IoT e Visão Computacional Aplicados a Segurança Residencial

Giovanni Bauermeister Santana[1], Zaida Aguila[2]

[1]Autor, Universidade Salesiana de São Paulo, giovanni.bauermeister@gmail.com,
[2]Orientador, Universidade Salesiana de São Paulo, zaida.aguila@sj.unisal.br

Resumo *– Este artigo apresenta a implementação de um sistema de segurança residencial que é, a um certo nível, inteligente e autônomo, que pode reconhecer variações do meio em que estão instalados com objetivo de aumentar a efetividade dos atuais sistemas de segurança residencial. Onde entende-se por variações do meio, toda movimentação, intrusão ou anormalidades que ocorrem em* uma *residência enquanto o sistema estiver ativo. Apesar de alguns sistemas no mercado já possuírem certo grau de inteligência, esse projeto visa analisar, implementar e testar uma harmonia entre os princípios de IoT (Internet das coisas) e visão computacional. O resultado desse trabalho implica em uma maior efetividade na segurança das residências e no aumento da confiabilidade dos sistemas de segurança.*

Palavras-chave*: Segurança residencial, IoT, visão computacional.*

Abstract *– The general purpose which this article presents, is a way to increase the effectiveness of current residential security systems. Implementing a system, in some ways intelligent and autonomous, the security system can recognize variations in the environment in which they are installed. By environment variations meaning any movements, trespassing or abnormalities that may occur in someone's residence while the system is active. Although some systems in the market already have some kind of intelligence, this projects aims to analyze, test and implement a harmony between principles of IoT (Internet of Things) and computational vision. The positive result of this work implies an increase of effectiveness in home security as well as an increase in security systems reliability.*

Keywords: *Home security, IoT, computational vision.*

I. INTRODUÇÃO

Segurança residencial é um tema que vem sendo estudado a muito tempo. Isso se deve ao fato de que 47,2% da população de 10 anos ou mais de idade, declarou não sentir-se segura na cidade onde residia. E, se tratando especificamente das vítimas de tentativa de roubo ou furto, o percentual subiu para 70,4% (Pesquisa IBGE, 2009). Outras pesquisas de mesmo âmbito, constataram que, de uma forma geral, cerca de 71,9% da população têm medo de ter a residência invadida ou roubada, e esses são crimes que geram maior medo na população (Pesquisa SENASP, 2013).

Por conta dessas pesquisas mostrarem dados alarmantes, novas tecnologias têm sido introduzidas no mercado de segurança para tentar-se reduzir a criminalidade nesse setor. Mesmo assim, muitas empresas de segurança ainda trabalham com sistemas, de certa forma, arcaicos e que muitas vezes não se provam efetivos. Existem várias opções para implementar um sistema de segurança, seja utilizando câmeras, sensores, vigilantes ou sistemas remotos. Entretanto, esses sistemas precisam mostrar-se efetivos e prevenir ao máximo qualquer tipo de criminalidade.

Podemos ver que muitos dos sistemas usados atualmente baseiam-se ainda no princípio de monitoramento local, com sensores de presença, sensores de abertura ou cercas elétricas que, quando acionados, ativam um sinal sonoro por exemplo. Outros sistemas também vão além, acionando a central de monitoramento da empresa de segurança caso aconteça algo na residência. Por sua vez, a central de monitoramento avisa ao usuário sobre o ocorrido. Sistemas mais avançados geralmente são mais caros por conter tecnologias proprietárias, o que dificulta a aquisição de tais sistemas por usuários de classe média.

A internet está entrando em fase da popularização. A queda no preço dos microcomputadores fez com que mais usuários com menor poder aquisitivo começassem a ter acesso a rede mundial (MARTINS, 2007). Atualmente muitas residências já possuem acesso à internet. Esse acesso, entretanto, é geralmente feito apenas por pessoas usando microcomputadores para acesso a redes sociais, realizar pesquisas on-line ou leitura de notícias. Sendo assim, quase todos os 50 *petabytes* de dados disponíveis que circulam hoje na internet foram criados por seres humanos. O problema é que seres humanos não são muito bons em processar dados provenientes de coisas do mundo real. Pessoas tem tempo, atenção e acurácia limitados (ASHTON, 2009). Um sistema de segurança residencial trabalha com dados, sendo eles imagens, vídeos ou estados de sensores. Deixar esses dados inteiramente sobre responsabilidade de interpretação de seres humanos já se provou não ter a melhor das eficácias.

Proporcionar aos objetos de segurança, como câmeras e sensores, a possibilidade de realizar interpretação de dados e também possibilitá-los de conectar-se a internet é a ideia por trás da Internet das Coisas (IoT). É uma nova perspectiva

sobre conexão em qualquer tempo e em qualquer lugar, que agora também considera a comunicação das coisas (DINIZ, 2006).

Baseando-se nas ideias acima, surge o conceito de criar-se um protótipo de sistema de segurança residencial que visa preencher a deficiência dos atuais sistemas. O objetivo tratado, é criar um sistema de certa forma inteligente e autônomo que, com uso de sensores, câmeras e visão computacional, detecta alterações no meio em que estão instalados. De forma autônoma o sistema realiza diferentes decisões de tomada a partir dos dados colhidos.

II. REFERENCIAL TEÓRICO

2.1 Internet of Things (IoT)

Internet of Things, traduzida para o português como Internet das Coisas, está aumentando a conectividade entre pessoas e coisas, e volume de dados gerados que a tempos atrás era inimaginável. Uma explosão de dispositivos e novas aplicações combinados com o alcance e força da Internet possibilita novas interações inteligentes entre as coisas (GREEN, 2014).

Do ponto de vista do usuário, os efeitos mais óbvios da introdução da IoT serão visíveis em ambos ambientes empresarias e domésticos. Alguns exemplos nesse contexto são: domótica, *e-health* e aprendizado melhorado. Já na perspectiva de negócios, as consequências da IoT serão visíveis em áreas como, automação, manufatura industrial, logística, gerenciamento de processos e transporte inteligente de pessoas e bens (ATZORI, 2010).

Em um sistema IoT, dados são gerados por vários tipos de dispositivos, processados de diversas maneiras, transmitidos para diferentes localidades, e servem como base para que aplicações realizem tomadas de decisão. Em geral, quando dados são gerados sob o controle de máquinas e equipamentos, e são enviados por uma rede, provavelmente é um sistema IoT (GREEN, 2014).

Para se ter uma ideia do atual estado da Internet das Coisas, em 2011, o número de dispositivos interconectados no planeta ultrapassou o atual número de pessoas. Atualmente existem 9 bilhões de dispositivos interconectados e é esperado que esse número atinja 24 bilhões até o ano de 2020 (GUBBI, 2013).

A figura 1 ilustra um modelo de referência IoT e suas camadas. Os dados trafegam em ambas direções, sendo que as informações de controle trafegam do topo do modelo até o nível mais baixo e as informações de monitoramento podem trafegar em ambas as direções. Pode-se notar que no nível 1 temos as "coisas" como sensores, dispositivos e máquinas que geram dados. Os dados passam pelo nível 4 onde se encontram os bancos de dados e então seguem para análise no nível 6 até chegar no nível 7 que é o nível de aplicação. Em aplicações IoT mais complexas os dados podem passar por outros níveis.

Figura 1 – Modelo de referência IoT e suas camadas.

Fonte: IoT Reference Model, 2014.

2.2 Processamento de Imagens Digitais e Visão Computacional

Os interesses em métodos de processamento de imagem digital se derivam de duas principais áreas de aplicação: o aperfeiçoamento das informações de imagem digital para interpretação humana; e processamento de imagem para armazenamento, transmissão e representação para interpretação autônoma por máquinas (GONZALES, 2002). Pode-se dizer que processamento de imagens consiste de um sistema em que a entrada se dá por uma imagem e a saída é um conjunto de valores numéricos que podem ou não compor uma outra imagem. Já a visão computacional, por outro lado, tem o objetivo de emular a visão humana, portanto também precisa de uma imagem como entrada, entretanto, a saída consiste em uma interpretação dessa imagem (MARENGONI, 2009).

Com o objetivo de prover uma infraestrutura de visão computacional simples de usar e de ajudar as pessoas a desenvolverem aplicações bastante sofisticadas de maneira rápida, foi criada uma biblioteca escrita em linguagem de programação C/C++. Esta biblioteca denomina-se OpenCV e pode ser executada em Linux, Windows e Mac OS X. A biblioteca OpenCV possui mais de 500 funções que se aplicam em várias áreas de visão incluindo inspeção de produto em fábrica, imagens na medicina, calibração de câmera e robótica (BRADSKI, 2008).

2.3 Single Board Computer (SBC) e Computer on Module (COM)

Um *Single Board Computer* é uma placa de circuito impresso que contém um computador completo, incluindo clock, microprocessador, memória RAM, memória Flash, Ethernet, controle I/O e *sockets* (*PC Magazine Encyclopedia*, 2016). Um *Single Board Computer* é um

computador que requer apenas uma fonte de energia para operar. Já um Computador em Módulo (COM) ou também chamado de *System on Module* (SOM), tipicamente não contém *sockets* de I/O para conexão direta com periféricos. Eles precisam ser inseridos em outros sistemas que contém *sockets* de I/O e completam as conexões com periféricos.

Os Computadores em Módulo possuem vantagens sobre os *Single Board Computers*, quando se necessita de um projeto altamente customizado de acordo com as necessidades do mesmo. Quando um SBC é usado, o projeto fica preso ás disponibilidades de I/O do SBC. O projeto também não tem flexibilidade no tamanho final do produto que depende muito das dimensões físicas do SBC. Já quando um COM é usado, apenas é escolhido o processador de acordo com a necessidade de processamento do produto, e então é feita uma placa base que contém apenas os I/O's necessários, tendo o tamanho necessário para o produto final (MOHAPATRA, 2015).

Computadores em Módulo podem ser usados em qualquer produto industrial que requer mais de um dos requisitos: suporte gráfico, memória externa, por volta de 100MHz de processamento, vídeo, áudio, I/O, WiFi, um amplo sistema operacional, etc. Um microcontrolador não poderia atender a todas essas necessidades (MOHAPATRA, 2016).

No trabalho desenvolvido, o computador embarcado é a peça essencial. É nele onde todo software e a lógica do sistema irá residir. Pelo fato da possibilidade de usar um sistema operacional que roda no computador embarcado, no caso usou-se Linux embarcado, é possível o acesso a uma variedade de bibliotecas, API's (*Application Programming Interface*) e linguagens de programação, o que não seria possível ou estaríamos muito limitados usando uma plataforma baseada em Arduino por exemplo.

III. MATERIAIS E MÉTODOS

Foi construída uma maquete simulando um bairro residencial, figura 2. Esta maquete consiste de 3 casas de madeira onde foi instalado o sistema proposto. O projeto é composto das teorias apresentadas anteriormente em conjunto com os equipamentos e software que serão apresentados a seguir. Utilizou-se dos recursos de hardware como também dos recursos de software apresentados a seguir.

Figura 2 – Modelo do projeto proposto.

Fonte: Elaborada pelo autor, 2016.

3.1 Materiais

Seguem os materiais usados para implementação do projeto.

- 1 Computador em Módulo Colibri iMX6 da Toradex
- 1 Placa base Íris da Toradex
- 1 Computador embarcado Beaglebone Black
- 1 Computador embarcado Raspberry Pi
- 2 Sensor infravermelho da Sparkfun
- 1 Câmera USB Logitech modelo C270

Computador em Módulo Toradex Colibri iMX6

Colibri iMX6DL, mostrado na figura 3, é um COM da Toradex baseado no processador iMX 6 Dual Lite da NXP. O iMX 6 DL é um processador com dois cores ARM *Cortex* A9 de 1GHz. Possui um controlador gráfico (GPU) Vivante GC880, além de várias opções de conectividade como I2C, SPI, UART, PWM, GPIO, entrada analógica, Ethernet, CAN e etc. O módulo, além do processador, também dispõe de 512MB de memória RAM e 1GB de Flash (TORADEX, 2016).

Optou-se por usar o COM Colibri iMX 6 DL pelo fato do mesmo possuir capacidade de processamento suficiente para realizar as tarefas de visão computacional com OpenCV deste trabalho. É no computador embarcado que residem todos os recursos de software apresentados neste trabalho.

Figura 3 – COM Colibri iMX6 da Toradex.

Fonte: Toradex, 2016.

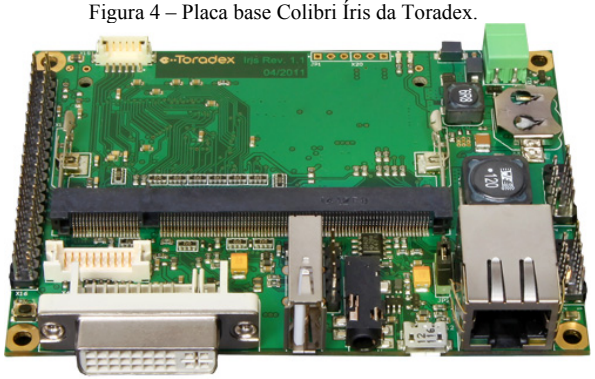
Placa base Íris da Toradex

A Toradex trabalha também com algumas placas base para seus módulos. Optamos por utilizar a placa base Colibri Íris, mostrada na figura 4. Essa placa base contém todos os requisitos necessários para o desenvolvimento do projeto incluindo, conector Ethernet para conexão com a internet, acesso a UART para desenvolvimento e entrada USB para a câmera.

Para reprodução e comercialização desse projeto, poderia ser desenvolvida uma placa base com design específico para o projeto. Poderíamos incluir mais entradas USB por exemplo.

No anexo B, figura 14, pode ser visto em detalhe a integração do COM Colibri iMX6 em conjunto com a placa base Íris.

Figura 4 – Placa base Colibri Íris da Toradex.



Fonte: Toradex, 2016.

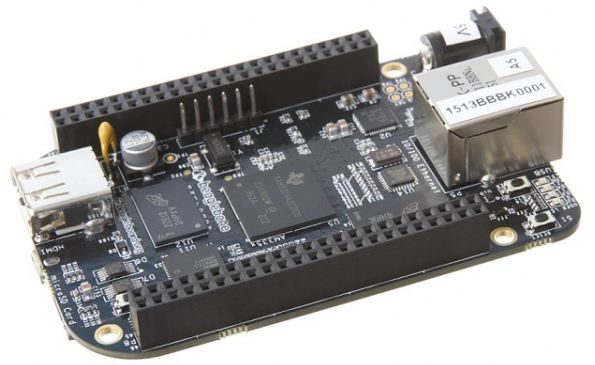
Computador embarcado Beaglebone Black

A Beaglebone Black, mostrada na figura 5, diferente do COM Colibri iMX6, é um SBC por si só. Em uma mesma placa eletrônica podemos encontrar o processador AM335x 1GHz ARM® Cortex A8, 512MB DDR3 RAM, 4GB armazenamento Flash, USB cliente, USB host, Ethernet, HDMI, e pinos I/O (ELEMENT14, 2014).

Foi usado o computador embarcado Beaglebone Black para exemplificar na prática o uso de um outro computador embarcado em outra residência, tendo assim uma rede de casas conectadas à nuvem.

A instalação do SBC Beaglebone Black pode ser vista em detalhe no anexo B, figura 13.

Figura 5 – SBC Beaglebone Black.



Fonte: Beagleboard, 2016.

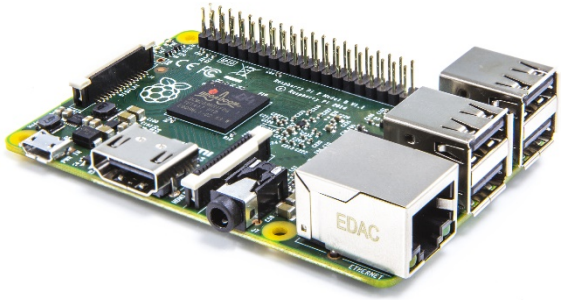
Computador embarcado Raspberry Pi

A placa Raspberry Pi, figura 6, é também um SBC com um grande foco em educação. Ela é bastante usada em escolas e universidades para o ensino de computação desde crianças até jovens e adultos.

A Raspberry Pi é baseada no processador Broadcom BCM2836 Quad Core de arquitetura ARM que trabalha a 900MHz. Tem 1GB de memória RAM e um *header* de 40 pianos com funções de GPIOS, I2C, SPI, UART e outros I/O's (ELEMENT14, 2016).

A instalação do SBC pode ser vista em detalhe no anexo B, figura 12.

Figura 6 – SBC Raspberry Pi 2 Modelo B.



Fonte: Raspberry Pi Org, 2016.

Sensor Infravermelho

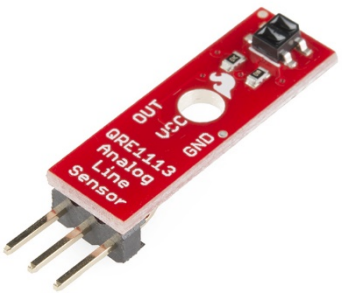
Para exemplificar os muitos sensores em que se pode encontrar em uma residência, foram utilizados em duas casas do modelo, um sensor infra vermelho. O Sparkfun RedBot Sensor, mostrado na figura 7, é baseado no sensor QRE11 que é, mais especificamente, um sensor refletivo de objetos. Ele trabalha medindo a quantidade de luz infravermelha refletida de seu próprio LED infravermelho. O sensor opera com uma tensão de 5V e possui uma saída analógica que varia de acordo com a quantidade de luz infravermelha refletida (FAIRCHILD, 2011). É um sensor bastante usado em robôs seguidores de linha.

Utilizou-se deste sensor para simular um sensor de movimento que é usado em residências para detectar intrusões. O software fará a leitura desse sensor que informará ao dono da residência sobre o ocorrido.

Quando o sensor é ativado, o mesmo envia um sinal para o computador embarcado que por sua vez altera um banco de dados na nuvem informando que o sensor foi ativado na específica residência. O computador embarcado também envia o status do sensor para o aplicativo Telegram do usuário.

A instalação do sensor pode ser vista em detalhe no anexo B, figura 11.

Figura 7 – Sparkfun RedBot Sensor.



Fonte: Sparkfun, 2016.

Câmera USB

Para utilizarmos OpenCV e visão computacional, é necessário um dispositivo de captura de imagem. Foi utilizada uma câmera USB HD da Logitech modelo C270, figura 8. O software faz a captura da imagem utilizando a câmera e então faz as chamadas de funções do OpenCV para detecção de padrões. No modelo proposto, quando o software detecta faces em um *stream* de vídeo o mesmo notifica o usuário com a imagem da face encontrada. Uma residência poderia fazer uso tanto do sistema de câmeras inteligentes, como também dos sensores de presença.

A câmera é conectada a placa base Íris utilizando uma porta USB.

Quando a câmera detecta uma face, o computador embarcado, além de enviar a imagem para o usuário via aplicativo Telegram, também envia ao banco de dados do Firebase uma solicitação para alteração do estado da casa no banco de dados.

No anexo B, figura 10, pode ser visto em detalhe a instalação da câmera no sistema.

Figura 8 – Webcam Logitech C270.

Fonte: Logitech, 2016.

Em conjunto com os componentes de hardware também foram usados os seguintes recursos de software:

- Sistema Operacional Linux Embarcado
- Linguagem de programação Python
- Biblioteca Python para visão computacional com OpenCV
- Biblioteca Python para Telegram Bots
- Biblioteca Python para Firebase
- API em Javascript para Firebase
- API em Javascript para Google Maps
- Hospedagem de servidor na nuvem com Amazon EC2
- Microsoft Azure IoT Hub, Stream Analytics e Power BI

Sistema Operacional Linux Embarcado

Linux se refere, tecnicamente falando, a um núcleo (*kernel*) de sistema operacional originalmente escrito por Linus Torvalds. Linux também se refere, de uma forma popular, ás distribuições de Linux que podem variar em propósito, tamanho, e preço. Na área dos embarcados também temos distribuições de Linux disponíveis embora o núcleo do sistema operacional é geralmente o mesmo para Desktops e sistemas embarcados. Linux embarcado pode estar presente em segmentos como aeroespacial, sistemas automotivos, eletrônicos de consumo, telecomunicações e etc. (YAGHMOUR, 2008).

No COM Colibri iMX 6 foi instalada uma imagem customizada do sistema operacional Linux. Optamos por utilizar Linux pelo fato do mesmo ser um SO aberto que pode ser facilmente customizado de acordo com a necessidade do projeto.

Para gerar uma imagem customizada, foram utilizadas as ferramentas do OpenEmbedded. Com OpenEmbedded podemos escolher entre os milhares pacotes que podem ser cross-compilados para nossa plataforma ARM. Em nossa imagem Linux temos suporte à linguagem Python, além das bibliotecas de OpenCV para detecção de faces.

Linguagem de Programação Python

Uma linguagem de alto nível, Python tem uma sintaxe simples e é *open source*. As bibliotecas padrão do Python já possibilitam a execução de uma vasta gama de recursos de software como banco de dados, web browsers, CGI, FTP, e-mail, HTML, interfaces gráficas de usuário e etc. Além das bibliotecas padrão, é possível a instalação de outros pacotes de biblioteca que podem ser encontrados no Python Package Index.

Para a execução desse projeto foram instaladas no sistema operacional Linux embarcado os pacotes de Python relacionados a OpenCV, Firebase e Telegram Bots. Cada pacote tem o propósito de possibilitar o desenvolvimento de cada recurso de software deste projeto.

Biblioteca Python para Visão computacional com OpenCV

Dentre as muitas linguagens que o OpenCV suporta, está a linguagem Python. Com uma biblioteca de funções que facilita bastante o desenvolvimento, foi possível implementar um trecho de código que detecta uma face, desenha um retângulo indicando a posição da face detectada e então salva a imagem na memória do computador embarcado para futuros trabalhos, como o envio dessa imagem para o aplicativo Telegram.

OpenCV se baseia originalmente na linguagem C/C++. As funções de Python para OpenCV são rápidas assim como as funções originais em C/C++ pelo fato de Python apenas fazer um serviço de abstração. Ou seja, quando um programa de visão computacional escrito em Python é executado, as funções originais de OpenCV em C/C++ são executadas em segundo plano (MORDVINTSEV, 2013).

Biblioteca Python para Telegram Bots

Telegram é um aplicativo de mensagens gratuito com foco em velocidade e segurança. É possível usar o telegrama em vários dispositivos ao mesmo tempo. As mensagens são

sincronizadas automaticamente entre *smartphones*, *tablets* e computadores. Com Telegram é possível enviar além de mensagens, fotos, vídeos e arquivos em vários formatos, criar grupos privados ou públicos e criar os chamados "*Telegram Channels*", onde é possível transmitir mensagens para grandes audiências.

Para os desenvolvedores de software, o Telegram, além de ser open-source, traz um serviço de Bots que são aplicações de terceiros que rodam dentro do Telegram. Os usuários podem interagir com um Bot através de mensagens e comandos.

Mais especificamente, Telegram Bots são contas especiais que não requerem um número de celular para serem configuradas. Para interagir com um Bot basta apenas iniciar uma conversa abrindo um chat com o Bot. As mensagens e comandos enviados pelo usuário passam pelo servidor com encriptação via interface HTTPS. Essa interface é chamada Bot API.

Existe uma biblioteca para Python que traz os recursos de Bots do Telegram para uso na linguagem Python. Essa biblioteca suporta a maioria dos recursos da API do Telegram Bots entre eles, enviar mensagem, enviar foto, etc.

Aproveitando os recursos do Telegram Bots, pode-se interagir com a residência de uma forma mais "humana". Foram desenvolvidos três softwares de Bot independentes que residem em cada computador embarcado do modelo. Os Bots são responsáveis por notificar ao usuário indicando estado de sensores e enviando fotos. Também informam a grupos e canais caso alguma residência na vizinhança esteja em alerta.

Biblioteca Python para Firebase

Firebase se utiliza de uma API do tipo REST (*Representational State Transfer*). Fizemos uso de uma biblioteca Python que, em segundo plano, trabalha com a API do Firebase. Com essa biblioteca podemos manipular todos os dados do banco de dados inclusive o mais importante que é o item "status". Este item recebe valores booleanos sendo *True* ou *False*.

Firebase e sua API Javascript

Firebase é um provedor de serviços na nuvem. Entre os serviços oferecidos temos banco de dados, autenticação de usuário, hospedagem e outros. O banco de dados do Firebase é armazenado como um objeto JSON e é sincronizado em tempo real com os clientes conectados. Todos os clientes conectados compartilham o mesmo banco de dados e recebem automaticamente uma atualização quando algum dado do banco de dados é alterado.

O banco de dados do projeto aqui apresentado contém informações das casas de uma determinada região. Essa região pode ser um bairro ou uma cidade. Esses dados consistem de valores como: endereço; número de identificação da residência; coordenadas geográficas; status.

Firebase disponibiliza uma API em Javascript. Com essa API é possível monitorar o banco de dados e tomar decisões caso haja mudanças no banco de dados.

Nosso banco de dados irá trabalhar em conjunto com um cliente que no caso é uma página do Google Maps. Mais detalhes sobre como usamos o Google Maps a seguir.

Google Maps JavaScript API

A Google oferece para desenvolvedores uma API do Google Maps também em JavaScript. Usamos essa API para exibir um mapa em uma página web e colocar *markers* em pontos específicos. Os *markers* ficam visíveis de acordo com o status da residência. Se o alarme é ativado, um *marker* aparece no mapa localizando a residência. A API do Firebase em Javascript é a responsável por monitorar o banco de dados e gerenciar o status "*visible*" de cada marker, seja *True* ou *False.*

Amazon EC2 – Hospedagem de Servidor Virtual

Amazon Elastic Compute Cloud (Amazon EC2) é um serviço que oferece capacidade de computação na nuvem. Com o Amazon EC2 podemos ter uma máquina virtual rodando o sistema operacional Linux com um servidor que hospeda nossa página com Google Maps. Nessa máquina virtual foi alocado o arquivo HTML em conjunto com arquivo Javascript que contém toda a lógica do Google Maps e Firebase. A máquina virtual fica ativa 24 horas disponibilizando a qualquer momento a página do Google Maps.

Microsoft Azure IoT Hub, Stream Analytics, Power BI

A Microsoft oferece alguns serviços de nuvem dentre eles o IoT Hub. Com o IoT Hub é possível conectar dispositivos IoT à nuvem de forma segura. É possível adicionar novos dispositivos e realizar a comunicação dispositivo-nuvem usando um protocolo bastante usado em IoT chamado MQ *Telemetry Transport* (MQTT). O serviço também possibilita configurar identidades individuais e credenciais para cada dispositivo conectado proporcionando confidencialidade de mensagens tanto nuvem-dispositivo como dispositivo-nuvem.

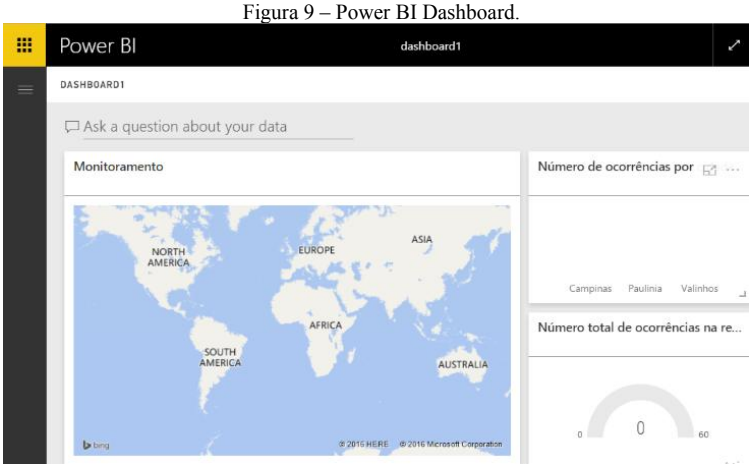
O *Stream Analytics* por usa vez trabalha em conjunto com o IoT Hub proporcionando análises em tempo real para soluções de IoT e fluxo de dados para outros serviços do Microsoft Azure como Power BI.

BI significa *Business Intelligence* e a Microsoft oferece o Power BI, que nada mais é do que um conjunto de ferramentas de análise de negócios para analisar dados e compartilhar ideias. É possível criar, em cima dos dados enviados pelo IoT Hub, vários relatórios e painéis dinâmicos chamados de *Dashboards* que possibilitam a análise e

visualização de dados. A partir da análise e visualização desses dados fica mais fácil de tomar decisões de mercado.

Aplicando os serviços da Microsoft foi possível enviar, usando o protocolo MQTT, os dados para o IoT Hub que por sua vez conecta-se ao *Stream Analytics*. Através do *Stream Analytics* os dados chegam ao Power BI em tempo real possibilitando a criação de alguns indicadores como número de ocorrências por cidade, número total de ocorrências e um mapa indicando o nível das ocorrências em cada cidade como visto na figura 9.

Figura 9 – Power BI Dashboard.



Fonte: Power BI, 2016.

3.2 Métodos

Aplicação da Internet das Coisas

O objetivo em foco deste artigo é tornar mais eficaz os sistemas de segurança residencial. Para tentar alcançar este objetivo, foram aplicados os conceitos básicos de IoT. O sistema foi distribuído em forma de camadas para assim, estar o mais próximo possível da arquitetura de IoT. Os recursos de hardware e software foram testados de forma individual e a partir daí foram sendo unificados no sistema como um todo.

Casa Como Objeto de IoT

A arquitetura IoT, tem como sua primeira camada os objetos ou dispositivos, sejam eles sensores, câmeras e etc., como já visto no referencial teórico. Em nosso projeto temos a casa como um todo, incluindo seus sensores e câmeras, na camada de dispositivos IoT. Duas casas do modelo contém o sensor infravermelho RedBot da Sparkfun e a terceira casa contém a câmera USB da Logitech. Sensores e câmera foram conectados ao computador embarcado.

Com os computadores embarcados presentes em cada uma das três residências do modelo, foi dado a cada uma delas a possibilidade de se conectar a nuvem via rede Ethernet. Isso significa que os dados gerados pelos sensores e câmera podem ser acessados de qualquer lugar do mundo se assim o software permitir.

Ainda na camada de dispositivos, reside o software que roda em cada computador embarcado. Esse software integra os recursos já apresentados anteriormente. Nos três computadores embarcados presentes nas três residências do modelo, o software é escrito em Python. Na residência onde a câmera USB está conectada ao COM Colibri iMX6, o software em Python integra as bibliotecas OpenCV, Firebase e Telegram Bots. Nos computadores Beaglebone Black e Raspberry Pi, foi feita a integração das bibliotecas Firebase e Telegram Bots apenas.

Camada de Comunicação

A comunicação entre as residências e a nuvem é feita através da própria internet existente na residência. Com a internet por si só, já resolve-se toda a problemática de comunicação com a nuvem. Os computadores embarcados estão conectados à internet da residência via Ethernet. A partir daí não nos importa como os dados chegarão ao destino na nuvem, seja wireless, fibra-ótica, torres de rádio, etc.

Para reproduzir essa rede de comunicação, foi compartilhada a conexão à internet de um computador, sendo este o Gateway, com os três computadores embarcados. Assim, cada computador embarcado recebeu um endereço IP local e acesso à internet.

Camada de Nuvem

É na nuvem onde reside o banco de dados do Firebase, o sistema de interação entre usuário e Bot do Telegram e a página dinâmica do Google Maps indicando o estado das residências caso os sensores ou câmera sejam ativados.

Cada residência tem seus dados gravados na nuvem no banco de dados do Firebase. Esses dados consistem em valores fixos como: endereço; número de identificação da residência; coordenadas geográficas; e também uma variável "*visible*" que varia entre "*True*" ou "*False*". A estrutura do banco de dados em formato JSON pode ser vista no anexo 2.

Nas duas casa onde existe apenas o sensor infravermelho, o software gerencia os sensores esperando dados do mesmo. Quando o sensor é ativado, ou seja, houve uma intrusão na residência, o software se conecta ao banco de dados e manipula a variável "*visible*" mudando-a para "*True*". Através de um botão podemos retornar essa varável para estado "*False*".

Já na casa onde a câmera está instalada, ao invés da variável "*visible*" ser manipulada baseando-se no estado de um sensor, a variável é manipulada quando uma face é detectada em seu campo de visão. Além de manipular o banco de dados, a câmera inteligente também salva e envia a

foto para o chat do Telegram. O usuário é logo notificado, seja em seu computador, *smartphone* ou *tablet*, recebendo a foto da face do intruso. Da mesma forma os sensores infravermelhos também notificam o dono da residência via Telegram que algo ocorreu.

Além do banco de dados no Firebase e recursos do Telegram, existe ainda uma página na web que mostra um mapa da região onde o sistema de segurança apresentado nesse artigo está instalado. Para alocar a página na nuvem, foram usados os serviços da Amazon Web Services como já visto anteriormente. Essa página fica alocada na nuvem em uma máquina virtual Linux. A página usa as API's do Google Maps em Javascript e também do Firebase para gerar um ponto no mapa nas coordenadas geográficas das residências. Lembrando que essas coordenadas estão entre os dados do banco de dados.

A página por sua vez monitora o banco de dados e quando algum sensor das casas é ativado, seja infravermelho ou câmera, a página coloca um ponto no mapa em cima da residência que foi invadida. No modelo feito para este artigo, quando são acionados os sensores e a câmera, três pontos devem aparecer no mapa.

Segurança Residencial na Camada de Aplicação Comercial IoT

É na última camada onde é visto o trabalho como um todo, ou seja, o objetivo de atingir a segurança residencial de uma forma eficaz e inteligente. Em IoT essa camada é muitas vezes chamada de camada de aplicação.

Com a página do Google Maps, é possível ter um sistema de segurança não mais individual mas sim, coletivo onde a vizinhança pode estar ciente se um vizinho se encontra em uma situação crítica. Foi criado um Telegram Channel onde os moradores de uma vizinhança podem se inscrever e receber mensagens. Quando um ponto aparece no mapa, uma notificação é enviada ao "Telegram Channel" da vizinhança avisando aos participantes do canal que uma casa da vizinhança pode estar em uma situação crítica. Os moradores por sua vez poderiam na medida do possível, verificar o que realmente está acontecendo e até mesmo notificar as autoridades. Para as vizinhanças que se utilizam de um vigia de bairro, ele poderia estar conectado a página monitorando as casas sem estar fisicamente olhando para as mesmas ou também fazer parte do canal e receber as notificações.

De outra maneira essa página poderia ter um cunho comercial e ser privada para uma empresa específica de segurança, onde poderia ser feito um monitoramento de uma certa região da cidade.

As informações mais detalhadas sobre as residências, como fotos e quais sensores foram acionados, são notificadas apenas para o dono da residência e não para o canal do Telegrama, ou seja, informações críticas não são notificadas para terceiros. O dono da residência por sua vez pode escolher criar grupos familiares no Telegram, informando mais de uma pessoa da família em caso de emergência.

Com os serviços da Microsoft aplicados ao projeto, como IoT Hub, *Stream Analytics* e Power BI, é possível gerar conteúdo a partir dos dados, ou seja, gerar informação relevante que pode ser usada em serviços de segurança de prefeituras, polícias municipais ou estaduais.

IV. RESULTADOS

Foi feita a montagem e integração dos recursos de hardware e software obtendo-se o seguinte resultado mostrado em foto no Anexo B, figura 15.

Após a montagem e integração de todos os recursos, tanto de *software* como de *hardware,* o sistema foi testado obtendo-se o seguinte funcionamento: as casas enviam os dados à nuvem e essa por sua vez, armazena, processa, analisa e envia os dados para os dispositivos móveis, Google Maps e Power BI.

O Power BI fica disponível para prefeituras, polícia, e qualquer entidade governamental que deseja possuir estatísticas sobre assaltos em residências.

O usuário, ao sair de sua residência, apenas ativa o sistema que por sua vez notifica o usuário caso haja algum ocorrido. O usuário, quando notificado, recebe em seu aplicativo Telegram, o estado do sensor como também uma foto do intruso no caso da residência com câmera. Uma página do Google Maps fica disponível a comunidade indicando se alguma residência no bairro foi invadida.

No Anexo A encontra-se o diagrama de blocos do funcionamento do sistema.

V. CONCLUSÕES

Conclui-se que foi possível implementar um modelo de sistema de segurança residencial utilizando-se dos recursos de software como linguagem de programação Python e OpenCV para visão computacional, e recursos de nuvem como banco de dados, servidor virtual, serviços IoT e aplicativo Telegram.

O custo dos computadores embarcados variou entre R$200 e R$600, o que é comparável a alguns modelos de centrais de alarme existentes no mercado. Porém, com um computador embarcado temos fácil conexão à internet e grande possibilidade de expansão do projeto.

Com relação ao custo dos sensores, este seria equivalente aos sensores utilizados em sistemas de segurança residencial convencionais.

O custo dos serviços de nuvem dependeria do número de residências utilizando o sistema, pois quanto maior o número de residências utilizando o sistema, menor seria o custo. Isso se deve ao fato de que os serviços de nuvem podem suportar um número muito grande de residências.

O Telegram, por ser um aplicativo gratuito, não encarece o custo do sistema.

Em termos de custo pode-se concluir que seria necessário um estudo mais profundo para saber-se o real valor de implementação do sistema embora estima-se que o custo seria equivalente a um sistema de segurança convencional, mas o valor agregado ao projeto proposto seria muito grande pelo fato do projeto estar pronto para expandir no mercado emergente de IoT.

Pode-se concluir que os recursos de hardware utilizados mostraram-se adequados ao sistema. O COM Colibri iMX6

mostrou-se adequado para atividades de detecção de faces com OpenCV por possuir processamento suficiente para tal.

Com a análise do modelo em funcionamento, pode-se concluir que é possível aperfeiçoar os atuais sistemas de segurança residencial através da aplicação da Internet das Coisas e visão computacional tornando o sistema mais colaborativo, inteligente e eficaz. Assim, fica claro que os resultados mostram que pode-se dar autonomia para os dispositivos avaliarem situações e tomarem decisões sem a interferência humana. Mostrou-se também que é possível que cada morador da vizinhança participe da segurança de uma forma coletiva tornando uma região mais segura.

Entre as propostas para trabalhos futuros e melhorias que poderiam ser aplicadas ao projeto realizado temos: implementação de um sistema de bateria em caso de quedas de energia da residência; implementação de conexão 3G/4G em caso de queda na internet da residência; e utilização dos computadores em módulo da Toradex em todas as residências, pelo fato da possibilidade de customização da placa base atendendo os requisitos e *design* de *hardware* exatos do projeto.

Foi possível aplicar de forma prática os conhecimentos de lógica de programação, linguagem de programação, eletrônica digital e outros adquiridos de forma teórica no curso de Engenharia de Automação e Controle do Centro UNISAL.

VI. AGRADECIMENTOS

Agradeço acima de tudo à Deus, por ter me proporcionado capacidade para produzir este trabalho e também por ter me dado inteligência para compreensão dos assuntos abordados durante o curso de Engenharia de Automação e Controle.

Agradeço a minha família pela paciência e pelo suporte nos tempos de estudo.

Agradeço também aos professores da UNISAL, pelo excelente ensino e por despertarem em mim o interesse pelo estudo e pela carreira na área de engenharia.

REFERÊNCIAS

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Características da vitimização e do acesso à justiça no Brasil**. Rio de Janeiro, 2010. Disponível em <http://www.ibge.gov.br/home/estatistica/populacao/vitimizacao_acesso_justica_2009/pnadvitimizacao.pdf> Acesso em 20 mar. 2016.

SECRETARIA NACIONAL DA SEGURANÇA PÚBLICA DO MINISTÉRIO DA JUSTIÇA. **Pesquisa Nacional de Vitimização**. 2013.

MARTINS, R. **Começa a popularização da internet**. Jornal Estadão, edição Agosto: pag. 50. São Paulo, 2007.

ASHTON, K. **That 'Internet of Things' Thing.** RFID Journal, edição Junho, 2009.

DINIZ, E. H. **Internet das Coisas**. Revista *GVexecutivo* 5, no. 1: pag. 59, 2006.

GREEN, J. **IoT Reference Model**. IoT World Forum Resources. Junho, 2014.

ATZORI, L. **The Internet of Things: A survey**. Jornal *Computer Networks,* volume 54, edição 15, pag. 2787-2805. Junho, 2010.

GUBBI, J. **Internet of Things (IoT): A vision, architectural elements, and future directions**. Future Generation Computer Systems 29.7, pag. 1645-1660. 2013.

GONZALEZ, R. C. & WOODS, R. E. **Digital Image Processing**. 2002.

MARENGONI, M. & STRINGHINI, S. **Introdução a Visão Computacional usando OpenCV**. Revista RITA v. 16, no. 1: pag. 125, 2009.

BRADSKI, G. & KAEHLER, A. **Learning OpenCV**. O'Reilly Media, Inc. Setembro, 2008.

PC MAGAZINE ENCYCLOPEDIA. **Definição de Single Board Computer.** Disponível em <http://www.pcmag.com/encyclopedia/term/51386/single-board-computer>. Acessado em Março, 2016.

MOHAPATRA, P. **Why choose System on Module over SBC for embedded product development.** Toradex Blog. Outubro, 2015.

MOHAPATRA, P. **What can you build with System on Module (SOM)?** Toradex Blog. Janeiro, 2016.

TORADEX. Datasheet: **Colibri iMX6**, Ver. 1.2. 2016.

ELEMENT14. **Beaglebone Black System Reference Manual**. Pag. 26. Abril, 2014.

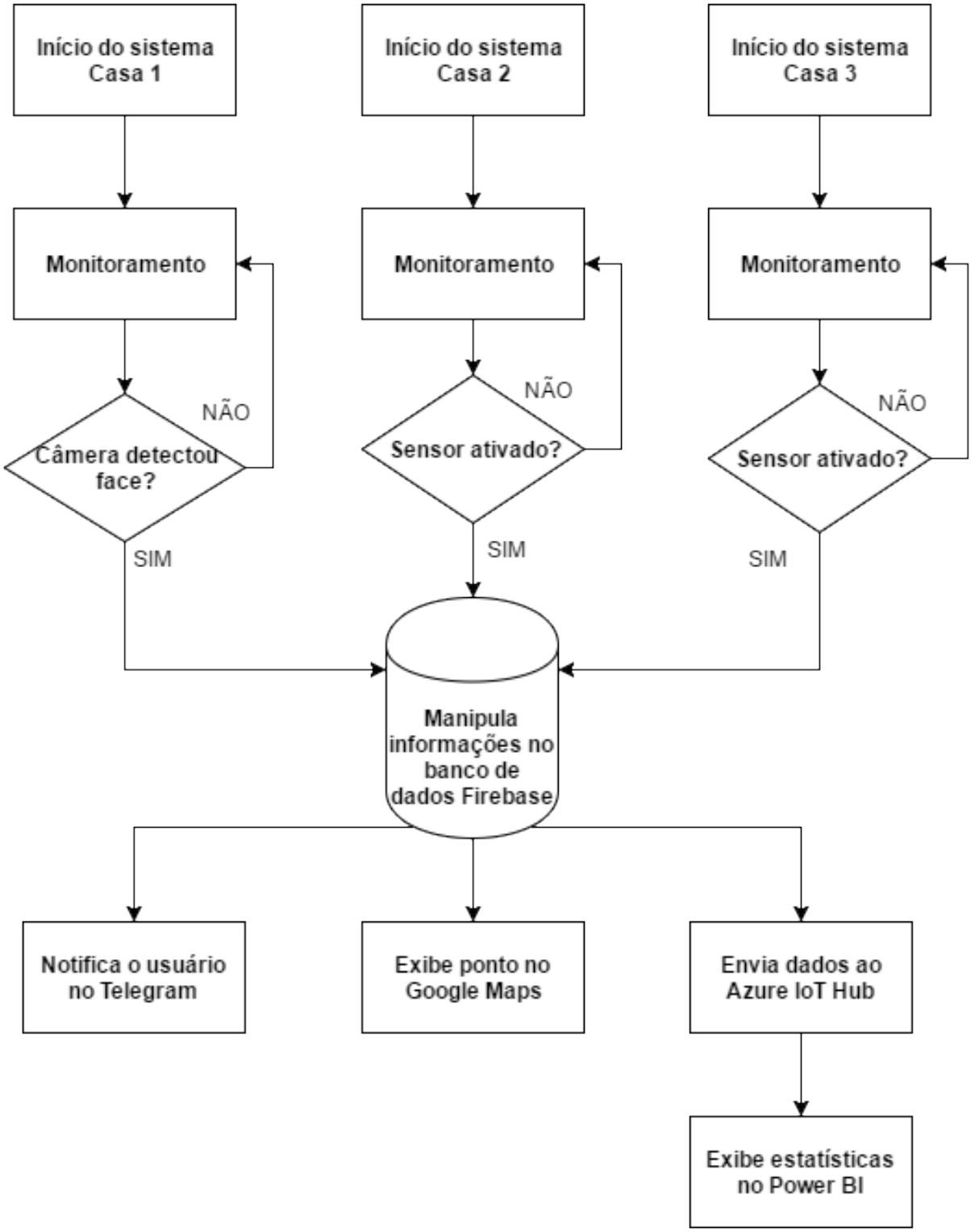
ELEMENT14. **Rapsberry Pi 2 Model B Technical Specification**. Fevereiro, 2016.

FAIRCHILD. Datasheet: **QRE1113 Miniature Reflective Object Sensor**. Agosto, 2011.

YAGHMOUR, K. **Building Embedded Linux Systems**, 2ª Ed. O'Reilly Media. Agosto, 2008.

MORDVINTSEV, A. & RAHMAN, A. **Introduction to OpenCV-Python**. 2013.

ANEXO A

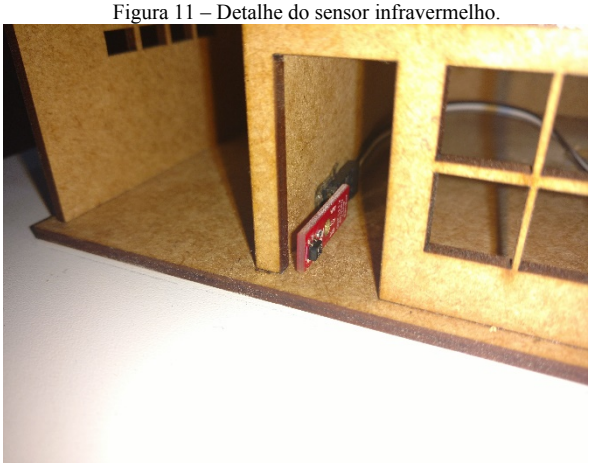

Início do sistema Casa 1
Início do sistema Casa 2
Início do sistema Casa 3
Monitoramento
Monitoramento
Monitoramento
NÃO
Câmera detectou face?
NÃO
Sensor ativado?
NÃO
Sensor ativado?
SIM
SIM
SIM
Manipula informações no banco de dados Firebase
Notifica o usuário no Telegram
Exibe ponto no Google Maps
Envia dados ao Azure IoT Hub
Exibe estatísticas no Power BI

ANEXO B

Figura 10 – Detalhe da câmera USB.

Fonte: Elaborada pelo autor, 2016.

Figura 11 – Detalhe do sensor infravermelho.

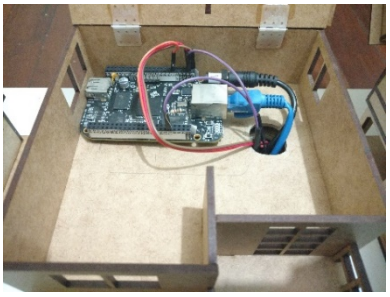


Fonte: Elaborada pelo autor, 2016.

Figura 12 – Instalação do SBC RaspberryPi.

Fonte: Elaborada pelo autor, 2016.

Figura 13 – Instalação do SBC Beaglebone Black.

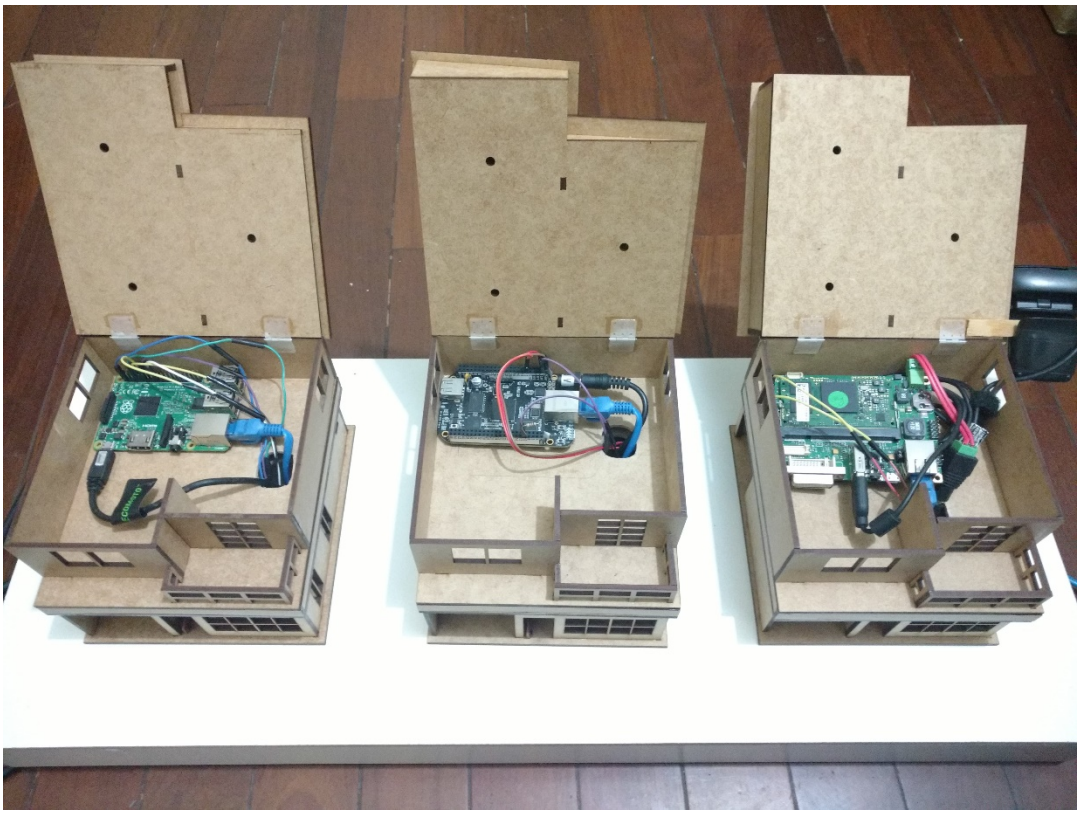


Fonte: Elaborada pelo autor, 2016.

Figura 14 – Instalação do SBC Toradex.

Fonte: Elaborada pelo autor, 2016.

Figura 15 – Instalação geral do sistema.



Fonte: Elaborada pelo autor, 2016.