Allan Fernando Lemus Monzón

23262

Ingeniería biomédica

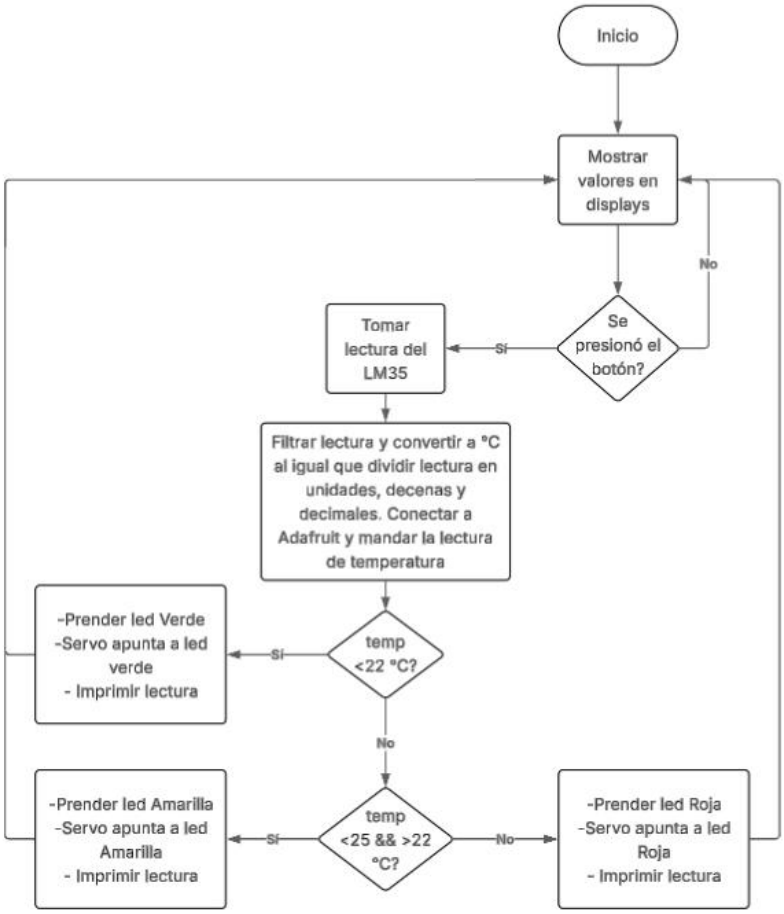
Proyecto 1

Electrónica Digital 2

1 de septiembre del 2025

Universidad Del Valle de Guatemala

Diagrama de Flujo del programa



Circuitos utilizados

Para este proyecto se utilizó un solo circuito el cual cuenta con un ESP32, 3 leds (Verde, amarillo y rojo), 3 transistores NPN 2N3904, ocho resistencias de 220 Ω, 3 displays de cátodo común, un botón, un sensor de temperatura LM35 y un servo motor de 180° de 5 voltios.

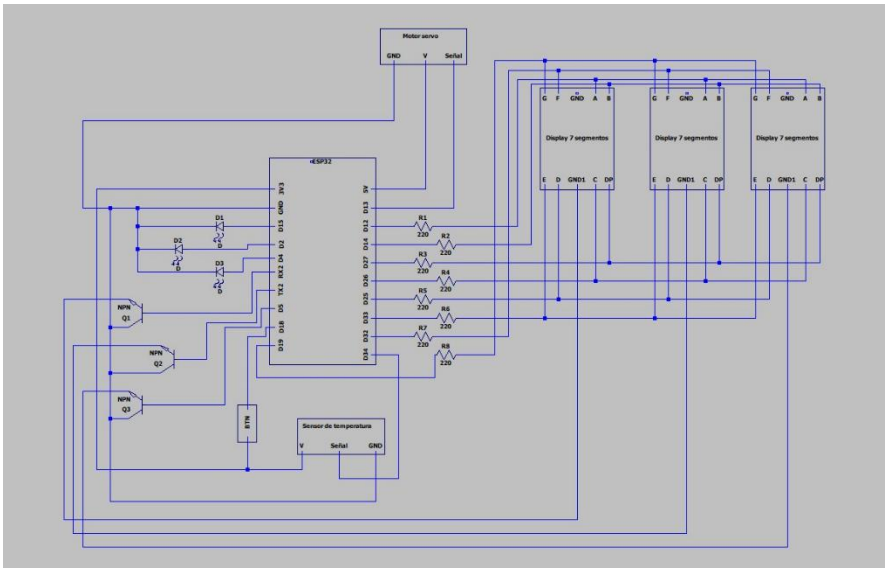
Estos son los componentes utilizados durante el cableado de este proyecto, el ESP32 funciona como el cerebro del proyecto ya este es el encargado de proporcionar el voltaje y la lógica detrás del proyecto. Por otro lado, los leds de colores y el servo motor sirven como indicador de la temperatura que se recibe por parte del sensor de temperatura ya que si se lee una temperatura menor a 22°C se prenderá el led verde, si se lee una temperatura entre 25°C y 22°C, se prenderá el led amarillo y si se detecta una temperatura de 25°C o superior, se prenderá el led rojo mientras que el servo apunta a el led que se encuentra encendida.

Los transistores se utilizan con el fin en mente de realizar un multiplexeo, esto ya que el ESP32 no cuenta con suficientes entradas como para conectar cada una de los leds de los 3 displays en diferentes pines,

por lo que se realiza un interconectado entre sí, de manera que los leds de los displays estén conectados de manera homogénea. Los transistores permiten controlar cual ellos se encuentran activo para evitar que todos los displays muestren el mismo número.

Todo esto da inicio cuando el usuario presiona un botón el cual provoca que se comience a tomar la lectura del sensor de temperatura LM35. A continuación se puede observar un esquemático para ejemplificar el cableado utilizado durante este proyecto.

Imagen 1. Esquemático final del proyecto



-En esta imagen se puede observar el cableado de todos los componentes utilizados en el proyecto 1 de electrónica digital 2.

DATOS EN USO

En este proyecto se utilizan datos binarios para prender los leds, PMW para controlar la dirección del servo motor y la diferencia de voltaje el cual se obtiene por parte del sensor de temperatura.

El PMW se utiliza únicamente para determinar la posición del servo motor ya que este si necesita una posición exacta mientras que los leds no requieren de un brillo especifico durante su funcionamiento y para obtener la medición de temperatura se utilizó un sensor LM35 el cual funciona de manera que genera un voltaje de salida proporcional a la temperatura que siente a su alrededor. Por lo que para poder determinar la temperatura que se detecta se debe de filtrar y adaptar el voltaje recibido por este dependiendo de el voltaje con el que se esté alimentando y como se encuentre calibrado.

GRÁFICOS

Imagen 2. Gráficas de referencia de lectura del sensor de temperatura LM35.

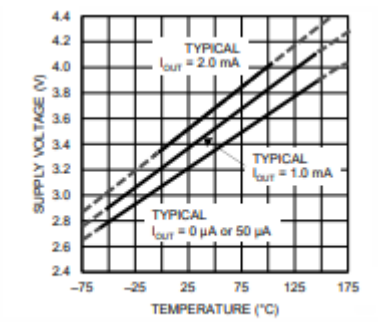


Figure 5. Minimum Supply Voltage vs Temperature

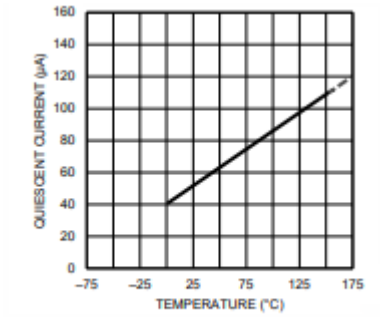


Figure 6. Quiescent Current vs Temperature (in Circuit of Figure 14)

-Estos gráficos sirvieron de referencia para poder calcular y filtrar la señal obtenida por medio del sensor de temperatura LM35

EXPLICACIÓN DEL CÓDIGO

Este proyecto se realizo con el fin en mente de realizar la tarea designada de la manera más comprensible posible para facilitar el proceso de entendimiento del código, por ello, se dividió el código en diferentes secciones, a continuación, se explicará cada una de estas secciones junto con los procesos que ocurren dentro estas.

Para comenzar contamos con la sección de librerías:

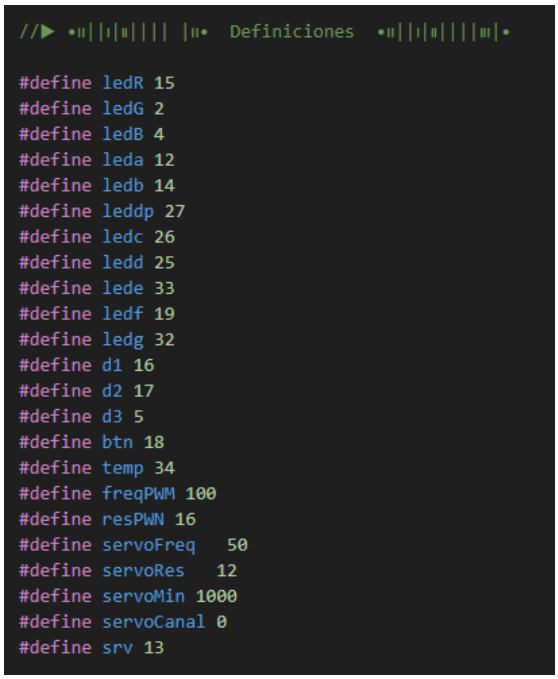
Imagen 3. Librerías utilizadas en el código

```
//▶ •ıı|ı|ıı|||| |ıı•  Librerías  •ıı|ı|ıı||||ıı|•

#include <Arduino.h>
#include "AdafruitIO_WiFi.h"
#include <stdint.h>
#include "driver/ledc.h"
#include "config.h"
```

En esta sección del código se puede observar que se utiliza la librería de Arduino para poder utilizar programación más familiar, librería de “AdafruitIO_Wifi.h” para permitir la conexión con el internet y poder utilizar la plataforma Adafruit.IO. También se utiliza la librería “stint.h” para poder restringir la cantidad de bits que se utilizará en algunas secciones del código. Por otro lado, se utiliza la librería “driver/ledc.h” para poder utilizar las señales PMW del servo motor y por ultimo se utiliza una librería creada para conectarse al servidor de Adafruit el cual se llama “config.h”

Imagen 4. Definiciones utilizadas en el código

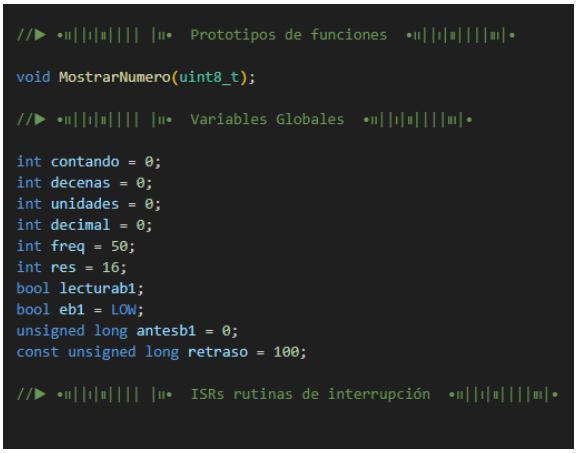


```
//▶ •ıı|||ı|ıı|||| |ıı•  Definiciones  •ıı|||ı|ıı||||ıı|•

#define ledR 15
#define ledG 2
#define ledB 4
#define leda 12
#define ledb 14
#define leddp 27
#define ledc 26
#define ledd 25
#define lede 33
#define ledf 19
#define ledg 32
#define d1 16
#define d2 17
#define d3 5
#define btn 18
#define temp 34
#define freqPWM 100
#define resPWN 16
#define servoFreq   50
#define servoRes   12
#define servoMin 1000
#define servoCanal 0
#define srv 13
```

En esta sección se puede observar cada una de las definiciones que se utilizaron a lo largo del código al igual que se designaron los pines para cada uno de los componentes del circuito.

Imagen 5. Prototipos de funciones, variables globales e ISRs del código.



```
//▶ •ıı|||ı|ıı|||| |ıı•  Prototipos de funciones  •ıı|||ı|ıı||||ıı|•

void MostrarNumero(uint8_t);

//▶ •ıı|||ı|ıı|||| |ıı•  Variables Globales  •ıı|||ı|ıı||||ıı|•

int contando = 0;
int decenas = 0;
int unidades = 0;
int decimal = 0;
int freq = 50;
int res = 16;
bool lecturab1;
bool eb1 = LOW;
unsigned long antesb1 = 0;
const unsigned long retraso = 100;

//▶ •ıı|||ı|ıı|||| |ıı•  ISRs rutinas de interrupción  •ıı|||ı|ıı||||ıı|•
```

En esta imagen se puede observar el prototipo de función utilizado el cual es la función designada para mostrar un número en un display de 7 segmentos. Por otro lado, se puede observar las variables globales que se utilizaron a lo largo del código. Sin embargo, no se observó la necesidad de realizar una ISR en ninguna parte del código por lo que dicha sección se encuentra vacía. Se tomó esta decisión ya que, debido a la escala del proyecto, no se presentó una situación en la cual una ISR fuera exclusivamente necesaria.

Imagen 6. Configuraciones en el código para conectar con Adafruit

```
//▶ •ıı|ı|ıı|ıı|  |ıı•  Adafruit  •ıı|ı|ıı|ıı|ıı|•

#define IO_USERNAME "SndSpn"
#define IO_KEY "aio_riux56ZW4WGOpkbNhCAKGKFQktP4"
#define WIFI_SSID "AllanWifi"
#define WIFI_PASS "07070707"
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
#define IO_LOOP_DELAY 5000
unsigned long lastUpdate = 0;

// set up the 'counter' feed
AdafruitIO_Feed *canaltemp = io.feed("temp");
```

En esta sección del código se puede observar la configuración para la conexión a Adafruit.IO. Se puede observar la conexión de Wifi que se utilizará para realizar dicha conexión al igual que la variable que se quiere mandar y el canal por el cual se transmitirá la información recibida por el ESP32.

Imagen 7. Configuraciones de definiciones del código

```
void setup() {
  pinMode(btn, INPUT);
  pinMode(temp, INPUT);
  pinMode(ledR, OUTPUT);
  pinMode(ledG, OUTPUT);
  pinMode(ledB, OUTPUT);
  pinMode(leda, OUTPUT);
  pinMode(ledb, OUTPUT);
  pinMode(leddp, OUTPUT);
  pinMode(ledc, OUTPUT);
  pinMode(ledd, OUTPUT);
  pinMode(lede, OUTPUT);
  pinMode(ledf, OUTPUT);
  pinMode(ledg, OUTPUT);
  pinMode(d1, OUTPUT);
  pinMode(d2, OUTPUT);
  pinMode(d3, OUTPUT);
  digitalWrite(d1, LOW);
  digitalWrite(d2, LOW);
  digitalWrite(d3, LOW);
  digitalWrite(leda, LOW);
  digitalWrite(ledb, LOW);
  digitalWrite(leddp, LOW);
  digitalWrite(ledc, LOW);
  digitalWrite(ledd, LOW);
  digitalWrite(lede, LOW);
  digitalWrite(ledf, LOW);
  digitalWrite(ledg, LOW);
  ledcSetup(servoCanal, servoFreq, servoRes);
  ledcAttachPin(srv, servoCanal);
  ledcWrite(servoCanal, 0);
  Serial.begin(115200);
  Serial.print("Conectando a Adafruit....");
  io.connect();
  while (io.status() < AIO_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("\nConectado a Adafruit ");
```

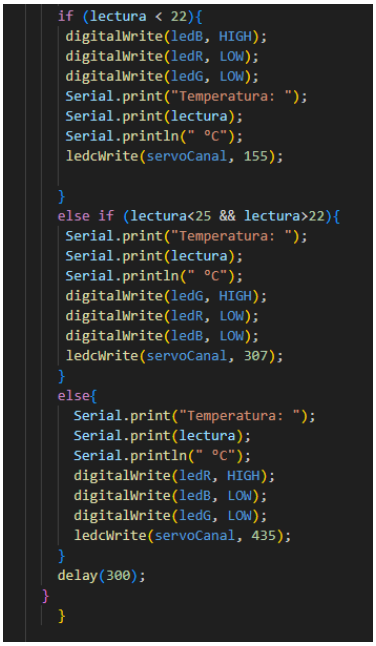
En la imagen 7 se puede observar las configuraciones de cada una de las variables que se mostraron previamente, en estas se define cuales son entradas y salidas del código al igual que como se mostrarán cada una de estas variables al momento de comenzar el código. Por otro lado, se observa la configuración de la señal PMW del servo motor en la cual se define la frecuencia, resolución y canal que se utilizará para este. Por último, se define la comunicación entre el ESP32 y la computadora y la conexión con Adafruit.IO.

Imagen 8. Loop infinito del código

```
//▶ •ıı|ı|ı|ı|||  |ıı•  Loop infinito  •ıı|ı|ı|ı|||ıı|•

void loop() {
    digitalWrite(d1, HIGH);
    digitalWrite(d2, LOW);
    digitalWrite(d3, LOW);
    MostrarNumero(decenas);
    digitalWrite(leddp, LOW);
    delay(5);
    digitalWrite(d1, LOW);
    digitalWrite(d2, HIGH);
    digitalWrite(d3, LOW);
    MostrarNumero(unidades);
    digitalWrite(leddp, HIGH);
    delay(5);
    digitalWrite(d1, LOW);
    digitalWrite(d2, LOW);
    digitalWrite(d3, HIGH);
    MostrarNumero(decimal);
    digitalWrite(leddp, LOW);
    delay(5);
    digitalWrite(d1, LOW);
    digitalWrite(d2, LOW);
    digitalWrite(d3, LOW);
    delay(5);
  if (digitalRead(btn) == HIGH){
    io.run();
    int sensorValue = analogRead(temp);
    float voltage = (sensorValue / 4095.0) * 3.3;
    float lectura = (voltage *100.0)+9;
    decenas = lectura/10;
    unidades = lectura-(decenas*10);
    decimal = (lectura*10)-(decenas*100)-(unidades*10);
    canaltemp->save(lectura);
```

Imagen 9. Segunda parte del Loop infinito del código

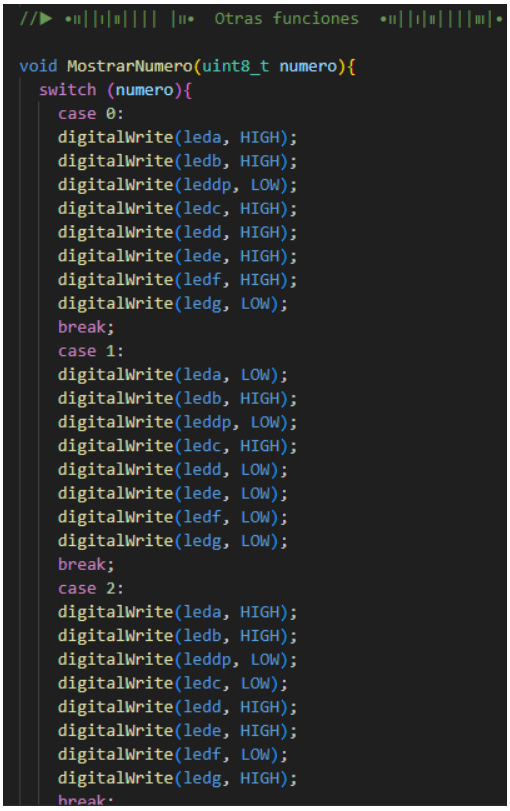


```
    if (lectura < 22){
     digitalWrite(ledB, HIGH);
     digitalWrite(ledR, LOW);
     digitalWrite(ledG, LOW);
     Serial.print("Temperatura: ");
     Serial.print(lectura);
     Serial.println(" °C");
     ledcWrite(servoCanal, 155);

    }
    else if (lectura<25 && lectura>22){
     Serial.print("Temperatura: ");
     Serial.print(lectura);
     Serial.println(" °C");
     digitalWrite(ledG, HIGH);
     digitalWrite(ledR, LOW);
     digitalWrite(ledB, LOW);
     ledcWrite(servoCanal, 307);
    }
    else{
      Serial.print("Temperatura: ");
      Serial.print(lectura);
      Serial.println(" °C");
      digitalWrite(ledR, HIGH);
      digitalWrite(ledB, LOW);
      digitalWrite(ledG, LOW);
      ledcWrite(servoCanal, 435);
    }
    delay(300);
  }
    }
```

El loop infinito comienza mostrando los valores de las variables “Decenas”, “Unidades” y “Decimal” los cuales se designaron como “0” para comenzar, luego de presionar el botón para comenzar la lectura, este se conecta a Adafruit, luego, el ESP32 recibe la lectura del sensor LM35 y la separa de tal manera que se separan los valores de dicha lectura en 3 variables, el primer valor se guarda en una variable llamada “Decena”, la segunda a una variable llamada “Unidades” y el último número en una variable llamada “Decimal” e imprime y manda la lectura al servidor de Adafruit. Luego de terminar esta sección, comienza el loop infinito en el cual se muestra el valor de las variables mencionadas anteriormente de tal manera que cada display muestra uno de estos valores hasta volver a recibir la señal del botón para repetir el proceso nuevamente.

Imagen 10. Función designada para mostrar números en el display.



```
//▶ •ıı|ı|ıı|||| |ıı•  Otras funciones  •ıı|ı|ıı|||ıı|•

void MostrarNumero(uint8_t numero){
  switch (numero){
    case 0:
    digitalWrite(leda, HIGH);
    digitalWrite(ledb, HIGH);
    digitalWrite(leddp, LOW);
    digitalWrite(ledc, HIGH);
    digitalWrite(ledd, HIGH);
    digitalWrite(lede, HIGH);
    digitalWrite(ledf, HIGH);
    digitalWrite(ledg, LOW);
    break;
    case 1:
    digitalWrite(leda, LOW);
    digitalWrite(ledb, HIGH);
    digitalWrite(leddp, LOW);
    digitalWrite(ledc, HIGH);
    digitalWrite(ledd, LOW);
    digitalWrite(lede, LOW);
    digitalWrite(ledf, LOW);
    digitalWrite(ledg, LOW);
    break;
    case 2:
    digitalWrite(leda, HIGH);
    digitalWrite(ledb, HIGH);
    digitalWrite(leddp, LOW);
    digitalWrite(ledc, LOW);
    digitalWrite(ledd, HIGH);
    digitalWrite(lede, HIGH);
    digitalWrite(ledf, LOW);
    digitalWrite(ledg, HIGH);
    break;
```

En la imagen 10 se puede observar una sección del código designada en prender los leds correctos para mostrar los números de 0 a 9, el cual se utiliza durante el loop infinito ya que al momento de colocar los valores filtrados dentro de esta función, los displays mostraran los valores filtrados de la lectura del LM35.

Link a GitHub

https://github.com/SoundSpoon/Proyecto-1-Digital-2-Allan-Lemus

Link a video

https://youtu.be/VVav_WGXRVw