

**FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS
FATEC PROFESSOR JESSEN VIDAL**

RAQUEL NATALI MARIZ

SAFEHOME 1.0

São José dos Campos
2016

RAQUEL NATALI MARIZ

SAFEHOME 1.0

Trabalho de Graduação apresentado à Faculdade de Tecnologia São José dos Campos, como parte dos requisitos necessários para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Me. Antônio Egydio Graça

São José dos Campos
2016

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

NATALI MARIZ, Raquel
SafeHome 1.0.
São José dos Campos, 2016.
49f. (número total de folhas do TG)

Trabalho de Graduação – Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, FATEC de São José dos Campos: Professor Jessen Vidal, 2016.
Orientador: Me. Antonio Egydio Graça.

1. Áreas de conhecimento. I. Faculdade de Tecnologia. FATEC de São José dos Campos: Professor Jessen Vidal. Divisão de Informação e Documentação. II. Título

REFERÊNCIA BIBLIOGRÁFICA –

NATALI MARIZ, Raquel. **SafeHome 1.0**. 2016. 49f. Trabalho de Graduação - FATEC de São José dos Campos: Professor Jessen Vidal.

CESSÃO DE DIREITOS –

NOME DO AUTOR: Raquel Natali Mariz
TÍTULO DO TRABALHO: SafeHome 1.0
TIPO DO TRABALHO/ANO: Trabalho de Graduação / 2016.

É concedida à FATEC de São José dos Campos: Professor Jessen Vidal permissão para reproduzir cópias deste Trabalho e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte deste Trabalho pode ser reproduzida sem a autorização do autor.

Raquel Natali Mariz

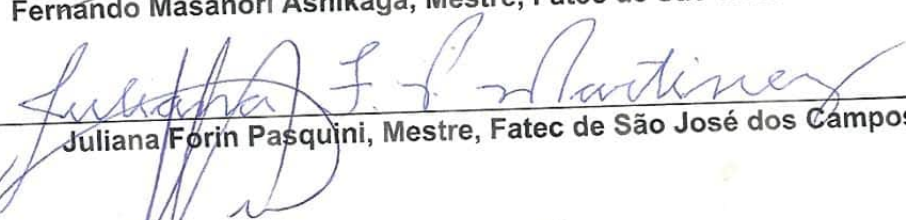
SAFEHOME 1.0

Trabalho de Graduação apresentado à Faculdade de Tecnologia São José dos Campos, como parte dos requisitos necessários para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Composição da Banca



Fernando Masanori Ashikaga, Mestre, Fatec de São José dos Campos



Juliana Forin Pasquini, Mestre, Fatec de São José dos Campos



Antonio Egydio Graça, Mestre, Fatec de São José dos Campos

09 / 12 / 2016

DATA DA APROVAÇÃO

Dedico este trabalho a minha avó Rita Amélia de Moraes Natali e aos meus pais que sempre me apoiaram, Raquel Denise Natali e Wilson Luiz Souza.

AGRADECIMENTOS

Agradeço ao meu orientador Egidio por todo o apoio, paciência e todas as ideias que me deu para o desenvolvimento deste trabalho, desde a concepção do tema até a parte final de sua construção.

Também agradeço a todos os professores que tive a honra de conhecer durante toda a minha jornada no curso de Análise e Desenvolvimento de Sistemas pois todos, sem exceção, contribuíram de alguma forma na minha vida, no meu aprendizado e na minha visão de mundo.

Não posso me esquecer também de todos os contribuidores e colegas anônimos que me ajudaram na construção deste aplicativo, que me deram seu tempo e sua disposição para sanar dúvidas mesmo apenas me conhecendo pelo meu nome de usuário em um fórum praticamente anônimo tanto na comunidade Kivy como na comunidade de makers do Arduino, entre outros.

Por fim, agradeço a Deus, a meus pais e a minha avó, pois sem eles não estaria aqui e sem o seu apoio, nunca conseguiria me tornar a pessoa que me tornei e ainda estou me tornando. Muito obrigada mesmo!

“Com cada avanço da Internet das Coisas, a distinção do real e do virtual começa a ficar menos clara, às vezes até mesmo de um modo criativo.”

Geoff Mulgan

RESUMO

A automação residencial está tornando-se cada dia mais popular, conectando objetos do dia-a-dia na rede mundial de computadores, facilitando a realização de atividades cotidianas e tornando o ambiente residencial mais seguro. O objetivo deste trabalho é apresentar o desenvolvimento de um aplicativo para dispositivos móveis que faça o monitoramento residencial e envie uma notificação no aparelho caso seja detectada alguma anormalia, como uma mudança súbita de temperatura ou de nível de umidade, ou um movimento em um cômodo que supostamente deveria estar vazio. Neste trabalho de graduação foram usados sensores de temperatura, de umidade e de movimento. Todos os sensores foram conectados ao Arduino, um microcontrolador que facilita a criação de projetos interativos utilizando componentes eletrônicos. O Arduino por sua vez, utilizou o *Ethernet Shield* para enviar os seus dados a um servidor online, de forma que o aplicativo possa acessá-los pela internet. No desenvolvimento foi utilizada uma tecnologia recente chamada Kivy, que permite a integração do aplicativo com diversas plataformas e sistemas operacionais.

Palavras-Chave: Arduino Uno, Automação Residencial, Monitoramento, Dispositivo Móvel.

ABSTRACT

Home Automation is increasing its popularity each day, connecting everyday objects in the World Wide Web, making chores more manageable and also homes safer. The goal of this project is to present the development of a mobile app that monitors homes and sends a notification if some sudden change is detected, like raises in temperature, drops in moisture or a movement in a room that should be empty. Sensors of humidity, temperature and movement were used in this project. All the sensors were connected to an Arduino Board, a microcontroller that helps in the creation of interactive projects where electronic components are used. This project also used the Ethernet Shield, to make the Arduino data available to its online server and accessible to the mobile app. Throughout the development, Kivy, a Python Library was used to allow easy integration and better compatibility with many platforms and operating systems.

Keywords: Arduino Uno, Home Automation, Monitoring, Mobile Device

SUMÁRIO

1	INTRODUÇÃO	10
1.1	MOTIVAÇÃO	10
1.2	OBJETIVO GERAL	12
1.3	OBJETIVOS ESPECÍFICOS	12
1.3.1	ABORDAGEM METODOLÓGICA	12
2	CONTEXTUALIZAÇÃO TECNOLÓGICA	13
2.1	Tecnologias Utilizadas	13
2.1.1	Arduino IDE	13
2.1.2	Kivy	14
2.1.2	Python	14
2.1.2	XML	14
2.2	Soluções Existentes	14
2.2.1	Home Assistant	14
2.2.2	HomeIO	15
2.2.3	SmartThings	15
2.2.4	Nest	15
2.3	Levantamento de Requisitos	15
2.3.1	Metodologia Utilizada	15
2.3.2	Requisitos Funcionais	16
2.3.2.1	Casos de Uso	16
2.3.2.2	Monitoramento Residencial	16
2.3.2.3	Transmissão de Dados	17
2.3.2.4	Sensores	17
2.3.3	Requisitos Não Funcionais	20
2.3.3.1	Desempenho	20
2.3.3.2	Disponibilidade	20
2.3.3.3	Usabilidade	21
2.4	Conceitos-chave para o entendimento do trabalho	21
2.4.1	A Definição de Internet	21
2.4.2	Internet das Coisas	22
2.4.3	Automação Residencial	24
3	DESENVOLVIMENTO	25

3.1	Arquitetura da Aplicação	25
3.2	Materiais Usados	26
3.2.1	Arduino Uno R3	26
3.2.2	Sensor de Umidade	27
3.2.3	Sensor de Movimento	27
3.2.4	Sensor de Temperatura	28
3.2.5	Ethernet Shield	28
3.3	Preparo do Ambiente de Desenvolvimento	29
3.4	Comunicação entre o Arduino e o Sensor de Umidade	31
3.5	Comunicação entre o Sensor de Temperatura e o Arduino	33
3.6	Comunicação entre o Sensor de Movimento e o Arduino	34
3.7	Arduino e Ethernet Shield	35
3.8	Desenvolvimento do Aplicativo	36
3.8.1	Recebimento de dados do Aplicativo	37
3.8.2	Protótipo do Relatório Geral	38
3.8.3	Elaboração do Layout do Aplicativo	38
3.9	Circuito Final	40
4	RESULTADOS	41
5	TRABALHOS FUTUROS	48
6	REFERÊNCIAS	49

1- INTRODUÇÃO

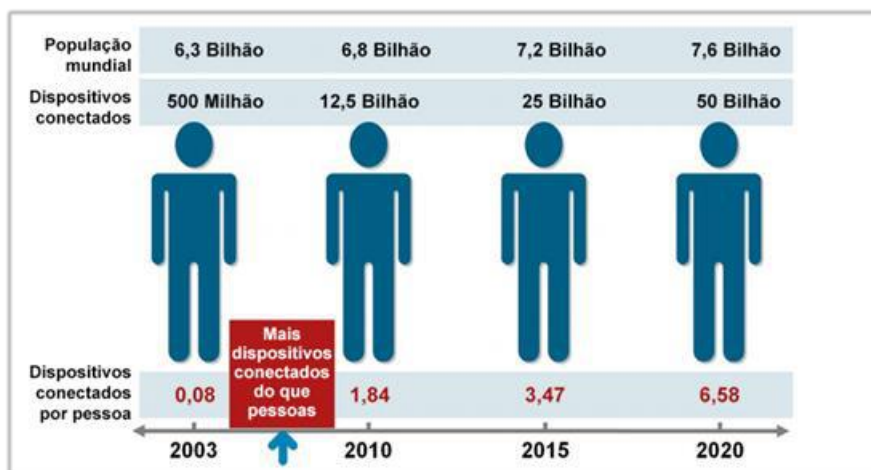
1.1 – Motivação

A Internet das Coisas é a integralização de objetos do cotidiano em redes interligadas, permitindo a coleta, troca e armazenamento de dados em uma nuvem, podendo gerar serviços em uma escala nunca vista (SBC, 2015). Considerada uma revolução tecnológica e com mercado mundial estimado em 1,7 trilhões de dólares em 2020 (IDC, 2015), ela gera impacto em diversos setores tal como indústria, saúde, educação e no modo como as pessoas consomem informação.

O especial da Internet das Coisas é como ela é interligada com diversas tecnologias. A diminuição de tamanho dos sensores facilita a coleta e transmissão de dados. Há uma estimativa em que haverá mais de 40 bilhões de dispositivos conectados em 2020 (ABI Research, 2013).

O mercado da computação em nuvem deve alcançar mais de 120 bilhões de dólares em 2018 (Forbes, 2015), justificando os 44 zettabytes de dados que serão manipulados diariamente no mundo em 2020 (EMC, 2014). A Figura 1.1 mostra o número estimado da população mundial e o número de dispositivos conectados a rede que ultrapassará esta população ao longo dos anos.

Figura 1.1 – Número de Pessoas e Dispositivos Conectados



Fonte: Cisco IBSG, abril de 2011

Com a conexão de objetos à rede, surgiu o conceito de automação residencial que pode ser definida como controle de funções encontradas no ambiente, deixando-o mais prático e até mesmo mais seguro (GDS, 2016). Uma das vantagens que é muito ilustrada por especialistas da área é a de poder, muitas vezes, economizar energia, pois dá a habilidade para o usuário de acender ou desligar luzes remotamente mesmo estando fora de casa ou determinar um horário específico para que aparelhos sejam ligados (ADT, 2016).

Também é muito ilustrado que a automação residencial faz as pessoas ganharem tempo e focarem em suas respectivas profissões e famílias ao eliminar as pequenas tarefas e ajustes que teriam que fazer caso sua residência não fosse automatizada (FRESHOME, 2016).

A Figura 1.2 exemplifica quais sensores poderão ser conectados à residência e as funções que exercerão.

Figura 1.2 – Sensores em uma residência conectados à Rede



Fonte: Adaptado de Instructables, 2016.

Com a automação residencial dando poder as pessoas de controlarem os objetos de suas casas, surgiu então o Arduino, um microcontrolador de código aberto, que permite com que aparelhos sejam conectados a um computador e a uma rede de uma maneira simples, eficiente e barata (ARDUINO, 2014).

1.2 – Objetivo Geral

Este trabalho tem como objetivo exemplificar a automação de uma residência, tornando uma casa inteligente com a instalação de sensores de umidade, temperatura e movimento e possibilitar o controle do usuário por meio de um dispositivo móvel que o alertará caso ocorra qualquer mudança brusca em seu ambiente.

1.3 – Objetivos Específicos

Os objetivos do projeto serão:

- a) Definição dos sensores utilizados.
- b) Definição da plataforma mobile que será utilizada e preparo do ambiente de programação para o aplicativo.
- c) Configuração do *Ethernet Shield*, tornando-o um servidor.
- d) Conexão do Arduino com os sensores.
- e) Conexão do *Ethernet Shield* com o arduino.
- f) Desenvolvimento do aplicativo.
- g) Transmissão dos dados para o aplicativo.

1.3.1 – Abordagem Metodológica

No desenvolvimento deste trabalho, serão utilizados o microcontrolador Arduino para possibilitar interação de objetos físicos (*hardware*) com programas de computador (*software*), um *Ethernet Shield* conectado ao Arduino para ser utilizado como um servidor transmitindo dados para o aplicativo mobile via rede e os sensores de temperatura, umidade e movimento para coletarem dados do ambiente.

2 – Contextualização Tecnológica

Este capítulo apresentará os modelos de caso de uso, requisitos de sistema e diagramas de UML e tecnologias utilizadas para o desenvolvimento do sistema e nas seções finais, abordará assuntos importantes de maneira sucinta que serviram como base deste trabalho de graduação. São eles: Conceito de Internet, Internet das Coisas e Automação Residencial.

2.1 – Tecnologias Utilizadas

Para facilitar e tornar possível o desenvolvimento deste trabalho de graduação, foram utilizadas as seguintes tecnologias:

Nome	Versão	Licença
Arduino IDE	1.6	Arduino
Kivy	1.9.1	MIT
Python	2.7.2	PSF
XML	1.0	Apache

2.1.1 – Arduino IDE

O ambiente de desenvolvimento da IDE do Arduino contém um editor de texto para escrever códigos, uma área de mensagens, um terminal, uma barra de ferramentas com botões para funções em comum e uma série de menus conectando o *hardware* do Arduino fazendo o upload e a comunicação entre ele e os programas escritos neste ambiente (arduino.cc, 2016).

2.1.2 - Kivy

Kivy é uma biblioteca Python para o desenvolvimento rápido de aplicações que fazem uso de interfaces inovativas para usuários, como aplicativos *multi-touch* e tem suporte para inúmeras plataformas tais como Windows, Linux, OS X, Android e iOS (kivy.org, 2016).

2.1.3 - Python

Python é uma linguagem interpretada, orientada a objetos, de alto nível com semântica dinâmica. Seu alto nível de construção de estruturas de dados combinado com escrita dinâmica a faz muito atrativa para desenvolvimento de aplicações rápidas e também para uma linguagem de script utilizada para interligar componentes (python.org, 2016).

2.1.4 - XML

XML é uma linguagem de marcação para a criação de documentos com dados organizados de maneira hierarquica (w3school, 2016).

2.2 – Soluções já existentes

Tabela 1 – Comparativo com Soluções Alternativas

Características	HomeAssistant	HomeIO	SmartThings	Nest
Mobile		x	x	x
Compatibilidade com diversos sistemas operacionais	x	x		x
Sensores Diversificados	x	x	x	
Relatórios			x	
Gratuito (app)	x	x	x	x
Opensource e comunidade ativa	x		x	

2.2.1 – Home Assistant

O Home Assistant é uma plataforma de automação residencial de código aberto que usa o Python3 para rastrear e controlar dispositivos em uma casa (home-assistant, .

2016). Requer conhecimentos avançados de programação mas há grandes opções de personalização.

2.2.2 – HomeIO

Atualmente, há poucas informações sobre o homeIO, mas no site do projeto é mencionado que será um aplicativo tanto para smartphones quanto para navegadores que permitirá que o usuário controle sensores remotamente (homeio, 2016).

2.2.3 - SmartThings

Aplicativo que funciona no sistema operacional iOS, que ao adquirir um hub por cerca de U\$99, se consegue controlar e customizar sensores de uma residência (smarththings, 2016).

2.2.4 – Nest

Ao comprar um termostato nest e fazer a instalação residencial (cerca de U\$ 250 mais a taxa opcional de instalação de U\$119), consegue-se controlar a temperatura da residência remotamente via aplicativo (hgtv, 2016).

2.3 – Levantamento de Requisitos

Esta seção mostra o levantamento de requisitos feito para desenvolver o projeto.

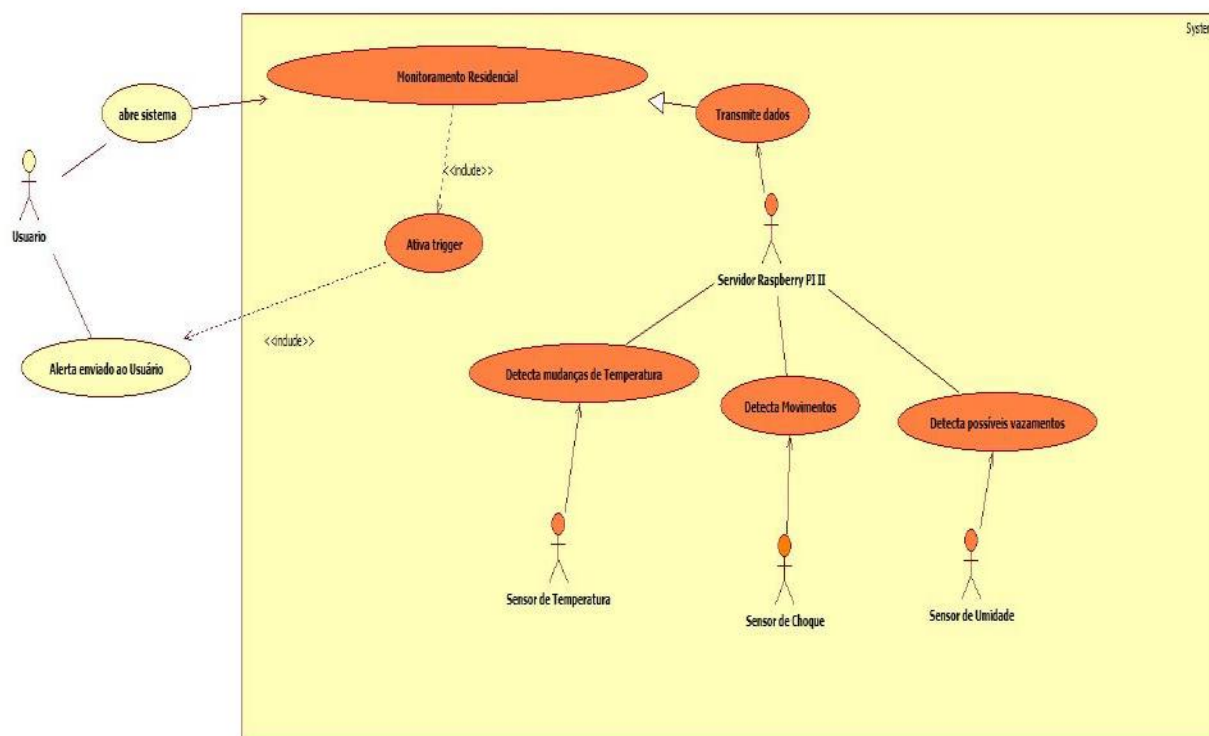
2.3.1 – Metodologia

Foram analisados diversos aplicativos que propõem solução semelhante para automação residencial e discutidos diversos aspectos deste projeto com o orientador.

2.3.2 – Requisitos Funcionais

2.3.2.1 – Casos de Uso

Figura 2.1 – Diagrama de Caso de Uso



Fonte: AUTOR(2016).

2.3.2.2 – Monitoramento Residencial

Ao abrir o aplicativo em um celular Android, o usuário se deparará com um menu com itens que listam os sensores instalados em uma residência. Ao clicar em um item, ele verá as informações pertinentes daquele sensor.

Tabela 2 – Caso de Uso: Monitoramento Residencial

Nome do Caso de Uso	Monitoramento Residencial
Ator Principal	Ethernet Shield
Resumo	Este caso mostra dados enviados do sensor para o Ethernet Shield que os disponibiliza no aplicativo
Pré-condições	Transmissão de dados realizada com sucesso
Fluxo Principal	
Ações do usuário	Ações do Sistema
1 – Clicar em um item do menu	
	2 – Mostrar dados do Sensor
Restrições	Deve-se ao menos ter um sensor instalado na Residência

2.3.2.3 – Transmissão de dados

A transmissão de dados será feita por sensores que estarão ligados em um microcontrolador Arduino que por sua vez estará conectado ao Ethernet Shield. Contanto que o aplicativo tenha acesso a internet, o servidor enviará os dados coletados para ele.

Tabela 3 – Caso de Uso: Transmitir Dados

Nome do Caso de Uso	Transmitir Dados
Ator Principal	Arduino
Resumo	Este caso mostra a transmissão de dados dos sensores para o servidor
Pré-condições	Executar casos de uso de detecções
Fluxo Principal	
Ações do Arduino	Ações do Sistema

1 – Enviar dados dos sensores ao servidor	
	2 – Retransmitir estes dados ao aplicativo
Restrições	Deve-se ao menos ter um sensor instalado na Residência e conexão com a internet
Fluxo de Exceção – Não consegue conectar-se a rede	
Ações do Arduino	Ações do Sistema
	1- Tentar retransmitir novamente
	2- Mostrar mensagem “Não conseguimos estabelecer conexão com o serviço.”

2.3.2.4 – Sensores

Os sensores instalados na casa estarão conectados ao Arduino que fará a conversão de seus sinais para dados. Os sensores farão toda a parte de monitoramento de temperatura, vazamento e movimentos e enviarão notificações caso seja detectada alguma anormalidade.

Tabela 4 – Caso de Uso: Detecção de Temperatura

Nome do Caso de Uso	Detecção de Temperatura
Ator Principal	Sensor de Temperatura
Resumo	Este caso ilustra o monitoramento de temperatura efetuada pelo seu respectivo sensor
Fluxo Principal	
Ações do Sensor	
1 – Detecção de Temperaturas	
2 – Transmitir dados ao controlador Arduino.	
Restrições	Deve-se estar propriamente conectado ao Arduino

Tabela 5 – Caso de Uso: Detecção de Umidade

Nome do Caso de Uso	Detecção de Umidade
Ator Principal	Sensor de Umidade
Resumo	Este caso ilustra o monitoramento de umidade efetuada pelo seu respectivo sensor
Fluxo Principal	
Ações do Sensor	
1 – Detecção de Umidade	
2 – Transmitir dados ao controlador Arduino.	
Restrições	Deve-se estar propriamente conectado ao Arduino

Tabela 6 – Caso de Uso: Detecção de Movimentos

Nome do Caso de Uso	Detecção de Movimentos
Ator Principal	Sensor de Movimento
Resumo	Este caso ilustra o monitoramento de movimentos efetuada pelo seu respectivo sensor
Fluxo Principal	
Ações do Sensor	
1 – Detecção de Movimentos	
2 – Transmitir dados ao controlador Arduino.	
Restrições	Deve-se estar propriamente conectado ao Arduino

Tabela 7 – Caso de Uso: Ativar Trigger

Nome do Caso de Uso	Ativar Trigger
Ator Principal	Sistema
Resumo	Este caso é acionado quando é detectada alguma anormalidade na residência, como uma mudança brusca de temperatura ou na porcentagem de umidade ou um movimento é detectado.
Fluxo Principal	
Ações do Sensor	
1 – Detecção de mudanças bruscas	
2 – Enviar notificação ao usuário.	
Restrições	Dados prévios para a comparação e detecção, conexão com internet.

2.3.3 - Requisitos Não-Funcionais

Os requisitos não-funcionais que foram levantados durante o desenvolvimento deste projeto serão apresentados nesta seção.

2.3.3.1 – Desempenho

O aplicativo deve acessar os dados mais recentes dos sensores, atualizando-se pelo menos, uma vez por minuto. O tempo de transição para as telas do aplicativo não deve demorar mais do que cinco segundos.

2.3.3.2 – Disponibilidade

Os dados dos sensores devem estar disponíveis durante todo o período em que eles estiverem conectados a internet.

2.3.3.3 – Usabilidade

O aplicativo deverá rodar adequadamente em dispositivos móveis.

2.4 – Conceitos-Chave para o entendimento do trabalho

2.4.1 – A definição de Internet

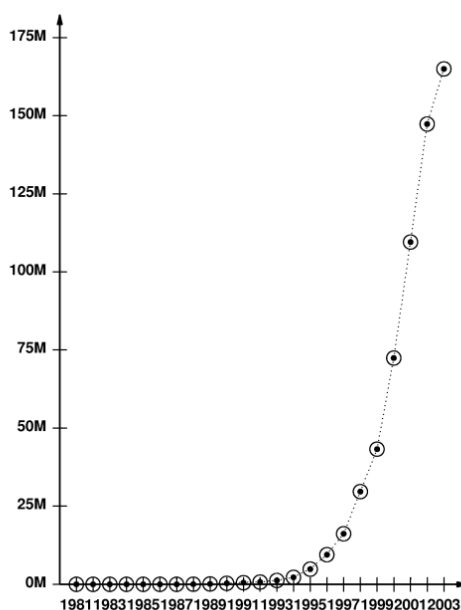
A internet é composta por milhões de redes interligadas seja em residências, empresas, universidades, governos, etc. Estes computadores utilizam protocolos em comum para que seja estabelecida uma comunicação (brasilecola.uol.com.br, 2016).

Desenvolvida durante a Guerra Fria (naquela época chamada de ArphaNet), tinha o propósito de manter a comunicação das bases militares dos Estados Unidos mesmo que o Pentágono fosse aniquilado por um ataque nuclear.

Ao término da Guerra Fria, o projeto teve acesso permitido para os cientistas que mais tarde cedeam a rede para as universidades, ganhando cada vez mais usuários (Cristina Kellen, 2016).

No final dos anos 80, um grupo do Laboratório de Partículas Europeu (CERN) desenvolveu um sistema que permitisse o compartilhamento de documentos científicos. Seus principais intuítos eram permitir o acesso remoto, acesso independente do sistema operacional, compartilhamento de arquivos e ligação a recursos externos. Em 1989, Tim Berners-Lee elaborou um sistema de hipertexto distribuído em rede, na qual a informação seria armazenada em documentos ligados. Surge então o protótipo da World Wide Web, mais semelhante a utilização que temos atualmente (PEUS, 2016).

Figura 2.2 – Aumento de computadores em rede com o passar dos anos



Fonte: PEUS, 2016.

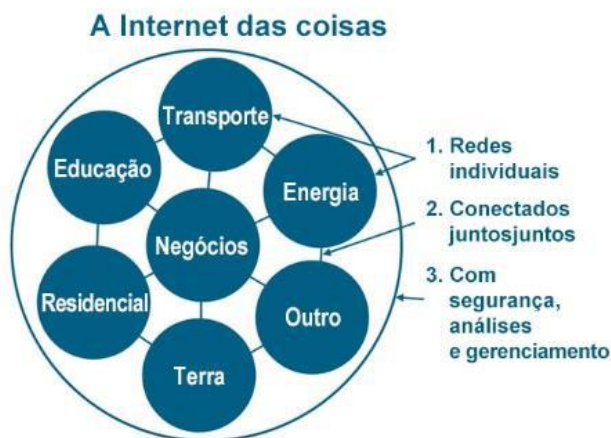
2.4.2 - Internet das Coisas

A ideia de conectar objetos entre si já foi colocada em pauta desde 1991, quando a conexão TCP/IP e a Internet que usamos hoje começou a se popularizar. O cofundador da Sun Microsystems, Bill Joy, sugeriu uma conexão de Device para Device (D2D).

Em 1999, Kevin Ashton do MIT propôs o termo “Internet das Coisas”. Segundo o especialista, a limitação do tempo e da rotina, fará com que as pessoas se conectem à Internet de maneiras diferentes (techtudo.com.br, 2014).

A figura 2.3 resume o que é a Internet das Coisas, mostrando diversos setores como redes individuais que ao serem conectados em conjunto, analisados e gerenciados, contribuem para um melhor desenvolvimento dos mesmos.

Figura 2.3 – IoT ilustrada como uma rede com várias redes em conjunto



Fonte: Cisco IBSG, abril de 2011.

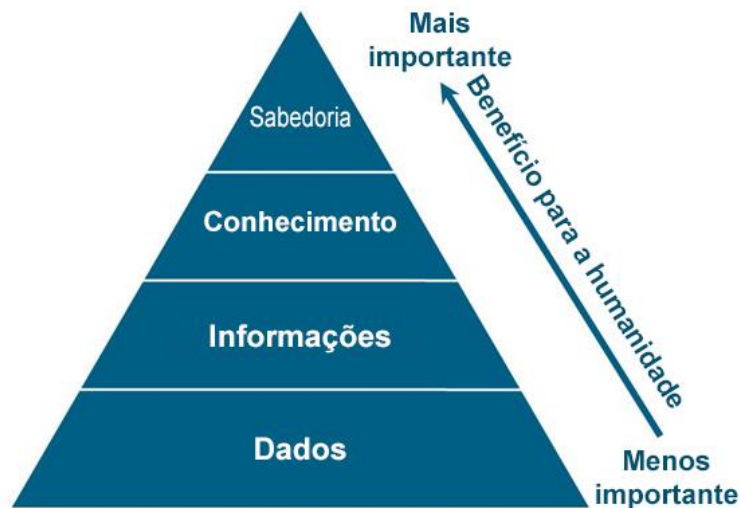
O conceito de IoT envolve três componentes principais:

1. Objetos
2. Redes de comunicação que os conectam
3. Sistemas computacionais que usam os dados que fluem destes objetos.

Com esta organização, objetos podem comunicar-se entre si e até mesmo otimizar a atividade que existe entre eles com base na análise de dados transmitidos entre eles (sas.com, 2016).

O princípio de compartilhamento de dados pode ser melhor entendido ao se observar a Figura 2.4. Os dados disponibilizados individualmente não são significantes, mas em números maiores ajudam a identificar certas tendências ou padrões que são usados para melhorar a qualidade da vida humana. (CISCO IBSG, 2011).

Figura 2.4 – Pirâmide ilustrando o processamento de dados.



Fonte: Cisco IBSG, abril de 2011.

Com o crescimento populacional, é importante dar o poder de administração de recursos para as pessoas. Ao combinar a capacidade de evolução da IoT de sentir, coletar, analisar e distribuir dados em grande escala, capacita ainda mais o acesso ao conhecimento para a humanidade, ajudando comunidades a prosperarem.

2.4.3 – Automação Residencial

Automação Residencial é o conjunto de serviços proporcionados por sistemas de tecnologia integrada como a forma mais almejada de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação (Henrique Paulo, 2016). O principal fator que define uma instalação residencial automatizada é a integração entre os sistemas aliada à capacidade de executar funções e comandos mediante instruções programáveis.

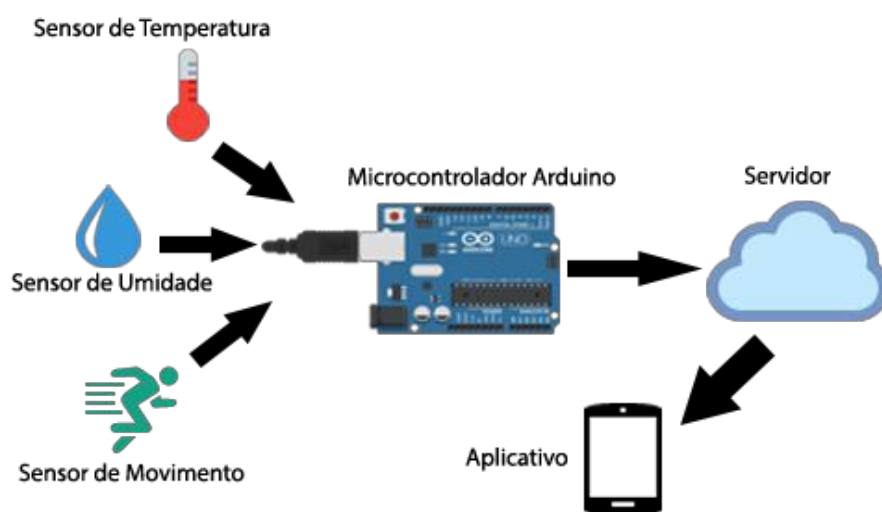
3 - DESENVOLVIMENTO

O objetivo deste capítulo é apresentar o desenvolvimento do aplicativo e está dividido nas seguintes seções: a Seção 3.1 irá apresentar a arquitetura do sistema usada no seu desenvolvimento, a Seção 3.2 relatará os componentes que foram instalados na residência, a Seção 3.3 mostra a implementação do sistema e a Seção 3.4 apresenta o desenvolvimento do aplicativo.

3.1 – Arquitetura da Aplicação

A arquitetura do sistema ilustrada na Figura 3.1 representa a conexão dos sensores com o Arduino, o envio dos dados dos sensores para o aplicativo através do Arduino conectado ao EthernetShield e a transmissão destes dados para o aplicativo SafeHome.

Figura 3 – Arquitetura do Sistema



Fonte: AUTOR(2016)

3.2 – Materiais Usados

Esta seção descreve os materiais utilizados no desenvolvimento do sistema para uma casa inteligente.

3.2.1 – Arduino Uno R3

A Figura 3.1 é o microcontrolador, Arduino Uno, que é responsável pela conversão de dados dos sensores.

Figura 3.1 – Arduino Uno



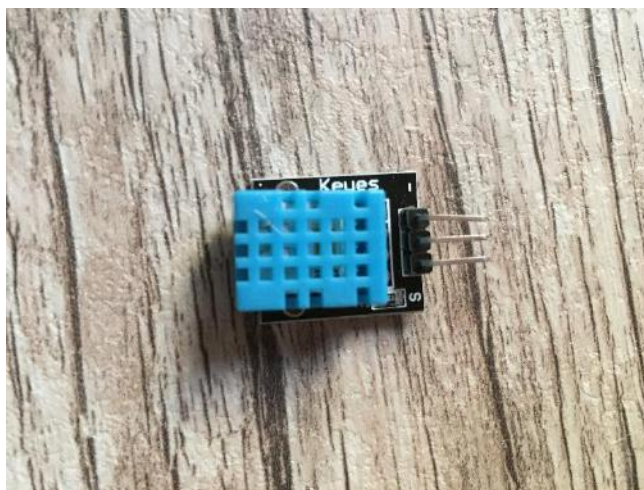
Fonte: AUTOR(2016)

Todos os sensores foram conectados no Arduino que transmitiu os dados através do Ethernet Shield.

3.2.2. – Sensor de Umidade

A Figura 3.3 mostra o sensor de umidade que é responsável por monitorar dados sobre umidade na residência e transmiti-los ao Arduino.

Figura 3.3 – Sensor de Umidade



Fonte: AUTOR(2016)

3.2.3 – Sensor de Movimento

A Figura 3.4 mostra o sensor de movimento que é responsável por enviar uma notificação ao dispositivo móvel do usuário caso detecte alguma presença no cômodo da residência.

Figura 3.4 – Sensor de Movimento



Fonte: AUTOR(2016)

3.2.4 – Sensor de Temperatura

A Figura 3.5 mostra o sensor de temperatura que é responsável por monitorar a temperatura do ambiente e enviar um alerta caso ocorra mudanças muito drásticas.

Figura 3.5 – Sensor de Temperatura



Fonte: AUTOR(2016)

3.2.5 – Ethernet Shield

A Figura 3.6 representa o Ethernet Shield que pode ser conectado ao Arduino de forma que ele se comunique com o aplicativo pela rede.

Figura 3.6 – Ethernet Shield



Fonte: AUTOR(2016)

3.3 – Preparo do Ambiente de Desenvolvimento

Para que o Arduino se comunique com o computador, é necessário que o conecte a ele pelo cabo USB. A conexão com o computador é apenas necessária na hora de fazer o upload do código que foi escrito nele para a placa.

O Ethernet Shield deve ser encaixado no Arduino para que ocorra a conexão entre eles conforme mostrado na Figura 3.7.

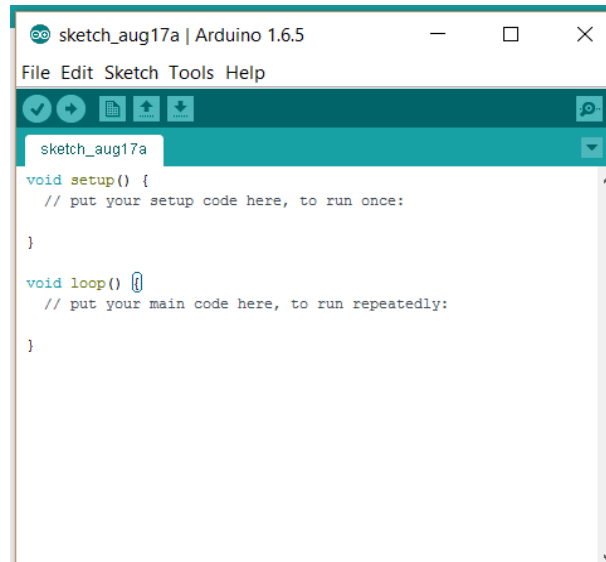
Figura 3.7 – Arduino e Ethernet Shield



Fonte: Vetco (2016).

A IDE usada para fazer upload do código para o Arduino pode ser encontrada no site oficial da placa. A primeira tela que é apresentada para o desenvolvedor é mostrada na Figura 3.8.

Figura 3.8 – Tela inicial da IDE do Arduino



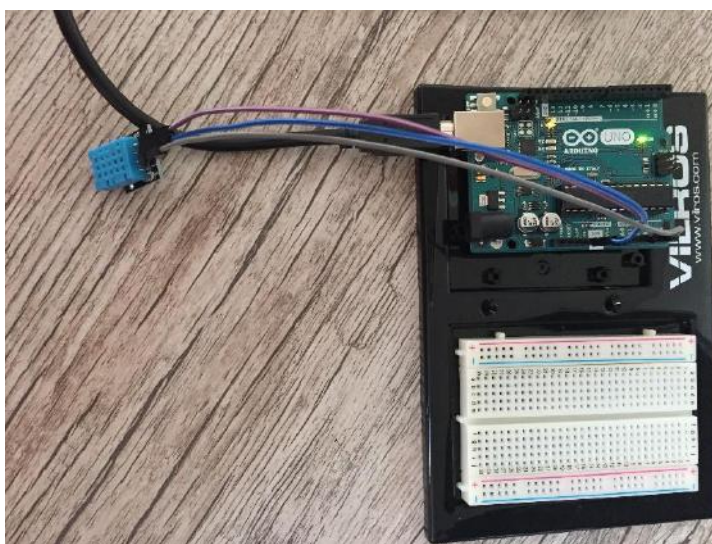
Fonte: AUTOR(2016)

Para o desenvolvimento do aplicativo mobile, foi utilizado o framework Kivy em uma máquina com o Sistema Operacional Ubuntu instalado.

3.4 – Comunicação entre o Arduino e o Sensor de Umidade

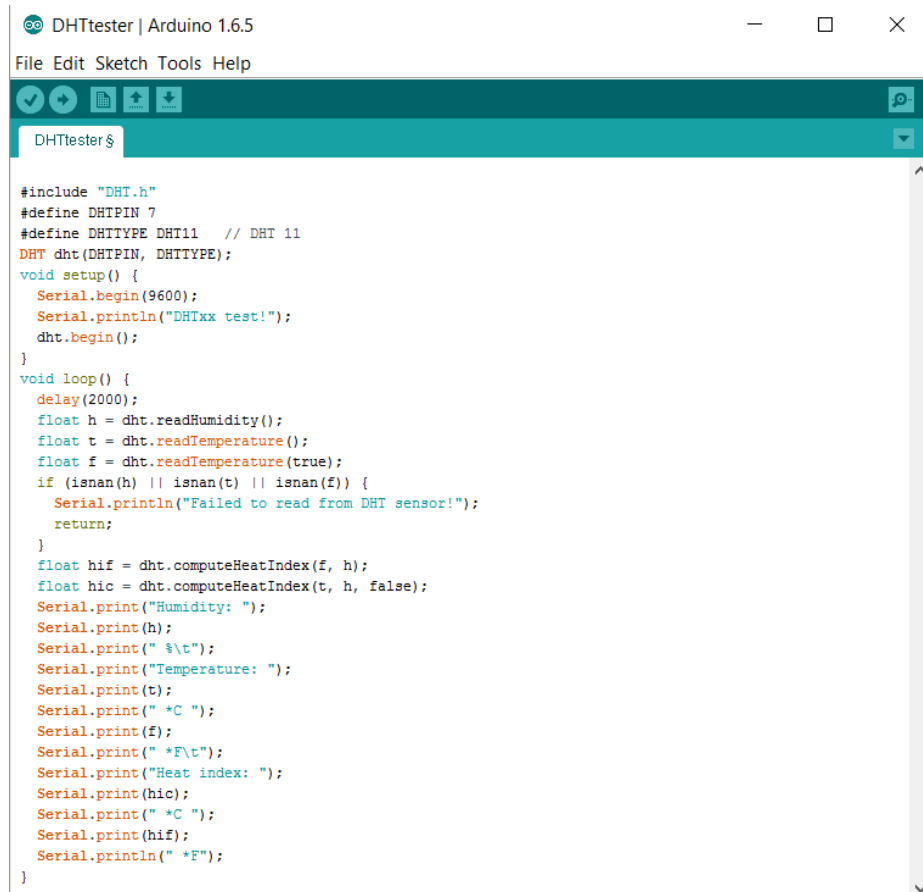
O sensor de umidade DHT11 monitora os níveis de umidade no local em que ele está instalado e os mostra na forma de porcentagem (0 – 100%). A função dele neste trabalho é enviar os dados para o Arduino que posteriormente os comunicará para o aplicativo mobile. Caso haja mudanças súbitas nos níveis de umidade, o sensor enviará uma notificação ao celular do usuário para alertar sobre vazamentos.

Figura 3.9 – Sensor de Umidade conectado ao Arduino



Fonte: AUTOR(2016)

Código 3.1 – Monitoramento de ambiente

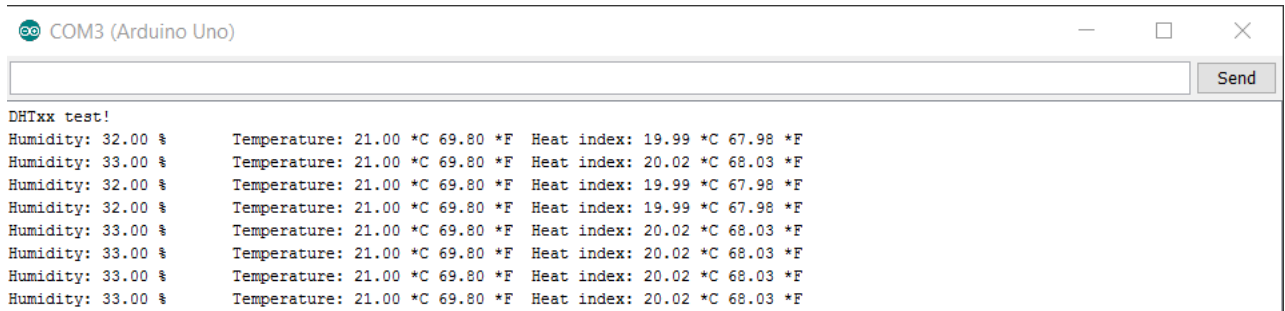
The image is a screenshot of the Arduino IDE interface. The title bar at the top reads "DHTTester | Arduino 1.6.5". Below the title bar is a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Underneath the menu bar is a toolbar with icons for saving, opening, and other standard IDE functions. The main area of the IDE is a text editor displaying a C++ sketch named "DHTTester". The sketch includes the DHT library, defines the pin and sensor type, and contains setup and loop functions for reading and printing environmental data.

```
#include "DHT.h"
#define DHTPIN 7
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");
  dht.begin();
}
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\n");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(f);
  Serial.print(" *F\n");
  Serial.print("Heat index: ");
  Serial.print(hic);
  Serial.print(" *C ");
  Serial.print(hif);
  Serial.println(" *F");
}
```

Fonte: AUTOR(2016)

Este trecho do código mostra a importação da biblioteca apropriada para o sensor de umidade DHT e em seguida é informado o pino do arduino no qual o sensor está conectado. Após isso, iniciamos a leitura dos dados com funções da própria biblioteca. É interessante notar que o DHT11 também pode ser usado para monitorar a temperatura do local.

Tela 3.1 – Saída de dados do sensor de umidade



```

COM3 (Arduino Uno)
DHTxx test!
Humidity: 32.00 %      Temperature: 21.00 *C 69.80 *F  Heat index: 19.99 *C 67.98 *F
Humidity: 33.00 %      Temperature: 21.00 *C 69.80 *F  Heat index: 20.02 *C 68.03 *F
Humidity: 32.00 %      Temperature: 21.00 *C 69.80 *F  Heat index: 19.99 *C 67.98 *F
Humidity: 32.00 %      Temperature: 21.00 *C 69.80 *F  Heat index: 19.99 *C 67.98 *F
Humidity: 33.00 %      Temperature: 21.00 *C 69.80 *F  Heat index: 20.02 *C 68.03 *F
Humidity: 33.00 %      Temperature: 21.00 *C 69.80 *F  Heat index: 20.02 *C 68.03 *F
Humidity: 33.00 %      Temperature: 21.00 *C 69.80 *F  Heat index: 20.02 *C 68.03 *F
Humidity: 33.00 %      Temperature: 21.00 *C 69.80 *F  Heat index: 20.02 *C 68.03 *F

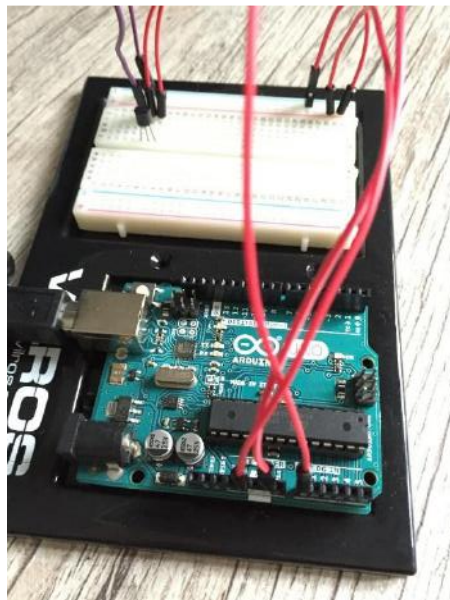
```

Fonte: AUTOR(2016)

3.5 – Comunicação entre o Sensor de Temperatura e o Arduino

O sensor de temperatura envia os dados de seu monitoramento para o arduino. Podemos exibir os dados tanto em Celsius quanto em Farenheit sendo esta apenas uma questão de conversão.

Figura 3.10 – Sensor de Temperatura conectado ao Arduino



Fonte: AUTOR(2016)

Código 3.2 – Monitoramento de Temperatura

```
void loop()
{
  float voltage, degreesC, degreesF;
  voltage = getVoltage(temperaturePin);
  degreesC = (voltage - 0.5) * 100.0;
  degreesF = degreesC * (9.0/5.0) + 32.0;
  Serial.print("voltage: ");
  Serial.print(voltage);
  Serial.print(" deg C: ");
  Serial.print(degreesC);
  Serial.print(" deg F: ");
  Serial.println(degreesF);
  delay(1000);
}

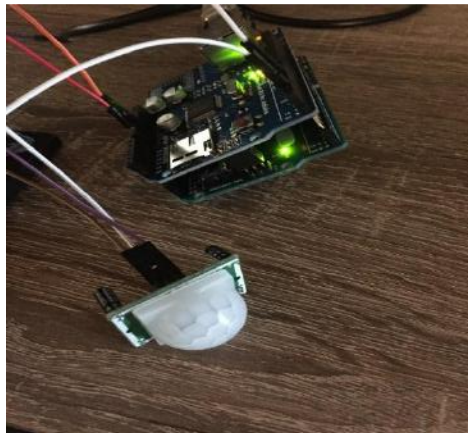
float getVoltage(int pin)
{
  return (analogRead(pin) * 0.004882814);
}
```

Fonte: AUTOR(2016)

3.6 – Comunicação entre o Sensor de Movimento e o Arduino

O sensor de movimento envia um sinal caso seja detectada presença humana na residência através de mudanças sutis no ambiente. Em casos positivos, também enviará uma notificação, como nos sensores anteriores.

Figura 3.11 – Sensor de Movimento Conectado ao Arduino



Fonte: AUTOR(2016)

O Código 3.3 apresenta a leitura dos dados do sensor de movimento, acendendo o LED e mostrando na tela da IDE quando um movimento é detectado e quando ele termina.

Código 3.3 – Monitoramento de Presença

```
int ledPin = 13; // escolhe o pin do led
int inputPin = 8; // input pin (sensor de movimento)
int pirState = LOW; // começa sem detectar movimento
int val = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(inputPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  val = digitalRead(inputPin); // ler o valor de input
  if (val == HIGH) { // checa se input esta alto
    digitalWrite(ledPin, HIGH); // liga o led
    if (pirState == LOW) {
      Serial.println("Movimento Detectado!");
      pirState = HIGH;
    }
  } else {
    digitalWrite(ledPin, LOW); // desliga o led
    if (pirState == HIGH) {
      Serial.println("Motion ended!");
      pirState = LOW;
    }
  }
}
```

Fonte: AUTOR(2016)

3.7 – Arduino e Ethernet Shield

Ao encaixar o Ethernet Shield na parte superior do Arduino, torna-se possível o envio de dados para um servidor web, fazendo com que o aplicativo possa acessá-lo posteriormente.

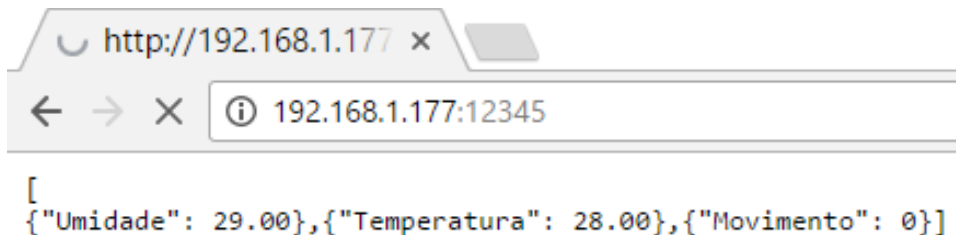
Código 3.4 – Conexão e Transmissão de dados utilizando o Ethernet Shield

```
#include <Ethernet.h>
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 1, 177);
EthernetServer server(12345);
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}
... ..
```

Fonte: AUTOR(2016)

O Código 3.4 mostra a atribuição de um endereço Mac e de um IP para o Arduino, além de uma porta para o servidor. Logo é aberta uma conexão na qual clientes podem acessar o microcontrolador pelo navegador e recebem os dados do microcontrolador.

Figura 3.12 – Dados enviados em formato JSON para o navegador



Fonte: AUTOR(2016)

A Figura 3.12 mostra como os dados são recebidos pelo aplicativo. Eles são enviados no formato JSON por ser fácil de decodificá-lo em um aplicativo Python.

3.8 – Desenvolvimento do Aplicativo

O desenvolvimento do aplicativo foi feito em um framework Python chamado Kivy (KIVY, 2016) que permite que o mesmo código construído rode em diversas plataformas como Android, iOS, Linux, OS X e Windows. Este framework possui *widgets* que são compatíveis com *multi-touch* e que são facilmente customizáveis.

Tendo isso em mente, o Safe Home 1.0 foi desenvolvido e testado com ênfase para celulares com o sistema operacional Android já que é o sistema operacional mais usado no Brasil.

As subseções abaixo mostrarão trechos de código das principais funções do Safe Home 1.0.

3.8.1 – Recebimento de dados do Aplicativo

O código 3.5 mostra a requisição de dados realizada ao endereço do servidor e a formatação da resposta recebida.

Código 3.5 – Requisição de Dados (Sensor de Temperatura)

```
def sensoresTemperatura(self,i):
    url = "http://192.168.1.177:12345"
    response = urllib.urlopen(url)
    data1 = json.loads(response.read())
    global maior_Temperatura
    global menor_Temperatura
    if float(data1[1]["Temperatura"]) > maior_Temperatura:
        maior_Temperatura = float(data1[1]["Temperatura"])
    if float(data1[1]["Temperatura"]) < menor_Temperatura:
        menor_Temperatura = float(data1[1]["Temperatura"])
    diferenca = float(data1[1]["Temperatura"]) - float(self.oldData1)
    if (diferenca >= 5 or diferenca <= -5) and self.count > 0:
        notification.notify("Alerta de Mudanca", "A temperatura da sua casa mudou muito em pouco tempo")
        self.sound.play()
    self.oldData1 = str(data1[1]["Temperatura"])
    if self.oldData1 != self.data:
        self.data = str(data1[1]["Temperatura"])
    if self.count == 0:
        self.count += 1
```

Fonte: AUTOR(2016)

Essa solicitação é programada para ser realizada a cada 15 segundos por meio de *threads*. *Thread* é uma forma de um processo de dividir em tarefas que podem ser executadas concorrentemente. Caso a temperatura aumente ou diminua significativamente em um curto período de tempo, uma notificação é enviada para o dispositivo do usuário, alertando-o desta mudança. O processo é similar com o sensor de umidade.

Código 3.6 – Requisição de Dados (Sensor de Movimento)

```
def sensorMovimento(self,i):
    url = "http://192.168.1.177:12345"
    response = urllib.urlopen(url)
    data1 = json.loads(response.read())
    if int(data1[2]["Movimento"]) == 1:
        notification.notify("Alerta de Movimento", "Movimento detectado na sua Residencia")
        self.sound.play()
        self.movimento = "Um movimento foi detectado na sua residencia as" + datetime.datetime.now().strftime("%H:%M:%S")
```

Fonte: AUTOR(2016)

O Código 3.6 também mostra uma solicitação que é concorrente, sendo realizada a cada 5 segundos no momento de execução do aplicativo. Quando é detectado um movimento, também uma notificação é enviada para o dispositivo do usuário e o horário que aconteceu este movimento é registrado.

3.8.2 – Protótipo do Relatório Geral

Esta tela registra os dados recebidos pelo aplicativo no momento de sua execução e os classifica para que o usuário possa ter acesso a informações como maior e menor temperatura e umidades registradas. O código desta tela encontra-se abaixo.

Código 3.7 – Tela de Relatório Geral

```
class ReportScreen(Screen):
    maiorTemp = StringProperty()
    menorTemp = StringProperty()
    maiorUmid = StringProperty()
    menorUmid = StringProperty()
    def settings(self,i):
        global maior_Temperatura
        global menor_Temperatura
        global maior_Umidade
        global menor_Umidade
        self.maiorTemp = str(maior_Temperatura)
        self.menorTemp = str(menor_Temperatura)
        self.maiorUmid = str(maior_Umidade)
        self.menorUmid = str(menor_Umidade)
    def __init__(self, **kwargs):
        super(ReportScreen,self).__init__(**kwargs)
        Clock.schedule_once(self.settings)
        Clock.schedule_interval(self.settings,8)
```

Fonte: AUTOR(2016)

3.8.3 – Elaboração do Layout do Aplicativo

O visual do aplicativo foi elaborado na linguagem kivy, uma linguagem própria da biblioteca Kivy para tornar seu código menos verboso e difícil de manter. Esta linguagem permite a criação de árvores widget, uma maneira fácil de deixar o código mais compreensível e organizado, separando a parte lógica do aplicativo e a parte visual (KIVY, 2016).

O código 3.8 exemplifica a construção de uma das telas do aplicativo, no caso, uma parte da tela principal. O aplicativo teve um total de 5 telas.

Código 3.8 – Layout (Árvore Widget)

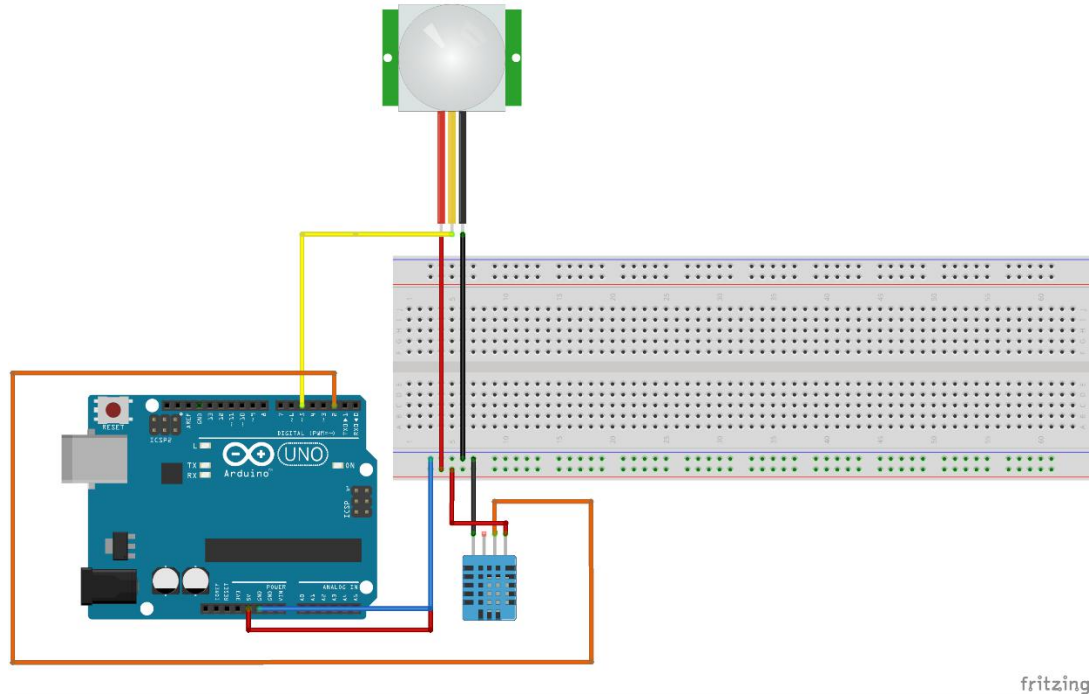
```
<HomeScreen>:
    name: 'home'
    GridLayout:
        rows: 2
        cols: 2
        Button:
            id: temp_button
            on_press: root.manager.current = 'temperatura'
            Image:
                source: 'temperatura.png'
                center_x: self.parent.center_x
                center_y: self.parent.center_y
            Label:
                text: 'Temperatura'
                center_x: self.parent.center_x
                center_y: self.parent.center_y - 50
        Button:
            id: impacto_button
            on_press: root.manager.current = 'movimento'
            Image:
                source: 'movimento.png'
                center_x: self.parent.center_x
                center_y: self.parent.center_y
            Label:
                text: 'Movimento'
                center_x: self.parent.center_x
                center_y: self.parent.center_y - 50
        Button:
            id:umidade_button
            on_press: root.manager.current = 'umidade'
            Image:
                source: 'umidade.png'
                center_x: self.parent.center_x
                center_y: self.parent.center_y
            Label:
                text: 'Umidade'
                center_x: self.parent.center_x
                center_y: self.parent.center_y - 50
        Button:
            id:relatorio_button
            on_press: root.manager.current = 'relatorio'
```

Fonte: AUTOR (2016).

3.9 – Circuito Final

A Figura 3.13 mostra a esquemática de como ficou o circuito do projeto completamente montado. O Arduino foi alimentado pela USB do computador e foram conectados os pinos 5v e grd na parte positiva e negativa da placa de ensaio respectivamente.

Figura 3.13 – Esquemática Final



Fonte: AUTOR(2016)

O sensor de umidade foi alimentado pela placa de ensaio e ligado ao pino digital número 2. Já o sensor de movimento foi conectado ao pino digital 5 e também alimentado pela placa de ensaio.

4. – RESULTADOS

Este capítulo apresenta os resultados obtidos com o processo de desenvolvimento feito no capítulo 3 baseado no conceito de *Internet of Things (IoT)* para o monitoramento residencial via rede por meio de um aplicativo instalado em dispositivos móveis.

4.1 – Instalação do Buildozer

Com o objetivo de compilar o aplicativo para aparelhos Android, foi usada a biblioteca *buildozer* que com poucas configurações, torna-o compatível com a maioria de aparelhos usando esse sistema operacional. Esta biblioteca é instalada com o comando mostrado na Figura 4.1.

Figura 4.1 – Instalação da biblioteca Buildozer

```
raquel@raquel-VirtualBox:~/Downloads/prototipoTG$ sudo pip install buildozer
```

Fonte: AUTOR (2016).

Quando o aplicativo estiver pronto para ser compilado e testado no celular, podemos dar um outro comando, como ilustrado na Figura 4.2.

Figura 4.2 – Comando init

```
raquel@raquel-VirtualBox:~/prototipoTG$ buildozer init  
File buildozer.spec created, ready to customize!
```

Fonte: AUTOR(2016).

Este comando gera um arquivo chamado *buildozer.spec*. Nele estão contidos algumas especificações importantes na hora da execução do aplicativo que serão abordadas abaixo.

4.1.2 – Itens no arquivo de especificação

O arquivo de configuração é essencial pois ele especifica diversos itens de configuração e permissões requeridas para o aplicativo ser executado em um determinado aparelho. A Figura 4.3 mostra os itens presentes no arquivo.

Figura 4.3 – Arquivo de Configuração

```
[app]
title = SmartHome
package.name = mysmarthome
package.domain = org.test
source.dir = .
source.include_exts = py,png,jpg,kv,atlas,wav
version = 0.1
requirements = plyer,android,kivy
orientation = all
fullscreen = 1
android.permissions = INTERNET
[buildozer]
log_level = 2
warn_on_root = 1
```

Fonte: AUTOR(2016).

Os itens presentes no arquivo são respectivamente: o título do aplicativo, o nome do pacote do aplicativo e seu domínio, o diretório em que ele se encontra e as extensões que devem conter no pacote que será enviado para o celular, a versão do aplicativo, as bibliotecas requeridas por ele, a orientação da tela no qual o aplicativo funcionará, se ele será executado em tela cheia, as permissões que deverão ser concedidas pelo celular ao aplicativo. Os itens restantes são importantes para o desenvolvimento pois especificam o nível de detalhes do relatório gerado no modo de testes do programa.

4.2 – Compilação e funcionamento do aplicativo no celular Android

Após o seu desenvolvimento, para finalmente testar o aplicativo rodando num aparelho celular, o dispositivo deve ser conectado ao computador e seu modo de depuração deve estar ativo para que ele seja identificado. Em seguida, no diretório do aplicativo, o comando ilustrado na Figura 4.4 é executado.

Figura 4.4 – Enviando o app para o celular

```
raquel@raquel-VirtualBox:~/Downloads/prototipoTG$ buildozer android deploy run debug
```

Fonte: AUTOR(2016)

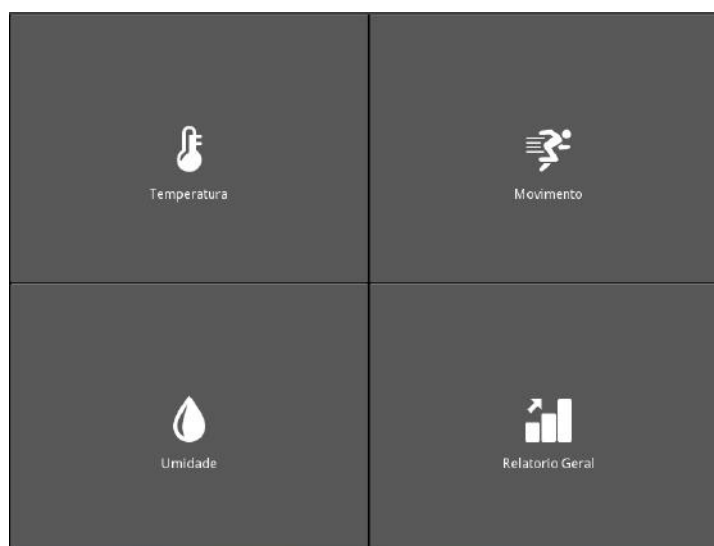
O parâmetro *android* especifica o sistema operacional, o *deploy* faz o pacote que será enviado para o armazenamento do aparelho, o *run* executa a aplicação imediatamente após o envio e o *debug* gera relatórios para identificação de possíveis erros ou bugs que possam ser encontrados no momento de sua execução.

4.3 – Telas do Aplicativo

4.3.1 – Tela Principal

Ao abrir o aplicativo, o usuário se deparará com a tela principal. Ela contém quatro botões que levam respectivamente para quatro outras telas que são: tela de umidade, tela de temperatura, tela de movimento e tela de relatório. A Figura 4.5 mostra o layout desta tela.

Figura 4.5 – Layout da Tela Principal



Fonte: AUTOR(2016).

4.3.2 – Tela de Temperatura

A tela de temperatura mostra os dados recebidos do sensor de temperatura e é atualizada uma vez por minuto. A Figura 4.6 mostra o layout desta tela.

Figura 4.6 – Layout da Tela de Temperatura

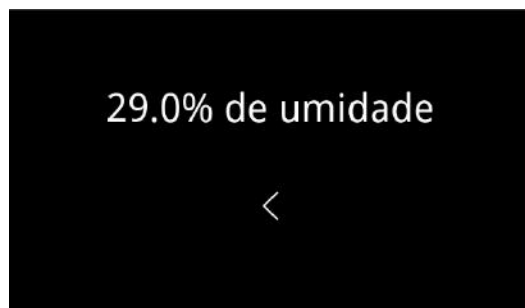


Fonte: AUTOR (2016).

4.3.2 – Tela de Umidade

A tela de umidade mostra os dados recebidos pelo aplicativo do sensor de umidade. Esta tela é atualizada a cada 45 segundos.

Figura 4.7 – Layout da Tela de Umidade

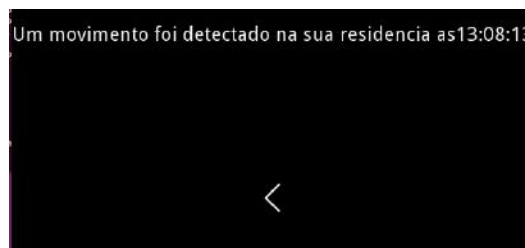


Fonte: AUTOR(2016).

4.3.3 – Tela de Movimento

A tela de movimento mostra o horário em que houve a última notificação em que o sensor detectou algum movimento na residência. A Figura 4.8 mostra o layout desta tela.

Figura 4.8 – Layout da Tela de Movimento

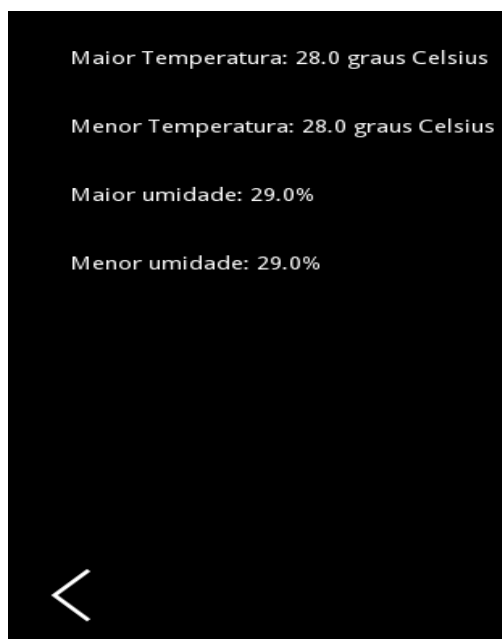


Fonte: AUTOR(2016).

4.3.4 – Tela de Relatório Geral

A tela de relatório geral mostra a temperatura e umidade (máxima e mínima) que foram detectadas na residência durante toda a execução do aplicativo. A Figura 4.9 mostra o layout desta tela. Note que o aplicativo estava sendo executado a pouco tempo, portanto estes valores serão iguais.

Figura 4.9 – Layout da Tela do Relatório Geral



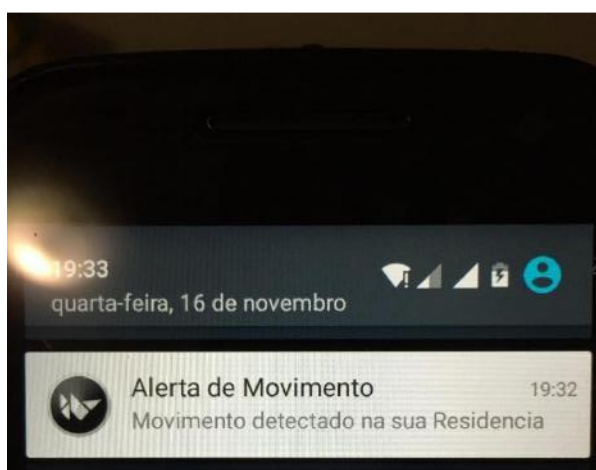
Fonte: AUTOR(2016).

4.4 – Sistema de notificação

4.4.1 – Funcionamento

O sistema compara os dados enviados anteriormente ao aplicativo com os mais recentes e caso haja uma discrepância muito grande entre eles, tanto para mais como para menos, uma notificação é enviada para o celular e é reproduzido um som de alerta. No caso do sensor de movimento, a notificação é enviada assim que é detectado um movimento. A notificação tem a aparência da Figura 4.10.

Figura 4.10 – Notificação exibida na tela do Celular



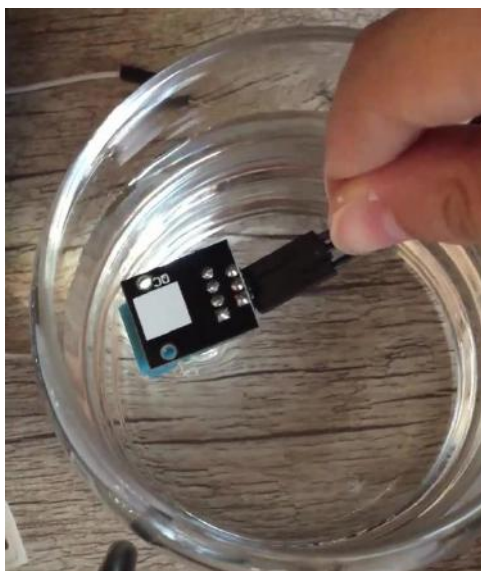
Fonte: AUTOR(2016)

4.5 – Testes

4.5.1 – Sensor de Umidade

Para efetuar o teste com o sensor de umidade detectando mudanças súbitas no ambiente causando então uma notificação no dispositivo móvel pelo aplicativo de monitoramento, foi simulado um ambiente de mudanças no qual ocorria um vazamento. O vazamento foi simulado ao submergir parte do sensor DHT11 em um copo de água como mostrado na Figura 4.11.

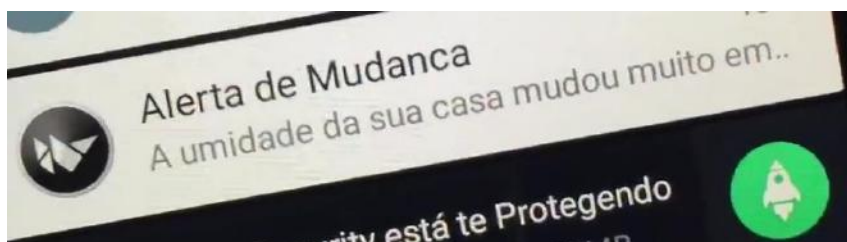
Figura 4.11 – Teste de Aumento de Umidade



Fonte: AUTOR(2016).

Como esperado, uma notificação foi enviada ao aparelho, alguns segundos depois de detectada a mudança, ilustrada n Figura 4.12.

Figura 4.12 – Notificação de Mudança de Umidade

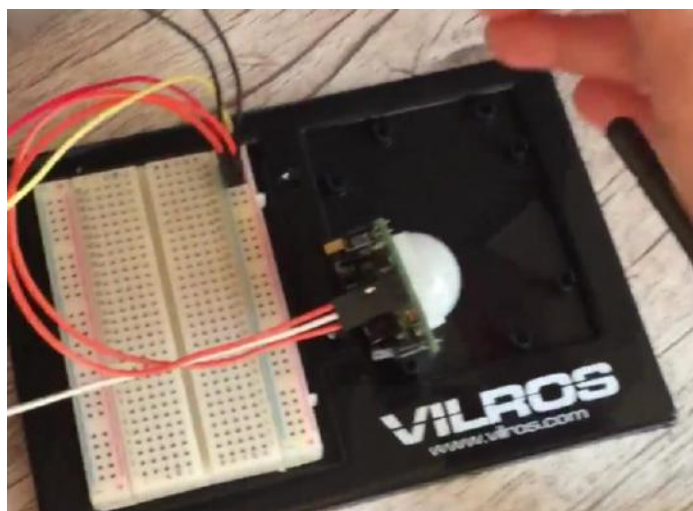


Fonte: AUTOR(2016).

4.5.2 – Sensor de Movimento

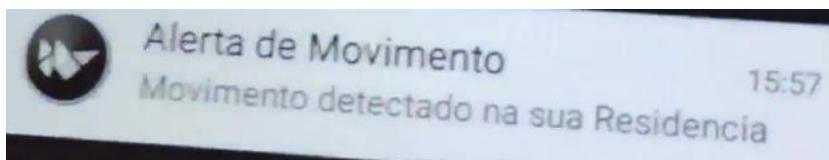
O teste com o sensor de movimento foi realizado ao mover-se um determinado objeto no campo de detecção do sensor exibido na Figura 4.13.

Figura 4.13 – Teste com o Sensor de Movimento



Fonte: AUTOR (2016).

A notificação foi acionada segundos depois, assim como mostra a Figura 4.14.



Fonte: AUTOR (2016).

5 – TRABALHOS FUTUROS

O objetivo deste capítulo é apresentar possíveis trabalhos futuros que poderão ser realizados dando continuidade a este TG, providenciando melhorias e novas funções.

As seguintes funções poderiam ser acrescentadas a este trabalho:

- a) Gerar relatórios mais detalhados;
- b) Armazenar em um banco os dados dos sensores para gerar relatórios de períodos maiores;
- c) Geração de gráficos com dados dos sensores;
- d) Instalação de ainda mais sensores na residência e integração deles no aplicativo;
- e) Um design mais bonito e intuitivo do aplicativo.

Referências

Almeida H. Tudo Conectado. Computação Brasil, Porto Alegre, p. 6-9, 2015.

Riley M. Programming Your Home with Arduino, Android, and Your Computer. Dallas: The Pragmatic Bookshelf, 2012. 229 p.

Kivy Cross-Platform Framework for NUI Development. Disponível em: <https://kivy.org/#home>. Acessado em 05/06/2016.

World Wide Web. U. Porto: PEUS, 2006. 19 slides, color. Acompanha texto.

Controlador Arduino. Disponível em: <https://www.arduino.cc/>. Acessado em 14/04/2016.

DHT11 Library. Disponível em: <https://github.com/adafruit/DHT-sensor-library>. Acessado em 20/07/2016.

Componentes Sunfounder. Disponível em: <https://www.sunfounder.com/>. Acessado em 20/07/2016.

How to Install and Use ADB, the Android Debug Bridge Utility. Disponível em: <http://www.howtogeek.com/125769/how-to-install-and-use-abd-the-android-debug-bridge-utility/>. Acessado em 18/07/2016.

Acendendo uma Lâmpada com o Sensor de Presença. Disponível em: <http://blog.filipeflop.com/sensores/acendendo-uma-lampada-com-sensor-de-presenca.html>. Acessado em 02/10/2016.