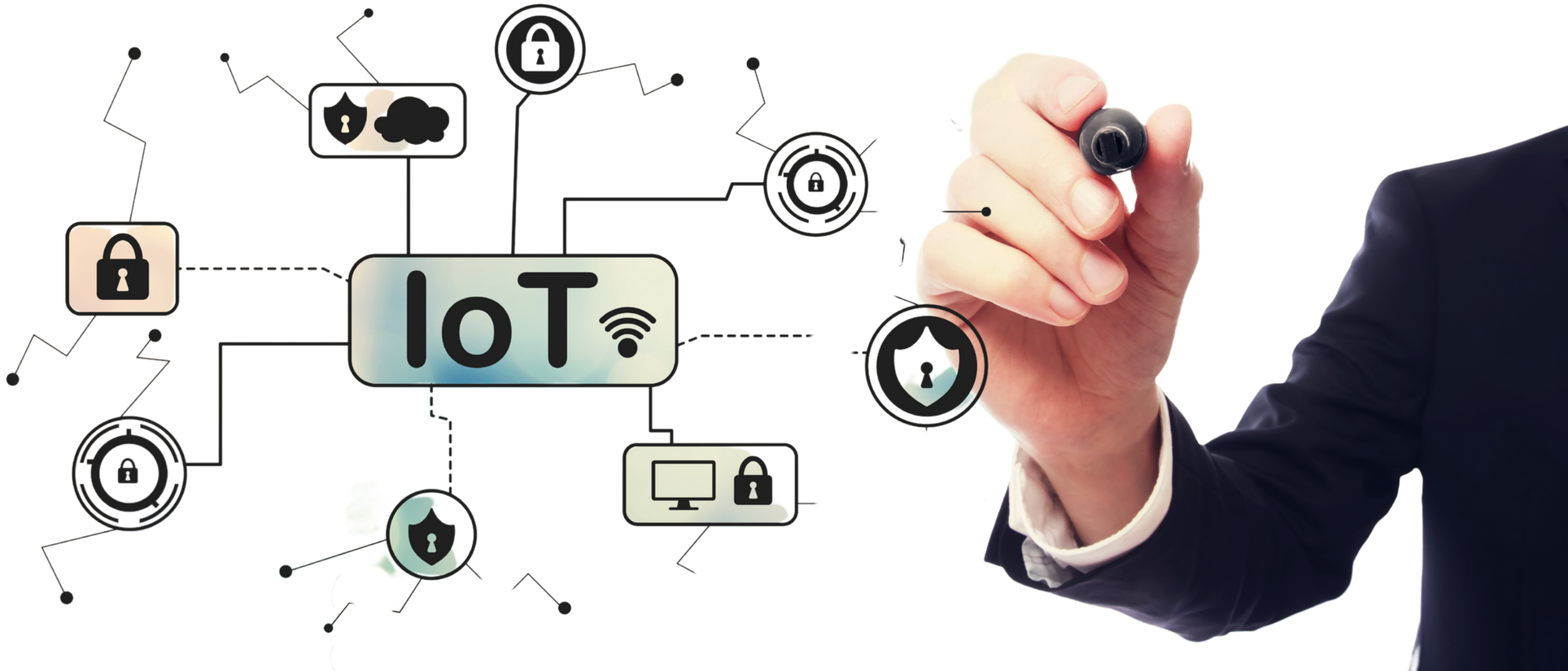
Aula 9

# SOLUÇÕES INTEGRADAS COM IOT
# (Tecnologia Lora e o protocolo LoraWan)

Prof. Lucas Carvalho

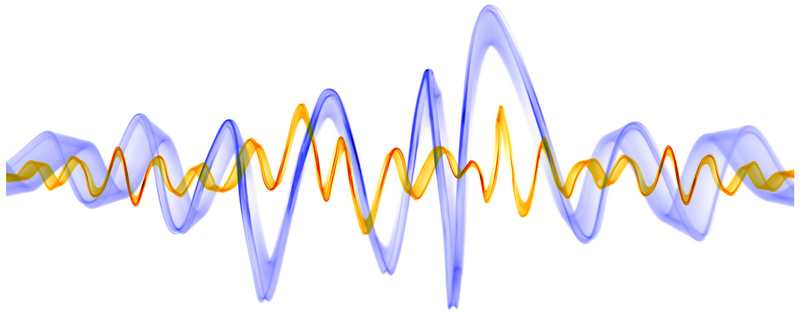



SENAI
Roberto Mange

# Tecnologia LoRa

**LoRa (Long Range) consiste de uma tecnologia de radiofrequência que permite comunicações em longas distâncias (na ordem de grandeza de alguns quilômetros), utilizando para isso um baixo consumo de energia elétrica.**

**Em termos de frequências de operação, a tecnologia LoRa utiliza frequências sub-gigahertz (abaixo de 1GHz), em bandas dedicadas de acordo com as regiões do planeta...**

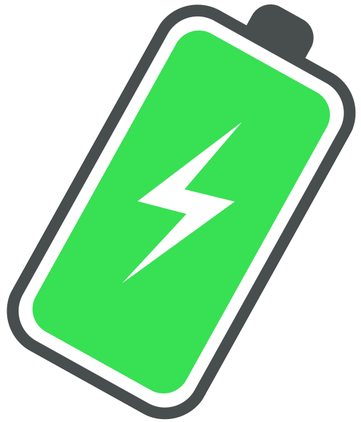


# CUIDADO

**Sempre respeite essa frequência  (abaixo de 1GHz) em seus projetos e produtos. Não obedecer isso pode levar a não homologação de seu produto (logo, a proibição de sua venda) e também interferências em outros sistemas. A depender do dano causado por tais interferências a sistemas terceiros, pode haver consequências legais / jurídicas para a empresa responsável pelo projeto.**

# Eficiência energética

**Uma das características distintivas do LoRaWAN é sua capacidade de estender a vida útil da bateria dos dispositivos conectados. Isso é alcançado por meio de estratégias como ajustes inteligentes de potência, que permitem que os dispositivos usem apenas a quantidade necessária de energia para transmitir dados.**



# Alcance longo

**A tecnologia LoRa, na qual o LoRaWAN se baseia, é conhecida por seu alcance de comunicação excepcionalmente longo (em áreas urbanas 3-4 Km de alcance, e em áreas rurais, até 12 Km ou mais). Isso permite que os dispositivos se comuniquem em distâncias consideráveis, tornando-a ideal para aplicações em áreas rurais ou urbanas densas.**

km

# Variedade de Aplicações

**O LoRaWAN é flexível o suficiente para suportar uma ampla gama de casos de uso. Desde sensores ambientais que monitoram a qualidade do ar e níveis de poluição até dispositivos industriais que controlam processos de fabricação, o protocolo LoRaWAN atende a diversos cenários de aplicação.**

# Segurança

**O protocolo LoRaWAN incorpora medidas de segurança para garantir que as comunicações entre dispositivos e a rede central sejam protegidas contra ameaças. Isso é vital para garantir a privacidade dos dados transmitidos e a integridade das operações.**

# Padrões Abertos

**O LoRaWAN é desenvolvido em conformidade com padrões abertos, permitindo que diferentes fabricantes e fornecedores implementem essa tecnologia em seus dispositivos e redes. Isso promove a interoperabilidade e a disponibilidade de opções no mercado.**

# PLACA WIFI LORA ESP32 / DISPLAY OLED HELTEC SX1276 868MHz-915MHz

# Utilização do módulo com a Arduino IDE

A primeira etapa a ser feita é instalar as bibliotecas básicas para uso do ESP32, adicionando o endereço abaixo na ABA preferencias da IDE:

https://dl.espressif.com/dl/package_esp32_index.json

Vá em gerenciador de placas e instale a placa esp32.

Após instalação da placa, é necessário instalar a biblioteca para utilizar o rádio LoRa. Para isso, instale o arquivo zip da biblioteca fornecida pelo professor, seguindo os passos abaixo:

Na Arduino IDE, vá em Sketch > Include Library > Add .zip Library.

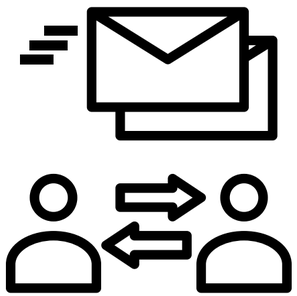
Na janela que surgir, vá até a pasta que você salvou a biblioteca baixada, clique sobre o arquivo .zip e clique em Ok.

Feito isso, a biblioteca está adicionada e pronta para uso.

Instale também as bibliotecas **Heltec Esp32 - Dev Boards** que é específica para dispositivos Heltec WiFi Kit. Ela oferece recursos para controlar o display OLED integrado e configurações iniciais do dispositivo, como inicialização da rede WiFi e configurações de rádio LoRa.

Para a aplicação da aula vamos utilizar duas placas Esp32 Lora, uma será configurada como "sender" e a outra como "receiver"

No modo sender, vamos preparar os dados que desejamos transmitir, no nosso caso iremos enviar uma mensagem de liga/desliga para a irrigação. Use a biblioteca LoRa para empacotar esses dados e transmiti-los.



# Programação do módulo sender

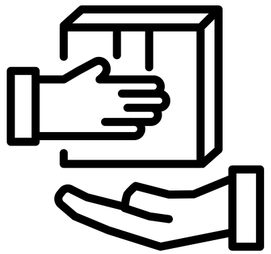
```
#include "heltec.h"
#include "images.h"

#define BAND    868E6  // Definição da frequência de operação

unsigned int counter = 0;  // Variável para contar algo, mas não está sendo usada aqui
String rssi = "RSSI --";   // Variável para guardar a intensidade do sinal
String packSize = "--";    // Variável para guardar o tamanho do pacote
String packet ;            // Variável para guardar o conteúdo do pacote
bool irrigationOn = false; // Variável para armazenar o estado da irrigação

const int buttonPin = 12;  // Pino onde o botão está conectado
int variable = 0;          // Variável para armazenar um valor (0 ou 1) que será enviado

int buttonState = HIGH;      // Estado inicial do botão
int lastButtonState = HIGH;  // Estado anterior do botão

void logo()
{
  Heltec.display->clear();
  Heltec.display->drawXbm(0,5,logo_width,logo_height,logo_bits);  // Exibe um logotipo no display
  Heltec.display->display();
}
```

```
void logo()
{
  Heltec.display->clear();
  Heltec.display->drawXbm(0,5,logo_width,logo_height,logo_bits);  // Exibe um logotipo no display
  Heltec.display->display();
}

void setup()
{
  pinMode(buttonPin, INPUT_PULLUP);  // Configura o pino do botão como entrada com resistor de pull-up interno

  Heltec.begin(true, true, true, true, BAND);  // Inicialização do dispositivo Heltec WiFi Kit

  Heltec.display->init();
  Heltec.display->flipScreenVertically();
  Heltec.display->setFont(ArialMT_Plain_10);
  logo();
  delay(1500);
  Heltec.display->clear();

  Heltec.display->drawString(0, 0, "Heltec.LoRa Initial success!");
  Heltec.display->display();
  delay(1000);
```

```
void loop()
{
  Heltec.display->clear();
  Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT);
  Heltec.display->setFont(ArialMT_Plain_10);

  Heltec.display->drawString(0, 0, "Enviando msg: ");
  Heltec.display->drawString(90, 0, String(variable));  // Mostra o valor da variável

  // Exibe o status da irrigação
  if (irrigationOn) {
    Heltec.display->drawString(0, 20, "Irrigação: Ligada");
  } else {
    Heltec.display->drawString(0, 20, "Irrigação: Desligada");
  }

  Heltec.display->display();

  // Envio do pacote
  LoRa.beginPacket();
  LoRa.print(variable);
  LoRa.endPacket();

  buttonState = digitalRead(buttonPin);  // Lê o estado atual do botão
```

```
  if (buttonState != lastButtonState) {  // Verifica se houve mudança no estado do botão
    if (buttonState == LOW) {  // Verifica se o botão foi pressionado
      variable = 1 - variable;  // Alterna o valor da variável entre 0 e 1
      irrigationOn = !irrigationOn;  // Alterna o estado da irrigação
    }

    delay(50);  // Pequena pausa para evitar o efeito de bouncing (repiqueteio) do botão
  }

  lastButtonState = buttonState;  // Atualiza o estado anterior do botão
}
```

No modo receiver, vamos configurar a placa para receber os dados. Vamos precisar definir um mecanismo para lidar com os dados recebidos, como desempacotá-los e tomar ações apropriadas com base nas informações contidas neles.



# Programação do módulo receiver

```
#include "heltec.h"  // Inclui a biblioteca Heltec para usar as funcionalidades do dispositivo
#include "images.h"   // Inclui a biblioteca de imagens (provavelmente ícones para o display)

#define BAND    868E6  // Define a frequência de operação do rádio LoRa

String rssi = "RSSI --";    // Variável para guardar a intensidade do sinal
String packSize = "--";     // Variável para guardar o tamanho do pacote
String packet ;             // Variável para guardar o conteúdo do pacote
const int relayPin = 13;    // Pino para controlar o relé
bool irrigationOn = false;  // Variável para guardar o estado da irrigação

void logo(){
  Heltec.display->clear();   // Limpa o display
  Heltec.display->drawXbm(0,5,logo_width,logo_height,logo_bits);  // Exibe um logotipo no display
  Heltec.display->display();  // Atualiza o display
}

void LoRaData(){
  Heltec.display->clear();   // Limpa o display
  Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT);  // Define o alinhamento do texto
  Heltec.display->setFont(ArialMT_Plain_10);         // Define a fonte do texto
  Heltec.display->drawString(0 , 15 , "Received "+ packSize + " bytes");  // Exibe informações sobre o tamanho do pacote
  Heltec.display->drawStringMaxWidth(0 , 26 , 128, packet);  // Exibe o conteúdo do pacote no display
  Heltec.display->drawString(0, 0, rssi);  // Exibe a intensidade do sinal
  Heltec.display->display();  // Atualiza o display
}
```

```
void cbk(int packetSize) {
  packet = "";  // Limpa o conteúdo do pacote
  packSize = String(packetSize,DEC);  // Converte o tamanho do pacote para String
  for (int i = 0; i < packetSize; i++) { packet += (char) LoRa.read(); }  // Lê o conteúdo do pacote
  rssi = "RSSI " + String(LoRa.packetRssi(), DEC);  // Lê a intensidade do sinal
  LoRaData();  // Atualiza o display com os dados recebidos
}

void setup() {
  pinMode(relayPin, OUTPUT);  // Configura o pino do relé como saída

  // Inicialização do dispositivo Heltec WiFi Kit
  Heltec.begin(
    true /*DisplayEnable Enable*/,
    true /*Heltec.Heltec.Heltec.LoRa Disable*/,
    true /*Serial Enable*/,
    true /*PABOOST Enable*/,
    BAND /*long BAND*/
  );
```

```
  Heltec.display->flipScreenVertically();  // Inverte a orientação do display
  Heltec.display->setFont(ArialMT_Plain_10);  // Define a fonte do texto
  logo();  // Exibe o logotipo no display
  delay(1500);
  Heltec.display->clear();

  Heltec.display->drawString(0, 0, "Heltec.LoRa Initial success!");
  Heltec.display->drawString(0, 10, "Wait for incoming data...");
  Heltec.display->display();
  delay(1000);

  LoRa.receive();  // Entra no modo de recepção de dados LoRa
}

void loop() {
  int packetSize = LoRa.parsePacket();  // Verifica se há um pacote disponível para leitura
  if (packetSize) { cbk(packetSize); }  // Se houver um pacote, processa-o
  delay(10);  // Pequena pausa
}

void checkIrrigation() {
  if (packet == "1") {  // Se o conteúdo do pacote for "1"
    if (!irrigationOn) {  // Se a irrigação ainda não estiver ligada
      irrigationOn = true;  // Liga a irrigação
      digitalWrite(relayPin, HIGH);  // Ativa o relé
    }
  } else {
    irrigationOn = false;  // Desliga a irrigação
    digitalWrite(relayPin, LOW);  // Desativa o relé
  }
}
```