Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

**GUILHERME RODRIGUES DE MELO**

# ARQUITETURA APOIADA POR INTERNET OF THINGS E PROTOCOLO MQTT PARA SMART AGRICULTURE

**Assis/SP**
**2017**

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

**GUILHERME RODRIGUES DE MELO**

# ARQUITETURA APOIADA POR INTERNET OF THINGS E PROTOCOLO MQTT PARA SMART AGRICULTURE

Projeto de pesquisa apresentado ao curso de Ciência da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

**Orientando(a): Guilherme Rodrigues de Melo**
**Orientador(a): Prof. MSc. Guilherme de Cleva Farto**

**Assis/SP**
**2017**

FICHA CATALOGRÁFICA

MELO, Guilherme Rodrigues de.
**Arquitetura apoiada por Internet of Things e protocolo MQTT para Smart Agriculture** / Guilherme Rodrigues de Melo. Fundação Educacional do Município de Assis –FEMA – Assis, 2017.
52p.

1. Internet of Things. 2. MQTT. 3. Smart Agriculture. 4. Digital Farming.

CDD: 001.6
Biblioteca da FEMA

# ARQUITETURA APOIADA POR INTERNET OF THINGS E PROTOCOLO MQTT PARA SMART AGRICULTURE

GUILHERME RODRIGUES DE MELO

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

**Orientador:** Prof. MSc. Guilherme de Cleva Farto

**Examinador:** _____

Assis/SP
2017

# AGRADECIMENTOS

Agradeço primeiramente a Deus, aquele que me deu a vida e tornou possível a realização de tantos sonhos, inclusive o de terminar a graduação.

Aos meus pais, Marisa Aparecida Rodrigues de Melo e Valdir Inácio de Melo, por tornarem possível esse sonho, além de toda educação e formação que me deram.

Ao meu professor e orientador Professor MSc. Guilherme de Cleva Farto por todo apoio desde o momento em que conversamos sobre futuros temas para a monografia.

A todos os professores que passaram pela minha vida até o instante e que, de seus modos, me passaram todo conhecimento possível de maneira exemplar.

Por fim, agradeço aos meus amigos de trabalho, curso e familiares que me incentivaram a não desistir dos meus sonhos, colaborando indireta e diretamente no meu viver.

# DEDICATÓRIA

Dedico este trabalho a Deus, por sempre ter me dado força necessária para não desistir dos meus sonhos, aos meus pais, por se esforçarem ao máximo para que eu pudesse ter uma boa educação. Dedico também aos professores que me acompanharam até aqui e aos meus amigos por todo apoio dado em diversos momentos do presente trabalho.

*“Só desperta paixão em aprender quem tem paixão em ensinar.”*

Paulo Freire (1921-1997)

# RESUMO

A Internet das Coisas atualmente é uma das áreas em crescimento no mercado, pois com o aumento de dispositivos móveis conectados a Internet, ficou mais fácil a implementação desta tecnologia. Prefeituras municipais estão utilizando a Internet das Coisas para melhorar o acesso dos cidadãos aos serviços urbanos, empresas como Cisco e IBM estão investindo em Internet das Coisas para criação de novas possibilidades em suas soluções. A proposta do trabalho foi a de utilizar os conceitos de Internet das Coisas e suas implementações para construir um protótipo utilizando o protocolo MQTT, Arduino e a tecnologia Node.js para a área de agroindústria. Notou-se que com ferramentas simples como o Arduino e alguns sensores, foi possível a criação de um dispositivo inteligente capaz de trocar mensagens por meio de um protocolo leve como o MQTT, que facilitou a comunicação entre o dispositivo e a aplicação web para que o agricultor tenha toda informação necessária em suas mãos.

**Palavras-chave:** Internet of Things; MQTT; Smart Agriculture; Digital Farming;

# ABSTRACT

Internet of Things is currently one of the growing areas in the market, because with the increase of mobile devices connected to the Internet, it became easier to implement this technology. City halls are using the Internet of Things to improve citizens' access to urban services, companies like Cisco and IBM are investing in Internet Things to create new possibilities in their solutions. The proposal of the work was to use the concepts of Internet of Things and their implementations to build a prototype using the MQTT protocol, Arduino and Node.js technology for the agroindustry area. It was noted that with simple tools like Arduino and some sensors, it was possible to create a smart device capable of exchanging messages through a light protocol such as MQTT, which facilitated the communication between the device and the web application so that the Farmer has all the necessary information in his hands.



**Keywords:** Internet of Things; MQTT; Smart Agriculture; Digital Farming.

# LISTA DE ILUSTRAÇÕES

# LISTA DE ABREVIATURAS E SIGLAS

AMQP - Advanced Message Queuing Protocol

API - *Application Programming Interface*

CoAP - Constrained Application Protocol

D2D - Device to Device

D2S - Device to Server

HTML – HyperText Markup Language

IoT - Internet of Things

ISBG - Internet Business Solutions Group

IoT-ARM - IoT Architetural Reference Model

IDE - Integrated Development Environment

JSON – JavaScript Object Notation

MCTIC - Ministério de Ciência, Tecnologia, Inovações e Comunicações

M2M - Machine to Machine

MQTT - Message Queue Telemetry Transport

REST - Representational State Transfer

RFID - Radio Frequency Identification

S2S - Server to Server

URL - Uniform Resource Locator

URI – Uniform Resource Identifier

XMPP - Extensible Messaging and Presence Protocol

## SUMÁRIO

# 1. INTRODUÇÃO

O número de dispositivos conectados à internet tem crescido nos últimos anos, conjuntamente a isso, o número de dados compartilhados entre esses dispositivos também. Segundo Evans (2011) da CISCO, até 2020, serão 50 bilhões de dispositivos conectados a rede. Com tantos dispositivos conectados que trocam informações entre si a todo o momento, surge o que chamamos hoje de Internet das coisas ou *Internet of Things* (IoT). A IoT refere-se a objetos físicos ou virtuais conectados à internet gerando e consumindo informações para que seja possível analisar dados e realizar uma tomada de decisão.

O desenvolvimento da IoT fez com que surgisse dentre tantos conceitos, o de cidade inteligente ou *smart city* e mais recentemente, *smart agriculture* ou agricultura inteligente. Harrison et al. (2010) definem essas cidades como sendo instrumentalizadas, interconectadas e inteligentes. O objetivo principal das cidades inteligentes é facilitar a interação das pessoas com os serviços disponibilizados pelas prefeituras por meio de sensores, redes sociais e celulares, por exemplo. Semelhantemente, na agricultura inteligente o maior objetivo é fazer com que o trabalhador rural ganhe mais praticidade, com aplicações simples, robustas e de pouco custo para o seu negócio. A captura de informações em grande quantidade, utilizando objetos pequenos que consomem pouca energia e de baixo custo é a chave para a internet das coisas na agricultura (QIU; XIAO; ZHOU; 2013).

Por ser um termo relativamente novo e ainda pouco explorado, há muitos pontos a serem estudados sobre a Internet das Coisas e suas aplicações. No contexto de cidades inteligentes, há trabalhos como o de Silva e Álvaro (2012) que se baseiam em uma plataforma para melhor distribuição de informações, integração entre aplicações e ambientes, além de arquiteturas de software para ajudar na construção de novas ferramentas de IoT.

No âmbito de agricultura inteligente, o trabalho de Brunelli e Sartori (2016) explora IoT para monitorar periodicamente e remotamente o nível de aquíferos e garantir a proteção e preservação dos mesmos, fazendo uso de uma fonte de alimentação com pouco consumo de energia. Com uma proposta semelhante, o trabalho de Qiu, Xiao e Zhou (2013)

apresenta uma plataforma para monitoramento de culturas em Xangai com o intuito de garantir a quantidade e qualidade das culturas, monitorando desde umidade do solo, pH do solo até pressão do ar.

Como apresentado nos parágrafos anteriores, há estudos utilizando IoT em várias áreas de forma a simplificar o dia a dia do usuário final, porém ainda são poucas as aplicações na agricultura inteligente utilizando o protocolo de troca de mensagens *Message Queue Telemetry Transport* (MQTT) para comunicação entre os dispositivos.

O objetivo desse trabalho é pesquisar e propor uma arquitetura experimental com base em IoT e conceitos de *Smart Agriculture* para comunicação entre os sensores conectados à placas em culturas no campo e a plataforma de desenvolvimento escalável NodeJS.

## 1.1. OBJETIVOS

### 1.1.1. OBJETIVOS GERAIS

O objeto geral desse trabalho é o de investigar e explorar os conceitos em torno de Internet of Things, bem como das tecnologias relacionadas em seu ecossistema com o intuito de propor e prototipar um dispositivo inteligente que apoie a agricultura por meio de sensores.

### 1.1.2. OBJETIVOS ESPECÍFICOS

Pretende-se, com o presente trabalho, implementar um dispositivo inteligente utilizando a plataforma NodeJS e o protocolo MQTT para a comunicação de forma simples e eficiente entre os dispositivos para soluções em *Smart Agriculture*. Para tornar possível a realização do projeto, os seguintes objetivos foram instituídos:

- Compreender as características e cenários de Smart Agriculture;
- Pesquisar e explorar o protocolo MQTT
- Pesquisar e analisar a plataforma NodeJS
- Modelar e implementar um protótipo de IoT utilizando o protocolo MQTT
- Prototipar e avaliar experimentalmente uma placa de IoT

## 1.2. JUSTIFICATIVAS

Tendo em vista o crescimento na utilização de pequenos dispositivos, como sensores conectados a Internet, é de grande valia que se utilize desses componentes para o avanço da tecnologia no meio agrícola, mudando o modo como o agricultor interage com o campo e melhorando o seu controle sobre o seu plantio.

Além disso, as soluções em Internet das Coisas tem o potencial de aumentar ainda mais a produtividade e a qualidade dos alimentos (MCTIC, 2016). O crescimento da agricultura e as perdas geradas pela falta de tecnologias que ajude no dia a dia de usinas e fazendas podem ser auxiliados pelo uso da Internet das Coisas.

## 1.3. MOTIVAÇÃO

O desenvolvimento do projeto consiste em que Internet das Coisas é uma área ainda pouco explorada e com alta perspectiva de crescimento, principalmente no contexto da agricultura.

Utilizar dos conceitos estudados sobre a comunicação entre sensores utilizando o protocolo de comunicação MQTT, conhecimentos adquiridos durante o projeto sobre a plataforma NodeJS e sua comunicação com a plataforma Arduino.

Outra motivação é a contribuição para futuros trabalhos com o intuito de ajudar profissionais na área de Internet das Coisas, visando o alto crescimento do uso de pequenos componentes para comunicação.

## 1.4. ESTRUTURA DO TRABALHO

Este trabalho será estruturado nas seguintes partes:

- **Capítulo 1 – Introdução**: Neste capítulo, contextualizará a área de estudo e apresentarão os objetivos, a justificativa, a lacuna, o estado da arte e a motivação para o desenvolvimento desta pesquisa.
- **Capítulo 2 – Internet das Coisas:** Neste capítulo, pretende-se abordar os conceitos de Internet das Coisas com ênfase na *Smart Agriculture.*

- **Capítulo 3 – Plataforma Arduino, NodeJS e Protocolo MQTT:** Neste capítulo, pretende-se abordar as plataformas Arduino e NodeJS, além do protocolo *Message Queue Telemetry Transport* (MQTT), apresentando suas vantagens para o desenvolvimento da pesquisa.
- **Capítulo 4 – Desenvolvimento de Dispositivos Inteligentes:** Neste capítulo, pretende-se abordar o desenvolvimento de dispositivos inteligentes, mostrando a importância de tais dispositivos nos dias atuais.
- **Capítulo 5 – Proposta de Trabalho:** Neste capítulo, pretende-se apresentar o dispositivo inteligente utilizando protocolo MQTT a ser desenvolvido.
- **Capítulo 6 – Desenvolvimento do Trabalho:** Neste capítulo, pretende-se apresentar o desenvolvimento do dispositivo inteligente.
- **Referências**

# 2. INTERNET OF THINGS

## 2.1. CONCEITOS E SURGIMENTO

A Internet das Coisas é um paradigma no qual as “coisas” físicas e virtuais do nosso dia-a-dia, tais como sensores, celulares, atuadores, são utilizados de forma inteligente em uma rede global dinâmica e autoconfigurada com identificadores únicos e capacidade de interagir entre eles próprios, sem intervenção humana e com objetivos em comum. Uma coisa, na Internet das Coisas, poderia ser uma pessoa com um implante de monitor cardíaco, um animal de fazenda com um *transponder*, um automóvel que tem sensores embutidos para alertar o condutor quando a pressão dos pneus é baixa - ou qualquer outro objeto real ou virtual que pode ser atribuído um endereço IP e fornecido com a capacidade de transferir dados através de uma rede (YANG et al, 2015).

Segundo CARISSIMI (2016), o termo Internet das Coisas ou *Internet of Things* surgiu em 1999 durante o trabalho de um grupo do Auto-ID Center no Massachusetts Institute of Technology (MIT) na área de frequência de rádio com *radio frequency identification* (RFID) e tecnologias emergentes com sensores.

Em 2003, havia aproximadamente 6,3 bilhões de pessoas vivendo no planeta e 500 milhões de aparelhos conectados na internet. Com o crescimento no uso de *smartphones* e *tablets*, o número de dispositivos conectados à Internet subiu para 12,5 bilhões sete anos depois e o número de pessoas subiu para 6,8 bilhões, fazendo com que o número de dispositivos passasse a ser mais de um por habitante (EVANS, 2011). Com essas informações, o Cisco Internet Business Solutions Group (ISBG) estima que a tecnologia IoT surgiu entre 2008 e 2009, época em que o número de dispositivos passou a ser maior que o número de habitantes no planeta.

## 2.2. PROTOCOLOS

Em seu trabalho, CARISSIMI (2016) apresenta um conjunto de protocolos para atender as diversas aplicações e seus requisitos. Os protocolos para Internet das Coisas podem ser agrupados em:

- *Device to device (D2D):* Utilizados para conexão entre dispositivos, garantindo alto desempenho, tempo real e entrega. Por serem mais associados a atividades de controle, são utilizados principalmente em hospitais, sistemas militares, indústria automotiva, entre outros.
- *Device to server (D2S): Destinados a coleta de dados para serem enviados a servidores.*
- *Server to server (S2S): Utilizados para o gerenciamento de informações entre servidores de aplicação.*

Inicialmente por estar se falando de internet, pensou-se em utilizar o modo de endereçamento IP, fazendo uso do protocolo HTTP. Porém, por se tratar de um protocolo que faz uso de *request-response,* seria necessário um *pooling* explícito, o que demandaria maior banda larga. Para resolver esse e outros problemas, CARASSIMI (2016) cita novos protocolos no formato *publish-subscriber.*

***Advanced Message Queuing Protocol (AMQP)****:* Fazendo parte do grupo de protocolos S2S, o AMQP foi criado para comunicação entre servidores de sistemas bancários, onde a sua atividade é garantir o envio e recebimento de mensagens. Para isso, faz uso do protocolo TCP, executando também um protocolo de confirmação de mensagens que são entregues.

***eXtensible Messaging and Presence Protocol (XMPP)****:* Protocolo desenvolvido para mensagens instantâneas entre pessoas, fazendo uso de XML para as mensagens e do protocolo TCP. Para realizar seu trabalho, o XMPP utiliza da infraestrutura do DNS.

***Messaging Queue Telemetry Transport (MQTT):*** Baseado no modelo *publish-subscriber,* onde clientes publicam mensagens que podem ser acessadas por outros clientes autorizados. As mensagens são disponibilizadas através de endereços chamados de tópicos e “gerenciadas” por um *broker*, que armazena as mensagens até que elas sejam lidas pelos clientes. Muito utilizado na área de IoT foi projetado para dispositivos computacionais limitados, conectividade não garantida e banda larga baixa.

**Constrained Application Protocol (CoAP):** Protocolo baseado no HTTP, em que os clientes utilizam os métodos GET, PUT, POST E DELETE para acessarem as

informações. O CoAP segue o modelo arquitetural REST, acessando os recursos por meio de URLs, sendo capaz de transportar os dados por meio de XML e JSON, por exemplo.

## 2.3. SEGURANÇA

A IoT terá um impacto muito grande no dia a dia de todos os humanos, pois faz uso de objetos do nosso cotidiano. Serão muitos os benefícios adquiridos com essa tecnologia, porém também são muitos os desafios, principalmente da área de segurança.

Segundo Shelby e Bormann (2011), há três pilares para a segurança em IoT:

- Confidencialidade: Os dados só podem ser entendidos por outros elementos participantes da comunicação, ou seja, os elementos que não são participantes da conversa sabem que há uma comunicação, porém não recebem os dados.
- Integridade: As mensagens não devem sofrer alterações, pois é muito comum que *hackers* alteram mensagens sem que seja notada a presença dos mesmos. Para resolver esse tipo de ataque, programa-se um tipo de criptografia ponta a ponta nas mensagens, sempre as verificando no lado do receptor.
- Disponibilidade: Manter o sistema sempre seguro contra ataques maliciosos, principalmente de *Denial of Service (DoS*).

De acordo com RAWLINSON (2014), um estudo realizado pela HP em 2014, revelou que 70% dos dispositivos mais comuns de IoT contém algum tipo de vulnerabilidade, desde segurança de senha até falta de criptografia. Empresas mais conhecidas como Google, Apple e Microsoft, por exemplo, levam em consideração a segurança, pois sabem da importância da mesma. Porém, empresas menores que fabricam pequenos dispositivos de IoT não consideram a segurança como algo importante.

## 2.4. ARQUITETURA

Um modelo arquitetural define uma estrutura de sistema que atenda as demandas atuais e futuras, de modo confiável e adaptável, principalmente na Internet das Coisas, onde os modelos de implementação podem ser os variados. Atualmente para IoT existem apenas arquiteturas de referência.

Uma arquitetura de referência fornece, de maneira não ambígua, regras de negócio, estilos e padrões arquiteturais de construção de sistemas, boas práticas de desenvolvimento, elementos de hardware e de software necessários à definição de um sistema real (CARISSIMI, 2016). Os benefícios da adoção de arquiteturas de referência é a qualidade, produtividade e a interoperabilidade entre sistemas distintos que seguiram a mesma arquitetura de referência.

A definição de arquiteturas de referência para Internet das Coisas é algo recente, até por conta do aumento de interesse de vários gigantes da tecnologia nesse ramo. Carissimi (2016) destaca as arquiteturas mais conhecidas e exploradas até o momento, que são a IoT *Architectural Reference Model* (IoT-ARM) e a Arquitetura de Referência proposta pela empresa WS02.

## 2.5. INTERNET OF THINGS NO AGRONEGÓCIO

Uma das áreas que tem um grande potencial para fazer uso da IoT, aumentando a produtividade e a qualidade dos alimentos, é a agroindústria. Atualmente, segundo o Ministério de Ciência, Tecnologia, Inovações e Comunicações (MCTIC, 2016), há perda de 30% da produção agropecuária durante o transporte de cargas. Essa situação citada anteriormente pode ser resolvida com o uso da IoT, por meio de tecnologias de rastreamento.

No Brasil, ainda há gargalos a serem solucionados nesse ramo de atuação, como por exemplo, a banda larga em áreas rurais, principalmente de pequenos produtores, além da falta de regulação para o funcionamento da tecnologia. Outro problema é a capacitação das pessoas do campo, pois ainda são poucas as tecnologias nacionais, dificultando o acesso e o entendimento de pessoas leigas (EMBRAPA, 2016).

## 2.6. DESAFIOS

Evans (2011) lista em seu trabalho alguns desafios e barreiras da Internet das Coisas:

- Adesão do IPv6: Com a limitação de endereços Ipv4, o IoT pode ser prejudicado devido ao grande número de sensores que precisarão de endereços IP únicos. Para resolver esse problema, é necessária a adesão do protocolo na versão mais

nova, o que pode ajudar tanto na configuração automática e na segurança dos dispositivos.

- Energia dos Dispositivos: Será necessário um modo para que os sensores gerem sua própria energia, pois seria muito trabalhoso ter que trocar de bateria de milhares de sensores em uma empresa, por exemplo.
- Padrões e Segurança: Por ser uma tecnologia nova, ainda há poucos padrões de normas do que se pode ou não usar, de como usar e qual a melhor prática em dispositivos. Por estar em expansão, organizações como IEEE já trabalha para resolver alguns desses problemas. Outro desafio é o de segurança dos dispositivos conectados à internet, pois serão muitos dados importantes em rede.
- Conectividade: Para a Internet das Coisas, é necessário garantir uma conectividade entre os dispositivos e a Internet. Porém, em muitos locais, principalmente em lugares distantes como fazendas, sítios e usinas distantes, esse ainda é um problema a ser solucionado.

# 3. PLATAFORMA ARDUINO, NODEJS E PROTOCOLO MQTT

## 3.1. ARDUINO

Arduino é uma plataforma *open-source* criada em 2005 por um grupo de pesquisadores no Interaction Design Institute Ivrea IDDI. Na época, os estudantes precisavam de placas eletrônicas para utilizar na universidade, porém as disponíveis no mercado eram caras e complicadas para um iniciante. Diante deste cenário, o objetivo da equipe de pesquisadores foi o de projetar uma placa barata e simples para amadores (ARDUINO, 2017).

O Arduino é uma plataforma livre, tanto o hardware quanto o software são livres para que qualquer pessoa possa fazer suas modificações necessárias. Com ela é possível desenvolver um medidor de qualidade do ar, sensor de presença, entre outras aplicações (FILIPEFLOP, 2014).

Há uma variação de placas Arduino disponíveis no mercado. A seguir serão apresentadas algumas das versões mais populares.

### 3.1.1. Arduino Leonardo

O Arduino Leonardo é baseado no processador ATmega32u4, que é o único processador da versão. Possui 20 pinos de entrada e saída, em que sete podem ser utilizadas como saídas PWM e doze como entradas analógicas. Possui ainda um cristal de 16 Mhz, conexão micro USB, entrada de energia e um cabeçalho ICSP. A diferença da versão Leonardo é que o ATmega32u4 já possui uma interface de comunicação USB, não existindo necessidade de outro processador (LEONARDO, 2017). Na Figura 1 é exibido o Arduino Leonardo.

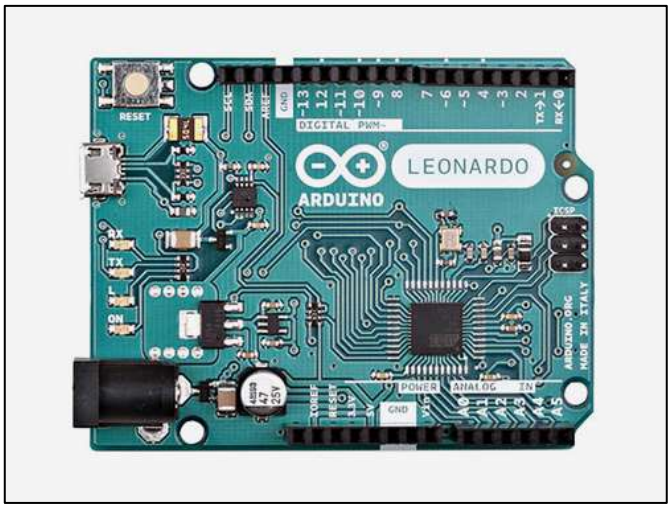


**Figura 1:** Arduino Leonardo
**Fonte:** (LEONARDO, 2017)

### 3.1.2. Arduino Mega ADK

O Arduino Mega ADK é baseado no ATmega2560. Por possuir uma interface de host USB, ele pode ser conectado a um celular Android. Possui 54 pinos de entrada e saída, onde 15 podem ser utilizadas como saídas PWM, 16 de entradas analógicas e 4 de portas seriais de hardware. Além disso, possui um cristal 16 Mhz, conexão USB, entrada de energia e cabeçalho ICSP (MEGA, 2017). Na Figura 2 é exibido o Arduino Mega.

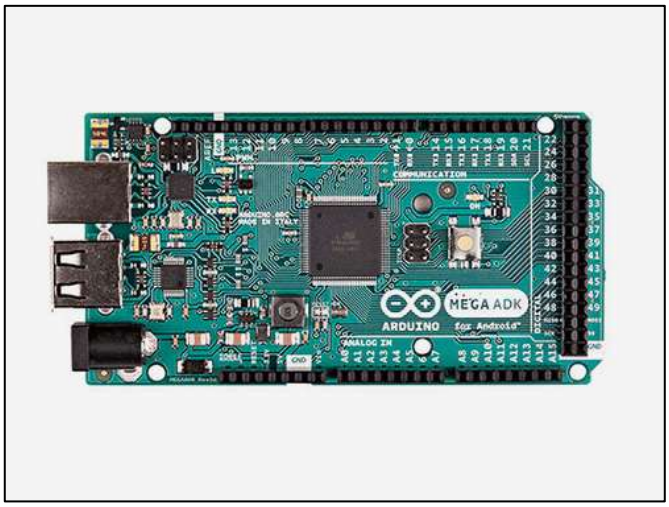


**Figura 2: Arduino Mega ADK**
**Fonte:** (MEGA, 2017)

### 3.1.3. Arduino Lilypad

O Arduino Lilypad é um microcontrolador utilizado em produtos têxteis. Possui formato redondo e é pequeno, sendo fácil de ser colocado em roupas, bolsas, dentre outros produtos. (SIMPLE, 2017). Possui quatro versões até o momento da escrita dessa monografia, são elas a Lilypad Arduino Simple, Lilypad Arduino Main Board, Lilypad Arduino USB e a Lilypad Arduino Simple Snap. Na Figura 3 são mostradas as quatro versões do Arduino Lilypad.

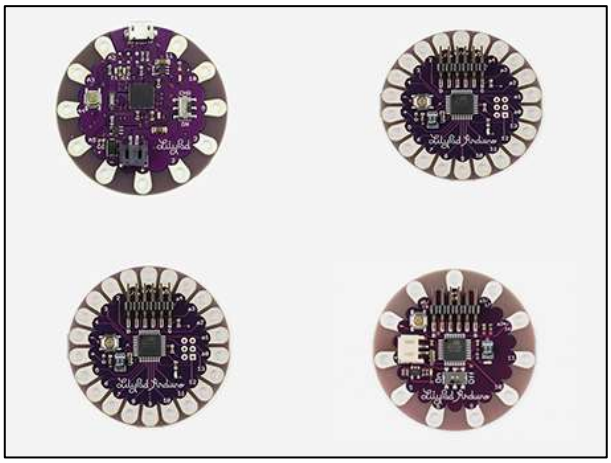

**Figura 3:** Arduinos Lilypad
**Fonte:** Baseado em (SIMPLE, 2017; MAIN, 2017; USB, 2017)

- Arduino Lilypad Simple: O LIlyPad Simple possui nove pinos de entrada e saída, um conector JST, um outro conector para carregamento de baterias e é baseada em um processador Atmega328 (SIMPLE, 2017).
- Arduino Lilypad Main Board: É a principal versão dentre as versões Lilypad. É baseado no processador ATmega168V (MAIN, 2017).
- Arduino Lilypad USB: É um microcontrolador baseado no processador ATmega32u4, contem nove pinos de entradas e saídas digitais, conexão de micro USB, ressonador de 8 Mhz e uma conexão JST para bateria (USB, 2017) .
- Arduino Lilypad Simple Snap: Parecido com a versão Lilypad Simple, porém possui uma bateria de polímero de lítio. Baseado no microcontrolador ATmega328, contendo 9 pinos de entrada e saída (SIMPLE, 2017).

### 3.1.4. Arduino Due

O Arduino Due possui um processador ARM de 32 bits, o processador Atmel SAM3x8E ARM Cortex-M3, tendo a mesma quantidade de pinos de entrada e saída do Arduino Mega. Possui também quatro portas serias de hardware, clock de 84Mhz, conexão USB OTG, dois TWI, uma entrada de energia, conector SPI e conector JTAG (DUE, 2017). Na Figura 4 é exibido o Arduino Due.

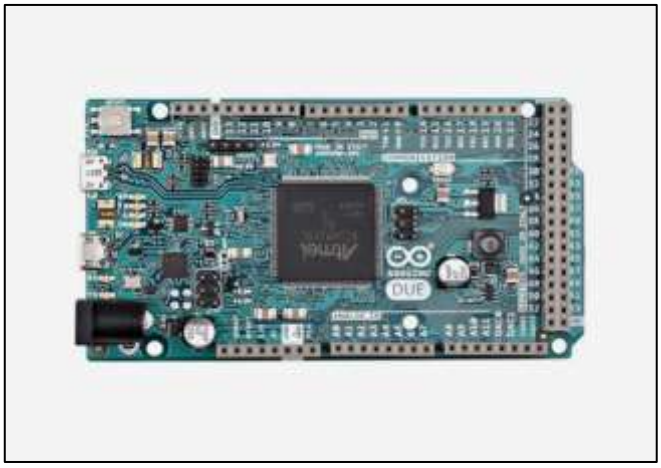


**Figura 4:** Arduino Due
**Fonte:** (DUE, 2017)

### 3.1.5. Arduino Micro

O microcontrolador Arduino Micro é beasdo no processador ATmega32U4, possuindo vinte pinos de entrada e saída, um oscilador de cristak de 16 Mhz, uma conexão micro USB, um cabeçalho e um botão de reset. Semelhantemente a versão Leonardo, o Micro também possui uma comunicação USB inserida no Atmega 32U4, eliminando um segundo processador (MICRO, 2017). Na Figura 5 é apresentado o Arduino Micro.

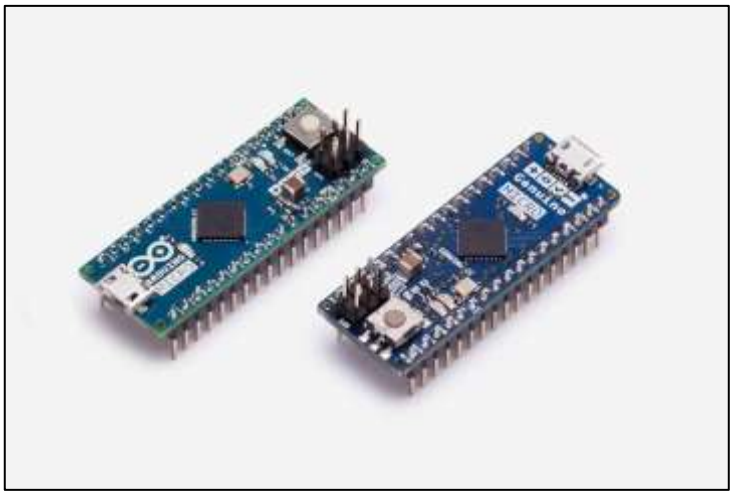

**Figura 5:** Arduino Micro
**Fonte:** (MICRO, 2017)

### 3.1.6. Arduino Robot

O Arduino Robot é o primeiro a ter rodas, tem dois processadores, um em cada lado do microcontrolador. Os motores são controlados pela placa que lê os sensores e decide como operar em determinados casos. Os dois processadores são baseados no Atmega32u4. Grande parte dos pinos são mapeados para sensores e controladores dos motores (ROBOT). Na Figura 6 é apresentado o Arduino Robot.

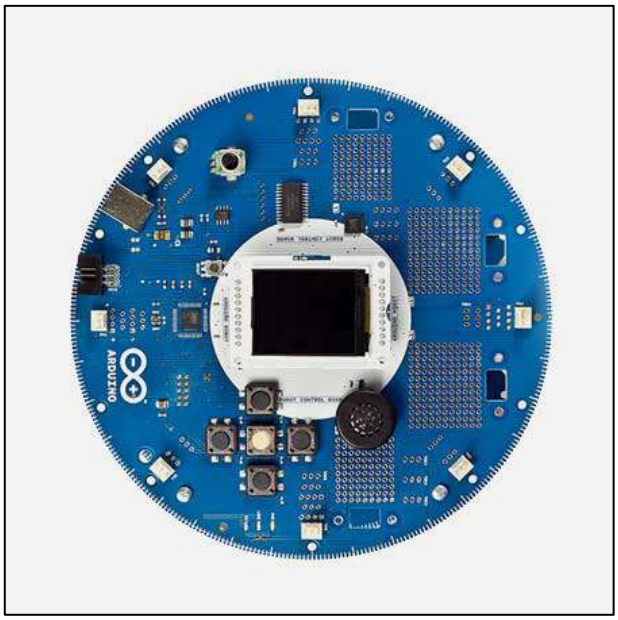

**Figura 6:** Arduino Robot
**Fonte:** (ROBOT, 2017)

Dentre as diversas versões da placa Arduino, a principal delas é a UNO, portanto, será utilizada como exemplo para identificarmos os principais componentes da placa. A Figura 7 mostra os principais componentes do modelo UNO nomeados com a letra “C” de componente e um número para fins de organização.

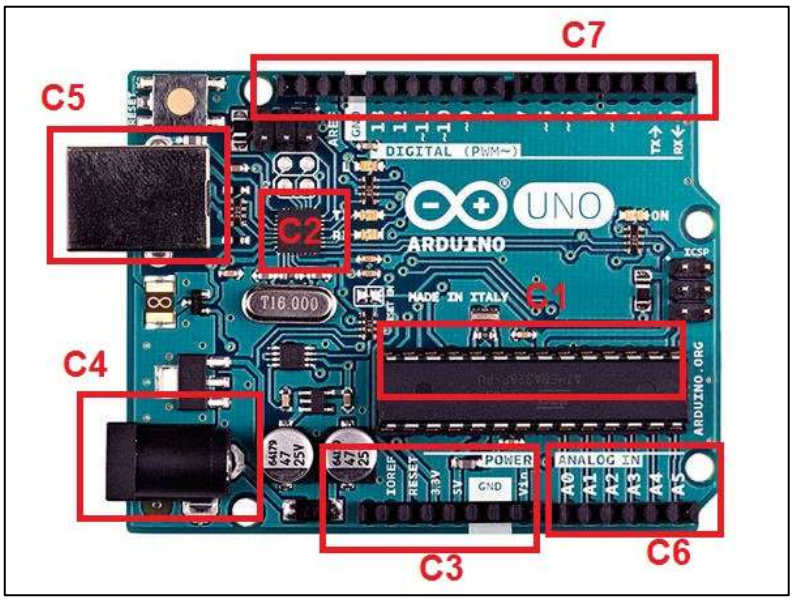


**Figura 7:** Arduino UNO

Fonte: Baseada em (UNO, 2017)

- C1 (Microprocessador ATMEL ATmega328): É o principal componente da placa Arduino UNO, sendo um dispositivo de 8 bits e com arquitetura RISC. Possui ainda, uma memória Flash de 32 KB, 2 KB de RAM e 1 KB de EEPROM, podendo operar em até 16 MHz (SOUZA, 2013). O ATmega328 é responsável por receber, enviar e interpreta sinais da serial que vêm do processador ATmega16U2, executa o software que existe nele e também interage com sensores e *shields* que podem ser adicionados a placa (FURLAN, 2013).
- C2 (Processador USB): Para a comunicação entre Arduino e o computador do programador, é necessário que haja uma conversão USB-Serial, pois o processador principal do Arduino, o ATmega328, não tem suporte a uma conexão direta com USB. O responsável por realizar essa conversão é o processador ATmega16U2 (FURLAN, 2013).
- C3 (Conectores de Alimentação): Conectores utilizados para alimentação de *shields*.

- C4 (Conector Jack): A alimentação externa do Arduino é feita através do conector Jack, que possui positivo no centro, em que o valor da tensão da fonte externa deve estar entre 6V e 20V (SOUZA, 2013).
- C5 (Conector USB): Conector USB do tipo B onde é plugado o cabo USB que será utilizado para conectar ao computador do programador.
- C6 (Conexões analógicas): Pinos de entradas analógicas em que cada uma delas possui resolução de 10 bits.
- C7 (Conexões digitais): São 14 pinos que operam em 5V, podendo receber 40 mA. Os pinos três, cinco, seis, dez e onze são utilizados como saídas PWM. Os pinos zero e um são utilizados para comunicação serial. Os pinos dois e três são utilizados para geração de interrupção externa.

O Arduino sozinho pode fazer projetos simples e pequenos, como acender luzes em determinado tempo, por exemplo. Porém, para projetos de prototipação mais robusta como monitoramento e irrigação de colheitas, é necessário um algo a mais. Para isso, o Arduino possibilita a utilização de *shields*, que são placas de circuitos que contêm outros dispositivos como módulo Ethernet, receptores GSP e display LCD (MCROBERTS, 2011). A principal utilização do Arduino é feita junto a atuadores e sensores para implementação de testes de futuros sistemas automatizados, como o de casas autônomas e até obras de arte contemporânea.

Para a programação da placa Arduino, é utilizada uma *Integrated Development Environment* (IDE) que pode ser executada em múltiplas plataformas (Windows, Linux e Mac) e uma linguagem de programação fortemente baseada em C/C++ e Java, sendo essas bem populares entre os desenvolvedores iniciantes e avançados (ARDUINO, 2017). Na Figura 8 é apresentada a interface gráfica do ambiente de desenvolvimento do Arduino.


**Figura 8:** IDE Arduino

Por ser facilmente utilizada por um iniciante em eletrônica e também pelas opções de alteração com um sistema *open-source*, o Arduino é uma plataforma barata e simples para o desenvolvimento de um projeto em *Internet of Things*.

## 3.2. NODE.JS

### 3.2.1. Conceitos

Node.js é uma tecnologia baseada na *engine* utilizada *no* Google Chrome chamada Javascript V8. O seu principal propósito é o de evitar a paralisação dos processamentos que estão utilizando uma entrada ou saída do servidor, o que a torna uma tecnologia inovadora, pois diferente das plataformas Java e PHP que possuem arquiteturas bloqueantes, Node.js possui uma arquitetura não-bloqueante (PEREIRA, 2014).

A arquitetura não-bloqueante é uma arquitetura onde não há um gerenciador de *threads* que bloqueia o acesso a recursos do sistema quando um usuário já está acessando, enfileirando outras *threads*. Por essa característica, o Node.js não é recomendado para sistemas transacionais, pois nesses sistemas é necessária a garantia de dados consistentes (DEV, 2012).

Criado por Ryan Dahl no final de 2009, o Node.js além de possuir arquitetura *non-blocking thread* (não-bloqueante), é orientada a eventos e *single-thread*, o que faz com que tenha boa performance, consumindo pouca memória e evitando *dead-locks* no sistema (RIBEIRO, 2014).

Com o Node.js, o desenvolvedor tem mais liberdade para programar suas aplicações e servidor, pois é possível programar com diversos tipos de protocolos de rede como HTTP, HTTPS e DNS, além de bibliotecas que utilizam de recursos do sistema operacional.

Diferentemente de plataformas como o Java, que utilizam de programação concorrente (*multi-thread*), cada aplicação Node.js tem apenas uma instância com um único processo, sendo *single-thread.*

### 3.2.2. Event-Loop

O Node.js é orientado a eventos e utiliza uma forma de chamada de eventos muito parecida com a realizada pelo Javascript no *client-side,* porém sem realizar tarefas com componentes HTML, apenas com eventos de input/output no servidor.

O responsável por escutar e emitir os eventos no sistema é o *Event-Loop.* Funcionando como um loop infinito, a cada iteração ele verifica se um evento foi emitido na fila de eventos. Caso um evento tenha sido emitido, ele é executado e adicionado à fila de eventos executados. Na Figura 9 é apresentada a arquitetura de funcionamento do Node.js, onde a aplicação escrita em Javascript é compilada pela *engine* V8, passando pela API do Node.js e assumindo um lugar na fila de eventos. Quando chega sua vez, a ação é executada retornando por callback.

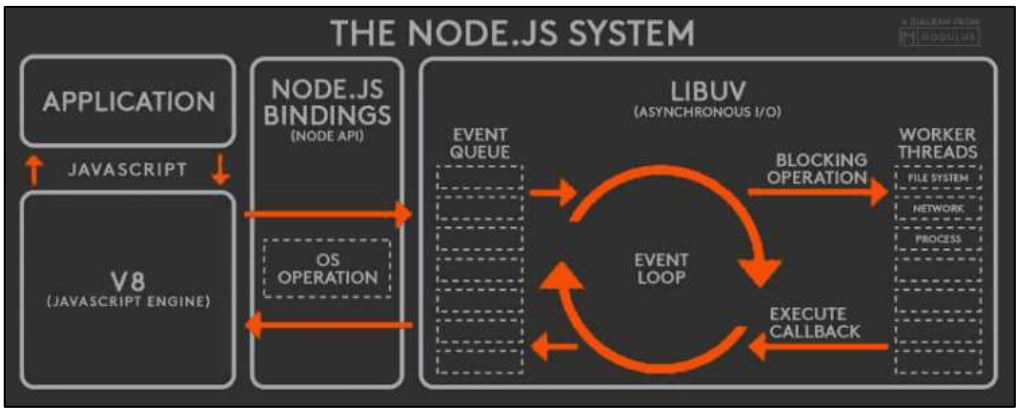


**Figura 9:** Arquitetura Node.js
**Fonte:** (PICOLOTTO, 2016)

### 3.2.3. NPM

Assim como o Java possui o Maven para gerenciamento de pacotes, o Node.js possui o NPM. O NPM torna mais fácil o compartilhamento de projetos e APIs, pois com apenas um arquivo JSON chamado “package.json”, todos os módulos necessários para que a API utilizada no projeto Node.js funcione estão contidos nesse arquivo (NPM, 2017). Na Figura 10 é apresentado o arquivo “package.json”.


**Figura 10:** Arquivo package.json

A seguir serão descritos alguns itens do package.json demonstrados na Figura 10:

- Name: Define nome do projeto
- Version: Versão atual do projeto
- Description: Descrição de funcionamento do projeto
- Main: Arquivo de inicialização
- Scripts: Script realizado ao iniciar o projeto
- Author: Autor da aplicação
- Dependencies: Dependências necessárias para que o projeto funcione caso algum outro programador utilize-o.

Existem diversos comandos possíveis de serem realizados com o NPM para a instalação, remoção, atualização e listagem de pacotes, por exemplo. A seguir são apresentados alguns desses comandos.

- npm –v: Verifica versão do NPM
- npm install sqlite: Instala pacote no projeto
- npm remove sqlite: Remove pacote do projeto

### 3.2.4. Desempenho

A linguagem Javascript utiliza programação assíncrona para executar diferentes funções de modo diferente. Esse modo de execução faz com que o Node.js tenha um alto desempenho, principalmente em operações de entrada e saída, pois funções assíncronas são executadas em paralelo com outras partes do código.

Em sua pesquisa (CROSS et al., 2011) comparou o desempenho do Node.js após reescreverem um aplicativo chamado IBM Passes, feito inicialmente em Java. Foram utilizados como parâmetros o nível de simultaneidade, o número de solicitações e a configuração da solicitação. Para o projeto Java foi utilizado o banco de dados DB2 e para o Node.js o banco NoSQL MongoDB. Como resultado, perceberam que, quando o nível de simultaneidade passou de 50, o Node.js apresentou tempo de resposta superior ao Java, pois houve mais operações de E/S, porém abaixo desse nível de simultaneidade, o Java foi superior, pois aumentou-se o número de operações dependente de CPU e diminuiu as operações de E/S.

## 3.3. PROTOCOLO MQTT

O protocolo *Message Queue Telemetry Transport* (MQTT) é um protocolo de rede baseado no protocolo TCP, criado pela IBM para a comunicação *Machine to Machine* (M2M) de forma simples e rápida, sendo ideal para bandas de Internet mais lentas. Para a troca de mensagens o MQTT utiliza um padrão chamado *publish/subscriber*, onde o elemento que deseja receber (*subscriber*) ou enviar (*publisher*) informações, realiza uma requisição a um broker, que trabalha como um intermediador entre os elementos (BARROS, 2015). Com essa estrutura, apenas o endereço do broker precisa ser

conhecido pelos elementos, havendo possibilidades de comunicação um para um(*one-to-one*), um para muitos(*one-to-many*) e até muitos para muitos(*many-to-many*) (TORRES, ROCHA, SOUZA, 2016).

Na Figura 11 é demonstrado o funcionamento do *broker*. Os *publishers* enviam dados por meio de *topics* ao *broker*, enquanto os clientes que são os *subscribers* recuperam e enviam dados ao *broker.*

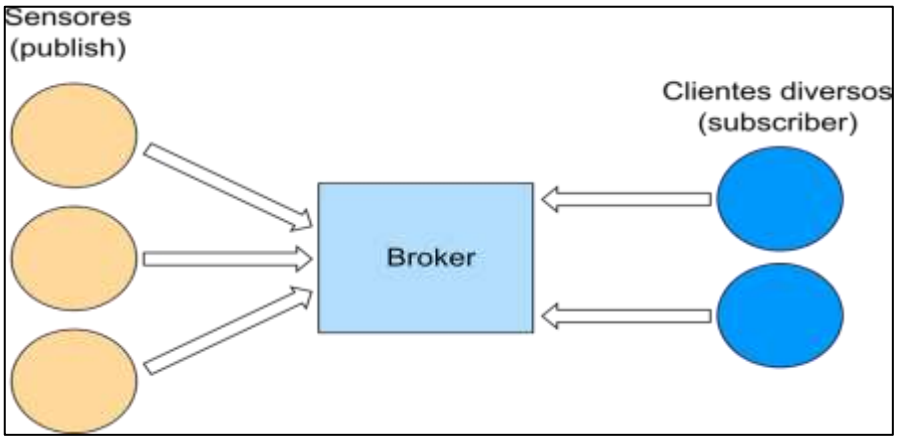


**Figura 11:** Funcionamento do MQTT
**Fonte:** (BARROS, 2015)

O *broker* é um servidor virtual responsável por armazenar as mensagens, recebendo e disponibilizando-as aos clientes. As mensagens são identificadas por tópicos (*topics*), que possuem um formato muito parecido com uma URI onde os níveis são separados por barra, e *payloads,* que são as mensagens propriamente ditas (SUHANKO, 2016).

Por possuir características que facilitam a comunicação M2M, podendo ser implementado em ambientes restritos, o protocolo MQTT é uma ótima alternativa para implementações no ambiente de *Internet of Things*. Empresas como o Facebook e IBM estão utilizando o protocolo. O Facebook faz uso do MQTT em seu aplicativo de mensagens, o Facebook Messenger e a IBM em diversas aplicações como hospitais, por exemplo.

# 4. DESENVOLVIMENTO DE DISPOSITIVOS INTELIGENTES

O principal componente para a realização da Internet das Coisas é um objeto. Objetos podem ter diversos tamanhos, formas e meios de se comunicar com outros objetos, como Bluetooth, *Wifi*, Zigbee, entre outros. Cada objeto possui responsabilidades, como por exemplo, aquisição de dados, gerenciamento de energia, firmware, segurança e gerenciamento de identidade, podendo também ser conectado a uma nuvem onde é representado por algum elemento.

Babu (2015) dividiu o desenvolvimento de dispositivos inteligentes para IoT em 4 etapas. Na primeira etapa, chamada de descoberta, é necessário pesquisar o que pode ser afetado ou auxiliado no negócio da empresa com o uso de Internet das Coisas, descobrindo o que pode ser conectado e o que não pode ser. Outro ponto importante na primeira etapa é o questionamento sobre a capacidade da equipe de desenvolvimento.

Na segunda etapa, chamada de protótipo, em que é identificado que tipo de tecnologia poderá ser utilizada para a criação da solução, o modo de conectividade entre os dispositivos, além dos recursos de segurança. Arduino e Raspberry são muito utilizados nessa etapa para pequenos testes antes de ir para experimentos mais reais. Nessa etapa é importante que toda a equipe aprenda a utilizar os componentes, podendo fracassar em diversos pontos para que haja um avanço do projeto.

A terceira etapa é onde são realizados os testes em campo, sendo a principal etapa, pois é quando a equipe realizará o teste efetivo do que foi testado e escolhido na fase de prototipação utilizando dados e ambientes reais. Nessa etapa os testes precisam ser os mais realistas possíveis, pois é o ambiente final onde a solução será utilizada. Essa fase tem o tempo determinado pelo tipo de dados buscado pela equipe, pelas mudanças no negócio, pela concorrência, escolhas de tecnologia e o ambiente normativo.

Na quarta e última etapa, a de transformação, é onde o negócio será realmente transformado pela solução de IoT. Nessa etapa haverá mudanças no envolvimento do cliente com as novas soluções.

Para o desenvolvimento de dispositivos inteligentes, é necessário identificar as limitações do projeto, pois, na maioria nas vezes, os dispositivos são implantados em lugares que

não têm conectividade e energia garantidas, segurança de alta confiança, além de atualizações de firmware e memória.

É necessário o desenvolvimento de protótipos simples no inicio de qualquer dispositivo de IoT, não apenas de hardware, mas também de software e rede. Para um protótipo simples de hardware e primeiras impressões sobre o produto IoT, é possível fazer a utilização do Arduino, por exemplo, pois é de fácil uso inicial e baixo custo. A utilização de componentes que utilizam protocolos padrão é uma escolha importante na criação do produto, pois facilita a escalabilidade de acordo com o surgimento de novas implementações, podendo trocar uma placa Arduino por um Raspberry, por exemplo.

A conectividade ainda é um grande desafio para a Internet das Coisas, principalmente no Brasil. Portanto, faz-se necessário quando se pretende desenvolver um dispositivo inteligente, o desenvolvimento primeiro de modo que o mesmo funcione offline, para que, caso os recursos não estejam disponíveis, o usuário final não seja afetado. Para o desenvolvimento no modo offline, Fisher (2015) listou os seguintes princípios gerais:

- Presumir que a conexão pode ser perdida a qualquer momento
- Fazer uso de mensagens pequenas e frequentes
- Utilizar filas para as mensagens no modo offline
- Aplicativo sempre avaliar a conectividade antes de qualquer função
- Utilizar sincronização de dados por trás para atualização do sistema
- MQTT para a comunicação de mensagens

## 4.1. SEGURANÇA NO DESENVOLVIMENTO

Os projetos em Internet das Coisas podem possuir grande quantidade de dispositivos inteligentes trocando e gerando dados a todo o momento. De modo que a necessidade de conexão a nuvens e outros serviços da web realizados pelos dispositivos, o risco de um furto de dados é grande. Algumas das ações a serem tomadas em relação a isso são: Garantir comunicação segura entre os dispositivos e a nuvem, monitoramento contínuo de todo o sistema e sempre se recuperar rápido após um ataque hacker. O mundo de IoT não se concentra apenas em empresas, portanto, é necessário o monitoramento e a

segurança em infraestruturas físicas, como um carro ou helicóptero que poderá ser *hackeado*, por exemplo (GANTAIT, 2016).

Dispositivos de IoT têm, normalmente, energia e potencial de processamento limitados. Esse é um dos desafios para a segurança dos mesmos, pois impossibilita o acoplamento de algoritmos de criptografia avançada que necessitem de alto desempenho. Outro desafio é a atualização constante de segurança para esses dispositivos, por conta da capacidade de armazenamento dos mesmos e da baixa banda larga.

Para a proteção no desenvolvimento das principais camadas dos projetos de dispositivos, podemos considerar a camada de dispositivos, camada de rede e transporte, além da camada de aplicativos. Na camada de dispositivos deve haver proteção contra falsos servidores que enviam comandos maliciosos ou hackers utilizando dados enviados por sensores. Na camada de rede pode haver dispositivos falsos enviando dados corrompidos ou que quebram o aplicativo, e na camada de aplicação pode haver uso inválido de dados.

Segundo Gantait (2016), o protocolo MQTT é o mais utilizado para o desenvolvimento de dispositivos inteligentes em IoT, pois é leve e fácil de usar. Com esse protocolo, é possível garantir três tipos de autenticações:

- Autenticação por usuário e senha: Usuário e senha são enviados ao conectar em um *broker* MQTT e ambos não são criptografados pelo protocolo, mas pode ser criptografado no transporte.
- Autenticação com *token* de acesso: O cliente recupera um *token* de acesso e utiliza o mesmo na mensagem de conexão pelo campo de *password*. O tamanho do *token* não pode ultrapassar 65535 bytes, tamanho limite dos campos.
- Autenticação baseada em certificado: O *broker* utiliza de certificados para usar como parte do processo de autenticação mútua.

# 5. PROPOSTA DE TRABALHO

Neste capítulo será apresentado o problema que se pretende resolver, a arquitetura do projeto desenvolvido e quais tecnologias foram adotadas, além de como, juntas, elas se relacionam para resolver o problema proposto. O problema definido consistiu em modelar e implementar um protótipo de Internet das Coisas fazendo uso do protocolo MQTT para transmissão de dados captados por meio de sensores juntos ao Arduino. Como complemento a implementação, a criação de um sistema utilizando a *stack* de desenvolvimento MEAN (MongoDB, ExpressJS, AngularJS e NodeJS) (MEAN, 2017).

## 5.1. DEFINIÇÃO DO PROBLEMA

A definição de um problema se estabeleceu no desenvolvimento de um dispositivo inteligente utilizando a plataforma NodeJS, o protocolo MQTT juntamente com um protótipo em Arduino. Para o desenvolvimento do trabalho foram utilizados conceitos de Internet das Coisas e do protocolo de comunicação MQTT para a comunicação com o protótipo e os sensores conectados ao Arduino. Os sensores podem ser de diversos estímulos, como luminosidade, temperatura, umidade e chuva, por exemplo. A manipulação e exibição dos dados ficaram sobre responsabilidade da *stack* MEAN, pois possibilita ao programador utilizar de apenas uma linguagem de programação em toda a pilha de ferramentas.

## 5.2. ARQUITETURA DO PROJETO

Na Figura 12 é apresentada a arquitetura do projeto com as tecnologias necessárias para a resolução do problema definido.

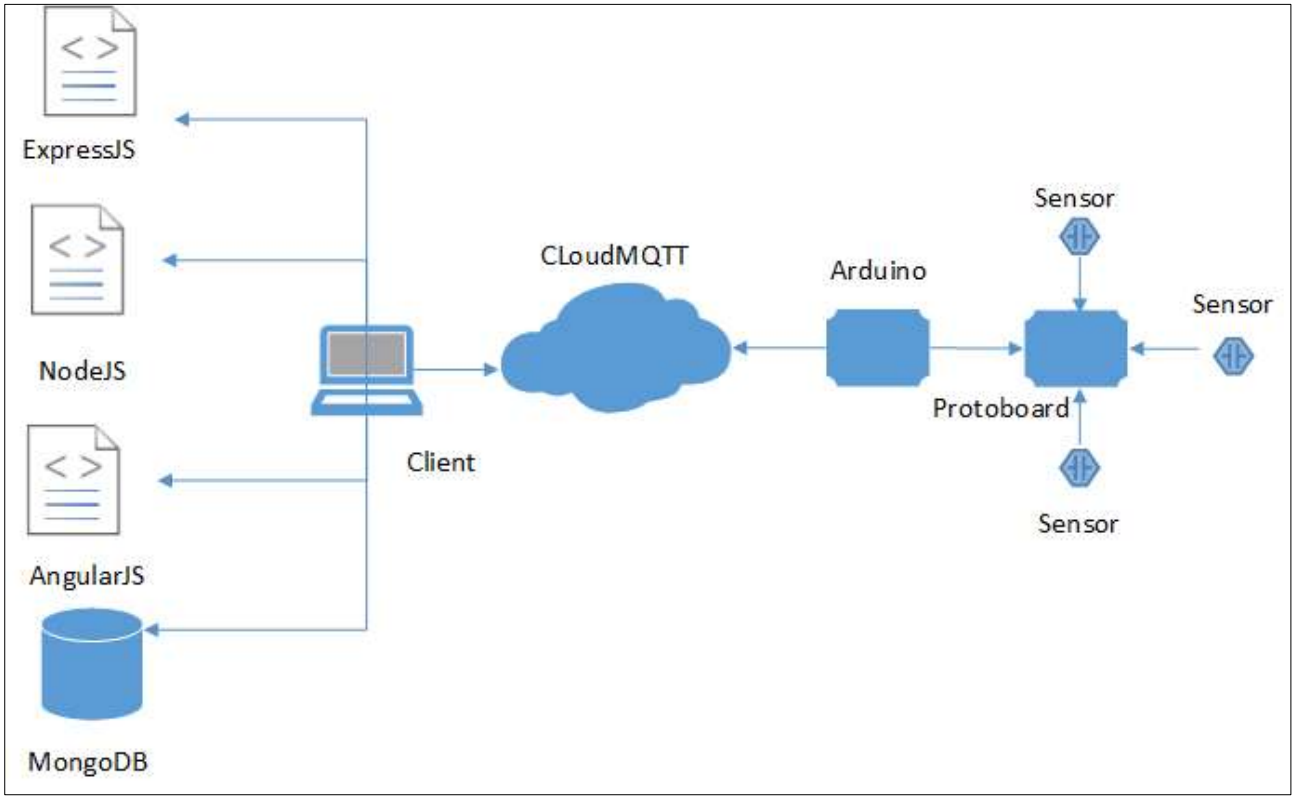


**Figura 12:** Arquitetura do Projeto

## 5.3. TECNOLOGIAS E PLATAFORMAS ADOTADAS

Para o desenvolvimento do protótipo é necessária utilização do Arduino com objetivo de comunicar com os sensores e o módulo w*ireless* para obtenção dos dados por meio do MQTT, um protocolo leve e muito utilizado em projetos de Internet das Coisas. O Arduino foi escolhido principalmente pela facilidade de alterar o seu hardware, preço acessível a grande parte dos iniciantes e fácil integração com diversos sensores. O NodeJS faz a comunicação com o Arduino através de um broker, onde o Arduino publica os dados obtidos dos sensores ao broker e a aplicação NodeJS, que está inscrita no broker, recebe esses dados. Para a exibição dos dados, o NodeJS faz uso de outros *frameworks* como o AngularJS para desenvolvimento do *frontend*, o ExpressJS para gerenciamento das rotas do sistema e o MongoDB para armazenamento dos dados por meio das *collections.*

## 5.4. UTILIZAÇÃO DO PROTOCOLO MQTT

O MQTT será utilizado para a comunicação entre o Arduino conectado aos sensores e a Web API (*Application Programming Interface*). Para que essa comunicação ocorra de forma correta, o protocolo MQTT disponibiliza algo parecido com URIs que são chamados de tópicos (BARROS, 2015). Para que, tanto a API possa receber os dados, quanto o Arduino possa distribuir as informações, deverá haver um tópico em que esse cliente (API) se conecte. O módulo *wireless* ESP8622 conectado ao Arduino se inscreverá nos tópicos e através deles, publicará as informações coletadas por meio dos sensores de umidade, temperatura e luminosidade, além do sensor de chuva.

## 5.5. PLATAFORMA CLOUDMQTT

O protocolo MQTT será utilizado junto a uma solução chamada CloudMQTT, que é um sistema web que disponibiliza um *broker* para aplicações em Internet das Coisas. O Arduino será um publicador (publisher), enquanto a Web API será o inscrito (subscriber) que receberá os dados publicados pelos *publisher*. O CloudMQTT proporciona maior facilidade na configuração de um *broker* e os tópicos necessários para a comunicação entre publicadores e inscritos.

Primeiramente, para a configuração de um *broker* no CloudMQTT, é necessário criar uma conta na solução, como sugere a documentação (CLOUDMQTT, 2017). Após a criação da conta, é preciso criar uma instância de um *broker* no painel de controle (*Panel Control*), como mostrado na Figura 13.

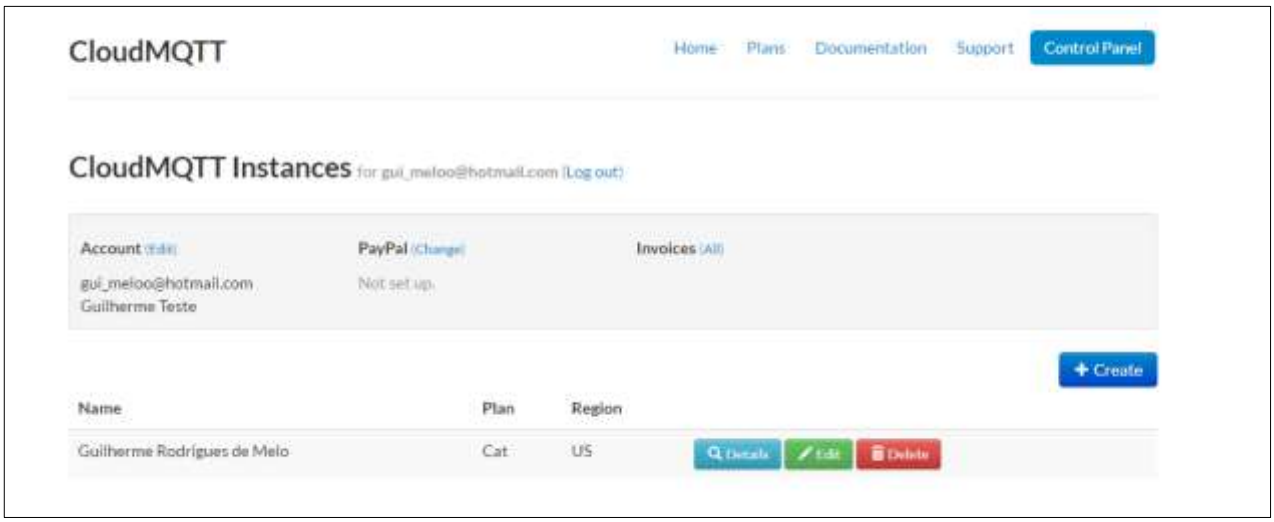


**Figura 13:** Painel de Controle do CloudMQTT

Para acessar as informações como tópicos, usuários e criação de novos tópicos, é necessário ir para os detalhes clicando no botão *Details* mostrado na Figura 13.

Na Figura 14 são mostradas configurações como o servidor, usuário, senha, porta e limite de conexões para aquele *broker.*


**Figura 14:** Configurações do Broker

Para que seja possível fazer o acesso das informações por meio dos tópicos, os mesmos devem estar registrados no servidor *broker*. O CloudMQTT fornece uma interface fácil e

agradável a quem é iniciante no protocolo MQTT. Na Figura 15 é exibido o formulário de criação de tópicos, onde deve ser inserido o usuário e a URI do tópico.

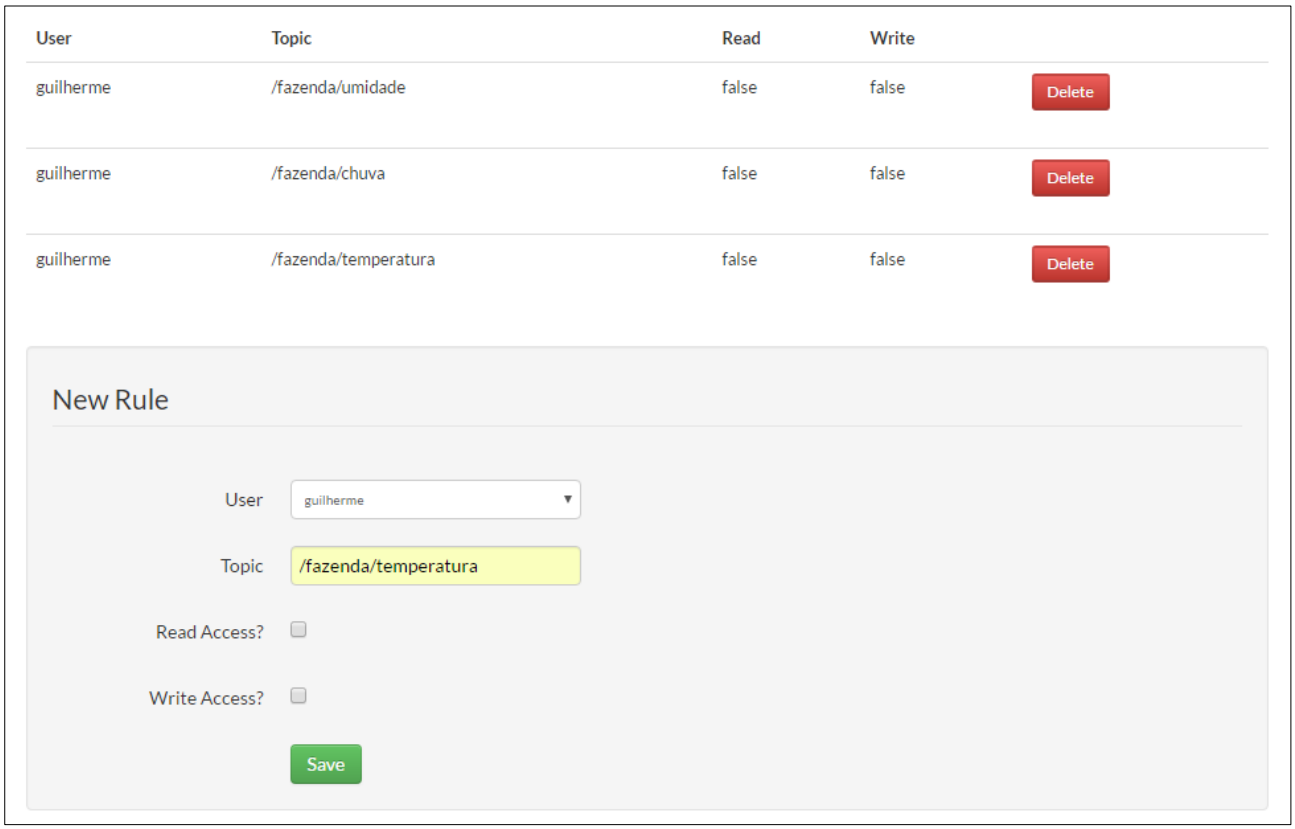


**Figura 15:** Tópicos MQTTCloud

## 5.6. APLICAÇÃO WEB COM MEAN STACK

A aplicação web será construída utilizando o conjunto de ferramentas chamado MEAN. Em um sistema utilizando MEAN, o NodeJS se torna responsável por todo processo que é executado no *backend*. Junto ao NodeJS e sendo um dos frameworks mais conhecidos no mundo JavaScript, o ExpressJS controla as rotas da aplicação, para que o programador tenha maior controle das rotas que poderão ser acessadas em sua aplicação, até mesmo tratando rotas que não existem. O MongoDB, também um dos mais conhecidos banco de dados NoSql (Banco de dados não relacional), é utilizado por sua facilidade na integração com o NodeJS e dinamicidade com relação as *collections*.

Para que o usuário final possa ver os dados obtidos pelo dispositivo de hardware e enviado ao NodeJS, é utilizado o AngularJS, que disponibiliza suas diretivas por meio de *controllers* e *modules.* A maior vantagem em toda a stack de ferramentas que o MEAN proporciona, é que a única linguagem de programação que o programador deve saber é o JavaScript, pois todas elas utilizam a linguagem de seu modo.

Na Figura 16, é mostrada a chamada REST que é realizada pelo *controller* do AngularJS. O AngularJS disponibiliza ao programador uma variável global chamada “angular”. Essa variável contém métodos como o *module, directive* e o *controller,* por exemplo, que auxiliam na criação de controles e módulos para a aplicação.

main-controller.js

```
angular.module('smartfarming').controller('MainController', function($scope, $http){

  $scope.sensores = [];
  $scope.filtro = '';

$http.get('/api/sensores')
     .then(function(resultado){
       $scope.sensores = resultado.data;
      },function(erro){
          console.log(erro);
      });
});

```

**Figura 16:** Chamada da API REST pelo C*ontroller* do AngularJS

.

O AngularJS, que fica no *frontend,* faz requisições ao *backend,* utilizando as rotas que o ExpressJS disponibiliza ao NodeJS para obter as informações armazenadas no MongoDB. A chamada REST mostrada na Figura 16 captura informações através da API que contém rotas específicas para cada chamada. Na Figura 17 é demonstrada a rota que retorna os dados para a aplicação AngularJS e também uma rota coringa, que será executada caso o usuário final desejasse acessar uma rota que não exista na aplicação. Nessa rota, o usuário é redirecionado para a página inicial que está dentro dos diretórios do AngularJS.

```
var path = require('path');

module.exports = function(app){

  var api = app.api.sensor;

  app.get('/api/sensores', api.recuperaSensor);

  app.all('/*', function(req, res) {
        res.sendFile(path.resolve('public/index.html'));
  });
};
```

**Figura 17:** Rotas da API em NodeJS

# 6. DESENVOLVIMENTO DO TRABALHO

Nesse capítulo serão mostradas as experiências e resultados obtidos após o aprofundado estudo sobre Internet das Coisas durante o trabalho. Será apresentada a arquitetura utilizada no projeto, as experiências realizadas com o CloudMQTT e outras plataformas, além do protótipo com Arduino.

## 6.1. REPRESENTAÇÃO ARQUITETURAL

A arquitetura do projeto apresentada no capítulo anterior foi utilizada como base para a criação da arquitetura final. O protótipo utilizando Arduino está inserido na parte física do projeto, onde é conectado aos sensores de chuva YL-83, sensor de temperatura e umidade DHT11, sensor de luminosidade LDR e módulo wireless ESP8266. A parte sistêmica do projeto é realizada pela *stack* MEAN, em que o AngularJS faz requisições por meio de REST para o servidor NodeJS que disponibiliza uma API. Essa API faz acesso aos dados armazenados pelo MongoDB. Os dados inseridos no banco de dados são gravados toda vez que uma *schedule* é executada. A *schedule* realiza a comunicação se inscrevendo no *broker* e verifica se há alguma informação no CloudMQTT. Caso haja alguma informação no *broker,* essa informação é recuperada e salva no banco de dados MongoDB.

## 6.2. EXPERIMENTAÇÃO DO PROJETO

No início do projeto prático foi necessário a utilização de duas plataformas consagradas nos dias de hoje para os testes com a plataforma CloudMQTT, o Java e o Python, com o objetivo de assegurar, por meio dos testes, que o *broker* disponibilizado pela empresa seria ideal para a realização do projeto. Com a plataforma Java foi utilizada uma biblioteca chamada Eclipse Paho, que disponibiliza classes e métodos para a conexão com os *brokers*.

Na Figura 18 é mostrada uma parte do código utilizado para a comunicação e publicação de uma mensagem para o *broker* MQTT, onde é configurado o usuário e senha do *broker* e definida a mensagem que será publicada.

```java
MqttConnectOptions connectOptions = new MqttConnectOptions();
connectOptions.setCleanSession(true);
connectOptions.setUserName("guilherme");
connectOptions.setPassword(new char[]{'t', 'e', 's', 't', 'e'});
mqtt.connect(connectOptions);
MqttMessage message = new MqttMessage("Testando comunicação entre Python e Java por MQTT".getBytes());
message.setQos(1);
System.out.println("Mensagem publicada: " + message);
while(true){
    mqtt.publish("/mqtt/teste", message);
    Thread.sleep(5000);
}
```

**Figura 18:** Configuração da conexão e envio da mensagem ao broker

Com o Python foi criado um *script* que se inscreve no *broker* MQTT utilizando o mesmo usuário e senha que foi utilizado na aplicação Java, também com o auxílio da biblioteca Paho, porém para Python. Na Figura 19 é mostrada uma parte do código utilizado na comunicação.

```python
def on_message(client, userdata, msg):
    print ("Topico: " + str(msg.topic) + "\nQoS: " + str(msg.qos) + "\nMensagem: " + str(msg.payload))

def on_publish(mosq, obj, mid):
    print ("Mensagem publicada...")

def on_subscribe(mosq, obj, mid, granted_qos):
    print("Inscrito: " + str(mid) + " " + str(granted_qos))

def on_log(mosq, obj, level, string):
    print(  string)

client = mqtt.Client()
client.on_message = on_message
client.on_connect = on_connect
client.on_publish = on_publish
client.on_subscribe = on_subscribe

client.username_pw_set("guilherme", "teste")

client.connect('m11.cloudmqtt.com', 10462, 60)

client.loop_start()

client.subscribe ("/mqtt/teste", 0)
```

**Figura 19:** Comunicação entre o Java e CloudMQTT

Com o sucesso do teste de comunicação entre Java e Python usufruindo do *broker* do CloudMQTT, o próximo passo foi a criação do primeiro script de teste com o JavaScript comunicando-se com o Python também utilizando o CloudMQTT. Após êxito da etapa anterior, utilizando o Arduino, foi modelado um projeto de circuito junto aos sensores.

O objetivo do projeto com o Arduino é de conectar o módulo ESP8266 a uma rede wireless e, definido um tempo de *loop*, são capturadas as informações dos sensores. Essas informações são adicionadas a um JSON montado no código embarcado, que é enviado ao CloudMQTT como um texto e capturado, por meio do protocolo MQTT, pelo servidor NodeJS.

## 6.3. COMUNICAÇÃO ENTRE PLATAFORMAS

Como mencionado anteriormente, a comunicação entre as plataformas NodeJS e Arduino é realizada utilizando o protocolo MQTT. Para a comunicação, foram utilizadas algumas bibliotecas. Na aplicação embarcada no Arduino, foi utilizada a biblioteca WifiEsp para a comunicação wireless, além da biblioteca PubSubClient (PUBSUBCLIENT, 2017), que possibilita a comunicação com o *broker* MQTT.

Na Figura 20 é apresentada a montagem e envio do JSON por meio de um tópico.

```
String json = "{";
    json += "\"temperatura\":"; json += "\""; json += temperatura; json += "\""; json += ",";
    json += "\"fazenda\":"; json += "\"Fazenda São José\""; json += ",";
    json += "\"umidade\":"; json += "\""; json += umidade; json += "\""; json += ",";
    json += "\"luminosidade\":"; json += "\""; json += luminosidade; json += "\""; json += ",";
    json += "\"codigo\":"; json += "\"10\"";
    json += "}";

Serial.print("Tamanho do json: ");
Serial.println(json.length());

char payload[200];
json.toCharArray(payload, 200);

if(client.publish("/sensores", payload)){
  Serial.print("Dados enviados ao broker:");
  Serial.println(json);
}else{
  Serial.print("Erro ao tentar enviar dados: ");
  Serial.println(client.state());
}
```

**Figura 20:** Envio e montagem do JSON

Na Figura 20 é apresentado o uso das bibliotecas PubSubClient e WifiEsp, em que são criados os objetos do cliente MQTT e também do wireless. Na sequência, o módulo wireless é iniciado e verificado a sua existência. Caso exista e não esteja conectado, é realizada uma nova tentativa de conexão à rede wireless.

```
WiFiEspClient espClient;

void callback(char* topic, byte* payload, unsigned int length) {}

PubSubClient client(mqtt_server, mqtt_port, callback, espClient);

SoftwareSerial soft(2, 3); // RX, TX

dht DHT; //cria objeto do tipo DHT

void setup() {
  Serial.begin(9600);
  soft.begin(9600);

  WiFi.init(&soft);

  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("Shield não encontrado!");
    while (true);
  }

  // attempt to connect to WiFi network
  while ( status != WL_CONNECTED) {
    Serial.print("Conectando a rede...");
    Serial.println(ESP_SSID);
    status = WiFi.begin(ESP_SSID, ESP_PASS);
  }
```

**Figura 21:** Comunicação Arduino

Na aplicação NodeJS, foram utilizados dois módulos principais, o “node-cron” para a criação de uma *schedule* que executa uma rotina de acordo com o tempo configurado e o “mqtt”, que possibilita a comunicação com o *broker* MQTT. Na Figura 22 é demonstrada a implementação utilizando os módulos, onde, quando recebida uma informação, no caso o JSON, é verificado se essa informação é um JSON, adicionada a data de ocorrência a esse objeto e por fim a persistência do mesmo no MongoDB.

```
var mqtt = require('mqtt');

function schedule(app){
  cron.schedule('*/1 * * * *', function(){
    var api = app.api.sensor;
    var cllientMQTT = mqtt.connect('mqtt://guilherme:teste@m11.cloudmqtt.com:10462');


    cllientMQTT.on('connect', function() {
      console.log('Conectado ao MQTT');
    });

    cllientMQTT.subscribe('/sensores', function(){
      cllientMQTT.on('message', function(topic, message){
        console.log("Mensagem: " + message.toString());
        if(message.toString().startsWith("{")){
            var json = JSON.parse(message.toString());
            json.dtHrRecuperado = moment().format('DD/MM/YYYY HH:mm:ss');
            //Salvar o sensor no banco de dados
            api.gravarSensor(json);
        }else{
          console.log("Payload não é um JSON!");
        }
        cllientMQTT.end();
      });
```

**Figura 22:** Comunicação MQTT com NodeJS

## 6.4. RESULTADOS OBTIDOS

Após a execução do projeto, notou-se que a Internet das Coisas vem sendo uma das principais tecnologias na área de computação e também de fácil acesso a pequenos investidores e leigos. Com ferramentas simples como o Arduino junto a sensores, foi possível criar um protótipo de dispositivo inteligente que se conecta na internet e produz informação para que agricultores possam tomar decisões em suas plantações.

O protocolo MQTT supriu a necessidade de uma comunicação leve entre o sistema web e o dispositivo inteligente, pois a quantidade de dados que são enviados pelo protocolo é muito grande e o consumo de internet também, o que foi facilitado pelo uso do mesmo. A *stack* MEAN possibilitou a experimentação da tecnologia JavaScript em todo o sistema, facilitando a comunicação entre o *frontend*, *backend* e banco de dados.

# 7. CONCLUSÃO

A Internet das Coisas vem sendo uma das principais tecnologias na área de computação, pois está possibilitando a conexão de objetos simples à internet que aceleram a criação de novos protótipos para ideias inovadoras e que agregam valor não só aos agricultores, mas também para outros usuários.

A utilização de ferramentas baratas e Internet das Coisas vêm de encontro ao problema da baixa conectividade em lugares isolados. Porém, para esse problema, o protocolo MQTT foi de extrema importância, pois nos traz a possibilidade de transportar dados que não pesem ao dispositivo e a rede.

Conclui-se que Internet das Coisas com o auxílio de ferramentas de baixo, tecnologias poderosas e simples como Node.js e o protocolo MQTT torna mais acessível e real a transformação no dia a dia de todos, trazendo novas ideias a diversas áreas como a agricultura.

## 7.1. TRABALHOS FUTUROS

A partir deste projeto, é possível criar uma rede inteira de sensores que se comuniquem por meio do protocolo MQTT em campos de produção de cana ou outros cultivos e, através de uma aplicação web ou mobile, gerar informações aos agricultores de forma mais ágil e simples, possibilitando, por exemplo, ver a posição dos sensores em mapas e até se os mesmos ainda têm energia o suficiente para continuarem enviando dados.

Outra contribuição do presente trabalho é a demonstração de que a Internet das Coisas ainda é um campo pouco explorado pela área de segurança tanto na implementação quanto no desenvolvimento de novos elementos inteligentes, fornecendo informações dos métodos mais utilizados para garantir a segurança.

Portanto, conclui-se que o presente trabalho fornece importantes conceitos sobre Internet das Coisas para que o pesquisador tenha base para que, de forma prática, desenvolva trabalhos mais avançados utilizando ferramentas mais desenvolvidas junto ao protocolo MQTT, incorporando não apenas o setor de agroindústria, mas também outros setores.

# REFERÊNCIAS

ARDUINO**. What is Arduino.** Arduino. <http://www.arduino.org/learning/getting-started/what-is-arduino>. Acesso em: 28 jan. 2017.

BABU, Suresh. **Aproveitando a Internet of Things(IoT).** IBMdeveloperWorks. <https://www.ibm.com/developerworks/br/library/iot-key-concepts/>. Acesso em: 18 fev. 2017.

BARROS, Marcelo. **MQTT – Protocolos para IoT.** Embarcados. < https://www.embarcados.com.br/mqtt-protocolos-para-iot/>. Acesso em: 30 jan. 2017.

BRUNELLI, D.; SARTORI, D.: **A Smart Sensor for Precision Agriculture Powered by Microbial Fuel Cells**. In: Sensors Applications Symposium (SAS), 2016.

CARISSIMI, Alexandre. **Internet das Coisas: Middlewares e outras coisas.** ResearchGate.<https://www.researchgate.net/publication/301298394_Internnet_das_Coisas_Middlewares_e_outras_coisas>. Acesso em: 10 fev. 2017.

CLOUDMQTT. **Documentação do CloudMQTT.** Documentation. Disponível em <https://www.cloudmqtt.com/docs.html>. Acesso em: 26 jul. 2017.

CROSS, Zach; POCHEC, Aga; SANTIAGO, Daniel; SINGH, Divit. **Desenvolvendo aplicativos móveis com Node.js e MongoDB, parte 1: Os métodos e resultados de uma equipe.** IBMdeveloperWorks. <http://www.ibm.com/developerworks/br/library/mo-nodejs-1/>. Acesso em: 22 jan. 2017.

DEV. **Node.js para Leigos – Introdução.** UDGWebDev. Disponível em < https://udgwebdev.com/nodejs-para-leigos-introducao>. Acesso em 13 mar. 2017.

DUE. **Arduino Due.** Arduino. Disponível em <https://www.arduino.cc/en/Main/arduinoBoardDue>. Acesso em: 10 mar. 2017.

EMBRAPA. **Especialistas apontam desafios e oportunidades para a Internet das Coisas na agricultura**. Embrapa Informática Agropecuária. Disponível em: <https://www.embrapa.br/informatica-agropecuaria/busca-de-noticias//noticia/18936006/especialistas-apontam-desafios-e-oportunidades-para-a-internet-das-coisas-na-agricultura>. Acesso em: 07 fev. 2017.

EVANS, D. **The Internet of Things: How the Next Evolution of the Internet Is Changing Everything**. 2011.

FILIPEFLOP. **O que é Arduino.** Filipeflop. <http://blog.filipeflop.com/arduino/o-que-e-arduino.html>. Acesso em: 27 jan. 2017.

FISHER, Andrew. **Melhores práticas para desenvolvimento de IoT.** IBMdevelopersWorks. <https://www.ibm.com/developerworks/br/library/iot-mobile-practices-iot-success/>. Acesso em: 12 fev. 2017.

GANTAIT, Amitranjan; MUKHERJEE, Ayan; PRATA, Joy. **Protegendo dispositivos e gateways de IoT.** IBMdeveloperWorks. <https://www.ibm.com/developerworks/br/library/iot-trs-secure-iot-solutions1/index.html>. Acesso em: 19 fev. 2017.

HARRISON, C.; ECKMAN, B.; HAMILTON, R.; HARTSWICK, P.; KALAGNANAM, J.; PARASZCZAK, J.; WILLIAMS, P. **Foundations for Smarter Cities**. IBM Journal of Research and Development, v. 54, n. 4, 2010.

LEONARDO. **Arduino Leonardo**. Arduino. Disponível em <https://www.arduino.cc/en/Main/ArduinoBoardLeonardo>. Acesso em: 09 mar. 2017.

MAIN. **Arduino Lilypad Main**. Arduino. Disponível em <https://www.arduino.cc/en/Main/ArduinoBoardLilyPad>. Acesso em: 10 mar. 2017.

MCROBERTS, Michael. **Arduino Básico**, 1. ed. Tradução de Rafael Zanolli, São Paulo. Novatec, 2011.

MCTIC. **Internet das Coisas eleva a produtividade da agricultura e a qualidade dos alimentos no Brasil**. Ministério da Ciência, Tecnologia, Inovações e Comunicações. Disponívelem:<http://www.mcti.gov.br/noticia/asset_publisher/epbV0pr6eIS0/content/internet-das-coisas-eleva-a-produtividade-da-agricultura-e-a-qualidade-dos-alimentos-no-brasil>. Acesso em: 06 fev. 2017.

MEAN. **JavaScript Fullstack**. MEAN. Disponível em <http://mean.io/>. Acesso em: 24 jul. 2017.

MEGA. **Arduino Mega.** Arduino. Disponível em <https://www.arduino.cc/en/Main/ArduinoBoardMegaADK>. Acesso em: 09 mar. 2017.

NPM. **What is npm?**. NPM. Disponível em <https://docs.npmjs.com/getting-started/what-is-npm>. Acesso em 15 mar. 2017.

PEREIRA, Caio Ribeiro. **Aplicações web real-time com Node.js**. Casa do Código, 2014.

PICOLOTTO, Douglas. **NodeJS – Arquitetura.** Douglas Picolotto. Disponível em <http://douglaspicolotto.com/2016/06/09/nodejs-arquitetura-basica/>. Acesso em 15 mar. 2017.

PUBSUBCLIENT. Pubsubclient. PUBSUBCLIENT. Disponível em < https://github.com/Imroy/pubsubclient>. Acesso em 20 ago. 2017.

QIU, T.; XIAO, H.; ZHOU, P.: **Framework and Case Studies of Intelligence Monitoring Platform in Facility Agriculture Ecosystem**. In: Second International Conference on Agro-Geoinformatics, II, 2013, Xangai, China.

RAWLINSON, Kristi**. HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack.** HP. <http://www8.hp.com/us/en/hp-news/press-release.html?id=1744676#.WLBfPfkrLIX>. Acesso em: 09 fev. 2017.

ROBOT. **Arduino Robot.** Arduino. Disponível em <https://www.arduino.cc/en/Main/Robot>. Acesso em: 10 mar. 2017.

SHELBY, Zach; BORMANN, Carsten**. 6LoWPAN: The Wireless Embedded Internet.** John Wiley Professio, 2009.

SILVA, V. H. S. F.; ALVARO, A.: **Uma Plataforma para Cidades Inteligentes baseada na Internet das Coisas**. In: Simpósio Brasileiro de Sistemas de Informação (SBSI), VIII, 2012, Sorocaba, Brasil.

SIMPLE. **Arduino Lilypad Simple**. Arduino. Disponível em <https://www.arduino.cc/en/Main/ArduinoBoardLilyPadSimple >. Acesso em: 10 mar. 2017.

SUCASAS, Ángel Luis. **26 bilhões de janelas abertas ao crime cibernético**. El País. <http://brasil.elpais.com/brasil/2015/06/15/tecnologia/1434357239_445236.html>. Acesso em 09 fev. 2017.

SUHANKO, James. **IoT – Configurando um MQTT broker**. Do Bit ao Byte. Disponível em <http://dobitaobyte.com.br/iot-configurando-um-mqtt-broker>. Acesso em 15 mar. 2017.

TORRES, Andrei Bosco Bezerra; ROCHA, Atslands Rego da; SOUZA, Jose. **Análise de Desempenho de Brokers MQTT em Sistema de Baixo Custo.** In: Workschop em Desempenho de Sistemas Computacionais e de Comunicação, 15, 2016, Porto Alegre, Brasil.

UNO. **Arduino UNO**. Arduino. Disponível em <http://www.arduino.org/products/boards/arduino-uno>. Acesso em 10 mar. 2017.

USB. **Arduino USB**. Arduino. Disponível em <https://www.arduino.cc/en/Main/ArduinoBoardLilyPadUSB>. Acesso em: 10 mar. 2017.

YANG, S.; CHEN, X.; CHEN, X.; YANG, L.; CHAO, B.; CAO, J. **A case study of Internet of Things: a wireless household water consumption monitoring system.** In: 2nd World Forum in Internet of Things, II, 2015.