UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

DANIEL HENRIQUE GREHS

Sistema de irrigação doméstico baseado em Internet das Coisas

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Alexandre Carissimi

Porto Alegre
2016

AGRADECIMENTOS

Agradeço primeiramente aos meus pais e ao meu irmão, pelo amor, incentivo e apoio incondicional. Meus agradecimentos aos amigos e companheiros de trabalhos que fizeram parte da minha formação e que vão continuar presentes em minha vida com certeza. Também agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. A palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos. Por fim, agradeço também a esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior.

RESUMO

Os recentes avanços tecnológicos estão oferecendo uma grande quantidade de dispositivos capazes de sentir o ambiente, interagir com ele, e transmitir dados coletados para a Internet. A Internet das Coisas, como essa evolução vem sendo chamada, oferece uma gama enorme de novas aplicações. Entre elas estão eletrodomésticos, meios de transporte e até mesmo roupas, tênis e maçanetas conectadas à rede. O presente trabalho apresenta uma proposta para um sistema de irrigação doméstico, com interface via aplicativo móvel e baseado nos conceitos da Internet Das Coisas, utilizando o protocolo MQTT (*Message Queueing Telemetry Transport*),e o *middleware* de comunicação Mosquitto, hospedado na Amazon Web Services.

Palavras-chave: Internet das Coisas, MQTT, irrigação doméstica, ESP8266, Mosquitto, Amazon Web Services.

Domestic irrigation system based on Internet of Things

ABSTRACT

Recent technological advances are offering a massive quantity of devices capable of sensing the environment, interacting with it, and transmit collected data to the Internet. The Internet of Things (IoT), as this evolution is called, offers a huge range of new applications, like transports, home appliances, clothes, shoes and even door handles connected to the Internet. This work proposes a domestic irrigation system, based on the concepts of Internet of Things, with mobile interface, using protocols like MQTT, Mosquitto as communication middleware, and hosted on Amazon Web Services.

Keywords: Internet of Things, MQTT, domestic irrigation, ESP8266, Mosquitto, Amazon Web Services.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
AWS	*Amazon Web Services*
CGNAT	*Carrier Grade Network Address Translation*
CoAP	*Constrained Application Protocol*
CPU	*Central Processing Unit*
GSM	*Global System for Mobile*
HTTP	*Hypertext Transfer Protocol*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
IoT	*Internet of Things*
IP	*Internet Protocol*
IPSO	*IP for Smart Objects*
IPv4	*Internet Protocol Version 4*
IPv6	*Internet Protocol Version 6*
ISP	*Internet Service Provider*
LSN	*Large Scale NAT*
MIT	*Massachusetts Institute of Technology*
MQTT	*Message Queueing Telemetry Transport*
NC	*Normally Closed*
NO	*Normally Opened*
OASIS	*Organization for the Advancement of Structured Info Standards*
PLC	*Power Line Communication*
REST	*Representational State Transfer*
RFID	*Radio-Frequency Identification*
RISC	*Reduced Instruction Set Computing*
SDK	*Software Development Kit*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UML	*Unified Modeling Language*
UMTS	*Universal Mobile Telecommunications System*
URI	*Universal Resource Identifier*
URN	*Uniform Resource Name*

WSN	*Wireless Sensor Network*
XML	*Extensible Markup Language*
XMPP	*Extensible Messaging and Presence Protocol*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, a Internet evoluiu bastante. Iniciada como uma rede acadêmica, passou a ser uma rede global, integrada e usada regulamente por bilhões de pessoas. A constante evolução se dá devido ao fato da Internet ser um sistema de comunicação aberto, movimentando diversas áreas do conhecimento e aceitando que novos serviços e protocolos sejam criados.

Um dos novos usos da Internet, é a Internet das Coisas. O termo foi utilizado pela primeira vez em 1999, por Kevin Ashton, durante uma apresentação em que foi proposto o uso da Internet e tecnologias de endereçamento à cadeias logísticas, sem a interferência direta do ser humano, com o objetivo de melhorar o fluxo de produtos, com movimentações mais rápidas e confiáveis, além de permitir a cooperação entre todas as partes da cadeia, com informação partilhada. Asthon afirma que a Internet atual é totalmente dependente do ser humano para criar dados, entretanto o ser humano é limitado para recuperar e tratar dados sobre o mundo físico [Ashton 2009].

Dessa forma, a Internet das Coisas é um paradigma em que os objetos do dia-a-dia ganharam capacidade de interagir entre si e com o meio onde estão inseridos, sem a participação direta do homem. Para que isso seja possível, diversos protocolos, modelos e arquiteturas estão surgindo e sendo estudados cuidadosamente, de modo a suportar o grande espectro de possibilidades de serviços e aplicações que essa inovação trará para as nossas vidas. Automação residencial, rede de energia inteligente, cadeias logísticas, mobilidade urbana e gerenciamento de processos são apenas alguns dos vários cenários potenciais. Todas essas possibilidades tornam fácil visualizar os grandes benefícios que essas aplicações podem trazer para a nossa sociedade [Atzori et al. 2010].

Entre as aplicações na área de saúde, o monitoramento remoto de pacientes permitiria colocar em prática, e de maneira eficiente, a medicina preventiva. No domínio pessoal, a localização de um pertence esquecido ou perdido seria muito mais fácil. Além disso, os ambientes inteligentes seriam possíveis através de sensores e atuadores, que poderiam ligar e desligar ar condicionado e lâmpadas remotamente.

A área que trata que das tecnologias de automação residencial, em que este trabalho está inserido,é chamada de domótica. Nela, permite-se automação de tarefas rotineiras, melhorias de segurança, e até mesmo redução do consumo de energia [Pinto 2010].

1.1 Objetivos do trabalho

Este trabalho terá como objetivo o desenvolvimento de um sistema de irrigação doméstico utilizando os conceitos da Internet das Coisas. Esse sistema permitirá ao usuário doméstico realizar a irrigação das suas plantas de maneira autônoma, através de um sensor de umidade e um atuador capaz de liberar água para a planta. Tal sistema poderá ser monitorado remotamente através de um aplicativo para celulares *Android,* que estará conectado à Amazon Web Services, onde o sistema estará hospedado. Entre as tecnologias utilizadas por esse sistema estarão sensores de umidade de solo, atuadores, microcontroladores e um serviço em nuvem para controle central.

1.2 Organização do texto

Este trabalho é organizado em 5 capítulos, além desta introdução. O capítulo 2 aborda os conceitos de Internet das Coisas, assim como arquiteturas, protocolos e *middlewares* que a tornam possível. Em seguida, o capítulo 3 fornece a solução proposta, desde a sua arquitetura até os seus requisitos, diagramas e interfaces. No capítulo 4, os detalhes da implementação são apresentados, tais como protocolos e softwares utilizados para atingir o objetivo final. Já no capítulo 5, uma avaliação do protótipo é apresentada, contendo as dificuldades encontradas, cenários de teste, resultados, infraestrutura e metodologia. Por fim, o capítulo 6 apresenta a conclusão do trabalho, mostrando os principais desafios e contribuições do trabalho, além de sugestões para melhorias futuras.

2 INTERNET DAS COISAS

Os avanços recentes na área de telecomunicação e de redes de computadores provocaram o surgimento do paradigma Internet das Coisas. A sua origem parte de um laboratório do MIT (*Massachusetts Institute of Technology*) chamado Auto-ID Labs. Nesse laboratório, a pesquisa no campo de localização e identificação usando sensores sem fio, em 1999, deu inicio a área de estudos da Internet das Coisas [Zhu et al. 2010].

Ao longo deste capítulo, será apresentado os conceitos e tecnologias envolvidas nas aplicações da Internet das Coisas. Vale salientar que o acrônimo IoT, do inglês *Internet of Things*, será utilizado para evitar repetição.

2.1 Definição de IoT

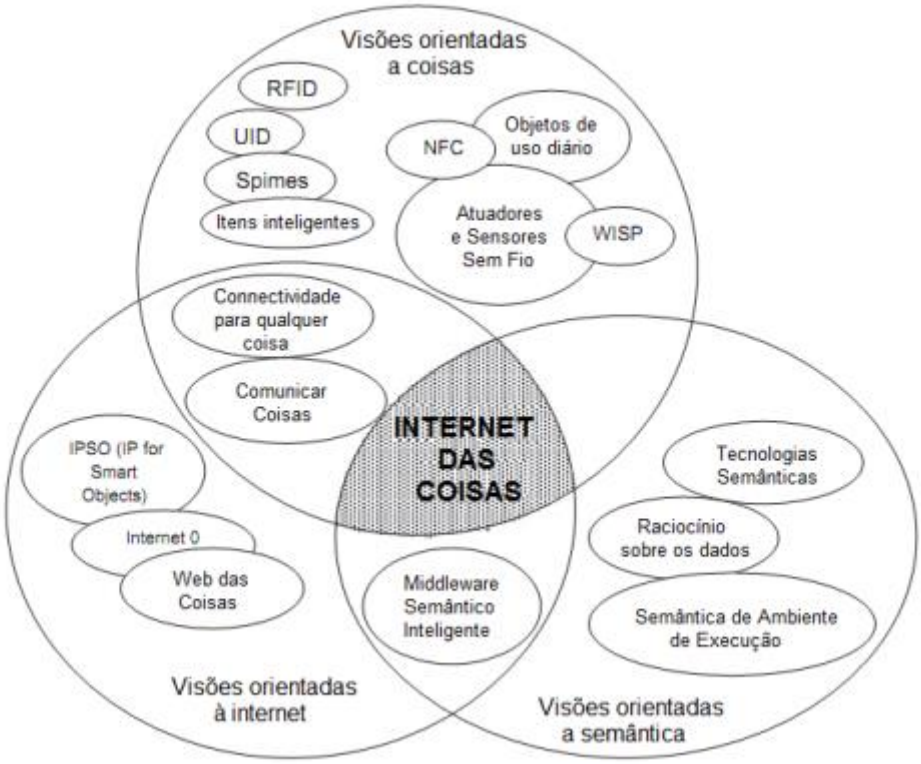
A definição do que é a Internet das Coisas não é algo simples, e vem causando uma certa confusão na literatura desde o surgimento do termo, em 1999. A razão dessa confusão pode ser consequência do próprio nome “Internet das Coisas”. Sintaticamente, o nome é composto por dois termos. O primeiro deles, “Internet”, sugere uma interpretação do ponto de vista orientado a rede. Já o segundo, “Coisas”, leva o foco para objetos genéricos conectados.

Essas diferentes interpretações geram duas possíveis visões: orientada à Internet, e orientada à Coisa. Ambas utilizadas pelo mercado/indústria conforme suas finalidades, conhecimentos e interesses [Atzori et al. 2010].

De qualquer forma, o significado semântico da expressão combinada “Internet das Coisas” leva a uma terceira visão, orientada à semântica, em que *Internet of Things* (IoT) é uma rede de objetos interconectados com um sentido associado aos seus dados, permitindo assim a escalabilidade dos mesmos e manipulação das informações geradas. [INFSO et al. 2008].

As diferentes interpretações de Atzori, assim como as tecnologias, padrões e conceitos principais são apresentados na Figura 2.1. Tais ideias estão relacionadas com a visão de Internet das Coisas que eles mais contribuem.

Figura 2.1 – Visões da Internet das Coisas



Fonte: Adaptado de Atzori et al. 2010

- **Visões orientadas às Coisas:** A primeira definição de IoT deriva de orientada a Coisa. Nessa visão, as Coisas são muito simples, como as etiquetas de *Radio-Frequency IDentification* (RFID). O RFID, e diversas tecnologias semelhantes, como o *Wireless Sensor Network* (WSN), vem sendo estudadas e utilizadas por instituições como o The Auto-ID Labs e o EPCglobal, a fim de definir os padrões globais de IoT. Mas, fica claro que a Internet das Coisas não pode ser apenas um sistema de endereçamento em que os únicos objetos são RFIDs.
- **Visão orientada à Internet:** A segunda definição de IoT deriva de orientada à Internet. Nessa visão, as Coisas podem comunicar-se automaticamente com computadores, provendo serviços para o benefício da raça humana [Dunkels et al. 2008]. Essa visão propõe uma Internet das Coisas como uma infraestrutura global que conecta objetos físicos, através da estrutura de Internet que já existe, adicionada de captura de dados autônoma, conectividade e interoperabilidade. Essa visão mais próxima da Internet se relaciona com o conceito de *Web of Things*. De acordo com esse conceito, os padrões utilizados na Web são reutilizados para conectar os objetos/coisas do mundo real [D. Guinard et al. 2009].

- **Visão orientada à semântica:** Por fim, a definição semântica de IoT diz que o número de dispositivos envolvidos na Internet do futuro é muito maior do que o esperado, o que trará problemas como armazenamento, interconexão, pesquisa e classificação das informações geradas por dispositivos IoT. Nesse cenário, o conhecimento semântico dos dados permitirá o tratamento e extração automatizada dessas informações, o que facilitará aspectos como escalabilidade de armazenamento e infraestrutura de comunicação.

Dito isso, fica claro que a visão de IoT deve ser, de fato, a combinação entre os aspectos das três visões discutidas anteriormente, sendo composta de elementos como o hardware (conjunto de sensores e atuadores), o *middleware* responsável pelo conhecimento semântico dos dados e a forma de apresentação de tais dados para o usuário. Cada um desses elementos, ou componentes, possui diversas tecnologias e papéis associados que serão apresentados na seção 2.3.

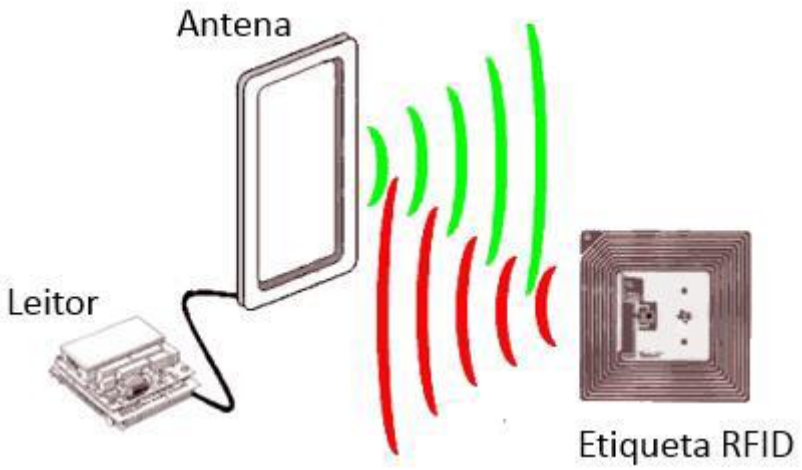
2.2 Dispositivo IoT

Um dispositivo, mesmo conectado à nuvem, talvez não tire proveito do real potencial da Internet das Coisas. Para isso, precisamos considerar vários aspectos durante o projeto de um dispositivo IoT: entre eles o hardware, o software embarcado, o protocolo e o meio de comunicação – além do serviço que receberá os dados.

Na Internet das Coisas, cada objeto conectado é associado a um identificador digital, como por exemplo, um nome, IP ou URI (*Universal Resource Identifier*). Há uma grande quantidade de tecnologias que permitem essa identificação, como o RFID, endereços IP, URNs (*Uniform Resource Name*) e URIs.

A tecnologia mais simples utilizada para esse fim é o RFID. Do ponto de vista físico, um sistema RFID é basicamente composto por uma etiqueta RFID e de um leitor RFID, também chamado de base transmissora, como mostrado na Figura 2.2.

Figura 2.2 – Sistema RFID



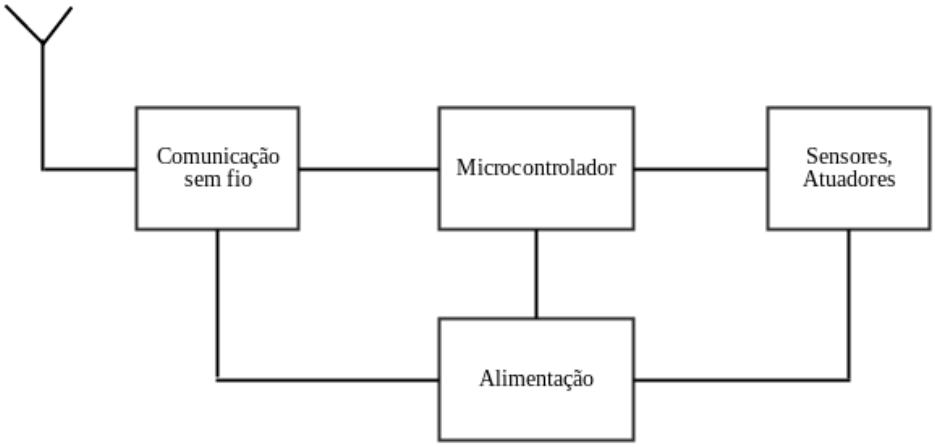
Fonte: Autor

No caso de uma etiqueta RFID passiva, o seu funcionamento é possível através da energia gerada pelo campo magnético da antena, permitindo assim uma resposta da etiqueta sem a necessidade de uma fonte de energia. Normalmente essa resposta são os dados relacionados à identidade virtual do objeto.

Superado o problema do endereçamento, um dispositivo IoT também deve ser capaz de interagir com o ambiente. Para isso, a definição do seu hardware é muito importante e pode ser separado em quatro componentes principais como mostrado na Figura 2.3.

O primeiro deles, componente de comunicação, é o módulo que permite ao dispositivo inteligente enviar e receber informações. Há muitos padrões de comunicação que podem ser usados para esse envio e recebimento, como IEEE 802.15.4 (*Zigbee*, ISA100.11a e *6LowPAN*), IEEE 802.11 (*WiFi*) ou *PowerLine Communication* (PLC). A escolha de qual padrão usar depende de aplicação e do ambiente físico [Vasseur, J.-P et al. 2010].

Figura 2.3 – Componentes do Dispositivo IoT



Fonte: Adaptado de Vasseur, J.-P et al. 2010

O segundo componente é o microcontrolador, onde está a inteligência do objeto. Dentro desse componente temos um sistema computacional completo, composto por CPU, memória e portas de entrada/saída, que deve ser programado para, baseado nas informações recebidas pelos sensores, encaminhá-las para fora do dispositivo, através do componente de comunicação descrito anteriormente. Comumente, utiliza-se microcontroladores com a aparência de um microchip tradicional para esse fim, e a sua alimentação podem ser desde tomadas elétricas, até baterias ou placas solares.

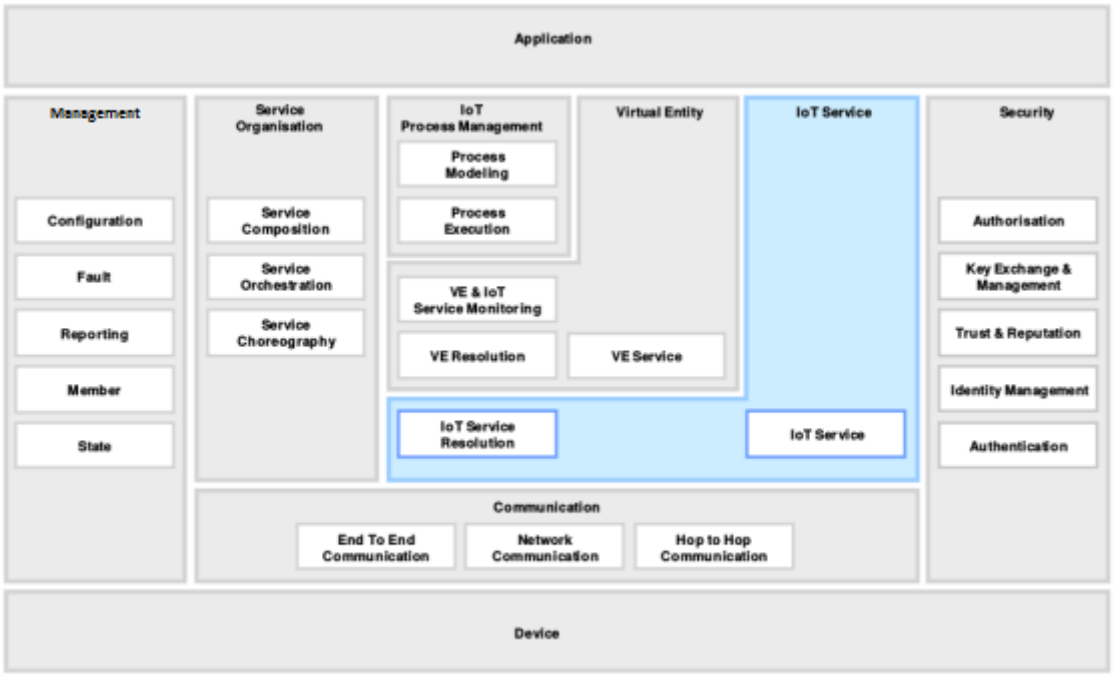
O terceiro componente é o conjunto de transdutores, é através desse componente que se tem a interação com o ambiente físico no qual o objeto encontra-se. Genericamente, um transdutor é um dispositivo que transforma uma fonte de energia em outra, como os sensores e atuadores. Um sensor é capaz de traduzir determinada condição física, como temperatura ou luminosidade, para um sinal elétrico, que será recebido pelo microcontrolador. Já um atuador, tem como principal função afetar o ambiente com alguma ação, ou mudança, normalmente eles recebem comandos como um sinal elétrico do microcontrolador e executam alguma tarefa mecânica.

Por fim, o último componente é a fonte de energia (alimentação) para os demais componentes, que pode ser desde uma tomada elétrica para os dispositivos que necessitam maior potência como atuadores eletromecânicos, passando por baterias e placas solares para os dispositivos mais simples, como microcontroladores de baixo consumo.

2.3 Arquitetura IoT

Uma das iniciativas para padronizar a arquitetura utilizada por soluções de IoT foi o projeto IoT-A, um consórcio europeu de empresas formado de 2010 a 2013, com o intuito de definir um modelo de referência. A visão da arquitetura proposta pelo IoT-A, apresentada na Figura 2.4, nos permite definir as camadas da arquitetura, cuja função é importantíssima já que permite um projeto que atende os requisitos esperados pelas indústrias e empresas.

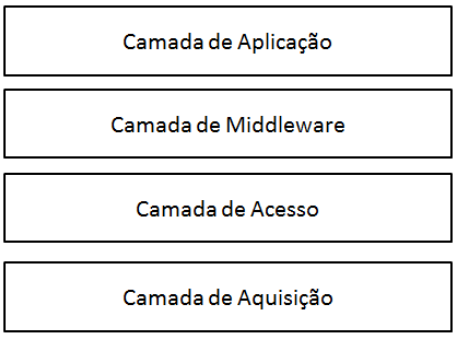
Figura 2.4 – Arquitetura IoT-A



Fonte: Bauer et al. 2014

Tanto a arquitetura proposta pelo IoT-A, quanto as sugeridas por [Atzori et al. 2010] e por [Bandyopadhyay and Sen 2011], podem ser separadas em quatro camadas genéricas como mostrado na Figura 2.5: aquisição de dados, acesso, *middleware* e aplicação. De acordo com as características dos dispositivos envolvidos e sofisticação da aplicação, as camadas serão mais ou menos complexas. Em outro extremo, podem até ser suprimidas.

Figura 2.5 – Camadas IoT



Fonte: Autor

A camada de aquisição tem duas funções básicas, obtenção de dados físicos e comunicação. Dentro dessa camada, estão localizados os sensores, atuadores, sistemas embarcados e etiquetas RFID. Esses hardwares serão os responsáveis por prover identificação, coletar informações físicas do meio em que se encontram (e.g sensores), atuar mecanicamente no meio (e.g atuadores) e permitir o envio/recebimento de informações sobre tais sensores e identificadores para as camadas posteriores (e.g sistemas embarcados) [Fleisch 2010]. Vale salientar que a comunicação dessa camada é apenas com a camada de acesso, e não com o usuário final da aplicação.

A camada de acesso é responsável pelo primeiro tratamento dos dados recebimentos, ou seja, executar atividades necessárias para que esses dados cheguem às próximas camadas de forma pertinente [Yang et al., 2011]. Entre essas atividades estão a comunicação com a Internet, normalmente utilizando WiFi ou tecnologias de dados celular, como *Universal Mobile Telecommunications System* (UMTS), *Global System for Mobile communications* (GSM) e *Long Term Evolution* (LTE).

A camada de *middleware* possui como funcionalidades principais: (i) ser a interface com protocolos da camada de acesso; (ii) oferecer abstração de dispositivo; (iii) atuar sobre o contexto dos dados e oferecer abstração para a aplicação [Zarghami 2013]. A fim de atingir a primeira funcionalidade, a camada de *middleware* deve ser capaz de definir os protocolos de troca de informação com as camadas inferiores, independente das diferenças físicas que podem existir entre os dispositivos. A segunda funcionalidade, abstração de dispositivo, será obtida através da definição da sintaxe das mensagens que serão transmitidas pelos protocolos de comunicação. Essa camada também deve ser capaz de atuar sobre o contexto dos dados, ou seja, coletar as informações oferecidas pelos dispositivos, e, posteriormente, selecionar a informação que pode ter algum impacto sobre o processamento, como a necessidade de tomar

uma decisão. A última função fundamental da camada de *middleware* é oferecer abstração para a aplicação IoT, ou seja, esse componente deve prover uma interface para comunicar-se com a aplicação e o cliente final. Um exemplo dessa abstração pode ser a utilização de uma interface *RESTful* (*Representational State Transfer*) para facilitar a interação entre o cliente e os dispositivos. Dessa forma, operações para recuperar informações de um dispositivo ou tarefa, remover algum recurso, criar novos recursos e atualizar um recurso podem ser feitas de maneira simples pelo usuário, sem precisar de conhecimento sobre as implementações e tarefas necessárias por trás dessas operações, usando primitivas como GET, DELETE, PUT e POST.

Por fim, na camada de aplicação, temos, finalmente, a entrega das funcionalidades aos usuários do IoT, como um conjunto do trabalho realizado por todas as camadas anteriores. Essa camada é responsável por apresentar os recursos oferecidos pelo sistema e permitir atuações e tarefas de acordo com as funcionalidades. Comumente, essa aplicação se encontra em aplicativos de celular ou websites, de fácil utilização pelo usuários. Entretanto, com o aumento da maturidade de diversas tecnologias (como RFID), tais aplicações também podem estar transparentes aos usuários, como casos de utilização de IoT em *Smart Grid*, logística e dispositivos inteligentes.

2.4 Protocolos

Um dos aspectos mais importantes para o desenvolvimento da Internet das Coisas é a padronização da comunicação. Esse ponto tem sido muito discutido nos últimos anos, trazendo consigo uma gama de protocolos para atender aos requisitos das aplicações da melhor maneira possível.

Como dito anteriormente, baseado na visão “Internet” do IoT, em que se espera uma integração entre os dispositivos IoT e a Internet que já utilizamos, a escolha do protocolo IP para endereçamento é a mais óbvia.

Entretanto, é necessário definir qual o protocolo de nível de aplicação será utilizado, sobre o IP, e é nesse ponto que a gama de protocolos é importante.

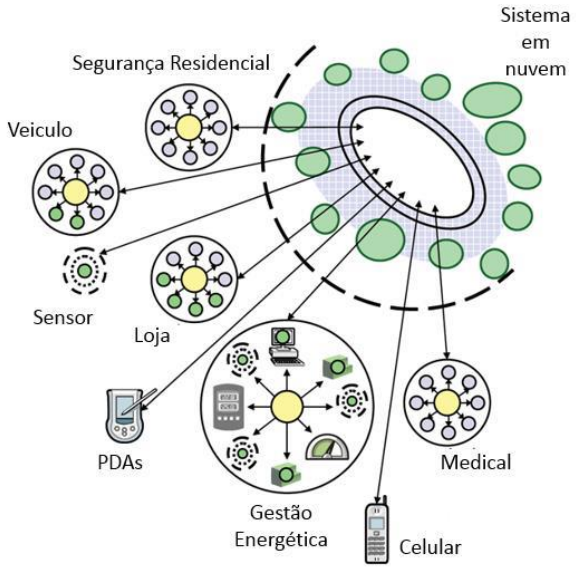
O *Message Queue Telemetry Transport*, como seu próprio nome diz, tem como principal propósito a telemetria, ou monitoramento remoto. O seu objetivo é coletar informações de diversos dispositivos e transportar tal informação para uma infraestrutura central. Normalmente, esse protocolo é utilizado para conectar grandes redes de pequenos dispositivos que serão monitorados ou controlados por um serviço em nuvem [Bauer et al. 2014].

Seguindo a arquitetura do MQTT, os dispositivos devem conectar-se a um servidor capaz de concentrar todas as informações. Comumente, o dispositivo é responsável por enviar (*publish*) as informações ao servidor, que opera como intermediário (*broker*), tendo conhecimento dos clientes que estão interessados nas informações enviadas (*subscribes*).

Visto que é desejável não perder informações, o protocolo opera sobre o *Transmission Control Protocol* (TCP), que oferece um *stream* simples e confiável, embora exista uma variação do MQTT utilizando *User Datagram Protocol* (UDP). Dentro do contexto desse protocolo, os dispositivos (*publishers*) são clientes de uma conexão TCP com o intermediário (*broker*), o qual mantem, também, conexões com os interessados nas informações (*subscribers*).

A utilização do MQTT permite o desenvolvimento de diversas aplicações, desde o monitoramento de vazamentos no meio industrial até dispositivos de segurança residencial. Qualquer outra aplicação em que a necessidade fundamental é dividir informações coletadas de diversas fontes e torná-la disponível ao usuário pode se beneficiar do uso do MQTT, como as aplicações exemplificadas na Figura 2.6.

Figura 2.6 – Usos do MQTT



Fonte: Bauer et al. 2014

O *Extensible Messaging and Presence Protocol* (XMPP) foi inicialmente desenvolvido como um protocolo para mensagem instantânea (IM) para conectar pessoas através de mensagens de texto. Esse protocolo utiliza *Extensible Markup Language* (XML) para formatação de texto e é executado sobre o TCP, assim como o MQTT. Seu ponto chave é

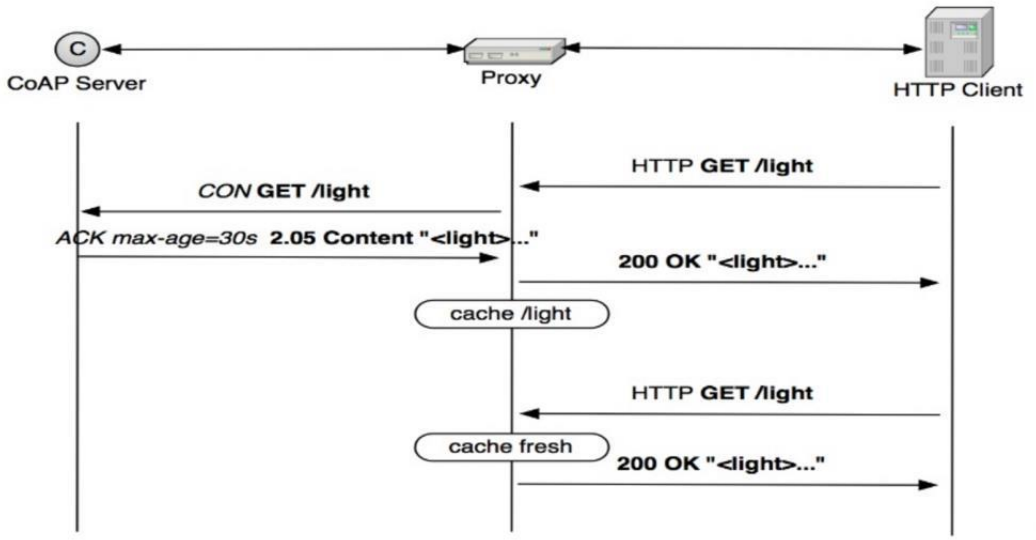
a utilização de *name@domain.com* como esquema de endereçamento, oferecendo assim um jeito fácil de endereçar um dispositivo IoT. Em suma, o XMPP é um ótimo jeito de conectar, por exemplo, o termostato da sua casa com um servidor Web que você pode acessar do seu telefone.

Há protocolos de IoT que não são apenas baseados no modelo *publish/subscribe*, como o *Constrained Application Protocol* (CoAP) que utiliza um modelo *request/response* e é também é bastante utilizado entre as soluções de IoT atualmente.

O CoAP um protocolo de transferência de documentos, mas diferentemente do *HyperText Transfer Protocol* (HTTP), ele foi projetado para atender dispositivos com capacidades restritas. Seus pacotes são muito menores e mais simples que os do HTTP, sendo de fácil interpretação e exigindo baixo consumo de memória. O protocolo executa sobre o *User Datagram Protocol* (UDP), ao invés do TCP, o que requer, se necessário, que um controle de duplicação e ordenamento seja implementado nas camadas superiores [Shelby 2014].

No modelo utilizado pelo CoAP, o cliente manda solicitações ao servidor, que as responde. Operações como GET, PUT, POST e DELETE normalmente são oferecidas nesse protocolo, junto com a interoperabilidade com HTTP e RESTful APIs (*Application Programming Interface*). Vale salientar que no escopo do UDP, os transdutores operam como servidores CoAP, ou seja, permitem que um cliente faça operações de GET, caso queira saber o estado de algum sensor daquele dispositivo. Um exemplo da troca de mensagens desse protocolo é apresentada na Figura 2.7, em que um sistema de controle de luminosidade, acessível através de um *proxy*, é exemplificado.

Figura 2.7 – Mensagens CoAP



Fonte: Shelby 2014

Na Figura 2.7, temos um transdutor (sensor de luminosidade), representado como um servidor CoAP, o qual contêm informações sobre a luminosidade. Após uma solicitação do tipo GET, o *proxy* recebe a informação da luminosidade com a possibilidade de mantê-la em cache por até 30 segundos (para esse exemplo), e então difunde essa informação ao cliente HTTP que havia solicitado inicialmente.

2.5 Middlewares IoT

O desenvolvimento de aplicações IoT por programadores é facilitada pela existência de uma gama de *middlewares*. Um *middleware* oferece uma camada de abstração entre a infraestrutura de rede e as aplicações em si. Essa camada ajuda a esconder detalhes tecnológicos, deixando o foco do desenvolvimento nas aplicações em si. Sendo assim, um *middleware* deve prover capacidade para lidar com: (i) interoperabilidade; (ii) gerenciamento de dispositivos; (iii) ciência de contexto; (iv) escalabilidade. [Chaqfeh et al. 2012].

Um primeiro requisito a ser endereçado para IoT diz respeito a prover interoperabilidade entre os diversos dispositivos e plataformas disponíveis no ambiente. Esse é um dos grandes desafios para a concretização do paradigma de IoT, devido ao grande número de dispositivos a serem integrados e sua heterogeneidade tanto em termos de hardware quanto de software. Além disso, é necessário prover mecanismos para o gerenciamento de dispositivos, como estado do dispositivo e localização permitindo, além disso, desconectar algum dispositivo não reconhecido, modificar remotamente configurações de hardware e software e configurações de segurança.

Ciência de contexto é outro requisito importante para plataformas de *middleware* para IoT. As informações de contexto, tais como o estado do objeto, seus vizinhos e sua localização, por exemplo, necessitam ser coletadas e processadas com o objetivo de efetuar ações ou reagir a estímulos com base nos dados extraídos [Perera et al. 2014]. Plataformas de *middleware* em IoT devem ser responsáveis pela coleta e processamento de informações de contexto, liberando as aplicações e usuários desta tarefa.

Considerando o amplo potencial de IoT, a consultoria americana Gartner, Inc. prevê que bilhões de dispositivos estarão aptos a serem utilizados por aplicações em curto prazo de tempo [Gartner 2014]. Sendo assim, uma plataforma de *middleware* para IoT deve dar suporte à escalabilidade, ou seja, deve ser capaz de assimilar um número crescente de dispositivos e requisições, sem deixar de funcionar corretamente.

A principal implementação do CoAP é o *Californium*, código aberto em Java, cuja utilização é bastante difundida. Entretanto, há também implementações em outras linguagens, tais como a libcoap em C, CoAP.NET em C# e node-coap em JavaScript.

Por outro lado, o MQTT é suportado pelo projeto *Eclipse Paho*, contando com clientes e outras bibliotecas para diferentes linguagens de programação como C, Java, Javascript e Python. No caso da necessidade de um servidor MQTT, a implementação mais difundida é o *Mosquitto* [Mosquitto 2013].

2.6 Considerações finais

Neste capítulo, foram apresentados os principais conceitos do paradigma da Internet das Coisas. Foram abordadas as principais tecnologias envolvidas e os domínios nos quais elas são úteis para IoT. Os conceitos vistos neste capítulo são empregados na sequência para definir uma proposta de arquitetura baseada em IoT para o sistema de irrigação doméstico.

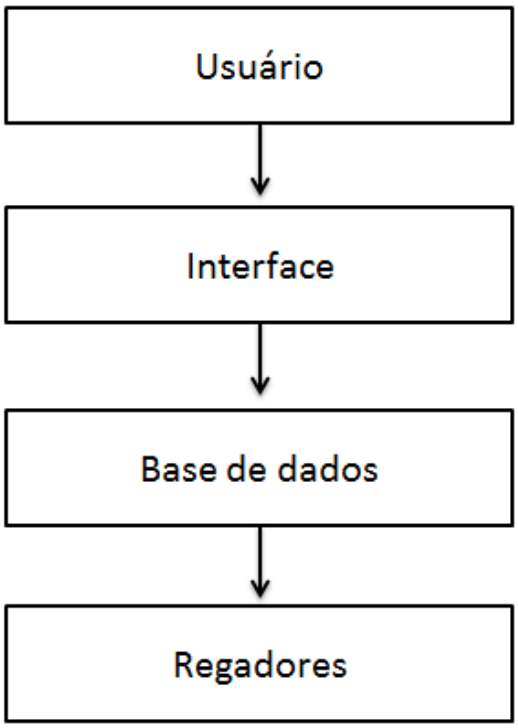
3 SISTEMA DE IRRIGAÇÃO DOMÉSTICO

Este capítulo apresenta a especificação do projeto proposto, começando pelo próposito do projeto, passando para seus requisitos e interfaces.

3.1 Descrição do sistema

A proposta deste projeto é de implementar um sistema de irrigação doméstico baseado nos conceitos da IoT. Esse sistema pode ser composto de diversos regadores, e basicamente um regador será composto por um conjunto de sensores e atuadores, responsáveis por medir a umidade do solo e controlar a liberação de água, respectivamente. Tal regador deverá ser capaz de informar a um sistema na nuvem sobre o valor da umidade do solo e de controlar a liberação da água. O usuário final, a partir de um aplicativo móvel, pode visualizar as informações referentes à umidade do solo, e configurar o nível de umidade em que o sistema deverá liberar àgua. Desse modo, o usuário final poderá controlar remotamente a irrigação de suas plantas, assim como monitorá-las. Para ter uma visão geral sobre o sistema, a Figura 3.1 mostra a arquitetura dos principais módulos do sistema.

Figura 3.1 – Sistema proposto



Fonte: Autor

3.2 Engenharia de requisitos

Ao longo de um processo de desenvolvimento de software ou sistemas para computação, o levantamento de requisitos é uma etapa muito importante. Basicamente, os requisitos são descrições dos serviços propostos pelo sistema com suas restrições operacionais [Sommerville 2007]. Os requisitos do sistema são comumente classificados em duas classes: requisitos funcionais e requisitos não funcionais.

Os requisitos funcionais definem como o sistema deve se comportar, o que deve fazer, e quais os recursos providos. No projeto proposto por este trabalho, temos:

- O sistema deve permitir a leitura da umidade do solo de determinada planta remotamente via uma interface de aplicativo móvel.
- O sistema deve permitir a configuração do nível de umidade em que a liberação da água será feita para uma planta.
- O sistema deve ser capaz de, sem a intervenção direta do usuário, acionar a liberação da água para uma planta no momento em que a sua umidade do solo atingir a configuração salva.
- O sistema deve permitir o acompanhamento dos momentos em que o acionamento da água foi feito.

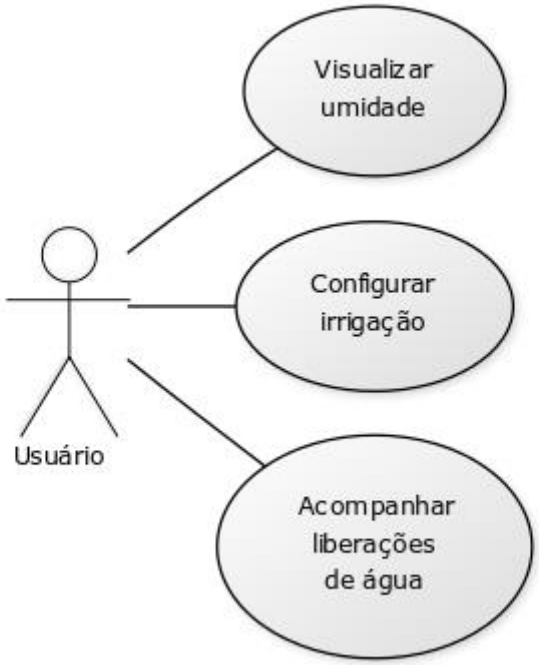
Por outro lado, temos os requisitos não-funcionais, também chamados de atributos de qualidade do sistema, os quais estão associados propriedades como disponibilidade, custo, tempo de resposta, confiabilidade e consumo [Sommerville 2007]. No projeto proposto, eles são:

- O sistema deve ser capaz de executar em celulares *Android*.
- O sistema deve ser intuitivo e amigável, sem exigir treinamento prévio do usuário final.
- O sistema deve empregar protocolos e tecnologias comumente utilizadas em aplicações IoT.
- O sistema deve possuir *hardware* de baixo custo e empregar software livre.

3.3 Diagrama de casos de uso

Os casos de uso de um sistema são muitos importantes para o projeto de um software, pois identificam quais as interações ocorrem no sistema e os agentes envolvidos. A documentação dos casos de uso pode ser feita através de texto puro ou da linguagem UML (*Unified Modeling Language*) que oferece diversos diagramas, entre eles, o diagrama de casos de uso. Um diagrama de casos de uso UML especifica um conjunto de funcionalidades necessárias ao sistema, assim como os agentes externos que integeram, chamados de “atores” [Sommerville 2007]. A Figura 3.2 apresenta o diagrama de casos de uso para o projeto proposto por este trabalho, baseado nos requisitos funcionais já apresentados.

Figura 3.2 – Diagrama de casos de uso



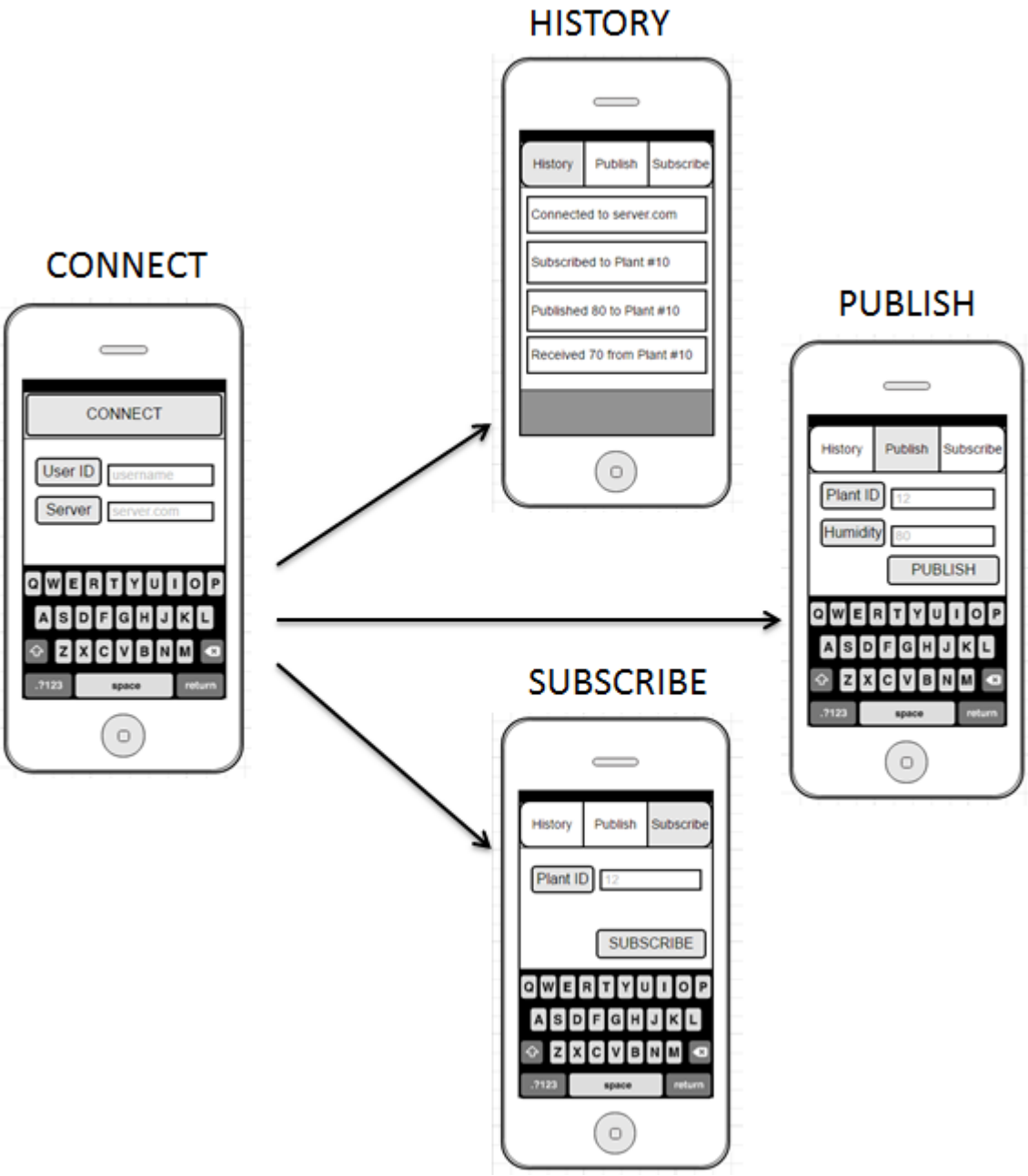
Fonte: Autor

3.4 Protótipos das interfaces

Os protótipos apresentados ilustram a interface do sistema, ou seja, como as informações são inseridas pelos usuários e recuperadas do sistema. Basicamente, há atividades relacionadas a administração do sistema, tal como ativar o monitoramento das plantas existentes e configuração do nível da irrigação para cada planta, e atividades

relacionadas ao usuário final do sistema, que seria o monitoramento da umidade e momentos de irrigação de suas plantas, oferecidas através de histórico. A Figura 3.3 mostra o protópico das telas do sistema, iniciando pela tela de *connect*, ponto de entrada no sistema, onde o usuário faz a conexão com a base de dados, previamente configurada com os usuários existentes, passando o seu identificador (ID) e o *hostname* da base de dados. Após inserir essas informações e pressionando o botão CONNECT, o usuário terá a sua conexão com a base de dados estabelecida.

Figura 3.3 – Protótipo - Telas



Fonte: Autor

Em seguida, o usuário terá acesso as telas de *history*, *publish* e *subscribe*. Na tela de *subscribe*, onde o usuário poderá registrar a sua intenção de receber informações de umidade de determinada planta, contida no sistema central. Esse registro ocorrerá através de um tópico MQTT, que será detalhado posteriormente no Capítulo 4. O usuário terá apenas que inserir o número do regador da planta desejada e pressionar o botão SUBSCRIBE.

Em seguida, o usuário terá que configurar o valor da umidade que ativará a liberação de água, através da tela de *publish*. O usuário terá apenas que inserir o número do regador da planta a ser configurada, e passar o valor limite da umidade antes da água ser liberada. Para completar a operação, o botão PUBLISH deve ser pressionado.

Para finalizar, o usuário poderá então visualizar todas as mensagens recebidas através da tela de *history*, possibilitando o monitoramento das ações do sistema e da umidade das plantas, conforme o protótipo apresentado na Figura 3.3.

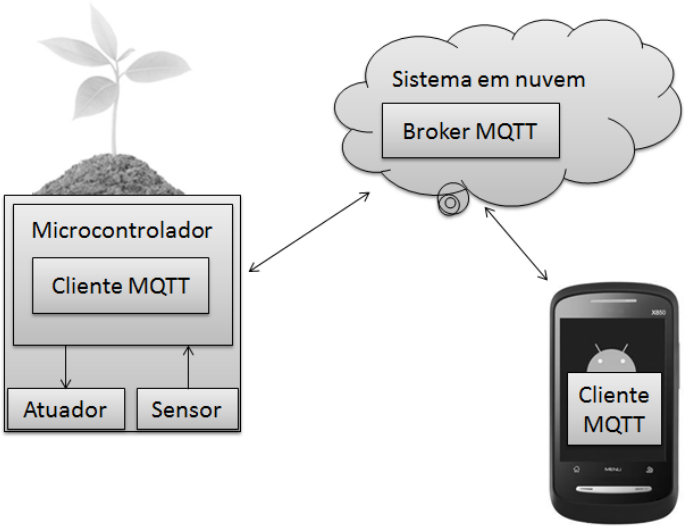
3.5 Considerações finais

Neste capitulo foi apresentada a especificação do sistema a ser implementado, usando as técnicas de engenharia de requisitos, diagrama de casos de uso e prototipagem de interfaces. Juntamente, foi apresentada a arquitetura da aplicação, que será implementada e detalhada tecnicamente no próximo capítulo.

4 DESENVOLVIMENTO E IMPLEMENTAÇÃO

Baseado no que foi discutido no Capítulo 3, a arquitetura do sistema possui componentes tanto de hardware quanto de software. Visto que o projeto proposto é baseado nos conceitos da Internet das Coisas, teremos que os dispositivos IoT serão os regadores. Esses regadores são capazes de sentir e atuar no ambiente em que se encontram, através de sensores e atuadores, respectivamente. Tais objetos utilizam um microcontrolador para se conectar em um serviço em nuvem, que permite ao usuário final da aplicação o monitoramento da umidade e configuração do regador através do seu aparelho celular. Para facilitar o entendimento da arquitetura do sistema, ela está apresentada na Figura 4.1.

Figura 4.1 – Arquitetura do sistema



Fonte: Autor

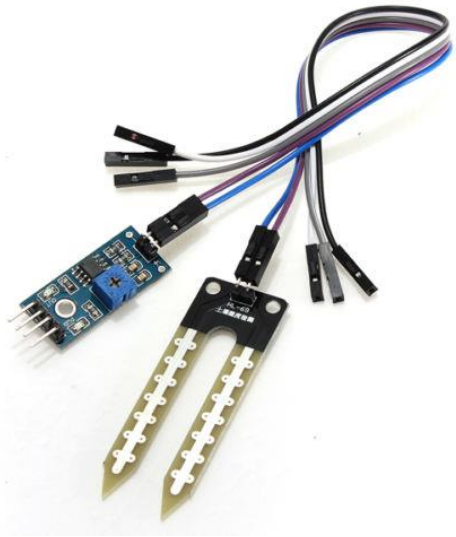
4.1 Dispositivo IoT

Com o objetivo de sentir e atuar no ambiente em que o sistema está inserido, foi utilizado um sensor de umidade de solo e uma válvula de solenoide (eletroválvula). Mais precisamente, o sensor de umidade de solo é responsável por determinar qual a umidade do solo onde a planta se encontra e a válvula de solenoide possui como função principal a liberação de água sobre o solo em que a planta está.

O sensor de umidade utilizado neste projeto é o conjunto sensor HL-69 – módulo HL-01 (Figura 4.2). O sensor HL-69 é um sensor passivo e intrusivo que deve ser fincado no solo próximo a planta e, através da medição de sua resistência, é possível obter a umidade do solo

onde ele se encontra. Tal medição de resistência e conversão é feita pelo módulo HL-01 que opera em conjunto com esse sensor passivo. O módulo é alimentado por uma tensão VCC de 3.3V e oferece duas saídas. A primeira delas é a saída digital, a qual passa de 0V para 3.3V (1 lógico) assim que o valor lido pelo sensor atinge o limite estabelecido manualmente através do potenciômetro embutido no módulo, conforme Figura 4.2.

Figura 4.2 – Sensor HL-69 e Módulo HL-01



Fonte: Autor

Embora essa saída digital seja fácil de utilizar, ela não é útil nesse projeto pois exige um ajuste manual do potenciômetro com o auxilio de uma chave de fenda. Nesse caso, o usuário teria que determinar o ponto ideal de umidade do solo através de tentativa e erro para diversos ajustes mecânicos neste sensor. Sendo assim, a saída utilizada por esse projeto é a analógica. O seu comportamento é simples, a tensão de saída varia de 0V (GND) a 3.3V (VCC) de acordo com a umidade, sendo 0V o valor máximo de umidade (idealmente 100% umidade relativa do ar) e 3.3V o valor mínimo de umidade (idealmente 0% umidade relativa do ar) em uma escala de precisão de 0.1%.

Tal conjunto sensor e módulo é encontrado facilmente à venda por valores entre 2 e 3 dólares, incluindo o transporte, em diversos sites de vendas da China. Sendo assim, esse sensor é a uma ótima opção dado a sua boa precisão e baixo custo.

A outra parte física do projeto é o conjunto de atuadores responsáveis pela liberação da água para a planta em determinados momentos.

Para tal função foi utilizado uma válvula de solenóide com entrada e saída 1/2” e alimentação de 12V. O funcionamento desse tipo de válvula é bastante simples, quando não estão alimentadas, se comportam como uma válvula fechada, e ao receberem a alimentação de 12V, trocam o seu estado para aberta enquanto tal alimentação permanecer. Especificamente, foi utilizada uma válvula plástica de 8cm (L) x 5cm (A), com partes metálicas em aço e vazão de operação variando de vazão mínima 7 litros/minuto até uma vazão máxima de 40 litros/minuto e temperatura máxima de operação de 60 graus. Tais especificações atendem tranquilamente as necessidades do projeto, já que a planta receberá uma quantidade pequena de àgua em temperatura ambiente, sem a necessidade de operação em temperaturas extremas ou em altas pressões/vazões. A válvula utilizada é apresentada na Figura 4.3, mostrando os seus dois terminais metálicos próximos a parte sinalizada e sua aparência geral.

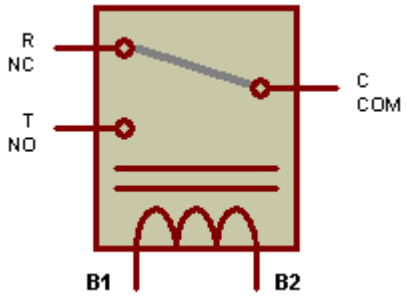
Figura 4.3 – Válvula de solenoide

Fonte: Autor

Para fazer uso dessa válvula, é preciso um sistema de comutação capaz de controlar a ativação da alimentação 12V. Sendo assim, fica evidente a necessidade de um módulo relé que, baseado em um pino de baixa tensão do microcontrolador, será responsável pela ativação da alimentação 12V da válvula, possibilitando o controle através do microcontrolador do acionamento. Um relé eletromecânico é um interruptor controlado por tensão. Internamente, ele possui duas partes, uma bobina e um conjunto de contatos elétricos. Comumente, quando a tensão de controle é igual a tensão de alimentação, a bobina é ativada, fazendo com que o contato elétrico altere o seu estado através do campo magnético induzido. A Figura 4.4 ilustra a estrutura interna de um relé comum. Apresenta-se duas opções de configuração, normalmente aberto (NO – *Normaly Opened*) e normalmente fechado (NC – *Normaly*

Closed). O contato comum se posiciona no NO quando a bobina está desligada, e chaveia para NC quando a bobina está alimentada.

Figura 4.4 – Relé



Fonte: Sonelec 2005

Nessa implementação foi utilizado um módulo com dois relés, embora apenas um seja de fato utilizado. Esse único relé está na configuração *Normaly Opened*, e é controlado por um dos pinos de saída do microcontrolador.

4.2 Microcontrolador

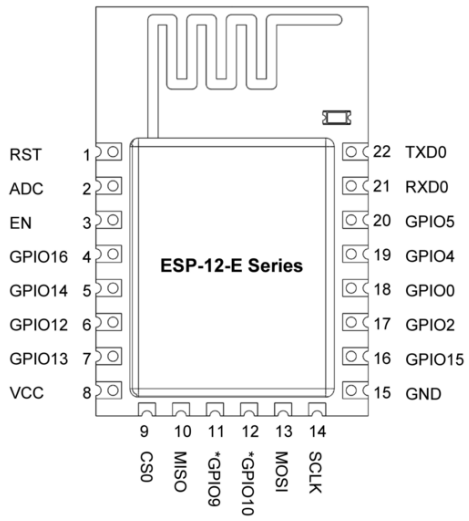
Através das seções anteriores fica claro como o funcionamento do regador é possível do ponto de vista físico. Entretanto, ainda falta detalhes de como o sinal analógico da umidade de solo e o controle da válvula de solenóide são feitos neste projeto.

Esse controle poderia ser facilmente realizado por alguma placa de desenvolvimento, como um Arduino, Raspberry ou Intel Galileo, porém o preço dessas placas pode variar de uma dezena de doláres até uma centena, no caso das placas mais elaboradas e com suporte embarcado para conexão sem fio. Portanto, buscou-se uma solução de baixo custo, ou seja, um microcontrolador não só com potencial para se conectar na rede sem fio doméstica do usuário, mas também com poder de processamento para suportar a lógica de negócio e um dos protocolos IoT descritos no Capítulo 2.

A alternativa mais recorrente utilizada por desenvolvedores domésticos de dispositivos IoT tanto em blogs quanto em fóruns foi o microcontrolador ESP8266, ilustrado pela Figura 4.5, fabricado pela chinesa Espressif.

Tal microcontrolador possui custo médio de 5 doláres, com frete incluido, nos mesmos sites de compras chineses usado como referência para o preço dos sensores.

Figura 4.5 – ESP8266



Fonte: Espressif 2015

O ESP8266 possui dimensões de 1.6cm (L) x 2.5cm (A) e é alimentado com 3.3V. Internamente, ele possui uma CPU-RISC (*Central Processing Unit - Reduced Instruction Set Computing*) 32-bit Tensilica Xtensa LX106 de 80MHz, 64KiB de RAM para instruções, 96KiB de RAM para dados, memória flash de 4MB, 16 pinos I/O de uso genérico, suporte para WiFI IEEE 802.11 b/g/n com WEP/WPA/WPA2, e um pino de conversor analógico-digital com precisão de 10 bits [Espressif 2015].

Felizmente, o fabricante disponibiliza um bom SDK (*Software Development Kit*) com diversos exemplos de usos interessantes do seu produto. Sendo assim, após a instalação do SDK e do *cross-compiler*, o desenvolvedor estará livre para desenvolver a sua própria expansão do firmware do microcontrolador, em ANSI-C, limitado apenas pelo tamanho do código, que deve caber dentro da memória RAM para instruções (64KiB).

Uma observação interessante a ser feita é que, graças ao suporte embarcado à WiFi, o nosso dispostivo IoT pode se conectar diretamente ao roteador doméstico do usuário final e comunicar-se com o sistema em nuvem, sem a necessidade de intermediários ou concentradores, que aumentaria o custo geral do projeto.

Através do SDK do ESP8266, somos capazes de realizar leituras do sensor de umidade, e ativar o pino de controle do relé que atua sobre a válvula de sonelóide para a liberação de água. Claro que essas operações não são tão simples, visto que o sensor de umidade envia um sinal analógico e o microcontrolador opera sobre dados digitais.

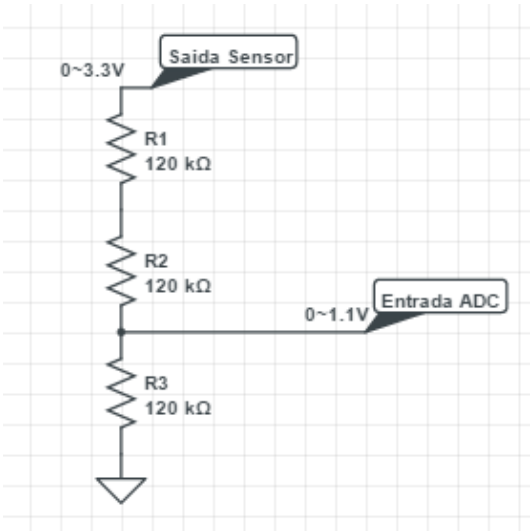
4.2.1 Conversor analógico-digital

Para fazer a conversão analógico-digital foi utilizado o conversor interno do ESP8266 acessado através do pino ADC. Esse conversor possui uma resolução de 10 bits, ou seja, é capaz de converter o valor de entrada em valores binários de 0000000000 (0 em decimal) até 1111111111 (1023 em decimal). Além disso, o sinal de entrada a ser convertido deverá variar de 0 a 1.1V, o que exige um ajuste visto que o sensor de umidade de solo pode retornar um valor de 0 a 3.3V.

Ao observar os valores, fica claro que podemos definir como sinal de entrada é justamente 1/3 do sinal de saída do sensor de umidade. Logo, um simples divisor de tensão resistivo é utilizado para essa adequação. Três resistores idealmente idênticos são ligados em série na saída do sensor de umidade, e a tensão sobre um deles é então a entrada para o conversor analógico-digital do ESP8266.

Para esse projeto, foi utilizado 3 resistores de 120k ohms, e a tensão do último deles é utilizada como entrada do ADC (*AD Converter*), como mostra a Figura 4.6.

Figura 4.6 – Divisor de tensão



Fonte: Autor

Para os pinos de saída, com o objetivo de controlar o relé e, posteriormente, a válvula de sonelóide, não foi necessário nenhuma adaptação, pois o relé opera com controle de 3.3V, que é justamente a tensão oferecida pelo ESP8266 nos seus pinos de saída quando o nível lógico 1 é definido através do seu *firmware*.

4.2.2 Firmware

O *firmware* desenvolvido neste projeto para o ESP8266 é composto de dois grandes módulos, o primeiro é a lógica de negócio e o segundo é o protocolo IoT implementado que permite ao microcontrolador comunicar-se com o sistema em nuvem.

A principal diferença entre os dois protocolos candidatos, CoAP e MQTT é o paradigma implementado. No CoAP, os objetos são servidores que recebem pedidos de leitura, através de chamadas GET, e deve estar aberto a chamadas PUT, onde os clientes (sistema em nuvem) podem configurar alguma variável interna do servidor CoAP (objeto IoT).

Se for considerado que o ESP8266 falaria diretamente com o sistema em nuvem, aspectos como *firewall* e CGNAT (*Carrier Grade Network Address Translation*) deveriam ser levados em conta e contornados para o correto funcionamento.

O CGNAT, também conhecido como LSN (*Large Scale NAT*) está se tornando um novo padrão entre os ISPs (*Internet Service Provider*). Inicialmente, o NAT tradicional era utilizado para traduzir IPv4s entre duas redes. Mais recentemente, NAT começou a ser utilizado para virtualizar os IPv4s de conexões empresariais, com o grande benefício de dividir um único IPv4 global público com diversos IPv4s locais privados. Devido ao esgotamento do IPv4 nos últimos anos, os provedores de Internet foram obrigados a usar NAT para atender a crescente demanda de seus clientes. Assim, enquanto a adoção do IPV6 não for completa, os usuários domésticos continuarão a utilizar endereços IPv4 privativos. Desta forma, fica muito complicado para usuários domésticos comuns terem qualquer tipo de servidor que possam aceitar conexões do mundo externo [A10Networks 2016].

Contornar o uso do NAT pelos provedores de Internet é algo que agregaria custo e complexidade para esse projeto, visto que a conexão CoAP na verdade teria que ocorrer através de um *proxy*, responsável pela inversão do cliente e servidor, em que o real servidor seria o sistema em nuvem.

Por outro lado, o uso do MQTT é simples, o Broker MQTT, que estaria localizado na nuvem, seria o servidor da conexão. Todos os clientes (objetos inteligentes e dispositivos móveis) apenas teriam que realizar conexões TCP, mantidas através de *keep-alives*, e toda troca de informações pertinentes ao protocolo seriam realizadas.

Além disso, o MQTT possui uma grande quantidade de implementações de clientes otimizados, implementados em ANSI-C, focados para sistemas embarcados como o ESP8266.

Para completar, as implementações de *broker* para o sistema em nuvem, e posteriormente exemplos de aplicações móveis também são largamente difundidas em foruns especializados.

Sendo assim, neste projeto foi utilizado a implementação do MQTT Paho para operar sobre o firmware, modificado para este projeto, do ESP8266, ao invés do padrão POSIX Windows [Paho 2015].

Do ponto de vista de lógica de negócio, a solução implementada possui duas variáveis principais e tempos associados a essas variáveis, obtidos através de testes experimentais. O nível de umidade limite para que a água seja acionada, configurada através do *publish* em um tópico MQTT especifico, e o valor da umidade do solo, lido uma vez por minuto do conversor analógico digital e disponível ao usuário através do *subscribe* de outro tópico MQTT.

Essas duas variáveis são comparadas constantemente, a cada um minuto, e no momento em que o solo fica mais seco do que a configuração permite, o pino de controle é acionado, levando o relé a alimentar a válvula de sonelóide, que libera água corrente no solo durante 3 segundos, o que corresponde a cerca de 50ml.

Desse modo, o solo terá a sua umidade relativa elevada, e na próxima vez que as variáveis configuração e umidade forem comparadas, não será necessário liberar mais água. Tal operação só se fará necessária após algumas horas, quando a planta tiver feito uso da água liberada anteriormente.

4.3 Broker MQTT e sistema em nuvem

Baseado no modelo de arquitetura apresentado, o sistema em nuvem é o módulo responsável por hospedar o *broker* MQTT do sistema como um todo. Neste projeto, o sistema em nuvem está hospedado na *Amazon Web Services* (AWS), que oferece uma instância t2.micro gratuitamente durante um ano, sendo assim, sem agregar nenhum custo ao projeto.

Uma instância t2.micro conta com 1GB de memória, 1 vCPU e 6 créditos de CPU/hora. Embora o sistema de créditos e cobrança de serviços em nuvem seja algo bem complexo, podemos simplificar dizendo que os 6 créditos representam o uso da CPU virtual dessa instância em 100% durante 6 minutos para cada hora. Comparado ao uso de computadores domésticos, pode-se parecer que isso é muito limitado, entretanto, o projeto conta apenas com um Ubuntu 14.04.3 LTS (Linux Kernel 3.13.0-74), dedicado exclusivamente a executar o *broker* MQTT escolhido, no caso, o Mosquitto.

O Mosquitto [Mosquitto 2013] é um *middleware* orientado a comunicação, desenvolvido em software livre, e que provê um servidor MQTT leve em suas versões 3.1 e 3.1.1, podendo ser instalado no Ubuntu 14.04.3 através do comando *apt-get*. Posto isto, os tópicos MQTT envolvidos nas comunicações entre clientes e *broker* são:

- */<userid>/esp/<sensornumber>/humidity/:* Esse tópico possui o valor da umidade relativa captada pelo sensor representado em <sensornumber>, podendo variar de 0 até n-1, em que n é o número de regadores do sistema. Sendo então publicado pelo ESP8266, uma vez por minuto, toda vez que o mesmo realiza a leitura do conversor analógico digital. O conteúdo é um valor inteiro no intervalo 0-1023, em que 0 representa uma umidade relativa idealmente de 100%, enquanto 1023 representa uma umidade relativa idealmente de 0%. Os usuários finais podem realizar o *subscribe* desse tópico em seus aplicativos móveis para obter a leitura instantânea da umidade do solo para determinado objeto IoT (regador).

- */<userid>/esp/<sensornumber>/watertrigger/:* Esse tópico possui o valor de umidade limite, ou seja, a configuração de qual é a umidade que ao ser atingida, deve acionar a liberação da água para a planta. O seu conteúdo também é um valor inteiro no mesmo intervalo da umidade do solo, e deve ser configurado através de uma operação *publish* do usuário final através do seu dispositivo móvel. O objeto IoT, representado pelo ESP8266 com seus sensores e atuadores, realiza um *subscribe* desse tópico para possuir a configuração mais recente.

- */<userid>/esp/<sensornumber>/wateron/*: Esse tópico pode ser visto como um histórico dos acionamentos de água por parte do objeto IoT (regador). Toda vez que a água é liberada, o ESP8266 realiza um *publish* nesse tópico do valor de umidade junto com a configuração de liberação de água, para assim, justificar o seu acionamento. O usuário final, através do seu aplicativo móvel, pode fazer *subscribe* nesse tópico para ter o controle de quando e em que momento o acionamento da água foi feito, junto com as leituras de umidade.

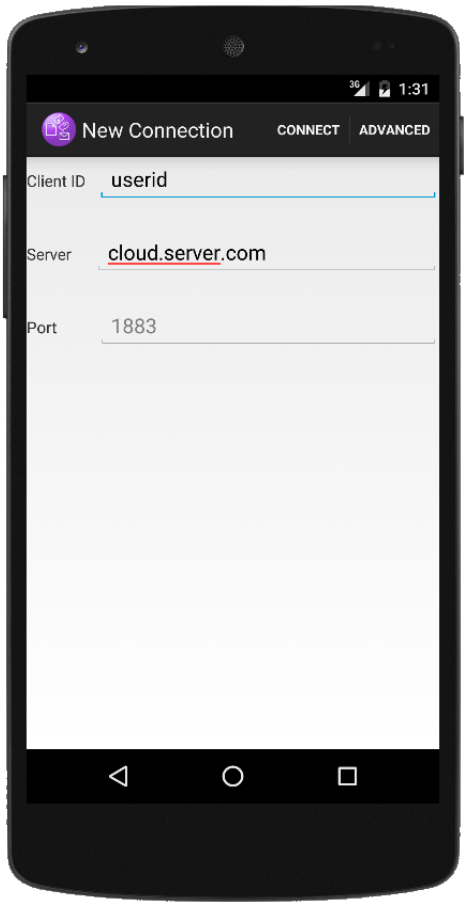
4.4 Aplicação móvel

O aplicativo móvel é a implementação pública do cliente MQTT Paho, e é utilizado nesse projeto como uma prova de conceito das interfaces ideais apresentadas no Capítulo 3. Portanto, o aplicativo móvel é basicamente um cliente MQTT que se conecta ao servidor em nuvem para obter ou enviar as informações referentes aos tópicos apresentados na Seção 4.3.

Inicialmente, o usuário deve fazer a configuração do sistema, ou seja, demonstrar interesse por monitorar a umidade de cada uma de suas plantas, assim como definir o nível de umidade que ativará a irrigação individualmente. Tais operações de administração são possíveis através das telas de *subscribe* e *publish*, enquanto o monitoramento da umidade poderá ser feita pelo usuário final através da tela de *history*.

A Figura 4.7 mostra a interface de *login*, onde o usuário faz a conexão com o *broker* MQTT hospedado na AWS, utilizando a porta TCP 1883 que corresponde ao protocolo MQTT, junto com o seu usuário (Client ID) e o *hostname* do servidor, que estará previamente configurado para operar como *broker* MQTT.

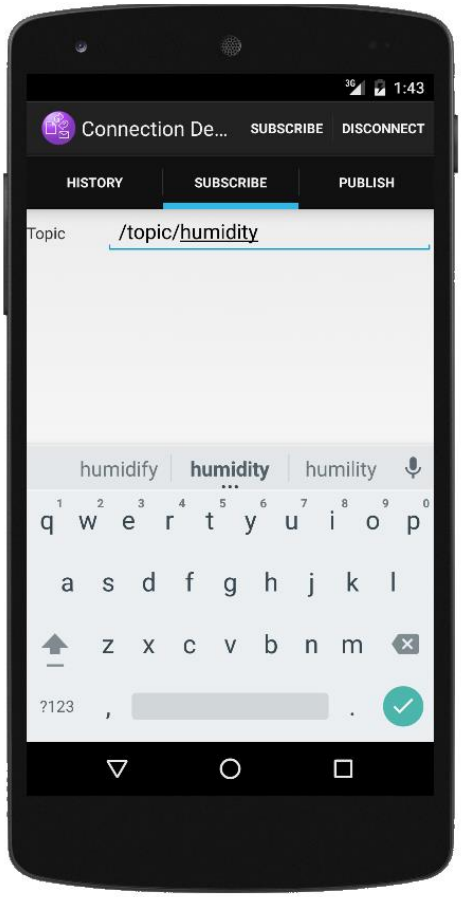
Figura 4.7 – Login



Fonte: Autor

Para poder fazer operações de *publish* e *subscribe*, o usuário terá acesso a outras duas telas. A Figura 4.8 mostra a interface de *subscribe*, onde o usuário registra a sua intenção de receber o valor da umidade contido no *broker* MQTT, através do caminho para o tópico MQTT que representa tal umidade. Vale salientar que no momento em que o usuário realiza o *subscribe* em um tópico previamente conhecido neste protótipo, o mesmo é criado na base de dados do servidor.

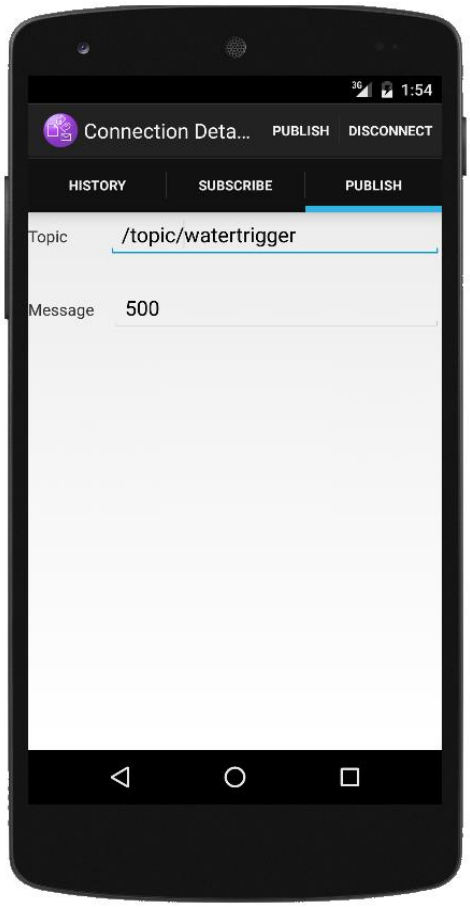
Figura 4.8 –*Subscribe*



Fonte: Autor

Para completar a configuração inicial, o usuário tem também que realizar o *publish* da configuração de umidade desejada através da interface de *publish* (Figura 4.9). Com o caminho do tópico de configuração (nesse projeto, para o primeiro sensor, */grehs/esp/0/watertrigger/*), e o valor limite da umidade (número inteiro no intervalo 0-1023) no campo *Message*, a configuração já pode ser realizada.

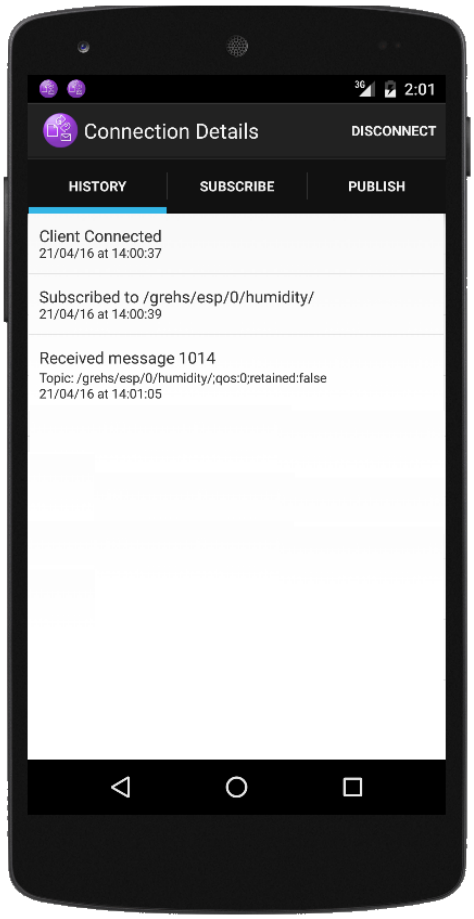
Figura 4.9 –*Publish*



Fonte: Autor

O usuário poderá então visualizar todas as mensagens recebidas (leituras de umidade) através da tela de *history*, como mostra a Figura 4.10, que ilustra como as atividades do sistema são exibidas para o usuário através dessa tela, exemplificando através do recebimento da umidade de uma planta, mostrado ao lado da frase *Received Message* (no exemplo, o valor de 1014 é recebido do primeiro sensor).

Figura 4.10 – H*istory*



Fonte: Autor

4.5 Dificuldades encontradas

A maior dificuldade encontrada neste projeto foi documentação do microcontrolador ESP8266 para a utilização de recursos com o conversor analógico digital embarcado e o controle dos pinos de saída. A Espressif, desde o começo deste trabalho, em agosto de 2015, lançou cerca de 4 versões da SDK de suporte ao *firmware* padrão, e dentro delas, há minúcias e detalhes específicos das versões físicas do ESP8266, que vem sendo desenvolvido a alguns anos e, atualmente, está na 14ª versão. Dito isto, fica claro que qualquer documentação

existente possui grande chance de estar desatualizada, ou incoerente, com as novas versões de SDK e chips lançados pela fabricante. Embora haja grande suporte para o desenvolvimento de dispositivos IoT utilizando o ESP8266, a quantidade não é comparável ao suporte que plataformas como Arduino, Raspberry e Galileo possuem.

4.5 Considerações finais

Neste capítulo foi feita uma abordagem às ferramentes empregadas para a realização do sistema de domótica proposto no Capítulo 3. Como um dos objetivos deste projeto é ser baixo custo, tal balanço é de grande relevância, e pode ser observado através da Tabela 4.1.

Tabela 4.1 – Custos do projeto (1 USD = 3,50 BRL)

Componente	*Preço (USD)*	*Preço (BRL)*
Sensor de umidade do solo	$ 2,00	R$ 7,00
Protoboard	-	R$ 8,00
Válvula de solenóide	-	R$ 32,00
Relés	-	R$ 20,00
ESP8266	$ 5,00	R$ 17,50
Fontes de alimentação	-	Reciclagem
Total	-	R$ 84,50

Fonte: Autor

Vale salientar que muitas vezes o custo de desenvolvimento de software não é zero, já que os softwares livres utilizados tiveram que ser adaptados e outros implementados inteiramente. Entretanto, por falta de uma métrica coerente para medir esse custo, o preço obtido através da Tabela 4.1 é apenas da parte física do projeto, considerando apenas um regador para uma única planta. Uma solução futura com diversos sensores em diversas plantas, teria um custo de R$ 84,50 vezes o número de plantas desejado.

5 AVALIAÇÃO E TESTES

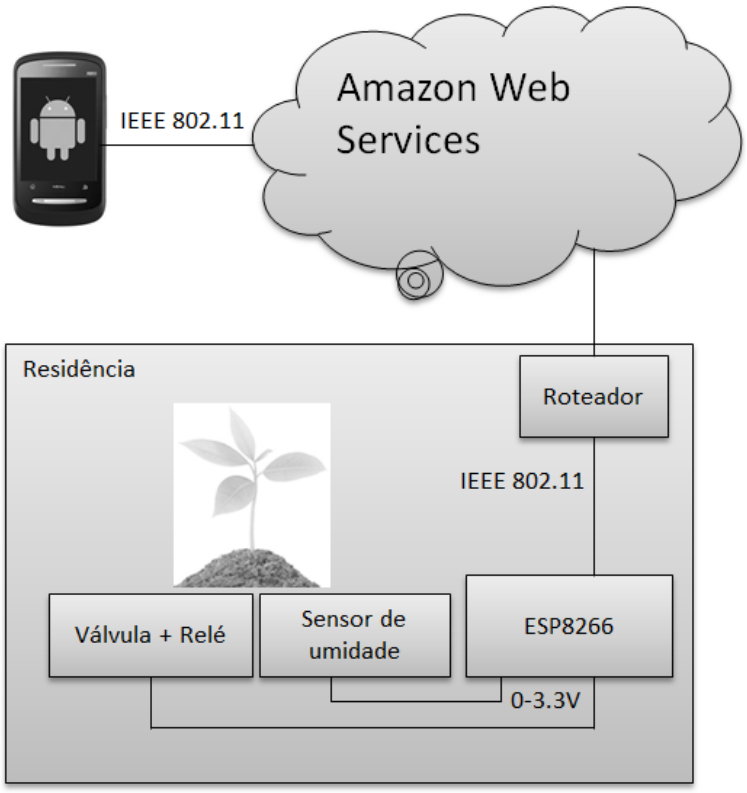
Para avaliar e validar o protótipo desenvolvido no Capítulo 4, este capítulo aborda os experimentos realizados e a discussão dos resultados obtidos. Inicialmente é apresentada a plataforma experimental, para então discutir a metodologia utilizada na realização dos experimentos e os resultados obtidos. Por fim, conclui-se este capítulo com algumas considerações gerais.

5.1 Plataforma experimental

A plataforma experimental escolhida para realizar os experimentos é composta por uma rede doméstica, sistema em nuvem na rede da AWS e um dispositivo móvel com 4G, conforme a Figura 5.1.

Nesse experimento, há apenas um regador (dispositivo IoT) composto pelo conjunto de transdutores e ESP8266, em que o sensor de umidade está inserido no solo de um pequeno vaso, e a saída da válvula de água apontada para o mesmo solo. Esse regador é alimentado por tomadas elétricas e possui acesso à Internet através do ponto de acesso *wireless* da residência.

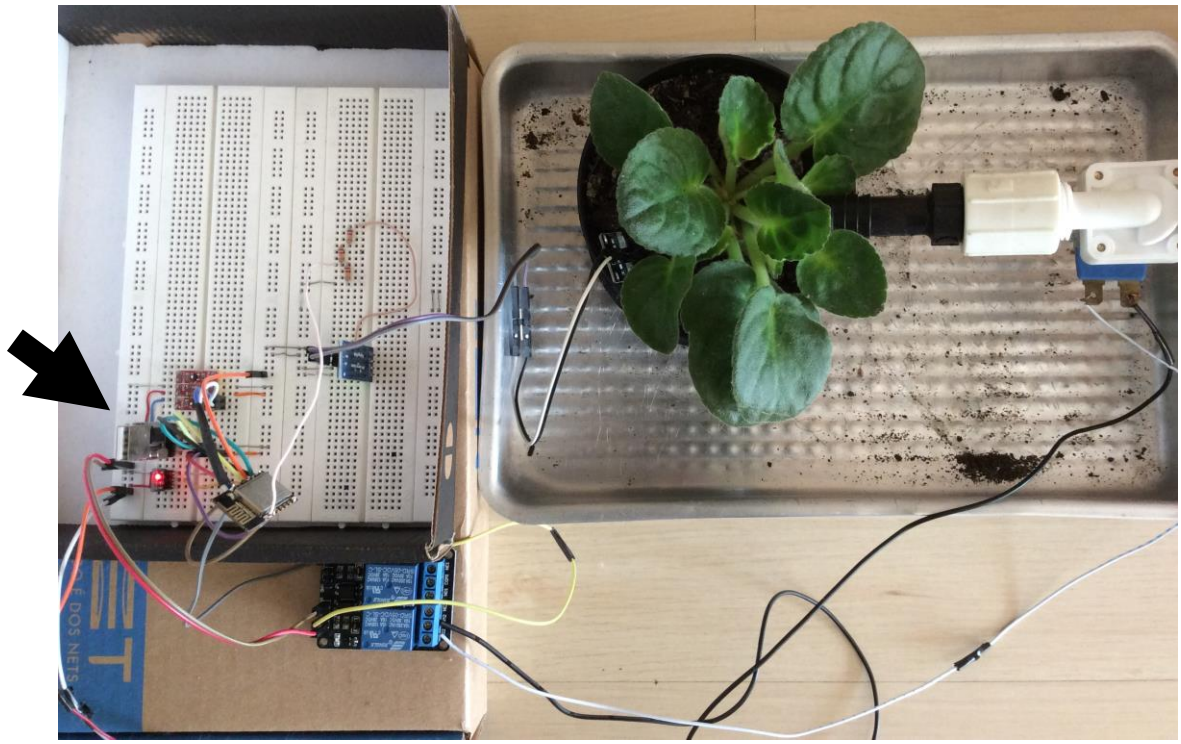
Figura 5.1 – Plataforma experimental



Fonte: Autor

A montagem do protótipo é ilustrada na Figura 5.2, em que o regador também conta com uma comunicação serial ligada ao ESP8266 (sinalizada pela seta), para permitir a gravação de *firmware* e console de *debug*.

Figura 5.2 – Protótipo



Fonte: Autor

5.2 Metodologia

Para validar o sistema de irrigação, dois cenários foram necessários. O primeiro para validar o sensor de umidade, e o segundo para validar o acionamento (atuação) da irrigação.

No sensor de umidade, foi conduzido experimentos em três tipos diferentes de solo. Vale salientar que nesses experimentos foi percebido a lenta variação de umidade do solo, chegando assim a conclusão que uma leitura da umidade por minuto seria uma periodicidade satisfatória.

Dessa forma, foram criados três testes:

- **Solo Seco:** Solo exposto ao sol, sem receber água, durante diversos dias.
- **Solo Úmido:** Solo exposto à luz do dia e regado uma vez por dia.
- **Solo Molhado:** Solo não exposto ao sol ou à luz do dia, e regado em abundância.

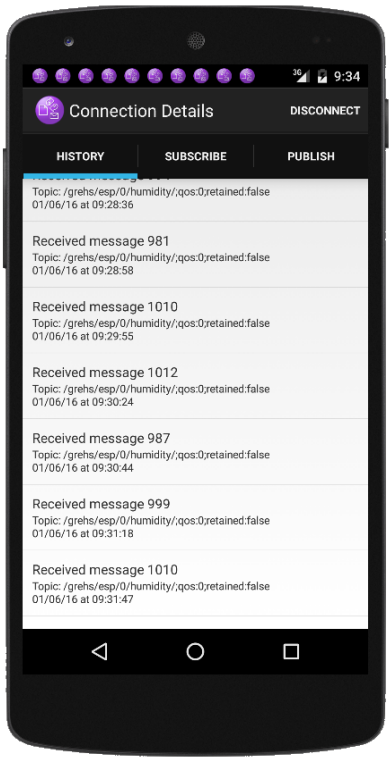
Para cada um desses testes, foi observado o valor da umidade. Para validar o funcionamento do sistema, seria necessário verificar o acionamento de água baseado nas

variáveis *humidity* e *watertrigger* descritas no Capítulo 4, portanto foi criado um quarto teste. Ainda, para contemplar o requisito de manter histórico e também comprovar a robustez do sistema, a plataforma experimental foi monitorada por 3 dias.

5.3 Resultados

Idealmente, um solo totalmente seco, ou seja, com umidade próxima a 0%, deve ter o valor de umidade nesse sistema próximo a 1023. Sabendo disso, a Figura 5.3 mostra o comportamento do sistema nesse cenário, observado durante alguns minutos, através do simulador do aplicativo móvel.

Figura 5.3 – Teste 1



Fonte: Autor

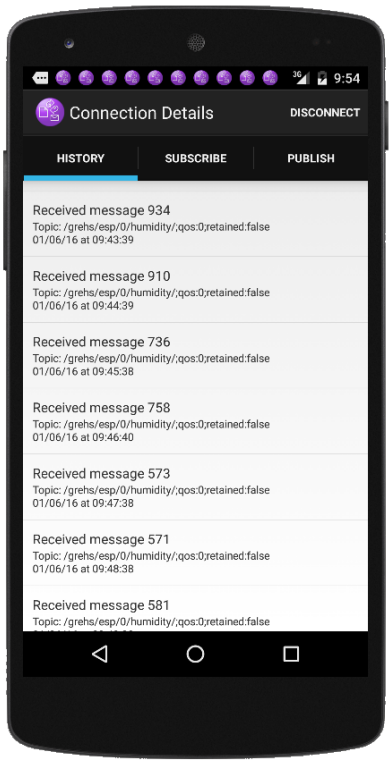
Fica claro que devido a imprecisões do sensor, do solo possivelmente não estar totalmente seco, e do conversor analógico-digital, o valor obtido não é totalmente estável, pois varia entre 990 e 1010 (valor informado ao lado das frases *Received Message* na Figura 5.3), mas ele atinge as expectativas de proximidade ao comportamento ideal descrito nesta seção.

Aproveitando esse teste com solo seco, pode-se chegar a conclusão de qual a periodicidade da amostragem necessária devido a propagação da água e a sua correspondência

no sensor de umidade. Para isso, o solo recebeu manualmente uma quantidade de 50ml de água, de modo a torná-lo úmido, como o segundo teste sugere.

No segundo teste, um solo úmido, com umidade em torno de 40-50%, deve ter o seu valor no sistema em torno de 500-600, considerando um comportamento linear do sensor de umidade, cuja validade não pôde ser comprovada neste trabalho por falta de documentação.

Figura 5.4 – Teste 2

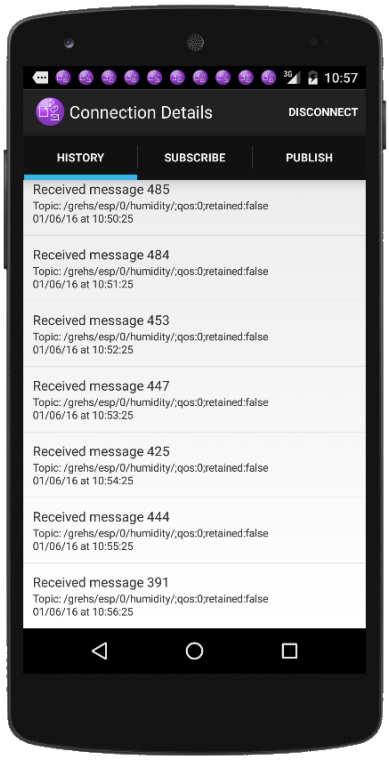


Fonte: Autor

Observa-se na Figura 5.4 que após o recebimento de água, o solo demorou cerca de 7 minutos para de fato chegar ao patamar de umidade esperado, entre 500-600, confirmando assim as escalas de tempo definidas no Capítulo 4, um minuto entre cada leitura da umidade.

No terceiro teste, foi considerado um solo muito molhado, com uma umidade próxima a 70%, em que o valor esperado no sistema será em torno de 400. Para tal, o sensor foi movimentado para o solo com essa característica e monitorado através do aplicativo móvel, como mostra a Figura 5.5.

Figura 5.5 – Teste 3



Fonte: Autor

Baseado nesses três testes, fica claro que o sensor de umidade se comporta conforme esperado e o sistema é capaz de oferecer ao usuário o valor da umidade em um determinado regador dentro de uma precisão suficiente para a aplicação.

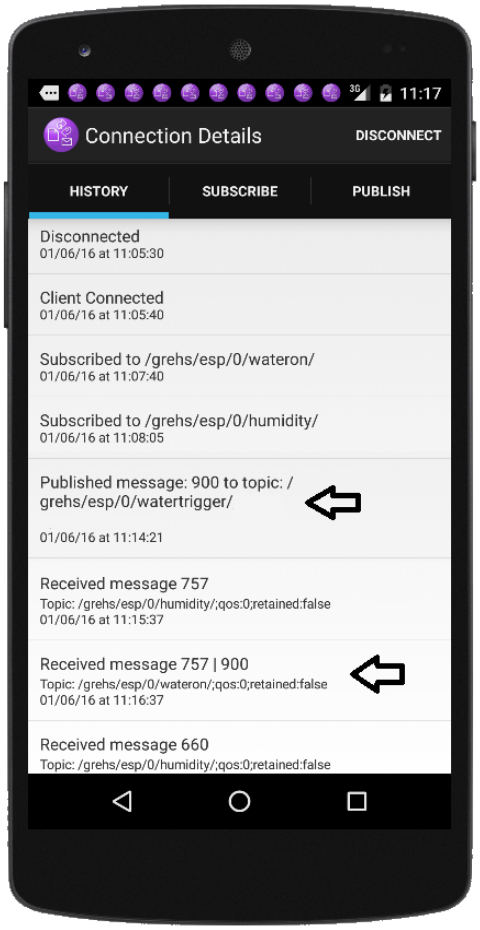
Entretanto, um quarto teste é necessário para validar de fato o sistema de irrigação, ou seja, se o controle da válvula de água é acionado no momento correto. Sendo assim, a configuração da variável *watertrigger* é necessária através dos tópicos MQTT do protótipo.

Para esse cenário, foi definido que o valor da umidade em que a água deveria ser acionada é 900 (operação marcada pela primeira seta na Figura 5.6), e o resultado deve ser mostrado no tópico *wateron (*operação marcada pela segunda seta na Figura 5.6), que somente envia dados quando a válvula for ativada, contendo o par *humidity* e *watertrigger*.

Na prática, essa configuração representa o desejo de liberar água para a planta no momento em que o percentual de umidade estiver abaixo do percentual representado pelo *watertrigger*. Visto que um solo seco é representado pelo valor de *humidity* superior a 700, a água só deve ser acionada (mensagem de *wateron* enviada) quando *watertrigger* for menor que *humidity*.

A confirmação de tal operação foi feita visualmente no protótipo, assim como no histórico do aplicativo móvel, embora o mesmo não tivesse que estar aberto para o correto funcionamento do sistema de irrigação.

Figura 5.6 – Teste 4



Fonte: Autor

Como mostra a Figura 5.6, o comportamento do sistema é exatamente o esperado, ou seja, no momento em que a *humidity* de 757 foi lida, o *watertrigger* no valor de 900 fez com que o tópico *wateron* informasse que a válvula foi de fato aberta.

5.4 Considerações finais

Posto todos os cenários de testes, foi possível comprovar o correto funcionamento do sistema proposto, em que o usuário é capaz de monitorar a umidade de seus regadores, assim como controlar quando eles receberão água. Embora o aplicativo móvel não tenha atendido inteiramente as especatitivas da proposta inicial (intuitivo e amigável), ele atende os propósitos deste trabalho como protótipo.

6 CONCLUSÃO

Ao longo deste trabalho foi abordado conceitos e tecnologias que formam o paradigma da Internet das Coisas. Por envolver muitos conceitos e tecnologias, nota-se que existe uma grande dificuldade em encontrar uma definição unificada sobre o termos e conceitos utilizados, devido a IoT ainda estar em desenvolvimento. Além deste estudo, foi proposto e implementado um sistema de irrigação doméstico, aplicação que encontra sua base nos conceitos da IoT.

O objetivo deste trabalho foi realizar um estudo sobre a Internet das Coisas e de implementar um sistema que permitisse controlar e monitorar remotamente um conjunto de regadores de plantas. Seu protótipo apresenta grande potencial para a área de automação residencial, apesar do requisito não-funcional pretendido neste trabalho ser o fácil uso pelo usuário final, os demais requisitos de funcionalidade propostos no trabalho foram atingidos com sucesso.

O ponto principal, que é controlar remotamente a irrigação de plantas, usando conceitos de Internet das Coisas foi satisfeito. Resta a parte de melhorias do protótipo para poder torná-lo um produto, estudando a melhor forma de se apresentar as informações. De qualquer forma, o protótipo desenvolvido se mostrou eficiente como prova de conceito e como forma de estudo para a Internet das Coisas.

A contribuição do trabalho vem do estudo dos conceitos da Internet das Coisas. Além de que ele guia a possibilidade de implementções de dispositivos IoT usando os microcontroladores da Espressif, sistema central em nuvem e baixo custo.

Este trabalho foi desenvolvido como uma prova de conceito para um sistema de irrigação doméstico, fazendo uso de hardware de baixo custo, além de software livre para soluções baseadas em Internet das Coisas. Como prova de conceito, estima-se que os resultados foram extremamente satisfatórios. Entretanto, durante o desenvolvimento surgiram algumas ideias não implementadas que podem ser consideradas como melhorias em implementações futuras, como:

- Estudar uma ferramenta de autenticação de usuário e adicioná-la ao sistema atual tornando mais seguro.
- Utilizar um conjunto de sensores de umidade, temperatura e luminosidade para determinar mais precisamente o momento em que a água será liberada para a planta.

- Adicionar uma interface mais amigável no aplicativo móvel para o usuário final, uma forma mais dinâmica de exibir e solicitar as informações, deixando assim abstraido os detalhes técnicos como protocolos e servidores utilizados.

Essas melhorias, entre outras, podem transformar esse protótipo em uma solução mais eficiente e robusta para a irrigação de plantas em uma residência.

REFERÊNCIAS

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, Elsevier, v. 54, n. 15, p. 2787–2805, 2010.

ASTHON, K. **That internet of things in the real world, things matter more than ideas**. RFID Journal, 2009

PINTO, F. D. M. **Desenvolvimento de um Protótipo de um Sistema Domótico**. Dissertation (Master) — Instituto Superior Técnico - Universidade Técnica de Lisboa, 2010.

ZHU, Q. et al. Iot gateway: **Bridging wireless sensor networks into internet of things**. In: IEEE. Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on. [S.l.], 2010. p. 347–352.

INFSO D.4 Networked Enterprise & RFID INFSO G.2 Micro & Nanosystems, in Co-operation with the Working Group RFID of the ETP EPOSS (2008), **Internet of Things in 2020**, Roadmap for the Future, Version 1.1.

GUINARD, D.. **Towards the web of things: web mashups for embedded devices**, in: Proceedings of the International World Wide Web Conference 2009 (WWW 2009), Madrid, Spain.

DUNKELS, A.; VASSEUR J.P. **IP for Smart Objects**, Internet Protocol for Smart Objects (IPSO) Alliance, White Paper #1 2009.

BANDYOPADHYAY, D.; SEN, J. **Internet of things: Applications and challenges in technology and standardization**. Wireless Personal Communications: An International Journal, 58 2011(1):49–69.

FLEISCH, E. **What is the internet of things? An economic perspective**. Technical report, AUTO-ID LABS 2010.

YANG, Z. et al. **Study and application on the architecture and key technologies for IoT**. In: IEEE. Multimedia Technology (ICMT), 2011 International Conference on. [S.l.], p. 747–751.

ZARGHAMI, S. **Middleware for Internet of things.** Dissertation (Master) — Faculty of Electrical Engineering, Mathematics and Computer Science Software Engineering - University of Twente 2013.

SHELBY, Z. **ARM IoT Tutorial: CoAP** - The Architecture for The Digital World 2014

CHAQFEH, M. A.; MOHAMED, N. **Challenges in middleware solutions for the internet of things**. In: IEEE. Collaboration Technologies and Systems (CTS), 2012 International Conference on. [S.l.], 2012. p. 21–26.

PERERA, C., ZASLAVSKY, A., CHRISTEN, P., GEORGAKOPOULOS, D.**Context aware computing for the Internet of Things**, IEEE Communications Surveys & Tutorials 2014, vol. 16, no. 1, pp. 414-454.

GARTNER: **Gartner says 4.9 billion connected 'things' will be used in 2015**. 2014. Disponível em: <http://www.gartner.com/newsroom/id/2905717>. Acesso em: Março 2016.

JACKSON, J. OASIS: **MQTT to be the protocol for the Internet of Things**. 2013. Disponível em: <http://www.pcworld.com/article/2036500/oasis-mqtt-to-be-the-protocol-for-the-internet-of-things.html>. Acesso em: Março 2016.

MOSQUITTO, P. **Mosquitto**. 2013. Disponível em: <https://www.eclipse.org/proposals/technology.mosquitto/>. Acesso em: Janeiro 2016.

SOMMERVILLE, I. **Engenharia de software. Tradução: Selma Shin Shimuzu Melnikoff, Reginaldo Arakaki, Edilson de Andrade Barbosa**. [S.l.]: São Paulo: Pearson Addison Wesley, 2007.

SONELEC. **Relais**. 2015. Disponível em: <http://www.sonelec-musique.com/electronique_theorie_relais.html>. Acesso em: Dezembro 2015.

ESPRESSIF. **Smart Connectivity Platform: ESP8266** 2015. Disponível em: <https://cdn-shop.adafruit.com/datasheets/ESP8266_Specifications_English.pdf>. Acesso em: Novembro 2015.

A10NETWORKS. **Carrier Grade Network Address Translation** 2016. Disponível em: <https://www.a10networks.com/products/carrier-grade-network-address-translation>. Acesso em: Março 2015.

PAHO, E. **Paho project**. 2015. Disponível em: <https://projects.eclipse.org/projects/iot.paho>. Acesso em: Janeiro 2016.