



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo fin de grado

Ingeniería informática - Ingeniería de computadores

Nombre del proyecto: RELEXIA

Realizado por:

Antonio Vázquez Pérez

Tutorizado por:

Angel Francisco Jiménez Fernández

Departamento de Arquitectura y Tecnología de Computadores (ATC)

Zafra, 7 de septiembre de 2020

Este trabajo de fin de grado trata sobre la realización de un reloj inteligente con posibilidad de integración en el ecosistema domótico Amazon Alexa, el asistente virtual de Amazon.

El reloj cuenta con varias funciones como la consulta del tiempo atmosférico actual y futuro, indicar la temperatura/humedad de la sala, mostrar la hora actual y además actuar como lámpara de ambiente.

La característica más destacable de este dispositivo es su gran pantalla pixelada de 16x16 píxeles RGB que permite ser vista desde una gran distancia y su sensor de presencia para poder apagarse o encenderse según hay o no una persona observándolo.

También es posible controlarlo por ordenes de voz a través de Alexa sin tocarlo, es de muy fácil uso, sin instalaciones y solamente tiene un botón para ir cambiando entre funcionalidades.

Se ha utilizado un microcontrolador ESP32, el cual cuenta con conectividad wifi para poder acceder a Internet y hacer uso de dos API REST para obtener el tiempo y la hora a tiempo real.

Por último, también mencionar que se ha utilizado la tecnología de modelado e impresión 3D para crear el alojamiento de los componentes.

En el documento se describe todo esto en detalle.

Trabajo fin de grado

Relexia

Reloj inteligente integrado con amazon alexa

Índice

Introducción.....	5
Descripción.....	5
¿Por qué este proyecto?.....	7
Objetivos académicos.....	8
Actualidad.....	9
Mercado.....	9
Comparativa y factores diferenciativos.....	14
Planificación y cronología.....	16
Planificación inicial.....	16
Cronología real y gestión del cambio.....	17
Características.....	19
Características funcionales.....	19
Características técnicas.....	20
Software utilizado.....	23
Visual Studio Code – PlatformIO.....	23
GitHub.....	24
FreeCAD.....	25
Cura slicer.....	25
GIMP.....	26
LibreOffice Writer.....	26
Desarrollo.....	27
Análisis y bibliotecas de los periféricos.....	27
Tarjeta SD.....	28
RTC (Real Time Clock).....	30
Humedad y temperatura.....	35
Sensor de presencia (PIR).....	43
Pantalla LED RGB 16x16.....	46
Sistema operativo.....	59
Funciones de red.....	59
Sistema principal – Escenas (tareas).....	62
Montaje.....	65
Coste del proyecto.....	72
Conclusiones.....	73

Introducción

Descripción

Tras la deliberación y consulta previa con mi respectivo tutor, *Ángel Jiménez*, del *Departamento de Arquitectura y Tecnología de Computadores* de la *Universidad de Sevilla*, se ha llegado a la conclusión con respecto a este trabajo de realizar un reloj basado en un microcontrolador, con funciones propias de un dispositivo domótico.

Dicho reloj no solamente realiza la tarea de un reloj al uso de entregar la hora, sino también la de proveer al usuario de otra información; tales como son las condiciones atmosféricas actuales/futuras, temperatura y humedad de la sala o la de actuar como lámpara de noche.

La peculiaridad principal de este proyecto es su pantalla LED RGB de 16x16 píxeles, capaz de representar de forma muy visual todo lo que desea comunicar.

Se trata de un proyecto muy completo que trata de englobar lo máximo posible todo lo aprendido en Ingeniería de computadores, más concretamente, en relación al departamento de arquitectura y tecnología de computadores.

Este documento va a relatar todo el proceso que se ha seguido para llevar a término este proyecto y toda la información que es necesaria saber para entenderlo.

Primero vamos a comenzar analizando que otras opciones se encuentran ya en el mercado que sean parecidas a esta solución que se propone para poder valorar como de novedoso o buena es la alternativa propuesta a la oferta ya existente.

Seguido por la planificación que se ha seguido en el proyecto, las características con las que contará el mismo y como ha sido su desarrollo.

Finalmente explicamos las conclusiones del proyecto y el resultado obtenido.

Dicho esto vamos a dar paso a la sección de *por qué nos hemos decantado por este proyecto*.

Pero antes, vamos a ver la imagen de la pantalla principal del dispositivo para irnos haciendo una idea, como vemos es capaz de mostrar bastante información únicamente con una matriz de 16 píxeles de ancho con un simple vistazo.

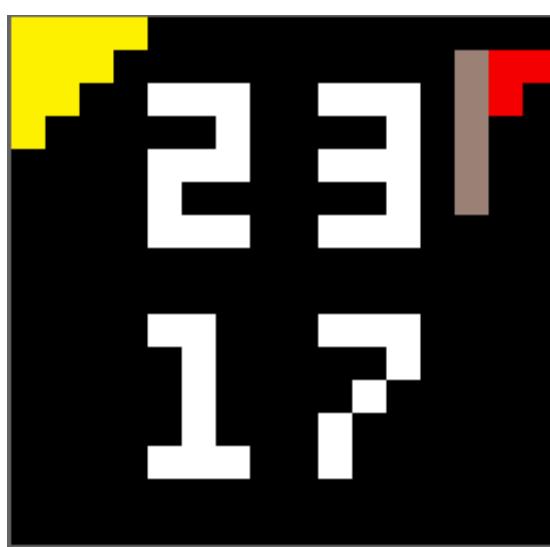


Figura 1: Pantalla principal (Clima → parte superior izq.(soleado); viento → banderín a la derecha; hora → en el centro)

¿Por qué este proyecto?

Aunque en un principio se barajaron otras opciones esta fue la mas viable debido a que es la que mejor se ajustaba al tiempo que se había de dedicar a esta asignatura.

Al estar basado en cosas que hemos dado en profundidad en la carrera sería menos confuso entrar al proyecto que si hubiera sido de otro tipo. La única parte mas desconocida es la de interacción con Alexa, que en cierto punto se encuentra ahí para añadir mas versatilidad y demostrar capacidad de resolución ante cosas no tan conocidas.

En la actualidad, el mundo de la domótica, el IoT o la inteligencia artificial se encuentran a la orden del día en la informática.

Existe ya un gran rango de dispositivos que cubren diversas funciones en nuestro día a día, de hecho, un reloj no es nada nuevo. Lo que lo hace único es su forma tan visual de comunicar las cosas, que es fácil de entender para todos los públicos y que además se integra perfectamente en un ecosistema domótico ya existente tan popular como es alexa, de Amazon.

Además de ser un proyecto que me permite cumplir la función de demostrar todos mis conocimientos, se trata de algo novedoso y actual.

Como veremos mas adelante, ya existen en el mercado relojes bastante parecidos a este en cuanto a la pantalla con píxeles anchos, pero no tan asequibles y con la misma capacidad de comunicación.

Objetivos académicos

El hecho de utilizar un microcontrolador, y tener que responder rápidamente a una interfaz humana le da el carácter de sistema empotrado en tiempo real.

Se han utilizado además otro tipo de software o técnicas que no hemos dado en la carrera que ponen de manifiesto el interés por parte del alumno en conocer sobre la Ingeniería Informática.

Las capacidades que se van a tratar de demostrar en este proyecto son las siguientes:

- Programación de microcontroladores.
- Estudio, comprensión, e integración de nuevos sensores y periféricos.
- Creación de un sistema que cuente con interfaz humana visual.
- Conocimiento del funcionamiento de la red y comunicaciones inalámbricas.
- Uso de APIs REST.
- Integración con dispositivos ya existentes en el entorno domótico.
- Uso de herramientas ya existentes para incrementar la funcionalidad de mi dispositivo a un bajo coste, como por ejemplo Alexa.
- Modelado 3D e impresión y su posterior uso en prototipado.
- Creación de bibliotecas gráficas

Actualidad

Mercado

El dispositivo que se pretende desarrollar en este proyecto se encuentra inspirado en gran medida en otras alternativas que ya existen en el mercado.

Conviene analizar estas alternativas para así poder comparar y saber si se trata realmente de una mejora o no con respecto a lo que ya había, además de comprobar que grado de competencia obtendríamos si se decidiera lanzar al mercado.

De entre los dispositivos del mercado mas parecidos a este solamente he podido encontrar una empresa:

Divoom (tivoo max)



Figura 3: Divoom tivoo max frontal Fuente: www.divoom.com



Figura 2: Divoom tivoo max trasera Fuente: www.divoom.com

Por un precio de 149\$ tenemos este Divoom tivoo max, por poner un ejemplo de esta compañía, ya que disponen de una amplia variedad de productos del mismo estilo pero diferente rango de precios.

De entre las funciones que es capaz de realizar el dispositivo de más alta gama tenemos estas que se muestran en la siguiente imagen :



Figura 4: Habilidades divoom tivoo max Fuente: www.divoom.com

Esta empresa también fabrica altavoces, de ahí el empeño en su publicidad por hacer énfasis en sus características sonoras.

El rango de precios que abarcan es desde los 49,90\$ hasta los 149\$ y varían en tamaño y características.

Aquí tenemos una imagen de su tienda online en la que se muestran todos los distintos tipos que tienen a la venta:

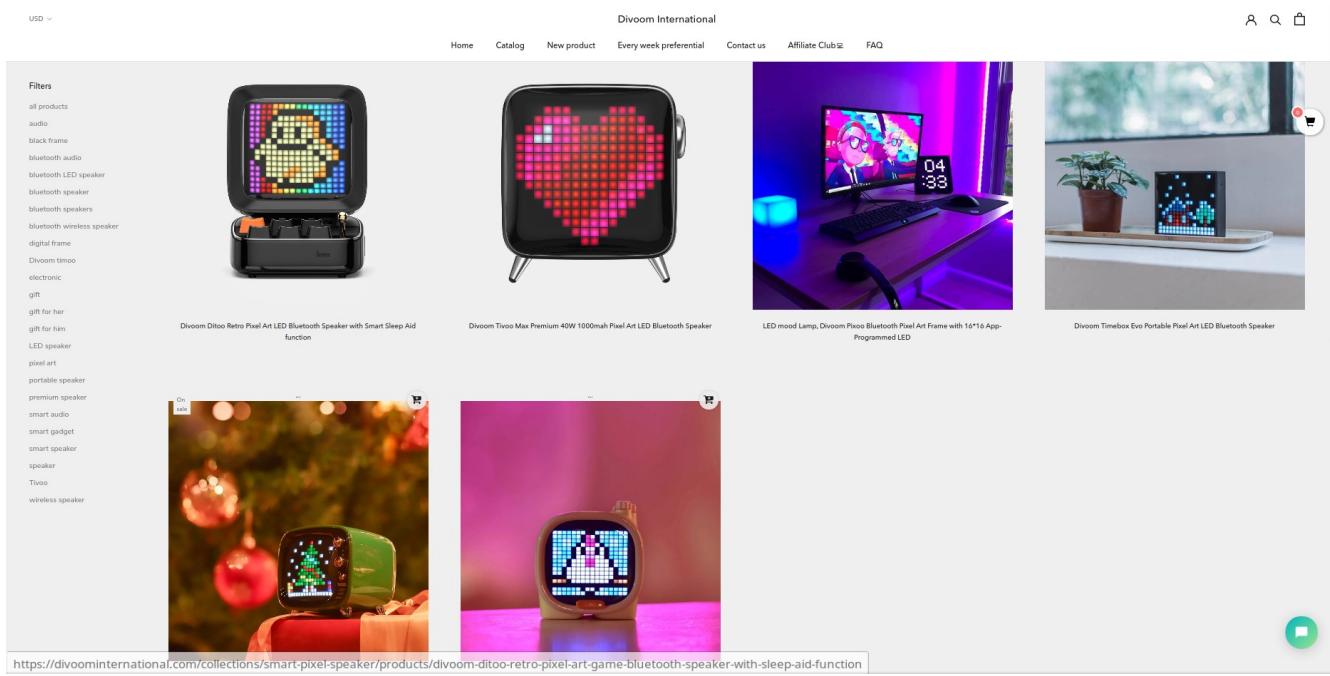


Figura 5: Tienda online divoom Fuente: www.divoom.com consultado el 4/9/2020

Echo Show

Apartándonos de los relojes que no dispongan una pantalla que sea una matriz de píxeles de gran tamaño RGB(pixeladas) podemos encontrar otros dispositivos como pueden ser los Echo Show de la propia Amazon.

Estos tienen integrados su sistema domótico alexa y además ofrece una interfaz visual en la que poder ver un rango amplio de información y otras posibilidades como videoconferencia, a un precio actualmente de 59,99€.



Figura 6: Echo show Fuente: Tienda amazon, www.amazon.es



Conecta por videollamada usando solo la voz con otros dispositivos Echo con pantalla, la app Alexa o Skype. Conecta rápidamente con otros dispositivos Echo compatibles de tu hogar. [Más información](#) acerca de las llamadas con Alexa.

Figura 7: Echo show conferencia Fuente: Tienda amazon, www.amazon.es

Este Echo Show es realmente una buena opción cuando se trata de competir tanto en funcionalidad como en precio.

Otros

Más vendido

LATEC Reloj Despertador Digital, LED Pantalla Reloj Alarma Inteligente con Temperatura, Puerto de Carga USB, 12/24 Horas, 4 Brillo Ajustable, Funció... ★★★★★ ~ 952 17,99€ <small>Entrega GRATIS el miércoles, 9 de septiembre para miembros de Prime</small>	HOMVILLA Reloj Despertador Digital con Pantalla LED de Temperatura, Alarma de Espejo Portátil con Alarma Doble Tiempo de Repetición 4 Niveles... ★★★★★ ~ 774 17,99€ <small>Entrega GRATIS para miembros de Prime</small>	Reloj despertador inteligente multifuncional Reloj digital inteligente Altavoz Bluetooth Radio FM Snooze Función AUX-IN con luz de noche LED... ★★★★★ ~ 23 28,99€ <small>prime Entrega GRATIS sábado, 5 de septiembre</small> <small>Clase de eficiencia energética: A+++</small>	Luces nocturnas Altavoz Bluetooth, Ranipobo Sensor táctil Lámpara de cabecera con reloj despertador, Reproductor de música MP3,Radio FM,... ★★★★★ ~ 277 26,98€ <small>prime Entrega GRATIS sábado, 5 de septiembre</small>
Annsky Despertador Digital, LCD Pantalla Reloj Alarma Inteligente Simple y con Pantalla de Fecha y Temperatura Función Despertador,... ★★★★★ ~ 845 11,99€ <small>prime Entrega GRATIS sábado, 5 de septiembre</small> <small>Más opciones de compra</small> 10,22 € (2 ofertas usadas y nuevas)	LBELL Luz Despertador, Wake up Light LED Digitales Wifi con Simulación de Amanecer/Atardecer reloj despertador con Radio FM Función Snooze Alarma ... ★★★★★ ~ 273 38,99€ <small>+29,99€</small> <small>Entrega GRATIS el miércoles, 9 de septiembre para miembros de Prime</small>	Amazon Echo Spot - Reloj despertador inteligente con Alexa, negro ★★★★★ ~ 1.341 89,99€ <small>+29,99€</small> <small>prime Entrega GRATIS lunes, 7 de septiembre</small>	LOFTER Despertador Luz Wifi Wake Up Light LED Despertador Amanecer Natural Lampara Inteligente Compatible con Alexa Echo y Google... ★★★★★ ~ 147 36,99€ <small>+29,99€</small> <small>prime Entrega GRATIS lunes, 7 de septiembre</small> <small>Clase de eficiencia energética: A</small>

Figura 8: Tienda digital Amazon Fuente: www.amazon.es consultado el 4/9/2020

Finalmente pues tenemos al resto de relojes despertador económicos que se pueden encontrar en tiendas online como amazon:

Estos relojes no se apartan mucho de dar la hora con displays LCD o LED simples y servir como lámpara nocturna o radio.

Como podemos ver suelen tener un precio de entrada sobre los 10€ y acabar en unos 40€.

Comparativa y factores diferenciativos

Hemos podido observar, que desde la perspectiva del mercado, este tipo de relojes pixelados van mas orientados a tener una estética retro o sacarle partido a este tipo de pantalla para mostrar imágenes con un atractivo pixelArt.

En cuanto a integración en un sistema domótico no se encuentran ejemplos de este estilo.

Tenemos a el dispositivo de Amazon, el cual posee una pantalla de mucha mejor calidad y está dotado de mas funciones que el anterior, pero aun así, creo que en un entorno realista serviría como entretenimiento los primeros días pero luego quedaría abandonado en un rincón ya que hay que acercarse para poder ver la pantalla y Alexa puede funcionar por voz.

Esto es algo que he intentado suplir colocando una pantalla grande, que pueda ser observada por personas de todas las edades y además poniendo un sensor de presencia, que dota al reloj de una facilidad de activación automática, preparado para ser visto en cualquier momento incluso desde largas distancias.

Como tiene una conexión con Alexa es posible controlarlo sin necesidad de tocarlo, solamente haciendo uso de la voz.

Creo además que ofrece la información oportuna simplemente a golpe de vista y de forma muy icónica, de nuevo, sin tener que preocuparte de activar nada ni necesitar ninguna aplicación.

A modo resumen podemos deducir las siguientes ventajas:

- Facilidad de visión
- No necesita una activación
- Sin instalaciones
- Información directa para el día a día
- Atractivo visualmente
- Sensores de temperatura y humedad
- Control con un solo botón

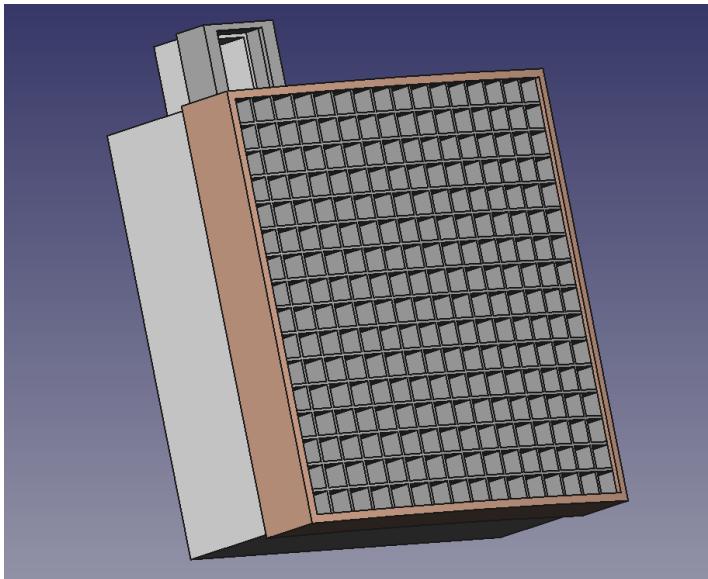
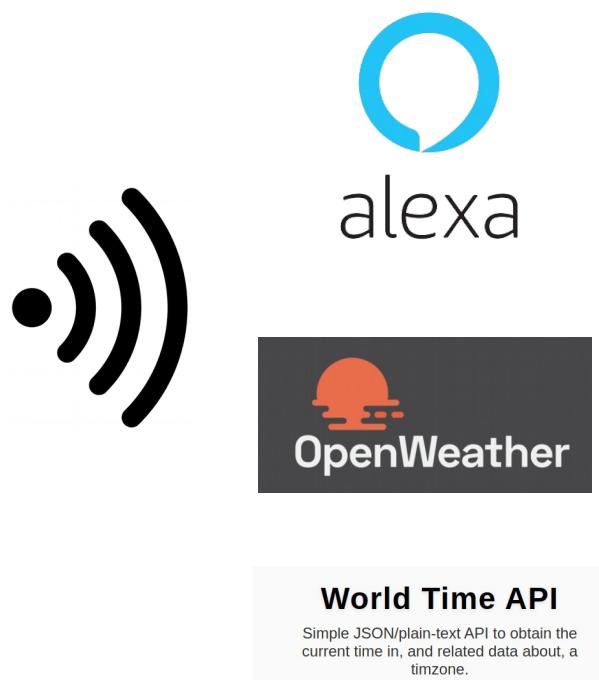


Figura 9: Diseño 3D del dispositivo, conexiones



World Time API

Simple JSON/plain-text API to obtain the current time in, and related data about, a timezone.

Planificación y cronología

Planificación inicial

Se elaboró inicialmente el siguiente diagrama de gantt para la planificación del proyecto tras la adjudicación del mismo:

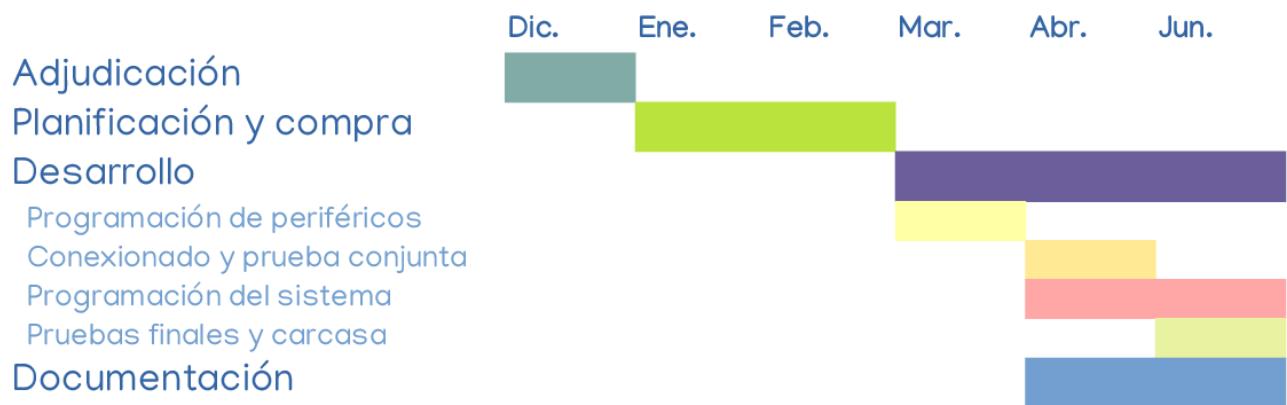


Figura 10: Diagrama de Gantt

Este diagrama no tardará en truncarse tras darse la situación extraordinaria del coronavirus. No se cumplieron los plazos debido a que ya para comenzar los componentes se retrasaron en la llegada.

A continuación vemos la cronología real.

Cronología real y gestión del cambio

- Fecha de adjudicación del proyecto: 17/12/2019

Comienza la búsqueda de los componentes a usar en el proyecto y planificación.

- Compra de componentes 14/2/2020

Todos los componentes electrónicos necesarios para su realización fueron adquiridos en esta fecha por la plataforma online AliExpress debido a los bajos costes en comparación a otras tiendas.

Por los tiempos de envío de dicha tienda, aproximadamente 15-30 días, se piensa comenzar este proyecto a partir de marzo.

- Fecha de inicio del proyecto: 8/3/2020

Debido a los problemas en china con el coronavirus los pedidos se retrasan entre 15 y 30 días más. Aun así comienza el desarrollo con los componentes que van llegando.

- El cambio de dinámica en las clases a no presencial hace que la cantidad de entregas y trabajo en casa de los alumnos se incremente, debatiéndose el resto del curso entre plazo y plazo de entrega, lo que lleva a un avance muy lento del proyecto.

- Aplazamiento del proyecto 8/6/2020

Llega la fecha de entrega y el proyecto no ha sido finalizado. Se aplaza hasta el periodo de verano para su posterior entrega en septiembre.

Con dicha situación resultó muy complicado llevar cualquier tipo de seguimiento e iba continuando con el proyecto en cuanto tenía un hueco libre.

Características

Características funcionales

Vamos a pasar a describir de una forma clara y concisa que funcionalidades se esperan que cumpla este proyecto.

Son las siguientes:

- *Entregar la hora actual*, con habilidad de conservar la hora programada incluso tras un corte de alimentación.
- *Detección de movimiento*, es capaz de apagarse o encenderse en función de si detecta movimiento en la sala.
- *Temperatura y humedad de la sala*, gracias a un sensor con precisión de +- 0.3ºC y +- 2% respectivamente.
- *Tiempo atmosférico actual / futuro*.
- *Lámpara de noche o de ambiente, su pantalla le permite poner todos los píxeles a un color fijo o siguiendo un patrón con brillo variable*.

Características técnicas

Placa de desarrollo microcontrolador ESP32-WROOM-32

Se ha escogido esta placa porque además de poseer una capacidad de cálculo y memoria RAM/ROM más que suficiente para las necesidades del proyecto, nos integra ya la conectividad wifi y bluetooth.

Su bajo coste le hace una excelente opción.

Tiene una función que nos será de mucha utilidad para ahorrar tanto electricidad como vida útil del procesador, es la de «*deep sleep*», en la que entrará cuando se detecte una inactividad desde los periféricos.

Este componente será el principal dentro de esta configuración hardware ya que albergará todo el funcionamiento del dispositivo y es el que más tareas cumple dentro del sistema.

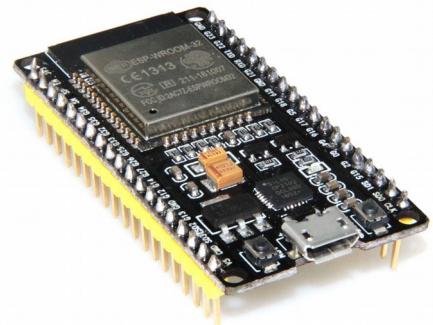


Figura 11: ESP32-WROOM-32

Sensor de temperatura y humedad AHT10

Este sensor se compone a su vez de otros dos, uno de temperatura y otro de humedad. Ambos tienen una precisión para nada despreciable $\pm 0,3^{\circ}\text{C}$ y $\pm 2\%$.

Su buena precisión a bajo precio lo convierte en uno de los mejores del mercado.



Figura 12: Sensor AHT10

Sensor de movimiento PIR(Passive infrared radiation)

Se trata de un sensor que es capaz de detectar movimiento utilizando un sensor de infrarrojos. Posee un gran ángulo de visión y devuelve una señal digital, lo que nos facilita en gran medida su uso.

Su sensibilidad se puede ajustar mediante dos potenciómetros.



Figura 13: Sensor infrarrojos PIR

Reloj en tiempo real (RTC)

Este módulo se ha utilizado para que, gracias a su pila, el reloj pueda mantener la hora programada incluso después de un corte de luz.

Tiene una interfaz de comunicación serie síncrona, utiliza dos cables, uno de datos y otro de reloj.



Figura 14: Reloj RTC

Matriz LED RGB 16x16 pixeles

Esta pantalla tiene la particularidad de que sus leds son capaces de retener su estado incluso si no reciben ningún tipo de señal. Toda la pantalla es capaz de recibir información por un único cable de datos.

Tiene una amplia gama de colores y un gran brillo, su disposición es como si todos los leds estuvieran en serie en zig-zag. Se hablará mas adelante de esto.

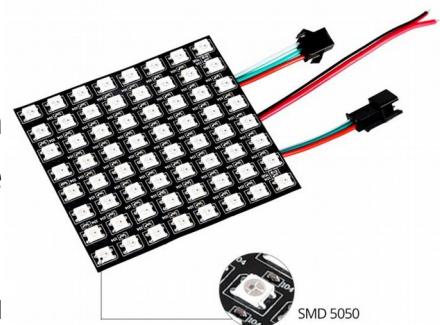


Figura 15: Matriz led RGB

Fuente de alimentación

La fuente de alimentación a utilizar es capaz de proveer hasta 10A. Esta alta intensidad es necesaria debido a que la pantalla led, con todas sus luces encendidas con un color blanco y a máxima intensidad es capaz de absorber toda esta potencia.

Como no se espera que se den estas condiciones se considera suficiente para alimentar al sistema.



DC 5V 10A 50W

Figura 16: Fuente de alimentación 50W

Software utilizado

Visual Studio Code – PlatformIO

Este es el software al que más tiempo he dedicado trabajando. Por si solo no serviría de nada sin la ayuda de PlatformIO, un complemento que añade la opción de análisis sintáctico, compilador y puerto serie para una gran variedad de microcontroladores, entre ellos el esp32.

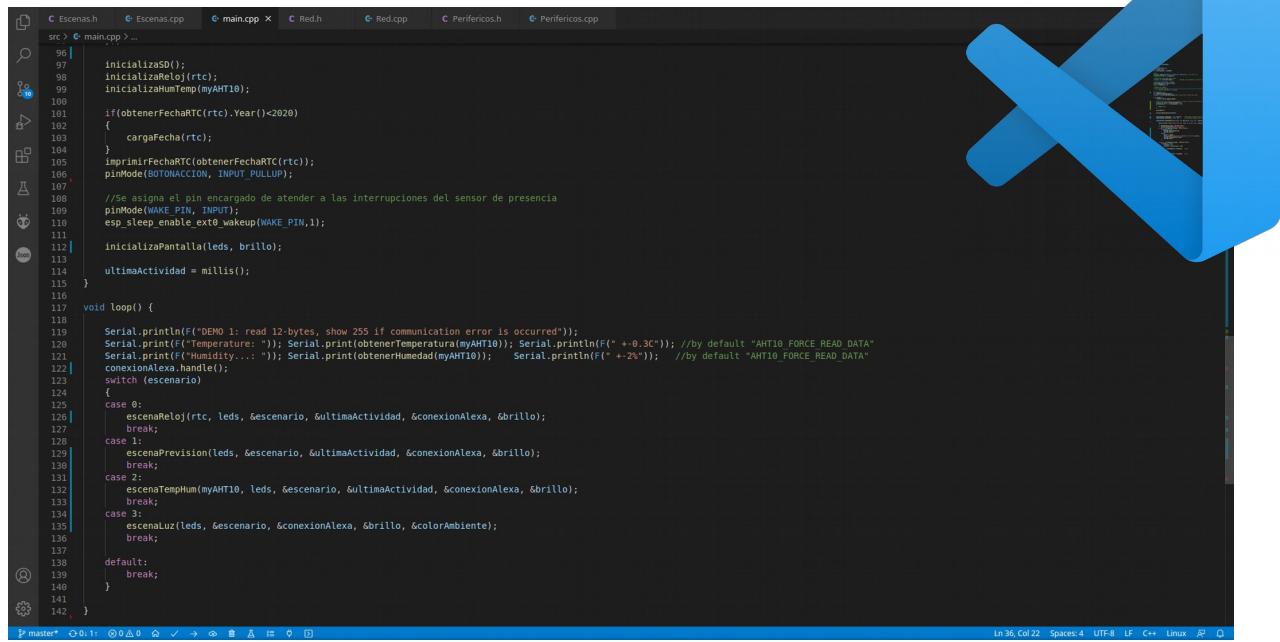


Figura 17: Software visual studio code

PlatformIO nos permite administrar todas las librerías del proyecto de una forma muy fácil e instalar las que deseemos con un click.

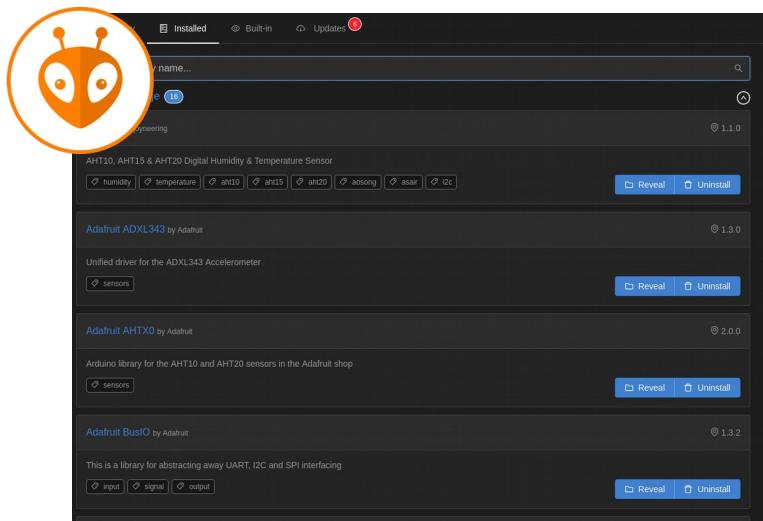


Figura 18: Complemento platformIO

GitHub

He llevado una copia de seguridad del proyecto en la plataforma GitHub dando también acceso también al tutor para el caso en que fuera necesario obtener ayuda.

También servirá como expositor del proyecto para futuras demostraciones.

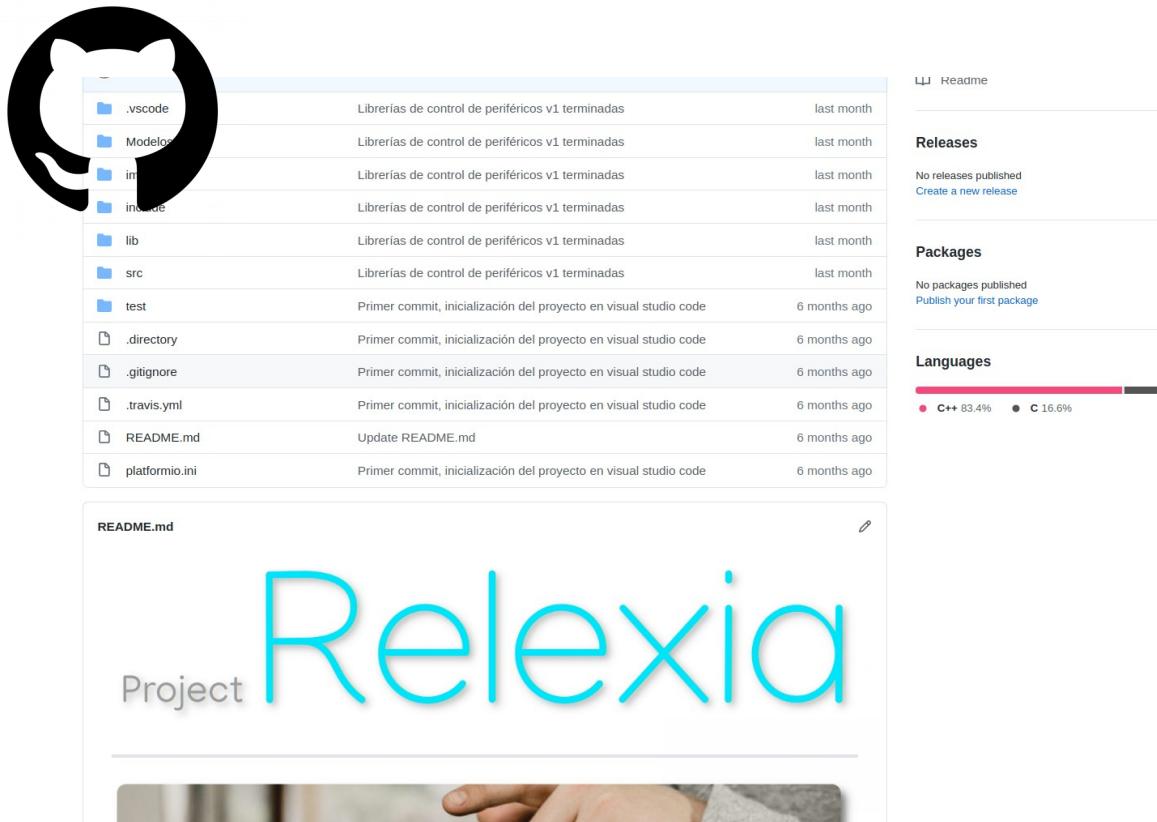


Figura 19: Repositorio en www.github.com

FreeCAD

Es un software de diseño 3D gratuito que me ha permitido hacer los modelos de las piezas del dispositivo para luego ser imprimidas.

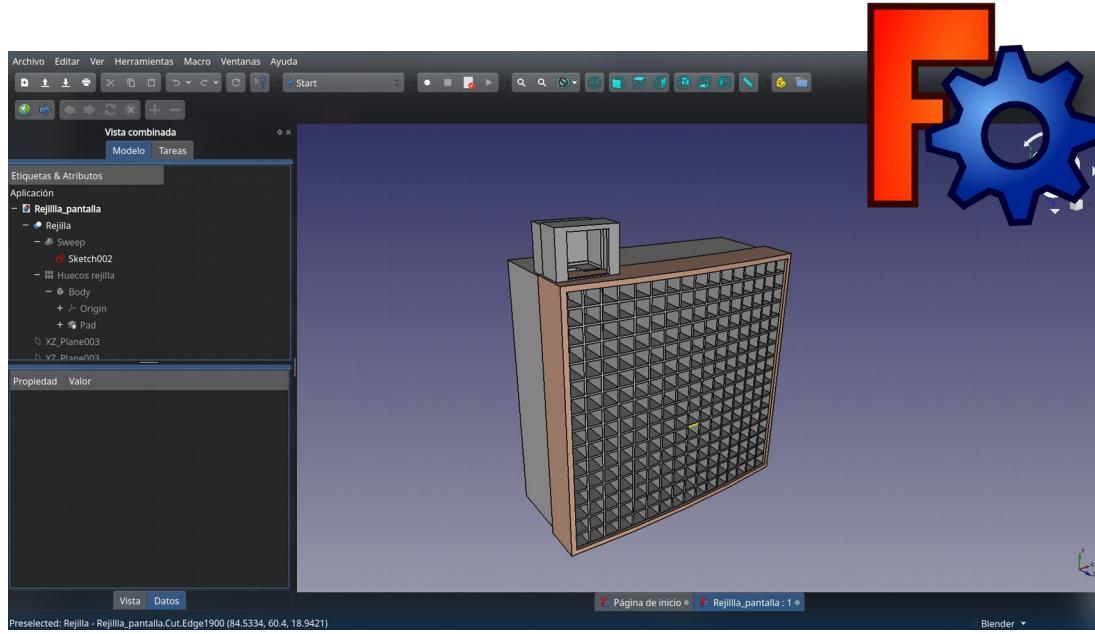


Figura 20: Software FreeCAD

Cura slicer

Utilizado para preparar las piezas a ser imprimidas.

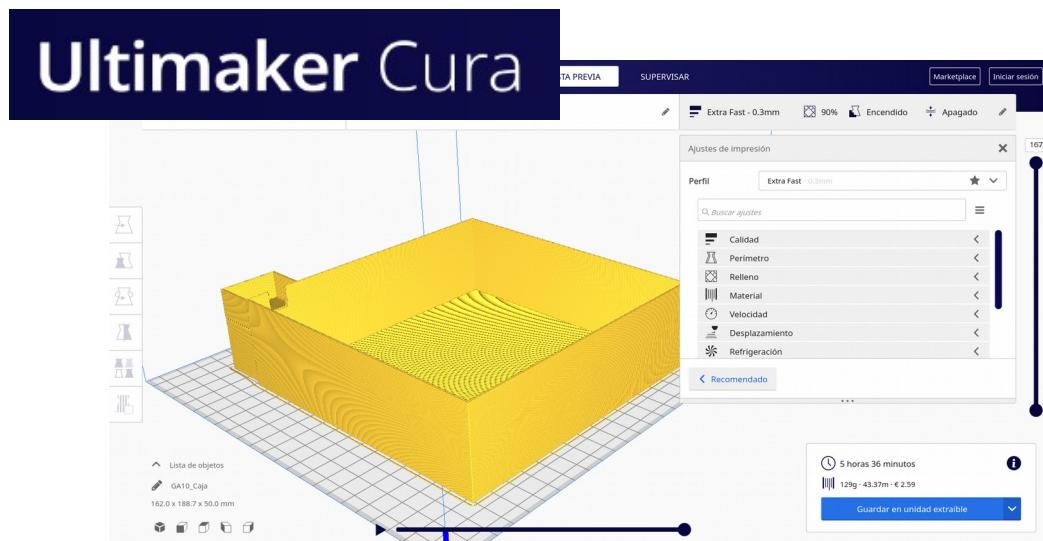


Figura 21: Software ultimaker Cura

GIMP

Este software me ha resultado muy útil para diseñar los iconos y escenas que luego se representarán en la pantalla.

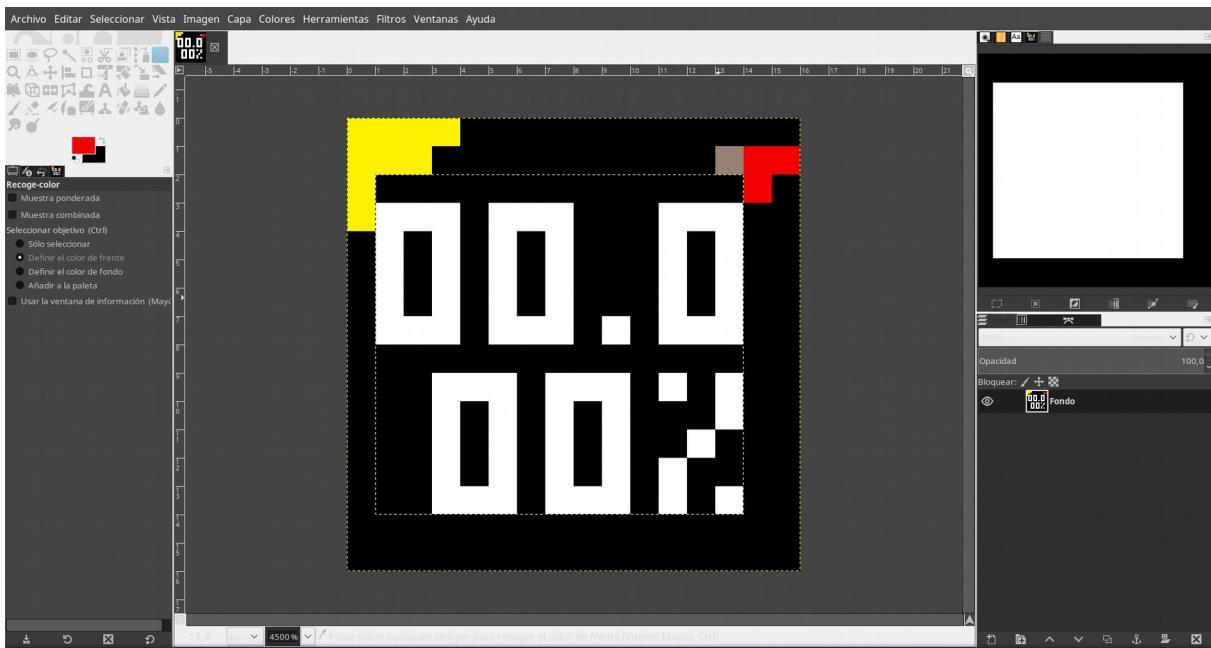


Figura 22: Software GIMP

LibreOffice Writer

Me he servido de el para elaborar esta memoria.

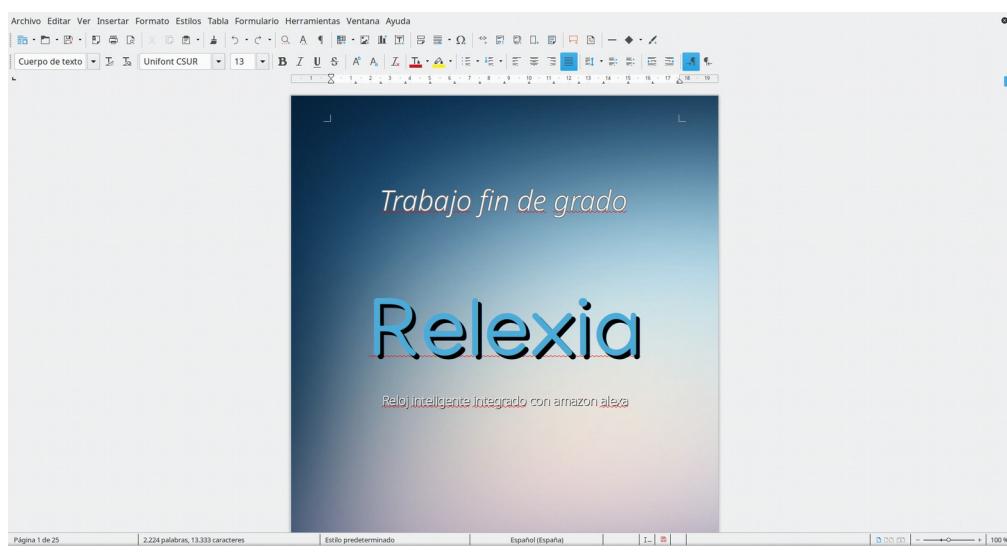


Figura 23: Software LibreOffice writer

Desarrollo

Análisis y bibliotecas de los periféricos

El primer paso que hemos tomado ha sido el de ir conectando todos los periféricos que se vayan a poner uno por uno, programándolos y comprobando su correcto funcionamiento, para luego acabar poniéndolos todos a la vez.

Para mantener el código ordenado y lo más modularizado posible se ha creado la biblioteca *Perifericos.h*, la cual va a contener todas las funciones y dependencias de otras bibliotecas correspondientes a nuestros periféricos.

Se va a proceder a describir las partes más destacadas de los periféricos y su funcionamiento.

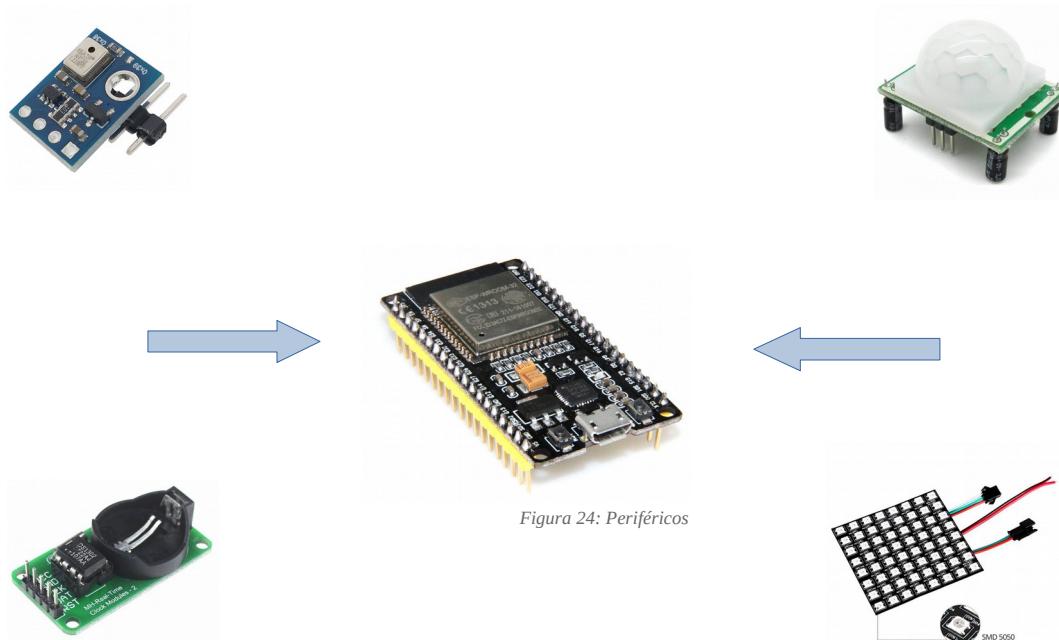


Figura 24: Periféricos

Para comprender el funcionamiento de cada dispositivo se ha tenido que investigar por internet y en algunos casos ver el datasheet del elemento.

Han sido de gran ayuda las librerías ya hechas para estos.

Tarjeta SD

Aunque en el dispositivo final no figure este periférico, he decidido ponerlo porque en un principio iba a estar en este proyecto debido a que se pensó que se necesitaría un almacenamiento relativamente grande para almacenar imágenes o fotogramas de animaciones para la pantalla. Pero luego, por la forma de programar y representar no ha sido necesario.

Aun así, he decidido dejar este código y el periférico conectado por si en un futuro efectúo una mejora en la que me pueda ser de utilidad.

En las imágenes tenemos el aspecto real de este lector de micro SD, en realidad no tiene mucha complejidad, es solo un conector para hacer mas accesibles los contactos y para poder sacar la SD cuando se desee.

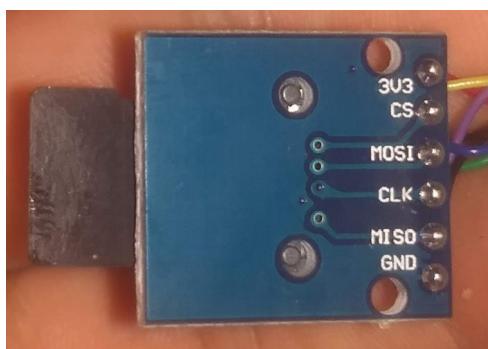


Figura 26: Módulo SD trasera

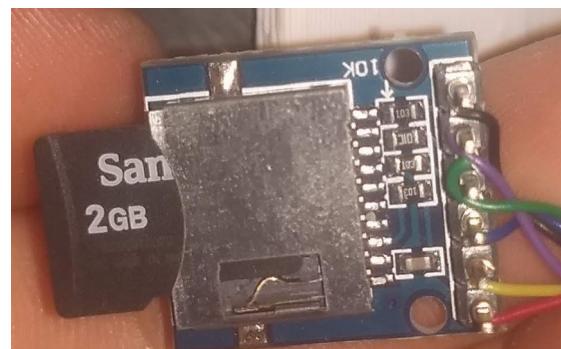


Figura 25: Módulo SD frontal

Como vemos tiene una interfaz SPI la cual se usa para transferir datos entre la tarjeta y el microcontrolador.

Para poder controlarlo nos han hecho falta las librerías SPI.h y SD.h.

Esencialmente, una micro SD y una SD de tamaño completo son iguales, para hacer las interconexiones he seguido el siguiente diagrama:

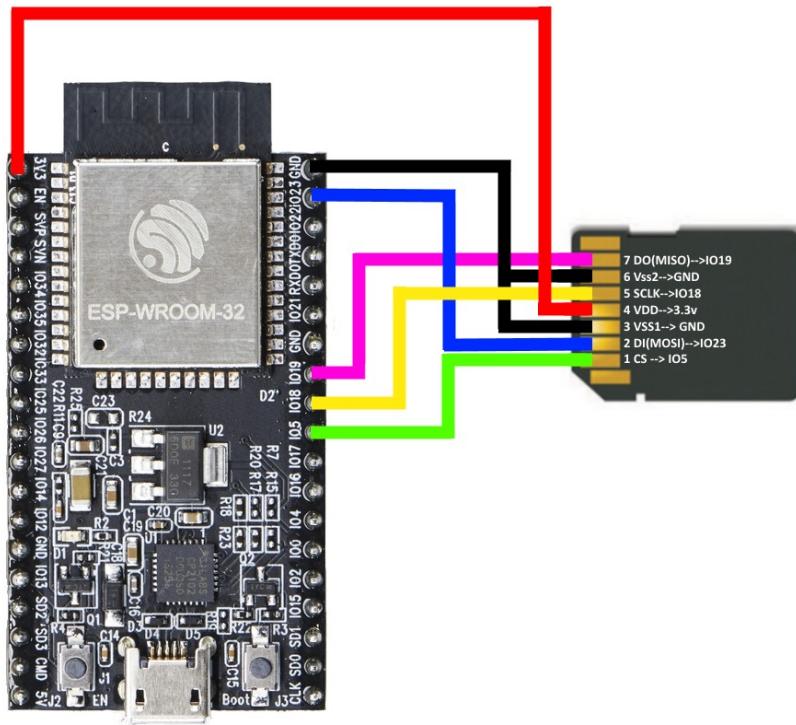


Figura 27: Conexionado SD Fuente: <https://alexlubbock.com/micro-sd-adapter-esp8266-esp32>

Se alimenta

con 3.3v y tenemos los conectores MISO, MOSI y CLK para transferencia de datos, disponemos también de un conector de CS(chip select) que nos permite activar o desactivar el periférico.

La interfaz SPI tiene la ventaja de que nos permite conectar multitud de dispositivos haciendo uso de los mismos pines MISO, MOSI y CLK con el único coste de un pin chip select por dispositivo.

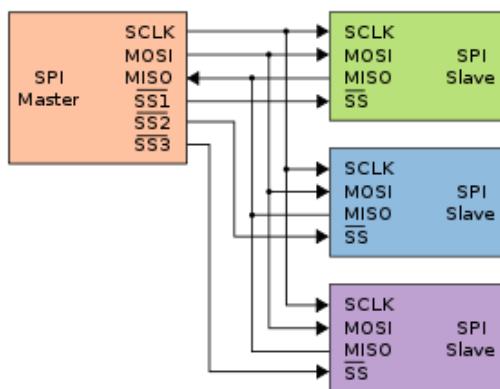


Figura 28: Conexionado SPI Fuente: https://es.wikipedia.org/wiki/Serial_Peripheral_Interface

RTC (Real Time Clock)

Este periférico es una de las partes más relevantes de este proyecto.

El chip contiene un reloj/calendario y 31 bytes de RAM estática. Se comunica con el microprocesador a través de una interfaz serie síncrona simple y es capaz de proveer información sobre los segundos, minutos, horas, día, mes y año.

El calendario es capaz de tener en cuenta la cantidad de días de cada mes a la hora contar un mes y también tiene en cuenta los años bisiestos.

Solamente son necesarios 3 cables para comunicarse con el: CE(chip enable), DAT(Línea de datos entrada/salida) y CLK(reloj). Los datos pueden ser accedidos para lectura o escritura tanto en ráfagas de 31bytes como byte a byte.

Este periférico es capaz de operar bajo condiciones de muy poca potencia y mantener los datos e información del reloj en condiciones con menos de 1 μW .

Aquí tenemos una foto del dispositivo real, como se puede ver tiene un cristal de cuarzo, cumple la función de regular el contador interno, el chip es el *DS1302*. La pila nos asegura su funcionamiento independientemente de la alimentación.



Figura 29: Periférico RTC real

Continuamos viendo como se encuentra interconectado internamente con este diagrama de bloques:

DS1302 BLOCK DIAGRAM Figure 1

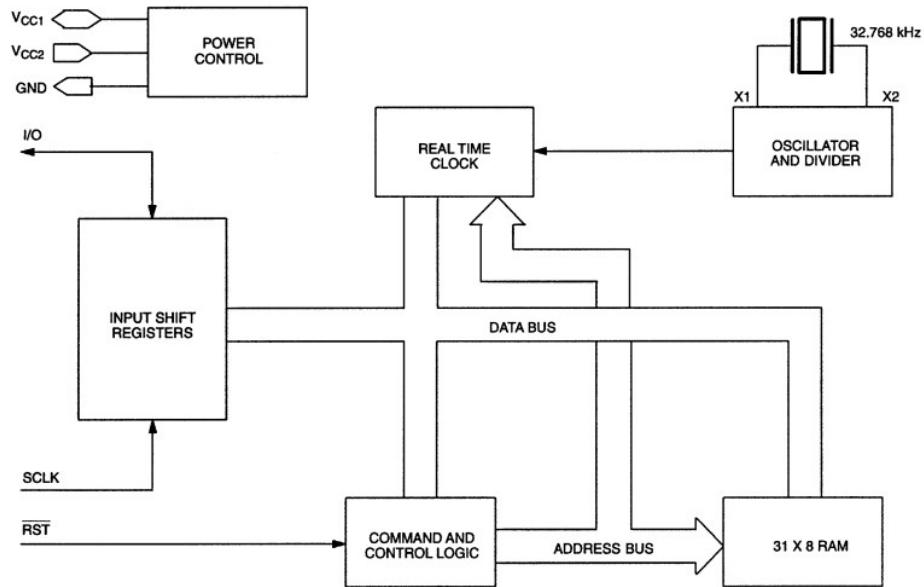


Figura 30: Diagrama de bloques RTC

En la parte superior, en el bloque *power control* podemos destacar las dos entradas de voltaje de las que dispone, este bloque se encarga de administrar la corriente que recibe tanto por una como por la otra entrada y suministrarla a los componentes necesarios.

Tenemos que la interfaz de entrada y de salida (*input shift register*) ha sido implementada con un registro de desplazamiento el cual desplaza los bits que recibe por el bus de datos o la entrada según la señal de reloj.

Vemos que el bus de direcciones de ambas memorias es controlado por el bloque *command and control logic*, el cual es activado con la señal RST y recibe la información para su funcionamiento a través del bus de datos. Veremos más adelante la estructura de datos que necesita para funcionar.

Arriba a la derecha encontramos el contador *divisor y el oscilador*, los cuales se encargan de contar los segundos de forma precisa enviando una señal al bloque del reloj cuando el contador desborda.

¿Pero cómo cuenta exactamente este reloj los segundos?

El cristal oscilador no tiene una frecuencia de 32.768 Hz por casualidad, es el equivalente a 2^{15} , lo que quiere decir que ese número exacto cabría en 15 bits.

Por lo tanto, si en un segundo ocurren 32.768 oscilaciones, basta con utilizar un contador de 15 bits para que cuente todas y llegará al desborde cuando transcurra un segundo.

Esta señal de desborde es la que le pasa al otro módulo del reloj.

Se utiliza el cuarzo porque tiene una precisión temporal aceptable para contar segundos en un reloj.

Proceso de transferencia de datos

Para comenzar una transferencia se debe enviar un primer byte de dirección/comando para indicar que tipo de acceso queremos hacer.

Tiene la siguiente forma:

ADDRESS/COMMAND BYTE Figure 2

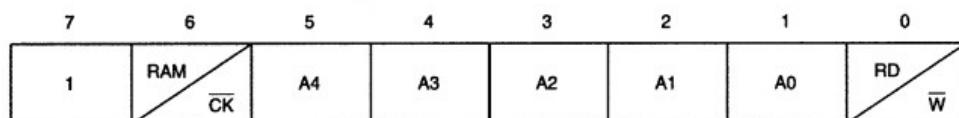


Figura 31: Byte dirección/comando

- El bit 7 debe ser siempre uno, si no, las escrituras al chip quedan desactivadas.
- El bit 6 especifica si acceder al reloj/calendario con 0 o a la ram con un 1.
- Los bits desde el 5 hasta el 1 especifican la dirección a acceder.
- El bit 0 indica que será una operación de escritura si es 0 o de lectura si es 1.

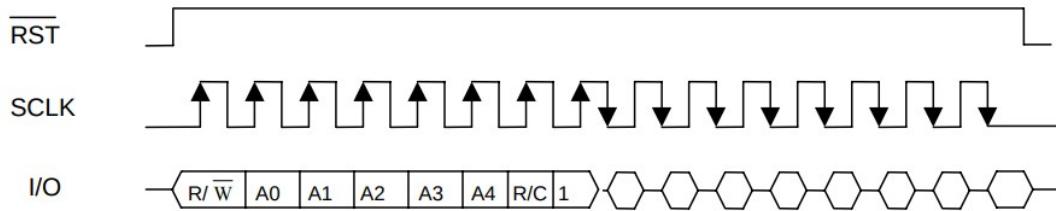
Se comienza a enviar a partir del bit 0.

A continuación veremos una transferencia de lectura y otra de escritura:

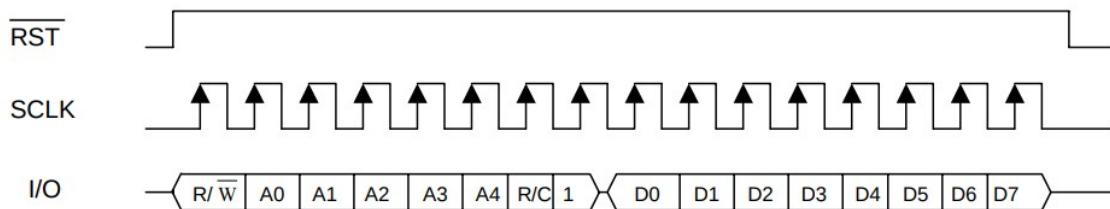
DS1302

DATA TRANSFER SUMMARY Figure 3

SINGLE BYTE READ



SINGLE BYTE WRITE



In burst mode, $\overline{\text{RST}}$ is kept high and additional SCLK cycles are sent until the end of the burst.

Figura 32: Resumen transferencia de datos

Como se puede apreciar primero se envía el byte de control y luego dependiendo de si sea escritura o lectura tomará los datos o los pondrá sobre el cable DAT.

Para usar el modo ráfaga solamente hay que mantener en alto la entrada RST.

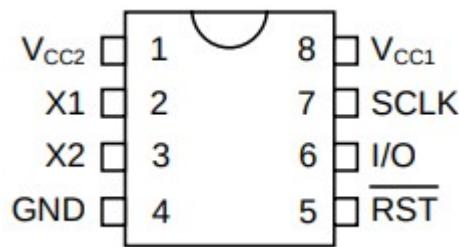


Figura 33: Circuito integrado DS1302

Programación

Para controlarlo nos hemos servido de bibliotecas que implementan este funcionamiento y que ya se encuentran preparadas para ofrecer una forma mas fácil de controlar el periférico, estas son: *ThreeWire.h* y *RtcDS1302.h*

Estas bibliotecas van a replicar el funcionamiento descrito anteriormente y nos van a ahorrar el tener que programarlo.

Primero se tiene que crear un objeto de la clase *RtcDS1302<ThreeWire>* que llamaremos «*rtc*» y podemos inicializar el reloj con la función *rtc.begin()*.

Se hacen además en la inicialización otra serie de comprobaciones como por ejemplo que no se encuentre en protección de escritura.

Para leerlo o cargar datos hay que utilizar una estructura de datos que se llama *RtcDateTime* y nos serviremos de las funciones *GetRtcDateTime()* y *SetDateTime()*.

Hemos creado funciones para inicializarlo, cargar una fecha, leer y para imprimir por el puerto serie su valor actual.

```
void inicializaReloj(RtcDS1302<ThreeWire> Rtc);
void cargaFecha(RtcDS1302<ThreeWire> Rtc);
RtcDateTime obtenerFechaRTC(RtcDS1302<ThreeWire> Rtc);
void imprimirFechaRTC(RtcDateTime dt);
```

Figura 34: Funciones del RTC

Información obtenida del datasheet en la web: <https://pdf1.alldatasheet.com/datasheet-pdf/view/447115/TGS/DS1302.html>

Humedad y temperatura

Este resulta ser un sensor que ofrece unas buenas prestaciones en cuanto a precisión por el precio que tiene, $\pm 0,3^{\circ}\text{C}$ para el caso de la temperatura y $\pm 2\%$ para la humedad.

Aun así cabe destacar que las prestaciones del sensor no son constantes y se incrementa el error conforme nos acercamos a los límites de medición del sensor.

Línea continua: Valor típico

Línea discontinua: Máximo

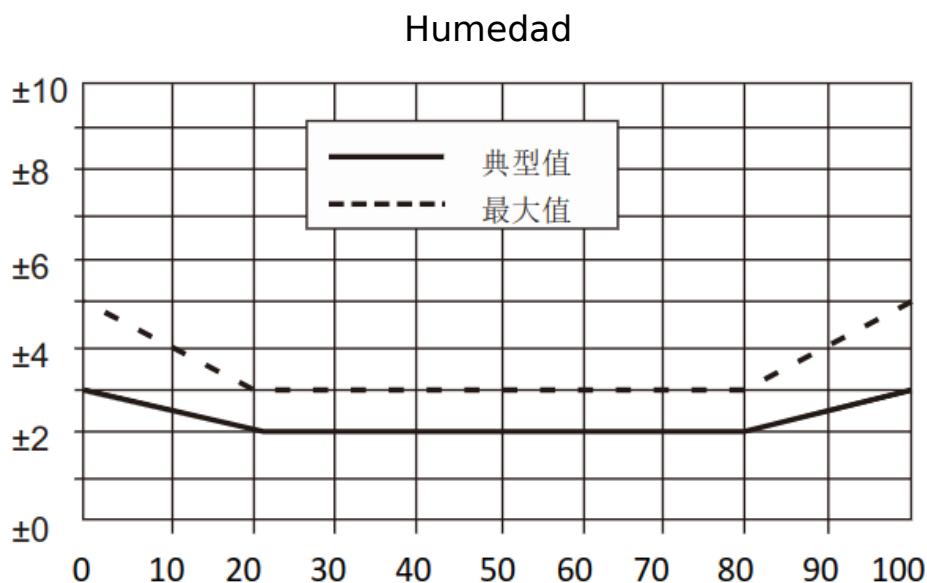


Figura 35: Gráfico error en humedad

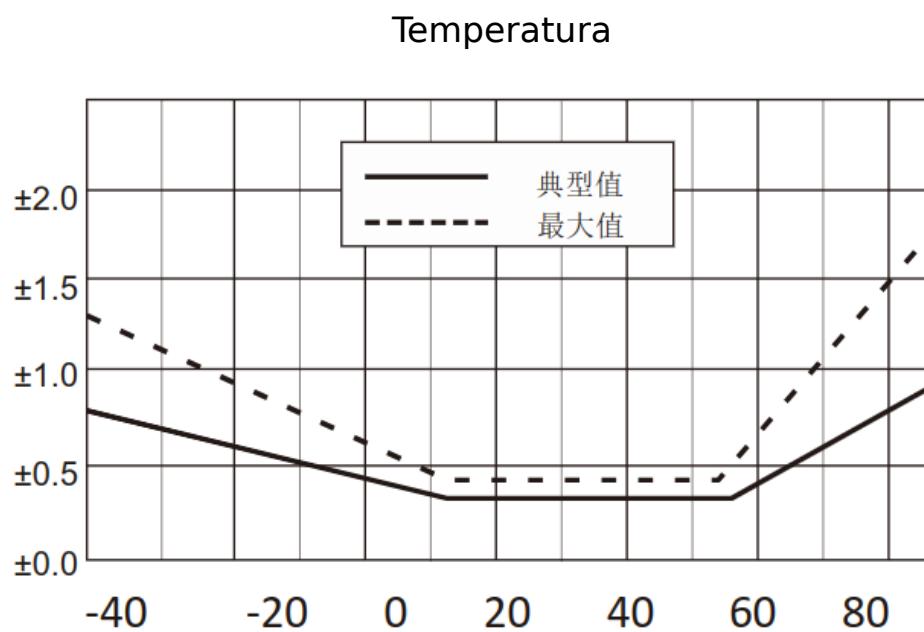


Figura 36: Gráfico error en temperatura

En estas imágenes de mi unidad podemos ver las conexiones que usa el sensor, se puede observar que el dispositivo se comunica por un puerto I2C.

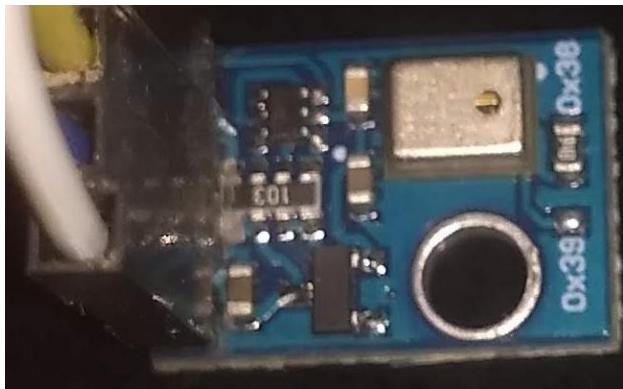


Figura 38: Sensor AHT10 real frontal

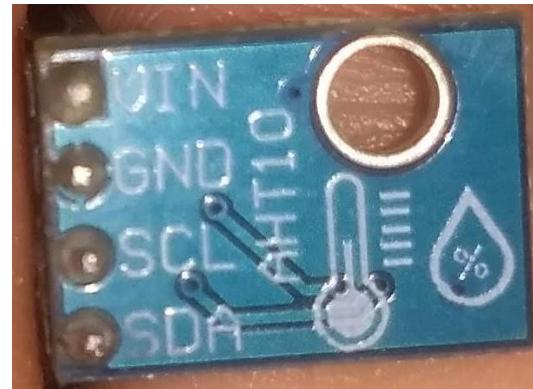


Figura 37: Sensor AHT10 real trasera

Vemos que en este tipo de sensores y periféricos muchos cuentan con esta interfaz debido a que es un estándar y permite conectar muchos dispositivos con solamente dos cables.

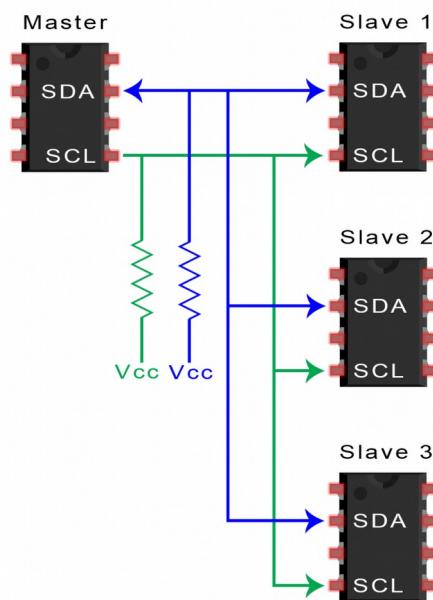


Figura 39: Comunicación I2C

Las dos resistencias que se ven en la imagen son resistencias de pull-up y mantienen las líneas en alto como estado por defecto.

Estas resistencias de pull-up no serán necesarias en este proyecto debido a que el microcontrolador que se va a utilizar tiene la capacidad de hacer pull-up o pull-down internamente desde sus pines.

Cada dispositivo conectado a las líneas SDA y SCL tienen una dirección que les identifica dentro de la red, en este caso la dirección de este módulo es la 0x38.

Cuando se comienza una conexión se indica primero la dirección del dispositivo al que te quieres dirigir, entonces los que no tengan esa dirección se ponen en alta impedancia y el elegido puede continuar con la comunicación.

Una comunicación de este tipo tiene la siguiente estructura:

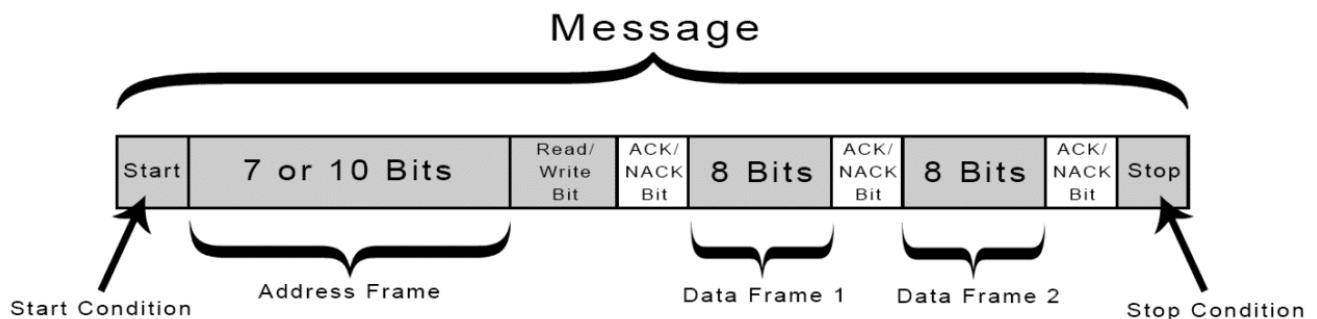


Figura 40: Comunicación I2C

Primero se tiene que dar una *condición de comienzo*, en la que la línea SDA cambia desde un valor alto de voltaje a uno bajo antes de que SCL cambie de alto a bajo.



Figura 41: Condición de comienzo

Tras esto los dispositivos de la red se ponen a la escucha y se inicia la transmisión de la dirección a la que se quiere acceder.

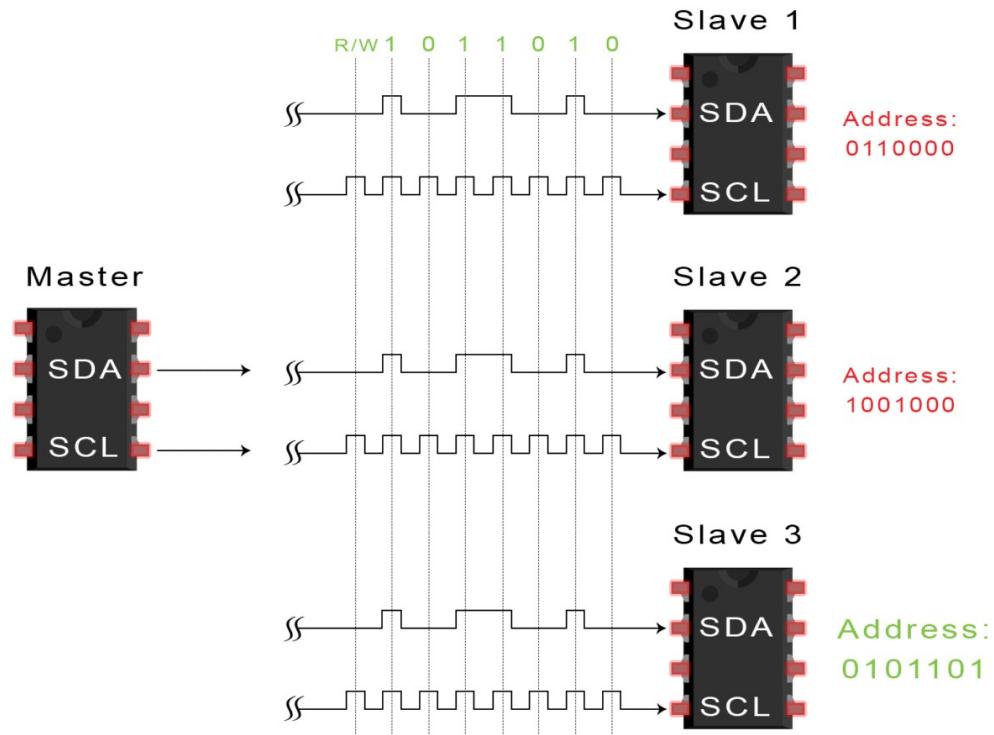


Figura 42: Comprobación de dirección

En esta primera transmisión también se envía el bit de lectura/escritura, es el que indica si se desea leer o escribir, completando el primer byte.

A esto el periférico que se por aludido debe responder bajando la línea SDA

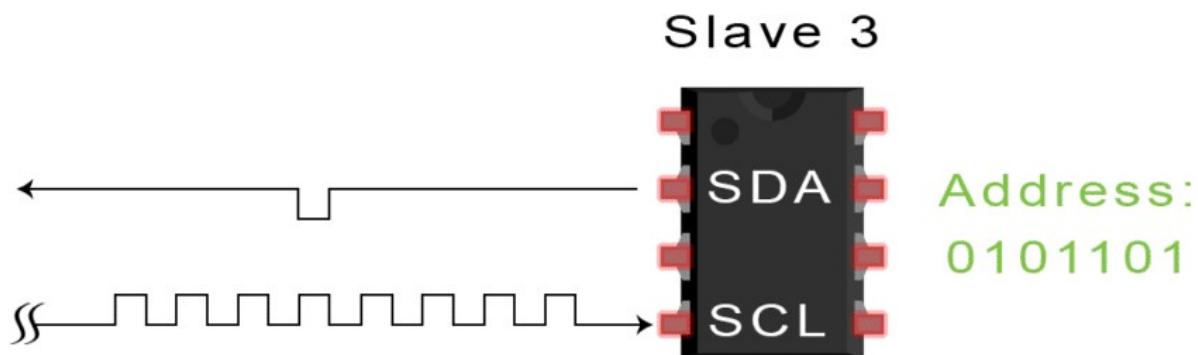


Figura 43: Confirmación

Ahora se puede iniciar la transferencia; por cada byte transmitido, el que reciba la información debe responder de la misma forma, bajando SDA como señal de confirmación de recepción de ese byte.

Una vez se ha acabado de enviar, el emisor deberá emitir la condición de parada, esto es cambiar la línea SDA desde un nivel bajo de voltaje a alto después de que SCL haga lo mismo.

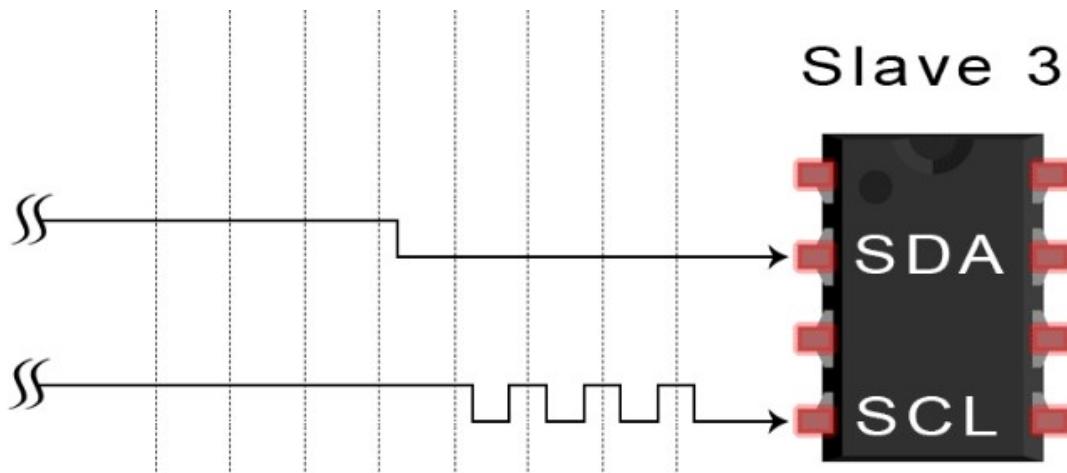


Figura 44: Condición de parada

(Recordemos que como tenemos una resistencia pull-up el valor 0 es la línea en alto)

Información sobre I2C e imágenes extraídas de: <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/#:~:text=I2C%20is%20a%20serial%20communication,the%20master%20and%20the%20slave.>

Una vez aclarada como es esta transferencia entre nuestro sensor y el microcontrolador con el protocolo I2C vamos a ver como se controla en nuestro caso el sensor.

Primero, enviamos la dirección 0x38 mas un bit 0 que indica escritura.

Tras recibir la primera confirmación por parte del dispositivo que se da por aludido se debe enviar un byte de comando.

Los bytes de comando disponibles son los siguientes:

command	Interpretation	Code
Initialization command	Keep the host	1110'0001
Trigger measurement	Keep the host	1010'1100
Soft reset		1011'1010

Figura 45: Tabla de comandos disponibles

Primeramente debemos enviar un comando de inicialización y luego uno de activar medición. (Cada uno con su nueva conexión)

En la siguiente imagen tenemos un ejemplo de envío de byte de medición

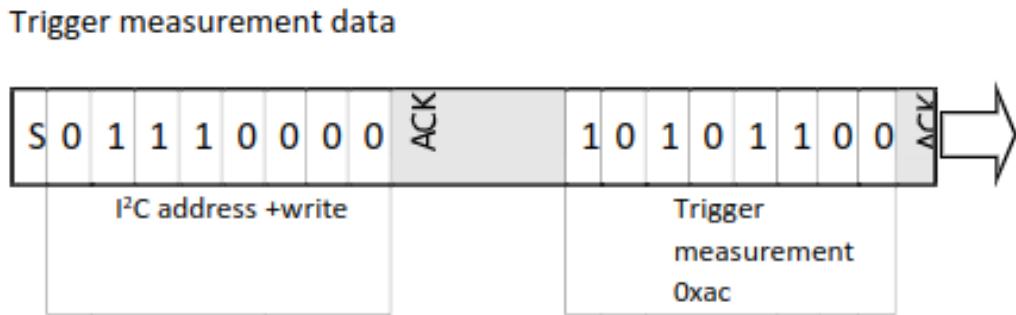


Figura 46: Envío de byte de medición

Tras enviar este último se cierra la conexión y se espera a que se tome la medición(unos 75ms). Tras este tiempo se vuelve a iniciar una conexión, esta vez de lectura, en la que el sensor devolverá un byte de estado, y, si están disponibles, todos los datos.

En la siguiente tabla se muestra la estructura y el significado de las respuestas que podemos obtener. Es aquí donde nos podemos fijar en si el dispositivo aún sigue midiendo.

Bit	significance	description
Bit[7]	Busy indication	1 -- The device is busy, 0 -- the device is in the measurement state, and the device is idle.
Bit[6:5]	Current working mode (Mode Status)	00 Currently NOR mode 01 Currently CYC mode 1x Currently CMD mode
Bit[4]	Reserved	Reserved
Bit[3]	Calibration enable bit CAL Enable	1-calibrated 0-- not calibrated

Figura 47: Tabla de respuestas del sensor

Por aquí tenemos un gráfico de ejemplo de como sería una respuesta, cada dato ocupa 20 bits. (Los caracteres en chino son la respuesta de estado)

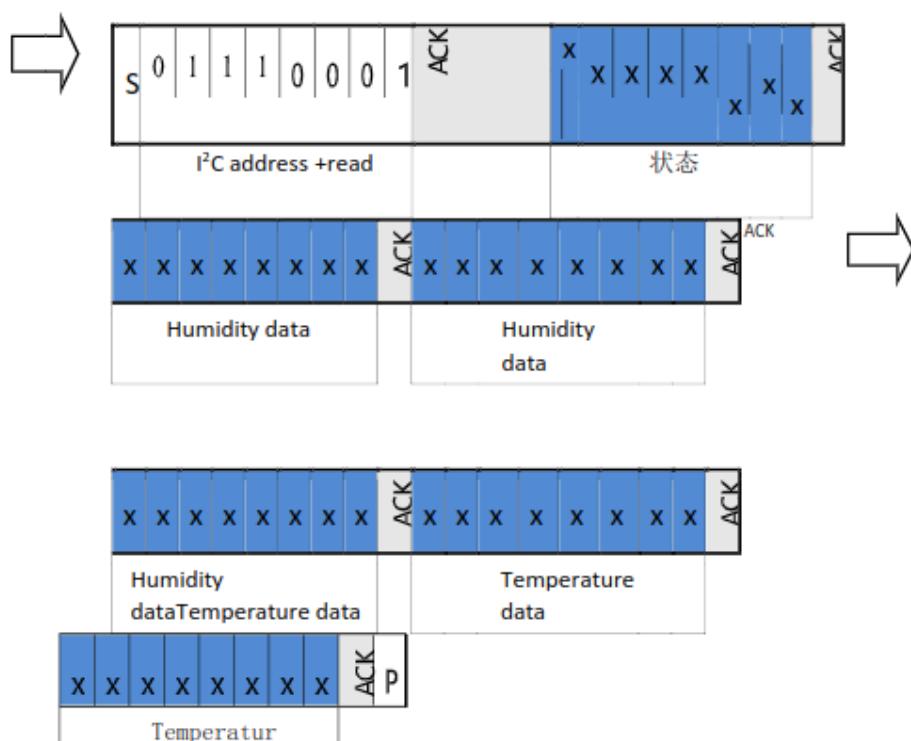


Figura 48: Respuesta con los datos de medición

Programación

Para este sensor me costó encontrar las bibliotecas adecuadas, por ser confusas o por no estar adaptadas al *esp32*.

La versión inglesa del datasheet del AHT10 está parcialmente en chino.

Al final, después de buscar, encontré una llamada *AHT10.h* la cual tiene dependencia de la biblioteca *Wire.h* y permite de una forma muy directa acceder a los datos del sensor.

Es tan fácil como crear un objeto de la clase *AHT10*, por ejemplo uno llamado *myAHT10* e inicializarlo con la función *myAHT10.begin()*. Para leer tenemos las funciones *readTemperature()* y *readHumidity()*.

Así se han creado funciones para inicializar, leer temperatura y leer humedad para de esta forma tener todas las funciones en el mismo archivo.

```
//==TEMPERATURA Y HUMEDAD==  
#include <AHT10.h>  
#include <Wire.h>  
void inicializaHumTemp(AHT10 myAHT10);  
float obtenerTemperatura(AHT10 myAHT10);      //Precisión temperatura +- 0.3°C  
float obtenerHumedad(AHT10 myAHT10);           //Precisión humedad +- 2%
```

Figura 49: Funciones para el sensor AHT10

Sensor de presencia (PIR)

Por sus siglas, en inglés, Passive Infrared Radiation.

Este sensor tiene un aspecto muy característico, tiene forma semiesférica, este elemento de plástico se encuentra formado internamente por lentes fresnel, las cuales dividen el rango de visión en zonas y enfoca los rayos de infrarrojos directamente hacia cada uno de los campos del sensor PIR, esto es lo que le dota de su amplio rango de visión de 110º.

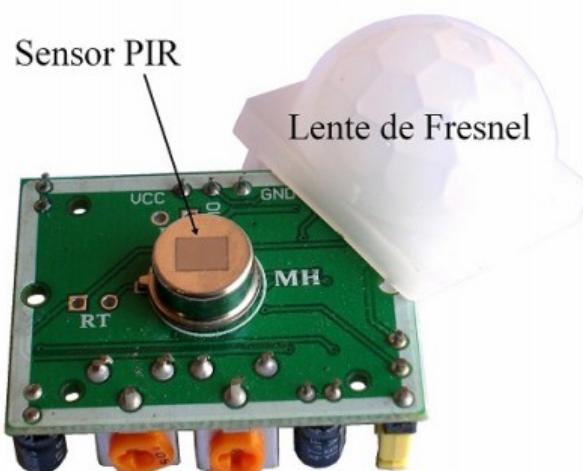


Figura 50: Sensor PIR

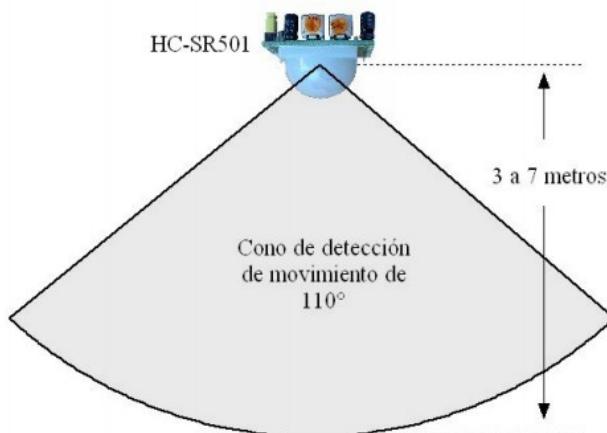


Figura 51: Campo de visión

Todos los cuerpos, cuanto más calientes están, mas radiación infrarroja emitirán, como todos los seres vivos de sangre caliente emitimos radiación infrarroja, esta radiación puede ser captada por el sensor y ser convertida en un impulso eléctrico.

Cada sensor posee dos campos y un circuito que se encarga de calcular la diferencia entre ellos. Solamente cuando existe una diferencia en el estado de excitación de ambos campos se considera que ha detectado un objeto.

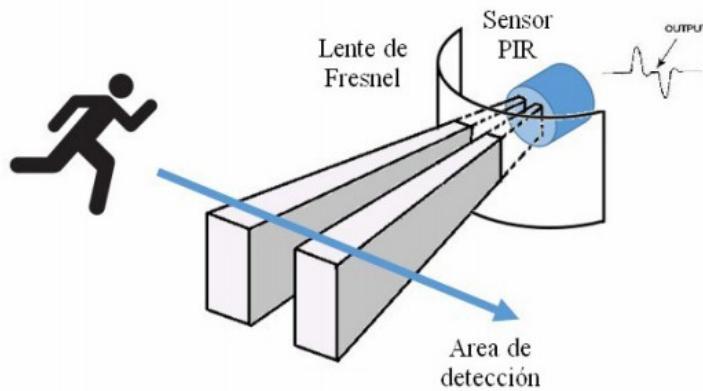


Figura 52: Detección infrarroja

Este sensor tiene un bajo coste, es preciso y consume muy poca energía.

Tiene una serie de controles para decidir el tipo funcionamiento que se quiere obtener de el.

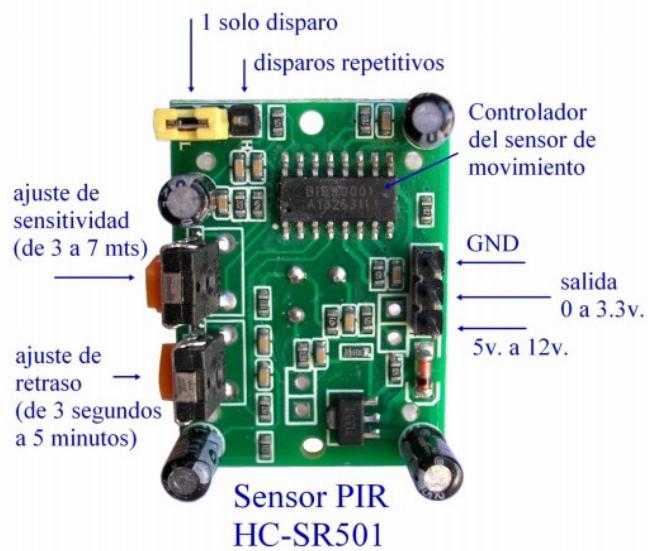


Figura 53: Controles del sensor PIR

La salida resultante de este dispositivo vendrá dada como una señal digital que oscilará entre los 0 y 3.3v, indicando con 0 ninguna detección y con 3.3v que se está detectando un movimiento o actividad en el sensor.

La configuración que se ha elegido para nuestra aplicación es el modo de disparo repetitivo para poder obtener un mayor tiempo de respuesta para nuestro prototipo detectando cambios mas rápidamente.

En cuanto al retraso, se ha puesto en 3 segundos y la sensibilidad en 7 metros, suficiente para detectar a alguien que entre en la sala de una manera eficiente y rápida.

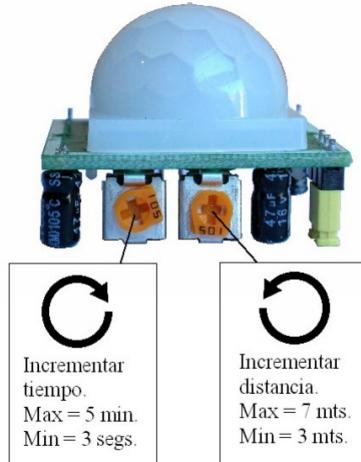


Figura 54: Potenciómetros del sensor PIR

Fuente de imágenes e información: <https://puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf>

Programación

En lo relativo a como ha sido implantando en el código esta señal, se ha modelado como una señal de interrupción externa recibida por el coprocesador cuando se encuentra en modo deep sleep que despierta al procesador principal cuando la detecta.

Pantalla LED RGB 16x16

Cuando se ha tratado de encontrar una pantalla adecuada para este proyecto, se barajó de entre las posibilidades utilizar matrices LED monocolor sin memoria, lo que implicaba además de utilizar circuitería externa como registros de desplazamiento un uso intensivo de la cpu para encargarse de realizar la actualización de la pantalla.

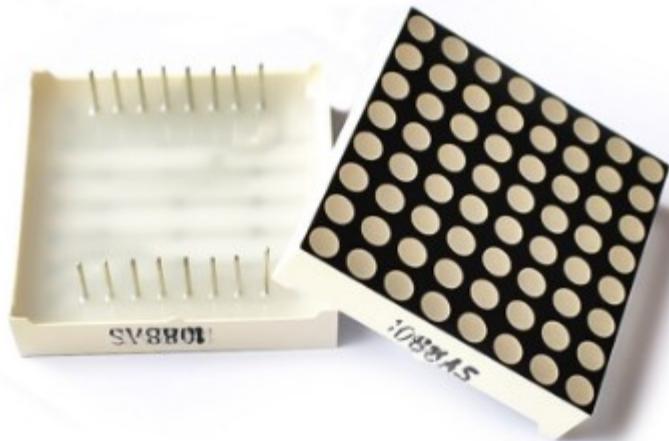


Figura 55: Matriz led rojo 8x8

Uniendo 4 de estos módulos se podría crear una pantalla de 16x16.

Aunque son módulos baratos, el esfuerzo que se requiere para hacerlos funcionar y que solo tenga un color me hizo descartarlos.

Tras una sesión con el tutor para preguntarle dudas me recomendó usar un panel led que es RGB y además sus píxeles mantienen el color y la intensidad incluso si se ha dejado de enviar información a los mismos.

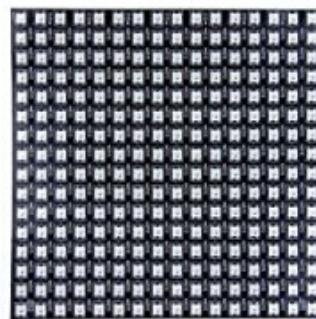
Estos se conocen como addressable led matrix.

En el mercado español este tipo de paneles no son especialmente baratos, sobretodo en el caso de una matriz de 16x16.

A la derecha tenemos una imagen con su precio a fecha 6/9/2020 en la tienda de Amazon.

Este precio se reduce a mas de la mitad si lo pedimos a una tienda china como AliExpress montando los mismos componentes.

Todo esto lo discutiremos mas adelante en el presupuesto cuando veamos el coste total de este proyecto.



ALITOVE For Arduino WS2812B LED Rainbow Matrix 16x16 256 Pixels led flexible panel screen 5050 RGB SMD Dream color array of led pixel

3

28,69€

Figura 56: Panel en venta en amazon.com

Una vez tenemos el panel en nuestras manos podemos continuar analizando como se encuentra hecho realmente.

Cada uno de los píxeles que en él se encuentran tiene su propio circuito integrado que se encarga de traducir la codificación de colores y retenerla para emitir su luz mientras permanezca conectado.

En este caso tenemos un tipo de LED al que se le conoce como NeoPixel por la empresa Adafruit. Aquí tenemos una imagen de su página de venta:

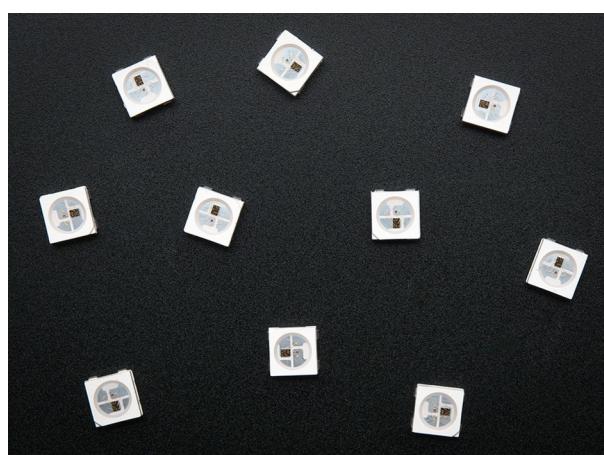


Figura 57: Leds NeoPixel Fuente: <https://www.adafruit.com/product/1655>

Estos leds son relativamente grandes si se comparan con los que tendíamos en una pantalla común como la de un ordenador, sin embargo, estos pueden dar mucha mas luz.

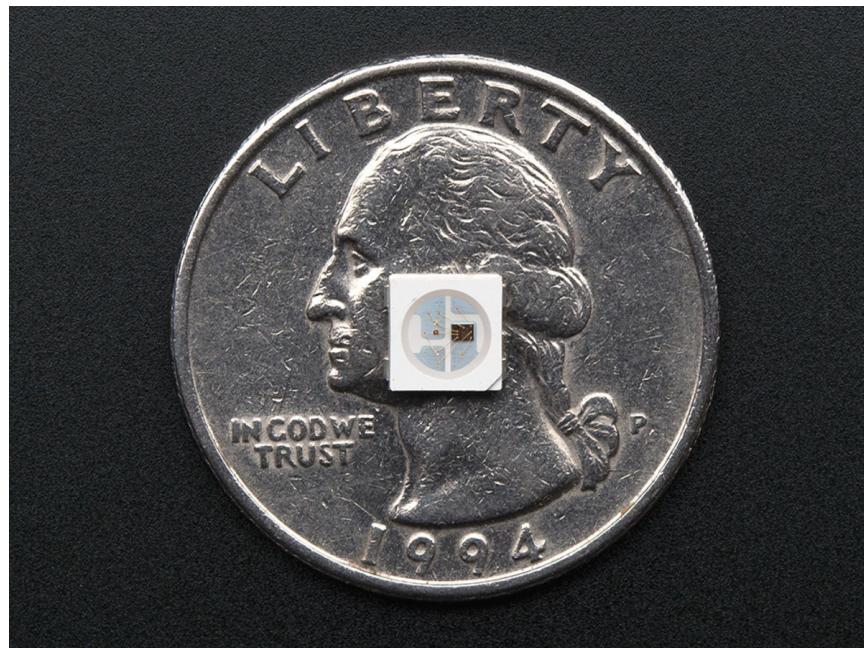


Figura 58: Tamaño led NeoPixel Fuente: <https://www.adafruit.com/product/1655>

Cada uno de estos NeoPixel se componen a su vez de 3 diodos (rojo, verde y azul) y un chip integrado.

Regulando la intensidad de brillo de cada uno de los diodos led se puede conseguir todo el espectro de colores, siendo el blanco todos los leds encendidos a la misma intensidad.

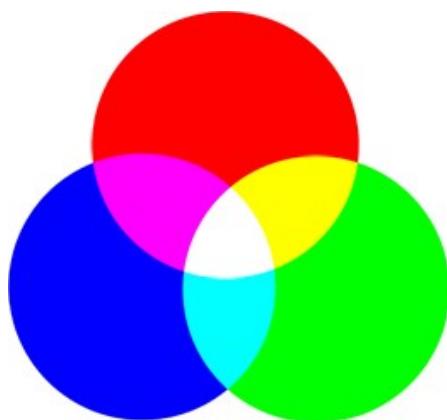


Figura 59: Combinación de colores aditiva Fuente: <https://mundotextilmag.com.ar/color-y-colorantes/>

Tienen la particularidad de que pueden ser encadenados uno tras otro, esto lo consiguen teniendo un conector de entrada y otro de salida, en cuanto un pixel recibe un nuevo color, este transmite la información que tenía almacenada por su salida, la cual está conectada a la entrada del siguiente led, y así consecutivamente.

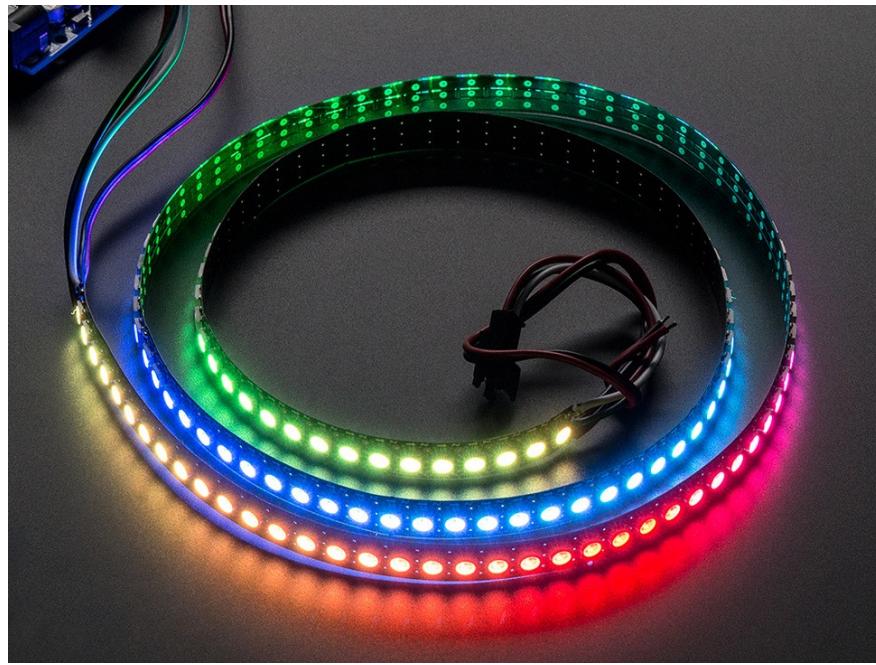


Figura 60: Tira de leds encadenados Fuente: <https://www.adafruit.com/product/1506>

De esta forma, enviando la información de todos los píxeles que contiene la tira al primer LED, conseguiremos que todos acaben obteniendo así su color correspondiente.

Ahora habiendo explicado esto veamos como se forma una matriz LED.

La matriz en cuestión que vamos a controlar, aunque tenga una forma cuadrada, internamente se encuentra conexionada como una tira led dispuesta en forma de zig-zag y posee dos circuitos diferenciados para su control: uno de potencia que va directamente a la fuente de 5v y otro para la señal. También tiene otra salida, en la que podríamos conectar otra pantalla.

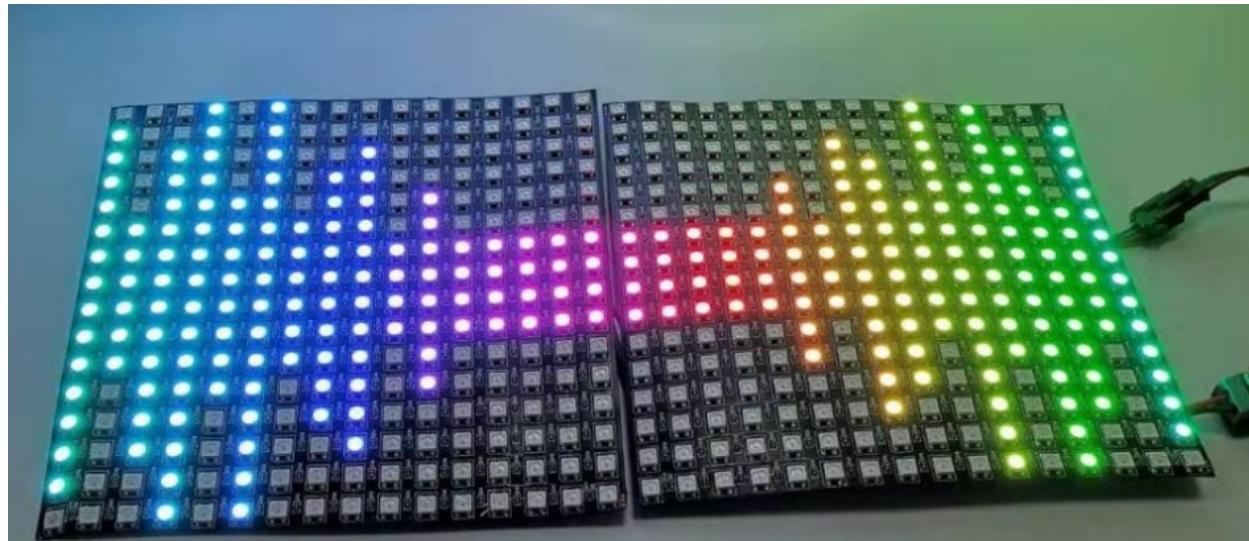


Figura 61: Imagen del vendedor del que fue adquirido (Enlace ya no disponible)

En la siguiente imagen tenemos representado de forma gráfica como se disponen con una tabla que realicé para representar la numeración de los píxeles.

0	31	32	63	64	95	96	127	128	159	160	191	192	223	224	255
1	30	33	62	65	94	97	126	129	158	161	190	193	222	225	254
2	29	34	61	66	93	98	125	130	157	162	189	194	221	226	253
3	28	35	60	67	92	99	124	131	156	163	188	195	220	227	252
4	27	36	59	68	91	100	123	132	155	164	187	196	219	228	251
5	26	37	58	69	90	101	122	133	154	165	186	197	218	229	250
6	25	38	57	70	89	102	121	134	153	166	185	198	217	230	249
7	24	39	56	71	88	103	120	135	152	167	184	199	216	231	248
8	23	40	55	72	87	104	119	136	151	168	183	200	215	232	247
9	22	41	54	73	86	105	118	137	150	169	182	201	214	233	246
10	21	42	53	74	85	106	117	138	149	170	181	202	213	234	245
11	20	43	52	75	84	107	116	139	148	171	180	203	212	235	244
12	19	44	51	76	83	108	115	140	147	172	179	204	211	236	243
13	18	45	50	77	82	109	114	141	146	173	178	205	210	237	242
14	17	46	49	78	81	110	113	142	145	174	177	206	209	238	241
15	16	47	48	79	80	111	112	143	144	175	176	207	208	239	240

Figura 62: Ordenación de leds en la matriz

A esta forma de conectar en serie los píxeles se le llama conexión en cascada:

Cascade method:

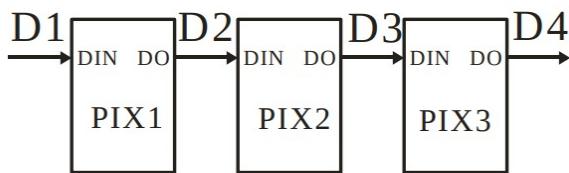


Figura 64: Leds en cascada

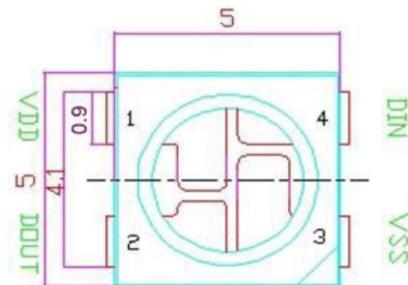


Figura 63: Led NeoPixel, terminales

La transmisión a estos LEDs de la información acerca del color que deseamos es muy sencilla.

Tenemos que enviar en total 24 bits, 1 byte por cada color. Estos no se envían en un orden RGB sino en GRB(Green, Red, Blue).

Aquí tenemos una imagen en la que se aprecian los 3 bytes y como deben ser enviados:

Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Note: Follow the order of GRB to sent data and the high bit sent at first.

Figura 65: Estructura de envío de bits al led

La codificación que se sigue para el envío de bits es una codificación sencilla, por pulsos:

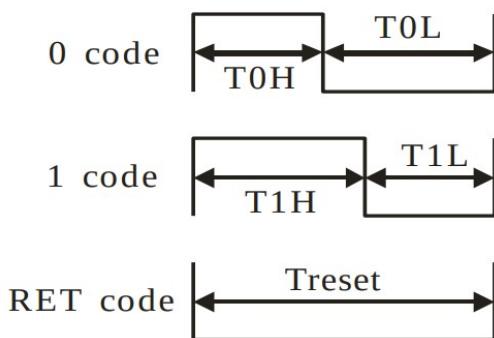


Figura 66: Codificación por pulsos

Un pulso corto es un cero, uno largo es un 1 y si se permanece durante un tiempo en bajo hace un reset(se pone a 0). Tiempos:

Data transfer time(TH+TL=1.25μs±600ns)

T0H	0 code ,high voltage time	0.4us	±150ns
T1H	1 code ,high voltage time	0.8us	±150ns
T0L	0 code , low voltage time	0.85us	±150ns
T1L	1 code ,low voltage time	0.45us	±150ns
RES	low voltage time	Above 50μs	

Figura 67: Tiempos para la transferencia

Imágenes e información: <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf>

Ya conociendo las características técnicas del panel, la siguiente parte continuará en la parte de montaje, en la que explicaremos como se ha creado la pantalla final.

Programación

Para lograr que funcionara, la propia Adafruit tiene unas bibliotecas que permiten controlar estos paneles sin “muchas complicaciones”.

Y pongo esto último entre comillas porque aunque intenté muchas combinaciones diferentes no logré hacer que funcionara con mi panel, ni cambiando el modelo en el código ni cambiando el cableado.

Tras buscar bibliotecas alternativas que fueran mas sencillas encontré una que se llama FastLED.h, la cual me permitió poner mi modelo exacto de pixel, el WS2812B.

Para inicializarlo solamente hay que utilizar la siguiente función:

```
FastLED.addLeds<WS2812B, PANTPIN, GRB>(leds, NUMPIXELS);
```

WS2312B es el modelo, PANTPIN el número de GPIO(pin), y GRB el orden de envío de los bytes. Entre parámetros se le pasa un puntero, que en este caso es leds y NUMPIXELS es el número de píxeles que tiene la tira. (Recordemos que se encuentra interconectado como tal)

Existen funciones muy fáciles de usar como `FastLED.setBrightness(brillo)`

que nos permite definir el brillo que queremos para los leds a partir de este momento.

Para colorear un pixel tenemos que hacer referencia a su posición dentro de el puntero a los pixeles. Ejemplo: `pixeles[posicion]`.

Y el valor a asignarle debe ser un color en formato HTML. Ej: `0x2020C0`

Todo esto no serviría de nada sin el comando `FastLED.show()`, que envía todos los valores que tenemos almacenados en el puntero de leds a la pantalla.

Como esta biblioteca es solo de control tenemos un problema a la hora de mover un píxel, no tenemos una abstracción que lo simplifique.

Esto es que debido a la peculiar disposición de los píxeles, mover por ejemplo un pixel a la derecha no es trivial, se necesita hacer un cálculo previo para saber en que posición debe de ir para que parezca que solamente se ha movido una posición a la derecha.

La solución a este problema es la que se ha abordado con la función:

```
int desplazaPixel(int posicion, int xOffs, int yOffs)
```

La cual dada la posición de un pixel y un desplazamiento en ejes cartesianos X e Y nos devuelve la posición que debería ocupar dentro de la tira LED.

Esto lo hace de la siguiente manera:

Para calcular el desplazamiento horizontal

Si el pixel a mover se encuentra en una columna par, es decir, si se cumple que $(\text{posicion} / 16) \% 2 == 0$.

Si se quiere desplazar a la derecha

Se ha de contar la distancia hacia el límite inferior(porque es lo que obtenemos usando el módulo), multiplicarlo por 2(para contar el cambio de sentido) y restárselo a 31(para contar hacia arriba y no hacia abajo, invertirlo)

$$31 - 2 * (\text{posicion \% 16})$$

0	31	32	63	64	95	96	127	128	159	160	191	192	223	224	255
1	30	33	62	65	94	97	126	129	158	161	190	193	222	225	254
2	29	34	61	66	93	98	125	130	157	162	189	194	221	226	253
3	28	35	60	67	92	99	124	131	156	163	188	195	220	227	252
4	27	36	59	68	91	100	123	132	155	164	187	196	219	228	251
5	26	37	58	69	90	101	122	133	154	165	186	197	218	229	250
6	25	38	57	70	89	102	121	134	153	166	185	198	217	230	249
7	24	39	56	71	88	103	120	135	152	167	184	199	216	231	248
8	23	40	55	72	87	104	119	136	151	168	183	200	215	232	247
9	22	41	54	73	86	105	118	137	150	169	182	201	214	233	246
10	21	42	53	74	85	106	117	138	149	170	181	202	213	234	245
11	20	43	52	75	84	107	116	139	148	171	180	203	212	235	244
12	19	44	51	76	83	108	115	140	147	172	179	204	211	236	243
13	18	45	50	77	82	109	114	141	146	173	178	205	210	237	242
14	17	46	49	78	81	110	113	142	145	174	177	206	209	238	241
15	16	47	48	79	80	111	112	143	144	175	176	207	208	239	240

Figura 68: Desplazamiento a la derecha, columna par

Si se quiere desplazar a la izquierda

Tenemos la misma operación que la anterior pero sin invertir al final(restando a 31), esta vez se resta a 1(para contar solo los huecos a avanzar)

$$1 + 2 * (\text{posicion \% 16})$$

0	31	32	63	64	95	96	127	128	159	160	191	192	223	224	255
1	30	33	62	65	94	97	126	129	158	161	190	193	222	225	254
2	29	34	61	66	93	98	125	130	157	162	189	194	221	226	253
3	28	35	60	67	92	99	124	131	156	163	188	195	220	227	252
4	27	36	59	68	91	100	123	132	155	164	187	196	219	228	251
5	26	37	58	69	90	101	122	133	154	165	186	197	218	229	250
6	25	38	57	70	89	102	121	134	153	166	185	198	217	230	249
7	24	39	56	71	88	103	120	135	152	167	184	199	216	231	248
8	23	40	55	72	87	104	119	136	151	168	183	200	215	232	247
9	22	41	54	73	86	105	118	137	150	169	182	201	214	233	246
10	21	42	53	74	85	106	117	138	149	170	181	202	213	234	245
11	20	43	52	75	84	107	116	139	148	171	180	203	212	235	244
12	19	44	51	76	83	108	115	140	147	172	179	204	211	236	243
13	18	45	50	77	82	109	114	141	146	173	178	205	210	237	242
14	17	46	49	78	81	110	113	142	145	174	177	206	209	238	241
15	16	47	48	79	80	111	112	143	144	175	176	207	208	239	240

Figura 69: Desplazamiento a la izquierda, columna par

Si se hubiera encontrado en una columna impar

Si se quiere desplazar a la derecha

Ahora haciendo el módulo de 16 nos dará la distancia que hay hacia el límite superior, por lo que si a 15 le restamos esta distancia obtenemos la distancia al inferior, tal como antes.

La operación es la misma que si moviéramos a la izquierda en una columna par.

$$1 + 2 * (15 - (\text{posicion \% } 16))$$

0	31	32	63	64	95	96	127	128	159	160	191	192	223	224	255
1	30	33	62	65	94	97	126	129	158	161	190	193	222	225	254
2	29	34	61	66	93	98	125	130	157	162	189	194	221	226	253
3	28	35	60	67	92	99	124	131	156	163	188	195	220	227	252
4	27	36	59	68	91	100	123	132	155	164	187	196	219	228	251
5	26	37	58	69	90	101	122	133	154	165	186	197	218	229	250
6	25	38	57	70	89	102	121	134	153	166	185	198	217	230	249
7	24	39	56	71	88	103	120	135	152	167	184	199	216	231	248
8	23	40	55	72	87	104	119	136	151	168	183	200	215	232	247
9	22	41	54	73	86	105	118	137	150	169	182	201	214	233	246
10	21	42	53	74	85	106	117	138	149	170	181	202	213	234	245
11	20	43	52	75	84	107	116	139	148	171	180	203	212	235	244
12	19	44	51	76	83	108	115	140	147	172	179	204	211	236	243
13	18	45	50	77	82	109	114	141	146	173	178	205	210	237	242
14	17	46	49	78	81	110	113	142	145	174	177	206	209	238	241
15	16	47	48	79	80	111	112	143	144	175	176	207	208	239	240

Figura 70: Desplazamiento a la derecha, columna impar

Si se quiere desplazar a la izquierda

De la misma forma se invierte a una distancia hasta el límite inferior y se hace la misma operación que si desplazáramos a la derecha en una columna par.

$$1 + 2 * (15 - (\text{posicion \% } 16))$$

0	31	32	63	64	95	96	127	128	159	160	191	192	223	224	255
1	30	33	62	65	94	97	126	129	158	161	190	193	222	225	254
2	29	34	61	66	93	98	125	130	157	162	189	194	221	226	253
3	28	35	60	67	92	99	124	131	156	163	188	195	220	227	252
4	27	36	59	68	91	100	123	132	155	164	187	196	219	228	251
5	26	37	58	69	90	101	122	133	154	165	186	197	218	229	250
6	25	38	57	70	89	102	121	134	153	166	185	198	217	230	249
7	24	39	56	71	88	103	120	135	152	167	184	199	216	231	248
8	23	40	55	72	87	104	119	136	151	168	183	200	215	232	247
9	22	41	54	73	86	105	118	137	150	169	182	201	214	233	246
10	21	42	53	74	85	106	117	138	149	170	181	202	213	234	245
11	20	43	52	75	84	107	116	139	148	171	180	203	212	235	244
12	19	44	51	76	83	108	115	140	147	172	179	204	211	236	243
13	18	45	50	77	82	109	114	141	146	173	178	205	210	237	242
14	17	46	49	78	81	110	113	142	145	174	177	206	209	238	241
15	16	47	48	79	80	111	112	143	144	175	176	207	208	239	240

Figura 71: Desplazamiento a la izquierda, columna impar

Ya por último para el desplazamiento vertical es tan directo como que si se encuentra en columna par se le resta el desplazamiento vertical y si está en la impar se le resta.

Se han creado además otras funciones que nos permiten representar un icono, número o carácter especial en cualquier parte de la pantalla haciendo uso de esta última función que acabamos de describir.

Podemos poner como ejemplo la función:

```
void pintarNUM(int num, CRGB color, CRGB *leds, int xOffs, int yOffs)
```

La cual recibiendo el número que queremos pintar, el color del que lo queremos y el puntero a los leds de la pantalla colocará la información necesaria en la parte de la ram reservada al puntero. Además tenemos un desplazamiento en X y otro en Y ya que todos los números se encuentran representados en la esquina superior izquierda de la pantalla y se pueden mover desde ahí a otro lugar.

Todos los números y caracteres han sido representados en una cuadrícula de 3x5. Los iconos del tiempo se salen de esta.

Cuando se representan varios números, caracteres o imágenes quedará por encima el último que haya sido dibujado (sin borrar las partes que estuvieran debajo pero sin que las toque, como si hubiera transparencia).

En esta imagen tenemos el ejemplo de un 2, en color verde tenemos la posición original en la que fue diseñado.

En color naranja tenemos como se vería si se hubiera desplazado 7 píxeles en el eje X y -5 en el eje Y

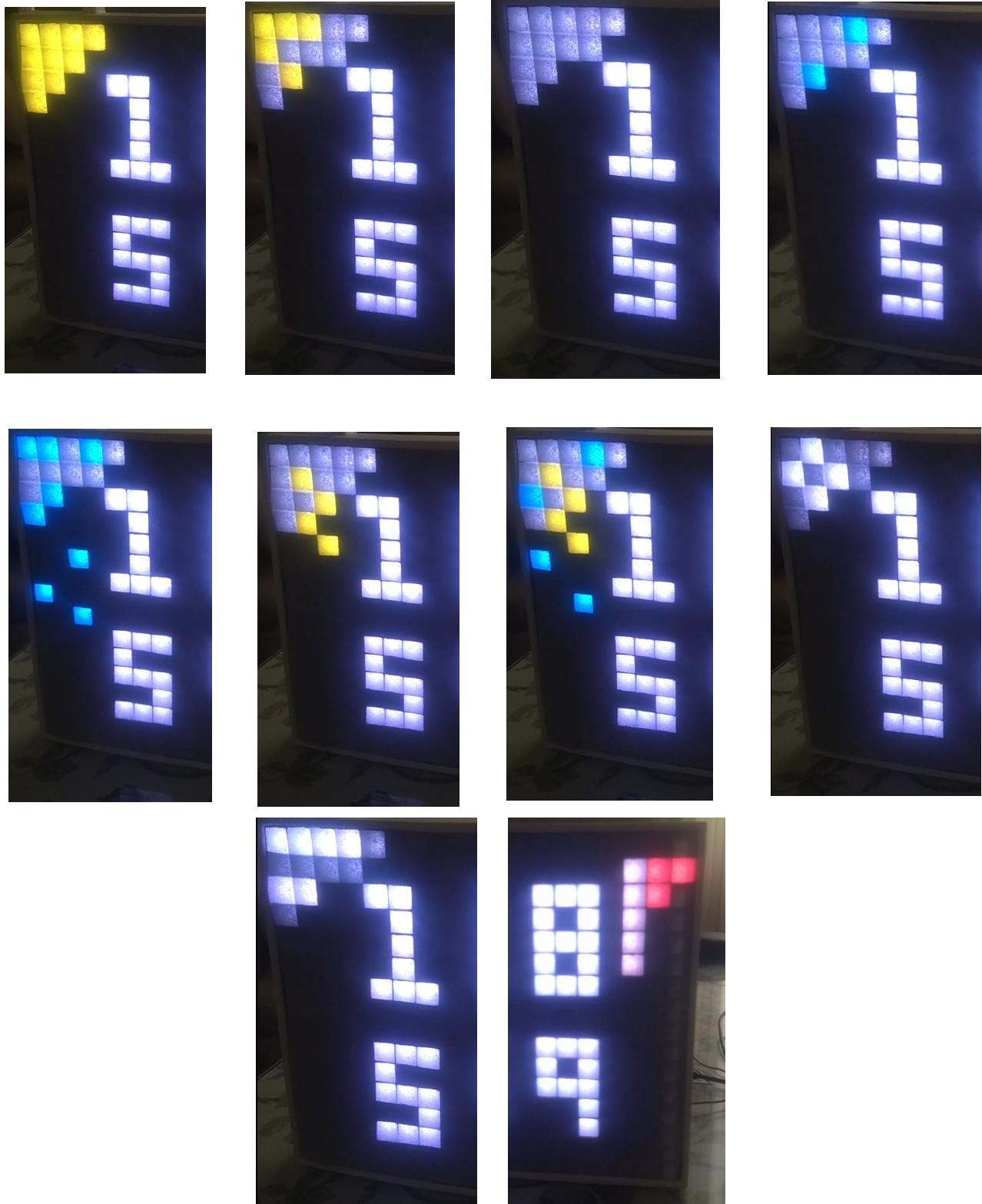
0	31	32	63	64	95	96	127	128	159	160	191	192	223	224	255
1	30	33	62	65	94	97	126	129	158	161	190	193	222	225	254
2	29	34	61	66	93	98	125	130	157	162	189	194	221	226	253
3	28	35	60	67	92	99	124	131	156	163	188	195	220	227	252
4	27	36	59	68	91	100	123	132	155	164	187	196	219	228	251
5	26	37	58	69	90	101	122	133	154	165	186	197	218	229	250
6	25	38	57	70	89	102	121	134	153	166	185	198	217	230	249
7	24	39	56	71	88	103	120	135	152	167	184	199	216	231	248
8	23	40	55	72	87	104	119	136	151	168	183	200	215	232	247
9	22	41	54	73	86	105	118	137	150	169	182	201	214	233	246
10	21	42	53	74	85	106	117	138	149	170	181	202	213	234	245
11	20	43	52	75	84	107	116	139	148	171	180	203	212	235	244
12	19	44	51	76	83	108	115	140	147	172	179	204	211	236	243
13	18	45	50	77	82	109	114	141	146	173	178	205	210	237	242
14	17	46	49	78	81	110	113	142	145	174	177	206	209	238	241
15	16	47	48	79	80	111	112	143	144	175	176	207	208	239	240

Figura 72: Ejemplo de representación de números

Para los números se han implementado todos los dígitos, del 1 al 9, si se introdujera un 10 en el parámetro *num* se mostraría un bloque negro de 3x5 (para hacer borrados parciales) y si es otro número distinto se muestra una exclamación.

En cuanto al clima aquí tenemos todos los estados climatológicos que se han diseñado para modelar la respuesta recibida de la API:

Esquina superior izquierda tiempo, esquina superior derecha viento



Sistema operativo

Para programar el comportamiento del reloj y poderle dar múltiples funciones se ha creado lo que podría llamarse un sistema operativo simple desde cero.

Se ha tratado de crear todo el código de la forma mas modular posible para que si en un futuro, en caso de querer añadir mas funciones, se pueda hacer de una manera fácilmente entendible y sin tener que tocar en otras partes de código.

Vamos a comenzar en esta parte por explicar las conexiones de red que tienen lugar, todas las funciones y código desarrollado con este fin se ha guardado en la biblioteca llamada Red.h.

Funciones de red

Primero de todo conectar nuestro módulo esp32 al router con conexión a Internet a través de wifi. Para esta tarea tenemos la librería WiFi.h.

Se habrán almacenado previamente en constantes las credenciales del router.

Se han creado dos funciones: para conectar por wifi y para comprobar la conexión:

```
//WiFi  
//Conecta con el wifi indicado en la configuración  
void conectaWiFi();  
//Devuelve 1 si la conexión wifi se encuentra activa  
int estadoWifi();
```

Figura 73: Funciones de conexión wifi

Para poder hacer peticiones HTTP a las APIs y para deserializar las respuestas Json se han necesitado las bibliotecas `HTTPClient.h` y `ArduinoJson.h`.

Las dos funciones dedicadas a obtener información de las APIs que tenemos son:

```
//Devuelve StaticJsonDocument con la fecha y hora; NULL si falla
StaticJsonDocument<5000> obtenerFecha();

//@param dia 0 para el día de hoy y 1 para mañana
DynamicJsonDocument obtenerMeteo(int dia);
```

Figura 74: Funciones de obtención de datos API REST

Se han creado tres constantes que guardan los enlaces a las API:

```
#define API_FECHAYHORA "http://worldtimeapi.org/api/timezone/Europe/Madrid"
#define API_TIEMPO "https://api.openweathermap.org/data/2.5/weather?q=Zafra"
#define API_PREVISION "https://api.openweathermap.org/data/2.5/onecall?lat=39.43&lon=-5.75&exclude=&appid=...&units=metric"
```

Figura 75: Enlaces a las API REST

Ambas funciones mostradas tienen un funcionamiento muy similar:

- Comienza la conexión HTTP, petición GET a la dirección.
- Se almacena la respuesta en un String.
- Se deserializa el Json de la respuesta.
- Se comprueban errores en la deserialización.
- Se devuelve un objeto JsonDocument con la información.

También tenemos dentro de las funciones de red la conexión con alexa, para ello utilizamos una librería llamada `FauxmoESP.h`.

Para nuestro dispositivo y la conexión con Alexa habían dos opciones:

- Simular contener dispositivos simples.
- Programar un skill para alexa con diálogos y una función lambda.

Debido a la complejidad que conlleva a desarrollar una skill, a que sería menos directa una actualización del dispositivo y a que sería necesaria la instalación de la skill para poder usar esta función la opción fue descartada.

Por ello se eligió la opción de simular que contiene multitud de dispositivos simples nombrados convenientemente y así poder enviar y recibir tanto señales analógicas como digitales.

En este caso hemos creado dos dispositivos, ambiente y brillo.

Aplicación de alexa:



Figura 76: Captura de pantalla de mi aplicación Alexa

Se le pueden dar órdenes como Alexa, pon ambiente o Alexa, pon brillo al 40%.

Esto lo consigue esta librería creando un servidor en el puerto 80 del dispositivo, el cual anuncia sus dispositivos alexa disponibles, todos son, a efectos prácticos, una bombilla para alexa.

De esta forma, nombrando inteligentemente los dispositivos podemos dar la ilusión de tener una “conversación coherente”.

Luego tiene una función llamada `onSetState()` que es una función handle, se llama cada vez que recibe una petición de Alexa y le pasa los datos que ha recibido de ella por parámetros.

Sistema principal – Escenas (tareas)

En este sistema operativo simple no vamos a tener multitarea, solamente va a haber una en ejecución y se va a ver representada en lo que vamos a llamar escenario, un número entero que va a guardar el índice de la tarea que debería encontrarse activa en ese momento.

Una escena o tarea va a ser una función encargada de mostrar y actualizar lo que se muestra en pantalla y de revisar el tiempo transcurrido desde la última actividad para irse a dormir o si se ha pulsado algún botón.

El código principal comienza con la función `setup()`, en la que se llaman a todas las funciones de inicialización de periféricos, se define el pin de interrupción externa, se define también el estado de los dispositivos alexa y la función de callback.

La función `loop()` es muy simple, y solamente se encarga de ir redirigiendo a las funciones escena con un switch. También se llama a la función handle de alexa.

Cada una de las funciones escena recibe un puntero a la variable escenario, para que puedan leerla y saber si en algún momento ha cambiado, si se hubiera pasado como un parámetro normal se crearía una copia y no se podrían percibir si se ha cambiado de alguna manera externa.

Hay una función muy importante, que se encarga de adelantar el número del escenario cuando detecta una pulsación del botón (detección por polling) y de comprobar el sensor de movimiento. Se llama de forma continua en todas las escenas. Esta es: administradorDeEscenas().

Tenemos también la función compruebaDormir() la cual comprueba si la última actividad del dispositivo fue hace más del tiempo definido hasta dormirse, si es así duerme al procesador principal.

En las escenas se puede identificar una estructura básica:

- Se borra la pantalla y se obtienen los datos necesarios a representar
- Se muestra en la pantalla
- While que continúa mientras el número de escena siga siendo el suyo.
 - Con cierta cadencia, se actualiza o no la pantalla.
 - Llama a administradorDeEscenas()
 - Llama a compruebaDormir() y al handle de Alexa.

Cabe destacar que la función handle de alexa puede en cualquier momento cambiar el número de escenario y quitarle la ejecución a esa escena para dársela a otra.

Una vez ya sabiendo esto vamos a describir como sería un funcionamiento normal a modo de aclaración.

Ejemplo de mecánica de funcionamiento de las tareas

- Es la primera ejecución, ejecuta el setup() y en el loop() entra a la escena número 0.
- Estando en el while de esa escena se pulsa el botón, esto provoca que dentro de administradorDeEscenas() el número de escenario se incremente.
- Cuando vuelve al while de la escena 0, al volver a iniciar, se da cuenta de que la escena ya no es suya y salta. Esto hace al programa volver al loop.
- Como ahora el número de escena es 1 entra en la escena 1.
- Si se pulsara ahora el botón ocurriría lo mismo que en el caso anterior. En el caso de que se cambiara el número de escenario dentro de la función handle de Alexa también funcionaría de la misma manera, al volver saldría de su ejecución y se iría a la escena indicada.

Como vemos este sistema no es muy complejo pero es efectivo, responde rápido y al momento en cuanto cambia la variable escenario.

Las tareas son ágiles y se mueven sin ningún problema independientemente entre ellas.

Todas las funciones de escenario han sido guardadas en la biblioteca Escenas.h. Si se quisiera añadir alguna, solamente se debería crear una nueva función escenario y añadirla al switch del loop con los escenarios posibles.

Montaje

Comencemos por la pantalla, como hemos visto anteriormente la pantalla sin nada eran solo píxeles redondos, y se está buscando en este proyecto que estos sean cuadrados y con un brillo uniforme.

Para lograrlo se ha diseñado e impreso en 3D una rejilla con el tamaño idóneo para que quepan los píxeles:

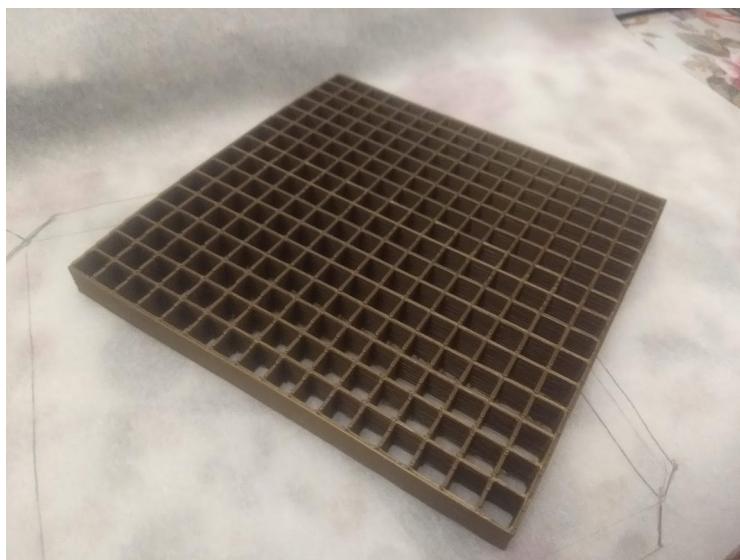


Figura 77: Rejilla de la pantalla

La matriz de LEDs se encontrará por debajo y por encima una pantalla que difuminará y homogeneizará la luz. Aquí se ve como es el papel difuminador:

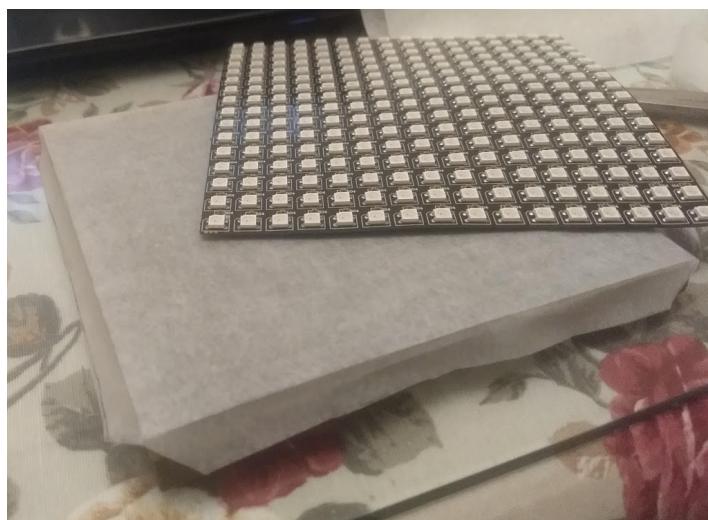


Figura 78: Rejilla, papel y matriz led

Se ha controlado que hubiera suficiente distancia entre el papel y los led, porque si estuvieran muy cerca no iluminarían totalmente el cuadrado y se seguiría viendo redondo debido a su ángulo de emisión.

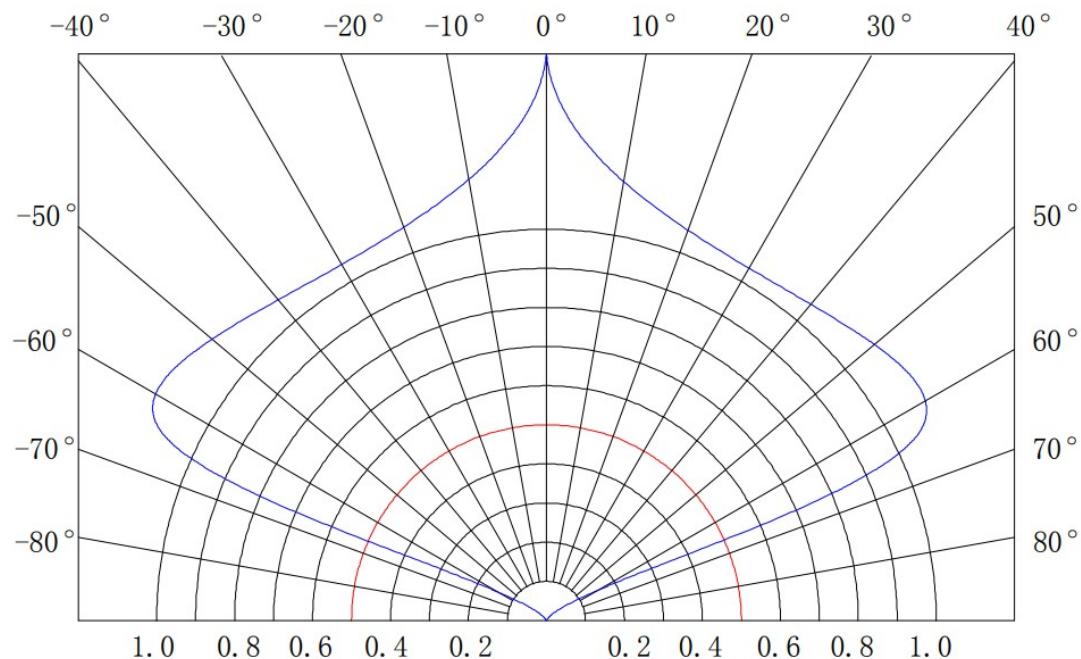


Figura 79: Ángulo de emisión de un led Fuente: <https://www.tweaking4all.com/wp-content/uploads/2014/01/5050LED.pdf>

Por eso se le ha dado una distancia de 1,4 cm.

Se ha diseñado también un marco para sostener la pantalla y el sensor de distancia:

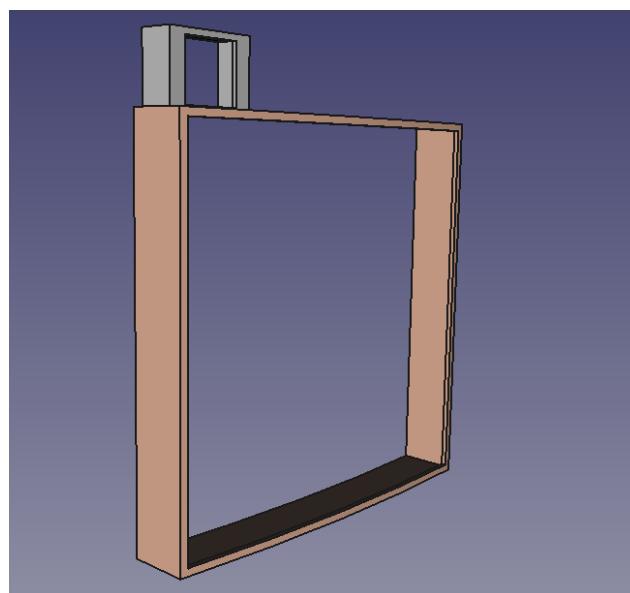


Figura 80: Modelo 3D del marco

Este es el resultado de la pantalla una vez montada:



Figura 81: Pantalla montada

Una vez esto hecho se imprimió la parte trasera de la caja:

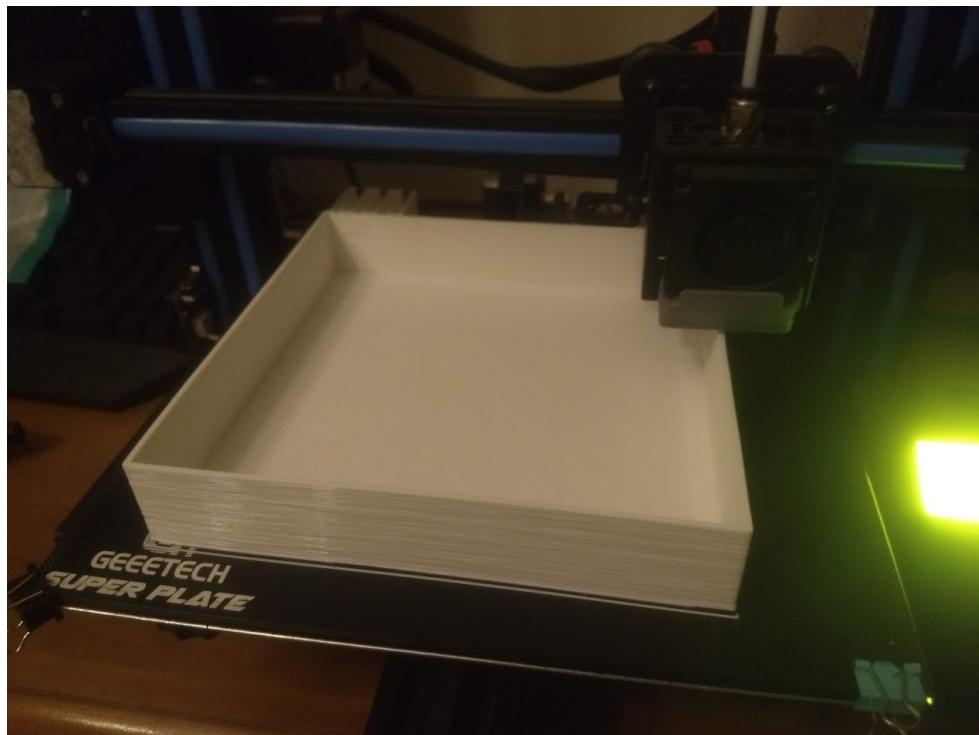


Figura 82: Caja trasera siendo imprimida

Esta es una imagen del terreno de pruebas previo a ser montado en la caja:

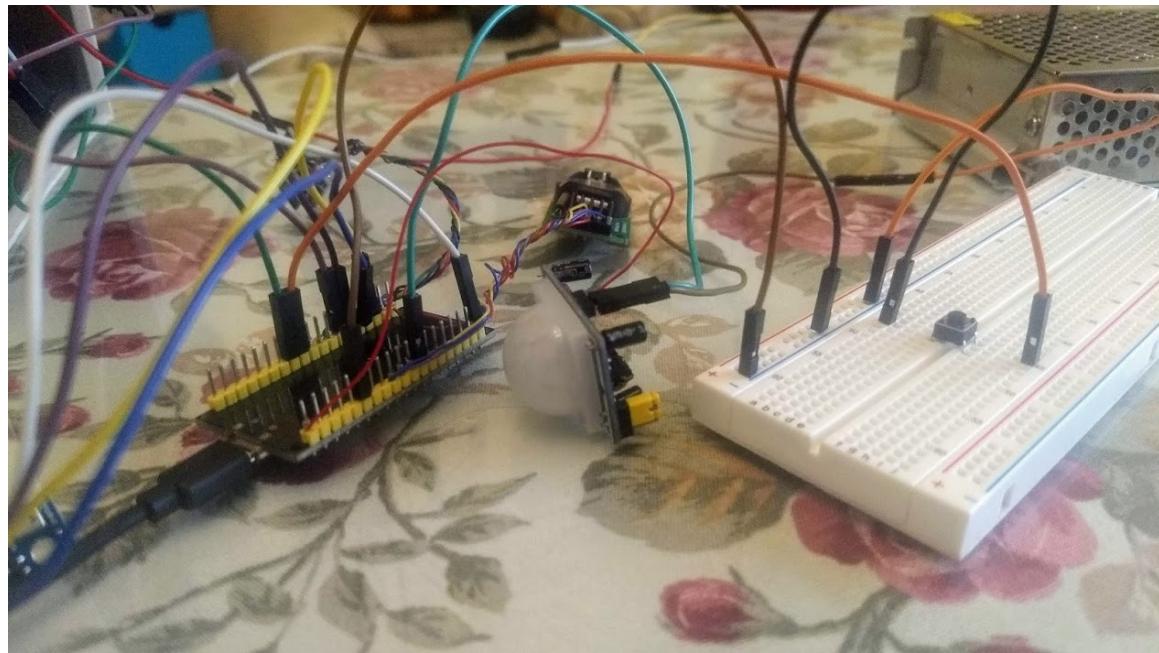


Figura 83: Pruebas antes de el montaje en la caja

A continuación se introducen los componentes:



Figura 84: Elementos montados en la caja

Todos los componentes fueron cableados con la técnica del wire wrapping:

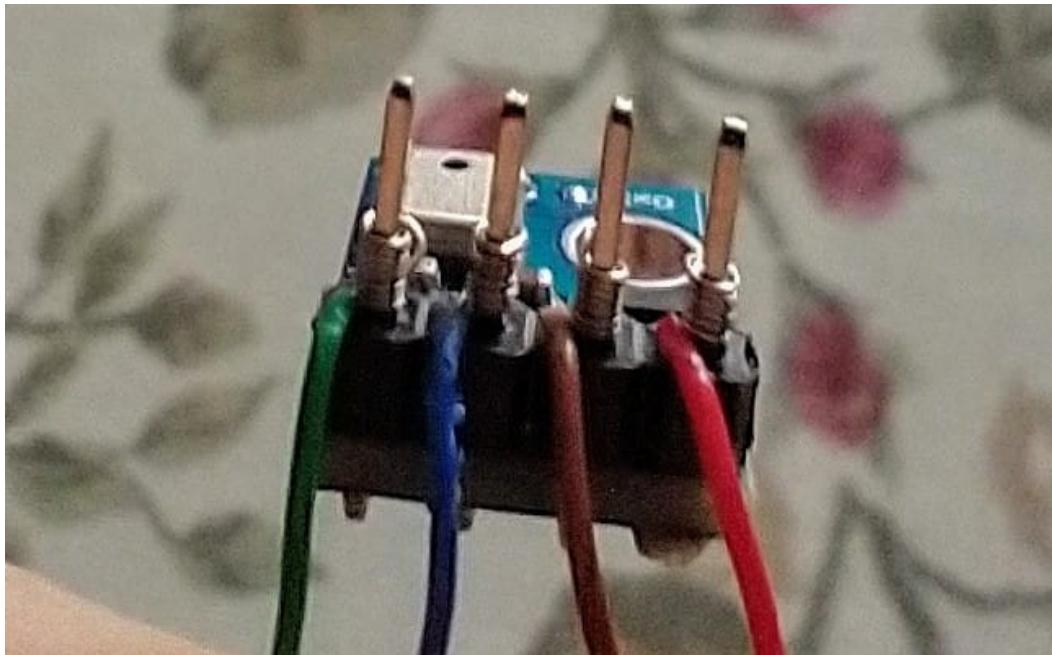


Figura 85: Imagen detalle del wire wrapping

Y además se trenzaron los cables para disminuir ruido en las señales y para una mejor organización dentro de la caja:

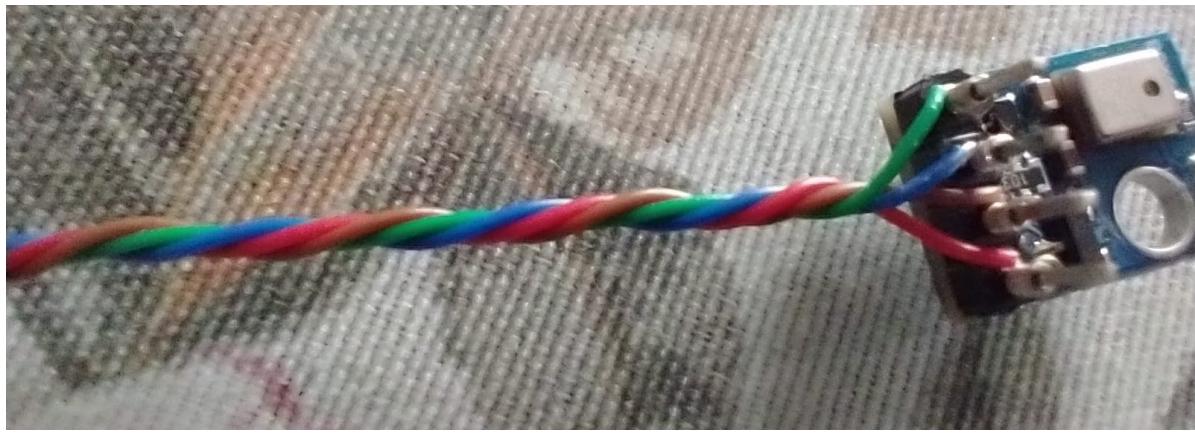


Figura 86: Imagen detalle del trenzado de cables

Pruebas del dispositivo ya en la caja:

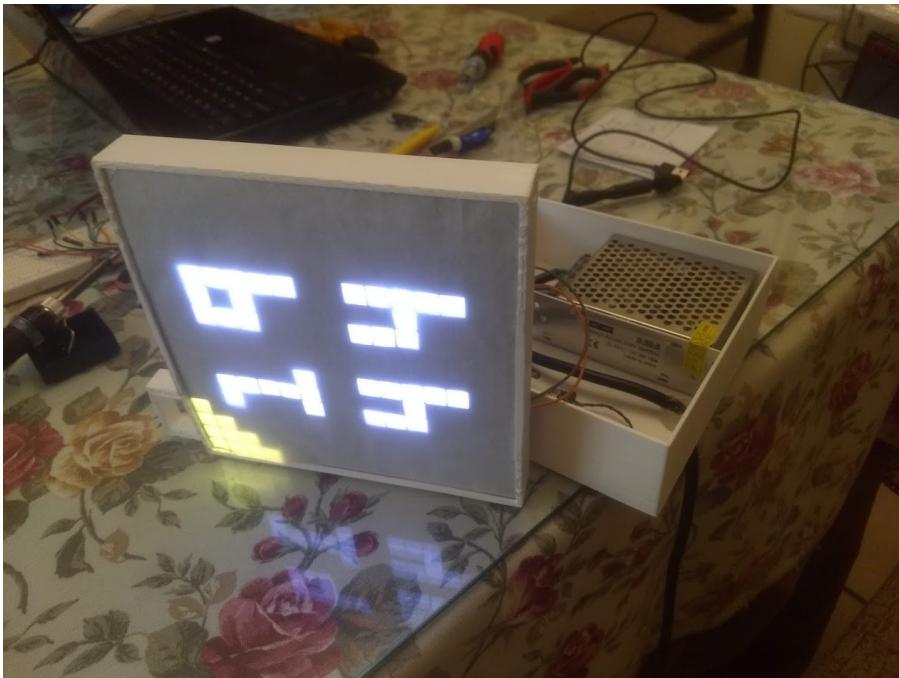


Figura 87: Probando la pantalla principal



Figura 88: Probando la pantalla ambiente

Resultado final:



Figura 90: Pantalla principal



Figura 89: Pantalla previsión de mañana (arriba temp. máxima, abajo temp. mínimas)



Figura 91: Pantalla temperatura y humedad de la sala

Estas son las 3 escenas disponibles, reloj/tiempo, tiempo de mañana y temperatura/humedad en la sala. (De arriba a la izquierda hacia abajo)

Coste del proyecto

Ahora a modo de curiosidad y para poder comparar con las otras alternativas del mercado se expone el coste total del proyecto.

Este es el precio de los componentes:

	Precio
Sensor humedad/temperatura	1,55 €
RTC x5	2,85 €
Fuente de alimentación	7,26 €
Bahía micro SD	1,01 €
ESP32	4,71 €
Pantalla LED	11,35 €
Sensor de presencia	1,55 €
Impresión de carcasa	8,00 €
Total	38,28 €

Figura 92: Tabla de precios de componentes

A continuación, vamos a contabilizar las horas de trabajo y el resto de factores a tener en cuenta para comprobar que precio de mercado tendría en el hipotético caso de que se quieran fabricar 1000 unidades.

Mano de obra

Para este proyecto se estima que se han trabajado 63 días, en los que, de media, se han trabajado unas 4,5 horas diarias.

Supongamos que se cotizan a 10€/h.

Se han de contar también las horas invertidas en la impresión de la carcasa, ya que, o era tiempo invertido en calibración y pruebas de la máquina, o vigilando las piezas. Por ejemplo, la pieza mas grande, la parte trasera, tardó 10 horas en ser imprimida. En total, aproximadamente 23h.

Estas por tener menos actividad se cotizarán a 6€/h.

Esto nos da 283,5h de mano de obra a 10€ y 23h a 6€.

En total, la mano de obra sale a: 2973€

Otros factores

Ahora tenemos que suponer que, si vamos a comprar materiales en grandes cantidades, nos saldrá mas barato al por mayor, esto puede suponer una reducción en el precio de los materiales del 25%.

Precio al por mayor del dispositivo: 28,71€

Tenemos además gastos en marketing y distribución, que vamos a valorar en 6.500€.

Los trámites y las licencias necesarias nos cuestan 2.500€

Esperamos obtener un beneficio por unidad del 30%.

Este cálculo se trata de un cálculo hipotético para que nos podamos imaginar cuál sería el precio real del dispositivo y no trata de ser preciso con la realidad.

El precio total para el fabricante para poder fabricar y comercializar estos dispositivos sería el siguiente:

	Precio
Dispositivos x1000	28.710,00 €
Mano de obra	2.973,00 €
Marketing y distribución	6.500,00 €
Licencias y trámites	2.500,00 €
Total	40.683,00 €

Figura 93: Total de costes

Si ahora repartimos de nuevo este gasto entre los 1000 dispositivos obtenemos que a el fabricante le cuesta 40,68€ por unidad.

Recordando que esperamos un beneficio del 30% tenemos que el precio final del dispositivo en el mercado sería de:

52,88€

Como vemos, el precio resultante no es descabellado y podría competir perfectamente con las otras alternativas del mercado.

Conclusiones

Como conclusión final se puede destacar que este proyecto ha servido en gran medida para saber en profundidad acerca de todo el proceso desde programar un microcontrolador y saber como funcionan sus componentes hasta diseñar un producto en todos sus otros ámbitos.

También puedo decir que me he enfrentado a lo que es llevar un trabajo mas grande por mi cuenta con todo lo que ello implica, pandemia global inclusive, y a saber llevarlo a buen término incluso si se están viendo las cosas por primera vez y no se sabe muy bien como funcionan.

Por último me gustaría agradecer al lector de esta memoria su interés en ella y espero que le haya resultado una lectura amena.

Muchas gracias.

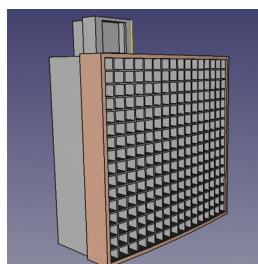


Figura 94: Modelo 3D del dispositivo

Antonio Vázquez Pérez.

Índice de figuras

Figura 1: Pantalla principal (Clima → parte superior izq.(soleado); viento→ banderín a la derecha; hora → en el centro).....	6
Figura 2: Divoom tivoo max trasera Fuente: www.divoom.com	9
Figura 3: Divoom tivoo max frontal Fuente: www.divoom.com	9
Figura 4: Habilidades divoom tivoo max Fuente: www.divoom.com	10
Figura 5: Tienda online divoom Fuente: www.divoom.com consultado el 4/9/2020.....	11
Figura 6: Echo show Fuente: Tienda amazon, www.amazon.es	12
Figura 7: Echo show conferencia Fuente: Tienda amazon, www.amazon.es	12
Figura 8: Tienda digital Amazon Fuente: www.amazon.es consultado el 4/9/2020.....	13
Figura 9: Diseño 3D del dispositivo, conexiones.....	15
Figura 10: Diagrama de Gantt.....	16
Figura 11: ESP32-WROOM-32.....	20
Figura 12: Sensor AHT10.....	20
Figura 13: Sensor infrarrojos PIR.....	21
Figura 14: Reloj RTC.....	21
Figura 15: Matriz led RGB.....	21
Figura 16: Fuente de alimentación 50W.....	22
Figura 17: Software visual studio code.....	23
Figura 18: Complemento platformIO.....	23
Figura 19: Repositorio en www.github.com	24
Figura 20: Software FreeCAD.....	25
Figura 21: Software ultimaker Cura.....	25
Figura 22: Software GIMP.....	26
Figura 23: Software LibreOffice writer.....	26
Figura 24: Periféricos.....	27
Figura 25: Módulo SD frontal.....	28
Figura 26: Módulo SD trasera.....	28
Figura 27: Conexionado SD Fuente: https://alexlubbock.com/micro-sd-adapter-esp8266-esp32	29
Figura 28: Conexionado SPI Fuente: https://es.wikipedia.org/wiki/Serial_Peripheral_Interface	29
Figura 29: Periférico RTC real.....	30
Figura 30: Diagrama de bloques RTC.....	31
Figura 31: Byte dirección/comando.....	32
Figura 32: Resumen transferencia de datos.....	33
Figura 33: Circuito integrado DS1302.....	33
Figura 34: Funciones del RTC.....	34
Figura 35: Gráfico error en humedad.....	35
Figura 36: Gráfico error en temperatura.....	35
Figura 37: Sensor AHT10 real trasera.....	36
Figura 38: Sensor AHT10 real frontal.....	36
Figura 39: Comunicación I2C.....	36
Figura 40: Comunicación I2C.....	37
Figura 41: Condición de comienzo.....	37
Figura 42: Comprobación de dirección.....	38
Figura 43: Confirmación.....	38
Figura 44: Condición de parada.....	39
Figura 45: Tabla de comandos disponibles.....	40

Figura 46: Envío de byte de medición.....	40
Figura 47: Tabla de respuestas del sensor.....	41
Figura 48: Respuesta con los datos de medición.....	41
Figura 49: Funciones para el sensor AHT10.....	42
Figura 50: Sensor PIR.....	43
Figura 51: Campo de visión.....	43
Figura 52: Detección infrarroja.....	44
Figura 53: Controles del sensor PIR.....	44
Figura 54: Potenciómetros del sensor PIR.....	45
Figura 55: Matriz led rojo 8x8.....	46
Figura 56: Panel en venta en amazon.com.....	47
Figura 57: Leds NeoPixel Fuente: https://www.adafruit.com/product/1655	47
Figura 58: Tamaño led NeoPixel Fuente: https://www.adafruit.com/product/1655	48
Figura 59: Combinación de colores aditiva Fuente: https://mundotextilmag.com.ar/color-y-colorantes/	48
Figura 60: Tira de leds encadenados Fuente: https://www.adafruit.com/product/1506	49
Figura 61: Imagen del vendedor del que fue adquirido (Enlace ya no disponible).....	50
Figura 62: Ordenación de leds en la matriz.....	50
Figura 63: Led NeoPixel, terminales.....	51
Figura 64: Leds en cascada.....	51
Figura 65: Estructura de envío de bits al led.....	51
Figura 66: Codificación por pulsos.....	51
Figura 67: Tiempos para la transferencia.....	52
Figura 68: Desplazamiento a la derecha, columna par.....	54
Figura 69: Desplazamiento a la izquierda, columna par.....	54
Figura 70: Desplazamiento a la derecha, columna impar.....	55
Figura 71: Desplazamiento a la izquierda, columna impar.....	55
Figura 72: Ejemplo de representación de números.....	57
Figura 73: Funciones de conexión wifi.....	59
Figura 74: Funciones de obtención de datos API REST.....	60
Figura 75: Enlaces a las API REST.....	60
Figura 76: Captura de pantalla de mi aplicación Alexa.....	61
Figura 77: Rejilla de la pantalla.....	65
Figura 78: Rejilla, papel y matriz led.....	65
Figura 79: Ángulo de emisión de un led Fuente: https://www.tweaking4all.com/wp-content/uploads/2014/01/5050LED.pdf	66
Figura 80: Modelo 3D del marco.....	66
Figura 81: Pantalla montada.....	67
Figura 82: Caja trasera siendo imprimida.....	67
Figura 83: Pruebas antes de el montaje en la caja.....	68
Figura 84: Elementos montados en la caja.....	68
Figura 85: Imagen detalle del wire wrapping.....	69
Figura 86: Imagen detalle del trenzado de cables.....	69
Figura 87: Probando la pantalla principal.....	70
Figura 88: Probando la pantalla ambiente.....	70
Figura 89: Pantalla previsión de mañana (arriba temp. máxima, abajo temp. mínimas).....	71
Figura 90: Pantalla principal.....	71
Figura 91: Pantalla temperatura y humedad de la sala.....	71
Figura 92: Tabla de precios de componentes.....	72
Figura 93: Total de costes.....	74
Figura 94: Modelo 3D del dispositivo.....	75

