UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Departamento de Sistemas de Computação

Sistema de irrigação inteligente

Tiago Vilela Tapparo

São Carlos - SP

Sistema de irrigação inteligente

Tiago Vilela Tapparo

Orientador: Eduardo do Valle Simões

Monografia referente ao projeto de conclusão de curso dentro do escopo da disciplina Projeto de Formatura I (SSC0670) do Departamento de Sistemas de Computação do Instituto de Ciências Matemáticas e de Computação – ICMC-USP para obtenção do título de Engenheiro de Computação.

Área de Concentração: Inteligência Computacional

USP – São Carlos
Novembro de 2016

“Se eu me sentisse ainda mais estúpido do que já sou, eu pensaria que teria atingido o auge da minha carreira; no entanto eu sei o quanto ainda me falta para alcançar a perfeição; e eu vejo cada vez mais claramente que vivo entre artistas de primeira qualidade e sei o que falta a cada um deles. ”

(Frédéric Chopin)

Dedicatória

Dedico este trabalho à minha avó, Aparecida de Jesus Tapparo. Em nossos corações você estará presente eternamente.

Agradecimentos

Agradeço, primeiramente, a meus pais (Maria de Lourdes e José Carlos) por todo amor, carinho, apoio e incentivo que me foi dado para que eu pudesse chegar onde cheguei.

Agradeço meu irmão (Guilherme) e a todos meus familiares por todo apoio e tudo que já fizeram por mim. Cada pequena coisa reflete no meu jeito de ser e fez de mim uma pessoa melhor.

Agradeço a minha namorada Marília e aos meus amigos, por todo companheirismo e por terem feito dessa etapa da minha vida algo único e inesquecível.

Agradeço ao meu orientador Eduardo Simões por todo auxílio e suporte para me ajudar nesse projeto. Além do professor Luiz Antônio Vasques Hellmeister por me ajudar no design e impressão da peça 3D utilizada em meu trabalho.

Resumo

Foi desenvolvido neste trabalho um sistema de irrigação inteligente que se baseia na análise de imagens, dados de sensores e da internet para a tomada de decisões. O sistema consiste em um microprocessador Raspberry Pi conectado a um microcontrolador Arduino UNO por meio de um conjunto de módulos de comunicação sem fio HM-10, que utiliza o padrão *Bluetooth* 4.0. O microcontrolador, por sua vez, controla a direção do *sprinkler*, por meio de um servo motor. O *software* para análise da imagem foi implementado em linguagem C++, por meio da biblioteca OpenCV, enquanto o conjunto de *software* para comunicação entre microprocessador e microcontrolador foi feita em linguagem C. O sistema foi implementado em um cenário de teste que melhor representasse um ambiente doméstico, e utilizou-se das vertentes da domótica para tomada de decisões sobre o escopo do projeto.

Sumário

Lista de Abreviaturas

RGB	Red Green Blue
BGR	Blue Green Red
HSV	Hue Saturation Value
MAC	Media Access Control Address
OpenCV	Open Computer Vision
TTL	Transistor –Transistor Logic
TX	Transmissor
RX	Receptor

Lista de Tabelas

Lista de Figuras

CAPÍTULO 1: INTRODUÇÃO

1.1. Considerações Iniciais

Este capítulo apresenta a contextualização e motivação para o desenvolvimento deste trabalho. Em seguida são apresentados os objetivos e a descrição da proposta. Por fim, é descrita a organização dos capítulos deste trabalho de conclusão de curso.

1.2. Contextualização e Motivação

Devido à elevada evolução dos sistemas eletrônicos e computacionais, associados a tecnologias de comunicação cada vez mais evoluídas, alcançou-se um novo domínio de aplicação tecnológica que tem por objetivo satisfazer necessidades de utilização racional da energia e outros recursos naturais, além de proporcionar uma maior sensação de conforto aos utilizadores das instalações. Essa integração das tecnologias de comunicação de dados com a eletrônica se baseia em um conceito que começou a emergir no início dos anos 80. Esse conjunto de tecnologias aplicada a ambientes residenciais permite a realização de uma vasta gama de aplicações de gestão, local ou remota, de segurança, conforto, gestão de energia, etc. Dessa forma surgiu o conceito que hoje conhecemos como domótica (BRANDÃO, 2008).

O consumo irresponsável (direta e indiretamente) fez com que o Brasil chegasse à marca de 40% de desperdício da água tratada, em 2010. Sendo que em outros países como o Japão, o qual é referência mundial no combate às perdas, em 2004, estas eram iguais a 6,8%. (AQUINO, 2007)

Tendo em vista as novas tendências de estilo de vida das pessoas, no qual se tem menos tempo para as tarefas domésticas, e de problemas com o desperdício de água em áreas residenciais, foi possível estabelecer a motivação e as bases para o projeto de um sistema de irrigação autônomo. Tal sistema tem em vista proporcionar maior conforto doméstico para os usuários, por facilitar uma de suas tarefas rotineiras, além de gerir melhor o consumo de água para tal tarefa.

1.3. Objetivos

Este trabalho visa a implementação e a simulação de um sistema de irrigação inteligente, ou seja, um sistema capaz de identificar se um gramado, ou parte dele, precisa ser regado e realizar a irrigação dessas regiões. Dentre os principais objetivos do projeto destacam-se: o desenvolvimento de um algoritmo para a análise da imagem do gramado, um *software* que realize o gerenciamento e a comunicação entre as diferentes partes do sistema e que faça a tomada de decisão, além do design de uma peça para acoplar o servo motor ao *sprinkler*.

Pretende-se obter um sistema *open source* que analise a imagem do gramado para selecionar as regiões que necessitam de irrigação e com base em informações obtidas da internet e de sensores possa decidir se o gramado deve ser irrigado ou não, e onde. Espera-se que o projeto possua uma taxa de acerto satisfatória e possa, de alguma forma, acarretar uma redução no gasto doméstico com água. Vale enfatizar no fato de que os *softwares* criados serão abertos para a disseminação da tecnologia e eventual mudança para a adequação do sistema para cada tipo de usuário.

1.4. Organização do Trabalho

Neste capítulo apresentou-se a motivação, no contexto atual, e o objetivo do trabalho. No Capítulo 2 é apresentado uma revisão dos métodos que serão adotados, bem como uma revisão bibliográfica referente ao projeto. No Capítulo 3 descreve-se o projeto do sistema de irrigação inteligente, suas principais informações e objetivo. Também se descreve as atividades realizadas durante o desenvolvimento deste projeto, assim como as dificuldades encontradas para a implementação do sistema e os resultados obtidos. Finalmente, no Capítulo 4, são apresentados as conclusões e trabalhos futuros.

CAPÍTULO 2: REVISÃO BIBLIOGRÁFICA

2.1. Considerações Iniciais

Este capítulo apresenta os conceitos adotados, a terminologia básica da área do projeto, assim como os trabalhos da literatura relacionados ao projeto que ajudaram, de alguma forma, a resolver certos problemas encontrados durante o desenvolvimento e implementação do sistema.

2.2. Domótica

A palavra “domótica” é a junção da palavra latina “*domus*” (casa) e do termo “robótica”. O significado se refere ao estudo e implementação de sistemas de sensoriamento e controle dentro do âmbito doméstico, com o objetivo de melhorar a qualidade de vida, aumentar a segurança e viabilizar o uso racional dos recursos para seus habitantes (TONIDANDEL, 2011).

Ainda segundo Tonidandel (2011), um sistema domótico é dividido em vários outros subsistemas, os quais atuam em um campo de controle específico. Atualmente esses sistemas são informatizados e computadorizados. A Figura 1 ilustra essa divisão de subsistemas de sensoriamentos específicos.

Tonidandel (2011) também fala que existem algumas características que tornam um sistema inteligente. Etsas são: ter noção temporal; ser de fácil interação com o usuário; ter capacidade de integração com outros sistemas do ambiente; atuar em várias condições; ser de fácil reprogramação e ter capacidade de se auto-corrigir.

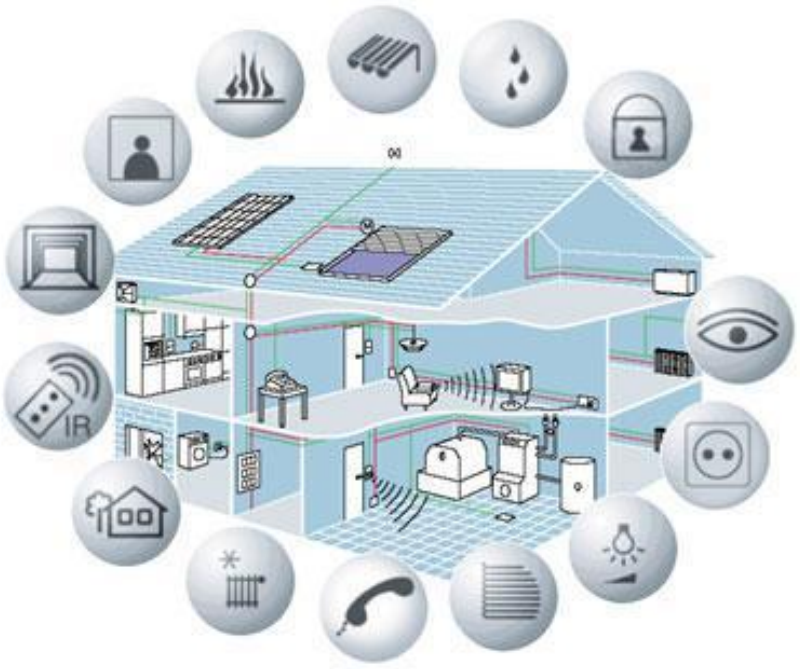

Figura 1: Casa com sistema domótico.

Fonte: www.sislite.pt/domus.htm

2.3. Visão Computacional

É a ciência que, a partir de técnicas computacionais e imagens digitais, visa estimar ou tornar explícitas as propriedades geométricas e dinâmicas do mundo. Essas técnicas têm como objetivo extrair informações relevantes de imagens digitais para uso em uma determinada aplicação.

A visão computacional se relaciona com a área de processamento de imagem, a qual tem como objetivo extração de propriedades das imagens e transformação de imagens. Exemplos práticos dessa área são: realce de imagem, compressão, restauração e extração de características.

2.4. Processamento de Imagem

2.4.1. Modelos de cores

O propósito de um modelo de cor é facilitar a especificação de cores, de algum modo padrão. Em essência, um modelo de cor é uma especificação de um sistema de

coordenadas e um subespaço dentro desse sistema em que cada cor é representada por um único ponto (GONZALES e WOODS, 2000).

Ainda segundo Gonzales e Woods (2000), a maioria dos modelos de cores em uso hoje são orientados tanto para *hardware* (monitores e impressoras) ou para aplicações onde a manipulação de cor é um objetivo (na criação de gráficos a cores para a animação). Em termos de processamento de imagens digitais, os modelos orientados para o *hardware* mais comumente utilizados na prática são o modelo RGB para monitores em cores e uma ampla classe de câmeras de vídeo em cores; o modelo HSV, o qual melhor corresponde como os seres humanos descrevem e interpretam a cor. O modelo HSV também tem a vantagem de desacoplar a informação de cor e da escala de cinzentos numa imagem, tornando-o adequado para muitas das técnicas de escala de cinzentos.

2.4.1.1. RGB

O modelo RGB é baseado em um sistema de coordenadas cartesianas no qual cada coordenada representa a intensidade de uma das cores primárias: vermelho, verde e azul.

Como é possível ver na Figura 2, uma cor é especificada por sua posição dentro do cubo (limites) do sistema RGB. O valor máximo para cada coordenada é de 255 e os eixos X, Y e Z correspondem às intensidades das cores primárias azul, vermelho e verde, respectivamente.

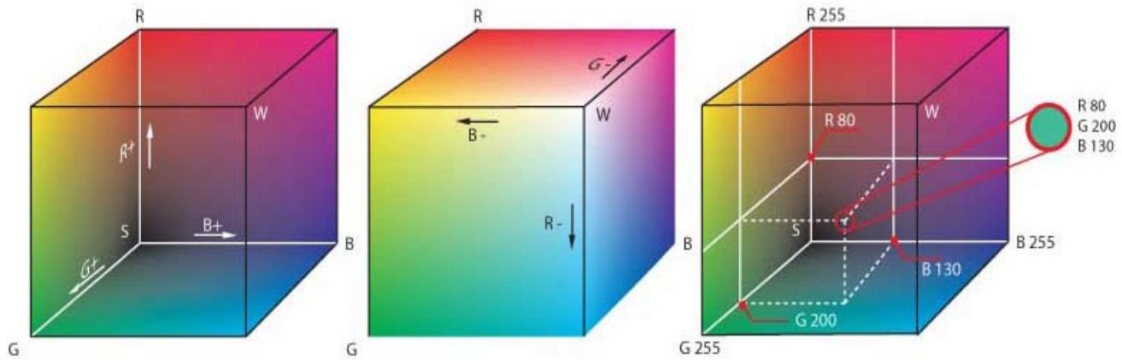

Figura 2: Sistema de coordenadas RGB.
Fonte: Slides Professor Valdir Grassi Jr.

2.4.1.2. HSV

O modelo HSV é baseado em um sistema no qual as cores são descritas de uma maneira mais natural e intuitiva para os seres humanos. O *hue* (ou nuança) é a propriedade da cor que se refere à cor pura, propriamente dito (azul, amarelo, laranja, vermelho, etc). A saturação apresenta a medida do quão diluída na luz branca a cor pura está. Por fim, o valor (ou intensidade) mede o brilho da cor, ou seja, a variação entre o branco e o preto (se uma cor é mais próxima do preto ela é escura, caso contrário, se for mais próxima do branco ela é clara).

Como é possível ver na Figura 3, podemos ilustrar os limites do sistema HSV com um prisma, onde a variação ao redor do perímetro de sua base representa o *hue*, a saturação é representada pela variação no raio da base e a intensidade é a variação ao longo da altura. O valor máximo para o *hue* é de 360, enquanto a saturação e a intensidade variam de 0 a 100.

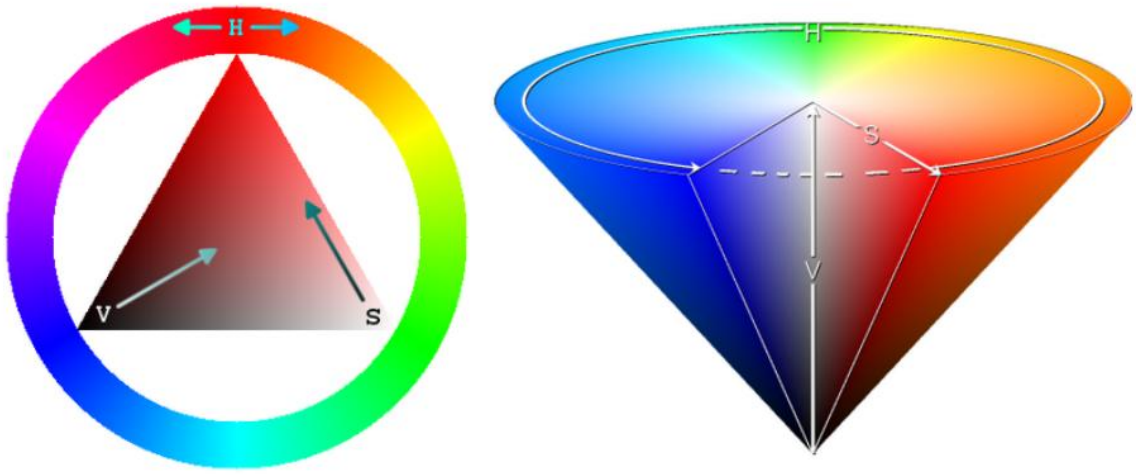

Figura 3: Sistema de coordenadas HSV.
Fonte: Slides Professor Valdir Grassi Jr.

2.4.2. OpenCV

OpenCV[1] é uma biblioteca de *software* de visão computacional e de aprendizagem de máquina. A mesma foi criada com o intuito de fornecer uma ferramenta comum para aplicações de visão computacional e acelerar seu uso em produtos comerciais. Por ser um produto licenciado-BSD, a OpenCV torna mais fácil para as empresas utilizar e modificar seu código. A biblioteca conta com uma gama de funções prontas para utilização, normalmente implementadas nas linguagens de programação C, C++ e Python.

2.4.3. Segmentação de imagens

Segmentação de imagem é uma técnica utilizada para separar uma determinada parte de uma imagem, a fim de se obter uma informação específica.

Segundo Gonzalez e Woods (2000), usualmente, os algoritmos de segmentação utilizam duas propriedades principais: descontinuidade e similaridade.

A segmentação baseada em descontinuidade baseia-se nas mudanças bruscas nos níveis de cores de uma imagem. Essas descontinuidades ocorrem por situações diversas: descontinuidade da normal das superfícies, descontinuidade em profundidade, descontinuidade na refletância da superfície e descontinuidade de iluminação (SIMÕES, 2000).

Já a segmentação baseada em similaridade, segundo Simões (2000), procura fazer o agrupamento das regiões com características semelhantes. No caso, as características mais relevantes seriam as cores e texturas da imagem.

2.4.4. Processamento Morfológico

O processamento morfológico descreve uma gama de técnicas de processamento de imagem que lida com a forma em uma imagem. Operações morfológicas são tipicamente aplicadas para remover imperfeições introduzidas durante a segmentação de uma imagem.

---

[1] http://opencv.org/

Usamos a matemática morfológica como uma ferramenta para extração de componentes de imagem que são úteis na representação e descrição da forma de uma região tal como limites, esqueleto, etc. O objetivo das operações morfológicas é tentar simplificar os dados de imagem, preservar características essenciais da forma e eliminar o ruído.

2.4.4.1. Abertura e Fechamento

A abertura e o fechamento são operações morfológicas utilizadas no processamento de imagens. A abertura tende a suavizar o contorno do objeto, romper os istmos e eliminar saliências finas. O fechamento também suaviza contorno. Entretanto, ele é utilizado para outras finalidades, como fundir descontinuidades estreitas, alongar golfos finos, eliminar buracos e preencher lacunas em um contorno. Na Figura 4 está representado uma operação de abertura e logo em seguida uma operação de fechamento.

Imagem original

Após Abertura

Após Fechamento da Abertura

Figura 4: Exemplo de operações de Abertura e Fechamento

Fonte: Elaborada pelo autor

2.5. Bluetooth

O *Bluetooth*[2] é uma especificação de rede sem fio de âmbito pessoal para transmissão de dados projetado para baixo consumo de energia com baixo alcance, baseado em microchips transmissores de baixo custo em cada dispositivo. As especificações do *Bluetooth* foram desenvolvidas e licenciadas pelo "Bluetooth Special Interest Group".

Esse padrão provê uma maneira de conectar e trocar informações entre dispositivos como telefones celulares, computadores, impressoras, câmeras digitais e consoles de videogames. Os dispositivos usam um sistema de comunicação via rádio, com comprimento de onda com frequência na banda de 2,4 GHz, por isso não há a necessidade de estarem na linha de visão um do outro, ou mesmo no mesmo ambiente, contanto que o sinal recebido tenha potência suficiente.

2.6. Topologia de rede em estrela

Na topologia em estrela, todos os nós periféricos são ligados em um centralizador. Esse nó central tem a função de iniciar e coordenar os nós periféricos, ele também é responsável pelo recebimento e envio de comandos entre os nós da rede, uma vez que os nós periféricos não se comunicam diretamente, pois só possuem conexão com o nó central. Ilustra-se na Figura 5 essa rede, onde o nó que coordena a rede permanece no centro, enquanto os demais nós são dispostos ao redor do mesmo, dando uma aparência de estrela.

[2] https://developer.bluetooth.org/

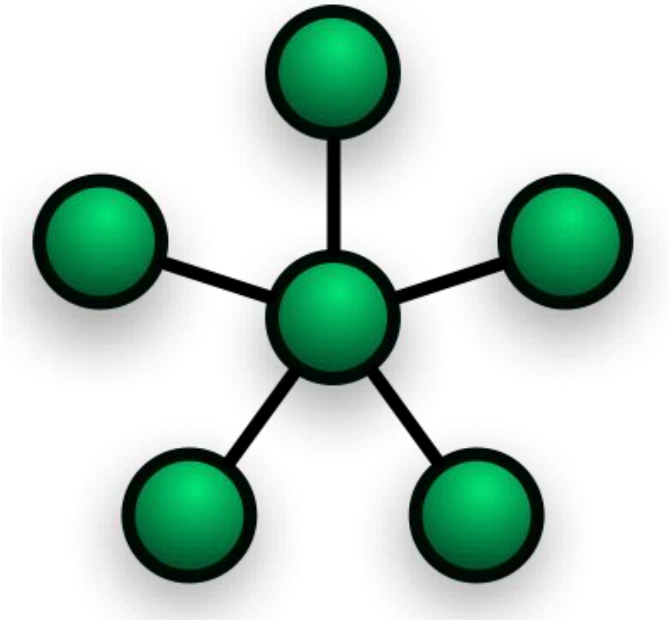

Figura 5: Topologia de rede em estrela.
Fonte: pt.wikipedia.org

Algumas das vantagens do uso dessa topologia são na facilidade de inserção de novos dispositivos na rede, na supervisão de toda a comunicação provida pelo nó central e, pelo fato de que se há algum problema com uma conexão, isso não gera problemas para o restante da rede.

2.7. Machine Learning

Machine Learning é um método de análise de dados que automatiza o desenvolvimento de modelos analíticos. É possível que os computadores encontrem *insights* ocultos sem serem explicitamente programados para procurar algo específico ao se utilizar algoritmos que aprendem interativamente a partir de dados.

Quando são expostos a novos dados, os modelos possuem a capacidade de se adaptar de forma independente. Eles aprendem com os cálculos anteriores para produzir decisões e resultados confiáveis e reproduzíveis. É uma ciência que não é nova, mas que está ganhando um novo impulso (SAS, 2016).

SAS (2016) também afirma que por causa das novas tecnologias de computação, o aprendizado de máquina de hoje não é como o do passado. Enquanto vários algoritmos já são usados há muito tempo, a aplicação de cálculos matemáticos complexos em big data é uma aplicação recente.

2.8. Considerações Finais

Neste capítulo, quis-se apresentar brevemente os conceitos utilizados para o desenvolvimento do projeto. No próximo capítulo será abordado com mais detalhes como os módulos do sistema foram projetados, assim como os resultados obtidos e algumas considerações relevantes sobre o projeto como um todo.

CAPÍTULO 3: DESENVOLVIMENTO DO TRABALHO

3.1. Considerações Iniciais

Este capítulo tem por objetivo ilustrar as principais implementações de *software* e *hardware* utilizadas no decorrer do projeto. Assim como toda metodologia utilizada em cada módulo do trabalho e decisões tomadas ao longo da implementação, tendo em vista praticidade, funcionamento e prevenção contra falhas. Por fim, este capítulo tem como finalidade a exposição e análise dos resultados obtidos, e uma análise crítica sobre o projeto, analisando dificuldades encontradas e limitações da aplicação.

3.2. Projeto

O sistema de irrigação inteligente foi projetado com o intuito de realizar uma tarefa doméstica de forma eficiente e autônoma. Também faz parte do escopo do projeto que ele seja de fácil uso e modificação, tanto para um usuário leigo, quanto para um que possua conhecimentos de programação e eletrônica. Todos os códigos fonte implementados são abertos e disponíveis em um repositório no GitHub[3]. Quanto à parte eletrônica, tentou-se utilizar componentes que se adequassem às necessidades do projeto e que fossem de baixo custo.

O sistema é composto por um módulo Raspberry (central) e vários módulos Arduino (periféricos), como está ilustrado na Figura 6. A função do módulo central é realizar o processamento bruto das informações e enviar comandos para os periféricos, cuja função é coletar dados sobre a umidade do solo para a tomada de decisão final e realizar os processos mecânicos.

[3] https://github.com/tiagotvt/HomeControl/tree/master/Tiago

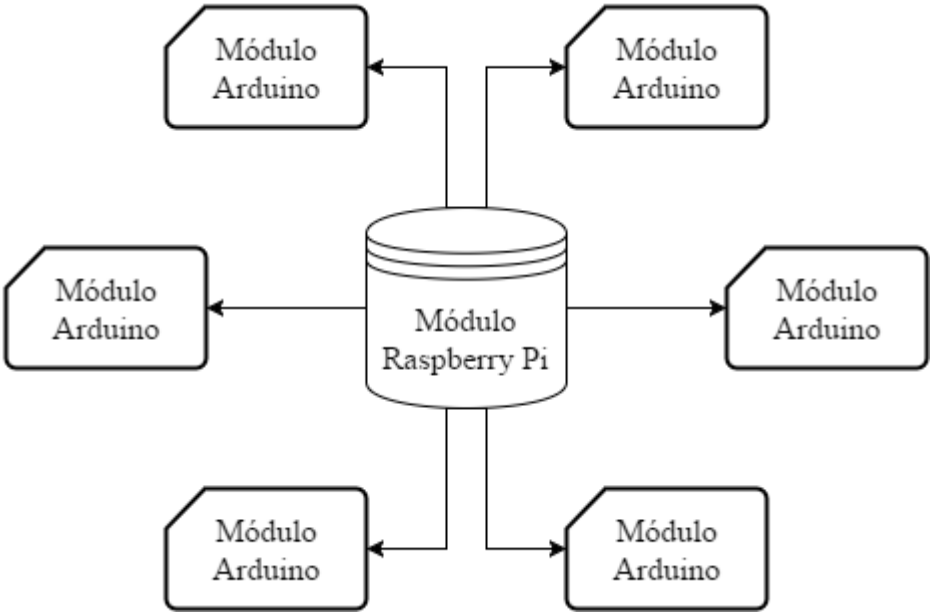

Figura 6: Desenho esquemático do sistema.

Fonte: Elaborada pelo autor

Pode-se observar um fluxograma do funcionamento do sistema como um todo na Figura 7. Os próximos parágrafos descrevem mais detalhadamente a execução do mesmo.

O sistema pode ser considerado inicializado quando o módulo central está executando a interface de usuário e os módulos periféricos não estão em seu estado de *stand-by*. Tendo o sistema iniciado, o módulo Raspberry Pi obtém os resultados do processamento da foto do gramado e a probabilidade de chuva daquele dia. Após a análise desses dados, o módulo central deve decidir se vai enviar o comando aos módulos periféricos ou não. Caso existam uma ou mais regiões com grama seca e a chance de chuva seja baixa, é criada uma lista com todos os *sprinklers* que serão utilizados e os ângulos que devem ser movimentados para molhar cada área. Após isso, envia-se os comandos de irrigação para os módulos periféricos através do módulo *Bluetooth* HM-10.

Ao receber o comando do módulo central, o módulo Arduino faz a verificação da umidade da grama. Caso o valor da umidade esteja abaixo do valor pré-estabelecido, o *sprinkler* é rotacionado para a posição recebida do comando e o jato d'água é liberado, caso contrário, o comando é ignorado e nada acontece.

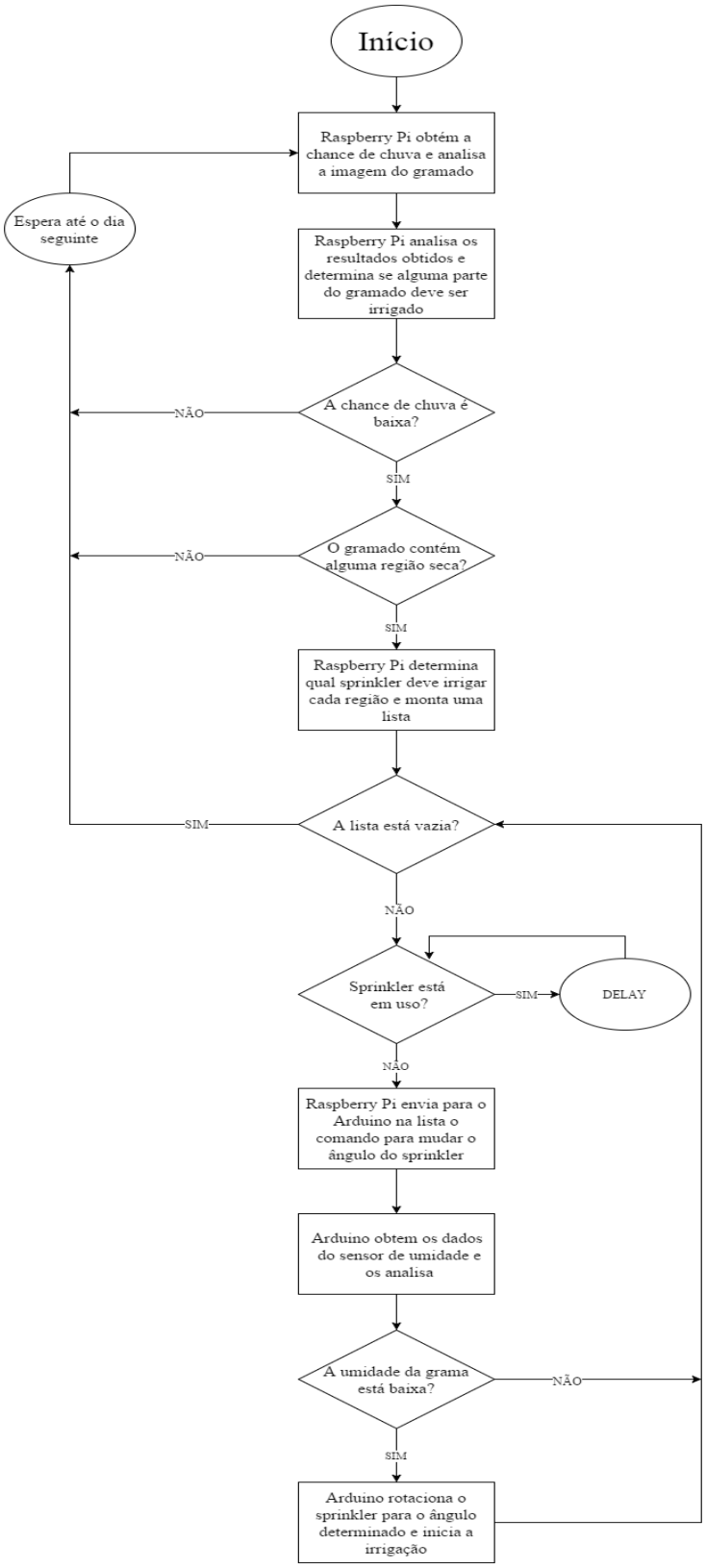

Figura 7: Fluxograma do funcionamento do sistema.

Fonte: Elaborada pelo autor

3.2.1. Módulos do Sistema

3.2.1.1. Módulo Raspberry Pi

O módulo Raspberry Pi é único no sistema. Sua função é processar as imagens da grama e buscar, através da internet, a probabilidade de chuva ao longo do dia, para que seja possível a tomada de decisão do envio de comando de irrigação. Esse módulo é composto pela placa Raspberry Pi B+ e o módulo *Bluetooth* HM-10. A representação do módulo central está ilustrada na Figura 9.

Para o desenvolvimento dessa parte do projeto, foram utilizados, no *Bluetooth*, os pinos 1 e 2 (TX e RX), para a comunicação serial com a Raspberry, e os pinos 12 e 13 (VCC e GND) para a alimentação do mesmo. A Figura 8 apresenta o desenho esquemático dos pinos do *Bluetooth*. Na Raspberry Pi, foram usados os pinos 1 e 6 (3,3V e GND) para alimentar o módulo HM-10 e os pinos 8 e 10 (TX e RX) para a comunicação serial. Vale ressaltar que, para que seja possível a comunicação serial, os pinos TX e RX devem ser conectados de maneira que um pino de transmissão de um dispositivo esteja conectado ao pino receptor do outro.

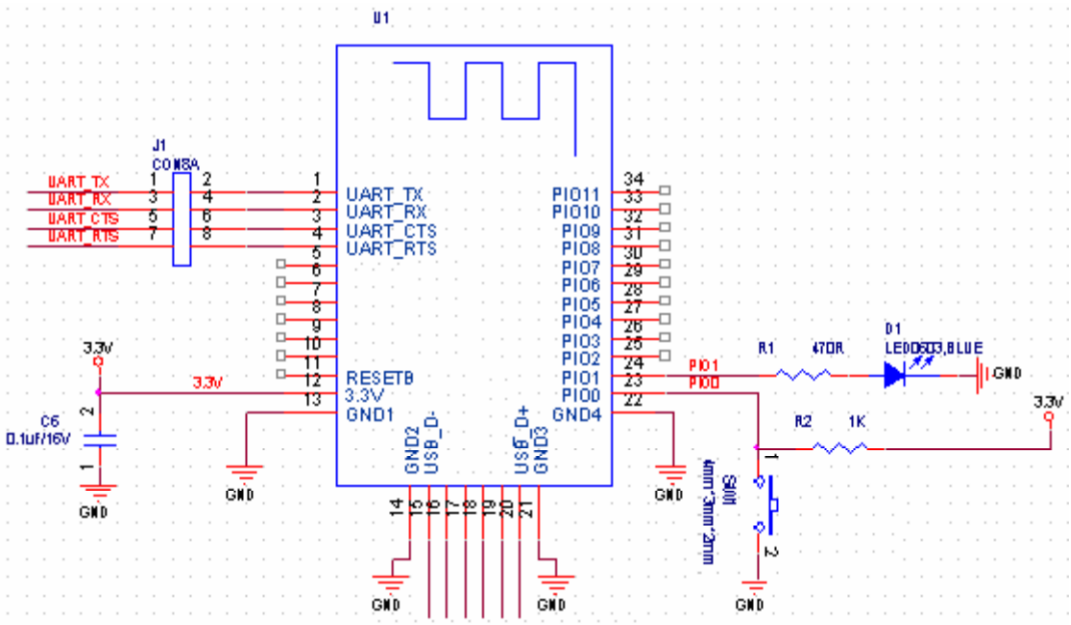

Figura 8: Desenho esquemático do módulo HM-10

Fonte: HM-10 Module Datasheet

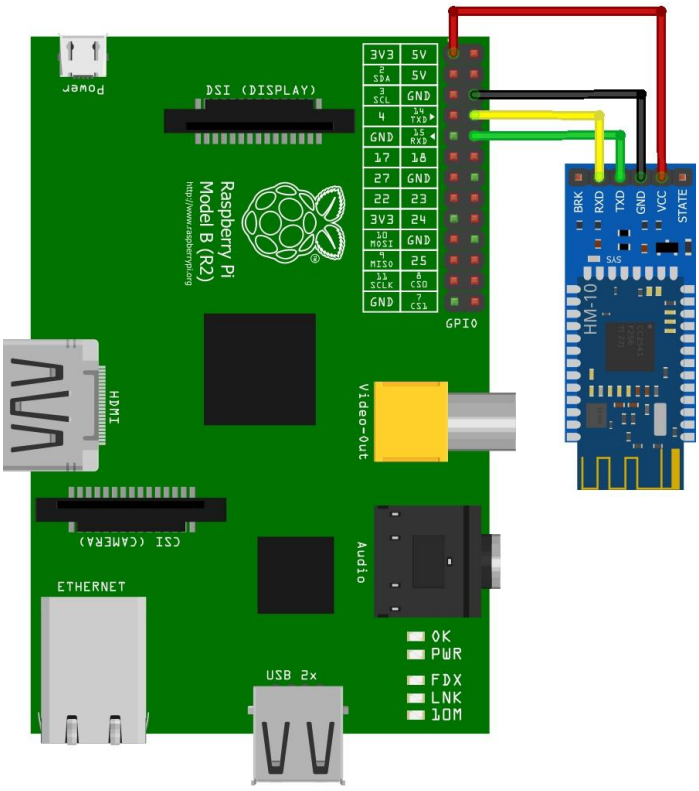

Figura 9: Módulo Raspberry Pi

Fonte: Elaborada pelo autor

3.2.1.2. Módulo Arduino

O módulo Arduino é responsável por todos os processos mecânicos do sistema e pela tomada de decisão final. Esse módulo é composto por uma placa Arduino UNO R3, um módulo *Bluetooth* HM-10, um servo motor, um sensor de umidade, um módulo relé, um *led* multicor, um botão ativador e um conversor de nível de tensão. A representação do módulo periférico está ilustrada na Figura 10.

Nessa parte do projeto, tudo é controlado a partir da placa Arduino, portanto cada pino foi utilizado para uma função específica. Quanto aos pinos digitais: o pino 3 foi usado para controlar o relé; os pinos 6, 10 e 11 controlam a cor do LED multicolorido; os pinos 7 e 8 foram programados para funcionar como receptor e transmissor, respectivamente; e o pino 9, por ser do tipo PWM, envia os sinais para o servo motor girar. O pino analógico 0 tem a função de fazer a leitura dos dados obtidos pelo sensor de umidade. E os pinos GND, 5V e 3,3V alimentam todos os componentes do circuito. Vale ressaltar que o módulo

Bluetooth opera a uma tensão de 3,3V em suas portas TX e RX, porém as portas do Arduino operam a 5V, por isso foi necessário incluir no circuito um conversor de nível de tensão bidirecional, para diminuir a tensão dessas portas, para que a conexão pudesse ser feita.

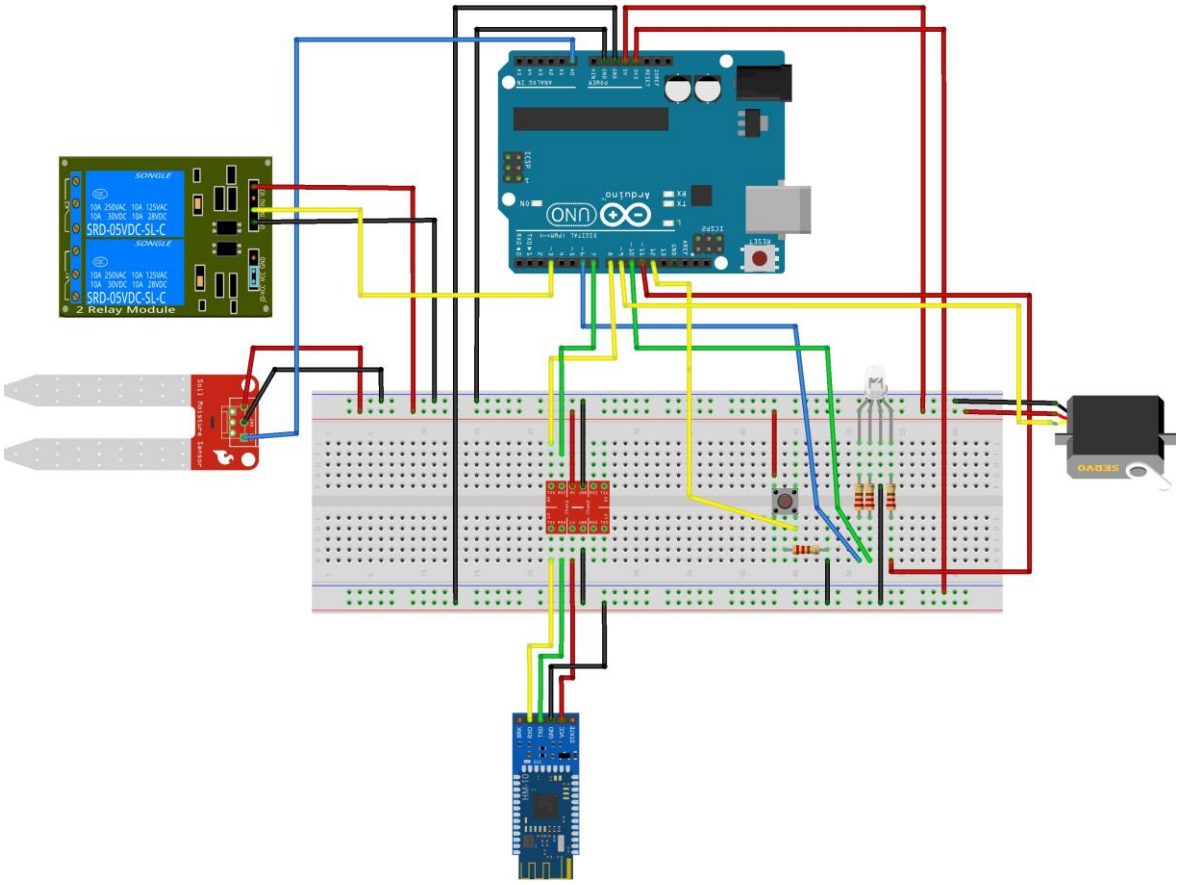

Figura 10: Módulo Arduino.

Fonte: Elaborada pelo autor

3.2.2. Hardware

3.2.2.1. Arduino UNO R3

Arduino é uma plataforma *open-source* baseada em um conceito de *hardware* e *software* de fácil uso. Essa plataforma hibrida é composta pela placa Arduino em conjunto com outros microcontroladores compondo a parte de *hardware*, enquanto a parte de *software* se faz presente com a programação da placa com o uso da linguagem Arduino (baseada na linguagem Wiring e muito similar à C) a partir de uma IDE de

desenvolvimento própria e aberta. A placa presente neste projeto é a Arduino UNO R3 (Figura 6), as especificações da mesma se encontram na Tabela 1.

Figura 11: Arduino UNO R3.

Fonte: www.farnell.com/datasheets/1682209.pdf

Tabela 1: Especificações do Arduino UNO R3.

Microcontolador	ATmega328
Voltagem de operação	5V
Voltagem de entrada	7-12V
Pinos de entrada/saída digital	14 (os quais 6 possuem saída PWM)
Pinos de entrada analógica	6
Corrente DC por pino de entrada/saída	40mA
Corrente DC no pino de 3,3 V	50mA
Memória Flash	32 KB
SRAM	2 KB
EEPROM	1 KB

Velocidade de Clock	16 MHz
Dimensões	75,14mm x 53,51mm x 15,08mm

O uso dessa plataforma se faz de grande valor neste projeto por possuir características semelhantes às que foram propostas inicialmente para o sistema: ser barata (U$24,00), para ser acessível a uma maior quantidade de pessoas; é multiplataforma, dessa forma pouco importa o sistema operacional utilizado pelo usuário que faça uso do projeto; possuir um ambiente de programação simples e intuitivo, para que até o mais iniciante dos usuários possa usá-lo; e possuir *hardware* e *software* extensivos e abertos, para que a reutilização de código para uso e modificação seja a maior possível, independente dos módulos de *hardware* utilizados.

3.2.2.2. Raspberry Pi B+

A Raspberry Pi é um microprocessador de baixo custo (as versões mais recentes custam U$30,00) que possui diversas funções de um computador convencional, como possibilidade de navegar na internet, exibir arquivos e editar documentos. Por ser um computador razoavelmente barato e por oferecer várias ferramentas *open source*, a Raspberry Pi é muito utilizada para o ensino de programação e eletrônica básica.

Alguns dos motivos pelos quais utilizamos essa plataforma no projeto é o fato de que a Raspberry Pi possui uma capacidade de armazenamento que o Arduino não oferece e processamento elevado para um computador tão portátil e de baixo consumo. Isso se mostra ideal para nossa aplicação, pois precisamos armazenar e realizar o processamento da imagem do gramado. O modelo utilizado no projeto foi a Raspberry Pi B+ (Figura 7), a qual será a unidade central do sistema, e as especificações da mesma se encontram na Tabela 2. Mesmo que seja um modelo antigo, o projeto pode ser aplicado em modelos mais recentes da Raspberry Pi sem que ocorra problemas no funcionamento do mesmo.

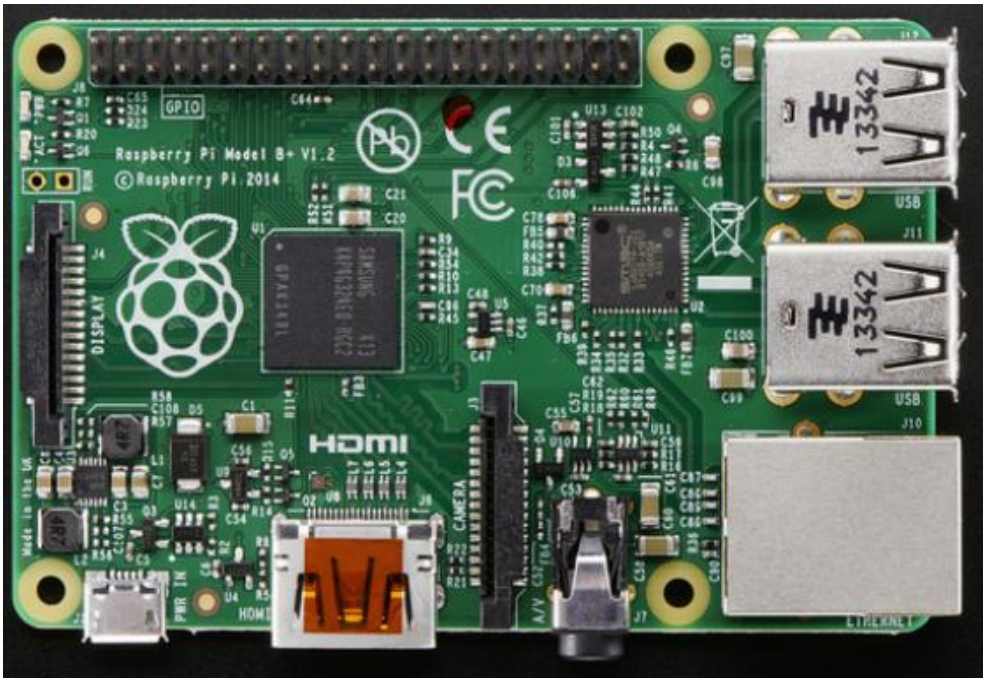

Figura 12: Raspberry Pi B+

Fonte: www.adafruit.com

Tabela 2: Especificações da Raspberry Pi B+.

Processador	Broadcom SoC
Clock do Processador	700 MHz
Memória RAM	512 MB
Conector de Rede	Ethernet 10/100
Conector de Vídeo	HDMI
Conectores USB	2.0 (4x)
Pinos de entrada/saída	40
Consumo	5V; 2A; entre 0,5W e 1W
Dimensões	85mm x 56mm x 17 mm

3.2.2.3. Módulo Bluetooth HM-10

O módulo *Bluetooth* HM-10 (Figura 8) permite a comunicação sem fio entre os módulos desse projeto. Tal módulo utiliza o padrão da tecnologia *Bluetooth* 4.0 que é mantido por um "grupo de interesse especial" (ou SIG - *special interest group* em inglês) formado por mais de 25 mil empresas, dentre elas Apple, Intel, Microsoft e Toshiba (Bluetooth SIG, 2016). As especificações do mesmo se encontram na Tabela 3.

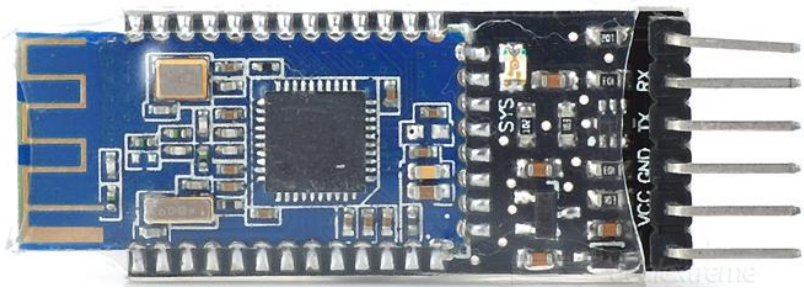

Figura 13: Módulo *Bluetooth* HM-10

Fonte: http://www.dx.com/

O uso dessa tecnologia se faz necessária e eficiente ao sistema proposto, pelo fato de possuir baixo consumo de energia e um alcance consideravelmente grande. Outro fator fundamental para o uso de uma comunicação sem fio no projeto, se deve à simples razão de ser um sistema de irrigação e a possibilidade de haver contato de algum componente eletrônico com a água deve ser evitado o máximo possível. Logo, a possibilidade de se evitar o uso de fios para a comunicação do módulo Arduino com o módulo Raspberry Pi, torna o sistema menos suscetível a possíveis falhas ou curtos-circuitos.

3.2.2.4. Servo motor Tower Pro MG996R

O servo motor Tower Pro MG996R (Figura 9) permite que o *sprinkler* seja posicionado no ângulo determinado, para que apenas a região considerada seca seja irrigada. A escolha do modelo em questão foi feita devido ao fato do mesmo possuir um torque de até 15 kg, o que facilita na rotação do *sprinkler*. As demais especificações desse servo motor se encontram na Tabela 3.

Figura 14: Servo motor Tower Pro MG996R

Fonte: aeroxing.com

Tabela 3: Especificações do servo motor Tower Pro MG996R

Tensão de operação	4,8V - 7,2V
Velocidade de operação (4,8V)	0.17sec/60graus
Velocidade de operação (6V)	0.13sec/60graus
Torque (4,8V)	13 kg.cm
Torque (6V)	15 kg.cm
Dimensões	40mm x 19mm x 43mm

3.2.2.5. Sensor de umidade de solo YL-69

Este sensor tem a capacidade de medir a umidade do solo. Ao se aplicar uma pequena tensão nos terminais do módulo YL-69, o valor da corrente que passa por esse módulo, que depende da resistência gerada no solo, é diretamente proporcional à umidade do solo.

O Sensor consiste em dois terminais separados e um módulo YL-38 que possui um circuito comparador LM393 SMD. O valor de retorno do sistema, que é lido pelo Arduino, é um valor inteiro entre 476 e 1023, sendo esses os valores para quando o sensor está imerso na água e completamente seco, respectivamente. As especificações do sensor se encontram na Tabela 4.

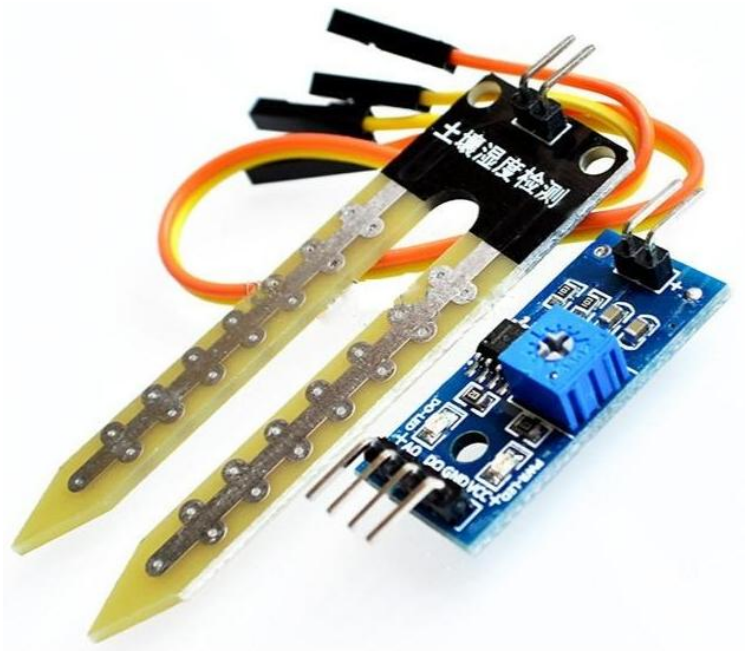

Figura 15: Sensor de umidade YL-69 e módulo YL-38.

Fonte: www.mercadolivre.com.br

Tabela 4: Especificações do sensor de umidade YL-69 e do módulo YL-38.

Voltagem de Entrada	3,3V - 5V
Voltagem de Saída	0V - 4,2V
Corrente	35mA
Dimensões YL-38	30mm x 16mm
Dimensões YL-69	60mm x 30mm

3.2.2.6. Módulo de relé de dois canais

Este módulo relé (Figura 11) possui 2 relés de 1 canal de 5V com interface padrão TTL, que pode ser controlado diretamente por um microcontrolador. Através dele, pode-se enviar sinais digitais para cada relé e, dessa forma, controlar equipamentos de alta corrente, como motores AC ou DC, eletroímãs, bombas hidráulicas, lâmpadas etc. Isso o torna ideal para aplicações de automação residencial. Para este projeto, esse módulo tem a função de ligar as válvulas de água dos *sprinklers*.

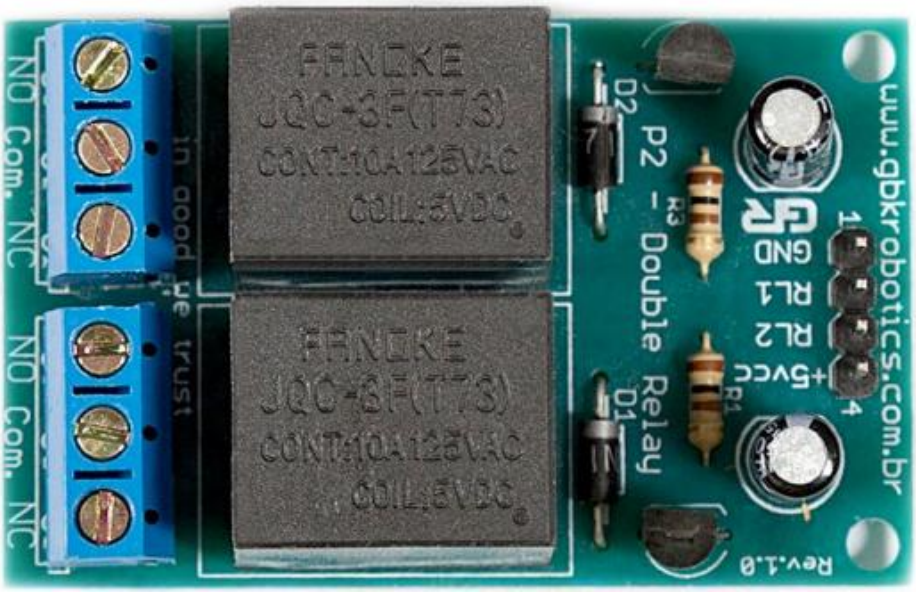

Figura 16: Módulo de relé de dois canais.

Fonte: www.gbkrobotics.com.br

3.2.2.7. Peça de impressão 3D

A impressão 3D é uma tecnologia que ao longo dos últimos anos vem ganhando muito destaque por suas aplicações e despertando o interesse de empresas e entusiastas de tecnologia. Assim como na impressão convencional, de fotos e documentos, para se imprimir um objeto em 3D necessita-se de um arquivo que contém todas as informações do mesmo.

Uma grande semelhança com a programação é que na modelagem de objetos 3D pode-se fazer o reaproveitamento de arquivos disponibilizados gratuitamente na internet

para uso próprio ou para criação de objetos diferentes, assim como ocorre com códigos fonte abertos. E a comunidade que disponibiliza esses arquivos para outros usuários já é grande e tende a crescer ainda mais nos próximos anos.

Tendo em vista uma das vertentes do projeto que é a criação de um sistema de fácil instalação e que fosse aberto a outros usuários, decidiu-se fazer peças (Figuras 12 e 13) de encaixe e de suporte, para que o servo motor pudesse controlar o *sprinkler*, utilizando essa tecnologia. A modelagem do objeto em 3D foi feita com o *software* Solid Edge[4], que possui uma licença gratuita para estudantes, e a impressão foi feita com a impressora 3D Cube[5].

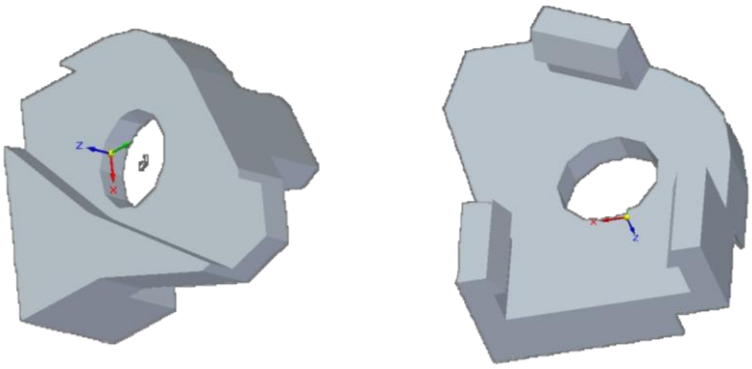

Figura 17: Peça para o encaixe do servo motor e o *sprinkler*.
Fonte: Elaborada pelo autor

[4] http://www.plm.automation.siemens.com/en_us/index.shtml

[5] http://www.3dsystems.com/shop/support/cube/videos

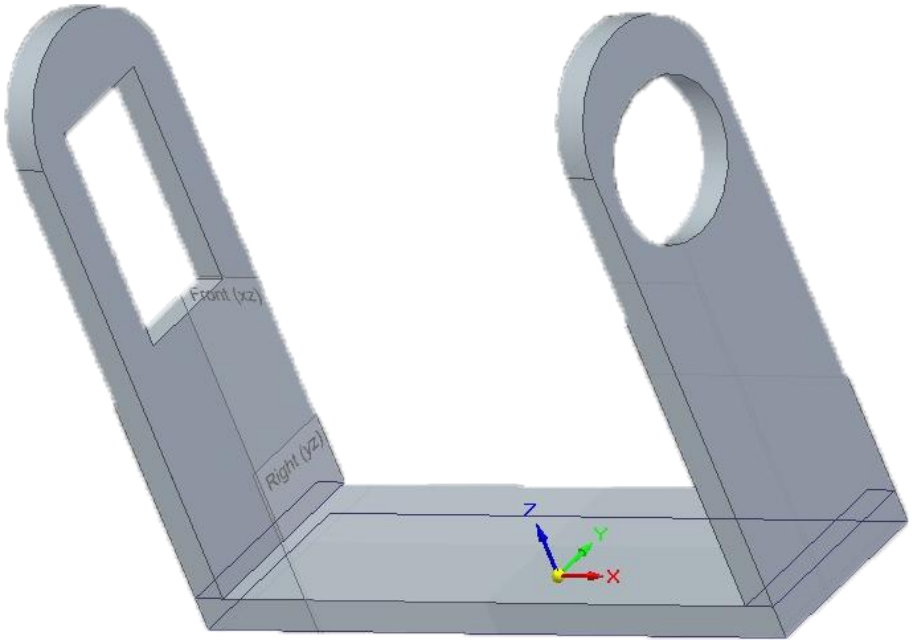

Figura 18: Peça para suporte do servo motor sobre o *sprinkler*.

Fonte: Elaborada pelo autor

3.2.2.8. Válvula de entrada de água

Para controlar a saída de água, utilizou-se uma válvula de entrada de água reta do modelo EVA 01 da marca EMICOL[6] (Figura 19). Essa peça é controlada pelo módulo de relé e é acionada após o *sprinkler* ser posicionado no ângulo correto. Enquanto desligada, ela impede a passagem de água e ao ser acionada, bombeia água com uma vazão que pode variar entre 7 l/min e 40 L/min. As demais especificações da mesma se encontram na Tabela 5. Essa peça se adequa bem ao padrão do projeto por ser de baixo custo (R$14,00) e ser eficiente em sua tarefa.

[6] http://www.emicol.com.br/site/arquivos/ProdutosCatalogo/EVA%2001_por.pdf

Figura 19: Válvula de entrada de água

Fonte: http://www.emicol.com.br/

Tabela 5: Especificações da válvula de entrada de água

Tensão da bobina	127V / 60Hz
Pressão de operação	0,2 kgf/cm² a 8 kgf/cm²
Vazão mínima	7 l/min
Vazão máxima	40 l/min

3.2.3. Software

3.2.3.1. Código para treinamento de máquina

O código para treinamento de máquina tem como objetivo a obtenção do melhor valor possível para os parâmetros que determinam se a grama está seca ou não. A partir de uma foto do gramado, o programa faz o seu próprio diagnóstico sobre qual região deve ser irrigada e esse resultado é comparado com o resultado que um usuário insere mecanicamente. Após essa comparação, é verificado se há alguma divergência entre os resultados e, caso exista alguma diferença, os parâmetros são atualizados. Este processo é

repetido várias vezes e com várias amostras de fotos do gramado, desse modo conseguimos ensinar ao programa quais os limites de cores para uma grama saudável e obter parâmetros que permitam obter uma alta taxa de acerto na identificação de regiões que precisam ser irrigadas.

3.2.3.2. Código de processamento da imagem do gramado

O código de processamento da imagem do gramado é, de certa forma, muito parecido com o utilizado para o treinamento de máquina, pois ele tem também tem como função identificar regiões secas do gramado. Entretanto, esse programa utiliza os parâmetros obtidos no treinamento para realizar um diagnóstico de forma autônoma, sem precisar da validação de um usuário.

3.2.3.3. Código de interface do usuário

O código de interface do usuário é o programa que tem a função de estabelecer comunicação entre a Raspberry Pi e o Arduino e enviar comandos para o mesmo, dependendo dos resultados obtidos no processamento da imagem do gramado. Outro fator que influencia nessa decisão de envio de comando é a previsão climática do dia, a qual é obtida pela internet e possui a porcentagem de chance de chuva ao longo do dia, que caso esteja acima do máximo especificado, impede o envio do comando.

3.2.3.4. Código do módulo Arduino

O código do módulo Arduino é o responsável pelo funcionamento de todas as funções que envolvem a placa Arduino neste projeto, sendo elas: a inicialização do módulo; a comunicação com a Raspberry Pi; a leitura do sensor de umidade; a tomada de decisão a partir da leitura do sensor; movimentação do servo motor; e o controle do relé para ligar a válvula de água e iniciar a irrigação do gramado.

3.3. Descrição das Atividades Realizadas

3.3.1. Módulo Raspberry Pi

3.3.1.1 Software de treinamento de máquina

O *software* para treinamento de máquina[7] foi desenvolvido em linguagem C++ e feito para ser executado via terminal. O programa tem como objetivo obter valores para os limites máximo e mínimo de cor RGB, usados no processamento das imagens do gramado. A representação esquemática do *software* de treinamento e o módulo Raspberry Pi está ilustrada na Figura 20.

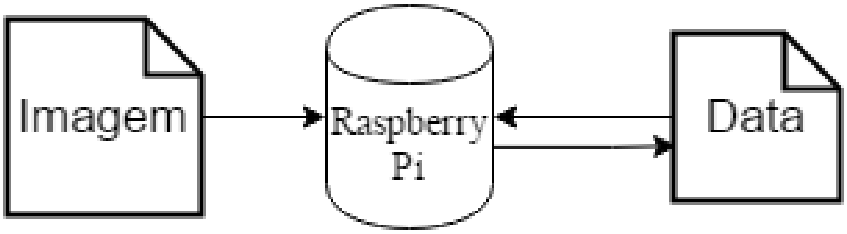

Figura 20: Desenho esquemático do treinamento de máquina.

Fonte: Elaborada pelo autor

Quando for executado, o nome do arquivo de imagem deve ser passado como parâmetro junto com o comando para execução. No início do programa, abre-se a imagem a ser testada (Figura 21) e é feita a conversão do padrão RGB para a HSV da mesma. Feita a conversão, executa-se uma função para fazer a segmentação da imagem, onde tudo que não faça parte do gramado é retirado, tal função retorna uma imagem (Figura 22) na qual todos os pixels que não estão dentro dos limites de cores HSV para cor de grama têm sua cor alterada para preto, enquanto os que estão dentro dos limites têm sua cor alterada para branco. Nota-se que permanecem alguns buracos ao longo da imagem, em uma região que deveria ser considerada como grama, isso é consertado usando uma função morfológica de fechamento, que elimina esses buracos. O resultado dessa função morfológica pode ser visto na Figura 23.

Figura 21: Imagem de teste original.

Fonte: Elaborada pelo autor

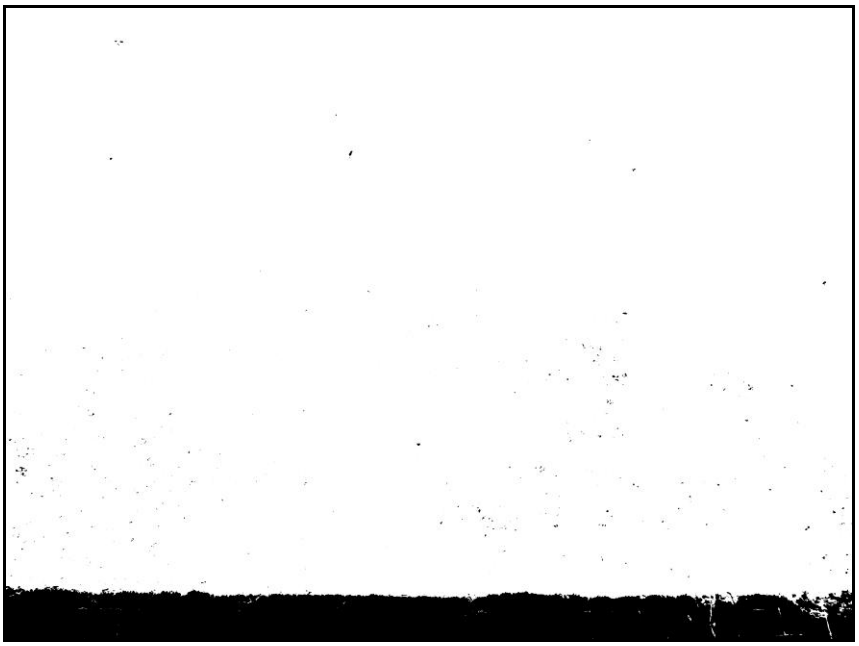

Figura 22: Imagem de teste após a função de segmentação.

Fonte: Elaborada pelo autor

[7] https://github.com/tiagotvt/HomeControl/tree/master/Tiago/treinamento

Figura 23: Imagem de teste após a função morfológica de fechamento.
Fonte: Elaborada pelo autor

Feita essa segmentação, compara-se um a um os pixels da imagem original (Figura 21) com os pixels da imagem segmentada (Figura 23), permanecendo apenas os pixels cuja cor na imagem que passou pela função morfológica seja branca. Dessa forma obtém-se uma imagem colorida que contém apenas o gramado (Figura 24).

Após obter essa nova imagem (Figura 24), é preciso manipulá-la uma última vez antes de realizar o treinamento. Para isso, faz-se a média das cores de todos os pixels dentro de uma área quadrada, com tamanho 100x100 pixels, e a cor de todos os pixels dentro dessa área é alterada para a cor média. Isso é feito ao longo de toda a imagem, tornando-a quadriculada. Locais onde o número de pixels pretos é acima de 40% do total de pixels, têm a cor média do quadrado alterada para preto. O resultado desse processamento pode ser observado na Figura 25. Por fim, faz-se a conversão da imagem para o padrão RGB e aplica-se um *grid* sobre a imagem (Figura 26), que a divide em vários blocos de quadrados, esses blocos serão as regiões analisadas pelo algoritmo de detecção de grama saudável.

Figura 24: Imagem de teste segmentada e colorida.

Fonte: Elaborada pelo autor

Figura 25: Imagem de teste quadriculada.

Fonte: Elaborada pelo autor

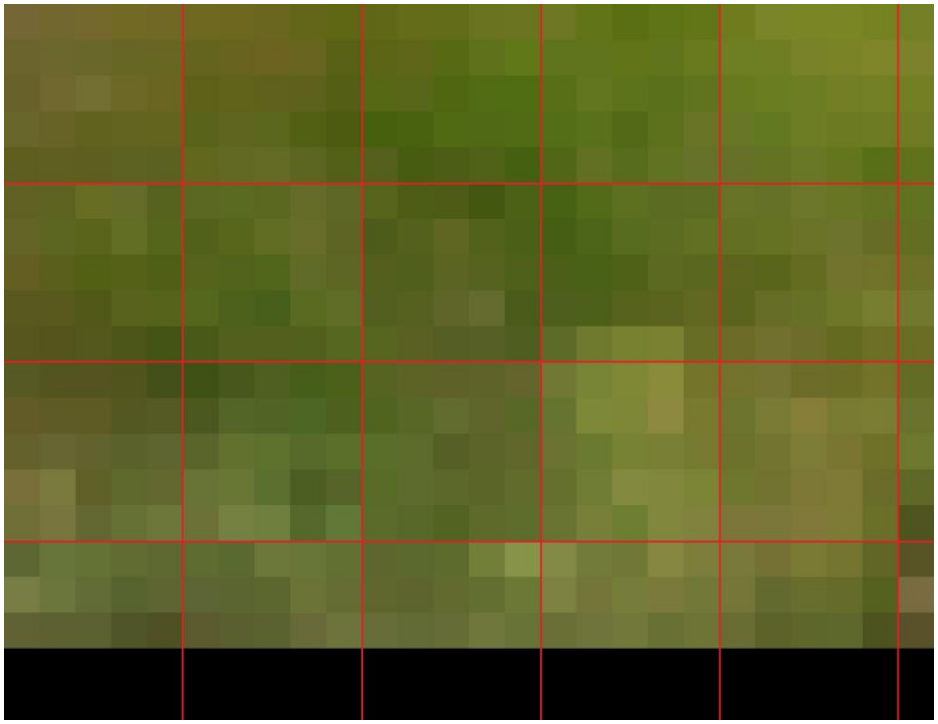

Figura 26: Imagem de teste com o *grid.*

Fonte: Elaborada pelo autor

Nessa parte do programa, é feita a análise de cada pequeno quadrado e os que não tiverem sua cor dentro dos limites mínimo e máximo, têm sua posição e sua região (bloco no qual fazem parte) inseridas em um vetor. No fim dessa análise, verifica-se quais blocos possuem pelo menos um quadrado que foi detectado como não saudável, e a posição desses blocos é guardada em um outro vetor. Logo em seguida, uma janela mostra a imagem com *grid* ao usuário, e o mesmo deve selecionar, com cliques sobre a imagem, as regiões que precisam ser irrigadas. Dessa forma é possível comparar, bloco a bloco, a resposta obtida pelo programa e a informada pelo usuário, e assim, verificar se os limites precisam ser alterados. Essa comparação pode obter quatro possíveis cenários, listados a seguir:

1. Se um bloco foi acusado pelo programa e também pelo usuário, nada acontece, pois o julgamento está correto.

2. Se um bloco não foi acusado pelo programa e nem pelo usuário, nada acontece, pois o julgamento está correto

3. Se um bloco foi acusado pelo programa, mas não pelo usuário, os limites devem ser alterados, pois o julgamento foi um falso positivo.

4. Se um bloco não foi acusado pelo programa, mas foi acusado pelo usuário, os limites devem ser alterados, pois o julgamento foi um falso negativo.

Ao se detectar um falso positivo, precisa-se aumentar a distância entre o limite mínimo e o máximo. Isso é feito da seguinte forma: procura-se no vetor os quadrados acusados de estarem fora dos limites, então aumenta-se o limite máximo (se necessário) e diminui-se o limite mínimo (se necessário) até que os valores RGB de cada quadrado daquela região esteja dentro do limite.

No caso da detecção de um falso negativo, precisa-se diminuir a distância entre o limite mínimo e o máximo. Isso é feito da seguinte forma: para o limite superior, calcula-se a média dos valores RGB de todos os blocos acusados de serem falso negativos e é selecionada a menor delas, então calcula-se a distância entre essa média e o limite máximo, por fim, o limite máximo é decrescido pela metade do valor dessa distância. No caso do limite inferior, também é calcuado a média de todos os blocos acusados, porém seleciona-se a maior média, e então calcula-se a distância entre essa média e o limite mínimo, por fim, adiciona-se metade do valor da distância ao limite mínimo.

Feita essa alteração, os novos valores dos limites são salvos em arquivo (com nome de data.txt) e a execução do programa é finalizada. Esse procedimento é repetido várias vezes e com imagens diferentes, a fim de se obter valores para os limites que funcionem com a maior variedade de imagens possível.

3.3.1.2 Software de processamento da imagem do gramado

O *software* para processamento da imagem do gramado foi desenvolvido em linguagem C++ e feito para ser executado via terminal. O programa tem como objetivo obter da internet os dados da previsão do tempo, salvá-las em arquivo, detectar quais regiões do gramado devem ser irrigadas, decidir qual *sprinkler* deve irrigar cada bloco e salvar em arquivo com os ângulos que os servos motores terão que girar. Uma

representação esquemática do *software* de processamento da imagem e o módulo Raspberry Pi pode ser observada na Figura 27.

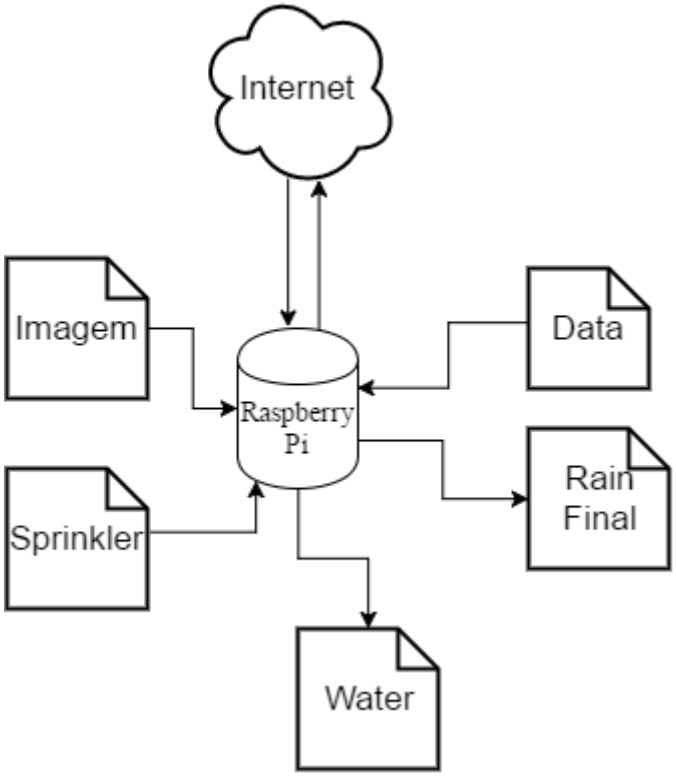

Figura 27: Desenho esquemático do processamento da imagem do gramado.

Fonte: Elaborada pelo autor

O funcionamento desse programa é muito parecido com o funcionamento do *software* de treinamento (seção 3.3.1.1), porém com algumas funcionalidades a mais. Ele também precisa que o nome do arquivo de imagem seja passado como parâmetro junto com o comando para execução. No início de sua execução, é enviado um comando para o sistema ("curl wttr.in/'Sao Carlos' > rain.txt") que acessa via internet um website[8] e obtém os dados da previsão do tempo do dia e os grava em um arquivo (rain.txt). A partir desse arquivo gerado, é possível obter apenas as informações que desejamos, no caso desse

[8] http://wttr.in/

programa são as probabilidades de chuva ao longo do dia, e então salvá-las em um outro arquivo (rainfinal.txt).

Logo em seguida, ocorre basicamente o mesmo procedimento que no programa da seção anterior. A imagem é aberta, modificada e o programa realiza os a análise das regiões. Entretanto, não é feita a checagem manual por um usuário, isso se deve pelo fato de que o sistema já passou por um treinamento e os valores dos limites de cor de grama saudável já estão bem determinados.

Após essa etapa, faz-se a seleção dos *sprinklers* a serem utilizados para irrigar cada região. As posições dos *sprinklers* são lidas de um arquivo (sprinkler.txt) que foi editado manualmente e então se faz os cálculos das distâncias entre o centro de cada bloco a ser irrigado e a posição dos *sprinklers*. Esse cálculo é facilmente feito utilizando a relação de Pitágoras, no qual se utiliza a diferença entre os valores das posições do eixo X de cada ponto como um dos catetos, e a diferença das posições do eixo Y de cada ponto como outro cateto, dessa forma pode-se encontrar a hipotenusa. Depois de calculado as distâncias, escolhe-se o *sprinkler* mais próximo de cada região para fazer a irrigação e determina-se o ângulo que cada um deve ser rotacionado para realizar essa tarefa. Assumindo que todos os *sprinklers* estão virados de frente para o gramado, a posição a sua esquerda é escolhida como referência para o 0 grau. Para descobrir os ângulos de rotação para cada *sprinkler*, faz-se algo semelhante ao que foi feito para o cálculo das distâncias descrito anteriormente: utiliza-se as diferenças das posições nos eixos X e Y para formarmos um triângulo e então se calcula o ângulo dentro desse triângulo usando arco tangente. Por fim, é calculado o ângulo formado entre à esquerda do *sprinkler* e a hipotenusa desse triângulo imaginário.

Feito isso, o programa salva em um arquivo os *sprinklers* selecionados e os ângulos de rotação de cada iteração do processo de irrigação e então o programa é finalizado. O fim do funcionamento do programa está representado na Figura 28, onde os pontos vermelhos simbolizam quadrados cujos valores RGB não estão dentro dos limites, os pontos amarelos simbolizam os *sprinklers* e as retas em amarelo são as direções que os *sprinklers* devem apontar para a irrigação de cada região.

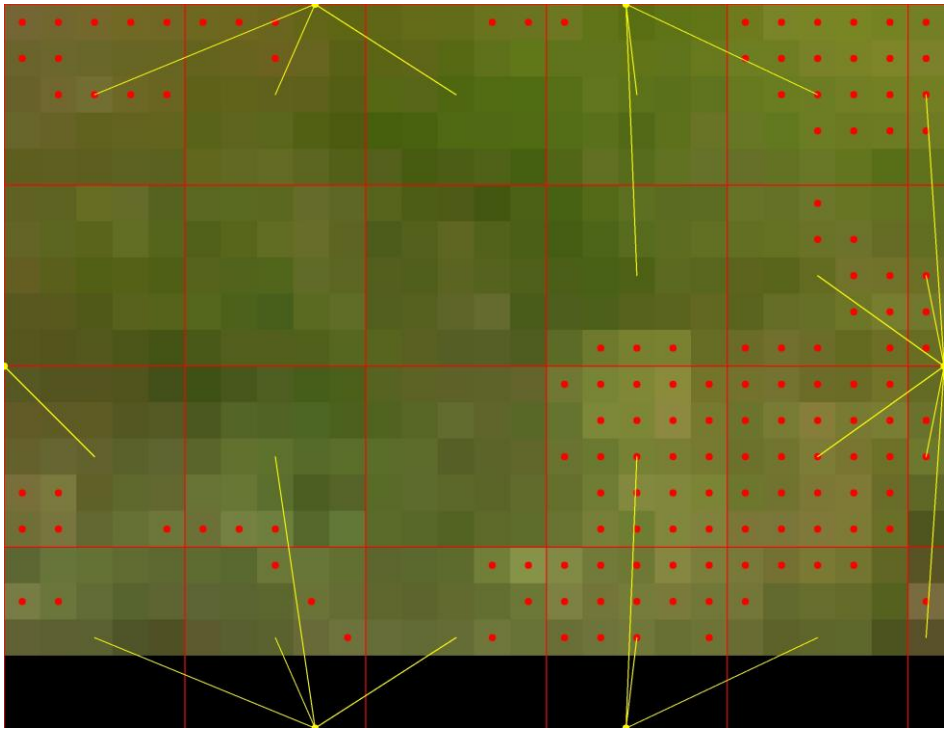

Figura 28: Imagem do resultado final do processamento da imagem do gramado.

Fonte: Elaborada pelo autor

3.3.1.3 Software de interface do usuário

O *software* de interface[9] do usuário foi desenvolvido em linguagem C e feito para ser executado via terminal. O programa tem como objetivo estabelecer a comunicação entre o módulo Raspberry e o módulo Arduino, por meio do *Bluetooth* HM-10. Uma representação esquemática do *software* de interface pode ser observada na Figura 29.

Ao ser inicializado, o programa envia um comando ao sistema ("sudo chmod a+rw /dev/ttyAMA0") para permitir a comunicação serial das portas RX e TX da Raspberry Pi. Feito isso, a interface exibe um menu para que o usuário escolha entre enviar um comando e finalizar o programa, como mostra a Figura 30. Se for escolhido enviar um comando, o arquivo que contém as probabilidades de chuva (rainfinal.txt) é aberto e analisado, caso a chance de chuva seja maior que 70%, o programa impede o envio de comandos para os

[9] https://github.com/tiagotvt/HomeControl/tree/master/Tiago/Interface

módulos Arduino, caso contrário, o comando será enviado. Após confirmado o envio do comando para os módulos periféricos, o arquivo que contém a numeração do *sprinkler* e o respectivo ângulo que ele deve movimentar (water.txt) é aberto e seus dados colocados em uma lista.

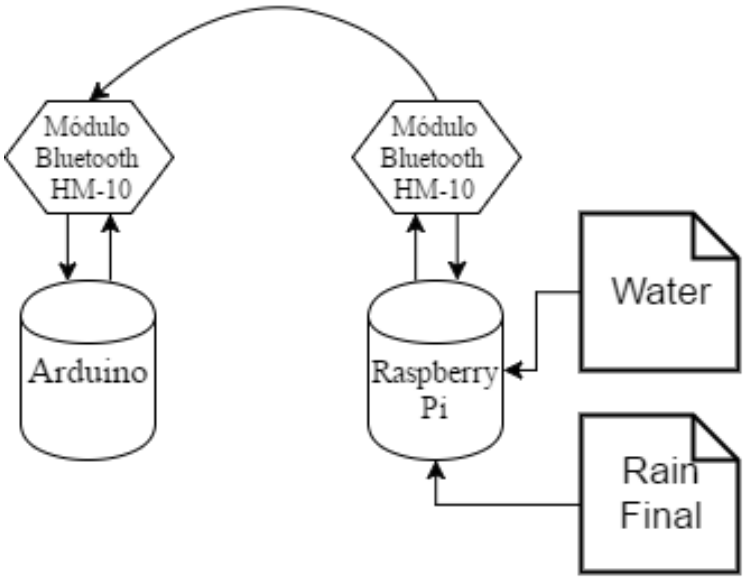

Figura 29: Desenho do esquemático da interface do usuário

Fonte: Elaborada pelo autor

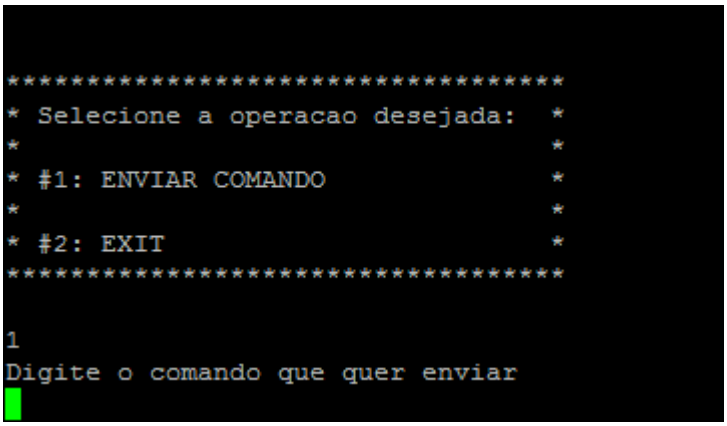

Figura 30: Menu da interface do usuário.

Fonte: Elaborada pelo autor

Neste momento, o programa entra em um laço para enviar todos os comandos. Caso a lista não esteja vazia, o programa envia um comando para o módulo *Bluetooth* pela porta

serial para se conectar ao outro módulo HM-10 desejado. Essa conexão é estabelecida da seguinte forma: envia-se pela porta serial os comandos “AT+ROLE1” e “AT+CON” mais o MAC do módulo desejado, esses são comandos do padrão *Bluetooth* 4.0 que tem como objetivo deixar o módulo HM-10 em modo de mestre e realizar a conexão com o aparelho que possuir o endereço MAC enviado, respectivamente. Após estabelecer a comunicação, envia-se o comando pela porta serial (o comando é uma *string* formada pela palavra “agua” mais o ângulo desejado, por exemplo: “agua90”, no qual o ângulo é de 90 graus) e o mesmo será enviado pelo módulo *Bluetooth* da Raspberry Pi e recebido pelo módulo HM-10 do Arduino. Feito o envio do comando, a posição na lista que continha o ângulo enviado na mensagem é apagada, espera-se o tempo necessário para a irrigação e uma nova iteração é iniciada. Esse processo de conexão e envio de comando é repetido até que a lista esteja vazia, determinando assim, o fim do comando de envio e o programa volta para o menu inicial.

3.3.2. Módulo Arduino

3.3.2.1 Software do módulo Arduino

O *software* do módulo Arduino[10] foi desenvolvido utilizando a linguagem própria para Arduino, na IDE de desenvolvimento do mesmo. O programa tem como objetivo estabelecer conexão entre o módulo Arduino e o módulo Raspberry Pi, além da leitura da umidade do solo, que será utilizada como último critério para a que aconteça a irrigação, movimentação do *sprinkler* e acionamento da válvula de entrada de água. Uma representação esquemática do *software* pode ser observada na Figura 31.

[10] https://github.com/tiagotvt/HomeControl/tree/master/Tiago/Arduino

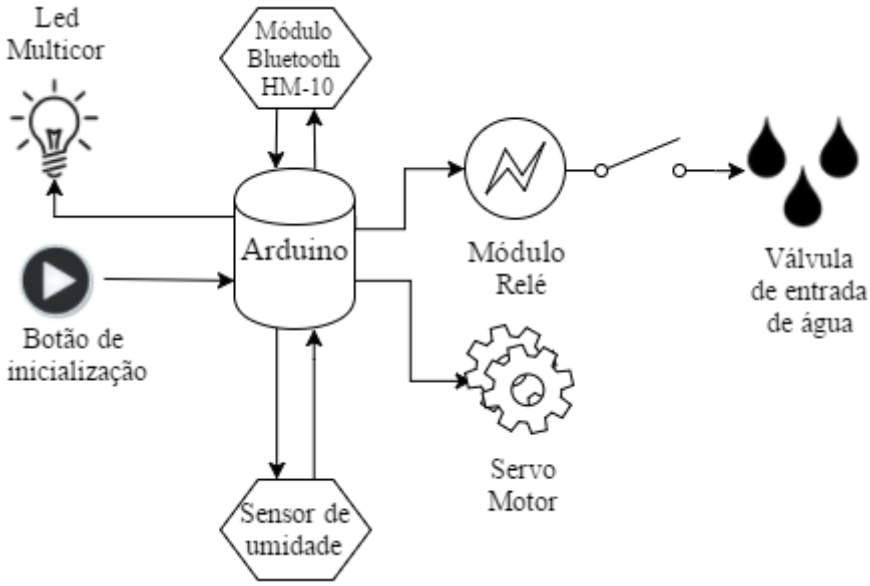

Figura 31: Desenho esquemático do módulo Arduino.

Fonte: Elaborada pelo autor

O programa é inicializado assim que a placa Arduino é ligada na fonte de alimentação. No início do programa, as variáveis são inicializadas, os pinos conectados ao relé (digital 3) e ao led multicor (digital 6, 10,11) são definidos como sendo de saída, e o pino conectado ao botão (digital 12) é definido como entrada. No caso dos pinos conectados ao sensor de umidade (analógico A0) e ao servo motor (digital 9), não foi preciso defini-los como sendo de entrada ou saída, isso se deve ao fato de que pinos analógicos não precisam ser inicializados para realizar leituras, e o servo motor utilizou o pino no modo PWM, que é iniciado pela biblioteca “PWMServo”. Ao fim do período de setup, as portas de comunicação serial são abertas, a frequência da comunicação é estabelecida e os comandos do padrão *Bluetooth* 4.0 “AT+ROLE0” e “AT+CON” mais o MAC do módulo *Bluetooth* central são enviados via porta serial para o módulo HM-10 do Arduino, dessa maneira o mesmo entra no estado escravo, para receber comandos, e se conecta ao módulo central.

Ao entrar na parte *loop* do programa, o *led* se torna vermelho e o sistema permanece em um estado de *stand-by*, realizando leituras do estado do botão, até que o mesmo seja apertado, e volta para ele quando o botão for apertado novamente. Isso foi feito com o intuito de simular uma ligação manual do sistema.

Estando no modo ativo, o programa altera a cor do *led* para verde e passa a verificar se algum dado foi captado pelo *Bluetooth* e está dentro da comunicação serial. Entretanto, muitas vezes existe lixo dentro dessa comunicação, por isso foi determinado no *software* de interface do usuário que a mensagem enviada seria “agua” mais o valor do ângulo (por exemplo: “agua120”), pois a cada *string* recebida pela porta serial, é verificado se é um comando válido ou apenas lixo.

Após a verificação da mensagem, extrai-se do comando o valor do ângulo e o coloca em uma variável. A seguir, são realizadas dez leituras do sensor de umidade e calculada a média das medidas, para garantir um valor mais preciso. Os valores obtidos pelo sensor variam de 476 a 1023, onde 476 representa o valor para máxima umidade (sensor imerso em água) e 1023 representa o valor para a mínima umidade (sensor completamente seco). Porém, para uma melhor representação, esses valores são mapeados, usando uma função própria do Arduino (*map* ()), para valores de 0 a 100 (onde 100 representa a máxima umidade e 0 a mínima), tornando o resultado da leitura do sensor de umidade muito mais intuitivo.

Caso o valor da leitura do sensor seja maior que 30, não se faz nada e o módulo volta a verificar a comunicação serial. Mas caso esse valor esteja abaixo de 30, o programa entra no estado de irrigação. Nesse estado, o *led* toma a cor azul, o servo motor é rotacionado para o ângulo recebido, usando o comando de escrita da biblioteca PWMServo, e após dois segundos, o valor do pino conectado ao relé (digital 3) é definido como *HIGH*, o que faz com que a tensão naquele pino passe a ser 5V e o módulo relé seja ativado. Dessa forma, a válvula de água passa a funcionar e bombeia a água para o *sprinkler* que inicia o processo de irrigação do gramado (Figura 32). Passado o período determinado para irrigação de cada região, o pino conectado ao relé é definido como *LOW*, o que faz com que a tensão no pino passe a ser 0V e o relé seja desativado, consequentemente a válvula de água também é desativada. Por fim, o *led* volta a ficar com a cor verde e retorna ao estado de verificação da porta serial.

Figura 32: Imagem do sistema de irrigação em funcionamento.

Fonte: Elaborada pelo autor

3.4. Resultados Obtidos

3.4.1. Treinamento de máquina e análise de imagem

Para que o sistema pudesse funcionar de forma autônoma, foi preciso realizar o treinamento do programa para se obter limites mínimo e máximo das cores RGB dos quadrados da imagem quadriculada.

Os testes com comparação manual foram feitos com um conjunto de amostra de 18 fotos do gramado, com tonalidades de cores e posições variadas. Os valores obtidos para os limites mínimo e máximo para cada cor se encontram na Tabela 6, lembrando que os valores RGB variam entre 0 e 255.

Tabela 6: Limites das cores RGB obtidos após o treinamento

Cor	Azul	Verde	Vermelho
Limite mínimo	13	80	60
Limite máximo	64	134	105

Para fazer uma validação dos limites encontrados, testou-se novamente com a amostra de fotos. O critério utilizado foi: cada imagem possui 24 blocos (como pode ser observado na Figura 26) e comparou-se a imagem final do programa com a imagem original para coletar a taxa de acerto em cada foto. Considerou-se um acerto quando um bloco que precisava ser irrigado foi apontado pelo programa como uma região seca, por mais que houvesse, naquele bloco, mais quadrados dentro dos limites. A partir da quantidade de acertos coletados em todas as fotos, calculou-se a média de acertos na detecção de blocos considerados secos.

Por fim, pode-se dizer que os valores limitantes obtidos são satisfatórios pois taxa de acerto, quanto à detecção de regiões com grama não saudável, foi de 93%.

3.4.2. Ambiente simulado

O ambiente simulado foi fundamental para a validação do sistema e identificação de pequenos erros que poderiam causar problemas nos testes em um ambiente real.

Os testes em ambiente simulado foram feitos a partir da mudança dos valores nos arquivos de dados e nas situações em que o sensor de umidade se encontrava. Dessa forma foi possível testar a execução de cada parte do sistema para todas as situações possíveis previstas no escopo do projeto.

Os resultados obtidos foram os esperados para o funcionamento normal do sistema: caso a probabilidade de chuva seja menor que 70%, envia-se o comando para o Arduino, e nenhuma ação é realizada caso contrário; se a umidade lida pelo sensor esteja acima de 30%, o Arduino não movimenta o servo motor, nem aciona o relé, caso contrário, o servo motor é rotacionado para a posição determinada pelo comando recebido e o relé é acionado; caso a lista de regiões secas esteja vazia, isso implica que ou não há nenhuma região que precise ser irrigada, ou todas os comandos já foram enviados para os módulos Arduino.

3.4.3. Ambiente real

Foram feitos testes em uma residência particular para a demonstração do funcionamento do sistema em um ambiente real, para a identificação de possíveis erros do projeto e para testar o bombeamento de água pelo *sprinkler*, que é a principal função do sistema.

Os testes realizados foram análogos aos feitos em um ambiente simulado e obteve-se resultados semelhantes no funcionamento do sistema. Entretanto, alguns outros resultados importantes foram observados: o *sprinkler* conseguiu ser movimentado corretamente pelo servo motor, a distância máxima entre os módulos sem que haja perda de comunicação entre os módulos *Bluetooth* é acima do esperado e a peça de encaixe, juntamente com o suporte do servo motor, suportaram muito bem à movimentação do *sprinkler* durante a irrigação. As Figuras 33, 34 e 35 mostram o protótipo do sistema durante os testes.

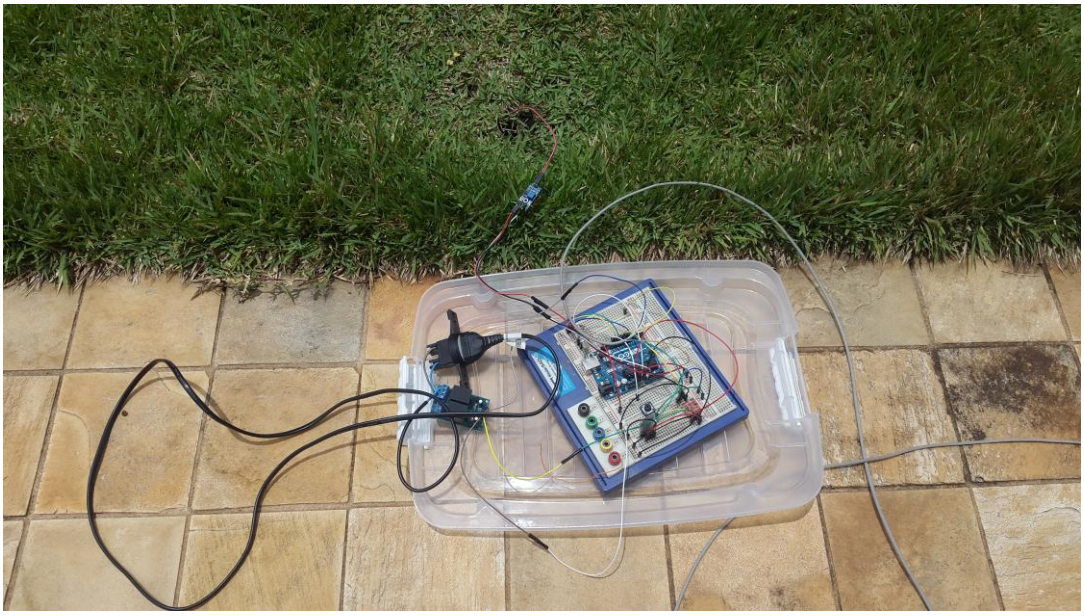

Figura 33: Módulo Arduino durante os testes.

Fonte: Elaborada pelo autor

Figura 34: Módulo Raspberry Pi durante os testes.

Fonte: Elaborada pelo autor

Figura 35: *Sprinkler* conectado à válvula de entrada de água.

Fonte: Elaborada pelo autor

3.5. Dificuldades e Limitações

As principais dificuldades encontradas durante o desenvolvimento deste trabalho foram com relação à implementação dos *softwares* de processamento de imagem e de comunicação *Bluetooth*. Isso se deve ao fato de que as ferramentas e funções disponíveis na OpenCV tiveram que ser estudadas para o uso correto no programa de detecção de regiões secas, e se trata de um assunto denso e embasado em muita teoria de processamento de imagem e visão computacional. Com relação às dificuldades com a comunicação *Bluetooth*, se deve ao fato de que para se usar essa tecnologia, deve-se utilizar corretamente o padrão *Bluetooth* 4.0, o qual também teve que ser estudado e possui várias peculiaridades.

Dois fatores observados que limitaram, de certa forma, o projeto foram: a análise de imagens muito claras ou muito escuras e a regulação da pressão da água bombeada pela válvula de água. Isso se deve pelo fato de que o valor RGB dos pixels de imagens com regiões muito claras (com exposição intensa ao sol) tendem ao branco, e em regiões muito escuras (com baixa iluminação) tendem ao preto, o que dificulta a segmentação de imagens que possuem esses efeitos. No caso do fator da pressão da água, o único meio de aumentar ou diminuir a pressão do jato d'água do *sprinkler* é regulando a vazão de água da torneira

que a mangueira está conectada, devido à válvula utilizada no projeto, que não consegue fazer controle de pressão.

3.6. Considerações Finais

A partir dos resultados obtidos, foi possível validar o sistema de irrigação inteligente pelo êxito em todos os critérios propostos neste projeto. Comparando os resultados com o que foi dito no Capítulo 1, seção 1.3, conseguiu-se desenvolver um sistema com um algoritmo para a análise da imagem do gramado e reconhecimento de regiões secas, um *software* que realize o gerenciamento e a comunicação entre as diferentes partes do sistema e que faça a tomada de decisão, além do design de uma peça de encaixe e de suporte feito em uma impressora 3D.

Além disso, manteve-se a vertente de *software* aberto, também proposto no escopo do projeto, pois tanto os códigos fonte quanto os arquivos dos objetos para impressão 3D utilizados no sistema estão disponíveis no GitHub.

CAPÍTULO 4: CONCLUSÃO

4.1. Considerações Iniciais

Neste capítulo comenta-se sobre o sistema proposto neste trabalho. Inicialmente, são descritas as principais contribuições obtidas com a realização deste projeto. Depois, comenta-se como o mesmo se relaciona com o curso de Engenharia de Computação. Por fim, são feitas algumas considerações sobre o curso supracitado e, finalmente, são apresentadas algumas propostas de trabalhos futuros.

4.2. Contribuições

O sistema desenvolvido neste projeto apresenta uma contribuição importante para a área da domótica. A arquitetura proposta, somada aos testes realizados compõem um material que pode ser usado como base para outros estudos, aplicações relacionadas e trabalhos futuros. Além disso, este trabalho pode me proporcionar um crescimento pessoal. Isso se deve pela oportunidade que ele me deu para trabalhar com algumas tecnologias que tenho muito interesse (Raspberry Pi e Arduino) e me fez perceber a aplicabilidade da domótica no dia-a-dia para economizar recursos naturais.

4.3. Relacionamento entre o Curso e o Projeto

A grade curricular do curso de Engenharia de Computação é bem ampla e aborda diversas áreas da Computação. Dessa forma, o aluno de Engenharia de Computação possui uma base muito sólida para poder atuar em diversas áreas.

Para esse projeto, utilizou-se conhecimentos obtidos em disciplinas como: programação orientada a objetos, sistemas digitais, sistemas embarcados, engenharia de software, visão computacional, circuitos elétricos e circuitos eletrônicos. Sem esse conhecimento não seria possível a realização deste trabalho.

4.4. Considerações sobre o Curso de Graduação

O curso de Engenharia de Computação da Universidade de São Paulo (USP), campus de São Carlos, oferece uma formação sólida, completa e diferenciada tanto para quem deseja atuar no mercado de trabalho como para aqueles que desejam atuar no meio acadêmico. O conhecimento obtido pelos alunos desse curso se transforma em projetos, soluções dentro de empresas, pesquisas de alto nível e criação de novas empresas.

Não obstante, apesar de todo o reconhecimento e qualidade, o curso de Engenharia de Computação desta instituição possui uma carga horária total extremamente alta se comparada com as outras engenharias do campus ou com cursos da mesma área em outras instituições de ensino. Algumas melhorias já foram propostas para tentar mudar essa situação, porém está longe do ideal. Isso se mostra um grande problema quando o aluno, que está no período ideal, fica tão sobrecarregado que precisa se dedicar exclusivamente às matérias da graduação, dificultando oportunidades de crescimento pessoal como iniciação científica e grupos de atividades extracurriculares.

4.5. Trabalhos Futuros

Como trabalhos futuros, podemos enumerar algumas melhorias do projeto atual:

1. Utilizar algum método HDR (*High Dynamic Range*) para combinar fotos de iluminações diferentes e tentar contornar a limitação das imagens muito claras ou muito escuras.

2. Adicionar ao sistema um *webcam* de resolução alta para tirar fotos do gramado automaticamente, o que tornaria o sistema ainda mais autônomo.

3. Substituir a válvula utilizada por uma que controle a pressão da água, desse modo seria possível controlar a distância do jato d'água do *sprinkler* via *software*.

4. Deixar o programa de análise de imagem ainda mais robusto, utilizando algum algoritmo de segmentação mais preciso.

5. Realizar um estudo sobre a economia de água proporcionada pelo uso do sistema de irrigação inteligente.

REFERÊNCIAS

ARDUINO. **What is Arduino?.** Disponível em: https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 11 set. 2016.

Bishop, C. M. **Neural Networks for Pattern Recognition.** Oxford: Oxford University Press. 1995.

BLUETOOTH SIG. **Tecnhical consideration.** Disponível em: <https://developer.bluetooth.org/TechnologyOverview/Pages/Topology.aspx>. Acesso em: 15 out. 2016

BRANDÃO, Roque. **A Domótica ao Serviço da Sociedade.** EUTRO À TERRA, Porto, Instituto Superior de Engenharia do Porto n. 1, Abril 2008 http://hdl.handle.net/10400.22/3670.

GONZALEZ, R.; WOODS, R. **Digital Image Processing.** 3. Ed. Edgar Blücher. 2000.

JNHUAMAO CORPORATION. **Bluetooth 4.0 BLE module Datasheet.** Disponível em: <http://fab.cba.mit.edu/classes/863.15/doc/tutorials/programming/bluetooth/bluetooth40_en.pdf >. Acesso em: 15 out. 2016.

KASTNER, Wolfgang; NEUGSCHWANDTNER, Georg; SOUCEK, Stefan; NEWMAN, Michael H. **Communication systems for building automation and control.** Proceedings of the IEEE invited paper, 2005.

OLIVEIRA, Nathan Partel Balduino. **Comunicação Sem Fio Entre Microcontroladores na Automação Residencial.** 68p. Trabalho de Conclusão de Curso (Graduação) – Curso de Engenharia Mecatrônica, Escola de Engenharia de São Carlos, USP, São Carlos, 2015.

RASPBERRY PI FOUNDATION. **What is a Raspberry Pi?.** Disponível em: <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>. Acesso em: 12 set. 2016.

AQUINO, Vicente. **A luta para combater as perdas de água**. Revista Saneas. v. 9, nº27, Set/Out 2007. p. 5-16

REINISCH, Christian; KASTNER, Wolfgang; NEUGSCHWANDTNER, Georg; GRANZER, Wolfgang. **Wireless technologies in home and building automation.** In proc. 5th IEEE International conference on industrial informatics, 2007.

SAS. **Machine Learning. O que é e por que é importante?.** Disponível em: <http://www.sas.com/pt_br/insights/analytics/machine-learning.html>. Acesso em 17 out 2016.

SIMÕES, Alexandre da Silva. **Segmentação de imagens por classificação de cores: uma abordagem neural.** 171 p. Tese (Mestrado). Escola Politécnica de São Paulo, São Paulo. 2000.

TONIDANDEL, F.. Domótica inteligente: automação residencial baseada em comportamento. Disponível em: http://fei.edu.br/~flaviot/pub_arquivos/WTDIA06.pdf; julho -2011. Acesso em: 06 set. 2016.