# Tudo

## Legenda de revisão:

DIA 3) 3060->3780

Azul: Copiado do relatório anterior

Vermelho: Apagar posteriormente/Suspeito que esteja errado

Laranja: Revisar

Verde: Me parece correto

XX: Substituir por algo posteriormente

## Capa

**PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA E TECNOLÓGICA**

**UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO**

CENTRO CCT

LABORATÓRIO LCMAT

Relatório do período: 04/2020 a 03/2021

**RELATÓRIO ANUAL ou RELATÓRIO FINAL**

Nome do Bolsista/Voluntário: João Vítor Fernandes Dias

Curso: Ciência da Computação

N° 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

CORPO DO RELATÓRIO DEVE CONTER OS SEGUINTES TÓPICOS:

### **Etapas propostas no plano de trabalho (justificar o não cumprimento de alguma etapa, se for o caso)**

#### Etapas propostas no plano de trabalho

1. 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.
2. Estudo sobre alternativas de desenvolvimento de aplicativos Android visando o desenvolvimento de interfaces de controle e monitoramento para dispositivos microcontrolados. Documentação do estudo.
3. Estudo, desenvolvimento e implementação do segundo protótipo de braço robótico. Escolha da plataforma microcontrolada. Documentação.
4. Desenvolvimento da interface de controle e monitoramento do braço robótico. Documentação.
5. Realização de testes de avaliação e desempenho da interface e do braço robótico.
6. 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 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 e até então não pôde ser substituído devido à necessidade de ir ao laboratório que se torna inviabilizada devido à pandemia do covid-19. A interface desenvolvida para o item *d* se mostrou eficiente, porém com vários bugs que podem 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.

#### Diversas tecnologias de controle remoto foram pesquisadas e documentadas. Sendo elas:

RFid – Não apropriado: Ações limitadas ao que for programado em cartões de RFid, inviável devido a necessidade da alta taxa de repetição e controle preciso dos servomotores.

Protocolo CAN – Não apropriado: Requere um dispositivo de envio e outro de recebimento, permite envio de até 11 bits ou 29 bits. Parece viável, mas não para o caso do controle por Smartphone, a não ser que fosse configurado com o uso de dois arduínos conectados entre si via MCP2515, um ligado aos servomotores e o outro para fazer conexão com o Smartphone precisaria ainda de outro tipo de conexão remota, o que aumentaria desnecessariamente a complexidade, visto que se pode fazer o mesmo com o bluetooth sem a necessidade do MCP.

ESP32 LoRaWan – Não apropriado: Depende de outro LoraWan, e o foco da pesquisa é o controle do braço robótico através do Smartphone, o que faz com que entre no mesmo critério anterior

ZigBee – Não apropriado: Segundo o [INSERIR SINTAXE CORRETA PARA REFERENCIAMENTO DAS FONTES CIENTÍFICAS – ARQUIVO “ZIGBEE”] 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. 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 o [INSERIR SINTAXE CORRETA PARA REFERENCIAMENTO DAS FONTES CIENTÍFICAS – ARQUIVO “6LoWPAN”] “o ‘IPv6 em rede sem fio pessoal de baixa energia’ (IPv6 over Low power Wireless Personal Area Network - 6LoWPAN) é uma adaptação se 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, ...

[BUSCAR O MOTIVO DE NÃO SE TER USANDO O 6LOWPAN]

WiFi (WLAN) – Apropriado: Tecnologia presente na maioria dos Smartphones atuais, geralmente utilizado para se conectar à internet. Apresenta diversos módulos para o uso com o Arduíno, podendo ser citados XX e XX. Também estão disponíveis vários outros microcontroladores, como o ESP32 e o ESP8266, que já têm seu próprio módulo WiFi embutido.

Bluetooth (WPAN) – Apropriado: Assim como o WiFi, também é uma tecnologia presente na maioria dos Smartphones atuais, geralmente utilizado para se conectar dispositivos sem fio, principalmente fones de ouvido. Apresenta diversos módulos para o uso com o Arduíno, podendo ser citados o HC-05 (utilizado nesse projeto) e o HC-06. Também estão disponíveis vários outros microcontroladores, como o ESP32 e o XX, que já têm seu próprio módulo Bluetooth embutido.

Bluetooth Low Energy (BLE) – XX: Pesquisar posteriormente.

Conclusão: Tecnologias apropriadas para o projeto atualmente são WiFi e Bluetooth. Podem ser utilizados os módulos XX e XX para a conexão WiFi com o Arduíno, ou as placas ESP8266 e ESP32 para a conexão WiFi embutida. Podem ser utilizados os módulos HC-05 e HC06 para a conexão Bluetooth com o Arduíno ou as placas ESP32 e XX para a conexão Bluetooth embutida.

#### Diversas linguagens de programação foram pesquisadas e documentadas. Sendo elas:

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 – 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.

Kotlin – XX: Possui as mesmas desvantagens do Java

Ionic – XX

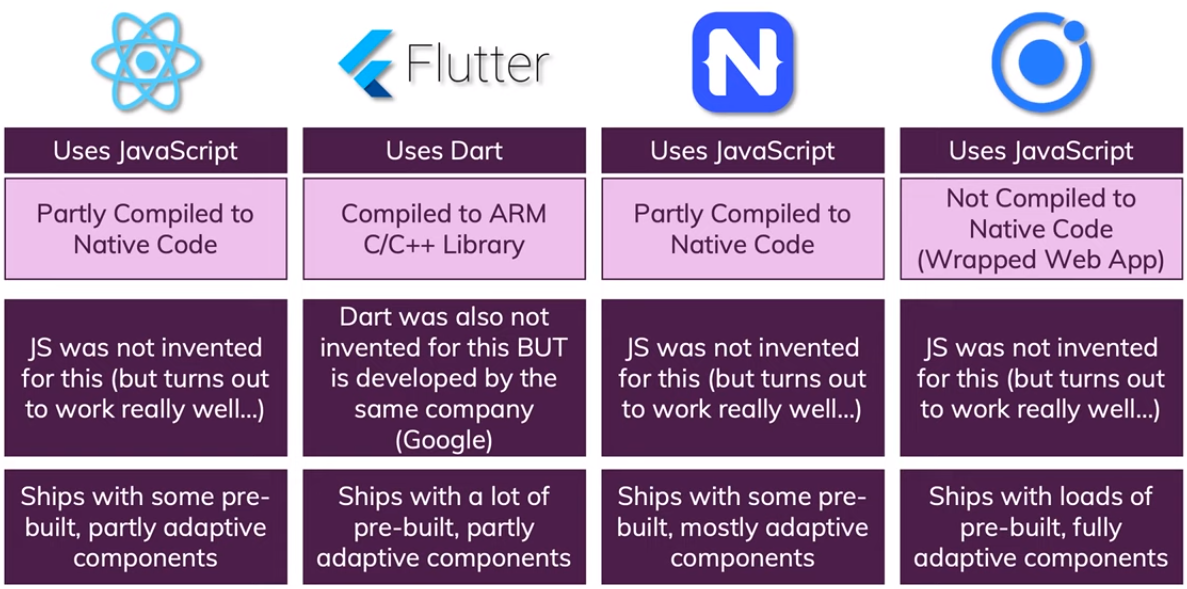
JavaScript (Back-end) – XX

NativeScript? – XX

Flutter – Apropriado

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 (XX para desenvolvimento web) e JavaScript (linguagem back-end). Além disso pode ser desenvolvido utilizando o Expo, que é um XX de XX. Com o Expo é possível programar em React Native e quase que instantaneamente ver o app resultante em um Smartphone conectado à mesma rede WiFi que tenha o app do Expo. Porém o Expo tem limitações: utilizando-o não é possível importar bibliotecas (agrupamentos de código que permitem o acesso a novas funcionalidades) que são/serão/foram [TANG, QUAL TEMPO VERBAL USAR? “SERÃO UTILIZADAS DURANTE O PROJETO” (CONSIDERANDO QUE O PROJETO ESTEJA EM ANDAMENTO AO LONGO DO TEXTO), “SÃO UTILIZADAS DURANTE O PROJETO” (CONSIDERANDO QUE AS BIBLIOTECAS ESTÃO ATUALMENTE SENDO UTILIZADAS NO PROJETO) OU “FORAM UTILIZADAS DURANTE O PROJETO” (CONSIDERANDO QUE O PROJETO JÁ ESTÁ ENCERRADO E QUE O QUE ESTÁ SENDO RELATADO É ALGO DO PASSADO)] utilizadas no projeto.

##### Tabelas comparativas







#### Diversos modelos de braços robóticos foram pesquisados, e também plataformas microcontroladas, como citado no item a.

Optou-se por um modelo de braço robótico com 5 graus de liberdade que tem 5 servomotores “Servo Motor MG996R Tower Pro” que são mais potentes e trazem maior resistência ao peso da estrutura.

Quanto às plataformas microcontroladas foram pesquisadas diversas opções como:

Placas que têm o chip ESP8266: ESP01, ESP12E, ESP12F, ESP201, NODEMCU-ESP12 e CP2102.

Placas que tem o chip ESP32: ESP32-WROVER, ESP32-WROOM-32U, ESP32S-CP2102 e NODEMCU V3 WIFI 802.11.

Placas que têm o chip WEMOS-D1-R2: Wemos d1, Wemos D1 WiFi Esp8266, D1 Mini NodeMCU Esp8266, Wemos D1R2 Wi-Fi.

De início optou-se por manter o desenvolvimento com o Arduíno, 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 ESP8266, NODEMCU-AMICA, NODEMCU-LOLIN e NODEMCUESP32 XX.

#### A interface de controle e monitoramento do braço robótico foi desenvolvida

Utilizando da XX React Native, foi desenvolvido um app que se conectasse ao módulo Bluetooth HC-05 e controlasse remotamente os servomotores presentes no braço robótico através de sliders. Porém apresenta alguns bugs e diversos pequenos aprimoramentos podem ser feitos, tanto em sua estética quanto em suas funcionalidades.

#### Diversos testes de avaliação e desempenho da interface e do braço robótico foram feitos e documentados.

Quanto a interface, um dos primeiros aprimoramentos foi em relação ao código. No React Native há a possibilidade de se dividir o código em diversos arquivos diferentes. Essa abordagem tem vantagens e desvantagens.

Vantagens:

* reduz a quantidade de código presente em um único arquivo, deixando-o mais limpo;
* Torna mais modular o projeto e permite que partes do app se resumam a blocos de código que poderão ser modificados sem interferir muito no resto do app;
* Torna mais fácil encontrar os erros que ocorrem durante o desenvolvimento.

Desvantagens:

* Aumenta o número de arquivos referentes ao projeto, o que pode causar confusão;
* Oculta os nomes das funções que se estivessem em um único programa estariam listadas.

Atualmente o código do app está com uma mistura. Parte do código está encapsulado em funções e componentes, porém encontram-se no arquivo principal. Vê-se uma mudança viável para um app mais modular com diversos arquivos organizados.

Esse encapsulamento permite, por exemplo, que um conjunto de componentes sejam agregados e generalizados, permitindo que a quantidade de código reduza e o torne mais limpo. Porém quando essa estratégia [O ATO DE ENCAPSULAR CÓDIGO É O QUÊ? UMA ESTRATÉGIA? UM MÉTODO? UMA FORMA? UM ESTILO DE PROGRAMAÇÃO?] foi utilizada no conjunto de Sliders um problema surgiu: Ao movimentar o slider para uma certa posição e soltá-lo, ele voltava para o centro. Esse problema foi resolvido ao colocar o componente do slider no escopo global do código, bem como as variáveis referentes aos valores iniciais, mínimo e máximo do slider.

Uma outra situação que ocorreu com o componente Slider e que independe do encapsulamento foi a taxa de envio e atualização dos valores presentes no slider. Isso ocorreu por causa de um tipo de variável (uma variável state) que está sendo utilizado para guardar o valor. Ao utilizar uma variável *var* ao invés de um *state* o envio de dados se torna mais fluido, porém o texto que representa o valor atual do slider não se modifica. Como medida provisória que tem o melhor envio de dados encontrado e que altera o texto é adicionar uma característica ao componente slider que permite que uma função seja executada assim que o slider conclua sua movimentação, permitindo que nesse momento o valor textual do slider seja modificado.

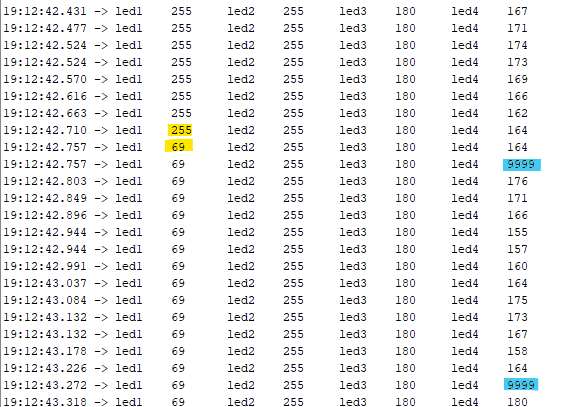
Não foi apenas o componente slider que deu problema. Também houveram diversas complicações com a conexão bluetooth.

Algo que ainda precisa melhor implementado são os estados presentes na conexão Bluetooth. São diversas situações que precisam ser consideradas e que ainda não foram implementadas ao programa, por exemplo: Ao se escanear os dispositivos, os que foram escaneados são adicionados a uma lista. Porém, quando um for desligado, ele deveria sair da lista e a mesma ser atualizada, porém isso não ocorre atualmente. Outro exemplo é que, caso esse dispositivo esteja conectado e se desconecte, além de sair da lista, o app deveria também encerrar a conexão que havia com ele.

Dois bugs foram encontrados durante o desenvolvimento da transmissão de dados via bluetooth. O primeiro ocorria da seguinte forma:

Estava sendo testado novamente o envio de valores via Bluetooth utilizando o app desenvolvido no MIT App Inventor para o projeto anterior. Nele estão definidas as variáveis LED1, LED2, LED3 e LED4, sendo que os LEDs 1 e 2 de fato representam LEDs, podendo seu valor variar entre 0 e 255. Entretanto, os LEDs 3 e 4 representam servomotores que têm 180° de rotação, então seu valor pode variar entre 0 e 180. Entretanto, enquanto um slider estava recebendo dados, algo acontecia internamente que fazia com que o valor de uma variável referente a outro slider acabava se alterando para um novo valor aparentemente aleatório. Como pode ser observado em amarelo na imagem abaixo.

Para tornar o erro mais visível foi adicionado uma linha no código para que qualquer valor acima de 257 se tornasse 9999 como pode ser observado em azul na imagem abaixo.



Após novo app ter sido desenvolvido e vários novos testes de aperfeiçoamento terem sido feitos e estarem funcionando normalmente, passou a se testar o recebimento de dados utilizando o módulo HC-05 com a placa NodeMCU Lolin inicialmente os valores estavam sendo recebidos normalmente, até em algum momento passaram a serem recebidos os seguintes valores:

[ADICIONAR A IMAGEM DO BUG AQUI]

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”.

Até então esse bug não foi solucionado. Supõe-se que esse erro se deva à possibilidade do módulo HC-05 estar danificado, porém essa afirmação não é assertiva, visto que não houve nenhuma modificação externa aparente que possa ter causado essa mudança.

[ADICIONAR AQUI A SOLUÇÃO DO PROBLEMA]

Um aprimoramento geral para o app é em relação a sua parte visual. Não apenas tornar os estilos mais atraentes, mas também tornar o visual mais limpo. Lista de aprimoramentos futuros:

1. Não mostrar os sliders enquanto não houver conexão com o Arduíno;
2. Não mostrar a listagem dos dispositivos enquanto estiver conectado;
3. 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;
4. Melhorar a estética dos componentes;
5. Ajustar os estados de conexão do dispositivo;
6. Ajustar a listagem dos itens da lista para remover os que forem desconectados;

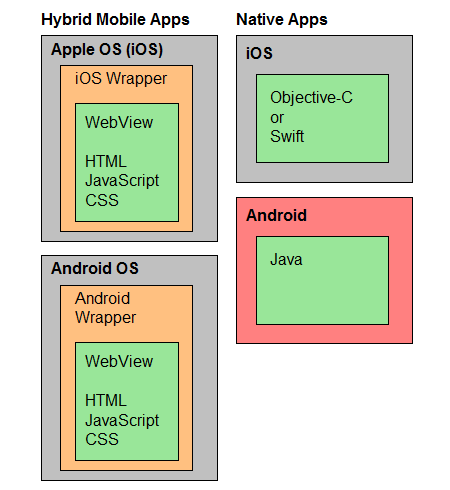
### **Introdução**

Os aplicativos nativos são aqueles que utilizam diretamente da base do sistema operacional em que estão instalados e utilizam de suas funções através do desenvolvimento pela mesma linguagem em que os Sistemas Operacionais foram desenvolvidos.

Os aplicativos não-nativos geralmente estão encapsulados o que permite até certo ponto mais liberdade em relação às diferentes linguagens que podem ser utilizadas, porém pode acabar se tornando limitada por causa dos componentes nativos que podem acabar não sendo acessados

[CHECAR A VERACIDADE DAS INFORMAÇÕES ESCRITAS]

#### Falar sobre a diferença entre aplicativos híbridos e nativos



### **Objetivos**

Os objetivos principais desse projeto são desenvolver através da programação um app que permita utilizar o Bluetooth e Wi-Fi de um smartphone Android para controlar remotamente um braço mecânico, e também montar um novo braço mecânico para tornar o projeto do novo bolsista independente do que foi desenvolvido pela bolsista anterior.

### **Metodologia [acrescentar posteriormente]**

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 é necessário para que o app possa ser atualizado rapidamente após a programação e também o Expo para que essa atualização seja ainda mais rápida.

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). 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.

[MELHORAR A FORMA DE EXPLICAR COMO O EXPO, NODE JS E REACT NATIVE FUNCIONAM E O QUE SÃO (TEREI QUE PESQUISAR, PORQUE EU NÃO SEI)]

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 Arduíno pode-se desenvolver o app para que ele enviasse os dados devidamente. Devido à forma como o Arduíno 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.

[BUSCAR FUNDAMENTO TEÓRICO PARA EXPLICAR O QUE É A BASE 64]

Após o desenvolvimento do app que enviava os valores dos sliders apropriadamente para o HC-05 (mesmo que estes pudessem ter sua eficiência aprimorada), passou a serem testados os outros dispositivos microcontrolados utilizando este mesmo app. Conectando o HC-05 ao NodeMCU os valores recebidos puderam ser lidos adequadamente, até que inesperadamente os valores recebidos pelo HC-05 mudassem sem nenhum motivo aparente. Esse bug até então não foi corrigido e nem compreendido, espera-se que até o final do relatório uma explicação e/ou uma solução sejam encontradas.

[MODIFICAR O TEXTO ACIMA PARA NÃO HAVER ESSA EXPECTATIVA NO FINAL. AO INVÉS DISSO HAVER UMA SOLUÇÃO DE ACORDO E REAL]

A montagem do braço robótico durou em torno de 10 horas. De um modo geral não foi muito complicado, porém as instruções no manual não eram muito claras o que reduziu o rendimento. Assim como os erros próprios cometidos por falta de atenção, além da existência de peças que não encaixavam adequadamente e necessitaram de um pouco de força para se fixarem devidamente, e também parafusos que eram mais longos do que necessário.

[ANOTAÇÕES DOS TESTES COM OS SERVOMOTORES:

O Arduino Sensor Shield v5.0 permite superar a capacidade do módulo Arduino

https://www.youtube.com/watch?v=IyGwvGzrqp8

Serial Communication Protocols:

UART – Universal Asynchronous Receive-Transmitter

Transmission and Reception pins

Transmit and Receive pins

TX and RX pins

1. Transmission speed (Baud/second)
2. Data Lenght
3. Start and Stop bits

Number 192: 11000110

Start bit:

End bit:

Usually the UART signal is Always high and the start bit will be represented as a low pulse

9600 bauds/second=

1/9600= 104 microseconds

After receiving everything sends from buffer to rest of the system

UART RS-232:

DTR and CTS pins

Used to notify that the Data Terminal is Ready (DTR) to receive and the receiver is Clear To Send (CTS).

Synchronous communication:

I2C (inter-integrated Circuit)

SDA, SCL

Serial Data (SDA) - Send the Data

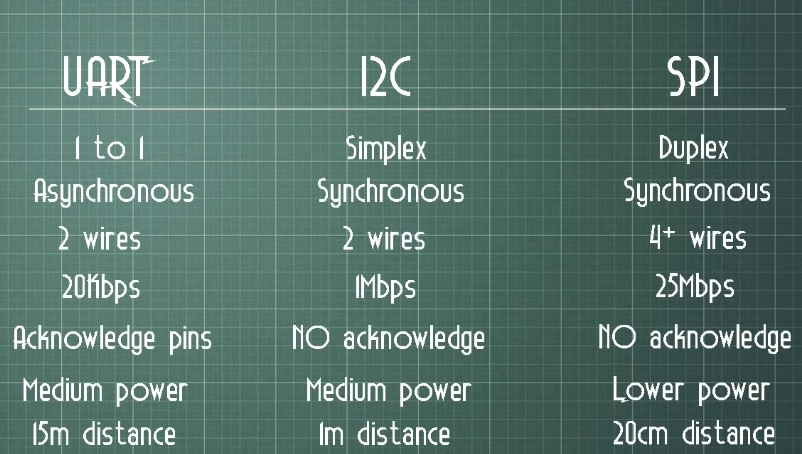
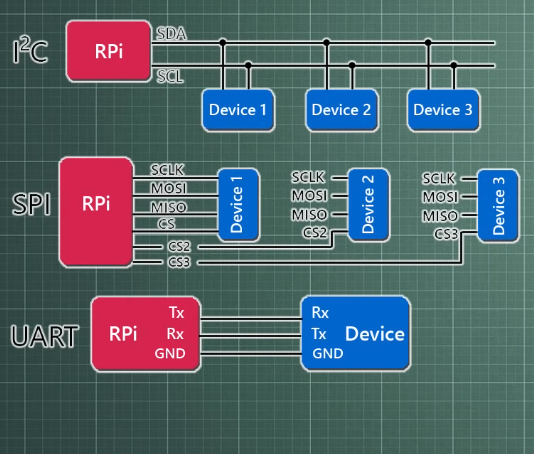
Serial Clock (SCL) – The clock

Needs data lenght and frequency

Another Synchronous Communication:

SPI – Serial Peripheral Interface

**Pins:  
Serial Clock (SCLK), Master Output Slave Input (MOSI), Master Input Slave Output (MISO), Slave Select (SS).**

****

**Bluetooth usa UART**

]

### **Resultados e Discussão (em um único tópico ou em tópicos separados)**

Como resultado vê-se positivamente a montagem do braço robótico e do aprendizado adquirido para o desenvolvimento do app. As etapas do plano de trabalho foram em sua maioria concluídos com êxito, restando apenas resolver o bug do HC-05, a implementação da conexão via Wi-Fi, analisar e montar o circuito elétrico necessário para manter ativos os servomotores do braço mecânico e aplicar as melhorias propostas ao app.

[AVANÇAR NO PROJETO O SUFICIENTE PARA REMOVER O MÁXIMO DE COISAS NEGATIVAS QUE FORAM ESCRITAS E QUE FALTAM SER CONCLUÍDAS]

### **Conclusões (demonstrando que cumpriu o plano de trabalho referente a este relatório)**

### **Referências**

[LEMBRE-SE DO QUE FOI DITO NA AVALIAÇÃO DO ÚLTIMO RELATÓRIO: **“Recomendo a atualização de todas as referências colocadas no relatório, ou seja, verificar a estrutura de escrita dessas referências.”**

**ANALISAR A ESTRUTURA UTILIZADA NO RELATORIO ANTERIOR (REALMENTE ESTÁ ESTRANHO. PARECE QUE NÃO FOI REVISADO)**

**LISTA DE ERROS:**

1. **LIVRO DE KLAUS SCHWAB SEM NUMERAÇÃO**
2. **LINKS GRANDES DEMAIS?**
3. **MUITOS LINKS?**
4. **LINKS SEM NUMERAÇÃO**

]

### **Perspectivas de continuidade ou desdobramento do trabalho, no caso de solicitação de renovação**

Espera-se para o prosseguimento do trabalho seja aplicadas todas as mudanças descritas em resultados e Discussão e também comentadas no código do app, 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.

[CONFERIR SE (CASO CONSIGA CONCLUIR A PARTE ENERGÉTICA DO BRAÇO A TEMPO) O BRAÇO CONSEGUE SE MANTER ESTÁVEL SEM CAPOTAR. CASO NÃO CONSIGA, ADICIONAR A SEGUINTE PARTE:

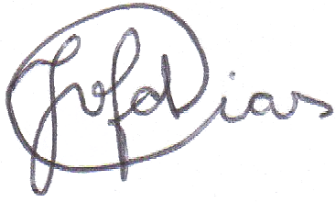
Como o braço apresenta ainda mais graus de liberdade do que o que foi montado pela bolsista anterior, ele apresenta maior instabilidade em sua base devido ao peso de seus componentes. Uma forma de estabilizá-lo seria uma outra melhoria promissora para o desempenho do projeto. Imagina-se que o uso de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

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

Como parte da Semana Acadêmica do Instituto Federal Fluminense, o bolsista participou de um minicurso sobre ambiente de programação visual chamado MIT App Inventor.

Foi publicado um trabalho científico relativo ao ano de 2020 dessa pesquisa no CONFICT 2020 que retrata o uso de um app para Smartphone Android desenvolvido na plataforma MIT App Inventor que envia dados para um braço robótico microcontrolado através da conexão Bluetooth.

### **Data e assinatura do bolsista (assinatura digitalizada)**



XX 30 de abril de 2020. de abril de 2021.

### **Data e assinatura do orientador (assinatura digitalizada)**

****

XX 30 de abril de 2020. de abril de 2021.

# Fim