UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROJETOS DE SISTEMAS EMBUTIDOS

Laboratório 0 – Pisca LED

Vincent Pernarh

BELO HORIZONTE 2024/2

Professor : Diogenes Cecilio da Silva Junior

1. **INTRODUÇÃO**

Este relatório tem como objetivo apresentar os conhecimentos fundamentais de projeto de sistemas embarcados. Serão realizadas diversas tarefas, incluindo a escolha e descrição do microcontrolador e seus periféricos, com o intuito de realizar um experimento de acionamento de LED controlado por um botão (Sistema Reativo). A Seção 4 abordará os procedimentos para a execução da prática e as instruções a serem seguidas.

2. **DESCRIÇÃO DO SDK E PERIFÉRICOS**

Para o projeto, utilizaremos o SDK do “ESP32-WROOM”, que é uma plataforma poderosa e versátil para o desenvolvimento de sistemas embarcados. A seguir, apresento os detalhes técnicos:

a. Nome do SDK: ESP32-WROOM
b. Modelo: ESP32-WROOM-32
c. Microcontrolador: ESP32-D0WDQ6 (dual-core Xtensa® 32-bit LX6)

d. **Memórias**
- Flash: 4MB
- SRAM: 520KB

e. Interfaces de E/S:
- 34 pinos GPIO
- 12 pinos de entrada analógica (ADC)
- 2 pinos de saída analógica (DAC)
- Interfaces de comunicação: UART, SPI, I2C, PWM, I2S, CAN
- Conectividade Wi-Fi e Bluetooth integrados

f. Localização dos Sinais de Interface

Abaixo está a imagem com a localização de todos os sinais de interface disponíveis no ESP32-WROOM-32, facilitando a conexão e utilização dos periféricos durante o desenvolvimento : Figure 1.

g. Ambiente de Desenvolvimento de Software.

O ambiente de desenvolvimento utilizado será o **Arduino IDE**, que oferece suporte completo para a programação e compilação de projetos com o ESP32-WROOM. O Arduino IDE permite um processo de desenvolvimento simplificado, utilizando uma ampla gama de bibliotecas e exemplos para controle de periféricos e funcionalidades integradas do microcontrolador.

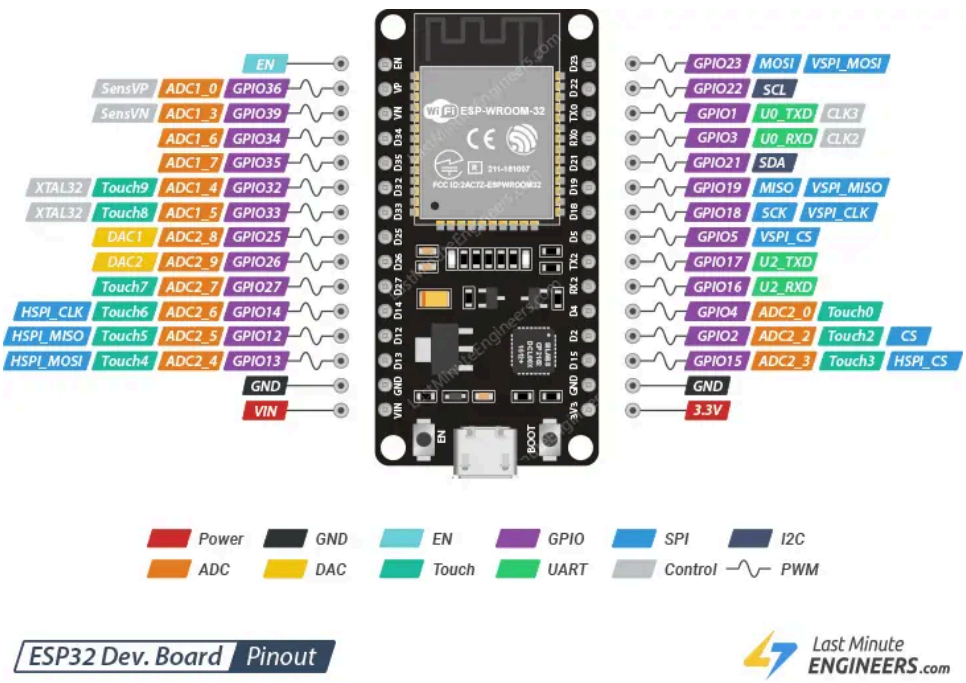

Figure 1: [ESP32-WROOM - 32 PinOut]

3. INTERFACE GPIO DE ENTRADA.

Em um circuito elétrico, o resistor de pull-up é usado para garantir um estado conhecido para um sinal,geralmente quando utilizamos o resistor pull-up interno do ESP32-WROOM, evitamos a necessidade de adicionar um resistor externo ao circuito. A lógica por trás do uso do resistor interno é simples: o ESP32 possui resistores pull-up e pull-down configuráveis em seus pinos GPIO,no caso deste projeto, conectamos a GPIO 4, que podem ser habilitados via software.

Neste caso, ao configurar o pino de entrada para usar o resistor pull-up interno, garantimos que, quando a chave estiver fechada/ pressionada, o pino é conectado diretamente ao 3.3V, resultando em um nível alto (HIGH), permitindo que o microcontrolador detecte essa mudança de estado.

No IDE do Arduino, essa função pode ser configurada usando o comando **pinMode()**, que recebe dois parâmetros: **pinMode(número do pino, configuração)**. No caso de um botão, podemos configurar como **INPUT_PULLUP**, utilizando o resistor pull-up interno do ESP32. Sabe-se que esse resistor pull-up também pode ser implementado externamente, utilizando um resistor de pelo menos 10kΩ conectado entre o pino do botão e o GND do ESP32. Nesse caso, o botão precisaria de uma alimentação externa, e um dos pinos seria a entrada do ESP32. Com essa configuração, usamos **INPUT** em vez de I**NPUT_PULLUP**. No entanto, por simplicidade, utilizaremos o resistor pull-up interno com a configuração **INPUT_PULLUP**.

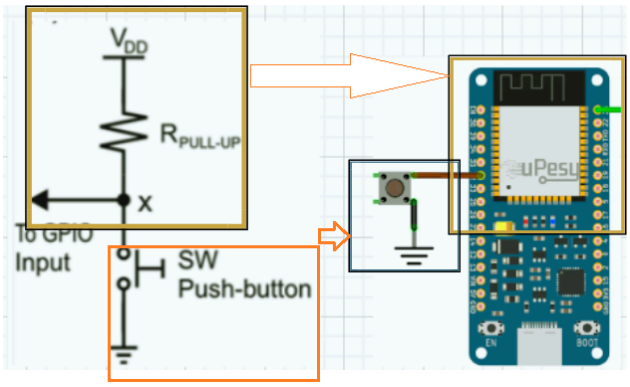

Figura 2 : Representação do resistor interno do ESP32-wroom (fonte : misto)

4. INTERFACE GPIO DE SAIDA.

A saída do projeto consiste em um LED conectado a um pino GPIO configurado como saída no ESP32. O LED será ligado ao pino GPIO de modo que, quando o pino estiver em nível alto (HIGH), o LED acenderá, e quando estiver em nível baixo (LOW), o LED apagará.Neste projeto, o LED foi conectado ao GPIO 15, onde o ânodo (terminal positivo) do LED é conectado em série com um resistor de 330Ω, enquanto o cátodo (terminal negativo) é conectado ao pino de saída (GPIO) do ESP32.A função do resistor é para limitar a corrente, protegendo tanto o LED quanto o microcontrolador.

No código, o pino GPIO será configurado como saída utilizando o comando **`pinMode(pino, OUTPUT)`** no setup do Arduino IDE, onde o comando digitalWrite() é usado amarrando o pino e estado a ele assim o LED acende ou apaga.

No diagrama abaixo, é possível observar que, quando o comando **digitalWrite(pino, HIGH)** é executado, o pino GPIO recebe 3.3V do ESP32. A corrente passa pelo resistor, que limita o fluxo, permitindo que o LED conduza e acenda. Por outro lado, ao executar o comando **digitalWrite(pino, LOW)**, o pino GPIO é conectado diretamente ao GND, interrompendo a condução no LED e fazendo com que ele apague.

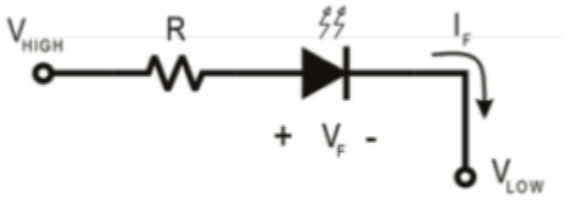

Figura 3 : Diagrama elétrico para conduzir um LED(Fonte : Slide do professor).

5. EXPERIMENTO.

Para realização do projeto, foi empregado o diagrama abaixo :

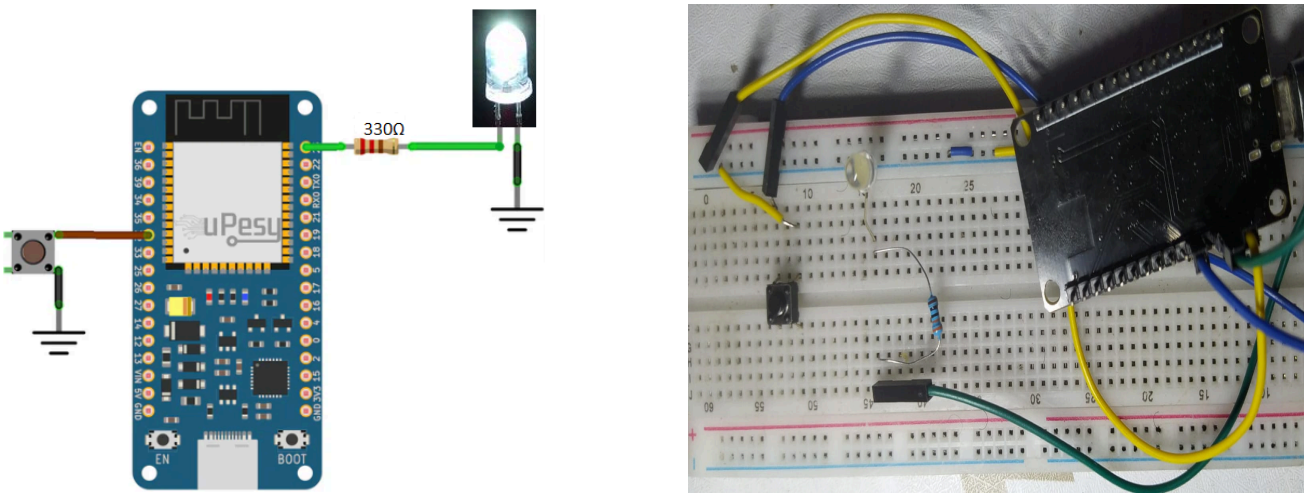

Figura 4 : Circuito elétrico esquemático e montagem real.

5.1 Requisitos do sistema :

a. Esperar a chave ser pressionada e liberada.
b. Acender o LED por 1 segundo.
c. Apagar o LED por 1 segundo.
d. Finalmente, piscar rapidamente o LED indicando fim de ciclo. Para isso acenda por 0,25 segundo e em seguida apague.
e. Volte ao passo 1.

5.2 Descrição Funcional.

O código começa definindo os pinos para o botão e o LED, além de duas variáveis para rastrear o estado do botão pressionado e não pressionado. O tempo de debounce é configurado para 10 milissegundos, o que ajuda a evitar leituras incorretas durante o acionamento do botão.

Na função **setup()**, o pino do botão é configurado como entrada com resistor pull-up, enquanto o pino do LED é configurado como saída, iniciando o LED apagado. A comunicação Serial é ativada para depuração.

A função **loop()** espera até que o botão seja pressionado. Uma vez detectado, ele aguarda que o botão seja liberado, respeitando o tempo de debounce. Quando o botão é pressionado e solto, o LED é aceso por um segundo.

Após isso, o LED é apagado, e uma sequência adicional de acendimento do LED por 250 milissegundos é executada para indicar o fim do processo e depois aguardam que o botão seja pressionado de novo para repetir o círculo.

6. **CONCLUSÃO**

O projeto foi concluído com sucesso, atendendo a todos os requisitos desejados. É importante destacar que os conhecimentos básicos em circuitos eletrônicos foram fundamentais para a execução do trabalho, facilitando o processo. Um vídeo demonstrando o funcionamento do sistema foi carregado no YouTube e será apresentado em sala de aula.

7. **REFERENCIAS**

a. https://lastminuteengineers.com/?__im-QrcQOQJp=15173755891247095350
b. https://www.upesy.com/blogs/tutorials/how-to-use-gpio-pins-of-esp32-with-arduino
c. https://lastminuteengineers.com/?__im-fcFKXFbU=8975801794328265667#google_vignette

8. **Codigo**

```
// Define the pin numbers
const int buttonPin = 4;    // GPIO pino para botão
const int ledPin = 15;      // GPIO pino para led


// tempo de debounce em milli segundos
long debounceDelay = 10;

bool buttonstate_pres =  LOW;
bool buttonstate_unpres =  HIGH;

void setup() {
  // configurando o pino de botão com o resistor interno como PULLUP
  pinMode(buttonPin, INPUT_PULLUP);

  // Pino de led como saida
  pinMode(ledPin, OUTPUT);

  // led inicializado apagado
  digitalWrite(ledPin, LOW);
```

```
  // Porta serial incializado para depuração.
  Serial.begin(115200);
}

void loop() {
  //espera o botão ser presionado.
  while(digitalRead(buttonPin)==HIGH){
    delay(debounceDelay);
    buttonstate_pres = HIGH;
  }

  //Espra se o botão ser apagado e espera o tempo de debounce.
  while(digitalRead(buttonPin) == LOW){
    delay(debounceDelay);
    buttonstate_unpres = LOW;
  }

  // Se o butão tiver presionado,estado de buttonstate_pres
  //mude para 1 e .upres que tava em 1 mude para Low.

  if (buttonstate_pres == HIGH && buttonstate_unpres == LOW) {
    // liga led por 1 segundo.
    digitalWrite(ledPin, HIGH);
    Serial.println("Butão presionado e solto");
    delay(1000);
    digitalWrite(ledPin, LOW);

    delay(1000);

    //Acende por 250ms indicando fim de curso.
    digitalWrite(ledPin, HIGH);
    delay(250);
    digitalWrite(ledPin, LOW);

  }

}
```