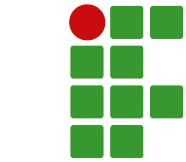

Instituto Federal de Goiás
Campus Formosa
Análise e Desenvolvimento de Sistemas
http://www.ifg.edu.br/formosa

SISTEMA DE CONTROLE DE ILUMINAÇÃO E IRRIGAÇÃO PARA ESTÁDIOS DE FUTEBOL UTILIZANDO O MICROCONTROLADOR ESP8266 (NODEMCU)

LUCAS ALVES DA COSTA

Trabalho de Conclusão de Curso

FORMOSA
2019

Instituto Federal de Goiás
Campus Formosa
Análise e Desenvolvimento de Sistemas
http://www.ifg.edu.br/formosa

SISTEMA DE CONTROLE DE ILUMINAÇÃO E IRRIGAÇÃO PARA ESTÁDIOS DE FUTEBOL UTILIZANDO O MICROCONTROLADOR ESP8266 (NODEMCU)

Lucas Alves da Costa

Trabalho de Conclusão de Curso apresentado ao Departamento de Áreas Acadêmicas da Instituto Federal de Goiás campus Formosa, como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Mario Teixeira Lemes

FORMOSA
2019

Lucas Alves da Costa
Sistema de Controle de Iluminação e Irrigação para Estádios de Futebol Utilizando o Microcontrolador ESP8266 (NodeMCU)/ Lucas Alves da Costa. – FORMOSA, 2019-
95 p.; 30 cm.

Orientador Prof. Me. Mario Teixeira Lemes

Trabalho de Conclusão de Curso – Instituto Federal de Goiás, 2019.

1. Internet das Coisas 2. Sistema de Iluminação e Irrigação 3. Dispositivos Embarcados I. Orientador: Prof. Me. Mario Teixeira Lemes. II. Instituto Federal de Goiás. IV. Título: Sistema de Controle de Iluminação e Irrigação para Estádios de Futebol Utilizando o Microcontrolador ESP8266 (NodeMCU)

CDU 02:141:005.7

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO	
Tecnologia em Análise e Desenvolvimento de Sistemas	
1 - Identificação do Aluno	
Lucas Alves Da Costa	Matrícula: 20151070130183
2 – Título do Trabalho de Conclusão de Curso	
Sistema de Controle de Iluminação e Irrigação para Estádios de Futebol Utilizando o Microcontrolador ESP8266 (NodeMCU)	
3 – Avaliação da Banca Examinadora (Notas)	
Mario Teixeira Lemes (Orientador)	
Uyara Ferreira Silva (Avaliador)	
Sirlon Thiago Diniz Lacerda (Avaliador)	
Média Final	
4 - Resultado:	
A Banca Examinadora, em 05 de Dezembro de 2019, após a *Defesa do Trabalho de Conclusão de Curso* e arguição, decidiu:	
☐ Pela aprovação do TCC *(correções)*.	☐ Pela reprovação do TCC
Preenchido pelo Orientador após a entrega da versão final do TCC:	
☐ Correções efetuadas conforme requerido pela Banca Examinadora e o TCC foi **aprovado**.	
☐ Correções não efetuadas conforme requerido pela Banca Examinadora e o TCC foi **reprovado**.	

_____
Prof. Me. Mario Teixeira Lemes - Orientador (a)

_____
Prof. Me. Sirlon Thiago Diniz Lacerda

_____
Profª. Ma. Uyara Ferreira Silva

Autenticação pelo Coordenador de Área:	Homologação pelo Professor da Disciplina de TCC:
Data: ___ / ___ / _____	Data: ___ / ___ / _____
_____ Assinatura / Carimbo	_____ Assinatura / Carimbo

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

[ ] Tese	[ ] Artigo Científico
[ ] Dissertação	[ ] Capítulo de Livro
[ ] Monografia – Especialização	[ ] Livro
[ ] TCC - Graduação	[ ] Trabalho Apresentado em Evento
[ ] Produto Técnico e Educacional - Tipo: _____	

Nome Completo do Autor:
Matrícula:
Título do Trabalho:

Restrições de Acesso ao Documento

Documento confidencial: [ ] Não [ ] Sim, justifique: _____

_____

Informe a data que poderá ser disponibilizado no ReDi/IFG:__/__/__
O documento está sujeito a registro de patente? [ ] Sim [ ] Não
O documento pode vir a ser publicado como livro? [ ] Sim [ ] Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

_____, ____/____/____.
Local Data

_____
Assinatura do Autor e/ou Detentor dos Direitos Autorais

Eu dedico este trabalho à todos que estiveram presentes em minha jornada acadêmica colaborando de alguma forma com meu crescimento profissional.

Agradeço primeiramente a Deus por abrir diversas portas em meu caminho me dando força, sabedoria e capacidade para alcançar meus objetivos.

A minha família por sempre me apoiar nesta jornada em busca do conhecimento.

Ao meu orientador professor Me. Mario Teixeira Lemes, pelo conhecimento, orientação e compreensão.

Aos amigos e amigas pelo apoio, incentivo e torcida pelo sucesso. A minha namorada pelo apoio, compreensão e conselhos durante esta etapa.

Ao professor Daniel Saad Nogueira Nunes por me incentivar a correr atrás dos meus objetivos. Obrigado por mostrar que sou capaz.

Aos professores de graduação que contribuirão com minha formação, repassando seus conhecimentos e me aconselhando em momentos de indecisão.

Aos colegas de graduação que compartilharam momentos incríveis, desde apresentações de trabalhos bem sucedidas até as mais simples conversas. Vocês tornaram esta etapa um momento prazeroso em minha vida.

A Coordenação de Assistência Estudantil (CAE) do Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG) Campus Formosa-GO por colaborar com o auxilio financeiro durante todos estes longos cinco anos que estive presente nesta Instituição. Este nobre ato garantiu minha frequência e o aprendizado nas disciplinas cursadas.

A todos que me ajudaram direta ou indiretamente nessa jornada; desde uma simples carona até o ponto de ônibus, conselhos ou ensinamentos sobre algo que não conseguiria resolver sozinho.

A vida é um processo fluente e em alguns lugares do caminho coisas desagradáveis ocorrerão. Podem deixar cicatrizes, mas a vida continua a fluir. É como a água fluente, que ao estagnar-se, torna-se podre; não pare! Continue bravamente... porque cada experiência nos ensina uma lição.

—BRUCE LEE

Resumo

Internet das Coisas pode ser considerada uma evolução da Internet, no qual permite que objetos do cotidiano se conectem a ela por meio de sensores e outros dispositivos inteligentes. Através deste paradigma é possível controlar, trocar informações e acessar serviços fornecidos pelos objetos inteligentes. O objetivo deste trabalho de conclusão de curso é propor o desenvolvimento de um sistema responsável por controlar a iluminação e a irrigação do gramado presente em estádios de futebol, visando a automação destes processos e o fornecimento de condições ideais para o desenvolvimento da grama, utilizando dispositivos típicos do contexto de IoT, tais como o microcontrolador ESP8266 (NodeMCU) e o protocolo de comunicação Message Queue Telemetry Transport. Uma interface externa será possível através do desenvolvimento de um aplicativo para smartphone para se obter acesso aos valores obtidos pelos sensores. O desenvolvimento desse projeto permitirá condições ideais para a desenvolvimento da grama em estádios de futebol, bem como facilitará a administração dos processos de iluminação e irrigação nestes ambientes.

Palavras-chave: Internet das Coisas, Sistema de Controle de Iluminação e Irrigação, Dispositivos Embarcados.

Abstract

Internet of Things can be considered an evolution of the Internet, it does not allow everyday objects to be connected to it through sensors and other smart devices. Through this paradigm, it is possible to control, exchange information and access services used by smart objects. The purpouse of this work is proportional to the development of a system responsible for controlling the lighting and irrigation of the lawn present in football stadiums, monitoring the automation in these processes and providing the ideal conditions requirements for the development of the grass, using typical features of the IoT context, such as the ESP8266 (NodeMCU) microcontroller and the Message Queue Telemetry Transport protocol. An external interface may enable the development of a smartphone application to gain access to the values used by the sensors. The development of this project allows ideal conditions for football grass development, as well as facilitating the administration of lighting and irrigation processes.

Keywords: Internet of Things, Lighting and Irrigation Control System, Embedded Devices.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

Sumário

1

Introdução

De acordo com CUNHA (2007), a tendência para o futuro é embarcar inteligência em equipamentos. Um sistema embarcado é aquele que possui componentes com capacidade computacional, interligados entre si, cada qual com uma finalidade. Sistemas embarcados são construídos para executar determinadas tarefas pré-definidas não possuindo, em diversos momentos, flexibilidade de *software* que os permitem executar tarefas diferentes das que foram projetadas/definidas.

(BARROS; CAVALCANTE, 2010) introduz que um dos principais motivos para a grande taxa de crescimento da industria eletrônica é a incorporação de sistemas eletrônicos (sistemas embarcados) em uma grande variedade de produtos. Anualmente milhões de equipamentos eletrônicos são fabricados para diversas finalidades, porém diferentes sistemas são produzidos para diversas propostas, sendo que os mesmos se encontram embutidos em equipamentos eletrônicos maiores e executam funções específicas, de forma transparente para o usuário do sistema.

De acordo com (SANTOS et al., 2016), houve um grande crescimento na produção de equipamentos com capacidade de processamento e comunicação. Este fato se dá ao conceito de Internet das Coisas que conecta estes objetos a rede promovendo a comunicação entre dispositivos e usuários. A Internet das Coisas é uma extensão da Internet atual, tornando possível que objetos com capacidade computacional (sistemas embarcados) e de comunicação se conectem a rede e forneçam serviços úteis aos usuários.

Os autores (FERRASA; BIAGGIONI; DIAS, 2010) da Faculdade de Ciências Agronômicas (FCA) - Botucatu, propõem, em uma produção cientifica publicada no periódico Repositório Institucional UNESP, um sistema (protótipo) de baixo custo para monitoramento, via radiofrequência, dos dados de temperatura e umidade de grãos armazenados em silos metálicos voltado aos pequenos produtores de grãos. Esta proposta constitui-se basicamente em um sistema para o monitoramento de temperatura e umidade de grãos armazenados em silos. Os autores apontam a economia financeira e a de instalação o ganho provido pela conclusão do trabalho.

FERNANDES (2010), em sua dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciências e Engenharia de Petróleo da Universidade Federal do Rio Grande do Norte, propõe o desenvolvimento de um sistema para controle da elevação artificial de petróleo e

gás natural. O sistema é constituído por uma rede de sensores sem fio e consiste basicamente na utilização de atuadores e sensores instalados nos poços para controlar unidades de elevação do petróleo.

No ano seguinte, AMORIM (2011) propõe a criação um protótipo para substituir os relés fotoelétricos utilizados pelo sistema de iluminação pública, com novas funcionalidades controladas através da comunicação sem fio. O objetivo é a economia de energia e a facilidade na administração, controle e monitoramento do sistema de iluminação publica.

PIGATTO (2012), em dissertação de mestrado apresentada ao Instituto de Ciências Matemáticas e de Computação (ICMC-USP), aborda o uso de algoritmos criptográficos em um protótipo para prover a comunicação segura entre sistemas embarcados críticos. O objetivo foi propor uma forma eficaz de comunicação entre componentes distribuídos de forma segura. Após analisar e comparar o desempenho de diversos algoritmos criptográficos obteve como resposta conclusiva qual algoritmo é de fato o mais eficiente.

Em produção cientifica apresentada ao curso de Engenharia de Computação, LAMPERT (2012) propõe o desenvolvimento de uma ferramenta digital para o auxílio aos projetistas de *software* no complexo processo de certificação, tendo como base a norma *IEC61508* em parte referente ao desenvolvimento de *software* seguro. Seu trabalho tem como objetivo o conhecimento da norma supracitada, além de oferecer uma ferramenta de auxílio para equipes de desenvolvimento e gerenciamento de projetos que buscam uma certificação internacional de segurança.

FARIAS FILHO (2013) em sua dissertação de mestrado apresentada ao Instituto de Ciências Matemáticas e de Computação (ICMC-USP), propôs desenvolver uma ferramenta conversora de sistemas embarcados, independentemente de plataforma, em modelos de plataforma específica. Este gerador consiste basicamente em uma infraestrutura no qual várias soluções são geradas automaticamente, sendo possível que o usuário determine suas preferências.

Para (SILVA et al., 2013), da Universidade Federal de Minas Gerais (UFMG), a segurança de *software* tornou-se um tema central em sistemas computacionais, porém componentes embarcados se diferenciam de sistemas computacionais convencionais pois dedicam-se a executar bem poucas tarefas. A maioria das propostas de segurança de *software* são concebidas para sistemas convencionais. Portanto, componentes embarcados por não terem suas peculiaridades levadas em conta necessitam de implementações de segurança. Neste trabalho, os presentes autores abordaram tipos de falhas que implicam na segurança de um sistema embarcado e posteriormente apresentam soluções para tais problemas de segurança.

Em dissertação apresentada a Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Computação, (LORENZON, 2014) mostra a diferença comportamental entre diferentes interfaces em termos de desempenho e consumo energético. Segundo o autor esta diferença varia de acordo com as características de cada aplicação e o processador utilizado. Um de seus objetivos está em mostrar que a eficiência de aplicações paralelas muda de acordo com a interface de programação paralela utilizada. Cada interface de programação paralela possui um

comportamento diferente em relação a quantidade de instruções realizadas e acesso a memória de dados, causando impacto no consumo energético e no desempenho do sistema. Ao adotar uma interface de programação paralela correta é possível ter ganhos tanto em consumo de energia quanto em desempenho.

Em produção cientifica disponível no periódico *Open Journal Systems*, (ROCHA et al., 2014) propõem o desenvolvimento de um mecanismo automatizado capaz de controlar com eficiência e simplicidade o processo de irrigação em lavouras. Seguindo o princípio de mobilidade, este sistema consiste em realizar o controle e monitoramento da irrigação através da Internet. O sistema proposto consiste em utilizar novas tecnologias que agreguem a utilização mais eficiente dos recursos naturais, além de um custo acessível para pequenos e médios agricultores. O sistema foi dividido em duas fases: o *hardware*, onde estão conectados os sensores/atuadores, e o *software* que recebe as informações coletadas pelos dispositivos de *hardware*. A implementação deste sistema resume-se basicamente em uma placa de arduíno conectada através de módulos de conexão em rede a um *software* desenvolvido utilizando a linguagem de programação Java.

(SOEIRO et al., 2014), do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), propõem em trabalho científico submetido ao XLI Congresso Brasileiro de Educação em Engenharia, um *software* para controle de horários e pontos através de sensores biométricos. O objetivo é gerenciar a presença dos professores em salas de aula através de um leitor biométrico, para então garantir a presença dos professores em sala de aula no horário correto. A visão geral sobre este sistema consiste em um *software* responsável pelo cadastro e gerenciamento de professores, além do sistema de biometria que identifica o professor cadastrado, tornando possível controlar a frequência e os horários de cada professor.

(MEDEIROS et al., 2015), da Universidade Federal do Rio Grande do Norte - UFRN, abordam em trabalho disponível no periódico *Blucher Mathematical Proceedings* o uso de sistemas embarcados para controlar a posição de painéis solares de acordo com a iluminação incidida sobre os mesmos. Este sistema consiste no uso de componentes embarcados, responsáveis por detectar a posição que apresente maior emissão de luminosidade e assim possibilitar a movimentação do painel solar para a posição mais adequada. A visão geral deste sistema baseia-se em um microcontrolador que possui sensores de luminosidade e servomotores. Através do uso dos sensores, o controlador pode acionar os servomotores para movimentar o painel solar.

Os autores (WEISS; BERNARDES; CONSONI, 2015) abordam o conceito de cidades inteligentes aplicada a cidade de Porto Alegre. O intuito é a colaboração para futuras pesquisas sobre espaços urbanos que funcionam sobre o conceito de cidades inteligentes como forma de viabilizar seu desenvolvimento sustentável. Estes pesquisadores analisaram e apresentaram a iniciativa de implementação de uma cidade inteligente em uma das principais metrópoles brasileiras.

(MATSURA et al., 2015) apontam um projeto que visa controlar níveis de energia e fluxos de água através de um *software* embarcado aplicado a um sistema hidropônico. Tendo em vista um sistema embarcado que permite a comunicação em tempo real, o projeto desenvolvido

busca resolver os problemas identificados no sistema de produção hidropônica como falta de energia e água que resultam na perca da produção. Com todos os requisitos estabelecidos foi possível a construção de uma aplicação embarcada que tem por objetivo melhorar o controle e a produtividade no cultivo hidropônico.

(CUNHA; ROCHA, 2016), da Faculdade de Tecnologia do Estado de São Paulo, buscam oferecer alternativas para gerenciar o processo de irrigação em propriedades agrícolas, de tal forma que seja acessível e de baixo custo para o produtor. Os autores utilizaram-se de um arduíno ligado a outros componentes responsáveis por detectar e monitorar as umidades do ar e solo, sendo também capaz de acionar o sistema de irrigação de forma eficiente.

SAMPAIO (2017), da Universidade Federal da Bahia (UFBA), aborda em sua dissertação de mestrado apresentada ao Programa de Pós-Graduação em Mecatrônica, o desenvolvimento de um sistema de baixo custo cujo objetivo é controlar uma aeronave não tripulada. O autor propôs o desenvolvimento de um algoritmo capaz de controlar atitudes de voo de uma aeronave não tripulada. Este sistema é formado por dois subsistemas que trocam informações através de um *link* de rádio.

(JÚNIOR; PINTO; BRAZ, 2018) abordam uma forma de resolver o problema na integração de alunos com deficiência visual na educação. Com o intuito de explorar o uso de todos os sentidos do aluno com deficiência visual foi projetado um produto que tem como objetivo unir o tato e a audição através de um modelo computacional embarcado num mapa tático. Baseando-se no conceito de Internet das Coisas, este produto possui como infraestrutura de *hardware* uma plataforma arduíno embarcada em um mapa tátil, podendo desta forma prover maior qualidade no processo de ensino/aprendizagem e autonomia das pessoas com deficiência visual.

(GONCALVES et al., 2018) apresentam um sistema para gerenciamento de irrigação automatizada dividido basicamente em três partes essenciais, a citar: um módulo NodeMCU, sensores de umidade conectados a este componente e uma pagina *web* responsável por possibilitar os ajustes e configurações destes componentes, além de permitir o monitoramento do sistema de irrigação em tempo real. Os critérios de funcionamento do microcontrolador NodeMCU foram definidos através da linguagem de programação *C* e os dados coletados pelo mesmo foram armazenados em um banco de dados MySQL e apresentados de forma conveniente por meio da página *web* deste sistema.

(OLIVEIRA; ASSIS; NOLLI, 2019) proporam o desenvolvimento de um sistema responsável por monitorar o consumo energia através de uma plataforma *web*. Para construção de tal sistema foram utilizados sensores, medidores e um componente de comunicação em rede (*Ethernet Shield*) conectados em um Arduíno. O modelo proposto consiste basicamente em um sensor de corrente ligado ao quadro de energia. Constatou-se através medições de energia elétrica a validade do protótipo proposto pois as mesmas se aproximavam com medições feitas por medidores convencionais.

(CAETANO; SOUZA GONÇALVES, 2019) abordam o desenvolvimento de um me-

canismo capaz de interligar pontos de iluminação pública através de uma rede *wi-fi* utilizando componentes embarcados, o que possibilita o controle e monitoramento do sistema de ilumina-ção, facilitando a identificação de falhas. Este mecanismo se baseia em um microcontrolador que recebe informações de corrente elétrica obtidas por um sensor ligado a rede e leituras de luminosidade. O microcontrolador utilizado trata-se de um ESP8266 que possui seu próprio sistema de comunicação *wi-fi*.

Nota-se que é possível utilizar de sistemas embarcados e sensores e atuadores no contexto de Internet das Coisas para construir aplicações úteis aos usuários, nos mais diversos ramos e finalidades. O ESP8266 é um microcontrolador que pode ser utilizado para este fim. Sabendo que sensores de luminosidade e irrigação podem ser acoplados no ESP8266 para medição destas variáveis e assim possibilitar a construção das aplicações, presume-se que o controle automatizado de luminosidade e irrigação pode ser aplicado aos estádios de futebol para melhorar as condições das partidas (luminosidade) e a qualidade da grama (utilizando-se de períodos corretos para os processos de irrigação).

Objetivo

O objetivo deste Trabalho de Conclusão de Curso (TCC) é o desenvolvimento de um sistema automatizado capaz de controlar os subsistemas de iluminação e irrigação presentes em estádios de futebol através da plataforma de prototipagem eletrônica arduíno, mais especi-ficamente o módulo ESP8266 - NodeMCU, bem como permitir a visualização e o controle da luminosidade e da irrigação por meio de um aplicativo para dispositivos móveis.

Objetivos Específicos

1. Realizar análise e coleta de requisitos necessários para produção do sistema proposto.

2. Identificar e conhecer as potencialidades do microcontrolador ESP8266.

3. Subdividir o sistema em módulos específicos para facilitar os processos de adminis-tração e de implementação.

4. Prototipar os módulos de automação a fim de projetar uma solução de controle de luminosidade e irrigação para estádios de futebol.

5. Produzir uma aplicação para dispositivos móveis capaz de controlar o sistema de iluminação e irrigação, bem como permitir a visualização dos dados coletados pelos sensores utilizados.

1.1 Descrição dos Capítulos

Este TCC está dividido em 5 (cinco) capítulos. Neste Capítulo foi apresentado o estado da arte sobre aplicações inteligentes que utilizam sistemas embarcados, com base em estudos realizados nos últimos 12 (doze) anos. Apresentamos também hipótese a ser validada por este trabalho e os objetivos geral e específicos que definem a finalidade e a delimitação desta pesquisa, respectivamente.

No Referencial Teórico, Capítulo 2, é realizada uma abordagem sobre as tecnologias e ferramentas utilizadas, tais como os dispositivos de *hardware* e *software*, o que possibilita melhor compreensão dos processos utilizados na construção do sistema automatizado desenvolvido.

O Capítulo 3 apresenta a classificação desta pesquisa (metodologia da pesquisa científica) e o método utilizado para construção do sistema de luminosidade e irrigação aplicado ao cenário de estádios de futebol.

Nos Resultados, Capítulo 4, são abordados os processos intrínsecos relacionados ao sistema desenvolvido, especificando os módulos que fazem parte do sistema de controle de luminosidade e irrigação. Para cada módulo, apresentamos o funcionamento geral e específico, inclusive com as prototipagens realizadas, e os parâmetros utilizados como critérios nas decisões de tomada, tais como acionamento dos processos de automatização. Neste Capítulo também são apresentados os experimentos e códigos desenvolvidos.

Por fim, apresentamos no Capítulo 5 as conclusões obtidas ao longo dos 12 (doze) meses de desenvolvimento deste trabalho, as contribuições científicas, dificuldades encontradas e sugestões para possíveis trabalhos futuros.

2

Referencial Teórico

2.1 Introdução

Este capítulo contém a explicação dos assuntos que foram importantes para o desenvolvimento deste trabalho a fim de facilitar a compreensão das decisões de tomadas. Apresentamos conceitos básicos relacionados à Internet das Coisas, elementos de *hardware* e software, bem como protocolos de comunicação típicos em aplicações inteligentes.

2.2 Internet das Coisas

Segundo (SANTOS et al., 2016) a Internet das Coisas, do inglês *Internet of Things (IoT)*, surgiu devido aos avanços dos dispositivos físicos, tais como sistemas embarcados e a microeletrônica. A IoT é uma extensão da Internet atual que proporciona os objetos presentes no nosso dia-a-dia a capacidade computacional e de comunicação através da conexão com a Internet, o que permite que os objetos físicos possam ser acessados remotamente. Com esta nova habilidade dos objetos surgiram inúmeras oportunidades, tanto no meio industrial quanto no acadêmico.

2.3 Sistemas Embarcados

De acordo com CUNHA (2007), sistema embarcado é um circuito ou um equipamento integrado com capacidade computacional. Trata-se de um sistema completo e independente preparado para realizar tarefas específicas, não sendo possível que o usuário final tenha acesso ao programa embutido neste dispositivo. Estes dispositivos, em alguns casos, não possuem flexibilidade de *hardware* e *software*, o que torna estes dispositivos incapazes de realizar tarefas diferentes daquelas que foram inicialmente programadas.

Dispositivos embarcados possuem um microprocessador ou um microcontrolador que os permitem realizar manutenção automática em seus próprios sistemas, além de tornar possível que os mesmos possam ler sinais externos, processá-los e enviá-los para atuadores. É desejável que estes sistemas apresentem sempre os menores tamanhos, pesos e eficiência no consumo de

energia, visto a limitação computacional recorrente em tais dispositivos. Dispositivos embarcados são típicos no cenário de IoT.

2.4 Dispositivos Computacionais Típicos de IoT

2.4.1 Microcontrolador Arduíno

Para (ARAUJO ELIAS et al., 2014), o arduíno é uma plataforma eletrônica projetada de forma a ser livre para baratear projetos de prototipação. Esta placa consiste em um microcontrolador, botão de *reset*, um *led* que indica o seu funcionamento e pinos analógicos e digitais que possibilitam a entrada e saída de outros dispositivos. O arduino possui uma saída *Universal Serial Bus (USB)* para que o mesmo possa ser conectado a um computador e uma saída para a fonte de alimentação. A Figura 2.1 representa uma placa arduíno do modelo UNO.

Figura 2.1: Arduíno Uno - Fonte: store.arduino.cc

2.4.2 Raspberry PI

De acordo com (CRUZ; LISBOA, 2014), o Raspberry PI é um computador pessoal, desenvolvido em 2006, no Reino Unido. A principal motivação foi o desenvolvimento de um dispositivo, com maior poder computacional quando comparado com o arduino, com preço acessivel, miniaturizado e com diversas funcionalidades capazes de se integrar facilmente ao desenvolvimento de projetos.

Esta placa possui dois modelos: o modelo “A” foi lançado com apenas uma entrada *USB* e com e 256 *Megabytes* (MB) de memória *Random Access Memory* (RAM). Já o modelo “B”, revisão dois, é uma atualização do modelo “A” pois, além de possuir funcionalidades semelhantes do modelo anterior, adiciona uma entrada USB, uma interface RJ45 e o aumento da memória RAM para 512 MB. O Raspberry, presente na Figura 2.2, diferentemente dos microcontroladores tradicionais, tais como o Arduíno, opera com o processamento sobre um Sistema Operacional que executa as ações de entrada, saída e de armazenamento.

Figura 2.2: Raspeberry PI - Fonte: Wikipedia

2.4.3 ESP8266

Segundo (DA SILVA et al., 2017) o ESP8266 é um sistema embarcado projetado como solução de comunicação em rede através do *Wi-Fi* e o protocolo *Transmission Control Protocol - Internet Protocol (TCP/IP)*. Esta placa encontra-se em diferentes modelos, porém existe apenas um tipo de processador que pode ser encontrado em todas as variações. Os modelos que possuem o microcontrolador ESP9266 se diferenciam devido a quantidade de pinos *General-purpose input/output (GPIO)* expostos e quantidade de memória *flash* disponível.

2.4.3.1 NodeMCU

O NodeMCU é um *kit* de desenvolvimento e *firmware* de código aberto, baseado no ESP8266, desenvolvido para a criação de protótipos e projetos de IoT. Possui uma interface USB-serial, um regulador de tensão 3.3 *Volts* e uma grande variedades de GPIOs, como pode ser visto na Figura 2.3. O NodeMCU possui apenas uma porta analógica. Para aumentar a quantidade de portas desse tipo é necessário utilizar um multiplexador (CAETANO; SOUZA GONÇALVES, 2019).

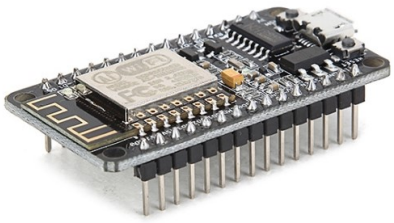

Figura 2.3: NodeMCU-ESP8266 - Fonte: filipeflop.com

2.4.4 Sensores/Atuadores

Sensores podem ser definidos como dispositivos utilizados para identificar, medir ou gravar fenômenos físicos tais como calor, radiação, luminosidade, umidade, entre outras, e que respondem transmitindo informações, iniciando mudanças ou operando controles. Estes dispositivos mudam seu comportamento sob a ação de uma grandeza física, podendo fornecer diretamente ou indiretamente um sinal que indica esta grandeza. (CAVALCANTI et al., 2018). Note na Figura 2.4 a ilustração de alguns sensores tipicamente utilizados em conjunto com os microcontroladores e sistemas embarcados.

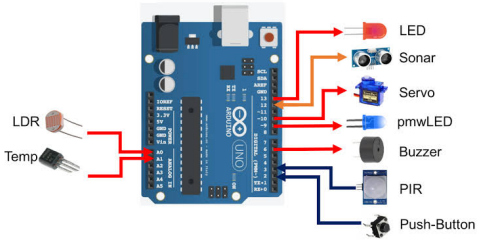

Figura 2.4: Sensores típicos em aplicações IoT - Fonte: pbx-brasil

2.4.4.1 Relé

O relé é um dispositivo comutador eletromecânico de extrema importância no ramo da robótica, domótica, eletrônica, mecatrônica e áreas afins. Note na Figura 2.5 a ilustração de um módulo relé tipico para Arduíno. O relé tem seu funcionamento baseado nos princípios eletromagnéticos: o interior de um indutor é composto de uma bobina de cobre que gera um campo magnético. Quando o relé está desativado ou nenhum pulso elétrico é fornecido, seus braços estão em uma posição que é conhecida como normalmente aberta. Quando o relé está ligado ou um pulso elétrico é enviado, o braço metálico se move em direção ao outro contato físico do relé possibilitando a passagem da energia elétrica. (ALMEIDA, 2010).

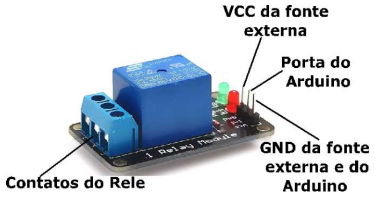

Figura 2.5: Módulo Relé - Fonte: oficinadanet.com.br

2.4.4.2 Sensor de Luminosidade - LDR

O *Light Dependent Resistor (LDR)* é um componente que tem uma resistência que se altera de acordo com a variação de luminosidade incidente sobre o mesmo. A relação entre resistência e intensidade de luz é inversamente proporcional, ou seja, a resistência aumenta

conforme a incidência de luz diminui sobre o dispositivo . Veja na Figura 2.6 a ilustração de um sensor de luminosidade LDR e suas especificações.

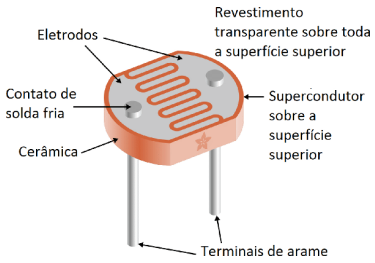

Figura 2.6: Sensor de luminosidade LDR - Fonte: researchgate.net

2.4.4.3 Sensor de Umidade do Solo - Higrômetro

O higrômetro é um instrumento utilizado para medir a umidade de gás ou vapor na atmosfera. Foi criado em 1820 pelo cientista inglês John Frederic Daniell e é feito de substâncias capazes de absorver a umidade atmosférica, como o cabelo humano (ARAUJO ELIAS et al., 2014). No caso do Higrômetro, utilizado em Arduíno, é possível detectar as variações de umidade presentes no solo. Observe na Figura 2.7 um sensor de umidade higrômetro.

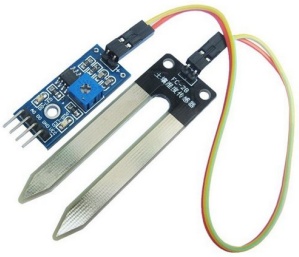

Figura 2.7: Sensor de umidade Higrômetro - Fonte: arduinolandia.com

2.4.4.4 Sensor de Umidade e Temperatura - DHT11

Os sensores de temperatura e de umidade DHT11 são construídos em 2 (duas) partes para possibilitar a medição de níveis de umidade e temperatura. O sensor possui um *chip* interno que faz uma conversão de um sinal analógico para digital e então o retorna com a temperatura e a umidade. Este sinal emitido pode ser lido com um microcontrolador. DHT11 possui maior precisão em leituras de umidade de 20% a 80%, podendo variar em 5%. No caso da temperatura, são indicados para leituras entre 0○ e 50○C, com precisão de 2 ○C para mais ou menos (GOMES, 2016). Note na Figura 2.8 o sensor de umidade e temperatura, modelo DHT11.

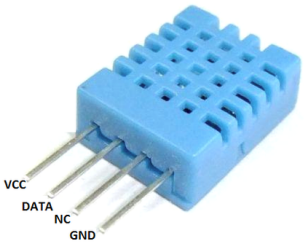

Figura 2.8: Sensor de umidade e temperatura DHT11 - Fonte: indiamart.com

2.4.4.5 Sensor de Nível de Água/Boia

Segundo (QUINTANILHA; ESTEVÃO FILHO, 2013), a versão mais simples deste sensor é a do tipo boia flutuante no qual uma chave interna abre ou fecha em um contato dependendo do nível de água. Quando o nível da água aumenta, a boia deste sensor é movida fazendo com que a mesma se feche. Desta forma, o circuito de alimentação da boia é acionado permitindo que a mesma conduza uma corrente elétrica entre sua entrada e saída. Quando o nível da água diminui, a boia é impulsionada naturalmente para baixo e assim não haverá passagem de energia entre os pontos de entrada e saída. Note na Figura 2.9 a ilustração de um sensor boia eletromagnético tipicamente utilizado em conjunto com microcontroladores.

Figura 2.9: Sensor de Nível de Água/Boia - Fonte: robohelp.com

2.4.4.6 Multiplexador

Um multiplexador é um dispositivo que tem por intuito reunir informações de duas ou mais fontes de dados em um único canal. São normalmente utilizados em situações no qual o dispositivo utilizado possui apenas um canal de entrada de dados, porém se encontra conectado a outras fontes de dados. Os multiplexadores comumente utilizados em circuitos eletrônicos possuem formas de montagem e implementação que permitem a escolha, dentre as fontes de dados conectadas, qual será utilizada pelo canal de entrada de dados do dispositivo. Veja na Figura 2.10 a ilustração de um Multiplexador comumente utilizado em microcontroladores

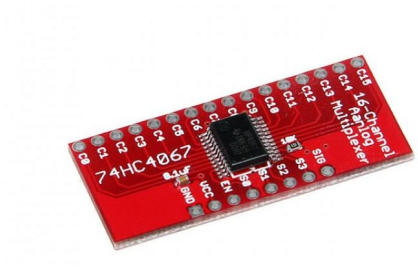

Figura 2.10: Multiplexador - Fonte: geeetech.com

2.5 Protocolos de Comunicação

2.5.1 HTTP

Hypertext Transfer Protocol (HTTP) é o protocolo básico da *World Wide Web (WWW)* e pode ser usado em qualquer aplicação cliente/servidor que envolva hipertexto (SOUZA SILVA; SILVA, 2009).

A confiabilidade das informações em uma transmissão HTTP é obtida com o protocolo *Transmission Control Protocol (TCP)*. Contudo, HTTP é um protocolo sem estado, ou seja, cada comunicação é tratada de forma independente. (SOUZA SILVA; SILVA, 2009).

2.5.2 MQTT

De acordo com (TORRES; ROCHA; DE SOUZA, 2016), o *Message Queuing Telemetry Transport (MQTT)* é um protocolo de mensagens aberto feito para operar em dispositivos com comunicação baseada em *Machine to Machine* (M2M). Este protocolo tem a capacidade de tolerar redes de alta latência, comunicação instável e baixa largura de banda.

O MQTT adota TCP e seu padrão de mensagens é o *publisher/subscriber* (publicador/-consumidor), no qual todos os dados são enviados para um intermediário, denominado *broker*, que se encarrega de enviar as mensagens aos destinatários corretos.

Esta estrutura permite o desacoplamento entre produtor e cliente. Dessa forma, apenas o endereço do *broker* precisa ser conhecido, possibilitando os variáveis estilos de comunicação: um-para-um, um-para-muitos ou muitos-para-muitos.

2.5.2.1 *Broker*

De acordo com (VELOSO et al., 2018), *broker* é um serviço na nuvem, responsável por receber ou filtrar as mensagens que serão encaminhadas ou enviadas a todos os *subscribers* (assinantes), de acordo com os interesses expressados nas assinaturas. O *broker* pode ser identificado por meio de uma conexão convencional, tais como login e senha. Desta forma, o dispositivo, sendo publicador ou subscritor, pode-se conectar para comunicar ou trocar dados com outros dispositivos IoT. Cada mensagem enviada pelo *publisher* (publicador) deve conter

um tópico, que será usado pelo *broker* para encaminhar a mensagem aos assinantes.

Segundo (CONCEIÇÃO; RESENDE COSTA, 2019) o CloudMQTT é um exemplo de *broker*. Assim que registrado no sistema, o usuário pode criar diversas instâncias, sendo que cada uma corresponde a um *broker*. Também pode ser especificado outros usuários no âmbito do acesso e com regras específicas, inclusive com a utilização de certificados digitais para validação de autenticação. Os clientes conectados ao *broker* podem ser acompanhados em tempo real.

2.5.2.2 MQTT *Dash*

Segundo (BORBA, 2018), MQTT *Dash* é uma aplicação que funciona como um cliente MQTT e que permite a visualização dos dados monitorados de diferentes maneiras. (PITON, 2017) aborda que neste aplicativo é possível criar painéis com funções de botões, barras ou textos que podem ser utilizados para enviar e receber mensagem via MQTT. Note na figura 2.11 a ilustração do mesmo.

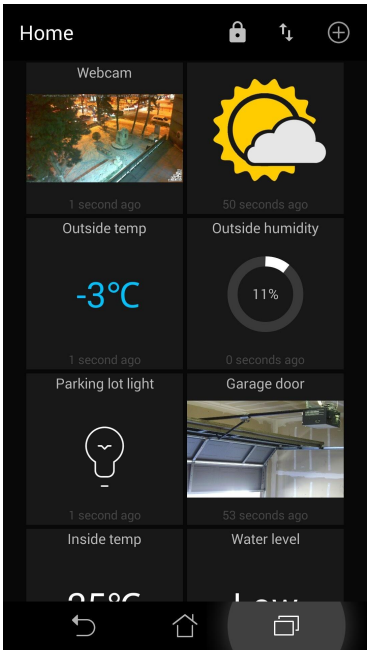

Figura 2.11: MQTT Dash - Fonte: apkpure.com

2.5.3 ZigBee

Zigbee é um conjunto de especificações, desenvolvido pela Zigbee Alliance, para utilização em *smart-home* e aplicações de IoT que definem as camadas subsequentes às camadas estabelecidas pelo *Institute of Electrical and Electronic Engineers (IEEE)* 802.15.4, oferecendo serviços de segurança, tolerância a erros e conexão de novos dispositivos. (ROTTA; CHARÃO; DANTAS, 2017)

2.6 *Softwares* para Prototipagem

2.6.1 TinkerCad

Desenvolvida pela empresa Autodesk, o TinkerCad é uma ferramenta *online* gratuita para *design* de modelos 3D. A mesma possibilita a montagem e simulação de circuitos elétricos, analógicos e digitais. Por de fácil uso, é possível desenvolver diversos projetos tais como desenhos 3D e montagem de componentes embarcados.

2.6.2 Fritzing

Desenvolvido na Universidade de Ciências Aplicadas de Potsdam, o Fritzing é um *software* livre para *design* de *hardware* eletrônico bastante utilizado para modelar circuitos baseados em Arduíno e microcontroladores, ou somente a matriz de contatos e alguns componentes eletrônicos. Com este *software* é possível desenhar de forma detalhada um circuito completo pois o mesmo transforma todo esquema desenhado em um diagrama elétrico/*layout* de uma placa de circuito impresso.

2.7 Linguagens de computadores

Linguagens de computadores são códigos que consistem em uma sequência de *bytes* que correspondem a instruções a serem executadas pelo processador.

2.7.1 Linguagem de Programação

Linguagem de programação é uma forma padronizada de se comunicar com o computador seguindo regras de sintaxe e semântica no desenvolvimento de um programa computacional. Com uma linguagem de programação é possível especificar detalhadamente sobre quais dados o computador deve atuar, ou seja, como o mesmo deve funcionar.

Linguagens de programação são bastantes utilizadas para expressar algoritmos com precisão. Portanto, são ferramentas que auxiliam tanto programadores quanto engenheiros de *software* a desenvolver programas com agilidade e organização.

2.7.1.1 C/C++

Segundo (SANTEE, 2005) a linguagem C foi criada em 1972, por Dennis M. Ritchie e Brian W. Kernighan, nos laboratórios da *Bell Labs Innovations*. A linguagem C é leve, poderosa e possui fácil interpretação.

Já a linguagem C++ foi integrada em 1983 por Bjarne Stroustrup, com novos e poderosos elementos e novas propostas para a programação. Esta linguagem possui várias bibliotecas-padrão tendo em grande parte as características exclusivas da linguagem C.

2.8 Banco de Dados

Um banco de dados é uma coleção lógica e coerente de dados com algum significado inerente onde os dados são fatos que podem ser gravados e que possuem um significado implícito. (ELMASRI et al., 2005).

2.8.1 SGBD

Um sistema gerenciador de banco de dados (SGBD) é uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é, portanto, um sistema de *software* de propósito geral que facilita os processos de definição, construção, manipulação e compartilhamento de bancos de dados entre vários usuários e aplicações. (ELMASRI et al., 2005).

2.8.1.1 FireBase

O FireBase é um banco de dados na nuvem, disponibilizado pela Google, que utiliza a estrutura *Not Only SQL (NoSQL)* orientada a documentos. A orientação a documento, também conhecida como semi-estruturada, é um banco de dados que utiliza *JavaScript Object Notation (JSON)* para o salvamento e busca de dados. (MEZZARI, 2019).

Firebase é uma plataforma para armazenamento e sincronização de dados em tempo real. Provê uma variedade de soluções de desenvolvimento para acelerar a integração de recursos baseados em nuvem em aplicativos móveis e *Web*. (SILVA, 2018).

2.9 IDE

De acordo com OLIVEIRA et al. (2012), um *Integrated Development Environment (IDE)* ou Ambiente Integral de Desenvolvimento é uma ferramenta que incorpora recursos necessários para criação de *software* em um linguagem de programação específica. Os principais recursos oferecidos por um IDE são os editores de programas, ambiente de execução, ambiente de depuração, gerenciamento de projetos, entre outros.

2.9.1 IDE para Arduíno

A IDE do Arduíno é uma aplicação desenvolvida na linguagem de programação Java usada principalmente para escrever e enviar programas para placas do tipo Arduíno mas também suporta outras placas de desenvolvimento por meio de núcleos disponibilizados por terceiros. Note na Figura 2.12 a interface da IDE para Arduíno.

A IDE do Arduíno suporta as linguagens de programação C e C++ desde que sejam utilizadas regras especiais na estruturação do código. Esta IDE fornece uma biblioteca de *software* do projeto *Wiring*, que traz muitos procedimentos comuns de entrada e saída. O código

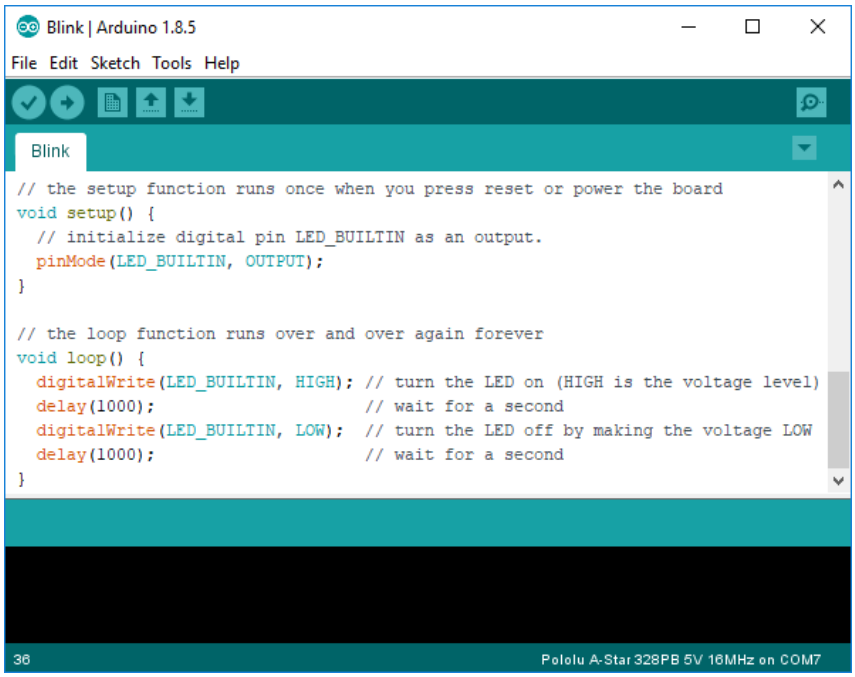

Figura 2.12: Ambiente de desenvolvimento para Arduíno - Fonte: pololu.com

gravado pelo usuário requer apenas duas funções básicas, o *sketch* e o *loop* do programa principal, que são compilados e vinculados a um *stub* do programa *main()* em um programa executivo cíclico executável com a cadeia de ferramentas GNU's Not Unix (GNU), também incluída na distribuição da IDE.

A IDE utiliza o programa *avrdude* para converter o código executável em um arquivo de texto com codificação hexadecimal que é carregado na placa do Arduíno por um programa no *firmware* da placa.

2.10 Tipos de Equipamentos de Iluminação

2.10.1 Projetores

Os projetores recomendados para os projetos de iluminação em estádios de futebol são os de fecho direto regulável com alta capacidade de foco e que possuam regulagem de fechos, o que facilita o direcionamento de iluminação em áreas com excesso ou falta de iluminação.

Ao se projetar o sistema de iluminação de um estádio deve-se optar por projetores de feixe direto. A tecnologia desses projetores facilita a adaptação em locais com diferentes tipos de montagem de coberturas ângulos e altura de cada projeto (DIAS, 2018).

2.10.2 Lâmpadas

Para a definição das lâmpadas para o projeto de iluminação para estádios de futebol devem ser levados em conta o tempo de vida, consumo de energia, luminosidade de saída, temperatura de cor e índice de reprodução das cores. A escolha dos equipamentos auxiliares

corretos é crucial para o correto acendimento das lâmpadas e permanecer o fluxo de luminosidade ideal (DIAS, 2018).

2.11 Tipos de Equipamentos de Irrigação

2.11.1 Aspersores

De acordo com (NETO, 2003), dentre os vários tipos e modelos de dispositivos utilizados na irrigação, destacam-se os aspersores. Os principais tipos são os aspersores *spray* e aspersores rotores.

2.11.1.1 Aspersores *Sprays*

São utilizados em pequenas áreas com bordas fechadas que necessitem de um direcionamento preciso de água, tais como áreas com vegetação densa que atrapalham significantemente a superposição de cobertura de rotores e para áreas com grande variedade de plantas que necessitam de diferentes quantidades de água.

2.11.1.2 Aspersores Rotores

Disponíveis em duas versões: a versão aparente é utilizada em grandes áreas de arbustos ou com alguma cobertura vegetal de alta densidade de plantio. Já a versão escamoteável é bastante utilizada em gramados e coberturas de pequeno porte.

2.11.2 Válvula Solenoide

Segundo (POUSO, 2012), a válvula solenoide é a combinação de duas unidades funcionais: o pacote eletromagnético que é constituído por um solenoide e seu correspondente núcleo móvel. O solenoide é uma bobina de fio isolado, energizada eletricamente para produzir um campo magnético no seu interior e que provoca um movimento mecânico em um núcleo ferromagnético. Quando a bobina está energizada, o núcleo está em uma posição, quando desenergizada, o núcleo está em outra posição, provocando assim o sistema de abertura e fechamento. Note na Figura 2.13 a ilustração de uma válvula solenoide.

O corpo de uma válvula pode conter um ou vários orifícios de entrada, passagem e saída. Estes orifícios permitem ou não a passagem de um fluído quando a haste é acionada pela força da bobina. Esta força faz com que o pino seja puxado para o centro da bobina, o que permite a passagem do fluído. Para o fechamento de uma válvula solenoide, uma perda de energia é necessária, o que faz com que o pino perca sua pressão. Assim, deste modo, a válvula é fechada.

O corpo da válvula possui um dispositivo que permite a solenoide ser aberta ou fechada pelo movimento do núcleo, que é acionado na solenoide quando a bobina é energizada e é usada para controlar a vazão de fluidos, principalmente no modo liga-desliga. Estas válvulas ou são do

Figura 2.13: Válvula solenoide - Fonte: autocorerobotica.com

tipo normalmente fechadas, que se abrem quando uma corrente é aplicada e se fecham caso esta corrente seja cortada ou do tipo normalmente aberta, que trabalham de forma inversa a anterior.

2.11.3 Bomba d'Água/Motobomba

Um dos itens presentes em diversos sistemas de irrigação são as motobombas, equipamentos que fornecem a energia necessária ao líquido para que este possa chegar até os emissores com a pressão de serviço requerida para o funcionamento ideal dos mesmos. Atualmente, as bombas d'águas/motobombas mais utilizadas são as bombas centrífugas movidas por motor elétrico (GRAH; BOTREL; DE MATOS, 2012). A Figura 2.14 apresenta a ilustração de uma bomba d'água aplicada em um cenário real.

Figura 2.14: Bomba d'água - Fonte: youtube.com/slideshow

As motobombas ou bombas d'água são máquinas acionadas que recebem energia de uma fonte motora, transformam em energia cinética e energia de pressão e a transmitem ao fluido bombeado. A utilização de bombas ocorre sempre que se necessita aumentar a pressão de um fluido, transportá-lo pela tubulação de um ponto a outro , respeitando as condições de vazão e pressão pré-estabelecidas. (GOUVEA, 2008).

2.12 Considerações Finais

Neste Capítulo foram apresentados conceitos inerentes ao contexto de IoT, tais como sistemas embarcados, dispositivos computacionais, protocolos de comunicação e *softwares* para prototipagem de circuitos eletrônicos. Foram abordados também conceitos clássicos de computação, tais como linguagens de programação, banco de dados e interfaces de desenvolvimento. Por fim, conceitos relacionados aos processos de aluminação e irrigação foram esclarecidos e as principais ferramentas utilizadas nestes processos foram exemplificadas.

3

Material e Métodos

3.1 Introdução

Neste Capitulo é abordado a metodologia de pesquisa científica utilizada no desenvolver deste trabalho e também as ferramentas e os métodos utilizados para a construção de um sistema de iluminação e irrigação para estádios de futebol.

3.2 Metodologia de Pesquisa

Em relação ao método de pequisa, este Trabalho de Conclusão de Curso (TCC) utilizou-se do método hipotético-dedutivo no qual são utilizadas possíveis soluções para resolver os problemas encontrados no decorrer da pesquisa. Partimos de premissas verdadeiras, tais como o uso de sistemas embarcados em IoT e a conexão de sensores e atuadores em dispositivos embarcados, para a formulação de uma hipótese e assim ser possível propor uma solução automatizada de um sistema de iluminação e irrigação aplicada a estádios de futebol.

Em relação a modalidade da pesquisa, aplicou-se a pesquisa explicativa, o que torna possível estruturar os modelos teóricos criados para o sistema proposto no qual as hipóteses serão relacionadas de forma que seja possível analisar os problemas de forma unitária. Através de métodos experimentais foi possível identificar fatores que contribuem para ocorrência dos fenômenos estudados. É possível que a pesquisa possa ser considerada em algumas partes como qualitativa, por conter descrições intuitivas do autor/pesquisador.

Optou-se também em utilizar a modalidade de pesquisa bibliográfica no qual são consideradas as contribuições cientificas realizadas sobre o tema central deste trabalho. Dessa forma, foram consideradas as contribuições nos últimos 12 (doze) anos.

A pesquisa bibliográfica permitiu o aprofundamento dos conhecimentos necessários para a realização e construção do sistema de controle de iluminação e irrigação para estágios de futebol. Portanto, bases eletrônicas, como *"SCOPUS"*, *"IEEE Xplore Digital Libray"*, *"ACM Digital Library"*, sites de busca e repositórios de Universidades foram utilizadas para construção do acervo bibliográfico necessário para a realização da pesquisa.

A pesquisa exploratória teve como objetivo gerar conhecimento relacionados ao ESP8266, sensores utilizados no projeto, montagem de circuitos digitais, uso de *softwares* de simulação de circuitos digitais e linguagens de programação utilizadas para programação dos dispositivos embarcados.

3.3 Ferramentas utilizadas

Para construção das figuras do funcionamento geral do sistema em um possível ambiente real foi utilizado o *software* de modelagem AutoDesk Tinkercad, em sua versão *online*, disponível em: www.tinkercad.com. O Banco de dados utilizado foi o FireBase, em sua versão *online*, disponível em: firebase.google.com. O *broker* utilizado para conectar os módulos e permitir a troca de informações sensíveis no projeto foi o CloudMQTT, em sua versão *online*, disponível em: api.cloudmqtt.com

Todos os módulos baseados no microcontrolador(NodeMCU) foram desenvolvidos na linguagem de programação C++ utilizando como ferramenta a "Arduíno IDE", na versão 1.8.9.

Já os desenhos esquemáticos de montagem dos circuitos foram realizados através do uso do *software* Fritzing, na versão 0.9.3-x64.

O módulo de gerenciamento do sistema foi desenvolvido utilizando o aplicativo MQTT *Dash*, na versão 4.4. O MQTT *Dash* operou em um Sistema Operacional Android, na versão 6.0.1.

O Sistema Operacional utilizado para o uso destas ferramentas, com exceção do MQTT *Dash*, foi o Windows 8.1 X64.

Primeiramente, desenvolvemos uma figura de propósito geral do funcionamento do sistema de iluminação e irrigação aplicados a estádios de futebol. Essa percepção inicial possibilitou que o sistema fosse subdividido em módulos, com intuito de prover facilidade no entendimento dos processos e de implementação, a citar: módulo de controle da bomba d'água/motobomba, módulo de controle da irrigação, módulo de controle da iluminação e o módulo de gerenciamento do sistema.

Cada um dos módulos foram desenvolvidos individualmente e posteriormente integrados para a composição da solução geral. Apresentamos, assim, uma solução integrada responsável pelo controle da iluminação e irrigação em estádios de futebol, utilizando o microcontrolador ESP8266 (NodeMCU) e dispositivos típicos presentes no cenário de IoT, tais como sensores e atuadores. Os detalhes do funcionamento dos módulos do sistema de iluminação e irrigação encontram-se no Capítulo 4

3.4 Considerações finais

Neste Capítulo foram abordados os métodos e ferramentas utilizados no desenvolver deste trabalho, além de apresentarmos a classificação desta pesquisa científica. Realizamos

uma descrição breve, porém suficiente para possibilitar a repetição para aqueles interessados no fenômeno observado.

4

Resultados

4.1 Introdução

Este Capitulo é responsável por abordar os módulos de composição do sistema de iluminação e irrigação proposto. Neste momento, descrevemos a a funcionalidade e as particularidades de cada um dos módulos, em termos de uso de dispositivos de *hardware*, e os códigos utilizados para o funcionamento correto do sistema. Apresentamos a modelagem geral e a prototipagem dos circuitos responsáveis pelo sistema automatizado de iluminação e irrigação de estádios de futebol.

4.2 Visão Geral do Sistema

No desenvolver deste sistema procurou-se atender as necessidades das pessoas responsáveis por administrar os processos de iluminação e irrigação em estádios de futebol. Dentre estas necessidades, destaca-se a necessidade de automatização dos processos de iluminação e irrigação.

Desta forma, o sistema proposto consiste na divisão dos processos de controle de iluminação e irrigação em 4 (quatro) módulos distintos, a citar: controle de irrigação, controle da bomba d'água, controle da iluminação e o módulo para gerenciamento do sistema e visualização das informações.

Todos os módulos, exceto o de gerenciamento do sistema, possuem seus funcionamentos baseados no microcontrolador (NodeMCU), que por sua vez é conectado a um servidor *broker*, por meio de seu componente integrado de comunicação em rede (ESP8266). Assim, deste modo, cada módulo baseado no NodeMCU se comunica com os demais.

A comunicação entre os módulos deste sistema é feita por meio de troca de mensagens, utilizando-se o protocolo MQTT, o que torna possível a comunicação dos mesmos em tempo real.

Os módulos foram desenvolvidos considerando a existência um sistema de irrigação clássico, no qual uma motobomba impulsiona a água presente em um reservatário para os irrigadores presentes no gramado. Também é considerada a existência dos equipamentos de iluminação tradicionais, tais como lâmpadas e refletores, acionados de forma mecânica por

interruptores ou chaves.

4.2.1 Módulo de Controle da Bomba D'água/Motobomba

Este módulo tem por objetivo controlar de forma eficaz o funcionamento da bomba d'água ou motobomba. Isto se faz necessário pois esta é a parte responsável por impulsionar a água até os irrigadores, ou seja, sem a mesma não há irrigação. O termo Motobomba e bomba d'água serão frequentemente utilizados no mesmo contexto pois se tratam do mesmo componente.

Este módulo foi desenvolvido sobre a plataforma de prototipagem NodeMCU que contou com o uso de sensores e atuadores para tornar possível a validação deste módulo. Sendo a bomba d'água um componente que necessita de interação humana de forma direta para operar de forma correta, logo há necessidade de automatizar tal processo. Diversos fatores que devem ser levados em conta para que o acionamento da motobomba seja feito de forma correta, neste sentido foi utilizado o NodeMCU para determinar o acionamento correto e eficiente da bomba d'água.

Sempre que solicitado o acionamento da motobomba pelo módulo de controle da irrigação são adotados alguns procedimentos para o correto funcionamento. Para que a bomba d'água funcione durante o processo de irrigação, é necessário que haja água suficiente no reservatório que alimenta a mesma. Desta forma, utilizamos um sensor de nível de água, também conhecido por sensor-boia, que tem por objetivo informar se o nível da água presente no reservatório é considerado suficiente para o acionamento da bomba d'água. Caso o nível da água esteja abaixo do estipulado, os módulos que apresentem interesse nesta informação irão receber esta informação até que a água seja reposta suficientemente para que a motobomba funcione.

Com o nível da água satisfatório para o acionamento da motobomba é possível liberar o fluxo de água para a mesma. Para a liberação do fluxo de água, foi utilizada uma válvula solenoide que atua entre o reservatório e a motobomba. Esta válvula é ligada a uma fonte externa de energia e acionada por um interruptor elétrico ligado diretamente ao NodeMCU.

Ao liberar o fluxo de água para a motobomba resta apenas realizar seu acionamento. Para isto, foi utilizado um relé. O desligamento da bomba d'água se dá por outros fatores: o primeiro deles é o de solicitação, ou seja, caso o módulo de controle da irrigação solicite o desligamento da mesma.

O segundo fator para desligamento da bomba d'água é condicionado ao nível da água presente no reservatório. Caso o nível de água no reservatório esteja muito abaixo, há possibilidades da motobomba funcionar sem alimentação de água, o que pode ocasionar comprometimento no funcionamento da motobomba.

Nestes casos é realizado apenas a interrupção do fluxo de água e o desligamento da motobomba, necessariamente nesta ordem, para garantir que a bomba d'água não opere sem água. O desligamento é feito de forma inversa ao acionamento, ou seja, interrompe-se primeiramente o funcionamento da válvula solenoide e em seguida o funcionamento da motobomba.

A válvula solenoide utilizada é do tipo normalmente fechada, no qual a mesma só se abre possibilitando a passagem do fluxo de água caso seja alimentada por uma determinada voltagem. O processo de funcionamento da motobomba é similar: uma fonte de energia permite o acionamento da motobomba. Para chavear o funcionamento, tanto da bomba d'água quanto da válvula solenoide, foi utilizado um relé de dois canais, sendo que cada canal funciona como um interruptor: um para válvula e o outro para a bomba d'água. Note na Figura 4.1 o modelo proposto de funcionamento deste módulo.

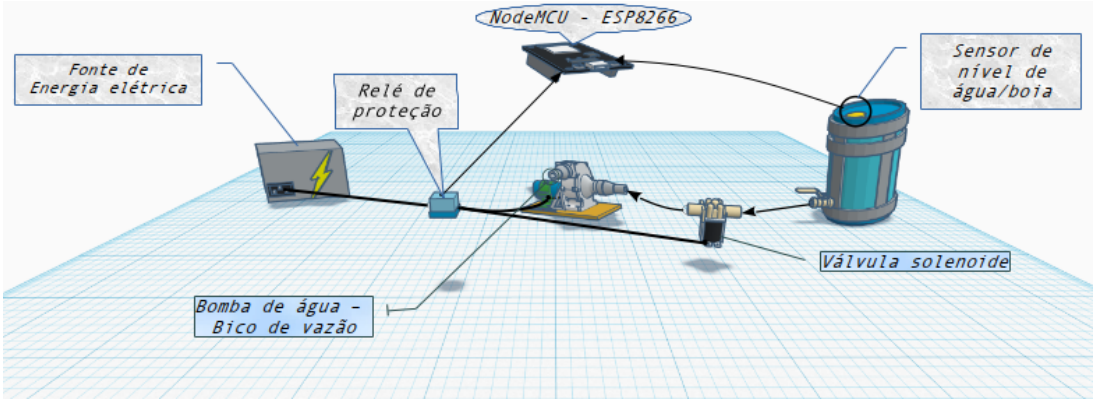

Figura 4.1: Módulo de Controle da Bomba D'Água/Motobomba

4.2.2 Módulo de Controle da Irrigação

Este módulo tem por objetivo controlar de foma eficiente a irrigação do gramado presente em um estádio de futebol. Este módulo é importante para que a lavoura gramada se desenvolva corretamente dentro de suas condições ideais. Este módulo foi construído, por base no micro-controlador NodeMCU, e seu funcionamento consiste basicamente em verificar as variáveis de umidade e temperatura do local a ser irrigado.

Caso seja necessário o início do processo de irrigação, o módulo de controle de irrigação solicita ao módulo de controle da bomba d'água o acionamento da mesma. Ao término do processo de irrigação, é solicitado ao mesmo módulo o desligamento da motobomba.

Para realizar a irrigação ideal do gramado torna-se necessário considerar qual tipo de grama será irrigada e quais os fatores de irrigação serão adotados. Neste trabalho, adotamos como os fatores inerentes a irrigação os valores mínimos e máximos de temperatura e umidade ideal do solo em que se encontra o gramado para assim definir os fatores que determinarão o funcionamento ideal do sistema de irrigação.

Segundo GOLOMBEK (2006), cada espécie de grama possui tolerâncias diferentes tanto ao frio quanto ao calor. Temperaturas acima de 51 ∘C e abaixo de 0 ∘C ocasionam a baixa atividade metabólica do gramado, sendo a faixa normal para o crescimento valores entre 4 ∘C e 40 ∘C. A temperatura medida no nível do solo, considerada ótima para o crescimento da grama, oscila entre 15,5 ∘C e 24 ∘C para gramados de clima frio. Para os gramados de clima quente o ideal são temperaturas de 27 ∘C a 35 ∘C. Para raízes de gramas predominantes em climas

quentes, os valores considerados ótimos para o crescimento estão entre 24 ∘C e 30 ∘C, sendo essas temperaturas medidas a 15 cm de profundidade do solo da grama.

De acordo com GURGEL (2003), as gramas que mais se acomodam ao Brasil são as especies de clima quente. Esse tipo de grama se divide em dois grandes grupos: rizomatosas e estoloníferas. As gramas de variedade rizomatosas são as mais indicadas para gramados esportivos devido ao fato de serem a base do crescimento vegetativo e se encontrarem enterradas em subsuperfícies, mantendo desta forma maior proteção e resistência a danos mecânicos diretos.

Alguns fatores específicos devem ser considerados no ato da irrigação. Um destes fatores é Evapotranspiração (ET), ou seja, a perda de água de um ecossistema para a atmosfera, causada pela evaporação a partir do solo e pela transpiração das plantas.Este fator está diretamente ligado ao clima local, no qual é possível definir a quantidade de água ideal a ser aplicada em determinada cultura a fim de manter o bom desenvolvimento da mesma.

Desta forma, durante o processo de irrigação, para cada planta, deve ser considerado a quantidade de água perdida por ET. Os valores de ET dependem da coleta destes dados por estações meteorológicas. Baseando-se em alguns cálculos é possível estimar a quantidade de água ideal para cada planta, sendo necessário também considerar que cada cultura exige uma quantidade específica de água.

Segundo (BIRD, 2013), o método tradicional para irrigação de gramados esportivos se baseia em dados de ET obtidos por estações meteorológicas. Alguns estádios de futebol possuem seu próprio centro de meteorologia, o que torna possível o processo de irrigação do gramado com maior exatidão. Neste trabalho não serão considerados fatores de ET pois os mesmos dependem de dados gerados por centros de meteorologia.

Adotamos a perspectiva de embasamento em valores fixos, ou seja, não há integração do sistema de irrigação com um centro de meteorologia. Portanto, o módulo de irrigação baseia-se em valores obtidos apenas através de seus próprios componentes sensoriais.

4.2.2.1 Irrigação Automatizada

Considerando os fatos expostos, a temperatura ideal para raízes dos gramados estão entre 24 ∘C e 35 ∘C. Por estes valores serem medidos a uma certa profundidade do solo, acredita-se ser possível controlar o nível máximo de temperatura ideal do gramado, impedindo que o mesmo ultrapasse este valor.

O controle da temperatura máxima do gramado é obtido através do ato da irrigação. Ao molhar o gramado é esperado que a temperatura do mesmo diminua. Para verificar estes valores foi utilizado um sensor de temperatura, instalado a uma profundidade de 15 cm do nível da grama. Utilizamos como sensor de temperatura o DHT11, conectado ao microcontrolador NodeMCU, com o intuito de captar a temperatura percebida, em ∘C. Em cenários reais é indicado o uso de sensores de temperatura mais robustos próprios para o solo.

Com os valores de temperatura coletados torna-se possível definir os fatores ideais de

irrigação. Se o valor de temperatura estiver acima de 35 ∘C, inicia-se o processo de irrigação. Já se o valor de temperatura estiver abaixo de 24 ∘C, a irrigação deve ser interrompida. Utilizando-se destes parâmetros, o NodeMCU, no qual o sensor de temperatura está conectado, publicará mensagens contendo as informações geradas e solicitará, o acionamento da bomba d'água, caso a temperatura ultrapasse o valor máximo definido ou o desligamento da mesma caso o valor mínimo de temperatura seja atingido, mantendo desta forma a faixa ideal de temperatura para o gramado.

Note que a hipótese sobre a temperatura das raízes do gramado só é válida na tentativa de diminuir a temperatura. Nos casos em que as raízes da grama apresentarem valores abaixo da ideal, o sistema não provê formas de aumentar a temperatura. Porém, o usuário do sistema é informado do evento (temperatura mínima atingida) para que o mesmo possa consultar um especialista em gramados e assim adotar medidas que mantém a temperatura na faixa ideal.

Outro fator a ser considerado é a umidade. Como não foram encontrados valores fixos julgados ideais, serão considerados níveis de umidade medidos por um sensor. Como dito anteriormente a temperatura ideal para um gramado é medida a 15 cm de profundidade do nível do solo pois nesta profundez se encontram as raízes da grama, sendo assim é levantada a hipótese de que se o gramado apresenta melhor desenvolvimento estando úmido e suas raízes se encontram a 15 cm de profundidade, logo acreditasse que ao manter esta profundidade úmida é possível obter um bom resultado no desenvolvimento da grama.

Sendo assim, utilizamos um sensor de umidade do solo (higrômetro), conectado a um microcontrolador NodeMCU. Este sensor foi posto em operação a uma profundidade de 15 cm do nível do solo, local onde se encontram as raízes do gramado. Com este sensor foi possível medir o quão úmido estão as raízes do gramado. Deste modo, é possível determinar a irrigação do local baseando-se nos valores coletados também por este sensor.

Com os valores de umidade devidamente coletados, torna-se possível definir qual momento apropriado para iniciar ou interromper a irrigação. Estes valores baseiam-se na capacidade de medição do higrômetro. Este sensor, em operação, retorna valores analógicos que variam entre 0 e 1023.

Ao converter esta capacidade de medição em porcentagem (%) são obtidos níveis de umidade em sua forma padrão. Neste trabalho foram considerados valores abaixo ou equivalentes a 50% resultantes a um solo pouco úmido, sendo necessário o início do processo de irrigação. Para os valores entre de 80 e 100%, equivalentes umidade máxima, deve ser solicitado o desligamento da irrigação.

Ao atingir a umidade mínima, este módulo tem a função de enviar uma mensagem para o módulo de controle da bomba d'água/motobomba solicitando a operação da mesma. Desta forma, o sistema de irrigação será acionado. Durante este procedimento, os valores de umidade do solo serão alterados devido ao processo de irrigação. Ao atingir valores superiores a umidade máxima, o módulo de controle da irrigação enviará uma mensagem ao módulo controle da bomba d'água/motobomba solicitando seu desligamento. Note a Figura 4.2 a representação do

módulo de controle de irrigação em conjunto com o próximo módulo a ser abordado, o módulo de controle da iluminação.

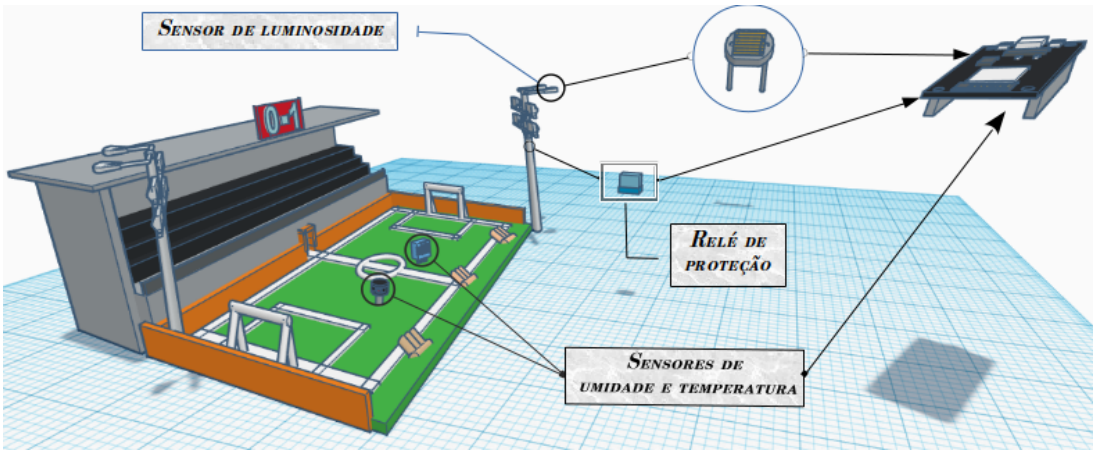

Figura 4.2: Funcionamento Geral dos Módulos de Controle de Irrigação e Iluminação

4.2.2.2 Irrigação Manual

Embora o intuito deste módulo seja a irrigação feita de forma autônoma, também é possível que este processo seja realizado de forma manual. A irrigação manual pode ser realizada através do módulo de gerenciamento do sistema. Neste caso, os valores de umidade e temperatura não implicarão no funcionamento da irrigação, apenas serão utilizados como dados informativos para que o usuário possa visualizá-los.

4.2.2.3 Período de Pausa

Além das condições definidas para que a irrigação se inicie ou seja interrompida, também definimos uma condição que impede o acionamento da irrigação, em período de eventos realizados dentro do gramado, independentemente dos valores de temperatura e umidade coletados.

Para ser possível acionar um período de pausa no sistema automatizado de irrigação, utilizamos informações provenientes do módulo de gerenciamento do sistema. O módulo de gerenciamento do sistema informa a ocorrência de um evento dentro do gramado, sendo possível assim definir quais momentos a irrigação deve estar suspensa.

O mesmo acontece para o acionamento manual do sistema de irrigação, ou seja, quando iniciado um evento a opção de acionamento manual da irrigação se encontrará suspensa até que o evento termine. Este período de pausa impede que ocorram possíveis incidentes com os equipamentos de irrigação.

4.2.3 Módulo de Controle da Iluminação

Este módulo tem como propósito controlar de forma eficaz os componentes de iluminação presentes em um estádio de futebol, mas precisamente a iluminação do gramado. Para tal

propósito foram utilizados 3 (três) componentes conectados a um NodeMCU, a citar: um sensor de luminosidade responsável por determinar o momento ideal para o acionamento do sistema de iluminação e 2 (dois) relés de controle. Sempre que um evento for iniciado, este módulo recebe essa informação a fim de determinar os fatores apropriados de acionamento da iluminação, definidos por meio de um sensor de luminosidade.

Este módulo consiste no acionamento do sistema de iluminação, tanto de forma manual quanto automática. O fato de utilizarmos 2 (dois) relés de controle justifica-se pela possibilidade de acionamento individual de partes especificas do sistema de iluminação. Considere um campo de futebol com uma quantidade X de refletores utilizados nos processos de iluminação. A quantidade de refletores pode ser dividida em duas partes, sendo uma delas acionada pelo primeiro relé e a outra pelo segundo relé.

Embora a iluminação possa ser acionada de forma individual, também é possível acionar manualmente todas as luzes pertencentes ao sistema de iluminação. O acionamento manual mencionado é possível através de um clique em uma das opções de acionamento presentes no módulo de gerenciamento do sistema.

O acionamento automático do sistema de iluminação acontece por meio de informações coletadas pelo sensor de luminosidade LDR utilizado no projeto. Caso um evento seja iniciado, este sensor coleta informações sobre os níveis de luminosidade para então serem devidamente tratados. Estas informações definirão o momento adequado para o acionamento automático da iluminação.

Para ALMEIDA (2010), os valores ideais para o acionamento automático de um sistema de iluminação, em qualquer ambiente, estão compreendidos entre 25 e 35 Lux (lx). O autor trás uma comparação entre os valores coletados por um Luxímetro e um sensor LDR, o que permite uma comparação de valores de luminosidade, obtidos por esses diferentes sensores.

Desta forma, utilizamos neste módulo o LDR que se trata de um sensor de luminosidade capaz de medir os níveis de iluminação do ambiente, através da corrente elétrica transitada entre seu resistor. O valor ideal a ser utilizado como valor padrão no acionamento automático da iluminação foi escolhido por meio de estudos comparativos entre um Luxímetro e o LDR.

O valor de luminosidade, que define o momento ideal para acionamento automático da iluminação, ainda poderá ser modificado, de acordo com as necessidades do usuário final. Caso um evento seja iniciado e o usuário opte em acionar manualmente o sistema de iluminação, ao invés de aguardar o processo automatizado, será então considerado que a definição de acionamento automático é inadequado para o local.

O valor de luminosidade para o acionamento automático do processo de iluminação é então atualizado. Este novo valor será armazenado em um banco de dados para que seja possível acessá-lo posteriormente, uma vez que o microcontrolador utilizado não possui opções de armazenamento de dados em memória.

O módulo de gerenciamento de iluminação também se conectará ao *broker* para que seja possível receber mensagens enviadas pelo módulo de gerenciamento do sistema, e para que seja

possível o envio de informações relacionadas os níveis de luminosidade coletadas.

O módulo de controle da iluminação, assim como os anteriores, tem seu funcionamento baseado no NodeMCU, no qual foram utilizados um sensor LDR para obtenção dos níveis de luminosidade presentes no local e um relé de 2 (dois) canais para o acionamento individual das partes do sistema de iluminação. A Figura 4.2 mostra representação deste módulo em um suposto cenário real.

4.2.4 Módulo de Gerenciamento do Sistema

O módulo de gerenciamento do sistema é responsável por gerenciar as informações trocadas pelos demais módulos, bem como permitir a visualização das informações coletadas entre os diferentes sensores utilizados pelos módulos.

Assim, o usuário é capaz de visualizar as informações do sistema, além de possibilitar os acionamentos manuais do sistema de iluminação e/ou irrigação ou outras funcionalidades, tais como o início de um evento. Veja na Figura 4.3 a representação geral do funcionamento do módulo de gerenciamento do sistema.

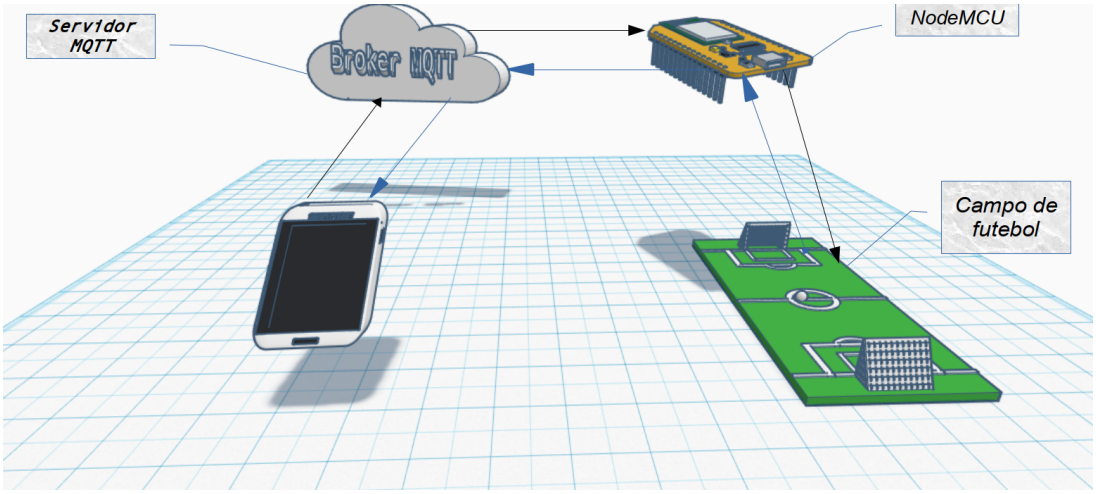

Figura 4.3: Módulo de Gerenciamento do Sistema

Além da possibilidade de acionar cada módulo manualmente, também é possível, através do módulo de gerenciamento do sistema, iniciar um evento. O início de um evento trata-se da utilização do estádio de futebol para a realização de algum evento (cerimônia, comemoração, espetáculo, *show* ou qualquer outra solenidade). Neste caso, os demais módulos do sistema serão notificados da realização da atividade.

O módulo de gerenciamento do sistema foi desenvolvido para dispositivos Android. Um painel de controle foi criado através do MQTT *Dash* para controle das funções necessárias apresentadas pelos módulos do sistema.

4.3 Visão Detalhada do Sistema

Nesta seção é apresentada a prototipagem e programação dos módulos deste sistema. Por se tratar de uma proposta de desenvolvimento estes mesmos não foram dispostos em um cenário de aplicação real. Neste momento, apresentamos como como cada componente físico foi conectado e deve ser instalado, a fim de possibilitar o processo de validação do funcionamento do sistema proposto.

4.3.1 Módulos de Controle da Iluminação e Irrigação - Prototipagem

Para comprovar o funcionamento destes módulos utilizamos uma matriz de contatos, no qual todos os sensores e atuadores foram conectados ao NodeMCU. Tendo por finalidade a economia de recursos de *hardware*, optou-se na utilização de apenas um microcontrolador para desenvolvimento destes módulos. Note na Figura 4.4 a disposição física dos dispositivos relacionados aos módulos de controle de irrigação e de iluminação.

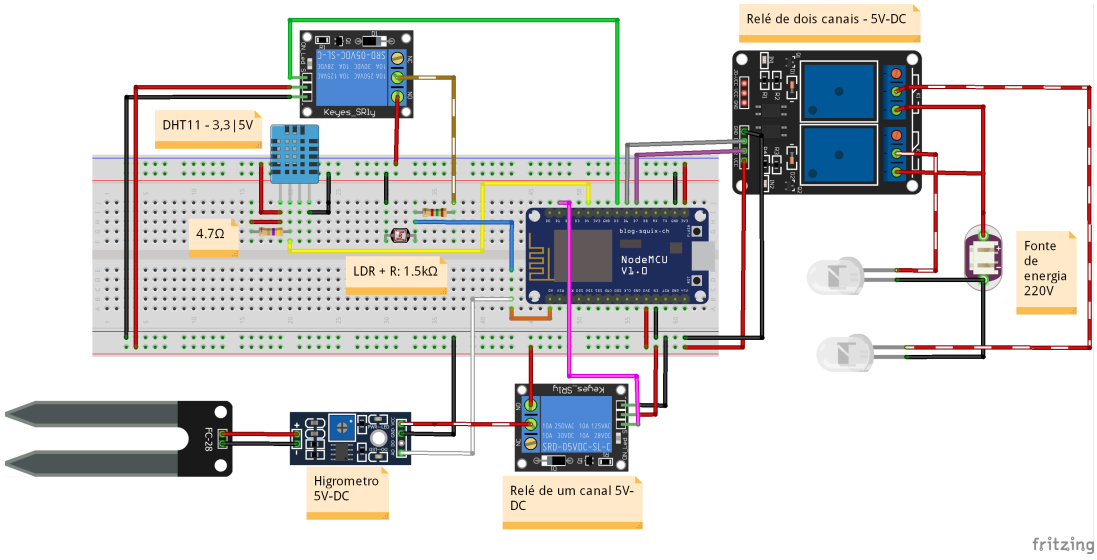

Figura 4.4: Prototipagem do Módulo de Controle de Iluminação e Irrigação.

Para obtenção da temperatura das raízes de um gramado é necessário a utilização de um sensor para tal propósito. Embora o DHT11 não seja indicado para ler valores de temperatura do solo, o mesmo foi utilizado apenas com a finalidade de teste. Em cenários reais devem ser utilizados sensores mais robustos com propósito de operar em profundidades específicas do solo.

Este sensor foi conectado a fonte de alimentação de 3,3 volts e na fonte GND presentes no NodeMCU, seu pino de saída de dados foi ligado ao pino digital D4 presente no microcontrolador. A fim de possibilitar uma leitura mais precisa deste sensor, utilizamos um resistor de *pull-up* de 4.7 ohms para alimentação o pino de dados do DHT11. Este resistor conta com a conversão de valores digitais para analógicos, realizada pelo *chip* controlador presente no DHT11. Desta forma, torna-se possível utilizar a porta D4 do NodeMCU como se fosse uma porta analógica.

O processo de leitura da umidade presente no gramado foi feito através do higrômetro: este sensor foi conectado a tensão de alimentação de 5 volts e ao GND presentes no NodeMCU. O pino de saída de dados foi conectado à porta A0 presente no microcontrolador.

O sensor LDR foi alimentado com uma tensão de 5 volts filtrada por um resistor de 1.5 kohms. Este mesmo pino alimentado com esta tensão filtrada foi utilizado para retornar dados analógicos ao NodeMCU. Deste modo, este pino também foi conectado junto a porta A0 e ao GND do microcontrolador.

O NodeMCU possui apenas uma entrada de dados analógicos. Neste módulo são utilizados 2 (dois) sensores ligados a porta analógica A0 do mesmo microcontrolador, o que poderia causar erros nas leituras caso os sensores funcionassem simultaneamente.

Para corrigir este problema foram utilizados 2 (dois) relés de proteção. Sabendo que estes sensores não funcionam sem a devida alimentação elétrica, um sensor é desligado enquanto o outro opera e vice-versa. Cada relé foi conectado a tensão de 5V do microcontrolador: o primeiro relé tem por objetivo interceptar a alimentação de 3,3V do sensor LDR. Isto foi feito da seguinte forma: um *jumper* de conexão conectado a tensão de 3,3V foi conectado a entrada NC (Normalmente Fechada) do relé e outro fio saindo da entrada COM do relé que foi ligado ao resistor do LDR. Este relé é controlado pelo pino D5, presente no NodeMCU, através de um sinal digital.

O segundo relé tem por função interromper o funcionamento do sensor de umidade da mesma forma que o relé anterior, ou seja, o fio de tensão 5V deste sensor é interceptado pela porta NC deste relé e devolvido para o sensor de umidade pela porta COM. Este relé é controlado pelo pino digital D1 do NodeMCU.

É importante notar que estes sensores não deverão operar simultaneamente: por este motivo o microcontrolador foi programado de tal forma que quando um sensor estiver em operação o outro se encontrará desligado.

O critério para alteração de funcionamento dos sensores foi definido da seguinte forma: no momento em que estiver ocorrendo um evento dentro do gramado, a irrigação automática não deverá ocorrer, logo não será necessário utilizar o sensor de umidade e assim o mesmo é desligado. Caso não esteja ocorrendo um evento dentro do gramado, não será necessário utilizar o sensor de luminosidade LDR. Deste modo, o o sensor de luminosidade é desligado.

Os resistores utilizados no projeto foram calculados por meio da seguinte fórmula:

$$R = \frac{Va - Vc}{I}$$

R: Resistência em ohms.
Va: Tensão em volts da fonte de alimentação ligada ao componente. Neste caso são 3,3V resultantes do microcontrolador.
Vc: Tensão de funcionamento, em volts, do componente.
I: Corrente do componente, em amperes.

Como mencionado anteriormente, o módulo de controle da iluminação poderá ser dividido no acionamento de X partes diferentes da iluminação. Neste módulo escolhemos o acionamento de 2 (duas) partes do sistema de iluminação, sendo cada parte responsável pela iluminação de um lado do gramado (esquerdo e direito). Para a implementação desta funcionalidade utilizamos um relé de 2 (dois) canais: cada um responsável por um lado no sistema de iluminação.

Neste módulo, um dos fios que se conecta a energia (110/220V) até os dispositivos de iluminação do lado direito do gramado é interceptado pelo primeiro canal deste relé. Assim como os relés anteriores, este fio foi interceptado pela porta NA deste canal e devolvida pela porta COM aos dispositivos de iluminação. Este canal é controlado pelo pino digital D6 que esta conectado a entrada IN1 deste relé. Dessa forma, quando este relé recebe um sinal digital, o mesmo permite que a energia transite até os dispositivos de iluminação ligados ao canal.

O segundo canal deste relé tem por objetivo controlar a iluminação do lado esquerdo do gramado. O processo de montagem é semelhante ao do canal anterior, porém utilizando outro pino digital (pino D7) conectado na entrada IN2 correspondente ao segundo canal deste relé. Desta forma, assim que recebido um valor digital este canal permitirá a passagem da energia até os dispositivos de iluminação presentes no lado esquerdo do gramado.

4.3.2 Módulo de controle da bomba d'água - Prototipagem

Assim como os módulos anteriores este módulo também foi montado sobre uma placa de ensaio onde os componentes externos deste mesmo foram ligados ao microcontrolador. Note na figura 4.5 como cada componente foi ligado ao NodeMCU.

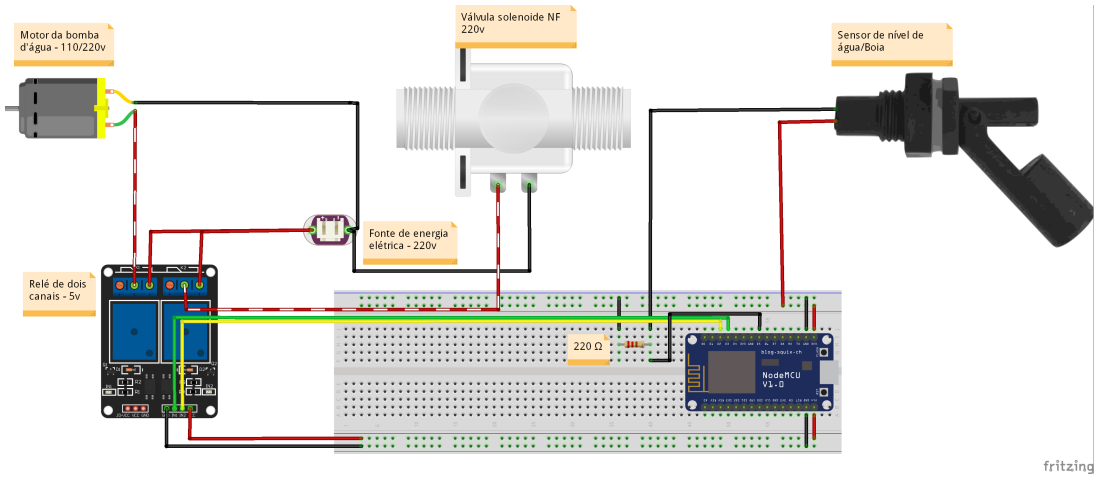

Figura 4.5: Prototipagem do Módulo de Controle da Bomba D'água/Motobomba

O sensor de nível de água/boia tem por objetivo informar em qual nível se encontra a água presente no reservatório utilizado pela motobomba. Este sensor possui 2 (dois) fios, sendo que o primeiro deles foi conectado a alimentação e o segundo ao pino digital D5 do microcontrolador.

Para equilibrar o sinal enviado a esse pino foi utilizado um resistor de 220 ohms ligado entre a tensão GND e o fio conectado ao pino digital. Por se tratar de um sensor de circuito fechado sempre que a haste deste mesmo se fechar a energia ligada a um dos fios é retornada pelo outro fio até a porta digital D5, deste modo é possível saber em qual nível encontra-se a água presente no reservatório.

Embora tenha sido utilizado apenas um sensor boia neste projeto, por questões de economia de recursos, é possível utilizar mais sensores dispostos dentro do reservatório de água. Note na Figura 4.6 a disposição ideal para uma maior precisão da medição do nível de água dos sensores boia.

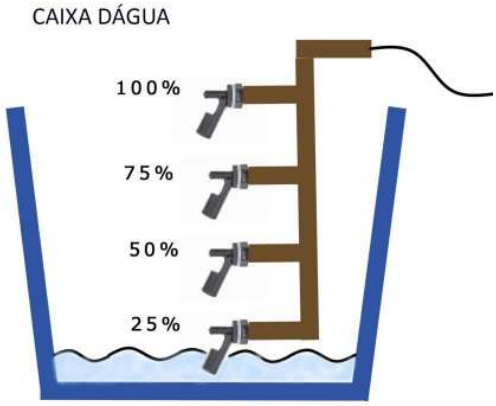

Figura 4.6: Esquema de Montagem Ideal para Sensores Boia - Fonte: blog.eletrogate.com

A válvula solenoide foi conectada a um relé de 2 (dois) canais, sendo utilizado apenas 1 (um) canal para chavear a alimentação elétrica da válvula, ou seja, um dos fios de energia utilizados pela válvula é interceptado pelo segundo canal do relé. Esta mesmo fio foi conectado a uma tensão elétrica de 220 Volts, porém o fio que leva a energia positiva até a mesma é interceptado pela porta NF presente no segundo canal do relé e devolvido pela porta COM do mesmo para a válvula solenoide. Este canal do relé é controlado pelo pino digital D2 presente no microcontrolador encontra-se conectado a entrada IN2 do relé. Portanto, ao receber um sinal digital este canal do relé, permite-se a passagem da tensão elétrica até a válvula solenoide, fazendo com que a mesma se abra liberando o fluxo de água até a motobomba.

A bomba d'água tem por objetivo impulsionar a água até os irrigadores presentes no gramado. Qualquer motobomba acionada por uma corrente elétrica que tenha pode ser utilizada pois é um componente que não faz parte deste módulo diretamente, sendo apenas considerado que a mesma exista juntamente com sistema de irrigação.

A bomba d'água utilizada neste projeto foi conectada em uma tensão de 220V, porém o fio que transita energia positiva até a mesma é interceptado pela porta NF presente no primeiro canal do relé utilizado anteriormente e retornado pela porta COM do mesmo. Este canal é controlado pelo pino digital D3 que se encontra conectado na entrada N1 presente no relé. Deste modo é possível acionar/desligar a motobomba por meio de um sinal digital enviado a este canal.

É notável no esquema de montagem deste módulo representado pela Figura 4.5 que foi utilizado apenas um motor elétrico de 6V, representando a motobomba. Porem em cenário real deve se considerar o uso de uma bomba d'água elétrica.

4.3.3 Conexão com *Broker* (CloudMQTT)

Ao criar uma conta gratuita e uma nova instância no CloudMQTT foram disponibilizadas informações para que seja possível conectar dispositivos a este *broker*. Note na Figura 4.7 os dados de autenticação necessários para conectar os dispositivos deste sistema ao CloudMQTT.

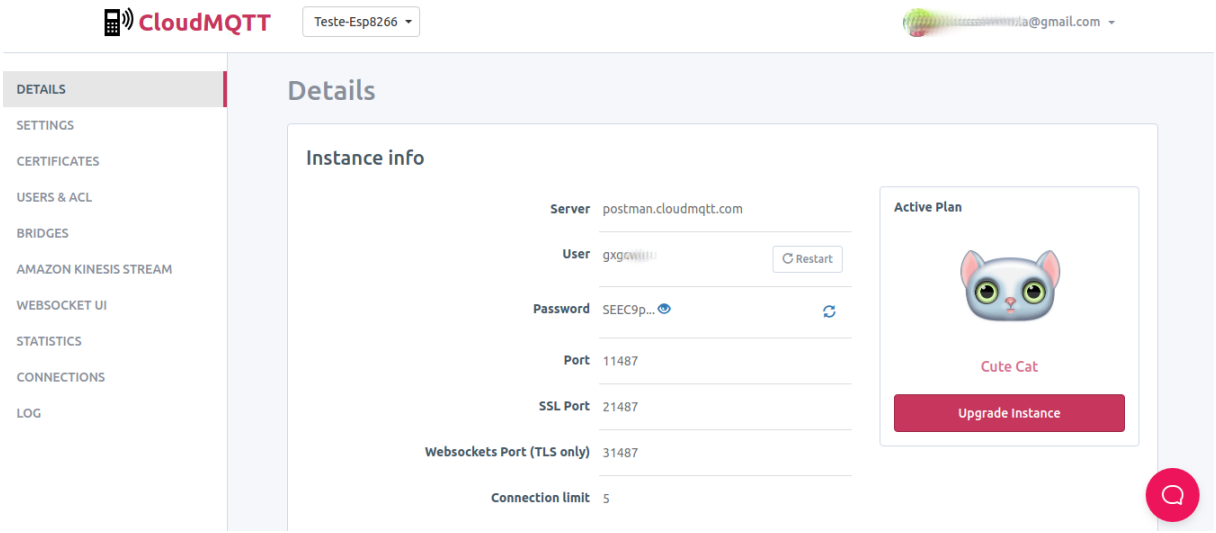

Figura 4.7: Informações Requeridas pelo CloudMQTT - Fonte: api.cloudmqtt.com

Com estes dados obtidos basta utilizá-los na aplicação. O seguinte trecho de código contém a estrutura de conexão utilizada por ambos os microcontroladores. Por motivos de segurança os dados de autenticação foram omitidos.

```
1  #include <ESP8266WiFi.h>
2  #include <PubSubClient.h>
3
4  const char* ssid = "SSID da rede wifi";
5  const char* password = "Senha da rede wifi";
6
7  const char* mqttServer = "XXXXXXXXX";
8  const int mqttPort = XXX;
9  const char* mqttUser = "XXXXX";
10 const char* mqttPassword = "XXXXXXXXXXXXX";
11 //---------------------------------------//
```

4.3.4 Conexão com o Banco de Dados (FireBase)

Ao criar uma conta gratuita no FireBase foi criado um novo banco de dados em tempo real, o *Real Time Database*, em modo de teste. Dentro deste banco de dados foi criada a variável LDR iniciada com o valor padrão utilizado para o acionamento automático do sistema de iluminação.

Nas configurações do banco de dados encontra-se a opção de configuração do projeto. Neste ponto é possível encontrar o nome de usuário na opção: SDK *Admim* do FireBase. Na opção de chaves secretas é disponibilizada a chave secreta do banco de dados criado. Estes dados foram utilizados para conectar-se ao FireBase por meio do NodeMCU. Após o processo de conexão com o banco de dados, inserimos o o valor padrão referente a iluminação automática na variável LDR.

Note na Figura 4.8 o painel que contem o URL do usuário e a chave secreta a ser utilizada.

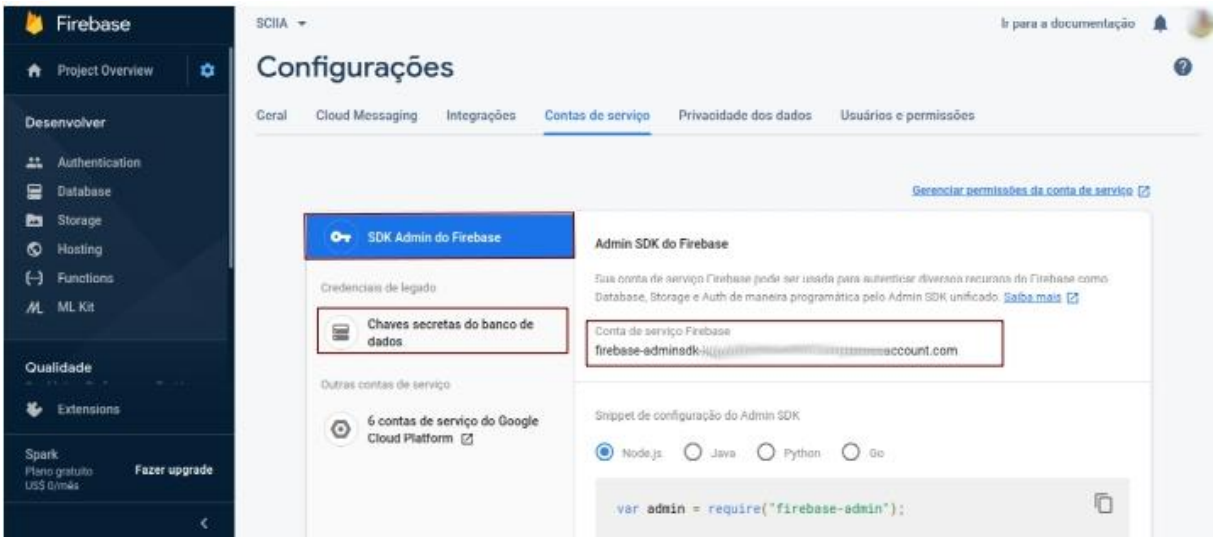

Figura 4.8: FireBase: Painel de configurações - Fonte: console.firebase.google.com

```
1  #include <Firebase.h>
2  #include <FirebaseArduino.h>
3  #include <FirebaseCloudMessaging.h>
4  #include <FirebaseError.h>
5  #include <FirebaseHttpClient.h>
6  #include <FirebaseObject.h>
7  #include <FirebaseArduino.h>
8
9  #define FIREBASE_HOST "XXXXXXXX.firebaseio.com"
10 #define FIREBASE_AUTH "secretkey-xxxxxxxxxxxxx"
```

Note neste código que as linhas 1 a 7 apresentam as bibliotecas necessárias para utilização do FireBase. As linhas 9 e 10 correspondem a autenticação necessária para conexão do

NodeMCU ao FireBase.

4.3.5 Sensores e atuadores - Iluminação e Irrigação

Nesta etapa são abordados os principais trechos de código correspondentes aos sensores e atuadores utilizados no sistema. Os experimentos realizados se encontram na Subseção 4.4, no formato de imagens.

4.3.5.1 Higrômetro

O seguinte trecho de código aborda a maneira como este sensor foi utilizado. Nota-se que nas linhas 6 e 7 que os valores obtidos foram convertidos em porcentagem.

```
1 float porcent = 0.0, porcentSeco = 0.0, temperaturaGrama;
2
3 //Le o valor do pino A0 do sensor
4 valor_analogico = analogRead(portaAnalogica);
5 temperaturaGrama = sensorDht11();
6 porcentSeco = (float)valor_analogico/1024*100;
7 porcent = 100 - porcentSeco;
```

O próximo trecho de código aborda como são tratados os valores de umidade. Note que neste projeto foram assumidos valores abaixo ou equivalentes a 50% parâmetros ideais para acionamento da irrigação, e valores acima de 80% pontos ideais para o desligamento da irrigação.

Nesta pesquisa não foram encontrados valores fixos de umidade para serem adotados, portanto foram realizados experimentos com o sensor de umidade higrômetro inserido em um copo de 200 mililitros contendo areia seca. Neste caso o sensor apresentou valores entre 20 e 40% de umidade.

Quando molhada a areia deste copo foram obtidos valores entre 80 e 100%. Ao observar os níveis de umidade, em intervalos distintos, foi compreendido que a areia apresentava uma certa taxa de umidade em valores um pouco maiores que 50% medidos pelo sensor.

```
1   if (porcent <= 50.0 && temperaturaGrama >= 24.0 && bombaAcionada !=
        1){
2   Serial.print("Porcentagem de umidade: ");
3 Serial.print(porcent);
4 Serial.println("Temperatura da Grama: ");
5 Serial.print(temperaturaGrama);
6 client.publish("IRG","ON");
7 delay(100);
8 if(respBomba == 1){
9 bombaAcionada = 1;
```

```
umidadeInterval = 5000;
}else{
if(respBomba == 0){
umidadeInterval = 50000;
bombaAcionada =0;
}else if(respBomba == -1) umidadeInterval = 1000;
}
}
Serial.print("Estado da Bomba :");
Serial.print(bombaAcionada);
if(porcent >= 80.0 && bombaAcionada == 1 || temperaturaGrama < 24.0){
Serial.println("Temperatura da Grama 2: ");
Serial.print(temperaturaGrama);
Serial.println("umidade 2: ");
Serial.print(porcent);
client.publish("IRG","OFF");
client.publish("AVS","Irrigacao desligada");
bombaAcionada = 0;
respBomba = -1;
umidadeInterval = 5000;
}
```

A temperatura da grama é avaliada por meio do sensor DHT11 que embora não tenha sido instalado dentro do solo da grama serviu de base para os testes realizados. Observe o trecho de código seguinte correspondente a leitura realizada por este sensor.

```
        float temperatura = dht.readTemperature();
        \*
        ........
        *\
        return temperatura;
```

Este sensor necessita de algumas bibliotecas e configurações para funcionar desta forma. Note no Apêndice o código completo responsável pela leitura do sensor de temperatura. Para o acionamento automático da iluminação do gramado, durante a ocorrência de um evento, são avaliados valores de luminosidade do ambiente por meio do sensor LDR. No decorrer desta pesquisa abordamos que os valores de luminosidade utilizados no acionamento automático seriam obtidos através da comparação de dados retornados pelo LDR e um Luxímetro, visando obter o valor de 35 Lux(lx), que por sua vez é o valor ideal considerado na pesquisa.

Pela ausência do medidor de luminosidade Luxímetro, o valor de luminosidade foi obtido através de experimentos, em um cenário real, com a finalidade de se obter um valor ideal de luminosidade para ser adotado pelo sistema. Este experimento foi realizado em um pequeno campo de futebol, localizado próximo ao Distrito do Bezerra, Município de Formosa-GO,

juntamente com o proprietário do local. Este módulo foi acionado ao fim do período vespertino, momento em que os valores de luminosidade do ambiente diminuem.

De acordo com as preferências setadas pelo proprietário, o valor de 350 microampe-res(mA) foi utilizado como parâmetro para acionamento do sistema de iluminação. Reforça-se que, para cenários reais de aplicação, deve ser utilizado um Luxímetro para encontrar o valor de 35 lx, comparado ao LDR.

```
1          float valorLume;
2         int ADC;
3         char MsgLumeMQTT[12];
4
5         ADC = analogRead (portaAnalogica);
6         Lux = (float)ADC;
7         sprintf(MsgLumeMQTT, "%f",Lux);
8         if(event == 1){
9         client.publish("LUME",MsgLumeMQTT);
10        }
11        if(Firebase.getFloat("LDR") != 350.0){
12        valorLume = Firebase.getFloat("LDR");
13        }else{
14        valorLume = 350.0;
15        }
16        if(Lux <= valorLume && event == 1){
17        controle_luz(3);
18        }else if(Lux > valorLume && luzON == 1 && event == 1){
19        controle_luz(-3);
20        luzON = 0;
21        }
22
23        return Lux;
```

Note que a obtenção do valor de 35 lx serve somente para definir um valor padrão para o acionamento automático da iluminação, porém existe a possibilita de alteração deste valor. Observe nas linhas 11 e 12 do código anterior que é realizada uma verificação no banco de dados se o valor padrão de acionamento da iluminação foi alterado. Se o mesmo for diferente do valor padrão, logo este novo valor é assumido como novo valor de referência.

4.3.5.2 Relés de Alternação

O seguinte código apresenta a logica utilizada para alternar os sensores que compartilham a porta analógica A0 do NodeMCU. Note que o sensor de umidade é acionado sempre que não houver um evento no gramado, pois o mesmo tem utilidade somente neste caso. O sensor LDR é

ligado apenas quando um evento é iniciado, pois o mesmo é necessário para medir os valores de luminosidade.

```
1         void alternador(){
2         int ativo1, ativo2;
3         if(event != 1 && ativo1 != 1){// Alterna para o sensor de
              umidade
4         digitalWrite(alter_1, LOW);
5         ativo1 = 1;
6         ativo2 = 0;
7         digitalWrite(alter_2, HIGH);
8         }else if(event == 1 && ativo2 != 1){//alterna para o sensor
              de luminosidade
9         digitalWrite(alter_2, LOW);
10        ativo1 = 0;
11        digitalWrite(alter_1, HIGH);
12        ativo2 = 1;
13        }
14        }
```

4.3.5.3 Relés de Acionamento da Iluminação

Estes componentes são acionados pela função presente no código a seguir. O objetivo é permitir o acionamento/desligamento dos dispositivos de iluminação do gramado, onde são utilizados como parâmetros valores de 1 a 3 para tipos diferentes de acionamento e valores de -1 a -3 para distintas formas de desligamento do sistema de iluminação.

```
1  void controle_luz(int state){
2  if(state == 1){
3  digitalWrite(lume_Rigth, LOW);
4  client.publish("LUZ","iluminacao 01: ON");
5  /*
6  .....
7  */
8   if(state == -1 && event != 1){
9  digitalWrite(lume_Rigth, HIGH);
10 client.publish("LUZ","iluminacao 01: OFF");
11 }
12 /*
13 .....
14 */
15 }
```

4.3.6 Sensores e Atuadores - Módulo da Bomba D'Água

Nesta parte são abordados os principais códigos de funcionamento utilizados pelos seguintes componentes:

4.3.6.1 Sensor de Nível Boia

Este sensor tem por objetivo informar, por meio de um sinal digital, se a água contida no reservatório é suficiente para o acionamento da motobomba. Tal ação é tratada pelo seguinte código:

```
if(!digitalRead(boia)){
client.publish("IRG","OK");
interruptor(1);
}else{
client.publish("IRG","WT");
tempoEspera = 5000;
}
```

Note que quando este sensor esta em estado fechado, o mesmo retorna o valor zero. Ao aplicar o operador de inversão foi possível identificar se o reservatório possui água suficiente para o acionamento da bomba d'água, indicando que a mesma pode ser acionada e executada a função de acionamento. Caso a boia indique que o nível de água encontra-se abaixo é publicada a informação no *broker*. Esta ação resulta em um tempo de espera para que o reservatório seja suficientemente completado.

4.3.6.2 Rele de Dois Canais no Acionamento da Válvula Solenoide e da Bomba D'Água/Motobomba

Para o acionamento destes componentes elétricos foram utilizados relés programados da seguinte forma:

```
void interruptor(int state){
if(state == 1){
digitalWrite(solenoide, LOW);
delay(1000);
digitalWrite(bomba, LOW);
}
if(state == -1){
digitalWrite(bomba, HIGH);
delay(100);
digitalWrite(solenoide, HIGH);
}
}
```

Note que para o parâmetro de acionamento (1) primeiramente é acionado a válvula solenoide e um segundo depois é acionada a bomba d'água, o que garante que a mesma não opere sem a devida alimentação de água. Para o parâmetro de desligamento (-1), a lógica anterior foi invertida para garantir o mesmo critério de funcionamento da bomba d'água. Note na Subseção 4.4 imagens reais de instalação destes componentes.

4.3.7 Mapeamento de Portas do NodeMCU

Na programação dos microcontroladores NodeMCU, por meio da linguagem C++, as portas especificadas são diferentes das identificadas no NodeMCU, isso se dá pois a IDE utilizada enxerga este microcontrolador como se fosse um Arduíno. Por está sendo utilizada a IDE do Arduíno é necessário compreender como as portas serão identificadas nos códigos desenvolvidos. A Tabela 4.1 apresenta o mapeamento das portas utilizadas na programação do NodeMCU na IDE do Arduíno.

NodeMCU-ESP8266	Programação - Arduíno IDE	Tipo	Funcionamento
D0	16	Digital	INPUT/OUTPUT
D1	5	Digital	INPUT/OUTPUT
D2	4	Digital	INPUT/OUTPUT
D3	0	Digital	INPUT/OUTPUT
D4	2	Digital	INPUT/OUTPUT
D5	14	Digital	INPUT/OUTPUT
D6	12	Digital	INPUT/OUTPUT
D7	13	Digital	INPUT/OUTPUT
D8	15	Digital	INPUT/OUTPUT
D9	3	Digital	INPUT/OUTPUT
D10	1	Digital	INPUT/OUTPUT
A0	A0	Analógico	INPUT/OUTPUT

Tabela 4.1: Mapeamento das portas do NodeMCU em relação a programação

4.3.8 Módulo de Gerenciamento do Sistema - Aplicativo móvel

Por meio do aplicativo MQTT *Dash*, gerou-se um painel de controle, o que tornou possível gerenciar todo o sistema. Note na Figura 4.9 e na Figura 4.10 a disposição utilizada para representação das informações do sistema automatizado de controle de irrigação e iluminação aplicado a estádios de futebol.

4.3.8.1 Painel de Controle - MQTT *Dash*

Os botões presentes no módulo de gerenciamento do sistema possuem as seguintes funcionalidades:

- **Iluminação 01**: liga/desliga a iluminação do lado direito do estádio.
- **Iluminação 02**: liga/desliga a iluminação do lado esquerdo do estádio.
- **Iluminação TOTAL**: liga/desliga todo sistema de iluminação.
- **Valor-iluminação**: valor de luminosidade retornado pelo sensor LDR.
- **Evento**: botão utilizado para sinalizar a ocorrência de um evento.
- **Temperatura-Grama**: temperatura da grama obtida pelo sensor DHT11
- **Umidade-Grama**: umidade da grama retornada pelo sensor higrômetro.
- **Irrigação**: liga/desliga, manualmente, o sistema de irrigação, além de indicar o *status* de funcionamento.
- **Avisos**: avisos diversos enviados pelos módulos que compõem o sistema proposto.

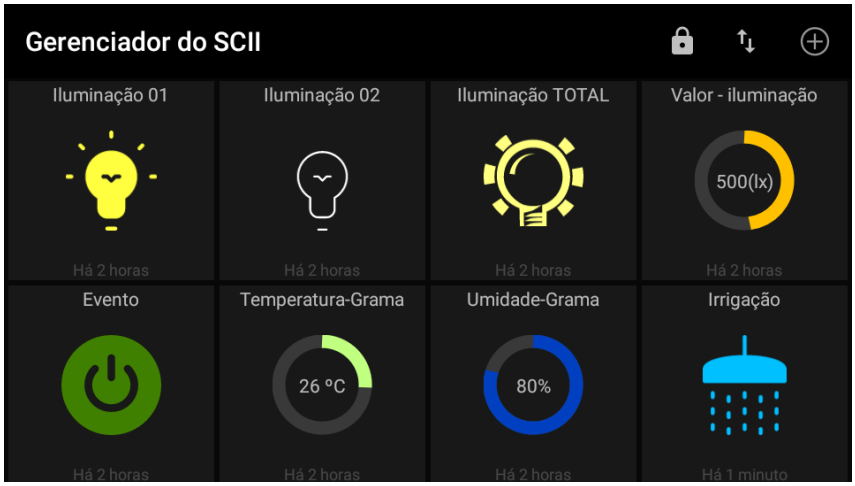

Figura 4.9: Gerenciamento do Sistema - MQTT *Dash* (Parte 1)

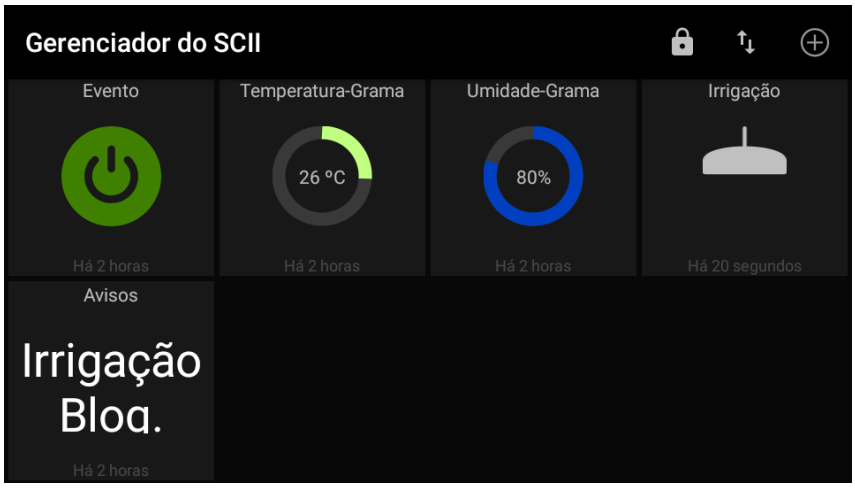

Figura 4.10: Módulo de Gerenciamento do Sistema - MQTT *Dash* (Parte 2)

4.4 Experimentos Realizados

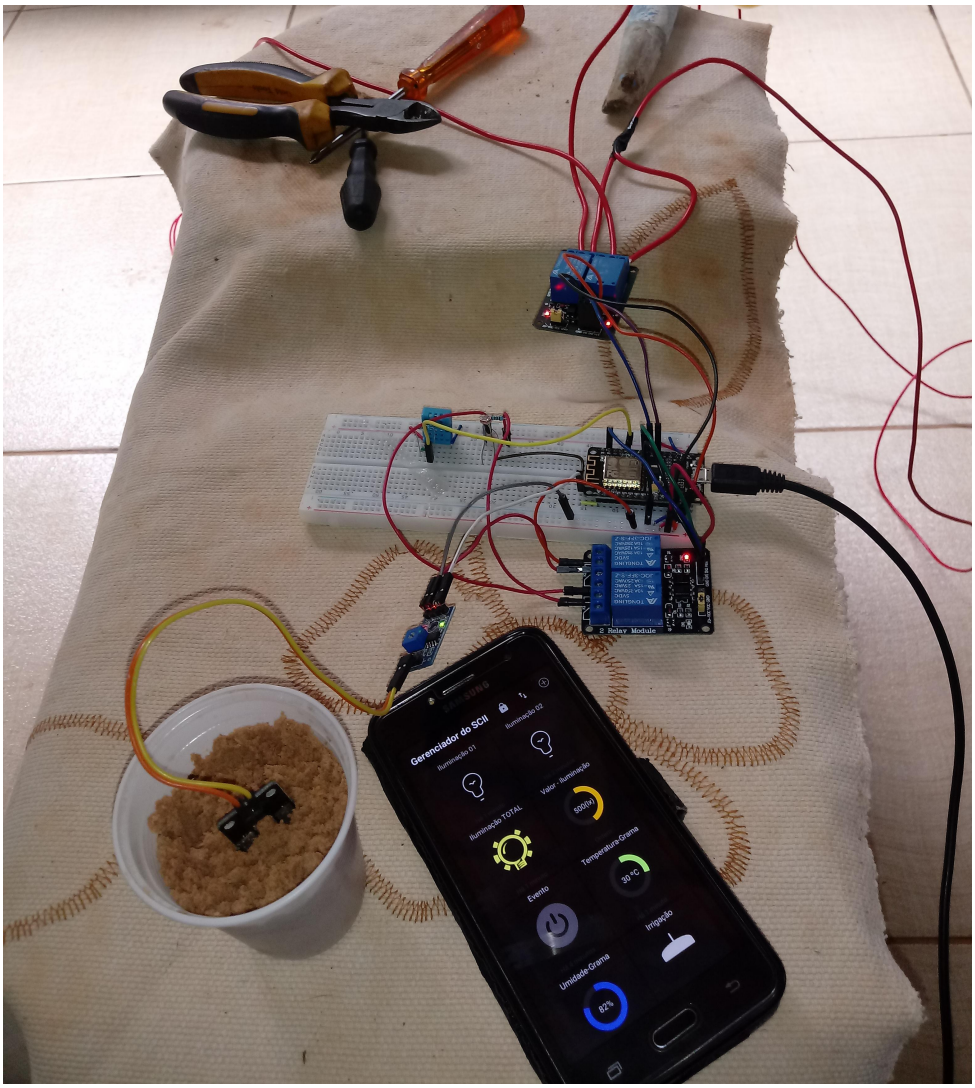

Figura 4.11: Módulos de Controle de Iluminação e Irrigação - (Parte 1)

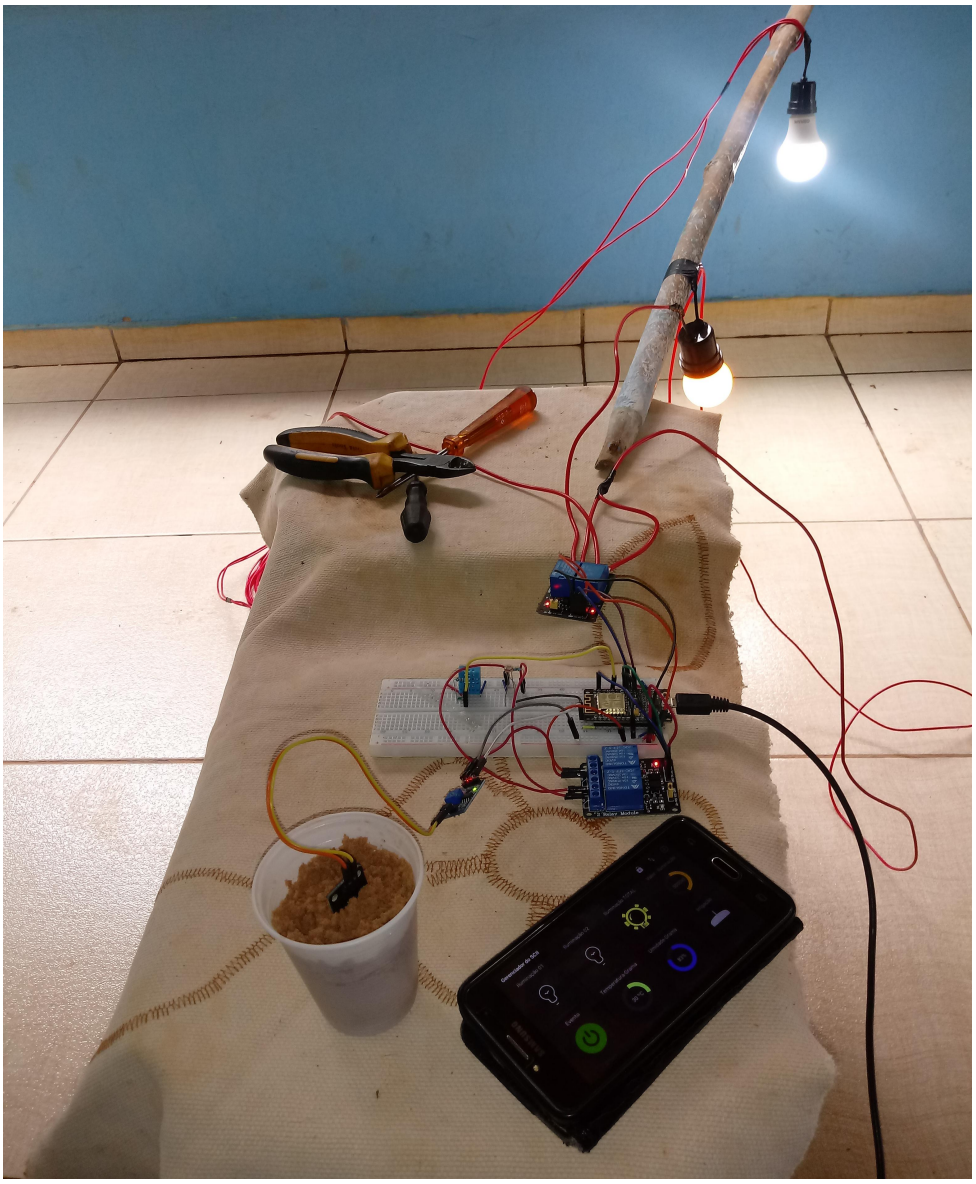

Figura 4.12: Módulos de Controle de Iluminação e Irrigação - (Parte 2)

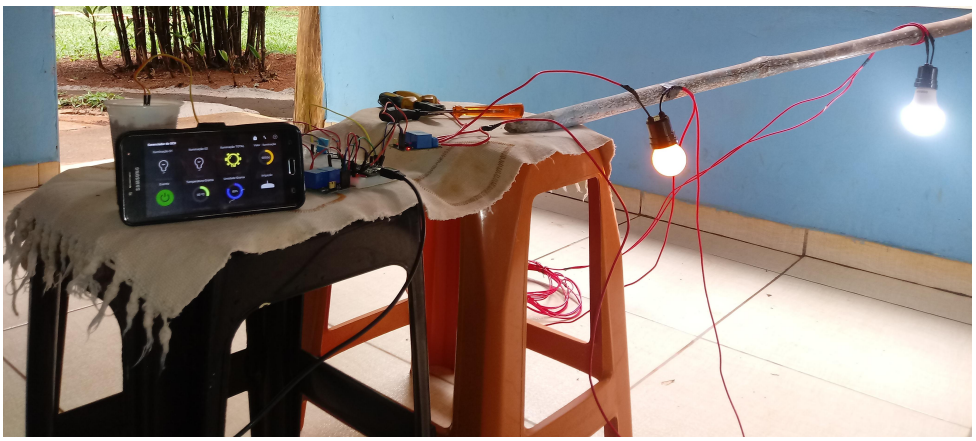

Figura 4.13: Módulos de Controle de Iluminação e Irrigação - (Parte 3)

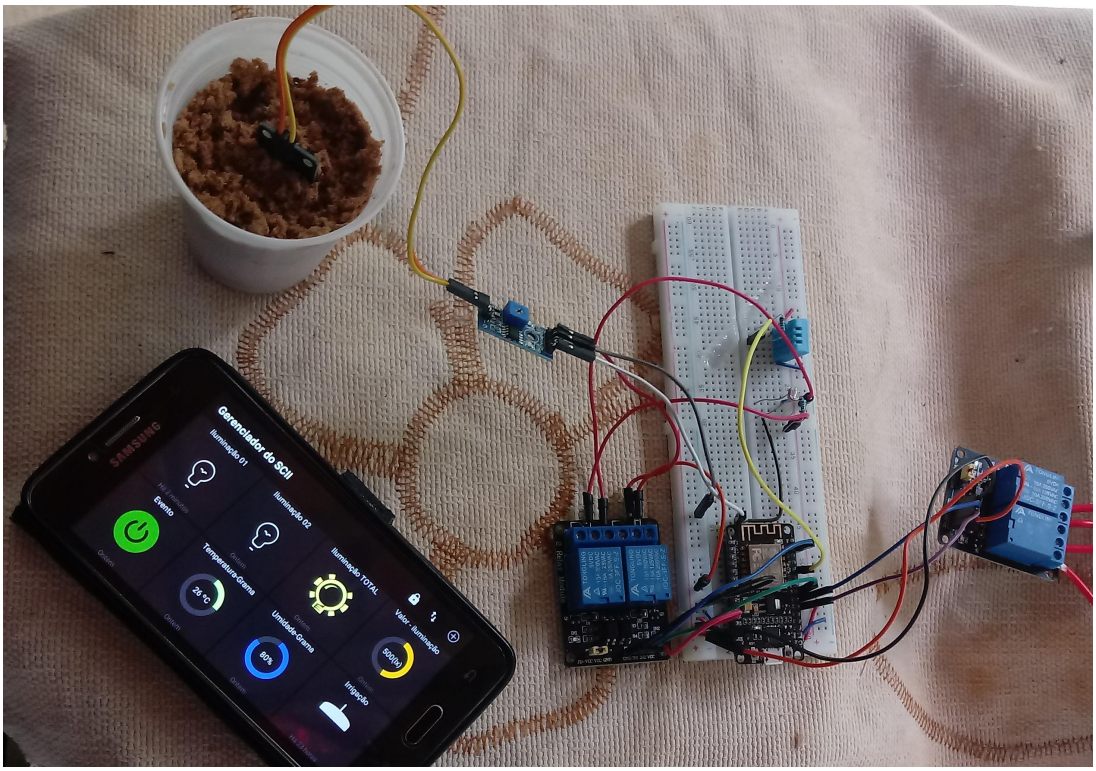

Figura 4.14: Módulos de Controle de Iluminação e Irrigação - (Parte 4)

Figura 4.15: Módulo de Controle da Bomba d'Água/Motobomba - (Parte 1)

Figura 4.16: Módulo de Controle da Bomba d'Água/Motobomba - (Parte 2)

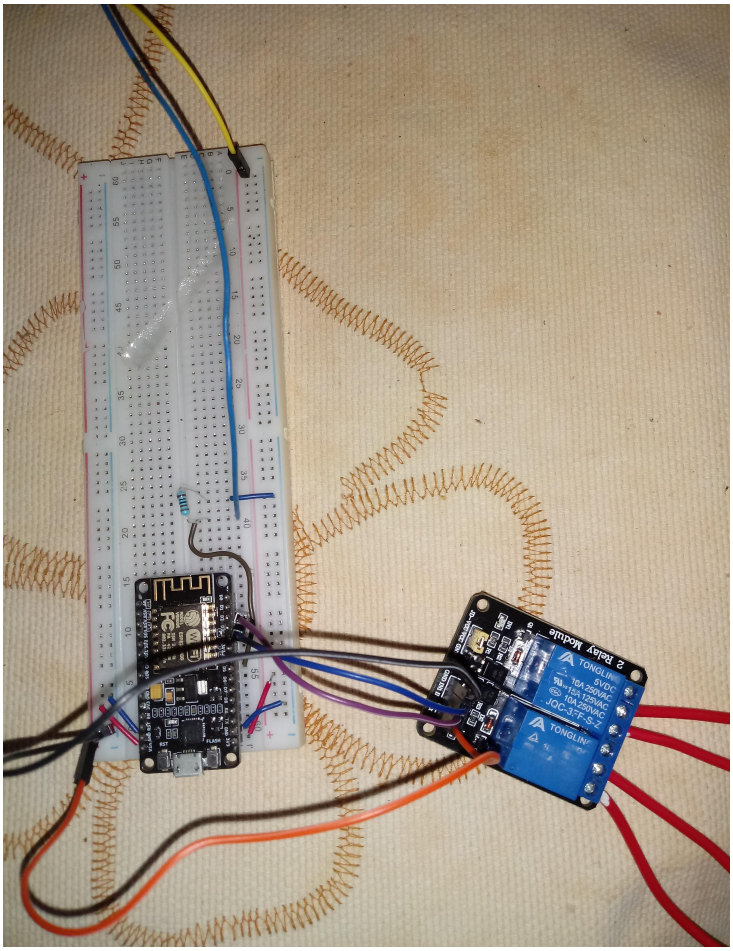

Figura 4.17: Módulo de Controle da Bomba d'Água/Motobomba - (Parte 3)

4.5 Considerações finais

Neste Capítulo foram apresentados os resultados obtidos com a construção de um sistema de controle de irrigação e iluminação aplicado ao cenário de estádios de futebol. Primeiramente, apresentamos uma visão geral do sistema, com as particularidades e detalhes de funcionamento de cada um dos módulos que compõe o sistema proposto. Apresentamos também as prototipagens dos principais módulos, enfatizando a forma de ligação dos dispositivos físicos utilizados e os códigos desenvolvidos. Por fim, algumas fotos dos experimentos são apresentadas, apenas com o intuito de apresentar o sistema em um possível modo de produção.

5

Conclusão

A popularização dos dispositivos e os avanços da eletrônica permitiram que diversas tecnologias fossem desenvolvidas. Além disso, a popularização do preço do *hardware* utilizado em IoT, tais como os sistemas embarcados Arduíno, Raspberry pi e os módulos baseados no microcontrolador ESP8266 representam um grande avanço para conexão, comunicação e consolidação de objetos inteligentes.

O objetivo deste trabalho foi propor o desenvolvimento de um sistema capaz de controlar de forma eficaz e automatizada os processos de iluminação e irrigação em estádios de futebol. Para atingir este objetivo foram utilizados dispositivos de *hardware* e *software* livres que por sua vez são fáceis de se utilizar e tem um baixo custo de aquisição.

O sistema desenvolvido, baseado no microcontrolador ESP8266 e diversos sensores/atuadores, mostrou-se eficaz em ambientes de testes, no qual constatamos que o mesmo é capaz de controlar de forma automatizada os sistemas de iluminação e irrigação presentes em um estádio de futebol. Embora não tenha sido instalado no ambiente proposto, os testes realizados apontam um comportamento similar caso seja implantado no objeto de estudo.

O módulo de gerenciamento do sistema permite, por meio de um painel de controle desenvolvido no aplicativo MQTT *Dash*, o controle manual dos processos de iluminação e/ou irrigação, bem como a visualização de informações importantes, tais como o *status* dos dispositivos de irrigação/iluminação e os valores de temperatura, luminosidade e umidade captados pelos diferentes sensores utilizados no projeto.

Os resultados obtidos neste trabalho foram satisfatórios e este sistema se mostrou útil e eficaz, podendo ser implementado em ambientes reais, tais como estádios de futebol, *societys* e clubes esportivos.

5.0.1 Dificuldades Encontradas e Sugestões para Trabalhos Futuros

Os microcontroladores NodeMCU possuem a limitação de possuir apenas uma porta analógica. Um sistema mais robusto, com uma maior quantidade de sensores/atuadores, necessitaria de um microcontrolador dotado com maior quantidade de portas analógicas.

Outro fato importante é o da Evapotranspiração (ET) que, como visto anteriormente, é uma taxa que mede a quantidade de água ideal a ser utilizada por uma planta em determinadas

condições climáticas. Para calcular esta taxa são necessários centros meteorológicos dispostos próximos ao local a ser irrigado. Deste modo, sugere-se a construção de um sistema que funcione como um centro de meteorologia para estimar a quantidade de água necessária para a lavoura de grama.

Uma outra versão do sistema de controle de irrigação e iluminação proposto neste TCC poderia incluir sensores de fluxo de água para medição da quantidade de água aplicada no gramado.

Este sistema se limita na segurança das informações trafegadas em rede por meio do protocolo MQTT, seria interessante implementar um padrão de segurança no processo de troca de informações, evitando desta forma possíveis incidentes com o uso indevido destas informações por terceiros.

Na programação destes módulos é percebível que, após uma interrupção nos componentes, seja por falta de energia elétrica ou reinicialização dos mesmos resultaria na perda das informações recebidas, como exemplo, a sinalização de ocorrência de um evento ou o acionamento total do sistema de iluminação. Quando reiniciados, estes microcontroladores perdem estas informações podendo por consequência executar ações indesejadas pelo justo fato de não conhecer os critérios estabelecidos para aquela determinada ocasião. Para resolver este problema, existe a possibilidade de armazenar todos os dados sensíveis dos módulos no banco de dados FireBase. Dessa forma, os dados não serão perdidos.

Para trabalhos de mesmo propósito e com maior robustez, sugere-se também a necessidade de se desenvolver um aplicativo próprio para a visualização dos dados do sistema, superando assim as limitações encontradas nos aplicativos utilizados para controlar aplicações que se comunicam por meio do protocolo MQTT.

Referências

ALMEIDA, G. C. d. Sistema controlador de iluminação de ambientes através de interface computadorizada. , [S.l.], 2010.

AMORIM, L. G. d. P. **Utilização de sistemas dedicados a protocolos de rede aplicados à eficiência energética da iluminação pública**. 2011. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Norte.

ARAUJO ELIAS, A. A. de et al. ArdWeather: uma estação meteorológica baseada no arduino e em web services restful. In: SAFETY, HEALTH AND ENVIRONMENT WORLD CONGRESS. **Proceedings...** [S.l.: s.n.], 2014. v.14, p.44–48.

BARROS, E.; CAVALCANTE, S. Introdução aos sistemas embarcados. **Artigo apresentado na Universidade Federal de Pernambuco-UFPE**, [S.l.], p.36, 2010.

BIRD, R. **IRRIGAÇÃO PROFISSIONAL NOS GRAMADOS DE CAMPOS FUTEBOL**. [S.l.]: Rain Bird, 2013.

BORBA, B. d. **Serviço de descoberta para o protocolo MQTT em um sistema de monitoramento de grupos motor-gerador baseado em internet das coisas**. 2018. Tese (Doutorado em Ciência da Computação) — .

CAETANO, M. R.; SOUZA GONÇALVES, E. de. SISTEMA DE INTERLIGAÇÃO E MONITORAMENTO DOS PONTOS DE ILUMINAÇÃO PÚBLICA. **Revista Científica Doctum Multidisciplinar**, [S.l.], v.1, n.2, 2019.

CAVALCANTI, L. V. d. S. et al. Automação residencial com NodeMCU. , [S.l.], 2018.

CONCEIÇÃO, W. N. E. da; RESENDE COSTA, R. M. de. Análise do Protocolo MQTT para Comunicação IoT através de um Cenário de Comunicação. **Caderno de Estudos em Sistemas de Informação**, [S.l.], v.5, n.2, 2019.

CRUZ, A. A.; LISBOA, E. F. Webhome–automação residencial utilizando raspberry pi. **Revista Ciência e Tecnologia**, [S.l.], v.17, n.31, 2014.

CUNHA, A. F. O que são sistemas embarcados. **Saber Eletrônica**, [S.l.], v.43, n.414, p.1–6, 2007.

CUNHA, K. C. B. da; ROCHA, R. V. da. Automação no processo de irrigação na agricultura familiar com plataforma Arduíno. **Revista Eletrônica Competências Digitais para Agricultura Familiar**, [S.l.], v.1, n.2, p.62–74, 2016.

DA SILVA, R. T. G. et al. **Sistema de automação residencial de baixo custo utilizando o esp8266**. [S.l.]: Ceará, 2017.

DIAS, D. M. d. R. G. **AVALIAÇÃO DA ILUMINAÇÃO ARTIFICAL DO ESTÁDIO LUSO BRASILEIRO**. 2018. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio de Janeiro.

ELMASRI, R. et al. Sistemas de banco de dados. , [S.l.], 2005.

FARIAS FILHO, R. d. M. **Um gerador de sistemas embarcados a partir de modelo independente de plataforma baseado no perfil MARTE**. 2013. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.

FERNANDES, J. D. **Desenvolvimento de sistemas embarcados para redes de sensores e atuadores sem fio aplicadas em unidades de elevação de petróleo do tipo plunger-lift**. 2010. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Norte.

FERRASA, M.; BIAGGIONI, M. A. M.; DIAS, A. H. Sistema de monitoramento da temperatura e umidade em silos graneleiros via radiofrequencia. **Energia na Agricultura**, [S.l.], p.139–156, 2010.

GOLOMBEK, C. H. Estresse em gramados e fatores fisiológicos correlatos. **SIMPÓSIO SOBRE GRAMADOS**, [S.l.], v.3, 2006.

GOMES, A. **Projeto de uma rede sensor sem fio para monitoramento ambiental utilizando protocolo Zigbee**. 2016. B.S. thesis — Universidade Tecnológica Federal do Paraná.

GONCALVES, D. et al. Sistema IoT para monitoramento e controle de irrigação. In: IV ESCOLA REGIONAL DE INFORMÁTICA DO PIAUÍ. **Anais...** [S.l.: s.n.], 2018. p.310–315.

GOUVEA, M. M. R. de. ESTUDO DE CONFIABILIDADE EM BOMBAS CENTRÍFUGAS. , [S.l.], 2008.

GRAH, V. d. F.; BOTREL, T. A.; DE MATOS, I. Solução alternativa para bombeamento de água e automação da irrigação sem o uso de energia elétrica. **Irriga, Botucatu**, [S.l.], p.309–323, 2012.

GURGEL, R. G. A. Principais espécies e variedades de grama. **Simpósio Sobre Gramados**, [S.l.], v.1, 2003.

JÚNIOR, E. d. S. S.; PINTO, S. C. C.; BRAZ, R. M. M. A Computação Embarcada, a Plataforma Arduíno e a Internet das Coisas como Tecnologia Assistiva na construção de Mapas Táteis para os Alunos com Deficiência Visual no Processo de Ensino e Aprendizagem. In: WORKSHOPS DO CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO. **Anais...** [S.l.: s.n.], 2018. v.7, n.1, p.53.

LAMPERT, L. Ferramenta de apoio ao desenvolvimento de software embarcado de acordo com a norma IEC-61508 Ed 2.0. , [S.l.], 2012.

LORENZON, A. F. Avaliação do desempenho e consumo energético de diferentes interfaces de programação paralela em sistemas embarcados e de propósito geral. , [S.l.], 2014.

MATSURA, D. U. et al. SISTEMA DE CONTROLE DE ABASTECIMENTO DE ÁGUA E ENERGIA NO CULTIVO HIDROPÔNICO NFT WATER SUPPLY CONTROL SYSTEM AND ENERGY GROWING HYDROPONIC NFT. , [S.l.], 2015.

MEDEIROS, I. et al. Uso de Sistemas Automatizados para Otimizar a captação de Energia em Painéis Solares. **Blucher Mathematical Proceedings**, [S.l.], v.1, n.1, p.12–19, 2015.

MEZZARI, L. T. Internet das Coisas: arduino, firebase, e android. **Revista Eletrônica em Gestão e Tecnologia**, [S.l.], v.5, n.1, p.93–97, 2019.

NETO, J. G. SISTEMAS DE IRRIGAÇÃO PARA GRAMADOS. , [S.l.], 2003.

OLIVEIRA, B. A. S.; ASSIS, S.; NOLLI, C. DEVELOPMENT OF A PROTOTYPE ELECTRICAL ENERGY MONITORING SYSTEM VIA INTERNET/DESENVOLVIMENTO DE UM PROTÓTIPO DE SISTEMA DE MONITORAMENTO DE ENERGIA ELÉTRICA VIA INTERNET. **Revista de Engenharia da Universidade Católica de Petrópolis**, [S.l.], v.12, n.1, p.48–61, 2019.

OLIVEIRA, D. et al. An integrated development environment for the NS-2 Network Simulator. **Scientia Plena**, [S.l.], v.8, n.3 (a), 2012.

PIGATTO, D. F. **Segurança em sistemas embarcados críticos-utilização de criptografia para comunicação segura**. 2012. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.

PITON, O. H. G. AUTOMAÇÃO RESIDENCIAL UTILIZANDO A PLATAFORMA EM NUVEM IBM BLUEMIX. , [S.l.], 2017.

POUSO, M. T. P. Sistema de automação e controle de um sistema de irrigação. , [S.l.], 2012.

QUINTANILHA, I. M.; ESTEVÃO FILHO, R. Sensores de nível. **Rio de Janeiro: UFRJ**, [S.l.], 2013.

ROCHA, F. B. et al. Plataforma de comunicação sem fio aplicada a sistemas de irrigação. **Holos**, [S.l.], v.5, p.269–282, 2014.

ROTTA, G.; CHARÃO, A.; DANTAS, M. Um estudo sobre protocolos de comunicação para ambientes de internet das coisas. In: XVII ESCOLA REGIONAL DE ALTO DESEMPENHO DO ESTADO DO RIO GRANDE DO SUL. **Anais...** [S.l.: s.n.], 2017.

SAMPAIO, R. P. Sistema de controle de atitude embarcado para vôo autônomo de aviões em escala. , [S.l.], 2017.

SANTEE, A. Programação de Jogos com C++ e DirectX. **São Paulo: Novatec**, [S.l.], 2005.

SANTOS, B. P. et al. Internet das coisas: da teoria à prática. **Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos**, [S.l.], 2016.

SILVA, B. et al. Segurança de software em sistemas embarcados: ataques & defesas. **Minicursos do XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais–SBSeg**, [S.l.], v.2013, p.101–155, 2013.

SILVA, W. C. S. d. Aplicações móveis nativas com react native e firebase: um estudo de caso. , [S.l.], 2018.

SOEIRO, R. R. et al. SISTEMA DE CONTROLE DE HORÁRIOS DE AULA UTILIZANDO LEITORES BIOMÉTRICOS. , [S.l.], 2014.

SOUZA SILVA, A. L. de; SILVA, R. C. M. F. Protocolo http x protocolo https. **Nucleus**, [S.l.], v.6, n.1, p.1–8, 2009.

TORRES, A. B.; ROCHA, A. R.; DE SOUZA, J. N. Análise de desempenho de brokers mqtt em sistema de baixo custo. In: XXXVI CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇA O. **Anais...** [S.l.: s.n.], 2016. p.2804–2815.

VELOSO, A. et al. Implementação de um controle em Real-Time para Sistemas Hidroponicos NFT e aquaponia utilizando Broker MQTT e tecnologias de IoT. In: IV ESCOLA REGIONAL DE INFORMÁTICA DO PIAUÍ. **Anais...** [S.l.: s.n.], 2018. p.32–37.

WEISS, M. C.; BERNARDES, R. C.; CONSONI, F. L. Cidades inteligentes como nova prática para o gerenciamento dos serviços e infraestruturas urbanos: a experiência da cidade de porto alegre. **Revista Brasileira de Gestão Urbana**, [S.l.], v.7, n.3, p.310–324, 2015.

Apêndice

A

Apêndice A - Código fonte

A.1 Código dos módulos de controle da iluminação e irrigação

```
#include <stdlib.h>
//Bibliotecas do sensor DHT11
#include <Adafruit_Sensor.h>
#include <DHTesp.h>
#include <DHT.h>
#include <DHT_U.h>
//Bibliotecas do Firebase
#include <Firebase.h>
#include <FirebaseArduino.h>
#include <FirebaseCloudMessaging.h>
#include <FirebaseError.h>
#include <FirebaseHttpClient.h>
#include <FirebaseObject.h>
#include <FirebaseArduino.h>
//Bibliotecas do ESP8266 e de conexao com o Broker
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
// variaveis pre definidas
#define FIREBASE_HOST "XXXXXXX.firebaseio.com"
#define FIREBASE_AUTH "XXXXXXXXXXXXXXXXXX"
#define DHTPIN D4 // o sensor dht11 no pino 2(D4 do nodeMCU)
#define DHTTYPE DHT11
// Update these with values suitable for your network.
DHT dht(DHTPIN, DHTTYPE);

const char* ssid = "XXXX"; // Nome da rede wifi
const char* password = "XXXX"; // Senha da rede wifi
const char* mqttServer = "SERVIDOR_NAME.COM"; // servidor MQTT
```

```
29 const int mqttPort = XXXX; //porta do servidor
30 const char* mqttUser = "XXXXX"; //Nome do usiario
31 const char* mqttPassword = "XXXX"; //Senha do usuario
32
33 WiFiClient espClient;
34 PubSubClient client(espClient);
35 // Pequeno mapeamento das portas utilizadas na programacao
36 /*
37 ------------------------------------------------
38 NodeMCU / ESP8266  |  NodeMCU / ESP8266  |
39 D0 = 16            |  D6 = 12            |
40 D1 = 5             |  D7 = 13            |
41 D2 = 4             |  D8 = 15            |
42 D3 = 0             |  D9 = 3             |
43 D4 = 2             |  D10 = 1            |
44 D5 = 14            |                     |
45 ------------------------------------------------
46 */
47 const int lume_Rigth = 12; //iluminacao lado direito
48 const int lume_Left = 13; //iluminacao lado esquerdo
49 const int alter_1 = 14;
50 const int alter_2 = 5;
51 const int LED2 = LED_BUILTIN;
52 const unsigned int portaAnalogica = A0;//Porta analogica do NodeMCU
53 int valor_analogico;
54 long previousMillis = 0; //Variavel de tempo do higrometro
55 long previousMillis2= 0; // Variavel de tempo do LDR
56 long umidadeInterval = 2000;// Tempo em ms do intervalo a ser
      executado
57 long ldrInterval = 5000;
58 float Lux;
59 int event;
60 int respBomba =0;
61 int luzON;
62 int alterValueLux;
63 int bombaAcionada;
64 void mqtt_callback(char* topic, byte* dados_tcp, unsigned int length)
      ;
65
66 void setup() {
67
```

```
68          pinMode(portaAnalogica, INPUT);//porta configurada como
               Entrada
69          pinMode(lume_Rigth, OUTPUT);//porta configurada como saida *
70          pinMode(lume_Left, OUTPUT);// *
71          pinMode(alter_1, OUTPUT);// *
72          pinMode(alter_2, OUTPUT);//*
73          digitalWrite(lume_Left, HIGH);//inicia os reles como
               desligados
74          digitalWrite(lume_Rigth, HIGH);
75
76          delay(300);
77          Serial.begin(115200);
78
79          WiFi.begin(ssid, password);
80          Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
81
82          while (WiFi.status() != WL_CONNECTED)
83          {
84          delay(100);
85          Serial.println("Conectando a WiFi..");
86          }
87          Serial.println("Conectado!");
88          client.setServer(mqttServer, mqttPort);
89          client.setCallback(callback);
90
91          while (!client.connected()) {
92          Serial.println("Conectando ao servidor MQTT...");
93
94          if (client.connect("Projeto", mqttUser, mqttPassword ))
95          {
96
97          Serial.println("Conectado ao servidor MQTT!");
98
99          } else {
100
101         Serial.print("Falha ao conectar ");
102         Serial.print(client.state());
103         delay(2000);
104
105         }
106 }
```

```
107
108 client.publish("Status ","Reiniciado!");//Publicacao em topicos
       especificos
109 client.publish("Placa","Em funcionamento");
110 client.subscribe("LED");//topicos de interesse subInscritos
111 client.subscribe("EVT");
112 client.subscribe("IRG");
113 }
114 /*-----CONTROLE DA LUZ-----*/
115 void controle_luz(int state){
116         if(state == 1){
117          digitalWrite(lume_Rigth, LOW);//logica para acionar um rele
118          client.publish("LUZ","iluminacao 01: ON");
119         }
120         if(state == 2){
121          digitalWrite(lume_Left, LOW);
122          client.publish("LUZ","iluminacao 02: ON");
123         }
124         if(state == 3){
125          digitalWrite(lume_Rigth, LOW);
126          digitalWrite(lume_Left, LOW);
127          luzON = 1;
128          client.publish("LUZ","iluminacao total: ON");
129         }
130         if(state == -1 && event != 1){//Logica para desligar um rele
131          digitalWrite(lume_Rigth, HIGH);
132          client.publish("LUZ","iluminacao 01: OFF");
133         }
134         if(state == -2 && event != 1){
135          digitalWrite(lume_Left, HIGH);
136          client.publish("LUZ","iluminacao 02: OFF");
137         }
138         if(state == -3 && event != 1){
139       digitalWrite(lume_Rigth, HIGH);
140       digitalWrite(lume_Left, HIGH);
141          luzON = 0;
142         client.publish("LUZ","iluminacao total: OFF");
143         }
144
145 }
146 /*-------SENSOR DE DEPENDENCIA LUMINOSA - LDR-------*/
```

```
147 int sensorLdr(){
148         Serial.println("Sensor LDR ");
149         float valorLume;
150         int ADC;
151         char MsgLumeMQTT[12];
152 
153         ADC = analogRead (portaAnalogica);
154         Lux = (float)ADC;
155         sprintf(MsgLumeMQTT, "%f",Lux);
156         if(event == 1){
157         client.publish("LUME",MsgLumeMQTT);
158         }
159         float testBase = Firebase.getFloat("LDR");//testando o valor
               do Firebase
160         Serial.print("Valor do Fire Base: ");
161         Serial.println(testBase);
162         if(Firebase.getFloat("LDR") != 350){
163         valorLume = Firebase.getFloat("LDR");// muda de valor padrao
               caso o mesmo seja alterado
164         }else{
165         valorLume = 350;// valor padrao da ilumina ao
166         Serial.print("Valor padrao: ");
167         Serial.println(valorLume);
168         }
169 
170         if(Lux <= valorLume && event == 1){
171         controle_luz(3);
172         }else if(Lux > valorLume && luzON == 1 && event == 1){
173         controle_luz(-3);
174         luzON = 0;
175         }
176 
177         return Lux;
178 }
179 /*----SENSOR DE TEMPERATRA DA GRAMA DHT11-----*/
180 float sensorDht11(){
181         Serial.println("Sensor DHT");
182         float umidade = dht.readHumidity();
183         float temperatura = dht.readTemperature();
184 
185         char MsgUmidadeMQTT[12];
```

```
186         char MsgTemperaturaMQTT[12];
187         float faketemp = 30.0;
188         if (isnan(temperatura) || isnan(umidade)) // caso o sensor
               apresente falha e retornado um valor falso para testes
189         {
190
191         Serial.println("Falha na leitura do dht11...");
192         return faketemp;
193
194         }
195         else{
196         return temperatura;
197         }
198
199 }
200 /*-------------UMIDADE DA GRAMA-HIGROMETRO----------------*/
201 int umidadeGrama(){
202         Serial.println("Sensor Higrometro");
203         String porcentStr = "";
204         char MsgUmidMQTT[12];
205         char MsgTempMQTT[12];
206         float porcent = 0.0, porcentSeco = 0.0, temperaturaGrama;
207         //Le o valor do pino A0 do sensor
208         valor_analogico = analogRead(portaAnalogica);
209         temperaturaGrama = sensorDht11();
210
211         //Mostra o valor da porta analogica no serial monitor
212          Serial.print("Porta analogica: ");
213         Serial.println(valor_analogico);
214
215         porcentSeco = (float)valor_analogico/1024*100;//Convertendo o
               valor em (%)
216         porcent = 100 - porcentSeco;// retirando a diferenca para
               obter a (%)umidade
217         sprintf(MsgTempMQTT,"%f",temperaturaGrama);
               sprintf(MsgUmidMQTT,"%f",porcent);
218
219         client.publish("hdd", MsgUmidMQTT);//publica o valor da
               umidade
220         client.publish("tmpr",MsgTempMQTT);//publica o valor da
               temperatura
```

```
221
222 //Caso nao esteja ocorrendo um evento esta estrutura e verificada
223         if(event != 1){
224         if (porcent <= 50.0 && temperaturaGrama >= 24.0 &&
               bombaAcionada != 1){
225                 Serial.print("Porcentagem de umidade: ");//Testes no
                       monitor serial
226                 Serial.print(porcent);
227                 Serial.println("Temperatura da Grama: ");
228                 Serial.print(temperaturaGrama);
229                 client.publish("IRG","ON");
230         delay(100);
231         if(respBomba == 1){
232                 bombaAcionada = 1;
233                 client.publish("AVS","Irrigacao Acionada");//Topico
                       de avisos
234            umidadeInterval = 5000;
235         }else{
236         if(respBomba == 0){
237           umidadeInterval = 50000;// Intervalo de verificacao
238         bombaAcionada =0;
239         }else if(respBomba == -1) umidadeInterval = 1000;// intervalo
               de verificacao
240         }
241         }
242         Serial.print("Estado da Bomba :");
243         Serial.print(bombaAcionada);
244         if(porcent >= 80.0 && bombaAcionada == 1 || temperaturaGrama
               < 24.0){// Caso a irrigacao seja acionada
245                 Serial.println("Temperatura da Grama 2: ");
246                 Serial.print(temperaturaGrama);
247                 Serial.println("umidade 2: ");
248                 Serial.print(porcent);
249                 client.publish("IRG","OFF");
250                 client.publish("AVS","Irrigacao desligada");
251                 bombaAcionada = 0;
252                 respBomba = -1;
253          umidadeInterval = 5000;
254         }
255         if(temperaturaGrama > 35.0 && porcent < 75.0 && bombaAcionada
               != 1){//Tratar a temperatura ideal
```

```
256          client.publish("IRG","ON");
257      delay(100);
258         if(respBomba == 1){
259           client.publish("AVS","Irrigacao acionada: Temper.");
260           umidadeInterval = 8000;
261          bombaAcionada = 1;
262         }else{
263         if(respBomba == 0){
264          umidadeInterval = 50000;
265          bombaAcionada = 0;
266         }else if(respBomba == -1) umidadeInterval = 1000;
267         }
268         }else if(temperaturaGrama < 24.0){//Caso a temperatura do
               gramado esteja muito baixa
269          client.publish("AVS","Temperatura do gramado muito Baixa.
               Gramado sujeito a doencas..");
270          client.publish("AVS",MsgTempMQTT);
271         }
272         }else{
273         client.publish("AVS","Irrigacao bloq..");
274         }
275
276 }
277 /*----------------------TOPICOS(DADOS)----------------------*/
278 void callback(char* topic, byte* dados_tcp, unsigned int length) {
279         Serial.println("Topicos MQTT");
280         for (int i = 0; i < length; i++) {
281         }
282         if(strcmp(topic,"IRG")== 0){// Se for o topico de IRRIGACAO
283         if ((char)dados_tcp[0] == 'O' && (char)dados_tcp[1] == 'K') {
284          respBomba = 1;//confirmacao da bomda d agua
285          bombaAcionada = 1;// simboliza o acionamento da bomba d agua
286         }
287         if ((char)dados_tcp[0] == 'W' && (char)dados_tcp[1] == 'T') {
288          respBomba = 0;//Bomba nao acionada-Aguardar...
289         }
290         }
291
292         if(strcmp(topic,"EVT") == 0){//se o topico for o de Evento
293         if ((char)dados_tcp[0] == 'O' && (char)dados_tcp[1] == 'N') {
294          sensorLdr();//verificar a iluminacao
```

```
295          event = 1;//evento acionado
296         } else if((char)dados_tcp[0] == 'O' && (char)dados_tcp[1] ==
              'F'){
297          event = 0;//evento encerrado
298          bombaAcionada = 0;//bomba desligada
299          controle_luz(-3);//desliga todas as luzes
300          alternador();//muda de sensor analogico
301         }
302         if((char)dados_tcp[0] == 'A' && (char)dados_tcp[1] == 'V'){
303         alterValueLux = 1;//variavel nao configurada aqui pois existe
                a limitacao
304         }                    // do aplicativo utilizado em trata-la,
              em outros casos
305         }                      //pode ser utilizada.
306         if(strcmp(topic,"LED")==0){
307         if ((char)dados_tcp[0] == 'L' && (char)dados_tcp[1] == '1') {
308          controle_luz(1);
309         } else if((char)dados_tcp[0] == 'D' && (char)dados_tcp[1] ==
              '1' && event != 1){
310          controle_luz(-1);
311         }
312         if ((char)dados_tcp[0] == 'L' && (char)dados_tcp[1] == '2') {
313          controle_luz(2);
314         } else if((char)dados_tcp[0] == 'D' && (char)dados_tcp[1] ==
              '2' && event != 1){
315          controle_luz(-2);
316         }
317         if ((char)dados_tcp[0] == 'L' && (char)dados_tcp[1] == 'A') {
318         //logica que verifica se o valor padrao da iluminacao deve
              ser alterado
319         if(event == 1 && luzON != 1){
320          float newLux = (float)sensorLdr();
321          Firebase.setFloat("LDR",newLux);
322         }
323         controle_luz(3);
324         } else if((char)dados_tcp[0] == 'D' && (char)dados_tcp[1] ==
              'A' && event != 1){
325          controle_luz(-3);
326         }
327         }
328 }
```

```
329 /*---------------ALTERNADOR DE SENSORES ANALOGICOS-------------------
       */
330 void alternador(){
331         int ativo1, ativo2;
332         if(event != 1 && ativo1 != 1){// Alterna para o sensor de
              umidade
333          digitalWrite(alter_1, LOW);
334          ativo1 = 1;
335          ativo2 = 0;
336          digitalWrite(alter_2, HIGH);
337         }else if(event == 1 && ativo2 != 1){//alterna para o sensor
              de luminosidade
338          digitalWrite(alter_2, LOW);
339          ativo1 = 0;
340          digitalWrite(alter_1, HIGH);
341          ativo2 = 1;
342         }
343 }
344 //Utilize se necessario
345 /*---------CONVERSOR DE PONTO FLUTUANTE----------------------------
346 String conversor(float val) {
347 int i;
348 char buff[10];
349 String valueString = "";
350 dtostrf(val, 4, 6, buff);  //4 is mininum width, 6 is precision
351 Serial.print("String convertida: ");
352 valueString += buff;
353 Serial.println(valueString);
354 return valueString;
355 }//-------------------------------------------------*/
356 void loop() {
357         client.loop();
358         unsigned long currentMillis = millis();//Tempo atual em ms
359         alternador();//inicia o alternador de portas em loop
360         if(event == 1){
361          if(currentMillis - previousMillis2 > ldrInterval){
362           previousMillis2 = currentMillis;//intervalo de
                funcionamento
363           sensorLdr();// sensor utilizado somente em eventos
364          }
365         }
```

```
366
367         //Logica de verificacao do tempo
368         if (currentMillis - previousMillis > umidadeInterval) {
369          previousMillis = currentMillis;    // Salva o tempo atual
370          umidadeGrama();// sensor utilizado em intervalos diferentes
371         }
372 }
```

A.2 Código do Módulo de controle da Bomba d'água

```
1  #include <stdlib.h>
2  #include <ESP8266WiFi.h>
3  #include <PubSubClient.h>
4
5  const char* ssid = "XXXXXX";
6  const char* password = "XXXXXX";
7  const char* mqttServer = "servidor.name.com";
8  const int mqttPort = XXXXX;
9  const char* mqttUser = "xxxxxxXxxx";
10 const char* mqttPassword = "XXXXXXXXX";
11
12 WiFiClient espClient;
13 PubSubClient client(espClient);
14 /*
15 ---------------------------------------------------
16 NodeMCU / ESP8266  |  NodeMCU / ESP8266  |
17 D0 = 16            |  D6 = 12            |
18 D1 = 5             |  D7 = 13            |
19 D2 = 4             |  D8 = 15            |
20 D3 = 0             |  D9 = 3             |
21 D4 = 2             |  D10 = 1            |
22 D5 = 14            |                     |
23 ---------------------------------------------------
24 */
25 const int boia = 14;//pino do sensor boia
26 const int solenoide = 4;//canal da valvula solenoide
27 const int bomba = 0;//canal da bomba d agua
28 int event;
29 int action;
30 const int LED2 = LED_BUILTIN;
```

```
31 int valor_analogico;
32 long previousMillis = 0; // Variavel de controle do tempo
33 long tempoEspera = 0;
34 void mqtt_callback(char* topic, byte* dados_tcp, unsigned int length)
      ;
35
36 void setup() {
37         pinMode(boia, INPUT);//configurado como entrada
38         pinMode(solenoide, OUTPUT);//configurado com saida *
39         pinMode(bomba, OUTPUT);// *
40         digitalWrite(solenoide, HIGH);// iniciado como desligado *
41         digitalWrite(bomba, HIGH);// *
42         delay(300);
43         Serial.begin(115200);
44
45         WiFi.begin(ssid, password);
46         while (WiFi.status() != WL_CONNECTED)
47         {
48         delay(100);
49         Serial.println("Conectando a WiFi..");
50         }
51         Serial.println("Conectado!");
52         client.setServer(mqttServer, mqttPort);
53         client.setCallback(callback);
54
55         while (!client.connected()) {
56         Serial.println("Conectando ao servidor MQTT...");
57
58         if (client.connect("Projeto", mqttUser, mqttPassword ))
59         {
60
61         Serial.println("Conectado ao servidor MQTT!");
62
63         } else {
64
65         Serial.print("Falha ao conectar ");
66         Serial.print(client.state());
67         delay(2000);
68
69         }
70 }
```

```
71
72 client.publish("Status ","Reiniciado!");
73 client.publish("motobomba","Em funcionamento");
74
75 client.subscribe("IRG");//Topicos de interesse
76 client.subscribe("EVT");
77 }
78 /*------------------INTERRUPTOR DA MOTOBOMBA---------------------*/
79 void interruptor(int state){
80         if(state == 1){//liga a solenoide e em seguida a bomba
81          digitalWrite(solenoide, LOW);
82          delay(1000);// atraso de 1 seg
83          digitalWrite(bomba, LOW);
84         }
85         if(state == -1){
86          digitalWrite(bomba, HIGH);// desliga a bomba e em segurida a
                solenooide
87          delay(1000);// atraso se 1 seg
88          digitalWrite(solenoide, HIGH);
89          client.publish("AVS","Bomba desligada");
90         }
91 }
92 /*-----------CONTROLE DA BOMBA------------------*/
93 void controle_bomba(){
94   if(action == 1 && event != 1){
95         if(!digitalRead(boia)){
96          client.publish("IRG","OK");
97          client.publish("AVS","Bomba acionada");
98          interruptor(1);
99         }else{
100          client.publish("IRG","WT");
101          tempoEspera = 5000;
102          client.publish("AVS","Reservatorio: Pouca agua");
103         }
104    }else{
105      Serial.println("Nenhuma acao para bomba de agua");
106    }
107
108 }
109
110 /*----------------TOPICOS(DADOS)-----------------*/
```

```
111 void callback(char* topic, byte* dados_tcp, unsigned int length) {
112         Serial.println("Topicos MQTT");
113         for (int i = 0; i < length; i++) {
114         }
115 
116         if(strcmp(topic,"EVT") == 0){
117         if ((char)dados_tcp[0] == 'O' && (char)dados_tcp[1] == 'N') {
118         event = 1;
119         } else if((char)dados_tcp[0] == 'O' && (char)dados_tcp[1] ==
              'F'){
120         event = 0;
121         }
122 
123         }
124         if(strcmp(topic,"IRG")== 0){//solicitacao de acionamento da
              bomba d agua
125          if ((char)dados_tcp[0] == 'O' && (char)dados_tcp[1] == 'N')
               {
126           action = 1;
127           controle_bomba();
128          } //solicitacao de desligamento da bomba d agua
129          if ((char)dados_tcp[0] == 'O' && (char)dados_tcp[1] == 'F'
               && (char)dados_tcp[2] == 'F') {
130           action = 0;
131           interruptor(-1);
132          }
133    }
134 }
135 
136 void loop() {
137   client.loop();
138   unsigned long currentMillis = millis(); //Tempo atual em ms
139   //Logica de verificacaoo do tempo
140   if (currentMillis - previousMillis > tempoEspera) {
141           previousMillis = currentMillis;    // Salva o tempo atual
142           controle_bomba();
143         if(!digitalRead(boia)){//informa entre intervalos o estado da
              boia
144           client.publish("AVS","Reservatorio: Cheio");
145         }
146   }
```

```
147
148 }
```