Olimpíada Brasileira de Satélites MCTI

Programação de *CanSats PION* e *CubeSats PION* por blocos usando BIPES

Olimpíada Brasileira de Satélites MCTI
Ministério da Ciência, Tecnologia e Inovações - MCTI

Fevereiro de 2022

Olimpíada Brasileira de Satélites MCTI

-- OBSAT --

OBSat

Programação de *CanSats PION* e *CubeSats PION* por blocos usando BIPES

Rafael Vidal Aroca
Wesley Flávio Gueta
Augusto Almeida de Jesus
Calvin Souto Trubiene
João Pedro Vilas Boas Silva
Jorge André Gastmaier Marques

Olimpíada Brasileira de Satélites MCTI
Ministério da Ciência, Tecnologia e Inovações - MCTI
Fevereiro de 2022

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

Programação de cansats pion e cubesats pion por blocos usando bipes [livro eletrônico] / Rafael Vidal Aroca ... [et al.]. -- 1. ed. -- São Carlos, SP : Rafael Aroca, 2022.
PDF.

Outros autores : Wesley Flávio Gueta, Augusto Almeida de Jesus, Calvin Souto Trubiene, João Pedro Vilas Boas Silva, Jorge André Gastmaier Marques.
Bibliografia.
ISBN 978-65-00-40369-5

1. Ciência da computação 2. Internet 3. Internet das coisas 4. Linguagem de programação (Computadores) 5. Programação (Computadores eletrônicos) I. Aroca, Rafael Vidal. II. Gueta, Wesley Flávio. III. Jesus, Augusto Almeida de. IV. Trubiene, Calvin Souto. V. Silva, João Pedro Vilas Boas. VI. Marques, Jorge André Gastmaier.

22-102518 CDD-004

Índices para catálogo sistemático:

1. Ciência da computação 004

Aline Graziele Benitez - Bibliotecária - CRB-1/3129

Sumário

OBSERVAÇÃO INICIAL

Olimpíada Brasileira de Satélites MCTI (OBSAT)
OBSAT

A Olimpíada Brasileira de Satélites MCTI é uma Olimpíada Científica de abrangência nacional, concebida pelo Ministério da Ciência, Tecnologia e Inovações, e organizada pela Universidade Federal de São Carlos (UFSCar) com apoio e parceria da Agência Espacial Brasileira (AEB/MCTI), do Instituto Nacional de Pesquisas Espaciais (INPE/MCTI) e da Escola de Engenharia de São Carlos (EESC), da Universidade de São Paulo (USP). As olimpíadas científicas são iniciativas para promover a popularização e difusão da ciência e tecnologia junto aos estudantes Brasileiros, além de despertar o interesse por carreiras na área de ciência e tecnologia de forma atrativa, e sempre que possível, prática.

A Olimpíada Brasileira de Satélites MCTI tem por objetivo promover experiências teóricas e práticas em projetos de satélites de pequeno porte, difundindo a cultura aeroespacial para estudantes e professores de instituições de ensino de nível médio, técnico profissionalizante, e universitários. A OBSAT é gratuita para qualquer aluno matriculado em instituições brasileiras de ensino fundamental, médio, técnico ou superior. Como objeto de trabalho, e ao mesmo tempo ferramenta de aprendizado, utilizam-se pequenos satélites, chamados de CanSats, TubeSats, PocketSats ou CubeSats.

PION LABS
PION Labs

A PION é uma startup espacial brasileira localizada em São Paulo que lançou o primeiro satélite desenvolvido totalmente por uma empresa do Brasil, o PION-BR1. A PION acredita que conectar sensores e coletar dados para um monitoramento mais inteligente, rápido e de baixo custo, é a forma de tornar o planeta mais sustentável com aumento de produtividade e eficiência. Além de tecnologias espaciais para uma sociedade do futuro, a PION também promove o fomento de recursos humanos por meio de produtos e serviços educacionais de forma pró-ativa que, desde 2019, já contribuiu com a formação de milhares de estudantes da América Latina! O picosatélite PION-BR1 é o demonstrador de tecnologia da PION de baixa potência como plataforma educacional. A PION possui o propósito de tornar o espaço acessível para todos.

Kits PION - OBSAT

Durante a realização da primeira edição da OBSAT MCTI, foi realizado um processo de compra pública através do pregão eletrônico No. 61/2020, com a participação de várias empresas. A PION foi a ganhadora deste processo seletivo, tornando-se a empresa responsável por fornecer cerca de 250 kits CanSat e CubeSat de arquitetura aberta para participantes da primeira Olimpíada Brasileira de Satélites MCTI.

BIPES: *Block based Integrated Platform for Embedded Systems* BIPES

BIPES: *Block based Integrated Platform for Embedded Systems* (BIPES) é uma plataforma totalmente aberta e livre, desenvolvida no Brasil, para programação de sistemas embarcados e aplicações de Internet das Coisas utilizando programação por blocos e suportando diversas plataformas de sistemas embarcados e Internet das Coisas. Uma das grandes vantagens do BIPES é que todo desenvolvimento pode ser feito em blocos e através de um navegador web, sem a necessidade de instalação ou configuração de softwares específicos para desenvolvimento de sistemas embarcados. O BIPES tem seu código aberto disponível na plataforma github (BIPES · GitHub), e tem o seu desenvolvimento apoiado pelo 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).

Considerando os objetivos educacionais comuns da UFSCar, da OBSAT MCTI, da PION Labs e do projeto BIPES, membros destas instituições e projetos se uniram para preparar este ebook aberto e gratuito, com a intenção de oferecer uma alternativa simples e didática para a programação de kits de satélites educacionais CanSats e CubeSats.

Agradecemos o constante apoio e incentivo do Ministério da Ciência, Tecnologia e Inovações na busca de ações efetivas de popularização da ciência, através da Secretaria de Articulação e Promoção da Ciência - SEAPC.

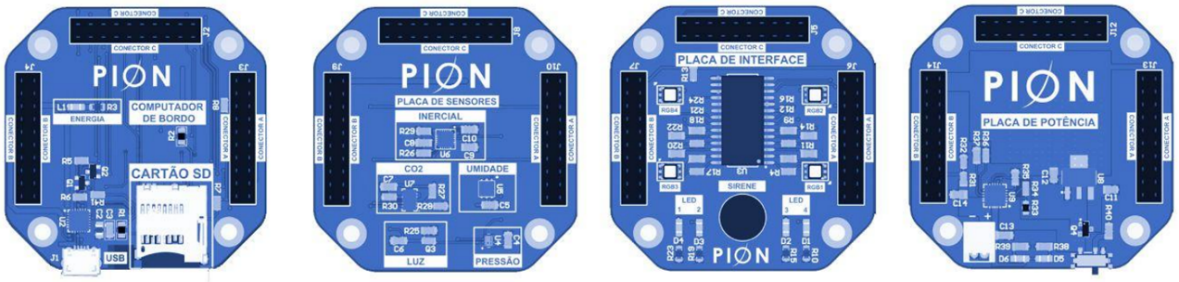
Kits disponíveis

CanSat

O CanSat PION é um satélite educacional criado pela *startup* brasileira PION. Ele é do tamanho de uma lata de refrigerante. Embora seja pequeno, o CanSat PION, utilizado pela OBSAT, nesta 1a edição, tem processador de 32 bits, e uma arquitetura modular com diversos sensores e possibilidades de uso.

Abaixo, foto de uma chave Allen, que você utilizará para desmontar e montar o kit:

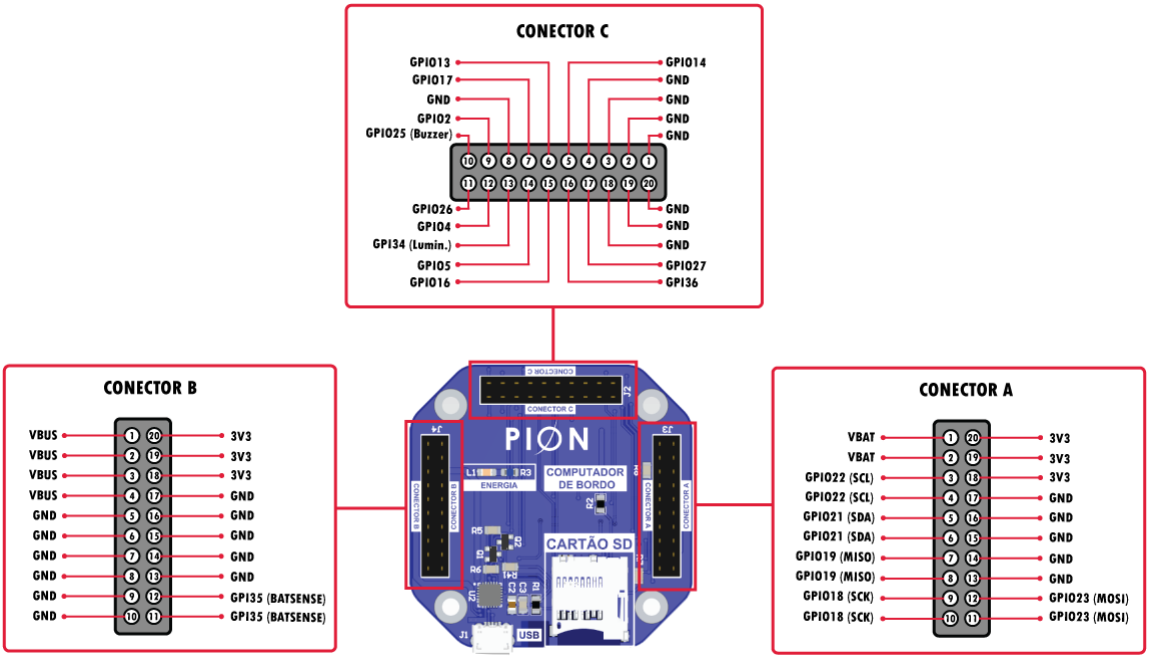
Caso queira desmontar o kit, você poderá conhecer um pouco mais sobre cada placa e sua **arquitetura modular**, tipicamente usada em missões espaciais, onde cada placa / módulo é responsável por um tipo de operação.



Computador de Bordo | Placa de Sensores | Placa de Interface | Placa de Potência

Fonte: pionlabs.com.br

Analise cada uma das placas e observe as barras de conectores, chamada de **barramento**. É o barramento que permite a interconexão dos módulos. Observe que este barramento possui diversos pinos, sendo que cada um tem uma função, conforme a figura abaixo:



Fonte: https://github.com/pion-labs/pion-educational-kits/wiki/Hardware-CanSat

Após conhecer o hardware do kit, podemos montá-lo cuidadosamente, tomando cuidado para manter as orientações e posições originais dos barramentos, e seguir para a próxima etapa da oficina.

Curiosidade: a estrutura externa deste kit foi produzida por impressão 3D. Note que estas peças não são projetadas para voos orbitais (ao redor do planeta Terra), mas com foco principal em estudo e pesquisas.

Além da barra de pinos, que oferece o barramento que interconecta todas as placas do CanSat, este pequeno satélite ainda conta com 2 outros tipos de barramentos. O I2C e o SPI.

O I2C, ou I²C (Inter-Integrated Circuit) é um barramento serial que interconecta vários dispositivos eletrônicos a um processador. No CanSat PION, diversos sensores são conectados ao módulo de processamento ESP32 através do barramento I2C. Neste sentido, cada dispositivo possui um endereço, para que ele possa ser acessado. A tabela abaixo mostra os pinos, sensores e outros componentes com seus endereços I2C no CanSat PION.

Sensor	Descrição e Faixa de Operação	Pino ou Endereço I2C
Temperatura	Sensor I2C SHT20 Faixa de operação: -40 a 125°C	0x40
Umidade relativa do ar	Sensor I2C SHT20 Faixa de operação: 0 - 100 %	0x40
Pressão atmosférica	Sensor barométrico I2C BMP280 Faixa de operação: 300 - 1100 hPa	0x76
Nível de Bateria	Leitura direta de uma entrada analógica com o conversor analógico-digital (ADC)	GPIO35
Luminosidade	Sensor analógico ALS-PT19-315C Leitura direta de uma entrada analógica com o conversor analógico-digital (ADC) Faixa de operação: 0 - 100 %	GPIO34
CO_2	Sensor I2C CCS811 400 - 29206 ppm	0x5A
Giroscópio	Módulo I2C MPU-9250 Faixa de operação: ±250, ±500, ±1000, e ±2000°/seg	0x68
Acelerômetro	Módulo I2C MPU-9250 Faixa de operação: ±2g, ±4g, ±8g e ±16g	0x68
Magnetômetro	Módulo I2C MPU-9250 Faixa de operação: ±4800 μT	0x68
I/O Exapander - LEDs	MCP23017 com 16 saídas ligadas à LEDs RGB espalhados na placa de interface	0x20
SD Card	Slot para SD Card	MOSI: 23 MISO: 19 SCK: 18 ChipSelect: 15

Outro tipo de barramento popular em sistemas embarcados é o SPI (*Serial Peripheral Interface*). Neste kit, ele é usado para a conexão do cartão de memória SD Card através dos pinos MOSI, MISO, SCK e Chip Select (CS ou SS), conforme última linha da tabela. Tal barramento pode também ser utilizado para a conexão de outros sensores e dispositivos que utilizam este protocolo.

CubeSat

O CubeSat tem o formato de um cubo de 100x100x100mm e já é usado amplamente em missões espaciais reais, que podem ser também um pouco maiores. O padrão de 100x100x100mm é definido como 1U, e versões maiores também existem, como 200x100x100mm (2U) ou 300x100x100mm (3U). O CubeSat PION possui arquitetura eletrônica interna similar ao CanSat, porém com uma montagem diferenciada.

Placas:

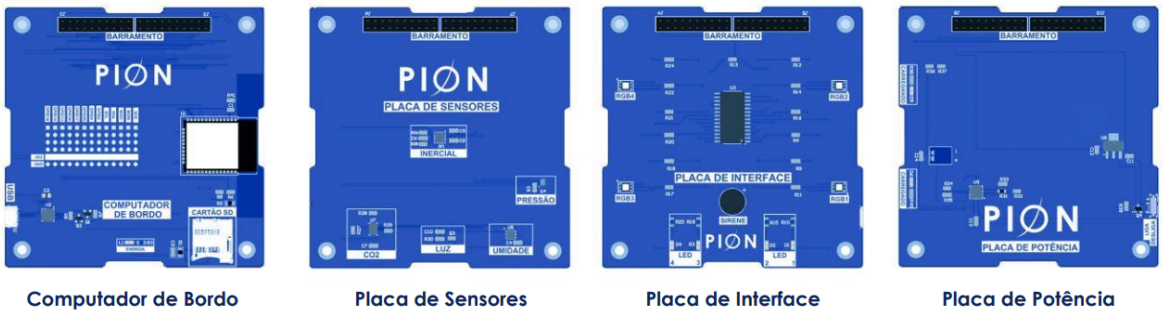
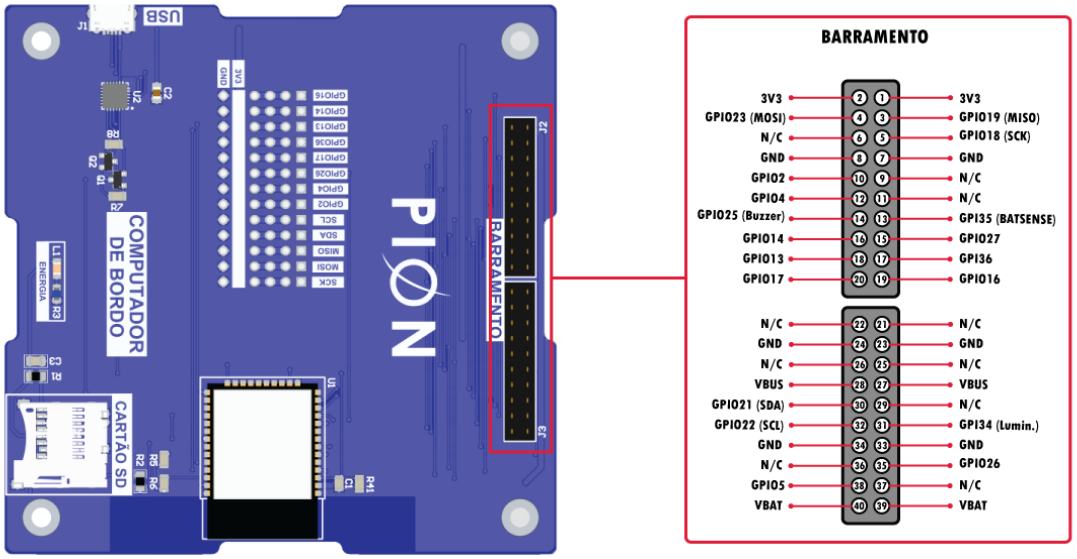


Figura: pionlabs.com.br

A próxima figura mostra o mapa de pinos do CubeSat PION:



Mapa de pinos do CubeSat. Fonte:
https://github.com/pion-labs/pion-educational-kits/wiki/Hardware-CanSat

Maiores detalhes sobre os kits podem ser obtidos no github:
https://github.com/pion-labs/pion-educational-kits/wiki/Hardware-CanSat

Nota: Os sensores e pinos do CubeSat são os mesmos / compatíveis com os dos CanSats, de forma que as informações mencionadas para os CanSats também valem para CubeSats.

Foto cedida pela PION Labs.

Utilizando o BIPES

Preparação do kit

BIPES: Block based Integrated Platform for Embedded Systems (BIPES) é uma plataforma totalmente aberta e livre, desenvolvida no Brasil, para programação de sistemas embarcados e aplicações de Internet das Coisas (IoT).

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.

Como você deve imaginar, o satélite é um **sistema embarcado**. No caso dos CanSats e CubeSats da PION, é um sistema embarcado baseado no módulo de processamento ESP32 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. Se você pensar bem, eles são muito mais poderosos que o computador que levou o módulo lunar da missão Apollo à superfície da Lua.

Para utilizar o BIPES, os módulos ESP32 devem possuir o sistema MicroPython instalado (MicroPython), que possibilita a programação dos módulos ESP32 em linguagem Python. O MicroPython permite o uso de Python em processadores embarcados, como ESP32, ARM, dentre outros. Dessa forma, o primeiro passo é instalar o MicroPython no módulo ESP32.

Vale lembrar que já existem relatos do uso do módulo ESP32 em aplicações espaciais, como o MORABA (*Mobile Rocket Base*) da Agência Espacial Alemã (DLR) (ESP32 Launched Into Space | Espressif Systems).

Este processo só precisa ser feito uma vez e todos os programas anteriores gravados no kit serão perdidos.

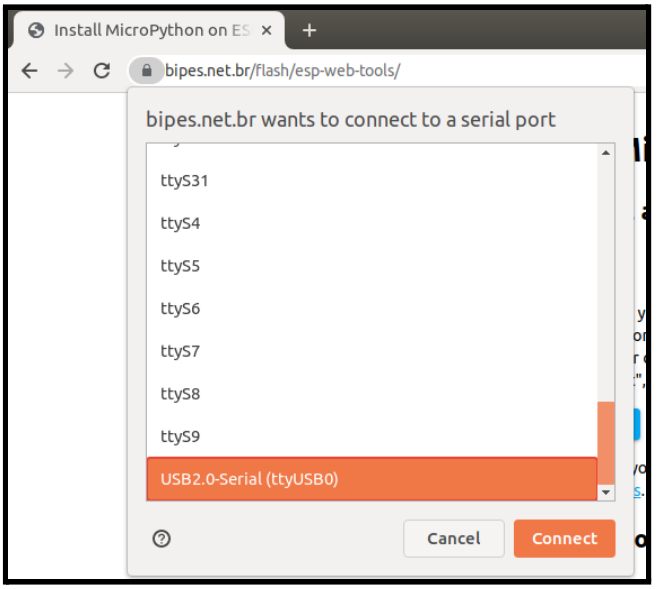
Para instalar o MicroPython no ESP32 do CanSat ou CubeSat:

1. Acesse: [Install MicroPython on ESP32 or ESP8266 to use with BIPES](#).

2. Conecte o kit à porta USB do PC e ligue o dispositivo.

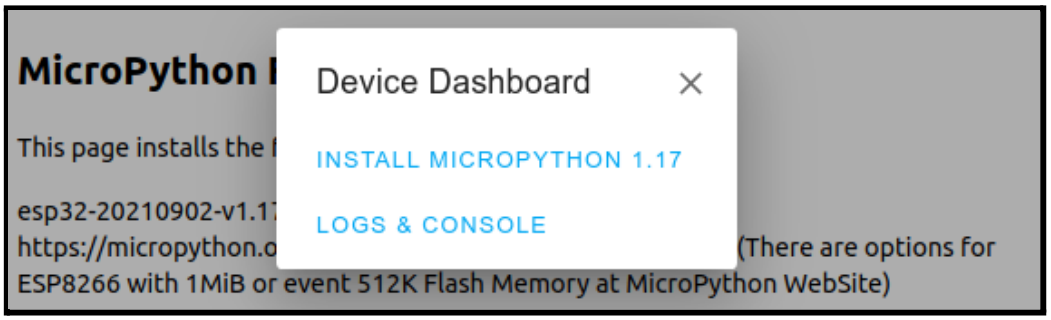
3. Clique em **CONNECT**.

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

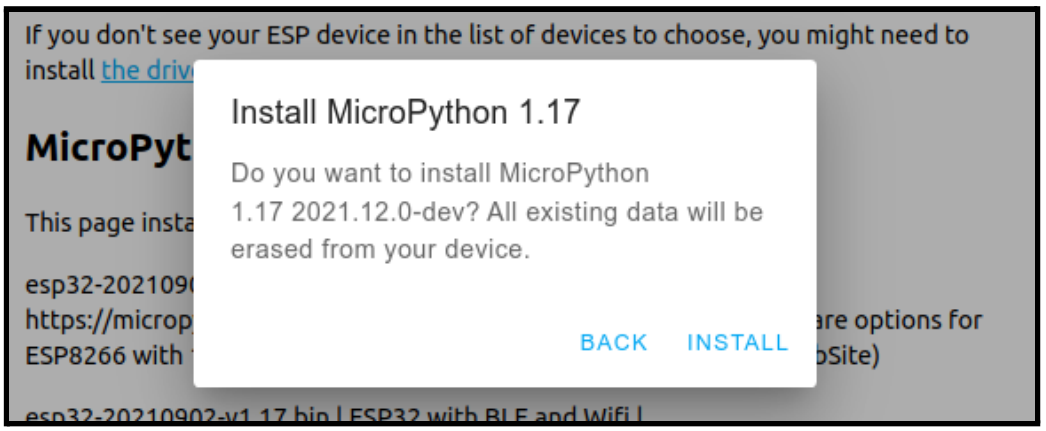


Obs.: Caso não apareça a porta serial do seu dispositivo, desça a página e baixe os drivers indicados.

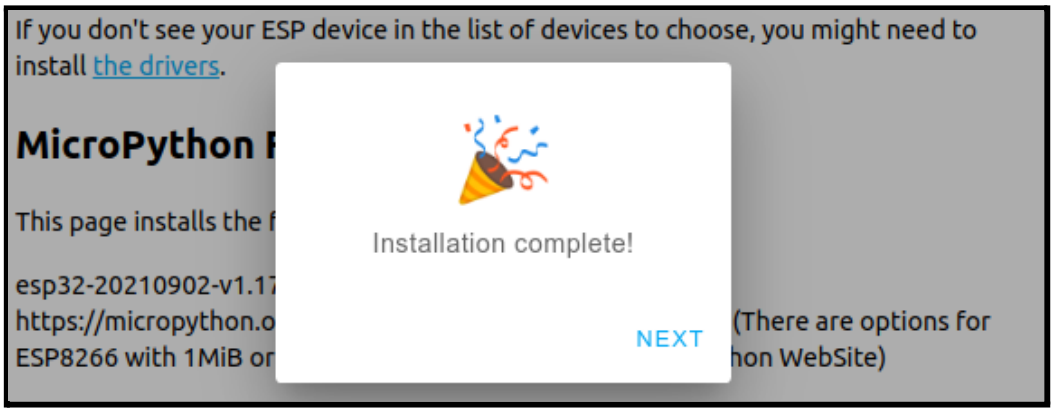
5. Em seguida, clique em **INSTALL MICROPYTHON**:



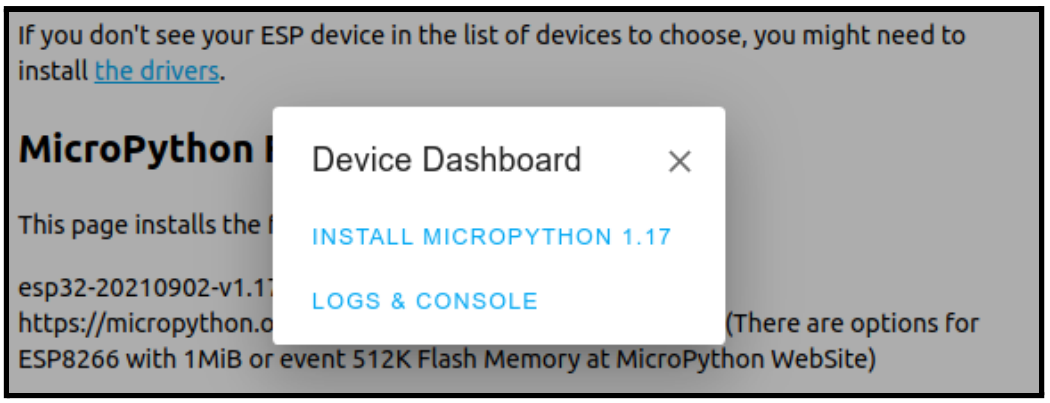
6. Depois clique em **INSTALL**:



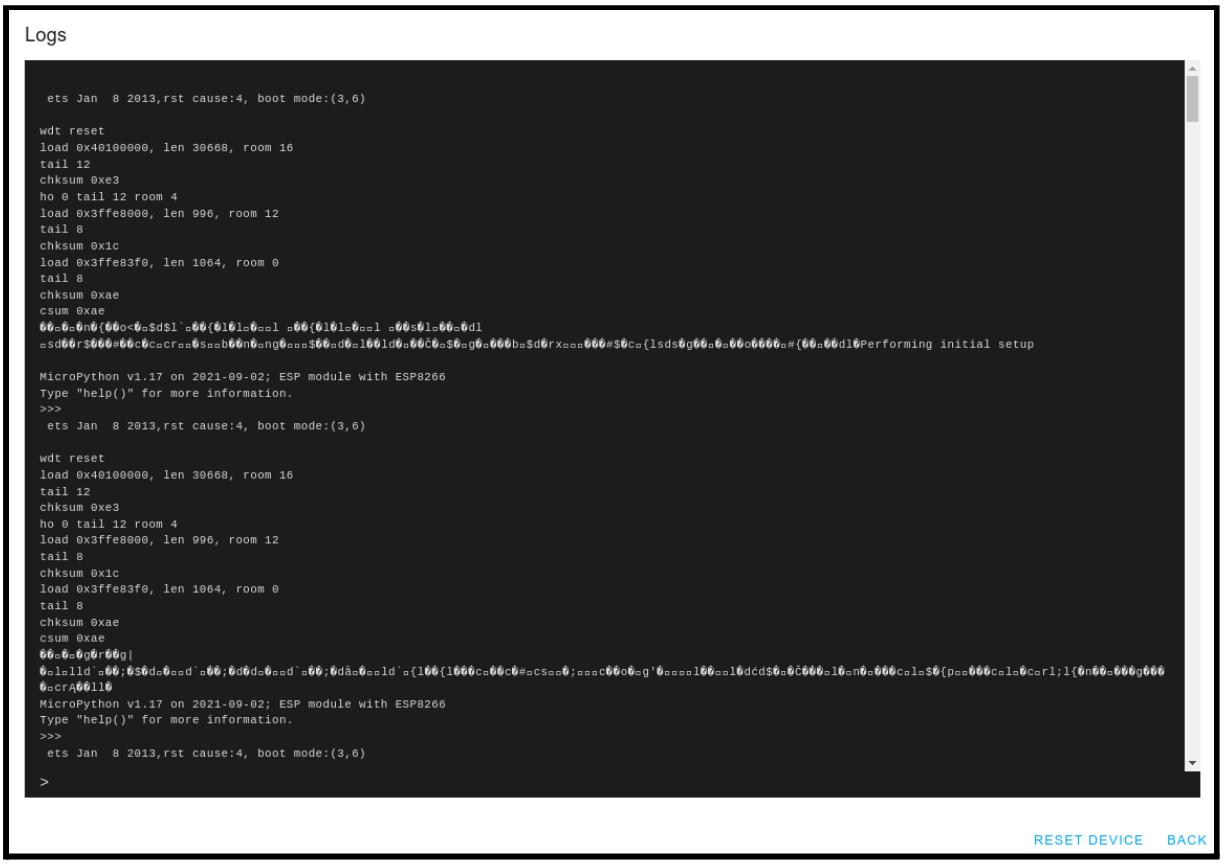
7. Aguarde o processo de instalação do MicroPython. Após isso, você verá a mensagem:



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



Observe o comportamento do sistema pelo console. Por padrão, o MicroPython cria uma rede “MicroPython”, para conexão e programação remota, via Wi-fi. Entretanto, eventualmente, algumas fontes de alimentação podem acabar fornecendo corrente elétrica insuficiente para suprir sua demanda de energia quando o módulo ESP32 inicia o serviço de rede (*Access Point*), e a placa passa a reiniciar sem parar (conforme imagem a seguir).



Dependendo da sua fonte / porta USB, este problema não vai ocorrer. De qualquer forma, caso ele ocorra, a solução é desativar o modo “*Access Point*” do módulo. Para fazer isso, copie a linha abaixo, e cole no terminal / console imediatamente após a placa reiniciar.

Você vai precisar observar o momento do *reboot* e digitar o comando imediatamente logo após o *reboot*. Caso contrário, a placa continuará reiniciando continuamente.

Comando a ser copiado e colado imediatamente após o *reboot* da placa:

import network; network.WLAN(network.AP_IF).active(False)

Com o MicroPython instalado e estável, você pode seguir para as próximas etapas.

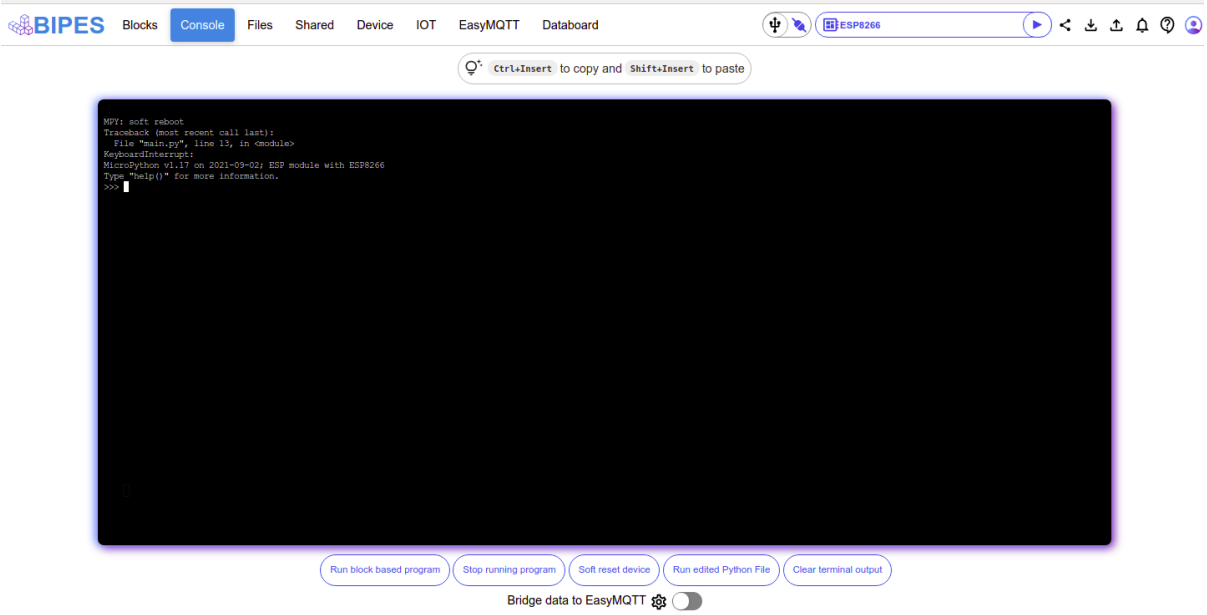
Primeiro programa

1. Conecte o kit na porta USB do PC e ligue o kit.

2. Acesse o ambiente de desenvolvimento do **BIPES**: [BIPES Project].

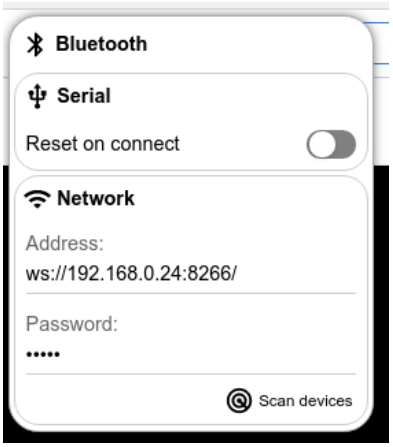
3. Escolha, no canto superior direito, a placa **PION CanSat (ESP32)** ou **PION CubeSat (ESP32)**:

4. Acesse a aba **Console**:

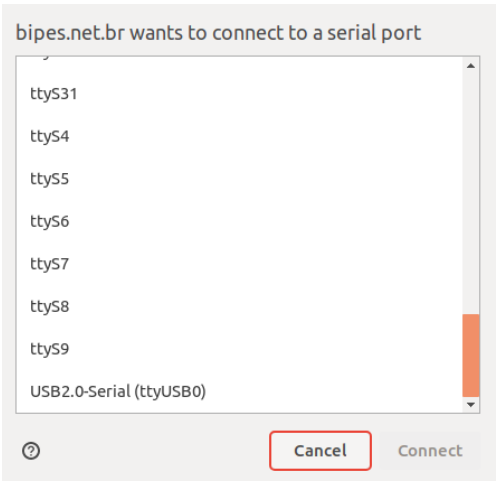


5. Clique no ícone para conectar: 

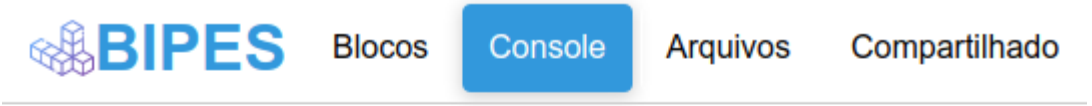
6. Clique em **Serial**:

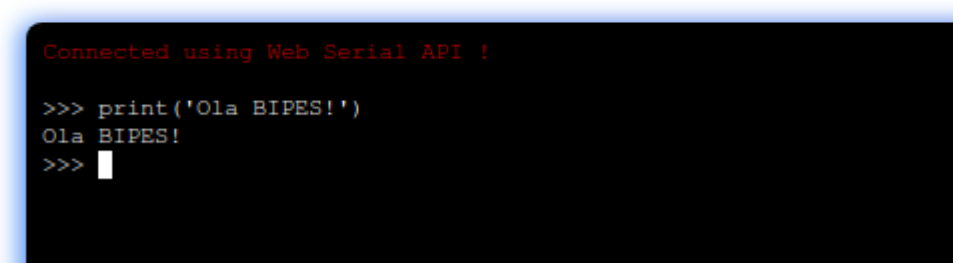


7. Escolha uma porta disponível e clique em **Connect**:



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).





9. Prepare o seguinte programa (na aba **Blocks / Blocos**) pelo **BIPES** (em inglês ou português). Clique em **Loops / Laços** e arraste o bloco "**repeat / repetir**". Após isso, clique em **Text / Texto** e arraste o bloco "**print / imprime**". Por fim, arraste o "**bloco com aspas**", também na aba **Text / Texto**.

Inglês:

Português:

Nota: Ao longo deste texto, para fins didáticos e de proporcionar maior familiriade com o BIPES, alguns exemplos são apresentados com blocos em inglês e outros com blocos em português.

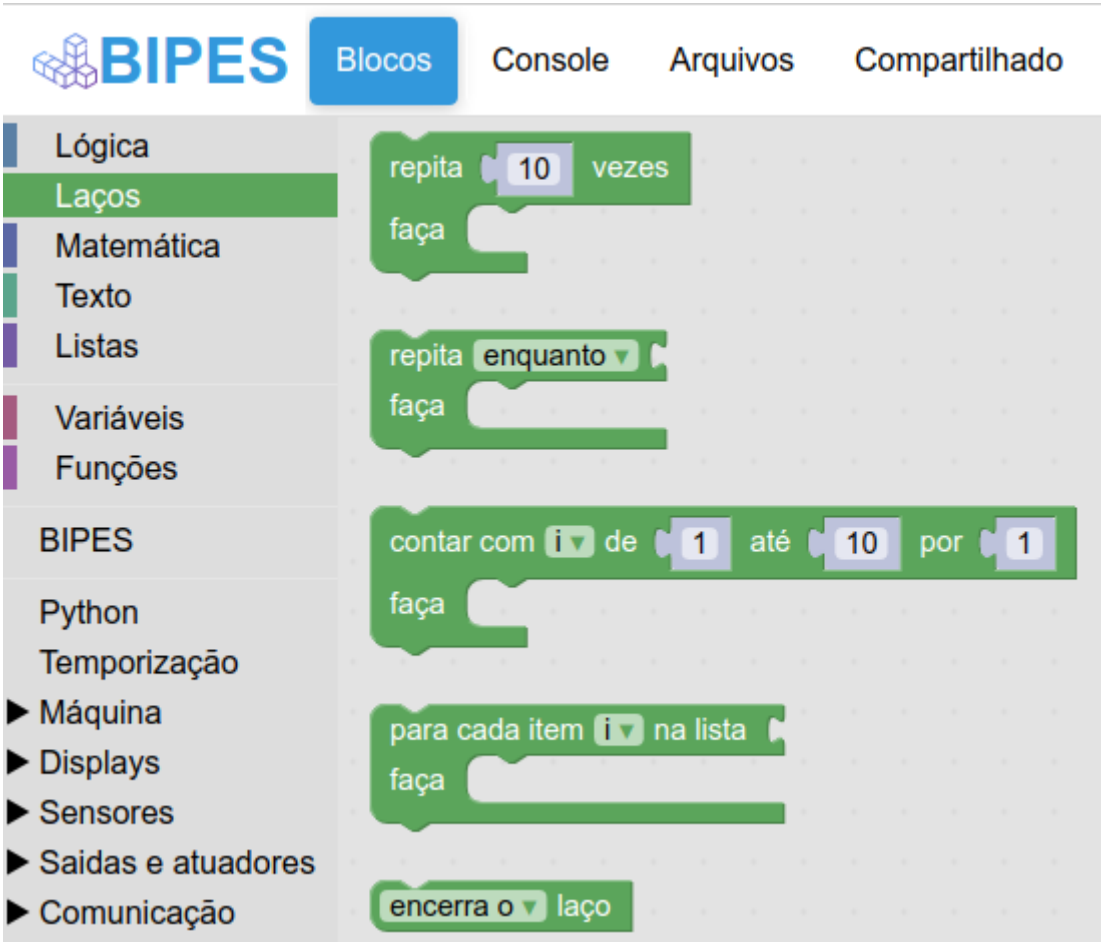
10. Após preparar o programa, clique no ícone para executar (símbolo **play**):

Verifique, pela aba console, o envio do programa para a placa. O programa irá exibir a mensagem especificada em "print / imprime" dez vezes.

Atividade: Mude o programa para que a mensagem seja exibida a cada 3 segundos. Caso tenha dúvidas, peça ajuda a comunidade da OBSAT no Discord.

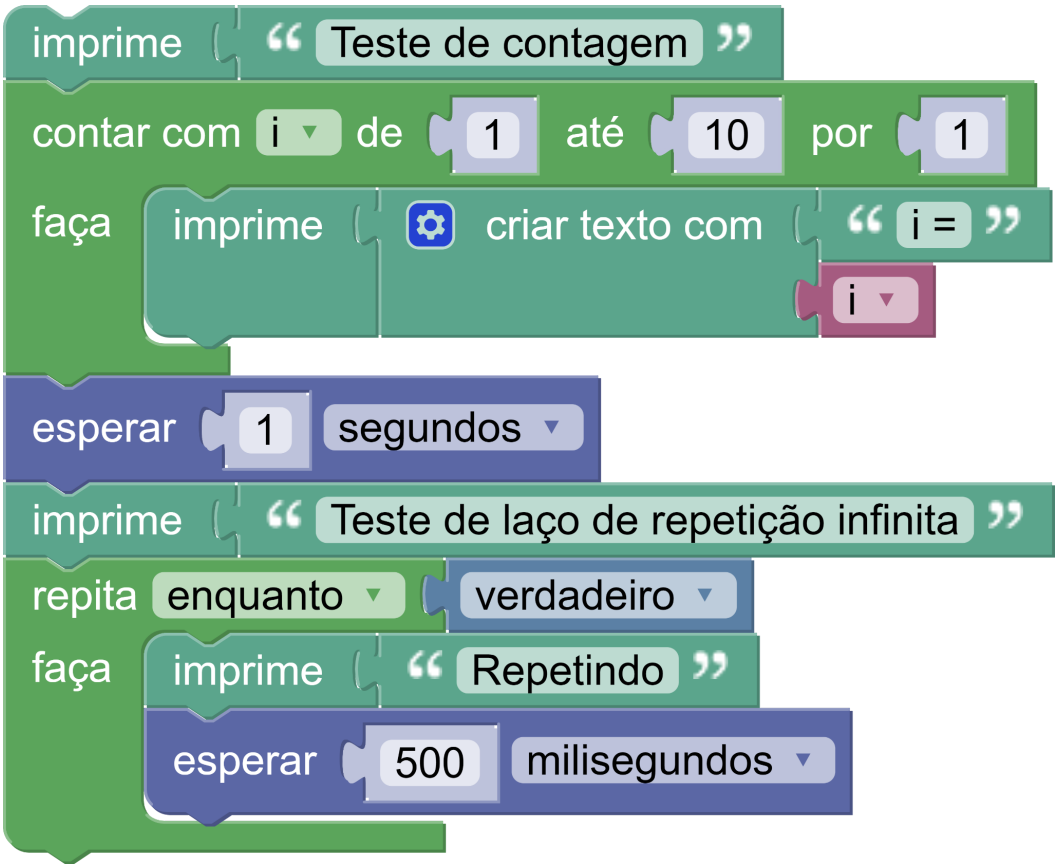
Note que utilizamos o bloco "**repeat 10 times**" / "**repetir 10 vezes**", disponível na guia lateral **Loops / Laços** à esquerda dos blocos **BIPES** para piscar enviar a mensagem para o console. Esta mesma guia possui outras opções de estruturas de repetição, como "**count with i from 1 to 10 by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

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

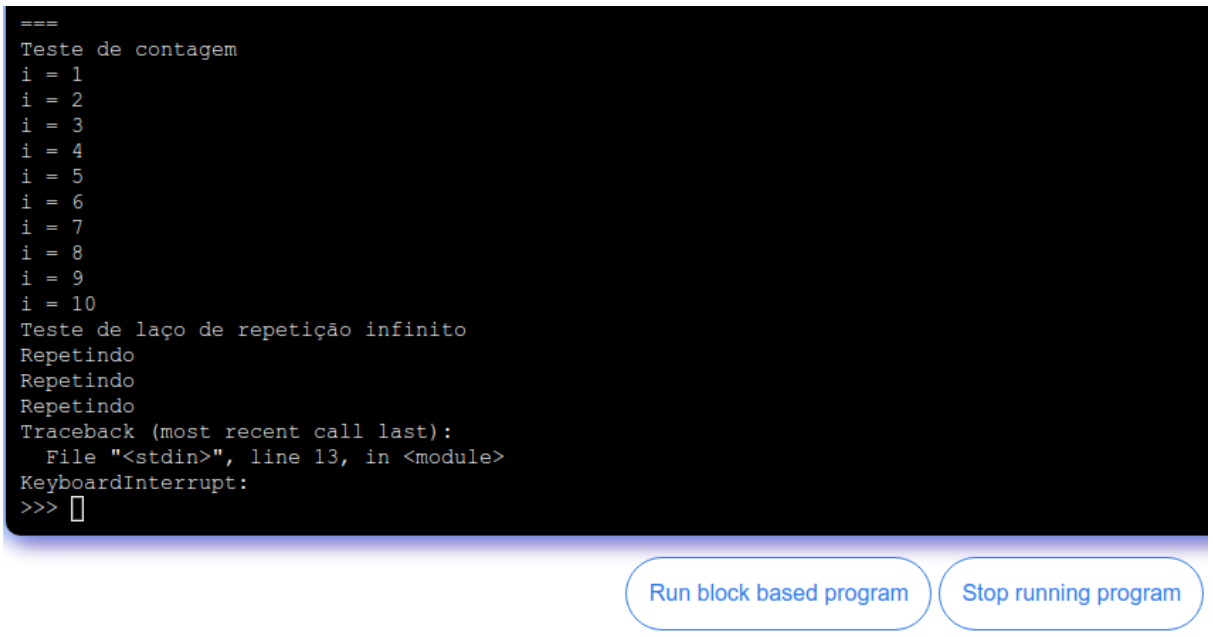


No caso de contar de 1 a 10, cada iteração do laço atualiza a variável *i* (um espaço de armazenamento de dados do programa), que pode ser usada na estrutura de repetição. No exemplo abaixo, o programa imprime no console de 1 a 10 com um texto inicial "**i =** " para mostrar o nome da variável. Já a 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" ao 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:

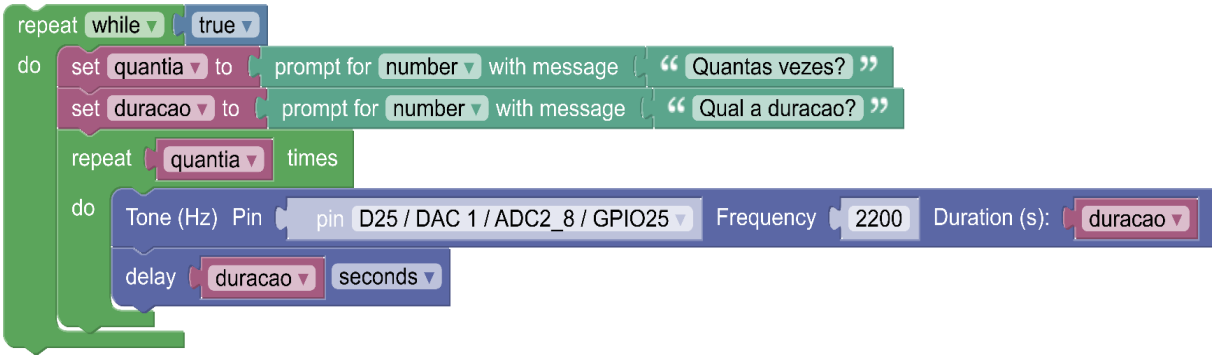


O resultado obtido no **Console** é o seguinte (clique em Stop para o laço infinito ser interrompido):

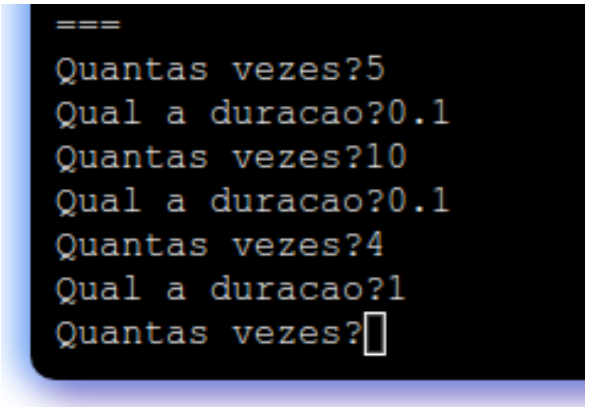


Outro recurso útil associado ao **Console**, é a possibilidade de solicitar dados do usuário de forma interativa. O bloco "**Pede um número" (*prompt for number)*** pode ser utilizado para esta finalidade, podendo ser encontrado na área **Texto** da barra de ferramentas. Dessa forma, o programa abaixo apresenta uma versão interativa de um programa que usa o BIPES para emitir BIPES sonoros! Para tanto, lembramos que há um emissor de sons (*buzzer)* conectado ao pino GPIO25 do módulo de processamento. Assim, o bloco ***Tone*** pode ser utilizado para emitir frequências sonoras de várias frequências e durações.

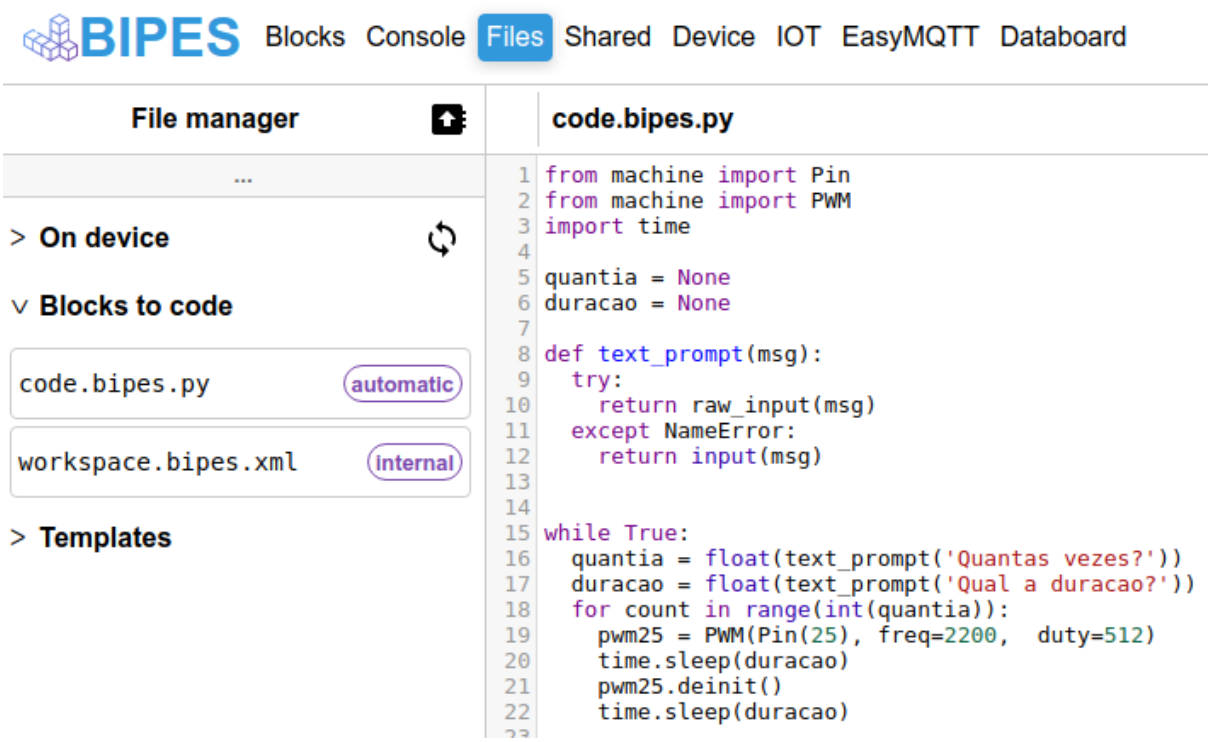
No programa abaixo, duas variáveis (quantia e duração) são definidas e o usuário deve informar os valores desejados através do **Console.** Em seguida a estrutura de repetição ***repeat*** controla a emissão de som pela quantia de vezes solicitada pelo usuário, com a duração também solicitada pelo usuário.



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



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**:



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, o módulo ESP32 do CubeSat ou CanSat 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**. Para isso, siga os seguintes passos:

1. Copie o seu código;
2. Clique em "main.py", na aba templates;
3. Exclua o código dessa página (é apenas um exemplo) e cole o seu código;
4. Clique em Save / Salvar.

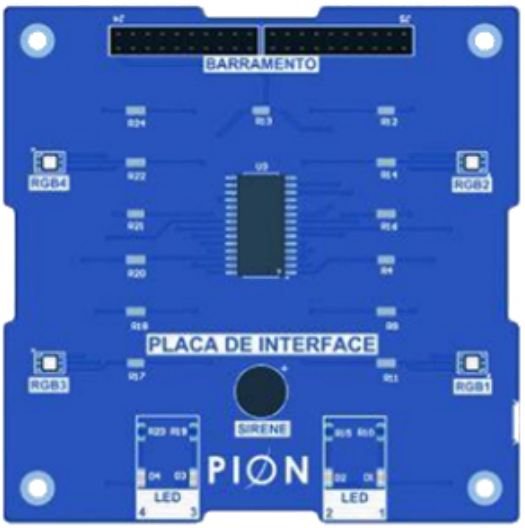
Salvo com este nome, o programa será executado automaticamente toda vez que a placa for ligada, de forma autônoma e independente. Ou seja, sua solução ficará salva e embarcada no computador de bordo de seu satélite e não dependerá de um computador para funcionar! Caso queira testar, basta salvar o programa, desconectar do PC e ligar o kit, que executará este programa utilizando a bateria interna como fonte de alimentação.

A seguir, serão apresentadas bibliotecas, blocos e dicas para uso dos sensores e outros módulos do kit CanSat PION ou CubeSat PION.

Portas de saída: LEDs da placa de *interface*

Placa de Interface

CanSat



Placa de Interface

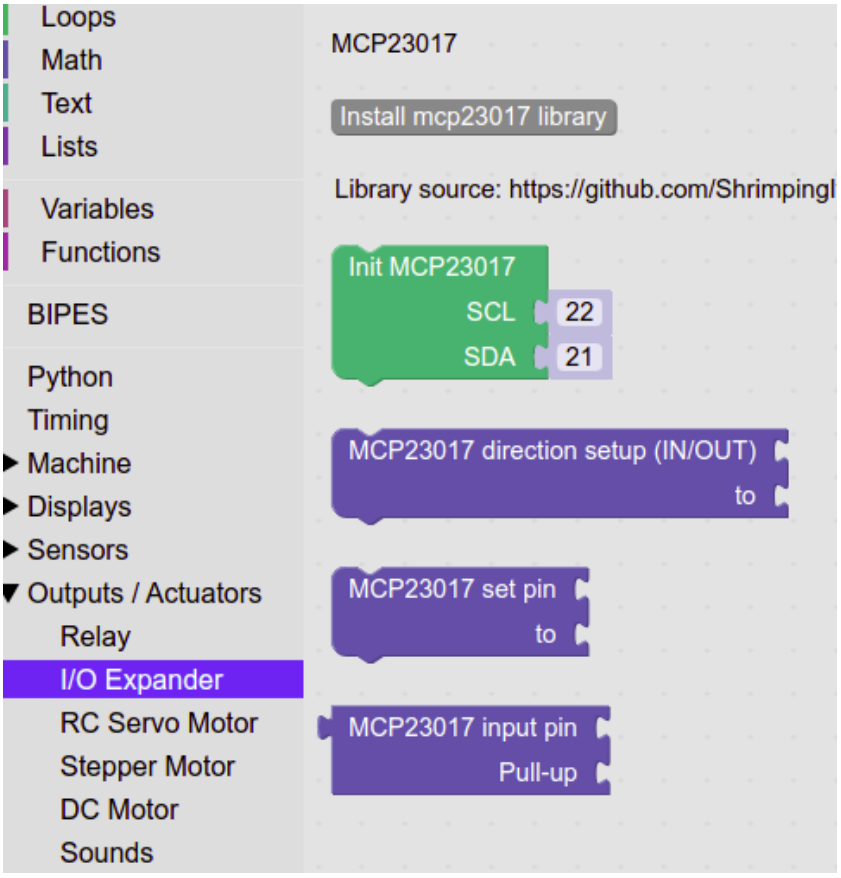
CubeSat

A placa de interface dos CanSats possui diversos LEDs coloridos (RGB). Os LEDs estão ligados a um expansor de pinos de saída, o MCP23017, que permite multiplicar os pinos de saída de um microcontrolador. Assim, o ESP32 envia comandos pelo barramento I2C para o que o MCP23017 controle os LEDs. São vários LEDs RGB acionados por pinos de 0 a 15, sendo que cada um aciona uma cor de um LED.

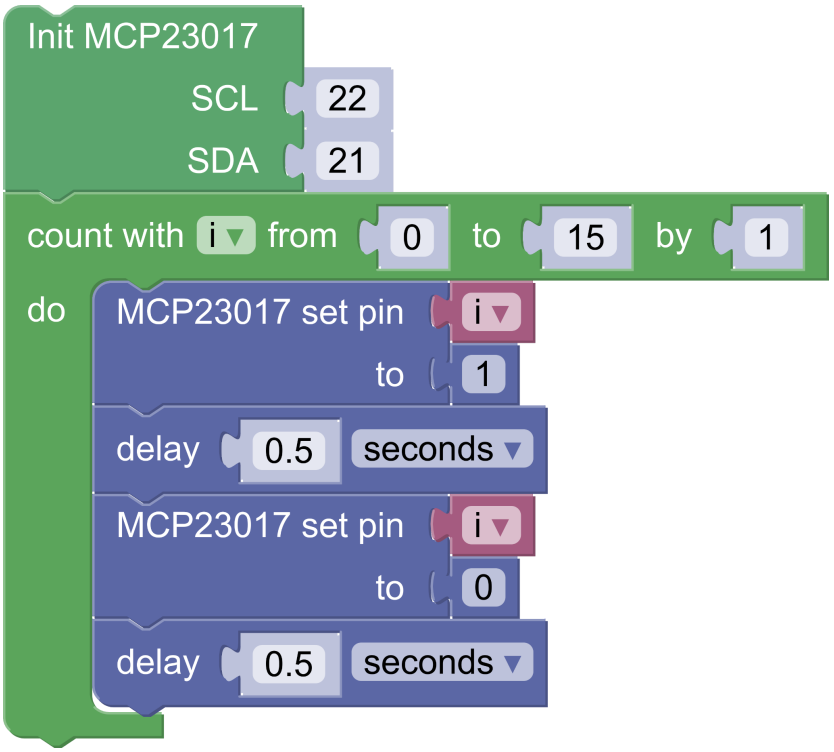
Para o uso do MCP23017, é necessário instalar uma biblioteca no computador de bordo do satélite (módulo ESP32). Para tanto, acesse o menu lateral, na opção Outputs → I/O Expander e clique em “Install mcp23017” library. Este procedimento só precisa ser feito uma vez, e caso você tenha curiosidade, poderá conhecer e visualizar a biblioteca na aba Arquivos / Files.

Atividade: utilize o programa da próxima página e identifique o número da porta de saída, de 0 a 15, associado a cada cor de cada LED RGB.

Instalação da biblioteca:



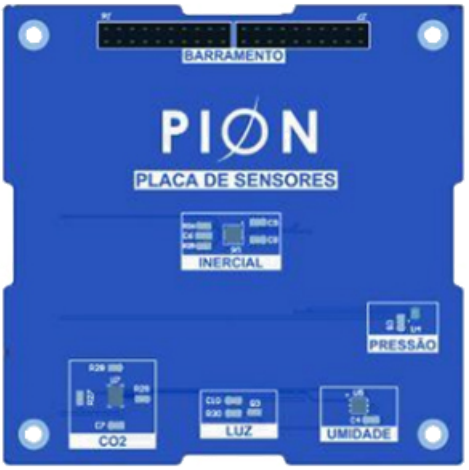
Exemplo de programa de controle dos LEDs:



Temperatura e Umidade

Placa de Sensores

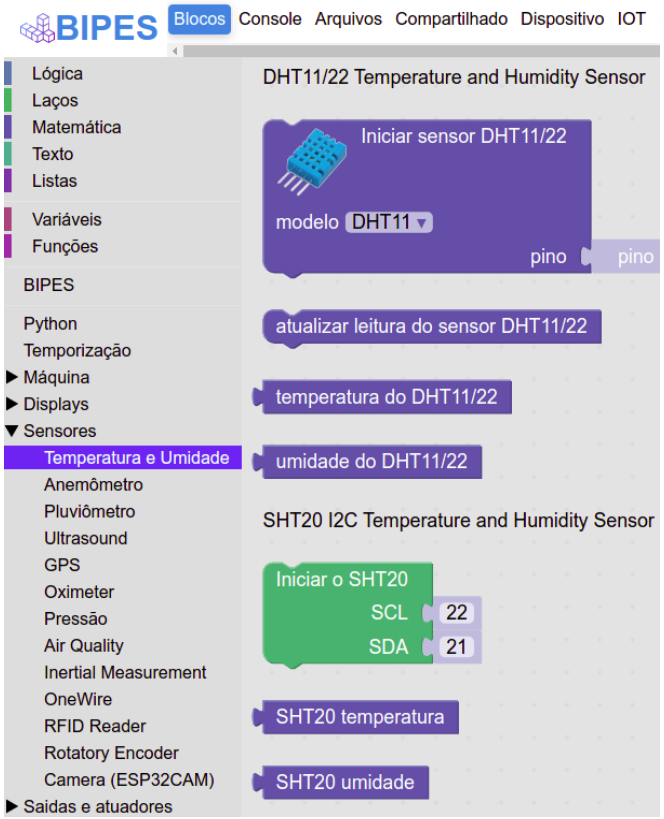
CanSat



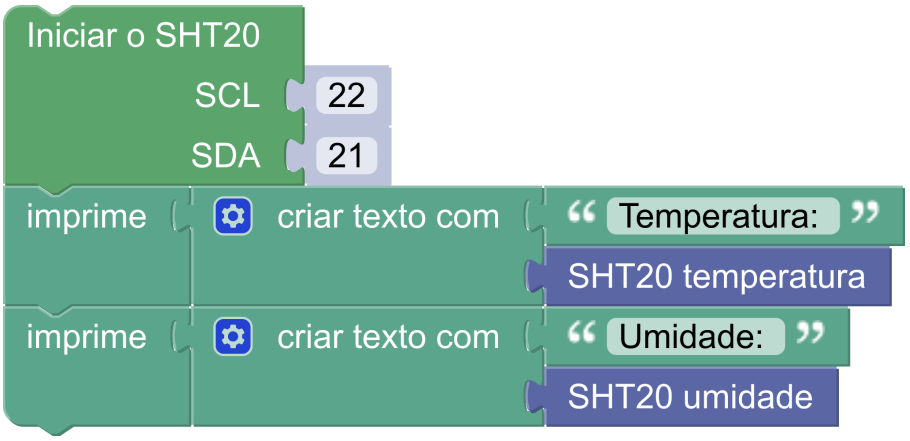
Placa de Sensores

CubeSat

A placa de sensores possui um sensor digital de temperatura e umidade conectado ao barramento I2C: o SHT20. O seguinte bloco possibilita seu uso (os 3 blocos na parte de baixo da figura):

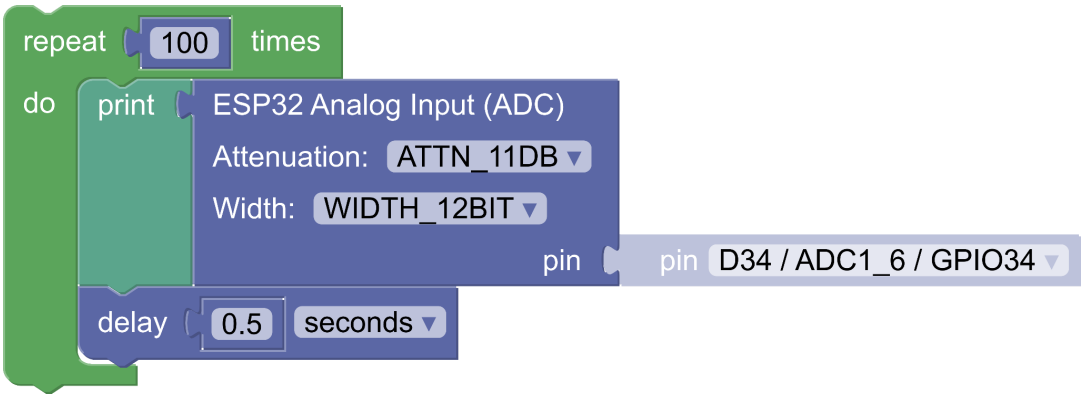


Exemplo de programa:

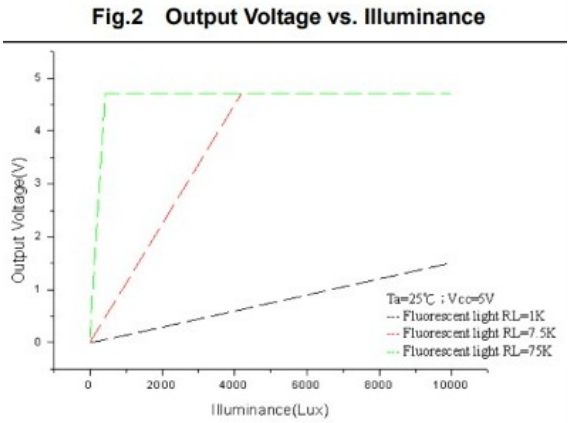


Sensor de luz (intensidade luminosa)

A placa de interface também possui um sensor de luz. O sensor de iluminação (ALS-PT19-315C) pode ser usado diretamente através da leitura do pino analógico ADC1_6 (GPIO34) do módulo ESP32:



Curva de calibração do sensor:

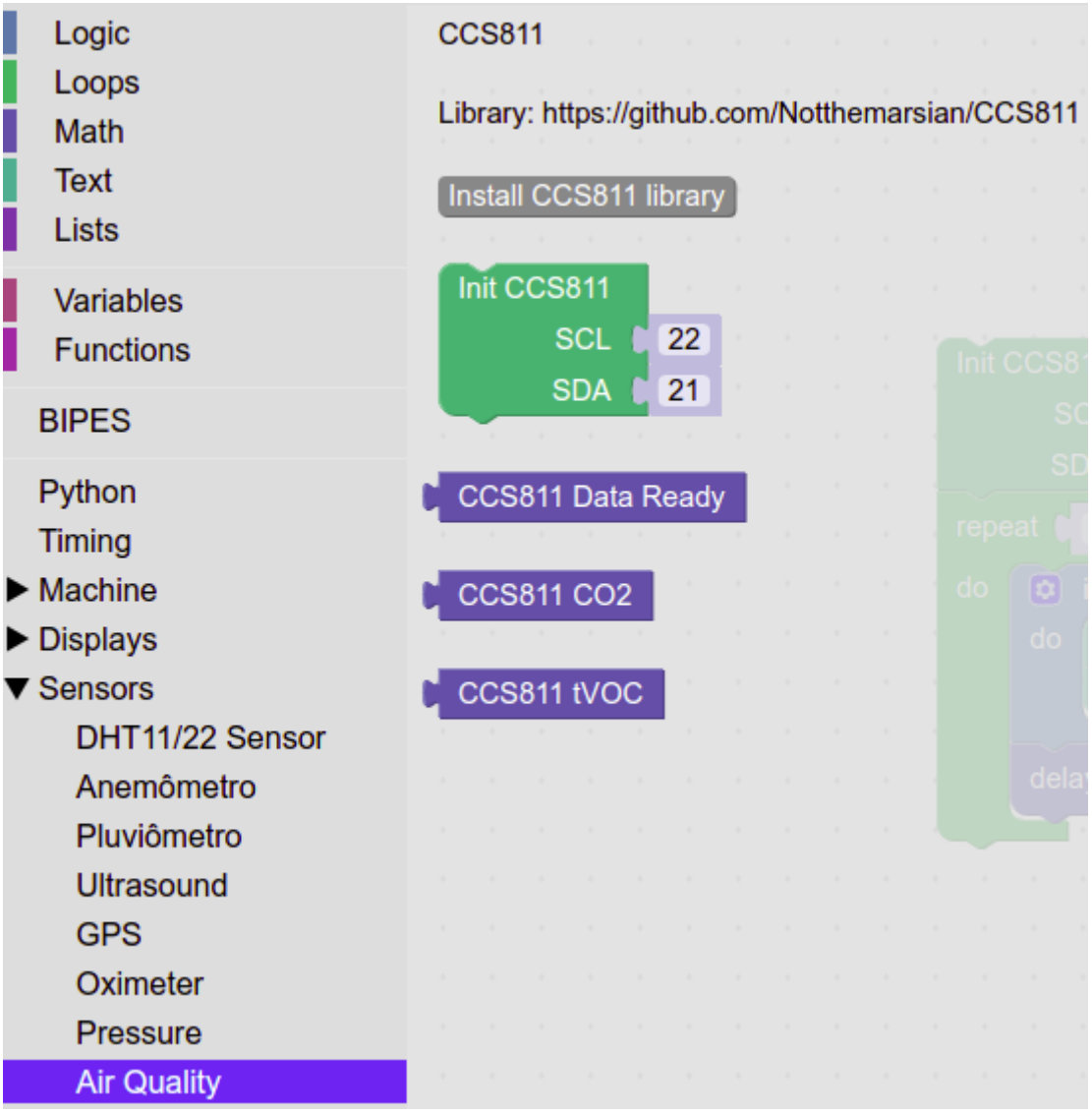


Para testar este programa, localize o sensor na placa de interface, e utilize uma lanterna apontada para o sensor. Observe a variação da leitura pelo console.

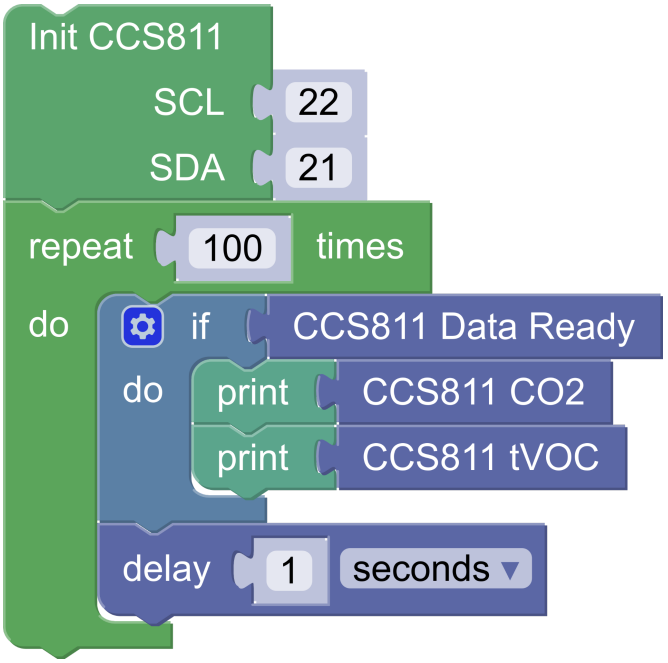
Sensor de Qualidade do Ar

O CanSat PION possui um sensor de qualidade do ar CCS811 que mede volatile organic compounds (VOR) em ppb (partes por bilhão) e nível de CO2 em ppm (partes por milhão). Este sensor também utiliza o protocolo I2C ligado aos pinos 22 (SCL) e 21 (SDA) da ESP32 com endereço 90. Uma forma fácil de testar o sensor, é ler seus dados e aproximar um isqueiro ou fósforo da placa de sensores, com cuidado!

Primeiro, instale a biblioteca:



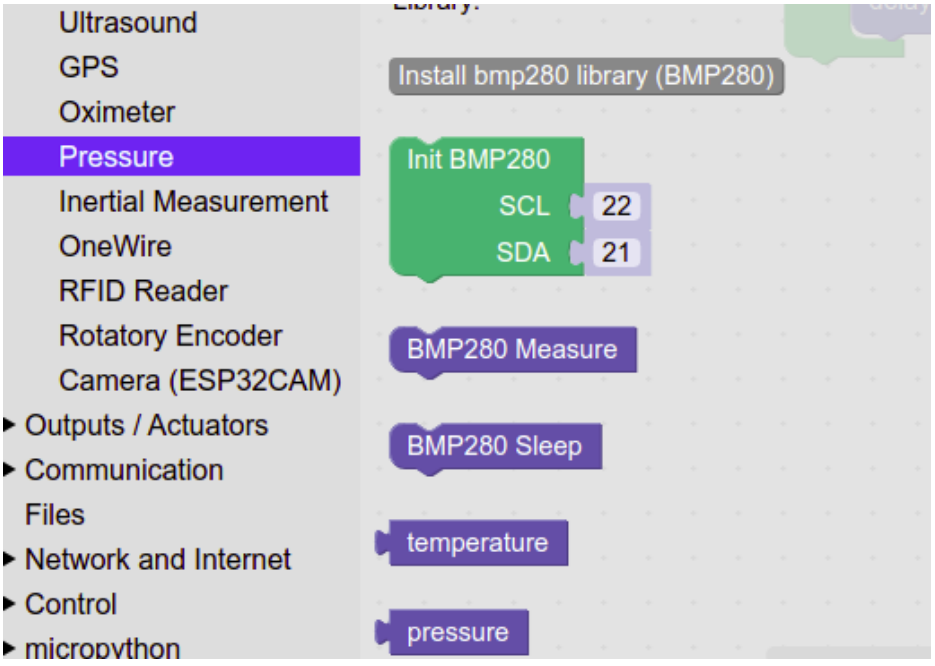
Exemplo de programa:



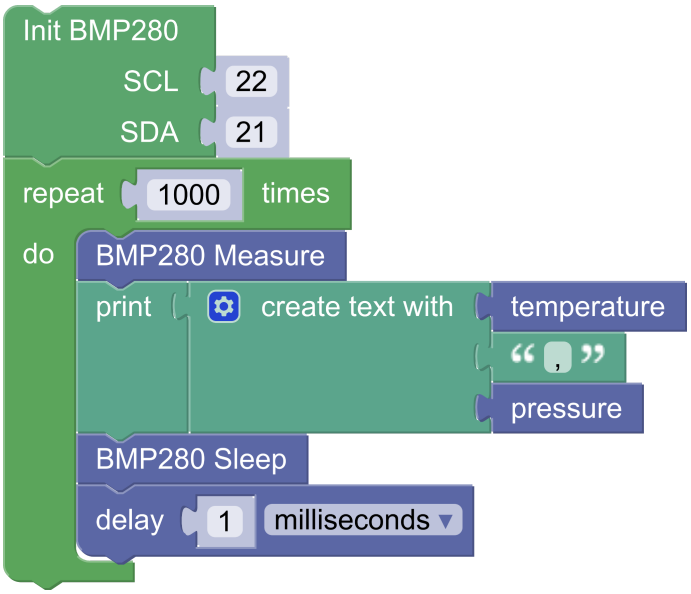
Sensor de pressão

A placa de interface também possui um sensor de pressão BMP280 que usa protocolo I2C ligado aos pinos 22 (SCL) e 21 (SDA) da ESP32.

1. Instale a biblioteca do sensor:



2. Use o sensor. Exemplo:

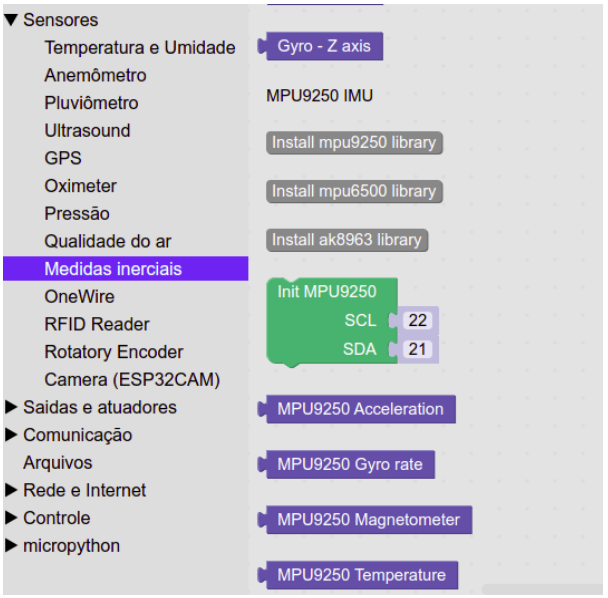


Medidas inerciais

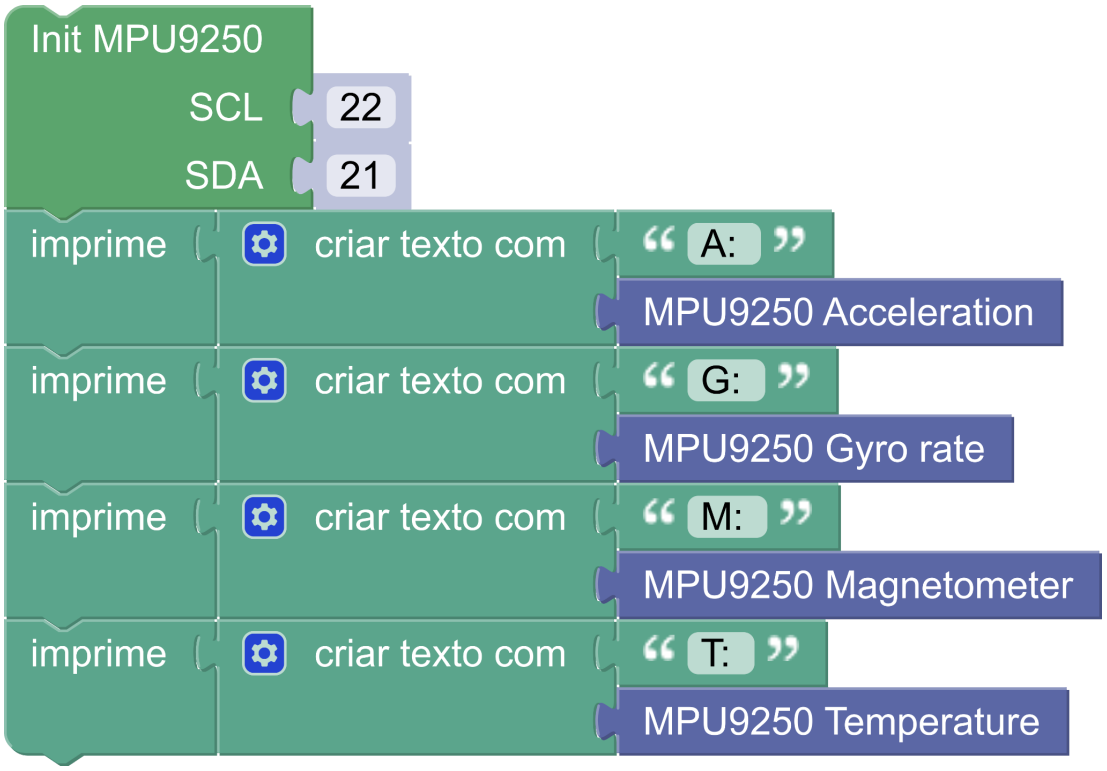
O satélite PION possui diversos sensores inerciais integrados no módulo MPU9250:

- Magnetômetro, de 3 eixos, para medir campos magnéticos, inclusive o do planeta Terra, permitindo que ele seja utilizado como bússola;
- Giroscópio de 3 eixos, para medir velocidades angulares;
- Acelerômetro de 3 eixos, para medir acelerações;

Para usar este módulo, instale as bibliotecas necessárias: **mpu9250**, **mpu6500**, e **ak8963.**



Exemplo de programa para coleta de dados inerciais:



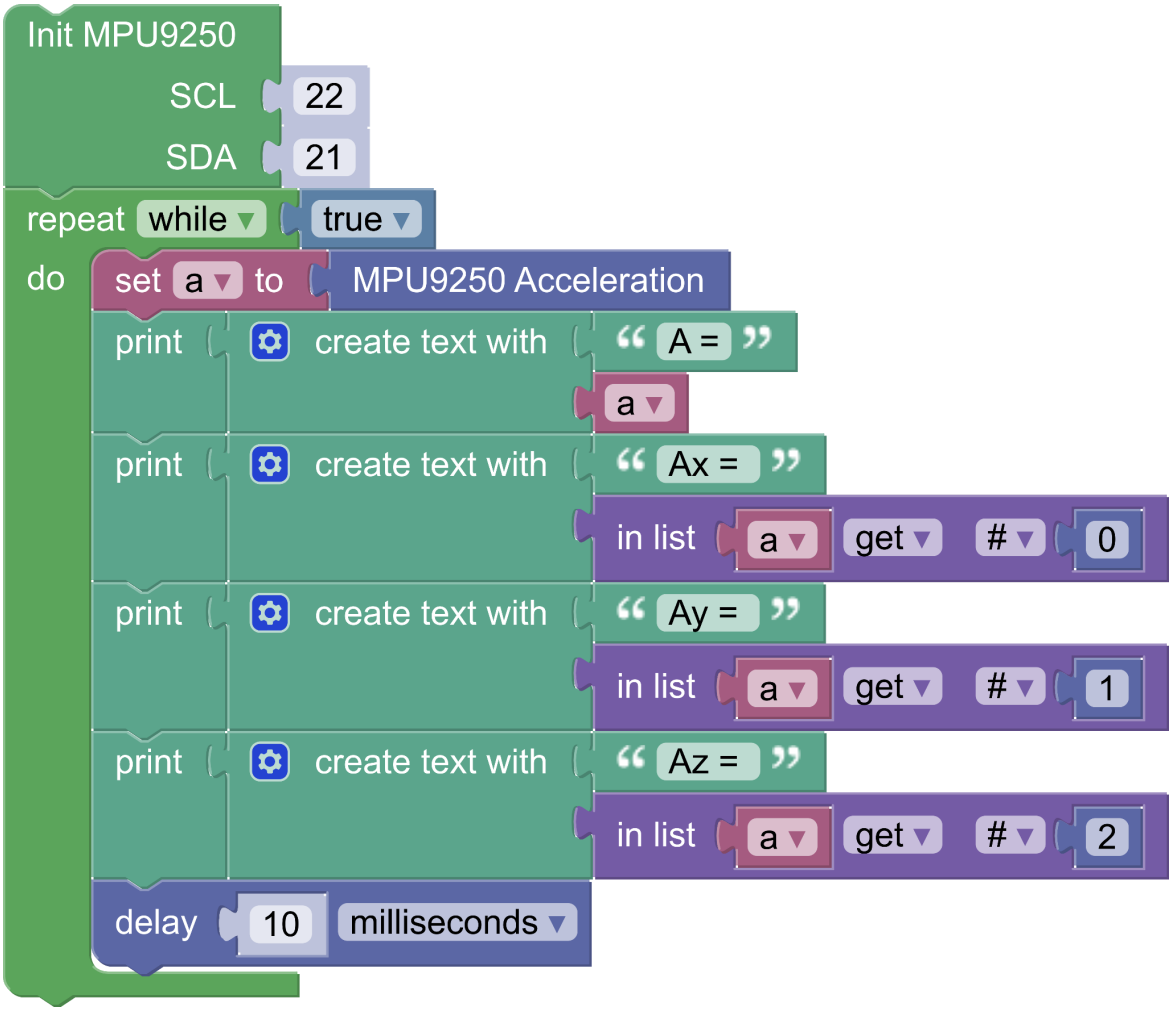
Ao executar o programa, note que o resultado é dado na forma de vetores:

```
A: (0.05746084, 0.1412579, 9.540894)
G: (0.03677183, -0.01385605, 0.007061257)
M: (13.18125, -1.072266, -61.80469)
T: 31.38428
```

Onde:

A: Aceleração nos eixos X, Y e Z
G: Taxa de giro nos eixos X, Y e Z
M: Campo magnético nos eixos X, Y e Z
T: Temperatura do sensor

Podemos separar estes dados e obter valores individuais a partir do vetor com três elementos. Por exemplo, para aceleração:



O bloco "**in list**" obtêm um valor da lista. Assim, o resultado desta programa será:

```
A =(0.04309563, 0.1101333, 9.564836)
Ax = 0.04309563
Ay = 0.1101333
Az = 9.564836
```

Note que, no teste acima, o Az está medindo 9.56 m/s² - praticamente 1g (aceleração da gravidade). A próxima figura ilustra esta situação, mostrando que o acelerômetro pode ser utilizado para medir ângulos, a partir da análise da medida da aceleração da gravidade em um determinado eixo.

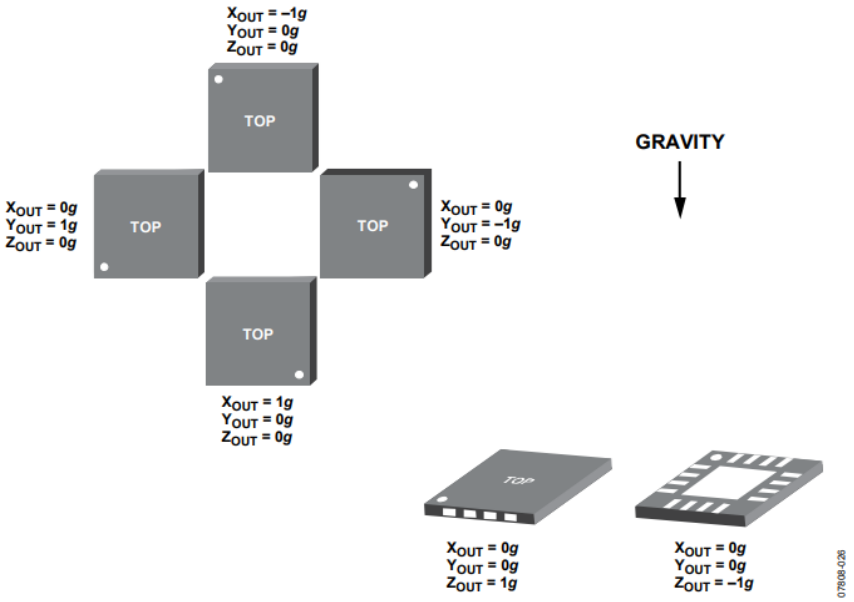


Figure 24. Output Response vs. Orientation to Gravity

Figura:
https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL335.pdf

Considerando que o satélite está na superfície do planeta, para fins experimentais, podemos calcular seu ângulo de inclinação. Este cálculo considera que a aceleração na superfície é de 1g (9.8 m/s²). Assim, através de algumas equações matemáticas, podemos transformar a aceleração em ângulos. A figura abaixo representa os chamados "Ângulos de *Euler*", muito usados em aviação e outras aplicações aeroespaciais. Os ângulos são a rolagem (*roll*), inclinação (*pitch*) e arfagem (*yaw*):

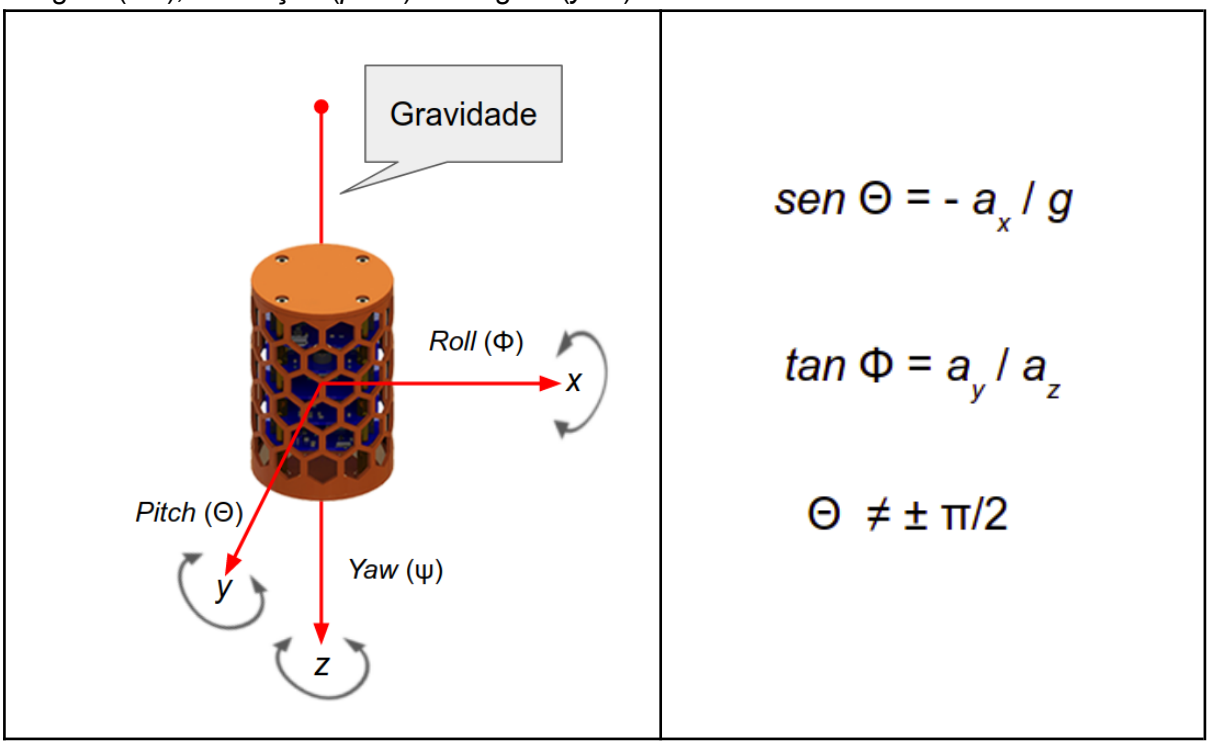
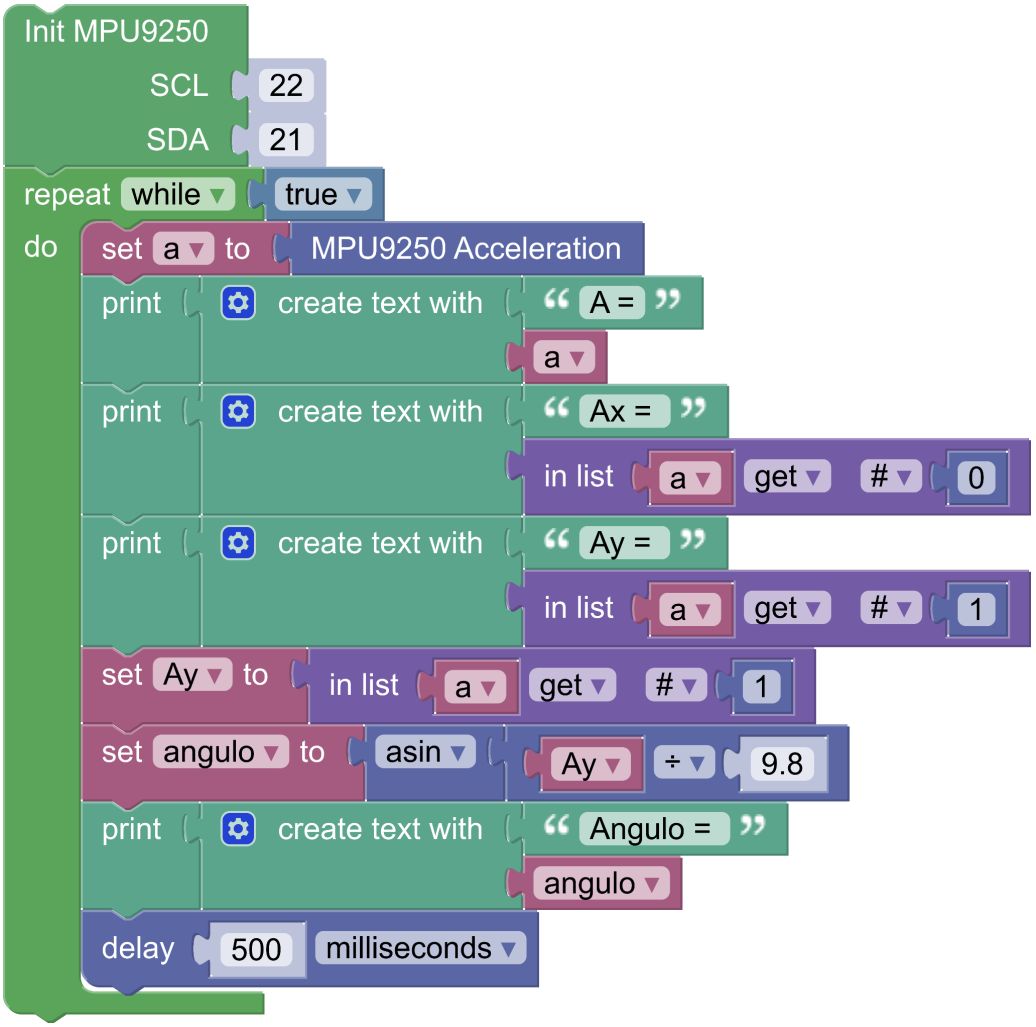


Figura adaptada de:
https://www.researchgate.net/figure/Roll-ph-Pitch-th-and-Yaw-ps-Angles-in-body-frame_fig1_270555146

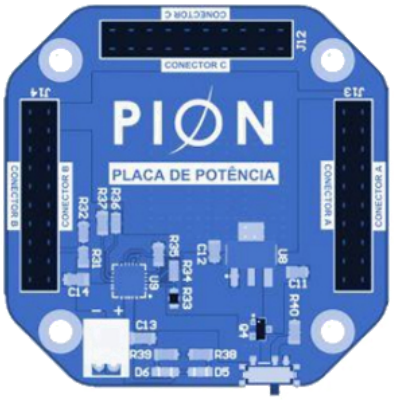
Assim, podemos fazer um programa que mede ângulo, com base na aceleração da medida pelo acelerômetro. De forma simplificada:



Estes ângulos também são chamados de “atitude” do satélite, algo fundamental para aplicações de satélite, pois, em muitos casos, é preciso que o satélite esteja “apontado” / orientado para uma direção. Especialmente, quando há câmeras.

Atividade: Crie um programa que emite diferentes bipes, dependendo do ângulo de inclinação do satélite.

Bateria e placa de potência



Placa de Potência

CanSat

Placa de Potência

CubeSat

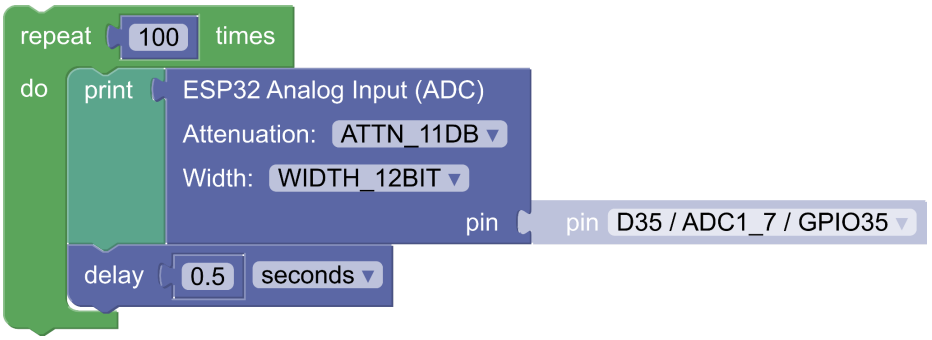
A placa de potência fornece energia para todas as outras placas / módulos do sistema. Ela possui um circuito de gerenciamento de carga da bateria, além do botão de liga e desliga do kit.

Medida de nível de bateria

Através da conexão da bateria com a placa de potência, a bateria fornece energia para o sistema. Neste sentido, é possível monitorar o nível de bateria utilizando o pino ADC1_7 / GPIO35. O valor varia de 0 até cerca de 2600.

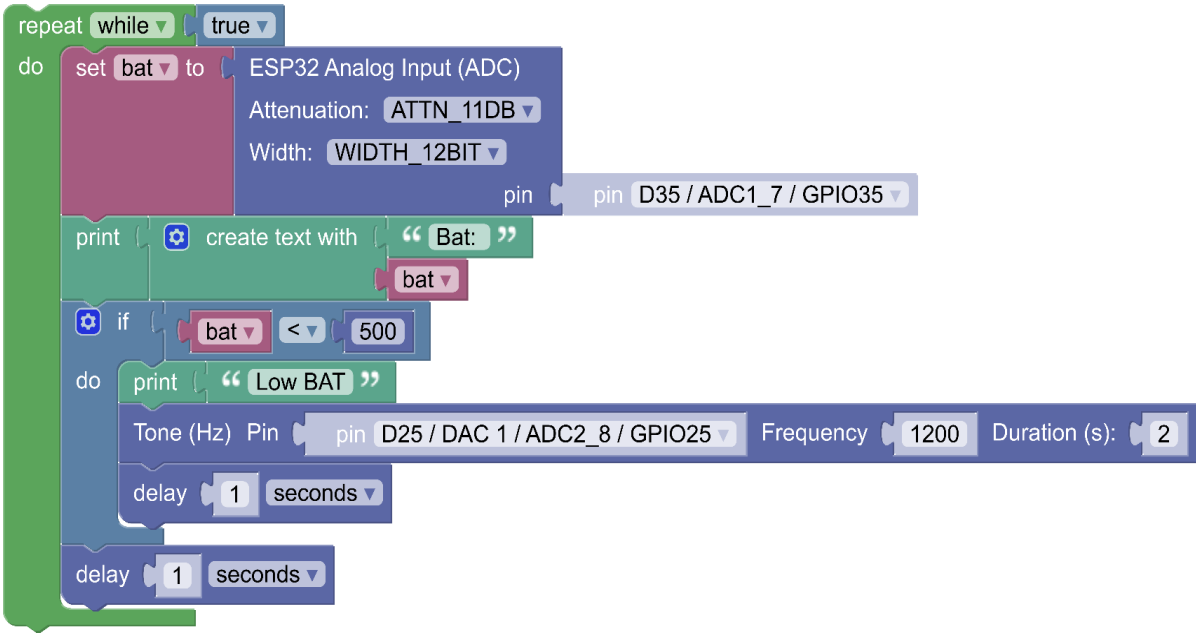
Atividade: Crie um programa que emite bipes constantes quando a bateria está próxima de terminar.

Dica: para medir o nível da bateria, utilize o bloco do conversor analógico-digital (ADC) com opção de 11db de atenuação e resolução de 12 bits.



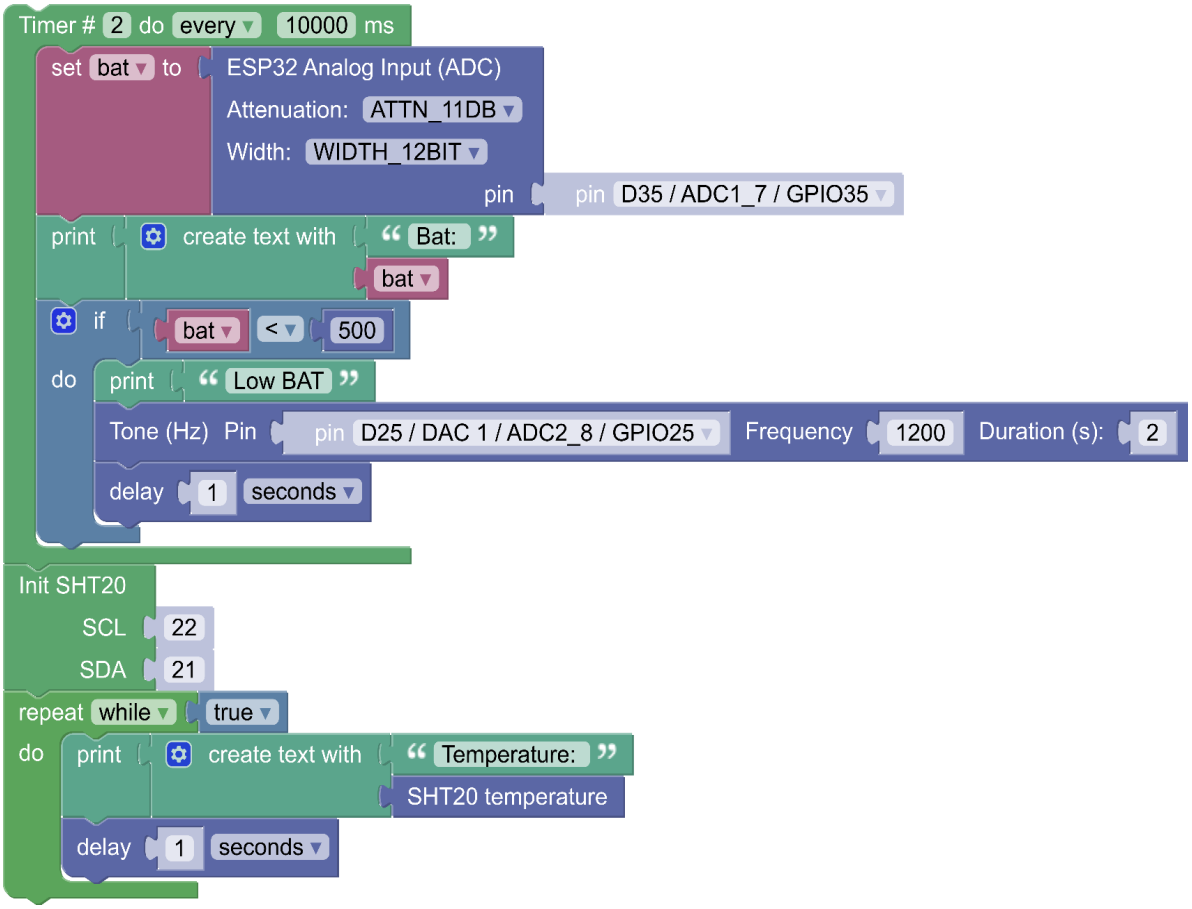
Verificando uma condição: alarme de nível de bateria e *timer*

Uma outra possibilidade é decidir com base na leitura de um sinal. Por exemplo, acionar um dispositivo quando um sensor detectar algum nível pré-determinado. O próximo exemplo mostra um programa que emite um BIPE sonoro de alerta quando o nível de bateria estiver abaixo de 500, através do uso do bloco IF.



Uma alternativa flexível para executar ações de forma assíncrona são os *timers*. Cada *timer* pode ter uma sequência de códigos executados em momentos pré-programados para acionamento do *timer*, sendo que os *timers* operam de forma independente do programa principal.

Dessa forma, o exemplo a seguir implementa um sistema com um *Timer* que verifica o nível da bateria a cada 10 segundos. Em paralelo, o laço principal **repeat while true** exibe a medida de temperatura a cada um segundo.

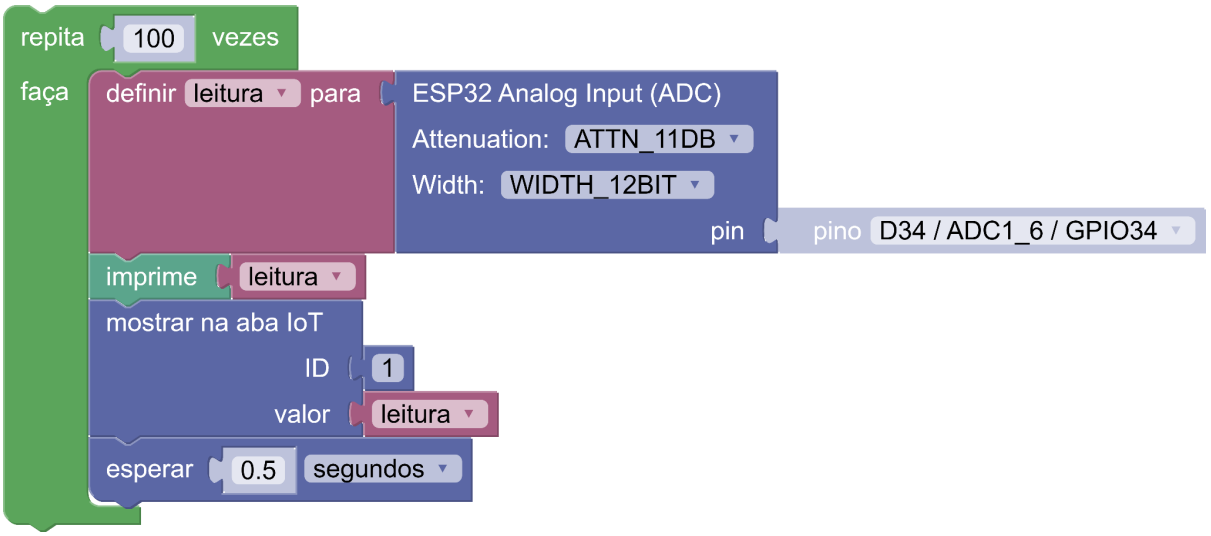


Resultado da execução deste programa:

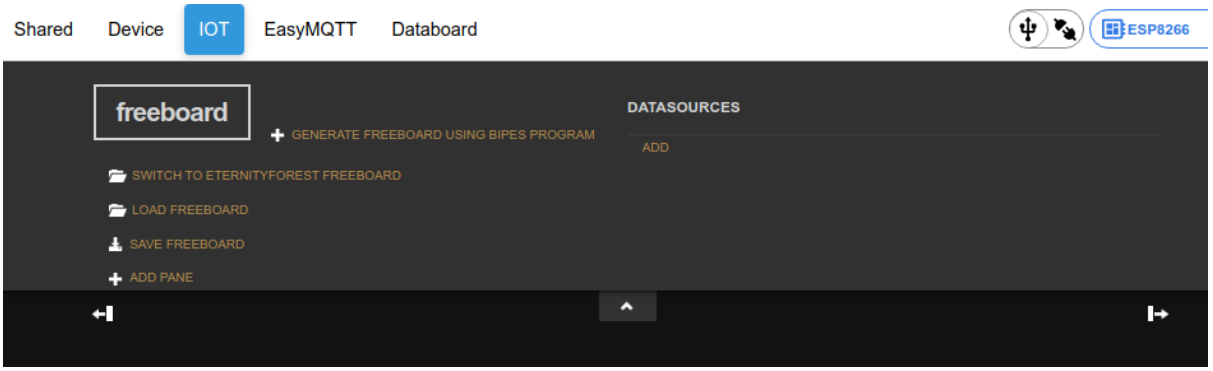
```
…

Temperature: 30.63001
Temperature: 30.63001
Temperature: 30.64074
Temperature: 30.63001
Temperature: 30.63001
Temperature: 30.63001
Bat: 2414
Temperature: 30.63001
Temperature: 30.63001
Temperature: 30.63001
Temperature: 30.63001
Temperature: 30.63001
Temperature: 30.63001
Temperature: 30.63001
Temperature: 30.63001
Temperature: 30.63001
Bat: 2411
…
```

Plotando dados em tempo real via cabo USB

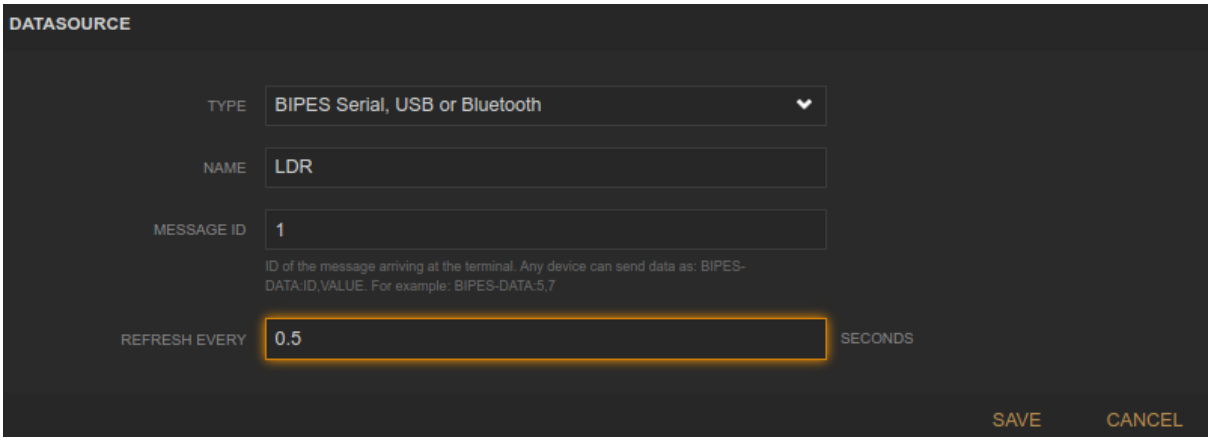
Os programas mostrados até agora mostram dados, em forma textual, no **console**. Mas podemos avançar mais com esta montagem e visualizar as leituras de forma gráfica. Para isso, adicione o bloco **“Mostrar na aba IoT”** / **“Show data on IOT tab”**:



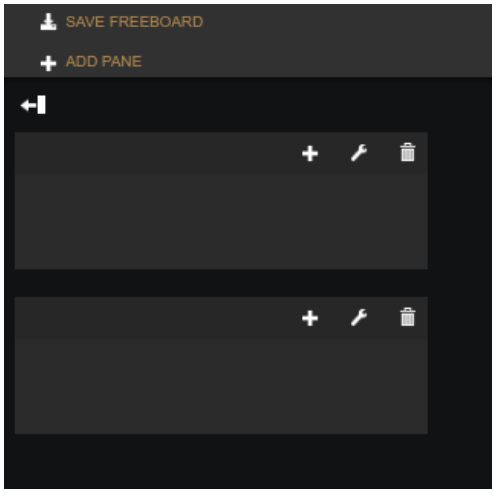
Depois utilize a aba IoT para definir uma tela de visualização de dados (*dashboard)*:



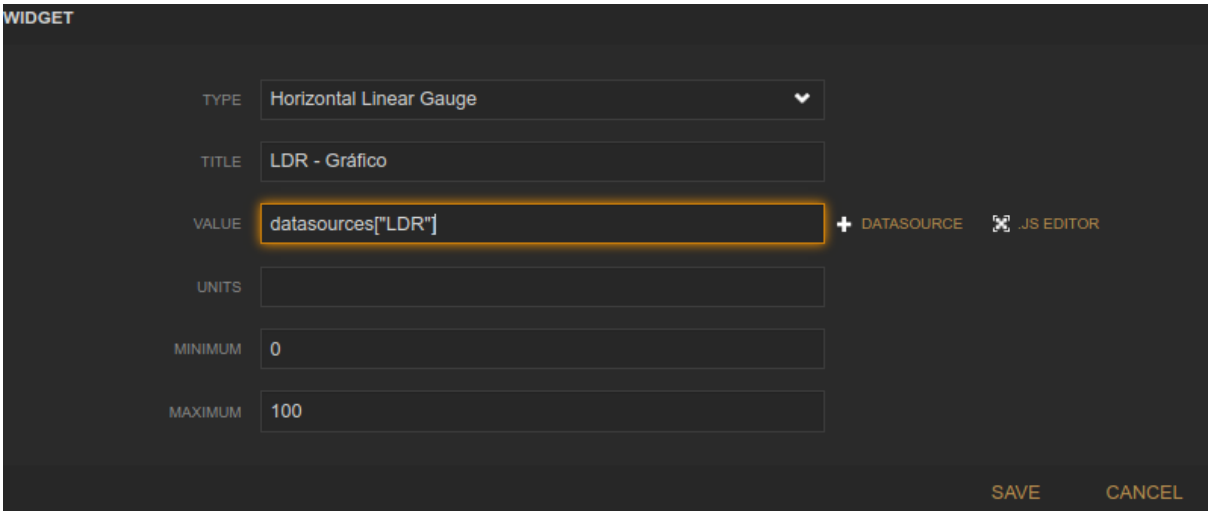
Em *datasource,* clique em ADD e escolha:



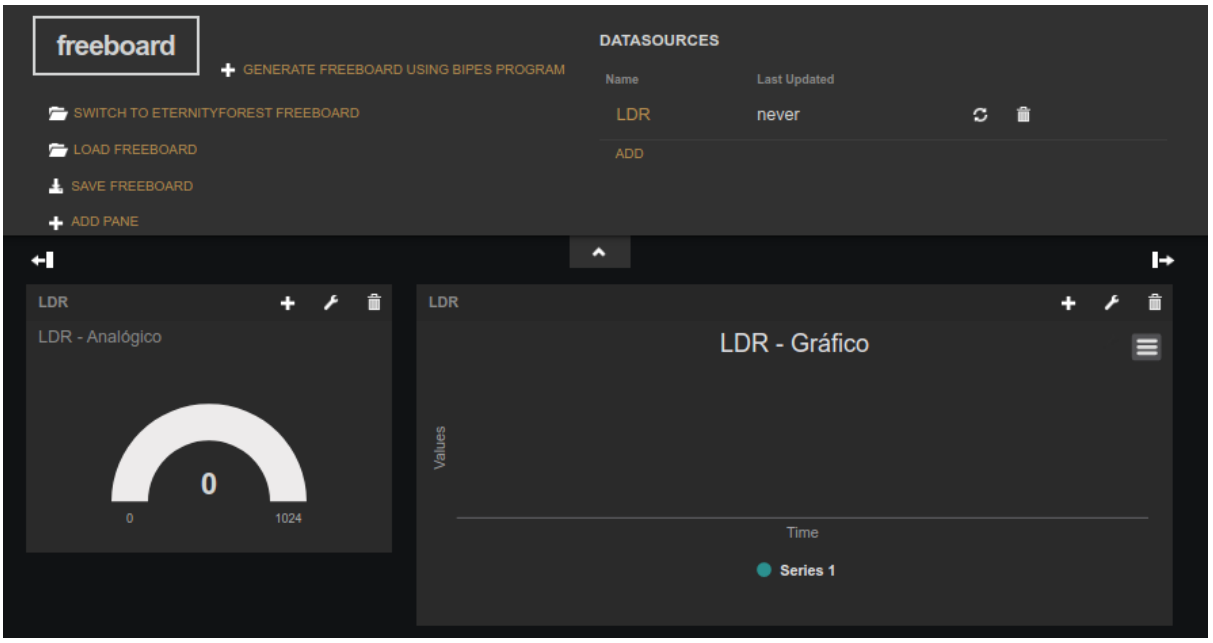
Salve e adicione um ou mais “panes”, com o botão ADD PANE:



Arraste e solte os “panes” para ajustar suas preferências. Em cada “pane”, adicione componentes de visualização, conforme você preferir, e escolha o datasource LDR:



Exemplo:

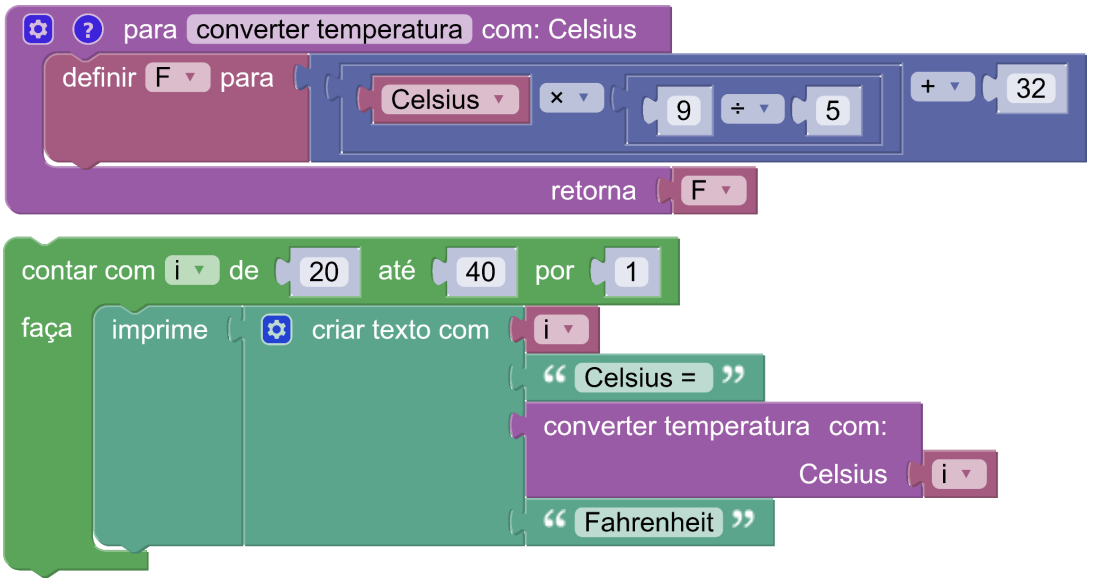


Execute o programa e verifique a visualização! Ao variar a iluminação que incide sobre o sensor de luz, as leituras vão mudar e ser exibidas em tempo real.

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 / *Functions***.

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 seu uso:



Note que o bloco criado “converter temperatura com” poderá ser reutilizado em várias partes do programa - sempre que necessário. 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

<< dados omitidos >>

39 Celsius = 102.2 Fahrenheit
40 Celsius = 104.0 Fahrenheit
>>>
```

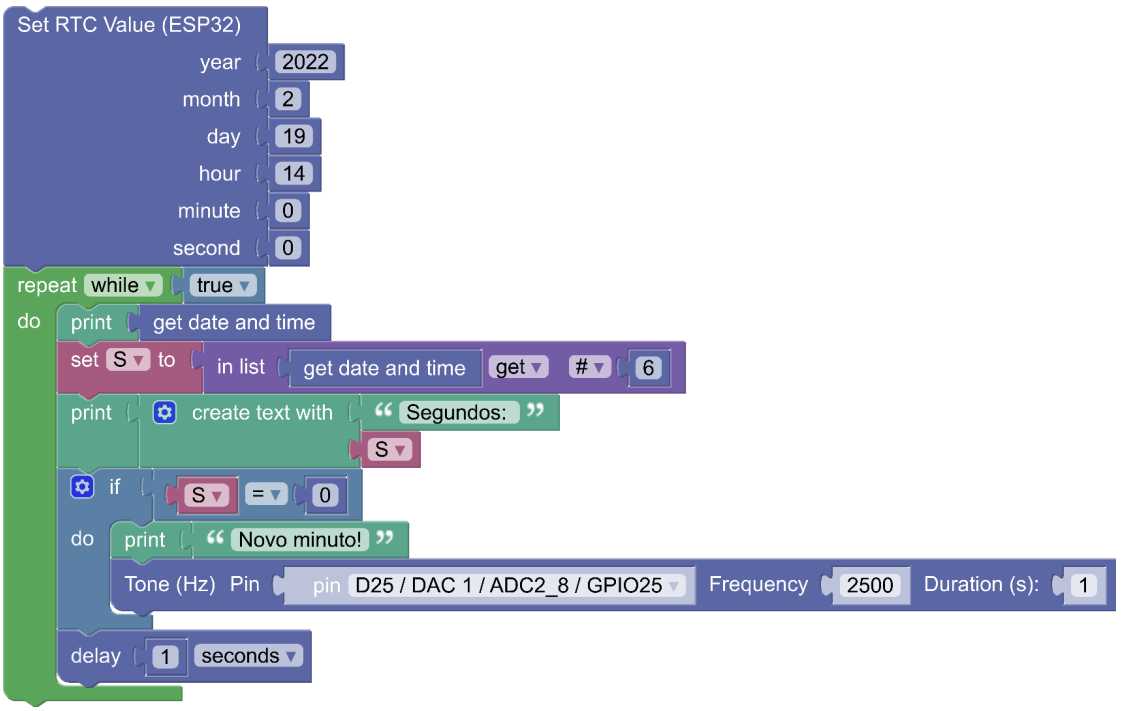
Após criar as funções, você pode compartilhar seu programa com o botão e usar as funções criadas em outros programas!

Data e hora (RTC)

O registro e uso de informações de data e hora é bastante útil em diversas aplicações embarcadas, como por exemplo, para associar a data e hora de medidas e eventos monitorados por um sistema embarcado.

Através do MicroPython, 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, pela Internet, utilizando o recurso / bloco ***Network Time Protocol*** **NTP**, ou até mesmo a partir da data e hora fornecidas por um receptor do sistema de posicionamento global (GPS), que fornece informações de data e hora extremamente precisas.

O exemplo a seguir mostra um programa que inicia o RTC com a data/hora de 19/02/2022 às 14:00:00. Note, que o programa trata a data e hora como uma estrutura de lista, separada por vírgulas: **(2022, 02, 19, 5, 14, 00, 00, 0)**, onde cada elemento do vetor é dado por: (ano, mês, dia, dia da semana, hora, minuto, segundo, milissegundo). É possível então, usar o bloco **na lista** para pegar o 6.º 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, um sinal sonoro é emitido através do *buzzer* ligado ao pino GPIO25 de seu satélite.

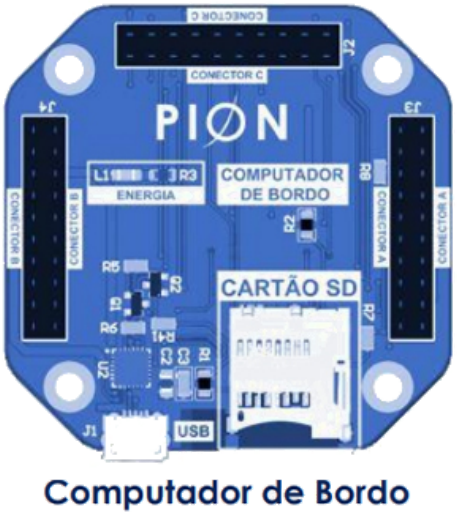


O resultado pode ser visto no **Console**:

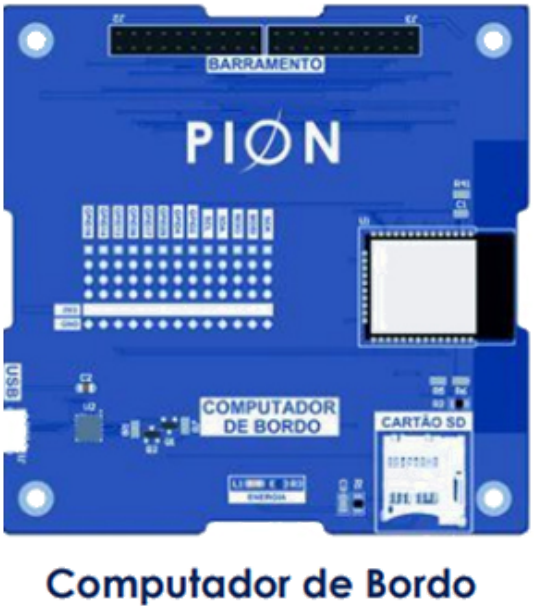
```
===
Segundos: 54
(2022, 02, 19, 1, 14, 00, 55, 00)
Segundos: 55
(2022, 02, 19, 1, 14, 00, 56, 00)
Segundos: 56
(2022, 02, 19, 1, 14, 00, 57, 00)
Segundos: 57
(2022, 02, 19, 1, 14, 00, 58, 00)
Segundos: 58
(2022, 02, 19, 1, 14, 00, 59, 00)
Segundos: 59
(2022, 02, 19, 1, 14, 01, 00, 00)
Segundos: 0
Novo minuto!
(2022, 02, 19, 1, 14, 01, 03, 00)
Segundos: 3
(2022, 02, 19, 1, 14, 01, 04, 00)
Segundos: 4
(2022, 02, 19, 1, 14, 01, 05, 00)
Segundos: 5
(2022, 02, 19, 1, 14, 01, 06, 00)
Segundos: 6
(2022, 02, 19, 1, 14, 01, 07, 00)
Segundos: 7
(2022, 02, 19, 1, 14, 01, 08, 00)
Segundos: 8
(2022, 02, 19, 1, 14, 01, 09, 00)
Segundos: 9
```

Note que, mesmo com a **espera** de 2 segundos (1 segundo no sinal sonoro, 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.

Computador de bordo



CanSat



CubeSat

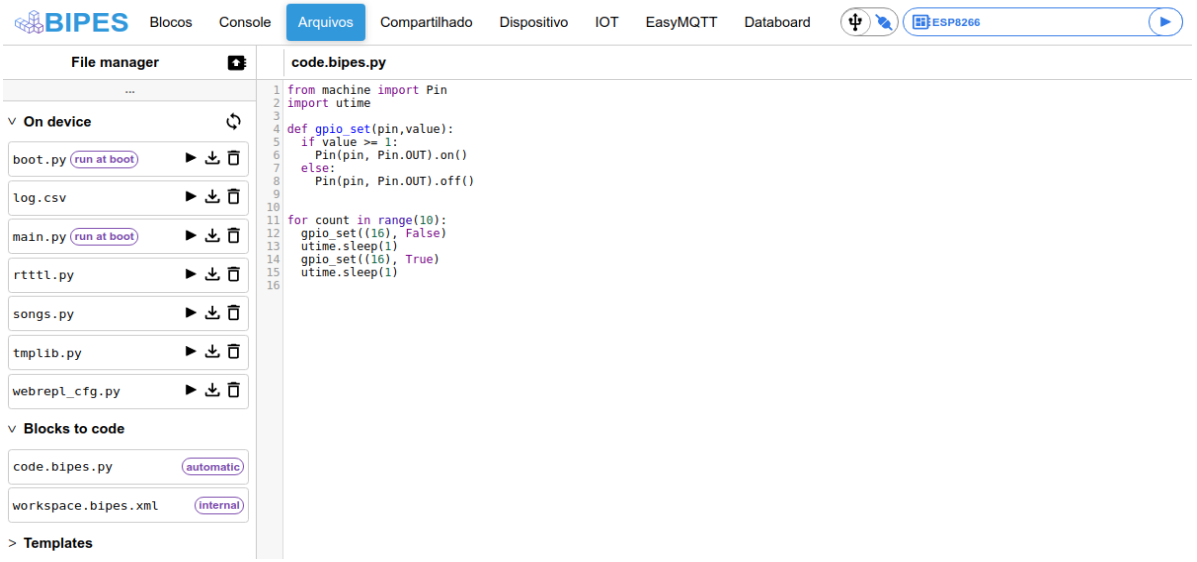
O computador de bordo dos kits é baseado no módulo ESP32. Além do processador *dual core*, memória RAM e memória FLASH, o módulo ESP32 também possui comunicação *Bluetooth* e *Wifi*. Dessa forma, utilizando o BIPES, é possível programar e interagir com o módulo via *Bluetooth* (utilize o menu Network → Bluetooth REPL) ou programar, monitorar e interagir com o módulo via WiFi através da tecnologia WebREPL, explicada a seguir. Com o módulo conectado à Internet via WiFi, é possível enviar telemetria e realizar diversas outras atividades!

Alguns detalhes adicionais do computador de bordo são mostrados na tabela à seguir:

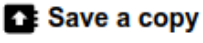
Especificação	Descrição
Microcontrolador	ESP32 (32-Bit / Dual Core / 240 MHz)
Tensão de operação	3.3V
Bateria	Li-Po 3.7V / 400mah
Memória Flash	4 Mb
Armazenamento Externo	Cartão microSD até 16 Gb
Wi-Fi	802.11 b/g/n 2.4 GHz ~ 2.5 GHz

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**. 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 e acessada via WebREPL. A figura a seguir mostra um exemplo da aba de arquivos do **BIPES**:



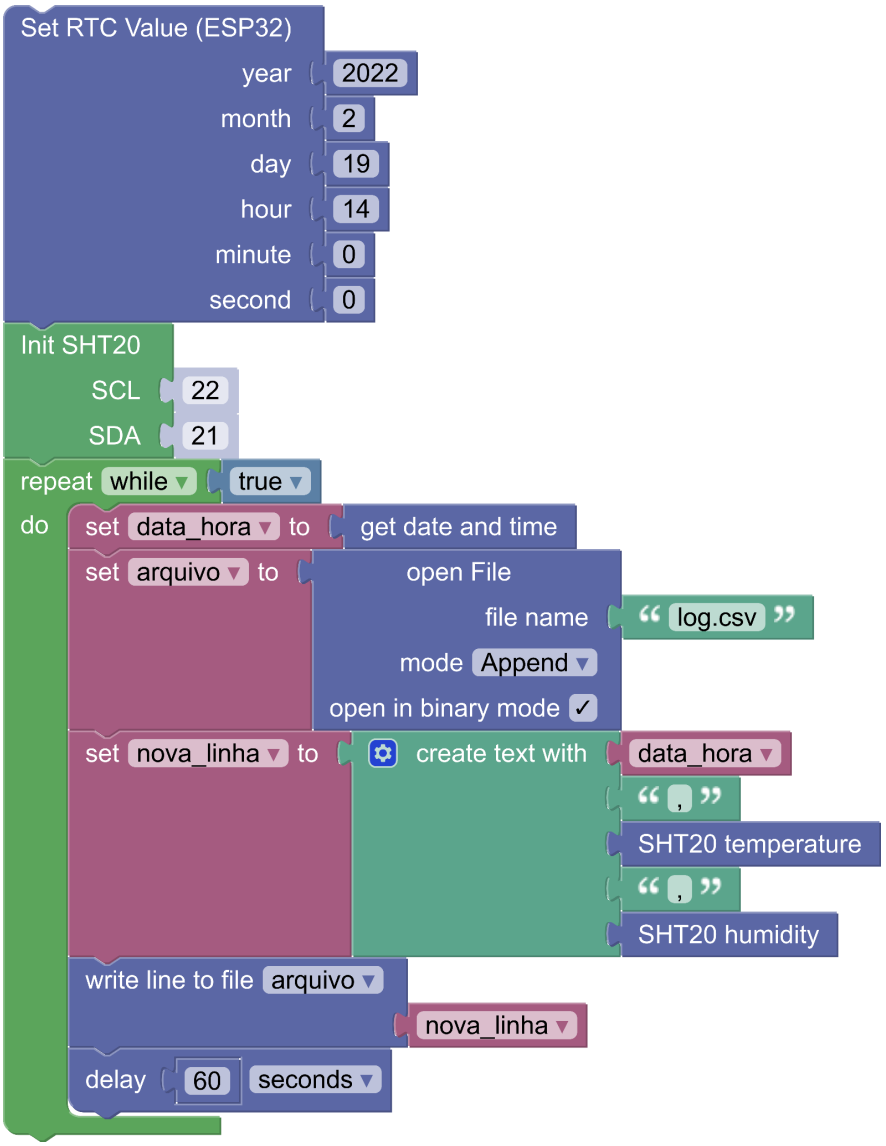
A tabela abaixo explica cada função disponível nesta guia do BIPES:

	**Upload script to device** Envia um arquivo do seu computador para a placa
	**Save a copy** 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.bipes.py	**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 medida da umidade do ar e temperatura e as 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.



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:

```
…
(2022, 02, 19, 2, 14, 00, 00, 00),29.1,70
(2022, 02, 19, 2, 14, 01, 00, 00),29.2,71
(2022, 02, 19, 2, 14, 02, 00, 00),29.3,70
(2022, 02, 19, 2, 14, 03, 00, 00),29.2,70
(2022, 02, 19, 2, 14, 04, 00, 00),29.1,69
…
```

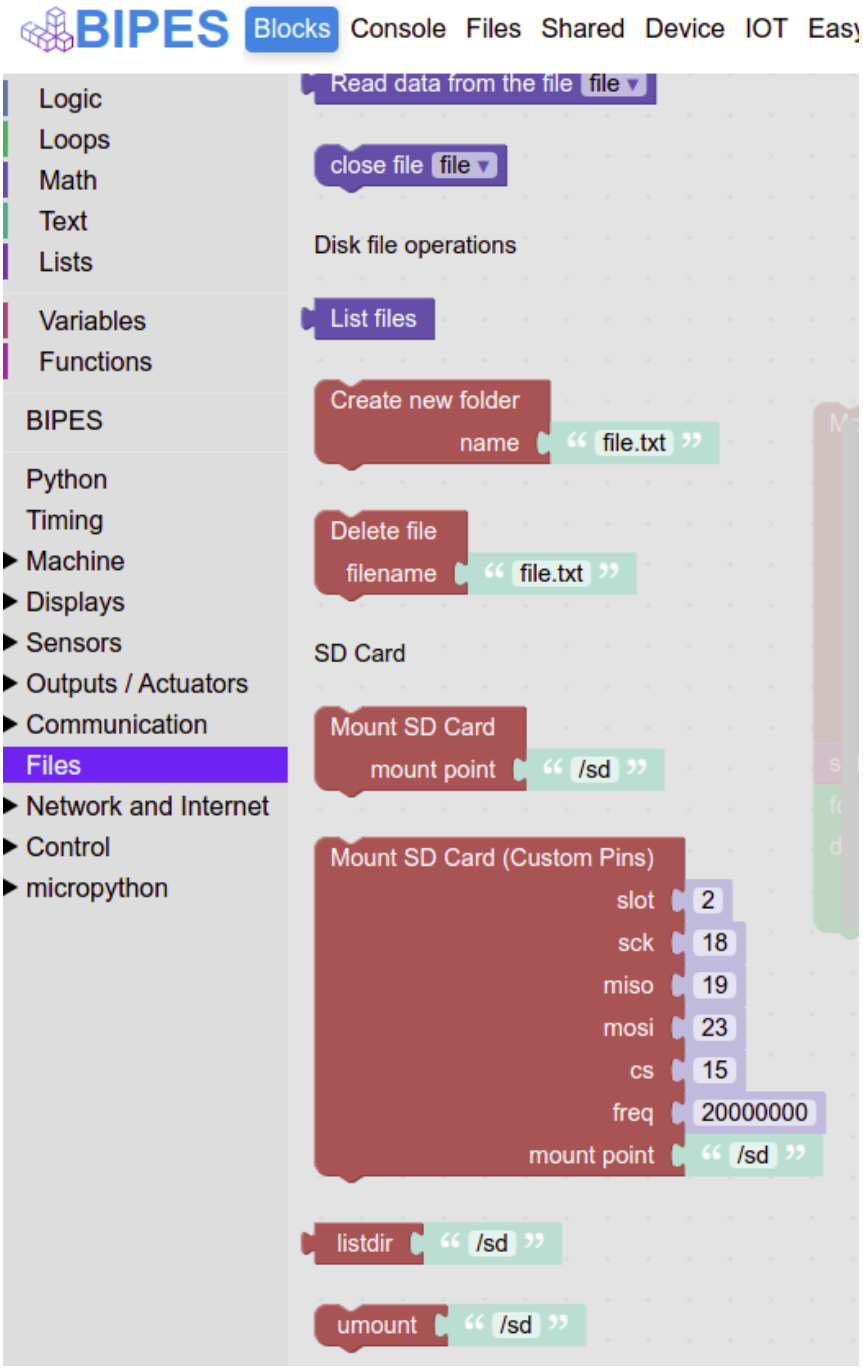
Como se pode observar, a data e hora são armazenadas, e depois a leitura da entrada analógica é salva após a vírgula “,”, conforme instruído no programa pelo bloco “**set nova_linha to …”**. Caso seja necessário, o MicroPython também permite que estas operações sejam feitas em um cartão de memória do tipo SD Card, permitindo um maior espaço de armazenamento para coleta de dados.

Atividade: Crie um programa que realiza a medida da concentração de CO2, luminosidade, temperatura e umidade e a armazena no arquivo **log.csv** de forma associada à data e hora de leitura, com coleta de dados a cada 1 segundo. Após alguns minutos, acesse o arquivo log.csv, e abra este arquivo no Excel ou LibreOffice Calc. Finalmente, crie gráficos com estes dados.

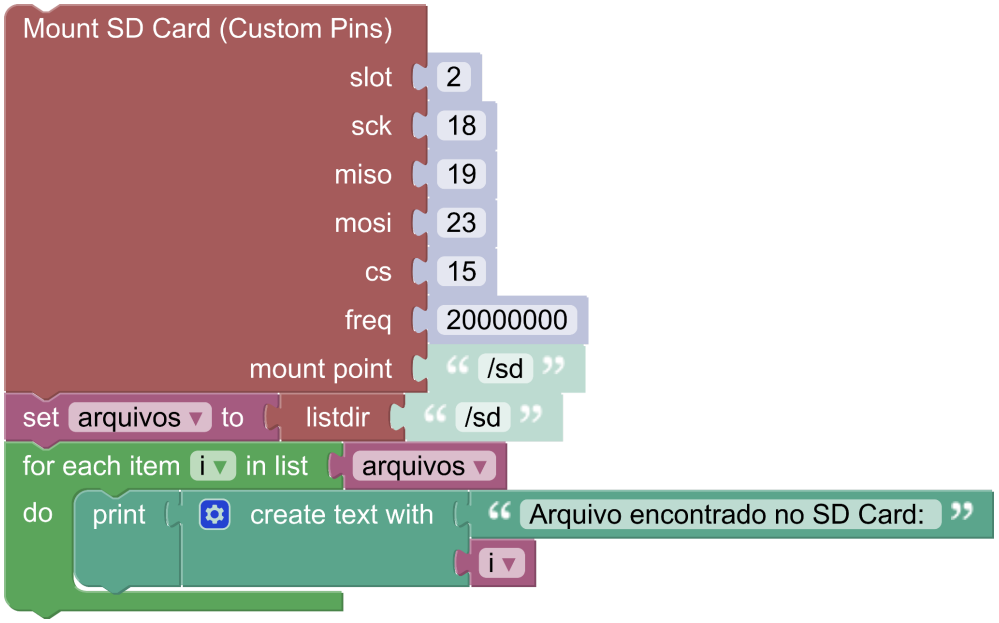
Arquivos e Cartão de memória

Os kits possuem uma entrada para cartão de memória micro SD, uma alternativa para gravação de dados fora a memória interna da placa ESP32. Dessa forma, os dados coletados utilizando os sensores podem ser gravados em um cartão externo e o BIPES tem recursos para realizar a leitura e gravação dos dados no cartão.

Para utilizar o cartão de memória, utilize o bloco **Mount SD Card (Custom Pins)**, disponível no item **Files / Arquivos** da barra lateral esquerda de ferramentas.

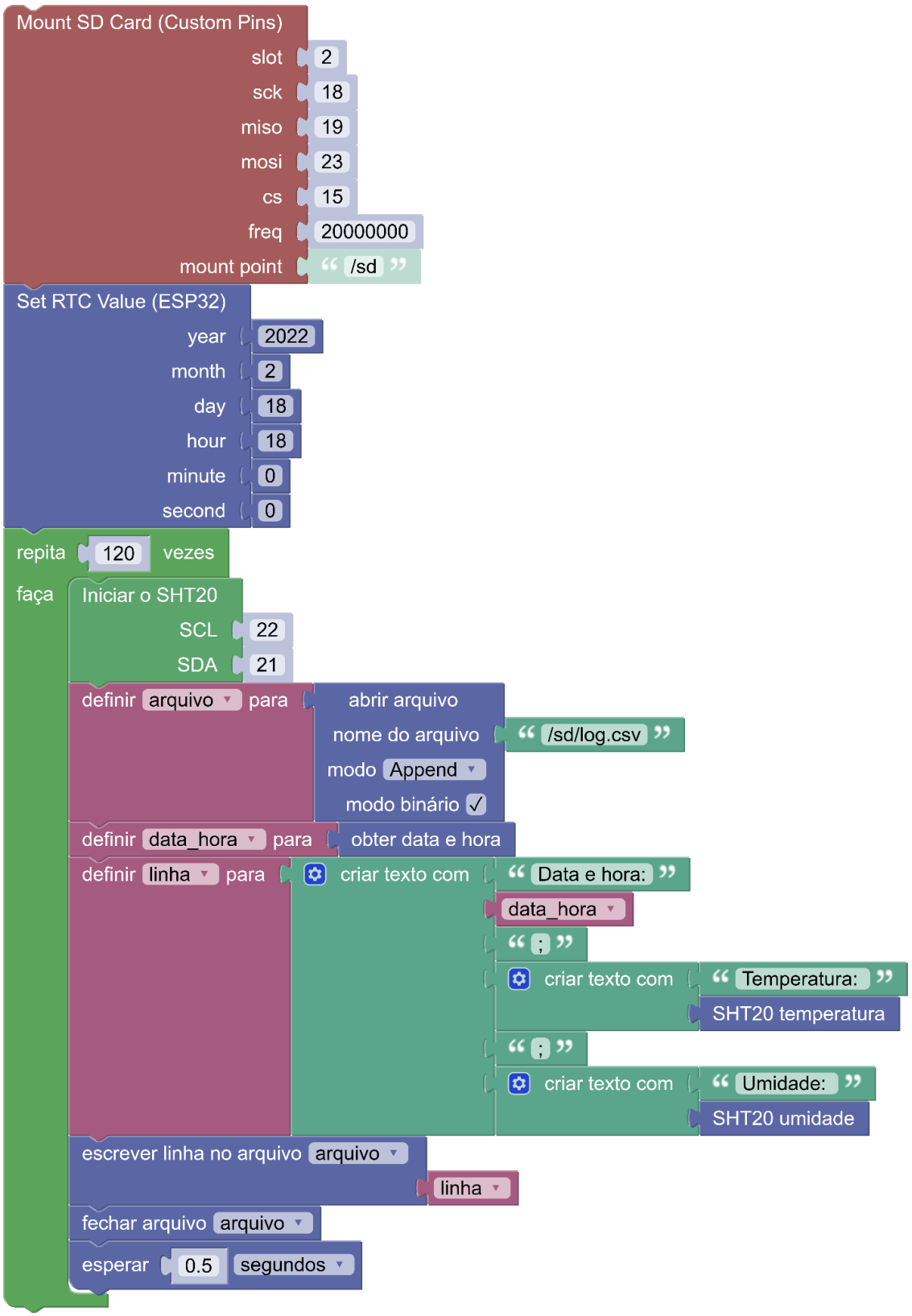


Com um cartão SD Card inserido em seu satélite, você pode verificar quais os arquivos presentes no cartão. Veja, por exemplo, como pode ser realizado essa busca:



Os arquivos presentes no cartão SD serão impressos no **Console**. Outras possibilidades para o seu uso, utilizando as ferramentas apresentadas nas seções anteriores, está relacionada a uma das principais funções do Cartão SD: a possibilidade de gravar dados. No nosso caso podemos gravar dados dos sensores presentes no satélite.

O programa a seguir realiza a montagem do cartão SD, define uma data e hora RTC e faz a leitura dos sensores de temperatura e umidade, que será gravado associado a data e hora durante um período de 60s. Note que este programa é similar a um programa anterior, mas agora, os dados são armazenados no cartão de memória SD Card.

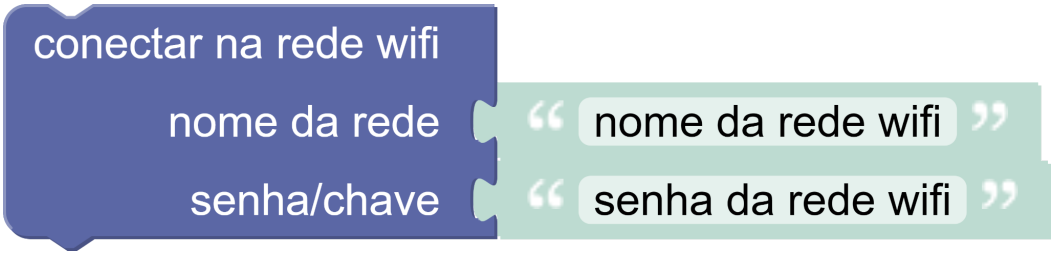

Mount SD Card (Custom Pins)
slot 2
sck 18
miso 19
mosi 23
cs 15
freq 20000000
mount point "/sd"
Set RTC Value (ESP32)
year 2022
month 2
day 18
hour 18
minute 0
second 0
repita 120 vezes
faça
Iniciar o SHT20
SCL 22
SDA 21
definir arquivo para
abrir arquivo
nome do arquivo "/sd/log.csv"
modo Append
modo binário ✓
definir data_hora para obter data e hora
definir linha para
criar texto com
"Data e hora:"
data_hora
";"
criar texto com
"Temperatura:"
SHT20 temperatura
";"
criar texto com
"Umidade:"
SHT20 umidade
escrever linha no arquivo arquivo
linha
fechar arquivo arquivo
esperar 0.5 segundos

WiFi: Listar redes

Nas próximas atividades, conectaremos o satélite à Internet! O primeiro passo é listar as redes WiFi existentes. Tente você mesmo. As opções de rede estão no final da barra de blocos, na opção “Network”. O seguinte programa lista as redes wifi disponíveis:

Conectando em redes WiFi / Internet

Para conectar à uma rede WiFi (com ou sem acesso à Internet), basta utilizar o seguinte bloco:



Informe o nome da rede e senha e tente rodar o programa para que a placa se conecte à Internet. Após listar as redes e conectar na Internet, já podemos enviar dados para a nuvem e acessar o **Console** do satélite via WiFi.

Acesso ao **Console** via WiFi

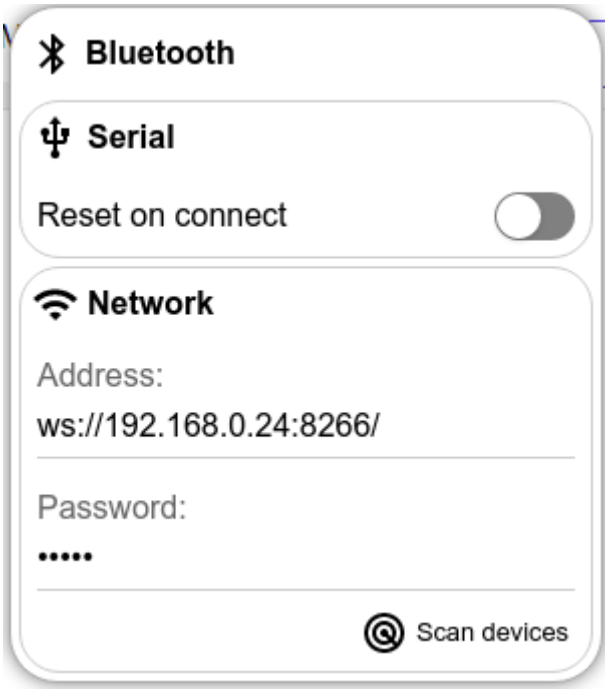
Você pode acessar, programar e monitorar o kit usando WiFi! Conecte ele na rede WiFi, e na sequência use a opção de conexão via wifi. Desconecte o cabo USB e programe-o sem fio!

Para tanto, após conectar seu satélite na rede, você precisará saber seu endereço IP. O IP pode ser verificado com os seguintes blocos:

Ao executar este programa, o endereço IP do satélite será exibido no **console.** Na sequência, para ser possível acessar e programar o satélite via WiFi, é preciso ativar o WebREPL. Para ativar o WebREPL, utilize o seguinte bloco:

Execute o programa com o bloco acima e siga as instruções no **console**, respondendo que sim (Y) ao receber a pergunta se você deseja ativar o WebREPL e informe uma senha de acesso. Esta senha ficará armazenada no arquivo webrepl.conf e você pode modificá-la sempre que precisar.

Com o WebREPL ativado, você já pode desconectar o seu satélite do cabo USB e utilizá-lo com fonte de energia a partir da bateria. Certifique-se de que ele está conectado na rede e lembre o IP obtido pelo seu satélite. Na sequência, clique no botão de conexão do BIPES ( ), digite o IP do seu kit e a senha que você informou ao configurar o WebREPL. Depois clique com o mouse em **Network**. Ao digitar o IP do seu kit, certifique-se de que o endereço inicia por ws:// e termina com :8266/, que significa que a conexão é via WeSockets na porta 8266.

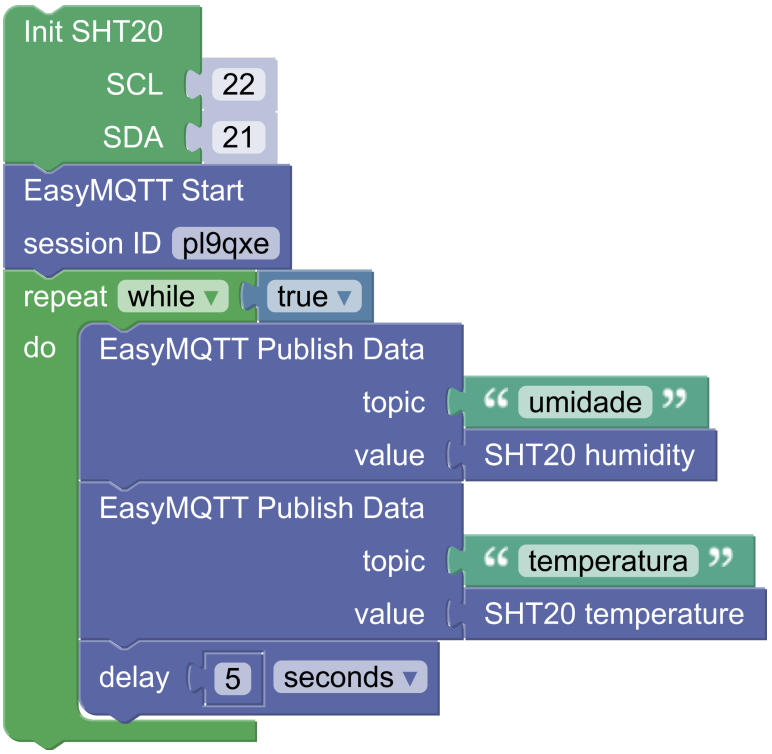


Finalmente, caso você não lembre o IP de seu satélite, mas esteja conectado na mesma rede WiFi que o satélite, utilize o botão ***Scan devices***, que irá listar todos dispositivos disponíveis em sua rede com MicroPython e WebREPL ativado.

Com a conexão ativa, você poderá acessar o **console**, enviar e gerenciar arquivos e executar operações de envio de dados para nuvem, como estudos e aplicações Internet das Coisas.

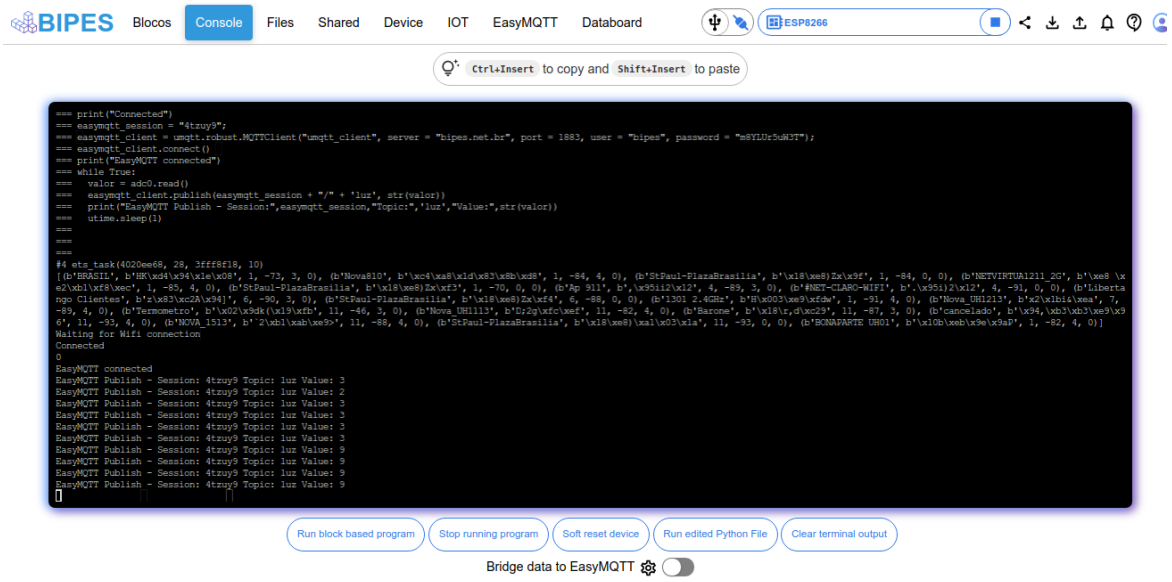
Plotando dados em tempo real via WiFi / nuvem

Com a placa conectada na Internet, via WiFi, é possível enviar dados para a nuvem (serviços *online* disponíveis via Internet). Para tanto, podemos utilize a opção EasyMQTT do BIPES:

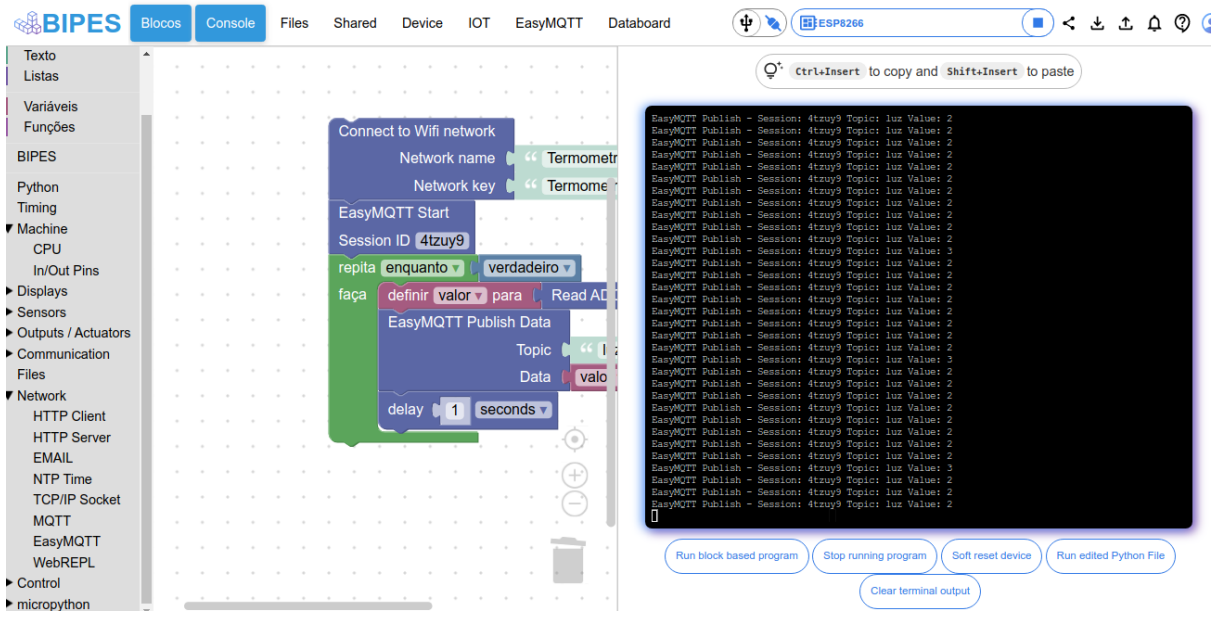


Note que o bloco **EasyMQTT Start** define automaticamente um ID de sessão (*session ID)*, que neste exemplo é **pl9qxe**. Este ID de sessão deverá ser usado para se referir a esta sessão de dados na aba IOT e ao compartilhar os dados deste programa. Note que podem ocorrer coincidências no uso de IDs de sessões, que eventualmente, podem ser iguais. Neste sentido, você pode verificar se uma sessão já possui dados através da aba EasyMQTT. Observe também que o ID desta sessão pode ser modificado, ao clicar no texto do bloco **EasyMQTT Start** e informar um novo ID.

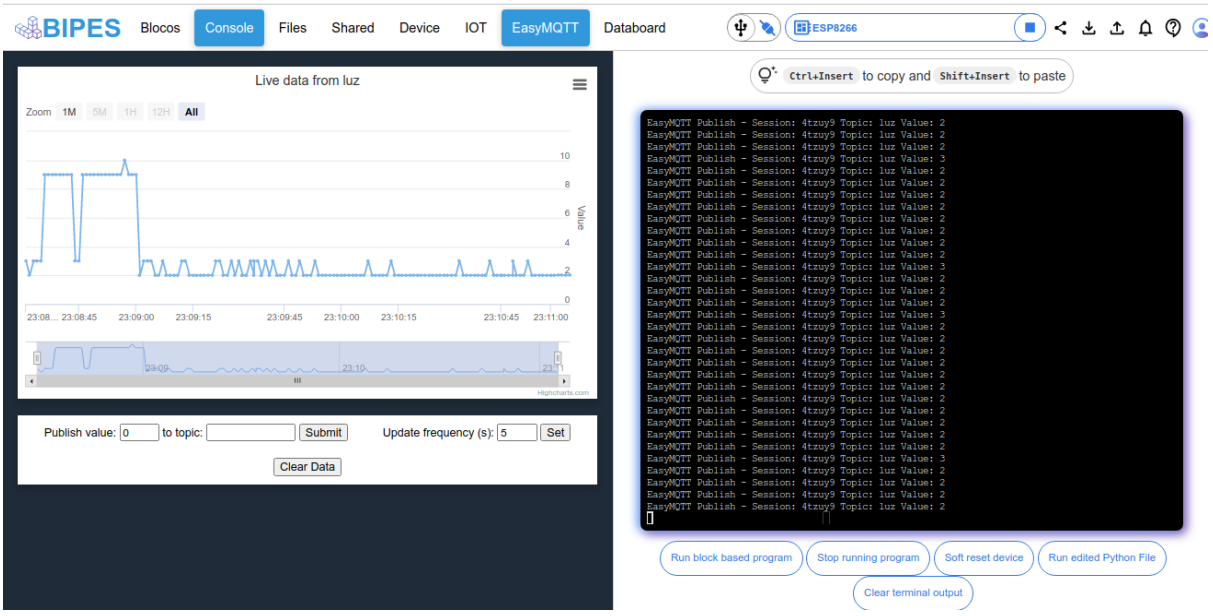
Este programa realiza a medida de umidade e temperatura e envia para a plataforma de armazenamento de dados do projeto BIPES, tornando os dados disponíveis para consulta a partir de qualquer dispositivo na Internet. Ao executar o programa, você pode ver o resultado na aba **Console**, indicando que os dados estão sendo enviados com sucesso.



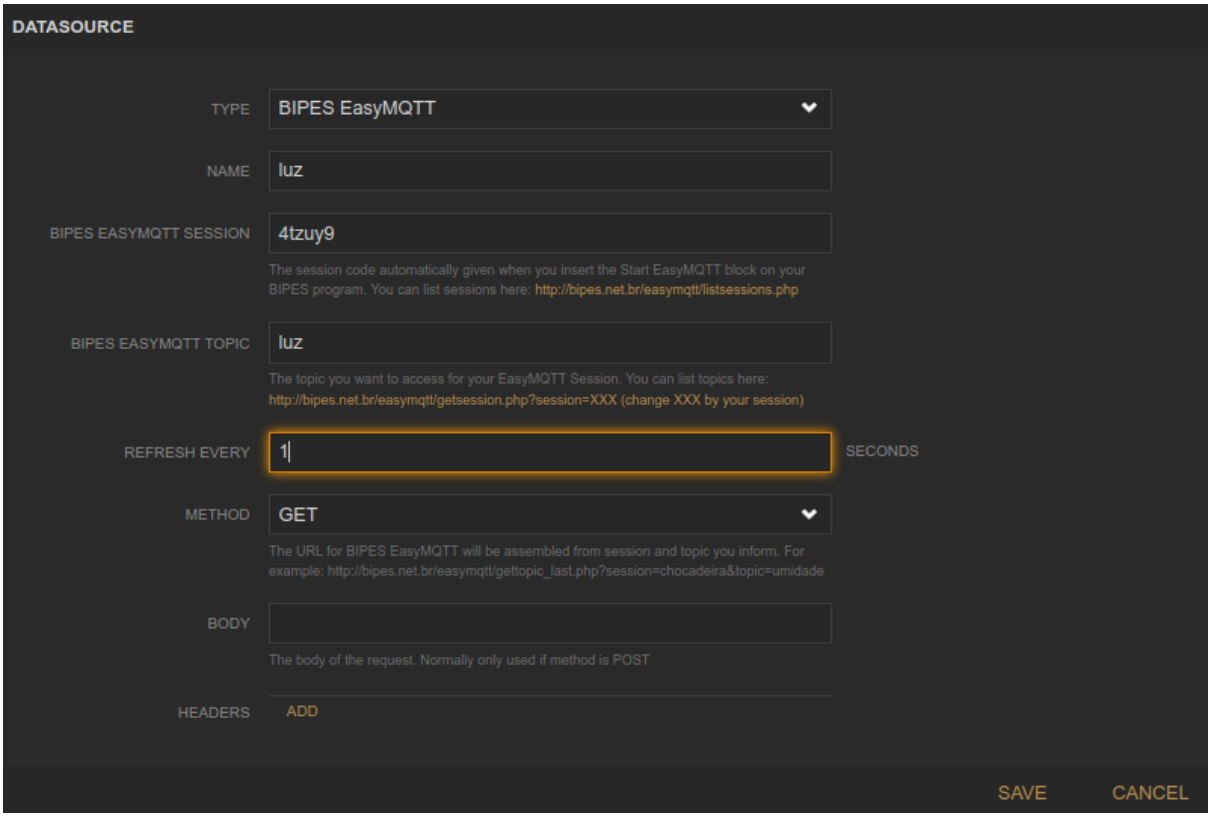
O BIPES tem um recurso muito útil: de ver duas abas ao mesmo tempo, clicando com o botão direito do mouse em uma das abas e com o esquerdo em outra. A imagem abaixo mostra um exemplo onde o programa e o console são visualizados simultaneamente:



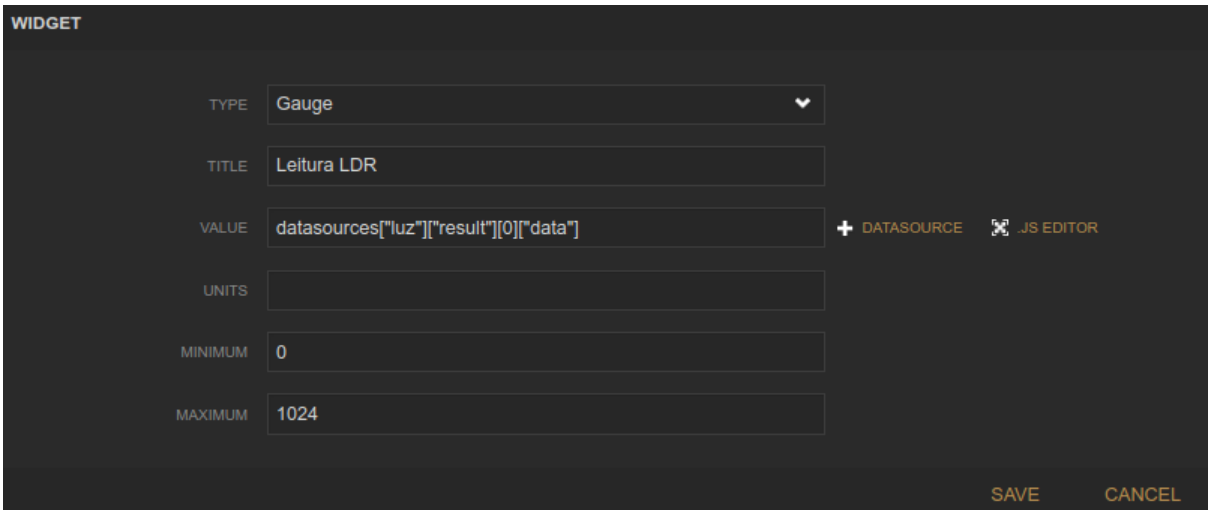
Agora, você pode clicar na aba EasyMQTT e ver os resultados enviados para a nuvem. Por exemplo:



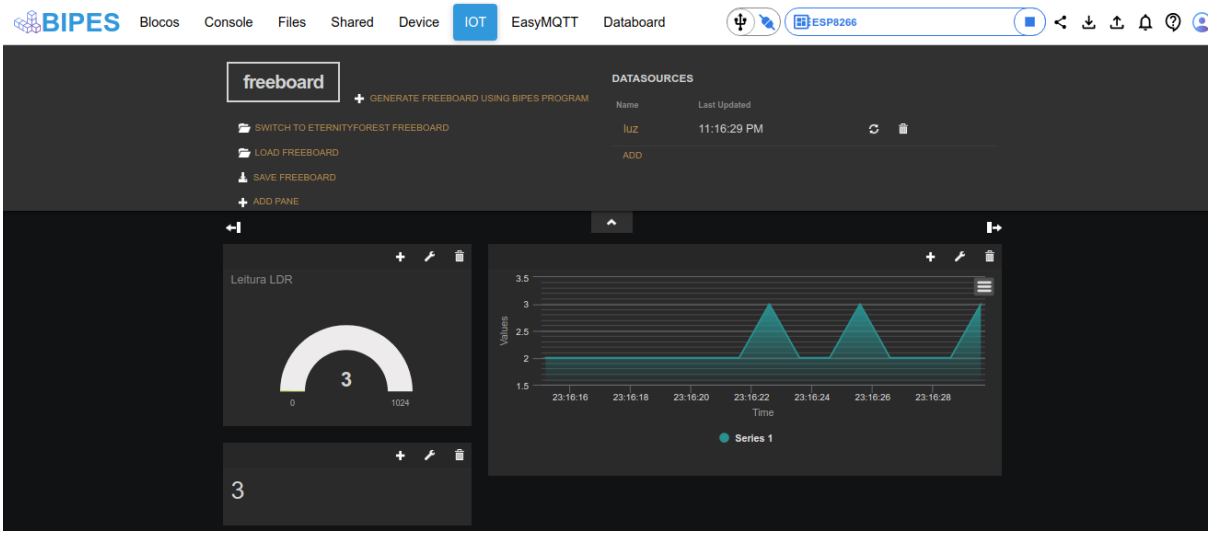
Também podemos criar um painel (*dashboard)* personalizado para visualizar estes dados, usando a aba IoT novamente. Mas desta vez, vamos usar o datasource EasyMQTT e informar os dados deste programa, como no exemplo à seguir. Note que você deverá preencher o **BIPES EasyMQTT Session** com o ID correto de sua sessão. Este exemplo utiliza dados do sensor de luz, podendo ser ajustado para qualquer outro sensor através dos campos abaixo.



Em seguida, vamos adicionar os “panes”:



Você pode adicionar outros componentes e personalizar seu painel. Por exemplo:



Note, que se você 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. Fique atento(a), pois o botão compartilhar vai compartilhar tanto o painel de visualização de dados (dashboard) quanto o seu programa

Também é possível compartilhar apenas o *dashboard* / painel de visualização de dados. Para tanto, basta clicar no botão **+ Share Board** na parte superior do *dashboard.* Ao clicar neste botão, um ***link*** será exibido juntamente com um QR-Code, que aponta para este link. Assim, qualquer dispositivo que acessar este link ou QR-Code, poderá visualizar o *dashboard* que você construiu, com acesso aos dados em tempo real. O *dashboard* também se ajusta para visualização em celulares / *smartphones.*

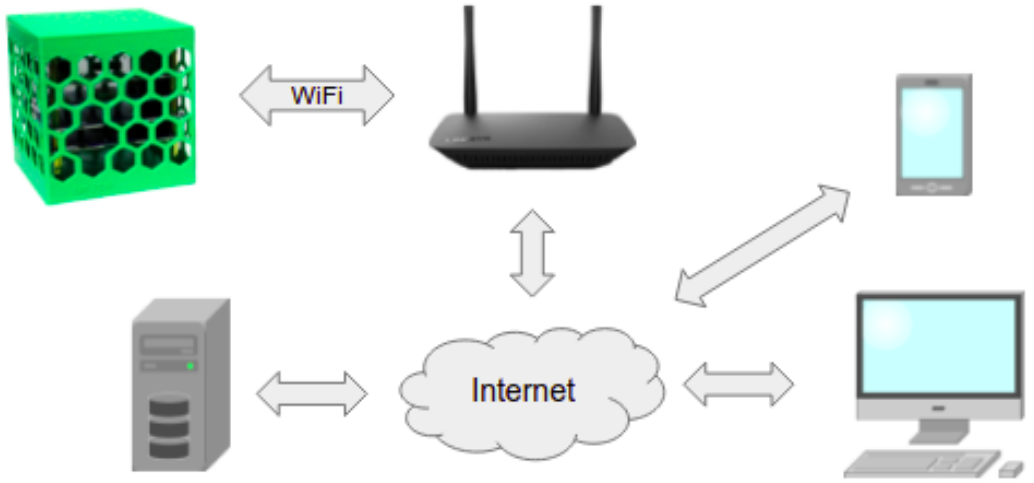
Comunicação web / HTTP

Uma atividade muito comum na atualidade é acessar sites web, como, por exemplo, BIPES. 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 um módulo ESP32 atua como cliente, realizando uma requisição para um servidor, via Internet. Note, que tal cenário também poderia ser invertido, onde a ESP32 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 dois *CubeSats* (ou *CanSats*) 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 dois dispositivos não possam se comunicar diretamente. Assim, também é possível que dois sistemas embarcados se comuniquem por intermédio de um servidor. Assim, os kits podem publicar ou consumir dados deste servidor, de forma que um dispositivo 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 é um serviço web que permite enviar dados, via HTTP, para o servidor MQTT do BIPES. Por exemplo:

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 *firewall* 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.

Existem dois principais tipos de requisições para clientes HTTP: o **HTTP GET** e o **HTTP POST**. À seguir, exploramos um exemplo com HTTP GET.

Exemplo de cliente HTTP: Previsão do tempo

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 um kit *CanSat* ou *CubeSat* e os vários LEDs RGB (*Red*, *Green* e *Blue*) presentes nos kits. 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. Assim, seu pequeno satélite também pode ser um dispositivo 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: Serviços Web - Previsão de tempo, índice ultravioleta e ondas em XML - CPTEC/INPE.

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 polo Brasileiro de ciência e tecnologia[1]. 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>2022-02-19</atualizacao>
  <previsao>
    <dia>2022-02-20</dia>
    <tempo>pn</tempo>
    <maxima>28</maxima>
```

[1] Saiba mais sobre São Carlos - SP em https://www.reportsancahub.com.br/

```
    <minima>18</minima>
    <iuv>13.0</iuv>
  </previsao>
  <previsao>
    <dia>2022-02-21</dia>
    <tempo>ps</tempo>
    <maxima>29</maxima>
    <minima>17</minima>
    <iuv>13.0</iuv>
  </previsao>
  <previsao>
    <dia>2022-02-22</dia>
    <tempo>pn</tempo>
    <maxima>30</maxima>
    <minima>16</minima>
    <iuv>13.0</iuv>
  </previsao>
  <previsao>
    <dia>2022-02-23</dia>
    <tempo>n</tempo>
    <maxima>30</maxima>
    <minima>17</minima>
    <iuv>13.0</iuv>
  </previsao>
</cidade>
```

Fonte: CPTEC/INPE (Consulta em Fev/2022)

Dentre as várias informações disponibilizadas, o campo **tempo** oferece a informação necessária para implementar o “Galinho do Tempo”. Note que, no formato XML, cada informação encontra-se entre marcadores especiais. Por exemplo, a informação sobre o tempo encontra-se sempre delimitada entre <tempo> e </tempo>. No exemplo acima, o dado mais recente informa: **`<tempo>pn</tempo>`**. A documentação deste serviço web do INPE/MCTI também descreve as possibilidades de informações sobre o tempo. A tabela a seguir mostra algumas destas siglas. Consulte o link Serviços Web - Previsão de tempo, índice ultravioleta e ondas em XML - CPTEC/INPE para lista completa de siglas e descrições.

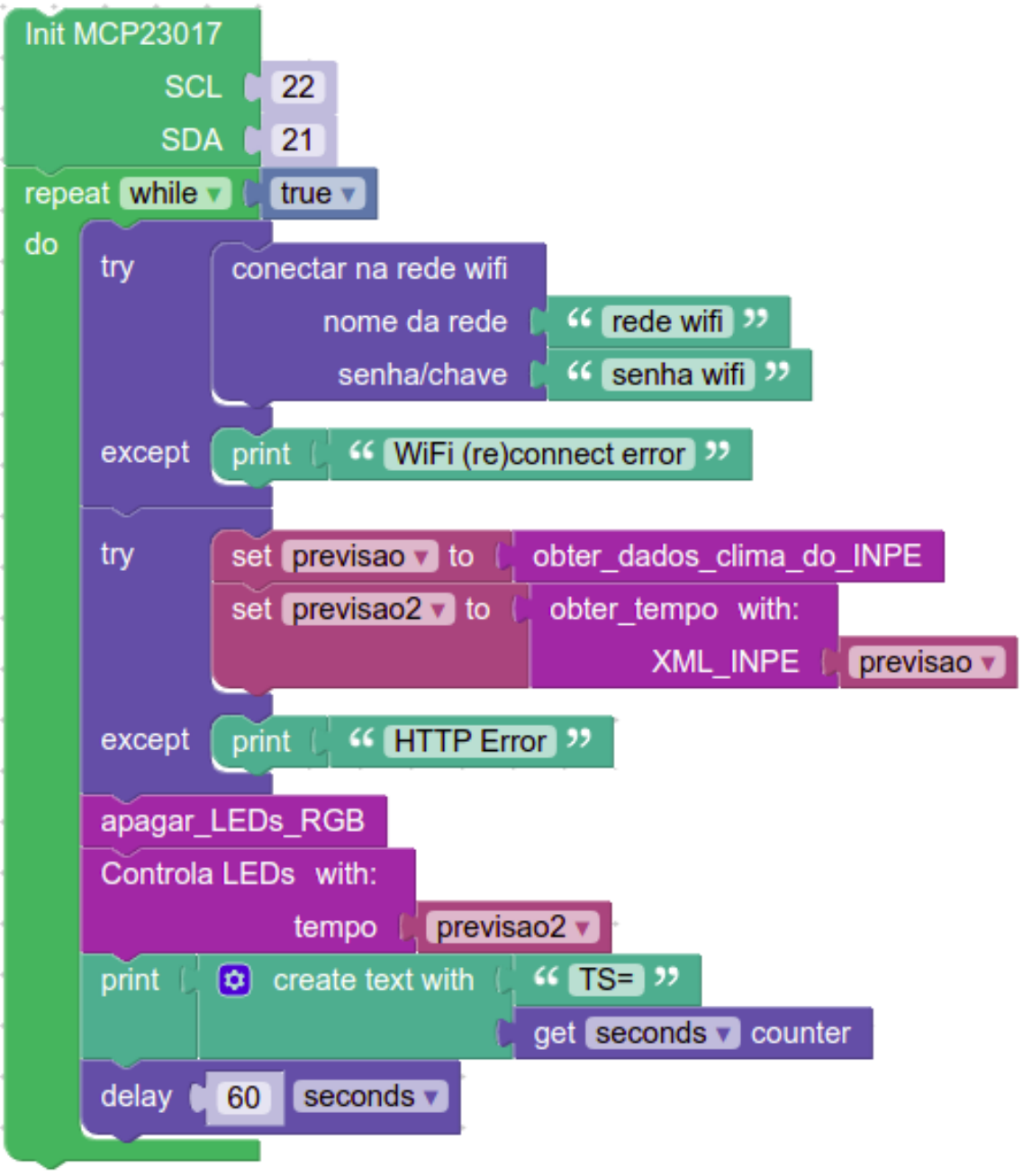
Sigla	Descrição
pn	Parcialmente Nublado
ci	Chuvas Isoladas
c	Chuva
pc	Pancadas de Chuva
t	Tempestade
cl	Céu Claro

Com base nestas informações, já é possível implementar o “*CubeSat* / *CanSat* Galinho do Tempo IoT”. Controlando pinos dos LEDs RGB com o módulo MCRP23017 relacionados com cada cor do LED, é possível acender o LED na cor vermelha, verde, azul, ou combinações destas cores.

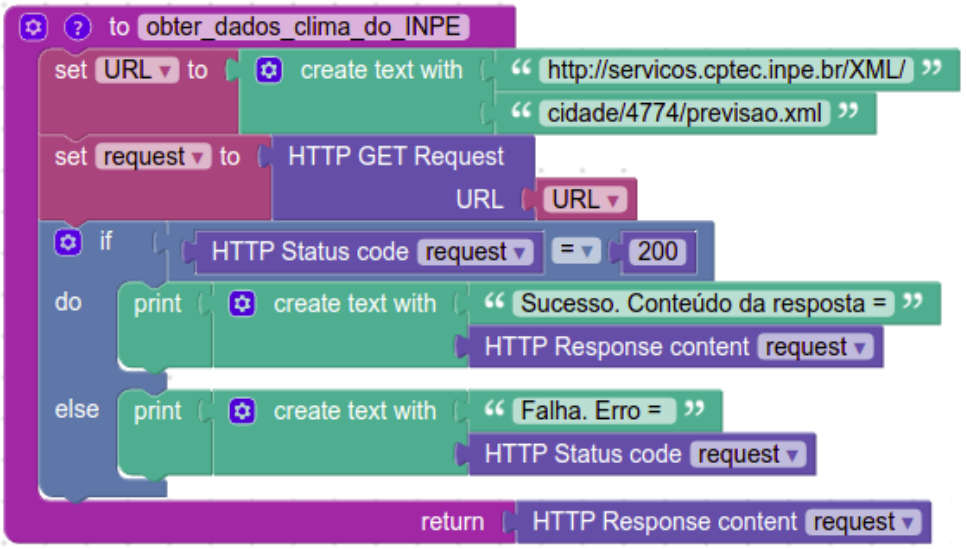
Programa:

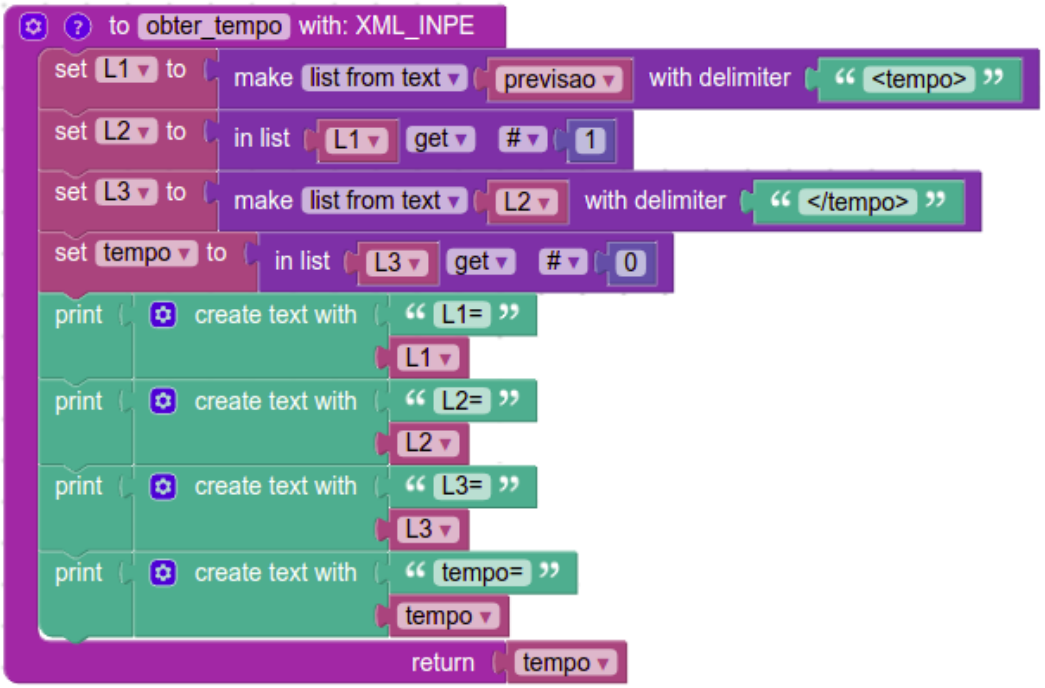
As próximas páginas apresentam o programa para a aplicação de exibição da previsão do tempo usando os LEDs. Por conveniência, o programa foi feito de forma modular, dividindo as atividades em funções (sub-rotinas). A primeira figura apresenta o programa principal, que tem uma estrutura de repetição infinita que executa as funções ali incluídas a cada 60 segundos. Quatro funções adicionais de suporte foram criadas, exibidas nas páginas seguintes: **obter_dados_clima_do_INPE**, que acessa o serviço web do INPE e baixa os dados meteorológicos. Em seguida, define-se a função **obter_tempo**, que interpreta o resultado disponibilizado pelo INPE e obtém apenas o dado de previsão do tempo desejado. Em seguida são apresentadas as funções **apagar_LEDs_RGB** e **Controla LEDs**, que permitem uma forma mais facilitada para acender e apagar os LEDs RGB dos kits CanSat e CubeSat.

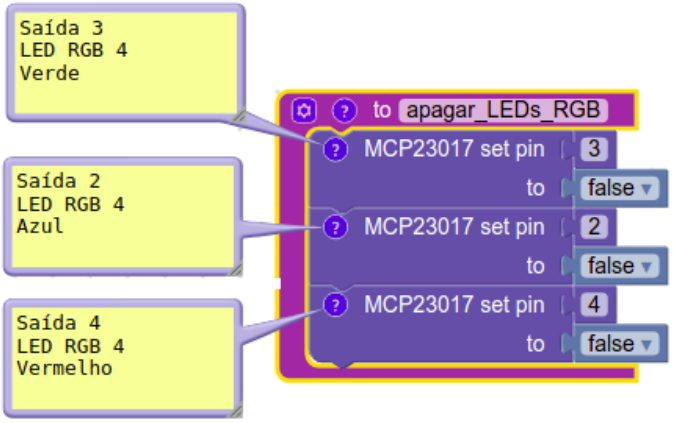
Programa principal:

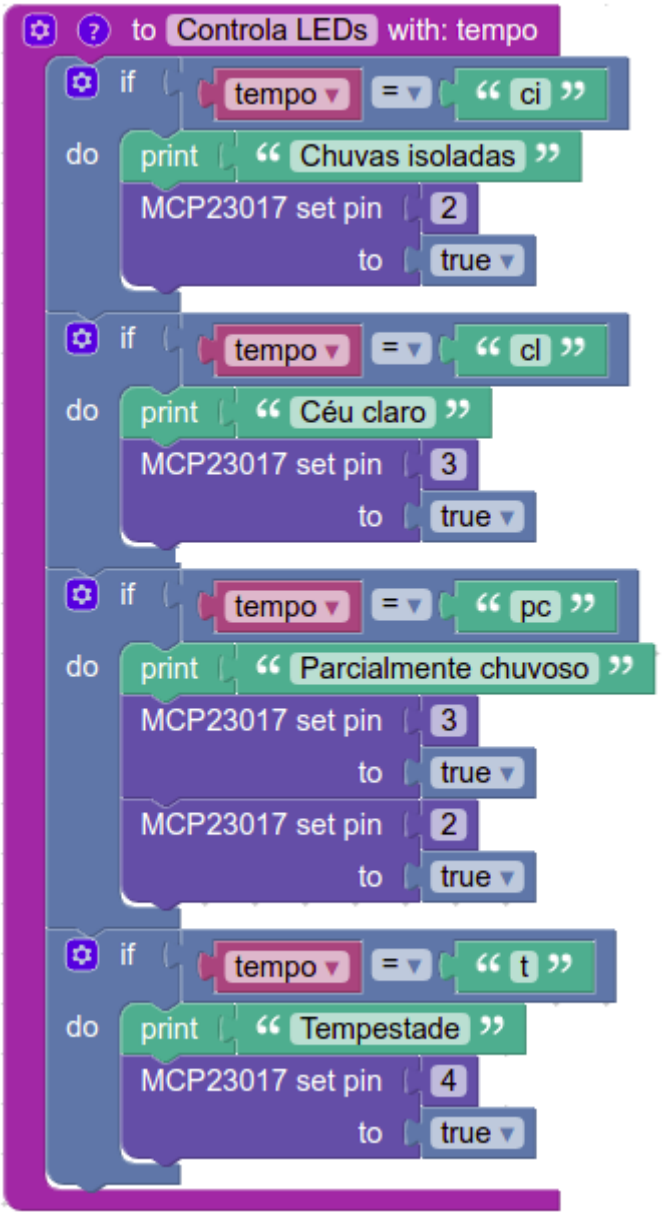


Funções:


to obter_dados_clima_do_INPE
set URL to create text with "http://servicos.cptec.inpe.br/XML/" "cidade/4774/previsao.xml"
set request to HTTP GET Request URL URL
if HTTP Status code request = 200
do print create text with "Sucesso. Conteúdo da resposta =" HTTP Response content request
else print create text with "Falha. Erro =" HTTP Status code request
return HTTP Response content request


to obter_tempo with: XML_INPE
set L1 to make list from text previsao with delimiter "<tempo>"
set L2 to in list L1 get # 1
set L3 to make list from text L2 with delimiter "</tempo>"
set tempo to in list L3 get # 0
print create text with "L1=" L1
print create text with "L2=" L2
print create text with "L3=" L3
print create text with "tempo=" tempo
return tempo


Saída 3
LED RGB 4
Verde
Saída 2
LED RGB 4
Azul
Saída 4
LED RGB 4
Vermelho
to apagar_LEDs_RGB
MCP23017 set pin 3 to false
MCP23017 set pin 2 to false
MCP23017 set pin 4 to false



Link direto para o programa: [BIPES Project]

Como mencionado, o programa principal apresentado possui um laço de repetição infinito com espera de um minuto. Assim, o sistema consulta o servidor do INPE/MCTI a cada um minuto e atualiza as cores dos LEDs 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.

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
Chuvas isoladas
```

Cliente HTTP: POST

Na última seção discutimos o uso das requisições HTTP do tipo GET para obter informações do tempo. As requisições GET também permite enviar dados para o servidor, porém são limitadas em relação ao formato e quantidade de dados que podem ser enviados. Outro tipo de requisição HTTP muito usada é a requisição POST, que permite enviar grandes quantias de dados para o servidor através de uma requisição HTTP. Quando você envia arquivos através de páginas web, fotos ou outros arquivos, normalmente uma requisição HTTP POST está sendo usada.

Uma requisição HTTP POST pode enviar dados em diversos formatos, incluindo dados brutos em formato binário ou compactados. Uma forma padronizada que vem sendo muito utilizada recentemente para transmissão de dados via HTTP é o formato JSON: *JavaScript Object Notation*. O padrão JSON oferece uma forma de armazenar e transportar informações em formato texto de forma padronizada, além de ser fácil de entender e auto-explicativo. Para maiores detalhes sobre JSON, consulte: JSON Introduction.

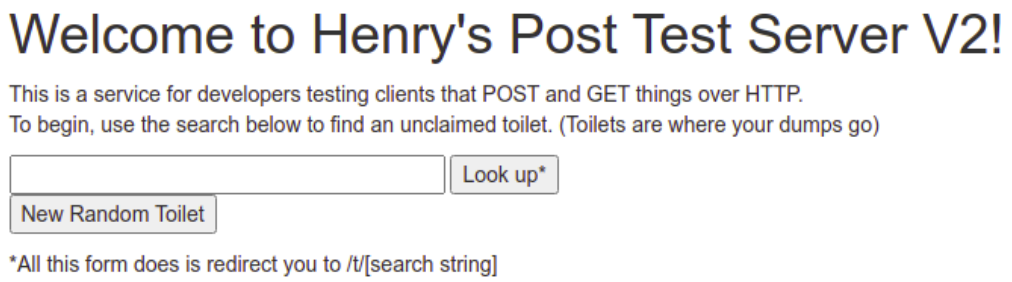
A lista abaixo apresenta alguns exemplos de mensagens JSON:

```
Exemplo 1:
'{"nome":"João", "idade":30, "kit":10}'

Exemplo 2:
'{"id":10, "temperatura":30, "umidade":75}'
```

Dessa forma, é possível elaborar um programa, usando o BIPES, para enviar dados via JSON usando o bloco **Make HTTP Post Request (JSON)**. Entretanto, ainda falta um aspecto: o servidor. É necessário um servidor HTTP que receberá a requisição. Assim, em diversas situações e projetos, o servidor pode já estar definido e configurado, e seu programa somente precisa realizar o POST. Para desenvolvimento e testes, existem vários sites na Internet que recebem requisições HTTP POST e as exibem, para fins de testes de sistemas que realizam POST.

Um exemplo é o *Henry's Post Test Server* (Post Test Server V2), que chamaremos de PTS (*Post Test Server*). O PTS permite definir sessões de teste, enviar requisições HTTP POST para estas sessões, e verificar o conteúdo / dados das requisições realizadas com sucesso. Para usar o PTS, o primeiro passo é acessar sua página principal (Post Test Server V2) e definir uma sessão (chamada de *toilet*, devido ao fato do teste ser considerado inútil).

A próxima figura mostra esta etapa:



Você pode criar um *toilet* com uma palavra, ou criar um *toilet* aleatório clicando em *New Random Toilet*. Use o botão *Look Up* e certifique-se de que o *toilet* que você escolheu está vazio. Por exemplo, vamos verificar por bipes-test:

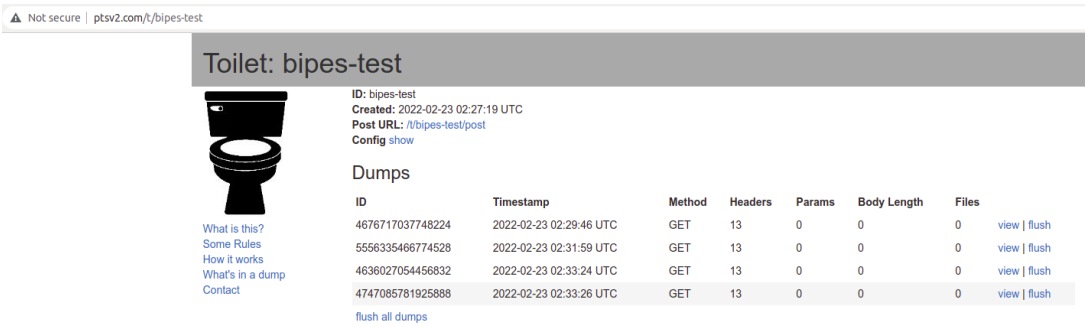
Verifica-se que esta sessão está vazia (Mensagem: *There are no dumps in this toilet*), então esta é uma boa sessão para ser usada:

A seguir, verificamos o **Post URL**, que é **/t/bipes-test/post.** Assim, podemos realizar requisições HTTP POST para esta URL:

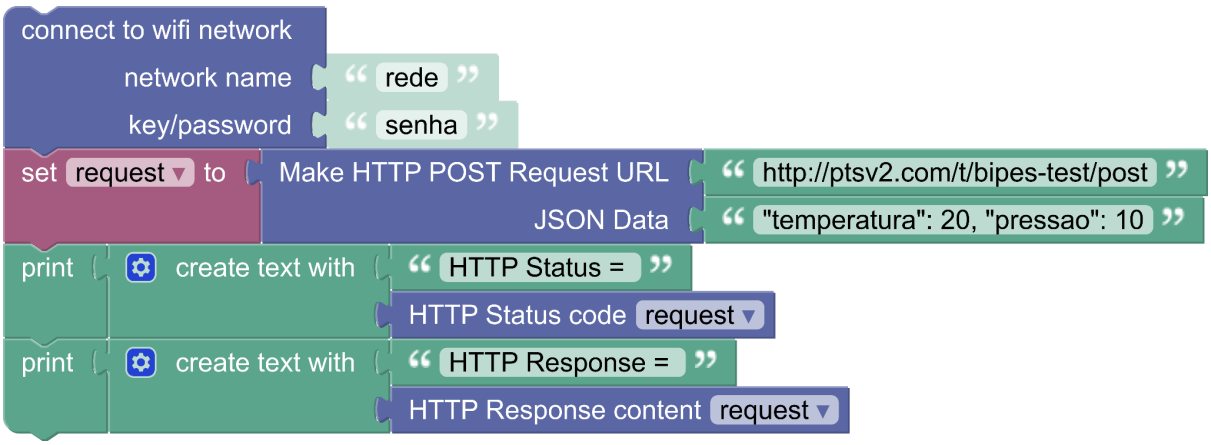
http://ptsv2.com/t/bipes-test/post

Ao clicar na URL acima com qualquer navegador Web, você verá a mensagem: "*Thank you for this dump. I hope you have a lovely day!*". Esta é a resposta enviada para a cliente HTTP que fizer requisições HTTP POST para este endereço. Você pode alterar esta mensagem acessando a URL deste *toilet* (http://ptsv2.com/t/bipes-test) e clicando em **Config show**.

Cada POST realizado é salvo e pode ser visualizado pela URL http://ptsv2.com/t/bipes-test. Por exemplo, após diversas requisições HTTP POST, a URL mencionada vai exibir:



Agora que temos a possibilidade de realizar testes com a ferramenta PTS, é possível realizar requisições HTTP POST, enviando dados em formato JSON, a partir do computador de bordo do nosso satélite. Por exemplo:



Considerando que o kit esteja conectado na Internet e a mensagem de resposta do POST tenha sido ajustada para "Thank you ESP32", ao realizar a requisição acima, o **console** exibirá "HTTP Status = 200 seguido de HTTP Response = b'Thank you ESP32'".

O resultado deste POST HTTP estará disponível como um *dump* da sua sessão (*toilet*) do servidor de testes de POST (PST):

Dumps

ID	Timestamp	Method	Headers	Params	Body Length	Files	
4676717037748224	2022-02-23 02:29:46 UTC	GET	13	0	0	0	view | flush
5556335466774528	2022-02-23 02:31:59 UTC	GET	13	0	0	0	view | flush
4636027054456832	2022-02-23 02:33:24 UTC	GET	13	0	0	0	view | flush
4747085781925888	2022-02-23 02:33:26 UTC	GET	13	0	0	0	view | flush
5494418144493568	2022-02-23 02:35:35 UTC	GET	13	0	0	0	view | flush
6094938726989824	2022-02-23 02:35:54 UTC	POST	8	0	34	0	view | flush

flush all dumps

Note a última linha, com método POST (vale lembrar que sempre que clicamos em um link o método da requisição é GET, ficando também registrado). Ao clicar em ***view*****,** é possível ver os detalhes deste *dump:*

ptsv2.com/t/bipes-test/d/6094938726989824

Dump View

What is this?
Some Rules
How it works
What's in a dump
Contact

Dump 6094938726989824

Parent Toilet
View this as: json or raw text
See what's in a dump for info on how these fields are populated.

Details

Posted: 2022-02-23 02:35:54 UTC
Method: POST
Source: 127.0.0.1:46842

Headers

Header	Values
Content-Length	34
Content-Type	application/json
Forwarded	for="187.119.231.163";proto=http
Traceparent	00-d26968c838d18a2bb08b2790697c2c88-350b203be6dbb711-00
X-Cloud-Trace-Context	d26968c838d18a2bb08b2790697c2c88/3822184150402316049
X-Forwarded-For	187.119.231.163, 169.254.1.1
X-Forwarded-Proto	http
X-Google-Apps-Metadata	domain=gmail.com,host=ptsv2.com

Parameters

No Parameters.

Post Body

```
{"pressao": 10, "temperatura": 20}
```

Multipart Files

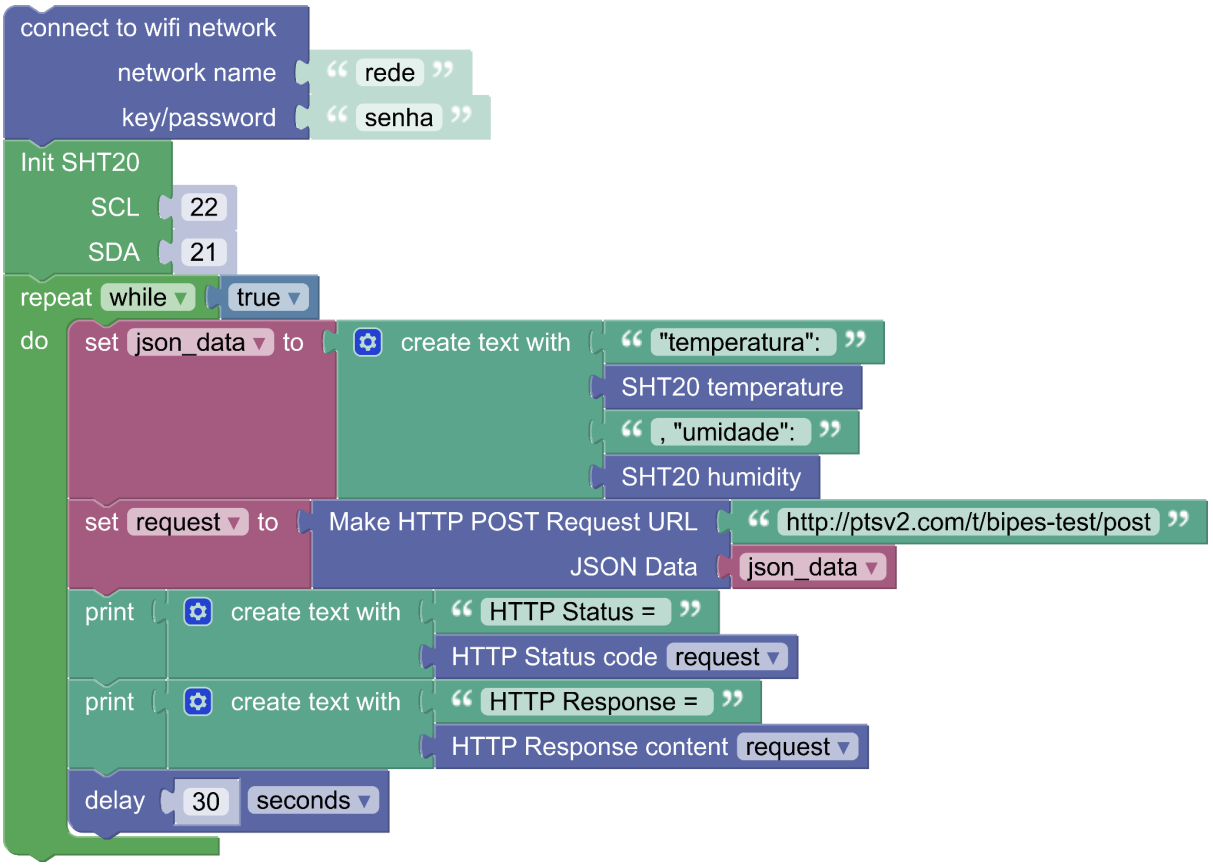
No Files

Multipart Values

No Multipart Values

Verifique que o Content-Type é **application/json** e o **Post Body** contém os dados enviados pelo bloco POST do BIPES. O próximo programa mostra um programa que envia a temperatura e umidade, via HTTP POST, para o PTS.

Programa:



Exemplo de lista de *dumps* do PTS:

5092184122458112	2022-02-23 02:44:02 UTC	POST	8	0	48	0	view | flush
5562932536541184	2022-02-23 02:44:04 UTC	POST	8	0	48	0	view | flush
5173397726167040	2022-02-23 02:44:06 UTC	POST	8	0	48	0	view | flush
6688832443383808	2022-02-23 02:44:08 UTC	POST	8	0	48	0	view | flush
4542585745965056	2022-02-23 02:44:11 UTC	POST	8	0	48	0	view | flush

Exemplo de resultado (ao clicar em *view* para um dos *dumps*):

Parameters

No Parameters.

Post Body

{'"temperatura": 25.83582, "umidade": 51.20226'}

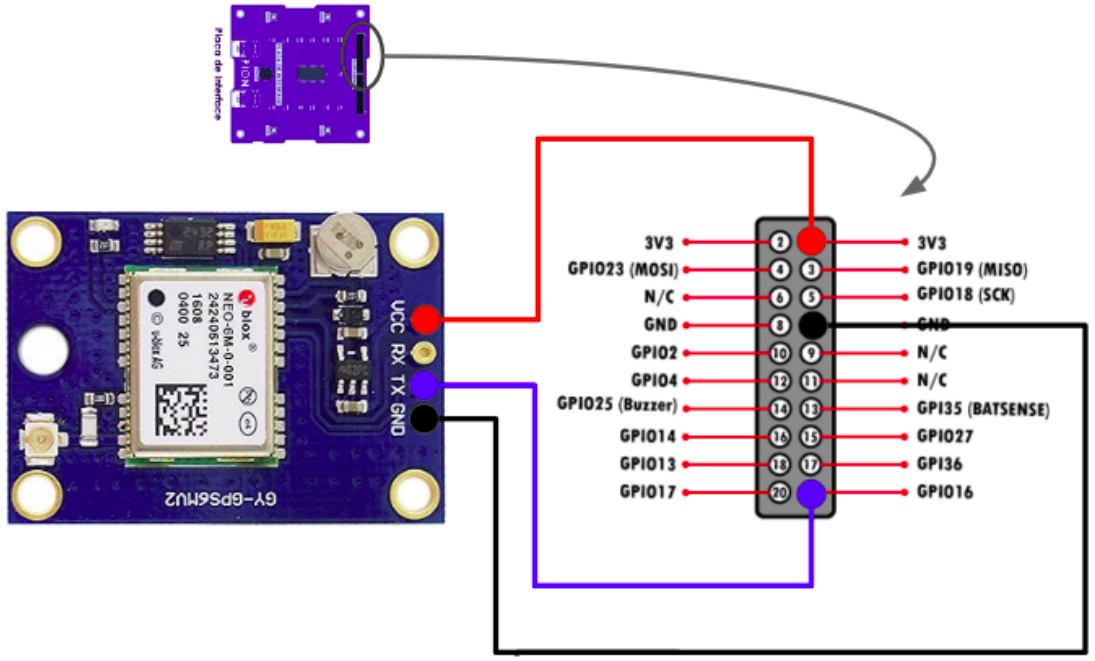
Finalmente, note que o uso das requisições HTTP POST com JSON podem ser úteis em vários outros cenários, como, por exemplo, para envio de dados dos CanSats e CubeSats para a sonda Zenith. Link para modelo básico: [BIPES Project]

Sensores externos

Os conectores modulares dos kits permitem incluir diversos sensores e/ou atuadores externos, ou outros componentes eletrônicos. Discutiremos, a seguir, o uso de um receptor de posicionamento global (GPS) e de uma câmera (ESP32-CAM).

GPS

A figura a seguir ilustra o uso de um GPS uBlox genérico, modelo NEO-6M, facilmente encontrado no mercado brasileiro. Outras marcas e modelos compatíveis com o padrão NMEA-183 também podem ser usados. Conexão elétrica (CubeSat):

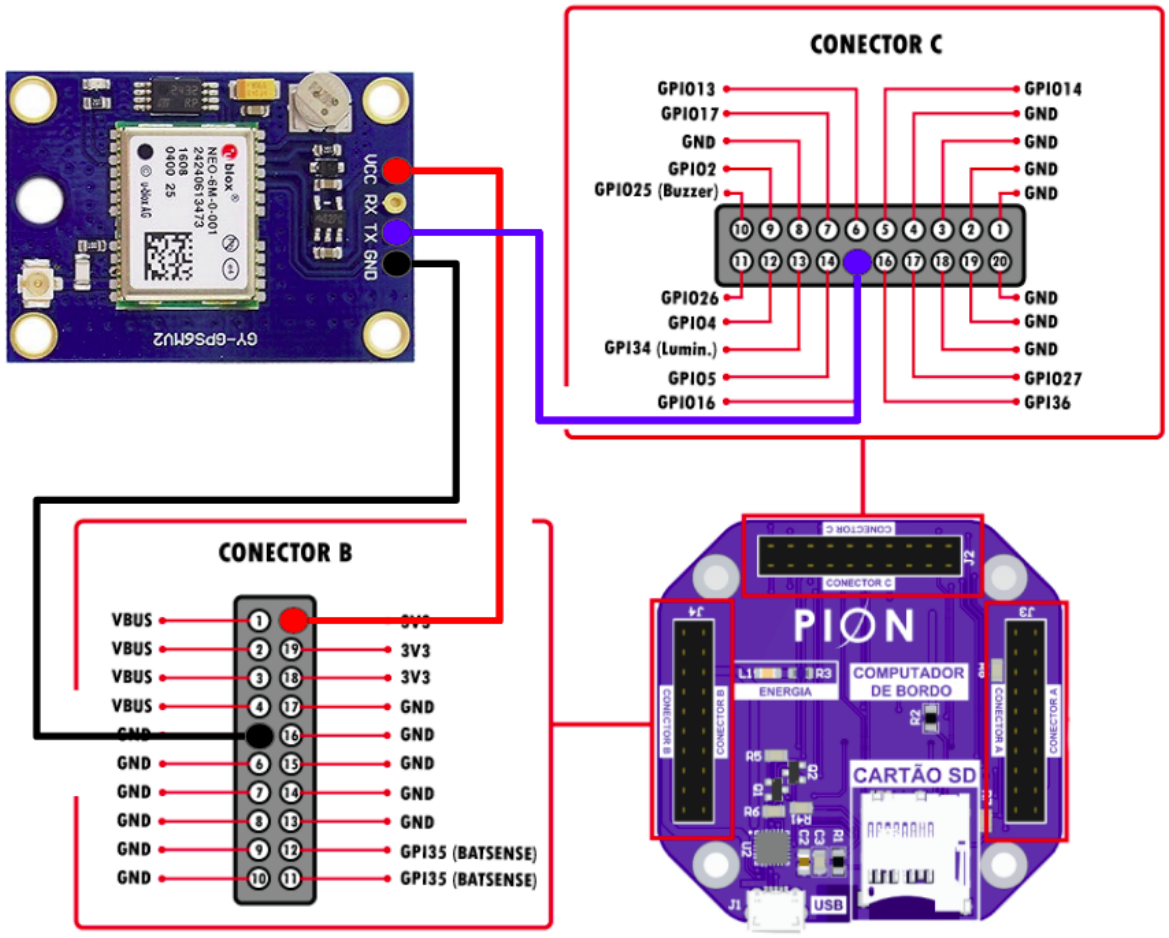


Conexões:

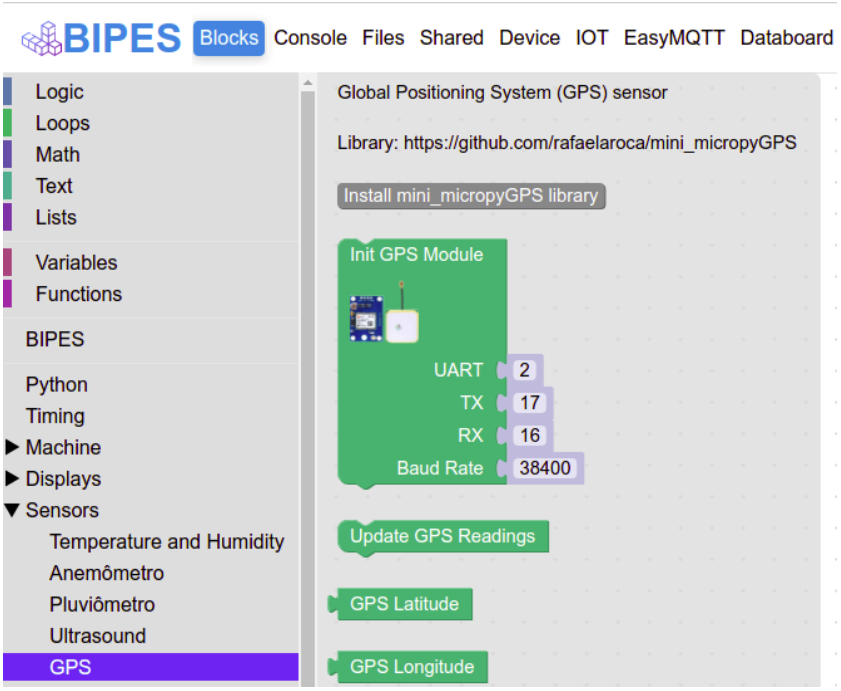
GPS	CanSat/CubeSat
Pino **GND** do GPS	**GND** do Satélite
VCC do GPS	**3v3** do Satélite
TX do GPS	**GPIO16** do kit

As imagens dos componentes e conexões não estão em escala. Note que as conexões podem variar segundo o modelo de GPS. Além disso, as conexões estão mostradas na placa de interface, mas podem ser feitas também no computador de bordo.

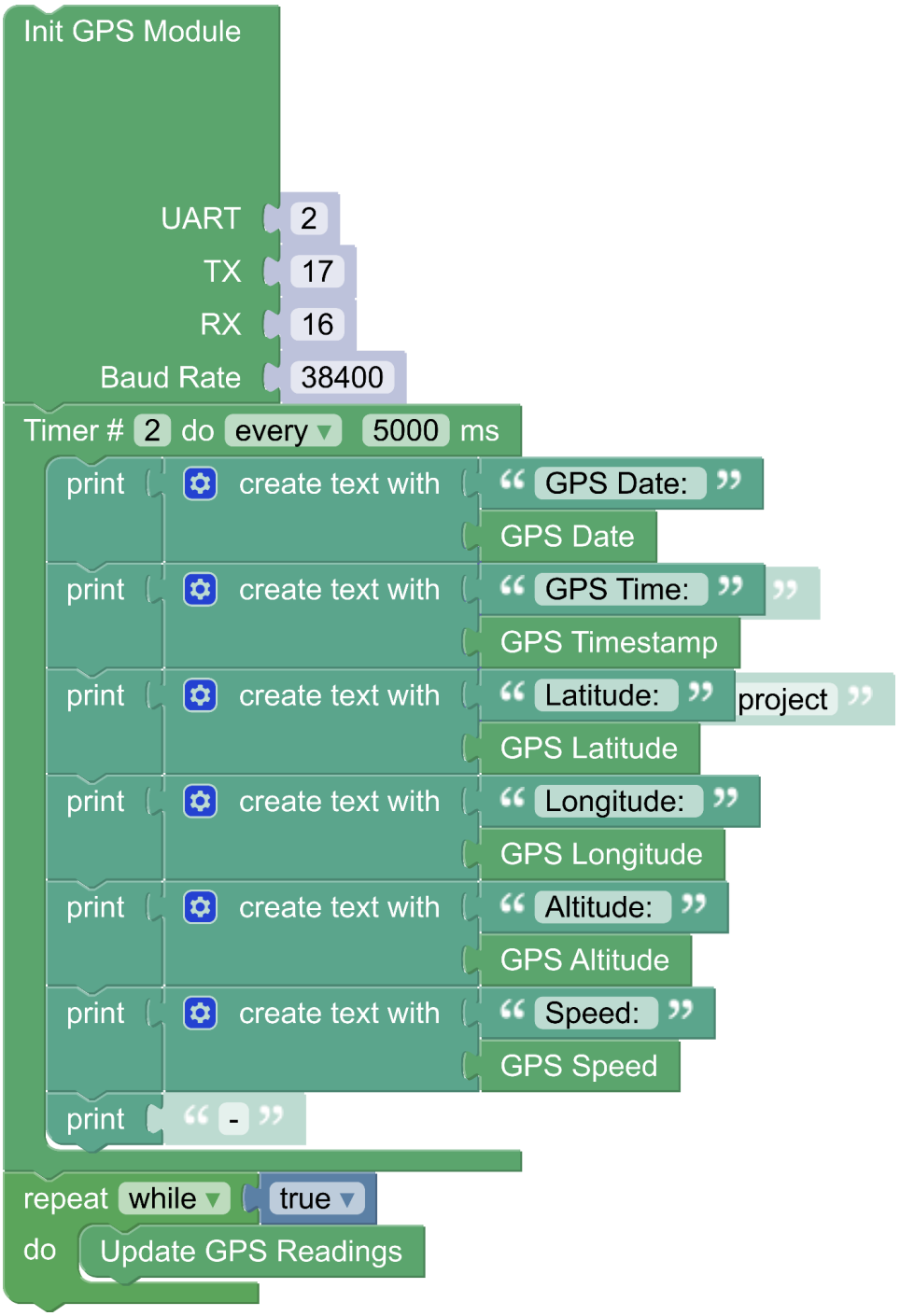
Conexão elétrica (CanSat):



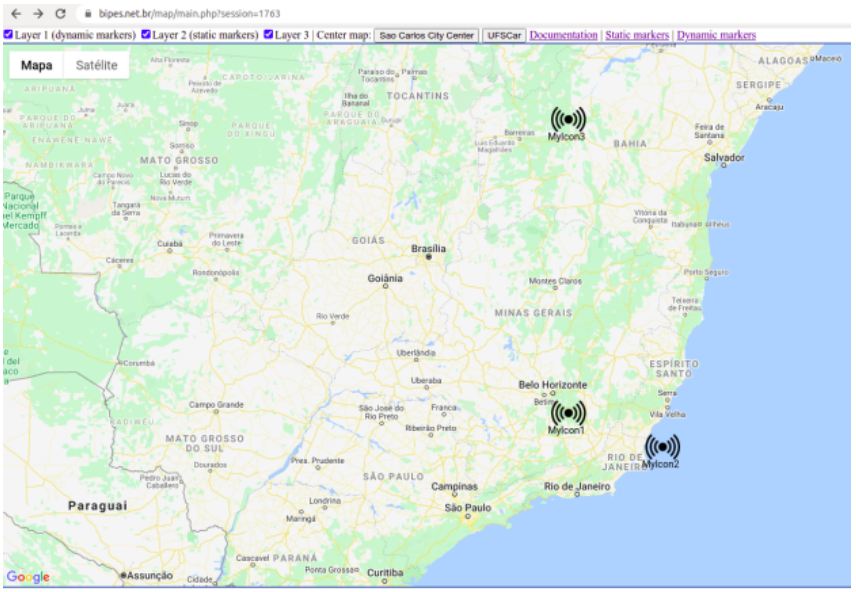
Após as conexões, instale a biblioteca (*Sensors* → GPS → Install library):



Depois construa o programa. O próximo exemplo mostra os dados básicos obtidos a partir do GPS. Note que, dependendo da marca e modelo de seu GPS, a velocidade, em bits por segundo, da porta serial pode diferir da utilizada no exemplo. Verifique o manual / *datasheet* do seu GPS para mais informações. Velocidades típicas de receptores GPS são: 4800 bps, 19200 bps, 38400 bps



Mapas: BIPES Maps



O BIPES também pode ajudar a plotar a posição obtida por um GPS em um mapa, que pode ser compartilhado via web, de várias formas: uma forma é usando o *widget* Google Maps na aba IOT e outra através do BIPES *Maps*, disponível em: https://bipes.net.br/map/main.php, com instruções de uso em: https://bipes.net.br/map/doc.html.

Já discutimos vários exemplos de uso de requisições HTTP para troca de dados. O BIPES Maps também utiliza requisições HTTP para envio de dados e gerenciamento de sessões. O uso do BIPES Maps é ilustrado a seguir através de um exemplo.

Primeiro, defina um ID de sessão para seu projeto e certifique-se de que sua sessão não está em uso. Para tanto, acesse:

https://bipes.net.br/map/main.php?session=X

Onde X pode ser qualquer número inteiro que será sua sessão. Verifique se o mapa está vazio. Estando vazio, você pode usar esta sessão. Por exemplo: 1024.

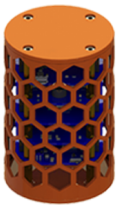
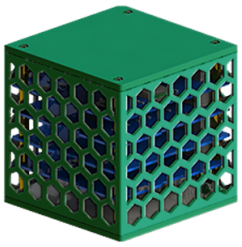
https://bipes.net.br/map/main.php?session=1204

O mapa aberto na URL anterior é constantemente atualizado, em tempo real, com novos marcadores e mudanças de posição de marcadores, pela URL:

https://bipes.net.br/map/crud/index.php/main/dynamic

Ao utilizar a URL acima, lembre-se de sempre utilizar a sessão definida para este projeto (1024 no nosso exemplo).

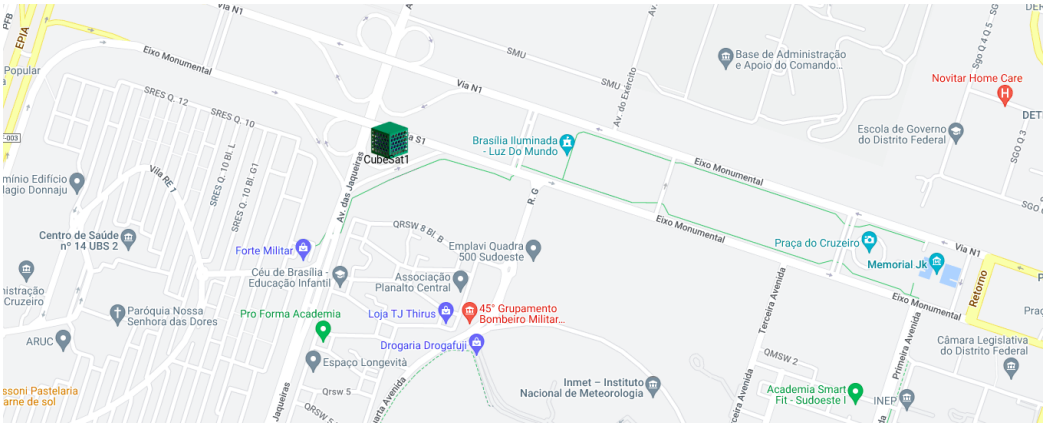
Além disso, precisaremos endereços web para ícones. Aqui temos 2 exemplos, mas você pode usar qualquer endereço de uma figura que queira incluir no mapa:

Imagem	URL
	https://obsat.org.br/inscricoes/n1.png
	https://obsat.org.br/inscricoes/n2.png

Conhecendo a latitude e longitude do marcador desejado, já seria possível adicionar um novo marcador neste mapa. Por exemplo, vamos incluir um marcador em Brasília-DF através do acesso à seguinte URL:

https://bipes.net.br/map/addMarker.php?name=CubeSat1&lat=-15.7801&long=-47.9292&session=1024&info=Teste%20CubeSat&icon=https://obsat.org.br/inscricoes/n2.png

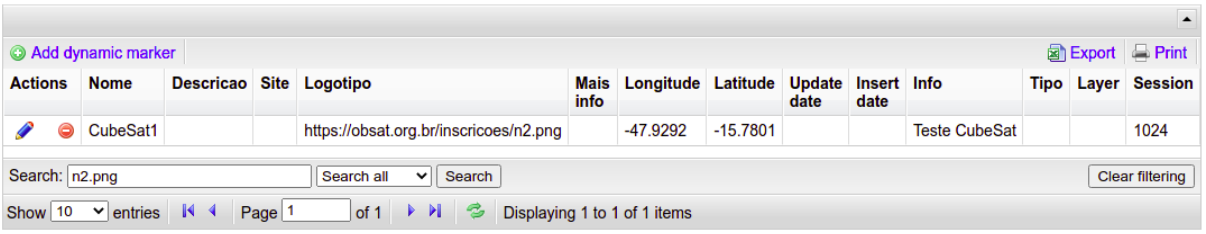
Ao acessar a URL acima, automaticamente o mapa será atualizado e o ícone de um CubeSat, conforme especificado, aparecerá no mapa:



Você pode adicionar quantos marcadores quiser, com diferentes posições, ícones e textos. Na sequência, é possível visualizar e gerenciar os marcadores, pelo endereço web:

https://bipes.net.br/map/crud/index.php/main/dynamic

Ao acessar o link acima, é possível visualizar e editar dados deste marcador:



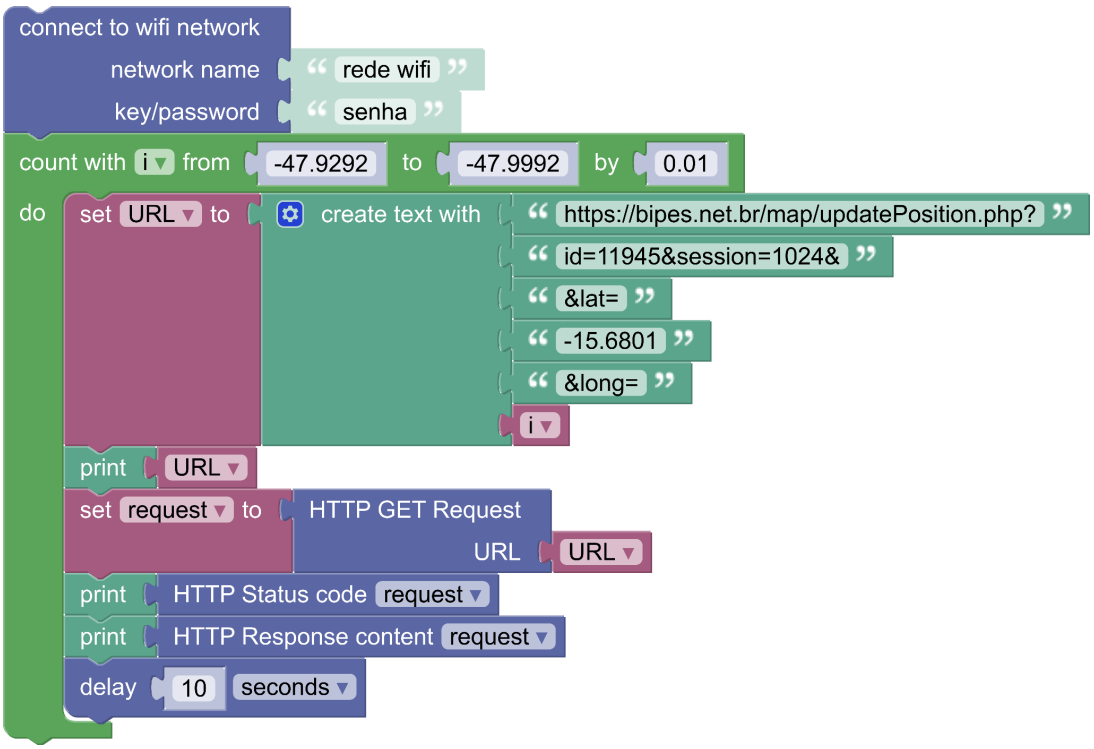
Clique no símbolo de lápis (edição do registro), e repare que uma página com endereço no formato a seguir será mostrada:

https://bipes.net.br/map/crud/index.php/main/dynamic/edit/11945

Onde 11945 é o ID do marcador. Anote este ID, pois o utilizaremos a seguir para atualizar a sua posição automaticamente. Por exemplo:

https://bipes.net.br/map/updatePosition.php?id=11945&lat=-15.6801&long=-47.9292&session=1024

Ao acessar a URL acima, note que o marcador se movimenta no mapa. Poderíamos fazer um programa que move o satélite automaticamente no mapa! Por exemplo:

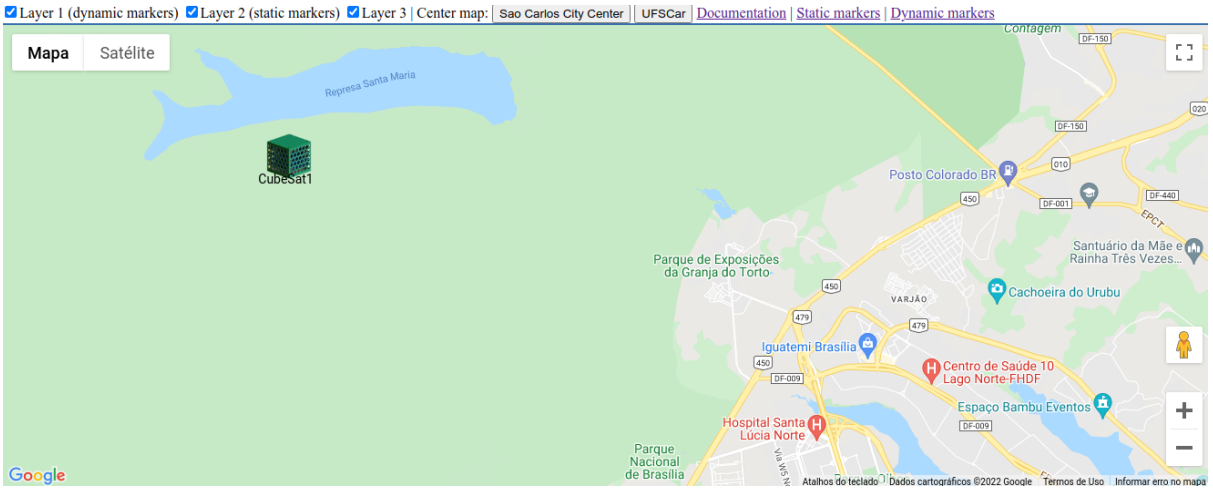


Você poderá acompanhar cada requisição HTTP e o resultado pelo mapa.

Console

```
===
https://bipes.net.br/map/updatePosition.php?id=11945&session=1024&&lat=-15.6801&long=-47.9292
200
b'Session = 1024 <br>IP = 187.119.231.163 <BR><BR>R: \n\n'
https://bipes.net.br/map/updatePosition.php?id=11945&session=1024&&lat=-15.6801&long=-47.9392
200
b'Session = 1024 <br>IP = 187.119.231.163 <BR><BR>R: \n\n'
https://bipes.net.br/map/updatePosition.php?id=11945&session=1024&&lat=-15.6801&long=-47.94919
200
b'Session = 1024 <br>IP = 187.119.231.163 <BR><BR>R: \n\n'
https://bipes.net.br/map/updatePosition.php?id=11945&session=1024&&lat=-15.6801&long=-47.95919
200
b'Session = 1024 <br>IP = 187.119.231.163 <BR><BR>R: \n\n'
https://bipes.net.br/map/updatePosition.php?id=11945&session=1024&&lat=-15.6801&long=-47.96919
200
b'Session = 1024 <br>IP = 187.119.238.91 <BR><BR>R: \n\n'
https://bipes.net.br/map/updatePosition.php?id=11945&session=1024&&lat=-15.6801&long=-47.97919
200
b'Session = 1024 <br>IP = 187.119.238.91 <BR><BR>R: \n\n'
https://bipes.net.br/map/updatePosition.php?id=11945&session=1024&&lat=-15.6801&long=-47.98919
200
b'Session = 1024 <br>IP = 187.119.238.91 <BR><BR>R: \n\n'
https://bipes.net.br/map/updatePosition.php?id=11945&session=1024&&lat=-15.6801&long=-47.99919
200
b'Session = 1024 <br>IP = 187.119.238.91 <BR><BR>R: \n\n'
>>>
```

Mapa (posição final):



Finalmente, é possível integrar dois assuntos já apresentados: o GPS e o mapa. Assim, um programa pode receber dados do GPS e realizar requisições HTTP para o BIPES Maps, permitindo plotar, em tempo real, a posição de um dispositivo. Assim, seria possível rastrear um CanSat, CubeSat, ou mesmo um veículo.

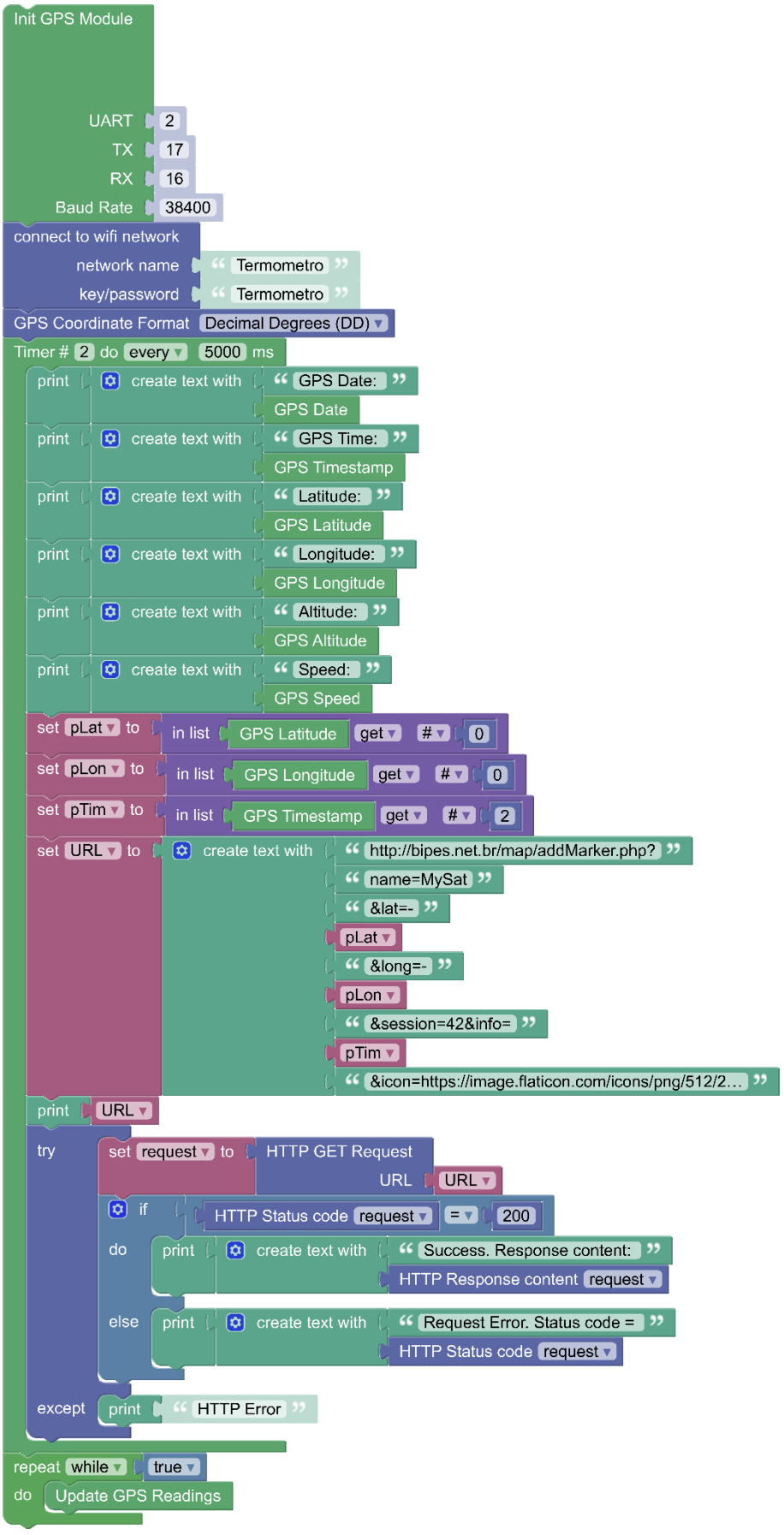
Para implementar este programa pode-se optar pelos serviços web **addMarker** ou **updatePosition**. O updatePostion vai proporcionar uma visão de movimento no mapa, enquanto o uso repetitivo do addMarker vai gerar um efeito de rastro, plotando no mapa cada posição reportada via HTTP para o BIPES Maps.

O seguinte programa mostra um exemplo de um sistema que envia e exibe dados históricos de movimento de um veículo plotados sob o mapa.

Link direto para o programa:

https://bipes.net.br/beta2/ui/#fayx52

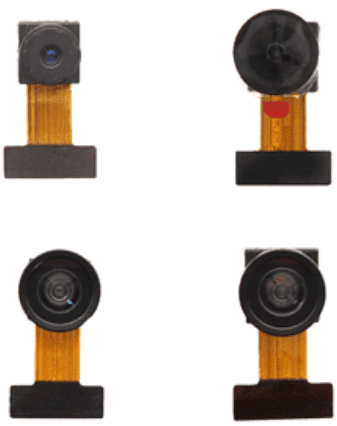
Programa:


Init GPS Module
UART 2
TX 17
RX 16
Baud Rate 38400
connect to wifi network
network name Termometro
key/password Termometro
GPS Coordinate Format Decimal Degrees (DD)
Timer # 2 do every 5000 ms
print create text with GPS Date: GPS Date
print create text with GPS Time: GPS Timestamp
print create text with Latitude: GPS Latitude
print create text with Longitude: GPS Longitude
print create text with Altitude: GPS Altitude
print create text with Speed: GPS Speed
set pLat to in list GPS Latitude get # 0
set pLon to in list GPS Longitude get # 0
set pTim to in list GPS Timestamp get # 2
set URL to create text with http://bipes.net.br/map/addMarker.php?
name=MySat
&lat=-
pLat
&long=-
pLon
&session=42&info=
pTim
&icon=https://image.flaticon.com/icons/png/512/2…
print URL
try
set request to HTTP GET Request
URL URL
if HTTP Status code request = 200
do print create text with Success. Response content:
HTTP Response content request
else print create text with Request Error. Status code =
HTTP Status code request
except print HTTP Error
repeat while true
do Update GPS Readings

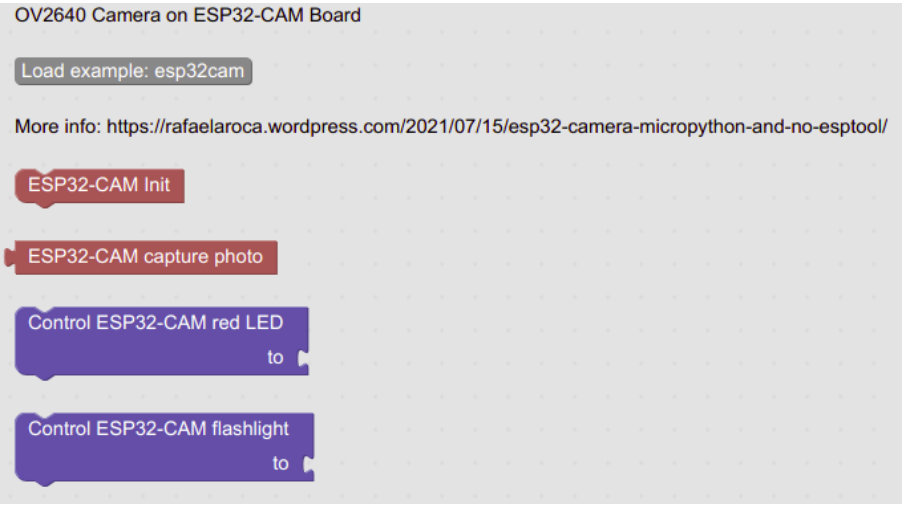
Exemplo de resultado:

Câmera - ESP32-CAM

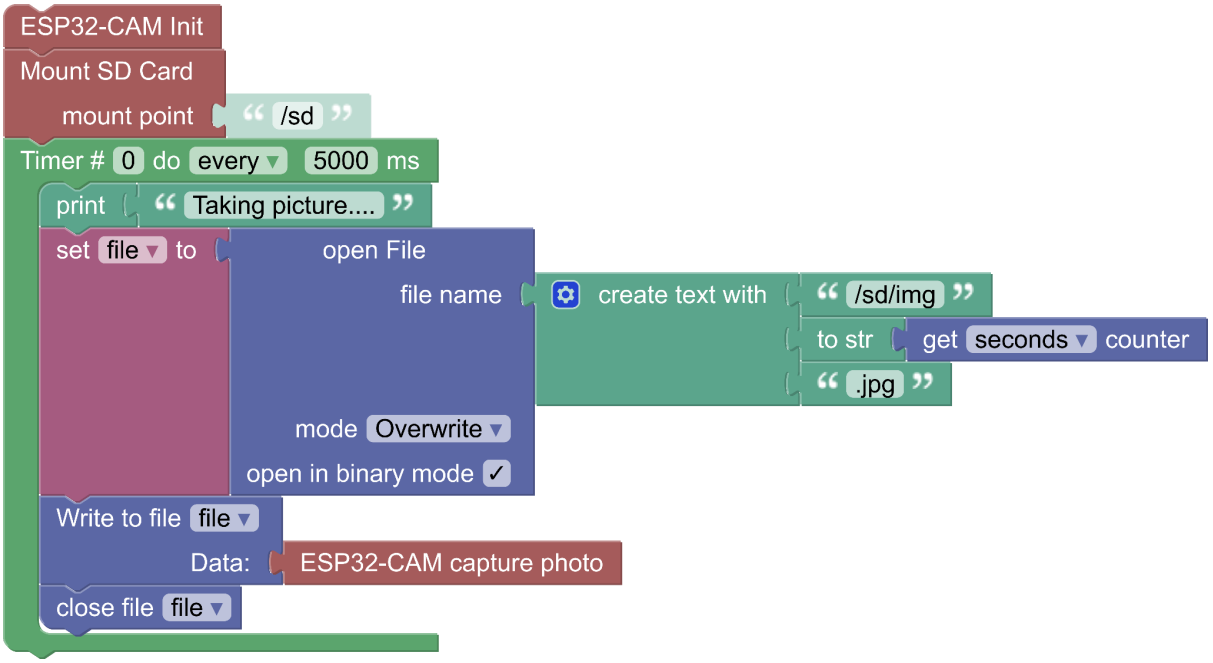
Dependendo de sua aplicação, pode ser importante adicionar uma câmera ao seu CubeSat ou CanSat. Existem diversas opções de módulos de câmeras que podem ser conectadas ao computador de bordo ESP32, através de interface serial UART, ou outros tipos de interface. A figura abaixo mostra possibilidade de uso de um módulo de baixo custo e amplamente disponível no mercado: O ESP32-CAM, mostrado na figura abaixo. Note que este módulo contempla um outro módulo de processamento ESP32, incluindo uma câmera e um cartão de memória. Assim, uma possibilidade é a integração deste módulo ao resto do seu *CanSat* ou *CubeSat* por uma expansão da arquitetura modular do satélite: o módulo ESP32-CAM fica responsável pela captura, armazenamento e até processamento de imagens, podendo se comunicar com o computador de bordo principal através de interfaces UART, SPI, I2C, ou mesmo outra, permitindo, assim, sincronizar ações e eventos.



Para utilizar o módulo ESP32-CAM, é possível utilizar o bloco Câmera do BIPES, disponível em sensors:



A imagem a seguir apresenta um exemplo de programa que tira uma foto a cada 5 segundos, e armazena a foto no cartão de memória, sendo o nome de cada arquivo definido como o número de segundos desde que o módulo foi comprado, seguido de “.jpg”.



Programação em C++

Caso você tenha interesse em se aprofundar mais no uso dos kits, ou prefira programar os kits usando C ou C++, a PION Labs disponibilizou bibliotecas e materiais para programação dos kits. Para saber mais detalhes, consulte a página:

https://github.com/pion-labs/pion-educational-kits/wiki/Software

Possíveis problemas

Eventualmente, alguns problemas podem ocorrer no uso dos kits PION com o BIPES. Algumas possíveis soluções são possíveis:

a. **Não é possível conectar ao kit, via USB**
   Dependendo do seu sistema operacional, pode ser necessário instalar drivers para comunicação Serial USB com o kit, ou ajustar permissões de acesso à porta - especialmente nos sistemas Linux e/ou Mac. Utilize, preferencialmente, o navegador web Google Chrome, com maior compatibilidade com o BIPES.

b. **OSError: Wifi Internal Error**
   Este erro é relacionado com o controle da interface WiFi pelo MicroPython na plataforma ESP32. Caso este erro ocorra, será necessário desligar e ligar o kit. De qualquer forma, planeje seu programa para conectar uma única vez na rede, já que a conexão se mantém sempre ativa.

c. **Brownout detector was triggered**
   Tente utilizar uma outra porta USB, ou outro computador, ou verificar se a carga da bateria do kit está completa. Este erro ocorre quando a porta USB ou *hub* USB não fornece energia adequada. Normalmente ocorre com o uso de alguns *hubs* USB.

Agradecimentos

Os autores agradecem o Ministro Astronauta Marcos Pontes e a Secretária de Articulação e Promoção da Ciência, Christiane Gonçalves Corrêa, idealizadores e entusiastas da Olimpíada Brasileira de Satélites MCTI. Também agradecemos os colegas do MCTI que trabalham com popularização da ciência: Prof. Daniel Lavouras, Profa. Silvana Copsceski e Zeily Teles. Agradecemos também os parceiros que contribuem ativamente na organização da OBSAT MCTI: a EESC-USP, através do projeto Zenith, o Instituto Nacional de Pesquisas Espaciais (INPE) e a Agência Espacial Brasileira (AEB). No âmbito da UFSCar, agradecemos à ProEx pelo apoio no projeto, a ProAd e a FAI-UFSCar, por todo apoio na execução do projeto OBSAT MCTI. Finalmente, também agradecemos o 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).

Maiores informações

Existem várias fontes de informações e dados relacionados com satélites educacionais. Procure por materiais do INPE, que oferecem minicursos, escolas de verão, textos informativos e textos informativos. A Agência Espacial Brasileira (AEB) também possui várias informações relevantes no programa AEB Escola e em sua nova plataforma de capacitação (AEB Escola Virtual), incluindo o curso “Pequenos Satélites Educacionais”, organizado em parceria com a OBSAT MCTI. Além disso, um ebook organizado em parceria com a OBSAT também deve ser disponibilizado em breve: “Pequenos satélites: grandes possibilidades”. O canal ciência em casa MCTI (Ciência em Casa) também oferece opções interessantes, bem como o ebook do BIPES, que pode fornecer informações adicionais sobre o BIPES (https://bipes.net.br/wp/book-livro/). O projeto Zenith (Zenith Aerospace || Aerospace Technology || EESC-USP) da EESC-USP também fornece capacitações e desafios de capacitação na área espacial. Finalmente, o site da OBSAT (OBSat) também possui links para o canal do YouTube da Olimpíada com apresentações, palestras e minicursos.

Programação de *CanSats PION* e *CubeSats PION* por blocos usando BIPES

Na foto, da esquerda para direita: Prof. Daniel Lavouras (MCTI), Secretaria Christiane Correa (MCTI), Ministro Astronauta Marcos Pontes (MCTI), Prof. Rafael Vidal Aroca (UFSCar), Profa. Silvana Copsceski (MCTI) e Calvin Trubiene (PION)
Foto:Neila Rocha (ASCOM/SEAPC/MCTI)

Olimpíada Brasileira de Satélites MCTI

ISBN: 978-65-00-40369-5

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA,
E INOVAÇÕES

PÁTRIA AMADA
BRASIL
GOVERNO FEDERAL