

Beacons

Os Beacons são aparelhos de proximidade que emitem informações, por meio da tecnologia bluetooth, diretamente aos smartphones através de aplicativo ou diretamente para o sistema operacional dos dispositivos no caso do google esse sistema nativo se chama nearby, já a empresa Apple o sistema dela se chama iBeacon. A ideia, que promete revolucionar o mercado e fortalecer a chamada Internet das Coisas, é permitir a interação mais rápida de possíveis clientes com seus interesses.

Kevin Ashton, pesquisador britânico do Massachusetts Institute of Technology (MIT), é considerado o primeiro especialista a usar o termo “Internet das Coisas (IoT, na sigla em inglês), em 1999.

A ideia de Ashton era simples, porém inovadora para sua época.

“ Se tivéssemos computadores que soubessem tudo sobre as coisas em geral, seríamos capazes de rastrear e contar tudo, e reduzir bastante o desperdício, a perda e os custos. [...] Precisamos capacitar os computadores com seus próprios meios de coletar informações, para que possam ver, ouvir e cheirar o mundo sozinhos, com toda a sua glória aleatória. “
Essa mensagem foi apresentada por Kevin Ashton, para o Jornal de RFID, deste então o termo IOT foi adotado pela mídia e empresa em todo o segmento de mercado para fazer referência com projetos inovadores e atrativos para os consumidores.

O Beacon é um pequeno dispositivo que utiliza uma tecnologia chamada Bluetooth Low Energy (BLE), que emite um sinal intermitente de ondas de rádio que consegue localizar seu smartphone em um determinado raio, com a vantagem que ele consome menos energia, alcançando uma penetração nas estruturas de concreto, madeira entre outros materiais devido a grande utilização do Beacon em estabelecimentos comerciais.

A história do Bluetooth começa por volta do ano de 1994, quando a empresa Ericsson, em busca de um diferencial, procurou investir no desenvolvimento de uma forma de comunicação entre aparelhos celulares e seus respectivos acessórios. Ela deveria utilizar sinais de rádio que não fossem caros, ultrapassando a ideia da rede por cabos. Esse estudo resultou na criação de um sistema de rádio de curto alcance chamado de MLink, que utilizava uma baixa potência, e, portanto, não consumia tanta energia. Com a possibilidade de uma implementação de um sistema relativamente fácil e barato, a Ericsson percebeu que a ideia do MLink poderia realmente dar certo, ampliando seus investimentos.

No ano de 1997, o promissor projeto atraiu a atenção de outras empresas, por esse motivo, em 1998, houve a criação do Bluetooth SIG (Special Interest Group). As empresas que participaram da sua criação eram consideradas gigantes nas suas áreas (desenvolvimento de chips e unidades de processamento, telecomunicações e produção de computadores), dando uma dimensão de valor a tecnologia que estava sendo aperfeiçoada. São elas: Intel, Ericsson,

IBM, Toshiba e Nokia. Com tantos setores diferentes influenciando o processo de aperfeiçoamento da tecnologia, permitiu-se a interoperabilidade e o uso dela nos mais variados tipos de aparelhos, não se restringindo somente aos telefones celulares.

O Bluetooth Special Interest Group (SIG) se traduziu numa organização privada e sem fins lucrativos. Ela não fabrica e nem vende produtos com a tecnologia Bluetooth, apenas trabalha continuamente no desenvolvimento da tecnologia de redes sem a utilização de fios para implementação desta nos produtos fabricados pelos membros do grupo. As principais tarefas do SIG são proteger a marca Bluetooth, publicar especificações, administrar a qualificação do programa e espalhar a idéia da tecnologia de comunicação sem fios.

O SIG implementou o The Generic Attributes (GATT) são conjuntos de características e relacionamentos com outros serviços que encapsulam o comportamento de parte de um dispositivo Bluetooth Low Energy (BLE) . Um perfil do GATT descreve um caso de uso, funções e comportamentos gerais baseados na funcionalidade do GATT, permitindo uma ampla inovação, mantendo a interoperabilidade total com outros dispositivos Bluetooth, Basicamente uma subdivisão do SIG.

O BLE pode ser entendida como uma especificação, ou funcionalidade que integra o Bluetooth desde 2010, a partir de sua versão 4.0. Vantagens são Baixo consumo de energia, múltiplas conexões, baixo custo de implementação, conectividade como aparelho antigos entre outros <https://www.bluetooth.com/specifications/gatt> .

O iBeacons é o protocolo criado pela Apple para a interagir com os beacons foi anunciado pela primeira vez no ano de 2013 na conferência worldwide developers conference, foi a forma encontrada pela apple de trazer mais contexto para de interação com seu sistema. Este permite a todo o tipo de smartphones e tablets (Android e iOS) desencadear determinadas ações quando se encontrarem na proximidade de um beacon. Estabelece ainda um conjunto de dados que cada beacon deve enviar.

iBeacon

<https://developer.apple.com/ibeacon/>

<https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>

Os iBeacons contém uma combinação de letras e números, divididos em grupos específicos. Cada código é único para cada sinalizador, e um aplicativo móvel só agirá quando reconhecer os dados relacionados a esse sinalizador. Uma vez que um beacon é detectado por um aplicativo, algum tipo de ação é acionado: um alerta de push para a tela inicial, um prompt para registrar algo no telefone, conectar-se a um servidor e assim por diante.

Característica do iBeacon seguintes parâmetros:

1. iBeacon prefix

- 1.1. O iBeacon prefix consiste no parâmetro de menor importância contendo a seguinte informação:
 - 1.1.1. Flags de identificação do tipo de anúncio, por exemplo:
 - 1.1.2. O bit 0 (OFF) LE Limited Discoverable Mode
 - 1.1.3. O bit 1 (ON) LE General Discoverable Mode
- 1.2. Número de bytes totais do pacote;
- 1.3. Identificação do fabricante do beacon: Este contém informações que permitem ao dispositivo receptor identificar o conteúdo que segue no restante pacote.

2. Proximity UUID;

- 2.1. A organização do UUID consiste em 32 dígitos hexadecimais, divididos em 5 grupos, separados por hífens e devem ser parecidos com isto:

UUID	D9B9EC1F-3925-43D0-80A9-1E39D4CEA95C
-------------	---

- 2.2. Cada um dos 5 grupos deve conter o seguinte número de caracteres por seção:

Primeira seção: 8 | Segunda seção: 4 | Terceira seção: 4 | Quarta seção: 4 | Quinta seção: 12

- 2.3. Os caracteres devem ser números de 0 a 9, ou letras de A a F. Um grupo pode ser composto apenas de números ou letras ou uma combinação de ambos.

3. Major;

- 3.1. Consiste num identificador de 2 bytes usado para agrupar um conjunto de beacons. Por exemplo, todos os beacons existentes dentro de uma determinada loja devem possuir o mesmo valor Major. Assim, a aplicação que comunicar com o beacon consegue identificar em que loja se encontra o cliente.

Localização do mercado		Baharmas Benfica	Baharmas São Pedro	Baharmas Teixeira
UUID		f7826da6-4fa2-4e98-8024-bc5b71e0893e		
Minor	Major	1	2	3
	Congelados	10	10	10
	Limpeza	20	20	20
	Bebidas	30	30	30

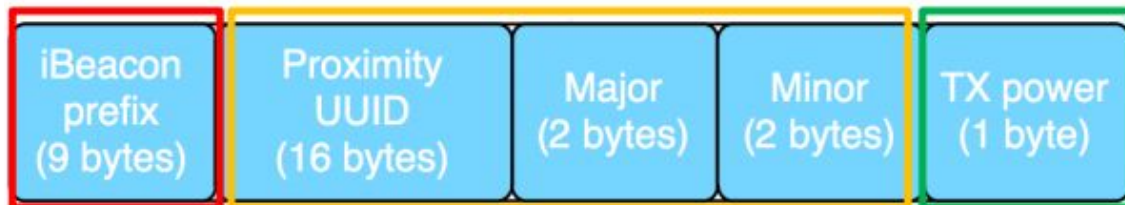
4. **Minor;**

- 4.1. Consiste num identificador de 2 bytes usado para identificar cada beacon, devendo possuir um valor único em cada loja (para o mesmo valor de Major). Assim, é possível identificar exatamente em que local se encontra um determinado cliente

5. **TX Power.**

- 5.1. Potência de Transmissão e o Intervalo de sinal (com que frequência os seus beacons enviam um sinal)
- 5.2. Sua configuração de Potência de Transmissão determina quão poderosamente o sinal será transmitido pelo seu beacon. Isso é medido em dBm (decibel-milliwatts) e corresponde a uma classificação numérica (de 0 a 7) que você ser alterada por meio do aplicativo Admin (onde 0 é o menos potente - 7 é o mais poderoso). Como você pode esperar, tornar sua transmissão mais poderosa aumentará o alcance de seu sinal. Também é verdade que quanto mais poderosa for a sua transmissão, maior será o consumo de energia e, conseqüentemente, menor a duração da bateria.

Dado a importância que cada elemento ocupa no modo de funcionamento de um iBeacon, é possível agrupá-los em três componentes distintos.



Limitações do iBeacon

- Existem duas regras de modo a que os dispositivos consigam receber e interpretar anúncios iBeacon, nomeadamente:
- Uso obrigatório do Bluetooth;
- Uso obrigatório de uma aplicação.
- Informar o cliente da necessidade de ligar o Bluetooth;

No que diz respeito ao primeiro ponto, o cliente deve ser informado que, para usufruir de uma determinada vantagem, o Bluetooth do dispositivo deve-se encontrar ligado. Essa informação deve ser apresentada diretamente na aplicação, e sobretudo em espaços físicos, devendo ser afixados placards de forma a funcionar como um lembrete para o cliente (à semelhança de como é apresentado atualmente para zonas Wifi grátis)

De modo a que o cliente descarregue a aplicação e faça uso dela é necessário que o vendedor lhe introduza valor acrescentado e transmita ao cliente as vantagens que o seu uso lhe traz. Devem ser feitas campanhas de marketing e publicidade de modo a divulgar a aplicação e direcionar o cliente para o seu uso, pois só assim irá obter resultados da tecnologia iBeacon. Devem ser criadas ainda funcionalidades que, dependendo de cada caso, criem valor para além da receção de descontos, como por exemplo, conseguir definir uma lista de compras. Com o crescimento exponencial da IoT essas limitações seja deixada vencidas ou minimizadas

Eddystone

<https://github.com/google/eddystone>

<https://developers.google.com/beacons/eddystone>

O Eddystone consiste num protocolo de comunicação open-source software, sobre a licença Apache, desenvolvido pela Google e apresentado em julho de 2015. O grande objetivo do seu desenvolvimento consistiu em colmatar uma das limitações da tecnologia iBeacon: uso obrigatório de uma aplicação, introduzindo o conceito de Physical Web. Teve ainda como objetivo melhorar a gestão de beacons à distância. Para isso, a norma Eddystone estabelece três tipos de pacotes: **Eddystone-UID**, **Eddystone-URL** e **Eddystone-TLM**.

Enquanto que o identificador de um iBeacon é composto por três partes: UUID, Major e Minor (com um tamanho total de 20 bytes), um **Eddystone-UID** é composto por duas: Namespace (10 bytes) e Instance (6 bytes).

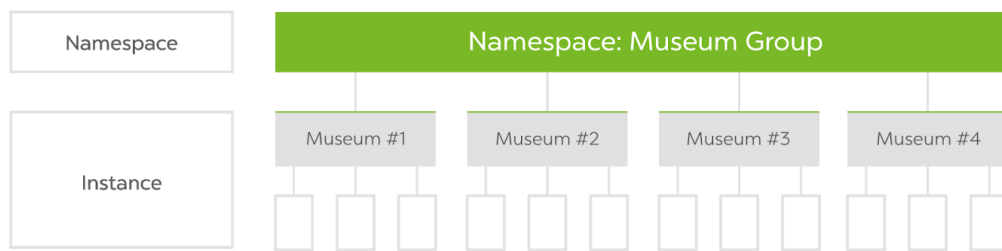
O **parâmetro Namespace** é semelhante ao parâmetro UUID da tecnologia iBeacon, sendo usado para distinguir beacons entre organizações.

iBeacon **UUID** B0CA750-E7A7-4E14-BD99-095477CB3E77

Eddystone-UID 8B0CA750095477CB3E77

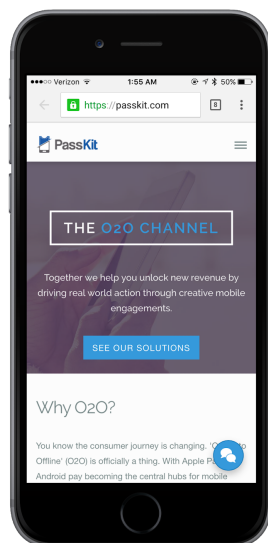
O **parâmetro Instance**, possui semelhança dos parâmetros Major e Minor da tecnologia iBeacon, serve para diferenciar cada beacon.

Exemplo de um valor para o parâmetro Instance: 0BDB87539B67..

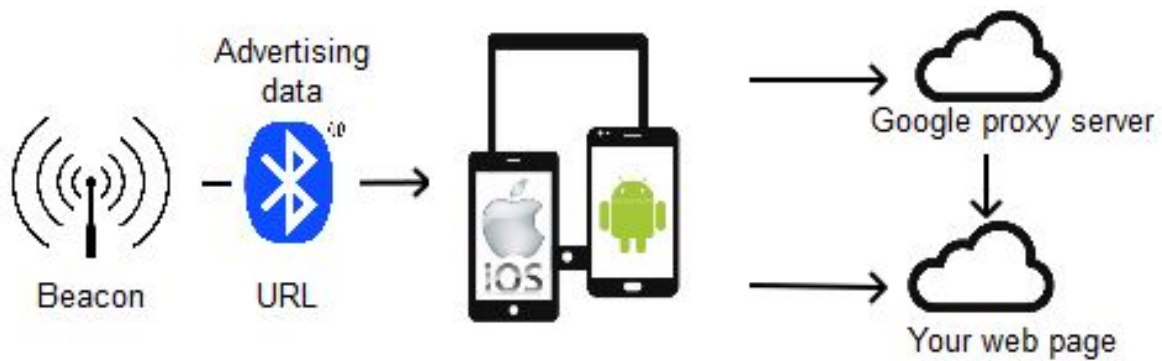


Eddystone-URL é composto apenas por um único campo: URL. O tamanho do pacote é variável e está associado ao tamanho do URL. O URL pode ser uma página Web, com informação relevante para o cliente, sendo o mesmo inserido no beacon através de uma API fornecida pelo fabricante.

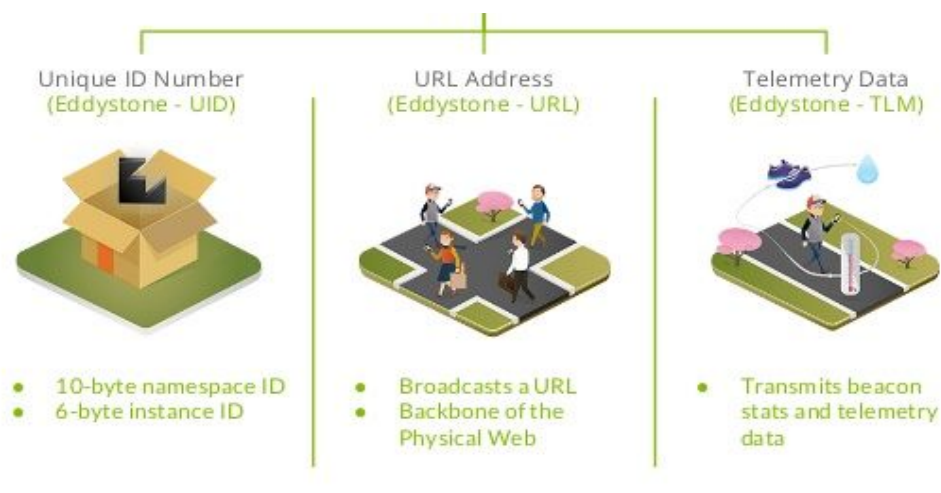
Exemplo de um URL: http://my-restaurant.com/checkin?restaurant_id=6523

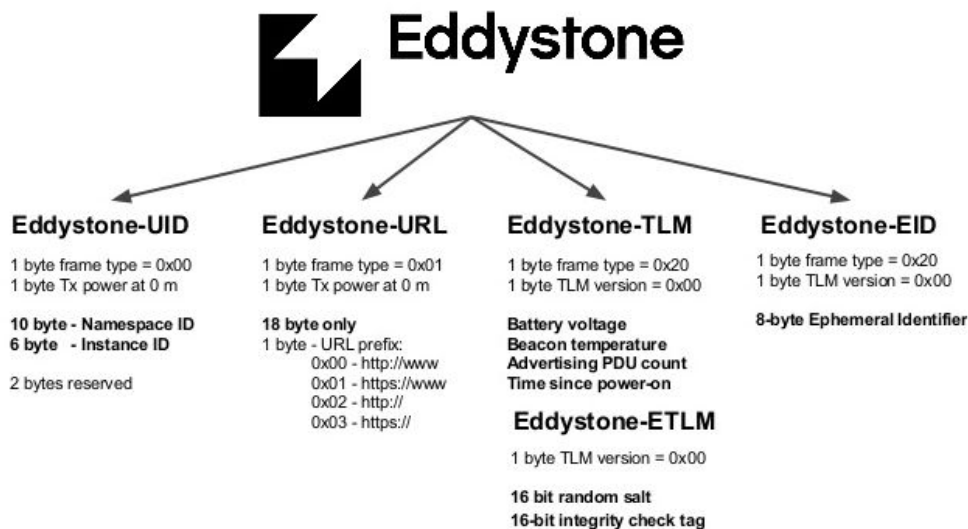


O **Eddystone-URL** é usado para serviços como marketing de varejo e espaços para visitantes, como museus. Nenhum aplicativo personalizado é necessário. Funciona no Android sem necessidade de aplicativo. O iOS precisa do navegador Google Chrome ou do aplicativo da Web física. Nenhuma solução de servidor de terceiros é necessária. Tudo o que você precisa é de um site seguro (HTTPS). NÃO é adequado para notificações de marketing de varejo não solicitadas. É útil apenas para cenários em que os usuários finais buscarão as transmissões de beacon.



Eddystone-TLM tem como objetivo enviar dados que permitam identificar o “estado” do beacon, nomeadamente: voltagem da bateria (usado para estimar o nível de bateria); temperatura do beacon; número de pacotes enviados desde que foi ligado; tempo que o beacon se encontra ligado (desde o último arranque). Estes dados são enviados apenas num único sentido, não podendo ser alterados no beacon. Apesar disso, através de uma aplicação no lado do cliente, é possível obter estes dados, enviar para a Cloud e, por exemplo, através de um CMS efetuar a gestão de beacons à distância.





As limitações são semelhantes a do Ibeacon com tudo o google possui a principal diferença é o Nearby um mecanismo nativo do celulares android que permite localizar componentes da web física beacon entre outros dispositivos onde o Nearby foi lançado originalmente em junho de 2010 desde então está presente nos dispositivos. Com o Nearby os dispositivos android não precisa de um aplicativo para interpretar a regra de negócio desde que o beacon envie a regra claramente para o dispositivo, contudo essa forma pode ser explorada podendo ser até uma forma de redirecionar para o app da empresa, uma oferta casada, promoção etc...

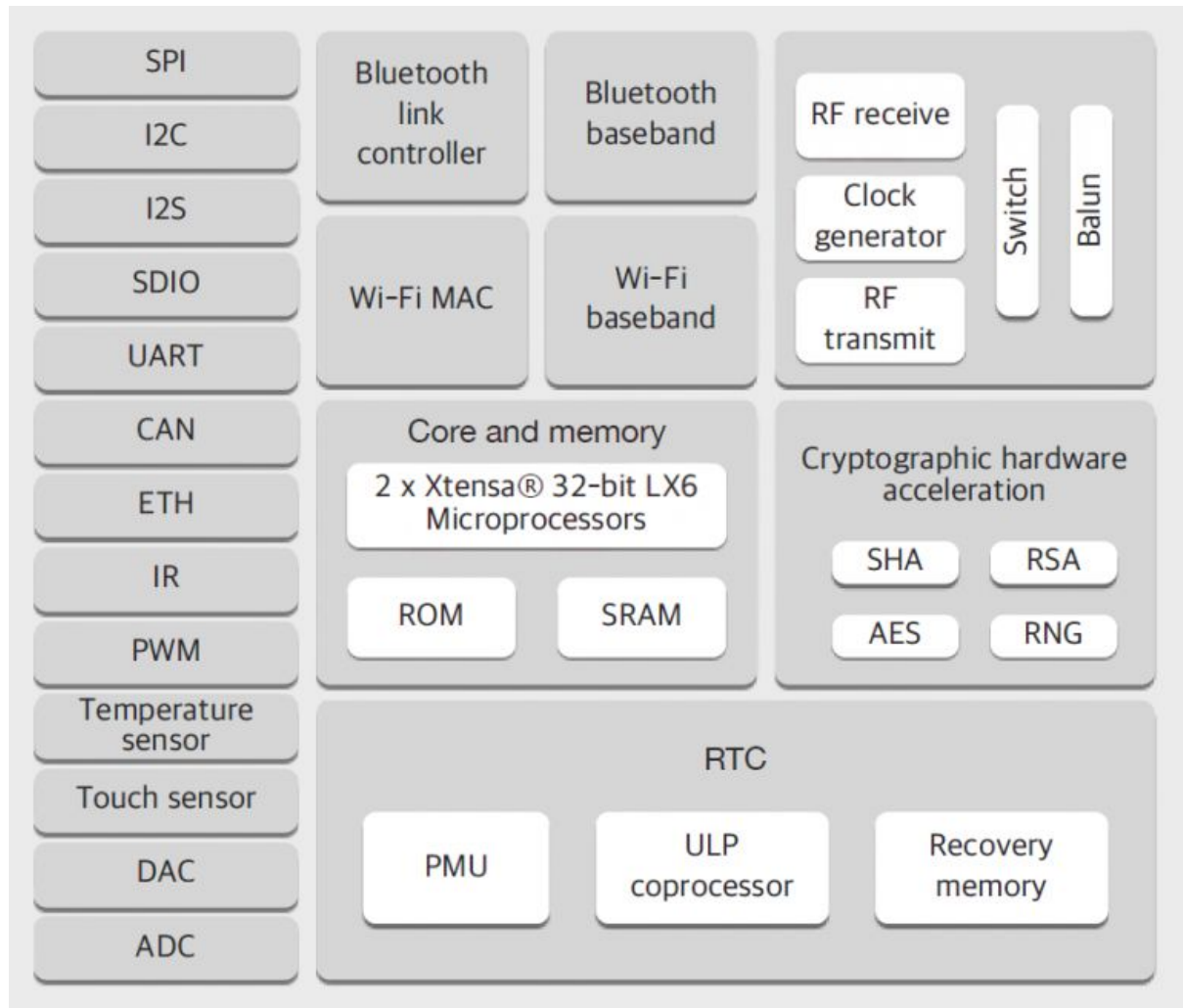
Outro diferencial do Eddystone e o fato de um protocolo Eddystone-URL onde dispositivo com navegadores móveis (Browser) pode interagir com o beacon, entrado os dispositivos IOS.

ESP32

O ESP32 possui muito recursos que se torna ideal para a Internet das Coisas algo bem interessante, dado que agora a presença de mais periféricos permite a sua integração com mais dispositivos. Dentre as interfaces de comunicação, ele possui suporte a SPI, UART e I2C (protocolos relativamente comuns), como também tem suporte a Infravermelho (IR) e SDIO (para interface com cartão de memória), e começa a se diferenciar, como mencionei antes, tendo CAN, Ethernet, DAC, Sensor de Toque, e I2S, que é uma interface de comunicação útil para comunicar com dispositivos de áudio, por último mas não menos importante também possui hardware para aceleração de criptografia embutido.

Sobre a conectividade sem-fio o ESP32 possui Bluetooth 4.2 capaz de BLE (modo de baixo consumo), WiFi assim como seu irmão mais velho o **ESP8266**. Usando BLE você pode implementar com o ESP32 um wearable (dispositivo vestível, como essas pulseiras inteligentes, acessório de roupa, etc) capaz de responder a requisições Bluetooth nos moldes

do BLE, onde cada “característica” tem sua assinatura, tal como temperatura, batimento cardíaco, posições x,y,z de um acelerômetro, etc.



Processadores:

- **Processador Principal:** Microprocessador Tensilica Xtensa 32-bit LX6
 - Núcleo: 2 ou 1 (depende da variação)
 - Todos os chips na série ESP32 são dual-core, com exceção do modelo ESP32-S0WD, que é single-core.
 - Frequência de Clock: até 240 MHz
 - Performance: até 600 DMIPS
- **Ultra low power co-processor:** Um coprocessador auxiliar, que consome bem pouca energia, e que é capaz de interagir com componentes tais como conversor ADC, realizar algumas instruções e tarefas enquanto os núcleos principais estão em modo deep sleep.

Conectividade Sem-Fio:

- Wi-Fi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz até 150 Mbit/s)
- Bluetooth: v4.2 BR/EDR e Bluetooth Low Energy (BLE)

Memória:

- Internal memory:

- ROM: 448 KiB – Usada em Boot e funções principais do ESP32
- SRAM: 520 KiB – Usada para dados e instruções (programas)
- RTC slow SRAM: 8 KiB – Para acesso do co-processador em modo deep-sleep.
- RTC fast SRAM: 8 KiB – Para armazenamento de dados e uso de CPU em boot de RTC (relógio de tempo-real) do modo deep-sleep.
- eFuse: 1 Kbit – Dos quais 256 bits são usados para sistema (endereço MAC e configurações do chip), e os restantes 768 bits são reservados para aplicações incluindo criptografia da Flash e Chip-ID.
- Flash Externa:
 - Suporte a até 16 MB de memória externa (4 MBytes na versão ESP-WROOM-32)

Periféricos de Entrada/Saída:

- Periféricos de comunicação com suporte a DMA.
- 10 GPIOs com suporte a toque capacitivo.
- 16 canais de conversor SAR ADCs (conversor analógico-digital) de 12-bits.
- 2 canais de 8 bits DACs (conversor digital-analógico).
- 2 Interfaces I²C (Inter-Integrated Circuit).
- 2 interfaces UART (universal asynchronous receiver/transmitter).
- Controlador CAN 2.0 (Controller Area Network).
- 4 interfaces SPI (Serial Peripheral Interface).
- 2 interfaces I²S (Integrated Inter-IC Sound).
- RMI (é a parte Ethernet do ESP32).
- 16 canais de PWM (modulação por largura de pulso).

Segurança:

- Conectividade IEEE 802.11 com suporte a protocolos de segurança WPA, WPA/WPA2 and WAPI.
- Boot seguro.
- Criptografia de Flash.
- Aceleração de Criptografia em Hardware usando: AES, SHA-2, RSA, ECC e RNG.

Através do site

<https://docs.espressif.com/projects/esp-idf/en/latest/get-started/index.html#get-started-connect>

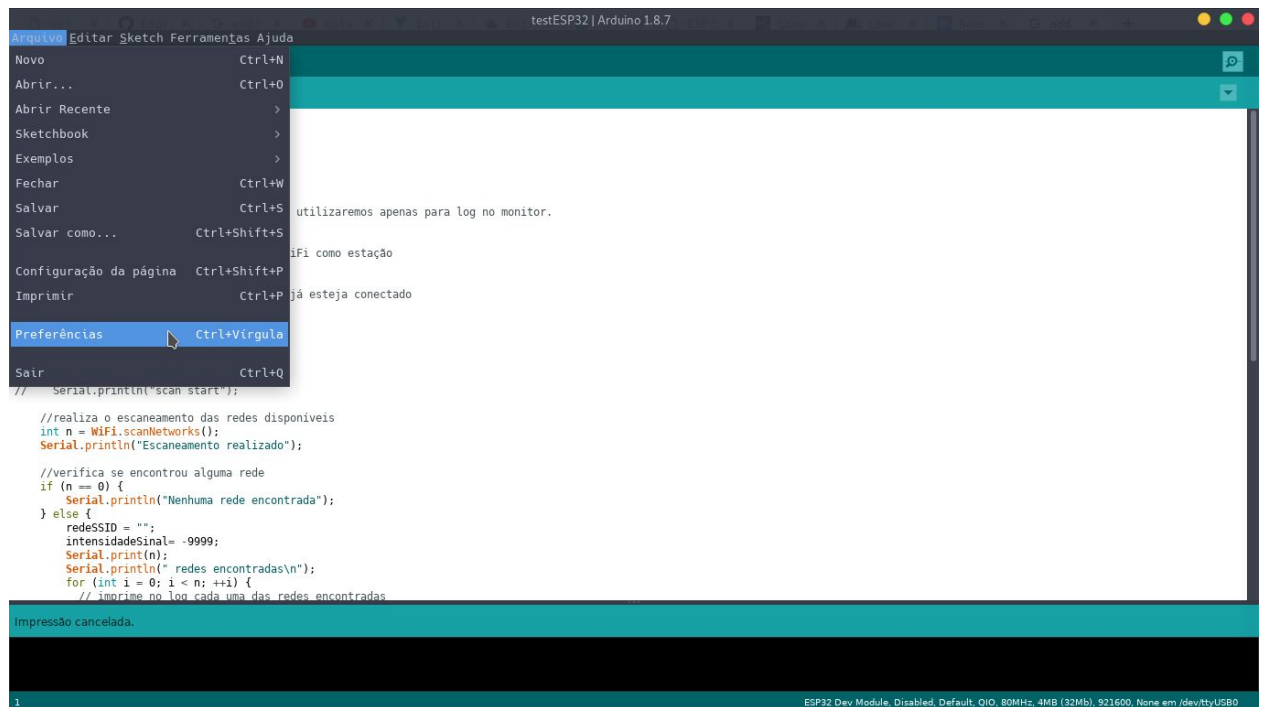
Comecei a conectar a placa com o sistema operacional que estou utilizando Manjaro Linux <https://manjaro.org/> com tudo não obtive sucesso por enquanto, outro link que também utilizei para conectar o ESP32 e enviar os códigos inclusive o tutorial é voltado para o mesmo Linux que eu utilizo mesmo assim não obtive sucesso

<http://pedrominatel.com.br/pt/esp32/esp32-e-idf-preparando-o-ambiente/>

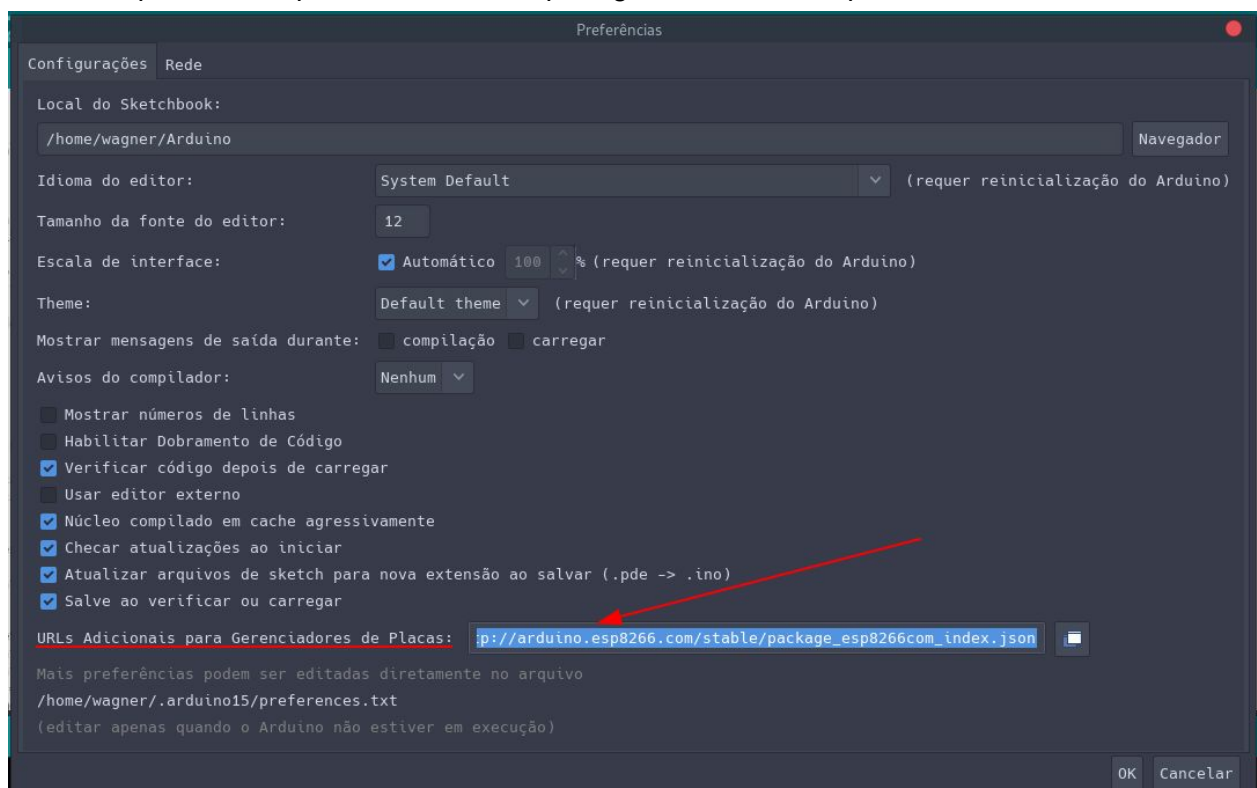
Outro caminho que tomei foi conectar o ESP32 através da IDE do Arduino depois de baixar do site

1. <https://www.arduino.cc/en/Main/Software>
2. Executar a IDE do arduino

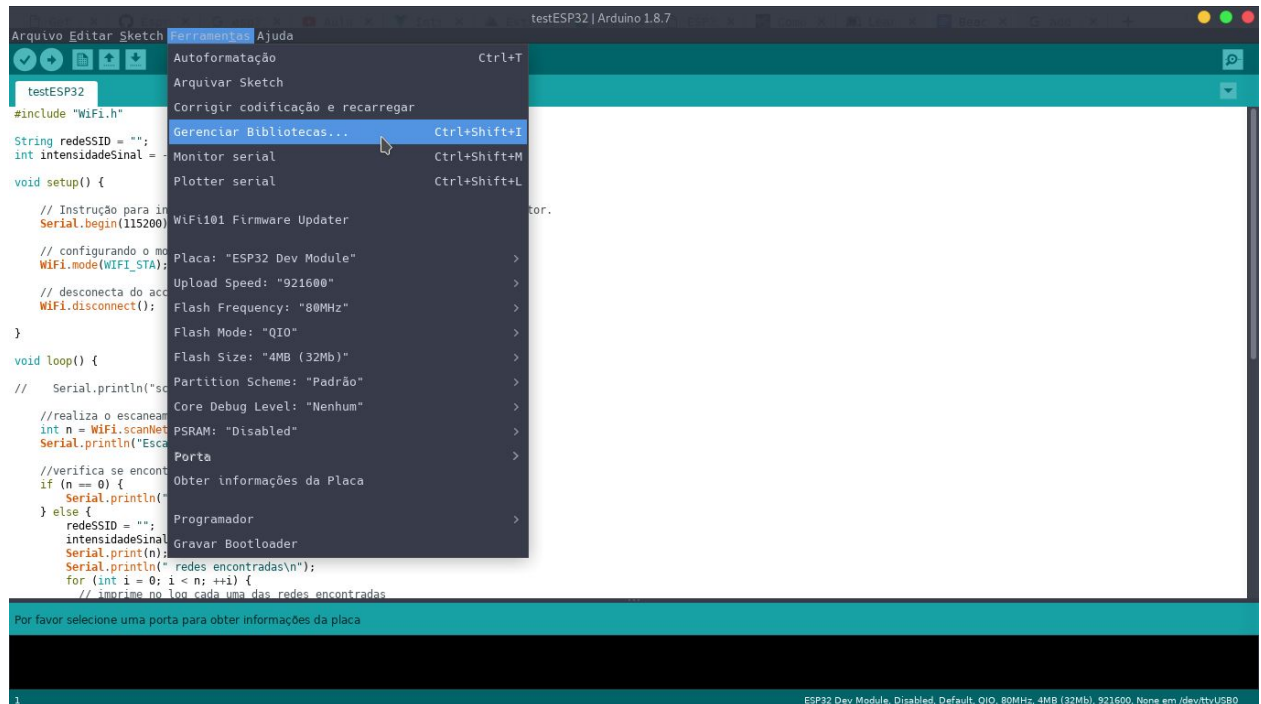
3. Ir em Preferências



4. Depois no campo URLs adicionar para gerenciadores de placas

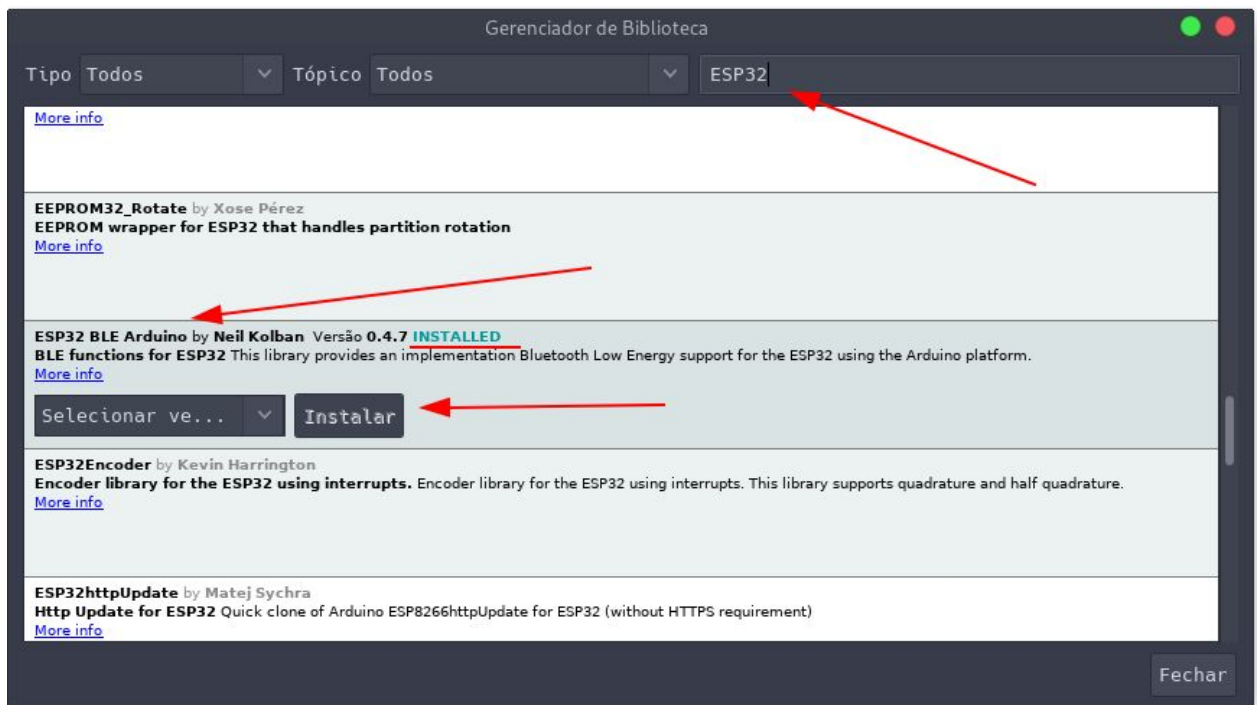


5. No campo adicionar a URL a seguir
https://dl.espressif.com/dl/package_esp32_index.json
http://arduino.esp8266.com/stable/package_esp8266com_index.json
6. clique em OK para voltar para a IDE logo em seguida selecione a aba FERRAMENTAS > Gerenciador Biblioteca

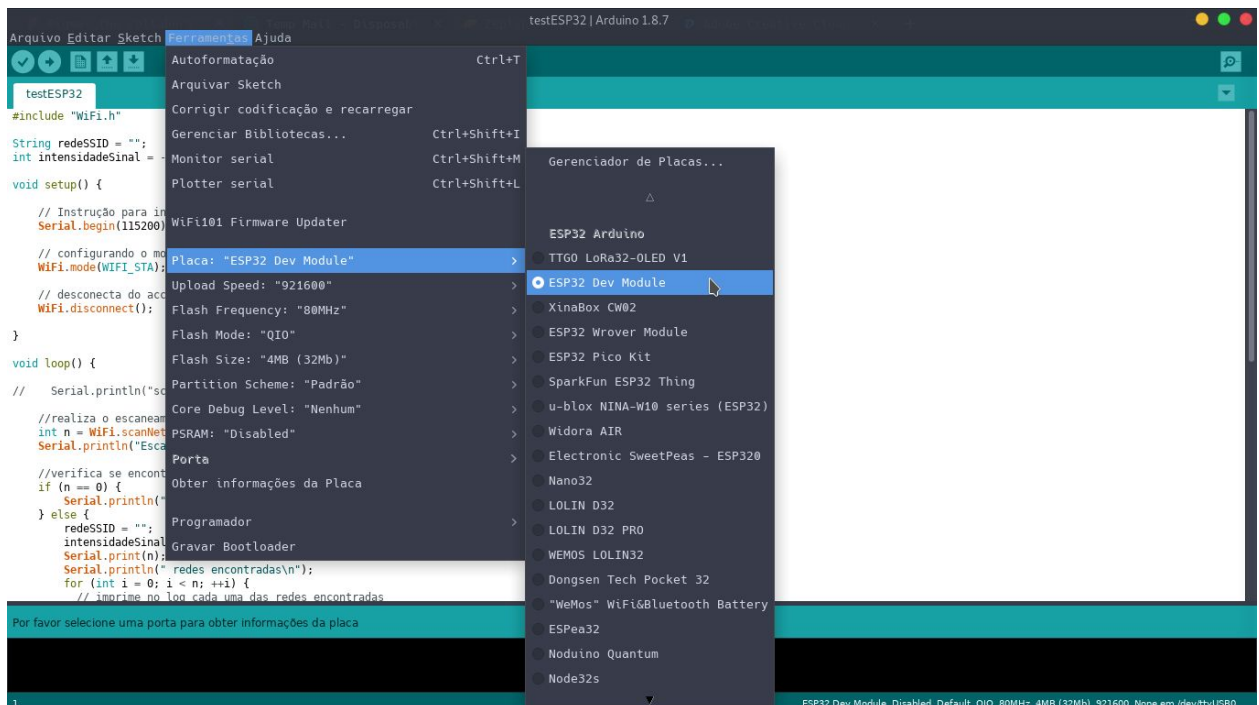


7. Na janela que abriu procure no campo de por ESP32 e baixe a biblioteca de Bluetooth como o nosso foco é implementar o beacon. Procure por ESP32 BLE Arduino e clique em instalar espere instalar, depois de instalado aparecerá a palavra INSTALLED logo

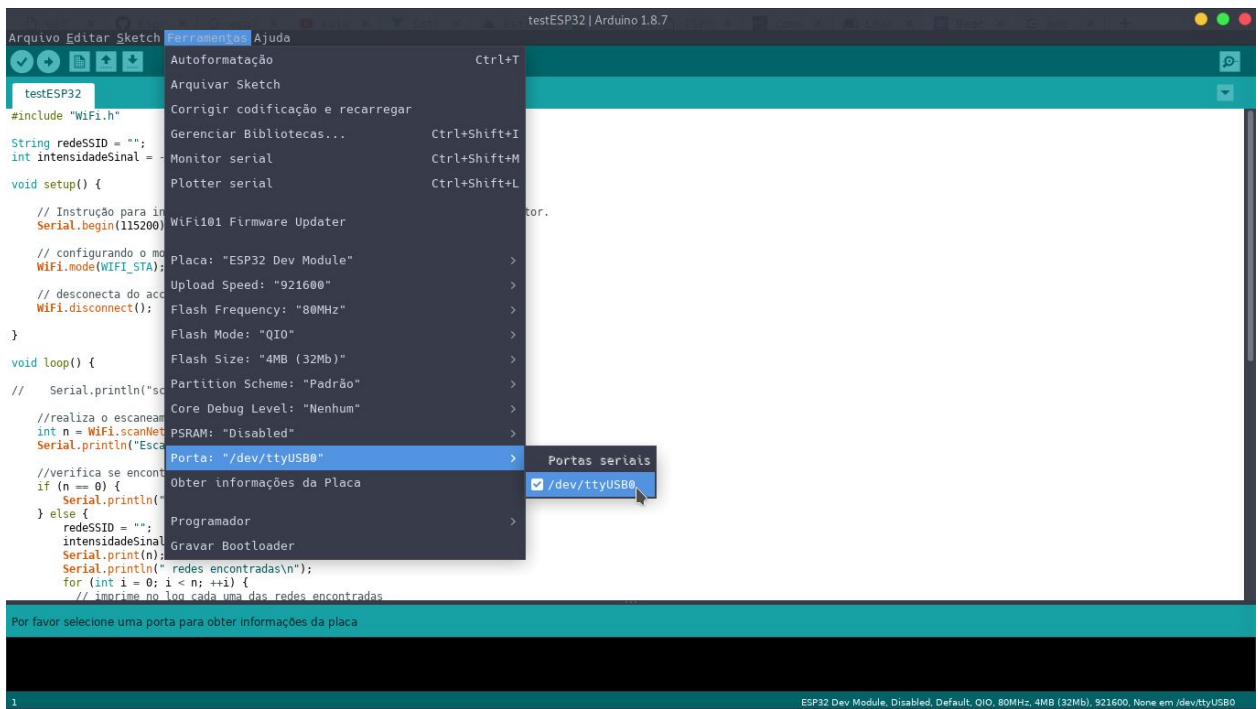
depois do nome da biblioteca terminado isso clique em fecha



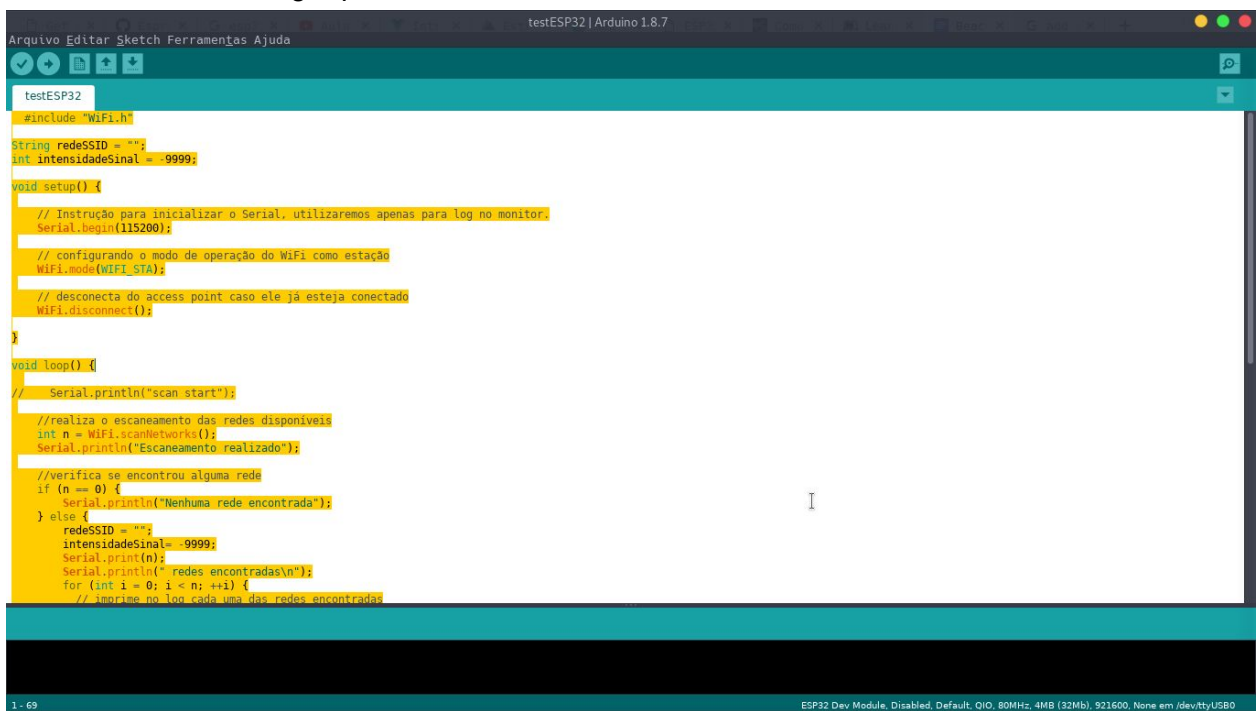
- Com isso ficará disponível na sessão de FERRAMENTAS > PLACAS a placa ESP32. selecione ela para a IDE saiba qual placa você está trabalhando.



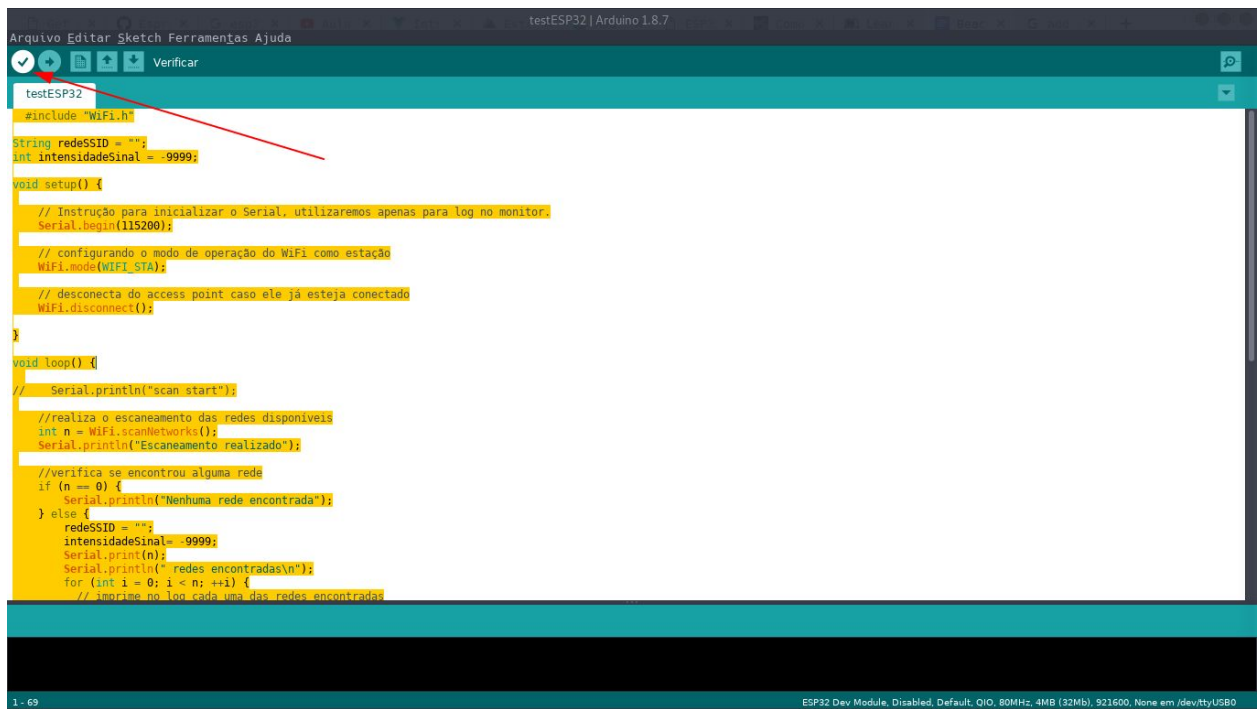
- Com a placa conectada no Computador vai na Aba FERRAMENTAS > PLACA e selecione a porta USB que a placa está conectada, no meu caso está na `/dev/ttyUSB0`



10. Escreva o código que do beacon



11. Clique em verificar se o código está correto



Arquivo Editar Sketch Ferramentas Ajuda

testESP32 | Arduino 1.8.7

Verificar

```
testESP32
#include "WiFi.h"

String redeSSID = "";
int intensidadeSinal = -9999;

void setup() {
  // Instrução para inicializar o Serial, utilizaremos apenas para log no monitor.
  Serial.begin(115200);

  // configurando o modo de operação do WiFi como estação
  WiFi.mode(WIFI_STA);

  // desconecta do access point caso ele já esteja conectado
  WiFi.disconnect();
}

void loop() {
  // Serial.println("scan start");

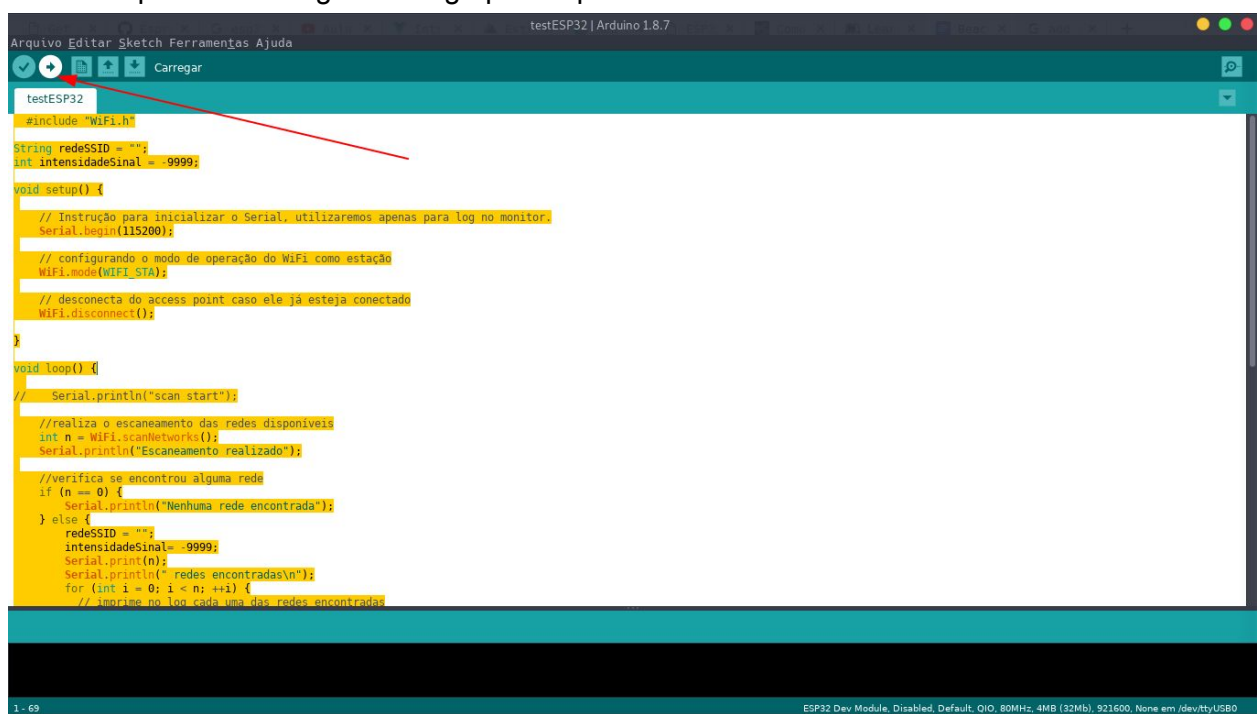
  //realiza o escaneamento das redes disponíveis
  int n = WiFi.scanNetworks();
  Serial.println("Escaneamento realizado");

  //verifica se encontrou alguma rede
  if (n == 0) {
    Serial.println("Nenhuma rede encontrada");
  } else {
    redeSSID = "";
    intensidadeSinal = -9999;
    Serial.println(n);
    Serial.println("redes encontradas\n");
    for (int i = 0; i < n; ++i) {
      // imprime no log cada uma das redes encontradas
    }
  }
}
```

1 - 69

ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None em /dev/ttyUSB0

12. Depois em carregar o código para a placa



Arquivo Editar Sketch Ferramentas Ajuda

testESP32 | Arduino 1.8.7

Carregar

```
testESP32
#include "WiFi.h"

String redeSSID = "";
int intensidadeSinal = -9999;

void setup() {
  // Instrução para inicializar o Serial, utilizaremos apenas para log no monitor.
  Serial.begin(115200);

  // configurando o modo de operação do WiFi como estação
  WiFi.mode(WIFI_STA);

  // desconecta do access point caso ele já esteja conectado
  WiFi.disconnect();
}

void loop() {
  // Serial.println("scan start");

  //realiza o escaneamento das redes disponíveis
  int n = WiFi.scanNetworks();
  Serial.println("Escaneamento realizado");

  //verifica se encontrou alguma rede
  if (n == 0) {
    Serial.println("Nenhuma rede encontrada");
  } else {
    redeSSID = "";
    intensidadeSinal = -9999;
    Serial.println(n);
    Serial.println("redes encontradas\n");
    for (int i = 0; i < n; ++i) {
      // imprime no log cada uma das redes encontradas
    }
  }
}
```

1 - 69

ESP32 Dev Module, Disabled, Default, QIO, 80MHz, 4MB (32Mb), 921600, None em /dev/ttyUSB0

13. Espere o processo terminar pronto o pode usar o seu beacon na aplicação.

Código de Exemplo da própria Espressif

IBEAICON

https://github.com/espressif/esp-idf/tree/master/examples/bluetooth/ble_ibeacon/main

EDDYSTONE

https://github.com/espressif/esp-idf/tree/master/examples/bluetooth/ble_eddystone

Physical WEB

A Web Física (ou Physical Web) é um projeto do Google para levar o conceito e o comportamento da Web para os espaços físicos. Segundo os autores do projeto, a Physical Web é uma tentativa de explorar o potencial mais importante da web: interação sob demanda.

“As pessoas devem poder se aproximar de qualquer objeto inteligente — uma vending machine, um pôster, um brinquedo, um ponto de ônibus, um carro alugado — e não ter que baixar um aplicativo antes para interagir com conteúdo web. Tudo deve estar a um ‘tap’ de distância.”

Exemplo da WEB FÍSICA

Vídeos:

<https://www.youtube.com/watch?v=BL6djal9mqY>

<https://www.youtube.com/watch?v=8AryiXsPQ1Y>

Artigos

<https://google.github.io/physical-web/>

<https://google.github.io/physical-web/examples>

<https://brasil.uxdesign.cc/o-que-%C3%A9-a-physical-web-e-por-que-me-importar-com-ela-b5805535548>

Referências

Documentação da Espressif fabricante do ESP32

<https://www.espressif.com/en/support/download/documents>

NodeMcu AMICA

<https://roboindia.com/tutorials/nodemcu-amica-esp8266-board-installation>

ESP32:

<https://www.filipeflop.com/blog/esp32-um-grande-aliado-para-o-maker-iot/>