**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

APAGAR POSTERIORMENTE \/ \/ \/

1. Etapas propostas no plano de trabalho (justificar o não cumprimento de alguma etapa, se for o caso)
2. Introdução
3. Objetivos
4. Metodologia
5. Resultados e Discussão (em um único tópico ou em tópicos separados)
6. Conclusões (demonstrando que cumpriu o plano de trabalho referente a este relatório)
7. Referências
8. Perspectivas de continuidade ou desdobramento do trabalho, **no caso de solicitação de renovação**
9. Participação em congressos e trabalhos publicados ou submetidos e outras atividades acadêmicas e de pesquisa
10. Data e assinatura do bolsista (assinatura digitalizada)
11. Data e assinatura do orientador (assinatura digitalizada)

espaço

1. Etapas propostas no plano de trabalho
2. Introdução
3. Objetivos
4. Metodologia
   1. Componentes
      1. Arduino Uno
      2. Protoboard
      3. Cabo USB
      4. Jumpers
      5. Botão
      6. Resistores
      7. LEDs
      8. Potenciômetros
      9. Joystick
      10. Micro Servo motor 9g
      11. Módulo Bluetooth HC-05
   2. MIT App Inventor 2
      1. Criando um projeto
      2. Componentes Visuais (Front-End)
      3. Blocos de Programação (Back-End)
   3. [Microprojetos](#microprojetos)
      1. [Header](#header)
      2. Ligar os quatro LEDs de acordo com a posição de dois potenciômetros representando as coordenadas x e y (potenciômetro posteriormente substituído por um joystick);

Discussão

Diagrama

Programação

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

Discussão

Diagrama

Programação

* + 1. Variar a angulação do servo motor de acordo com a posição de um potenciômetro;

Discussão

Diagrama

Programação

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

Discussão

Diagrama

Programação

App

*Front*

*Back*

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

Discussão

Diagrama

Programação

App

*Front*

*Back*

* + 1. Uso do módulo Bluetooth para controlar a luminosidade de um LED com app específico desenvolvido no MIT App Inventor 2;

Discussão

Diagrama

Programação

App

*Front*

*Back*

* + 1. Uso de um app desenvolvido no App Inventor para testar as formas de passagem de informação através do módulo Bluetooth e a forma de recepção desses dados pelo Arduino;

Discussão

Programação

App

*Front*

*Back*

* + 1. Uso do módulo Bluetooth para controlar a luminosidade de quatro LEDs;

Discussão

Diagrama

Programação

App

*Front*

*Back*

* + 1. Uso do módulo Bluetooth para controlar a angulação de quatro Servo motores;

Discussão

Diagrama

Programação

App

*Front*

*Back*

* + 1. Aplicar a programação do microprojeto 9 no projeto.

1. Resultados e Discussão (em um único tópico ou em tópicos separados)
2. Conclusões (demonstrando que cumpriu o plano de trabalho referente a este relatório)
3. Referências
4. Perspectivas de continuidade ou desdobramento do trabalho, **no caso de solicitação de renovação**
5. Participação em congressos e trabalhos publicados ou submetidos e outras atividades acadêmicas e de pesquisa
6. Data e assinatura do bolsista (assinatura digitalizada)
7. Data e assinatura do orientador (assinatura digitalizada)

APAGAR POSTERIORMENTE /\ /\ /\

**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 concluído as etapas a) e b) e parcialmente as 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, ArduinoUNO 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 motor, 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.

Como a bolsista anterior já realizou pesquisas sobre formas de integrar os microcontroladores NodeMCU, Arduino UNO e Raspberry Pi, assim como foi executado um projeto simples na plataforma Raspberry Pi, não foi visto necessidade de repetir essas mesmas etapas.

**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.

[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 ou 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. No dia a dia tem se tornado mais frequente o uso da tecnologia Bluetooth para enviar informações e conectar dispositivos, o que torna muito mais acessível, por exemplo, ouvir músicas em fones de ouvido sem fio. Mas essa tecnologia não se limita a isto.

Nas indústrias, com o 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 um mesmo movimento 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 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?]

**4. - 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. 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 as funcionalidades necessárias visadas, que são a de conseguir controlar simultaneamente 4 diferentes motores utilizando. 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):

**4.1.1. – Componentes**

4.1.1. -Arduino uno

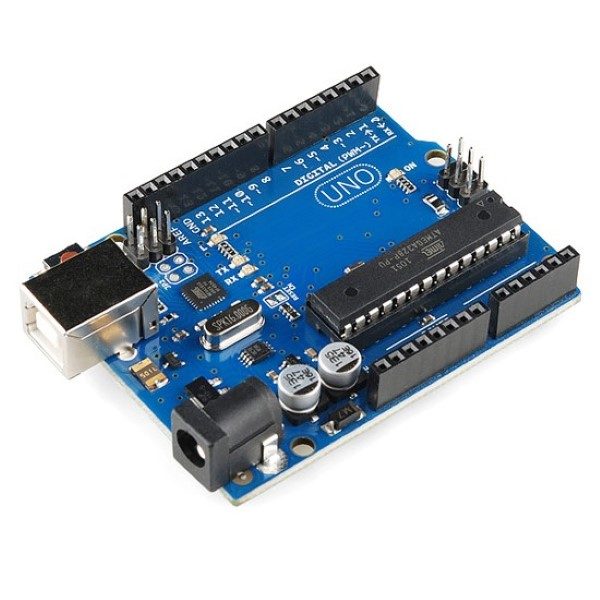


Figura : Arduino UNO - Fonte [1]

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. Cabo usb



Figura : 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. Protoboard (Breadboard)

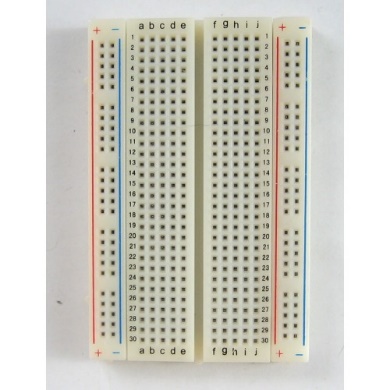


Figura : Protoboard - Fonte [3]

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. Jumpers



Figura : 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 (Switch)



Figura : Botões - Fonte [5]

Componente que mantém aberto (sem passagem de eletricidade) o circuito em que está conectado, fechando-o quando pressionado.

1. Resistores de 330 Ω

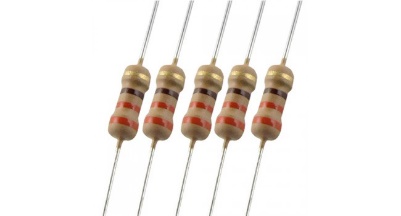


Figura : Resistores - Fonte [6]

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.

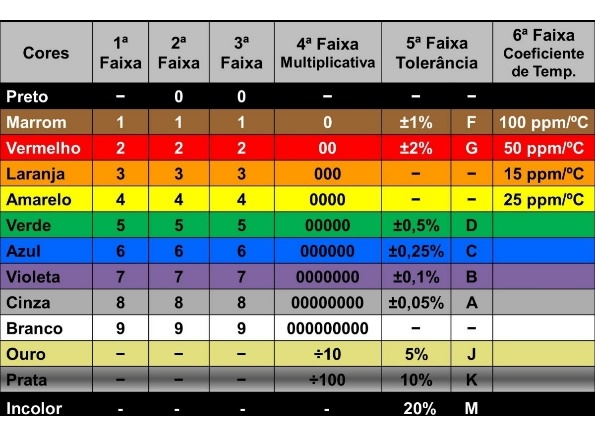


Figura : Código Colorido Resistores - Fonte [7]

1. LED (Light-Emitting Diode – diodo emissor de luz)



Figura : LEDs - Fonte [8]

Componente utilizado para converter energia elétrica em energia luminosa. Pode apresentar diferentes luminosidades de acordo com a programação no Arduino.

1. Potenciômetros



Figura : Potenciômetro - Fonte [9]

São resistores que variam a sua resistência à medida em que o seu regulador variável é girado. O seu pino central retorna um valor que varia entre 0 e 1023 de acordo com a posição de rotação do eixo.

1. Joystick



Figura : 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 plástica móvel está localizada.

1. Micro Servomotor 9g



Figura : 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 rs232 hc-05

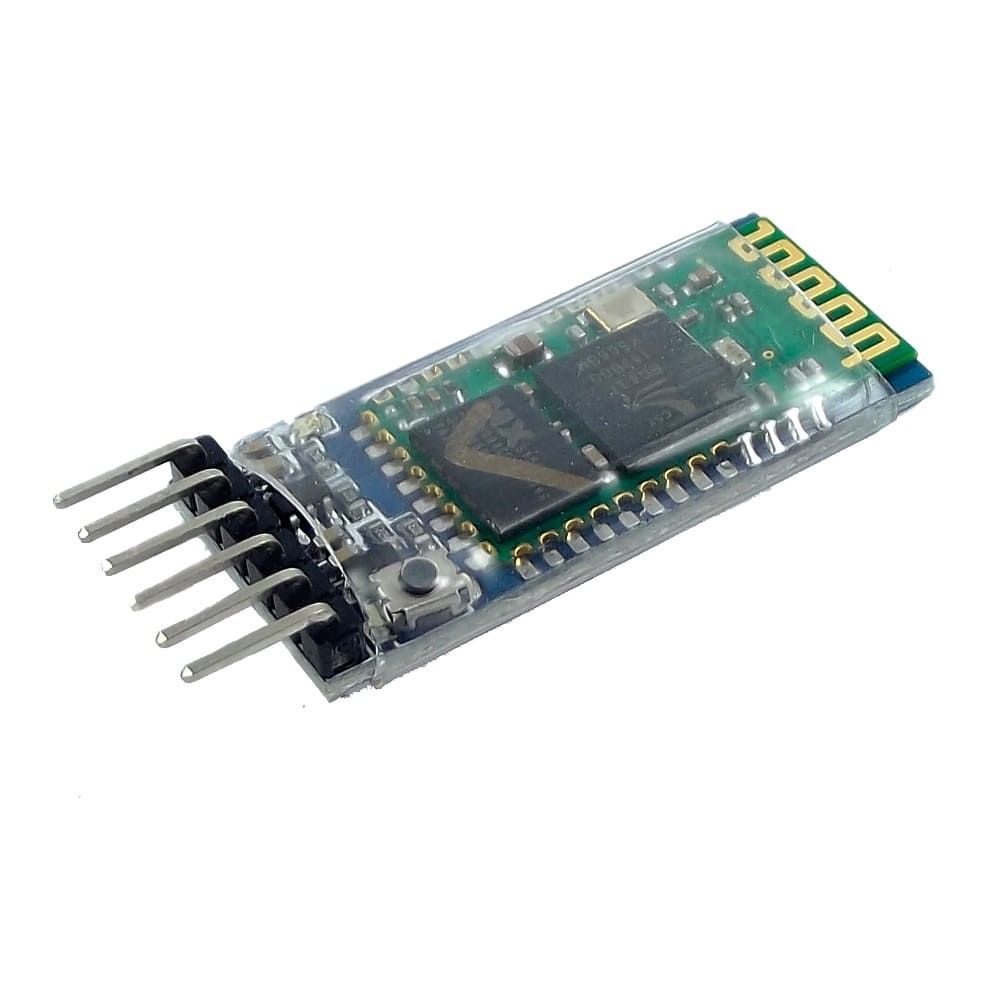


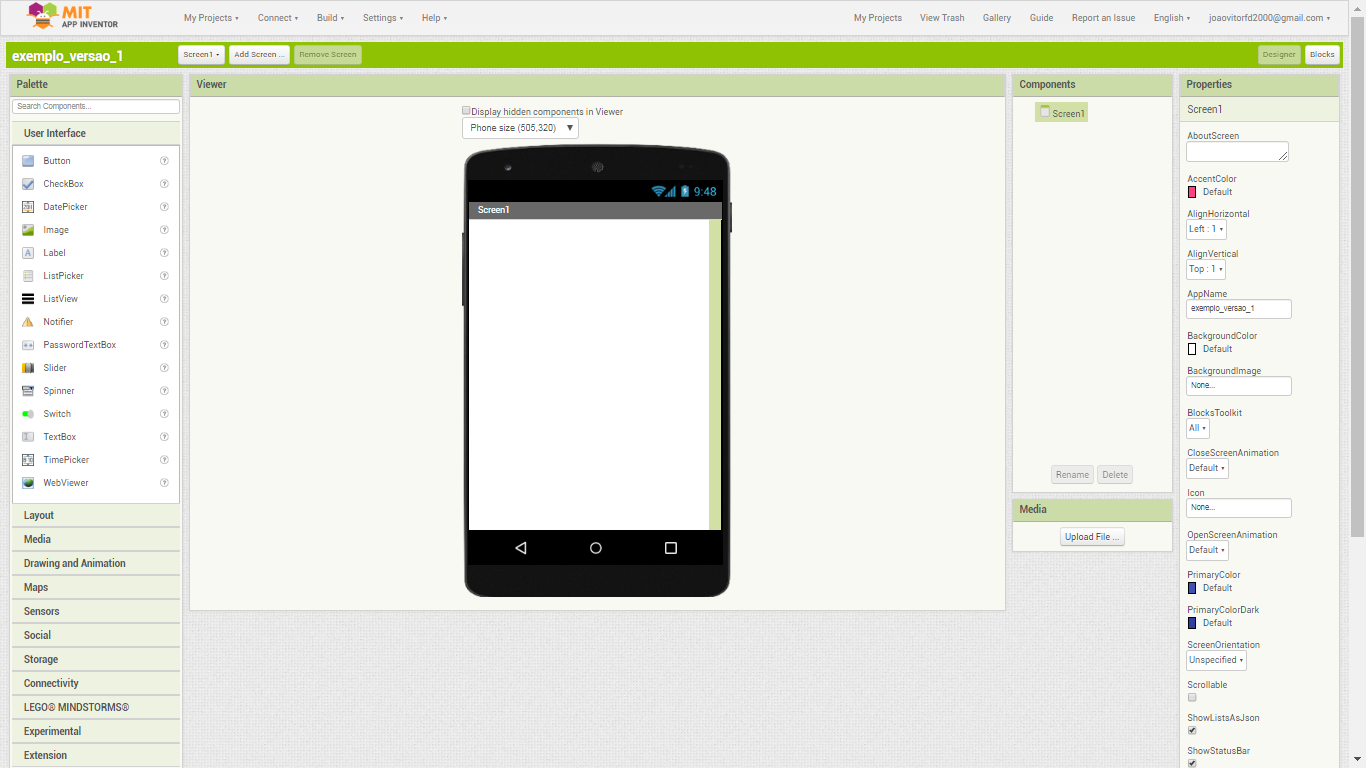
Figura : HC-05 - Fonte [12]

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.

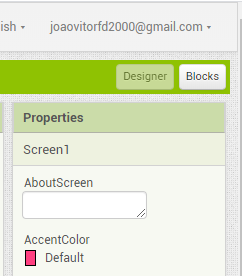
1. MIT App Inventor 2

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.

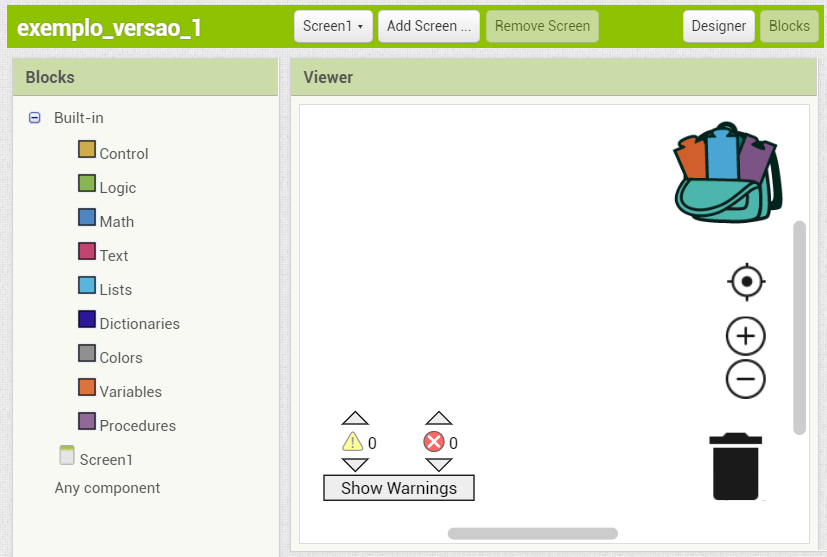
A página inicial de desenvolvimento do app nessa plataforma é a página de desenvolvimento visual (Designer).



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”:



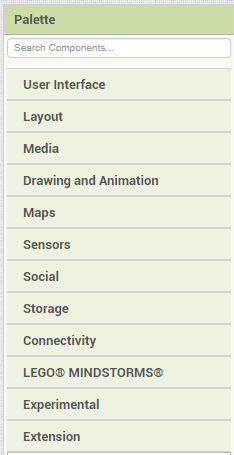
No App Inventor, tanto a parte de desenvolvimento visual (Front-End) quanto a parte de desenvolvimento de retaguarda (Back-End) são montadas utilizando o sistema de “blocos arrasta-e-solta”, ou seja, para utilizar algum bloco de design ou programação no aplicativo, basta arrastá-lo respectivamente, das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

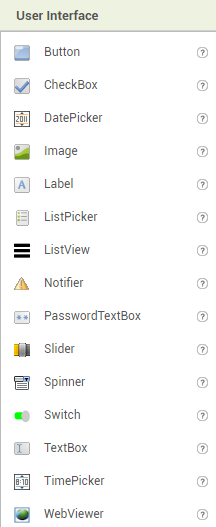
**Palette:**

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.

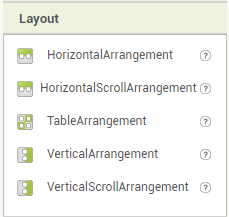


Cada uma dessas categorias contém vários componentes a elas relacionados. Como exemplos temos:

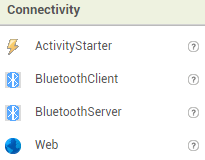
Em User Interface, encontram-se diversas formas de interação direta entre o app e o usuário, sendo alguns deles o Botão, o Slider e a Caixa de Texto;



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

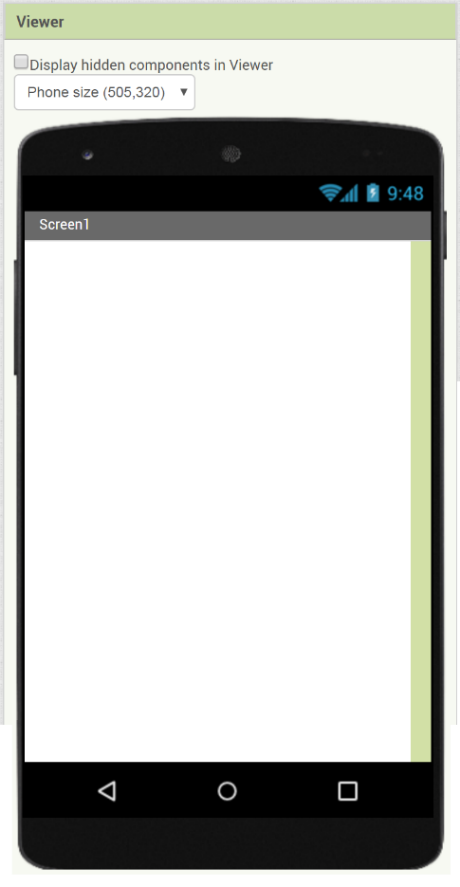


Em Conectividade, estão algumas funcionalidades não visíveis ao usuário, mas que permitem, por exemplo, que o aplicativo tenha acesso a conexão Bluetooth.



**Viewer:**

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



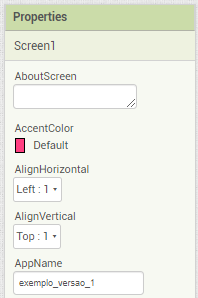
**Components:**

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.



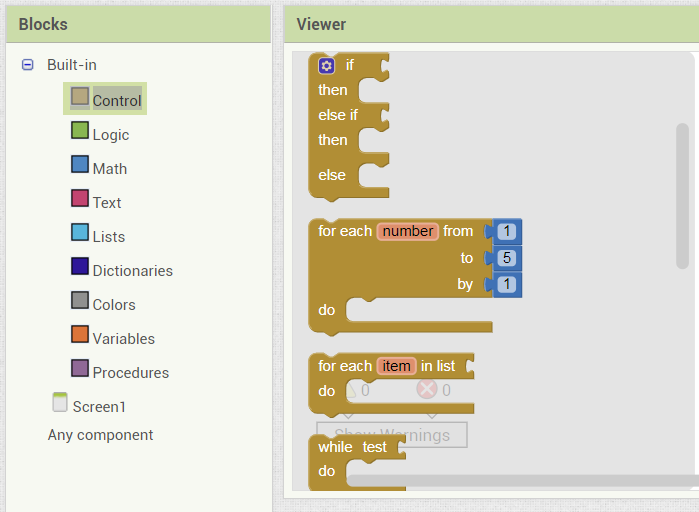
* 1. Blocos de Programação (Back-End)

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

**Blocks**:

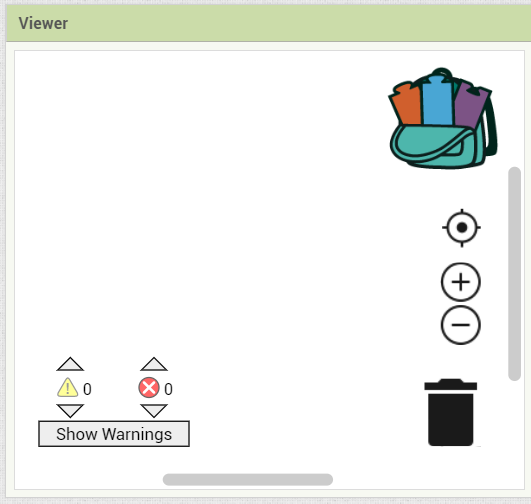
É onde estão organizados diversos componentes divididos por categorias, dentre eles: Control (Controle de fluxo), Math (Matemática) e Variables (Variáveis).

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.

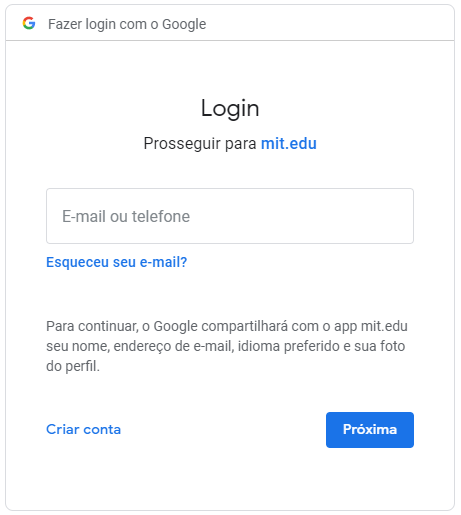
* 1. Criando um projeto

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

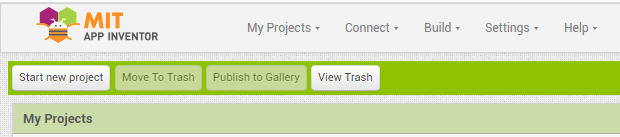
Na página principal do App Inventor (<http://appinventor.mit.edu/>) clique em “Create Apps!”



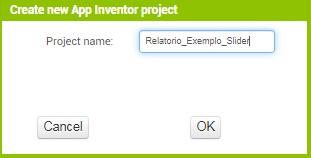
Faça Login com sua conta do Google



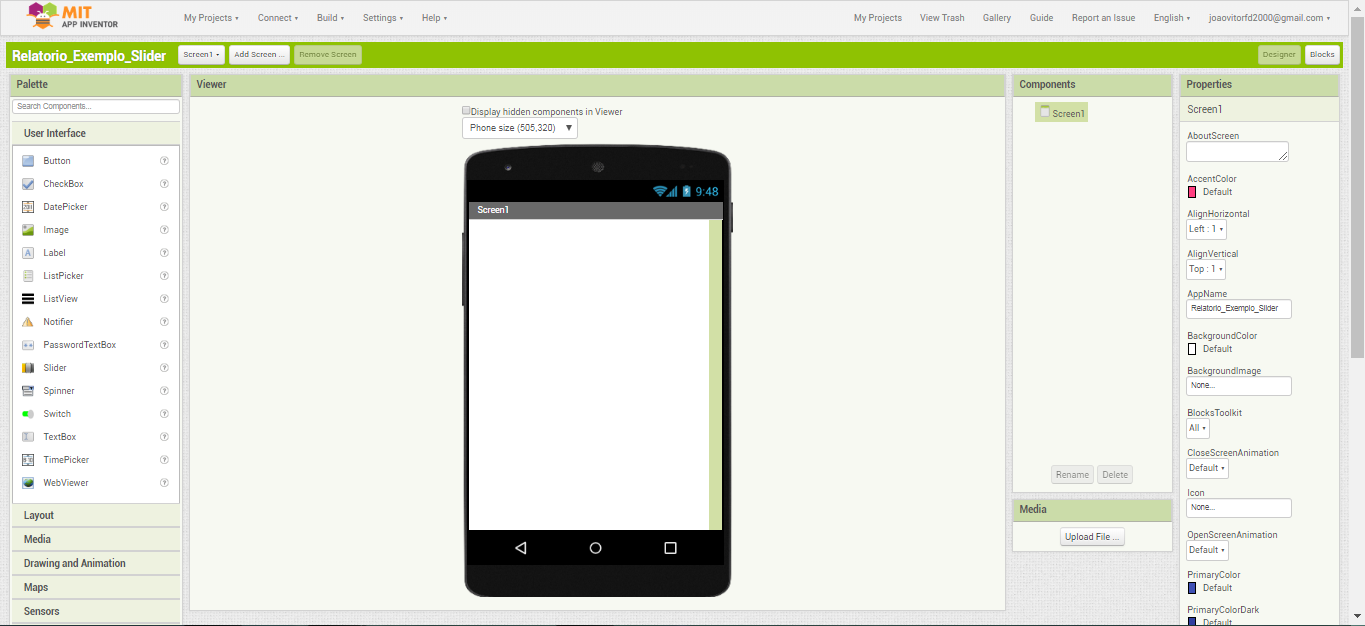
Clique em “Start new project”



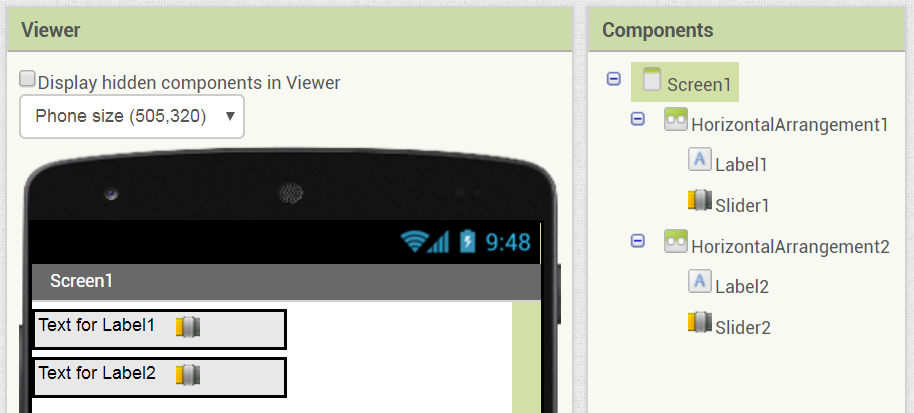
Escolha um nome para o projeto e clique “OK”



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).

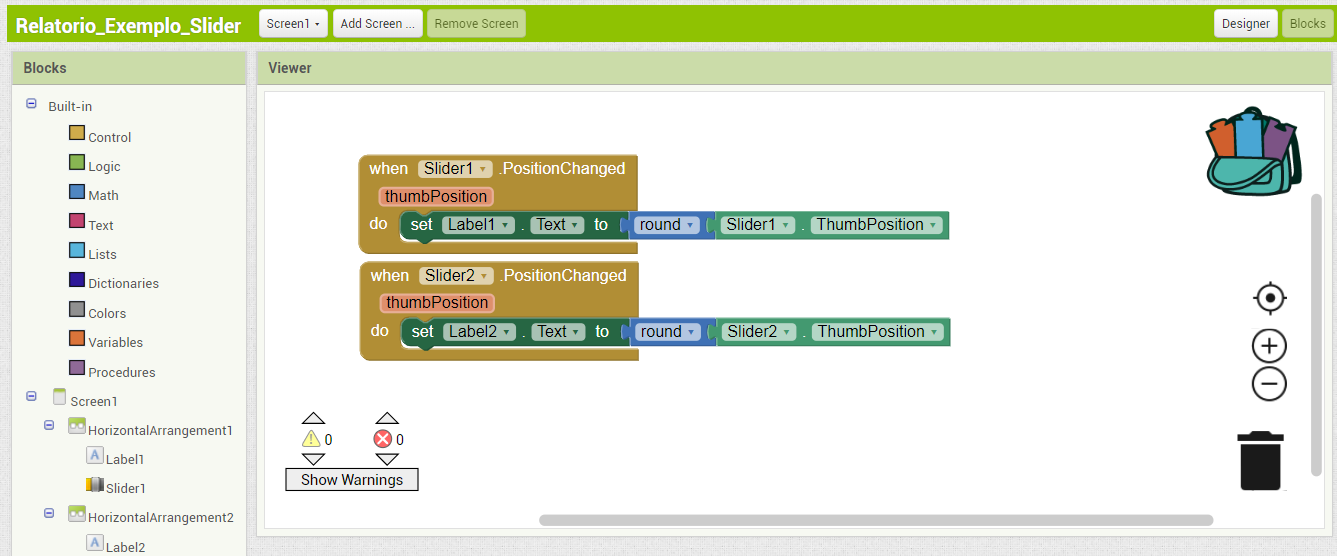
Label1 e Label2: Tornar texto em negrito, definir o texto como o valor inicial do Slider1 (126), definir a largura suficiente para que caiba o texto (30 pixels), alinhar o texto ao centro.

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:

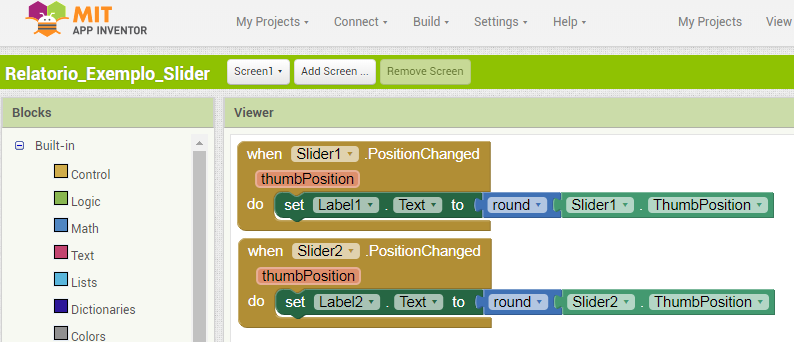


Quando a posição do botão deslizante do Slider1 for alterado, definir o texto do Label1 como sendo o número da posição atual do Slider1 como número inteiro.

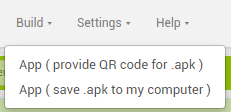


Quando a posição do botão deslizante do Slider2 for alterado, definir o texto do Label2 como sendo o número da posição atual do Slider2 como número inteiro.

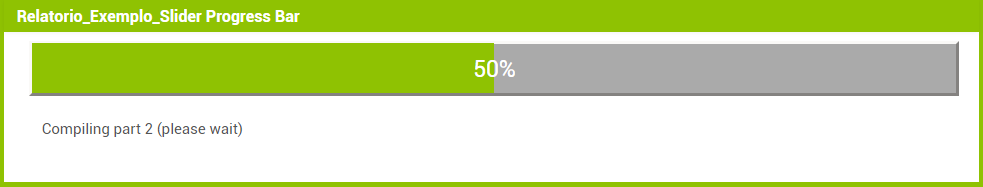
Para concluir o projeto e torná-lo um aplicativo de fato, basta clicar em Build no topo da página e escolher o formato de preferência.



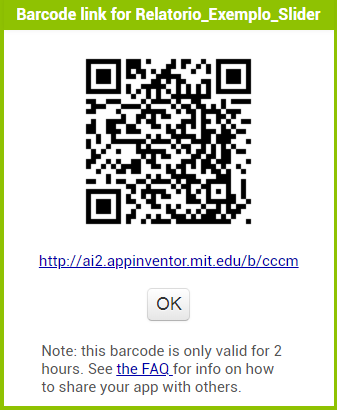
Caso prefira instalar o app através do QR, selecione a primeira opção.



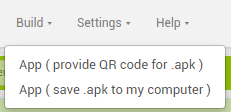
Espere um momento para que a plataforma compile o app.



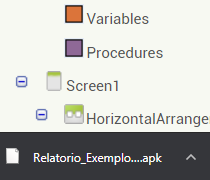
Assim que for compilado, aparecerá na tela um QRCode, a partir daí, basta ler o código QR com um App ou acessar o link diretamente no navegador do seu Smartphone.



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 concluindo um aplicativo de variação de slider, utilizando a plataforma App Inventor.

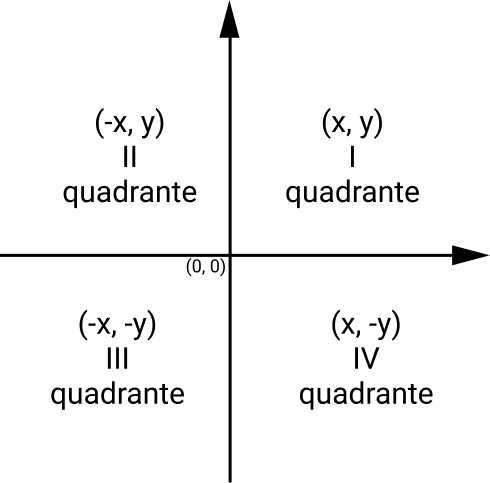
* 1. Microprojetos

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.

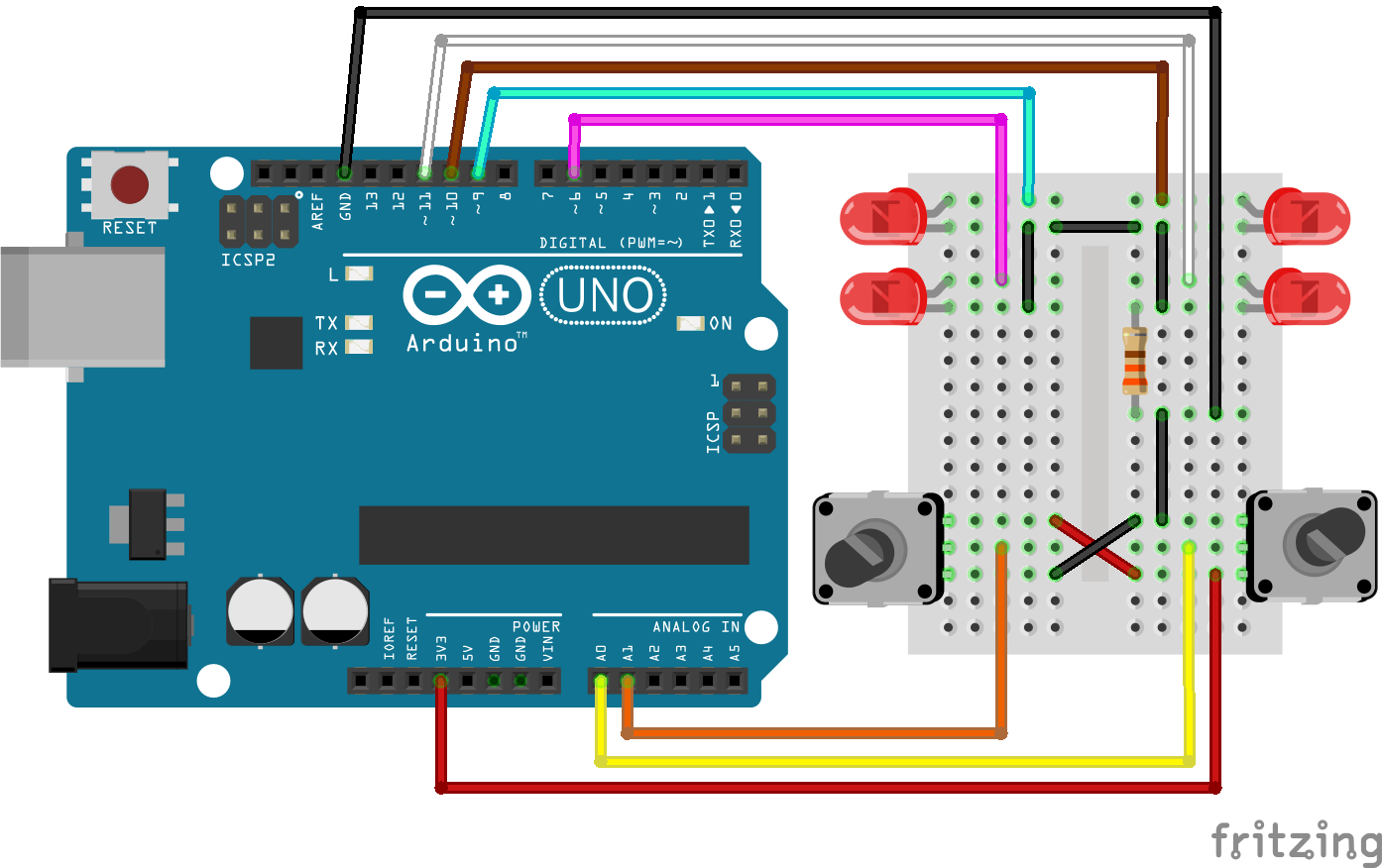
Para tornar o relatório mais conciso, alguns outros microprojetos mais simples do que os que serão citados abaixo foram ocultados do relatório, pois foram simples demais ou apresentaram muita repetição com o que já havia sido feito anteriormente.

* + 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);}

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);

}

}

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.

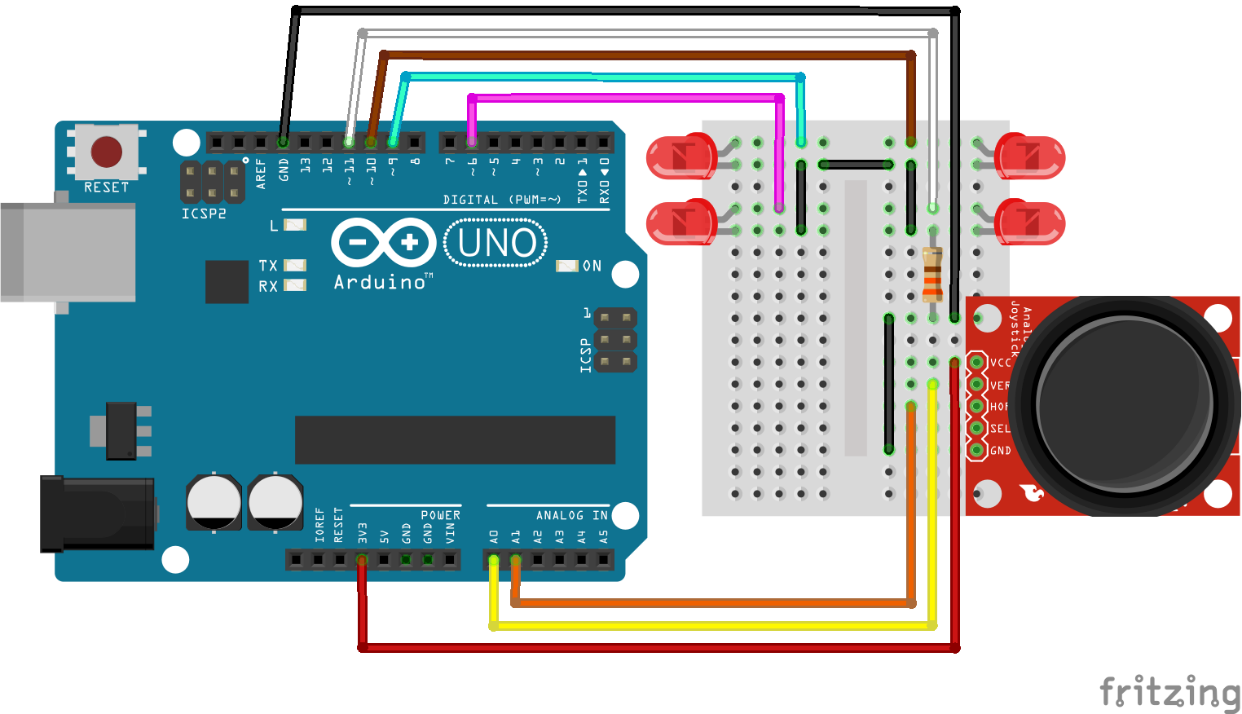
**Potenciômetros substituídos por um joystick:**

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 forem 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;

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);

}

}

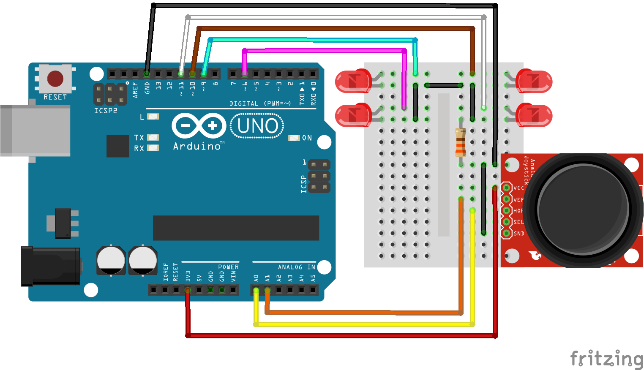
}

* + 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, porém, neste caso, a medida 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

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

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

int EntradaX, EntradaY, MapX, MapY;

int MeioX, MeioY, MaiorMap;

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);}

void loop() {

EntradaX = analogRead(X);

EntradaY = analogRead(Y);

MapX = (EntradaX>512)? (map(EntradaX, MeioX, 1023, 0, 255)):(map(EntradaX, MeioX, 0, 0, 255));

MapY = (EntradaY>512)? (map(EntradaY, MeioY, 1023, 0, 255)):(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);

}

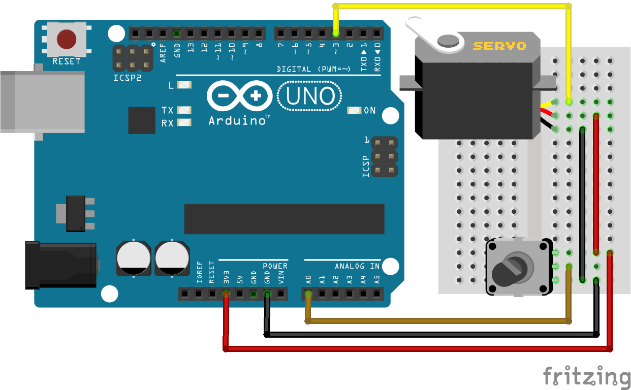
}

}

* + 1. Variar a angulação do servo motor 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

#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);

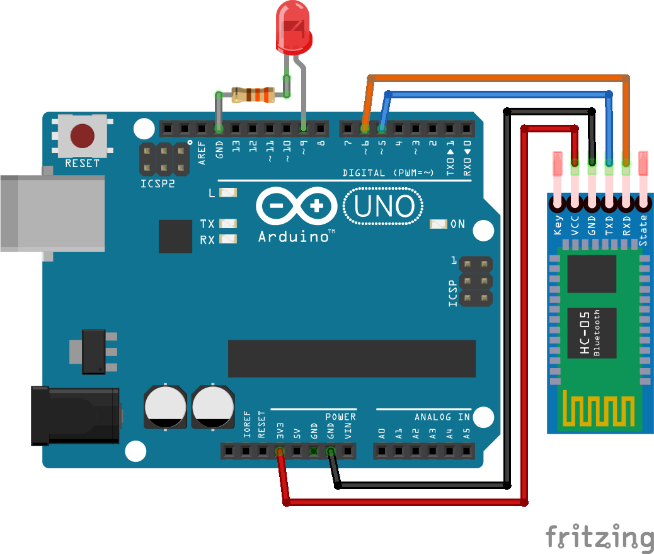
}

* + 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

#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);

}

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

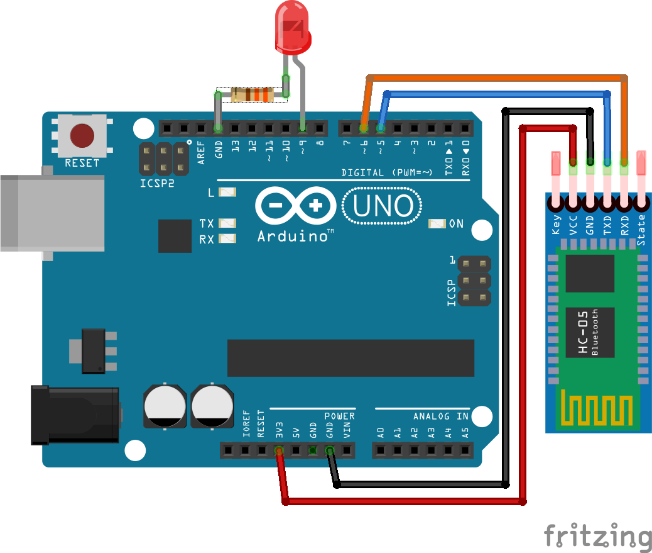
Similar ao microprojeto anterior, desta vez ainda foi utilizado um aplicativo disponível na Google Play Store. Desta vez, utilizando o sinal analógico ao invés do digital, o que permite o controle da luminosidade através do envio de valores entre 0 e 255 do app para o módulo, porém o aplicativo sendo 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 era possível o objetivo de alterar a luminosidade do LED, mas a responsividade não estava de acordo com o esperado.

Diagrama



Programação

#include <SoftwareSerial.h>

SoftwareSerial bluetooth(5, 6);

// TX, RX do HC-05

const int led = 9;

int dimmer;

String stringGeral;

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

void loop()

{

if (Serial.available())

{

stringGeral = String("");

while (Serial.available())

{

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

delay(1);

}

dimmer = stringGeral.toInt();

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

analogWrite(led, dimmer); }

}

}

* + 1. Uso de um app desenvolvido no App Inventor para testar as formas de passagem de informação através do módulo Bluetooth e a forma de recepção desses dados pelo Arduino;

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.

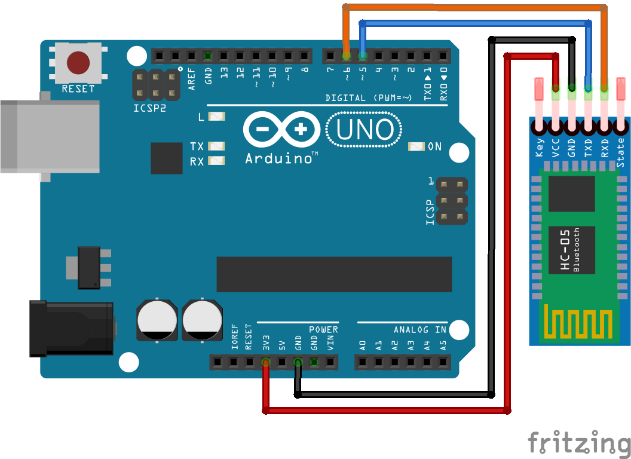
Foram testadas várias formas de envio de informação, assim como várias formas diferentes de recepção de informação.

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” no fim dos valores enviados, caractere ASCII que representa uma nova linha. Aparentemente o Arduino, quando lê o valor da variável recebida pelo aplicativo, não apenas lê, ele 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 envio de 1 byte que estava sendo utilizado no App Inventor para o envio de texto, e no envio de texto 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: HC05\_testes\_transmissao\_valores\_separados

#include <SoftwareSerial.h>

SoftwareSerial bluetooth(5, 6);

// TX, RX do HC-05

String valor, parte1, parte2;

char c;

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

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

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

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*

*Back*

* + 1. Uso do módulo Bluetooth para controlar a luminosidade de quatro LEDs com Slider;

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.

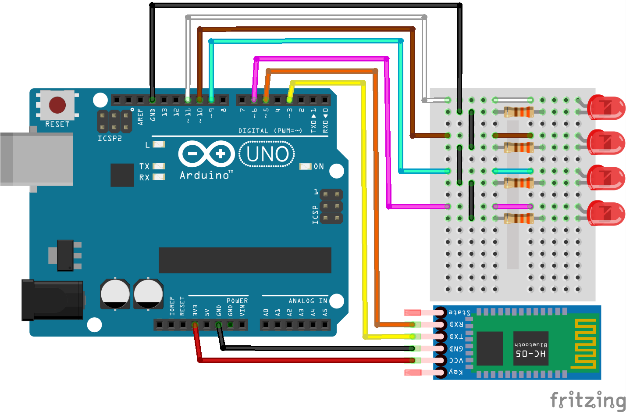
A segunda alternativa seria utilizar a casa do milhar como digito diferenciador, assim, quando o algarismo na casa do milhar for 2, o LED 2 receberia o valor de sua intensidade, no caso do valor “2200” o LED 2 teria o seu brilho em 200, e assim para os outros valores. O benefício dessa alternativa seria 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 enviados 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 assim fossem feitas 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-se 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, porém, 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

#include <SoftwareSerial.h>

SoftwareSerial bluetooth(3, 5);

// TX, RX do HC-05

const int led1 = 11, led2 = 10;

const int led3 = 9, led4 = 6;

int valorint, led, intensidade,

char c;

String valor;

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);

}

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*

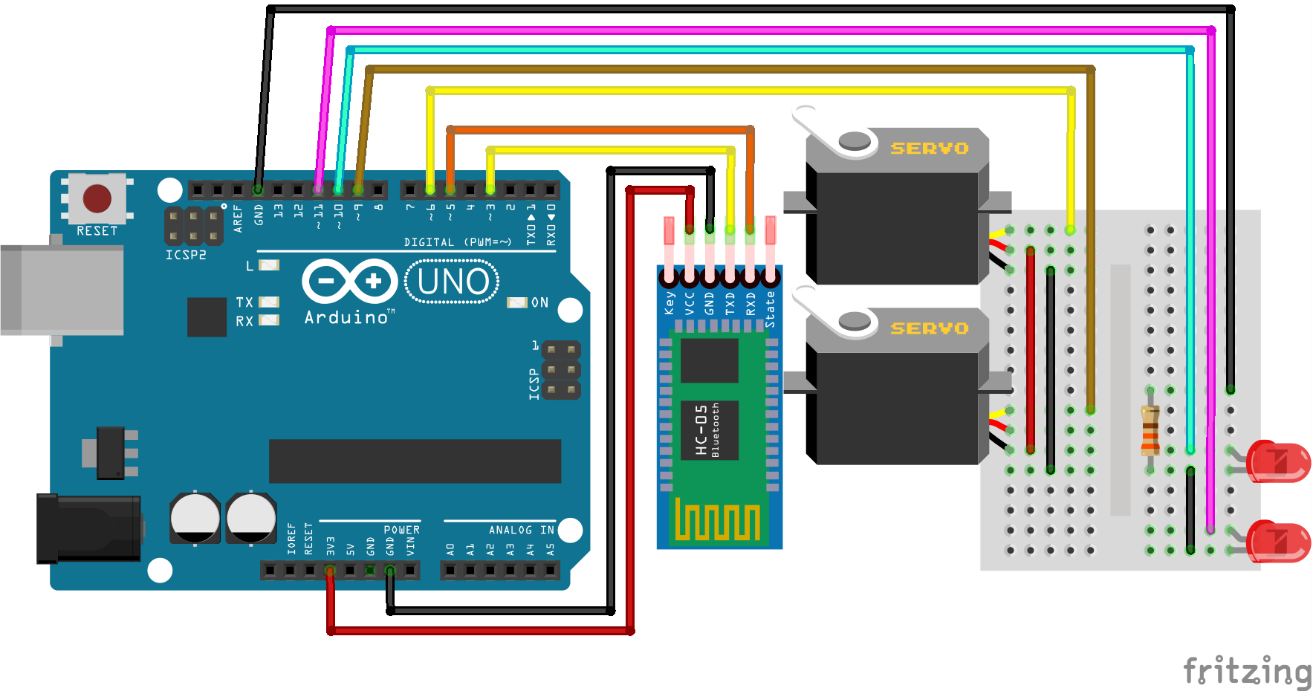
*Back*

* + 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

#include <SoftwareSerial.h>

SoftwareSerial bluetooth(3, 5);

// TX, RX do HC-05

const int led1 = 11, led2 = 10;

const int motor1 = 9, motor2 = 6;

int valorint, led, intensidade;

char c;

String valor;

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);

}

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();

}

}

}

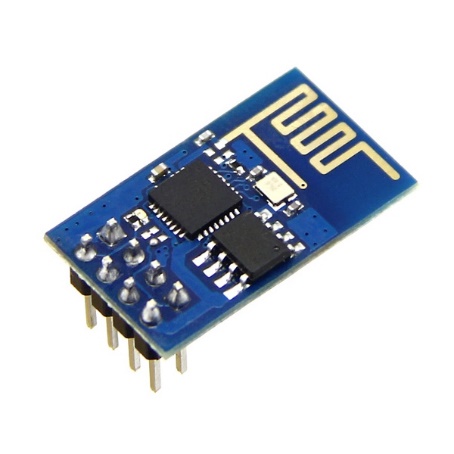
App

*Front*

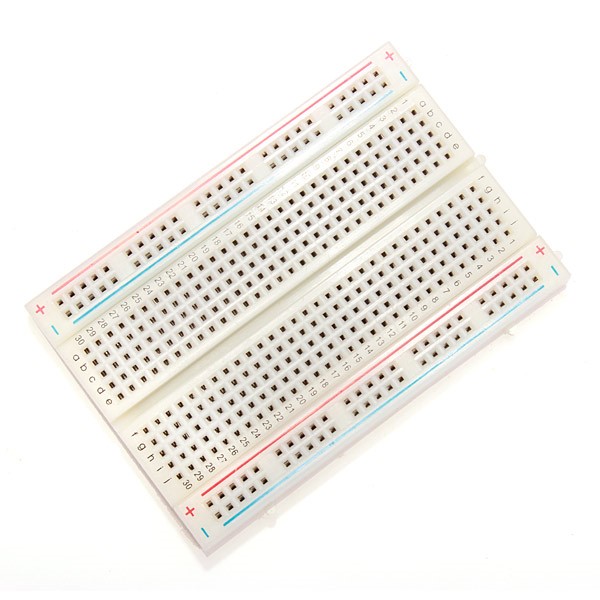
*Back*

1. Braço Robótico

**4.1.1-ESP8266-01**



*Protoboard*



**Figura 1** – ESP8266-01 – Fonte [3]

**Figura 6 –** Protoboard – Fonte [8]

**5.- Resultados e Discussão**

Como resultado desta pesquisa, além dos estudos e testes relacionados aos diversos componentes disponíveis para conexão com a internet e do estudo das linguagens de programação necessárias para o desenvolvimento de aplicações web, foi desenvolvido um projeto para Internet das Coisas. O projeto escolhido, denominado Plata IoT consiste em medir a umidade do solo de uma planta, a fim de saber quando é necessário regá-la, e é dividido em duas partes. Na primeira parte é possível saber quando o solo está seco demais, através dos dados enviados para uma plataforma online, que por sua vez gera um *tweet* avisando que a planta precisa ser regada. A segunda parte traz o desafio de fazer a planta ser regada sozinha sempre que necessário, utilizando uma válvula que permite ou não a passagem de água. Paralelamente ao desenvolvimento desse projeto, teste foram feitos para descobrir qual era a melhor maneira de controlar o braço robótico, desenvolvido no primeiro ano de pesquisa. Conclui-se que é utilizando o NodeMCU, assim como foi feito no projeto Planta IoT. No entanto, esse dispositivo traz uma limitação, por só ter um único pino analógico, tornou inviável o controle dos quatro servo motores presentes no braço, já que cada um deles exige um pino analógico individual. Devido ao pouco tempo hábil para contornar essa situação, foi dada preferência ao projeto principal, que, por sua vez, foi concluído com sucesso, permitindo com que o objetivo desse trabalho, de controlar dispositivos via internet, fosse concluído.

**5.1.- Protótipo de monitoramento de uma Planta IoT**

Nesta seção serão apresentados os detalhes sobre desenvolvimento do protótipo para o monitoramento de uma planta através da internet.

Este projeto de monitoramento utiliza a plataforma NodeMCU baseada no microcontrolador ESP8266-12E. O módulo NodeMCU é conectado a um sensor de umidade do solo, que faz leituras periódicas, calcula a umidade em termos percentuais e envia essa informação via rede *WiFi*, a cada trinta segundos, para um servidor online, chamado *ThingSpeak*. Esse servidor é um tipo de plataforma IoT, que oferece serviços de forma gratuita, permitindo o armazenamento de dados numéricos mediante *upload*, assim como a execução de alguns serviços. Tais dados ficam armazenados na nuvem, onde então é permitida a visualização dos mesmos em gráficos em tempo real. Ele possui a limitação de que o tempo entre upload de dados deve ser, no mínimo, de 15 segundos. Caso isso for desobedecido, os dados que forem enviados em intervalo de tempo inferior serão ignorados e perdidos.

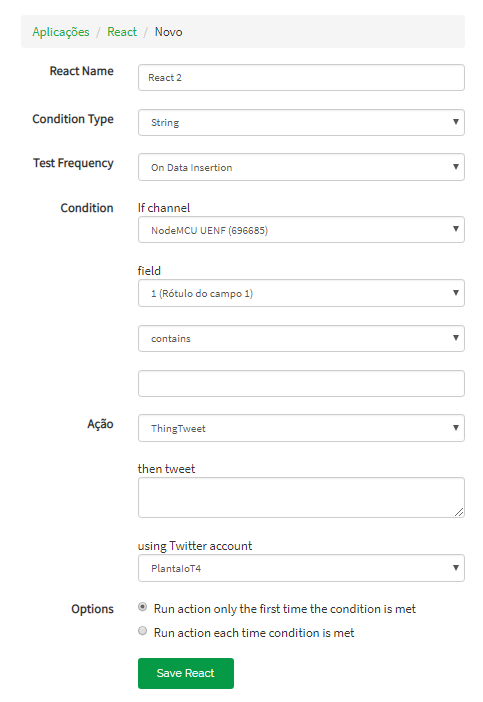
Para enviar dados ao *ThingSpeak* é necessária a criação de uma chave de escrita que será associada ao canal de comunicação utilizado na plataforma. O envio de dados ao servidor é realizado na forma de uma requisição HTTP. Após enviar os dados para a plataforma, o dispositivo avalia o grau de umidade do solo, de maneira que sempre que a planta estivesse precisando de água, isto é, sempre que a porcentagem de umidade do solo esteja abaixo de 60%, é gerado um *tweet* informando a necessidade de regar a planta. O envio do *tweet* é um serviço fornecido por uma aplicação, chamada *React*, disponível no servidor utilizado. Em geral, essa aplicação permite programar a ação que deverá ser tomada, desde que seja atendida uma determinada condição.

*Configurações Iniciais*

Antes de começar o desenvolvimento do projeto, foi preciso configurar o NodeMCU para que ele pudesse ser programado no IDE do Arduino. Para isso, foi necessário instalar o pacote correspondente ao módulo ESP8266 e verificar se o dispositivo estava conectado na porta serial correta.

Após essa etapa, foi criada uma conta na plataforma *ThingSpeak*, para que uma chave de escrita pudesse ser gerada e um canal de comunicação definido. Assim, a chave criada serve para direcionar para qual canal os dados devem ser enviados. Por último, foi necessário criar uma conta no *Twitter* para representar a Planta IoT e configurar a aplicação *React* dentro do servidor *ThingSpeak.* A configuração do *React* segue as seguintes etapas:

1. Fazer login na plataforma *ThingSpeak* com a conta criada inicialmente;
2. Na página principal, clicar em Aplicações, e logo em seguida em *React*, conforme ilustrado na Figura 10;
3. Clicar em “New React” e preencher os dados de acordo com a Figura 11;
4. Salvar a aplicação clicando em “Save React”.



**Figura 11 –** Configurar tweets no ThingSpeak – Fonte: A autora

A apresentação do desenvolvimento deste protótipo é dividida nas seguintes seções: i) Componentes utilizadas, ii) Diagrama do dispositivo iii) Código de controle para o microcontrolador e iv) Protótipo do dispositivo.

*Componentes Utilizados*

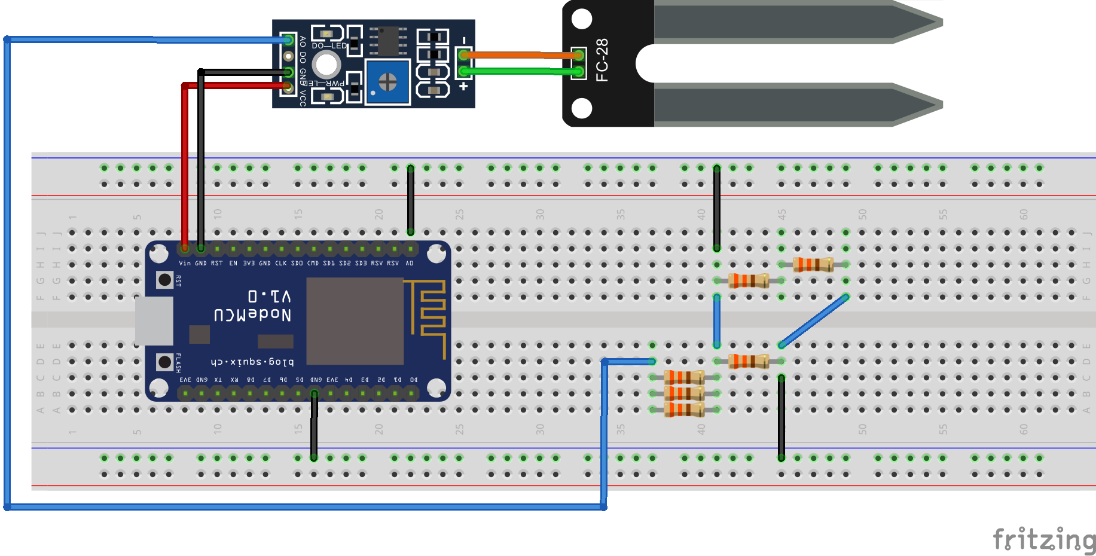
Foram utilizados os seguintes componentes, descritos na sessão anterior:

1. NodeMCU;
2. Jumpers;
3. Resistores de 330 ohms;
4. Protoboard;
5. Sensor de Umidade do Solo (Higrômetro).

*Diagrama de Conexão*

O diagrama de conexão entre as componentes mencionadas, definem o circuito de controle. A figura 12 ilustra o diagrama correspondente. A alimentação de energia é feita pelo cabo USB. Os dados do sensor higrômetro são lidos através do único pino analógico do NodeMCU (A0), e posteriormente convertidos em unidades percentuais (0 – 100%). Para realizar esta conexão, foi necessário usar um divisor de tensão, pois a tensão de saída do higrômetro (linha azul) fornece 5V, enquanto a entrada analógica (A0) do NodeMCU aceita no máximo 3,3V. O diagrama original exigia um resistor de 100 ohms e um de 200 ohms, mas como não haviam esses resistores disponíveis, foi feita uma adaptação. Para substituir o resistor de 100 ohms, foram usados três de 330 ohms arranjados de forma paralela, de forma a obter um valor aproximado. Algo parecido foi feito para substituir o resistor de 200 ohms, mas dessa vez foi necessário dispor dois resistores de 330 ohms de forma serial e um de 330 paralelo a esses dois, obtendo um valor de resistência próximo a 220 ohms, o que foi suficiente nesse circuito.

Vale mencionar que o sensor higrômetro possui um potenciômetro (pequeno quadrado azul) que permite ajustar a sensibilidade do mesmo.



**Figura 12 –** Diagrama de Conexão para

Monitoramento da Planta IoT – Fonte: A autora

*Código de Controle*

Como mencionado anteriormente, o circuito para o monitoramento da umidade do solo é controlado mediante uma placa NodeMCU que possui um módulo ESP8266 com capacidade de conexão à internet através da tecnologia *WiFi*.

O código de controle para este circuito realiza um conjunto de tarefas de configuração e controle. Na parte de configuração, define-se o nome da rede local que será utilizada assim como o endereço da plataforma de IoT, que funcionará como um servidor que publicará as informações sobre a umidade do solo. Na parte de controle, o dispositivo realiza medições periódicas sobre a umidade do solo, em intervalos de 30 segundos, e envia essa informação à plataforma de IoT, para que de acordo com o nível de umidade, caso seja necessário se envie uma mensagem de alerta via *Twitter* informando-se a necessidade de regar a planta. O código utilizado é uma adaptação do código proposto pelo site da FilipeFlop [12]. Alguns comentários foram introduzidos no código para facilitar a compreensão.

Na parte inicial deste código, mostrada no Quadro 1, primeiramente inclui-se a biblioteca <ESP8266Wifi.h> que disponibiliza as funções necessárias para a conexão do módulo ESP8266 à rede *WiFi.* Define-se o nome da rede local que será utilizada assim como a respectiva senha. Além disso, especifica-se o intervalo de tempo para o envio de dados de umidade do solo à plataforma de IoT, definido em 30.000 milissegundos. Define-se também o endereço da plataforma de IoT, que corresponde a plataforma *ThingSpeak*, junto a sua chave de escrita gerada com antecedência. Além disso, define-se variáveis para medição do tempo de conexão e criação do cliente que solicita o serviço a plataforma de IoT. Finalmente, são declarados os protótipos das funções para envio de informações ao servidor, conexão *WiFi* e leitura de umidade do solo.

#include <ESP8266WiFi.h>

//defines

#define SSID\_REDE "Pesquisa CC" //coloque aqui o nome da rede que se deseja conectar

#define SENHA\_REDE "qr5dr1jw" //coloque aqui a senha da rede que se deseja conectar

#define INTERVALO\_ENVIO\_THINGSPEAK 30000//intervalo entre envios de dados ao ThingSpeak (em ms)

//constantes e variáveis globais

char EnderecoAPIThingSpeak[] = "api.thingspeak.com";

String ChaveEscritaThingSpeak = "UC5MNQ5X5G3OIYU3";

long lastConnectionTime;

WiFiClient client;

//prototypes

void EnviaInformacoesThingspeak(String StringDados);

void FazConexaoWiFi(void);

float FazLeituraUmidade(void);

**Quadro 1. -** Código de Controle:Definição da Rede e Plataforma IOT

No Quadro 2, apresenta-se a função **EnviaInformacoesThingspeak()** que inicia uma nova conexão do cliente ao servidor, mediante o comando *client.connect()* e verifica o estado dessa conexão, e em caso de verificação positiva, envia à plataforma *ThingSpeak* a informação sobre umidade de solo, recebida na forma de uma *string*, como uma requisição HTTP. Além disso, a função registra o tempo transcorrido até o momento, desde o início da execução, mediante o comando *millis ()* como forma de registrar o tempo da nova conexão.

void EnviaInformacoesThingspeak(String StringDados)

{

if (client.connect(EnderecoAPIThingSpeak, 80))

{

//faz a requisição HTTP ao ThingSpeak

client.print("POST /update HTTP/1.1\n");

client.print("Host: api.thingspeak.com\n");

client.print("Connection: close\n");

client.print("X-THINGSPEAKAPIKEY: "+ChaveEscritaThingSpeak+"\n");

client.print("Content-Type: application/x-www-form-urlencoded\n");

client.print("Content-Length: ");

client.print(StringDados.length());

client.print("\n\n");

client.print(StringDados);

lastConnectionTime = millis();

**Serial**.println("- Informações enviadas ao ThingSpeak!");

}

}

**Quadro 2. -** Código de Controle:Função de Envio de Dados ao *ThingSpeak*

A função **FazConexãoWiFi()** é mostrada no Quadro 3. Esta função finaliza a conexão do cliente que estiver usando o servidor e inicia uma nova conexão após um segundo de espera. Para isso, inicia uma nova conexão *Wifi*, de acordo com os parâmetros de identificação de rede e senha definidos previamente. Enquanto o estado da conexão não for verificado, aguardará por mais meio segundo e imprimira uma mensagem no monitor serial. Após verificada a conexão, imprime no monitor serial o endereço de IP local do servidor.

void FazConexaoWiFi(void)

{

client.stop();

**Serial**.println("Conectando-se à rede WiFi...");

**Serial**.println();

delay(1000);

WiFi.begin(SSID\_REDE, SENHA\_REDE);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

**Serial**.print(".");

}

**Serial**.println("");

**Serial**.println("WiFi conectado com sucesso!");

**Serial**.println("IP obtido: ");

**Serial**.println(WiFi.localIP());

delay(1000);

}

**Quadro 3.-** Código de Controle:Função de conexão *WiFi*

O Quadro 4 apresenta o código correspondente a função que faz a leitura da umidade do solo a partir do sensor higrômetro. Esse sensor está conectado a uma porta analógica, que no caso do NODEMCU é capaz de registrar tensões DC entre 0 até 3,3V, convertendo essa tensão em um valor numérico na escala de 0 a 978. Vale observar que quanto maior a leitura do sensor menor será a umidade do solo e vice-versa. Com base na leitura do sensor, a função calcula uma medida de umidade em termos percentuais (que pode variar de 0 a 100%), definido pela seguinte expressão:

A função retorna essa medida percentual.

float FazLeituraUmidade(void)

{

int ValorADC;

float UmidadePercentual;

ValorADC = analogRead(0); //978 -> 3,3V

**Serial**.print("[Leitura ADC] ");

**Serial**.println(ValorADC);

//Quanto maior o numero lido do ADC, menor a umidade.

//Sendo assim, calcula-se a porcentagem de umidade por:

//

// Valor lido Umidade percentual

// \_ 0 \_ 100

// | |

// | |

// - ValorADC - UmidadePercentual

// | |

// | |

// \_|\_ 978 \_|\_ 0

//

// (UmidadePercentual-0) / (100-0) = (ValorADC - 978) / (-978)

// Logo:

// UmidadePercentual = 100 \* ((978-ValorADC) / 978)

UmidadePercentual = 100 \* ((978-(float)ValorADC) / 978);

**Serial**.print("[Umidade Percentual] ");

**Serial**.print(UmidadePercentual);

**Serial**.println("%");

return UmidadePercentual;

}

**Quadro 4.-** Código de Controle:Função de Leitura de Umidade

Uma vez descritas todas as funções do código de controle, resta apresentar o programa principal de controle. Este programa é composto basicamente por duas funções: A função **setup()** que é executada uma única vez e serve para configuração e inicialização dos dispositivos e a função **loop() ,** que é de fato o programa principal de controle, e é executado de forma continua e ininterrupta. Ambas as funções são mostradas no Quadro 5.

Na função **setup()**, primeiro inicializa-se o monitor serial da IDE Arduino, que poderá ser utilizado para mostrar mensagens referentes ao estado corrente da aplicação. Depois chama-se a função **FazConexãoWiFi()** que permite que o NodeMCU inicie sua conexão com a internet.

Na função **loop()**, em cada iteração, primeiro verifica-se que o cliente esteja desconectado. Caso contrário, a conexão é concluída. Após essa verificação realiza-se a leitura do sensor de umidade. Finalmente, verifica-se se o intervalo de tempo transcorrido desde a última conexão é superior a 30 segundos. Em caso afirmativo os dados de umidade são enviados ao *ThingSpeak*. É possível observar essas operações no monitor serial da IDE do Arduino.

void **setup**()

{

**Serial**.begin(9600);

lastConnectionTime = 0;

FazConexaoWiFi();

**Serial**.println("Planta IoT com ESP8266 NodeMCU");

}

//loop principal

void **loop**()

{

float UmidadePercentualLida;

int UmidadePercentualTruncada;

char FieldUmidade[11];

//Força desconexão ao ThingSpeak (se ainda estiver conectado)

if (client.connected())

{

client.stop();

**Serial**.println("- Desconectado do ThingSpeak");

**Serial**.println();

}

UmidadePercentualLida = FazLeituraUmidade();

UmidadePercentualTruncada = (int)UmidadePercentualLida; //trunca umidade como número inteiro

//verifica se está conectado no WiFi e se é o momento de enviar dados ao ThingSpeak

if(!client.connected() &&

(millis() - lastConnectionTime > INTERVALO\_ENVIO\_THINGSPEAK))

{

sprintf(FieldUmidade,"field1=%d",UmidadePercentualTruncada);

EnviaInformacoesThingspeak(FieldUmidade);

}

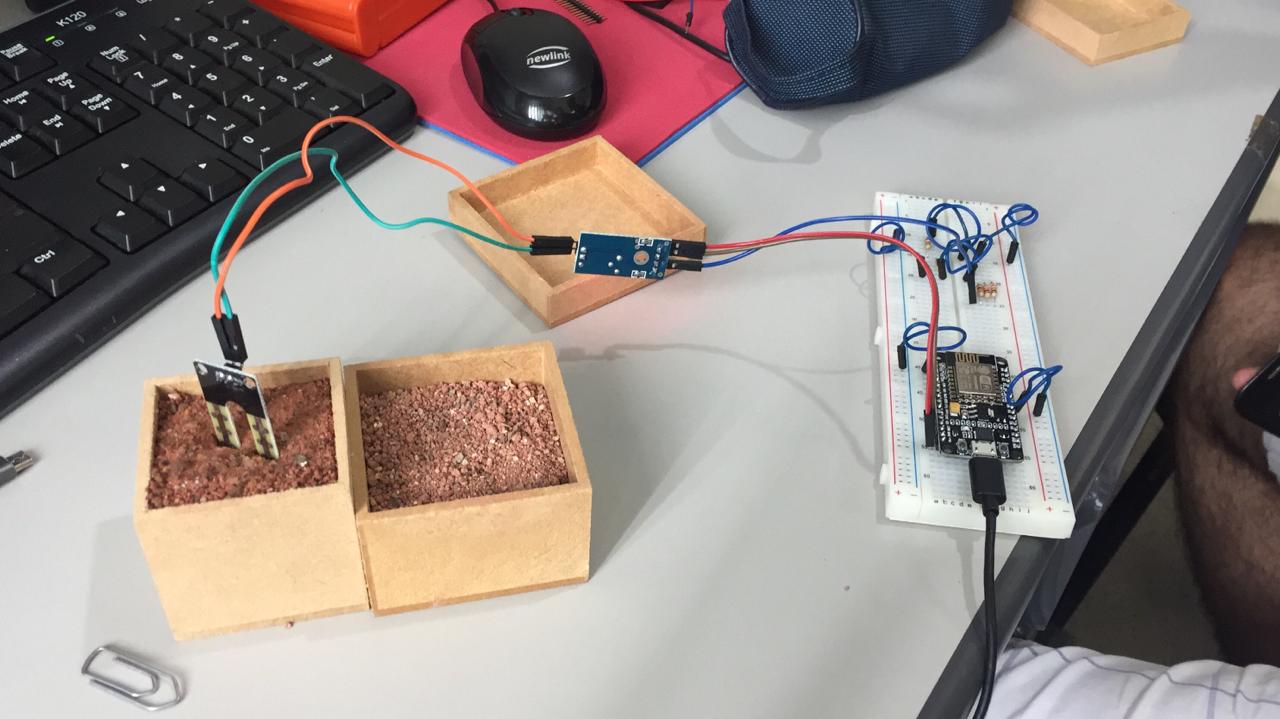
delay(1000);

}

**Quadro 5.-** Código de Controle:Funções de Inicialização e Controle

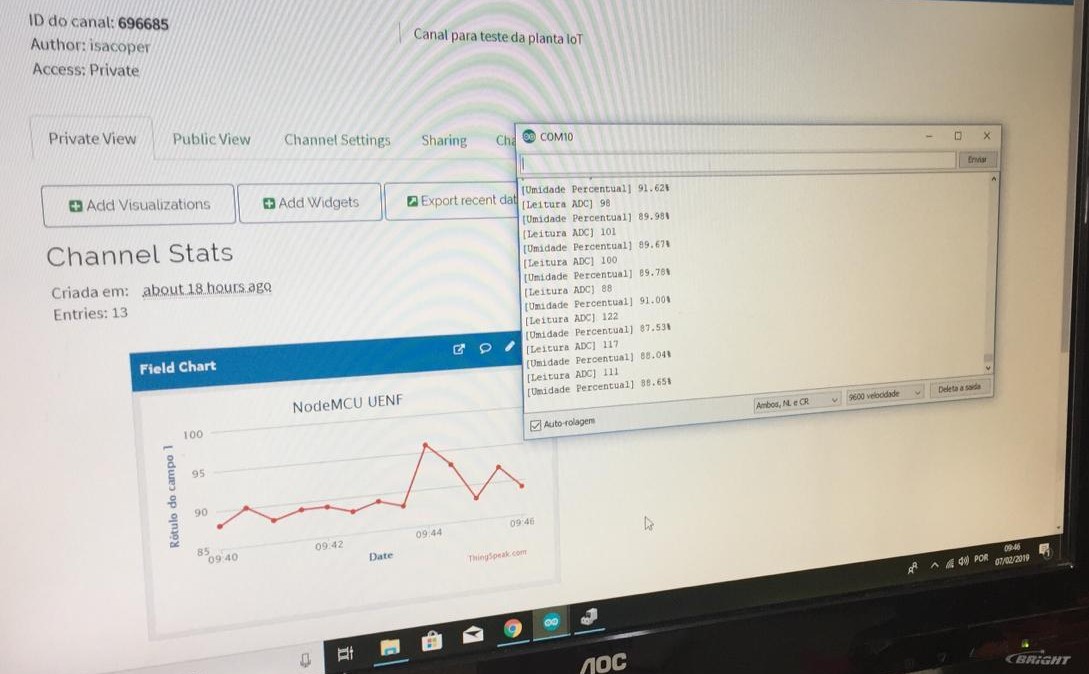
*Protótipo do Dispositivo*

Um primeiro protótipo para o monitoramento da umidade do solo foi desenvolvido e devidamente testado. Para isso foram utilizadas duas pequenas caixas em material mdf, que foram preenchidas com terra coletada na própria instituição. Na primeira caixa a terra foi umedecida enquanto na segunda ela foi mantida seca. A figura 13 ilustra o protótipo de monitoramento de umidade do solo devidamente montado.



**Figura 13 –** Simulação dePlanta IoT:

Protótipo do Dispositivo – Fonte: A autora



**Figura 14 –** Dados na plataforma *ThingSpeak* – Fonte: A autora

Já a Figura 14 mostra os dados de umidade do solo recebidos pela plataforma de IoT *ThingSpeak* e apresentados de forma gráfica ao longo do tempo, em termos percentuais e em intervalos de 30 segundos. Na imagem, observam-se as leituras de umidade na terra molhada, variando de 88,6% até 97%. A imagem mostra também os dados numéricos das leituras no monitor serial do IDE do Arduino.

Finalmente, a Figura 15 mostra o perfil da planta IoT no Twitter, ilustrando como seria a interação da mesma ao informar quando ela deveria ser regada.



**Figura 15 –** Perfil no Twitter da Planta IoT – Fonte: A autora

**5.2.- Protótipo para irrigação autônoma da Planta IoT**

Nesta etapa apresenta-se uma modificação do dispositivo anterior, que é uma tentativa de melhoria que visa garantir a toma de decisão automática por parte do dispositivo. No caso anterior, temos a planta nos informando, através de *tweets* enviados pelo servidor, quando precisa ser regada. Porém, essa situação pode não ser considerada totalmente adequada para o usuário. Uma alternativa possível seria fazer com que a planta pudesse se regar “sozinha” toda vez que o sensor identificasse um nível de umidade considerado baixo. Na verdade, propõe-se desenvolver um dispositivo que tome essa decisão de maneira autônoma. Este tipo de dispositivo permitiria, por exemplo, que uma pessoa que precisa viajar constantemente, pudesse cultivar plantas em sua residência, sem se preocupar nos períodos de ausência.

Assim, em vez do servidor *ThingSpeak* enviar um *tweet* informativo, agora, ao receber uma medida de umidade abaixo de 60%, utilizaria um sistema de irrigação que acionaria uma válvula de vazão solenoide para permitir a passagem de água no sistema, que conectaria uma fonte de agua (ou torneira) a uma mangueira (ou tubulação) até chegar na planta.

A apresentação do desenvolvimento deste protótipo é dividida nas seguintes seções: i) Componentes utilizadas, ii) Diagrama do dispositivo iii) Código de controle e iv) Protótipo do dispositivo.

*Componentes Utilizados*

Para simular um sistema de irrigação, foram usados os componentes descritos abaixo:

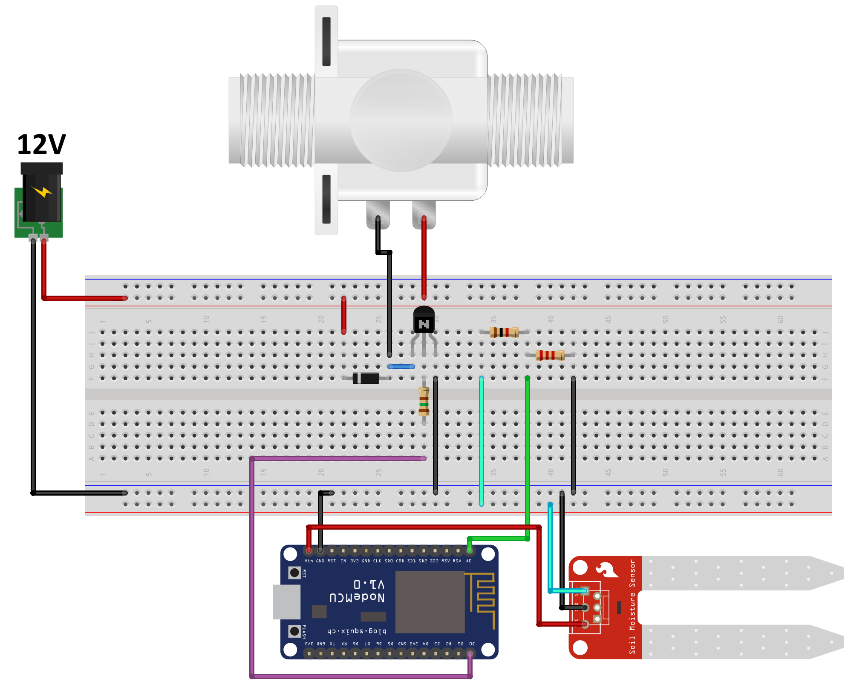
1. NodeMCU;
2. Válvula de Vazão Solenóide 12V;
3. Fonte 12V;
4. Sensor de Umidade do solo Higrômetro;
5. Resistor 1K ohms;
6. Resistor 2K ohms;
7. Protoboard;
8. Transistor BC337;
9. Diodo 1N4007;
10. Jumpers;
11. Mangueira;
12. Galão de água 20L;

*Diagrama de Conexão*

O diagrama de conexão entre as componentes mencionadas, definem o circuito de controle. A Figura 16, ilustra o diagrama para este circuito. A válvula solenóide está conectada a uma fonte de 12V, no entanto, a sua ativação é controlada utilizando-se um transistor BC337, que funciona como um interruptor eletrônico que permite a passagem de corrente sempre que um pequeno sinal é enviado pelo controlador do NodeMCU ao pino central do transistor (linha roxa). Vale observar que o controle da válvula é semelhante ao de um motor DC. De maneira semelhante, quando essa válvula é fechada é necessário um mecanismo de proteção para a corrente inversa. Ou seja, uma vez desativada, devido à inércia, ela vai continuar funcionando por um período de tempo, e nesse curto espaço, ela pode gerar uma corrente elétrica contrária, que pode voltar para a placa NodeMCU e danificá-la. Devido a isso, o circuito utiliza um diodo como proteção.

O transistor tem três pernas: base, coletor e emissor. Juntas elas funcionam como um interruptor, ou seja, aplicando tensão à base, a corrente flui entre o emissor e o coletor, permitindo que uma corrente maior do que a inicialmente permitida, possa circular. Já o diodo é uma válvula de sentido único, que vai impedir que a corrente volte e garantir que o circuito funcione corretamente.

Como a tensão de saída do higrômetro, que registra a umidade do solo (linha azul clara) fornece 5V, mais uma vez, estão sendo usados resistores para regular a tensão, já que o NodeMCU aceita no máximo 3.3V, regulando assim a tensão de entrada na porta analógica (A0) do NodeMCU.(linha verde).



**Figura 16 –** Diagrama de Irrigação autônoma de Planta IoT – Fonte [12]

*Código de Controle*

O código de controle é muito semelhante ao código apresentado na seção anterior, apenas será necessário substituir a geração do *tweet* como resposta ao evento de ter o solo com baixa umidade pela ativação da válvula solenoide. Neste caso, o servidor deverá enviar uma mensagem de ativação ao NodeMCU. Por motivos de espaço essa parte do projeto não será apresentada.

*Protótipo do Dispositivo*

Para poder viabilizar o teste do dispositivo de irrigação, foi montado um sistema para simular uma fonte de água, representado por um galão de 20 litros, conectado a uma mangueira, que na sua outra extremidade foi conectada na válvula solenoide, conforme ilustrado na Figura 17. Para a construção deste protótipo o galão de agua foi perfurado e a mangueira fixada ao galão e à válvula utilizando como material de vedação massa durepoxi, que funcionou perfeitamente com quase nenhum vazamento de agua.



**Figura 17 –** Protótipo do Sistema de Irrigação – Fonte: A autora

Com o galão cheio pela metade, o protótipo mostrou funcionamento adequado. Sempre que energizada, a válvula solenoide liberou a passagem de água, como mostra a imagem da Figura 18. Embora o volume de água liberado fosse adequado para regar a planta, observa-se que outras aplicações podem demandar uma bomba de agua para aumentar o fluxo.



**Figura 18 –** Protótipo do Sistema de Irrigação – Fonte: A autora

**5.3.- Testes de comunicação com o Braço Robótico**

Paralelamente aos trabalhos descritos anteriormente, foram realizados alguns testes de comunicação remota com o dispositivo do braço robótico, desenvolvido no primeiro ano da Iniciação Científica, e composto por quatro servo motores controlados através de módulos *joysticks.* Como proposta de melhoria, pretende-se mudar esse controle para que seja feito via internet, através de algum dispositivo como computador ou celular. Para isso, foram feitos alguns testes a fim de descobrir qual a placa ideal nesse caso.

Inicialmente foi estudada e testada a placa ESP8266 ESP-01 e, logo no início do processo foi observado que para uso dessa placa, é preciso que nela seja gravado um firmware para que seja possível a conexão à rede WiFi e que ela seja utilizada junto ao Arduino.

Nesse momento foi necessário o uso do Adaptador USB para Módulo ESP8266, descrito na sessão 4.2. Após alguns testes, verificou-se que o adaptador, na forma como comercializado, somente pode ser utilizado para conectar a placa ao computador. No entanto, para poder programá-la e necessária a conexão dos pinos GND e GPIO0, através de um fio com uma ponta soldada em cada um deles, conforme a figura 16. Somente com essa alteração no hardware é possível ativar o modo programação da placa ESP-01.



**Figura 19 –** Adaptador USB com modo de

programação ativado – Fonte: A autora

Uma vez ativado o modo programação, basta conectar o adaptador ao computador e fazer a instalação do firmware desejado, de acordo com [13] e [14]. São diversos os comandos AT disponíveis, mas os necessários para esse caso foram os que permitem fazer a configuração WiFi através do monitor serial da IDE do Arduino. São eles: AT+GMR para verificar a versão do firmware instalada, AT+CWMODE? Para verificar em que modo a placa está trabalhando, AT+CWLAP para fazer a conexão à rede, e por último, AT+CIFSR para verificar o número de IP fornecido pela rede.

Essa etapa de configuração foi bem sucedida, mas toda vez que o ESP-01 era desconectada do computador, todas as informações gravadas na placa eram apagadas, impedindo continuar com sua utilização.

Por isso, em consenso com o orientador, foi decido que, apesar da placa ESP-01 ser uma opção mais económica, ela não parece ser uma opção prática e confiável. Além disso, por já ter sido usado no projeto principal da Planta IoT e, por ter funcionado corretamente, o NodeMCU se traduz na melhor opção para controle dos servos, visto que ficaríamos muito tempo presos à uma única placa, sem sucesso.

Na continuação das pesquisas, foi visto que só seria possível controlar um único servo utilizando a placa NodeMCU, uma vez que ele possui um único pino analógico e não é capaz de alimentar os quatro servos dispostos no braço robótico. Por fim, chegamos à conclusão de que embora seja é possível fazer esse controle remotamente, usando a placa NodeMCU, seria necessária uma pesquisa mais aprofundada para saber como fazer a alimentação de energia dos servomotores de forma independente a essa placa, para evitar a sua sobrecarga.

**6.- Conclusões**

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.

Terminada essa parte, foi decidido em acordo entre orientador e orientada que o projeto a ser desenvolvido seria o da Planta IoT, que consiste em um sistema de monitoramento e irrigação autônomo. O desenvolvimento do projeto foi concluído com sucesso, de acordo com as etapas especificadas no plano de trabalho. A orientada conseguiu, através dos conhecimentos adquiridos ao longo da pesquisa, além de montar todo o protótipo e circuito da planta, fazer com que ela se comunique com a internet através de um servidor online, enviando dados para o mesmo. Além disso, foi estudada uma maneira de melhorar o projeto e concluímos que é possível fazer com que a planta consiga ser regada sem interferência humana, através do uso de uma válvula solenoide. A etapas do plano de trabalho referentes à construção do dispositivo, foram em sua grande maioria cumpridas de maneira satisfatória.

Em relação ao braço robótico, tínhamos a expectativa de mudar o controle de seus motores para uma via remota, podendo ser usado um dispositivo móvel, por exemplo. Devido à problemas técnicos de alimentação e falta de recursos no laboratório, foi descoberta a maneira ideal de implementação desse controle via web, porém ainda não foi possível conectar os quatro servos instalados no braço. Ainda assim, o estudo de qual placa seria utilizada e de como fazer essa conexão, também foi concluído.

**7.- Referências**

A quarta revolução industrial livro por klaus martin schwab

[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/>

[4] <http://blog.eletrogate.com/nodemcu-esp12-introducao-1/>

[5] http://blogmasterwalkershop.com.br/embarcados/wemos/conhecendo-a-wemos-d1/

[6] <https://www.vidadesilicio.com.br/sensor-umidade-solo-higrometro>

[7] https://www.filipeflop.com/produto/valvula-de-vazao-solenoide-agua-12vdc/

[8] <https://www.filipeflop.com/produto/protoboard-400-pontos/>

[9] <https://www.creatroninc.com/product/8-pin-arduino-jumper-wire/>

[10] <https://www.robotics.org.za/RES-330E-50>

[11] <https://www.filipeflop.com/produto/adaptador-usb-para-modulo-wifi-esp8266-esp-01/>

[12] https://www.filipeflop.com/blog/planta-iot-com-esp8266-nodemcu-parte-4/

[13] <http://binarycodec.blogspot.com/2016/03/atualizar-o-firmware-do-esp8266.html>

[14] <https://www.youtube.com/watch?v=Na7CSbq467M>

<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

**8. - Desdobramento do trabalho**

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.

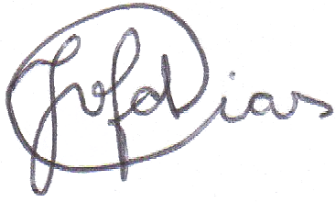
Havendo sucesso na implementação do controle do braço robótico via internet, propõe-se também o desenvolvimento de algum dispositivo IoT que possa ter alguma funcionalidade prática no contexto da UENF. Não havendo nenhuma ideia fixa, deixando a decisão do tipo do dispositivo para o bolsista e seu orientador.

[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]

**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, 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?]



**10. - Data e assinatura do bolsista**

XX de março de 2020.

**11. - Data e assinatura do orientador**

****

XX de março de 2020.