



INTRODUÇÃO À INTERNET DAS COISAS

Ferramentas Práticas



Ph.D. Andouglas Gonçalves da Silva Júnior

Ph.D. Manoel do Bonfim Lins de Aquino

Marcos Fábio Carneiro e Silva

Autor da apostila

Ph.D. Andouglas Gonçalves da Silva Júnior

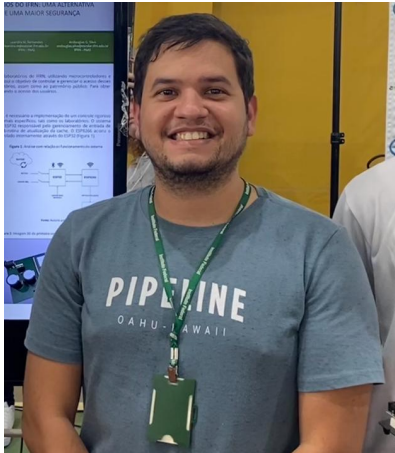
Ph.D. Manoel do Bonfim Lins de Aquino

Instrutor do curso

Larissa Jéssica Alves – Analista de Suporte Pedagógico

Revisão da apostila

Autor



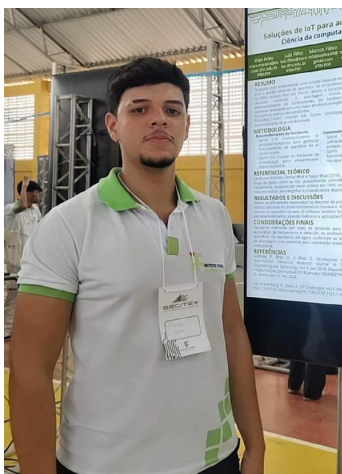
Andouglas Gonçalves da Silva Júnior

Doutor em Engenharia Elétrica e da Computação - UFRN. Mestre em Engenharia Mecatrônica na área de Sistemas Mecatrônicos. Bacharel em Ciências e Tecnologia pela EC&T - Escola de Ciências e Tecnologia - UFRN. Engenheiro Mecatrônico - UFRN. Professor de Ensino Básico, Técnico e Tecnológico no Instituto Federal de Educação Tecnológica do Rio Grande do Norte (IFRN). Integrante da Rede de Laboratórios NatalNet, LAICA e colaborador ISASI-CNR-Itália. Desenvolve projetos na área de Machine Learning, Internet das Coisas e Holografia Digital. Colabora no projeto do N-Boat (Veleiro Robótico Autônomo), principalmente no desenvolvimento de sistemas para monitoramento da qualidade da água e identificação de micropartículas usando holografia digital e IA.



Manoel do Bonfim Lins de Aquino

Possui graduação (2006), mestrado (2008) e doutorado (2022) em Engenharia Elétrica e da Computação, pela Universidade Federal do Rio Grande do Norte - UFRN. Tem experiência na área de projetos de Telecomunicações, atuando na Siemens como engenheiro de Telecomunicações (2008-2010). Sou Professor (2010 - atual) do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte - IFRN, membro do NADIC, Núcleo de Análise de Dados e Inteligência Computacional, onde venho desenvolvendo projetos de P&D nas áreas de desenvolvimento de sistemas, IoT e Inteligência Artificial.



Marcos Fábio Carneiro e Silva

Estudante de informática no Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte. Participou, como bolsista, de projeto de pesquisa voltado à automação em ambiente escolar com IoT (2022 - 2023), Possui experiência em redes de computadores, eletrônica e IoT, com ênfase na utilização de microcontroladores e desenvolvimento de Hardware. Além disso, possui conhecimentos básicos em Python e C++. Atualmente é membro do NADIC (Núcleo de Análise de Dados e Inteligência Computacional) do IFRN.



Sumário

1 Ferramentas Práticas.....	12
1.1 Linguagens de Programação.....	12
1.1.1 C++ para Microcontroladores.....	12
1.1.2 MicroPython.....	15
Bibliotecas Comuns.....	15
1.2 Wokwi.....	17
1.3 Bipes.....	22
1.3.1 Instalação do Micropython no ESP.....	24
1.3.2 Instalação no Raspberry Pi Pico.....	25
1.4 Aplicação Prática.....	26



1 Ferramentas Práticas

No capítulo anterior vimos os principais dispositivos utilizados no desenvolvimento de aplicações IoT. Mais a frente veremos como esses dispositivos trocam informações entre si, fazendo com que os dados sejam transferidos entre eles.

Para mantermos teoria e prática juntos durante todo o nosso curso, iremos introduzir, neste capítulo, as ferramentas práticas que usaremos para criar os projetos. Nesse sentido, iremos utilizar duas poderosas ferramentas de código aberto (*open-source*) que serão utilizadas tanto para simulação como para programação dos microcontroladores: Wokwi e Bipes. Entretanto, antes de partir para prática, precisamos saber como iremos programar os dispositivos.

1.1 Linguagens de Programação

Diferentes linguagens de programação podem ser usadas na programação de microcontroladores. Uma das mais comuns é o C++, muito popularizado por ser o padrão de programação nas placas Arduino. Uma outra linguagem que tem ganhado espaço é o MicroPython, que basicamente consiste em uma versão do python para sistemas embarcados. Também falaremos de um novo conceito chamado no-code que vem sendo muito utilizado e que pode ser uma alternativa interessante para quem está começando na programação de microcontroladores.

1.1.1 C++ para Microcontroladores

Como comentado anteriormente, a linguagem C++ tem sido muito usada na programação de microcontroladores, principalmente porque o Arduino popularizou muito o acesso e, conseqüentemente, a sua utilização em aplicações de sistemas embarcados. Essa popularização aconteceu porque a

própria IDE (Ambiente de Desenvolvimento Integrado) do Arduino, permite a programação no formato .ino, que nada mais é do que a adaptação do C++ para o Arduino e outras placas de desenvolvimento.

Apesar de o nosso curso ser focado mais no desenvolvimento de aplicações em MicroPython, tentaremos sempre trazer exemplos de códigos também em C++ para que o aluno possa escolher a linguagem que melhor se adapta a sua realidade.

Primeiro, lembre-se que um microncontrolador é geralmente usado em aplicações específicas. Isso é importante para entendermos a estrutura básica do programa que será enviado para a placa. A Figura 2.1 mostra a tela inicial da IDE do Arduino v2, com a explicação para alguns componentes da interface.



Fig. 2.1 - Tela da IDE do Arduino v2.

Perceba que no espaço para edição de texto temos um programa escrito em C++ com apenas duas funções que retornam vazio (*void*): **setup** e **loop**. No método **setup**, que é chamado apenas uma vez na inicialização da placa,



inserimos os códigos relacionados a configuração da placa, como por exemplo, inicialização da serial, definição dos pinos de entrada e saída, criação das instâncias das bibliotecas, etc. Já no método **loop** é onde programamos a lógica da nossa aplicação. Por exemplo, se quisermos fazer uma porta permanecer em nível alto por 1 segundo e depois permanecer em nível baixo por 1 segundo, repetindo esse processo, inserimos essa lógica dentro desse método. O código abaixo mostra como ficaria esse programinha em C++, muitas vezes chamado de *blink*, já que podemos ligar um LED a essa porta da placa e vê-lo piscando (faremos isso um pouco mais pra frente).

Exemplo 2.1 - Blink C++



```
// a função setup é executada apenas uma vez quando você  
pressiona o reset ou liga a placa  
void setup() {  
    // inicializa o pino digital chamado LED_BUILTIN como  
uma saída  
    pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// a função Loop roda para sempre  
void Loop() {  
    // Aciona a pino do LED colocando em nível "alto"  
    digitalWrite(LED_BUILTIN, HIGH);  
    delay(1000); // espera por um segundo  
    digitalWrite(LED_BUILTIN, LOW);  
    // Coloca o pin em nível baixo  
    delay(1000); // espera por um segundo  
}
```

Algumas observações sobre a utilização da IDE do Arduino para programação em C++:

1. As bibliotecas básicas, como as de interação com as portas de entrada/saída e a de temporização, já estão pré-configuradas na IDE, não sendo necessário fazer a definição dessas bibliotecas.
2. A instalação e utilização de outras bibliotecas é facilitada pela IDE do Arduino. Isso pode ser feito usando a aba de “Gerenciamento de Bibliotecas”.



3. A palavra `LED_BUILTIN` é reservada a porta que está diretamente ligada ao led embutido na placa do Arduino. Caso queira usar alguma outra porta, basta inserir o número na função `pinMode`.
4. Podemos usar a biblioteca **Serial** para escrever dados na porta **Serial** da placa e visualiza-los na Monitor Serial da IDE. Para isso basta usar `Serial.print(" ");`

1.1.2 MicroPython

Segundo seu site oficial, o “MicroPython é uma implementação eficiente e enxuta da linguagem de programação Python 3 que inclui subconjuntos da biblioteca padrão Python e é otimizado para rodar em microcontroladores e em ambientes restritos”.

No nosso curso, usaremos o MicroPython para programar os microcontroladores nos projetos que iremos desenvolver. Para isso é necessário fazer a sua instalação na placa. Podemos fazer essa instalação usando alguma ferramenta para fazer o upload da imagem diretamente na placa ou podemos usar a própria plataforma Bipes, caso estejamos utilizando o ESP32 ou ESP 8266. Explicaremos como realizar esse procedimento em breve.

Bibliotecas Comuns

Vamos apresentar algumas bibliotecas que são usadas com mais frequência, tanto em aplicações complexas, como em aplicações mais simples. É possível encontrar toda a documentação do MicroPython por meio da url: <https://docs.micropython.org/en/latest/index.html>.

Iniciaremos com a biblioteca **Machine** que reúne as principais formas de interação com a placa, como acesso aos pinos digitais e analógicos, configuração e utilização dos principais protocolos de transmissão de dados (UART, SPI e I2C), temporizador para funções sleep e interrupt, etc. Vamos mostrar a utilização desta biblioteca por meio de alguns exemplos.



Atenção Os comentário nos códigos de exemplos ajudam a melhor compreender o que cada linha desempenha. Além disso, testaremos esses códigos um pouco mais a frente nas plataformas práticas que iremos utilizar.



Uma das funcionalidades mais básicas em aplicações IoT é o acesso às portas de entrada e saída, seja para ler ou para enviar algum dados. Lembre-se que essas portas podem ser digitais ou analógicas a depender do tipo de informação que será lida ou enviada.

O código mostrado no exemplo 2.2 usa a biblioteca machine para atribuir a uma variável chamada *led* a porta digital 5 (GPIO 5) do microcontrolador no modo saída, ou seja, uma porta que pode ser acionada, e depois coloca essa porta na posição alto (HIGH) atribuindo o valor 1.

Exemplo 2.2 - Usando porta digital (MicroPython)



```
#importamos a classe Pin da biblioteca machine
from machine import Pin

#Usa a classe Pin para definir um pino chamado "led"
#Pin(id_da_porta, modo)
# modo: IN (porta de entrada) - Receber dados
# modo: OUT (porta de saída) - Envida dados
led = Pin(5, Pin.OUT)

#Atribui o valor 1 (HIGH) para o pino
led.value(1)
```


Já no Exemplo 2.3 é utilizado a classe ADC do microptyhon que permite utilizarmos as portas do microcontrolador que possuem conversor digital-analógico, e a classe Pin. Além disso, também usamos uma outra biblioteca também muito utilizada em aplicações IoT chamada time, que permite controlar ações de tempo (**time**), como a função de sleep (dormir) da placa.

Exemplo 2.3 - Usando porta analógica (ADC) e Time (MicroPython)



```
#importando as classes ADC e Pin da biblioteca machine
from machine import ADC
from machine import Pin

#importando a biblioteca time
import time
```



```
#inicializando uma variável chamada adc0 que usa a classe  
ADC #e o id_da_porta que será usada.  
adc0=ADC(0)  
  
#inicia o laço de repetição infinito  
while True:  
    print(adc0.read_u16()) #exibe no terminal a leitura do  
    adc0  
    time.sleep_ms(500) #dorme por 500 ms
```

1.2 Wokwi

O Wokwi é um simulador gratuito de eletrônica direto no navegador. Nele é possível simular Arduino, ESP32, Raspberry Pi Pico e muitas outras placas, componentes e sensores mais utilizados em projetos de sistemas embarcados e IoT. Dado o caráter híbrido do nosso curso, usaremos o wokwi para realização de simulações das atividades práticas.

Essa plataforma tem algumas características bem interessantes que facilitam a programação e o teste de aplicações IoT. Por exemplo, ela simula a utilização de Wi-Fi usando a conexão de rede do próprio computador. Isso facilita muito o teste de aplicações que utilizam protocolos de rede, como MQTT e HTTP. Além disso também permite capturar sinais digitais (UART, I2C, SPI), simula cartão SD, permite a criação de chips personalizado e possui integração com o Visual Studio Code. Toda a documentação do Wokwi pode ser acessada por meio do link: <https://docs.wokwi.com/pt-BR/> .

A seguir, introduziremos a interface da plataforma e faremos a nossa primeira aplicação.

Conhecendo o simulador Wokwi

Para acessar o simulador, basta acessar o link <https://wokwi.com/> . Inicialmente, o wokwi já mostra 4 placas de desenvolvimento (ou micronconladores) que podem ser programados dentro da plataforma: Arduino (Uno, Mega, Nano), ESP32, STM32 e o Pi Pico. Além disso, a página inicial também mostra como iniciar o seu projeto usando diferentes linguagens de programação e os últimos projetos desenvolvidos.



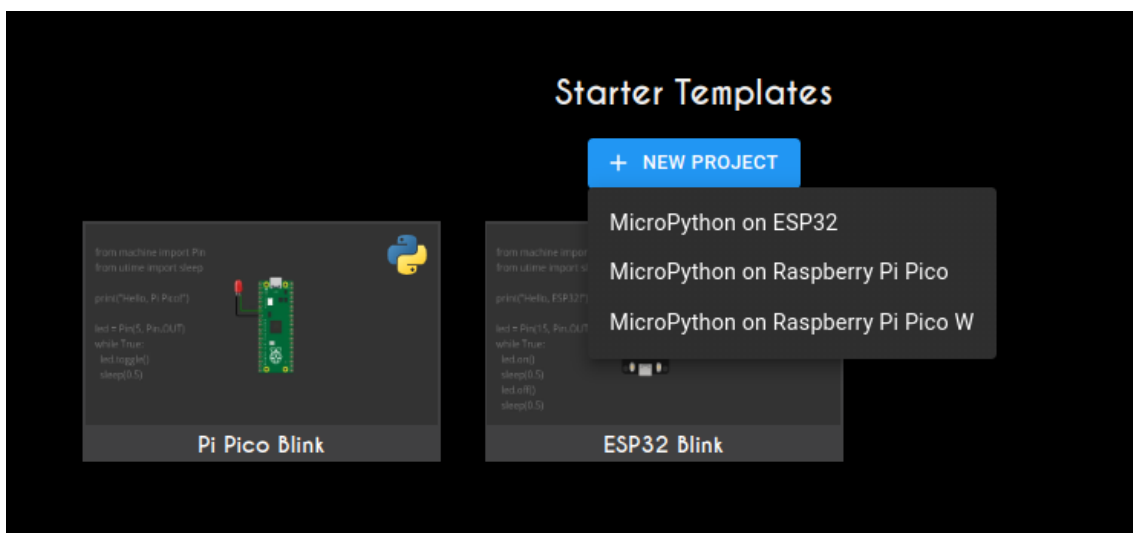
No canto superior direito da tela, é possível realizar o login. Recomendamos fazer o cadastro no sistema para ter acesso a funcionalidades como a de salvar e compartilhar projetos.

Vamos criar o nosso primeiro projeto em MicroPython.

1. Selecione a opção “**MicroPython**” na tela inicial do Wokwi.

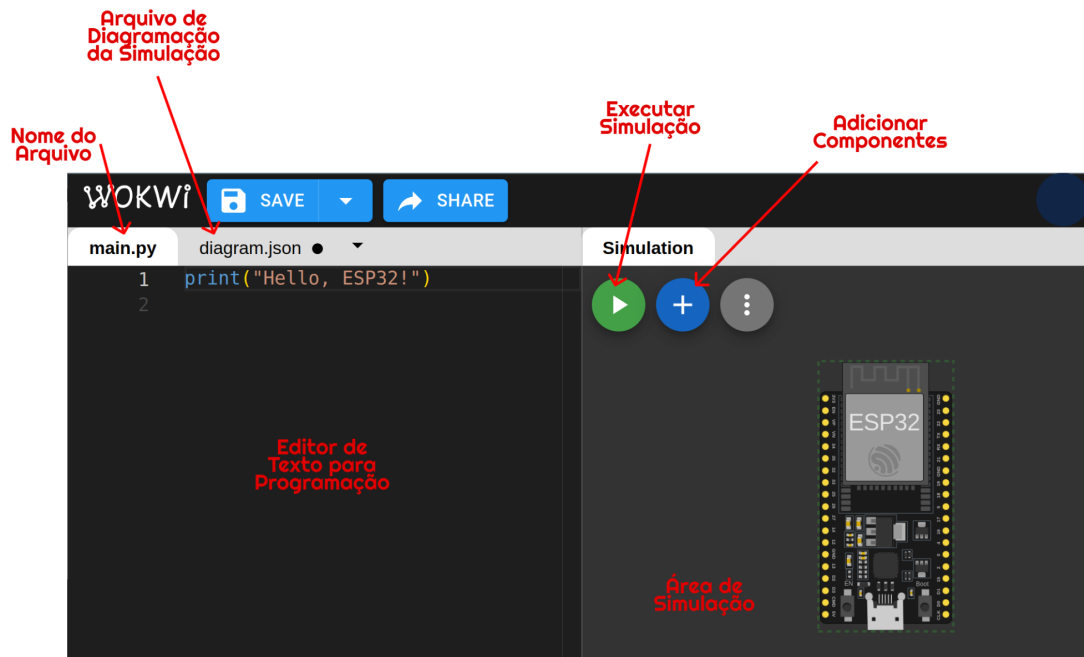


2. Na nova tela serão apresentados alguns exemplos básicos e de aplicações IoT. Você pode explorar esses projetos depois. Vamos escolher a opção **ESP32 Blink** em “**Starter Templates**” e escolher a opção “**Micropython on ESP32**”.

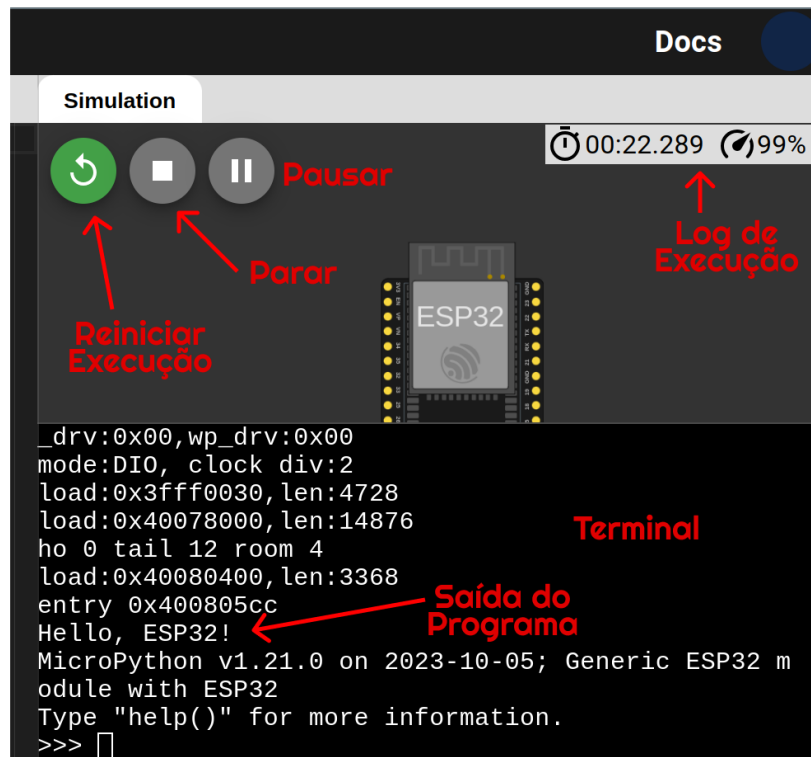


3. A próxima imagem mostra a tela do projeto. Na área de simulação, podemos interligar diferentes componentes a placa que iremos programar usando o botão de “+”. Toda essa diagramação é salva,

automaticamente, no arquivo “**diagram.json**” e é usado internamente pelo Wokwi no momento da simulação.

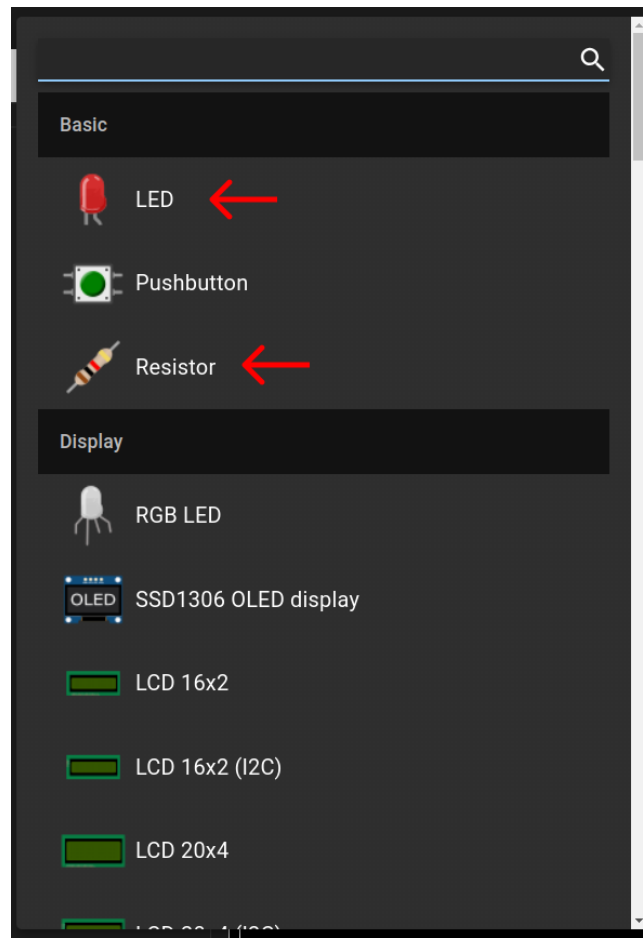


4. Quando clicamos no *play*, um terminal é apresentado na área de simulação. No exemplo, temos apenas um “print” que vai apresentar no terminal a mensagem: “Hello, ESP32!”.

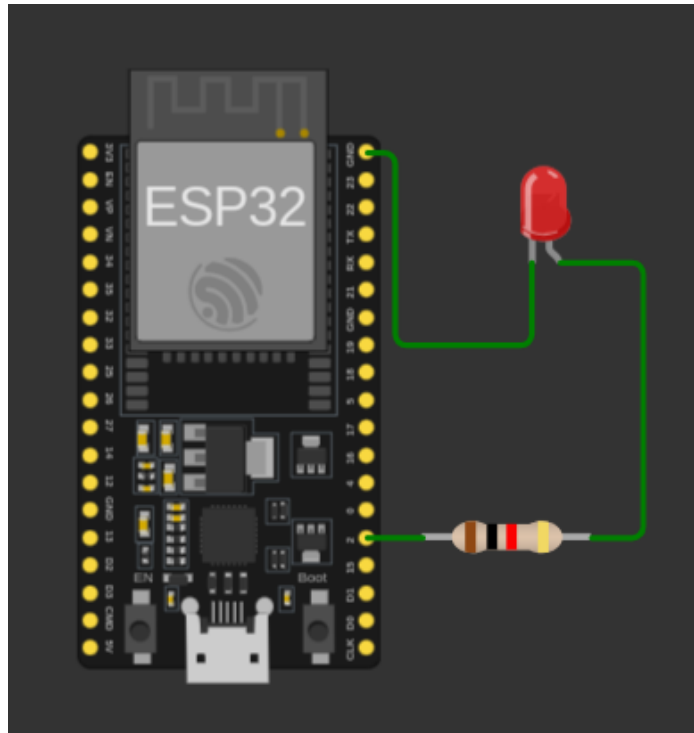


Agora que conhecemos um pouco do Wokwi, vamos criar uma aplicação básica. No Exemplo 2.1, apresentamos um código de um Blink que basicamente consistem em ativar e desativar uma porta do microncontrolador em intervalos de tempo definidos. Podemos ligar um LED a essa porta para vê-lo piscar. Para fazer isso, vamos realizar os seguintes passos:

Passo 1 - No mesmo projeto que já estamos, vamos inserir um LED e um resistor, ligado a pino 2. Para isso, basta clicar no botão “+” e procurar os componentes.



Passo 2 - Agora vamos fazer a ligação dos componentes com a placa. Usaremos a porta 2 do microcontrolador pra fazer o acionamento do LED passando pelo resistor de $1k\Omega$ (padrão). Para isso, basta clicar nos terminais ou portas que queremos fazer as ligações. OBS.: Lembre-se que o LED precisa estar polarizado diretamente para poder conduzir corrente e, portanto, acender. A figura abaixo mostra como ficou nossa montagem no simulador.



Perceba que um dos terminais do LED está ligado a primeira porta do lado direito do ESP que, nessa versão, corresponde ao GND, para fechar o circuito. Quando o pino 2 é colocado em nível alto, a corrente flui, passando pelo resistor e pelo LED, pela diferença de potencial entre as duas portas.

Agora, podemos partir para a programação! No Exemplo 2.1 é apresentado o código do Blink em C++ (Se você tiver usando esta linguagem, é só copiar e colar o código no editor do Wokwi, inserir a porta que estamos usando e rodar a simulação). Aqui, iremos fazer em MicroPython. A lógica é a mesma, a sintaxe é que vai ser alterada. O código comentado é apresentado a seguir.

Exemplo 2.4 - Blink (MicroPython)



```
#importamos as bibliotecas machine e time
import Pin from machine
import time

#definimos o nosso pino de saída (onde o LED está ligado)
led = Pin(2, Pin.OUT)
#definimos um laço de repetição infinito que funciona da
mesma forma da função loop no c++
while True:
    #mudamos a porta para nível alto.
    led.on()
    #esperamos 1 segundo
    time.sleep(1)
    #mudamos a porta para nível baixo
    led.off()
    #esperamos 1 segundo
    time.sleep(1)
```

Agora é só rodar a simulação e ver o led piscar com intervalos de 1 segundo!



Atenção Caso o LED não pisque, você deve verificar se fez as ligações de forma correta, se acionou a porta correta no código ou se houve alguma resposta de erro no terminal da simulação.

Assim finalizamos o nosso primeiro projeto simulado no Wokwi. Usaremos muito essa ferramenta no decorrer do nosso curso para testarmos nossas aplicações antes de introduzirmos o programa na placa.

1.3 Bipes

O BIPES (citar) significa Plataforma Integrada baseada em blocos para sistemas embarcados (Block based Integrated Platform for Embedded Systems), foi criado por Rafael Vidal Aroca juntamente com outros pesquisadores e pode ser acessado por meio do link: <http://bipes.net.br/ide/?lang=pt-br> (versão em português). Inicialmente pensado para programação em blocos, o BIPES permite a programação de uma variedade de placas usadas em sistemas embarcados, diretamente do



navegador, sem a necessidade de instalar qualquer programa na máquina local do desenvolvedor.

Apesar de recente, o BIPES vem se desenvolvendo de forma rápida e consistente e hoje já conta com diversas funcionalidades, como o upload de arquivos para a placa de desenvolvimento ou instalação de firmware direto do navegador, programação usando editor de texto, terminal de interação com a placa, ambiente de apresentação de dados para IoT, MQTT Client, entre outras.

A interface do Bipes é simples e intuitiva. Cada aba corresponde a uma funcionalidade específica da aplicação como explicado a seguir.

1. **Blocos** - Como comentamos anteriormente, o Bipes foi projetado para funcionar dentro do conceito de *no-code*, a partir da programação em blocos. São vários os blocos que podem ser usados para programar os microcontroladores de forma simples e intuitiva. É uma ótima opção para quem está iniciando a programação de sistemas embarcados e aplicações IoT.
2. **Console** - Uma vez que a placa está conectada, podemos interagir com ela por meio de comandos usando o REPL que é iniciado quando instalamos o MicroPython nela (abordaremos como fazer instalação na próxima seção). Basicamente, é um terminal interativo que nos possibilita enviar comandos para a placa por meio do Python.
3. **Arquivos** - Nesta aba podemos criar ou editar arquivos que serão enviados ou que já estão na placa. Usaremos muito essa funcionalidade aqui no nosso curso.
4. **Compartilhado** - Nesta aba é possível ter acesso a diversos projetos disponibilizados pela comunidade, para diferentes aplicações.
5. **Dispositivo** - Existe uma variedade de placas que podem ser utilizadas em aplicações IoT, inclusive placas que utilizam a mesma família de microcontroladores, mas de diferentes modelos. Nesta aba, temos acesso a diagramas de pinagem de inúmeras placas que vão nos auxiliar na montagem correta do nosso circuito.
6. **IoT, EasyMQTT e Databoard** - As três últimas abas nos permitem, basicamente, enviar e receber dados, de diferentes formas, em



diferentes contextos. Por exemplo, podemos receber os dados diretamente da serial da placa e apresentá-los em um gráfico, ou participar de uma comunicação usando o protocolo MQTT. Usaremos essas abas durante o nosso curso.

Além das abas, a interface principal do Bipes também traz ferramentas que auxiliam na escolha e comunicação com a placa que será programada, além de facilitar o compartilhamento de arquivos, organização de projeto, entre outras funcionalidades. A Figura 2.2 mostra a interface inicial da IDE do Bipes.

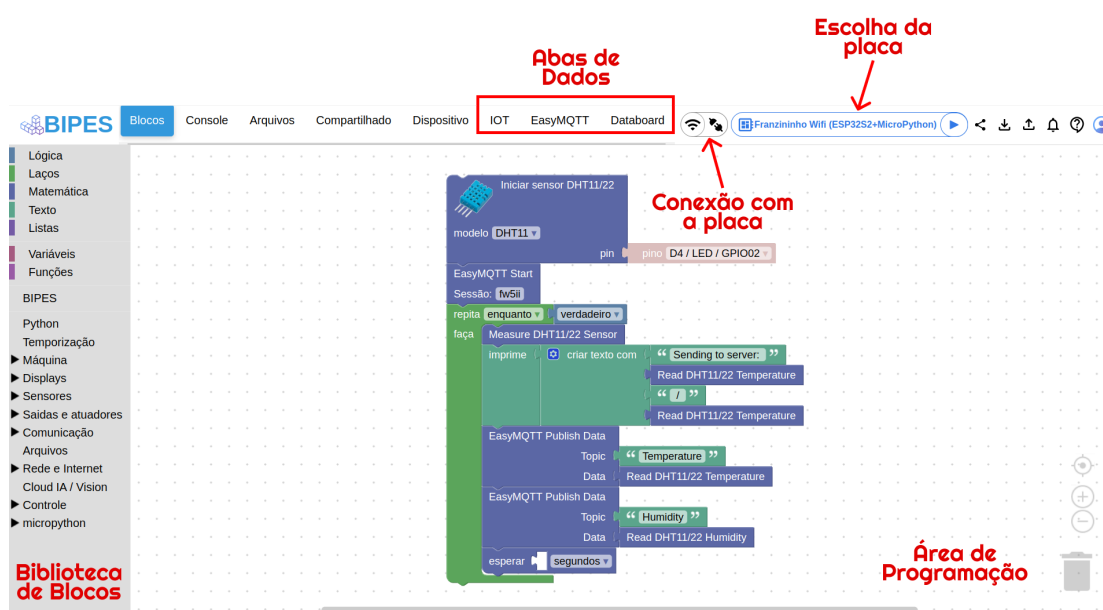


Fig. 2.2 - Tela inicial da IDE do Bipes.

Basicamente, o Bipes converte os blocos em código micropython e envia para a placa de desenvolvimento. Desta forma, para que possamos usar o MicroPython de forma física precisamos fazer a instalação dele na placa.

A seguir, mostraremos o passo-a-passo de como fazer a instalação do MicroPython nas placas de desenvolvimento ESP e Pico.

1.3.1 Instalação do Micropython no ESP

O Bipes facilita muito a instalação do MicroPython nas placas de desenvolvimento do ESP32 e ESP8266. Para isso, siga os passos abaixo.



Atenção O passo a passo descrito nessa seção foi testado usando navegadores Google Chrome e Microsoft Edge em ambientes MAC, Linux e Windows para as placas ESP32 e ESP8266.

Passos para instalação do MicroPython usando o Bipes

1. Acesse: <https://bipes.net.br/flash/esp-web-tools/>
 2. Conecte a placa ESP32 ou ESP8266 à porta USB do seu computador.
 3. Clique no botão “**Connect**” e escolha a porta serial (COM) que sua placa está conectada.
 4. Uma caixa de diálogo vai aparecer para conduzir a instalação. Clique em “**INSTALL MICROPYTHON**” e depois em “**INSTALL**”. No término da instalação você verá uma mensagem de “Installation Complete” na sua tela.
 5. Para verificar que realmente deu tudo certo, clique em “**LOGS & CONSOLE**” e observe o comportamento da placa no console.
-

Alguns problemas que podem acontecer nesse processo de instalação são:

- 1) Caso o botão “Connect” não apareça ao acessar a página de instalação, certifique-se que você está usando “https” e não “http”. Isso também é válido para fazer as conexões do navegador direto com a placa de desenvolvimento.
- 2) O MicroPython cria uma rede “MicroPython”, para conexão remota via Wi-Fi. Porém, algumas placas ESP8266 tem uma limitação de corrente para suprir a demanda de energia quando inicia o serviço de *Access Point*. Se isso ocorrer, a placa entrará em um loop infinito de *reset*.

1.3.2 Instalação no Raspberry Pi Pico

A instalação no Pico é mais simples do que no ESP. Siga o passo a passo no quadro a seguir.



Passos para instalação do MicroPython no Raspberry Pi Pico ou Pico W

1. Pressione e segure o botão BOOTSEL e conecte sua placa ao computador usando um cabo USB. Solte o botão depois da placa ter sido conectada.
2. Quando conectado, o computador identifica a placa como se fosse um dispositivo de armazenamento, chamado **RPI-RP2**. Desta forma, você pode acessá-la diretamente como uma pasta em seu computador.
3. Faça o download do arquivo MicroPython UF2 de acordo com a placa que você está usando:
 - a. Pi Pico:
<https://micropython.org/download/rp2-pico/rp2-pico-latest.uf2>
 - b. Pi Pico W (Versão com WiFi e Bluetooth LE)
<https://micropython.org/download/rp2-pico-w/rp2-pico-w-latest.uf2>
4. Agora, basta copiar o arquivo para a placa. Sua placa será reiniciada e você já terá acesso ao REPL via serial.

Todo esse processo é descrito na página oficial da Raspberry disponível em:
<https://www.raspberrypi.com/documentation/microcontrollers/micropython.html>.

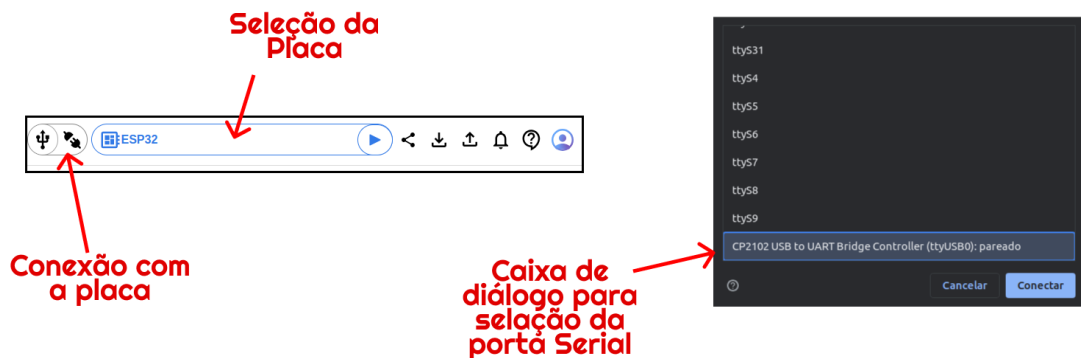
1.4 Aplicação Prática

Agora que o MicroPython já está instalado na placa, podemos testar a nossa aplicação Blink diretamente nela, fazendo o LED interno piscar. Vamos começar com o ESP! Com a placa conectada ao computador, abra o Bipes e selecione a versão da placa que você vai usar.

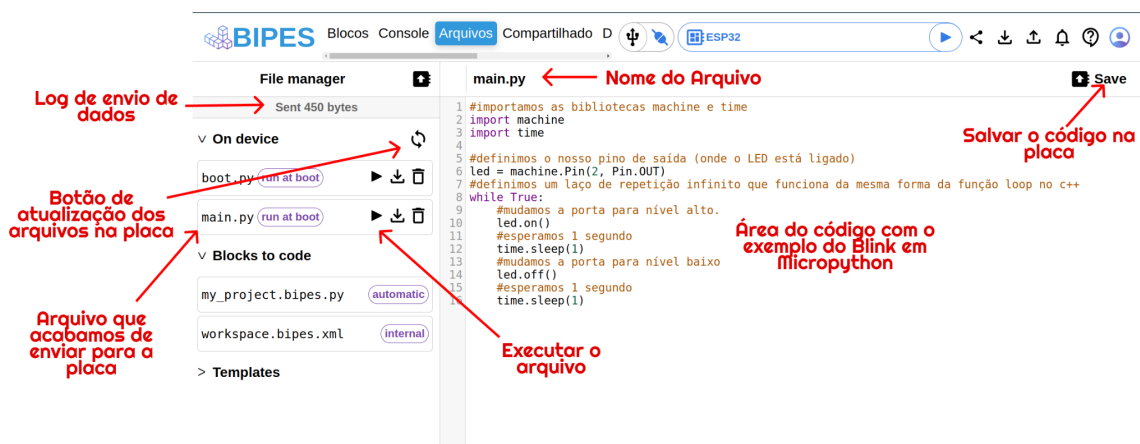


Lembre-se! Use o protocolo HTTPS para rodar o BIPES. A conexão segura faz com que o navegador possa acessar a porta serial do seu computador.

No caso do nosso exemplo, usaremos a placa ESP32. Depois de selecionada, clique no botão de conexão e escolha a porta que sua placa está conectada.



Depois que o Bipes estiver conectado a placa (o ícone de conexão mudou), selecione a aba “**Arquivos**”. No campo *filename* insira **main.py** que será o nome do arquivo que estamos editando. Na área do código, copie e cole o código do Exemplo 2.4. Por fim, clique no botão “**Save**” no canto direito superior.



Depois de pressionado o botão de salvar, o Bipes envia o arquivo para a placa (podemos verificar esse envio no log de envio de dados). Se o arquivo não aparecer automaticamente no painel, podemos clicar no botão de atualização, e veremos o nosso arquivo “**main.py**” já na placa. Agora, é só clicar no botão de **executar** e ver o led interno da placa piscar (Figura 2.3).



Fig. 2.3 - ESP32 executando o Blink (Autoria Própria)

Lembre-se que no código colocamos um loop infinito de forma que o programa ficará rodando até que ele seja parado. Caso necessite parar, basta ir na aba **Console**, selecionar o Terminal e usar as teclas “Ctrl+C” para “matar” o processo.

Agora, vamos inserir o mesmo programa em um Raspberry Pi Pico. A única diferença do código que usamos no ESP é na indicação do pino que queremos controlar. No caso do Pi Pico, usaremos o nome “LED” no lugar do número da porta. Ou seja, a linha de definição da porta será

```
led = Pin("LED", Pin.OUT)
```

Altere a placa para a versão do Raspberry Pi Pico que vocês está usando. No caso do nosso exemplo, usaremos a versão básica (sem acesso a internet e Bluetooth). Seguimos o mesmo procedimento que usamos anteriormente: conectamos a placa, criamos um arquivo “**main.py**”, inserimos o código com a alteração anterior, salvamos e executamos.



Pronto! Agora o Blink também está rodando no Raspberry Pi Pico como podemos ver na Figura 2.4.

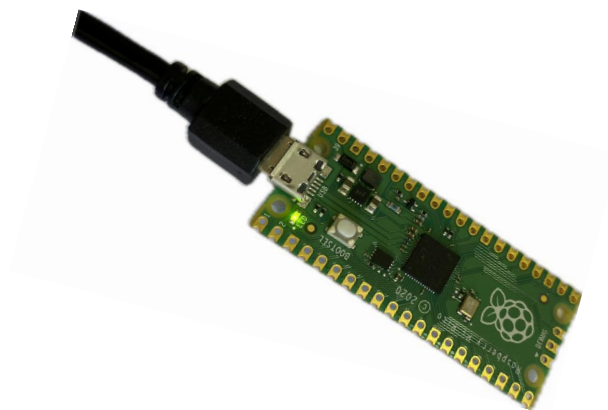


Fig. 2.4 - Raspberry Pi Pico executando o Blink. (Autoria Própria)





BOM CURSO!



UNIVERSIDADE DE SÃO PAULO