**UNIVERSIDADE REGIONAL DE BLUMENAU**

**CENTRO DE CIÊNCIAS EXATAS E NATURAIS**

**CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO**

# PROTÓTIPO DE SISTEMA DE AUTOMAÇÃO RESIDENCIAL INTEGRADO COM RASPBERRY PI UTILIZANDO WINDOWS

**PLAMEDI L. LUSEMBO**

**BLUMENAU**
**2017**

**PLAMEDI L. LUSEMBO**

# PROTÓTIPO DE SISTEMA DE AUTOMAÇÃO RESIDENCIAL INTEGRADO COM RASPBERRY PI UTILIZANDO WINDOWS

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Miguel Alexandre Wisintainer, Mestre – Orientador

**BLUMENAU**
**2017**

# PROTÓTIPO DE SISTEMA DE AUTOMAÇÃO RESIDENCIAL INTEGRADO COM RASPBERRY PI UTILIZANDO WINDOWS

Por

**PLAMEDI L. LUSEMBO**

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

| | |
|---|---|
| Presidente: | Prof. Miguel Alexandre Wisintainer, Mestre – Orientador, FURB |
| Membro: | Prof. Francisco Adell Péricas, Mestre – FURB |
| Membro: | Prof. Marcos Rodrigo Momo, Especialista – FURB |

Blumenau, 11 de dezembro de 2017

Dedico este trabalho à minha mãe que nunca mediu os esforços para me oferecer todo apoio necessário nos estudos.

## AGRADECIMENTOS

A Deus pela graça e pelo dom da vida.

À minha mãe Antoinette Lusembo por todo incentivo e estimulo à perseverança.

Às minhas irmãs Nadine, Patricia e Elsa Luzolo, ao meu irmão Hervé Luzolo e ao meu primo Serge Makindu por todo apoio e confiança.

Ao Instituto Bíblico de Blumenau pela assistência proporcionada para o bom andamento dos meus estudos.

Ao meu orientador Miguel Alexandre Wisintainer, por seu suporte, sua disposição e o tempo investido neste trabalho.

A Ronaldo Weingatner, Cláudio Baumgarten, Aurel Rothus, Oceane Aboh, Maicon M. Gerardi e Andrey da Silva pela contribuição para a concretização deste trabalho.

Aos demais amigos que contribuíram de um modo ou de outro para a realização deste trabalho.

Os que com lágrimas semeiam com júbilo ceifarão. Quem sai andando e chorando, enquanto semeia, voltará com júbilo, trazendo os seus feixes.

Salmos 126: 5-6

# RESUMO

A compatibilidade do sistema operacional Windows com as novas gerações do minicomputador Raspberry Pi ampliou as possibilidades de prototipagem para o desenvolvimento na área de IoT. Este trabalho apresenta o desenvolvimento de um protótipo de sistema de automação residencial baseado no Raspberry Pi 3 Model B utilizando o sistema operacional Windows 10 IoT Core. O desenvolvimento requereu a realização de levantamento bibliográfico e pesquisas relacionadas às técnicas e ferramentas utilizadas em desenvolvimento de sistemas de automação residencial usando conceitos da IoT. O sistema desenvolvido, batizado de Winberry, usa sensores para capturar as informações do ambiente residencial e atuadores para executar determinadas ações de controle. Foi construído uma central de controle utilizando como dispositivos principais um módulo relé, um sensor de movimento, um sensor de temperatura e umidade, um mini Cooler, um Buzzer ativo, um motor de passo, LEDs e um Raspberry Pi 3 Model B. A consulta das informações e o controle do ambiente residencial são realizados remotamente por meio de um aplicativo móvel conectado à rede WLAN. Para codificar o software embarcado na central de controle foi utilizado a linguagem de programação C# e para implementar o aplicativo móvel foi utilizado a linguagem de programação Java. No final do desenvolvimento, foram efetuados testes com todos os dispositivos e os resultados obtidos das funcionalidades do protótipo como um todo foram satisfatórios e validados. Por fim, o trabalho desenvolvido atingiu os objetivos propostos, observando que o uso do Raspberry Pi com o Windows para desenvolvimento IoT é de fato viável.

Palavras-chave: Raspberry Pi. Windows. IoT. Internet of things. UWP. Automação residencial.

# ABSTRACT

The compatibility of the Windows operating system with new generations of Raspberry Pi minicomputer has expanded the possibilities of prototyping for developement in the IoT area. This work presents the development of a prototype of home automation system based on the Raspberry Pi 3 Model B using the Windows 10 IoT Core operating system. The development required a bibliographical survey and research related to the techniques and tools used in the development of residential automation systems using IoT concepts. The developed system, called Winberry, uses sensors to capture information from the residential environment and actuators to perform certain control actions. A control center was constructed using as main devices a relay module, a motion sensor, a temperature and humidity sensor, a mini Cooler, an active Buzzer, a stepper motor, LEDs and a Raspberry Pi 3 Model B. The consultation of information and the control of the residential environment are carried out remotely through a mobile application connected to the WLAN network. To codify the software embedded in the control central the C# programming language was used and to implement the mobile application the Java programming language was used. At the end of the development, all devices were tested, and the results obtained from the prototype functionalities were satisfactory and validated. Finally, the developed work reached the proposed goals, noting that the use of Raspberry Pi with Windows for IoT development is in fact feasible.

Key-words: Raspberry Pi. Windows. IoT. Internet of things. UWP. Home automation.

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

CPU – Central Processing Unit

CSI – Camera Serial Interface

DSI – Display Serial Interface

GPIO – General Purpose Input/Output

GPU – Graphics Processing Unit

HDMI – High-Definition Multimedia Interface

$I^2C$ – Inter Integrated Circuit

IDE – Integrated Development Environment

IoT – Internet of Things

IP – Internet Protocol

LED – Light Emitting Diode

PC – Personal Computer

RAM – Random Access Memory

SO – Sistema Operational

UART – Universal Asynchronous Receiver/Transmitter

UML – Unified Modeling Language

USB – Universal Serial Bus

UWP – Universal Windows Platform

WLAN – Wireless Local Area Network

# SUMÁRIO

# 1 INTRODUÇÃO

Nos dias atuais, a visão do mundo moderno tem evoluído consideravelmente devido à influência da tecnologia que tem alcançado freneticamente as mais diversas áreas da vida do ser humano, despertando nele a necessidade de controlar tudo no seu entorno. As necessidades de controlar e automatizar as rotinas e tarefas em residências deram início a um novo domínio de aplicação tecnológico designado por Domótica ou Automação Residencial.

> A Domótica é a automatização e o controle aplicados à residência. Esta automatização e controle se realizam mediante o uso de equipamentos que dispõem de capacidade para se comunicar interativamente entre eles e com capacidade de seguir as instruções de um programa previamente estabelecido pelo usuário da residência e com possibilidades de alterações conforme seus interesses (CEDOM, 2012 apud SAKAGUCHI, 2014, p. 11).

O termo Domótica resulta da junção da palavra latina “Domus” (casa) com “Robótica” (controle automatizado) (FRENZEL, 2013, p. 187). O significado está intrinsecamente associado à integração de serviços e tecnologias em residências, possibilitando a interação com o usuário e viabilizando a supervisão dos equipamentos e dispositivos domiciliares e a execução automática de tarefas domésticas diversas.

A Domótica é uma tecnologia relativamente antiga, pois, segundo Angel (1993, p. 18, tradução nossa), “As primeiras experiências domóticas começaram na década de 80”. No entanto todo seu potencial de automação ainda não foi explorado, pois o mercado da automação residencial está passando por um processo decisivo. Ao longo desse processo, uma quantidade significativa de aplicativos e equipamentos surgirá em uma velocidade maior do que a da própria internet, por conta da padronização mundial de sistema simples, robusto e escalável (BOLZANI, 2004). As técnicas e ferramentas utilizadas para o desenvolvimento desse sistema afetam essencialmente o desempenho do processo de implementação e implantação, e consequentemente, têm um impacto considerável no investimento e na acessibilidade à tecnologia.

A automação residencial atualmente pode contar com o advento da Internet of Things.

> A Internet of Things (IoT) é um conceito e um paradigma que considera a presença generalizada no ambiente de uma variedade de coisas / objetos que através de conexões com fio e sem fio e esquemas de endereçamento exclusivos, são capazes de interagir uns com os outros e cooperar com outras coisas / objetos para criar novos aplicativos / serviços e alcançar objetivos comuns. (VERMESAN; FRIESS, 2013, p. 7, tradução nossa).

A Internet of Things estimulou a inovação na criação de dispositivos de hardware e plataformas de prototipagem eletrônico como micro controladores e minicomputadores de placa única, e também no desenvolvimento de softwares e sistemas operacionais compatíveis, simplificando consideravelmente o desenvolvimento de sistemas de automação residencial.

Diante do cenário acima exposto, este trabalho tem o propósito de apresentar, mediante uma pesquisa acurada e o desenvolvimento de um protótipo, uma implementação de um sistema de automação residencial integrado com o minicomputador Raspberry Pi 3 Model B utilizando o sistema operacional Windows 10 IoT Core.

## 1.1 OBJETIVOS

O objetivo deste trabalho é implementar um protótipo de um sistema de controle e automação residencial via dispositivo móvel, baseado no minicomputador Raspberry Pi 3 Model B com o sistema operacional Windows 10 IoT Core.

Os objetivos específicos são:

a) pesquisar sobre como utilizar o Raspberry Pi 3 Model B com o Windows 10 IoT Core;
b) construir um hardware baseado no Raspberry Pi 3 Model B para servir de central de controle para a comunicação entre o aplicativo gerenciador e os equipamentos a serem controlados;
c) desenvolver o aplicativo gerenciador da automação residencial;
d) desenvolver um software embarcado para rodar no Raspberry Pi 3 Model B como servidor responsável por processar e executar as instruções recebidas do aplicativo gerenciador.

## 1.2 ESTRUTURA

Este trabalho está estruturado em quatro capítulos. O primeiro apresenta a introdução e os objetivos do trabalho. O segundo capítulo referente à fundamentação teórica aborda os conceitos e as ferramentas envolvidos no desenvolvimento de sistemas integrados com dispositivos IoT. O terceiro capítulo contempla o desenvolvimento do protótipo, descrevendo a análise e especificação dos requisitos, a especificação do problema de pesquisa através de diagramas, as técnicas e ferramentas utilizadas, o detalhamento da implementação do protótipo e os resultados obtidos. Por fim, o quarto capítulo apresenta as conclusões, descreve as vantagens do trabalho, e apresenta as sugestões para extensões do trabalho.

# 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os assuntos que fundamentam a pesquisa realizada e o desenvolvimento do protótipo. As próximas seções descrevem os assuntos tais como: IoT, Raspberry Pi, Window IoT, Automação residencial e os trabalhos correlatos a este trabalho.

## 2.1 IOT

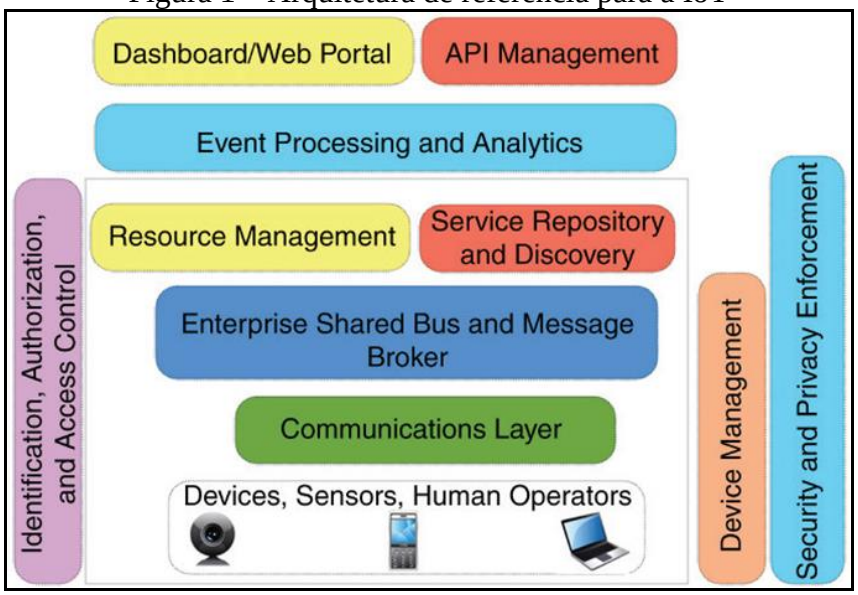
A IoT, sigla para o termo Internet of Things, é um conceito originado no laboratório de Auto-ID do Instituto de Tecnologia de Massachusetts durante o desenvolvimento de um sistema de identificação de produtos por radiofrequência conectado à internet. Segundo Buyya e Dastjerdi (2016, p. 5), “Kevin Ashton é reconhecido por usar o termo ‘Internet of Things’ pela primeira vez durante uma apresentação em 1999 [...]”. O termo se baseia na concepção de um sistema que permita que os objetos do mundo físico estejam conectados à rede e sejam capazes de perceber seu ambiente e outros objetos, e assim coletar, armazenar, processar e transmitir dados e agir de acordo com esses dados. Conforme indica Bell (2016, p. 1, tradução nossa), “A essência da IoT é simplesmente interligar dispositivos que geram e trocam dados de observações, fatos e outros dados, tornando-os disponíveis para qualquer um.”. Basicamente, a IoT visa tornar os dispositivos e equipamentos normalmente não-conectáveis capazes de serem identificados no mundo digital e conectados à rede. A IoT representa uma visão em que a Internet se estende para o mundo real, conectando numerosos objetos todos os dias. Esses objetos do mundo físico podem ser controlados remotamente e podem atuar como pontos de acesso físico a serviços de internet uma vez que são conectados no mundo virtual (BABU; LAXMIGANESH; GOWRISANKAR, 2016, p. 47).

### 2.1.1 Arquitetura para a IoT

Existem diversos dispositivos utilizados de forma ubíqua juntos com os objetos no mundo físico para contribuir para a viabilização da IoT. Lexinnova Technologies (2015, p. 2, tradução nossa) observa que “a IoT se refere ao uso de sensores, atuadores e tecnologia de comunicação embutidos em objetos físicos que permitem que esses objetos sejam rastreados e controlados através de redes como a internet.”. Lexinnova Technologies (2015, p. 2, tradução nossa) ainda constata que “o uso desses dispositivos envolveria três etapas principais: captura de dados usando sensores, recolha de dados através da rede e tomada de decisão com base na análise de dados.”. Portanto, para que as soluções de IoT sejam eficientes, devem contemplar uma arquitetura que possibilite a realização de ações remotas e a utilização de informações em tempo real. A figura 1 ilustra uma arquitetura de referência para a IoT com diferentes

camadas que representam dispositivos de controle e tecnologias específicos envolvidos com a IoT.

Figura 1 – Arquitetura de referência para a IoT



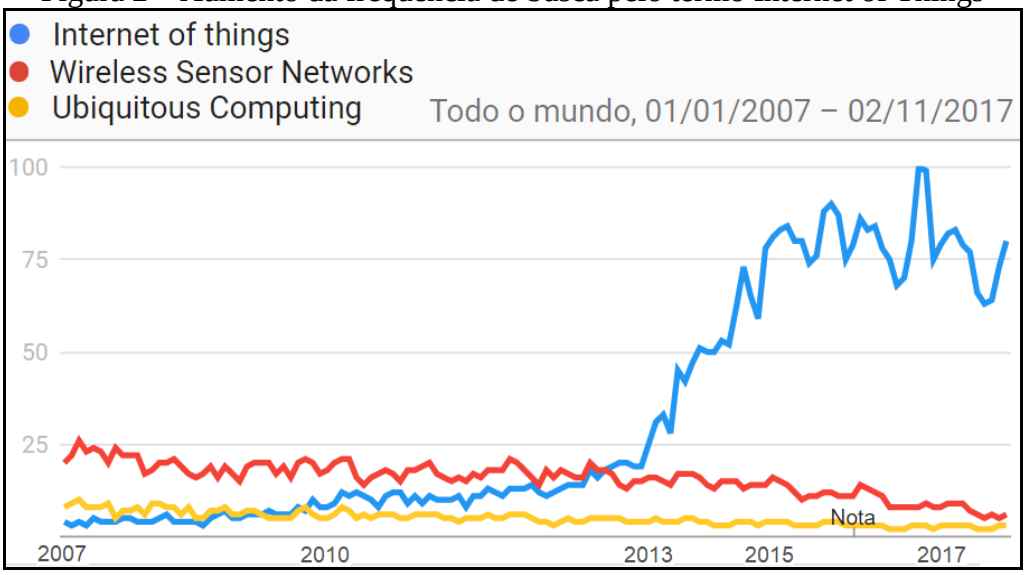

Fonte: Buyya e Dastjerdi (2016).

Nesta arquitetura são representadas as camadas de processamento e análise de eventos, gerenciamento de recursos, serviço de repositório e a agregação de *menssage broker* com serviços ESB (Enterprise Service Bus) representada em cima da camada de comunicação. A arquitetura inclui também as APIs essenciais para a definição e o compartilhamento de serviços de sistema e dispositivos de controle baseados em aplicativos móveis ou aplicações web para acessar e gerenciar as APIs. As camadas de gerenciamento de dispositivos, segurança e privacidade, e de identificação, autorização e controle de acesso são representadas de forma independente nesta arquitetura.

### 2.1.2 Popularidade da IoT

Na última década, a IoT tem se expandido de modo surpreendente. A figura 2 elaborada através da ferramenta Google Trends, apresenta um gráfico com a frequência em que os termos Internet of Things, Wireless Sensor Networks e Ubiquitous Computing foram pesquisados por meio do motor de busca Google Search desde 2007 em todo o mundo. Pode ser constatado o aumento considerável da procura pelo termo Internet of Things a partir do segundo semestre do ano 2013.

Figura 2 – Aumento da frequência de busca pelo termo Internet of Things




Fonte: elaborado pelo autor.

A expansão da IoT promove sua usabilidade, assim como afirma Bell (2016, p. 1, tradução nossa), "De fato, através dos esforços de muitas empresas, inclusive Microsoft, você pode explorar a Internet of Things sem treinamento intensivo ou hardware e software custosos.".

## 2.2 RASPBERRY PI

O Raspberry Pi é um minicomputador de placa única desenvolvido pela Fundação Raspberry Pi. Esse pequeno computador de baixo custo, do tamanho de um cartão de crédito, pode ser conectado a um monitor de computador ou TV, e pode ser utilizado com um teclado padrão e um mouse (KURNIAWAN, 2016). O Raspberry Pi foi concebido com o objetivo principal de promover o aprendizado da programação de computadores nas escolas. Os primeiros modelos do Raspberry Pi foram lançados no ano 2012. O modelo utilizado no desenvolvimento deste trabalho é o Raspberry Pi 3 Model B. No Quadro 1 é apresentado um comparativo entre os diferentes modelos e versões do Raspberry Pi.

Quadro 1 – Comparativo entre os modelos do Raspberry Pi

| Raspberry Pi | SoC | CPU | RAM | Porta USB | Ethernet | Wireless/ Bluetooth | Lançamento | Preço |
|---|---|---|---|---|---|---|---|---|
| Model A+ | BCM2835 | 700Mhz | 512MB | 1 | ✗ | ✗ | 11/2014 | U$ 20 |
| Model B+ | BCM2835 | 700Mhz | 512MB | 4 | ✓ | ✗ | 07/2014 | U$ 25 |
| 2 Model B | BCM2836 | 900Mhz | 1GB | 4 | ✓ | ✗ | 02/2015 | U$ 35 |
| 3 Model B | BCM2837 | 1200Mhz | 1GB | 4 | ✓ | ✓ | 02/2016 | U$ 35 |
| Zero | BCM2835 | 1000Mhz | 512MB | 1 | ✗ | ✗ | 11/2015 | U$ 5 |
| Zero W | BCM2835 | 1000Mhz | 512MB | 1 | ✗ | ✓ | 02/2017 | U$ 10 |

Fonte: adaptado do Raspberry PI Foundation (2017).

### 2.2.1 Especificações técnicas do Raspberry Pi 3 Model B

Para suceder ao Raspberry Pi 2 Model B, a terceira geração do Raspberry Pi denominada Raspberry Pi 3 Model B, foi lançada no 29 de fevereiro de 2016. Pode ser notado através do Quadro 1 que o novo modelo do minicomputador se destaca pelo aumento de performance, contudo, seu preço foi mantido no valor de 35 dólares americanos. Veja as especificações técnicas que possui o Raspberry Pi 3 Model B, apresentas no Quadro 2.

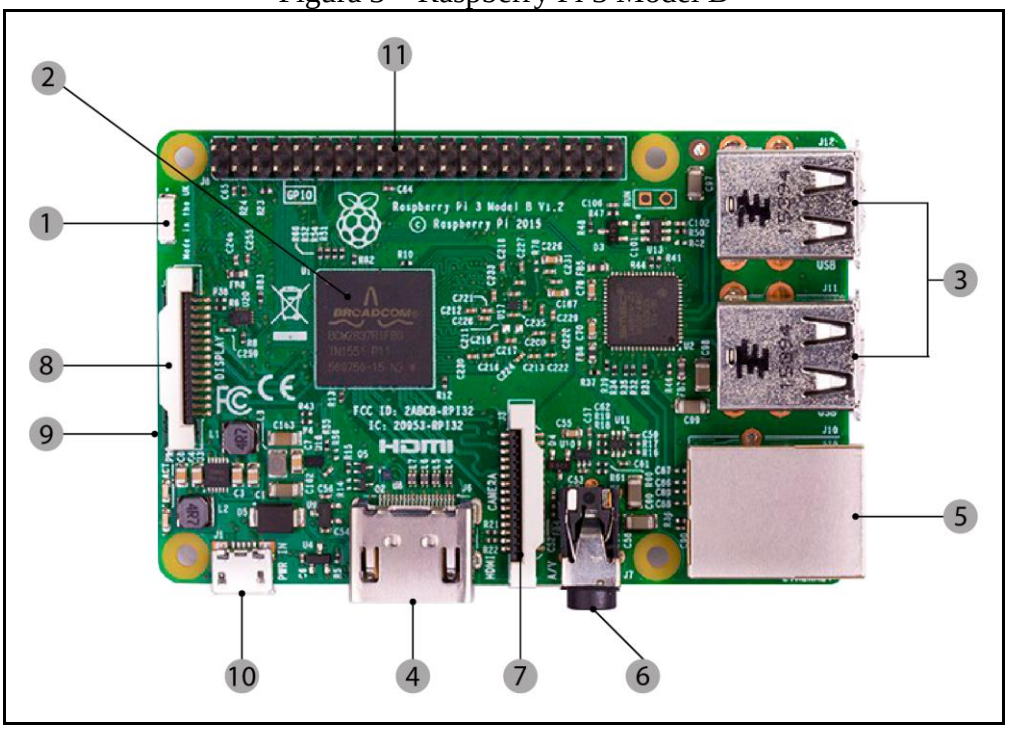
Quadro 2 – Especificações técnicas do Raspberry Pi 3 Model B

| Características | Raspberry Pi 3 Model B |
|---|---|
| SoC | Broadcom BCM2837 |
| CPU | 1.2 GHz 64-bit Quad-core ARMv8 Cortex-A53 |
| GPU | Broadcom VideoCore IV @ 250 MHz<br>OpenGL ES 2.0<br>MPEG-2 e VC-1 (licenciado)<br>1080p30 H.264/MPEG-4 AVC high-profile decodificador e codificador |
| RAM | 1 GB (compartilhado com o GPU) |
| Unidade de armazenamento de dados | Entrada MicroSDHC |
| USB | 4 portas USB 2.0 |
| Conectividade | 10/100 Mbps Ethernet<br>Wi-Fi 802.11n<br>Bluetooth 4.1 |
| Entrada vídeo | Conector de interface de câmera MIPI (CSI) |
| Saída vídeo | HDMI rev 1.3<br>Vídeo Composto RCA Jack<br>Interface de display MIPI (DSI) |
| Entrada áudio | via I²S |
| Saída áudio | Analógico via stéréo Jack<br>Digital via HDMI<br>Composto<br>via I²S |
| Periféricos de baixo nível | 17 × GPIO<br>UART<br>I²C bus<br>SPI bus com 2 chip selects<br>I²S audio<br>+3.3 V<br>+5 V |
| Potência nominal | 800 mA (4 W) |
| Fonte de alimentação | 5 V via MicroUSB ou GPIO header |
| Consumo | 720 mA |
| Dimensões | 85.60 mm × 53.98 mm × 17 mm |
| Peso | 45 g |

Fonte: elaborado pelo autor.

Conforme destaca Kurniawan (2016, p. 4, tradução nossa), “Com base no último processador Broadcom 2837 ARMv8 64bit a nova geração Raspberry Pi 3 Model B é mais rápida e mais potente que seus predecessores.”. Consequentemente, o aumento de desempenho do Raspberry Pi 3 Model B, permite que o minicomputador apresentado na Figura 3, suporte a tecnologia dos componentes e periféricos incorporados na sua placa.

Figura 3 – Raspberry Pi 3 Model B




Fonte: adaptado do Raspberry PI Foundation (2017).

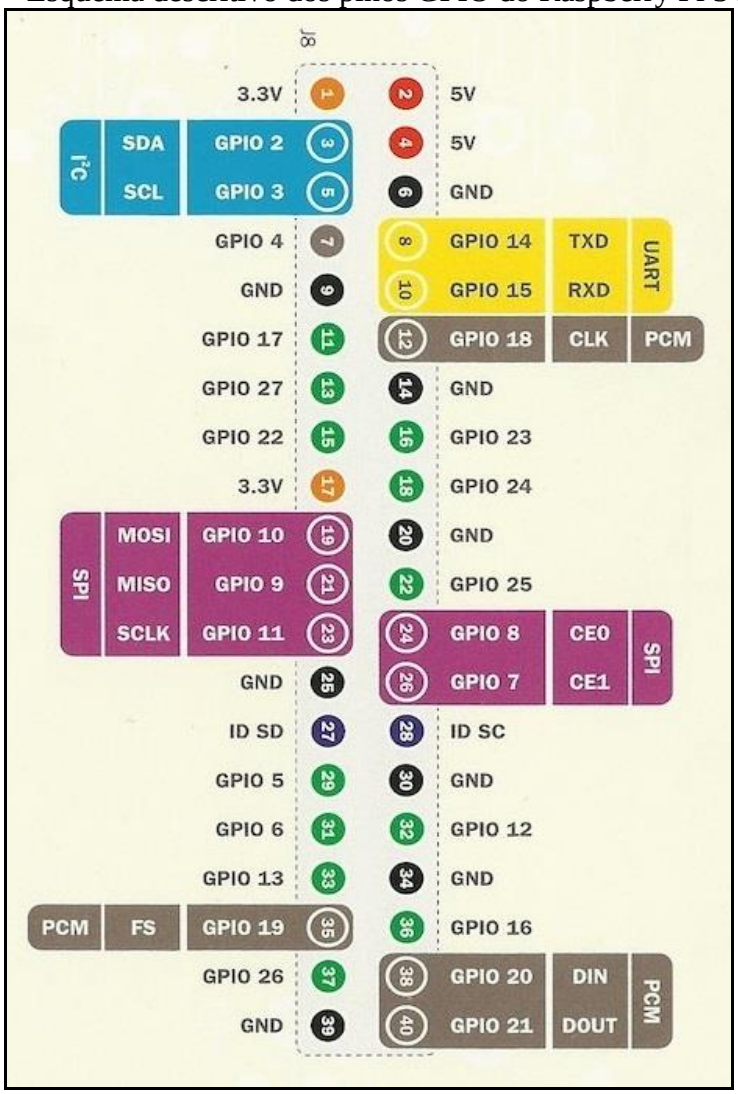
Conforme pode ser observado no Raspberry Pi 3 Model B ilustrado na Figura 3, a integração das novas opções de conectividade: WLAN 802.11n (1) e Bluetooth 4.1 dispensa os adaptadores; o novo processador Broadcom BCM2837 (2), aumentado para 1.2GHz, é capaz de trabalhar em 64 bits, todavia, mantem os quatro núcleos; as 4 portas USB 2.0 (3) disponibilizadas para conectar o teclado, mouse e outros periféricos; a entrada Full HDMI (4) para conectar o minicomputador a uma TV ou um monitor de computador. O cabo pode transmitir vídeo de alta resolução; a entrada Ethernet (5) com velocidade máxima de 100Mbps, que permite conectar o minicomputador a uma rede; a saída de áudio Jack 3.5mm e vídeo composto (6) combinados. O cabo HDMI conectado a um monitor também pode ser utilizado para obter o áudio a partir do Raspberry Pi 3 Model B; a interface CSI (7) para conectar a câmera do Raspberry Pi; a interface DSI (8) para conectar o display táctil do Raspberry Pi; a entrada *push-pull* para cartão MicroSD (9). O Raspberry Pi não possui HD, portanto é necessário o uso de um cartão MicroSD com no mínimo 8GB para instalar o sistema operacional e armazenar dados. A classe de velocidade de gravação sequencial

mínima de cartão MicroSD recomendada para o Raspberry Pi 3 Model B é 10, referente a 10MB/s; a porta MicroUSB 5V (10) para conectar a fonte de alimentação do minicomputador; os pinos GPIO do Raspberry Pi 3 Model B (11).

### 2.2.2 Pinos GPIO do Raspberry Pi 3 Model B

A quantidade de pinos GPIO no Raspberry Pi 3 Model B foi aumentada para 40 pinos machos. Na Figura 4 pode ser observado o esquema descritivo dos pinos GPIO do Raspberry Pi 3 Model B.

Figura 4 – Esquema descritivo dos pinos GPIO do Raspberry Pi 3 Model B




Fonte: adaptado do Cana Kit (2017).

Em referência à Figura 4, os pinos GPIO do Raspberry Pi 3 Model B são organizados de tal forma que os indicados com a cor:

a) verde são 12 pinos GPIO que podem ser programados para fazer a comunicação de entrada e saída de sinais digitais;

b) rosa são 5 pinos SPI (Serial Peripheral Interface) que podem ser utilizados para

fazer uma comunicação serial *Full Duplex* síncrona com periféricos externos e também podem ser programados para a comunicação de entrada e saída de sinais digitais;

c) amarela são 2 pinos que utilizam o protocolo UART para fazer uma comunicação serial *Half* ou *Full Duplex* síncrona com periféricos externos. Também podem ser programados para a comunicação de entrada e saída de sinais digitais;
d) azul são 2 pinos que podem ser programados para a interface $I^2C$ para fazer uma comunicação serial *Half Duplex* síncrona com periféricos externos. Também podem ser programados para a comunicação de entrada e saída de sinais digitais;
e) cinza são 5 pinos PCM (Pulse-Code Modulation) que podem ser utilizados para a saída de áudio digital. Também podem ser programados para a comunicação de entrada e saída de sinais digitais;
f) azul escuro são 2 pinos ID SC reservados para a memória EEPROM (Electrically-Erasable Programmable Read-Only Memory).
g) vermelho são 2 pinos de saída de alimentação com uma tenção de 5V;
h) laranja são 2 pinos também de saída de alimentação, porém possuem uma tenção de 3.3V;
i) preto são 8 pinos de terra. Todos estão conectados eletricamente, portanto não importa qual usar se o Raspberry Pi estiver conectado a uma fonte de tensão.

### 2.2.3 Sistema operacional do Raspberry Pi 3 Model B

Para utilizar o Raspberry Pi é necessário instalar um sistema operacional. O Raspbian é o sistema operacional oficial da Raspberry Foundation suportado pelo Raspberry Pi (RASPBERRY FOUNDATION, 2017). O Raspbian é baseado no Debian, uma distribuição do Linux. Além do Raspbian, alguns sistemas operacionais de terceiros são suportados pelo Raspberry Pi tais como: Ubuntu Mate, Snappy Ubuntu Core, OSMC, Libreelec, Pinet, Risc OS, Weather Station, Windows 10 IoT Core. O Windows 10 IoT Core é o sistema operacional utilizado neste trabalho para ser instalado no Raspberry Pi 3 Model B.

Neste trabalho, o Raspberry Pi 3 Model B é utilizado para funcionar como o principal dispositivo do hardware que constitui a central de controle onde o servidor do sistema de automação é executado para estabelecer a comunicação com o aplicativo gerenciador e controlar os equipamentos da residência.

## 2.3 WINDOWS IOT

A Microsoft Corporation, empresa que desenvolve o sistema operacional Windows, tem trabalhado no desenvolvimento de um conjunto de versões do sistema operacional Windows denotado Windows Embedded, projetado para uso em dispositivos embarcados. No entanto, com o lançamento do sistema operacional Windows 10 foi disponibilizado o Windows IoT, uma versão adaptada para ser utiliza em dispositivos pequenos com ou sem minitor e é executada em processador ARM e x86/x64 (MICROSOFT, 2017). O Windows Embedded foi substituído pelo Windows IoT e a versão inicial da nova plataforma de sistemas operacionais projetados para uso em processadores de dispositivos IoT é então designada por Windows 10 IoT Core.

### 2.3.1 Windows 10 IoT Core

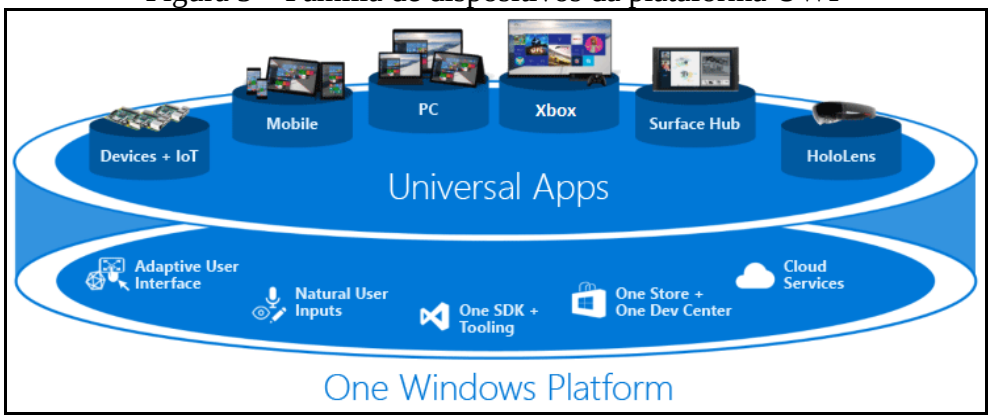
O Windows 10 IoT Core é uma versão compacta do Windows 10 otimizada para rodar em dispositivos embarcados, projetada para abstrair a implementação das camadas de plataforma, hardware e software, simplificando o processo de desenvolvimento de aplicativos para dispositivos IoT (BORYCKI, 2017). Embora o Windows 10 IoT seja uma versão do Windows 10, não possui o mesmo tipo de interface gráfica das versões convencionais do Windows 10, pois é uma versão designada exclusivamente para desenvolvedores.

Entre as características principais do Windows 10 IoT Core são: a interação direta com o hardware, como por exemplo, o acesso aos pinos GPIO; a possibilidade de executar sistemas sem monitor ou display, uma vez que a maioria dos dispositivos IoT não tem suporte a monitor ou display; e a integração da API Universal Windows Platform (UWP).

### 2.3.2 UWP

A UWP é uma plataforma de desenvolvimento de software lançada pela Microsoft junto com o Windows 10. Com esta plataforma, “[...] a Microsoft adotou uma filosofia ‘one Windows’, onde os desenvolvedores podem desenvolver seu código uma vez e executá-lo em qualquer instalação do Windows” (BELL, 2017, p. 22, tradução nossa). A UWP possibilita que um software desenvolvido para a plataforma Windows 10 possa ser executado em diferentes dispositivos do Windows 10, como desktop, mobile, tablet, HoloLens, Xbox, Surface Hub, e dispositivos IoT, sem a necessidade de alterar o código fonte, desde que este seja compatível com o hardware. Veja toda a família de dispositivos da plataforma UWP ilustrada na Figura 5.

Figura 5 – Família de dispositivos da plataforma UWP




Fonte: Microsoft (2017).

Para utilizar a extensão da API UWP em desenvolvimento para Windows 10, inclusive soluções IoT, o aplicativo precisa ser escrito em uma linguagem de programação suportada pela plataforma. As opções de linguagem de programação são: C++, VB.NET, C#, F# e JavaScript. Neste trabalho a linguagem de programação utilizada é C#.

### 2.3.3 Requisitos de hardware para Windows 10 IoT Core

De acordo com Bell (2017, p. 23), entre os requisitos de hardware para executar o Windows 10 IoT Core estão os seguintes:

a) memória (sem display): 256MB de RAM (no mínimo 128MB livre para o SO);
b) memória (com display): 512MB de RAM (no mínimo 128MB livre para o SO);
c) armazenamento: 2GB (pode ser cartão SD, memória não volátil ou disco);
d) processador: 400MHz ou ARM mais rápido ou Intel x86.

Atualmente, o Windows 10 IoT Core é executado no Raspberry Pi 2 e Raspberry Pi 3, inclusive nas placas compatíveis com o MinnowBoard Max e no Arrow DragonBoard 410c (BELL, 2017, p. 23).

## 2.4 AUTOMAÇÃO

Segundo Rosario (2009, p. 23), “Automação é todo processo que realiza tarefas e atividades de forma autônoma ou que auxilia o homem em suas tarefas do dia-a-dia.”. Inicialmente a automação foi aplicada exclusivamente nas indústrias, pois de acordo com as afirmações de Rosario (2009, p. 15), “Com a globalização, as indústrias passaram por grandes transformações, [...] dando origem a um conjunto de técnicas e procedimentos designados de AUTOMAÇÃO.”.

### 2.4.1 Automação residencial

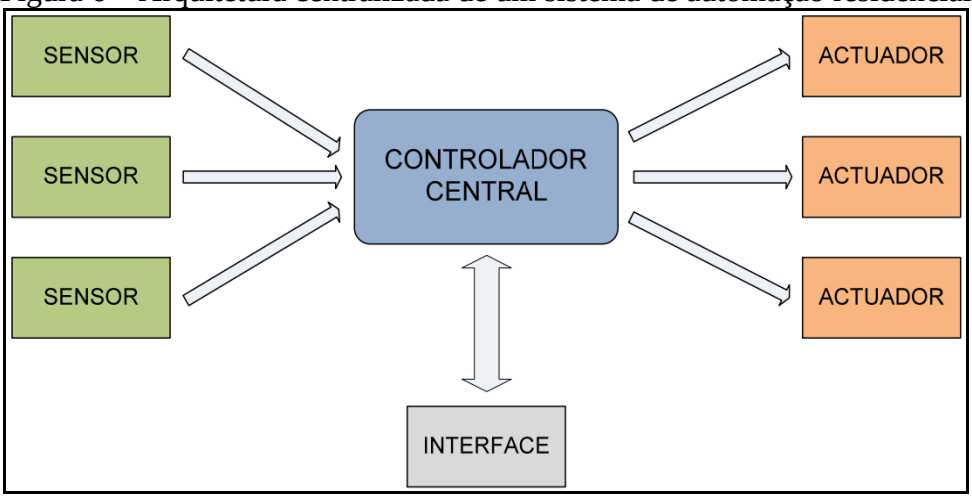
A aplicação da automação em residências é caracterizada por Automação Residencial ou Domótica. Assim como afirma Frenzel (2013, p. 187, grifo nosso), “O ramo da tecnologia que cuida das aplicações de automação em habitações é denominado **Domótica**”.

A automação residencial tem o propósito de simplificar a vida diária das pessoas em suas habitações, possibilitando que as tarefas domésticas mais corriqueiras sejam executadas de forma automática. “Conforto, segurança e economia: a automação residencial aplica tecnologias no ambiente doméstico para atender basicamente a essas três necessidades.” (ROSARIO, 2009, p. 430).

### 2.4.2 Sensores, atuadores, controlador e interface

O sistema de automação residencial é basicamente formado por um conjunto de dispositivos ou equipamentos interconectados por um sistema de comunicação de modo a permitir a troca de informações sobre o estado da residência, possibilitando assim a supervisão e o controle dos equipamentos dessa residência. conforme ilustrado na Figura 6, quatro camadas podem ser destacadas em uma arquitetura de sistema de automação residencial, consistindo em sensores para perceber o estado do ambiente e capturar as informações necessárias, atuadores para executar as ações no ambiente, controlador para controlar os sensores e atuadores, e interface para a supervisão do usuário.

Figura 6 – Arquitetura centralizada de um sistema de automação residencial




Fonte: Ferreira (2008).

Devido à influência de diversas tecnologias que permitem controlar sensores e atuadores, a visão da automação residencial tem sido ampliada significativamente nos últimos anos. A disponibilização vertiginosamente afluente de dispositivos de baixo custo como o

Raspberry Pi, tem contribuído consideravelmente para descomplexificar as possibilidades de desenvolvimento de sistema de automação residencial.

> A automação residencial agora parece estar dando o próximo passo para se tornar amplamente aplicada, e o Raspberry Pi se encaixa perfeitamente neste mundo, fornecendo àqueles que querem personalizar o controle de seus dispositivos com uma ferramenta fácil e barata para alcançá-lo [...]. (DENNIS, 2013, p. 21, tradução nossa).

### 2.4.3 Aplicações da automação residencial

Uma solução eficiente de automação residencial deve permitir o gerenciamento e controle remoto através de dispositivos móveis, smartwatch, PC, entre outros aparelhos. Neste trabalho todas as ações de gerenciamento e controle são realizadas através de um smartphone utilizando a rede WLAN.

Muitas aplicações podem ser implementadas em um sistema de automação residencial. Segundo Miguel, Henning e Turatti (2017, p. 1), “Aplicações possíveis são: Ajuste de temperatura de ar condicionado; Ascendimento de lâmpadas e abajures; Controle de eletrodomésticos; Programar irrigação; Alimentar pets.”. Neste trabalho, o protótipo desenvolvido possibilita as aplicações seguintes:

a) controle da iluminação;
b) abertura e fechamento do portão;
c) ativação e desativação do alarme;
d) disparo do alarme em caso de detecção de movimento;
e) envio automático de e-mail em caso de disparo do alarme;
f) consulta da temperatura e umidade do ambiente residencial.

## 2.5 TRABALHOS CORRELATOS

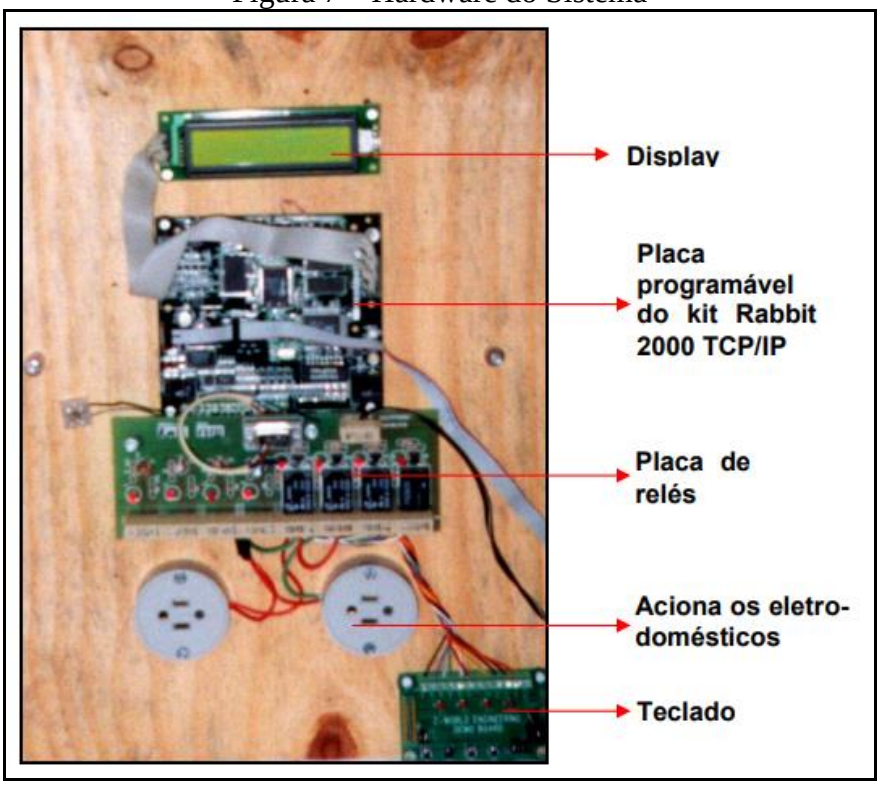
Nesta seção, são apresentados três trabalhos com características similares aos principais objetivos do trabalho desenvolvido. O primeiro é um sistema de automação residencial cujo controle é feito por meio de envio de e-mail (CENSI, 2001), o segundo é uma aplicação integrada com a plataforma Arduino para a automação de residências (BOTKE, 2014), o terceiro é uma solução para automação residencial através de comunicação com o Twitter, independente de um serviço de protocolo de internet fixo (GADOTTI, 2010).

### 2.5.1 Sistema para automação e controle residencial via e-mail

Censi (2001) desenvolveu um sistema de automação residencial que executa as tarefas através de comandos recebidos por e-mail, bem como comandos digitados localmente. Para o

desenvolvimento do sistema proposto, foi utilizado o *kit* de desenvolvimento Rabbit 2000 TCP/IP composto de uma placa programável, uma fonte de alimentação 12V e um cabo serial. Foram utilizadas a linguagem de programação Dynamic C e as bibliotecas de funções disponibilizadas pela Z-Word. Um display de cristal líquido (LCD) e uma placa de interface também foram utilizados para respectivamente visualizar a tarefa a ser executada pelo sistema e o andamento da execução, e transformar o sinal das saídas digitais em ações de abertura ou fechamento dos contatos do relé. A Figura 7 mostra o hardware do sistema.

Figura 7 – Hardware do Sistema

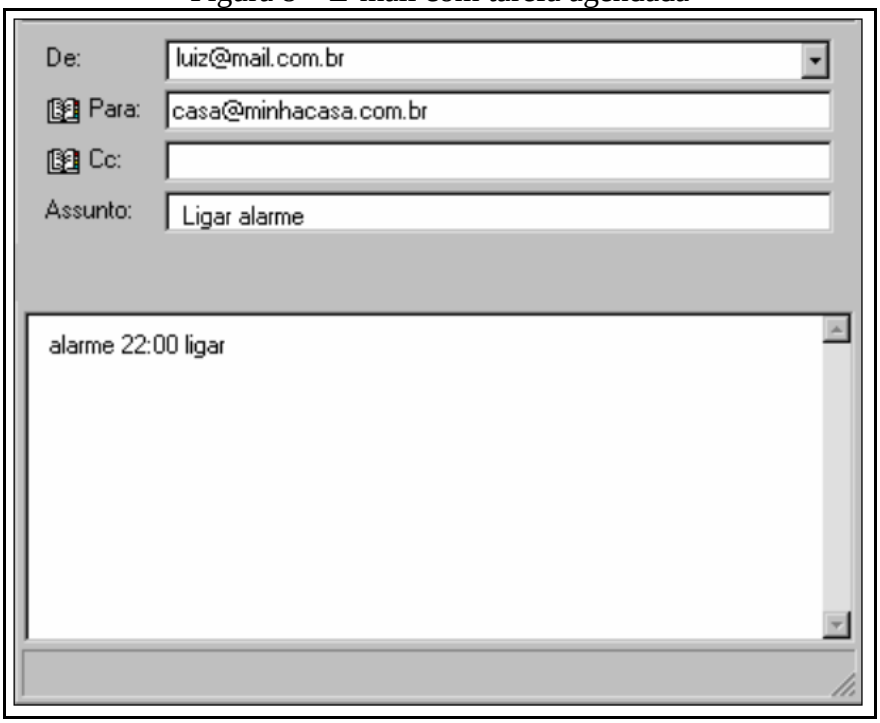



Fonte: Censi (2001).

O sistema executa basicamente as tarefas de ligamento e desligamento de eletrodomésticos, iluminação, ar condicionado e alarme, conforme os comandos são passados por envio de e-mail, o qual é interpretado e validado pelo sistema e em seguida a tarefa solicitada é agendada.

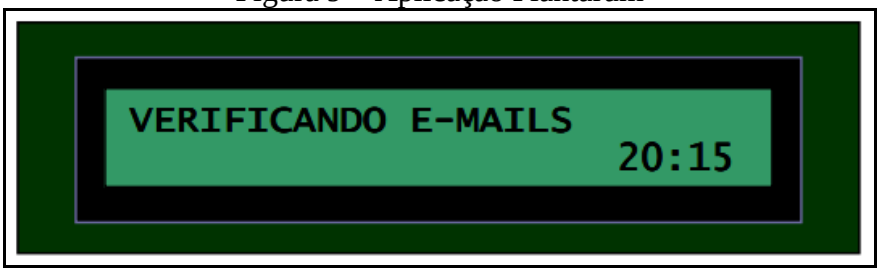
A Figura 8 ilustra um e-mail a ser enviado para o sistema de controle residencial com tarefa a ser agendada. A Figura 9 mostra o sistema de controle verificando e-mails, buscando as mensagens. Pode ser notada uma falta de praticidade no fato do utilizador do sistema precisar memorizar o formato de cada comando que deve ser enviado por e-mail.

Figura 8 – E-mail com tarefa agendada




Fonte: Censi (2001).

Figura 9 – Aplicação Plantarum




Fonte: Censi (2001).

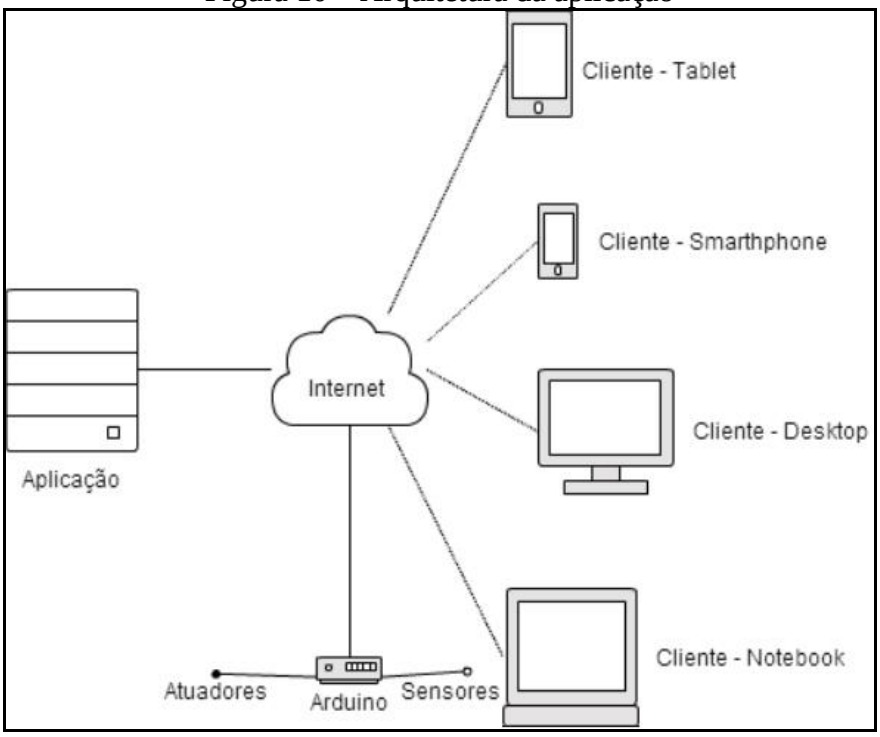
### 2.5.2 Automação de residência através de aplicação integrada com arduino

Botke (2014, p. 18) observa que "O Arduino é uma plataforma de microcontrolador que tem como principal diferencial a sua facilidade de uso e natureza aberta.". Portanto, no seu trabalho foi desenvolvida "[...] uma aplicação [baseada no microcontrolador Arduino] para controle de residências que facilite a execução das atividades cotidianas em uma residência." (BOTKE, 2014, p. 13, grifo do autor).

Para o desenvolvimento da aplicação, foi construído um circuito composto por um controlador, um Arduino Mega 2560, uma *shield* Ethernet, um conjunto de peças eletrônicas como fios e resistores, dois módulos relés, um sensor de contato, e dois sensores infravermelho. O controle do Arduino foi realizado por uma classe denominada sketch, desenvolvida na linguagem de programação C++, utilizando o Ambiente de Desenvolvimento Integrado Arduino IDE.

Para o desenvolvimento da aplicação web, foi utilizado o framework JavaServer Faces 2.1, juntamente com a biblioteca Primefaces 4.0, no Ambiente de Desenvolvimento Integrado NetBeans 7.2. A Figura 10 apresenta a arquitetura da aplicação.

Figura 10 – Arquitetura da aplicação




Fonte: Botke (2001).

Para utilizar a aplicação, o usuário precisa realizar a autenticação, em seguida, aparece um menu apresentando todos os cômodos da residência e uma opção para aceder às ações programáveis. A Figura 11 ilustra a tela de um cômodo da casa, do qual são apresentados todos os equipamentos existentes neste cômodo, com sua situação atual e comandos. Ao aperta algum botão de controle na tela, um comando é enviado ao Arduino para ser executado na residência.

Figura 11 – Tela de um cômodo da casa


Fonte: Botke (2014).

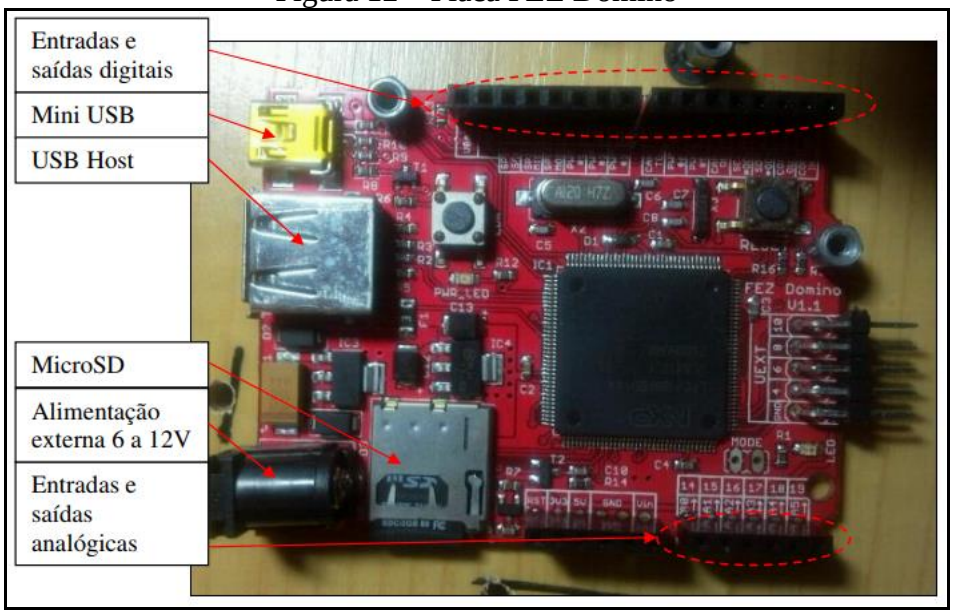
Botke (2014) observa que o "resultado obtido foi a construção de uma aplicação adaptável, ou seja, ela é válida para qualquer configuração de qualquer residência, independentemente da quantidade de cômodos ou equipamentos. ".

### 2.5.3 Sistema para automação e controle residencial via twitter

Gadotti (2010), desenvolveu um protótipo para automação de residência utilizando a rede social Twitter como forma de comunicação para envio de comandos. Para o desenvolvimento do sistema foi utilizada a linguagem de programação C# no Ambiente de Desenvolvimento Integrado Microsoft Visual Studio 2008. O sistema foi compilado no *framework* Micro Framework 4.0. Um *Web Service* foi implementado essencialmente para realizar uma comunicação externa entre o sistema e o Twitter, através da Ethernet com protocolos TCP/IP, para gerenciar o envio e recebimento de mensagens.

Para construir o hardware, foram utilizadas uma placa FEZ Domino, uma placa Ethernet Shield e uma placa controladora CLPIC-628. A Figura 12 mostra a placa FEZ Domino, que contém um cartão MicroSD no qual são armazenados todos os parâmetros e arquivos de configuração do sistema.

Figura 12 – Placa FEZ Domino




Fonte: Gadotti (2010).

Foi utilizado o modo de autenticação oAuth que é um protocolo de autenticação que possibilita que os usuários aprovem o aplicativo para agir em seu nome sem, portanto, compartilhar sua senha (TWITTER, 2017). Foi desenvolvido um programa chamado TwitterGeradorToken para gerar as informações de chave privada e chave pública (*Token* e *TokenSecret*).

Para utilizar o sistema, o usuário precisa executar o programa TwitterGeradorToken ilustrado na Figura 13. É necessário o acesso à internet para que o programa redirecione o usuário à página de autenticação de aplicação do Twitter a fim de liberar o acesso. Após a liberação, o programa irá gerar os *token* que serão usados com a aplicação.

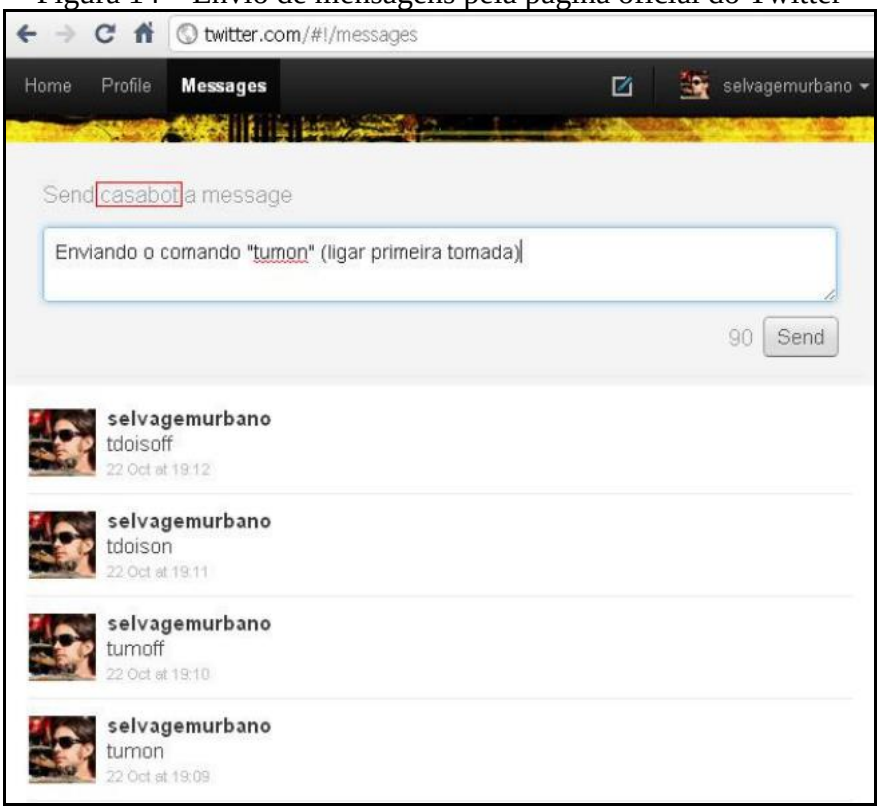
Figura 13 – Tela inicial do sistema TwitterGeradorToken


Fonte: Gadotti (2010).

Para enviar uma mensagem ao Twitter para controle de residência, utiliza-se a página oficial do Twitter (www.twitter.com) e seleciona-se a opção *Message*, para envio de mensagem privada. Além da página oficial do Twitter, há diversas outras possibilidades, entre outras: a aplicação TweetDeck para PC que realiza a comunicação com o Twitter, o cliente TweetCaster para celulares com sistema operacional Android, celular sem acesso à internet, para envio de mensagens através de SMS.

A Figura 14 ilustra o envio de mensagens privadas pela página oficial do Twitter. Pode ser observado que o sistema se torna praticamente independente de uma conexão com a internet, uma vez que se pode utilizar um celular simples para enviar mensagens de controle ao Twitter através de SMS.

Figura 14 – Envio de mensagens pela página oficial do Twitter




Fonte: Gadotti (2010).

# 3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo serão apresentadas as etapas do desenvolvimento do protótipo. Na seção 3.1 são listados os requisitos funcionais e não funcionais. A seção 3.2 apresenta a especificação do protótipo através de diagramas da UML. A seção 3.3 descreve os detalhes da implementação, mostrando as técnicas e ferramentas utilizadas, assim como a operacionalidade do sistema desenvolvido. Por fim, a seção 3.4 apresenta os resultados obtidos com o desenvolvimento deste trabalho.

## 3.1 REQUISITOS

Nesta seção é realizada a análise e especificação dos requisitos do protótipo desenvolvido neste trabalho. No Quadro 3 são descritos os Requisitos Funcionais (RF) e seus respectivos casos de uso (UC), e no Quadro 4 os Requisitos Não Funcionais (RNF).

Quadro 3 – Rastreabilidade dos Requisitos Funcionais (RF) com os Casos de uso (UC)

| Requisitos Funcionais (RF) | Casos de uso (UC) |
|---|---|
| RF01: dispor de um controle de acesso ao aplicativo gerenciador por meio de um mecanismo de autenticação do usuário baseado no nome de usuário e senha | UC01 |
| RF02: permitir que o usuário altere as configurações de conexão via rede | UC02 |
| RF03: possibilitar o controle da iluminação, permitindo ligar e desligar remotamente as luzes da residência | UC03 |
| RF04: possibilitar ligar e desligar o ar condicionado | UC04 |
| RF05: possibilitar a consulta da temperatura e umidade | UC05 |
| RF06: possibilitar a abertura e o fechamento do portão automaticamente | UC06 |
| RF07: possibilitar a ativação e a desativação do sistema de alarme | UC07 |
| RF08: permitir o envio automático de e-mail de notificação sobre o disparo do alarme | |
| RF09: possibilitar a visualização do histórico de disparos do alarme através do aplicativo gerenciador | UC08 |
| RF10: possibilitar a visualização do status dos dispositivos e equipamentos através do aplicativo gerenciador | UC09 |

Fonte: elaborado pelo autor.

Pode ser notado no Quadro 3 que o requisito funcional RF08 referente ao envio automático de e-mail de notificação sobre o disparo do alarme não consta como caso de uso pois a ação é realizada automaticamente pelo servidor.

Quadro 4 – Requisitos Não Funcionais (RNF)

| Requisitos Não Funcionais (RNF) |
|---|
| RNF01: possibilitar que o aplicativo gerenciador seja compatível com o sistema operacional Android |
| RNF02: utilizar ícones intuitivos na interface gráfica do usuário no aplicativo gerenciador |
| RF03: utilizar sockets TCP/IP para a comunicação entre o aplicativo gerenciador e o software embarcado no Raspberry Pi 3 Model B |
| RNF04: utilizar o sistema de gerenciamento de banco de dados SQLite |
| RNF05: ser implementado na linguagem de programação C# utilizando o IDE Visual Studio para o desenvolvimento do software embarcado no Raspberry Pi 3 Model B |
| RNF06: ser implementado na linguagem de programação Java utilizando o IDE Android Studio para o desenvolvimento do aplicativo gerenciador |

Fonte: elaborado pelo autor.

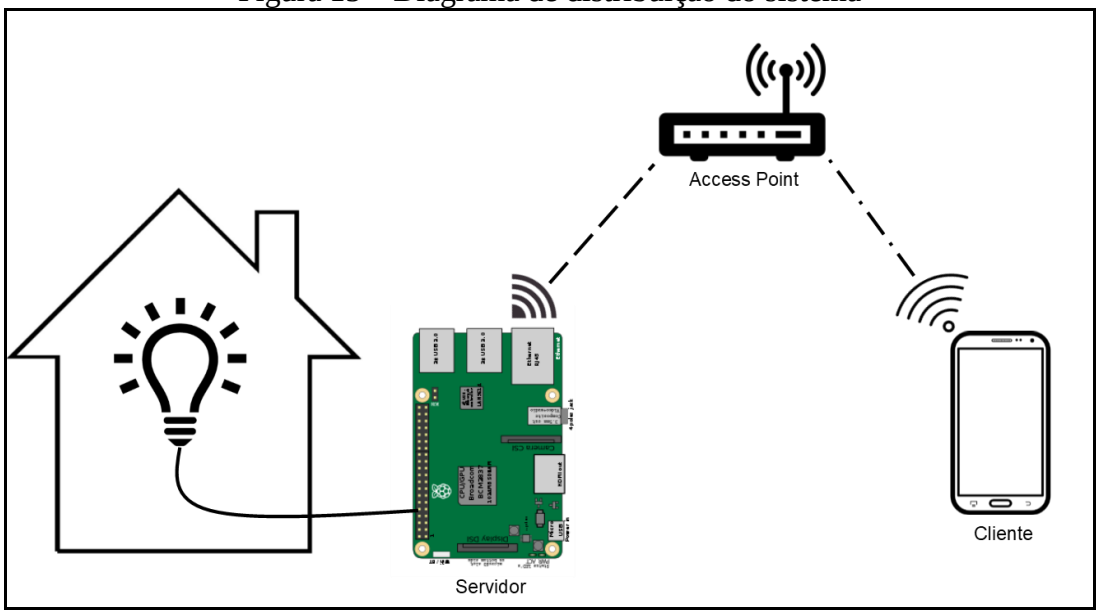
## 3.2 ESPECIFICAÇÃO

Esta seção apresenta a especificação do protótipo através dos diagramas UML que representam logicamente o trabalho desenvolvido. Foram especificados os diagramas de casos de uso e de atividades utilizando a ferramenta Enterprise Architect (EA). Também foi especificado um diagrama de distribuição para representar o sistema desenvolvido.

### 3.2.1 Diagrama de distribuição

Na Figura 15 é apresentada o diagrama de distribuição do sistema desenvolvido identificando os componentes do hardware do sistema e as redes de comunicação. O cenário pode ser descrito da seguinte forma: o smartphone executa o serviço do lado cliente que interage diretamente com o Raspberry Pi que contém o serviço do lado servidor. O cliente envia as requisições e os comandos para o servidor, o servidor recebe e processa os comandos, executa as ações requisitadas e responde ao cliente com o resultado do processamento. A comunicação entre cliente e servidor foi realizada via rede WLAN utilizando os recursos da API de sockets.

Figura 15 – Diagrama de distribuição do sistema

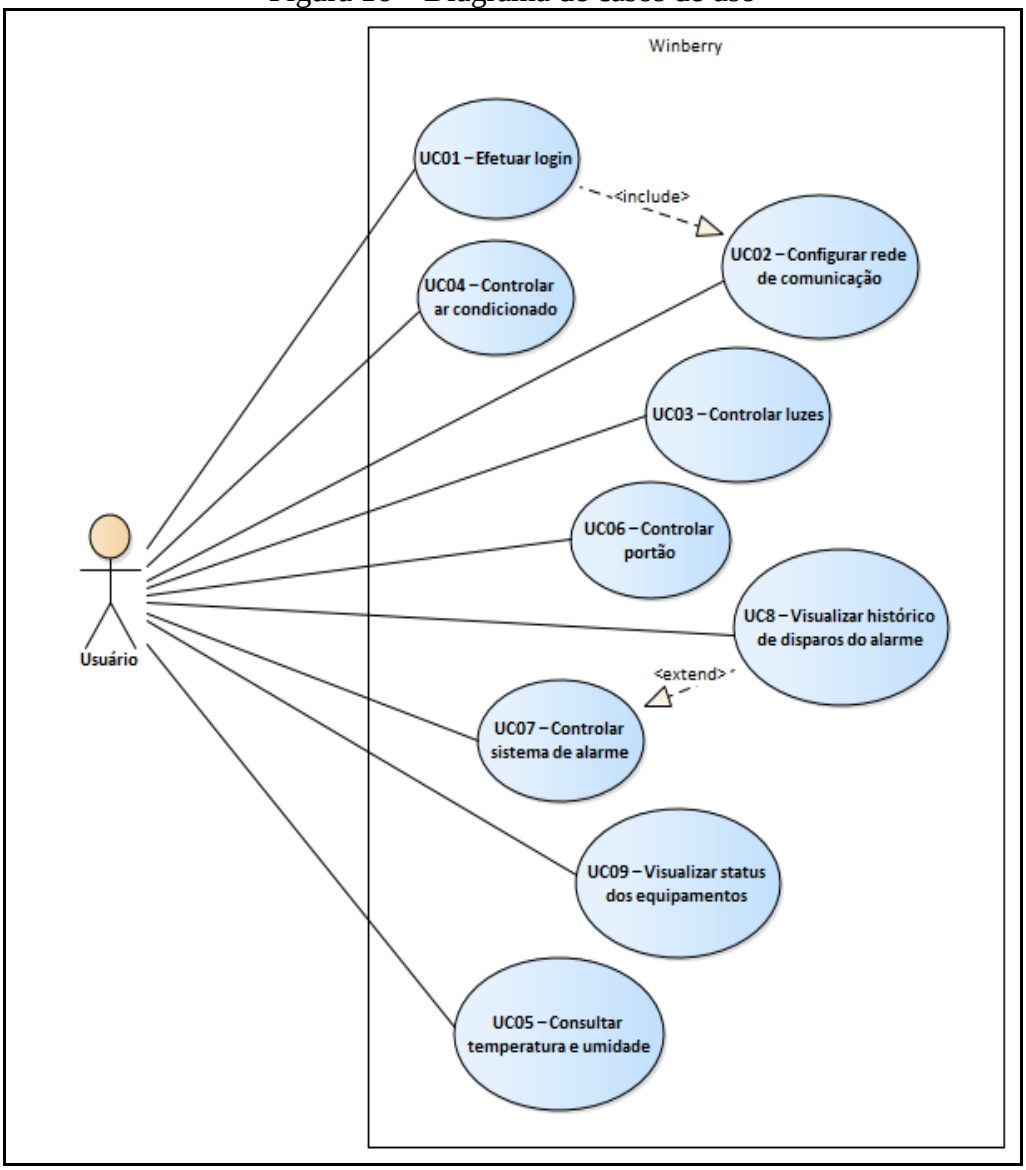



Fonte: elaborado pelo autor.

### 3.2.2 Diagrama de casos de uso

O diagrama de casos de uso representado na Figura 16 descreve as principais funcionalidades do aplicativo gerenciador, mostrando a interação entre o aplicativo e o ator denominado `Usuário`. Conforme pode ser observado neste diagrama, No `UC01 – Efetuar login` é realizado a autenticação do `Usuário` por meio do e-mail e senha. Caso a rede de comunicação não seja configurada corretamente, é necessário fazer devidamente toda a configuração no `UC02 – Configurar rede de comunicação` para ter acesso ao aplicativo gerenciador. No `UC03 – Controlar luzes`, o `Usuário` pode ligar e desligar as luzes da residência. Também é possível ligar e desligar o ar condicionado no `UC04 – Controlar ar condicionado`, consultar a temperatura e a umidade no `UC05 – Consultar temperatura e umidade`, e abrir e fechar o portão no `UC06 – Controlar portão`. Quando o caso de uso `UC07 – Controlar sistema de alarme` é executado para ativar ou desativar o sistema de alarme, o histórico de disparos do alarme também pode ser visualizado de forma praticamente simultânea no `UC8 – Visualizar histórico de disparos do alarme`. No `UC09 – Visualizar o status dos equipamentos` é possível visualizar o estado ou a posição dos equipamentos controlados pelo sistema de automação.

Figura 16 – Diagrama de casos de uso

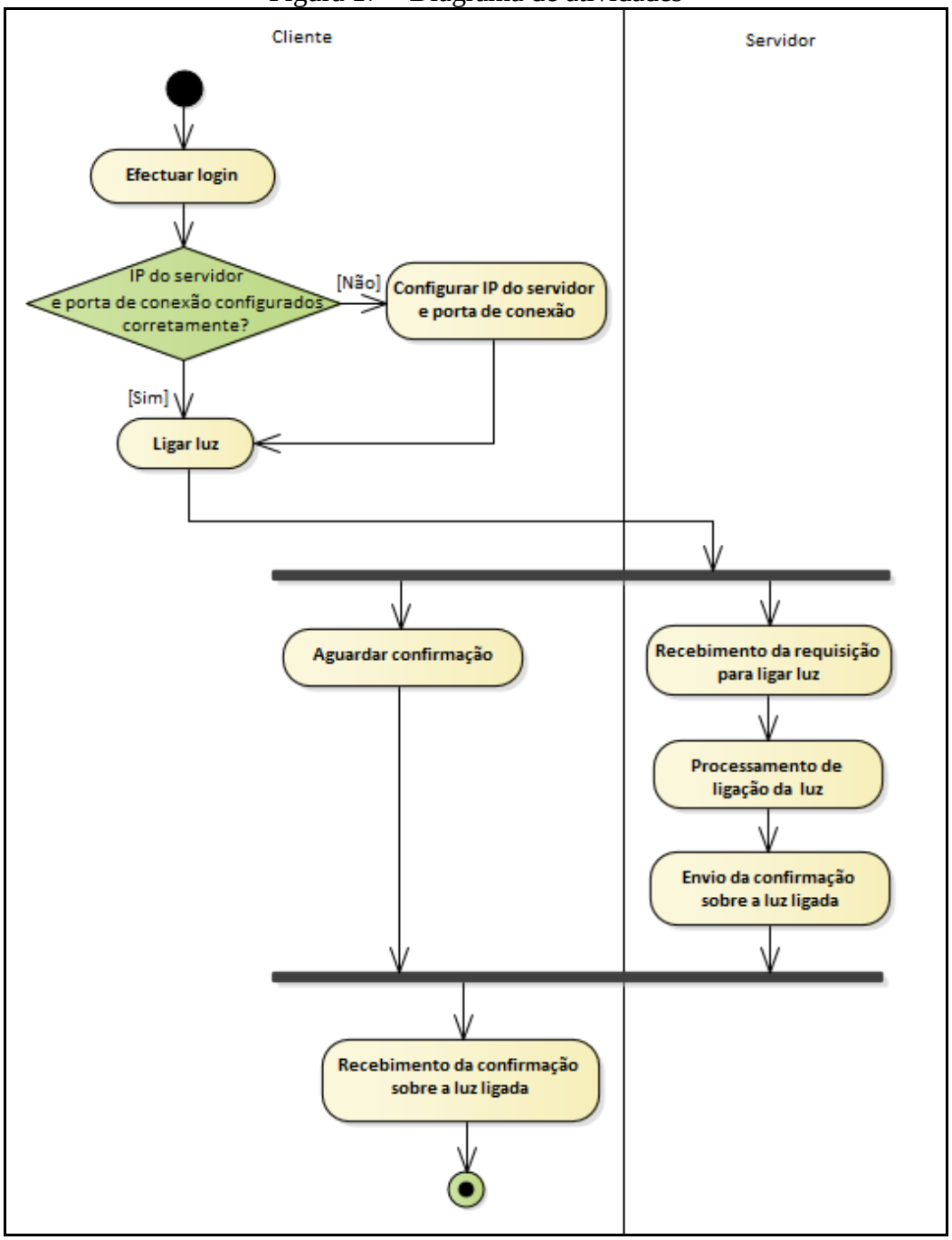



Fonte: elaborado pelo autor.

### 3.2.3 Fluxo de atividades do protótipo

Na Figura 17 é apresentado o diagrama de atividades que especifica o comportamento do protótipo, mostrando o fluxo de execução de uma atividade para outra.

Figura 17 – Diagrama de atividades




Fonte: elaborado pelo autor.

É possível observar na figura 17 que o protótipo utiliza o modelo cliente-servidor. Inicialmente o usuário acessa o sistema por meio de autenticação. Caso as configurações de IP do servidor e da porta de conexão não estejam especificadas corretamente, o usuário tem a opção de fazer essas configurações para poder acessar o sistema. Após o acesso, o usuário é habilitado a solicitar a execução de uma tarefa específica como por exemplo ligar a luz, ao enviar um comando de controle que é recebido pelo servidor que por sua vez processa o comando enquanto o cliente aguarda a conformação do processamento. No final do processamento o servidor retorna uma confirmação sobre a execução da tarefa solicitada.

## 3.3 IMPLEMENTAÇÃO

Nessa seção são mostradas as técnicas e ferramentas utilizadas para o desenvolvimento do protótipo e a operacionalidade da implementação.

### 3.3.1 Técnicas e ferramentas utilizadas

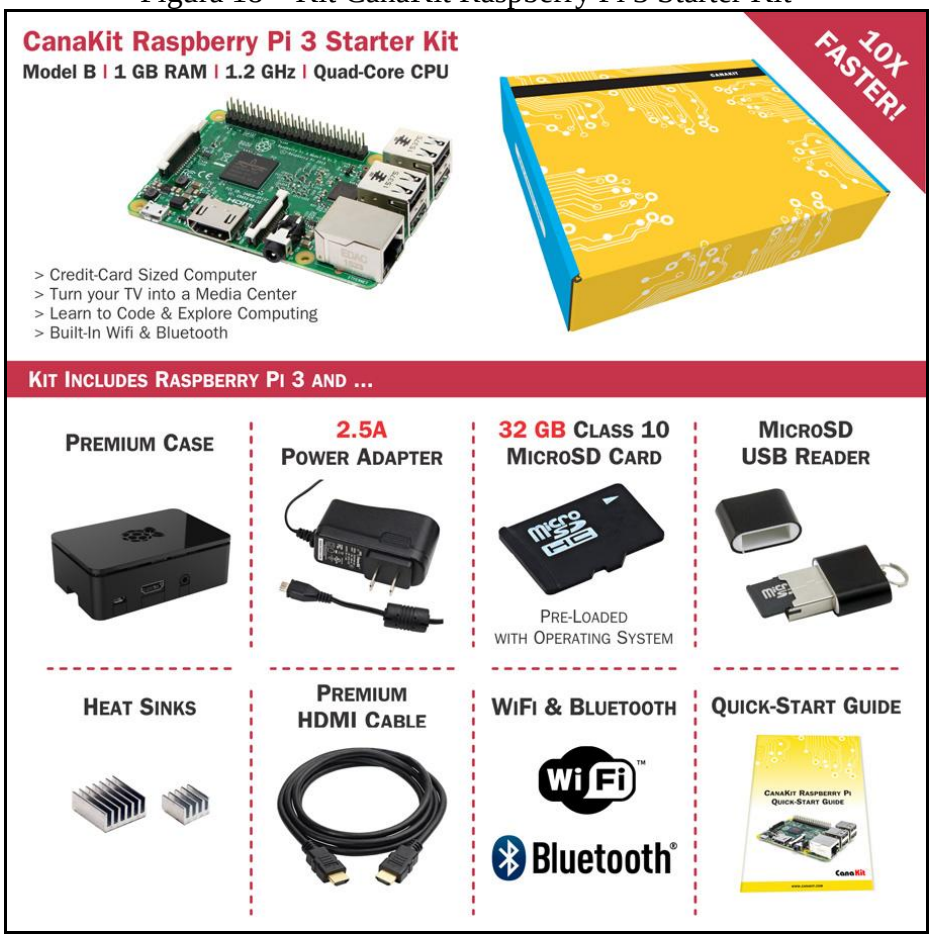
#### 3.3.1.1 Hardware do sistema

A seguir são apresentados os componentes que constituem o hardware do sistema e a construção da central de controle baseada no Raspberry Pi 3 Model B.

##### 3.3.1.1.1 Kit de desenvolvimento CanaKit Raspberry Pi 3 Starter Kit

Para o desenvolvimento do protótipo foi utilizado o kit de desenvolvimento CanaKit Raspberry Pi 3 Starter Kit. Este *starter kit* é um conjunto de peças especialmente projetado para incluir todos os componentes necessários para o desenvolvimento iniciante com o Raspberry Pi 3. A Figura 18 apresentada os componentes incluídos no kit CanaKit Raspberry Pi 3 Starter Kit.

Figura 18 – Kit CanaKit Raspberry Pi 3 Starter Kit




Fonte: CanaKit (2017).

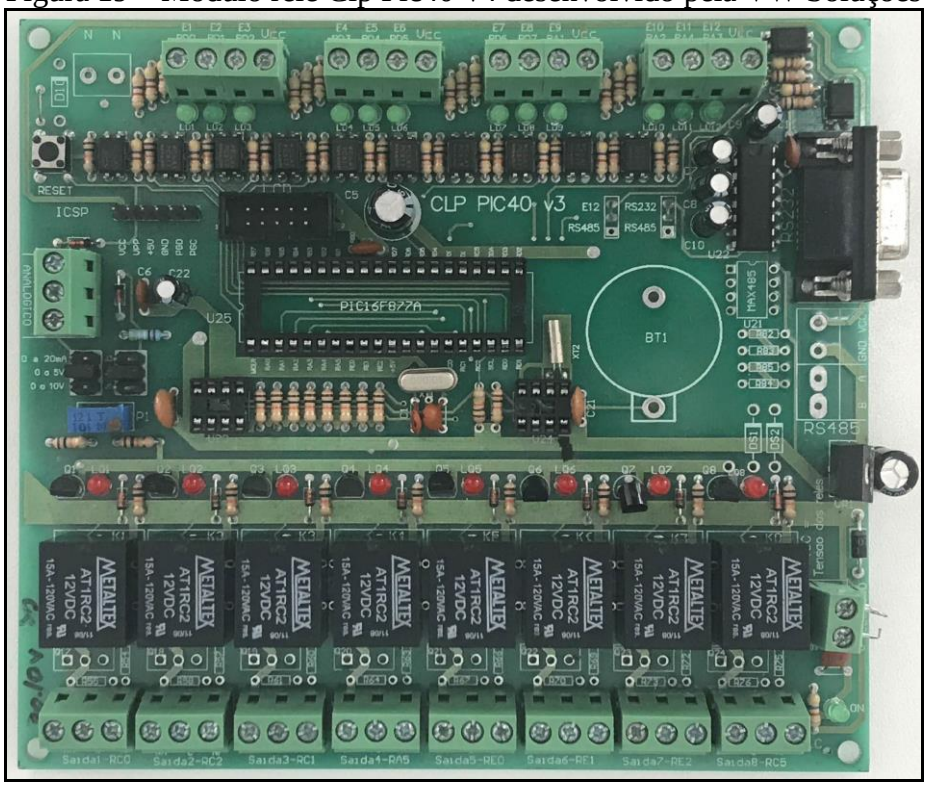
Com base na Figura 18 é possível observar que o kit CanaKit Raspberry Pi 3 Starter Kit inclui essencialmente:

a) 1 Raspberry Pi 3 Model B;
b) 1 *case* em plástico para o Raspberry Pi 3 Model B;
c) 1 fonte de alimentação MicroUSB de 2.5A especialmente projetado e testado para o Raspberry Pi 3;
d) 1 cartão MicroSD Class 10 de 32GB;
e) 1 leitor USB de cartão MicroSD;
f) 2 dissipadores de calor em alumínio;
g) 1 cabo HDMI;
h) 1 prospecto de referência rápida para os pinos GPIO do Raspberry Pi 3 Model B ilustrado na Figura 4;
i) 1 manual de instruções.

#### 3.3.1.1.2 Módulo Relé Clp Pic40-v4

Neste trabalho foi utilizado o módulo relé Clp Pic40-v4 projetado pela VW Soluções. Este módulo foi desenvolvido com base no microcontrolador PIC16F887 que controla todas as funções da placa Clp Pic40, como as saídas (Relês, Transistor ou Triac), comunicação Serial (RS232 ou RS485), barramento $I^2C$, entradas digitais (E1 a E12), entrada analógica, entre outras funções. Através das saídas a Relês, é possível ligar/desligar dispositivos conectados à rede elétrica. No desenvolvimento do protótipo, o módulo relé Clp Pic40-v4 foi necessário principalmente para a utilização das saídas do tipo contato de relés para acionar os LEDs e o mini Cooler controlados pelo Raspberry Pi 3 Model B. Para a alimentação da placa Clp Pic40 foi usada uma fonte de 12V. Os reles de saída recebem a mesma tensão de alimentação da placa em sua bobina, quando são acionados. A Figura 19 exibe o Módulo Relé Clp Pic40-v4.

Figura 19 – Módulo relé Clp Pic40-v4 desenvolvido pela VW Soluções

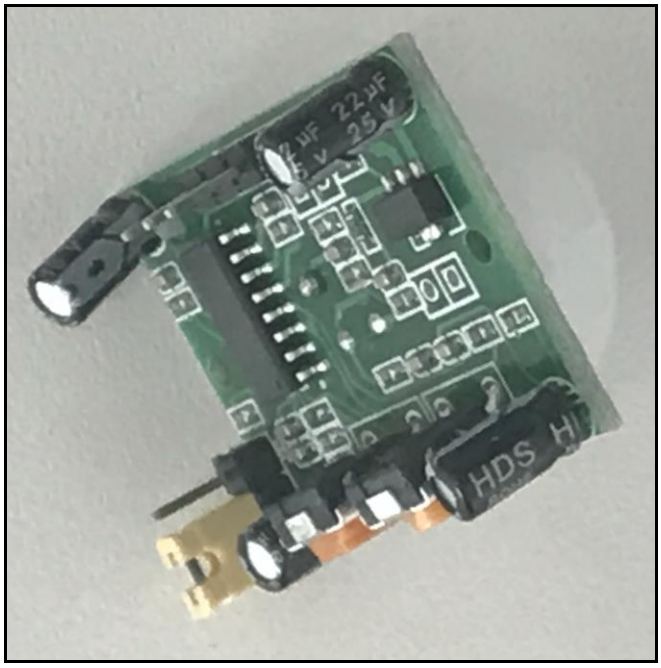



Fonte: digitalizado pelo autor.

3.3.1.1.3 Sensores, atuadores e componentes da estrutura elétrica do protótipo

Foi utilizado o Módulo sensor de movimento PIR - HC-SR501. Neste módulo automático de controle é incorporado um sensor piroelétrico PIR (Passive Infrared) capaz de detectar movimentos por causa da variação de luz infravermelha emitida pela radiação do corpo humano e de animais dentro da área de detecção de até 7 metros e o ângulo do cone de aproximadamente 120°. Como ilustrado na Figura 20, o módulo possui na parte de trás 3 pinos: VCC, OUTPUT e GND. Para utilizar o sensor no desenvolvimento do protótipo, o módulo foi alimentado com uma tensão de 5V no pino VCC e uma tensão de 0V junto ao pino GND. No momento em que o sensor detecta algum movimento, o módulo emite um sinal de 3.3V através do pino OUTPUT. O módulo possui também 2 potenciômetros configuráveis, dos quais um determina o tempo em que o módulo continuará emitindo o sinal através do pino de saída quando o sensor detecta algum movimento, ao passo que o outro potenciômetro serve para o ajuste de sensibilidade, isto é, a distância máxima de alcance para a detecção de movimentos.

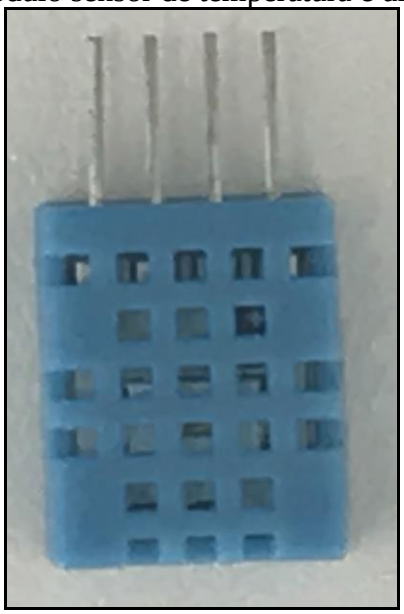
Figura 20 – Módulo sensor de movimento PIR - HC-SR501




Fonte: elaborado pelo autor.

Também foi utilizado o Módulo sensor de temperatura e umidade DHT11. Este módulo inclui um sensor termistor do tipo NTC para medir a temperatura nas escalas de 0 a 50°C e o sensor do tipo HR202 para medir a umidade relativa do ar nas faixas de 20 a 90%. Ambos os sensores são conectados a um minicontrolador de 8-bits. Como pode ser observado na Figura 21, o Módulo sensor DHT11 possui 4 pinos: VCC, OUTPUT, N/A e GND. Neste trabalho foi necessário a utilização apenas dos pinos VCC, OUTPUT e GND. O sensor pode operar com uma alimentação entre 3 e 5.5V. Na Figura 21 veja o Módulo sensor de temperatura e umidade DHT11.

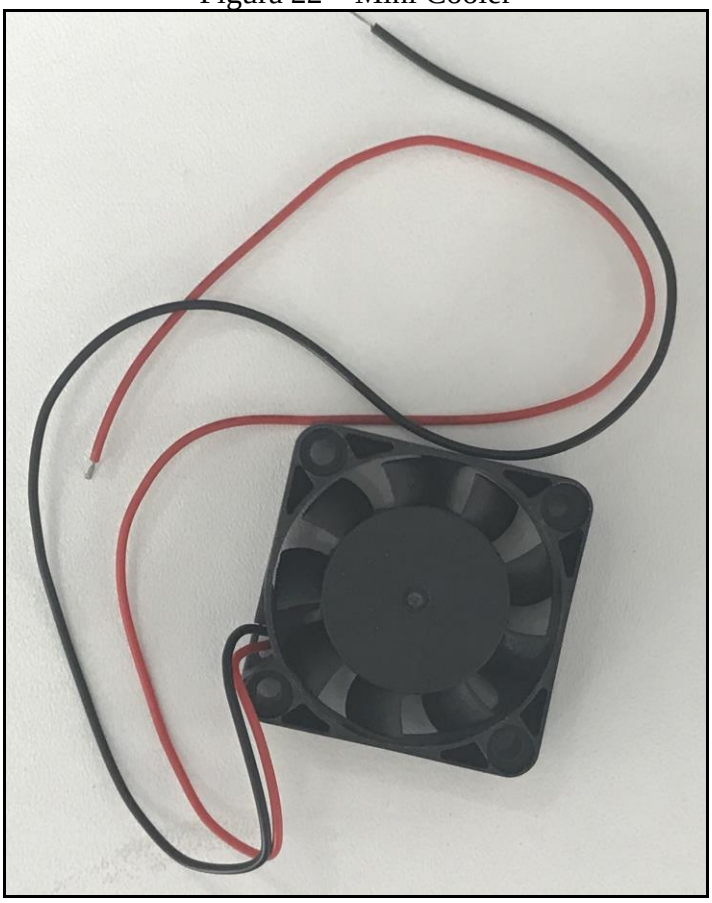
Figura 21 – Módulo sensor de temperatura e umidade DHT11



Fonte: elaborado pelo autor.

Por causa da necessidade de simular o ar condicionado no protótipo desenvolvido, foi utilizado um mini Cooler de 5V que possui 2 fios para alimentação (positivo e negativo). Segue a ilustração na Figura 22 que apresenta o mini Cooler utilizado no hardware do sistema.

Figura 22 – Mini Cooler



Fonte: elaborado pelo autor.

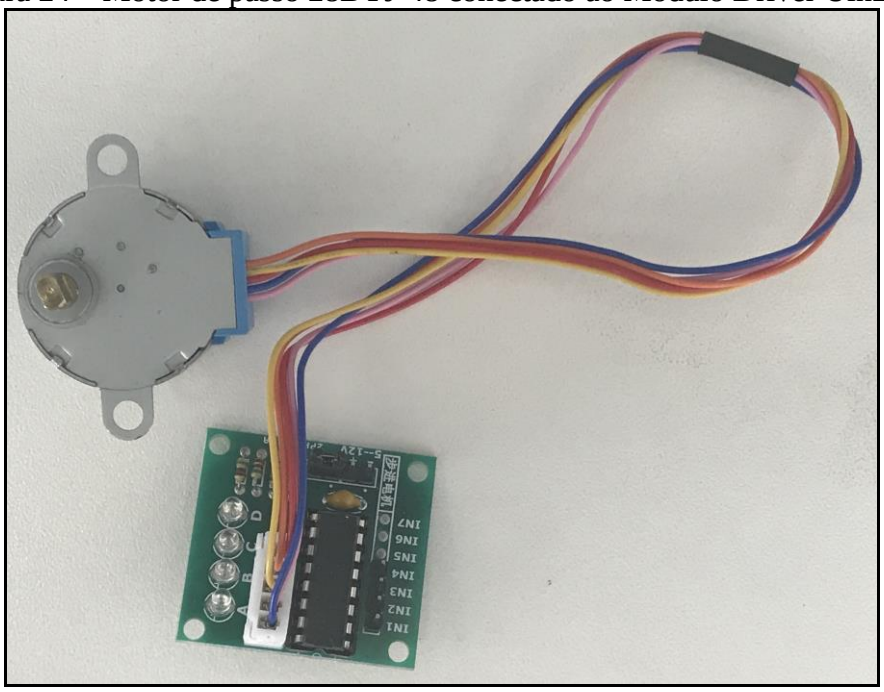
Com a finalidade de emitir *beeps* sonoros como forma de alarme no momento em que o sensor HC-SR501 detecta algum movimento, foi utilizado no hardware do sistema um Buzzer Ativo Bip Contínuo – PCI 12mm. Este Buzzer emite sinais sonoros em frequência única quando o módulo é ativado com uma alimentação de 5V. A Figura 23 apresenta o Buzzer Ativo Bip Contínuo - PCI 12mm.

Figura 23 – Buzzer Ativo Bip Contínuo - PCI 12mm

Fonte: elaborado pelo autor.

Para simular um portão eletrônico no desenvolvimento do protótipo, foi utilizado o Motor de passo 28BYJ-48 junto ao Módulo Driver Uln2003. O motor possui um torque de 34.3mN.m, uma tensão de alimentação de 5V e uma redução de 1/64 que possibilita dar uma volta completa com 4096 passos. O Módulo Driver Uln2003, módulo de controle do motor, usa uma interface com 4 pinos. Estes pinos são utilizados neste trabalho para conectar os pinos GPIO do Raspberry Pi 3 Model B. Veja na Figura 24 o Motor de passo 28BYJ-48 conectado ao Módulo Driver Uln2003.

Figura 24 – Motor de passo 28BYJ-48 conectado ao Módulo Driver Uln2003

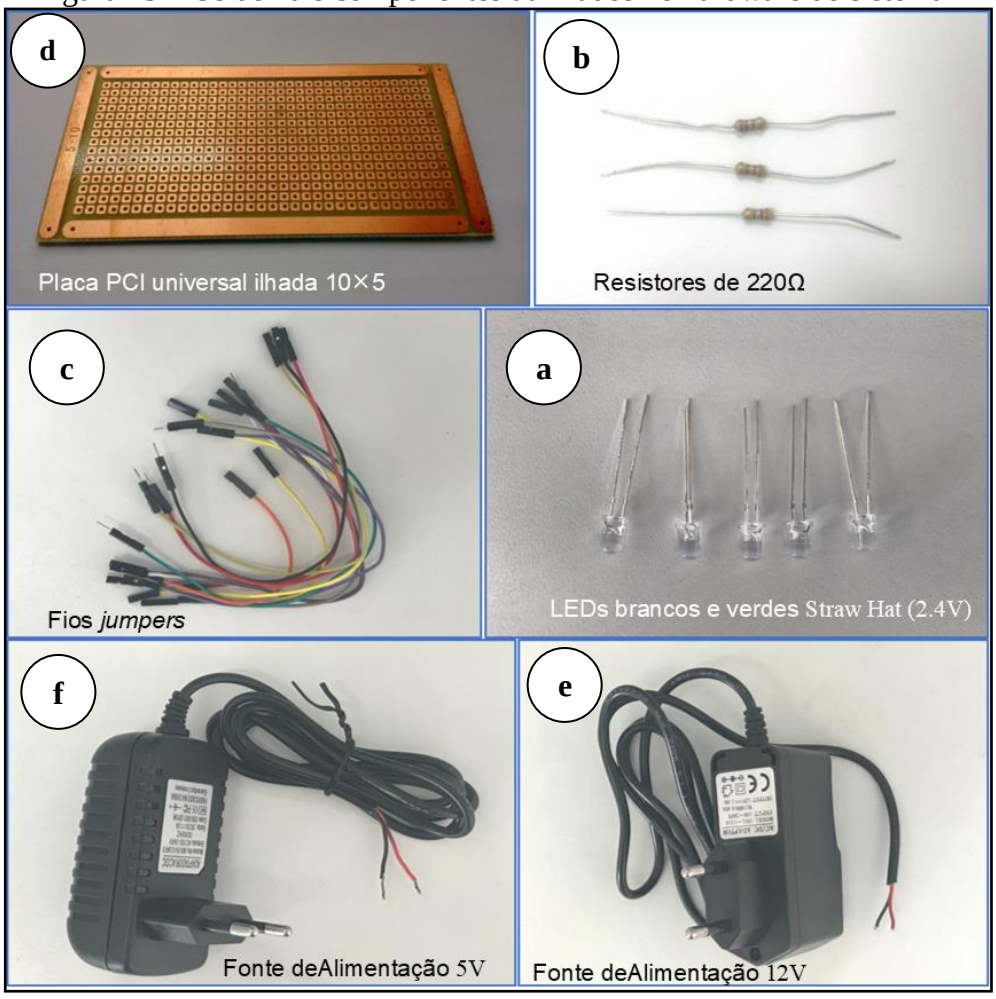



Fonte: elaborado pelo autor.

Além dos componentes acima descritos, mais componentes foram utilizados no desenvolvimento do protótipo para o hardware do sistema. De acordo com a Figura 25, segue os demais componentes utilizados no hardware do sistema para a estrutura elétrica do protótipo:

a) LEDs Straw Hat (2.4V) brancos e verde para simular a iluminação dos cómodos da residência e do jardim;
b) resistores de 220Ω para ligar os LEDs em 5V;
c) fios *jumpers* para interligar os componentes;
d) placa PCI universal ilhada 10×5 para servir como matriz de contatos para conectar os dispositivos e componentes por meio de fios *jumpers* de forma organizada;
e) fonte de alimentação de 12V para alimentar o Módulo Relé Clp Pic40-v4;
f) fonte de alimentação de 5V para alimentar, através da placa PCI universal ilhada, os relés do Módulo Relé Clp Pic40-v4, o Módulo sensor PIR - HC-SR501, o Módulo sensor DHT11, o Buzzer e o Motor de passo.

Figura 25 – Os demais componentes utilizados no hardware do sistema



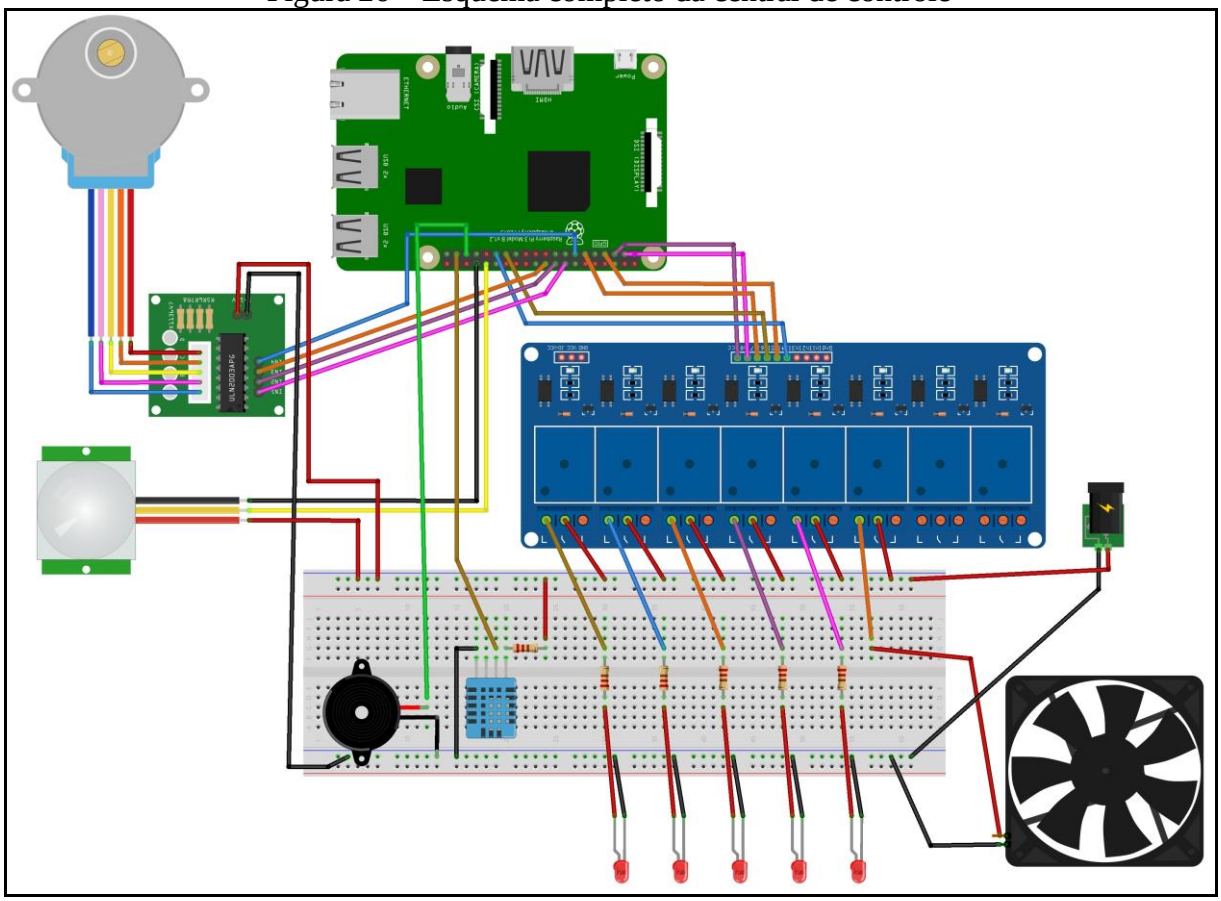

Fonte: elaborado pelo autor.

3.3.1.1.4 Construção da central de controle

A central de controle é composta pelo Raspberry Pì 3 Model B, o Módulo relé Clp Pic40-v4, a placa PCI universal ilhada alimentada por uma fonte de 5V e os demais componentes interligados. A conexão entre o Raspberry Pi 3 Model B e o Módulo relé Clp Pic40-v4 foi feita através da ligação dos pinos GPIO do Raspberry Pi com os pinos do Microcontrolador PIC16F887 do Módulo relé por meio de junpers para controlar as saídas a relés.

Os LEDs e o mini Cooler foram ligados às saídas a relés do Módulo relé por meio da placa PCI universal ilhada com os resistores. O Módulo sensor de movimento PIR - HC-SR501, o Módulo sensor de temperatura e umidade DHT11, o Buzzer Ativo Bip Contínuo, e o Motor de passo 28BYJ-48 foram conectados cada um diretamente com os pinos GPIO do Raspberry Pi e alimentados através da placa PCI universal ilhada. Veja na figura 26 o esquema completo da central de controle baseada nas ligações entre o Raspberry Pi 3 Model B e os demais dispositivos do hardware do sistema. No esquema a placa PCI universal ilhada foi substituída pelo *protoboard* para possibilitar uma melhor visibilidade. O esquema foi montado utilizando a ferramenta Fritzing.

Figura 26 – Esquema completo da central de controle



Fonte: elaborado pelo autor.

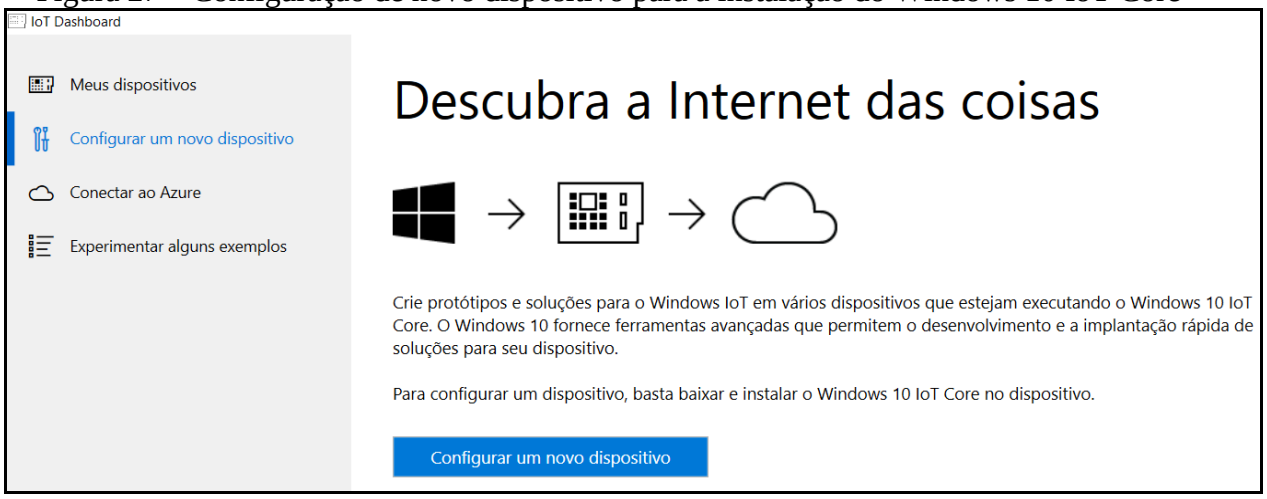
### 3.3.1.2 Desenvolvimento do software

Para o desenvolvimento do software foi necessário realizar a instalação do Windows 10 IoT Core no Raspberry Pi 3 Model B.

### 3.3.1.3 Instalação do Windows 10 IoT Core no Raspberry Pi 3 Model B

A instalação do Windows 10 IoT Core no Raspberry Pi 3 Model B foi realizada utilizando o método de gravação da imagem ISO do sistema operacional no cartão MicroSD por meio de um computador. A maneira utilizada para gravar a imagem ISO no MicroSD foi usando o Windows IoT Core Dashboard que é o aplicativo oficial da Microsoft para o controle de dispositivos compatíveis com o Windows IoT. Veja na Figura 27 a configuração de novo dispositivo para a instalação do Windows 10 IoT Core.

Figura 27 – Configuração de novo dispositivo para a instalação do Windows 10 IoT Core

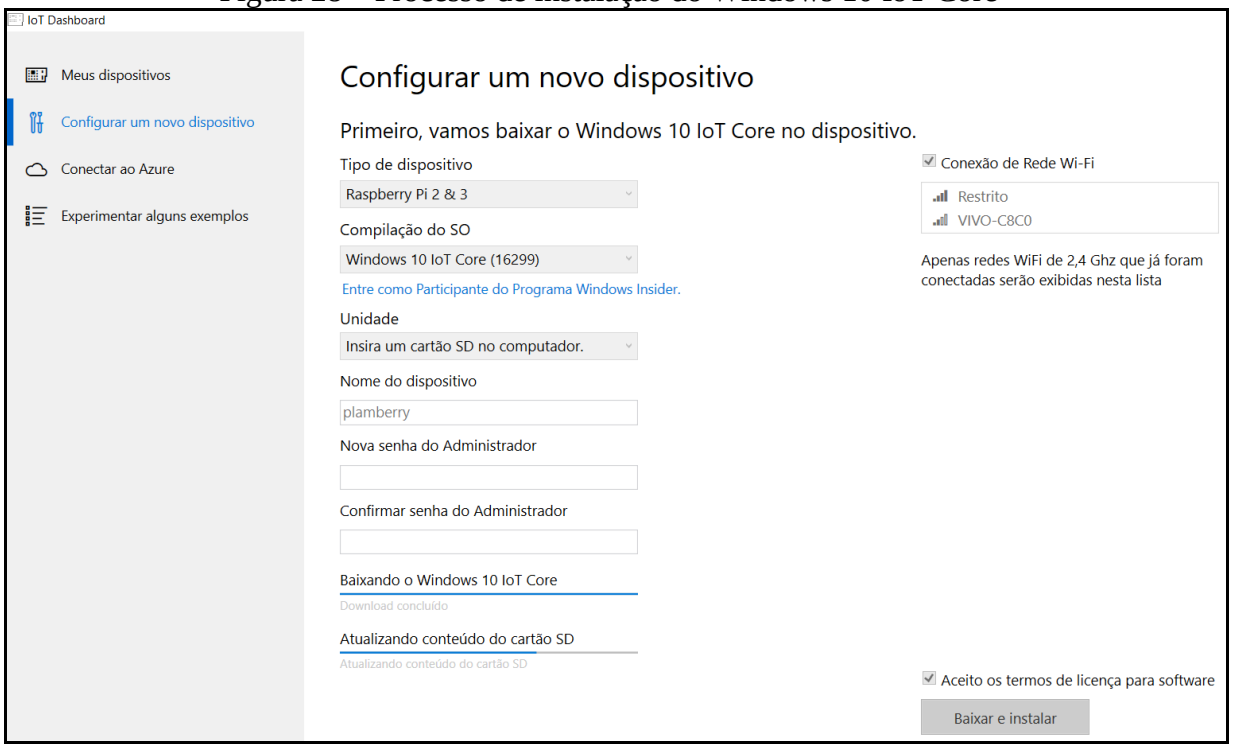



Fonte: elaborado pelo autor.

A instalação do Windows 10 IoT Core com a assistência do Windows IoT Core Dashboard foi realizada seguindo estes passos:

a) ir em `Configurar um novo dispositivo`, clicar no botão `Configurar um novo dispositivo` para configurar um novo dispositivo como é demonstrado na Figura 27;
b) selecionar o tipo de dispositivo, o sistema operacional, o cartão SD e aceitar o termo de licença, em seguida clicar em `Baixar e instalar` e aguardar a finalização da instalação conforme a figura 28.

Figura 28 – Processo de instalação do Windows 10 IoT Core

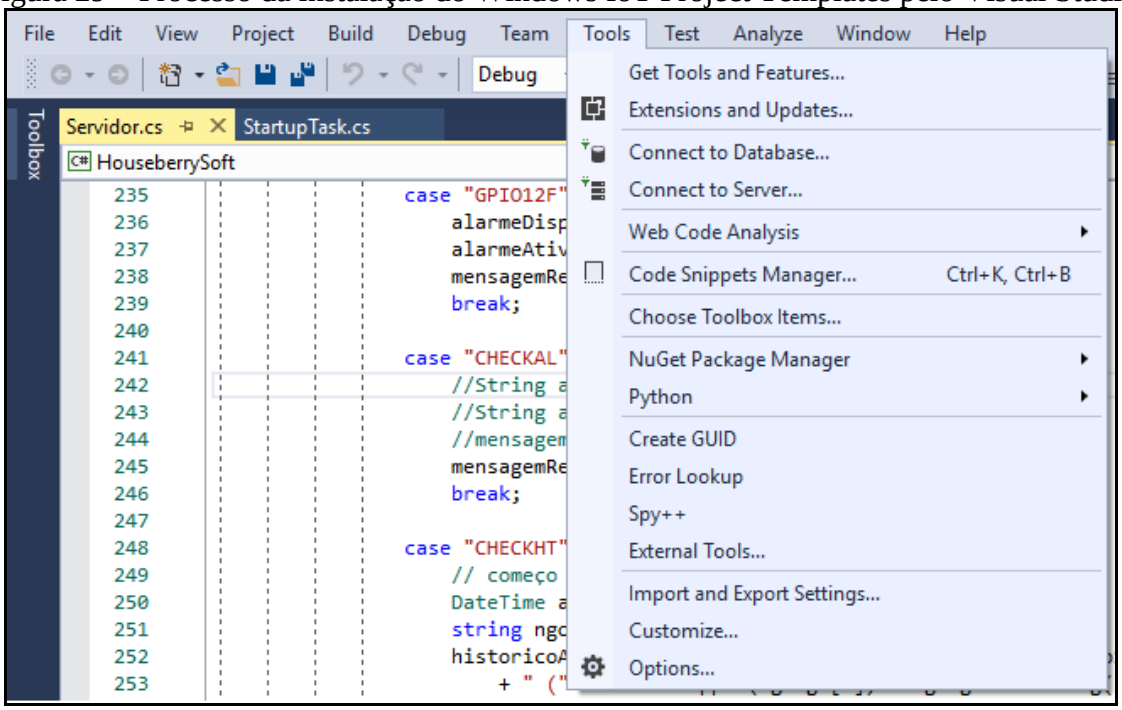



Fonte: elaborado pelo autor.

### 3.3.1.4 Desenvolvimento do software embarcado

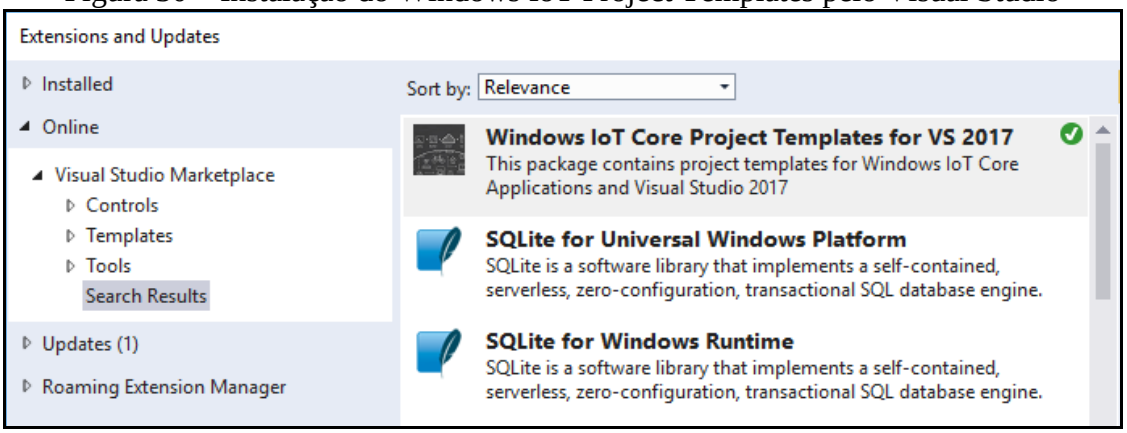
O software embarcado no Raspberry Pi 3 Model B foi escrito na linguagem de programação C# utilizando o IDE Visual Studio. Para explorar e executar projetos da IoT é necessário configurar o ambiente de desenvolvimento. Para tanto, foi primeiramente instalado a extensão do Windows IoT Project Templates diretamente do Visual Studio no menu `Tools`, clicando em `Extensions and Updates` conforme exibido na Figura 29. Na caixa de diálogo que aparece para extensões e atualizações, indo em `Online` e pesquisando pelo Windows IoT Project Templates foi realizada a instalação da extensão conforme demostrado na Figura 30.

Figura 29 – Processo da instalação do Windows IoT Project Templates pelo Visual Studio




Fonte: elaborado pelo autor.

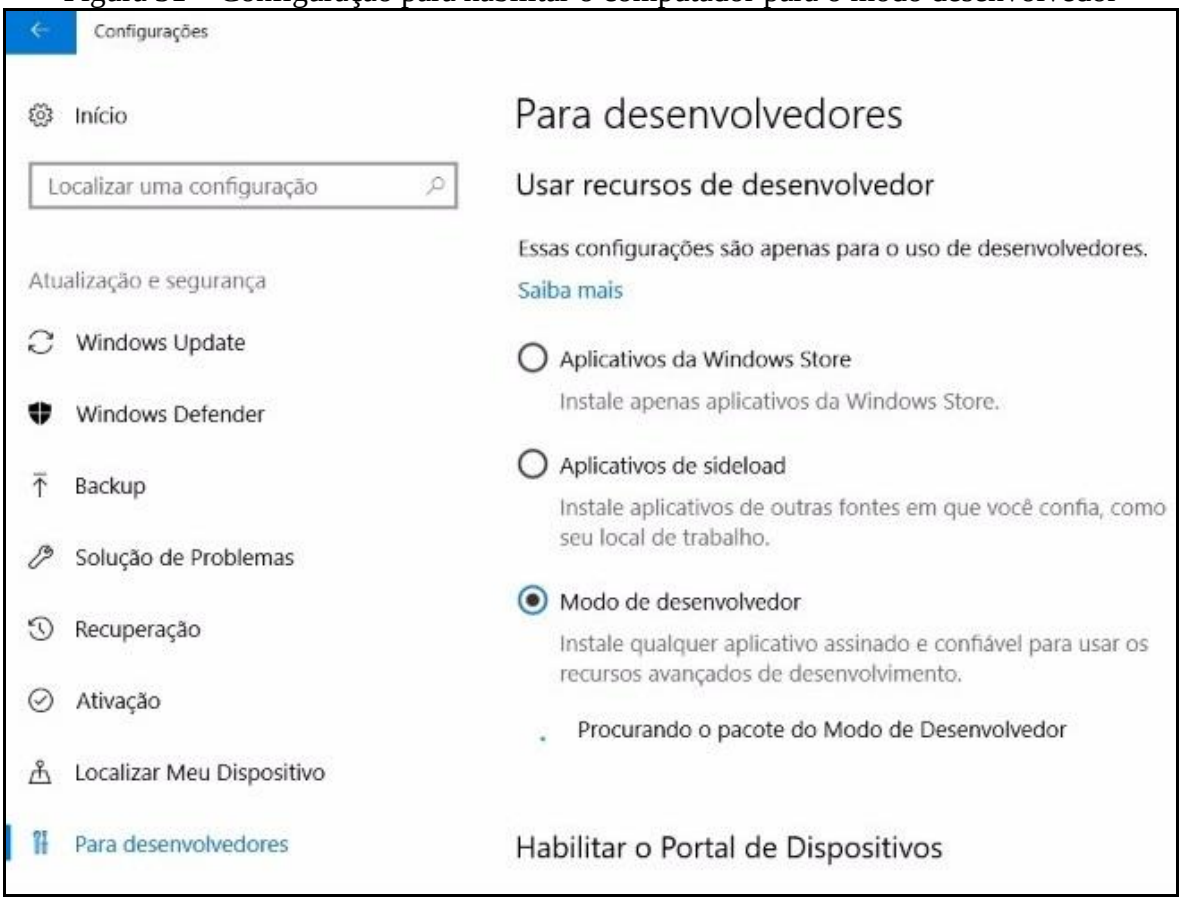
Figura 30 – Instalação do Windows IoT Project Templates pelo Visual Studio




Fonte: elaborado pelo autor.

Depois da instalação do Windows IoT Project Templates, foi necessário habilitar o computador para o modo desenvolvedor antes de iniciar o desenvolvimento do software. Veja na Figura 31 a configuração para habilitar o computador para o modo desenvolvedor.

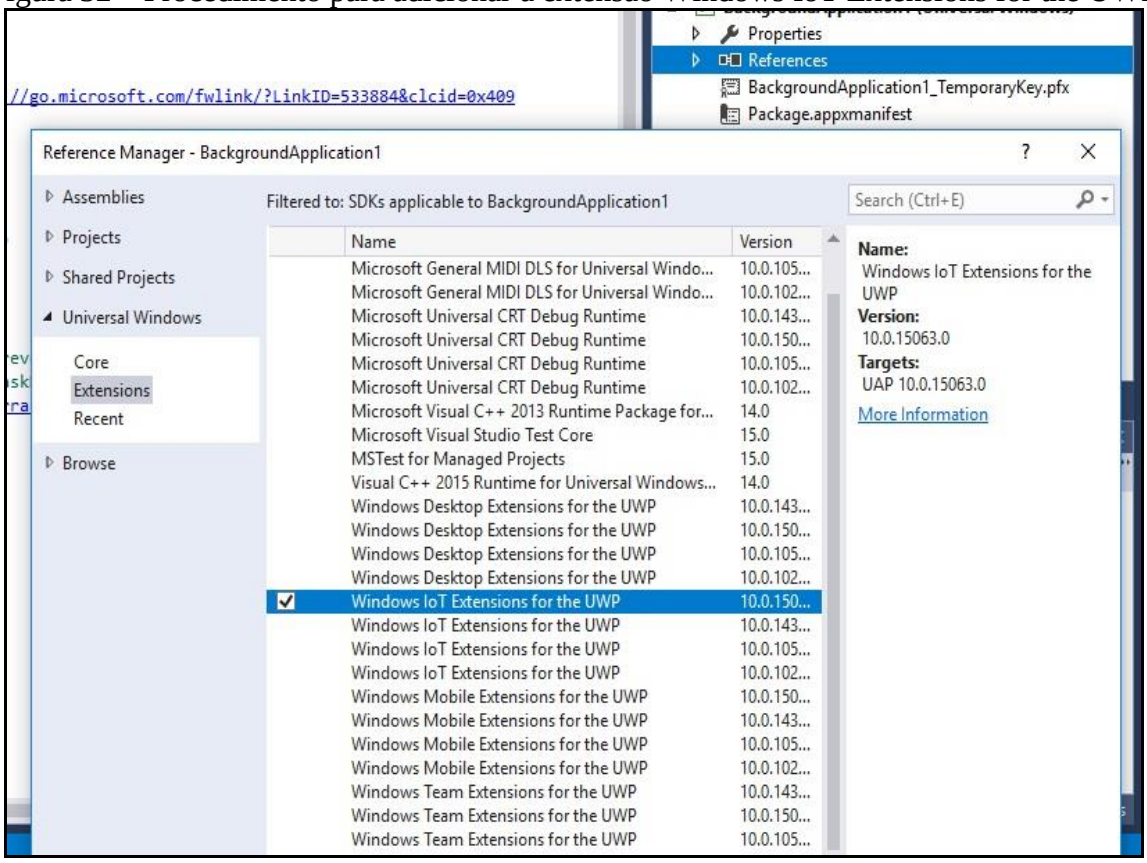
Figura 31 – Configuração para habilitar o computador para o modo desenvolvedor




Fonte: elaborado pelo autor.

Para o desenvolvimento do software embarcado no Visual Studio foi adicionado nas referências do projeto a extensão Windows IoT Extensions for the UWP necessária para o desenvolvimento de aplicativos UWP. A Figura 32 demonstra o procedimento para adicionar a extensão Windows IoT Extensions for the UWP.

Figura 32 – Procedimento para adicionar a extensão Windows IoT Extensions for the UWP




Fonte: elaborado pelo autor.

O software embarcado é um servidor socket configurado para rodar constantemente em uma porta especificada, aguardando uma conexão do cliente para a transferência dos comandos de controle. Em C# foi utilizada a biblioteca `Windows.Networking.Sockets` que fornece as classes de sockets necessárias para comunicações de rede para aplicativos UWP. Ao efetuar e aceitar a conexão, o servidor chama a função determinada para processar a informação recebida e em seguida retorna ao cliente o resultado do processamento, liberando a porta para permitir eventualmente que o cliente se conecte novamente. Veja o trecho de código no Quadro 5 que mostra o processo de inicialização do servidor, de aceitação da conexão com o cliente, do processamento de informação e do retorno do resultado, tudo feito através de threads.

Quadro 5 – Inicialização do servidor, aceitação da conexão, processamento da informação e retorno do resultado

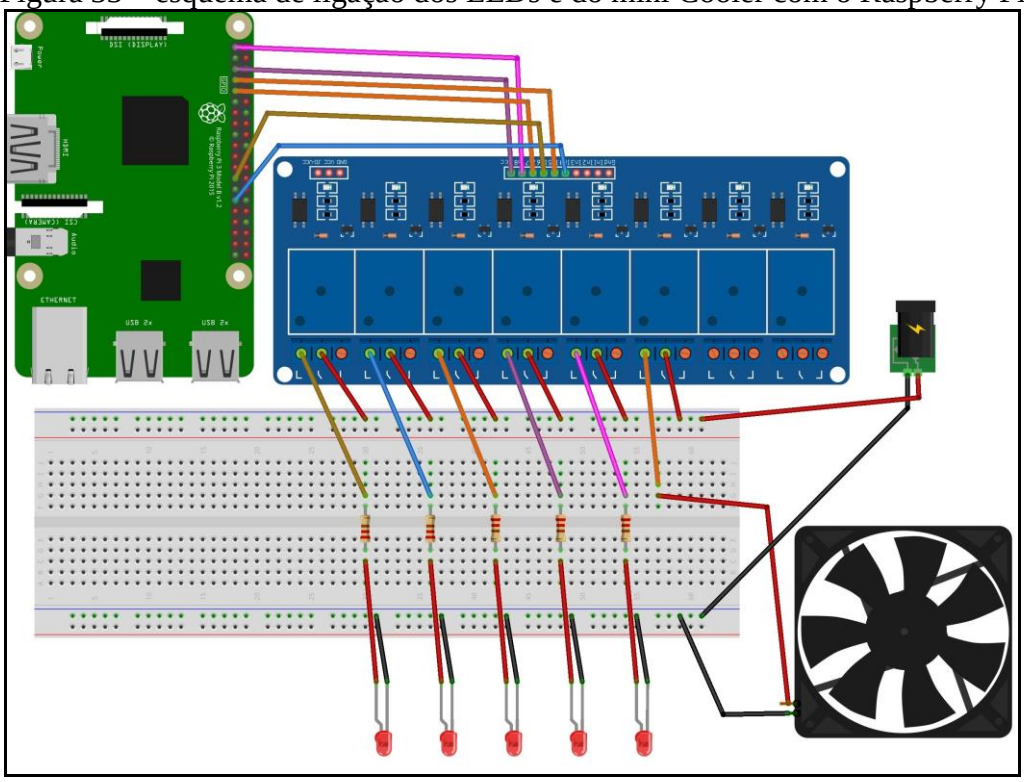
```
using Windows.Networking.Sockets;
using Windows.Storage.Streams;
//...

        private StreamSocketListener ssl;
        private const uint TAMANHO_BUFFER = 8192;

        public Servidor(int porta) {
            ssl = new StreamSocketListener();
            ssl.BindServiceNameAsync(porta.ToString());
            ssl.ConnectionReceived += async (sender, args) => {
                inicializa(sender, args);
            };
        }

        private async void inicializa(StreamSocketListener sender,
                                      StreamSocketListenerConnectionReceivedEventArgs args) {
            StringBuilder requisicao = new StringBuilder();
            using (IInputStream input = args.Socket.InputStream) {
                byte[] data = new byte[TAMANHO_BUFFER];
                IBuffer buffer = data.AsBuffer();
                uint leituraDado = TAMANHO_BUFFER;
                while (leituraDado == TAMANHO_BUFFER) {
                    await input.ReadAsync(buffer, TAMANHO_BUFFER, InputStreamOptions.Partial);
                    requisicao.Append(Encoding.UTF8.GetString(data, 0, data.Length));
                    requisicao.Append(Encoding.UTF8.GetString(data, 0, data.Length));
                    leituraDado = buffer.Length;
                }
            }
            string informacaoRecebida = requisicao.ToString().Substring(0, 7);
            string informacaoResposta = "";

            switch (informacaoRecebida) {
                case "GPIO02V":
                    hajaLuz(pinoGpio02, true);
                    informacaoResposta = $"Luz da Sala ligada!";
                    break;
                //...
            }

            using (IOutputStream output = args.Socket.OutputStream)
            using (Stream resposta = output.AsStreamForWrite()) {
                var header = Encoding.UTF8.GetBytes($"\n{informacaoResposta}\n");
                await resposta.WriteAsync(header, 0, header.Length);
                await resposta.FlushAsync();
            }
        }
```

Fonte: elaborado pelo autor.

Para a implementação do controle das luzes representadas por LEDs e o ar condicionado representado pelo mini Cooler, os pinos GPIO 2, GPIO 3, GPIO 4, GPIO 5, GPIO 6 e GPIO 17 do Raspberry Pi foram conectados com os pinos RA5, RE0, RE1, RC0, RC1 e RC2 do Módulo relé Clp Pic40-v4 que por sua vez foi ligado aos LEDs e o mini Cooler nas saídas a Relés por meio da placa PCI universal ilhada. A figura 33 apresenta o esquema de ligação dos LEDs e do mini Cooler com o Raspberry Pi. No software embarcado foi utilizada a estrutura `GpioPinValue` para atribuir o valor de cada pino GPIO e foram instanciadas variáveis do tipo `GpioPin` para representar cada pino GPIO. Ambos os recursos são da biblioteca `Windows.Devices.Gpio`. As instâcias do `GpioPin` foram primeiramente inicializadas com o valor dos pinos GPIO correspondentes para possibilitar o controle dos pinos através do software. Foi utilizada a função `GpioController.GetDefault()` para obter o controlador de GPIO do Raspberry Pi. Em seguida foi chamada a função

`GpioController.OpenPin()` para abrir cada pino GPIO. Uma vez abertos, os pinos foram configurados para inicializar como ligados usando a função `Write(GpioPinValue.Low)`. Cada pino foi especificado para executar em modo *output* utilizando a função `SetDriveMode(GpioPinDriveMode.Output)`. Uma vez a instância do `GpioOutputPin` acessado, pode ser alterado o estado dos pinos para ligar ou desligar os LEDs e o mini Cooler. Para ligar os LEDs ou o mini Cooler, basta escrever o valor `GpioPinValue.Low` para o pino e o valor `GpioPinValue.High` para desligar os LEDs ou o mini Cooler. Veja no Quadro 6 o código para ligar e desligar a luz e o ar condicionado.

Figura 33 – esquema de ligação dos LEDs e do mini Cooler com o Raspberry Pi



Fonte: elaborado pelo autor

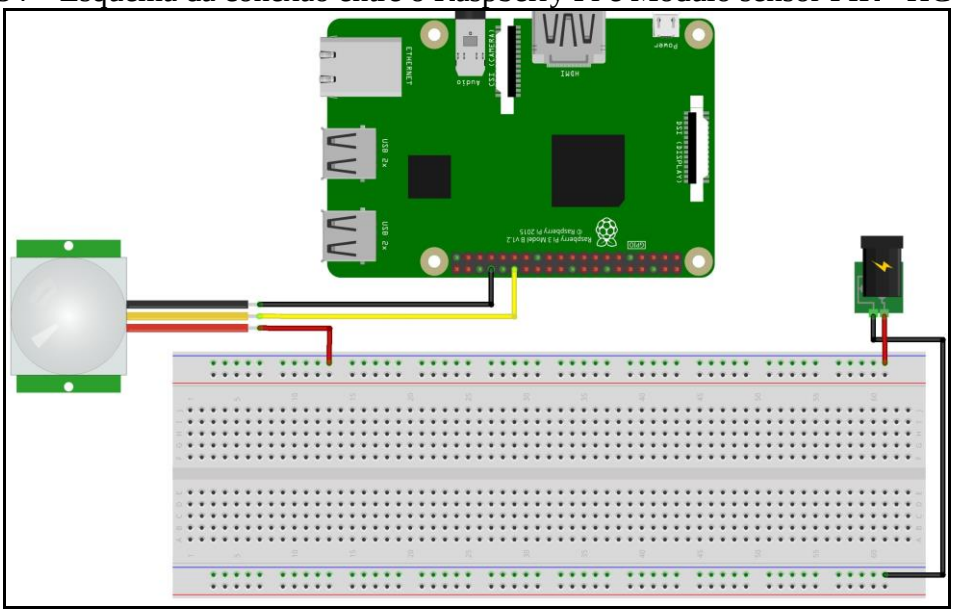
Quadro 6 – Código para ligar e desligar a luz e o ar condicionado

```
using Windows.Devices.Gpio;
//...

        private GpioPinValue valorPinoGpio;
        //Luzes
        private GpioPin pinoGpio02, pinoGpio03, pinoGpio04, pinoGpio05, pinoGpio06;
        //Ar condicionado
        private GpioPin pinoGpio17;

        private void inicializaGPIO() {
            var gpio = GpioController.GetDefault();
            //Luzes
            pinoGpio02 = gpio.OpenPin(2);
            pinoGpio02.SetDriveMode(GpioPinDriveMode.Output);
            pinoGpio02.Write(GpioPinValue.Low);
            //Ar condicionado
            pinoGpio17 = gpio.OpenPin(17);
            pinoGpio17.SetDriveMode(GpioPinDriveMode.Output);
            pinoGpio17.Write(GpioPinValue.Low);
        }
        private async void inicializa(StreamSocketListener sender,
                        StreamSocketListenerConnectionReceivedEventArgs args) {
            //...

            switch (informacaoRecebida) {
                case "GPIO02V":
                    hajaLuz(pinoGpio02, true);
                    informacaoResposta = $"Luz da Sala ligada!";
                    break;
                case "GPIO02F":
                    hajaLuz(pinoGpio02, false);
                    informacaoResposta = $"Luz da Sala desligada!";
                    break;
                case "GPIO17V":
                    ligarArCondicionado(true);
                    informacaoResposta = $"Ar Condicionado Ligado!";
                    break;
                case "GPIO17F":
                    ligarArCondicionado(false);
                    informacaoResposta = $"Ar Condicionado Desligado";
                    break;
            }
            //...
        }
        //Iluminacao
        private void hajaLuz(GpioPin pinoGpio, bool e) {
            valorPinoGpio = e ? GpioPinValue.High : GpioPinValue.Low;
            pinoGpio.Write(valorPinoGpio);
        }
        //Ar Condicionado
        private void ligarArCondicionado(bool e) {
            valorPinoGpio = e ? GpioPinValue.High : GpioPinValue.Low;
            pinoGpio17.Write(valorPinoGpio);
        }
```

Fonte: elaborado pelo autor.

Para utilizar o Módulo sensor de movimento PIR - HC-SR501 no protótipo, os pinos OUTPUT e GND do módulo foram conectados diretamente ao pino GPIO 12 e um dos pinos GND do Raspberry Pi. o Módulo foi alimentado com 5V por meio da placa PCI universal ilhada. A Figura 34 mostra o esquema da conexão entre o Raspberry Pi e Módulo sensor PIR - HC-SR501. Para acessar o sensor de movimento PIR - HC-SR501 através do software embarcado, foi aplicado o mesmo procedimento utilizado com os LEDs e o mini Cooler, porém, o pino GPIO conectado ao sensor de movimento foi configurado para executar em modo *intput* utilizando a função `SetDriveMode(GpioPinDriveMode.Input)`. Para realizar a captura da detecção de movimentos foi utilizada a função `GpioPinEdge.FallingEdge` que retorna o tipo de alteração ocorrida ao valor do pino GPIO quando o sensor detecta algum movimento. No Quadro 7 veja o trecho de código para realizar a captura da detecção de movimentos no software.

Figura 34 – Esquema da conexão entre o Raspberry Pi e Módulo sensor PIR - HC-SR501



Fonte: elaborado pelo autor.

Quadro 7 – Captura da detecção de movimentos

```
using Windows.Devices.Gpio;
//...

        private void inicializaGPIO() {
            var gpio = GpioController.GetDefault();

            //Sensor de movimento
            pinoGpio12 = gpio.OpenPin(12);
            pinoGpio12.SetDriveMode(GpioPinDriveMode.Input);
            alarmeAtivado = false;
            pinoGpio12.ValueChanged += detectaMovimento;

            //...
        }

        private async void detectaMovimento(GpioPin sender,
                                            GpioPinValueChangedEventArgs args) {
            var detectou = args.Edge == GpioPinEdge.FallingEdge;
            if (alarmeAtivado) {
                if (detectou){
                    qtdAlarme++;

                    agora = DateTime.Now;
                    mukolo = agora.ToString("dddd", cult);

                    enviaEmail();
                    disparaAlarme();
                    addHistorico();
                }
            }
        }
```

Fonte: elaborado pelo autor.

Após a detecção de movimentos, é realizado automaticamente o envio de e-mail para o usuário a partir do software embarcado para notificar sobre o disparo do alarme. Para possibilitar o envio do e-mail foi utilizado um cliente SMTP útil para aplicações UWP. Desse modo, foi necessário incluir no projeto a biblioteca `LightBuzz.SMTP`. O Quadro 8 apresenta a função para o envio de e-mail de notificação.

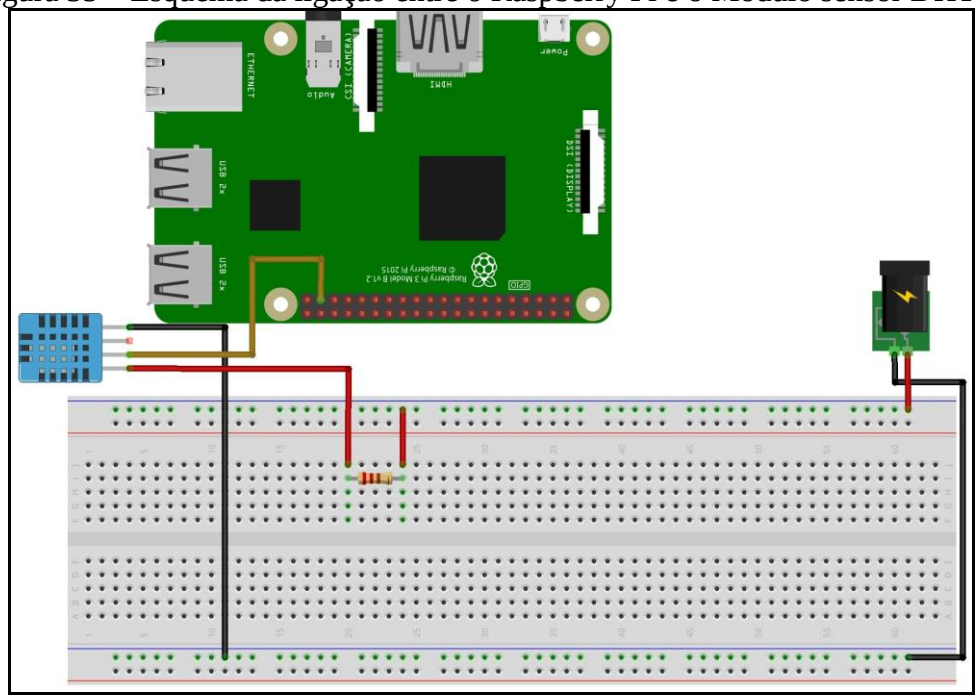
Quadro 8 – Envio de e-mail de notificação sobre o disparo do alarme

```
using LightBuzz.SMTP;
using Windows.ApplicationModel.Email;
//...

        private async void enviaEmail() {

            using (SmtpClient client = new SmtpClient("smtp-mail.outlook.com", 587, false,
                                                      "plam.l@live.fr", "#Jo31619")) {
                EmailMessage emailMessage = new EmailMessage();
                emailMessage.To.Add(new EmailRecipient("plam.lusembo@gmail.com"));
                emailMessage.Subject = "Alerta - Alarme disparada";
                emailMessage.Body = "Disparo Alarme Qtd.O "
                    + (qtdAlarme > 9 ? ""+qtdAlarme : "0"+ qtdAlarme)
                    +": Foi detectado movimentos suspeitos na sua residência neste horário: "
                    + agora.ToString("dd/MM/yyyy HH:mm:ss") + " (" + char.ToUpper(mukolo[0])
                    + mukolo.Substring(1) + ") - Winberry by Plamedi L. Lusembo";

                await client.SendMailAsync(emailMessage);
            }
        }
```

Fonte: elaborado pelo autor

O Módulo sensor de temperatura e umidade DHT11 foi conectado ao pino GPIO 26 do Raspberry Pi e à placa PCI universal ilhada para ser alimentado com 5V. A Figura 35 ilustra o esquema de ligação entre o Raspberry Pi e o Módulo sensor DHT11. O sensor DHT11 se comunica usando o protocolo OneWire. Este protocolo requer um tempo muito preciso para obter os dados do sensor. O Windows 10 IoT Core não possui atualmente uma implementação nativa de biblioteca para dar suporte ao protocolo OneWire. Neste trabalho foi utilizada a biblioteca `Sensor.Dht` desenvolvida por terceiro. A biblioteca foi inicialmente escrita em C++ e depois traduzida em C#, possibilitando sua utilização no desenvolvimento do software embarcado. A biblioteca possui uma classe chamada Dht11. O pino GPIO correspondente foi aberto e passado para o construtor da classe Dht11 e foi configurado para executar em modo *imput*. Para fazer a leitura dos dados retornados pelo sensor foi usado o método `GetReadingAsync()`. Os dados da leitura de temperatura estão na escala de grau Celsius e a umidade em porcentagem. O DHT11 retorna somente valores inteiros. Veja no Quadro 9 o trecho do código para a leitura da temperatura e umidade.

Figura 35 – Esquema da ligação entre o Raspberry Pi e o Módulo sensor DHT11



Fonte: elaborado pelo autor.

Quadro 9 – Leitura da temperatura e umidade

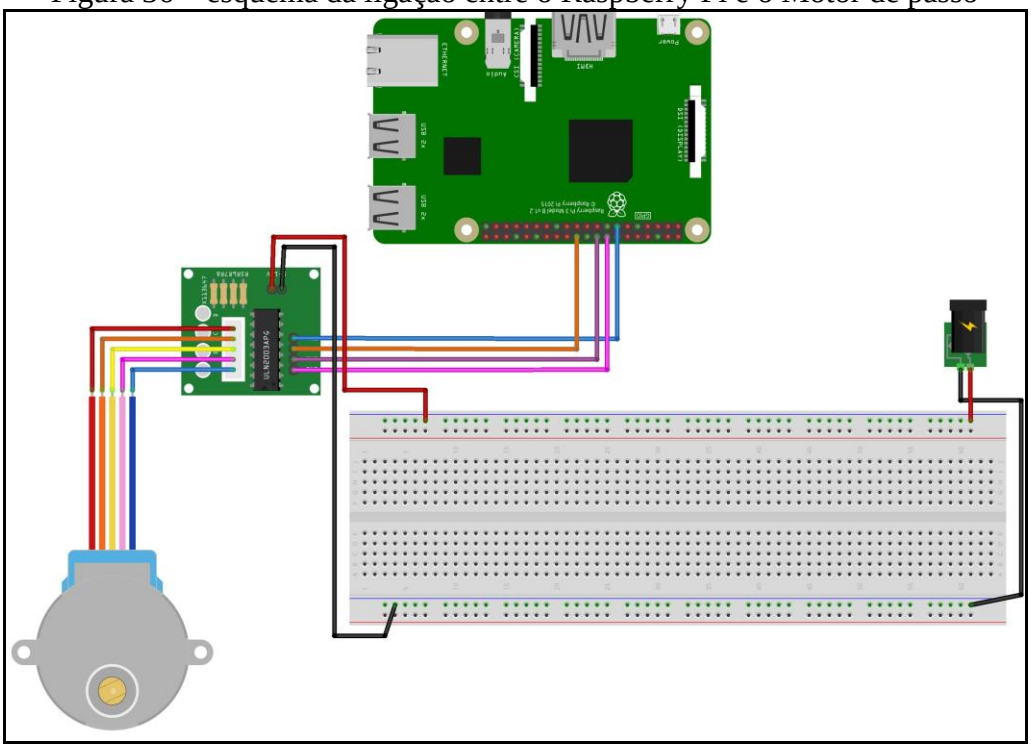
```
using Sensors.Dht;
//...

        private IDht dht = null;
        private String resultadoDHT;

        //DHT
        private async Task medeDHT() {
            DhtReading reading = new DhtReading();

            reading = await dht.GetReadingAsync().AsTask();

            if (reading.IsValid) {
                this.Temperatura = Convert.ToSingle(reading.Temperature);
                this.Humidade = Convert.ToSingle(reading.Humidity);

                this.resultadoDHT = this.Temperatura + "-" + this.Humidade;
            }
        }
```

Fonte: elaborado pelo autor

Com a interface de 4 pinos do Módulo Driver Uln2003 do Motor de passo, foram conectados os pinos GPIO 22, GPIO 23, GPIO 24, GPIO 25 do Raspberry Pi e o motor foi alimentado com 5V através da placa PCI universal ilhada. A Figura 36 exibe o esquema da ligação entre o Raspberry Pi 3 Model B e o Motor de passo. No desenvolvimento do software embarcado os 4 pinos GPIO foram especificados para executar em modo o*utput* utilizando a função `SetDriveMode(GpioPinDriveMode.Ouput)`. Foi implementada uma matriz quadrada bidimensional do tamanho 4 do tipo `GpioPinValue` para agrupar o conjunto de estados atribuídos aos pinos e assim permitir que o motor gire em uma direção determinada.

Somente nas posições da diagonal principal da matriz foi atribuído o valor `GpioPinValue.High` e nas demais posições `GpioPinValue.Low` para permitir que o motor gire no sentido horário. Para que o motor gire no sentido anti-horário, foi utilizado um loop no método `abrePortao(bool abrir)` para atribuir o valor `GpioPinValue.Low` nas posições da diagonal principal da matriz e nas demais posições `GpioPinValue.Low`. É possível inverter o sentido com o valor do parâmetro passado no método `abrePortao(bool abrir)`. Não possui fim de curso, o motor para de girar quando completa a quantidade de passos definidos no software. No quadro 10 veja o código para abrir e fechar o portão.

Figura 36 – esquema da ligação entre o Raspberry Pi e o Motor de passo



Fonte: elaborado pelo autor

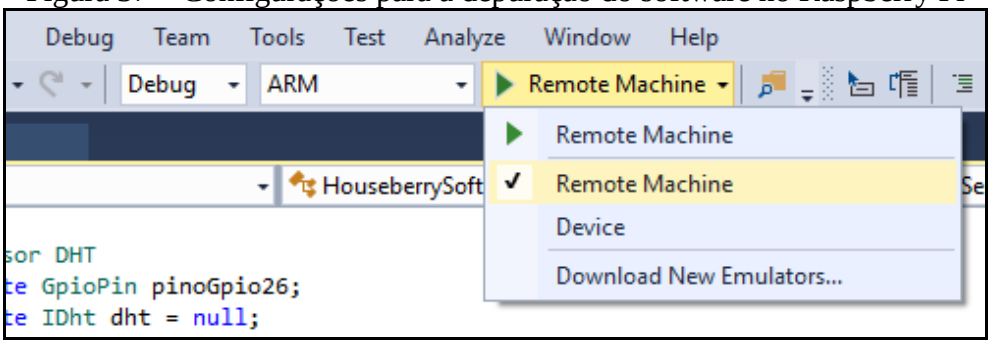
Quadro 10 – Código para abrir e fechar o portão

```
//Portão
private const int QTD_PINOS = 4;
private readonly GpioPin[] pinosGpio22a25 = new GpioPin[QTD_PINOS];
private readonly GpioPinValue[][] valoresPinosGpio = {
    new[] {GpioPinValue.High, GpioPinValue.Low, GpioPinValue.Low, GpioPinValue.Low},
    new[] {GpioPinValue.Low, GpioPinValue.High, GpioPinValue.Low, GpioPinValue.Low},
    new[] {GpioPinValue.Low, GpioPinValue.Low, GpioPinValue.High, GpioPinValue.Low},
    new[] {GpioPinValue.Low, GpioPinValue.Low, GpioPinValue.Low, GpioPinValue.High}
};

private void inicializaGPIO() {
    var gpio = GpioController.GetDefault();

    pinosGpio22a25[0] = gpio.OpenPin(22);
    pinosGpio22a25[1] = gpio.OpenPin(23);
    pinosGpio22a25[2] = gpio.OpenPin(24);
    pinosGpio22a25[3] = gpio.OpenPin(25);

    foreach (var pinoGpio in pinosGpio22a25) {
        pinoGpio.SetDriveMode(GpioPinDriveMode.Output);
        pinoGpio.Write(GpioPinValue.Low);
    }
}

private async Task abrePortao(bool abrir) {
    int passos = 400;

    for (int x = 0; x < passos; x++) {
        for (int y = 0; y < QTD_PINOS; y++) {
            for (int z = 0; z < QTD_PINOS; z++) {
                pinosGpio22a25[z].Write(valoresPinosGpio[abrir ? z : 3 - z][y]);
            }
            await Task.Delay(1);
        }
    }

    foreach (var gpioPin in pinosGpio22a25) {
        gpioPin.Write(GpioPinValue.Low);
    }
}
```

Fonte: elaborado pelo autor.

O *deploy* do software no Raspberry Pi é realizado remotamente via rede a partir do Visual Studio. Antes de efetuar o *deploy* do software, foi definido a arquitetura `ARM` no menu localizada na barra de ferramentas do Visual Studio. Em seguida, no menu encontrado diretamente à direita foi selecionado `Remote Machine` como demonstrado na Figura 37. Foi selecionada a arquitetura ARM por que é utilizada no processador do Raspberry Pi 3 Model B. Esta arquitetura também é utilizada no processador dos demais dispositivos de baixo custo e também em smartphones e tabletes diferentemente da arquitetura x86 utilizada geralmente em processadores de CPU de desktop.

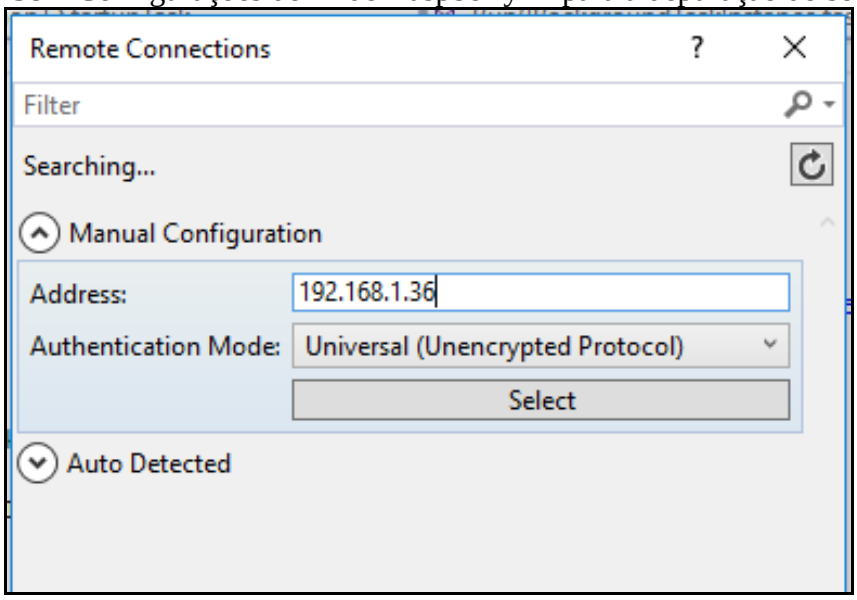
Figura 37 – Configurações para a depuração do software no Raspberry Pi




Fonte: elaborado pelo autor

Depois do Visual Studio apresentar a caixa de diálogo para conexões remotas, foi inserido o endereço IP do Rasberry Pi 3 Model B e selecionado `Universal` para `Windows Authentication` conforme demonstrado na Figura 38.

Figura 38 – Configurações do IP do Raspberry Pi para a depuração do software




Fonte: elaborado pelo autor.

### 3.3.1.5 Desenvolvimento do aplicativo móvel

O aplicativo móvel para o gerenciamento do sistema de automação foi desenvolvido para a plataforma Android utilizando a linguagem de programação Java e a linguagem XML para a definição da interface gráfica no IDE Android Studio. O aplicativo foi configurado como cliente socket para se conectar ao servidor através da porta especificada e usando o IP do servidor para formar o endereço da aplicação a fim de estabelecer a conexão e possibilitar o envio de comandos de controle. Foi utilizado o banco de dados SQLite para guardar as informações sobre o usuário do sistema e as configurações de IP do servidor e da porta para conexão via socket. Foi utilizada a API de sockets do Java para criar o aplicativo cliente. O método `transmite(Context context, String cmd)` foi criado para ser responsável

por solicitar a conexão com o servidor e enviar o comando de controle passado por parâmetro que, consequentemente, é recebido pelo servidor do outro lado. Este método é chamado sempre que uma ação de controle é executada, por exemplo para ligar a luz, abrir o portão, ativar o alarme, consultar a temperatura e umidade. O método retorna o resultado do processamento do servidor. O Quadro 11 mostra o conteúdo deste método.

Quadro 11 – Método responsável pela conexão com o servidor e o envio de comandos

```
public String transmite(Context context, String cmd) {
    configuraServidor(context);
    cmdEnviado = cmd;
    cmdRecebido = "";
    Runnable r = new Runnable() {
        @Override
        public void run() {
            try {
                final Socket client = new Socket(ipServidor, porta);
                PrintStream saida = new PrintStream(client.getOutputStream());
                saida.println(cmdEnviado);
                cmdRecebido = "";
                Runnable runnable = new Runnable() {
                    @Override
                    public void run() {
                        try {
                            Scanner s = new Scanner(client.getInputStream());
                            while (s.hasNextLine()) {
                                cmdRecebido = s.nextLine();
                            }
                        } catch (IOException ioe) {
                        }
                    }
                };
                Thread thread = new Thread(runnable);
                thread.start();
                while (cmdRecebido.isEmpty()) {
                    try {
                        Thread.sleep(0);
                    } catch (InterruptedException ex) {
                    }
                }
                saida.close();
                client.close();
            } catch (IOException ioe) {
            }
        }
    };
    Thread t = new Thread(r);
    t.start();
    while (cmdRecebido.isEmpty()) {
        try {
            Thread.sleep(0);
        } catch (InterruptedException ex) {
        }
    } return cmdRecebido;
}
```

Fonte: elaborado pelo autor.

Quando uma ação de controle é realizada, o método `onClick(View view)` é subscrito para executar a implementação que invoca o método mostrado no Quadro 11 passando por parâmetro o comando específico referente a ação solicitada. A mesma implementação é executada na abertura de uma página do aplicativo habilitada para o controle

a fim de atualizar o status dos dispositivos que aparecem nessa página. Veja no Quadro 12 o trecho de código da implementação para a ação de controle de luz.

Quadro 12 – Implementação para a ação de controle de luz

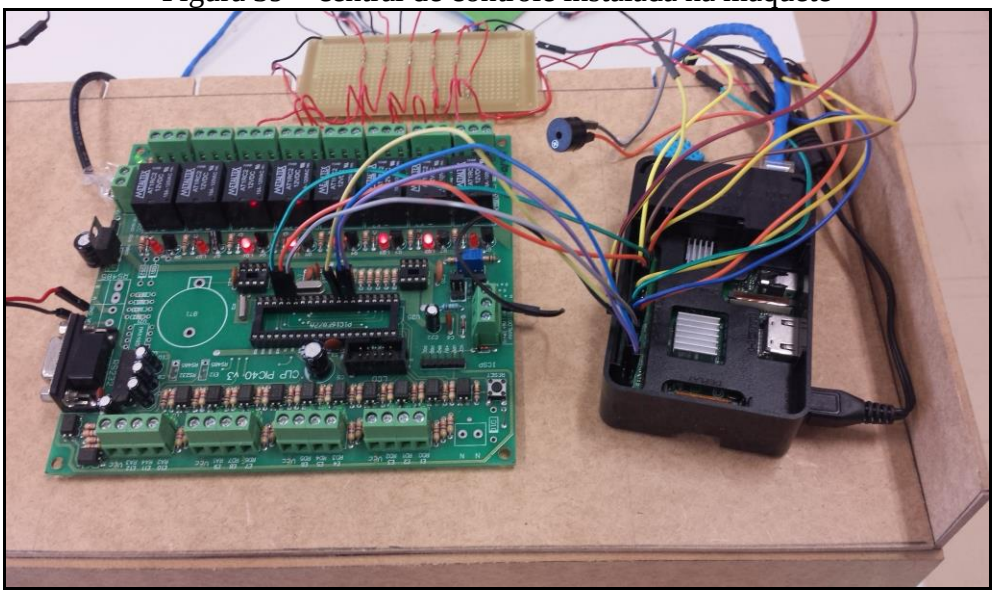
```
@Override
public void onClick(View view) {
    String inf = "";
    switch (view.getId()) {
        case R.id.swtLuzSala:
            checkarTodasAsLuzes();
            inf = Comunicador.getInstance().transmite(getActivity(),
                                        swtLuzSala.isChecked() ? "GPIO02V" : "GPIO02F");
            break;
        case R.id.swtLuzCozinha:
            checkarTodasAsLuzes();
            inf = Comunicador.getInstance().transmite(getActivity(),
                                        swtLuzCozinha.isChecked() ? "GPIO03V" : "GPIO03F");
            break;
        case R.id.swtLuzQuartoI:
            checkarTodasAsLuzes();
            inf = Comunicador.getInstance().transmite(getActivity(),
                                        swtLuzQuartoI.isChecked() ? "GPIO04V" : "GPIO04F");
            break;
        case R.id.swtLuzQuartoII:
            checkarTodasAsLuzes();
            inf = Comunicador.getInstance().transmite(getActivity(),
                                        swtLuzQuartoII.isChecked() ? "GPIO05V" : "GPIO05F");
            break;
        case R.id.swtLuzesJardim:
            checkarTodasAsLuzes();
            inf = Comunicador.getInstance().transmite(getActivity(),
                                        swtLuzesJardim.isChecked() ? "GPIO06V" : "GPIO06F");
            break;
```

Fonte: elaborado pelo autor.

### 3.3.2 Operacionalidade da implementação

Foi construído uma maquete de demonstração para simular o protótipo do sistema de automação residência. A central de controle foi instalada na parte superior da maquete conforme pode ser visualizado na figura 39.

Figura 39 – central de controle instalada na maquete



Fonte: elaborado pelo autor.

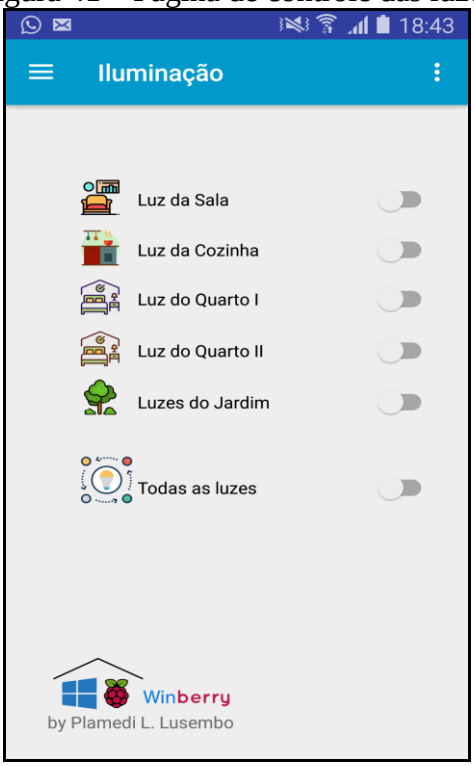
Para utilizar o sistema, é necessário rodar primeiramente o servidor e, logo depois, o aplicativo gerenciador pode ser acessado para executar o cliente. O aplicativo gerenciador dispõe de um mecanismo de autenticação onde o usuário informa o e-mail e a senha a fim de acessar ao sistema de automação. Caso as configurações do IP do servidor e a porta de conexão não estejam corretas, o usuário pode alterar as configurações através da seleção no *checkbox* para configurar o servidor após o *login*, como pode ser visualizado na Figura 40.

Figura 40 – Acesso ao sistema de automação através do aplicativo gerenciador


Fonte: elaborado pelo autor.

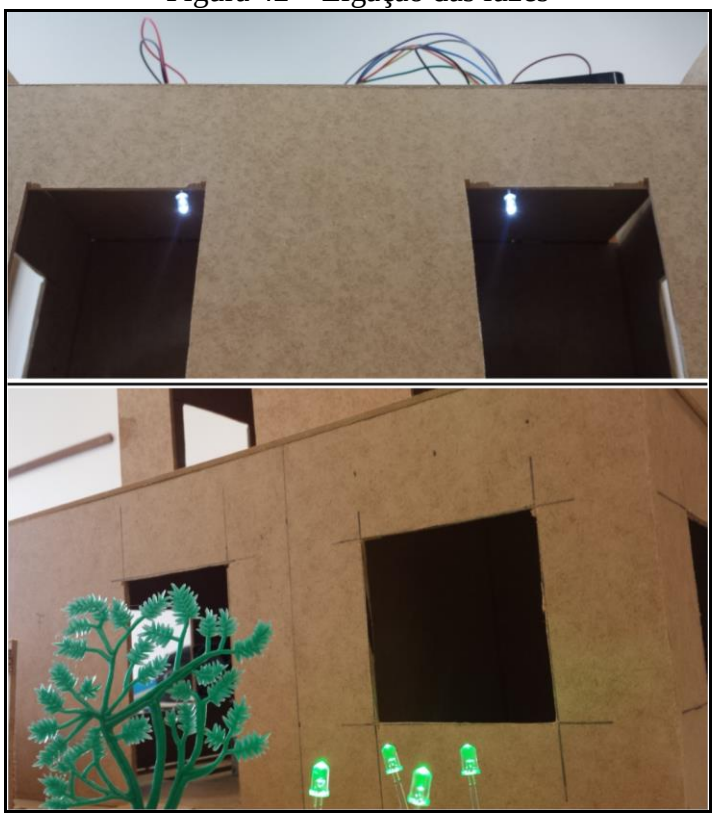
Após efetuar a autenticação, o aplicativo gerenciador inicia o funcionamento exibindo a página de controle das luzes de todos os cômodos com o status atual de cada luz. Esta página apresenta 6 botões *switch* de alternância de dois estados que possibilita selecionar entre duas opções para ligar ou desligar a luz ao clicar no *switch*. A Figura 41 ilustra a página de controle das luzes. Ao clicar no *switch* para ligar ou desligar a luz de determinado cômodo da residência, é enviado o comando da solicitação ao servidor, o servidor processa a solicitação, executa o comando sobre o dispositivo específico de luz e devolve ao cliente a informação sobre o estado atual da luz. A Figura 42 mostra o comando de ligação das luzes executado na maquete.

Figura 41 – Página de controle das luzes




Fonte: elaborado pelo autor.

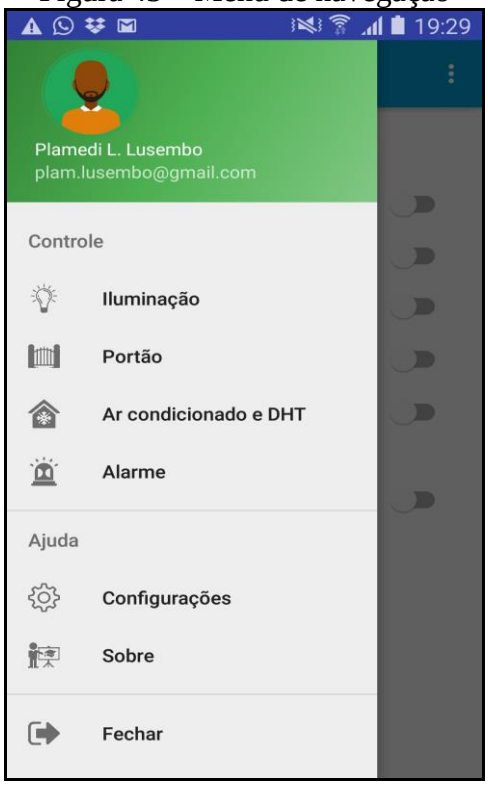
Figura 42 – Ligação das luzes



Fonte: elaborado pelo autor.

Outra alternativa para acessar a página de controle de luz e as demais páginas de funções de controle é através do menu de navegação acessível no lado esquerdo superior da tela do aplicativo gerenciador. A figura 43 exibe o menu de navegação.

Figura 43 – Menu de navegação




Fonte: elaborado pelo autor.

No menu de navegação, ao acessar na opção `Ar condicionado e DHT` que dá acesso à página da consulta de temperatura e umidade, o usuário tem a possibilidade de ligar ou desligar o ar condicionado. Esta funcionalidade pode ser visualizada na Figura 44.

Figura 44 – Consulta de temperatura e umidade e controle do ar condicionado

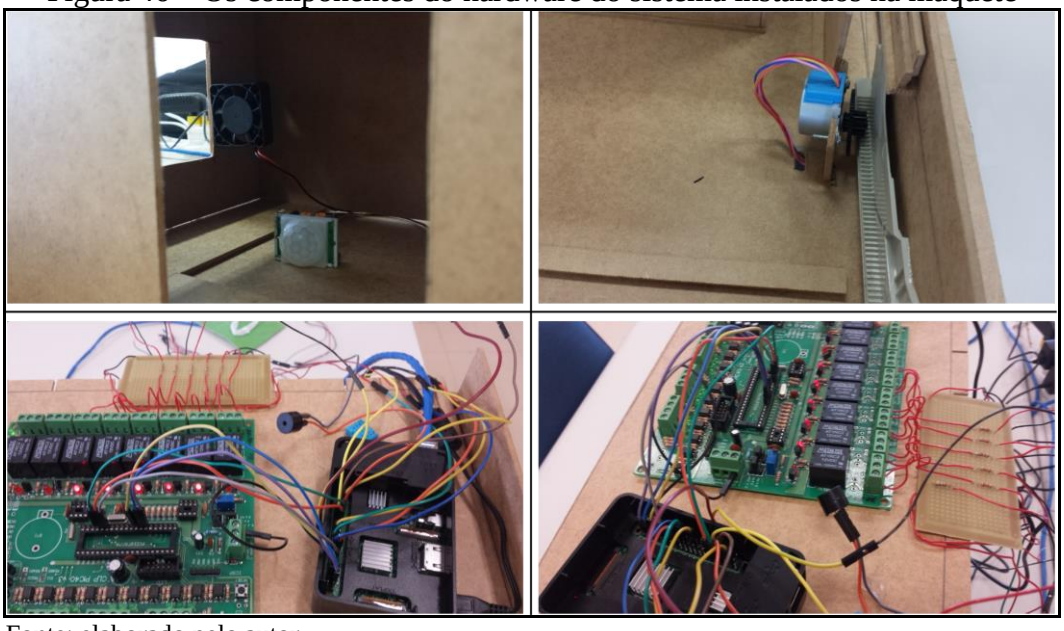

Fonte: elaborado pelo autor.

Ao acessar a opção `Alarme`, no menu de navegação é exibida a página com a funcionalidade de ativar ou desativar o alarme, possibilitando que o módulo sensor PIR - HC-SR501 seja habilitado ou desabilitado para a captura de movimentos. Nesta página também é possível visualizar o histórico dos últimos disparos do alarme conforme pode ser observado na Figura 45. A instalação do sensor de movimento na maquete e dos demais componentes do hardware do sistema pode ser visualizada na figura 46.

Figura 45 – Controle do alarme e visualização o histórico dos disparos do alarme


Fonte: elaborado pelo autor.

Figura 46 – Os componentes do hardware do sistema instalados na maquete



Fonte: elaborado pelo autor.

## 3.4 ANÁLISE DOS RESULTADOS

O desenvolvimento deste trabalho resultou na implementação de um protótipo de sistema de automação residencial utilizando o Raspberry Pi 3 Model B com o sistema operacional Windows 10 IoT Core batizado de Winberry. O sistema possibilita a consulta das informações e o controle dos dispositivos do ambiente residencial remotamente por meio de um aplicativo móvel. Principalmente, foram elaborados testes exaustivos por meio de simulação e execução do sistema para asseverar o bom funcionamento de todos os artefatos. Não foi identificado nenhuma falha de comunicação entre a central de controle e os sensores e atuadores. Os testes de funcionalidade, de interface e de desempenho foram realizados com sucesso e permitiram verificar a usabilidade e avaliar o desempenho na utilização do sistema. Também foi aplicado um conjunto de condições de teste com o objetivo de validar os requerimentos descritos nos requisitos. Estes testes permitiram garantir que todos os requisitos funcionais do protótipo desenvolvido foram plenamente atendidos assim como pode ser observado no relatório apresentado na tabela 1.

Tabela 1 – Percentual da atendibilidade dos Requisitos Funcionais (RF)

| Requisitos Funcionais (RF) | Atendido |
|---|---|
| RF01: dispor de um controle de acesso ao aplicativo gerenciador por meio de um mecanismo de autenticação do usuário baseado no nome de usuário e senha | 100% |
| RF02: permitir que o usuário altere as configurações de conexão via rede | 100% |
| RF03: possibilitar o controle da iluminação, permitindo ligar e desligar remotamente as luzes da residência | 100% |
| RF04: possibilitar ligar e desligar o ar condicionado | 100% |
| RF05: possibilitar a consulta da temperatura e umidade | 100% |
| RF06: possibilitar a abertura e o fechamento do portão automaticamente | 100% |
| RF07: possibilitar a ativação e a desativação do sistema de alarme | 100% |
| RF08: permitir o envio automático de e-mail de notificação sobre o disparo do alarme | 100% |
| RF09: possibilitar a visualização do histórico de disparos do alarme através do aplicativo gerenciador | 100% |
| RF10: possibilitar a visualização do status dos dispositivos e equipamentos através do aplicativo gerenciador | 100% |

Fonte: elaborado pelo autor.

O Quadro 13 apresenta um comparativo das características entre os trabalhos correlatos apresentados na fundamentação teórica, confrontando com os resultados do trabalho desenvolvido.

Quadro 13 – Comparativo entre trabalhos correlatos

| **Características** | **Trabalhos correlatos** | | | **Winberry** |
|---|---|---|---|---|
| | **Censi (2001)** | **Botke (2014)** | **Gadotti (2010)** | |
| Utiliza conceito da IoT | × | × | × | × |
| Dispositivo IoT | Rabbit 2000 TCP/IP | Arduino Mega 2560 | FEZ Domino | Raspberry Pi 3 Model B |
| SO do Dispositivo IoT | | | | Windows 10 IoT Core |
| Linguagem de programação | Dynamic C | C++ | C# | C# e Java |
| Utiliza sockets para comunicação via rede | × | | × | × |
| Modo de comunicação para envio de comandos | E-mail | Wi-Fi | Twitter | Wi-Fi |
| Plataforma (Desktop, Web, Android, iOS) | Web | Web | Web | Android |

Fonte: elaborado pelo autor.

No Quadro 13 pode ser observado que o conceito da IoT foi aplicado em todos os trabalhos. Embora não tenha sido principalmente o objeto do estudo proposto nos trabalhos, o paradigma da IoT foi implicitamente envolvido em virtude da necessidade de tornar os equipamentos de uma residência conectáveis à rede. Isto posto, todos os trabalhos utilizaram um dispositivo IoT. Censi (2001) utlizou o Rabbit 2000 TCP/IP, Botke (2014) utilizou o Arduino Mega 2560, Gadotti (2010) ulilizou o FEZ Domino e neste trabalho foi utilizado o Raspberry Pi 3 Model B. Nos trabalhos correlatos não foi especificado o sistema operacional utilizado nos dispositivos IoT. De forma geral, estes dispositivos dispõem de softwares e bibliotecas disponibilizadas para facilitar a programação do processador. Já alguns utilizam um RTOS (Sistema Operacional de Tempo Real). Neste trabalho a utilização do sistema operacional Windows 10 IoT Core no RaspberryPi 3 Model B fez parte do objeto da pesquisa e o resultado obtido foi satisfatório.

Em cada trabalho correlato foi usada uma linguagem de programação diferente. Neste trabalho, além da linguagem C#, usada também no trabalho de Botke (2014), foi utilizado o Java. Todos os trabalhos, com a exceção de Botke (2014), implementaram a tecnologia socket para estabelecer a comunicação entre as aplicações via rede. Referente ao modo de comunicação para realizar o envio de comandos de controle, Censi (2001) usou o envio por E-mail, ao passo que Gadotti (2010) optou por postagem via Twitter, já no trabalho de Botke (2014) e neste trabalho foi usado o padrão Wi-Fi. O sistema gerenciador de todos os trabalhos correlatos foi desenvolvido para rodar na plataforma web. O software gerenciador foi implementado para a plataforma Android.

# 4 CONCLUSÕES

No decorrer do desenvolvimento deste trabalho foi realizada uma pesquisa sobre a utilização do Raspberry Pi 3 Model B com o Windows 10 IoT Core para a implementação de um sistema de automação residencial. As informações levantadas foram necessárias para alcançar os principais objetivos deste trabalho no âmbito do desenvolvimento de um protótipo. Foi construído um hardware baseado no Raspberry Pi 3 Model B e também foi desenvolvido um aplicativo gerenciador e um software embarcado no Raspberry Pi 3 Model B para rodar como servidor responsável por processar e executar as instruções recebidas do aplicativo gerenciador.

As técnicas e ferramentas utilizadas foram facilmente adaptadas de acordo com as necessidades do projeto. Em relação ao hardware e componentes utilizados, os sensores e atuadores funcionaram perfeitamente e supriram as necessidades do protótipo. O uso do Raspberry Pi 3 Model B foi simples e proveitoso. No que se refere ao software, A instalação do Windows 10 IoT Core no Raspberry Pi por intermédio do Windows IoT Core Dashboard foi um processo bastante simples. O Visual Studio ofereceu um ambiente de desenvolvimento amplamente integrado, notavelmente para a extensão da UWP e a utilização das bibliotecas úteis para o projeto, e também para a realização do *deploy* do software embarcado no Raspberry Pi.

O trabalho foi relevante por implementar uma forma de automatização utilizando o sistema operacional Windows instalado no Raspberry Pi. A utilização do Raspberry Pi 3 Model B com o Windows 10 IoT Core é indubitavelmente viável e pertinente, apesar das dificuldades encontradas por causa da pouca documentação. Assim como foi mencionado na análise dos resultados, os resultados obtidos com o protótipo foram de fato satisfatórios. Pode ser então concluído que o trabalho atingiu os objetivos propostos. É importante ressaltar que o custo do desenvolvimento do protótipo foi acessível.

## 4.1 EXTENSÕES

Algumas das extensões possíveis para este trabalho são:

a) permitir o controle utilizando a comunicação das aplicações em redes diferentes;
b) implementar o sistema de gerenciamento multiusuário;
c) integrar o monitoramento por câmera;
d) integrar o sensor de temperatura e umidade DHT22;
e) implementar um sistema de interfone de tal forma que ao tocar o interfone o aplicativo gerenciador receba uma chamada e permita a comunicação via VoIP.

# REFERÊNCIAS


ANGEL, Patricia Marta. **Introducción a la domótica**. Córdoba: EBAI, 1993.

BABU, T.Ram; LAXMIGANESH; GOWRISANKAR, A.. Iot for Self Monitoring Analysis Remote Transducers. **International Journal of Engineering and Technical Research (IJETR)**, Bobbili, v. 6, n. 3, p. 47-50, Nov. 2016.

BELL, Charles. **Windows 10 for the Internet of Things**. Warsaw: Apress, 2016. 467 p.

BOLZANI, Caio Augustus Morais. **Residências Inteligentes**. São Paulo: Livraria de Física, 2004.

BORYCKI, Dawid. **Programming for the Internet of Things**: Using Windows 10 IoT Core and Azure IoT Suite. Redmond: Microsoft Press, 2017.

BOTKE, Daniel Ponick. **Automação de Residências através de Aplicação integrada com Arduino**. 2014. 77 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

BUYYA, Rajkumar; DASTJERDI, Amir Vahid. **Internet of Things**: Principles and Paradigms. Cambridge: Elsevier, 2016.

CANA KIT. **Raspberry Pi 3 Complete Starter Kit - 32 GB Edition**. [S.l.], 2017. Disponível em: < https://www.canakit.com/raspberry-pi-3-starter-kit.html>. Acesso em: 26 nov. 2017.

CENSI, A. **Sistema para automação e controle residencial via e-mail**. 2001. 59 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

DENNIS, Andrew K. **Raspberry Pi Home Automation with Arduino**: Automate your home with a set of exciting projects for the Raspberry Pi!. Birmingham: Packt Publishing, 2013.

FERREIRA, João Alexandre Oliveira. **Interface homem-máquina para domótica baseado em tecnologias Web**. 2008. 74f. Dissertação (Mestrado Integrado em Engenharia Electrotécnica e de Computadores) - Curso de Pós-Graduação em Engenharia Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto, Porto.

FRENZEL, Louis E. Jr. **Fundamentos de Comunicação Eletrônica**: Modulação, Demodulação e Recepção. 3. ed. Porto Alegre: AMGH Editora, 2013.

GADOTTI, Eduardo Felippi. **Sistema para automação e controle residencial via Twitter**. 2010. 55 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KURNIAWAN, Agus. **Getting Started with Raspberry Pi 3**. Berlin: PE Press, 2016.

LEXINNOVA TECHNOLOGIES, LLC. **Internet of Things**: IoT Day Special. San Jose, 2015. Relatório.

MICROSOFT. **Windows IoT Core developer documentation**. [S.l.], 2017. Disponível em: <https://docs.microsoft.com/en-us/windows/iot-core/>. Acesso em: 28 maio 2017.

MICROSOFT. **Intro to the Universal Windows Platform**. [S.l.], 2017. Disponível em: <https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide#main>. Acesso em: 30 aut. 2017.

MIGUEL, Jose Otavio; HENNING, Thomas Cortiano; TURATTI, Matheus Vinicius Barcaro. **Automação Residencial**: Como acender e apagar lâmpadas pela internet. Curitiba, 2017. Relatório.

RASPBERRY PI FOUNDATION. **Downloads**. [S.l.], 2017. Disponível em: <https://www.raspberrypi.org/downloads>. Acesso em: 26 nov. 2017.

RASPBERRY PI FOUNDATION. **FAQS**. [S.l.], 2017. Disponível em: <https://www.raspberrypi.org/help/faqs>. Acesso em: 28 maio 2017.

RASPBERRY PI FOUNDATION. **Raspberry Pi 3 Model B**. [S.l.], 2017. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b>. Acesso em: 28 maio 2017.

ROSARIO, Joao Mauricio. **Automação industrial**. São Paulo: Baraúna, 2009.

SAKAGUCHI, André Osti. **Sistema de monitoramento e controle para residências com uso de software livre**. 2014. 92 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Departamento Acadêmico de Engenharia Elétrica, Universidade Federal do Paraná, UFPR. Curitiba.

TWITTER. **What is OAuth**. [S.l.], 2017. Disponível em: <http://dev.twitter.com/pages/oauth_faq>. Acesso em: 21 mar. 2017.

VERMESAN, Ovidiu; FRIESS, Peter. **Internet of Things**: Converging Technologies for Smart Environments and Integrated Ecosystems. Aalborg / Denmark: River Publishers, 1993.