

PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA E TECNOLÓGICA
UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO
CENTRO CCT
LABORATÓRIO LCMAT

Relatório do período: 05/2020 a 04/2021

RELATÓRIO ANUAL

Nome do Bolsista: João Vítor Fernandes Dias

Curso: Ciência da Computação

Nº Matrícula: 00119110377

Orientador: Fermín Alfredo Tang Montané

Título do Projeto: Estudo sobre o controle remoto de dispositivos microcontrolados utilizando dispositivos móveis

Título do Plano de Trabalho: Estudo sobre a Integração de Plataformas Microcontroladas para Internet das Coisas

Fonte financiadora da Bolsa: PIBIC / CNPq

1 Etapas propostas no plano de trabalho

O plano de trabalho contempla as seguintes propostas:

- a) Estudo sobre diferentes tecnologias de comunicação remota para controle de dispositivos microcontrolados, entre elas: Bluetooth, Radiofrequência, WiFi, etc. Pesquisa sobre módulos eletrônicos disponíveis no mercado e documentação do estudo.
- b) Estudo sobre alternativas de desenvolvimento de aplicativos Android visando o desenvolvimento de interfaces de controle e monitoramento para dispositivos microcontrolados. Documentação do estudo.
- c) Estudo, desenvolvimento e implementação do segundo protótipo de braço robótico. Escolha da plataforma microcontrolada. Documentação.
- d) Desenvolvimento da interface de controle e monitoramento do braço robótico. Documentação.
- e) Realização de testes de avaliação e desempenho da interface e do braço robótico.
- f) Elaboração de relatório técnico.

Quanto ao item *a* foi decidido utilizar as tecnologias de Bluetooth e wi-fi visto que são as únicas amplamente utilizadas nos Smartphones. No item *b* foram vistas diferentes linguagens de programação apropriadas para o desenvolvimento de apps, e dentre elas foi escolhida a linguagem React Native devido à possibilidade de rápida checagem de erros no código e também poder utilizar o aplicativo (app) para dispositivos iOS futuramente. O item *c* foi parcialmente concluído. As plataformas microcontroladas escolhidas para teste foram NodeMCU Amica/Lolin e se mostraram viáveis. Porém após o teste com essas placas, o módulo Bluetooth acabou gerando resultados imprevistos que ainda não foram solucionados, suspeita-se de uma falha no hardware e foi encomendado um novo módulo de reposição. A interface desenvolvida para o item *d* se mostrou eficiente, porém apresenta alguns bugs que deverão ser resolvidos no futuro. Os testes realizados para o item *e* mostram que, apesar de funcional, diversos pequenos detalhes podem ser aprimorados para resultar em uma melhor manipulação do braço mecânico, principalmente em relação ao app.

2 Introdução

A Internet das Coisas (Internet of Things – IoT) tem como essência a interligação de diferentes tecnologias de rede agregadas a objetos, coisas, como smartphones, sensores pessoais, braços robóticos na automação, dentre outros. Essa interligação se torna uma única malha de dispositivos conectados. Mesmo que inicialmente tivessem propósitos únicos, ao se interligar passam a ter uma nova gama mais extensa de funcionalidades que se complementam [Ste18], [San14].

Como citado acima, o uso de braços robóticos está presente na realidade do IOT na escala industrial relacionado a automação, porém não está limitado a ela. Seu uso se expande a diversas funções [aKa18], tais como usar um braço mecânico leve e de alta velocidade para jogar badminton, utilizar um braço mecânico para a escrita [Lee20] ou até mesmo utilizado nas sondas espaciais Perseverance, Spirit, Opportunity e Curiosity para carregar câmeras [Dun171], [Dun17].

O controle desses braços robóticos pode ser feito de diversas formas: controle automático e repetitivo como em uma linha de produção ou com ações baseadas em respostas a sensores, ou também controlados remotamente por uma interface. Com o aumento da popularidade de smartphones e seus aplicativos [Bjø19], a capacidade de controlar dispositivos à distância através dos dispositivos mobile se torna mais almejado. Para o desenvolvimento de uma interface que permita essa função, busca-se um bom aproveitamento de código para as diversas plataformas disponíveis atualmente, com destaque para os dispositivos iOS e Android, sem que suas funcionalidades sejam consideravelmente comprometidas, essa capacidade se vê presente na linguagem React Native o que o torna uma alternativa viável [Jaw18].

3 Objetivos

Este projeto tem como objetivo o estudo de tecnologias de comunicação remota, assim como o desenvolvimento de um aplicativo para smartphones Android com interface de controle para um novo tipo de braço robótico, com cinco graus de liberdade e motores de maior potência. Dessa forma, procura-se tornar o projeto do novo bolsista independente do que foi desenvolvido pela bolsista anterior. Procura-se acrescentar novos conhecimentos à experiência anterior.

4 Metodologia

Para o desenvolvimento do plano de trabalho foi seguida a seguinte metodologia:

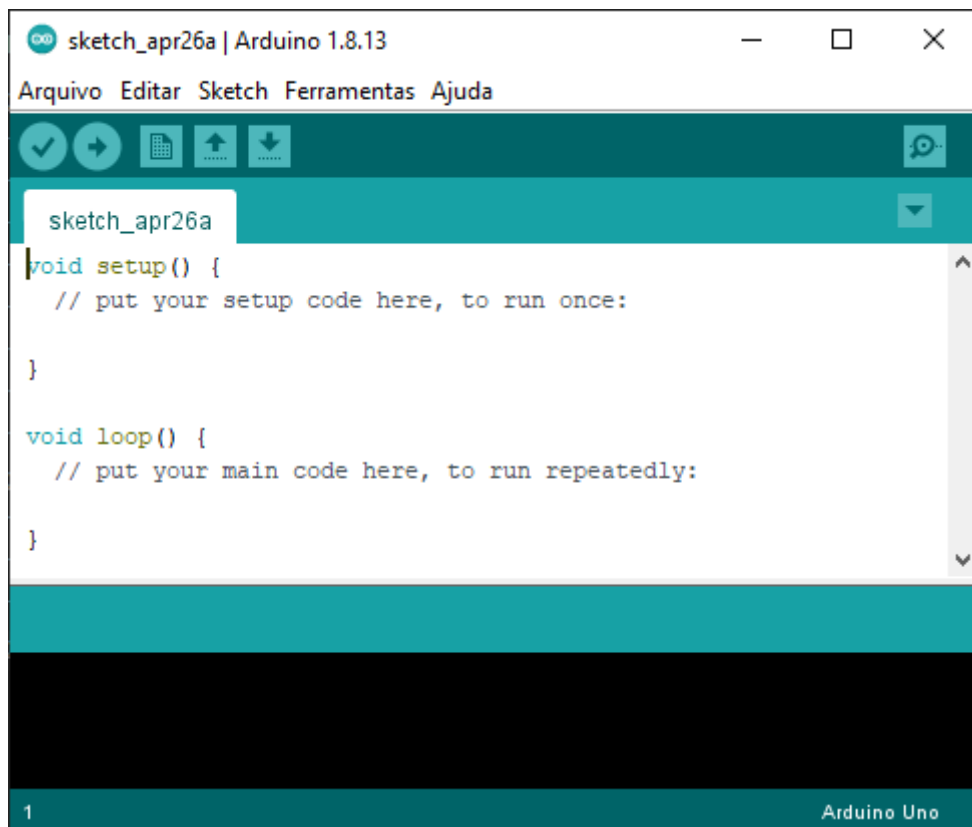
4.1 Pesquisas e minicursos

Pesquisa de blogs explicativos sobre módulos de comunicação remota [Koy21] e servomotores [Wil21], palestras sobre Bluetooth [Lee18]. Pesquisa de diversos artigos científicos na plataforma Periódicos Capes [Cap21] Participação de minicursos gratuitos de React Native ministrados pelo canal Developer Plus [Plu18], [Plu19] e a instituição educacional RocketSeat [Sea21] durante a Next Level Week #5. Participação de minicursos online voltados ao aprendizado da pesquisa científica em portais de periódicos ministrado por funcionários relacionados à plataforma Periódicos Capes [Tre21].

4.2 Programação no Arduino IDE

Desenvolvimento no ambiente Arduino IDE, que é um ambiente de desenvolvimento integrado (em inglês: *Integrated Development Environment* – IDE) que disponibiliza um editor de texto para a escrita do código, um console de texto, barra de ferramentas com diversas funções e permite a conexão com o Arduino UNO para o envio do programa desenvolvido e se comunicar com o dispositivo [Ard21]. É o software comumente utilizado para se desenvolver projetos utilizando microcontroladores Arduino, porém também pode ser utilizado para o desenvolvimento em outros microcontroladores. A Figura 1, ilustra a interface do Arduino IDE.

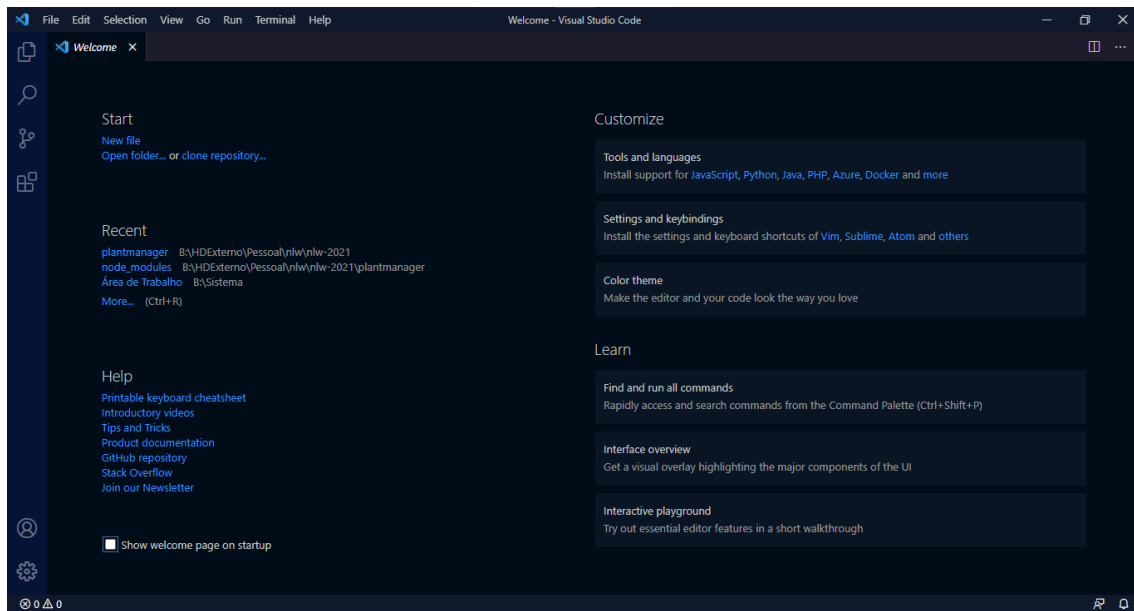
Figura 1: Tela inicial do Arduino IDE 1.8.13



4.3 Programação de aplicativos no Visual Studio Code

Desenvolvimento no ambiente Visual Studio Code (ou também "VS Code"), que é um editor de código simplificado com suporte para operações de desenvolvimento como debug e controle de versões [FAQ21]. Possui também a característica IntelliSense que permite o complemento automático do código enquanto está sendo escrito o que reduz a chance de erros provenientes da datilografia (geralmente chamado de "typo" na área de programação) e também torna mais eficiente e rápido o desenvolvimento do código [Vis21]. Esse foi o ambiente utilizado para o desenvolvimento de aplicativos mobile e que com o uso de extensões também permite a substituição do Arduino IDE, se mostrando uma ferramenta ainda mais completa. Este ambiente é ilustrado na Figura 2.

Figura 2: Visual Studio Code 1.55.2



4.4 Montagem do braço robótico

A montagem da estrutura e componentes do novo braço robótico é uma tarefa manual, que exige atenção aos detalhes, e o uso de ferramentas. Foram utilizados uma chave Philips e o manual disponibilizado pelo fornecedor das peças do braço robótico, para montar o novo braço robótico com 5 graus de liberdade, sendo este novo braço feito em acrílico, diferentemente da primeira versão feita em material MDF.

4.5 Junção dos códigos com os dispositivos

Após o reaproveitamento dos códigos desenvolvidos anteriormente no Arduino IDE, e o aprendizado sobre programação de aplicativos seguido de seu desenvolvimento, ambos os códigos foram integrados e permitiram a transmissão de dados entre um dispositivo Android e um modulo Arduino através de conexão Bluetooth. Posteriormente com a montagem do braço robótico, a transmissão de dados, já funcional, foi utilizada para movimentar o braço robótico.

5 Resultados e Discussão

Como resultado vê-se de forma sucinta a montagem do braço robótico e do aprendizado sobre tecnologias de comunicação remota adquirido para o desenvolvimento do app.

5.1 Seleção das tecnologias de comunicação remota

Nesta pesquisa foram encontradas diversas tecnologias de comunicação remota, muitas delas utilizadas no contexto do IOT. Apresentam-se as descrições dessas tecnologias junto a um parecer quanto a sua adequação ao atual projeto.

RFid (*Radio-Frequency Identification* – Identificação por Radiofrequência) – Não apropriado: Ações limitadas ao que for programado em cartões de RFid, não apropriado devido a necessidade da alta taxa de repetição e controle preciso dos servomotores.

Protocolo CAN – Viável, porém não apropriado: Permite envio de até 11 bits ou 29 bits e requer um dispositivo de envio e outro de recebimento, isso o torna viável, ou seja, seria possível desenvolver o controle do braço robótico utilizando essa tecnologia, porém, ao inserir módulos MCP2515 [Koy181] como intermediários aumentaria desnecessariamente a infraestrutura em comparação com as outras tecnologias escolhidas, além disso, não pode ser controlado diretamente por um smartphone que é um dos objetivos da pesquisa, o que faz com que não seja apropriado.

ESP32 LoRaWan – Viável, porém não apropriado: Depende de outro LoraWan [Koy18], e o foco da pesquisa é o controle do braço robótico através do Smartphone, o que faz com que entre no mesmo critério eliminatório que o do Protocolo CAN.

ZigBee – Viável, porém não apropriado: O ZigBee é um protocolo de comunicação feito para se adequar aos requisitos de dispositivos embarcados. Provê baixo consumo de energia e suporta grande número de dispositivos através de longas distâncias com várias topologias diferentes [Ban19]. Dependendo do avanço desse projeto, pode vir a ser apropriado o uso, desse protocolo, porém atualmente não se mostra adequado às necessidades.

6LoWPAN – Apropriado (mas não escolhido): Segundo [Nik19] “o ‘IPv6 em rede sem fio pessoal de baixa energia’ (IPv6 over *Low Power Wireless Personal Area Network* - 6LoWPAN) é uma adaptação da subcamada do IPv6 e provê conectividade por IP em redes de baixa energia e com perda de dados; o IETF padronizou essa subcamada. Hoje, 6LoWPAN é a tecnologia chave para vários modelos de rede no IoT assim como automação residencial, controle de sistemas industriais e cidades inteligentes.” Mesmo havendo a possibilidade do uso do módulo L-Tek 6LoWPAN Arduino Shield 900MHz ou do AxAvior 6LoWPAN Module, por enquanto optou-se por não utilizar essa tecnologia por causa da necessidade da obtenção dos módulos. Porém pode ser alvo de maiores pesquisas posteriormente, assim como o BLE (*Bluetooth Low Energy*).

Wi-fi (*Wireless Fidelity*) – Apropriado: Também chamado de WLAN (*Wireless Local Area Network* – Rede Sem Fio de Área Local) é uma tecnologia presente na maioria dos Smartphones atuais, geralmente utilizado para se conectar à internet. Apresenta diversos módulos para o uso com o Arduino. Estão disponíveis vários microcontroladores, como o ESP32 e o NodeMCU ESP8266, que já vêm com o módulo wi-fi embutido. Como o NodeMCU ESP8266 está disponível para uso sem a necessidade de aquisição, ele foi escolhido para ser testado.

Bluetooth (Classic) – Apropriado: O Bluetooth, também chamado de WPAN (Wireless Personal Area Network – Rede Sem Fio de Área Pessoal), assim como o wi-fi, também é uma tecnologia presente na maioria dos Smartphones atuais, geralmente utilizado para se conectar dispositivos sem fio, principalmente fones de ouvido, teclados e mouses. Apresenta diversos módulos para o uso com o Arduino, podendo ser citados os módulos HC-05 (utilizado nesse projeto) e o HC-06. Essa tecnologia também está disponível no módulo ESP32 citado anteriormente que embora não disponível, pode ser adquirido para projetos futuros.

Conclusão: As tecnologias apropriadas e escolhidas para o projeto atualmente são wi-fi e Bluetooth. Foram testados os microcontroladores NodeMCU Amica com ESP8266 e NodeMCU Lolin com ESP8266 para a conexão wi-fi e foi utilizado o módulo HC-05 para a conexão Bluetooth com o Arduino e também com ambos os microcontroladores Amica e Lolin.

5.2 Seleção dos microcontroladores

Quanto às plataformas microcontroladas foram pesquisadas diversas opções como:

Placas que têm o microchip ESP8266: ESP01, ESP12E, ESP12F, ESP201, NodeMCU-ESP12 e CP2102 e placas que têm o chip ESP32: ESP32-WROVER, ESP32-WROOM-32U, ESP32S-CP2102 e NodeMCU V3 WIFI 802.11.

De início optou-se por manter o desenvolvimento com o Arduino, visto que a quantidade de falhas poderia ser reduzida devido à familiaridade, e por isso o desenvolvimento pode ser mais dinâmico. Assim que o desenvolvimento fosse concluído, novos testes poderiam ser realizados com as outras placas para analisar se vale ou não a pena a mudança. As placas disponíveis para testes no laboratório e que foram definidas como viáveis para o projeto através de pesquisa temos o NodeMCU Amica e NodeMCU Lolin ambos com o microchip ESP8266 com acesso à conexão wi-fi.

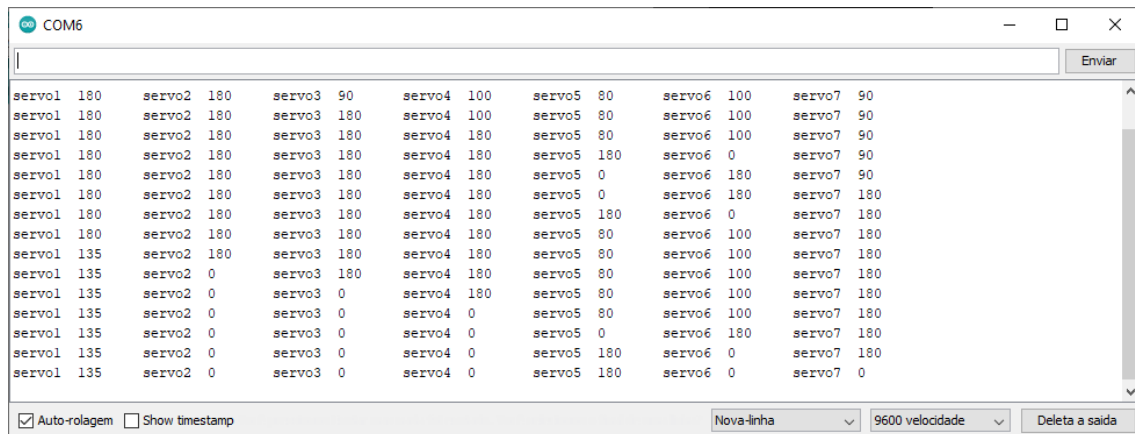
5.3 Códigos no Arduino IDE

Com o objetivo final de controlar o braço robótico através de tecnologias de comunicação remota, foram desenvolvidos programas no Arduino IDE para se alcançar este resultado. Nas subseções seguintes apresentam-se as ferramentas utilizadas assim como as etapas realizadas.

5.3.1 Monitor e Plotter Serial

A forma utilizada para a análise dos valores recebidos pontualmente é o uso do monitor Serial. A figura 3 ilustra o uso do monitor serial para mostrar o valor dos ângulos de cada um dos servomotores separadamente.

Figura 3: Demonstração do Monitor Serial



De acordo com as seguintes funções:

Código 1: Função genérica para imprimir o valor de qualquer servo

```
void printServo (int pos, char servo){  
    Serial.print("servo");  
    Serial.print(servo);    tab();  
    Serial.print(pos);      tab();  
}
```

O código 1 mostra a função genérica *printServo ()* que imprime a informação de angulação e identificação do servo. Para isso, imprime o texto “servo”, seguido de um caractere referente a qual servo terá sua angulação mostrada, logo após há uma função *tab ()* que apenas imprime uma tabulação para deixar devidamente organizado o texto na saída, e enfim imprime um valor inteiro correspondente à posição do servo (variável “pos”) com mais uma tabulação.

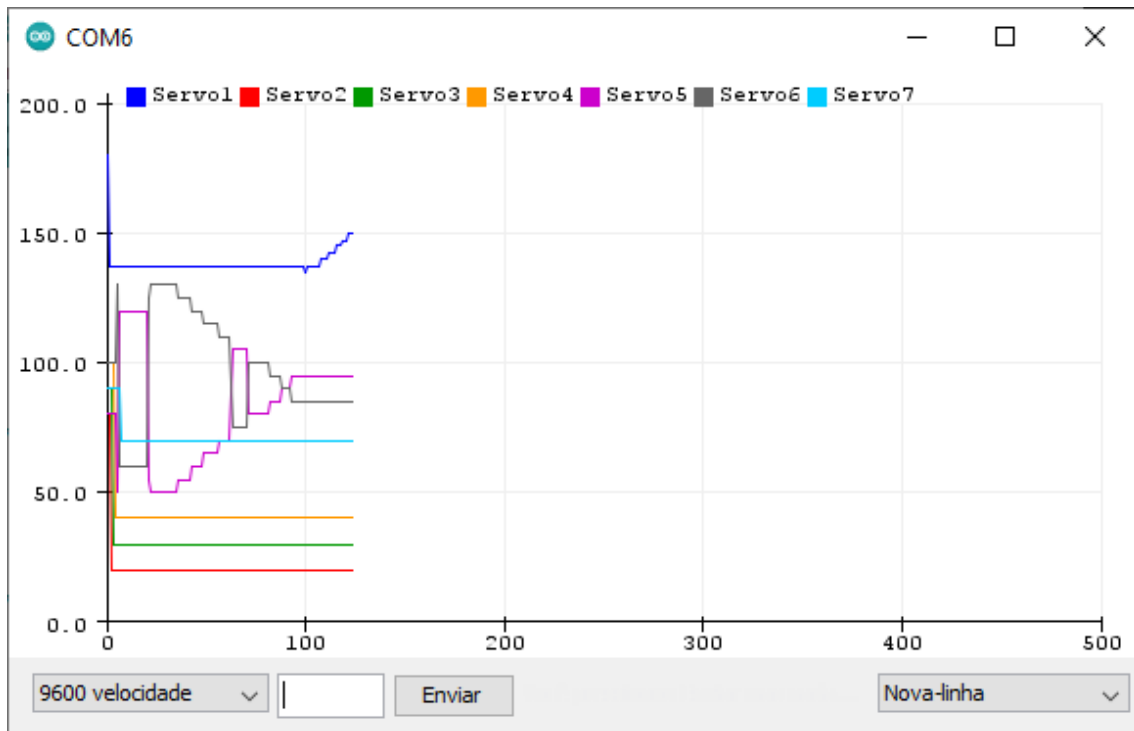
Código 2: Função que chama a função “printServo ()” para cada um dos servos

```
void servoMonitor () {  
    printServo (ang1,'1');  
    printServo (ang2,'2');  
    printServo (ang3,'3');  
    printServo (ang4,'4');  
    printServo (ang5,'5');  
    printServo (ang6,'6');  
    printServo (ang7,'7');  
    line ();  
}
```

O código 2 apresenta a função *servoMonitor ()* que imprime as informações referentes a todos os servos. Esta função utiliza a função *printServo ()*, descrita anteriormente, que recebe a angulação e o número do servo correspondente para impressão e termina com a função *line ()* que encerra a linha.

Como alternativa do Monitor Serial, podemos ver em forma de gráficos a variação de angulação dos servomotores através do plotter Serial (ver Figura4).

Figura 4: Demonstração do Plotter Serial



Este gráfico foi elaborado utilizando a função `servoPlotter()` mostrada no código 3.

Código 3: Função que imprime cada um dos ângulos dos servos para o Plotter Serial

```
Void servoPlotter(){
    Serial.print("Servo1:");
    Serial.print(ang1);
    Serial.print(",Servo2:");
    Serial.print(ang2);
    Serial.print(",Servo3:");
    Serial.print(ang3);
    Serial.print(",Servo4:");
    Serial.print(ang4);
    Serial.print(",Servo5:");
    Serial.print(ang5);
    Serial.print(",Servo6:");
    Serial.print(ang6);
    Serial.print(",Servo7:");
    Serial.println(ang7);
}
```

Cada variável “ang” representa a angulação de um servomotor. Assim, ao imprimir esses valores no plotter serial, o plotter interpreta cada um e os imprime com cores diferentes conforme ilustrado na legenda.

5.3.2 Recebimento de dados via *Bluetooth*

O recebimento de dados via *Bluetooth* é bem simples. Como exemplificado pelo código a seguir (Código 4):

```
void setup () {  
    Serial.begin(9600);  
}  
  
void loop () {  
    if (Serial.available()) {  
        Serial.println(Serial.read());  
    }  
}
```

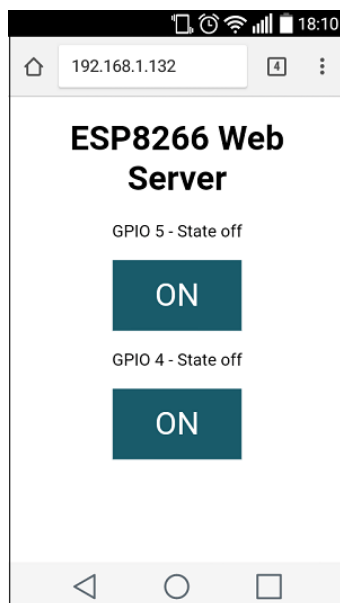
Na função *setup ()* ele apenas inicia a comunicação serial na velocidade 9600 bauds, ou seja, 9600 bits por segundo. Na função *loop ()* se o valor estiver disponível na entrada serial, ele irá fazer a leitura e imprimir o valor lido.

5.3.3 Recebimento de dados via wi-fi

O recebimento de dados via wi-fi apresenta uma complexidade muito superior à do envio de dados via Bluetooth. Foram realizados testes com o módulo NodeMCU Amica que possui wi-fi integrado. Nesses testes pode-se observar demora superior quando comparado ao Arduino, na configuração inicial do microcontrolador, assim como para fazer o upload dos códigos para o microcontrolador. O código utilizado no teste é uma junção de várias linguagens de programação, o que pode gerar certa confusão ao se misturar comandos em C com as tags do HTML. Mas apesar dessas dificuldades foi possível realizar a transmissão dos dados.

Com o código base obtido de [San18] e [San181] que transmite os dados ao pressionar dois botões, modificou-se o código para que ao invés de transmitir os dados a partir dos botões, fosse transmitido após a mudança de posição de um *slider* (Controle deslizante). Entretanto, a informação só é enviada após soltar o *slider* na nova posição, o que reduz a quantidade de dados que é transmitida.

Figura 5: ESP8266 Web Server - Imagem demonstrativa disponível em [San18]



Mesmo com esses pontos negativos, é esperado um estudo mais aprofundado em relação à aplicação wi-fi e que as características ruins sejam amenizadas.

5.4 Seleção da linguagem de programação para desenvolvimento de apps

Diante da vasta gama de linguagens de programação disponíveis foi feita uma pesquisa sobre algumas linguagens de programação e os seus benefícios, sendo classificadas quanto à sua aplicabilidade no atual projeto [Sch19].

Swift – Não apropriado: É uma linguagem específica para aplicativos nativos do Sistema Operacional da Apple, o Apple OS (iOS). O que impede que sejam desenvolvidos apps para Smartphones Android que são o foco do projeto.

Java – Viável e apropriado: É uma linguagem específica para aplicativos nativos do Android, o que permite o uso amplo dos recursos disponíveis nos Smartphones, porém, acaba não sendo prático caso seja visado no futuro o desenvolvimento de um app que aceite também os Smartphones da Apple. Por esse critério foi deixado de lado, porém não se descarta a possibilidade de seu uso em algum projeto futuro.

Kotlin – Viável e apropriado: Possui as mesmas desvantagens do Java.

Ionic e Flutter – Viáveis e apropriados: São opções consideravelmente similares ao React Native em sua funcionalidade. E por isso também poderiam ser utilizadas nesse projeto.

JavaScript – Viável e apropriado: É a linguagem utilizada no *back-end* do app. É a linguagem que em conjunto com o framework React Native, torna o app funcional.

React Native – Apropriado: Permite que um app utilizando da linguagem JavaScript seja convertido para a linguagem nativa tanto do iOS quanto do Android. Também podendo ser considerado que a lógica presente no desenvolvimento se assemelha (e utiliza de conceitos) de linguagens mais consolidadas, sendo elas HTML e CSS (mais utilizadas em desenvolvimento web) e também de React (uma biblioteca de JavaScript para desenvolvimento de interfaces) e JavaScript (linguagem *back-end*). Além disso pode ser desenvolvido utilizando o Expo. Que torna possível programar utilizando React Native e quase que instantaneamente ver o app resultante em um Smartphone conectado à mesma rede wi-fi que tenha o app do Expo. Mesmo com a velocidade o Expo tem suas limitações: ao utilizá-lo não é possível importar bibliotecas (conjunto de código que permitem o acesso a novas funcionalidades) foram necessárias para o projeto. Também tem seu uso consolidado em diversos apps conhecidos como Facebook, Instagram, Discord, dentre outros [Nat21].

Com isso foi decidido o uso do React Native em conjunto com o JavaScript para o desenvolvimento do App do atual projeto.

5.5 Aplicativos

Para entender melhor o uso e a sintaxe da linguagem escolhida, foram desenvolvidos diversos apps.

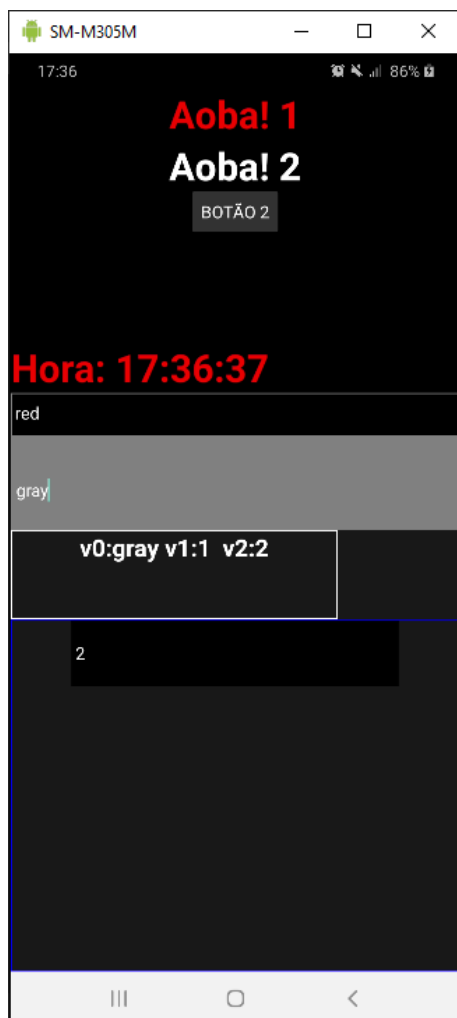
5.5.1 Familiarização com React Native

Para poder desenvolver o app, primeiramente foi necessário aprender sobre a linguagem que seria utilizada. Primeiro aprendendo a montar o ambiente de trabalho com os softwares necessários, sendo eles VS Code para programar o app, Android Studio para poder

emular um dispositivo Android em que o desenvolvimento do app seria observado e também o Node.js que disponibiliza algumas ferramentas de desenvolvedor e também o Expo que permite uma rápida atualização através de seu app.

Durante o aprendizado utilizando o Expo, foram desenvolvidos diversos apps com funcionalidades diferentes para se explorar as capacidades da linguagem e entender o seu funcionamento. Foram testados componentes como Bloco de Texto, Imagem, Relógio, Caixa de Texto, e o View (que é um encapsulador de componentes que pode ter seu estilo alterado para poder modelar graficamente o aplicativo).

Figura 6: Primeiro app desenvolvido misturando diversas funcionalidades



Um primeiro app (ver Figura 6) foi desenvolvido apenas com propósito de familiarização com estruturas visuais e aprendizado da linguagem. Das diversas funcionalidades testadas, algumas serão citadas: o botão permitia mostrar um aviso em formato de pop-up; o bloco cinza claro permitia o input de texto e caso fosse o nome em inglês de uma cor válida, todo o bloco mudaria de cor. O bloco inferior, abaixo do bloco cinza claro, apresenta os valores das variáveis nos inputs de texto.

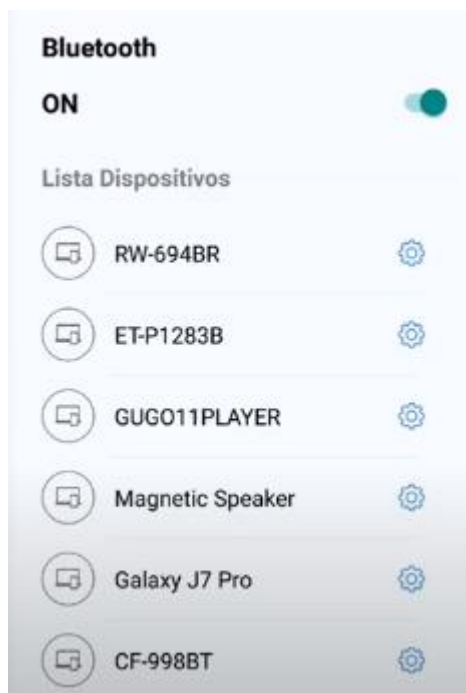
Após diversos testes com esses componentes e variáveis, percebeu-se uma limitação presente no Expo: quando a biblioteca para o uso do Bluetooth foi adicionada, o Expo não a reconheceu, com isso, foi necessário dispensar o Expo para que o desenvolvimento continuasse.

Utilizando o Node.js para poder atualizar o app visualmente, mesmo que a velocidade de atualização tenha reduzido, não havia mais limitação em relação ao uso da biblioteca necessária. E assim, utilizando apenas o módulo Bluetooth HC-05 conectado ao Arduino pode-se desenvolver o app para que ele enviasse os dados devidamente. Devido à forma como o Arduino recebe os dados, foi necessário primeiramente converter os dados que seriam enviados utilizando a base 64 que é uma forma de compactar os dados enviados em caracteres mais comuns a fim de reduzir a quantidade de bytes necessários para enviar texto através de dispositivos, assim reduzindo o risco de perda de dados por incompatibilidade de caracteres entre diferentes plataformas.

5.5.2 Conexão com dispositivos Bluetooth

Após desenvolver um app com conexão Bluetooth acompanhando o [Lar20], foi possível desenvolver um aplicativo próprio que conseguisse se conectar ao módulo Bluetooth HC-05 conectado ao Arduino, fato confirmado pela mudança no padrão luminoso emitido pelo módulo. Este aplicativo foi desenvolvido com base em um Workshop [Lar20] (Figura 7).

Figura 7: Aplicativo desenvolvido no WorkShop - Imagem demonstrativa disponível em [Lar201]

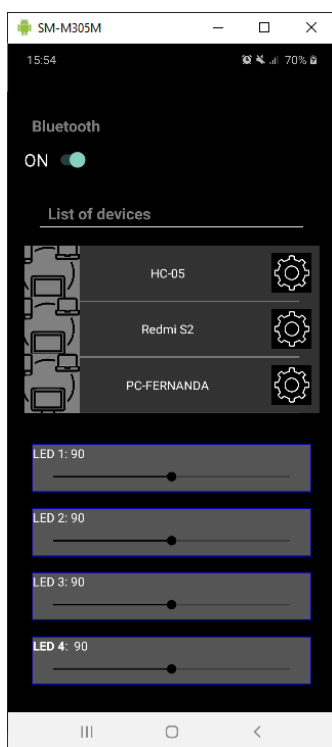


5.5.3 Envio de dados via Bluetooth com botões e sliders

Após pesquisas na documentação de bibliotecas de comunicação Bluetooth [Dav20], [Dav21], pôde-se entender e utilizar as funções necessárias para o envio de dados via Bluetooth.

Baseado no que foi aprendido com o app de treinamento, e na pesquisa das documentações, foi possível alterar algumas funções já existentes para poder enviar dos dados utilizando a biblioteca *react-native-base64* [era20] para converter os dados enviados para a base 64 já citada anteriormente.

Figura 8: Um dos primeiros layouts do app de envio de dados por slider



O app mostrado na Figura 8 permite ligar e desligar o Bluetooth no dispositivo, se conectar a um dispositivo disponível e enviar para ele os valores dos *sliders* quando eles eram movimentados. Com isso foi possível se conectar ao módulo HC-05 e enviar os dados apropriadamente.

Vale ressaltar que o aplicativo é capaz de cumprir com o objetivo do projeto da pesquisa, conseguindo se conectar através de uma tecnologia de comunicação remota à um dispositivo microcontrolado capaz de movimentar os servomotores do novo braço robótico.

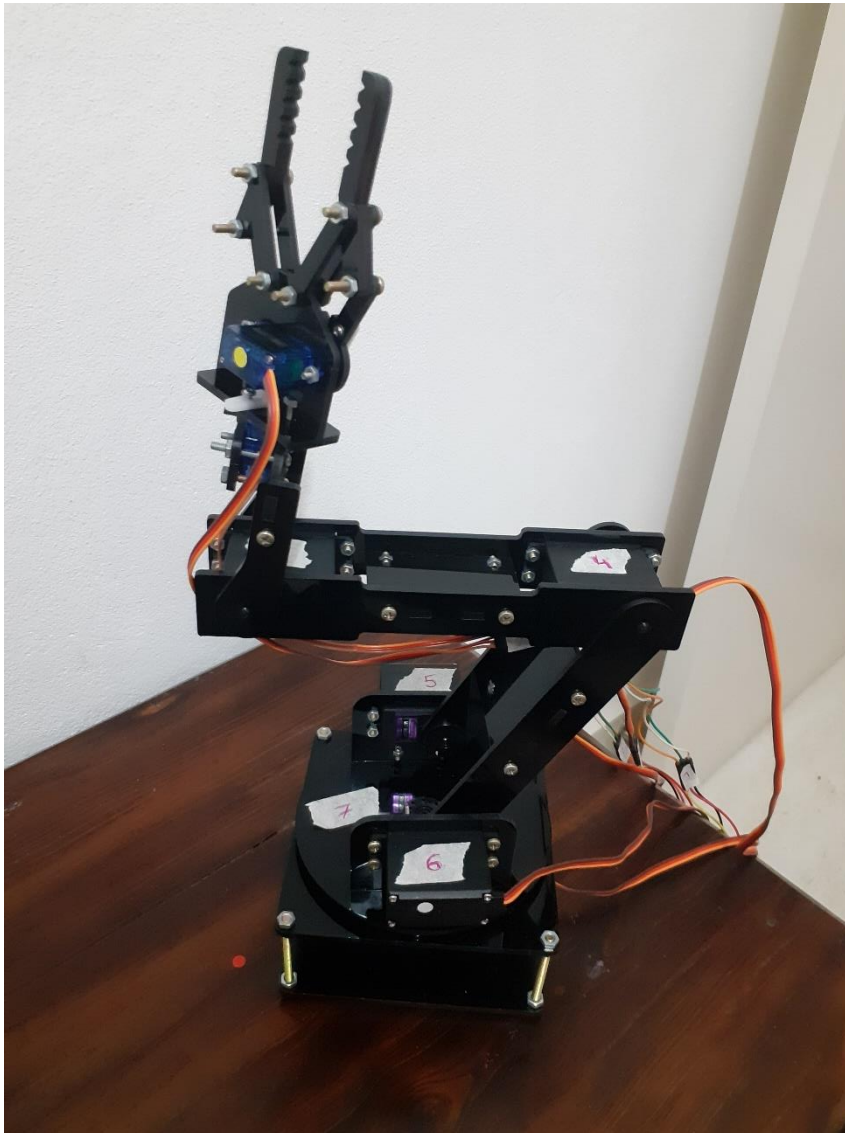
5.6 Braço robótico montado

A montagem do braço robótico foi bastante trabalhosa e durou em torno de 10 horas. De um modo geral não foi muito complicado, porém a falta de instruções claras presentes no manual reduziu o rendimento. Dessa forma, foi inevitável que acontecessem erros de montagem. Outros problemas foram a existência de peças que não encaixavam adequadamente e necessitaram de um pouco de força para se fixarem devidamente, e também alguns parafusos que eram mais longos do que necessário e outros que eram mais curtos do que o necessário.

Optou-se por um modelo de braço robótico com 5 graus de liberdade que tem 5 servomotores “*Servo Motor MG996R Tower Pro*” [ETC14] que são mais potentes e trazem maior resistência ao peso da estrutura do que os servomotores “*Micro Servo 9g SG90 TowerPro*” [Pro14], [Flo21], presentes no primeiro protótipo construído em pesquisa anterior.

A Figura 9 mostra o segundo protótipo devidamente montado.

Figura 9: Braço robótico montado



5.7 Movimentação do braço robótico com controle via Bluetooth

Após o desenvolvimento do app que enviava os valores dos *sliders* apropriadamente para o HC-05 e da conclusão da montagem do braço robótico, os diferentes segmentos do projeto foram conectados, resultando na movimentação esperada do braço robótico, necessitando apenas de alguns ajustes em relação à angulação máxima de cada servo motor. Após esses ajustes, o braço robótico deveria estar completamente funcional.

Após algumas tentativas percebeu-se que após determinada angulação, os servos não eram capazes de suportar o peso da estrutura e acabavam deixando a estrutura descer lentamente. Essa situação inesperada não se encontra resolvida e espera-se que uma solução seja encontrada na continuação da pesquisa.

5.8 Testes do Bluetooth com outros microcontroladores

Como citado anteriormente, com a conclusão do objetivo do projeto utilizando o Arduino UNO com o módulo Bluetooth HC-05 e o Sensor Shield v5.0 [Ele20]. Passou-se a testar os outros microcontroladores utilizando o mesmo app. Após alguns ajustes nas configurações

do Arduino IDE o HC-05 pode ser conectado ao NodeMCU Amica sem maiores problemas e os valores recebidos puderam ser lidos adequadamente.

Após alguns testes bem sucedidos os valores recebidos pelo HC-05 mudaram de padrão aparentemente sem motivo. Passaram a serem recebidos os seguintes valores:

192 5 0 0 250 192

Após variar os valores enviados e a quantidades de caracteres, percebeu-se que os valores 192 e 0 não se alteravam. Apenas o segundo e penúltimo valor se alteraram. Sendo o segundo referente a quantidade de caracteres e o penúltimo sendo o resultado do cálculo “255-quantidade de caracteres”.

Após novos testes com quantidades ainda maiores de caracteres, percebeu-se que quando a mensagem enviada era maior do que 255 caracteres, o primeiro 0 recebido se tornava 1. Se fosse maior que 2×255 (510) caracteres o primeiro 0 recebido se tornaria 2. Sendo assim, estima-se que caso a mensagem seja maior que 255×255 , o segundo 0 se torne, permitindo então uma mensagem de tamanho total $255 \times 255 \times 255$ (16.581.375) caracteres. Alguns valores também apresentaram comportamentos inesperados, e estão passíveis de maiores análises. De uma forma geral a quantidade de caracteres é enviada, logo o dado aparentemente ainda está sendo enviado, porém não está sendo lido ou interpretado corretamente.

Quanto a solução do bug, foram testados diversos códigos diferentes, em diversos microcontroladores, com diversos apps diferentes e diversos softwares de terminal. Em todos eles apresentaram o mesmo bug que antes não ocorria. Com isso cogita-se a possibilidade de falha no hardware do HC-05. Porém essa afirmação não é assertiva, visto que não houve nenhuma modificação externa aparente que possa ter causado essa mudança, mesmo assim, espera-se que posteriormente um outro módulo Bluetooth seja testado para comprovar a hipótese.

6 Conclusões

As etapas do plano de trabalho foram em sua maioria concluídas com êxito, restando apenas resolver o bug do módulo *Bluetooth* HC-05, a implementação da conexão via wi-fi, analisar e montar o circuito elétrico necessário para manter ativos os servomotores do braço robótico e aplicar as melhorias propostas ao app.

Este relatório apresenta informações sobre diferentes tecnologias de comunicação remota, restringindo o filtro para as que fossem aplicáveis à microcontroladores e à utilização pelo Smartphone. Sendo então escolhidas as tecnologias wi-fi e Bluetooth para que fossem testadas.

Houve também pesquisa sobre diferentes linguagens de programação que permitissem o desenvolvimento de aplicativos. Dentre várias opções, foi decidido utilizar a linguagem React Native devido a praticidade proporcionada pelo Expo.

Foram pesquisados diversos modelos de braços robóticos, com isso foi escolhido um modelo de acrílico com 5 graus de liberdade que foi adquirido e montado. As plataformas microcontroladas escolhida foi o Arduino por causa da disponibilidade do módulo *Sensor Shield*

que permitia a gestão de vários servomotores. Mesmo que também tenha sido testado o uso do módulo NodeMCU para proporcionar conexão via wi-fi.

Foi desenvolvido um aplicativo para controle do braço robótico via Bluetooth que permitiu controlar o braço robótico com precisão, e também um site que permitiu enviar informações via wi-fi para o NodeMCU.

Diversos testes foram realizados para avaliar o desempenho das conexões disponíveis, e também o funcionamento do braço robótico. Com isso foi constatado que a conexão via Bluetooth se mostrou suficientemente rápida e precisa, porém apresentou um bug durante os testes que impossibilitou o uso do módulo (Um novo módulo foi encomendado para descartar possível falha física). A conexão wi-fi aparenta estar limitada ao envio de dados apenas quando o slider for solto, porém mais testes podem ser feitos para confirmar essa análise. O braço robótico não conseguiu suportar o próprio peso. Suspeita-se que a falta de força se deva à fonte de energia pois o uso de dois servomotores MG996R demonstram ser mais do que suficiente para manter firme o braço como um todo. Também foi visto que o braço robótico conseguiu ser muito responsivo em relação aos dados enviados pelo Bluetooth. Porém ele tem o potencial de tombar devido aos movimentos bruscos, precisando então ser preso à uma base temporária enquanto as melhorias descritas logo abaixo não são implementadas.

7 Referências

- [aKa18] a, K. R., b, T. A., & Abbas, N. (27 de março de 2018). Kinematic analysis and geometrical improvement of an industrial. *Science Direct*, 30, pp. 218-223. doi: <https://doi.org/10.1016/j.jksues.2018.03.005> Citado na página 1.
- [Ard21] Arduino. (2021). *environment*. (SM, Editor) Fonte: Arduino: <https://www.arduino.cc/en/guide/environment> Citado na página 2.
- [Ban19] Bane, A., Daoui, M., Bouzefrane, S., & Muhlethaler, P. (Abril de 2019). NDN-over-ZigBee: A ZigBee support for Named Data Networking. *Science Direct*, 93, pp. 792-798. doi: <https://doi.org/10.1016/j.future.2017.09.053> Citado na página 5.
- [Biø19] Biørn-Hansen, A., Grønli, T.-M., Ghinea, G., & Alouneh, S. (03 de janeiro de 2019). An Empirical Study of Cross-Platform Mobile. (G. Canfora, Ed.) *Hindawi*, 2019, p. 12. doi: <https://doi.org/10.1155/2019/5743892> Citado na página 2.
- [Cap21] Capes. (2021). Buscar Assunto. Fonte: Periódicos Capes: <https://www-periodicos-capes-gov-br.ez81.periodicos.capes.gov.br/index.php> Citado na página 2.
- [Dav20] Davidson, K. (18 de novembro de 2020). *react-native-bluetooth-classic*. Fonte: GitHub: <https://github.com/kenjdavidson/react-native-bluetooth-classic> Citado na página 12.
- [Dav21] Davidson, K. (s.d.). *React Native Bluetooth Classic*. Acesso em 2021, disponível em kenjdavidson: <https://kenjdavidson.com/react-native-bluetooth-classic/> Citado na página 12.
- [Dun17] Dunbar, B. (03 de março de 2017). *Curiosity Rover*. (T. Greicius, Editor) Acesso em 2021, disponível em NASA: https://www.nasa.gov/mission_pages/msl/overview/index.html Citado na página 1.
- [Dun171] Dunbar, B. (03 de março de 2017). *Spirit and Opportunity*. (T. Greicius, Editor) Acesso em 2021, disponível em NASA: https://www.nasa.gov/mission_pages/mer/overview/index.html Citado na página 1.
- [Ele20] Electronics, E. G. (2021). Fonte: Curto Circuito: https://curtocircuito.com.br/datasheet/arduino_sensor_shield.pdf Citado na página 14.
- [era20] eranbo. (dezembro de 2020). *react-native-base64*. Acesso em 2021, disponível em npmjs: <https://www.npmjs.com/package/react-native-base64> Citado na página 12.
- [ETC14] ETC. (2014). Acesso em 2021, disponível em datasheetpdf: <https://datasheetpdf.com/pdf/942981/ETC/MG996R/1> Citado na página 13.
- [FAQ21] FAQ. (31 de março de 2021). Fonte: Visual Studio Code: <https://code.visualstudio.com/docs/supporting/faq> Citado na página 3.
- [Flo21] Flop, F. (s.d.). *micro-servo-9g-sg90-towerpro*. Acesso em 2021, disponível em Filipe Flop: <https://www.filipeflop.com/produto/micro-servo-9g-sg90-towerpro/> Citado na página 13.

- [Jaw18] Jaworski, T. S., & Noichl, S. (03 de abril de 2018). Evaluation of Cross-Platform App Development using React Native. doi: <http://doi.org/10.18154/RWTH-2018-223392>
Citado na página 2.
- [Koy18] Koyanagi, F. (09 de janeiro de 2018). *ESP32 Longa Distância - LoRaWan*. Fonte: Fernando K: <https://www.fernandok.com/2018/01/esp32-longa-distancia-lorawan.html>
Citado na página 5.
- [Koy181] Koyanagi, F. (31 de julho de 2018). *Protocolo CAN - Yes, We Can!* Fonte: Fernando K: <https://www.fernandok.com/2018/07/protocolo-can-yes-we-can.html> Citado na página 5.
- [Koy21] Koyanagi, F. (2021). *Tutoriais, Tecnologias, Tendências*. Fonte: Fernando K Tecnologias: <https://www.fernandok.com/> Citado na página 2.
- [Lar201] Lara, C. (1 de março de 2020). *7 - React Native en español | Utilizando la dependencia bluetooth*. Acesso em 2021, disponível em Youtube: <https://youtu.be/LlIkwiK4hz8?list=PLTYm84ujubwL3aYYWkU9FoZBrkW9GFy2G&t=469>
Citado na página 12.
- [Lar20] Lara, C. (29 de fevereiro de 2020). *WorkShop React Native + Bluetooth*. Acesso em 2021, disponível em Youtube: <https://youtube.com/playlist?list=PLTYm84ujubwL3aYYWkU9FoZBrkW9GFy2G> Citado na página 12.
- [Lee18] Leenheer, N. (10 de janeiro de 2018). *Fun With Bluetooth*. Munique, Baviera, Alemanha. Acesso em 2021, disponível em Youtube: <https://youtu.be/XDc5HUVMI5U> Citado na página 2.
- [Lee20] Lee, H.-W. (19 de junho de 2020). The Study of Mechanical Arm and Intelligent Robot. *IEEE access*, 8, pp. 119624-119634. doi: <https://doi.org/10.1109/ACCESS.2020.3003807>
Citado na página 1.
- [Nat21] Native, R. (2021). Acesso em 2021, disponível em React Native: <https://reactnative.dev/> Citado na página 10.
- [Nik19] Nikravan, M., Movaghar, A., & Hosseinzadeh, M. (11 de maio de 2019). Springer. *Peer-to-Peer Networking and Applications*, 12, pp. 209-226. doi: <https://doi.org/10.1007/s12083-018-0659-8> Citado na página 5.
- [Plu18] Plus, D. (2018). *Curso de React Native*. Acesso em 2021, disponível em Youtube: <https://youtube.com/playlist?list=PLxF2lyHGcERApnjQPgeeElzJdGurraMW> Citado na página 2.
- [Plu19] Plus, D. (2019). *curso-de-react-native*. Acesso em 2021, disponível em Developer Plus: <https://developerplus.com.br/category/curso-de-react-native/> Citado na página 2.
- [Pro14] Pro, T. (2014). Acesso em 2021, disponível em datasheetpdf: <https://datasheetpdf.com/pdf/791970/TowerPro/SG90/1> Citado na página 13.
- [San181] Santos, R. (20 de fevereiro de 2018). *Build an ESP8266 Web Server with Arduino IDE - Code and Schematics*. Acesso em 2021, disponível em YouTube: https://www.youtube.com/watch?v=dWM4p_KaTHY Citado na página 9.

- [San14] Santos, R. L. (2014). *Internet das Coisas e 6LoWPAN*. Monografia, Universidade Tecnológica Federal do Paraná, Curitiba. Acesso em 02 de Abril de 2021, disponível em http://repositorio.utfpr.edu.br/jspui/bitstream/1/17312/2/CT_GESER_V_2014_12.pdf Citado na página 1.
- [San18] Santos, R., & Santos, S. (20 de fevereiro de 2018). *ESP8266 Web Server with Arduino IDE*. Acesso em 2021, disponível em Random Nerd Tutorials: <https://randomnerdtutorials.com/esp8266-web-server-with-arduino-ide/> Citado na página 9.
- [Sch19] Schwarzmüller, M. (14 de agosto de 2019). *Which one is best for you? Flutter, React Native, Ionic or NativeScript?* Acesso em 2021, disponível em YouTube: <https://youtu.be/PKRXbLnfXXk> Citado na página 10.
- [Sea21] Seat, R. (2021). *React Native*. Fonte: Next Level Week: <https://nextlevelweek.com/episodios/reactnative/3/edicao/5> Citado na página 2.
- [Ste18] Stevan Jr., S. (2018). *IoT Internet das coisas Fundamentos e aplicações em Arduino e NodeMCU*. (S. C. Ferreira, Ed.) São Paulo, São Paulo, Brasil: Saraiva. Acesso em 2021 Citado na página 1.
- [Tre21] *Treinamentos*. (2021). Fonte: Periódicos Capes: https://www.periodicos-capes.gov.br/ez81.periodicos.capes.gov.br/index.php?option=com_ptreinamentos&Itemid=306 Citado na página 2.
- [Vis21] (2021). Fonte: Visual Studio Code: <https://code.visualstudio.com/> Citado na página 3.
- [Wil21] William. (2021). *DroneBot Workshop*. Fonte: DroneBot Workshop: <https://dronebotworkshop.com/> Citado na página 2.

8 Perspectivas de continuidade do trabalho

Espera-se que para o prosseguimento do trabalho seja aplicadas todas as mudanças descritas na Seção 5, bem como a análise de um design mais prático para o armazenamento de energia do braço, algo que seja prático de se colocar e remover. Seria interessante também ter um retorno do quanto de bateria ainda resta até que o braço pare de funcionar. Essa informação seria passada do braço para o app que a está controlando, permitindo um melhor monitoramento do braço e de seus acessórios anexos.

Como o braço apresenta ainda mais graus de liberdade do que o que foi montado pela bolsista anterior, ele apresenta maior instabilidade em sua base devido ao peso de seus componentes. Uma forma de estabilizá-lo seria uma outra melhoria promissora para o desempenho do projeto. Imagina-se que o uso de contrapesos possa ajudar na estabilidade, ou então graxa nas articulações, ou então um tipo de grampo que permita fixá-lo à bancada de trabalho ou a quaisquer superfícies necessárias.

Na continuidade da pesquisa, existem diversos aprimoramentos possíveis ao atual projeto. Os seguintes aspectos podem ser contemplados: hardware, código do Arduino e código do aplicativo.

8.1 Hardware

- a) Analisar o consumo de energia para averiguar se é isto que causa a insuficiência de força no braço robótico;
- b) Impedir que o braço robótico tombe (possivelmente utilizando pesos em sua base, aumentando a área da base, ou fixando sua base);
- c) Utilizar o módulo Bluetooth e o módulo wi-fi simultaneamente (melhoria operacional);
- d) Remover excessos dos parafusos e buscar parafusos mais apropriados (estética);
- e) Pesquisar sobre o uso de capacitores para evitar o “*jittering*” (efeito que causa tremor durante a operação dos servomotores);
- f) Pesquisar sobre o uso do Módulo I2C p/ Servo Motor - PCA9685 (alternativa para o módulo Sensor Shield, que permite o uso da comunicação serial I2C);
- g) Pesquisar sobre as tecnologias 6LoWPAN e BLE
- h) Adquirir ESP32

8.2 Software Arduino

- a) Posição inicial dos Servomotores (Configurar para manterem na mesma posição ao ligar);
- b) Impedir que o braço robótico encoste no chão (Limitar as angulações dos diversos servomotores);
- c) Tornar Servos mais suaves durante a movimentação (Pesquisar sobre o uso da Biblioteca ServoEasing e sobre o controlador PID);
- d) Garantir a proteção dos servos ante esforço excessivo (youtu.be/LKJLCJvyVdk);
- e) Pesquisar sobre o desenvolvimento de interface gráfica para microcontrolador: remotexy.com (youtu.be/2cjufbgOBYo);
- f) Mudança em relação a movimentação do braço:
 - a. Se mover nos eixos X, Y e Z;

- b. Se mover nas angulações yaw (guinada), pitch (arfagem) and roll (rolamento), mantendo a extremidade do braço centralizada;
- c. Estudo sobre Cinemática Direta (Input: ângulos das articulações; output: coordenadas x, y, z) e Cinemática inversa (Input: coordenadas x, y, z; Output: ângulos das articulações)

8.3 Software App

- a) Reformular toda a estrutura de variáveis referentes aos estados de conexão;
- b) Não mostrar na lista de dispositivos aqueles que estiverem desligados;
 - a. Após se conectar, não mostrar o componente de conexão até que perca a conexão.
 - b. Quando um dispositivo estiver conectado mostrar um único botão com os dados do dispositivo conectado que possa ser tocado para desconectar e voltar a mostrar a lista de componentes disponíveis;
- c) Atualizar a lista de dispositivos disponíveis sempre que algum deles mudar de estado;
- d) Permitir a conexão via wi-fi no mesmo App;
- e) Fazer com que os Sliders desapareçam enquanto estiver desconectado;
- f) Colocar a lista de conexão *Bluetooth* em uma tela diferente de onde estiverem os sliders;
 - a. Se a conexão for interrompida e o usuário estiver na tela dos sliders, retornar o usuário de volta para a tela de conexões;
- g) Tornar o App funcional mesmo que esteja na horizontal;
- h) Consertar a demora de resposta do Switch do Bluetooth;
- i) Entender o motivo da função “setTextSlider” reduzir a frequência de repetição da função “sendSlider” e encontrar uma solução;
- j) Tornar os componentes do app modulares;
- k) Melhorar o visual de todo o aplicativo tornando mais intuitivo e atraente.


9 Participação em congressos e trabalhos publicados ou submetidos e outras atividades acadêmicas e de pesquisa

Parte desta pesquisa foi publicada pelo bolsista no ano 2020 no evento CONFICT 2020, que retrata o uso de um app para Smartphone Android desenvolvido na plataforma MIT App Inventor e envia dados mediante conexão Bluetooth para uma versão mais simples de braço robótico microcontrolado.

O bolsista participou de um programa multidisciplinar de Web Treinamento do Portal de Periódicos da CAPES sobre Pesquisas Avançadas em bases de dados acadêmicas e científicas via plataforma EBSCOhost.

10 Data e assinatura do bolsista

30 de abril de 2021.

A handwritten signature in dark ink, appearing to read "Jofelias". The signature is written in a cursive style with a large, circular loop at the beginning.

11 Data e assinatura do orientador

30 de abril de 2021.

A handwritten signature in dark ink, appearing to read "H. L. F. M.". The signature is written in a cursive style with a large, circular loop at the beginning.