MINISTÉRIO DA DEFESA
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
Seção de Engenharia Elérica (SE/3)

Projeto de Sistemas Embarcados

Trabalho 3 - Medidor de nível de ruído sonoro com ESP32

1º Ten Alberto Felipe De Lima LUCKWU Silva
Asp Of R2 Matheus Calixto de Lima PONTES

Rio de Janeiro, RJ
Junho de 2025

O seguinte trabalho teve como finalidade desenvolver, com o auxílio de um microcontrolador ESP32, um sistema para monitorar o nível de ruído em um ambiente utilizando um microfone I2S (INMP441) e enviar um valor representativo do nível sonoro para a plataforma de IoT ThingSpeak. O projeto consistiu em configurar o hardware, desenvolver o código na Arduino IDE para leitura do sensor via I2S, processamento básico do sinal (cálculo de RMS) e envio dos dados via WiFi para um canal configurado no ThingSpeak, permitindo o monitoramento remoto.

1 Descrição do sistema:

O sistema consiste em:

- **Microcontrolador ESP32**: Responsável pelo controle lógico, comunicação WiFi e processamento dos dados do sensor.
- **Microfone I2S INMP441**: Sensor responsável pela captura do áudio ambiente.
- **Pilha 12V 23A**: Para alimentação do dispositivo mantendo dimensões reduzidas.
- **Regulador de tensão L7805CV**: Regulador de tensão linear para fornecer ao ESP32 uma tensão de 5V.
- **Conexão WiFi**: Para envio dos dados para a plataforma ThingSpeak.
- **Plataforma ThingSpeak**: Serviço para recebimento, armazenamento e visualização dos dados.

Para a montagem do sistema, considerou-se o esquemático apresentado na figura 1.

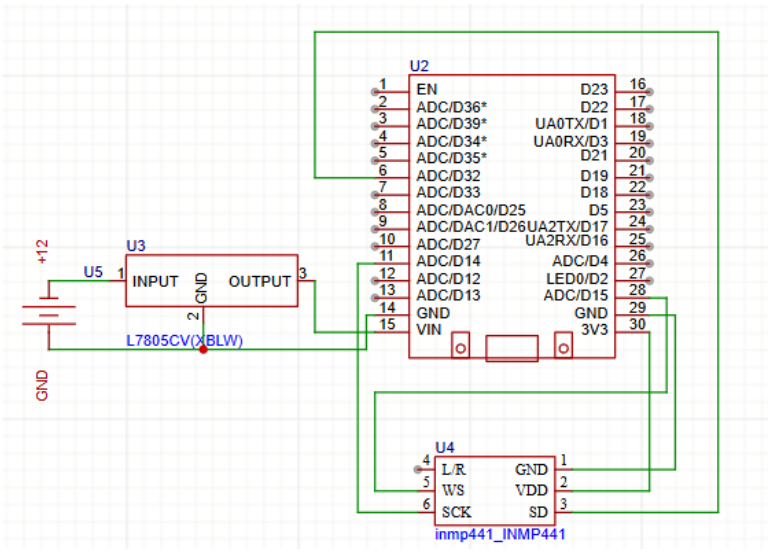

Figura 1: Esquemático do Circuito ESP32 com INMP441

Os componentes foram dispostos e conectados na placa da seguinte forma:

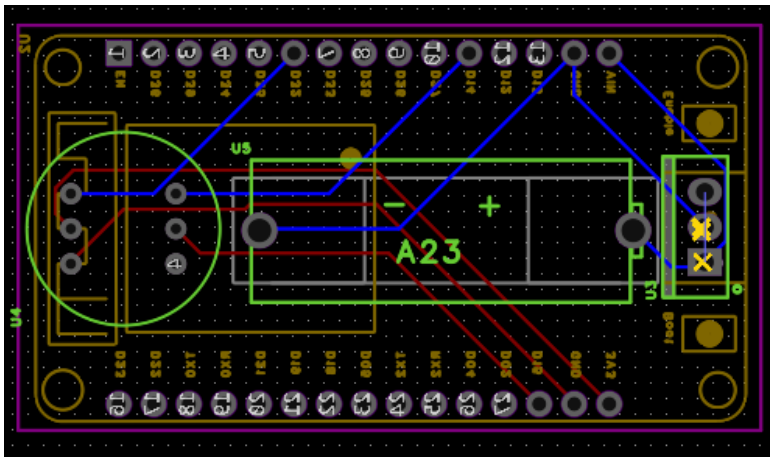

Figura 2: Footprint das conexões da PCB projetada

Dessa forma, pode-se visualizar abaixo como ficaria o resultado final do projeto, faltando apenas posicionar o componente microfone e o suporte para a pilha 12V 23A:

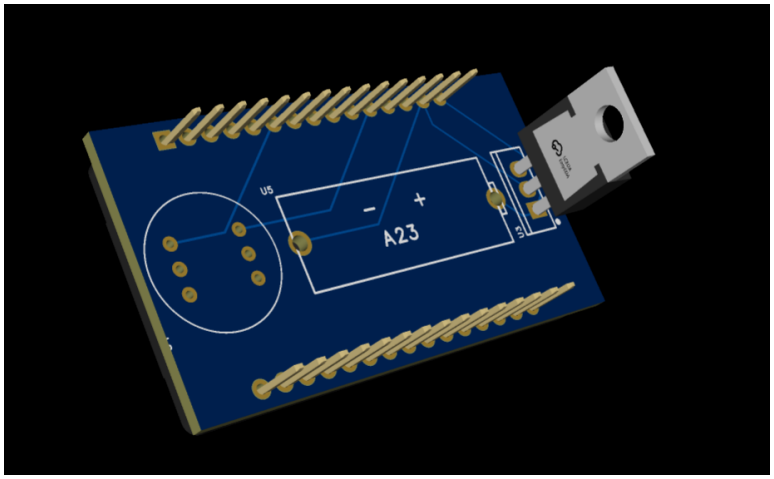

Figura 3: Visão frontal do dispositivo medidor de ruído

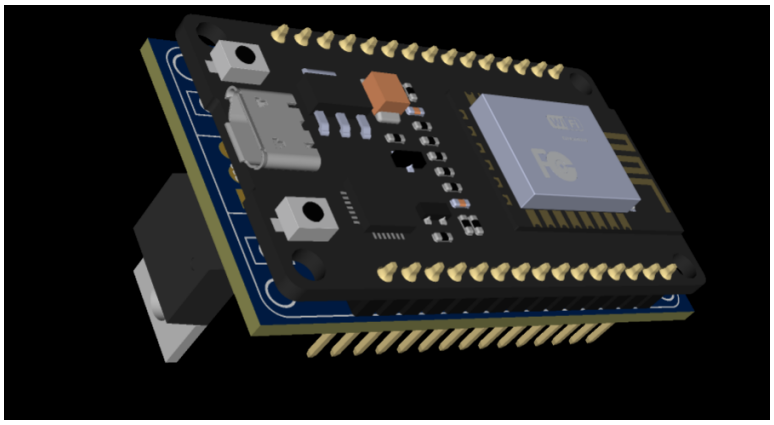

Figura 4: Visão traseira do dispositivo medidor de ruído

1.1 Código Fonte

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <driver/i2s.h>
#include <math.h>

// --- CONFIGURA  O WIFI ---
const char* ssid = "S23+ de Alberto";
const char* password = "ccsb2106";

// --- CONFIGURA  O ThingSpeak ---
const char* thingSpeakServer = "http://api.thingspeak.com
    /update";
const char* apiKey = "2DWI6UT6RF23HYAE";

// --- CONFIGURA  O DE CONEXAO COM MICROFONE
// --- I2S PINS ---
#define I2S_WS 15      // L/R clock
#define I2S_SD 32      // Serial data in
#define I2S_SCK 14     // Serial clock

#define SAMPLE_RATE 48000
#define SAMPLES 1024

#define MIC_REF_AMPL 0.00631 // 1 Pa em volts (calibrar)

float samples[SAMPLES];

// Filtro A-weighting (coeficientes para 48kHz)
class AWeightingFilter {
public:
    AWeightingFilter() {
        b0 = 0.25574113;
        b1 = -0.51148225;
        b2 = 0.25574113;
        a1 = -0.64718561;
        a2 = 0.14223739;

        x1 = x2 = y1 = y2 = 0;
    }

    float process(float x0) {
        float y0 = b0 * x0 + b1 * x1 + b2 * x2 - a1 * y1
            - a2 * y2;

        x2 = x1;
        x1 = x0;
        y2 = y1;
```

```
        y1 = y0;

        return y0;
    }

private:
    float b0, b1, b2, a1, a2;
    float x1, x2, y1, y2;
};

AWeightingFilter aFilter;

void setupI2S() {
  const i2s_config_t i2s_config = {
    .mode = i2s_mode_t(I2S_MODE_MASTER | I2S_MODE_RX),
    .sample_rate = SAMPLE_RATE,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = I2S_COMM_FORMAT_I2S,
    .intr_alloc_flags = 0,
    .dma_buf_count = 8,
    .dma_buf_len = 1024,
    .use_apll = true
  };

  const i2s_pin_config_t pin_config = {
    .bck_io_num = I2S_SCK,
    .ws_io_num = I2S_WS,
    .data_out_num = I2S_PIN_NO_CHANGE,
    .data_in_num = I2S_SD
  };

  i2s_driver_install(I2S_NUM_0, &i2s_config, 0, NULL);
  i2s_set_pin(I2S_NUM_0, &pin_config);
  i2s_set_clk(I2S_NUM_0, SAMPLE_RATE,
     I2S_BITS_PER_SAMPLE_32BIT, I2S_CHANNEL_MONO);
}

void connectWiFi() {
  Serial.print("Conectando␣em␣WiFi␣");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  int tentativas = 0;
  while (WiFi.status() != WL_CONNECTED && tentativas <
     20) {
    delay(500);
    Serial.print(".");
```

```
93      tentativas++;
94    }
95
96    if (WiFi.status() == WL_CONNECTED) {
97      Serial.println("\nWiFi conectado!");
98      Serial.print("IP: ");
99      Serial.println(WiFi.localIP());
100   } else {
101     Serial.println("\nFalha ao conectar WiFi.");
102   }
103 }
104
105 void setup() {
106   Serial.begin(115200);
107   setupI2S();
108   connectWiFi();
109 }
110
111 void loop() {
112   if (WiFi.status() != WL_CONNECTED) {
113     connectWiFi();
114   }
115
116   int32_t buffer[SAMPLES];
117   size_t bytesRead;
118
119   i2s_read(I2S_NUM_0, &buffer, SAMPLES * sizeof(int32_t),
          &bytesRead, portMAX_DELAY);
120
121   float sum = 0;
122   for (int i = 0; i < SAMPLES; i++) {
123     float sample = (float)buffer[i] / INT32_MAX;
124     float weighted = aFilter.process(sample);
125     samples[i] = weighted;
126     sum += weighted * weighted;
127   }
128
129   float rms = sqrt(sum / SAMPLES);
130   float dbA = 20.0 * log10(rms / MIC_REF_AMPL) + 94.0; //
        FATOR DE CORRE  O DO MICROFONE UTILIZADO
131
132   Serial.print("SPL (A-weighted): ");
133   Serial.print(dbA, 2);
134   Serial.println(" dB(A)");
135
136   if (WiFi.status() == WL_CONNECTED) {
137     HTTPClient http;
138
```

```
139     String url = String(thingSpeakServer) + "?api_key=" +
            apiKey + "&field1=" + String(dbA, 2);
140
141     http.begin(url);
142     int httpCode = http.GET();
143
144     if (httpCode > 0) {
145       Serial.printf("Dados enviados. C digo HTTP: %d\n",
              httpCode);
146     } else {
147       Serial.printf("Erro ao enviar dados: %s\n", http.
              errorToString(httpCode).c_str());
148     }
149     http.end();
150   } else {
151     Serial.println("WiFi desconectado. Tentando
          reconectar...");
152   }
153
154   delay(20000); // ThingSpeak limita para 1 atualiza  o
          a cada 15 segundos
155 }
```

Listing 1: Código implementado na Arduino IDE

2 Instruções de Uso

1. **Configuração Inicial**:
   - Clone ou baixe o código-fonte do projeto (arquivo ‘.ino‘) do repositório GitHub da dupla: pse2025 grupo-2.
   - Abra o código na Arduino IDE, certificando-se de ter o suporte à placa ESP32 instalado.
   - Instale as bibliotecas necessárias: ”WiFi.h”, ”HTTPClient.h”, ”driver/i2s.h” e ”math.h.
   - Configure suas credenciais de WiFi (SSID e senha) e os dados do seu canal ThingSpeak (Channel ID e Write API Key) no código.
   - Faça o upload do código para o microcontrolador ESP32.
2. **Operação**:
   - Alimente o ESP32. O sistema tentará se conectar à rede WiFi configurada.
   - Após a conexão WiFi, o ESP32 inicializará a interface I2S e começará a ler os dados do microfone INMP441.

- O sistema calculará periodicamente um valor representativo do nível sonoro (ex: RMS) a partir das amostras lidas.
- A cada intervalo definido no código (ex: 20 segundos), o ESP32 enviará o valor calculado para o campo configurado no seu canal ThingSpeak.
- Acesse seu canal no ThingSpeak para visualizar os dados sendo registrados e plotados em gráficos, como exemplificado na Figura 5.

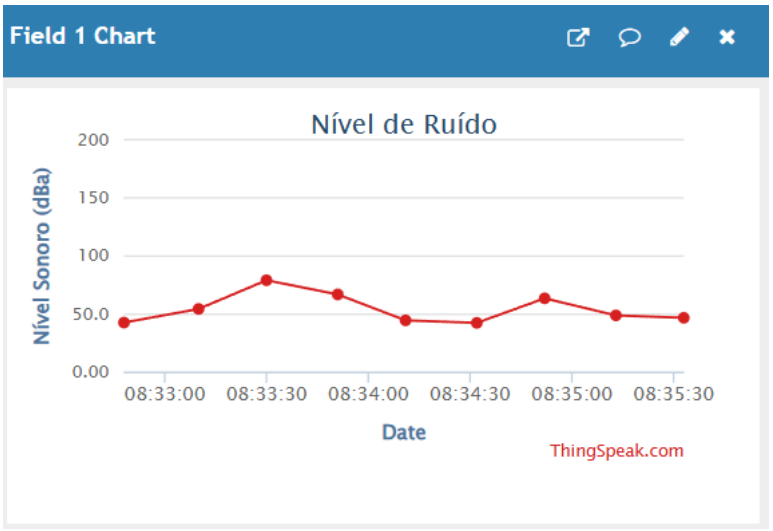

Figura 5: Exemplo de visualização dos dados no canal ThingSpeak

3 Conclusão

Conclui-se, portanto, que este trabalho contribuiu significativamente para o aprofundamento dos conhecimentos em engenharia, principalmente no que tange à disciplina de Projetos de Sistemas Embarcados. A implementação de um sistema IoT utilizando o ESP32 para interagir com um sensor I2S (INMP441), processar os dados localmente e transmiti-los via WiFi para uma plataforma na nuvem (ThingSpeak) foi substancial para a compreensão prática da integração entre hardware, software embarcado e serviços de internet. O projeto teve grande valia na formação de engenheiro e no entendimento de projetos de sistemas embarcados modernos, abordando conceitos de comunicação sem fio, protocolos de sensores digitais e plataformas de IoT.