Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

GABRIEL DE SOUZA TAVARES

AUTENTICAÇÃO E AUTORIZAÇÃO DE ACESSOS EM AMBIENTES INTELIGENTES COM TECNOLOGIA NEAR FIELD COMMUNICATION

Assis/SP

2017

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

GABRIEL DE SOUZA TAVARES

AUTENTICAÇÃO E AUTORIZAÇÃO DE ACESSOS EM AMBIENTES INTELIGENTES COM TECNOLOGIA NEAR FIELD COMMUNICATION

Projeto de pesquisa apresentado ao Curso de Ciência da Computação de do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando: Gabriel de Souza Tavares.
Orientador: Prof. MSc. Guilherme de Cleva Farto.

Assis/SP

2017

DEDICATÓRIA

Dedico este trabalho a minha família, principalmente aos meus pais por terem me dado todo amor, carinho, educação e serem exemplos de pessoas, pois sem eles este trabalho e muitos dos meus sonhos não se realizariam.

AGRADECIMENTOS

Primeiramente agradeço a **Deus**, pois Ele me permitiu chegar até o final, adquirindo conhecimento necessário para superar todas as dificuldades.

Agradeço ao meu pai **Márcio Tavares**, minha mãe **Tânia Maria de Souza Tavares** e irmão **Matheus Tavares** que mesmo na distância estão sempre me apoiando, orientando para o melhor.

Ao meu amigo e Professor MSc. **Guilherme de Cleva Farto**, excelente orientador, sempre atencioso e motivador. Obrigado por compartilhar seu conhecimento para concluir este projeto.

Aos amigos, pelo apoio, amizade e companheirismo. Em especial ao **Vinicius Nunes Maciel**.

Por fim agradeço a todos que contribuíram diretamente ou indiretamente para concluir esta etapa de minha vida, deixo aqui meu muito obrigado.

“As tecnologias mais profundas são aquelas que desaparecem. Tecem-se no tecido da vida cotidiana, até que são indistinguíveis a partir dele.”

Mark Weiser (1952 – 1999)

RESUMO

O mundo em que vivemos tornou o uso da tecnologia cada vez mais frequente na vida cotidiana das pessoas, fazendo com que inúmeros dispositivos estejam conectados com diferentes finalidades e funcionalidades. Neste contexto, a tecnologia NFC (Near Field Communication) tem grande importância ao se tratar de segurança, sua comunicação ocorre entre dispositivos a centímetros de distância, sendo considerada segura, pois mantem salva a integridade dos itens compartilhados entre eles. O presente trabalho tem como proposta construir uma camada extra de segurança ao fazer uso da tecnologia NFC, comunicação por radiofrequência, abordando conceitos, propriedades, vantagens e benefícios da tecnologia em plataformas IoT, com a finalidade de construir um protótipo para autenticar e autorizar acessos em ambientes inteligentes. A metodologia foi baseada em revisões bibliográficas, para assim ampliar o embasamento teórico sobre NFC e Plataformas de IoT, no qual auxiliam o desenvolvimento do experimento.

Palavras-chave: NFC; Ambientes Inteligentes; Autenticação; Autorização; IoT.

ABSTRACT

The world we live in has made the use of technology more and more frequent in people's daily lives, making numerous devices connected to different purposes and functionalities. In this context, NFC (Near Field Communication) technology has great importance when it comes to security, its communication occurs between devices within inches away, being considered safe as it keeps the integrity of the items shared between them. The present work aims to build an extra layer of security by using NFC technology, radiofrequency communication, addressing concepts, properties, advantages and benefits of technology in IoT platforms, in order to build a prototype to authenticate and authorize accesses in smart environments. The methodology was based on bibliographical revisions, in order to expand the theoretical foundation on NFC and IoT Platforms, in which they help the development of the experiment.

Keywords: NFC; Smart Environments; Authentication; Authorization; IoT.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 – INTRODUÇÃO

Com o avanço da tecnologia computacional o mundo se desenvolve cada vez mais, este deixa de ser analógico e se torna digital e conectado. Os dispositivos conectados estão presentes na vida cotidiana das pessoas, isto proporciona facilidades na comunicação, dessa forma, pessoas se conectam com diferentes finalidades sejam elas comerciais, educacionais ou sociais. Com o objetivo de incluir o indivíduo e torná-lo participante ativo na sociedade atual, este domínio de tecnologias de computação e comunicação torna-se um fator essencial (SOARES et al., 2008).

De acordo com o IBSG (*Cisco Internet Business Solutions Group*), em 1984, apenas 1000 dispositivos estavam conectados à Internet. Em 2003, o número de dispositivos conectados foi modificado para aproximadamente 500 milhões, e em 2010, passou a ser representado pela estatística de 12,5 bilhões. O Cisco IBSG prevê que haverá 50 bilhões de dispositivos conectados até 2020 (EXAME, 2014).

Consequentemente, os aparelhos utilizados no cotidiano estão por desaparecer, pois suas funcionalidades estão sendo atendidas por novos dispositivos conectados. Essa tendência do uso de apenas uma estrutura para prover um arsenal de serviços é chamada de Convergência Tecnológica (LIMA, 2013).

Neste cenário de evolução tecnológica, a IoT (*Internet of Things*) tem grande importância, pois dá um grande salto na capacidade de coletar, analisar e distribuir dados que podem ser transformados em informações e conhecimento (EVANS, 2011).

Na *Internet of Things* um conjunto de dispositivos físicos são chamados objetos inteligentes. Esses objetos conectam-se à Internet e, dessa forma, tornam-se capazes de recepcionar e enviar informações sem intervenção direta do homem na execução destas ações. Deste modo, essas tecnologias de informação e comunicação, estão presentes no contexto das cidades, mais especificamente das cidades inteligentes (WHITMORE et al., 2014).

Uma cidade inteligente é definida como um território no qual as tecnologias de informação e comunicação são utilizadas para capturar, analisar e integrar informações relevantes no núcleo de seus sistemas. Essas cidades propiciam inovações em diversos segmentos em benefício da população, instituições locais e órgãos governamentais (KOMNINOS, 2014). Assim, possui a competência de tomar decisões inteligentes para diferentes tipos de necessidades, o que inclui aspectos diários, proteção ambiental, segurança pública, serviços da cidade e atividades industriais e comerciais (HAUBENSAK, 2014).

Sob o mesmo ponto de vista, cidades instrumentalizadas, interconectadas e inteligentes também são definidas como *smart cities*. Essa instrumentalização possibilita a captura e integração de dados do mundo real por meio de sensores, medidores, aparelhos, câmeras, *smartphones*, dispositivos médicos implantados em pessoas, Internet e outros sistemas de aquisição de dados, o que também inclui redes sociais como redes de sensores humanos (HARRISON et al., 2010).

Sendo assim, a rede *wireless*, comunicação sem fio, é de extrema importância, pois apresenta, nos últimos anos, grandes inovações tecnológicas no setor de comunicações (ZEINDIN et al., 2003). Dessa forma, torna-se muito mais viável que a rede cabeada, pois possibilita a troca de dados e informações entre dispositivos, sem ser necessária uma mudança física, portanto não há problemas e limitações estruturais (MENESES, 2009).

Com relação à comunicação sem fio, o NFC (*Near Field Communication*) ganha destaque no cenário tecnológico. O NFC sem dúvidas possui grande impacto ao se tratar de serviços como autenticações e troca de dados, pois trata de uma comunicação a curto alcance entre dispositivos. Como a comunicação só ocorre entre dispositivos a centímetros de distância, tal comunicação é considerada uma transmissão segura, assim, mantêm salva a integridade dos itens compartilhados entre eles (SPINSANTE et al., 2015).

A NFC é uma tecnologia de comunicação de curto alcance, de alta frequência, baseada no princípio da identificação por radiofrequência (*Radio-Frequency Identification* - RFID). As aplicações NFC tem conquistado uma popularidade

crescente, já que muitos aparelhos possuem um transceptor NFC (SPINSANTE et al., 2015).

Assim, de fato, a NFC é destinada a ser de fácil uso, como a interação é realizada por meio de um simples toque, a NFC coloca um sentido de controle humano em sistemas complexos. O toque é um gesto natural e expressivo e pode ser usado para gerar interações satisfatórias (JARABA et al., 2011).

1.1. OBJETIVOS

1.1.1. Objetivos Gerais

O objetivo geral deste projeto é pesquisar, explorar e analisar conceitos da tecnologia NFC tal que, exponha seus principais conceitos, propriedades, vantagens e benefícios, com a finalidade de obter informações necessárias para realizar a sua inserção em serviços de autenticações.

Também se destaca, como objetivo geral, a concepção, modelagem e implementação de um protótipo de aplicação e cenário experimental para a validação dos conceitos de NFC.

1.1.2. Objetivos Específicos

Pretende-se com este trabalho, explorar e implementar a tecnologia NFC, em um cenário em que se deseja controlar acessos, deste modo tornando possível a elaboração e execução, tanto das etapas teóricas e praticas do projeto. Os seguintes objetivos estabelecidos foram:

- Obter conhecimento a respeito da tecnologia NFC.
- Pesquisar e desenvolver uma aplicação para controle de acesso, baseado na tecnologia NFC;
- Obter conhecimento das plataformas *Arduino* e *Raspberry PI* para integração de módulo NFC;
- Desenvolver um experimento.

1.2. JUSTIFICATIVAS

Tendo em vista o cenário de evolução tecnológica, a NFC é de extrema importância, pois oferece um alto nível de proteção contra diversos tipos de fraudes ao proporcionar um vínculo entre sistema e usuário, uma vez que somente o individuo presente poderá ser submetido à identificação. Dessa forma, não há preocupações quanto à falsa identificação realizada por terceiros, já que a identificação pelos dispositivos é em curta distância.

1.3. MOTIVAÇÃO

O interesse em desenvolver o tema deste projeto, consiste em adquirir aprendizados sobre o meio da comunicação NFC. Devido a sua praticidade e ao grande crescimento tecnológico, pois por meio de ações simples tem a capacidade de transformar sistemas complexos e onipresentes em sistemas dinâmicos, inteligente e personalizado o que cria um vínculo entre sistema e usuário, além da sua implantação possuir baixo custo de componentes e sensores eletrônico-digitais.

1.4. METODOLOGIA DE PESQUISA

Para o desenvolvimento deste trabalho, serão utilizadas diversas fontes para pesquisa bibliográfica, como por exemplo, artigos, teses, livros e outras informações retiradas da Internet, assim, tornando possível o desenvolvimento da pesquisa acadêmica e do experimento.

A metodologia para a execução deste projeto de conclusão de curso é experimental, visto que seu objetivo, além da fomentar a pesquisa acadêmica, é o desenvolver uma aplicação de protótipo para aplicar os conceitos NFC para ambientes inteligentes com as plataformas *Arduino* e *Raspberry PI.*

1.5. ESTRUTURA DO TRABALHO

Este trabalho será estruturado nas seguintes partes:

- **Capítulo 1 – Introdução**
- **Capítulo 2 – Tecnologia Near Field Communication (NFC)**
- **Capítulo 3 – Plataformas de IoT**
- **Capítulo 4 – Proposta do Trabalho**
- **Capítulo 5 – Desenvolvimento do Trabalho**
- **Capítulo 6 – Conclusões**
- **Referências**

2 – TECNOLOGIA NEAR FIELD COMMUNICATION (NFC)

O NFC é uma tecnologia *wireless,* tecnologia de contato sem fio por proximidade de curto alcance, e de alta frequência, baseada em padrões que tornam a vida mais cômoda e acessível para os seus consumidores. O que torna mais simples fazer transações, trocar conteúdo digital e conectar dispositivos eletrônicos com um toque. (NFCFORUM, 2017).

Esta tecnologia nasceu da cooperação entre a *Nokia Corporation*, a *Royal Philips Electronics* e a *Sony Corporation* em 2002, e promovido no NFC fórum em 2004, que é conduzido por empresas como *Samsung*, *Microsoft*, *Nokia*, *Google*, *Intel* e *Visa.* A finalidade deste é promover o avanço da tecnologia, definir especificações e regulamentações de uso, este é padronizado pela ISO 18092/ ECMA 340 (NFCIP-1) que especifica a interface e o protocolo de comunicação de fio entre dispositivos acoplados. (NFCFORUM, 2017).

O surgimento desta tecnologia ocorreu a partir do princípio da identificação por radiofrequência (*Radio-Frequency Identification* - RFID), por isso ambas possuem benefícios compartilhados. Há poucas diferenças entre as tecnologias, uma delas está relacionada aos possíveis modos de comunicação (NASSAR et all., 2014).

O RFID consiste em técnicas de identificação através de ondas de rádio, a comunicação tem apenas um único sentido, o dispositivo leitor envia os sinais para a etiqueta, esta apenas responde aos sinais. Já um dispositivo NFC pode gerar alternadamente as ondas de rádio com outro dispositivo, sendo possível atuar tanto como uma etiqueta como um leitor. Isto possibilita uma comunicação *peer-to-peer*. Outra diferença está relacionada ao alcance (SAB et all., 2013).

Sistemas RFID tradicionais podem apresentar um bom alcance, dezenas de metros, porém varia de acordo com o tipo de etiqueta. Já os dispositivos NFC apresentam um alcance pequeno, máximo de 10cm. Porém, esta limitação no alcance apresenta algumas vantagens, pois oferece maior segurança na comunicação, já que dificulta a ocorrência de ataques, já que necessita de uma proximidade muito pequena para a transferência de dados (SAB et all., 2013).

O desenvolvimento da tecnologia de radiofrequência fez com que as aplicações do NFC adquirissem popularidade crescente nos usos de novas tecnologias na atualidade, pois muitos aparelhos possuem um transceptor NFC (SPINSANTE et al., 2015). Neste contexto o NFC tem embarcado, principalmente, *smartphones* devido as suas múltiplas funcionalidades, como a conectividade a redes móveis, poder de processamento e por ser um dispositivo muito comercializado mundialmente.

Ao se comparar com o *Bluetooth* pode-se destacar algumas diferenças significativas envolvendo alcance, taxa de transmissão de dados, consumo de energia e o tempo estimado para iniciar uma conexão entre dispositivos, sendo que o *Bluetooth* necessita de alguns passos manuais para estabelecer a comunicação (SAB et all., 2013).

2.1. FUNCIONAMENTO DO NFC

O NFC é uma tecnologia criada para permitir a comunicação entre dois dispositivos. No protocolo NFCIP-1, o equipamento é identificado como *target* ou *intiator*, este inicia a comunicação e controla a troca de dados, ou pode responder às requisições do *initiator*, respectivamente. (ALECRIM, 2012).

O protocolo *NFCIP-1* apresenta duas formas de funcionamento: o modo de comunicação ativa, onde ambos dispositivos geram ondas eletromagnéticas para transportar dados, e o modo de comunicação passivo, onde um dos dispositivos gera as ondas enquanto o outro usa a modulação recebida para retransmitir dados. O NFC opera na frequência de 13,56 MHz, quanto à velocidade de comunicação, a taxa de transferência de dados pode ocorrer em 106, 212 ou 424 kbit / seg. Em modulação por modulação de amplitude *ASK*. A transmissão ocorre em torno de 10 centímetros de distância, o que proporciona uma segurança nas comunicações. (ECMA INTERNATIONAL, 2004).

Na figura 1 é ilustrada a comunicação entre um dispositivo e a antena NFC.

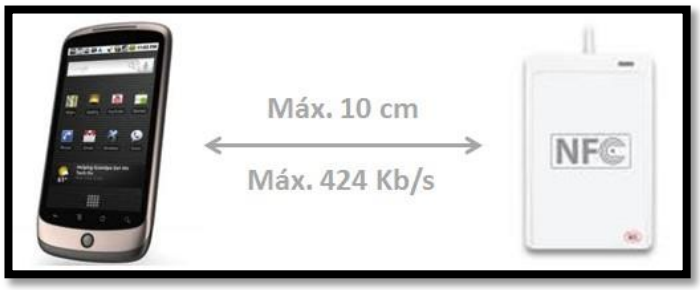


Figura 1: NFC transmissão em torno de 10 centímetros de distância.
Fonte: (INFOWESTER, 2017).

2.1.1. Modos de operação

O sistema NFC foi desenvolvido para funcionar de três modos de operação, que aumentam suas possibilidades de uso (CUNHA, 2016).

• *Card Emulation* – o dispositivo NFC se apresenta como um *smart card* ou um cartão de crédito *contactless*. Assim, pode-se usar um celular NFC para efetuar pagamentos de compras, bilhetes de transporte público, realizarem transações financeiras, acesso a portas e catracas. Existem diversos níveis de segurança que podem embutir neste modo, estes níveis de segurança implementados incluem recursos de hardware no chip NFC dentro do telefone celular e de software dos aplicativos que conversarão com este chip.

Na figura 2 é ilustrado o NFC atuando no modo *Card Emulation, com dispositivos.*

Figura 2: NFC atuando no modo *Card Emulation.*
Fonte: (EXELANZ, 2017).

• *Reader/Writer*– nesta configuração o dispositivo se comporta como um leitor de dados de *tags* NFC. Deste modo, podem-se ler as informações que contêm em uma *tag e* carregar seu conteúdo. Se estas *tags* contiverem informações como o preço de um produto, data de validade e site do fabricante, estas informações podem ser lidas a ponto de interagir com aplicativos de um dispositivo móvel.

Na figura 3 é ilustrado o NFC atuando no modo Reader/Writer.

Figura 3: NFC atuando no modo *Reader/Writer.*
Fonte: (NFC WORLD, 2017).

• *Peer-to-Peer* – quando o dispositivo NFC se comunica com outro dispositivo NFC, deste modo criando uma comunicação bidirecional que gera a troca de dados entre eles, ou seja, cada um pode tanto receber quanto enviar dados para o outro. Este modo é utilizado, por exemplo, para fazer seu móvel parear instantaneamente com uma caixa de som, um roteador de internet e pegar dados de manutenção da sua geladeira ou máquina de lavar roupas.
Neste modo temos dois dispositivos NFC ativos, e eles devem conversar entre si trocando informações.

Na figura 4 é ilustrado o NFC atuando no modo *Peer-to-Peer*.

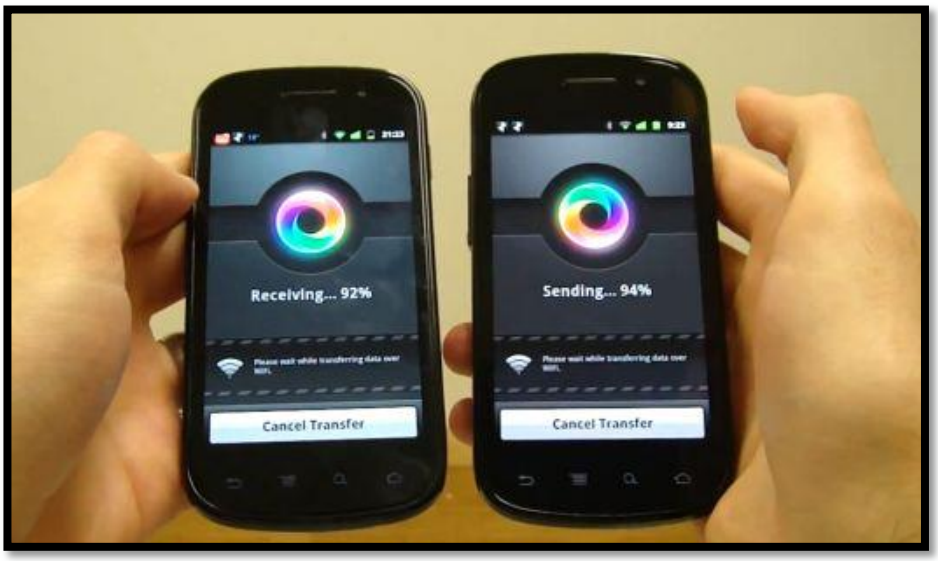


Figura 4: NFC atuando no modo *Peer-to-Peer.*

Fonte: (EMAZE, 2017).

2.1.2. Casos de uso

A tecnologia NFC pode ser utilizada em diversos tipos de aplicações (CRUZ, 2013).

- Transportes – cartões de embarque ou cartões de fidelização para créditos de pontos, onde os viajantes aproximam seus celulares ao leitor NFC no momento de embarque e *ticket.*
- Pagamentos – cartões de crédito para efetuar pagamentos de diversas compras em estabelecimentos comerciais;
- Serviços médicos – usado para armazenamento de informações médicas sobre seu portador, como auxilio em atendimentos de emergência, bem como o histórico médico, o que consta tratamentos realizados e os dados dos profissionais de saúde responsáveis; tipo sanguíneo e doenças.
- Capturas de informações de avisos e propagandas – é usado para ler informações contidas em *tags* afixados em propagandas, cartazes e em revistas, também conhecidos como *smart poster*, ou até mesmo disparar

um aplicativo para requisição de um serviço *on-line*, a partir da leitura de uma *tag* contida em um anúncio, por exemplo, um pedido de entregas.

2.2. TAGS

As *tags* também conhecidas como NFC *Stickers, SmartTag,* adesivos ou etiquetas NFC são finos dispositivos eletrônicos que se assemelham a adesivos, possuem uma antena e uma pequena memória, estes são alimentadas ou lidas através de indução magnética campo magnético. Estas podem ser incorporadas a qualquer tipo de produtos, como cartões de visitas, imãs de geladeira, pulseiras de identificação e chaveiros (CRUZ, 2013).

Na figura 5 é ilustrado *Tags NFC.*

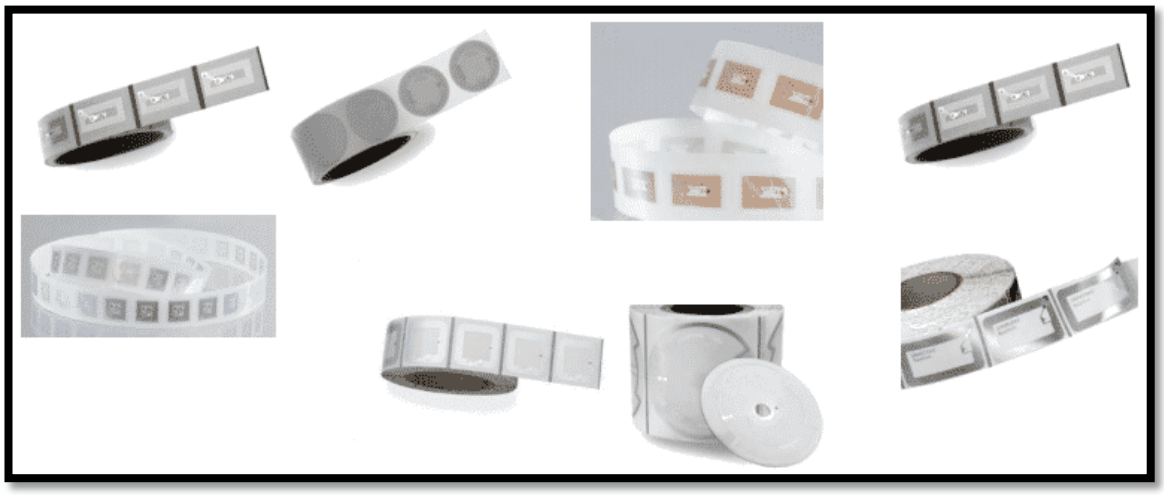


Figura 5: Diversos formatos e modelos de *Tag* NFC disponíveis no mercado.
Fonte: (EMBARCADOS, 2017).

A NFC Fórum especifica quatro tipos de plataformas diferentes de *tags* para NFC, cada uma apropriada para cada tipo de tarefa e depende de como será usada (CRUZ, 2013).

- **Tipo 1** – utiliza um modelo de memória simples, com uma capacidade física total de 120 *bytes*, mas apenas 96 *bytes* são disponíveis para

alocação de dados dos usuários, e é dividida em blocos de 8 *bytes* cada. Sua velocidade de comunicação é de 106 *Kb/*s.

- **Tipo 2** – este tipo é semelhante ao tipo 1, diferencia-se apenas em seu tamanho de memória total de 64 *bytes*, pois armazena entre 48 *bytes* e 2 KB para dados de usuários, além de ser dividida em blocos de 4 *bytes*.
- **Tipo 3** – baseada em uma tecnologia da *Sony* chamada *FeliCa*. A compatibilidade com outros padrões existe, porém não é garantida, não possui capacidade de memória definida, mas sua organização é disposta em blocos de 16 *bytes*, tem velocidade de 212 Kb/s.
- **Tipo 4** – sua capacidade de memória é variável de até 32 *Kbytes* e sua velocidade de comunicação chega a 424 Kb/s. Possui sistema de arquivo flexível com diferentes tipos de arquivos e acesso, também inclui verificação de integridade e opção de cifragem como algumas de suas características principais. É compatível com os tipos 1 e 2.

3 – PLATAFORMAS DE IOT

No contexto de Internet das Coisas, um conjunto de dispositivos físicos é chamado objetos inteligentes. Esses objetos conectam-se à Internet e, dessa forma, tornam-se capazes de recepcionar e enviar informações sem intervenção direta do homem na execução destas ações (WHITMORE et al., 2014).

Neste capítulo serão abordados características e funcionalidades das plataformas *Arduino* e *Raspberry Pi*, ambas possuem grande importância no cenário de IoT, pois são fundamentais para seu desenvolvimento. Será feita uma descrição dos assuntos presentes com a finalidade de fornecer uma base de conhecimento pela área estudada.

3.1. *ARDUINO*

O *Arduino* é uma plataforma de código aberto, tanto de *hardware* quanto de *software,* baseado numa placa microcontroladora, bem como em um ambiente de desenvolvimento para escrever o código para a placa. Criada em 2005 pelo italiano Massimo Banzi e outros colaboradores para auxiliar no ensino de eletrônica para estudantes. O objetivo principal foi o de criar uma plataforma de baixo custo, para que os estudantes pudessem desenvolver seus protótipos com o menor custo possível. Outro ponto interessante do projeto foi à possibilidade de uma plataforma de código aberto, disponível para a comunidade o que ajudou em muito no seu desenvolvimento (SOUZA, 2013).

O *Arduino* pode ser usado para controlar uma variedade de luzes, motores ou outras saídas físicas, desenvolver objetos interativos, o que admite entradas de uma série de sensores ou chaves. O microcontrolador na placa é programado com a linguagem de programação *Arduino*, baseada na linguagem *Wiring*, e o ambiente de desenvolvimento *Arduino*, baseado no ambiente *Processing*. Os projetos desenvolvidos podem ser autônomos ou podem comunicar-se com um computador para a realização da tarefa, com uso de software específico *Flash, Processing, MaxMSP*. Os circuitos podem ser montados à mão ou comprados

pré-montados; o software de programação de código-livre pode ser baixado de graça. (ARAÚJO et al, 2009).

3.1.1. Hardware do Arduino

Existem diversas placas oficiais de *Arduino*, que com o passar dos anos foram modificadas e evoluíram ao ponto de se tornarem mais acessíveis e sofisticados. A seguir serão abordadas as placas *Arduino Duemilanove* (2009), *Arduino* Mega, *Arduino* Nano e *Arduino Uno.*

***Arduino Duemilanove* (2009):**

- É uma placa baseada no microcontrolador ATmega168 ou ATmega328;
- Possui 14 pinos de entrada ou saída digital (dos quais seis podem ser utilizados como saídas PWM);
- 6 entradas analógicas, um oscilador de cristal 16 MHz, controlador USB, uma tomada de 21 alimentação, um conector ICSP, e um botão de reset;
- Para sua utilização basta conectá-lo a um computador com um cabo USB ou ligá-lo com um adaptador AC para DC ou bateria.

Na figura 6 é ilustrada a placa *Arduino Duemilanove (2009).*

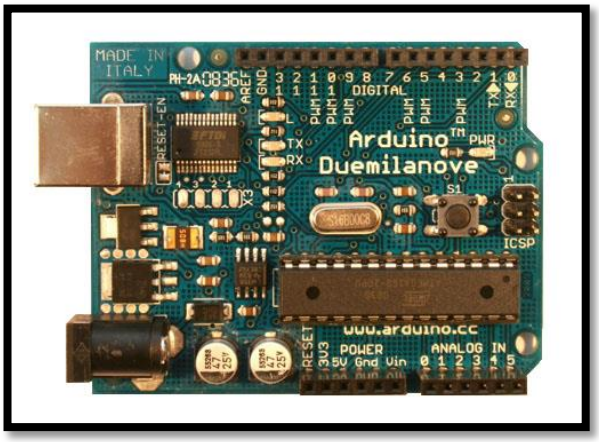


Figura 6*: Arduino Duemilanove* (2009).

Fonte: (ARDUINO, 2017).

***Arduino* mega 2560**:

- É uma placa de microcontrolador baseado no ATmega2560 (folha de dados). Ele possui 54 entradas / saídas digitais (dos quais 15 podem ser usados como saídas PWM);
- 16 entradas analógicas, 4 UARTs (portas seriais de hardware), a 16 MHz cristal oscilador, uma conexão USB, um conector de alimentação, um cabeçalho ICSP e um botão de reset;
- Ele contém tudo o que é necessário para suportar o microcontrolador, basta conectá-lo a um computador com um cabo USB ou ligá-lo a um adaptador AC para DC ou bateria para começar;
- O 22 mega é compatível com a maioria dos *Shields* projetados para o *Arduino Duemilanove* ou *Diecimila.* (ARDUINO, 2017);

Na figura 7 é ilustrada a placa *Arduino Mega 2560*

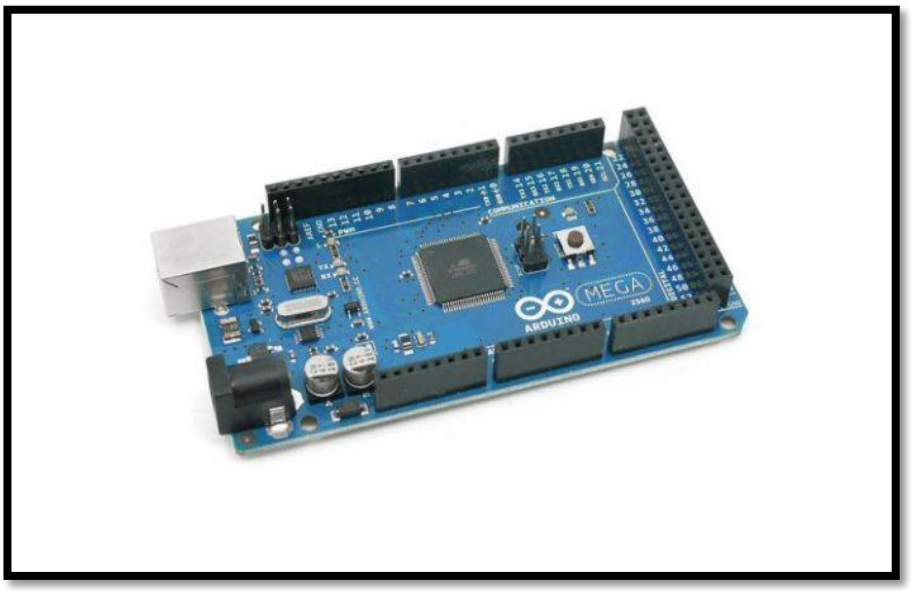


Figura 7: *Arduino Mega* 2560.

Fonte: (EMBARCADOS, 2017).

***Arduino* Nano:**

- É uma pequena versão de *Arduino* parecida com o *Arduino UNO*, pois também possui um chip ATMega328, na versão SMD;
- Possui um conector para cabos *Mini-USB* para gravação;
- Uma das diferenças entre esta placa e a Arduino UNO ou Arduino 2009, é que esta placa possui duas entradas analógicas a mais e um jumper de +5V AREF.;
- Não possui um conector para fonte externa, mas é possível alimentá-la pelo pino Vin. O Arduino Nano automaticamente seleciona a maior alimentação fornecida.

Na figura 8 é ilustrada a placa *Arduino Nano*

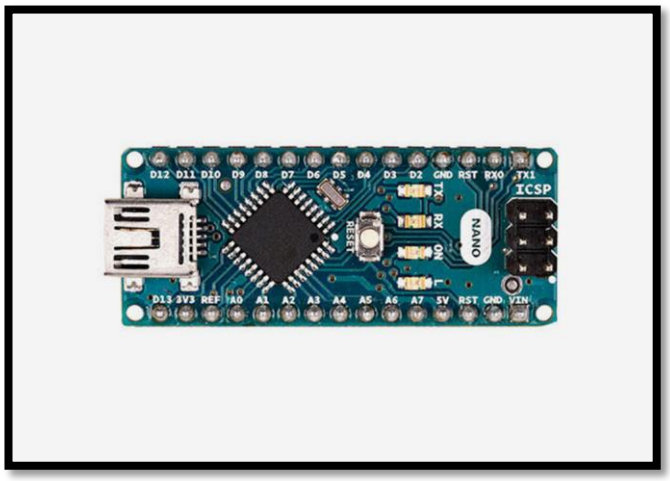


Figura 8*: Arduino Nano.*

Fonte: (ARDUINO, 2017).

***Arduino* Uno:**

- Uma das placas mais recentes no mercado, o *arduino uno* é uma placa de microcontrolador baseado no ATmega328 (datasheet);
- Possui 14 entradas / saídas digitais (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um cristal oscilador de

16MHz, uma conexão USB, um conector de alimentação, um cabeçalho ICSP, e um botão de reset;

- Ele contém tudo o que é necessário para suportar o microcontrolador, basta conectá-lo a um computador com um cabo USB ou ligá-lo com um adaptador AC para DC ou bateria para começar;
- A placa contém todo o necessário para usar o micro controlador. Simplesmente conecte-a a um computador com o cabo USB ou ligue a placa com uma fonte ACDC (ou bateria). O Uno seleciona automaticamente a fonte de alimentação (USB ou fonte externa). (RODRIGUES; SARTORI; GOUVEIA, 2012).

Na figura 9 é ilustrada a placa *Arduino Uno.*

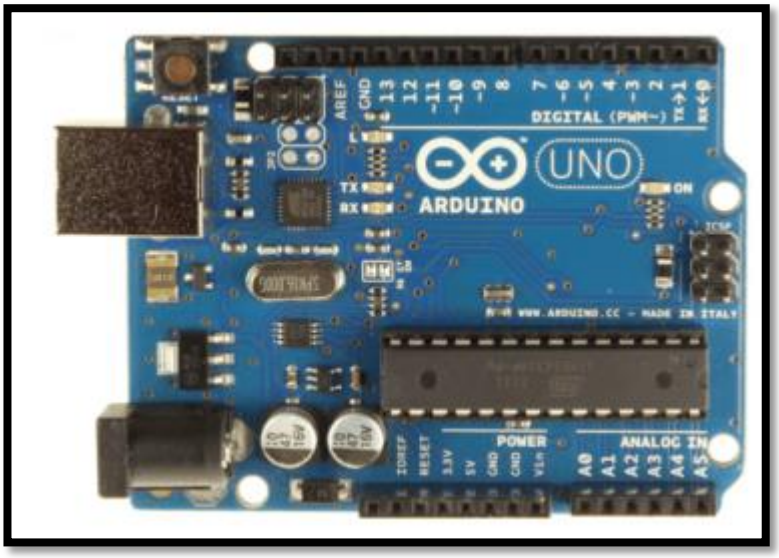


Figura 9: *Arduino Uno*.

Fonte: (EMBARCADOS, 2017).

Através da imagem acima podemos ver que a placa Arduino UNO possui diversos conectores que servem para interface com o mundo externo. Os pinos na placa estão organizados da seguinte maneira:

- 14 pinos de entrada e saída digital (pinos 0-13): podem ser utilizados como entradas ou saídas digitais de acordo com a necessidade do projeto e conforme foi definido no *sketch* criado na IDE.

- 6 pinos de entradas analógicas (pinos A0 - A5): são dedicados a receber valores analógicos, por exemplo, a tensão de um sensor. O valor a ser lido deve estar na faixa de 0 a 5 V onde serão convertidos para valores entre 0 e 1023.
- 6 pinos de saídas analógicas (pinos 3, 5, 6, 9, 10 e 11): são pinos digitais e podem ser programados para ser utilizados como saídas analógicas, utilizando modulação PWM.

A alimentação da placa pode ser feita a partir da porta USB do computador ou através de um adaptador AC. Para o adaptador AC recomenda-se uma tensão de 9 a 12 volts. (SOUZA, 2013).

3.1.2. Shields

O *Arduino* também pode ser estendido ao utilizar *Shields* (escudos), estes são placas de circuito que possuem outros dispositivos (por exemplo, receptores GPS, *displays* de LCD, módulos de *Ethernet* etc.), que aprimoram as capacidades do *Arduino* e podem ser plugadas na parte superior da placa e se fundamentam no mesmo objetivo do *Arduino*, montagem simplificada, produção barata e de distribuição *open-source*. (ARDUINO, 2017).

Na figura 10 é ilustrado o empilhamento de placas *Shields.*

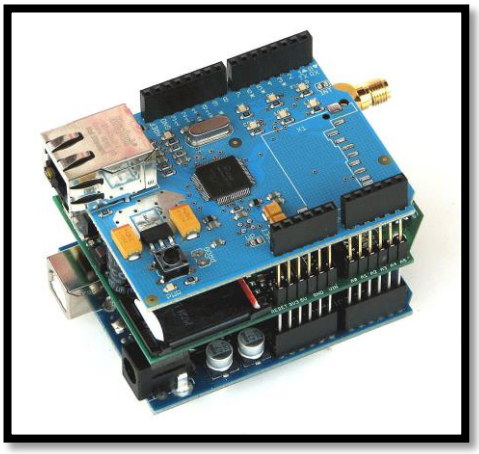


Figura 10: Empilhamento de *Shields*.

Fonte: (HWKITCHEN, 2017).

3.1.3. Sintaxe e funções básicas do *Arduino*

A linguagem *Arduino* é derivada do *Wiring* desenvolvido por Hernando *Barragan*, basicamente C/C++, possui funções simples e específicas para trabalhar com as portas do *Arduino*, e necessita de duas funções elementares para seu funcionamento: setup () e loop () (ARDUINO, 2017).

Existem inúmeras funções que podem ser utilizadas com o *Arduino* (ARDUINO, 2017), destaca-se:

- ***setup():*** No *Arduino* a função é chamada no momento em que o programa começa. É usada para inicializar variáveis, definir os modos de entrada ou saída dos pinos, indicar bibliotecas e outras tarefas. Essa função é executada somente uma vez, quando o *Arduino* é iniciado ou quando é resetado.
- ***loop():*** a função *loop* () faz precisamente o que seu nome indica: ela repete-se continuamente permitindo que seu programa funcione dinamicamente. É utilizada para controlar de forma ativa a placa *Arduino*.
- ***pinMode():*** configura o pino do *Arduino* apontado para que se comporte como entrada ou saída. Deve-se informar o número do pino que se deseja configurar e em seguida, se o pino será determinado como entrada (*INPUT*) ou saída (*OUTPUT*).
- ***digitalWrite():*** gera um valor *HIGH* (5 v ou 1) ou *LOW* (0 v ou 0) em um pino digital.
- ***digitalRead():*** lê o valor de um pino digital nomeado e retorna um valor *HIGH* ou *LOW*.
- ***analogRead():*** faz leitura de um pino analógico. O *Arduino* contém um conversor analógico-digital de 10 *bits*, com isto ele pode mapear voltagens de entrada entre 0 e 5 *Volts* para valores inteiros entre 0 e 1023.
- ***analogWrite():*** gera um valor analógico entre 0 e 255. Onda PWM *Pulse Width Modulation* ou Modulação por Largura de Pulso (MLP) é um método para obter resultados analógicos com meios digitais, esta função

no *Arduino UNO* está disponível nos pinos 3, 5, 6, 9,10 e 11. Para usar esta função deve-se informar o pino ao qual deseja escrever e em seguida informar um valor entre 0 e 255.

- ***Serial.begin():*** ajusta a taxa de transferência em *bits* por segundo para uma transmissão de dados pelo padrão serial. Para comunicação pode se usar uma destas taxas: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 57600, 115200.
- ***delay()***: suspende a execução do programa pelo tempo (em milissegundos) mencionado.

Estas funções fazem parte da linguagem de programação *Arduino* e torna a escrita do *sketch* simples e de fácil compreensão. Possibilita que iniciantes comecem a programar para o *Arduino* de forma muito rápida e sem se intimidar com a linguagem. Mais informações podem ser consultadas no *site* oficinal <https://www.arduino.cc/> (ARDUINO, 2017).

3.2.4. Ambiente de desenvolvimento do *Arduino*

O IDE está disponível em seu site oficial, quando conectado ao *hardware* do *Arduino* permite realizar o upload de programas e a comunicação entre eles. O código fonte dos programas escritos é chamado *sketches*, elas são salvas com a extensão *ino.* (RODRIGUES; SARTORI; GOUVEIA,2012).

Na figura 11 é ilustrada a interface da IDE do *Arduino.*

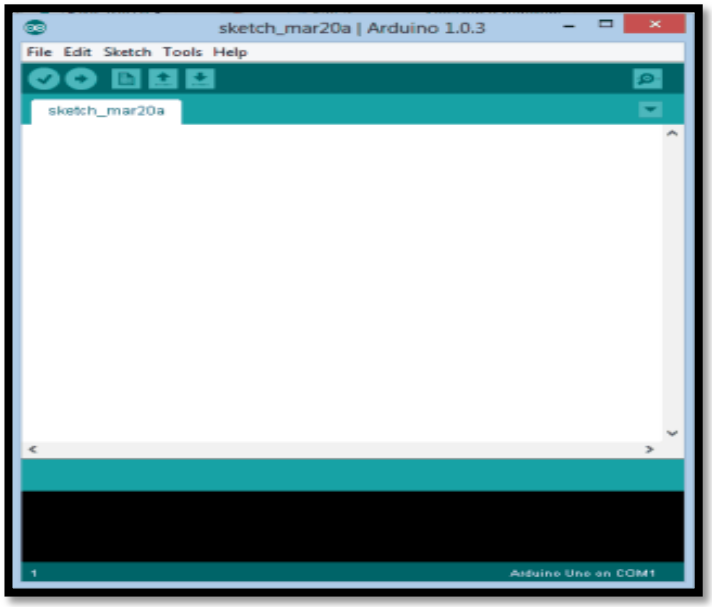


Figura 11: Interface da IDE do *Arduino.*

Fonte: (EMBARCADOS, 2017).

O IDE é dividido em três partes: A *Toolbar* no topo, o código ou a *Sketch Window* no centro, e a janela de mensagens na base. Na *Toolbar* há uma guia, ou um conjunto de guias, com o nome do *sketch*. Ao lado direito há um botão que habilita o serial monitor. No topo há uma barra de menus, com os itens *File, Edit, Sketch, Tools* e *Help*. Os botões na *Toolbar* fornecem acesso rápido às funções mais utilizadas dentro desses menus, com finalidade de facilitar aos desenvolvedores o papel de manipular o código.

Os principais atalhos para o ambiente de desenvolvimento do *Arduino* são:

- ***Verify***
    - Verifica se existe erro no código digitado.

- ***Upload***
    - Compila o código e grava na placa *Arduino* se corretamente conectada;

- ***New***
    - Cria um novo *sketch* em branco.
- ***Open***
    - Abre um *sketch*, presente no *sketchbook*.
- ***Save***
    - Salva o *sketch* ativo
- **Serial monitor**
    - Abre o monitor serial.

3.2. *RASPBERRY PI*

O *Raspberry PI* é um computador do tamanho de um cartão de crédito, foi anunciado e idealizado pelo inglês Pete Lomas em 2011, é considerado o menor computador, e é capaz de ser usado em projetos eletrônicos e diversas coisas, no qual o computador comum faz, como planilhas, processamento de textos, além de navegar na internet e jogar jogos. Seu código é aberto, portanto é uma tecnologia acessível (RASPBERRY, 2017).

Na figura 12 é ilustrada uma placa *Raspberry PI.*

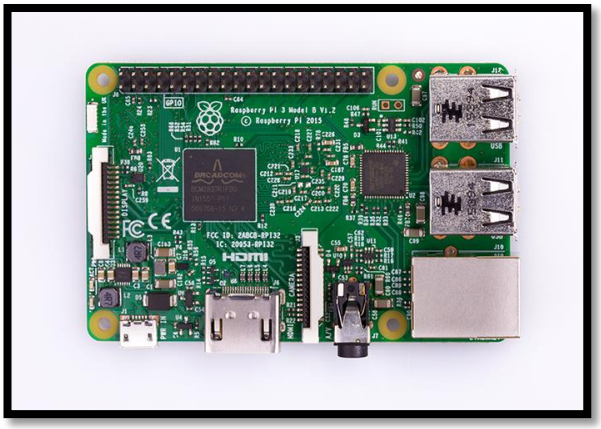


Figura 12: *Raspberry PI* 3 Modelo B.

Fonte: (RASPBERRY, 2017).

Atualmente existem dois tipos de *Raspberry PI* o modelo A e o Modelo B. Para um computador do tamanho de um cartão de crédito, pode-se dizer que tanto o modelo A quanto o modelo B são bem completos e eficientes em relação ao desempenho e Hardware, o que o torna um dispositivo bastante promissor. Existem poucas diferenças entre ambos, como por exemplo: o modelo A possui 256 MB de memória RAM, possui apenas uma porta USB 2.0 e não possui porta Ethernet. Atualmente o modelo B possui 512 MB de memória RAM, este contém duas portas USB 2.0 e uma porta Ethernet (RICHARDSON et al., 2013).

3.2.1. Sistema operacional

O *Raspberry PI* é utilizado como sistema operacional o Linux e também o *Free* BSD, uma plataforma de código aberto, onde o usuário é capaz de visualizar e alterar o código fonte de todo o sistema. É possível rodar várias distribuições do sistema operacional *Linux*. Existem diferenças no quesito de softwares em relação a um computador *desktop* ou *notebook* por conta do computador *Raspberry PI* ser baseado em chipset de dispositivo móvel. O processador tem características que exigem drivers de dispositivos especiais e softwares que não são encontrados em nenhuma distribuição *Linux* padrão (RICHARDSON et al., 2013).

Dentre os sistemas operacionais com *Raspberry PI* o mais utilizado é o *Raspbian*, sendo necessário para sua utilização um cartão SD de 4 GB (PEREIRA, 2015).

3.2.2. *Raspbian*

O *Raspbian* é o sistema operacional oficial do *Raspberry PI* baseado no Debian. Ele é encontrado no site do *Raspberry PI* e disponibiliza mais de 35.000 pacotes pré-compilados para o melhor desempenho possível no Raspberry *PI*. Este é distribuído pela Raspberry *PI* Foundation. Sendo um sistema quase completo, pois vem com diversas aplicações pré-instaladas, os drivers mais usuais, e ferramentas para facilitar algumas configurações

necessárias, entre outros. Muitas aplicações e módulos dedicados à programação vêm inclusos na imagem do Raspbian, basta iniciar o sistema para acessá-las (RASPBERRY, 2017).

3.3. MANIOT (MANAGEMENT FOR INTERNET OF THINGS)

É a plataforma para o gerenciamento dos dispositivos que compõe ambientes da IoT. Este ambiente corresponde a um domínio de aplicações, como sensores instalados fisicamente. As aplicações implementam um cenário beneficiado por uma infraestrutura IoT, como o gerenciamento de iluminação e segurança de uma residência, entre outros. A plataforma estabelece dois objetivos de gerenciamento, Local e Global/Remoto. O gerente local, por exemplo, atua dentro de um cenário e pode controlar os eventos em uma aplicação, como ligar ou desligar uma lâmpada (ANTUNES et al., 2016).

O gerente global/remoto uniformiza as ações realizadas em diferentes cenários a partir de diretivas de alto nível. Assim, uma concessionaria de energia através do gerente global, por exemplo, pode definir cotas máximas de consumo por região ou residência em períodos de potenciais blackouts (ANTUNES et al., 2016).

A plataforma estabelece um modelo de dados e de informações com o objetivo de padronizar o formato dos dados utilizados na comunicação entre aplicações, dispositivos e serviço. O estado dos dispositivos (ligado/desligado) e o *id* (identificação do dispositivo) são exemplos de características no modelo de informação. Visando a extensibilidade e integração com outros sistemas, a plataforma opera com protocolos e padrões populares da indústria para modelos de dados, como o XML (*eXtensible Markup Language*) e o *REST* (ANTUNES et al., 2016).

A ManIoT não requer grandes modificações ou instalações de softwares adicionais nos dispositivos da rede ou nos dos usuários, com exceção aos dispositivos Smartphones, pois leva em conta a heterogeneidade dos dispositivos. O acesso as aplicações da plataforma, é realizado através de uma interface Web, o acesso é restrito por contas de usuários e um administrador

define as aplicações e recursos dos dispositivos que esses usuários podem acessar (ANTUNES et al., 2016).

4 – PROPOSTA DO TRABALHO

Neste capítulo será apresentada a proposta do trabalho, que foi desenvolvida com a utilização das Plataformas de IoT Arduino e Raspberry PI e diferentes tecnologias, utilizadas para o desenvolvimento da modelagem e implementação de um protótipo de aplicação e cenário experimental para a validação dos conceitos sobre NFC.

4.1. TECNOLOGIAS E PLATAFORMAS ADOTADAS

4.1.1. NFC

Como mencionado anteriormente, o NFC trata-se de uma comunicação por rádio frequência a curto alcance, por meio desta caraterística torna-se ideal para realizar a troca de dados entre dispositivos de maneira segura, pois mantém a integridade dos itens compartilhados entre eles (ALECRIM, 2012).

4.1.2. Arduino

O *Arduino* é uma plataforma de código aberto, tanto de *hardware* quanto de *software,* baseado numa placa microcontroladora, bem como em um ambiente de desenvolvimento para escrever o código para a placa. O microcontrolador na placa é programado com a linguagem de programação *Arduino*, baseada na linguagem *Wiring*, e o ambiente de desenvolvimento *Arduino*, baseado no ambiente *Processing*. Os projetos desenvolvidos podem ser autônomos ou podem comunicar-se com um computador para a realização da tarefa, com uso de software específico *Flash, Processing, MaxMSP*. Os circuitos podem ser montados à mão ou comprados pré-montados; o software de programação de código-livre pode ser baixado de graça. (ARAÚJO et al, 2009).

4.1.3. RaspBerry PI

O *Raspberry Pi* é um computador do tamanho de um cartão de crédito capaz de ser usado em projetos eletrônicos e diversas coisas, no qual o computador comum faz, como planilhas, processamento de textos, além de navegar na internet e jogar jogos. Seu código é aberto, portanto é uma tecnologia acessível (RASPBERRY, 2017).

4.1.4. Java

Java é uma linguagem de programação e plataforma computacional lançada pela primeira vez pela Sun Microsystems em 1995. Os mentores eram Patrik Naughton, Mike Sheridan e James Gosling. O objetivo do projeto não era a criação de uma nova linguagem de programação, mas sim antecipar e planejar a próxima plataforma do mundo digital. Acreditavam que em algum tempo haveria uma convergência dos computadores com os equipamentos e eletrodomésticos usados no cotidiano. Desde seu início a tecnologia *Java* teve uma enorme utilização, logo empresas como a IBM, começaram a rodar aplicativos feitos em *Java*. As estimativas apontam que a tecnologia *Java* foi a mais incorporada em pouco tempo na história da informática, em 2003 o *Java* já tinha mais de 4 milhões de desenvolvedores. A tecnologia *Java*, é a base para praticamente todos os tipos de aplicações em rede, é o padrão global para o desenvolvimento e distribuição de aplicações móveis e incorporadas, jogos, conteúdo baseado na *Web* e *softwares* corporativo, pois o *Java* é rápido, seguro e confiável, sua tecnologia está em todo lugar e as vantagens dessa linguagem de programação são decorrentes do fato dela ser orientada a objetos, portável entre diferentes plataformas e sistemas operacionais (JAVA, 2017).

4.1.5. JavaFX

O *JavaFX* é uma plataforma desenvolvida pela *Oracle* baseada em *Java*. Têm como finalidade criações de *softwares* multimídia que podem ser executados em vários dispositivos diferentes, como por exemplo: criar *interfaces* gráficas, efeitos, animações, gráficos, tocar vídeo e áudio (ORACLE, 2017).

4.1.6. RxTx

O *RxTx* é uma biblioteca *Java* utilizada para criar aplicações que necessitam de comunicação com as portas serial/paralela do computador. Esta biblioteca usa *libs* nativas através da API JNI (*Java Native Interface*) para interagir com as portas serial/paralela (DEVMEDIA, 2017).

4.1.7. MongoDB

O *MongoDB* é o banco de dados *NoSQL* baseado em documento, livre e de código aberto, publicados sob a GNU (*Affero General Public License)*. Possui um modelo de dados flexíveis, como JSON, o que significa facilidade em armazenar e combinar dados de qualquer estrutura, sem renunciar as regras de validação, acesso a dados e funcionalidade de indexação rica. Podendo modificar dinamicamente o esquema sem tempo de inatividade, gastando menos tempo preparando dados para o banco de dados e mais tempo colocando seus dados para trabalhar. O modelo de documento mapeia os objetos no código do aplicativo, tornando os dados fáceis de trabalhar com consultas, indexação e agregação em tempo real, no qual fornecem maneiras poderosas de acessar e analisar seus dados. Possui um rápido desenvolvimento iterativo. Um modelo de dados flexível, juntamente com o esquema dinâmico e controladores idiomáticos tornam rápido para os desenvolvedores criar e evoluir aplicativos. O provisionamento e gerenciamentos automatizados possibilitam a integração continua e operações altamente produtivas (MONGODB, 2017).

4.1.8. Nodejs

O *Nodejs* é uma plataforma construída sobre o motor *JavaScript* do Google *Chrome* para garantir a facilidade de construir aplicações de rede rápidas. Utiliza o modelo de I/O direcionada a evento não bloqueante que o torna leve e eficiente ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos. Sendo assim, permite ao programador desenvolver aplicações web utilizando o JavaScript como linguagem de back-end. Com o Nodejs é possível criar servidores facilmente escaláveis capazes de responder a milhares de requisições simultâneas (NODEBR, 2016).

4.1.9. Nodemon

O módulo *Nodemon* é um utilitário responsável por monitorar todas as alterações nos arquivos de sua aplicação, assim sendo capaz de reiniciar o servidor toda vez que um arquivo do projeto for modificado (NODEMON, 2017).

4.1.10. NPM

O NPM (*Node Package Manager*) é um repositório online para publicação de projetos de código aberto para o *Nodejs,* também é um utilitário de linha de comando que interage com este repositório online que ajuda na instalação de pacotes, gerenciamento de versão e gerenciamento de dependências. Ele conta com mais de 35 mil pacotes, estes podem ser encontrados através do portal de busca da NPM. Assim que localizado o pacote pode ser instalado como uma única linha de comando (NODEBR, 2016).

4.1.11. Mongoose

O *Mongoose* é uma biblioteca do *Nodejs* no qual, proporciona uma solução baseada em esquemas para modelar os dados da sua aplicação. Possui um sistema de conversão de tipos, criação de consultas, validação e *hooks* para lógica de negócios, além de fornecer um mapeamento de objetos do *MongoDB* similar ao ORM (*Object Relational Mapping*), ou ODM *(Objetct Data Mapping*) no caso do *Mongoose*, ou seja, significa que traduz os dados do banco de dados para objetos *JavaScript* para que possam ser utilizados por sua aplicação (NODEBR, 2016).

4.1.12. Express

O *Express* é um dos *frameworks* mais populares de aplicação web para *Nodejs* leve e flexível que fornece uma camada fina de funcionalidades fundamentais para aplicação *web* e móveis. Construir aplicativos *web* com o *Nodejs* permite iterar e mudar a estrutura de seu aplicativo muito facilmente, porque é *JavaScript* tanto no lado do servidor e no lado do cliente. O *Express* cuida de várias coisas, permanecendo leve e flexível. Isso se encaixa muito bem em uma abordagem experimental e interativa (EXPRESS, 2017).
O Express suporte os diferentes métodos de roteamento, onde são derivados a partir de um dos métodos HTTP, e é anexado a uma instância da classe Express. O roteamento refere-se à determinação de como um aplicativo responde a uma solicitação do cliente por um *endpoint* específico, que é uma URI e uma solicitação HTTP (EXPRESS, 2017).

4.1.13. RESTful Web Service

Um *Web Service* é uma coleção de padrões usados e protocolos abertos, que são utilizados para trocar dados entre aplicativos ou sistemas. Os aplicativos de software escritos em diferentes linguagens de programação e executado em

várias plataformas simultaneamente pode-se utilizar *web services* para realizar trocas de dados (TUTORIALSPOINT, 2017).

O *RESTful* é um *Web Service* baseado na arquitetura *REST.* Ele utiliza métodos HTTP (*HyperText Transfer Protocol*) e define um URI *(Uniform Resource Identifier*), pois é um serviço que fornece a representação de recursos, como JSON e conjunto de Métodos HTTP (TUTORIALSPOINT, 2017).

4.2. ARQUITETURA PROPOSTA

O sistema será dividido em duas aplicações. A primeira aplicação faz o uso da plataforma *Raspberry PI* como servidor, onde realizara todo o gerenciamento de dados. A segunda aplicação é a cliente e faz uso da placa *Arduino* conectado com um *Shield* atuando como um leitor de cartões NFC, ela irá consumir os serviços oferecidos pelo servidor. Desta forma, esta aplicação será responsável por fornecer novos dados e consultar informações existentes e válidas a ponto de gerenciar usuários, assim permitindo novos usuários, modificar informações existentes, apagar e visualizar todos os cadastrados, além de monitorar e controlar o usuário a ponto de permitir ou não o seu acesso.

Na figura 13 é ilustrado o diagrama conceitual da proposta.

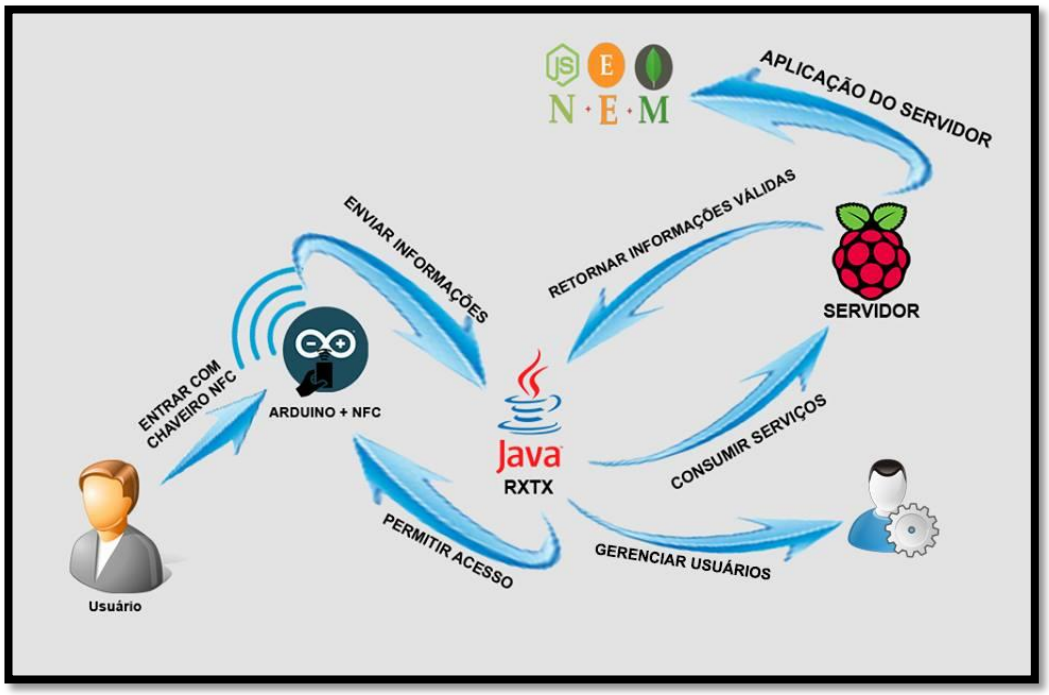


Figura 13: Diagrama conceitual da proposta.

A aplicação servidor será um *RESTful*, seu desenvolvimento fará uso do *Nodejs* e os módulos: *Nodemon* responsável por inicializar a aplicação. *Express* capaz de comunicar e expor seus serviços para a segunda aplicação. *Mongoose* o qual realizara a comunicação com o banco de dados *MongoDB*, que foi utilizado para armazenar todas as informações recebidas. A aplicação cliente será construída em *Java*, assim, sendo capaz de consumir os serviços expostos pelo servidor, além de receber e enviar informações para o *Arduino* através do *RxTx*. O *JavaFX* será o responsável pela interface gráfica da aplicação cliente, com o objetivo de facilitar a interação com o usuário.

Na figura 14 é ilustrado o diagrama de tecnologia e *Hardware.*

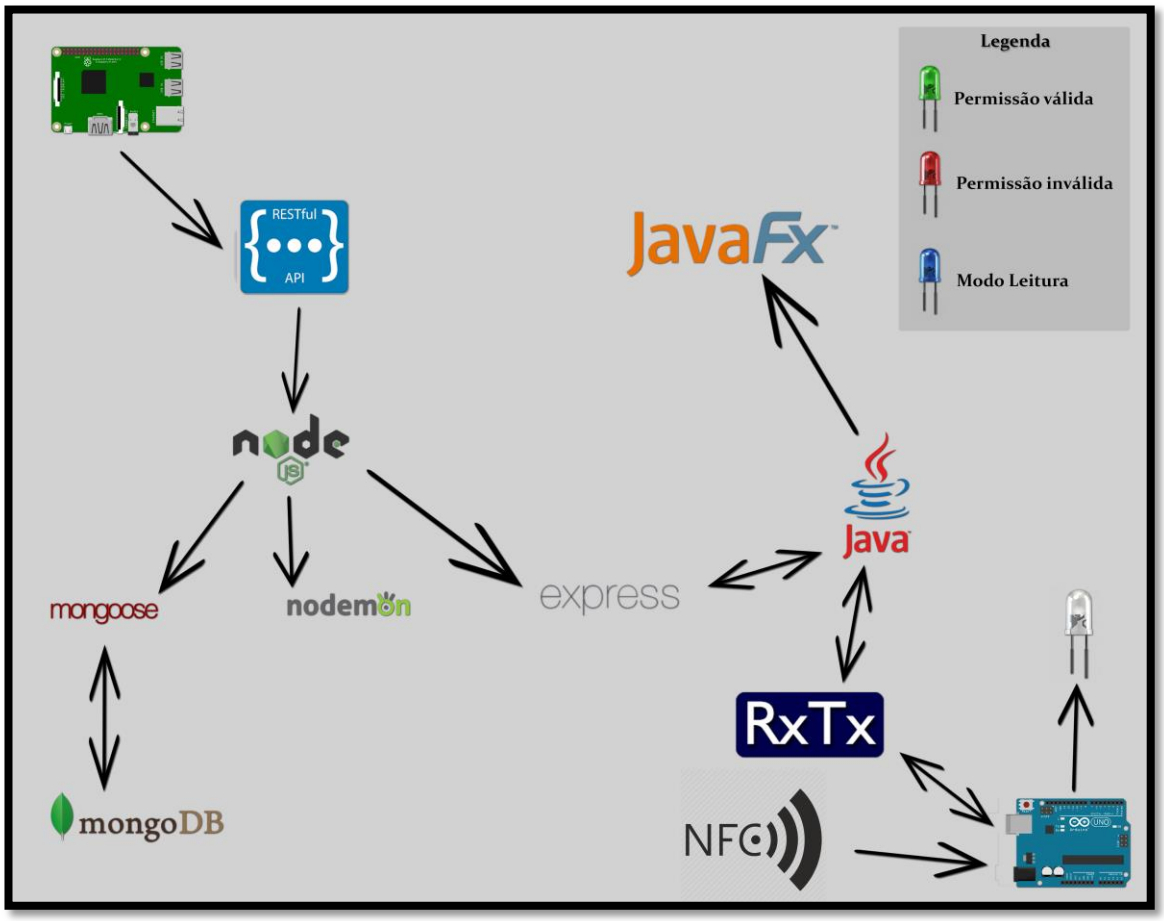


Figura 14: Diagrama tecnologia e Hardware.

5 – DESENVOLVIMENTO DO TRABALHO

O desenvolvimento do sistema para controlar acessos foi feito em etapas. Assim foram criadas duas aplicações, sendo elas, Servidor e Cliente.

5.1. SERVIDOR

O Servidor foi desenvolvido a partir do Raspberry PI 3, Modelo B utilizando o Sistema operacional *Raspbian Jessie.*

Na figura 15 ilustra a placa Raspberry PI 3 em funcionamento.

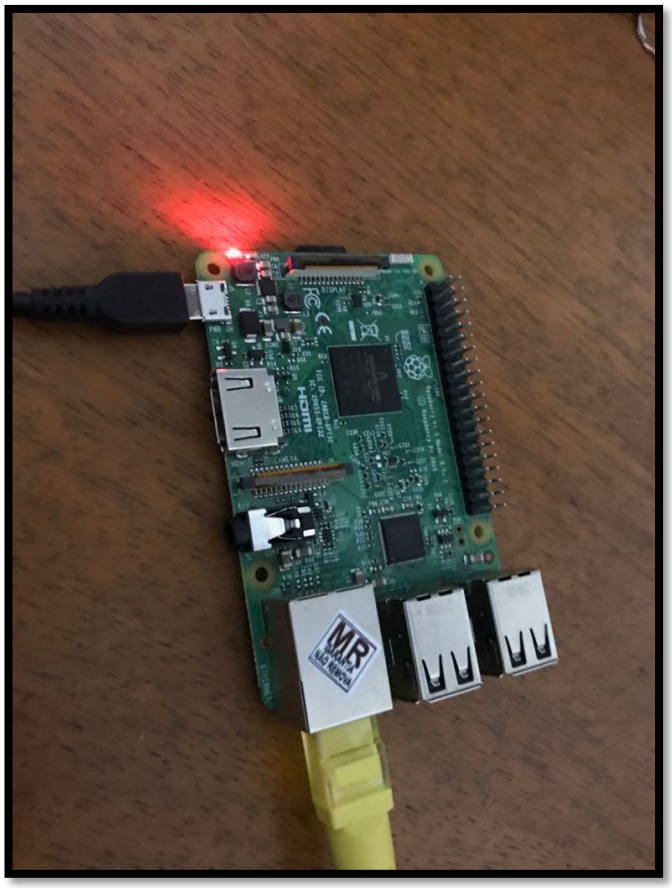
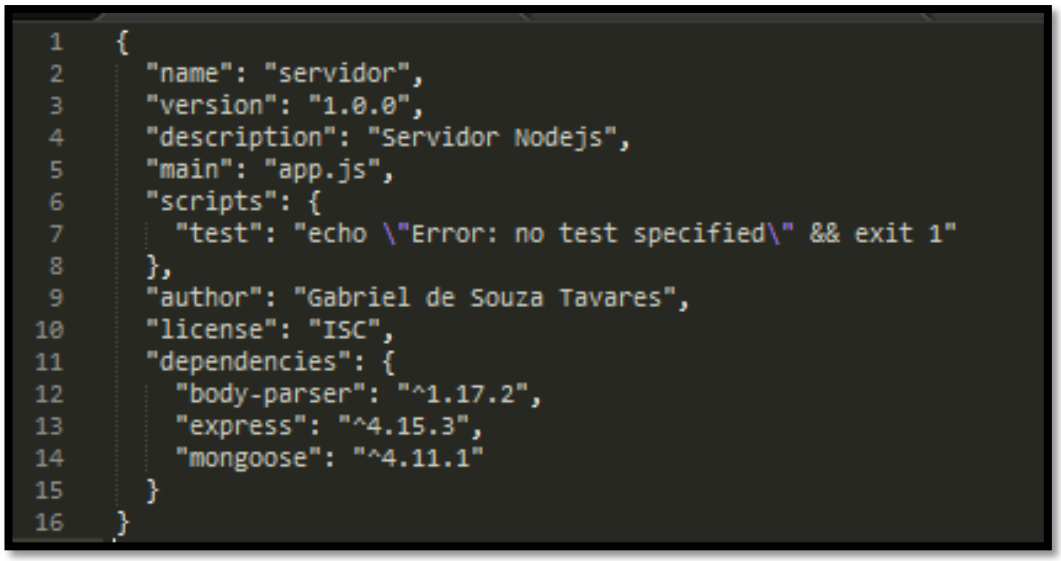


Figura 15: *Raspberry PI* 3 em funcionamento.

No desenvolvimento da aplicação servidor, foi utilizado o editor de texto *Sublime Text* 3, para implementar códigos os quais foram criados em camadas. Com o *Nodejs* e o seu gerenciador de pacotes NPM instalados, através do terminal com o comando npm init é criado o projeto com um documento *json* obtendo as informações da aplicação, e o comando npm *install* para instalar as dependências utilizadas, sendo elas o *body-parser*, *express* e o *mongoose*.

A imagem 16 ilustra o projeto construído através do NPM.



```
{
  "name": "servidor",
  "version": "1.0.0",
  "description": "Servidor Nodejs",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Gabriel de Souza Tavares",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.17.2",
    "express": "^4.15.3",
    "mongoose": "^4.11.1"
  }
}
```

Figura 16: Projeto construido.

No arquivo de configuração do servidor são listadas as informações da API sendo elas: porta do servidor e *setHeader* para possibilitar que aplicações externas possam consumir os serviços oferecidos além de configurar os métodos HTTP existentes na aplicação e especificar o formato do documento que será criado.

Na imagem 17 ilustra o arquivo de configuração da API.

```
var express = require ('express');
var bodyParser = require ('body-parser');
var port = '3000';

var app = module.exports =  express();

app.listen(port);

app.use(bodyParser.urlencoded({extended:true}));
app.use(bodyParser.json());
app.use(function(req, res, next){
    res.setHeader('Acess-Control-Allow-Origin','*');
    res.setHeader('Acess-Control-Allow-Methods', 'GET,POST, PUT, DELETE');
    res.setHeader('Acess-Control-Allow-Headers', 'Application/json, content-type, Authorization');
    next();

})
```

Figura 17: Arquivo configuração da API.

O módulo *Mongoose* é responsável por criar o *schema* usuário e realizar a comunicação entre a aplicação e o banco de dados *MongoDB*, através do seu método *connect*.

A figura 18 ilustra o arquivo configuração responsável por comunicar-se com o banco de dados.

```
var mongoose = require ('mongoose');

var urlString = 'mongodb://localhost/API';

mongoose.connect(urlString, function(err, res){
    if(err){
        console.log('Não foi possivel conectar a: ' + urlString);
    }
    else{

        console.log('Conectado a: ' + urlString);
    }


});
```

Figura 18: Arquivo configuração do banco de dados.

O *Express* é responsavel por criar as rotas da aplicação, através delas pode-se gravar e apagar informações do servidor pelos metodos *HTTP Get, Post, Put* e *Delete*.

Na figura 19 são ilustradas as rotas do *Express* utilizando os métodos HTTP.

```
userrouter.post('/cadastrar', function(req,res){
    var nome = req.body.nome;
    var cpf = req.body.cpf;
    var tag = req.body.tag;

    userController.save(nome, cpf, tag, function(resp){
        res.json(resp);
    });
});

userrouter.put('/atualizar/:id', function(req, res) {

    var id = req.params.id;
    var nome = req.body.nome;
    var cpf = req.body.cpf;
    var tag = req.body.tag;

    userController.update(id, nome, cpf, tag, function(resp) {

        res.json(resp);
    });
});

userrouter.delete('/deletar/:id', function(req,res){
    var id = req.params.id;
    userController.delete(id, function(resp){
        res.json(resp);
    });
});
userrouter.get('/listart/:tag', function(req, res) {
    var tag = req.params.tag;

    userController.buscatag(tag, function(resp) {
        res.json(resp);
    });
});
```

Figura 19: Arquivos das rotas do *Express*.

5.2. CLIENTE

A construção do ambiente físico da aplicação é constituída por um *Arduino Uno,* para enviar e receber informações da aplicação em *Java*, um *shield* NFC a ponto de transmitir para a placa *Arduino* a característica de ler dispositivos NFC, *led* RGB, para gerar diferentes cores com a finalidade de ilustrar o *status* de validação aos acessos de usuários. Para conectar o led com a placa são necessários três resistores de 330 ohms, quatro jumpers e uma *Protoboard*.

A figura 20 ilustra os componentes utilizados para o desenvolvimento do ambiente físico.

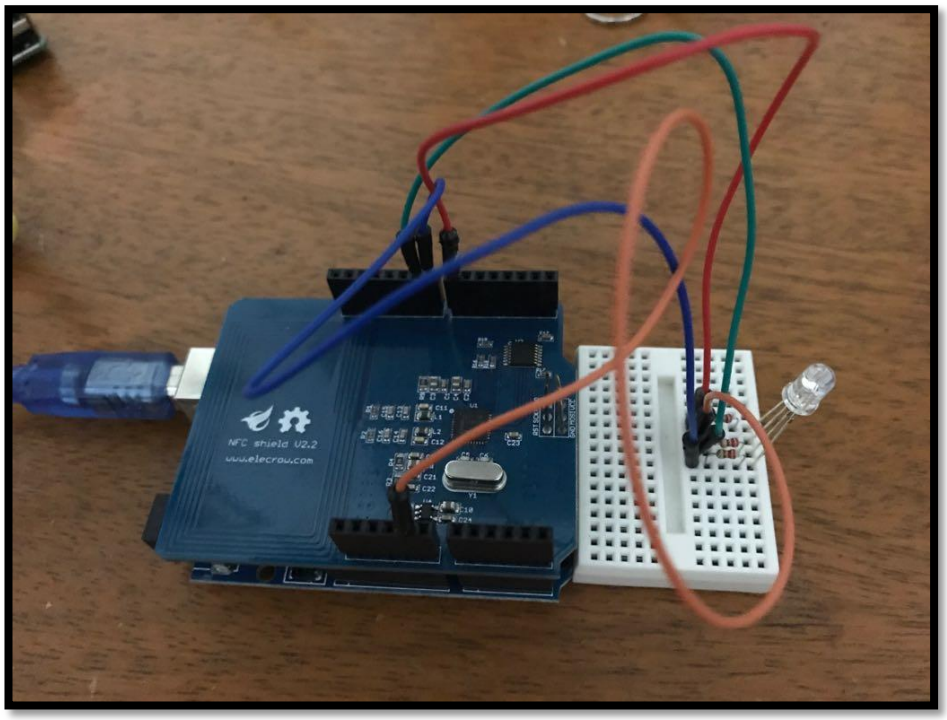


Figura 20: Ambiente físico montado.

Cada chaveiro NFC possui uma identificação nativa, será por meio desta identificação que o servidor irá reconhecer o usuário existente. É necessário importar a biblioteca *PN532* para o funcionamento da *shield*, assim ela atuará com sua funcionalidade *reader* para realizar a leitura dos dispositivos.

A identificação do chaveiro é formada por blocos, o servidor armazenará os últimos dez caracteres do quarto bloco.

Na figura 21 ilustra a leitura de chaveiros realizada pelo *Arduino* através do serial monitor.

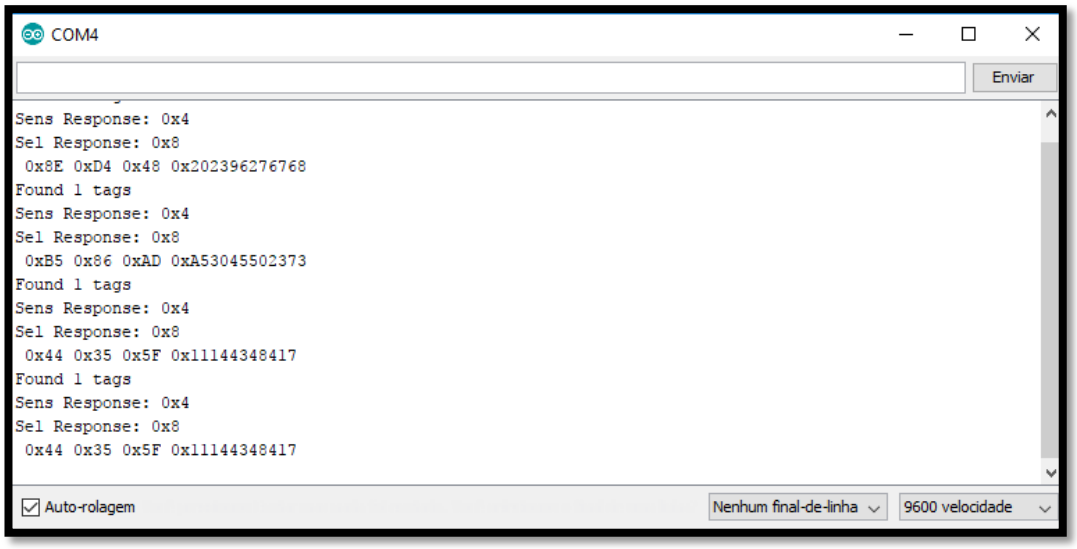


Figura 21: Leitura de chaveiros NFC.

A comunicação entre o *Arduino* e o *Java* ocorre através do RxTx, ele é invocado como interface e seus métodos são implementados nos arquivos que necessitam receber informações dos chaveiros, além de retornar para *Arduíno,* no qual o led RGB deverá acender a cor dependendo de cada ação.

Na figura 22 é ilustrado o trecho de código onde o componente *Text View* recebe a identificação do chaveiro NFC.

```
public void serialEvent(SerialPortEvent spe) {
    if (spe.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
        try {
            if (input.ready()) {
                String inputLine = input.readLine();
                if (inputLine.length() > 25) {
                    txtTag.setText("");
                    txtTag.setText(inputLine.substring(inputLine.length() - 10));
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }

    }

}
```

Figura 22: Recebendo informações NFC.

A aplicação cliente comunica-se com o servidor através do protocolo HTTP, para isso é necessário expor o *IP* e a porta atual do servidor. Assim, é capaz de realizar requisições para consultar informações já existentes, enviar novos dados ou até mesmo solicitar que informações existentes sejam removidas.

Na figura 23 é ilustrado o trecho de código que realiza a comunicação com o servidor e cadastra novos usuários.

```
String uri = "http://192.168.15.6:3000/";

@Override
public boolean inserir(User user, Response resp) throws Exception {
    try {
        Client client = ClientBuilder.newClient(new ClientConfig());
        WebTarget webTarget = client.target(uri).path("users/cadastrar");

        Invocation.Builder invocationBuilder = webTarget.request(MediaType.APPLICATION_JSON);

        resp = invocationBuilder.post(Entity.json(new Gson().toJson(user)));


        System.out.println(resp.getStatus());
        String responseString = resp.readEntity(String.class);
        System.out.println(responseString);

        return true;
    } catch (Exception e) {
        throw e;
    } finally {
        resp.close();
    }

}
```

Figura 23: Comunicação entre cliente e servidor.

5.3. FUNCIONAMENTO

Para o funcionamento do cliente é necessário que os serviços do servidor estejam em funcionamento. Assim, para iniciar o *mongoDB* utiliza-se o comando /etc/init.d/mongodb/ start. Após o serviço de banco de dados ser iniciado, basta ir ao diretório da aplicação e digitar *nodemon* para executar o servidor.

Na figura 24 ilustra o servidor em funcionamento.

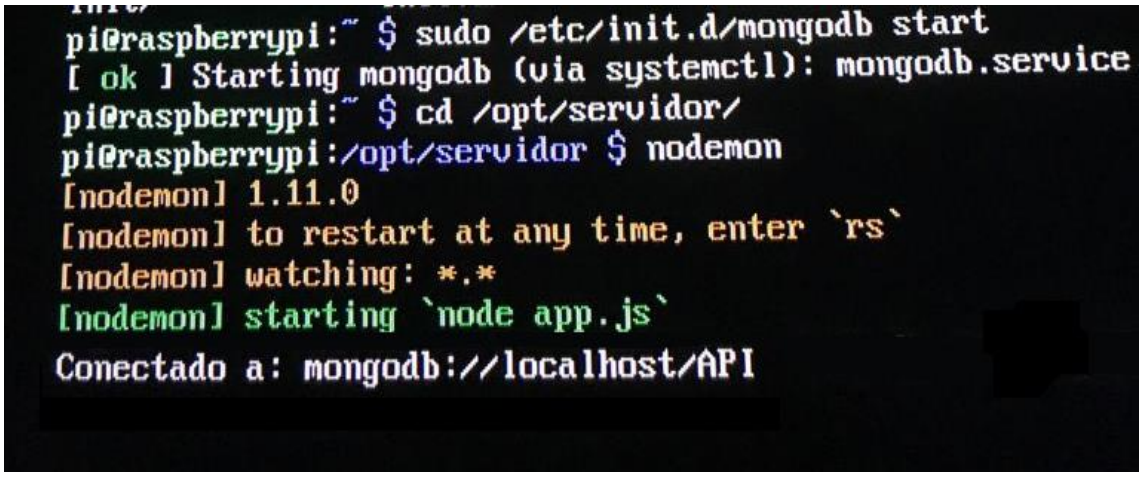


Figura 24: Servidor em funcionamento.

Ao iniciar a aplicação cliente uma tela é aberta, com dois ícones, o primeiro sendo o *play,* responsável por ligar o monitoramento, sendo ele responsável por realizar o controle de acessos. O segundo botão, irá se comunicar com a tela responsável por gerenciar usuários.

Na figura 25 é ilustrada a tela inicial da aplicação.

Figura 25: Aplicação inicial.

A tela de gerenciar é constituída por quatro botões, eles completam todas as funcionalidades para controlar usuários, sendo elas: cadastrar novos, visualizar todos os existentes, modificar informações e remover. Ao clicar em uma das opções o modo de leitura NFC é ativado.

Na figura 26 é ilustrada a tela responsável por gerenciar usuários.

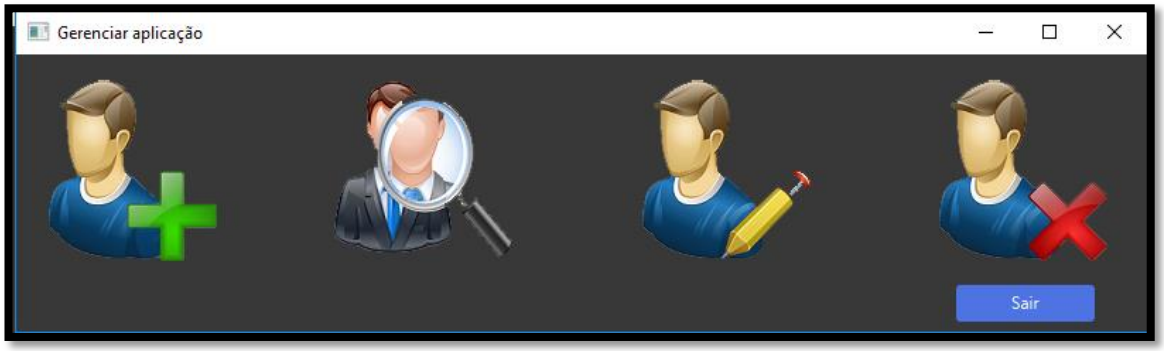


Figura 26: Gerenciar usuários.

Ao clicar no botão *play* será aberta a tela para monitorar o controle de acessos de usuários, inicialmente a tela irá mostrar uma imagem de um *led* azul, onde

representa o modo de leitura dos chaveiros e uma tabela em branco por não haver nenhuma informação do *Arduino*.

Na figura 27 é ilustrado o controlador de acessos no seu estado inicial.

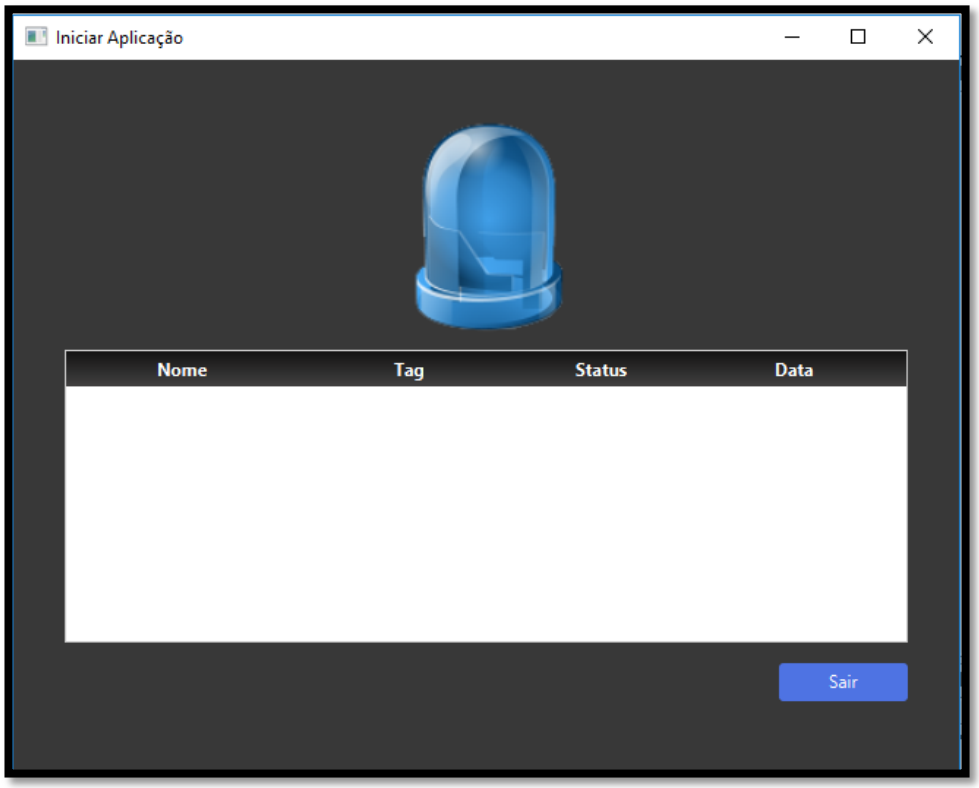


Figura 27: Controlador de acessos no estado inicial.

Simultaneamente a placa *Arduino* irá ativar o modo de leitura para detectar os chaveiros NFC.

Na figura 28 é ilustrada a placa *Arduino* no modo de leitura.

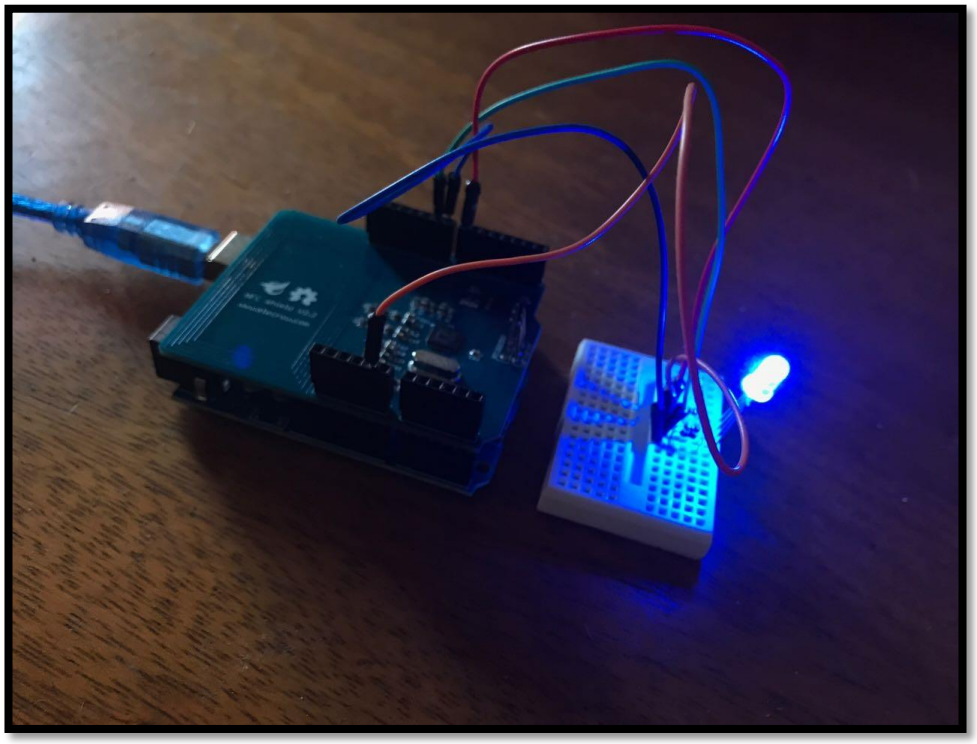


Figura 28: Arduino no modo leitura.

Ao aproximar o chaveiro com informações existentes à tela será atualizado automaticamente deixando visíveis informações como nome, *tag*, status e data de acesso, além de ilustrar momentaneamente um *led* verde representando o *Arduino* recebendo informações válidas.

Na figura 29 é ilustrado o controlador de acessos recebendo um chaveiro válido.

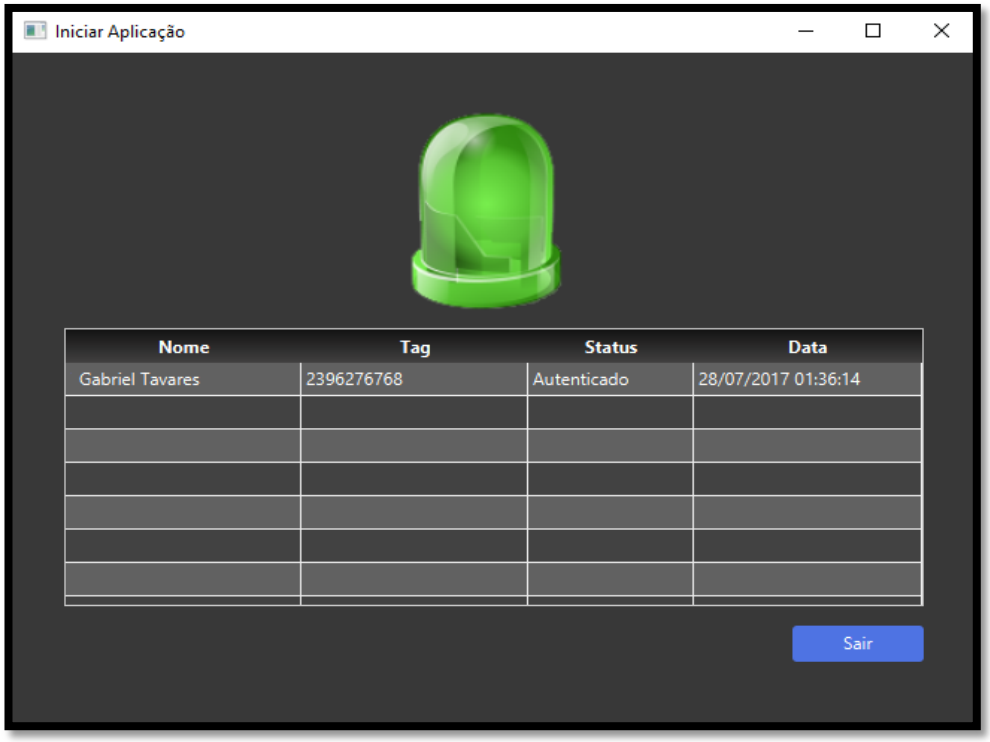


Figura 29: Controlador de acessos recebendo informações validas.

Ao detectar chaveiros cadastrados no servidor o *Arduino* é responsável por modificar a cor do *led* para verde, assim torna-se visível para o usuário seu acesso válido.

A figura 30 ilustra o *Arduino* recebendo informações válidas.

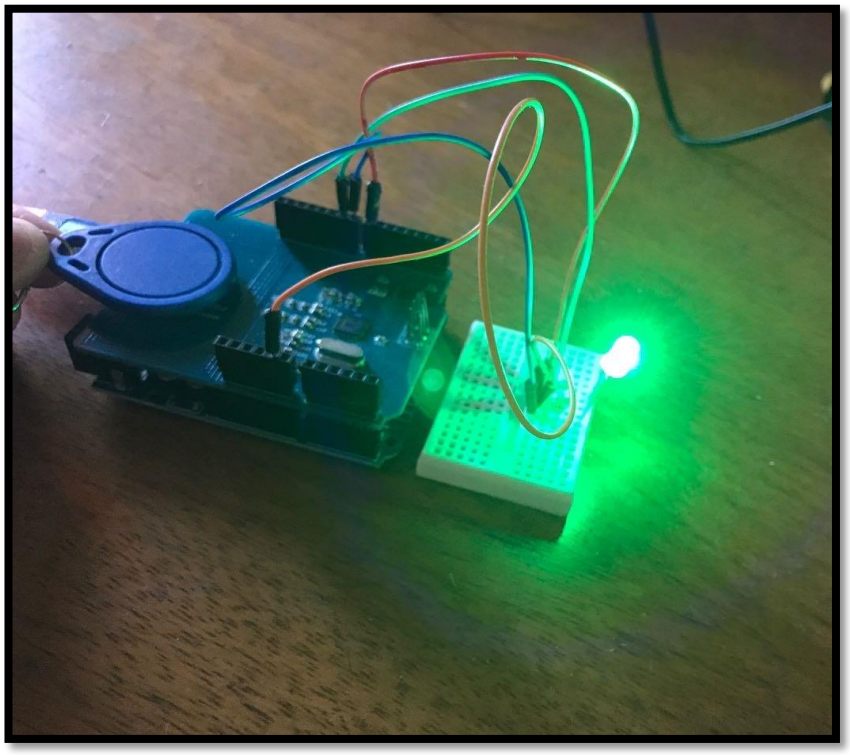


Figura 30: Arduino recebendo informações válidas.

Na tentativa de utilizar um chaveiro com informações inexistentes a tela será atualizada com um *led* vermelho momentâneo que irá representar o *Arduino* negando o acesso do chaveiro não cadastrado. A tabela responsável pelo monitoramento irá gravar a tentativa de acesso do chaveiro inexistente.

Na figura 31 mostra o controlador de acessos recebendo informações inválidas.

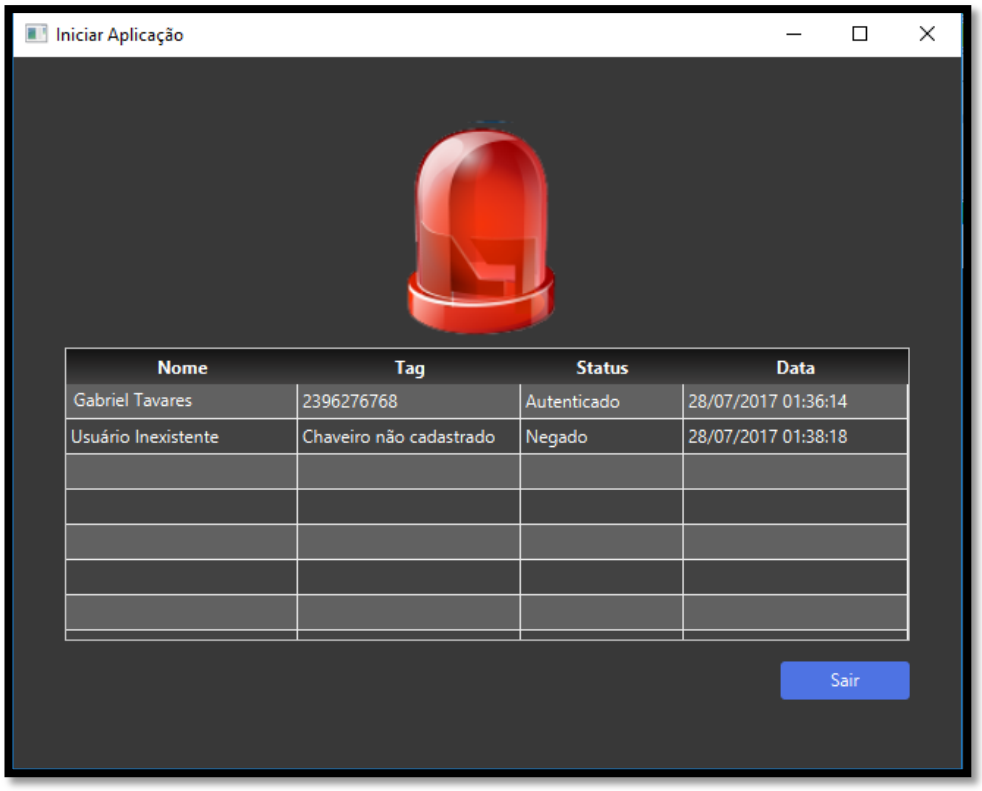


Figura 31: Controlador de acessos recebendo informações inválidas.

Quando detectar chaveiros não cadastrados no servidor o *Arduino* é responsável por modificar a cor do *led* para vermelho, assim torna-se visível que o acesso foi recusado.

Na imagem 32 mostra o *Arduino* recebendo um chaveiro inválido.

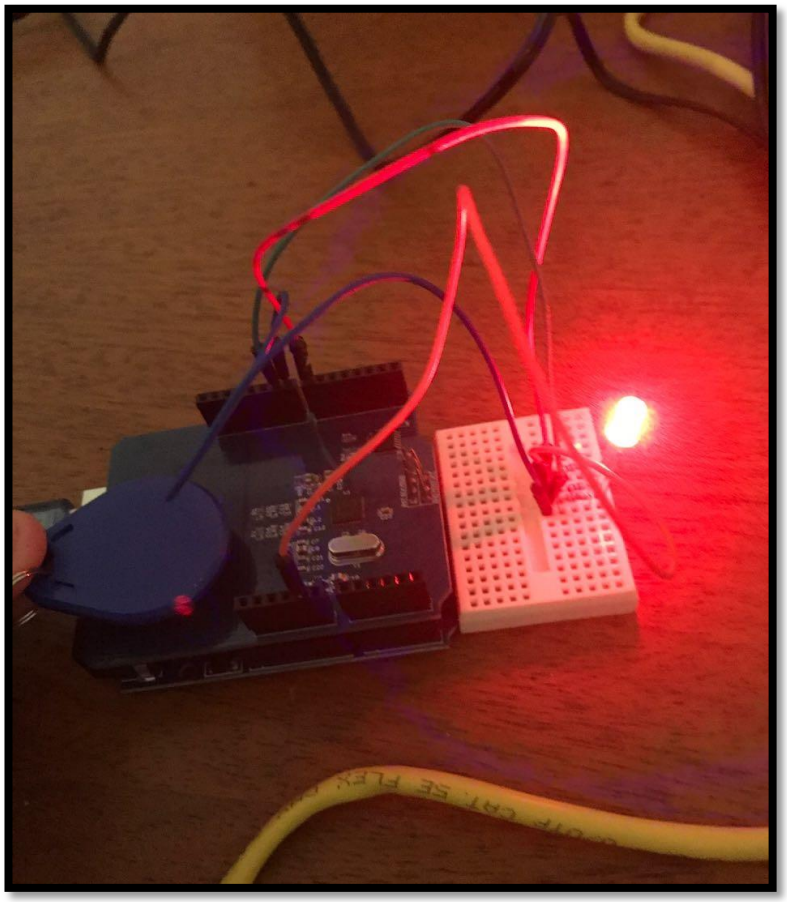


Figura 32: Arduino recebendo informações inválidas.

6 – CONCLUSÃO

A ideia básica por trás do conceito de IoT é a presença generalizada de uma variedade de objetos, tais como etiquetas, sensores, atuadores, telefones celulares, etc. Que tem o potencial de contribuir em vários aspectos da vida contemporânea, propiciando uma vasta gama de aplicações que facilitem tarefas cotidianas.

O estudo sobre o NFC permite compreender que esta é uma tecnologia muito promissora e com aplicações múltiplas, explorando suas características, funcionalidades e peculiaridades, podemos concluir então, que a tecnologia NFC tem definitivamente várias utilidades. Pode ser utilizada para pagamentos, sincronização de dados, *marketing*, jogos, interação entre usuários através da troca de arquivos, imagens e vídeos, entre outras ocasiões.

Considerando as características do *Raspberry Pi* pode-se dizer que é uma plataforma economicamente viável e possui um alto poder computacional para todo o gerenciamento dos softwares e aplicativos instalados nele. Em relação à tecnologia *Arduino* fica visível a facilidade de implementar aplicações mesmo sem ter um grande conhecimento de microcontroladores, programação e eletrônica, pode concluir também que é muito amplo as variedades de aplicações que podem ser criadas com ele, pois com a utilizações de placas *shields* e módulos a placa *arduino* passa a possuir novas características e funcionalidades.

6.1. TRABALHOS FUTUROS

A partir da arquitetura implementada neste trabalho é possível utilizar outros componentes, como travas eletrônicas e *displays* LCD, a fim de complementar a aplicação em trabalhos futuros. Pode-se também vincular Shields Wireless para realizar uma comunicação direta com o servidor através de requisições HTTP.

É possível utilizar o *AngularJS* para comunicar-se com o *Arduino*. Assim o sistema será completamente em *JavaScript*, tanto o lado cliente quanto o

servidor, esta combinação de tecnologias é conhecida como *MEAN Stack* e é ideal para rápida prototipação de software e desenvolvimento de aplicações escaláveis.

REFERÊNCIAS

ALECRIM, Emerson. ***O que é NFC (Near Field Communication)?.*** 2012. Disponível em: <https://www.infowester.com/nfc.php#funcionamento>. Acesso em 17 de fevereiro de 2017.

ANTUNES, Josué B; DÉNES, Istvan L; SANTOS Marconi; CASTRO, Tiago O; MACEDO, Daniel F; SANTOS, Aldri L. **ManIoT: Uma Plataforma para Gerenciamento de Dispositivos da Internet das Coisas.** 2016. 14p. Departamento de Ciência da Computação. UFMG, Minas Gerais. Núcleo de Informática, IFNMG, Araçuaí. Departamento de Informática. UFPR, Curitiba.

ARDUINO. **Arduino**. Disponível em: <https://www.arduino.cc/>. Acesso em: Fevereiro de 2017.

ARAÚJO, Guilherme; FRAULOB, Halph; LAMBERT, Pedro; BONADIO, Vagner. 2009. **Projeto 39**. Curso de Engenharia Elétrica da Escola Politécnica da USP. Disponível em: <https://projeto39.wordpress.com/o-arduino/>. Acesso em 20 de fevereiro de 2017.

ASSIS, F. N; RODRIGUES, L; SARTORI, E; GOUVEIA, B. **Introdução ao Arduino.** Apostila. Faculdade de Computação, Universidade Federal de Mato Grosso do Sul. Campo Grande, MS. 25p, 2012.

CRUZ, Kelly Cristina de Oliveira. **Estudo sobre o *Near Field Communication* e seu papel em pagamentos via dispositivos móveis**. 2013. 64p. Trabalho de Conclusão de Curso (Pós- Graduação *Lato Sensu* na área de Rede de Computadores, com ênfase em Segurança) – Centro Universitário de Brasília, Brasília, 2013.

CUNHA, Alessandro. **NFC (Near Field Communication) – Aplicações e uso.** 2016. Disponível em: <https://www.embarcados.com.br/nfc-near-field-communication/>. Acesso em 26 de fevereiro de 2017.

DEVMEDIA. **Raspberry Pi e a biblioteca Java RxTx.** 2017. Disponível em < http://www.devmedia.com.br/raspberry-pi-e-a-biblioteca-java-rxtx/32722>. Acesso em 15 de maio de 2017.

ECMA INTERNATIONAL. **Near Field Communication – White Paper**. 2004. Disponível em: <https://www.ecma international.org/activities/Communications/tc32-tg19-2005-012.pdf>. Acesso em 20 de fevereiro de 2017.

ELECROV. **Protetor NFC.** 2017. Disponível em <https://www.elecrow.com/wiki/index.php?title=NFC_Shield>. Acesso em 16 de maio de 2017.

EMAZE. Disponível em <https://www.emaze.com/@ALRTFIQO/Presentation-Name>. Acesso em 19 de fevereiro de 2017.

EVANS, Dave. **A Internet das Coisas- Como a próxima evolução da Internet está mudando tudo.** Cisco Internet Business Solutions Group (IBSG). 2011. Disponível em <http://www.cisco.com/c/dam/global/pt_br/assets/executives/pdf/internet_of_things_iio_ibsg_0411final.pdf>. Acesso em 10 de novembro de 2016.

EXAME. **50 bi de dispositivos estarão conectados à internet até 2020.** São Paulo. 2014. Disponível em <http://exame.abril.com.br/tecnologia/50-bi-de-dispositivos-estarao-conectados-a-internet-ate-2020/>. Acesso em 09 de novembro de 2016.

EXELANZ. **Mobile Payments Enabled by Host Card Emulation (HCE).** Disponível em <http://www.exelanz.com/blogs/mobile-payments-enabled-by-host-card-emulation-hce/>. Acesso em 19 de fevereiro de 2017.

EXPRESS, 2017. Disponível <http://expressjs.com/pt-br/>. Acesso em 15 de maio de 2017.

EXPRESS, 2017. Disponível em <http://expressjs.com/pt-br/guide/routing.html>. Acesso em 15 de maio de 2017.

HARRISON, Colin; ECKMAN, Barbara; HAMILTON, Rick; HARTSWICK, Perry; KALAGNANAM, Jayant; PARASZCZAK, Jurij; WILLIAMS, Petter. **Foundations for Smarter Cities**. IBM Journal of Research and Development, Armonk, New York. 2010. Disponível em < http://fumblog.um.ac.ir/gallery/902/Foundations%20for%20Smarter%20Cities.pdf>. Acesso em 14 de novembro de 2016.

HAUBENSAK, Oliver. Smart cities and Internet of things. Business Aspects of the Internet of Things. In: TOMAS, GUSTAVO H R P. (Org.). **Uma Arquitetura de Software para Cidades Inteligentes baseada na Internet das Coisas,** Recife, UFPE, fev. 2014.

HWKITCHEN. Disponível em: <http://www.hwkitchen.com/products/gsm-playground/>. Acesso em 19 de fevereiro de 2017.

JARABA, Francisco Borrego; RUIZ, Irene Luque; NIETO, Miguel Ángel Gómez. **A NFC–based pervasive solution for city touristic surfing.** 2011. University of Córdoba, Espanha. Disponível em <http://cmapspublic2.ihmc.us/rid=1NTPTTG1H23CGML224RM/NFC%20For%20City%20Touristic%20Surfing.pdf>. Acesso em 11 de novembro de 2016.

JAVA. **O que é a tecnologia Java e por que preciso dela?**. 2017. Disponível em: <https://www.java.com/pt_BR/download/faq/whatis_java.xml>. Acesso em: 17 de maio de 2017.

KOMNINOS, Nicos. Intelligent Cities: Innovation, Knowledge Systems and Digital Spaces. In: TOMAS, GUSTAVO H R P. (Org.). **Uma Arquitetura de Software para Cidades Inteligentes baseada na Internet das Coisas**, Recife, UFPE, fev. 2014.

LIMA, Danilo Souza. **Proposta para controle de acesso físico seguro baseada em Near Field Communication (NFC) e Android.** 2013. 99p. Trabalho de Conclusão de Curso (Engenharia Elétrica com ênfase em Eletrônica) – Departamento de Engenharia Elétrica - Escola de Engenharia de São Carlos da Universidade de São Paulo, São Carlos, 2013.

MENESES, Emerson Barros de. **Rede Wireless: uma solução sem fio.** 2009. 27p. Trabalho de Conclusão de Curso – Departamento de Pesquisa e Pós-Graduação – Universidade Cândido Mendes, Rio de Janeiro, 2009.

MONGODB. **O que é MongoDB?.** 2017. Disponível em <https://www.mongodb.com/what-is-mongodb>. Acesso em 16 de maio de 2017.

NASSAR, Victor; VIEIRA, Milton Luiz Horn. **A Internet das coisas com as Tecnologias RFID E NFC.** 2014. 13p. 11º Congresso Brasileiro de Pesquisa e Desenvolvimento em Design. Gramado, RS. Universidade Federal de Santa Catarina, SC.

NFCFORUM. Disponível em: <http://nfc-forum.org/>. Acesso em 18 de fevereiro de 2017.

NFC WORD. Disponível em: < https://www.nfcworld.com/2012/02/24/313725/orange-uk-announces-quick-tap-treats-for-200000-nfc-customers/>. Acesso em 19 de fevereiro de 2017.

NODEBR**. Nodejs e MongoDB- Introdução ao Mongoose.** 2016. Disponível em <http://nodebr.com/nodejs-e-mongodb-introducao-ao-mongoose/>. Acesso em 17 de maio de 2017.

NODEBR. **O que é Node.js?.** 2016. Disponível em <http://nodebr.com/o-que-e-node-js/>. Acesso em 17 de maio de 2017.

NODEBR. **O que é a NPM do Node.JS**. 2016. Disponível em <http://nodebr.com/o-que-e-a-npm-do-nodejs/>. Acesso em 20 de julho de 2017.

NODEMON, 2017. Disponível em <https://nodemon.io/>. Acesso em 20 de julho de 2017.

ORACLE. **JavaFX Technologies At a Glance.** 2017. Disponível em <http://www.oracle.com/technetwork/pt/java/javafx/tech/index.html>. Acesso 21 de julho de 2017.

PEREIRA, Renata I. S; JUCA, Sandro C. S. **Sistema Embarcado Linux baseado em Raspberry Pi para gravação online de Microcontroladores PIC.** 2015. 6p. Departamento de Engenharia Elétrica. Universidade Federal do Ceará. Campus do PICI. Departamento de Telemática. Instituto Federal de Educação, Ciência e Tecnologia do Ceará. Fortaleza, 2015.

RASPBERRY. Disponível em: <http://raspberrypi.org/>. Acesso em 24 de fevereiro de 2017.

RICHARDSON, Matt; WALLACE, Shawn. **Primeiros Passos com o Raspberry Pi**, 1. ed. São Paulo: Novatec Editora Ltda, 2013.

SAB, Gabriel Augusto Amim; FERREIRA Rafael Cardoso; ROZENDO, Rafael Gonçalves. **Near Field Communication**. Engenharia de Computação e Informação. UFRJ, 2013.

SOARES, Cristiane da Silva; ALVES Thays de Souza. **Sociedade da Informação no Brasil: Inclusão digital e a importância do profissional de TI.** Centro Universitário Carioca, 2008.

SOUZA, Fábio. **Arduino – Primeiros Passos.** 2013. Disponível em: <https://www.embarcados.com.br/arduino/>. Acesso em 19 de fevereiro de 2017.

SPINSANTE, Susanna; GAMBI, Ennio. **NFC–Based User Interface for Smart Environments.** 2015. Departamento de Engenharia de Informação, Universidade Politécnica de Marche, Ancona, Itália. Disponível em <https://www.hindawi.com/journals/ahci/2015/854671/#B5>. Acesso em 10 de novembro de 2016.

TUTORIALSPOINT, **Node.js- API RESTful**, 2017. Disponível em <https://www.tutorialspoint.com/nodejs/nodejs_restful_api.htm>. Acesso em 16 de maio de 2017.

WHITMORE, Andrew; AGARWAL, Anurag; XU, Li Da. **The Internet of Things—A survey of topics and trends.** 2014. Information Systems Frontiers, New York. Disponível em <http://docplayer.net/678790-The-internet-of-things-a-survey-of-topics-and-trends.html>. Acesso em 14 de novembro de 2016.

ZEINDIN, Denise Carla; DALFOVO, Oscar; AZAMBUJA, Ricardo Alencar de; DIAS, Paulo Roberto. **A Tecnologia do Futuro WI-FI (*Wireless Fidelity*).** 2003. 9p. Departamento de Sistemas e Computação. Universidade Regional de Blumenau, Santa Catarina, Blumenau, 2012.