**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**

**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**

Nome do Bolsista: João Vítor Fernandes Dias

Curso e N° Matrícula: Ciência da Computação / 00119110377

Orientador: Fermín Alfredo Tang Montané

Fonte financiadora da Bolsa: PIBIC / CNPq

**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)

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.

O bolsista substituto optou 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, 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, ArduinoUNO 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?]

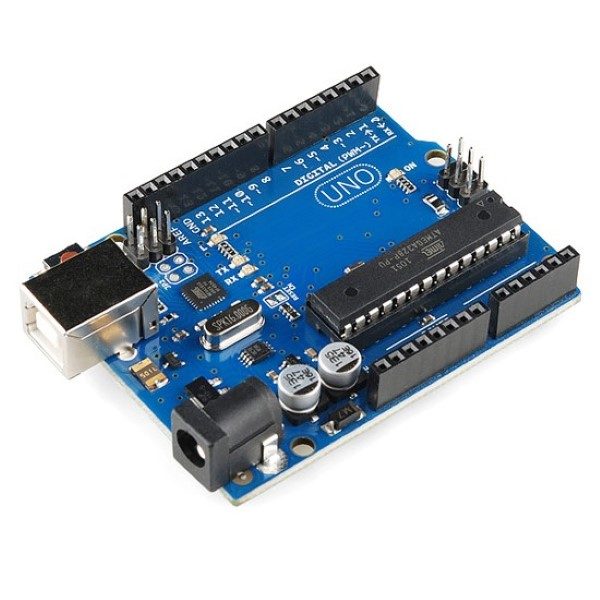
**4. - Metodologia**

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.

Abaixo estão listados os componentes, utilizados nos experimentos, explicados de forma básica (Imagens fora de escala e meramente ilustrativas):

1. Arduino uno



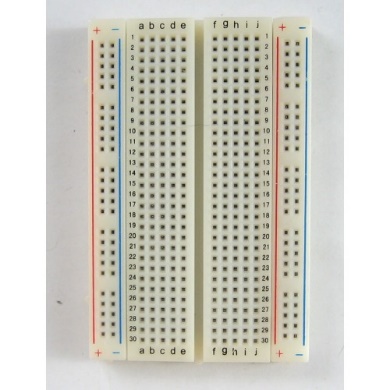
Minicomputador (microcontrolador) que permite programar em uma linguagem baseada em C os valores de energia recebidos e enviados pelas portas de entrada e saída presentes na placa.

1. Cabo usb



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)



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



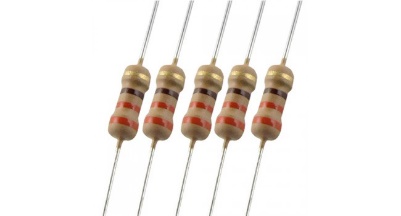
Cabos condutores maleáveis e com ponta sólida, próprios para o uso nas protoboards e nos componentes Arduino. Servem para fazer as conexões elétricas no circuito.

1. Botão (Switch)

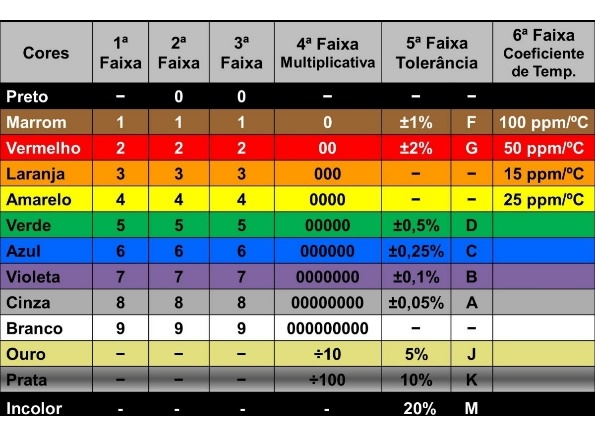


Componente que mantém aberto (sem passagem de eletricidade) o circuito em que está conectado, fechando-o quando pressionado.

1. Resistores de 330 Ω



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.



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



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

1. Potenciômetros



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

1. Joystick



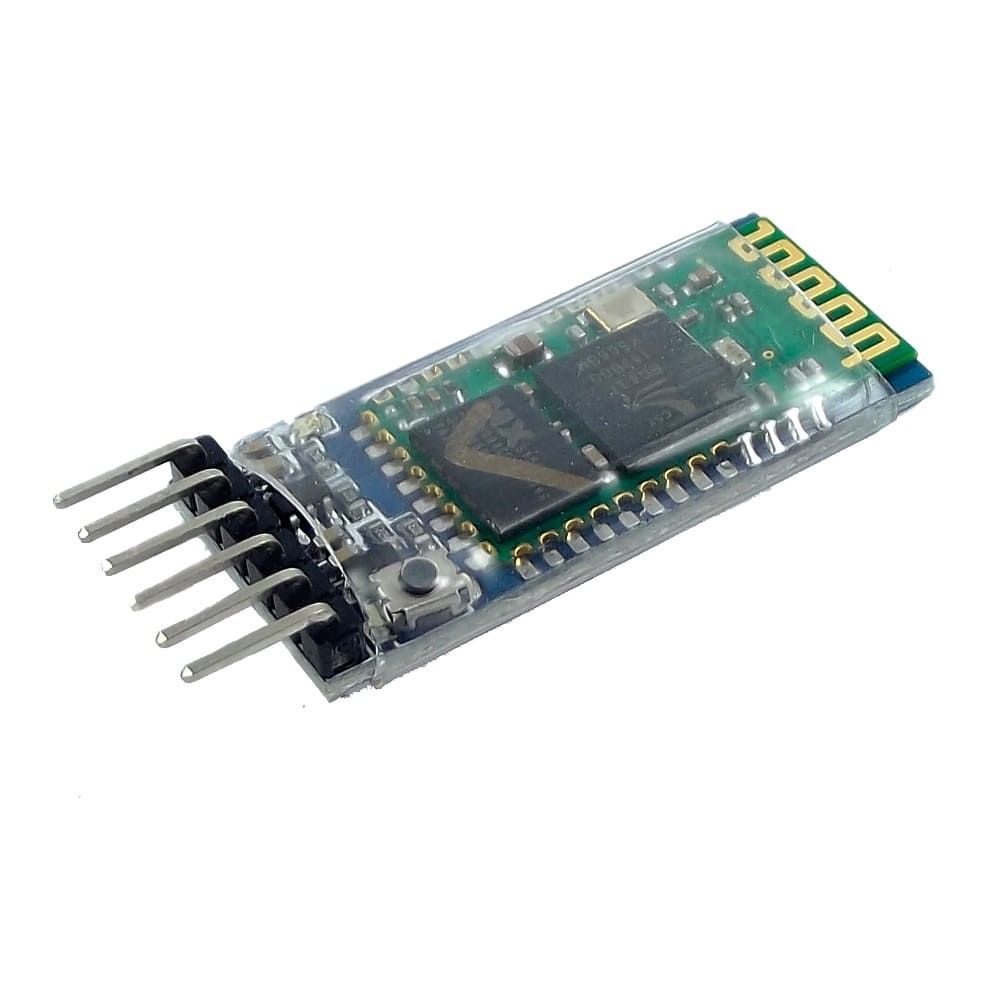
São dois potenciômetros acoplados a uma peça plástica de tal forma que representam as coordenadas X e Y em que essa peça está localizada.

1. Micro Servomotor 9g



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



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. Celular com conexão Bluetooth.

Qualquer aparelho de telefonia móvel capaz de efetuar conexão Bluetooth e possa instalar aplicativos externos, preferencialmente via Código QR.

1. Aplicativo de controle Bluetooth

Aplicativo desenvolvido através da plataforma MIT App Inventor 2, com o objetivo de parear os dispositivos Bluetooth e controlar os componentes conectados.

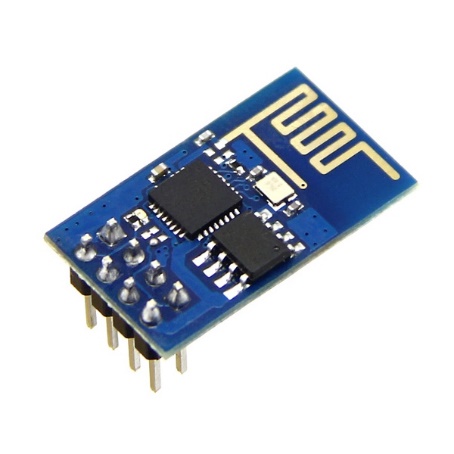
Inicialmente, como forma de comunicação remota com o Arduino, foi utilizado um aplicativo genérico encontrado na Google Play que enviava informações via Bluetooth, porém, ele não apresentava as funcionalidades necessárias visadas, que são a de conseguir controlar simultaneamente 4 diferentes motores 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 “frontend”), quanto para a lógica de programação que acontece “por detrás das cortinas” (o que em computação é chamado de “backend”).

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.

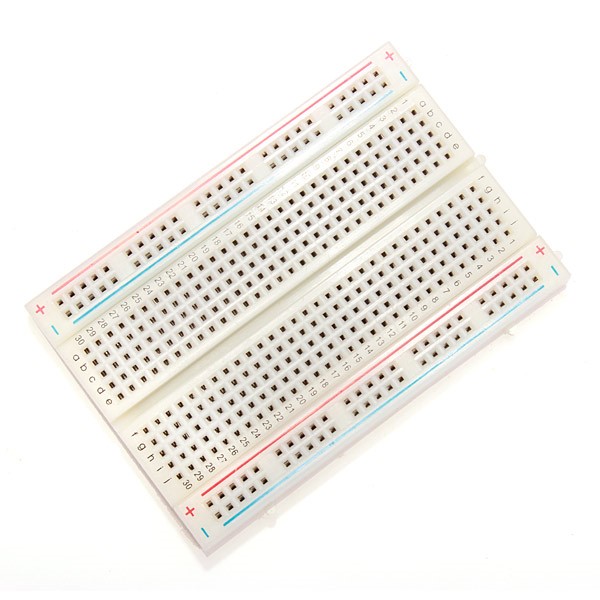
1. Microprojetos
   1. A
   2. A
   3. A
   4. A
   5. A
   6. A
2. MIT App Inventor 2
3. Braço Robótico

**4.1.1-ESP8266-01**



**Figura 1** – ESP8266-01 – Fonte [3]

*Protoboard*



**Figura 6 –** Protoboard – Fonte [8]

*Jumper*



**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.

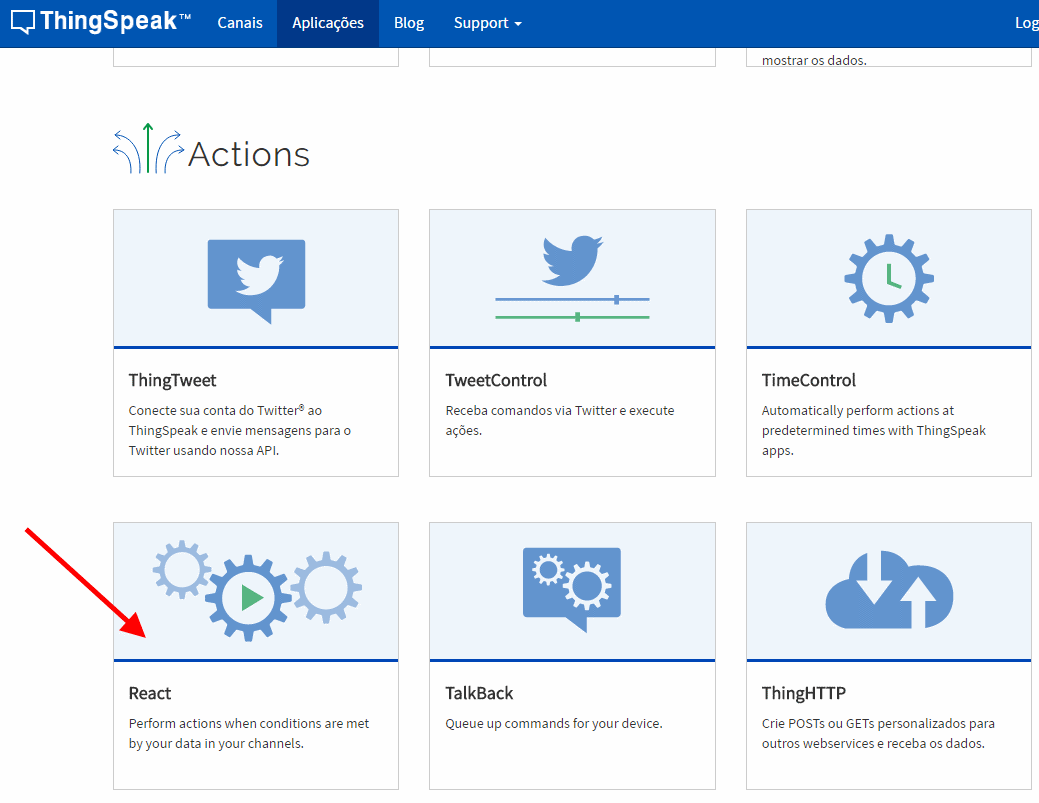
Para enviar dados ao *ThingSpeak* é necessária a criação de uma chave de escrita que será associada ao canal de comunicação utilizado na plataforma. O envio de dados ao servidor é realizado na forma de uma requisição HTTP. Após enviar os dados para a plataforma, o dispositivo avalia o grau de umidade 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.

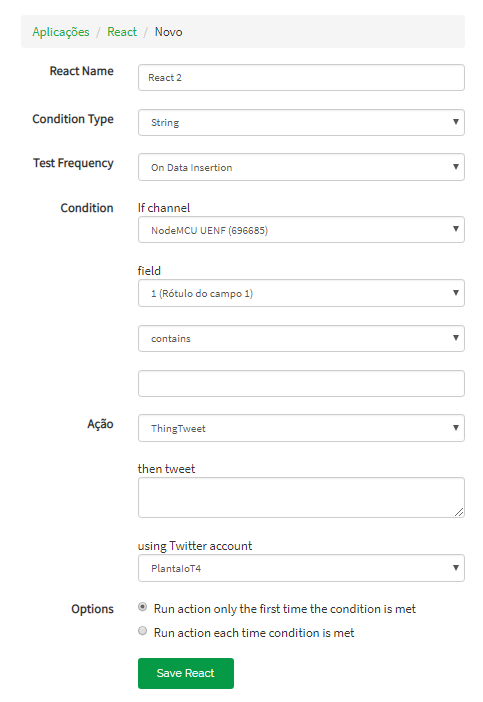
Após essa etapa, foi criada uma conta na plataforma *ThingSpeak*, para que uma chave de escrita pudesse ser gerada e um canal de comunicação definido. Assim, a chave criada serve para direcionar para qual canal os dados devem ser enviados. Por último, foi necessário criar uma conta no *Twitter* para 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;



**Figura 10 –** Configurar tweets no ThingSpeak – Fonte: A autora

1. Clicar em “New React” e preencher os dados de acordo com a Figura 11;
2. 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*

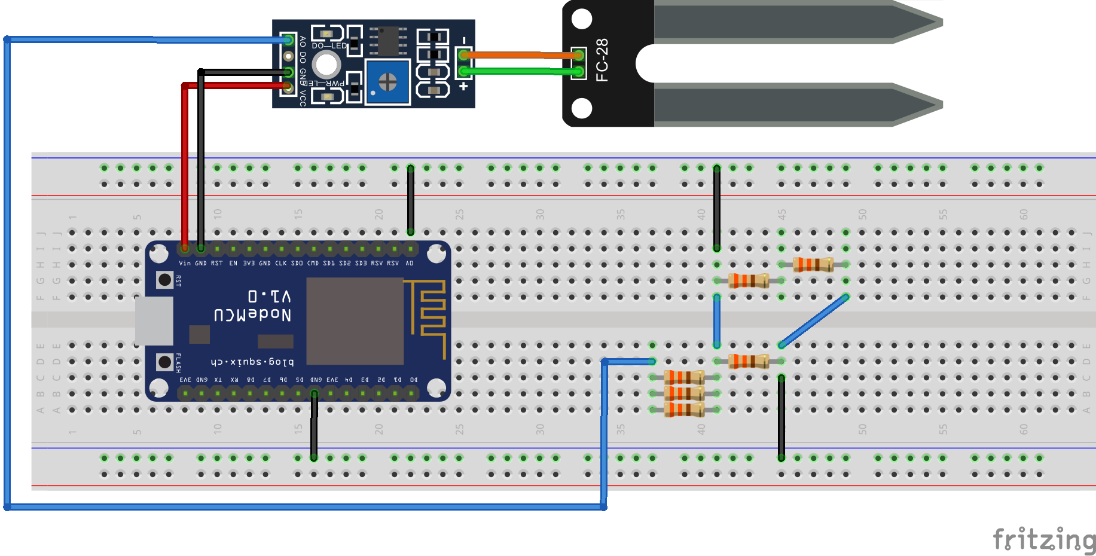
Foram utilizados os seguintes componentes, descritos na sessão anterior:

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);

}

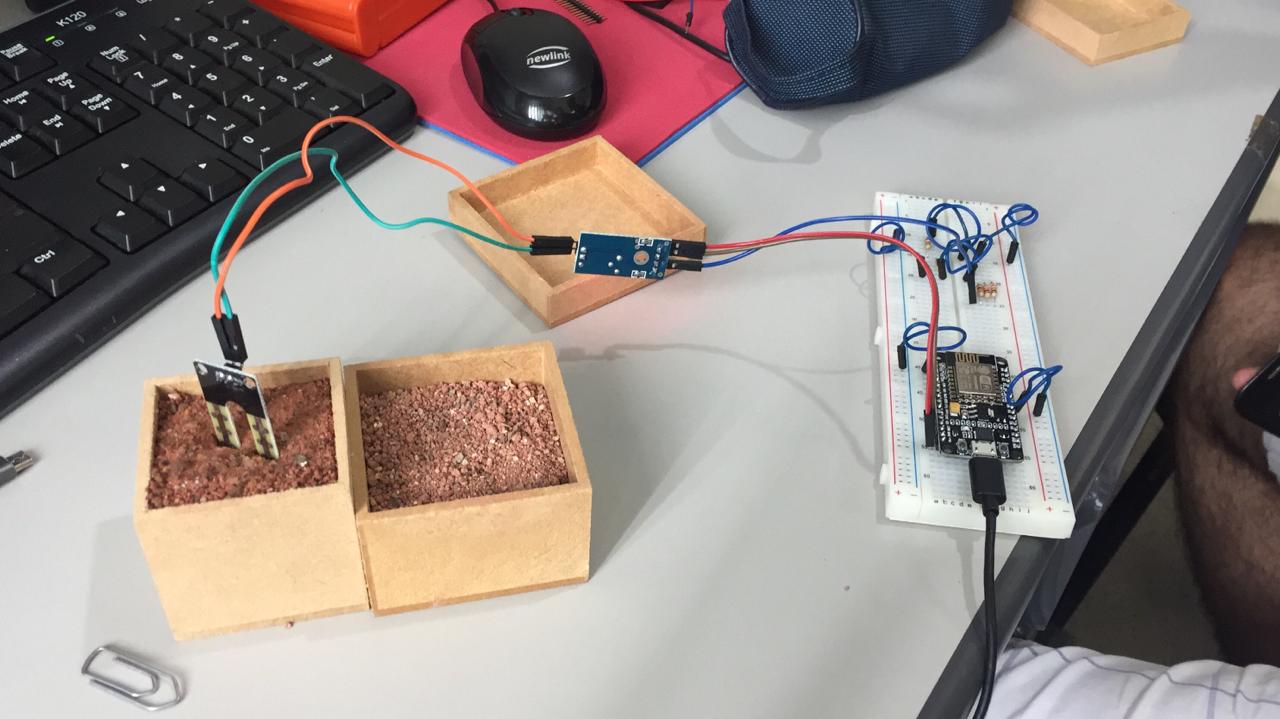
delay(1000);

}

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

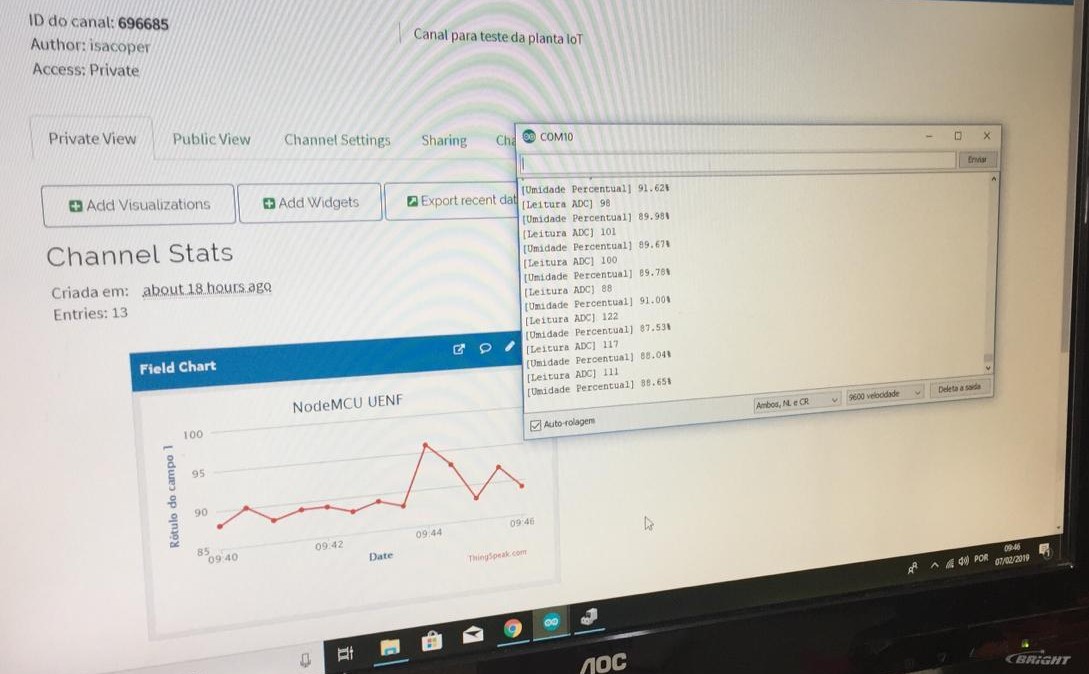
*Protótipo do Dispositivo*

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:

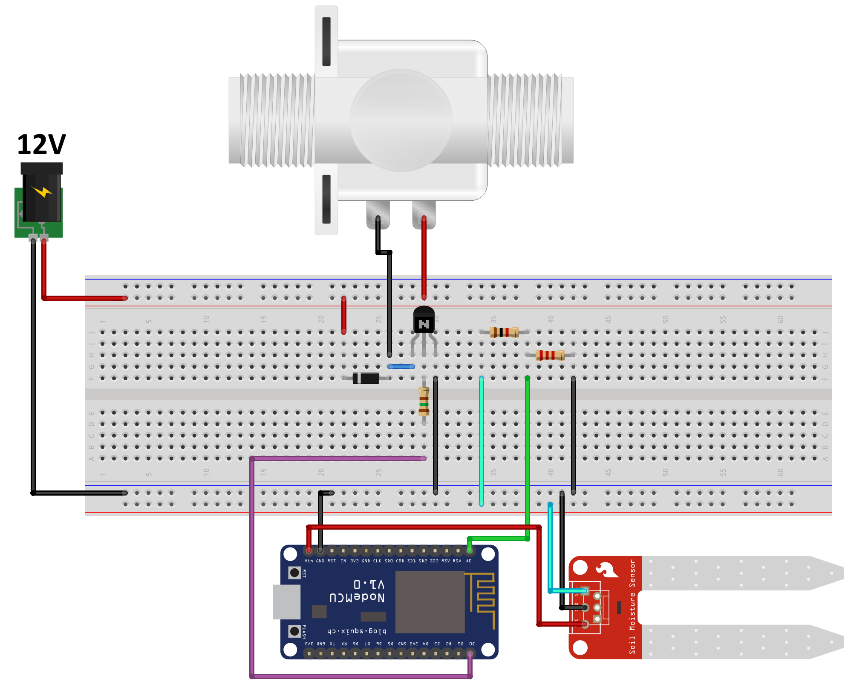
1. NodeMCU;
2. Válvula de Vazão Solenóide 12V;
3. Fonte 12V;
4. Sensor de Umidade do solo Higrômetro;
5. Resistor 1K ohms;
6. Resistor 2K ohms;
7. Protoboard;
8. Transistor BC337;
9. Diodo 1N4007;
10. Jumpers;
11. Mangueira;
12. Galão de água 20L;

*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**

Ao começar a pesquisa, o orientador disponibilizou livros sobre Internet das Coisas, além de compartilhar com a orientada artigos e sites sobre esses temas. A primeira etapa visou focar no estudo das placas utilizadas nessa pesquisa, além das linguagens de programação necessárias para desenvolvimento web. Também houve uma familiarização com a área de redes, incluindo o estudo e conhecimento dos protocolos exigidos para estabelecer uma comunicação entre um dispositivo e a internet.

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>

**8. - Desdobramento do trabalho**

Na continuação da pesquisa, conforme descrito no plano de trabalho proposto na solicitação de renovação, propõe-se o aprimoramento de dois dispositivos, desenvolvidos em anos anteriores: Braço Robótico e Sistema de monitoramento da Planta IoT. Deverão ser estudadas possíveis melhorias na interface de controle dos dispositivos, na comunicação remota, no sistema de alimentação de energia, assim como no controle otimizado dos motores e válvulas. Com esta finalidade, propõe-se introduzir o uso de uma placa Raspberry Pi3 B+, recentemente adquirida, que conta com maior poder de processamento e propõe-se estudar a integração entre as plataformas disponíveis: Arduino, NodeMCU e Rasberry Pi, de modo a tirar melhor proveito das capacidades de cada uma delas.

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

Como parte da disciplina de inteligência artificial ministrada no semestre 2017-2, a aluna bolsista fez modificações em um veículo robótico incorporando um sensor de ultrassom de maneira a detectar a presença de obstáculos e corrigir a sua direção mudando de trajetória, sendo assim capaz de introduzir inteligência ao dispositivo.

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



XX de Março de 2020.

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

****

XX de Março de 2020.