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

**UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO**

CENTRO CCT

LABORATÓRIO LCMAT

Relatório do período: 10/2019 a 04/2020

**RELATÓRIO ANUAL ou RELATÓRIO FINAL**

Nome do Bolsista/Voluntário: João Vítor Fernandes Dias

Curso e N° Matrícula: Ciência da Computação / 00119110377

Orientador: Fermín Alfredo Tang Montané

Título do Projeto:Desenvolvimento e controle de dispositivos para Internet das Coisas

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**

[HAVER ESSA DISTINÇÃO ENTRE O QUE ESTAVA PROPOSTO NO PLANO DE TRABALHO E A FORMA E ORDEM QUE OPTEI POR UTILIZAR É ACEITÁVEL?]

O plano de trabalho compreende as seguintes etapas propostas pela antiga bolsista:

1. Estudo da Plataforma Raspberry PI. Configuração da plataforma. Execução de projetos simples para familiarização com a plataforma. Documentação do estudo e dos projetos executados.
2. Estudo sobre formas de integração entre as plataformas Arduino, NodeMCU e Raspberry Pi. Pesquisa sobre projetos de integração e documentação do estudo.
3. Estudo sobre controle aprimorado de motores e servomotores. Introdução das melhorias nos projetos do braço robótico e da Planta IoT. Documentação.
4. Estudo sobre formas otimizadas de alimentação de energia para projetos com motores e servomotores. Introdução das melhorias nos projetos do braço robótico e da Planta IoT. Documentação.
5. Pesquisa sobre interfaces de controle para plataformas microcontroladas. Documentação.
6. Desenvolvimento das interfaces de controle nos projetos do braço robótico e da Planta IoT.
7. Realização de experimentos de avaliação e desempenho dos dispositivos.
8. Elaboração de relatório técnico.

Optou-se por seguir uma sequência diferente de prioridades, visto que a bolsista anterior já havia executado um projeto simples na plataforma Raspberry Pi e também já havia realizado pesquisas sobre formas de integrar os microcontroladores NodeMCU, Arduino UNO e Raspberry Pi, referentes às etapas a) e b). Também, parcialmente, pesquisou sobre formas de controle aprimorado de motores e servomotores, e formas de otimização da alimentação de energia, referentes às etapas c) e d). Isso resultou na seguinte ordem de forma resumida:

1. Estudar e aprimorar o controle (motor e servomotor) dos projetos (parcialmente concluída);
2. Pesquisar sobre interfaces de controle de plataformas Microcontroladas;
3. Desenvolver a interface de controle dos projetos;
4. Avaliar o desempenho dos dispositivos com experimentos;
5. Estudar e aprimorar a alimentação dos projetos (parcialmente concluída);
6. Pesquisar projetos e formas de integrar NodeMCU, Arduino UNO e Raspberry Pi (documentando a pesquisa) (Concluída);
7. Estudo e execução de projetos simples na plataforma Raspberry Pi (Concluída);
8. Fazer relatório.

Analisando as possibilidades, foi escolhido o controle do braço robótico via Bluetooth devido a praticidade e maior precisão no controle. Focando na evolução do braço robótico, inicialmente foi desenvolvida uma série de microprojetos destinados a consolidar o conhecimento com a plataforma Arduino e seus componentes, evoluindo a complexidade dos projetos até alcançar o nível desejado de controle do braço robótico via Bluetooth, assim aprimorando a responsividade e precisão do controle do projeto.

Como interface de controle, foi cogitada a possibilidade de alguma linguagem de programação específica para tal tarefa, porém, essa opção demandaria muito mais tempo para o aprendizado. Sendo assim, optou-se por usar a plataforma online chamada MIT App Inventor 2 para criar o aplicativo que se conectaria via Bluetooth com o Arduino. Essa plataforma foi escolhida pela praticidade na junção entre a programação com a parte visual utilizando de blocos para a montagem de ambas.

[ALTERAR POSTERIORMENTE, POIS É ALGO QUE POSSO APRIMORAR ANTES DA CONCLUSÃO DO RELATÓRIO]

Até então, o projeto mais recentemente montado apresentou responsividade ideal, porém, ao controlar um servomotor, acaba sendo insuficiente a energia provida pelo cabo USB ao Arduino, o que comprova a necessidade do estudo sobre as diferentes formas de alimentação dos projetos.

[COMO FOI FEITO PELO ORIENTADOR E PARCIALMENTE PELA BOLSISTA ANTERIOR, NÃO SEI SE DEVO INCLUIR ESSA PARTE NO RELATÓRIO] Com a ajuda do orientador, foi feita uma pesquisa sobre diferentes tipos de pilhas e baterias, analisando suas diferenças em busca de alguma que possa representar o resultado ideal para o projeto.

1. **Introdução**

Com o advento da terceira revolução industrial, por volta de 1950, foi intensificado o uso da robótica como forma de aprimorar a eficiência e precisão dos processos fabris, tornando a linha de montagem ainda mais autônoma e mecânica. A ascensão de novas tecnologias no ramo computacional abriu portas para o que pode ser considerado a quarta revolução industrial.

[QUAL A FORMA CERTA DE CITAR UM TRECHO DO LIVRO?]

Segundo Klaus Schwab (2019, p. 1)

Imagine as possibilidades ilimitadas de bilhões de pessoas conectadas por dispositivos móveis, dando origem a um poder de processamento, recursos de armazenamento e acesso ao conhecimento sem precedentes. Ou imagine a assombrosa profusão de novidades tecnológicas [...]: inteligência artificial (IA), robótica, a internet das coisas (IoT na sigla em inglês), [...] armazenamento de energia e computação quântica, para citar apenas algumas. Muitas dessas inovações estão apenas no início, mas já estão chegando a um ponto de inflexão de seu desenvolvimento, pois elas constroem e amplificam umas às outras, fundindo as tecnologias dos mundos físico, digital e biológico.

A utilização dessas novas tecnologias emergentes no cotidiano de cada um de nós, bem como automação industrial e até mesmo na medicina, tem um potencial exponencial em relação a sua eficiência e maleabilidade, facilitada pela intensa digitalização de informações (Big Data) e a crescente interconexão de objetos que trocam informações entre si (IoT).

Essa nova realidade tecnológica está se tornando cada vez mais presente, como por exemplo, com o uso da tecnologia Bluetooth para enviar informações e conectar dispositivos, que os torne muito mais acessíveis. Porém, essa tecnologia não se limita a isto.

Nas indústrias, com o seu uso mais consolidado, e nas salas de cirurgia, é possível encontrar braços robóticos atuando de forma eficiente e precisa nas tarefas que lhes são conferidas, mesmo depois de dias de trabalho consecutivo, mantém a mesma qualidade de serviço pois, diferentes dos braços humanos, os robóticas não se cansam, não precisam dormir e nem se alimentar.

Ao juntarmos as duas tecnologias, podemos efetuar um controle preciso e eficiente de um braço robótico sem a necessidade de estar conectado fisicamente ao mesmo. Isso torna ainda mais maleável e prático a execução de diversas tarefas mais delicadas e específicas, como as cirurgias, diferentemente das indústrias que podem simplesmente ser programadas para sempre repetir determinados movimentos diversas vezes.

1. **Objetivos**

Este trabalho tem como objetivo dar continuidade ao projeto anterior, primeiramente com diversos microprojetos com a intenção de aprender na prática sobre os recursos e conceitos necessários para o aprimoramento dos protótipos já existentes. Também tem como objetivo a criação de interfaces virtuais para o controle dos protótipos microcontrolados que visam a Internet das Coisas (IoT, sigla em inglês), sendo eles um braço robótico e um sistema autônomo de monitoramento e irrigação, assim como aprimorar a eficiência e responsividade dos mesmos. Tendo como foco inicial a melhora do controle do braço robótico, para posteriormente visar o monitoramento e irrigação.

[ADICIONAR MAIS ALGUMA COISA?]

1. **Metodologia**

O método utilizado foi o desenvolvimento de vários microprojetos simples, porém, gradativamente mais complexos com o intuito de alcançar a complexidade que o controle de 4 servomotores utilizando a conexão Bluetooth e o microcontrolador Arduino UNO. Esse método assemelha-se à técnica de desenvolvimento de software chamada “Test Driven Development” (TDD) ou em português” Desenvolvimento Guiado por Testes”, que consiste em As etapas do desenvolvimento dos projetos serão descritas logo abaixo.

Inicialmente foram utilizadas apenas os componentes básicos do Arduino e posteriormente, foi introduzido um novo componente adquirido pelo orientador: O Módulo Bluetooth RS232 HC-05. O que aumentou consideravelmente o nível de complexidade nos projetos, visto que não há tanta documentação oficial sobre a programação necessária para projetos envolvendo esse componente.

Também foi necessário o uso de um aparelho de telefonia móvel capaz de efetuar conexão Bluetooth e que possa instalar aplicativos externos, preferencialmente via Código QR. Este aplicativo de controle Bluetooth foi desenvolvido através da plataforma MIT App Inventor 2, com o objetivo de parear os dispositivos Bluetooth e controlar os componentes conectados.

Como forma de comunicação remota com o Arduino, foi utilizado um aplicativo genérico encontrado na Google Play que enviava informações via Bluetooth, porém, ele não apresentava a funcionalidade necessária visada, que é a de conseguir controlar simultaneamente 4 diferentes motores. Como seria muito pouco provável que existisse um aplicativo já feito que atenda a esses critérios, optou-se por desenvolver um app (forma simplificada de “aplicativo” em inglês) para tal fim.

Um dos meios de se desenvolver um app é através de linguagens de programação voltadas para esse propósito, tais como Java, Kotlin e C++. Outra forma de desenvolver apps é com utilização de plataformas criadas para desenvolver apps sem a necessidade de aprender uma linguagem de programação específica. A plataforma escolhida é MIT App Inventor 2, que simplifica a criação do app, pois utiliza um sistema de criação similar a montagem de blocos, tanto para o design da parte visual (em computação, usa-se o termo “front-end”), quanto para a lógica de programação que acontece “por detrás das cortinas” (o que em computação é chamado de “back-end”).

A vantagem deste segundo método é a praticidade de criar rapidamente um app, sem a necessidade de aprender toda uma linguagem de programação para esse fim. A desvantagem, porém, é a limitação de desenvolvimento, sendo restrito apenas as funcionalidades pré-existentes na plataforma. Porém, como a necessidade do projeto se enquadra no que está disponível pela plataforma, então acabou sendo o caminho mais viável a se seguir.

Em seguida, estão listados os componentes

utilizados nos experimentos, explicados de forma básica (Imagens fora de escala e meramente ilustrativas):

* 1. **Componentes**
     1. **Arduino Uno**

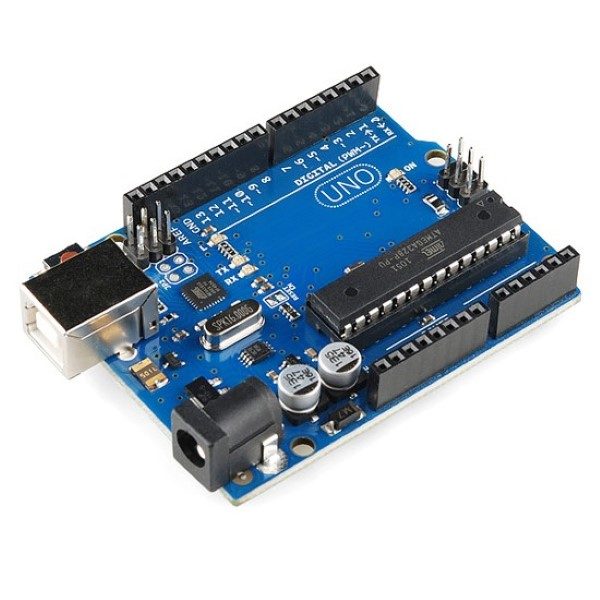


Figura 1: Arduino UNO - Fonte [1]

Minicomputador (microcontrolador) que permite programar em uma linguagem baseada em C os valores de energia recebidos e enviados pelas portas de entrada e saída presentes na placa.

* + 1. **Protoboard**

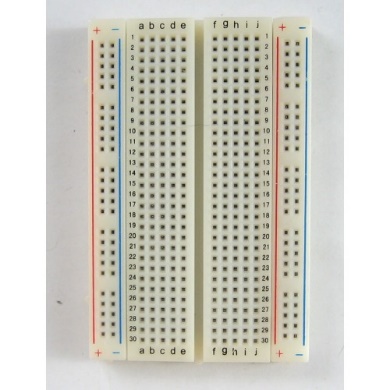


Figura 3: Protoboard - Fonte [3]

A protoboard, também chamada de Breadboard é uma placa utilizada para fazer o contato direto entre os componentes sem a necessidade de soldagem. Pois seus orifícios estão conectados interntamente e permitem a conexão e desconexão dos jumpers sem dano.

* + 1. **Cabo USB**



Figura 2: Cabo USB - Fonte [2]

Cabo USB tipo A-B. Serve para transferir a programação do computador para o Arduino, e também para alimentá-lo com energia.

* + 1. **Jumpers**



Figura 4: Jumpers - Fonte [4]

Cabos condutores maleáveis e com ponta sólida, próprios para o uso nas protoboards e nos componentes Arduino. Servem para fazer as conexões elétricas no circuito.

* + 1. **Botão**



Figura 5: Botões - Fonte [5]

Botão, também chamado de switch (em inglês) é um componente que mantém aberto (sem passagem de eletricidade) o circuito em que está conectado, fechando-o quando pressionado.

* + 1. **Resistores**

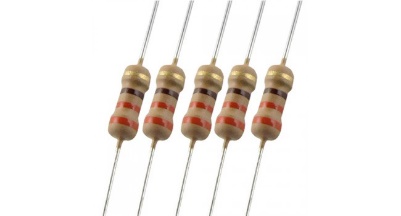


Figura 6: Resistores - Fonte [6]

Servem para regular a corrente elétrica seguindo a fórmula “V = R \* I” (Tensão = Resistência \* Corrente), transformando-a em energia térmica. Existem diversos tipos de resistores, tendo eles diferentes valores de resistividade, esses valores são ilustrados através da sequência de cores.

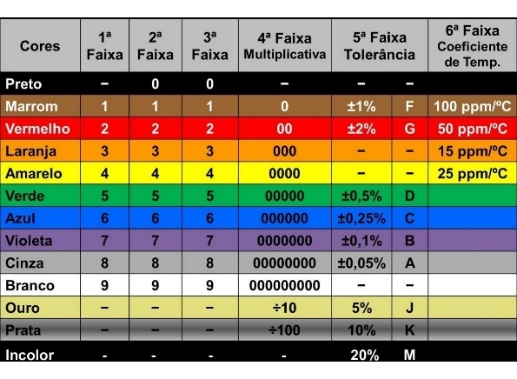


Figura 7: Código Colorido Resistores - Fonte [7]

* + 1. **LEDs**



Figura 8: LEDs - Fonte [8]

LED (*Light-Emitting Diode* – diodo emissor de luz) é um componente utilizado para converter energia elétrica em energia luminosa. Pode apresentar diferentes luminosidades com o uso devido das portas PWM (*Pulse Width Modulation* – modulação por largura de pulso*)* e da programação no Arduino.

* + 1. **Potenciômetros**



Figura 9: Potenciômetro - Fonte [9]

São resistores que variam a sua resistência à medida em que o seu regulador variável é girado.

* + 1. **Joystick**



Figura 10: Joystick - Fonte [10]

São dois potenciômetros acoplados a uma peça plástica de tal forma que representam as coordenadas X e Y em que essa peça está localizada.

* + 1. **Micro Servomotor 9g**



Figura 11: Micro Servomotor - Fonte [11]

É um motor que funciona rotacionando até a angulação que é enviada pelo sinal de controle. O servomotor 9g tem capacidade de rotação de até 180°.

* + 1. **Módulo Bluetooth HC-05**

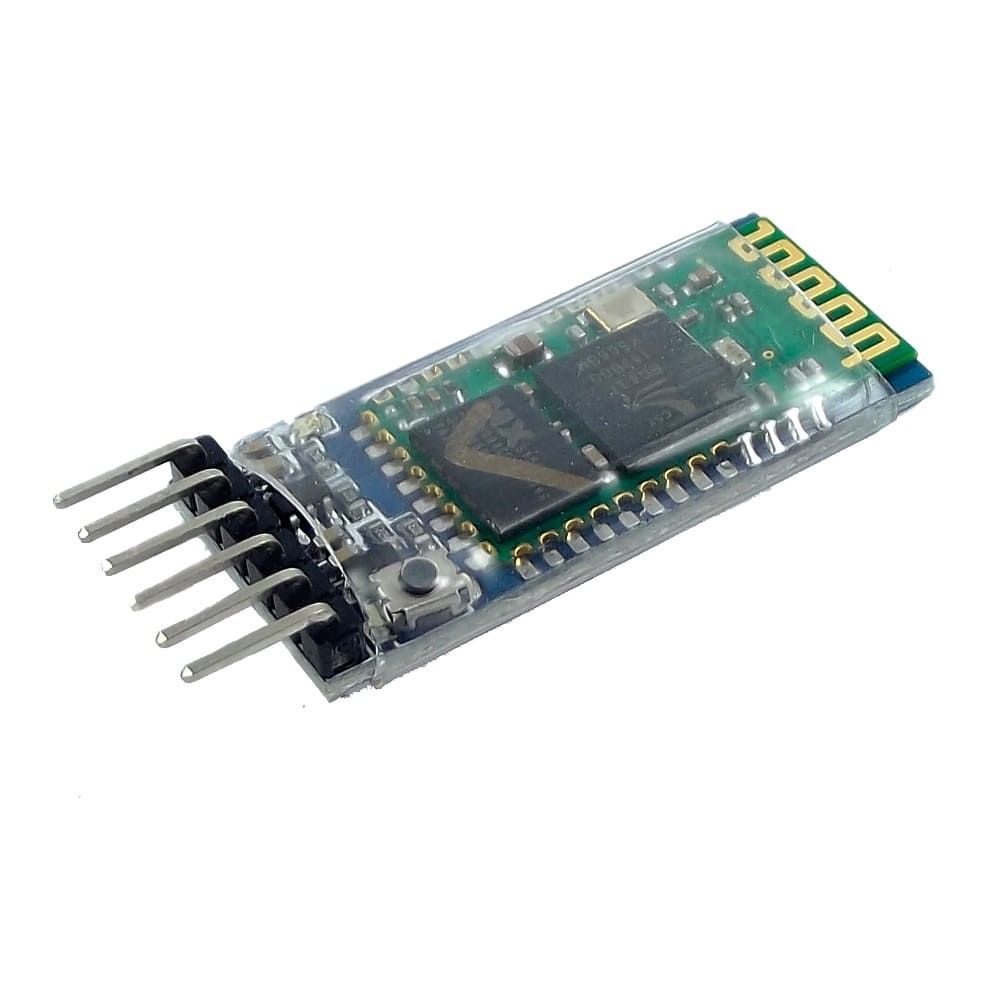


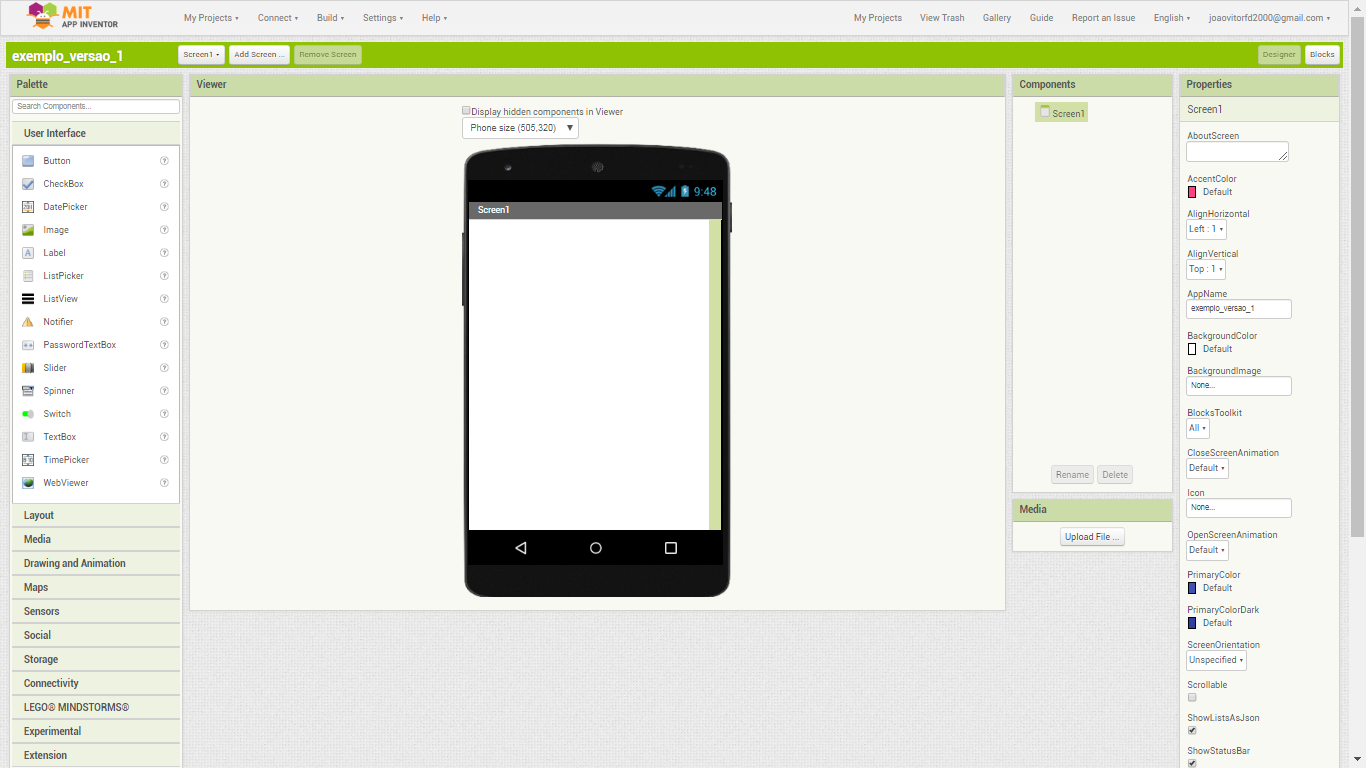
Figura 12: HC-05 - Fonte [12]

Dispositivo capaz de enviar e receber informações através da tecnologia Bluetooth, sendo capaz de parear com o dispositivo em questão e efetuar as trocas de informações.

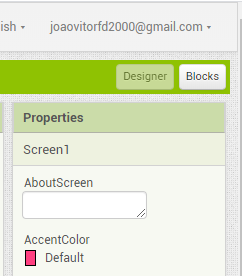
* 1. **MIT App Inventor 2**

Como dito no site oficial da plataforma, o MIT App Inventor 2 é um ambiente de programação visual que, de forma intuitiva, proporciona fácil aprendizagem e rápido desenvolvimento de aplicativos para smartphones e tablets. Utilizando de blocos para a programação e configuração da aparência do aplicativo, o App Inventor torna bem prática e acessível a criação de aplicativos, até mesmo para crianças, transformando os consumidores de tecnologia em criadores de tecnologia. A programação baseada em blocos fomenta a criatividade, permitindo que novas ideias aflorem e tenham impacto social através dos aplicativos desenvolvidos.

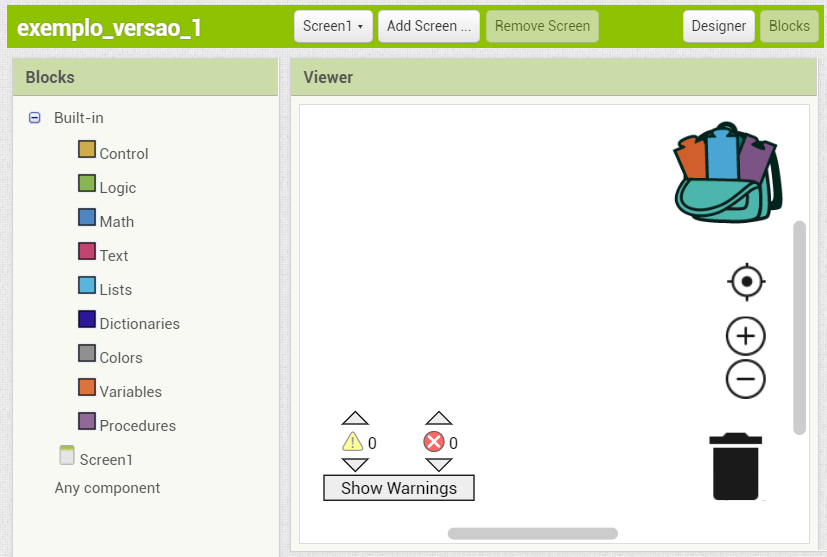
A página inicial de desenvolvimento do app nessa plataforma é a página de desenvolvimento visual (Designer).



No canto superior direito é possível alternar entre a página Designer (Front-End) e a página Blocks (Back-End).



Ao clicar em “Blocks”, seremos encaminhados para a página “Blocks”:



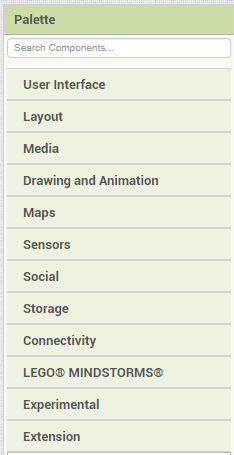
No App Inventor, tanto a parte de desenvolvimento visual (Front-End) quanto a parte de desenvolvimento de retaguarda (Back-End) são montadas utilizando o sistema de “blocos arrasta-e-solta”, ou seja, para utilizar algum bloco de design ou programação no aplicativo, basta arrastá-lo respectivamente, das abas Palette e Blocks para a aba Viewer.

* + 1. **Componentes Visuais (Front-End)**

Na página Designer existem 4 abas de maior importância, sendo elas: Palette, Viewer, Components e Properties, que serão explicadas abaixo.

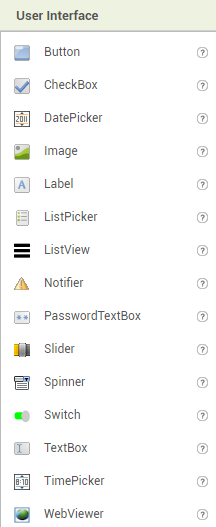
**Palette:**

É uma palheta onde estão todos os blocos de funcionalidades do App Inventor. Divididos em diversas categorias como User Interface (Interface do usuário), Layout, Connectivity (Conectividade), dentre outras.

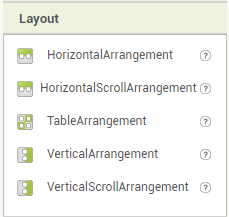


Cada uma dessas categorias contém vários componentes a elas relacionados. Como exemplos temos:

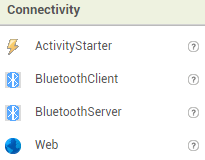
Em User Interface, encontram-se diversas formas de interação direta entre o app e o usuário, sendo alguns deles o Botão, o Slider e a Caixa de Texto;



Em Layout, estão componentes utilizados para organizar a tela como tabelas, arranjos horizontais e verticais;

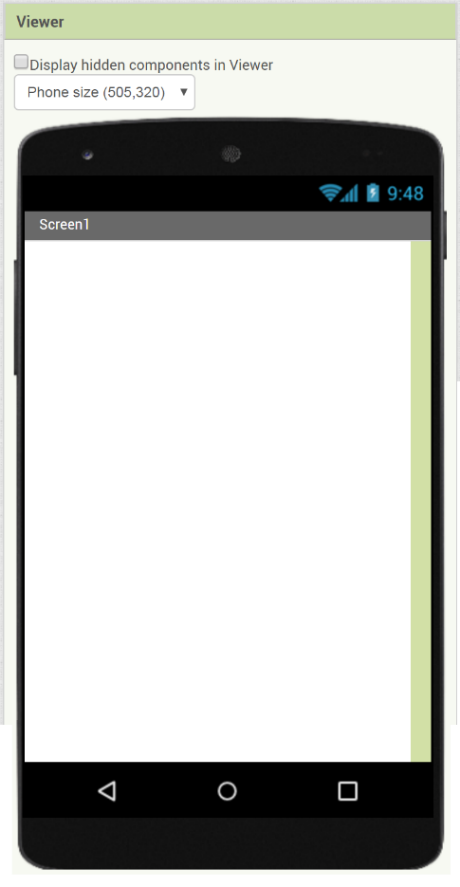


Em Conectividade, estão algumas funcionalidades não visíveis ao usuário, mas que permitem, por exemplo, que o aplicativo tenha acesso a conexão Bluetooth.



**Viewer:**

Apresenta uma visão geral de como ficará o aplicativo visualmente quando concluído.



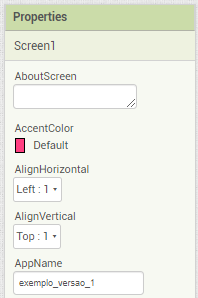
**Components:**

Aqui estão listados todos os componentes visuais utilizados no desenvolvimento do aplicativo. Como todo projeto inicia-se apenas com o componente “Screen1”, neste caso, apenas ele está sendo mostrado, porém, a medida em que novos componentes forem sendo adicionados ao projeto, eles serão, automaticamente, listados nessa aba.



**Properties:**

São mostradas as propriedades dos componentes. Cada componente tem sua respectiva gama de características. Na imagem demonstrativa, são mostradas algumas das propriedades do componente Screen1.

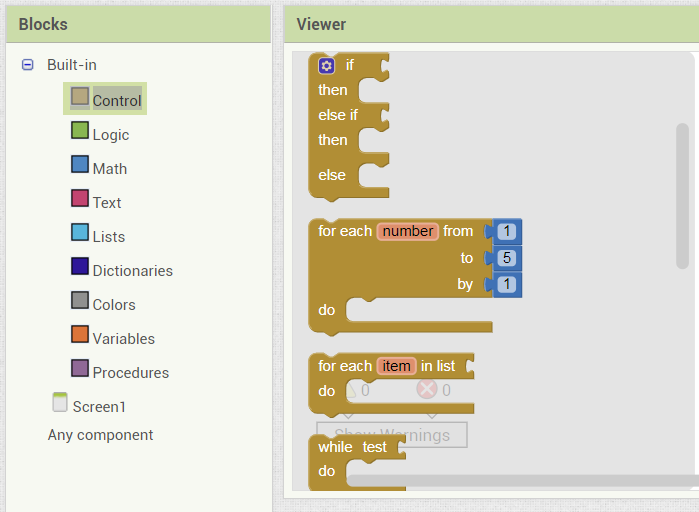


* + 1. **Blocos de Programação (Back-End)**

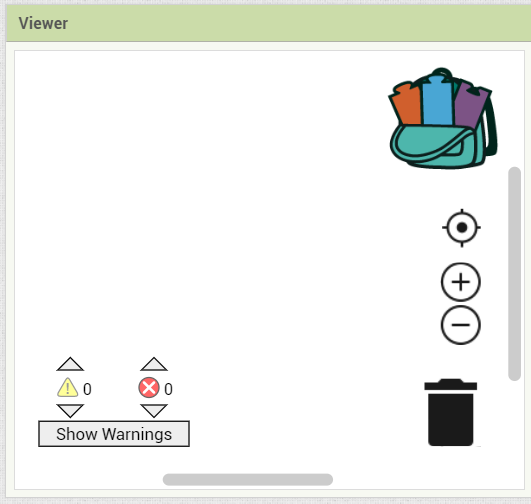
Na página Blocks existem 2 abas, sendo elas: Blocks e Viewer, que serão explicadas abaixo.

**Blocks**: É onde estão organizados diversos componentes divididos por categorias, dentre eles: Control (Controle de fluxo), Math (Matemática) e Variables (Variáveis).

Como exemplo de componente, temos os blocos de controle de fluxo com a cor amarela que poderão ser utilizados nos aplicativos que serão desenvolvidos.



**Viewer:** Diferentemente da aba Viewer da página Designer, este Viewer funciona como uma área de trabalho. Nela, serão organizados diversos blocos de comando que funcionarão em conjunto para resultar num aplicativo funcional.



Nele temos as seguintes funcionalidades:

Os avisos de erros que aparecem quando alguma coisa não está correta no desenvolvimento;

A mochila que funciona como uma forma de guardar grupos de blocos que serão frequentemente utilizados com o objetivo de facilitar a reutilização;

O centralizador do Viewer, que permite voltar o zoom para o centro do desenvolvimento;

Aumentar e reduzir o zoom nos blocos de comando;

A lixeira para descartar os blocos desnecessários.

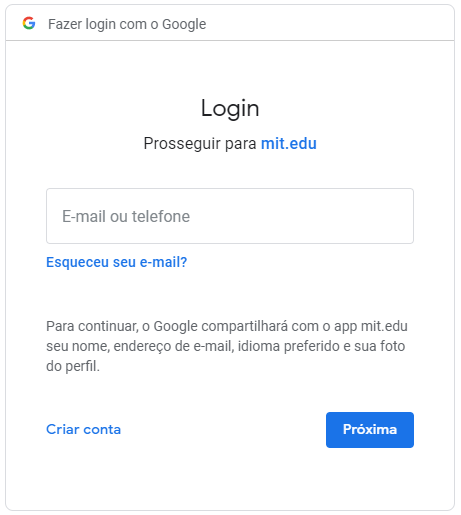
* + 1. **Criando um projeto**

O projeto de exemplo demonstrado faz parte de alguns aplicativos desenvolvidos para os microprojetos.

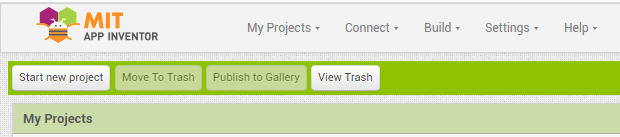
Na página principal do App Inventor (<http://appinventor.mit.edu/>) clique em “Create Apps!”



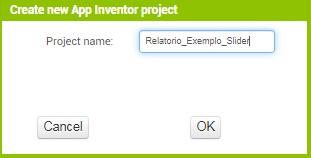
Faça Login com sua conta do Google



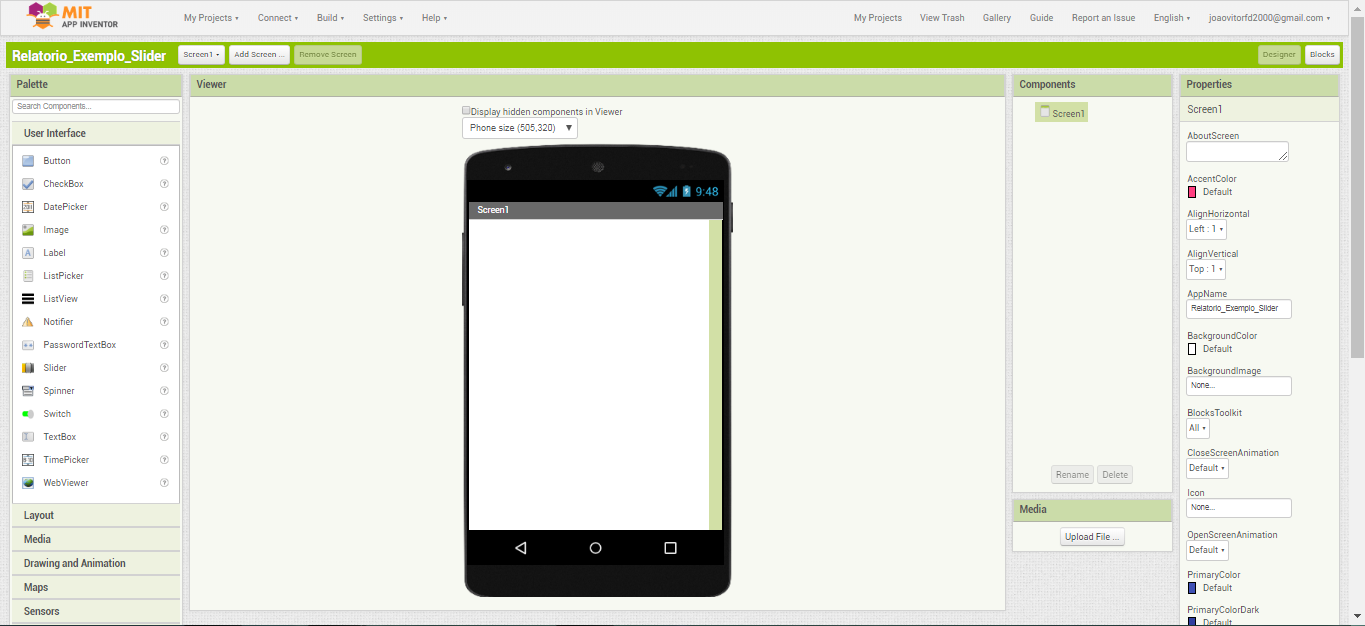
Clique em “Start new project”



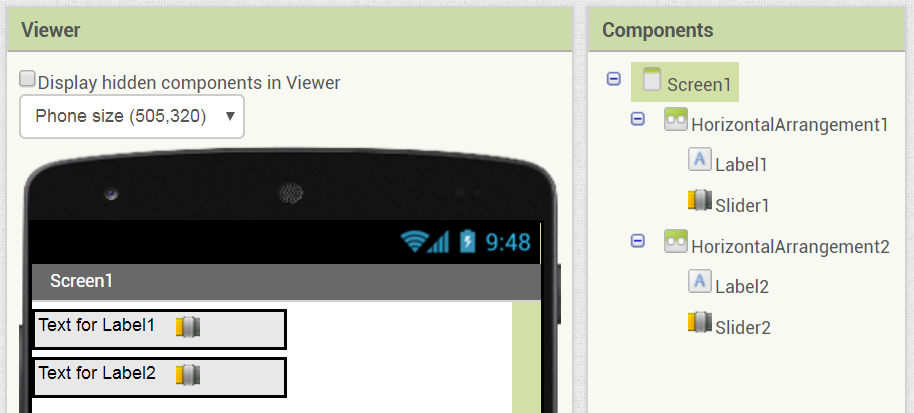
Escolha um nome para o projeto e clique “OK”



Logo em seguida, você será redirecionado para a página de desenvolvimento visual (designer).



Neste exemplo na página designer, foram arrastados os componentes HorizontalArrangement1 (arranjo horizontal), Label1, Label2 (caixas de texto), Slider1 e Slider2 (botão deslizante).



Para aprimorarmos o visual inicial do app é necessário fazer algumas mudanças nas propriedades de alguns componentes, como por exemplo:

Screen1: Alinhar no centro horizontal, tornar o título invisível.

HorizontalArrangement1: Alinhar no centro vertical, tornar a largura igual a todo o espaço disponível.

Slider1 e Slider2: Selecionar cores de preferência para o lado esquerdo e direito, tornar a largura igual a todo o espaço disponível, definir o valor máximo (nesse caso, será utilizado como exemplo o valor máximo de brilho de um LED, ou seja, 255), definir o valor mínimo (nesse caso, será utilizado como exemplo o valor mínimo de brilho de um LED (quando ele está desligado), ou seja, 0), definir o valor inicial do botão variável como o centro (126) (para melhor visibilidade).

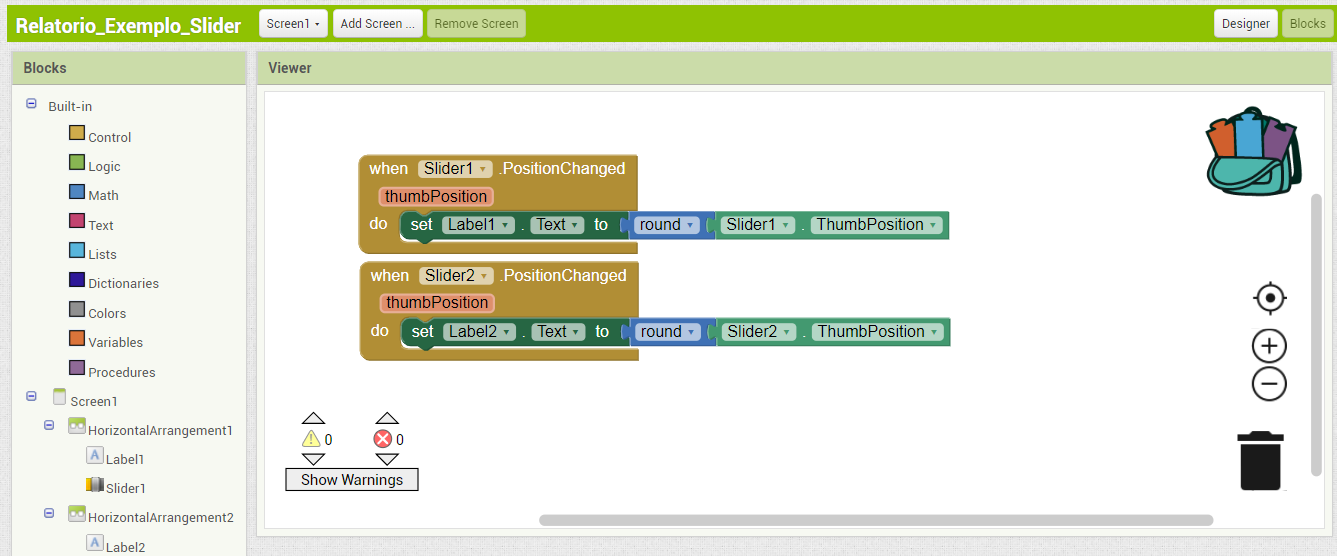
Label1 e Label2: Tornar texto em negrito, definir o texto como o valor inicial do Slider1 (126), definir a largura suficiente para que caiba o texto (30 pixels), alinhar o texto ao centro.

Assim obtemos o seguinte resultado:



Obs.: Embora não seja visível no Viewer do App Inventor, quando o app é instalado, ambos os Sliders preenchem todo o espaço disponível.

Na página Blocks, foram arrastados estes blocos de comando:



Eles representam as seguintes funcionalidades:

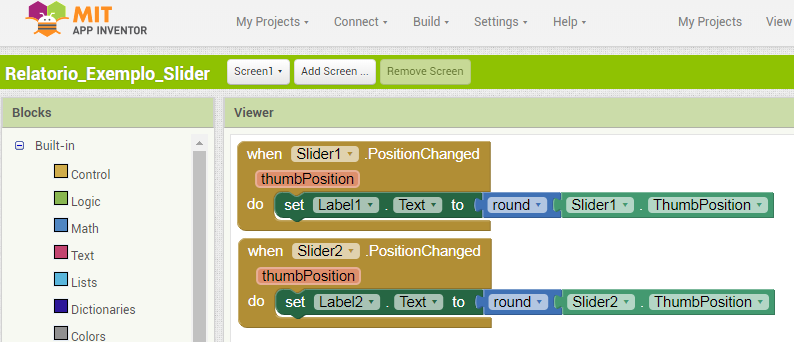


Quando a posição do botão deslizante do Slider1 for alterado, definir o texto do Label1 como sendo o número da posição atual do Slider1 como número inteiro.

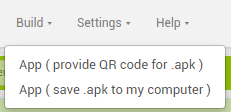


Quando a posição do botão deslizante do Slider2 for alterado, definir o texto do Label2 como sendo o número da posição atual do Slider2 como número inteiro.

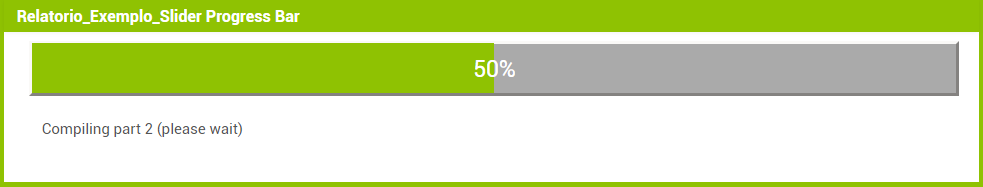
Para concluir o projeto e torná-lo um aplicativo de fato, basta clicar em Build no topo da página e escolher o formato de preferência.

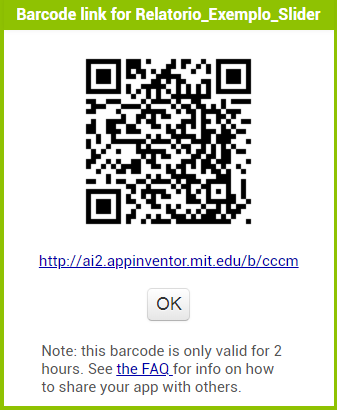


Caso prefira instalar o app através do QR, selecione a primeira opção.



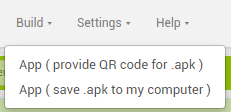
Espere um momento para que a plataforma compile o app.



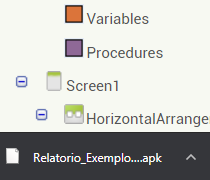


Assim que for compilado, aparecerá na tela um QRCode, a partir daí, basta ler o código QR com um App ou acessar o link diretamente no navegador do seu Smartphone.

Caso opte pela instalação por .apk, selecione a segunda opção.



Um arquivo tendo o nome do seu projeto será baixado, a partir daí, basta copiar este arquivo para o teu Smartphone e utilizar algum app de gerenciamento de arquivos para executar este arquivo e efetuar a instalação. Em alguns smartphones, a instalação de aplicativos por fontes externas a loja principal é bloqueada. Nesses casos, essa opção deve ser liberada nas configurações do celular.



Assim conclui-se um aplicativo de variação de slider, utilizando a plataforma App Inventor e que será frequentemente utilizado nos microprojetos posteriores.

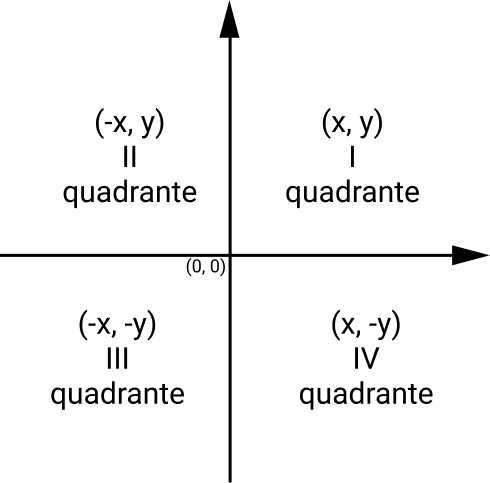
* 1. **Microprojetos**

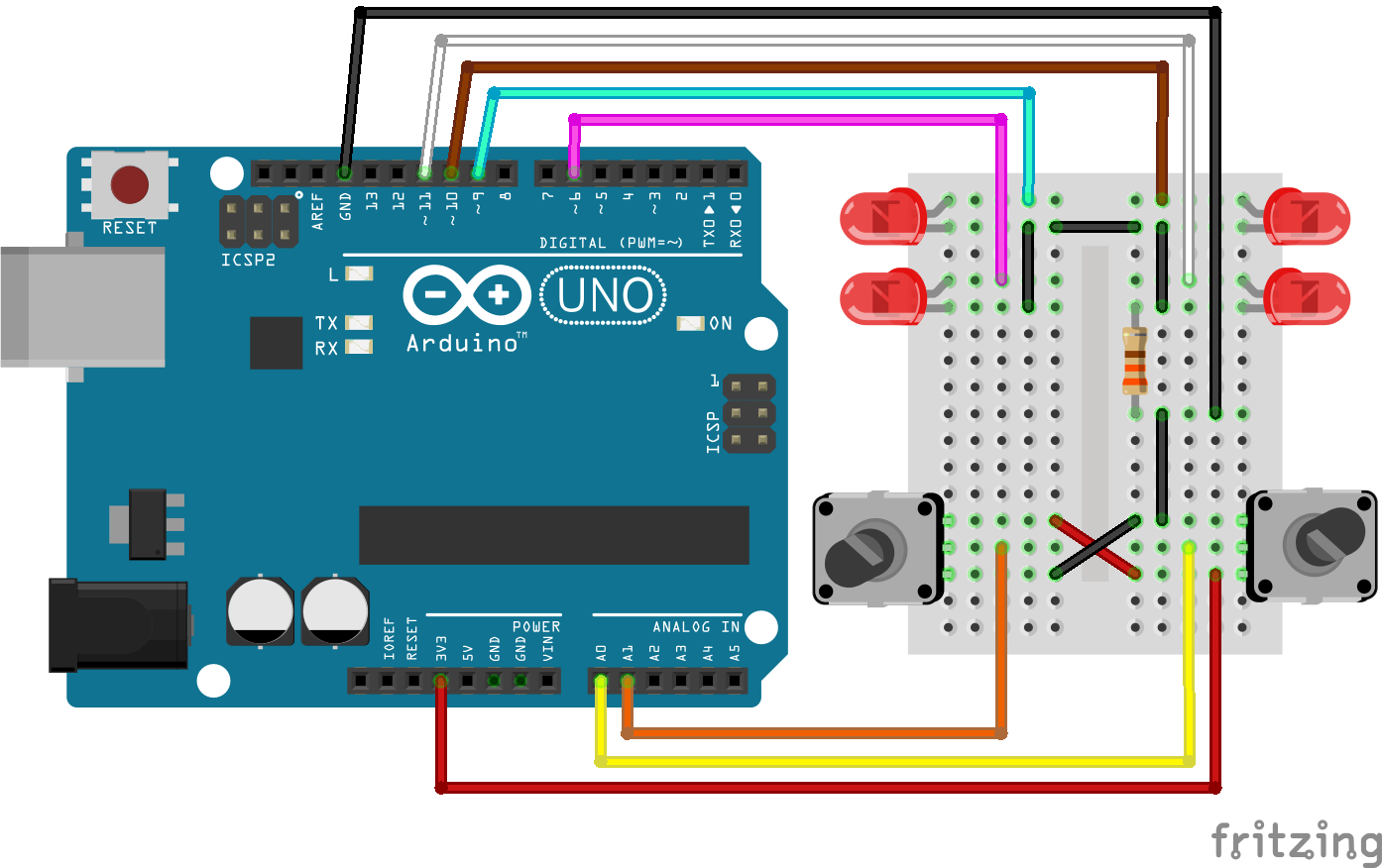
Como comentado anteriormente, foi adotada uma tática de desenvolvimento do projeto de aprimoramento do braço robótico através de um aumento constante no desenvolvimento de microprojetos que, de certa forma, é um único projeto que foi sofrendo metamorfose.

Para tornar o relatório mais conciso, alguns outros microprojetos mais simples do que os que serão citados abaixo foram ocultados do relatório, pois foram simples demais ou apresentaram muita repetição com o que já havia sido feito anteriormente.

* + 1. **Ligar os quatro LEDs de acordo com a posição de dois potenciômetros representando as coordenadas X e Y**

Como o braço robótico recebe quatro valores digitais (que variam entre 0 e 179) que representam a sua angulação foi escolhido utilizar primeiramente dois potenciômetros (que variam entre 0 e 1023) para controlar quatro LEDs (que variam entre 0 e 255) de acordo com a saída dos potenciômetros.

Utilizando o sistema dos quadrantes num plano cartesiano, um potenciômetro representará o eixo x e outro representará o eixo y, criando o ponto cartesiano (x, y). Quando o valor de ambos for 512, ou seja, o ponto (512, 512), funcionaria como sendo o centro (0,0), logo, seria como se valores maiores que 512 fossem positivos e os valores menores que 512 fossem negativos. Assim, quando os dois potenciômetros tiverem valores acima de 512, ou seja (+, +), apenas o LED que representa o primeiro quadrante (LED 1) ficará ligado. Quando o potenciômetro que representa o eixo x for “negativo” (como dito, menor que 512), e o eixo y for “positivo” (maior que 512), ou seja (-, +), apenas o LED do segundo quadrante (LED 2) ficará ligado, e assim para cada um dos quadrantes.

**Diagrama**

**Programação**

Definição de variáveis. As variáveis led1, led2, led3, led4, X e Y serão apenas utilizadas para a definição de seus pinos, não serão alteradas, por isso são chamadas de constantes. Diferentemente das variáveis EntradaX e EntradaY que receberão diversos valores durante a execução do programa.

const int led1 = 10, led2 = 9, led3 = 6;  
const int led4 = 11, X = A0, Y = A1;  
int EntradaX, EntradaY;

No procedimento “setup” são definidos todos os LEDs como pinos de saída. São criados dois procedimentos. Quando o procedimento “liga” for chamado, será enviado o valor do pino de um LED para que o mesmo seja ligado, o oposto ocorre quando o procedimento “desliga” for chamado, nesse caso, o LED que teve o valor de seu pino enviado, será apagado.

void setup () {  
 pinMode(led1, OUTPUT);  
 pinMode(led2, OUTPUT);  
 pinMode(led3, OUTPUT);  
 pinMode(led4, OUTPUT);  
}

void liga (int led) {digitalWrite (led, HIGH);}

void desliga (int led) {digitalWrite (led, LOW);}

Tudo o que for digitado no procedimento “loop” será continuamente repetido. O valor da posição dos potenciômetros que representam os eixos X e Y são lidos e guardados, respectivamente, nas variáveis “EntradaX” e “EntradaY”. Em seguida, esses valores serão comparados para distinguir qual dos quadrantes está sendo representado, assim, caso o quadrante 1 esteja sendo representado, ou seja, os valores de “EntradaX” e “EntradaY” são ambos maiores ou iguais a 512, apenas o LED 1 será ligado, enquanto os LEDs 2, 3 e 4 serão desligados. Caso seja representado o quadrante 2, “EntradaX” menor que 512 e “EntradaY” maior que 512, apenas o LED 2 será ligado, enquanto os LEDs 1, 3 e 4 serão desligados, a mesma lógica acontece para os quadrantes 3 e 4.

void loop () {

EntradaX = analogRead(X);

EntradaY = analogRead(Y);

if (EntradaY >= 512) {

desliga(led3);

desliga(led4);

if (EntradaX >= 512) {

liga(led1);

desliga(led2);

}

else {

desliga(led1);

liga(led2);

}

}

else {

desliga(led1);

desliga(led2);

if (EntradaX >= 512) {

desliga(led3);

liga(led4);

}

else {

liga(led3);

desliga(led4);

}

}

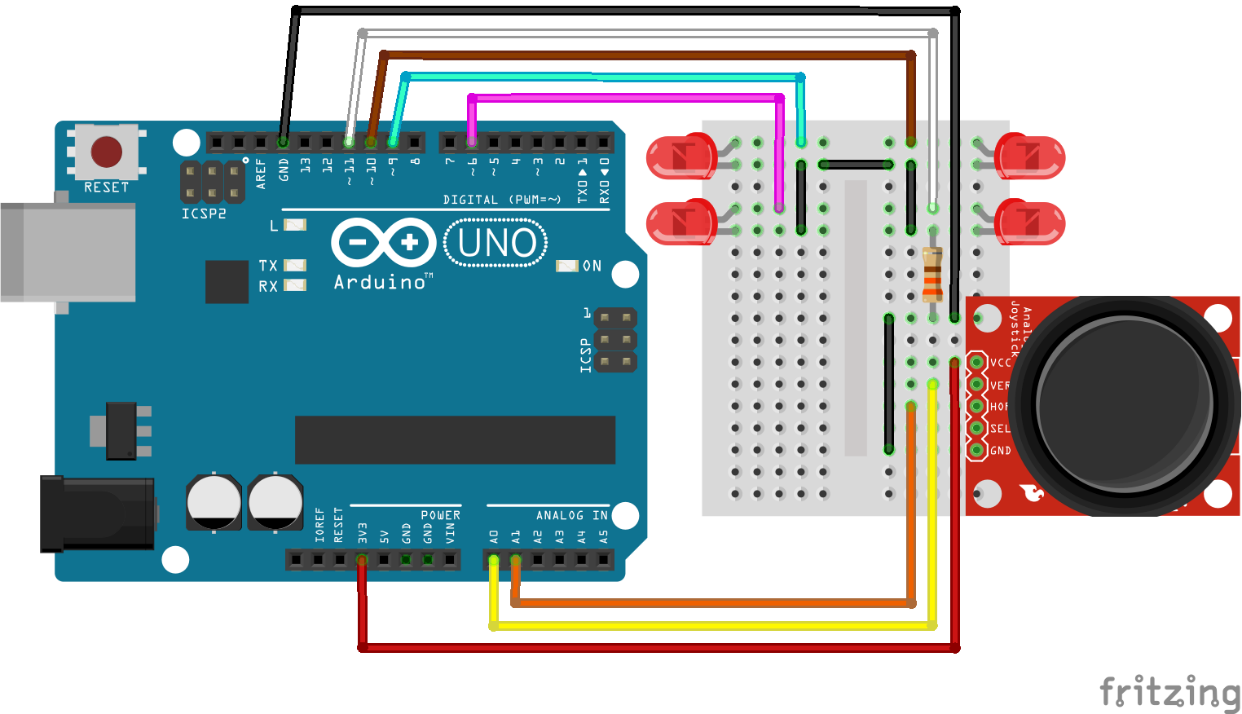
**Potenciômetros substituídos por um joystick**

Como o braço robótico foi construído anteriormente utilizando joysticks como controles, foram substituídos os dois potenciômetros por um único joystick, porém mantendo objetivos similares. Então, quando ele for posicionado em qualquer posição próxima ao canto superior direito, retornaria os valores maiores que o centro, logo (+,+), então ele acenderia o LED 1, próximo ao canto superior esquerdo, retornaria valores (-,+), então acenderia o LED 2, e assim por diante. De forma similar, quando o joystick estiver solto (centralizado), nenhum LED se acenderia.

Porém, um problema encontrado através do Monitor Serial indicava que, mesmo que ele estivesse solto, ele não retornava o valor central esperado, que seria (512,512). Imaginou-se que o problema estaria naquele joystick em específico, então todo a montagem foi mantida, mas trocou-se o joystick por um outro disponível. A variação permaneceu, porém não era igual a anterior. Ambos os joysticks variavam sem nenhum padrão no valor da posição central nos dois eixos.

Como forma de amenizar o problema, o centro não mais passou a ser o ponto (512,512), mas sim a posição inicialmente retornada pelo joystick solto, independente de qual fosse. E as comparações posteriores foram baseados nesse novo centro. Então, para acender o LED 3, anteriormente seriam necessários que ambos os valores fossem “negativos” (ambos fossem menores que 512), porém, nesse caso, ambos deverão ser inferiores ao valor inicialmente lido pelo joystick em sua posição solta em seus dois eixos.

**Diagrama**



**Programação**

Este bloco de definição das variáveis permanece essencialmente o mesmo que o anterior, entretanto, são adicionadas as variáveis “MeioX” e “MeioY” que representarão o valor inicial do Joystick, que será representado como sendo o centro dos quadrantes.

const int led1 = 10, led2 = 9, led3 = 6;

const int led4 = 11, X = A1, Y = A0;

int EntradaX, EntradaY, MeioX, MeioY;

Este bloco de programação tem as mesmas funcionalidades que o bloco do projeto com o potenciômetro respectivo, porém tendo como única mudança a adição da definição da primeira leitura da posição X e Y do Joystick como sendo o centro dos quadrantes.

void setup () {

pinMode (led1, OUTPUT);

pinMode (led2, OUTPUT);

pinMode (led3, OUTPUT);

pinMode (led4, OUTPUT);

MeioX = analogRead(X);

MeioY = analogRead(Y);

}

void liga (int led) {

digitalWrite (led, HIGH);}

void desliga (int led) {

digitalWrite (led, LOW);}

void loop () {

EntradaX = analogRead(X);

EntradaY = analogRead(Y);

if (EntradaY >= MeioY) {

desliga(led3);

desliga(led4);

if (EntradaX >= MeioX) {

liga(led1);

desliga(led2);

}

else {

desliga(led1);

liga(led2);

}

}

else {

desliga(led1);

desliga(led2);

if (EntradaX >= MeioX) {

desliga(led3);

liga(led4);

}

else {

liga(led3);

desliga(led4);

}

}

}

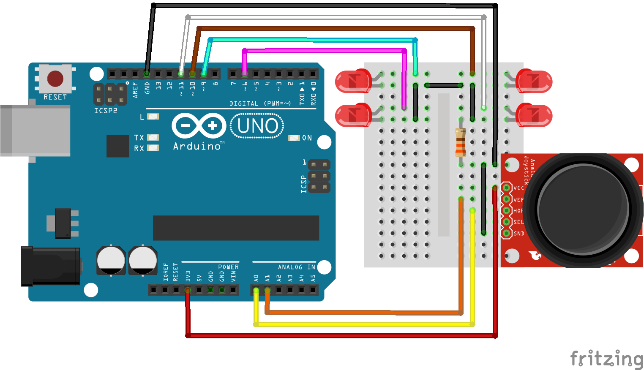
Assim como os outros dois blocos de programação, a única mudança presente é a mudança do valor 512 que representava o centro dos quadrantes pelos valores de MeioX e MeioY.

* + 1. **Variar a intensidade luminosa dos quatro LEDs de acordo com a posição do joystick em relação as coordenadas cartesianas**

Como os servomotores do braço robótico funcionam com valores analógicos e não digitais, apenas ligar e desligar LEDs não seria o suficiente. Então, o novo objetivo será variar a intensidade de cada um deles, representando assim o uso do sinal analógico nos servomotores. Seguindo a mesma lógica de que o centro funcionará como o primeiro valor do Joystick solto e que os quadrantes representariam quais LEDs seriam ligados, assim, a medida em que o joystick se aproximasse da extremidade, seja ela o eixo x ou o eixo y, o brilho do LED aceso ficaria mais intenso.

Para isso então, os valores maiores e menores que os valores deste centro serão mapeados para corresponder a um valor entre 0 e 255 que representará a luminosidade dos LEDs. No caso em que o LED 1 esteja aceso, significa que ambos os valores são positivos (+, +), porém, o valor que definirá sua intensidade é o que for maior entre eles.

**Diagrama**



**Programação**

Duas novas variáveis foram adicionadas para representar a intensidade luminosa dos LEDs de acordo com os eixos X e Y e uma outra para representar qual dos dois eixos tem maior valor.

const int led1 = 10, led2 = 9, led3 = 6;

const int led4 = 11, X = A0, Y = A1;

int EntradaX, EntradaY, MeioX, MeioY;

int MapX, MapY, MaiorMap;

Nesse bloco, apenas foi incluído o procedimento de desligar LED com o intuito de simplificar o código, podendo substituir todos os comandos “digitalWrite (‘nome\_do\_led’, LOW)”, pelo comando “desliga (‘nome\_do\_led’)”, tornando mais suscinto o código pela redução do nome da função e também da ocultação do termo “LOW”. O mesmo não ocorreria no caso da função “liga” porque seria necessário a utilização do termo responsável por sua intensidade, Sendo assim, apenas o nome da função seria alterado, o que não teria tanto impacto no tamanho do código.

void setup () {

pinMode (led1, OUTPUT);

pinMode (led2, OUTPUT);

pinMode (led3, OUTPUT);

pinMode (led4, OUTPUT);

MeioX = analogRead(X);

MeioY = analogRead(Y);

}

void desliga (int led) {

digitalWrite (led, LOW);}

[ACHO QUE ESSA EXPLICAÇÃO FICOU MUITO CONFUSA, COMO PODERIA SER MELHOR EXPLICADA?]

void loop () {

EntradaX = analogRead(X);

EntradaY = analogRead(Y);

if (EntradaX> MeioX) {

MapX = map (EntradaX, MeioX, 1023, 0, 255);

} else {

MapX = map (EntradaX, MeioX, 0, 0, 255);

}

if (EntradaY> MeioY) {

MapY = map (EntradaY, MeioY, 1023, 0, 255);

} else {

MapY = map (EntradaY, MeioY, 0, 0, 255);

}

MaiorMap = (MapX > MapY)? MapX: MapY;

if (EntradaY >= MeioY) {

desliga(led3);

desliga(led4);

if (EntradaX >= MeioX) {

digitalWrite (led1, MaiorMap);

desliga(led2);

}

else {

desliga(led1);

digitalWrite (led2, MaiorMap);

}

}

else {

desliga(led1);

desliga(led2);

if (EntradaX >= MeioX) {

desliga(led3);

digitalWrite (led4, MaiorMap);

}

else {

digitalWrite (led3, MaiorMap);

desliga(led4);

}

}

}

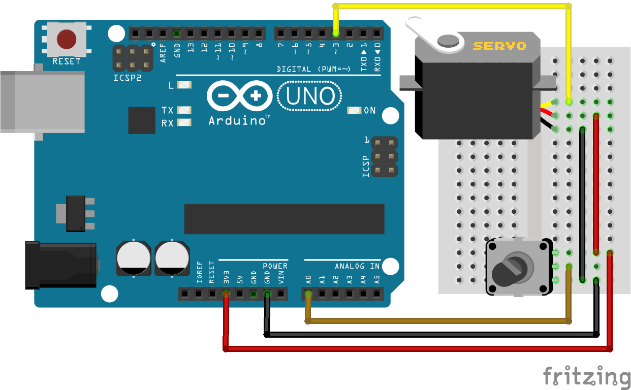
A mudança presente nesse bloco de códigos é utilizar da posição do Joystick como sendo um regulador da intensidade dos LEDs. Assim, se o valor lido pelo Joystick for maior que valor representando o centro, será feito um cálculo para que o valor lido seja convertido de um valor entre o MeioX e 1023, para um valor entre 0 e 255. O inverso ocorre caso o valor de Entrada seja inferior ao valor do meio, nesse caso, ele será inversamente proporcional ao anterior. Ele, então, será representado da seguinte forma: a partir do meio até as extremidades da esquerda e inferior, os valores que iriam entre aproximadamente 512 até 0, passarão a ir de 0 até 255. Isso ocorre tanto no eixo X quanto no Y.

Posteriormente, os valores mapeados no eixo X e Y serão comparados e o maior valor será atribuído à variável “MaiorMap”.

Assim, como foi feito nos microprojetos anteriores, é feita uma sequência de comparações que resultam no quadrante referente a posição do Joystick para ativar o LED com a intensidade igual ao valor do MaiorMap.

* + 1. **Variar a angulação do servomotor de acordo com a posição de um potenciômetro**

Embora os testes com os LEDs representem bem o que será posto em prática, algumas mudanças precisam ser feitas quando lida-se com o servomotor, como por exemplo o uso da biblioteca Servo.h para poder controlar a angulação do servomotor.

**Diagrama**

**Programação**

É adicionada a biblioteca Servo.h que é necessária para a utilização do servomotor. É criado um objeto servo1 e uma variável constante referente ao pino de entrada do potenciômetro e uma outra relativa ao valor que será recebido pela posição do potenciômetro.

#include <Servo.h>

Servo servo1;

const int Pot = A0

int = ValPot;

O objeto servo é anexado ao pino 3.

void setup () {servo1.attach (3);}

o valor do potenciômetro é lido e mapeado para ser representado entre os valores 0 e 180. Esse valor é enviado para o servo que assume essa angulação.

void loop ()

{

ValPot = analogRead (Pot);

ValPot = map (ValPot, 0, 1023, 0, 180);

servo1.write(ValPot);

delay (15);

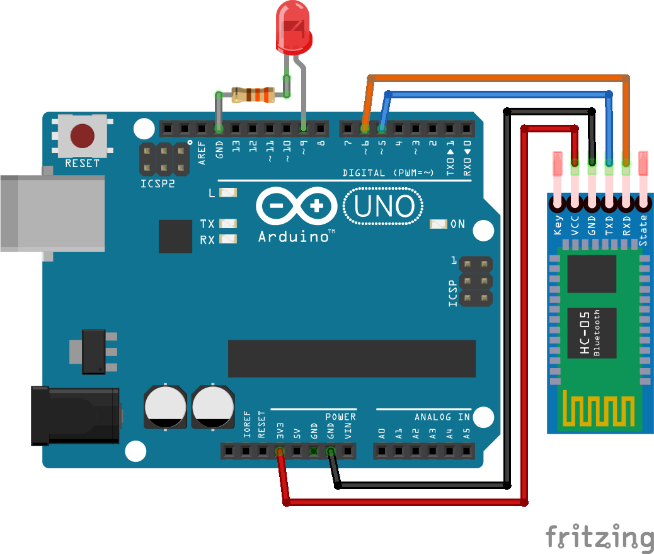
}

* + 1. **Uso do módulo Bluetooth para acender e apagar LED**

A partir daqui começa realmente o desafio com a tecnologia IoT utilizando o módulo Bluetooth. Para entender melhor o funcionamento do módulo, foi feito o seu uso para ligar e desligar um LED. Como já existem aplicativos disponíveis para tal fim, bem como aplicativos que funcionam como terminal Bluetooth para auxiliar nos testes de dispositivos, como por exemplo o aplicativo “Arduino Bluetooth Controller” encontrado na Google Play Store que permite o pareamento entre o smartphone e o módulo, assim permitindo o envio de valores entre eles.

Para acender e apagar um único LED, basta atribuir algum valor ou caractere, que quando lido, acenderá o LED, o mesmo vale caso sejam utilizados vários LEDs.

**Diagrama**



**Programação**

Inclui a biblioteca que permite definir quais entradas serão utilizadas pelo TX e RX do Módulo Bluetooth.

#include <SoftwareSerial.h>

SoftwareSerial bluetooth (5, 6);

// TX, RX do HC-05

const int led = 9;

char data;

É definida a variável referente ao pino do LED e também uma variável do tipo caractere para receber o valor que definirá se o LED ligará ou apagará.

O pino do led é definido como Output.

void setup () {pinMode (led, OUTPUT);}

Durante o bloco de repetição, é verificado se há algum valor sendo recebido pela porta serial. Caso haja, o valor será maior que zero o que torna verdadeira a condicional. Assim, o valor lido pela porta serial será atribuída a variável data, para que, de acordo com o valor recebido o LED seja ligado (caso seja recebido a letra ”a”) ou desligado (caso seja recebido a letra “b”). Se nenhuma das duas for recebida, nada será feito e esperará pelo próximo envio de informação

void loop () {

if (Serial.available() > 0)

{

data = Serial.read();

switch (data)

{

case 'a': digitalWrite (led, HIGH); break;

case 'b': digitalWrite (led, LOW); break;

default: break;

}

}

delay (40);

}

* + 1. **Uso do módulo Bluetooth para controlar a luminosidade de um LED;**

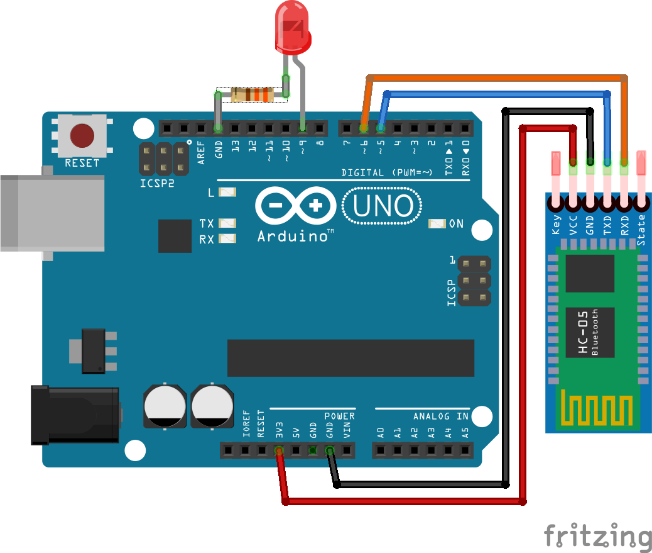
Similar ao microprojeto anterior, ainda é utilizado um aplicativo disponível na Google Play Store. Desta vez, utilizando o sinal analógico ao invés do digital, o que permite o controle da luminosidade através do envio de valores entre 0 e 255 do app para o módulo, porém o aplicativo sendo utilizado tinha a limitação de apenas enviar o valor, sem conseguir distinguir qual LED deveria ter sua luminosidade alterada.

Por causa dessa limitação foi desenvolvido um aplicativo através da plataforma online, que pareava o módulo Bluetooth com o Smartphone. A ideia inicial era mudar a intensidade do brilho de quatro LED de acordo com a posição de quatro Sliders no aplicativo. Assim, sempre que uma mínima mudança fosse feita, esse valor seria enviado para o Arduino que alteraria o brilho. Porém, para simplificar e evitar problemas extras, foram utilizados botões que enviariam quatro valores (0, 64, 128, 255), para cada um dos quatro LEDs, porém, testado em apenas um.

Logo foi encontrado outro problema: Sempre que a luminosidade do LED era alterada muito rapidamente, os valores recebidos pelo módulo e mostrados pelo Monitor Serial, que deveriam ser entre 0 e 255, estavam virando, sem nenhuma razão aparente, números enormes ou até mesmo números negativos, o que os tornavam inutilizáveis para essa situação.

Até certo ponto, o aplicativo estava funcionando de maneira devida: Já havia sido alcançado o objetivo de alterar a luminosidade do LED, mas a responsividade não estava de acordo com o esperado.

**Diagrama**



**Programação**

Diferentemente do microprojeto anterior, a informação que será enviada neste será mais longa, por isso utilizar apenas um caractere não seria viável. Logo, é utilizado uma variável do tipo String que receberá cada um dos caracteres enviados via bluetooth, e também uma variável do tipo inteiro chamada brilho, representando a intensidade luminosa do LED.

#include <SoftwareSerial.h>

SoftwareSerial bluetooth (5, 6);

const int led = 9;

String stringGeral;

int brilho;

É definido o pino do LED como saída (Output).

void setup () {pinMode (led, OUTPUT);}

Se houver algum valor sendo recebido, executa o código interno. Ao ser executado, a String “stringGeral” é “zerada”.

void loop () {

if (Serial.available()) {

stringGeral = String ("");

while (Serial.available()) {

stringGeral = stringGeral + char (Serial.read());

delay (1);

}

brilho = stringGeral.toInt();

if (brilho >= 0 && brilho <= 255) {

analogWrite (led, brilho);}

}

}

Enquanto algum valor ainda estiver sendo recebido, a “stringGeral” terá como novo valor tudo o que já havia sido escrito nela mesma, acrescido do valor lido pela porta serial que recebeu casting (forma de transformar valores de um tipo de variável em outro) para se tornar char.

Após todos os valores terem sido recebidos, é utilizado uma função para converter toda a variável String para uma variável inteira, assim, se o valor estiver entre 0 e 255, o LED acenderá com o valor recebido.

* + 1. **T****estes de passagem de informação através do módulo Bluetooth e a forma de recepção desses dados pelo Arduino (App Inventor).**

Por causa do bug encontrado no microprojeto anterior, optou-se por desenvolver um programa e app especificamente para aprender e entender melhor o que acontece durante o envio de informações entre o App e o Módulo Bluetooth e com isso conseguir corrigir o que estiver causando o bug.

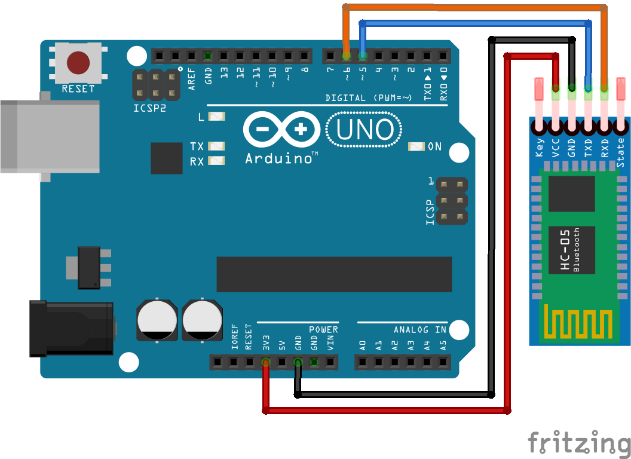
Foram testadas várias formas de envio de informação, assim como várias formas diferentes de recepção de informação.

Após várias tentativas, várias mudanças no código e no aplicativo, o motivo do erro ainda não havia sido compreendido. Então, buscou-se outros aplicativos pré-existentes no Google Play para comparar o que poderia ser feito diferente no aplicativo desenvolvido. Em um certo aplicativo encontrado, o erro permanecia, entretanto, observou-se que havia a opção de adicionar o caractere “\n” ao envio da informação, após testar essa funcionalidade, os LEDs passaram a acender e alterar seu brilho com a velocidade esperada.

Assim, foi descoberto que o que causava o problema de acúmulo dos valores era a ausência de um “\n” (caractere ASCII que representa uma nova linha) no fim dos valores enviados. Aparentemente, o Arduino quando lê o valor da variável recebida pelo aplicativo, não apenas lê, ele também aguarda uma confirmação de que o recebimento foi finalizado através da leitura de um “\n”. Enquanto o programa não recebe esse caractere, ele permanece lendo e adicionando ao valor inicial tudo aquilo que for lido, até que após alguns segundos, ele imprimiria aquele valor absurdo.

Assim, foi necessário trocar o bloco de envio de 1 byte, que estava sendo utilizado no App Inventor, para o bloco de envio de texto (em um aplicativo mais simples que não foi mostrado no relatório). E neste bloco deveria haver o valor numérico seguido do caractere “\n”, o valor seria lido, a leitura finalizada e já estava pronto para receber um novo valor, tendo assim a responsividade esperada.

**Diagrama**



**Programação**

Nesse bloco, foram criadas 3 variáveis do tipo String, sendo elas “valor”, “parte1” e “parte2” que representarão respectivamente o valor completo recebido, o primeiro dígito e o resto do valor.

#include <SoftwareSerial.h>

SoftwareSerial bluetooth (5, 6);

String valor, parte1, parte2;

char c;

Também foi criada uma variável do tipo char que receberá o valor a ser adicionado a String “valor”.

Para facilitar a visualização das informações no monitor serial, foram criados dois procedimentos para simplesmente escrever um espaço de tabulação com o procedimento “tab” e “pular uma linha” com o procedimento “linha” e no “setup” é iniciado o monitor serial.

void tab () {Serial.print("\t");}

void linha () {Serial.print("\n");}

void setup () {Serial.begin(9600);}

Se houver informações disponíveis para serem lidas, um caractere será lido e atribuído à variável c, se o valor for diferente de um “\n” (que representa o “pular linha”, segundo a tabela ASCII, é representado pelo valor 10 em decimal), ele será adicionado à String “valor”, senão, a variável “parte1” receberá o primeiro caractere da String “valor” e “parte2” receberá os outros 3 caracteres. Assim sendo, ambos impressos no monitor serial e a variável “valor” zerada.

void loop () {

if (Serial.available() > 0) {

c = Serial.read();

if (c != 10) {

valor += c;

}

else {

parte1 = valor.substring (0, 1);

parte2 = valor.substring (1, 4);

Serial.print(parte1); tab ();

Serial.print(parte2); tab ();

valor = "";

linha ();

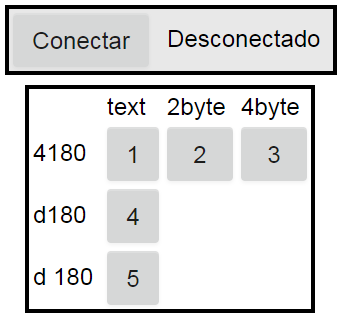
}

}

}

**App**

*Front*

Nessa tela está presente o ListPicker1, renomeado para LP1, que tem como texto “Conectar”, ele fará a seleção do dispositivo Bluetooth a ser conectado. Ao seu lado, está o Label1, renomeado para L1, que tem como texto “Desconectado” que tornará visível se a conexão está sendo estabelecida ou não. Essa configuração estará presente em todos os outros aplicativos criados, logo, para evitar repetição desnecessária, pode-se concluir que essa explicação sirva para todos os outros também.

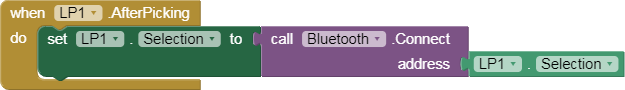
Logo abaixo, estão os botões que enviarão os diferentes valores/textos de diferentes formas.

*Back*

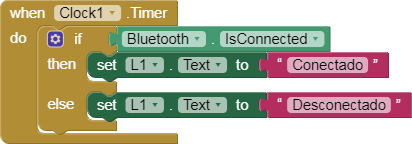
Estes são os blocos de comando que gerenciam toda a parte de bluetooth dos aplicativos que foram criados. Eles estarão presentes em todos os microprojetos abaixo, logo, para evitar constante repetição, serão omitidos, mas pode-se subentender que estarão lá.

**

Define os elementos da lista do “LP1” (ListPicker1) como sendo os endereços e nomes de dispositivos disponíveis através da conexão Bluetooth. Obs.: o “ListPicker1” é o “botão” escrito “Conectar”.

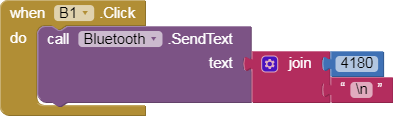
**

Quando algum dos itens da lista é selecionado, conecta o aplicativo com o dispositivo em questão. No caso dos microprojetos, o dispositivo será o módulo bluetooth HC-05.

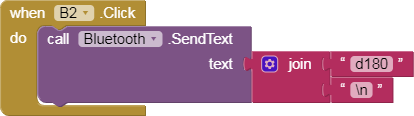
**

Esta seção fica constantemente checando se a conexão Bluetooth ainda está ativa, assim definindo a caixa de texto “L1” como “Conectado” caso e esteja ativa, ou como “Desconectado” caso não esteja.

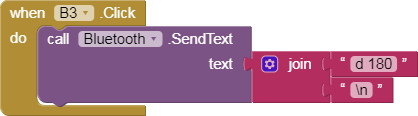
Agora, em relação ao envio da informação, como observado pelos botões numerados de 1 a 5, segue o padrão apresentado na tabela da parte visual. Cada um deles apresenta alguma mudança entre si justamente porque o intuito deste app é o de testar o envio dos dados através do Bluetooth. Cada um dos botões apresenta as seguintes funcionalidades:



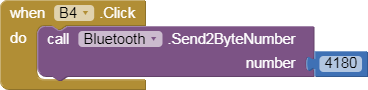
Quando o botão 1 é clicado, é enviado o texto “4180\n” via bluetooth.



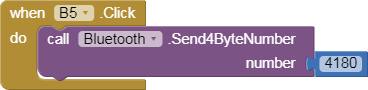
Quando o botão 2 é clicado, é enviado o texto “d180\n” via bluetooth.



Quando o botão 3 é clicado, é enviado o texto “d 180\n” via bluetooth.



Quando o botão 4 é clicado, é enviado o valor “4180” em 2 bytes via bluetooth.



Quando o botão 5 é clicado, é enviado o valor “4180” em 4 bytes via bluetooth.

Através desse teste foi possível concluir que o envio através de Bytes não era a ideal, diferente do que foi imaginado inicialmente em microprojetos não expostos. Ao invés de Bytes, a utilização do envio de informações através de texto se mostrou mais eficiente, pois, como a plataforma Arduino ao ler as informações espera o valor “\n” para que a leitura seja finalizada, é necessário que seja enviado como texto, agregado à informação desejada.

* + 1. **Uso do módulo Bluetooth para controlar a luminosidade de quatro LEDs com Sliders (App Inventor)**

Quando apenas um LED era controlado, apenas um byte de informação era necessário, pois a intensidade luminosa do LED varia entre 0 e 255, justamente a quantidade de valores presentes em um byte (8 bits), então era utilizada a opção de “send 1 byte” no App Inventor. Sendo assim, como distinguir o valor que cada LED receberia?

Duas alternativas foram analisadas:

A primeira alternativa seria o envio de um texto havendo uma letra inicial, podendo ela ser qualquer uma, mas que representasse um LED. Então, por exemplo, o texto “a122” significaria que o LED 1 teria o seu brilho no nível 122, o texto “d000”, significaria que o LED 4 seria desligado.

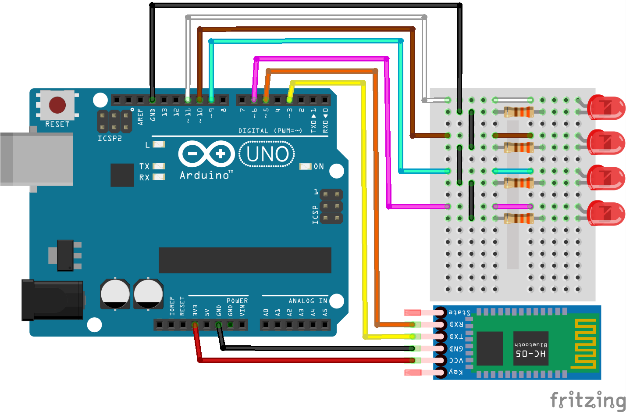
A segunda alternativa seria utilizar a casa do milhar como digito diferenciador, assim, quando o algarismo na casa do milhar for 2, o LED 2 receberia o valor de sua intensidade, no caso do valor “2200” o LED 2 teria o seu brilho em 200, e o mesmo para os outros valores. O benefício dessa alternativa seria poder utilizar o tipo inteiro para os dados, ao invés de uma String, pois sendo um número, poderiam ser feitos cálculos para separar a casa do milhar dos outros.

Essa segunda alternativa foi utilizada, porém não de forma tão simples. Isso se deve ao fato de que os valores não eram enviados como números inteiros, mas sim como caracteres individuais. Assim sendo necessário que fossem inicialmente lidos e reagrupados em uma String, para que ocorressem as mudanças devidas.

A partir daí podem ser seguidos dois caminhos para separar o algarismo do milhar: O recorte da String e a posterior transformação de ambos os valores em inteiro ou a transformação da String em um inteiro e a utilização de cálculos para separar os dois valores.

Além disto, passou a ser utilizado o slider como forma de controle de cada um dos LEDs. Entretanto, houve um problema com o tipo do dado retornado pelo Slider. Ele retornava um número real, mas, a intensidade dos LEDs apenas varia utilizando números inteiros. Então, foi utilizado no App a funcionalidade “round” para poder arredondar o valor recebido em um número inteiro.

**Diagrama**



**Programação**

Assim como em outros microprojetos anteriores, são criadas as variáveis relativas aos quatro LEDs que serão utilizados, e utilizando o conhecimento obtido no teste anterior, será utilizada a variável “c” para receber os caracteres enviados e guarda-los na variável “valor”, que terá seu primeiro digito atribuído como número inteiro à variável “led”, que definirá qual LED será acendido, e os últimos três dígitos serão atribuídos à variável “intensidade” que representa a intensidade com que o LED será acendido.

#include <SoftwareSerial.h>

SoftwareSerial bluetooth (3, 5);

const int led1 = 11, led2 = 10;

const int led3 = 9, led4 = 6;

char c;

String valor;

int led, intensidade,

São criados os procedimentos para facilitar a visualização das variáveis no monitor serial. E no “setup” é iniciada a visualização do monitor serial, e definidos os pinos dos quatro LEDs como sendo pinos de saída.

void tab () {Serial.print("\t");}

void linha () {Serial.print("\n");}

void setup () {

Serial.begin(9600);

pinMode (led1, OUTPUT);

pinMode (led2, OUTPUT);

pinMode (led3, OUTPUT);

pinMode (led4, OUTPUT);

}

Baseado no microprojeto anterior e no teste feito, segue-se a mesma linha de raciocínio. A variável “led” passa a ser utilizada como forma de decidir qual dos LEDs será aceso, assim caso a variável “led”, que foi obtida a partir da String “valor”, tenha como valor “2”, o LED 2 será ativado, com a intensidade também obtida através da variável “valor”.

void loop () {

if (Serial.available() > 0) {

c = Serial.read();

if (c != 10) {

valor += c;

}

else {

led = (valor.substring(0, 1)).toInt ();

intensidade = (valor.substring(1, 4)).toInt();

valor = "";

Serial.print("Led: ");

Serial.print(led);

tab ();

Serial.print("Intensidade: ");

Serial.print(intensidade);

switch (led) {

case 1: analogWrite (led1, intensidade);

break;

case 2: analogWrite (led2, intensidade);

break;

case 3: analogWrite (led3, intensidade);

break;

case 4: analogWrite (led4, intensidade);

break;

}

Linha ();

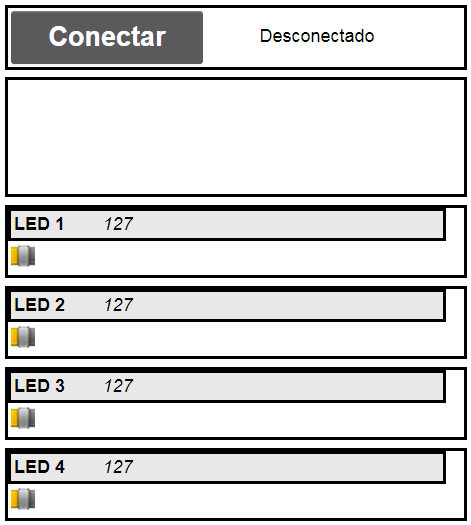
}

}

}

**App**

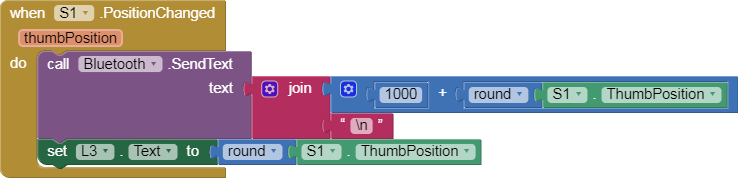
*Front*

A utilizada o ListPicker1, “Conectar” já foi supracitado, assim podendo pular essa explicação.

Abaixo dos blocos relacionados à conexão Bluetooth, estão vários Sliders e Labels. Os Sliders são barras com um botão deslizante em seu centro, podendo ir de uma extremidade a outra representando seus valores máximos e mínimos configurados em suas propriedades como foi exemplificada em sua montagem no tópico 1.1, já os Labels são caixas de textos que são alteradas para tornar visíveis as informações necessárias. Nesse caso, estão sendo indicados a qual LED o Slider e seu respectivo valor se referem. Seu valor, também presente em uma caixa de texto, é constantemente alterado para estar de acordo com a posição do Slider.

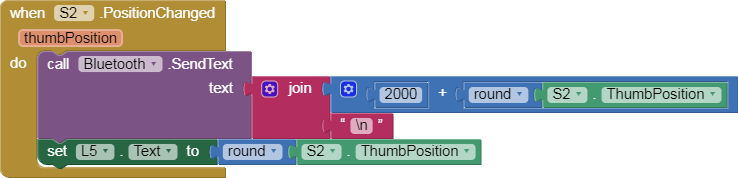
*Back*

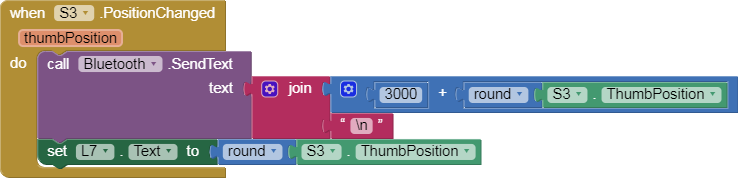
Quando a posição do Slider 1 é alterada, é enviado ao dispositivo conectado via Bluetooth um texto que é a junção da posição do Slider aproximada para um número inteiro, somada a 1000 (que representará que o LED a ser acendido é o led1) e acrescida do texto “\n”, que representa o término do envio da informação. Logo após, o texto presente na caixa de texto respectivo ao valor do LED1 também é atualizado com o novo valor do Slider1.

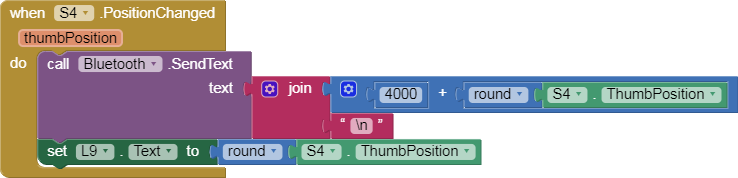


Todos os outros Sliders segue a mesma lógica, apenas fazendo certas mudanças respectivas aos Sliders, LEDs e Labels correspondentes.

[POSSÍVEL REMOÇÃO DA REPETIÇÃO DOS BLOCOS DE COMANDOS]





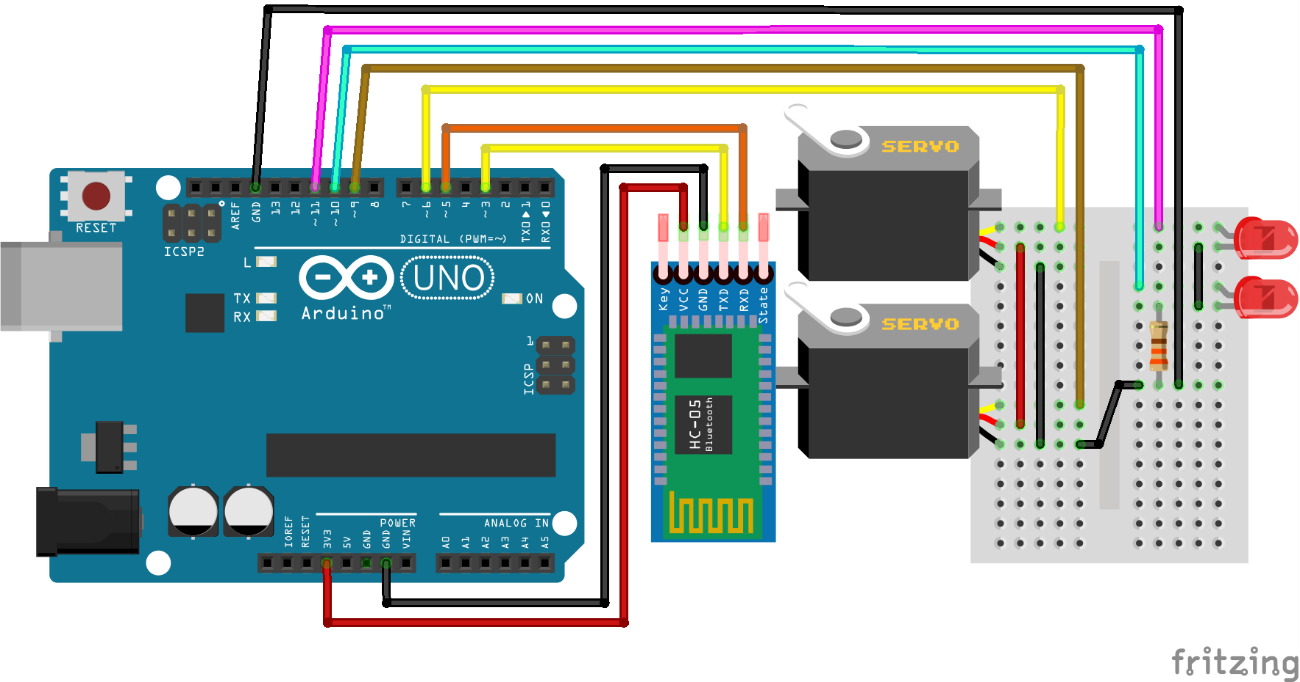


* + 1. **Aplicar as programações dos microprojetos no projeto.**

Como forma de transição entre os microprojetos e esta etapa, optou-se por primeiro fazer o controle de dois LEDs e dois Servomotores como comprovação da prática mudança do código e do aplicativo entre o que era utilizado para os LEDs e o que precisará ser utilizado no caso do braço robótico.

Um novo problema surgiu. Embora os controles estivessem funcionando normalmente quando modificados lentamente, quando a mudança era muito brusca, a conexão bluetooth se perdia. Supõe-se que o erro se deve à falta de energia recebida pelo Arduino através do cabo USB. Confirmando a necessidade da utilização de uma fonte de energia aprimorada.

Diagrama



Programação

Duas variáveis referentes aos LEDs foram substituídas por duas referentes aos pinos dos motores, mas as outras variáveis permanecem as mesmas que as do microprojeto anterior.

#include <SoftwareSerial.h>

SoftwareSerial bluetooth (3, 5);

const int led1 = 11, led2 = 10;

const int motor1 = 9, motor2 = 6;

int led, intensidade;

char c;

String valor;

Similar ao microprojeto anterior, os pinos referentes aos LEDs e motores são definidos como pinos de saída e o monitor serial é iniciado.

void tab () {Serial.print("\t");}

void linha () {Serial.print("\n");}

void setup () {

Serial.begin (9600);

pinMode (led1, OUTPUT);

pinMode (led2, OUTPUT);

pinMode (motor1, OUTPUT);

pinMode (motor2, OUTPUT);

}

A essência do programa permanece a mesma, alterando apenas que agora os LEDs 3 e 4 agora estão sendo representados por dois motores, que estão utilizando do valor da variável “intensidade” não para brilhar, mas para definir um ângulo de rotação.

void loop () {

if (Serial.available() > 0) {

c = Serial.read();

if (c != 10) {

valor += c;}

else {

led = (valor.substring(0, 1)).toInt();

intensidade = (valor.substring(1, 4)).toInt();

valor = "";

Serial.print("Led: ");

Serial.print(led);

tab ();

Serial.print("Intensidade: ");

Serial.print(intensidade);

switch (led) {

case 1: analogWrite (led1, intensidade);

break;

case 2: analogWrite (led2, intensidade);

break;

case 3: analogWrite (motor1, intensidade);

break;

case 4: analogWrite (motor2, intensidade);

break;

}

linha ();

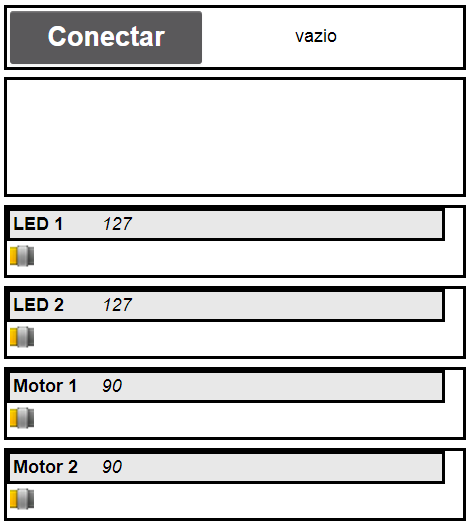
}

}

}

**App**

*Front*



Segue o mesmo padrão do microprojeto anterior, apenas tendo como alteração os dois últimos Sliders que passaram a controlar 2 motores. Então, os seus valores máximos passaram a ser 180 que é a angulação máxima aceita pelo micro servomotor. Sendo assim, o valor do meio, e também o da posição inicial dos Sliders, é 90.

*Back*

Todos os blocos são exatamente iguais aos blocos do back-end do microprojeto anterior, assim, para evitar repetição, não serão mostrados.

1. **Resultados e Discussão**

No início da Iniciação Científica (IC), foram analisados os relatórios antigos relacionados a este projeto, com o objetivo de entender mais profundamente como funciona o IC, e estudar sobre o que foi desenvolvido no projeto até então. Tendo finalizada essa etapa, focou-se em se familiarizar com o ambiente de trabalho e com os componentes disponíveis, bem como desenvolver microprojetos para relembrar o básico da programação Arduino.

Esses microprojetos deixaram de ter apenas o intuito de refamiliarização e passaram a se tornar a base de todo o desenvolvimento do projeto. Utilizando do método TDD (Test-Driven Development – Desenvolvimento Guiado por Testes) pode-se manter uma consistência no desenvolvimento. Um exemplo dado no livro “Test-Driven Development By Example” de Kent Beck, é a analogia da programação com uma manivela para que seja içado um balde com água: Se o balde não for muito pesado ou se a distância for pequena, não é necessário tanto esforço então é mais fácil de se alcançar o objetivo, entretanto, se a distância for grande e/ou o balde for pesado, seria extremamente exaustivo e não se conseguiria completar o objetivo sem que o balde caísse no chão. Como forma de auxiliar a tarefa e evitar que o balde caia no chão e todo o trabalho seja perdido, adiciona-se um mecanismo de catraca que servirá como uma constante atualização de um ponto de segurança.

Utilizando dessa tática, foi possível desenvolver o projeto num ritmo constante e gradativamente encontrando os resultados esperados. Ao longo do caminho foram sendo encontrados diversos bugs que puderam ser resolvidos pois foram encontrados parcialmente separados uns dos outros devido à caminhada lenta. Caso fosse visado o objetivo final logo de início, os bugs poderiam ter se amontoado o que tornaria ainda mais complexa as suas resoluções.

O resultado final encontra-se principalmente relacionado ao desenvolvimento de uma interface de controle e ao aprimoramento do controle do braço robótico. A montagem e programação anteriores não demonstravam muita precisão em seu controle. Assim, uma nova programação foi refeita do zero com esse objetivo. Utilizando de um aplicativo e da conexão bluetooth, é possível controlar manualmente cada um dos motores, como foi exemplificado pelo último microprojeto. Entretanto, ainda é necessário buscar uma forma alternativa de alimentação que seja suficiente para suprir toda a demanda.

1. **Conclusões**

As etapas do plano de trabalho foram em sua maioria concluídas com satisfação. As que ficaram foram a de aprimorar o controle do projeto chamado “Planta IOT” e desenvolver sua interface, e também aprimorar a fonte de alimentação para esses projetos.

Esperava-se que não fosse necessária mudança na forma de alimentação de energia, entretanto, essa expectativa mostrou-se sem fundamento, por isso espera-se que essa situação seja solucionada em projetos posteriores.

Todo o desenvolvimento impactou positivamente no aprendizado dos conceitos presentes na Internet das Coisas e inspira a futura evolução do projeto ampliando o seu alcance. Atualmente, está sendo utilizada a tecnologia Bluetooth, que embora cumpra o papel esperado de controle à distância, não tem acesso à internet. Como já está havendo a recepção de informações através de uma via remota, essa parte do código talvez não precise sofrer muita modificação.

1. **Referências**

A quarta revolução industrial livro por Klaus Martin Schwab

https://www.devmedia.com.br/tdd-desenvolvimento-guiado-por-teste/14406

https://direitodacomunicacao.com/wp-content/uploads/2018/09/IoT-bei-aConTech-1200x686.jpg

[1] CISCO. A Internet das Coisas - Como a próxima evolução da Internet está mudando tudo

[2] MATTERN, Friedman. From the Internet of Computers to the Internet of Things

[3] https://www.filipeflop.com/produto/modulo-wifi-esp8266-esp-01/

http://ai2.appinventor.mit.edu/reference/other/IoT.html

http://appinventor.mit.edu/

http://mundoprojetado.com.br/modulo-bluetooth-criando-aplicativo-parte-2/

ai2.appinventor.mit.edu/reference/components/connectivity.html#BluetoothClient

Test-Driven Development By Example Kent Beck, Three Rivers Institute

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

Como proposta a continuação do projeto, sugere-se aprimorar o aplicativo desenvolvido no MIT App Inventor para que controle o braço robótico através da internet ao invés do módulo Bluetooth, assim permitindo que seja controlada de qualquer lugar com acesso à internet. Caso não seja possível o uso do App Inventor, buscar formas alternativas, como por exemplo o uso do NodeMCU e ESP8266. Para isso, será necessário um aprendizado mais aprofundado sobre as redes computacionais que envolvem a internet.

Havendo sucesso na implementação do controle do braço robótico via internet, propõe-se também o desenvolvimento de algum dispositivo IoT que possa ter alguma funcionalidade prática no contexto da UENF. Não havendo nenhuma ideia fixa, deixando a decisão do tipo do dispositivo para o bolsista e seu orientador.

[DEVERÍAMOS JÁ AFIRMAR QUAL TIPO DE DISPOSITIVO SERÁ? SE SIM, QUE TIPO VOCÊ SUGERE?]

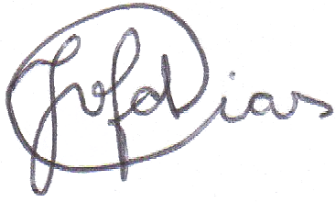
[TALVEZ USAR O SISTEMA DE IRRIGAÇÃO PREVIAMENTE MONTADO, EM MAIOR ESCALA JUNTO COM O CBB PARA AUXILIAR EM ALGUM POSSÍVEL SISTEMA DE IRRIGAÇÃO PRESENTE NA UENF, OU ISSO PASSARIA A SER RELACIONADO A BOLSA DE EXTENSÃO? ]

1. **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, onde foi ensinado o básico sobre a plataforma, proporcionando melhor desenvolvimento do atual projeto.

[ESSE TIPO DE COISA SE ENQUADRA NESSE TÓPICO? CASO NÃO: O QUE DEVO COLOCAR? CASO NÃO HAJA NADA PARA COLOCAR: DEVO APAGAR ESSE TÓPICO? DEVO DEIXAR EM BRANCO?]

1. **Data e assinatura do bolsista (assinatura digitalizada)**



XX de março de 2020.

1. **Data e assinatura do orientador (assinatura digitalizada)**

****

XX de março de 2020.