CENTRO PAULA SOUZA
FATEC SANTO ANDRÉ

RONER RIVA BERGONCI

DESENVOLVIMENTO DE UMA METODOLOGIA PARA APLICAÇÃO DE AAL COM UTILIZAÇÃO EM AMBIENTES PARA SEGURANÇA RESIDENCIAL

SANTO ANDRÉ
2017

RONER RIVA BERGONCI

DESENVOLVIMENTO DE UMA METODOLOGIA PARA APLICAÇÃO DE AAL COM UTILIZAÇÃO EM AMBIENTES PARA SEGURANÇA RESIDENCIAL

Trabalho de Conclusão de Curso apresentado a Fatec Santo André como requisito parcial para obtenção do título de Tecnólogo em Mecatrônica Industrial.

Orientador: Prof. Me. Murilo Zanini de Carvalho.

SANTO ANDRÉ
2017

LISTA DE PRESENÇA

SANTO ANDRÉ, 13 de Dezembro de 2017

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA **"DESENVOLVIMENTO DE UMA METODOLOGIA PARA APLICAÇÃO DE AAL COM UTILIZAÇÃO EM AMBIENTES PARA SEGURANÇA RESIDENCIAL."** DO ALUNO DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. MURILO ZANINI DE CARVALHO _____

MEMBROS:
PROF. PAULO TETSUO HOASHI _____

PROF. FERNANDO GARUP DALBO _____

ALUNO:

Roner Riva Bergonci _____

FICHA CATALOGRÁFICA

B499d

Bergonci, Rone Riva

Desenvolvimento de uma metodologia para aplicação de AAL com utilização em ambientes para segurança residencial / Roner Riva Bergonci. - Santo André, 2017. – 73f: il.

Trabalho de Conclusão de Curso – FATEC Santo André. Curso de Tecnologia em Mecatrônica Industrial, 2017.

Orientador: Prof. Me. Murilo Zanini de Carvalho

1. Segurança residencial. 2. Internet das coisas. 3. Sistemas de segurança. 4. Tecnologia. 5. Prototipagem. I. Desenvolvimento de uma metodologia para aplicação de AAL com utilização em ambientes para segurança residencial.

629.89

Dedico este trabalho a todos os amigos, professores e familiares que apoiaram esta jornada e a todos aqueles que buscam mudar o mundo para melhor.

AGRADECIMENTOS

Agradeço a todo o corpo docente da FATEC Santo André por proporcionar conhecimento e habilidades de valor inestimável durante o curso de Mecatrônica Industrial

Agradeço ao Orientador Prof. Me. Murilo Zanini de Carvalho por seu incentivo e apoio durante todo o percurso.

Agradeço a todos aqueles que de alguma forma contribuíram para que este projeto se tornasse uma realidade.

RESUMO

O crescimento de tecnologias destinadas à auxiliarem o usuário em sua residência vem se mostrando cada vez mais forte principalmente devido ao aumento da expectativa de vida dos seres humanos. Outra área que apresenta grandes avanços é a de *Machine Learning*, possibilitando resolver problemas cujas aproximações lógicas tradicionais não são capazes de cobrir todos os casos de alguns problemas. Um destes problemas é a criação de um sistema de segurança residencial que seja capaz de proteger o usuário em qualquer cenário independentemente das circunstâncias em que o mesmo se encontra. Este trabalho foi elaborado visando suprir essa demanda, podendo assim proporcionar um ambiente mais seguro para se viver. Os métodos de pesquisa e prototipagem resultaram na verificação de viabilidade da proposta e na observação de possíveis estudos mais aprofundados.

Palavras-chave: Internet das Coisas, *Machine Learning*, Segurança residencial, *Ambient Assisted Living*, Raspberry Pi.

ABSTRACT

The growth of technologies designed to assist the user in their residence has been increasing mainly due to the rise in the life expectancy of human beings. Another area that presents great advances is the Machine Learning, which allows solving problems whose traditional logical approaches are not able to cover all the cases in some problems. One of these problems is the creation of a home security system that is capable of protecting the user in any scenario regardless of the circumstances. This project was developed to meet this demand, providing a safer environment to live. The methods of research and prototyping have resulted in the verification of feasibility of the proposal and the observation of possible further studies.

Key-words: Internet of Things, Machine Learning, Home security, Ambient Assisted Living, Raspberry Pi.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AAL	*Ambient Assisted Living*
IoT	*Internet of Things*
OOS	*Open Source Software*
GPIO	*General-purpose input/output*
IDE	*Integrated Development Environment*
GHz	*Gigahertz*
USB	*Universal Serial Bus*
HDMI	*High-Definition Multimedia Interface*
SD	*Secure Digital*
LAN	*Local Area Network*
BLE	*Bluetooth Low Energy*
MQTT	*Message Queue Telemetry Transport*
TI	Tecnologia da Informação
AWS	Amazon Web Services
App	*Application*
MIT	*Massachusetts Institute of Technology*
URL	*Uniform Resource Locator*
JSON	*JavaScript Object Notation*

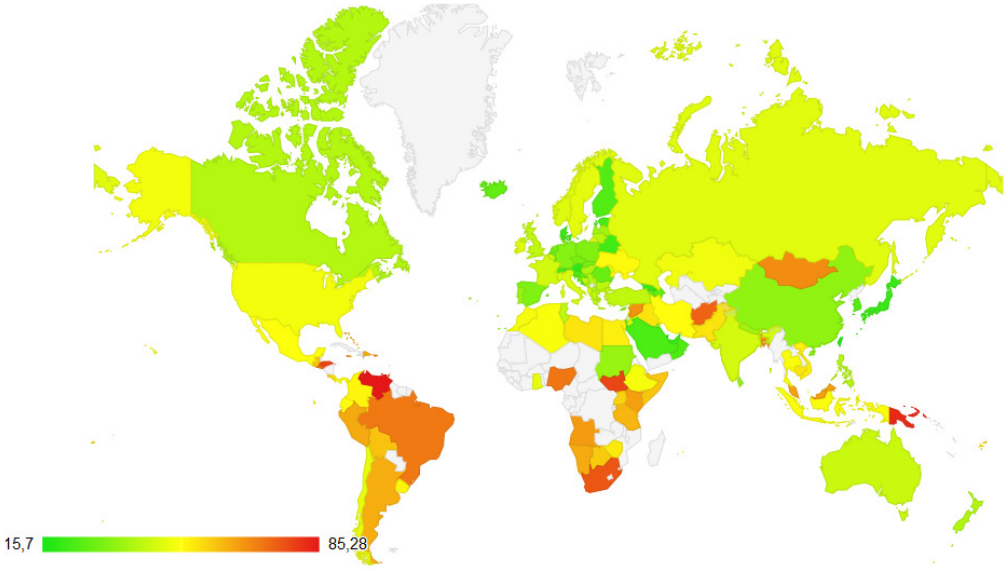
SUMÁRIO

1 Introdução

Nessa era onde mesmo a evolução da tecnologia seguindo de forma cada vez mais acelerada nos proporcionando quantidades de confortos que jamais foram possíveis em épocas passadas, é ainda demasiadamente grande a quantidade de problemas com qual a sociedade precisa lidar. Evitando entrar em uma área muito abrangente sobre este assunto, foquemos no tema principal que se trata sobre os desafios e perigos apresentados na segurança domiciliar.

A preocupação e a sensação de segurança do cidadão variam devido a diversos fatores tais como o local onde ele reside, as leis que governam este local, taxa de criminalidade, atividade policial. Como observado no mapa a seguir (Figura 1) contendo dados do índice de criminalidade ao redor do mundo onde os números mais elevados indicam os países com mais crimes.

Figura 1 – Índice de criminalidade no mundo



Fonte: (Numbeo, 2017)

Mais detalhe sobre o cálculo deste índice pode ser encontrado na página https://www.numbeo.com/crime/indices_explained.jsp. Este trabalho propôs um sistema totalmente autônomo para que seja possível amenizar tais estatísticas.

1.1 Problema

Sistemas de segurança tradicionais tais como trancas e cercas elétricas já se mostraram incapazes de proteger os usuários de uma residência contra invasões.

Muitas soluções que fazem uso dos avanços tecnológicos foram criadas tornando os sistemas de segurança cada vez mais inteligentes e eficazes, como por exemplo, “assistentes de segurança” proporcionam um nível de segurança muito maior quando comparado aos sistemas tradicionais. Essa são funções encontradas em alguns deles:

- Sensores de movimento;
- Monitoramento por câmeras;
- Reconhecimento facial;
- Ativar/Desativar o sistema de forma programada;
- Notificações de emergência;
- Reconhecimento de voz;
- Opções de privacidade;
- Armazenamento local e/ou nuvem;
- Bateria para autonomia;
- Comunidades de usuários.

No entanto, mesmo todas essas funções juntas não são capazes de proteger contra todos os tipos de invasões. Consideremos a situação onde o invasor conseguiu render o usuário e ele não consegue se mover em hipótese alguma sem colocar em risco a sua integridade. Começamos a perceber que as formas que estes dispositivos utilizam para proteger o usuário se tornam ineficazes. Dado este cenário, foram observados alguns pontos que chamaram nossa atenção:

- **Ativar/Desativar o sistema de forma programada:** Apesar de deixar o alarme ligado na maioria do tempo, isso ainda deixa aberto uma brecha na segurança, o invasor pode ter observado a rotina do usuário para rendê-lo no momento em que o mesmo está entrando em casa, dessa forma o sistema de segurança poderá ser ultrapassado.

- **Opções de privacidade:** Algumas pessoas se sentem desconfortáveis por estarem sendo observadas por câmeras o tempo todo e a opção de desativar ou não utilizar esse sistema em determinados momentos pode ser confortável para o usuário, porém, isso abre uma brecha para os invasores por determinado tempo. Outro problema que temos que considerar é a possibilidade de uma invasão no sistema, o que pode fornecer para o invasor, informações cruciais sobre a rotina do usuário devido as imagens coletadas.

- **Notificações de emergência:** Esse é o maior problema que desejamos eliminar através deste trabalho. Quando o sistema de segurança detecta alguma anormalidade que possa estar colocando o usuário de alguma forma em um potencial risco, o mesmo informa ao usuário e/ou outras pessoas de que alguma coisa está errada, possibilitando que o indivíduo cujo receber o alerta possa tomar decisões apropriadas. Em uma situação onde o usuário e outras pessoas se encontram incapaz de reagir, tais informações se tornam ineficazes.

1.2 Objetivos

Foram encontradas duas maneiras impactantes onde a segurança poderá ser rompida, abusando do sistema de acionamento do alarme, ou da ineficiência do procedimento de monitoramento.

As soluções propostas têm como premissa de que o usuário estará em situações tais como até mesmo onde seu corpo e membros estarão amarrados e sua boca atada.

Para solucionar os problemas citados acima foram desenvolvidas formas de eliminar tais possibilidades, ou ao menos, reduzi-las consideravelmente a fim de efetivamente proteger o usuário contra diversas situações de risco. A possibilidade de o sistema de segurança ficar desligado devido a alguma situação específica foi anulada, o mesmo permanece ativo o tempo todo. Já quanto a rendição e a incapacidade de tomar providências perante a situação foram implementados:

- Algoritmos capazes de determinar quando o usuário se encontra em uma situação de risco mesmo sem sua interação;
- Capacidade de o usuário conseguir enviar sinais de socorro sem o levantamento da suspeita do invasor.

Tivemos como objetivo o desenvolvimento de um sistema no qual seria possível verificar o resultado da aplicação dos conhecimentos e métodos definidos nas soluções propostas para que assim pudéssemos concluir se existe a possibilidade de aplicação em um sistema real.

1.3 Justificativa

A criminalidade prejudica e ameaça a vida de pessoas inocentes que em muitos casos não podem fazer nada além de assistir seus esforços, bens materiais e até mesmo entes queridos, sendo injustamente retirados de suas vidas.

Tendo como sonho poder proporcionar melhorias nas condições de vida da humanidade como um todo, almejo que este trabalho seja útil para criar uma maior sensação de segurança para o máximo de pessoas possíveis e que os estudos sobres os assuntos aqui levantados sirvam de base para novas ideias ainda melhores e mais eficazes voltadas ao convívio em sociedade.

1.4 Organização do trabalho

No capítulo 2 é informado todos os conceitos que foram necessários como fundamentação teórica para a realização do projeto. No capítulo 3 são citados quais foram os métodos utilizados durante a etapa de pesquisa. No capítulo 4 é explicado quais foram as ações tomadas para tornar a proposta em algo real. No capítulo 5 são exibidos os resultados obtidos. No capítulo 6 contam as considerações finais e sugestões para o desenvolvimento de trabalhos futuros.

2 Fundamentação Teórica

Ao longo deste capítulo foram apresentados os resultados das pesquisas feitas com os trabalhos dos autores mais renomados da área de AAL, IOT e de segurança.

2.1 Segurança

Uma casa inteligente voltada a segurança deve ser capaz de proteger seus moradores de ameaças externas e internas, essa proteção é feita com uso de vários sensores capazes de detectarem determinadas situações de perigo e auxiliar na tomada de decisões automaticamente. (Md, G, G, DharmaSavarini, & Mittal, 2016)

As tecnologias envolvidas na segurança doméstica evoluíram nos últimos tempos e devem continuar evoluindo. Proporcionando maiores funcionalidades, mais conforto e menos preocupações. Essas tecnologias consistem de dispositivos eletrônicos capazes de se comunicarem e preverem diversas situações de perigos à serem tratadas.

Para que uma casa consiga ser capaz de atuar perante uma situação de risco é necessário que a mesma consiga perceber tal situação. Devido a este fato, é de extrema importância que a quantidade de dados observados pelos equipamentos seja suficiente para que através de uma lógica bem desenvolvida ocorra a tomada correta de decisão pelo sistema.

2.1.1 Situações

A seguir seguem situações onde o usuário pode se encontrar em risco.

- **Rendições:** É comum encontrar sistemas de segurança providos de processos de autenticação de usuário para que ocorra a liberação de uma porta ou algum outro dispositivo, sendo bastante eficaz contra pessoas desconhecidas que tentarem o acesso e até mesmo sendo possível enviar alguma reação em caso de tentativa indevida. Uma das táticas que acabam

sendo efetuadas contra sistemas de segurança é não permitir que o usuário tenha a possibilidade de usufruir do mesmo. Supondo que uma pessoa possua uma casa equipada com trava eletrônica na porta, ou leitor de impressão digital, informações muito difíceis de serem adquiridas. Agora essa mesma pessoa ao entrar ou sair com seu carro de sua garagem foi abordada por um assaltante que ao ameaçar a sua vida, não deixa escolha para a vítima além de ter que destravar a segurança do que for ordenado. Todo um sistema de segurança se tornou ineficaz pois não foi capaz de proteger o usuário no momento inicial da ação.

Exemplo de ação similar em: http://liberal.com.br/cidades/americana/quadrilha-amordaca-medicos-e-faz-limpa-em-residencia-607809/

- **Sistema desabilitado:** Independentemente da quantidade de sensores e métodos de segurança colocados em uma residência, eles não serão capazes de proteger o usuário se estiverem desativados, sejam eles por falta de recursos ou por configurações do sistema. Um criminoso que possua de alguma forma informações privilegiadas sobre o período de funcionamento do sistema possuirá uma grande ferramenta em suas mãos.

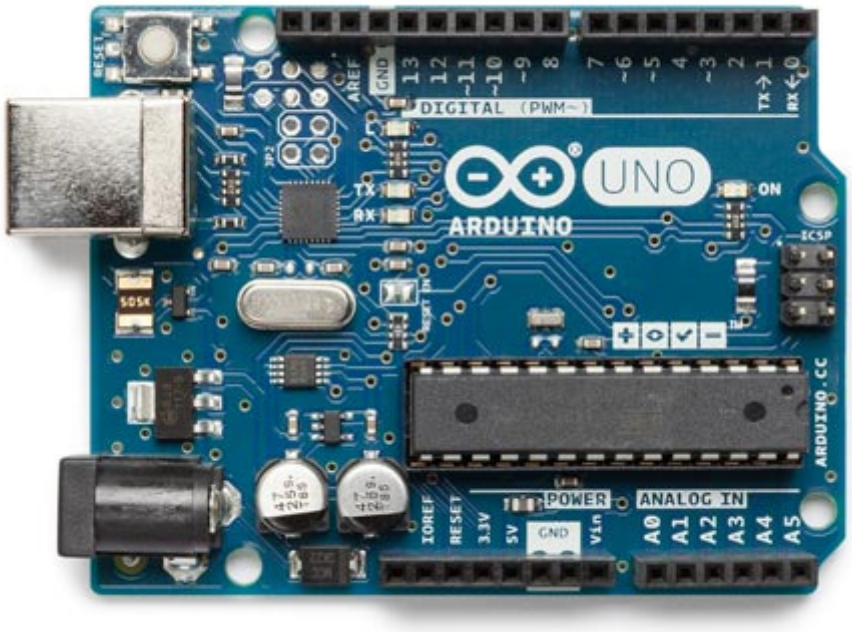
2.2 Aquisição de dados

Para fazer o uso de informações do ambiente residencial é necessário que dispositivos sejam capazes de ler e/ou escrever estas informações em outros dispositivos para seu tratamento posterior.

2.2.1 Arduino

O Arduino (Figura 2) é uma placa eletrônica OSS projetada para ser de fácil aprendizado, inicialmente era destinada a estudantes e pessoas com poucos conhecimentos nas áreas de eletrônica e programação, no entanto com o aumento da popularidade, as placas foram se adaptando as necessidades e cada vez mais ganhando novas funções. (Arduino, 2017)

Figura 2 – Arduino Uno R3



Fonte: (Arduino, 2017)

Com o Arduino é possível ler variáveis de sensores, mandar um sinal para algum atuador, realizar comunicação Serial com outros dispositivos, ou tarefas quais muitas vezes possuem bibliotecas. (Arduino, 2017)

2.2.2 Esp8266

O Esp8266 é um modulo Wi-Fi adequado para aplicações IoT – Internet of Things, cujo conceito será abordado no capítulo 2.7, caracterizado por possuir baixo consumo e custo, que pode ser facilmente acoplado a dispositivos externos e reprogramado. (Kodali & Mahesh, 2016) (Kodali & Soratkal, 2016)

Existem diversos modelos de Esp8266, normalmente a diferença se encontra no número de pinos GPIO, como a caraterística mais importante do projeto é a conectividade o modelo usado em cada aplicação pode variar de acordo com as necessidades da residência.

A placa Wemos (Figura 3) é um módulo WiFi para IoT baseado no ESP-8266EX semelhante ao Arduino UNO. Pode ser programada com a IDE do Arduino, através de uma porta micro USB. (Kodali & Sahu, An IoT based Weather Information Prototype Using WeMos, 2016)

Figura 3 - Wemos D1

Fonte: Autoria própria

2.3 Raspberry Pi

O Rapsberry Pi 3 (Figura 4) é um computador montado em uma única placa que tem como intuito proporcionar a estudantes e educadores a possibilidade de desenvolver conhecimentos envolvendo tecnologias computacionais. (Rao & Uma, 2015)

Figura 4 - Rapsberry Pi 3 Model B

Fonte: (Raspberry Pi Foundation, s.d.)

Características:

- Processador de 1.2GHz 64-bit quad-core ARMv8
- 1GB RAM, 4 portas USB
- 40 pinos GPIO, porta full HDMI
- Porta Ethernet
- Saída de video e audio composto
- Interface de câmera
- Interface de display
- Slot para cartão Micro SD
- Núcleo gráfico VideoCore IV 3D
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)

(Raspberry Pi Foundation, s.d.)

Dispositivos de automação presentes nos dias atuais fazem normalmente uso de micro controladores ou de um computador. Sistemas micro controlados não são capazes de lidar de forma eficaz com múltiplas tarefas, como por exemplo, aplicações em alguma atividade da casa e monitoramento simultâneo do ambiente. Já um computador convencional não apresenta tais dificuldades, porém, na maioria dos casos se torna um equipamento superdimensionado aumentando os custos.

O Raspberry Pi pode realizar tarefas simultâneas e de complexidade relativamente elevadas, eliminando assim as limitações encontradas em sistemas micro controlados, com a vantagem de poder ser adquirido por preço mais acessível do que se utilizando um computador convencional, evitando o uso de um dispositivo muito potente para tarefas que requisitem menos poder computacional. Por este motivo pode-se baratear o projeto, assim como reduzir o espaço físico gasto pelo mesmo. (Patchava, Kandala, & Babu, 2015)

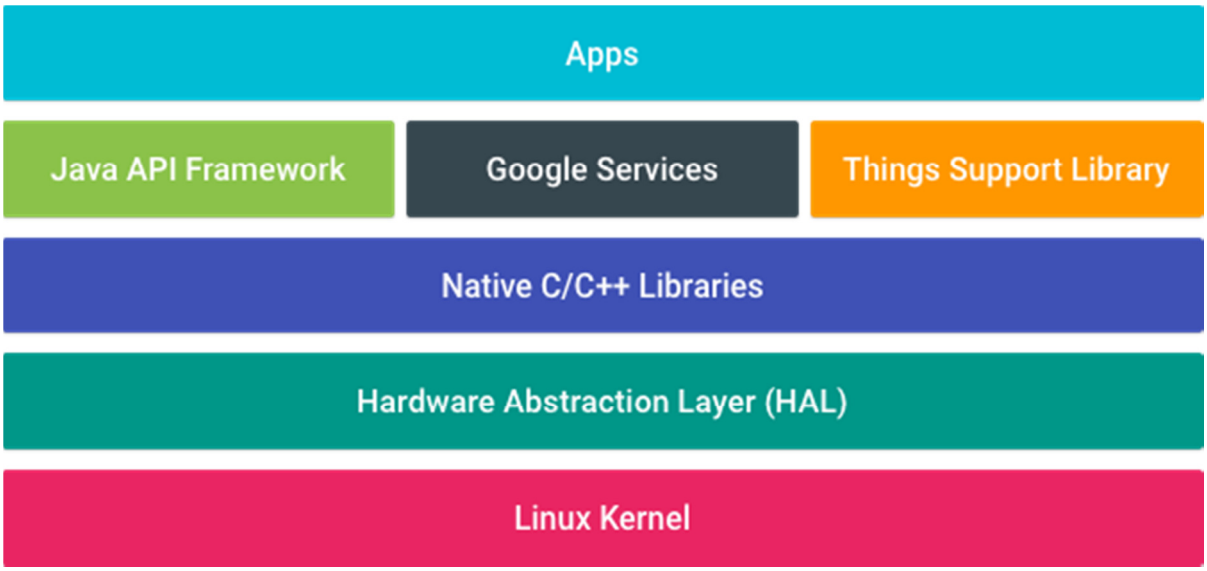
Ele pode ser utilizado em aplicações residenciais para disponibilizar ao usuário a capacidade de monitorar e controlar dispositivos tais como portas, iluminações, temperatura e câmeras de segurança. (Rao & Uma, 2015)

2.4 Android Things

O Android Things é um sistema operacional desenvolvido pela Google com o objetivo de aproximar seus usuários a sistema embarcados sem necessidade de conhecimento adicionais. (Google Inc., 2017)

O Android Things é caracterizado por possuir semelhanças ao desenvolvimento de aplicações para dispositivos móveis equipados portadores do Android (Figura 5), sendo necessário apenas uma das placas compatíveis com o sistema, (Google Inc., 2017)

Figura 5 - Ferramentas do AndroidThigns



Fonte: (Google Inc., 2017)

Aplicativos desenvolvidos tendo como alvos dispositivos embarcados possuem uma maior aproximação com hardwares periféricos do que dispositivos móveis, essa aproximação é disponibilizada com a integração da Things Support Library as ferramentas do Android, um documento contendo as maiores diferenas encontradas entre o Android Things e o Android pode ser conferida em https://developer.android.com/things/sdk/index.html.

2.5 MIT App inventor

O MIT App Inventor é um ambiente de programação de aplicativos Android utilizando uma linguagem visual baseada em blocos, possibilitando um rápido aprendizado a pessoas iniciantes e o desenvolvimento de programas complexos em menor tempo quando comparados a ambientes de programação tradicionais. Utilizado em 195 países por pessoas de diversas níveis de conhecimento, sendo que 50% das mesmas não possuem o ensino direto por escolas ou outros meios educação.

Um projeto desenvolvido no App Inventor é composto de um conjunto de componentes presentes no sistema Android, cuja funcionalidade de cada é determinada através de blocos. (Benjamin Xie, 2016)

2.5 MQTT

Inventado por Dr. Andy Stanford-Clark da IBM - *International Business Machines Corporation* e Arlen Nipper da Arcom (atualmente Eurotech) em 1999, o MQTT – *Message Queue Telemetry Transport* é um protocolo para transmissão de mensagens caracterizado por ser extremamente leve e simples, utilizado em áreas de IoT e comunicação entre máquinas. Foi desenvolvido principalmente visando ser utilizado em dispositivos que não possuem uma alta conectividade com a internet ou quando não se deseja consumir uma porcentagem significativa do pacote de dados fornecido pelas provedoras, minimizando os gastos de rede enquanto fornece uma confiabilidade considerável em vista de seus baixos gastos. (MQTT.org, 2017)

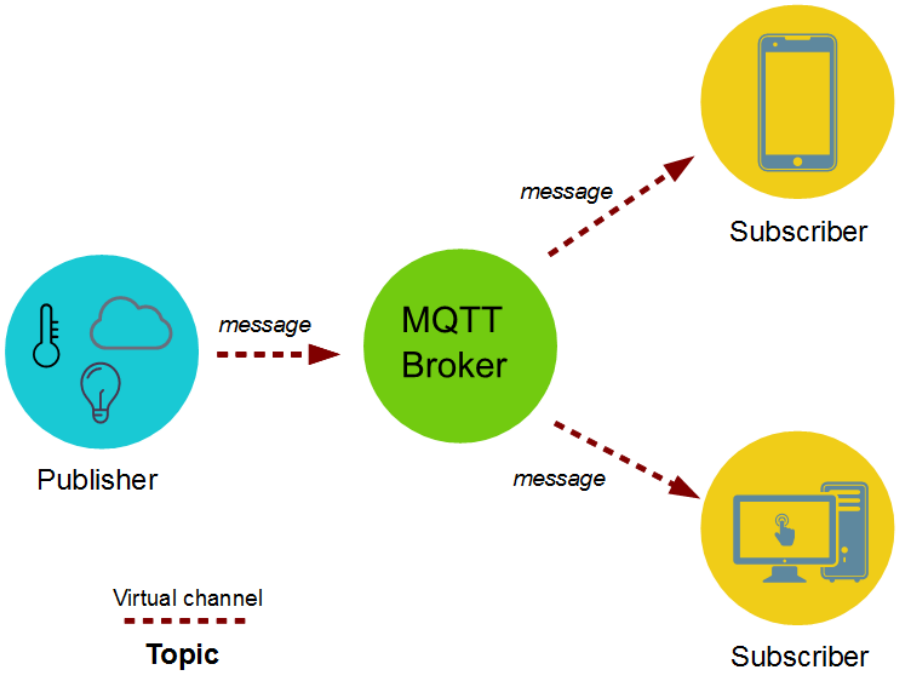
2.5.1 *Publish/Subscribe/Broker/Topic*

O funcionamento do protocolo MQTT pode ser explicado da seguinte maneira, imagine que existem diversos dispositivos com acesso à uma rede onde um desses dispositivos é o nosso centralizador (*broker*) das mensagens que serão transmitidas.

Cada um desses dispositivos poderá estar inscrito (*subscriber*) ou publicar (*publisher*) em um tópico (*topic*). Tópicos são como canais privados de comunicação, quando uma mensagem é publicada neste canal, apenas os dispositivos

previamente inscritos nesse canal serão capazes de receber a mensagem transmitida. A mensagem enviada pelo *publisher* chega ao *broker* que repassa a mesma para todos os dispositivos *subscribers* daquele tópico onde a mensagem foi enviada. (Kodali & Soratkal, MQTT based Home Automation System Using Esp8266, 2016) (Kodali & Sahu, An IoT based Weather Information Prototype Using WeMos, 2016)

Figura 6- Comunicação com protocolo MQTT

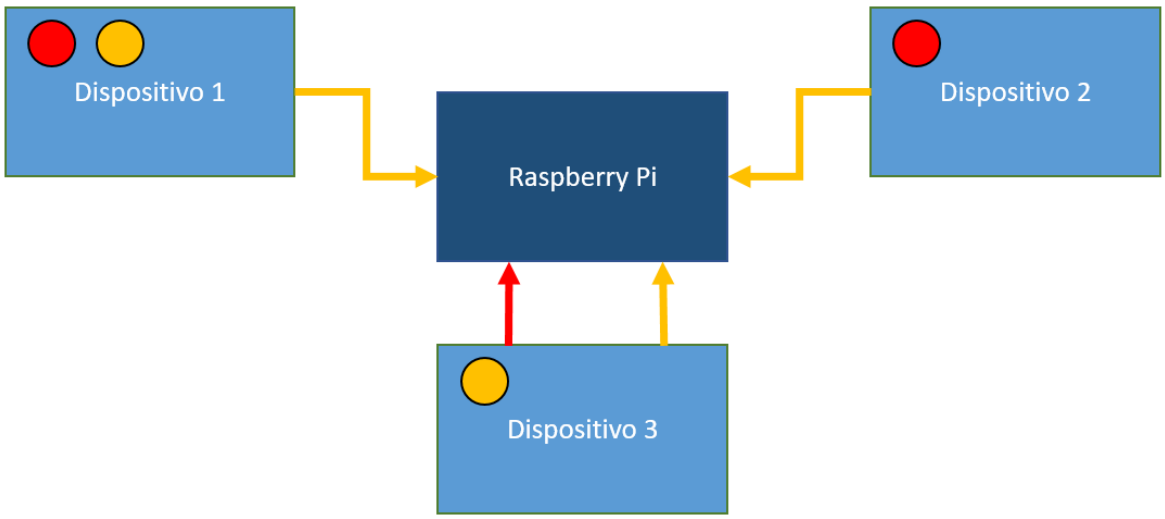


Fonte: (Azzola, 2016)

Exemplificando: Consideremos 3 dispositivos conectados via wifi à um Raspberry Pi que será o nosso broker, comunicando através de 2 canais: tópicos A - vermelho e B -- laranja (Figura 7).

- Dispositivo 1 – Inscrito nos tópicos A e B e Publisher no tópico B;
- Dispositivo 2 – Inscrito no tópico A e Publisher no tópico B;
- Dispositivo 3 – Inscrito no tópico B e Publisher nos tópicos A e B;

Figura 7 - Exemplificação de comunicação MQTT



Fonte: Autoria própria

O único dispositivo que pode enviar mensagens pelo tópico A é o número 3, sua mensagem será ouvida por 1 e 2;

Os dispositivos que podem enviar mensagens pelo tópico B são os números 1, 2 e 3, sua mensagem será ouvida por 1 e 3;

Repare que uma mensagem enviada por um dispositivo em determinado tópico pode ser recebida por ele mesmo caso o mesmo esteja inscrito neste tópico. (Kodali & Soratkal, 2016) (Kodali & Sahu, 2016)

Com a finalidade de tornar mais familiar alguns termos utilizados no restante do trabalho, foram adicionados alguns tópicos com alguns conceitos utilizados para o monitoramento e assistência de residências.

2.6 AAL

Ambiente Assisted Living (AAL) consiste na aplicação de tecnologias capazes de auxiliar pessoas em suas rotinas, mantendo autonomia, segurança e conforto no ambiente em que vivem. (Dohr, R.Modre-Osprian, Drobics, Hayn, & G.Schreier, 2010) (Sernani, Claudi, Palazzo, Dolcini, & Dragoni, 2013)

Figura 8 – Exemplo de equipamentos de um ambiente AAL

Fonte: (SecurityOne, 2016)

Ao discorrer sobre AAL nos referimos a um ambiente inteligente, para tal ambientação são necessários diversos dispositivos providos de recursos muito utilizados em aplicações de IoT, para funções como, monitoramento, comunicação e auxílio direto as pessoas (Figura8), o que deixa nítido a relação presente entre os dois temas. (Dohr, R.Modre-Osprian, Drobics, Hayn, & G.Schreier, 2010)

Uma etapa muito importante para um sistema de AAL é sua coleta de dados, envolvendo conceitos muitas vezes utilizados em áreas de Big Data, uma vez visto que os mesmos precisam lidar com o armazenamento e transporte de dados de tipos variados em determinada quantidade e com tempos de respostas que também variam de acordo com a aplicação. (Spinsante, Gamibi, Montanini, & Raffaeli, 2015)

A necessidade de tais aplicações vem crescendo principalmente devido ao fato de que ao longo do tempo existe a tendência de aumento na expectativa de vida e a redução da taxa de natalidade. (Dohr, R.Modre-Osprian, Drobics, Hayn, & G.Schreier, 2010)

Isso torna o AAL altamente indicado para realizar o cuidado de pessoas idosas que devido a forma como a sociedade os encara tendem a optar pela isolação e acabam vivendo sozinhos, dessa forma familiares e médicos podem ter um cesso mais simples a informações importantes. (Zaric, Djurisic, & Mihovska, 2016)

Devido a necessidade de suprir as demandas de cada usuário os sistemas de AAL devem ser capazes de se adaptarem à diferentes tipos de condições. (Sernani, Claudi, Palazzo, Dolcini, & Dragoni, 2013)

Um exemplo da aplicação da tecnologia é demonstrado em um tratamento de pacientes portadores de diabetes, o tratamento indica a dose de medicamento recomendado através de uma tabela, o que deixa faixas de valores muito amplas, ao monitorarmos os dados individuais podemos ajustar as doses com base nos mesmos diminuindo os casos de dosagem excessiva ou insuficiente. (Vara & Skarmeta, 2010) (Bygholm & Kanstrup, 2015)

AAL é uma tecnologia que se encontra em desenvolvimento muito grande atualmente, tornando-se difícil encontrar uma definição em comum capaz de abranger todo seu escopo. (Bygholm & Kanstrup, 2015)

Apesar de o termo AAL estar muito relacionado com aplicações destinadas ao auxílio nos cuidados com a saúde de pessoas portadoras de maiores necessidades, seu conceito de ambientação capaz de prover melhores condições para aqueles presentes no mesmo pode ser aplicado visando criar um ambiente capaz de prover segurança aos usuários.

2.7 IoT – Internet of Things

Internet of Things é um conceito que trata capacidade de objetos inteligentes se interagirem entre si formando assim uma rede de comunicação (Figura 9), um fenômeno decorrente dos avanços tecnológicos envolvendo conceitos de informação e comunicação tais como: capacidade de dispositivos se comunicarem tanto com pessoas como entre outros dispositivos, construindo-se assim uma rede de coisas; aumento do poder de processamento dos dispositivos envolvidos; capacidade destes objetos de sentirem ou atuarem em diversas situações. (Dohr, R.Modre-Osprian, Drobics, Hayn, & G.Schreier, 2010)

Figura 9 - Rede de alguns dispositivos IoT

Fonte: (Singh, 2016)

Objetos comuns passam a estar conectados à internet aproximando o mundo virtual do mundo físico, consequentemente proporcionando maiores interações. O objetivo principal ao se fazer um sistema de *Internet of Things* é adquirir apenas informações relevantes ao processo e aplica-los somente onde são necessários para aproximar estes dois mundos. (Dohr, R.Modre-Osprian, Drobics, Hayn, & G.Schreier, 2010)

2.8 Nuvem

A computação em nuvem consiste no fornecimento de poder computacional, armazenamento, entre outras funcionalidades de TI baseado na demanda de sua aplicação.

Supondo que uma aplicação será fornecida para atender clientes estimados em aproximadamente 1000 pessoas. Caso seu *hardware* seja capaz de atender até 1200 pessoas, mas apenas 800 fazem utilização do serviço, 200 de capacidade se tornaram prejuízos no investimento. Por outro lado, caso seu *hardware* seja capaz de atender até 1000 pessoas e 1200 desejam utilizar o serviço, haverá uma queda na qualidade e consequentemente perda de público. Tornar a capacidade

computacional de sua aplicação variável em função da demanda de sua aplicação, poupa gastos desnecessários com erros de dimensionamento.

Uma outra vantagem adquirida é a velocidade com a qual se pode realizar alterações na capacidade de atendimento da aplicação. Em um sistema utilizando *hardware* seria necessário a manutenção dos equipamentos e/ou aquisição de novos equipamentos capazes de suportar a nova demanda. Com a computação em nuvem, essa alteração pode ser realizada em poucos minutos alterando as configurações de sua aplicação. (Amazon Web Services, 2017)

A seguir seguem informações de dois sistemas de nuvem, sendo um deles apenas para armazenamento de informações e outro para utilização de serviços de *Machine Learning*.

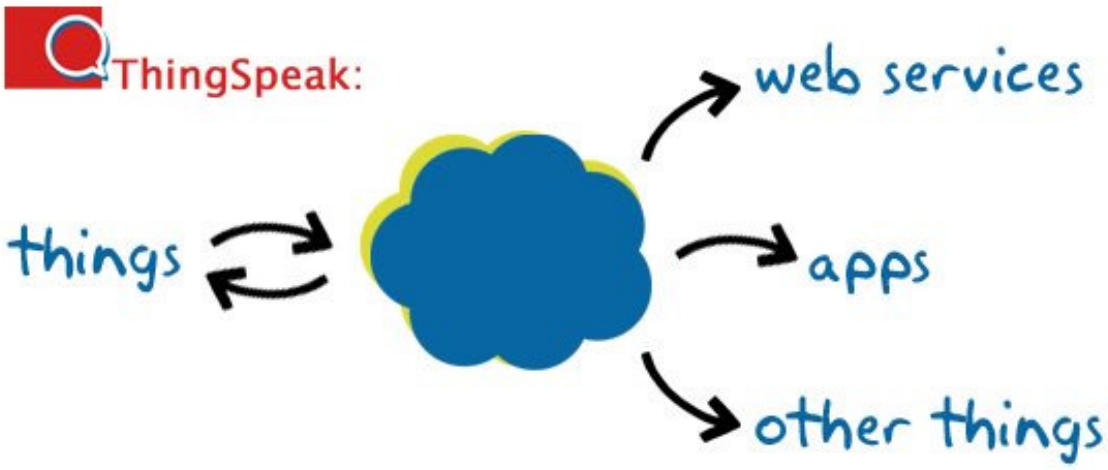
2.8.1 ThingSpeak

O ThingSpeak é um sistema de nuvem para aplicações IoT capaz de possibilitar que os dados enviados por seus dispositivos sejam em pouco tempo visualizadas online estando disponível para serem analisados ou utilizados em outras aplicações (Figura 10).

Com o Thingspeak é possível configurar o envio de dados através de protocolos IoT conhecidos eliminando a necessidades de criar servidores ou *softwares* web. O envio de dados assim como a possibilidade de agir de acordo com os mesmos em tempo real torna o ThingSpeak uma ferramenta de análise de dados simples e eficaz.

Sensores em nossos equipamentos nos entregam respostas, sejam em valores numéricos ou sinais elétricos, uma vez que esses dados estejam armazenados na nuvem a amplitude de onde tais informações podem ser aplicadas deixa de ser o dispositivo local e passa a ser qualquer equipamento com conexão à internet. (The MathWorks, Inc, 2017)

Figura 10 - Funcionamento da nuvem



Fonte: (Scharler, 2011)

O armazenamento de dados permite a possibilidade de descobrir padrões, relações e tendências nos dados, onde através de tais informações armazenadas em canais que podem ser tanto privados como públicos, gerando gráficos para análises. Esse armazenamento de dados pode ser definido para ocorrer apenas em determinado tempo, evitando leituras e consumo de energia desnecessária.

As ações que podem ser tomadas em resposta à coleta de dados podem variar desde posts no Twitter informando a situação da leitura até acionamento de outros equipamentos. (The MathWorks, Inc, 2017)

2.8.2 Amazon AWS

A Amazon Web Services (AWS) oferece um amplo conjunto de soluções em nuvem tais como armazenamento, computação, serviços de IoT, *Big Data, Machine Learning* entre diversos outros, baixando custos nas áreas de TI e proporcionando maior flexibilidade na ampliação da aplicação. (Amazon Web Services, 2017)

Entre um dos serviços disponibilizados está o Amazon *Machine Learning* que permite o uso de aprendizado de máquina de forma simples evitando consumo de tempo com implementações de algoritmos, além de proporcionar auxílios visuais para auxiliar na construção da aplicação. Utiliza os mesmos modelos utilizados pela comunidade de cientistas da Amazon para gerar previsões e gerenciar infraestruturas. (Amazon Web Services, 2017)

2.9 Machine Learning

Na época atual é possível encontrar uma enorme quantidade de dados estruturados ou não. O *Machine Learning* vem se desenvolvendo como uma subcategoria da Inteligência Artificial que envolve o desenvolvimento de algoritmos para auto aprendizado para gerar previsões através do conhecimento extraído de determinados dados, eliminando-se a necessidade de esforços humanos para definir regras para estes dados e adquirir novas informações. Através de sistemas capazes de encontrar padrões em dados e aplica-los podemos usufruir de sistemas como o filtro de e-mails e ou simuladores de xadrez entre muitos outros existentes e possivelmente ainda mais recursos estão por vir. (Raschka, 2015)

De forma mais simplificada, *Machine Learning* pode ser definido como a capacidade de fazer com que máquinas sejam capazes de aprender a lidar com tarefas por conta própria. Este ensinamento é realizado ao prover exemplos do que fazer e o que não fazer, estes exemplos serão usados como parâmetros para encontrar soluções. Muitas vezes estas soluções encontradas serão imperfeitas, alguns casos serão perdidos enquanto outros que não deveriam existir irão aparecer, tornando-se um processo que leva um certo tempo para ser aperfeiçoado. (Coelho & Richert, Building Machine Learning Systems with Python, 2015)

Existem três tipos distintos de *Machine Learning*, sendo eles: (Raschka, 2015)

- Aprendizado supervisionado: o sistema já possui as respostas corretas que desejamos alcançar e verificamos se as respostas condizem com o esperado;
- Aprendizado não supervisionado: o sistema não possui as respostas corretas;
- Aprendizado reforçado: o sistema já possui as respostas corretas que desejamos alcançar, porém diferentemente do aprendizado supervisionado, o resultado dado é uma porcentagem de sucesso em relação ao desejado.

Após uma comparação realizada entre técnicas de *Machine Learning* para reconhecimento de atividade com aquisição dados de sensores em um ambiente residencial, foi constatado que o modelo *Meta-Layer Network* obteve resultados melhores do que modelos baseados em memória. (Fan, Zhang, Leung, & Miao, 2016).

3 Metodologia

A seguir foram descritos os procedimentos realizados para confecção deste projeto. Os métodos utilizados tiveram como foco verificar a viabilidade de comunicação entre os dispositivos, assim como a aplicação de uma rede neural tratando um caso simples para que futuramente seja possível ampliar seu escopo de utilização para situações ainda mais complexas e com melhores resultados.

3.1 Soluções propostas para cada situação

Tendo como alvo as situações apresentadas no capítulo 2.2.1 e as ferramentas abordadas nos capítulos anteriores a este, foram pensadas em possíveis soluções para contribuir com a segurança do usuário.

3.1.1 Solução (Sistema insuficiente):

Uma vez assumido que o criminoso conseguiu entrar em sua residência sem que houvesse nenhum tipo de detecção por parte de sistemas de segurança anteriores, será necessário que a sua residência perceba algo estranho ocorrendo dentro de si mesma. Muitos dados poderão ser comparados com dados armazenados previamente, como por exemplo:

- Número de pessoas na residência não corresponde com o previsto;
- Movimentação de pessoas dentro da casa muito fora do normal;
- Acionamento de dispositivos não previstos;

Em alguns lugares existem lojas que possuem botões de emergência para o caso de assaltos que podem ser pressionados discretamente, geralmente tais botões se encontram camuflados em algum lugar em baixo do balcão que ao serem pressionados, acionam as autoridades alertando-as sobre o assalto. Para criar um “botão” que seja capaz de informar as autoridades e ser pressionado em qualquer situação precisamos considerar os tipos de situações mais incapacitantes possíveis ao usuário.

Ao detectar uma anomalia nos quesitos acima o sistema foi dotado da capacidade de solicitar ao usuário uma confirmação de que o mesmo está bem, essa ação também é bastante flexível, por exemplo:

- Acionar a lâmpada de um cômodo específico;
- Um comando de voz;
- Se dirigir até um local;

A requisição de uma ação por parte do usuário não poderá ser notada pelo criminoso, uma vez que o mesmo pode se enfurecer perante o ocorrido, colocando ainda mais em risco a vida do usuário. Tendo isso em mente, algum sinal sonoro ou luminoso é a forma mais discreta para se informar a necessidade de confirmação, caso o usuário não responda em um tempo pré-determinado poderá ser acionado o sistema de emergência que tomará as providências necessárias.

3.1.2 Solução (Sistema desabilitado):

Não existiram casos onde o sistema ficará desativado, o mesmo permanece ligado para sempre após seu funcionamento. A ausência de fornecimento de energia poderia derrubar todo o sistema de segurança, portanto a aplicação de um nobreak para eventuais emergências pode ser uma solução simples e eficaz.

3.1.3 Redirecionamento da ação do usuário:

A proposta deste trabalho é fornecer sistemas de proteção que sejam independentes de ações do usuário, no entanto, isso tornaria necessário que o sistema em questão tivesse uma precisão perfeita em suas decisões, isso ocorre pelo fato de que o acionamento dos procedimentos de segurança será realizado de forma automática, e não é sempre que algo anormal acontece que devemos supor que o usuário está em risco, isso causaria muito transtorno tanto ao usuário quanto aos sistemas de emergência que poderiam ser acionados sem necessidade.

Tendo este fato em mente foi realocada a participação do usuário no sistema. Em contrapartida com os sistemas tradicionais o usuário terá a responsabilidade de

informar ao sistema quando o mesmo encontrar uma situação de emergência de forma equivocada, prevenindo situações de acionamento indevido. Essa forma de envolvimento se torna mais adequada já que o usuário poderá agir em uma situação onde o mesmo não se encontra em risco, e nos casos de risco o acionamento ocorrerá

3.2 Pesquisa

Após a definição do escopo inicial do projeto foi consultado o orientador Murilo Zanini de Carvalho quais seriam os possíveis recursos a serem utilizados para que fosse possível elaborar a construção do protótipo. Dentre as possibilidades, houve algumas pesquisas de viabilidade técnica e financeira para selecionar opções que possuíssem o melhor custo benefício. Após os resultados obtidos foram definidos os principais recursos utilizados conforme a Tabela 1.

Tabela 1 - Recursos utilizados

Recurso	Função
Raspberry Pi 3 Model B	*Machine Learning*
Wemos D1	Coleta de dados
ThingSpeak	Armazenamento
Aplicativo Android	Interação com o usuário

Fonte: Autoria própria

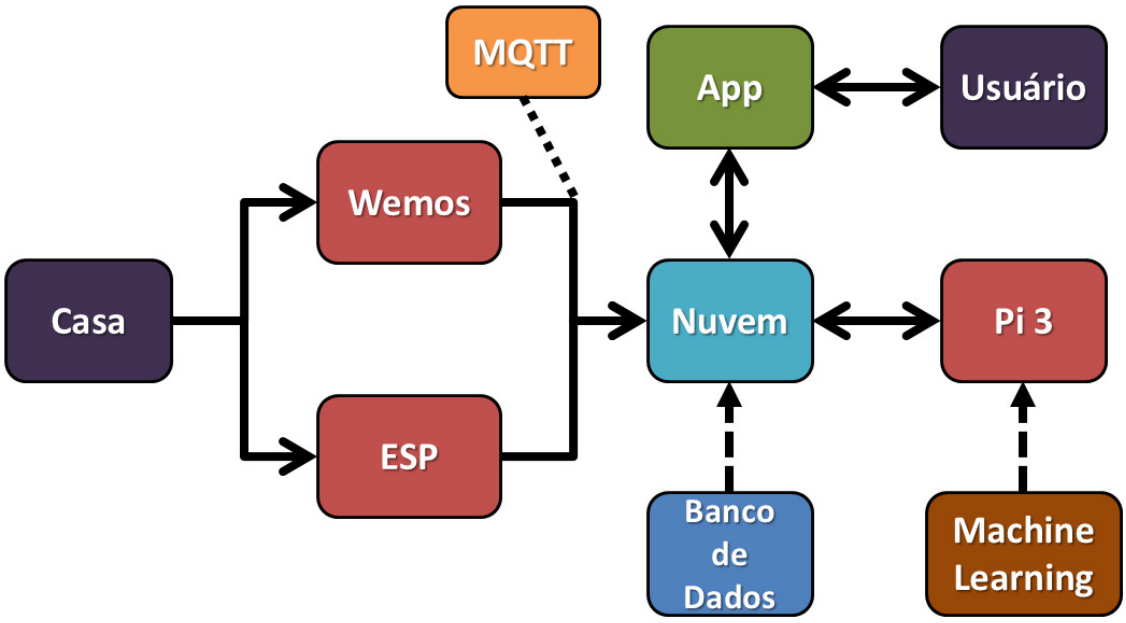
Os principais motivos para a escolha destes recursos foram suas disponibilidade, custo e familiaridade de uso.

Em seguida foram iniciadas pesquisas nos segmentos cujo eram englobados pelo projeto como AAL e IoT, assim como conceitos destinados a parte técnica como *Machine Learning*, desenvolvimento de aplicativos móveis e comunicação via nuvem.

3.1 Proposta para desenvolvimento do trabalho

Com base nos conhecimentos citados no capítulo anterior que foram adquiridos através das pesquisas realizadas e tendo em mente as características finais desejadas, foi elaborado o projeto cuja arquitetura pode ser observada de acordo com o diagrama demonstrado na Figura 11.

Figura 11 - Diagrama do fluxo de dados



Fonte: Autoria própria

Ações em uma residência tais como abertura e fechamento de portas, acionamento de pontos de iluminação, detecção de pessoas nos cômodos, dentre outros, que são monitorados por dispositivos capacitados de se conectarem à internet, caracterizando os conceitos de IoT, onde objetos são colocados em rede, e AAL, um ambiente capaz de auxiliar na vivência através do monitoramento e/ou manipulação de variáveis.

Essa comunicação entre os dispositivos e ao Raspberry Pi é realizada fazendo uso do protocolo de comunicação MQTT para enviar os dados à nuvem, caracterizado por possuir um baixo consumo de dados. Esses dados dos sensores são requisitados no Raspberry Pi e são utilizados no algoritmo de *Machine Learning* responsável por identificar necessidades de atuações em situações de emergência alertando o usuário para que o mesmo decida como prosseguir.

Caso o algoritmo detecte uma anomalia no comportamento o mesmo enviará um novo dado à nuvem responsável pela comunicação entre os dispositivos, indicando ao aplicativo que uma resposta precisa ser enviada. A resposta recebida seja ela por ação do usuário, ou esgotamento do tempo limite, é enviada do aplicativo à nuvem pelo menos campo de comunicação e poderá ser recebida pelo Raspberry para que sejam tomadas as devidas ações.

3.2 Abordagem para modelamento do comportamento do usuário

Foi tomado como cenário fictício para este projeto, uma residência onde existe apenas um morador contendo os seguintes cômodo equipados com sensores de presença: Sala, Cozinha, Sala de jantar, Quarto, Banheiro e Lavanderia. Como não possuíamos os recursos necessários para tornar possível a coleta dados de uma situação real, foi desenvolvida uma lógica via *hardware* para simular o comportamento do usuário e gerar os dados necessários para o treinamento do algoritmo que será explicado no Capítulo 4.

3.3 Sistema de troca de mensagens cliente servidor

Na coleta de dados, serão adquiridas as seguintes informações: quais cômodos da residência proposta possuem pessoas e qual o horário em que os dados foram coletados. O Wemos fica constantemente enviando os dados dos sensores para o ThingSpeak onde cada informação de cada cômodo é armazenada em um campo.

Esses dados são requisitados pelo Raspberry, processados, e caso exista a necessidade, enviará um novo dado de solicitação de resposta através de campo adicional utilizado para comunicação com o aplicativo, cujo após ação do usuário, utilizará o mesmo canal para responder a solicitação

3.4 Caso de uso do sistema e feedback ao usuário

Quando requisitada uma resposta o usuário poderá retornar três informações:

- A primeira indica que o mesmo se encontra em uma situação normal, totalmente livre de qualquer ação que posa lhe causar risco, significando que o sistema enviou um falso positivo.
- A segunda indica que o usuário se encontra em uma situação fora do seu cotidiano, no entanto, a mesma não apresenta perigo no momento. Essa informação pode ser útil por exemplo quando o usuário estiver recebendo alguma visita em casa ou quando está sendo realizado algum serviço em sua residência onde é necessário que a presença de um profissional. Dessa forma o sistema entende que sua decisão não se trata de um falso positivo, uma vez que em outra situação semelhante poderia ser comprometedora a integridade do usuário, no entanto não é acionada nenhuma medida de intervenção.
- A terceira consiste em um caso de perigo, o usuário pode enviar essa informação caso exista a possibilidade do mesmo acelerar o envio da resposta para que o sistema possa reagir com antecedência. Para as situações propostas que visamos cobrir, não se recomenda a utilização desta função pois é classificada como possível reação e poderá apresentar riscos ao usuário. A forma segura de informação de risco é através de esgotamento do tempo disponível para resposta, que automaticamente corresponderá a situação de perigo. Por este motivo também é necessário que sempre que uma anomalia é detectada o usuário deve respondê-la para evitar o uso inadequado do sistema.

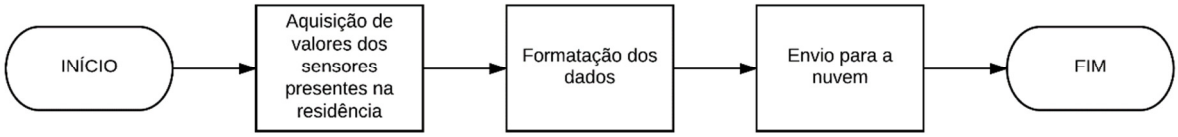
4 Desenvolvimento

Com os conhecimentos adquiridos foi optado pelo desenvolvimento de um protótipo cujo objetivo é realizar a comunicação entre os dispositivos e utilizá-los em uma rede neural podendo assim verificar a viabilidade de futuros aprofundamentos no tema

4.1 Programa do Wemos

A seguir seguem as funções desenvolvidas para possibilitar a coleta de dados e sua comunicação com a nuvem (Figura 12). A programação completa pode ser verificada no Apêndice A.

Figura 12 - Fluxo de dados no Wemos



Fonte: Autoria própria

4.1.1 Função setup()

Responsável por conectar a placa a rede e definir as propriedades para comunicação MQTT com o ThingSpeak (Figura 13).

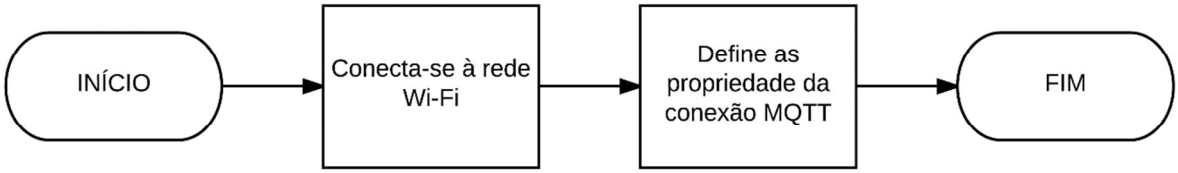


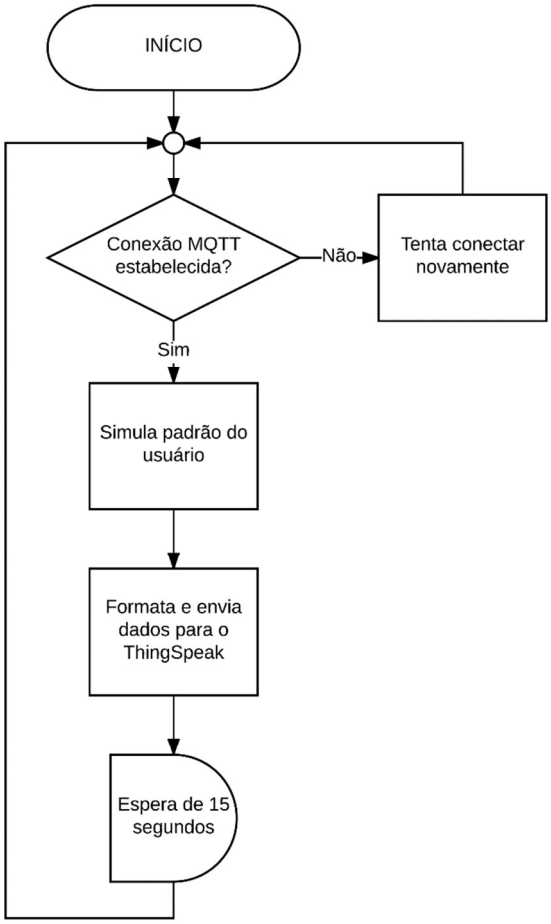
Figura 13 - Fluxograma da função setup

Fonte: Autoria própria

4.1.2 Função loop()

Responsável por conectar-se ao servidor MQTT. Após conexão, atualiza os valores dos cômodos e publica os dados na nuvem (Figura 14).

Figura 14 - Fluxograma da função loop



Fonte: Autoria própria

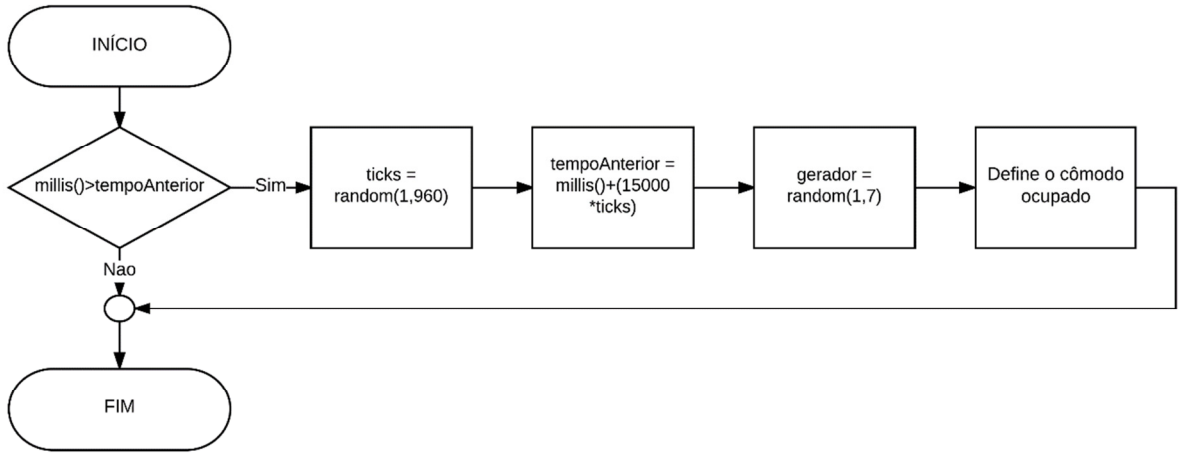
4.1.2.1 Simulação de comportamento do usuário

Para simular a residência proposta inicialmente tomou-se como método de teste a utilização de seis botões nas entradas digitais do Wemos para que fosse possível simular a movimentação do usuário, no entanto, devido a necessidade de um grande número de dados para serem utilizados no treinamento foi necessário criar uma abordagem que pudesse permanecer ligada durante vários dias sem necessidade de alterações manuais. Tal aproximação consiste na permanência do

usuário em um mesmo cômodo, ou fora de casa, aleatoriamente durante intervalos múltiplos de quinze segundos até um limite de quatro horas.

O primeiro número gerado aleatoriamente varia de um a novecentos e sessenta corresponde a quantidade multiplicativa de quinze segundos por qual o usuário irá permanecer no mesmo local. Por exemplo, caso o número gerado seja duzentos e quarenta, esse valor será multiplicado por quinze indicando a presença do usuário por uma hora no local. O segundo número gerado aleatoriamente varia de um a sete e define onde o usuário se encontra, sendo os valores de um a seis equivalentes ao cômodo e o valor sete, equivalente a ausência do usuário na residência (Figura 15).

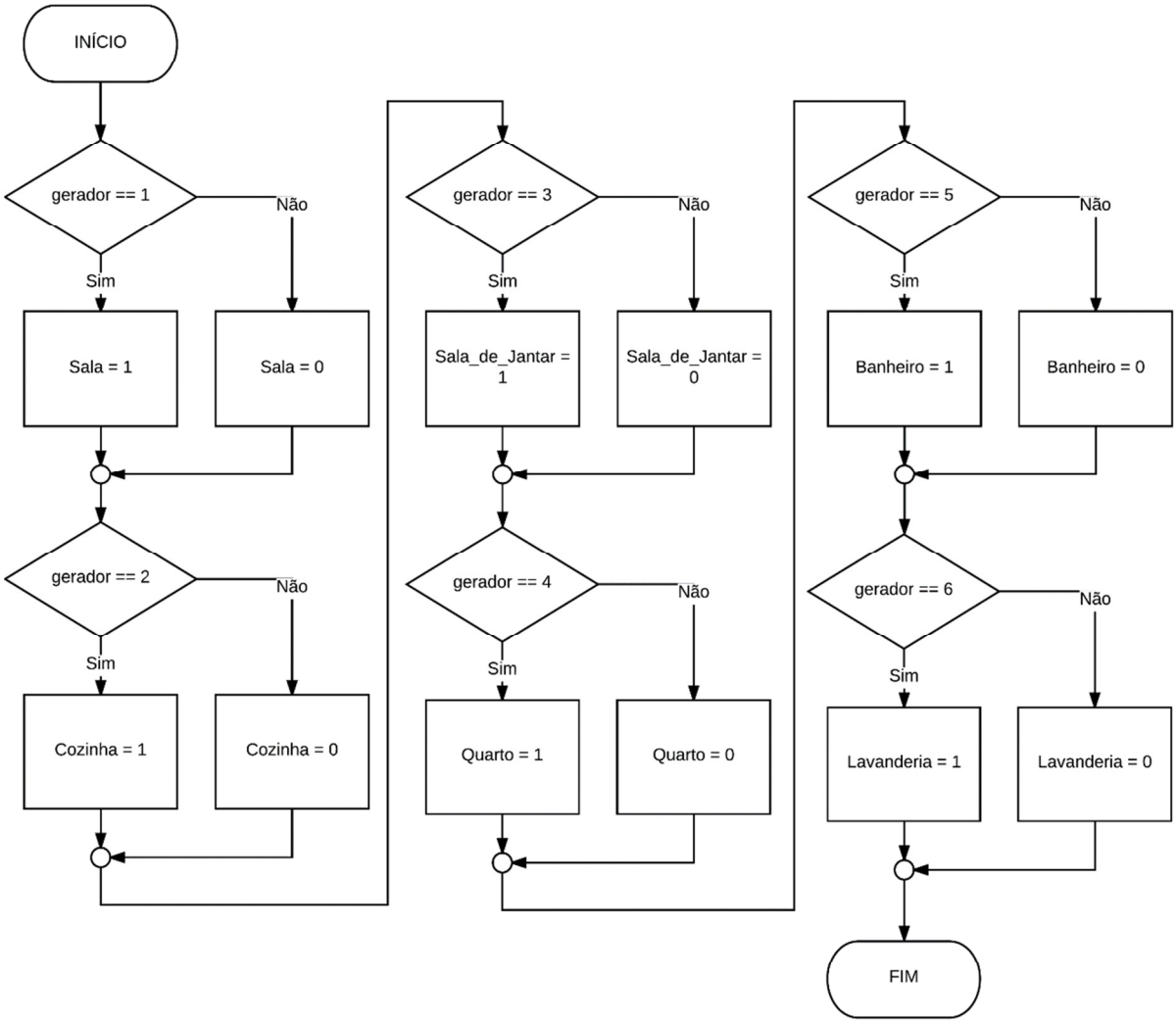
Figura 15 - Fluxograma da simulação de comportamento do usuário



Fonte: Autoria própria

Para realizar a definição do cômodo ocupado foi utilizado uma lógica de comparação conforme a Figura 16. Caso o usuário se encontre em um cômodo será atribuído o valor um a variável do mesmo.

Figura 16 - Fluxograma da definição de cômodo



Fonte: Autoria própria

Este método não cobre todas as situações possíveis de um real comportamento, como por exemplo, a movimentação sequencial entre os cômodos durante a locomoção, no entanto, mesmo com a aplicação desta lógica, este problema continuaria presente devida a versão gratuita do ThingSpeak que limita o número de entradas que podem ser inseridas em seu banco de dados em uma mensagem a cada 15 segundos.

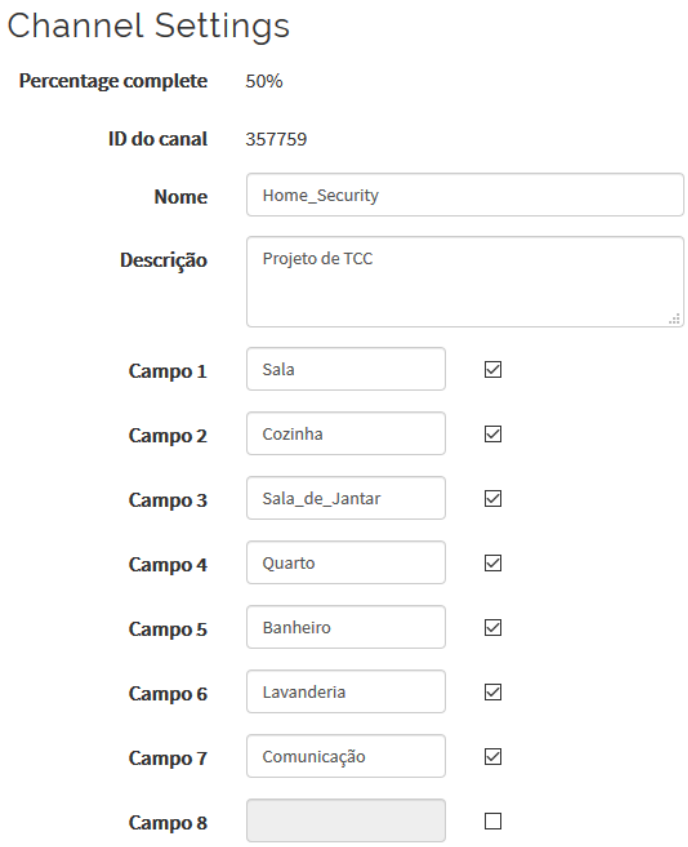
4.1.2.2 Envio de dados ao ThingSpeak

Após a geração de dados eles devem ser formatados para atender ao padrão solicitado e enviados. De acordo com a documentação do ThingSpeak referente a conexões MQTT, quando desejamos enviar informações para atualizar vários campos em apenas uma mensagem, deve ser utilizado a seguinte formatação para que ocorra a publicação:

“channels/id_do_canal/publish/chave_de_escrita/field1=valor1&field2=valor2...&field 8=valor_8”.

O canal do ThingSpeak deverá estar previamente configurado para receber esta quantidade de dados. Na aba *Channel Settings* deverão estar ativados o mesmo número de campos requeridos pelo projeto conforme exibido abaixo (Figura 17).

Figura 17 - Configuração do canal do ThingSpeak



Fonte: Autoria própria

4.2 Raspberry Py 3

Para o desenvolvimento do projeto escolhemos utilizar como sistema operacional o Ubuntu Mate, optamos pela linguagem Python para usufruir de sua biblioteca gratuita scikit-learn que contém funções de *Machine Learning*, além de outras bibliotecas para armazenamento, controle de tempo e formatação de dados. A programação completa pode ser verificada no Apêndice B.

4.2.1 Criação do Banco de Dados

Para treinar uma rede neural precisamos fornecer dados de entrada cujo possuam respostas conhecidas para que a mesma possa melhorar seu desempenho. O sistema em questão irá analisar a rotina do usuário, podendo ser necessário dados de semanas ou mais longos, portanto precisamos concentrá-los em uma massa de dados útil.

Com essa finalidade é criado um arquivo de banco de dados em SQLite cuja tabela Casa contém os seguintes índices: Id, Sala, Cozinha, Sala_de_Jantar, Quarto, Banheiro, Lavanderia, Hora, Minuto, Segundo e Dia_da_Semana.

4.2.2 Aquisição e armazenamento dos dados

Após a criação do banco de dados podemos requisitar as informações do ThingSpeak para que sejam adicionadas, no entanto, elas precisam passar previamente por uma adequação no formato de seus dados para que possam ser avaliados e em seguida inseridos no banco (Figura 18).

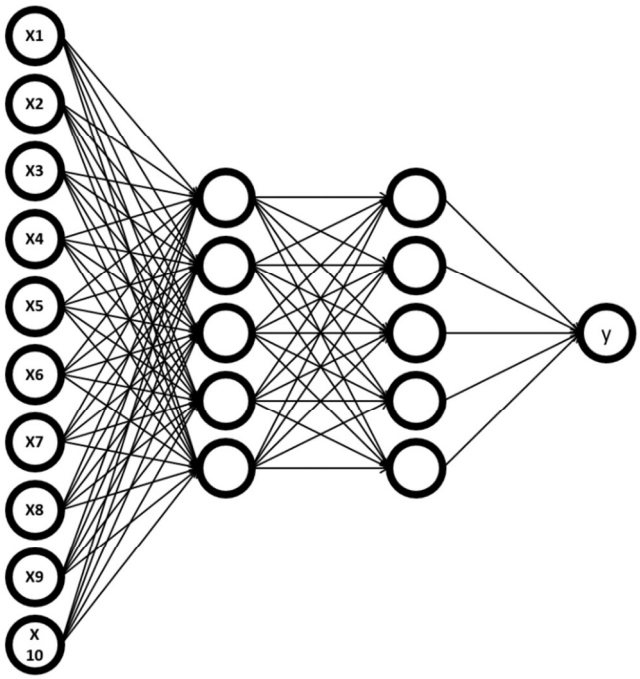
Figura 18 - Fluxograma da aquisição e armazenamento de dados

Fonte: Autoria própria

4.2.3 Algoritmo de segurança

Para que haja o treinamento da rede neural é necessário que o sistema possua uma grande quantidade de dados de entrada, cujos os quais também possuam respostas já definidas. Para o algoritmo foi utilizada uma rede neural *Multi-layer Perceptron* com 2 camadas ocultas de 5 neurônios (Figura 19).

Figura 19 - Rede neural



Fonte: Autoria própria

As entradas utilizadas são descritas na Tabela 2, essas são representações dos sensores de presença utilizados na residência e o tempo de coleta dos dados.

Tabela 2 - Entradas da rede neural

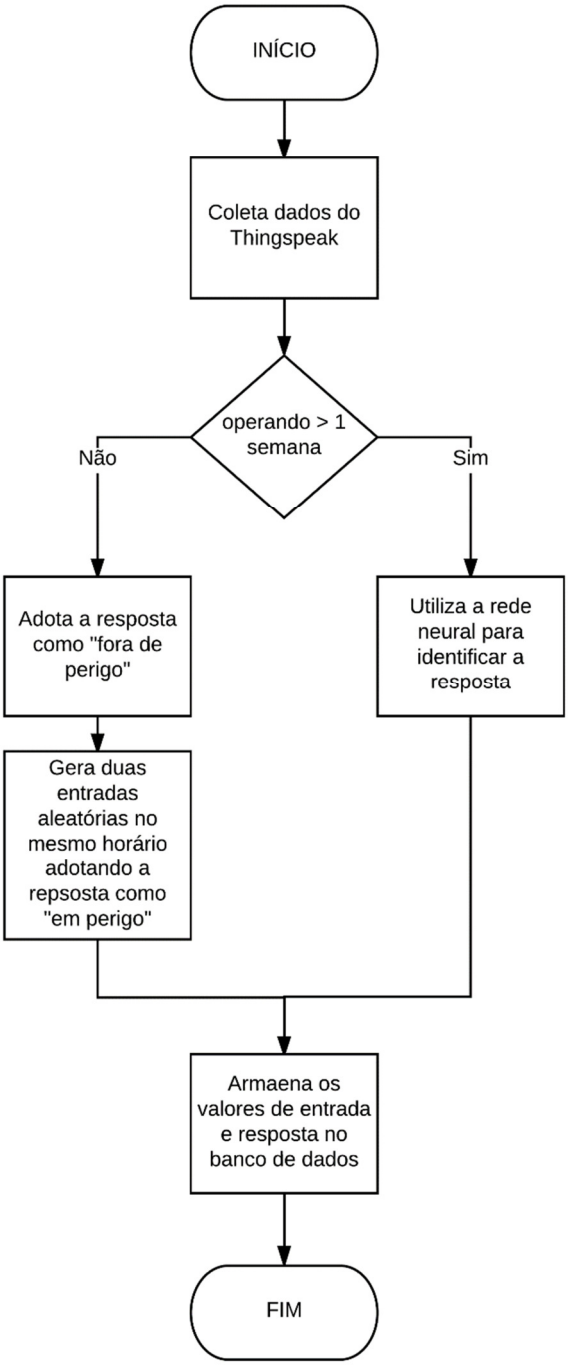
Entrada	Dado
X1	Sala
X2	Cozinha
X3	Sala_de_Jantar
X4	Quarto
X5	Banheiro
X6	Lavanderia
X7	Hora
X8	Minuto
X9	Segundo
X10	Dia_da_Semana

Fonte: Autoria própria

Inicialmente o sistema se encontrará incapaz de operar uma vez que estes dados precisam ainda precisam ser acumulados. Por esse motivo o sistema irá apenas coletar valores durante um determinado tempo, durante os testes foi utilizado de forma empírica o tempo de uma semana de coleta. Os dados de entrada coletados serão automaticamente associados a resposta de situação normal.

Após o tempo de coleta inicial teremos os padrões do usuário para utilizar no treinamento, no entanto, a rede também precisa saber quais situações se referem à uma situação de risco. Por esse motivo, durante a coleta inicial, a cada leitura dos valores provenientes dos sensores o sistema também irá armazenar no seu banco de dados duas entradas onde a presença nos cômodos é gerada de forma aleatória e o tempo é mantido igual ao da última leitura associada a resposta de situação normal. Nestas entradas geradas de forma aleatória a resposta pré-definida será uma situação de perigo (Figura 20).

Figura 20 - Fluxograma da inserção de dados no banco conforme algoritmo

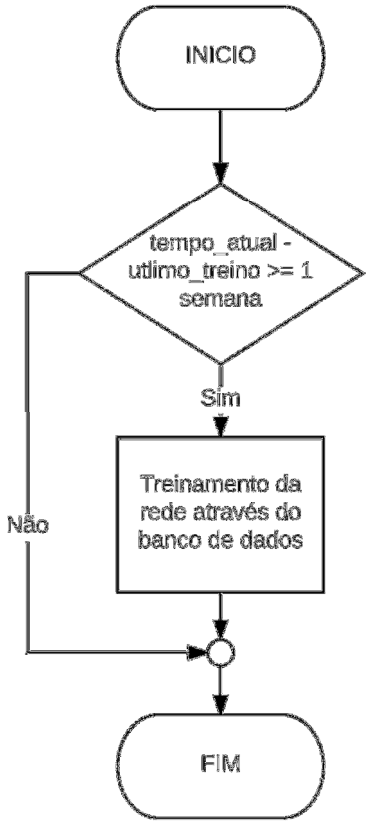


Fonte: Autoria própria

Passado o tempo inicial de coleta de dados a rede utilizará estes dados para realizar o seu primeiro treinamento. Após o tempo inicial de treinamento as respostas deixam de ser pré-definidas e se tornam a resposta da rede neural, em seguida são armazenadas no banco de dados. Como a rotina do usuário poderá se

alterar o sistema realizará o processo de treinamento da rede com os dados presentes no banco a cada semana, mantendo-se assim atualizado (Figura 21).

Figura 21 - Fluxograma de treinamento da rede



Fonte: Autoria própria

4.2.4 Comunicação entre Raspberry Pi e Aplicativo Android em caso de suspeita de perigo

Em situações de operação normal o sistema manda um valor nulo para o canal de comunicação onde nenhuma ação é tomada. Caso o algoritmo detecte uma anomalia ao analisar os dados provenientes dos canais de cada cômodo, o mesmo irá enviar o valor quatro para o sistema, este valor é então recebido pelo aplicativo que fica constantemente alerta para esta mudança.

Os dados de comunicação (Tabela 3) têm como finalidade comunicar o algoritmo presente no Raspberry Pi com usuário utilizando o aplicativo Android.

Tabela 3 - Dados de comunicação

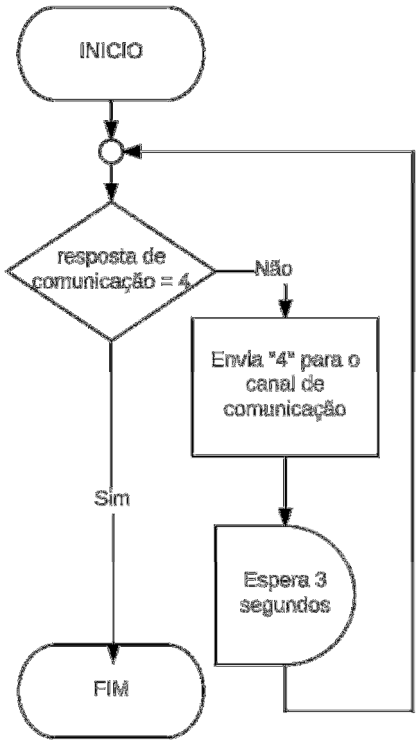
Dado	Significado
1	Resposta de falso positivo
2	Reposta de exceção
3	Resposta de perigo
4	Aguardando resposta

Fonte: Autoria própria

Devido a limitação da versão gratuita do ThingSpeak de possibilitar apenas uma entrada de dados a cada quinze segundos, se torna difícil a utilização do mesmo por três dispositivos diferentes, por esse motivo foi elaborado um sistema onde a comunicação tenta se comunicar em um intervalo de tempo menor do que quinze segundos durante situações de emergência, dessa forma a possibilidade de o canal ficar ocupado devido ao envio constante do Wemos é reduzida.

Caso o sistema perceba uma anomalia o mesmo constantemente envia o valor quatro ao canal de comunicação e requisita a resposta do mesmo em um intervalo de três segundos, dessa forma o envio da comunicação será cessado somente após receber confirmação de que o dado foi recebido (Figura 22).

Figura 22 - Fluxograma do envio de pedido de resposta



Fonte: Autoria própria

Após estar confirmado de que o canal recebeu o valor desejado o sistema entra mais uma vez em uma lógica recursiva como a apresentada anteriormente, onde nesse caso o sistema ficará esperando a resposta proveniente do aplicativo móvel indicando uma das três respostas citadas na Tabela 3.

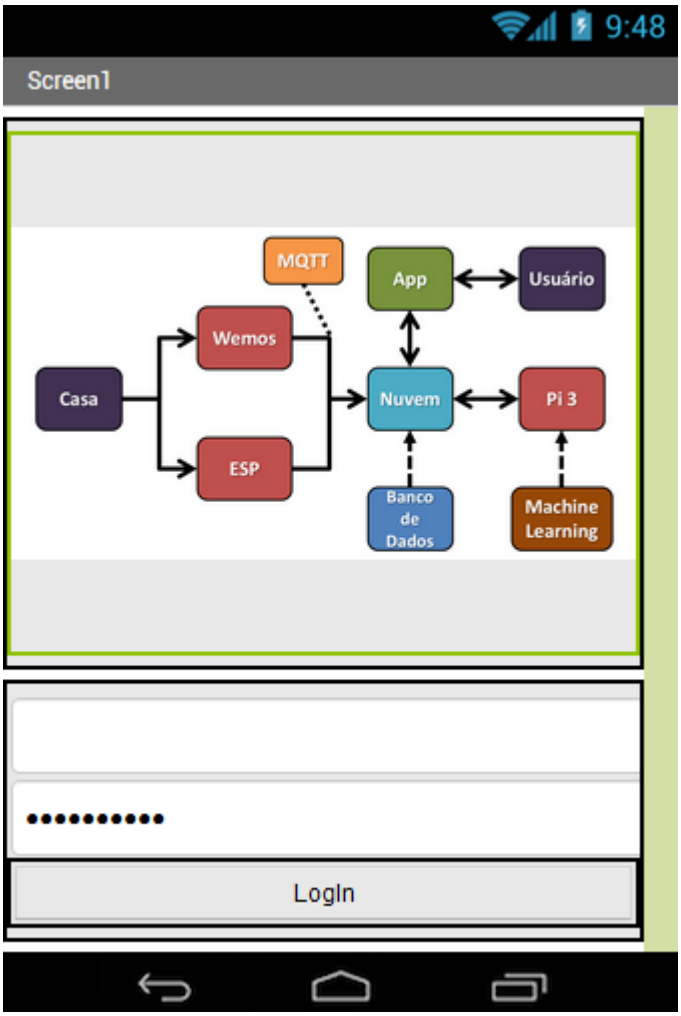
4.3 Aplicativo Android

O aplicativo Android desenvolvido utilizando o MIT App Inventor 2 tem como finalidade permitir que o usuário interaja com o sistema enviando as respostas quando solicitado. A programação completa pode ser verificada no Apêndice C.

4.3.1 Tela inicial

A primeira tela (Figura 23) consta de um simples processo de utilizar um usuário e senha para evitar a utilização inadequada por crianças ou pessoas sem conhecimento do funcionamento do aplicativo. Caso os dados estiverem corretos será aberta uma nova tela.

Figura 23 - Tela inicial



Fonte: Autoria própria

4.3.2 Tela de operação

Na tela de operação (Figura 24) é possível observar qual o estado dos sensores presentes na residência e enviar as respostas quando solicitado.

Figura 24 - Tela de operação

Fonte: Autoria própria

Quando iniciada a mesma define algumas propriedades tais como URLs para aquisição e envio de dados assim como um cronômetro cuja utilidade será explicada posteriormente no item 4.3.2.3.

4.3.2.1 Aquisição de dados do ThingSpeak

Para realizar a aquisição dos dados presentes na nuvem foi utilizado o recurso *Web* presente no MIT App Inventor onde a URL, assim como no Raspberry Pi, foi definida como sendo a que retorna um objeto JSON contendo os últimos valores inseridos no canal. Após a decodificação dos dados, estes são exibidos através dos indicadores na parte superior da tela, o campo “resp” é responsável por

guardar a informação do canal de comunicação que será usada no algoritmo de envio de resposta explicado posteriormente no capítulo.

4.3.2.2 Botões de resposta

Os botões de reposta irão definir qual será o valor que deverá ser enviado no canal de comunicação quando houver a requisição. Neste sistema é necessário que o usuário esteja com o aplicativo aberto e na tela de operação para que as respostas sejam enviadas. Quando o valor é quatro é identificado no canal de comunicação, a cor do plano de fundo da tela muda de branco para vermelho indicando que uma ação foi requisitada, habilitando as ações dos botões de resposta: “OK”, “Exceção” e “Perigo”. Caso o usuário não responda dentro do prazo de trinta segundos, o aplicativo automaticamente enviará a resposta de perigo para o sistema que indicará a ação que deve ser tomada e/ou atualizará os campos do banco de dados caso seja necessário.

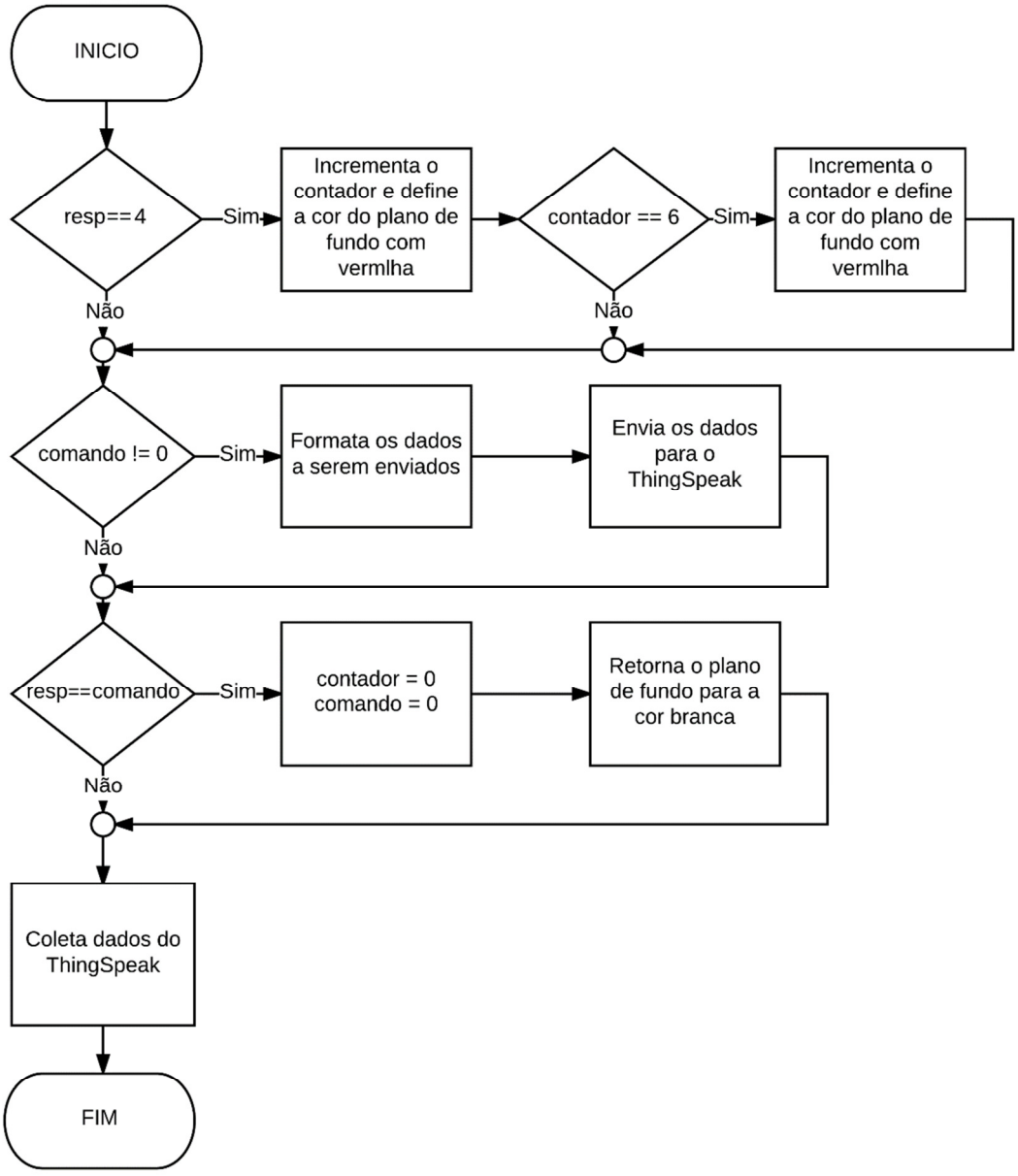
A resposta “OK” indica que o usuário está bem e que se tratou apenas de um alarme falso, alterando a resposta que será salva no banco de dados. A resposta “Exceção” indica que realmente se trata de uma situação anormal, como por exemplo, em encanador realizando um serviço, indicando que não devem ser tomadas medidas de emergência,no entanto não existe a necessidade de alterar a resposta que será salva. A resposta “Perigo” existe apenas para acelerar o pedido de socorro caso seja possível, ela será enviada através do esgotamento do tempo limite.

4.3.2.3 Algoritmo de envio de resposta

Para que seja enviada a resposta utilizaremos o mesmo conceito aplicado para realizar a comunicação do Raspberry à nuvem. A cada intervalo de tempo definido anteriormente no cronômetro o aplicativo irá executar a seguinte lógica: quando o dado de valor quatro for recebido pelo aplicativo o mesmo irá inicializar o contador e trocar o fundo da imagem para vermelho. Caso o usuário aperte um botão ou o tempo limite se esgotar a variável comando receberá o valor que deverá ser enviado, uma vez este valor definido o sistema irá continuamente enviar a

informação para o ThingSpeak até que a leitura do mesmo indique que o dado foi recebido, retornando o fundo da tela para a cor branca e zerando o contador (Figura 25).

Figura 25 - Fluxograma do algoritmo de comunicação do aplicativo

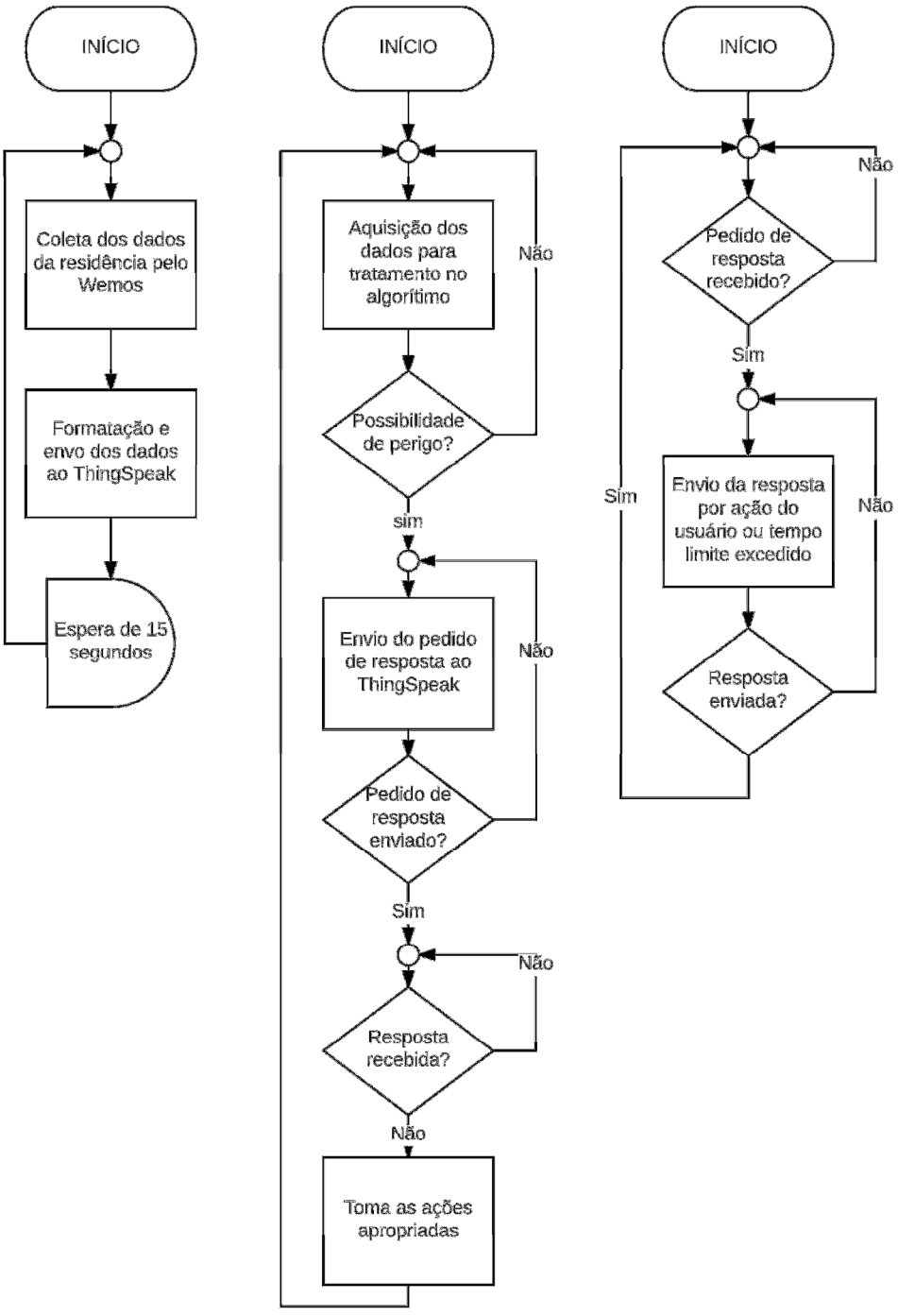


Fonte: Autoria própria

4.4 Visão Geral

De forma geral, o sistema opera de acordo com Figura 26, a coluna da esquerda representa o Wemos, a coluna central representa o Raspberry Pi 3 e a coluna da direita representa o aplicativo Android.

Figura 26 - Fluxograma geral



Fonte: Autoria própria

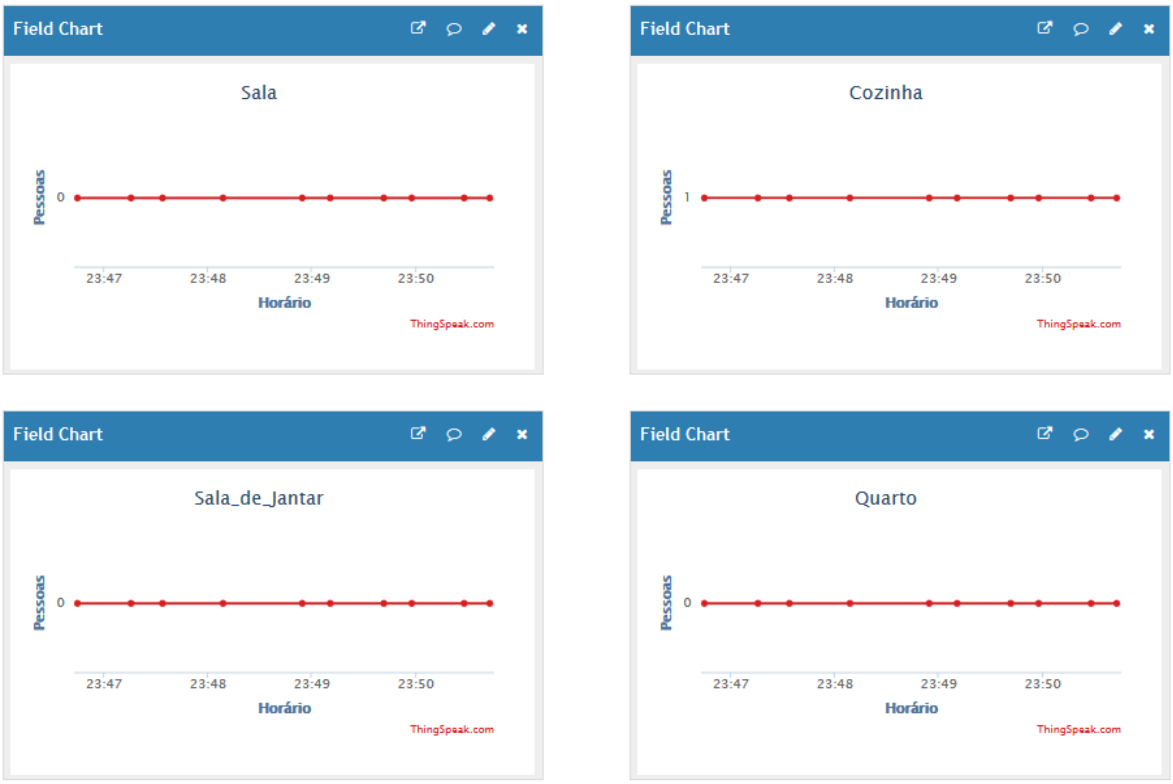
5 Resultados

Neste capítulo são apresentados os resultados obtidos após o desenvolvimento do projeto.

5.1 Comunicação

A comunicação ocorreu de forma perfeita, os dados enviados pelo Wemos, Raspberry e aplicativo foram recebidos pelo ThingSpeak (Figura 27) e puderam ser adquiridos conforme a necessidade.

Figura 27 - Dados recebidos pelo ThingSpeak



Fonte: Autoria própria

5.2 Algoritmo

Após aquisição dos dados durante uma semana foi utilizado um método que consiste na divisão dos dados, uma porcentagem para treinamento da rede e o restante utilizado como teste para verificar se a resposta enviada condiz com aquelas que estavam previstas. Os resultados obtidos ficaram em torno de 95% de precisão ao definir se uma situação é perigosa ou não.

5.3 Oportunidade de melhorias e observações

Após o desenvolvimento do protótipo foram constatados pontos de melhoria e ou adições que poderiam ser feitos para melhorar os resultados obtidos, sendo eles:

- A criação de um sistema de cadastro para que usuários diferentes tenham acesso ao sistema.
- Informar o usuário da necessidade de responder ao sistema sem estar obrigatoriamente na tela de operação.
- Aquisição da versão paga do ThingSpeak permitindo adições de dados a cada segundo.

6 Considerações Finais

Com base nos resultados obtidos seguem neste capítulo as conclusões e sugestões para trabalhos futuros.

6.1 Conclusões

Tendo como base os resultados apresentados no Capítulo 5 podemos concluir que a metodologia utilizada durante a etapa de pesquisa se mostrou eficaz para prover os conhecimentos necessário para a realização do projeto.

A conclusão deste trabalho comprova a possibilidade de comunicação entre os dispositivos na residência utilizando a arquitetura proposta, assim como a viabilidade do uso de *Machine Learning* voltado a aplicações de segurança residencial.

Os resultados obtidos indicam que existe a possibilidade de realizar um estudo mais aprofundado na área de *Machine Learning* para aumentar a eficácia do sistema, assim como um aprimoramento nas funcionalidades do aplicativo e possivelmente criar um produto final para ser comercializado.

6.2 Trabalhos futuros

Tendo como base os resultados obtidos as propostas futuras são:

- Identificar o melhor modelo de rede para o sistema.
- Aquisição de dados reais ou aprimoramento de simulação.
- Instalação de um sistema de rede e acionamento para situações de emergência.
- Tornar o sistema e o chamado de socorro funcionais

6.2.1 Identificar o melhor modelo de rede para o sistema

Durante a elaboração deste projeto não foram realizados testes comparativos para identificar qual o modelo de rede neural cujo apresenta maior precisão na tomada de decisões. Um estudo comparativo com os dados obtidos poderia ser realizado para aprimorar a precisão do sistema.

6.2.2 Aquisição de dados reais ou aprimoramento de simulação

Como foi citado anteriormente, a lógica proposta para simular o comportamento do usuário não representa uma situação real. Melhorias no algoritmo de localização do usuário poderão ser realizadas, assim como a aquisição dados reais de uma residência para garantir viabilidade do projeto como um produto.

6.2.3 Instalação de um sistema de rede e acionamento para situações de emergência.

Faltas de conectividade seriam fatais uma vez que o sistema está dependente da mesma. Uma conexão reserva 3G poderia ser acionada para suprir essa demanda. A inclusão de um sistema de último recurso, como por exemplo um botão em algum local da residência, poderia ser utilizado em situações onde nenhuma rede estivesse disponível.

6.2.4 Tornar o sistema e o chamado de socorro funcionais

Durante este projeto a decisão proveniente da rede neural foi disponibilizada apenas como uma resposta em forma de texto para verificar o seu funcionamento. A distribuição de sensores em uma residência, assim como a criação de um sistema de acionamento de autoridades como por exemplo, chamadas programadas, poderá ser desenvolvido para efetivamente proteger o usuário.

Bibliografia

Amazon Web Services. (2017). *Computação em nuvem*. Fonte: Site da Amazon Web Services: https://aws.amazon.com/pt/what-is-cloud-computing/. Acesso em: 16 jun. 2017

Amazon Web Services. (2017). *Machine Learning*. Fonte: Site da Amazon Web Services: https://aws.amazon.com/pt/machine-learning/?p=tile. Acesso em: 16 jun. 2017

Amazon Web Services. (2017). *Produtos*. Fonte: Site da Amazon Web Services: https://aws.amazon.com/pt/products/. Acesso em: 16 jun. 2017

Arduino. (2017). *Introdução ao Arduino*. Fonte: Site da Arduino: https://www.arduino.cc/en/Guide/Introduction. Acesso em: 22 mai. 2017

Arduino. (2017). *Produtos*. Fonte: Site do Arduino: https://www.arduino.cc/en/Main/Products. Acesso em: 22 mai. 2017

Azzola, F. (2016). *Tutorial Protocolo MQTT*. Fonte: Side do Surviving With Android: https://www.survivingwithandroid.com/2016/10/mqtt-protocol-tutorial.html. Acesso em: 1 jun. 2017

Benjamin Xie, H. A. (2016). Skill Progression in MIT App Inventor. *IEEE*, pp. 213-217.

Bygholm, A., & Kanstrup, A. M. (2015). The Living Chalange of Ambient Assisted Living - a literature review.

Coelho, L. P., & Rechert, W. (2015). *Building Machine Learning Systems with Python.* Packt Publishing Ltd.

Coelho, L. P., & Richert, W. (2015). *Building Machine Learning Systems with Python.* Birmingham: Packt Publishing Ltd.

Dohr, A., R.Modre-Osprian, Drobics, M., Hayn, D., & G.Schreier. (2010). The Internet of Things for Ambient Assited Living.

Fan, X., Zhang, H., Leung, C., & Miao, C. (2016). Comparative Study of Machine Learning Algorithms for Activity Recognition with Data Sequence in Home-like Enviroment.

Google Inc. (2017). *Configurando seu ambiente de desenvolvimento*. Fonte: Site do Android Things: https://developer.android.com/things/preview/index.html#set_up_your_development_environment. Acesso em: 20 jun. 2017

Google Inc. (2017). *Trazendo produção em dispostivos para todos*. Fonte: Site do Android Things: https://developer.android.com/things/hardware/index.html. Acesso em: 20 jun. 2017

Google Inc. (2017). *Visão Geral*. Fonte: Site do Android Things: https://developer.android.com/things/sdk/index.html. Acesso em: 20 jun. 2017

Kodali, R. K., & Mahesh, K. S. (2016). A low cost implementation of MQTT using ESP8266.

Kodali, R. K., & Sahu, A. (2016). An IoT based Weather Information Prototype Using WeMos.

Kodali, R. K., & Soratkal, S. (2016). MQTT based Home Automation System Using Esp8266.

Md, S., G, V., G, R., DharmaSavarini, & Mittal, V. K. (2016). Muti-Functional Secured Smart Home.

MQTT.org. (2017). *Perguntas Frequentes*. Fonte: Site do MQTT: http://mqtt.org/faq. Acesso em: 1 jun. 2017

Numbeo. (2017). *Índice de Crimes*. Fonte: Site do Numbeo: https://www.numbeo.com/crime/gmaps_rankings_country.jsp. Acesso em: 22 abr. 2017

Patchava, V., Kandala, H. B., & Babu, P. R. (2015). A Smart Home Automation Technique with Raspberry Pi using IoT.

Rao, P. B., & Uma, S. (Maio de 2015). Raspberry Pi Automation with Wireless Sensors Using Smart Phone. *International Journal of Computer Science and Mobile Computing*, pp. 797-803.

Raschka, S. (2015). *Python Machine Learning.* Birmingham: Packt Publishing Ltd.

Raspberry Pi Foundation. (s.d.). *Raspberry Pi 3 Model B*. Fonte: Site da Raspberry Pi Foundation: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Acesso em: 5 jun. 2017

Scharler, H. (16 de Fevereiro de 2011). *ThingSpeak - Comunique coisas facilmente com a nuvem*. Fonte: Blog Adafruit: https://blog.adafruit.com/2011/02/16/thingspeak-easily-get-stuff-to-to-talk-to-the-cloud/. Acesso em: 28 mai. 2017

SecurityOne. (16 de Agosto de 2016). *Básico sobre alarmes de segurança residencial*. Fonte: Site da SecurityOne: http://securityoneonline.com/home-alarm-security-basics-part-6-wireless-home-security/. Acesso em: 24 abr. 2017

Sernani, P., Claudi, A., Palazzo, L., Dolcini, G., & Dragoni, A. F. (2013). Home Care Expert Systems for Ambient Assisted Living: A Multi-Agent Approach.

Singh, B. (15 de Junho de 2016). *Impacto do IoT em seus negócios*. Fonte: Site da Business 2 Comunity: http://www.business2community.com/big-data/internet-things-iot-going-impact-business-01572401#ObOx4OfoKiiXeazi.97. Acesso em: 18 mai. 2017

Spinsante, S., Gamibi, E., Montanini, L., & Raffaeli, L. (2015). Data Management in Ambient Assisted Living Plataforms Approaching IoT: a Case Study.

The MathWorks, Inc. (2017). *aprenda mais sobre o ThingSpeak*. Fonte: Site do ThingSpeak: https://thingspeak.com/pages/learn_more. Acesso em: 28 mai. 2017

Vara, A. J., & Skarmeta, A. (2010). An Internet of Things-Based Pesonal Device for Diabetes Therapy Management in Ambient Assisted Living (AAL).

Zaric, N., Djurisic, M. P., & Mihovska, A. (2016). Ambient Assisted Living Systems in the Context of Human Centric Sensing and IoT Concept: eWall Case Study.

APÊNDICE A – Código utilizado no Wemos

```
#include <SPI.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
// substituir com seus dados
const char* ssid = "DarkBlader"; //ssid da rede
const char* password = "23101997"; //senha da rede
const char* server = "mqtt.thingspeak.com";

WiFiClient WFclient;
PubSubClient mqttClient(WFclient);
int ticks = random(1,960); //quantidade de 15 segundos
int gerador=random(1,7); //gerador de posicao do usuario
unsigned long tempoAnterior = 15000 * ticks; //tempo de permanencia
void setup()
{
Serial.begin(115200);
delay(10);
Serial.println();
Serial.print("Conectando-se em ");
Serial.println(ssid);

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED)  //tenta continuamente se conectar a
rede
{
delay(1000);
Serial.print(".");
}
Serial.println();
Serial.println("Conexão WiFi estabelecida");
mqttClient.setServer(server, 1883); //define as propriedades de conexao
mqtt
}
void loop()
{
 //Tenta se conectar continuamente ao servidor
    while (!mqttClient.connected()) //
  {
    Serial.print("Estabelecendo conexão");
    if (mqttClient.connect("Id_do_cliente")) //conecta-se ao ThingSpeak
utilizando o ID informado
    {
      Serial.println("Conectado");
    }
    else
      delay(10000);
  }

  // Estabele conexao continua com o servidor
  mqttClient.loop();

float sala=digitalRead(D2);
float cozinha=digitalRead(D3);
```

```
float sala_de_jantar=digitalRead(D4);
float quarto=digitalRead(D5);
float banheiro=digitalRead(D6);
float lavanderia=digitalRead(D7);
if (millis()>tempoAnterior){ //caso o tempo de permanencia tenha se
esgotado
ticks = random(1,960);
tempoAnterior = millis()+(15000 * ticks);
gerador=random(1,7);
} //define novo local e tempo de permanencia
if(gerador==1) //define o comodo com base no valor gerado
  sala=1;
  else
  sala=0;
 if(gerador==2)
  cozinha=1;
  else
  cozinha=0;
 if(gerador==3)
  sala_de_jantar=1;
  else
  sala_de_jantar=0;
 if(gerador==4)
  quarto=1;
  else
  quarto=0;
 if(gerador==5)
  banheiro=1;
  else
  banheiro=0;
 if(gerador==6)
  lavanderia=1;
  else
  lavanderia=0;

if (mqttClient.connected()) {
  String dados = "field1=" + String(sala) + "&" + "field2=" +
String(cozinha) + "&" + "field3=" + String(sala_de_jantar) +
  "&" +"field4=" + String(quarto) + "&" + "field5=" + String(banheiro) +
"&" + "field6=" + String(lavanderia);// + "&" + "field7=0.0"; //gera a
string a ser enviada para o ThingSpeak
  int tamanho = dados.length(); //tamanho da string a ser enviada
  char envioDeDados[tamanho]; //incializa um vetor de caracteres
  dados.toCharArray(envioDeDados,tamanho+1); //converte a String para vetor
de caracteres
  //Envia os dados para o ThingSpeak

mqttClient.publish("channels/357759/publish/WDEZ44HJMFV32RZH",envioDeDados)
; //substituir o id do canal e a chave de escrita
}
delay(15000);//Aguarda 15 segundos para enviar os dados
}
```

APÊNDICE B – Código utilizado no Raspberry Pi 3

```
#--------------------Bibliotecas-----------------------------------------
-------------------------------------------------
#!/usr/bin/env python
import urllib.request #responsavel pela aquisicao da resposta proveniente
do ThingSpeak
import json #responsavel pelo tratamento da resposta em JSON
import sqlite3 #responsavel pelas funcoes do banco de dados
from sklearn.neural_network import MLPClassifier #responsavel pela decisao
do sistema (Machine Learning)
from sklearn.model_selection import train_test_split
import numpy as np #responsavel pelo tratamento de vetores/listas para o
formato adequado
import itertools #responsavel pelo tratamento de vetores/listas para o
formato adequado
import time #responsavel pelos tempos de espera do programa
import datetime #responsavel pela aquisição da data do sistema para
controlar o tempo de treinamento e de execucao incial
#--------------------Bibliotecas-----------------------------------------
-------------------------------------------------

#--------------------Rede Neural-----------------------------------------
-------------------------------------------------
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(10, 8),
random_state=1) #Modela a rede neural
#--------------------Rede Neural-----------------------------------------
-------------------------------------------------

#--------------------Banco de dados--------------------------------------
-------------------------------------------------
db = sqlite3.connect('database2') #conexao ao banco de dados

cursor=db.cursor() #inicializa o cursor de navegação no banco de dados

cursor.execute('''
  CREATE TABLE IF NOT EXISTS Casa(id INTEGER PRIMARY KEY, Sala TEXT,
Cozinha TEXT, Sala_de_Jantar TEXT,
                                        Quarto TEXT, Banheiro TEXT,
Lavanderia TEXT,
                                        Horario TEXT, Comunicacao TEXT,
Resposta TEXT, Hora TEXT,
                                        Minuto TEXT, Segundo
TEXT,Dia_da_Semana TEXT)
''') #cria a tabela de dados
#--------------------Banco de dados--------------------------------------
-------------------------------------------------

#--------------------Dados do Canal do Thingspeak------------------------
-------------------------------------------------
READ_API_KEY='LIZMJW0D00WNDPL8' #chave para escrita no canal do ThingSpeak
CHANNEL_ID='357759' #ID do canal do ThingSpeak
#--------------------Dados do Canal do Thingspeak------------------------
-------------------------------------------------
```

```
#---------------------Leitura e armazenamento dos dados--------------------
-------------------------------------------------
def main():
    inicio_sistema = int(datetime.date.today().strftime("%j"))
    while(1):
        tempo_atual = int(datetime.date.today().strftime("%j"))
        if tempo_atual==1:
            inicio_sistema = 1
        X = []    #incializacao do vetor contendo os dados de treinamento
        y = []    #incializacao do vetor contendo as respostas
        dados_para_avaliacao=np.array([0,0,0,0,0,0,0,0,0,0]) #inicialização
do vetor contendo os dados atuais dos sensores
        db = sqlite3.connect('database2') #conexao ao banco de dados
        cursor=db.cursor() #inicializa o cursor de navegação no banco de
dados
        time.sleep(15)#espera 15s para que novos dados provenientes do
Wemos sejam recebidos pelo ThingSpeak
        resp=0 #resposta de feedback para garantir valor de comunicação
        conn =
urllib.request.urlopen("http://api.thingspeak.com/channels/%s/feeds/last.js
on?api_key=%s" \
                               % (CHANNEL_ID,READ_API_KEY)) #aquisicao da
resposta do ThingSpeak

        response = conn.read() #leitura da resposta
        data=json.loads(response.decode('latin1')) # formata a resposta
recebida

        dia_da_semana=datetime.datetime.today().weekday()

        dados_para_avaliacao[0]=float(data['field1']) #Preenche o vetor a
ser avaliado com os dados de interesse (sala)
        dados_para_avaliacao[1]=float(data['field2']) #Preenche o vetor a
ser avaliado com os dados de interesse (cozinha)
        dados_para_avaliacao[2]=float(data['field3']) #Preenche o vetor a
ser avaliado com os dados de interesse (sala_de_jantar)
        dados_para_avaliacao[3]=float(data['field4']) #Preenche o vetor a
ser avaliado com os dados de interesse (quarto)
        dados_para_avaliacao[4]=float(data['field5']) #Preenche o vetor a
ser avaliado com os dados de interesse (banheiro)
        dados_para_avaliacao[5]=float(data['field6']) #Preenche o vetor a
ser avaliado com os dados de interesse (lavanderia)
        horario=data['created_at'].split('T') #divide a resposta recebida
no campo 'created_at' (dia+T+hora+Z) em horario+Z
        rel=horario[1].split(':') #divide o horario em hora, minutos e
segundos+Z
        hora=rel[0]
        minu=rel[1]
        seg=rel[2]
        seg=seg[:-1] #eliminha o caractere Z dos segundos
        dados_para_avaliacao[6]=hora #Preenche o vetor a ser avaliado com
os dados de interesse (hora)
        dados_para_avaliacao[7]=minu #Preenche o vetor a ser avaliado com
os dados de interesse (minutos)
        dados_para_avaliacao[8]=seg  #Preenche o vetor a ser avaliado com
os dados de interesse (segundos)
        dados_para_avaliacao[9]=dia_da_semana  #Preenche o vetor a ser
avaliado com os dados de interesse (segundos)

        for i in range(3):
```

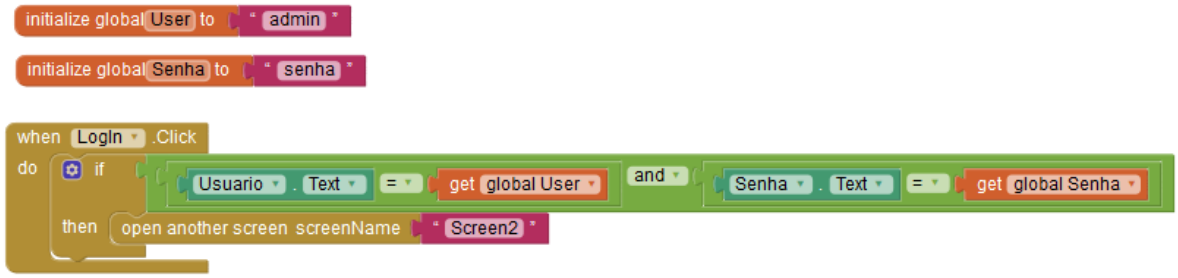
```
            randinput=np.random.randint(2, size=9) #gera um vetor de
valores aleatórios entre 0-1 de tamnho 6
            randinput.astype(float) #transforma os valores para o tipo
numpy.float

            cursor.execute('''INSERT INTO Casa(Sala, Cozinha,
Sala_de_Jantar, Quarto, Banheiro, Lavanderia, Horario,
Comunicacao,Resposta,Hora,Minuto,Segundo,Dia_da_Semana)
                           VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?)''',
(str(float(randinput[0])),str(float(randinput[1])),str(float(randinput[2]))
,str(float(randinput[3])),

str(float(randinput[4])),str(float(randinput[5])),data['created_at'],data['
field7'],"1.0",hora,minu,seg,dia_da_semana))
                           #insere os dados na tabela

#---------------------Leitura e armazenamento dos dados--------------------
-------------------------------------------------

#---------------------TREINAMENTO-----------------------------------------
-------------------------------------------------
        cursor.execute('''SELECT id FROM Casa ''') #Seleciona a coluna id
da tabela
        ids = cursor.fetchall() #retorna toda a coluna
        tamanho=len(ids) #retorna o tamanho da coluna id coletada

        for i in range(tamanho):
                cursor.execute('''SELECT Sala, Cozinha, Sala_de_Jantar,
Quarto, Banheiro, Lavanderia, Hora, Minuto, Segundo, Dia_da_Semana FROM
Casa WHERE id=? ''',(i+1,))
                                #seleciona os campos da tabela de acordo
com o id
                user = cursor.fetchall() #retorna as colunas
correspondentes (apenas 1 linha pois depende do id)
                usernp = np.array(user) #transforma em um array no formato
numpy
                userastype = usernp.astype(np.float) #transforma os valores
para o tipo numpy.float
                linha=list(itertools.chain.from_iterable(userastype))
                #print(linha) #exibe o vetor gerado
                X.append(linha) #adiciona o vetor a X

        cursor.execute('''SELECT Resposta FROM Casa''')
        #seleciona os campos da tabela de acordo com o id
        user = cursor.fetchall() #retorna as colunas correspondentes
(apenas 1 linha pois depende do id)
        usernp = np.array(user) #transforma em um array no formato numpy
        userastype = usernp.astype(np.float) #transforma os valores para o
tipo numpy.float
        y=list(itertools.chain.from_iterable(userastype))

        #print()
        #print(X)
        #print()
        #print(y)

        if (tempo_atual - inicio_sistema) % 7 == 0 and tempo_atual !=
inicio_sistema:
            #Executa o algoritmo de treinamento
            inicio_sistema = datetime.date.today().strftime("%j")
```

```
            X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=0)

            clf.fit(X_train, y_train)
            print(clf.score(X_test,y_test))

#--------------------TREINAMENTO-----------------------------------------
-------------------------------------------------

#--------------------ALGORITMO-------------------------------------------
-------------------------------------------------
        resposta="0.0"
        if tempo_atual-inicio_sistema>=7:
            if clf.predict([dados_para_avaliacao]) == 1:
                print('!POSSIBLE DANGER!')

                while resp != '4':
                    post =
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=WDEZ44HJM
FV32RZH&field1=4&field2=4&field3=4&field4=4&field5=4&field6=4&field7=4")

                    conn =
urllib.request.urlopen("http://api.thingspeak.com/channels/%s/feeds/last.js
on?api_key=%s" \
                                % (CHANNEL_ID,READ_API_KEY)) #aquisicao
da resposta do ThingSpeak

                    response = conn.read() #leitura da resposta
                    data=json.loads(response.decode('latin1')) # formata a
resposta recebida
                    resp=data['field7']
                    print(resp)
                    time.sleep(3)

                while resp != '3' and resp != '2' and resp != '1':
                    conn =
urllib.request.urlopen("http://api.thingspeak.com/channels/%s/feeds/last.js
on?api_key=%s" \
                                % (CHANNEL_ID,READ_API_KEY)) #aquisicao
da resposta do ThingSpeak
                    response = conn.read() #leitura da resposta
                    data=json.loads(response.decode('latin1')) # formata a
resposta recebida
                    resp=data['field7']
                    print("Esperando por resposta")
                    time.sleep(3)

                if resp == '3':
                    print("Socorro")
                    resposta="1.0"
                elif resp == '2':
                    resposta="1.0"
                    print("Exceção")
                elif resp == '1':
                    print("OK")

        cursor.execute('''INSERT INTO Casa(Sala, Cozinha, Sala_de_Jantar,
Quarto, Banheiro, Lavanderia, Horario,
Comunicacao,Resposta,Hora,Minuto,Segundo,Dia_da_Semana)
```

```
                    VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?)''',
(data['field1'],data['field2'],data['field3'],data['field4'],data['field5']
,data['field6'],

data['created_at'],data['field7'],resposta,hora,minu,seg,dia_da_semana))
                    #insere os dados na tabela

#---------------------ALGORITMO-----------------------------------------
-----------------------------------------------

        conn.close()    #termina a conn (conexão?)
        db.commit()     #salva as alterações no banco de dados
        db.close()      #fecha o banco de dados



if __name__ == '__main__':
    main()
```

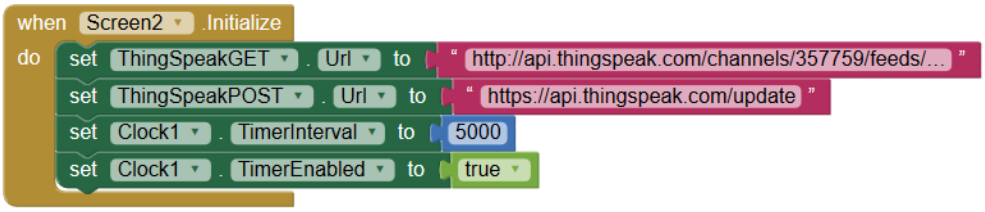
APÊNDICE C – Código utilizado no MIT App Inventor

Tela Inicial



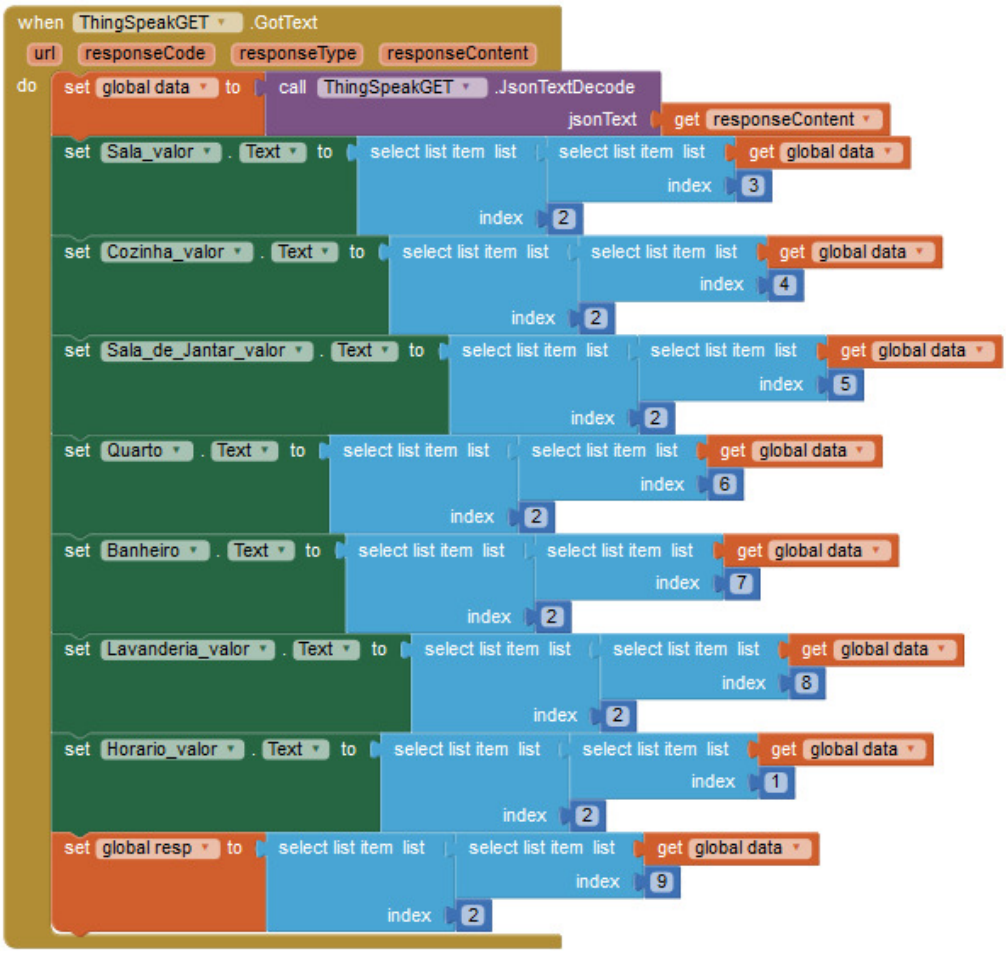
Tela de operação

Configurações iniciais



Botões de resposta

Aquisição de dados do ThingSpeak



Algoritmo de envio de resposta