**Materiais Necessários**

* 1x ESP32
* 1x Sensor de chuva (placa com eletrodos + módulo comparador, ex: YL-83)
* Jumpers macho/macho
* Protoboard
* (Opcional) 1x LED e 1x resistor de 220Ω para alerta visual

**1. Conecte o módulo do sensor de chuva na protoboard**

Geralmente o módulo tem 4 pinos: **VCC, GND, AO, DO**

**2. Faça as conexões com o ESP32**

* **VCC (módulo)** → **3V3 (ESP32)**
* **GND (módulo)** → **GND (ESP32)**
* **DO (módulo)** → **GPIO 13 (ou qualquer pino digital do ESP32)**
* (Opcional) **AO (módulo)** → **GPIO 34 (entrada analógica)** se quiser ler o nível de água

**3. (Opcional) Conecte um LED de alerta**

* **Anodo (+) do LED** → **GPIO 25 (ou outro pino digital livre)**
* **Catodo (-) do LED** → **Resistor de 220Ω** → **GND**

**Resumo das conexões**

| **Sensor de chuva** | **ESP32** |
| --- | --- |
| VCC | 3V3 |
| GND | GND |
| DO | GPIO 13 |
| AO (opcional) | GPIO 34 |

## ****Materiais Necessários****

* 1x ESP32
* 1x Sensor de fumaça (MQ-2, MQ-135, etc.)
* Jumpers macho/macho
* Protoboard
* (Opcional) LED e resistor de 220Ω (para alerta visual)

## ****Passo a Passo de Montagem****

### ****1. Conecte o sensor à protoboard****

### ****2. Faça as conexões com o ESP32****

* **VCC (sensor)** → **3V3 (ESP32)**
* **GND (sensor)** → **GND (ESP32)**
* **AO (sensor)** → **GPIO 35 (ou outro pino analógico)**
* (Opcional) **DO (sensor)** → **GPIO 12 (ou outro pino digital)**

### ****3. (Opcional) Conecte um LED de alerta****

* **Anodo do LED** → **GPIO 25 do ESP32**
* **Catodo do LED** → **Resistor de 220Ω** → **GND**

## 🔌 ****Resumo das conexões****

| **Sensor de Fumaça** | **ESP32** | |
| --- | --- | --- |
| VCC | | 3V3 | |
| GND | | GND | |
| AO | | GPIO 35 | |
| DO (opcional) | | GPIO 12 | |

## ****Passo a Passo para a criação do código****

### ****1. abra o Visual Studio Code****

### ****2. Crie um novo projeto na extensão plataforma.IO para poder começar a criar o código****

### 

## ****Criação do código****

**#include <arduino.h>**

**#include <LiquidCrystal\_I2C.h>**

**LiquidCrystal\_I2C lcd(0x27, 20, 4);**

**void serialRead(char \*buffer, int maxSize);**

**//\* VARIAVEIS DA LEITURA DO TEOR DE CO**

**const int smokePin = 34;**

**unsigned long lastReadTime = 0;**

**const unsigned long interval = 1000;**

**//\* VARIAVEIS DA LEITURA DA UMIDADE**

**const int rainSensorPin = 35;**

**unsigned long lastCheckTime = 0;**

**void setup()**

**{**

**lcd.init();**

**lcd.backlight();**

**Serial.begin(9600);**

**Serial.println("Sensor de fumaça iniciado...");**

**Serial.println("Sensor de chuva iniciado...");**

**Serial.println("Para ler o teor de umidade e a concentracao de CO, digite 'leitura'.");**

**lcd.print("Chuva: ");**

**lcd.setCursor(8, 0);**

**lcd.print("Gases no ar:");**

**}**

**void loop()**

**{**

**//\* VARIAVEIS QUE REGISTRAM OS VALORES DAS LEITURAS, PARA SEREM CHAMADOS VIA COMANDO**

**static int COContent;**

**static bool rainState;**

**//\* LEITURA DO TEOR DE CO**

**unsigned long currentTime = millis();**

**if (currentTime - lastReadTime >= interval)**

**{**

**lastReadTime = currentTime;**

**int smokeValue = analogRead(smokePin);**

**//? Mapeia para escala de 0 a 10**

**int intensity = map(smokeValue, 300, 2000, 0, 2000);**

**intensity = constrain(intensity, 0, 2000);**

**COContent = intensity;**

**if (intensity > 0)**

**{**

**lcd.setCursor(9, 1);**

**lcd.print(intensity);**

**lcd.print(" ");**

**if (intensity < 750)**

**{**

**lcd.setCursor(8, 2);**

**lcd.print("Seguro ");**

**lcd.setCursor(8, 3);**

**lcd.print(" ");**

**}**

**else if (intensity >= 750 && intensity < 1500)**

**{**

**lcd.setCursor(9, 2);**

**lcd.print("Alerta");**

**lcd.setCursor(8, 3);**

**lcd.print("Use EPI ");**

**}**

**else if (intensity >= 1500)**

**{**

**lcd.setCursor(9, 2);**

**lcd.print("Perigo");**

**lcd.setCursor(8, 3);**

**lcd.print("Saia rapido!");**

**}**

**}**

**}**

**//\* LEITURA DA UMIDADE**

**if (currentTime - lastCheckTime >= interval)**

**{**

**lastCheckTime = currentTime;**

**int rainValue = analogRead(rainSensorPin);**

**int intensity = map(rainValue, 2500, 500, 0, 10);**

**intensity = constrain(intensity, 0, 10);**

**if (intensity > 0)**

**{**

**lcd.setCursor(0, 1);**

**lcd.print("Molhado");**

**rainState = true;**

**}**

**else**

**{**

**lcd.setCursor(0, 1);**

**lcd.print("Seco ");**

**rainState = false;**

**}**

**}**

**if (Serial.available())**

**{**

**static char reading[100];**

**serialRead(reading, 100);**

**if (strlen(reading) > 0)**

**{**

**if (strcmp(reading, "leitura") == 0)**

**{**

**sprintf(reading, "Concentracao de CO: %d. Estado da umidade: %s.", COContent, rainState ? "Umido" : "Seco");**

**Serial.println(reading);**

**}**

**}**

**}**

**}**

**//\* LEITURA DA SERIAL**

**void serialRead(char \*buffer, int maxSize)**

**{**

**static int i = 0; //? Indice estatico para manter posiçao entre chamadas**

**while (Serial.available() > 0 && i < maxSize - 1) //? Enquanto houver dados e espaco no buffer**

**{**

**char c = Serial.read(); //? Le proximo caractere da serial**

**if (c == '\n' || c == '\r') //? Verifica se eh caractere de fim de linha**

**{**

**if (i > 0) //? Se ja recebeu caracteres validos**

**{**

**buffer[i] = '\0'; //? Termina a string com null**

**i = 0; //? Reseta indice para proxima leitura**

**return; //? Retorna com comando completo**

**}**

**i = 0; //? Linha vazia - reseta e ignora**

**}**

**else**

**{**

**buffer[i++] = c; //? Armazena caractere valido no buffer**

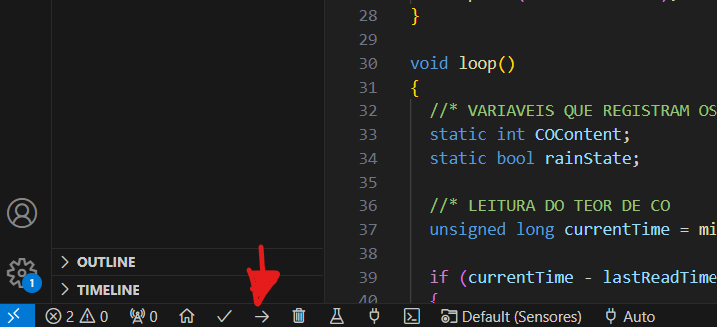
**}**

**}**

**buffer[i] = '\0'; //? Garante terminaçao nula para strings parciais**

**}**

* **Depois de digitar o código, faça o upload para o esp32 e esta pronto para uso.**

****