UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

VINICIUS DE OLIVEIRA PEIXOTO

# IMPLEMENTAÇÃO DE MELHORIAS NA PLATAFORMA BIPES E NO SNEK PARA MICROCONTROLADORES DE PEQUENO PORTE

SÃO CARLOS - SP
2022

VINICIUS DE OLIVEIRA PEIXOTO

# IMPLEMENTAÇÃO DE MELHORIAS NA PLATAFORMA BIPES E NO SNEK PARA MICROCONTROLADORES DE PEQUENO PORTE

Trabalho de conclusão de curso apresentado ao Departamento de Computação da Universidade Federal de São Carlos, para obtenção do título de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Rafael Vidal Aroca

SÃO CARLOS - SP
2022

**FUNDAÇÃO UNIVERSIDADE FEDERAL DE SÃO CARLOS**

**COORDENAÇÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO - CCEC/CCET**

Rod. Washington Luís km 235 - SP-310, s/n - Bairro Monjolinho, São Carlos/SP, CEP 13565-905

Telefone: (16) 33518234 - http://www.ufscar.br

DP-TCC-FA nº 1/2022/CCEC/CCET

**Graduação: Defesa Pública de Trabalho de Conclusão de Curso**

**Folha Aprovação (GDP-TCC-FA)**

## FOLHA DE APROVAÇÃO

**Vinicius de Oliveira Peixoto**

# Implementação de melhorias na plataforma bipes e no snek para microcontroladores de pequeno porte

**Trabalho de Conclusão de Curso**

**Universidade Federal de São Carlos – Campus São Carlos**

São Carlos, 25 de março de 2022

**ASSINATURAS E CIÊNCIAS**

| Cargo/Função | Nome Completo |
|---|---|
| Orientador | Rafael Vidal Aroca |
| Membro da Banca 1 | Tatiana Pazelli |
| Membro da Banca 2 | Luciano de Oliveira Neris |

**RESUMO**

A forma com que se aprende um assunto é tão importante quanto o próprio assunto em si. É necessário que o novo conteúdo seja apresentado de um jeito que a pessoa não encontre resistência para fazer as conexões e julgamentos necessários para desenvolver conhecimento. Por conta disso, o ensino de Computação, um tema extremamente importante nos dias atuais, tem sofrido alguns entraves, pois há algumas barreiras iniciais quando se começa a aprender a programar, como lidar com linguagens de programação complexas, por exemplo. A programação visual oferece uma alternativa interessante para diminuir essas dificuldades, e muitas plataformas aproveitam desse recurso para integrar esse facilitador na sua área. O BIPES (*Block-based Integrated Platform for Embedded Systems*) é uma plataforma *open source* que usa essa forma de programação por blocos para facilitar o aprendizado e desenvolvimento de soluções em sistemas embarcados. Seus blocos são de fácil manuseio e possuem uma boa variedade de funções para uso em diversos tipos de placas, sensores e atuadores diferentes. Há um interesse em expandir a capacidade dessa plataforma, adicionando ainda mais possibilidades para novatos conseguirem aprender a desenvolver aplicações com sistemas embarcados. Essa expansão envolve o suporte de novos tipos de placas, como o Arduino, que não era suportado até recentemente, mas que foi integrado ao projeto após a adição da capacidade de programar usando uma nova linguagem de programação, baseada em Python, extremamente leve, chamada Snek. Esse projeto buscou expandir a capacidade do BIPES adicionando um novo bloco para controle de um servo motor usando Arduino através da linguagem Snek. O bloco possui interação com as funcionalidades IoT (*Internet of Things*) da plataforma e foi validado por usuários com e sem experiência prévia em programação como sendo um recurso muito fácil de se usar.

**Palavras-chave**: BIPES. Internet das coisas. Programação em blocos. Servo motor. Sistemas embarcados. Snek.

**ABSTRACT**

The way you learn a subject is as important as the subject itself. It is necessary that the new content is presented in such a way that the person won't find resistance to make the connections and judgments needed to develop knowledge. That's why teaching Computer Science, an area of great importance nowadays, has met some obstacles. There are barriers that one faces when they start learning programming, like dealing with hard-to-learn programming languages. Visual programming is seen as an interesting alternative to help lower these difficulties, and for that a lot of platforms have been using this resource to ease the learning problem in their field. BIPES (Block-based Integrated Platform for Embedded Systems) is an open source platform that uses block programming to make learning and developing solutions with embedded devices easy. The platform has a lot of easy to handle blocks that span through many different topics, for many different types of boards, sensors and actuators. Expanding the capacity of this platform is desired, giving even more possibilities for computer newbies to be able to build applications for embedded systems. This expansion can happen by adding support to new types of boards, like Arduino, a board that was not supported up until recently, but was added to BIPES due to the introduction of a new extremely lightweight Python-inspired programming language called Snek. This project aims to widen BIPES capabilities by making a new block design to control a servo motor using an Arduino board programmed with Snek code. The block can be used to incorporate monitoring and controlling the servo motor through IoT (Internet of Things) and was validated by random users with and without prior programming experience as being really easy to use.

**Keywords**: BIPES. Block-based programming. Embedded systems. Internet of Things. Servo motor. Snek.

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS

Hex – Hexadecimal

ID – Identificador

## LISTA DE SIGLAS

API – *Application Programming Interface*

BIPES – *Block-based Integrated Platform for Embedded Systems*

EEPROM – *Electrically Erasable Programmable Read-Only Memory*

HTML – *HyperText Markup Language*

HTTP – *HyperText Transfer Protocol*

IDE – *Integrated Development Environment*

IoT – *Internet of Things*

ISP – *In-circuit Serial Programmer*

MIT – *Massachusetts Institute of Technology*

PC – *Personal Computer*

PWM – *Pulse-Width Modulation*

RAM – *Random Access Memory*

RC – *Radio Control*

REPL – *Read–Eval–Print Loop*

ROM – *Read-Only Memory*

SBC – Sociedade Brasileira de Computação

URL – *Uniform Resource Locator*

USB – *Universal Serial Bus*

## LISTA DE SÍMBOLOS

kB – *Kilobyte*

Kgf.cm – Kilograma-força centímetro

kΩ – *Kiloohms*

μF – *Microfarad*

ms – Milissegundos

V – Volts

**SUMÁRIO**

# 1 INTRODUÇÃO

Quando pensamos em Computação, logo vem à nossa mente a imagem de computadores modernos, várias linhas de códigos, robôs e internet. Embora todos esses elementos façam parte da Computação, ela não se restringe a unicamente trabalhar com computadores. Computação é uma área que usa métodos e técnicas para resolver problemas, identificar padrões e criar formas de automatizar soluções, seja usando componentes físicos ou virtuais (RAABE et al., 2017). Podemos ver que, partindo dessa definição, uma tarefa comum como “quero comer uma sobremesa” pode se tornar uma tarefa de Computação, levando em conta dados sobre que tipo de sobremesa a pessoa gosta de comer para uma tomada de decisão, e seguindo um algoritmo (a chamada receita) para fazer a sobremesa. Esse tipo de ação está tão presente no nosso dia-a-dia que nem nos damos conta de como a Computação tem forte envolvimento com todas as demais áreas do conhecimento. Por isso, entender conceitos básicos de Computação pode ser fundamental para o desenvolvimento de qualquer pessoa, tanto quanto Matemática, Filosofia, Física e outras ciências (Raabe et al., 2017).

A forma com que uma pessoa enfrenta um problema impacta em que tipo de solução será obtida. Um problema que foi analisado e modelado, com certeza irá receber uma solução muito bem ajustada à ele. Sendo esse problema recorrente, há a possibilidade de reduzir seu impacto devido a presença de um método bem estudado sobre como lidar com ele. Toda essa forma de pensar sobre como resolver problemas vem de um dos pilares da Computação que a SBC (Sociedade Brasileira de Computação) chama de “Pensamento Computacional”. É através do Pensamento Computacional que uma pessoa consegue criar boas soluções para problemas, fazendo bom uso das três competências que o regem: Abstração, Análise e Automação (Raabe et al., 2017).

Ensinar Computação ainda é algo que demanda investimento. Escolas e universidades precisam de ferramental para que os alunos tenham contato com as tecnologias mais atuais, e assim o aprendizado não ficar defasado em relação à atualidade. Entretanto, alguns recursos não estão sempre disponíveis, o que faz com que os cursos busquem alternativas de qualidade com custo baixo. Dentre as alternativas, uma delas vem se destacando nos últimos anos: o Arduino.

Arduino é um projeto que disponibiliza dispositivos e programas de alta qualidade por um preço bem competitivo. Suas placas são capazes de executar tarefas dos mais variados portes, sejam puramente para *hobby*, educacionais e até industriais. O uso de Arduinos não é necessariamente restrito à educação especializada, como cursos superiores da área de Computação. O aumento da abrangência do contato com a tecnologia expande a possibilidade de desenvolvimento do Pensamento Computacional. Sohn (2014) desenvolveu um sistema de educação de programação baseado em Arduino, e aplicou em alunos do ensino primário, obtendo sucesso no aumento da capacidade de resolução de problemas pelos alunos.

Embora tenha todos esses benefícios, o Arduino ainda pode encontrar problemas para se difundir nos centros de aprendizagem. Por exemplo, há o pré-requisito de conhecimento de linguagem de programação para conseguir usar a ferramenta de forma satisfatória. Assim, não é tão simples ensinar Computação usando algo que demanda conhecimento em Computação. Com isso, há uma abertura interessante para tecnologias que simplifiquem os primeiros passos na Computação a ponto de não ser impeditivo para o aluno não saber linguagem de programação. A Programação Visual se encaixa perfeitamente nessa situação.

Blockly (BLOCKLY, 2022) é uma linguagem de programação visual. Permite que o usuário crie programas a partir de manipulação de blocos que representam linhas de códigos pré-construídas em linguagens mais populares, como Python e JavaScript. Com isso, o usuário não precisa se preocupar se está escrevendo a linguagem de programação da forma correta, retirando assim uma das barreiras de entrada para a programação. O Blockly também é capaz de retirar os erros de tipagem de valores, restando ao usuário apenas a tarefa de conectar os blocos de acordo com o fluxo desejado. Por exemplo, O MIT App Inventor do *Massachusetts Institute of Technology* é uma ferramenta que foi implementada usando o Blockly, e tem o propósito de facilitar a criação de aplicações Android e iOS totalmente funcionais (MIT APP INVENTOR, 2022). O BIPES (*Block-based Integrated Platform for Embedded Systems*) é outra plataforma que utiliza o Blockly com o intuito de gerar código para sistemas embarcados a partir de blocos.

BIPES é uma plataforma de código aberto (*open source*) que tem por objetivo facilitar o desenvolvimento de aplicações para sistemas embarcados através de programação visual. O projeto da plataforma é totalmente *web*, funcionando

diretamente no navegador, sem a necessidade de downloads adicionais. O usuário também tem a opção de baixar a página do BIPES, e a aplicação continua funcionando mesmo sem conexão com a Internet, o que elimina problemas causados por conexão de Internet instável. O BIPES oferece suporte para Internet das Coisas (IoT - *Internet of Things*) (AROCA et al., 2021), e tem a capacidade de trabalhar com uma grande variedade de tipos de placas diferentes através do uso de linguagens de programação como Python, CircuitPython ou MicroPython, basta conectá-las ao computador através de um cabo USB (*Universal Serial Bus*), ou via *Bluetooth* ou Wi-Fi (JUNIOR et al., 2020). Um esquemático da estrutura da plataforma é apresentado na Figura 1.

Embora o desenvolvimento em uma diversa gama de placas seja suportada no BIPES, havia um grupo muito famoso que inicialmente não fazia parte da coletânea. As placas Arduino não rodavam no BIPES devido às suas limitações de *hardware*. Como os programas do BIPES são convertidos de blocos para código, é necessário que as placas possuam uma forma de executar esse código, ou seja, elas precisam de um interpretador. Como as placas Arduino têm poucos recursos de *hardware*, não era possível encontrar um interpretador que conseguisse executar os códigos, nem mesmo MicroPython cujos requisitos já são baixos. Mas isso mudou com a integração da linguagem Snek na plataforma.

Snek é uma linguagem de programação baseada em Python criada pelo desenvolvedor de *software* Keith Packard (SNEKLANG, 2022). Seu propósito era ser tão pequena a ponto de conseguir rodar em placas com pouquíssimos recursos, o que é perfeito para os Arduinos. Com o Snek integrado ao BIPES, a plataforma agora pode ser usada em Arduinos, um grupo de placas muito famoso com milhões de dispositivos espalhados pelo mundo, que são utilizados das mais diversas formas, desde projetos educacionais até industriais.

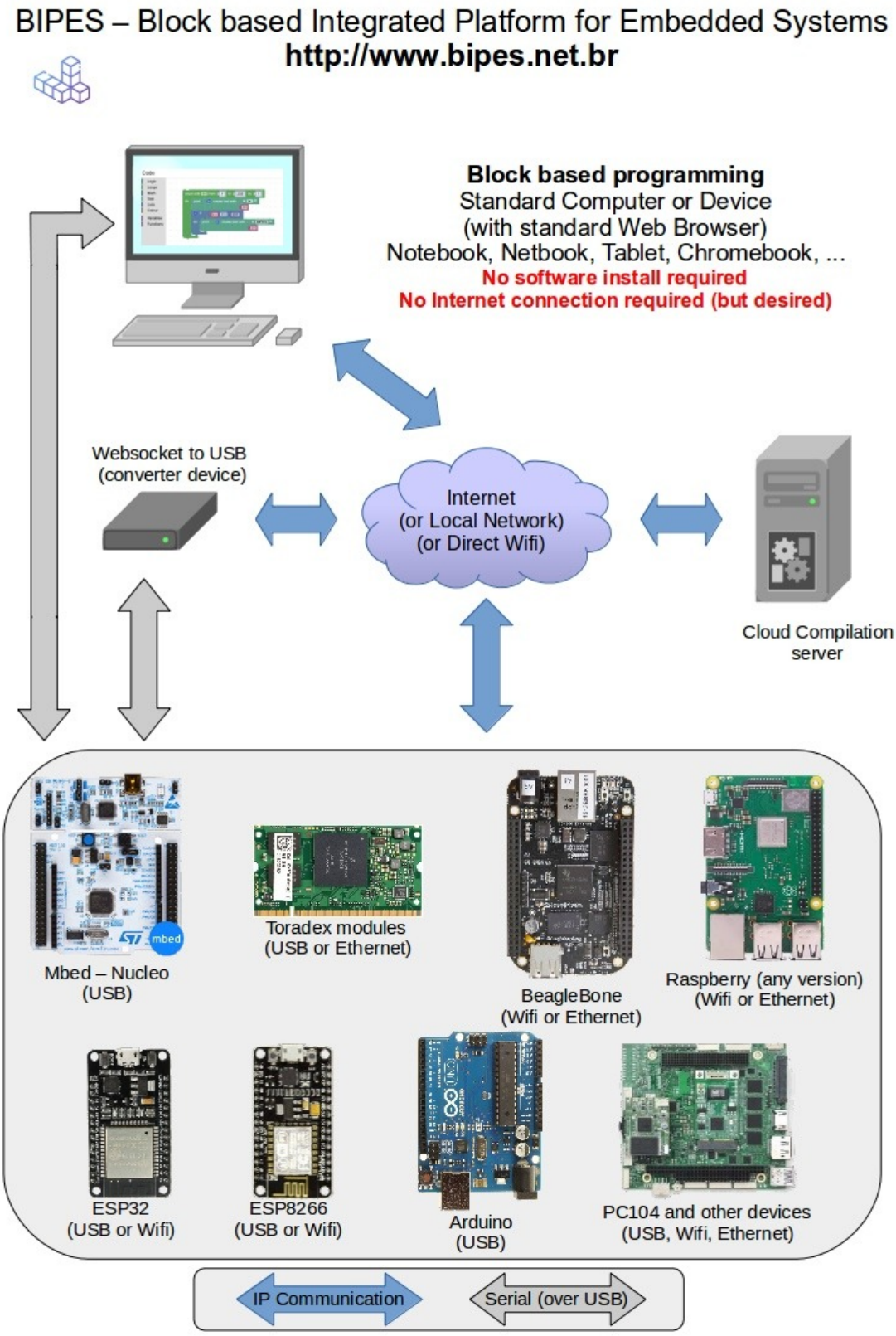
Embora a adição do Snek seja um grande passo para a expansão do uso do BIPES, ainda é cedo demais para que todo seu potencial seja explorado. O BIPES não possui os blocos referentes à todas as possibilidades que o Snek possui, portanto há uma limitação em seu uso. Este trabalho tem como objetivo expandir a capacidade do BIPES em usar o Snek para controlar Arduinos através do desenvolvimento de novos conjuntos de blocos capazes de acionar um servo motor, bem como abrir a possibilidade de configurar as placas para executar programas Snek gerados pelo BIPES direto na inicialização, sem a necessidade de comandos

enviados pelo usuário. Também faz parte do escopo desse projeto a integração da linguagem Snek com IoT através de painéis de controle e monitoramento através de protocolo MQTT[1].

No capítulo 2 é apresentada uma análise da teoria que envolve os temas relacionados ao projeto. Essa teoria baseia-se em documentação da literatura científica para contextualizar a utilidade da plataforma BIPES, a importância de formas alternativas para se ensinar Computação e as tecnologias utilizadas. O capítulo 3 apresenta a forma com que o presente projeto foi desenvolvido, incluindo detalhes dos materiais utilizados e o passo-a-passo para a criação das melhorias que foram adicionadas à plataforma. No capítulo 4 são apresentados os resultados práticos obtidos com demonstração de utilização. Também faz parte desse capítulo a apresentação da coleta de informação referente ao uso dessas novas funcionalidades por usuários reais dos mais variados níveis de expertise. As avaliações sobre os resultados obtidos, as críticas e as sugestões para melhorias futuras estão reportadas no capítulo 5. O capítulo 6 resume todo o conhecimento adquirido durante o período de desenvolvimento do projeto, e os apêndices presentes no final contêm informações extra relevantes ao projeto.

[1] Embora pareça, MQTT não é uma sigla, e sim o nome do protocolo.

**Figura 1 –** Diagrama de funcionamento do BIPES




**Fonte:** BIPES – Tech Details[2]

[2] Disponível em: <http://bipes.net.br/docs/get-started/tech-details.html>. Acesso em: 04 abr 2022

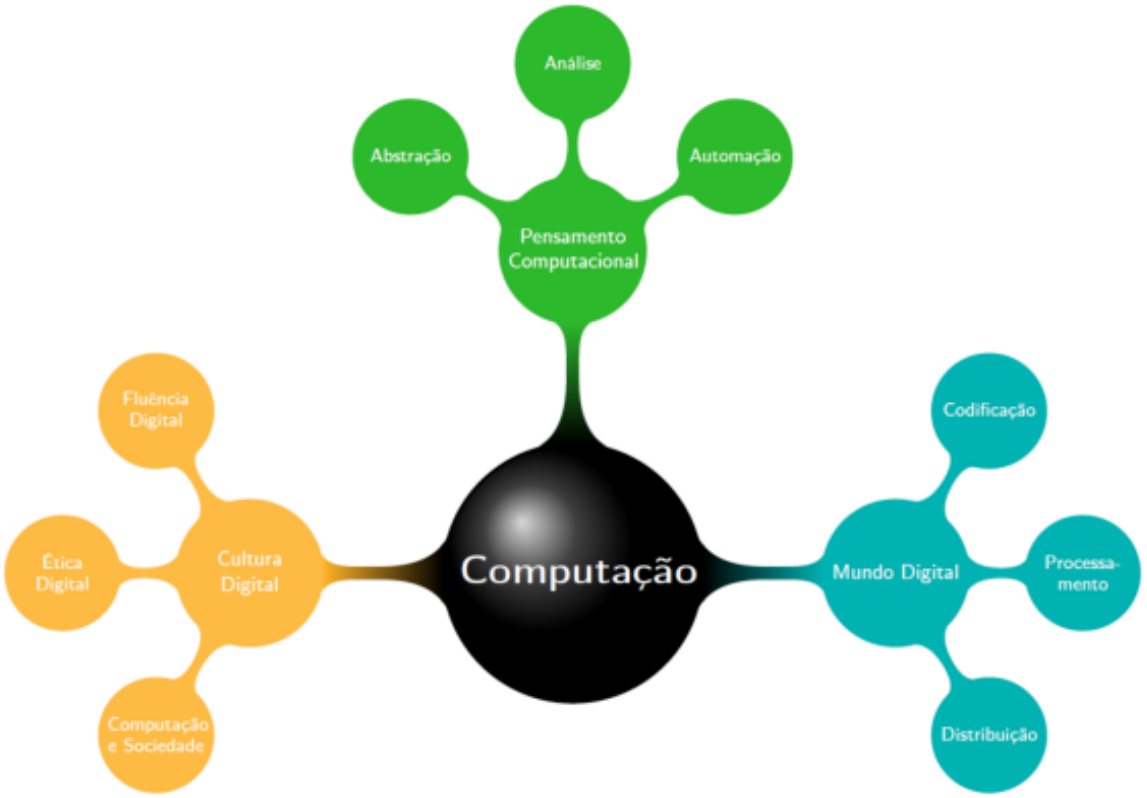
## 2 FUNDAMENTAÇÃO TEÓRICA

Aprender programação é uma tarefa que pode possuir algumas dificuldades, tais como desenvolver a habilidade de “pensamento computacional”, compreender estruturas de controle, até o próprio uso das linguagens de computação. Essas dificuldades podem trazer baixo desempenho, desmotivação e até desistência em alunos de computação (DA SILVA, 2018). O desenvolvimento de um pensamento computacional auxilia o aluno a potencializar sua capacidade de solução de problemas através das suas três competências, conforme apresentadas na Figura 2: abstração, automação e análise. Abstração ajuda o aluno a compreender e utilizar representações adequadas para descrever informações e processos. Automação faz com que ele seja capaz de descrever soluções por meios de algoritmos e assim poder construir modelos computacionais de sistemas complexos. E a análise aguça seu julgamento para verificar se existe uma forma de resolver um problema com automação, e se uma solução já existente pode ser melhorada (Raabe et al., 2017).

Na busca de meios de ensinar programação, professores recorrem a diversos métodos. Embora as formas tradicionais sejam inevitáveis na carreira dos alunos, acabam sendo muito complexas para alguns deles (COSTA e PIEDADE, 2021), causando desmotivação (EL-ABD, 2017). Isso cria um desafio para quem ensina programação: aumentar o aprendizado e diminuir a complexidade dos métodos empregados. Uma das formas de simplificar a programação para os alunos é usando ferramentas que reduzem a carga sintática e semântica das linguagens. Um exemplo disso é a programação em blocos, que permite aos alunos criar programas como se estivessem montando um castelo com peças de Lego (COSTA e PIEDADE, 2021).

Programação em blocos já se mostrou bem efetiva em diminuir a dificuldade em que se aprende a trabalhar com programação de sistemas embarcados. Resultados obtidos por Torres, Aroca e Burlamaqui (2014) mostraram que a percepção dos alunos quanto a dificuldade de programação cai em média 57% quando se usa programação em blocos ao invés do método tradicional. Em outros estudos, a programação em blocos estimulou entusiasmo, comprometimento e diversão nos alunos, segundo avaliação feita usando perguntas adaptadas da escala de Laros e Steenkamp (LÓPEZ et al., 2021).

**Figura 2** – Os eixos da Computação



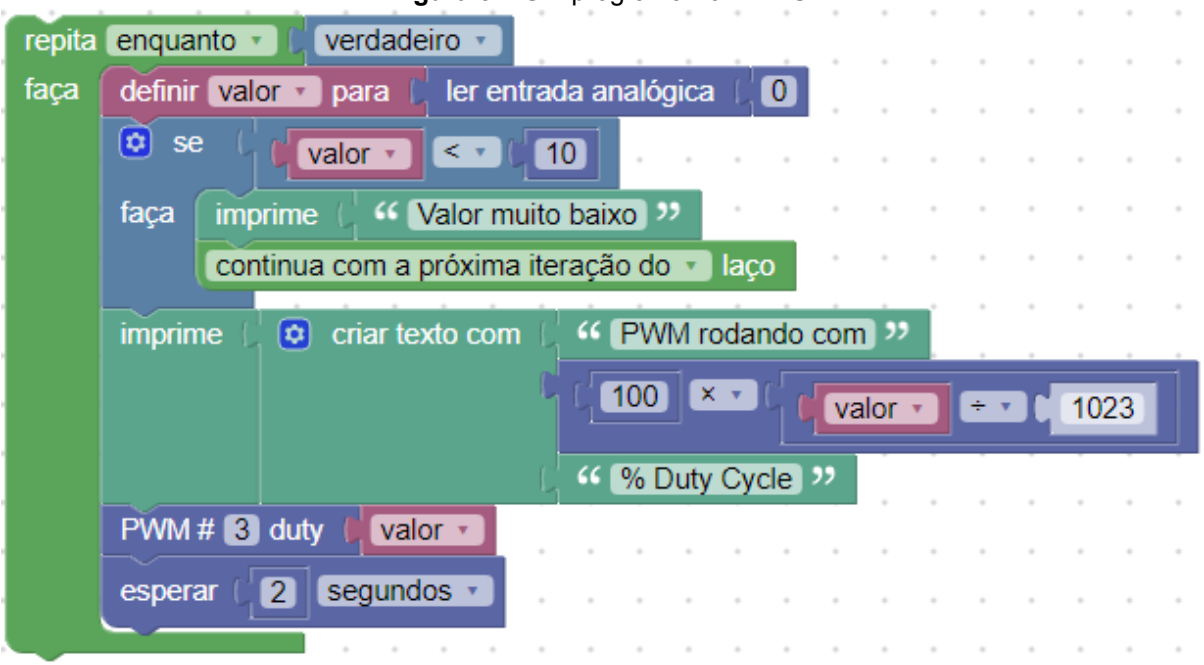

Fonte: Raabe et al., 2017

Atualmente, um forte uso de programação está em sistemas embarcados. El-Abd (2017) os descreve como “sistemas que não necessariamente executam tarefas computacionais, mas ainda sim são controlados por uma unidade computacional”. Tais sistemas estão presentes em diversos lugares, desde aplicações domésticas, em escritórios, eletroeletrônicos, até na indústria automotiva, e usam microcontroladores com poucos recursos de hardware quando comparados a um computador comum. Embora muito presentes, ainda há várias barreiras para novos e inexperientes programadores que desejam explorar as possibilidades que os sistemas embarcados oferecem, como uso de linguagem de programação complexa e plataformas de desenvolvimento que precisam ser instaladas em PCs (*Personal Computer*) (DEVINE et al., 2019).

O BIPES faz o uso da programação em blocos para programar sistemas embarcados, funciona pelo navegador sem necessidade de recursos adicionais, e ainda pode ser utilizado sem conexão com a Internet. Um exemplo de código em blocos do BIPES pode ser visto na Figura 3. A plataforma gera um código Python que pode ser interpretado por diversos tipos de placas diferentes com interpretadores Python, MicroPython, CircuitPython ou Snek (AROCA, 2021). Todos

os recursos apresentados levam a facilitação do aprendizado de programação de sistemas embarcados. Além disso, o BIPES conta com recursos de comunicação úteis para o uso em IoT, como protocolo HTTP (*HyperText Transfer Protocol*) e EasyMQTT (SILVA, 2020).

**Figura 3** – Um programa no BIPES




Fonte: Elaborado pelo autor

Como as placas precisam rodar os interpretadores, há uma demanda mínima de requisitos de *hardware* para que tudo funcione. Por exemplo, MicroPython tem um requisito mínimo de 128kB de ROM (*Read-Only Memory*) e 8kB de RAM (*Random Access Memory*) (MICROPYTHON, 2022). Com isso em mente, o BIPES adotou o desenvolvimento também na linguagem Snek. Essa linguagem é extremamente enxuta e foi projetada justamente para ser executada em processadores com pouquíssimos recursos de *hardware*, nem o suficiente para rodar MicroPython (SNEKLANG, 2022).

## 2.1 REFERENCIAL TÉCNICO

O BIPES é uma aplicação *open source* web com base em HTML (*HyperText Markup Language*) e JavaScript. Sua concepção é um passo a mais vindo de outros projetos também *open source*, como Blockly e MicroPython (BIPES, 2022). É capaz de programar vários tipos diferentes de placas, embora o presente projeto se limitou a usar o Arduino Uno. Essa placa tem recursos bem limitados a ponto de não

conseguir executar MicroPython, e serviu de base para que pudessem ser desenvolvidos e testados novos módulos do BIPES em Snek. O módulo de acionamento do servo motor usa instruções em Snek para controlar as saídas PWM (*Pulse-Width Modulation*) do Arduino.

### 2.1.1 A Programação Em Blocos

Blockly é “uma biblioteca JavaScript usada para construir editores de programação visual” (BLOCKLY, 2022). Seus blocos são capazes de gerar código sem necessidade de se preocupar com sintaxe, reduzindo um dos problemas de programação. Vale lembrar que Blockly não é uma linguagem de programação, e sim que seus blocos são criados carregando os elementos da linguagem de programação que seus desenvolvedores querem (ATHAWALE, 2018). Blockly está integrado no BIPES convertendo os blocos gerados pelo time de desenvolvimento da plataforma em código Python, MicroPython, CircuitPython e Snek.

### 2.1.2 A Programação Em Texto

A linguagem usada no contexto desse projeto é a Snek. Essa linguagem foi desenvolvida pensando nos dispositivos embarcados que possuem poucos kB de memória *flash* e RAM[3], como por exemplo o Arduino Uno, que possui somente 32 kB de memória *flash* e 2 kB de RAM. A linguagem se baseia na semântica da linguagem Python, mas é muito mais reduzida, contendo uma pequena parcela do que Python é capaz (SNEKLANG, 2022). Nesse projeto, um interpretador Snek é instalado no microcontrolador utilizado, e com isso o código gerado pelo BIPES pode ser executado.

Ao adicionar o suporte para Snek na plataforma, o time do BIPES também disponibilizou uma forma simplificada de se instalar Snek em placas Arduino sem a necessidade de software adicional. Através da página *web* Snek *Web Uploader* (SNEK UPLOADER, 2022) é possível conectar um Arduino através da porta serial e

[3] Uma lista mais completa de quais dispositivos são suportados pela linguagem Snek pode ser encontrada aqui: https://github.com/keith-packard/snek#supported-hardware. Acesso em 26 abr 2022

gravar o Snek versão 1.5 na memória da placa através de poucos cliques. Uma visualização da página é apresentada na Figura 4.

**Figura 4** – Snek *Web Uploader*


Fonte: SNEK UPLOADER, 2022

### 2.1.3 O Hardware

Toda a parte de *software* é usada para controlar elementos de *hardware*. É o *hardware* que executa tarefas no mundo físico como, por exemplo, elétrons fluindo em uma trilha ou hastes empurrando ou puxando objetos.

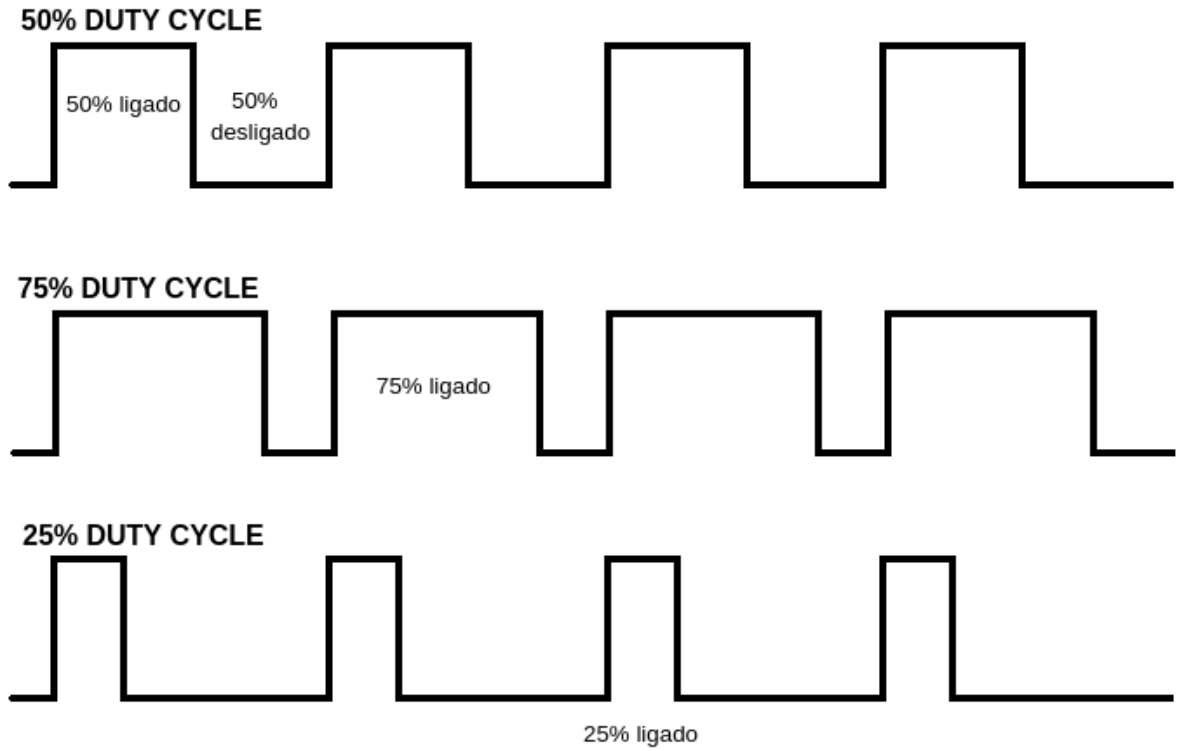
#### 2.1.3.1 O Motor

Um servo motor RC (*Radio Control*), daqui para frente referido somente por “servo motor”, é um servo motor utilizado em aplicações controladas remotamente, como aeromodelos e robôs, que permite o ajuste do ângulo do seu eixo com grande grau de precisão e eficiência. Nesse sistema, o controle da velocidade e ângulo do eixo é feito através de retroalimentação, onde a corrente de saída que aciona o

servo motor é usada como entrada de referência para controle, gerando mais ou menos corrente na saída de acordo com a necessidade.

Este tipo de servo motor geralmente têm atuação de 0º até 180º, e usam um sinal PWM como sinal de acionamento (SOCIETY OF ROBOTS, 2022). Um sinal PWM é um sinal elétrico cuja razão entre o seu semiciclo de valor alto e seu semiciclo de valor baixo, denominada *duty cycle*, pode ser ajustada. Com isso, o valor de potência entregue pelo sinal varia de acordo com essa razão. É possível ver exemplos de sinais com *duty cycle* diferentes na Figura 5. Para o acionamento de um servo RC, o sinal deve ter seu semiciclo alto entre 1ms e 2ms, e o seu período deve ter por volta de 20ms.

**Figura 5** – Exemplos de sinais PWM




Fonte: Elaborada pelo autor

### 2.1.3.2 O Arduino

Arduino é uma empresa que projeta, produz e dá suporte a dispositivos eletrônicos e software (ARDUINO, 2022). Sua popularidade se formou através da criação de placas com todos os recursos necessários para gravação de novos programas, unidos a outros adicionais úteis como interface USB. Isso simplificou todo o processo de criação de novos projetos, pois não era mais necessário

programar os microcontroladores em linguagem Assembly e usar uma placa dedicada para gravar esse código.

Os Arduinos vêm ganhando bastante notoriedade nos últimos anos, crescendo em número de busca na Internet, em presença em artigos científicos e até no uso em meio industrial (EL-ABD, 2017). As placas Arduino são muito usadas em escolas e universidades, ou por pessoas que têm por *hobby* criar projetos caseiros.

### 2.1.4 A Comunicação

Por se tratar de uma plataforma integrada com IoT, o BIPES possui como parte integral do projeto meios de comunicação entre sistemas e a Internet, permitindo o acesso de dados e o controle de placas de forma totalmente remota. Isso é possível devido a uma combinação de ferramentas integradas ao BIPES, sendo elas: WebSocket, WebREPL (JUNIOR et al., 2020), Freeboard e MQTT.

WebSocket (WEBSOCKET, 2022) é uma tecnologia que possibilita a comunicação bidirecional entre o navegador do usuário e um servidor. Essa tecnologia cria um canal de comunicação persistente que permite a troca de mensagens entre as partes através de uma única conexão. Após a primeira troca de mensagens onde o navegador e o servidor se reconhecem (chamado *handshake*), o canal fica aberto para envio e recebimento de mensagens. Os dados de uma mensagem são agrupados em um cabeçalho e em um corpo da mensagem.

WebREPL consiste em um console de comandos usando *Read-Eval-Print Loop* (REPL) sendo executado no navegador. Esse console é capaz de receber linhas de comando, interpretá-las e exibir dados relativos aos comandos recebidos. Combinando com o uso de WebSocket e Serial API (*Application Programming Interface*), é possível enviar e receber comandos e dados de uma placa de sistemas embarcados conectada em uma porta USB através do console no navegador *web*. Os comandos geralmente são linhas de código escritas pelo usuário ou geradas automaticamente pelos blocos de programação.

Freeboard (FREEBOARD, 2022) é um sistema HTML que provê interface visual para operação de painéis de dados. Através dessa ferramenta é possível criar e gerenciar painéis de dados, alocando as diferentes formas que esses dados serão

coletados e customizando como esses dados serão exibidos para o usuário.

MQTT é um protocolo de comunicação muito utilizado para IoT. O protocolo utiliza um modelo de comunicação “publicador-assinante”, onde um cliente do sistema pode ser um publicador, enviando mensagens para um tópico qualquer, ou pode ser um assinante, que marca quais tópicos quer acompanhar. Quando um publicador envia uma mensagem para determinado tópico, todos os assinantes desse tópico recebem a mensagem. Um servidor especial denominado *broker* fica responsável em receber, filtrar e distribuir as mensagens.

# 3 MATERIAIS E MÉTODOS

Este projeto foi desenvolvido usando uma placa Arduino Uno com processador ATMEGA328P, um cabo USB, um servo motor SG90, um potenciômetro de 1 kΩ e um capacitor de 100 µF. O desenvolvimento do código foi feito em um computador através da IDE (*Integrated Development Environment*) *Visual Studio Code*. O servo motor SG90 é pequeno, leve, funciona com tensão de 5V e tem torque de 2,5 kgf.cm.

## 3.1 ARDUINO UNO

O Arduino Uno é uma placa de prototipação equipada com o *chip* ATMEGA328P. O nome Uno vem do italiano (um) e faz referência à primeira versão do *software* de ambiente de programação Arduino Software (IDE), ambos tidos como referência para “Arduino”. É uma placa versátil, possuindo 14 pinos digitais de entrada e saída, 6 dos quais funcionam como saída PWM, e 6 pinos de entrada analógica. Possui 32 kB de ROM e 2 kB de RAM. Um exemplar está à mostra na Figura 6. O Arduino Uno pode ser programado usando a Arduino IDE através do protocolo STK500 quando conectado ao computador através de um cabo USB. A interface USB é também fonte de alimentação para seus circuitos (ARDUINO UNO, 2022).

**Figura 6** – O Arduino Uno


Fonte: ARDUINO UNO, 2022

## 3.2 O SOFTWARE DA PLACA ARDUINO

Ao inicializar, a placa executa um pequeno *software* cuja função é carregar as informações do sistema principal e em seguida transferir o controle do *hardware* para esse sistema. Este pequeno *software* é denominado *bootloader* (ou *boot loader*) (GNU, 2022). Nas placas Arduino UNO, o *bootloader* ocupa um espaço considerável já que o espaço disponível da placa é muito reduzido. Portanto, foi optado por se carregar um novo *bootloader* menor, liberando assim mais espaço para o propósito principal: ler e executar programas em Snek. O *bootloader* escolhido foi o Optiboot versão 8.0, pois contém somente 512 *bytes* e especificações compatíveis com o chip ATMEGA328P (OPTIBOOT, 2022). Vale a menção de que muitas das placas Arduino vendidas atualmente já possuem o Optiboot como *bootloader* padrão, embora esse não tenha sido o caso da placa utilizada neste trabalho.

### 3.2.1 Instalando o Optiboot

O passo-a-passo descrito aqui está presente no repositório oficial do projeto Optiboot no GitHub[4], em inglês, e deve ser seguido somente se a placa Arduino a ser utilizada não possui o *bootloader* já previamente instalado, o que pode ser identificado através de erros causados por falta de espaço.

Para obter o Optiboot é preciso ir à página das versões no Github do Optiboot e selecionar a versão desejada. Estando lá, copiar o *link* para o arquivo de extensão json associado à versão. Com o *link* copiado, abrir o ambiente de desenvolvimento oficial do Arduino (Arduino IDE). Selecionar a opção “Preferências” e adicionar o *link* copiado no campo “URLs do Gerenciador de Placas Adicionais” (*Uniform Resource Locator*). Após esses passos, haverá uma opção para Optiboot no Gerenciador de Placas. Após instalar, o Optiboot está disponível no sistema da Arduino IDE, mas ainda não está na placa.

Para gravar o *bootloader* na placa é necessário uma segunda placa Arduino funcionando como ISP (*In-circuit Serial Programmer*). ISP é um modo de gravação

[4] Disponível em https://github.com/Optiboot/optiboot#to-install-into-the-arduino-software. Acesso em 26 abr 2022

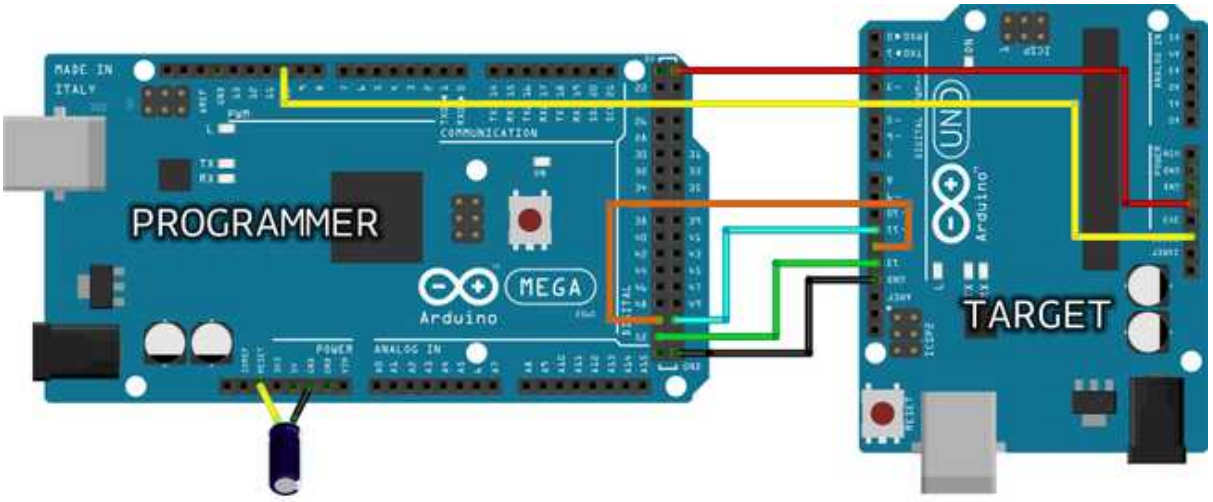
que acessa toda a memória flash do Arduino, incluindo a área reservada para o bootloader, através de terminais específicos. Os terminais envolvidos neste processo estão listados na Tabela 1. O Arduino IDE possui um código de exemplo salvo para essa finalidade chamado "*Arduino as ISP*". Basta executar esse código em um Arduino auxiliar e conectá-lo ao Arduino que receberá o novo *bootloader* conforme indicado na Figura 7. Nesse projeto foi utilizada a placa Arduino Mega 2560 como placa auxiliar. Uma vez feita a conexão, é selecionado na Arduino IDE o Optiboot para *chips* de 32 pinos, o processador ATMEGA328P, a porta USB correspondente ao Arduino Uno, selecionar a opção "Programador: *Arduino As ISP*" e por fim "Gravar *Bootloader*". Ao final, não havendo erros, o Optiboot estará instalado na placa, liberando mais espaço da memória para os programas.

**Tabela 1** – A conexão para gravar bootloader

| Placa Arduino | MOSI | MISO | SCK | Level |
|---|---|---|---|---|
| UNO ou Duemilanove | 11 ou ICSP-4 | 12 ou ICSP-1 | 13 ou ICSP-3 | 5V |
| Mega1280 ou Mega2560 | 51 ou ICSP-4 | 50 ou ICSP-1 | 52 ou ICSP-3 | 5V |

Fonte: Arduino Built-in Exemples[5]

**Figura 7** – Interconexão de dois Arduinos para programação ISP




Fonte: Arduino Built-in Exemples

[5] Disponível em: <https://docs.arduino.cc/built-in-examples/arduino-isp/ArduinoISP>. Acesso em: 04 abr 2022

### 3.2.2 Instalando o Snek

O *software* a ser executado na placa é o interpretador Snek. Com ele, os programas gerados no blocos do BIPES e convertidos em instruções Snek são finalmente transformados em ações que controlam o *hardware* do Arduino UNO. O binário do interpretador pode ser baixado já compilado diretamente da página de downloads ou, caso o desenvolvedor queira fazer alterações na linguagem Snek, pode ser construído a partir do código-fonte disponibilizado. A seção 2.1.2 apresentou uma forma mais simples de instalar o binário do interpretador através do uso do Snek *Web Uploader*, disponibilizado pelo time do BIPES. Para isso, basta acessar a página, conectar a placa através de uma porta serial, clicar no botão "*Upload Snek to nano/uno/pro/pro-mini*" e aguardar a gravação terminar. Caso não faça nenhuma alteração, não há diferença entre baixar a versão já compilada ou construir manualmente. Caso escolha usar o Snek *Web Uploader*, o usuário estará restrito a usar a versão disponibilizada pelo time do BIPES.

Para construir o Snek a partir do código-fonte são necessários vários outros *softwares* que controlam o sistema operacional com o intuito de gerar o binário que contém as instruções do interpretador.

O primeiro passo é se certificar que todas as dependências para construir o Snek estão cumpridas. São elas:

- Lola: *parser* de linguagens do tipo LL
- Gawk: ferramenta de re-formatação de dados
- Gcc-avr: compilador C para microcontroladores AVR
- Avr-libc: biblioteca C para microcontroladores AVR
- Gcc-arm-none-eabi: compilador para microcontroladores ARM
- Gcc-riscv64-unknown-elf: compilador RISCV
- Picolibc: biblioteca C para sistemas embarcados com pouca RAM
- Python3 : interpretador da linguagem de programação Python
- Pyserial: módulo de acesso à porta serial via Python
- Python curses: módulo de manuseio de comandos no terminal
- Readline: biblioteca para trabalhar com linhas de comando

Além disso, Picolibc possui como dependências para sua construção:

- Meson: sistema para construção de código à partir de instruções

- Ninja: sistema para construção de código de alta velocidade

Uma vez que todas as dependências estão instaladas, é possível construir o código do Snek.

Após a construção com sucesso, um arquivo “snek-uno-<versão>.hex” é gerado. Para esse projeto foi utilizada a versão 1.7, portanto o arquivo correspondente é “snek-uno-1.7.hex”. Este é o arquivo contendo as instruções do interpretador Snek. Com um terminal aberto no diretório em que esse arquivo se encontra, é executado um comando para a gravação do interpretador na placa:

```
avrdude -p ATMEGA328P -c arduino -P "/dev/ttyUSB0" -b 115200 -D -U flash:w:"snek-uno-1.7.hex:i"
```

Atenção para o argumento -P, pois deve ser informado a porta a qual a placa está conectada ao computador. Ao final desse passo, a placa possui o interpretador Snek instalado e está pronta para receber comandos Snek em seu console.

## 3.3 O SOFTWARE DO BIPES

Por ser um projeto *open source*, o código-fonte do BIPES encontra-se disponível para download[6], e com isso é possível inserir melhorias, expandindo a capacidade de se ensinar sistemas embarcados. No projeto aqui proposto foram inseridos novos blocos para fazer o acionamento de um servo motor via Snek.

### 3.3.1 Bloco Do Servo

O BIPES já possui um bloco de acionamento de servo motor para outras linguagens. Portanto, foi possível basear-se nesse bloco para adicionar o bloco Snek.

Para adicionar um bloco no BIPES é necessário modificar três arquivos:

- block_definitions.js: responsável pela definição dos pinos e conexões dos blocos
- Arduino.xml: responsável por organizar a visualização dos blocos
- generator_stubs.js: responsável pela conversão dos blocos para código

[6] Repositório: https://github.com/BIPES/BIPES

Analisando o arquivo block_definitions.js, procura-se um bloco já existente de servo motor para tomar como referência. Esse bloco é o “init_servo”. Ele inicializa um servo motor em relação a um pino do Arduino, e depois outro bloco faz o movimento do eixo para um ângulo determinado. O bloco de acionamento do servo motor via Snek pode fazer as duas coisas, portanto a alteração feita foi adicionar um pino que recebe um valor de ângulo no bloco, efetivamente combinando os dois blocos num só.

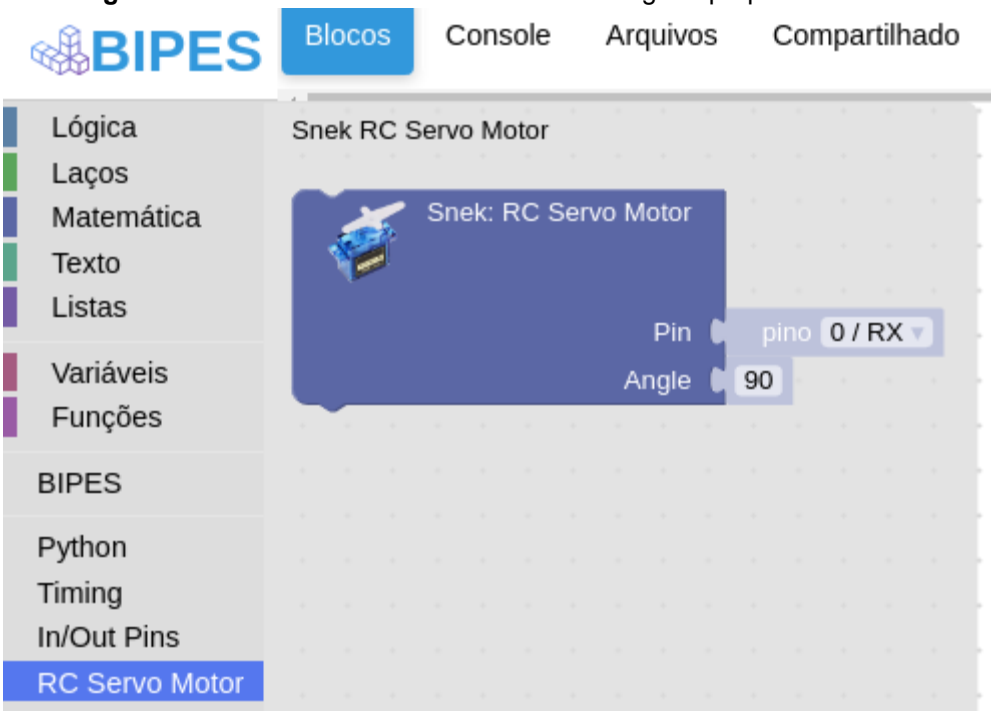
O próximo passo foi alterar o arquivo “arduino.xml”, para adicionar o novo bloco na plataforma. Aqui é importante frisar que uma nova categoria foi criada para comportar o bloco do servo motor. Outro fator interessante foi a adição do parâmetro “*shadow*” para já dar uma sugestão de qual é o tipo de dado que vai ser inserido no bloco: tipo “pino” para o “pin” e tipo “número” para o “angle”. Com isso, o bloco gerado já vem com os campos pré configurados, restando ao usuário somente a tarefa de colocar os valores desejados, sem a necessidade de conectar blocos referentes aos valores.

Por fim, foi feita a adição do código do bloco do servo motor no arquivo “generator_stubs.js”. Os valores dos pinos são guardados em variáveis para serem adicionados nas *strings* das linhas de código Snek. Esses códigos serão enviados para o console do BIPES, e então interpretados e executados pelo Arduino. Após esse último passo, o bloco está disponível para uso na plataforma, como pode ser visto na Figura 8.

Vale ser mencionado que Snek é uma linguagem em constante desenvolvimento, e mudanças podem acontecer em versões mais recentes que alterem a estrutura dos comandos Snek, tendo a possibilidade de inviabilizar o funcionamento desse bloco. Caso o usuário instale versões diferentes da 1.7 e o bloco não funcionar, há duas formas de se resolver:

1. Modificar o código-fonte do Snek e compilar novamente, revertendo manualmente a mudança introduzida. Essa opção está fora do escopo deste trabalho.
2. Utilizar uma versão cujo funcionamento seja consistente com o apresentado neste trabalho, como a versão 1.7.

**Figura 8** – O bloco do servo motor em sua categoria própria no BIPES




Fonte: Elaborada pelo autor

### 3.3.2 Adicionando O Código Na Inicialização

Os códigos criados via blocos são lidos no console existente na plataforma pelo interpretador Snek, para então serem executados na placa. Com isso, é necessário que a placa esteja conectada toda vez que se deseja executar o programa criado. A plataforma BIPES e o Snek dão a opção de escrever esse código diretamente na memória da placa, fazendo com que ele seja executado logo após a inicialização do interpretador pelo *bootloader*, permitindo execução autônoma, independente do PC.

O Snek oferece a opção de gravar o código lido diretamente na EEPROM (*Electrically Erasable Programmable Read-Only Memory*) da placa. Para isso, é necessário usar a função já disponibilizada pela linguagem “eeprom.write()”. Ao utilizar essa função, o interpretador entra no modo de leitura específica para gravação, coletando todos os caracteres enviados no console até que ele receba o caractere que representa “Fim de Transmissão” (Hex: 0x04, Ctrl+D no terminal). Ao final disso, as linhas de código enviadas são gravadas na EEPROM da placa da forma que estão, independente de se há erros de sintaxe ou semântica no programa. Ao iniciar a placa, seja ligando ou através do botão de *reset*, o código é

executado logo depois do interpretador ser carregado. Caso haja algum erro, ele é exibido no console e a execução é interrompida. Esse erro será exibido toda vez que houver uma nova inicialização até que o código errôneo seja sobrescrito ou apagado com o comando “eeprom.erase()”. Vale lembrar que, caso a placa esteja desconectada do computador no momento de inicialização, os comandos são processados normalmente pela placa mesmo que não seja possível ver no console.

Para que essa funcionalidade esteja presente no BIPES, foi adicionado um controle de fluxo no código responsável pelo botão “Save” do canto superior direito da guia “Arquivos”. Este código está presente no arquivo “/ui/core/utils.js”, e a função responsável é a “put_file()”.

Foi necessário adicionar uma verificação de qual *workspace* estava ativo, consequentemente qual placa foi escolhida. Essa verificação assegura dois pontos importantes:

- Ao usar o Snek no Arduino, o caractere referente a “Nova Linha” (Hex: 0x0A, Ctrl+J no terminal) deve ser “\n”, portanto a substituição “\n por \r” foi restringida a ser feita somente se o *workspace* não for o de uma placa Arduino Uno.
- Todo o código referente às outras placas não funcionam no Snek, além de o Snek precisar receber o caractere de “Fim de Transmissão” ao final da programação. Com isso, foi aberto uma condicional para *workspaces* de placas Arduino Uno que executa o código de gravação, insere todo o código presente na aba “Arquivos” e depois envia um caractere de “Fim de Transmissão”. Logo em seguida, a execução da função “put_file()” é interrompida, pois tudo que era necessário para a gravação foi feito.

### 3.3.3 Snek Com EasyMQTT

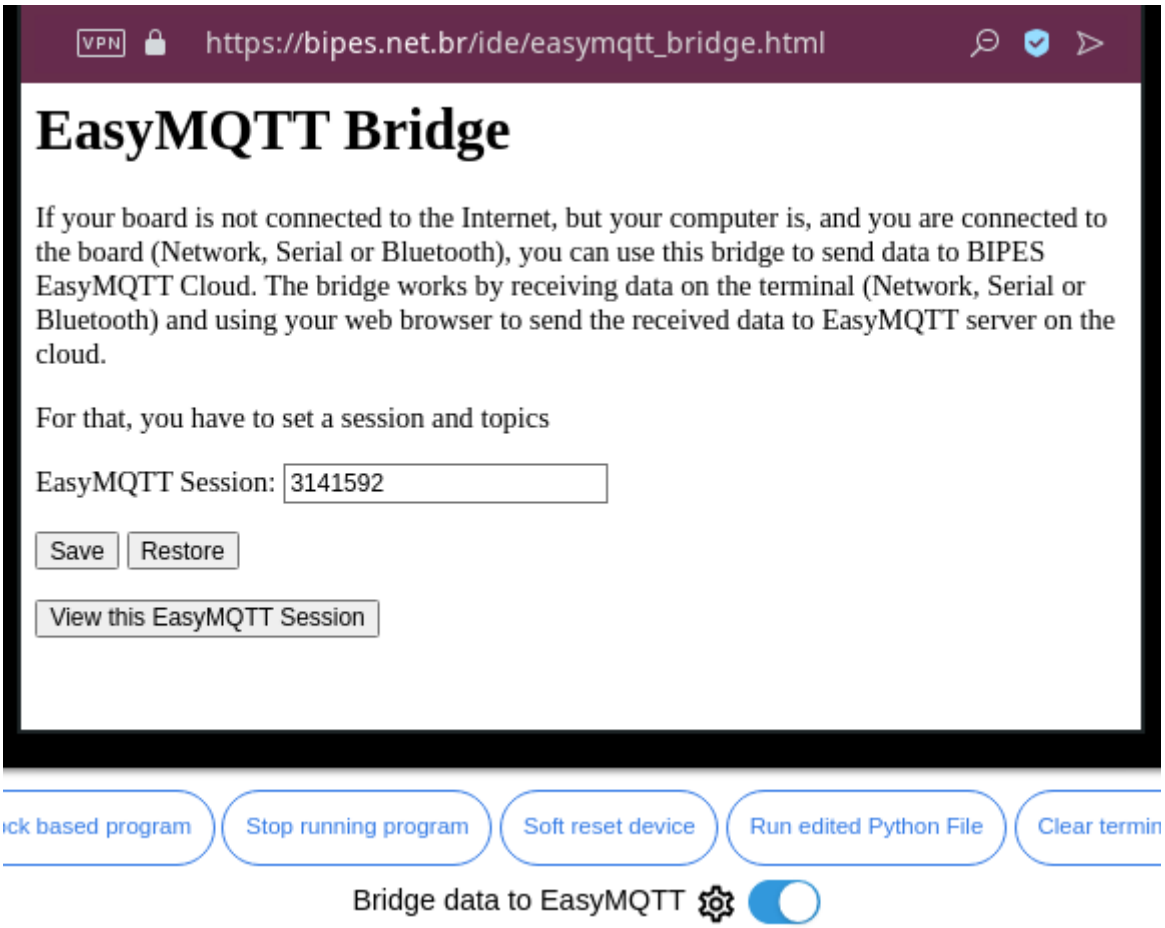
A integração com o EasyMQTT foi feita de maneira extremamente simples. A plataforma já oferece suporte, portanto nenhuma adição foi necessária. Entretanto, como o Snek é uma novidade, viu-se interesse em validar a interação da linguagem com a ferramenta.

Para verificar seu funcionamento, foi colocado um bloco “mostrar na aba IoT” no meio dos blocos de código do BIPES. À esse bloco foi designado um ID e foi

informado o valor do ângulo do servo motor para ser monitorado. Ao ser executado, o código começa a imprimir o valor do ângulo em um formato especial para transmissão.

Na aba do console no BIPES, é selecionado a opção "*Bridge data to EasyMQTT*" e depois clicar na engrenagem, a qual abre uma nova janela com opções de configuração da sessão MQTT, conforme visto na Figura 9. Nessa janela, selecionamos qual é o valor da sessão para qual os dados que estão sendo exibidos no console serão enviados. Uma vez selecionada a sessão, clica-se em "*Save*" e em seguida "*View this EasyMQTT session*". Uma nova aba é aberta com a sessão selecionada. Essa sessão não é exclusiva da aba atual do BIPES, e pode ser acessada de qualquer navegador conectado à internet.

**Figura 9** – A configuração da sessão EasyMQTT.



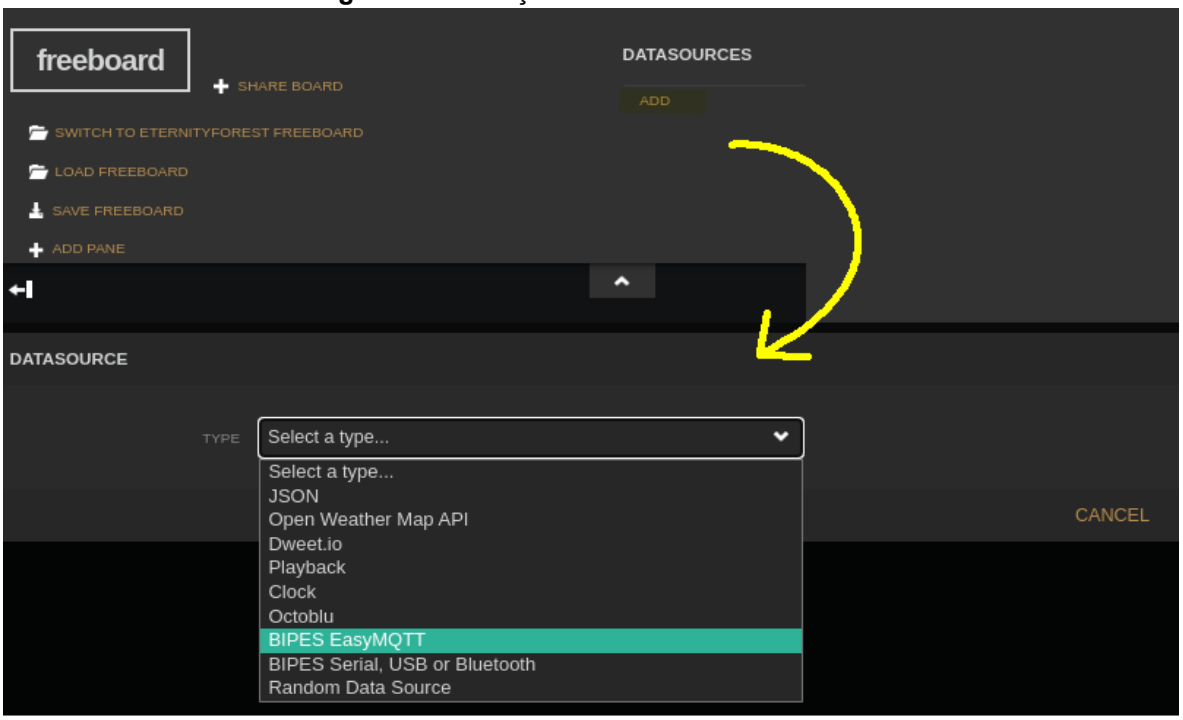

Fonte: Elaborado pelo autor

### 3.3.4 Widget para o Freeboard

Uma outra forma de visualizar os dados enviados pela Internet seguindo a ideia de IoT é através de painéis gerados pelo Freeboard. Freeboard é uma ferramenta poderosa no campo de visualização de dados. Sua operação também é bem simples, e os resultados são muito úteis. Seus painéis funcionam tanto com dados online quanto com dados locais, através de comunicação serial.

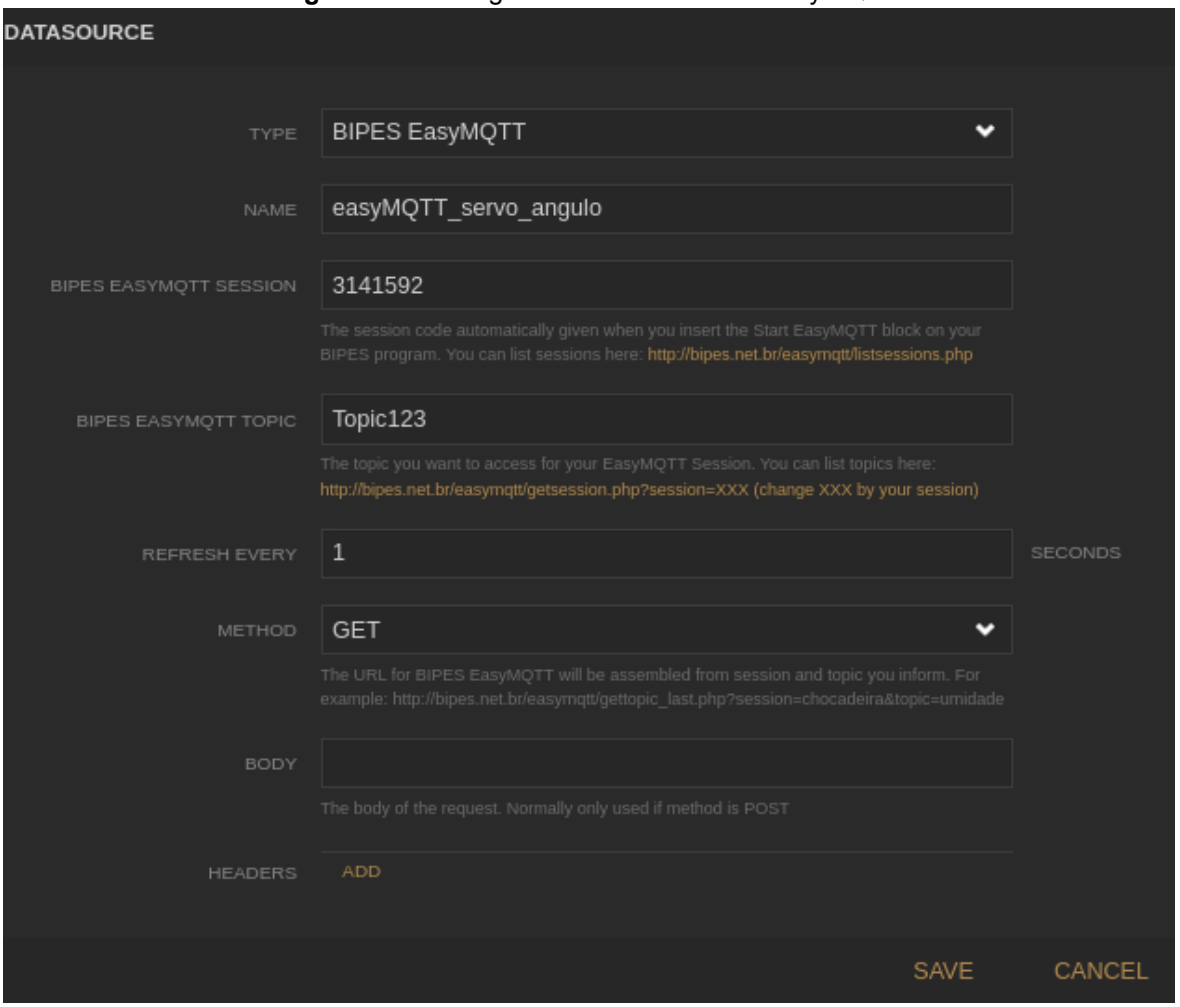
Para iniciar, é preciso que haja uma fonte de dados sendo carregada no console. Conforme dito na seção 3.3.3, foi utilizado o bloco “mostrar na aba IoT” do BIPES para que os dados sejam transmitidos. Como é desejado que sejam enviados pela internet, uma sessão EasyMQTT deve estar ativa, seguindo os passos descritos na seção 3.3.3. Com esses dados prontos, é feita a configuração do Freeboard para exibição. Na área “*data sources*”, a opção “*Add*” é selecionada. Na janela que abre é selecionado o tipo “*BIPES EasyMQTT*”, como está na Figura 10. Isso libera outros campos para serem preenchidos, tal qual visto na Figura 11. O campo “*name*” é a identificação do dado que será coletado. No campo “BIPES *EasyMQTT Session*” é necessário informar a sessão EasyMQTT criada. O campo “*BIPES EasyMQTT Topic*” é o campo referente ao número de identificação do dado, que é formado pela palavra “*Topic*” concatenada com o mesmo número informado no bloco “mostras na aba IoT”. O próximo campo é a taxa de atualização, que define que o dado vai ser coletado novamente a cada “x” segundos, sendo “x” o valor informado no campo. No campo “*Method*” é selecionada a opção “*GET*” para que os dados sejam coletados do servidor online EasyMQTT. Clicar em “*Save*” para guardar essas opções.

**Figura 10** – Seleção de uma nova *data source*




Fonte: Elaborada pelo autor

**Figura 11** – Configurando a *data source* EasyMQTT



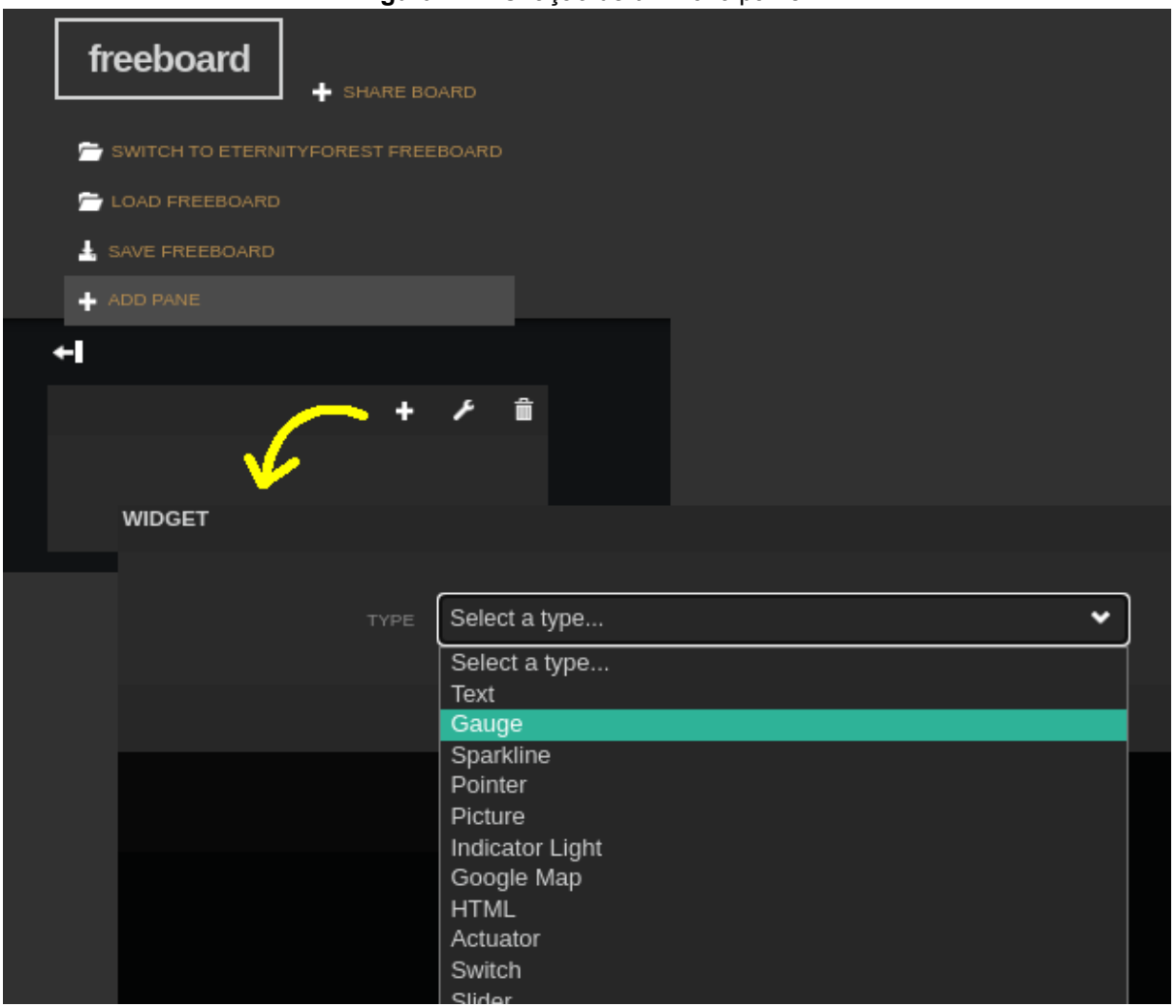

Fonte: Elaborada pelo autor

Com os dados prontos, é necessário agora configurar os painéis. Uma boa forma de exibição do ângulo do servo motor é através do painel do tipo “*Gauge*”, pois seu formato em meia-lua possui correspondência aos 180° que o servo motor tem de atuação. Selecionamos esse tipo de painel clicando na opção “*Add Pane*”, e depois no ícone “+” no painel que foi adicionado. Após isso, selecionamos o tipo “*Gauge*”, igual ao apresentado na Figura 12. Após novas opções serem liberadas, em “Title”, colocamos o título do painel, que nesse caso foi “*Ângulo do servo motor*”. Em “*Value*”, é possível selecionar o valor que foi configurado em *data sources* clicando na opção “*Data Source*” e selecionando o mesmo “*name*” dado, em seguida é selecionada a opção “result”, depois o índice do dado, que nesse caso foi “0”, e por fim selecionar o campo “data”. No campo “*Units*” é possível colocar a unidade representativa do dado, embora esse campo não seja obrigatório. Em seguida, os campos “*Minimum*” e “*Maximum*” levam o alcance dos valores possíveis do dado a ser representado, que no caso do servo motor é no mínimo 0 e no máximo 180. Com todos os dados preenchidos, como visto na Figura 13, clique no botão “*Save*” para guardar essas opções. Agora é exibido o painel representativo no formato de medidor que vai de 0 a 180. Quando o valor do dado é atualizado, ele é enviado via EasyMQTT pela Internet para o *data sources*, e o painel faz a coleta desse dado para preencher o medidor de acordo.

Outra forma de uso da aba “IoT” é enviar comandos através de comunicação serial para o console, e os painéis do Freeboard também são capazes disso. Para isso, cria-se um novo painel, novamente no botão “*Add Pane*” e em seguida no ícone “+”. Dessa vez, o tipo selecionado é “HTML”. As opções desse tipo de painel são “HTML” e “Height Blocks”, a primeira sendo a parte funcional do bloco e a outra o ajuste de tamanho do bloco. Para a parte funcional, é selecionada a opção “.js Editor”, que abre um painel de edição de texto para que um código HTML/JavaScript seja adicionado, gerando assim um bloco com elementos interativos. Foram selecionados dois elementos para esse bloco: uma barra deslizante e um botão. Há também um valor numérico correspondente à posição da barra deslizante. O funcionamento do código adicionado é simples: o usuário consegue ajustar a posição da barra deslizante, o que corresponde a um valor de ângulo, e quando ele apertar o botão esse valor será enviado para o console via comunicação serial, o console enviará os comandos para a placa e o servo motor irá se ajustar à posição solicitada. Os comandos são criados através de *strings* e enviados para o Arduino

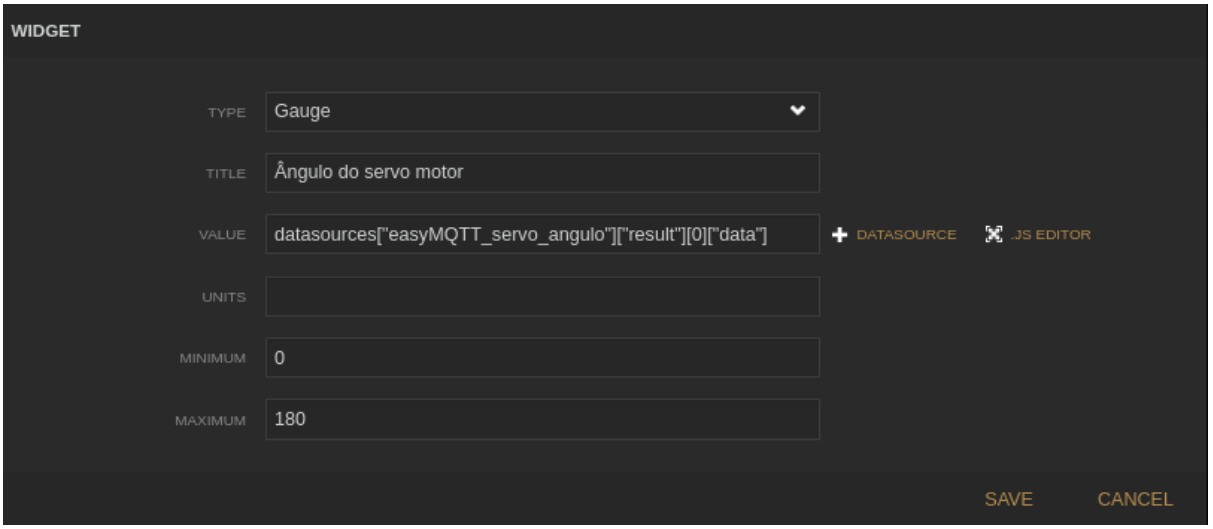
usando uma função chamada “bufferPush()” presente no projeto, que é responsável por carregar os códigos no console de comando.

**Figura 12** – Criação de um novo painel




Fonte: Elaborada pelo autor

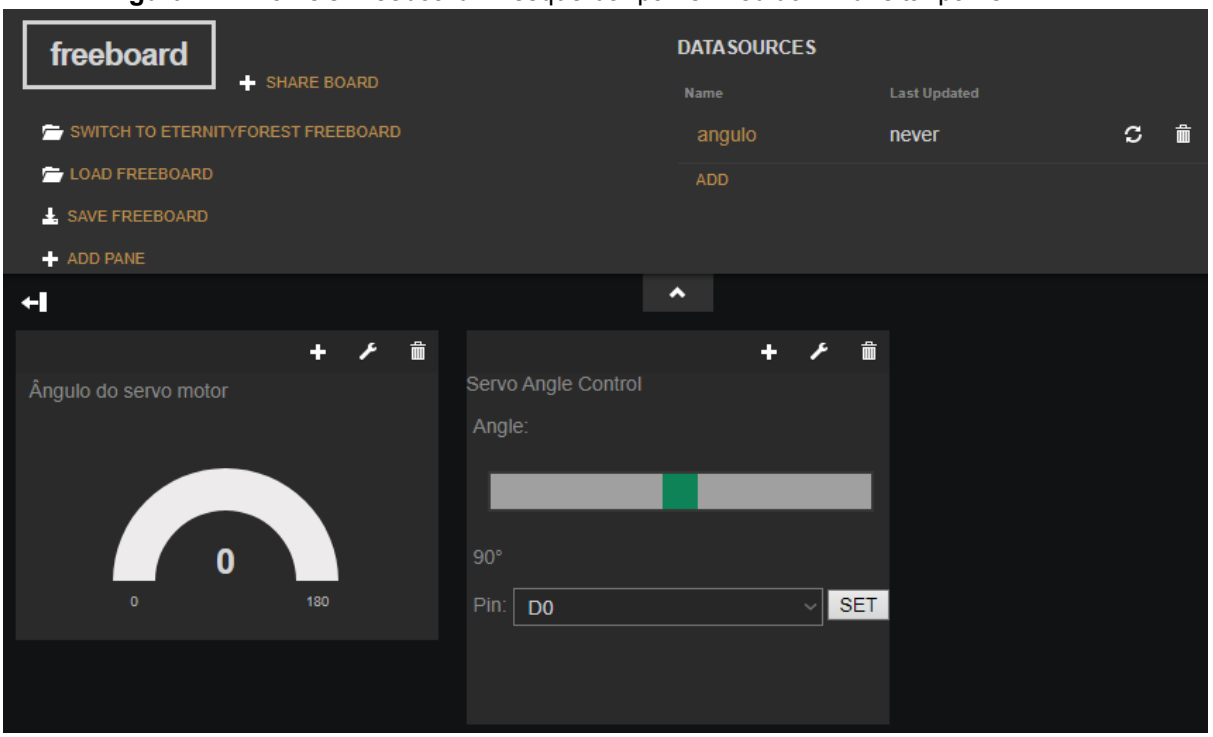
**Figura 13** – Configurando o painel do medidor




Fonte: Elaborada pelo autor

Após os dois painéis serem criados, foi usada a opção "*Share Board*" para gerar um *link*[7] referente ao conjunto de painéis gerados. Os dois painéis podem ser vistos na Figura 14, e o código para o painel HTML de controle do ângulo está no Apêndice B.

**Figura 14** – Painéis Freeboard. À esquerda: painel medidor. À direita: painel HTML.




Fonte: Elaborado pelo autor

[7] Link para o *dashboard*: http://bipes.net.br/freeboard#dhhr45

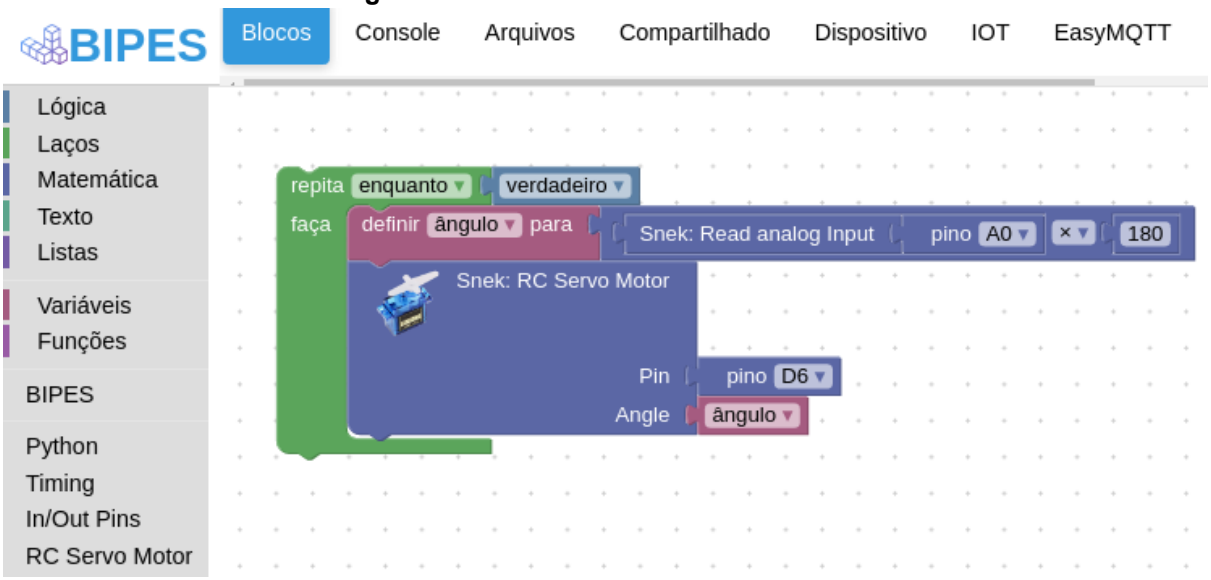
# 4 RESULTADOS

As mudanças propostas por esse trabalho foram adicionadas ao repositório do projeto no GitHub. Com elas ativas no site, foram feitos testes para comprovar suas funcionalidades.

O primeiro teste foi verificar que a plataforma estava realmente executando código Snek e que era capaz de controlar o Arduino através dessa linguagem. Para isso, a placa foi conectada a um computador via USB e, através do console, comandos simples em Snek foram enviados. O Bipes foi capaz de identificar o interpretador Snek presente na placa, e a placa foi capaz de executar os comandos enviados pelo BIPES.

O próximo teste foi verificar o correto funcionamento do bloco referente ao servo motor. O bloco possui uma entrada do tipo “pino”, que foi configurada para o valor D6, e uma entrada para o ângulo, que recebeu uma variável BIPES. Essa variável foi configurada para receber o valor da operação matemática “180 vezes o valor da entrada analógica A0”, cujo alcance dos valores varia entre 0 e 1 dependendo da entrada de tensão no pino A0, que é ajustada através de um potenciômetro. Isso garante que o valor de ângulo que estará na variável seja um valor entre 0 e 180. Todos esses blocos de ajuste do valor de variável e configuração do ângulo do servo motor foram colocados dentro de um bloco “*while True*”, que tem como consequência fazer com que esse código seja executado indefinidamente até ser interrompido manualmente. O programa em blocos gerado nesse passo pode ser visto na Figura 15. Na placa, o servo motor foi conectado ao pino 6 e o pino ajustável de um potenciômetro foi conectado ao pino A0. Por fim, um capacitor de 100 µF foi colocado entre os terminais de energia do servo motor para que ruídos não causassem interferência no seu funcionamento. O circuito totalmente conectado pode ser visto na Figura 16, e a Tabela 2 mostra os pinos envolvidos. Com a configuração da placa montada e os blocos no BIPES posicionados, foi selecionada a opção “*Run block based program*” e o programa em blocos foi convertido para Snek e enviado para o Arduino. O servo motor passou a ser controlado pelo potenciômetro: ao girar sua haste, o valor entrando no pino A0 era alterado, o que mudava o valor do ângulo na variável, e por fim era transformado em giro do eixo do servo motor através do bloco criado.

**Figura 15** – O bloco do servo motor em uso



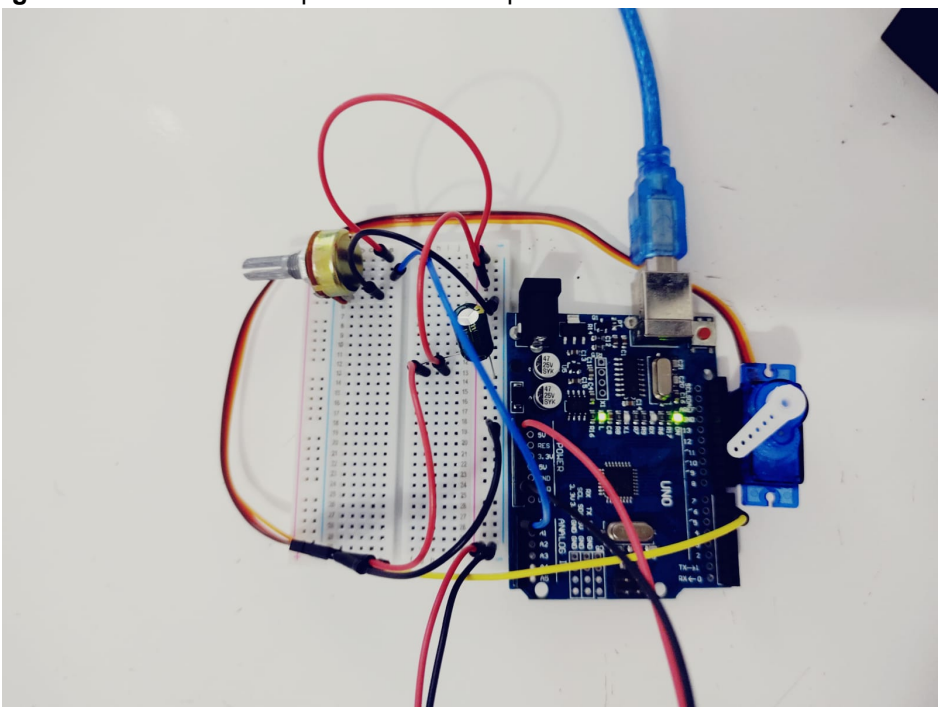

Fonte: Elaborado pelo autor

**Tabela 2** – Pinos para funcionamento do servo motor

| Terminal | Conexão |
|---|---|
| Arduino: 5V | Barramento +VCC |
| Arduino: GND | Barramento GND |
| Arduino: A0 | Pino central do potenciômetro |
| Arduino: D6 | Entrada de sinal do servo motor |
| Potenciômetro: Superior | Barramento +VCC |
| Potenciômetro: Inferior | Barramento GND |
| Servo motor: VCC | Barramento +VCC |
| Servo motor: GND | Barramento GND |
| Capacitor: + | Barramento +VCC |
| Capacitor: - | Barramento GND |

Fonte: Elaborada pelo autor

**Figura 16** – Conexão dos pinos no Arduino para o funcionamento do servo motor



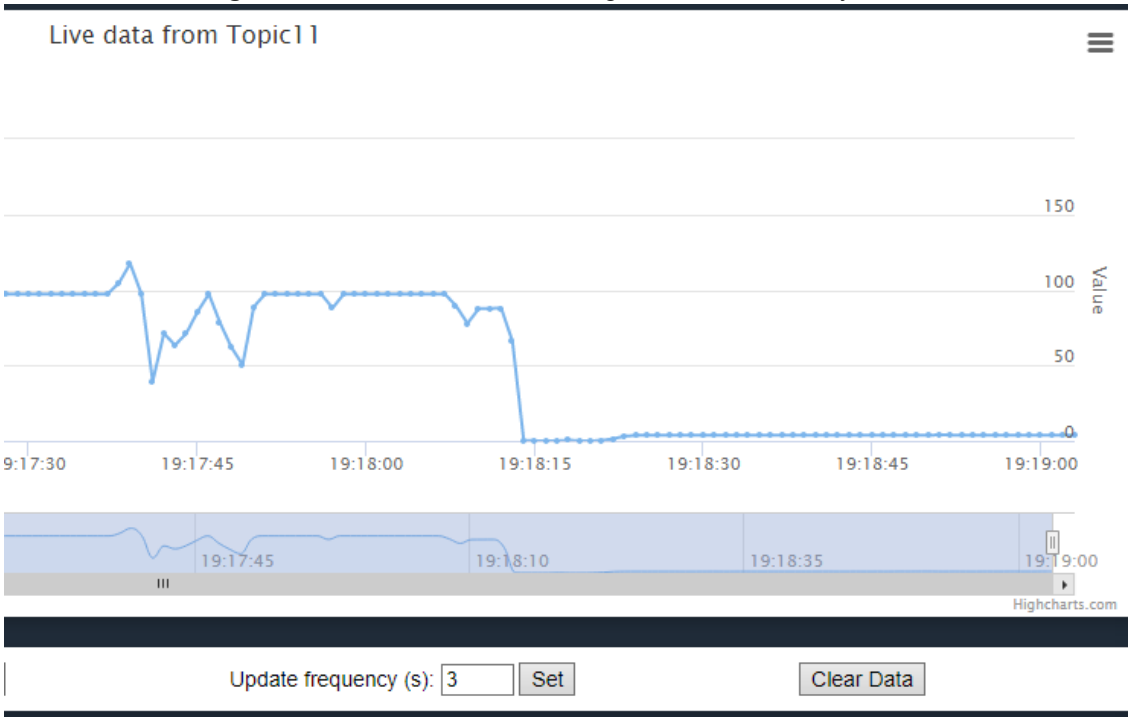
Fonte: Elaborado pelo autor

O terceiro teste cuidou de verificar o funcionamento da integração com o EasyMQTT. Para não haver viés de funcionamento devido ao código estar sendo executado no mesmo navegador que a sessão EasyMQTT, o monitoramento foi feito através da internet em um navegador em outra cidade. Foi feito todo o procedimento descrito em 3.3.3, com exceção da abertura da nova aba para a sessão EasyMQTT, que foi feita por outra pessoa em outra cidade. Foi possível acompanhar a alteração do ângulo através da sessão EasyMQTT corretamente. Como é possível ver na Figura 17, foi possível monitorar o ângulo do servo motor através da ferramenta.

O teste seguinte tratou de verificar o funcionamento do painel tipo “medidor”. Usando o link gerado ao final da seção 3.3.4, foi verificado o valor apresentado no medidor. Ao girar o potenciômetro, foi possível acompanhar a mudança do valor do medidor de acordo com o ângulo do servo motor.

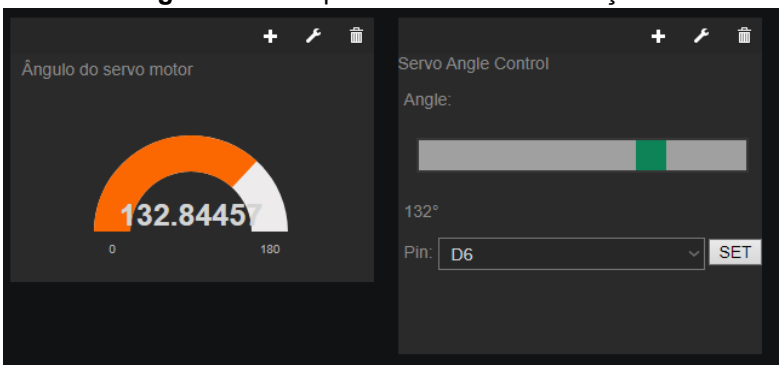
Por fim, o último teste ocorreu com o painel do tipo HTML, verificando se o funcionamento estava de acordo com o esperado. O pino do servo motor foi selecionado com o valor D6, a barra deslizante foi ajustada para vários valores de ângulo e quando apertado o botão “SET”, o servo motor moveu seu eixo corretamente. Os dois últimos testes estão apresentados na Figura 18.

**Figura 17** – Monitoramento do ângulo através do EasyMQTT




Fonte: Elaborada pelo autor

**Figura 18** – Os painéis Freeboard em ação



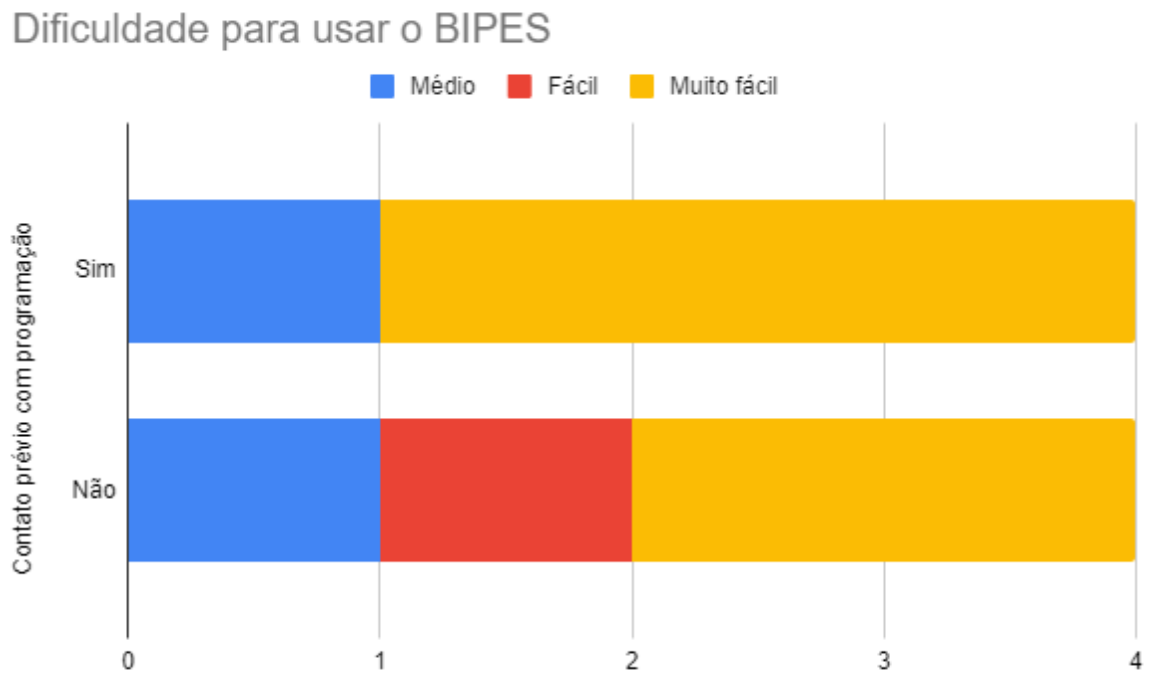

Fonte: Elaborado pelo autor

## 4.1 AVALIAÇÃO DOS USUÁRIOS

Quando colocado em prática, com o projeto sendo usado por usuários comuns, pudemos avaliar alguns dados interessantes. Uma pesquisa foi realizada com 8 pessoas, com idades entre 24 e 55 anos, escolaridade entre Ensino Médio e Ensino Superior, e com diferentes níveis de contato prévio com programação. Metade dos usuários avaliados já tiveram contato com programação anteriormente, e dentre eles, nenhum conhecia a linguagem Snek. Dentre as linguagens de programação citadas por eles, Python foi a mais presente. Quando se trata de conhecimento de programação, não há correlação entre o usuário já ter tido contato

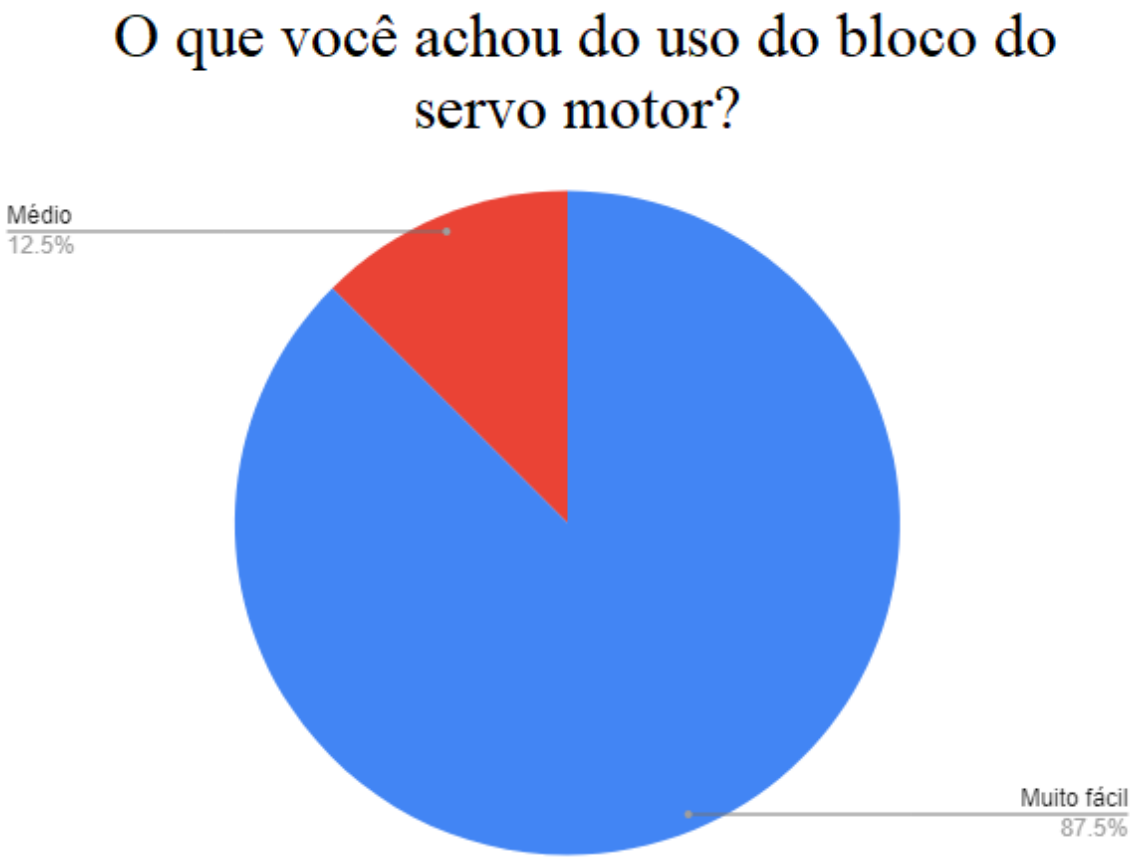
com programação antes e achar o uso do BIPES e programação em blocos fáceis. Embora não tenha sido unanimidade, a maior parte dos usuários, mesmo os que nunca tiveram contato com programação, achou o uso de programação em blocos fácil ou muito fácil, conforme os dados presentes no Gráfico 1. O Gráfico 2 mostra a avaliação exclusivamente do uso do bloco do servo motor, e foi visto que quase 90% dos entrevistados consideraram o seu uso “muito fácil”. Quando perguntados o motivo para essa avaliação, a maior parte citou a simplicidade do bloco (poucos pinos para serem configurados), e a conveniência dos valores padrão (não há necessidade de se plugar novos blocos à ele para funcionar). A maior dificuldade veio no uso do painel IoT: houve dificuldade para configurar os painéis de monitoramento, o que gerou certa frustração. Uma vez configurados, os usuários reportaram que os dados exibidos são fáceis de se entender. Considerando uma escala de facilidade de 1 a 5, onde 5 é muito fácil e 1 é muito difícil, a facilidade em usar o painel IoT da Freeboard teve desempenho melhor para aqueles que já tiveram contato com programação anteriormente, como é mostrado no Gráfico 3.

**Gráfico 1** – Facilidade em usar o BIPES



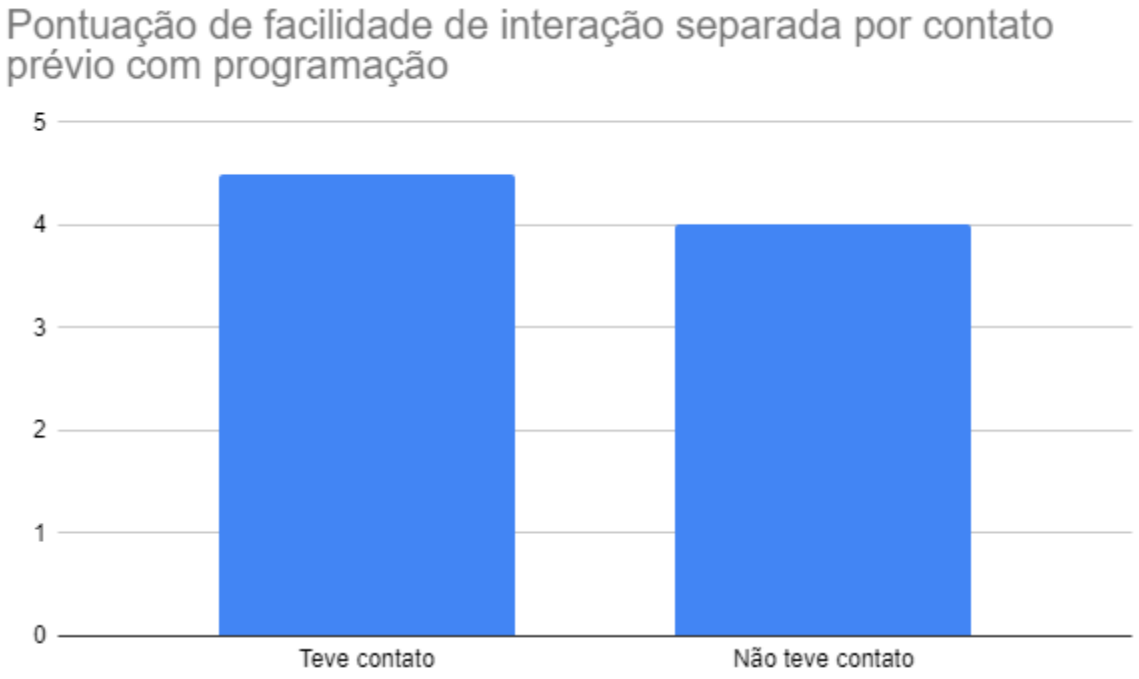

Fonte: Elaborado pelo Autor

**Gráfico 2** – Facilidade em usar o bloco do servo motor no BIPES




Fonte: Elaborado pelo autor

**Gráfico 3** – Facilidade em usar o painel Freeboard dado contato prévio com programação




Fonte: Elaborado pelo autor

O cálculo da pontuação de facilidade foi feito através da média ponderada da avaliação dos usuários.

# 5 DISCUSSÕES

Com a finalização do projeto, foi possível compreender de fato a alta capacidade de expansão e alta versatilidade do BIPES. A simplicidade no processo de adição de novos blocos, de novas tecnologias, de novas linguagens, auxiliam no acesso ao ensino de sistemas embarcados e computação como um todo.

Ao adicionar novos recursos à plataforma, estamos liberando novas possibilidades para professores ensinarem, para alunos aprenderem, para criadores inovarem. Estamos possibilitando que o conhecimento atinja lugares onde antes havia limitações por questões de estrutura ou financeiras, estamos dando viabilidade para projetos científicos decolarem.

Por ser um projeto *open source*, ele é aberto para qualquer pessoa sugerir melhorias, desde a adição de novas funcionalidades, como foi este trabalho, quanto o aprimoramento das funcionalidades já existentes. Baseado nos resultados obtidos neste trabalho, podemos identificar potenciais lacunas e possíveis correções a serem implantadas no futuro:

- Snek é uma biblioteca ainda muito pouco explorada no BIPES. Há uma grande diversidade de comandos. O suporte a sensores e atuadores pode ser útil na criação de projetos mais complexos (SNEK-LEGO, 2022).
    - A proposta é aumentar a gama de sensores e atuadores que possam ser suportados via Snek, como sensores de temperatura e proximidade, para que projetos de robótica possam ser desenvolvidos usando BIPES e Arduino.
- O acionamento do servo motor ainda depende de um bloco que retenha sua execução por um período de tempo, dando tempo para que o eixo chegue à posição do ângulo selecionado. O uso dessa forma de acionamento impede que um programa seja criado com um bloco do servo motor em conjunto de muitos blocos ao mesmo tempo, pois isso causa uma sobrecarga no envio de comandos para a placa que impede novos comandos de serem compreendidos, ou até mesmo carregados no console de comandos.
    - É necessário identificar o motivo do sistema sobrecarregar, e assim suportar completamente a execução do bloco, ou impor limitações na estrutura da plataforma para que falhas como essa não ocorram. Outra

forma de lidar com esse problema é rever a operação do servo motor e fazê-la sem a dependência de um *loop* que segure a execução. Para isso, é preciso modificar a linguagem Snek para suportar o controle dos *timers* do Arduino.

- O funcionamento do painel de IoT é excelente, entretanto, o painel HTML não consegue enviar comandos através da Internet, sendo limitado somente à comandos locais através da comunicação serial.
    - Tentar integrar a comunicação *online* sentido “Internet  para Console” a fim de permitir que o usuário não só monitore como também controle o ângulo do servo motor de qualquer lugar através de um navegador conectado à Internet.

# 6 CONCLUSÃO

Este trabalho contribuiu para a expansão da capacidade da plataforma BIPES através da adição de um novo bloco de operação para acionamento de um servo motor, aumentando as possibilidades de uso e aprendizado de uma nova linguagem de programação, a linguagem Snek, e fortalecendo o suporte de um dispositivo que está muito presente no universo de sistemas embarcados, o Arduino. Esse novo bloco foi atestado ser de fácil usabilidade e não estar condicionado ao usuário ter conhecimento prévio de programação.

Também foi apresentada uma forma de integrar o uso da linguagem Snek ao monitoramento de sistemas embarcados através do conceito de Internet das Coisas, onde o ângulo do servo motor foi avaliado através da internet de duas formas possíveis: uma usando o EasyMQTT e outra usando painéis do Freeboard, ambos já presente na plataforma. Se tratando dos painéis Freeboard, um foi criado com o intuito de se controlar o ângulo do servo motor, e não só monitorá-lo através da internet.

A adição de novos recursos no BIPES é fundamental para seu crescimento e difusão da programação em blocos voltada para o desenvolvimento de aplicações em sistemas embarcados. A facilidade de uso desses recursos auxilia o usuário a gastar mais tempo pensando na lógica do seu programa do que em como utilizar a plataforma.

# REFERÊNCIAS


ARDUINO. **About Arduino**. Disponível em: <https://www.arduino.cc/en/about>. Acesso em: 27 mar 2022.

ARDUINO UNO, **Arduino Uno & Genuino Uno**. Disponível em: <https://www.arduino.cc/en/main/arduinoBoardUno>. Acesso em: 30 mar 2022.

AROCA, R.V. et al. "**Uma introdução à Internet das Coisas e Sistemas Embarcados utilizando programação por blocos com BIPES e ESP8266 / ESP32**". 1a Edição. São Carlos. Editora dos Autores. 2021. 99 p. Disponível em: <https://bipes.net.br/wp/book-livro/>. Acesso em: 29 mar 2022.

ATHAWALE, Uttara, et al. "Review of Google Blockly and Its Innovative Use." **International Journal of Computer Sciences and Engineering**, v. 6, n. 6, p. 1262–1266, 2018.

BIPES. **Tech Details**. Disponível em: <http://bipes.net.br/docs/get-started/tech-details.html>. Acesso em: 21 mar. 2022

BLOCKLY. **Blockly**. Disponível em: <https://developers.google.com/blockly>. Acesso em: 23 mar 2022

COSTA, R.G., PIEDADE, J.M.N. 2021. "Uso do aplicativo MIT app inventor na aprendizagem de programação: uma revisão sistemática da literatura entre 2011 e 2020". **Revista Intersaberes**. v. 16, n. 37, p. 160-177, abr 2021.

DA SILVA, T.S.C, et al. "Um Modelo Para Promover o Engajamento Estudantil No Aprendizado De Programação Utilizando Gamification." **Revista Brasileira De Informática Na Educação**, v. 26, n. 3, p. 120, 2018.

DEVINE, J, et al. "MakeCode and CODAL: Intuitive and Efficient Embedded Systems Programming for Education." **Journal of Systems Architecture**, v. 98, p. 468–483, 2019.

EL-ABD, M. "A Review of Embedded Systems Education in the Arduino Age: Lessons Learned and Future Directions." **International Journal of Engineering Pedagogy**, v. 7, n. 2, p. 79–93, 2017.

FREEBOARD. **freeboard – Dashboard for the Internet of Things**. Disponível em: <https://freeboard.io/>. Acesso em: 02 abr 2022.

GNU. **GNU Grub**. Disponível em: <https://www.gnu.org/software/grub/index.html>. Acesso em: 24 mar 2022.

JUNIOR, A. G. D. S. et al. "BIPES: Block Based Integrated Platform for Embedded Systems,". **IEEE Access**, v. 8, p. 197955-197968, 2020.

LÓPEZ, J.M.S, et al. “Introducing Robotics and Block Programming in Elementary Education.” **Revista Iberoamericana De Educación a Distancia**, v. 24, n. 1, p. 95–113, 2021.

MICROPYTHON. **MicroPython code statistics**. Disponível em: <https://micropython.org/resources/code-dashboard/>. Acesso em: 21 mar. 2022

MIT APP INVENTOR. **About Us**. Disponível em: <https://appinventor.mit.edu/about-us>. Acesso em: 28 mar 2022.

OPTIBOOT. **Optiboot Bootloader for Arduino and Atmel AVR**. Disponível em: <https://github.com/Optiboot/optiboot>. Acesso em: 24 mar 2022.

RAABE, A.L.A, et al. Referenciais de Formação em Computação: Educação Básica. **Sociedade Brasileira de Computação**. Disponível em: <https://www.sbc.org.br/noticias/10-slideshow-noticias/1996-referenciais-de-formacao-em-computacao-educacao-basica>. Acesso em: 27 mar 2022.

SILVA, C.A. **Desenvolvimento e validação de módulo de comunicação MQTT para plataforma BIPES para aplicações de Internet das Coisas**. 2020. 89f. Monografia (Trabalho de Graduação em Engenharia de Computação) - Universidade Federal de São Carlos, 2020

SNEKLANG. **Snek: A Python-inspired Language for Embedded Devices**. Disponível em: <https://sneklang.org/>. Acesso em: 21 mar. 2022

SNEK-LEGO. **Snek and Lego: Building Robots!**. Disponível em: <https://sneklang.org/snek-lego/>. Acesso em: 02 abr 2022.

SNEK UPLOADER. **Snek Lang firmware uploader**. Disponível em: <https://bipes.net.br/snek-web-uploader/>. Acesso em 02 abr 2022.

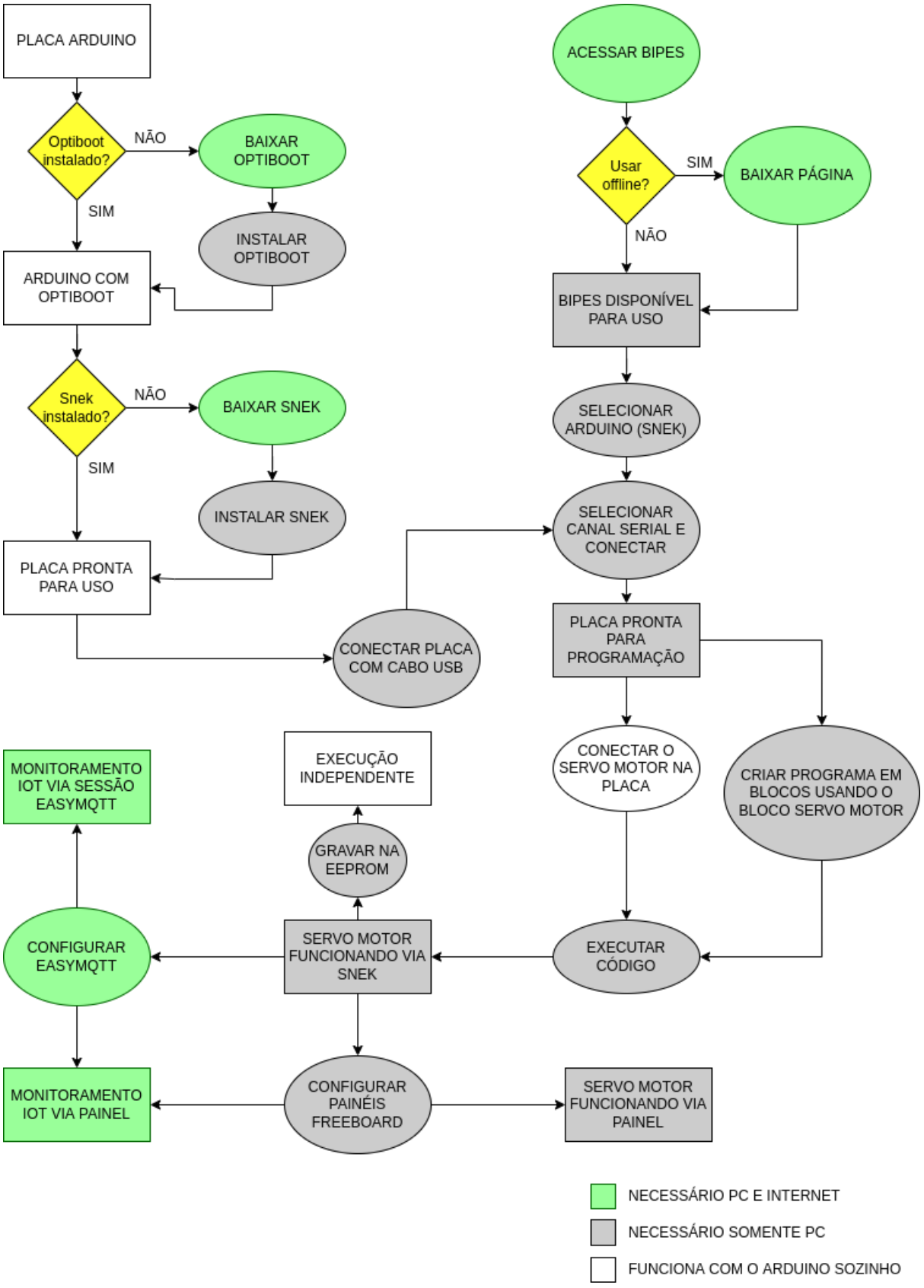
SOCIETY OF ROBOTS. **Actuators - Servos**. Disponível em: <https://www.societyofrobots.com/actuators_servos.shtml>. Acesso em: 23 mar 2022.

SOHN, W.S. “Design and Evaluation of Computer Programming Education Strategy using Arduino”. **Advanced Science and Technology Letters**. v. 66, pp.73-77, 2014.

TORRES, V., AROCA, R., & BURLAMAQUI, A.. “Ambiente de Programação Baseado na Web para Robótica Educacional de Baixo Custo”. **HOLOS**, v. 5, n. 30, p. 261-268, 2014.

WEBSOCKET. **The WebSocket API**. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API>. Acesso em 02 abr 2022.

# APÊNDICE A – Fluxograma de Uso do Projeto



## APÊNDICE B - Código do Painel HTML

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">

// Configuração visual do controlador deslizante
<style>
.slidecontainer {
  width: 100%;
  margin-left: 5px;
}

.slider {
  -webkit-appearance: none;
  width: 90%;
  height: 25px;
  background: #d3d3d3;
  outline: none;
  opacity: 0.7;
  -webkit-transition: .2s;
  transition: opacity .2s;
}

.slider:hover {
  opacity: 1;
}

.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 25px;
  height: 25px;
  background: #04AA6D;
  cursor: pointer;
}

.slider::-moz-range-thumb {
  width: 25px;
  height: 25px;
  background: #04AA6D;
  cursor: pointer;
}
</style>
</head>
```

```
<body>

<h2>Servo Angle Control</h2>

<div class="slidecontainer">

// Controlador deslizante
  <p>Angle:</p>
  <input type="range" min="0" max="180" value="90" class="slider"
id="myRange">
  <p><span id="angle"></span>°</p>

// Seletor de pino do servo motor
  <label for="pin">Pin:</label>
  <select id="pin" name="pin">
    <option value="D0">D0</option>
    <option value="D1">D1</option>
    <option value="D2">D2</option>
    <option value="D3">D3</option>
    <option value="D4">D4</option>
    <option value="D5">D5</option>
    <option value="D6">D6</option>
    <option value="D7">D7</option>
    <option value="D8">D8</option>
    <option value="D9">D9</option>
    <option value="D10">D10</option>
    <option value="D11">D11</option>
    <option value="D12">D12</option>
    <option value="D13">D13</option>
  </select>

  <button onclick="set_command()">SET</button>
</div>

<script>
  var slider = document.getElementById("myRange");
  var angle_value = document.getElementById("angle");
  var selected_pin = document.getElementById("pin");
  angle_value.innerHTML = slider.value;

// Configurando a ativação do servo motor
  function set_command() {
    var value_pin = selected_pin.value;
    var t_wait = "(" + slider.value + "/180)*0.002 + 0.0005";
          var cmd = 'talkto(' + value_pin + ')\\n';
          cmd += 'off()\\n';
          cmd += 'setpower(1)\\n';
          cmd += 'for turning in range(30) :\\n';
          cmd += '  on()\\n';
```

```
        cmd += '  time.sleep(' + t_wait + ')\\n';
        cmd += '  off()\\n';
        cmd += '\\n';

    // Envia o comando do painel para o console
    parent.bufferPush(cmd);
  }
  slider.oninput = function() {
    angle_value.innerHTML = this.value;
  }
</script>
</body>
</html>
```