en la ilustración de abajo un esquema de conexión realizado con la aplicación Frizting, de forma ilustrativa y esquemática.

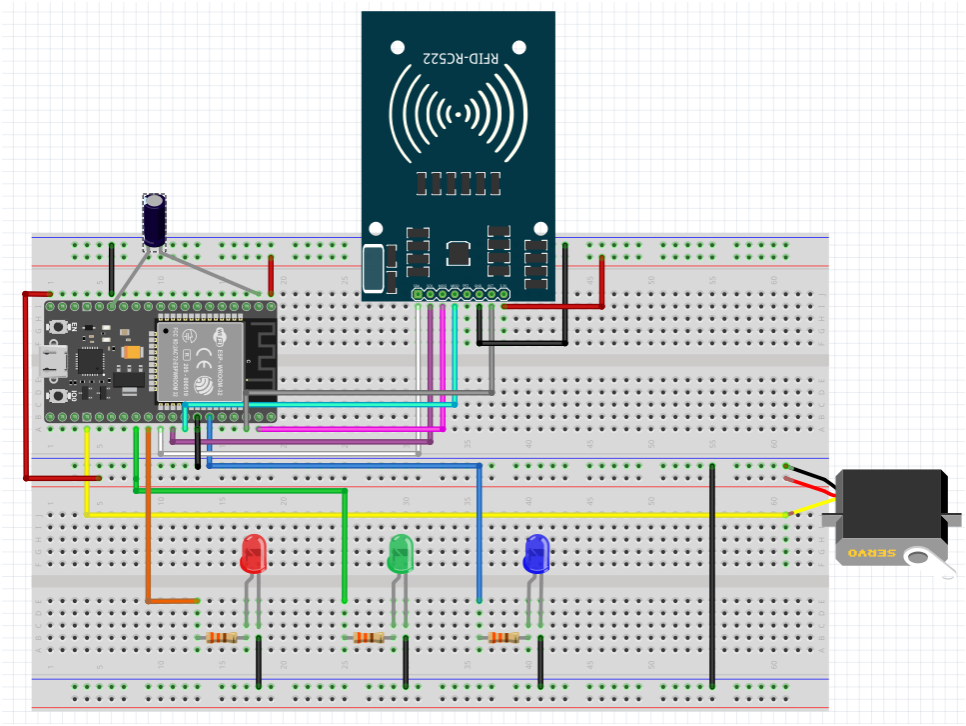


Ilustración 17. Esquema de conexiones del circuito de la entrada del puesto de trabajo

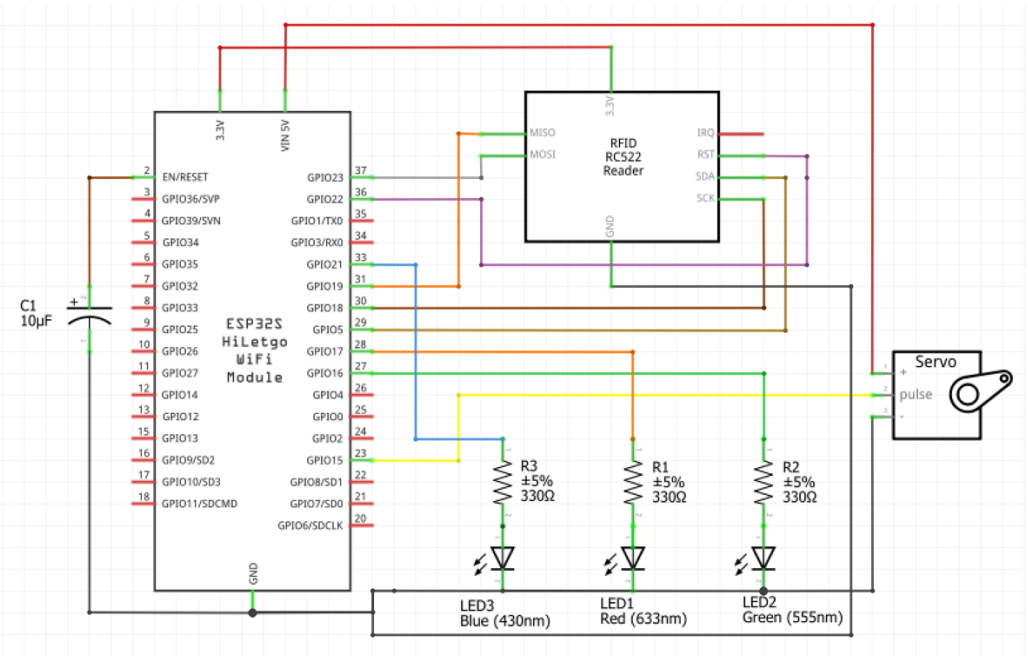


Ilustración 18. Vista esquemática del circuito de control de acceso

Cada uno de los 3 diodos LED tienen el cátodo conectado a masa (GND), y el ánodo conectado a una resistencia de 330Ω que a su vez está conectada a uno de los de pines GPIO y del microcontrolador.

El servomotor tiene conectada la alimentación del microcontrolador, tanto masa como VCC, y con el cable amarillo que se observa se conecta a otro de los pines GPIO (21) del microcontrolador.

Finalmente, el módulo RFID, tiene conectada la alimentación igual que el servomotor, además de hacer uso de 4 pines GPIO (MOSI, MISO, SDA y SCK), para poder establecer la comunicación maestro-esclavo que se describió anteriormente, además de un pin de reset.

3.2. Circuito de monitorización de datos meteorológicos

El propósito de este circuito es el de realizar la monitorización de los datos meteorológicos que hay en el puesto de trabajo en cada momento, para ello se hará uso del sensor DHT11 [3] comentado anteriormente para realizar las mediciones de temperatura y humedad relativa en un intervalo de medición de 2 segundos, además, si la temperatura detectada es superior a 25º, se encenderá un diodo LED rojo indicando que se está trabajando a una temperatura elevada.

También se conectará un ventilador conectado a un módulo de relé para poder accionarlo, será controlado por dos vías, la primera, es la activación o desactivación del ventilador mediante un botón físico.

La segunda vía es la activación o desactivación del ventilador por medio del asistente de Google, en este caso se ha implementado la instrucción clara de “Activa el/Activa el apagado del ventilador” para proporcionar al usuario una vía más cómoda.

En la ilustración de debajo se puede observar el esquema de conexión realizado con la aplicación Frizting, de forma ilustrativa y esquemática.

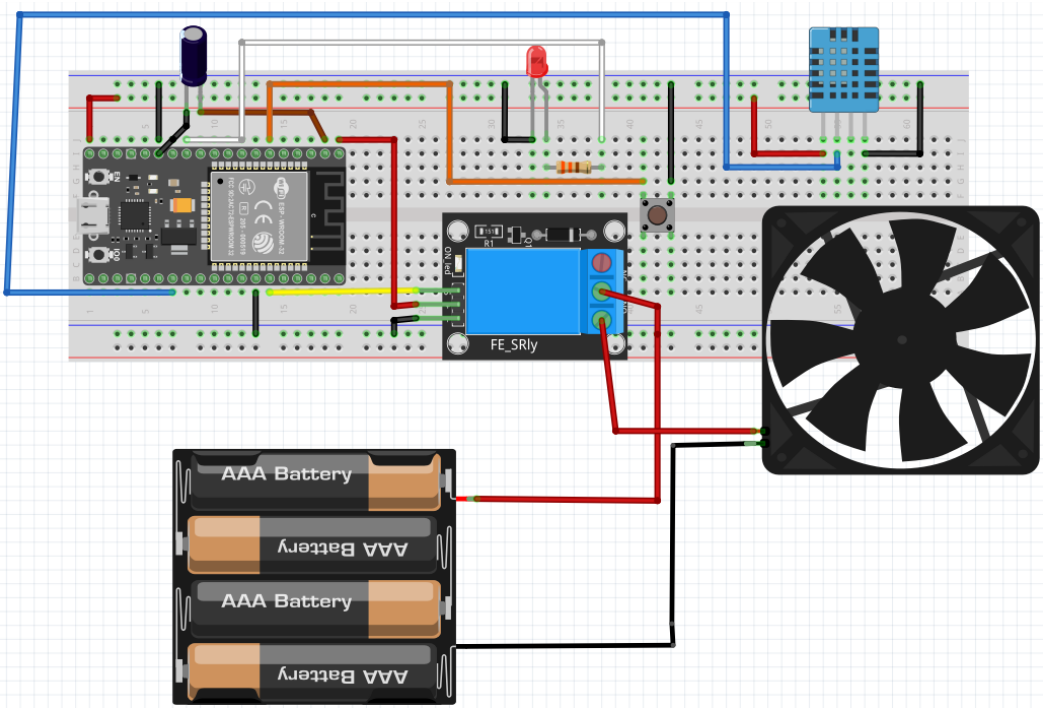


Ilustración 19. Esquema de conexiones del circuito del interior del puesto de trabajo

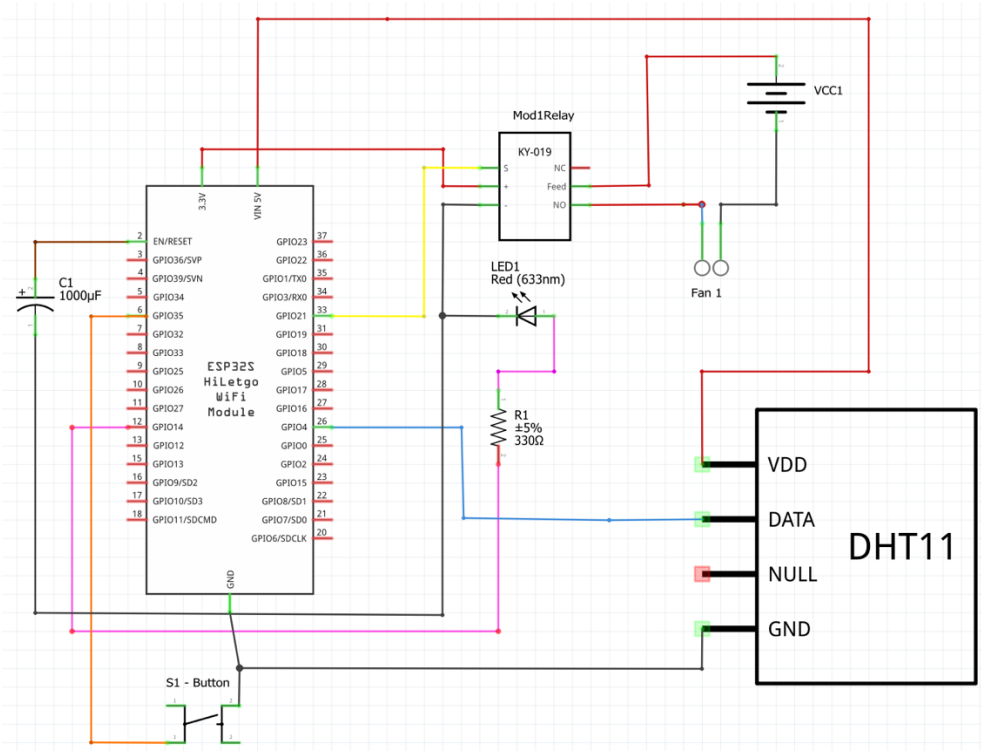


Ilustración 20. Vista esquemática del circuito de toma de datos meteorológicos

Se puede observar que el diodo está conectado como en el circuito anterior, con el cátodo a masa y el ánodo al pin GPIO (14) del microcontrolador.

El módulo del relé está conectado a la alimentación del microcontrolador (VCC y masa) además del pin del que recibirá el dato para realizar el cambio de estado (GPIO 21), la alimentación externa del ventilador está conectada a la entrada COM del relé mientras que la alimentación del propio ventilador está conectada a la entrada NO.

El botón para accionar el ventilador está conectado a dos pines, al pin de masa del circuito y al pin GPIO 35 para poder gestionar ese pin desde el editor de código y así aplicar la lógica necesaria para que accione el relé [10].

4. IMPLEMENTACIÓN

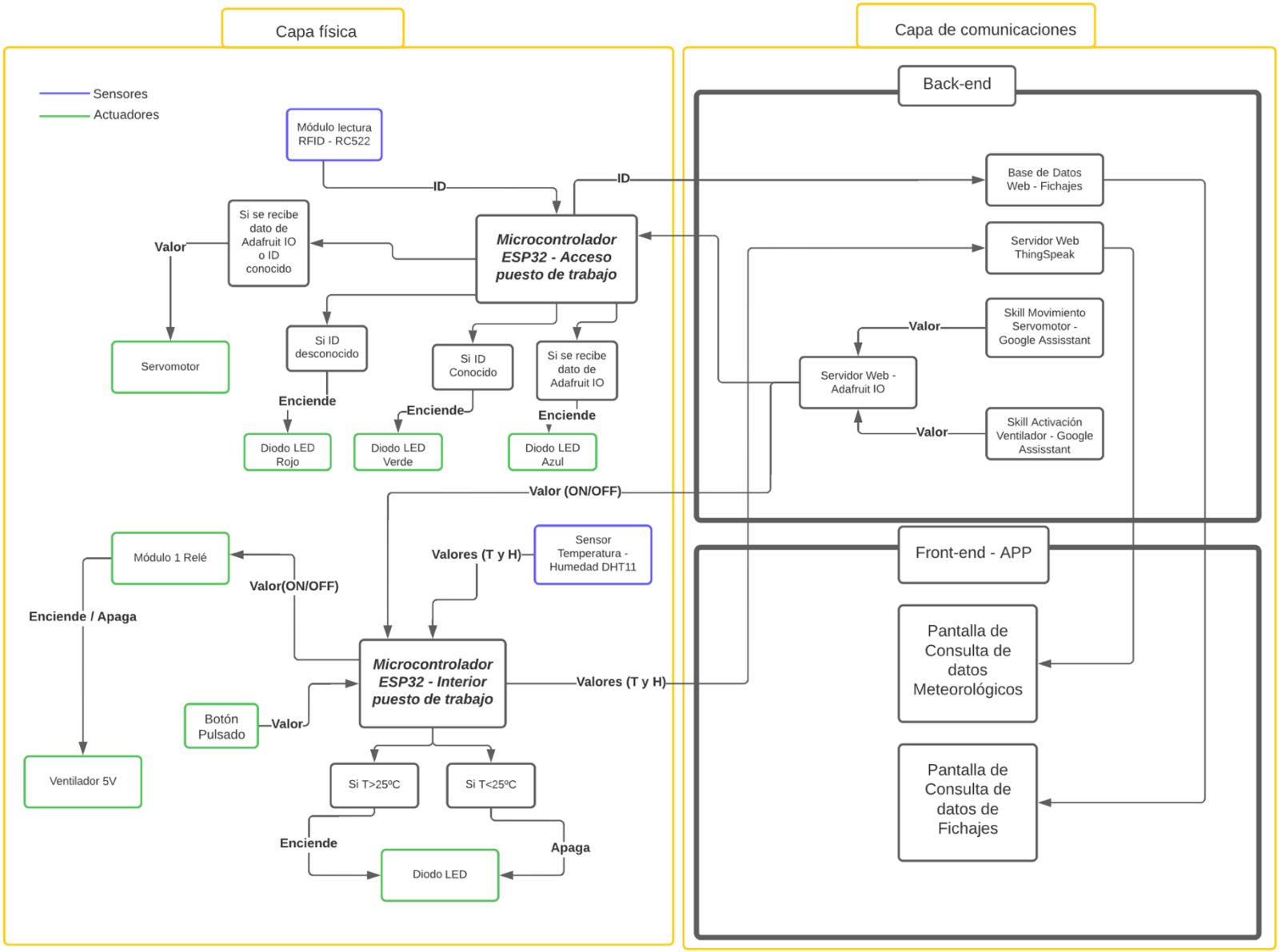
Para describir la implementación del proyecto, en primer lugar, es necesario contar con un diagrama de bloques que contemple todo el sistema separado en las diferentes capas de comunicación, el cual sirve para observar de una manera más general todo el sistema antes de ahondar en cada uno de los apartados por separado.

En el diagrama que se puede observar en la página siguiente, se observan dos bloques de color amarillo que diferencian las dos capas que componen el proyecto, la capa física y la capa de comunicaciones.

La capa física es la que incluye los microcontroladores, los sensores y los actuadores que realizan las funciones implementadas en la lógica del código del proyecto.

La capa de comunicaciones es la que se encarga de intercomunicar tanto todos los elementos de la capa física entre sí como del envío de datos a los servidores web que los alojarán, que se pueden considerar el back-end del proyecto, y la interfaz gráfica de la aplicación de consulta, que se puede considerar el front-end del proyecto.

En los siguientes subapartados se profundizará en cada una de las capas para comprender mejor el funcionamiento de todo el sistema.


Capa física
Capa de comunicaciones
Sensores
Actuadores
Módulo lectura RFID - RC522
ID
ID
Si se recibe dato de Adafruit IO o ID conocido
Valor
Microcontrolador ESP32 - Acceso puesto de trabajo
Servomotor
Si ID desconocido
Si ID Conocido
Si se recibe dato de Adafruit IO
Enciende
Enciende
Enciende
Diodo LED Rojo
Diodo LED Verde
Diodo LED Azul
Back-end
Base de Datos Web - Fichajes
Servidor Web ThingSpeak
Skill Movimiento Servomotor - Google Assisstant
Valor
Servidor Web - Adafruit IO
Valor
Skill Activación Ventilador - Google Assisstant
Valor (ON/OFF)
Módulo 1 Relé
Valor(ON/OFF)
Valores (T y H)
Sensor Temperatura - Humedad DHT11
Enciende / Apaga
Microcontrolador ESP32 - Interior puesto de trabajo
Valores (T y H)
Botón Pulsado
Valor
Front-end - APP
Pantalla de Consulta de datos Meteorológicos
Ventilador 5V
Si T>25ºC
Si T<25ºC
Pantalla de Consulta de datos de Fichajes
Enciende
Apaga
Diodo LED

4.1. Capa física

Como se ha explicado anteriormente, esta capa está compuesta por los microcontroladores, sensores y actuadores, en este apartado se van a tratar las diferentes funciones implementadas y librerías usadas para lograr el completo funcionamiento del sistema.

En primer lugar, se tratarán las librerías usadas para los sensores.

4.1.1. Sensores

El primer sensor por tratar es el sensor DHT11, encargado de medir la temperatura y la humedad relativa en el interior del puesto de trabajo.

Para que realice la función deseada, se ha hecho uso de la librería DHT.h, la cual es la única disponible para este sensor y el uso de microcontroladores mediante la IDE de Arduino, está desarrollada por la empresa Adafruit, y a continuación se puede observar un diagrama que explica su funcionamiento.

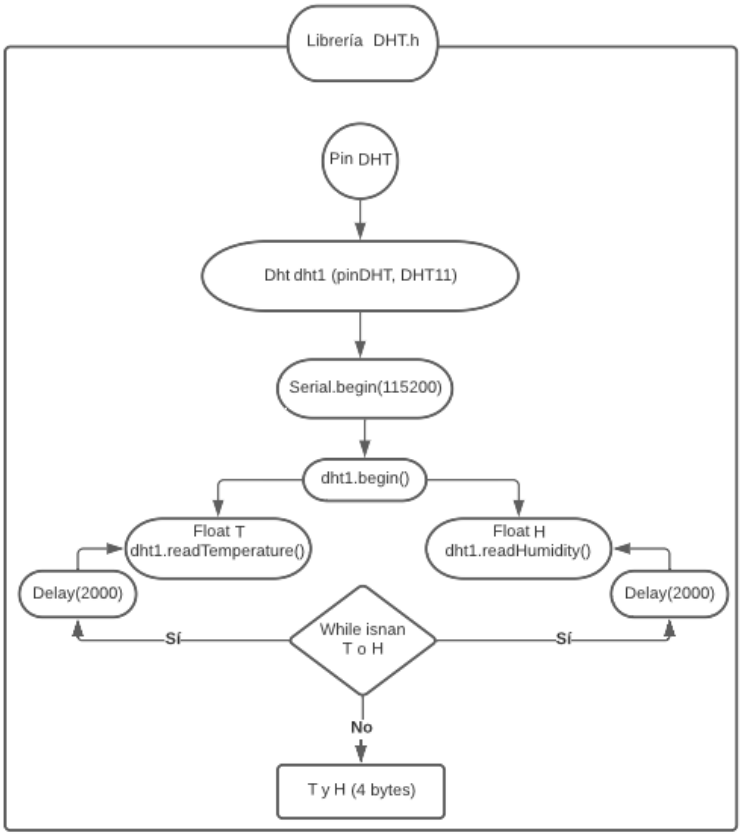


Ilustración 21. Diagrama de la librería DHT.h

El funcionamiento de la librería es el siguiente:

En primer lugar, se debe definir el pin al que está conectado el pin de datos del sensor en el microcontrolador, que será el encargado de enviar los datos de las mediciones al mismo.

A continuación, mediante la función **DHT dht1 (pinDHT, DHT11)**, se inicializa el sensor DHT con el nombre dht1, indicándole el pin antes definido además del tipo del sensor DHT a utilizar, en este caso DHT11.

Se inicia la comunicación serial a una velocidad de 115200 baudios entre el microcontrolador y el monitor serie de la IDE de Arduino del PC al que esté conectado para poder observar datos o comportamientos del sensor, esto se hace mediante la función **Serial.begin(115200)**.

Tras establecer esta comunicación, se inicializa el sensor en sí mediante la función **dht1.begin()**, estableciendo así la comunicación entre este y el microcontrolador.

Para comenzar a realizar dos mediciones, se crean dos objetos tipo float, que es la forma de representar números decimales con un tamaño de 4 bytes, **T** para la temperatura y **H** para la humedad, con cada uno de estos se llama a los métodos **dht1.readTemperature()** y **dht1.readHumidity()**, que comienzan a leer los datos del ambiente y los almacena en estos objetos.

Para comprobar que las medidas son correctas, mediante un bucle while se comprueba que las medidas no sean algo distinto a un número (isnan), si lo fueran, se haría un retardo de 2000ms necesario puesto que es el tiempo que tarda el sensor en realizar una nueva medida, y se volvería al paso anterior, si la medida es correcta, queda almacenada en los objetos **H** y **T** para su explotación.

Otro sensor utilizado en el proyecto es el lector de RFID MF-RC522 utilizado para realizar un control de acceso al puesto de trabajo, este lector leerá las identificaciones RFID que se aproximen al sensor y en función de la lógica implementada realizar determinadas acciones como mover el servomotor.

A continuación, se puede observar el diagrama de la librería MFRC522.h [14], utilizada para la integración del sensor en el proyecto

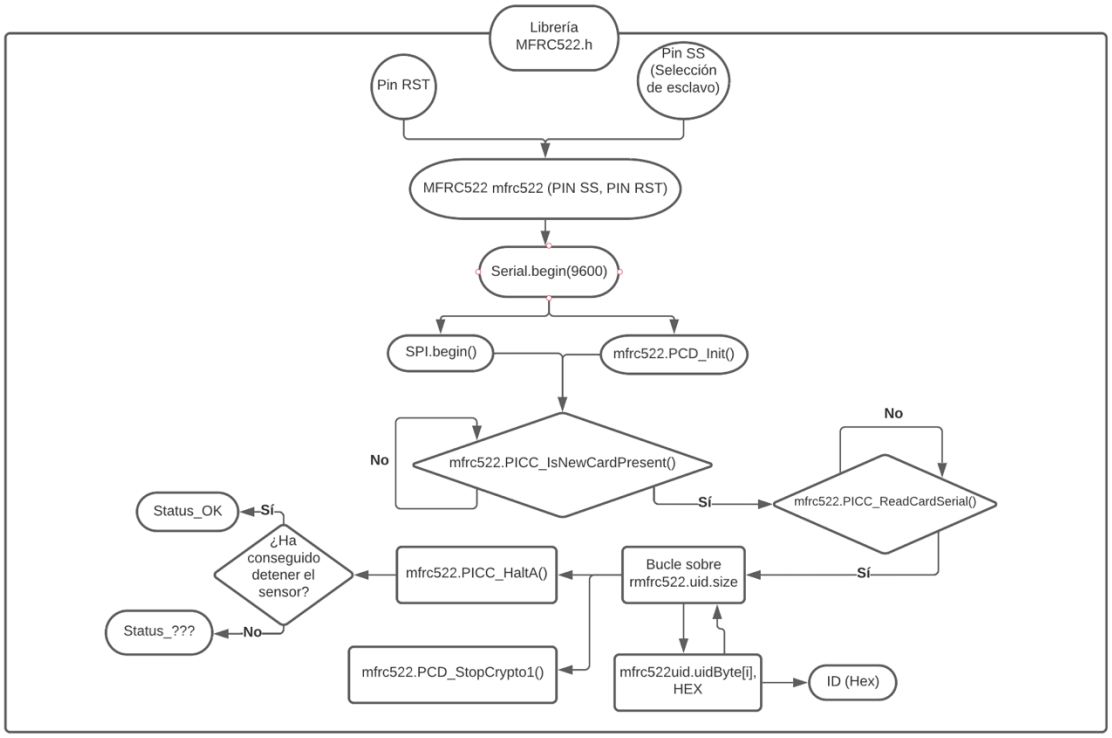


Ilustración 22. Diagrama de la librería MFRC522.h

El funcionamiento de la librería es la siguiente:

En primer lugar, se deben definir dos pines indicando a la librería donde están conectados en el microcontrolador, uno se encargará de realizar el reseteo del sensor al iniciar la comunicación y serial y el otro permitirá establecer el protocolo de comunicación SPI, permitiendo que tanto el microcontrolador como el sensor actúen como emisores y receptores de mensajes.

Una vez definidos los pines, mediante la función **MFRC522 mfrc522 (Pin SS, Pin RST)** se crea un objeto llamado mfrc522 de la clase del sensor, indicando los pines declarados anteriormente para que se pueda establecer la comunicación SPI.

Mediante la función **Serial.begin(9600**) se inicializa la comunicación serial entre el microcontrolador y el PC para monitorizar diferentes parámetros o resultados y poder observarlos en el puerto serie de Arduino IDE.

A continuación, se debe llamar a dos funciones, **SPI.begin()** la cual inicializa la comunicación SPI entre el microcontrolador y el sensor para poder intercomunicar ambos dispositivos, también se debe llamar a la función **mfrc522.PCD_Init()** la cual inicializa el chip del sensor MFRC522 para poder leer los identificadores que se aproximen.

Una vez se han realizado los pasos anteriores, el módulo se encuentra en funcionamiento a la espera de que se aproxime un identificador RFID, por lo que dentro de un bucle se introduce la función **mfrc522.PICC_IsNewCardPresent()** la cual es una función booleana que devolverá true cuando un identificador se haya aproximado al sensor, y cuando esto se produzca se saldrá del bucle.

Cuando el sensor detecta que un identificador RFID se ha aproximado al sensor, llama a la función **mfrc522.PICC_ReadCardSerial()**, la cual es, de nuevo, una función booleana que devuelve true si es capaz de leer el UID (Identificador único) del identificador aproximado, esta función es necesaria puesto que el sensor no es compatible con todos los tipos de identificadores RFID, además, almacenara la identificación leída en el objeto **uid** para su explotación.

Al tener almacenado el identificador en el objeto **uid**, se traduce a hexadecimal para poder realizar las operaciones deseadas con él mediante la clase **mfrc522.uid.uidByte,HEX,** que devolverá el UID del identificador aproximado en hexadecimal.

Finalmente, para pasar el sensor de estado activo a estado de **HALT** (espera), se debe llamar a la función **mfrc522.PICC_HaltA(),** que devolverá un STATUS_OK si se ha realizado con éxito o un STATUS_??? si no se ha realizado.

Es necesario finalizar la comunicación del módulo para poder establecer una nueva comunicación y así poder empezar de nuevo el proceso, esto se hace mediante la función **mfrc522.PCD_StopCrypto1()**.

4.1.2. Actuadores

En este apartado se van a tratar las librerías y funciones implementadas para el uso de los diferentes actuadores en el proyecto.

La única librería utilizada en este caso es la librería **ESP32Servo.h**, que está desarrollada para la utilización de servomotores conectados a los pines del microcontrolador ESP32 y su programación mediante la IDE de Arduino, ya que replica la semántica y funciones de la librería Servo.h, desarrollada para microcontroladores Arduino.

A continuación, se puede observar un diagrama de la librería ESP32Servo.h:

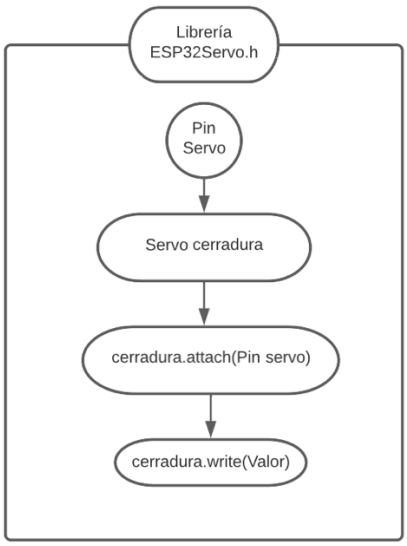


Ilustración 23. Diagrama de la librería ESP32Servo.h

Como se puede observar es una librería sencilla, pero las funciones que usa son las siguientes:

En primer lugar, se declara el pin del microcontrolador al que está conectado el pin de datos del servo, a través del cual recibirá el valor en grados en el que debe de situarse.

Es necesario crear un objeto de la clase servo, en este caso llamado cerradura, haciendo uso de la función **Servo cerradura**, para poder utilizar las demás funciones de la librería.

Con la función **cerradura.attach(Pin Servo)**, se le indica al microcontrolador el pin al que está conectado el servomotor, para así poder enviar los valores por el mismo.

Finalmente, con la función **cerradura.write(Valor)**, se le indica al servomotor el valor en grados al que debe de realizar su movimiento, este valor, dependiendo del tipo de servomotor, irá entre 0 y 180 grados o entre 0 y 360 grados, siendo el primer caso el de este proyecto.

Una vez descritas las librerías utilizadas dentro de la capa física, en el siguiente apartado se describirán las librerías y funciones implementadas en la capa de comunicaciones para desarrollar las funcionalidades del sistema.

4.2. Capa de comunicaciones

En este apartado se van a tratar todas las funciones y librerías utilizadas para conectar con los diferentes servidores web utilizados, así como el desarrollo de la aplicación móvil que servirá para consultar los datos obtenidos de los dos microcontroladores.

En primer lugar, se observa un diagrama que une la librería **Thingspeak.h**, usada para enviar datos al servidor web de Thingspeak [6], y algunas funciones de la librería **DHT.h** [7] que ha sido tratada con anterioridad, dentro de una función llamada **leerdht1()**.

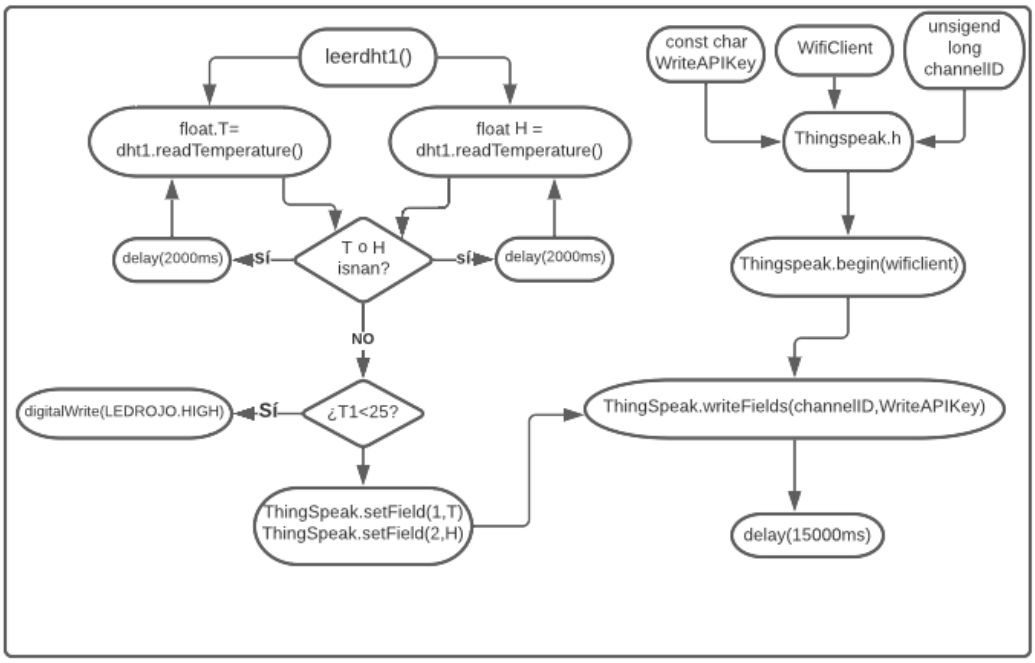


Ilustración 24. Diagrama de la función leerdht1() y la librería Thingspeak.h

Para utilizar la librería Thingspeak.h, es necesario definir una WriteAPIKey que proporcionará la propia web, necesario para poder enviar los datos que capte el sensor al servidor, y el channelID para indicar el canal en el que se almacenarán, además de un cliente WiFi para utilizar la conexión a internet del microcontrolador.

Una vez hecho esto se inicializa la conexión al servidor ThingSpeak del microcontrolador mediante la función **Thingspeak.begin(wificlient)** indicándole el cliente WiFi al que está conectado.

La función leerdht1() tomará las medidas de temperatura y humedad relativa haciendo uso de las funciones comentadas anteriormente de la librería DHT.h, y si el dato de temperatura es superior a 25ºC encenderá el LED rojo mediante la función **digitalWrite(LEDROJO,HIGH)**, es decir, poniendo el pin al que está conectado el LED a nivel alto.

Finalmente, mediante las funciones **Thingspeak.setField(1,T)** y **Thingspeak.setField(2,H),** se indica que el campo 1 del servidor ThingSpeak será la temperatura y el 2 la humedad relativa.

Usando la instrucción **ThingSpeak.writeFields(channelID,WriteAPIKey)**, se escribirán los campos almacenados en el servidor web creado.

Para conseguir la activación del ventilador mediante el uso del asistente de Google, se hace uso de un "applet" de la web IFTTT [19] y de un servidor en Adafruit.io unido a las librerías necesarias para activarlo en función de la instrucción al asistente.

En primer lugar, se crea el "Applet" en la página de IFTTT, conectando la cuenta de Google a la cual estará conectado el asistente que ejecutará la acción e indicándole la frase textual que dirá el usuario tras "Ok,Google activa…", y el dato que deberá enviar al servidor de Adafruit.io [15], también vinculado a una cuenta creada, al producirse esta instrucción [20].

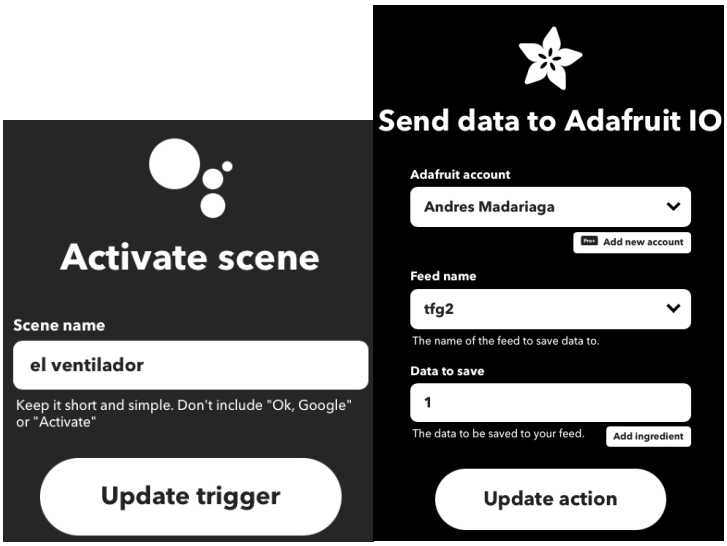


Ilustración 25. Creación de la instrucción de Google y Adafruit.io en la web de IFTTT

Una vez hecho esto, se comprueba en el Dashboard del servidor web de Adafruit.io que se recibe el dato enviado desde la web de IFTTT.

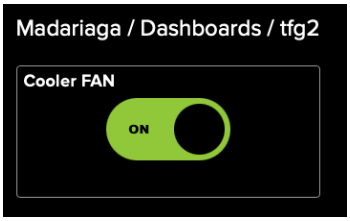


Ilustración 26. Dashboard del servidor creado en Adafruit.io

Conseguida la conexión entre servidores, se conectan los microcontroladores al servidor web de Adafruit.io, ya que es este el que recibe los datos que desencadenan de la instrucción realizada por el usuario, haciendo uso de la librería AdafruitIO_WiFi.h [17] que facilita el propio servicio de Adafruit, que se explica en el siguiente diagrama.

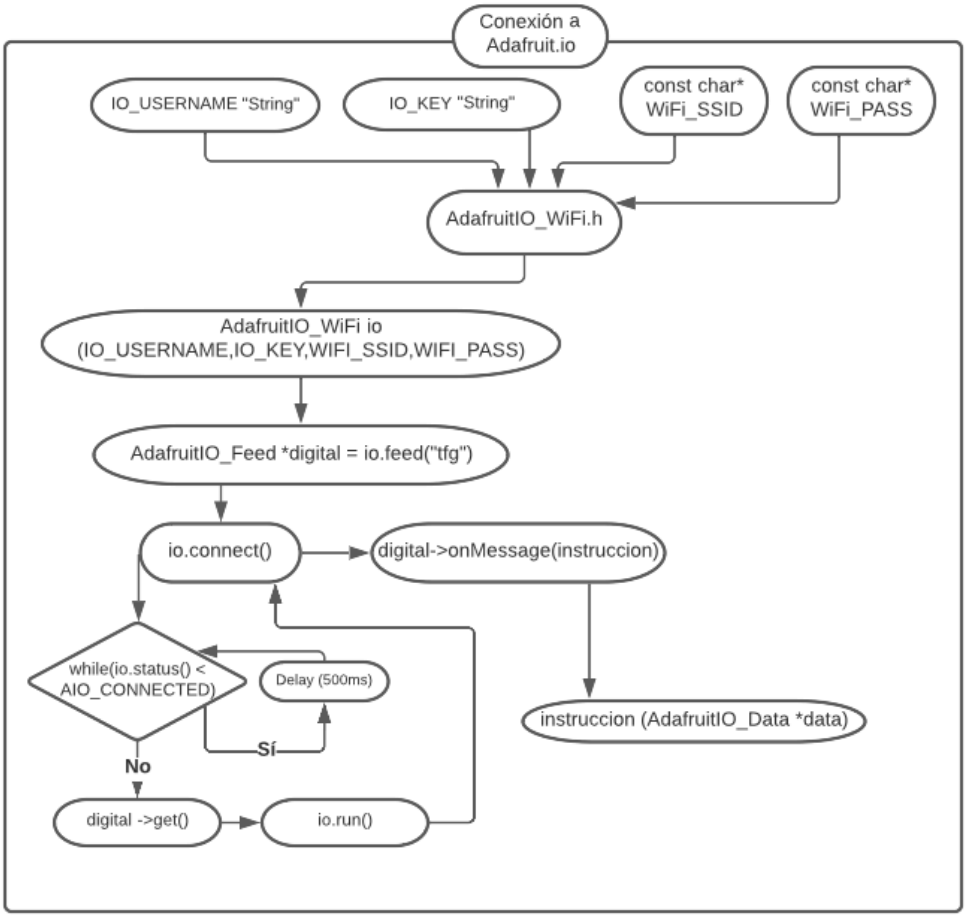


Ilustración 27. Diagrama de la librería AdafruitIO_WiFi.h

Esta librería necesita para funcionar la definición de 4 constantes, **IO_USERNAME**, que es un string con el nombre de usuario de la cuenta de Adafruit_IO a la que se quiere realizar la conexión, **IO_KEY**, que incluye la clave única de acceso al servidor, y el **SSID** y la **contraseña** de la red WiFi que usa el microcontrolador para realizar la conexión.

Una vez definidos estos elementos, se usan para la conexión “io” al servidor web de Adafruit.io mediante la instrucción **AdafruitIO_WiFi io (IO_USERNAME,IO_KEY,WIFI_SSID,WIFI_PASS),** que conecta el microcontrolador con el servidor creado por el usuario.

A continuación, es necesario conectarse al “feed” del cual quieren leerse los datos, que son los contenedores donde se crearán los contenedores de datos deseados para cada caso, esto se consigue haciendo uso de la función **AdafruitIO_Feed *digital =io.feed(“tfg”),** con lo cual creamos un “feed” dentro del código llamado “digital”, que estará conectado con el ya disponible en el servidor web llamado “tfg”.

Con los dos “feed” ya conectados entre sí, se debe inicializar la conexión al servidor web creada anteriormente mediante la función **io.connect().**

Esta conexión entra en un bucle while que espera a que el estado devuelto por la función **io.status()** sea AIO_CONNECTED, lo cual indicará que la conexión al servidor ha sido satisfactoria, si no se obtiene ese estado esperará 500ms para hacer una nueva comprobación.

Cuando la conexión entre el microcontrolador y el servidor web ha sido satisfactoria, mediante la instrucción **digital ->get()**, se obtiene el dato disponible en el “feed” seleccionado con anterioridad en el servidor web, el cual será un bit con valor 1 o 0.

Completados todos los pasos anteriores, en la parte del código que se repite indefinidamente, conocida como “Loop”, se usa la función **io.run()** la cual vuelve al punto en el cual la conexión aún no está realizada para que compruebe que sigue funcionando correctamente, y mediante la función **digital->onMessage(instruccion),** cada vez que se reciba un dato desde el servidor web, se llamará a la función llamada instrucción, cuyo diagrama se puede observar debajo.

Ilustración 28. Diagrama de la función Instruccion

Esta función ha sido utilizada en el código de ambos circuitos, en primer lugar, se tratará el de monitorización de datos meteorológicos.

En este caso, la función recibe como parámetro de entrada el dato que ha sido enviado por el "feed" del servidor web en el que está conectado el microcontrolador, usando la función **AdafruitIO_Data *data.**

Una vez almacenado ese dato en el objeto "data", mediante la función **toPinLevel()** se convierte ese dato (1 o 0) en un nivel de pin que pueda interpretar el microcontrolador (1=HIGH, 0=LOW), si el dato recibido es 1, o HIGH para el microcontrolador, quiere decir que se ha usado la instrucción que enciende el ventilador, puesto que así ha sido configurado en el applet creado en la web de IFTT anteriormente.

Se hace uso de la función propia de Arduino **digitalWrite(PINRELE, HIGH)**, la cual establece el valor de un pin digital en concreto al estado (HIGH o LOW) que se le indique en la llamada a dicha función, en este caso, se establece el valor de PINRELE, que es el pin digital al que está conectado el pin de datos del módulo de relé, que, al cambiar su estado a HIGH, acciona el electroimán de este, encendiendo así el ventilador al proporcionarle alimentación.

En el caso contrario, es decir, que el dato recibido desde el servidor sea un 0, ocurrirá lo contrario, estableciendo el estado del pin digital al que está conectado el módulo de relé a LOW, apagando el ventilador.

En el caso del circuito de control de acceso, se hace uso de la misma función, pero cambian las acciones realizadas al leer el dato recibido, en este caso en un "feed" diferente al anterior, que es al que está conectado el microcontrolador situado en este sistema.

Al leer el dato recibido, si este, una vez usada la función **toPinLevel(),** si el valor obtenido es "HIGH", se encenderá el LED de color azul que indica que se ha recibido la instrucción mediante la función **digitalWrite(LED_AZUL, data->toPinLevel())**y se usa la función **cerradura.write(180)** para mover el servomotor hasta la posición de 180º, simulando así la apertura de la puerta.

Tras esto, se crea un retardo de 4000ms antes de devolver el servomotor a su posición inicial con **cerradura.write(0)**, para finalmente, apagar el LED de color azul indicando que el movimiento del servomotor ha terminado.

Finalmente, para la inserción de los datos de cada uno de los accesos satisfactorios realizados en el circuito de control de acceso en la base de datos web de consulta, se ha hecho uso de la librería **HTTPClient.h** [16] para realizar peticiones HTTP al servidor donde está alojada la base de datos usando un script **PHP**, además de la función **insertarenDB()**.

A continuación, se puede observar un diagrama que unifica la librería y la función:

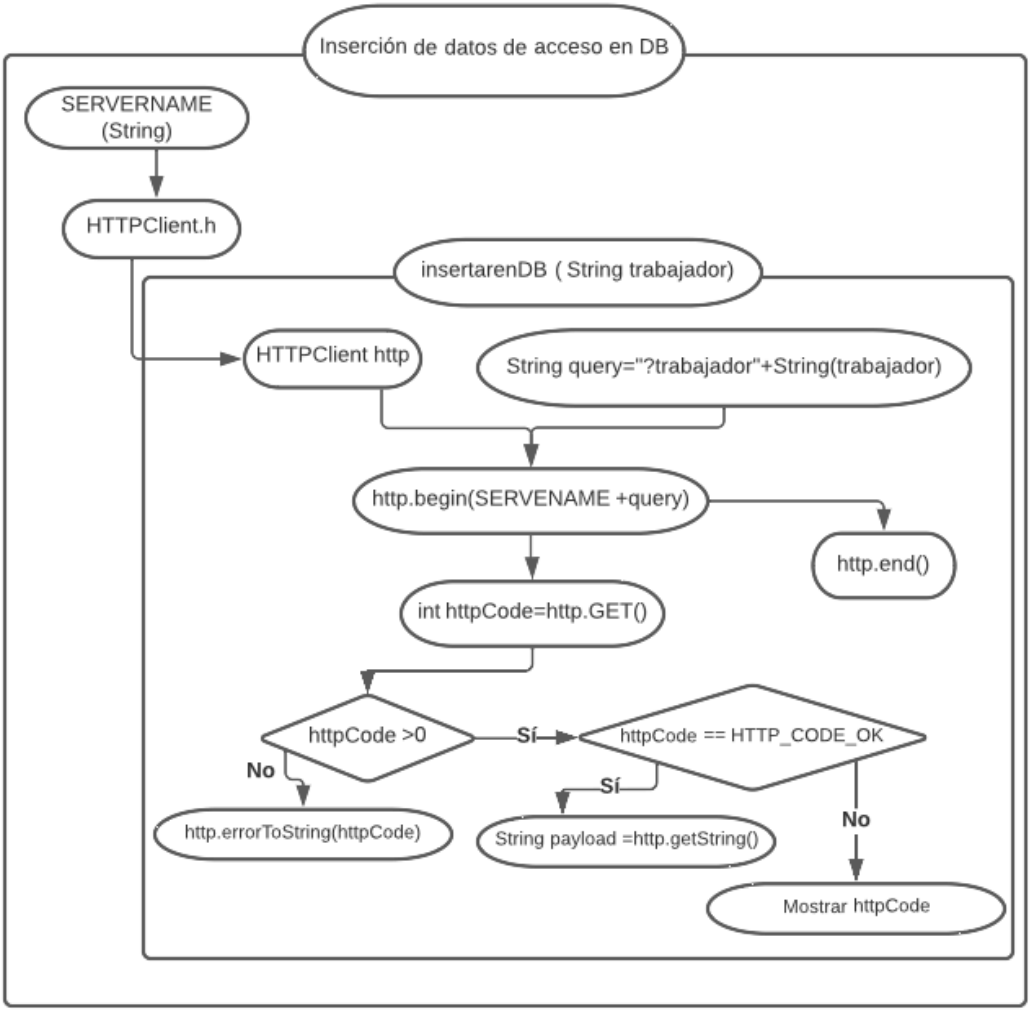


Ilustración 29. Diagrama de la función para la inserción de los datos de acceso en BD

Para poder usar la librería HTTPClient.h [18], se debe definir un servername, que en este caso será https://tfgmadariaga.000webhostapp.com/insert_nombre.php , haciendo uso del hosting web gratuito que ofrece la web 000webhostapp, e indicando el script PHP [22] que se encargará de hacer el inserto en la base de datos de los datos de acceso, como se explicará más adelante [21].

Una vez definido el nombre del servidor al que se hará la conexión HTTP, se va a describir el funcionamiento de la función **insertarenDB**, que a su vez hace uso de determinadas funciones de la librería HTTPClient, la cual recibe un objeto tipo String con el nombre del trabajador en cuestión del cual se insertarán los datos de acceso, obtenido en función del identificador aproximado al sensor RFID.

A continuación, se crea un objeto de la clase HTTPClient llamado http mediante la función **HTTPClient http**, y se añade un objeto tipo string llamado query el cual unirá el nombre del trabajador al string "?trabajador", necesario para poder hacer uso del script PHP que se describe a continuación.

Este script está almacenado en un fichero dentro del servidor web de la página de hosting, y su estructura es la siguiente:

```php
if (isset($_GET["trabajador"])){
 $trabajador =$_GET["trabajador"]; //Obtener nombre del trabajador del HTTP GET

 $servername= "localhost";
 $username="id19370746_fichajes_admin";
 $password="6H&82c!{oRo5Msqn";
 $database_name="id19370746_db_fichajes";
```

Ilustración 30. Cabecera del script PHP de inserción de datos de acceso en la BD

En primer lugar se comprueba que el nombre del trabajador viene en la petición HTTP recibida, puesto que sin él no se podrá ejecutar el script y mostrará un mensaje de error.

A continuación se definen los datos de conexión a la base de datos proporcionando los identificadores y contraseñas necesarios.

Posteriormente, se realiza la conexión a la base de datos mediante mysql y si esta es correcta, se realiza la consulta SQL que insertará los datos de acceso en la misma.

```php
//Conectar mediante MySQL la DB y el PHP
    $connection = new mysqli ($servername , $username, $password, $database_name);
//Se comprueba la conexion
    if ($connection -> connect_error){
     die("Ha fallado la conexión MySQL: " . $connection->connect_error);
    }// if connection error
//Si la conexion es correcta se realiza la insercion en la base de datos del nombre del trabajador
    $sql = "INSERT INTO tbl_fichajes (fichaje_nombre) VALUES ($trabajador)";

    if($connection->query($sql)==TRUE){
        echo "Se ha registrado el fichaje del trabajador satisfactoriamente";
    }//if connection true
    else{
        echo "Error: " . $sql . " => " . $connection->error;
    }//else

    $connection->close();
```

Ilustración 31. Conexión SQL e inserción de los datos de acceso en el script PHP

Una vez realizada la petición HTTP haciendo uso del script PHP, se almacena en la variable **httpCode** el código que devuelve la petición GET, el cual será erróneo si es negativo, comprobándose esto con un condicionante superior a 0, y en el caso de ser positivo, corrobora que el código recibido es HTTP_CODE_OK, indicando que la petición se ha realizado de forma correcta, y en un caso diferente se obtiene el código recibido para procesar el posible error.

Finalmente, se concluye la petición HTTP mediante la instrucción **http.end().**

Para poder visualizar en la aplicación móvil los datos de acceso disponibles en la base de datos alojada en el servidor web, se hace uso de un script PHP similar, al cual se accederá usando un fichero HTML que contiene el script de consulta de datos.

El script de consulta de datos, cuenta con la misma cabecera que la anterior, y la estructura de la consulta es la siguiente:

```php
$sql = "SELECT * FROM tbl_fichajes";

if($resultado = mysqli_query($connection,$sql)){

  if(mysqli_num_rows($resultado) > 0){


   echo "<table>";
     echo "<tr>";
  echo "<th> ID fichaje </th>";
  echo "<th> Nombre </th>";
  echo "<th> Fecha entrada </th>";
   echo "</tr>";

  while($row = mysqli_fetch_array($resultado)){

  echo "<tr>";
    echo "<td>" .$row['fichaje_id'] . "</td>";
    echo "<td>" .$row['fichaje_nombre'] . "</td>";
    echo "<td>" .$row['fichaje_timestamp'] . "</td>";
  echo "</tr>";

}//while

 echo "</table>";

 mysqli_free_result($resultado);

 } else{

    echo "No hay coincidencias con la consulta realizada";
```

Ilustración 32. Estructura del script de consulta de datos

Simplemente la consulta hecha a la base de datos obtiene todos los datos disponibles en la tabla de “fichajes”, y siempre y cuando obtenga algún resultado ($resultado >0), crea una tabla con el ID del acceso, que incrementa automáticamente, el nombre del trabajador y la hora del acceso, y va rellenando de forma dinámica todas las filas necesarias con los datos disponibles, y una vez termina, mediante la instrucción **mysqli_free_result($resultado)** libera(vacía) el objeto resultado para que no se acumulen los datos en futuras consultas.

4.3. Aplicación móvil de consulta

La aplicación móvil de consulta ha sido desarrollada con la herramienta MIT APP Inventor, haciendo uso de la programación por bloques, esta consta de 4 pantallas diferentes conectadas entre sí.

La primera pantalla, es una pantalla de bienvenida con el título del proyecto y el logo de la universidad, con un botón que al ser pulsado accede a la segunda pantalla, llamada Dashboard, la cual muestra tres botones mediante los cuales se podrá acceder a las pantallas que mostrarán los datos tanto de consulta de datos meteorológicos como de accesos al puesto de trabajo, o volver a la pantalla anterior.

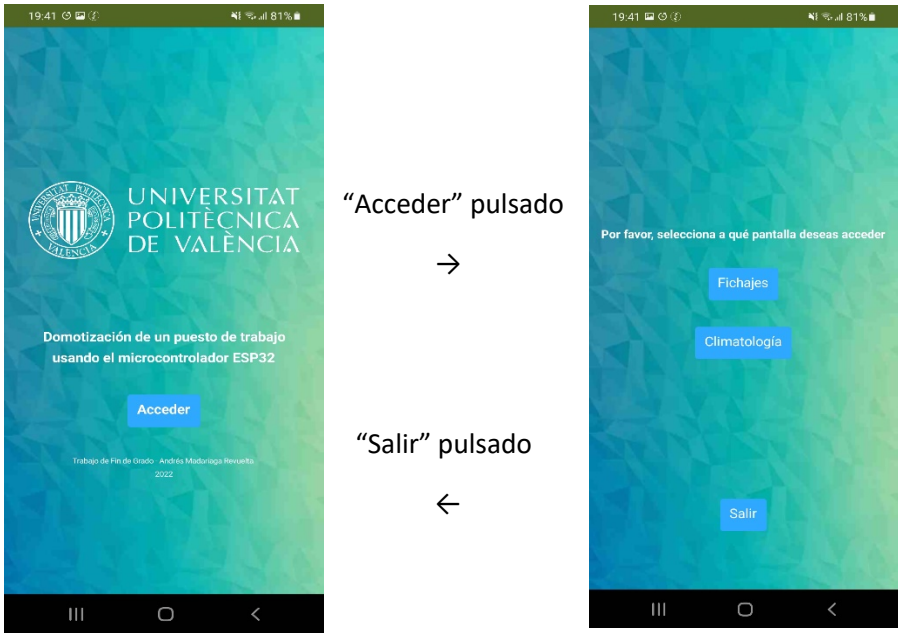


Ilustración 33. Pantalla "Principal" y "Dashboard" de la app móvil

Una vez se pulsa el botón "fichajes" se accede a la pantalla de consulta de los datos de acceso que han sido insertados en la base de datos, a continuación, se puede observar la interfaz gráfica de la pantalla, así como los bloques utilizados para su funcionamiento [25].

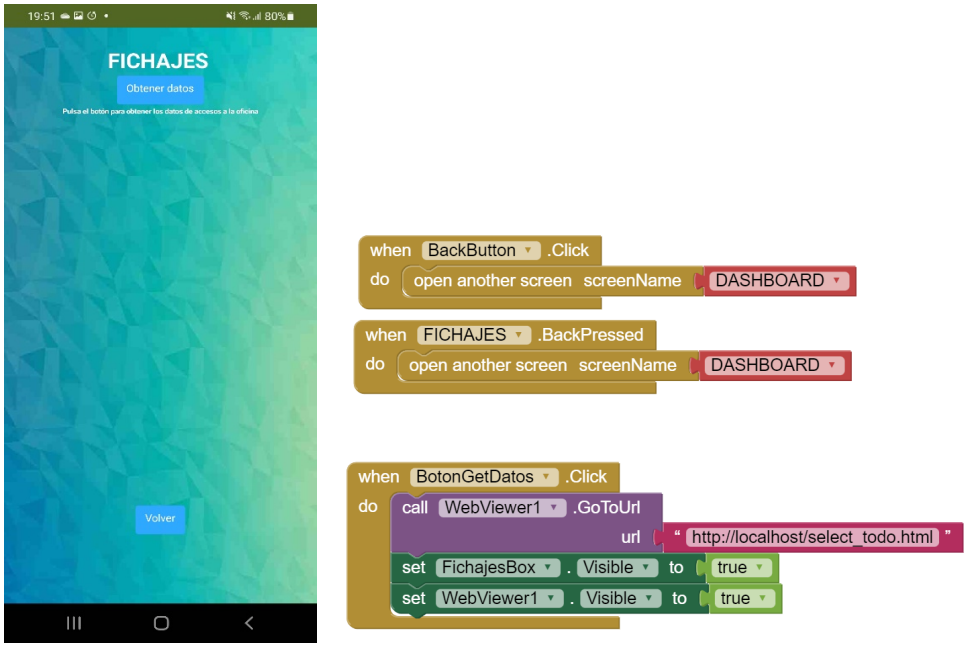


Ilustración 34. Interfaz gráfica y bloques de la pantalla "Fichajes"

La pantalla se muestra en primer lugar como se observa en la ilustración superior, y al pulsar el botón “Obtener datos” abrirá el documento HTML “select_todo”, el cual mediante el uso de un script PHP hará una consulta a la base de datos obteniendo todos los datos de accesos disponibles, que se mostrarán en pantalla como se ve en la ilustración a continuación.

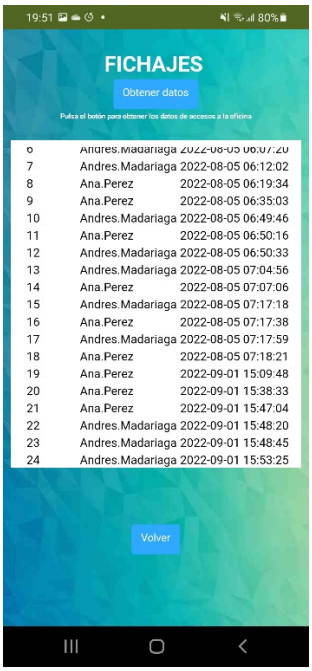


Ilustración 35. Interfaz gráfica de la pantalla "Fichajes" tras obtener los datos

Pulsando el botón “Volver”, se regresa a la pantalla “Dashboard” comentada anteriormente.

La pantalla de consulta de datos meteorológicos muestra 4 widgets, uno para la temperatura, otro para la humedad relativa, ambas instantáneas, y 2 gráficos de puntos que muestra los últimos datos tomados de estas para la consulta de las variaciones de estas, en las ilustraciones inferiores se pueden observar tanto la interfaz gráfica como los bloques usados para su funcionamiento [24].

Ilustración 36. Interfaz gráfica y bloques de la pantalla de consulta de datos meteorológicos

Los datos de temperatura y humedad instantáneos se muestran con un WebViewer accediendo directamente al Widget disponible en el servidor Web de ThingSpeak, mientras que para observar los gráficos se accederán a dos ficheros HTML, que accederán también a los widgets de ThingSpeak aunque mediante código HTML y CSS se han modificado visualmente para que se puedan consultar en el dispositivo móvil correctamente.

Al pulsar el botón “Volver”, como en el caso anterior se volverá a la pantalla “Dashboard”.

5. PRUEBAS Y RESULTADOS

El testeo del funcionamiento del sistema se ha llevado a cabo en varias fases, y por capas, en primer lugar, se ha testeado la capa física del sistema comprobando el funcionamiento de los sensores y actuadores y monitorizando los resultados mediante la comunicación serial disponible en la IDE de Arduino, comprobando que las mediciones eran correctas y no surgían errores, se comienza haciendo todas las pruebas del circuito que monitoriza los datos meteorológicos.

5.1. Testeo del circuito de monitorización de datos meteorológicos

Se comienza testeando el funcionamiento del sensor DHT11, conectándolo al microcontrolador y mostrando en la comunicación serial las medidas realizadas:

```
Temperatura: 30.50 °C.
Humedad: 76.00 %.
```

Ilustración 37. Medición del sensor DHT11

En la ilustración superior se pueden observar la temperatura y humedad medidas por el sensor, y en la ilustración inferior el resultado mostrado por la aplicación de tiempo disponible en Windows 11 en el mismo instante de tiempo.

Ilustración 38. Datos de medición de temperatura y humedad

Se puede observar que la medida de la temperatura coincide con el medido por el sensor, aunque el de humedad difiere, esto puede ser debido a que la casa desde la cual está tomada la medida se encuentra en un punto cercano al mar.

A continuación, se prueba que el LED que se ilumina al sobrepasar los 25ºC se enciende, como actualmente la temperatura ambiente es elevada y siempre se encontraría encendido, se sube ese margen de activación a 45ºC y se acciona un secador de pelo cerca para comprobar que esto se cumple.

En la ilustración siguiente se puede observar la temperatura medida por el sensor:

```
------------------------------
Temperatura: 53.30 °C.
Humedad: 26.00 %.
------------------------
```

Ilustración 39. Medición del sensor DHT11 forzando la temperatura

Y el LED encendido con el secador calentando el sensor:

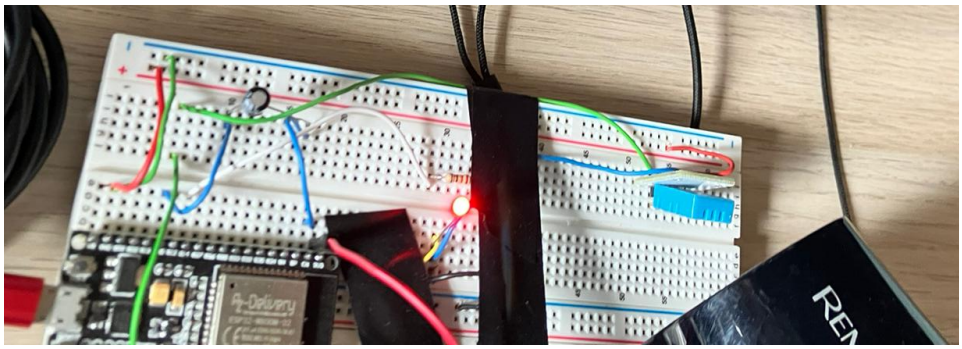


Ilustración 40.Encendido del LED indicador de temperatura

Una vez comprobado el funcionamiento del sensor de temperatura y el LED, se prueba el funcionamiento del ventilador mediante el uso del módulo de relé, en primer lugar, se conecta al microcontrolador y se comprueba que se acciona el electroimán, encendiendo el LED de color verde.

Ilustración 41. Activación del módulo del relé

Se conecta el ventilador y su alimentación externa al relé y se comprueba que al accionar el electroimán el ventilador comienza a funcionar, y de la misma manera al accionar el botón conectado al microcontrolador.

Una vez comprobado el funcionamiento de la capa física, se procede a conectar la capa de comunicaciones y testear su funcionamiento, en primer lugar, se prueba que los datos recogidos por el sensor se envían al servidor web de ThingSpeak y se almacenan allí.

En el puerto serie se muestran los siguientes datos:

```
Temperatura: 30.30 °C.
Humedad: 75.00 %.
-------------------------
Se han enviado los registros a ThingSpeak
------------------------------------------------
Temperatura: 30.40 °C.
Humedad: 75.00 %.
-------------------------
Se han enviado los registros a ThingSpeak
```

Y su valor almacenado en el servidor de ThingSpeak:

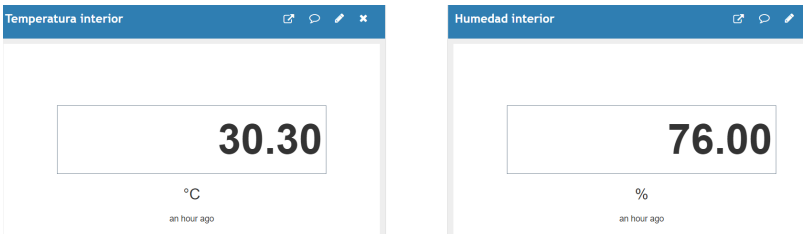


Ilustración 42. Datos mostrados en ThingSpeak

También una captura de cómo

se muestran en la aplicación móvil, haciendo uso del widget web que ofrece el servidor ThingSpeak:

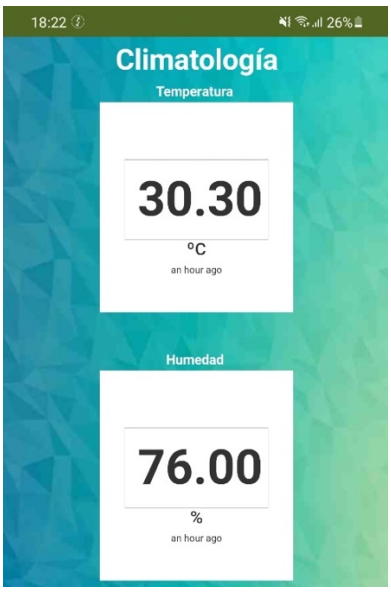


Ilustración 43.Visualización de datos en la pantalla de consulta de climatología

Finalmente, se prueba el funcionamiento del accionamiento del ventilador mediante el uso del asistente de voz de Google, para poder comprobar que funciona, se muestra en el monitor serie un mensaje en función de la instrucción recibida a través del asistente, que una vez realizada, mediante el uso de un applet de IFTTT, envía un dato (1 o 0) al servidor de Adafruit.io al que está conectado el microcontrolador y en función de ese dato recibido, muestra el mensaje en el monitor, también se puede consultar el funcionamiento en el video disponible como anexo.

```
Alguien ha usado un comando para <- ENCENDER LA VENTILACION
Temperatura: 30.30 °C.
Humedad: 77.00 %.
-------------------------
Se han enviado los registros a ThingSpeak
-----------------------------------------
Alguien ha usado un comando para <- APAGAR LA VENTILACION
```

Ilustración 44. Datos recibidos de Adafruit.io

5.2. Testeo del circuito de control de acceso

Se comienza testeando el funcionamiento del módulo RFIDRC522, aproximando a él un identificador RFID y comprobando que muestra en el monitor serie su UID en hexadecimal.

```
Scan PICC to see UID, SAK, type, and data blocks...
Card UID: CC 70 F0 2F
Card SAK: 08
PICC type: MIFARE 1KB
Sector Block   0  1  2  3   4  5  6  7   8  9 10 11  12 13 14 15  AccessBits
  15      63   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
          62   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
          61   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
          60   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
```

Ilustración 45. UID del identificador aproximado al sensor RFID

Una vez comprobado esto, se almacenan dos identificadores y se comprueba que cuando uno de los identificadores aproximados coincide con estos se enciende un LED de color verde, y en el caso contrario un LED rojo.

```
const String ANDRES="757B162C";
const String ANA="CC70F02F";
```

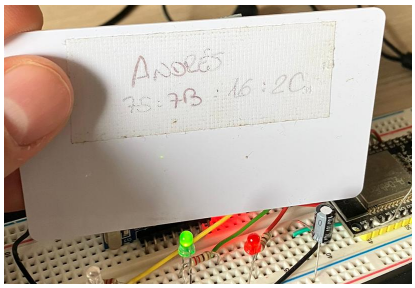


Ilustración 46. Encendido de LED con UID conocido

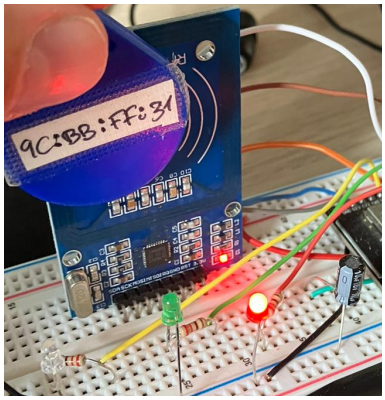


Ilustración 47. Encendido del LED rojo con UID desconocido

A continuación, se conecta el servomotor y se testea que cuando el identificador es uno de los registrados, se produce movimiento en este.

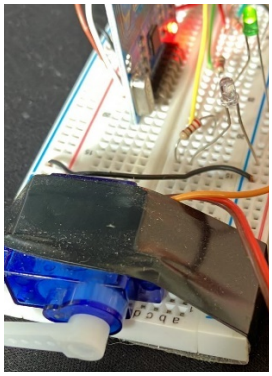


Ilustración 48. Movimiento del servomotor con el LED verde encendido (UID conocido)

Una vez testeada toda la capa física, se conecta la capa de comunicaciones y se comienzan a realizar las pruebas de la misma, en primer lugar, como se ha hecho en el anterior circuito, se comprueba el funcionamiento del servomotor haciendo uso de la instrucción del asistente de Google, en este caso, para indicar que el servomotor se está moviendo debido al uso de dicha instrucción, se encenderá un LED de color azul, cuyo encendido estará condicionado por el dato (1 o 0) que se reciba por parte del servidor web de Adafruit.io al que está conectado el microcontrolador.

```
.....
Adafruit IO connected.
Alguien ha usado un comando para <- ABRIR
```

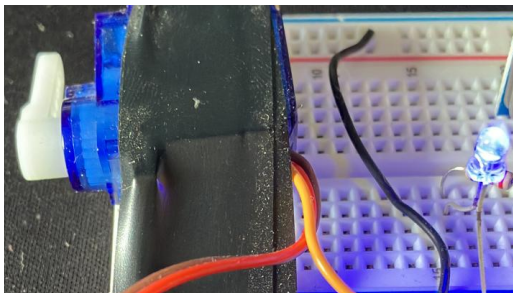


Ilustración 49. Movimiento del servomotor mediante comando de voz

Finalmente, se debe de comprobar que al aproximar una identificación conocida al módulo RFID, se inserta en la base de datos ubicada en un servidor Web el nombre de la persona asociada a dicha identificación y la fecha en la que se ha realizado dicha aproximación, lo que en el proyecto se ha llamado “fichaje”.

La petición HTTP que va dentro de un archivo PHP cogerá el nombre de la persona asociada y lo unirá al insert de la base de datos para que esto se produzca.

```
Se ha detectado una nueva tarjeta de identificacion
Bienvenido ANDRES
 SE HA ABIERTO LA PUERTA
Se ha registrado el fichaje del trabajador satisfactoriamente
```

Ilustración 50. Mensaje de confirmación de inserción en BD del acceso

Se puede observar debajo una captura de cómo se han insertado los fichajes en phpmyadmin, un gestor de bases de datos.

☐	Editar	Copiar	Borrar	19	Ana.Perez	2022-09-01 15:09:48
☐	Editar	Copiar	Borrar	20	Ana.Perez	2022-09-01 15:38:33
☐	Editar	Copiar	Borrar	21	Ana.Perez	2022-09-01 15:47:04
☐	Editar	Copiar	Borrar	22	Andres.Madariaga	2022-09-01 15:48:20
☐	Editar	Copiar	Borrar	23	Andres.Madariaga	2022-09-01 15:48:45
☐	Editar	Copiar	Borrar	24	Andres.Madariaga	2022-09-01 15:53:25

Ilustración 51. Accesos registrados en la BD

Finalmente se comprueba mediante la aplicación móvil, haciendo uso de un archivo PHP con una consulta SQL, que se puede consultar la base de datos desde el dispositivo.

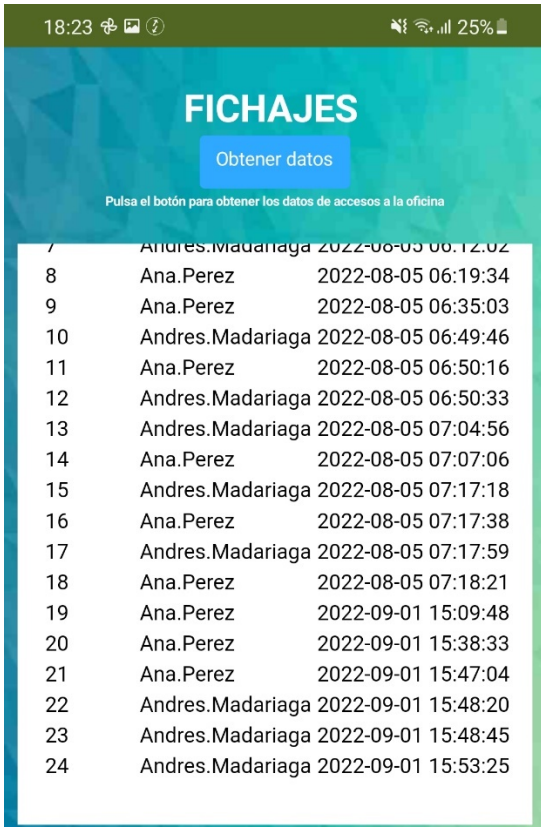


Ilustración 52. Visualización de los accesos en la pantalla "Fichajes" de la aplicación

6. CONCLUSIONES

En líneas generales se puede considerar que el objetivo con el que fue concebido el proyecto se ha logrado, el puesto de trabajo dispone de un control de acceso monitorizado y funcional y en el interior de este existe un control de la temperatura y humedad con la posibilidad de activar un ventilador mediante el uso de un botón o incluso un comando de voz al asistente de Google.

La idea de realizarlo surgió por la curiosidad sobre el mundo de la domótica y el descubrimiento de microcontroladores con una relación coste-potencia muy grande como es el caso del ESP32.

Como experiencia personal, he adquirido diversos conocimientos sobre la programación en lenguaje C++, dispositivos y sensores Arduino y compatibles, desarrollo de aplicaciones móviles y la integración de estos con algo tan cotidiano hoy en día como los asistentes de voz, facilitando o automatizando algunas funciones como en este caso.

He podido sobreponerme a diversas dificultades e incompatibilidades que han ido surgiendo durante el desarrollo de este, principalmente por realizar una documentación insuficiente, lo cual me ha servido para mejorar mis procesos de trabajo.

Como mejoras futuras que se pueden aplicar a este proyecto, la principal sería la diseñar una PCB en la que realizar todas las conexiones de forma más profesional, para poder darle un uso diario real.

Cabe mencionar la posibilidad de conectar un ventilador de tamaño real para hacer efectivo el objetivo de la existencia del mismo, simplemente colocando un relé compatible con el voltaje de este, conectar una cerradura funcional al servo para que realmente realice el bloqueo de la puerta de acceso o el añadir más sensores y módulos a los pines que han quedado libres en el microcontrolador, el código está preparado para albergar nuevas funciones ya que está implementado y comentado de forma sencilla, lo que facilitaría la tarea.

Como conclusión, la realización de este trabajo de fin de grado me ha permitido aplicar los conocimientos adquiridos durante el desarrollo de mis estudios y desarrollarme como profesional para afrontar nuevos retos.

REFERENCIAS BIBLIOGRÁFICAS

[1] L. Aljundi. "Using the arduino software (IDE) | arduino documentation". Arduino Docs. https://docs.arduino.cc/learn/starting-guide/the-arduino-software-ide (accedido el 2 de junio de 2022).

[2] R. Santos. "Installing ESP32 in Arduino IDE". Randommer Tutorials. https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/ (accedido el 2 de junio de 2022).

[3] "ESP32 - temperature humidity sensor". ESP32.io. https://esp32io.com/tutorials/esp32-temperature-humidity-sensor (accedido el 5 de junio de 2022).

[4] D.-Robotics. "DHT11 datasheet". Datasheetspdf.com. https://datasheetspdf.com/pdf-file/785590/D-Robotics/DHT11/1 (accedido el 5 de junio de 2022).

[5] Multicomp. "Datasheet diodo LED". Farnell | Electronic Component Distributors. https://www.farnell.com/datasheets/1498852.pdf (accedido el 8 de junio de 2022).

[6] "Sitio web de ThingSpeak". ThingSpeak. https://thingspeak.com (accedido el 11 de junio de 2022).

[7] "DHT11 with NodeMCU". how2electronics.com. https://how2electronics.com/dht11-humidity-temperature-nodemcu-thingspeak/ (accedido el 13 de junio de 2022).

[8] "ESP32 upload issue". Electronics Stack Exchange. https://electronics.stackexchange.com/questions/569788/how-esp32-auto-upload-issue-has-been-fixed-by-adding-capacitor (accedido el 16 de junio de 2022).

[9] "Datasheet Módulo 1 relé". Components101. https://components101.com/switches/5v-single-channel-relay-module-pinout-features-applications-working-datasheet (accedido el 19 de junio de 2022).

[10]"Interface one channel relay module with arduino". Last Minute Engineers. https://lastminuteengineers.com/one-channel-relay-module-arduino-tutorial/ (accedido el 21 de junio de 2022).

[11]"Datasheet ventilador 5V". Octopart. https://datasheet.octopart.com/KD0506PHB3-Sunon-Fans-datasheet-7277221.pdf (accedido el 23 de junio de 2022).

[12]"SG-90 servo datasheet". ee.ic.ac.uk. http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf (accedido el 25 de junio de 2022).

[13]"MFRC522 datasheet". NXP® Semiconductors Official Site | Home. https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf (accedido el 26 de julio de 2022).

[14]"GitHub - miguelbalboa/rfid: Arduino RFID Library for MFRC522". GitHub. https://github.com/miguelbalboa/rfid (accedido el 26 de julio de 2022).

[15]T. Treece. "Adafruit IO basics: ESP8266 + arduino". Adafruit Learning System. https://learn.adafruit.com/adafruit-io-basics-esp8266-arduino/arduino-io-library (accedido el 29 de junio de 2022).

[16]"ESP32 useful wi-fi library functions (arduino IDE) | random nerd tutorials". Random Nerd Tutorials. https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/ (accedido el 4 de junio de 2022).

[17]"GitHub - adafruit/adafruit_io_arduino: Arduino library to access adafruit IO from wifi, cellular, and ethernet modules". GitHub. https://github.com/adafruit/Adafruit_IO_Arduino (accedido el 3 de julio de 2022).

[18]"HttpClient-arduino library". Arduino - Home. https://www.arduino.cc/reference/en/libraries/httpclient/ (accedido el 8 de julio de 2022).

[19]"Sitio web de IFTTT". IFTTT. https://ifttt.com/explore (accedido el 19 de julio de 2022).

[20]T. Treece. "Using IFTTT with adafruit IO to make an iot door detector". Adafruit Learning System. https://learn.adafruit.com/using-ifttt-with-adafruit-io (accedido el 19 de julio de 2022).

[21]A. Alam. "Create a Web Server and save form data into MySQL database using PHP (Beginners Guide)". DEV Community. https://dev.to/satellitebots/create-a-web-server-and-save-form-data-into-mysql-database-using-php-beginners-guide-fah (accedido el 25 de julio de 2022).

[22]"PHP: MySQL Database". W3Schools Online Web Tutorials. https://www.w3schools.com/php/php_mysql_intro.asp (accedido el 28 de julio de 2022).

[23]"MIT app inventor". MIT App Inventor. https://appinventor.mit.edu (accedido el 28 de julio de 2022).

[24]J. A. Villalpando. "Cliente de thingspeak con app inventor". App inventor. Basic4Android. B4A. Estación meteorológica. Elastix. PHP MySQL. http://kio4.com/arduino/117B_Wemos_ThingSpeaker.htm (accedido el 29 de julio de 2022).

[25]M. Sabin. "Integrating app inventor applications with sql database services". Academia.edu-Share research. https://www.academia.edu/11789819/INTEGRATING_APP_INVENTOR_APPLICATIONS_WITH_SQL_DATABASE_SERVICES (accedido el 2 de agosto de 2022).

[26]"Basics of the SPI Communication Protocol". Circuit Basics. https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/ (accedido el 9 de julio de 2022).