



Uma introdução à Internet das Coisas e Sistemas Embarcados utilizando programação por blocos com BIPES e ESP8266 / ESP32

Acessar endereço web com:

URL

“ <https://bipes.net.br/> ”

Rafael Vidal Aroca
Wesley Flavio Gueta
Jorge André Gastmaier Marques
Tatiana de Figueiredo Pereira Alves Taveira Pazelli

Dezembro de 2021 - 1ª Edição

**Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)**

Uma introdução à internet das coisas e sistemas embarcados utilizando programação por blocos com BIPES e ESP8266 / ESP32 [livro eletrônico] / Rafael Vidal Aroca ... [et al.]. -- São Carlos, SP : Ed. dos Autores, 2021. PDF.

Outros autores : Wesley Flavio Gueta, Jorge André Gastmaier Marques, Tatiana de Figueiredo Pereira Alves Taveira Pazelli.

Bibliografia.

ISBN 978-65-00-36921-2

1. Base de dados 2. Ciência da computação 3. Internet das coisas 4. Internet (Rede de computador) 5. Programação (Computadores) - Estudo e ensino 6. Python (Linguagem de programação para computadores) I. Aroca, Rafael Vidal. II. Gueta, Wesley Flavio. III. Marques, Jorge André Gastmaier. IV. Pazelli, Tatiana de Figueiredo Pereira Alves Taveira.

21-95610

CDD-004.678

Índices para catálogo sistemático:

1. Internet das coisas : Ciência da computação
004.678

Aline Grazielle Benitez - Bibliotecária - CRB-1/3129

Sumário

Internet das coisas com BIPES	4
Introdução relâmpago	4
Lista de materiais	9
Preparação da placa	12
Piscar o LED da placa	16
Entrada digital e verificação de uma condição	24
Entrada analógica e sensor de luz (LDR)	27
Data e hora (RTC)	32
Arquivos na placa	35
Verificando uma condição (Parte 2)	38
Listar redes Wi-fi	39
Conectando na Internet	39
Envio de dados para a Internet / “Nuvem”	39
Verificando erros	45
BIPES: Múltiplos projetos	46
Atividades em paralelo: temporizadores (timers)	47
Sensor de temperatura e umidade	49
Compartilhe o dashboard com celulares e outros dispositivos!	51
PWM: Controle de brilho do LED	53
BIPES: Subrotinas / funções	55
Controlando dispositivos via Internet	57
Cliente web e servidor web (HTTP)	63
Cliente HTTP	65
Cliente HTTP: Envio de mensagens SMS	66
Cliente HTTP: Galinho do tempo IoT	67
Cliente HTTP: Outras possibilidades	73
Servidor HTTP	75
Integração com Google Home ou Amazon Alexa	78
Atividades extra	90
Envio de email	90
Unidade de Medidas Inerciais - MPU6050	92
Servo motor de aeromodelos (RC Servo)	93
Música	94
BIPES em outras plataformas	96
Python com Arduino - SNEK	96
Outras	96
Obtendo ajuda e ajudando	96
Considerações Finais	97
Agradecimentos	98

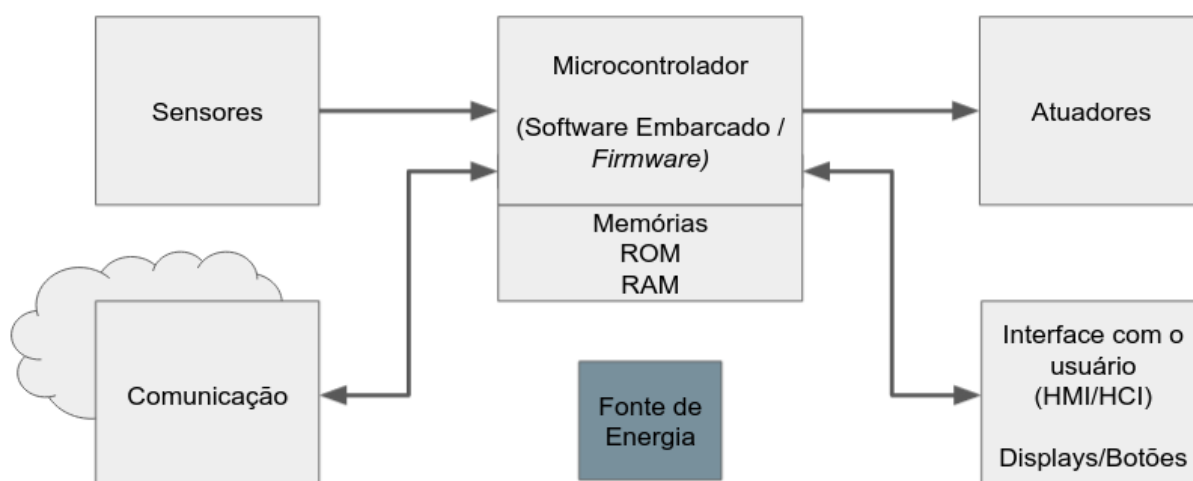
Internet das coisas com BIPES

Introdução relâmpago

BIPES: Block based Integrated Platform for Embedded Systems (<http://bipes.net.br>) é uma plataforma totalmente aberta e livre, desenvolvida no Brasil, para programação de sistemas embarcados e aplicações de Internet das Coisas [1].

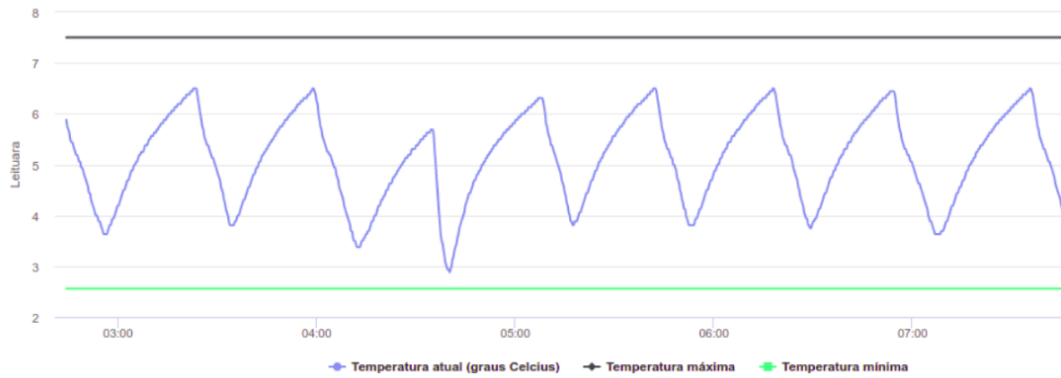
Sistemas embarcados estão presentes em toda parte da vida cotidiana: no comércio, na indústria, em carros, fornos, televisões, copiadoras, portões de garagem, alarmes, geladeiras e outros dispositivos. Basicamente, um sistema embarcado é composto por hardware e/ou software que realiza uma tarefa específica e “embarcada” em um produto. Daí surgem os nomes: sistema embarcado, hardware embarcado e software embarcado. Para implementar sistemas embarcados, existem diversos dispositivos disponíveis, como microprocessadores e microcontroladores. Os microcontroladores, em especial, são pequenos circuitos integrados (chips), normalmente de baixo custo, que demandam poucos componentes externos e podem implementar uma lógica de funcionamento com base em entradas e informações pré-definidas para controlar saídas. Eles são como pequenos computadores completos, mas projetados para serem direcionados a aplicações específicas. O software embarcado, em geral, é o que determina a lógica de funcionamento de um sistema embarcado.

A figura abaixo mostra uma visão geral de um sistema embarcado que pode ser composto pelo microcontrolador em conjunto com diversos outros dispositivos, embora nem todos sejam necessários para todas aplicações. Assim, cada aplicação irá determinar a necessidade de sensores, atuadores, e outros elementos em um sistema embarcado.



- **Microcontrolador:** É um componente eletrônico, normalmente composto por diversas portas de entrada e saída de sinais, por memórias e um processador capaz de executar a lógica definida pelo software embarcado ou *firmware*;
- **Sensores:** Sensores são dispositivos eletrônicos que detectam / medem o estado / nível de uma certa grandeza física do ambiente, como por exemplo, temperatura, umidade, iluminação, vibração, campo magnético, entre outros;
- **Atuadores:** Atuadores são dispositivos que alteram o estado do ambiente, como motores, aquecedores, eletroímãs, e outros dispositivos que possam, através de comandos do microcontrolador, realizar uma ação sobre o ambiente. A grande maioria dos microcontroladores não possui a capacidade de acionar diretamente um atuador, de forma que circuitos de potência (*drivers*), que podem incluir transistores e/ou pontes-H devem ser usados para que microcontroladores possam acionar atuadores;
- **Comunicação:** Alguns sistemas embarcados podem incluir mecanismos de comunicação de dados a distância, permitindo que enviem dados para monitoramento remoto, ou recebam comandos remotos. Este tipo de conectividade pode acontecer via cabos ou via rádio, e pode ou não ter conexão com a Internet;
- **Fonte de energia:** Um sistema embarcado requer uma fonte de alimentação para fornecer energia para seu funcionamento. O fornecimento de energia pode ser obtido a partir da rede elétrica, baterias, painéis solares, dentre outras fontes de energia elétrica. Uma tendência em alguns sistemas embarcados e sistemas de Internet das Coisas são os equipamentos que obtêm energia do ambiente, por exemplo, de ondas de rádio;
- **Interface com o usuário:** Diversos sistemas embarcados possuem a necessidade de interagir com o usuário, seja por displays, botões, teclados, mensagens de voz, comandos de voz, ou mesmo, através de interfaces de controle e visualização em smartphones.

Considerando essas definições, a figura abaixo mostra um exemplo de sistema embarcado composto por um microcontrolador, um sensor de temperatura e um compressor (atuador) controlando a temperatura interna de uma geladeira. O sensor de temperatura e o microcontrolador, juntos, custam menos que um hambúrguer, sendo possível implementar diversas lógicas e funcionalidades. Nesta figura, pode-se visualizar que toda vez que a temperatura passa de 6 graus Celsius, algo ocorre e ela começa a cair. De fato, o sistema embarcado detecta que a temperatura ultrapassou os 6 graus Celsius e aciona o compressor da geladeira, que resfria o ambiente. Conseqüentemente, essa alteração no ambiente também é detectada pelo sensor de temperatura. Ao alcançar a temperatura de 4 graus, o atuador é desligado. Assim, nessa aplicação, o ciclo se repete em uma ação de controle conhecida como “*on-off*”, seguindo a lógica implementada no software embarcado.



Este sistema também pode ser conectado à Internet, permitindo que o ajuste de temperatura desejada seja configurado remotamente, por um celular, por exemplo, e que a temperatura seja monitorada / visualizada remotamente. Além disso, o sistema poderia emitir alertas e alarmes no caso de falha do compressor, ou temperaturas inadequadas. Esta associação de um sistema embarcado, com ajustes e monitoramento via Internet, encontra-se na categoria de **Internet das Coisas** ou **Internet of Things (IoT)**. Assim, atualmente existem milhares de dispositivos conectados à Internet, como lâmpadas, medidores inteligentes, geladeiras, aparelhos de ar condicionado, dentre outros.

Ainda no contexto de sistemas embarcados, é impressionante analisar os avanços dos últimos anos. Este livro apresenta atividades práticas utilizando o módulo ESP8266 ou o módulo ESP32, ambos da Espressif Systems. Estes módulos, classificados como “Sistemas em um Chip” (*System on Chip - SoC*) são computadores completos, incluindo processador, memória RAM, memória FLASH para guardar programas, conexão sem fio Wi-Fi, entradas para sensores, saídas para atuadores e vários outros recursos. E custam menos do que um hamburger! Se você pensar bem, eles são muito mais poderosos que o computador que levou o módulo lunar da missão Apollo à superfície da Lua. Veja a figura abaixo para uma comparação.

Apollo Guidance Computer



Clock: 2MHz
RAM: 2KB
ROM: 36KB
Peso: 32000 gramas
Energia: 55 Watts
Custo: US\$ 150000,00

Arduino UNO



Clock: 16MHz
RAM: 2KB
ROM: 32KB
Peso: 25 gramas
Energia: 0,2 Watts
Custo: US\$ 7,00

ESP8266



Clock: 160MHz
RAM: ~ 100KB
ROM / Flash: 16MB
Peso: 1.5 gramas
Energia: 0,3 Watts
Custo: ~US\$ 1,00

Wifi / MicroPython + BIPES
www.bipes.net.br

Embora a figura não apresente o módulo ESP32 na comparação, o ESP32 é uma versão mais poderosa do ESP8266, com vários recursos adicionais, incluindo processador de dois núcleos, Bluetooth, maior capacidade de memória e processamento, além de outros recursos. É importante notar que todas as explicações e atividades apresentadas neste livro são apresentadas com exemplos para o ESP8266, mas funcionam, da mesma forma, no ESP32, mudando, eventualmente, os pinos, já que os dois módulos têm diferentes configurações de pinos (terminais eletrônicos) de entrada e saída. Ao longo do texto, estes módulos são mencionados indistintamente, e em alguns momentos chamados apenas de “ESP”.

O **BIPES** (<http://bipes.net.br>), por sua vez, será utilizado no desenvolvimento da lógica de funcionamento das aplicações. Versátil, aberta e gratuita, a plataforma integrada BIPES possibilita a programação em diversas placas microcontroladoras como ESP32, ESP8266, Arduino, mBed, dentre outras. O **BIPES** pode ser usado diretamente em navegadores web de vários dispositivos e não requer a instalação de nenhum software, plugin ou configuração especial. Além da facilidade de desenvolvimento através de programação por blocos, o **BIPES** também oferece uma plataforma de Internet das Coisas, viabilizando a construção de painéis de visualização e controle remoto para aplicações IoT [2].

Com relação à programação por blocos, inicialmente focada em aplicações educacionais pela facilidade, ela vem sendo adotada cada vez mais em ambientes de desenvolvimento comercial e industrial, onde já se percebeu diversas vantagens: redução de tempo de desenvolvimento, redução de falhas lógicas, maior facilidade de entendimento e manutenção da programação. De fato, recentemente, uma publicação do Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) disponibilizou um artigo sobre o nascimento da era da programação sem código [3].

Como mencionado, esta é uma introdução relâmpago ao tema. Várias publicações de qualidade estão disponíveis para você se aprofundar neste assunto e devem ser consultadas. Nosso foco aqui, em uma abordagem prática, é guiar você em testes rápidos para implementação de sistemas IoT de forma prática e rápida.

Referências:

[1] A. G. D. S. Junior, L. M. G. Gonçalves, G. A. de Paula Caurin, G. T. B. Tamanaka, A. C. Hernandez, R. V. Aroca. BIPES: Block Based Integrated Platform for Embedded Systems. **IEEE Access**, v. 8, p. 197955-197968, 2020.




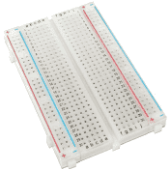

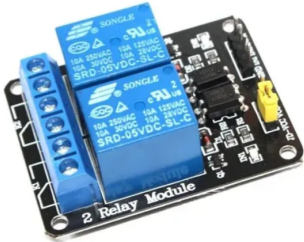

Disponível em: <<https://ieeexplore.ieee.org/document/9246562>>.


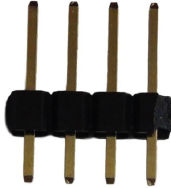
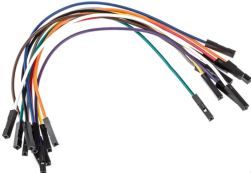




[2] C. A. Silva. **Desenvolvimento e validação de módulo de comunicação MQTT para plataforma BIPES para aplicações de Internet das Coisas**. 2020. Trabalho de Conclusão







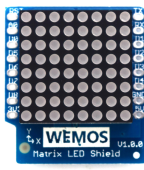
de Curso (Graduação em Engenharia de Computação) - Universidade Federal de São Carlos, 2020. Disponível em: <<https://repositorio.ufscar.br/handle/ufscar/13656>>.

[3] R. D. Caballar. Programming without code: The rise of no-code software development. **IEEE Spectrum**, Tech Talks, 11 mar. 2020. Disponível em: <<https://spectrum.ieee.org/programming-without-code-no-code-software-development>>

Lista de materiais

Item		Descrição
1		Qualquer placa com módulo ESP8266 ou ESP32. Para os exemplos do livro, o principal módulo utilizado foi o Wemos D1 Mini com ESP8266.
2		Sensor de umidade e temperatura DHT11
3		Sensor de luz (LDR)
4		Placa de prototipagem - protoboard
5		Plugue para tomada com pino
6		Placa com relé
7		Tomada

8		Bocal para lâmpada
9		Barras de terminais
10		Cabinhos pré-montados (<i>jumper cables</i>)
11		<i>Power bank</i>
12		Lâmpada 127 Volts
13		Potênciometro
14		Cabo paralelo (branco) para conexão na rede elétrica

15		Botão (<i>Switch / Push Button</i>)
16		Micro servo motor
17		Resistores diversos com valores de 1K, 570 ohms e 220 ohms
18		Emissor de som piezoelétrico (<i>Buzzer</i>)
19		Diodo emissor de luz (LED)
20		Diodo emissor de luz (LED) de 3 cores (RGB) Opcional: apenas para a prática do “Galinho do Tempo”
21		<i>Shield</i> matriz de LEDs com controlador TM1640 Opcional: apenas para a segunda parte da prática do “Galinho do Tempo”

Preparação da placa

Diversas placas podem ser utilizadas para realizar as atividades propostas neste texto, entretanto, nesta ocasião, iremos focar no uso da placa com módulo ESP8266. Além disso, é necessário que o MicroPython (<https://micropython.org/>) esteja instalado nesta placa, permitindo programar, depurar e gerenciar aplicações de forma prática e dinâmica, com uma implementação minimalista da linguagem de programação Python. Utilizando os recursos do projetos BIPES, a instalação do MicroPython na ESP8266 é bastante simples. Os passos a seguir funcionam nos navegadores Google Chrome e Microsoft Edge em ambientes MAC, Linux e Windows.

Para instalar o MicroPython na ESP8266:

1. Acesse: <https://bipes.net.br/flash/esp-web-tools/>.
2. Conecte a placa ESP8266 à porta USB do PC.
3. Clique em **CONNECT**.

BIPES MicroPython Web Installer

(for ESP32 and ESP8266)

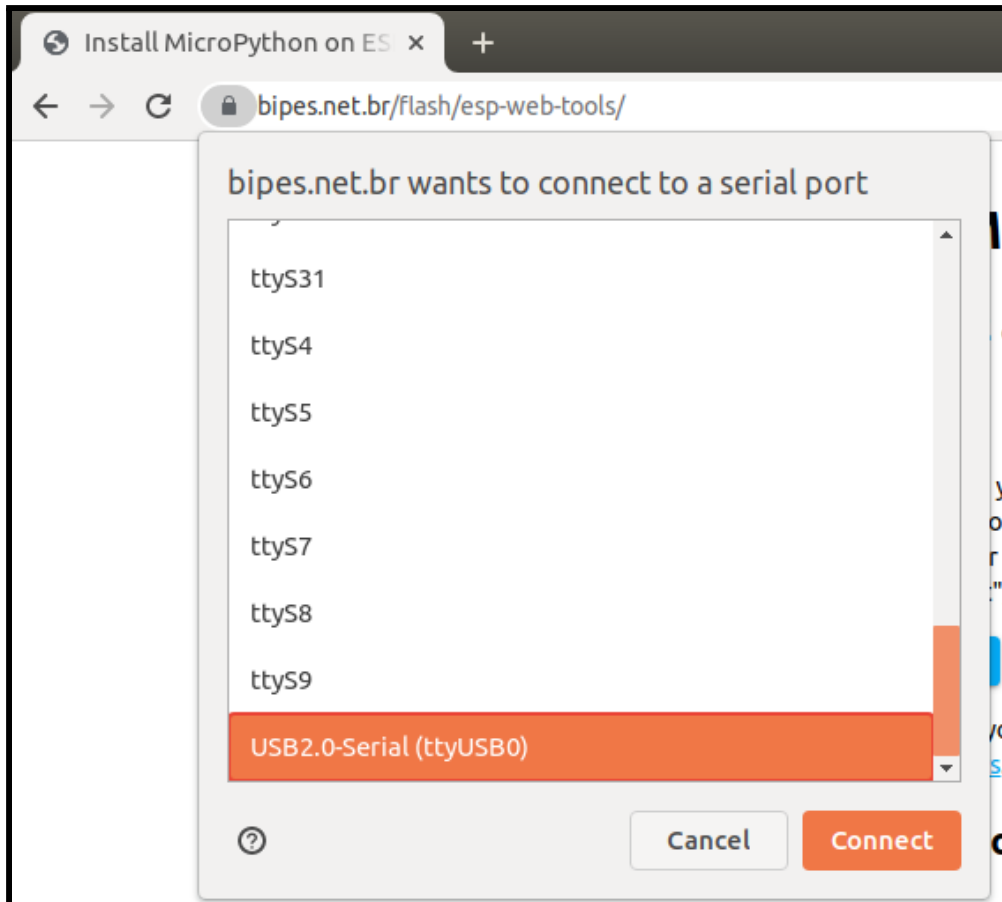
Install

This page allows you quickly and easily install [MicroPython](#) on ESP32 or ESP8266 board directly from the browser, without the need of using esptool or any other software on your computer. Simply connect the board to an USB port, click on the button "Connect", select the device and have MicroPython running on your board!

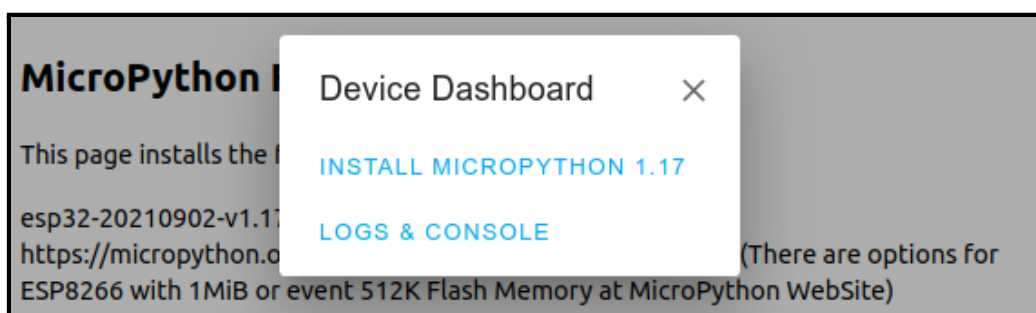
CONNECT

If you don't see your ESP device in the list of devices to choose, you might need to install [the drivers](#).

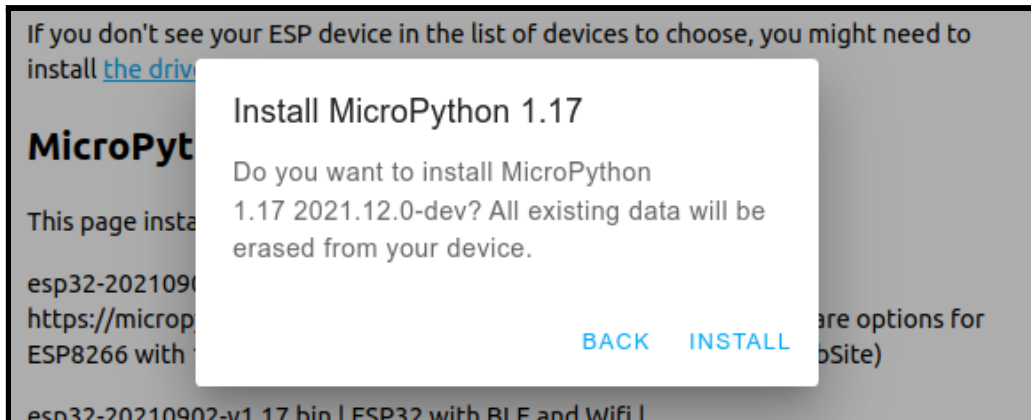
4. Escolha a porta serial (COM) usada pelo seu dispositivo e clique em **Connect**:



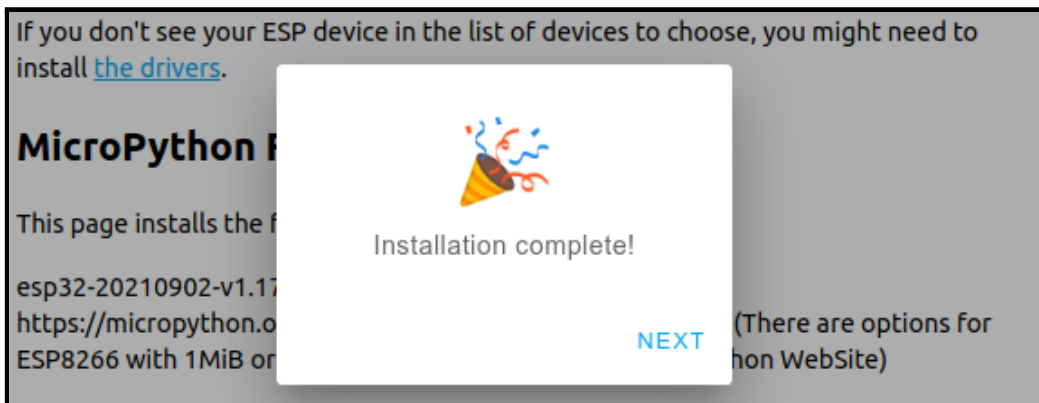
5. Em seguida, clique em **INSTALL MICROPYTHON**:



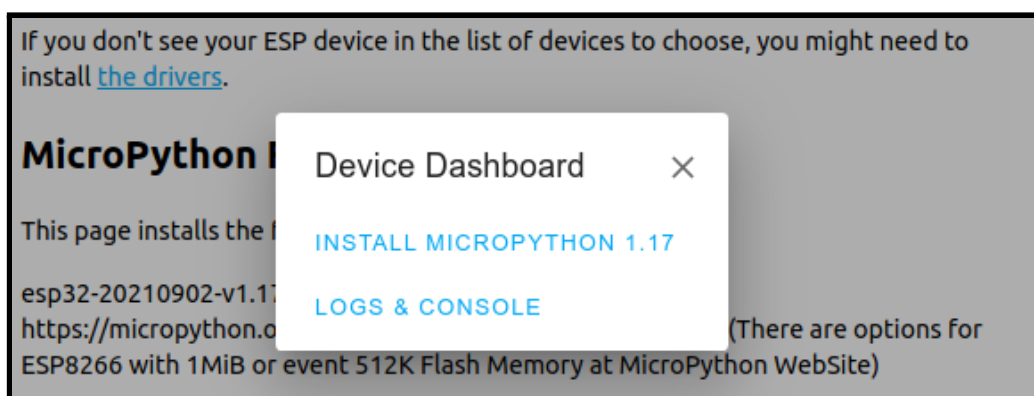
6. Depois clique em **INSTALL**:



7. Após a instalação, você verá a mensagem:



8. Clique em **NEXT** e depois em **LOGS & CONSOLE**.



Observe o comportamento da placa pelo console. Por padrão, o MicroPython cria uma rede "MicroPython", para conexão e programação remota, via Wi-fi. Entretanto, algumas placas ESP8266 genéricas possuem circuito de alimentação que fornecem corrente insuficiente para suprir sua demanda de energia quando a placa ESP8266 inicia o

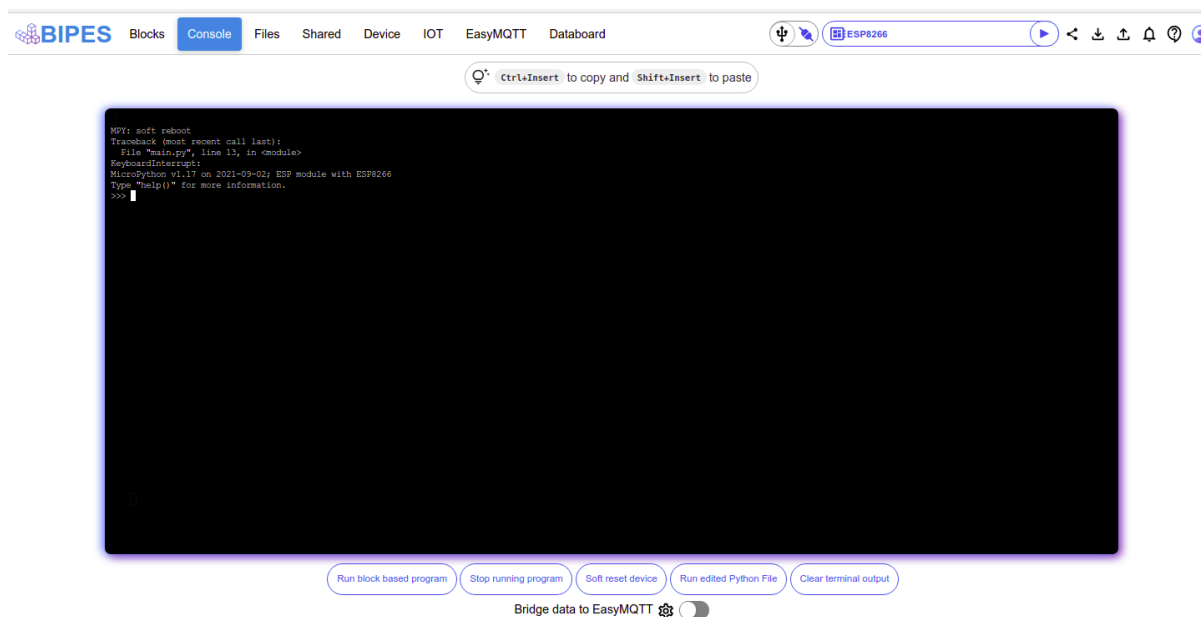
Piscar o LED da placa

O diodo emissor de luz, ou LED, é um indicador luminoso usado em várias situações para indicar o estado de um sistema embarcado. Por exemplo, em um modem, ele pode indicar se a Internet está conectada. Em geral, a atividade de “pisca um LED” é uma das primeiras atividades realizadas para testar / conhecer uma nova placa / dispositivo usado para construir um sistema embarcado. Nesta seção, faremos uma atividade passo a passo para piscar um LED já existente na placa ESP8266. Ou seja, não é preciso conectar nenhum LED externo. Basta usar o que já está na placa.

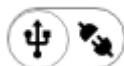
1. Conecte a placa com ESP8266 na porta USB do PC.
2. Acesse o ambiente de desenvolvimento do **BIPES**: <https://bipes.net.br/ide/>.
3. Escolha, no canto superior direito a placa **ESP8266**:



4. Acesse a aba **Console**:



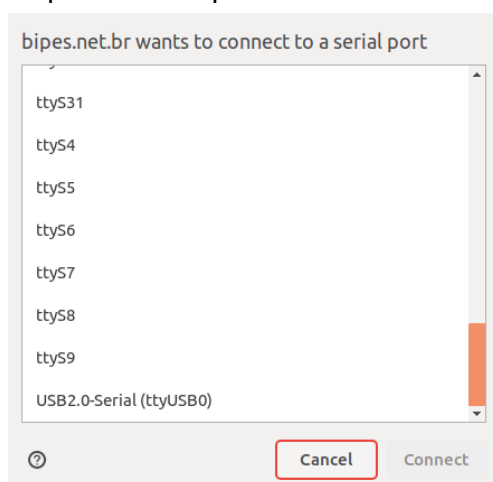
5. Clique no ícone para conectar:



6. Clique em **Serial**:



7. Escolha uma porta disponível e clique em **Connect**:



8. Verifique se a conexão deu certo, tentando enviar comandos para a placa. Por exemplo, tente digitar `help()` ou `print("Ola BIPES")`. Caso a placa não esteja respondendo, você também pode tentar reiniciar a placa pressionando o botão *RESET*, ou desconectando-a da porta USB, conectando novamente e refazendo a conexão (volte ao passo 5).



Blocos

Console

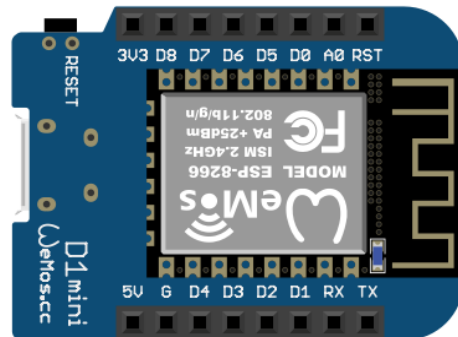
Arquivos

Compartilhado

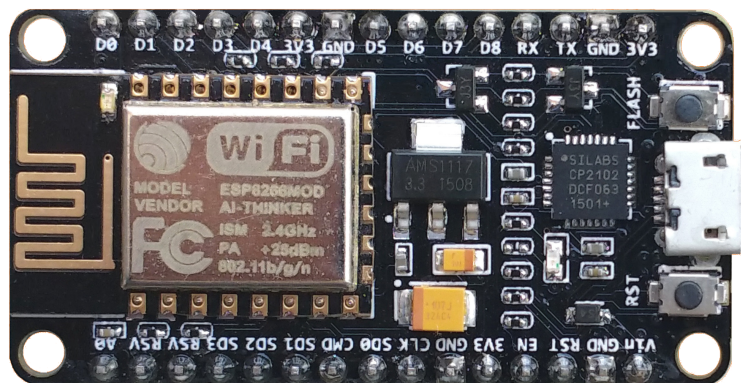
```
Connected using Web Serial API !
```

```
>>> print('Ola BIPES!')  
Ola BIPES!  
>>> █
```

Após a conexão, verifique que esta placa tem um mapa de pinos. Em especial, o LED interno da placa, indicador luminoso, que queremos ligar, está ligado ao pino D4 / GPIO2. A figura abaixo mostra um exemplo de mapa de pinos da placa WeMos D1 mini, com módulo ESP8266:

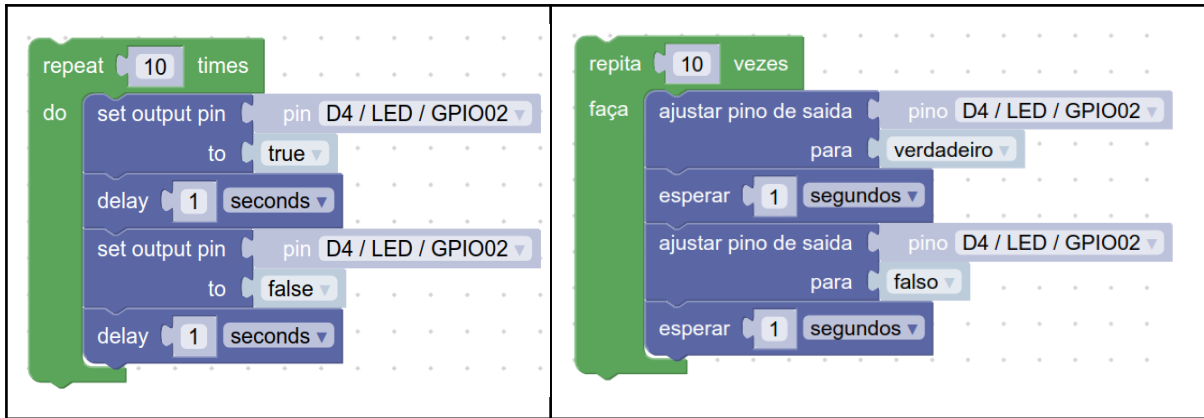


Na imagem acima observe que cada terminal, para conexão de um dispositivo externo possui uma indicação, como D8, D7, A0, D2, etc. O 5V fornece 5 Volts para circuitos externos, e o 3V3 fornece 3.3 Volts. O pino G é o Terra (*Ground*) do circuito. Note que existem dezenas de modelos / opções de placas com módulo ESP8266, de forma que você precisará identificar os pinos / LEDs, conexões da sua placa. A figura abaixo mostra outra opção de placa com módulo ESP8266:



9. Prepare o seguinte programa pelo **BIPES** (em inglês ou português). A tabela abaixo mostra o mesmo programa, na versão em português e na versão em inglês. Você só precisa preparar um deles.

Dicas: Utilize as ferramentas disponíveis em: Laços (**Loops**); Temporização (**Timing**) e Máquina (**Machine**) -> In/Out Pins; Lembre-se de sempre incluir uma espera (**delay**) para que seja possível ver as mudanças no LED entre cada ação. Sem o delay, o LED irá piscar tão rápido, que será impossível perceber com os olhos.



Observação: Devido a características do circuito interno da placa WeMos D1 mini, o LED interno desta placa acende com nível lógico baixo (0 / false) e apaga com nível lógico alto (1 / true). Esta característica pode variar de placa para placa.

10. Após preparar o programa, clique no ícone para executar (símbolo **play**):




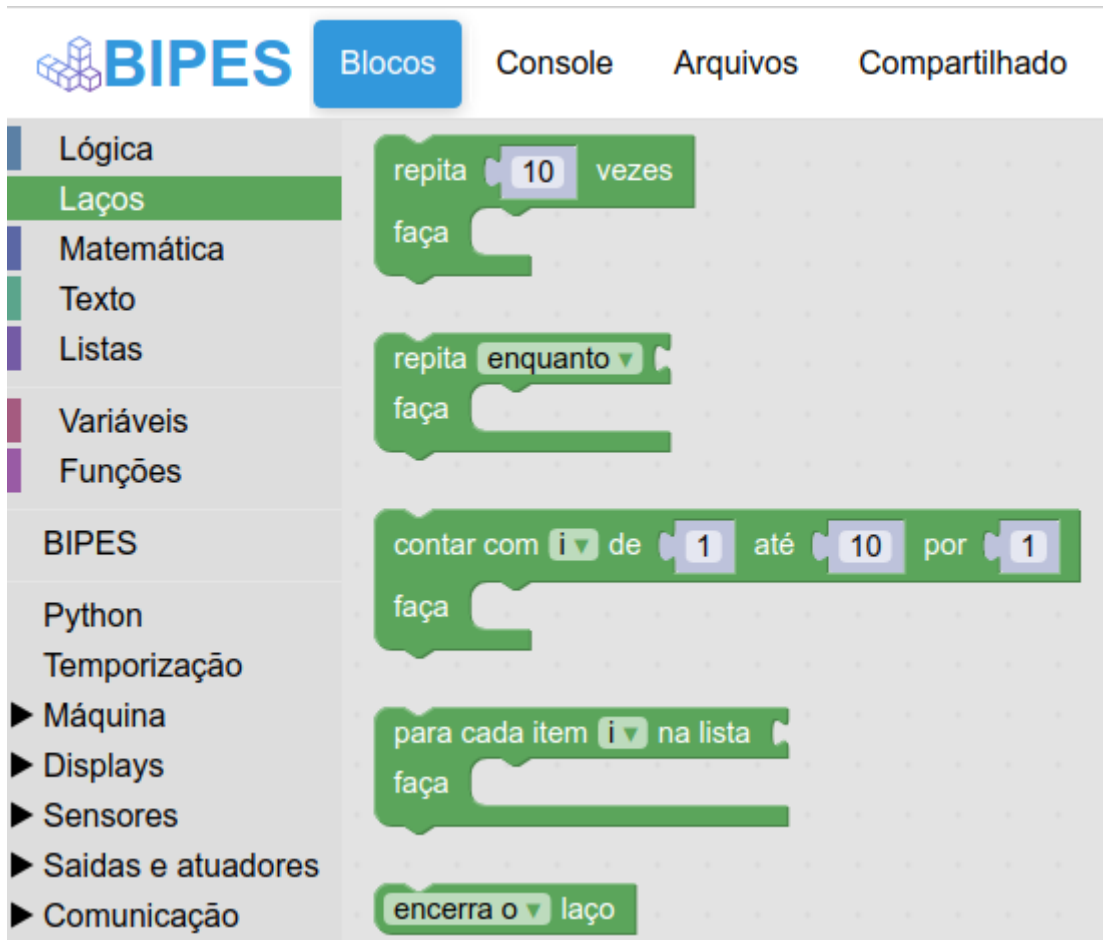
Caso queira, você pode verificar pelo console o envio do programa para a placa. O programa fará o LED piscar 10 vezes e encerrará.

Atividade:

Mude o programa para que o LED pisque com um intervalo de tempo diferente e com um número de vezes diferente.

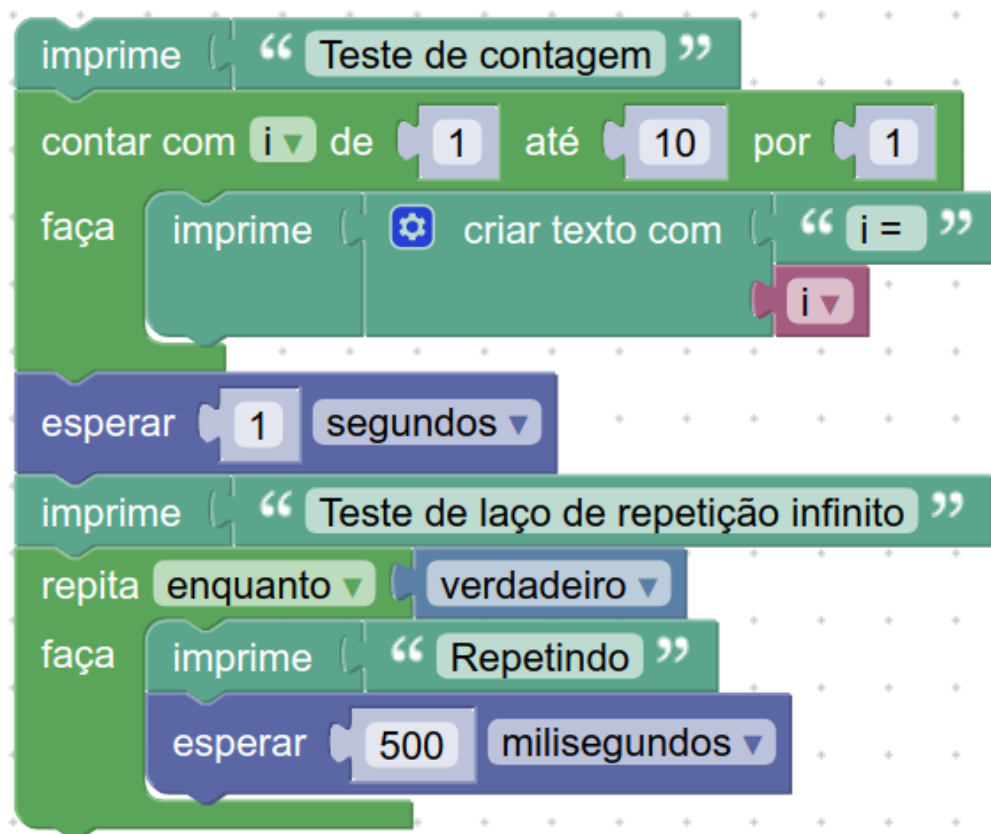
Note que utilizamos o bloco “**repeat 10 times**” / “**repetir 10 vezes**”, disponível na guia lateral **Loops / Laços** à esquerda dos blocos **BIPES** para piscar o LED. Esta mesma guia possui outras opções de estruturas de repetição, como “**count with i from 1 to 10 by 1**” / “**contar com i de 1 a 10 por 1**” (os valores podem ser alterados) e “**repeat while**” / “**repita enquanto**”.

	<p>Dica: Utilize o ícone de usuário, no canto superior direito da tela, para escolher o idioma da interface do BIPES (inglês ou português).</p>
---	---



No caso de contar de 1 a 10, cada iteração do laço atualiza a variável i (um espaço de armazenamento de dados do programa), que pode ser usada dentro da estrutura de repetição. No exemplo abaixo, o programa imprime no console de 1 a 10 com um texto inicial “ $i =$ ” para mostrar o nome da variável. Já a estrutura repetição infinita “**repita enquanto verdadeiro**”, trata-se de uma das estruturas de repetição mais usadas em sistemas embarcados, já que muitos sistemas embarcados executam uma sequência de operações de forma contínua e repetitiva enquanto o sistema está ligado. Por exemplo: ler dados de um sensor e armazenar; verificar a temperatura de um ambiente e ligar ou não o compressor de uma geladeira; dentre outros. Porém, é importante, ao criar um laço infinito, incluir um critério para interrupção do laço quando uma condição é atendida. Isso pode ser feito com o bloco “**Loops >> break out of loop**” / “**Laços >> encerra o laço**”. Deste modo é possível “desligar” ou entrar em outra rotina sem acessar o botão RESET diretamente ou executar CTRL+C no REPL. De qualquer forma, o comando CTRL+C no **Console** ou botão “Stop Running Program” irão interromper um laço infinito.

Exemplos de contagem e repetição infinita:

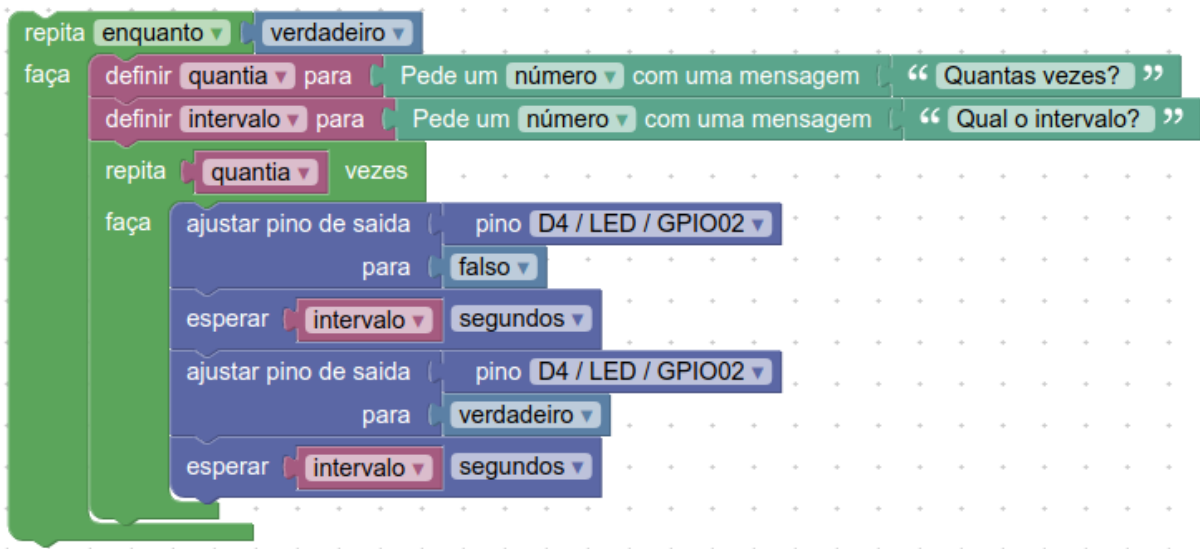


O resultado obtido no **Console** é o seguinte (clique em Stop para o laço infinito ser interrompido):

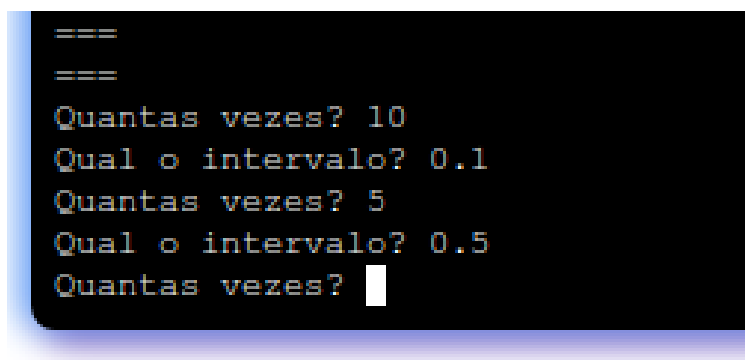
```
===
Teste de contagem
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
Teste de laço de repetição infinito
Repetindo
Repetindo
Repetindo
Traceback (most recent call last):
  File "<stdin>", line 13, in <module>
KeyboardInterrupt:
>>> □
```

[Run block based program](#) [Stop running program](#)

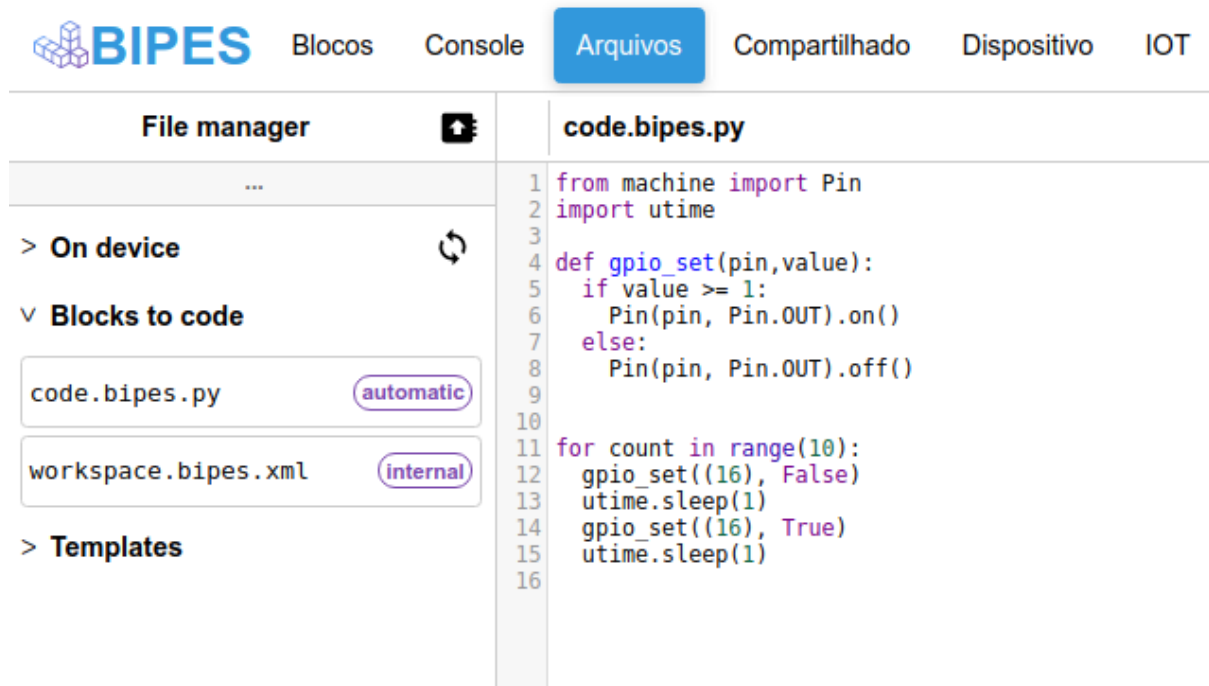
Um outro recurso útil associado ao **Console**, é a possibilidade de solicitar dados do usuário de forma interativa pelo **Console**. O bloco “**Pede um número**” pode ser utilizado para esta finalidade, podendo ser encontrado na área **Texto** da barra de ferramentas. Dessa forma, o programa abaixo apresenta uma versão interativa do programa que pisca o LED. Nesta versão, o sistema pergunta quantas vezes o usuário deseja piscar o LED e o intervalo de tempo com que o LED alterna seu estado.



A Figura abaixo mostra um exemplo de utilização do programa, onde o usuário digitou 10 para a quantia de vezes e 0.1 para o intervalo de tempo. Após o programa executar a sequência de piscar o LED 10 vezes, o programa volta a perguntar para o usuário as opções, sendo que da segunda vez o usuário digitou 5 e 0.5.



Para os curiosos: Verifique que este programa em blocos gerou um código fonte em linguagem Python (na aba **Arquivos** do **BIPES**). Você pode ver o programa gerado automaticamente na aba **Arquivos >> Blocks to code**:



The screenshot shows the BIPES interface with the 'Arquivos' tab selected. The 'File manager' on the left lists 'code.bipes.py' (automatic) and 'workspace.bipes.xml' (internal). The main area displays the Python code for 'code.bipes.py':

```
1 from machine import Pin
2 import utime
3
4 def gpio_set(pin,value):
5     if value >= 1:
6         Pin(pin, Pin.OUT).on()
7     else:
8         Pin(pin, Pin.OUT).off()
9
10
11 for count in range(10):
12     gpio_set((16), False)
13     utime.sleep(1)
14     gpio_set((16), True)
15     utime.sleep(1)
16
```

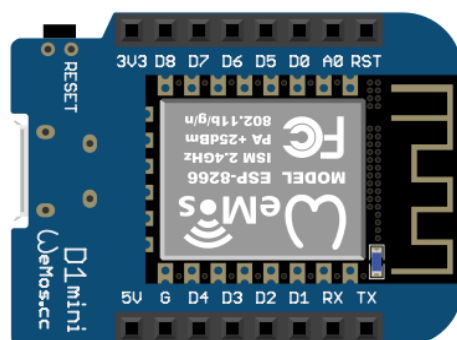
Avançado: Caso queira, você pode editar o arquivo em Python, salvar (botão “**Save a copy**” / “**Salvar uma cópia**”) e rodar ele a partir da edição textual. Como mencionamos, a placa ESP8266 utiliza MicroPython, uma versão de Python otimizada para sistemas embarcados e microcontroladores. Além disso, você pode salvar este arquivo como “**main.py**”, pela própria aba Arquivos do **BIPES**. Caso ele seja salvo com este nome, este programa será executado automaticamente toda vez que a placa for ligada, de forma autônoma e independente. Ou seja, sua solução ficará salva e embarcada na placa e não dependerá de um computador para funcionar! Caso queira testar, você pode ligar a placa em um power bank ou na tomada, e verificar que o programa / sistema funciona de forma autônoma.

Entrada digital e verificação de uma condição

Os microcontroladores, em geral, possuem dois tipos principais de entrada: as digitais e as analógicas. Estas entradas costumam estar associadas a pinos de propósito geral GPIO (*General Purpose Input Output*) indicados nas placas como GPIOX, onde X é o número da porta ou com códigos como D0 ou D1. Ao escolher o modelo de placa no BIPES, os pinos desta placa serão automaticamente listados para escolha, facilitando a programação, conforme a figura abaixo:



Note que esta lista de pinos estará associada aos pinos físicos da placa:

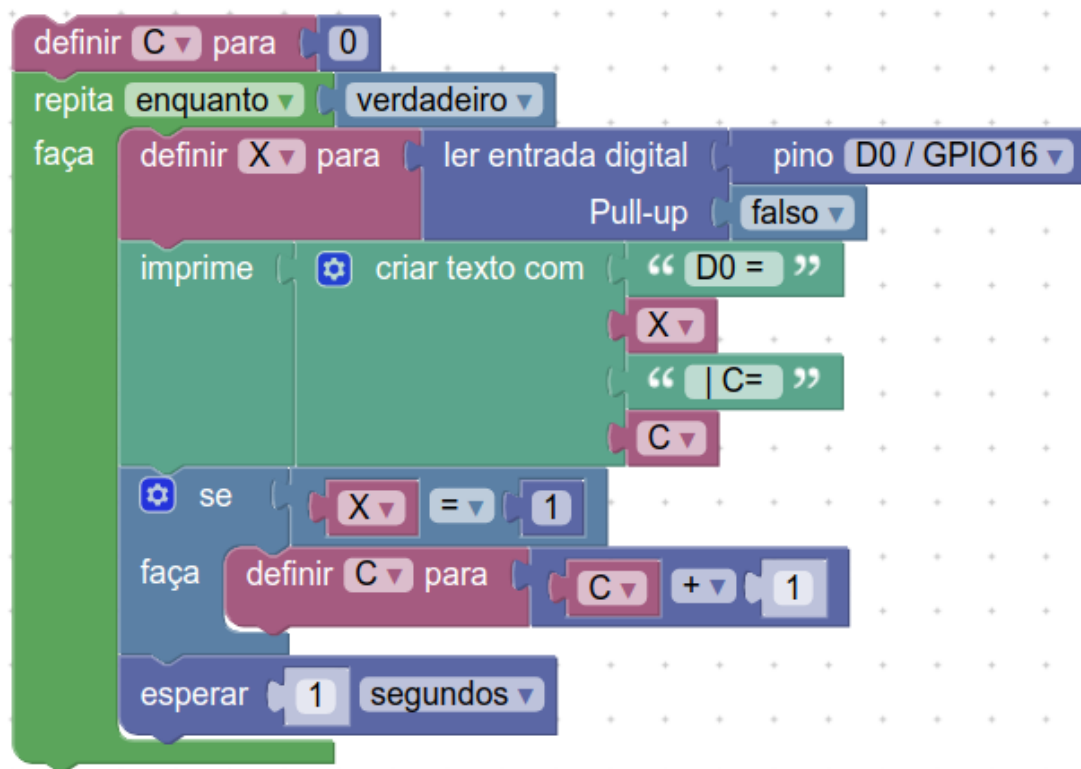



No caso do modo digital, estes pinos podem ser usados para acionar dispositivos de saída, podendo assumir dois estados: [ligado / true / 1] ou [desligado / false / 0]. Assim,

diversos dispositivos podem ser conectados e controlados por estes pinos. Os mesmos pinos também podem ser utilizados no modo entrada, de forma a realizarem a “leitura” de sinais de entrada do tipo digital: [ligado / true / 1] ou [desligado / false / 0].

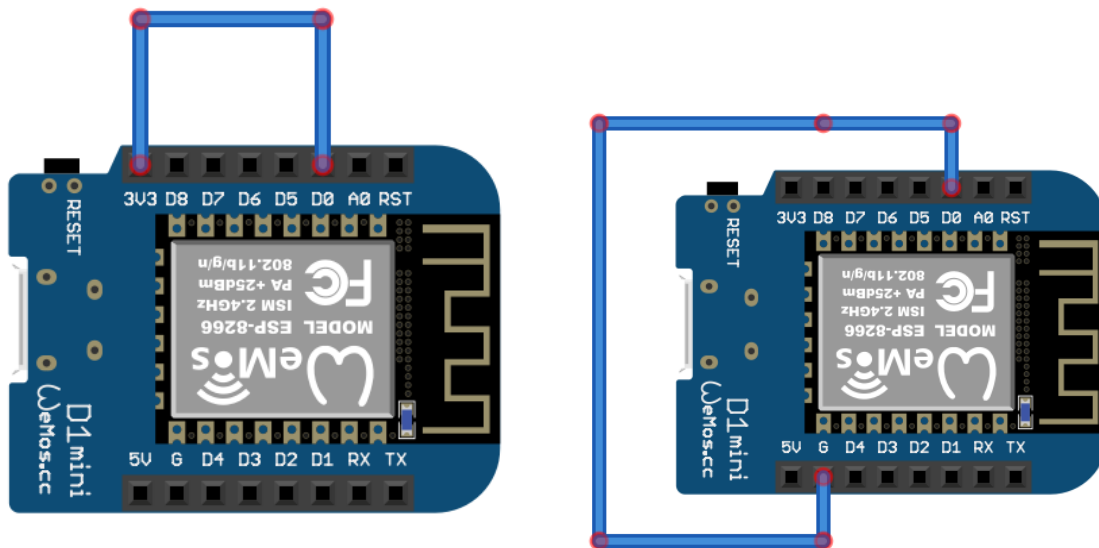
Como o módulo ESP8266 funciona com 3.3 Volts, uma diferença de potencial de 3.3V entre pino referido e a referência (G) representa o estado lógico 1, enquanto uma diferença de potencial nula (0V) representa o estado lógico 0.

O programa abaixo mostra um exemplo que une a entrada digital a uma condição (**se**), de forma que uma variável (espaço de armazenamento do programa) tem seu valor incrementado de uma unidade a cada 1 segundo caso o estado deste pino seja 1 (acionado). O programa inicia ajustando o valor da variável C, um espaço temporário de armazenamento, para 0 [**definir C para 0**]. Em seguida, o programa segue para uma estrutura de repetição em **laço infinito [repita enquanto verdadeiro]**. Dentro desta estrutura de repetição, a variável X recebe a leitura do pino digital D0 [**definir X por para entrada digital pino D0**], de forma que a variável X irá receber o valor 0 se o pino D0 tiver 0 Volts ou 1 se o pino D0 indicar 3.3 Volts. Na sequência, o programa mostra no **Console** o valor da variável X e da variável C, no formato “D0 = n | C = n”, através do comando **imprime**. Em seguida, a instrução “**se**” verifica se a variável X, que possui a leitura do pino digital, é igual a 1, e caso seja, incrementa em 1 o valor de C. Finalmente, o programa aguarda 1 segundo e reinicia a sequência descrita aqui.



	Verifique que o bloco “ criar texto com ” possui uma pequena engrenagem azul. Ao clicar nela, é possível configurar o número de entradas que este bloco recebe. Ao longo das próximas práticas, utilize este botão azul para configurar opções dos blocos sempre que necessário.
---	---

A figura a seguir mostra uma forma simples de testar o programa. Inicie a execução do programa e use cabo simples (pedaço de fio) ou *jumper cable*, para alternar o valor inserido no pino configurado no programa (D0) conectando-o ao pino 3V3 (nível lógico 1) e posteriormente ao pino G (terra, nível lógico 0). Note a mudança na aba **Console** e o contador aumentando a cada segundo quando o nível lógico recebido é 1.



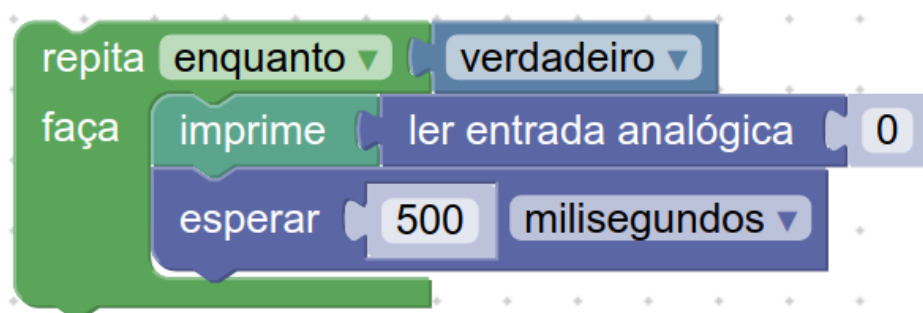
Exemplo de resultado que pode ser visto no **Console**:

```
D0 = 1 | C = 391
D0 = 1 | C = 392
D0 = 1 | C = 393
D0 = 1 | C = 394
D0 = 1 | C = 395
D0 = 1 | C = 396
D0 = 1 | C = 397
D0 = 1 | C = 398
D0 = 1 | C = 399
D0 = 1 | C = 400
D0 = 1 | C = 401
D0 = 1 | C = 402
D0 = 1 | C = 403
D0 = 1 | C = 404
D0 = 1 | C = 405
```

Com base neste exemplo, diversas aplicações podem ser feitas: como contar número de pessoas ou veículos que passaram por um portal, verificar se uma porta está aberta ou fechada, verificar a presença de pessoas por sensores de alarme, verificar o nível de uma caixa d'água (cheia ou não) com um sensor digital do tipo boia, dentre outras possibilidades.

Entrada analógica e sensor de luz (LDR)

Uma outra possibilidade de leitura de entradas externas, é por meio de pinos de entrada analógica. Enquanto os pinos digitais só detectam sinais **0** ou **1**, os pinos analógicos são sensíveis a valores variáveis da medida dentro de um intervalo pré-definido. Por exemplo, eles podem medir a tensão, em Volts, temperatura ou outros valores que variam entre várias possibilidades. A ESP8266 possui apenas uma entrada analógica (pino A0), que pode medir valores de entrada entre 0 e 3.3 Volts, internamente mapeados entre 0 e 1023, tal que 0 equivalente a 0 Volts e 1023 equivalente a 3.3 Volts¹. O ESP32, por sua vez, já possui diversos pinos de entradas analógicas, que podem ser usados simultaneamente. Para utilizar a entrada analógica do módulo ESP8266, é possível usar o seguinte programa (o bloco **ler entrada analógica**, que faz a leitura do pino analógico, está disponível na guia lateral **Máquina >> In/Out Pins**):



Ao executar o programa acima, será possível ver os valores variando entre 0 e 1023 na aba **Console**. Você pode utilizar um *jumper cable* e conectar o pino A0 ao G e observar a leitura 0. Depois conecte o pino A0 ao pino 3V3 e note que um valor próximo de 1023 deve ser apresentado. Atenção: os pinos da placa ESP8266 suportam no máximo 3.3 Volts. Não aplique valores maiores de tensão elétrica, sob risco de danificar ou queimar o módulo.

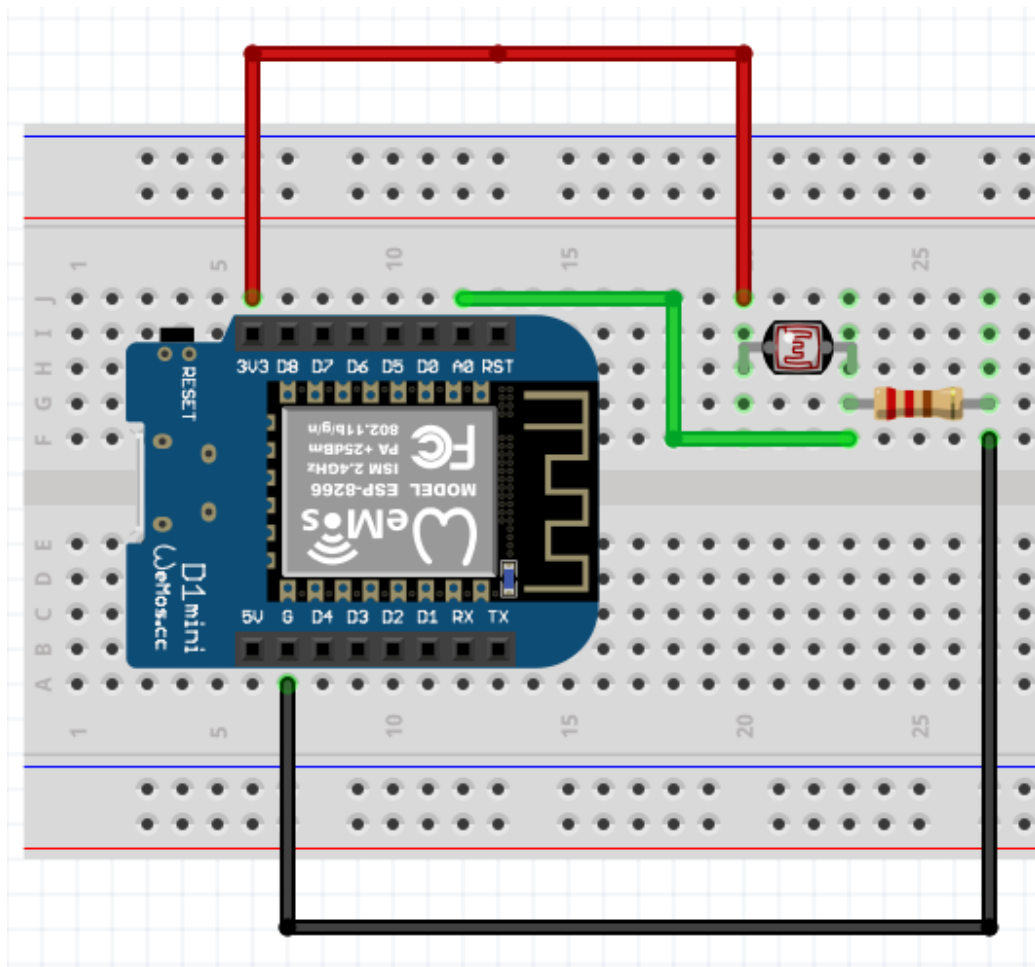
Atividade:

Considerando que a leitura 1023 corresponde a 3.3 Volts, modifique o programa acima para que ele mostre, na aba **Console**, o valor em Volts aplicado ao pino A0 do módulo.

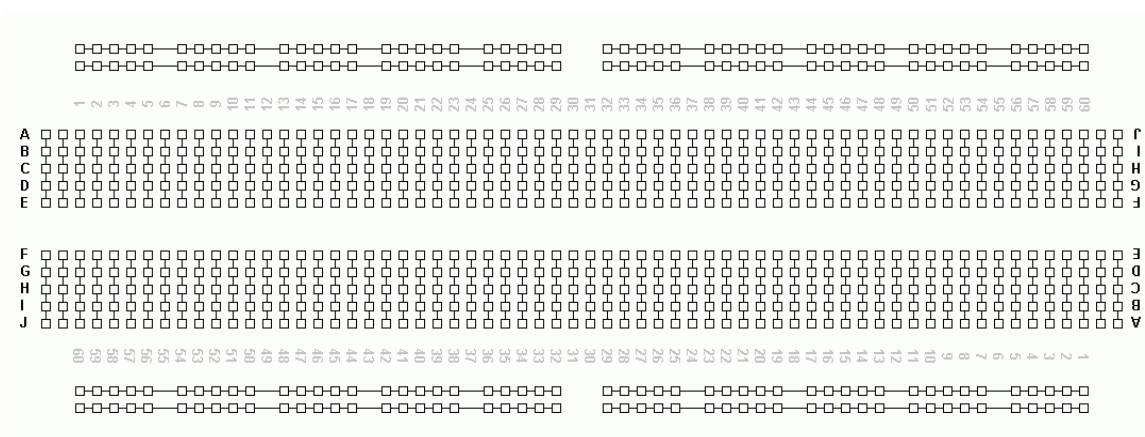
Agora vamos utilizar um sensor LDR (*Light Dependent Resistor*), que varia sua resistência de acordo com a iluminação, através de uma entrada analógica do módulo ESP8266. Faça a montagem conforme a próxima figura, utilizando uma placa de prototipação (*protoboard*).

¹ O comando **ler entrada analógica** realiza, de fato, uma conversão do valor do sinal de tensão analógico na entrada para um valor digital de 10 bits, uma vez que um microcontrolador é capaz de processar apenas sinais digitais.

Montagem da placa com módulo ESP8266, sensor de luz LDR e um resistor de 220 ohms (cores: vermelho, vermelho, marrom).

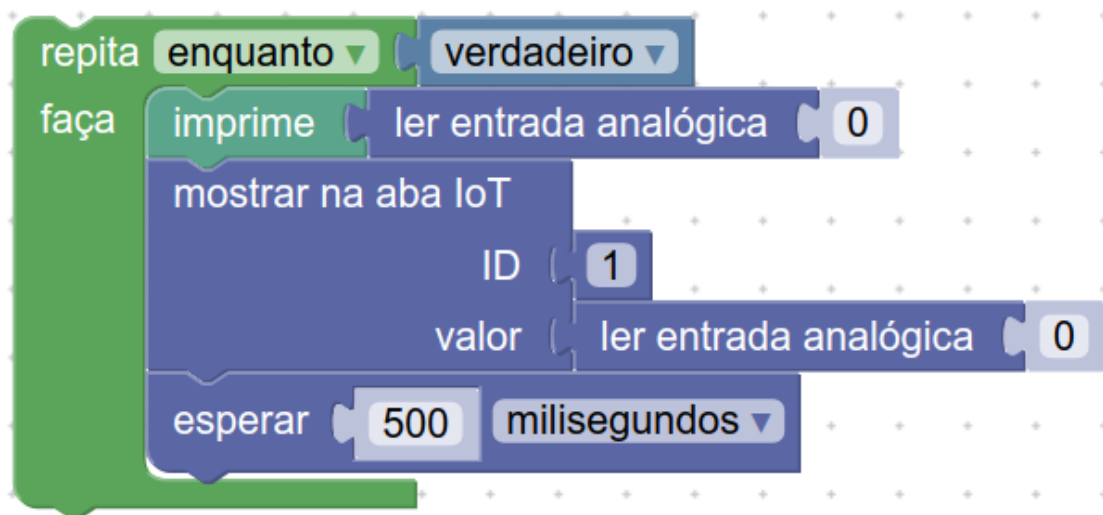


Observação: Note, na figura abaixo, que a *protoboard* realiza a interconexão de componentes entre colunas em seu centro e entre linhas nas extremidades. Assim, todos componentes conectados na mesma coluna, estarão, automaticamente, com seus terminais interconectados.

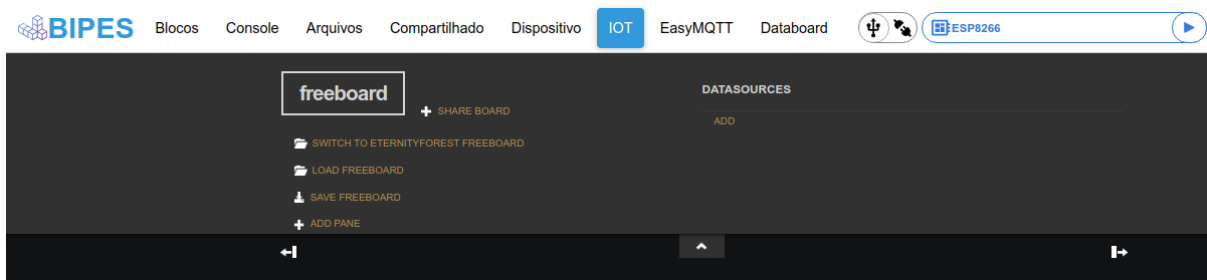


A montagem apresentada explora uma configuração eletrônica chamada divisor de tensão, de forma que a adequada seleção de 2 resistores ligados em série pode configurar o valor da tensão de saída no terminal comum entre os dois resistores. No caso deste circuito, o LDR é um resistor variável, e conforme a iluminação muda, a resistência do LDR também muda, alterando a tensão de saída, que é medida pelo pino A0 (fio verde). O cabo vermelho (V) é ligado ao 3V3 da placa ESP8266, alimentando o circuito. O cabo de saída (verde) é conectado na entrada analógica da placa (AIN / ADC0) e o fio preto conecta a outra ponta do resistor de 220 ohms ao pino terra da placa (G ou GND), fechando o circuito. Agora podemos usar o programa criado para analisar a variação de iluminação sobre o LDR.

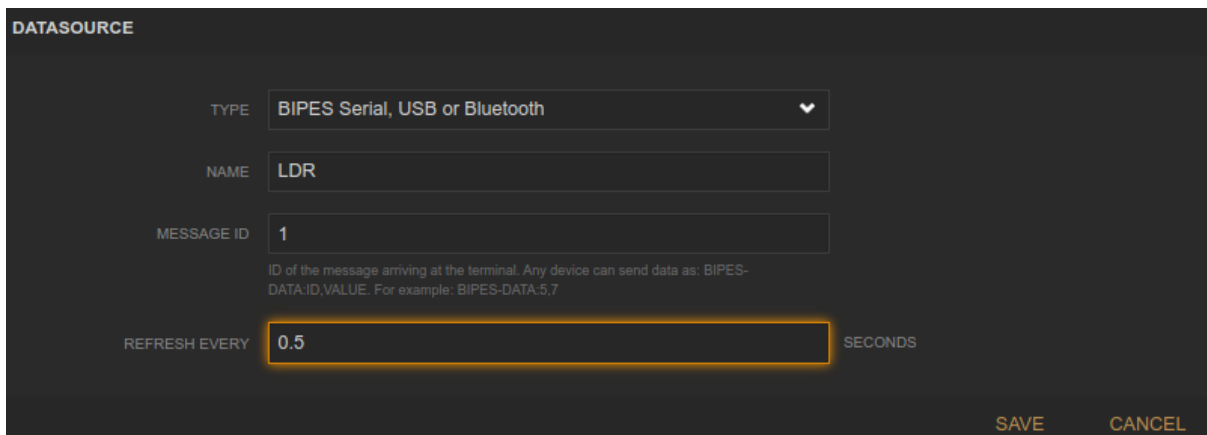
O programa acima irá “ler” a entrada analógica e mostrar no **Console** do **BIPES**. Mas podemos avançar mais com esta montagem e visualizar as leituras de forma gráfica. Para isso, adicione o bloco **mostrar na aba IOT**:



Agora utilize a aba **IOT** do **BIPES**:

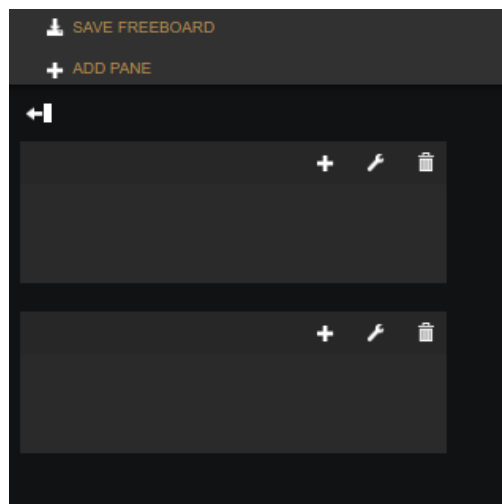


Em **DATASOURCES**, clique em **ADD** e escolha “**BIPES Serial, USB or Bluetooth**”:

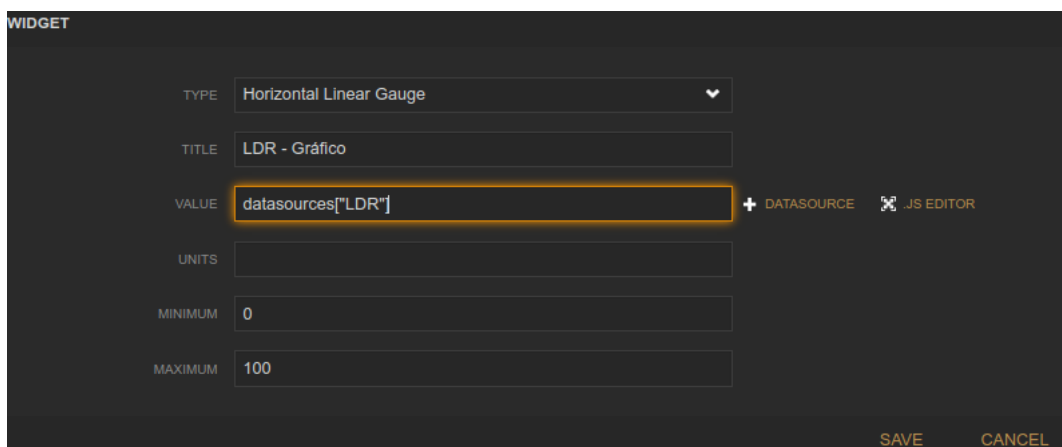


The screenshot shows the 'DATASOURCE' configuration interface. It includes a dropdown menu for 'TYPE' set to 'BIPES Serial, USB or Bluetooth', a text input for 'NAME' with the value 'LDR', and another text input for 'MESSAGE ID' with the value '1'. Below these is a small explanatory text: 'ID of the message arriving at the terminal. Any device can send data as: BIPES-DATA:ID,VALUE. For example: BIPES-DATA:5,7'. There is a 'REFRESH EVERY' field with '0.5' and 'SECONDS' next to it. At the bottom right, there are 'SAVE' and 'CANCEL' buttons.

Salve e adicione um ou mais “*panes*” (painéis do *dashboard*), com o botão **ADD PANE**:

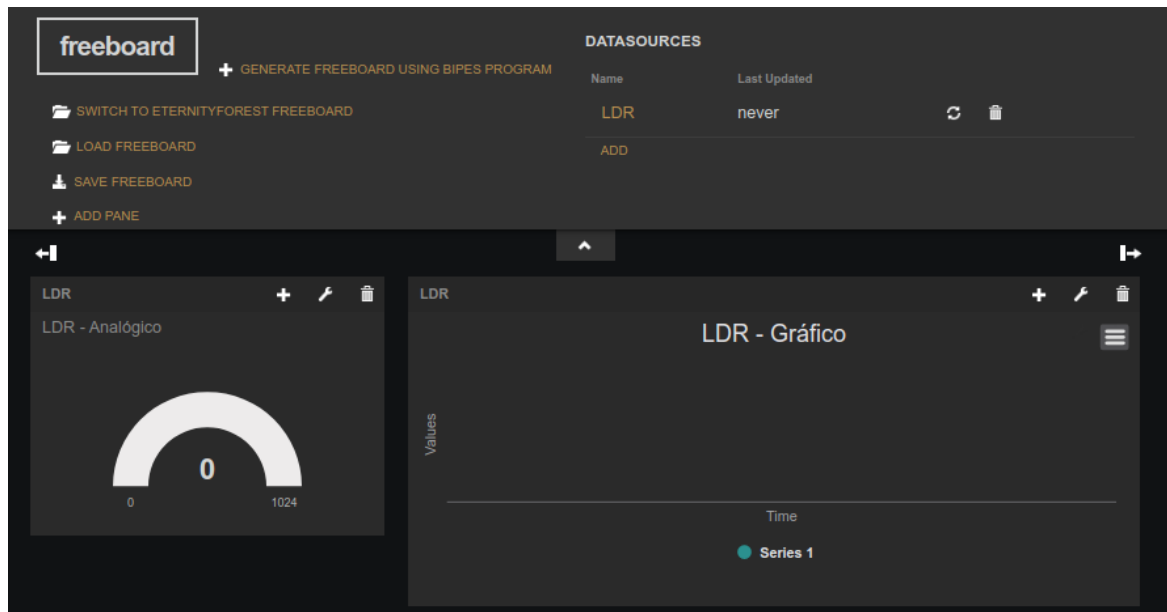


Arraste e solte os “*panes*” para ajustar suas preferências. Em cada “*pane*”, adicione componentes de visualização (clcando no ícone +), como você preferir, e escolha no campo **VALUE** a fonte dos dados para visualização: “`datasources[“LDR”]`”:



The screenshot shows the 'WIDGET' configuration interface. It includes a dropdown menu for 'TYPE' set to 'Horizontal Linear Gauge', a text input for 'TITLE' with the value 'LDR - Gráfico', and a text input for 'VALUE' with the value 'datasources[“LDR”]'. To the right of the 'VALUE' field are buttons for '+ DATASOURCE' and 'JS EDITOR'. Below these are text inputs for 'UNITS', 'MINIMUM' (set to '0'), and 'MAXIMUM' (set to '100'). At the bottom right, there are 'SAVE' and 'CANCEL' buttons.

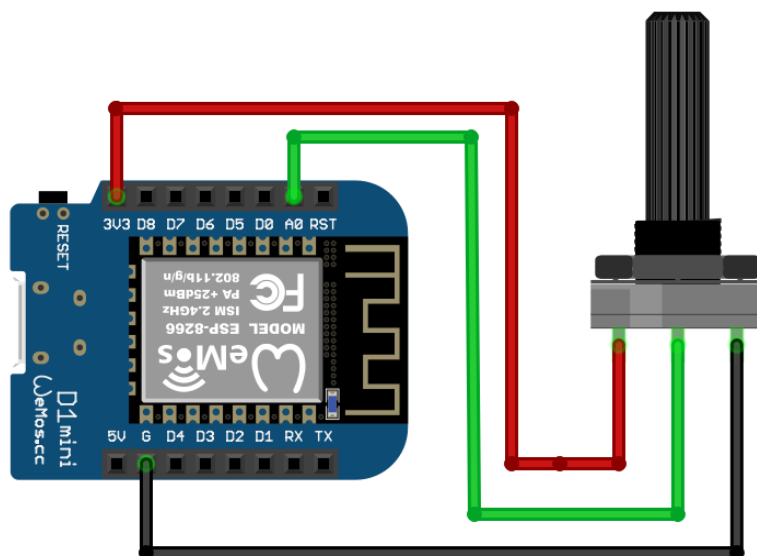
No exemplo abaixo selecionamos um indicador gráfico do tipo **Gauge** e um do tipo **Time series (Highcharts)**:



Execute o programa e verifique a visualização. Ao variar a iluminação que incide sobre o sensor LDR, as leituras vão mudar e serão exibidas em tempo real nos gráficos que você inseriu no painel.

Atividade:

Troque o LDR por um potenciômetro, e verifique a mudança nos componentes conforme você gira o eixo do potenciômetro. Discuta / pesquise o que é / como funciona o potenciômetro. Sugestão de montagem:

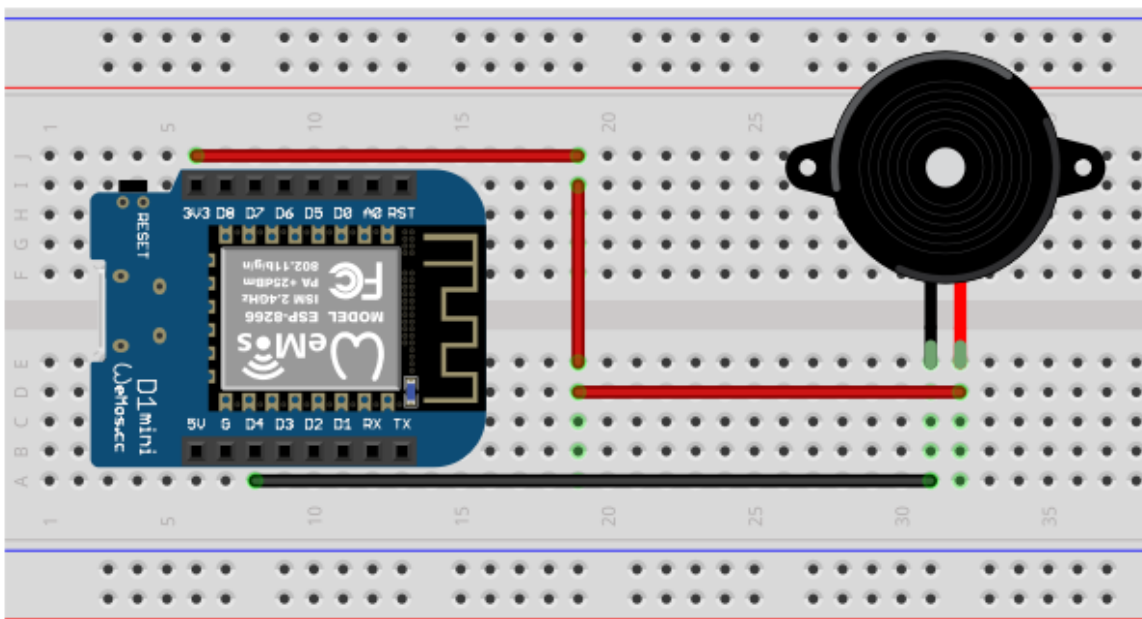


Data e hora (RTC)

Através do MicroPython, instalado nas placas ESP8266 ou ESP32, é possível utilizar um recurso chamado *Real Time Clock* (RTC). O RTC permite que o sistema configure a data e hora, e mantenha um registro atualizado da passagem do tempo. O RTC interno do sistema pode ser iniciado de várias formas: manualmente, a partir de um RTC externo com bateria, que fornece data e hora corretas, mesmo se o sistema for desligado, ou pela Internet, utilizando o recurso / bloco **Network Time Protocol NTP**.

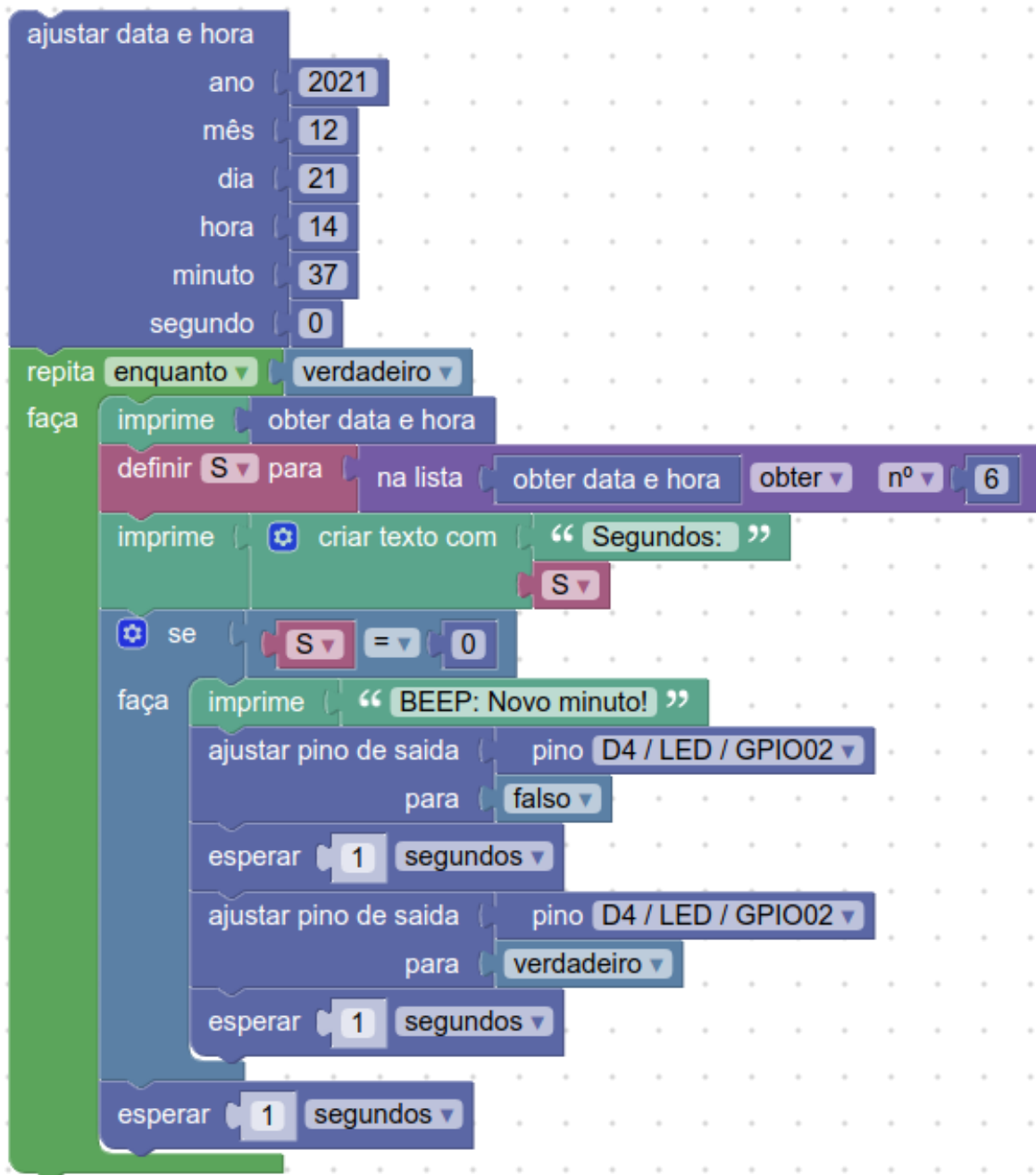
O exemplo a seguir mostra um programa que inicia o RTC com a data/hora de 21/12/2021 às 14:37:00. Note, que o programa trata a data e hora como uma estrutura de lista, separada por vírgulas: **(2021, 12, 21, 1, 14, 37, 00, 0)**². É possível então, usar o bloco **na lista** para pegar o 6º elemento da lista (a contagem inicia em 0), que corresponde aos segundos, e armazenar este valor na variável S. Em seguida, verifica-se se a variável S é igual a 0, e quando isso ocorrer, o LED é acionado por 1 segundo. **Dica:** Um *buzzer* pode ser conectado entre o pino 3V3 e o pino digital do LED (D4) e então um sinal sonoro com duração de 1 segundo também seria emitido a cada 1 minuto. A figura abaixo mostra uma possibilidade de conexão de um *buzzer* utilizando os pinos 3V3 ligado ao + do *buzzer* e o D4 ao outro terminal do *buzzer*.

Montagem:



² (ano, mês, dia, dia da semana, hora, minuto, segundo, milissegundo)

Programa:



O resultado pode ser visto no **Console**:

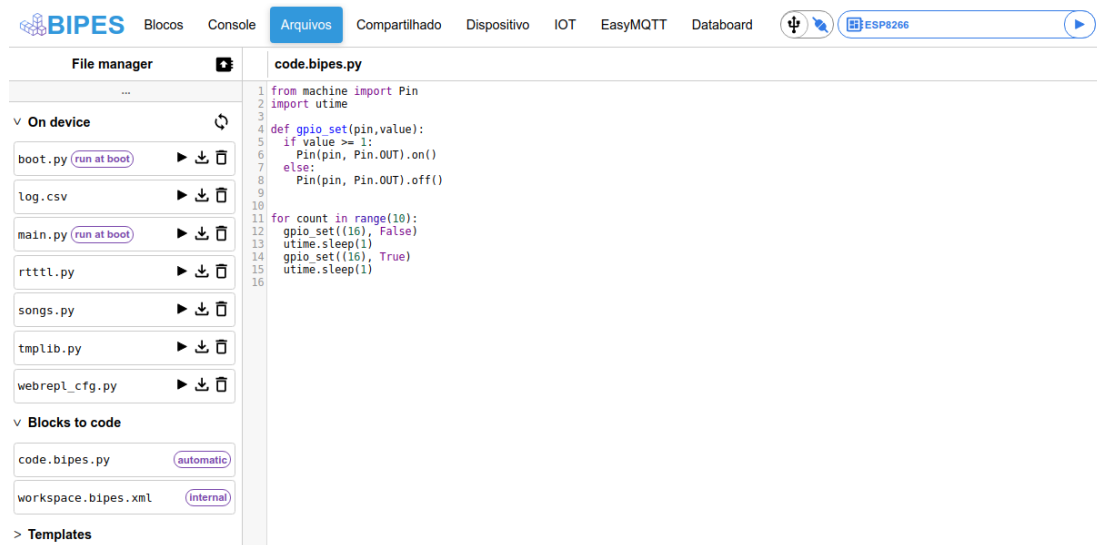
```
===  
Segundos: 54  
(2021, 12, 21, 1, 14, 37, 55, 222)  
Segundos: 55  
(2021, 12, 21, 1, 14, 37, 56, 226)  
Segundos: 56  
(2021, 12, 21, 1, 14, 37, 57, 231)
```

Segundos: 57
(2021, 12, 21, 1, 14, 37, 58, 236)
Segundos: 58
(2021, 12, 21, 1, 14, 37, 59, 241)
Segundos: 59
(2021, 12, 21, 1, 14, 38, 0, 245)
Segundos: 0
BEEP: Novo minuto!
(2021, 12, 21, 1, 14, 38, 3, 233)
Segundos: 3
(2021, 12, 21, 1, 14, 38, 4, 232)
Segundos: 4
(2021, 12, 21, 1, 14, 38, 5, 227)
Segundos: 5
(2021, 12, 21, 1, 14, 38, 6, 225)
Segundos: 6
(2021, 12, 21, 1, 14, 38, 7, 222)
Segundos: 7
(2021, 12, 21, 1, 14, 38, 8, 220)
Segundos: 8
(2021, 12, 21, 1, 14, 38, 9, 215)
Segundos: 9






Note que, mesmo com a **espera** de 3 segundos (2 segundos dentro do **SE** e 1 segundo no laço de repetição), a data e hora não foram perdidas. Isso ocorre, pois o RTC mantém um registro atualizado da data e hora, independente do funcionamento do programa. Sempre que o bloco **obter data e hora** é usado, a data e hora atuais serão obtidas, a partir do seu ajuste inicial. Posteriormente, você pode utilizar um RTC externo, que, com apoio de uma bateria, manterá a data e hora sempre atuais, ou obter a data e hora da Internet através do bloco NTP.


Arquivos na placa

Se você já usou Arduino, deve lembrar que no Arduino não há o conceito de arquivos na placa. Já no caso do **BIPES** com MicroPython, cada placa pode possuir diversos arquivos, que podem ser programas, dados, configurações ou qualquer outra informação que você queira gravar em arquivos. O gerenciamento de arquivos no **BIPES** é bastante simples e pode ser feito através da aba **Arquivos**. Todo este ambiente de gerenciamento de arquivos pode ser utilizado através de cabo USB ou Wi-Fi, caso a placa esteja conectada a uma rede Wi-Fi. A figura a seguir mostra um exemplo da aba de arquivos do **BIPES**:



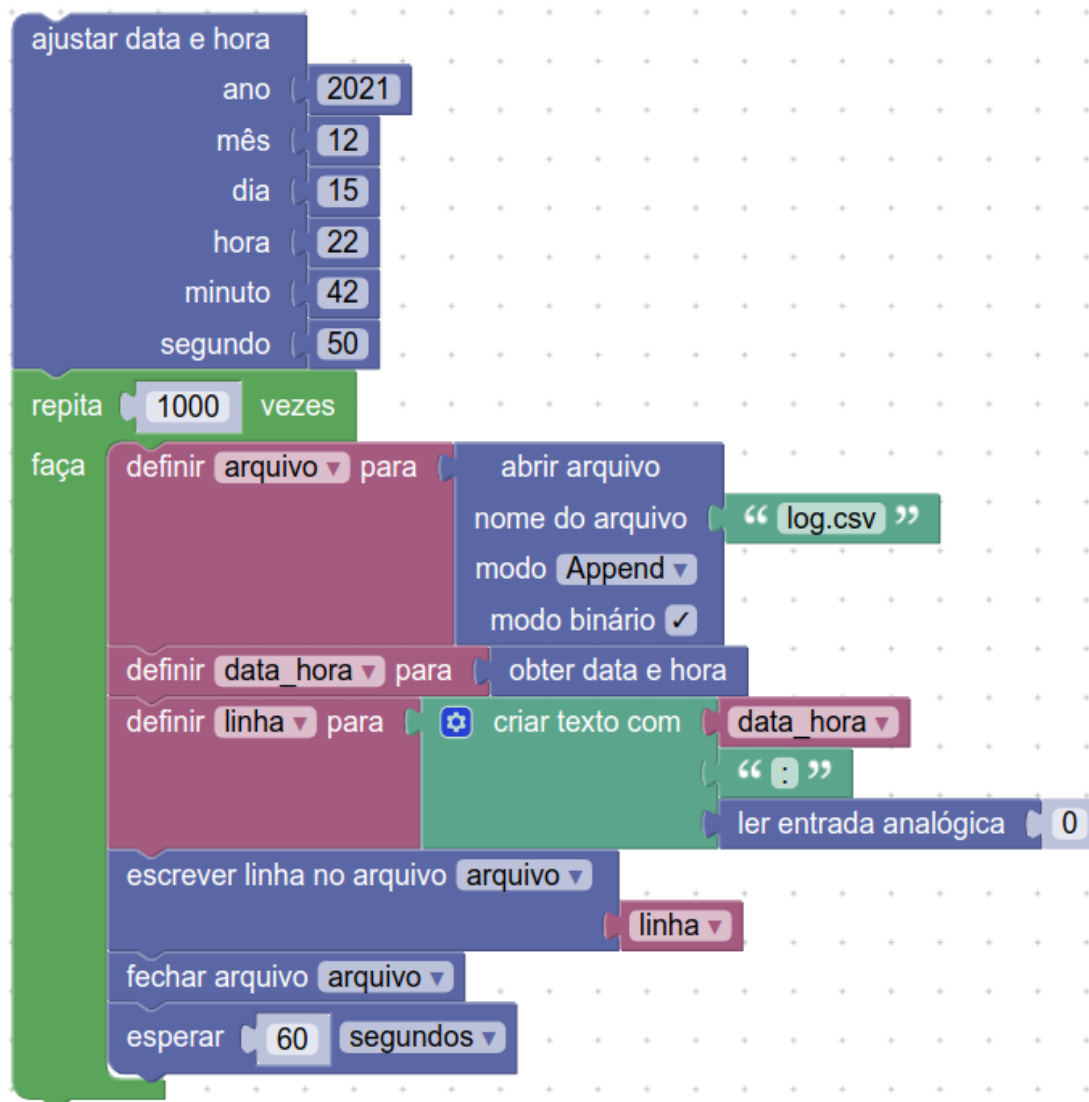
A tabela abaixo explica cada função disponível nesta guia do BIPES:

	Upload script to device Envia um arquivo do seu computador para a placa
 Save a copy	Save a copy Salva, na placa, uma cópia do arquivo com nome personalizado
	Refresh device file list Atualiza a listagem de arquivos
	Run file Executa um programa em Python armazenado naquele arquivo Observação: Também é possível utilizar a tecla de atalho do teclado Ctrl+Shift+R para iniciar a execução do programa e Ctrl+Shift+S para interromper, a partir de qualquer guia do BIPES.
	Download file Faz o download do arquivo da placa para seu computador

	Delete file Apaga o arquivo da placa
<code>code.bipes.py</code>	File name Nome do arquivo a ser salvo pelo comando Save a copy . É possível clicar nesta área e editar o nome do arquivo. Assim é possível salvar cópias ou salvar modificações no mesmo arquivo.

Na área **Blocks to Code**, é possível ver o programa gerado automaticamente a partir dos blocos do **BIPES**. Você pode editar este programa e salvar cópias na placa.

Além de armazenar programas e configurações, também é possível utilizar arquivos para armazenar dados coletados, sem a necessidade de conexão com a Internet. O programa abaixo realiza a leitura da entrada analógica e a armazena no arquivo **log.csv** de forma associada à data e hora de leitura, com coleta de dados a cada 60 segundos. Tal programa poderia ser utilizado para algum sistema de coleta de dados ambientais, por exemplo.



```

ajustar data e hora
  ano 2021
  mês 12
  dia 15
  hora 22
  minuto 42
  segundo 50

repita 1000 vezes
  faça
    definir arquivo para abrir arquivo
      nome do arquivo "log.csv"
      modo Append
      modo binário [checked]
    definir data_hora para obter data e hora
    definir linha para criar texto com
      data_hora
      ":"
      ler entrada analógica 0
    escrever linha no arquivo arquivo
      linha
    fechar arquivo arquivo
    esperar 60 segundos
  
```

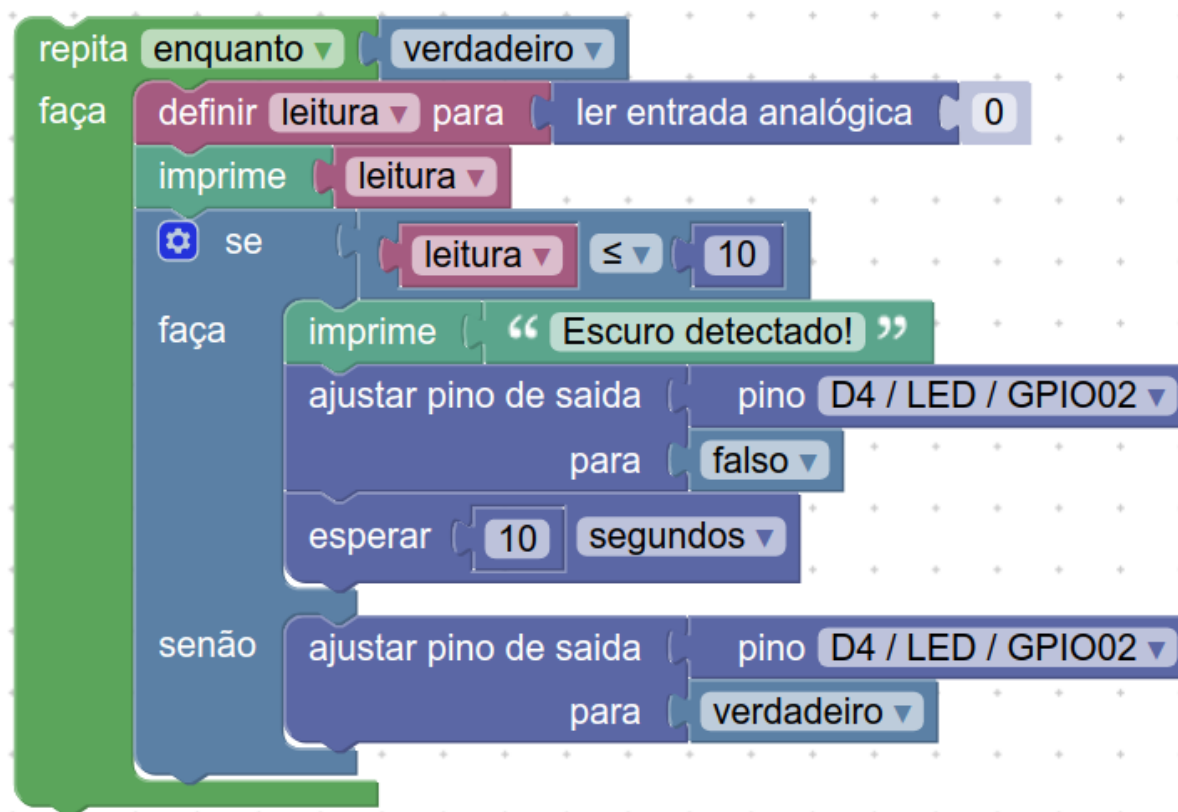

Os dados coletados ficam armazenados na memória FLASH da placa, no formato de arquivos, e são mantidos, mesmo que falte energia. Ao conectar a placa ao PC com **BIPES**, é possível fazer o download do arquivo *log.csv*, que possuirá o seguinte formato:

```
(2021, 12, 15, 2, 22, 42, 50, 3):8  
(2021, 12, 15, 2, 22, 43, 50, 60):8  
(2021, 12, 15, 2, 22, 44, 50, 119):8  
(2021, 12, 15, 2, 22, 45, 50, 173):8  
(2021, 12, 15, 2, 22, 46, 50, 228):8
```

Como se pode observar, a data e hora são armazenadas, e depois a leitura da entrada analógica é salva após os dois pontos “:”, conforme instruído no programa pelo bloco **definir linha para criar texto com data_hora : ler entrada analógica 0**. Caso seja necessário, o MicroPython também permite que estas operações sejam feitas em um cartão de memória do tipo SD Card, permitindo um maior espaço de armazenamento para coleta de dados.

Verificando uma condição (Parte 2)

Já utilizamos a estrutura condicional **SE** para tomar uma decisão com base em uma entrada digital. Agora discutimos a tomada de decisão com base na leitura de um sensor analógico. Por exemplo, acionar um dispositivo quando um sensor detectar algum nível pré-determinado. No caso do circuito com o LDR, um exemplo seria acionar o LED por 10 segundos quando o sensor detectar que ficou escuro. Para isso definimos “escuro” quando o LDR fornecer uma medida menor que 10 (dentro da faixa de 0 a 1023). Exemplo:



Teste o programa e tente ajustar diferentes níveis de detecção de luz.

Atividade:

Faça o LED piscar enquanto a iluminação está abaixo do nível especificado, e parar de piscar quando estiver acima deste nível.

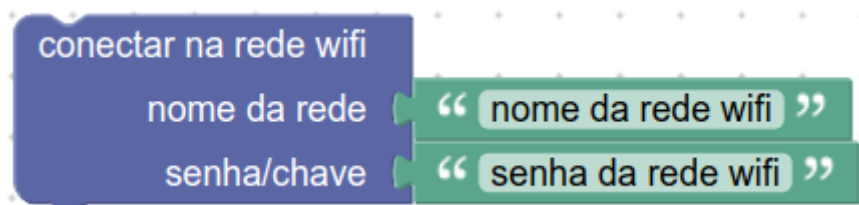
Teste também o sistema desconectado do PC, ou seja, ligado em um Power Bank ou fonte de celular. Note que o programa está no ESP8266 e não no PC!

Listar redes Wi-fi

Nas próximas atividades, conectaremos nossa placa à Internet! O primeiro passo é listar as redes existentes. Tente você mesmo. As opções de rede estão no final da barra de blocos, na opção “**Rede e Internet**”. O seguinte programa lista na aba **Console** as redes Wi-fi disponíveis:



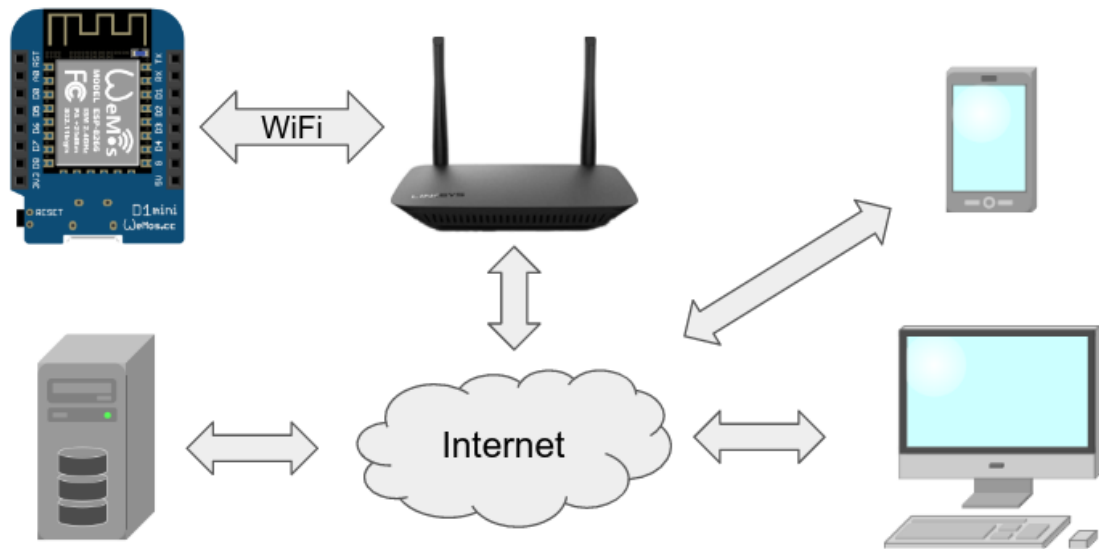
Conectando na Internet



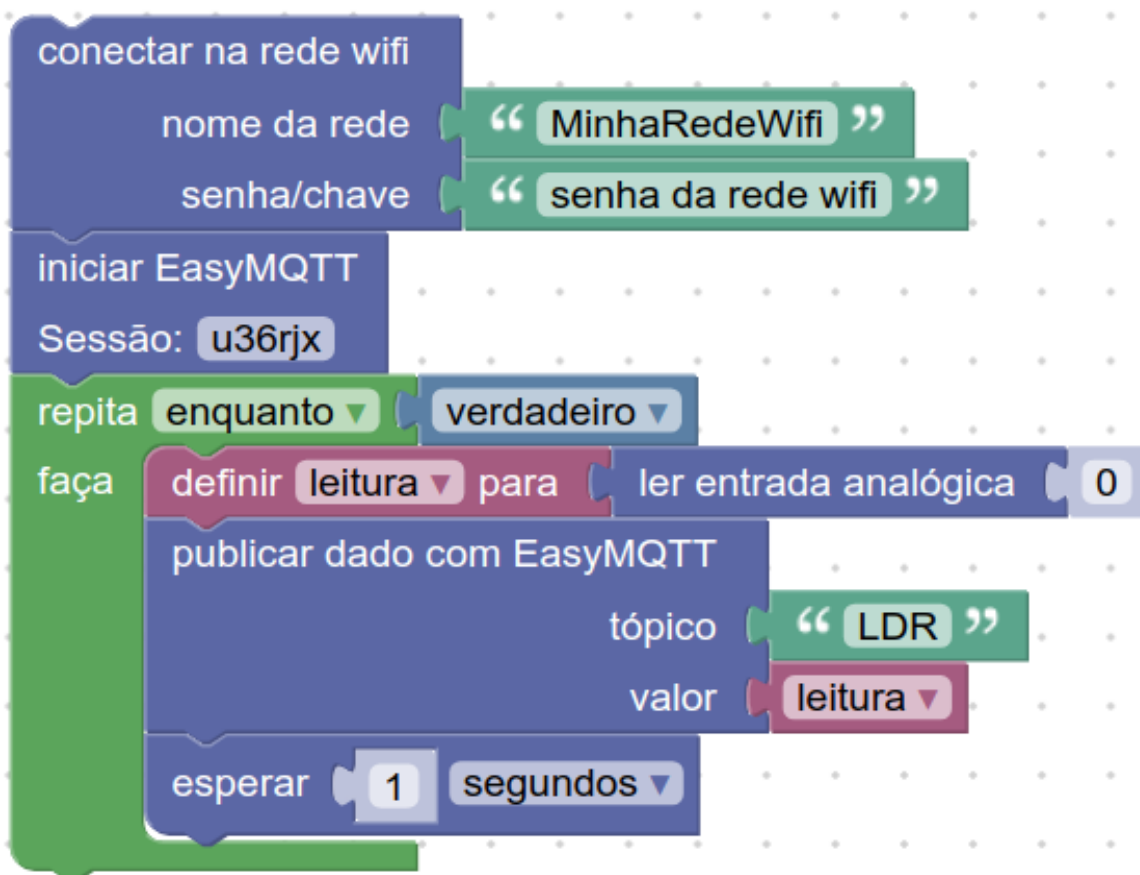
Utilize o bloco acima para que a placa se conecte à Internet e informe o nome da rede e senha. Tente rodar o programa para que a placa se conecte à Internet. Após listar as redes e conectar sua placa na Internet, já podemos enviar dados para a nuvem!

Envio de dados para a Internet / “Nuvem”

A Figura abaixo ilustra um cenário típico de uma aplicação que realiza o envio de dados para um computador remoto, possuindo software adequado para recepção, armazenamento e disponibilização destes dados recebidos. Este computador é chamado de **servidor**, sendo representado na parte inferior esquerda da figura e, graças à Internet, pode estar em qualquer parte do mundo. Este **servidor** pode ser uma máquina física ou virtual, podendo também estar ativo em serviços de “nuvem”, como o *Google Cloud Computing* (GCC) ou *Amazon Web Services* (AWS). O servidor do projeto BIPES, por exemplo, está hospedado na plataforma de nuvem do Google. A figura ainda mostra que, graças à conectividade da Internet, diversos dispositivos, como computadores, *tablets* e *smartphones* podem acessar os dados armazenados neste servidor. A figura também ilustra o sistema embarcado, baseado na placa ESP8266, que se conecta à Internet através de um ponto de acesso sem fio (*Access Point*).



Com a placa conectada à Internet, utilize os blocos da guia lateral **Rede e Internet** >> **EasyMQTT** do **BIPES** para enviar dados para a nuvem, onde basta definir o “Tópico” (*Topic*) e a fonte dos dados ou valor (*Data*). O ID da sessão (*Session ID*) é criado automaticamente, mas você também pode definir a sessão manualmente, caso queira.



O **EasyMQTT**, do **BIPES**, oferece uma forma fácil de utilizar o protocolo MQTT, o padrão mais conhecido e utilizado para troca de dados de Internet das coisas³. O **BIPES** também permite que você faça programas que se comuniquem com qualquer outro dispositivo ou serviço MQTT, como ThingSpeak, ThingsBoard, ou mesmo a plataforma IoT da SAP! Utilize os blocos MQTT para estas integrações.

Ao executar o programa, você pode ver o resultado na aba **Console**:

```
Waiting for Wifi connection
Connected
0
EasyMQTT connected
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 1
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 7
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 2
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 8
EasyMQTT Publish - Session: u36rjx Topic: LDR Value: 1
```

Run block based program Stop running program


Vale lembrar que o **BIPES** tem um recurso muito útil: ver duas abas ao mesmo tempo, clicando com o botão direito do mouse em uma das abas e com o esquerdo em outra. Assim:

³ C. A. Silva. **Desenvolvimento e validação de módulo de comunicação MQTT para plataforma BIPES para aplicações de Internet das Coisas**. 2020. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) - Universidade Federal de São Carlos, 2020. Disponível em: <<https://repositorio.ufscar.br/handle/ufscar/13656>>.

The screenshot shows the BIPES IDE interface with a block-based program. The program starts with 'conectar na rede wifi' using 'nome da rede' and 'senha/chave' both set to 'Termometro'. It then 'inicia EasyMQTT' with 'sessão: U36rjx'. A loop 'repita enquanto verdadeiro' contains a 'faça' block with 'definir leitura para ler entrada analógica 0', 'publicar dado com EasyMQTT' (topic: 'LDR', value: 'leitura'), and 'esperar 1 segundos'.

Agora, você pode clicar na aba **EasyMQTT** e ver os resultados enviados para a nuvem:

The screenshot shows the BIPES IDE interface with the EasyMQTT dashboard. The dashboard displays 'Live data from luz' as a line graph with a zoom level of 'All'. Below the graph, there are input fields for 'Publish value: 0 to topic:' and 'Update frequency (s): 5'. The terminal output shows a stream of MQTT publish messages: 'EasyMQTT Publish - Session: 4truy9 Topic: luz Value: 2'.

Também é possível usar botão “**compartilhar**” () para enviar o link do seu programa para qualquer pessoa, e quando ela acessar este link, poderá visualizar, em tempo real, os dados enviados pela sua placa através da aba **EasyMQTT**.

Na aba **IOT**, vamos novamente criar um painel (*dashboard*) personalizado para visualizar estes dados (já fizemos isso para cabo USB, mas agora faremos usando a

Internet). Desta vez, vamos usar o **DATASOURCE EasyMQTT** e informar os dados deste programa:

DATASOURCE

TYPE: BIPES EasyMQTT

NAME: luz

BIPES EASYMQTT SESSION: 4tzuy9
The session code automatically given when you insert the Start EasyMQTT block on your BIPES program. You can list sessions here: <http://bipes.net.br/easymqtt/listsessions.php>

BIPES EASYMQTT TOPIC: luz
The topic you want to access for your EasyMQTT Session. You can list topics here: <http://bipes.net.br/easymqtt/getsession.php?session=XXX> (change XXX by your session)

REFRESH EVERY: 1 SECONDS

METHOD: GET
The URL for BIPES EasyMQTT will be assembled from session and topic you inform. For example: http://bipes.net.br/easymqtt/gettopic_last.php?session=chocadeira&topic=umidade

BODY:
The body of the request. Normally only used if method is POST

HEADERS: ADD

SAVE CANCEL

Em seguida, vamos adicionar os “*panes*”:

WIDGET

TYPE: Gauge

TITLE: Leitura LDR

VALUE: `datasources["luz"]["result"][0]["data"]` + DATASOURCE JS EDITOR

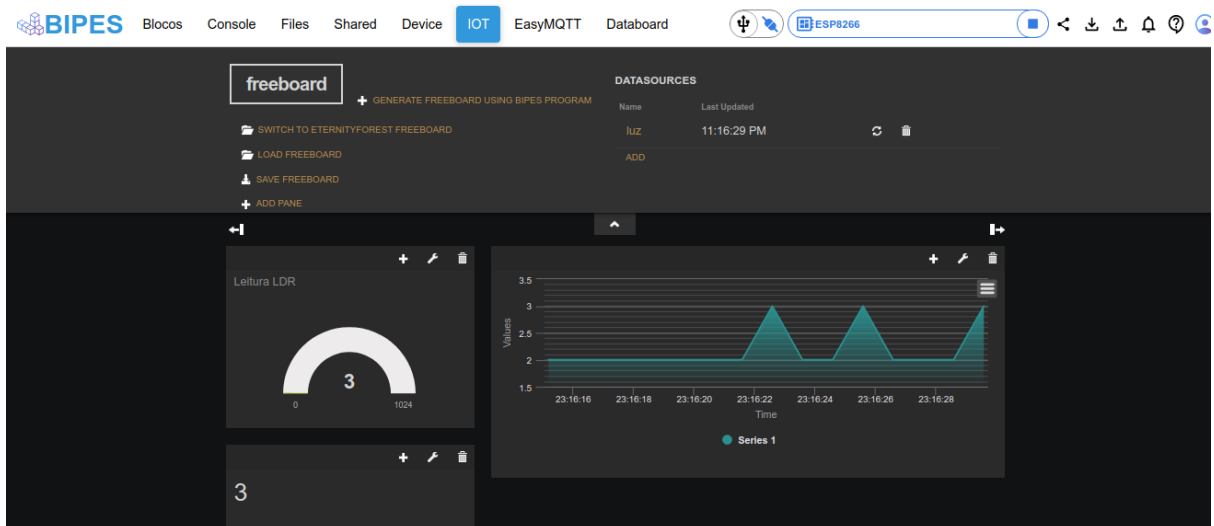
UNITS:


MINIMUM: 0

MAXIMUM: 1024

SAVE CANCEL

Você pode adicionar outros componentes e personalizar seu painel, incluindo gráficos, mapas, textos, e até códigos personalizados em Java Script. Por exemplo:



Novamente, note, que você pode compartilhar este programa com outra pessoa, pelo link, usando o botão . Esta pessoa terá acesso, a partir de qualquer lugar do mundo, aos dados sendo enviados pelo seu dispositivo. De fato, a opção de compartilhar, irá compartilhar o programa em blocos e o acesso ao *dashboard* IoT, bem como a aba **EasyMQTT**, que possui o histórico de cada tópico. Também é possível compartilhar apenas o painel de visualização (*dashboard*) através do botão “**Share Board**”, ao lado do ícone **freeboard**, que é o software livre integrado ao BIPES para visualização. Ao clicar em “**Share Board**”, um link será disponibilizado apenas para o acesso ao painel de visualização, bem como um QR-Code para acesso rápido a partir de *smartphones*.

Verificando erros

Diversas situações inesperadas podem ocorrer em um sistema embarcado, e, em muitos casos, não haverá um usuário para reinicializar o sistema. De fato, alguns sistemas embarcados podem estar em locais de difícil acesso.

Dessa forma, é importante ter formas de detectar e tratar erros. O bloco “**tente / no erro**” (*try - except*), disponível na guia lateral **Python** do **BIPES**, permite que o sistema tente executar uma operação, e caso a operação dê errado, execute um código alternativo. Isso é bastante útil em sistemas com conexão de rede, pois, por diversos motivos, a conexão pode não estar disponível. Além disso, em alguns casos, a falha de conexão de rede pode interromper a execução de todo programa até que um usuário / operador reinicie o sistema ou a execução deste programa.

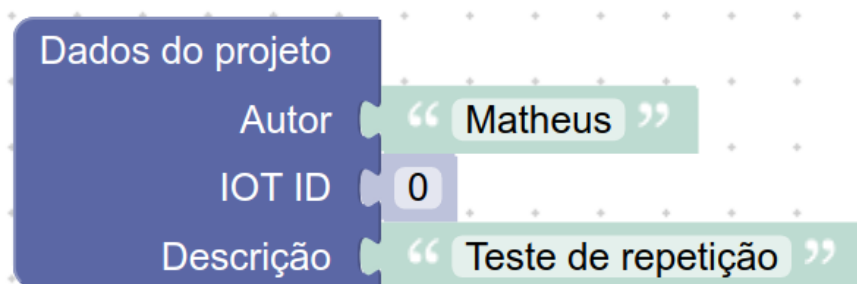
O programa a seguir ilustra um exemplo de sincronia de data e hora a partir da Internet usando o bloco NTP (disponível na guia lateral **Rede e Internet**). Caso ocorra uma falha na comunicação com o servidor NTP, o programa continua normalmente.




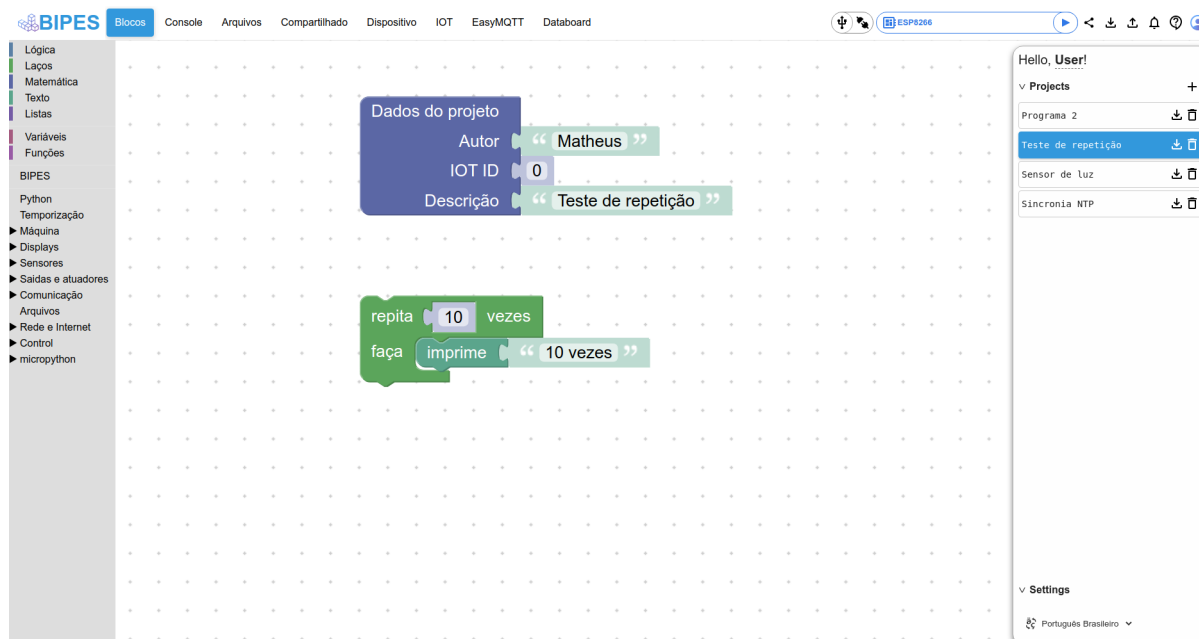
Sem a estrutura **Tente**, caso o programa falhe ao tentar conectar com o servidor NTP, o programa terá a execução abortada por falha de comunicação e nunca executará os blocos dentro da estrutura **repita**.

BIPES: Múltiplos projetos

Com a facilidade e praticidade oferecidas pelo **BIPES**, você não vai resistir a tentar vários programas e experimentos. Para facilitar este processo, note que sempre que você criar um novo programa no **BIPES**, automaticamente o seguinte bloco é disponibilizado na área de trabalho:



Com este bloco é possível definir o nome do programa e o seu autor. Note também, no canto superior direito da tela do **BIPES**, que um ícone de perfil pode ser utilizado: . Este ícone irá acessar a seguinte área do **BIPES**:



Nesta área é possível criar novos programas, navegar entre seus programas já criados, apagar, ou baixar para o PC o arquivo XML correspondente àquele programa. A mesma área também permite mudar o idioma da interface do **BIPES**.

Atividades em paralelo: temporizadores (*timers*)

Um outro recurso muito usado em sistemas embarcados são os temporizadores (*timers*). Os temporizadores são gerenciados pelo hardware da placa e, dessa forma, possibilitam uma contagem de tempo absoluta. São muito úteis para a implementação de procedimentos periódicos, que devem acontecer independentemente do fluxo de execução do programa principal. Nesse tipo de aplicação, o evento gerado pelo temporizador provoca uma interrupção no programa principal, desviando o fluxo de execução do programa para a rotina implementada no escopo do temporizador. Ao concluir a rotina associada ao temporizador, o sistema volta a executar o programa principal do ponto que foi interrompido.

O programa abaixo mostra um exemplo onde o **Timer 0** é configurado para imprimir no **Console**, a cada 1000 ms, a contagem de tempo, em milissegundos, desde que o ESP8266 foi ligado [**get milliseconds counter**] seguido da leitura analógica. O **Timer 1** também é configurado para desligar o **Timer 0** após 30 segundos. O programa também inclui um laço de repetição que pisca o LED no pino D4 enquanto o pino digital D0 tiver sinal lógico alto (1), mostrando também o instante de tempo da ocorrência deste evento.

```
Timer # 0 do every 1000 ms
  imprime criar texto com get milliseconds counter
  " : Leitura analógica = "
  ler entrada analógica 0

Timer # 1 do every 30000 ms
  imprime " Desligando Timer 0..."
  Stop Timer 0

repita enquanto verdadeiro
  faça
    se ler entrada digital pino D0 / GPIO16 = 1
      faça
        imprime criar texto com get milliseconds counter
        " : LED piscando "
        ajustar pino de saída pino D4 / LED / GPIO02 para verdadeiro
        esperar 1 segundos
        ajustar pino de saída pino D4 / LED / GPIO02 para falso
        esperar 1 segundos
```

A listagem abaixo mostra a saída do console após este programa ser executado por alguns segundos. Os textos entre os símbolos “<< >>” indicam observações nossas, que não foram impressas pelo programa no **Console**.

```
===  
<< PROGRAMA INICIADO COM PINO D0 EM NÍVEL LÓGICO 0>>  
3816: Leitura analógica = 1  
4816: Leitura analógica = 1  
5816: Leitura analógica = 1  
6816: Leitura analógica = 1  
7816: Leitura analógica = 1  
8816: Leitura analógica = 1  
9816: Leitura analógica = 1  
10816: Leitura analógica = 1  
11816: Leitura analógica = 1  
12816: Leitura analógica = 7  
13816: Leitura analógica = 7  
14816: Leitura analógica = 8  
<< PINO D0 RECEBE NÍVEL LÓGICO 1 (3V3)>>  
15445: LED piscando  
15816: Leitura analógica = 7  
16816: Leitura analógica = 7  
17453: LED piscando  
17816: Leitura analógica = 8  
18816: Leitura analógica = 8  
19461: LED piscando  
19816: Leitura analógica = 8  
20816: Leitura analógica = 8  
<< ALGUNS SEGUNDOS DE DADOS OMITIDOS >>  
31510: LED piscando  
31816: Leitura analógica = 1  
32816: Leitura analógica = 8  
Desligando Timer 0...  
<< APÓS 30 SEGUNDOS, TIMER 0 É DESLIGADO E LAÇO PRINCIPAL CONTINUA >>  
33518: LED piscando  
35526: LED piscando  
37534: LED piscando  
39543: LED piscando  
41551: LED piscando  
43559: LED piscando
```

Os números apresentados antes do “:” mostram o instante, em milissegundos de cada evento. Observe que os eventos do laço principal e dos dois temporizadores (*timers*) ocorrem de forma independente.

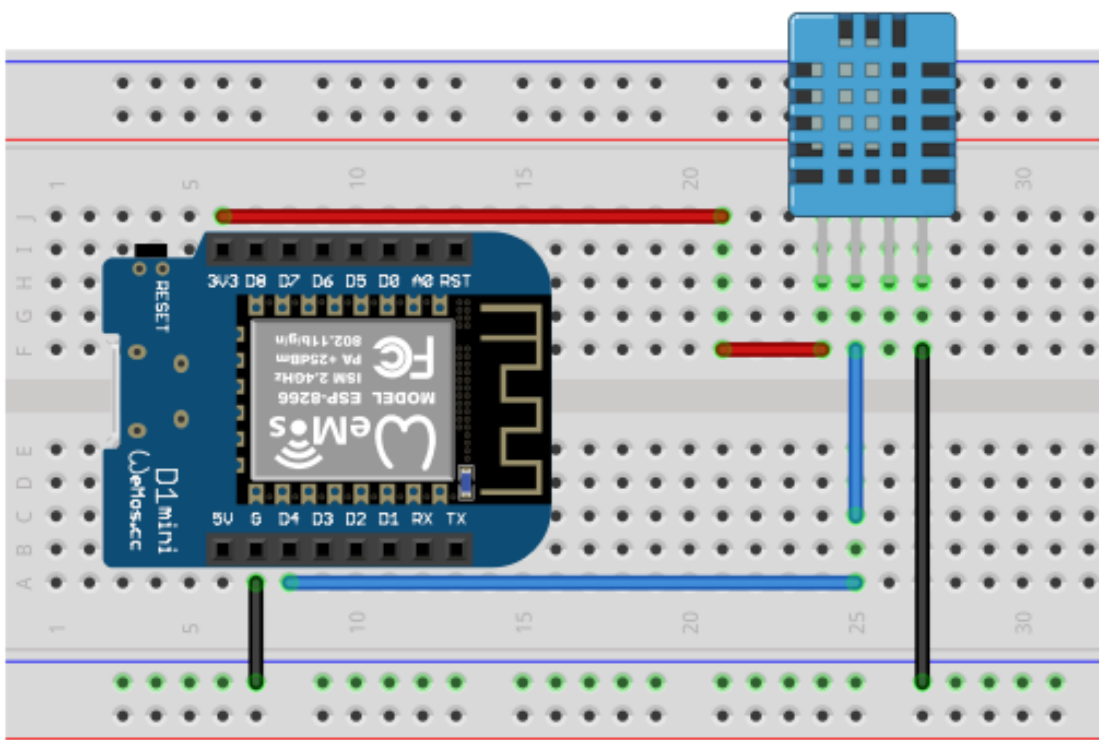
Sensor de temperatura e umidade

Também podemos incluir um sensor de temperatura e umidade no sistema, e enviar seus dados para a nuvem! Use *jumper cables*, diretamente, ou use a *protoboard* para ligar o sensor de umidade e temperatura DHT11 ao módulo ESP8266. Note que o sensor deve ser conectado ao pino 3V3 para receber alimentação, ao terra (G) e a saída (pino 2) ao pino de entrada D1 ou D4 (ou outro que você achar adequado) da placa.

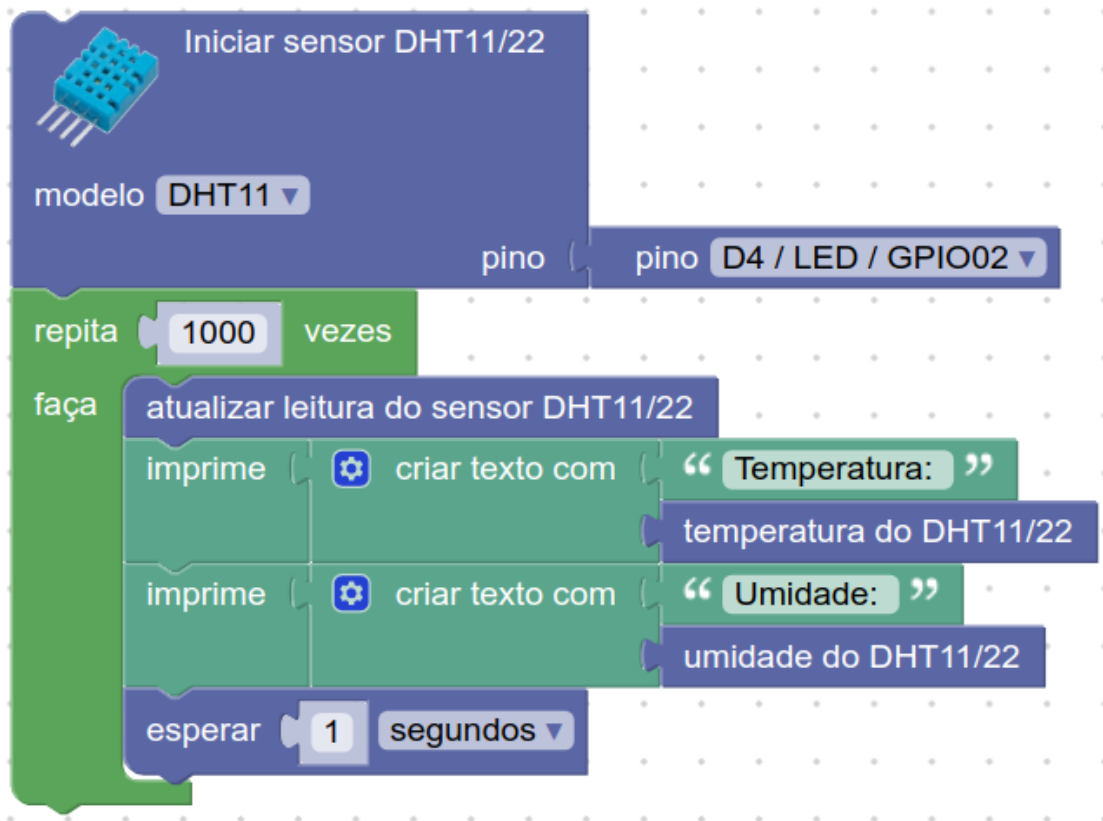
Sugestão de conexões:

<u>Pino do DHT11</u>	<u>Pino do ESP8266</u>
<u>GND (Pino 4)</u>	<u>G</u>
<u>VCC (Pino 1)</u>	<u>5V</u>
<u>DATA (Pino 2)</u>	<u>2 (D4)</u>

Sugestão de conexões (Placa WeMos D1 mini com ESP8266 conectada ao DHT11):



Programa:



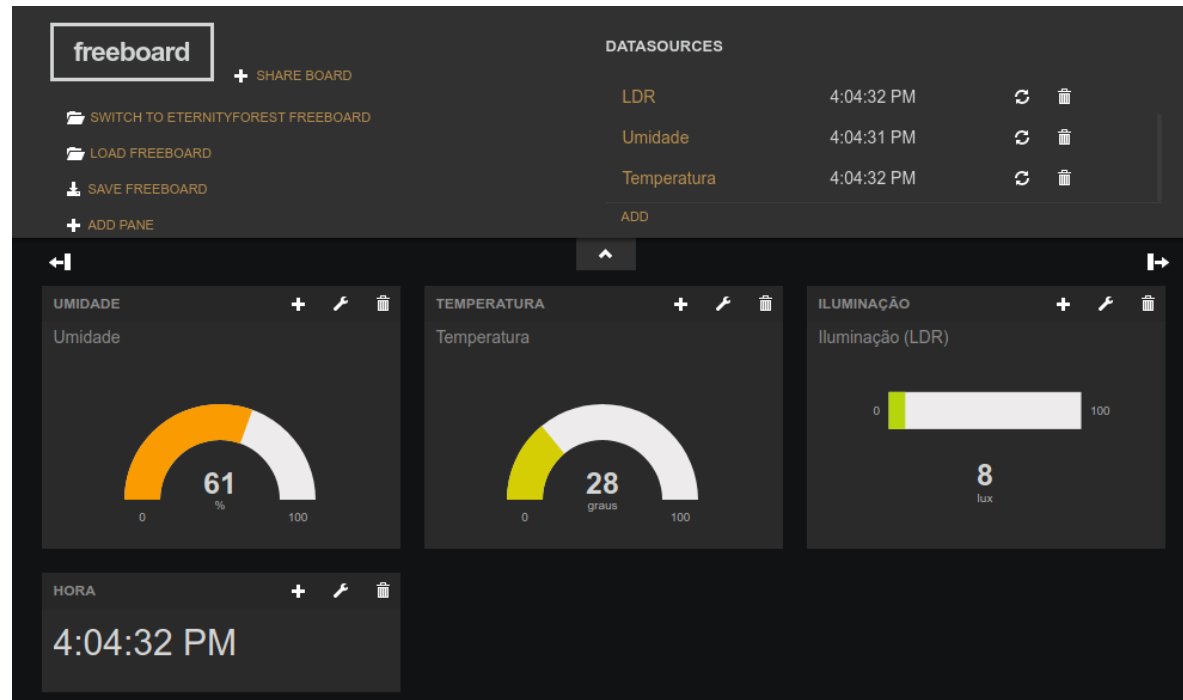
Execute o programa, e visualize o resultado na aba **Console**. Resultado:

```
Umidade: 61 %  
Temperatura: 28 graus Celsius  
Umidade: 61 %  
Temperatura: 28 graus Celsius  
Umidade: 62 %  
Temperatura: 28 graus Celsius  
Umidade: 62 %  
Temperatura: 28 graus Celsius  
Umidade: 62 %  
Temperatura: 28 graus Celsius  
Umidade: 62 %  
Temperatura: 28 graus Celsius  
Umidade: 62 %
```

Atividade:


Inclua o envio da umidade e da temperatura para a nuvem, juntamente com o LDR. Configure um painel IoT (*dashboard*) e verifique as leituras a partir de outro dispositivo / computador usando o datasource **BIPES EasyMQTT** na aba **IOT**

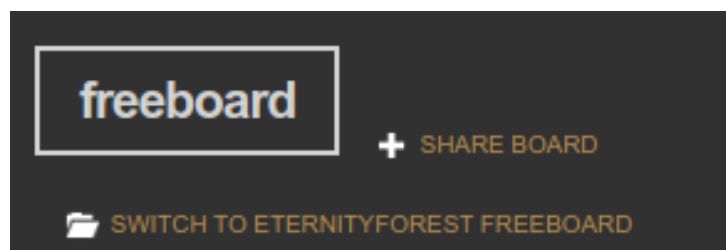
Exemplo de resultado da atividade:



Compartilhe o dashboard com celulares e outros dispositivos!

Após a aplicação estar pronta, é possível compartilhar o link apenas do painel de visualização (*dashboard*) com qualquer dispositivo, inclusive com otimização para celulares (tecnologia web responsiva).

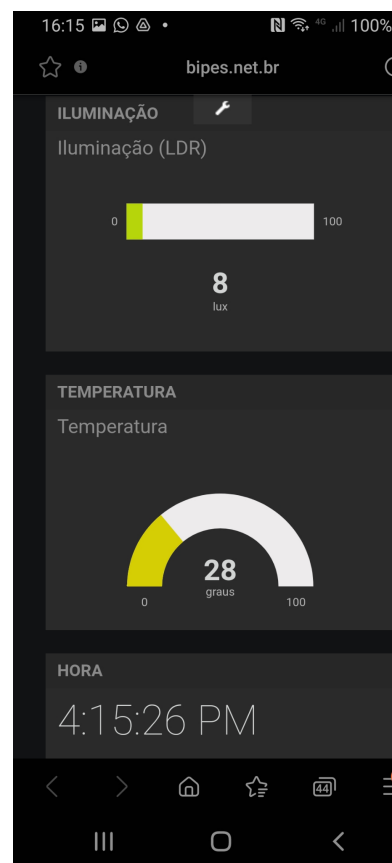
Ao clicar no botão  do BIPES, um link será gerado, similar a: <http://bipes.net.br/beta2/ui/#4c6bfm>. Substitua o termo *beta2/ui/* por *freeboard*, de forma que o novo link será: <http://bipes.net.br/freeboard#4c6bfm>. Este último link compartilhará apenas o *dashboard* (freeboard). Outra possibilidade, é utilizar a opção **Share Board**, ao lado do ícone do **freeboard**:



Ao clicar neste botão, um link será gerado com seu QR-Code associado, e ambos serão exibidos em uma nova janela:



Agora você pode compartilhar este link ou QR-Code com qualquer usuário que poderá visualizar os dados, e inclusive enviar comandos para seu sistema (se você incluir em seu painel uma aplicação interativa com botões (*switches*)). Ao acessar o QR-Code, ou link, pelo celular, o usuário também tem acesso à interface mostrada ao lado.

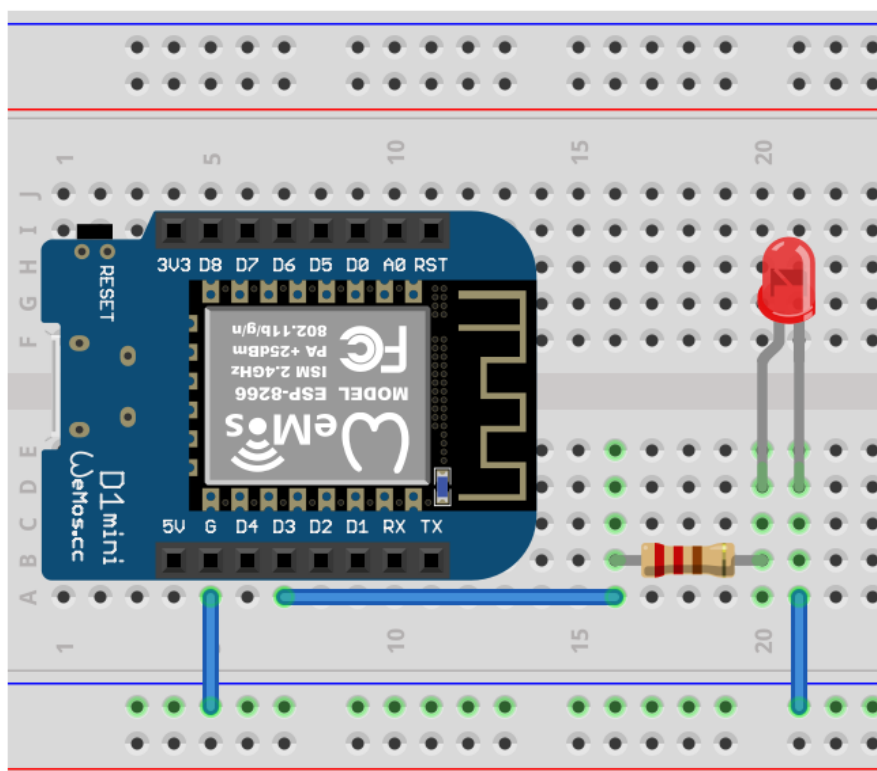


PWM: Controle de brilho do LED

Nós já discutimos como controlar uma saída digital e como verificar o estado de uma entrada digital e de uma entrada analógica. Uma outra possibilidade é usar uma saída digital do tipo PWM (**Pulse Width Modulation**), que permite, por meio de um sinal digital na forma de uma onda quadrada, aproximar um valor de tensão equivalente a uma saída analógica. Dessa forma, é possível usar um pino com saída PWM para controlar o brilho de um LED, a velocidade de um motor, a temperatura de uma resistência de aquecimento, dentre outras possibilidades. Note que nem todos os pinos suportam saída PWM, de forma que o LED já incluído na placa ESP8266 não está conectado a um pino PWM. Dessa forma, precisaremos incluir um LED externo, conectado a um pino com saída PWM, para conseguir visualizar esse efeito. No exemplo abaixo, utilizamos o pino D3 (GPIO0).

Montagem:

(Cuidado: Observe que o LED possui polaridade, ou seja, pinos positivo (terminal mais longo) e negativo (terminal mais curto, ao lado do chanfro no encapsulamento). O terminal negativo do LED deve ser conectado ao G (terra) e o terminal positivo ao resistor, que por sua vez é conectado ao pino D3)



O PWM possui dois parâmetros principais: a frequência do sinal da onda quadrada gerada (**Frequency**) e o ciclo de trabalho / *duty cycle* (**Duty**). Neste momento estamos interessados no **duty**. Esse parâmetro indica a porção da onda quadrada que estará em nível alto (ligado), podendo variar de 0 a 100%. Dessa forma, um duty cycle de 100%

significa potência máxima, 50% equivale a metade da potência e 0% a nenhuma potência. Na ESP8266 o PWM tem resolução de 10 bits (indo de 0 para 0% até 1023 para 100%).

O programa a seguir faz o LED ir aumentando o brilho aos poucos de 0 até o máximo e depois reduz seu brilho aos poucos até o 0, repetindo este processo 10 vezes. Observe como, nesta aplicação, o valor do parâmetro **duty** varia com a contagem da variável *i*.

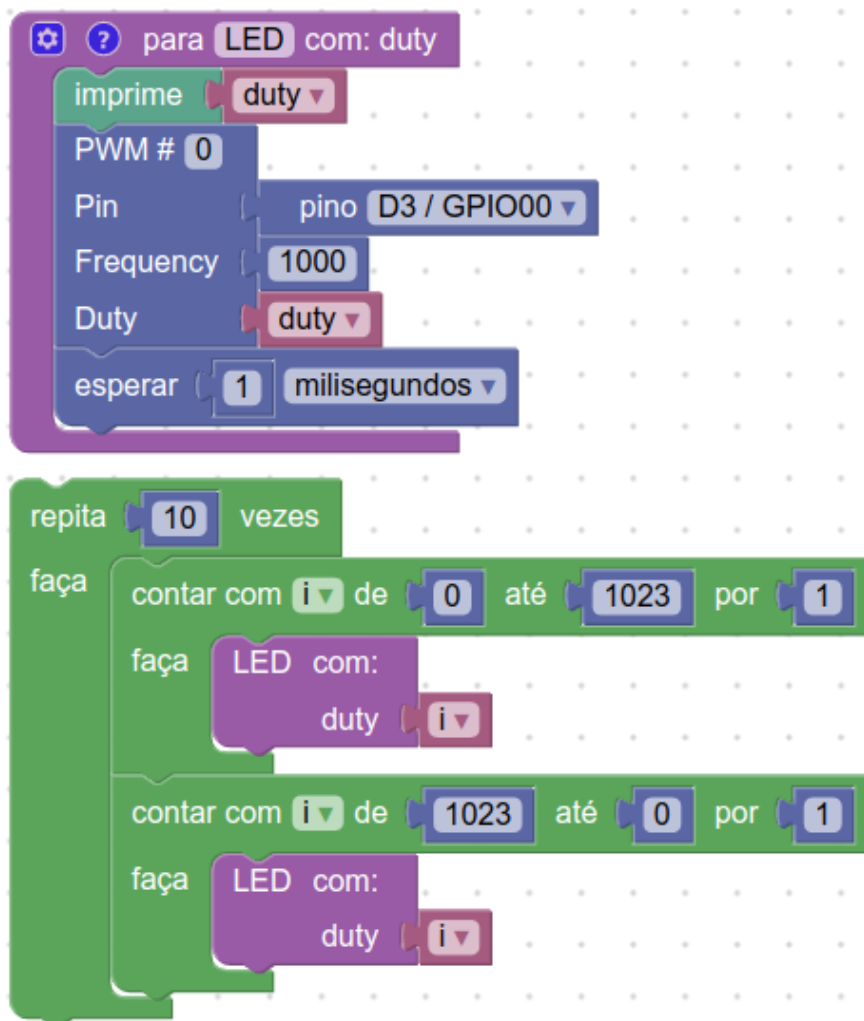


Atividade:

Utilize **EasyMQTT** e a aba **IOT** e implemente um **slider** (barra deslizante) na aba **IOT** que permita controlar o brilho do LED a partir de um celular ou outro dispositivo. Controle, então, o brilho do LED a partir do seu celular! Dica: use o bloco de inscrição em tópico EasyMQTT.

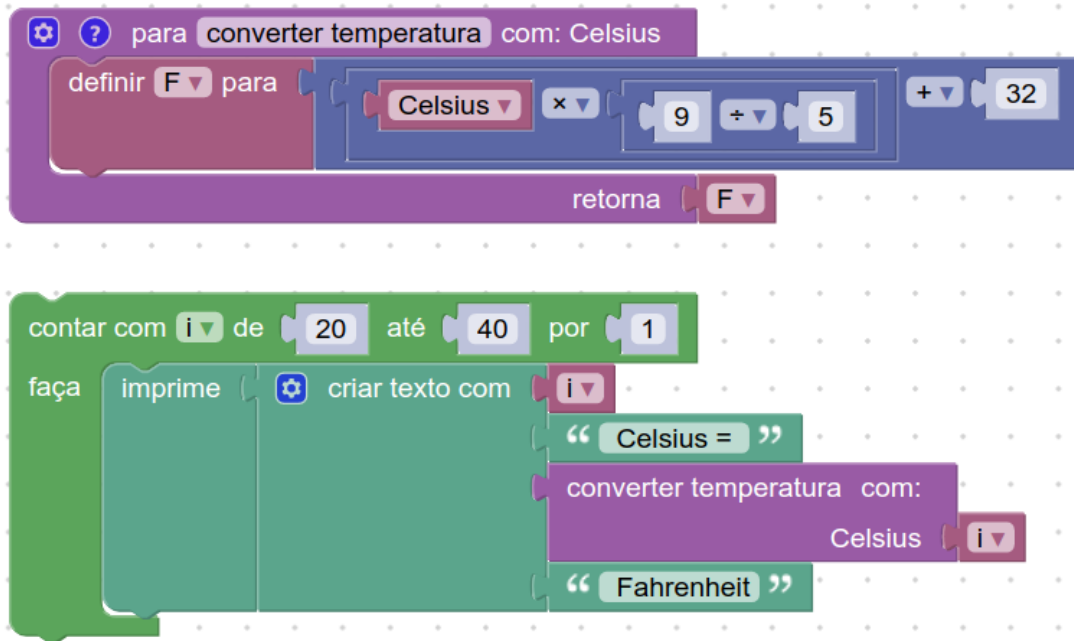
BIPES: Subrotinas / funções

Conforme o seu programa ganha novas funcionalidades, alguns trechos de código podem ficar repetidos, sendo útil criar blocos reutilizáveis: são as funções / sub-rotinas que podem ser criadas pelos blocos da guia lateral **Funções**. Se voltarmos ao exemplo anterior, do LED, note que há 3 comandos que se repetem 2 vezes (*print*, *PWM*, *delay*). Estes 3 comandos podem ser agrupados em uma função e reutilizados como no exemplo abaixo:




Note que no conjunto de blocos superior, criamos a **função LED**, que recebe um parâmetro **duty**, que, por sua vez, é usado internamente naqueles blocos. Já nos blocos de baixo, que constituem o programa principal, simplesmente chamamos a **função LED** com o parâmetro desejado (valor da variável *i*). A cada chamada, todos os comandos dentro da função são executados.

Você pode criar várias funções, com ou sem recebimento de parâmetros, e pode também criar funções que retornam valores. Veja, por exemplo, a seguinte função para conversão de unidades:



E o resultado no **Console** será:

```
===
20 Celsius = 68.0 Fahrenheit
21 Celsius = 69.8 Fahrenheit
22 Celsius = 71.6 Fahrenheit
23 Celsius = 73.4 Fahrenheit
24 Celsius = 75.2 Fahrenheit
25 Celsius = 77.0 Fahrenheit
26 Celsius = 78.8 Fahrenheit
27 Celsius = 80.6 Fahrenheit
28 Celsius = 82.4 Fahrenheit
29 Celsius = 84.2 Fahrenheit
30 Celsius = 86.0 Fahrenheit
31 Celsius = 87.8 Fahrenheit
32 Celsius = 89.6 Fahrenheit
33 Celsius = 91.4 Fahrenheit
34 Celsius = 93.2 Fahrenheit
35 Celsius = 95.0 Fahrenheit
36 Celsius = 96.8 Fahrenheit
37 Celsius = 98.6 Fahrenheit
38 Celsius = 100.4 Fahrenheit
39 Celsius = 102.2 Fahrenheit
40 Celsius = 104.0 Fahrenheit
>>>
```

Após criar as funções, você pode compartilhar seu programa com o botão  e usar as funções criadas em outros programas!

Controlando dispositivos via Internet

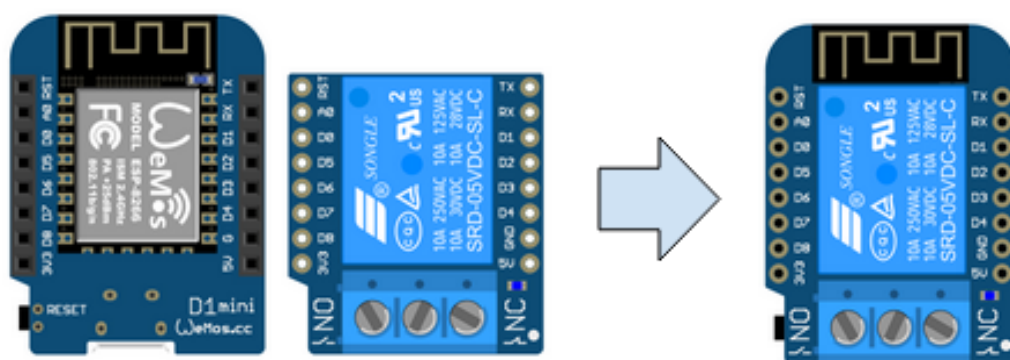
Nesta prática vamos utilizar um relé, tomadas e plugues para ligar / desligar qualquer dispositivo conectado à tomada (rede elétrica) via Web / Wi-fi. O dispositivo pode ser uma lâmpada, ventilador, bomba de água, aquecedor ou qualquer outro dispositivo com potência suportada pelo relé. Será necessária uma rede Wi-fi / roteador para conexão. Sugestão: pesquise sobre o funcionamento do relé e a história do primeiro *bug* de computador.

Cuidado:

Esta atividade inclui conexão com a rede elétrica com tensões de 127 Volts ou 220 Volts, apresentando risco de choque elétrico ou outros danos. Somente faça a montagem se você se sentir seguro, ou consulte um técnico especializado antes de ligar o sistema na rede elétrica.

De forma simples e rápida, o relé é um dispositivo eletromecânico que permite acionar cargas elétricas a partir de sinais digitais de um microcontrolador. Dessa forma, a placa ESP8266 aciona um módulo relé, que por sua vez aciona uma carga ligada a uma tomada elétrica. Para esta atividade é necessário conectar o relé à placa ESP8266 e a carga elétrica externa ao relé.

Uma possibilidade é utilizar um módulo relé do tipo *shield*, que encaixa diretamente sobre a placa ESP8266 (modelo WeMos D1 mini), não demandando cabos de conexão, como no exemplo abaixo, onde as duas placas mostradas podem ser empilhadas, encaixando-as através de seus terminais de conexão. A placa com o relé possui 3 terminais com parafusos, onde se pode conectar a carga a ser controlada. Notamos aqui que o relé opera como um interruptor controlado eletronicamente, sendo que ele possui 3 terminais de controle da carga: C (Comum); Normalmente Aberto (NO) e Normalmente Fechado (NC). Você deve utilizar 2 destes terminais em conjunto (C com NO) ou (C com NC). A escolha da combinação vai definir se o relé liga ou desliga a carga, quando é acionado pela ESP8266.



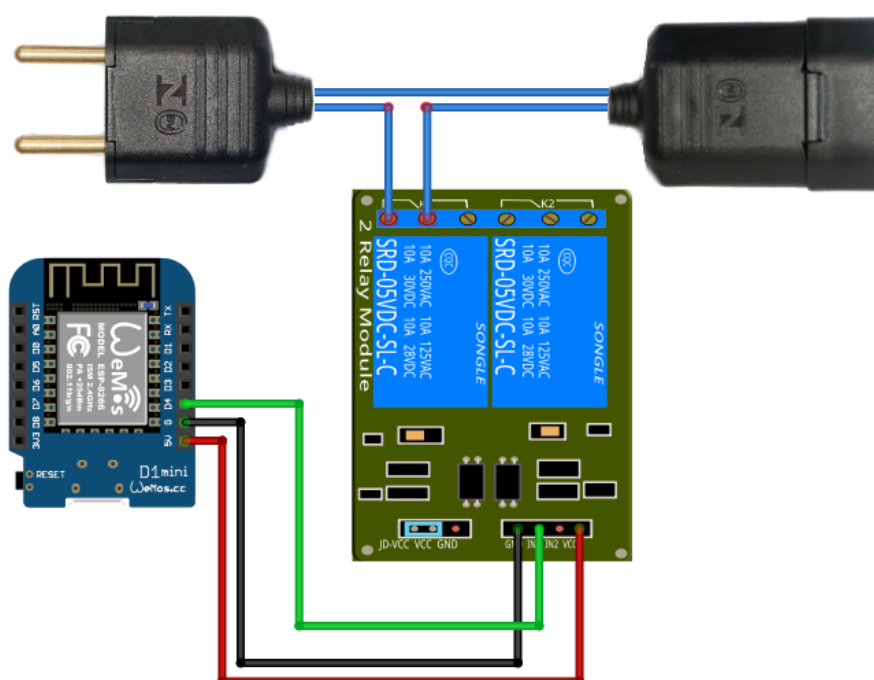
De outra forma, é possível utilizar um módulo relé simples ou duplo (permitindo controlar 2 dispositivos), que pode ser conectado por cabos pré-montados (*jumper wires*) à placa ESP.

Ligação do relé com o módulo ESP8266 - Conectar um módulo Relé ao ESP8266:

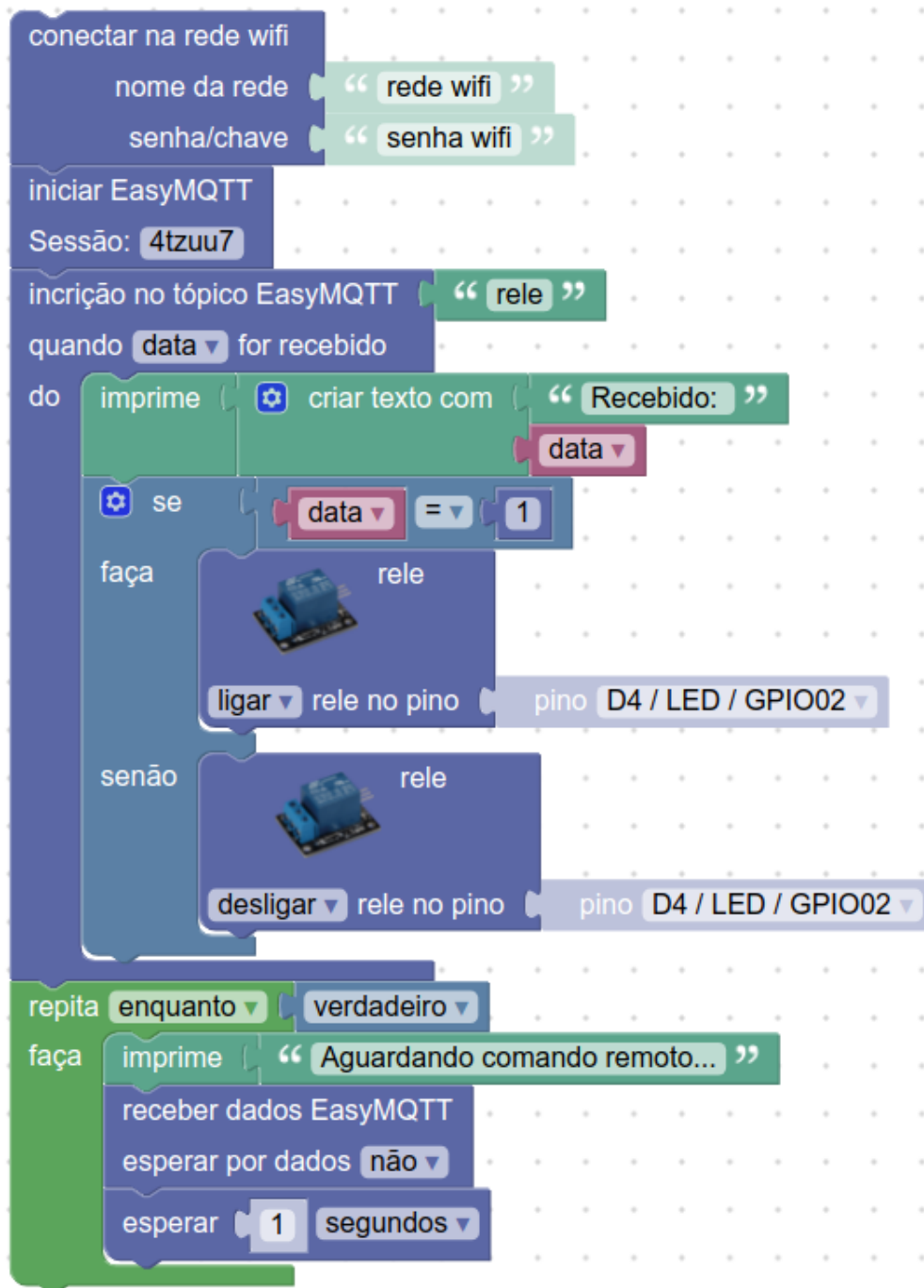
Pino do módulo rele	Pino da placa com módulo ESP8266
S ou IN1	pino D4 do ESP8266 (sinal de controle)
+ do relé	VIN (5 Volts) do ESP8266 (alimentação)
- do relé	GND do ESP8266 (terra)

Atenção: Todo o procedimento de montagem deve ser realizado com os dispositivos desconectados da rede elétrica.

O diagrama abaixo ilustra a montagem com plugue e tomada de dois fios. Um fio conecta diretamente o plugue e a tomada, enquanto a outra conexão passa pelo relé (normalmente aberto - pinos C e NO; ou normalmente fechado - pinos C e NC). Verifique as conexões com cuidado, e se necessário, peça ajuda para um técnico em eletrônica ou eletricidade. Após cuidadosa verificação e validação da montagem, conecte a carga a ser controlada no plugue (usaremos uma lâmpada nesta aplicação) e, por último, conecte a tomada na rede elétrica.



Lembrando que o objetivo desta prática é o acionamento da lâmpada via Internet. O programa em blocos para essa aplicação é o seguinte:



Para testar, utilize a aba **EasyMQTT**. Digite "rele" (sem acento, como está no programa), preencha o dado com 0 ou 1 e envie o dado. Verifique se a lâmpada vai mudar seu estado entre acesa / apagada ao seu comando. Note que o controle pode ser feito a partir de qualquer dispositivo conectado na Internet por meio de um endereço web. O **BIPES** também oferece uma API com requisições web (HTTP) possibilitando que outras aplicações possam ser integradas através do envio de requisições HTTP para o servidor do

BIPES, permitindo atualizar valores de tópicos em sessões EasyMQTT. A atualização é feita pela página **publish.php**, onde a sessão, tópico e valor desejados devem ser informados. Por exemplo, para a sessão **4tzuu7** e tópico rele, pode-se utilizar as seguintes requisições:

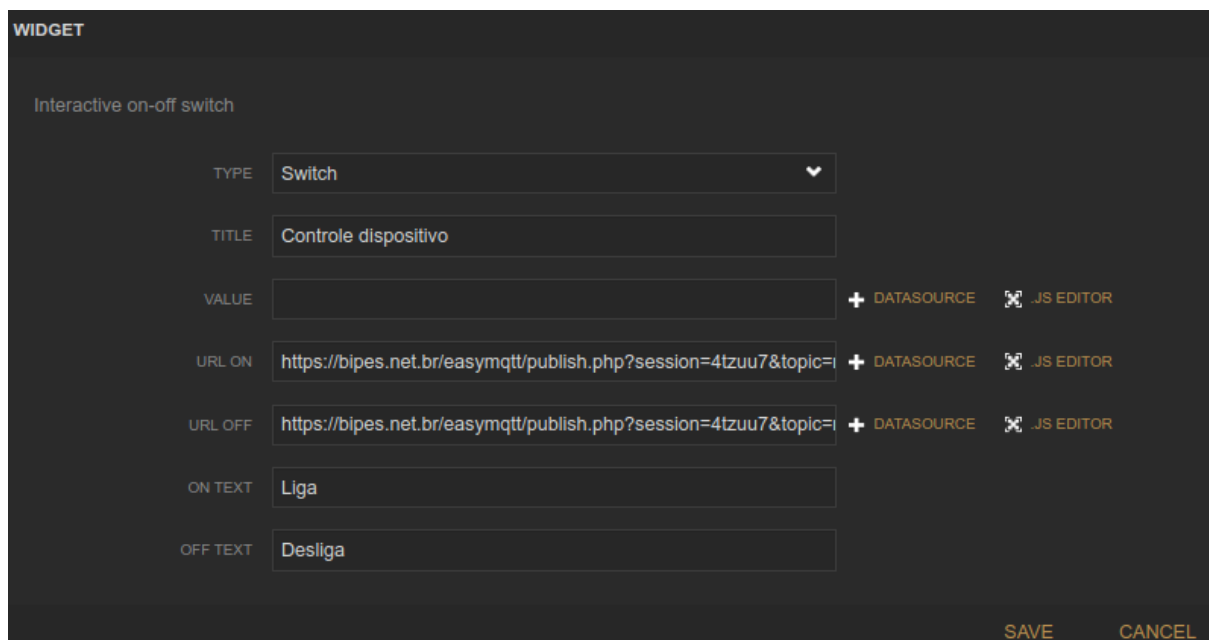
Para ligar o dispositivo:

<http://bipes.net.br/easymqtt/publish.php?session=4tzuu7&topic=rele&value=1>

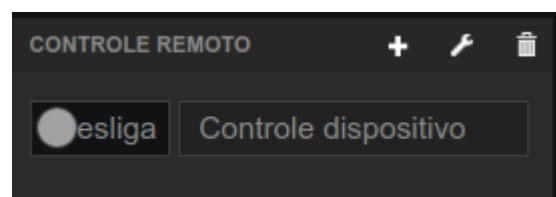
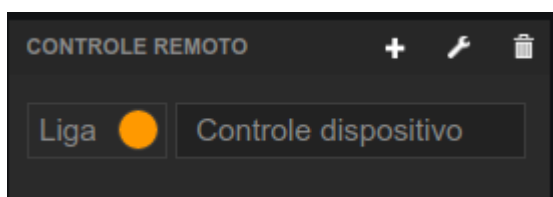
Para desligar o dispositivo:


<http://bipes.net.br/easymqtt/publish.php?session=4tzuu7&topic=rele&value=0>

Note que cada programa terá uma sessão diferente (*4tzuu7*, neste caso). Ajuste a sessão, ou crie sua própria! Você pode incluir estes endereços web em ações de botões na aba **IOT**. Além disso, verifique se seu acesso do BIPES foi via HTTP ou HTTPS. Os endereços de ligar/desligar citados acima devem iniciar por http ou https, dependendo da forma de acesso ao BIPES. Por exemplo:



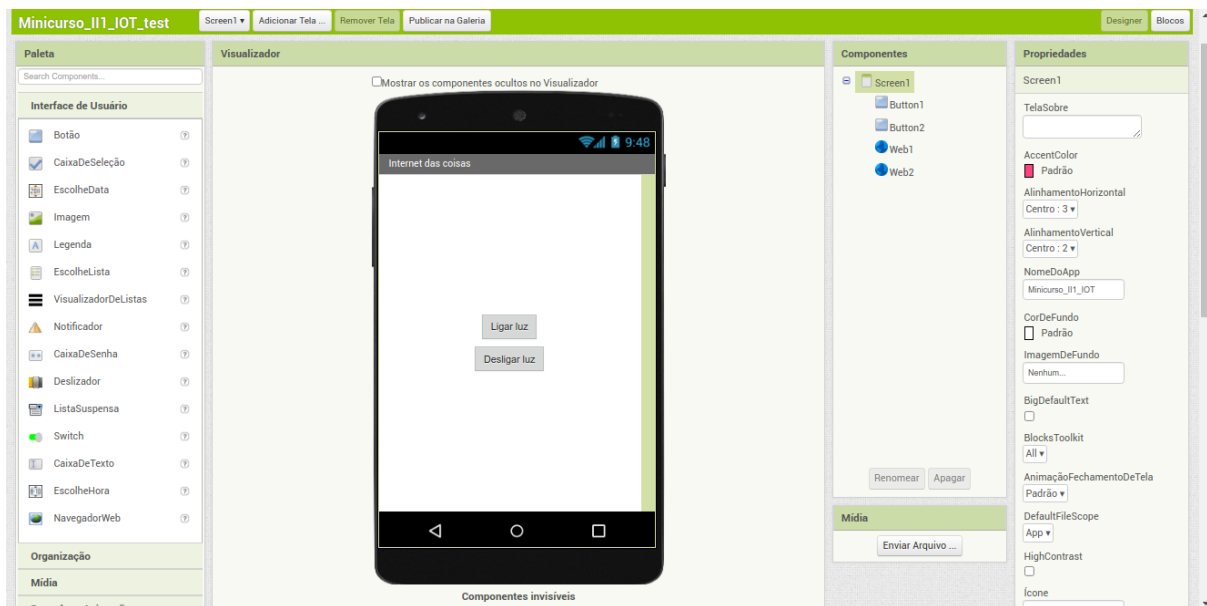
Agora, ao clicar nos botões da aba **IOT**, você controlará o dispositivo ligado ao relé, como uma lâmpada, por exemplo:



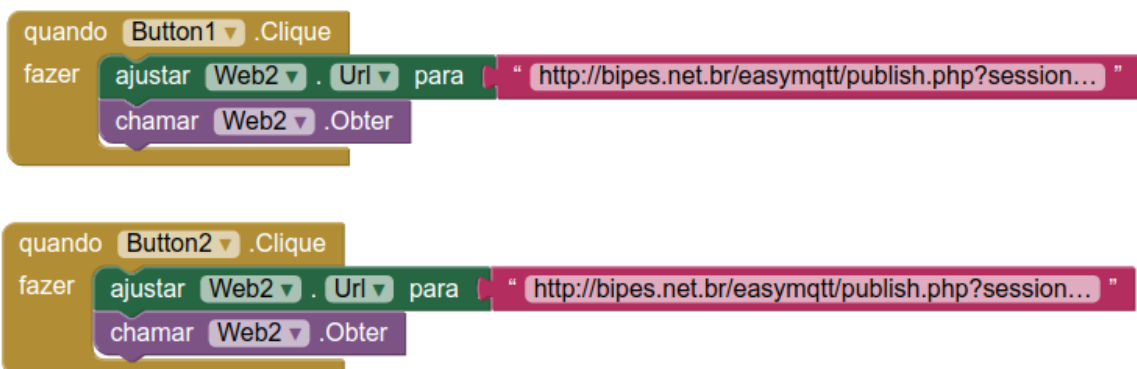
Note, que se você compartilhar este programa com outra pessoa, pelo link, usando o botão , esta pessoa terá acesso ao painel de controle, a partir de qualquer lugar do mundo, podendo controlar o dispositivo remotamente.

Para os curiosos: Você também pode usar o **MIT App Inventor** (<http://ai2.appinventor.mit.edu/>) para criar um aplicativo para celular que controla o dispositivo. Isso é bem simples! Veja o exemplo completo abaixo:

No modo **"Designer"**, crie um programa novo e inclua 2 botões. Configure as propriedades visuais de cada botão (texto, cor, posição, etc) na aba "Propriedades" à direita da tela.



No modo **"Blocos"** (canto superior direito), implemente dois conjuntos de blocos que definem as ações dos botões, cada um aplicando um dos links já mencionados:



Pronto! Os blocos acima constroem um aplicativo completo para celulares Android, que possibilita ligar/desligar dispositivos remotamente. Note como ele é similar ao **BIPES** e fácil de usar, além de poder também ser facilmente integrado ao BIPES!

Atividade:

Lembre do exemplo mencionado na introdução deste livro: o controle “on/off” de uma geladeira. Você já tem o conhecimento e as ferramentas para implementar este tipo de controle!

Utilize a placa ESP8266 ou ESP32 em conjunto com a montagem com o relé, feita nesta última prática, em conjunto com um sensor de temperatura e umidade DHT11. Prepare um programa que liga um relé, responsável por acionar um ventilador toda vez que a temperatura for superior a 30 graus Celsius e desliga o relé quando a temperatura estiver abaixo de 25 graus Celsius.

Após testar o sistema, inclua a funcionalidade de monitorar a temperatura remotamente, e inclua também a opção de ajustar, via *dashboard*, a temperatura mínima e máxima que mantém o relé, que controla o ventilador, acionado.

Cliente web e servidor web (HTTP)

Uma atividade muito comum na atualidade é acessar sites web, como, por exemplo, <http://www.bipes.net.br/>. Para este tipo de acesso, um navegador web, como o Google Chrome, por exemplo, utiliza o protocolo de transferência de hipertexto - *Hyper Text Transfer Protocol (HTTP)*. A troca de informações por HTTP é iniciada por um dispositivo **cliente** que inicia o processo através de uma requisição enviada ao **servidor**. A seguinte figura ilustra uma possível situação, onde uma placa ESP8266 atua como cliente, realizando uma requisição para um servidor, via Internet. Note, que tal cenário também poderia ser invertido, onde a ESP8266 poderia atuar como servidor.



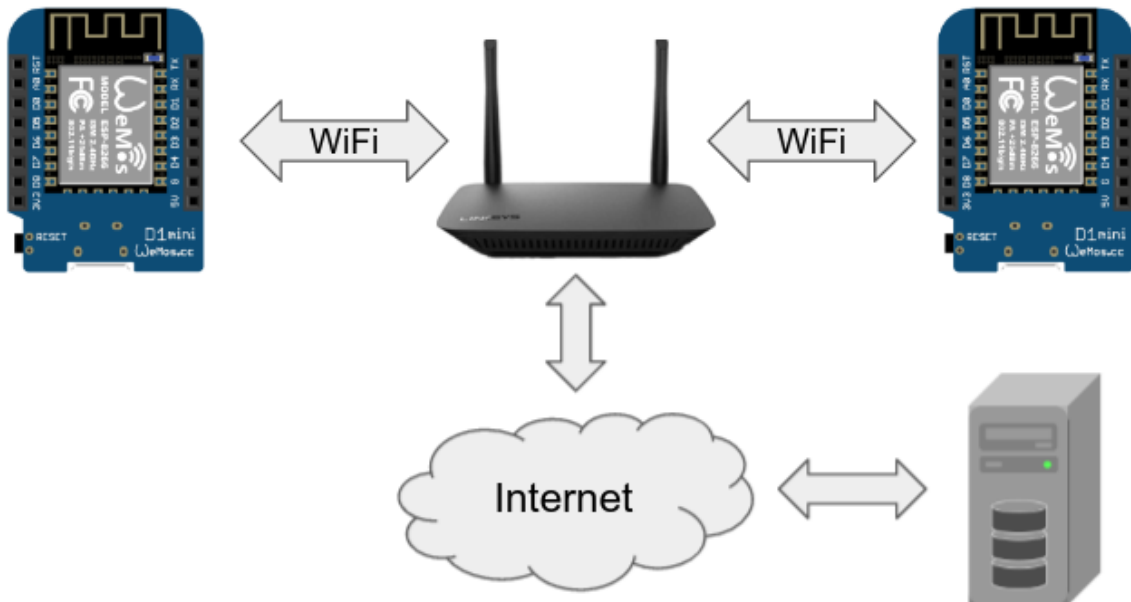
Outras arquiteturas (opções de conexão) também são possíveis, inclusive em cenários sem conexão com a Internet. A figura a seguir mostra a possibilidade de duas placas ESP8266 trocando dados diretamente entre si. Nesta situação, uma placa deve ser configurada em modo ponto de acesso, disponibilizando uma rede WiFi para conexão, e outra como estação WiFi. Além disso, uma placa atua como servidor HTTP e outra como cliente HTTP. Este tipo de conexão/rede sem fio é chamada *ad-hoc*.



Uma arquitetura similar pode ser obtida utilizando-se um ponto de acesso sem fio, que disponibiliza uma rede WiFi, através da qual as duas placas se conectam. Do ponto de vista da troca de mensagens HTTP, o funcionamento é o mesmo já descrito. Esta arquitetura de conexão é chamada de modo infraestrutura, já que o ponto de acesso sem fio é considerado um equipamento de infra-estrutura.



Em algumas configurações de rede, pode ser que as duas placas não possam se comunicar diretamente. Assim, também é possível que dois sistemas embarcados se comuniquem por intermédio de um servidor. Cada uma das placas ilustradas na figura abaixo, realiza requisições para um servidor. As placas podem publicar ou consumir dados deste servidor, de forma que uma placa pode deixar mensagens que serão disponibilizadas para outras aplicações ou dispositivos.



Para realizar a requisição HTTP, o cliente precisa conhecer o localizador uniforme de recurso - *Uniform Resource Locator (URL)*, que, resumidamente, possui a seguinte estrutura:

```
http://endereço:porta/caminho/recurso?query_string
```

Um exemplo real de URL já foi utilizado neste livro, na atividade de acionar um dispositivo via Internet:

```
http://bipes.net.br/easymqtt/publish.php?session=4tzuu7&topic=rele&value=1
```

Detalhando cada parte:

Parte	Descrição
http://	Protocolo: HTTP ou HTTPS (seguro, com transmissão criptografada) No exemplo: http://
endereço	Endereço do servidor na Internet, podendo, por exemplo, estar no formato de endereço IP ou nome, como 34.95.149.23 ou bipes.net.br No exemplo: bipes.net.br
:porta	A porta é um parâmetro opcional, que tipicamente é omitida. Quando omitida, subentende-se que a porta é 80, a porta padrão para serviços web. O uso de outros valores de porta é útil quando há mais de um servidor web associado a um mesmo IP / endereço, ou quando algum <i>firewall</i> ou sistema de segurança bloqueia a porta 80. No exemplo, a porta foi omitida.
/caminho/	O caminho do recurso no servidor. No exemplo: /easymqtt/
recurso	O nome do recurso a ser utilizado/acessado naquele servidor. No exemplo: publish.php
?query_string	A query_string permite enviar parâmetros para o recurso. Vários parâmetros podem ser incluídos na query_string separados pelo símbolo & . No exemplo: ?session=4tzuu7&topic=rele&value=1

Cliente HTTP

O acesso a serviços web via HTTP não se resume apenas a conteúdo. De fato, o protocolo HTTP se tornou uma forma padrão de acessar milhares de serviços, enviar e receber dados entre dispositivos via HTTP. Aqui, um conceito interessante é o M2M ou *Machine to Machine*, que significa a comunicação direta entre máquinas. No contexto de serviços web, a URL de acesso a este serviço também é chamada de **endpoint**.

Conhecendo a URL do serviço web que será utilizado, um cliente web pode realizar requisições HTTP para acessar conteúdo e executar ações associadas ao acesso. Os clientes HTTP mais comuns, certamente são os navegadores web. Assim, é possível usar o Google Chrome ou Firefox para acessar URL de serviços web que serão discutidos aqui.

Neste sentido, existem serviços gratuitos e pagos, que podem ser utilizados via HTTP, disponibilizados em um formato chamado de *Application Programming Interface* (API). Assim, existem milhares de opções de serviço acessíveis por clientes HTTP, como por exemplo, consultar a previsão do tempo, realizar operações financeiras, obter cotações de moedas e ações, realizar pagamentos, enviar mensagens via SMS, postar mensagens via Twitter, enviar mensagens via Telegram, realizar consultas de geolocalização, acionar dispositivos, acessar dados de sensores, dentre outras. O site web <https://any-api.com/> lista mais de 1400 APIs para possível integração em outras aplicações.

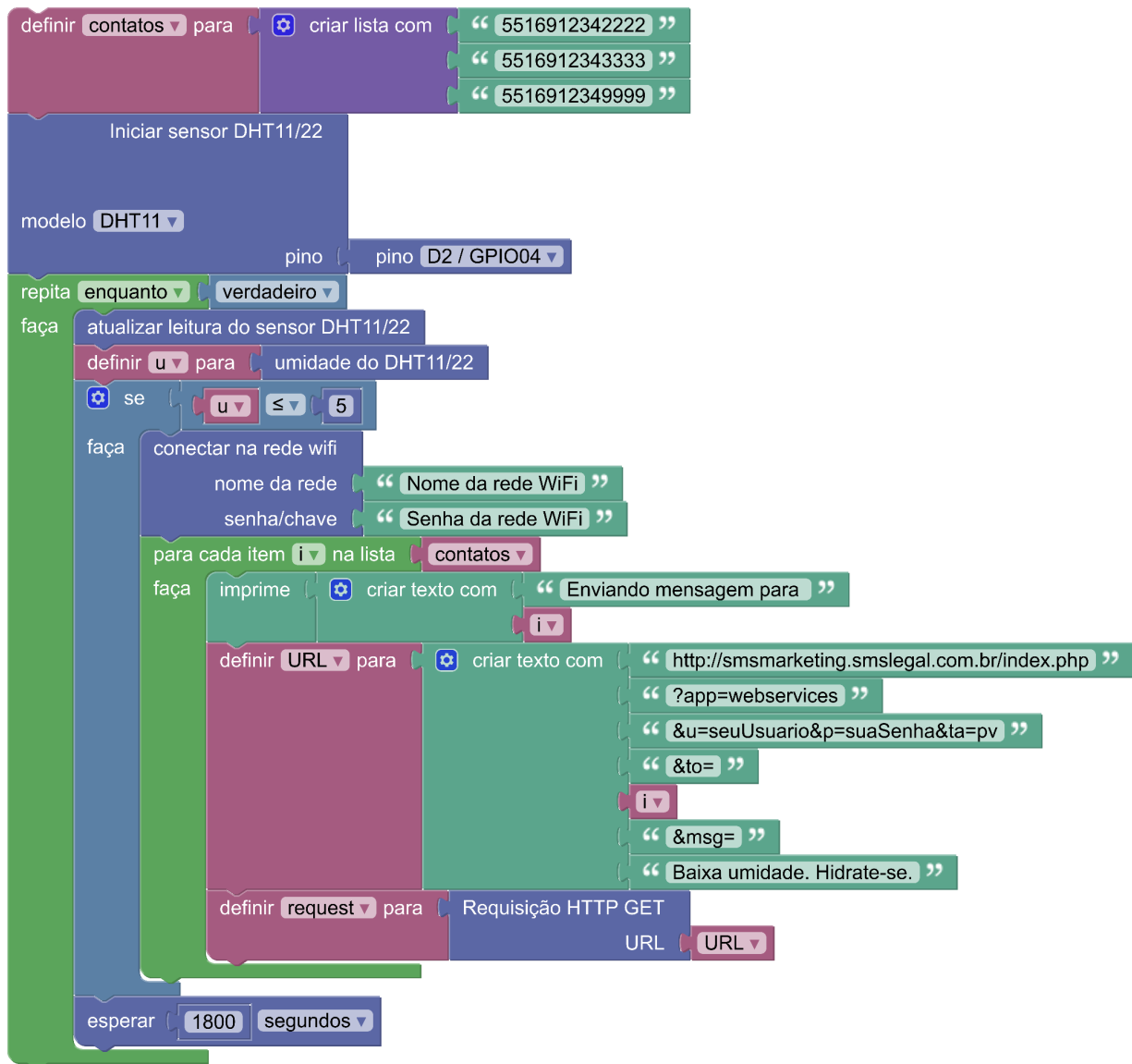
Cliente HTTP: Envio de mensagens SMS

A próxima figura mostra um exemplo de programa que envia mensagens SMS para uma lista de telefones, alertando sobre baixa umidade quando um sensor DHT11 quando a medida da umidade relativa do ar menor que 5%. Note que para enviar as mensagens SMS, o sistema proposto não possui nenhum componente eletrônico adicional de telefonia celular ou SIM Card. O envio da mensagem SMS ocorre através de um serviço web disponibilizado pelo seguinte recurso HTTP (**endpoint**): <http://smsmarketing.smslegal.com.br/index.php>. Ao acessar este recurso, é possível enviar parâmetros (como telefone de destino e mensagem) pela própria query_string da URL. Por exemplo:

<http://smsmarketing.smslegal.com.br/index.php?app=webservices&u=seuUsuario&p=suaSenha&ta=pv&to=5516997970000&msg=BIPES>

A URL acima envia uma mensagem com conteúdo “BIPES” para o telefone celular de número 5516997970000. Este serviço web é pago, de forma que tanto este, quanto outros serviços web, necessitam de autenticação. Por este motivo, os parâmetros **u** e **p** devem ser usados para informar usuário e senha.

O programa completo é mostrado na próxima página.



Cliente HTTP: Galinho do tempo IoT

Talvez você se lembre daqueles galinhos do tempo que mudam de cor conforme as condições climáticas, variando entre cor de rosa e azul. Podemos construir uma versão IoT deste galinho, utilizando uma placa ESP e um LED RGB de três cores - vermelho, verde e azul (*Red*, *Green* e *Blue*). A fonte das informações climáticas é o Instituto Nacional de Pesquisas Espaciais do Ministério da Ciência, Tecnologia e Inovações (INPE/MCTI), que oferece um serviço web de consulta às informações climáticas. Além disso, é possível conectar a placa ESP a um *power bank*, possibilitando até mesmo transformar este galinho IoT em um item de decoração funcional.

Considerado uma referência em pesquisas espaciais, o INPE/MCTI é a principal fonte de informações climáticas e previsão do tempo no Brasil. O Centro de Previsão de Tempo e Estudos Climáticos (CPTEC) do INPE/MCTI oferece um serviço web gratuito de

consulta às informações climáticas e previsão do tempo. Maiores informações sobre o serviço e seus termos de uso estão disponíveis em: <http://servicos.cptec.inpe.br/XML/>.

O primeiro passo para utilizar este serviço, é verificar o código (ID) da cidade para a qual se deseja obter informações. Para tanto, o serviço web <http://servicos.cptec.inpe.br/XML/listaCidades> oferece uma lista de cidades cobertas pelo serviço e seus respectivos códigos. Note que esta lista é fornecida no formato XML (*eXtensible Markup Language*), um padrão internacional para troca de informações entre máquinas e serviços web.

Para este exemplo, utilizaremos o exemplo da cidade de São Carlos - SP, um pólo Brasileiro de ciência e tecnologia⁴. O código para a cidade de São Carlos - SP é 4774. Dessa forma, é possível utilizar o recurso `previsao.xml`: <http://servicos.cptec.inpe.br/XML/cidade/4774/previsao.xml>. O acesso a este **endpoint** gera o seguinte resultado em XML:

```
<cidade>
  <nome>São Carlos</nome>
  <uf>SP</uf>
  <atualizacao>2021-12-22</atualizacao>
  <previsao>
    <dia>2021-12-22</dia>
    <tempo>ci</tempo>
    <maxima>29</maxima>
    <minima>19</minima>
    <iuv>13.0</iuv>
  </previsao>
  <previsao>
    <dia>2021-12-23</dia>
    <tempo>ci</tempo>
    <maxima>27</maxima>
    <minima>19</minima>
    <iuv>13.0</iuv>
  </previsao>
  <previsao>
    <dia>2021-12-24</dia>
    <tempo>c</tempo>
    <maxima>26</maxima>
    <minima>18</minima>
    <iuv>14.0</iuv>
  </previsao>
  <previsao>
    <dia>2021-12-25</dia>
    <tempo>ci</tempo>
    <maxima>24</maxima>
    <minima>18</minima>
    <iuv>14.0</iuv>
  </previsao>
</cidade>
```

Fonte: CPTEC/INPE (Consulta em Dez/2021)

⁴ Saiba mais sobre São Carlos - SP em <https://www.reportsancahub.com.br/>

Programa:

```
repita enquanto verdadeiro
  faça
    conectar na rede wifi
      nome da rede "Termometro"
      senha/chave "Termometro"
    definir URL para criar texto com "http://servicos.cptec.inpe.br/XML"
      "/cidade/4774/previsao.xml"
    definir request para Requisição HTTP GET
      Make HTTP GET Request URL URL
    se Status da resposta HTTP request = 200
      faça
        imprime criar texto com "Sucesso. Conteudo da resposta ="
          Conteúdo da resposta HTTP request
        definir L1 para Fazer uma lista a partir do texto Conteúdo da resposta HTTP request com delimitador "<tempo>"
        definir L2 para na lista L1 obter nº 1
        definir L3 para Fazer uma lista a partir do texto L2 com delimitador "<tempo>"
        definir tempo para na lista L3 obter nº 0
        imprime criar texto com "L1="
          L1
        imprime criar texto com "L2="
          L2
        imprime criar texto com "L3="
          L3
        imprime criar texto com "Tempo ="
          tempo
        ajustar pino de saída pino D1 / GPIO05 para falso
        ajustar pino de saída pino D2 / GPIO04 para falso
        ajustar pino de saída pino D3 / GPIO00 para falso
        se tempo = "c"
          faça
            imprime "Chuva"
            ajustar pino de saída pino D3 / GPIO00 para verdadeiro
        se tempo = "cl"
          faça
            imprime "Céu claro"
            ajustar pino de saída pino D2 / GPIO04 para verdadeiro
        se tempo = "pc"
          faça
            imprime "Parcialmente chuvoso"
            ajustar pino de saída pino D3 / GPIO00 para verdadeiro
            ajustar pino de saída pino D1 / GPIO05 para verdadeiro
        se tempo = "t"
          faça
            imprime "Tempestade"
            ajustar pino de saída pino D1 / GPIO05 para verdadeiro
        senão
          imprime criar texto com "Falha. Erro ="
            Status da resposta HTTP request
        imprime criar texto com "TS="
          get seconds counter
      esperar 600 segundos
```

O programa apresentado possui um laço de repetição infinito com espera de 10 minutos. Assim, o sistema consulta o servidor do INPE/MCTI a cada 10 minutos e atualiza as cores do LED conforme condições climáticas daquele momento. Logo após realizar a requisição **HTTP GET** para o servidor, o programa verifica se o servidor respondeu com o código 200, que significa que a requisição foi recebida, processada com sucesso e a resposta está disponível. Neste caso, a resposta é um texto no formato XML, conforme já ilustrado. O programa, então, realiza uma sequência de operações nas variáveis L1, L2, L3 e tempo, para obter apenas a sigla com informações sobre o tempo. Na sequência, são realizados testes condicionais **SE** para verificar algumas condições e acender os LEDs relacionados: vermelho (pino D1) para tempestade, verde (pino D2) para céu claro, azul (pino D3) para chuva. Também foi implementada uma possibilidade para a sigla pc (parcialmente chuvosa), que acende os LEDs conectados aos pinos D1 e D3 simultaneamente (vermelho e azul), resultando em uma cor similar ao roxo.

A listagem abaixo mostra um exemplo de uma saída do **Console** ao executar o programa. Note que a impressão dos valores das variáveis L1, L2, L3 e tempo não seriam necessárias na versão de produção do dispositivo. Colocamos estas impressões de dados no programa para mostrar cada passo da obtenção da informação desejada por meio dos blocos de manipulação de listas. Alguns trechos foram destacados em negrito para facilitar a compreensão das mensagens abaixo.

```
Waiting for Wifi connection
Connected
```

```
Sucesso. Conteudo da resposta = b"<?xml version='1.0'
encoding='ISO-8859-1'?><cidade><nome>São
Carlos</nome><uf>SP</uf><atualizacao>2021-12-22</atualizacao><previsao><dia>2021
-12-22</dia><tempo>ci</tempo><maxima>29</maxima><minima>19</minima><iuv>13.
0</iuv></previsao><previsao><dia>2021-12-23</dia><tempo>ci</tempo><maxima>27</
maxima><minima>19</minima><iuv>13.0</iuv></previsao><previsao><dia>2021-12-24</
dia><tempo>c</tempo><maxima>26</maxima><minima>18</minima><iuv>14.0</iuv></p
revisao><previsao><dia>2021-12-25</dia><tempo>ci</tempo><maxima>24</maxima><
minima>18</minima><iuv>14.0</iuv></previsao></cidade>"
```

```
L1=[b"<?xml version='1.0' encoding='ISO-8859-1'?><cidade><nome>São
Carlos</nome><uf>SP</uf><atualizacao>2021-12-22</atualizacao><previsao><dia>2021
-12-22</dia>',
'ci</tempo><maxima>29</maxima><minima>19</minima><iuv>13.0</iuv></previsao><p
revisao><dia>2021-12-23</dia>',
'ci</tempo><maxima>27</maxima><minima>19</minima><iuv>13.0</iuv></previsao><pr
evisao><dia>2021-12-24</dia>',
'c</tempo><maxima>26</maxima><minima>18</minima><iuv>14.0</iuv></previsao><pr
evisao><dia>2021-12-24</dia>',
'ci</tempo><maxima>25</maxima><minima>18</minima><iuv>14.0</iuv></previsao></ci
dade>"]
```

```
L2=ci</tempo><maxima>29</maxima><minima>19</minima><iuv>13.0</iuv></previsao
```

```
><previsao><dia>2021-12-23</dia>
```

```
L3=['ci',
```

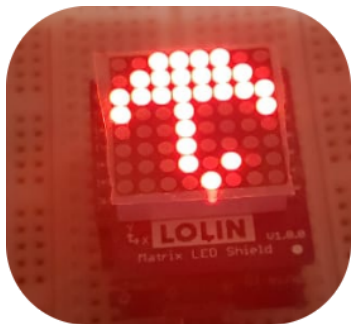
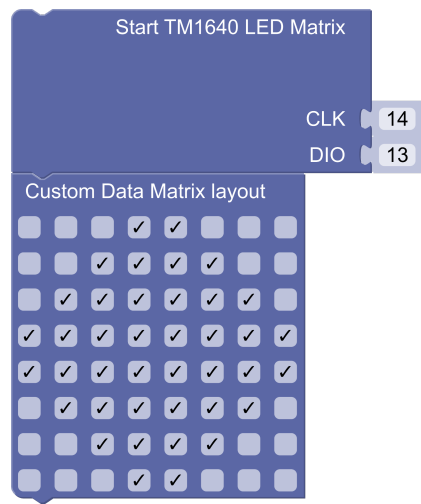
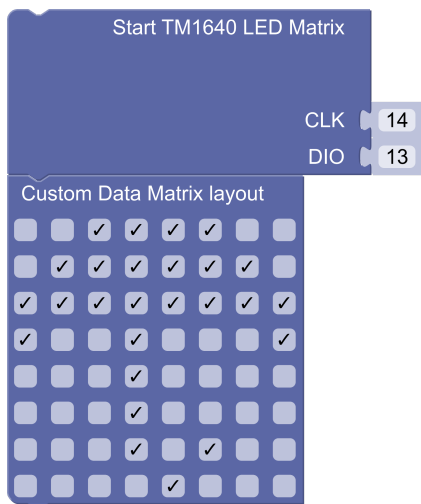
```
'<maxima>29</maxima><minima>19</minima><iuv>13.0</iuv></previsao><previsao><di  
a>2021-12-23</dia>']
```

```
Tempo = ci
```

A figura que segue mostra o sistema funcionando em uma placa Franzininho Wifi (<https://franzininho.com.br/>), uma placa de arquitetura aberta, desenvolvida no Brasil e baseada no processador ESP32S2. A placa possui um LED RGB e está alimentada por um *power bank* USB, mostrando a previsão do tempo em seu LED RGB, ao lado de um tradicional “Galinho do Tempo”.



Opcionalmente, é possível, inclusive, incluir uma matriz de LEDs e apresentar ícones de um guarda-chuvas ou do sol, conforme as condições do tempo. No exemplo abaixo, foi utilizada matriz de LEDs com controlador TM1640, e cada pixel da matriz foi definida no bloco “**Custom Data Matrix Layout**” do BIPES. Esta matriz de LEDs pode ser encontrada no formato de *shield*, que é um acessório que encaixa diretamente sobre outra placa, evitando a necessidade de cabos de conexão. Neste caso, foi utilizada a placa WeMos D1 mini com o *shield* Matriz de LEDs TM1640. A próxima figura mostra os blocos utilizados para acionar a matriz de LEDs e o seu resultado.

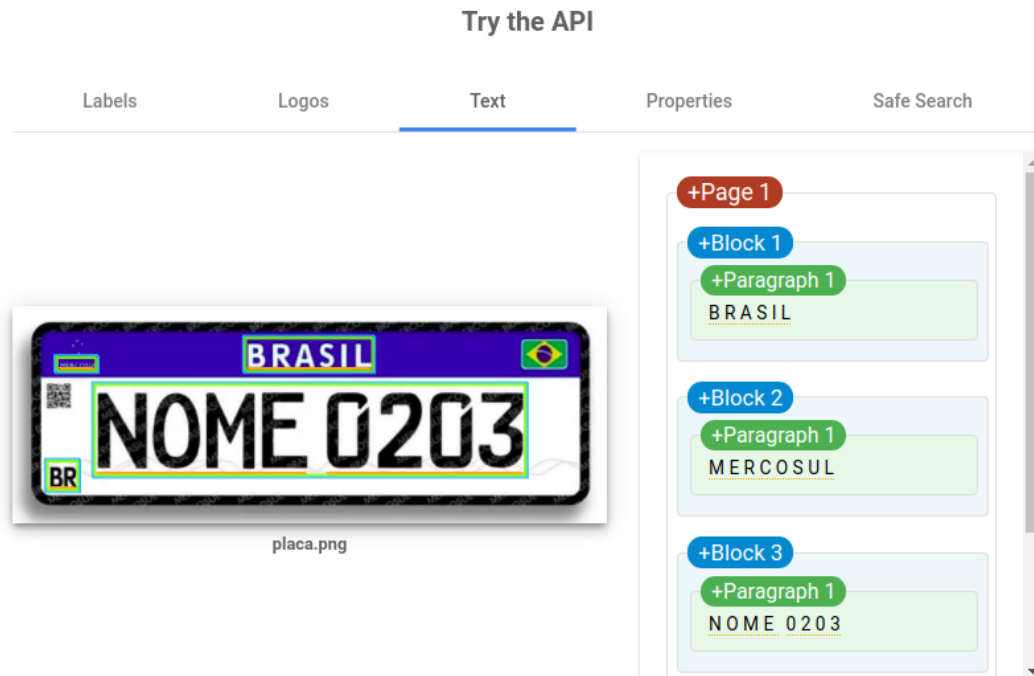


Cliente HTTP: Outras possibilidades

Como mencionado, existem milhares de opções de APIs para integração através de requisições HTTP. Nossos exemplos demonstraram envio de SMS e consulta às condições do clima, mas outro exemplo de serviço útil inclui o envio de mensagens via telegram através do **endpoint** <https://api.telegram.org/bot<token>/sendMessage>.

Uma outra possibilidade, mais avançada, é incluir funcionalidades de inteligência artificial e visão de máquina a um sistema embarcado, através do auxílio de serviços web de aprendizado de máquina. Assim, mesmo em um sistema embarcado com recursos computacionais limitados, é possível usar todo o poder de processamento de serviços em nuvem. Um exemplo de possibilidade está em sistemas de análise de imagens e identificação de objetos, pessoas e caracteres. Assim, um circuito de baixo custo, como um módulo ESP32-CAM, por exemplo, pode capturar uma imagem e a enviar, via HTTP, para um serviço em nuvem. O Google, a Amazon e a IBM possuem serviços avançados de processamento de imagem em nuvem, que permitem, inclusive inferir se uma pessoa está feliz ou triste, dentre outras funcionalidades. A figura a seguir mostra um exemplo de uma imagem de uma placa de carro, que pode ser enviada, via requisição HTTP, para o *Google Cloud Vision API*. Como resultado, o serviço do Google retorna uma lista dos caracteres

presentes na placa, realizando o serviço de reconhecimento óptico de caracteres (OCR). Note, inclusive, que o sistema detectou a frase “MERCOSUL” presente na placa, em sua parte superior esquerda, em pequena dimensão. Maiores informações sobre o uso desta API podem ser encontradas em: <https://cloud.google.com/vision/docs/drag-and-drop>.



Note que este é um exemplo de reconhecimento de texto, mas o teste poderia incluir reconhecimento de objetos, emoções, dentre outras características da imagem. Você também pode pesquisar na Internet por exemplos e projetos usando o módulo ESP32-CAM com o *Google Cloud Vision API* para verificar algumas possibilidades que já foram implementadas e disponibilizadas pela comunidade de desenvolvedores de sistemas embarcados.

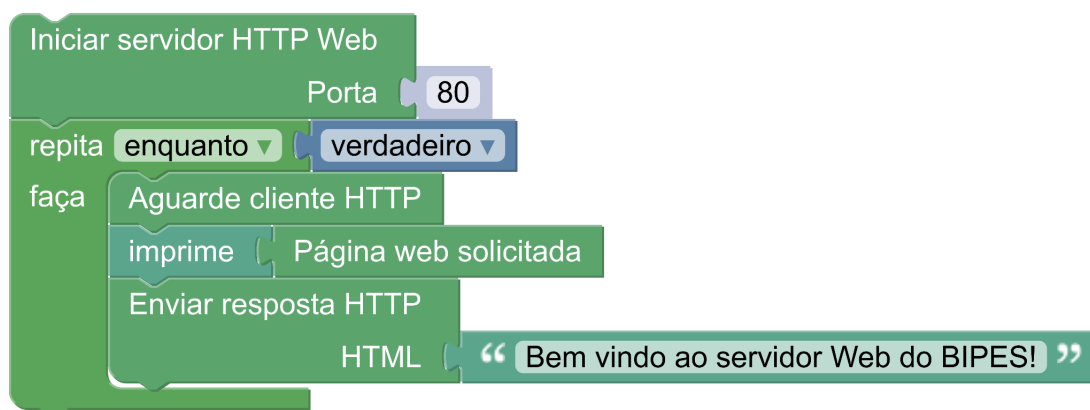
Estes recursos de nuvem também podem ser bastante úteis em aplicações de robótica e mecatrônica. Para maiores detalhes sobre o uso de nuvem integrado com robôs, uma possibilidade de leitura é:

Aroca, R. V., Péricles, A., de Oliveira, B. S., Marcos, L., & Gonçalves, G. (2012, June). *Towards smarter robots with smartphones*. In 5th workshop in applied robotics and automation, Robocontrol (pp. 1-6). sn.

Servidor HTTP

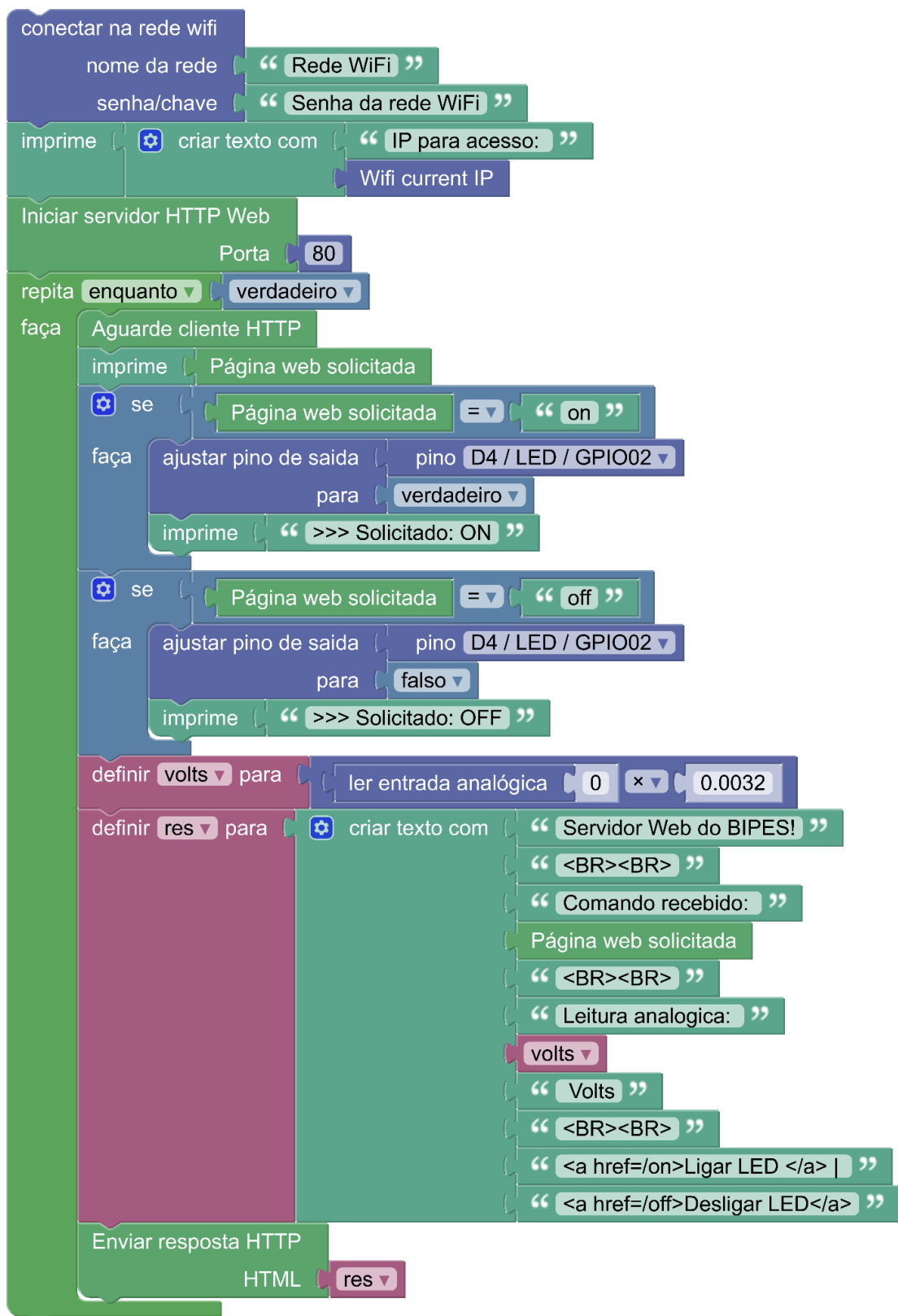
O BIPES também permite que uma placa ESP atue como servidor HTTP, viabilizando o acesso direto a partir de qualquer cliente HTTP, como por exemplo, um navegador web em um PC, um *smartphone*, ou mesmo outra placa ESP32 ou ESP8266 utilizando o bloco **Requisição HTTP GET**.

A próxima figura mostra a implementação de um servidor HTTP no BIPES. O servidor é iniciado na porta 80 e fica aguardando requisições HTTP de clientes continuamente.

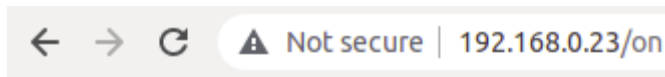


Também é possível adicionar mais funcionalidades neste servidor, como ilustrado na próxima figura. Neste segundo exemplo de servidor, ao ser acessado pelo seu endereço IP, o servidor exibe a leitura da entrada analógica da placa ESP8266 e oferece 2 links para páginas web neste servidor: */on* e */off*. O acesso a estas páginas é verificado pelos blocos **“SE Página web solicitada = on” (ou off)** e gera uma ação de acionamento de pino GPIO para ligar/desligar o LED ligado ao pino D4. Utilizando códigos HTML, é possível inserir figuras, caixas de texto para solicitar dados do usuário, dentre outras possibilidades de desenvolvimento web / HTML.

O bloco **“Wifi Current IP”** também foi utilizado para verificar o endereço IP de sua placa, facilitando identificar o IP para acessar a placa a partir de outros dispositivos em sua rede.



O resultado pode ser visto a partir de um navegador web, acessando o IP da placa ESP:



Servidor Web do BIPES!

Comando recebido: on

Leitura analogica: 0.0128 Volts

[Ligar LED](#) | [Desligar LED](#)

Resultado visto no **Console** quando o link “Ligar LED” foi acessado:

```
IP para acesso: ('192.168.0.23', '255.255.255.0')
BIPES HTTP Server Listening on ('0.0.0.0', 80)
client connected from ('192.168.0.10', 54604)
b'GET /on HTTP/1.1\r\n'
Request page = on
b'Host: 192.168.0.23\r\n'
b'Connection: keep-alive\r\n'
b'Upgrade-Insecure-Requests: 1\r\n'
b'User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4399.90 Safari/537.36\r\n'
b'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8\r\n'
b'Referer: http://192.168.0.23/off\r\n'
b'Accept-Encoding: gzip, deflate\r\n'
b'Accept-Language: en-US,en;q=0.9,pt-BR;q=0.8\r\n'
b'\r\n'
on
>>> Solicitado: ON
```

Note que, na maior parte dos casos, este servidor web somente vai poder receber requisições HTTP a partir de dispositivos conectados na mesma rede WiFi da placa ESP executando a aplicação servidor HTTP. Esta limitação depende da arquitetura da rede utilizada: na maioria dos casos, o ponto de acesso sem fio e/ou roteador utilizado obtém um único endereço de Internet (IP) que é público e acessível para receber conexões vindas da Internet. Tal roteador e/ou ponto de acesso distribui endereços IPs internos / privados que não são acessíveis a partir de outro dispositivo na Internet. Estes IPs internos têm o formato 192.168.x.y, 172.x.y.z ou 10.x.y.z. O único IP público é então compartilhado com todos dispositivos internos através de um processo chamado de *Network Address Translation* (NAT). Dessa forma, como a placa ESP tem um IP privado, para que ela seja acessada a partir da Internet, é preciso configurar uma estratégia de **redirecionamento de porta**, que será apresentada na próxima seção.

Mesmo com o IP com acesso apenas na rede interna, todos dispositivos na mesma rede poderão acessar o servidor web da placa ESP. O IP da placa poderia ser configurado, por exemplo, em uma aplicação feita no MIT App Inventor, permitindo integrar um aplicativo para *smartphone* com a placa ESP através de requisições HTTP.

Para conhecer mais detalhes sobre o processo de NAT e endereçamento IP, consulte um livro especializado de redes de computadores, como por exemplo, o livro do Prof. Tanenbaum:

TANENBAUM, A. S. – Redes de Computadores – 4ª Ed., Editora Campus (Elsevier)

Integração com Google Home ou Amazon Alexa

A área de Internet das Coisas tem tido um crescimento relevante na área de automação residencial, também chamada de domótica. Neste contexto, dois dispositivos, que também são sistemas embarcados, oferecem opções de assistentes pessoais com alta qualidade e flexibilidade: o **Google Home** e a **Amazon Alexa**. Dentre as várias funcionalidades destes dispositivos, a principal é a interação com o usuário por comandos e respostas de voz, onde o usuário pode fazer solicitações por voz de forma bastante simples. Por exemplo, alguns comandos possíveis que podem ser utilizados são: “Agendar um compromisso”, “Ligar a luz”, “Enviar uma mensagem para André”, “Lembre-me de tirar as roupas da máquina de lavar em 10 minutos”.

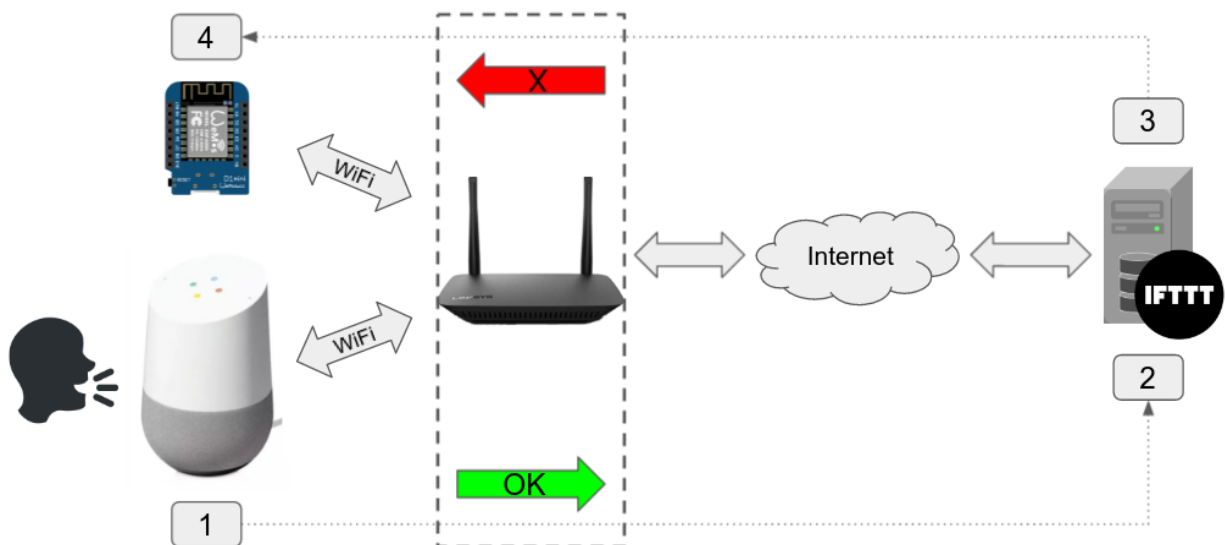
A flexibilidade destes assistentes pessoais inclui a possibilidade de integrá-los com novos serviços e aplicações, além de personalizar ações. Dentre as várias possibilidades de integração, utilizaremos o serviço IFTTT: **If This Then That** (<https://ifttt.com/>), que é gratuito para até 5 ações de automação, chamadas de *Applets*. Basicamente, o IFTTT oferece centenas de opções de eventos (**If This**) que podem disparar outras centenas de opções de ações (**Then That**). Além das centenas de opções de ações, o IFTTT também é compatível com o Google Home e com a Alexa da Amazon. E, em especial, o IFTTT possui a opção de ação **WebHook** que permite realizar uma requisição HTTP quando algum evento é detectado: ou seja, a requisição HTTP é realizada no contexto da ação (**Then That**).

Assim, é possível criar um *Applet* no IFTTT que aguarda uma frase do tipo “Acenda a Luz”, e quando esta frase é falada, realiza uma requisição HTTP para o servidor Web do BIPES embarcado na própria placa ESP. Contudo, como já mencionado, o servidor HTTP embarcado pelo BIPES na placa ESP estará inacessível para o servidor do serviço IFTTT, de forma que será necessário realizar uma adequação técnica para que a requisição chegue até a placa ESP. A seguir, apresentam-se duas opções para que os dados do Google Home ou Amazon Alexa, passando pelo IFTTT, cheguem até a placa ESP que realiza o controle do dispositivo desejado.

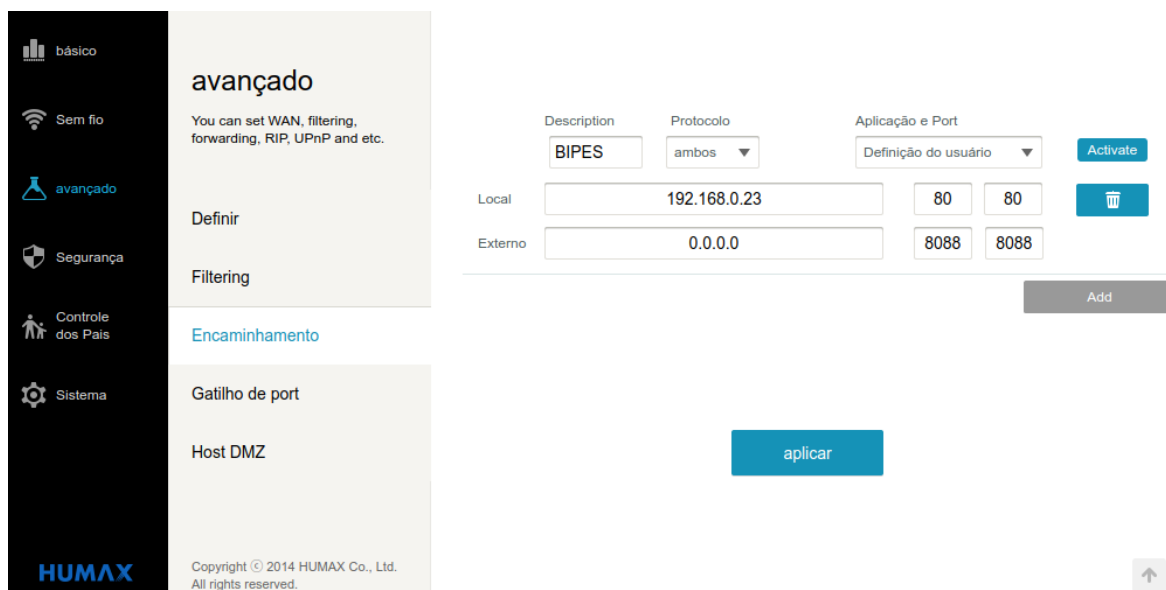
Opção 1: Redirecionamento de porta / encaminhamento

A próxima figura ilustra a arquitetura de um sistema onde um comando de voz feito para o Google Home resulta em um acionamento de um dispositivo através de uma placa ESP8266. Na Figura, o usuário fala o comando de voz que definimos como “*Shine my light*”. O Google Home suporta comandos em português, mas o IFTTT ainda não tem suporte à língua portuguesa. Sendo assim, optamos pelo comando em inglês.

Ao receber o comando de voz, o Google Home (1) envia dados para o servidor do IFTTT (2), um serviço em nuvem processa o comando em seu *Applet* e realiza uma requisição HTTP (3) que é recebida pela placa ESP8266 (4). Note que o envio de dados [(1) para (2)] apresenta uma seta verde, onde, na maioria dos casos, não há restrições para envio de requisições de um endereço IP privado interno para um endereço IP público externo. Contudo, a requisição do servidor do IFTTT para a placa ESP8266 [(3) para (4)] pode não chegar até a placa ESP8266. Para tanto é necessário realizar um redirecionamento de portas no roteador e/ou ponto de acesso e/ou firewall (mencionamos vários e/ou, pois a arquitetura de cada de cada rede pode demandar diferentes configurações).



Para realizar esta operação, é necessário configurar um redirecionamento (encaminhamento) de porta TCP do IP público/externo de sua rede para o IP interno da placa ESP8266. Verifique as opções de seus equipamentos de rede. Como ilustração, a próxima figura mostra a configuração de encaminhamento de portas em um roteador de Internet via cabo da marca HUMAX.



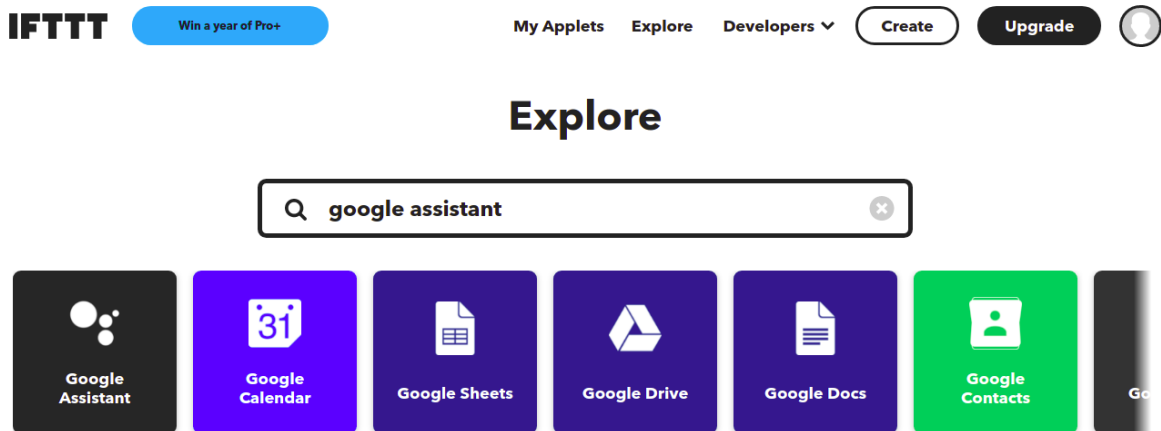
Conforme teste que já realizamos na seção anterior, uma placa ESP8266 recebeu o IP privado 192.168.0.23 e pode ser acessada, a partir da rede interna, pelo endereço <http://192.168.0.23/>. Note que o roteador HUMAN foi configurado em opções avançadas → Encaminhamento, ativando um encaminhamento de porta do endereço externo 0.0.0.0 (que significa qualquer endereço externo) para o endereço interno 192.168.0.23. Além disso, também foi configurado para que a porta externa 8088 seja mapeada para a porta interna 80, do IP 192.168.0.23.

O próximo passo é verificar qual o seu endereço IP. Uma possibilidade é pesquisar no Google pela frase “*what is my ip address*”. O próprio Google irá mostrar um endereço IP identificado como “*Your public IP address*”, ou seja “seu endereço IP público”. Com o encaminhamento de porta ativo, você pode acessar, a partir de qualquer parte da Internet, o seu IP seguido da porta 8088. Por exemplo, se o IP for 187.66.80.187, acesse, em um navegador web, o endereço <http://187.66.80.187:8088/>. Certifique-se de que o programa do servidor HTTP esteja ativo na placa ESP8266. Caso você consiga acessar a placa por este endereço IP público, o IFTTT também conseguirá.

Antes de prosseguir, uma pequena observação: o seu endereço IP público pode mudar dinamicamente depois de alguns dias ou semanas. Esta possibilidade de mudança pode ocorrer dependendo das condições de seu provedor de Internet. Caso você precise utilizar o sistema com frequência, é necessário configurar um serviço de DNS dinâmico, que lhe fornecerá um nome na Internet sempre sincronizado com seu IP, mesmo que ele mude. Um exemplo de serviço de DNS dinâmico é o NOIP: <https://www.noip.com/pt-BR>.

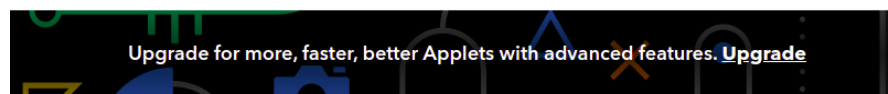
Conhecendo o IP público para realizar requisições HTTP, já é possível configurar o *Applet* do IFTTT. Primeiro, acesse o <https://ifttt.com/> e crie uma conta, caso você ainda não tenha. É importante usar o mesmo email/conta já utilizado no Google Home ou Alexa. A seguir uma configuração para o Google Home é ilustrada, sendo similar para a Amazon

Alexa. Após criar a conta, clique em **Explore** e procure por *Google Assistant*, conforme a próxima figura.



Siga os próximos passos e confirme a associação do IFTTT com o Google Home. Este procedimento só precisa ser feito uma única vez, e esta conexão ficará ativa. Na sequência é possível criar um *Applet* do IFTTT. Para tanto, clique em **Create** na tela principal do IFTTT:

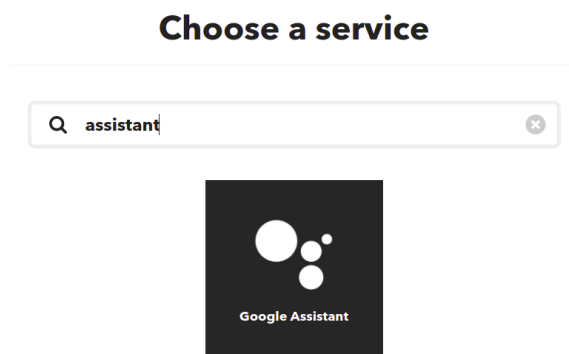
Create your own



You're using 2 of 5 Applets



Clique em **“If This”** e escolha **Google Assistant**:



Escolha a opção **“Say a simple phrase”** e defina uma frase. Neste exemplo, usamos a frase: **“Shine my light”**. Também é possível especificar o que o sistema vai falar em resposta a este comando através do campo **“What do you want the Assistant to say in response?”**.



Clique em **Create trigger**. Em seguida, vamos adicionar a ação a ser feita. Clique em **“Then That”** e escolha **Webhook** e depois escolha a opção **“Make a web request”**.

Choose a service

Q webhooks



Webhooks



Webhooks

Make a web request

This action will make a web request to a publicly accessible URL. NOTE: Requests may be rate limited.

Em “**Make a web request**”, informe a **URL** completa do serviço HTTP que acende a luz através da ESP8266. A URL deve incluir o seu endereço IP público com o caminho e nome do recurso. Por exemplo, <http://187.66.80.187:8088/on> ou <http://187.66.80.187:8088/off>. A figura abaixo mostra a URL <http://X.Y.Z.K/on>, de forma genérica. Note que o endereço público correto deve ser digitado neste campo. Escolha também o método GET para a requisição HTTP a ser realizada.



Make a web request

This action will make a web request to a publicly accessible URL. NOTE: Requests may be rate limited.

URL

http://X.Y.Z.K/on

Surround any text with <<< and >>> to escape the content. Surround any text with <<< and >>> to escape the content. See [FAQ](#) if using an IPv6 URL.

Add ingredient

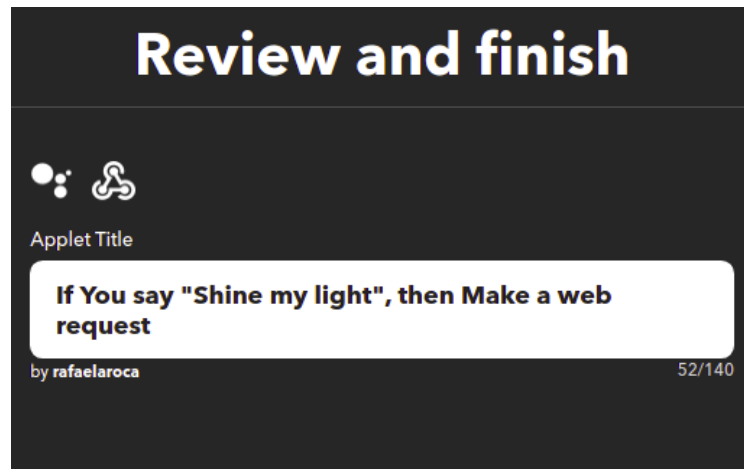
Method

GET



The method of the request e.g. GET, POST, DELETE

Depois clique em **Create action** e depois em **Continue**, seguido de **Finish**. Finalmente, para testar, vá até o Google Home e fale: **“Ok Google, Shine my Light!”**.



Receive notifications when this Applet runs

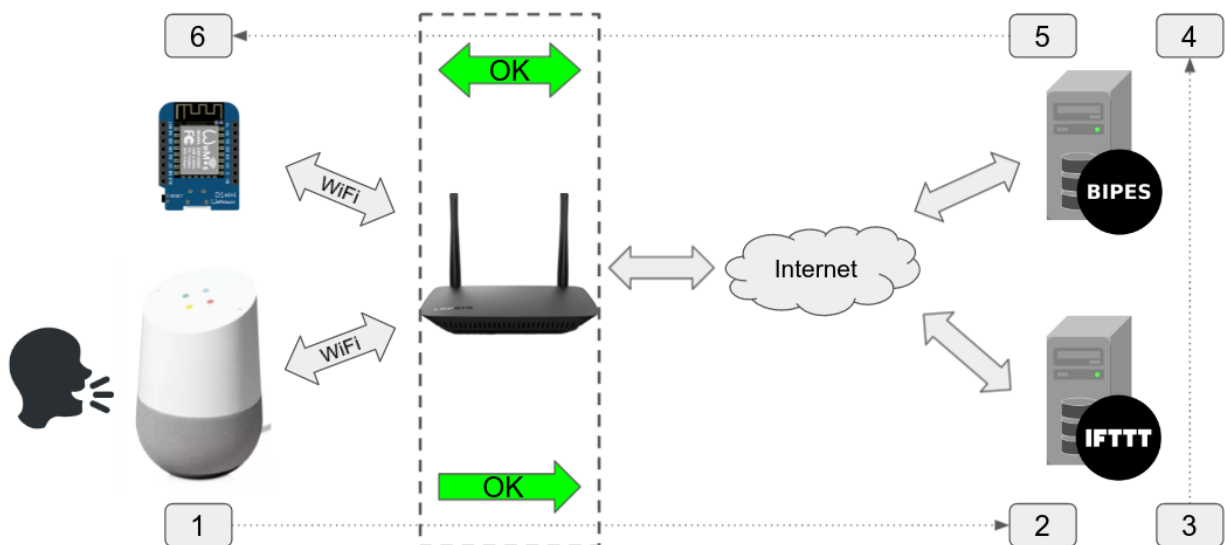


Finish

Opção 2: Integração com o IFTTT através do BIPES EasyMQTT

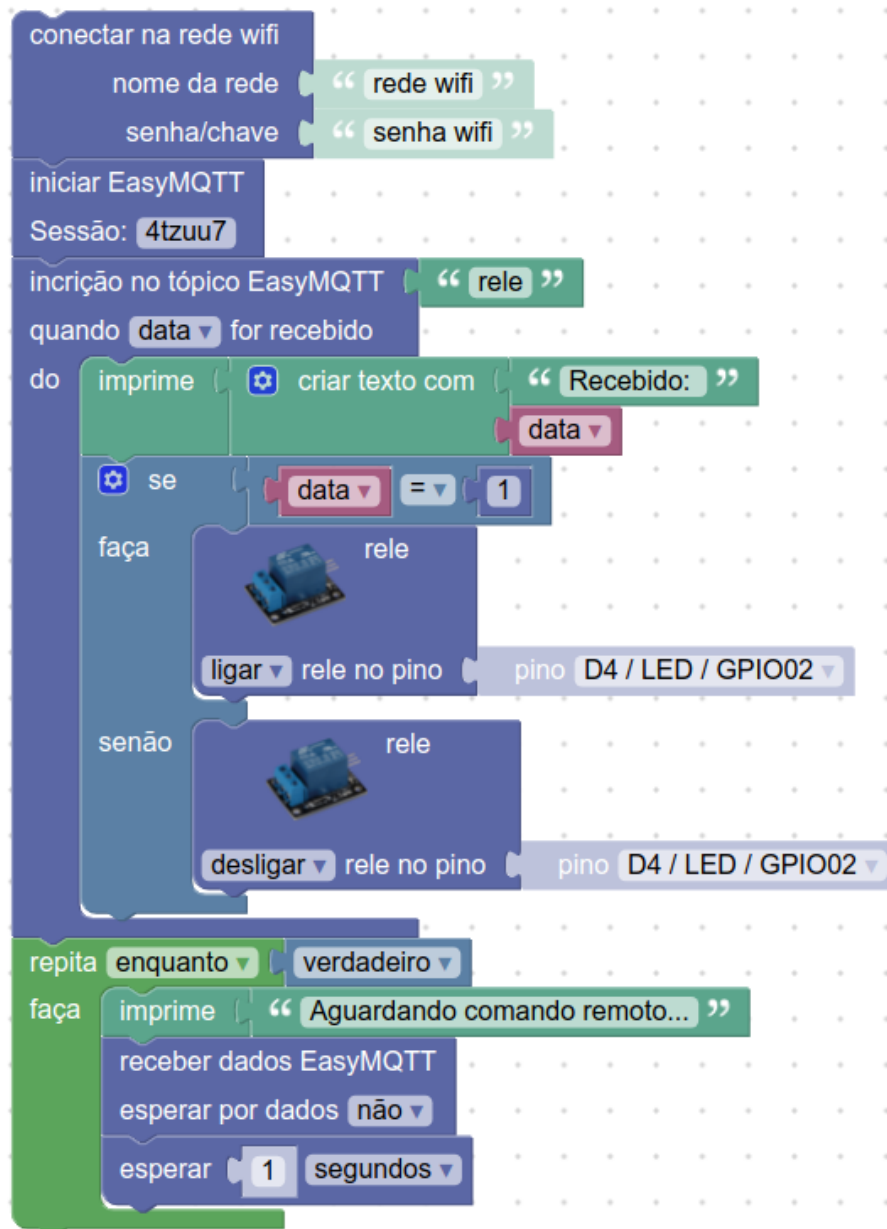
Podem ocorrer situações em que a senha do roteador / ponto de acesso / *firewall* não está disponível, não sendo possível realizar a configuração de redirecionamento de porta. Alguns provedores de Internet, também podem acabar não fornecendo IPs públicos para seus clientes, ou, por segurança, oferecem conexões de Internet com *firewalls* bastante restritivos, limitando a recepção de fluxos de dados da Internet que não tenham sido iniciados a partir da sua rede. Para estes e outros casos, é possível realizar uma integração entre o BIPES e o IFTTT através do servidor web <http://bipes.net.br> do próprio projeto BIPES.

A próxima figura ilustra esta arquitetura de sistema: o usuário emite um comando de voz “**Shine my light**” para o Google Home (1) que faz uma requisição para o servidor do IFTTT (2). O IFTTT executa então um *Applet* (3) que realiza uma requisição HTTP GET para o servidor web do projeto BIPES (<http://bipes.net.br>) (4), publicando um valor em um tópico de uma sessão EasyMQTT do BIPES. O servidor do BIPES, por sua vez, atualiza aquela sessão (5) de forma sincronizada com a placa ESP8266, que deve estar executando um programa inscrito em um tópico EasyMQTT, que então recebe o dado e executa uma ação (6).



Note que, desta vez, não há um servidor HTTP na placa ESP8266, mas apenas um programa inscrito em um tópico de uma sessão EasyMQTT. Este programa já foi ilustrado neste livro na seção “Controlando Dispositivos via Internet”. Para facilitar a compreensão, apresentamos o mesmo programa aqui novamente. Este programa se inscreve no tópico rele da sessão EasyMQTT **4tzuu7** do BIPES, e, a qualquer mudança neste tópico, executa a ação descrita dentro do bloco “**inscrição no tópico**”.

Programa:



Lembre agora que já utilizamos uma URL do servidor do projeto BIPES para alterar dados de um tópico de uma sessão EasyMQTT e controlar um dispositivo a partir de um aplicativo para *smartphones* feito com o MIT App Inventor. A URL utilizada para integração com o aplicativo para *smartphone* foi:

<http://bipes.net.br/easymqtt/publish.php?session=4tzuu7&topic=rele&value=1>

Agora, esta mesma URL pode ser utilizada para integração com o IFTTT. O *Applet* do IFTTT é idêntico ao anterior, mudando apenas a URL acionada pelo IFTTT quando o evento ocorre:

Edit action fields

Make a web request

URL

```
http://bipes.net.br/easymqtt/publi  
sh.php?  
session=4tzuu7&topic=rele&value  
=1
```

Surround any text with <<< and >>> to escape the content. Surround any text with <<< and >>> to escape the content. See [FAQ](#) if using an IPv6 URL.

Add ingredient

Salve as mudanças e faça o teste novamente, solicitando ao Google Home: “**Shine my light**”. Neste caso não é preciso nenhuma configuração especial de redirecionamento de portas / encaminhamento de pacotes, como na opção 1, de forma que o sistema funciona independentemente da configuração e propriedades da rede, porém depende do servidor do BIPES.

Observação: note que o BIPES é um projeto aberto e gratuito e a sessão usada como exemplo aqui (**4tzuu7**) pode estar sendo utilizada por outros usuários/leitores, causando efeitos inesperados (mudança no estado de um tópico). Assim, defina uma sessão exclusiva para seu projeto. Você pode consultar se uma sessão EasyMQTT está vazia/disponível pela URL: <https://bipes.net.br/beta2/easymqtt/getsession.php?session=12>, onde 12 é a sessão que se deseja consultar.

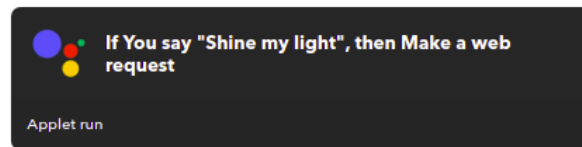
Finalmente, pode ser útil saber que o IFTTT oferece a opção “**View Activity**”, que permite verificar os registros (*logs*) de recepção de eventos e resultados de ações. Em especial, no caso dos **Webhooks**, é possível verificar se o comando foi enviado ao servidor, e o resultado da requisição HTTP. A figura a seguir mostra um exemplo de requisição realizada para o servidor EasyMQTT do projeto BIPES.

If You say "Shine my light", then Make a web request



Applet ran

Jan 02 - 6:43 PM



Google Assistant

Say a simple phrase

Trigger ran, 6:43 PM

CreatedAt

January 2, 2022 at 06:43PM



Webhooks

Make a web request

Action ran, 6:43 PM

url

<http://bipes.net.br/easymqtt/publish.php?session=4tzuu7&topic=rel&value=1>

method

```
{"label": "GET", "value": "GET"}
```

Atividade: servidor e cliente - controle remoto via WiFi



Utilize um par de placas ESP8266, programando uma para atuar como ponto de acesso sem fio e servidor HTTP. Configure placa outra como estação WiFi e cliente, de forma que esta segunda placa, a cliente, envie uma requisição HTTP toda vez que um pino digital for acionado (por um sensor, um botão ou outro dispositivo). Quando a placa configurada como servidor receber a requisição HTTP, ela deverá acionar um pino de saída, de forma a ligar/desligar algum dispositivo. Dessa forma, uma placa ESP8266 atuará como controle remoto sem fio, via WiFi, de um dispositivo conectado à placa ESP8266 atuando como "servidora".

* * * *

Parabéns! Você desenvolveu aplicações Internet das Coisas (IoT) completas com nuvem, sensores, controle remoto e painel (dashboard) de visualização e controle!

* * * *

Atividades extra

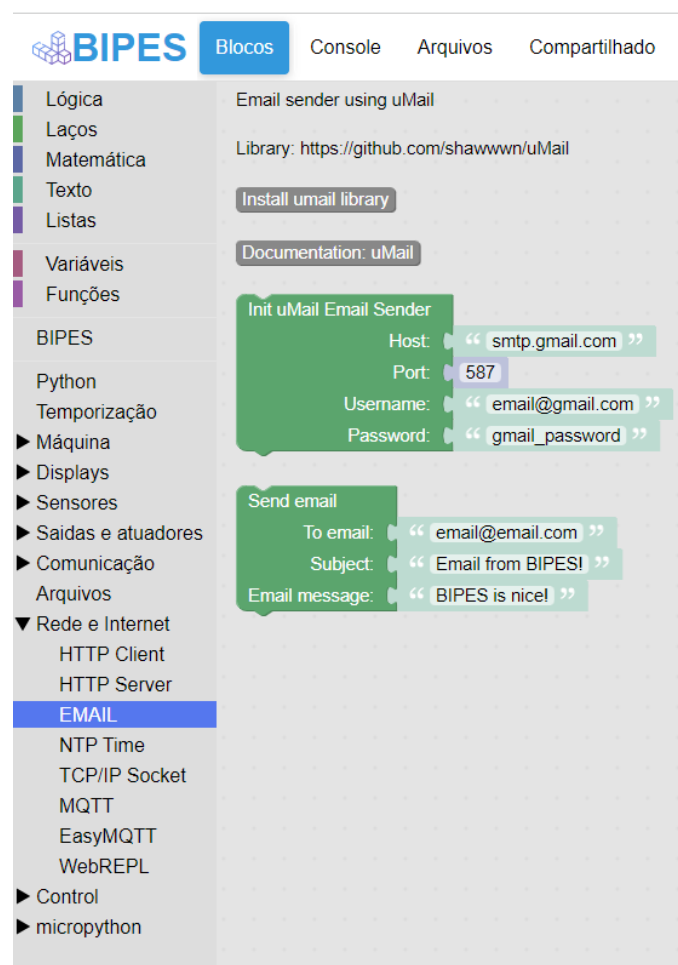
Envio de email

Atividade:

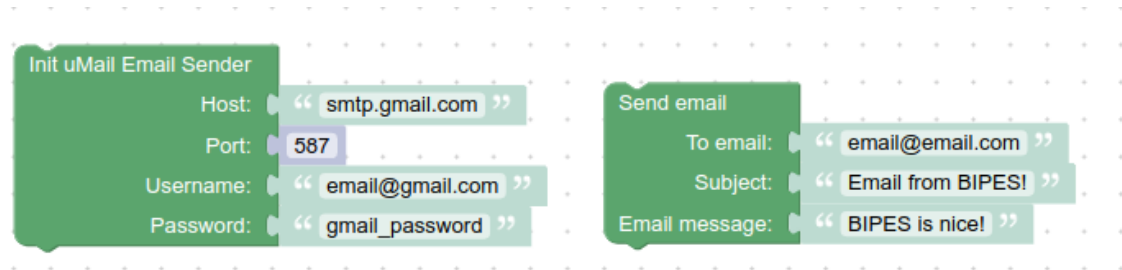
Elabore um programa que envie um email para você quando um pino de entrada digital (GPIO) da ESP8266 detectar uma mudança no seu sinal de entrada (de 0 para 1, por exemplo). Este pino poderá ser conectado a um sensor de presença ou sensor de porta aberta (*reed switch*). Ao detectar a mudança, o programa deve realizar o envio do email.

Dicas:

1. Para realizar esta atividade, a placa ESP8266 deve estar conectada à Internet;
2. O **BIPES** possui um sistema modular de bibliotecas. Para enviar email, é preciso instalar a biblioteca *uMail*. Isso pode ser feito facilmente ao acessar a guia lateral **Rede e Internet >> EMAIL** e depois clicar em “**Install uMail library**”.



3. Após instalar a biblioteca na placa, você pode usar os blocos de inicialização e de envio de email:



GMAIL

O **BIPES** utiliza a biblioteca `umail` do MicroPython, que permite envio de email por praticamente qualquer provedor de email. Entretanto, cada provedor pode precisar de configurações específicas.

O GMAIL, por exemplo, requer que se crie uma senha de aplicação (***application password***) para que a ESP8266 possa enviar emails diretamente.

A conta GMAIL que será utilizada deve ter a autenticação de 2 fatores ativada e o "App Password" ativado.

O link a seguir mostra uma explicação mais detalhada:

<https://myaccount.google.com/apppasswords>

Neste link, selecione:

App >> E-mail,
Device >> Other

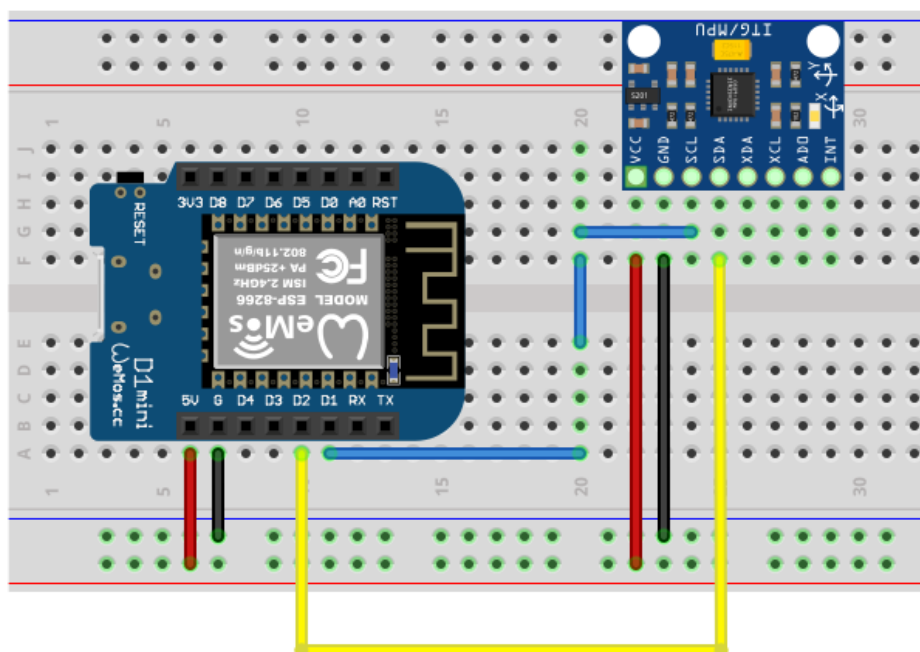
O GMAIL irá gerar uma senha dedicada para a ESP8266. Copie esta senha e cole no bloco **Init uMail Email Sender**.

Unidade de Medidas Inerciais - MPU6050

Caso você tenha uma unidade de medidas inerciais MPU6050, você pode utilizá-la conforme diagramas abaixo e uso o bloco IMU / MPU6050:

A tabela a seguir mostra as conexões necessárias e ilustradas na imagem:

<u>MPU6050</u>	<u>Pino da ESP8266</u>
<u>GND</u>	<u>GND</u>
<u>VCC</u>	<u>5V</u>
<u>SCL</u>	<u>5 (D1)</u>
<u>SDA</u>	<u>4 (D2)</u>



Atividade:

Elabore um dashboard IoT que mostra a variação dos ângulos de orientação da placa em gráficos em tempo real.

Desafio:

Implemente um programa para calcular os ângulos (neste caso, usando apenas acelerômetros) de inclinação nos três eixos a partir das medidas de aceleração da gravidade, e prepare uma seta no dashboard IoT que indique a inclinação da placa.

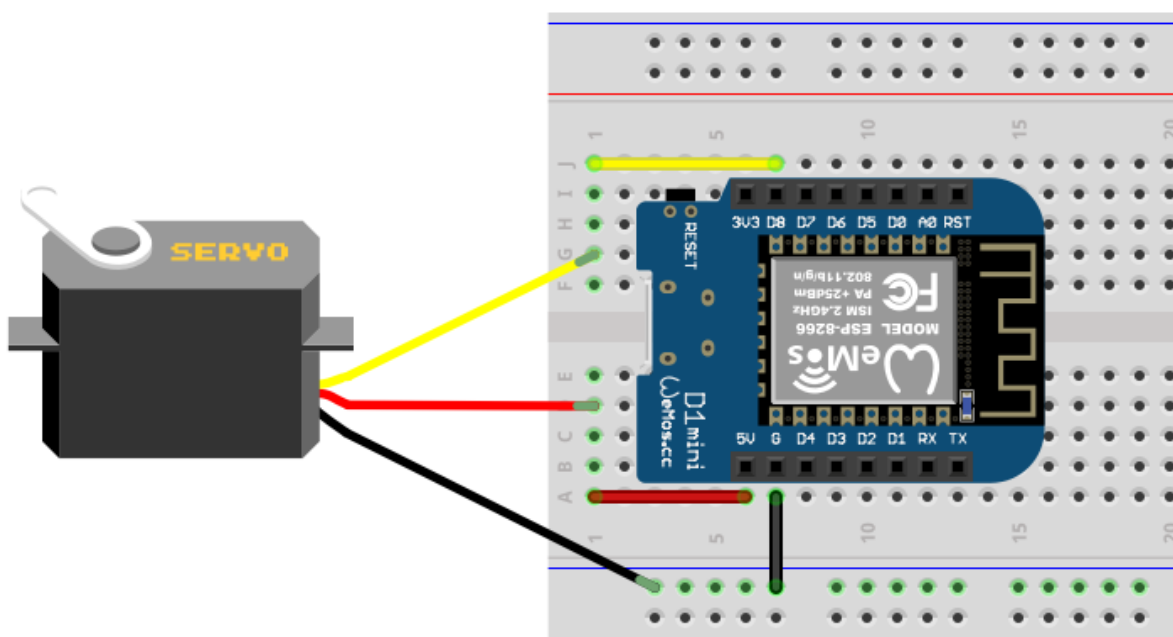
Servo motor de aeromodelos (RC Servo)

Atividade:

Elabore um programa com um dashboard IoT contendo um controle deslizante (*slider*). Quando o usuário alterar o valor do *slider*, o servo deve se mover para o ângulo ajustado no *slider*. **Dica:** utilize o bloco **RC Servo Motor**, na guia lateral **Saídas e atuadores**.

A tabela a seguir mostra as conexões necessárias e ilustradas na imagem:

<u>Servo</u>	<u>Pino do ESP8266</u>
<u>GND (marrom)</u>	<u>GND</u>
<u>VCC (vermelho)</u>	<u>5V</u>
<u>S / Sinal (laranja)</u>	<u>15 (D8)</u>



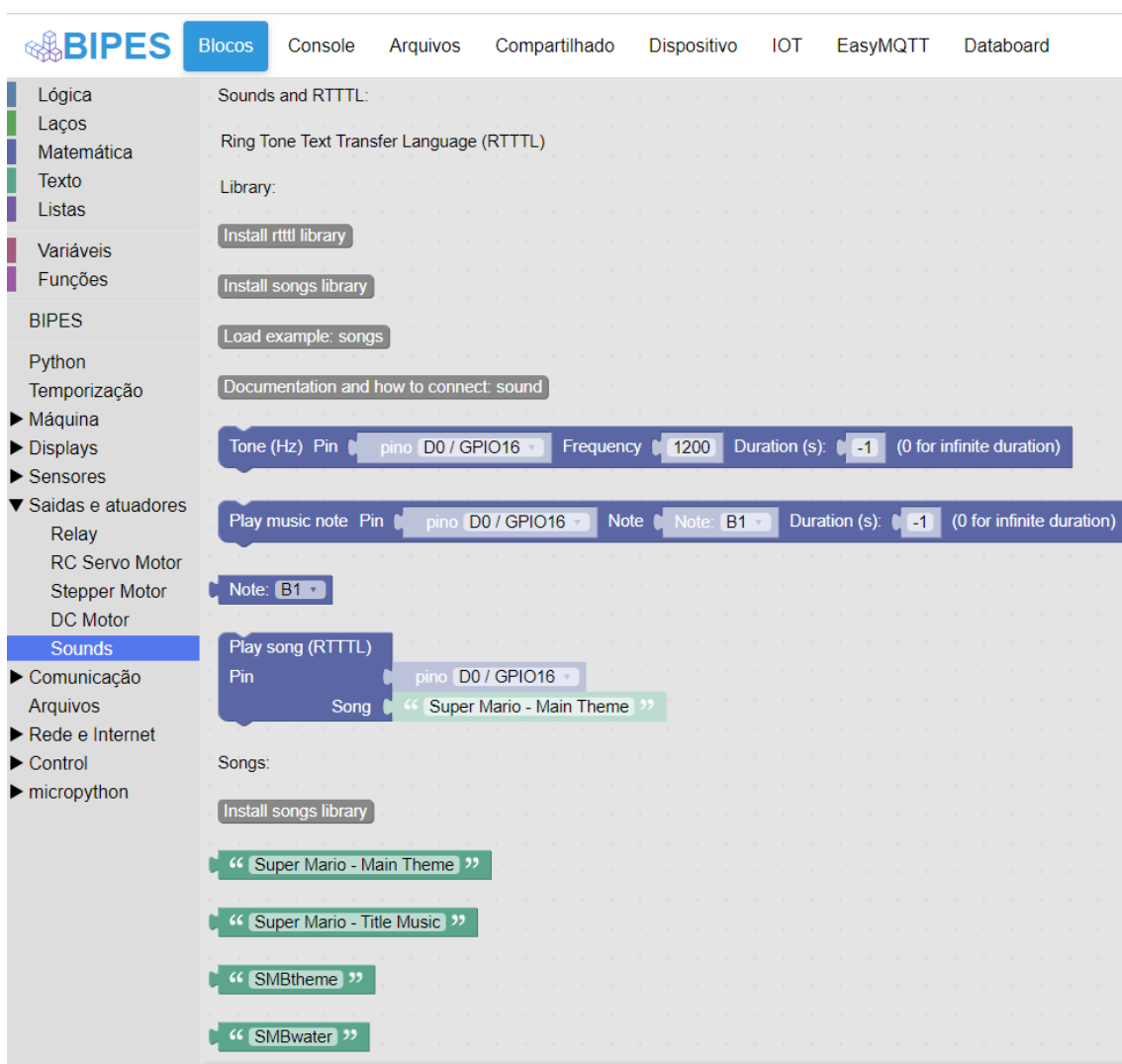
Desafio:

Implemente um programa do tipo "gimbal", que ajusta a posição do servo, conforme o ângulo medido pela unidade inercial MPU6050 (neste caso, usando apenas acelerômetros).

Música

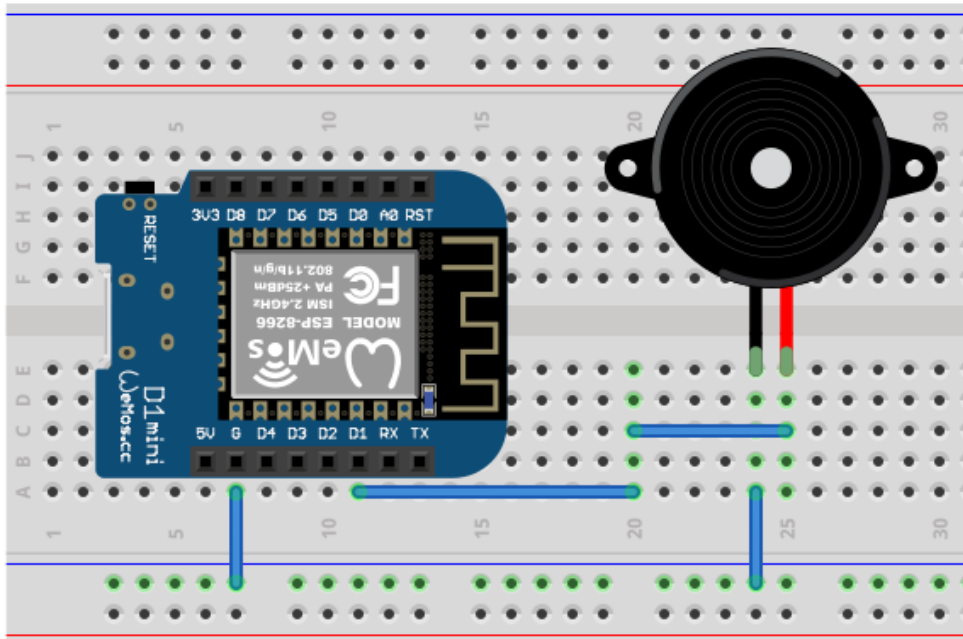
O BIPES também possui blocos para gerar tons sonoros e tocar músicas completas! Verifique blocos e opções na guia lateral **Saídas e atuadores >> Sounds**. Para reproduzir uma música, basta utilizar um *buzzer*.

Para produzir sons o **BIPES** utiliza uma biblioteca externa. Acesse a guia lateral **Saídas e atuadores >> Sounds**, clique em **Install rtttl library** e depois clique em **Install songs library**. Dois arquivos serão baixados e instalados na memória da placa para reprodução de áudio. A placa deve estar conectada à Internet para executar estes comandos.



The screenshot displays the BIPES web interface. The top navigation bar includes 'BIPES', 'Blocos', 'Console', 'Arquivos', 'Compartilhado', 'Dispositivo', 'IOT', 'EasyMQTT', and 'Databoard'. A left sidebar lists various categories: Lógica, Laços, Matemática, Texto, Listas, Variáveis, Funções, BIPES, Python, Temporização, Máquina, Displays, Sensores, Saídas e atuadores (Relay, RC Servo Motor, Stepper Motor, DC Motor), Sounds (highlighted), Comunicação, Arquivos, Rede e Internet, Control, and micropython. The main content area is titled 'Sounds and RTTTL:'. It features a 'Library:' section with buttons for 'Install rtttl library', 'Install songs library', and 'Load example: songs'. Below this is a 'Documentation and how to connect: sound' link. The 'Tone (Hz)' block is configured with Pin 'pino D0 / GPIO16', Frequency '1200', and Duration '(s)' '-1'. The 'Play music note' block is configured with Pin 'pino D0 / GPIO16', Note 'Note: B1', and Duration '(s)' '-1'. A 'Note: B1' block is also present. The 'Play song (RTTTL)' block is configured with Pin 'pino D0 / GPIO16' and Song 'Super Mario - Main Theme'. A 'Songs:' section contains an 'Install songs library' button and a list of songs: 'Super Mario - Main Theme', 'Super Mario - Title Music', 'SMBtheme', and 'SMBwater'.

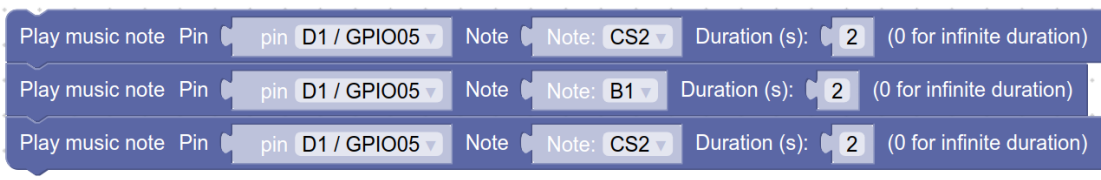
Depois disso, você pode montar um circuito com a placa ESP8266 e um *buzzer*, conforme sugerido a seguir:



Na sequência você já pode tocar músicas! Note que existe uma biblioteca com várias músicas prontas, que você pode usar diretamente, ou até incluir / criar suas músicas, editando o arquivo `songs.py`.



Também é possível reproduzir uma sequência de notas musicais tocando sua própria música:



Disponibilizamos um vídeo no YouTube com exemplos de algumas músicas sendo reproduzidas pela placa ESP32: <https://www.youtube.com/watch?v=a7U-sz70Wrg>

BIPES em outras plataformas

O **BIPES** está disponível para várias outras plataformas e oferece várias outras possibilidades, incluindo blocos de controle Proporcional-Integral-Derivativo (PID) em malha fechada, RFID, visão computacional com OpenCV, dentre outros. Explore, aproveite, compartilhe e contribua com o projeto **BIPES**!

Python com Arduino - SNEK

O **BIPES** também funciona com Arduino, graças ao SNEK Lang (<https://sneklang.org/>). Saiba mais em: <https://bipes.net.br/snek-web-uploader/>.

Outras

O **BIPES** suporta várias plataformas de microcontroladores e SoCs (*System on Chip*), como mBed, MicroBit, STM32 dentre outras. O **BIPES** também pode ser usado com placas Raspberry Pi Pico com MicroPython ou Raspberry Pi (e similares) com Linux. Inclusive, o BIPES foi o primeiro projeto a permitir programação por blocos da Raspberry Pi Pico. Verifique mais detalhes no site do projeto BIPES: <https://bipes.net.br/>.

Obtendo ajuda e ajudando

O **BIPES** possui uma comunidade de suporte aos usuários no github. Caso tenha dúvidas, críticas ou sugestões, acesse nossa comunidade e sinta-se a vontade para perguntar, ajudar e contribuir com o projeto através do link:

<https://github.com/BIPES/BIPES/discussions/categories/português>

Você também pode contribuir com o **BIPES** de diversas formas. Conheça mais sobre o projeto em <https://github.com/BIPES> e <https://bipes.net.br/>.

Considerações Finais

Aplicações de sistemas embarcados e Internet das Coisas (IdC ou *IoT*) podem ser úteis em várias ocasiões. O uso do BIPES visa facilitar a prototipação rápida de aplicações embarcadas e de *IoT*, além de contribuir para um acesso mais fácil e intuitivo à programação destes sistemas para usuários menos experientes. Além disso, espera-se que o BIPES também aumente a produtividade de usuários experientes.

O BIPES pode contribuir para projetos de robótica educacional e projetos *maker*, oferecendo uma plataforma intuitiva de programação, coleta, armazenamento e visualização de dados, além de dispensar a necessidade de instalação de softwares ou configurações especiais, facilitando o uso em diversos tipos de computadores e dispositivos. Oferece ainda uma forma diferenciada de aprender programação por código, já que o programa em blocos é convertido automaticamente para Python, sendo que o código Python pode ser editado, modificado e testado de forma rápida e simples.

No ambiente acadêmico, pode ser usado como ferramenta para automatizar experimentos de laboratório, coletar dados ou facilitar a realização de práticas que precisem de algum monitoramento ou automação.

Este livro se baseou, também, no uso de dispositivos de baixo custo, como módulos ESP8266 e ESP32, já amplamente utilizados em aplicações de automação residencial, comercial, acadêmicas, e, em alguns casos, aplicações de monitoramento de máquinas em indústrias, no campo e em outros ambientes.

Um outro pilar do projeto BIPES é o MicroPython ou sua variante, o CircuitPython. Embora seja uma versão de Python otimizada para microcontroladores, existem vários casos de sucesso de aplicações industriais usando MicroPython e até mesmo projetos espaciais baseados em MicroPython. A Agência Espacial Europeia, a ESA, por exemplo, possui um projeto de uso de MicroPython em aplicações espaciais⁵. Também existem vários kits didáticos de satélites educacionais no mercado baseados em processadores ESP32, como os satélites educacionais da PION, da MySatKit e da IdeaSpace.

Esperamos que este texto tenha sido útil e agradecemos sua atenção!

⁵ <https://essr.esa.int/project/micropython-for-leon-pre-qualified-version>

Agradecimentos

O projeto **BIPES** utiliza / integra várias ferramentas de software livre, em especial: o **Google Blockly**, o **MicroPython**, o **freeboard**, o **CodeMirror**, bibliotecas para MicroPython, o **esp-web-tools**, dentre outras. A equipe do **BIPES** deixa um grande agradecimento para todos os desenvolvedores destas ferramentas livres.

Agradecemos aos colegas da equipe do projeto **BIPES**, incluindo Gustavo Tamanaka pelo logotipo do projeto e Caio Augusto Silva pela implementação do módulo EasyMQTT.

Agradecemos ao projeto **Fritzing**, que viabilizou os desenhos neste texto. Agradecemos o apoio da Universidade Federal de São Carlos por meio de sua Pró-Reitoria de Extensão.

Agradecemos ao Matheus Calbo Aroca por ser o *beta tester* deste livro e seus exemplos.

Também agradecemos ao Conselho Nacional de Desenvolvimento Científico e Tecnológico e ao Ministério da Ciência, Tecnologia e Inovações (CNPq/MCTI) pelo apoio ao projeto **BIPES** (Processo CNPq 306315/2020-3).



Block based Integrated Platform for Embedded Systems
Plataforma Integrada baseada em blocos para Sistemas Embarcados

<http://bipes.net.br>

Apoio:



MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÕES



Uma introdução à Internet das Coisas e Sistemas Embarcados utilizando programação por blocos com BIPES e ESP8266 / ESP32 (<https://bipes.net.br>)

Dezembro/2021 - 1ª Edição

Rafael Vidal Aroca
Wesley Flavio Gueta
Jorge André Gastmaier Marques
Tatiana de Figueiredo Pereira Alves Taveira Pazelli

ISBN: 978-65-00-36921-2

BR



9 786500 369212