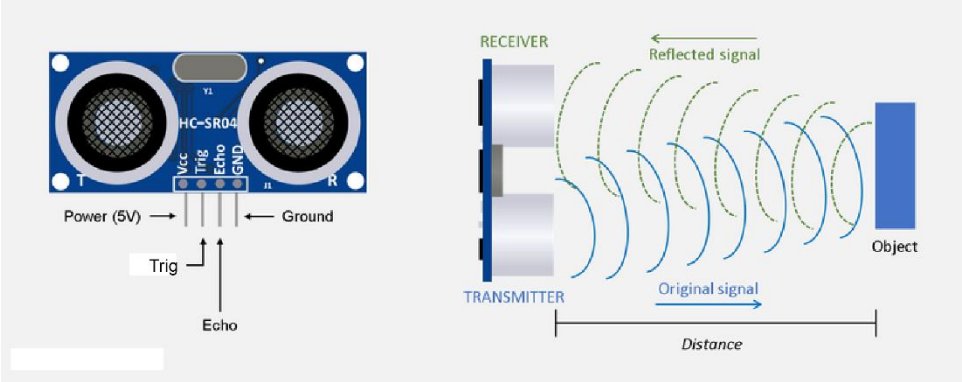
Table of Contents

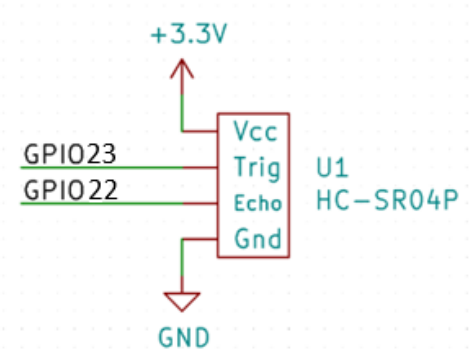
Contents	Page No.
Part A: HC-SR04P Ultrasonic Distance Sensor	**P2**
Part B: OV7670 Camera	**P3**
Part C: FSM	**P7**
Part D: PID	**P9**
Part E: ESP32 Dual Core	**P12**

Part A: HC-SR04P Ultrasonic Distance Sensor

1. In this scenario, we are going to study about Finite State Machine (FSM).
2. Hardware Required
    a. ESP32
    b. HC-SR04P
    c. Jumper Wires
    d. Breadboard
    e. Micro USB cable
3. Circuit Schematic
    a. Trig -> GPIO23
    b. Echo -> GPIO22
    c. VCC -> Vin
    d. GND -> GND

    Note that HC-SR04P is compatible with both 3.3V and 5V power supply and logic levels, while HC-SR04 is compatible with only 5V power supply logic level.





4. Source Code

```
#define TRIG_PIN 23 // ESP32 pin GIOP23 connected to Ultrasonic Sensor's TRIG pin
#define ECHO_PIN 22 // ESP32 pin GIOP22 connected to Ultrasonic Sensor's ECHO pin

float duration_us, distance_cm;

void setup() {
  // begin serial port
```

```
  Serial.begin (9600);

  // configure the trigger pin to output mode
  pinMode(TRIG_PIN, OUTPUT);
  // configure the echo pin to input mode
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  // generate 10-microsecond pulse to TRIG pin
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // measure duration of pulse from ECHO pin
  duration_us = pulseIn(ECHO_PIN, HIGH);

  // calculate the distance
  distance_cm = 0.017 * duration_us;

  // print the value to Serial Monitor
  Serial.print("distance: ");
  Serial.print(distance_cm);
  Serial.println(" cm");

  delay(500);
}
```

5. Sample Result

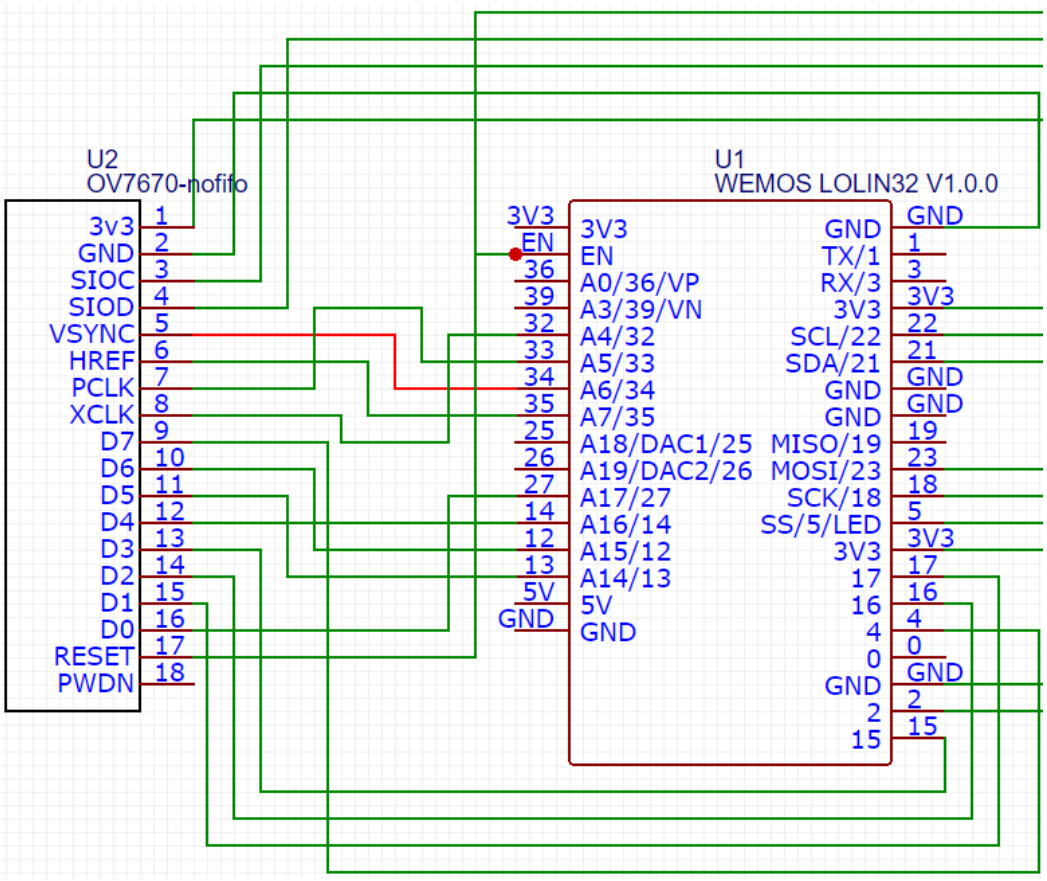
Your Testing: Take a video record where you should show and explain when you are varying the distance and it effect the values in serial monitor.

Part B: OV7670 Camera

1. In this scenario, we are going to study about OV7670 Camera.
2. Hardware Required
    a. ESP32
    b. OV7670 Camera
    c. Jumper Wires
    d. Breadboard
    e. Micro USB cable
3. Circuit Schematic

Camera Pin	**ESP32 Pin**
3.3 V	3.3 V
GND	GND
SIOC/SCL	GPIO22/SCL
SIOD/SDA	GPIO21/SDA
VSYNC/VS	GPIO34
HREF/HS	GPIO35
PCLK/PLK	GPIO33

XCLK/XLK	GPIO32
D7	GPIO4
D6	GPIO12
D5	GPIO13
D4	GPIO14
D3	GPIO15
D2	GPIO16
D1	GPIO17
D0	GPIO27
RESET/RET	EN



4. Source Code

```
#include "OV7670.h"
#include <WiFi.h>
#include <WiFiMulti.h>
#include <WiFiClient.h>
#include "BMP.h"
#include "Config.h"

const int SIOD = 21; //SDA
const int SIOC = 22; //SCL

const int VSYNC = 34;
const int HREF = 35;
```

```
const int XCLK = 32;
const int PCLK = 33;

const int D0 = 27;
const int D1 = 17;
const int D2 = 16;
const int D3 = 15;
const int D4 = 14;
const int D5 = 13;
const int D6 = 12;
const int D7 = 4;

//DIN <- MOSI 23
//CLK <- SCK 18

OV7670 *camera;

WiFiMulti wifiMulti;
WiFiServer server(80);

unsigned char bmpHeader[BMP::headerSize];

void serve()
{
  WiFiClient client = server.available();
  if (client)
  {
    //Serial.println("New Client.");
    String currentLine = "";
    while (client.connected())
    {
      if (client.available())
      {
        char c = client.read();
        //Serial.write(c);
        if (c == '\n')
        {
          if (currentLine.length() == 0)
          {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();
            client.print(
              "<style>body{margin: 0}\nimg{height: 100%; width: auto}</style>"
              "<img id='a' src='/camera' onload='this.style.display=\"initial\";
var     b     =     document.getElementById(\"b\");     b.style.display=\"none\";
b.src=\"camera?\"+Date.now(); '>"
              "<img     id='b'     style='display:     none'     src='/camera'
onload='this.style.display=\"initial\"; var a = document.getElementById(\"a\");
a.style.display=\"none\"; a.src=\"camera?\"+Date.now(); '>");
            client.println();
            break;
          }
          else
          {
            currentLine = "";
          }
        }
        else if (c != '\r')
        {
          currentLine += c;
        }
```

```
        if(currentLine.endsWith("GET /camera"))
        {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:image/bmp");
            client.println();

            client.write(bmpHeader, BMP::headerSize);
            client.write(camera->frame, camera->xres * camera->yres * 2);
        }
      }
    }
    // close the connection:
    client.stop();
    //Serial.println("Client Disconnected.");
  }
}

void setup()
{
  Serial.begin(115200);

  wifiMulti.addAP(ssid1, password1);
  //wifiMulti.addAP(ssid2, password2);
  Serial.println("Connecting Wifi...");
  if(wifiMulti.run() == WL_CONNECTED) {
      Serial.println("");
      Serial.println("WiFi connected");
      Serial.println("IP address: ");
      Serial.println(WiFi.localIP());
  }

  camera = new OV7670(OV7670::Mode::QQVGA_RGB565, SIOD, SIOC, VSYNC, HREF, XCLK,
PCLK, D0, D1, D2, D3, D4, D5, D6, D7);
  BMP::construct16BitHeader(bmpHeader, camera->xres, camera->yres);

  server.begin();
}

void loop()
{
  camera->oneFrame();
  serve();
}
```

Note that you must change WiFi SSID and password in the "Config.h" source code to match your selected access point.

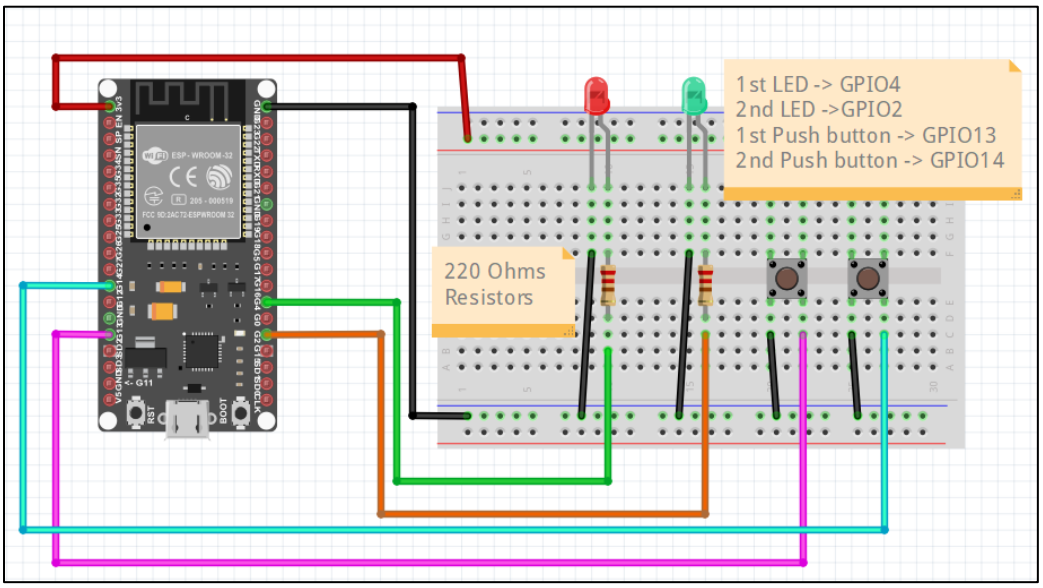
5. Result

Find assigned IP address to ESP32 in the serial monitor. In a web server on your computer that connects to the same network as ESP32, type in http://<ESP32's IP Address>.

Your Testing: Take a video record where you should show the camera result at browser together with your circuit.

Part C: FSM

1. In this scenario, we are going to study about Finite State Machine (FSM).
2. Hardware Required
    a. ESP32
    b. 2 x LEDs
    c. 2 x Push Buttons
    d. 2 x 220Ω Resistors
    e. Micro USB cable
    f. Jumper Wires
    g. breadboard
3. Circuit Diagram
    a. 1st LED ->220Ω Resistor -> GPIO4
    a. 2nd LED ->220Ω Resistor ->GPIO2
    b. 1st Push button -> GPIO13
    c. 2nd Push button -> GPIO14

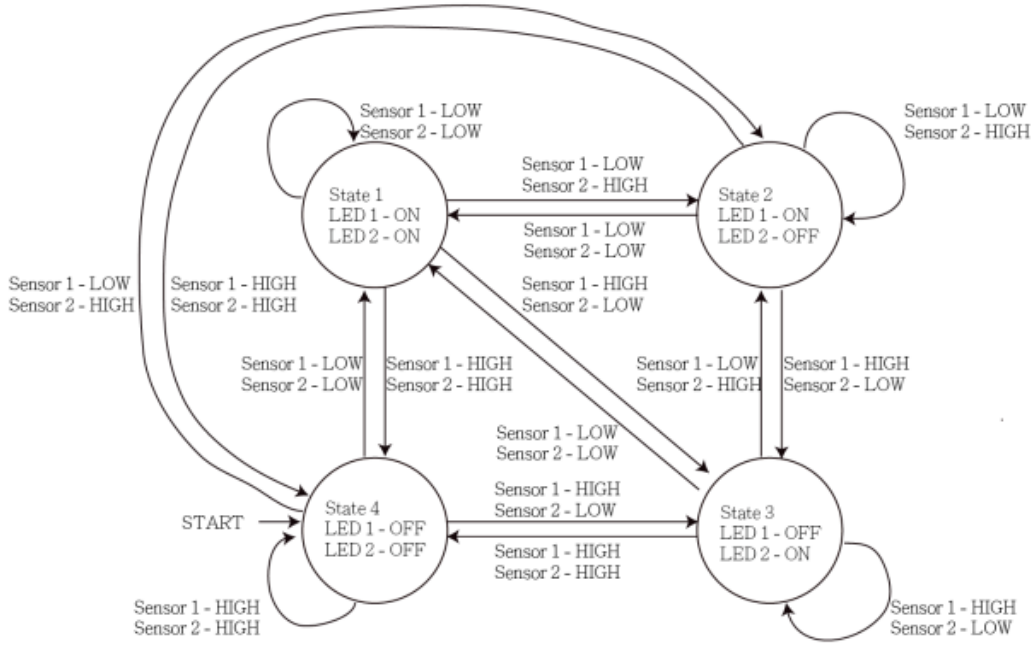


4. Source Code

Complete "`ESP32_FSM.ino`" according to the following FSM diagram.

```
//Complete this file according to the FSM
#define STATE1 1
#define STATE2 2
#define STATE3 3
#define STATE4 4
#define STATE_END 100
int sensor1 = 13;
int sensor2 = 14;
int led1 = 4;
int led2 = 2;
unsigned char state=4;
```

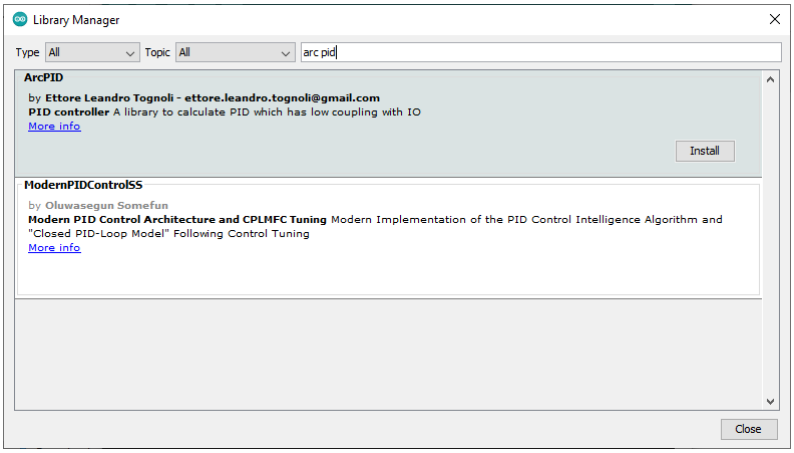
```
void setup() {
  pinMode (sensor1, INPUT);
  pinMode (sensor2, INPUT);
  pinMode (led1, OUTPUT);
  pinMode (led2, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  Serial.println(state);
  Serial.println(digitalRead(sensor1));
  Serial.println(digitalRead(sensor2));
  switch(state) {
    case STATE1:
      digitalWrite(led1, HIGH);
      digitalWrite(led2, HIGH);
      if((digitalRead(sensor1)==LOW) && (digitalRead(sensor2)==HIGH))
        state = STATE2;
      else if((digitalRead(sensor1)==HIGH) && (digitalRead(sensor2)==LOW))
        state = STATE3;
      else if((digitalRead(sensor1)==HIGH) && (digitalRead(sensor2)==HIGH))
        state = STATE4;
      break;

      //To be continued
  }
  delay(100);
}
```



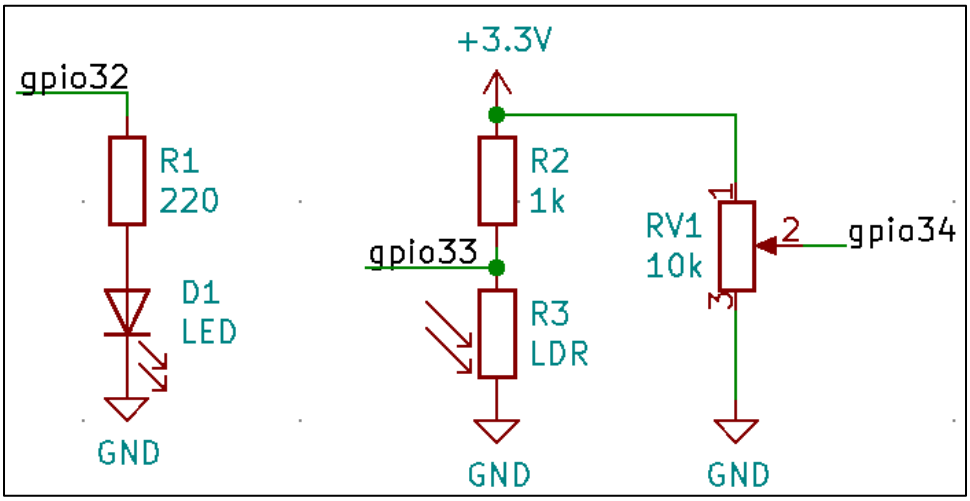
Your Testing: Take a video record where you should explain your circuit together with your complete source code.

Part D: PID

1. In this scenario, we are going to study about Proportional-Integral-Derivative (PID) with ESP32, LDR, and POT.
2. Hardware Required
    a. ESP32
    b. LED
    c. 220 Ω resistor and 1kΩ resistor
    d. LDR
    e. 10kΩ POT
    f. Bread board
    g. Micro USB cable
    h. Jumper Wires
3. Software Required
    a. ArcPID library



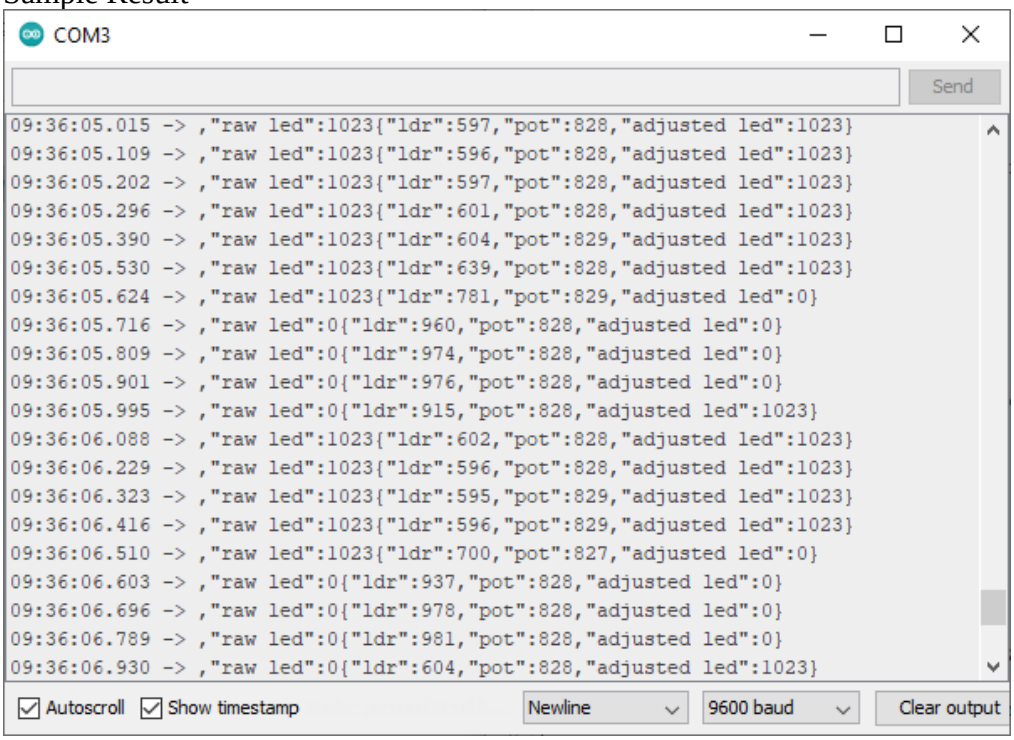
4. Circuit Schematic
    a. LED -> 220 Ohms -> GPIO32
    b. POT pin 2 -> GPIO34
    c. LDR -> GPIO33 ->1k Ohms ->3.3V



5. Source Code

```
#include <PID.h> //ArcPID library
#define LED 32 //Output
#define LDR 33 //Input sensor
#define POT 34 //This potentiometer is to set PID target
#define PWM_CH_1 0
#define PWM_FREQ 15000
#define PWM_RES 10 // Resolution in bits
#define MAX 1023.0 //10-bit max value
arc::PID<double> ledPid(5,4,3); //Kp, Ki, Kd
void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
  pinMode(LDR, INPUT);
  pinMode(POT, INPUT);
  ledcSetup(PWM_CH_1, PWM_FREQ, PWM_RES);
  ledcAttachPin(LED, PWM_CH_1);
  analogReadResolution(PWM_RES); //Set analog input resolution to be the same
as PWM
}

unsigned int ledValue = 0;
void loop() {
  delay(100);
  unsigned short ldrRaw = analogRead(LDR);
  unsigned short potRaw = analogRead(POT);
  ledPid.setTarget(potRaw);
  ledPid.setInput(ldrRaw);
  Serial.print(",\"raw led\":");
  Serial.print(ledValue,DEC);
  ledValue = min(MAX, max(0.0,ledValue + ledPid.getOutput()));
  Serial.print("{\"ldr\":");
  Serial.print(ldrRaw,DEC);
  Serial.print(",\"pot\":");
  Serial.print(potRaw,DEC);
  Serial.print(",\"adjusted led\":");
  Serial.print(ledValue,DEC);
  Serial.println("}");
  ledcWrite(PWM_CH_1, ledValue);
}
```

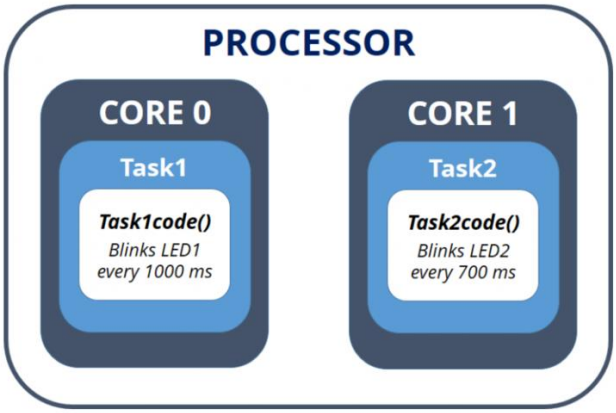
6. Sample Result



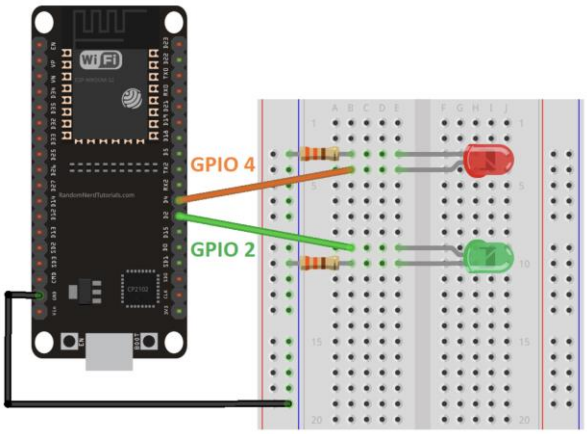
Your Testing: Take a video record where you should explain your circuit together with the result from serial monitor.

Part E: ESP32 Dual Core

1. In this scenario, we are going to study how to use ESP32 Dual Core with Arduino IDE. We will create two tasks running on different cores.
   Task 1 runs on Core 0 and Task 2 runs on Core 1.



2. Hardware Required
   a. ESP32
   b. 2 x LED
   c. 2 x 220 Ω resistors
   d. Bread board
   e. Micro USB cable
   f. Jumper Wires
3. Circuit Diagram



4. Source Code

```
TaskHandle_t Task1;
TaskHandle_t Task2;

// LED pins
const int led1 = 2;
const int led2 = 4;

void setup() {
  Serial.begin(115200);
```

```
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);

  //create a task that will be executed in the Task1code() function, with
priority 1 and executed on core 0
  xTaskCreatePinnedToCore(
                    Task1code,   /* Task function. */
                    "Task1",     /* name of task. */
                    10000,       /* Stack size of task */
                    NULL,        /* parameter of the task */
                    1,           /* priority of the task */
                    &Task1,      /* Task handle to keep track of created task
*/
                    0);          /* pin task to core 0 */
  delay(500);

  //create a task that will be executed in the Task2code() function, with
priority 1 and executed on core 1
  xTaskCreatePinnedToCore(
                    Task2code,   /* Task function. */
                    "Task2",     /* name of task. */
                    10000,       /* Stack size of task */
                    NULL,        /* parameter of the task */
                    1,           /* priority of the task */
                    &Task2,      /* Task handle to keep track of created task
*/
                    1);          /* pin task to core 1 */
    delay(500);
}

//Task1code: blinks an LED every 1000 ms
void Task1code( void * pvParameters ){
  Serial.print("Task1 running on core ");
  Serial.println(xPortGetCoreID());

  for(;;){
    digitalWrite(led1, HIGH);
    delay(1000);
    digitalWrite(led1, LOW);
    delay(1000);
  }
}

//Task2code: blinks an LED every 700 ms
void Task2code( void * pvParameters ){
  Serial.print("Task2 running on core ");
  Serial.println(xPortGetCoreID());

  for(;;){
    digitalWrite(led2, HIGH);
    delay(700);
    digitalWrite(led2, LOW);
    delay(700);
  }
}

void loop() {}
```

Your Testing: Take a video record where you should explain your circuit together with the result from serial monitor.