MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LUCIO ENZO HORIE
DANIEL ECCARD BASTOS VIVAS

MÓDULO EMBARCADO UTILIZANDO ESP32

RIO DE JANEIRO
2025

Sumário

1 Objetivos

Este trabalho tem como objetivo apresentar o desenvolvimento de um sistema embarcado utilizando microcontroladores ESP32, aplicado a um interruptor de inclinação. O sistema, ao ser conectado a uma rede sem fio, é capaz de enviar dados remotamente ao usuário. Inicialmente, o projeto foi implementado em uma protoboard para validação funcional, e posteriormente, foi elaborado o esquemático da placa de circuito impresso utilizando a plataforma EasyEDA. A leitura e o monitoramento dos dados foram realizados por meio da plataforma ThingSpeak.

2 Desenvolvimento

2.1 Código

O código foi desenvolvido a partir do ambiente de desenvolvimento integrado (IDE) ArduinoIDE. Foi configurado no código tanto a conexão via Wi-Fi quanto a lógica para coleta de dados do módulo escolhido.

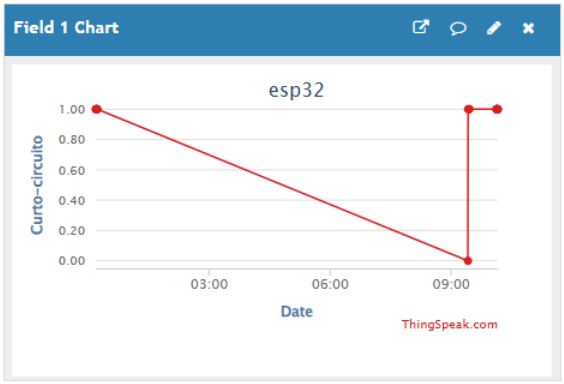


Figura 1 – Coleta de dados.

2.2 Sistema embarcado

Materiais utilizados:

1. Placa ESP32 WiFi / Bluetooth DEVKit V1
2. Módulo de interruptor de inclinação
3. Jumpers
4. Protoboard
5. Bateria

A montagem foi realizada para teste na protoboard conforme a Figura 2.

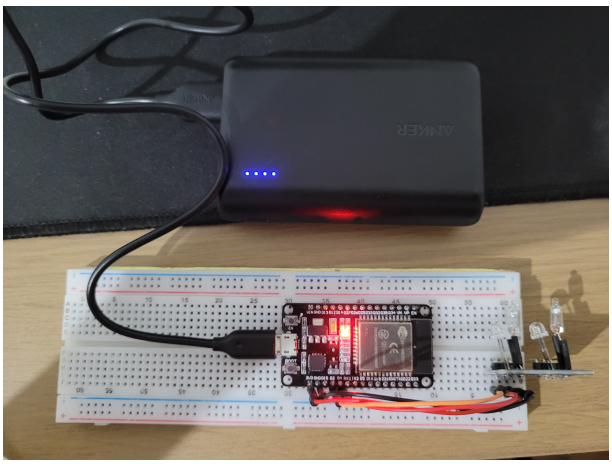


Figura 2 – Modelo montado na protoboard.

2.3 Design utilizando EasyEDA

O projeto para o circuito impresso foi desenhado conforma as Figuras 3 e 4.

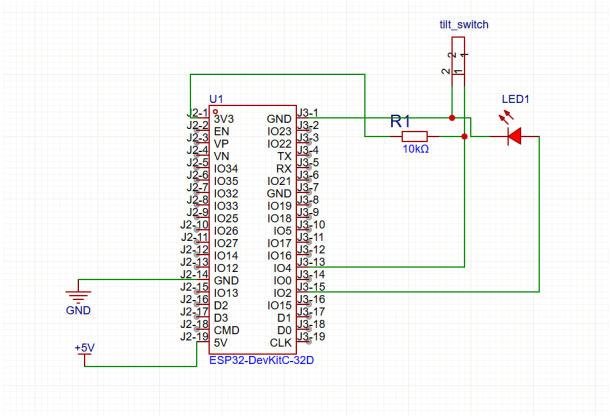
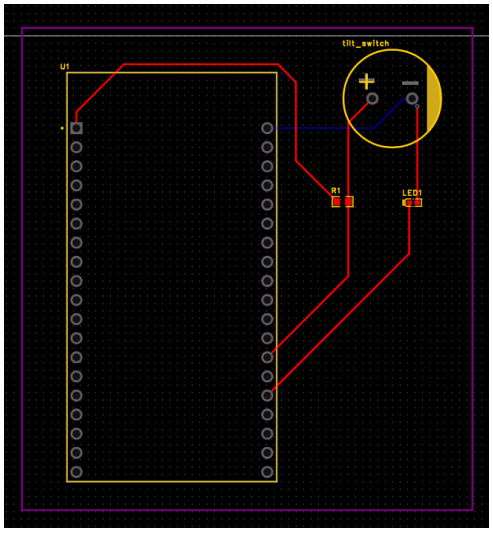


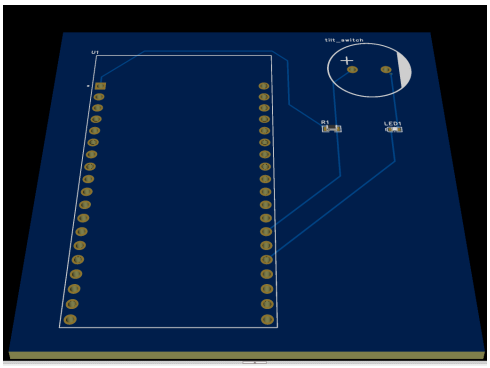
Figura 3 – Esquemático do circuito.

3 Guia para o Usuário

Para utilizar o sistema embarcado, basta conectá-lo a uma bateria. O dispositivo irá se conectar automaticamente à rede Wi-Fi previamente configurada e, uma vez estabelecida a conexão, iniciará a medição da inclinação do aparelho. A cada 15 segundos, o sistema envia aos servidores da plataforma ThingSpeak os dados referentes ao estado atual do objeto, indicando se houve ou não alteração em sua posição. O módulo apenas mede dois tipos de sinais: se o circuito foi virado ou não, acendendo um LED caso esteja virado.



(a) Placa de circuito impresso em 2D.



(b) Placa de circuito impresso em 3D.

Figura 4 – Representações do projeto eletrônico: esquemático, layout 2D e visualização 3D.

4 Resultados e Conclusões

O módulo desenvolvido para o ESP32 funcionou conforme o esperado, realizando corretamente as medições e permitindo a comunicação remota com o usuário por meio de conexão Wi-Fi. Adicionalmente, foi possível projetar virtualmente uma placa de circuito impresso utilizando a plataforma selecionada para o desenvolvimento do trabalho.

Dessa forma, o projeto apresentou resultados satisfatórios, uma vez que os objetivos estabelecidos foram alcançados.

5 Anexo

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <Arduino.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <esp_ipc.h>

const char* ssid = "####";            // Nome da Rede Wi-Fi
const char* password = ""################";";  // Senha da Rede Wi-Fi
// const int shock_sensor = 4;             // Pino ADC no ESP32
const char* server = "http://api.thingspeak.com/update";
const char* apiKey = "################";  // chave de escrita da API

// Module
int led = 2;            // LED output pin
int shock_sensor = 4;   // Sensor input pin
int value;              // Temporary variable

void readDataFromModule(void* parameters) {
  for (;;) {
    value = digitalRead(shock_sensor);  // Read current signal from sensor
    if (value == HIGH) {
      digitalWrite(led, HIGH);  // Turn on LED
      delay(1000);
    } else {
      digitalWrite(led, LOW);  // Turn off LED
    }
    Serial.println("Value: ");
    Serial.println(value);
    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
}

void setup() {
  Serial.begin(9600);
  delay(500);
  WiFi.begin(ssid, password);
  Serial.print("Conectando-se ao Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi conectado!");

```

```
45        // Module
46        pinMode(led, OUTPUT);                // Initialize output pin
47        pinMode(shock_sensor, INPUT);        // Initialize sensor pin
48        digitalWrite(shock_sensor, HIGH);  // Activate internal pull-up
              resistor
49
50        xTaskCreate(
51          readDataFromModule,
52          "Read data from the module",
53          2000,
54          NULL,
55          2,
56          NULL);
57      }
58
59
60      void loop() {
61
62
63        int digitalValue = digitalRead(shock_sensor);
64        Serial.print("Valor lido do ADC: ");
65        Serial.println(digitalValue);
66        if (WiFi.status() == WL_CONNECTED) {
67          HTTPClient http;
68          String url = String(server) + "?api_key=" + apiKey + "&field1="
                + String(digitalValue);
69          http.begin(url);
70          int httpResponseCode = http.GET();
71          if (httpResponseCode > 0) {
72            Serial.print("C digo de resposta HTTP: ");
73            Serial.println(httpResponseCode);
74          } else {
75            Serial.print("Erro ao enviar dados: ");
76            Serial.println(http.errorToString(httpResponseCode).c_str());
77          }
78          http.end();
79        } else {
80          Serial.println("WiFi desconectado. Tentando reconectar...");
81          WiFi.begin(ssid, password);
82        }
83
84        delay(15000);  // Intervalo de 15s (m nimo permitido pelo
              ThingSpeak)
85      }
```

Listing 1 – Código