****

**U.B.A. FACULTAD DE INGENIERÍA**

**Departamento de Electrónica**

**LABORATORIO 66-02**

**Informática**

**TRABAJO PRÁCTICO INTEGRADOR**

**Curso 2020 - 2do Cuatrimestre**

**Turno : Lunes**

|  |  |
| --- | --- |
| **GRUPO N** | |
| **APELLIDO, Nombres** | **N PADRÓN** |
| Costa, Facundo | 99765 |
| Moralejo Salinero, Agustin | 101591 |
|  |  |
| **Alumno Responsable :** Moralejo Salinero, Agustin | |
| **Fecha de Realización :** 21/02/2021 |  |
| **Fecha de Aprobación :** |  |
| **Calificación :** |  |
| **Firma de Aprobación :** |  |



**ÍNDICE**

[**OBJETIVO Y MOTIVACIÓN**](#_pvk9jemm8oxg) **3**

[**INTRODUCCIÓN**](#_pi1q228v026j) **4**

[**DESARROLLO**](#_y6deza7l4ls) **7**

[**CIRCUITO**](#_4guozkeytsdi) **12**

[**COMPLICACIONES**](#_a2n14ojlvoo9) **13**

[Lectura de datos en el sensor](#_mn6hwo46mbsm) 13

[Uso del delay](#_4mxhuiy2rprg) 13

[Impresión por display oled](#_xzq997pby7qn) 14

[Rebote en los pulsadores](#_egrgc0j8ib9v) 15

[**FUNCIONAMIENTO DEL PROGRAMA**](#_j0mlh5za07vl) **17**

[Pantalla principal](#_ridbdyjl5il8) 17

[Pantalla de sensores](#_ohmgxkez5bbl) 18

[Pantalla de relés](#_8om358o2to24) 18

[Pantalla de configuración](#_n11qg8884nup) 18

[Video demostrativo de su funcionamiento](#_ig1ovly5qeyd) 21

[**MEDICIONES EN EL CIRCUITO**](#_8zga56h0qoit) **22**

[Medición para corroborar las tensiones y masas del arduino](#_ly5kpjrsk8hp) 22

[Medición de tensión en los relés](#_p90wpaoitto8) 24

[Medición de tensión en los pulsadores](#_x8l2nwgwsjgw) 27

[Medición con Osciloscopio](#_1ifrm6qhr2k9) 29

[**EJEMPLOS DE APLICACIONES, BENEFICIOS Y REFERENCIAS DE OTROS TRABAJOS**](#_nlht66ilmoc6) **31**

[**CONSIDERACIONES A FUTURO**](#_ysuj3vhn0t1u) **32**

[**AGRADECIMIENTOS**](#_i7tvcq5sv3cx) **33**

[**REFERENCIAS**](#_ies44zme1w9t) **33**

[**APÉNDICE**](#_3qrh53hbvha5) **34**

[COMPONENTES Y HERRAMIENTAS](#_ex7pnw3uprpf) 34

[HOJA DE DATOS UTILIZADAS](#_uk4j66y8zrnr) 36

[CÓDIGO EN ARDUINO](#_wlajs6a80czt) 36

# OBJETIVO Y MOTIVACIÓN

El objetivo del trabajo práctico integrador de Laboratorio (66.02) de la Facultad de Ingeniería reside en realizar un proyecto con todos los conocimientos adquiridos en la materia, que comprende investigación, análisis, desarrollo y puesta en marcha del proyecto.

Nuestra motivación principal es, no solo utilizar todos los conocimientos de esta materia, sino también poder canalizar los adquiridos en otras, para complementar este proyecto final.

El equipo está formado por alumnos de la carrera de Ingeniería en Informática, y por tal motivo, decidimos realizar un proyecto que pueda unir los conocimientos de electrónica e informática.

Elegimos para el proyecto de la asignatura, diseñar un dispositivo para lograr un ambiente controlado en temperatura y humedad, basado en el microcontrolador Arduino.

Para la realización del mismo utilizamos:

* Un módulo de cuatro relés para manejar diferentes tipos de aparatos eléctricos.
* Dos sensores de temperatura/humedad.
* Un display oled para la muestra de las diferentes mediciones.
* Tres pulsadores para el manejo del display.
* Un microcontrolador Arduino.

# 

# INTRODUCCIÓN

Un ambiente controlado por temperatura y humedad consiste en poder establecer y controlar diferentes umbrales de temperatura y humedad, para que de esta forma el mismo se mantenga dentro los parámetros establecidos.

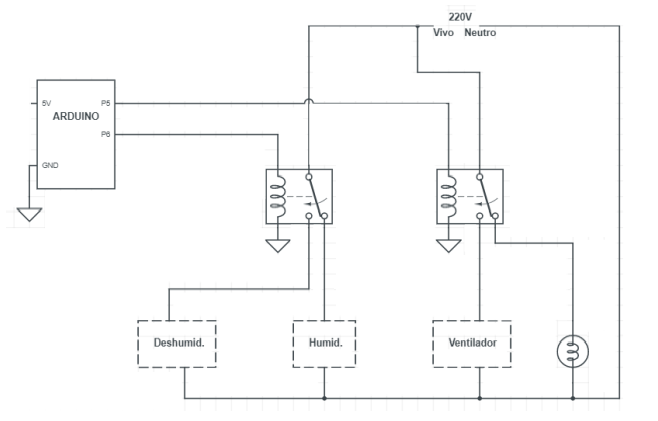
A continuación se describe brevemente el funcionamiento del sistema:

Contamos con dos sensores que miden los valores de temperatura (en °C) y humedad en (%RH, el porcentaje de humedad relativa que hay en el aire).

El microcontrolador Arduino mediante el programa desarrollado controlará estas mediciones y los comparará con los umbrales mínimos y máximos seleccionados. Estos valores serán configurados por el usuario y se mostrarán mediante un display. En la salida de los relés contaremos con diferentes dispositivos eléctricos que se utilizarán para aumentar/disminuir tanto la temperatura como la humedad ambiente.

Un ejemplo podría ser la conexión de una luz incandescente para aumentar la temperatura y un ventilador para bajar la misma, y para controlar la humedad se podría utilizar un humidificador para aumentarla y deshumidificador para disminuirla.

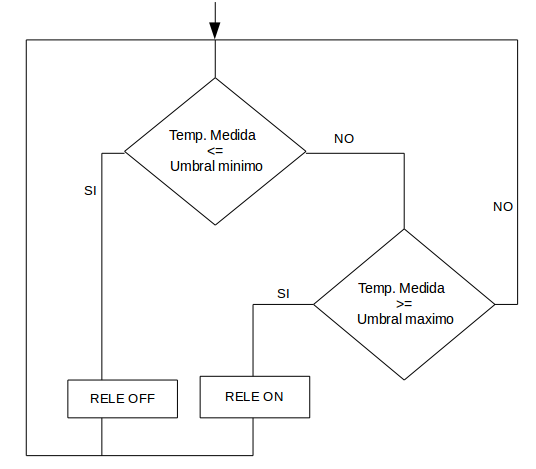
El esquema de conexión es el siguiente:



*Circuito con sus respectivas conexiones*

El usuario seteará un umbral mínimo y un umbral máximo, (considerando que el Umin < Umax), y conectará dos dispositivos eléctricos (luz incandescente, ventilador) a las salidas del relé, una de ellas al estado normal cerrado y el otro al estado normal abierto, y ambas conectadas a la tensión de línea.

En el siguiente diagrama se muestra la lógica de automatización:



*Diagrama de flujo de la automatización de los sensores para un sensor de temperatura*

El programa solicitará la temperatura al sensor y preguntará si es menor o igual al umbral mínimo, si lo es, apagará el relé, si no pregunta si la temperatura medida es mayor o igual al umbral máximo, en caso de ser mayor, activará el relé.

De esta forma la lámpara conectada al NC del relé estará siempre prendida, cuando alcanzamos la temperatura del umbral máximo, activará el relé apagando la lámpara y encendiendo el ventilador, haciendo que la temperatura comience a bajar.

Cuando llegue al umbral mínimo el relé se apagará, encendiendo nuevamente la luz, manteniendo este ciclo en forma continua, de esta forma obtendremos un ambiente controlado por temperatura, para el caso de la humedad se realizará el mismo procedimiento conectando otros dispositivos.

En el proyecto utilizamos dos sensores de temperatura/humedad y cuatro relés, además mostramos por pantalla los valores de temperatura y humedad, los estados de los relés, y los valores de los umbrales (mínimo y máximo) para cada sensor.

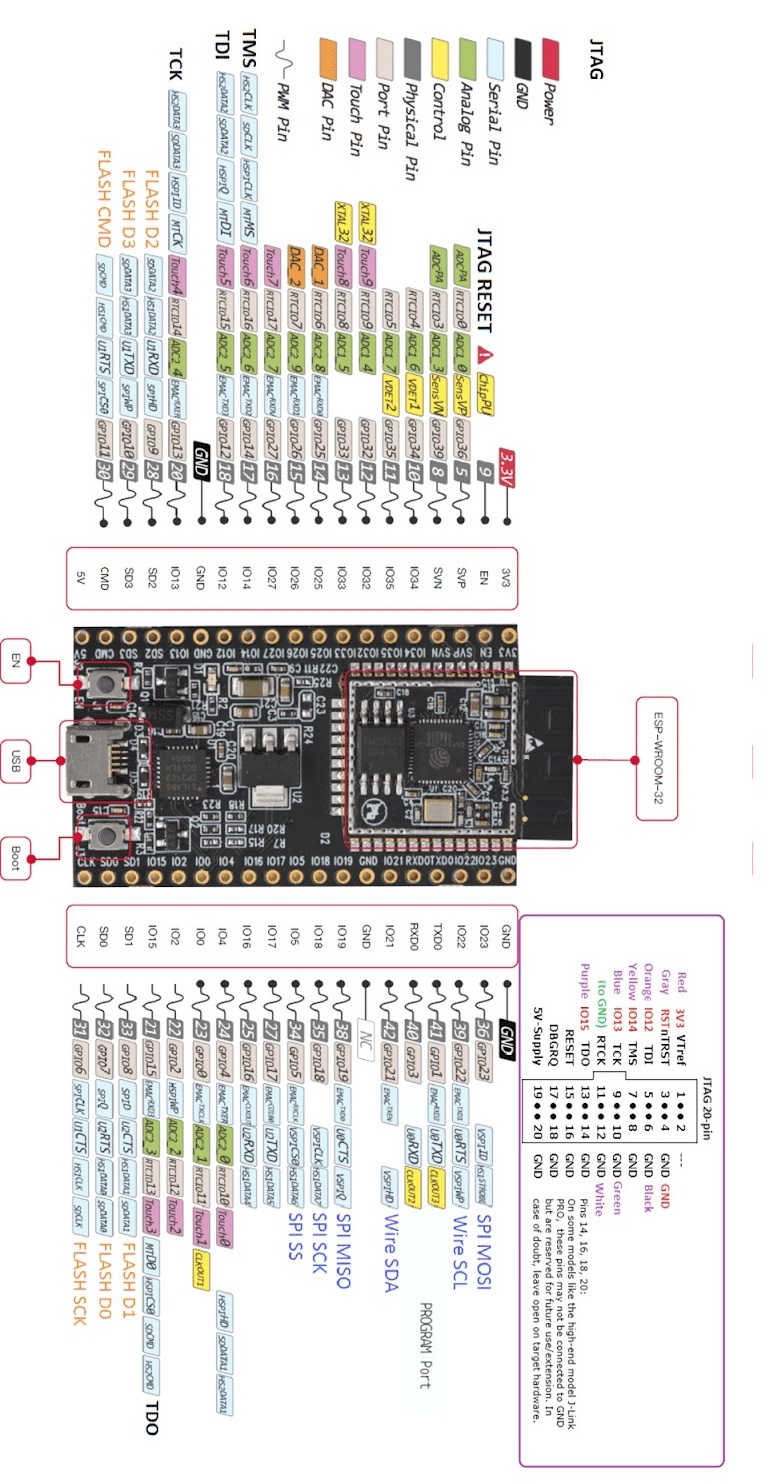
Para movernos por la pantalla utilizamos 3 pulsadores, uno para avanzar a la izquierda, otro para la derecha y uno para aceptar o salir de la configuración de los umbrales. La modificación de los mismos se realizan en esta última pantalla, y para ello utilizamos nuevamente los tres botones.

# DESARROLLO

Sabiendo previamente el funcionamiento básico de la automatización, comenzamos a realizar el circuito en protoboard, utilizamos dos protoboard conectadas uno al lado del otro, para colocar en el centro el Arduino.

Si bien el arduino utilizado es un microcontrolador de 38 patas, no se utilizaron la totalidad de las mismas, la elección del arduino fue pensada en base a poder realizar nuevos proyectos a futuro, ya que cuenta con Wi-Fi integrado, Bluetooth, tiene bajo consumo, entre muchas otras características. No obstante, si quisiéramos limitarnos exclusivamente al proyecto en cuestión deberíamos utilizar un micro con un mínimo de 11 pines de I/O (2 para los sensores, 2 pines I2C para el display, 3 para los pulsadores y 4 para los relés).

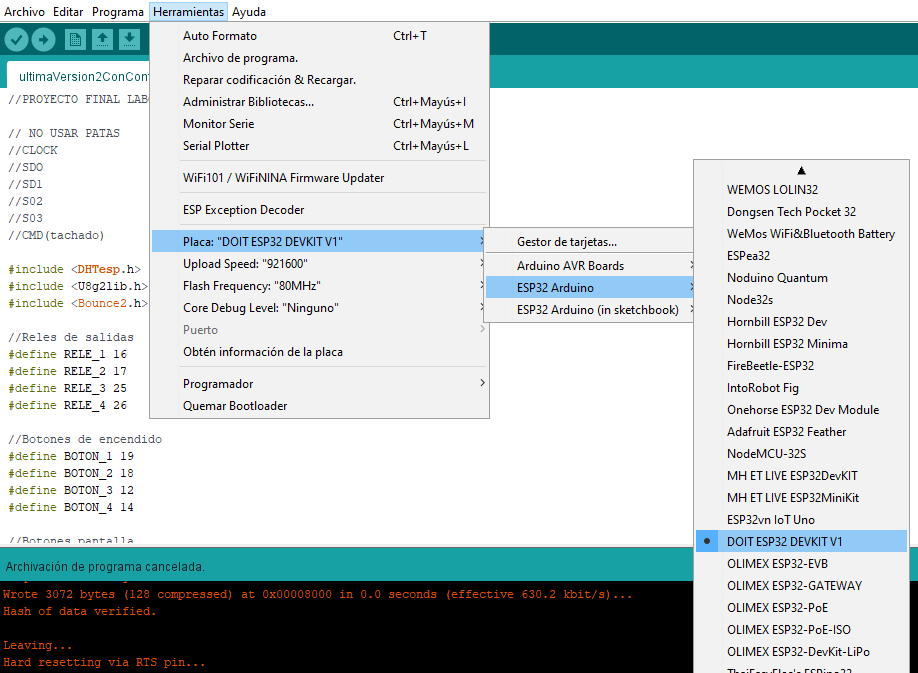
Como primer paso, descargamos la IDE de Arduino, buscamos en la página oficial el pin-out (dentro de la hoja de datos), para saber como alimentarlo, cuales pines son de I/O y que podemos utilizar.



Posteriormente, realizamos una conexión básica, el arduino cuenta con un puerto USB, que nos servirá para alimentarlo y programarlo.

La primera conexión realizada fue la de los relés, luego procedimos a buscar el modelo de arduino en la IDE para programar ya que hay que elegir el adecuado para cada uno.

Debido a que el modelo ESP32 WROOM no se encontraba en la lista, tuvimos que ponernos a buscar en internet, luego de pasar por los foros de arduino, un link nos redireccionó a <https://github.com/espressif/arduino-esp32> una libreria que nos permite instalar en la IDE la placa para poder conectar los diferentes arduinos esp32.



Arduino se programa en C/C++ por lo que la parte de programación no tuvo mucha complicación con los conocimientos adquiridos, no obstante primero hay que entender como funciona arduino.

El programa posee dos funciones principales: un setup() y un loop().

El setup se utiliza para inicializar el microcontrolador, con las rutinas tales como “pinMode” podemos elegir si un pin es de entrada o salida, o “digitalWrite” para escribir un 0 o 1 como estado inicial. Por otro lado, el loop vendría a ser nuestro main(), solamente que el arduino está corriendo constantemente por lo que nuestra función principal es un ciclo infinito, aquí realizaremos nuestro programa principal.

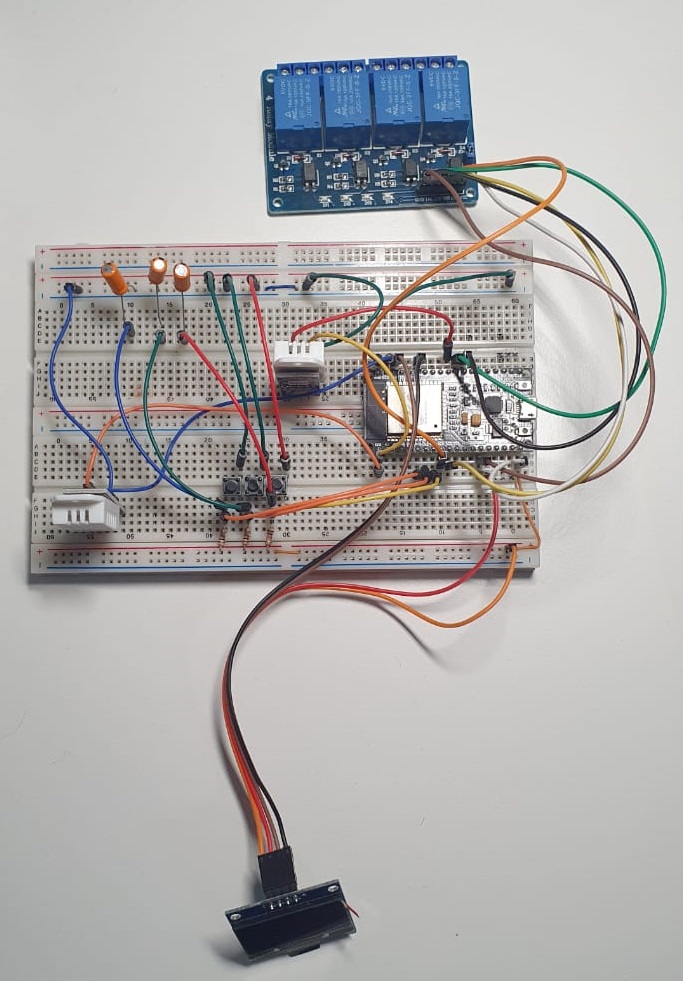
Luego de hacer funcionar los relés, conectamos los sensores de temperatura y humedad, alimentados con 3,3V y GND, y un pin para obtener los datos. Para poder inicializar y utilizar los sensores DHT22 buscamos una librería que nos permitiera leer los datos, la libreria <https://github.com/beegee-tokyo/DHTesp> funciono perfectamente, y pudimos obtener datos y mostrarlos por el monitor serie (por pantalla de la PC).

Finalmente, y lo más importante fue la conexión del display oled, el mismo cuenta con 2 pines para la trasmisión, a través de un bus I2C, el SCK y el SDA, un pin para la transmisión de datos SDA y el SCK que maneja el tiempo del reloj, de ahí CK (Clock). El ESP32 cuenta con 2 pines para esta conexión, por lo que lo conectamos en estos pines.

Para la inicialización del display, buscamos nuevamente en internet sobre el modelo del display SH1106 y encontramos diferentes librerías, luego de algunas pruebas y errores probando diferentes librerías la <https://github.com/olikraus/u8g2/> funcionó perfectamente, no solo eso, sino que soporta una gran lista de displays y tiene varias instrucciones para poder realizar diferentes dibujos y funcionalidades.

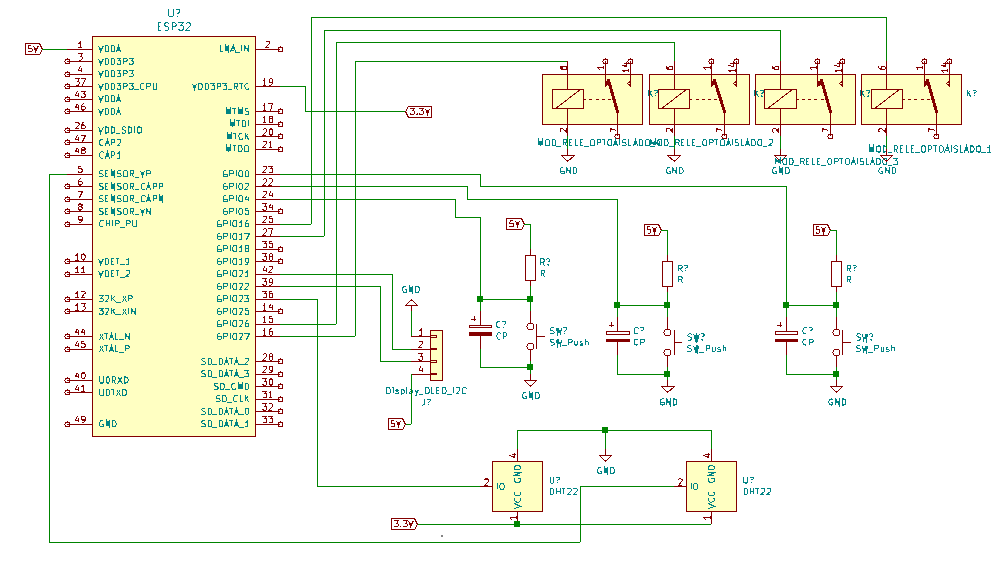
Para finalizar, conectamos 3 pulsadores en forma de PULL-UP, con resistencias de 10K, para poder utilizar los pulsos como teclado para movernos por el display.

El modelo en el protoboard quedó de la siguiente forma:



# CIRCUITO

Con todo lo explicado anteriormente, el circuito ilustrativo quedó de la siguiente forma:



# COMPLICACIONES

En esta sección hablaremos de los diferentes problemas que fueron surgiendo mientras realizamos el proyecto, y las soluciones que fuimos encontrando.

## Lectura de datos en el sensor

El sensor DHT22 nos permite obtener los valores de temperatura y humedad relativa, la inicialización en el programa es bastante sencilla y no hubo dificultad para hacerlo funcionar. No obstante, un problema que surgió fue cuando quisimos obtener los datos del sensor.

Para obtener los mismos, la instrucción “getTempAndHumidity()” nos permite obtener los valores de temperatura y humedad correspondientes, ahora como el programa se encuentra en un loop infinito, si yo quiero mostrar la temperatura y humedad y estoy constantemente pidiéndole al sensor los valores, el mismo no nos devuelve un valor legible.

Investigando un poco más en la librería del DHT22, notamos que para obtener los valores hay que esperar un cierto periodo de tiempo. Para ello, la función “getMinimumSamplingPeriod()” nos devuelve el mínimo periodo que hay que esperar para obtener un nuevo valor, por lo que al hacer un delay de este periodo de tiempo solucionamos el problema.

## Uso del delay

La solución de un problema puede traernos otro inconveniente, y esto fue lo que nos pasó con el uso del delay() para obtener los datos de los sensores.

Como mencionamos, el programa se encuentra corriente en un ciclo infinito, y un delay hace que el programa se detenga por un tiempo determinado, y luego siga funcionando. Si bien el delay fue la solución al problema anterior, ahora nuestro programa se queda congelado por dos segundos, y al tener un sistema con botones y pantallas, si nosotros queremos hacer que un botón cambie la pantalla, el programa nunca va a detectar el pulso del botón, puesto que el delay hace que el programa no avance.

Buscando en los foros de arduino, encontramos una solución que hace uso de la instrucción millis(), la misma nos devuelve el tiempo que transcurrió el circuito desde que se inició. Por consecuencia, si establecemos una variable global de tiempo, podemos determinar con esta instrucción cuanto tiempo paso.

//Variable globlal para manejar el tiempo de ejecución y poder realizar un delay

unsigned long tiempoDHT1=0;

Entonces, para realizar un delay lo que hacemos es llamar dentro de la función a millis y almacenarlo en una variable llamada tiempoNuevo, hacemos la resta de este tiempo con la variable global y preguntamos si pasó el tiempo de muestra para obtener los datos, si es asi se resetea la variable global con el nuevoTiempo, y el programa seguirá funcionando, por lo que solo obtendrá los valores de temperatura y humedad si pasaron los segundos establecidos por el sampling period, solucionando el problema del uso del delay.

En el siguiente ejemplo, un extracto del código donde se utiliza millis():

unsigned long nuevoTiempo = millis();

if ((nuevoTiempo - tiempoDHT1) > (dht.getMinimumSamplingPeriod())){

tiempoDHT1 = nuevoTiempo;

TempAndHumiditytempYhum = dht.getTempAndHumidity();

}

## Impresión por display oled

Uno de los desafíos que tuvimos que afrontar fue el manejo del display y la disposición de todos los valores que quisimos mostrar, desde la temperatura y humedad, hasta los estados de los relés, entre otras.

Si bien no hubo muchos problemas a la hora de realizar esto, tuvimos que organizar bien la pantalla, elegir diferentes fuentes para escribir en el display, ubicarnos en la posición correcta, y manejar diferentes valores de posiciones para poder escribir de forma correcta para que se pueda entender.

Uno de los problemas que tuvimos, fue que la pantalla a veces mostraba lo que queríamos pero comenzaba a presentar ruido y la impresión no limpia, esto nos sucedió al querer mostrar varios valores al mismo tiempo, y fue solucionado a través del recurso de millis() previamente comentados.

Otro de los desafíos fue el envío de datos mediante la sentencia sendBuffer(). El display funciona de forma que uno escribe todo lo que necesita en “memoria”, y al final se envía por buffer todos los valores al display.

Hay que tener precaución con el uso de esta instrucción sendBuffer(), y debemos hacerlo al final de todo, de forma tal de no repetir el envío, para no pisar la pantalla con los valores enviados previamente.

## Rebote en los pulsadores

Finalmente, nos encontramos con un último inconveniente a la hora de interactuar con el arduino, como mencionamos, el manejo de la pantalla se hace a través de tres pulsadores, un pulsador para avanzar a la izquierda, otro para la derecha y uno para aceptar o salir de la pantalla de configuración.

El problema que tuvimos fue que al presionar el pulsador, el arduino no siempre reconocía de forma adecuada el pulso, investigando en internet encontramos que cuando se presiona o se suelta el pulsador, se produce una fluctuación entre sus contactos internos. Estas fluctuaciones, denominadas Rebote, son también leídas por el Arduino y producen un comportamiento inesperado en el funcionamiento de nuestro proyecto.

Para solucionar esto abarcamos dos enfoques, uno de hardware y otro de software:

* Software:

Una solución que encontramos para solventar este problema fue la utilización de un retardo para leer el estado del pulsador, la implementación del mismo puede ser un poco tediosa, por lo que buscando encontramos una librería que nos permite hacer esto de forma sencilla: <https://github.com/thomasfredericks/Bounce2>

La inicialización no es muy compleja, creamos una clase Bounce2 para cada botón, y en el setup le enlazamos el pin donde se encuentra el pulsador y el modo en el que está conectado, en nuestro caso INPUT\_PULLUP, luego le seteamos un tiempo de retardo al pulsador de 25ms, según varias pruebas realizadas.

Para utilizar esta clase, lo que uno hace para preguntar si se presiono o no el botón es primero actualizar el botón definido en la clase Bounce y luego preguntar si está presionado, un ejemplo es el siguiente:

void checkEnter(){

botonCen.update();

if (botonCen.pressed()) {

//Hacer algo

}

}

* Hardware:

Otra solución adicional que se puede agregar es utilizando un capacitor en paralelo al pulsador, las conexiones de los pulsadores se hicieron como PULL-UPS, por lo que el micro siempre está leyendo un 1 y al pulsar el mismo cae a 0.

El capacitor lo que permite es reducir el ruido generado por el bouncing (rebote) del pulsador, para que el micro reciba el valor correspondiente.

En la siguiente imagen podemos observar el ruido que se produce al pulsar un botón, esta imagen fue sacada de internet debido a que no contamos con un osciloscopio en nuestro hogar y no tenemos acceso a los laboratorios del Labi que se encuentran en la Facultad de Ingeniería, debido a la pandemia global.



Para que el micro no lea estos errores de bouncing, utilizamos el capacitor previamente mencionado.

# 

# FUNCIONAMIENTO DEL PROGRAMA

El diseño realizado para el proyecto cuenta con diferentes partes, como mencionamos tenemos un display en el que mostraremos diferentes pantallas, con los pulsadores avanzaremos por las mismas y con el pulsador del centro entraremos a la configuración de los umbrales mínimos y máximos de las temperaturas y humedades.

Las siguientes imágenes son las cuatro pantallas que se muestran en el programa, siendo la primera la pantalla principal:

## Pantalla principal

## 

## Pantalla de sensores

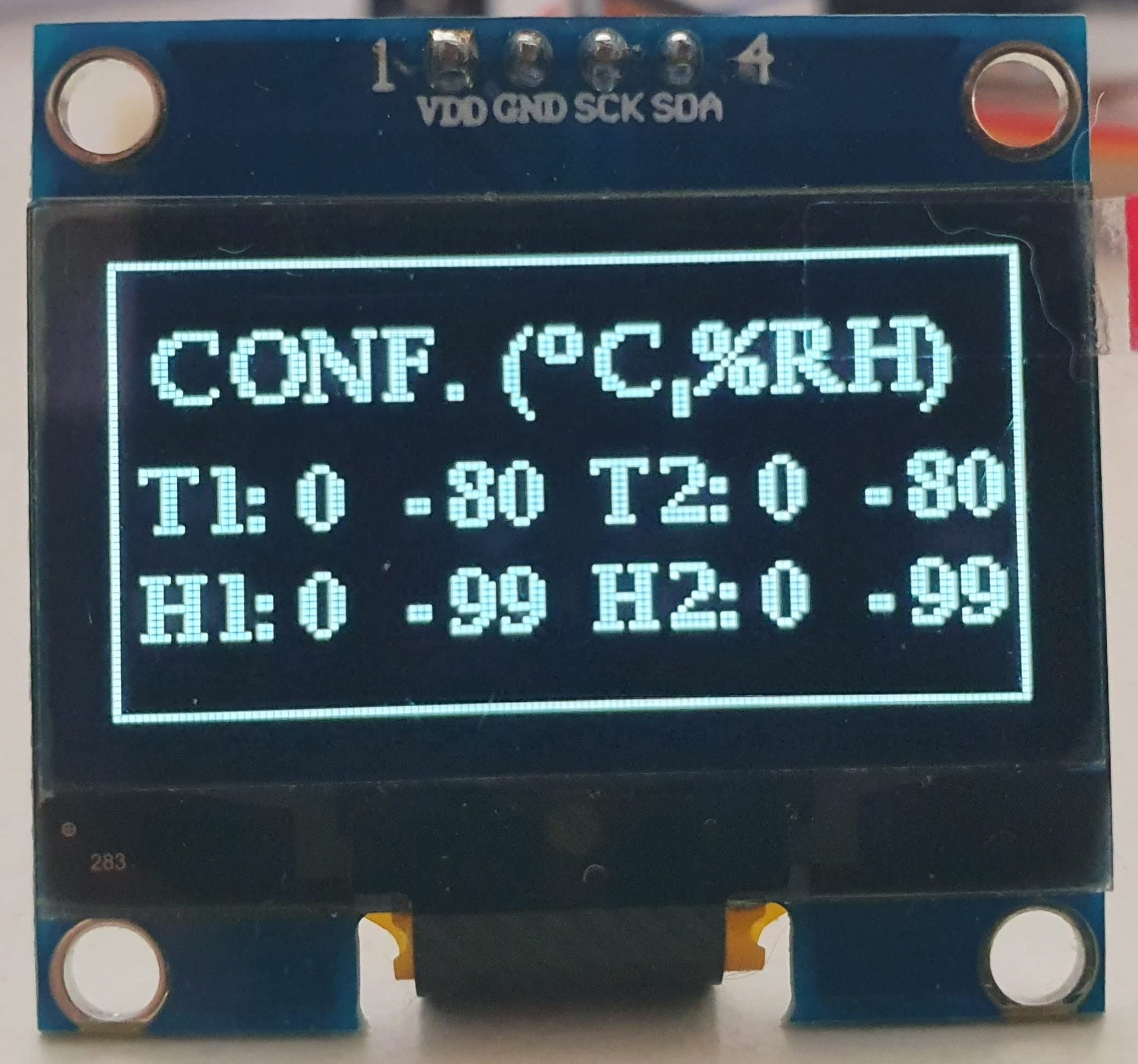
## 

## Pantalla de relés

## 

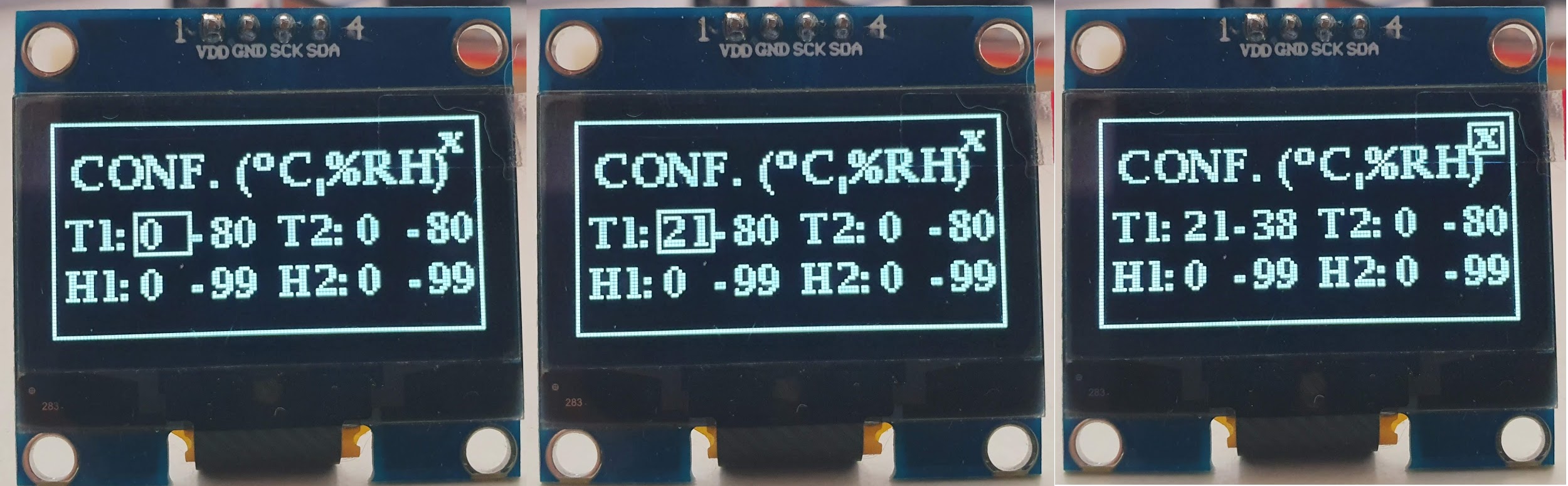
## 

## Pantalla de configuración

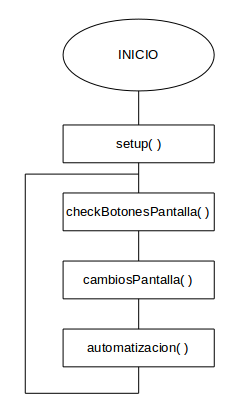


En esta última, podemos movernos entre los diferentes umbrales mínimos y máximos y modificar los valores, los umbrales de temperatura podrán ir desde 0 hasta 80 ºC y los valores de humedad de 0 a 100%RH.

Las siguientes pantallas muestran como se puede establecer los umbrales:



En el loop() se escribe nuestro programa principal, el mismo funciona de la siguiente manera:



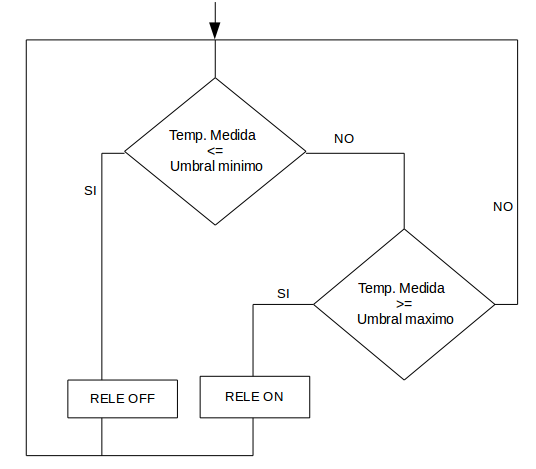
La función “checkBotonesPantalla()” estará preguntando si se pulsa alguno de los pulsadores de izquierda, derecha o centro, si se pulsa uno de derecha o izquierda se aumentará una variable que indica que se cambió de pantalla.

cambioPantalla() mostrará las distintas pantallas dependiendo de la variable mencionada, para ello utilizamos un switch, cada valor 0,1,2 y 3 imprimirá una pantalla diferente.

Si se pulsa el botón central cuando estamos en la pantalla de configuración entramos en la configuración de los umbrales mínimos y máximos. Estos valores dependen del sensor, en nuestro caso (DHT22) la temperatura puede variar entre 0 y 80ºC y la humedad entre 0 y 100 %RH.

Finalmente la función “automatizacion()” será la encargada de activar/desactivar los relés, no obstante la misma chequeara si los umbrales están bien establecidos (umbralMin < umbralMax), en caso de no serlo nos redirigirá a la pantalla de configuración.

El funcionamiento del mismo es el siguiente:



Pedirá la temperatura al sensor y preguntará si es menor o igual al umbral mínimo, si lo es, apagará el relé, si no pregunta si la temperatura medida es mayor al umbral máximo, en caso de ser mayor, activará el relé.

De esta forma la lámpara conectada al NC del relé estará siempre prendida, cuando alcanzamos la temperatura del umbral máximo, activará el relé apagando la lámpara y encendiendo el ventilador, haciendo que la temperatura comience a bajar, cuando llegue al umbral mínimo el relé se apagará, encendiendo nuevamente la luz, manteniendo este ciclo en forma continua, de esta forma obtendremos un ambiente controlado por temperatura.

Esta lógica también se repite para la humedad, y nuevamente si conectamos otro sensor se repetirá para el segundo sensor. En el caso de la humedad se realizará el mismo procedimiento conectando otros dispositivos.

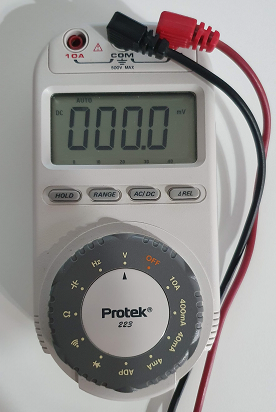
## Video demostrativo de su funcionamiento

<https://youtu.be/bfL6o9PeQtk>

# 

# MEDICIONES EN EL CIRCUITO

En este apartado realizaremos diferentes mediciones con el objetivo de corroborar que el proyecto funciona como se espera, para ello realizamos algunas mediciones de tensiones y corrientes con un multímetro digital.



Según las especificaciones del multímetro digital empleado en el trabajo (Protek 223) la incertidumbre de clase será: para un alcance de 400mV es de y para un alcance de 4V, 40V, 400V, 1000V es de . Por otro lado, como estamos trabajando con un multimetro digital no existirá la incertidumbre de apreciación.

El instrumento es de dígitos lo que nos indica que el multímetro tiene 4000 cuentas.

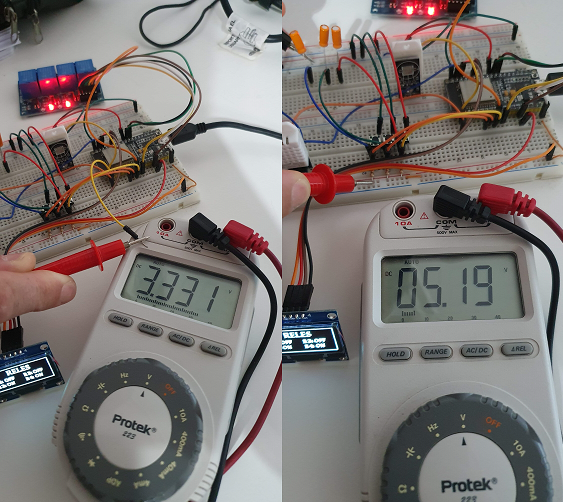
Finalmente consideramos que el error de carga es despreciable, la resistencia interna que tiene el tester es de 10 MΩ para los valores de 4V hasta los 1000V y posee una resistencia de 100 MΩ para un alcance de 400mV.

Si bien el efecto de carga siempre existe ya que aparece cuando introducimos el instrumento que va a realizar una medición, los valores medidos son muy precisos y los errores de carga fueron menores al 1%.

## Medición para corroborar las tensiones y masas del arduino

Como primer paso, para saber si el arduino está funcionando correctamente medimos las diferentes caídas de tensión en los pines del arduino, el mismo cuenta con un pin de 5V y otro de 3,3V, el primero se utiliza para alimentar los relés mientras que el segundo para alimentar a los sensores de temperatura y humedad.

Para realizar la medición colocamos la punta negativa en masa y la positiva en los pines del arduino. Debido a que estos están muy cerca uno de otros utilizamos un pequeño cable para apoyar la punta positiva y no generar ningún cortocircuito.



*Medición de 3,3V y 5V del Arduino*

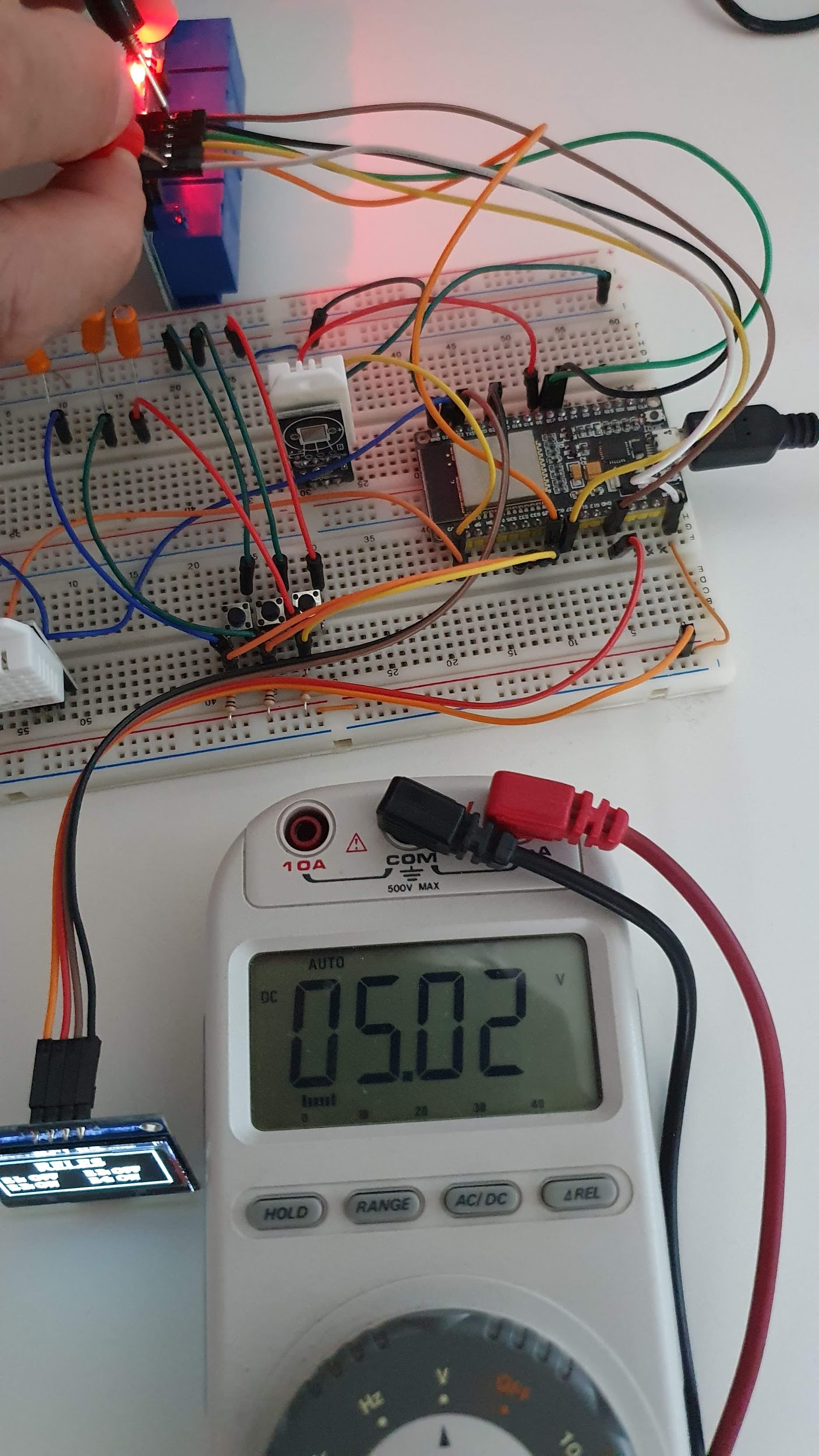
Vamos a medir la incertidumbre en primer lugar para el caso de los 5V, para este caso como nos pasamos del alcance de 4V tenemos que usar el de 40 V por lo tanto, al instrumento tener 4000 cuentas, se podrían medir 39.99V por lo que su último dígito representativo (udr) será 0.01 V. Por lo tanto:

Para el segundo caso en el que se mide con 3,3 V podemos usar el alcance de 4 V, por lo tanto, al instrumento tener 4000 cuentas, se podrían medir 3,999 V por lo que su último dígito representativo (udr) será 0.001 V. Por lo tanto:

En la medición de los 5V se observan debido a la dispersión que tiene el USB de la computadora, el cual entrega 5,2V, mientras que observamos en el pin de 3,3V.

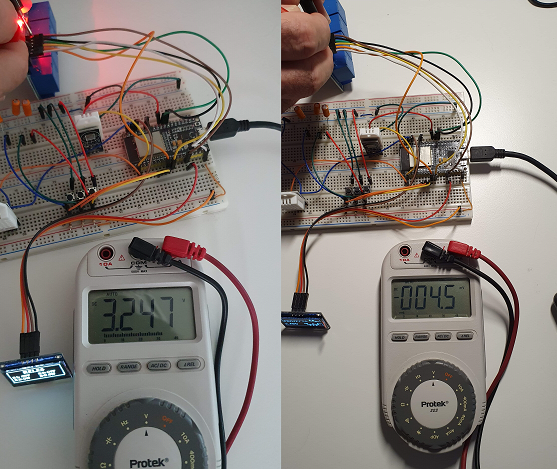
## Medición de tensión en los relés

Para medir los relés, lo hicimos directamente en la entrada de cada relé, en este ejemplo tenemos 2 relés encendidos y 2 relés apagados, y medimos uno de los apagados y otro de los encendidos. El módulo de cuatro relés se alimenta con 5V, y se medirá 3,3V si está encendido y 0V si está apagado.



*Alimentación del módulo de relés con 5V*

Vamos a medir la incertidumbre en primer lugar para la alimentación de los 5V, para este caso como nos pasamos del alcance de 4V tenemos que usar el de 40 V por lo tanto, al instrumento tener 4000 cuentas, se podrían medir 39.99V por lo que su último dígito representativo (udr) será 0.01 V. Por lo tanto:



*Tensión de 3,3V cuando el relé está encendido vs ≈ 0V cuando está apagado*

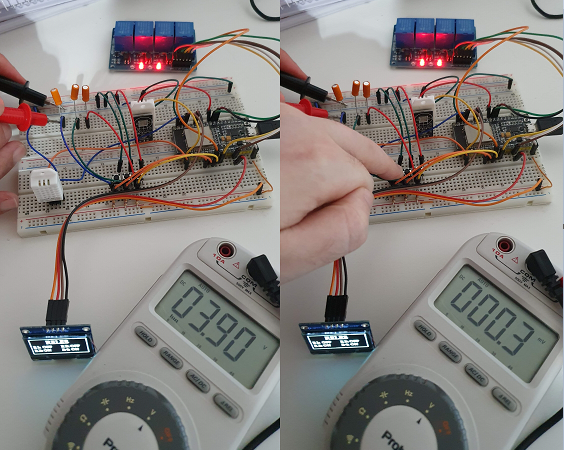
Para el caso en el que se mide con 3,3 V podemos usar el alcance de 4 V, por lo tanto, al instrumento tener 4000 cuentas, se podrían medir 3,999 V por lo que su último dígito representativo (udr) será 0.001 V. Por lo tanto:

Si ahora medimos para el caso de los 0V, podemos usar el alcance de 400 mV, por lo tanto, al instrumento tener 4000 cuentas, se podrían medir 0,3999 V por lo que su último dígito representativo (udr) será 0.0001 V. Por lo tanto:

Obtuvimos un valor de cuando el relé está encendido y cuando el relé está apagado.

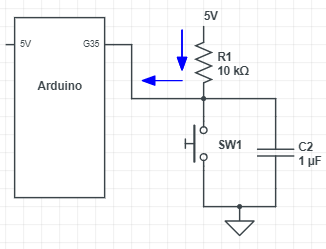
## Medición de tensión en los pulsadores

Finalmente probamos que los pulsadores funcionan como corresponde, en nuestro caso tenemos pulsadores conectados como PULL UP mediante una resistencia de 10KΩ a 5V. En estado de reposo (sin presionar), el Arduino debe interpretar un estado alto o “1 lógico” y cuando presionamos el pulsador la entrada del arduino pasará a estado bajo o “0 lógico”.



*Podemos observar en la imagen de la izquierda (en reposo, estado alto o 1 “lógico”) que la medición es de aproximadamente 3,9V, y en la imagen de la derecha al pulsar el microswitch (estado bajo o 0 “lógico”) la medición pasa a 0V.*

Al analizar la medición del porqué el 1 lógico es de 3,9 V, observamos que hay una corriente que ingresa al Arduino de 0,124 mA y eso provoca una caída de tensión sobre la resistencia de 10 KΩ, segun:



Realizamos dicha medición de corriente con el tester digital, obteniendo .

Según las especificaciones del multímetro digital empleado en el trabajo (Protek 223), para este instrumento no existirá la incertidumbre de apreciación y la incertidumbre de clase será: para un alcance de 4mA, 40mA, 400mA es de y para un alcance de 10A es de .

El instrumento es de dígitos lo que nos indica que el multímetro tiene 4000 cuentas.

Realizando los cálculos obtenemos:



Ahora, si calculamos la caída de tensión en la resistencia por la ley de Ohm:

Vr = Ir.R = 10KΩ . 0,124mA = 1,24V.

Por ende, si la tensión de alimentación es de 5,2V y hay una caída de tensión en la resistencia de 1,24V, la tensión a medir es de 3,96V como se puede observar en el instrumento, puesto que 5,2V - 1,24V = 3,96V. (Como se observa en la primera imágen).

## Medición con Osciloscopio

Si bien no contamos con un osciloscopio, si tuviésemos uno podremos realizar varias mediciones:

La primera medición que se podría hacer es medir el pulsador y la respuesta al rebote que tiene, para ello conectaremos la punta x1 a un extremo del pulsador y la otra punta a masa. Si bien la punta x1 tiene un efecto de carga por la capacidad del cable, como estamos operando a baja frecuencia el efecto de carga es bajo.

Otra medición que puede hacer es medir la señal del clock del Arduino, en este caso deberemos utilizar la punta x10 ya que estamos midiendo una frecuencia alta, no obstante hay que tener en cuenta que la medición tendrá una amplitud 10 veces más chica. Se apoyará la punta x10 en la pata CLOCK del integrado y el cable negro en GND.

También podemos medir los diferentes valores de tensión, no obstante como estamos trabajando en continua esta medición no tiene mucha utilidad.

# EJEMPLOS DE APLICACIONES, BENEFICIOS Y REFERENCIAS DE OTROS TRABAJOS

Si bien los ejemplos de aplicación podrían ser muchos, el objetivo principal con el que pensamos el proyecto fue el de un jardín o huerta inteligente para el hogar.

Los beneficios que esto conlleva son muchos, pero podemos nombrar como uno de los principales la idea de tener una huerta sin la necesidad de estar constantemente verificando que todas las condiciones sean adecuadas para el desarrollo de la misma.

El tener todos o la mayoría de las funcionalidades automatizadas nos ayuda en dos grandes aspectos, en primer lugar al no tener que estar controlando constantemente los vegetales, ganaríamos mucho tiempo el cual podemos emplear en otras cosas.

Por otro lado, al ya tener un patrón definido de cómo debe actuar frente a cada situación que se presente, es menos probable que los cultivos de la huerta se echen a perder ya que los mismos estarían en condiciones ideales en todo momento.

Otro de los grandes beneficios es que se disminuye radicalmente el consumo de agua en comparación con el cultivo convencional ya que se logra dosificar en forma automática la cantidad necesaria, frente al riego en forma manual.

Hoy en día la idea de huerta inteligente ya está más que instalada en el mundo, y pueden ser vistas desde en casas de pequeños agricultores hasta en las empresas más importantes del mundo como por ejemplo Walt Disney World, que en unos de sus parques temáticos (Epcot), una de las atracciones consiste en recorrer una huerta inmensa cultivada mediante métodos modernos en los que se busca que el hombre tenga la menor interacción posible con los alimentos.

También existen grandes agricultores que basan su producción en el uso de estas huertas inteligentes como por ejemplo la empresa brasileña Pink Farms.

# 

# CONSIDERACIONES A FUTURO

Si bien la funcionalidad del proyecto nos parece amplía, en un futuro nos gustaría desarrollar una aplicación móvil la cual nos permita realizar estas mismas acciones que venimos hablando hasta el momento pero de forma remota, es decir mediante una conexión WiFi o algo similar, poder manipular los diferentes umbrales y visualizar todo los parámetros desde el celular o computadora sin estar cerca del aparato físico, esto nos aportaría una gran comodidad a la hora de manipular el artefacto en cuestión.

También nos interesaría agregar mayores funcionalidades al proyecto, tales como medidores de dióxido de carbono, sistema de riego automático, medición de oxígeno, entre otras cosas.

Otra mejora para implementar a futuro es utilizar un relé que actúa como corte del suministro de 220V de los dispositivos eléctricos (lampara, ventilador, humidificador, etc). De esta forma podemos comenzar nuestro programa con este relé apagado y que solo se encienda si cumple ciertas condiciones establecidas.

Consideramos que estas adaptaciones enriquecerán el proyecto, ya que aprovechando los mismos sensores de temperatura y humedad, y agregando algún que otro más, tendríamos que realizar adaptaciones en el programa y el circuito electrónico para poder dejarlas funcionales.

# 

# AGRADECIMIENTOS

Queríamos agradecer a los profesores y al espacio y conocimiento generado por esta materia que nos permite llevar a cabo un proyecto como el que estamos realizando.

# REFERENCIAS

Arduino: <https://www.arduino.cc/>

ESP32: <https://github.com/espressif/arduino-esp32>

Display Oled: <https://github.com/olikraus/u8g2/>

Bounce: <https://github.com/thomasfredericks/Bounce2>

Sensor TyH: <https://github.com/beegee-tokyo/DHTesp>

DATASHEET - Tester Protek 223:

<http://www.maporo.com/body/body-1/mitek/protek-223.htm>

DATASHEET - DHT22 AM2302:

<https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf>

DATASHEET - Arduino ESP32 WROOM

* Modelo ESP32: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>
* Modelo ESP32 E & UE: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf>
* Modelo ESP32 SE: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32se_datasheet_en.pdf>

Pink Farms: <https://www.pinkfarms.com.br/>

# APÉNDICE

## COMPONENTES Y HERRAMIENTAS

* (1) Arduino NODEMCU32 ESP-WROOM-32.
* (2) Sensor de humedad relativa y temperatura - DHT22 AM2302
* (1) Display OLED 1.3 Blanco de 128x64 - SH1106 I2C.
* (1) Módulo de 4 Reles Octoacopladores (Bobina de 5V - 220V/10A).
* (3) Pulsadores.
* (3) Resistencias de 10KΩ.
* (3) Capacitores de 1µF.
* (2) Protoboard.
* (1) Cable USB de tipo B.
* (1) Manojo de cables.
* (1) Multímetro digital - Protek 223

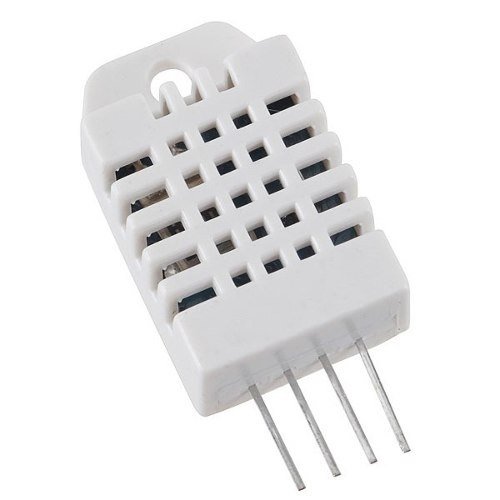
Entre las herramientas, utilizamos una computadora para programar el arduino mediante la IDE Arduino, la misma se puede obtener en la página web oficial <https://www.arduino.cc/>. En esta plataforma hay documentación, tutoriales y mucha información que nos puede hacer de base para comenzar a programar, incluso hay un sector de APIs para manejar IoT desde la nube.

Desde la página web podemos bajar la IDE de Arduino para comenzar a programar, la utilización de la aplicación es bastante sencilla y tiene muchos ejemplos con los que podemos utilizar de guía para comenzar nuestro desarrollo.

Fotos de los componentes principales utilizados



*Arduino NODEMCU32 ESP-WROOM-32*

**

*Sensor de humedad relativa y temperatura - DHT22.*

****

*Display OLED 1.3 Blanco de 128x64 - SH1106 I2C.*

****

*Módulo de 4 Reles Octoacopladores (5V - 220V/10A).*

## HOJA DE DATOS UTILIZADAS

Utilizamos la hoja de datos (datasheet) del Arduino y el sensor de temperatura y humedad DHT22 AM3202

DATASHEET - DHT22 AM2302:

<https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf>

DATASHEET - Arduino ESP32 WROOM

* Modelo ESP32: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>
* Modelo ESP32 E & UE: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf>
* Modelo ESP32 SE: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32se_datasheet_en.pdf>

## CÓDIGO EN ARDUINO

|  |
| --- |
| *//PROYECTO FINAL LABO*  *// NO USAR PATAS* *//CLOCK* *//SDO* *//SD1* *//S02* *//S03* *//CMD(tachado)*  #include <DHTesp.h> #include <U8g2lib.h> #include <Bounce2.h>  *//Reles de salidas* #define RELE\_1 16 #define RELE\_2 17 #define RELE\_3 26 #define RELE\_4 25  *//Botones pantalla* #define BOTON\_DER 33 #define BOTON\_CEN 32 #define BOTON\_IZQ 35  *//Pines sensores* #define SENSOR\_1 5 #define SENSOR\_2 23  *//LOGO - Mapa de Bits* #define LOGO\_ANCHURA 58 #define LOGO\_ALTURA 58  *//Defino las alturas de mis dos renglones para mostrar datos* #define RENGLON\_1 38  #define RENGLON\_2 53  *//Defino el comienzo de mis columnas (posiciones x)* #define COL\_1 4 #define COL\_2 67  *//Defino la posicion de una x (cruz) para salir de la configuracion* #define POSX\_X 116 #define POSY\_X 22  *//Defino los parametros maximos de las mediciones de los sensores* #define MIN\_HUM 0 #define MAX\_HUM 99 #define MIN\_TEM 0 #define MAX\_TEM 80  *//Defino retardo de pulsadores* #define RETARDO\_PUL 25  *//Defino nombre de las pantallas principales* #define PANT\_INICIO 0 #define PANT\_SENSOR 1 #define PANT\_RELE 2 #define PANT\_CONFIG 3  *//Defino el estado inicial y final de la pantalla de configuracion* #define \_POS\_CONFIG\_INI 0 #define \_POS\_CONFIG\_FIN 8  *//Defino un retraso para estabilizar la pantalla* #define DELAY\_ESTABILIZADOR 500  *//PINES DISPLAY OLED* *// SDA G21* *// SCK G22*  *//Constructor OLED* U8G2\_SH1106\_128X64\_NONAME\_F\_HW\_I2C u8g2(U8G2\_R0,U8X8\_PIN\_NONE);  *//Defino las clases BotonCen/BotonIzq/BotonDer utilizada para botones con rebote* Bounce2::Button botonCen = Bounce2::Button(); Bounce2::Button botonDer = Bounce2::Button(); Bounce2::Button botonIzq = Bounce2::Button();  *//Defino clase DHTesp* DHTesp dht1; DHTesp dht2;  *//Defino estructura para los sensores* struct Sensor {  String tipo;  int posUmax;  int posUmin;  int posY;  int umbralMax;  int umbralMin; };  *//Iniciailizo los sensores con sus posicones en pantalla y umbral Max y Min* Sensor temp1={"Temperatura 1",47,26,RENGLON\_1,MAX\_TEM,MIN\_TEM}; Sensor temp2={"Temperatura 2",112,91,RENGLON\_1,MAX\_TEM,MIN\_TEM}; Sensor hum1={"Humedad 1",47,26,RENGLON\_2,MAX\_HUM,MIN\_HUM}; Sensor hum2={"Humedad 2",112,91,RENGLON\_2,MAX\_HUM,MIN\_HUM};  *//Lista con la posicion de memoria de los sensores ordenados, utilizada para actualizar los valores* Sensor \*listaSensores[8] = {&temp1,&temp1,&hum1,&hum1,&temp2,&temp2,&hum2,&hum2};  *//ICONO en XBM* static const unsigned char logo[] PROGMEM = {  0x00, 0x00, 0x00, 0x30, 0xc0, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78,  0xe0, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0xe0, 0x03, 0x00, 0x00,  0x00, 0x00, 0x00, 0x78, 0xe0, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78,  0xe0, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0xe0, 0x03, 0x00, 0x00,  0x00, 0x00, 0x00, 0x78, 0xe0, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78,  0xe0, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0xe0, 0x03, 0x00, 0x00,  0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0x01, 0x00, 0x00, 0x00, 0x00, 0xff,  0xff, 0xff, 0x01, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0x01, 0x00,  0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0xe0, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0xe0, 0xff, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x03,  0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x03, 0x00, 0x00, 0x00, 0x00,  0x00, 0xe0, 0xff, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x01,  0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0xe0, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0xe0, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x01, 0x00, 0x00, 0x00, 0x00,  0x00, 0xe0, 0xff, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x03,  0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x01, 0x00, 0x1e, 0x00, 0x00,  0x00, 0xe0, 0x01, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00,  0x00, 0x1e, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00, 0x00, 0x1e, 0x00, 0x00,  0x00, 0xe0, 0x01, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00,  0x00, 0x1e, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00, 0xfe, 0x1f, 0x00, 0x00,  0x00, 0xe0, 0xff, 0x01, 0xff, 0x1f, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x03,  0xff, 0x1f, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x03, 0xff, 0x1f, 0x00, 0x00,  0x00, 0xe0, 0xff, 0x03, 0xfe, 0x1f, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x01,  0x00, 0x1e, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00, 0x00, 0x1e, 0x00, 0x00,  0x00, 0xe0, 0x01, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00, 0xe0, 0x01, 0x00,  0x00, 0xfe, 0xff, 0xff, 0xff, 0xff, 0x01, 0x00, 0x00, 0xfe, 0xff, 0xff,  0xff, 0xff, 0x01, 0x00, 0x00, 0xfe, 0xff, 0xff, 0xff, 0xff, 0x01, 0x00,  0x00, 0xfe, 0xff, 0xff, 0xff, 0xff, 0x01, 0x00, 0x00, 0x00, 0x1f, 0x78,  0xe0, 0x03, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x78, 0xe0, 0x03, 0x00, 0x00,  0x00, 0x00, 0x1f, 0x78, 0xe0, 0x03, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x78,  0xe0, 0x03, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x78, 0xe0, 0x03, 0x00, 0x00,  0x00, 0x00, 0x1f, 0x78, 0xe0, 0x03, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x78,  0xe0, 0x03, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x78, 0xe0, 0x03, 0x00, 0x00,  0x00, 0x00, 0x0e, 0x30, 0xc0, 0x01, 0x00, 0x00 };  *//------------- RETARDOS UTILIZADOS PARA LA PANTALLA Y LA OBTENCION DE DATOS --------------------* *//Retardo al pedir los valores al sensor (los sensores necesitan 2000ms para obtener los valores)* unsigned long tiempoDHT1 = 0; unsigned long tiempoDHT2 = 0;  *//Retardo para hacer un clear de pantalla para valores de Temp. y Hum.* unsigned long tiempoTyH=0;  *//Retardo que maneja blink de la pantalla de inicio* unsigned long tiempoBlink=0;  *//Retardo para estabilizar la pantalla al imprimir los recuadros*  unsigned long delayEstabilizador = 0;  *//Retardo para obtener valores de sensores para la automatizacion* unsigned long tiempoEspera1 = 0; unsigned long tiempoEspera2 = 0;  *// ------- VARIABLES UTILIZADAS PARA MANEJAR DIFERENTES ESTADOS -------- (posicion de pantalla, pantalla de configuracion, si se presiono o no un boton)* *//Inicializo variable globar para el estado inicial de la pantalla* int estadoPantalla=PANT\_INICIO;  *//Inicializo el boton enter para entrar a la configuracion* bool estadoBotonEnter=false;  *//Valor entero que varia de 0 a 8 para ubicarme en las posiciones de la pantalla* int posPantallaConfig = 0;  *//Estado booleano para entrar/salir de la configuracion de la pantalla* bool salir = false;  *//Estado booleano para darle enter para aceptar cambios en los umbrales y para comenzar a modificar* bool estadoBotonConfig = false;  *//Defino 3 listas* *//Lista con los valores (umbral max y min) en pantalla ordenados como se recorren al moverse en la configuracion* int valoresPantalla[8]={temp1.umbralMin,temp1.umbralMax,hum1.umbralMin,hum1.umbralMax,temp2.umbralMin,temp2.umbralMax,hum2.umbralMin,hum2.umbralMax};  *//Listas con las posiciones X e Y de los umbrales, y la posicion de la cruz para salir* int posSensoresPantallaY[9]={temp1.posY,temp1.posY,hum1.posY,hum1.posY,temp2.posY,temp2.posY,hum2.posY,hum2.posY,POSY\_X}; int posSensoresPantallaX[9]={temp1.posUmin,temp1.posUmax,hum1.posUmin,hum1.posUmax,temp2.posUmin,temp2.posUmax,hum2.posUmin,hum2.posUmax,POSX\_X};  void setup() {    *//Seteo reles con salida*  pinMode(RELE\_1, OUTPUT);  pinMode(RELE\_2, OUTPUT);  pinMode(RELE\_3, OUTPUT);  pinMode(RELE\_4, OUTPUT);   *//Seteo el Debouncer a los botones*  botonCen.attach(BOTON\_CEN,INPUT\_PULLUP); *//Setea el debouncer con el boton y el modo*  botonCen.interval(RETARDO\_PUL); *//Setea intervalo de 25 ms*  botonCen.setPressedState(LOW); *// Indica que el estado LOW es cuando se presiono el boton*   botonDer.attach(BOTON\_DER,INPUT\_PULLUP);  botonDer.interval(RETARDO\_PUL);  botonDer.setPressedState(LOW);   botonIzq.attach(BOTON\_IZQ,INPUT\_PULLUP);  botonIzq.interval(RETARDO\_PUL);  botonIzq.setPressedState(LOW);   *//Estado inicial de los reles*  *//ESTADO HIGH -> Reles sin energizar (Estado inicial NC)*  digitalWrite(RELE\_1, HIGH);  digitalWrite(RELE\_2, HIGH);  digitalWrite(RELE\_3, HIGH);  digitalWrite(RELE\_4, HIGH);   *//Inicializo sensores*  dht1.setup(SENSOR\_1, DHTesp::DHT22);  dht2.setup(SENSOR\_2, DHTesp::DHT22);   *//Inicializo pantalla*  u8g2.begin(); }  void loop() {  checkBotonesPantalla();  cambioPantalla();  automatizacion(); }  *//Funcion que chequea los botones y avanza o retrocede dependiendo del input* int checkBotonesPantalla(){  botonCen.update();  if (botonCen.pressed() && estadoPantalla == PANT\_CONFIG) {  estadoBotonEnter=true;  *//Reseteo del buffer*  u8g2.clearBuffer();  u8g2.sendBuffer();  }  botonDer.update();  if (botonDer.pressed()) {  u8g2.clearBuffer();  estadoPantalla++;  if (estadoPantalla > PANT\_CONFIG){  estadoPantalla = PANT\_SENSOR;  }  }  botonIzq.update();  if (botonIzq.pressed()) {  u8g2.clearBuffer();  estadoPantalla--;  if (estadoPantalla < PANT\_SENSOR){  estadoPantalla = PANT\_CONFIG;  }  } }  *//Switcher que dependiendo el estadoPantalla imprime las diferentes pantallas* void cambioPantalla(){  switch (estadoPantalla) {  case PANT\_INICIO:  pantallaInicio();  break;  case PANT\_SENSOR:  pantallaSensores();  break;  case PANT\_RELE:  pantallaEstadoRele();  break;  case PANT\_CONFIG:  pantallaConfig();  break;  } }  *//Funcion que chequea si umbralMin < UmbralMax y viceversa* bool umbralEsValido(){  bool errorUmbral = true;  *//CHECK T1*  if (temp1.umbralMin >= temp1.umbralMax || temp2.umbralMin >= temp2.umbralMax || hum1.umbralMin >= hum1.umbralMax || hum2.umbralMin >= hum2.umbralMax){  errorUmbral = false;  }  return errorUmbral; }  *//Funcion encarga de automatizar todo el sistema* void automatizacion(){    *//Si el umbral no es valido, resetea*  if (!umbralEsValido()){  estadoPantalla = PANT\_CONFIG;  estadoBotonEnter = true;   }else{  unsigned long nuevoTiempo1 = millis();   *//SENSOR 1*  *//Obtiene el tiempo y se fija si paso el minimun sampling period, si es asi automatiza*  if ((nuevoTiempo1 - tiempoEspera1) > (dht1.getMinimumSamplingPeriod())){  tiempoEspera1 = nuevoTiempo1;  TempAndHumidity tempYhum1 = dht1.getTempAndHumidity();  automatizarReles(tempYhum1,RELE\_1,RELE\_2,temp1,hum1);  }   *//SENSOR2*  *//Obtiene el tiempo y se fija si paso el minimun sampling period, si es asi automatiza*  unsigned long nuevoTiempo2 = millis();  if ((nuevoTiempo2 - tiempoEspera2) > (dht2.getMinimumSamplingPeriod())){  tiempoEspera2 = nuevoTiempo2;  TempAndHumidity tempYhum2 = dht2.getTempAndHumidity();  automatizarReles(tempYhum2,RELE\_3,RELE\_4,temp2,hum2);  }  } }   *//Funcion que automatiza los reles* void automatizarReles(TempAndHumidity temperaturaYhumedad, int rele\_1,int rele\_2,Sensor temp, Sensor hum){  float temperatura = temperaturaYhumedad.temperature;  float humedad = temperaturaYhumedad.humidity;  int largo=2;    *//Incializo una lista con las temperaturas y humedades*  float tempYhum[largo] = {temperatura,humedad};   *//Creo 2 listas con los reles y sensores para recorrerlas*  int reles[largo] = {rele\_1,rele\_2};  Sensor sensor[largo] = {temp,hum};    for (int i=0; i <= largo; i++){     if (tempYhum[i] <= sensor[i].umbralMin){  digitalWrite(reles[i], HIGH);  }  else if (tempYhum[i] >= sensor[i].umbralMax){  digitalWrite(reles[i], LOW);  }  } }   *//Pantalla de inicio* void pantallaInicio(){  u8g2.drawFrame(0,0,128,64);  u8g2.drawXBMP(66,3,LOGO\_ANCHURA,LOGO\_ALTURA,logo);  u8g2.setFont(u8g2\_font\_logisoso16\_tf);  u8g2.drawStr(4,30,"PROYECTO");  u8g2.drawStr(4,54,"LABO");  flecha\_blink(); }  *//Funcion que hace blink de una flecha* void flecha\_blink(){  unsigned long nuevoTiempo = millis();  u8g2.setFont(u8g2\_font\_logisoso34\_tf);    if ((nuevoTiempo - tiempoBlink) > 500){  u8g2.drawStr(88,50," ");  u8g2.sendBuffer();  tiempoBlink = nuevoTiempo;  while((nuevoTiempo - tiempoBlink) < 500){  nuevoTiempo=millis();  }  tiempoBlink = nuevoTiempo;  }else{  u8g2.drawStr(88,50,">");  }  u8g2.sendBuffer(); }  *//Funcion que imprime la pantalla de los sensores* void pantallaSensores(){  clearBufferSensores(dht1);  u8g2.drawFrame(0,0,128,64);  u8g2.setFont(u8g2\_font\_lubB12\_tf);  u8g2.drawStr(20,20,"SENSORES");  u8g2.setFont(u8g2\_font\_lubB08\_tf);   *//T1 y H1*  u8g2.drawStr(COL\_1, RENGLON\_1, "T1: ");  u8g2.drawStr(COL\_1, RENGLON\_2, "H1: ");   *//T2 y H2*  u8g2.drawStr(COL\_2, RENGLON\_1, "T");  u8g2.setCursor(COL\_2+2+u8g2.getStrWidth("T"),RENGLON\_1);  u8g2.print("2: ");   u8g2.drawStr(COL\_2, RENGLON\_2, "H");  u8g2.setCursor(COL\_2+2+u8g2.getStrWidth("H"),RENGLON\_2);  u8g2.print("2: ");    *//Imprimir temperatura y humedad*  controlTempYhum1(dht1);  controlTempYhum2(dht2);  u8g2.sendBuffer(); }  *//Funcion que borra el dato de los sensores (esto se realiza para no tener el valor anterior cuando obtengo una nueva medicion)* void clearBufferSensores(DHTesp dht){  unsigned long nuevoTiempo=millis();  if ((nuevoTiempo - tiempoTyH) > (dht.getMinimumSamplingPeriod())){  u8g2.clearBuffer();  tiempoTyH=nuevoTiempo;  } }   *//Funcion que controla la humedad y temperatura del sensor 1* *//Obtiene nueva información cada 2000ms, el tiempo se maneja sin contador para no utilizar delay* void controlTempYhum1(DHTesp dht){  unsigned long nuevoTiempo = millis();    if ((nuevoTiempo - tiempoDHT1) > (dht.getMinimumSamplingPeriod())){  tiempoDHT1 = nuevoTiempo;  TempAndHumidity tempYhum = dht.getTempAndHumidity();  imprimirTemp(tempYhum.temperature,dht);  imprimirHum(tempYhum.humidity,dht);  } }   *//Funcion que controla la humedad y temperatura del sensor 1* *//Obtiene nueva información cada 2000ms, el tiempo se maneja sin contador para no utilizar delay* void controlTempYhum2(DHTesp dht){  unsigned long nuevoTiempo = millis();    if ((nuevoTiempo - tiempoDHT2) > (dht.getMinimumSamplingPeriod())){  tiempoDHT2 = nuevoTiempo;  TempAndHumidity tempYhum = dht.getTempAndHumidity();  imprimirTemp(tempYhum.temperature,dht);  imprimirHum(tempYhum.humidity,dht);  } }  *//Funcion que imprime la temperatura* void imprimirTemp(float temp,DHTesp dht){  u8g2.setFont(u8g2\_font\_lubB08\_tf);  if(dht.getPin() == SENSOR\_1){  u8g2.setCursor(u8g2.getStrWidth("T1: ")+COL\_1, RENGLON\_1);  }  else{  u8g2.setCursor(u8g2.getStrWidth("T2: ")+COL\_2+2, RENGLON\_1);  }  u8g2.print(temp,2); }  *//Funcion que imprime la humedad* void imprimirHum (float hum,DHTesp dht){  u8g2.setFont(u8g2\_font\_lubB08\_tf);  if(dht.getPin() == SENSOR\_1){  u8g2.setCursor(u8g2.getStrWidth("H1: ")+COL\_1-1, RENGLON\_2);  }  else{  u8g2.setCursor(u8g2.getStrWidth("H2: ")+COL\_2, RENGLON\_2);  }    u8g2.print(hum,2); }  *//Funcion que imprime la pantalla de los estados de los reles* void pantallaEstadoRele(){  u8g2.clearBuffer();  u8g2.drawFrame(0,0,128,64);  u8g2.setFont(u8g2\_font\_lubB12\_tf);  u8g2.drawStr(36,20,"RELES");  estadoRele();  u8g2.sendBuffer(); }  *//Funcion que obtiene los estados de los reles y los imprime por pantalla* void estadoRele(){  String estado1 = checkEstadoRele(RELE\_1);  String estado2 = checkEstadoRele(RELE\_2);  String estado3 = checkEstadoRele(RELE\_3);  String estado4 = checkEstadoRele(RELE\_4);   u8g2.setFont(u8g2\_font\_lubB08\_tf);   *//Impresion de R1 y su estado*  int posX1 = COL\_1+u8g2.getStrWidth("R1: ");  u8g2.drawStr(COL\_1, RENGLON\_1, "R1: ");  u8g2.setCursor(posX1, RENGLON\_1);  u8g2.print(estado1);   *//Impresion de R2 y su estado*  u8g2.drawStr(COL\_1, RENGLON\_2, "R");  u8g2.setCursor(COL\_1+2+u8g2.getStrWidth("R"),RENGLON\_2);  u8g2.drawStr(COL\_1+2+u8g2.getStrWidth("R"),RENGLON\_2,"2: ");  u8g2.setCursor(posX1, RENGLON\_2);  u8g2.print(estado2);   *//Impresion de R3 y su estado*  int posX2=COL\_2+u8g2.getStrWidth("R3: ");  u8g2.drawStr(COL\_2, RENGLON\_1, "R");  u8g2.drawStr(COL\_2+2+u8g2.getStrWidth("R"),RENGLON\_1,"3: ");  u8g2.setCursor(posX2, RENGLON\_1);  u8g2.print(estado3);   *//Impresion de R4 y su estado*   u8g2.drawStr(COL\_2, RENGLON\_2, "R");  u8g2.drawStr(COL\_2+1+u8g2.getStrWidth("R"),RENGLON\_2,"4: ");  u8g2.setCursor(posX2, RENGLON\_2);  u8g2.print(estado4); }  *//Funcion que devuelve ON si el rele esta prendido o OFF si esta apagado* String checkEstadoRele(int rele){  String estado = "OFF";  if (digitalRead(rele) == LOW){  estado = "ON";  }  else{  estado = "OFF";  }  return estado; }  *//Funcion que maneja la pantalla de configuracion* void pantallaConfig(){  mostrarConfig();   *//Si estadoBotonEnter esta en 1 solo cuando estas en pantalla 3 y presionas enter*  if (estadoBotonEnter){  estadoBotonEnter=false;  *// Loop de sistema de configuracion (aparece x y el cuadrado)*  sistemaDeConfig();   } }  *//Funcion que muestra la configuracion de los sensores de temperatura y humedad* void mostrarConfig(){  u8g2.drawFrame(0,0,128,64);  u8g2.setFont(u8g2\_font\_lubB10\_tf);  u8g2.drawStr(6,20,"CONF. (\xba");  u8g2.setCursor(8+u8g2.getStrWidth("CONF. (\xba"),20);  u8g2.print("C,%RH)");  u8g2.setFont(u8g2\_font\_lubB08\_tf);    *//T1*  u8g2.drawStr(COL\_1, RENGLON\_1, "T1: ");  u8g2.setCursor(temp1.posUmin, RENGLON\_1);  u8g2.print(temp1.umbralMin);  u8g2.drawStr(temp1.posUmax-6,RENGLON\_1,"-");  u8g2.setCursor(temp1.posUmax, RENGLON\_1);  u8g2.print(temp1.umbralMax);    *//T2*  u8g2.drawStr(COL\_2, RENGLON\_1, "T");  u8g2.setCursor(COL\_2+2+u8g2.getStrWidth("T"),RENGLON\_1);  u8g2.print("2: ");  u8g2.setCursor(temp2.posUmin, RENGLON\_1);  u8g2.print(temp2.umbralMin);  u8g2.drawStr(temp2.posUmax-6,RENGLON\_1,"-");  u8g2.setCursor(temp2.posUmax, RENGLON\_1);  u8g2.print(temp2.umbralMax);    *//H1*  u8g2.drawStr(COL\_1, RENGLON\_2, "H1: ");  u8g2.setCursor(hum1.posUmin, RENGLON\_2);  u8g2.print(hum1.umbralMin);  u8g2.drawStr(hum1.posUmax-6,RENGLON\_2,"-");  u8g2.setCursor(hum1.posUmax, RENGLON\_2);  u8g2.print(hum1.umbralMax);    *//H2*  u8g2.drawStr(COL\_2, RENGLON\_2, "H");  u8g2.setCursor(COL\_2+2+u8g2.getStrWidth("H"),RENGLON\_2);  u8g2.print("2: ");  u8g2.setCursor(hum2.posUmin, RENGLON\_2);  u8g2.print(hum2.umbralMin);  u8g2.drawStr(hum2.posUmax-6,RENGLON\_2,"-");  u8g2.setCursor(hum2.posUmax, RENGLON\_2);  u8g2.print(hum2.umbralMax);    u8g2.sendBuffer(); }  *//Sistema de configuracion, si se pulsa enter en las posiciones dentro de los umbrales se puede modificiar los valores* *//Si se pulsa enter en la cruz, sale de la configuracion* void sistemaDeConfig(){  *//Salir solo puede ser 1 si la posicon esta en X y presiono enter*  while(!salir){  *//tiempoNuevo se utiliza para hacer un retardo para mejorar la estabilidad de la impresion en el display*  unsigned long tiempoNuevo = millis();  u8g2.setCursor(POSX\_X,POSY\_X-11);  u8g2.print("x");  u8g2.sendBuffer();    if ((tiempoNuevo - delayEstabilizador) > (DELAY\_ESTABILIZADOR)){  *//LLama funcion recorrerValores para poder modificarlos*  recorrerValores();  delayEstabilizador = tiempoNuevo;   }  *//Chequea los botones izq,cent,der*   checkBotonesConfig();  }  *//Limpio Buffer y reseteo salir y estadoBotonConfig*  u8g2.clearBuffer();  salir=false;  estadoBotonConfig=false; }   *//Obtiene la posicion de X e Y de los valores de las listas dependiendo de la posPantallaConfig* *//Estas listas estan creadas de tal forma que se pueda acceder dependiendo de la posicion* void recorrerValores(){  int posicionX = posSensoresPantallaX[posPantallaConfig];   int posicionY = posSensoresPantallaY[posPantallaConfig];    imprimirRectangulo(posicionX,posicionY);  cambiarValoresTyH(posicionX,posicionY); }  *//Funcion que dibuja un cuadro dependiendo de la posicion en la que estemos* void imprimirRectangulo(int posX,int posY){  if (posPantallaConfig != \_POS\_CONFIG\_FIN){  u8g2.drawFrame(posX-2, posY-11,17,13);   }else{  u8g2.drawFrame(114,3,11,10);   } }  *//Funcion que se encarga de obtener los valores de los umbrales para luego modificarlos al llamar a sumaRestaUmbral* void cambiarValoresTyH(int posX, int posY){  *// Si estadoBotonConfig es 1, la pantalla posPantallaConfig != 8*  if (estadoBotonConfig){  int umbral = valoresPantalla[posPantallaConfig];   estadoBotonConfig = false;  sumaRestaUmbral(posX,posY,umbral);  }  }  *//Funcion encargada de manejar los umbrales, al pulsar enter, se modifican los valores de los umbrales en las estructuras* void sumaRestaUmbral(int posX, int posY, int umbral){  int nuevoUmbral = umbral;  while (!estadoBotonConfig){  *//Chequea si se pulsa el boton del centro (enter)*  checkEnter();  nuevoUmbral = modUmbral(posX,posY,nuevoUmbral);  u8g2.setCursor(posX, posY);  u8g2.print(nuevoUmbral);  u8g2.sendBuffer();  }  actualizarValoresSensor(nuevoUmbral);  *//Resetea el valor del booleano estadoBotonConfig (para cambiar valores de umbral)*  estadoBotonConfig = false; }  *//Funcion que chequea los botones y modifica el valor del umbral* *//Tambien chequea los MAX y MIN de la Temperatura y Humedad* int modUmbral(int posX, int posY, int umbral){    botonDer.update();  if (botonDer.pressed()) {  umbral++;  clearAreaUmbral(posX,posY);   *//SI ES TEMPERATURA sino es HUM*  if (RENGLON\_1 == posY){  if (umbral > MAX\_TEM){  umbral = MIN\_TEM;  }   }else{  if (umbral > MAX\_HUM){  umbral = MIN\_HUM;  }  }  }  botonIzq.update();  if (botonIzq.pressed()) {  umbral--;  clearAreaUmbral(posX, posY);   *//SI ES TEMPERATURA sino es HUM*  if (RENGLON\_1 == posY){  if (umbral < MIN\_TEM){  umbral = MAX\_TEM;  }   }else{  if (umbral < MIN\_HUM){  umbral = MAX\_HUM;  }  }  }  return umbral; }  *//Se dibuja un cuadrado en blanco para refresar los valores de los umbrales en pantalla* void clearAreaUmbral(int posX, int posY){  u8g2.setDrawColor(0);  u8g2.drawBox(posX-1,posY-10,15,10);  u8g2.sendBuffer();  u8g2.setDrawColor(1); }  *//Funcion que chequea si se pulso el boton del medio, permitiendo aceptar los cambios de los umbrales previamente modificados* void checkEnter(){  botonCen.update();  if (botonCen.pressed()) {  estadoBotonConfig = true;  } }   *//Funcion encargada de guardar los datos de los umbrales MAX y MIN* void actualizarValoresSensor(int umbral){  Sensor \*sensor = listaSensores[posPantallaConfig];  *//Si la posicion es 0 o Par, entonces cambio el umbral minimo, sino el maximo*  if ((posPantallaConfig == \_POS\_CONFIG\_INI) || ((posPantallaConfig%2) == 0)){  sensor->umbralMin = umbral;  }else{  sensor->umbralMax = umbral;  }  *//Actualizo la lista de los valores obtenidos*  valoresPantalla[posPantallaConfig] = umbral; }  *//Refresca los valores de los umbrales* void refrescoCuadro(){  int posX = posSensoresPantallaX[posPantallaConfig];   int posY = posSensoresPantallaY[posPantallaConfig];  u8g2.setDrawColor(0);  if (posPantallaConfig != \_POS\_CONFIG\_FIN){  u8g2.drawFrame(posX-2, posY-11,17,13);  }else{  u8g2.drawFrame(114,3,11,10);   }  u8g2.setDrawColor(1); }  *//Funcion que chequea donde nos encontramos, y avanza o retrocede si se pulsa los botones derecha e izquierda* void checkBotonesConfig(){  botonCen.update();  if (botonCen.pressed() && posPantallaConfig == \_POS\_CONFIG\_FIN) {  salir = true;  posPantallaConfig = \_POS\_CONFIG\_INI;  }  if (botonCen.pressed() && posPantallaConfig != \_POS\_CONFIG\_FIN) {  estadoBotonConfig = true;  }  botonDer.update();  if (botonDer.pressed()) {  refrescoCuadro();  posPantallaConfig++;  if (posPantallaConfig > \_POS\_CONFIG\_FIN){  posPantallaConfig = \_POS\_CONFIG\_INI;  }  }  botonIzq.update();  if (botonIzq.pressed()) {  refrescoCuadro();  posPantallaConfig--;  if (posPantallaConfig < \_POS\_CONFIG\_INI){  posPantallaConfig = \_POS\_CONFIG\_FIN;  }  } } |