

*ESTAÇÃO METEOROLÓGICA*

*CloudIA*

*comunicação*

Alunos - Bruno, Mathias, Vinícius e Pedro Augusto.

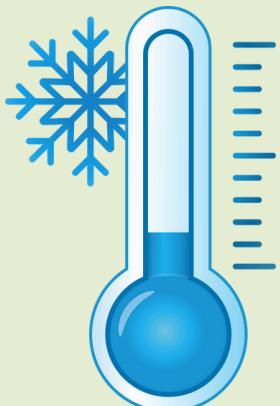
Professor - Carlos Henrique Barriquelo



# OBJETIVO GERAL:

Desenvolver e implementar o sistema de comunicação da estação meteorológica, garantindo a transmissão eficiente dos dados coletados para o banco de dados Firebase desenvolvido pelo outro grupo, utilizando ESP32 via comunicação LoRa.

LORA



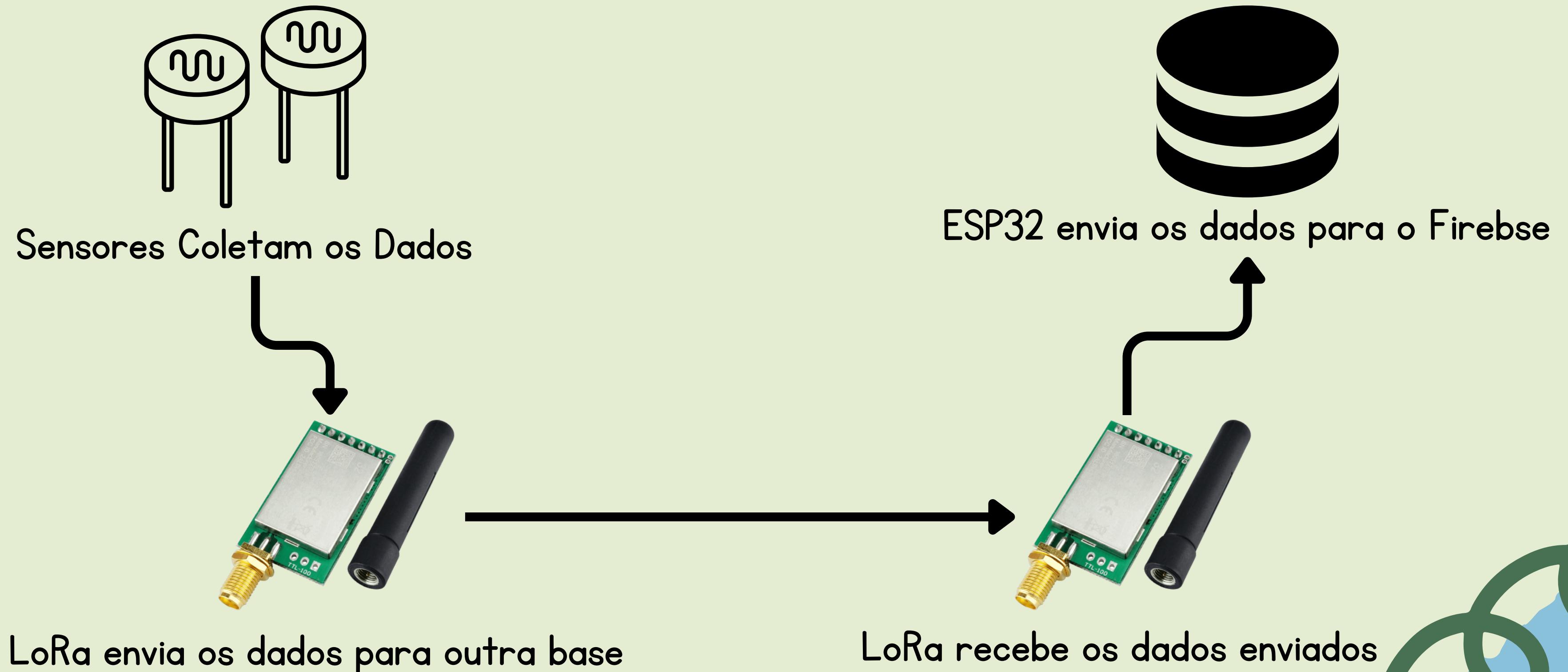
ESP32



# TECNOLOGIAS UTILIZADAS

- Microcontrolador ESP32 para comunicação via Wi-Fi com o servidor;
- Módulos LoRa para comunicação de longa distância entre a estação e a base de envio pro servidor;
- Protocolo de Comunicação HTTP para Wi-Fi;
- Banco de Dados: Firebase para armazenamento e análise de dados

# ARQUITETURA DO SISTEMA

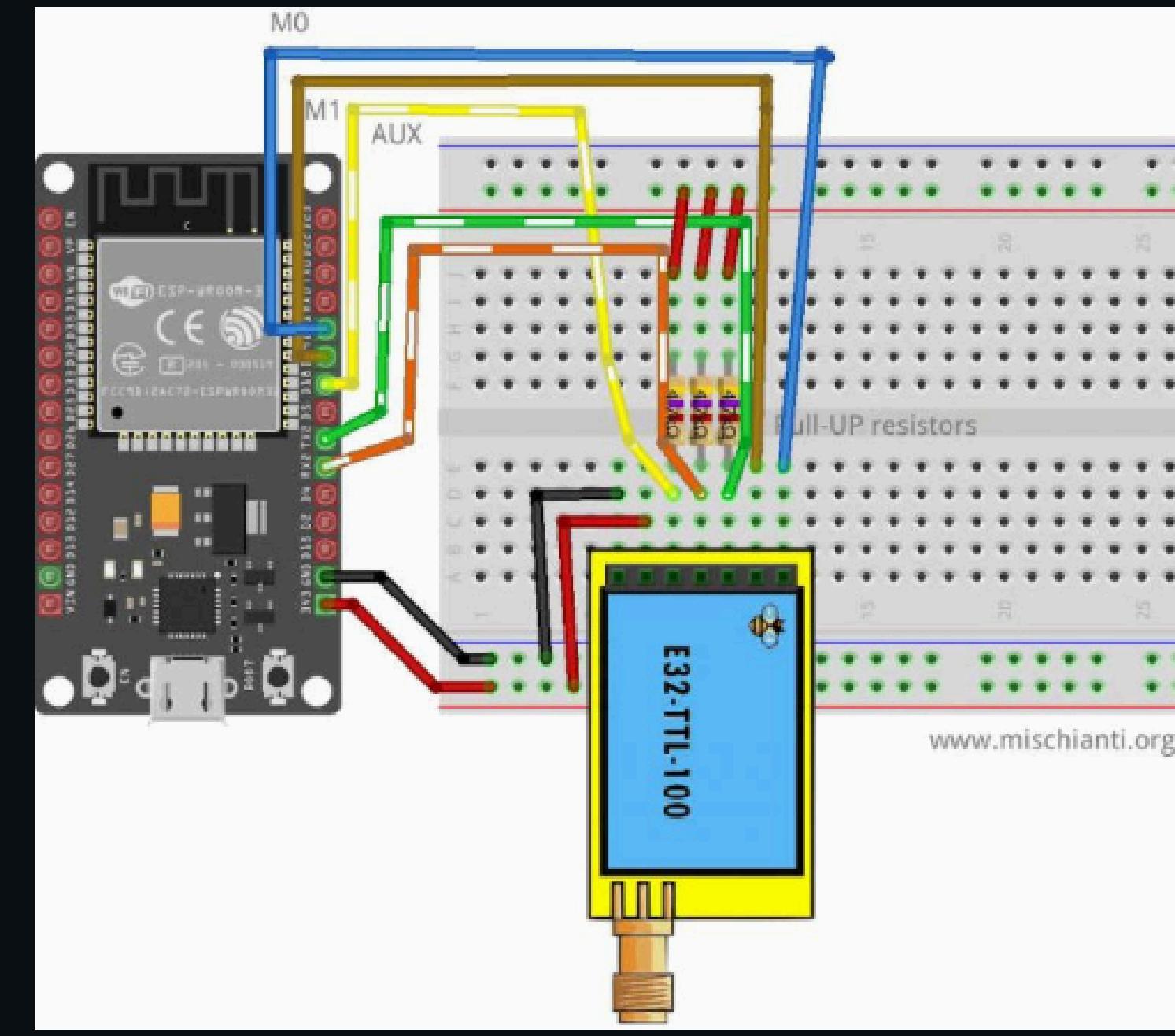


# ENVIO DOS DADOS

- Formatação dos dados coletados em pacotes JSON;
  - Inclusão de carimbos de tempo para sincronização;
- Comunicação com o servidor Firebase:
  - Uso de Firebase ESP32 CLient para envio de dados;
  - Validação de pacotes recebidos;

## esp32

Similar connection schema for esp32, but for RX and TX, we use RX2 and TX2 because, by default, esp32 doesn't have SoftwareSerial but has 3 Serial.



E22	esp32
M0	D21
M1	D19
TX	PIN RX2 (PullUP 4,7KΩ)
RX	PIN TX3 (PullUP 4,7KΩ)
AUX	PIN D18 (PullUP 4,7KΩ) (D15 to wake up)
VCC	5V (but work with less power in 3.3v)
GND	GND

[https://github.com/xreef/EByte\\_LoRa\\_E220\\_Series\\_Library?tab=readme-ov-file](https://github.com/xreef/EByte_LoRa_E220_Series_Library?tab=readme-ov-file)



*LINK DO VÍDEO*

<https://drive.google.com/file/d/1NMRpJQ5XkVNKbAByBrYlIYPvqY6l8L/view?usp=sharing>

Checkpoint 3  
09/05

ESTAÇÃO METEOROLÓGICA

CloudIA

comunicação



Alunos - Bruno, Mathias, Vinícius e Pedro Augusto.

Professor - Carlos Henrique Barriquelo

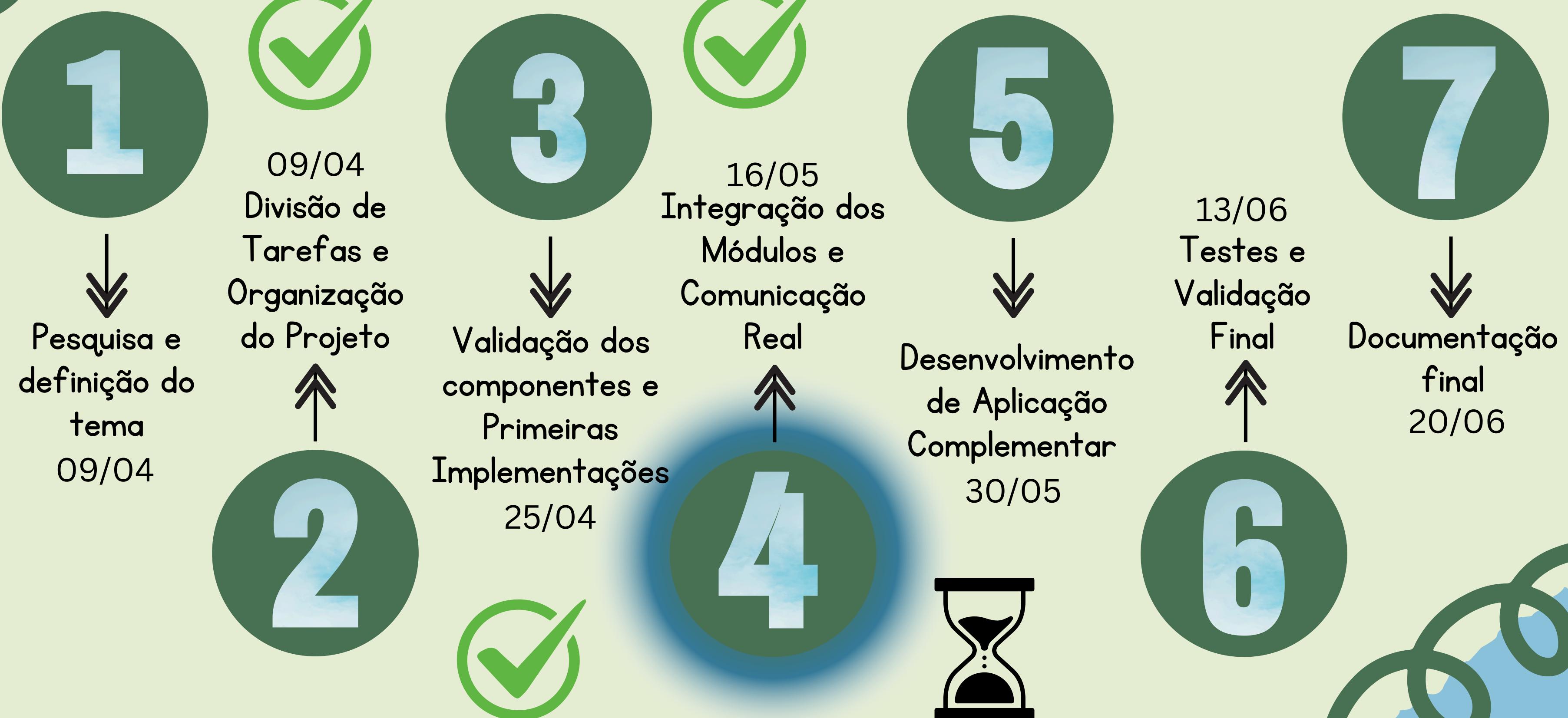
# MARCOS- DIA 09 / 04



# MARCOS- DIA 25/05



# MARCOS- DIA 09 / 05



# Projeto de um Repetidor LoRaWAN de Baixo Custo e Baixo Consumo de Energia

Lucas Maziero, Carlos Barriquello, Tiago B. Marchesan, Filipe G. Carloto, William. D. Vizzotto

Universidade Federal de Santa Maria (UFSM)

Santa Maria, Brasil

lucas.mazie.ro@hotmail.com

**Resumo**— Neste trabalho, é explanado experimentos e conceitos iniciais relacionados a construção de um repetidor LoRaWAN de baixo custo. Os experimentos iniciais são provas de conceito a fim de validar e verificar a viabilidade de inserção de um repetidor na rede LoRaWAN de comunicação. O experimento foi realizado com dispositivos de Classe A. É mostrado que adicionando um repetidor na rede LoRaWAN tem-se um aumento significativo na taxa de entrega de pacotes e também possibilitando o aumento de cobertura da rede, sem a necessidade de adicionar um novo gateway, assim, mitigando custos, aumentando a robustez e confiabilidade da rede LoRaWAN.

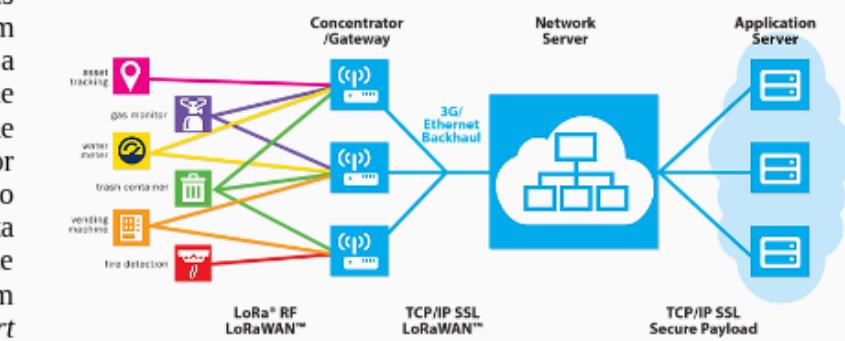
**Palavras chaves**— Baixo custo, LoRa, LoRaWAN, Repetidor, Gateway.

## I. INTRODUÇÃO

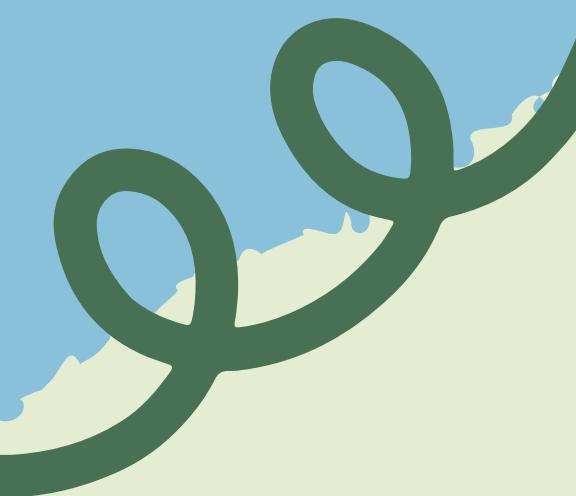
O protocolo LoRaWAN foi projetado para Internet das Coisas (IoT), onde os dispositivos geralmente necessitam de um baixo consumo de energia. O protocolo LoRaWAN utiliza a modulação LoRa, o que proporciona um baixo consumo de potência do dispositivo e longo alcance, chegando na ordem de quilômetros (km). Quando se fala em IoT a segurança é um fator indispensável, sendo que esta já vem embutida no protocolo LoRaWAN, ou seja, todos os dados são criptografados de ponta a ponta, assim assegurando a privacidade da informação que trafega pela rede LoRaWAN. O dispositivo final pode ser um sensor, um medidor de consumo de energia elétrica (*smart meter*), hidrômetros, *beacons* e muitos outros.

## B. LoRaWAN

O LoRaWAN define o protocolo de comunicação e a arquitetura de rede, enquanto a camada física LoRa permite o link de comunicação de longo alcance. O protocolo LoRaWAN também é responsável por gerenciar as frequências de comunicação, taxa de dados e energia para todos os dispositivos de forma individual [2]. Na rede LoRaWAN os dispositivos finais enviam os dados de forma assíncrona para qualquer gateway que esteja na sua área de cobertura. A topologia da rede se baseia num arranjo do tipo estrela, similar a uma rede Wi-Fi ou de telefonia celular. As informações enviadas pelo protocolo LoRaWAN são criptografadas (AES-128), de modo que somente a aplicação final pode ter acesso aos dados. A arquitetura da rede LoRaWAN é mostrada na Figura 1.



[https://www.ufsm.br/app/uploads/sites/553/2020/07/91663-field\\_submission\\_abstract\\_file2.pdf](https://www.ufsm.br/app/uploads/sites/553/2020/07/91663-field_submission_abstract_file2.pdf)



## RESUMO

### ESTUDO DO PROTOCOLO LORAWAN E IMPLEMENTAÇÃO DE UMA REDE PRIVADA COM NÓS DISTRIBUÍDOS

AUTOR: Matheus Henrique Baumgarten Rabuske

ORIENTADOR: Carlos Henrique Barriquello

O presente trabalho mostra um estudo geral realizado sobre o LoRaWAN, um protocolo de longo alcance e baixo custo energético desenvolvido para a Internet das Coisas. Serão abordadas suas características principais, diferenças para outros protocolos comumente utilizados na área, arquitetura da rede, funcionamento, segurança, uso legal no Brasil e partes de *software* e *hardware* que o implementam. No final do trabalho, será apresentada a implementação ponta-a-ponta de uma rede LoRaWAN e o desenvolvimento de uma aplicação cliente-servidor simples que utiliza a rede implementada para comunicação bidirecional entre dois ou mais dispositivos.

**Palavras-chave:** Protocolo. Internet. Coisas. Comunicação. Alcance. Economia. Energia. Bidirecional.

[https://www.ufsm.br/app/uploads/sites/429/2018/11/TCC\\_Matheus\\_Rabuske.pdf](https://www.ufsm.br/app/uploads/sites/429/2018/11/TCC_Matheus_Rabuske.pdf)



# PROTÓCOLO JSON

- Para comunicação com o LORA é necessário formatar o conjunto de dados em uma string.
- Para um conjunto grande de dados é recomendado utilizar um protocolo para facilitar essa comunicação e a codificação.



# PROTÓCOLO JSON

Arquivo JSON estruturado

```
{  
  "sensor": "gps",  
  "time": 1351824120,  
  "data": [48.756080, 2.302038]  
}
```



Arquivo JSON serializado

```
{"sensor": "gps", "time": 1351824120, "data": [48.75608, 2.302038]}
```



STRING

# BIBLIOTECA ARDUINOJSON

```
#include "ArduinoJson.h"

JsonDocument doc;
String temp = "";

void setup(){
    Serial.begin(115200);
}

void loop(){
    doc["sensor"] = "gps";
    doc["time"] = 1351824120;
    doc["data"][0] = 48.756080;
    doc["data"][1] = 2.302038;

    serializeJson(doc, coisa);
    Serial.println(temp);
}
```

Código para serialização no emissor

```
{"sensor":"gps","time":1351824120,"data":[48.75608,2.302038]}
{"sensor":"gps","time":1351824120,"data":[48.75608,2.302038]}
 {"sensor":"gps","time":1351824120,"data":[48.75608,2.302038]}
 {"sensor":"gps","time":1351824120,"data":[48.75608,2.302038]}
 {"sensor":"gps","time":1351824120,"data":[48.75608,2.302038]}
 {"sensor":"gps","time":1351824120,"data":[48.75608,2.302038]}
 {"sensor":"gps","time":1351824120,"data":[48.75608,2.302038]}
 {"sensor":"gps","time":1351824120,"data":[48.75608,2.302038]}
 {"sensor":"gps","time":1351824120,"data":[48.75608,2.302038]}
```

Monitor Serial

# BIBLIOTECA ARDUINOJSON

```
#include "ArduinoJson.h"

JsonDocument doc;
JsonDocument doc2;
String temp = "";

void setup(){
  Serial.begin(115200);
}

void loop(){
  doc["sensor"] = "gps";
  doc["time"] = 1351824120;
  doc["data"][0] = 48.756080;
  doc["data"][1] = 2.302038;

  serializeJson(doc, coisa);
  deserializeJson(doc2, coisa);
  coisa2 = doc["time"];
  Serial.println(coisa2);
}
```

1351824120  
1351824120  
1351824120  
1351824120  
1351824120  
1351824120  
1351824120  
1351824120

Monitor Serial

Código para desserialização no receptor

Checkpoint 7  
11/07

ESTAÇÃO METEOROLÓGICA

CloudIA

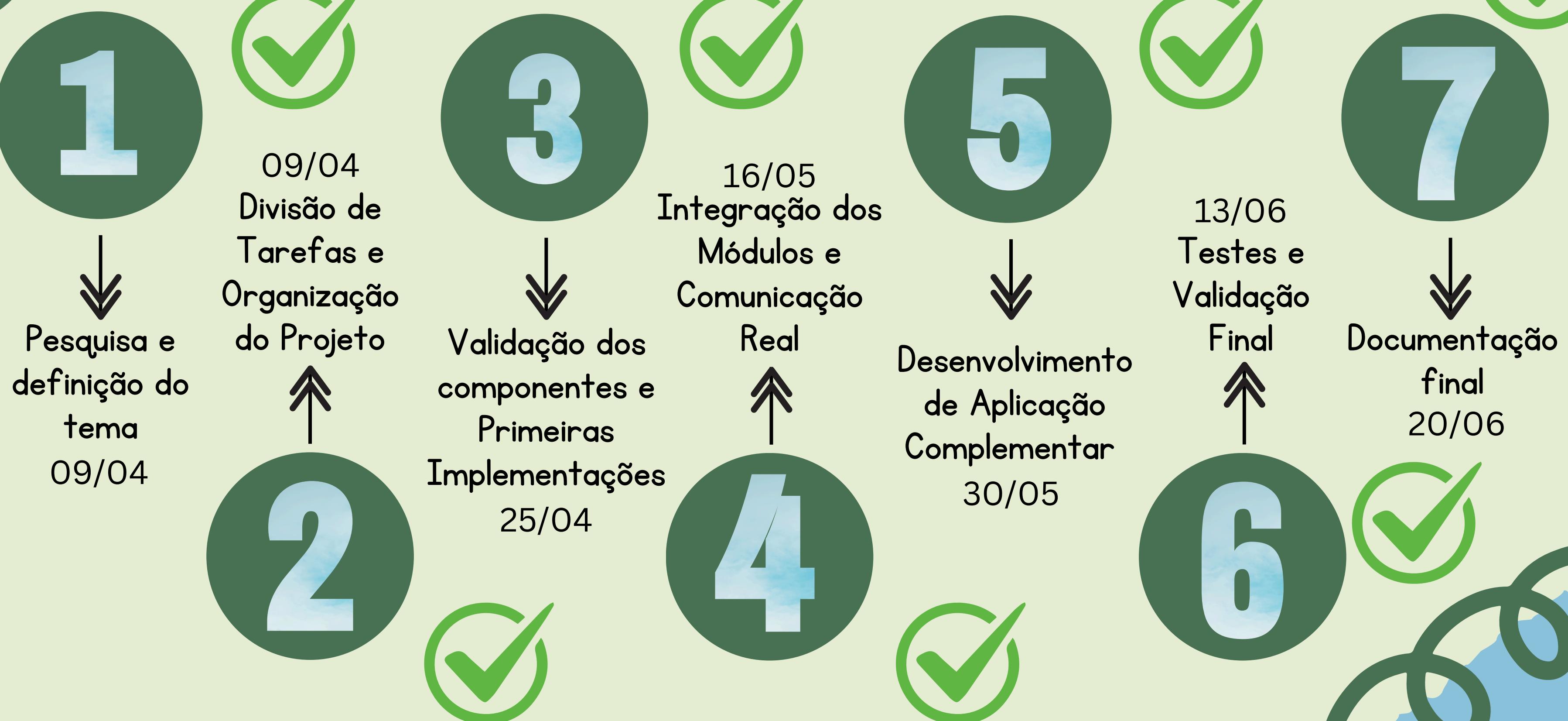
comunicação



Alunos - Bruno, Mathias, Vinícius e Pedro Augusto.

Professor - Carlos Henrique Barriquelo

# MARCOS - DIA 11/07



# O QUE FOI FEITO

Validação dos componentes 

Entendimento sobre a comunicação 

Simulação de dados 

Modelo de caixa 3D para o gateway 

Impressão de caixa 3D para o gateway 

Comunicação LoRa 

Utilização de pacotes json 

# O QUE NÃO FOI FEITO

- Sugestões de trabalhos futuros para a próxima turma

1. Utilização do Lorawan 

2. Criptografia dos dados para uma comunicação mais segura 

3. Técnicas low-power 

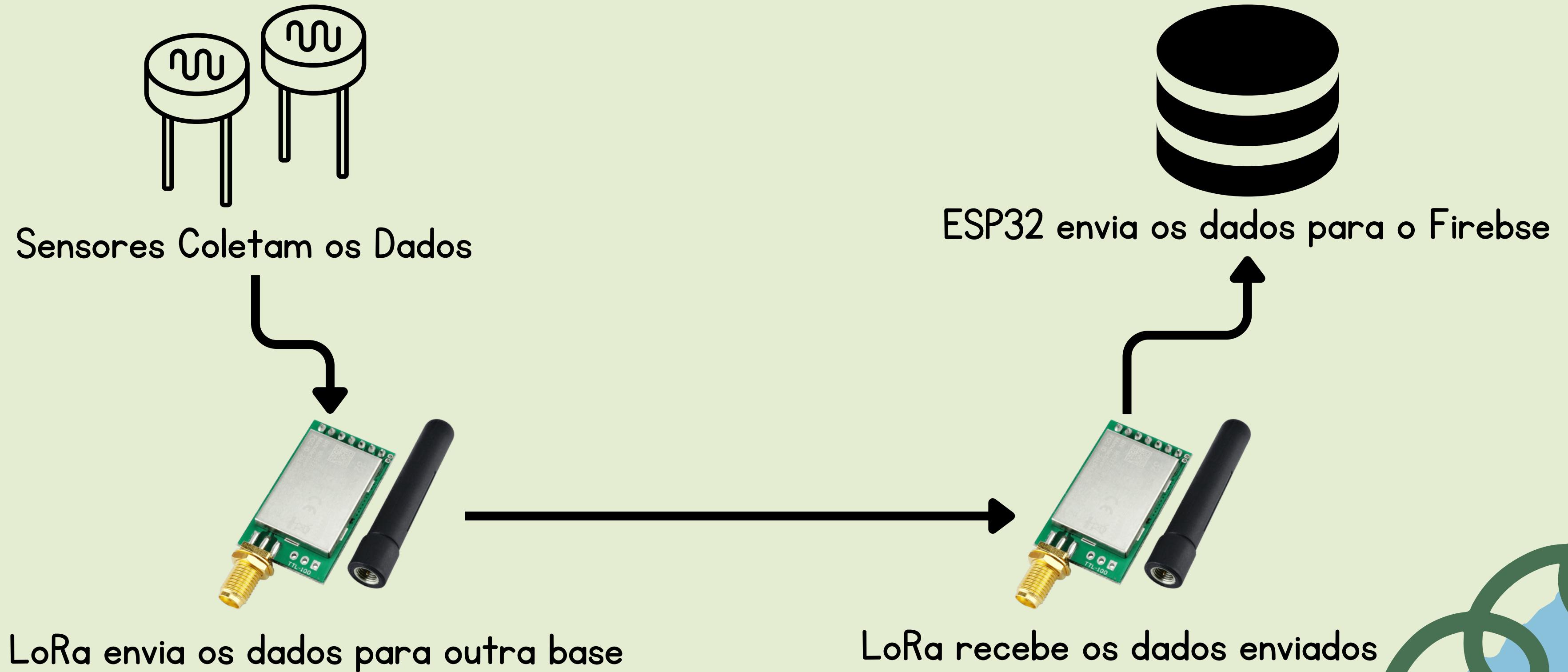


O QUE FOI FEITO

# MODELO DE CAIXA 3D PARA O GATEWAY

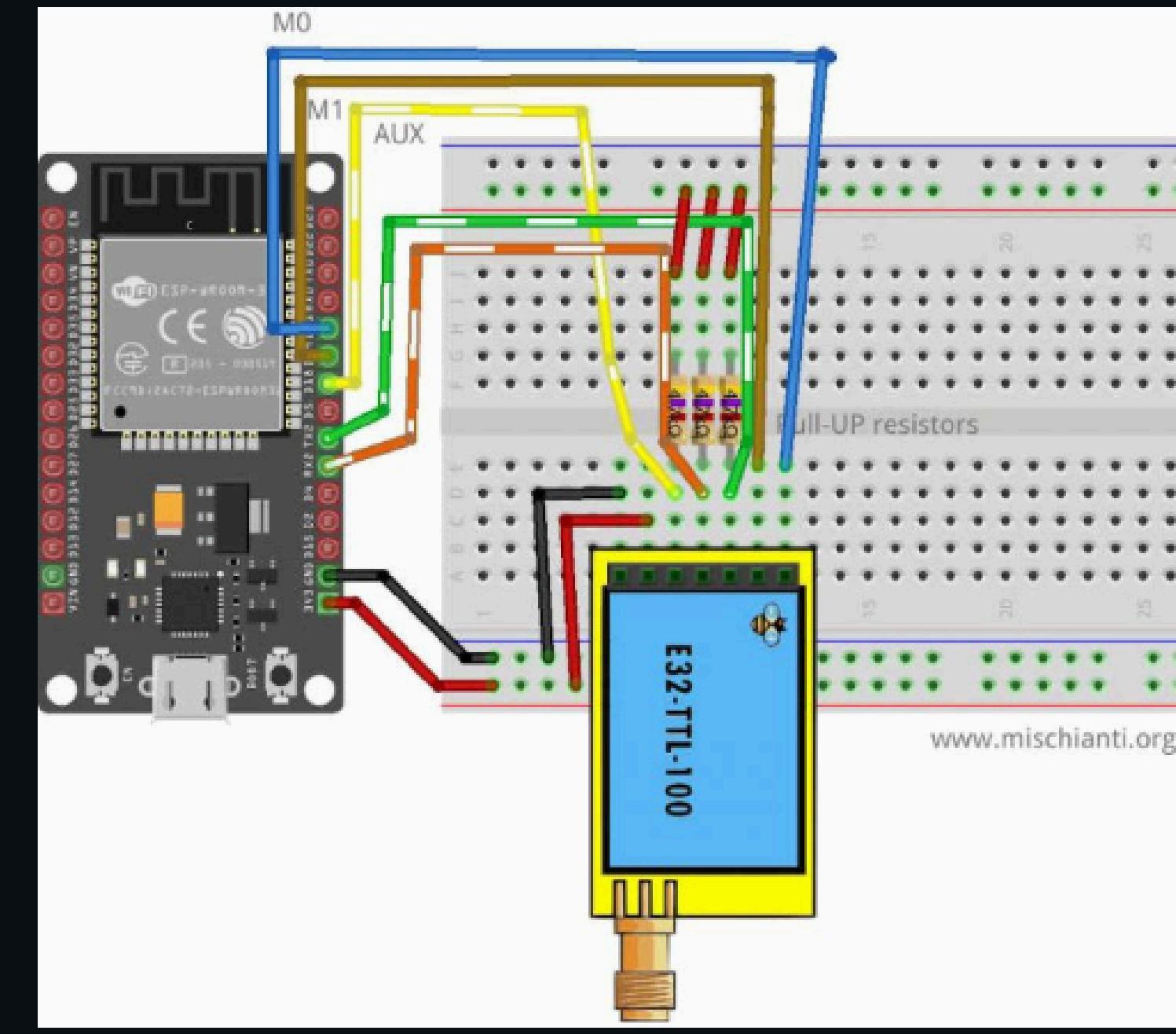


# ARQUITETURA DO SISTEMA



## esp32

Similar connection schema for esp32, but for RX and TX, we use RX2 and TX2 because, by default, esp32 doesn't have SoftwareSerial but has 3 Serial.



E22	esp32
M0	D21
M1	D19
TX	PIN RX2 (PullUP 4,7KΩ)
RX	PIN TX3 (PullUP 4,7KΩ)
AUX	PIN D18 (PullUP 4,7KΩ) (D15 to wake up)
VCC	5V (but work with less power in 3.3v)
GND	GND

[https://github.com/xreef/EByte\\_LoRa\\_E220\\_Series\\_Library?tab=readme-ov-file](https://github.com/xreef/EByte_LoRa_E220_Series_Library?tab=readme-ov-file)

# ENVIO DOS DADOS

- Formatação dos dados coletados em pacotes JSON;
  - Inclusão de carimbos de tempo para sincronização;
- Comunicação com o servidor Firebase:
  - Uso de Firebase ESP32 CLient para envio de dados;
  - Validação de pacotes recebidos;

# SIMULAÇÃO DE DADOS

```
umidade_ar = random(200, 1000) / 10.0;           // 20.0 a 100.0 %
porcentagem_umidade_solo = random(100, 1000) / 10.0; // 10.0 a 100.0 %
temperatura_ar = random(150, 450) / 10.0;          // 15.0 a 45.0 °C
temperatura_solo = random(100, 400) / 10.0;          // 10.0 a 40.0 °C
kmh = random(0, 1000) / 10.0;                      // 0.0 a 100.0 km/h
chuva_mm = random(0, 200);                          // 0 a 200 mm
direcao = direcoes[random(0, 8)];
```

# UTILIZAÇÃO DE PACOTES JSON

```
// Formata os dados dos sensores em um json serializado e envia pelo LoRa
void envia_dados_lora(){
    // Atribui ao objeto json os dados dos sensores
    sensors["ua"] = umidade_ar;
    sensors["us"] = porcentagem_umidade_solo;
    sensors["ta"] = temperatura_ar;
    sensors["ts"] = temperatura_solo;
    sensors["vv"] = kmh;
    sensors["cm"] = chuva_mm;
    sensors["di"] = direcao;

    // Serializa o json
    serializeJson(sensors, message_json);
    // Envia o json serializado pelo LoRa
    LoRaSerial.println(message_json);
    Serial.println("Enviado: " + message_json);
}
```

# INICIALIZAÇÃO LORA

```
void setup() {  
  
    randomSeed(analogRead(0));  
    Serial.begin(115200);  
  
    // Pinos do LoRa  
    pinMode(PIN_M0, OUTPUT);  
    pinMode(PIN_M1, OUTPUT);  
    pinMode(PIN_AUX, INPUT);  
  
    // Inicializa o LoRa  
    LoRaSerial.begin(9600, SERIAL_8N1, 16, 17);  
    Serial.println("Iniciando TRANSMISSOR LoRa...");  
    setLoRaMode(LOW, LOW); // Modo 0 - Transparente  
    Serial.println("Modo Transparente - Pronto para transmitir.");  
  
    // Salva o tempo em que terminou o setup para começar a contagem do tempo de envio  
    send_time = millis();  
}
```

# comunicação LORA

```
void loop() {  
  
    // Temporizador para envio de dados  
    if (millis() - send_time >= DELAY_MIN_SEND*1000*SIZE_MIN){  
        // Envia os dados e reinicia o temporizador  
        envia_dados_lora();  
        send_time = millis();
```

# ENVIO PARA A NUVM

```
// ===== CONFIG WIFI =====  
  
#define WIFI_SSID "████████"  
#define WIFI_PASSWORD "████████"  
  
// ===== CONFIG FIREBASE =====  
  
#define API_KEY "AIzaSyBZe1S0j53PZxSHDf9LQ3MIqPREXy8t-jg"  
#define DATABASE_URL "https://cloudia-22e4d-default-rtdb.firebaseio.com/"  
#define USER_EMAIL "cloudia.firebaseio@gmail.com"  
#define USER_PASSWORD "████████"
```

# ENVIAR DADOS PARA A NUVEM

```
// ====== Firebase ======
config.api_key = API_KEY;
config.database_url = DATABASE_URL;
auth.user.email = USER_EMAIL;
auth.user.password = USER_PASSWORD;

Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);

// Esperar Firebase estar pronto
Serial.println("Conectando ao Firebase...");
unsigned long timeout = millis() + 10000;
while (!Firebase.ready()) {
    if (millis() > timeout) {
        Serial.println("Falha ao conectar com Firebase.");
        break;
    }
    delay(100);
}

if (Firebase.ready()) {
    Serial.println("Firebase pronto!");
}
```