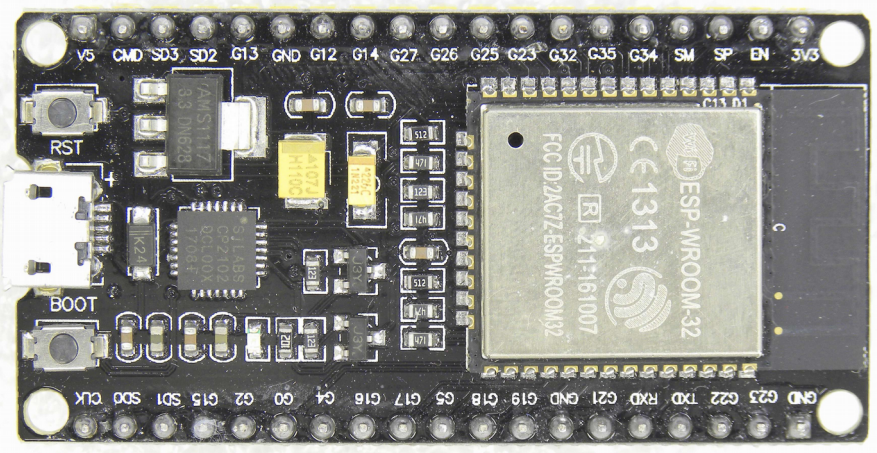
Prof. Israel Gomes (SILVA, I.G.)

Periféricos Integrados e GPIO



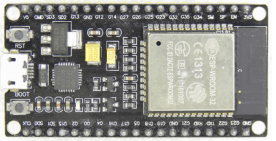
Introdução aos periféricos do ESP32:

GPIO (General Purpose Input/Output):

- Os pinos GPIO do ESP32 podem ser configurados como entrada ou saída, permitindo interagir com uma ampla variedade de dispositivos externos, como LEDs, sensores, motores, e mais.

- Cada pino pode ser configurado para diferentes funções, como leitura de sinais digitais (entrada) ou acionamento de atuadores (saída).

Periféricos Integrados e GPIO



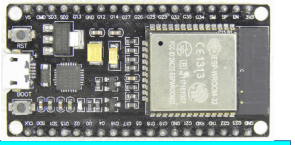
ADC (Analog-to-Digital Converter):

O ESP32 possui conversores ADC integrados que permitem ler sinais analógicos (como de sensores) e convertê-los para valores digitais.

O ADC tem uma resolução de 12 bits, o que significa que ele pode representar valores analógicos com até 4096 níveis de precisão (0 a 4095).

DAC (Digital-to-Analog Converter):

O ESP32 possui conversores DAC, permitindo converter valores digitais em sinais analógicos. Isso é útil para controlar dispositivos analógicos como motores ou ajustar a intensidade de LEDs.

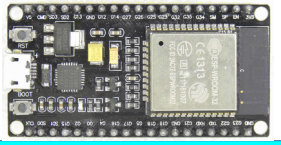


Prof. Israel Gomes (SILVA, I.G.)

PWM (Pulse Width Modulation):

O **PWM** é uma técnica usada para gerar sinais analógicos a partir de sinais digitais. No ESP32, pode ser usado para controlar a velocidade de motores, o brilho de LEDs, e outras aplicações que requerem modulação de potência.

O **duty cycle** controla a relação entre o tempo "**ligado**" e o tempo "**desligado**" do sinal, ajustando a potência entregue ao dispositivo.



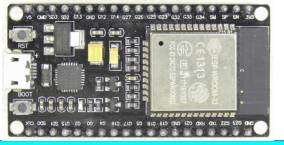
Prof. Israel Gomes (SILVA, I.G.)

Uso de GPIO para leitura e controle de dispositivos externos:

Exemplos de controle de LEDs e leitura de botões.
Explicação de como configurar pinos GPIO como entrada ou saída:

```
pinMode(LED_BUILTIN, OUTPUT);  // Configura pino como saída
digitalWrite(LED_BUILTIN, HIGH);  // Liga o LED
pinMode(buttonPin, INPUT);  // Configura pino como entrada
int buttonState = digitalRead(buttonPin);  // Lê o estado do botão
```

Int* => indica o endereco de memória onde um valor inteiro (ou uma sequência

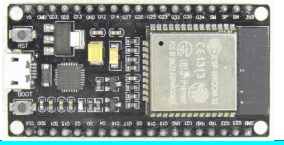


Prof. Israel Gomes (SILVA, I.G.)

Uso de GPIO para leitura e controle de dispositivos externos:

Exemplos de controle de LEDs e leitura de botões.
Explicação de como configurar pinos GPIO como entrada ou saída:

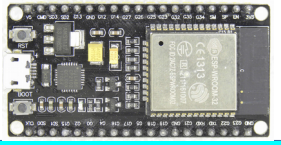
```
const int LED_BUILTIN = 2;     // Pino do LED externo no GPIO 2
const int buttonPin = 12;      // Pino onde o botão está conectado

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);       // Configura o LED no GPIO 2 como saída
  pinMode(buttonPin, INPUT_PULLDOWN); // Configura o pino do botão como entrada com pull-d
}

void loop() {
  int buttonState = digitalRead(buttonPin);  // Lê o estado do botão

  if (buttonState == HIGH) {
    digitalWrite(LED_BUILTIN, HIGH);  // Liga o LED externo se o botão estiver pressionado
  } else {
    digitalWrite(LED_BUILTIN, LOW);   // Desliga o LED externo se o botão não estiver pres
  }
}
```



Prof. Israel Gomes (SILVA, I.G.)

Leitura de Temperatura com ADC e Exibição no Display (parte 1 de 2)

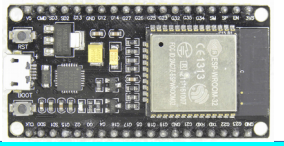
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Configurações do LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Endereço I2C do display e dimensões

const int sensorPin = 34;  // Pino ADC onde o sensor está conectado

void setup() {
  // Inicializa o LCD
  lcd.init();       // Inicializa o display
  lcd.backlight();  // Liga a luz de fundo do LCD
  lcd.print("Temp: "); // Exibe "Temp: " no LCD

  pinMode(sensorPin, INPUT); // Configura o pino do sensor como entrada
}
```



Prof. Israel Gomes (SILVA, I.G.)

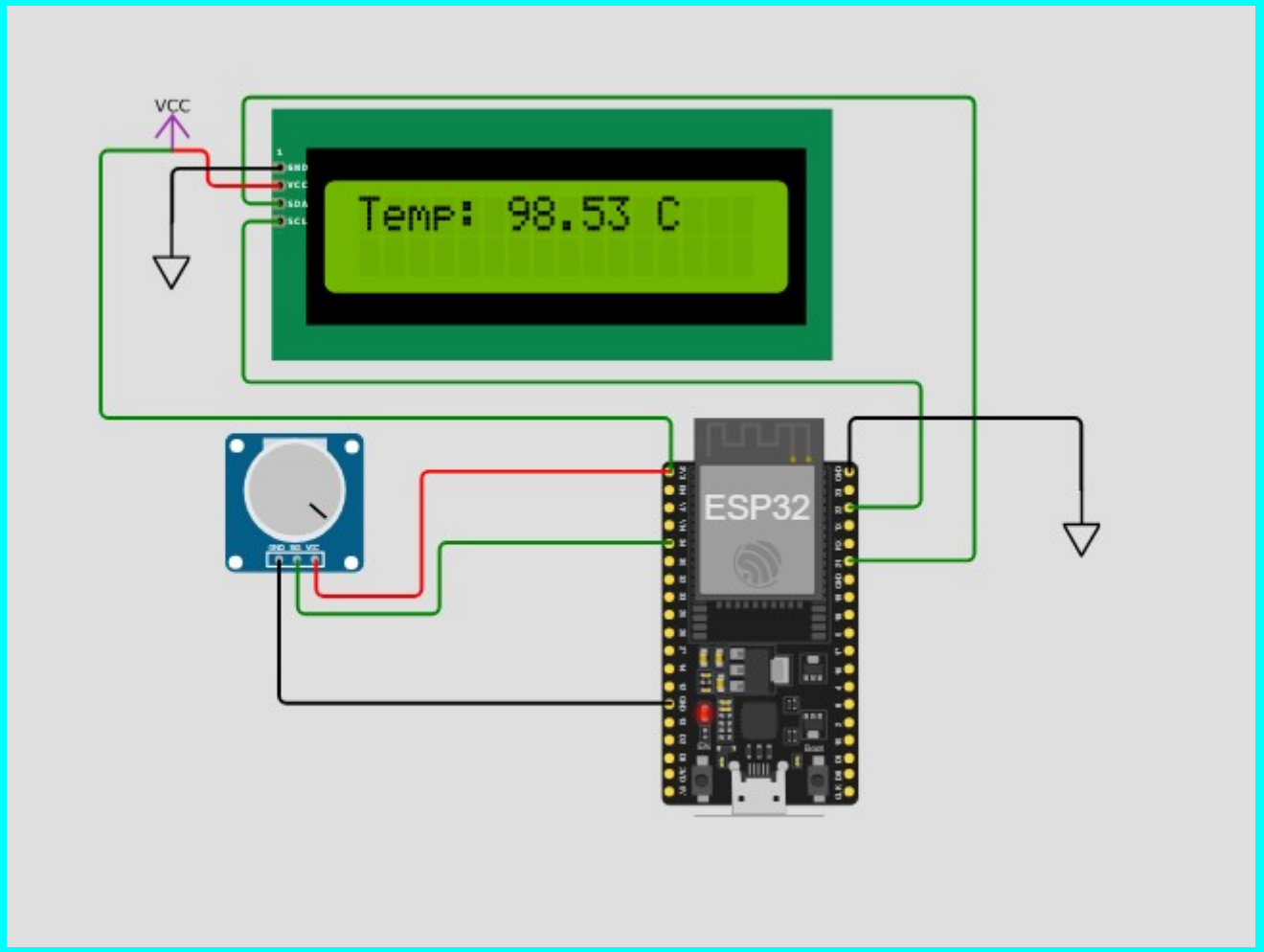
Leitura de Temperatura com ADC e Exibição no Display (parte 2 de 2)

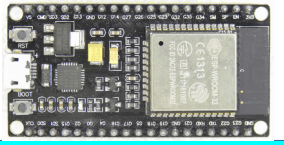
```
void loop() {
  int sensorValue = analogRead(sensorPin);  // Lê o valor analógico do sensor (ADC)

  // Converte o valor lido em uma escala de temperatura (0 a 100°C)
  float voltage = sensorValue * (3.3 / 4095.0); // Converte para tensão (0 a 3.3V)
  float temperature = (voltage / 3.3) * 100.0;  // Converte a tensão em temperatura (0V =

  // Exibe a temperatura no LCD
  lcd.setCursor(6, 0);          // Move o cursor para a posição da temperatura
  lcd.print(temperature);       // Exibe a temperatura
  lcd.print(" C  ");            // Exibe a unidade "C"

  delay(1000); // Atualiza a cada 1 segundo
}
```



Prof. Israel Gomes (SILVA, I.G.)

Leitura de Temperatura com ADC e Exibição no Display (Circuito)

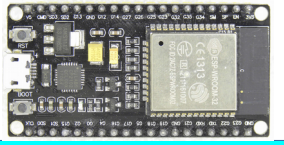




Prof. Israel Gomes (SILVA, I.G.)

Conversor simulando DAC com leds (Ver material complementar) (parte 1 de 3)

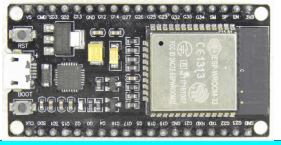
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Configurações do LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Endereço I2C do display e dimensões

const int sensorPin = 34;  // Pino ADC onde o sensor está conectado
const int redPin = 27;      // Pino onde o LED vermelho está conectado
const int greenPin = 25;    // Pino onde o LED verde está conectado
const int yellowPin = 26;   // Pino onde o LED amarelo está conectado

void setup() {
  // Inicializa o LCD
  lcd.init();       // Inicializa o display
  lcd.backlight();  // Liga a luz de fundo do LCD
  lcd.print("Temp: "); // Exibe "Temp: " no LCD

  pinMode(sensorPin, INPUT); // Configura o pino do sensor como entrada
  pinMode(redPin, OUTPUT);    // Configura o pino do LED vermelho como saída
  pinMode(greenPin, OUTPUT);  // Configura o pino do LED verde como saída
  pinMode(yellowPin, OUTPUT); // Configura o pino do LED amarelo como saída
}
```



Prof. Israel Gomes (SILVA, I.G.)

Conversor simulando DAC com leds (Ver material complementar) (parte 2 de 3)

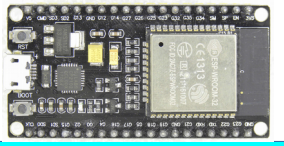
```
void loop() {
  int sensorValue = analogRead(sensorPin);  // Lê o valor analógico do sensor (ADC)

  // Converte o valor lido em uma escala de temperatura (0 a 100°C)
  float voltage = sensorValue * (3.3 / 4095.0); // Converte para tensão
  float temperature = (voltage / 3.3) * 100.0;  // Agora o valor vai de 0 a 100

  // Exibe a temperatura no LCD
  lcd.setCursor(6, 0);          // Move o cursor para a posição da temperatura
  lcd.print(temperature);       // Exibe a temperatura
  lcd.print(" C   ");           // Exibe a unidade "C"

  // Controle dos LEDs com base na temperatura
```



Prof. Israel Gomes (SILVA, I.G.)

Conversor simulando DAC com leds (Ver material complementar) (parte 3 de 3)

```
  // Controle dos LEDs com base na temperatura
  if (temperature <= 26) {
    // Verde
    digitalWrite(greenPin, HIGH);
    digitalWrite(yellowPin, LOW);
    digitalWrite(redPin, LOW);
  } else if (temperature <= 30) {
    // Amarelo
    digitalWrite(greenPin, LOW);
    digitalWrite(yellowPin, HIGH);
    digitalWrite(redPin, LOW);
  } else {
    // Vermelho
    digitalWrite(greenPin, LOW);
    digitalWrite(yellowPin, LOW);
    digitalWrite(redPin, HIGH);
  }

  delay(1000); // Atualiza a cada 1 segundo
}
```



Prof. Israel Gomes (SILVA, I.G.)

Conversor simulando DAC com leds (Ver material complementar) (parte 3 de 3)