Microprocessors

Final project (Hardware)

display alphanumeric characters on Seven Segment Display

Name	ID
Mohamed Hosny Qasem	21011117
Ziad Ismail Hassan	21010559
Adham Abu Bakr Morsy Mohamed	21010219
Nardeen Refaat Fuad	20012076
Asmaa Hassan Mokhtar Aboushady	19015430

```
#define REMOTEXY_MODE__ESP32CORE_WIFI_POINT
#include <WiFi.h>

// RemoteXY connection settings
#define REMOTEXY_WIFI_SSID "team 35"
#define REMOTEXY_WIFI_PASSWORD "qasemunisco"
#define REMOTEXY_SERVER_PORT 6377


#include <RemoteXY.h>
//////////////////////////////////////////////////////////////////////////////////////////////
// RemoteXY GUI configuration
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =   // 82 bytes
  { 255,3,0,0,0,75,0,17,0,0,0,16,1,106,200,1,1,4,0,10,
  25,107,54,54,49,203,26,31,79,78,0,31,79,70,70,0,129,7,54,50,
  6,203,69,78,84,69,82,32,84,72,69,32,73,78,80,85,84,0,129,38,
  101,27,6,203,76,79,79,80,32,65,76,76,0,7,7,62,92,15,4,16,
  203,2 };

// this structure defines all the variables and events of your control interface
struct {

    // input variables
  uint8_t INLoop; // =1 if state is ON, else =0

    // output variables
  char INText[2]; // string UTF8 end zero

} RemoteXY;
#pragma pack(pop)
//////////////////////////////////////////////////////////////////////////////////////////////
const int A = 12;
const int B = 14;
const int C = 2;
const int D = 5;
const int E = 4;
const int F = 13;
const int G = 15;
const int DP = 35;

void setup() {
  Serial.begin(9600);
    RemoteXY_Init ();  // Initialize Remotexy

  pinMode(A, OUTPUT);
```

```
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(E, OUTPUT);
  pinMode(F, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(DP, OUTPUT);
  allOff(); // Turn off all segments Initially
}



// Turn off all segments
void allOff() {
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(DP, HIGH);
}


void displayCharacter(int number) {
  switch(number) {
    case 0:
      zero();
      break;
    case 1:
      one();
      break;
    case 2:
      two();
      break;
    case 3:
      three();
      break;
    case 4:
      four();
      break;
    case 5:
      five();
      break;
    case 6:
      six();
```

```
      break;
    case 7:
      seven();
      break;
    case 8:
      eight();
      break;
    case 9:
      nine();
      break;
    case 10:
      a();
      break;
    case 11:
      b();
      break;
    case 12:
      c();
      break;
    case 13:
      d();
      break;
    case 14:
      e();
      break;
    case 15:
      f();
      break;
    default:
      allOff();  // If the input is out of range
      break;
  }
}


void loop() {
  RemoteXY_Handler();

  if (RemoteXY.INLoop) {
    for (int i = 0; i <= 15; i++) {
      if (!RemoteXY.INLoop) break; // Exit loop if switch is turned off
      displayCharacter(i);
      delay(1000);
    }
  }
  else {
```

```
    interpretText(RemoteXY.INText); //input text to display characters
  }
}

void interpretText(const char* text) {
  if (strcmp(text, "0") == 0) displayCharacter(0);
  else if (strcmp(text, "1") == 0) displayCharacter(1);
  else if (strcmp(text, "2") == 0) displayCharacter(2);
  else if (strcmp(text, "3") == 0) displayCharacter(3);

  else if (strcmp(text, "4") == 0) displayCharacter(4);
  else if (strcmp(text, "5") == 0) displayCharacter(5);
  else if (strcmp(text, "6") == 0) displayCharacter(6);
  else if (strcmp(text, "7") == 0) displayCharacter(7);

  else if (strcmp(text, "8") == 0) displayCharacter(8);
  else if (strcmp(text, "9") == 0) displayCharacter(9);
  else if (strcmp(text, "A") == 0) displayCharacter(10);
  else if (strcmp(text, "B") == 0) displayCharacter(11);

  else if (strcmp(text, "C") == 0) displayCharacter(12);
  else if (strcmp(text, "D") == 0) displayCharacter(13);
  else if (strcmp(text, "E") == 0) displayCharacter(14);
  else if (strcmp(text, "F") == 0) displayCharacter(15);

  else allOff();
}


void zero(){
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C,LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F,LOW);
  digitalWrite(G, HIGH);
  digitalWrite(DP, LOW);
}

void one(){
  digitalWrite(A, HIGH);
  digitalWrite(B,LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
```

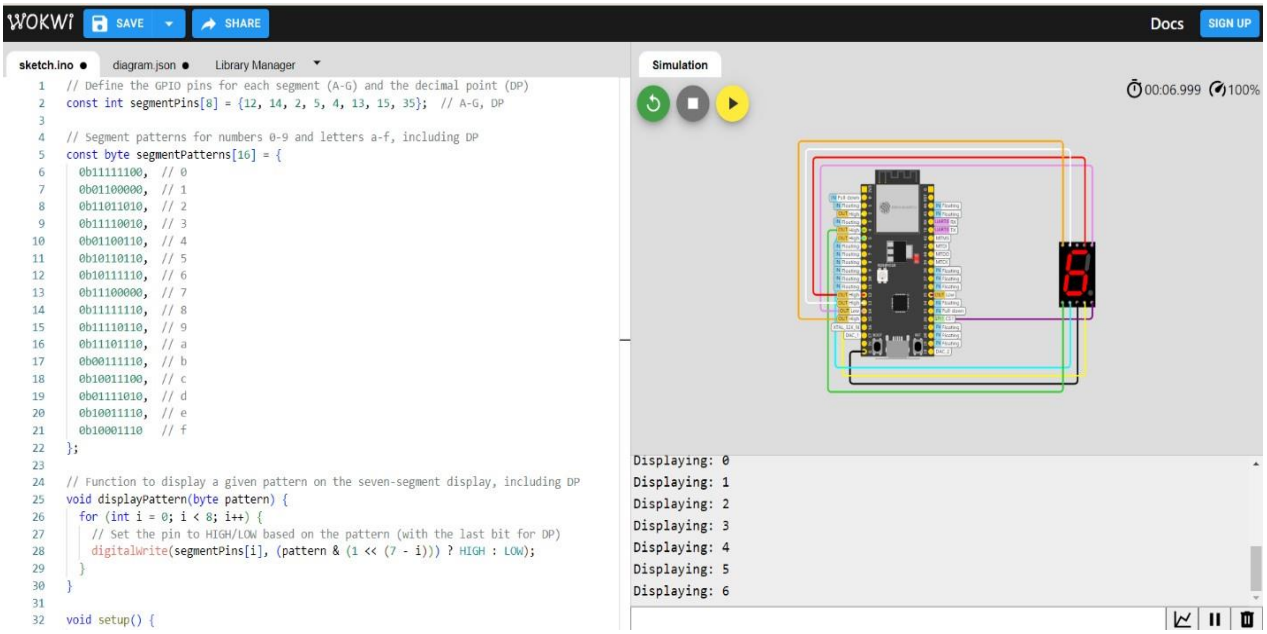
```
  digitalWrite(G,HIGH);
  digitalWrite(DP, HIGH);
}

void two(){
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E,LOW);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(DP,  HIGH);
}

void three(){
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(DP, HIGH);
}

void four(){
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(DP,  HIGH);
}

void five(){
  digitalWrite(A, LOW);
  digitalWrite(B,HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D,LOW);
  digitalWrite(E, HIGH);
  digitalWrite(F,LOW);
  digitalWrite(G, LOW);
```

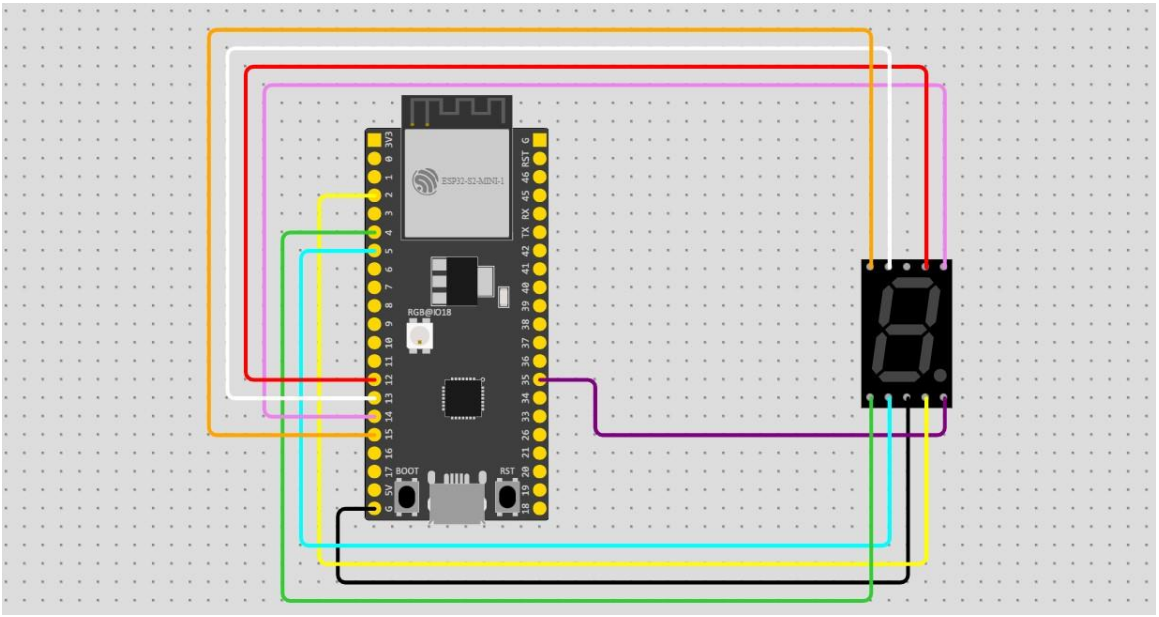
```
  digitalWrite(DP, HIGH);
}

void six(){
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(DP,  HIGH);
}
void seven(){
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, HIGH);
  digitalWrite(F, HIGH);
  digitalWrite(G, HIGH);
  digitalWrite(DP,  HIGH);
}

void eight(){
  digitalWrite(A, LOW);
  digitalWrite(B,LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(DP,  HIGH);
}

void nine(){
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D,LOW);
  digitalWrite(E, HIGH);
  digitalWrite(F, LOW);
  digitalWrite(G,LOW);
  digitalWrite(DP,  HIGH);
}
```

```
void a(){
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(DP,  HIGH);
}

void b(){
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(DP, HIGH);
}

void c(){
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C,HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E,LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, HIGH);
  digitalWrite(DP, HIGH);
}
void d(){
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
  digitalWrite(E,LOW);
  digitalWrite(F, HIGH);
  digitalWrite(G, LOW);
  digitalWrite(DP,  HIGH);
}

void e(){
  digitalWrite(A, LOW);
```

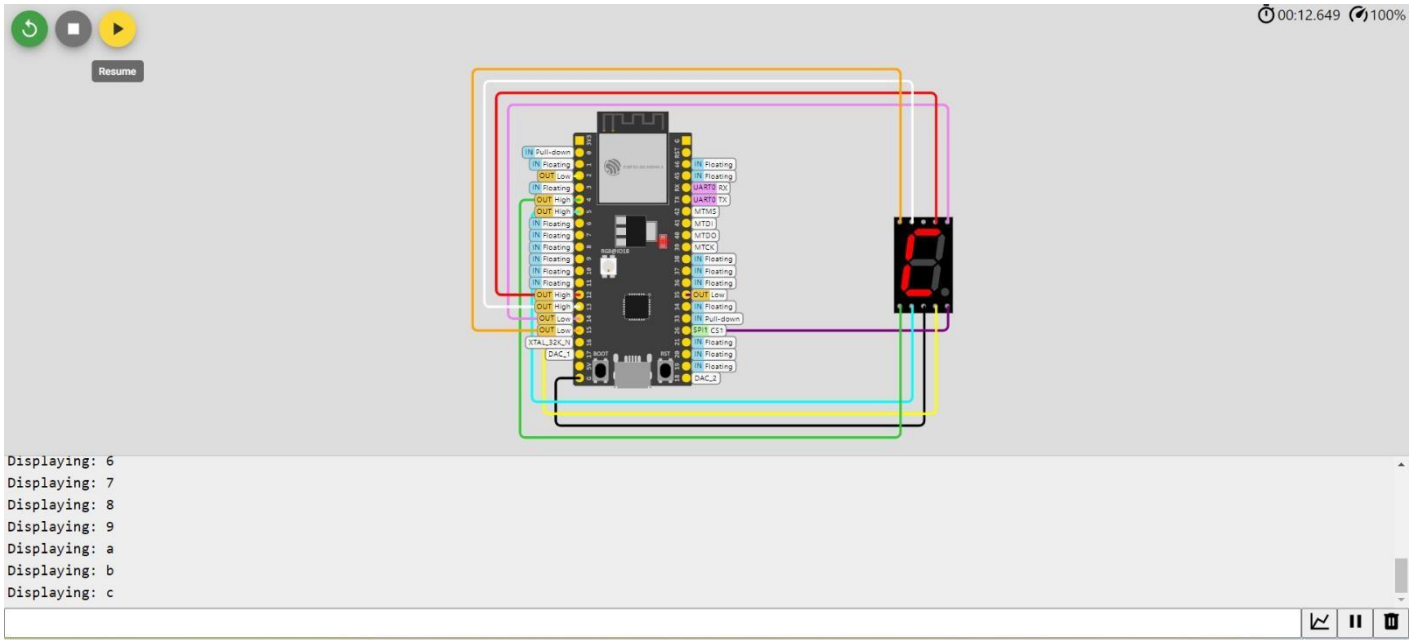
```
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(DP, HIGH);
}

void f(){
  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, HIGH);
  digitalWrite(E, LOW);
  digitalWrite(F, LOW);
  digitalWrite(G, LOW);
  digitalWrite(DP, HIGH);
}
```

Simulation:

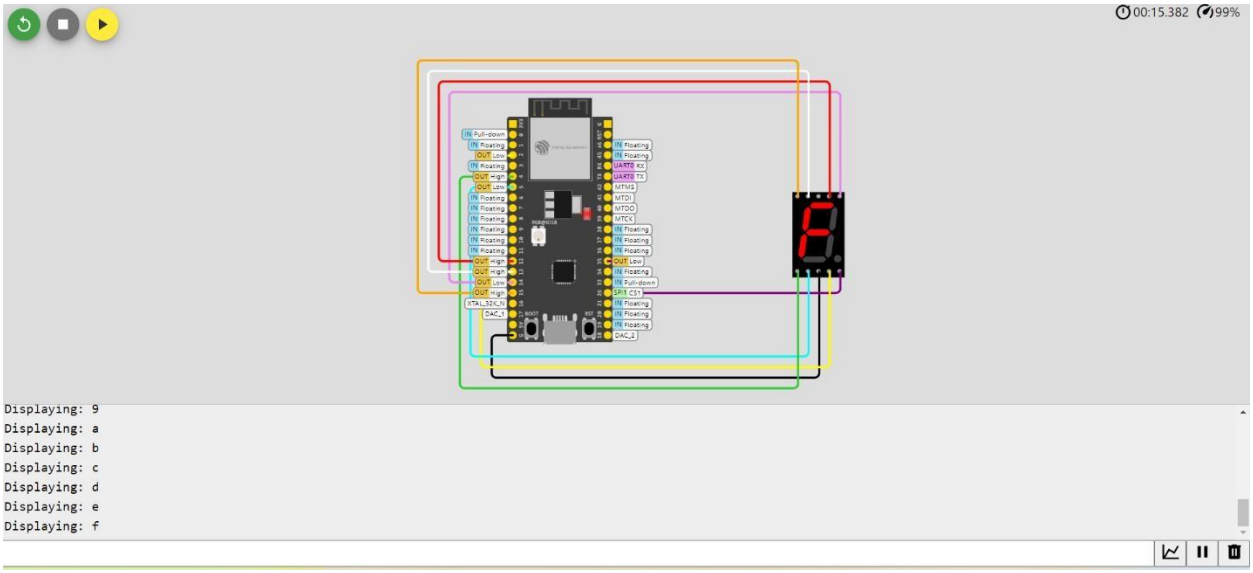

WOKWI
SAVE
SHARE
Docs
SIGN UP
sketch.ino
diagram.json
Library Manager
Simulation
00:06.999
100%
// Define the GPIO pins for each segment (A-G) and the decimal point (DP)
const int segmentPins[8] = {12, 14, 2, 5, 4, 13, 15, 35}; // A-G, DP
// Segment patterns for numbers 0-9 and letters a-f, including DP
const byte segmentPatterns[16] = {
0b11111100, // 0
0b01100000, // 1
0b11011010, // 2
0b11110010, // 3
0b01100110, // 4
0b10110110, // 5
0b10111110, // 6
0b11100000, // 7
0b11111110, // 8
0b11110110, // 9
0b11101110, // a
0b00111110, // b
0b10011100, // c
0b01111010, // d
0b10011110, // e
0b10001110   // f
};
// Function to display a given pattern on the seven-segment display, including DP
void displayPattern(byte pattern) {
for (int i = 0; i < 8; i++) {
// Set the pin to HIGH/LOW based on the pattern (with the last bit for DP)
digitalWrite(segmentPins[i], (pattern & (1 << (7 - i))) ? HIGH : LOW);
}
}
void setup() {
Displaying: 0
Displaying: 1
Displaying: 2
Displaying: 3
Displaying: 4
Displaying: 5
Displaying: 6



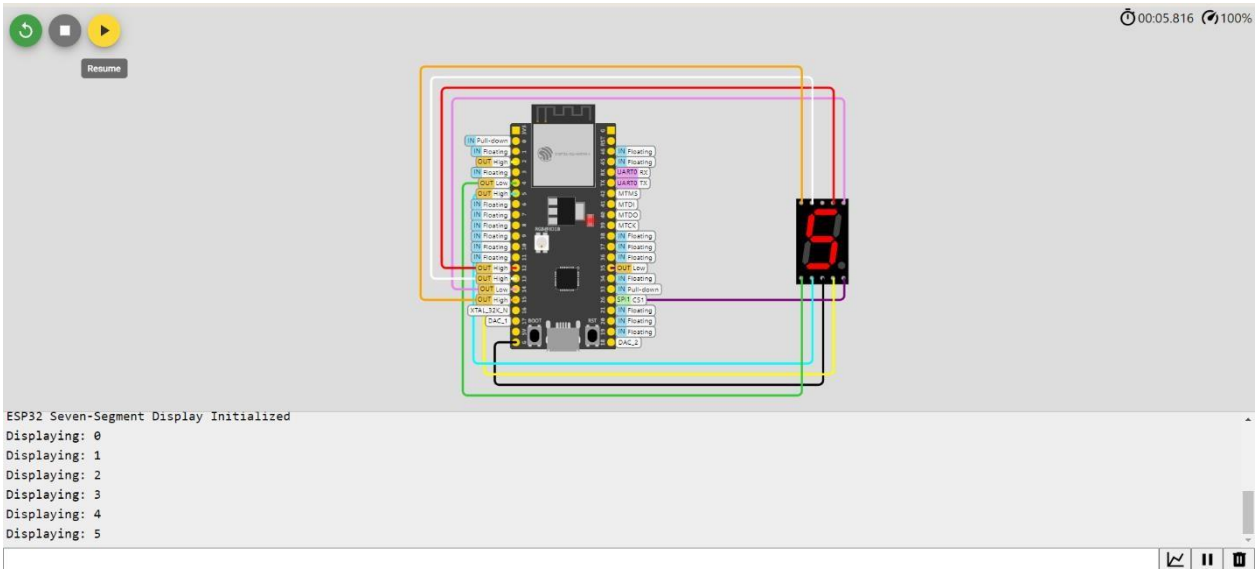
Displaying “C”



Displaying “F”



Displaying “5”



The explanation of the code:

1. Pin Definitions

- **segmentPins**: An array containing the GPIO (General Purpose Input/Output) pins on the ESP32 that connect to the 7-segment display. This array has eight elements, representing the seven segments (A-G) and the decimal point (DP).

2. Segment Patterns

- **segmentPatterns**: An array of 16 values (in binary) representing patterns for numbers 0-9 and letters a-f on the 7-segment display. Each bit in the byte represents whether a segment is on (**1**) or off (**0**).
- The first seven bits represent segments A-G, and the eighth bit represents the decimal point.
- For example, **0b11111100** turns on segments to display the number **0**, with the decimal point off.

3. Function to Display Patterns

- **displayPattern**: A function that takes a **pattern** (a byte) and sets each segment pin (including DP) to either **HIGH** or **LOW**.
- **digitalWrite**: A function that sets a GPIO pin to **HIGH** (3.3V) or **LOW** (0V).
- **(pattern & (1 << (7 - i)))**: This checks whether the i-th bit from the left in the pattern is **1** (turn on the segment) or **0** (turn off the segment).
- If **1**, the corresponding pin is set to **HIGH**; if **0**, it's set to **LOW**.

4. Initialization (Setup)

- **setup**: A function that runs once when the ESP32 starts.
- **pinMode**: Configures each pin as an **OUTPUT**, so it can control the 7-segment display.
- **Serial.begin**: Initializes serial communication at a baud rate of 9600. This allows you to send messages to the Serial Monitor for debugging or output.
- **Serial.println**: Sends a message to the Serial Monitor, which can be useful for monitoring.

5. Main Loop

- **loop**: The function that runs repeatedly on the ESP32.
- The first **for** loop iterates through numbers 0-9, displaying each on the 7-segment display and sending a message to the Serial Monitor.
- The second **for** loop iterates through letters a-f, similarly displaying each and sending a message.
- **delay(1000)**: Pauses execution for 1 second, allowing the display to remain visible for some time.

Summary

- This code uses the ESP32 to control a 7-segment display. It initializes the display's segment pins, defines patterns for numbers and letters, and then displays them with a one-second delay.
- The Serial Monitor provides debugging information, showing which number or letter is currently being displayed.