UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA ELÉTRICA

Jean Dupont Muenchen

UMA PROPOSTA DE DETECÇÃO DE INCÊNDIO UTILIZANDO O PROTOCOLO MQTT PARA APLICAÇÕES IOT

Santa Maria, RS
2018

Jean Dupont Muenchen

UMA PROPOSTA DE DETECÇÃO DE INCÊNDIO UTILIZANDO O PROTOCOLO MQTT PARA APLICAÇÕES IOT

Monografia apresentada ao Curso de Graduação em Engenharia Elétrica, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Natanael Rodrigues Gomes

Ficha catalográfica elaborada através do Programa de Geração Automática
da Biblioteca Central da UFSM, com os dados fornecidos pelo(a) autor(a).

Endereço: Rua, Dr. Bozano Bairro Bonfim nº 227 ap. 312 Santa Maria, RS. CEP: 97015 001

Fone (0xx)55 991536282 E-mail: jeanmuenchen@gmail.com

Santa Maria, RS
2018

Jean Dupont Muenchen

UMA PROPOSTA DE DETECÇÃO DE INCÊNDIO UTILIZANDO O PROTOCOLO MQTT PARA APLICAÇÕES IOT

Monografia apresentada ao Curso de Graduação em Engenharia Elétrica, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 20 de dezembro de 2018

_____

Prof. Dr. Natanael Rodrigues Gomes (UFSM)
(Presidente/Orientador)

_____

Prof. Ms.Thiago Cattani Naidon (UTFPR)

_____

Ms.Fernando de Souza Savian (UFSM)

DEDICATÓRIA

Dedico a família, pelo incansável apoio e sustentação na minha caminhada.

Santa Maria, RS
2018

AGRADECIMENTOS

Nesse espaço gostaria de deixar minha gratidão às pessoas que contribuíram para a construção desse trabalho.

Inicio pelos meus pais, Rosane Dupont Muenchen e Roque Julci Muenchen, por sempre me incentivarem a buscar conhecimento, investiram na minha educação e por serem exemplos de como dedicação gera bons frutos.

Ao meu orientador Dr. Natanael Rodrigues Gomes, que mesmo com um grupo grande de orientados, aceitou meu pedido e me auxiliou em todas as etapas desde trabalho, desde a definição do tema, configurações e testes, até os resultados finais.

Aos meus amigos do Rotaract Club de Santa Maria Legio Invicta, que me aguentaram, durante os dias mais puxados, incentivando a escrita e desenvolvimento, quando eu não aguentava mais e com ideias para incorporar no trabalho.

E aos demais professores e colegas do curso de Engenharia Elétrica da Universidade Federal de Santa Maria, que contribuíram, direta ou indiretamente, na elaboração deste trabalho, o meu reconhecimento.

Santa Maria, RS
2018

Trabalhe duro e em silêncio, deixe que o seu sucesso faça barulho (Dale Cargnie)

Santa Maria, RS
2018

RESUMO

UMA PROPOSTA DE DETECÇÃO DE INCÊNDIO UTILIZANDO O PROTOCOLO MQTT PARA APLICAÇÕES IOT

AUTOR: Jean Dupont Muenchen
ORIENTADOR: Dr. Natanael Rodrigues Gomes

Este trabalho apresenta uma pesquisa sobre a Internet das Coisas (*Internet of Things* - IoT) que veio para conectar coisas e pessoas atualmente desconectadas. É a nova era na transformação dos sistemas existentes para alterar a qualidade de serviços de baixo custo para a sociedade. No ano de 2013, o incêndio na Boate Kiss, na cidade de Santa Maria, RS, tornou-se à maior tragédia deste tipo no país. Desde então, as normas e instruções técnicas tem enfoque especial, bem como o estudo de novos instrumentos que instrumentalizam a segurança de locais. O trabalho propõe a pesquisa sobre um detector de fumaça inteligente, conectado à internet, explicando o protocolo MQTT, o protocolo I2C, o microcontrolador ESP32, bem como especificações dos sensores explicarem o problema, e propor uma solução simples e com baixo custo. O referido trabalho aborda o estudo de um sistema capaz de detectar incêndio e comunicar órgãos competentes através da internet. O objetivo geral do trabalho é aplicações de um sistema capaz de detectar fumaça, diferentes gases inflamáveis e fogo através da comunicação Wi-Fi. Portanto, utilizando componentes baratos, e com uma simples implementação, é possível salvar vidas, agindo imediatamente após o incêndio começar, o LED ligado e desligado por um relé pode representar acionamentos diversos para auxiliar na evacuação.

Palavras-chave: ESP32, IOT, MQTT, Incêndio, Microcontrolador.

Santa Maria, RS
2018

ABSTRACT

A FIRE DETECTION PROPOSAL USING THE MQTT PROTOCOL FOR IOT APPLICATIONS

AUTHOR: Jean Dupont Muenchen
ADVISOR: Dr. Natanael Rodrigues Gomes

This work presents a research on the Internet of Things (IoT) that came to connect things and people currently disconnected. It is the new era in the transformation of existing systems to change the quality of low-cost services to society. In the year of 2013, the fire in the Boate Kiss, in the city of Santa Maria, RS, became the biggest tragedy of this type in the country. Since then, standards and technical instructions have special focus, as well as the study of new instruments that instrumentalize the safety of sites. The work proposes research on an intelligent smoke detector, connected to the internet, explaining the MQTT protocol, the I2C protocol, the ESP32 microcontroller, as well as sensor specifications explain the problem, and propose a simple and low cost solution. This work addresses the study of a system capable of detecting fire and communicating competent bodies through the Internet. The overall objective of the work is to apply a system capable of detecting smoke, different flammable gases and fire through Wi-Fi communication. Therefore, using cheap components and with a simple implementation, it is possible to save lives by acting immediately after the fire starts, the LED on and off by a relay can represent various drives to aid evacuation.

Keywords: ESP32, IOT, MQTT, Fire, Microcontroller.

Santa Maria, RS
2018

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IOT	*Internet of Things*
MQTT	*Message Queuing Telemetry Transport*
I2C	*Inter-Integrated Circuit*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
ADC	Conversor Analógico-Digital
DAC	Conversor Digital-Analógico
SSL	*Secure Sockets Layer*
QoS	Qualidade de Serviço
SLC	*Serial Clock*
SDA	*Serial Data*
USB	*Universal Serial Bus*
GPIO	General Purpose Input/Output
CI	Circuito Integrado
UART	Transmissão e Recepção Assíncrona Universal
LCD	*Liquid Crystal Display*
LED	*Light Emitting Diode*
WPA	*Wi-Fi Protected Access*
WEP	*Wired Equivalent Privacy*
RTC	*Real-Time Clock*
ULP	*Universal Logic Plug*

SUMÁRIO

1 INTRODUÇÃO

O ser humano evoluiu, buscou maneiras sofisticadas de se comunicar, passando por vários fatores determinantes como o surgimento da escrita, criando possibilidades de registro. Com base nos registros, a escrita foi a primeira tecnologia de pensamento e inteligência desenvolvida pelo homem representando mudanças de paradigmas quanto ao modo de acumular os conhecimentos historicamente construídos.

A década de 50 é caracterizada pela origem da internet, onde militares americanos começavam a pensar em interligar seus computadores para que as informações não fossem perdidas. Para muitos pesquisadores a internet surgiu como uma nova forma de comunicação em 1969 com a Agencia de Pesquisa e Projetos Avançados (*Advanced Research Projects Agency* – Arpa) voltada à pesquisa de informações para o serviço militar, caso os Estados Unidos fossem atacados por outros países.

Atualmente vivemos em uma sociedade da informação, onde há uma cultura informático - mediática, alterando os significados das relações humanas, principalmente no quesito comunicativo. Atualmente, o uso da internet é imprescindível, o que acabou contribuindo para surgimento e aperfeiçoamento da tecnologia. A internet é o meio mais interativo que existe e tem capacidade de reunir todos os outros meios em um único ambiente, trazendo uma forma significativa para a capacidade de criação e disseminação das informações nas áreas humanas, exatas e engenharias.

A Internet das Coisas (*Internet of Things* - IoT) veio para conectar coisas e pessoas atualmente desconectadas. É a nova era na transformação dos sistemas existentes para alterar a qualidade de serviços de baixo custo para a sociedade. Para apoiar a visão de cidade inteligente, os planos de projeto de IoTs urbanos exploram serviços de valor agregado para os cidadãos, bem como a administração da cidade com as mais avançadas tecnologias de comunicação. Para tornar a resposta de emergência (procedimento que define atribuições e responsabilidades de seus diretores e funcionários, relativos aos procedimentos necessários diante da ocorrência de emergências ISO 14001) em tempo real, a IoT aprimora a maneira como os socorristas respondem e fornece aos gerentes de emergência as informações e comunicações necessárias para usar esses recursos. A IoT abrange muitos dos

desafios para a resposta de emergência, incluindo problemas como uma rede de comunicação fraca e atraso na informação.

O trabalho propõe a pesquisa sobre um detector de fumaça inteligente, conectado à internet, explicando o protocolo MQTT, o protocolo I2C, o microcontrolador ESP32, bem como especificações dos sensores explicarem o problema, e propor uma solução simples e com baixo custo.

Para implementar o esquema proposto, o módulo ESP-32 da Espressif, um sensor de detecção de chamas, fumaça e gás inflamável foi usado. Os sensores detectam o perigo e alerta as organizações de resgate de emergência locais, enviando informações para um servidor na nuvem com a localização aproximada do local do incêndio.

Como esquema proposto apresenta-se um microcontrolador, conectado a um servidor, por meio da comunicação Wi-Fi. O mesmo realizará a leitura de sensores de temperatura, umidade, gás e chamas, e envia-las a ao servidor. Dessa forma, possível avisar os bombeiros, deslir as linhas de energia principal, ligar as luzes de emergência e os exaustores, de modo que a fumaça e o gás oxigênio sejam expelidos para que o controle do fogo seja obtido.

1.1 MOTIVAÇÃO

O sistema de detecção de fumaça via comunicação é uma área no mercado que ainda foi pouco explorada por empresas do mundo embarcado, muitos sistemas ainda são analógicos e operando isoladamente, com simples avisos sonoros. Mas sem nenhum meio de comunicação externo para agilizar o socorro das entidades competentes.

Na década de 50 ocorreram diversos incêndios, devastando grandes áreas, como por exemplo, fábricas de automóveis nos Estados Unidos, na Inglaterra entre outros. Pesquisas tem ganhado importância na área, com pesquisas públicas para estudo e novas normativas de segurança. No Brasil o maior caso recorrente de tragédia que causou incêndio é da Boate Kiss, em Santa Maria – RS no ano de 2013.

Desde então, as normas e instruções técnicas tem enfoque especial, bem como o estudo de novos instrumentos que instrumentalizam a segurança de locais. O referido trabalho aborda o estudo de um sistema capaz de detectar incêndio e comunicar órgãos competentes através da internet.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Aplicações de um sistema capaz de detectar fumaça, diferentes gases inflamáveis e fogo através da comunicação Wi-Fi. O sistema deverá prover uma localização aproximada de onde está acontecendo o incêndio, sendo configurado na hora da instalação. Este sistema de detecção de risco de incêndio com a estrutura de IoT sistemática apresenta uma inovação de aplicação para o setor de serviços de segurança pública e de subsistência.

1.2.2 Objetivo Específicos

- Pesquisar formas de detectar gases na atmosfera.
- Implementar uma topologia barata e funcional para detectar fumaça
- Propor um sistema de baixo custo e alta confiabilidade no envio de dados para um servidor.

1.3 ORGANIZAÇÃO DA MONOGRAFIA

Este trabalho está estruturado em 6 capítulos de acordo com a separação descrita abaixo.

Primeiro depara-se com a introdução geral do trabalho, apresenta-se a motivação do estudo, bem como o objetivo geral, objetivo específicos a metodologia e a organização da monografia.

O segundo capítulo abrange o referencial bibliográfico. A revisão bibliográfica inicia com uma breve explicação sobre Message Queuing Telemetry Transport (MQTT), além de abranger o conceito do protocolo 12C. Traz a definição e uma breve analise de programação de Microcontroladores.

No terceiro capítulo é apresentado o método de estudo e a forma que ele será utilizado para alcançar os objetivos propostos. O quarto capítulo trata sobre o protótipo desenvolvido, suas características, suas funções e seus componentes. No quinto capitulo são apresentados dados do consumo e custos.

O sexto capitulo traz conclusões obtidas com este estudo e também são apresentadas sugestões de futuras implementações para o trabalho.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão abordados os principais elementos de um sistema IoT e as tecnologias usadas para o desenvolvimento do projeto. Num sistema IoT o elemento gerenciador é *Message Queuing Telemetry Transport* (MQTT) que tem como função rotear informações de seus clientes. Os clientes são denominados *publisher* e/ou *subscriber* e recebem e mandam informação via TCP/IP. Além do mais, o microcontrolador, que será o cliente do servidor MQTT, receberá informações de sensores através do protocolo I2C e sensores analógicos, através das portas que aceitem uma entrada *Analog to Digital Converter* (ADC). Nas próximas seções são descritos cada um desses elementos.

2.1 *INTERNET OF THINGS* (IOT)

A Internet das coisas (IoT) já não é um assunto tão novo, em 1999 utilizou-se pela primeira vez o termo por Kevin Ashton, cofundador do *Auto-ID Center do Massachusetts Institute of Technology* (MIT). Conforme a Associação Brasileira de Internet das Coisas, ABINC (2017), a IoT é um termo "'guarda-chuva' que abrange diferentes tecnologias e conceitos com implicações profundas nos negócios e na vida da sociedade e, em linhas gerais, ela pode ser resumida como a capacidade de "Capturar, Analisar e Agir por meio de dados gerados por objetos e máquinas conectados à Internet".

Para alguns autores como Rozsa (2017, p. 257) "a IoT pode ser considerada a nova geração tecnológica que pode elevar nosso campo tecnológico a novos patamares vários avanços tecnológicos permitiram o surgimento da IoT". As interconexões em massa entre as pessoas aconteceram com a chegada da Internet o próximo ciclo a acontecer dar-se-á de objetos interconectados para criar ambientes inteligentes.

As Tecnologias da IoT estão empregadas em alguns avanços tecnológicos, alocados nas redes de sensores sem fio, comunicação móvel e a computação ubíqua, bem como a diversidade de ambientes de tecnologias, caracterizando-se em um conjunto extenso de tecnologias e casos de uso sem uma definição precisa, necessitando de ferramentas como: sensores para detecção de eventos; protocolos de comunicação; organização, armazenamento, representação e recuperação de informações; fusão e processamento de dados; entre outras.

A IoT caracteriza-se no uso de dispositivos conectados em rede, incorporados ao ambiente físico, afim de melhorar alguns processos existentes e ou ativar um novo panorama o qual antes não era possível. Minerva, Biru, Rotondi (2015, p 27)

> [...] é parte integrante da Internet do Futuro e pode ser definida como uma infraestrutura de rede global dinâmica com recursos de autoconfiguração baseados em padrões e protocolos de comunicação interoperáveis onde "coisas" físicas e virtuais têm identidades, atributos físicos e personalidades virtuais e usar interfaces inteligentes, e são perfeitamente integrados na rede de informação.

Para Minerva, Biru, Rotondi (2015) acreditam que na IoT as "coisas" devem se tornar ativas participantes em negócios, informações e processos sociais, capazes de interagir e comunicar entre si e com o ambiente, trocando dados e informações "sentido" sobre o meio ambiente, reagindo de forma autônoma aos eventos do "mundo real / físico", dessa forma influenciaram e executaram processos que acionam ações e criam serviços com ou sem intervenção humana direta. Para os autores “Interfaces na forma de serviços facilitam as interações com essas “coisas inteligentes" pela Internet, consultar e alterar seu estado e qualquer informação associada com eles, levando em conta questões de segurança e privacidade”. (MINERVA, BIRU, ROTONDI. 2015 p. 27).

Nesse contexto os dispositivos, ou *coisas*, conectam-se à rede para fornecer informações que coletam do ambiente por sensores ou para permitir que outros sistemas contatem e influenciem o mundo. Com isso, podem ser conectadas através de objetos comuns, dispositivos novos e ou específicos, utilizam-se dispositivos que que estejam em posse da pessoa que esta no ambiente, ou podem ser incorporados a equipamentos de fábrica, fazendo parte de algum programa que a cidade ofereça para a segurança pública, identifica-se a aplicação das tecnologias da IoT e a geração de soluções para desafios em diferentes contextos.

Nos diferentes contextos, os dispositivos deverão ser capazes de converter informações valiosas do mundo real em dados digitais que proporcionam maior visibilidade sobre como os usuários interagem com produtos, serviços ou aplicativos. Para Minerva, Biru, Rotondi (2015 p 27).

> A Internet das Coisas (IoT) é parte integrante da Internet do Futuro e pode ser definida como uma infraestrutura de rede global dinâmica com recursos de autoconfiguração baseados em padrões e protocolos de comunicação interoperáveis onde "coisas" físicas e virtuais têm identidades, atributos

> físicos e personalidades virtuais e usar interfaces inteligentes, e são perfeitamente integrados na rede de informação.

Com a IoT surge um mundo de oportunidades específicas em diferentes áreas, por mais que seja um tema que já é explorado, em algumas áreas, o mundo da IoT está apenas começando. Para Zanella *et al*. (2014) a IoT apresenta-se como uma das principais tecnologias emergentes que contribuem para concretizar novos domínios de aplicação das tecnologias de informação e comunicação e sensoriamento apontando serviços de valor agregado para os órgãos administrativos de tais cidades e para seus cidadãos.

Este cenário caracteriza-se em um conjunto de desafios e padrões comuns o desenvolvimento dos projetos de IoT têm dimensões ímpares, a complexidade dos projetos aumenta em comparação com outros aplicativos de tecnologia centrada na nuvem, os quais são necessários observar: *hardware* diferente; sistemas operacionais e *software* diferentes nos dispositivos; requisitos de *gateway* de rede diferentes. A computação em nuvem proporciona a utilização destes serviços sem a necessidade de investimento em equipamento físico necessário para o armazenamento das ferramentas de análise e descoberta de padrões e sistemas gerenciadores de banco de dados. (BRITO, 2017. p. 29)

O contínuo desenvolvimento e a adoção da IoT vêm ganhando espaço referente à pesquisas e mercado, dessa forma exigem cada mais inovação e observação nas plataformas de dispositivos utilizadas em projetos.

2.1.1 Plataformas de dispositivo

Com o passar do tempo, a IoT possibilitará o surgimento e evolução de ambientes existentes que incorporam IoT em seus processos de funcionamento, ambientes (locais) especializados caracterizando-se em um espaço inteligente. “Há uma série de desafios a serem superados para alavancar a ampla disseminação da IoT, principalmente com relação ao desenvolvimento de aplicações e à alta heterogeneidade decorrente da inerente da diversidade de tecnologias de *hardware* e *software* desse ambiente”. (PIRES, 2015 *et al.* p.02)

Existe uma diversidade de hardware específicos disponíveis para criar aplicativos de IoT, percebe-se esta diversidade com as opções de plataformas de hardware, em que cada plataforma permite conectar vários tipos de módulos de

sensor e atuador por uma interface de *hardware*. São várias as características de comunicação da IoT, dessa forma é

> [...] necessário prover mecanismos para o gerenciamento de dispositivos, que diz respeito à capacidade de fornecer informações de localização e estado do dispositivo permitindo, dentre outras funcionalidades, desconectar algum dispositivo roubado ou não reconhecido, atualizar software embarcado, modificar configurações de segurança, modificar remotamente configurações de hardware, localizar um dispositivo perdido, apagar dados sensíveis de dispositivos, e até mesmo possibilitar a interação entre dispositivos. (PIRES, 2015, p. 5)

As plataformas interagem, as informações necessitam ser coletadas e processadas com o objetivo de efetuar ações ou reagir a estímulos com base nos dados extraídos, interagindo com os módulos, em uma abordagem de camadas semelhante à da computação. “As plataformas são responsáveis pela coleta, gerenciamento e processamento de informações de contexto providas por múltiplas fontes, liberando as aplicações e usuários da tarefa de manipulá-las e tornando transparente tal manipulação”. (PIRES, 2015, p.5) Para simplificar, o diagrama na Tabela 1 a seguir omite o sistema.

Tabela 1- Diferenças do desktop e IoT

	DESKTOP	**IOT**
APLICAÇÃO	Google Earth	Automação residencial
DRIVER	HID	MRAA
INTERFACE	USB	I2C
PERIFÉRICOS	Mouse	Acelerômetro

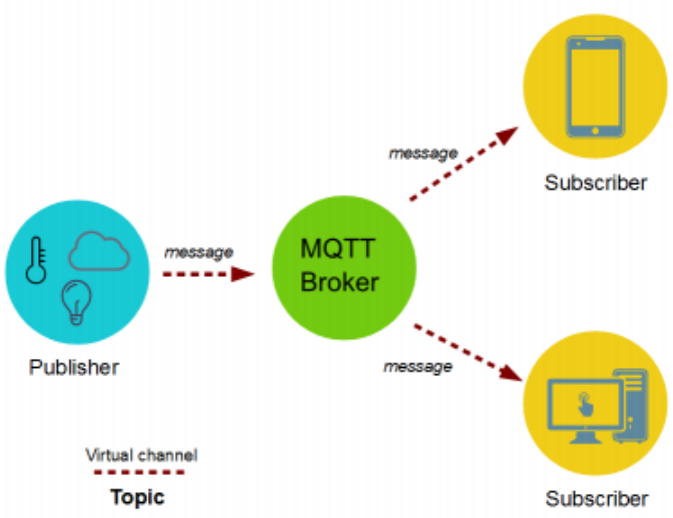
Fonte: (https://cloud.google.com/solutions/iot-overview, 2018)

2.2 MESSAGE QUEUING TELEMETRY TRANSPORT (MQTT)

Desenvolvido em 1999 pela IBM com o objetivo de ser um protocolo mais eficiente que os existentes do ponto de vista de largura de banda e de energia, o *Message Queuing Telemetry Transport (*MQTT) é um protocolo (padrão ISO/IEC PRF 20922) de mensagens leves baseado em publicação e subscrição (*Publish and*

Subscriber) que roda sob o protocolo TCP/IP, como indica a figura 1 com um exemplo de topologia.

Figura 1- Topologia de rede MQTT



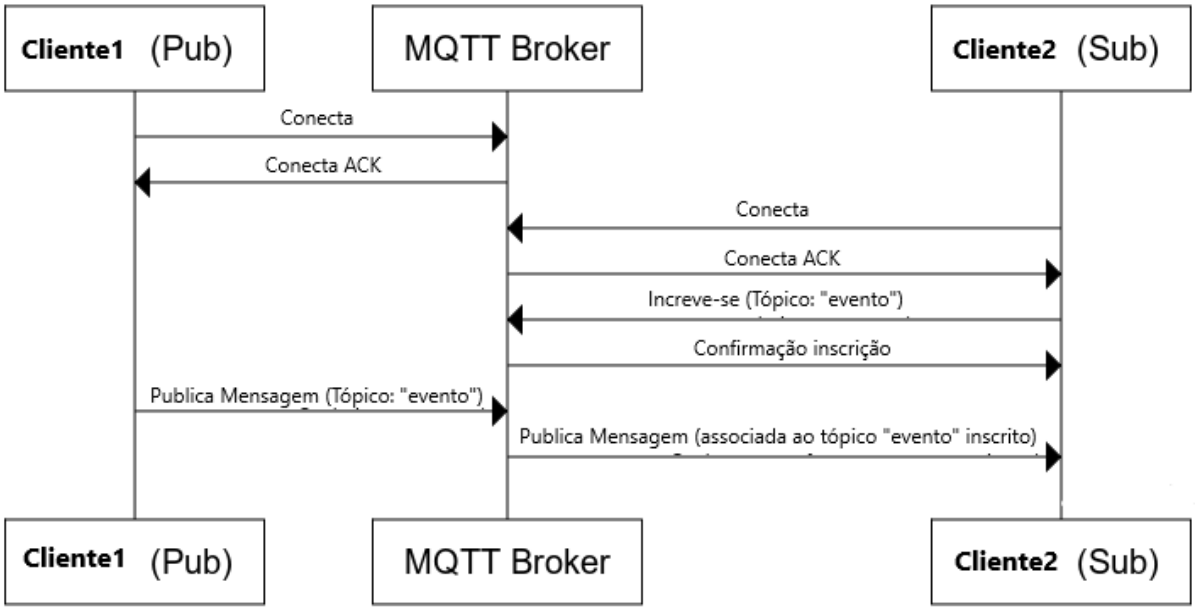
Fonte: (Azzola, 2015)

O padrão de troca de mensagens no MQTT é o *publish/subscriber* (publicador/subscritor). Neste padrão, quando um elemento da rede deseja receber uma determinada informação, ele a subscreve, fazendo uma requisição para um outro elemento da rede capaz de gerir as publicações e subscrições. Na rede MQTT este elemento é conhecido como broker, o intermediário no processo de comunicação. Elementos que desejam publicar informações o fazem também através do broker, enviando-lhe as informações que possuem. Esse padrão não é novo e existe em outros protocolos. Por exemplo, a troca de informação de controle (links) em redes *Foundation Fieldbu segue* o paradigma *publish/subscriber.*

A conexão do cliente ao *broker*, seja ele subscritor ou publicador, é originalmente feita via TCP, com opções de *login* (usuário e senha) e uso de criptografia (SSL/TLS). É possível encontrar também outros meios físicos, com MQTT rodando em links seriais, por exemplo.

A figura 2, mostra como funciona o protocolo MQTT, um cliente envia o pedido de conexão e recebe um Ack de confirmação, a partir desse momento ele está conectado com o servidor Broker, e publica uma mensagem no tópico *events*.

Figura 2 - Passo a passo para uma publicação MQTT



Fonte: (PIERRE-LUC, 2015)

Outro cliente, também faz requisição da conexão e conecta-se ao servidor, e se pede para ter acesso ao tópico *events*, se o login estiver certo, e seguindo algumas medidas de segurança, recebe uma confirmação que está inscrito (*subscribe*).

A partir desse momento, sempre que uma mensagem for enviada para o servidor, todos os inscritos nela, irão receber, utilizando-se de 3 formas para enviar os dados QoS *(Quality of Service)*.

QoS 0: (*at most once*)

QoS 1: (*at least once*)

QoS 2: (*exactly once*)

Não existe um QoS melhor ou pior, isto irá depender de cada cenário de aplicação do MQTT, da qualidade do link de comunicação, dos recursos disponíveis no seu sensor, etc. E, é importante ressaltar, cada nível de QoS é negociado entre o cliente e o broker (pequeno comerciante), não entre o publicador e o subscritor. Assim, é possível ter uma publicação em QoS 0 e uma subscrição em QoS 2, para um mesmo tópico.

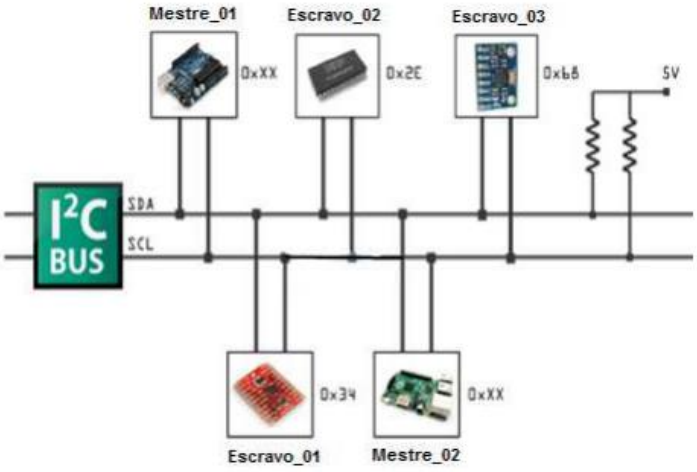
O MQTT foi projetado para redes não confiáveis e de alta latência, destinado principalmente para sensores e pequenos dispositivos móveis com uma pequena

largura de banda de conexão com a internet. A conexão do cliente ao broker possui uma opção de login (usuário e senha) e uso de criptografia (SSL/TLS). O padrão de troca de mensagens no MQTT é o publicador/subscritor. Neste padrão, quando um nó da rede deseja receber uma determinada informação, ele deve realizar uma subscrição a um broker em um determinado tópico. Quando outro nó da rede realiza uma publicação para o broker, ele repassa a mensagem de publicação todos os nós que estiverem subscritos naquele tópico. Apesar do broker parecer como um elo fraco na rede ao centralizar as comunicações, ele permite um desacoplamento entre os nós que estão trocando informações, algo não possível em modelos de comunicação do tipo cliente/servidor.

2.3 COMUNICAÇÃO I2C

I2C é a sigla de *Inter-Integrated Circuit*, descreve o funcionamento de um barramento de comunicação serial que utiliza apenas dois fios, inventado pela Philips no início da década de 90, este protocolo é muito utilizado para conectar periféricos de baixa velocidade a placa-mãe, microcontroladores e afins.

Figura 3 - Diagrama básico de conexão do barramento 12C



Fonte: Rodrigues (2014)

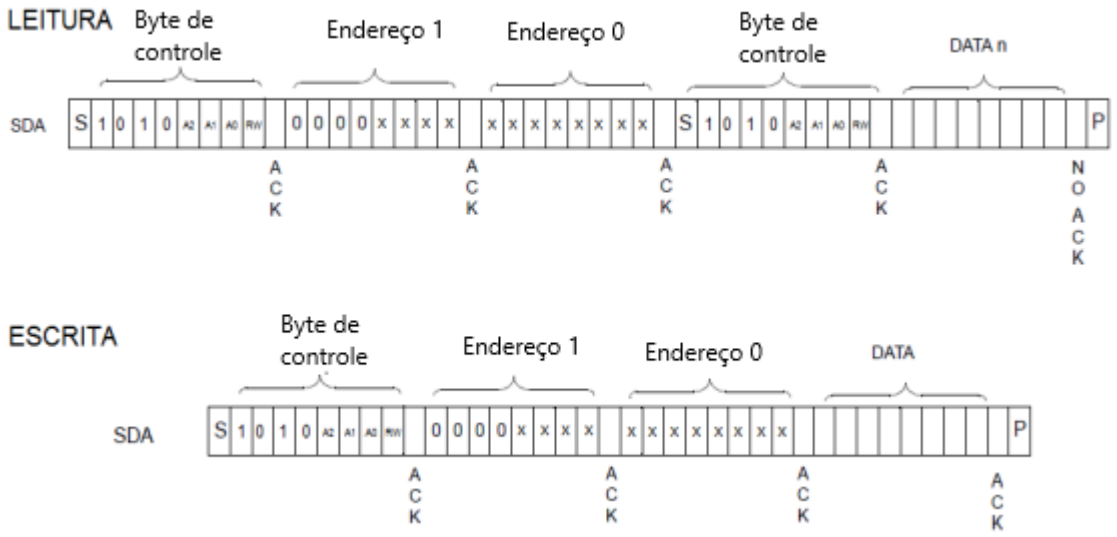
Como é possível a comunicação de um ou mais dispositivos usando apenas dois fios? Para isso, cada periférico deve apresentam uma ID (número de

identificação) predefinido, ou o endereço de um dispositivo exclusivo, dessa forma o mestre da comunicação pode escolher com que estará se comunicando.

Os dois fios são chamados de Serial Clock (SCL) e Serial Data (SDA). A linha SCL é o sinal de relógio, responsável por sincronizar a transferência de dados entre os dispositivos conectados ao barramento I2C, o sinal SCL é gerado pelo mestre. A outra linha, SDA é responsável pelo transporte de dados (Figura 10).

O protocolo de comunicação I2C funciona da seguinte forma, o sinal com os dados é transferido em uma sequência de 8 bits. Dessa forma, após a chegada de uma condição de início, vem uma sequência de 8 bits que indica o endereço do dispositivo escravo para qual os dados estão sendo enviados. Depois do endereço do escravo, vem 1 bit, chamado de *Acknowledge*. Os próximos 8 bits indicam, na maioria dos casos, uma sequência de endereçamento interna do dispositivo escravo. Posteriormente, os próximos bytes (sequência de 8 bits) representam os dados que são estão sendo transferidos, até a chegada da condição de parada (Stop bit). A Figura 4 representa o diagrama de blocos básico da comunicação I2C.

Figura 4- Ilustrações básicas para a escrita e leitura usando 12C



Fonte: Rodrigues 2014

2.4 INTERFACES DE HARDWARE

Muitos produtos físicos (projetados e criados) na IoT são impulsionados por interfaces digitais e serviços baseados em dados. Nesse contexto, o produto necessita atender aos usuários finais, onde os fornecedores trabalharão, de forma conjunta para construir e nutrir o complexo ecossistema.

Projetar, construir e nutrir o complexo ecossistema, demanda de interfaces de interação com o usuário na IoT, as superfícies inteligentes, em que praticamente qualquer superfície em uma construção inteligente pode tornar-se uma interface. Rozsa (2017, p. 217) aponta que “é característica da IoT de suportar diferentes formas de interação com o usuário pode ser explorada”. Por exemplo, uma interface para controle por voz e interfaces capazes de obter feedback visual do usuário, entre outras.

Na IoT a maioria das interfaces de hardware são interfaces seriais, nas interfaces seriais utilizam-se vários fios, que têm a finalidade de controlar o fluxo e a temporização das informações binárias, ao longo do fio de dados principal. Existem vários tipos de interface de hardware o qual define um método de comunicação entre um periférico e o processador central. As plataformas de hardware de IoT usam várias interfaces comuns, onde os módulos de sensor e atuador podem aceitar uma ou mais dessas interfaces, como por exemplo:

O Universal Serial Bus (USB) conforme Morimoto (2018, p. 212) **“**é o primeiro barramento para micros PC realmente *Plug-and-Play*” pode-se conectar periféricos com o aparelho ligado, basta fornecer o driver do dispositivo para funcionar sem ser necessário nem mesmo reinicializar o aparelho. A controladora USB também é suficientemente inteligente para perceber a desconexão de um periférico.

Os pinos do tipo *General-Purpose Input/Output* (GPIO) são conectados diretamente ao processador. O periférico GPIO fornece pinos dedicados de uso geral que podem ser configurados como entradas ou saídas. Quando configurado como uma saída, você pode gravar em um registro interno para controlar o estado o pino de saída. Quando configurado como uma entrada, você pode detectar o estado da entrada lendo o estado de um registro interno. (TEXAS INSTRUMENTS 2008, p.7)

O GPIO digital aceita modulação por largura de pulso (PWM, na sigla em inglês). A PWM permite ligar e desligar uma fonte de alimentação muito rapidamente,

sendo que cada fase "ligar" é um pulso de duração específica, ou *largura*. O efeito no dispositivo pode ser um nível de potência mais baixo ou mais alto. Por exemplo, você pode usar a PWM para alterar o brilho de um LED. Quanto mais amplos os pulsos "ligar", mais intenso o brilho do LED.

Os pinos analógicos têm acesso a um circuito integrado de conversão analógico-digital (ADC, na sigla em inglês). Um ADC faz amostras periodicamente de uma forma de onda analógica e contínua, como um sinal de áudio analógico, concedendo a cada amostra um valor digital entre zero e um, em relação à tensão do sistema.

Quando você lê o valor de um pino digital de E/S no código, ele precisa ser ALTO ou BAIXO, sendo que um pino de entrada analógico pode ser de qualquer valor em um intervalo, em qualquer momento. O intervalo depende da resolução do ADC. Mais valores significa maior resolução e, portanto, uma representação digital mais fiel de qualquer sinal analógico.

A *taxa de amostragem* do ADC determina o intervalo de frequência que um ADC pode reproduzir. Uma taxa de amostragem maior resulta em uma maior frequência máxima nos dados digitais.

O barramento *Inter-Integrated Circuit* (I2C) usa um protocolo que permite atribuir um endereço separado no barramento a vários módulos. I2C às vezes é pronunciado como "I dois C", "I-I-C" ou "I ao quadrado C".

O circuito que coordena o envio e recebimento de dados através da porta serial. (Morimoto, 2018, 461) dispositivos com padrão *Universal Asynchronous Receiver/Transmitter* (UART) converte dados seriais em paralelos e vice-versa no ponto em que os dados são usados pelo processador. O UART é necessário quando é preciso dispor os dados seriais na memória de maneira paralela.

2.5 MICROCONTROLADOR

O microcontrolador é um CI (circuito integrado) que consiste de um conjunto de alguns a até milhares de transistores reunidos em um mesmo encapsulamento (OLIVEIRA e ANDRADE, 2010). Esses CI são capazes de efetuar processos lógicos com extrema rapidez e precisão. Sua grande vantagem é a possibilidade de se poder programar seu conteúdo, tornando muito adaptável a finalidade desejada (ASSIS, 2004).

Conforme Assis (2004) explica, o microcontrolador pode ser visto como um pequeno componente eletrônico que possui certo tipo de inteligência. Essa inteligência permite que o programador determine qual a sua finalidade.

O microcontrolador é empregado para gerenciamento de processos lógicos, estando englobados dentro desses processos as operações lógicas e matemáticas e o controle de periféricos, tais como sensores, relês, resistências, LCD (*Liquid Crystal Display* ou *Displays* de Cristal Líquido), *displays, LEDs (Light Emitting Diodes* ou Diodos Emissores de Luz), entre outros.

O microcontrolador difere do microprocessador em muitos sentidos, sendo sua maior distinção a sua funcionalidade, pois o microprocessador é dependente de periféricos, como memória e componentes para entrada e saída de dados. Assim, um microprocessador é o núcleo de um computador, tendo que ser assessorado por vários periféricos. Já o microcontrolador é projetado para operar independente de periféricos. Dessa forma, o microprocessador apresenta um baixo custo e maior autonomia, em detrimento de uma menor potência de processamento, menor espaço de armazenagem e menor velocidade de *clock*. (ASSIS, 2004).

Consoante aos ensinamentos de Oliveira e Andrade (2006), quanto maior a complexidade da instrução, maior é o espaço ocupado no chip, podendo chegar a um ponto em que o conjunto de instruções é tão extenso que compromete o desempenho. Com um conjunto de instruções extenso, essas podem ser pouco utilizadas ou não utilizadas durante a execução do programa, tornando mais vantajosa a sua construção por outras instruções mais básicas. É preciso ressaltar a maior flexibilidade de programação (maior quantidade de instruções disponíveis, 24 promovendo maior quantidade de soluções) na utilização de um conjunto de instruções mais completo.

Ainda segundo Oliveira e Andrade (2006), na arquitetura RISC, as instruções são simplificadas ao máximo, possuem formato fixo e devem durar, teoricamente, um ciclo de *clock* na sua execução. Porém, cumprir uma tarefa em apenas um ciclo é algo praticamente impossível, pois o acesso às memórias é um fator de grande gargalo no circuito. As instruções que dependem apenas do acesso a registradores (espaço reservado de memória) e armazenam os resultados em registradores podem ser executadas em um ciclo, mas as instruções que são carregadas a partir da memória ou armazenadas em memória levam mais ciclos para serem concluídas.

Na Tabela 1, pode-se ver um comparativo entre 3 microcontroladores, um ESP32, um ESP8266 e um ATMEL ATMEGA328, nesse caso o ArduinoUNO. É fácil notar pela tabela a diferença que existe entre eles, enquanto o Arduino possui apenas 1 núcleo, com uma arquitetura de 8 bits, o ESP32 possui 2 núcleos e 32bits, além disso a velocidade dos dois varia em mais de 10 vezes, sendo o ESP32 tendo 160MHz de Clock, enquanto o Arduino tem apenas 16MHz. Outra grande vantagem do ESP32 é a presença de módulos Wi-Fi e Bluetooth integrados, que possibilitam uma gama maior de aplicações, principalmente para o uso de circuitos integrados voltados a IoT.

Tabela 2 - Comparativo entre um microcontrolador PIC e um AVR

	ESP32	**ESP8266**	**ARDUINOUNO R3**
CORES	2	1	1
ARQUITETURA	32 bits	32 bits	8 bits
CLOCK	160 MHz	80 MHz	16 MHz
WIFI	Sim	Sim	Não
BLUETOOTH	Sim	Não	Não
RAM	512KB	160KB	2KB
FLASH	16Mb	16Mb	32KB
GPIO	36	17	14
INTERFACES	SPI / I2C / UART/ I2S/ CAN	SPI / I2C / UART / I2S	SPI / I2C / UART
ADC	18	1	6
DAC	2	0	0

3 DESENVOLVIMENTO DO TRABALHO

Neste capítulo serão descritos os equipamentos utilizados, bem como os *softwares* e configurações necessárias para realizar o projeto.

a) Um computador com Ubuntu.

b) Gravador *in-circuit Silicon Labs* CP210x USB to UART para gravar no ESP32

c) Dois módulos ESP32 com conexão UART

d) Osciloscópio e analisador lógico USB *Analog Discovery da DIGILENT*;

Os principais softwares utilizados neste projeto foram:

a) *gcc-arm-none-eabi*, compilador e montador de linguagem C. Utilizado nas compilações dos *firmwares* para o módulo ESP32.

b) *WaveForms31*, para análise das leituras do Analog Discovery;

c) *Adafruit* IO MQTT, *broker* da comunicação MQTT;

3.1 ESP 32

O ESP 32 é um dos mais avançados microcontroladores da *Espressif*. Está integrado com antenas, amplificadores. O *design* compacto inclui memória Flash e tem ESP32SoC e PCB antena para uma performance melhor para RF. ESP32 é bem conhecido pela habilidade hibrida de funcionalidades que consistem em *Bluetooth* e *WiFi.*

Figura 5- Módulo ESP32

Fonte: Espressif

Tem suporte para codificação avançada WPA/WPA2 e WEP, para questões de segurança. Para ambientes industriais também pode dispor de mais confiabilidade, porque pode se adaptar a mudanças do clima. Operando a temperaturas num raio de

-40ºC até +120ºC. Pode se comunicar com outros dispositivos pela interface I2C/UART ou SPI/SIDO. Tem sensores embutidos como sensor *Hall*, amplificador analógico Ultra *low noise* e interface *touch*. Esp32 é principalmente desenvolvido para aplicações de baixo consumo de energia, como eletrônicos industriais baseados em IoT.

O ESP32 é o nome de um micro controlador projetado pela *Espressif Systems*. *Espressif* é uma empresa chinesa sediada em Xangai. O ESP32 anuncia-se como uma solução de rede *WiFi* que se oferece como uma ponte entre os microcontroladores existentes para *WiFi* e *Bluetooth*.

Também é capaz de executar aplicativos independentes. A produção de volume do ESP32 não começou até o final de 2016, o que significa que, em IoT, esta é uma nova entrada na linha de processadores.

3.2 RECURSOS DO ESP32

Ele possui dois Microprocessadores Xtensa® 32-bit LX6 com até 600 DMIPS (velocidade de processamento). A frequência do *clock* pode ser de até 240 MHz, dependendo do modelo. Frequência mais comum – 160 MHz.

- memória ROM interna de 448K Bytes (para Boot e Core)
- memória RAM estática interna de 520K Bytes
- Memória externa (total 4) – suporte para até 16M Bytes Flash e 16M Bytes SRAM
- 1 K Bit de Fusíveis eletrônicos (para segurança e criptografia)
- Real Time Clock com 16K Bytes de SRAM
- Interface WIFI 802.11 b/g/n – 802.11 n (2.4 GHz), até 150 Mbps
- Interface Bluetooth v4.2 BR/EDR e Bluetooth LE (low energy)
- Dois grupos de *Timers* – 4 *timers* de 64 Bits
- Aceleradores de *hardware* (criptografia) – AES, SHA, RSA e ECC
- Alimentação VCC de 2,3V a 3,6V CC
- Consumo de corrente máxima com Wifi – 240 mA

Tabela 3- Recursos do microcontrolador

ATRIBUTO	DETALHE
VCC	3,3V
CONSUMO DE CORRENTE	0,2μA até 240mA
PROCESSADOR	Temsilica L108 32 bit
VELOCIDADE DO PROCESSADOR	Dual core 160MHz
RAM	520K
GPIOS (ENTRADAS/SAIDAS)	34
ANALOG DIGITAL CONVERTER (ADC)	7
802.11 SUPPORT	11b/g/n/e/i
BLUETOOTH	BLE
MAXIMUM CONCURRENT TCP CONNECTIONS	16
SPI	3
I2S	2
I2C	2
UART	3

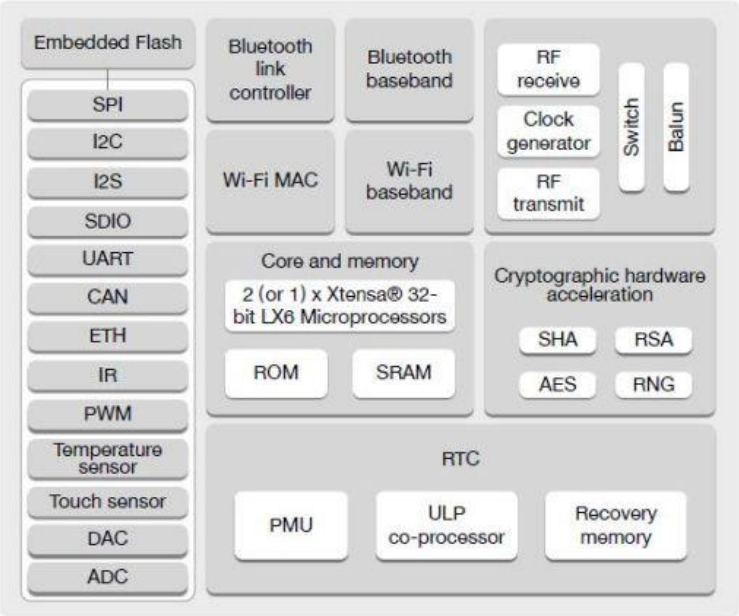
Fonte: Autor

Periféricos do Chip ESP32

- 34 × Portas programáveis GPIOs
- 2 x Conversores ADC SAR 12-bits com até 18 canais
- 2 × Conversores DAC de 8-bits
- 10 × sensores de toque
- Sensor de Temperatura
- 4 × interfaces SPI – clock ate 40 MHz!
- 2 × interfaces I2S – clock até 40 MHz!
- 2 × interfaces I2C– até 5 Mbps
- 3 × interfaces seriais UART – até 5 Mbps!
- 1 *Host* (SD/eMMC/SDIO) para controle de SD *Cards*
- 1 *Slave* (SDIO/SPI)

- Interface Ethernet MAC (necessita acessório)
- Interface CAN 2.0
- Interface infravermelha (Tx/Rx)
- Controle de Motor PWM
- Controle de LED PWM até 16 canais
- Sensor interno Hall

Figura 6- Diagrama em Blocos do Chip ESP32



Fonte: Espressif

A questão de determinar quanto tempo o ESP32 pode durar usando baterias é um ponto bem interessante. O consumo atual está longe de ser constante. Ao transmitir com potência total, pode consumir 260mA, mas quando está em sono profundo, precisa apenas de 20uA. Isso é muita diferença.

Isso significa que o tempo de execução de um ESP32 em um reservatório de corrente fixa não é apenas uma função do tempo, mas também do que está fazendo durante esse tempo e isso é um função do programa implantado sobre ele.

3.2.1 *DeepSleep*

Se o dispositivo ESP32 estiver constantemente ligado, consome constantemente corrente. Se a fonte de energia é constante, então isso não precisa necessariamente ser um problema, no entanto, quando funcionando com baterias ou outro suprimento finito, podemos precisar minimizar o consumo.

Uma maneira de conseguir isso é suspender o funcionamento do dispositivo quando não estiver em uso. Quando o dispositivo está suspenso, a noção é que o consumo será reduzido. Tem três modos de sono definidos. Estes são chamados de modem-sleep, light-sleep, deep-sleep.

Ao observar a tabela a seguir, podemos ter uma noção das habilidades em cada modalidade.

Tabela 4- Modo de redução de consumo de energia

FUNÇÃO	MODEM	LIGHT	DEEP
WIFI	Off	Off	Off
CLOCK DO SISTEMA	On	Off	Off
RTC	On	On	On
CPU	On	On	Off
CONSUMO DE CORRENTE	15mA	0,5mA	20μA

Fonte: Autor

O modem-sleep só pode ser usado quando o ESP32 está no modo de estação conectado um ponto de acesso. A aplicação deste modo é quando o ESP32 ainda precisa executar trabalho, mas minimiza a quantidade de transmissões sem fio. O modo de suspensão Light é o mesmo que o modem-sleep, mas neste caso os relógios serão suspenso.

No modo DeepSleep, o dispositivo está realmente adormecido. Nem CPU nem atividades WiFi, maior parte da RAM, e todos os periféricos digitais com clock estão desligados. O dispositivo esta desligado para todos os eventos, com uma exceção, as únicas partes do chip que ainda podem ser ligadas são: controlador RTC, periféricos RTC (incluindo o coprocessador ULP) e memórias RTC.

E com acesso a esse bloco, é possivel despertar o ESP32 quando se encontra no modo DeepSleep, pode se configurar um tempo pré programado pra quanto tempo ficar adormecido, ou configurar para que monitore uma das portas dos periféricos RTC, que ao notar variação na entrada, pode despertar novamente o ESP32. Sendo estas fontes de despertar citadas podendo ser configuradas a qualquer momento antes de entrar no modo DeepSleep.

3.3 SENSOR MQ-GAS

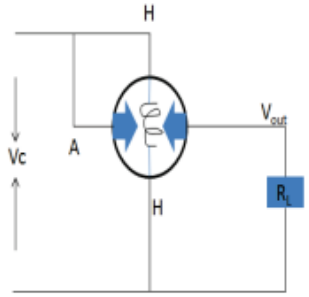
Este sensor de gás analógico - MQ2 é usado em equipamentos de detecção de vazamento de gás em eletrônicos de consumo e mercados industriais. Este sensor é adequado para a detecção de GLP, I-butano, propano, metano, álcool, hidrogênio e fumaça. Tem alta sensibilidade e resposta rápida.

Na concepção destes dois sensores, o dióxido de estanho (SnO2) é usado como camada de detecção de gás, Au, Pt são usados como eletrodos, liga de Ni-Cr é usada como bobina de aquecimento, SUS36 100- malha (gaze de aço inoxidável) é usado como rede anti-explosão e baquelite é usado como base de resina.

No ar limpo, o material de sensibilidade SnO2 do sensor de gás MQ inicialmente tem menor condutividade. Se o sensor detectar o gás alvo, o sensor aumenta a condutividade do material proporcionalmente com a concentração de gás. A variação na condutividade, por sua vez convertida em variação na tensão sobre a resistência de carga. A resistência do sensor (Rs) pode ser medida a partir da fórmula abaixo indicada.

$$R_s = \left(\frac{V_C}{V_{RL}} - 1\right) * R_L \quad (1)$$

Figura 7 - Circuito Interno do MQ2



Fonte: Datasheet

As características do sensor MQ é mostrado na Tabela 5 abaixo.

Tabela 5- Características do MQ2

ESPECIFICAÇÃO	MQ-2
SENSING RESISTANCE	2KΩ-20KΩ
VARIAÇÃO (PPM)	300-1000ppm
GASES DETECTADOS	LPG, Propano, CO, Metano
APLICAÇÕES	Sistema de alarme detecção de fumaça
CONDIÇÃO DE TEMP E UMIDADE	18ºC to 20ºC 60% to 70%
CONSUMO DE ENERGIA	≤ 900mW

Fonte: Autor

3.3.1 Sensibilidade do sensor de gás MQ2

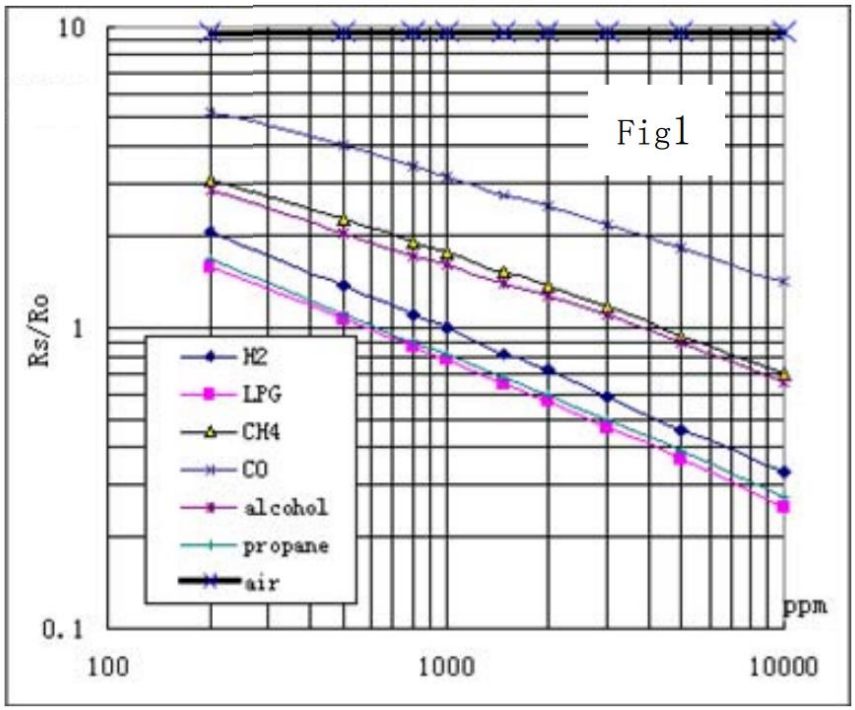
A resistência (Rs/Ro) do sensor é mais alta ou mais baixa de acordo com a concentração de gás (ppm) existente. Esta concentração pode ser visualizada pela saída do pino A0.

O sensor de gás MQ-2 tem alta sensibilidade aos gases:

- LPG (Gás liquefeito de petróleo);
- Propano (C3H8);
- Hidrogênio (H2);
- Metano (CH4);
- Gases combustíveis, como o Propano, CO, Álcool e Butano (utilizado em isqueiros).

Na Figura 8, a sensibilidade para diferentes tipos de gás varia para cada um, sendo o Propano e o LPG mais sensíveis.

Figura 8- Características sensitivas



Fonte: Datasheet

3.4 SHT 30 TEMPERATURA E UMIDADE

Outro sensor a fazer parte desse protótipo é o SHT30, um sensor de Temperatura e umidade que através do barramento I2C, irá dar dados importantes para averiguação do local, e cuidar contra variações bruscas na temperatura e umidade do ambiente.

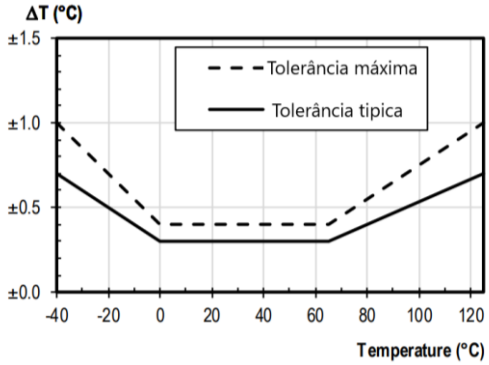
Ele permite ser alimentado de 2.4V até 5.5V, comunicação I2C com velocidade de comunicação alcançando 1MHz e inicializa rapidamente, já permitindo acesso rápido aos dados.

O sensor mostra melhor performance quando está operando em temperatura e umidade recomendadas, numa faixa de valores de 0 ºC – 60 ºC e 20 %RH – 80 %RH, respectivamente, como pode ser visto na Figura 9 e na Figura 10, retirada do datasheet do SHT30, mostrando o gráfico de sensibilidade da temperatura e umidade, respectivamente.

Existem outros sensores como SHT31 e SHT35 que fornecem uma faixa de valores tolerados de sensibilidade maior, porém, a propósitos de projeto, foi usado

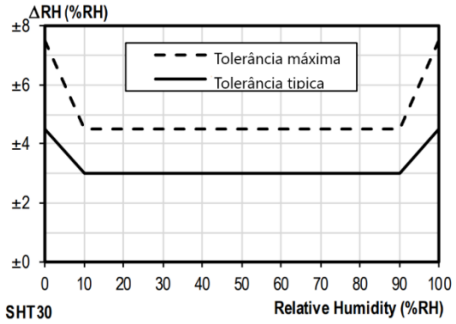
uma versão mais econômica e como todos eles operam de formas similares, então a substituição não afetaria a sua função.

Figura 9 - Sensibilidade da Temperatura



Fonte: Datasheet

Figura 10 - Sensibilidade da umidade



Fonte: Datasheet

Valores estes dentro do padrão para experimentos, que serão monitorados continuamente, caso aconteça uma variação brusca na temperatura, poderá ser um indicio de incêndio, e somado com os outros 2 sensores do circuito, ter uma confiança maior no resultado a ser enviado para o servidor MQTT, e posteriormente, enviando alertas a alguma autoridade competente.

A Tabela 6 mostra as especificações elétricas do sensor, válidas para 25ºC, detalhando valores de tensão de alimentação, consumo, e temperatura que pode dissipar durante a operação.

Tabela 6 - Especificações elétricas do sensor

PARÂMETROS	SIMBOLO	CONDIÇÃO	IN.	TÍPICO	MÁX	UNID.
TENSÃO DE ENTRADA	V_{DD}		.4	3.3	5.5	V
CORRENTE DE ENTRADA	I_{DD}	Modo ocioso (aquisição de dados periódicos)	-	45	7	µA
		Medindo	-	800	1500	µA
		Média	-	2	-	µA
POTÊNCIA DISSIPADA	P_{HEADER}	Temperatura operando	4.5		33	mW

Fonte: Autor

3.5 ADAFRUIT

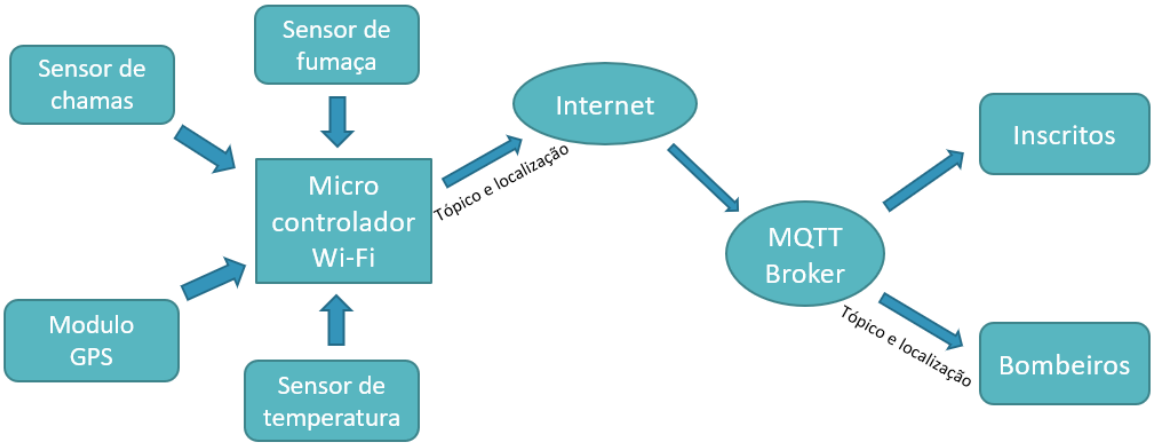
O *Adafruit* IO é um serviço na nuvem que torna os dados lidos pelo microcontrolador úteis. É bem conhecido pela facilidade de uso e permite conexões de dados simples e com pouca programação. As bibliotecas do cliente que envolvem a APIs do MQTT estão disponíveis para receber e enviar dados com o Adafruit IO. Pode ser construído em *Node.js*, C++ e *Ruby on Rails.* A *Adafruit* MQTT *Library* é uma biblioteca cliente muito popular do MQTT, usada para o Arduino IDE e C++ para acessar o Adafruit IO.

A ideia principal é de que os dados podem ser enviados e recebidos pela definição do feed. Os dados podem ser publicados ou inscritos no feed. O cliente MQTT está conectado ao Adafruit IO como número de porta 1883 e o nome de usuário da conta e a chave de segurança do servidor. Os recursos importantes do MQTT são a capacidade de especificar uma QoS e impõem um limite de taxa para evitar uma carga excessiva ao servidor.

4 RESULTADOS

Esse capitulo irá tratar do problema que levou a criação do projeto desenvolvido, de como o projeto funciona, suas características, detalhando passo a passo do código, bem como descrevendo como ele poderia auxiliar a evitar tais problemas. A Figura 11 ilustra como o projeto é arquitetado.

Figura 11- Funcionamento do projeto



Fonte: Autor

4.1 ANÁLISE DO PROBLEMA

Em um incêndio, ou qualquer tipo de fogueira, deve-se priorizar e prevenir qualquer tipo de surpresa. Embora o fogo seja o maior, ameaçador e perigoso foco, é na fumaça que se deve ter a maior atenção.

Uma fogueira ou incêndio nasce da combustão de materiais orgânicos: um combustível e um comburente. Como consequência há liberação de CO_2 e H_2O. O CO_2 é um gás inodoro e incolor, porém se ingerido pelas pessoas, pode provocar uma série de consequências para o corpo humano.

O CO_2 se ingerido, provoca desorientação e medo, além de lacrimejamento e irritação nos olhos, vômito, tosse, asfixia e alteração nos batimentos cardíacos.

Para evitar ou diminuir esta interferência de CO_2, deve-se tomar algumas precauções em relação ao fogo.

A instalação de um sistema de prevenção, e combate a incêndio, em locais estratégicos e de fácil acesso, como extintores de incêndio em quantidade adequada e de tipos adequados.

O caso Kiss ficou famoso mundialmente, e um dos principais fatores que contribuíram com o desastre foi a falta de comunicação entre os seguranças, e a falta de sinalização para a saída de emergência, um sistema integrado poderia ter ajudado a comunicação entre os seguranças avisando o incêndio que estava acontecendo, permitindo assim a liberação das pessoas do estabelecimento. Grande problema relacionada a morte das pessoas foi devido ao impedimento a saída por conta dos seguranças, trancando as pessoas dentro do local.

Tomando como exemplo o caso, não houve comunicação entre os seguranças que estavam no palco e os seguranças que estavam na saída da boate, os mesmos não permitiram no inicio do incêndio que as pessoas saíssem pela porta (entrada e saída) do local, por acreditar que no local estava acontecendo uma briga. Foi necessário que muitos jovens forçassem a saída, procuraram qualquer meio que os ajudassem a sair do local, mas acabaram se confundindo com as portas dos banheiros ao invés de serem portas de emergência, sem acesso para a rua, dessa forma dados apresentados evidenciaram que 95% dos corpos dos jovens vítimas do incêndio estavam no banheiro.

4.2 DESENVOLVIMENTO DO PROTÓTIPO

Após o estudo da literatura e a pesquisa das alternativas possíveis para atender a demanda, implementou-se um protótipo para verificação da funcionalidade da solução.

O protótipo conta com os seguintes componentes:

- 2 módulos ESP 32
- 2 Gravador in-circuit Silicon Labs CP210x USB to UART
- 1 Sensor MQ2
- 1 Sensor flame
- 1 Potenciômetro 0-1MΩ
- Resistores de 220Ω
- 1 Buzzer
- 1 Módulo Relay 5V
- Software de controle

O microcontrolador ESP32 contém um circuito integrado nele, módulos de WiFi e Bluetooth, o módulo Bluetooth não será utilizado no projeto inicial, por isso não será feita referências a ele no trabalho.

A escolha desse microcontrolador foi feita por base que ele atende todas as necessidades do projeto, como a conexão Wi-Fi e várias portas de entrada e saída com conexão necessária para realizar todas as funções, além de já existir familiaridade com o mesmo, facilitando assim a construção do projeto.

Ao ligar o protótipo, e programar a rede *Wifi*, o microcontrolador será colocado em modo *DeepSleep*, para economizar energia, como já detalhado nesse trabalho, é nesse modo que a maior parte de seus componentes fica desligados, consumindo valores irrisórios de corrente como 20 μA. Apesar o microcontrolador estar “dormindo”, os sensores continuam funcionando.

E é nesse estado que o microcontrolado irá permanecer até que algum sensor acorde o mesmo, a saída digital do MQ2 servirá para acordar o ESP32 nesse exemplo. O Módulo MQ2 contém 4 pinos responsáveis pelo funcionamento e comunicação com o microcontrolador, O pino A0 está conectado a GPIO35, uma porta com um ADC integrado, que através da programação, é feita a leitura analógica do sensor para o microcontrolador.

E o pino D0 do módulo sensor MQ2 está ligada a GPIO32, o pino digital responde a fórmula (1), determinando o valor da resistência que fará o pino mudar de status. A GPIO32 por sua vez, faz parte do bloco RTC do ESP32, permitindo ser configurado o *DeepSleep* sensível à mudança de status do pino.

Outro fator muito positivo do ESP32 é a velocidade de BOOT que ele possui, em menos de 2 segundos, ele já está operando e pronto para executar o programa instalado dentro dele, assim que ele conectou a uma conexão estável de internet configurada anteriormente, ele tenta se conectar ao servidor MQTT, e ao obter sucesso, ele verifica os sinais dos sensores, compara eles entre uma lista de valores calibrados, e se algum desse valor, estiver fora do padrão aceitável, iniciará a enviar os dados coletados por todos os sensores ao broker do MQTT.

Utilizando 2 canais ADCs de 12 bits, serão feitas leituras recebidas do sensor flame e do sensor MQ2, os quais variam suas resistências, a partir da presença de luz ultravioleta e gases tóxicos ao ser humano respectivamente.

Alimentando o sensor flame e o sensor MQ2 com a saída 5V da placa do módulo ESP 32, o sensor flame, passa por um divisor de tensão, controlado pelo potenciômetro, afim de regular a sensibilidade do sensor e para dar uma saída máxima de 3.3V, suportada pelo ESP-32, da mesma forma que o MQ2, que pode chegar a níveis de saída de 5V ficando limitado para não danificar o microcontrolador.

Além disso, usando a comunicação I2C, será conectado um sensor de temperatura e umidade, que ajudará a medir e controlar ainda melhor a situação do ambiente analisado, variações bruscas na temperatura em um pequeno intervalo de tempo pode dar mais certeza ao sistema que está acontecendo um incêndio, caso algum dos outros sensores falhem em detectar por alguma razão, com análises do clima da região, é possível determinar também valores padrões máximos e mínimos, que se passados, poderá sinalizar a possível existência de um incêndio.

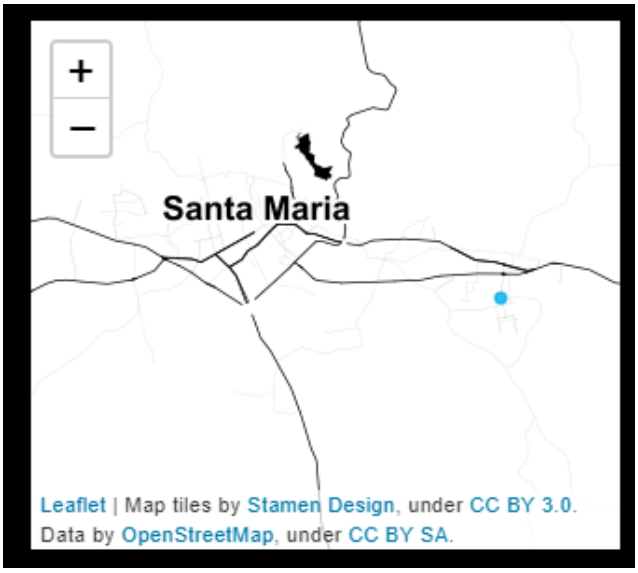
O módulo de temperatura e umidade SHT30 é alimentado por uma fonte de 3.3V e não tem problema com a saída, feita digitalmente pelo barramento I2C.

Fazendo verificações periódicas de 3 segundos, o microcontrolador ESP32 realiza mais de 1000 leituras do sensor, afim de evitar falsos positivos vindos do sensor, nesse intervalo de tempo, ele realiza a média das 1000 leituras e com isso realiza a calibragem de valores, no próprio software, que quando passa do nível aceitável, envia as informações para um servidor MQTT, nesse projeto utilizando o Adafruit IO.

Além disso, o software de controle e o servidor MQTT suportam o envio de coordenadas GPS, enviando valores para latitude, longitude e elevação, o próprio servidor MQTT fornece uma interface para fácil localização.

O fato de poder enviar tais coordenadas é muito importante para facilitar a comunicação com os órgãos competentes, recebendo os dados de um módulo GPS, que será simulado nesse protótipo, é possível não só enviar a posição plana, mas também a elevação em relação ao nível do mar, podendo assim identificar até mesmo o andar que poderia estar acontecendo um possível incêndio. Como mostrado na Figura 12.

Figura 12 - Mapa da localização recebida



Fonte: Interface Adafruit IO

A escolha pelo servidor Adafruit IO foi feita para facilitar a integração com outros dispositivos, porém é possível fazer alterações no software de controle, e criar um próprio servidor MQTT utilizando um *Raspberry PI*, que cumpriria de forma econômica e eficiente o papel de Broker para armazenar as informações publicadas pelo microcontrolador ESP32.

Mais um módulo ESP32 também é conectado ao servidor, com um simples objetivo de demonstrar o funcionamento do servidor MQTT e exibir outros recursos do projeto, após as mesmas configurações de Wi-Fi e conexão serem programadas no módulo, ele irá se inscrever nos mesmo tópicos onde serão enviadas as informações referentes a temperatura, ao sensor flame e ao sensor MQ2.

Ao receber algum valor potencialmente perigoso, ele irá tomar algumas medidas de precaução, ele servirá para demonstrar como o dispositivo pode ser usado como uma medida de segurança, conectado um *relay*, um *buzzer* e alguns *LEDs*, servirão para simular possíveis ações.

O microcontrolador controlará *LEDs*, e um *buzzer*, a fim de alertar os moradores do local que está acontecendo um incêndio, os *LEDs* irão representar as luzes de emergência, para facilitar que as pessoas encontrem a saída o mais rápido possível em um momento de pânico, e o *buzzer* irá simular um sistema de alarme, para dar o aviso sonoro e chamar a atenção das pessoas para o problema que está acontecendo, e se dirigirem para as saídas mais próximas.

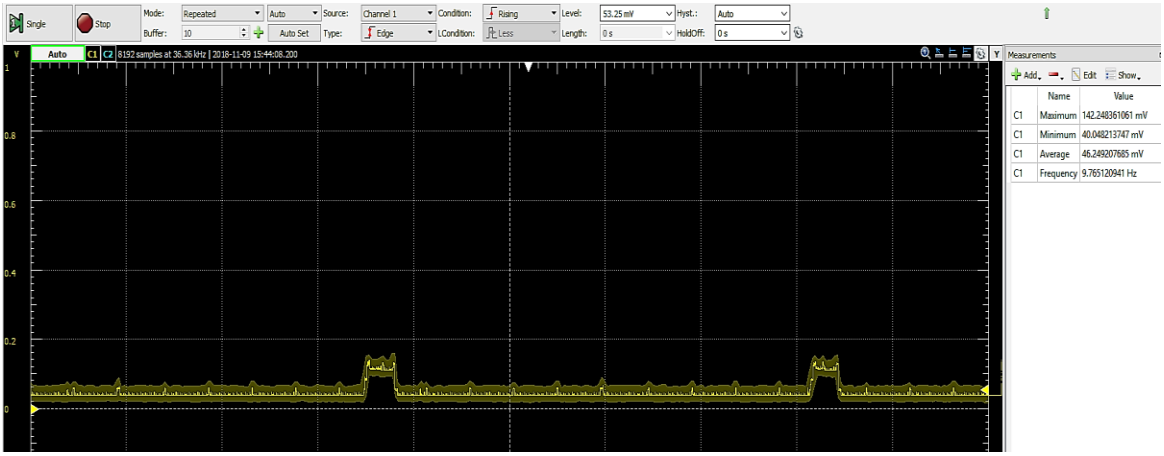
O relay sendo acionado pelo microcontrolador, pode operar para desligar as linhas de transmissão principais da caixa de distribuição do prédio, deixando apenas a linha de emergência operando, para evitar que o fogo se potencialize com falhas elétricas que podem vir a acontecer, ou também, pode ser usado para acionar exaustores e evacuar a fumaça e oxigênio do local, diminuindo possíveis problemas a saúde que a fumaça pode causar a população do local naquele instante.

5 CONSUMO E CUSTO

Os primeiros testes realizados foram os relacionados ao consumo de energia. Foram efetuadas 2 medições diferentes, uma em modo de publicação em MQTT e uma durante o modo *Deepsleep*. Para medir a energia consumida e os tempos, foi utilizado um osciloscópio e analisador lógico USB da DIGILENT chamado de Analog Discovery.

A Figura 13 e a Figura 14 a seguir mostra a medição da corrente de entrada do módulo durante o anúncio. O fato de aparecer tensão e não corrente nas imagens é devido ao fato do osciloscópio estar medindo a tensão sobre o resistor de 1Ω na entrada do módulo. Portanto o valor em tensão pode ser convertido para o mesmo valor em corrente.

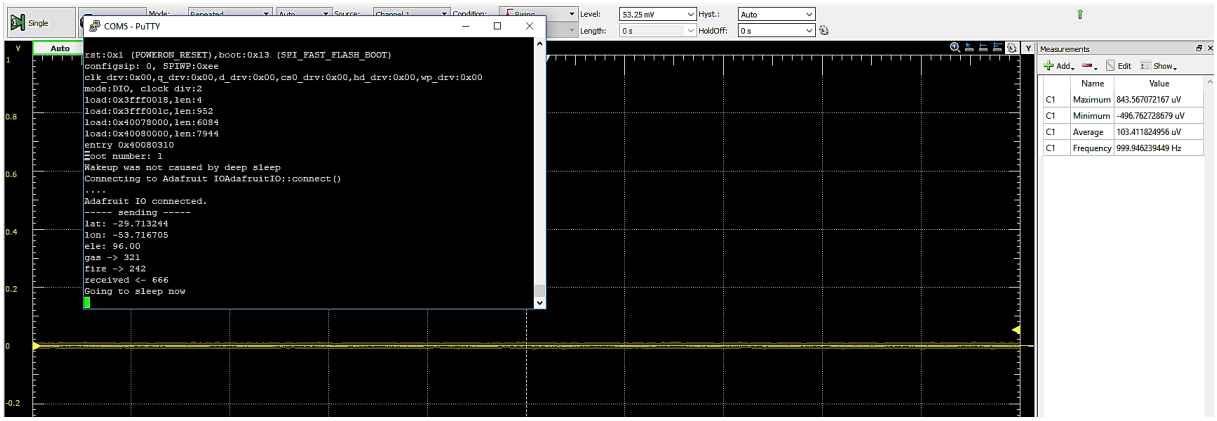
Figura 13- Consumo transmitindo dados



Fonte: Autor

Os valores de pico chegam a 146mV quando está efetuando a transmissão de dados para o servidor MQTT, e o valor mínimo fica aproximadamente em 45mV, que é o consumo médio que o módulo consome para manter a conexão Wi-Fi.

Figura 14 - Consudo durante DeepSleep



Fonte: Autor

O valor médio de consumo durante o modo *DeepSleep* não ultrapassa o valor de 100uV.

Existem aparelhos eletrônicos semelhantes no mercado internacional, como é o caso do produto chamado *Nest Protect*, eleito a melhor compra de 2018 no segmento de dispositivos de detecção de fumaça inteligentes. Ele consta com integração direta ao celular, via uma conta de login e senha, onde é possível monitorar o *status* do ambiente 24 horas por dia, porém não consta com nenhum sistema para avisar órgãos competentes.

Ele pode ser ligado com uma bateria ou diretamente na luz, e pode ser encontrado modelos a venda na Amazon, por preços variando de $110 até $160, na cotação atual do dólar, sem impostos, o valor fica na faixa de preço de R$400,00 a R$600,00 respectivamente, em comparação a esse produto já existente no mercado, a Tabela 7 mostra os valores gastos para a produção desse projeto.

Tabela 7- Orçamento

ITEM	UNIDADE	VALOR UNITÁRIO
ESP32 DEVKIT V1	2	R$ 45,00
RELAY	1	R$ 1,81
SENSOR MQ2	1	R$ 3,81
SENSOR SHT30	1	R$ 5.45
Frete		R$10.00

Fonte Mercado Livre

Com um valor bem inferior ao produto vendido na Amazon, é possível ter um aparelho com mais atributos, além das funcionalidades similares entre os dois produtos. Pois além do projeto conter integração com o celular, assim como o modelo vendido, a proposta adiciona funcionalidades para avisar órgãos competentes e comunica-los de possíveis anomalias no ambiente, enviando junto a posição do aparelho.

6 CONCLUSÃO

Esse projeto é uma solução simples para um problema complicado, sistemas de alarme de incêndio atuais ainda são feitos de formo analógica, após a detecção de fumaça, os sensores emitem um alarme, a espera que as pessoas reajam, evacuando do local e ligando para as autoridades competentes.

Utilizando componentes baratos, e com uma simples implementação, é possível salvar vidas, agindo imediatamente após o incêndio começar, o *LED* ligado e desligado por um relé pode representar acionamentos diversos para auxiliar na evacuação.

Ligar exaustores, luzes de emergência, sistemas de alarme, tudo acionado por um microcontrolador, operando em 3.3V com um consumo de energia muito baixo.

A internet das coisas tem recebido bastante atenção tanto da academia quanto da indústria, devido ao seu potencial de uso nas mais diversas áreas das atividades humanas.

Um problema que precisa ser aprimorado nesse projeto, é o sensor MQ2, ainda que efetue o trabalho com uma certa precisão, existem outros modelos no mercado mais eficazes, com um consumo de potência menor, e uma forma mais eficaz e confiável de detecção de gás, como é o caso do sensor SGP30, que pela indisponibilidade no mercado nacional, não foi possível comprar a tempo para a realização do projeto. Mesmo assim a incorporação do mesmo não modificaria muito o projeto proposto, visto que é conectado pela interface I2C, e já existem bibliotecas para a sua utilização no mercado.

A Internet das Coisas, e com um sistema integrado, é possível ter o controle de muitas coisas, como a iluminação, temperatura, ativação de eletrodomésticos, como programar a torradeira para o horário que você costuma acordar ou preparar um cafezinho para quando você estiver chegando em casa. São inúmeras possibilidades de integração com sistemas e aparelhos disponíveis no mercado e muitas outras estão em desenvolvimento. Tudo isso está relacionado com o movimento da Internet das Coisas (IoT), que tem como objetivo conectar os itens do dia a dia com a internet, para coletar informações em tempo real e auxiliar na vida das pessoas.

BIBLIOGRAFIAS

ARDUINO. Arduino Products. Disponível em: <https://www.arduino.cc/en/Main/Products>. Acesso em 30 nov. 2018.

ARDUINO. Building na Arduino on a Breadboard. Disponível em: <https://www.arduino.cc/en/Main/Standalone>. Acesso em 30 novo. 2018.

ARDUINO. LanguageReference. Disponível em: <https://www.arduino.cc/reference/en/>. Acesso em 30 nov. 2018.

ASSIS, P. D. K. B. Microcontrolador. p.92. Dissertação (Bacharel em Ciências da Computação), Universidade Presidente Antônio Carlos, Barbacena, MG,2004.

AZZOLA, Francesco. MQTT Protocol Tutorial: Step by step guide. 2015. Disponível em: <https://www.survivingwithandroid.com/2016/10/mqtt-protocol-tutorial.html>. Acesso em: 30 novembro. 2018.

BRITO. Rafael Leonardo de Lima. **Potencial da Internet das Coisas na Saúde, Educação e Segurança Pública no Brasil.** Recife. 2017. Disponível em: http://www.cin.ufpe.br/~tg/2017-2/rllb-tg.pdf. Acesso em 14 dez 2018.

MORIMOTO. Carlos. **Manual de hardaware completo.** 3ed. Disponível em: http://www.guiadohardware.net. Acesso em 15 dez 2018.

MINERVA. Roberto, BIRU Abyi, ROTONDI Domenico. Towards a definition of the Internet of Things (IoT) **Telecom Italia S.p.A.** Issue 1 – Published 13 MAY 2015 https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Issue1_14MAY15.pdf

OLIVEIRA, A.; ANDRADE, F. Sistemas Embarcados - Hardware e Firmware na Prática, 2ed. Érica, 06/2010.

OLIVEIRA, C. V.; ZANETTI, H P. Arduino Descomplicado – Como Elaborar Projetos de Eletrônica. Érica, 06/2015.

PIERRE-LUC. Node and MQTT, do something on message. 2015. Disponível em: <https://stackoverflow.com/questions/32538535/node-and-mqtt-do-something-on-message>. Acesso em: 30 novembro. 2018.

PIRES. Paulo F. DELICATO Flavia C., BATISTA. Thais, BARROS. Thomaz, CAVALCANTE. Everton, PITANGA. Marcelo. **Plataformas para a Internet das Coisas.** Disponível em: http://sbrc2015ufes.br/wp-content/uploads/ch3.pdf . Acesso em 15 dez 2018.

ROZSA, Vitor. DUTRA, Moisés Lima. PINTO, Adilson Luiz. TORRADO Enrique Muriel. O paradigma tecnológico da internet das coisas e sua relação com a ciência da informação. **Inf. & Soc.:Est.,** João Pessoa, v.27, n.3, p. 255-266, set./dez. 2017. Disponível em: Acesso em: 15 dez 2018.

SOUSA, J. P. Porgramação em C e Assembly na Família 51. Disponível em: <https://web.fe.up.pt/~jms/SUC/AulaLab3.pdf>. Acesso em 30 novembro. 2018.

TEXAS INSTRUMENTS. TMS320C642x DSP General-Purpose Input/Output (GPIO). **User's Guide**. Literature Number: SPRUEM8A March 2008. Disponível em: http://wwww.ti.com/lit/ug/sprufl8b/sprufl8b.pdf Acesso em: 17 dez 2018.

ZANELLA, A. et al. Internet of for Smart Cities. IEEE.Internet of Things Journal, v. 1, p. 22-32. 2014. Disponível em: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?aenumber=6740844> Acesso em 16 dez 2018.

ANEXOS

ANEXO 1 CÓDIGO PUBISHLER

```
1   //Inicializa as bibliotecas
2   #include <Arduino.h>
3   #include "AdafruitIO_WiFi.h"
4   #include "WEMOS_SHT3X.h"
5
6   //Login e senha para o servidor
7   #define IO_USERNAME    "Illuminight"
8   #define IO_KEY         "716520ae70aa443caed03af6878974c1"
9
10  // Configuração da WIFI
11  #define WIFI_SSID       "Ubiratan"
12  #define WIFI_PASS       "Carandi4.2"
13
14  AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
15
16  #define MQ2_PIN 35 //GPIO35
17  #define FIRE_PIN 34 //GPIO34
18  #define ONLINE 2 //GPIO2
19  #define MQ2_DIG 33 // GPIO33
20  #define CODESLEEP 666 // Código para o modo sleep
21
22  #define MINGAS 700 // Valor definido para envio de dados sensor MQ2
23  #define MINFIRE 300 // Valor definido para envio de dados sensor flame
24  #define MINTEMP 35 // Valor definido para envio de dados sensor SHT30
25
26  #define IO_LOOP_DELAY 6000
```

```
29  int angas = 0;
30  int anfire = 0;
31  double antemp = 0;
32  double anhumi = 0;
33
34  int count = 0;
35  int send =0;
36  unsigned long lastUpdate = 0;
37
38  SHT3X sht30(0x45);
39
40  //GPS
41  double lat = -29.71324429;
42  double lon = -53.71670473;
43  double ele = 96;
44
45  int lastgas = 0; //Váriavel para armazenar último valor do MQ2
46  int lastfire = 0; //Váriavel para armazenar último valor do flame
47  double lasttemp = 0; //Váriavel para armazenar último valor do SHT30
48  double vartemp = 0; //Váriavel para armazenar último valor do SHT30
49
50  // Configura os feeds do MQTT
51  AdafruitIO_Feed *mq2 = io.feed("data");  //Feed do MQTT para o sensor MQ2
52  AdafruitIO_Feed *fire = io.feed("dtfire"); //Feed do MQTT para o sensor flame
53  AdafruitIO_Feed *dsleep = io.feed("sleep"); //Feed do MQTT para o modo sleep
54  AdafruitIO_Feed *tmp = io.feed("temp"); //Feed do MQTT para o sensor SHT30
55  AdafruitIO_Feed *umid = io.feed("humidity"); //Feed do MQTT para o sensor SHT30
56
57  RTC_DATA_ATTR int bootCount = 0;
58
59  //Função para imprimir a razão do modo sleep ser interrompido
60  void print_wakeup_reason(){
61    esp_sleep_wakeup_cause_t wakeup_reason;
62
63    wakeup_reason = esp_sleep_get_wakeup_cause();
64
```

```
  switch(wakeup_reason)
  {
    case 1  : Serial.println("Wakeup caused by external signal using RTC_IO"); break;
    case 2  : Serial.println("Wakeup caused by external signal using RTC_CNTL"); break;
    case 3  : Serial.println("Wakeup caused by timer"); break;
    case 4  : Serial.println("Wakeup caused by touchpad"); break;
    case 5  : Serial.println("Wakeup caused by ULP program"); break;
    default : Serial.println("Wakeup was not caused by deep sleep"); break;
  }
}

void goSleep(){   //Função a ser chamada para iniciar o modo Sleep
  Serial.println("Going to sleep now");
  dsleep->save(0);
  delay(50);
  esp_deep_sleep_start();

}
//Função a ser chamada para verificar o código e entrar no modo sleep
void handleMessage(AdafruitIO_Data *data) {
  int value = data->toInt();
  if(value == CODESLEEP){
    Serial.print("received <- ");
    Serial.println(data->value());
    goSleep();
  }
}

```

```
93  void setup() {
94    // Inicia conexão serial
95    Serial.begin(115200);
96
97    //Definição dos pinos de entrada e saída
98    pinMode(MQ2_PIN, INPUT);
99    pinMode(FIRE_PIN, INPUT);
100   pinMode(MQ2_DIG, INPUT);
101   pinMode(ONLINE, OUTPUT);
102
103   //Seta a resolução do ADC para 11 bits
104   analogReadResolution(11); //11 bits
105   //Seta atenuação do ADC para um ganho de 11db
106   analogSetPinAttenuation(MQ2_PIN, ADC_11db);  //FOR PIN MQ2
107   analogSetPinAttenuation(FIRE_PIN, ADC_11db);  //For PIN FIRE
108
109   // Contador para o número de vezes que o ESP32 saiu do modo sleep
110   ++bootCount;
111   Serial.println("Boot number: " + String(bootCount));
112
113   //Printa a razão do ESP32 ter acordado
114   print_wakeup_reason();
115
116   //Define como o ESP32 irá ser acordado pelo pino 33
117   esp_sleep_enable_ext0_wakeup(GPIO_NUM_33,0); //1 = High, 0 = Low
118
119   // Conecta ao servidor MQTT io.adafruit.com
120   Serial.print("Connecting to Adafruit IO");
121   io.connect();
122
123   // Espera a conexão
124   while(io.status() < AIO_CONNECTED) {
125     Serial.print(".");
126     delay(500);
127   }
128
```

```
129     // Se já conectou, exibe o status e analisa se deve operar
130     // no modo sleep ou não
131     Serial.println();
132     Serial.println(io.statusText());
133     dsleep->get();
134     dsleep->onMessage(handleMessage);
135
136   }
137
138   void loop() {
139
140     io.run();
141
142     //Se está conectado liga o LED da GPIO2
143     if (io.status() == AIO_CONNECTED){
144       digitalWrite(ONLINE, HIGH);
145     }
146     else { //Se está desconectado, desliga o LED
147       digitalWrite(ONLINE, LOW);
148     }
149
150     lastgas = 0;
151     lastfire = 0;
152
153     //Faz a leitura dos sensores definido pela variavel sample
154     for(int i=0;i<sample;i++) {
155       lastgas = lastgas + analogRead(MQ2_PIN);
156       lastfire = lastfire + analogRead(FIRE_PIN);
157     }
158
159     // Faz a média da leitura dos valores, para evitar falsos positivos
160     angas = lastgas/sample;
161     anfire = lastfire/sample;
162
```

```
163     if(sht30.get()==0){ // Salva os valores da temperatura e umidade nas váriaveis
164       antemp = sht30.cTemp;
165       anhumi = sht30.humidity;
166     }else { // Se não conseguiu conexão com o sensor, seta as váriaveis para -1
167       antemp = -1;
168       anhumi = -1;
169     }
170 
171     //Se o tempo total do sensor é maior que a ultima atualização mais o ultimo
172     // delay inicia, a fimd e evitar congestinamento na rede
173     if (millis() > (lastUpdate + IO_LOOP_DELAY)) {
174 
175       //Calcula a váriação de temperatura nos últimos instantes, devidido pelo
176       // delay do loop
177       if (lasttemp > antemp)
178       vartemp = lasttemp - antemp;
179       if (antemp > lasttemp)
180       vartemp = antemp - lasttemp;
181       lasttemp = antemp;
182 
183       //Se algum dos valores mínimos aceitaves estiver muito alto, envia as
184       // informações para o servidor MQTT
185       if(angas >= MINGAS  ||
186         anfire >= MINFIRE ||
187         antemp >= MINTEMP){
188           Serial.println("### Sending Info ###");
189           Serial.printf("Gas: %d\nFire:  %d\nTemp: %4.2f\nHumidity: %4.2f\nVar:%4.2f\n", angas, anfire, antemp, anhumi, vartemp);
190           Serial.println("### Sending Location ###");
191           Serial.printf("lat: %4.8f long: %4.8f ele: %4.2f\n", lat, lon, ele);
192 
193           //Envia os valores para o servidor, junto com a posição do GPS
194           mq2->save(angas, lat, lon, ele);
195           fire->save(anfire, lat, lon, ele);
196           tmp->save(antemp, lat, lon, ele);
197           umid->save(anhumi, lat, lon, ele);
198           send =1;
199         }
```

```
201         //Se todos os valores voltarão ao valor normal, envia os útlimos dados
202         // e para de enviar
203         else if ((angas < MINGAS &&
204                   anfire < MINFIRE &&
205                   antemp < MINTEMP) && send == 1){
206
207           mq2->save(angas, lat, lon, ele);
208           fire->save(anfire, lat, lon, ele);
209           tmp->save(antemp, lat, lon, ele);
210           umid->save(anhumi, lat, lon, ele);
211           send=0;
212         }
213
214         //Se os valores estiverem ok, printa na conexão serial os dados de
215         // cada sensor e posição GPS
216         else {
217           Serial.println("----- SHOWING -----");
218           Serial.print("Latitude : ");
219           Serial.println(lat, 4);
220           Serial.print("Longitude : ");
221           Serial.println(lon, 4);
222           Serial.print("Elevação : ");
223           Serial.println(ele, 2);
224
225           Serial.print("Gas : ");
226           Serial.println(angas);
227
228           Serial.print("Fire : ");
229           Serial.println(anfire);
230
231           Serial.print("Temperature : ");
232           Serial.println(antemp);
233           Serial.print("Relative Humidity : ");
234           Serial.println(anhumi);
235           Serial.print("Var : ");
236           Serial.println(vartemp);
237         }
238         lastUpdate = millis();
239       }  }
```

Anexo 2 Código Subscribe

```
1   //Inicializa as bibliotecas
2   #include <Arduino.h>
3   #include "AdafruitIO_WiFi.h"
4
5   //Login e senha para o servidor
6   #define IO_USERNAME    "Illuminight"
7   #define IO_KEY         "716520ae70aa443caed03af6878974c1"
8
9   // Configuração da WIFI
10  #define WIFI_SSID       "Ubiratan"
11  #define WIFI_PASS       "Carandi4.2"
12
13  AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
14
15  #define BUZZER 4 //GIPO4
16  #define ONLINE 2 //GPIO2
17  #define FIRE_PIN 15 //GPIO15
18  #define RELAY_PIN 23 //GPIO23
19  #define SHT_PIN 5 //GPIO5
20
21  #define MINFIRE 300 // Valor definido para tomar uma ação do sensor flame
22  #define MINGAS 700 // Valor definido para tomar uma ação do sensor MQ2
23  #define MAXTEMP 40 //Valor definido para tomar uma ação do sensor SHT30
24  //#define MAXUMID 20
25
26  //Configuração da Buzzer PWM
27  int channel         = 0;
28  int frequence       = 2E5;
29  int resolution      = 12;
30
31  AdafruitIO_Feed *mq2 = io.feed("data"); //Feed do MQTT para o sensor MQ2
32  AdafruitIO_Feed *fire = io.feed("dtfire"); //Feed do MQTT para o sensor flame
33  AdafruitIO_Feed *tmp = io.feed("temp");  //Feed do MQTT para o sensor SHT30
34  //AdafruitIO_Feed *umid = io.feed("humidity");
35
```

```
36  //Função a ser chamada para verificar o feed do MQ2 e tomar uma ação
37  void handleGas(AdafruitIO_Data *data) {
38    int value = data->toInt();
39   //Se o valor passar do mínimo estiupaldo, libera o RELAY
40    if(value >= MINGAS && value <=1500){
41      Serial.println("GAS HIGH");
42      digitalWrite(RELAY_PIN, HIGH);
43      Serial.print("received gas <- ");
44      Serial.println(data->value());
45    }
46
47    else if (value >= 1501){
48      ledcWriteTone(0,4095);
49
50    }
51    //Se o valor estiver abaixo do minimo estipulado, não faz nada,
52    // ou desliga o LED
53    else{
54      Serial.println("GAS LOW");
55      ledcWriteTone(0,0);
56      digitalWrite(RELAY_PIN, LOW);
57    }
58
59  }
```

```
//Função a ser chamada para verificar o feed do flame e tomar uma ação
void handleFire(AdafruitIO_Data *data) {
  int value = data->toInt();
//Se o valor passar do mínimo estiupaldo, liga o LED
  if(value >= MINFIRE){
    Serial.println("FIRE HIGH");
    digitalWrite(FIRE_PIN, HIGH);

    Serial.print("received fire <- ");
    Serial.println(data->value());
  }
  //Se o valor estiver abaixo do minimo estipulado, não faz nada,
  // ou desliga o LED
  else{
    Serial.println("FIRE LOW");
    digitalWrite(FIRE_PIN, LOW);
  }
}
//Função a ser chamada para verificar o feed do SHT30 e tomar uma ação
void handleTemp(AdafruitIO_Data *data) {
  int value = data->toInt();
//Se o valor passar da máxima temperatura estiupalda, liga o LED
  if(value >= MAXTEMP){
    Serial.println("Temperature HIGH");
    digitalWrite(SHT_PIN, HIGH);

    Serial.print("received Temperature <- ");
    Serial.println(data->value());
  }
  //Se o valor estiver abaixo do minimo estipulado, não faz nada,
  // ou desliga o LED
  else{
    Serial.println("Temperature LOW");
    digitalWrite(SHT_PIN, LOW);
  }
}
```

```
112  void setup() {
113    // inicia conexão serial
114    Serial.begin(115200);
115
116    //Configura buzzer PWM
117    ledcSetup(channel, frequence, resolution);
118    ledcAttachPin(BUZZER, channel);
119
120    //Define pinos de enrtada e saida
121    pinMode(ONLINE, OUTPUT);
122    pinMode(FIRE_PIN, OUTPUT);
123    pinMode(RELAY_PIN, OUTPUT);
124    pinMode(SHT_PIN, OUTPUT);
125
126    //Seta os pinos para nível baixo
127    digitalWrite(BUZZER, LOW);
128    digitalWrite(FIRE_PIN, LOW);
129    digitalWrite(RELAY_PIN, LOW);
130    digitalWrite(SHT_PIN, LOW);
131
132    // conecta ao servidor MQTT io.adafruit.com
133    Serial.print("Connecting to Adafruit IO");
134    io.connect();
135
136    // Espera a conexão
137    while(io.status() < AIO_CONNECTED) {
138      Serial.print(".");
139      delay(500);
140    }
141    // we are connected
142    Serial.println();
143    Serial.println(io.statusText());
144
```

```
145     //Se está conectado liga o LED da GPIO2
146     if (io.status() == AIO_CONNECTED){
147       digitalWrite(ONLINE, HIGH);
148     }
149     else { //Se está desconectado, desliga o LED
150       digitalWrite(ONLINE, LOW);
151     }
152
153     // Chama as funções para cada feed, para saber o que será feito
154     // com os dados recebidos.
155     mq2->onMessage(handleGas);
156     fire->onMessage(handleFire);
157     tmp->onMessage(handleTemp);
158     //umid->onMessage(handleUmid);
159
160     // Recebe os valores assim que conecta.
161     mq2->get();
162     fire->get();
163     tmp->get();
164     //umid->get();
165   }
166
167   void loop() {
168     // process messages and keep connection alive
169     io.run();
170   }
```