

**UNIVERSIDADE FEDERAL DE SANTA MARIA  
COLÉGIO TÉCNICO INDUSTRIAL DE SANTA MARIA  
CURSO TÉCNICO EM INFORMÁTICA PARA INTERNET**

**RELATÓRIO DE ESTÁGIO REALIZADO NO COLÉGIO TÉCNICO  
INDUSTRIAL DE SANTA MARIA**

Santa Maria, RS  
2022

## FORMULÁRIO DE IDENTIFICAÇÃO

|   |  |   |
|---|--|---|
| <b>Nome:</b><br>ANA CAROLINE SCHERER DE OLIVEIRA                                      | <b>Matrícula:</b><br>201811276002              |   |
| <b>Período de estágio:</b><br><b>Início:</b> 19/07/2021<br><b>Término:</b> 31/12/2021 | <b>Total de horas de estágio:</b><br>400 Horas | <input checked="" type="checkbox"/> ( X ) Estágio<br><input type="checkbox"/> ( ) Aproveitamento profissional |
| <b>Local de realização do estágio:</b><br>COLÉGIO TÉCNICO INDUSTRIAL DE SANTA MARIA   |  |   |
| <b>Nome do supervisor de estágio:</b><br>FÁBIO TEIXEIRA FRANCISCATO                   |  |   |
| <b>Nome do orientador de estágio:</b><br>FÁBIO TEIXEIRA FRANCISCATO                   |  |   |

**UNIVERSIDADE FEDERAL DE SANTA MARIA COLÉGIO  
TÉCNICO INDUSTRIAL DE SANTA MARIA CURSO TÉCNICO EM  
INFORMÁTICA PARA INTERNET**

**RELATÓRIO DE ESTÁGIO REALIZADO NO COLÉGIO TÉCNICO  
INDUSTRIAL DE SANTA MARIA**

Relatório de Estágio apresentado ao Colégio Técnico Industrial de Santa Maria – CTISM, da Universidade Federal de Santa Maria – UFSM, realizado no Colégio Técnico Industrial de Santa Maria como requisito para obtenção do Grau de Técnico em informática para internet.

Elaborado por:



Ana Caroline Scherer de Oliveira

Avaliado por:

---

Fábio Teixeira Franciscato (orientador e supervisor)

Santa Maria, RS

2022

## APRESENTAÇÃO

Durante meses de procura independente por um estágio na área do curso e divergências com empresas, foi ao conversar com o professor orientador e supervisor do estágio Fábio Teixeira Fransiscato que surgiu uma verdadeira oportunidade de realização de estágio e descoberta de novos conhecimentos. Ao buscar pelo seu auxílio, ele logo ofertou uma vaga participante do desenvolvimento **HARDWARE MULTI SENSOR COM COMUNICAÇÃO A LONGAS DISTÂNCIAS PARA MEDIÇÕES DE FENÔMENOS EM AMBIENTES REMOTOS**.

O projeto surge da cooperação de iniciativa público e privada no objetivo comum de oferecer auxílio na medição de fenômenos em tempo real, de maneira compacta e facilitadora, na propriedade de pequenos agricultores, principalmente ao que concerne a agricultura familiar. Nele se propõe uma solução de hardware e software integrados junto às tecnologias atuais, como a *Internet of Things*.

O projeto ainda tem um longo caminho a percorrer para sua conclusão e execução plena de seus objetivos. O início pode parecer confuso pela incerteza de como será o produto final, mas o desafio foi aceito e assim se foram 6 meses de aprendizado e prática que resultaram no presente relatório.

## LISTA DE FIGURAS

|   |    |
|---|----|
| Figura 1: Logo do Arduino.....                                    | 10 |
| Figura 2: Placa LoRa ESP32 V2.....                                | 11 |
| Figura 3: Logo do Tinkercad.....                                  | 11 |
| Figura 4: Representação da placa Arduino Uno da atividade 1.....  | 12 |
| Figura 5: Código da atividade 1.....                              | 13 |
| Figura 6: Representação da placa Arduino Uno da atividade 2.....  | 14 |
| Figura 7: Código da atividade 2.....                              | 14 |
| Figura 8: Representação da placa Arduino Uno da atividade 3.....  | 15 |
| Figura 9: Código da atividade 3.....                              | 16 |
| Figura 10: Representação da placa Arduino Uno da atividade 4..... | 17 |
| Figura 11: Código da atividade 4.....                             | 17 |
| Figura 12: Representação da placa Arduino Uno da atividade 5..... | 18 |
| Figura 13: Código da atividade 5.....                             | 18 |
| Figura 14: Representação da placa Arduino Uno da atividade 6..... | 19 |
| Figura 15: Código da atividade 6.....                             | 19 |
| Figura 16: Barra de arquivos do Arduino.....                      | 20 |
| Figura 17: Preferências dos Arduino.....                          | 21 |
| Figura 18: Zoom nas preferências do Arduino.....                  | 21 |
| Figura 19: Barra de ferramentas do Arduino.....                   | 22 |
| Figura 20: Gerenciado de placas do Arduino.....                   | 22 |
| Figura 21: Barra de sketch do Arduino.....                        | 23 |
| Figura 22: Gerenciador de bibliotecas do Arduino.....             | 23 |
| Figura 23: Primeira parte do código de conexão WEB.....           | 24 |
| Figura 24: Segunda parte do código de conexão WEB.....            | 25 |
| Figura 25: Página WEB criada para a atividade.....                | 25 |
| Figura 26: Primeira parte do código do emissor.....               | 26 |
| Figura 27: Segunda parte do código do emissor.....                | 27 |
| Figura 28: Terceira parte do código do emissor.....               | 27 |
| Figura 29: Primeira parte do código do receptor.....              | 28 |
| Figura 30: Segunda parte do código do receptor.....               | 28 |
| Figura 31: Terceira parte do código do receptor.....              | 29 |
| Figura 32: Quarta parte do código do receptor.....                | 29 |
| Figura 33: Quinta parte do código do receptor.....                | 30 |

## SUMÁRIO

|   |    |
|---|----|
| 1. Introdução.....                                  | 7  |
| 1.1 O Projeto.....                                  | 7  |
| 1.2 O Estágio.....                                  | 7  |
| 2. Aprendizado.....                                 | 8  |
| 2.1 IoT.....  | 8  |
| 2.2 Tecnologias de Hardware.....                    | 9  |
| 2.2.1 Arduino.....                                  | 9  |
| 2.2.2 LoRa ESP32.....                               | 10 |
| 2.3 Tinkercad.....                                  | 11 |
| 2.4 Atividades.....                                 | 12 |
| 2.4.1 Arduino 1 – Acender um LED.....               | 12 |
| 2.4.2 Arduino 2 – Ligar três LEDs em sequência..... | 13 |
| 2.4.3 Arduino 3 – Semáforo.....                     | 15 |
| 2.4.4 Arduino 4 – Push button.....                  | 16 |
| 2.4.5 Arduino 5 – Potenciômetro.....                | 18 |
| 2.4.6 Arduino 6 – Display de sete segmentos.....    | 19 |
| 2.4.7 Configuração da IDE para ESP32.....           | 20 |
| 2.4.8 ESP32 1 – Comunicação com WEB.....            | 24 |
| 2.4.9 ESP32 2 – Comunicação entre duas placas.....  | 26 |
| 3. Dificuldades.....                                | 30 |
| 4. Sugestões.....                                   | 31 |
| 5. Conclusão.....                                   | 31 |
| 6. Referências.....                                 | 32 |

## **1. Introdução**

Diante a conclusão do estágio, este relatório é escrito com intenção de descrever as atividades desenvolvidas nos seus cinco meses e meio de duração e, por meio deste, adquirir o diploma de conclusão do curso. Com orientação e supervisão do professor Fábio Teixeira Franciscato, o estágio foi realizado no Colégio Técnico Industrial de Santa Maria através do projeto de pesquisa denominado **HARDWARE MULTI SENSOR COM COMUNICAÇÃO A LONGAS DISTÂNCIAS PARA MEDIÇÕES DE FENÔMENOS EM AMBIENTES REMOTOS**.

### **1.1 O Projeto**

O projeto se propõe a facilitar a medição de fenômenos em propriedades de agricultura familiar de pequeno porte de maneira tecnológica e eficiente desenvolvendo uma solução de comunicação a longas distâncias e com baixo consumo de energia para uma tecnologia de hardware e software de coleta de dados a campo integrado a uma plataforma web de armazenamento de dados (FABIO, 2021).

As medições de fenômenos, tais como temperatura, umidade, molhamento foliar, pH e condutividade elétrica, devem ocorrer em tempo real, de maneira automática e remota, sendo integrada em um mesmo hardware de precisão de valor acessível e preciso em suas funcionalidades. Este hardware também deve ser compacto, de forma a poder ser levado a campo e/ou locais sem rede elétrica, fazer uso de tecnologia sem fio e ser passível de utilização em meio sólido ou aquoso. Durante o estágio inicial do projeto, é definido uma tecnologia de comunicação de baixo custo a ser utilizada, integrando-o com um meio de comunicação a longas distâncias.

### **1.2 O Estágio**

Devido ao período de pandemia de Covid-19, desde o ano de 2020 as aulas presenciais na Universidade Federal de Santa Maria estavam suspensas – incluindo o Colégio Técnico Industrial de Santa Maria e quaisquer atividades relacionadas à este pelos alunos da instituição. Foi mantido o funcionamento de apenas parte da administração da escola e, por consequência, o estágio se deu por meio remoto, isto é, meio online, até um mês antes de sua finalização. Contudo, isso não impediu a execução das atividades prescritas para o mesmo. Segundo o documento do projeto, era previsto a conclusão da primeira, segunda e terceira fase durante a duração do estágio.

| Atividades<br>/ano  | Jul/Ago<br>/21 | Set/<br>21 | Out/<br>21 | Nov/<br>21 | Dez/<br>21 | Jan/<br>22 | Fev/<br>22 | Mar/<br>22 | Abr/<br>22 | Mai/<br>22 | Jun/<br>22 | Jul/<br>22 | Ago/<br>22 |
|---|----------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| Definir a tecnologia de comunicação e hardware a ser utilizada.     |                |            |            |            |            |            |            |            |            |            |            |            |            |
| Integrar ao hardware o meio de comunicação a longas distâncias.     |                |            |            |            |            |            |            |            |            |            |            |            |            |
| Integrar ao hardware o funcionamento por baterias.                  |                |            |            |            |            |            |            |            |            |            |            |            |            |
| Otimizar hardware e software para diminuição do consumo de energia. |                |            |            |            |            |            |            |            |            |            |            |            |            |

*Tabela 1: Cronograma do primeiro ano do projeto retirado do mesmo.*

No decorrer do estágio foi realizada a definição da tecnologia de comunicação e hardware a ser utilizado e integração do hardware ao meio de comunicação de longas distâncias.

## 2. Aprendizado

Para ser possível realizar as funções do estágio, antes foi necessário familiarizar-se com o conteúdo a ser criado dentro do objetivo geral do projeto, criando uma metodologia própria de estudos e resolução de exercícios práticos a fim de gerar conhecimento e experiência o suficiente para práticas posteriores.

### 2.1 IoT

A IoT - do inglês *Internet of Things* ou Internet das Coisas em português é descrita como uma revolução da tecnologia da informação, sendo capaz de mudar as relações acadêmicas, indústrias e individuais para com a tecnologia. Ela surge de diferentes áreas da tecnologia, podendo ser aplicada nos mais diferentes contextos.



Através da IoT, é possível a amplificação do que conhecemos na Internet dentro do ambiente online para além deste, proporcionando objetos do dia a dia se comunicarem com a rede mundial de computadores, estes são os chamados objetos inteligentes. Uma consequência direta deste fato é o impulso no avanço da criação de novas tecnologias capazes de facilitar a vida. TVs, automóveis, consoles de jogos, geladeiras, câmeras, entre muitos outros itens da vida comum, ganham através da IoT novas funcionalidades que condizem aos avanços tecnológicos da própria época dos quais estão inseridos.

Usando os recursos desses objetos será possível detectar seu contexto, controlá-lo, viabilizar troca de informações uns com os outros, acessar serviços da Internet e interagir com pessoas (BRUNO,2021).

## **2.2 Tecnologias de Hardware**

Para confecção de um dispositivo capaz de fazer medições de fenômenos em ambientes remotos antes foi necessário escolher um microcomputador capaz de satisfazer todas as exigências, sendo elas:

- Baixo custo;
- Tamanho reduzido;
- Baixa utilização de bateria;
- Possibilidade de integração com sensores;
- Passível de comunicação a longa distância.

Os dois dispositivos que melhor satisfazem as exigências são o Arduino LoRa ESP32 V2, sendo o último o escolhido para o projeto por sua vantagem de deter um meio de comunicação a longa distância já acoplado na própria placa de prototipagem, ao contrário do Arduino cuja funcionalidade só seria possível através do uso de sensores. Contudo, o Arduino ainda está presente no projeto, principalmente pela facilidade de aprendizado deste através de softwares de simulação, além de uma plataforma no qual serão realizados os programas a serem executados pela placa LoRa ESP32.

### **2.2.1 Arduino**

O Arduino consiste em uma placa de prototipagem e plataforma de código aberto. Sua utilização pode se fazer através de uma vasta gama de aplicações, podendo ir até mesmo de ligar um simples LED até complexos instrumentos científicos. Ademais, por ser uma platafor-

ma de código aberto, muito do conhecimento produzido dentro do mesmo é compartilhado através de uma comunidade formada dos mais variados perfis - estudantes, professores, profissionais, leigos, artistas, entre outros, juntos criam conhecimento acessível para todos.

Inicialmente o Arduino foi criado como uma ferramenta simples de prototipagem direcionada a um público leigo em eletrônica e programação, porém ao atingir maior alcance e, consequentemente, público, o projeto simples nascido no Instituto de Design de Interação Ivrea acabou por se adaptar para novas funcionalidades. A aplicabilidade mais comum do Arduino ocorre em associação a IoT e na impressão 3D.

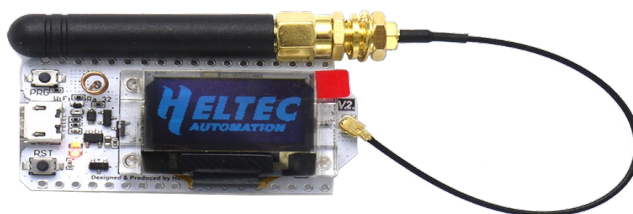


*Figura 1: Logo do Arduino*

### 2.2.2 LoRa ESP32

O LoRa ESP32 V2 é um microcontrolador com WiFi e Bluetooth integrados e uma tela OLED no qual é possível ver a execução de alguns programas simples. Além disso, outra funcionalidade se refere ao uso de outro tipo de comunicação a longa distância chamado LoRa - uma sigla para *Long Range*, em português longa distância. Esse tipo de sistema visa ser utilizado em dispositivos alimentados por baterias de longa duração, em que o consumo de energia é de suma importância.

Sua funcionalidade pode se manter de maneira plena em ambientes industriais, operando em temperaturas de mais de 120°C e até -40°C. A programação deste microcontrolador pode se dar por diversos meios, sendo o mais comum, e utilizado durante o período de estágio, a IDE do Arduino.



*Figura 2: Placa LoRa ESP32 V2*

### 2.3 Tinkercad

O Tinkercad é uma plataforma interativa onde é possível criar circuitos e projetos 3D através de uma interface extremamente simples e intuitiva. Nele é simulado todas as funcionalidades de um circuito, incluindo modelos de placa e sensores. Além da possibilidade de programação através de blocos de comando dentro da plataforma, tornando o aprendizado mais inclusivo podendo ser utilizada até mesmo por pessoas leigas sobre a lógica de programação.



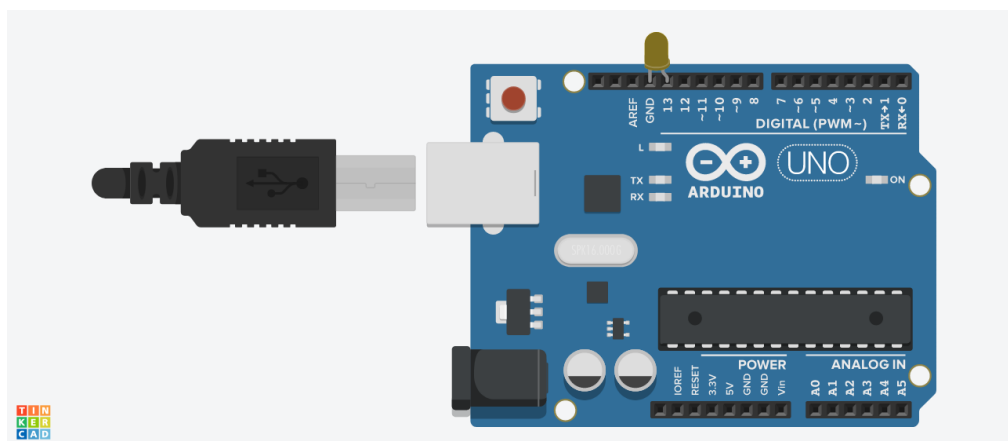
*Figura 3: Logo do Tinkercad*

## 2.4 Atividades

A partir de uma análise geral do conteúdo necessário para o desenvolvimento das funções do estágio, foi seguido um cronograma de exercícios para familiarização da plataforma IDE e suas aplicações. Foram realizadas oito atividades relacionadas ao Arduino na plataforma Tinkercad e duas atividades utilizando a placa de desenvolvimento LoRa ESP 32 V2 e IDE.

### 2.4.1 Arduino 1 – Acender um LED

A primeira atividade realizada utilizando a plataforma Tinkercad consistiu em uma programa simples com o objetivo de acender um LED. Para realização do programa no simulador foi necessário uma placa de Arduino e um Led.



*Figura 4: Representação da placa Arduino Uno da atividade 1*

A perna maior do LED, ao qual se refere ao pólo positivo do mesmo, é conectada ao pino treze para que seja executado o comando de ligado e desligado a partir deste pino, enquanto a perna menor do LED, referente ao pólo negativo, é conectada ao GND (abreviatura da palavra inglês *ground*, ou em português terra) cujo a função é servir de referência para tensão em um circuito, trabalhando como aterramento do mesmo. Dessa forma, junto ao simulador da IDE do Arduino dentro da plataforma, se obtém o seguinte código para que o objetivo da atividade seja atingido com êxito.

```

1 void setup(){
2
3   pinMode(13, OUTPUT); //OUTPUT = saída |
4   // O pino de porta número 13 funcionará como saída
5
6 }
7
8 void loop(){
9
10  digitalWrite(13, HIGH); //HIGH = ligado | Liga a LED
11  delay(1000); //espera 1000 mili segundos para executar um comando
12  digitalWrite(13, LOW); //LOW = desligado | Desliga a LED
13  delay(1000);
14 }

```

Figura 5: Código da atividade 1.

O comando *void* é utilizado apenas ao declarar funções. Ela indica que é esperado que a função não retorne nenhuma informação para a função da qual foi chamada, *setup()* e *loop()* são nomes padrão de funções. A função *pinMode*(pino,modo) se refere à configuração do pino especificado como entrada ou saída. O pino se refere ao número do pino do qual está sendo configurado (ex: pino de número 13) e modo se refere ao modo no qual está sendo colocado o pino (*OUTPUT* (saída), *INPUT* (entrada) ou *INPUT\_PULLUP*(saída e entrada com *pull-up* ativado). A função *digitalWrite*(pino,valor) define o valor do pino.

Esse valor pode ser *HIGH* (ativado) ou *LOW* (desativado). A função *delay*(tempo em mili segundos) define o intervalo de tempo para execução do programa.

#### 2.4.2 Arduino 2 – Ligar três LEDs em sequência

A segunda atividade foi a confecção de um programa que ligasse três LEDs em sequência. Para realização do programa foi necessário uma placa de Arduino, três LEDs, seis *jumpers* macho/macho, uma *protoboard* e um resistor.

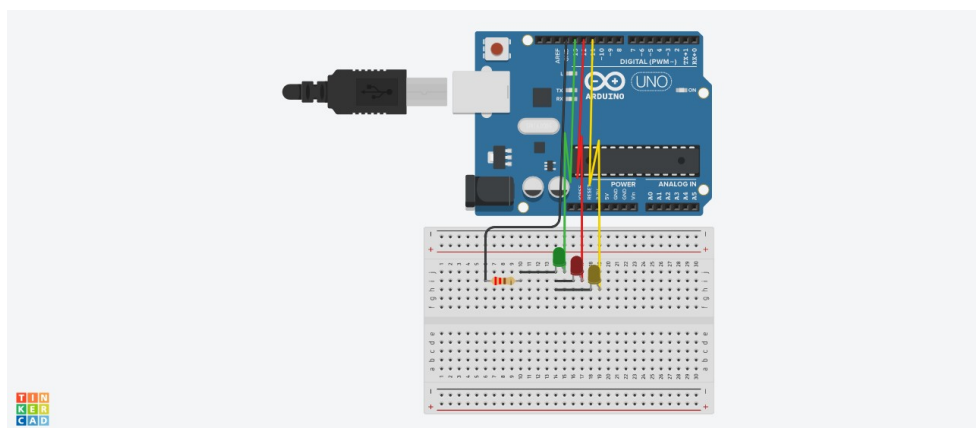


Figura 6: Representação da placa Arduino Uno da atividade 2.

Os *jumpers* são chaves elétricas utilizadas em placas e alguns dispositivos, como discos rígidos, para ativar, regular ou desativar funções específicas do sistema que não são acessíveis via software.

A *protoboard* é uma mesa de ensaios utilizada para a montagem de protótipos e projetos em estado inicial. Na parte do meio (sem indicador de + e -) as correntes passam de forma verticalmente, não se encontrando horizontalmente, já na parte de cima e de baixo as correntes passam de forma horizontal.

O resistor é um elemento que apresenta resistência à passagem de eletricidade, sua unidade de medida é ohms ( $\Omega$ ). Na atividade foi utilizado um resistor de valor de 220 ohms.

O código desenvolvido para a realização do programa é o seguinte:

```

1 // C++ code
2 //
3 void setup(){
4   pinMode(13, OUTPUT);
5   pinMode(12, OUTPUT);
6   pinMode(11, OUTPUT);
7 }
8
9 void loop()
10 {
11
12   digitalWrite(13, HIGH);
13   delay(1000);
14   digitalWrite(13, LOW);
15   delay(100);
16   digitalWrite(12, HIGH);
17   delay(1000);
18   digitalWrite(12, LOW);
19   delay(100);
20   digitalWrite(11, HIGH);
21   delay(1000);
22   digitalWrite(11, LOW);
23   delay(100);
24

```

Figura 7: Código da atividade 2

Os comandos se repetem para que sejam executadas a função do LED um por vez, fazendo que um LED acenda por um determinado tempo e ao apagar acenda outro imediatamente. Foram usados *jumpers* para que a corrente elétrica dos LEDs não se cruzassem.

### 2.4.3 Arduino 3 – Semáforo

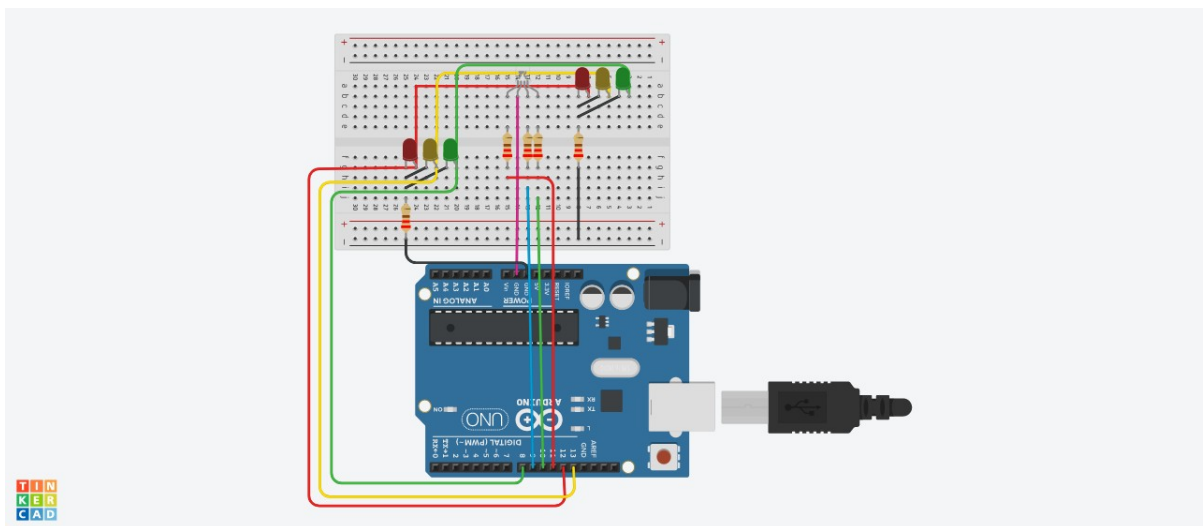


Figura 8: Representação da placa Arduino Uno da atividade 3.

O terceiro programa desenvolvido foi criado tendo como referência a lógica de uma rua com dois semáforos funcionando simultaneamente, representados por três LEDs de cores individuais, e uma faixa de pedestre utilizando um LED RGB. Foram utilizados para este programa uma placa Arduino, um LED RGB, seis LEDs (das cores verde, amarelo e vermelho, dois de cada), treze *jumpers* macho/macho, uma *protoboard* e cinco resistores de valor 220 ohms.

LEDs RGB tem quatro pernas, das quais três representam o vermelho, verde e azul, e uma que é comum a todas. As entradas para RGB devem ser aquelas contendo o til (~).

Alguns conceitos novos foram introduzidos no programa, são eles:

- *analogWrite*(pino, valor): Aciona uma onda PWM (*Pulse Width Modulation*) em um pino. Pode ser usada para variar o brilho de um LED ou acionar um motor a diversas velocidades. Foi utilizado para ligar os LEDs RGB.
- *if*(condição): É uma estrutura de controle que checa uma condição e executa o comando a seguir ou um bloco de comandos delimitados por chaves, se a condição é verdadeira. Foi utilizado para mudar as cores do LED.

Um resistor foi usado para todos os LEDs de cores iguais, cada com 220 ohm, e uma saída por cor, podendo assim ser diferenciadas. Para os LEDs de cor única foram necessários usar o mesmo *jumper*, possibilitando assim de serem acesos ao mesmo tempo.

Foi utilizado também variáveis inteiras e uma variável booliana, ambas atribuindo a cor do LED RGB, sendo possível formar novas cores além do vermelho, verde e azul. Abaixo segue o código utilizado para este programa:

```

1  /*CRIAR UM SEMAFORO UTILIZANDO 6 LEDS E UM RGB*/
2  boolean anoC = true;
3  int pinoA = 13;
4  int pinoV = 12;
5  int pinoR = 11;
6  int pinoG = 10;
7  int pinoB = 9;
8  int pinoV02 = 8;
9
10 void setup(){
11   pinMode(pinoR, OUTPUT);
12   pinMode(pinoG, OUTPUT);
13   pinMode(pinoB, OUTPUT);
14   pinMode(pinoV, OUTPUT);
15   pinMode(pinoV02, OUTPUT);
16   pinMode(pinoA, OUTPUT);
17 }
18
19 void loop(){
20   setCor(255,0,0);
21   digitalWrite(pinoV02, HIGH);
22   delay(2000);
23   digitalWrite(pinoV02, LOW);
24   setCor(255,255,0);
25   digitalWrite(pinoA, HIGH);
26   delay(2000);
27   digitalWrite(pinoA, LOW);
28   setCor(0,255,0);
29   digitalWrite(pinoV, HIGH);
30   delay(2000);
31   digitalWrite(pinoV, LOW);
32 }
33
34 void setCor(int verm, int verd, int azul){
35   if(anoC == true){
36     verm = verm++;
37     verd = verd++;
38     azul = azul++;
39   }
40   analogWrite(pinoR, verm);
41   analogWrite(pinoG, verd);
42   analogWrite(pinoB, azul);
43 }
44 }

```

Figura 9: Código da atividade 3

O programa ao ser executado mostra a mudança de cores do LED RGB conforme as cores dos LEDs de cor única acendem, indicando a passagem ou parada de um “pedestre”.

#### 2.4.4 Arduino 4 – Push button

No quarto exercício foi utilizado pela primeira vez um *push button*. Para realização desse programa foi necessário uma placa Arduino, um LED, quatro *jumpers* macho/macho, uma *protoboard* e dois resistores. Neste exemplo os resistores possuem medidas diferentes, sendo um de valor de 220 ohm para o LED e outro com valor de 10k (10.000 ohms) para o *push button*.



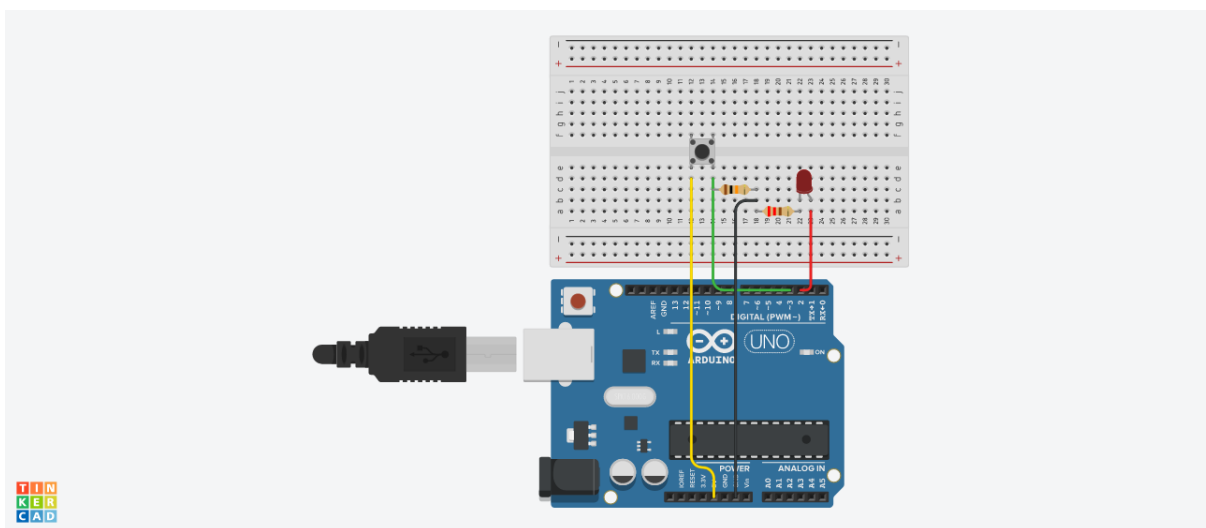


Figura 10: Representação da placa Arduino Uno da atividade 4

O *push button*, ou botão de pressão, é uma chave que conecta dois pontos de um mesmo circuito ao ser pressionado. Neste primeiro exemplo ele tem a função de ligar um LED ao ser pressionado e desligar o mesmo ao ser pressionado uma segunda vez. O código para que isso seja possível é o seguinte:

```

1 //pushbutton porém liga ao apertar o botão e desliga ao apertar novamente
2 int led = 2;
3 int botao = 3;
4 int press = 0;
5 int ligado = 0;
6
7 void setup()
8 {
9     pinMode(led, OUTPUT);
10    pinMode(botao, INPUT); //recebe informação do botão
11 }
12
13 void loop(){
14
15     press = digitalRead(botao);
16
17     if(press == HIGH){
18         delay(100);
19         switch(ligado){
20             case 0:
21                 digitalWrite(led, HIGH);
22                 ligado = 1;
23                 break;
24             case 1:
25                 digitalWrite(led, LOW);
26                 ligado = 0;
27                 break;
28         }
29     }
30 }
31 }

```

Figura 11: Código da atividade 4

### 2.4.5 Arduino 5 – Potenciômetro

Na quinta atividade será utilizado um potenciômetro para controle de brilho de uma LED. Para execução deste programa foi necessário uma placa Arduino, um LED, cinco *jumers* macho/macho, uma *protoboard*, um resistor de 220 ohms e um potenciômetro de 10k ohms.

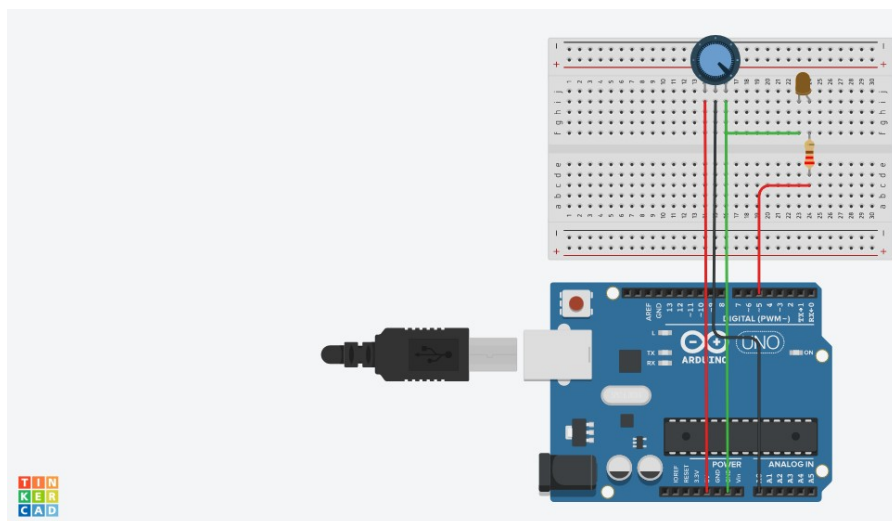


Figura 12: Representação da placa Arduino Uno da atividade 5

O potenciômetro é um componente eletrônico que regula sua resistência elétrica. Neste contexto, ele foi utilizado para diminuir ou aumentar o brilho do LED ao ser regulado. Este é o código para realização do programa:

```

1
2  int led = 5; //tem que ser pwm por ser analogWrite
3  int val_pot = 0; //valor potenciometro/posição
4  int brilho = 0;
5
6  void setup(){
7      pinMode(led, OUTPUT);
8
9      }
10
11 void loop(){
12     val_pot = analogRead(A0);
13     brilho = map(val_pot, 0, 1023, 0, 255); /*pega o valor que
14     está entre 0 e 1023 e converte para 0 a 255*/
15     analogWrite(led, brilho);
16
17 }
```

Figura 13: Código da atividade 5

### 2.4.6 Arduino 6 – Display de sete segmentos

O sexto programa a ser executado na plataforma resumiu-se em um contador de números, do valor zero até o nove, em um *display* de sete segmentos. Para efetivação do programa foi necessário uma placa Arduino, um *display* de sete segmentos, um resistor de 300 ohms, um resistor de 220 ohms e sete *jumpers* macho/macho.

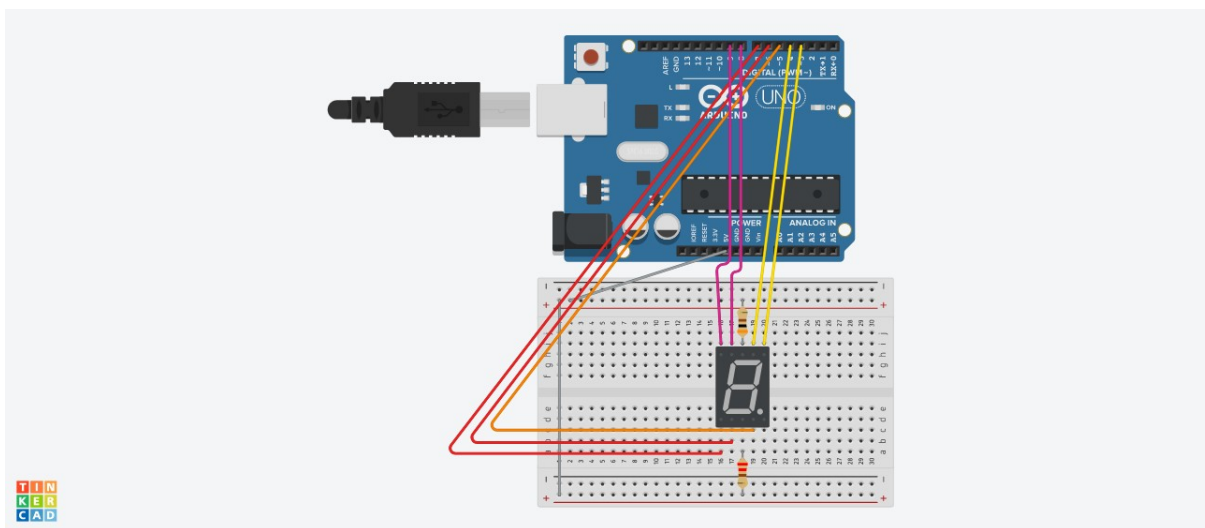


Figura 14: Representação da placa Arduino Uno da atividade 6

Cada um dos *jumpers* é utilizado para conectar-se a um pino diferente, de modo que seja possível controlar qual LED do *display* estará ligada ou desligada. O código de execução é o seguinte:

```

1  int a = 4, b = 3, c = 5, d = 6, e = 7, f = 8, g = 9;
2  int num[10][7]{ //matriz (array)
3    {a,b,c,d,e,f}, //zero
4    {b,c}, //um
5    {a,b,e,d,g}, //dois
6    {a,b,c,d,g}, //três
7    {b,c,f,g}, //quatro
8    {a,c,d,f,g}, //cinco
9    {a,c,d,e,f,g}, //seis
10   {a,b,c}, //sete
11   {a,b,c,d,e,f,g}, //oito
12   {a,b,c,f,g}, //nove
13 };
14
15 void setup(){
16   pinMode(a, OUTPUT);
17   pinMode(b, OUTPUT);
18   pinMode(c, OUTPUT);
19   pinMode(d, OUTPUT);
20   pinMode(e, OUTPUT);
21   pinMode(f, OUTPUT);
22   pinMode(g, OUTPUT);
23 }
24
25 void loop()
26 {
27   for(int i = 0; i < 10; i++){
28     apaga();
29     numero(i);
30     delay(1000);
31   }
32 }
33
34 void apaga(){
35   digitalWrite(a, HIGH);
36   digitalWrite(b, HIGH);
37   digitalWrite(c, HIGH);
38   digitalWrite(d, HIGH);
39   digitalWrite(e, HIGH);
40   digitalWrite(f, HIGH);
41   digitalWrite(g, HIGH);
42 }
43
44 void numero(int n){
45   for(int i = 0; i < 7; i++) digitalWrite(num[n][i], LOW);

```

Figura 15: Código da atividade 6

### 2.4.7 Configuração da IDE para ESP32

A instalação da IDE do Arduino é simples, basta apenas ir até o site oficial do Arduino e baixar a versão desejada na página de software. Porém, para utilização de diferentes placas, antes se faz necessário uma configuração.

A configuração para a placa LoRa ESP32 V2 da Heltec pode ser feita a partir de dois métodos: por meio do Git e pelo Gerenciador de Placas do Arduino, sendo o último o método a ser utilizado.

Primeiramente é necessário baixar o suporte do Arduino para ESP32, isso será feito indo até a barra de tarefas na parte superior esquerda da interface, clicando em Arquivos e então Preferências.

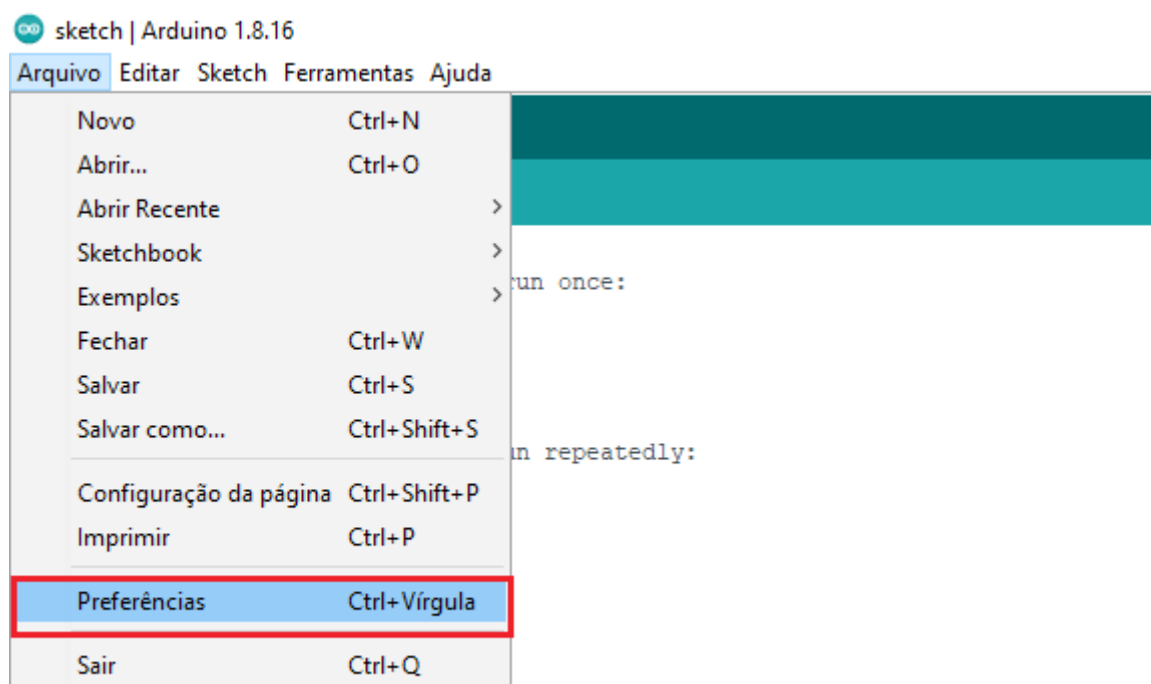


Figura 16: Barra de arquivos do Arduino

Dentro das preferências, é inserido no campo URLs Adicionais para Gerenciadores de Placas a URL do GitHub no qual se encontra o pacote do ESP32.

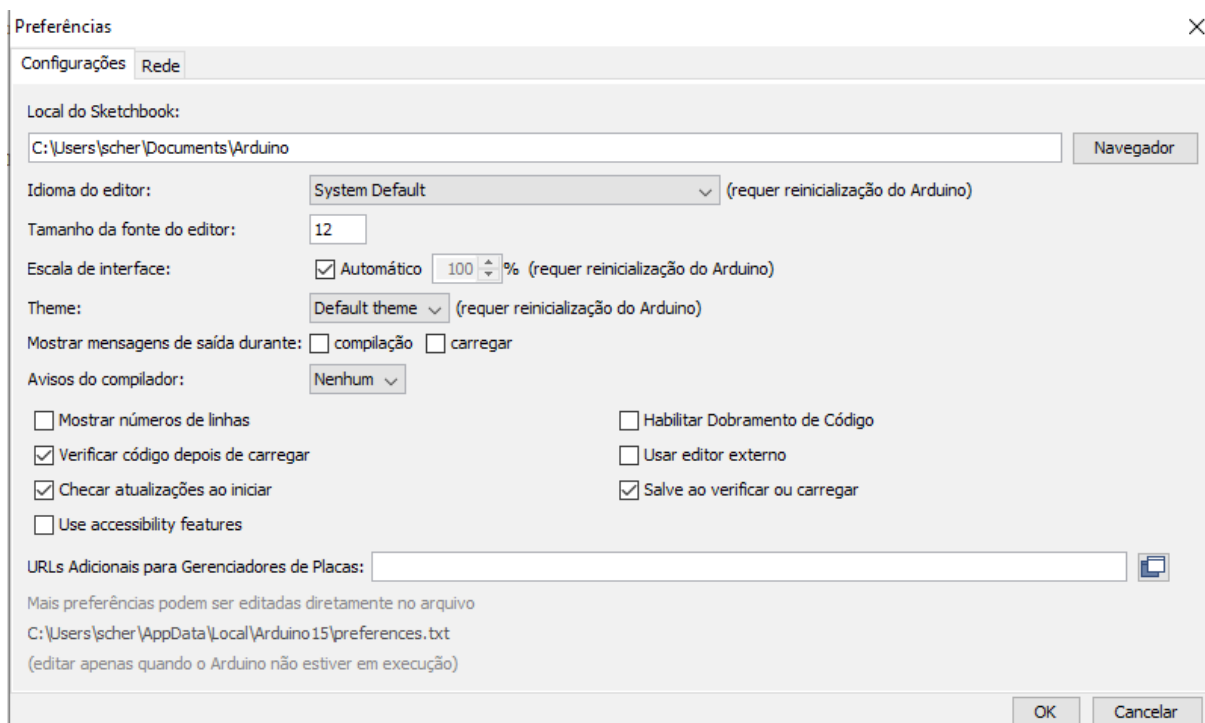


Figura 17: Preferências dos Arduino

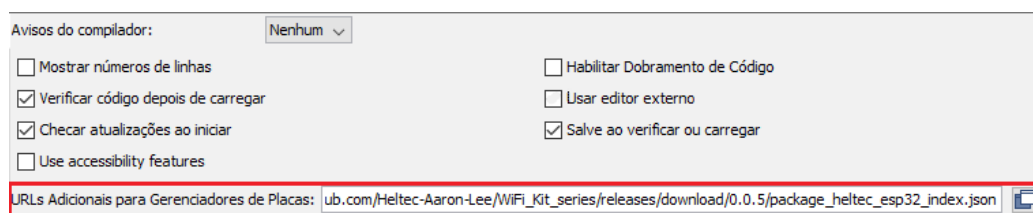


Figura 18: Zoom nas preferências do Arduino

Em seguida, a estrutura do ESP32 será baixada. Isso é feito indo até as Ferramentas, depois Placa: (nome da placa) e então Gerenciar Placas.

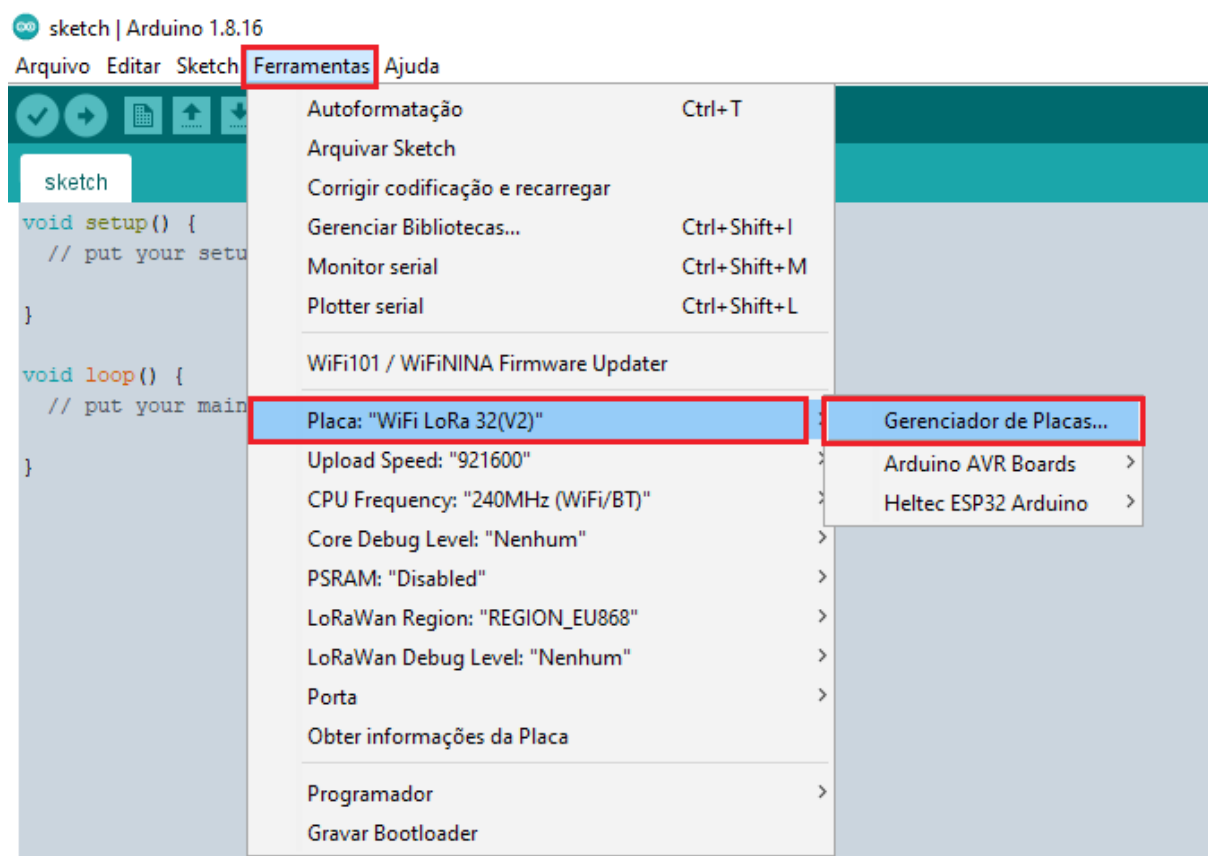


Figura 19: Barra de ferramentas do Arduino.

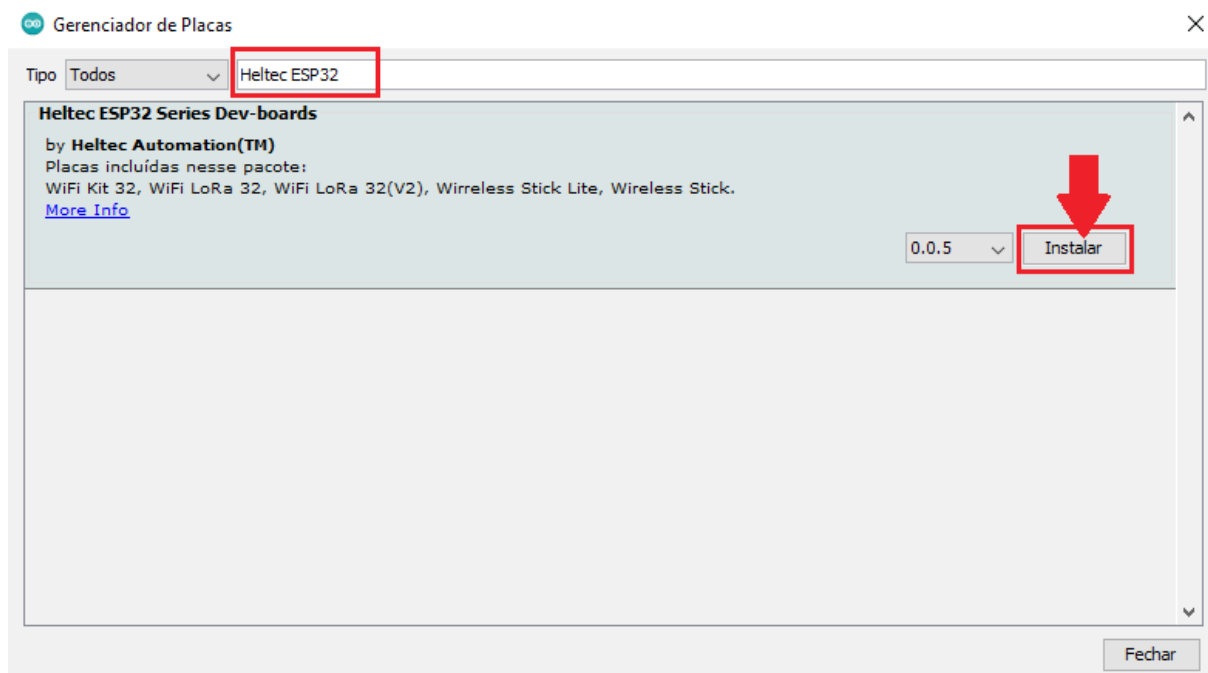


Figura 20: Gerenciado de placas do Arduino

Por fim, é necessário baixar a biblioteca da Heltec ESP32, indo até o Gerenciar Bibliotecas, dentro do Sketch.

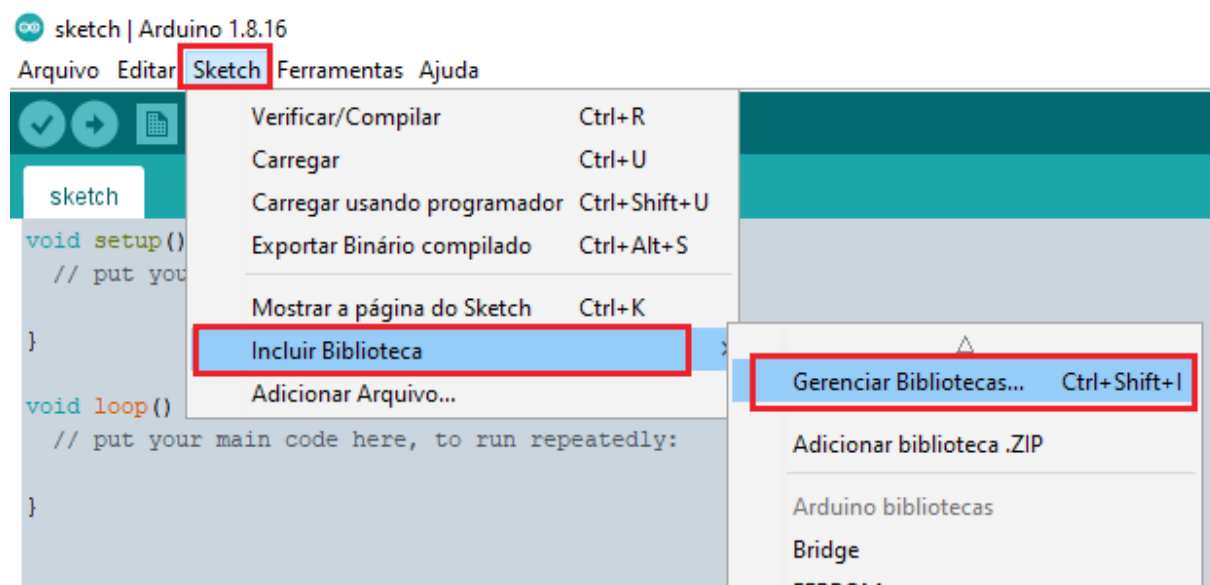


Figura 21: Barra de sketch do Arduino

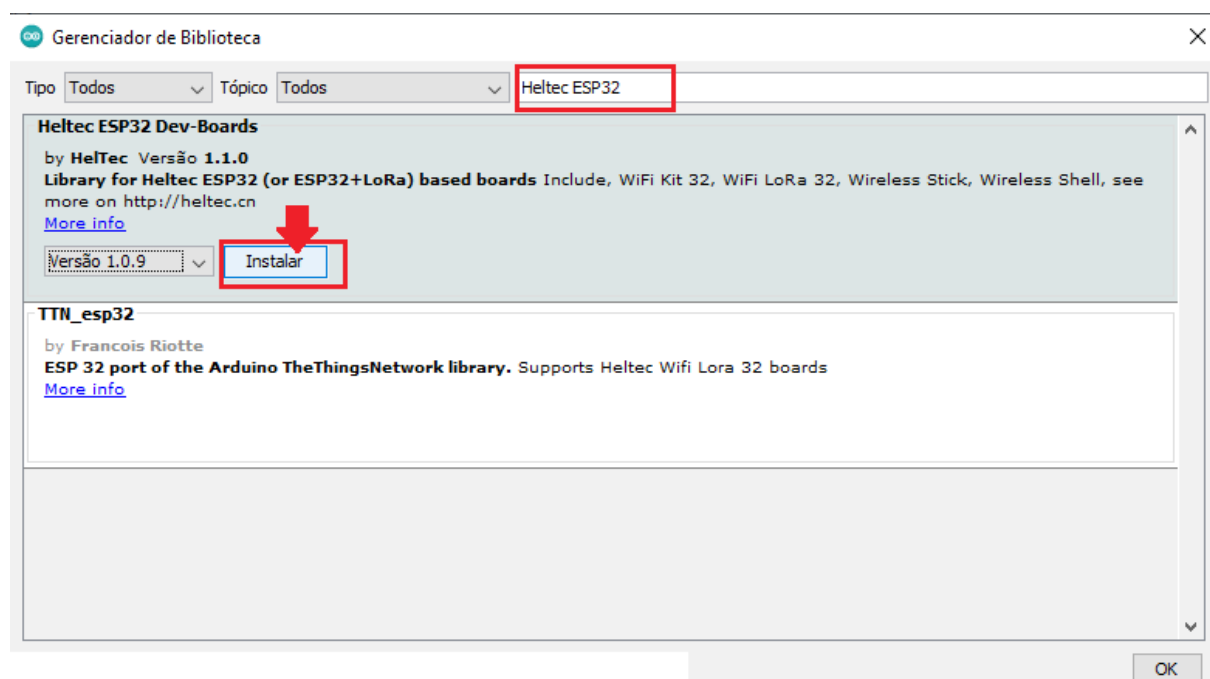


Figura 22: Gerenciador de bibliotecas do Arduino

### 2.4.8 ESP32 1 – Comunicação com WEB

Com a configuração da IDE pronta, finalmente é possível realizar as atividades com a placa física. Ambas as atividades realizadas foram sugestões do professor orientador, a primeira delas é a comunicação da placa com um servidor WEB.

Apesar da mesma lógica de programação ser utilizada por ambos Arduino e ESP32, a inclusão da biblioteca Heltec na configuração da IDE traz algumas funcionalidades novas que só podem ser utilizadas em uma placa Heltec. Além disso, a Heltec também dispõe de uma série de exemplos e programas básicos prontos. Em vista disso, o código a seguir foi construído tendo como base alguns o código já pronto da Heltec denominado *Simple Wifi Server*.



```
esp32WEB - ESP32WEB.ino | Arduino 1.8.16
Arquivo Editar Sketch Ferramentas Ajuda

ESP32WEB

#include <WiFi.h>
#include "heltec.h"

const char* ssid    = "Nome"; //define nome da rede wifi
const char* password = "Senha"; //define senha da rede wifi

WiFiServer server(80);

void setup()
{
    Serial.begin(115200);
    Heltec.begin(true /*DisplayEnable Enable*/, true /*LoRa Enable*/, true /*Serial Enable*/); //inicializa a placa
    Heltec.display->clear(); //limpa tela display
    delay(10);

    Serial.print("Conectando-se a ");
    Serial.println(ssid); //mostra rede

    WiFi.begin(ssid, password); //inicializa wifi

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("WiFi conectado. Endereço de IP:");
    Serial.println(WiFi.localIP()); //mostra endereço ip

    server.begin(); //inicia o server
```

Figura 23: Primeira parte do código de conexão WEB

Primeiramente são incluídas as bibliotecas de WiFi e da Heltec para configuração do WiFi e funcionamento da placa, respectivamente. Em seguida, através de duas constantes char é armazenado o nome e a senha do WiFi que será utilizado para comunicação com o servidor WEB. Então o servidor será criado na porta oitenta.





```

esp32WEB - ESP32WEB.ino | Arduino 1.8.16
Arquivo Editar Sketch Ferramentas Ajuda

ESP32WEB

void loop() {

  //Código exemplo do IDE para wifi server com algumas modificações
  WiFiClient client = server.available(); // procura por clientes

  if (client) {
    Serial.println("Novo Cliente."); // se houver um cliente
    String currentLine = ""; // ele vai informar no monitor serial
    while (client.connected()) { // e criar uma string para armazenar dado ele
      if (client.available()) { // enquanto o cliente estiver conectado
        char c = client.read(); // se houver alguma informação do cliente ela será lida
        Serial.write(c); // e então escrita no monitor serial
        if (c == '\n') {

          // ao final do pedido do cliente a seguinte mensagem será escrita
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            client.print("<br> Clique em <a href=\"/ON\"> ligado</a>, para escrever ON no display.");
            client.print("<br>Clique em <a href=\"/OFF\"> desligado</a>, para escrever OFF no display.");

            client.println();

            break;

```

Figura 24: Segunda parte do código de conexão WEB

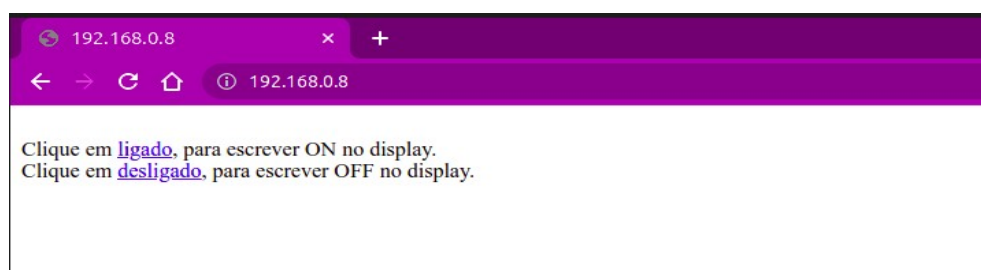


Figura 25: Página WEB criada para a atividade.

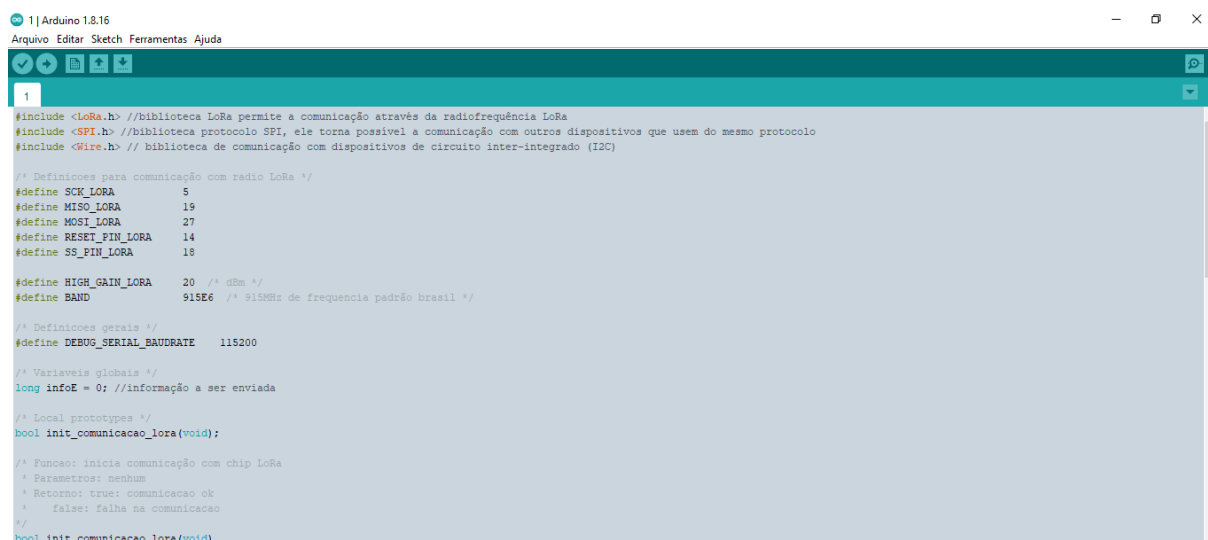
Logo no início do `setup()` é iniciada a placa, mostrando uma mensagem que denota seu funcionamento e então limpa a mesma. No monitor serial será traduzido todo o processo de conexão da rede através da função `Serial.print(texto)` e `Serial.println(variável)`, o servidor será iniciado através da função `server.begin()`.

Dentro da função `loop()`, em suma, será realizada a comunicação com o cliente através de uma página WEB criada a partir de html limitado. O link para acesso desta página é dado pelo endereço de IP do WiFi, do qual pode ser visualizado através do monitor serial

Por fim, o conteúdo mostrado na página se comunica com a placa, de maneira que ao clicar no link escrito *ligado* a placa ESP32 mostrará em sua tela a palavra *ON*, ligado em inglês. No caso do link escrito *desligado* ser clicado, na tela será mostrado a palavra *OFF*, desligado em inglês. Após isso a conexão é desligada e o cliente desconectado.

### 2.4.9 ESP32 2 – Comunicação entre duas placas

A última atividade desenvolvida dentro do estágio é a segunda com utilização do ESP32, se refere a realização de comunicação entre duas placas ESP32 LoRa. Pela comunicação ocorrer entre duas placas é necessário dois códigos para diferenciar quem entrega e quem recebe a informação. O primeiro código se refere a quem entrega a informação, isto é, o emissor.



```

1 | Arduino 1.8.16
Arquivo Editar Sketch Ferramentas Ajuda

1
#include <LoRa.h> //biblioteca LoRa permite a comunicação através da radiofrequência LoRa
#include <SPI.h> //biblioteca protocolo SPI, ele torna possível a comunicação com outros dispositivos que usem do mesmo protocolo
#include <Wire.h> // biblioteca de comunicação com dispositivos de circuito inter-integrado (I2C)

/* Definições para comunicação com radio LoRa */
#define SCK_LORA      5
#define MISO_LORA     19
#define MOSI_LORA     27
#define RESET_PIN_LORA 14
#define SS_PIN_LORA   18

#define HIGH_GAIN_LORA 20 /* dBm */
#define BAND           915E6 /* 915MHz de frequência padrão brasil */

/* Definições gerais */
#define DEBUG_SERIAL_BAUDRATE 115200

/* Variáveis globais */
long infoE = 0; //informação a ser enviada

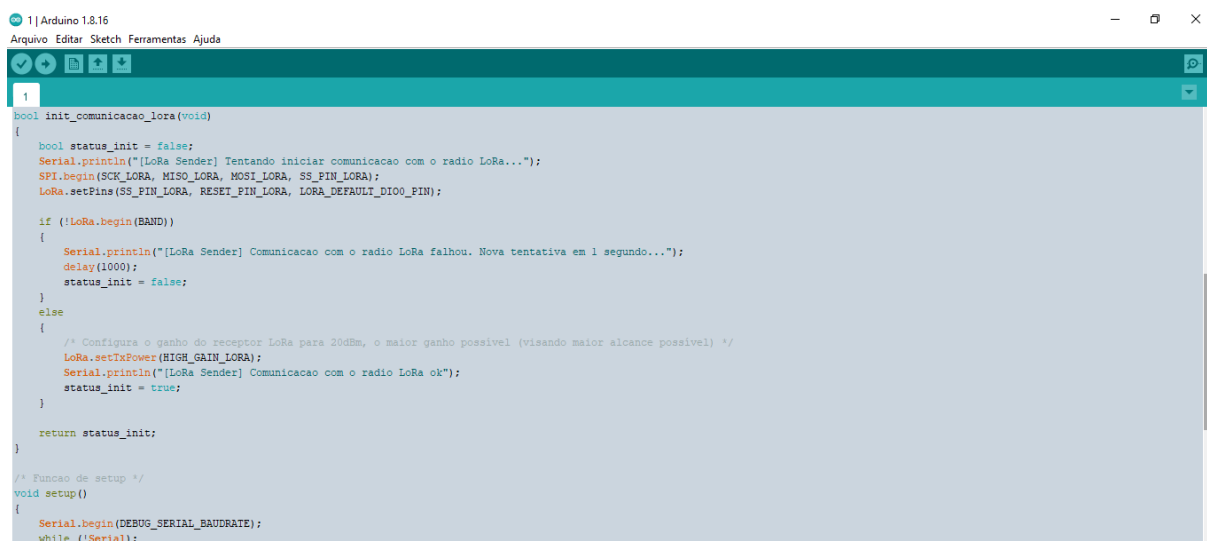
/* Local prototypes */
bool init_comunicacao_lora(void);

/* Funcao: inicia comunicação com chip LoRa
 * Parametros: nenhum
 * Retorno: true: comunicacao ok
 *         false: falha na comunicacao
 */
bool init_comunicacao_lora(void)

```

Figura 26: Primeira parte do código do emissor

Antes de tudo foram incluídas as bibliotecas necessárias para a comunicação e definições gerais para comunicação via radiofrequência por LoRa. A variável referente a informação a ser enviada é definida como *long*, pois a informação é incrementada a cada segundo, sendo assim uma variável extensiva, ou seja, que muda de valor. Isso se deve ao fato da informação a ser enviada ser o RSSI, parâmetro de medição da intensidade do sinal LoRa.



```

1
bool init_comunicacao_lora(void)
{
    bool status_init = false;
    Serial.println("[LoRa Sender] Tentando iniciar comunicacao com o radio LoRa...");
    SPI.begin(SCK_LORA, MISO_LORA, MOSI_LORA, SS_PIN_LORA);
    LoRa.setPins(SS_PIN_LORA, RESET_PIN_LORA, LORA_DEFAULT_DIO0_PIN);

    if (!LoRa.begin(BAND))
    {
        Serial.println("[LoRa Sender] Comunicacao com o radio LoRa falhou. Nova tentativa em 1 segundo...");
        delay(1000);
        status_init = false;
    }
    else
    {
        /* Configura o ganho do receptor LoRa para 20dBm, o maior ganho possivel (visando maior alcance possivel) */
        LoRa.setTxPower(HIGH_GAIN_LORA);
        Serial.println("[LoRa Sender] Comunicacao com o radio LoRa ok");
        status_init = true;
    }

    return status_init;
}

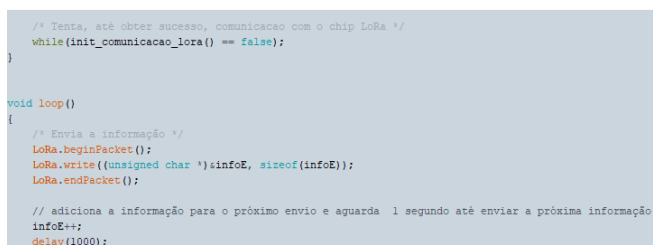
/* Funcao de setup */
void setup()
{
    Serial.begin(DEBUE_SERIAL_BAUDRATE);
    while (!Serial);
}

```

Figura 27: Segunda parte do código do emissor

Para iniciar a comunicação é criada uma variável booleana sem qualquer parâmetro. Em seguida é criada uma função indicando o status da inicialização, começando em *false*, junto a isso uma mensagem será escrita no monitor serial e os protocolos necessários para a comunicação são iniciados, além da função *LoRa.setPins* substituir o padrão de pinos usado. Se o valor da frequência não for igual a frequência definida anteriormente (915hz), uma mensagem de erro será mostrada no monitor serial e o status da inicialização permanecerá *false*. Do caso contrário, isto é, do valor da frequência ser o valor definido, é configurado o ganho do receptor para 20dBm, a comunicação com o rádio LoRa acontece e o status de inicialização muda para *true*.

Na função *setup* realiza-se a tentativa de comunicação com o chip LoRa até que se obtenha sucesso. Por fim, na função *loop* é enviada a informação.



```

/* Testa, até obter sucesso, comunicacao com o chip LoRa */
while (init_comunicacao_lora() == false);
}

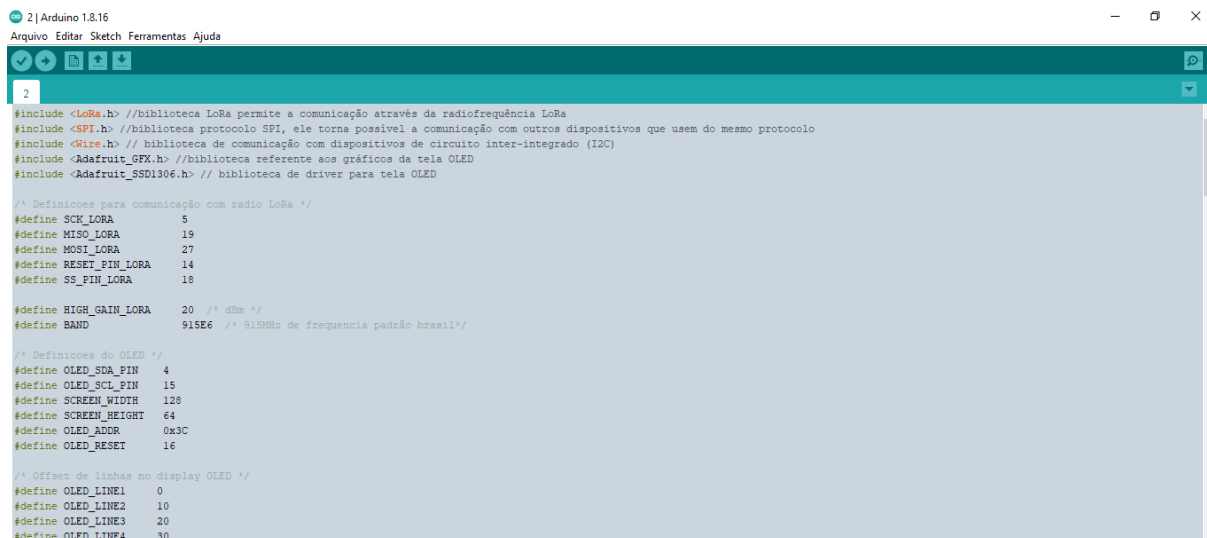
void loop()
{
    /* Envia a informacao */
    LoRa.beginPacket();
    LoRa.write((unsigned char *)infoE, sizeof(infoE));
    LoRa.endPacket();

    // adiciona a informacao para o proximo envio e aguarda 1 segundo até enviar a proxima informacao
    infoE++;
    delay(1000);
}

```

Figura 28: Terceira parte do código do emissor

O segundo código pertence a placa que receberá a mensagem, isto é, a placa receptora.



```

2 | Arduino 1.8.16
Arquivo Editar Sketch Ferramentas Ajuda

#include <LoRa.h> //biblioteca LoRa permite a comunicação através da radiofrequência LoRa
#include <SPI.h> //biblioteca protocolo SPI, ele torna possível a comunicação com outros dispositivos que usem do mesmo protocolo
#include <Wire.h> // biblioteca de comunicação com dispositivos de circuito inter-integrado (I2C)
#include <Adafruit_GFX.h> //biblioteca referente aos gráficos da tela OLED
#include <Adafruit_SSD1306.h> // biblioteca de driver para tela OLED

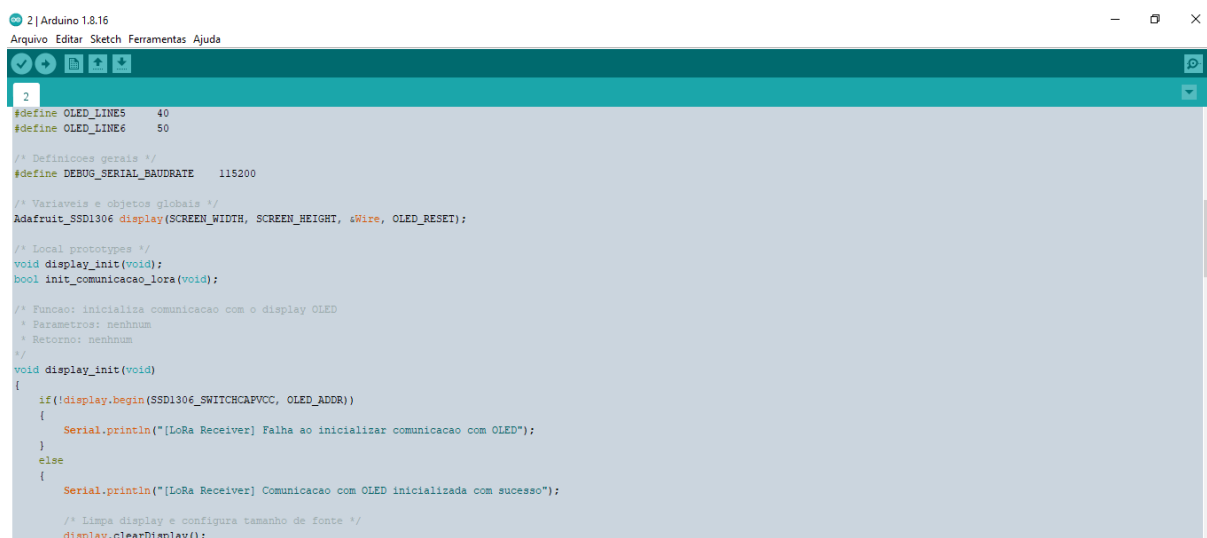
/* Definições para comunicação com radio LoRa */
#define SCK_LORA 5
#define MISO_LORA 19
#define MOSI_LORA 27
#define RESET_PIN_LORA 14
#define SS_PIN_LORA 18

#define HIGH_GAIN_LORA 20 /* dBm */
#define BAND 915E6 /* 915MHz de frequência padrão brasil*/

/* Definições do OLED */
#define OLED_SDA_PIN 4
#define OLED_SCL_PIN 15
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_ADDR 0x3C
#define OLED_RESET 16

/* Offset de linhas no display OLED */
#define OLED_LINE1 0
#define OLED_LINE2 10
#define OLED_LINE3 20
#define OLED_LINE4 30
  
```

Figura 29: Primeira parte do código do receptor



```

2 | Arduino 1.8.16
Arquivo Editar Sketch Ferramentas Ajuda

#define OLED_LINE5 40
#define OLED_LINE6 50

/* Definições gerais */
#define DEBUG_SERIAL_BAUDRATE 115200

/* Variáveis e objetos globais */
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

/* Local prototypes */
void display_init(void);
bool init_comunicacao_lora(void);

/* Função: inicializa comunicação com o display OLED
 * Parâmetros: nenhum
 * Retorno: nenhum
 */
void display_init(void)
{
  if(!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR))
  {
    Serial.println("[LoRa Receiver] Falha ao inicializar comunicação com OLED");
  }
  else
  {
    Serial.println("[LoRa Receiver] Comunicação com OLED inicializada com sucesso");
  }

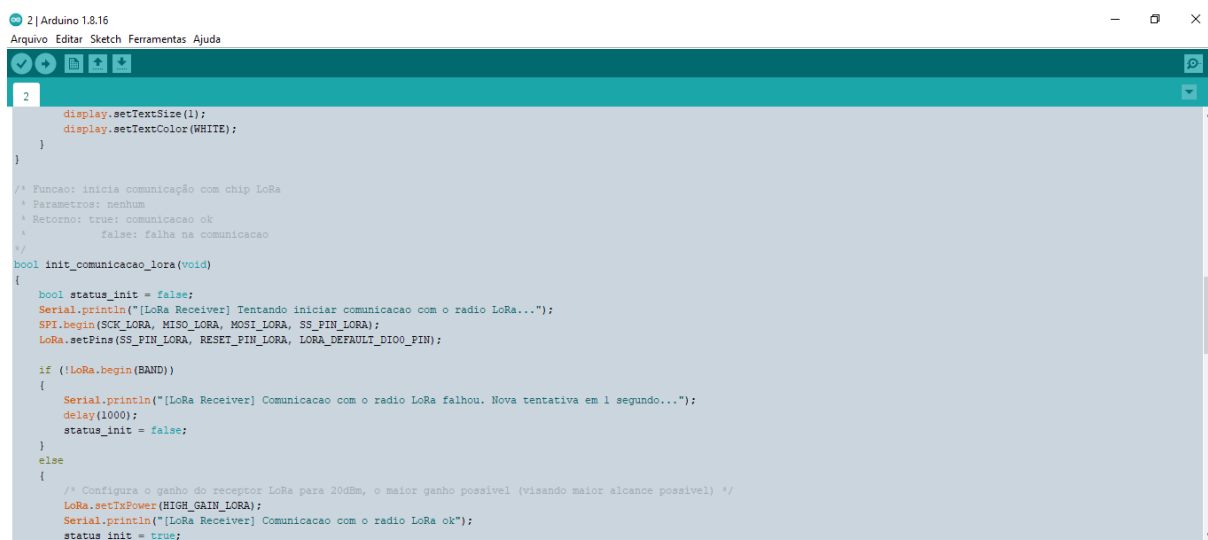
  /* Limpa display e configura tamanho de fonte */
  display.clearDisplay();
  }
  
```

Figura 30: Segunda parte do código do receptor

As primeiras bibliotecas a serem incluídas e suas definições permanecem as mesmas, com exceção as novas bibliotecas que se referem a tela OLED do ESP32 e suas configurações.

A primeira função a ser declarada certifica se há comunicação com a tela OLED da placa, a seguir o display é configurado para logo mais mostrar a informação. A função defini-

da para inicialização da comunicação é idêntica à do emissor. No setup é realizado a configuração do OLED, no qual será escrito uma mensagem inicial de espera, enquanto isso busca-se conexão com o LoRa até obter sucesso.



```

2 | Arduino 1.8.16
Arquivo Editar Sketch Ferramentas Ajuda

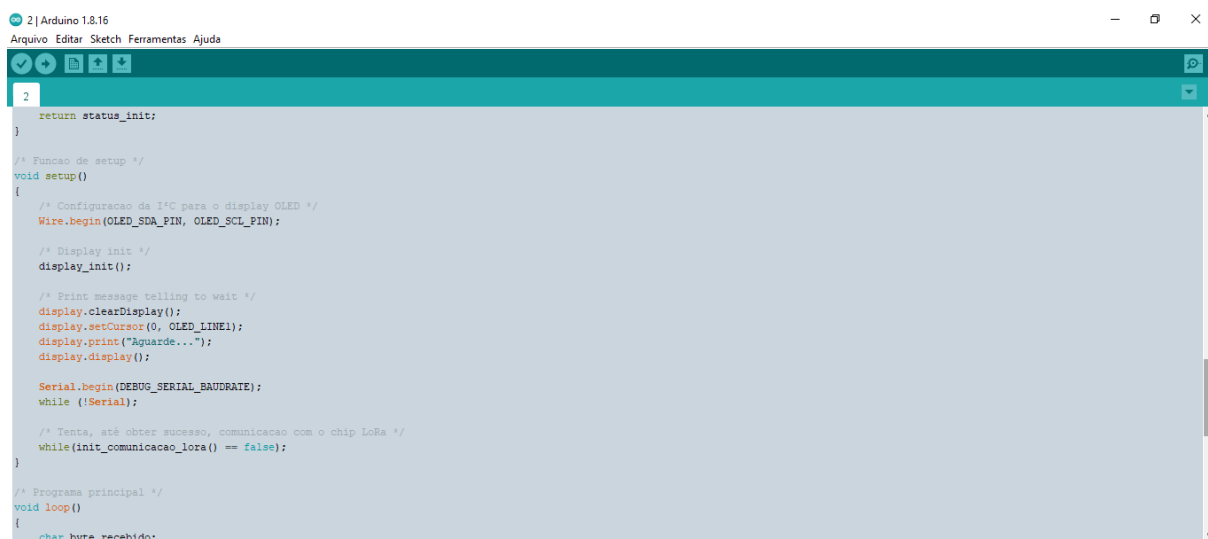
2
    display.setTextSize(1);
    display.setTextColor(WHITE);
}

/* Funcao: inicia comunicacao com chip LoRa
 * Parametros: nenhum
 * Retorno: true: comunicacao ok
 *           false: falha na comunicacao
 */
bool init_comunicacao_lora(void)
{
    bool status_init = false;
    Serial.println("[LoRa Receiver] Tentando iniciar comunicacao com o radio LoRa...");
    SPI.begin(SCK_LORA, MISO_LORA, MOSI_LORA, SS_PIN_LORA);
    LoRa.setPins(SS_PIN_LORA, RESET_PIN_LORA, LORA_DEFAULT_DIO0_PIN);

    if (!LoRa.begin(BAND))
    {
        Serial.println("[LoRa Receiver] Comunicacao com o radio LoRa falhou. Nova tentativa em 1 segundo...");
        delay(1000);
        status_init = false;
    }
    else
    {
        /* Configura o ganho do receptor LoRa para 20dBm, o maior ganho possivel (visando maior alcance possivel) */
        LoRa.setTxPower(HIGH_GAIN_LORA);
        Serial.println("[LoRa Receiver] Comunicacao com o radio LoRa ok");
        status_init = true;
    }
}

```

Figura 31: Terceira parte do código do receptor



```

2 | Arduino 1.8.16
Arquivo Editar Sketch Ferramentas Ajuda

2
    return status_init;
}

/* Funcao de setup */
void setup()
{
    /* Configuracao da I²C para o display OLED */
    Wire.begin(OLED_SDA_PIN, OLED_SCL_PIN);

    /* Display init */
    display_init();

    /* Print message telling to wait */
    display.clearDisplay();
    display.setCursor(0, OLED_LINE1);
    display.print("Aguarde...");
    display.display();

    Serial.begin(DEBUG_SERIAL_BAUDRATE);
    while (!Serial);

    /* Tenta, até obter sucesso, comunicacao com o chip LoRa */
    while (init_comunicacao_lora() == false);
}

/* Programa principal */
void loop()
{
    char byte recebido;
}

```

Figura 32: Quarta parte do código do receptor

Na função principal é verificado se há informação, se existente ela será recebida pelo receptor conforme os protocolos. Finalmente, informação será escrita na tela OLED.

```

2 | Arduino 1.8.16
Arquivo Editar Sketch Ferramentas Ajuda

2
int packet_size = 0;
int lora_rssi = 0;
long infoR = 0;
char * ptInformacaoRecebida = NULL;

/* Verifica se chegou alguma informação do tamanho esperado */
packet_size = LoRa.parsePacket();

if (packet_size == sizeof(informacao_recebida))
{
    Serial.print("[LoRa Receiver] Há dados a serem lidos");

    /* Recebe os dados conforme protocolo */
    ptInformacaoRecebida = (char *)infoR;
    while (LoRa.available())
    {
        byte_recebido = (char)LoRa.read();
        *ptInformacaoRecebida = byte_recebido;
        ptInformacaoRecebida++;
    }

    /* Escreve RSSI de recepção e informação recebida */
    lora_rssi = LoRa.packetRssi();
    display.clearDisplay();
    display.setCursor(0, OLED_LINE1);
    display.print("RSSI: ");
    display.println(lora_rssi);
    display.setCursor(0, OLED_LINE2);
    display.print("Informacao: ");
    display.setCursor(0, OLED_LINE3);
    display.println(informacao_recebida);
    delay(100);
}

```

Figura 33: Quinta parte do código do receptor.

### 3. Dificuldades

O estágio foi um período de acúmulo de conhecimento e amadurecimento constante. Este resultado só foi possível através de muitos erros cometidos e dificuldades enfrentadas ao longo de sua duração, sendo o anteriormente citado muitas vezes resultado destes erros.

A pandemia foi um dos agentes que promoveu a maioria das dificuldades enfrentadas durante o percurso do estágio. Além da adaptação constante enfrentada devido às restrições resultantes da pandemia, também houve mudança de dinâmicas, como o ensino presencial sendo substituído pelo ensino remoto.

Tal adaptação que trocou sala de aula físicas por chamadas online que simulavam escolas, acabou por trazer outra dificuldade que teve um impacto direto no estágio: a autogestão de tempo. Durante o período presencial, mesmo se tratando de um estágio, é necessário o cumprimento de afazeres e atividades dentro do horário, e ao fim deste, voltar para casa - o tempo sendo administrado indiretamente pela escola, mas o modelo remoto repassou esta responsabilidade para o estudante. O tempo de aula estabelecido para conclusão de atividades e/ou funções a serem realizadas e a forma como este tempo é usado passa a ser de obrigação total do aluno.

O ensino online abre espaço para muitas distrações que podem afetar o desempenho durante o estágio ou até mesmo auxiliar o mesmo. Um dos obstáculos foi estabelecer um perí-

odo de tempo exato diário a cumprir o estágio, pois apesar do horário do estágio ser determinado antes, o estágio online abre brecha para que este horário mude. Havia dias de trabalho de duas horas manhã e duas horas tarde, e outros dias de oito horas diárias de trabalho.

#### **4. Sugestões**

A grade curricular do Curso Técnico de Informática para Internet não é capaz de incorporar todo conteúdo existente de cada área relacionada à informática no curso - e nunca será capaz, visto que a internet é um campo de conhecimento vasto e em constante mudança. Contudo, é necessário adicionar componentes que possam trabalhar de maneira conjunta a outros cursos, como por exemplo ensinar conceitos básicos de eletrônica para alunos do curso de Informática para Internet ou conceitos básicos de programação para alunos do curso de mecânica. Apesar de constituírem grades curriculares diferentes entre si, os cursos técnicos eventualmente se cruzam em suas disciplinas, podendo aplicar conceitos de uma área a outra.

Além disso, também seria interessante o incentivo aos alunos para que utilizem-se de ferramentas de software livre e código aberto, de modo a ampliar seus conhecimentos através da interação direta com programas e/ou aqueles que também o utilizam. Desta maneira também se amplia uma perspectiva mais inclusiva e criativa no que diz respeito ao uso de softwares, sendo passível de adaptá-los para outros fins, por exemplo.

#### **5. Conclusão**

Desde o surgimento da oportunidade até sua conclusão, o estágio foi repleto de obstáculos dos mais variados tipos, sendo alguns deles uma pandemia, o ambiente de trabalho ou os recursos limitados, além de dificuldades resultantes destes mesmos obstáculos. Ainda sim, por mais difíceis que tenham sido, todos foram superados - ou vêm sendo superados, no caso da pandemia - da melhor forma possível.

O estágio para sempre ficará marcado como uma fase de crescimento pessoal em todos os sentidos, acadêmico, profissional ou individual, sendo uma parte importante de autoconhecimento e preparo para um futuro, talvez, promissor.

## 6. Referências

TEIXEIRA FRANCISCATO, Fábio. Hardware multi sensor com comunicação a longas distâncias para medições de fenômenos em ambientes remotos. p. 4-16, jul. 2021.

SANTOS, Bruno; SILVA, Lucas; CELES, Clayson;*et al.* Capítulo 1 Internet das Coisas: da Teoria à Prática. Disponível em: <<https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf>> Acesso em 2022.

DE OLIVEIRA, Sérgio. Internet das Coisas: com ESP8266, Arduino e Raspberry Pi 2a edição: Atualizado para ESP32. 2. ed. Novatec Editora, 2021. 312 p. Disponível em: <<https://play.google.com/books/reader?id=rnojEAAQBAJ&pg=GBS.PP1&hl=pt>>. Acesso em 2022.

BERTOLETI, Pedro. Projetos com ESP32 e LoRa. Editora NCB, 2019. 214 p. Disponível em: <[https://books.google.com.br/books/about/Projetos\\_com\\_ESP32\\_e\\_LoRa.html?id=fnCiDwAAQBAJ&source=kp\\_book\\_description&redir\\_esc=y](https://books.google.com.br/books/about/Projetos_com_ESP32_e_LoRa.html?id=fnCiDwAAQBAJ&source=kp_book_description&redir_esc=y)> Acesso em 2022.

Arduino Reference - Arduino Reference. Arduino.cc. Disponível em: <<https://www.arduino.cc/reference/en/>>. Acesso em 2022.

Circuit design 1 - Programa Teste 01 | Tinkercad. Tinkercad. Disponível em: <<https://www.tinkercad.com/things/a8o60Gqw5RX-1-programa-teste-01>> Acesso em 2022.

Circuit design 2 - Programa teste 02 | Tinkercad. Tinkercad. Disponível em: <<https://www.tinkercad.com/things/dV4J2G2f2IT-2-programa-teste-02>>. Acesso em 2022.

Circuit design 3 - Programa teste 03 | Tinkercad. Tinkercad. Disponível em: <<https://www.tinkercad.com/things/gAGwbZ33WZP-4-programa-teste-04>>. Acesso em 2022.

Circuit design 4 - Programa Teste 04 | Tinkercad. Tinkercad. Disponível em: <<https://www.tinkercad.com/things/gAGwbZ33WZP-4-programa-teste-04>>. Acesso em 2022.

Circuit design 5 - Programa teste 05 | Tinkercad. Tinkercad. Disponível em: <<https://www.tinkercad.com/things/dHxz6nnlGGy-5-programa-teste-05>>. Acesso em 2022.

Circuit design 6 - Programa Teste 06 | Tinkercad. Tinkercad. Disponível em: <<https://www.tinkercad.com/things/csTMnJOJCcB-6-programa-teste-06>>. Acesso em 2022.

SCHERER DE OLIVEIRA, Ana Caroline. GitHub - acsoscher/Estagio: Códigos utilizados no estágio. Inclui Arduino e ESP32. GitHub. Disponível em: <<http://github.com.acsoscher/Estagio>>. Acesso em 2022.

Heltec ESP32+Arduino Series Quick Start — Heltec Automation Docs V0.0.1 documentation. Readthedocs.io. Disponível em: <[https://heltec-automation-docs.readthedocs.io/en/latest/esp32%2Barduino/quick\\_start.html](https://heltec-automation-docs.readthedocs.io/en/latest/esp32%2Barduino/quick_start.html)>. Acesso em 2022.



