Universidade Federal de Minas Gerais

Projeto Orientado em Computação II

- Projeto Tecnológico -

Smart Connected Door System

Guilherme Resende Vieira

Orientador: Daniel Fernandes Macedo

Dezembro de 2019

Resumo

O principal objetivo do presente trabalho é criar um dispositivo de IoT, controlado por aplicativo mobile (também criado neste trabalho), capaz de trancar e destrancar uma porta residencial, utilizando a própria chave e o mecanismo já existente na porta. O dispositivo deve ser capaz de receber, via internet ou bluetooth, comandos emitidos pelo aplicativo e reagir fisicamente, girando a chave acoplada. Além disso, o aplicativo deve possibilitar o recebimento de alertas de "porta aberta"e toques de campainha, compartilhamento de chaves virtuais e registro de atividades por usuários.

Palavras Chave:
Internet das Coisas, fechadura, porta, segurança, automação, domótica

Abstract

The main objective of the present work is to create an IoT device, controlled by a mobile application(also developed in this work), capable of locking and unlocking a residential door, using the own key and the existing mechanism inside the door. The device must be able to receive, via internet or bluetooth, commands sent by the application and physically react by rotating the coupled key. In addition, the application must enable "door open"alerts and ringtones, virtual key sharing, and user activity logging.

Keywords:
Internet of Things, lock, door, security, automation, domotic

Sumário

Lista de Figuras

Capítulo 1

Introdução

O mercado de Automação Residencial vem crescendo exponencialmente a cada ano, no Brasil e no mundo, tendo apresentado em 2016 um valor total superior a US$39 Bilhões, com potencial de alcançar US$81 Bilhões em 2023, segundo pesquisa realizada pela *Allied Market Research*[1]. Isso se deve a uma maior preocupação da sociedade com fatores como consciência de gastos energéticos, preocupação com segurança pessoal e residencial, e os próprios avanços tecnológicos.

O objetivo deste projeto é implementar uma solução voltada para segurança residencial, com foco em praticidade do usuário: um sistema de fechadura inteligente capaz de se comunicar com dispositivos móveis a fim de proporcionar mais segurança, conforto e comodidade, atendendo aos padrões das casas brasileiras.

A ideia é a criação de uma fechadura modular, que pode ser acoplada em um mecanismo de tranca convencional pelo lado de dentro da casa, mantendo a estética e a funcionalidade original pelo ambiente externo, porém oferecendo alternativas práticas e seguras para trancar e destrancar a porta sem o uso da chave, mesmo à distancia. Também será integrado um sistema de comunicação que permitirá ao proprietário saber quando alguém está tocando sua campainha, mesmo longe de casa.

Capítulo 2

Contextualização

Consumidores estão cada vez mais aderindo soluções automatizadas e aplicando-as em seu dia a dia, principalmente em nichos de iluminação, segurança, climatização e entretenimento. Os motivos podem variar entre economia de tempo, praticidade, necessidade de gerenciamento, entusiasmo tecnológico e até mesmo luxo.
Hoje o Brasil apresenta mais de 300 mil residencias equipadas com tecnologia voltada à automação residencial, incluindo-o no grupo de países que contribuem para um aumento anual de 11,35% na adesão à automação residencial global[2]. Tendo isso em mente, e a preocupação constante dos brasileiros com segurança, produtos como fechaduras inteligentes têm potencial de apresentar grande sucesso em território nacional.
Infelizmente a maioria dos produtos vendidos no exterior não chegam ao conhecimento geral da população, não são compatíveis com padrões de porta brasileiros ou apresentam preços excessivamente elevados.

2.1 Canvas

Com foco na possibilidade de tornar deste projeto um produto, foi criado um Canvas[3] para organizar um plano de modelo de negócios:

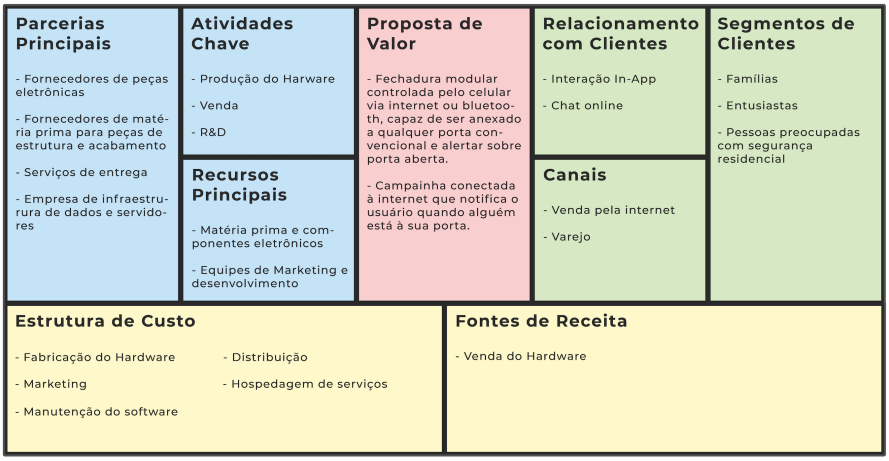

Figure 2.1: *Business Model Canvas*

2.2 Soluções já existentes

Existem muitos modelos de fechaduras eletrônicas à venda no mundo, capazes de oferecer um estilo de vida mais prático, normalmente excluindo a necessidade de carregar chaves nos bolsos e mochilas. Porém, a maioria destes produtos atua de forma isolada, ou seja, não são capazes de se comunicar com outros dispositivos ou com a internet, deixando de lado funcionalidades como controle a distância.
Quando observamos o segmento de fechaduras eletrônicas conectadas o número de marcas que oferecem tais produtos é reduzido, mas possuem um grande potencial. Empresas como *Nest*[4] e *August*[5] produzem soluções que podem ser integrados com diferentes softwares de automação residencial e assistentes pessoais como *Google Assistant*[6], *Alexa*[7] e *Siri*[8]. Essas opções proporcionam controle à distancia sobre portas e campainhas, além de manter o proprietário informado em tempo real sobre o estado da entrada de sua casa.
Este projeto tem como inspiração um produto de muito sucesso chamado **"Smart Lock Pro"**[9], da marca *August*. Ele propõe um módulo de fácil instalação no lado interior da porta, oferecendo em conjunto um aplicativo para controle de funcionalidades, entre elas: trancamento/destrancamento remoto, avisos de segurança, destrancamento por proximidade e compartilhamento de chaves virtuais. Entretanto, é um produto de preço muito elevado que não é compatível com os padrões convencionais de porta dos brasileiros, restringindo-se apenas à portas com o mecanismo "deadbolt"[10].

Capítulo 3

Desenvolvimento

Este trabalho foi dividido em 4(quatro) principais projetos: Um hardware para agir como interface física capaz de girar uma chave na fechadura e captar a ação de uma campainha; Um hardware para estabelecer comunicação entre o dispositivo da fechadura e a internet; Um aplicativo para dispositivos Android para controlar o Hardware, criar chaves virtuais e receber notificações; e a configuração de servidores online para rotear mensagens, hospedar um banco de dados e realizar autenticação de usuários.

3.1 Hardware 1 - Fechadura

A fechadura foi implementada utilizando um ESP32 como microcontrolador principal, que já inclui oferece conectividade Bluetooth Low Energy[11] nativa. A partir de mensagens Bluetooth, associadas a um serviço e características próprias, o dispositivo é capaz de receber comandos para trancar e destrancar a porta, mesmo que não haja conexão com a internet. Porém, também há um módulo XBee[12][13] incluído neste sistema, que é responsável pela troca de mensagens com um hub conectado à internet, permitindo comandos remotos de qualquer lugar do mundo e a alimentação de uma base de dados com o atual estado da fechadura. Ligado ao microcontrolador por meio de um driver ULN2003AN, temos um motor de passo 28131J-48[14], programado para executar ciclos no sentido horário e anti-horário de acordo com comandos recebidos. Para uso final, um sistema de duas engrenagens torna possível que o motor gire a chave anexada à porta, desde que essa esteja acoplada na engrenagem correta. Há também dois botões, um que executa a ação direta de trancar/destrancar a porta, e outro que age como uma campainha, enviando dados para a internet para indicar que a campainha foi tocada (este segundo botão foi anexado ao dispositivo apenas para testes e demonstração, para melhor aproveitar os recursos disponíveis).

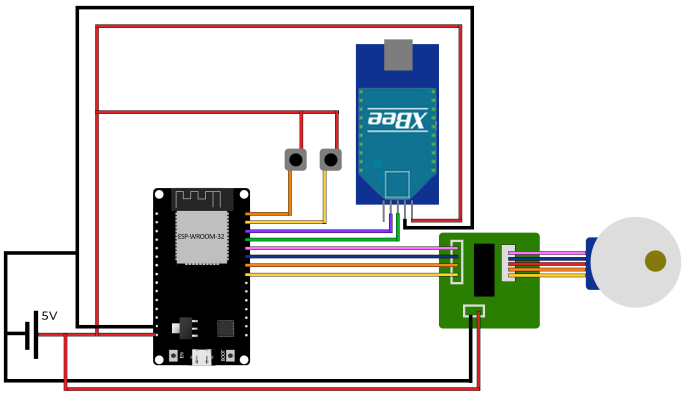

Figure 3.1: *Esquemático Fechadura*

3.1.1 Bibliotecas

As bibliotecas utilizadas neste projeto foram:

- **Stepper.h**[15]**:** Biblioteca para manipulação do motor de passo.
- **BLEDevice.h/BLEServer.h/BLEUtils.h/BLE2902.h**[16]**:** Bibliotecas para configuração de serviços Bluetooth Low Energy.

3.1.2 Motor de Passo

O motor de passo 28BYJ-48 é um motor de 5VCC, 4 fases, e caixa de redução de razão 1/64, ou seja, a velocidade final de seu eixo externo é 64 vezes menor que sua velocidade básica, e seu numero total de passos para completar uma volta passa de 32 para 2048(64x32). Por consequência, O motor possui uma excelente precisão e torque satisfatório.

3.1.3 Engrenagens

As engrenagens que tornam possível a ação de girar a chave acoplada a uma porta foram desenhadas e projetadas de modo que uma fosse encaixada no eixo do motor de passo, e a outra fosse capaz de segurar uma chave convencional, mantendo uma proporção de raio *1x1*.

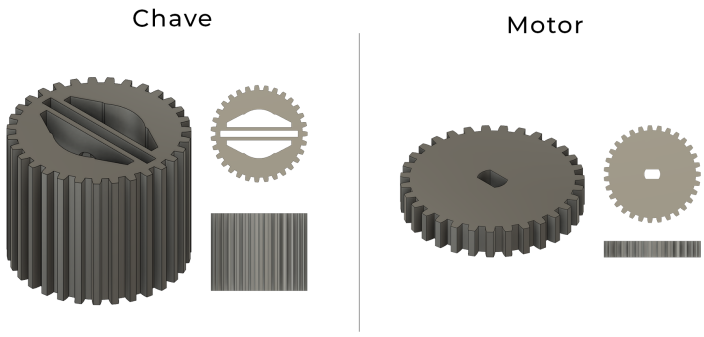

Figure 3.2: *Engrenagens projetadas*

3.1.4 Rede ZigBee

Dois módulos *XBee S1* foram utilizados para este projeto. Um deles foi programado como como *Coordenador*, enquanto o outro foi programado como *End Point*, ambos compartilhando o mesmo *PAN ID*[17]. Isso permite que uma rede estrela seja criada, onde qualquer nó programado como *End Point* com o determinado *PAN ID* seja capaz de se comunicar com o nó *Coordenador* e vice-versa. As mensagens são estruturadas no formato **[id_do_dispositivo]/[comando]**, permitindo que o mesmo Hub Wi-Fi coordene diversos *end points*, que simplesmente descartam mensagens destinadas a outros dispositivos que não o próprio destinatário

3.2 Hardware 2 - Hub Wi-Fi

O Hub Wi-Fi foi desenvolvido com o intuito de economizar a energia gasta pelo dispositivo da fechadura, uma vez que a manutenção de uma conexão Wi-Fi contínua pode prejudicar a vida útil de pilhas e/ou baterias. Para isso, o Hub assume a responsabilidade de efetuar a comunicação com a internet enquanto ligado a uma fonte de energia contínua (tomada), e transmitindo/recebendo comandos da fechadura via ZigBee[13], um protocolo de comunicação sem fio de baixo consumo, muito utilizado em soluções de IoT. O Hub foi implementado utilizando um NodeMCU (ESP8266) como microcontrolador principal, que já inclui compatibilidade Wi-Fi nativa. Ele estabelece conexão com um servidor que atua como Broker MQTT[18] e um servidor Google Firebase. As mensagens recebidas por MQTT são repassadas para a fechadura para que essa realize a ação, enquanto status enviados pela fechadura e toques de campainha são transmitidos à RealtimeDatabase.

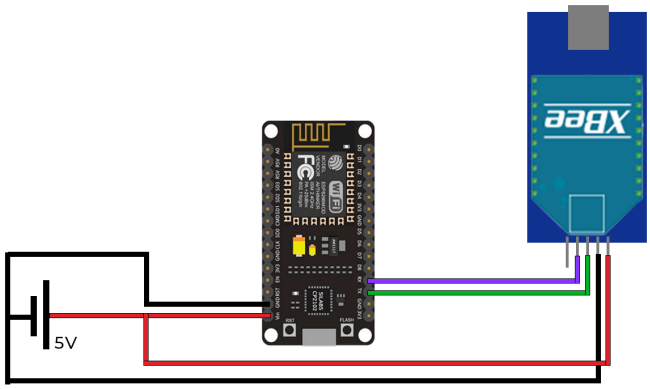

Figure 3.3: *Esquemático Hub Wi-Fi*

3.2.1 Bibliotecas

As bibliotecas utilizadas neste projeto foram:

- **ESP8266WiFi.h**[19]**:** Principal biblioteca para o microcontrolador ESP8266. Utilizada para estabelecer conexão Wi-Fi com a internet e controlar o WatchDog da placa.

- **PubSubClient.h**[20]**:** Utilizada para estabelecer conexão com o *MQTTBroker* e criar a função de callback para recebimento de mensagens MQTT.
- **FirebaseArduino.h**[21]**:** Biblioteca utilizada para estabelecer comunicação com a base de dados e atualizar os dados de status do dispositivo.

3.3 Aplicativo

O Aplicativo foi desenvolvido em Java, com foco no sistema Android. Ele estabelece conexão aos mesmos MQTT Broker e Firebase Services que o Hub Wi-fi, porém ele envia comandos ao Hardware em forma de mensagens MQTT e agir como receptor de mensagens do Firebase Cloud Messaging (as mensagens recebidas são convertidas em notificações, mesmo quando o aplicativo não está executando em background). Para ter acesso às funcionalidades do aplicativo, o usuário deve fazer um cadastro, o qual permitirá acessar a lista de dispositivos e chaves virtuais. Chaves virtuais podem ser criadas para conceder acesso limitado a demais usuários registrados, permitindo-os controlar a fechadura via internet em horários pré-estabelecidos pelo usuário.
Para permitir controle local da fechadura, mesmo sem acesso à internet, o aplicativo é capaz de buscar e se conectar com a fechadura via Bluetooth, e enviando comandos de trancar/destrancar a fechadura ao escrever em uma característica atrelada ao serviço BLE[22] definido pelo dispositivo.

3.3.1 Dependências

As dependências do projeto do aplicativo são:

- **Paho**[23]**:** Biblioteca para possibilitar que o aplicativo publique e receba mensagens MQTT.
- **Firebase SDK**[24]**:** Kit de desenvolvimento para utilização dos serviços Firebase.
- **BLE**[25]**:** Bibliotecas e serviços para integração com dispositivos Bluetooth 4.0 ou superior.

3.3.2 Screenshots

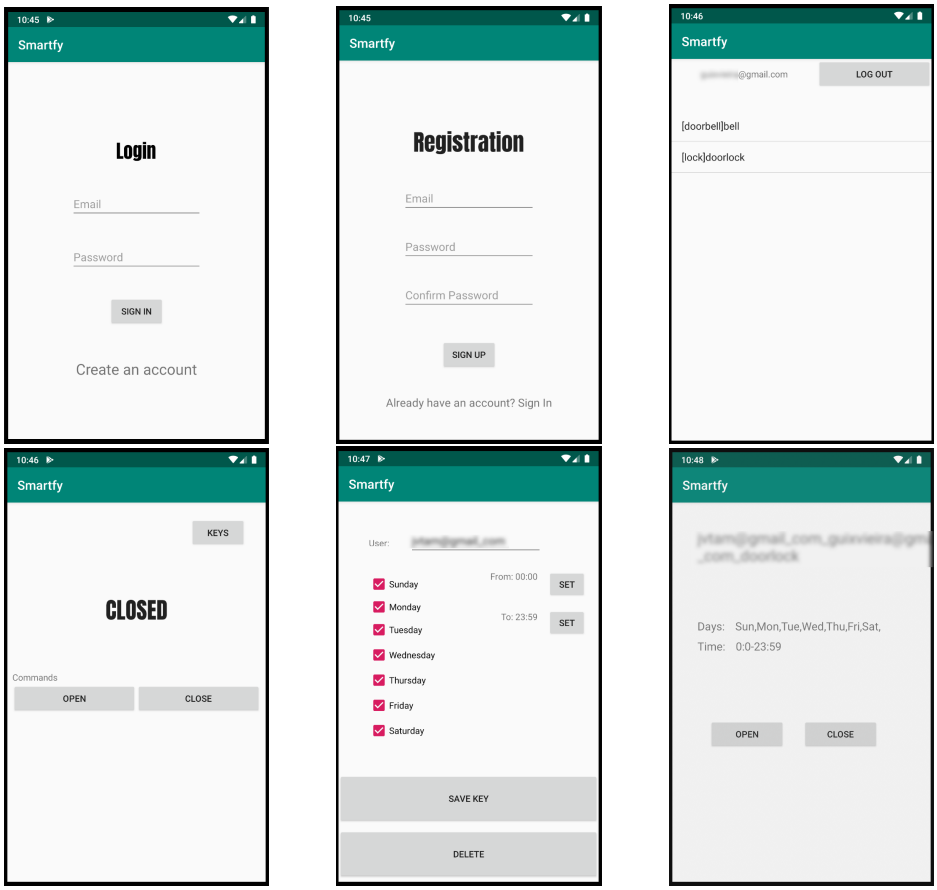

Figure 3.4: *Screenshots de Atividades do Aplicativo*

3.4 Servidores e Serviços

3.4.1 MQTT

Para transmissão de comandos do aplicativo ao Hardware, foi utilizado o protocolo MQTT para envio de de mensagens de um aparelho a outro utilizando TCP-IP. Cada mensagem é vinculada a um tópico, definido no envio da mensagem, com o objetivo de oferecer contexto à mensagem enviada, de forma que seja possível se inscrever para mensagens de determinado tópico.

Para utilizar a internet como canal para troca de mensagens MQTT, foi criado um Broker hospedado pela *CloudMQTT*[26]. O serviço de Broker online permite que as mensagens possuam um canal sempre conectado à rede mundial de computadores, e protegido por chaves de autenticação próprias.

3.4.2 Firebase

Diversos serviços da Google Firebase[27] foram utilizados, principalmente com o objetivo de autenticar usuários, administrar chaves virtuais e enviar notificações ao aplicativo a partir de ações no Hardware (mesmo quando o aplicativo não esteja rodando em background). Para isso uma Firebase RealtimeDatabase[28] grava os estados do hardware e os dados de chaves compartilhadas, o Firebase Authentication[29] foi configurado para registrar usuários a partir de e-mails, e um back-end foi programado para acionar o Firebase Cloud Messaging[30] sempre que o status da campainha gravado na base de dados é alterada pelo hardware.

Firebase RealtimeDatabase

A RealtimeDatabase possui estrutura similar a um documento JSON, e foi estruturada segundo o esquema abaixo:

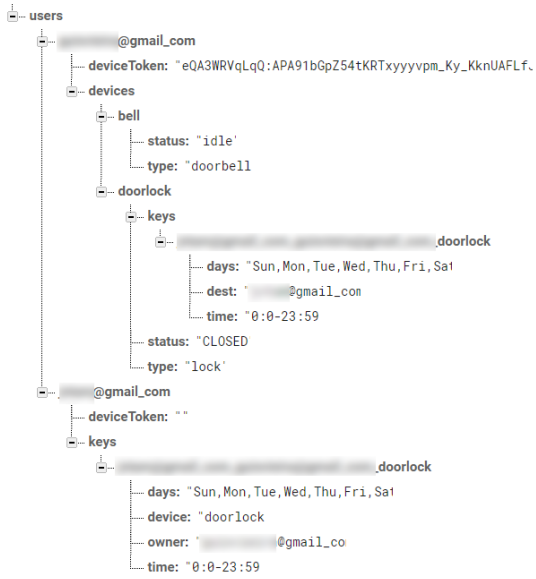

Figure 3.5: *Estrutura da base de dados*

Logo abaixo da raiz, está a coleção de usuários, que são identificados por seus respectivos e-mails. Abaixo de cada usuário, é armazenado um *DeviceToken* que identifica o aparelho celular para o qual devemos mandar notificações, e as coleções de dispositivos e/ou chaves. Cada dispositivo possui um tipo e seu respectivo estado, podendo também, no cado de uma fechadura, armazenar a lista de suas chaves virtuais compartilhadas. As chaves são compostas pelos dias da semana em que pode ser utilizada, o espaço de tempo que podem ser utilizadas em cada dia (horário_de_início-horário_de_término), e o destinatário da chave virtual. Elas são armazenadas em 2(duas) posições na base de dados: na lista de chaves do próprio dispositivo, e na lista de chaves do destinatário (onde são necessários o identificador do nome do aparelho a qual a chave pertence, e o respectivo dono do aparelho).

Firebase Authentication

O Firebase Authentication é o serviço responsável pelo registro e autenticação de usuários. Ele garante que não hajam registros a partir de um mesmo endereço de e-mail, salva as senhas de acesso de maneira segura, e oferece um método prático de login para o aplicativo.

Firebase Cloud Messaging

O Firebase Cloud Messaging é um serviço que foi adicionado ao aplicativo permitindo o recebimento de mensagens dos servidores do Google Firebase mesmo quando o Aplicativo se encontra fora de execução. As mensagens carregam os dados de *título* e *corpo* de notificações ao usuário.

Firebase Functions

Firebase Functions[31] é um meio de integração para os diversos serviços do Google Firebase. Este serviço pode ser programado em TypeScript e atua como back-end dos servidores do projeto.
A principal função configurada exerce o papel de criar e enviar uma mensagem sempre que o dado */status* de um dispositivo do tipo *doorbell* é atualizado para *ring*, e automaticamente o modifica para *idle* após a mensagem ser enviada com sucesso. Este evento é iniciado pela função de callback *onChange*, que entra em ação sempre que um dado da base de dados é alterado.

Capítulo 4

Resultados

Como resultado há um aplicativo para dispositivos android, servidores online configurados, e dois protótipos físicos funcionais, sendo eles um Hub Wi-Fi e uma Fechadura capazes de comunicar entre si. Com este sistema é possível controlar o motor de passo do dispositivo via internet e/ou bluetooth através da interface do aplicativo, e enviar notificações ao celular onde o aplicativo foi instalado, sempre que o botão de campainha do dispositivo for acionado. O esquema abaixo ilustra o sistema e seu funcionamento:

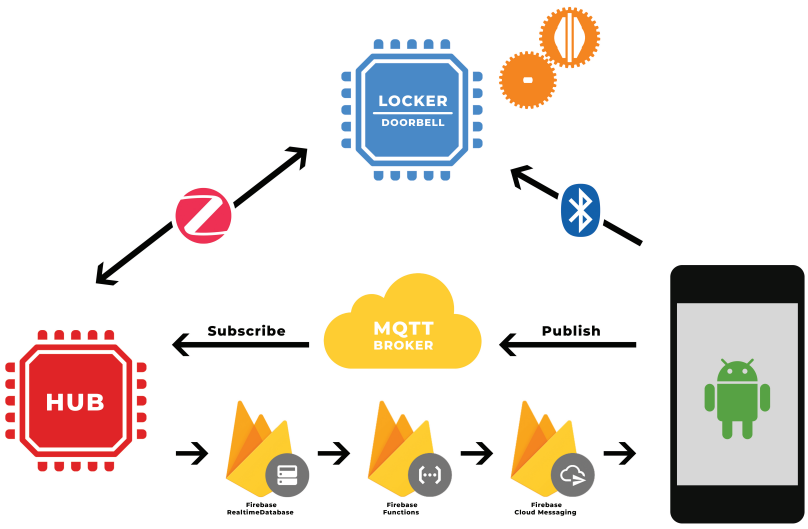

Figure 4.1: *Esquema do Sistema*

Ao entrar na tela de controle da fechadura no aplicativo, o smartphone procura o exato dispositivo através do Bluetooth, e estabelece uma conexão caso o encontre. Os comandos de trancar e destrancar a fechadura podem ser acionados pelo aplicativo, mandando um comando via Bluetooth ou via internet (caso o aparelho não esteja conectado por Bluetooth). Pela internet o aplicativo envia o comando para um Broker MQTT, de onde o Hub Wi-Fi recebe a mensagem e a direciona via ZigBee para a fechadura. Ao receber um comando por Bluetooth

ou ZigBee, o dispositivo da fechadura aciona o motor de passo até que este complete um giro completo, girando a chave na porta através das engrenagens impressas por uma impressora 3D. Ao completar a ação, é enviado via ZigBee uma atualização de status, que é registrada na base de dados pelo Hub Wi-Fi. O toque de campainha, assim como o estado da fechadura, é enviado ao Hub e então registrado na base de dados, porém tal mudança é detectada pelo backend do servidor e uma notificação é enviada para o dono da campainha, mesmo se o aplicativo não estiver ativo ou rodando em background.
Um vídeo demonstrando o funcionamento do protótipo pode ser encontrado em:
https://youtu.be/Ds9w-vlipao

Capítulo 5

Conclusão

Os principais desafios enfrentados durante a implementação deste projeto incluem o domínio de diversas tecnologias, como protocolos de comunicação sem fio, eletrônica e desenvolvimento mobile, assim como a criação de interfaces gráficas para usuários e o design das engrenagens. No entanto, o resultado se mostrou funcional e recompensador, proporcionando um grande aprendizado acerca de múltiplas tecnologias e áreas de conhecimento complementares à Ciência da Computação a partir do desenvolvimento de uma solução para um problema do mundo real e físico.
Contudo, o projeto ainda pode ser melhorado a fim de atingir seu objetivo final. Para tornar o produto final mais útil e desejável, seria importante a implementação de recursos como "reconhecer se a porta está aberta ou fechada"e registro e visualização de um histórico de ações. Além disso, é necessário revisar o modelo mecânico das engrenagens e o modelo do motor de passo utilizado, a fim de garantir que o dispositivo tenha força suficiente para trancar e destrancar qualquer fechadura.

Referências Bibliográficas

[1] Preksha Verma. Home automation market by application (lighting, safety & security, hvac, entertainment, and others), type (luxury, diy, managed, and mainstream), and technology (wired and wireless) - global opportunity analysis and industry forecast, 2017-2023. *Allied Market Research*, 2017.

[2] DINO. Mercado de automação residencial segue com boas perspectivas no país. *Info Money*, 2018.

[3] Adolfo Felipe da Silva. O que é business model canvas e como fazer um? *Guia Empreendedor*, 2019.

[4] Nest. https://nest.com.

[5] August. https://august.com.

[6] Google assistant. https://assistant.google.com.

[7] Amazon alexa. https://developer.amazon.com/pt-br/alexa.

[8] Siri. https://www.apple.com/br/siri.

[9] Smart lock pro. https://august.com/products/august-smart-lock-pro-connect.

[10] What is deadbolt lock and why it is important. https://www.reviewsworthy.net/home-security/what-is-deadbolt-lock-and-its-benefits.

[11] Bluetooth Low Energy. https://web.archive.org/web/20170310111443/https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/low-energy.

[12] XBee. https://www.digi.com/resources/documentation/digidocs/pdfs/90000982.pdf.

[13] ZigBee. https://zigbee.org/.

[14] Gustavo Murta. Guia completo do motor de passo 28byj-48 + driver uln2003. *Eletrogate*, 2019.

[15] Stepper Library. https://www.arduino.cc/en/Reference/Stepper.

[16] ESP32 BLE for Arduino. https://github.com/nkolban/ESP32$_BLE_Arduino$.

[17] PAN ID. https://www.digi.com/resources/documentation/Digidocs/90002002/Concepts/c$_z b_p an_i d.htm?TocPa$
$Zigbee\%20networks\%7CZigbee\%20networking\%20concepts\%7CPAN\%20ID\%7C\_\_\_\_0$.

[18] MQTT. http://mqtt.org.

[19] ESP8266WiFi library. https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html.

[20] Arduino Client for MQTT. https://pubsubclient.knolleary.net/api.html.

[21] firebase arduino. https://firebase-arduino.readthedocs.io/en/latest/.

[22] GATT. https://www.bluetooth.com/specifications/gatt/.

[23] Eclipse Paho Android Service. https://www.eclipse.org/paho/clients/android/.

[24] Firebase Android SDK. https://firebase.google.com/docs/reference/android/packages.

[25] Android BLE. https://developer.android.com/guide/topics/connectivity/bluetooth-le.

[26] CloudMQTT. https://www.cloudmqtt.com/docs/index.html.

[27] Google Firebase. https://firebase.google.com/docs.

[28] Firebase Realtime Database. https://firebase.google.com/docs/database.

[29] Firebase Realtime Authentication. https://firebase.google.com/docs/auth.

[30] Firebase Cloud Messaging. https://firebase.google.com/docs/cloud-messaging.

[31] Cloud Functions para Firebase. https://firebase.google.com/docs/functions.