CS-684-2018 Final Report

Smart Locks

Team : Hack Street Boys

Pratik Sanjay Wagh, 173050077
Rahul Sharma, 173050019
Rahul Gorwadkar, 173074017

Table of Contents

1. Introduction

Security is one of the major concern of every human being in this planet. Everybody wants to be secure at every point of time in life. There are various fields in which security are a big concern. Some of them are security of data transfer, security of personal information stored in cloud, security of assets (including home, lockers, vehicles) in our lives. Among the various fields we have chosen to address the problem of security that every single being is associated with. The motivation behind our work comes from the outmoded lock systems still in use to lock the doors and the trouble we face to keep our keys safe in our daily lives. We have come up with an idea to develop a smart lock that needs no physical key to unlock the lock. The smart lock can be operated from any remote location using an android application and internet connection. The lock has also been provided with an offline feature of knocking based unlock which makes it unique and upto the minute. Smart locks will prove to be very useful as it will obliterate the need to keep the physical keys safe all the time.

2. Problem Statement

Smart is the most common adjective, verb and noun used in the era of technological revolution. Every physical objects are evolving into smarter objects then why not making the physical locks around us smart. Therefore we came up with an idea to develop Smart Locks. Smart Lock is a device used to lock/unlock a door in two ways : a) remotely from mobile phone using android application, b) knocking based lock/unlock. Smart lock includes a hardware solenoid based lock mounted on the door, an accelerometer mounted on the door to detect the knocking pattern, a microcontroller and an android application to access the lock remotely.

3. Requirements

3.1 Functional Requirements

3.1.1 Login

3.1.1.1 Introduction: There are two type of users : a) Master user, b) Slave user. User can login using the login interface provided to the user as soon as user opens the application.

3.1.1.2 Input: For this feature the inputs will be the user_id and password.

3.1.1.3 Processing: The details entered above are collected via the form field and authenticated in the cloud via AWS cognito services.

3.1.1.4 Output: On successful authentication the user is logged in and redirected to the homepage of the application.

3.1.2 Unlock

3.1.2.1 Introduction: This feature is used to unlock the lock from android application using the cloud service.

3.1.2.2 Input: Unlock button is present in the home page of the smart lock application.

3.1.2.3 Processing: Pressing of the unlock button is reflected in the shadow created for the device using the AWS IoT Core service.

3.1.2.4 Output: The status of shadow is fetched by the device(lock) from the cloud and the signal is sent to the relay to unlock the lock.

3.1.3 Create User

3.1.3.1 Introduction: This feature is available only for the master user. It is used to create a duplicate user (slave user) to give access of the lock.

3.1.3.2 Input: The master user provides user_id, password and email_id of user as the input to the form.

3.1.3.3 Processing: The details entered above are collected via the form and saved in the cloud database (In our case DynamoDB and Cognito).

3.1.3.4 Output: As a result of this a user is created and added to the list of user in the User table. Also a verification password is sent to the email address entered by the master user to verify the account.

3.1.4 Generate Pattern

3.1.4.1 Introduction: This feature is used to generate a random list of integers and send the list to the shadow of IoT Core of AWS.

3.1.4.2 Input: User press the generate pattern button present in the generate pattern section of the dropdown menu.

3.1.4.3 Processing: The request to generate pattern triggers the module written in java to generate a random list of numbers and it return the list to the user on the screen.

3.1.4.4 Output: A list of random number is generated and printed on the screen of the user. The list is also updated in the shadow of the device in the cloud.

3.1.5 Change Password

3.1.5.1 Introduction: This feature is available for all type of users. It allow users to change their password anytime for security reasons.

3.1.5.2 Input: User has to provide the old and new password in the form fields.

3.1.5.3 Processing: Old password of the current logged in user is verified from the table of the user and replaced with the new password.

3.1.5.4 Output: The old password is replaced with the new password in the database in cloud(DynamoDB)

3.1.6 Logout

3.1.6.1 Introduction: This feature allows the user to log out from the application.

3.1.6.2 Input: User press the logout button present in the drop down menu.

3.1.6.3 Processing: A request is made to the cloud in order to terminate the session and log out the user.

3.1.6.4 Output: Successful log out of the user and return to the login page of the application.

3.1.7 Check Log

3.1.7.1 Introduction: This feature is available only for the master user. It allows the user to check the log of the unlock requests made by the users and the time at which the requests were made.

3.1.7.2 Input: The master user presses the check log button in the dropdown menu.

3.1.7.3 Processing: A request is made to the AWS cloud service to fetch the list of all the access to locks in the past.

3.1.7.4 Output: List of all the users is displayed along with the user id, date and time on which the unlock request was made.

3.1.8 Knock Detection

3.1.8.1 Introduction: This is an offline feature which allows a user to knock on the door and successfully unlock the door on correct knocking while not unlocking the door on incorrect knocking.

3.1.8.2 Input: Physical knocks on the door with the desired time interval between two successive knocks.

3.1.8.3 Processing: The device mounted on door activate the module after first knocks and stores the time interval between each knock and the total number of knocks.

3.1.8.4 Output: On successful knock detection the micro-controller signals the relay to unlock the lock and to keep the lock active on unsuccessful knocking.

3.2 Non-Functional Requirements

3.2.1 Security :

Security is met by the smart locks in the following ways:

1. Only authenticated users are allowed to log-in and unlock the lock.
2. Logs are maintained for every user which keeps track of the activity of other users.
3. Knocking patterns are generated randomly with different random seeds everytime.
4. The lock reports a notification alarm to the master user on 5 unsuccessful attempt to unlock the lock using knocking pattern.

3.2.2 User Friendly: The android application is made very interactive and user friendly with simple functionalities so that everybody can use it comfortably.

3.2.3 Maintainability: Since all the data including the list of users, check logs and other other data are present in cloud therefore there is no need to give extra effort to maintain database.

3.2.4 Availability

Smart lock android application is a light-weight application and all the data is present in cloud which increases the availability of the system. Also the lock mounted on the door needs a constant power source which is either supplied through a socket or battery(in case of power failure).

3.2.5 Accuracy and Performance

The accuracy of knocking is ensured by refining the value recorded by accelerometer and sending the data to controller only when a threshold value is reached or crossed. In order to provide laxity to the users regarding the timing between each knocks (+/-)500 milliseconds of lag is acceptable by the device.

3.3 Hardware Requirements

Following are the list of hardware required to make smart locks:

1. EPS8266 Board with WiFi Module
2. Electrically operated Solenoid based Lock
3. Accelerometer Sensor (ADXL 312)
4. Power Supply - 12V & 5V
5. Relay Module

6. Android Phone

3.4 Software Requirements

Following are the software required in the making of smart locks:
1. Android Studio - To make android application
2. Arduino IDE - To edit and compile code for ESP8266
3. AWS - To store the data of user(DynamoDB), to authenticate users(Cognito), to publish and subscribe data to and from the android application and microcontroller.

4. System Design

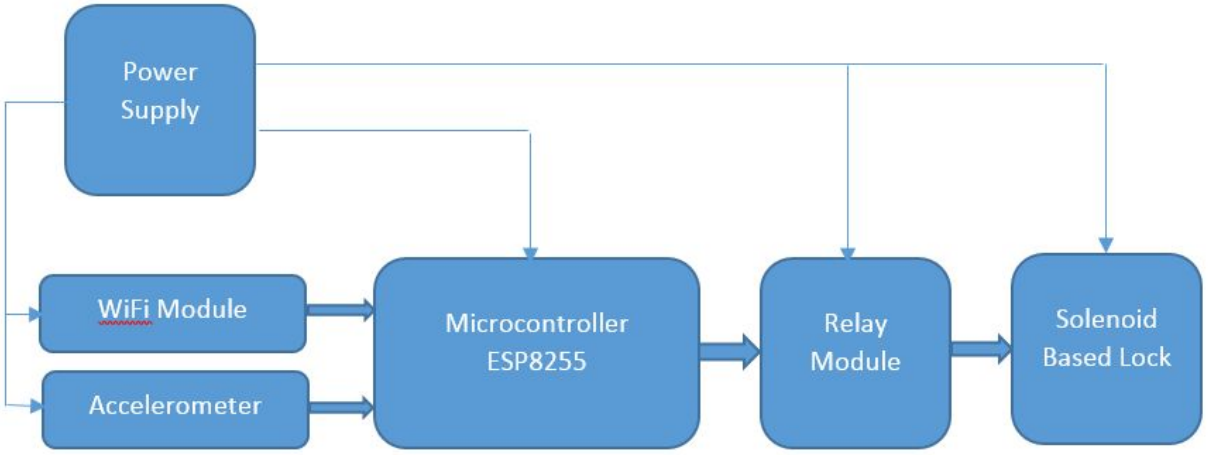


Figure 1 : System Architecture of Lock

Figure 1 highlights the system architecture of a smart lock. The architecture described above includes the following module :

1. Power Supply : Power supply of 5V and 12V is used to drive the micro-controller and the solenoid based lock.
2. ESP8266 with Wifi Module : An ESP8266 Board with wifi module is used which acts as the core of the lock. It contains the pins configuration code and the knocking detection logic running in it. It is also connected to the accelerometer that provides the data whenever a knock is detected by the accelerometer. It is also connected to the relay module to signal the lock to unlock.
3. Accelerometer : MPU6050 sensor is used to detect the knocks on the door. It transfer data to the microcontroller.
4. Relay Module : It act as a switch to lock or unlock the door. It is connected to the ESP8266 module and the physical lock.
5. Solenoid Based Lock : It is a physical lock which creates a magnetic field when current is allowed to pass through it. In our case the flow of current is controlled by the relay module.

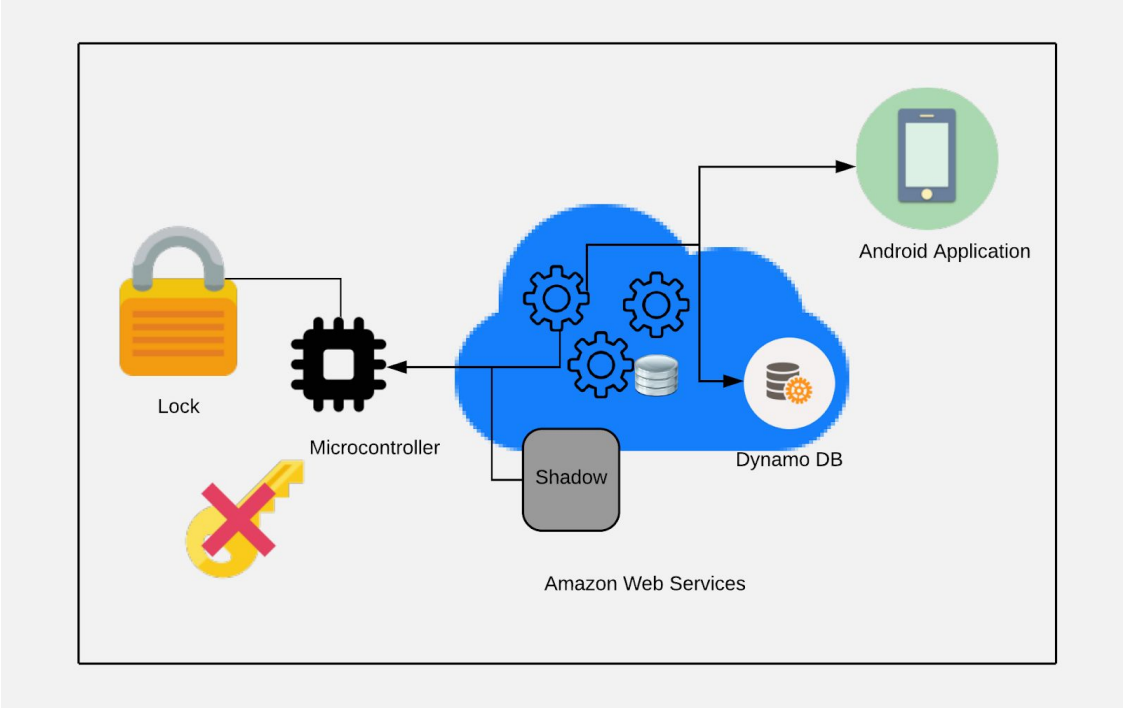


Figure 2: Overall architecture of the system

5. Working of the System and Test results

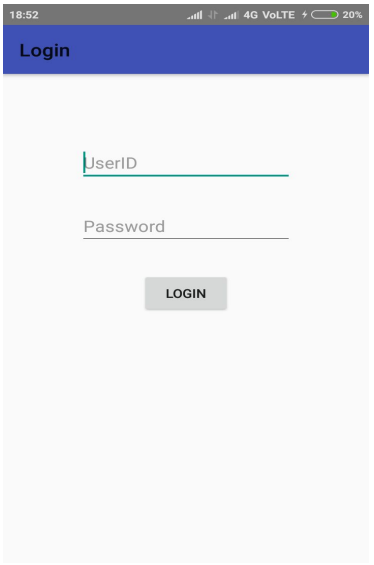
5.1 Working of Android Application:

Android Application is for accessing the lock, thus we can give an unlock and knocking pattern to the lock. So in our android application we have two types of users, one is master and other is slave. Master user is the user who creates a slave users.

So initially a master user is given a username and password whenever he purchase our lock. So in our case a master user has

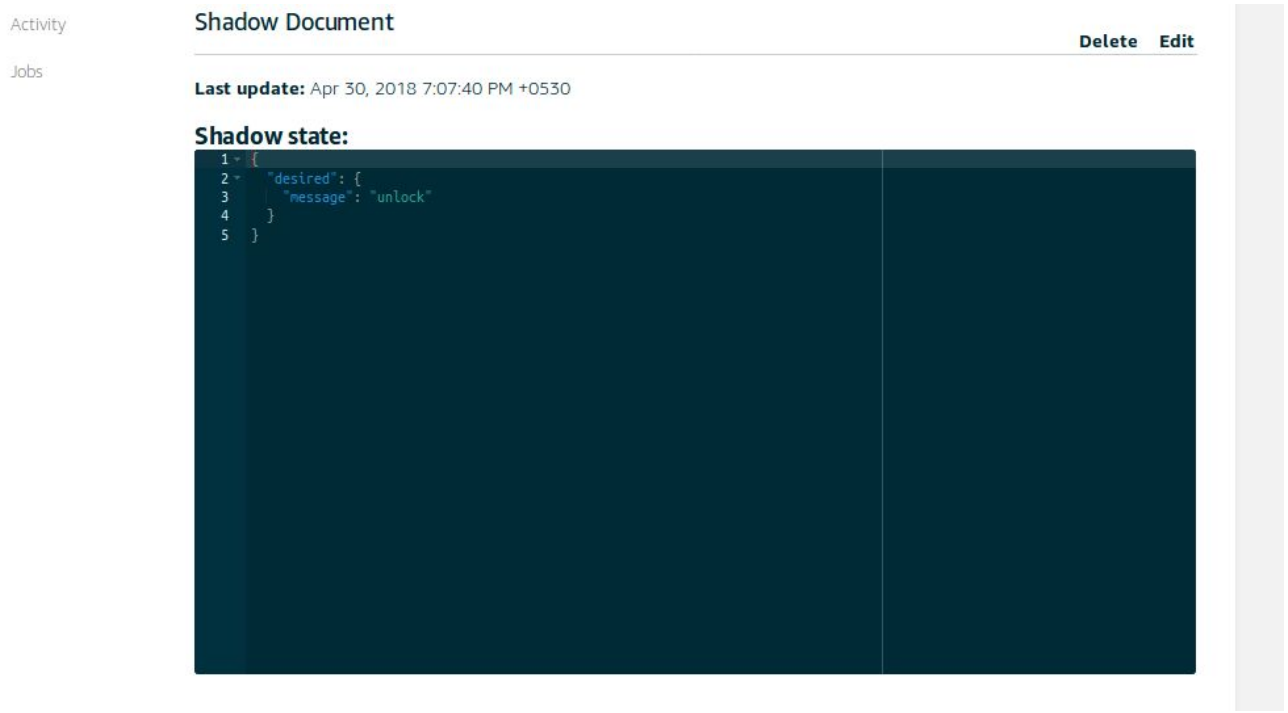
UserID :- embedded

Password :- embedded

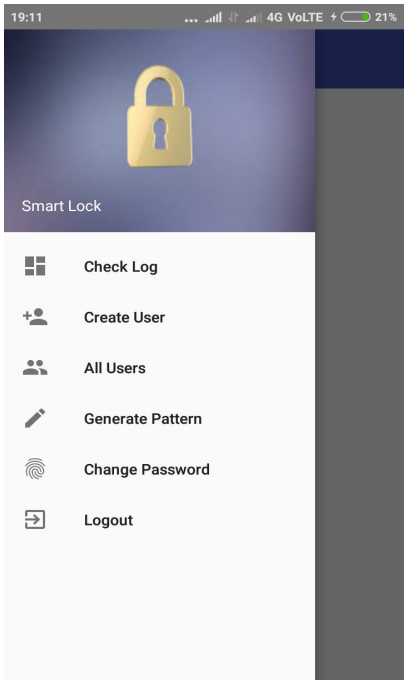


So this is the home page of an android application. Once a user enters a userID and password in the given fields correctly. He fill be successfully logged into the app and we will be taken to the HomePage which looks like the below image.

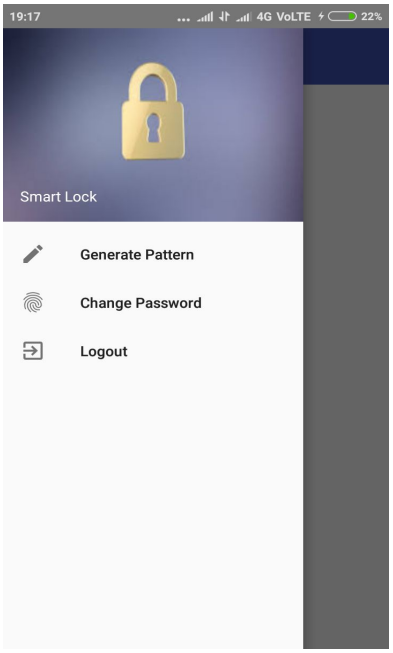
The master user can unlock the lock by pressing the unlock button which is visible in the above image. Once the user enters the unlock button an unlock message will be send to the shadow link of the aws iot. The shadow state will get update as seen in the below image.



The master user can use various feature as seen in the side navbar of the below image. Check Log, Create User, All Users, Generate Pattern, Change Password, Logout are the features available only for master user.

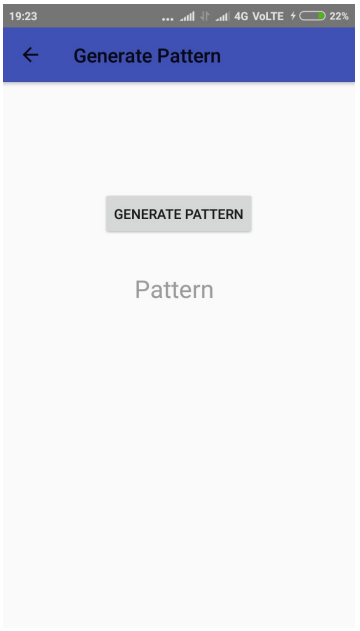


The slave user has less features (Generate Pattern, Change Password and Logout) which can be seen in the below image.



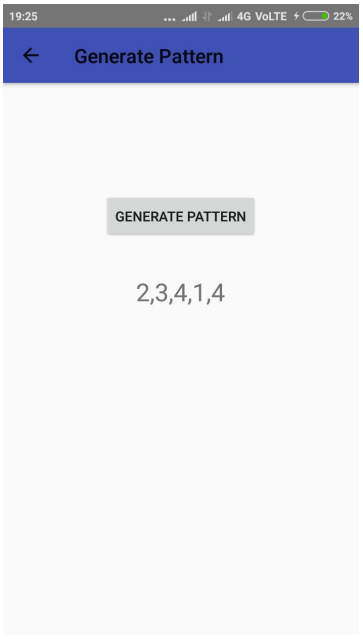
Now if a user wants to send a generate pattern to some other user. He will open a Generate Pattern Page which can be opened by pressing the Generate Pattern option present in the side navbar which can be seen in the above image.

The Generate Pattern page will look like the below figure.

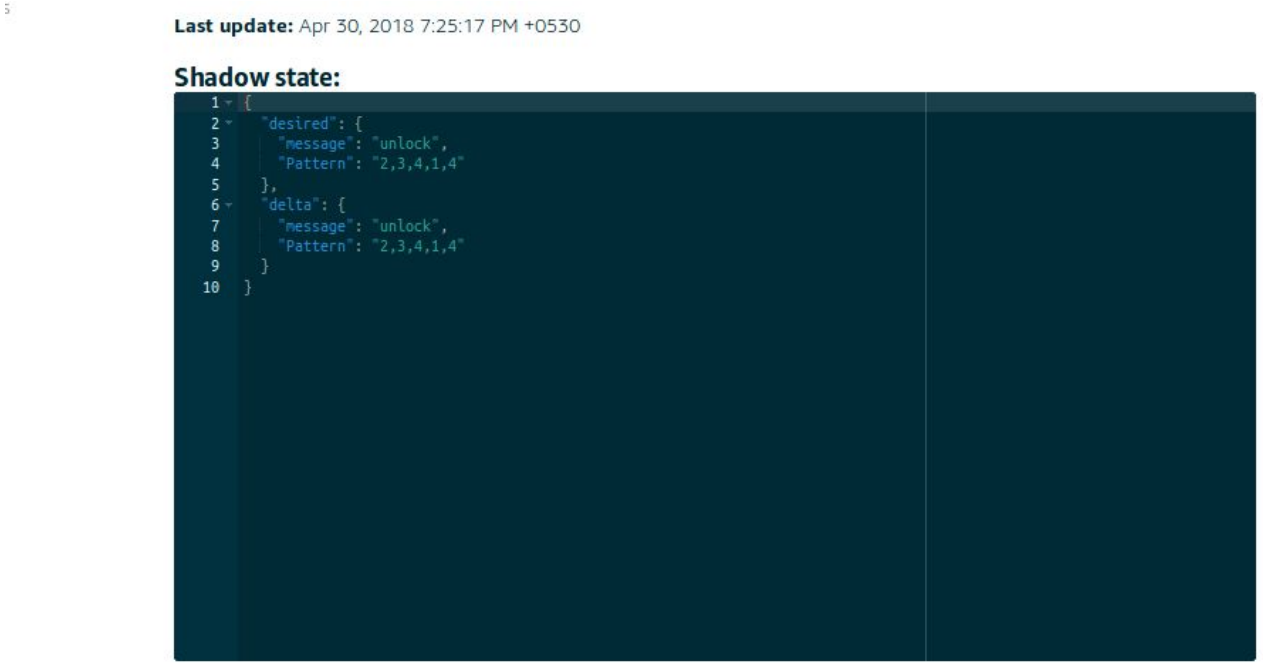


Once the user presses a generate pattern button which can be seen in the above image, the pattern generate will be send to the shadow link of the aws iot.

The generated pattern in the android application after pressing the generate pattern button looks like the below image.



And the shadow link will look like the below image.



Thus in this way the master user and slave user sends the unlock and knocking pattern to the shadow link of the aws iot and this data is fetched by the lock once any update happens to this shadow link.

5.2 Working of Device (Lock):

ESP8266 is brain of the device which is doing all the processing required for connectivity with AWS, communication with MPU6050 over I2C and running an algorithm to detect knocking pattern and to compare it with predefined knocking pattern to decide whether to lock or unlock the door. MPU6050 is an accelerometer which provides values of acceleration subjected to accelerometer module over I2C communication channel.

MPU6050 will be mounted on door surface to detect knocking on door with good sensitivity. ESP8266 will fetch data of accelerometer data at every 10 milliseconds and this data will be filtered to remove high frequency components. Once first knock is detected then for every knock, time difference between successive knocks is stored in an array which will be compared with predefined array provided by master user application. If both the patterns(arrays) match the controller will make a decision to unlock the door by turning the port value low. This port is pulled up internally and default value available on the port is high. This default state is lock state. This port is interfaced with a relay through a transistor. Once port is pulled down, the relay will be actuated. This relay can connect or disconnect supply of electromagnetic lock thus making this lock modular.

ESP8266 can connect with android app using either DynamoDB or Shadow service. These both the services are offered by AWS for IoT based communication between master and slave device. For DynamoDB communication and programming ESP8266, Arduino IDE is used along with ESP8266AWS SDK. This SDK is in developmental phase. We need to configure following configurations for this code to work:

i. AWS Region

ii. WiFi SSID & Password

iii. DynamoDB Table Name, Hash Key Name, Hash Key Value & Range Name

iv. AWS Key ID & Secret key

v. DynamoDB data updation based on events, inputs and outputs

Testing of a Device:

Faithful knock detection is of prime importance and nuisance knock detection may completely cause smart lock to fail. Following screenshot shows knock detection shown as ‘Knock = 1’ after every knocking and after 10 second timeout knocking pattern is shown in numbers by showing difference between successive knocks. In the end, this pattern is compared with predefined pattern to take decision to unlock the door. There are two tests shown in screenshot shown below.

In first test, intentionally different knocking pattern was provided and after timeout it can be seen that door was kept locked and location of knock which does not match with predefined pattern is also shown.

In second test, predefined knocking pattern was provided and after timeout it can be seen that door was unlocked.

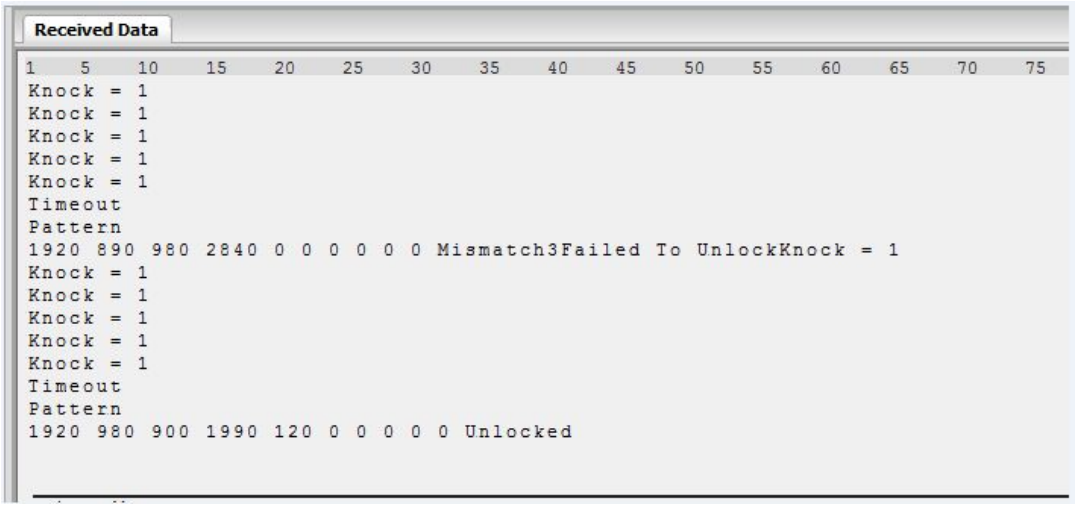


Figure showing knock detection, pattern recognition and Locking/Unlocking

DynamoDB connectivity was established between ESP8266 and AWS using ESP8266AWS SDK. Following screenshot shows a random data generated in ESP8266 as a temperature and updated on DynamoDB table after few seconds.

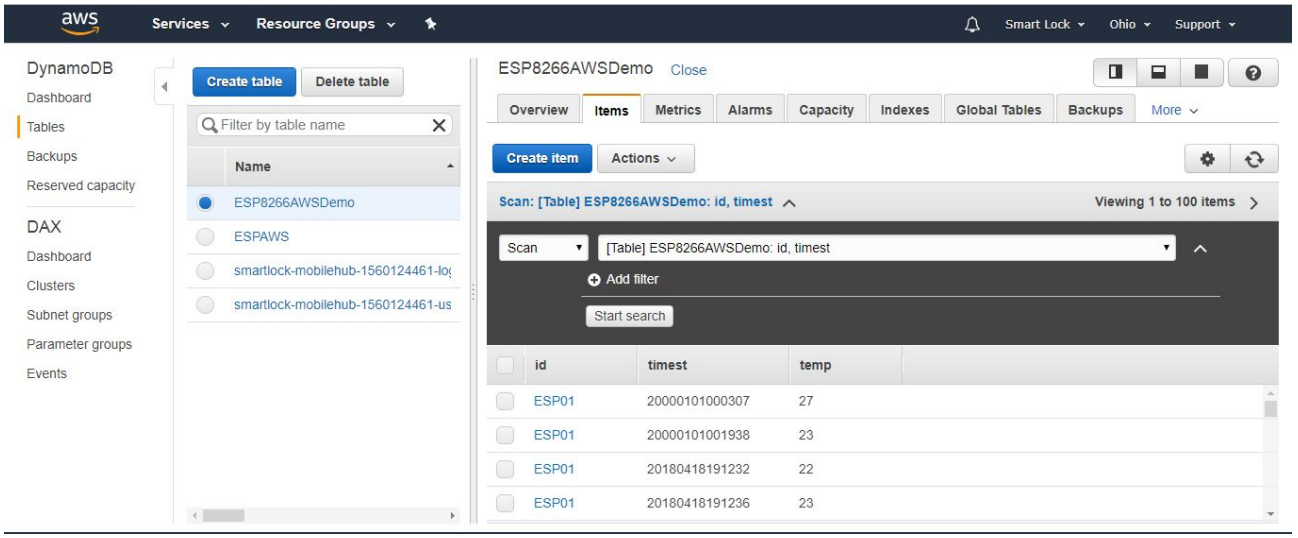


Image showing DynamoDB table updated through ESP8266

6. Discussion of System

Components of your project worked as per plan

As per the requirements specified by us in the report, following are the parts of our system working as per our plan:

1. Successful creation of duplicate user by master user.

2. Successfully maintaining and updating the database of users and the logs.
3. Successful authentication of user using AWS Cognito service.
4. Successfully updating the shadow from the android device using AWS IoT Core service..
5. Successfully generating random knocking pattern.
6. Checking the logs of all the users.
7. Successfully interfaced accelerometer over I2C protocol.
8. Successfully implemented filter for knocking detection.
9. Successfully integrated ESP8266 with accelerometer and relay module in the prototype.

7. Future Work

Following are the additional features we can implement as part of future work:

1. Mounting an led in the door which glows in interval of 1 second to make it easier for user to keep note of time interval between two consecutive knocks.
2. Adding a buzzer which blows an alarm if more than 5 incorrect knocking pattern is detected and also if lock is accessed using unwanted physical means.
3. Also unlocking of locks can be done using music pattern recognition, voice detection and face recognition.

8. Conclusions

We have successfully built Smart Locks with the fulfillment of all the functional and non-functional requirements as stated in the report. We created the android application with the stated features. We have also successfully implemented the knocking pattern detection code. So we are done with major security concern where a user sends an unlock and knocking pattern to the AWS IoT Core and a unique knocking pattern detection method to unlock the door. As a prototype model we implemented the system without solenoid based lock by mounting the device in a carton and successfully tested the working of the lock.

9. References

1. https://github.com/daniele-salvagni/aws-sdk-esp8266
2. https://github.com/awslabs/aws-sdk-android-samples/tree/master/AndroidPubSubWebSocket
3. https://aws.amazon.com/documentation/cognito/
4. https://aws.amazon.com/documentation/dynamodb/
5. https://docs.aws.amazon.com/iot/latest/developerguide/iot-gs.html
6. https://github.com/sanamshakya/interfacing-AWS-IoT