Diogo Fernandes, Rafael Rio, Ricardo Fortuna

Nº22102558, Nº22100521, Nº22107919

Introdução **3**
RFID **4**
Baixa Frequência (LF): 4
Alta Frequência (HF): 4
Ultra Alta Frequência (UHF): 4
Tag RFID 5
Tags Passivas 5
Tags Semi-Passivas 5
Sistema de Processamento de Dados 5
Funcionamento do RFID 5
Aplicabilidade RFID 5
Vantagens e Desvantagens 6
LoraWan **7**
Arquitetura do LoRaWAN 7
Funcionamento do LoRaWAN 7
Vantagens do LoRaWAN 7
Desafios do LoRaWAN 8
Aplicações Práticas do LoRaWAN 8
Arduino Uno 8
DIAGRAMA ESQUEMÁTICO/ ESQUEMA ELÉTRICO **9**
1º Fase **10**
Receptor: 14
2º Fase **15**
Emissor(LoraWan): 15
3º Fase(Telegram) **16**
Desafios e Obstáculos **17**
Conclusão **19**
WebGrafia **20**

Introdução

Este projeto foi elaborado no âmbito da disciplina de Componentes Distribuídos com o propósito de investigar a tecnologia de RFID. Consequentemente, houve a necessidade de dividir o trabalho em três partes distintas.

A primeira fase consiste no desenvolvimento de uma comunicação entre 2 microcontroladores usando a tecnologia LoRaWAN, foi nos atribuído o módulo REYAX RYLRD890 e REYAX RYLRD990.

A segunda etapa consiste na implementação do Módulo Leitor RFID RC522 onde encaramos o trabalho como um desafio para “replicar” um projeto de controle de acesso e sistemas de segurança.

A terceira fase foi concebida para explorar a comunicação entre microcontroladores e o aplicativo Telegram devido à sua versatilidade em dispositivos móveis e computadores. Portanto, decidimos criar um bot de alertas no Telegram para este propósito.

RFID

A tecnologia RFID (Radio Frequency Identification, ou Identificação por Rádio Frequência) é um sistema de identificação automática que utiliza ondas de rádio para capturar e armazenar dados remotamente. Esse sistema permite a leitura de informações armazenadas em dispositivos conhecidos como tags ou etiquetas RFID, sem a necessidade de contato físico direto ou linha de visão entre o leitor e a etiqueta.

As frequências mais utilizadas em sistemas RFID podem ser divididas em três categorias principais, cada uma com suas características e aplicações específicas:

Baixa Frequência (LF):

Frequência: 30 kHz a 300 kHz (geralmente 125 kHz ou 134.2 kHz)
Alcance de Leitura: Até 10 cm
Características: Boa penetração em materiais não metálicos, menos suscetível a interferências, adequada para ambientes com metais e líquidos.
Aplicações: Identificação de animais, controlo de acesso, rastreamento de ativos em condições adversas.

Alta Frequência (HF):

Frequência: 3 MHz a 30 MHz (geralmente 13.56 MHz)
Alcance de Leitura: Até 1 metro
Características: Boa capacidade de transmissão de dados, menos afetada por metais do que UHF, compatível com NFC (Near Field Communication).
Aplicações: Cartões de transporte público, passaportes eletrónicos, bilhetes de eventos, pagamento sem contato (contactless payment).

Ultra Alta Frequência (UHF):

Frequência: 300 MHz a 3 GHz (geralmente 860 MHz a 960 MHz)
Alcance de Leitura: Vários metros (até 12 metros ou mais, dependendo do ambiente e da potência do leitor)
Características: Maior alcance de leitura, rápida taxa de leitura, mais suscetível a interferências de metais e líquidos.
Aplicações: Gestão de inventário e cadeia de abastecimento, logística, rastreamento de veículos, controle de acesso em larga escala.
Cada uma dessas faixas de frequência oferece vantagens específicas que tornam a tecnologia RFID adaptável a uma ampla variedade de necessidades e ambientes operacionais. A escolha da frequência adequada depende das exigências particulares da aplicação, incluindo o alcance necessário, o tipo de material do objeto a ser etiquetado e o ambiente em que o sistema será utilizado.
Um sistema RFID é composto por três componentes principais:

Tag RFID

Também chamada de transponder, a tag é um microchip acoplado a uma antena. O microchip armazena os dados e a antena permite a comunicação com o leitor RFID. As tags podem ser passivas, ativas ou semi-passivas:

Tags Passivas

Não possuem fonte de energia interna e são ativadas pela energia do sinal do leitor. Tags Ativas: Possuem uma fonte de energia interna (bateria), permitindo uma maior distância de leitura.

Tags Semi-Passivas

Possuem uma bateria interna para alimentar o microchip, mas utilizam a energia do sinal do leitor para a comunicação.
Leitor RFID: Também conhecido como interrogador, o leitor emite ondas de rádio que ativam a tag e capturam os dados armazenados nela. O leitor pode ser fixo ou portátil e está conectado a um sistema de processamento de dados.

Sistema de Processamento de Dados

Inclui o software e a infraestrutura de TI que processam e armazenam os dados capturados pelo leitor. Esses dados podem ser integrados a sistemas de gerenciamento de inventário, controle de acesso, rastreamento de ativos, entre outros.

Funcionamento do RFID

O funcionamento do RFID baseia-se na emissão e recepção de sinais de rádio. Quando o leitor RFID emite um sinal, ele cria um campo eletromagnético que ativa a tag RFID. A tag, ao ser ativada, transmite os dados armazenados no microchip de volta ao leitor. O leitor, por sua vez, decodifica essa informação e a envia para o sistema de processamento de dados.

Aplicabilidade RFID

A tecnologia RFID tem uma ampla gama de aplicações em diversos setores, incluindo:

Logística e Cadeia de Suprimentos: Rastreamento e gerenciamento de inventário em tempo real, melhorando a eficiência e reduzindo perdas.
Varejo: Controle de stock, prevenção de furtos e aprimoramento da experiência do cliente através de checkouts automatizados.
Saúde: Rastreamento de equipamentos médicos, monitorização de pacientes e gestão de medicamentos.
Transporte e Logística: Gestão de frotas, rastreamento de pacotes e controle de acesso em transportes públicos.
Segurança: Controle de acesso a áreas restritas e autenticação de produtos.

Vantagens e Desvantagens

Vantagens:

Eficiência: Automação de processos reduz a necessidade de intervenção manual.
Precisão: Melhora a precisão do inventário e rastreamento de ativos.
Velocidade: Leitura rápida e simultânea de múltiplas tags.
Durabilidade: Tags RFID podem ser mais duráveis que códigos de barras em ambientes adversos.
Desvantagens:

Custo: Implementação inicial pode ser cara, especialmente para sistemas ativos.
Interferência: Metais e líquidos podem interferir na leitura dos sinais de rádio.
Privacidade: Preocupações com a segurança e privacidade dos dados armazenados nas tags.

LoraWan

O termo LoRaWAN advém da junção de LoRa, da expressão em inglês “Long Range” (longo alcance) com WAN (Wide Area Network – rede de longo alcance).

LoRaWAN é um protocolo de rede de longa distância e baixa potência desenvolvido para conectar dispositivos IoT de maneira eficiente. Ele utiliza a modulação LoRa (Long Range) para permitir comunicação de longo alcance com baixo consumo de energia. A combinação dessas características torna o LoRaWAN ideal para aplicações onde a duração da bateria e a cobertura são críticas.

Arquitetura do LoRaWAN

A arquitetura do LoRaWAN é composta por quatro componentes principais:

Dispositivos Finais (End Devices): São os sensores que recolhem e transmitem dados, como por exemplo, sensores de temperatura, humidade, movimento, entre outros.

Gateways: Atuam como intermediários que recebem sinais de rádio dos dispositivos finais e os encaminham para o servidor de rede. Funcionam como pontes entre a rede LoRa e a Internet.

Servidor de Rede (Network Server): Responsável por gerir a rede, processar os dados recebidos dos gateways e garantir a segurança da comunicação. Também coordena a comunicação entre os dispositivos finais e as aplicações.

Servidor de Aplicação (Application Server): Recebe os dados processados pelo servidor de rede e disponibiliza-os para as aplicações finais. Pode ser uma plataforma de monitorização ou um sistema de controlo automatizado.

Funcionamento do LoRaWAN

Transmissão de Dados: Os dispositivos finais transmitem dados usando a modulação LoRa.

Recepção pelos Gateways: Os sinais são captados pelos gateways, que possuem uma ampla cobertura geográfica.

Encaminhamento para o Servidor de Rede: Os gateways encaminham os dados para o servidor de rede por meio de uma conexão IP (Internet Protocol).

Processamento e Segurança: O servidor de rede processa os dados, executa a autenticação dos dispositivos e aplica criptografia para garantir a segurança.

Entrega ao Servidor de Aplicação: Os dados processados são então enviados para o servidor de aplicação, onde podem ser usados para diferentes finalidades, como monitorização ou controlo.

Vantagens do LoRaWAN

Longo Alcance: LoRaWAN pode cobrir distâncias de até 15 km em áreas rurais e 5 km em áreas urbanas.

Baixo Consumo de Energia: Dispositivos LoRaWAN podem operar por anos com uma única bateria, graças ao baixo consumo de energia.

Alta Capacidade de Dispositivos: Suporta um grande número de dispositivos conectados numa mesma rede.

Segurança: Implementa criptografia de ponta a ponta e autenticação de dispositivos, garantindo a segurança dos dados transmitidos.

Escalabilidade: Pode ser facilmente escalado para suportar novos dispositivos e aumentar a cobertura da rede.

Desafios do LoRaWAN

Largura de Banda Limitada: A tecnologia é adequada para dados de baixa taxa de transmissão, não sendo ideal para aplicações que exigem alta largura de banda.
Interferência: Em áreas densamente povoadas, a interferência pode afetar a qualidade da comunicação.

Regulamentação: A frequência utilizada pelo LoRaWAN pode estar sujeita a regulamentações locais, variando de acordo com a região.

Aplicações Práticas do LoRaWAN

Monitorização Ambiental: Sensores de qualidade do ar, temperatura e humidade em áreas urbanas e rurais.

Agricultura Inteligente: Monitorização do solo, irrigação automatizada e rastreamento de gado.

Cidades Inteligentes: Gestão de iluminação pública, recolha de lixo e monitorização de tráfego.

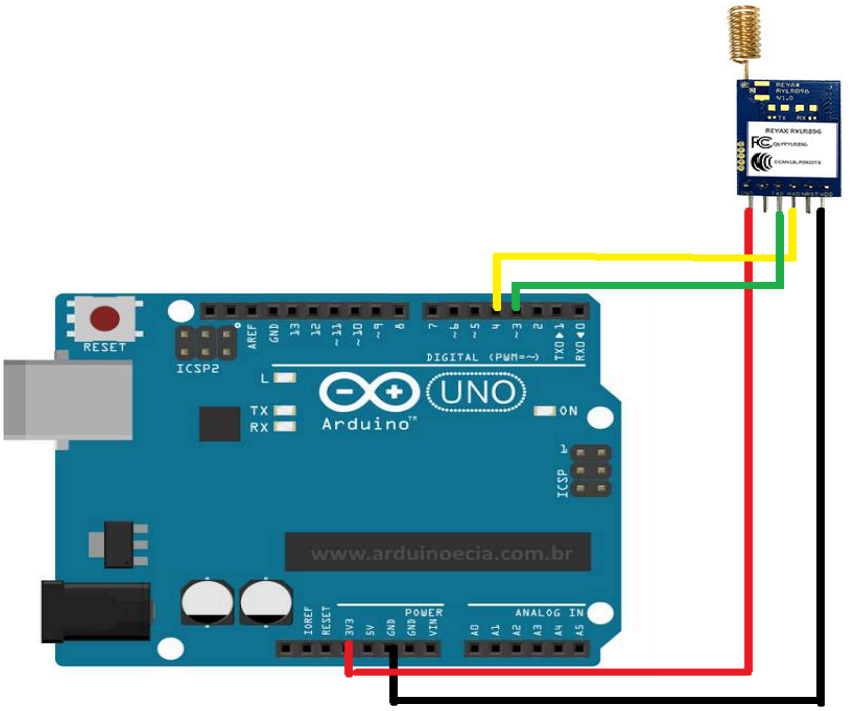
Medidores Inteligentes: Medição remota de água, gás e eletricidade.

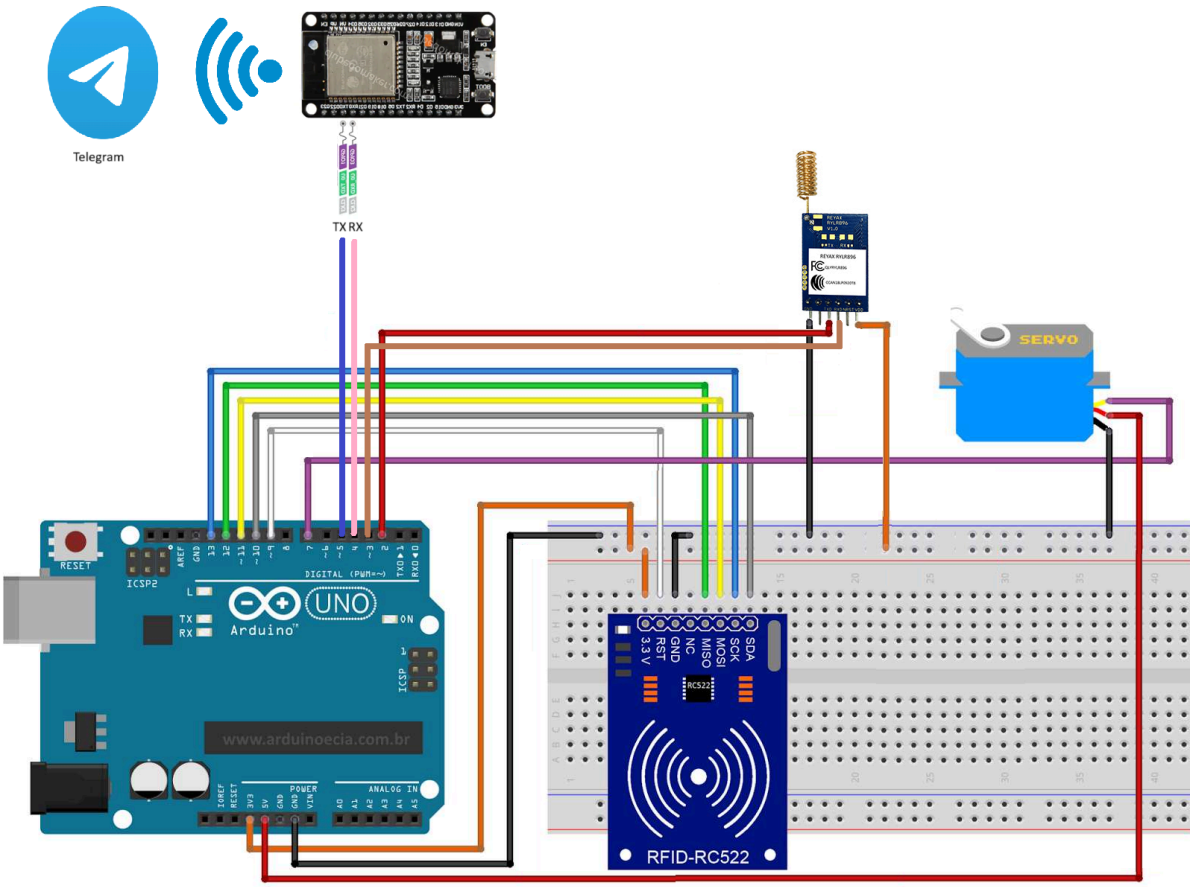
Rastreamento de Ativos: Localização e monitorização de bens móveis, como contentores e veículos.

Arduino Uno

“Arduino é uma plataforma open-source de prototipagem eletrónica com hardware e software flexíveis e fáceis de usar, destinado a artistas, designers, hobbistas e qualquer pessoa interessada em criar objetos ou ambientes interativos.” fonte - https://www.arduino.cc/

DIAGRAMA ESQUEMÁTICO/ ESQUEMA ELÉTRICO





1º Fase

A primeira fase consiste no desenvolvimento de uma comunicação entre 2 microcontroladores usando a tecnologia LoRaWAN, iniciamos o projeto com o modelo REYAX RYLRD890, mas ao longo do projeto decidimos testar o modelo REYAX RYLRD990. Por serem da mesma família os módulos reagem de uma maneira semelhante.

Começamos o projecto com o esp32 wrover e o esp32 wroom, no entanto não tivemos sucesso no desenvolvimento com estes microcontroladores, infelizmente não conseguimos identificar a razão de não funcionar com os ESP's32 e decidimos utilizar o arduino UNO, desconfiamos que tenha sido pela sua arquitetura diferenciada e mais simples que acabamos por ultrapassar o problema.

Importamos a biblioteca do softwareSerial para podermos "simular" uma comunicação de TX e RX virtual, ao usarmos este método, previne-nos problemas de comunicação com os pinos TX e RX originais, pois não nos é permitido o upload do programa através da porta USB se o pino TX e RX dos microcontroladores estejam conectados.

```
#include <SoftwareSerial.h>

//Portas para configurar SoftwareSerial
#define txLoraPin 5  // Pino onde o Tx do Sensor está conectado
#define rxLoraPin 6  // Pino onde o Rx do Sensor está conectado
```

Ao consultar a documentação dos dois módulos que é disponibilizado no site da própria marca do REYAX, percebemos que existem parâmetros dos módulos que são gravados em memória flash com o IPR, BAND, ADRESS e NETWORKID. No entanto foi decidido que essas configurações iam ser sempre definidas ao ligar a placa, porque deparamo-nos com situações que o módulo era resetar a memória devido a problemas de conexão dos cabos.

```
//Parametros para configurar o Modulo Lora
#define IPR 115200
#define BAND 870000000  // Frequência do rádio LoRa
#define NETWORKID 18
#define ADDRESS 2
#define NETWORKIDSender 1

#define SPREADING_FACTOR 9  // Spreading Factor do rádio LoRa
#define BANDWIDTH 7         // Bandwidth do rádio LoRa (125KHz)
#define CODING_RATE 1       // Coding Rate do rádio LoRa
#define PREAMBLE 12         // Preamble Length do rádio LoRa
```

De seguida fizemos a configuração dos módulos com o seguinte código:

```
// Método que inicializa o módulo LoRa
void setupLoRaRYLR896() {
  Serial.print("Começar Configuração:\r\n");
  delay(tempoDelay);

  lora.print("AT\r\n");
  delay(tempoDelay);
  Serial.print("AT:" + lora.readString());

  lora.print("AT+IPR=" + String(IPR) + "\r\n");
  delay(tempoDelay);
  Serial.print("IPR:" + lora.readString());

  lora.print("AT+BAND=" + String(BAND) + "\r\n");  //Bandwidth set to 868.5MHz
  delay(tempoDelay);
  Serial.print("BAND:" + lora.readString());

  lora.print("AT+ADDRESS=" + String(ADDRESS) + "\r\n");  //needs to be unique
  delay(tempoDelay);
  Serial.print("ADDRESS:" + lora.readString());

  lora.print("AT+NETWORKID=" + String(NETWORKID) + "\r\n");  //needs to be same for receiver and transmitter
  delay(tempoDelay);
  Serial.print("NETWORKID:" + lora.readString());

  lora.print("AT+PARAMETER=" + String(SPREADING_FACTOR) + "," + String(BANDWIDTH) + "," + String(CODING_RATE) + "," + String(PREAMBLE) + "\r\n");
  delay(tempoDelay);
  Serial.print("PARAMETER:" + lora.readString());
  delay(tempoDelay);
}
```

É importante relembrar que deve-se ler a documentação dos módulos, pois os módulos têm a necessidade de ter comandos “\r\n” ou println para simular esses mesmos comandos, caso contrário os comandos enviados não funcionam.
Uma vez que enviamos os códigos no Serial do lora, usamos o Serial do ecrã para mostrar se deu algum erro ou foram recebidos:

```
AT:+OK
BAND:+OK
ADDRESS:+OK
NETWORKID:+OK
PARAMETER:+OK
```

Caso os módulos retornem algum erro, podemos consultar na documentação do módulo adquirido no próprio site o significado de cada erro.

Uma vez que a configuração foi feita, decidimos confirmar as configurações visualmente e deparamos que os módulos ao comunicarem com o Arduino trazia ruído e apareciam caracteres estranhos.

```
void showSetupLoRaRYLR896() {
  Serial.print("\r\n");
  Serial.print("Confirmar Configuração:\r\n");
  delay(tempoDelay);

  lora.print("AT\r\n");
  Serial.print("AT:" + lora.readString());
  delay(tempoDelay);


  lora.print("AT+IPR?\r\n");
  Serial.print("IPR:" + lora.readString());
  delay(tempoDelay);


  lora.print("AT+BAND?\r\n");
  Serial.print("BAND:" + lora.readString());
  delay(tempoDelay);

  lora.print("AT+PARAMETER?\r\n");
  Serial.print("PARAMETER:" + lora.readString());
  delay(tempoDelay);

  lora.print("AT+NETWORKID?\r\n");
  Serial.print("NETWORKID:" + lora.readString());
  delay(tempoDelay);

  lora.print("AT+ADDRESS?\r\n");  //needs to be unique
  Serial.print("ADDRESS:" + lora.readString());
  delay(tempoDelay);
}
```

```
Confirmar Configuração:
AT:+OK
IPR:+IPR=11520`
BAND:+BAND=870000000
PARAMETER:+PP□PSQU□□ON□□)b□□j
NETWORKID:+N□TWORKID=18□ADDRESS:+ADDRESS=2
```

Uma vez os módulos configurados e validados, foi criado uma função de “escuta” para que o módulo esteja sempre apto a receber informação. A informação é recebida sempre com uma estrutura como demonstra a imagem a seguir:

+RCV Show the received data actively.

Syntax	Response
+RCV=<Address>,<Length>,<Data>,<RSSI>,<SNR> <Address> Transmitter Address ID <Length> Data Length <Data> ASCII Format Data <RSSI> Received Signal Strength Indicator <SNR> Signal-to-noise ratio	
Example: Module received the ID Address 50 send 5 bytes data, Content is HELLO string, RSSI is -99dBm, SNR is 40, It will show as below. +RCV=50, 5, HELLO, -99, 40	

O código a seguir é uma função para validar se está para receber alguma comunicação, caso exista uma comunicação, o valor dessa comunicação é guardada numa variável em formato de String.

```
if (lora.available()) {

  String received = lora.readString();
```

De seguida o código começa por filtrar as mensagens para que sejam apenas mensagens recebidas que começam sempre por +RCV=, depois começamos por trabalhar a mensagem de forma a conseguir obter o conteúdo da mensagem, pois temos que obter o tamanho da string que é passado como argumento dentro de vírgulas, depois de obter o tamanho da string fazemos o substring a partir do index da depois da vírgula até o index da vírgula + o tamanho da mensagem.

```
if (received.indexOf("+RCV") >= 0) {
  int delimiter, delimiter_1, delimiter_2, delimiter_3;
  delimiter = received.indexOf(",");
  delimiter_1 = received.indexOf(",", delimiter + 1);
  delimiter_2 = received.indexOf(",", delimiter_1 + 1);
  delimiter_3 = received.indexOf(",", delimiter_2 + 1);
  int lengthMessage = received.substring(delimiter + 1, delimiter_1).toInt();

  String message = received.substring(delimiter_1 + 2, delimiter_1 + lengthMessage + 1);
```

Até agora o código demonstrado é genérico para os dois dispositivos, para o receptor e para o emissor.

tendo agora a necessidade neste relatório de separar o código do receptor que corresponde a explicação a seguir, com o código do emissor que será apresentado na 2º fase do trabalho pois tem uma ligação com a tecnologia de RFID.

Receptor:

Depois do tratamento da String, é validada com uma chave RFID e responde 1 caso seja para activar o Servo, caso contrário, nada acontece.

Explicar o comando AT

```
if (message == "93 F5 25 1F") {  // Substitua pelo UID do seu cartão válido
  Serial.print("Cartao valido!: \n");
  //delay(tempoDelay);

  lora.print("AT+SEND=1,1,1\r\n");
  Serial.println(lora.readString());
}
```

2º Fase

A tecnologia RFID (Radio Frequency Identification, ou Identificação por Rádio Frequência) é um sistema de identificação automática que utiliza ondas de rádio para capturar e armazenar dados remotamente. Este sistema permite a leitura de informações armazenadas em dispositivos conhecidos como tags ou etiquetas RFID, sem a necessidade de contacto físico direto ou linha de visão entre o leitor e a etiqueta.

Emissor(LoraWan):

```
if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
  // Mostra UID na serial
  String conteudo = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    conteudo.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : " "));
    conteudo.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  conteudo.toUpperCase();
  Serial.println("UID do Cartao:" + conteudo);
  ESP.println("UID do Cartao:" + conteudo);

  lora.print("AT+SEND=0," + String(conteudo.length()) + "," + String(conteudo) + "\r\n");
  Serial.print(lora.readString());
}
```

mfrc522.PICC_IsNewCardPresent()

Esta função verifica se um novo cartão RFID foi apresentado ao leitor. Retorna true se um novo cartão for detectado.

mfrc522.PICC_ReadCardSerial()

Esta função tenta ler o UID do cartão. Se a leitura for bem-sucedida, retorna true.

- Uma **string conteudo** é inicializada para armazenar o UID do cartão.
- Um loop percorre cada byte do UID (mfrc522.uid.uidByte[i]), formatando cada byte como um valor hexadecimal.
- Se o byte for menor que 0x10, é adicionado um "0" à string para manter a formatação consistente (dois dígitos por byte).
- Todos os caracteres são convertidos para letras maiusculas com toUpperCase() para normalizar a string e executar uma validação mais eficiente.

3º Fase(Telegram)

O objectivo desta fase do projecto é criar uma ponte do arduino para o telegram, para isso foi usado uma comunicação Serial através de cabos do arduino para o esp32, visto que o arduino não tem módulo de wifi, assim conseguimos descentralizar mais a comunicação para facilitar falhas ou problemas caso existem.

Na imagem a seguir, demonstra as bibliotecas necessárias, e a preparação das variáveis para a execução do código.

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

// Dados do Wifi
#define WIFI_SSID "Esp32_TG"
#define WIFI_PASSWORD "Esp32_TG_Project"

//Telegram Bot Token (Botfather)
#define BOT_TOKEN "7080721921:AAHV-j3ue0-E7r0UxKZ4FLJ_d0PX2rJ4UDg"

//Use Midbot (IDBot) para saber qual o seu ID do Telegram
#define CHAT_ID "341240590"
```

Inicialização do wifi e do telegram

```
// attempt to connect to Wifi network:
void setupWifi() {
  Serial.print("Connecting to Wifi SSID ");
  Serial.print(WIFI_SSID);

  // Inicia a tentativa de conexão à rede WiFi utilizando o SSID e a password fornecidos
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  // Define o certificado raiz para a comunicação segura com a API do Telegram
  secured_client.setCACert(TELEGRAM_CERTIFICATE_ROOT);

  // Espera até que a conexão WiFi seja estabelecida
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  // Imprime na consola que a conexão WiFi foi estabelecida e mostra o endereço IP local
  Serial.print("\nWiFi connected. IP address: ");
  Serial.println(WiFi.localIP());
```

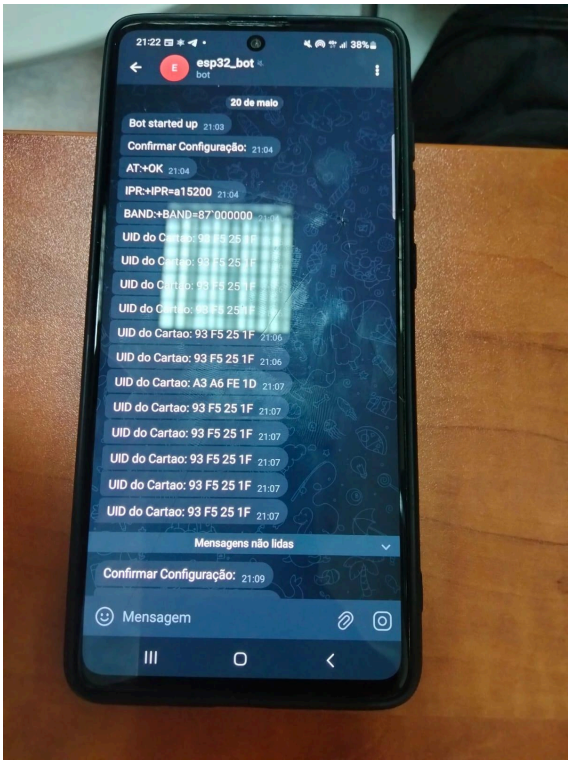
```
  Serial.println(now);
  // Envia uma mensagem através do bot do Telegram para o CHAT_ID especificado, indicando que o bot foi iniciado
  bot.sendMessage(CHAT_ID, "Bot started up", "");
}
```

```
void loop() {

  // Verifica se há dados disponíveis na porta Serial
  if (Serial.available()) {

    // Lê a mensagem da porta Serial e armazena-a na variável 'mensage'
    // Imprime na consola o número de bytes disponíveis na porta Serial
    String mensage = Serial.readString();
    Serial.print(Serial.available());

    // Envia a mensagem lida através do bot do Telegram para o CHAT_ID especificado
    bot.sendMessage(CHAT_ID, String(mensage), "");
  }
```

O objetivo desta fase era um alerta, sendo a única função a ser implementada, além da configuração do Wi-Fi, que inclui o Wi-Fi e o Telegram, apenas o envio do ESP32 para um bot do Telegram. No entanto, durante a pesquisa, foi investigada a função inversa do bot do Telegram enviando para o ESP32, caso fosse necessária a comunicação inversa. Na imagem seguinte, ficou documentada a existência da função; contudo, não foi necessário implementá-la para o projeto.

```
// Função para lidar com novas mensagens recebidas pelo bot
void handleNewMessages(int numNewMessages) {
  // Imprime mensagem indicando que está lidando com novas mensagens
  Serial.println("handleNewMessages");
  // Imprime o número de novas mensagens recebidas
  Serial.println(String(numNewMessages));

  // Loop para iterar sobre cada uma das novas mensagens
  for (int i=0; i<numNewMessages; i++) {
    // Obtém o ID do chat da mensagem atual e converte para String
    String chat_id = String(bot.messages[i].chat_id);

    // Verifica se o ID do chat não é autorizado
    if (chat_id != CHAT_ID){
      // Envia mensagem informando que o usuário não está autorizado e continua para a próxima mensagem
      bot.sendMessage(chat_id, "Utilizador não autorizado", "");
      continue;
    }

    // Obtém o texto da mensagem atual
    String text = bot.messages[i].text;
    // Imprime o texto da mensagem
    Serial.println(text);

    // Obtém o nome do remetente da mensagem
    String from_name = bot.messages[i].from_name;
  }
}
```

No bot, é recebido a configuração dos modulos do Lora, e os cartões que sao lidos no momento, este sistema nao mostra os que são validados ou não, simplesmente mostra os cartões que foram lidos.
Exemplo a baixo, foi usado dois cartões
1-Cartão:93 F5 25 1F
2-Cartão: A3 A6 FE1D



Desafios e Obstáculos

Neste capítulo, exploraremos os desafios enfrentados durante o desenvolvimento do projeto, desde dificuldades de comunicação até obstáculos na integração de diferentes componentes.

Problemas com Antenas Lora REYAX RYLRD890 e REYAX RYLRD990:
Inicialmente, enfrentamos problemas significativos com as antenas Lora da marca REYAX, especificamente os modelos RYLRD890 e RYLRD990. Apesar dos esforços, não conseguimos estabelecer uma comunicação bem-sucedida. Investigamos minuciosamente, mas o problema persistiu, exigindo uma abordagem mais profunda.

Interferência e Ruído com Módulo Leitor RFID RC522:
Juntamente com os contratempos enfrentados com as antenas Lora, nos deparamos com um desafio adicional ao trabalhar com o módulo leitor RFID RC522. Descobrimos que havia uma interferência significativa na comunicação Lora devido ao ruído gerado pelo módulo RFID RC522. Esta descoberta foi atribuída ao fato de que ambas as tecnologias operam em faixas de frequência sub gigahertz. A resolução deste problema exigiu um meticuloso processo de depuração para identificar e mitigar a fonte do ruído, garantindo assim uma comunicação Lora mais estável e confiável.

Integração com ESP32 e Arduino Uno:
Ao tentar integrar os componentes ao ESP32, nos deparamos com complicações inesperadas. Embora tenhamos feito esforços significativos, não conseguimos fazer o ESP32 funcionar de maneira satisfatória com os dispositivos mencionados. No entanto, encontramos alguma medida de sucesso ao realizar tentativas de integração com o Arduino Uno. Embora não tenhamos alcançado a solução ideal, essas experiências forneceram-nos insights valiosos para abordagens futuras.

Conclusão

Os resultados obtidos ao longo do projeto não foram os esperados, devido à presença de ruído e à ineficiência da comunicação, não tendo sido possível identificar a origem desse problema. No entanto, a investigação realizada ao longo do projeto contribuiu significativamente para o desenvolvimento da área da computação distribuída.

Foi desenvolvido conhecimento em sub-gigahertz para compreender a comunicação e a importância das frequências na atualidade.

A tecnologia RFID é uma ferramenta poderosa para a identificação e rastreamento de objetos e pessoas, proporcionando eficiência, precisão e automação em uma ampla gama de aplicações. No entanto, os desafios relacionados ao custo, interferência e privacidade precisam ser cuidadosamente considerados na implementação de sistemas RFID.

WebGrafia

RFID:
https://www.ncontrol.com.pt/o-que-e-rfid.html

Microcontroladores:
https://docs.arduino.cc/learn/starting-guide/whats-arduino/
https://docs.arduino.cc/learn/built-in-libraries/software-serial/

Módulo REYAX RYLRD890:
https://reyax.com/products/RYLR8907

Módulo Leitor RFID RC522:
https://mauser.pt/catalog/product_info.php?products_id=096-8517

Telegram:
https://web.telegram.org/k/