UNIVERSIDADE ESTADUAL DO RIO GRANDE DO SUL
UNIDADE UNIVERSITÁRIA EM PORTO ALEGRE
CURSO SUPERIOR DE TECNOLOGIA EM AUTOMAÇÃO INDUSTRIAL

MÁRCIO BARBIANI

INTERNET DAS COISAS: CONECTANDO SISTEMAS DE AR CONDICIONADO

PORTO ALEGRE
2018

MÁRCIO BARBIANI

INTERNET DAS COISAS: CONECTANDO SISTEMAS DE AR CONDICIONADO

Monografia apresentada como requisito parcial para obtenção do grau de Tecnólogo em Automação Industrial, na Universidade Estadual do Rio Grande do Sul.

Orientador: Prof. Dr. André Borin Soares.

PORTO ALEGRE
2018

MÁRCIO BARBIANI

INTERNET DAS COISAS: CONECTANDO SISTEMAS DE AR CONDICIONADO

Monografia apresentada como requisito parcial para obtenção do grau de Tecnólogo em Automação Industrial, na Universidade Estadual do Rio Grande do Sul.

Orientador: Prof. André Borin Soares.

Aprovado em: / /

BANCA EXAMINADORA

_____

Orientador: Prof. Dr. André Borin Soares
Universidade Estadual do Rio Grande do Sul

_____

Prof. Me. Emerson Fernandes da Cunha
Universidade Estadual do Rio Grande do Sul

_____

Prof. Dr. Luiz Fernando Gonçalves
Universidade Estadual do Rio Grande do Sul

PORTO ALEGRE
2018

Dedico esta monografia à minha família que me apoiou em todos os momentos, às pessoas que de alguma forma contribuíram para sua realização, em especial aos professores orientadores e à Universidade Estadual do Rio Grande do Sul.

RESUMO

Nos séculos passados, as condições de conforto em locais fechados eram controladas por processos adaptativos relacionados à vestimenta, exaustão e do uso de fornos e lareiras para efetuar o controle da temperatura. Os sistemas de resfriamento e aquecimento como os inventados por Willis Carrier em 1902 possibilitaram que o indivíduo ajustasse as características de ambientes interno e, consequentemente, criaram uma demanda para a melhoria das condições de conforto nas construções. Essa demanda pela climatização de ambientes tem, nos últimos anos, aumentado drasticamente o consumo energético. Para ajudar na otimização desse consumo energético, o trabalho propõe e implementa um conjunto de soluções baseadas no protocolo MQTT, na linguagem Javascript e na ferramenta Node-Red, todos provenientes do conceito Internet das Coisas, para o controle central de sistemas de ar condicionado distribuídos. Como resultado deste trabalho, obteve-se um sistema expansível para o aumento da eficiência energética de uma edificação através do controle minimamente invasivo de condicionadores de ar.

Palavras-chave: ar-condicionado, energia elétrica, conforto térmico, MQTT, Node-red, Javascript, Internet das Coisas.

ABSTRACT

In past centuries, comfort conditions in enclosed spaces were controlled by adaptive processes related to dress, exhaustion, and the use of stools and fireplaces to effect temperature control. Cooling and heating systems such as those invented by Willis Carrier in 1902 enabled the individual to adjust the characteristics of indoor environments and consequently created a demand for improved comfort conditions in buildings. This demand for ambient air conditioning has, in recent years, drastically increased energy consumption. To help optimize this energy consumption, the work proposes and implements a set of solutions based on the MQTT protocol, the Javascript language and the Node-Red tool, all of which come from the Internet concept of Things, for the central control of air systems conditioning. As a result of this work, an expandable system was obtained to increase the energy efficiency of a building through the minimally invasive control of air conditioners.

Keywords: air conditioning, electric power, thermal comfort, MQTT, Node-red, Javascript, Internet of Things.

LISTA DE FIGURAS

Figura 1 - Disposição dos condicionadores de ar 16
Figura 2 - Máquina de refrigeração 18
Figura 3 - Sinal infravermelho demodulado. 19
Figura 4 - Raspberry PI 3B 25
Figura 5 - Microcontrolador Xtensa ESP32 26
Figura 6 - Ar condicionado split LG 27
Figura 7 - Controle remoto infravermelho 28
Figura 8 - Espruino WEBIDE 29
Figura 9 - NODE-RED 30
Figura 10 - Win32 Disk Imager 33
Figura 11 - Um *workflow* do node-red 33
Figura 12 - Placa de desenvolvimento ESP32-EVB. 35
Figura 13 - ESP32 DOWNLOAD TOOL 35
Figura 14 - Aplicação em funcionamento 36
Figura 15 - Sinal do receptor infravermelho 37
Figura 16 - *Workflow* e saída do Node-red 38
Figura 17 - Múltiplas salas controladas 39

LISTA DE TABELAS

Tabela 1 - Códigos dos controles infravermelhos.....37

LISTA DE SIGLAS E ABREVIATURAS

A	Ampère.
ASHRAE	*American Society of Heating, Refrigerating and Air-Conditioning Engineers*.
º C	Grau Celsius.
IDE	*Integrated Development Environment.*
HTML	*HyperText Markup Language*.
I2C	*Inter-Integrated Circuit*
IHM	Interface Homem Máquina
IoT	*Internet of Things*.
TCP/IP	*Transmission Control Protocol/Internet Protocol.*
IrDA	*Infrared Data Association*.
MQTT	*Message Queue Telemetry Transport*
TFT	*Thin Film Transistor*.
PWM	*Pulse Width Modulation*
SRAM	*Static Random Acess Memory*
ROM	*Read Only Memory*
SPI	*Serial Peripheral Interface*
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 PROBLEMÁTICA 14
1.2 HIPÓTESE 15
1.3 OBJETIVOS 16
2 REFERENCIAL TEÓRICO 17
2.1 SISTEMAS DE REFRIGERAÇÃO 17
2.2 RADIAÇÃO INFRAVERMELHA 19
2.3 INTERNET DAS COISAS 20
2.4 PROTOCOLO MQTT 20
2.5 LINGUAGEM JAVASCRIPT EM MICROCONTROLADORES 21
2.6 PROJETOS RELACIONADOS 22
3 MATERIAIS E MÉTODOS 24
3.1 UNIVERSO DE ABRANGÊNCIA DO TRABALHO 24
3.2 PLACA DE DESENVOLVIMENTO RASPBERRY PI 3B 24
3.3 XTENSA ESP32 25
3.4 AR CONDICIONADO SPLIT LG 26
3.5 PROGRAMAS UTILIZADOS 28
3.5.1 Espruino Web IDE 28
3.5.2 Node.Js 29
3.5.3 Node-Red 30
3.5.4 Mosca 31
3.6 METODOLOGIA 31
4 DESENVOLVIMENTO 32
4.1 COMPUTADOR DE PEQUENO PORTE RASPBERRY PI 3B 32
4.2 PLACA DE DESENVOLVIMENTO ESP32-EVB 34
4.2.1 Engenharia reversa dos códigos de controle infravermelho 36
5 RESULTADOS 38
6 CONCLUSÃO 40
REFERÊNCIAS 41
ANEXO A – DIAGRAMA ELÉTRICO OLIMEX ESP32-EVB 44
ANEXO B – CÓDIGO DA APLICAÇÃO MOSCA-APP.JS 45

ANEXO C – COMANDOS PARA INSTALAR FERRAMENTAS 46
APÊNDICE A – DIAGRAMA DE LIGAÇÃO ELETRÔNICA 47
APÊNDICE B – LIGAÇÃO DO LED EXTERNO 48
APÊNDICE C – CÓDIGOS DE PROGRAMA DA ESP32-EVB 49
APÊNDICE D – WORKFLOW NODE-RED 51
APÊNDICE E – SAÍDA DO ESPRUINO 52

1 INTRODUÇÃO

Nos séculos passados, as condições de conforto em locais fechados eram controladas por processos adaptativos relacionados à vestimenta, exaustão e do uso de fornos e lareiras para efetuar o controle da temperatura. Não sendo possível agir sobre o conforto nas salas, não era útil estudar os parâmetros que influenciariam no conforto. Além de estudar o conforto, era necessário modelar o ambiente como um sistema aberto e aplicar as leis da termodinâmica, uma disciplina nascida na segunda metade do século XIX. Já no século XX, as teorias da arquitetura (Modernismo, Funcionalismo, Bauhaus, Le Corbusier com Le Modulor, De Stijl, CIAM, Estilo internacional, etc.) e manuais técnicos, colocaram o homem no centro, como um indivíduo de dimensão física, e fundaram um interesse no design e construção de edifícios residenciais.

Graças aos sistemas de resfriamento e aquecimento como os inventados por Willis Carrier em 1902 (DEPARTMENT OF ENERGY, 2018), tornou-se possível que o indivíduo ajustasse as características de ambientes internos e, consequentemente, estes sistemas criaram uma demanda para a melhoria das condições de conforto nas construções.

Como definido pelo Centro Canadense de Saúde do Trabalho, o conforto térmico é definido como uma condição mental que expressa satisfação com o ambiente térmico circunjacente. Ter conforto térmico significa que uma pessoa usando uma quantidade normal de roupas não sente nem frio nem calor demais. O conforto térmico é muito importante em relação a fatores relacionados ao trabalho. Ele pode afetar os níveis de distração dos trabalhadores, afetando o seu desempenho e produtividade.

Fatores que afetam esse conforto térmico como temperatura, velocidade e umidade do ar, temperatura média radiante, atividade metabólica e resistência térmica do vestuário foram explorados experimentalmente nos anos setenta. Muitos desses estudos levaram à criação e refinamento do padrão *ASHRAE* (*American Society of Heating, Refrigerating and Air-Conditioning*) 55 (Condições ambientais aceitáveis para ocupação humana), e foram executados na Universidade estadual do Kansas por Ole Fanger e outros. Eles observaram que o conforto percebido era uma interação de todos aqueles fatores. Ainda, descobriram que a maioria dos indivíduos ficariam

satisfeitos com temperaturas entre 23°C e 26°C e umidade relativa do ar de 50%.

Desviando progressivamente desse ideal, cada vez menos as pessoas se sentiam confortáveis. Essa observação pôde ser expressa estatisticamente como condições de conforto e voto médio previsto (*predicted mean vote PMV).* Esse método foi confrontado pelo modelo de controle adaptativo desenvolvido pelo projeto *ASHRAE* 884, o qual revelou que os indivíduos se sentiam confortáveis numa faixa ainda maior de temperaturas.

Nos Estados Unidos, em 1970, uma crise energética ocorreu com a grande demanda e uso dos sistemas de ar condicionado. Em resposta, os legisladores aprovaram leis para reduzir o consumo de eletricidade, criando um programa que estabelecia um único padrão de eficiência energética para os fabricantes desses sistemas. Esse programa já conseguiu melhorias nas novas tecnologias e economizou energia e dinheiro para os consumidores.

No Brasil, somente agora essa demanda aumentou exponencialmente e uma crise energética se aproxima. De acordo com a Empresa de Pesquisa Energética, o consumo elétrico dos sistemas de ar condicionado dos lares brasileiros saltará de 17.126 GWh em 2014 para 36.216 GWh em 2024. Para prevenir isso, o Ministério de Minas e Energia (MME) anunciou uma proposta para exigir uma maior eficiência energética dos aparelhos de ar-condicionado vendidos no País. Os fabricantes estão de acordo com as novas normas, que, se aprovadas, eliminarão do mercado cerca de 40% dos modelos atuais.

Tais medidas servem para economizar energia e evitar um apagão energético. Porém, alguma inteligência pode ser adicionada, otimizando o uso desses equipamentos e fazendo com que eles operem estritamente em horários pré-determinados.

1.1 PROBLEMÁTICA

Com a crescente necessidade de redução e otimização do consumo energético mundial, faz-se necessário uma implantação de controles inteligentes nos equipamentos mais antigos, evitando que consumam energia quando não são necessários.

O tipo de controle oferecido para o usuário pelos modelos fabricados atualmente é geralmente um controle remoto com emissor infravermelho que deve estar sempre na proximidade do sistema de ar condicionado. Em caso de perda desse controle, uma atuação em seu painel, caso ele exista, ou o corte de energia elétrica é necessário para efetuar o seu desligamento, dificultando sua operação e contribuindo muitas vezes para o desperdício de energia.

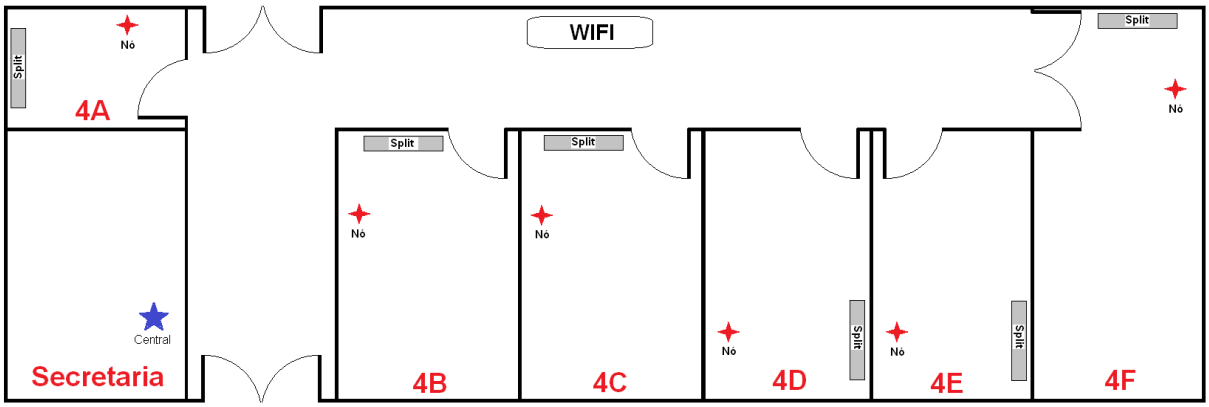
Um novo conceito, Internet das Coisas (IoT - *Internet of Things*), criado em setembro de 1999 por Kevin Ashton (MANCINI, 2017), um pioneiro tecnológico britânico que concebeu um sistema de sensores omnipresentes conectando o mundo físico à Internet, possibilita a modernização daquele simples controle remoto com emissor infravermelho trocando-o por um sistema de computação mais atual e conectando-os. Um sistema desse tipo facilitaria o estudo das condições térmicas do ambiente e também na criação de métodos e algoritmos que ajudassem na redução de consumo energético em cada ambiente controlado.

Neste contexto, o presente trabalho abordará a implementação de uma solução para o controle central para sistemas de ar condicionados distribuídos em diversas salas de um edifício envolvendo, para isso, tecnologias que implementam a internet das coisas.

1.2 HIPÓTESE

Como hipótese deste trabalho se assume que é possível desenvolver um sistema capaz de monitorar a variável temperatura do ambiente e fazer o controle remoto de unidades de ar condicionado distribuídos em salas de um edifício através de um sistema central, um sistema remoto e tecnologias provenientes da internet das coisas. A figura 1 mostra uma possível disposição dos condicionadores de ar no prédio da UERGS.

Figura 1 - Disposição dos condicionadores de ar



Fonte: Autor (2010)

Assume-se que o uso de uma linguagem de alto nível como Javascript para a programação tanto do sistema central quanto o remoto, ao adicionar uma camada a mais de abstração entre o desenvolvedor do sistema e o *hardware*, permite ao desenvolvedor focar somente na solução e não nos detalhes técnicos de cada sistema computacional utilizado. Isto aumentará a produtividade em comparação com implementações básicas de sistemas equivalentes ao proposto, as quais são escritas com a linguagem C e bibliotecas de código aberto como as oferecidas pelo outro projeto de código aberto chamado Arduino.

O trabalho mostra que um ar condicionado split modelo USNW182CSZ2 pode ser controlado remotamente via rede sem fio e que o sistema proposto é capaz de ser ampliado para controlar diversas unidades no raio de alcance da rede.

1.3 OBJETIVOS

O trabalho tem como objetivo o projeto e configuração de protótipos das unidades denominadas central e remota. Dentro do trabalho podem-se caracterizar como objetivo específico os seguintes passos: montar uma unidade central e uma remota; programá-las; executar efetivamente o controle remoto em pelo menos uma sala da edificação; mostrar que o sistema pode ser ampliado significativamente.

Ao concluir estes passos o objetivo deste trabalho de conclusão de curso será atingido e, por consequência, será comprovada a veracidade da hipótese.

2 REFERENCIAL TEÓRICO

Neste tópico são apresentados os referenciais teóricos necessários para a compreensão dos elementos deste trabalho.

Duas partes distintas podem ser observadas. O *hardware*, constituindo todos os componentes eletrônicos utilizados e o *software* desenvolvido com a função específica de atingir os resultados esperados do trabalho.

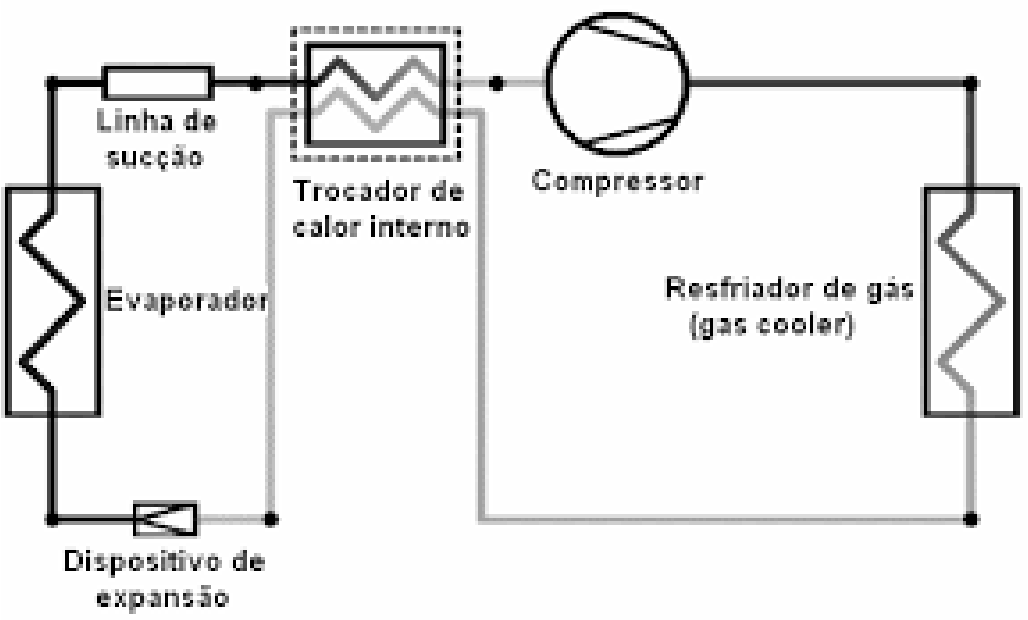
2.1 SISTEMAS DE REFRIGERAÇÃO

Para melhor entender esse componente do trabalho que é o sistema de ar condicionado split, deve-se rever antes os conceitos que estão por trás desse equipamento.

Segundo (SERWAY, 2008), calor é uma energia térmica em trânsito de um corpo para outro em resultado da diferença de temperatura entre corpos. Refrigeração é a ação de resfriar determinado ambiente de forma controlada para viabilizar processos industriais ou efetuar climatização para conforto térmico. Esse resfriamento é causado pela movimentação de energia térmica de determinado corpo ou meio através de um ciclo termodinâmico. Um ciclo termodinâmico se constitui de qualquer série de processos termodinâmicos tais que, ao transcurso de todos eles, o sistema regresse a seu estado inicial; ou seja, que a variação das grandezas termodinâmicas próprias do sistema seja nula.

Ainda, um processo termodinâmico é um evento caracterizado pela mudança de uma ou mais variáveis de estado do sistema. As variáveis de estado são pressão, temperatura e volume, e são chamadas dessa forma pois influenciam fortemente nas propriedades e no comportamento de um gás. Num ciclo de refrigeração por compressão de vapor existem basicamente 5 componentes: compressor, condensador, dispositivo de expansão, evaporador e fluido refrigerante. Esses componentes estão conectados conforme o circuito da figura 2.

Figura 2 - Máquina de refrigeração



Fonte: ABCM (2010)

As etapas do ciclo de refrigeração que ocorrem nesse circuito, segundo (ALTHOUSE et al, 2003) são: evaporação, compressão, condensação e expansão.

A evaporação é a etapa onde o fluido refrigerante entra no evaporador como uma mistura predominantemente líquida, que absorverá calor do ar ao redor das suas aletas. Ao receber calor, o fluido refrigerante saturado vaporiza-se, absorvendo calor do ambiente.

Compressão é a etapa que eleva elevando a pressão do fluido, sempre no estado de vapor. Em um ciclo ideal, as perdas de carga são desprezadas. Na prática perde-se calor ao ambiente nessa etapa, porém não é significativo em relação à potência de compressão necessária.

A condensação é a etapa onde ocorre a rejeição de calor do ciclo. No resfriador de gás, o fluido na forma de gás saturado é condensado ao longo do trocador de calor, que em contato com o ar cede calor ao meio ambiente.

A expansão é a etapa onde ocorre uma perda de pressão brusca, porém controlada que vai reduzir a pressão do fluido, da pressão de condensação para a pressão de evaporação, absorvendo calor. Em um ciclo ideal despreza-se as variações de energia cinética e potencial.

2.2 RADIAÇÃO INFRAVERMELHA

A radiação infravermelha (IV) é uma radiação não ionizante na porção invisível do espectro eletromagnético que está adjacente aos comprimentos de ondas longos, ou final vermelho do espectro da luz visível. Ainda que em vertebrados não seja percebida na forma de luz, a radiação IV pode ser percebida como calor, por terminações nervosas especializadas da pele, conhecidas como termo receptores.

Foi descoberta em 1800 por William Herschel e é muito utilizada nas trocas de informações entre computadores, celulares e outros eletrônicos, através do uso de um adaptador *IrDA* (*Infared Data Association*, Definição de padrões de comunicação entre equipamentos de comunicação wireless).

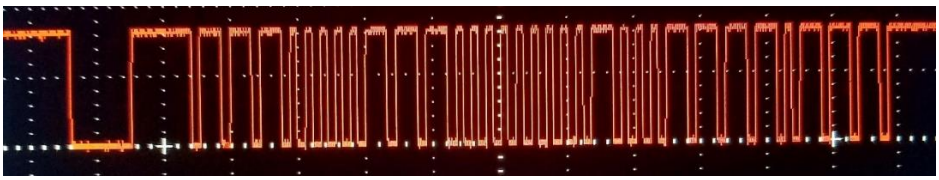
O dispositivo empregado no trabalho para troca de dados com o condicionador de ar trata-se de um Diodo Emissor de Luz (LED). É um semicondutor feito de arseneto de gálio e junção p-n que emite luz infravermelha quando ativado. Eles apareceram na década de 1960 e os primeiros LEDs emitiam luz infravermelha de baixa intensidade.

Os LEDs são capazes de serem ligados e desligados milhões de vezes por segundo, e essa característica é explorada por esse trabalho para fazer a modulação de dados nessa luz infravermelha pulsada. Sua luz é pulsada a 38 kHz formando uma onda portadora onde são modulados sinais lógicos marca e espaço. Marca é a presença da portadora e espaço é a ausência dela.

Um bit logico verdadeiro transmitido consiste em um espaço de 562,5 us e uma marca de 1,6875 ms (3 vezes 562,5 us). Um bit lógico falso consiste em um espaço de 562,5 us e uma marca de 562,5 us.

A transmissão de um comando consiste na modulação de um sinal de 28 bits correspondente à uma função logo após um sinal de aviso de transmissão que consiste em um espaço de 9 ms e uma marca de 4,5 ms, como mostrado na figura 3.

Figura 3 - Sinal infravermelho demodulado**.**



Fonte: Autor (2018)

2.3 INTERNET DAS COISAS

A Internet das Coisas, em poucas palavras, nada mais é que uma extensão da Internet atual, que proporciona aos objetos do dia-a-dia (quaisquer que sejam), mas com capacidade computacional e de comunicação, se conectarem à Internet. A conexão com a rede mundial de computadores viabiliza, primeiro, controlar remotamente os objetos e, segundo permitir que os próprios objetos sejam acessados como provedores de serviços. Estas novas habilidades, dos objetos comuns, geram um grande número de oportunidades tanto no âmbito acadêmico quanto no industrial. Todavia, estas possibilidades apresentam riscos e acarretam amplos desafios técnicos e sociais.

2.4 PROTOCOLO MQTT

De acordo com o *The Eclipse IoT Working Group*, o protocolo MQTT (*Message Queue Telemetry Transport*, usado na IoT) é projetado para conectar os dispositivos e redes do mundo físico, com aplicativos e *middleware* (Rotinas que fornecem serviços para aplicações de *software* além dos disponíveis pelo sistema operacional) usados no desenvolvimento de TI e *Web*, tornando-o um protocolo de conectividade ideal para uso na IoT e comunicação entre máquinas.

O protocolo MQTT segue o paradigma *publisher*/*subscriber*, onde um dispositivo central na rede, chamado de *broker*, é responsável por receber, enfileirar e disparar as mensagens recebidas dos *publishers* para os *subscribers*. O *publisher* é o cliente responsável por se conectar ao *broker* e publicar mensagens. Já o *subscriber* é o cliente responsável por se conectar ao *broker* e receber as mensagens relevantes.

Por se tratar de um protocolo que exige pouco processamento, ele é executado em dispositivos embarcados e plataformas móveis, ao mesmo tempo em que se conecta a servidores corporativos e *Web* altamente escalonáveis em redes com e sem fio. É útil para conexões com sistemas embarcados remotos onde um pequeno tamanho de código é necessário e / ou largura de banda de rede é um prêmio ou conectividade imprevisível e, para aplicativos móveis que exigem tamanho pequeno, baixo consumo de energia, pacotes de dados minimizados e distribuição eficiente de

informações para um ou muitos receptores.

Com baixo acoplamento e garantia de qualidade de serviço, o protocolo MQTT é otimizado para ambientes de sistemas dinâmicos, onde grandes volumes de mensagens e eventos do mundo físico precisam ser disponibilizados para servidores *Web*, corporativos e outros consumidores.

Como vantagens do protocolo MQTT pode-se mencionar é de domínio público, possui ferramentas de controle de acesso e criptografia da comunicação, suporta comunicações assíncronas e é baseado no protocolo tradicional da internet, o TCP/IP (*Transmission Control Protocol/Internet Protocol*).

2.5 LINGUAGEM JAVASCRIPT EM MICROCONTROLADORES

Javascript é uma linguagem de programação baseada em ECMAScript e padronizada pela Ecma *international* nas especificações ECMA-262[3] e ISO/IEC 16262. Tem fortes capacidades de programação orientada a objetos e a eventos. Por isso, ela é uma ótima escolha para ser usada na solução dos problemas encontrados na lida com comunicação em rede e na interação com seres humanos.

A linguagem foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido (FLANAGAN, 2001).

Espruino é um projeto que implementa um interpretador Javascript e permite que alguém com pouco treinamento possa usar para programar microcontroladores ESP32 e fazer experimentos com retornos imediatos, ou seja, não é necessário um programa completo para que o microcontrolador já execute alguma função ou resposta à eventos.

Essas características da linguagem, maturação do projeto Espruino e variedade das bibliotecas são grandes vantagens sobre as alternativas Python e Arduino. Por isso, a linguagem Javascript foi escolhida.

O interpretador é projetado para ser executado em dispositivos com pouca memória RAM e, mesmo assim, mantém o código fonte da aplicação completamente

nela. Ao encontrar o arquivo de programa, o interpretador lê o código fonte do um caractere por vez, reconhecendo os comandos e performando as ações até encontrar o final dele.

A velocidade de execução, por ter código interpretado sofre um pouco. De acordo com o desenvolvedor, um laço que inverte o estado lógico de um pino de saída do microcontrolador gera uma onda quadrada de 2 kHz em um microcontrolador rodando a 72 MHz. O ESP32, entretanto, gera uma frequência de 1,5 KHz.

2.6 PROJETOS RELACIONADOS

Alguns trabalhos, projetos e ideias semelhantes podem ser encontrados em publicações especializadas e na internet diferenciando desse projeto na aplicação especifica ou na implementação. Alguns desses trabalhos que foram pesquisados estão detalhados a seguir.

IRKit (IRKIT, 2018) é um sistema de código e *hardware* abertos de controle remoto infravermelho e acesso via WI-FI. Ele pode controlar qualquer eletroeletrônico que tenha controle remoto infravermelho como TVs, condicionadores de ar, luzes etc. Como já dito, ele usa a plataforma Arduino e a linguagem C++ para a escrita do código.

Control Anything (CONTROL ANYTHING, 2018) é um sistema proprietário que serve para conectar sistemas eletrônicos à internet através de sinais infravermelhos. Ele faz isso através de seu *hardware*, um *beacon* (um pequeno dispositivo eletrônico com função específica e faz parte de um sistema maior. Conecta a rede ao mundo físico) instalado nas redondezas do aparelho a ser conectado e um aplicativo para *smartphone.* Por ser um sistema proprietário, não se pode afirmar qual é a linguagem utilizada na sua programação.

No trabalho de conclusão de curso, (RODRIGUES, 2012) apresentou um controlador para sistemas de ar condicionado que transmite a grandeza lida via comunicação sem fio de e para um telefone celular. Foi modificada a placa de controle do sistema de ar condicionado e implementados os circuitos eletrônicos, utilizando como base a plataforma Arduino. Ainda, foi desenvolvido um aplicativo para o sistema Android que serviu como interface homem-máquina.

Já (LONGO, 2015), demonstrou a implementação de um sistema com o Raspberry PI se comunicando com um microcontrolador através de um par de chips de rádio nRF04L01. Foi demonstrada a implementação de um protocolo de comunicação e a criação de uma página WEB que faz a função de interface homem máquina de um controlador residencial composto por atuadores e sensores de temperatura, luminosidade, umidade e gás.

(SANTANA, Vitória R. S. et al, 2017) implementou um sistema de monitoramento e controle de condicionadores de ar baseado em um sensor de presença e relés conectados a um Arduino. Ao detectar a presença pessoas, o Arduino avisa via *bluetooth* a um telefone com sistema operacional Android e o habilita, então, a enviar sinais de controle para os relés. Esses relés, conectados ao ar condicionado, fornecem energia somente enquanto há pessoas por perto.

3 MATERIAIS E MÉTODOS

O presente trabalho toma como base documentações providas pelos fabricantes dos equipamentos, os projetos relacionados engenharia reversa executada no controle remoto do split LG.

3.1 UNIVERSO DE ABRANGÊNCIA DO TRABALHO

Devido às diversas possibilidades de variações e aplicações desse tipo de trabalho, faz-se necessária uma limitação da abrangência. São utilizados um sistema de processamento embarcado XTENSA ESP32, um sistema rodando Linux e um condicionador de ar LG. Todos conectados a uma rede WIFI, exceto o último.

O roteador WIFI, com rede protegida por senha, fornece uma proteção básica ao acesso não permito aos nós da rede. Apesar de existirem níveis maiores de proteção, o trabalho deixa oportunidade para uma futura exploração.

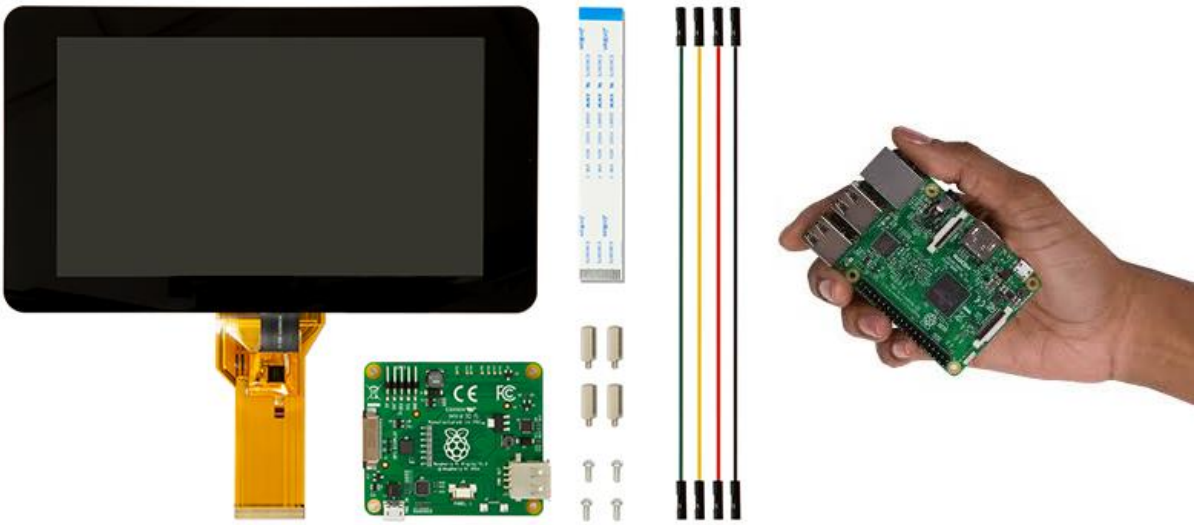
O sistema Linux faz o papel de IHM (Interface Homem Máquina) e fica na central de controle gerando comandos para o nó ESP32 através do protocolo MQTT.

3.2 PLACA DE DESENVOLVIMENTO RASPBERRY PI 3B

O Raspberry PI é uma série de computadores de placa única desenvolvidos no Reino Unido pela Fundação Raspberry PI para promover o ensino de ciência da computação em escolas e países emergentes. De acordo com (RASPBERRY PI, 2018), até março de 2018, mais de 19 milhões desses computadores já tinham sido vendidos.

Sua placa contém *hardware* que possibilita conexão à internet e execução do sistema operacional Linux e, quando conectada a um display *TFT* (*Thin Film Transistor*) com sensor de toques capacitivo, permite a criação de um sistema supervisório completo. Na figura 4 são mostrados os componentes desse computador.

Figura 4 - Raspberry PI 3B



Fonte: Raspberry Pi (2018)

3.3 XTENSA ESP32

O ESP32 é um sistema de núcleo duplo com duas unidades de processamento com arquitetura Harvard Xtensa LX6. Toda a memória embarcada, memória externa e periféricos estão locados no barramento de dados e/ou barramento de instruções dessas duas unidades. Com algumas exceções, o mapeamento de endereços das duas unidades é simétrico, significando que elas usam os mesmos endereços para acessarem a mesma memória. Múltiplos periféricos no sistema conseguem acessar a memória diretamente. As unidades de processamento são chamadas “PRO_CPU” e “APP_CPU” (para “protocolo” e “aplicação”), entretanto, para a maioria dos usos elas são intercambiáveis.

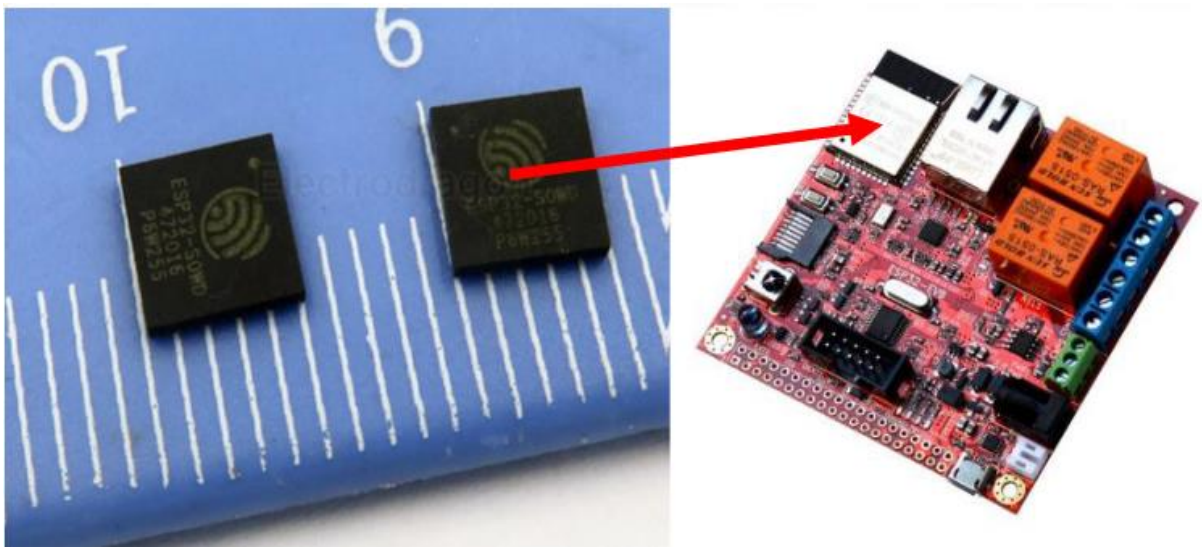
Características:

- Controlador WiFi: padrão 802.11 B / G / N, operando nas faixas de 2,4 a 2,5 GHz
- Controlador Bluetooth v4.2 BLE (*Bluetooth Low Energy*)
- Modos de operação: cliente, ponto de acesso, estaçao + ponto de acesso
- Microprocessador de dois núcleos Tensilica Xtensa 32-bit LX6

- *Clock* ajustável de 80 MHz até 240 MHz
- Tensão de operação: 3.3 VDC
- Memória *SRAM* (*Static Random Access Memory*) de 512KB
- Memória *ROM* (*Read Only Memory*) de 448KB
- Memória *flash* externa (4 megabytes)
- 36 pinos de uso geral GPIO
- GPIOs com *PWM* / *I2C* e função *SPI*

A figura 5 mostra, em comparação com uma régua, o tamanho diminuto (5 mm por 5 mm) do componente.

Figura 5 - Microcontrolador Xtensa ESP32



Fonte: ELECTRODRAGON (2013)

.

3.4 AR CONDICIONADO SPLIT LG

É um sistema de refrigeração usado em equipamentos condicionadores de ar, que são similares aos equipamentos do tipo janela, sendo divididos em dois módulos, denominados unidade interna (evaporadora) e unidade externa (condensadora).

Essas duas partes estão unidas por tubulações de cobre onde acontece a passagem do gás refrigerante e do dreno. O dreno é necessário para que aconteça o escoamento da água da evaporadora, que ocorre devido a condensação da umidade

do ambiente interno.

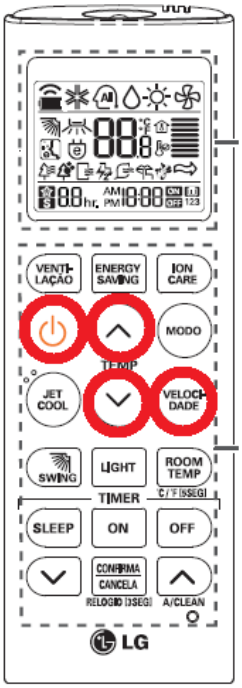
A característica principal desse sistema é o fato da unidade interna ser mais silenciosa do que o sistema convencional. O modelo escolhido é o USNW182CSZ2, um sistema residencial com capacidade térmica de 18.000 BTUs. A figura 6 mostra o sistema split completo utilizado nesse trabalho.

Figura 6 - Ar condicionado split LG

Fonte: MAGAZINE LUIZA (2013)

Esse equipamento possui diversas funções, mas somente algumas funções básicas são emuladas. Elas são: velocidade do ventilador, liga-desliga e temperatura. O controle remoto do sistema tem fluxo de dados unidirecional, ou seja, somente o controle emite dados. Esses dados são codificados de forma absoluta, e não incremental como é feito nos televisores. Portanto, ao receber um comando de nova temperatura do *Broker*, ela será enviada como tal pelo nó ESP32 para o ar condicionado. Quando desligado, o ar condicionado memoriza a última configuração feita. A figura 7 apresenta os botões que tem suas funções imitadas pela ESP32-EVB.

Figura 7 - Controle remoto infravermelho



Fonte: LG (2013)

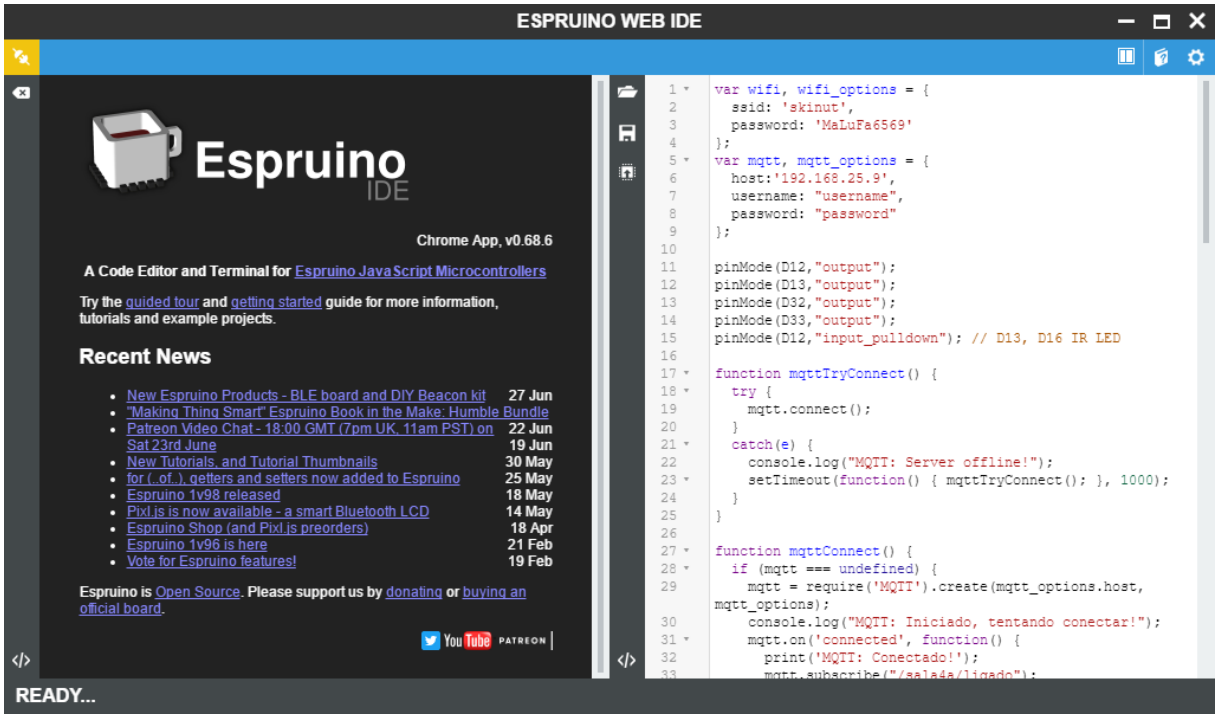
3.5 PROGRAMAS UTILIZADOS

O trabalho apresentado é bastante abrangente e são necessárias duas plataformas de *hardware* diferentes para o alcance dos seus objetivos. Essas plataformas e todas as ferramentas envolvidas são programadas na linguagem Javascript.

3.5.1 Espruino Web IDE

A Espruino WEB IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado) é a Interface para a programação em Javascript do ESP32. Ela tem um editor com sintaxe destacada, suporta carregamento dinâmico de módulos e é capaz de fazer a depuração do código enquanto ele é executado. A figura 8 mostra como é sua janela principal.

Figura 8 - Espruino WEBIDE



Fonte: ESPRUINO WEBIDE (2018)

3.5.2 Node.Js

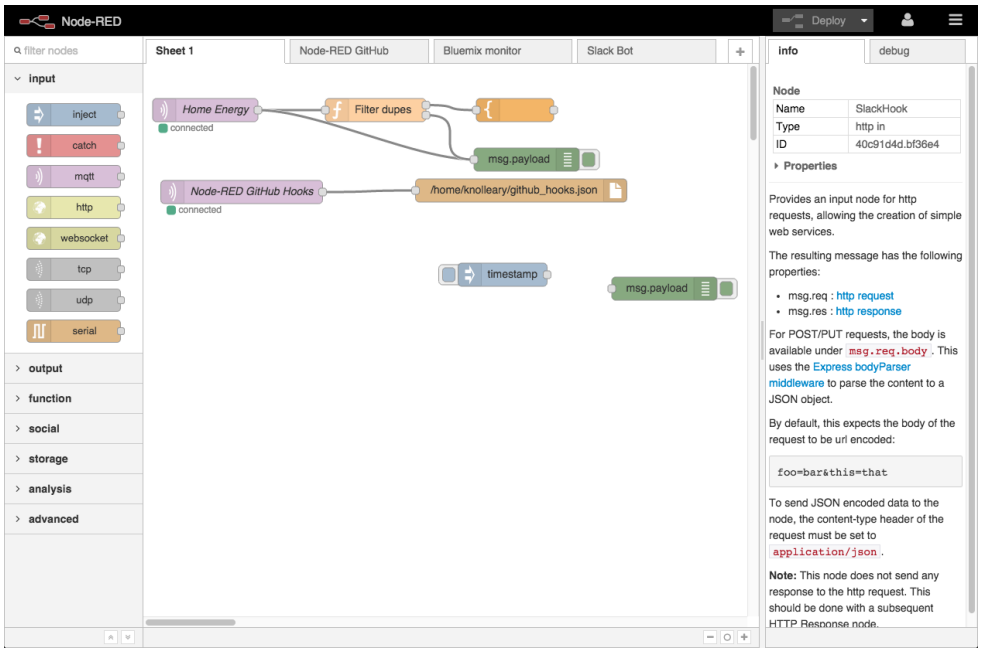
Node.js é um interpretador de código Javascript com o código aberto, focado em migrar o Javascript do lado do cliente para servidores. Seu objetivo é ajudar programadores na criação de aplicações de alta escalabilidade (como um servidor web), com códigos capazes de manipular dezenas de milhares de conexões simultâneas, numa única máquina física. O Node.js é baseado no interpretador *V8 Javascript Engine* (interpretador de Javascript de código aberto implementado pelo Google em C++ e embutido no Chrome). Foi criado por Ryan Dahl em 2009, e seu desenvolvimento é mantido pela fundação Node.js em parceria com a Fundação Linux (NODEJS, 2018).

Sua grande utilidade está na capacidade de criação de módulos externos que são executados como programas e podem ser rodados independentemente ou em conjunto para criar aplicações ainda maiores e mais complexas.

3.5.3 Node-Red

Node-RED é uma ferramenta de programação para fazer a interligação de diversos *hardwares*, APIs e serviços online em novas e interessantes maneiras. A solução implementa um editor baseado em navegador *web*, que possibilita a ligação de diversos nós e é capaz de gerar um código final com um simples clique. Cada nó é capaz de gerar e consumir mensagens. A figura 9 mostra a interface da ferramenta (NODE-RED, 2018).

Figura 9 - NODE-RED



Fonte: Autor (2018)

Essas conexões entre nós dizem ao Node-Red como fazer a translação de mensagens e as propriedades configuradas nos nós descrevem se as imagens dos nós aparecem ou não na página criada. Essa página pode conter controles como botões, gráficos, imagens, etc. Esses controles são atualizados de acordo com o fluxo de mensagens entrando e saindo do *workspace*.

O Node-red é criado em cima do Node.js e tira toda a vantagem do seu modelo sem bloqueios e baseado em eventos. Isso o faz ideal para ser usado em conjunto

com o Raspberry PI. Existem certa de 225.000 módulos disponíveis em seu repositório e é simples de adicionar novas capacidades. Funções podem ser criadas em Javascript utilizando o editor embutido ou então selecionadas da biblioteca disponível com o pacote.

3.5.4 Mosca

Mosca é um módulo Node.js, de código aberto, para a criação de um *Broker* MQTT e faz a ligação de vários nós da Internet das Coisas através do protocolo MQTT. É um *software* que é executado em plano de fundo, como serviço e, portanto, não tem interface gráfica. Um programa em Javascript mostrado no anexo B faz a interligação entre os dois projetos.

3.6 METODOLOGIA

A unidade central deve gerenciar o controle de uma ou mais unidades remotas através de um computador de placa única capaz de rodar o sistema operacional Linux, Node.js, *broker* MQTT e a *framework* node-red. Já a unidade remota deve ser construída com baixo custo em mente, requerendo para isso, o uso de microcontroladores ao invés de processadores como no caso da unidade central e um interpretador Javascript.

O desenvolvimento do trabalho consiste em 6 passos listados a seguir.

a) montar uma unidade central baseado no Raspberry PI 3B;
b) montar uma unidade remota baseada no microcontrolador ESP32;
c) programar a unidade central utilizando o *framework* node-red;
d) programar a unidade remota utilizando um interpretador Javascript;
e) executar efetivamente o controle remoto em pelo menos uma sala da edificação;
f) mostrar que o sistema pode ser ampliado significativamente.

4 DESENVOLVIMENTO

Para a obtenção de um sistema completo, o Raspberry PI e a ESP32-EVB precisam ser configuradas e programadas. Cada uma delas usa uma série de ferramentas e procedimentos diferentes, mas tais que, no final do processo, seja possível a realização de uma comunicação eficaz entre as mesmas.

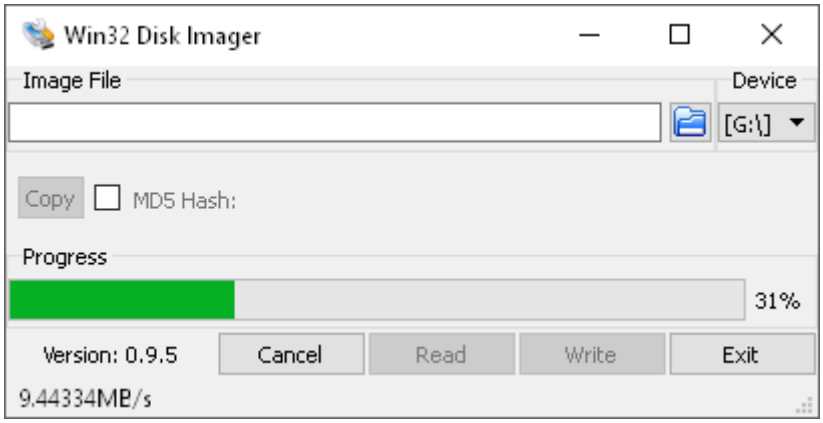
Em outros trabalhos, diversas implementações básicas de sistemas equivalentes ao proposto nesse trabalho são efetuadas com linguagem C e bibliotecas de código aberto como as oferecidas pelo outro projeto de código aberto Arduino. Já a técnica proposta neste trabalho não faz alteração física no sistema controlado e inclui uma linguagem de alto nível para a programação tanto do sistema central quanto o remoto, pois tais linguagens adicionam uma camada a mais de abstração entre o desenvolvedor do sistema e o *hardware* envolvido. Assim é possível focar-se somente na solução e não nos detalhes técnicos de cada sistema computacional utilizado.

4.1 COMPUTADOR DE PEQUENO PORTE RASPBERRY PI 3B

O computador de pequeno porte Raspberry PI vem de fábrica sem qualquer *software* instalado, portanto, um procedimento de inicialização é necessário antes que se faça uso dela.

O fabricante disponibiliza gratuitamente uma distribuição do Linux (RASPBIAN) especialmente adaptada ao *hardware* do Raspberry no seu site, em um arquivo de imagem de um cartão de memória. Essa imagem é utilizada para a criação de uma memória pronta para uso através de uma ferramenta chamada WIN32 DISK IMAGER. A interface dessa ferramenta é mostrada na figura 10.

Figura 10 - Win32 Disk Imager



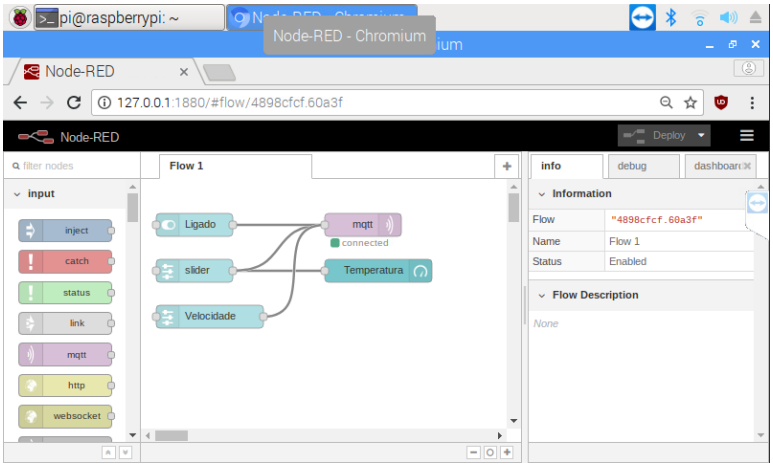
Fonte: Autor (2018)

O cartão já preparado pela ferramenta é inserido no soquete para cartão de memória instalado no Raspberry PI e em seguida uma fonte de alimentação de 5 volts com capacidade de corrente de pelo menos 2,5 Ampères conectada aos seus terminais de alimentação.

O Raspberry PI inicia o processo de carregamento do Linux e fornecerá um terminal de linha de comando para que se possa fazer as configurações complementares listadas no anexo C.

Nesse ponto, o *broker* MQTT mosca já está sendo executado em segundo plano e o sistema aceitará a conexão do ESP32. O Node-red executará no sistema local na porta 1880 e o acesso mostrará, como na imagem da figura 11, o seu *workflow*, que é a interface de entrada da sua programação.

Figura 11 - Um *workflow* do node-red



Fonte: Autor (2018)

Os blocos necessários para o funcionamento desse trabalho são duas barras de rolagem, um botão, um dial e uma conexão ao *broker* MQTT. Cada um desses blocos gera uma mensagem para um tópico a ser publicado. A interconexão entre esses itens mostra também como se dá o fluxo de mensagens. Os blocos têm entrada à esquerda e saída à direita.

O *broker* Mosca é executado em segundo plano no Raspberry e cria, internamente, os tópicos (ligado, temperatura e velocidade) assim que um cliente os acessa.

Os acessos vêm tanto do nó MQTT do *workspace* do node-red quanto de cada placa ESP32 conectada. O Raspberry PI, executando a interface do node-red faz as publicações e as placas ESP32 recebem esses dados.

4.2 PLACA DE DESENVOLVIMENTO ESP32-EVB

A seleção de uma placa de desenvolvimento adequada para a implementação desse trabalho, idealmente, seria uma já disponível no mercado. Por já prover *hardware* de conectividade WI-FI, foi escolhida a ESP32-EVB, da Olimex. Somente são necessários adição de um LED emissor infravermelho e um programa de controle para que se tenha um protótipo funcional.

A placa já possui um LED emissor e um receptor infravermelho, mas a ligação do emissor não permite o uso de uma técnica simples para fazer a modulação do sinal. A solução escolhida foi a adição de um LED externo entre dois pinos do microcontrolador. Essa ligação permite que um deles forneça o nível lógico a ser transmitido enquanto outro faz a modulação em 38 KHz.

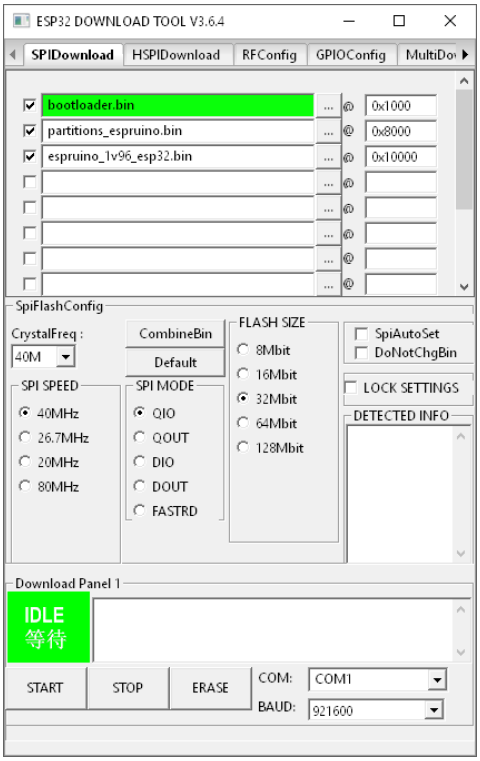
A figura 12 mostra a placa de circuito impresso do protótipo, seu diagrama elétrico é visto no anexo A e a ligação do LED externo é mostrada nos apêndices A e B.

Figura 12 - Placa de desenvolvimento ESP32-EVB.

Fonte: OLIMEX (2018)

Da mesma forma que a Raspberry, a ESP32 vem sem *software* instalado e outro procedimento é necessário para que ela execute o código do Espruino. A ferramenta fornecida pela Xtensa é a ESP32 *DOWNLOAD TOOL*, que deve ser configurada com a seleção dos arquivos do Espruino e seus endereçamentos como mostra a figura 13.

Figura 13 - ESP32 *DOWNLOAD TOOL*

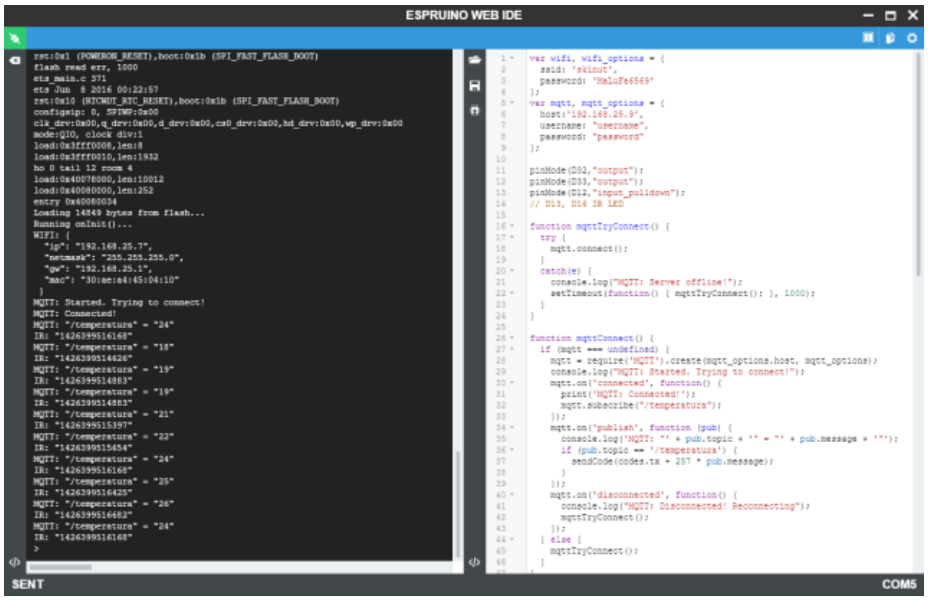


Fonte: Autor (2018)

Ao finalizar a gravação dos arquivos obtidos de (ESPRUINO 2017), a execução do seu código inicia imediatamente e, para ter acesso aos seus comandos, deve-se usar a sua IDE. Fazendo a conexão com um cabo *USB* (*Universal Serial Bus*), o terminal do interpretador é mostrado e a placa está pronta para ser programada com a aplicação final em Javascript.

O código do apêndice C são as rotinas em Javascript que fazem a conexão ao roteador Wifi e ao Mosca, no Raspberry PI. A variável ‘sala’ serve para configurar o nome da sala onde o nó está. A figura 14 mostra a aplicação em funcionamento.

Figura 14 - Aplicação em funcionamento

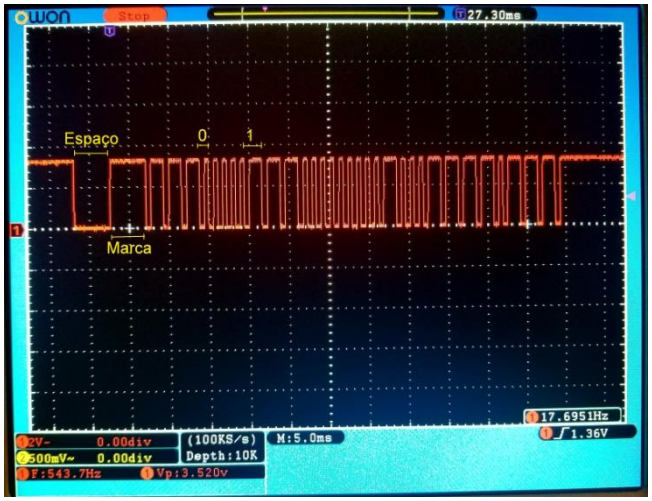


Fonte: Autor (2018)

4.2.1 Engenharia reversa dos códigos de controle infravermelho

Para fazer a reprodução do sinal necessário para controlar o ar condicionado LG foi necessária uma análise do sinal do seu receptor através de um osciloscópio que capturou formas de onda geradas pelo controle remoto infravermelho original do ar condicionado e decodificadas pelo componente marcado como U7 na placa ESP32-EVB. A onda capturada é mostrada na figura 15.

Figura 15 - Sinal do receptor infravermelho



Fonte: Autor (2018)

Seguindo a lógica descrita na seção 2.2, os comandos liga, desliga, diferentes temperaturas e velocidades do ventilador foram analisados e a suas sequências binárias anotadas, gerando a tabela 1. Os comandos de temperatura (de 18 °C a 30 °C) e velocidade (de 1 a 5) geram sequências com valores em intervalos regulares de que podem ser descritos por uma equação matemática, a fim de diminuir o tamanho da matriz de dados a ser inserida no programa. Essas sequências binárias são usadas pelo programa para gerar a temporização do acionamento do LED infravermelho.

Tabela 1 - Códigos dos controles infravermelhos

Comando	Sequência binária (Base)	Parâmetro
Liga	10001000000000001111101000011	
Desliga	10001000110000000000001010001	
Temperatura = Base + 257 * Parâmetro	10001000000010000011101001111	Intervalo de 18 a 30
Velocidade = Base + 34 * Parâmetro	10001000000010000011100001011	Intervalo de 1 a 5

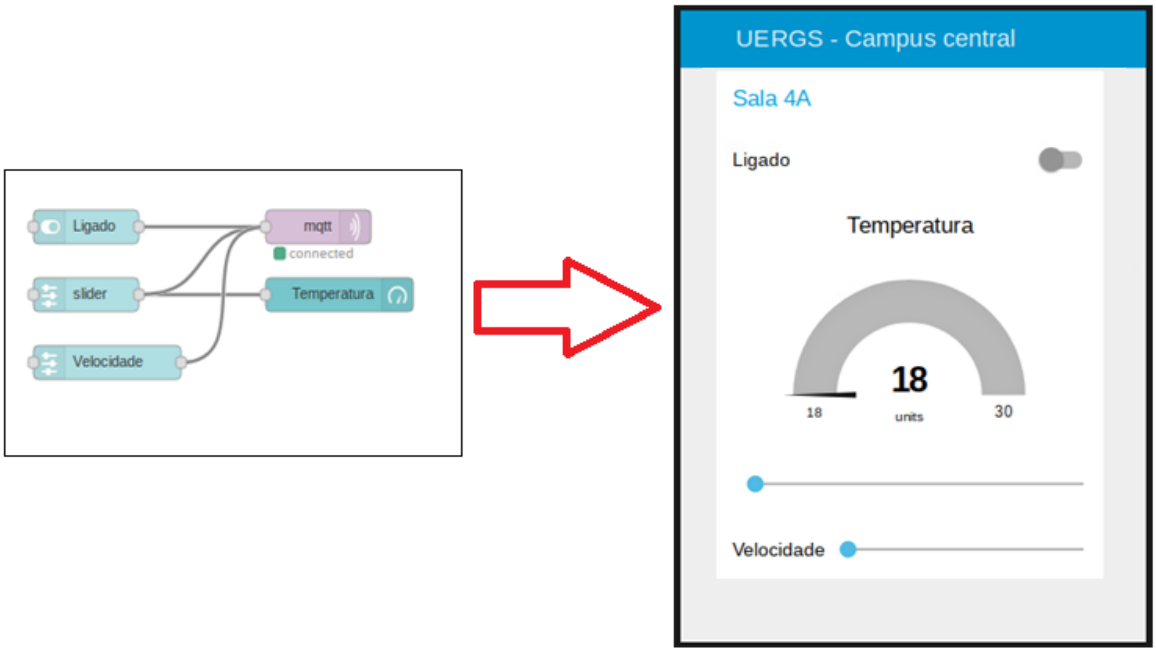
Fonte: Autor (2018)

A linguagem Javascript possui uma construção e sintaxe que facilita muito o trabalho de geração de pulsos com duração pequena e grande precisão. Um vetor é formado com a posição, no tempo, de cada borda do sinal a ser gerado e a função digitalPulse(), presente no Espruino, gera o trem de pulsos e retorna somente após o término do último pulso no pino do ESP32.

5 RESULTADOS

Com o workflow do apêndice D carregado no Node-red, e saída mostrada na figura 16, obtém-se a funcionalidade mínima necessária para a publicação nesses três tópicos: “/sala4a/ligado”, “/sala4a/temperatura” e “/sala4/velocidade”.

Figura 16 - *Workflow* e saída do Node-red



Fonte: Autor (2018)

Ao ligar a alimentação da placa ESP32-EVB, ela automaticamente se conecta à rede Wifi configurada no código fonte e ao *Broker* Mosca, que é executado no Raspberry PI. O ESP32 se inscreve nos tópicos ligado, velocidade e temperatura. A saída do programa é mostrada no apêndice E. O ESP32 é capaz de se reconectar automaticamente tanto ao roteador Wifi e ao *Broker* caso a conexão falhe.

Tanto o mouse como o toque com um dedo acionam os controles que, ao serem movimentados, disparam mensagens para os tópicos predefinidos.

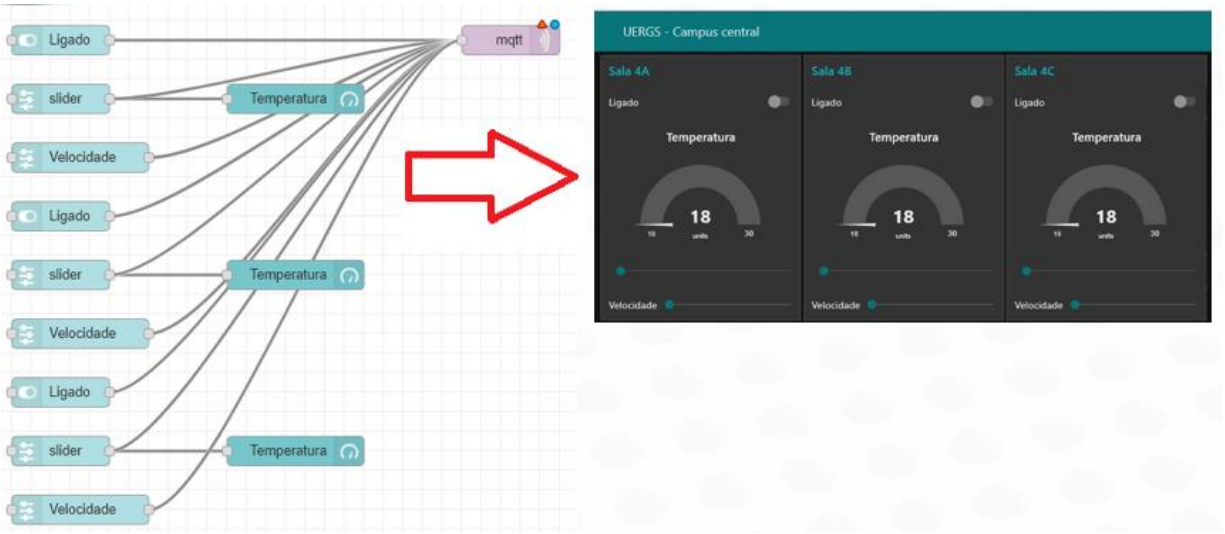
Ao receber mensagens de um tópico, o ESP32 faz uma chamada à função sendCode(code), que aciona o LED infravermelho e faz o comando remoto do ar condicionado.

Cada ESP32 se inscreve à uma porção de tópicos. A quantidade de tópicos que o Mosca é capaz de rotear só é limitado pela memória do computador. O

Raspberry PI tem 1 GB de memória RAM, que é suficiente para milhares de salas gerenciadas.

A replicação do *workflow* do apêndice D, com alteração nos nomes dos tópicos, permite a ampliação do sistema de forma muito simples. Para cada nova tela, uma ESP32-EVB extra é adicionada à sala correspondente. A figura 17 mostra o resultado de uma ampliação para o controle de três salas.

Figura 17 - Múltiplas salas controladas



Fonte: Autor (2018)

6 CONCLUSÃO

Neste trabalho foi feito um estudo sobre algumas técnicas para conectar à internet dispositivos inicialmente não projetados com essa função, e testes demonstraram na prática o seu funcionamento. Isto foi feito através do desenvolvimento de um sistema expansível para o aumento da eficiência energética de uma edificação através do controle de condicionadores de ar. Alguns autores modificaram tais dispositivos para adicionar funcionalidades e este autor escolheu beneficiar-se dos meios de acesso já providos pelos fabricantes, como o controle remoto infravermelho, atuando de forma minimamente invasiva. Isto apresenta também como vantagem a conveniência de permitir a substituição do equipamento por um equivalente sem retrabalho de conversão do equipamento.

O trabalho com a linguagem C ainda exige grandes cuidados por parte do programador, devido à essa linguagem estar tão perto da linguagem de máquina. A linguagem Javascript utilizada em conjunto com o projeto Espruino simplificou a tarefa de programação e implementação da comunicação em rede. O custo pago é o desempenho na execução do programa, mas isso não é crítico para esta aplicação.

Nesse trabalho, o Node-red gerencia toda a comunicação entre o *Broker* e a interface, além de renderizar todas as telas utilizando a linguagem HTML (*HyperText Markup Language* ou Linguagem de Marcação de Hipertexto) e um navegador padrão de internet. A questão de segurança e controle de acesso é tratada pelo Node-red através de arquivos de configuração.

Consegue-se mostrar, também, que a ampliação desse sistema se dá pela recriação de mais módulos ESP32-EVB e multiplicação das telas do Node-red, alterando somente os tópicos no *Broker* MQTT.

REFERÊNCIAS

ABCM. **Associação Brasileira De Engenharia E Ciências Mecânicas**. 2018. Disponível em: <http://www.abcm.org.br/app/webroot/anais/conem/2010/PDF/CON10-1751.pdf>. Acesso em 13 jun. 2018.

ALTHOUSE, Andrew D.; TURNQUIST, Carl H.; BRACCIANO, Alfred F. **Modern Refrigeration and Air Conditioning 18th Edition** ed. [S.l.]: Goodheart-Wilcox Publishing. 2003.

CCOHS. **Canadian Centre for Occupational Health and Safety**. 2018. Disponível em: <http://www.ccohs.ca/oshanswers/phys_agents/thermal_comfort.html>. Acesso em 13 jun. 2018.

CONTROL ANYTHING. **Control Anything**. 2018. Disponível em: <http://www.beaconsandwich.com/control-anything.html>. Acesso em 13 jun. 2018.

DE DEAR, Richard; BRAGER, Gail**. Developing an adaptive model of thermal comfort and preference**. ASHRAE Transactions. 1998.

DEPARTMENT OF ENERGY. **History of Air Conditioning**. 2018. Disponível em: <https://www.energy.gov/articles/history-air-conditioning>. Acesso em 13 jun. 2018.

ECMA-262. **ECMAScript® 2017 Language Specification**. Disponível em: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf>. Acesso em 13 jun. 2018.

EDISONTECHCENTER. **The First Practical LED**. 2015. Disponível em: <http://www.edisontechcenter.org/lighting/LED/TheFirstPracticalLED.pdf>. Acesso em 13 jun. 2018.

ELECTRODRAGON. **ESP32 Wifi & Bluetooth IC**. 2013. Disponível em: <http://www.electrodragon.com/product/esp32-wifi-bluetooth-ic-model/>. Acesso em 13 jun. 2018.

ESPRUINO. **Original Espruino Board**. 2017. Disponível em: <http://www.espruino.com/Original>. Acesso em 13 jun. 2018.

ESPRUINO WEBIDE. **Espruino Web IDE**. 2018. Disponível em: <https://www.espruino.com/Web+IDE>. Acesso em 13 jun. 2018.

FABBRI, K. **A Brief History of Thermal Comfort: From Effective Temperature to Adaptive Thermal Comfort.** In: Indoor Thermal Comfort Perception. Springer International Publishing. Cham. 2015.

FLANAGAN, David; FERGUSON, Paula. **JavaScript: The Definitive Guide 4th ed**. [S.l.]: O'Reilly & Associates. 2002.

GREEN, Hank. **Transmitting Data Through LED Light Bulbs**. EcoGeek. 2008.

HALLIDAY, Resnick, Walker. **Física Vol. 2**. Rio de Janeiro: Livros Técnicos e Científicos. LTC. 1996. 293p.

IRKIT. **Opensource Infrared Remote Controller**. 2018. Disponível em: <http://getirkit.com/en/>. Acesso em 13 jun. 2018.

K. FABBRI. **Indoor Thermal Comfort Perception**. Springer International Publishing**,** Suíça, 2015.

LONGO, Lucas**. Internet das coisas: uso de sensores e atuadores na automação de um protótipo residencial**. (trabalho de conclusão de curso) Universidade Tecnológica Federal do Paraná. Departamento Acadêmico de Informática. Curso de Engenharia de Computação. 2015**.**

LEITE, Mário; **Técnicas de programação: Uma abordagem moderna**. Rio de Janeiro: Brasport, 2006. 405 p.

LG. **Manual do usuário**. 2013. Disponível em: <http://gscs-b2c.lge.com/downloadFile?fileId=KROWM000559527.pdf>. Acesso em 13 jun. 2018.

MAGAZINE LUIZA. **Ar-Condicionado Split LG Inverter 11.500 BTUs**. 2018. Disponível em: <https://www.magazineluiza.com.br/ar-condicionado-split-lg-inverter-11.500-btus-quente-frio-libero-e-us-w122hsg3/p/0114745/ar/arsp/>. Acesso em 13 jun. 2018.

MANCINI, Mônica. **Internet das Coisas: História, Conceitos, Aplicações e Desafios**. 2017. PMI São Paulo. 13 de jun. 2018

MINISTÉRIO DE MINAS ENERGIA. **MME abre consulta pública sobre eficiência de Refrigeradores, congeladores e condicionadores de ar e transformadores de distribuição**. 2018. Disponível em: <http://www.mme.gov.br/web/guest/pagina-inicial/outras-noticas/-/asset_publisher/32hLrOzMKwWb/content/mme-abre-consulta-publica-sobre-eficiencia-de-refrigeradores-congeladores-e-condicionadores-de-ar-e-transformadores-de-distribuicao>. Acesso em 13 jun. 2018.

NODEJS. **NODE.JS**. 2018. Disponível em: <https://pt.wikipedia.org/wiki/Node.js>. Acesso em 21 jun. 2018

NODE-RED. **NODE-RED**. 2018. Disponível em: <https://nodered.org/>. Acesso em 21 jun. 2018.

OLIMEX. **ESP32-EVB**. 2018. Disponível em: <https://www.olimex.com/Products/IoT/ESP32-EVB/open-source-hardware>. Acesso em 13 jun. 2018.

PHILCHEN. **How to setup a mosca, node.js, mqtt and broker service on Ubuntu 14.04**. 2018. Disponível em: <http://www.philchen.com/2015/05/04/how-to-setup-a-mosca-node-js-mqtt-broker-service-on-ubuntu-14-04>. Acesso em 21 jun. 2018.

RASPBIAN. **Raspberry PI Linux Distribution**. 2018. Disponível em:<https://www.raspberrypi.org/downloads/raspbian/>. Acesso em 13 jun. 2018.

OKON, Thomas M.; biard, James R. **The First Practical LED**. EdisonTechCenter. 2015

RASPBERY PI. **Raspberry PI**. 2018. Disponível em: <https://www.raspberrypi.org/>. Acesso em 13 jun. 2018.

RODRIGUES, Douglas A. de Souza**. Monitoramento e controle sem fio de sistemas de refrigeração**. Florianópolis. 2012. Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina. 2012.

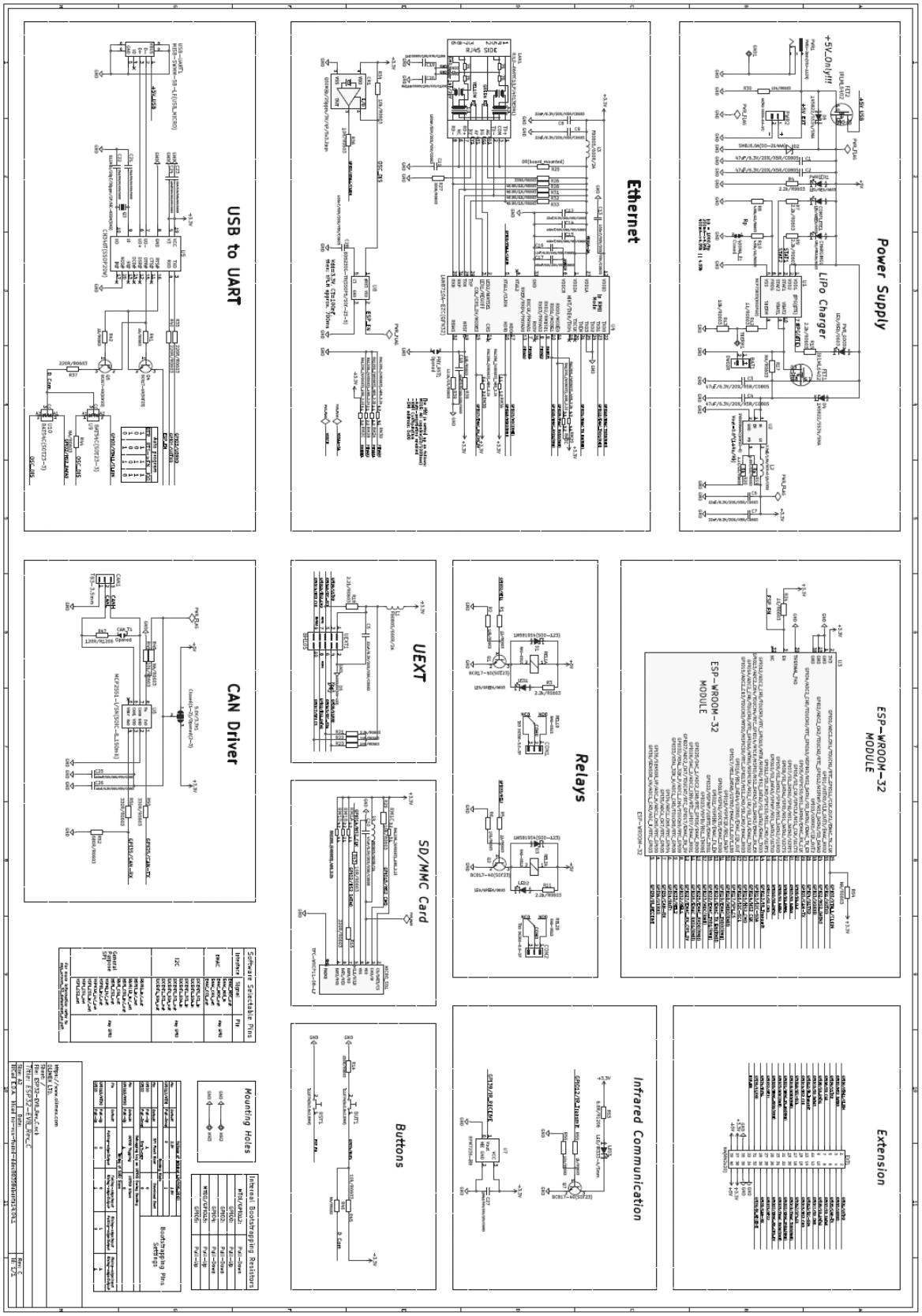
SANTANA, Vitória R. S. et al. **Desenvolvimento de um sistema de monitoramento e controle de condicionadores de ar a baixo custo. In: encontro nacional de jovens investigadores**. 2017. Fortaleza**.** Editora Realize, 2017.

SANTOS, Bruno P. et al. **Internet das Coisas:** da Teoria à Prática. Departamento de Ciência da Computação Universidade Federal de Minas Gerais. 2018.

SERWAY, Raymond. **Física para Engenheiros e Cientistas**. California State Polytechnic University, Pomona: brooks/Cole. 2008. 587 p.

WIN32 DISK IMAGER. **WIN32 DISK IMAGER**. 2018. Disponível em: <https://sourceforge.net/projects/win32diskimager/>. Acesso em 13 jun. 2018.

ANEXO A – DIAGRAMA ELÉTRICO OLIMEX ESP32-EVB



Fonte: OLIMEX (2018)

ANEXO B – CÓDIGO DA APLICAÇÃO MOSCA-APP.JS

```
console.log(process.pid);
require('daemon')();
var mosca = require('mosca')

var backjack = {
  type: 'redis',
  db: 12,
  port: 6379,
  return_buffers: true,
  host: "localhost"
};

var moscaSettings = {
  port: 1883,
  backend: backjack,
  persistence: {
    factory: mosca.persistence.Redis
  }
};

var server = new mosca.Server(moscaSettings);
server.on('ready', setup);

server.on('clientConnected', function(client) {
    console.log('client connected', client.id);
});

server.on('published', function(packet, client) {
  console.log('Published', packet.payload);
});

function setup() {
  console.log('Mosca server is up and running')
}
console.log(process.pid);
```

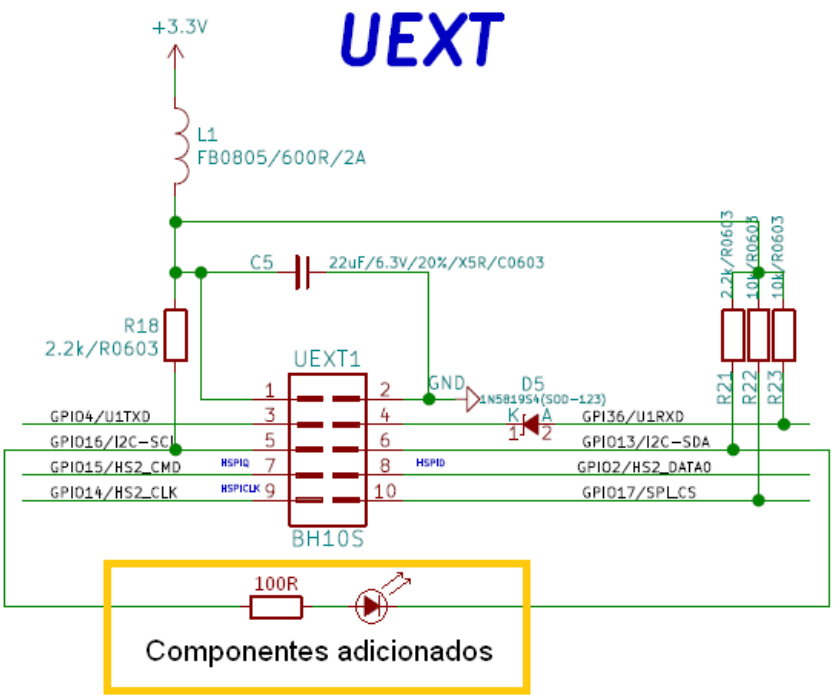
Fonte: PHILCHEN (2015)

ANEXO C – COMANDOS PARA INSTALAR FERRAMENTAS

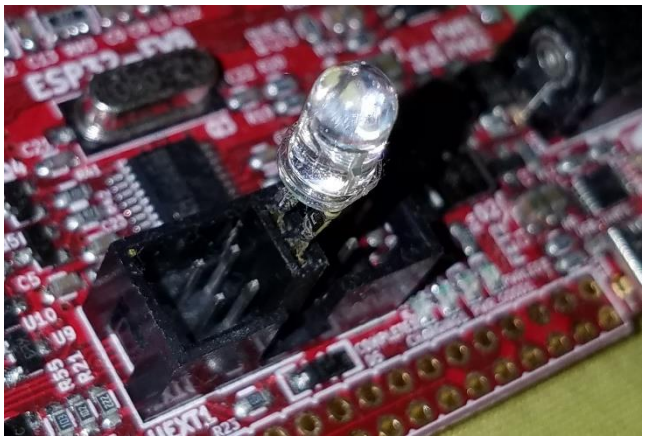
```
sudo apt-get update
sudo ntpdate pool.ntp.org
sudo apt-get install ntp
sudo apt-get install build-essential
sudo apt-get install tcl8.5
wget http://download.redis.io/releases/redis-3.0.0.tar.gz
gunzip -c redis-3.0.0.tar.gz | tar -xvf -
cd redis-3.0.0/
sudo make
sudo make install
cd utils/
sudo ./install_server.sh
cd ..
sudo apt-get update
sudo apt-get install nodejs
sudo ln -s /usr/bin/nodejs /usr/bin/node
sudo apt-get install npm
sudo npm install debug
sudo npm install mosca bunyan -g
sudo npm install daemon
sudo npm install node-red
node mosca-app.js
node node-red
```

Fonte: PHILCHEN (2015)

APÊNDICE A – DIAGRAMA DE LIGAÇÃO ELETRÔNICA



APÊNDICE B – LIGAÇÃO DO LED EXTERNO



APÊNDICE C – CÓDIGOS DE PROGRAMA DA ESP32-EVB

```
var wifi, wifi_options = {
  ssid: '****',
  password: '****'
};
var mqtt, mqtt_options = {
  host:'192.168.25.9',
  username: "username",
  password: "password"
};
var sala = 'sala4a'; // nome da sala

pinMode(D12,"input_pulldown");

function mqttTryConnect() {
  try {
    mqtt.connect();
  }
  catch(e) {
    console.log("MQTT: Server offline!");
    setTimeout(function() { mqttTryConnect(); }, 1000);
  }
}

function mqttConnect() {
  if (mqtt === undefined) {
    mqtt = require('MQTT').create(mqtt_options.host, mqtt_options);
    console.log("MQTT: Iniciado, tentando conectar!");
    mqtt.on('connected', function() {
      print('MQTT: Conectado!');
      mqtt.subscribe('/'+sala+'/ligado');
      mqtt.subscribe('/'+sala+'/velocidade');
      mqtt.subscribe('/'+sala+'/temperatura');
    });
    mqtt.on('publish', function (pub) {
      console.log('MQTT: "' + pub.topic + '" = "' + pub.message + '"');
      if (pub.topic == '/'+sala+'/ligado') {
        if (pub.message == 1) {
            sendCode(codes.on);
        } else {
            sendCode(codes.off);
        }
      }
      if (pub.topic == '/'+sala+'/temperatura') {
        sendCode(codes.tx + 257 * pub.message);
      }
      if (pub.topic == '/'+sala+'/velocidade') {
        sendCode(codes.vx + 34 * pub.message);
      }
    });
    mqtt.on('disconnected', function() {
      console.log("MQTT: Desconectado! Reconectando...");
      mqttTryConnect();
    });
  } else {
    mqttTryConnect();
  }
}

function wifiConnect() {
  wifi = require('Wifi');
  wifi.connect(wifi_options.ssid, {password: wifi_options.password},
    function(e) {
      if (!e) {
        setTimeout(function() { mqttConnect(); }, 1000);
      } else {
        console.log("WIFI: " + e);
      }
```

```
  });
  wifi.on('connected', function() {
    console.log('WIFI:', wifi.getIP());
    setTimeout(function() { mqttConnect(); }, 1000);
  });
  wifi.on('disconnected', function() {
    console.log("WIFI: Desconectado! Reconectando...");
    wifi.connect(wifi_options.ssid, {password: wifi_options.password});
  });
}

function onInit() {
  wifiConnect();
}

var codes = {
  on :[0b10001000000000000111101000011], // liga
  off:[0b10001000110000000000001010001], // desliga
  tx :[0b10001000000001000001101001111], // + 257 * C
  vx :[0b10001000000001000001100001011], // + 34 * V
};

function sendCode(code) {
  print('IR: "' + code + '"');
  var t = [];
  t.push(9.0, 4.0);
  for (var i=27;i>=0;i--) {
    t.push(0.65);
    if (code&(1<<i)) {
      t.push(1.6);
    } else {
      t.push(0.45);
    }
  }
  analogWrite(D13, 0.9, {freq:38000} ); // Inicia portadora
  digitalPulse(D16, 0, t); // Envia os pulsos
  digitalPulse(D16, 1, 0); // Espera enviar o último pulso
  digitalWrite(D13, 0);    // Desliga a portadora
}
```

APÊNDICE D – WORKFLOW NODE-RED


```
[{"id":"1c53a287.59357d","type":"ui_slider","z":"4898cfcf.60a3f","name":"","label":"","group":"c8da5a75
.cebb98","order":3,"width":0,"height":0,"passthru":true,"topic":"/sala4a/temperatura","min":"18","max":
"30","step":1,"x":70,"y":120,"wires":[["9dd16ca7.add53","5bf3da7f.259854"]]},{"id":"9dd16ca7.add53","ty
pe":"ui_gauge","z":"4898cfcf.60a3f","name":"","group":"c8da5a75.cebb98","order":2,"width":0,"height":0,
"gtype":"gage","title":"Temperatura","label":"units","format":"{{value}}","min":"18","max":"30","colors
":["#00b500","#e6e600","#ca3838"],"seg1":"22","seg2":"25","x":310,"y":120,"wires":[]},{"id":"62c45bc3.9
c0774","type":"ui_switch","z":"4898cfcf.60a3f","name":"","label":"Ligado","group":"c8da5a75.cebb98","or
der":1,"width":0,"height":0,"passthru":true,"decouple":"false","topic":"/sala4a/ligado","style":"","onv
alue":"1","onvalueType":"num","onicon":"","oncolor":"","offvalue":"0","offvalueType":"num","officon":""
,"offcolor":"","x":70,"y":60,"wires":[["5bf3da7f.259854"]]},{"id":"c9f619d.8040ce8","type":"ui_slider",
"z":"4898cfcf.60a3f","name":"","label":"Velocidade","group":"c8da5a75.cebb98","order":4,"width":0,"heig
ht":0,"passthru":true,"topic":"/sala4a/velocidade","min":"1","max":"5","step":1,"x":90,"y":180,"wires":
[["5bf3da7f.259854"]]},{"id":"5bf3da7f.259854","type":"mqtt
out","z":"4898cfcf.60a3f","name":"","topic":"","qos":"","retain":"","broker":"e46e2d92.86ce5","x":290,"
y":60,"wires":[]},{"id":"c8da5a75.cebb98","type":"ui_group","z":"","name":"Sala
4A","tab":"f36bb77.dfdd848","disp":true,"width":"6","collapse":false},{"id":"e46e2d92.86ce5","type":"mq
tt-
broker","z":"","name":"Localhost","broker":"localhost","port":"1883","clientid":"","usetls":false,"comp
atmode":true,"keepalive":"60","cleansession":true,"birthTopic":"","birthQos":"0","birthPayload":"","wil
lTopic":"","willQos":"0","willPayload":""},{"id":"f36bb77.dfdd848","type":"ui_tab","z":"","name":"UERGS
- Campus central","icon":"dashboard"}]
```

APÊNDICE E – SAÍDA DO ESPRUINO

```
 ____                 _
|  __|___ ___ ___ _ _|_|___ ___
|  __|_ -| . |  _| | | |   | . |
|____|___|  _|_| |___|_|_|_|___|
        |_| espruino.com
 1v96.33 (c) 2018 G.Williams
Espruino is Open Source. Our work is supported
only by sales of official boards and donations:
http://espruino.com/Donate
>
WIFI :{
  "ip": "192.168.25.7",
  "netmask": "255.255.255.0",
  "gw": "192.168.25.1",
  "mac": "30:ae:a4:45:04:10"
 }
MQTT: Conectado!
MQTT: "/sala4a/temperatura" = "22"
IR: "1426399515654"
MQTT: "/sala4a/velocidade" = "2"
IR: "14263988368"
MQTT: "/sala4a/ligado" = "1"
IR: "142610243"
```