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

Nome do Bolsista: João Vítor Fernandes Dias

Curso: Ciência da Computação

N° Matrícula: 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

O plano de trabalho original, compreende as seguintes etapas:

a) Estudo da Plataforma Raspberry PI. Configuração da plataforma. Execução de projetos simples para familiarização com a plataforma. Documentação do estudo e dos projetos executados.
b) Estudo sobre formas de integração entre as plataformas Arduino, NodeMCU e Raspberry Pi. Pesquisa sobre projetos de integração e documentação do estudo.
c) Estudo sobre controle aprimorado de motores e servomotores. Introdução das melhorias nos projetos do braço robótico e da Planta IoT. Documentação.
d) 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.
e) Pesquisa sobre interfaces de controle para plataformas microcontroladas. Documentação.
f) Desenvolvimento das interfaces de controle nos projetos do braço robótico e da Planta IoT.
g) Realização de experimentos de avaliação e desempenho dos dispositivos.
h) Elaboração de relatório técnico.

Visto que a bolsista anterior, Isabela Correia, já havia executado boa parte do plano de trabalho itens a) a d). Especificamente, 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); se obtou por dar continuidade a pesquisa a partir dos estudos do item c) e concentrar nos itens e) a g).

No item c) foi escolhido o controle do braço robótico via Bluetooth uma vez que a bolsista anterior tinha abordado o projeto de irrigação da planta. 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, 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.

2. 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.

Segundo Klaus Schwab:

> 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. SCHWAB (2019, p. 1)

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.

3. 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: i) um braço robótico e ii) 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.

4. Metodologia

O método utilizado foi o desenvolvimento de vários microprojetos simples, porém, gradativamente mais complexos com o intuito de alcançar a complexidade necessária para o controle de 4 servomotores, presentes no braço robótico, 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 ilustradas na Figura 1.

Figura 1 – Etapas do TDD. Fonte: O Autor

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, de forma intuitiva e rápida, sem a necessidade de utilizar 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.

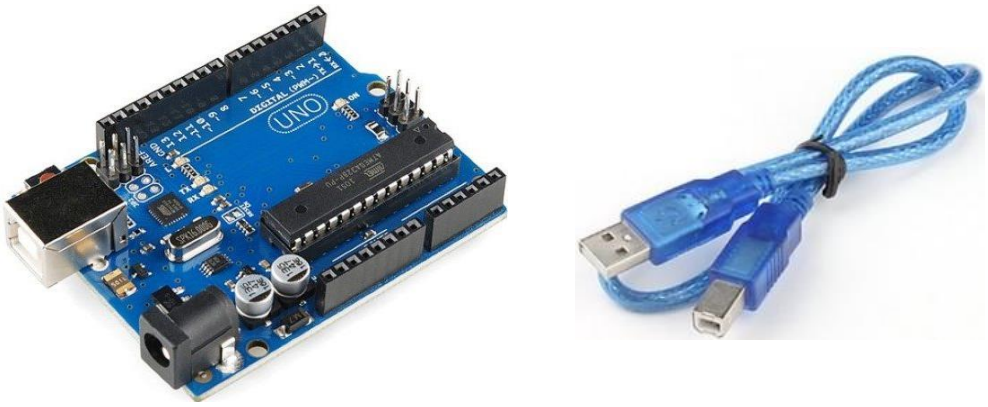
4.1. Componentes

Nesta seção são descritos de forma breve os componentes utilizados nos projetos que foram trabalhados.

4.1.1. Plataforma Arduino Uno e Cabo USB

O Arduino Uno é uma plataforma microcontrolada de baixo custo que utiliza um microcontrolador capaz de executar tarefas simples. Essa plataforma suporta um número pequeno de entradas e saídas digitais e algumas entradas analógicas (vide Figura 2a). O microcontrolador suporta uma linguagem específica baseada na linguagem C.

A plataforma Arduino utiliza um cabo USB do tipo A-B (vide Figura 2b). Este cabo serve para transferir a programação do computador para o Arduino, e também para alimentá-lo com energia.

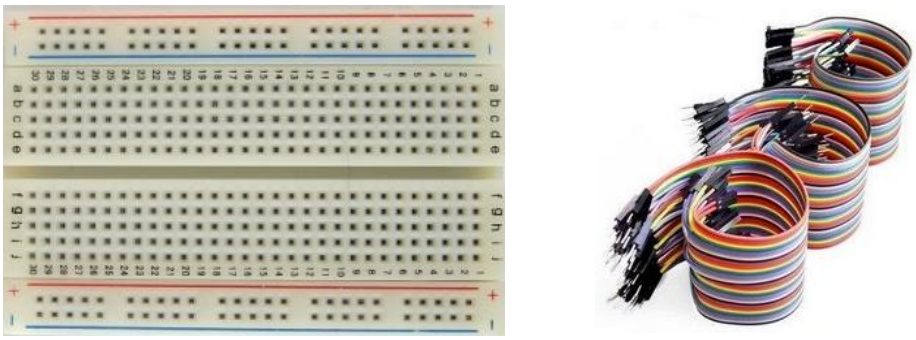


(a) (b)

Figura 2 – Plataforma Arduino Uno: a) Placa Arduino 1; b) Cabo USB. - Fonte [1]

4.1.2. Protoboard e Jumpers

A protoboard, também chamada de *Breadboard,* é uma matriz de contatos utilizada para conectar entre si diversos componentes eletrônicos sem necessidade de soldagem. Para isso, a placa possui orifícios que estão conectados internamente em linhas verticais e horizontais e que permitem o encaixe dos terminais dos dispositivos eletronicos e/ou de cabos especias denominados jumpers apenas por pressão (vide Figura 3a).



(a) (b)

Figura 3 – Protoboard e Jumpers:a) Protoboard de 400 pontos;
b) Jumpers machoxmacho. - Fonte [1]

Por outro lado, os Jumpers são 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 (vide Figura 3b).

4.1.3. Botão

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 (Ver Figura 4).

Figura 4 – Ilustração de Botões - Fonte [5]

4.1.4. Resistores

Os resistores são dispositivos que permitem regular a corrente elétrica transformando-a em energia térmica. Essa transformação depende do material utilizado e segue a fórmula (V = R X I) (Tensão = Resistência X Corrente). O nível de resistência de um resistor é representado mediante um código de cores que geralmente utiliza três faixas para identificação do valor da resistência e uma faixa adicional para denotar a tolerância (Ver Figura 5).

Figura 5*: Resistores - Fonte [6]*

4.1.5. LEDs

O 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 adequado nas portas PWM (*Pulse Width Modulation* – modulação por largura de pulso*)* e da programação no Arduino (Ver Figura 6).

Figura 6*: Resistores - Fonte [6]*

4.1.6. Potenciômetros e Joysticks

Um potenciômetro é um resistor variável que variam a sua resistência à medida em que o seu regulador é girado (Figura 7). Um Joystick é basicamente formado por dois potenciômetros acoplados a uma peça plástica de tal forma que é possível fazer variações em duas dimensões representando assim coordenadas X e Y em um plano cartesiano. (Figura 8).

Figura 7: Potenciômetro - Fonte [9]

Figura 8: Joystick - Fonte [10]

4.1.7. Micro Servomotor 9g

Um micro servomotor é um pequeno motor acoplado a um conjunto de engrenagens que permitem reduzir a velocidade de giro do motor de forma a controlá-lo. Possui um circuito que controla a rotação do eixo em graus, através de um sinal de controle. O servomotor 9g tem capacidade de rotação de até 180° (Figura 9).

Figura 9 *- Micro Servomotor - Fonte [11]*

4.1.8. Módulo Bluetooth HC-05

Um módulo Bluetooth é um dispositivo capaz de enviar e receber dados através da tecnologia Bluetooth. Para isso é necessário parear o módulo com outro dispositivo como um *Smartphone* ou computador (Figura 10).

Figura 10 – Módulo Bluetooth HC-05 - Fonte [12]

4.2. MIT App Inventor 2

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 desse ambiente é a página de desenvolvimento visual (Designer) (Figura 11). Além da página Designer existe a página de Blocos de Comando (Blocks). No canto superior direito, existem botões específicos para alternar entre ambas as páginas.

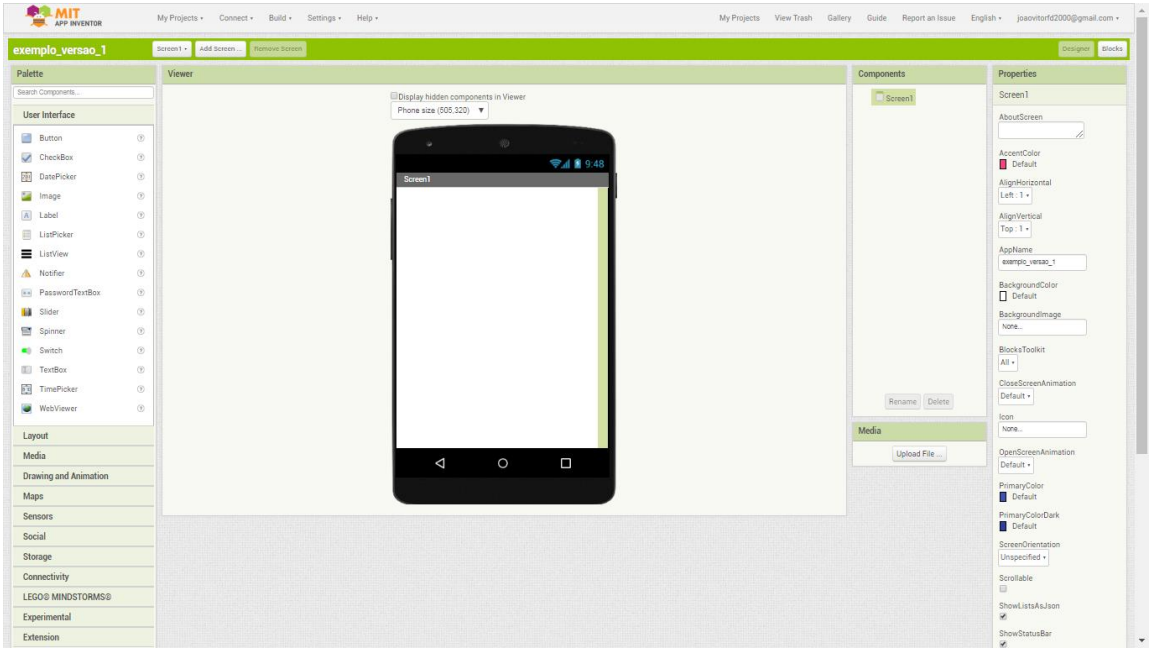


Figura 11 – Página inicial de desenvolvimento do App Inventor. Fonte:

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 arrastar-e-soltar", ou seja, para utilizar algum componente de design ou bloco de programação no aplicativo, basta arrastá-lo respectivamente, das janelas *Palette* e *Blocks* para a janela Viewer.

4.2.1. Componentes Visuais (*Front-End*)

Na página Designer existem quatro janelas que auxiliam no desenvolvimento da parte visual do aplicativo, sendo elas: Palette, Viewer, Components e Properties. Cada uma dessas janelas será apresentada brevemente.

Palette:

Esta janela dá acesso a todas as componentes visuais disponíveis no App Inventor para o desenvolvimento da interface do aplicativo. As diversas componentes estão agrupadas, segundo a sua funcionalidade, em categorias (Figura 12). Para acessar as componentes de uma categoria, basta clicar no nome dessa categoria e selecionar e arrastar a componente desejada na janela Viewer.

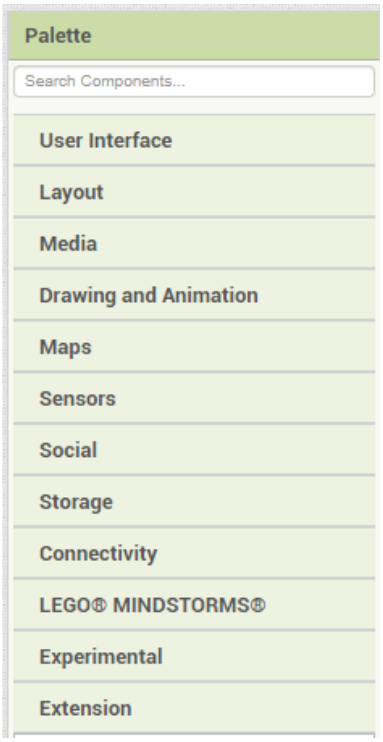


Figura 12 – Janela de Componentes Visuais do App Inventor. Fonte:

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

- User Interface.- Compreende componentes que permitem a interação do usuário com o aplicativo. Entre elas vale mencionar: Botão, Caixas de Seleção, Label, ListPicker, Sliders entre outras;
- Layout.- Compreende componentes utilizados para organização e arranjo da disposição de outras componentes.
- Conectividade.- Compreende componentes não visíveis ao usuário, que permitem que o aplicativo tenha acesso a conexão Bluetooth.

Viewer:

É uma janela que apresenta uma representação visual das telas do aplicativo sendo construído. Durante o desenvolvimento do aplicativo, as componentes visuais são selecionadas e incluídas nesta janela.

Components:

Esta janela apresenta uma lista de todos os componentes visuais que foram utilizados no desenvolvimento do aplicativo. Todo projeto inicia-se apenas com o componente "*Screen1*", porém, a medida em que novos componentes forem sendo adicionados ao projeto, eles serão listados nessa janela.

Properties:

Esta janela permite mostrar todas as propriedades de um componente previamente selecionado. Caso seja necessário, é possível modificar os valores dessas propriedades.

4.2.2. Blocos de Programação (*Back-End*)

Na página Blocks existem duas janelas que auxiliam na programação do aplicativo, são elas: Blocks e Viewer, que serão apresentadas brevemente.

Blocks: Esta janela compreende todos os blocos de comando disponíveis no App Inventor para a programação do aplicativo. Os diversos blocos estão organizados de acordo com a sua funcionalidade, em várias categorias identificadas por cores. Por exemplo, a cor laranja claro é usada para identificar as componentes que pertencem a categoria Control (controle de Fluxo). Já a cor azul é usada para identificar componentes da categoria Math (Matemática). (Figura 13) Para acessar as componentes de uma categoria, basta clicar no nome dessa categoria e selecionar e arrastar a componente desejada na janela Viewer, encaixando adequadamente a componente na posição desejada.

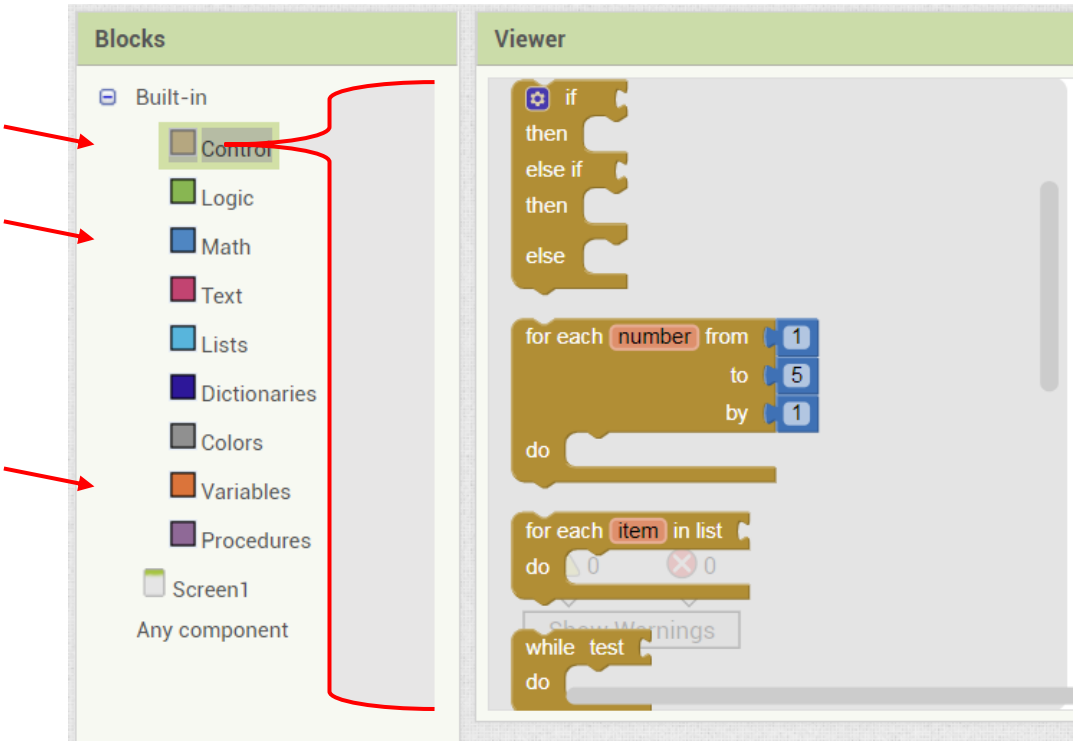


Figura 13 – Janela de Blocos do Comando do App Inventor. Fonte:

Viewer: Esta janela funciona como uma área de trabalho para construir um programa através de blocos de comandos. Nela, serão organizados e/ou encaixados diversos blocos de comando que funcionarão em conjunto de maneira a produzir um aplicativo funcional (Figura 14). Além disso, esta janela apresenta alguns recursos adicionais: Um par de avisos de erros que mostram a quantidade e o tipo erro apresentado no aplicativo; Uma mochila que funciona como uma forma de guardar grupos de blocos que serão frequentemente utilizados com o objetivo de facilitar a reutilização; Um botão centralizador, que permite voltar o zoom para o centro do desenvolvimento; Controles para aumentar e reduzir o zoom nos blocos de comando; e uma lixeira para descartar os blocos desnecessários.

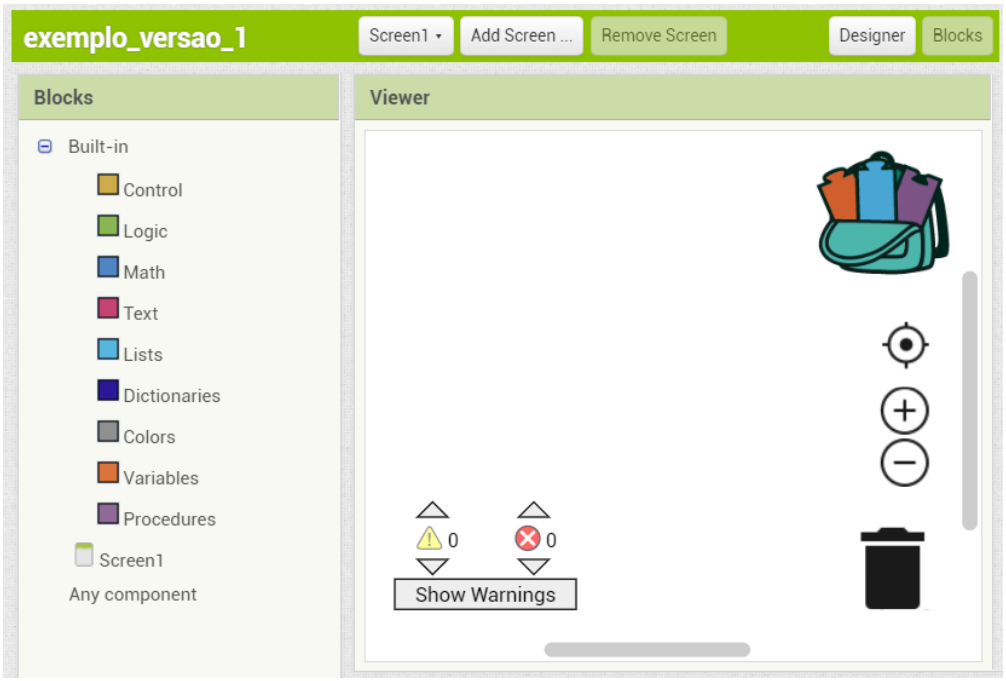


Figura 14 – Janela Viewer dos Blocos de Comando do App Inventor. Fonte:

4.2.3. Criando um projeto

Nesta seção será apresentado um projeto de exemplo que ilustra o uso de controles deslizantes (*sliders*). Este tipo de controle será utilizado em alguns aplicativos desenvolvidos nos projetos da seção 4.3. Para criar um aplicativo para dispositivo móvel usando o MIT App Inventor, primeiro deve acessar a página: http://appinventor.mit.edu/. Nesta página, deverá clicar no botão "Create Apps!" e fazer login na sua conta Google. Depois, deverá clicar no botão "Start new project" para iniciar um novo projeto e atribuir um nome a esse projeto. O exemplo desta seção será denominado "Relatório_Exemplo_Slider".

Para criar o aplicativo, deve-se trabalhar basicamente em duas frentes: i) desenvolvimento visual do aplicativo que envolve o design das telas do aplicativo, seus menus e controles e ii) a programação das funcionalidades do aplicativo que é realizado através de blocos de comando disponíveis no MIT App Inventor.

Na página de desenvolvimento visual (*designer*), neste exemplo, temos apenas uma componente de tela (*Screen1*) e nela foram incluídos dois componentes de arranjo horizontal

(*HorizontalArrangement1 e HorizontalArrangement2*). Dentro de cada um desses componentes foram incluídos um Rótulo (*Label*) e um botão deslizante (*Slider*) (Figura 15).

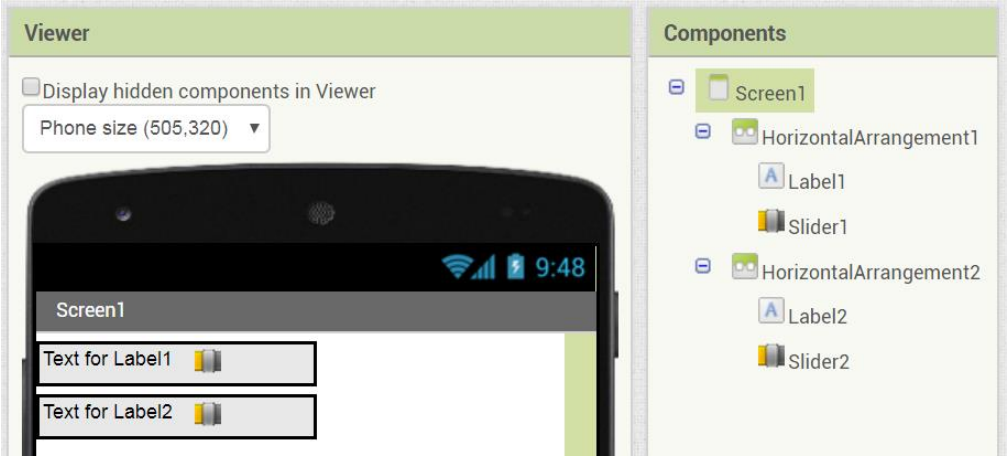


Figura 15 – Design da parte visual do aplicativo de exemplo – Fonte: O autor

Para aprimorarmos o visual do aplicativo foi necessário fazer alguns ajustes nas propriedades dos componentes , dentre elas podemos mencionar:

Screen1: Alinhar a sua posição horizontal no centro; tornar o título invisível.

HorizontalArrangement1: Alinhar a sua posição vertical no centro da tela; tornar a sua largura igual a todo o espaço disponível na tela.

Slider1: Selecionar cores de preferência para os extremos esquerdo e direito; tornar a largura igual a todo o espaço disponível; definir como 0 o valor mínimo correspondente ao brilho de um LED; definir como 255 o valor máximo correspondente ao brilho de um LED; definir como 126 a posição inicial do botão deslizante ou posição central.

Label1: Tornar texto em negrito; definir como texto visível o valor do Slider1; definir a largura como 30 pixels; alinhar o texto ao centro.

Na parte de programação realizada através de blocos de comando, basicamente foi programado o comportamento dos controles deslizantes *Slider1* e *Slider2*. Sempre que um desses botões deslizantes for movimentado (*Slider1* ou *Slider2*), a componente rótulo associada (*Label1* ou *Label2*) terá o seu texto atualizado com o novo valor da posição do botão deslizante. A figura 16 ilustra os blocos de comandos correspondentes ao *Slider1*, sendo semelhante para o Slider2.

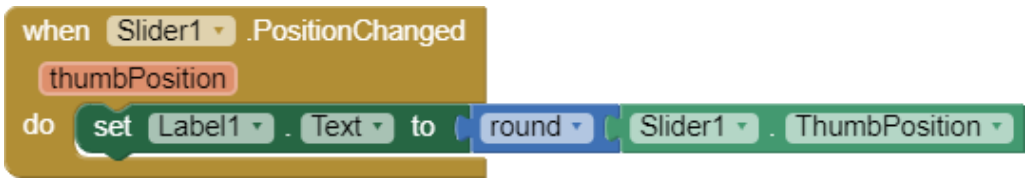


Figura 16 – Bloco de Comando para o *Slider1* – Fonte: O autor

Para concluir o projeto e instalá-lo em dispositivo móvel, será necessário clicar na opção Build do menu principal do MIT App Inventor (Figura 17) e escolher entre duas alternativas: i) instalar o pacote .apk através de QR code; ou ii) salvar o pacote .apk localmente no computador. Em ambos casos, o código associado ao projeto será compilado gerando um pacote .apk.

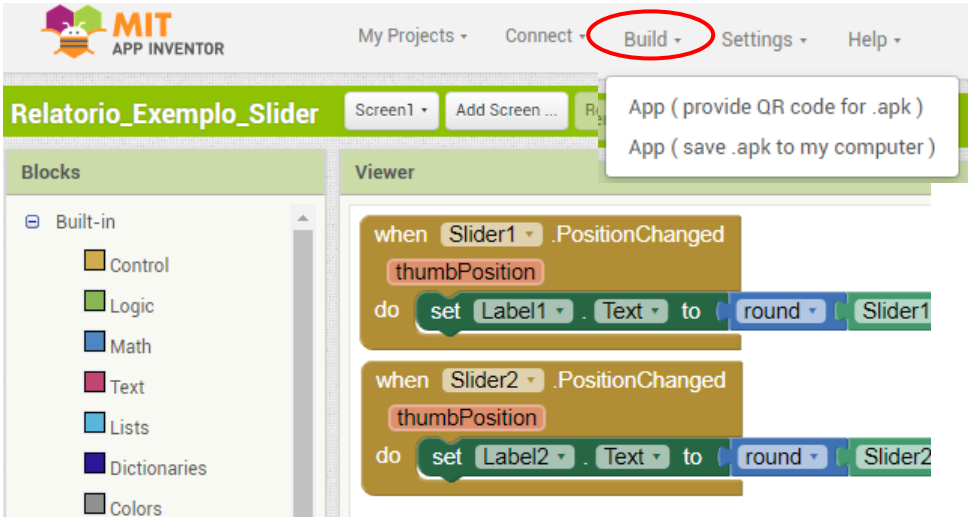


Figura 17 – Geração do pacote .apk e instalação do aplicativo – Fonte: O autor

No caso de escolher a primeira alternativa, aparecerá na tela um QRCode e abaixo dele um link (Figura 18). Para instalar o aplicativo será necessário ler o código QR usando um aplicativo apropriado ou acessar o link diretamente no navegador do seu dispositivo móvel.

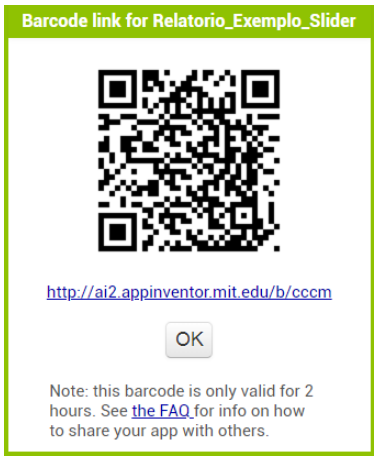


Figura 18 – QR Code gerado pelo App Inventor – Fonte: O autor

No caso de escolher a segunda alternativa, um arquivo com o nome do projeto e extensão .apk será salvo no computador. Para instalar o aplicativo no seu dispositivo móvel, basta copiar esse arquivo no seu dispositivo e realizar a sua instalação utilizando algum gerenciador de pacotes .apk.

4.3. Resultados: Microprojetos

Como mencionado no início da seção 4, o presente projeto visa o aprimoramento do mecanismo de controle do braço robótico desenvolvido em etapas anteriores. Com essa finalidade foi adotada uma estratégia de estudo e aprimoramento gradativo através do desenvolvimento de microprojetos. Os microprojetos visam primeiro entender o controle através de potenciômetros e joysticks, depois o controle sem fio Bluetooth e finalmente o controle dos servomotores através do bluetooth.

4.3.1. Controle de acendimento de quatro LEDs usado potenciômetros

O projeto apresentado nesta seção mostra o controle de acendimento de quatro LEDs usando dois potenciômetros conectados a uma placa Arduino. Uma vez que o braço robótico, objeto de estudo, é controlado mediante o envio de dados para quatro servomotores, neste primeiro projeto, simula-se o controle de quatro dispositivos mediante o acendimento de quatro LEDs e para isso utiliza-se dois potenciômetros. Os potenciômetros são dispositivos analógicos que registram valores numéricos na escala entre 0 e 1023. A ideia neste projeto é simular um plano cartesiano, com pontos (x, y) definidos a partir de valores do eixo x, representados pelo potenciômetro X, e de valores do eixo y, representados pelo potenciômetro Y. Como a escala de valores em cada eixo varia de 0 a 1023, considera-se que o ponto central do plano é o ponto (512, 512). Com base nesse ponto central é possível identificar quatro quadrantes (Figura 19).

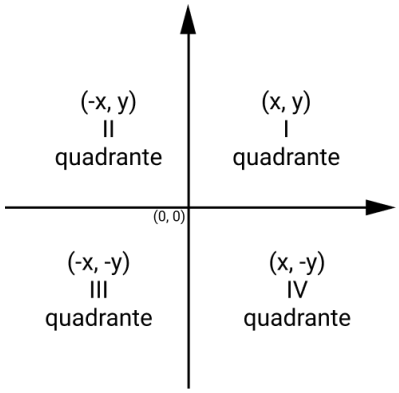


Figura 19 – Plano Cartesiano com quatro quadrantes - Fonte [1]

Por exemplo, quando os dois potenciômetros tivessem valores acima de 512, estaríamos no primeiro quadrante e apenas o LED1 ficará ligado. Por outro lado, se o potenciômetro X tiver valor inferior a 512, enquanto o potenciômetro Y tiver valor superior a 512, estaríamos no segundo quadrante e apenas o LED2 ficará ligado.

Diagrama

O diagrama da Figura 20 ilustra as conexões entre dois potenciômetros, o Arduino Uno e quatro LEDs. Os potenciômetros alimentam as entradas analógicas A0 e A1 na placa Arduino Uno. A combinação desses entradas analógicas ativam quatro saídas digitais, pinos 6, 9, 10 e 11 que são usados para acendimento dos quatro LEDs.

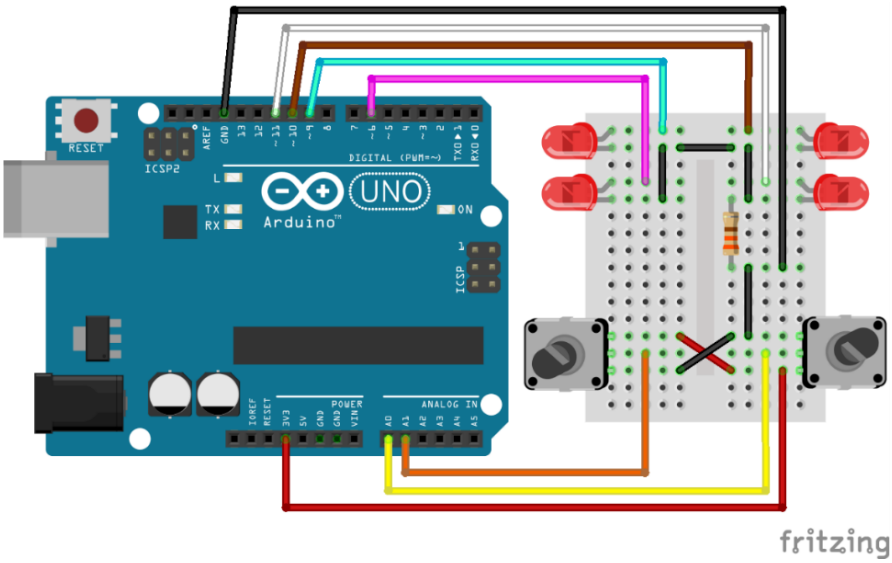


Figura 20 – Controle de acendimento de LEDs usando potenciometros. Fonte: O autor

Programação

O programa compreende quatro seções: Inicialização, Configuração, Funções Auxiliares e Loop Principal. Na seção de inicialização, define-se as constantes e variáveis que serão utilizadas. As constantes led1, led2, led3 e led4 representam os pinos de saída usados para ativar os quatro LEDs. Já as constantes X e Y representam os pinos analógicos de entrada usados pelos potenciômetros. Define-se duas variáveis inteiras EntradaX e EntradaY que receberão diversos valores numéricos provenientes dos potenciômetros. Finalmente, as variáveis inteiras “MeioX” e “MeioY” representarão a posição central (X,Y) do plano cartesiano.

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

Na seção de configuração, função setup(), são definidos todos os LEDs como pinos de saída. Além disso, inicializa-se a posição central do plano cartesiano de acordo com a leitura de ambos potenciômetros. Na seção de funções auxiliares são criados duas funções: liga() e desliga(). Essas funções realizam o acendimento e desligamento de um LED passado como parâmetro.

```
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);}
```

Finalmente, na seção de loop principal, realiza-se a leitura das posições de ambos potenciômetros, e com base neles determina-se o quadrante correspondente e procede-se ao acendimento do LED associado e ao desligamento dos LEDs restantes. 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. Os outros quadrantes são resolvidos de maneira semelhante.

```
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);
  }
 }
}
```

4.3.2. Controle de luminosidade de 4 LEDS através de um joystick

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

O diagrama da Figura 21 ilustra as conexões entre um joystick, o Arduino Uno e quatro LEDs. O Joysticks é um dispositivo compacto que substitui o uso de dois potenciômetros. Ele usa as entradas analógicas A0 e A1 na placa Arduino Uno, para representar os eixos X e Y. A combinação dessas entradas analógicas ativam quatro saídas digitais, pinos 6, 9, 10 e 11 que são usados para acendimento dos quatro LEDs.

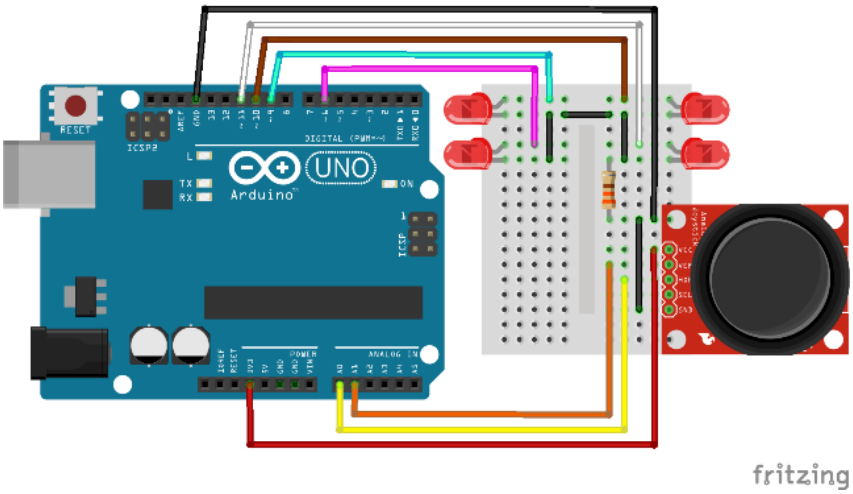


Figura 21 – Controle de luminosidade de LEDs usando um joystick. Fonte: O autor

Programação

O programa compreende quatro seções: Inicialização, Configuração, Funções auxiliares e Loop Principal. Na seção de inicialização, define-se as constantes e variáveis que serão utilizadas. Esta seção é quase idêntica a seção correspondente do projeto anterior. No entanto, as variáveis inteiras MapX e MapY foram adicionadas para representar a intensidade luminosa dos LEDs de acordo com a posição (X,Y) no plano cartesiano. Além disso, foi adicionada a variável inteira MaiorMap para representar qual dos dois eixos tem maior valor.

```
const int led1 = 10, led2 = 9, led3 = 6, led4 = 11;
const int X = A0, Y = A1;
int EntradaX, EntradaY, MeioX, MeioY;
int MapX, MapY, MaiorMap;
```

A seção de configuração, função setup(), é idêntica a seção correspondente do projeto anterior. Na seção de funções auxiliares, foi definida apenas uma função para desligar LEDs, função de desliga(). Não foi considerado pertinente ter uma função para ligar LEDs uma vez que precisaria incluir também a intensidade. O acendimento é realizado diretamente usando a função digitalWrite().

```
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);}
```

A seção de loop principal é semelhante a seção correspondente no projeto anterior. Neste caso, realiza-se a leitura da posição do joystick no plano cartesiano (EntradaX, EntradaY) e com base nesses valores é feita uma sequência de comparações que resultam no quadrante referente a posição do Joystick para ativar o LED correspondente com a intensidade igual ao valor do MaiorMap. Para determinar o valor da intensidade MaiorMap, se o valor lido pelo Joystick, tanto no eixo X quanto no eixo Y, for maior que o valor central (aprox. 512), será feita uma conversão da escala (MeioX, 1023) ou (Meio Y,1023) para um valor na escala (0, 255), resultando nos valores MapX e MapY. Caso contrário, se o valor lido for menor que o centro, será feita uma conversão de escala inversa entre (MeioX, 0) ou (Meio Y, 0) para um valor na escala (0, 255). Finalmente os valores MapX e MapY serão comparados e o maior valor será atribuído como valor da intensidade em MaiorMap.

```
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);
     }
   }
}
```

4.3.3. Controle de um Servomotor mediante um Potenciômetro

Os testes anteriores ilustraram o controle de LEDs através de potenciômetros e joysticks. Dado que o projeto de braço robótico exige o controle de servomotores, o projeto abordado nesta seção ilustrará as particularidades no uso desses dispositivos, como por exemplo, o uso da biblioteca Servo.h.

Diagrama

O diagrama da Figura 22 ilustra as conexões entre o potenciômetro, o Arduino Uno e o Servomotor. O potenciômetro alimenta uma entrada analógica na placa Arduino Uno, A0, e o valor dessa entrada é transformada em uma saída, na porta digital 3, que representa o ângulo de giro para o braço do potenciômetro.

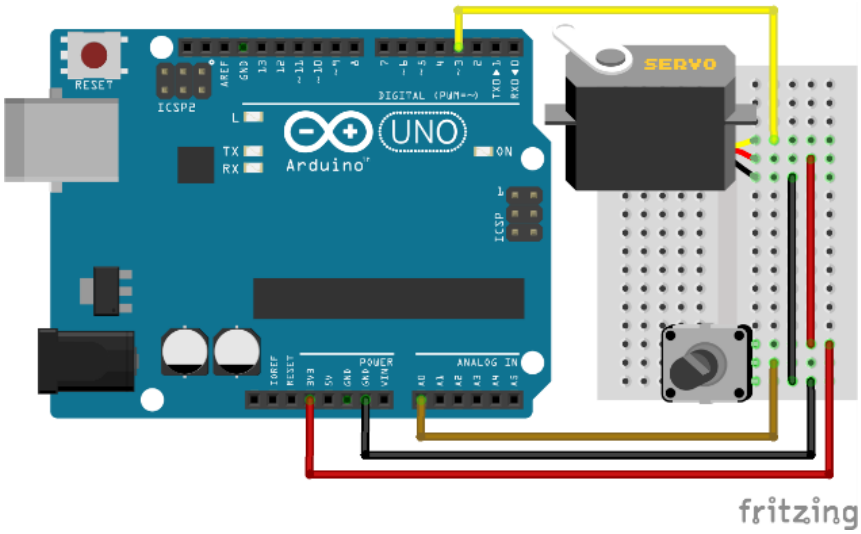


Figura 22 – Controle de um Servomotor mediante potenciômetro. Fonte: O autor

Programação

O programa compreende três seções. Na primeira seção, inclui-se bibliotecas, objetos, constantes e variáveis que serão utilizadas. Adiciona-se a biblioteca Servo.h que permite criar um objeto servomotor, denominado servo1. Define-se a constante inteira Pot associada a entrada analógica A0 e uma variável inteira ValPot, para armazenar o valor correspondente a posição do potenciômetro.

```
#include <Servo.h>
Servo servo1;
const int Pot = A0
int = ValPot;

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

void loop ()
{
  ValPot = analogRead (Pot);
  ValPot = map (ValPot, 0, 1023, 0, 180);
  servo1.write(ValPot);
  delay (15);
}
```

Na segunda seção, de configuração, associa-se o pino 3 da placa Arduino a entrada do servomotor. Na última seção, de loop principal, o valor do potenciômetro é lido mediante a função analogRead(). Depois o valor analógico (entre 0 e 1023) é convertido um valor entre 0 e 179 para representar um ângulo de giro para o braço do servomotor. A conversão é realizada pela função map(). Finalmente, esse valor é enviado para o servo, usando o método write(), que assume essa angulação.

4.3.4. Controle de acendimento de um LED mediante módulo Bluetooth

O projeto apresentado nesta seção ilustra o funcionamento de um módulo Bluetooth para ligar e desligar um LED conectado ao módulo Arduino. Este projeto representa o primeiro desafio quanto ao uso da tecnologia de Internet das Coisas (IoT). Neste sentido, foi observado que existem aplicativos para dispositivos Android, disponíveis no Google Play Store, para auxiliar no controle de dispositivos Arduino através de Bluetooth. Dentre esses aplicativos foi escolhido o aplicativo "Arduino Bluetooth Controller" que permite o pareamento entre o smartphone e um módulo Bluetooth, assim permitindo o envio de dados 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

O diagrama da Figura 23 ilustra as conexões entre o módulo Bluetooth, o Arduino Uno e um LED. Vale observar que a comunicação do módulo Bluetooth com o Arduino é realizada através dos terminais TX (Transmissor) e RX (Receptor).

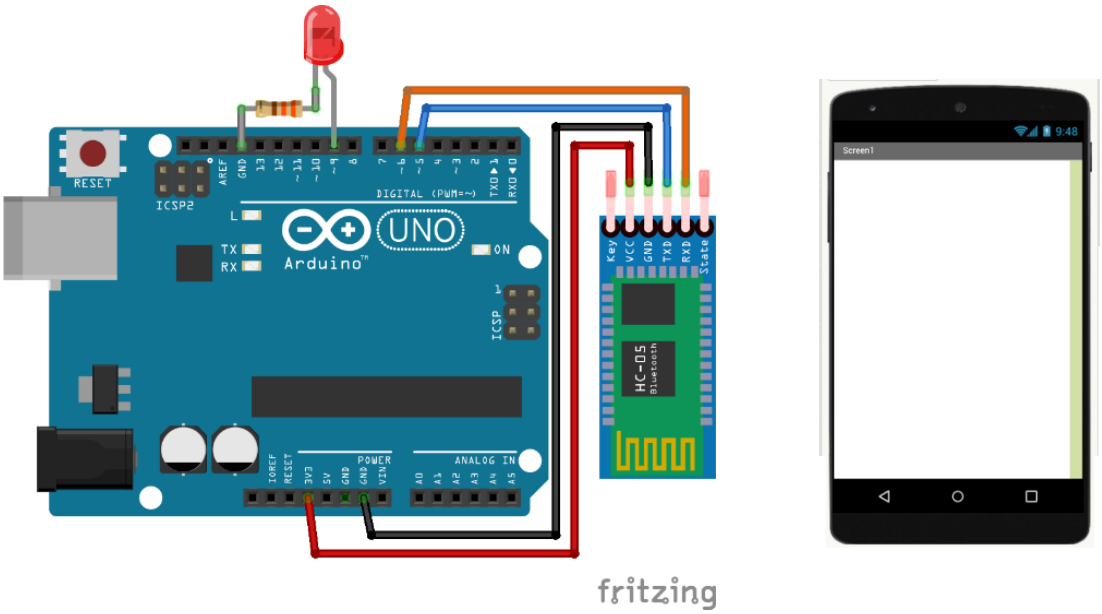


Figura 23 – Controle de acendimento de um LED mediante Bluetooth. Fonte: O autor

Programação

O programa compreende três seções. Na primeira seção, inclui-se bibliotecas, objetos, constantes e variáveis que serão utilizadas. Inclui-se a biblioteca SoftwareSerial.h que permite criar objetos para representar dispositivos de comunicação serial, como os dispositivos bluetooth. Assim, foi criado um objeto denominado bluetooth e configurado para que os pinos digitais 5 e 6 do Arduino, fossem utilizados como transmissor, TX, e receptor, RX, do módulo Bluetooth. Além disso, foi definida uma constante inteira, led, associada ao pino digital 9, e também uma variável do tipo caractere para receber o valor que definirá se o LED ligará ou apagará.

Na segunda seção, de configuração, define-se o pino 9 da placa Arduino como saída. Na última seção, de loop principal, 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 após 40 milisegundos.

```
#include <SoftwareSerial.h>
SoftwareSerial bluetooth (5, 6); // TX, RX do HC-05
const int led = 9;
char data;

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

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);
}
```

4.3.5. Controle de luminosidade de um LED mediante módulo Bluetooth

Similar ao microprojeto anterior, utiliza-se um aplicativo android para acender um único led conectado a uma placa Arduino e acessível através de um módulo bluetooth. No entanto, desta vez, pretende-se controlar a luminosidade do led. Para isso, utiliza-se um sinal analógico ao invés do sinal digital, o que permite o controle da luminosidade através do envio de valores entre 0 e 255 do aplicativo para o módulo bluetooth. Neste projeto, ao invés de utilizar um aplicativo disponível no Google Play foi implementado um aplicativo próprio, que pareava o módulo Bluetooth com o Smartphone. Foram utilizados quatro botões para enviar os seguintes valores numéricos: 0, 64, 128, 255 respectivamente para um LED.

Diferentemente do microprojeto anterior, a informação que será enviada neste caso será mais longa, e por isso será utilizada uma variável do tipo String ao invés de apenas um carater.

Diagrama

Neste caso, o diagrama de conexões entre os componentes é idêntico ao diagrama da seção anterior e por isso será omitido.

Programação

O programa compreende três seções. Na primeira seção, inclui-se bibliotecas, objetos, constantes e as variáveis que serão utilizadas. Esta seção é identica a seção do projeto anterior, com duas exceções. Inclui-se 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. Na segunda seção, de configuração, define-se o pino 9 da placa Arduino como saída.

```
#include <SoftwareSerial.h>
SoftwareSerial bluetooth (5, 6); // TX, RX do HC-05
const int led = 9;
String stringGeral;
int     brilho;

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

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);}
  }
}
```

Na terceira seção, de loop principal, é verificado se existe algum valor sendo recebido na porta serial. Em caso afirmativo, inicializa-se a string denominada stringGeral e repete-se a leitura de dados da porta serial, que convertidos para o tipo char são anexados na string. Após todos os dados terem sido recebidos, é utilizado uma função para converter a variável String em um valor inteiro e armazená-lo na variável brilho. Com isso, se o valor estiver entre 0 e 255, utiliza-se a função analogWrite() para acender o LED com intensidade correspondente a esse valor.

Durante o teste do aplicativo e do protótipo, embora se tivesse alcançado o objetivo de alterar a luminosidade do LED foram observados alguns problemas com a responsividade do aplicativo. Sempre que a luminosidade do LED era alterada muito rapidamente, os valores recebidos pelo módulo e mostrados pelo Monitor Serial, que deveriam se encontrar entre 0 e 255, estavam virando, sem nenhuma razão aparente, números enormes ou até mesmo números negativos, produzindo um comportamento errôneo.

Posteriormente, foi observado que era necessário adicionar o caractere “\n” após o envio dos dados. Após fazer esse teste, o LED passou a acender e alterar seu brilho com a velocidade esperada.

4.3.6. Envio de dados para o Arduino desde um Dispositivo Android

Neste projeto foram testadas várias formas de envio de dados entre o aplicativo Android e a placa Arduino via Bluetooth. Optou-se por desenvolver um projeto para estudar melhor o que acontece durante o envio de informações via Bluetooth. Foi testado o envio de dados no formato de texto e no formato numérico de bytes, assim como o uso do caráter especial "\n" ("nova linha") como delimitador entre o envio desses dados.

Diagrama

O diagrama da Figura 24 ilustra as conexões entre o módulo Bluetooth, o Arduino Uno e um LED. Vale observar que a comunicação do módulo Bluetooth com o Arduino é realizada através dos terminais TX (Transmissor) e RX (Receptor).

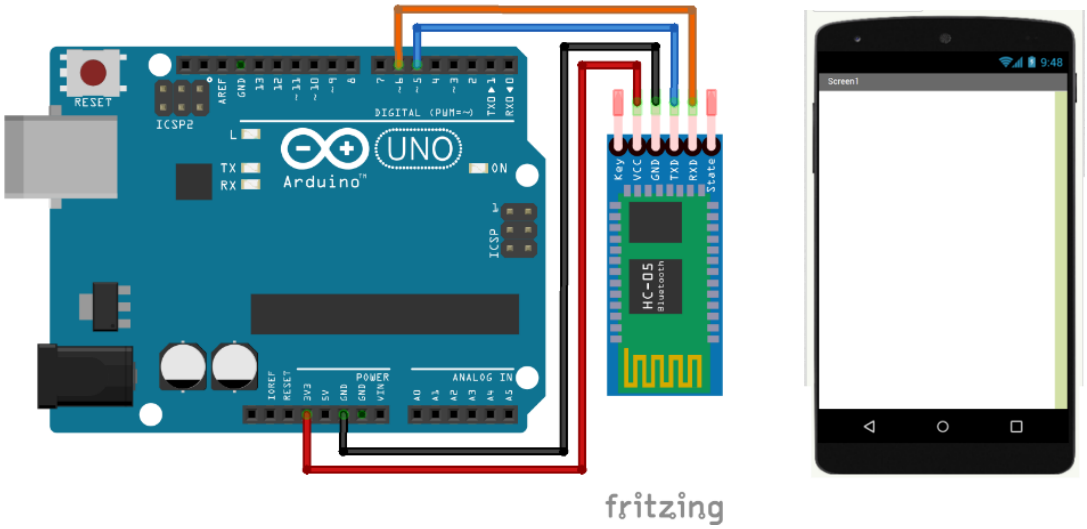


Figura 24 – Envio da dados para Arduino desde o dispositivo Android. Fonte: O autor

Apresenta-se também os detalhes da interface do aplicativo desenvolvido no App Inventor (Ver Figura 25) para o envio de dados para o dispositivo Arduino usando Bluetooth. Na parte superior da tela, no lado esquerdo, foi incluído um componente ListPicker1, renomeado para LP1, que tem como texto "Conectar", ele fará a seleção do dispositivo Bluetooth a ser conectado. No lado direito, foi incluído um componente Label1, renomeado para L1, que serve para mostrar o estado da conexão entre o dispositivo móvel e a placa Arduino (Inicialmente aparece como "Desconectado"). Na parte central da tela, foram incluídos cinco botões junto a seus respectivos rótulos. Cada botão corresponde ao um tipo de envio de dados diferente, sendo possível o envio de texto e de valores numéricos de 1 até 2 bytes.

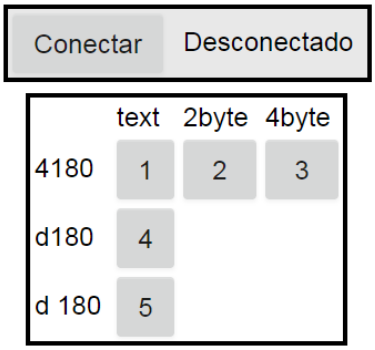


Figura 25 – Design do aplicativo android para o envio de dados via Bluetooth. Fonte: O autor

Programação

Em primeiro lugar apresenta-se a programação do módulo Arduino. Assim como em outros projetos anteriores, o programa compreende várias seções. Destacam-se quatro seções: Inicialização, Configuração, Funções auxiliares e Loop Principal. Na seção de inicialização, inclui-se bibliotecas, objetos e as variáveis que serão utilizadas. Foram definidas três variáveis do tipo String, sendo elas "valor", "parte1" e "parte2". A variável "valor" serve para armazenar um valor numérico enviado em forma de texto. A variável "parte1" armazenará apenas o primeiro dígito desse valor numérico. Já a variável "parte2" armazenará os dígitos restantes. Também foi definida uma variável do tipo char como variável auxiliar que receberá um dígito de cada vez.

```
#include <SoftwareSerial.h>
SoftwareSerial bluetooth (5, 6);
String valor, parte1, parte2;
char c;
```

Na seção de configuração, setup(), é iniciada a visualização do monitor serial. Já, na seção de funções auxiliares, são definidos as funções tab() e linha() para facilitar a visualização das variáveis no monitor serial.

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

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

Na última seção, de loop principal, realiza-se a leitura de dados transmitidos caracter por caracter e armazenados na string "valor". Se houver informações disponíveis para serem lidas, um caractere será lido e atribuído à variável c. Sempre que o valor for diferente de um "\n" ("nova linha" representado pelo valor decimal 10 no código ASCII), ele será adicionado à String "valor". Caso contrário, o primeiro dígito dessa string será extraído na variável "parte1"

```
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 ();
     }
  }
}
```

enquanto os outros três dígitos dessa string serão armazenados na variável “parte2”. O conteúdo de ambas variáveis será mostrado no monitor serial.

Apresenta-se também a programação do aplicativo android utilizando o App Inventor mediante blocos de comandos. Em primeiro lugar, apresenta-se os blocos de comando que gerenciam a conexão bluetooth entre o dispositivo Android e a placa Arduino. Esses blocos também estarão presentes em todos os microprojetos subsequentes, mas serão omitidos para evitar constante repetição (Ver Figura 26).

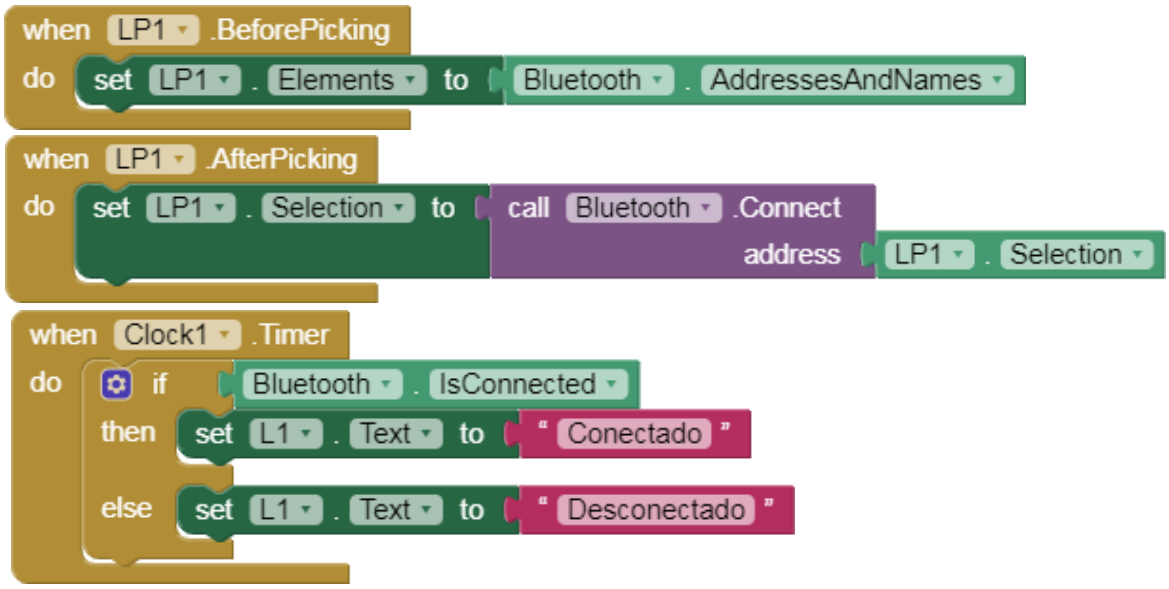


Figura 26 – Blocos de controle para conexão do Arduino via Bluetooth. Fonte: O autor

O primeiro bloco configura o botão “Conectar” definido como uma lista de elementos de seleção “LP1” (*ListPicker1*) de forma que permita selecionar os endereços e nomes dos dispositivos disponíveis através da conexão Bluetooth. O segundo bloco é executado após algum dispositivo bluetooth da lista LP1 ser selecionado. Este bloco realiza a conexão entre o aplicativo android e o dispositivo bluetooth, que neste projeto será o módulo bluetooth HC-05. O terceiro bloco, atualiza o estado da conexão bluetooth e a visualiza na caixa de texto L1. Este bloco fica constantemente checando se a conexão Bluetooth ainda está ativa, e definindo a caixa de texto “L1” como “Conectado” caso e esteja ativa, ou como “Desconectado” caso não esteja.

Apresenta-se também os blocos de comando referentes ao envio da dados desde o aplicativo Android para a placa Arduino via Bluetooth. Trata-se de cinco blocos de comandos, ilustrados na Figura 27, cada um deles associado a um dos cinco botões apresentados na tela de interface. Cada bloco de comando permitiu testar uma forma de envio de dados seja usando texto ou valores numéricos.

Os três primeiros blocos enviam dados na forma de texto. O primeiro bloco envia o texto “4180\n” quando o botão 1 é clicado. De maneira semelhante, o segundo e terceiro blocos enviam os textos “d180\n” e “d 180\n” respectivamente. Já os dois últimos blocos enviam dados como valores numéricos. O quarto bloco envia o valor numérico “4180” em 2 bytes, enquanto o quinto bloco envia o valor “4180” em 4 bytes.

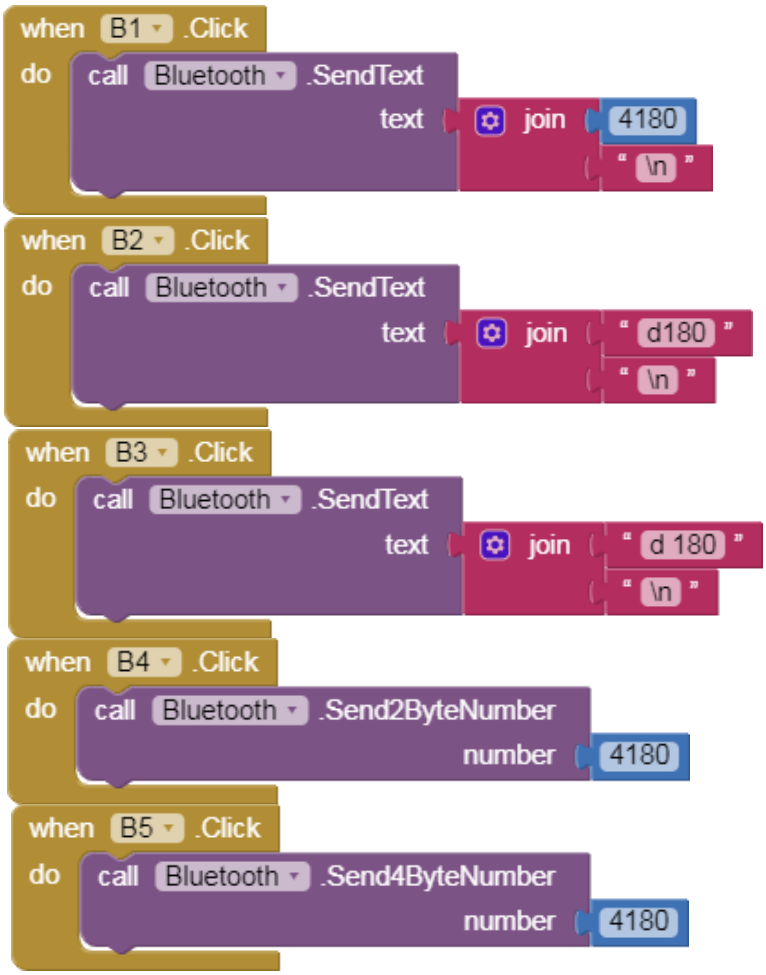


Figura 27 – Blocos de controle para envio de dados via Bluetooth. Fonte: O autor

Através deste projeto foi possível concluir que a ausência do caracter especial "\n" no fim dos valores enviados produz um acúmulo de valores e uma interpretação errônea. 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. Por este motivo, resultou mais conveniente utilizar o bloco de envio de texto seguido do caractere "\n", ao invés do bloco de envio de bytes.

4.3.7. Controle de luminosidade de LEDs mediante um aplicativo com controles deslizantes

Nesta seção apresenta-se um projeto para o controle de luminosidade de quatro LEDs conectados a uma placa Arduino Uno e acessíveis através de um modulo bluetooth. O controle remoto dos LEDs é realizado através de um aplicativo desenvolvido no App Inventor que utiliza controles deslizantes (*Sliders*). Quando apenas um LED era controlado, apenas um byte (8 bits) de dados era necessário, pois a intensidade luminosa do LED varia entre 0 e 255. Neste caso, foi utilizada a opção de "Send1byte" no App Inventor.

No caso envolvendo o controle de 4 LEDs, são necessários dados adicionais para identificador do LED que será manipulado. Para abordar esta questão foi utilizado um digito adicional, que corresponderia a casa do milhar, como digito diferenciador. Por exemplo, no caso do valor "2200", o dígito na casa do milhar com valor 2, identifica o LED 2 que terá o seu brilho ajustado no valor 200.

Diagrama

Primeiro ilustramos as conexões entre os componentes utilizados no projeto. O diagrama da Figura 28 ilustra as conexões entre o módulo Bluetooth, o Arduino Uno e 4 LEDs. Enquanto o controle dos LEDs é feito de maneira remota pelo aplicativo usando controles deslizantes através do Bluetooth.

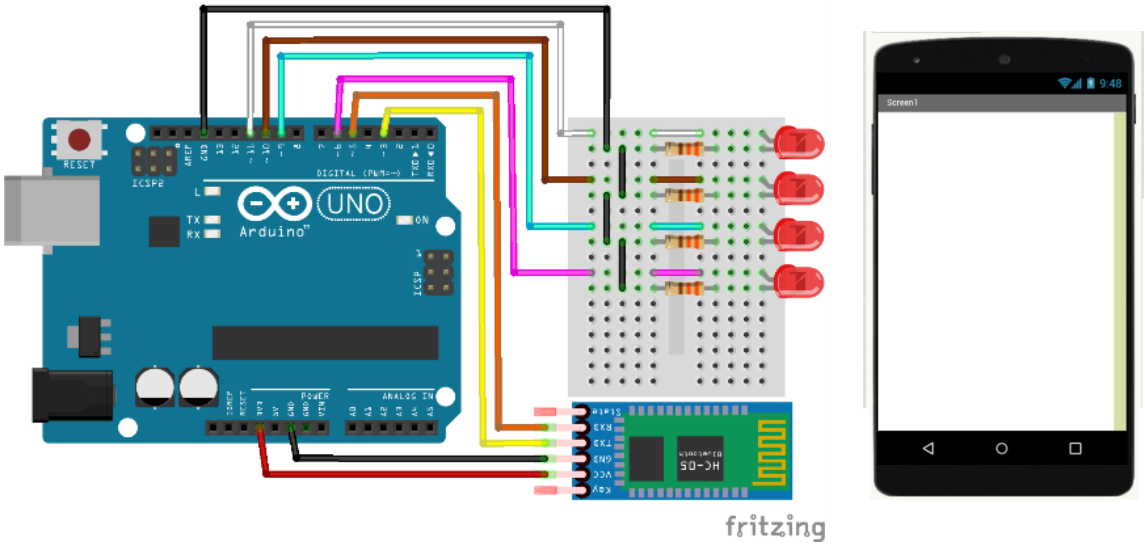


Figura 28 – Controle de acendimento de 4 LEDs mediante controles deslizantes.
Fonte: O autor

Apresenta-se também os detalhes da interface do aplicativo desenvolvido no App Inventor usando controles deslizantes na Figura 29. Na parte superior encontra-se os blocos relacionados à conexão Bluetooth descritos no projeto anterior. Abaixo destes blocos, associados a cada um dos quatro LEDs, foram inseridos um rótulo (*label*), e um controle deslizante (*slider*). Os *labels* são caixas de textos que podem ser alteradas para tornar visíveis informações necessárias. Os Sliders são barras com um botão deslizante em seu centro, que pode variar de uma extremidade a outra representando valores entre um mínimo e máximo. Neste caso, os slides foram configurados para representar valores entre 0 e 255, mas inicialmente definidos na posição central 127.

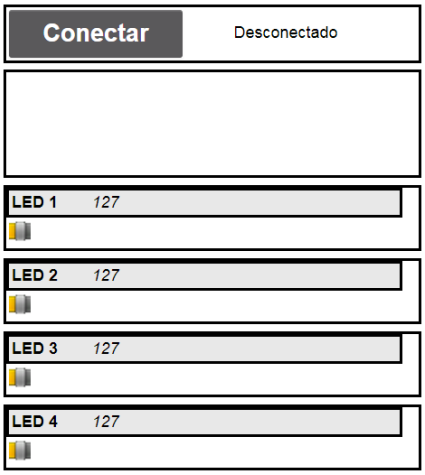


Figura 29 – Design da interface do aplicativo com controles deslizantes. Fonte: O autor

Programação

Em primeiro lugar apresenta-se a programação do módulo Arduino. Assim como em outros projetos anteriores, o programa compreende várias seções. Destacam-se quatro seções: Inicialização, Configuração, Funções auxiliares e Loop Principal. Na seção de inicialização, são criadas as variáveis relativas aos quatro LEDs que serão utilizados. Declara-se também uma variável "c" do tipo char, para receber os caracteres que serão enviados, um de cada vez, e guarda-los na variável do tipo string "valor". O primeiro digito de valor será extraído e 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,
```

Na seção de configuração, setup(), é iniciada a visualização do monitor serial, e definidos os pinos 10, 11, 9 e 6, como pinos de saída. Na seção de funções auxiliares, são definidos as funções tab() e linha() para facilitar a visualização das variáveis no monitor serial.

```
void setup () {
  Serial.begin(9600);
  pinMode (led1, OUTPUT);
  pinMode (led2, OUTPUT);
  pinMode (led3, OUTPUT);
  pinMode (led4, OUTPUT);
}

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

Na última seção, de loop principal, realiza-se a leitura de dados transmitidos caracter por caracter e armazenados na string "valor". O primeiro dígito dessa string é extraído na variável "led" e passa a ser utilizada como forma de decidir qual dos LEDs será acesso. Os outros três dígitos dessa string são armazenados na variável "intensidade" que será utilizada para definir o brilho do LED escolhido.

```
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 ();
     }
  }
```

Apresenta-se também a programação do aplicativo android utilizando o App Inventor mediante blocos de comandos. Dado que a programação de cada componente *slider* é quase idêntica, apresenta-se apenas o bloco de comandos correspondente ao slider 1, denotado como S1, conforme ilustrado na Figura 30.

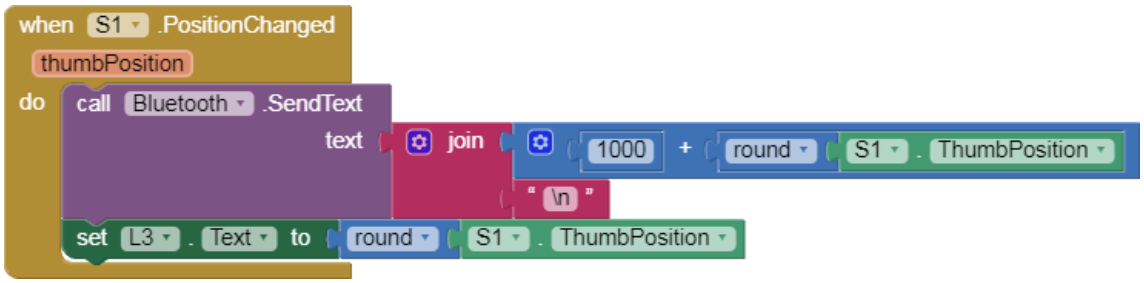


Figura 30 – Bloco de comandos para o Slider 1. Fonte: O autor

Quando a posição do *slider* S1 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 e acrescida do caracter especial fim de linha "\n", que representa o término do envio da informação. Logo após, o texto presente na caixa de texto correspondente ao LED1 é atualizado com o novo valor do *slider* S1. Outros três blocos de comando semelhantes são necessários para controlar os *sliders* S2, S3 e S4. Esses blocos seguem a mesma lógica, mudando-se apenas a referência ao *slider*, o LED e o valor numérico para 2000, 3000 e 4000 respectivamente.

Vale observar que os dados correspondentes a identificação do led e a sua intensidade não são enviados como números inteiros, mas sim como caracteres individuais. Assim, foi necessário que esses dados fossem lidos individualmente e reagrupados em uma String. Depois

foi realizada o recorte da String em duas partes e a posterior transformação de ambos os valores em inteiros.

4.3.8. Controle de dois servomotores mediante um aplicativo com controles deslizantes

Este projeto é uma adaptação do projeto anterior, no qual se substitui parcialmente o uso de LEDs por servomotores. Vale lembrar que no projeto do braço robótico, existem 4 servomotores. No entanto, como forma de transição entre o microprojeto anterior e o objetivo final, optou-se por primeiro fazer o controle de dois LEDs e dois Servomotores. Além disso, observa-se também que o consumo de energia de um servomotor é um pouco maior que o de um LED, sendo assim recomendável uma alimentação externa para o uso de quatro servomotores.

Diagrama

Primeiro ilustramos as conexões entre os componentes utilizados no projeto. O diagrama da Figura 31 ilustra as conexões entre o módulo Bluetooth, o Arduino Uno e 2 Servomotores e 2 LEDs. Enquanto o controle de LEDs e servomotores é feito de maneira remota pelo aplicativo usando controles deslizantes através do Bluetooth.

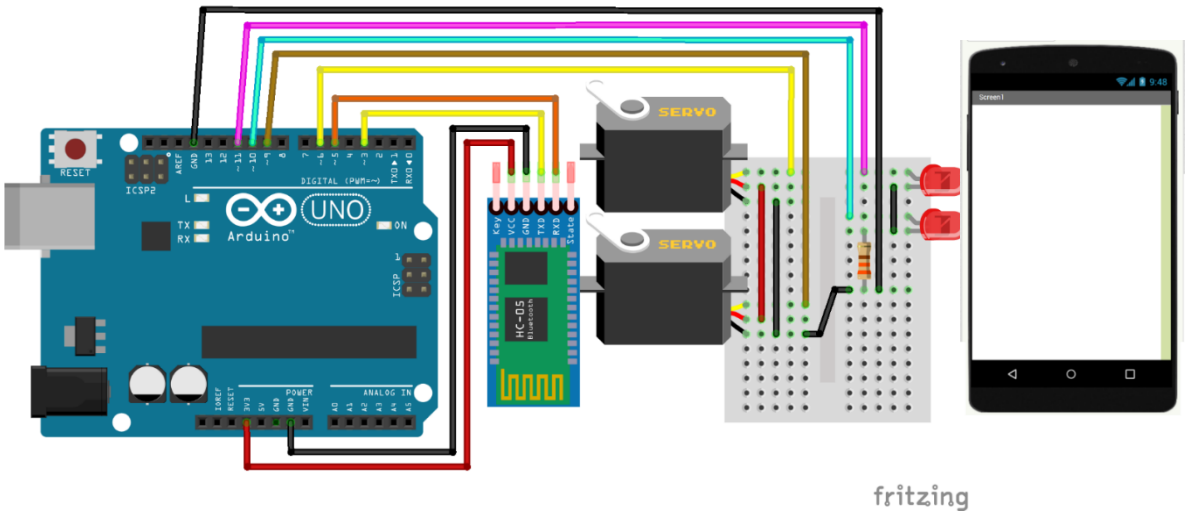


Figura 31 – Controle de Servomotores mediante aplicativo App Inventor. Fonte: O autor

Com relação a interface do aplicativo desenvolvido no App Inventor, mostrada na Figura 32, ela segue o mesmo design do microprojeto anterior, apenas sofrendo alteração nos dois últimos controles deslizantes (*sliders*) que passaram a controlar 2 servomotores. Sendo assim, os *slides* foram configurados para representar valores entre 0 e 180, que representam os ângulos de giro aceitos pelo micro servomotor. Neste caso, define-se como o valor inicial desses *sliders* o valor central de 90 graus.

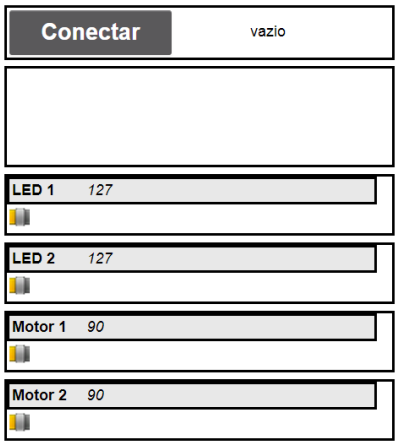


Figura 32 – Design da interface do aplicativo com controles deslizantes 2. Fonte: O autor

Programação

A programação do protótipo Arduino é basicamente idêntica ao projeto anterior, substitui-se os LEDs 3 e 4 pelos servomotores, denominados motor 1 e 2. Os motores utilizam o valor da variável "intensidade" para definir um ângulo de rotação. Os pinos 9 e 6 da placa Arduino são associados aos motores 1 e 2 respectivamente.

```
const int motor1 = 9, motor2 = 6;
```

Na seção de configuração, setup(), os pinos 1 e 9 são definidos como pinos de saída.

```
void setup () {
…
  pinMode (motor1, OUTPUT);
  pinMode (motor2, OUTPUT);
…
}
```

No loop principal, a rotação dos servomotores é realizada usando a função analog Write().

```
void loop () {
…
    switch (led) {
…
      case 3: analogWrite (motor1, intensidade); break;
      case 4: analogWrite (motor2, intensidade); break;
…
    }
…
}
```

A programação do aplicativo Android segue o mesmo padrão do microprojeto anterior, apenas tendo como alteração os dois últimos Sliders que passaram a controlar 2 motores. Todos os blocos de controle são exatamente iguais aos blocos do *back-end* do microprojeto anterior, assim, para evitar repetição, não serão mostrados.

Nos testes realizados neste projeto foi observado o seguinte problema. Embora o controle funcionasse corretamente quando modificados lentamente, quando a mudança era muito brusca, a conexão bluetooth se perdia. Acredita-se que este comportamento possa ser atribuído à falta de energia recebida pelo Arduino através do cabo USB. Confirmando a necessidade da utilização de uma fonte de energia aprimorada.

5. 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. 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.

6. 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.

7.- 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

8.- Perspectivas de continuidade ou desdobramento do trabalho

Como proposta 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.

9.- 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.

10.- Data e assinatura do bolsista (assinatura digitalizada)

30 de Abril de 2020.

11.- Data e assinatura do orientador (assinatura digitalizada)

30 de Abril de 2020.