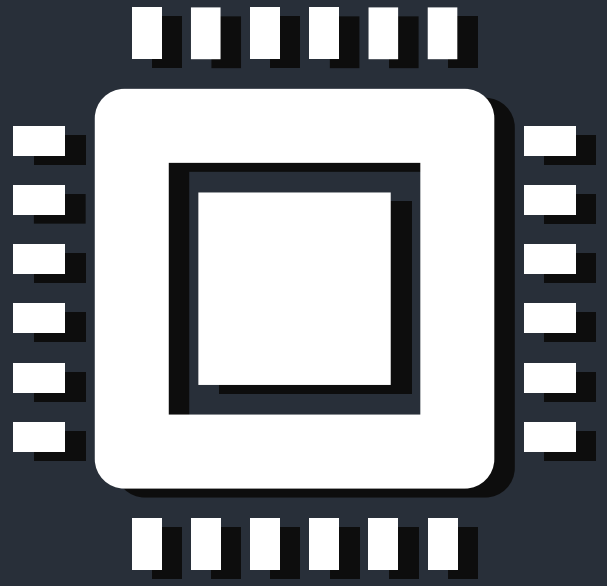


Sistema Contra Incendios

MICROCONTROLADORES
ESP8266 & ARDUINO





LLUVIA DE IDEAS

01 Estación meteorologica con alarmas de tiempo
-No soluciona algo importante

02 Control remote
-No tiene programación web

03 Sistema de alarma sobre incendios o gases

04 Cerradura electronica
-Muy parecido al Proyecto del led

LA PROPUESTA GANADORA

Sistema contra incendios y detector de gases

¿POR QUÉ?

- Es un Proyecto que puede salvar cosas materiales e incluso puede llegar a salvar vidas.
- Cumple con todos los requisitos del proyecto

FUNCIONES

- Se monitorea en todo momento tanto la temperatura como los niveles de gases que detecta el sensor MQ2.
- En caso de que algún nivel suba mucho se enciende la alarma y activa las alarmas sonoras, visual y También tiene poder para levantar un Sistema externo por ejemplo un Sistema de riego.

CASOS DE USO

- Detectar indendios para poder ser sofocados a tiempo
- Deteccion de fugas de gas licuado en estaciones de gas
- Deteccion de altas temperaturas
- Poder detector alto nivel de contaminacion en el ambiente
- Centro de comandos para un sistema de sofocación de fuego

LA CONSTRUCCIÓN DE LA IDEA

Antes de programar o comenzar a comprar todo, primero se tiene que hacer un análisis y para ello necesitamos:

01 Diagrama de flujo

02 Diagrama de bloques

03 Diagrama electronico

DIAGRAMA DE FLUJO

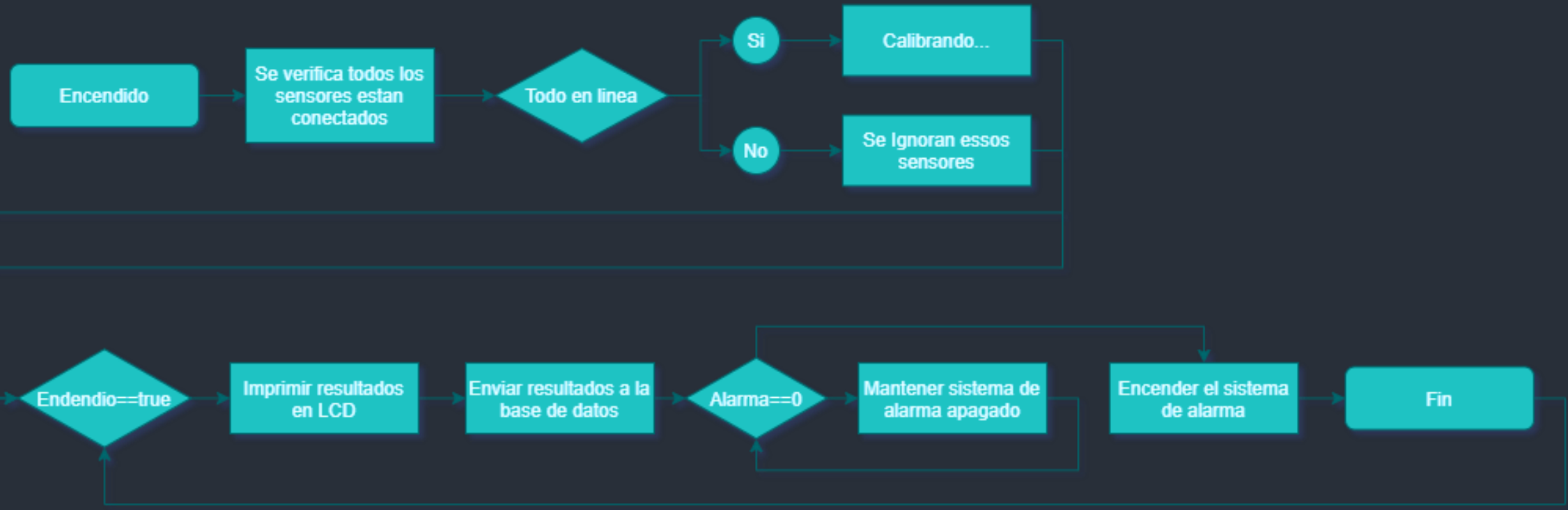


DIAGRAMA DE BLOQUES

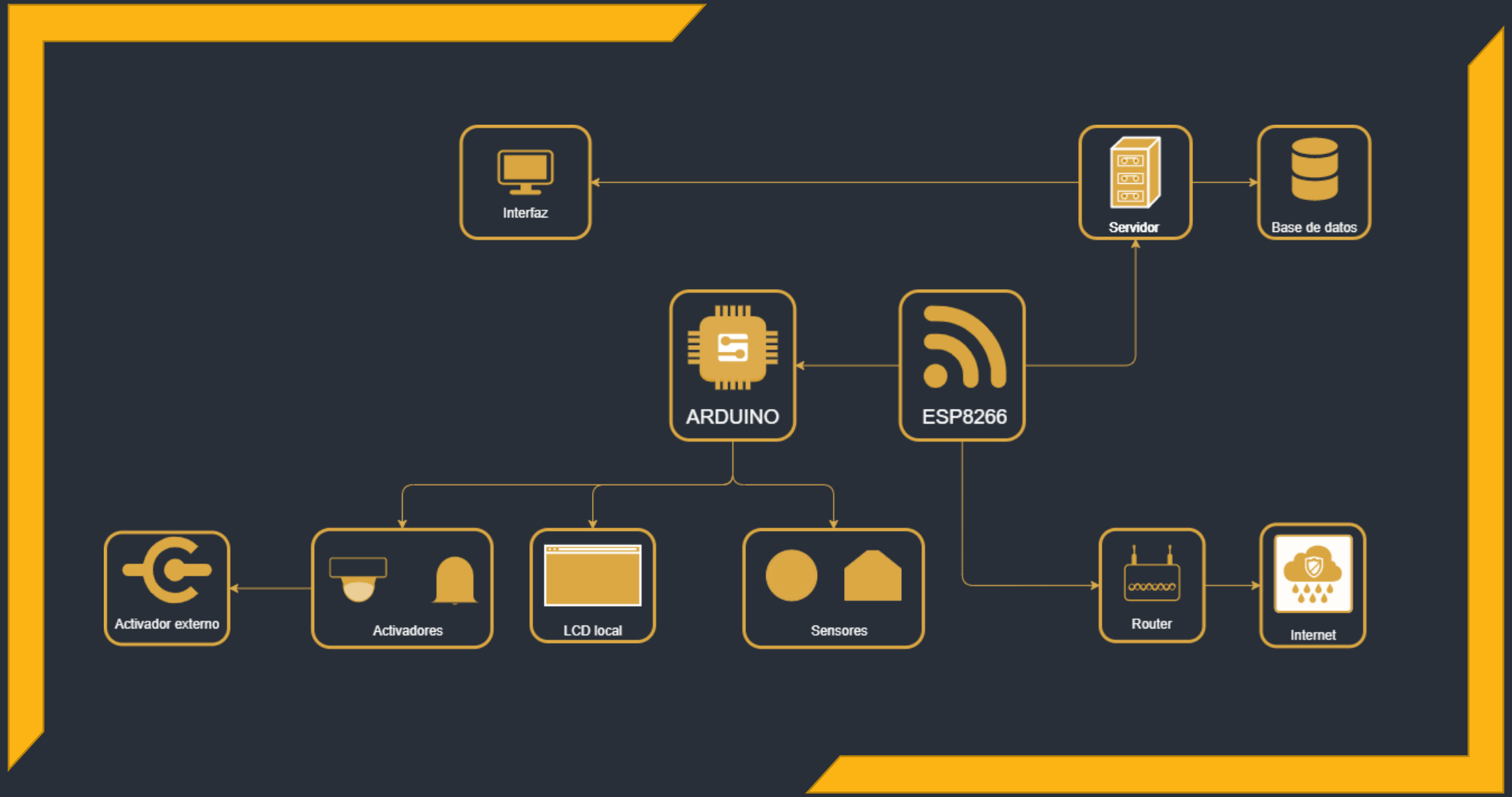
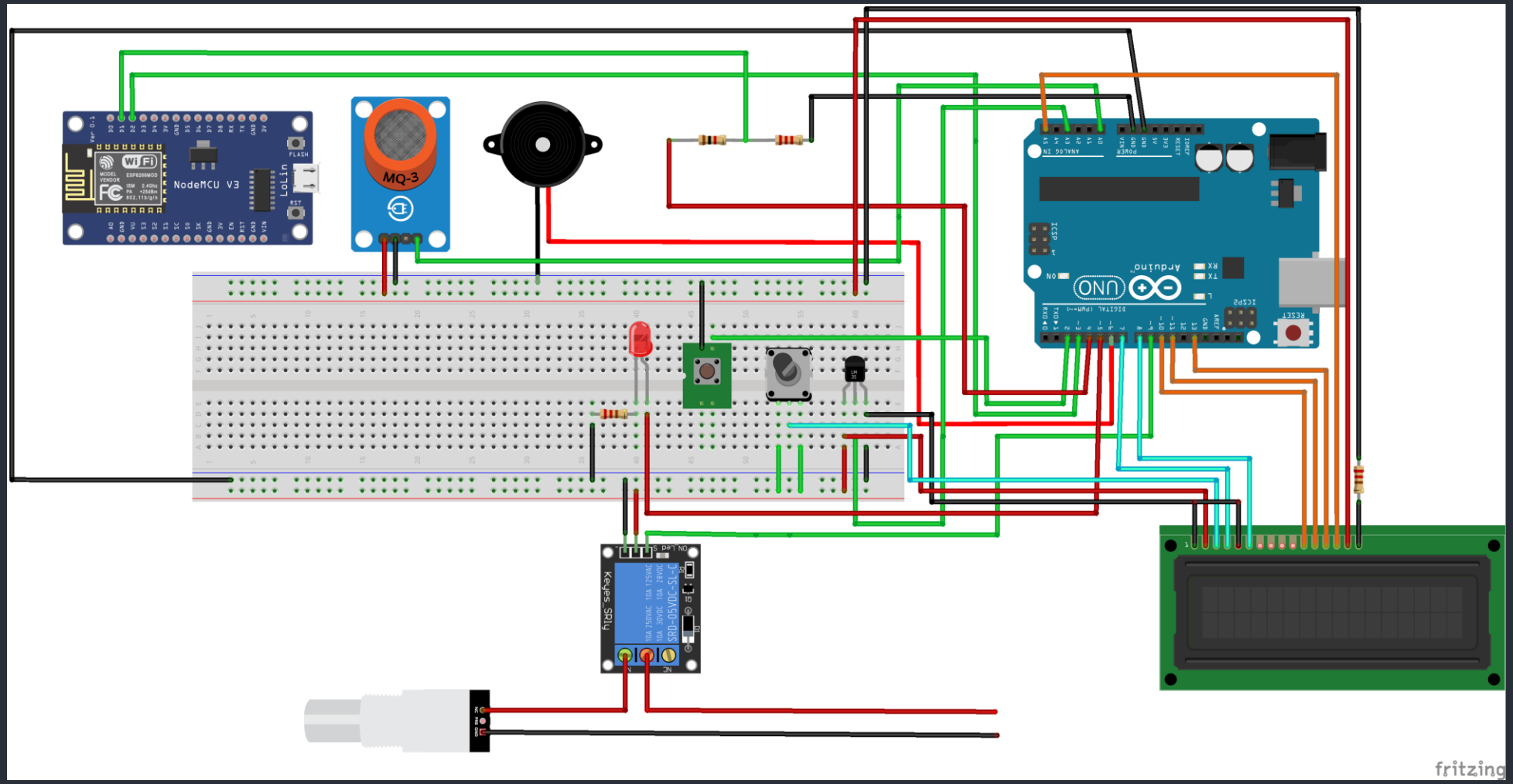
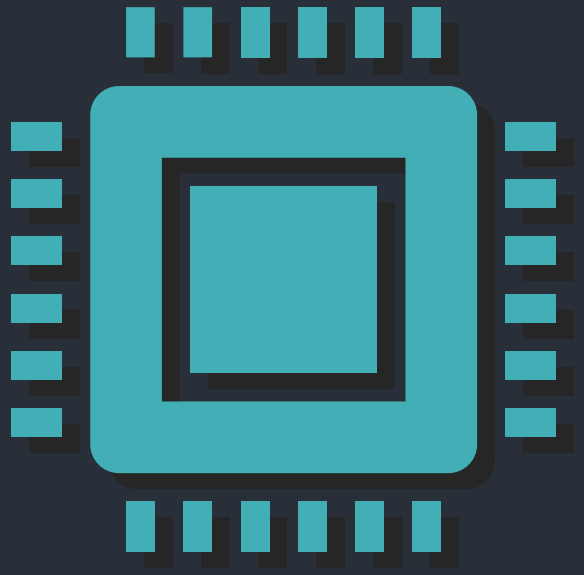


DIAGRAMA ELECTRÓNICO



PROGRAMACION Y CONTRUCCION

MATERIALES



- Microcontrolador Arduino UNO
- NodeMCU ESP8266 LUA
- Protoboard de 820 puntos
- Sensor de gases MQ2
- Sensor de temperatura LM35
- Display LCD 1602A
- Buzzer activo
- Relay 5V-110V
- Potenciometro 10k
- Tira de 40 pines macho recto
- Resistencias de 220 ohms
- Resistencias de 2.2K
- Resistencias de 1K
- Cables jumpers dupont 20 cm
- Caja de 30x15x20 cm
- Enchufe eléctrico
- Contacto electrico

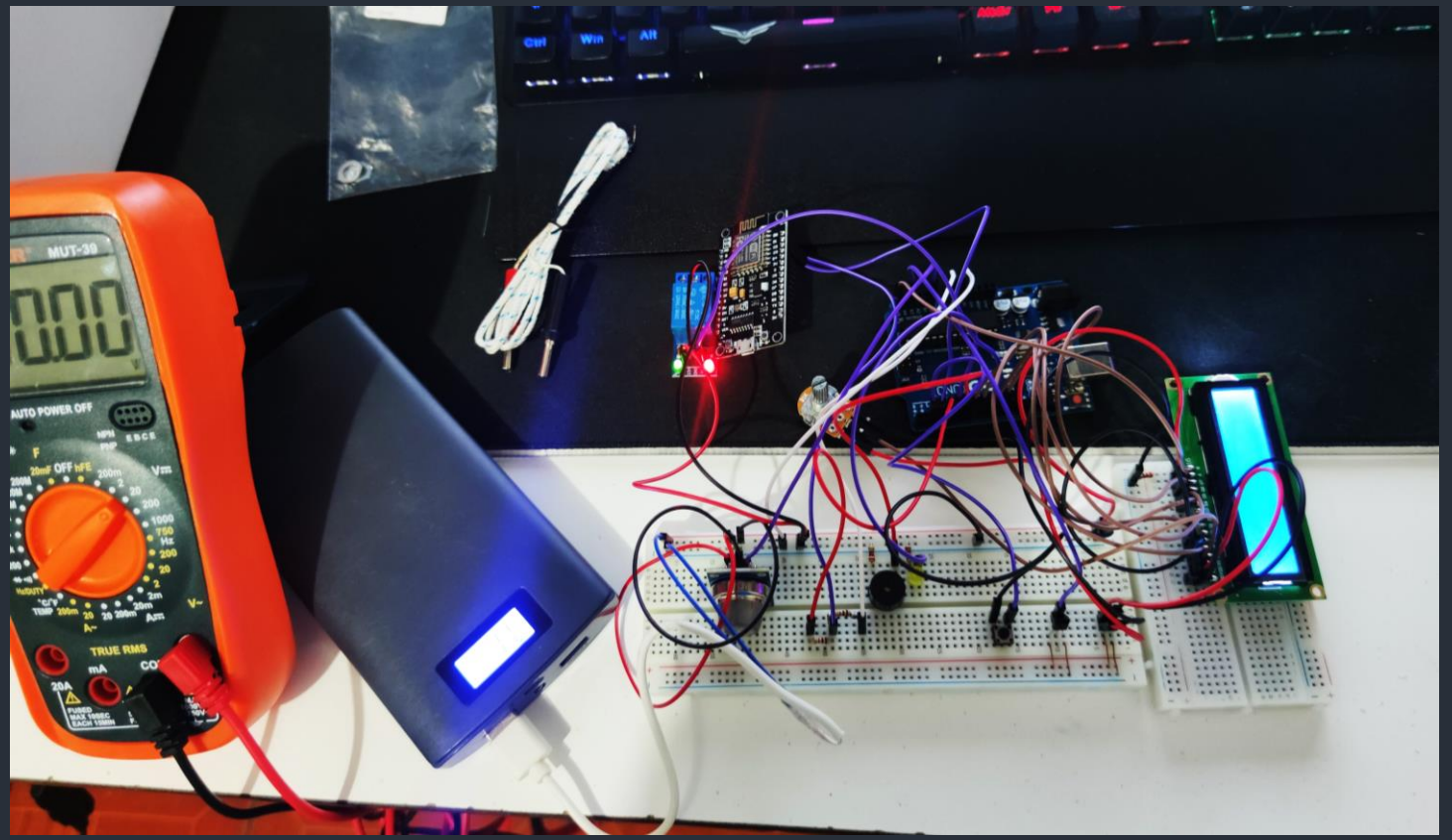
CODIFICACIÓN DE ARDUINO

Se unieron todos los componentes de acuerdo al diagrama mostrado en la diapositiva 7 pero a modo de prueba sin aun montar todo para poder hacer cambios en caso de necesitarlos

```
Codigo_Arduino Arduino 1.8.15 Hourly Build 2021/05/31 10:33
Archivo  Editar  Programa  Herramientas  Ayuda
Codigo_Arduino §

#include <SoftwareSerial.h>
#include <LiquidCrystal.h>


const byte espRx = 3;
const byte espTx = 4;

// alarma
const byte ledAlarm = 5;
const byte buzzerAlarm = 6;
const byte relay = 9;

// lcd
const byte rs = 7;
const byte e = 8;
const byte d4 = 10;
const byte d5 = 11;
const byte d6 = 13;
const byte d7 = 19;  //A5

// sensores
const byte mq = A0;
const byte lm35 = A3;

// bboton
const byte ack = 2; // pin capaz de interrumpir
/****** Pin References ******/

// varibles globales
long co = 0;    //(if saturated > 65535)
long lpg = 0;
long smoke = 0;
float temperature = 0;
bool generalAlarm = false; // si se dispara alguna alarma será cierto
volatile bool ackPressed = false;

// set de alarma
const int lpgSet = 2100;
const int coSet = 200;
const int smokeSet = 1150;
const float temperatureSet = 45.0; // °C


// Intancias
SoftwareSerial SerialEsp(espRx, espTx); // RX, TX
LiquidCrystal lcd(rs, e, d4, d5, d6, d7); // Crea un objeto LCD


// Tiempos
long startingLoopTime = 0;
long currentLoopTime = 0;
const int loopTimeSet = 1000; // 1.5s

long startingSerialTime = 0; // para enviar datos a esp
long currentSerialTime = 0;
const int serialTimeSet = 10000; // 30s


// Caracter personalizado de campana
 byte bell[8] = {
  0b00100,
  0b01110,
  0b01110,
  0b01110,
  0b11111,
  0b11111,
  0b11111,
  0b00000
 }; // eof bell


/************************ MQ & GASES *****************************************/
/************************Macros relacionadas con el hardware*******************************/
// ledAlarm = 4;                        //cuando comience la calibración, el LED pin 13 se iluminará, se apagará cuando
//mq = A0;                                //defina qué canal de entrada analógica va a utilizar
int RL_VALUE = 1;                                //definir la resistencia de carga en la placa, en kilo ohms
```

UNIÓN

Se carga el sketch en el Arduino y se unieron todos los componentes de acuerdo al diagrama mostrado en la diapositiva 7 pero a modo de prueba sin aun montar todo para poder hacer cambios en caso de necesitarlos



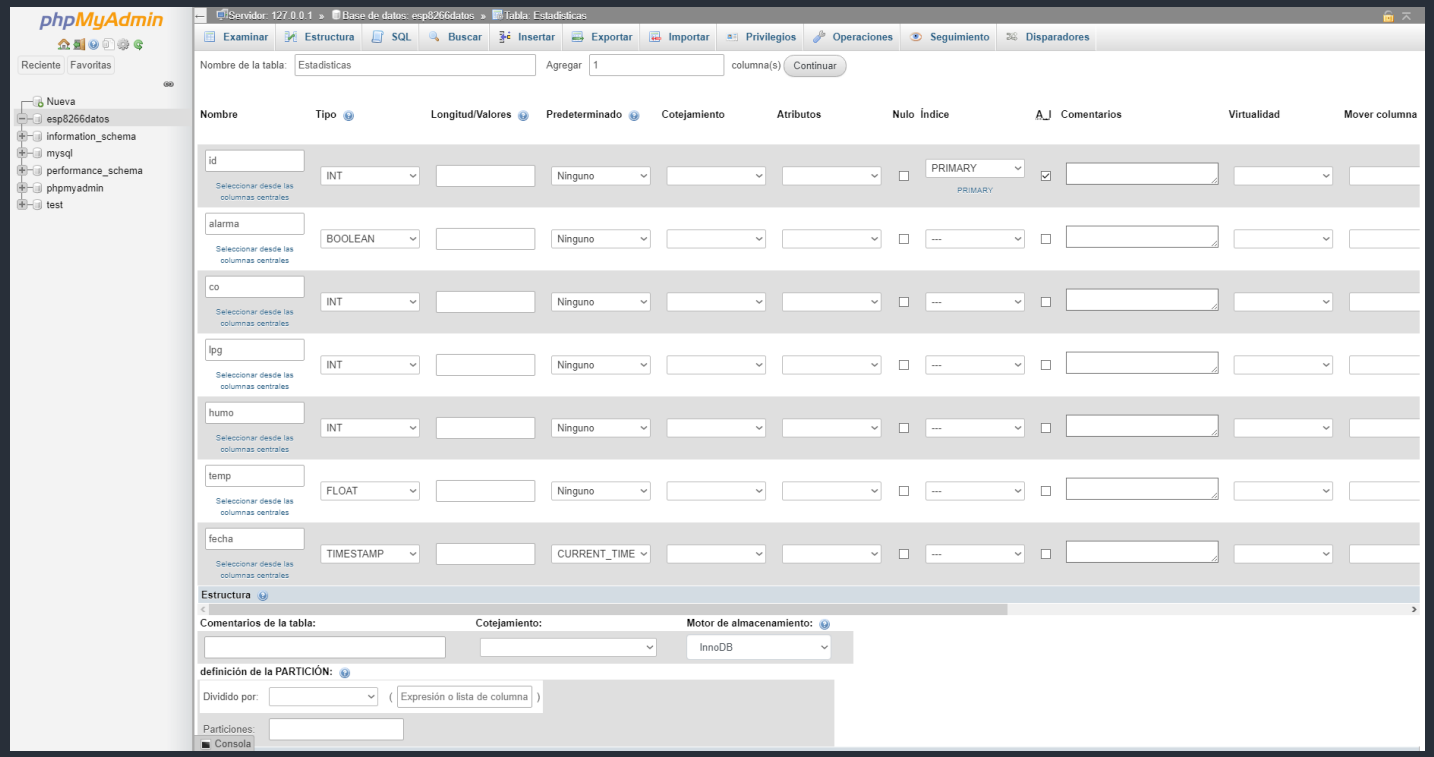
CODIFICACIÓN DE ESP8266

Una ve que comprobamos que el programa de Arduino funciona y que el circuito esta trabajando de forma correcta procedemos a programar el código para el modulo NodeMCU ESP8266 para que este envié los datos al servidor y los guarde en la base de datos

(Mas sin embargo no se sube el programa hasta que tengamos el servidor simulado y la base de datos creada)

CREACIÓN DE LA BASE DE DATOS

Se crea la base de datos en el servidor que hayamos creado o comprado, en este caso la instalación se hará en un localhost mediante XAMPP y el administrador de base de datos que se usara será phpMyAdmin



CONEXIÓN CON LA BASE DE DATOS

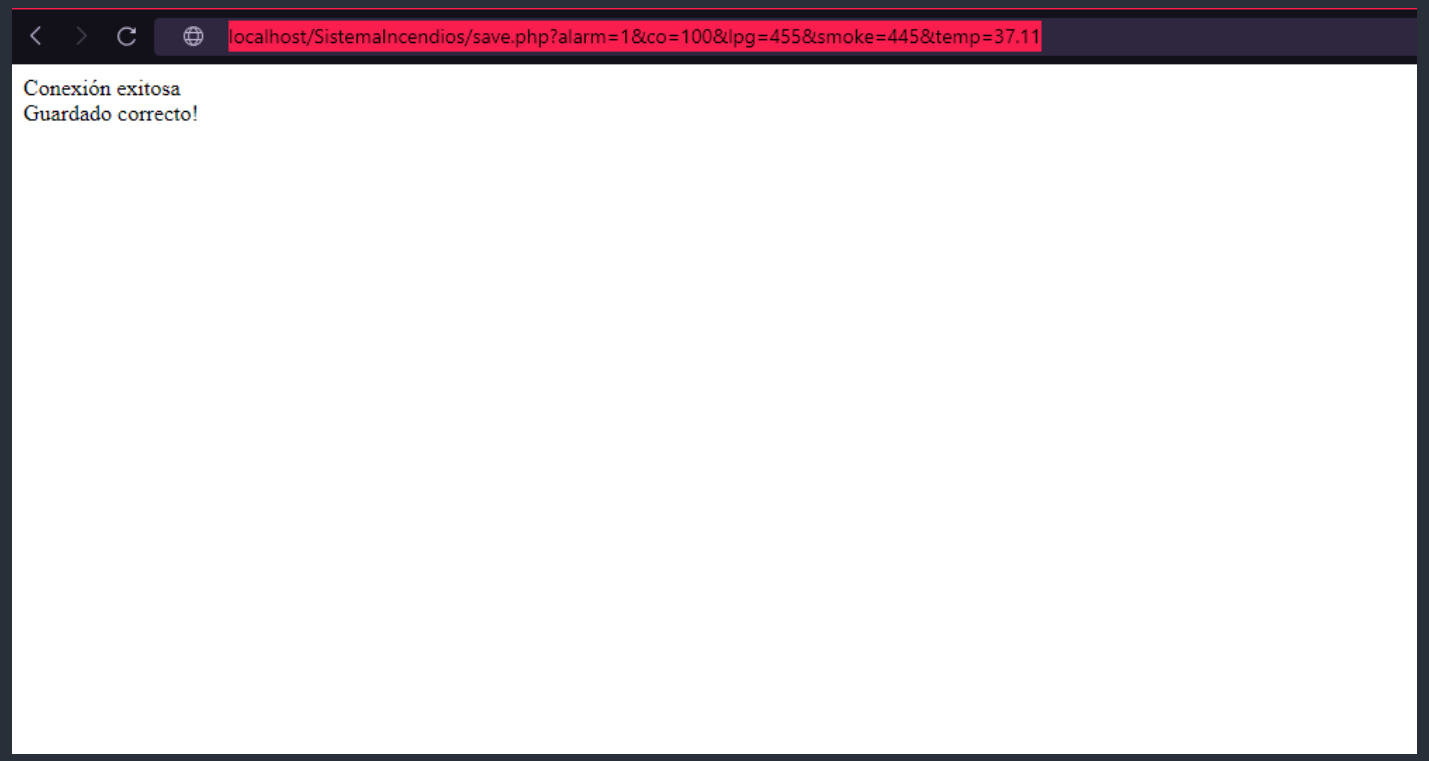
Se crea la conexión con la base de datos para poder utilizarla con el modulo esp8266 y también con la pagina web

PRUEBA DE LA CONEXIÓN CON LA BASE DE DATOS

Entramos a la dirección localhost/esp8266/connection.php y comprobamos que el acceso a la base de datos sea correcto

JQUERY

Usamos la función jQuery y mandamos información como parámetros para que se haga el guardado y comprobamos

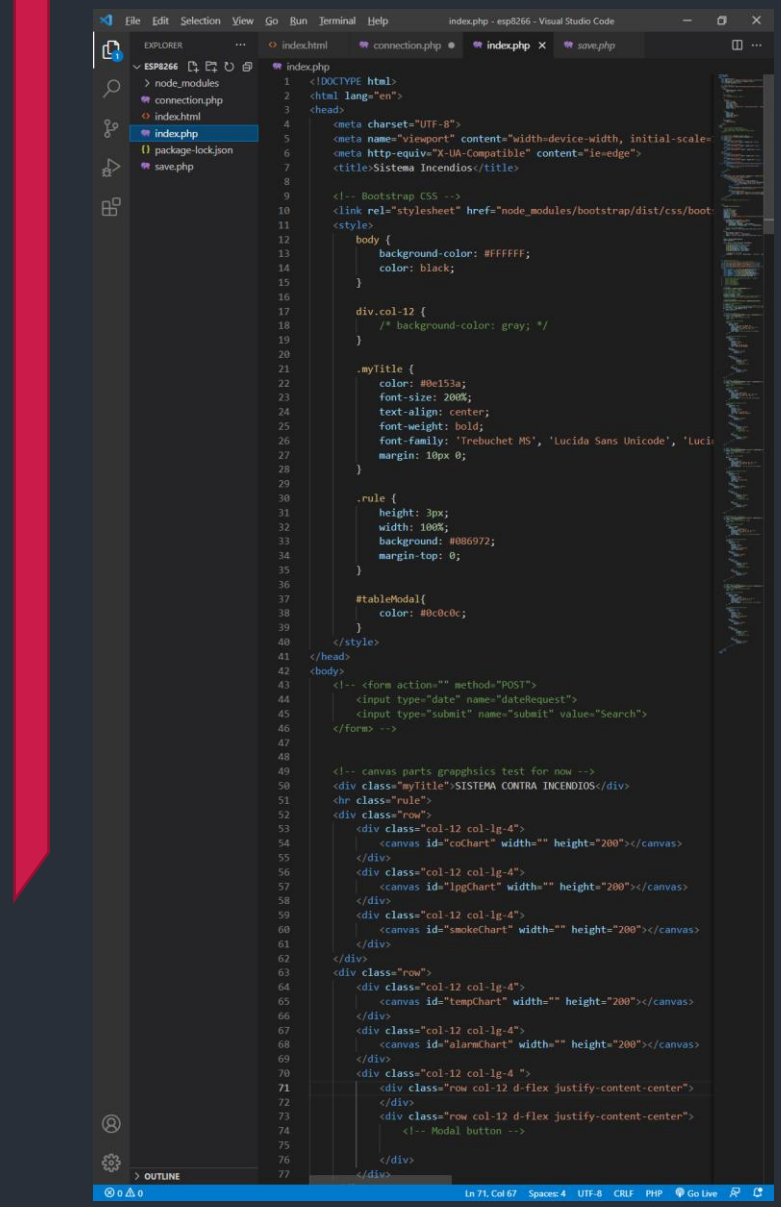


CREACIÓN DE LA PAGINA WEB

Para la parte de la pagina web se dividió en 2 archivos, el primero es un archivo .HTML el cual prestablece la pagina, así como su canvas y algunos otros parámetros y el segundo que es .PHP se dice mas la lógica y el diseño completo de la pagina web, así como la solicitud a la base de datos

CREACIÓN DE LA PAGINA WEB

.PHP



PRUEBA DE LA PAGINA WEB

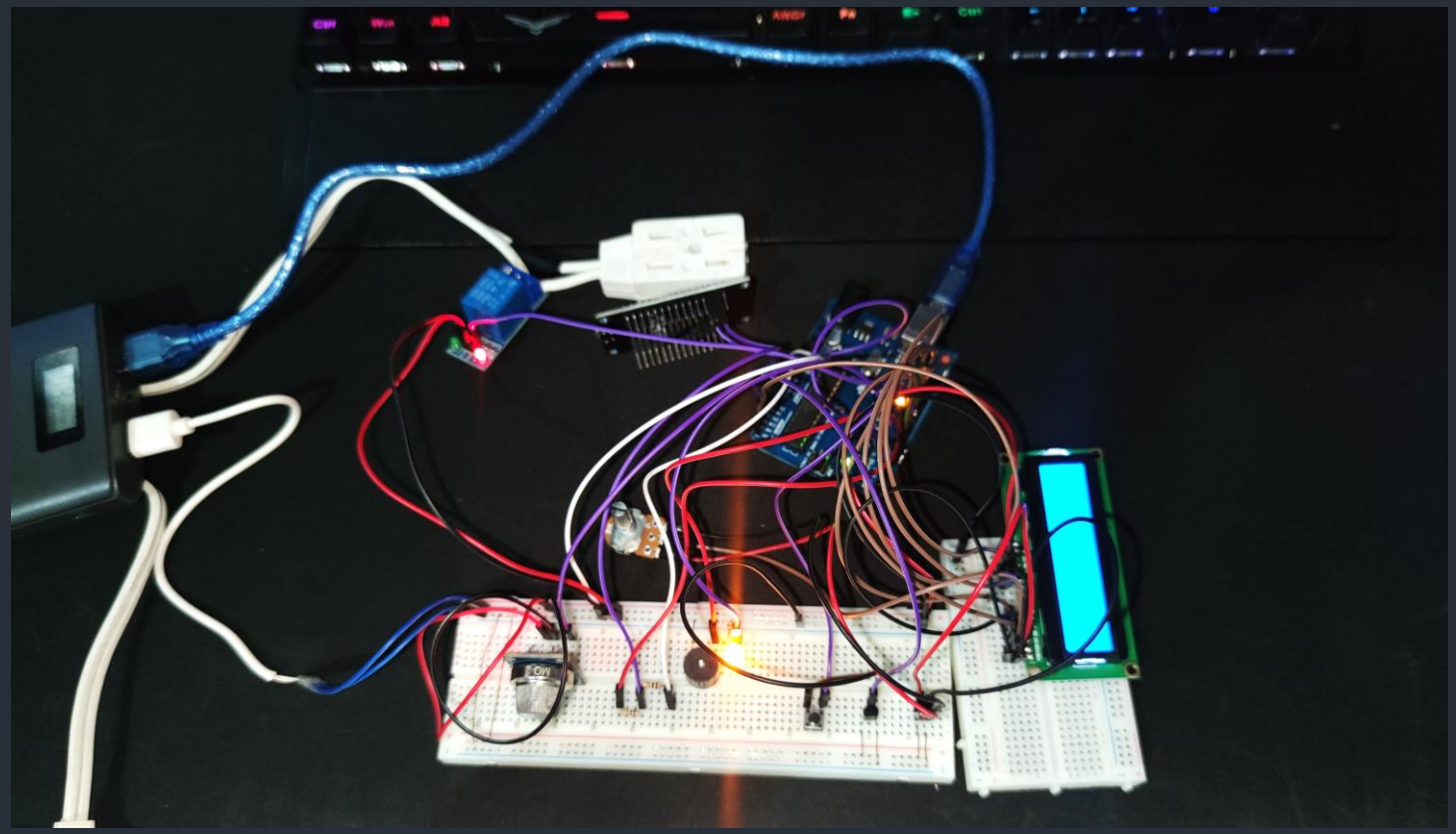
Entramos a la dirección donde hemos alojado la pagina web y comprobamos que este funcionando tanto la pagina web como la obtención de datos de la BD

PAGINA WEB RESPONSIVA

PROCESO DE CONTRUCCIÓN

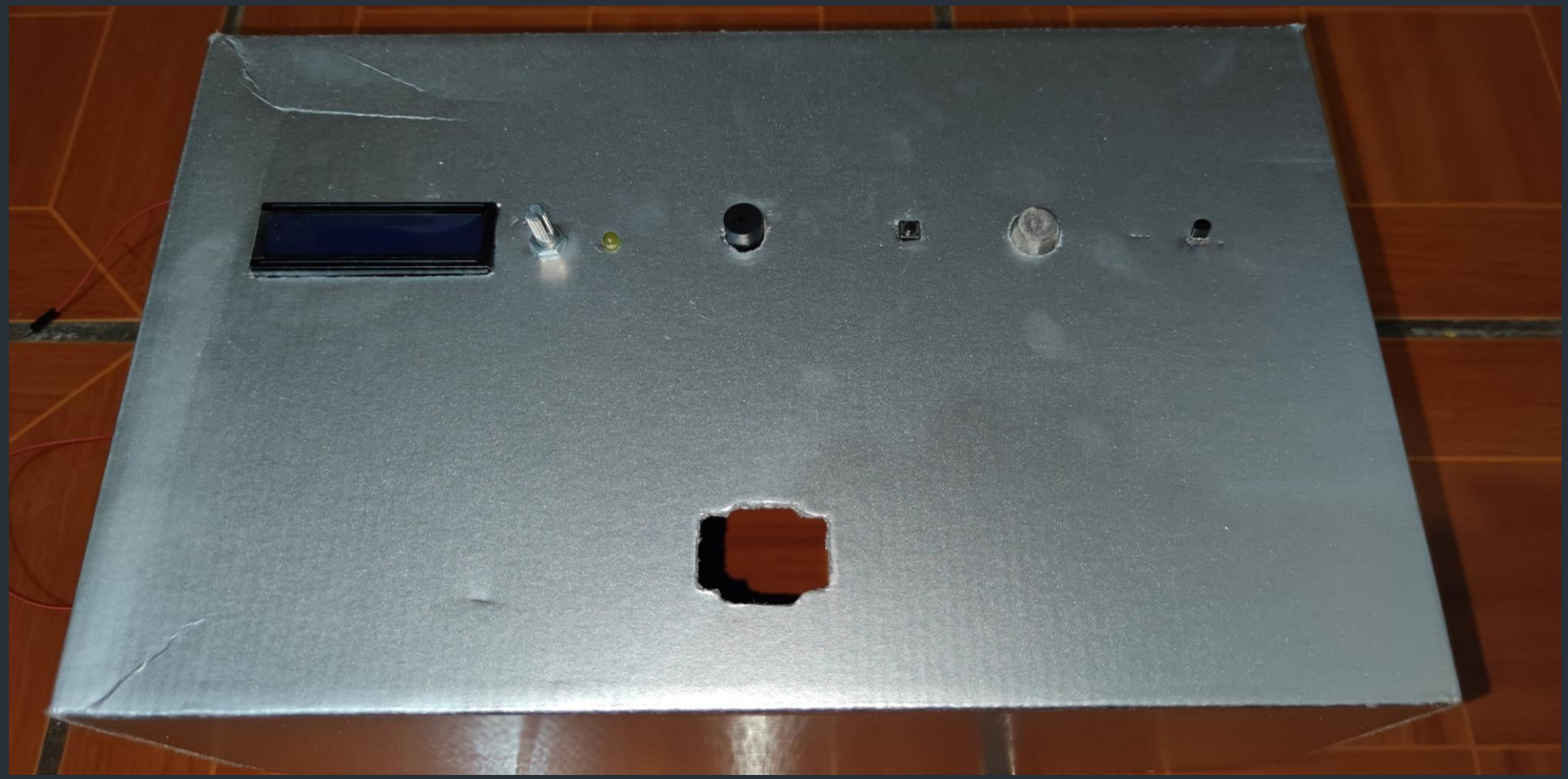
PRUEBAS PREFINALES

Se hace pruebas con todo el setup ya conectado entre si



GABINETE DE MONTAJE

PROCESO DE MONTAJE



PRUEBAS FINALES

MONTAJE FINAL

REQUISITOS

- Aplicar todo lo visto en el taller
- Resolver una problematica
- Debe tener front-end
- Debe tener back-end
- Debe ser responsivo
- Debe tener IOT

LENGUAJES USADOS

EXTENSIONES USADAS

CÓDIGO EN GITHUB

PRESENTADOR Y ALUMNO

Gustavo de Jesus Quintero Cerpa
219294307