**IOT BASED SMART DOOR LOCK**

**A PROJECT REPORT**

***Submitted by***

**Divya (21BCS11530)**

**Nandita (21BCS6390)**

**Oorja Tiwari (21BCS11557)**

**Satyam Kumar Singh (21BCS11016)**

**Tanya Chauhan (21BCS11218)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINNERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**Chandigarh University**

MAY 2022

****

**BONAFIDE CERTIFICATE**

Certified that this project report “**IOT BASED SMART DOOR LOCK”** is the bonafide work of **“Divya (21BCS11530), Nandita (21BCS6390), Oorja Tiwari (21BCS11557), Satyam Kumar Singh (21BCS11016), Tanya Chauhan (21BCS11218)”** who carried out the project work under my supervision.

**Signature Signature**

Mr. Vikas Wasson Head Of Department

**SUPERVISOR**

(HOD AU-1)

**External examiner Internal examiner**

# Acknowledgments

*We thank our IP supervisor HOD AU-1 Vikas Wasson and our IP mentor Dr. Ritu for guiding us throughout the project, and for their great flexibility in agreeing to make this project an actual IP.*

## **Table of Contents**

**Chapter 1: Introduction**

(Including Identification of client & need, Relevant contemporary issues, Problem Identification, Task Identification, Timeline, and organization of the report)

**Chapter 2: Literature survey and comparison**

Timeline of the reported problem as investigated throughout the world, bibliometric analysis, proposed solutions by different researchers, summary linking literature review with the project, Problem Definition, Goals, and Objectives.

**Chapter 3: Design flow/Process**

Concept Generation, Evaluation & Selection of Specifications/Features, Design Constraints– Regulations, Economic, Environmental, Health, manufacturability, Safety, Professional, Ethical, Social & Political Issues considered in the design, Analysis, and Feature finalization subject to constraints, Design Flow (at least 2 alternative designs to make the project), Best Design selection (supported with comparison and reason) and Implementation plan ((Flowchart /algorithm/ detailed block diagram).

**Chapter 4 Results and conclusion**

Implementation of design using Modern Engineering tools in analysis, design drawings/schematics/ solid models, report preparation, project management, and communication, Testing/characterization/interpretation/data validation.

**Chapter 5: Validation and future work**

deviation from expected results and way ahead

References

Appendix

User manual (Complete step by step instructions along with pictures necessary to run the project)

Achievements

**CHAPTER 1**

**INTRODUCTION**

Using physical keys to lock or unlock the door is the most natural way and everyone is acquainted with it. The physical key is a well-tested and well-known technology, but it also has its flaws. There can only be one unique key for a lock. For different locks, you have different keys. Furthermore, carrying a large number of keys is a burden and increases the chance of keys getting stolen, misplaced, or lost. Our goal is to design a solution for secure access control that can replace physical keys for accessing the door. We propose a solution using digital keys on smartphones providing wireless and automatic unlocking via Bluetooth. The design will allow easy implementation and distribution of keys and the device will work autonomously. This will enhance security and will eliminate the need of carrying physical keys.

## **Problem Definition**

The security aspect is the highest concern of IoT-connected entities. The data can be personal, enterprise or consumer. To reach an acceptable implementation for the smart door lock (SDL), security should be taken as a major challenge. We can summarize the problems into different questions

1. How do we set up high and strong authentication between the user point entity(e.g Smartphone) and the API and will this property provide strong privacy guarantees?
2. How do we generate an access token for the user that has the privilege to unlock the door and how do we secure this token from being exposed?
3. Which connection protocols can be used in the product and offers the ability to authenticate, and access control? Does the local WiFi network fulfill the security obligation?
4. What kind of microcontroller would satisfy the aims of the product by offering a secure IoT system? 5. Which IoT architecture would fit the aim of SDL?

## **Objective**

1. To have a review of the existing locking mechanisms with their methodologies and drawbacks to have a proper problem formulation.
2. To design an advanced smart locking mechanism using ESP 32 board.
3. To compare the features of existing locking systems with our smart lock.
4. Adding essential features to our product to make it different and better than the existing ones.
5. Creating an app as the endpoint and securing the unlock feature.

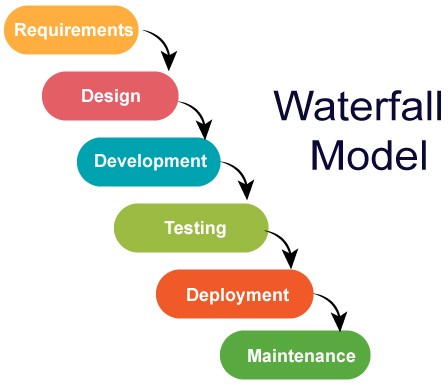
## **Methodology**

#### Introduction

The main goal of this project was to design and build a door lock system that allows users to unlock a door via face recognition, through a camera implanted on the door. In this methods chapter, we will discuss how we detailed the process of implementing this mechanism. We started our research by confirming the need of potential users for such a system and then followed a modified version of the Software Development Life Cycle (SDLC) approach to design and build a door lock system.

#### Why We are Choosing this Strategy

SDLC is a strategy that is used to ensure that products that are developed, are optimized for their users, based on a set of requirements. It is a very common strategy in software development projects. SDLC involves several steps shown in Figure 2.1: Requirements Gathering, Analysis, System Design, Object Design, Testing, and Implementation. Each step ensures that the developers are ready for the next one and the approach tries to minimize the development time by having predefined expectations for each step. By following this process, the developers ensure that their finished product meets and addresses the needs of their users.



*Figure 2.1 Software Development Life Cycle Phases (WATERFALL MODEL)*

#### 

#### Requirements Gathering

The objective of this step is to obtain information about the potential need for such a door lock system as described in Section 2.1. Before we designed a solution to an acknowledged problem, we needed to know the specifics of user behaviors, which would especially help in designing the

**CHAPTER 2**

**Literature Survey and Comparison**

In this section, we begin with introducing the Internet of Things, continuing it further with the interest, people have shown toward IoT devices, and how door access control systems have adhered to or missed the requirement of people. We show examples of some products or prototypes designed to offer convenient door lock systems.

#### Internet of Things

In the past few years, the Internet of Things (IoT) has grabbed the attention of both the worlds i.e academic and industrial worlds. IoT is a concept describing an idea in which day-to-day objects are connected to the Internet, and are advanced to communicate with other devices. These interconnected devices are generally referred to as smart devices. Each of these smart devices is a physical component with specific functionality, having a unique identifier, from a common IoT market perspective, the need for IoT devices will soon reach about 30 billion by 2023. Smart devices are very common, and useful as we get to observe them in regular life such as smartwatches. Smartwatches such as the Apple Watch and Fitbit joined the market not very long ago. Many devices have also entered our homes to improve their lifestyle, with products such as Siri, a voice assistant, which users can control with a voice command, provide weather, and control other IoT devices. A less common example, but helpful, can be seen in Barcelona, where the city has implemented some IoT initiatives that have helped enhance smart parking. Exponential growth is expected for the IoT market and an increasing number of smart devices show the inclination of the industry toward offering these interconnected solutions to increase convenience, but also the high adoption rate of the population.

#### Access Control Credentials and Readers

We have researched a few access control methods that are currently being used around the world. In this section, we will explain three of the most prevalent types, focusing on their advantages and disadvantages.

#### Keypads

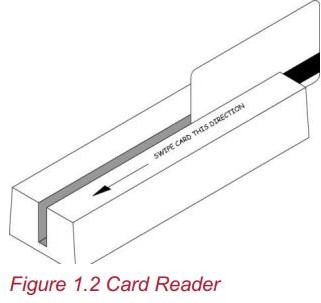


Figure 1.1

The most basic types of ID readers are keypads, based on simple twelve-digit keypads, containing the numbers 0-9 and a \* and # sign, shown in figure 1.1. These types of access control locks are very simple and cheap, and over time they have advanced to include some more secure functionality, like scrambling the LED number keys on the keyboard so that others who can detect the gesture and movement wouldn’t be able to use it. The main disadvantage include people’s need to share the code,

like with package or food delivery or friends. If people share this code, you can’t know how many people have access or who accessed the keypad at what time.

#### Access Cards and Card Readers



There are several different card reader types. Some common ones include magnetic stripes, barcodes, etc. Magnetic stripe cards have a magnetic band laminated to the back of the card. The card can contain a code, which serves for identification, the person’s name, and other useful data, although only the ID code is commonly encoded. ID cards are effective, simple to build, and cheap, but they are very easy to duplicate. Other types of ID cards work in a very similar way to magnetic stripe cards.

#### Biometric readers



Any device that reads the identity of a person by comparing some attribute of their physiological being or behavioral traits against a sample database is called a Biometric Reader. Some common biometric readers include facial recognition, fingerprint, and iris recognition. Advantages of biometric readers include increased security and the convenience of not carrying a key or card. Disadvantages include the need to enroll users in a database by providing sufficient samples of the user’s face (for facial recognition) or fingerprint (for fingerprint scanners). Some modern-day inconveniences include users who try to fool these systems. A common example is how facial recognition systems can be fooled by high-quality photos of a user’s face, even without the user being present.

#### Smart door locks

The idea behind the modern-day smart locks is to unify the functionality provided by convenient and secure access control methods with the convenience of internet connectivity and remote control. There are a few products on the market that offer these systems and we will talk about two of them, once again focusing on the features they offer and potential disadvantages, as these products will serve as

inspiration for our actual product implementation.

#### August Door Locks

PCMag has rated the August door locks as one of the best smart door lock systems. They offer a mobile application, where you can check the history of the activity, like who locked or unlocked it at different times, as shown in Figure 1.4. From the mobile app, you can also grant unlimited digital keys to different friends. The hardware installed on the door, shown in figure 1.5 can also indicate whether the door is completely closed or not. Different versions with a range of prices are offered. Some of the higher-end versions include a camera for users to see who is at the front door and whether they want to unlock the door for them. Whoever you share the Mount 4 code with needs to have the mobile application for the sensor in the lock to detect them when they are close.



#### Kwikset Kevo Locks

Kevo locks, shown in figure 1.6, are another brand of door lock highly rated by PCMag. They offer similar features, but to unlock the door they use a digital keypad. Each user has its access code, so the lock’s accompanying mobile application can give a history of the lock’s activity. For friends or

temporary users needing to access the house, you can easily send a new access code and assign the period for the temporary user to access the lock. Since every user needs to use an access code to unlock the door, you can’t unlock it via proximity to your phone.

Our background research indicated that smart locks are becoming a trend, but there is little research into:

* 1. What is the most common nuisance of all the traditional door lock systems?
  2. Whether people are ready to switch to a digital smart lock, completely controlled by their smartphones and with authentication based on facial recognition?

#### 

#### Survey

Most of the motivation behind this project came from our own experience with traditional door locks and their inadaptability to the changing habits of modern times. Our first step involved discovering if others share the same experience with their door locks and inquiring about other potential issues that we had missed. The best way to get a quick idea of a potential problem was to survey people of our age about their habits. This provided some quick feedback which helped with the initial system and mobile application design. It also helped us in choosing a development platform: smartphone vs. computer vs. tablet; iOS vs. Android; etc.

The results from the 123 responses to our initial survey were very useful. The full scope of the answers can be found in Appendix C. Here were some of the most important observations:

1. Around 90% of people needed to physically go to the door whenever they had a visitor
2. A smartphone was people’s preferred way to possibly unlock a door remotely
3. 50% of the people surveyed always or often find themselves with their hands full while trying to reach for the key to open a door
4. Around 85% of the people surveyed used an android device as their primary mobile device
5. Lock picking ( the practice of unlocking a lock by manipulating the components of the lock device without the original key) is a very common practice and one of the major drawbacks of a mechanical lock

#### Analysis

During analysis, developers aim to produce a model of the system that is correct, complete, consistent, and unambiguous. From the initial survey, we created user stories, use cases, and UI mockups that we then discussed with the team and project advisors. A complete set of user stories can be found in Appendix B. During the analysis phase, we ensured that there was no lack of understanding of particular needs or possible features of the system we were building. Furthermore, by prioritizing requirements, we needed to create a scope of the project that should consider what we could build in the available time. Our analysis included:

1. a set of user stories created from requirement gatherings that will later turn into features of the application
2. Use cases, describing the system functionality from the user’s point of view
3. UI Mockups, describing how each feature will look on a screen

#### 

#### CHAPTER 3

#### DESIGN FLOW / PROCESS

#### Design

This step of the Software Development Life Cycle process is concerned with the methodology behind creating an effective platform that is useful and appealing to our target population. This section starts with comparing a few software platforms and our reasoning for deciding to develop a mobile application for android, our reasoning for choosing ESP-32 as our onboard logical unit, and the methodology behind turning the system requirements into small building blocks.

#### Components

|  |  |
| --- | --- |
| Lock Socket |  |
| Door Lock Actuator |  |
| Wooden Board |  |
|  |  |
| BREADBOARD | Introduction | Breadboards for Beginners | Adafruit Learning System |

#### HARDWARE REQUIRED



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | RELAY | 2-Channel 12V Relay Module | ElectroPeak | | LEDs | Light-emitting diode - Wikipedia | | Wi-Fi | Wireless or Wi-Fi 2 WiFi Router, Eagle Eye Surveillance | ID: 15079338091 | | WINDOW 10 MACHINE | 21.5 Inch Hp Windows 10 Computer, Multi Purpose, Hard Drive Capacity: 1TB,  | ID: 11788770697 |   ESP-32 MODULE | SquadPixel Esp-32 Wifi, Bluetooth, Dual Core Chip Development Board (ESP -WROOM-32) : Amazon.in: Industrial & Scientific |

**WEBSITES USED**

|  |  |
| --- | --- |
| MIT App INVENTOR | MIT App Inventor - YouTube |
| Ubidots | IoT platform | Internet of Things | Ubidots |

**SOFTWARES USED**

|  |  |
| --- | --- |
| ARDUINO IDE |  |
| ANDROID STUDIO |  |

#### Platform Choice

**Mobile Websites vs. Mobile Applications**

Websites and mobile applications are the two main ways of delivering a digital platform to a target audience. Before we started to design our features based on the needs of the surveyed community, we needed to decide on the most effective and useful platform for the targeted users. Table 2.1 gives a summarized comparison between these two basic methods.

*Table 2-1 Comparison between a website and a mobile app*

|  |  |  |  |
| --- | --- | --- | --- |
| **Website** | | **Mobile App** | |
| + | **Easier to develop, websites are compatible across devices.** | - More variability in development, because of different phone systems, types, and operating system versions. | |
| + | **Easier to maintain.** | It- Difficult to maintain, because of the manufacturer’s constant system upgrades. | |
| + | Instantly available, by providing the URL. | - Need to download it from the respective application store first. | |
| + | Broader reach because they are instantly available and compatible across devices. | + **Personalization - If target users have personal information and usage, then an app offers a better experience, including push notifications** | |
| - **No push notifications** | | + **Push notifications for real-time user response and information** | |
| - **Less integration with specific device resources.** | | + **Easier access to device resources, like GPS or camera sensors.** | |
| - Relies on an active internet connection. | | + | Offline access. |

Table 2.1 offers a good summary of the most important advantages and disadvantages of each method, but we could only decide by factoring in our system interactions. For example, having interactive push notifications is necessary for our needs, so that the user can easily see who is at the door and open it with a single tap. Furthermore, easier access to device resources, like GPS and cameras, is integral to our needs. The bolded points in the table above are the ones that helped us ultimately choose between a website and a mobile application.

#### Platform Choice - iOS vs. Android

A very important development decision was whether to create an app for an iOS or Android platform. There are software solutions that provide the ability to write and maintain a single code base for both iOS and Android. React Native is such a tool. We have compiled a list of the pros and cons of using React.

*Table 2-2 A Comparison of React Native Pros and Cons*

|  |  |  |
| --- | --- | --- |
| **React Native Pros** | | **React Native Cons** |
| + | Mostly write a single code base | - Still need to maintain parts of the Objective-C (iOS) and Java (Android) codebase, as not everything translates properly |
| + | React uses JavaScript, which is fast and popular | - Lack of documentation, as it is a young platform |
| + | Uses Native modules whenever it can, which translates into a fast and seamless experience | - Lack of third-party community-built components, which cannot be guaranteed to work in the next OS releases |
| + | Great for quick UI design | - Still in development: Instability, compatibility issues, and errors |

Although we wanted both an Android and iOS app, as that would increase our product’s exposure and the speed of adoption, if it ever goes to market, React Native did not sound like a safe choice, considering especially our lack of experience with the tool and all the potential instabilities, particularly with third-party component integration. As we can’t develop our app for both platforms, we made sure to include a question on the survey about what platform each of our respondents uses. The mobile OS with the most votes was iOS and that is why iOS is our platform of choice.

#### Platform Choice – ESP-32 vs. Arduino

For our systems, a computational device is required. This device needed to fulfill a series of requirements, which we divided into three separate categories, Development, Network connectivity, and physical ports. Each of the following conditions has been defined based on the system that makes the features of our platform possible with the most convenient development environment.

The three development requirements that we looked into were as below:

* Ability to run a scripting language
* Extensive community support
* Availability of documentation

The development requirements were essential to successfully achieve the project goals in time and to create a platform that can be easily maintained. A device that uses a scripting language such as C++ will allow the integration of multiple subsystems that are required in our product. Having to work with cloud systems, a camera, and different types of motors makes a scripting language the most favorable choice. Because this device will be the controller for almost every component in our platform, good documentation and support will make the development significantly easier.

The three Network and Connectivity requirements that we looked into were as below:

* On-board WiFi connectivity
* On-board Bluetooth connectivity
* Ethernet connectivity

Because the system needed to connect to cloud services to process the authentication of a user and their device, as well as video streaming, network and wireless connectivity stand at the core of our product requirements.

The two I/O ports that we thought of as crucial were as below:

* A port for the camera connection
* Ports for motor connection

With the implementation of the camera to provide a live feed of the door view and motors to operate the lock and the door itself, the computing device needed to be able to operate all the hardware that makes these functionalities possible.

After determining the requirements for the computational device, market and specifications research was conducted. A few products from popular providers such as ESP-32and Arduino were taken into consideration. Table 2.3 offers a side-by-side comparison between the specifications of the most popular microcontrollers in the market.

*Table 2-3 A comparison between ESP-32and Arduino Uno*

|  |  |
| --- | --- |
| **ESP-32** | **Arduino Uno** |
|  |  |
| Stronger and quicker processor, multitasking available | Can run one code at a time, so can't multitask activities, slower speed |
| Built-in Ethernet port, Wi-Fi, and Bluetooth capability | No internet connectivity (requires shield or module) |
| OS can be switched easily, and a scripting language used (Python) | Bigger learning curve since it uses C/C++ language and needs outside sources |
| Audio output, Camera port, USB ports, and HDMI output included | Requires shields to implement most of the I/O devices |

|  |  |
| --- | --- |
| Requires installation of programs to get simple actions going | Easier to connect to analog sensors, motors, and other electronic components |

From the comparison of the features offered by the ESP-32and the Arduino boards, we determined that the ESP-32, unlike the Arduino, is more suitable for projects that involve several functionalities at the same time, need easy access to the internet, and need media accessibility. On one hand, the Arduino fits better in projects that require easy reads from sensors and have to do operations based on the data, while being able to communicate easily with a variety of motors or machine parts. However, this board does require extensions in the form of shields, to expand on features that are required to accomplish this project. Therefore, the ESP-32was the most suitable board for our system.

#### Platform Choice - Motion Sensor Ultrasonic vs PIR

To automate the operation of the door lock system a motion sensor is required. The sensor is used to detect any activity that will be happening in front of the door, like a person approaching it. Therefore, the sensor required has to be able to detect movements within a reasonably controlled area. The detection range needs to be limited to prevent the system from being triggered by background activity.

#### Ultrasonic Sensor

An ultrasonic sensor is generally used to measure the distance of an object using sound waves. The sensor consists of a transmitter and receiver pair. The distance can be measured by taking into account the elapsed time between the sound wave being generated and the sound wave bouncing back to the receiver.

#### PIR Sensor

Passive Infrared Sensors are made of pyroelectric sensors, which can detect levels of infrared radiation. Everything radiates some low level of radiation. However, the hotter the object is, the more radiation is emitted. The sensor in a motion detector is split into two halves to detect change between the two. The two halves work with each other by canceling out normal IR levels in the background. When one half detects more IR radiation than the other, the output will swing high or low, therefore detecting motion. The heat release of the human body is enough to trigger this sensor, making it one of the most commonly used in security and lighting appliances [15].

Table 2.4 provides a side-by-side comparison between the Ultrasonic and PIR sensors. By comparing the characteristics of each sensor type it was determined that the PIR sensor is more suitable for our product because of its higher accuracy in the types of environments it will be exposed to, as well as the wider detection angle.

*Table 2-4 A comparison between an ultrasonic and a PIR sensor*

|  |  |  |
| --- | --- | --- |
| **Type** | **Ultrasonic** | **PIR** |
| Image |  |  |
| Range | 2-400 cm | 2.2 m |
| Angle | 30 degrees | 90 degrees for 1 m/s  44 degrees 0.5 m/s |
| Type of Info | The time that can be converted to distance | HIGH, LOW |
| Price ($) | 3.95 | 12.62 |

#### Platform Choice - Actuator

The implementation of such a door lock system involved some mechanical engineering to make the door adaptable for people with disabilities and to offer the feature of fully opening and closing the door via commands from the mobile application, or just by approaching, i.e. hands- free. To achieve this functionality, we used an actuator. There are many different types of actuators and during the design phase of the project, we decided which one is the most appropriate for our needs. Table 2.5 offers an overview of the advantages and disadvantages of each type of actuator we considered: pneumatic, hydraulic, and electric actuators.

#### Pneumatic Actuators

Pneumatic Actuators consist of a piston inside a hollow cylinder. Pressure from an external compressor moves the piston inside. With increasing pressure, a linear force is created that makes the cylinder move along the axis of the piston. The piston is then returned to its original position by either a spring or fluid being supplied to the other side of the piston [16].

#### Hydraulic Actuators

Hydraulic actuators consist of a cylinder or fluid motor that uses hydraulic power to facilitate mechanical operation. These are similar in operation to pneumatic actuators, but an incompressible liquid from a pump moves the cylinder [16].

#### Electric Actuators

Electric actuators convert electrical energy into torque. Usually, their inside mechanism is constructed out of an electric motor that turns a lead screw. When the screw rotates, the nuts get driven along the threads. The nut moves depending on the direction the screw rotates [16].

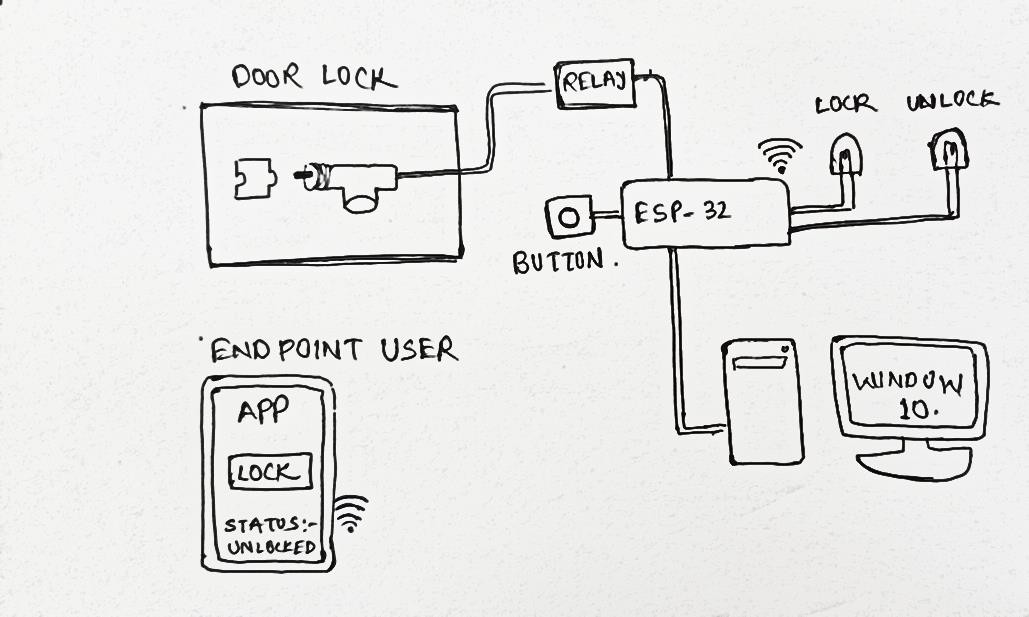
*Table 2-5 A comparison among a pneumatic, hydraulic, and electric actuators*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pneumatic** | | | **Hydraulic** | **Electric** |
| + | Simple and precise: a small pressure change can enable considerable forces (citation) | | + High force, 25 times greater than pneumatic cylinders of equal size | + Precise and quiet: they generate linear motion within +/- 0.000315 inches |
| + | From ½ to 8 inches, forces from 30-7500 | with lb | + Efficient, can hold force and torque constant without supplying more pressure | + Development: can be networked and reprogrammed quickly. Immediate feedback for diagnostics |
| + | Cost-effective ($50-  %150) | |  |  |
| - Low efficiency due to air | | | - Leakage: this type will leak fluid, leading to maintenance issues | - Difficulty holding a |
| pressure losses and air | | | locked position |
| compression. Additional | | |  |
| regulators and valves | | |  |
| maybe needed, raising | | |  |
| the costs. | | |  |
| - Maintenance: compressed air usually gets contaminated | | | - Complex: they need many companion parts, including fluid reservoirs, motors, pumps, release valves, etc. | - Expensive, price range from $150-$2000 |

Table 2.5 offers an overview of the main advantages and disadvantages that we considered when picking our actuator. Based on our initial studies of the system model, we chose an electric actuator, as we believed it to be the right one for our use, particularly because of its very easy development setup and connection to our electrical components of the door lock system.

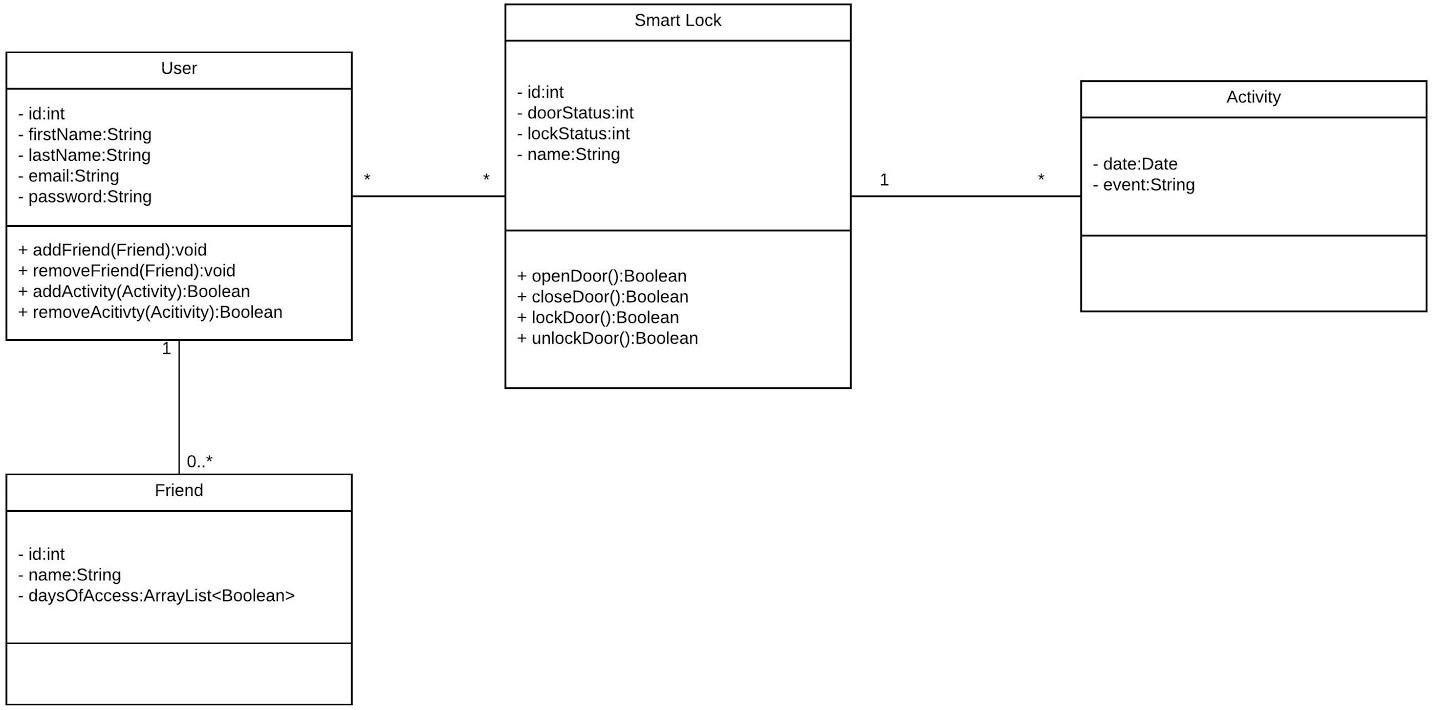
#### System and Object Design

During this step, the analysis of requirements, as explained in section 2.4, are turned into well-defined parts of the software. The goal of this step was to give the team all the resources needed to start implementing the system. The products of this step included a few different kinds of diagrams, which built upon the previously defined User Stories, Use Cases and UI Mockups.



#### UML Class Diagrams

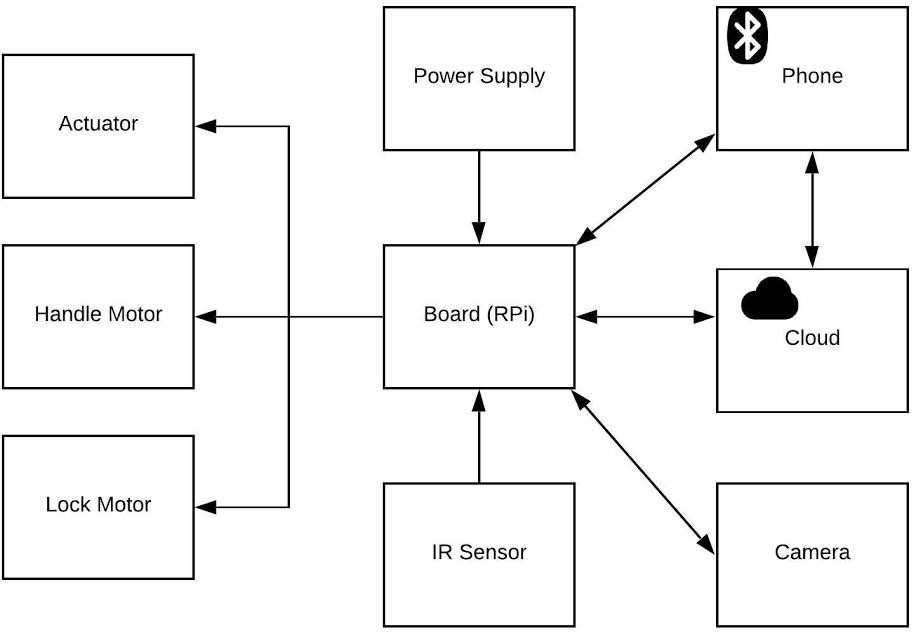
Class diagrams follow a clean object-oriented approach to software development, by clearly showing the classes of the system, their interrelationships, and the operations and attributes of the classes. We created a class diagram for the mobile application, which indicated the different objects of a Door Lock: Users, Administrators, Friends of each user, and a representation of the Door Lock itself. The high-level class diagram of our system is shown in Figure 2.2. It shows the most important objects in our mobile app: User, Friend, Smart Lock, and Activity. It also shows the relationships among these objects, e.g each User can add many Friends and each Smart Lock can have many Activities associated with it.



*Figure 2.2 High-level class diagram of the Smart Lock main objects*

#### Components Diagrams

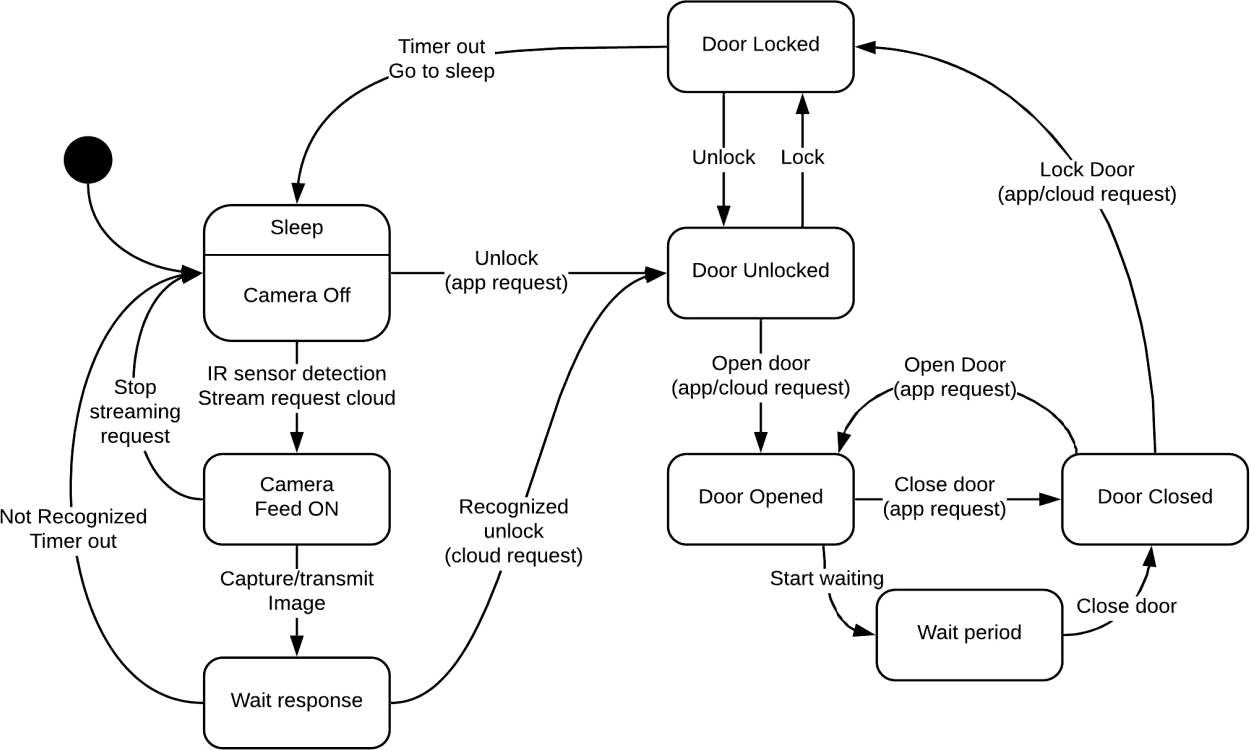
With the Components Diagrams, the team could see the interrelations among all the components of our system: Door Lock, the onboard logical unit mounted on the door; Mobile App, running on a mobile device connected to the Internet; the Cloud: an OS running our software, which is responsible for authenticating faces and sending commands from the Mobile App to the on-board unit. The initial ideation of such a diagram is shown in Figure 2.3.



*Figure 2.3 Components Diagram*

#### State Machine Diagrams

State Diagrams are particularly useful for complex components that can’t fully be represented in a Class Diagram. UML state machine diagrams depict the various states that an object may be in and the transitions between those states [18]. For our system, we are going to build a state machine diagram that indicates the different states of our onboard logical unit, which will run an algorithm that controls the state of the door based on commands from the mobile app or the presence of a person next to the door. The initial ideation of a state machine diagram is shown in Figure 2.4.



*Figure 2.4 State Machine Diagram for the Smart Lock system mounted on the door*

**Implementation and Testing**

In this section, we discuss the implementation of the IoT security monitoring system dan evaluation result. First, we integrate and implement the designed system as which shows the connection between ESP32 with Button Reset, PIR Sensor, green LED, red LED, Magnetic Sensor, Internal Touch Sensor, Alarm Buzzer, and Electric Strike door lock.

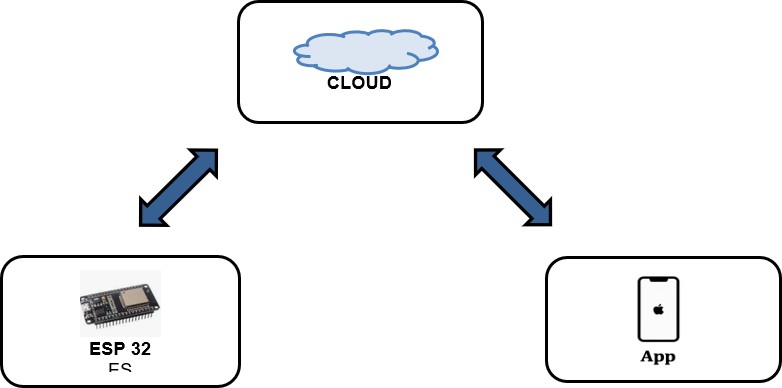
#### Implementation of the Design

The phase of development and implementation is conducted with an iterative strategy to construct a prototype that would match the specifications of the design. By breaking down the design into small chunks, we can develop and test it in repeated sequences. In each iteration, new features can be developed and tested until we have a fully functional system that fulfills the purpose of the thesis.

After Requirements Gathering, Analysis, and Design, there is the Implementation phase. In this step, the deliverable product was implemented. For the software part of our system, during this step, we translated the solution domain into source code, which included implementing the attributes and methods of each object and integrating all the objects such that they function as a single system. For the hardware part of the system, during this step, we implemented the state diagrams on an ESP 32 and connected the unit to the Cloud, and paired it with a mobile application, from which it will receive commands. For the mechanical engineering part of our system, we used an electric actuator, modified it according to our needs by using 3D printed parts, and installed it on a simple made-up door. This step took into account all previous design decisions we had made in precedent steps: tools like programming languages, platforms like mobile operating systems, and hardware choices like the onboard logical unit. Most of these design

choices should not be changing in this step, although minor improvements were needed. Whenever we had to change a considerable amount of the previously defined models, then we reiterated through all the steps of SDLC. In this section, we talk about the implementation of each of the three main components of our system, We conclude with a subsection on how we connected all these components. State Machine Diagram for the Smart Lock system mounted on door 16 An important part of this section is our discussion on what changed from the submitted proposal and the reason behind the change.

#### The three main components of the Door Lock System



**First Component - Smart Lock App**

Table 2.6 shows a set of features that we wanted the Smart Lock app to support. The “Must Have” column shows a set of features that we submitted to our project proposal. The column marked all the features that we deemed important at the beginning and that were very important for our application to have. The “Good to Have” features were also submitted in the proposal and the column marks all the features that we would implement if time permitted. We underline all the features that we were not able to implement due to time constraints.

#### Table 2.6 Smart Lock App features submitted in the proposal. Underlined features were not implemented

|  |  |
| --- | --- |
| Must-Have | Good to Have |
| Common UI/UX patterns, used in  similar apps (familiar application design, for high adaptability) |  |
| User personal account, including verification email and alternative  sign up with a Facebook or Google account. | Support for Admin level - admin users can add/remove other normal access users |
| Open/Close and Lock/Unlock door commands | Speaker and Microphone - for communication with different visitors while they are at the door |
| Livestream of the camera in the door |  |

|  |  |
| --- | --- |
| lock |  |
| Activity History - a list of events/commands accessible by the  user |  |
| Support for one Door Lock Device | Multiple “Smart Door Lock” devices - ability to add multiple door locks to the same profile, for different doors  inside the same residence or other residences |
| Friends - list of people with  moderate access, as defined by the user |  |
| Push Notifications - informational notifications about people being at  the door | Ability to open/close and lock/unlock doors right from the notification,  without opening the app |

Table 2.6 indicates that we’re able to implement all the “Must Have” features, while also providing support for the Admin level, a “Good To Have features”. For a detailed description of what each feature does and how to use it, see the Smart Lock User Manual in Appendix X. In the User Manual, we also provide screenshots of the app indicating each feature.

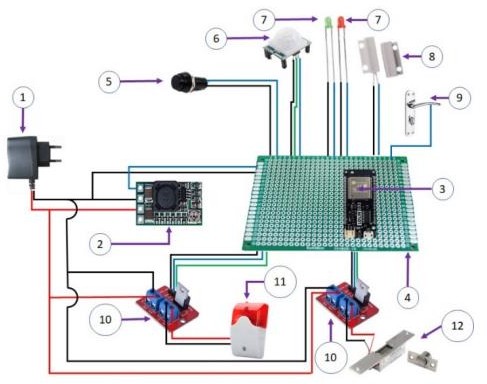
As our survey indicated (Appendix C), 65% of our surveyed population indicated that they use an iOS device, which is why we chose iOS as our mobile platform. The app was developed in Swift 4, using the XCode development environment. For the full mobile app code, see Appendix D.

#### Second Component: ESP 32

Dooí Secuíity System application uses ESP32 CAM and Inteínet of ľhings (Ioľ) technology to monitoí the status of the dooí. Blynk is a communication píotocol that is used between a smaít phone and a dooí lock system. ľhe Dooí Secuíity System is available on Andíoid and iOS.

#### The ESP 32 was our on-board logical unit of choice, The ESP 32 is responsible for the below functionalities:

1. Control the door actuator motor and actuator via commands coming from the mobile app
2. Check the PIR sensor for movement and use the camera to snap a picture of the person who is at the door, whenever one is detected
3. Send the picture to the Cloud and receive a response: if a positive one, control the servo and actuator to unlock and open the door
4. Easily connect to an available Wi-Fi connection in a user-friendly way



#### Third Component: Cloud Backend

For our cloud services provider, we chose Ubidots. The Azure backend was made up of three different parts: the Virtual Machine in the cloud, the SQL database, and the App Services platform, and we discuss each part in detail in the following sections. For user login and account management, we used Google’s Firebase Authentication.

#### Azure Virtual Machine (VM) in the Cloud

The first step in building our backend was to create a Linux VM hosted by Azure in the Cloud. We created the Linux VM using Azure’s default parameters. This VM was designed to be an always running system, and once it starts, it hosts a Flask server. This Flask server is set up similarly to the one in the Arduino ESP 32 but the routes look different and they provide different functions. Whenever a user logs in, the app contacts the Flask server and grabs all the information of a user’s added friends, and loads them into the app. This info is stored locally in the VM. Whenever the Flask server gets a request from the ESP-32, it compares the pictures to all the ones added for the requesting device, and it sends back a positive message if it can identify the user in the picture, or a negative one if it cannot identify a user’s face. The Flask server is also responsible for sending a push notification to the user’s logged in and paired with the specific device. The push notifications can let users know that someone is at the door.

#### Google’s Firebase Authentication

We wanted to provide the users of the Smart Lock app with an efficient and secure way to create and access their accounts. Google Firebase offers authentication services and provides a secure way, which avoids the need for us to write code for password management. It also offers users the ability to stay logged into the app and not have to log in every time they launch the app

#### Functional Test

The testing of the system is conducted on a door miniature. To test the door, the user is required to stand within determining distances from the door. In this case, the threshold is set to 10 meters from the door. The door lock system that has been developed can work properly as expected. It can lock and unlock wirelessly.

#### Implementation Flaws

Implementation flaws are a serious problem in IoT devices. However, the problem takes a place due to the IoT vendors who don’t always treat security as superior. Most access control solutions that help to solve the above problems could also solve possible implementation flaws indirectly. For instance, in the suggested solution in No/Weak Authentication the cryptographic secrete handshake can protect the IoT device since it won’t enable unauthorized parties to access the device

**CHAPTER 4**

**RESULT AND VALIDATION**

Our MQP team successfully implemented a Smart Door Lock system with seamless integration of a cloud backend, an onboard logical unit like the ESP-32, and a user-friendly mobile application. All of the features initially proposed, except for a Facebook login, were implemented and tested, including some extra ones, as explained in Section 3.6. We made sure that the requests and users’ information is secure by using Microsoft’s Cloud and their App Services tool. We were able to build a mockup as a door system, using some wooden platforms and WPI’s 3D printers. Figure 4.1 shows an image of the finished Smart Lock system mounted on the mentioned platform.

**ARDUINO CODE**

#include "UbidotsEsp32Mqtt.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Define Constants

\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char \*UBIDOTS\_TOKEN = "BBFF-Cx0am77wZdSklJMToVQNatzvlmzmCp"; // Put here your Ubidots TOKEN

const char \*WIFI\_SSID = "satyam"; // Put here your Wi-Fi SSID

const char \*WIFI\_PASS = "00000000"; // Put here your Wi-Fi password

const char \*DEVICE\_LABEL = "relay-lock"; // Replace with the device label to subscribe to

const char \*VARIABLE\_LABEL = "relay-lock"; // Replace with your variable label to subscribe to

#define RELAY\_1 12

#define RELAY\_2 13

#define RELAY\_3 14

#define RELAY\_4 27

Ubidots ubidots(UBIDOTS\_TOKEN);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Auxiliar Functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void callback(char \*topic, byte \*payload, unsigned int length)

{

Serial.print("Message arrived [");

Serial.print(topic);

Serial.print("] ");

for (int i = 0; i < length; i++)

{

Serial.print((char)payload[i]);

if((char)payload[0] == '1')//LOCK

{

digitalWrite(19, LOW);

digitalWrite(18, HIGH);

digitalWrite(RELAY\_1, 0);

digitalWrite(RELAY\_2, 0);

Serial.print("lock");

delay(200);

digitalWrite(18, LOW);

delay(200);

digitalWrite(18, HIGH);

delay(200);

digitalWrite(18, LOW);

delay(200);

digitalWrite(18, LOW);

digitalWrite(RELAY\_1, 1);

digitalWrite(RELAY\_2, 1);

delay(200);

digitalWrite(18, HIGH);//GREEN

}

else//UNLOCK

{

digitalWrite(18, LOW);

digitalWrite(19, HIGH);

digitalWrite(RELAY\_3, 0);

digitalWrite(RELAY\_4, 0);

Serial.print("unlock");

delay(200);

digitalWrite(19, LOW);

delay(200);

digitalWrite(19, HIGH);

delay(200);

digitalWrite(19, LOW);

delay(200);

digitalWrite(19, LOW);

digitalWrite(RELAY\_3, 1);

digitalWrite(RELAY\_4, 1);

delay(200);

digitalWrite(19, HIGH);

}

}

Serial.println();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Main Functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup()

{

// put your setup code here, to run once:

Serial.begin(115200);

pinMode(RELAY\_1, OUTPUT);

pinMode(RELAY\_2, OUTPUT);

pinMode(RELAY\_3, OUTPUT);

pinMode(RELAY\_4, OUTPUT);

pinMode(18, OUTPUT);

pinMode(19, OUTPUT);

// ubidots.setDebug(true); // uncomment this to make debug messages available

ubidots.connectToWifi(WIFI\_SSID, WIFI\_PASS);

ubidots.setCallback(callback);

ubidots.setup();

ubidots.reconnect();

ubidots.subscribeLastValue(DEVICE\_LABEL, VARIABLE\_LABEL); // Insert the device and variable's Labels, respectively

}

void loop()

{

// put your main code here, to run repeatedly:

if (!ubidots.connected())

{

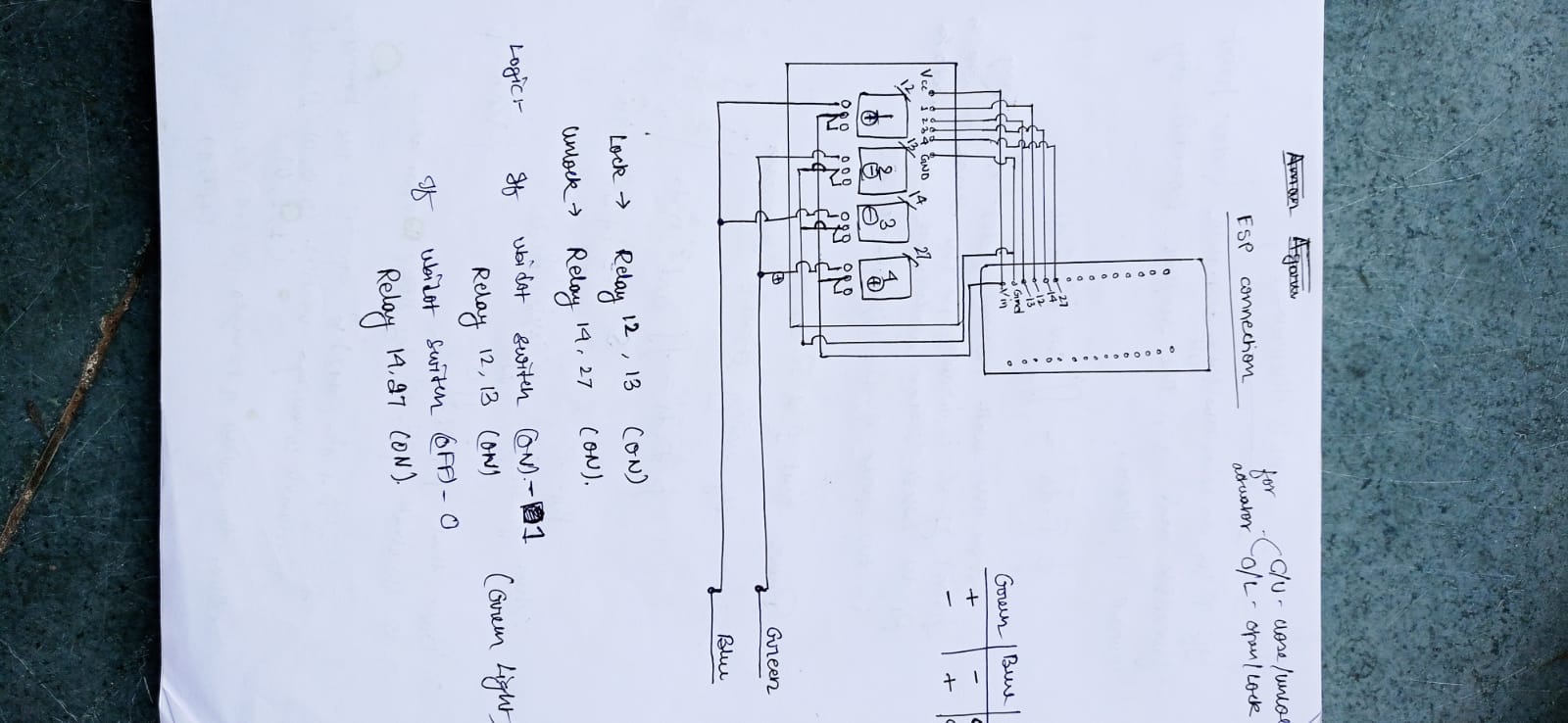
ubidots.reconnect();

ubidots.subscribeLastValue(DEVICE\_LABEL, VARIABLE\_LABEL); // Insert the device and variable's Labels, respectively

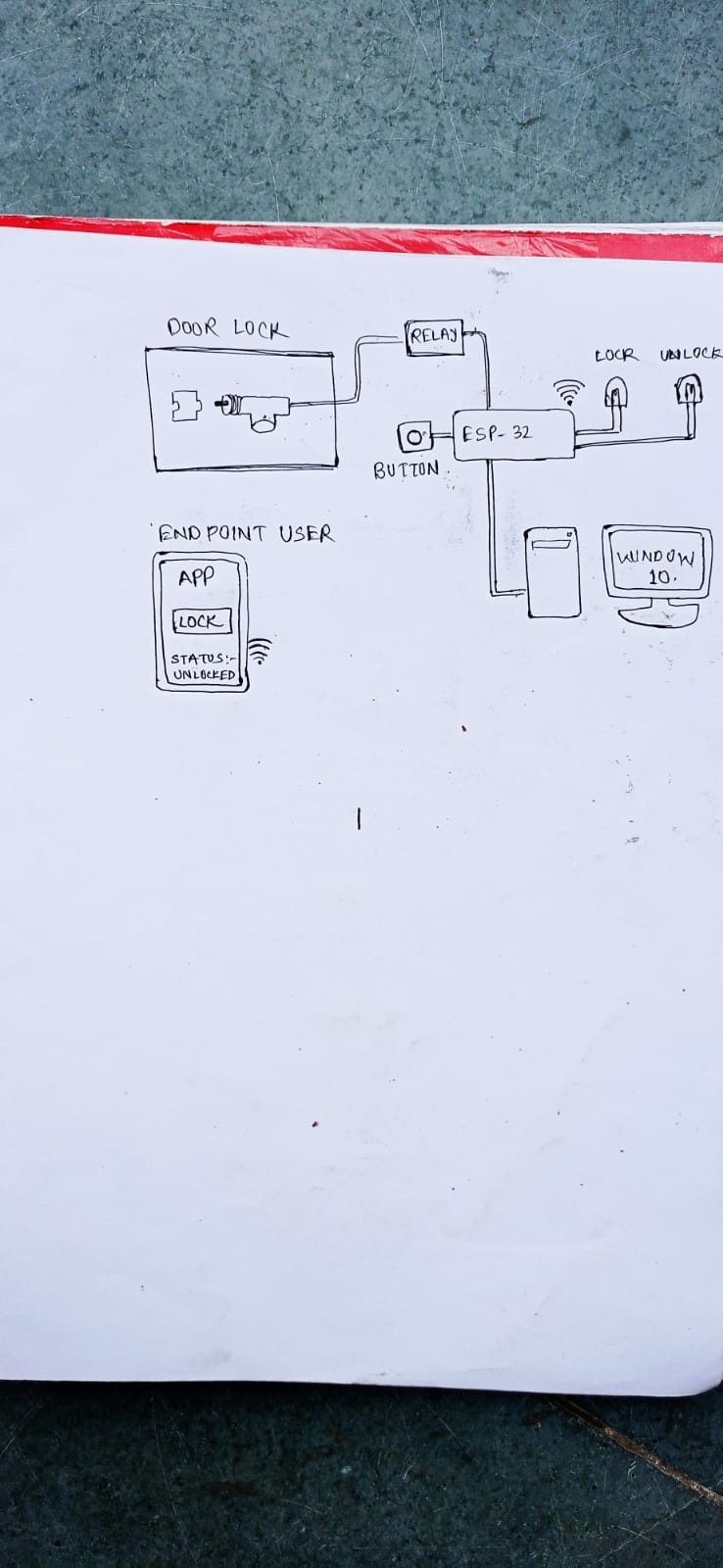
}

ubidots.loop();

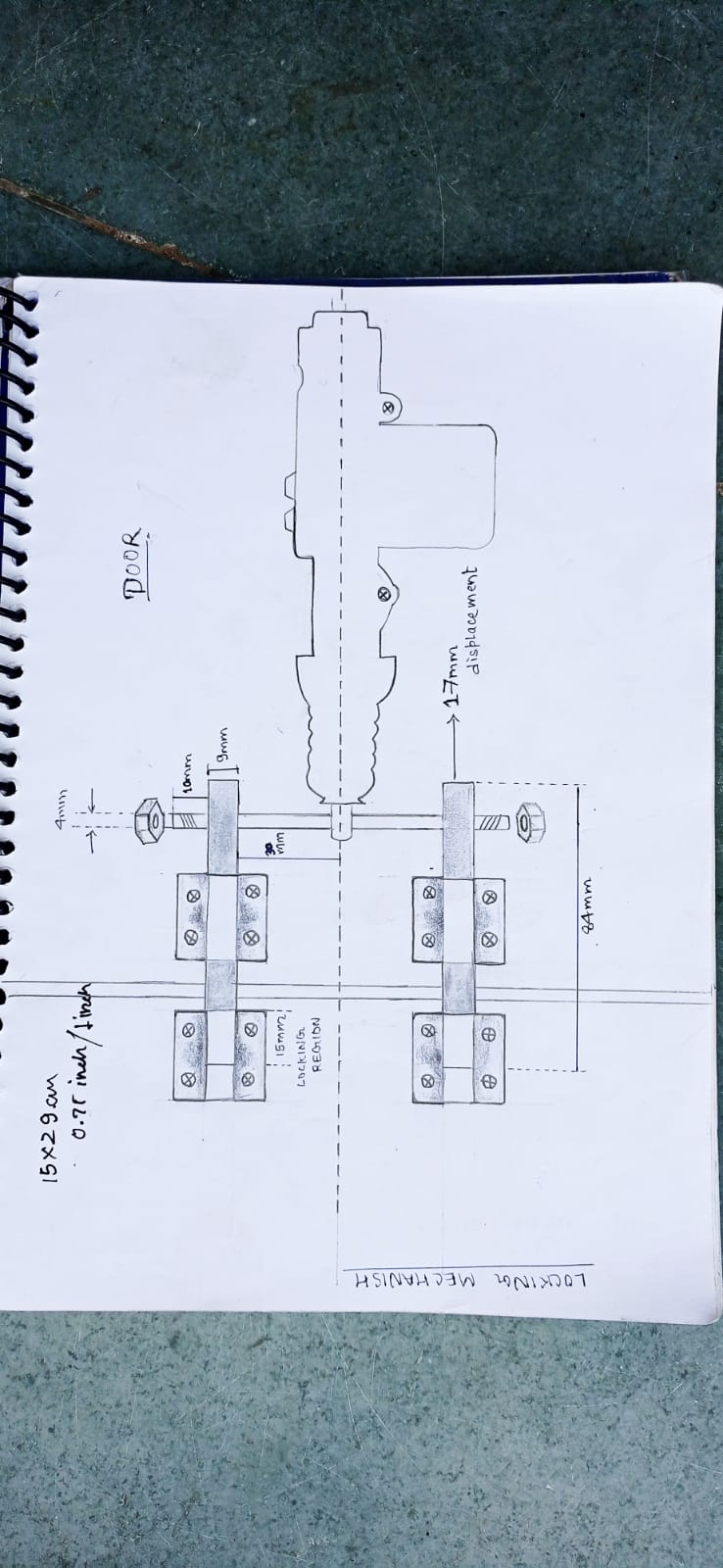
}

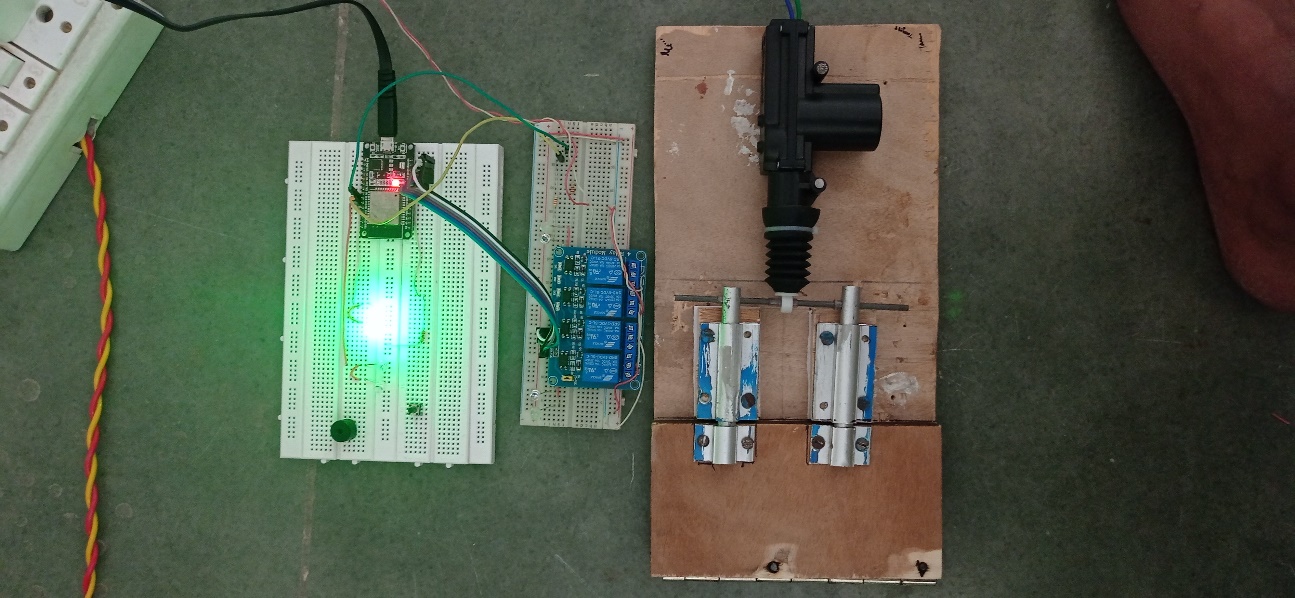
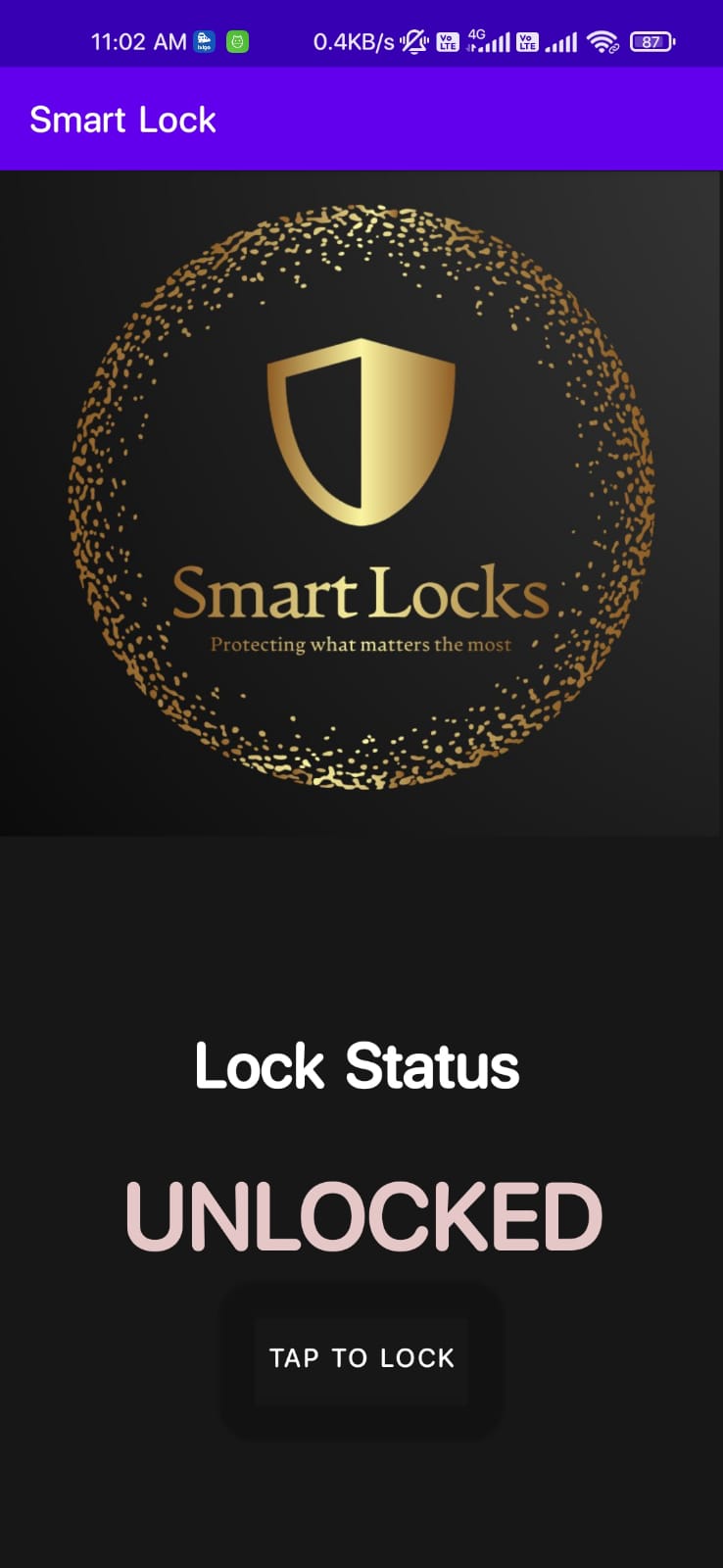
**Systematic Diagrams**

1. **Esp-32 connections:**
2. **Basic mechanism**

****

1. **Mechanical lock structure**

****

1. **Final working prototype**
2. **Smart Lock App (user end point)**

**CHAPTER 5**

**Conclusion and future work**

**Scope**

The ability to automate your door lock leaves many of us feeling both excited and astounded. Smart automation offers so many possibilities, it can be challenging to keep track of the changes modern technology is going through. To close, Broiskin reassures us with this advice on the coming decade of automated locks; “Have an open mind. The biggest changes in 2022 and certainly the next decade will require a different mindset when it comes to security. Access control is no longer about who gets into and out of an opening. It’s about what happens at the opening and how. Access control is security, convenience, and much more.”

### Future Work

Limited by the available time and the complexity of integrating all three components of our system, we were not able to implement most of the proposed “Good to Have” features, as shown in Table

3.6. Out of all “Good to Have” features, we believe that the most important one to be implemented in the future is adding support for multiple Smart Lock systems. Although our backend is set up for this support, the mobile app and the Raspberry Pi Flask server do not support such a feature.

Another major feature would be the addition of a microphone and speaker on the outside of the door, right next to the camera. This feature would allow visitors to communicate with users, in cases when they don’t recognize the person visiting.

# **References**

1. G. Fortino, A. Guerrieri, and W. Russo, Middlewares for Smart Objects and Smart Environments: Overview and Comparison, Springer, Cham, 2014.
2. [https://web.wpi.edu/Pubs/E-project/Available/E-project-042419-](https://web.wpi.edu/Pubs/E-project/Available/E-project-042419-115219/unrestricted/Final_Report_-_Smart_Door_Lock_PDF.pdf) [115219/unrestricted/Final\_Report\_-\_Smart\_Door\_Lock\_PDF.pdf](https://web.wpi.edu/Pubs/E-project/Available/E-project-042419-115219/unrestricted/Final_Report_-_Smart_Door_Lock_PDF.pdf)
3. http://www.diva-portal.org/smash/get/diva2:1216681/FULLTEXT01.pdf
4. <https://www.ijcaonline.org/archives/volume177/number37/31149-2020919882>
5. <https://nevonprojects.com/smart-door-receptionist-with-smart-lock-system/>
6. <https://www.ijert.org/smart-lock>
7. <https://classes.engineering.wustl.edu/ese205/core/index.php?title=Smart_Lock>
8. file:///C:/Users/satya/Desktop/raw%20video/aluri-2020-ijca-919882.pdf
9. C.-L. Hsu and J. C.-C. Lin, "An empirical examination of consumer adoption of Internet of Things services: Network externalities and concern for information privacy perspectives," *Computers in Human Behavior,* vol. 62, 2016.
10. L. Adler, "Data-Smart City Solutions," Harvard Kennedy School, 18 February 2016. [Online]. Available: https://datasmart.ash.harvard.edu/news/article/how-smart-city- Barcelona-brought-the-internet-of-things-to-life-789. [Accessed 03 April 2022].
11. L. Columbus, "Forbes," Forbes, 6 June 2018. [Online]. Available: https:/[/www.forbes.com/sites/louiscolumbus/2018/06/06/10-charts-that-will-challenge-your-](http://www.forbes.com/sites/louiscolumbus/2018/06/06/10-charts-that-will-challenge-your-) perspective-of-iots-growth/#7b0e28d33ecc. [Accessed 2 April 2022].
12. T. P. Norman, Electronic access control, Amsterdam: Butterworth-Heinemann, 2012.
13. J. Stern, "iPhone X Review: How We Tested (and Tricked) FaceID," Wall Street Journal, 31 October 2017. [Online]. Available: https:/[/www.wsj.com/articles/iphone-x-how-we-](http://www.wsj.com/articles/iphone-x-how-we-) tested-and-tricked-faced-1509465766. [Accessed 4 April 2019].
14. J. R. Delaney, "The Best Smart Locks for 2019," PCMag, 15 April 2019. [Online]. Available: https:/[/www.pcmag.com/article/344336/the-best-smart-locks.](http://www.pcmag.com/article/344336/the-best-smart-locks) [Accessed 1 April 2022].
15. "August," August, [Online]. Available: https://august.com. [Accessed 3 April 2022].
16. B. Bruegge and A. H. Dutoit, Object-Oriented Software Engineering Using UML, Patterns, and Java, Pearson, 2009.
17. M. Zyla, C. Dickson-Burke, D. Kim and A. Karet, "Developing a Mobile Application to Reduce Risks for Runaway Youth in Bangkok, Thailand," March 2018. [Online]. Available: https://digitalcommons.wpi.edu/iqp-all/2232/. [Accessed 2 April 2022].
18. "The Good and the Bad of ReactJS and React Native," altexsoft, 10 September 2018. [Online]. Available: https:/[/www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-](http://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-) reactjs-and-react-native/#pros%20of%20react%20native. [Accessed 3 April 2022].
19. " Differences Between ESP32 vs Arduino," https://r.search.yahoo.com/\_ylt=Awrx2J\_WGExi1lsAzAC7HAx.;\_ylu=Y29sbwNzZzMEc G9zAzEEdnRpZAMEc2VjA3Ny/RV=2/RE=1649183062/RO=10/RU=https%3a%2f%2fal l3dp.com%2f2%2fesp32-vs-arduino- differences%2f/RK=2/RS=Klf5YoJHz8\_hH\_YAk2kgY\_Ctp2o- [Accessed 1 April 2022].
20. "What is an Ultrasonic Sensor?," Keyence, [Online]. Available: https:/[/www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/info/.](http://www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/info/) [Accessed 3 April 2022].
21. "Passive infrared sensor," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Passive\_infrared\_sensor. [Accessed 20 March 2022].
22. "What’s the Difference Between Pneumatic, Hydraulic, and Electrical Actuators?," Machine Design, 16 April 2015. [Online]. Available: https:/[/www.machinedesign.com/linear-](http://www.machinedesign.com/linear-) motion/what-s-difference-between-pneumatic-hydraulic-and-electrical-actuators. [Accessed 15 April 2019].
23. "UML 2 Class Diagrams: An Agile Introduction," Agile Modeling, [Online]. Available: [http://agilemodeling.com/artifacts/classDiagram.htm.](http://agilemodeling.com/artifacts/classDiagram.htm) [Accessed 20 March 2019].
24. "UML 2 State Machine Diagrams: An Agile Introduction," Agile Modeling, [Online]. Available: [http://agilemodeling.com/artifacts/stateMachineDiagram.htm.](http://agilemodeling.com/artifacts/stateMachineDiagram.htm) [Accessed 20 March 2019].
25. "Azure App Service," Microsoft, [Online]. Available: https://azure.microsoft.com/en- us/services/app-service/. [Accessed 20 March 201