



UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

Sistema de Automação para Irrigação Doméstica

Edson Nascimento Silva Neto

Manaus - AM

Julho de 2021

Edson Nascimento Silva Neto

Sistema de Automação para Irrigação Doméstica

Monografia de Graduação apresentada à Faculdade de Tecnologia da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de bacharel em Engenharia da Computação.

Orientador(a)

Prof. Dr. Raimundo da Silva Barreto

Universidade Federal do Amazonas

Faculdade de Tecnologia

Manaus - AM

Julho de 2021

Monografia de Graduação sob o título *Sistema de Automação para Irrigação Doméstica* apresentada por Edson Nascimento Silva Neto e aceita pelo Instituto de Computação da Universidade Federal do Amazonas, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Titulação e nome do(a) orientador(a)

Orientador(a)

Departamento

Universidade

Titulação e nome do(a) membro da banca examinadora

Co-orientador(a), se houver

Departamento

Universidade

Titulação e nome do membro da banca examinadora

Departamento

Universidade

Titulação e nome do membro da banca examinadora

Departamento

Universidade

Manaus - AM, data de aprovação (por extenso).

À minha avó, familia, e amada Pauliina.

Agradecimentos

Agradeço a Deus por ser minha luz e caminho durante as muitas dificuldades. Agradeço a meus pais Eduardo e Sidineia pelo apoio e suporte. À minha avó, Auxiliadora, por acreditar em mim por nós dois e por não medir esforços para me ajudar em tudo que estivesse ao seu alcance. Agradeço à minha amada Pauliina, por ser compreensível, gentil e paciente. Ao Yan Matheus, por me impedir de usar um tema roxo em meu aplicativo e ser um incrível parceiro. Agradeço aos amigos e companheiros que colecionei durante esta trilha que foi o curso e que estiveram comigo em todos os momentos, emprestando seu apoio e me ajudando a crescer. Por fim, agradeço aos professores da UFAM em especial aos professores Raimundo Barreto e Edson Júnior pelo apoio técnico e orientação profissional.

Se não pode mudar as circunstâncias, mude sua atitude.

Viktor Frankl

Sistema de Automação para Irrigação Doméstica

Autor: Edson Nascimento Silva Neto

Orientador: Prof. Dr. Raimundo da Silva Barreto

Resumo

Hortas domésticas apresentam vários benefícios, com o cuidado correto, são capazes de fornecer alimentação livre de agrotóxicos e pesticidas, chás medicinais e temperos, porém requerem cuidados constantes, irrigação e proteção contra intempéries. No atual contexto de desenvolvimento de tecnologias *wireless* e internet das coisas se torna possível o gerenciamento remoto e automatizado de uma horta doméstica a partir de sensores e atuadores conectados à rede. O foco deste trabalho é a implementação de um sistema que realize tal gerenciamento, este é composto de um controlador, sensores de nível e motores, estrutura de rede e uma interface para acompanhamento e controle de irrigação pelo usuário. O objetivo do sistema é automatizar e proporcionar comodidade e customização de preferências a uma tarefa outrora manual. Como resultado, o sistema mostrou-se capaz de realizar irrigações precisas em horários e dias pré determinados pelo usuário.

Palavras-chave: Irrigação, Automatização, Internet das coisas, Interface.

Sistema de Automação para Irrigação Doméstica

Autor: Edson Nascimento Silva Neto

Orientador: Prof. Dr. Raimundo da Silva Barreto

Abstract

Home gardens have many benefits, with the right care, they are able to provide pesticide free food, medicinal teas and spices, but they require constant care, irrigation and weather protection. In the current context of developing wireless and IoT technologies, remote and automated management of a home garden is possible from sensors and actuators connected to the network. The focus of this work is the implementation of a system that performs such management, it consists of a controller, level sensors and motors, network structure and an interface for monitoring and irrigation control by the user. The purpose of the system is to automate and provide convenience and customization of preferences to a once manual task. As a result, the system proved to be able to perform accurate irrigations at schedules predetermined by the user.

Keywords: Irrigation, Automatization, Internet of Things, Interface.

Lista de figuras

Figura 1 – Padrão Publicante/Aassinante.	14
Figura 2 – Protocolo MQTT.	15
Figura 3 – Protocolo TLS.	19
Figura 4 – Diagrama em blocos do sistema de irrigação	21
Figura 5 – Esquema de comunicação.	23
Figura 6 – Certificado para acesso TLS.	28
Figura 7 – Pacote MQTT desprotegido.	29
Figura 8 – Pacote criptografado.	30
Figura 9 – Tanque de água.	31
Figura 10 – Sensoriamento do tanque de água e esquema de pull-up.	32
Figura 11 – Tela Inicial da Interface de Usuário.	33
Figura 12 – Tela Meus Planos.	35
Figura 13 – Tela de Configurações.	36
Figura 14 – Tela Adicionar Plano.	37
Figura 15 – Fluxograma de configuração do microcontrolador.	38
Figura 16 – Visão de usuário da Interface de Configuração.	39
Figura 17 – Fluxograma do Módulo de Gerenciamento.	41
Figura 18 – Mensagens de status após conexão com o broker.	43
Figura 19 – Mensagens de status do sensor de umidade.	43
Figura 20 – Mensagens de usuário do sistema.	44
Figura 21 – Logs de Inicialização do Módulo de Gerenciamento.	45
Figura 22 – Logs de recebimento de mensagens retidas.	46
Figura 23 – Logs de recebimento de mudança de nível de água.	46
Figura 24 – Protótipo do Sistema de Irrigação.	47

Sumário

1	INTRODUÇÃO	11
1.0.1	Objetivo Geral	12
1.0.2	Objetivos Específicos	12
2	FUNDAMENTAÇÃO TEÓRICA	13
2.1	Padrão <i>Publish/Subscribe</i>	13
2.1.1	Funcionamento do Padrão <i>Publish/Subscribe</i>	13
2.2	Protocolo de Comunicação MQTT	14
2.2.1	Características do MQTT	14
2.2.2	Qualidade de Serviço	15
2.2.3	Mensagens Retidas	17
2.2.4	Último Desejo e Testamento	17
2.3	<i>Transport Layer Security</i>	18
3	METODOLOGIA	20
3.1	Introdução	20
3.2	Escopo do Sistema	20
3.3	Comunicação	23
3.3.1	Tópicos do Sistema	24
3.3.2	Formatação de Mensagens	25
3.3.3	Criptografia TLS	27
3.4	Tanque de água	30
3.5	Interface de Usuário	32
3.5.1	Tela Inicial	33
3.5.2	Tela Meus Planos	34
3.5.3	Tela de Configurações	35
3.5.4	Tela Adicionar Plano	36
3.6	Interface de Configuração	37

3.7	Módulo de Gerenciamento de Irrigação	39
4	TESTES REALIZADOS	42
4.1	Objetivos	42
4.2	Mensagens de status retidas	42
4.3	Mensagens contínuas e mensagens de usuário	43
4.4	Testes do Módulo de Gerenciamento	44
5	CONCLUSÕES	48
5.1	Propostas para Trabalhos Futuros	49
	Referências	50

1 Introdução

Irrigação é definida como a aplicação artifical de agua sobre terra ou solo. O processo de irrigação pode ser usado para o cultivo de plantas durante um período de chuvas inadequadas ou paisagismo e manutenção de jardins e hortas[1].

A manutenção de hortas domésticas não é apenas uma atividade que promove sustentabilidade, mas também proporciona alívio para estresse e uma fonte alternativa de alimentação, economizando no orçamento familiar com vegetais e legumes livres de agrotóxicos[2].

Porém o cuidado de hortas requer atenção e tempo, além de conhecimentos em irrigação e agricultura visto que plantações inteiras pode ser perdidas devido ao excesso de água fornecida às plantas[2]. Neste sentido, um sistema de irrigação automático desempenha um papel importante, impactando de forma positiva no cultivo da horta. Uma vez instalada no campo de cultivo a distribuição automatizada de água à culturas se torna fácil e não requer apoio humano para operar permanentemente.

Este trabalho busca aliar a praticidade da irrigação automática à elementos de Internet das Coisas para proporcionar funcionalidades como feedback e customização ao usuário para que este possa cuidar de sua horta doméstica mesmo não estando próximo a ela. Para este fim, desenvolvemos um sistema de irrigação microcontrolado e equipado com sensores e atuadores para monitorar culturas e controlar o cuidado da horta.

O sistema desenvolvido é composto por três partes: o hardware, compreendido pelo conjunto de microncontrolador, sensores, atuadores, tanque de água e outros equipamentos tradicionais de irrigação como mangueiras e conectores. A interface de interação com o usuário, compreendida por um aplicativo em plataforma android para visualização de dados e modificação de variáveis de sistema e um software de roteamento de mensagens entre os

elementos do sistema que opera usando protocolo de comunicação MQTT.

A comunicação entre as partes do sistema se dá sobre o MQTT, um protocolo de rede leve e flexível otimizado para redes TCP/IP[3] cujo esquema de funcionamento está apoiado no padrão *publisher/subscriber* ou publicante/assinante. Durante a comunicação entre os elementos, os dados são criptografados utilizando a tecnologia TLS (*Transport Layer Security*), uma versão atualizada e mais segura do SSL (*Secure Sockets Layer*).

1.0.1 Objetivo Geral

Desenvolver um sistema de irrigação microcontrolado com controles customizáveis e interface de usuário para emissão de comandos e visualização de dados.

1.0.2 Objetivos Específicos

- Realizar irrigação de plantas por meio de atuadores conectados a um sistema embarcado.
- Medir níveis de água, temperatura e umidade do solo por meio de sensores conectados ao sistema embarcado.
- Tornar os dados extraídos de sensores disponíveis ao usuário em tempo real por meio de uma estrutura de comunicação web e interface em plataforma android.
- Permitir ao usuário customizar a frequência e volume da irrigação por meio da interface.

2 Fundamentação Teórica

2.1 Padrão *Publish/Subscribe*

Publish-subscribe (pub/sub ou Publicante/Assinante em tradução livre) é um padrão de mensagens onde publicantes mandam mensagens para assinantes. Em arquitetura de software, mensagens nesse padrão fornecem notificações instantâneas de eventos para aplicações distribuídas, especialmente aquelas formadas por pequenos blocos independentes[4].

2.1.1 Funcionamento do Padrão *Publish/Subscribe*

Os três elementos centrais para entender o padrão pub/sub são:

- Publicante ou *Publisher*: Dispositivo ou software que publica mensagens na infraestrutura de comunicação.
- Assinante ou *Subscriber*: Assina uma categoria de mensagens.
- Infraestrutura de Comunicação: Recebe mensagens de publicantes e mantém as assinaturas dos assinantes.

O publicante categoriza mensagens publicadas em classes através das quais os assinantes receberão as mensagens. Basicamente, o publicante tem um canal de entrada de mensagens que se divide em múltiplos canais de saída, um para cada assinante. Assinante podem expressar interesse em uma ou mais classes e somente receber mensagens de seu interesse[4].

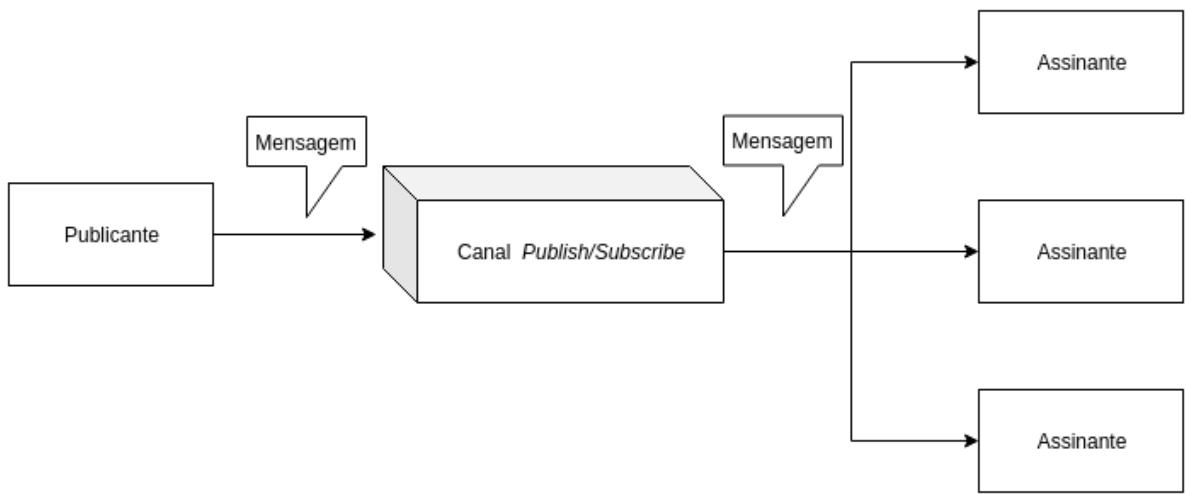


Figura 1 – Padrão Publicante/Assinante.

Fonte: Adaptado de [4].

2.2 Protocolo de Comunicação MQTT

O MQTT (*Message Queue Telemetry Transport*) é um protocolo de comunicação que utiliza o padrão *publish/subscribe*, simples e leve, o MQTT foi desenvolvido para dispositivos restritos em termos de memória e banda operando em redes não confiáveis e com alta latência[5].

2.2.1 Características do MQTT

O protocolo MQTT define dois tipos de entidades na rede: um *message broker* e inúmeros clientes. O *broker* é um servidor que recebe todas as mensagens para os clientes de destino relevantes. Um cliente é qualquer dispositivo ou software que possa interagir com o broker e receber mensagens. Um cliente pode ser um sensor de IoT em campo ou um aplicativo em um *data center*[3].

O funcionamento básico em uma interação com o broker se dá da seguinte forma:

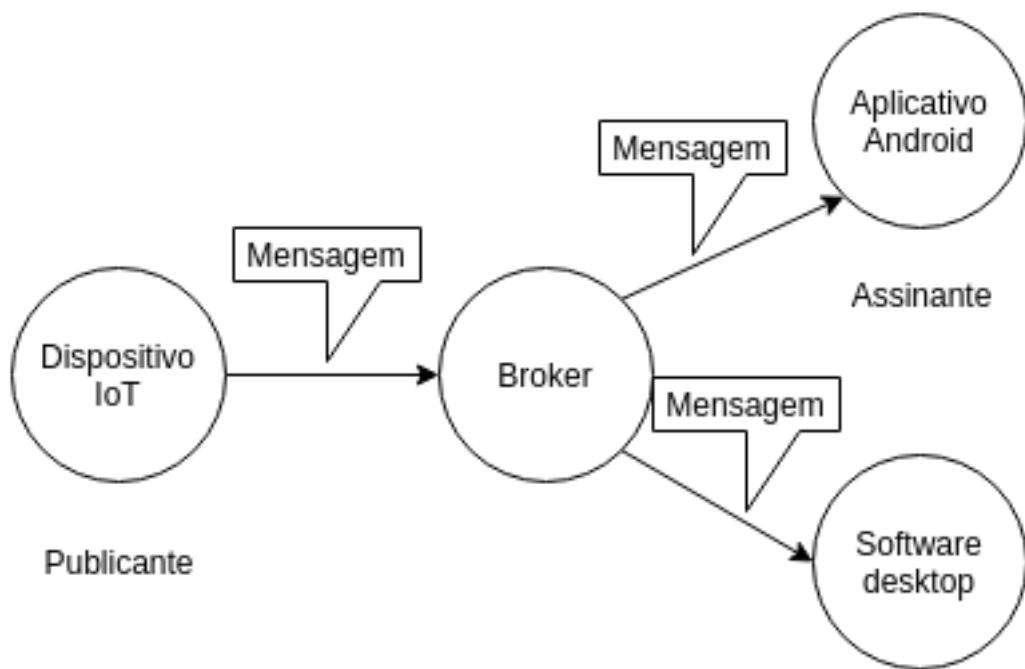


Figura 2 – Protocolo MQTT.

Fonte: Adaptado de [3].

- O cliente conecta-se ao broker. Ele pode assinar qualquer "tópico" ou classe de mensagens no broker. Essa conexão pode ser uma conexão TCP/IP simples ou uma conexão TLS criptografada para mensagens sensíveis.
- O cliente publica as mensagens em um tópico, enviando a mensagem e o tópico ao broker.
- Em seguida, o broker encaminha a mensagem a todos os clientes que assinam esse tópico.

2.2.2 Qualidade de Serviço

O nível de Qualidade de Serviço (termo em inglês *Quality of Service*, normalmente abreviado para QoS) é um acordo entre remetente e destinatário de uma mensagem e define a garantia de entrega para uma mensagem específica

(cada mensagem deve especificar o nível de QoS). Há 3 níveis de QoS no MQTT [6]:

- Nível 0 - **No máximo uma vez** (*At most once*). O nível mínimo de serviço de entrega, também chamado de "*best-effort delivery*" ou "*fire and forget*", descreve um nível de serviço em que não há garantias de entrega, o receptor não reconhece o recebimento da mensagem e esta não é guardada para reenvio futuro pelo remetente.
- Nível 1 - **Pelo menos uma vez** (*at least once*). Este nível garante que a mensagem será entregue pelo menos uma vez ao destinatário. Após enviar uma mensagem PUBLISH (publicada) o remetente a guarda até que receba um pacote PUBACK do destinatário, que reconhece o recebimento da mensagem. A mensagem pode ser reenviada múltiplas vezes até que haja confirmação.
- Nível 2 - **Exatamente uma vez** (*exactly once*). O Maior nível de serviço no MQTT. Este nível garante que cada mensagem é recebida somente uma vez por todos os destinatários. Esta garantia é fornecida por pelo menos dois fluxos de requisição/resposta entre remetente e destinatário.

Quando o destinatário recebe uma pacote PUBLISH com QoS 2, responde com um pacote PUBREC que reconhece o pacote PUBLISH. Sem este reconhecimento o remetente envia o pacote PUBLISH novamente até que o reconhecimento aconteça. Uma vez que o remetente recebe o reconhecimento PUBREC do destinatário, este descarta o pacote PUBLISH inicial com segurança e envia um pacote PUBREL ao destinatário. Após receber o pacote PUBREL, o destinatário então descarta todos os estados armazenados e responde com um pacote PUBCOMP. Até que o remetente receba o pacote PUBCOMP, este mantém uma referência ao pacote PUBLISH original. Este passo é importante para impedir que a mensagem seja processada uma segunda vez. O fluxo do pacote com QoS 2 é finalizado quando ambos os lados da transição tem certeza de que a mensagem

foi recebida com sucesso e o remetente tem confirmação de entrega. Por causa deste processo, o QoS 2 é o mais seguro e mais lento nível de serviço que o protocolo pode fornecer.

2.2.3 Mensagens Retidas

No MQTT, um cliente que assina tópicos no broker não tem garantias de quando um cliente publicante irá publicar uma mensagem em algum tópico de seu interesse, pode levar segundos, minutos, horas ou dias para que um publicante envie uma mensagem. Até que a próxima mensagem seja publicada, o cliente assinante não tem idéia acerca do status do tópico. Para esta situação existem as mensagens retidas[7].

Uma mensagem retida é uma mensagem MQTT normal com a flag *retained* habilitada. O broker guarda a última mensagem retida e o QoS para este tópico. Cada cliente que assinar o tópico desta mensagem a receberá imediatamente após assinatura. O broker guarda apenas uma mensagem retida por tópico.

2.2.4 Último Desejo e Testamento

O protocolo MQTT é geralmente usado em situações onde a conexão com a internet não é confiável. É fácil pensar que nesses cenários alguns clientes podem ocasionalmente se desconectar do broker de forma abrupta, algo que pode acontecer devido a perda de conexão, baterias fracas ou outras razões. Em uma situação de desconexão normal e voluntária do cliente, uma mensagem MQTT DISCONNECT é enviada ao broker. Porém, em uma desconexão abrupta, o protocolo MQTT define uma funcionalidade chamada *Last Will and Testament*[8] (Último Desejo e Testamento).

Esta funcionalidade é definida como uma função para notificar clientes acerca de uma desconexão abrupta de cliente. Cada cliente pode especificar sua mensagem de último desejo quando conecta com o broker. Esta mensagem é um mensagem MQTT normal que é guardada pelo broker e enviada a todos os clientes escritos no tópico caso o cliente seja desconectado de forma abrupta. Caso o cliente envie um pacote MQTT DISCONNECT antes de sair, a mensagem de último desejo é descartada.

2.3 *Transport Layer Security*

Para adicionar uma camada de segurança ao sistema de irrigação e impedir ataques do tipo *man-in-the-middle*, onde terceiros podem observar a conexão entre cliente e servidor e possivelmente adquirir dados pessoais dos envolvidos, utilizamos o protocolo TLS para criptografar os dados trocados entre os elementos do sistema.

Simplificadamente, TLS e SSL (*Secure Sockets Layer*) são protocolos criptográficos que utilizam um mecanismo de *handshake* para negociar vários parâmetros e criar uma conexão segura entre cliente e servidor[9]. Após o *handshake* ser concluído, uma comunicação criptografada é estabelecida entre cliente e servidor, e terceiros não podem observar nenhuma parte desta comunicação.

O TLS é a versão mais atual e padronizada do protocolo SSL, originalmente desenvolvido pela Netscape para fornecer segurança ao *e-commerce* na Web, criptografando os dados de cliente e assegurando uma transação segura. Assim como o MQTT, SSL foi implementado sobre o TCP, como ilustrado na figura 3, habilitando assim que protocolos em camadas superiores (HTTP, email e outros) operassem sem mudanças enquanto fornece segurança para comunicações na rede[10].

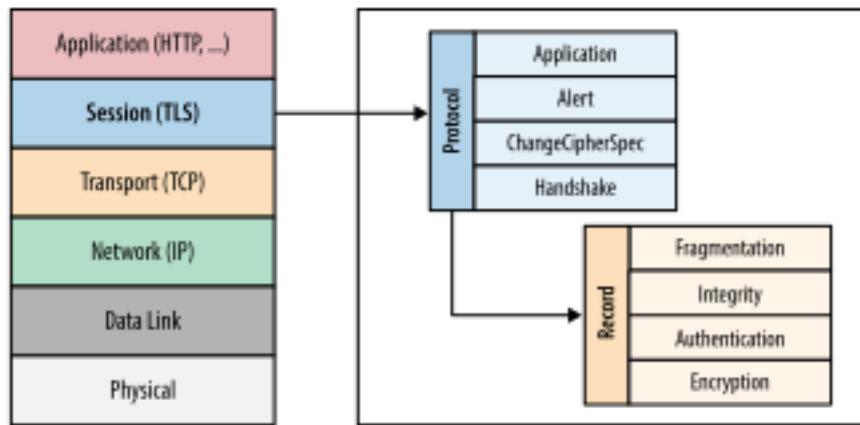


Figura 3 – Protocolo TLS.

Fonte: *O'Reilly: High Performance Browser Networking* [7].

Quando o TLS/SSL é usado corretamente, observadores externos podem apenas inferir os terminais de comunicação, tipo de criptografia além de frequência e quantidade aproximada de dados enviados, mas não podem ler ou modificar nada nos pacotes em si.

3 Metodologia

3.1 Introdução

O sistema de irrigação proposto foi desenvolvidos em blocos, alguns mais simples, outros mais complexos. A primeira seção deste capítulo, intitulada *Escopo do Sistema* trata da visão de alto nível do sistema de irrigação, os diferentes blocos que o compõe e como estes interagem entre si. A seção seguinte descreve o bloco MQTT e os detalhes da comunicação, bem como a criptografia aplicada a ela. Em seguida, na seção *Tanque de Água*, o bloco de Sensores de Nível de água é descrito. As seções seguintes são relacionadas ao bloco de *Interface de Usuário*, *Interface de Configuração* e *Módulo de Gerenciamento de irrigação*, esta última detalhando o funcionamento do sistema embarcado que realiza o agendamento e regagem das plantas bem como os blocos mais simples de *controle do motor de bombeamento* e *sensor de umidade do solo*.

3.2 Escopo do Sistema

A figura 4 mostra o esquema em blocos do sistema de irrigação, as setas indicando a direção do fluxo de dados, o sistema ilustrado realiza irrigação de plantas e monitoramento de níveis de água e umidade do solo, informação que é enviada a outros dispositivos da rede que estiverem conectados ao sistema através do *broker*.

No bloco *Interface de usuário*, uma pessoa utilizando a aplicação android desenvolvida para este sistema pode verificar os níveis de umidade do solo e quantidade de água, depurados a partir dos dados analógicos enviados pelo módulo de gerenciamento. O usuário mandar um comando de irrigação imediata e também modificar as opções de irrigação, configurando por exemplo,

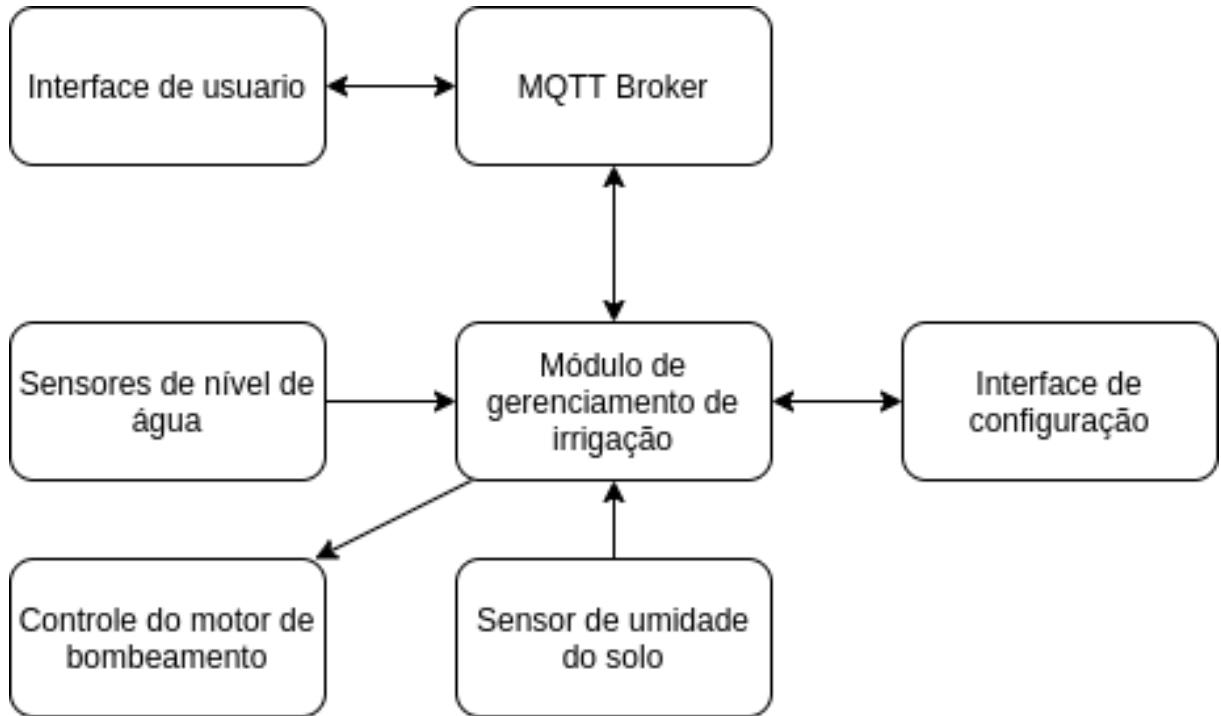


Figura 4 – Diagrama em blocos do sistema de irrigação

um perfil que irrigue as plantas apenas aos finais de semana, durante as horas de maior incidência do sol e um outro perfil que realize a irrigação somente quando o solo estiver seco. Ambos estes perfis podem ser guardados e alternados pelo usuário.

O bloco *MQTT Broker* é parte do arcabouço de comunicação MQTT, recebe todas as mensagens indo e vindo dos diferentes componentes, aplica filtros, determinando quem é assinante de qual mensagem, e as envia a estes clientes escritos. O broker também possui informações sobre o estado de conexão de todos os componentes(clientes) conectados, e especificamente no caso do módulo de gerenciamento de irrigação, envia um aviso aos clientes quando este módulo é abruptamente desconectado, seja por motivos de conexão ou falta de energia.

O bloco *Módulo de gerenciamento de irrigação* realiza a verificação dos sensores de nível de água e umidade do solo, agrupa estes dados e os envia ao broker para reenvio aos clientes conectados. Este bloco também é responsável por receber e executar os sinais de controle enviados através do broker pelo

bloco de interface do usuário. A partir do detalhamento da quantidade de água a ser utilizada, o módulo de gerenciamento regula o tempo em que o motor deve permanecer ligado bombeando água para as plantas.

Este bloco também possui internamente um módulo WiFi para conexão com a internet e, caso haja alguma mudança na situação da conexão externa (uma mudança na senha do roteador, por exemplo), o módulo de gerenciamento para a ativar a interface de configuração.

O bloco *Interface de Configuração* é ativado pelo Módulo de Gerenciamento quando o status da conexão muda e é necessário configurar uma nova rede WiFi. Para este fim, o módulo wifi do bloco Módulo de Gerenciamento muda seu papel na rede de cliente (estação ou STA) para servidor (ponto de acesso ou AP), e passa a aceitar conexões de clientes e servir localmente uma interface web onde o usuário pode introduzir os dados de autenticação de uma nova rede à qual o módulo deve se conectar.

O bloco *Sensores de nível de água* comprehende três sensores do tipo chave que se fecham em estado baixo, ou seja, enquanto o nível de água está abaixo do sensor em questão, o Módulo de Gerenciamento está recebendo um sinal alto ou *high*, e quando a agua passa pelo sensor e ele se fecha, o Módulo de Gerenciamento recebe um sinal baixo ou *low*.

O bloco *Sensor de umidade do solo* realiza o sensoriamento da umidade do solo e retorna uma valor natural inversamente proporcional à umidade verificada, ou seja, quando mais úmido o solo, menor o valor capturado pelo sensor (por volta de 100 dentro da água e por volta de 4000 em ambiente seco).

Por fim, o bloco *Controle do motor de bombeamento* recebe um sinal de 3.3 volts do Módulo de Gerenciamento e fecha um circuito relé entre uma bateria de 9V e o motor de bombeamento, que realiza a distribuição de água para as plantas através de uma mangueira provida de cabeças de irrigação, este circuito relé se mantém fechado por um tempo determinado pelo usuário através da interface de usuário e controlado pelo Módulo de Gerenciamento.

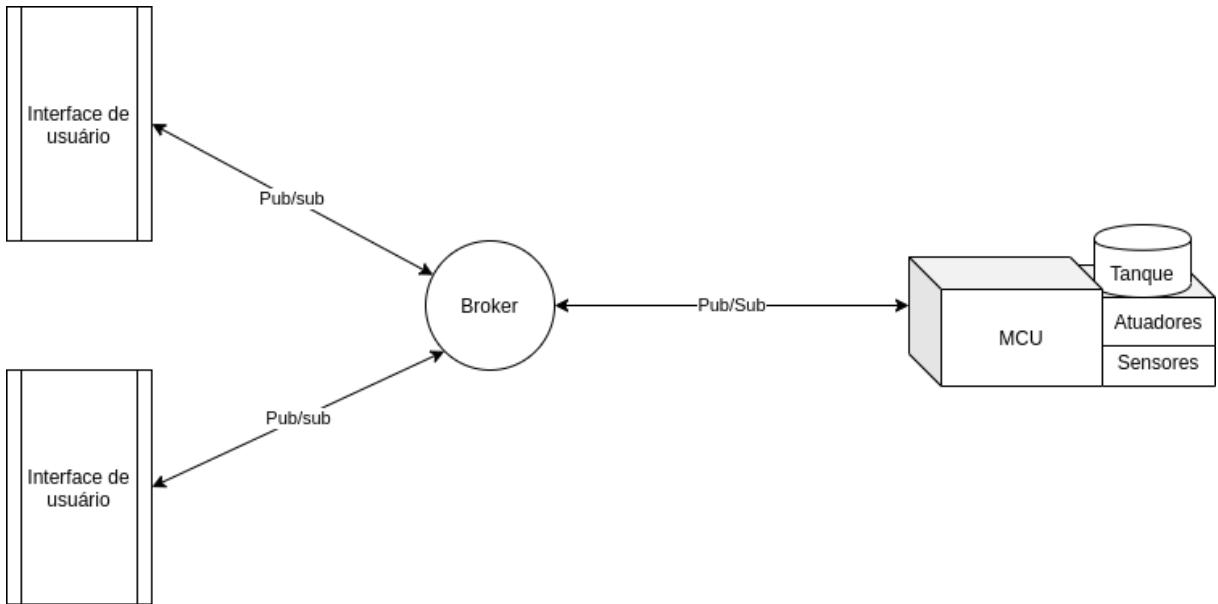


Figura 5 – Esquema de comunicação.

3.3 Comunicação

Uma das partes principais do sistema de irrigação é o esquema de comunicação entre os componentes. Estes podem ser divididos em três blocos que podem ser visualizados na figura 5:

- *Interface de usuário* - Aplicação através da qual o usuário pode selecionar e modificar características do sistema, assim como gerenciar aspectos da irrigação e visualizar o *status* dos componentes e plantas. Manda comandos ao sistema embarcado e recebe informações de sensoriamento e consumo.
- *Sistema Embarcado* - Conjunto dos componentes eletrônicos, CIs, sensores e atuadores que realizam ações de irrigação e sensoriamento nas plantas. Recebe comandos da aplicação de usuário e manda informações de sensoriamento e consumo para a aplicação e servidor.
- *MQTT Broker* - Componente do protocolo MQTT que gerencia e redireciona as mensagens entre as diferentes partes do sistema.

3.3.1 Tópicos do Sistema

Como já explicado anteriormente, a forma de envio e recebimento de mensagens do protocolo MQTT é através de tópicos, sob os quais clientes podem receber e realizar publicações. O sistema de irrigação possui três tipos de mensagens trocadas entre os componentes:

- Controle - Mensagens partindo da Interface de Usuário para o Módulo de Gerenciamento do sistema com instruções para irrigação imediata ou troca de modos de irrigação que utilizam os seguintes tópicos:
 - *system/control/dispenseWater* - Tópico de regagem imediata, mensagens enviadas sempre que o usuário solicita uma irrigação imediata.
 - *system/control/setupProgram* - Tópico para troca de modos de irrigação, mensagens enviadas sempre que há mudança no programa de irrigação pelo usuário.
- Sensores - Mensagens partindo do Módulo de Gerenciamento para a Interface de Usuário com relatórios sobre sensores de nível e umidade do solo.
 - *system/sensor/soil* - Tópico para mensagens relacionadas à umidade do solo, mensagens enviadas a cada meia hora se o modo de irrigação automática estiver habilitado.
 - *system/sensor/level* - Tópico para informações do sensor de nível de água, mensagens enviadas sempre que há mudança no nível da água no tanque.
- Conectividade - Mensagem enviada do Módulo de Gerenciamento para a Interface de Usuário quando este se conecta ao broker. Alternativamente, o broker utiliza este tópico para enviar uma mensagem automática à Interface de Usuário quando detectar que o sistema embarcado se desconectou de forma abrupta, isto é a funcionalidade *Last-Will* do protocolo MQTT.

- *system/connected*

3.3.2 Formatação de Mensagens

Apenas um dos tópicos utilizados pelo Sistema de Irrigação recebe e envia mensagens estáticas: o tópico de controle *system/control/dispenseWater* não requer uma mensagem específica, pois sua utilização não tem outro papel além de iniciar o evento de irrigação pelo Módulo de Gerenciamento. Mensagens enviadas através de outros tópicos tem os seguintes formatos e descrições:

- Troca de Modo de Irrigação - Mensagem enviada da Interface de Usuário para o Módulo de Gerenciamento e descreve os detalhes do programa de irrigação que o sistema deve seguir:

```
{ "amountWater":1021, "gmtTimezone":-14400, "deadlineHour":18,  
"deadlineMinute":57, "deadlineSecond":0, "deadlineDays":[1, 2,  
3, 4, 5, 6, 0], "automaticWatering":false}
```

- **amountWater** - Um número que determina o tempo em que o motor se manterá ligado e bombeando água para as plantas. Tipo Integer com variação entre 0 e 10000.
- **gmtTimezone** - Fuso horário para contagem do tempo no formato GMT (*Greenwich Mean Time*), importante para o agendamento de irrigações. Tipo Integer fixo em -14400, número utilizado pela biblioteca de temporização para representar o fuso horário de Manaus.
- **deadlineHour** - Número que define a informação de hora do horário agendado para irrigação, Tipo Integer com variação entre 0 e 23.
- **deadlineMinute** - Número que define a informação de minutos do horário agendado para irrigação, Tipo Integer com variação entre 0 e 60.

- **deadlineSecond** - Número que define a informação de segundos do horário agendado para irrigação, Tipo Integer com variação entre 0 e 60.
- **deadlineDays** - Vetor de dias em que as irrigações acontecerão, o vetor tem um tamanho máximo de 7, significando que o sistema pode irrigar as plantas sete vezes na semana. Cada valor no vetor indica um dia de irrigação e pode variar de -1 a 6, com -1 sendo um valor para preencher espaços vazios no vetor (caso o usuário selecione um número de dias menor que 7) e os valores de 0 a 6 representando os dias de Domingo a Sexta.
- **automaticWatering** - Valor booleano que indica se o sistema deve usar a umidade do solo como parâmetro de irrigação (irrigar apenas quando o solo estiver seco) ao invés de agendar a irrigação. Caso este campo seja *true*, as variáveis de tempo e dia são desconsideradas pelo Módulo de Gerenciamento. Tipo Booleano com variação entre *true* e *false*.

- Mensagens de sensores - Mensagens enviadas do Módulo de Gerenciamento para a Interface de Usuário e contém as informações fornecidas pelos sensores e algumas variáveis globais do sistema:

```
{ "type": "moistureSensor", "lastWatered": 18, "Dryness": 1072 }
```

```
{ "type": "levelSensor", "lastWatered": 18, "waterLevel": 50 }
```

- **type** - Indica o tipo de variável que a mensagem se refere. Tipo *String* com variação entre "*moistureSensor*", "*levelSensor*" e "*powerStatus*".
- **lastWatered** - Indica a informação de horas em que a última irrigação foi localizada. Variável usada para depuração e que está relacionada ao sensor de umidade, impedindo que múltiplas irrigações sejam realizadas em um curto espaço de tempo caso haja algum problema com o sensor de umidade (ex: caso este seja desenterrado). Tipo *Integer* com variação entre 0 e 23.

- **Dryness** - Valor analógico capturado pela medição do sensor de umidade. Tipo *Integer* com variação entre aproximadamente 100 e aproximadamente 4000.
- **waterLevel** - Valor numérico que indica a quantidade de água presente no tanque de água, que pode apresentar os seguintes valores:
 - * **0** - Abaixo de 30% .
 - * **30** - Entre 30% e 50%.
 - * **50** - Entre 50% e 75%.
 - * **75** - Entre 75% e completamente cheio.
 - * **-1** - Caso haja, por motivos de falha técnica, alguma incoerência na ordem em que as chaves do módulo de sensor de nível são ativadas (ex: chaves 1 e 3 ativada porém chave 2 desativada) esta mensagem de erro será enviada.
- Mensagem de conectividade - Enviada à Interface de Usuário pelo Módulo de Gerenciamento para indicar conexão com a rede, ou enviado pelo broker para indicar desconexão abrupta do Módulo de Gerenciamento. Contém os seguintes campos:

```
{ "type": "powerStatus", "powerOn": true }
```

 - **powerOn** - Variável booleana que indica se o Módulo de Gerenciamento está ou não conectado à rede. Variação entre *true* ou *false*.

3.3.3 Criptografia TLS

A comunicação do sistema, principalmente a conexão entre o Módulo de Gerenciamento e o broker faz uso de longas sessões para a contínua transmissão de dados, portanto o tráfego de dados na rede pode ser facilmente visualizado utilizando um software de interceptação *sniffer*, o que aumenta o

```

45 // certificado para acesso usando TLS
46 const char* ca_cert = \
47 "-----BEGIN CERTIFICATE-----\n" \
48 "MIIDSjCCAjKgAwIBAgIQRK+wgNajJ7qJMDmGLvhAazANBhkqkiG9w0BAQUFADA/\n" \
49 "MSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT\n" \
50 "DKRTVCBSb290IENBIFgzMB4XDTAwMDkzMIDIxMTIxOVoXDTIxMDkzMDE0MDExNVow\n" \
51 "PzEkMCIGA1UEChMbRGlnaXRhbCBTaWduYXR1cmUgVHJ1c3QgQ28uMRcwFQYDVQQD\n" \
52 "Ew5EU1QgUm9vdCBDQSBYMzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB\n" \
53 "AN+v6ZdQCINXtMxiZfaQguzH0yxrMMpb7NnDfcdAwRgUi+DoM3ZJKuM/IUmTrE40\n" \
54 "rz5Iy2Xu/NMhD2XSktkyj4z193ewEnu1lcCJo6m67XMuegwGMo0ifooUMM0Ro0Eq\n" \
55 "OLl5CjH9UL2AZd+3UW0DyOKIYepLYYHsUmu5ouJLGiifSK0eDNoJJj4XLh7dIN9b\n" \
56 "xiqKqy69cK3FCxo1kHRyxXtqqzTWMIn/5WgTe1QLyNau7Fqckh49ZL0Mxt+/yUFw\n" \
57 "7BZy1Sbs0FU5Q9D8/RhcQPGX69Wam40dutoIucbY38EVAjqr2m7xPi71XAicPNaD\n" \
58 "aeQQmxkqt1lX4+U9m5/wAl0CAwEAAsNCMEAwDwYDVR0TAQH/BAwAwEB/zA0BgnNV\n" \
59 "HQ8BAf8EBAMCAQYwhQYDVR0OBByEFMSnsaR7LHH62+FLkHX/xBVghYkQMA0GCSqG\n" \
60 "SIB3DQEBBQUAA4IBAACjGiybFwBcqR7uKGY30r+Dxz9Lwwmg1SBd49lZRNI+DT69\n" \
61 "ikugdB/0EIKcdBodfpga3csTS7MgR0SR6cz8faXbauX+5v3gTt23ADq1cEmv8uXr\n" \
62 "AvHRAosZy5Q6XkjEGB5YGV8eAlrwDPGxrancWYaLbumR9YbK+r1mM6pZW87ipxZz\n" \
63 "R8srzJmwN0jP41ZL9c8PDHIyh8bwRLtTcm1D9SZIm1Jnt1ir/mdl2cXjbDaJWFbm5\n" \
64 "JDGFoqgCWjBH4d1QB7wCCZAA62RjYJsWvIjJEubSfZGL+T0yjWW06XyxV3bqxbYo\n" \
65 "Ob8VZRzI9neWagqNdwvYkQsEjgfbKbYK7p2CNUQ\n" \
66 "-----END CERTIFICATE-----\n";
67

```

Figura 6 – Certificado para acesso TLS.

risco de ataques de *sniffing* caso os pacotes trocados entre módulos estiverem desprotegidos.

Sendo assim, o Sistema de Irrigação faz uso do protocolo de segurança TLS tanto no lado do Módulo de Gerenciamento quanto no lado da Interface de Usuário para criptografar os dados estabelecer uma comunicação segura entre estes componentes e o broker.

No lado do Módulo de Gerenciamento, o certificado TLS de cliente é inserido no código fonte do software como visto na figura 6 (obtido a partir da Autoridade de Certificação que certifica o broker) e utilizado pela biblioteca de comunicação para abrir uma conexão segura com o broker, no processo de estabelecimento da conexão, o certificado é enviado pelo Módulo de Gerenciamento para o servidor broker para autenticação de cliente e subsequente troca de chaves entre cliente e servidor.

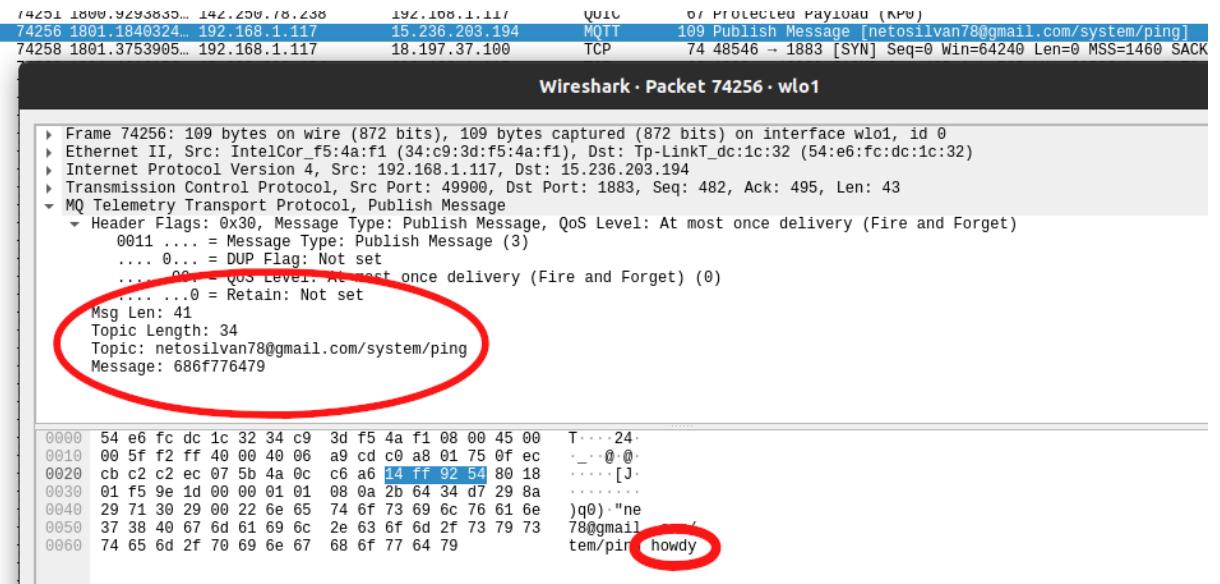


Figura 7 – Pacote MQTT desprotegido.

Utilizando o software de interceptação de dados **Wireshark** podemos verificar o conteúdo visível dos pacotes trocados entre o Módulo de Gerenciamento e Broker antes e depois da implementação da criptografia TLS. A figura abaixo mostra o quão vulneráveis são os dados em uma conexão não criptografada: pode-se ver desde parâmetros da mensagem como nível de QoS, tópico (system/ping, um tópico para testes) ou se ela será retida ou não pelo broker até seu conteúdo propriamente dito (a mensagem de teste "howdy" circulada).

Em seguida podemos ver na imagem 8, utilizando o mesmo software, uma mensagem criptografa usando TLS: pode-se ver que todo o conteúdo relevante da mensagem está contido em um conjunto de letras e números que pode ser lido apenas por um componente da rede que possui a chave pública correspondente, ou seja, o destinatário.

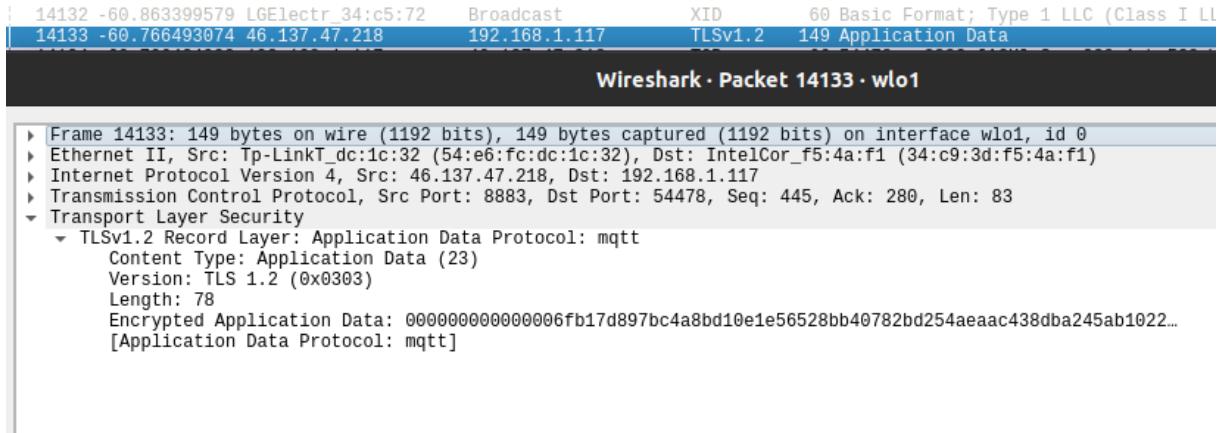


Figura 8 – Pacote criptografado.

3.4 Tanque de água

O tanque usado para armazenar a água utilizada na irrigação consiste em um galão plástico de 5 litros que pode ser encontrado em qualquer supermercado como embalagem para produtos de limpeza.

Para possibilitar a medição dos níveis de água no tanque fizemos buracos de 1 cm de diâmetro - medida equivalente ao diâmetro da seção de parafusamento do sensor - na parte externa do plástico. Os sensores foram então, introduzidos através das aberturas feitas e estas foram seladas utilizando as roscas por fora e as borrachas na seção de parafusamento dos sensores por dentro. Os resultados foram como na figura 9.



Figura 9 – Tanque de água.

Utilizamos três sensores para verificar os níveis de água. Estes sensores funcionam como chaves, que foram posicionadas para se fechar quando a água subir a um nível acima do sensor, estas chaves estão conectadas a um circuito de *pull-up* para impedir que ruídos sejam introduzidos à placa de Módulo de Gerenciamento caso algo aconteça aos fios que a conecta aos sensores. Este circuito impõe um nível alto (3.3V) às portas GPIO (*General Purpose Input/Output* ou Porta de Entrada/Saída de Uso Geral) que recebem os sinais dos sensores quando estes estiverem abertos, porém quando a agua passa pelo nível em que um dos sensor se encontra e a chave de nível se fecha, a porta correspondete àquele sensor passa a receber o nível lógico baixo (0V) como ilustrado na figura a seguir.

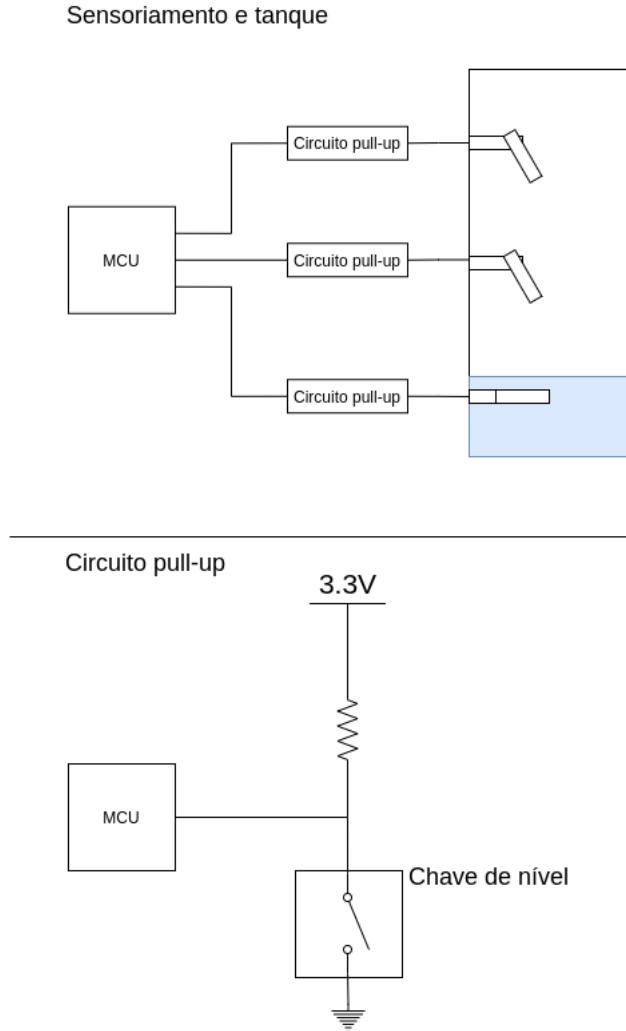


Figura 10 – Sensoriamento do tanque de água e esquema de pull-up.

3.5 Interface de Usuário

A Interface de Usuário é um módulo onde este pode verificar o estado do nível de água, as preferências configuradas(nesta aplicação denominados de *plans*), além de realizar modificações e controle do sistema. Desenvolvido nesta versão como uma aplicação da plataforma android.

Esta seção detalhará as funcionalidades e características da Interface de Usuário.

3.5.1 Tela Inicial

Na tela inicial todas as informações relevantes relacionadas ao controle e estado do sistema e preferências configuradas podem ser visualizadas. Os dados são dispostos da forma ilustrada na figura 11.

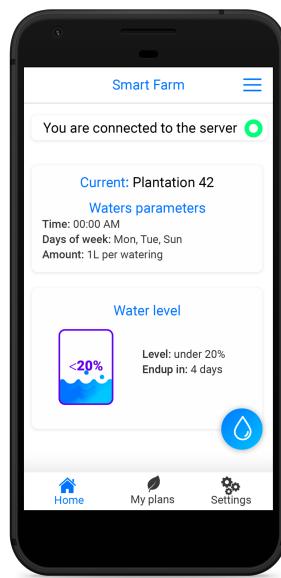


Figura 11 – Tela Inicial da Interface de Usuário.

- Verificação da conexão com o servidor - O primeiro bloco da tela mostra o estado da conexão do Módulo de Gerenciamento com o broker MQTT localizado na nuvem.
- Preferências de irrigação configuradas - O bloco seguinte (*Current*) detalha as variáveis do programa de irrigação configurado no momento.
 - *Time* - Horário de irrigação.
 - *Days of week* - Dias da semana em que a irrigação deve acontecer.
 - *Amount* - Quantidade de água usada em cada irrigação.
- Nível de água - O terceiro bloco da tela mostra o percentual de água atualmente no tanque. Como estamos utilizando sensores de nível que so-

mente serão ativados ou desativados com a passagem de água, o aplicativo mostra o nível em termos de maior que (>) ou menor que (<).

Utilizando como base o programa de irrigação selecionado, informações sobre a umidade do solo também podem ser encontradas neste bloco.

- Botão de irrigação imediata - No canto inferior direito da tela inicial há o botão de irrigação imediata, que após ser pressionado, envia um sinal de controle ao sistema para que este realize a irrigação imediatamente, com os parâmetros de quantidade de água selecionados e mostrados no bloco de variáveis.

3.5.2 Tela Meus Planos

Na aba "Meus Planos"(*My Plans*) o usuário pode visualizar os detalhes de todos os planos ou programas de irrigação (*Plans*), as diferentes preferências cadastradas pelo usuário. Ao tocar no nome de qualquer dos planos, um tela em cascata revela os detalhes desta preferência, nesta pode-se também remover planos.

No canto inferior direito da tela há um botão com o símbolo "+", cuja finalidade é cadastrar novas preferências. Detalhes da tela são mostrados na imagem 12.

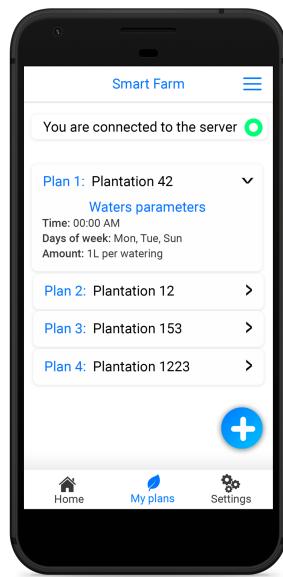


Figura 12 – Tela Meus Planos.

3.5.3 Tela de Configurações

Na tela de Configurações (*Settings*) o usuário pode selecionar um dos programas de irrigação(planos) anteriormente cadastrados para estar ativa no sistema. Somente uma preferência pode ser selecionada por vez e os detalhes desta são enviados através do broker para o Módulo de Gerenciamento. Detalhes da tela são mostrados na imagem abaixo.

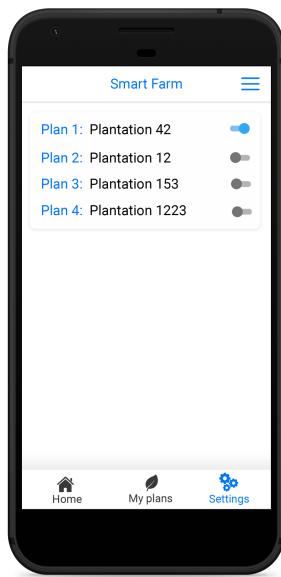


Figura 13 – Tela de Configurações.

3.5.4 Tela Adicionar Plano

Nesta tela o usuário pode cadastrar novos planos, ajustando as variáveis do programa de irrigação à sua preferência. A tela contém um formulário onde o usuário escolhe as configurações, este formulário não é muito diferente de uma tela de cadastro de alarmes de um smartphone, a diferença sendo que o formulário possui um campo para ajuste da quantidade de água.

Ao fim do preenchimento, no canto inferior direito, há um botão para finalizar e cadastrar o novo plano. Detalhes desta tela são encontrados na imagem 13.

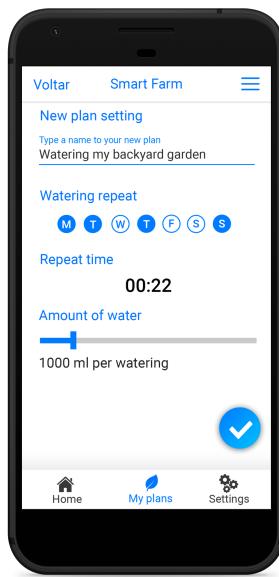


Figura 14 – Tela Adicionar Plano.

3.6 Interface de Configuração

O módulo Interface de Configuração consiste em uma página da web armazenada no sistema de arquivos do microcontrolador. A partir de um evento de solicitação do usuário (pressionamento do botão), esta página passa a ser servida pela placa e estar disponível para acesso pelo usuário quando este se conecta à placa utilizando um dispositivo smartphone ou computador com Wi-Fi, podendo assim, configurar o microcontrolador para se conectar a uma rede Wi-Fi em particular.

Para que o usuário possa se conectar à placa microcontroladora por Wi-Fi, foi utilizada a funcionalidade de *SoftAP* (*Software Enabled Access Point* ou Ponto de Acesso Habilitado por Software) que a placa Esp32 é capaz de fornecer. Esta funcionalidade permite ao dispositivo se tornar um ponto de acesso Wi-Fi. Uma vez conectado à rede, o usuário tem acesso à pagina de configuração do dispositivo dirigindo-se a um endereço específico.

Acessando a página de configuração, o usuário pode visualizar as redes Wi-Fi encontradas pela placa microcontroladora após processo de scan, escolher uma, inserir as credenciais, se necessário, e submeter os dados para que a placa microcontroladora possa se conectar à rede.

Como ilustrado no fluxograma de funcionamento da figura 15, após a submissão dos dados de rede para o microcontrolador, este reinicia a conexão no modo estação (cliente) e tenta se conectar utilizando as informações recebidas, caso este processo falhe, um alerta visual é emitido (led vermelho) e o processo de configuração pode ser refeito. A figura 16 mostra a visão de usuário da Interface de Configuração.

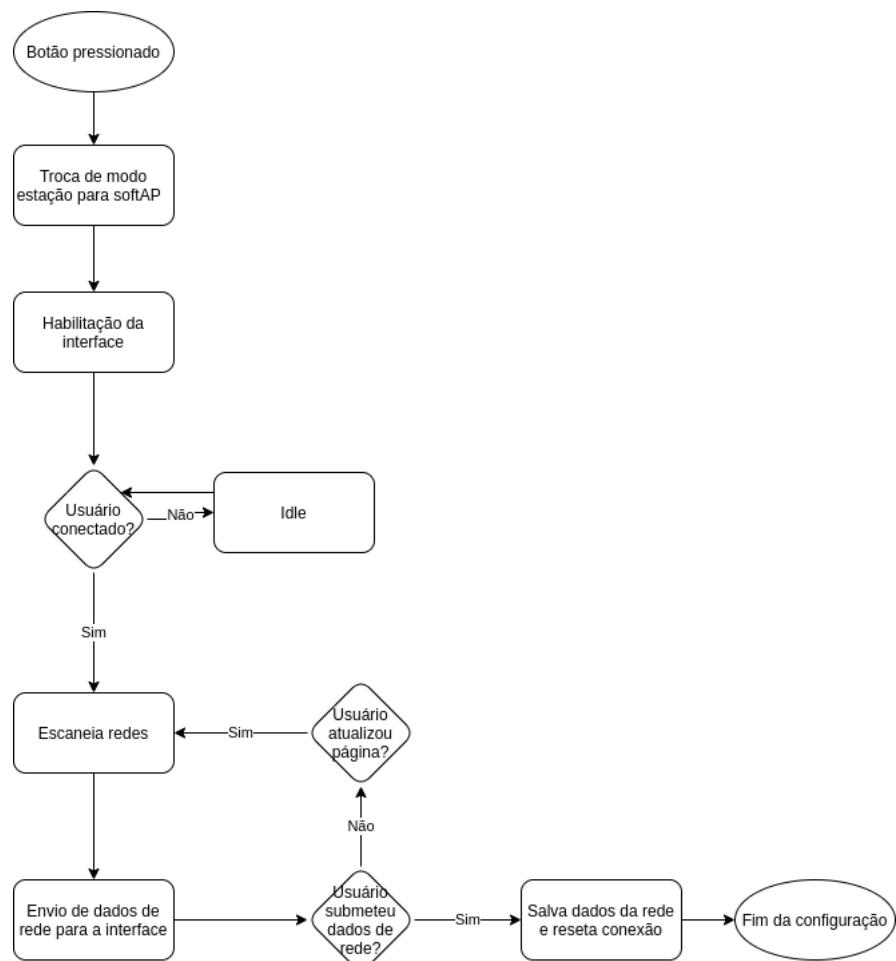


Figura 15 – Fluxograma de configuração do microcontrolador.



Figura 16 – Visão de usuário da Interface de Configuração.

3.7 Módulo de Gerenciamento de Irrigação

A figura 17 apresenta o fluxograma da máquina de estados pela qual o Módulo de Gerenciamento de Irrigação realiza o controle dos módulos adjacentes. Eventos como recebimento de solicitação de troca do programa de irrigação ou mensagens de irrigação imediata acontecem de forma assíncrona e as rotinas de substituição de programa e irrigação forçada causam o bloqueio da thread principal visto que utilizam recursos globais do módulo.

O bloco que denominado "rotina de configuração" é a forma simplificada da rotina descrita na seção sobre Interface de Configuração. No bloco de estabelecimento de conexão com o broker, o Módulo de Gerenciamento realiza o *Handshake TLS* com o broker.

No bloco de Ajuste de Horário, o Módulo de Gerenciamento se conecta a um servidor NIST (um servidor que fornece dados de hora e data na web) utilizando um pacote NTP (*Network Time Protocol* ou Protocolo de Tempo da Rede) para realizar a sincronização do relógio interno da placa (uma biblioteca de contagem de tempo) com a rede.

No bloco de checagem de sensores o módulo realiza uma verificação do sensor de umidade e caso a irrigação automática esteja habilitada, pode

inicializar a rotina de irrigação de acordo com a umidade do solo. Além disso neste bloco o módulo compara os três sensores que compõem o sensoriamento de nível entre si e entre seus estados anteriores e envia uma mensagem ao broker caso mudanças sejam detectadas.

O bloco que realiza a irrigação consiste de um conjunto de funções assíncronas que tem a função de enviar um sinal lógico alto para a porta de controle do motor de bombeamento, contar o intervalo de tempo que este nível lógico deve se manter alto e impedir qualquer chamada de irrigação subsequente enquanto esta estiver em andamento. Isto é feito através da mudança de uma *flag* global que indica que uma irrigação está em ocorrendo, estas funções utilizam o valor *amountWater* selecionado pelo usuário como parâmetro do intervalo de tempo.

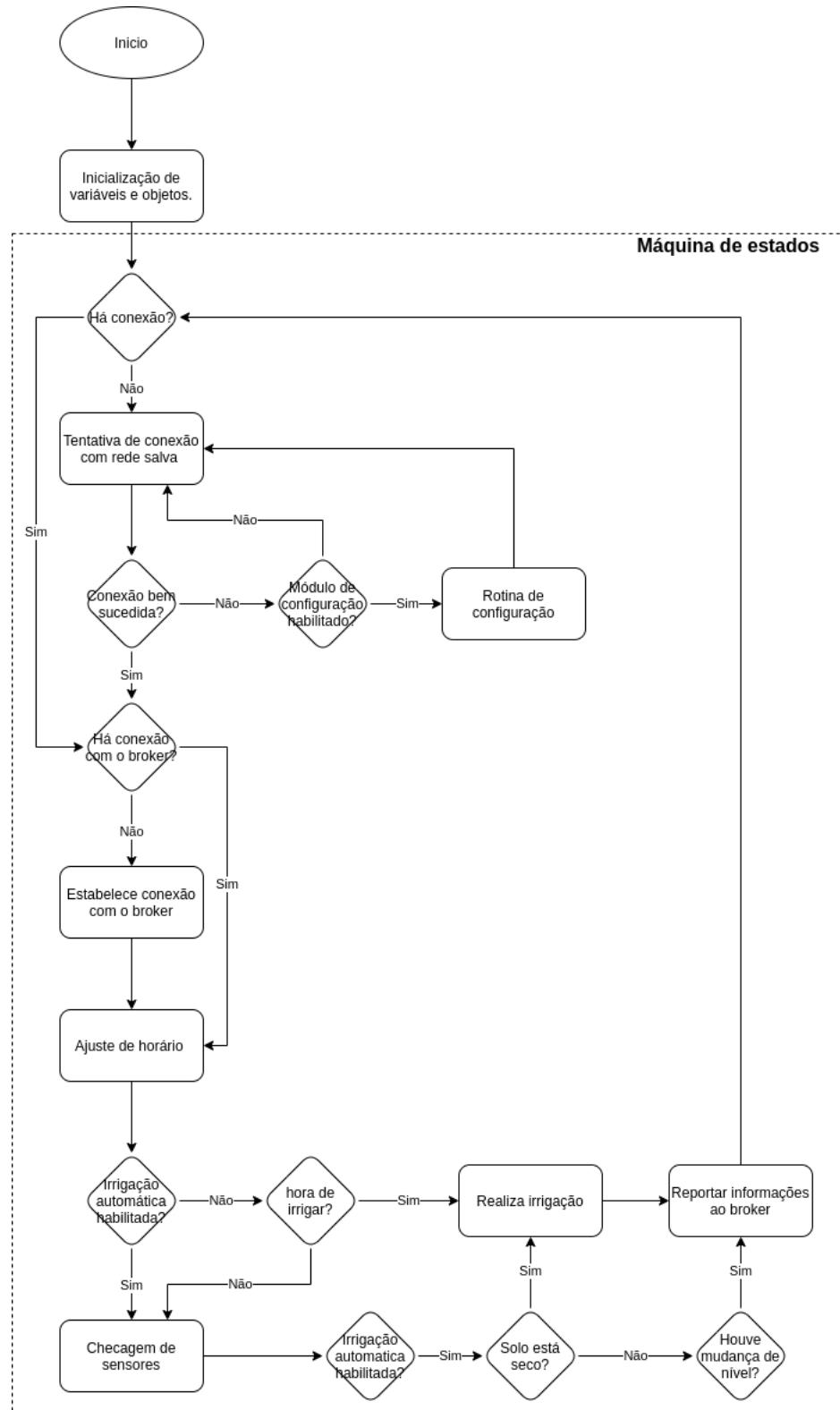


Figura 17 – Fluxograma do Módulo de Gerenciamento.

4 Testes realizados

4.1 Objetivos

Durante o desenvolvimento do projeto foram realizados continuamente dois tipos de testes: testes de comunicação entre os componentes do sistema (Interface de Usuário e Módulo de Gerenciamento) por meio do broker e testes de sanidade (teste para assegurar o continuado funcionamento do componente após mudanças) entre os blocos de um mesmo componente.

O primeiro tipo de teste envolve principalmente a leitura dos logs exibidos pelo broker para verificação das mensagens trocadas entre os componentes do sistema. Para este fim, se fez necessária a criação de um componente "expectador", um usuário conectado ao broker e assinante de todos os tópicos do sistema. Este tipo de testes foi realizado usando a plataforma **mosquitto CLI**, um cliente que permite a visualização das mensagens do broker a partir da linha de comando linux ou cmd windows.

O segundo tipo de teste envolve a leitura de logs internos do microcontrolador para verificação do correto funcionamento da máquina de estados e da Interface de Configuração que é inicializada a partir do input de usuário.

4.2 Mensagens de status retidas

Um dos objetivos das mensagens de status é que possam ser visualizadas a qualquer momento pelos componentes, por isto essas mensagens são do tipo **retido** e devem aparecer sempre que um cliente que assina o tópico relacionado a mensagem se conecta.

Na figura 18 pode-se ver que após a execução de **monitorMqtt.sh**, um script que inicializa a plataforma mosquitto CLI e assina todos os tópicos

relevantes, as mensagens de status relevantes para o sistema: programa selecionado, nível de água e status da conectividade do Módulo de Gerenciamento aparecem imediatamente. Estas mensagens contém a informação mais atualizada sobre estes dados.

```
edson@Epsilon:~/Documents/waterSystem$ bash monitorMqtt.sh | ts '[%Y-%m-%d %H:%M:%S]'  
[2021-07-07 18:29:14] {"type": "levelSensor", "waterLevel": 30}  
[2021-07-07 18:29:14] {"type": "powerStatus", "powerOn": true}  
[2021-07-07 18:29:14] {"amountWater": 2432, "gmtTimezone": -14400, "deadlineHour": 0, "deadlineMinute": 0, "deadlineSecond": 0, "deadlineDays": [-1, -1, -1, -1, -1, -1, -1], "automaticWatering": true}  
[]
```

Figura 18 – Mensagens de status após conexão com o broker.

4.3 Mensagens contínuas e mensagens de usuário

Outro tipo de mensagens trocadas entre os componentes do sistema são mensagens de umidade do solo, que são enviadas continuamente com um intervalo de tempo maior para que a vida útil do sensor seja preservada. Estas mensagens não são retidas devido à possibilidade de se desativar a irrigação automática por sensor a partir da interface de usuário. Na figura 19 pode-se ver mensagens recebidas ao longo de 15 minutos (o intervalo entre as mensagens foi reduzido para 5 minutos para maior facilidade de teste).

```
ond .0, deadlineDays .[-1, -1, -1, -1, -1, -1], automaticWatering :true]  
[2021-07-07 18:30:01] {"type": "moistureSensor", "lastWatered": 12, "Dryness": "862"}  
[2021-07-07 18:35:01] {"type": "moistureSensor", "lastWatered": 12, "Dryness": "831"}  
[2021-07-07 18:40:01] {"type": "moistureSensor", "lastWatered": 12, "Dryness": "814"}  
[]
```

Figura 19 – Mensagens de status do sensor de umidade.

O Módulo de Interface de Usuário pode mandar dois tipos de mensagens para o Módulo de Gerenciamento: mensagens de irrigação imediata e mensagens de troca de programa. Mensagens de irrigação imediata neste sistema consistem apenas de um caracter (mensagens vazias não serão repassadas pelo broker, elas tem a funcionalidade de limpar um tópico que possui uma fila de mensagens ou mensagens retidas) enviado através de um tópico específico,

o Módulo de Gerenciamento ignorará a mensagem mas saberá que se trata de uma irrigação imediata por causa do tópico.

Mensagens de troca de programa representaram a maioria dos problemas de comunicação encontrados, visto que são as mensagens que contém mais informação e qualquer erros de formatação poderiam levar a dificuldade e comportamentos inesperados no parseamento pelo Módulo de Gerenciamento. O mais comum deles sendo programas de irrigação corrompidos causando paradas na irrigação e até mesmo reset do sistema, situação essa que força o Módulo de Gerenciamento a utilizar um programa padrão escrito no código fonte com irrigação baseada em sensoriamento.

Na figura 20 pode-se ver um exemplo de mudança de programa, mais especificamente de um programa que irriga plantas todo dia, ao meio dia, para um programa de irrigação baseada em sensoriamento do solo. Abaixo destas, um exemplo de mensagem enviada para irrigação forçada e uma mensagem para fins de debug enviada pelo Módulo de Gerenciamento em resposta.

```
[2021-07-07 18:59:06] {"amountWater":800,"gmtTimezone":-14400,"deadlineHour":12,"deadlineMinute":0,"deadlineSecond":0,"deadlineDays":[1, 2, 3, 4, 5, 6, 0],"automaticWatering":false}
[2021-07-07 18:59:17] {"amountWater":2432,"gmtTimezone":-14400,"deadlineHour":0,"deadlineMinute":0,"deadlineSecond":0,"deadlineDays":[-1, -1, -1, -1, -1, -1],"automaticWatering":true}
[2021-07-07 18:59:24] {"amountWater":2432,"gmtTimezone":-14400,"deadlineHour":0,"deadlineMinute":0,"deadlineSecond":0,"deadlineDays":[-1, -1, -1, -1, -1, -1],"automaticWatering":true}
[2021-07-07 18:59:26] 1
[2021-07-07 18:59:27] okei! Vou jogar uma aguinha
□
```

Figura 20 – Mensagens de usuário do sistema.

4.4 Testes do Módulo de Gerenciamento

Os testes de sanidade do Módulo de Gerenciamento consistem principalmente de situações de depuração em que precisamos verificar o correto funcionamento dos diferentes blocos ligados ao Módulo de Gerenciamento, bem como a correta inicialização de variáveis do sistema e a correta conexão com o broker. Na figura 21 pode-se ver o processo de boot do módulo, após a conexão com uma rede Wi-Fi, o que segue-se é a tentativa de conexão com o broker, de

acordo com o fluxograma de funcionamento.

```
23:28:41.669 -> Initialization...
23:28:41.768 -> [FUNCTION] setWProgramFromMemory()
23:28:42.900 -> STA START
23:28:42.900 -> SCAN DONE
23:28:42.900 -> scan done
23:28:42.900 -> no networks found
23:28:42.900 ->
23:28:42.900 -> last saved net: Os Silva Wi Fi
23:28:42.900 -> last saved password: f0r@c0ntr0l3
23:28:42.900 -> Connecting to network Os Silva Wi Fi...
23:28:43.399 -> connectando com wifi Os Silva Wi Fi
23:28:43.898 -> connectando com wifi Os Silva Wi Fi
23:28:44.398 -> connectando com wifi Os Silva Wi Fi
23:28:44.897 -> connectando com wifi Os Silva Wi Fi
23:28:45.397 -> connectando com wifi Os Silva Wi Fi
23:28:45.896 -> connectando com wifi Os Silva Wi Fi
23:28:46.396 -> connectando com wifi Os Silva Wi Fi
23:28:46.895 -> connectando com wifi Os Silva Wi Fi
23:28:47.395 -> connectando com wifi Os Silva Wi Fi
23:28:47.894 -> connectando com wifi Os Silva Wi Fi
23:28:48.027 -> STA CONNECTED
23:28:48.394 -> connectando com wifi Os Silva Wi Fi
23:28:48.394 -> Connectado com a rede Os Silva Wi Fi
23:28:48.394 -> IP: 192.168.1.103
23:28:48.431 -> saving Os Silva Wi Fi to memory
23:28:49.393 -> Netowork saved
23:28:49.393 -> CURRENT PROGRAM: [
23:28:49.427 -> IRRIGATION TIME: 0:0:0
23:28:49.427 -> AMOUNT OF WATER: 2432
23:28:49.427 -> TIMEZONE: -14400
23:28:49.427 -> Automatic watering: 0
23:28:49.427 -> Watering days: -1 -1 -1 -1 -1 -1
23:28:49.959 -> Tentando conexão com o broker ca cert
23:28:49.959 -> -----BEGIN CERTIFICATE-----
23:28:49.959 -> MIIDSjCCAjKgAwIBAqIQRK+wqNajJ7qJMDmGLvhAazANBqkqhkiG9w0BAQUFADA/
```

Figura 21 – Logs de Inicialização do Módulo de Gerenciamento.

Outra situação verificada é o envio e recebimento de mensagens de status, estas mensagens retidas pelo broker atualizam os componentes imediatamente após inicialização. Na figura 22 o módulo envia uma mensagem de status de conectividade imediatamente após conexão com o broker, e no momento seguinte recebe uma mensagem de configuração de programa de irrigação que estava retida no broker.

```

23:12:08.079 -> (TLS) 59082674fbcc44d0b95579ca81e3201a2.s1.eu.hivemq.cloud
23:12:08.079 -> Usuario: avalon
23:12:08.079 -> Porta: 8883
23:12:08.079 -> tentativas 0
23:12:10.737 -> Conectado ao Broker!
23:12:10.737 -> Sending status msg:
23:12:10.737 -> {"type":"powerStatus", "powerOn":true}
23:12:10.737 ->
23:12:11.036 -> Callback called
23:12:11.036 -> Data : dataCallback. Topic : [system/hardware/setupProgram]
23:12:11.036 -> Data : dataCallback. Payload : {"amountWater":2432,"gmtTimezone":-
23:12:11.069 -> [FUNCTION] changeDefaultProgram()
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> default program changed:
23:12:11.069 -> [FUNCTION] saveProgramToMemory()
23:12:11.069 -> Auto watering: 1
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> -1
23:12:11.069 -> CURRENT PROGRAM: [
23:12:11.069 -> IRRIGATION TIME: 0:0:0
23:12:11.069 -> AMOUNT OF WATER: 2432
23:12:11.069 -> TIMEZONE: -14400
23:12:11.069 -> Automatic watering: 1
23:12:11.102 -> Watering days: -1 -1 -1 -1 -1 -1 -1 -1
23:12:11.102 -> 

```

Figura 22 – Logs de recebimento de mensagens retidas.

A maneira utilizada para verificação do funcionamento dos blocos adjacentes é a de identificar cada função chamada por eventos externos, na figura 23 pode-se ver um conjunto de funções chamadas quando há uma mudança no nível de água, um processo que culmina no envio de uma mensagem de status para o broker.

```

23:16:37.331 -> [FUNCTION] setLevel()
23:16:37.331 -> [FUNCTION] broadcastWaterLevel()
23:16:37.331 -> [FUNCTION] getLastWateredStatus()
23:16:37.331 -> Sending status msg:
23:16:37.331 -> {"type":"levelSensor", "waterLevel":0, "lastWatered":-1}
23:16:37.331 ->
23:16:37.331 -> Mensagem enviada e retida: {"type":"levelSensor", "waterLevel":0, "lastWatered":-1}
23:16:37.331 -> Para: system/hardware/temp

```

Figura 23 – Logs de recebimento de mudança de nível de água.

Na figura 24 pode-se ver o protótipo do sistema de irrigação em atividade utilizando os módulos sensores e com uma cobertura de 4 plantas.



Figura 24 – Protótipo do Sistema de Irrigação.

5 Conclusões

Neste trabalho, foi proposto um sistema de irrigação para hortas residenciais que utiliza o protocolo de comunicação MQTT em conjunto com uma aplicação para aparelhos smartphones. O sistema oferece a possibilidade de customização de variáveis relacionadas à irrigação como quantidade de água irrigada, intervalo entre as irrigações e até mesmo horário agendado para irrigação. Bem como possui a funcionalidade de automatizar a irrigação utilizando um sensor de umidade do solo como base. O sistema utiliza criptografia TLS para assegurar a integridade da comunicação cliente-servidor e dos pacotes trocados entre os componentes.

O desenvolvimento foi dividido em partes que eventualmente passaram a ser desenvolvidas concorrentemente devido ao aumento de funcionalidades e complexidade dos programas. Primeiramente foi desenvolvida uma versão inicial do Módulo de Gerenciamento junto com o bloco de sensoriamento e o controle do motor de bombeamento. Após terminada esta parte inicial, foi desenvolvida a estrutura de comunicação, e em seguida a Interface de Usuário.

Ao longo do projeto constatou-se a necessidade da adição de uma Interface de Configuração para troca de credenciais de rede, visto que caso houvesse uma troca permanente de roteador, o sistema estaria impossibilitado de receber informações do usuário ou mesmo mandá-las. Durante o desenvolvimento deste bloco, diversos problemas surgiram devido à exclusão mútua entre o modo de Estação cliente e o modo de Ponto de Acesso que o microcontrolador pode assumir.

Após o desenvolvimento do bloco de Interface de Configuração, foi adicionado o sensor de umidade ao sistema. Por este ser um método que verifica o quanto seco o solo se encontra para realizar uma regagem, questionou-se a necessidade da opção de agendamento para a irrigação. Porém ao longo dos testes verificou-se que o formato de vasos, o tipo de planta e a qualidade do

solo influenciam a leitura do sensor, e um processo mais intervencionista de irrigação ainda seria válido pois seria inviável colocar sensores em cada vaso.

O sistema atual ainda é incompatível com hortas grandes, devido a ser um protótipo com apenas um motor de bombeamento, mas obteve sucesso em manter irrigadas as plantas sob sua cobertura e prover relatórios do status de irrigação através do app.

5.1 Propostas para Trabalhos Futuros

Sugere-se como propostas de adição ao presente projeto:

- Utilizar e gerenciar múltiplos motores de bombeamento para uma cobertura maior do sistema.
- Diminuir a dependência da rede Wi-Fi pelo sistema adicionando módulos de verificação e ajuste local do tempo.
- Realizar a configuração de redes Wi-Fi do sistema via Bluetooth como substituto da Interface de Configuração devido aos problemas causados por esta.
- Tornar a visualização e controle das variáveis globais do sistema menos dependente da rede Wi-Fi, através de displays LCD e módulos de controle remoto.

Referências

- [1] Elprocus. *Ways to Automatic Plan Irrigation System using Microcontroller* <https://www.elprocus.com/microcontroller-based-automatic-irrigation-system/>. Acesso em 20 de Fevereiro de 2020.
- [2] Money Crashers. *How to Start a Home Vegetable Garden – Benefits* <https://www.moneycrashers.com/how-to-save-money-with-a-home-garden/>, acesso em 21 de Fevereiro de 2020.
- [3] Portal IBM Developer:<https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html>, acesso em 12 de Novembro de 2019.
- [4] <https://realtimeapi.io/hub/publishsubscribe-pattern/>, acesso em 12 de Novembro de 2019.
- [5] MQTT. *MQTT: The Standard for IoT Messaging*. Disponível em <http://mqtt.org>. Acesso em 12 de Novembro de 2019.
- [6] AssetWolf. *Quality of Service (QoS)*. Disponível em <https://assetwolf.com/learn/mqtt-qos-understanding-quality-of-service>, Acesso em 26 Jun. 2021.
- [7] HiveMQ. *Retained Messages - MQTT Essentials: Part 8* . Disponível em <https://www.hivemq.com/blog/mqtt-essentials-part-8-retained-messages/>, Acesso em 26 Jun. 2021.
- [8] HiveMQ. *Last Will and Testament - MQTT Essentials: Part 9*. Disponível em <https://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament/>, Acesso em 26 Jun. 2021.
- [9] HiveMQ. *TLS/SSL - MQTT Security Fundamentals* <https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl/>. Acesso 03 de Março de 2020
- [10] I. Grigorik, *High Performance Browser Networking* O'Reilly Media, 2013.