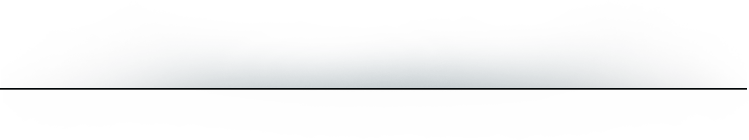
|  |  |
| --- | --- |
|  | **MINISTRY OF EDUCATION AND TRAINING** |



**CAPSTONE PROJECT DOCUMENT**



**HOME SECURITY SYSTEM**

Members: Truong Van Cuong

Dinh Tru Ngoc Diep

Vo Tuan Hung

Dinh Thanh Dung

Supervisor: PhD. Phan Duy Hung

Project Code: HSS

**Hanoi, April 25th, 201****6**

*“We wish to express our happiness and say thanks to our supervisor because of the willingness for overseeing and taking whole capstone project under control through this semester. The successful completion of this capstone project is just a small start point in our career but an incredible milestone in student life. We also want to say thanks to our family, all my friends for the best encouragement and support."*

- HSS Team -

**TABLE OF CONTENTS**

[ACRONYMS AND DEFINITION 6](#_Toc449296230)

[THE PEOPLE 7](#_Toc449296231)

[CHAPTER 1 – INTRODUCTION 8](#_Toc449296232)

[1.1. Project Information 8](#_Toc449296233)

[1.2. Background 8](#_Toc449296234)

[1.3. Existing System 9](#_Toc449296235)

[1.3.1. IDEAL Security System 9](#_Toc449296236)

[1.3.2. SimpliSafe Home Security System 10](#_Toc449296237)

[1.3.3. iSmartAlarm Premium Package 11](#_Toc449296238)

[1.4. Ideas and Objectives 11](#_Toc449296239)

[1.5. The Scope of System 13](#_Toc449296240)

[CHAPTER 2 – PROJECT MANAGEMENT PLAN 14](#_Toc449296241)

[2.1. Tools and Software 14](#_Toc449296242)

[2.3. Risk Management Plan 16](#_Toc449296243)

[2.4. Communication Plan 17](#_Toc449296244)

[CHAPTER 3 – SYSTEM REQUIREMENT SPECIFICATION 18](#_Toc449296245)

[3.1. Overview 18](#_Toc449296246)

[3.2. Functional Requirements 18](#_Toc449296247)

[3.3. Non-Functional Requirements 19](#_Toc449296248)

[3.3.1. Security 19](#_Toc449296249)

[3.3.2. Reliability 19](#_Toc449296250)

[3.3.3. Safety 19](#_Toc449296251)

[3.3.4. Usability 19](#_Toc449296252)

[3.4. HSS-App Use Cases 20](#_Toc449296253)

[3.4.1. Login 20](#_Toc449296254)

[3.4.2. Show Activity Log 21](#_Toc449296255)

[3.4.3. Show Camera IP List 22](#_Toc449296256)

[3.4.4. Call Doorbell 23](#_Toc449296257)

[3.4.5. Control Pushing Sensors 24](#_Toc449296258)

[3.4.6. Change Password 25](#_Toc449296259)

[3.4.7. Clear Activity Log 26](#_Toc449296260)

[3.4.8. Logout 27](#_Toc449296261)

[CHAPTER 4 – SYSTEM DESIGN DESCRIPTION 28](#_Toc449296262)

[4.1. Hardware Components 28](#_Toc449296263)

[4.1.1. Raspberry Pi 2 28](#_Toc449296264)

[4.1.2. Arduino Uno 29](#_Toc449296265)

[4.1.3. Sensors 30](#_Toc449296266)

[4.1.4. Support Components 31](#_Toc449296267)

[4.2. System Architecture Overview 33](#_Toc449296268)

[4.1.1. System Architecture 33](#_Toc449296269)

[4.1.2. Architectural Explanation 34](#_Toc449296270)

[4.1.3. Sequence Diagram 36](#_Toc449296271)

[4.3. Hardware design 38](#_Toc449296272)

[4.3.1. System Schematic 38](#_Toc449296273)

[4.3.2. After Mounting on Demo Box 39](#_Toc449296274)

[4.4. Embedded and HSS-App Design 39](#_Toc449296275)

[4.4.1. Database 39](#_Toc449296276)

[4.4.2. Login 41](#_Toc449296277)

[4.4.3. Change Password 42](#_Toc449296278)

[4.4.3. Pushing Notification 43](#_Toc449296279)

[4.4.4. Control Pushing Notification 46](#_Toc449296280)

[4.4.5. Calling Doorbell 46](#_Toc449296281)

[4.4.6. Take and Load Photo 48](#_Toc449296282)

[4.5. HSS-App User Interface Design 50](#_Toc449296283)

[4.5.1. Login Screen 50](#_Toc449296284)

[4.5.2. Home Screen 50](#_Toc449296285)

[4.5.3. Activity Log Screen 51](#_Toc449296286)

[4.5.4. Camera IP Screen 51](#_Toc449296287)

[4.5.5. Doorbell Screen 52](#_Toc449296288)

[4.5.6. Setting Screen 52](#_Toc449296289)

[CHAPTER 5 – IMPLEMENTATION 53](#_Toc449296290)

[5.1. Login 53](#_Toc449296291)

[5.2. Change Password 53](#_Toc449296292)

[5.3. Delete Activity Log 54](#_Toc449296293)

[5.4. Pushing Notification 55](#_Toc449296294)

[5.5. Control Pushing Notification 56](#_Toc449296295)

[5.6. Calling Doorbell 56](#_Toc449296296)

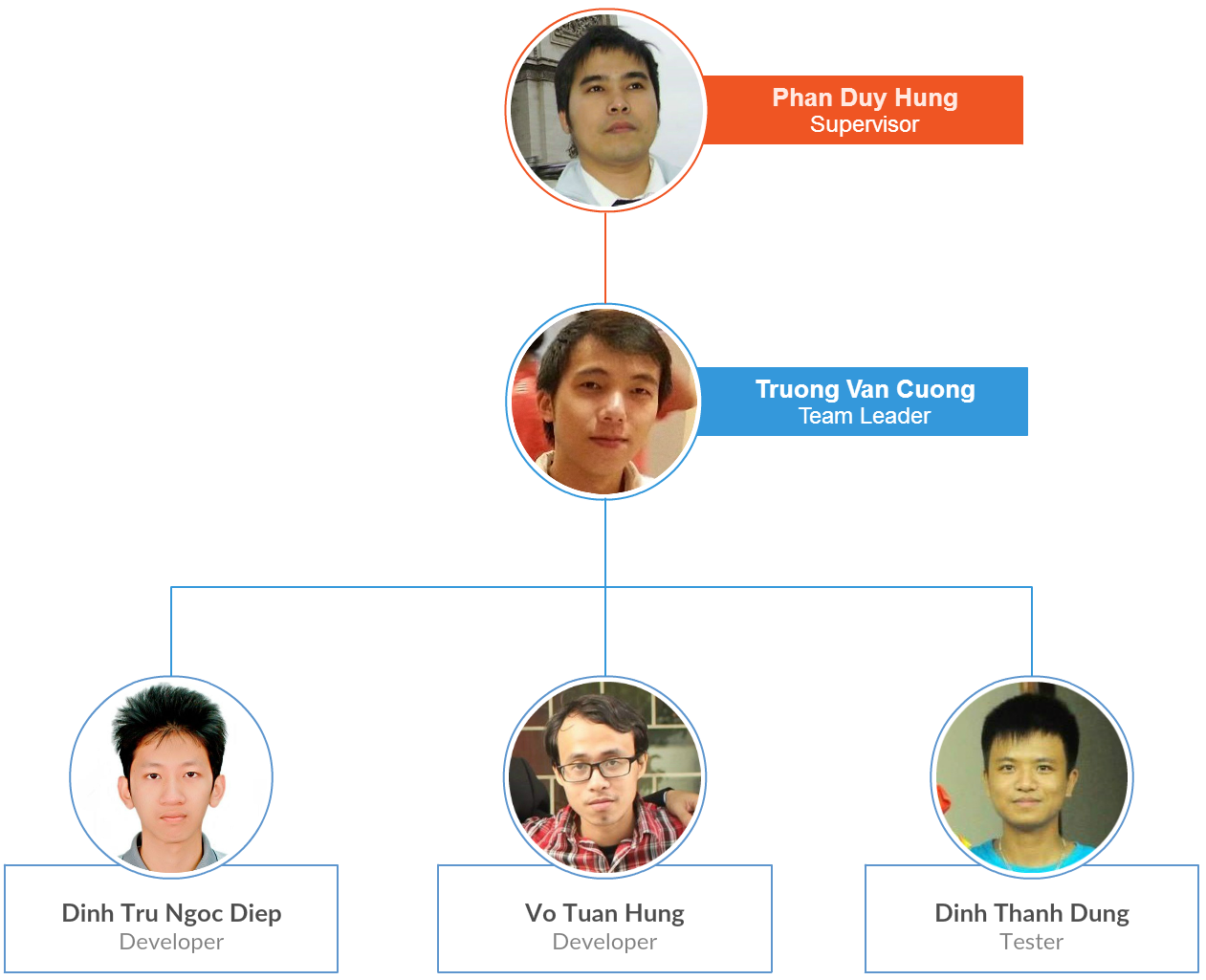
[CHAPTER 6 – FUTURE WORKS 58](#_Toc449296297)

[REFERENCES 59](#_Toc449296298)

# ACRONYMS AND DEFINITION

|  |  |
| --- | --- |
| Acronym | Definition |
| HSS | Home Security System |
| HSS-App | Home Security System Client Application |
| HSS-Board | HSS Central Circuit Bboard |
| GPIO | General-Purpose Input/Output |
| WLAN | Wireless Local Area Network |
| UDP | User Datagram Protocol |
| TCP | Transmission Control Protocol |
| IP | Internet Protocol |
| N/I | No Information |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# THE PEOPLE



# CHAPTER 1 – INTRODUCTION

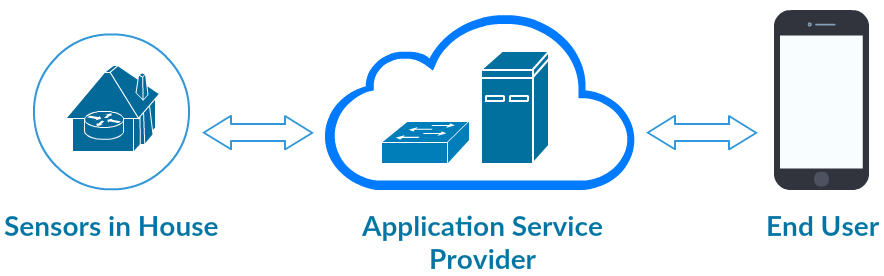
## 1.1. Project Information

* Project name: Home Security System
* Project code: HSS
* Group name: HSS4T
* Product type: Software Embedded System and Android Application
* Timeline: Jan 2016 to April 2016

## 1.2. Background

The Internet of Things (IoT) is an important topic in technology industry, policy, and engineering circles and has become headline news in both the specialty press and the popular media. This technology is embodied in a wide spectrum of networked products, systems, and sensors, which take advantage of advancements in computing power, electronics miniaturization, and network interconnections to offer new capabilities not previously possible. An abundance of conferences, reports, and news articles discuss and debate the prospective impact of the “IoT revolution”—from new market opportunities and business models to concerns about security, privacy, and technical interoperability[[[1]](#footnote-1)].

There are 2 ways of connection in IoT: device-to-device communication, with devices communicating with each other directly via protocols like Bluetooth, Zigbee… and device-to-cloud communication, with the IoT device connects directly to an Internet cloud service to exchange data and control message traffic. In HSS project, we choose device-to-cloud communication model. In details, it means the IoT devices connect directly to an Internet cloud service like an application service provider (Pusher, Linphone SIP Service). This approach frequently takes advantage of existing communications mechanisms like traditional wired Ethernet or Wi-Fi connections to establish a connection between the device and the IP network, which ultimately connects to the cloud service.



*Figure 1: Device-To-Cloud Communication Model*

## 1.3. Existing System

### 1.3.1. IDEAL Security System

(<http://www.idealinc.com/ideal/securityprotection.php>)

This allows user to create own security system. When the AUTO DIALER is connected to a telephone land line, an electrical outlet and set to ARM mode it will alert user when any of the sensors have been triggered. The auto-dialer will activate the siren or chime and dial out to up to 5 pre-programmed telephone numbers with your own pre-recorded message. User can choose to remotely listen in & broadcast to the protected areas. It includes 2 wireless door and window contacts, 1 motion sensor, 2 remote controls, 1 wireless siren and a telephone dialer.



*Figure 2: IDEAL Security System*

The disadvantages are battery light for 9 volt backup start flashing after 3 days as the battery has run down. In addition, if user has power failure it won't work and loses its memory. Its currently price is about $250.

### 1.3.2. SimpliSafe Home Security System

(<http://simplisafe.com/home-security-comfort>)

SimpliSafe is dedicated to making security simple for people to use. All devices come ready to install and operate using a plug and play system. The system includes various types of sensor: sensors for your doors and windows, motion detectors, panic button for emergencies, carbon monoxide detection, etc…

The price is about $260, but if user wants to control the system from smartphone, they'll have to pay up to $24.99 per month. SimpliSafe packages don't include video cameras, and you won't be able to connect with locks, lights, or other home automation devices over Z-Wave, Zigbee, or any other wireless network. The devices also look a little dated.



*Figure 3: SimpliSafe House Security System*

### 1.3.3. iSmartAlarm Premium Package

(<https://www.ismartalarm.com/gb/>)



*Figure 4: iSmartAlarm Premium Package*

The iSmartAlarm comes in two sizes; the $199 Preferred Package, which includes two door/window sensors, one motion sensor, two remote tags with presence sensors, and the brains of the system, the CubeOne. You also get one iSmartAlarm sign to stick on your window. We received the $349 Premium Package, which includes everything in the Preferred Package as well as an iCamera. You can purchase additional door/window sensors for $29.99 each and remote tags for $24.99 each. Additional motion sensors go for $34.99 each.

## 1.4. Ideas and Objectives

The field “Home Security” is booming vigorously. However, companies which offering this solution have a downside – they will only provide an entire solution package of home surveillance monitoring with automated solution, typical of BKAV with SmartHome solution. The demand of market is very huge, but the user do not have sufficient funding to find a compact monitoring solution and have effective at cost, not too expensive and of course, can be deployed immediately at existing house.

Typically the first choice of people is a supplier which provides monitoring system solution via camera IP. But the fact that, camera IP is just only support solution, and it’s passive, has a lot of weak points when do the actual deployment. For example: the last observation camera IP must always be carried out continuously from afar; appear the dead viewing angle (the camera range could not reach). Or when you are at home, the camera IP will not promote effective supervision in a comprehensive way in many different positions (many floors or rooms).

Therefore, we want to build a project to solve limitations of camera IP by implementing a comprehensive monitoring solution, also put the IP camera into its own place in system architecture. Whether you're away from home or in another room, just a phone in hand with an Internet connection, the system would alarm immediately on your phone for the suspicious activity (like opening the door or motion detection in particular room…).

Not only that, when you are away from home, a lot of people could visit your house, and ring the bell. People always want to be notified immediately to their smartphone and initiating a connection to the doorbell block (speaker/microphone inside). Then, they can start a conversation with the visitor immediately, like a normal call.

Why did we choose developing mobile application but for laptop or desktop computer even they also can connect to internet? The answer is very simple: According to a reliable study of Morgan Stanley from Technology & Internet Trends, “91% of people keep their phone within 3 feet 24 hours a day” [[[2]](#footnote-2)]. It’s the most important reason.

In summary, our Home Security System should have below features:

- Authenticating between smartphone application and server

- Detecting suspicious activities like motion in house or opening door illegally

- Alarm to user by sending notifications in real-time technology

- Save activity log to database on server, and user can check that log later on smartphone application

- Controlling the doorbell by seeing who the visitor is and start a conversation with them

- Have ability to support viewing traditional camera IP that deployed in the house

- Some available settings for user to optimize system’s performance: change password, clear activity log, disable pushing notification or particular sensor or logout

## 1.5. The Scope of System

This project is a prototype of comprehensive solution for home security surveillance problems in modern life. Because the limitation of using public Internet network in FPT University Hoa Lac campus, the demo system for our project is limit in two rooms in a typical house.

* HSS-Board using WLAN or Ethernet to connect to Internet
* HSS-Board and smartphone are in local network but both connected well to Internet
* HSS-Board using 2 sensors (PIR sensor, wired door sensor) and a push button, both of them are connect to fixed digital PIN on Arduino board
* System has login mechanism to initiate connection between server and client application
* System receives and sends notifications in real-time, with maximum 5s delay due to speed of Internet connection between HSS-board and smartphone
* Audio quality of the call depends on quality of Internet connection in both user’s house and smartphone
* Image capturing quality is 640x480 pixel

# CHAPTER 2 – PROJECT MANAGEMENT PLAN

## 2.1. Tools and Software

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| Operating System | Ubuntu 14.04 |  |
| Windows 10 |  |
| Programming Language | Python |  |
| C++ |  |
| Java |  |
| Library and API | Liblinphone |  |
| Pusher API |  |
| Support Tool and Service | Qt | Development tool |
| Github | Source code management tool |
| Google Drive | Document management tool |
| Microsoft Office Tools | Write and edit document tool |
| Astah Community | Drawing tool |
| Fritzing | Draw physical prototyping to actual product |
| Android Studio | Official IDE for Android app development |
| Arduino Software | Writing code tool |
| VMWare Workstation 10 | Test-and-development environment |

**Library Liblinphone**:

Liblinphone is a high level library integrating all SIP video calls feature into a single easy to use API. Usually telecommunications is made of two things: media (transport of voice or video, encoding and decoding...), and signaling (routing calls, ringing, accepting a call etc...).

Liblinphone aims at combining the two things together and doing most things automatically. This makes it easier to the programmer to implement video calls in any application, without being an expert in VoIP and telecommunications. Liblinphone is an open source library based on Mediastreamer2 for voice/video streaming, and belle-sip for SIP signaling [[[3]](#footnote-3)].

**Pusher API:**

Pusher is a simple hosted API for quickly, easily and securely adding real-time bi-directional functionality via WebSockets to web and mobile apps, or any other Internet connected device. Pusher offers a rich suite of libraries that user can use within applications, including a JavaScript client library for web and HTML5 apps. Its event based abstraction makes it simple to bind UI interactions to events that are triggered from any client or server. It uses WebSockets (with 'fallback to Flash' in the JavaScript client library) to future proof user' applications and make it easy for user to add bi-directional communication to apps whilst keeping data usage to a minimum [[[4]](#footnote-4)].

## 2.3. Risk Management Plan

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Risk Name** | **Category** | **Description** | **Reason** | **Risk Owner** | **Impact** | **Response** |
| 1 | Over deadline | People | Did not submit task on time that it must be completed by | Didn’t work as planned, sickness or lack of attitude | Members | Very high | Leader had some penalties for all members |
| 2 | Conflict ideas | People | Conflict ideas or solution with another member | Poor communication, difference in personalities or values | All | High | Ask our lecture for ideas and stop making conflict |
| 3 | Cannot reach project scope | Technical | Could not reach project scope as our expectations | We couldn’t understand ourselves and our abilities, so think bigger but do smaller | All | Very high | Try our best to reach our maximum abilities |
| 4 | Unrealistic project schedule | Strategic | Unrealistic project schedule | Couldn’t manage well the schedule and did not understand members | Leader | High | Stay down and do the project schedule again |
| 5 | Hardware suddenly break | Technical | Hardware in comment (devices, components…) are suddenly break without any reason | Environment, weather, humidity or even quality of device… | N/I | Normal | This is objective reason so we could not handle it, due to limited project budget |
| 6 | Limited project budget | Technical | Could not get fully devices or components as the first scope overview and our expectations | Some devices and/or components are not cheap, so especially made some difficulties for us | All | High | Asking for borrow some components from friends and lecture |
| 7 | Working environment | Other | When internet connection is lost, all connection between Raspberry Pi and server lost, and can automatically connect again | Speed of internet in campus Hoa Lac is very slow. And login mechanism for using Internet in dormitory prevented us from working well with internet and Raspberry Pi | N/I | Very high | This is objective reason so we could not handle it |

## 2.4. Communication Plan

Communication must circulate through the project management leader and external stakeholders to maintain the health and the viability of the project. The existence of a documented communication plan does not by itself guarantee good communication during the project planning stage. Attention must be given to the type of information that is required at each level of the project and take steps to make sure that information is freely transmitted.

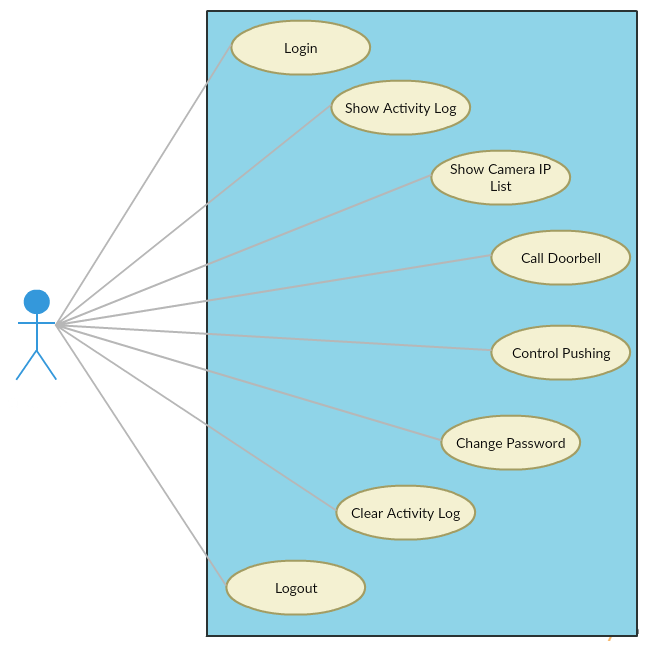
* Weekly offline meeting: Monday’s night
* Weekly meeting with supervisor: Thursday or Friday
* Un-scheduled meeting: when necessary and urgent
* Online discussing:
  + Facebook Group Chat
  + Facebook Group
* Synchronize/Support Tool: Google Drive, GitHub
* Mobile phone: when necessary and urgent

# CHAPTER 3 – SYSTEM REQUIREMENT SPECIFICATION

## 3.1. Overview

It describes design constraints, and other factors necessary to provide a complete and comprehensive description of the requirements of the system. All members will work (design, code, test) based on information provided in this chapter.

## 3.2. Functional Requirements



*Figure 5: Functional Requirements*

## 3.3. Non-Functional Requirements

### 3.3.1. Security

Security is a critical non-functional requirement of HSS. It is included in HSS to ensure all of unauthorised access to the system and its data on database is not allowed

### 3.3.2. Reliability

Reliability is the ability of a system to perform its required functions under stated conditions for a specific period of time. Can be considered under two separate headings:

* Availability - is the system available for service when requested by end-users.
* Low Failure Rate - low frequency of failing to deliver the functional requirements as expected by end-users.

### 3.3.3. Safety

The supply voltage of hardware is about 5V and totally be safe with user

### 3.3.4. Usability

Usability is the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of system or component

Usability requirements include:

* Well-structured functional requirement
* Informative error messages
* Well-formed graphical user interfaces

## 3.4. HSS-App Use Cases

### 3.4.1. Login

|  |  |  |  |
| --- | --- | --- | --- |
| **UC ID and Name** | **UC-1: Login** | | |
| **Created By:** | **CuongTV** | Date Created: | **2/15/2016** |
| **Primary Actor** | **User** | Secondary Actors: | **N/A** |
| **Trigger:** | Trigger when first login to Android application for authenticating | | |
| **Description:** | **- User** logins in the first time to application to access the system  **- User** uses a username and password that stored in database on Raspberry Pi before for confirming the access | | |
| **Preconditions:** | PRE-1. User is on login application screen | | |
| **Post Conditions:** | POST-1. Display home screen | | |
| **Normal Flow:** | 1. Actor enters username and password in text view “Username” field and “Password” field 2. Actor pushes “Login” button. 3. After logging in successfully then go to the Home screen | | |
| **Alternative Flows:** | N/A | | |
| **Exceptions:** | **E1 – Blank Textbox**   * System detects whether or not user entered the username or password into text field * If the text box is empty, show the error message in small box: “Login Failed”   **E1 – Invalid Username or Password**   * System indicates that the username or password entered is not valid * A message “Login Failed” will appear | | |
| **Priority:** | High | | |
| **Frequency of Use:** | Low | | |

### 3.4.2. Show Activity Log

|  |  |  |  |
| --- | --- | --- | --- |
| **UC ID and Name** | **UC-2: Show Activity Log** | | |
| **Created By:** | **CuongTV** | Date Created: | **2/15/2016** |
| **Primary Actor** | **User** | Secondary Actors: | **N/A** |
| **Trigger:** | Pushing the “Activity Log” button | | |
| **Description:** | **- User** pushes “Acitivity Log” button on home screen  **- The Application** shows on screen the list of activities, from today back to the very beginning or the last clear-time | | |
| **Preconditions:** | PRE-1. User is on home application screen | | |
| **Post Conditions:** | POST-1. Display activity log screen | | |
| **Normal Flow:** | * Actor pushes the “Activity Log” button on home screen * After pushing, then go to the Activity Log screen | | |
| **Alternative Flows:** | N/A | | |
| **Exceptions:** | N/A | | |
| **Priority:** | High | | |
| **Frequency of Use:** | High | | |

### 3.4.3. Show Camera IP List

|  |  |  |  |
| --- | --- | --- | --- |
| **UC ID and Name** | **UC-3: Show Camera IP List** | | |
| **Created By:** | **CuongTV** | Date Created: | **2/15/2016** |
| **Primary Actor** | **User** | Secondary Actors: | **N/A** |
| **Trigger:** | Pushing “Camera” button | | |
| **Description:** | **- User** pushes “Camera” button on home screen  **- The Application** shows on screen the list of working cameras in your system  - **User** can choose a particular camera for watching purpose | | |
| **Preconditions:** | PRE-1. User is on home application screen | | |
| **Post Conditions:** | POST-1. Display camera screen | | |
| **Normal Flow:** | * Actor pushes the “Camera” button on home screen * After pushing, then go to the Camera IP List screen | | |
| **Alternative Flows:** | N/A | | |
| **Exceptions:** | N/A | | |
| **Priority:** | High | | |
| **Frequency of Use:** | High | | |

### 3.4.4. Call Doorbell

|  |  |  |  |
| --- | --- | --- | --- |
| **UC ID and Name** | **UC-4: Call doorbell** | | |
| **Created By:** | **CuongTV** | Date Created: | **2/15/2016** |
| **Primary Actor** | **User** | Secondary Actors: | **System** |
| **Trigger:** | Pushing “Call Doorbell” button | | |
| **Description:** | **User** can make a call from smartphone to the doorbell system to communicate with guest outside your home | | |
| **Preconditions:** | PRE-1. User is on home application screen | | |
| **Post Conditions:** | POST-1. Display Calling screen | | |
| **Normal Flow:** | * Actor pushes the “Door Bell” button on home screen * After pushing, then go to the Calling screen * Actor pushes the “Start Call” button on Calling screen * Actor pushes the “End Call” button on Calling screen to end the call | | |
| **Alternative Flows:** | N/A | | |
| **Exceptions:** | N/A | | |
| **Priority:** | High | | |
| **Frequency of Use:** | High | | |

### 3.4.5. Control Pushing Sensors

|  |  |  |  |
| --- | --- | --- | --- |
| **UC ID and Name** | **UC-5: Control Pushing Sensors** | | |
| **Created By:** | **CuongTV** | Date Created: | **2/15/2016** |
| **Primary Actor** | **User** | Secondary Actors: | **N/A** |
| **Trigger:** | Touching on switch button | | |
| **Description:** | **User** can activate/de-activate sensors to receiving/stop receiving real time notifications | | |
| **Preconditions:** | PRE-1. User is on Setting screen | | |
| **Post Conditions:** | POST-1. The switch button change from On to Off or vice versa | | |
| **Normal Flow:** | * User pushes “Setting” button on home screen * After pushing, then go to Setting screen * User touch the switch button on screen at each sensor field to activate/deactivate pushing notification | | |
| **Alternative Flows:** | N/A | | |
| **Exceptions:** | N/A | | |
| **Priority:** | Medium | | |
| **Frequency of Use:** | Medium | | |

### 3.4.6. Change Password

|  |  |  |  |
| --- | --- | --- | --- |
| **UC ID and Name** | **UC-6: Change Password** | | |
| **Created By:** | **CuongTV** | Date Created: | **2/15/2016** |
| **Primary Actor** | **User** | Secondary Actors: | **N/A** |
| **Trigger:** | Pushing “Change Pwd” button on setting screen | | |
| **Description:** | **User** can change password for username when logging in | | |
| **Preconditions:** | PRE-1. User is on setting screen of HSS-App | | |
| **Post Conditions:** | POST-1. Display Login screen | | |
| **Normal Flow:** | * Actor pushes “Setting” button on home screen * Actor pushes “Change Pwd” button on Setting screen * After pushing, then go to the Change Password screen | | |
| **Alternative Flows:** | N/A | | |
| **Exceptions:** | N/A | | |
| **Priority:** | Medium | | |
| **Frequency of Use:** | Low | | |

### 3.4.7. Clear Activity Log

|  |  |  |  |
| --- | --- | --- | --- |
| **UC ID and Name** | **UC-7: Clear Activity Log** | | |
| **Created By:** | **CuongTV** | Date Created: | **2/15/2016** |
| **Primary Actor** | **User** | Secondary Actors: | **N/A** |
| **Trigger:** | Pushing “Delete Activity Log” button in Settings screen | | |
| **Description:** | **User** can send the clear activity log command to clear its database on server | | |
| **Preconditions:** | PRE-1. User is on Settings screen | | |
| **Post Conditions:** | POST-1. Delete activity log on database and HSS-App | | |
| **Normal Flow:** | * User pushes “Setting” button on home screen * After pushing, then go to Setting screen * User pushes “Delete Activity Log” in Settings screen | | |
| **Alternative Flows:** | N/A | | |
| **Exceptions:** | N/A | | |
| **Priority:** | Medium | | |
| **Frequency of Use:** | Medium | | |

### 3.4.8. Logout

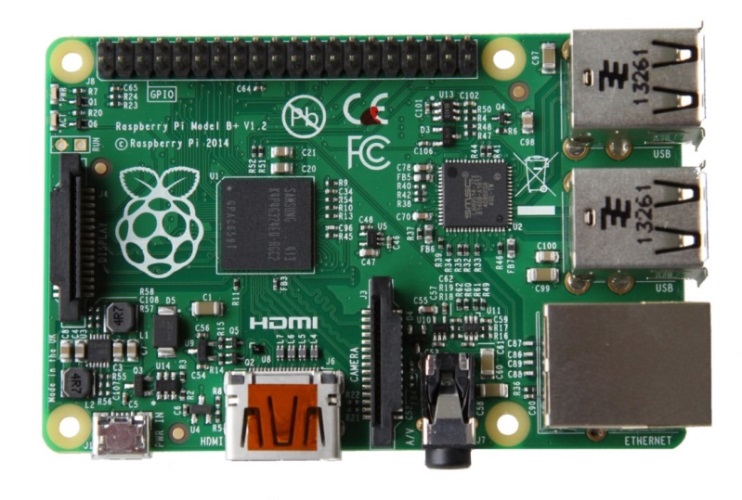
|  |  |  |  |
| --- | --- | --- | --- |
| **UC ID and Name** | **UC-8: Logout** | | |
| **Created By:** | **CuongTV** | Date Created: | **2/15/2016** |
| **Primary Actor** | **User** | Secondary Actors: | **N/A** |
| **Trigger:** | Trigger when user pushing the “Logout” button in Setting screen | | |
| **Description:** | **- User** can logout from the application when un-neccessary to prevent from unwanted security issues | | |
| **Preconditions:** | PRE-1. User is on Settings screen | | |
| **Post Conditions:** | POST-1. Display Login screen | | |
| **Normal Flow:** | * User pushes “Settings” button on home screen * After pushing, then go to Settings screen * User pushes “Logout” in Settings screen | | |
| **Alternative Flows:** | N/A | | |
| **Exceptions:** | N/A | | |
| **Priority:** | High | | |
| **Frequency of Use:** | Low | | |

# CHAPTER 4 – SYSTEM DESIGN DESCRIPTION

## 4.1. Hardware Components

### 4.1.1. Raspberry Pi 2

The Raspberry Pi is a series of credit card–sized single-board computers developed in England, United Kingdom by the Raspberry Pi Foundation with the intent to promote the teaching of basic computer science in schools and developing countries. We use Raspberry Pi 2 as a server (or a hub) or whole system.

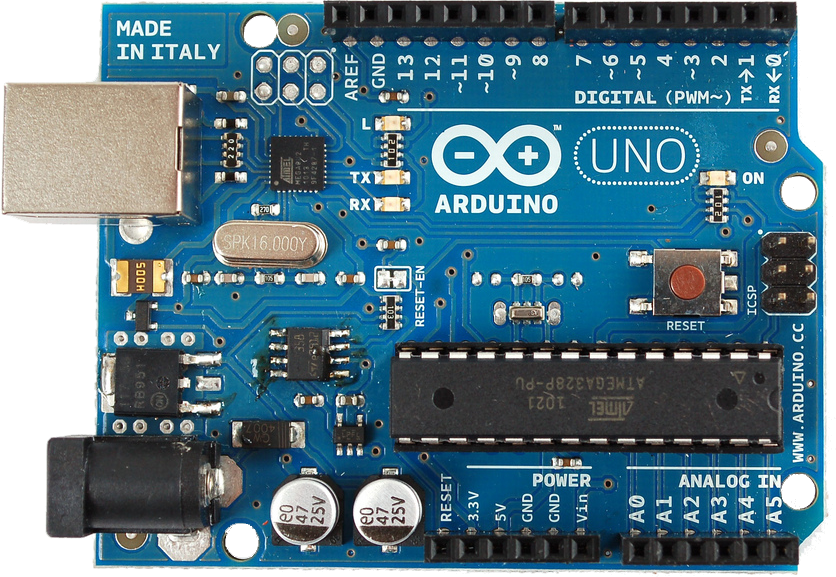


*Figure 6: Raspberry Pi 2 board*

|  |  |
| --- | --- |
| *CHARACTERISTICS* | *VALUE* |
| SoC | Broadcom BCM2835 |
| CPU | 700 MHz single-core ARM1176JZF-S |
| CPU | BCM2837: 3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz |
| Memory (SDRAM) | 512 MB (shared with GPU) |
| USB 2/0 Ports | 4 |
| Audio Output | Analog via 3.5 mm phone jack; digital via HDMI |
| On-board Network | 10/100 Mbit/s Ethernet (8P8C) or USB hub |
| Power | 600 mA (3.0 W) |

### 4.1.2. Arduino Uno

The Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. We use Arduino for receiving signal on digital PIN from sensors and button.

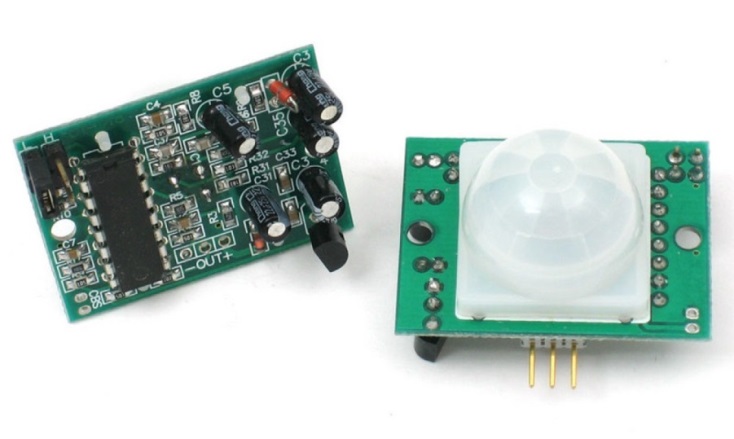


*Figure 7: Arduino UNO board*

|  |  |
| --- | --- |
| *CHARACTERISTICS* | *VALUE* |
| Microcontroller | ATmega328P |
| Operating Voltage | 5 V |
| Input Voltage (recommended) | 7-12 V |
| Input Voltage (limits) | 6-20 V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| Analog Input Pins | 6 |

### 4.1.3. Sensors

A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors.All objects with a temperature above absolute zero emit heat energy in the form of radiation. Usually this radiation is invisible to the human eye because it radiates at infrared wavelengths, but it can be detected by electronic devices designed for such a purpose.



*Figure 8: PIR sensor*

|  |  |
| --- | --- |
| *CHARACTERISTICS* | *VALUE* |
| SoC | BISS0001 PIR Chip |
| Output | Digital pulse high (3V) when triggered (motion detected) digital low when idle (no motion detected). Pulse lengths are determined by resistors and capacitors on the PCB and differ from sensor to sensor |
| Sensitive Range | up to 20 feet (6 meters) 110° x 70° detection range |
| Power Supply | 3V-9V input voltage, but 5V is ideal. |

We also use wired door sensor (based on reed switch inside, so you can also get reed switches that work the opposite way). The two contacts are normally snapped together. When you bring a magnet up to the switch, the lower contact is attracted to the magnet; the upper one is repelled, so the contacts split apart, opening the switch and breaking the circuit. Reed switches like this are called normally closed (NC) (normally switched on), and they switch off when you bring a magnet up to them.



*Figure 9: Wired Door Sensor (Reed Switch inside)*

### 4.1.4. Support Components

Default Raspberry Pi can only access your home network using a network cable. as a hard one to get working on Raspberry Pi (even Raspberry Pi B+ 2014 v1.2) But let’s face it; your Raspberry Pi project is not always going to be deployed close to a network outlet – so what should we do? The best solution is to buy a cheap USB Wi-Fi adapter and use one of the USB ports to access our wireless home network. TL-WN725N V2 USB Adapter is the best choice for this issue. It is known

****

*Figure 11: USB Wifi TP-Link 725N*

|  |  |
| --- | --- |
| *CHARACTERISTICS* | *VALUE* |
| Interface | USB 2.0 |
| Antenna | Internal antenna |
| LED | Status |

In addition, a block of webcam, microphone and a speaker are used for simulating smart doorbell feature. Raspberry support many kinds of camera that connected to USB port without driver. Speaker and microphone was connect via 3.5 audio in/out jack.

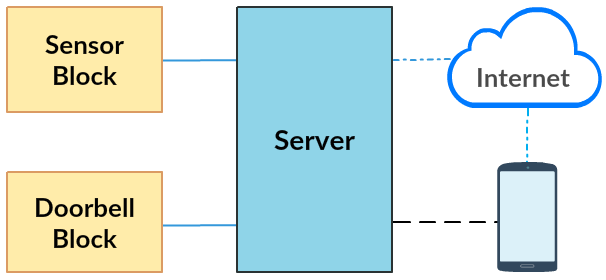
  
*Figure 12: Speaker and webcam with built-in microphone*

A push button is used as a doorbell button:

  
*Figure 12: Push button is used as doorbell button*

## 4.2. System Architecture Overview

### 4.1.1. System Architecture



*Figure 13: System Architecture Overview*

**Note:**

Server: Raspberry Pi 2

Sensor Block: Arduino Uno, PIR sensor, reed switch, push button

Doorbell Block: Microphone, Webcam, Speaker

Internet: Internet and third-party services on cloud

 Wired Connection

 Wireless Communication (Outside network)

 Wireless Communication (Local network)

### 4.1.2. Architectural Explanation

HSS’s system architecture has 5 parts above: server, sensor block, doorbell block, internet and mobile application. And we divided them into 2 main parts: **HSS-Board** (whose devices are being connected together by wired connection includes Raspberry Pi server) and **HSS-App** (which includes Android application and other party services connected through Internet).

In **HSS-App**: Via WLAN that was launched by your existing router in house, Android application will connected to the Internet and start connecting to your server on Raspberry Pi via TCP socket (In this project, because the lack of using public internet and not having root access to router, we use local network for TCP connection between HSS-App and server). Then, user should login to server by entering username and password which stored in server’s database. The login process is only for recognizing between server and user application. From now on, application can be used without depending on server. In this Android app, you can manage whole system or sensors by activating or deactivating them, one-by-one or all.

When developing this solution, as mentioned before, we want to put camera IP to right place it would be. We only consider camera IP is just third-party service to help users figuring out what’s going on in their house. In mobile application, we add a feature to help user what their camera IP via existing URL (as long as all of their cameras had been deployed and set up port-forwarding in existing router).

In **HSS-Board**, sensors are connected to digital pins on Arduino, and Arduino is connected to Raspberry Pi via UART USB cable. When signal is detected on each sensor, data was sent directly to Raspberry Pi through Arduino board.

We did setup a server on Raspberry Pi. When be received data from Arduino, it will push notification in real-time to third-party service for pushing messages to Android client application. In HSS, we use Pusher as third-party service to do this. Are you worried about this service? It’s not needed! Pusher provide a secure mechanism for controlling who has access to given channels, seamlessly integrating with existing authentication strategies.

  
*Figure 14: Example of pushing notification in real-time*

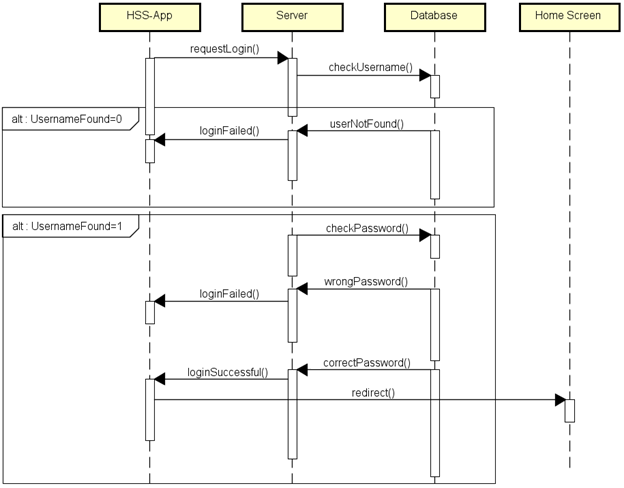
It’s pushing real-time notification. Now we move to smart doorbell feature of HSS. We put its button (as a doorbell button in real life) in sensor block; button’s role is when someone pushing it, a notification will be sent immediately to your Android application, providing your mobile device is being connected to Internet (Wifi or cellular). Once user receiving the notification, they can take action now: ignoring it or opening Android application and see what’s going on in front of their door in real-time). In addition, you also can speak out to the door with the guest to talk with them. Microphone, camera and speaker in your doorbell will help you to do this well. This feature is based on VOIP technology. In real life, you can make a call to your door bell anytime you want, not depending on someone have to push doorbell button. The button is just for sending a notification to HSS-App.



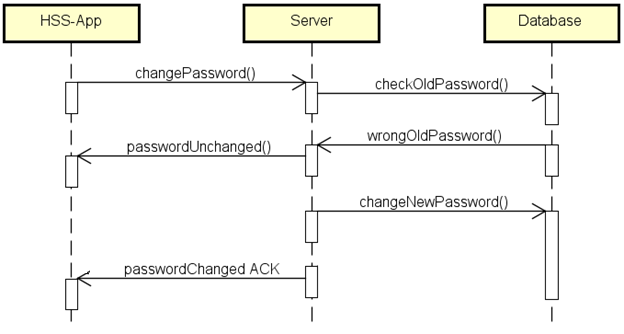
*Figure 15: Doorbell Block*

### 4.1.3. Sequence Diagram

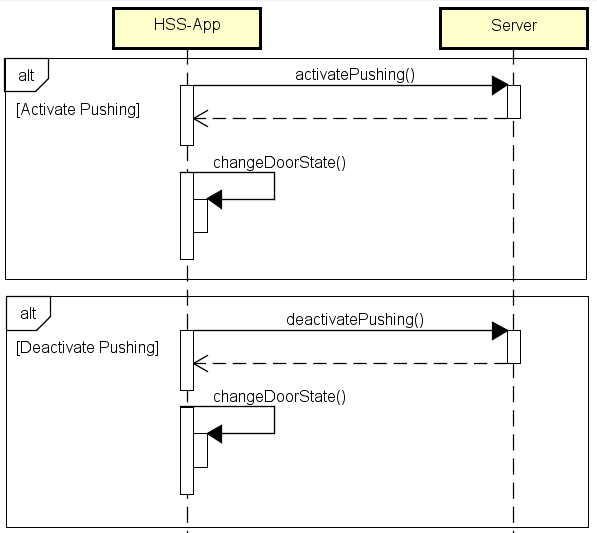
**4.1.4.1. Login**



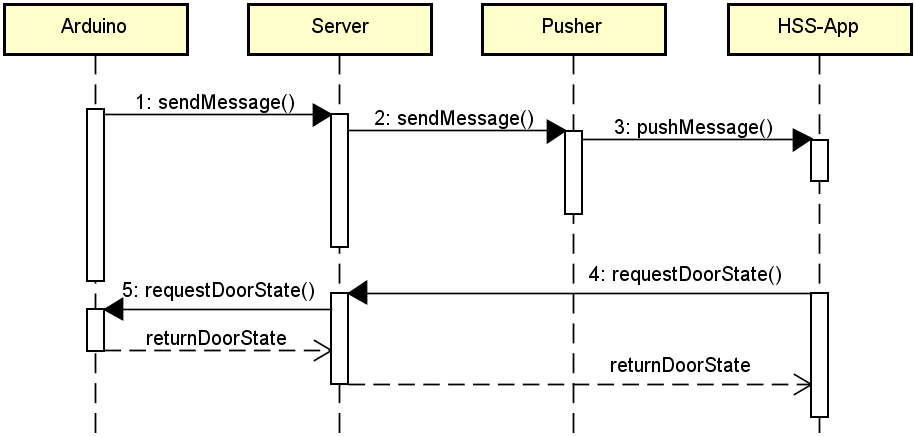
**4.1.4.2. Change Password**



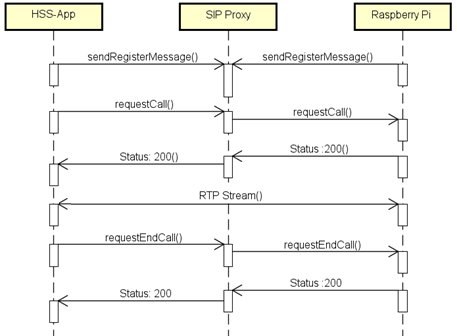
**4.1.4.3. Activate/Deactivate sensor**



**4.1.4.4. Pushing notifications**



**4.1.4.5. Call to doorbell**



## 4.3. Hardware design

### 4.3.1. System Schematic

The speaker for doorbell is connected to server via generic 3.5mm jack

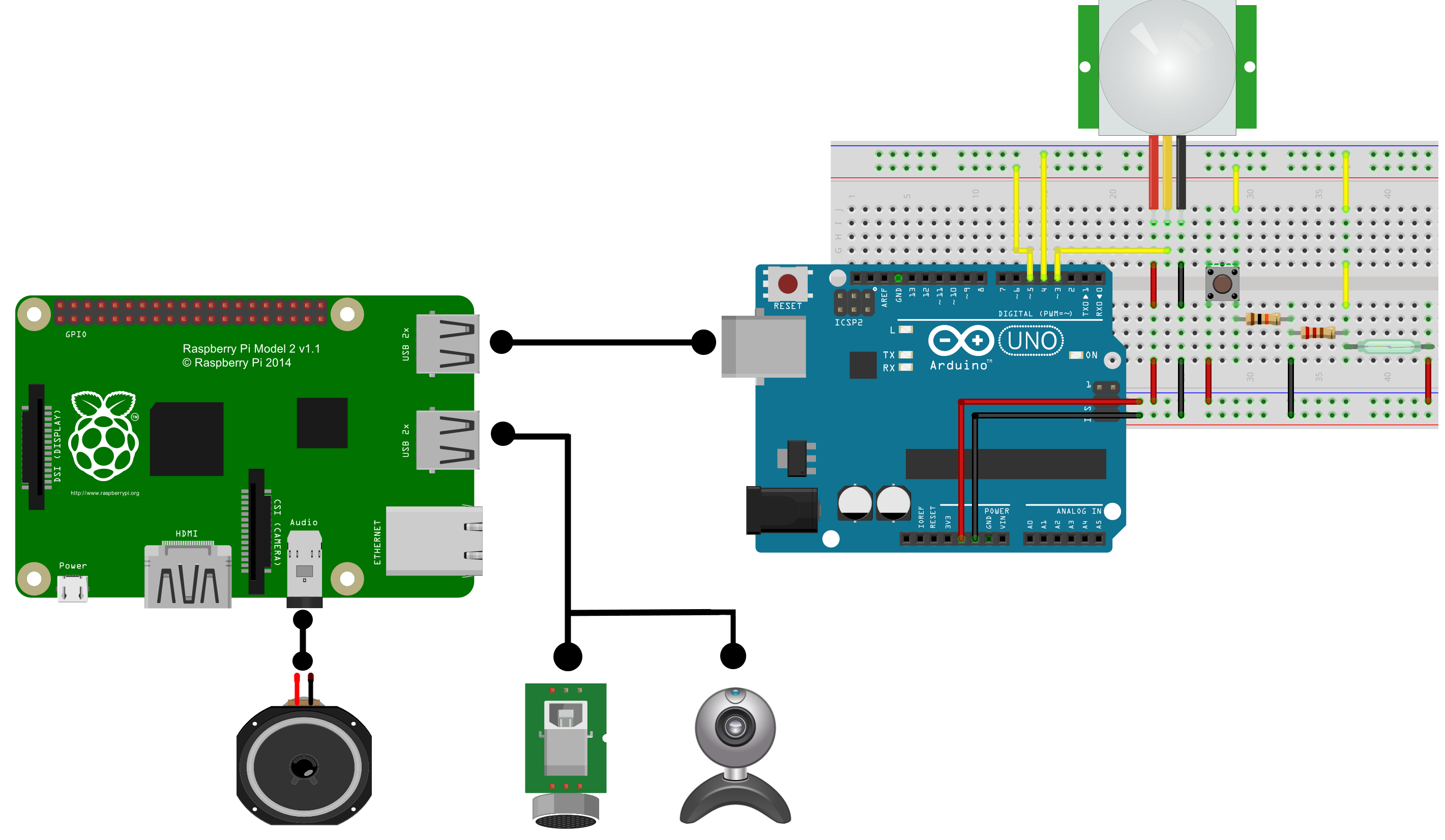
The microphone and webcam for doorbell are connected to server via USB

We connect Arduino board to Raspberry Pi by USB Serial cable. All sensors and button are connected to fixed digital PIN on Arduino following below:

PIR\_PIN: 3

DOOR\_PIN: 4

BUTTON\_PIN: 5

**

*Figure 16: HSS-Board Connection Schematic*

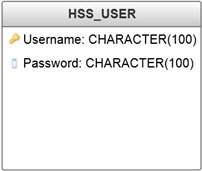
### 4.3.2. After Mounting on Demo Box

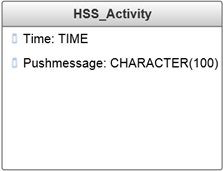
*<ảnh thật>  
Figure 17: HSS Board after mounting*

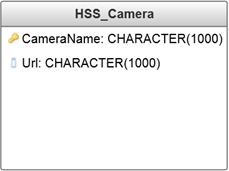
## 4.4. Embedded and HSS-App Design

### 4.4.1. Database

Database is used for storing data on server; include user information (username and password), activity log of sensor and list of camera IPs. We choose SQLite as database management system because it’s written in a C programming library. In addition, SQLite is a popular choice as embedded database software for local/client storage in application software.

*  
Figure 18: HSS\_User table*

*  
Figure 19: HSS\_Activity table*

*  
Figure 20: HSS\_Camera table*

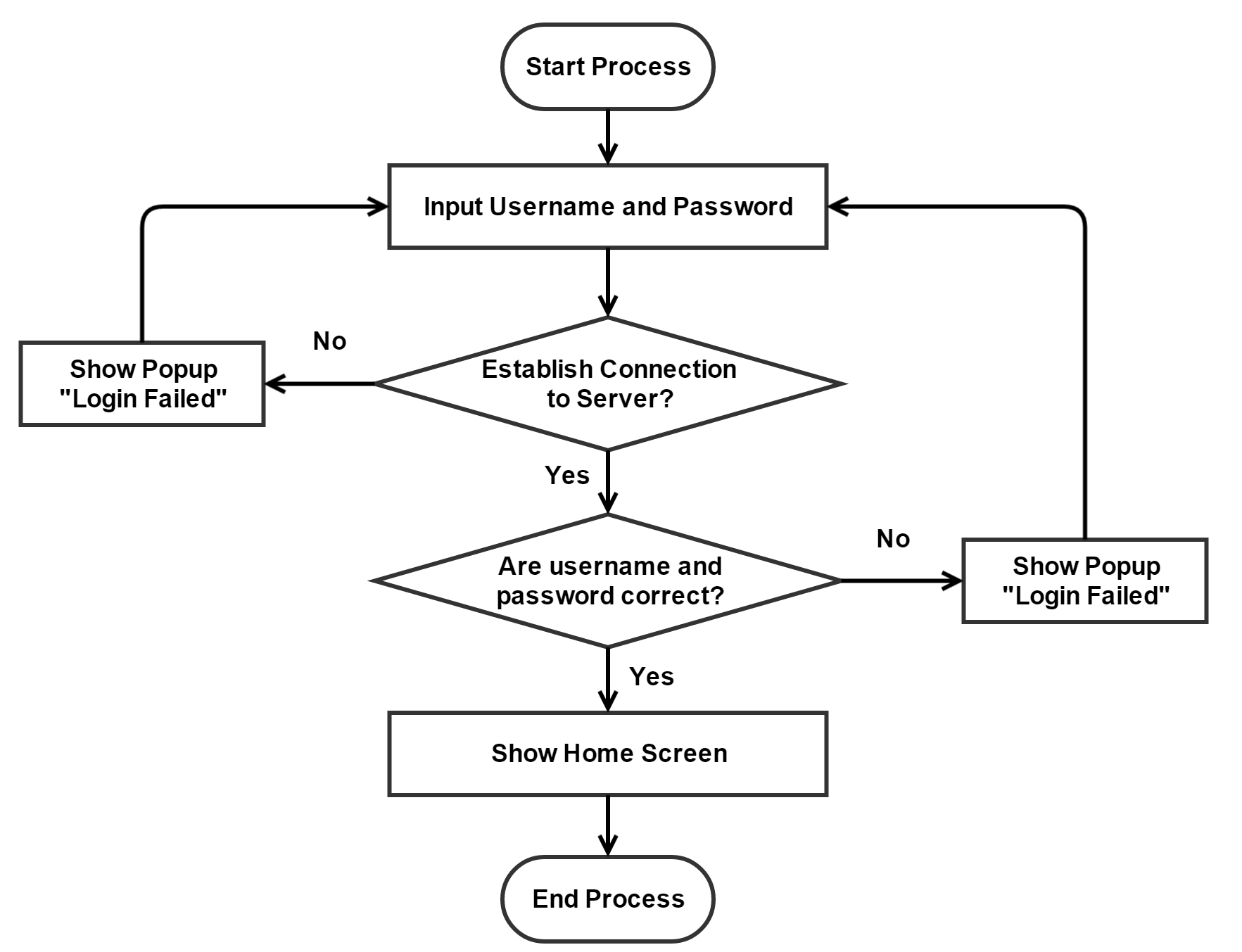
Database Explain:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Meaning** |
| Username | CHARACTER(100) | Username for login |
| Password | CHARACTER(100) | Password for login |
| Time | TIME | Time of triggered event |
| Pushmessage | CHARACTER(100) | Message shown in Activity Log screen |
| CameraName | CHARACTER(1000) | Name of Camera IP |
| Url | CHARACTER(1000) | URL of Camera IP |

**How to change data in database?**

Within the scope of HSS project, we don’t accept changing username in table HSS\_User and all in table HSS\_Camera. User can change password from Setting screen. To table HSS\_Activity, all new events triggered will be added automatically to this table and show in Activity Log screen on HSS-App. User can request on HSS-App to clear activity log in database.

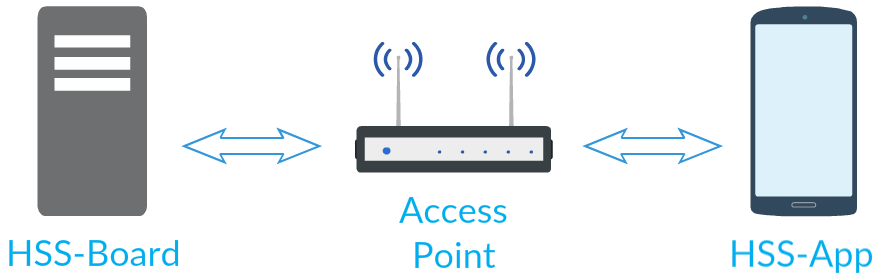
### 4.4.2. Login

**

*Figure 21: Login flowchart*

Every time user touching HSS-App icon on smartphone screen, HSS-App requires username and password to login. They are stored in database. First, HSS-Board is assigned a static IP in local network, and Android application has this IP address (in real life we set it fixed in HSS-App). To connect with HSS-Board in order to receive notifications and using other functions, every time user start application or open it again (assume that it’s running in background), a login screen will appear and require username and password. The database for storing username – password is on HSS-Board.

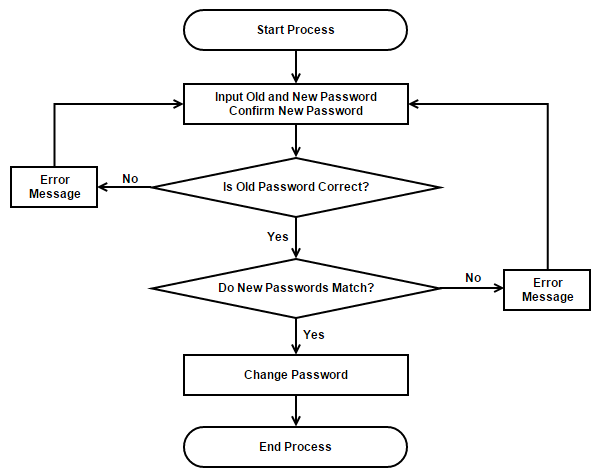
HSS-App will establish a TCP/IP socket connection to server and do the remote procedure call function in order to check whether user and password are correct.

  
*Figure 22: The system in local network*

If successful logging, HSS-App will show the Home Screen. Otherwise, a login failed message will appear on screen. The success of this process allows HSS-App can retrieve data from database, includes activity log and list of camera IP.

### 4.4.3. Change Password

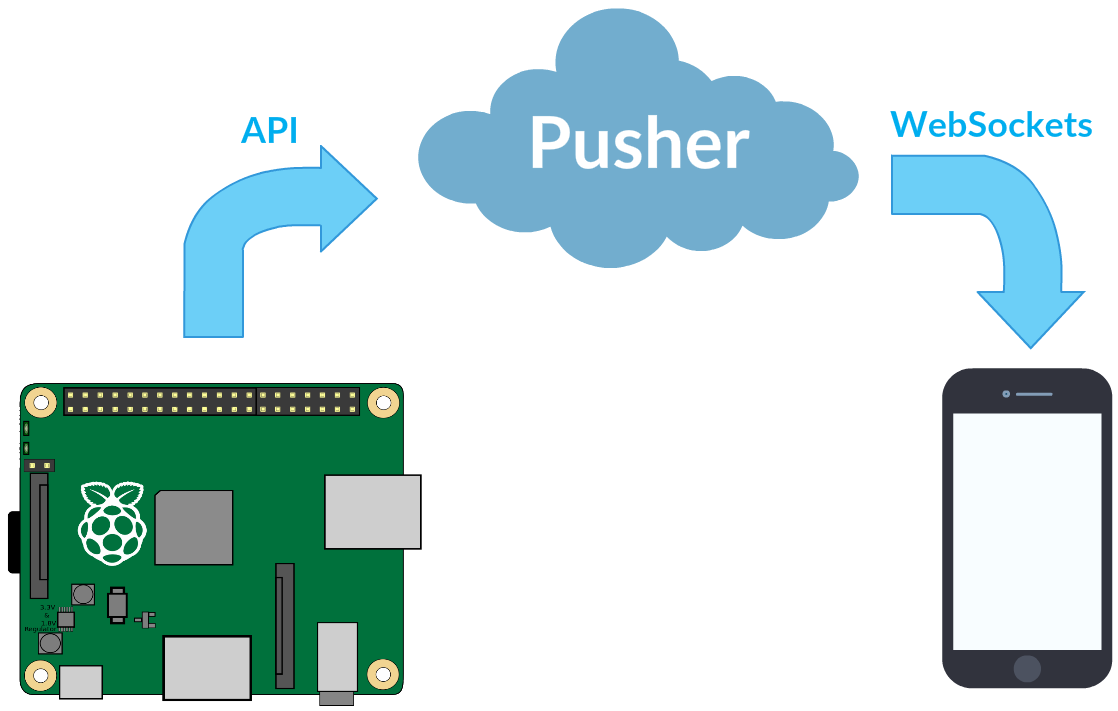
For security reason, user can request to change password anytime. Changing password process is shown below:



*Figure 23: Changing password flowchart*

### 4.4.3. Pushing Notification

WebSockets represents a long awaited evolution in client/server web technology. They allow a long-held single TCP socket connection to be established between the client and server which allows for bi-directional, full duplex, messages to be instantly distributed with little overhead resulting in a very low latency connection.

*  
Figure 24: How Pusher works in HSS*

As mentioned before, we use third-party service on cloud called Pusher to push message to Android application in real-time technology when any event triggered on Raspberry side. The mechanism of Pusher is based on WebSockets that allow a long-held single TCP socket connection to be established between the client and server for bi-directional, full duplex, messages to be instantly distributed with little overhead resulting in a very low latency connection The process of pushing notification is explained in detail below:

**On Arduino:**

When Arduino triggers an event, it will use Remote Procedure Call (RPC) to return the sensor’s value to server. RPC is a powerful technique for constructing distributed, client-server based applications. It is based on extending the notion of conventional or local procedure calling, so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them. Both Arduino and server have the same RPC function declaration.

**On Raspberry Pi server:**

RPC server function will parse the messages from RPC client function (Arduino) and call corresponding function (read the value of sensors)

Then, server will push the message Pusher. For testing purposes we’ll trigger an event called PIR on a channel called *Channel* (we can modify corresponding channel for every system in the future, but in this project we set it fixed – it’s called public channel). We’ll give this event an arbitrary payload, such as *{message: 'Motion detected'}*. HSS\_AppID, HSS\_AppKey and HSS\_AppSecret are provided on Pusher web server after registering. Python script on server Raspberry Pi:

|  |  |
| --- | --- |
| *1.* | import pusher |
| *2.* | p = pusher.Pusher( |
| *3.* | app\_id='HSS\_AppID', |
| *4.* | key='HSS\_AppKey', |
| *5.* | secret='HSS\_AppSecret') |
| *6.* | p.trigger('Channel','PIR', {'message': 'Motion detected'}) |

**On HSS-App client:**

**Step 1: Get API keys from Pusher server**

Create an account on Pusher and make note of app\_id, app\_key and app\_secret

**Step 2: Install the library**

One of the library’s dependencies is hosted on Clojars.org so you will have to add it to your repositories list. In your build.gradle script for your application module, add the library as a dependency:

|  |  |
| --- | --- |
| *1.* | repositories { |
| *2.* | maven { url 'http://clojars.org/repo' } |
| *3.* | } |
| *4.* | dependencies { |
| *5.* | compile 'com.pusher:pusher-java-client:0.3.3' |
| *6.* | } |

The pusher-websocket-java library is also available in Maven Central. Add a *dependency* element as a child of *dependencies* with the following in *pom.xml*:

|  |  |
| --- | --- |
| *1.* | <dependency> |
| *2.* | <groupId>com.pusher</groupId> |
| *3.* | <artifactId>pusher-java-client</artifactId> |
| *4.* | <version>0.3.3</version> |
| *5.* | </dependency> |

**Step 3: Create a connection**

|  |  |
| --- | --- |
| *1.* | import com.pusher.client.Pusher; |
| *2.* | … |
| *3.* | Pusher pusher = new Pusher("APP\_KEY"); |
| *4.* | pusher.connect(); |

**Step 4: Subscribe to a public channel**

|  |  |
| --- | --- |
| *1.* | import com.pusher.client.Pusher; |
| *2.* | Channel channel = pusher.subscribe("my-channel"); |

**Step 5: Listen for event**.

Once we have created an instance of *Channel*, we can set up event bindings

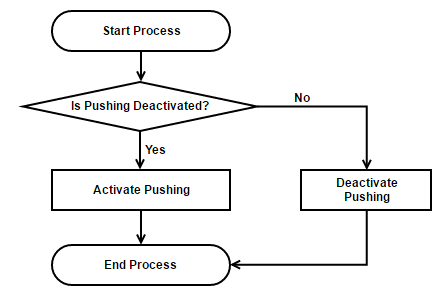
|  |  |
| --- | --- |
| *1.* | import com.pusher.client.Pusher; |
| *2.* | import com.pusher.client.channel.Channel; |
| *3.* | import com.pusher.client.channel.SubscriptionEventListener; |
| *4.* | Pusher pusher = new Pusher("HSS\_AppKey"); |
| *5.* | pusher.connect(); |
| *6.* | Channel channel = pusher.subscribe("Channel"); |
| *7.* | channel.bind("my-event", new SubscriptionEventListener() { |
| *8.* | @Override |
| *9.* | public void onEvent(String channelName, StringeventName, final String data) { |
| *10.* | System.out.println(data); |
| *11.* | } |
| *12.* | }); |

**Triggering Conditions:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sensor** | **Trigger** | **Push Action** | **Message** |
| Button | Press or hold once | Push to HSS-App once | “Someone press doorbell” |
| Door | Initial state: Closing | Show once on screen when change Door Switch state in Setting | “Door is being closed” |
| Initial state: Opening | Show once on screen when change Door Switch state in Setting | “Door is being opened” |
| Change state from closing to open | Push to HSS-App once | “Door has been opened” |
| Change state from open to close | Push to HSS-App once | “Door has been closed” |
| PIR | Detecting motion | Push to HSS-App once every 5s | “Motion detected” |
| Motion ends | Push to HSS-App after 5s without motion detected | “Motion ended” |

### 4.4.4. Control Pushing Notification

When users are at their home, they still want the wired door sensor is always activated and sends push message to HSS-App, but the PIR sensor. The best solution is deactivating a particular sensor in Setting screen. When event is triggered, it’s written to the database (activity log) as usual and user can view it in Activity Log screen later.

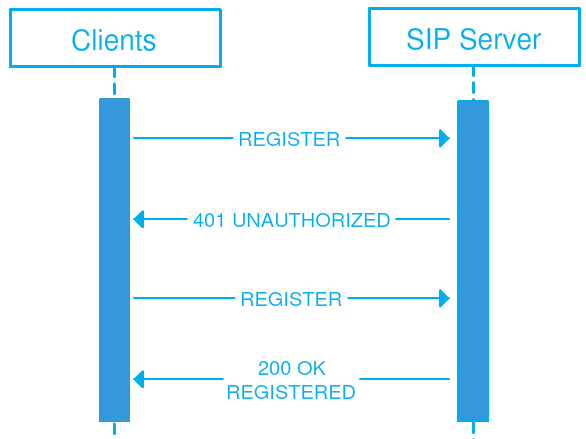
  
*Figure 25: Activate/Deactivate Pushing Notification*

### 4.4.5. Calling Doorbell

Calling doorbell is a feature that we developed based on VoIP. VoIP (Voice over IP) is a methodology and group of technologies for the delivery of voice communications and multimedia sessions over Internet Protocol (IP) networks, such as the Internet. Other terms commonly associated with VoIP are IP telephony, Internet telephony, broadband telephony, and broadband phone service.

A related term we use is SIP server (Session Initiation Protocol Server). SIP is a communications protocol for signaling and controlling multimedia communication sessions. The most common applications of SIP are in Internet telephony for voice and video calls, as well as instant messaging, over Internet Protocol (IP) networks. In HSS, a SIP server is responsible for keep registering with clients in both two sides: Raspberry Pi and HSS-App; then routing the connection between two sides when initiating a VoIP call. So, it controls channel for Voice over IP. In HSS, we choose Linphone as SIP server because of its stability. All connection from client to server like STUN protocol (a protocol that helps find the client in NAT network), registration signal to SIP server, etc… would be taken care by its server.

In a time, a control channel – known in the telephone world as a signaling channel – does call setup. It locates all the clients, determines if they’re available, asks the endpoint to alert the called party, passes back status to the caller, etc.

  
*Figure 26: Client to SIP Server with Register Message*

**On Raspberry Pi client:**

On Raspberry Pi, get the latest version of Linphone Python for Raspberry Pi from Linphone homepage (filename linphone4raspi.whl), then install some packages needed for it. In HSS, we use version 3.9.0-cp27. Finally, import Linphone library to Python shell.

|  |
| --- |
| $ sudo apt-get install python-setuptools |
| $ sudo easy\_intall pip |
| $ sudo apt-get install wheel |
| $ sudo pip install –upgrade pip |
| $ sudo pip install linphone4raspi.whl |
| $ python import linphone |

It’s almost done. Now we have to code a Python script to turn your Raspberry Pi into a VoIP client and auto answer call from user. Some attributes in Python program should be set below:

|  |  |
| --- | --- |
| *Parameter* | *VALUE* |
| SIP Server | sip.linphone.org |
| STUN Server | stun.linphone.org |
| Auto Answer Call | Yes |
| Audio Codec | PCMU or PCMA |
| Sound Input | ALSA: USB PnP Sound Device |
| Sound Output | Default |

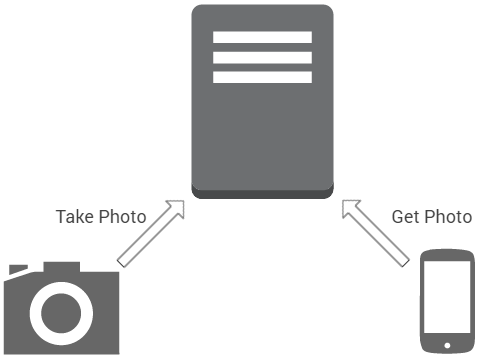
Raspberry Pi client will only receive call from user in white list for security purpose. According this, in real life, no one can access your doorbell except its owner.

**On HSS-App client:**

On HSS-App client, we use Linphone SDK for Android to build own application to connect to any SIP server – in HSS it is Linphone SIP server. The operating principle is similar with Raspberry Pi client but developed on Android Environment SDK.

### 4.4.6. Take and Load Photo

To take photo by the time the visitor pushing the doorbell button, we use fswebcam and Apache HTTP Server. Its operation is shown below:

  
*Figure 27: How taking photo process works*

**On Raspberry Pi:**

**Step 1: Install fswebcam package**

It’s a small and simple webcam app for Unix core. It can capture images from a number of different sources and perform simple manipulation on the captured image. The image can be saved as one or more PNG or JPEG files.

$ sudo apt-get install fswebcam

**Step 2: Install Apache HTTP Server**

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

$ sudo apt-get install apache2

**Step 3: Take photo and save to Apache server**

The command is shown below:

fswebcam -r 640x480 --no-banner /var/www/html/door.jpg -S

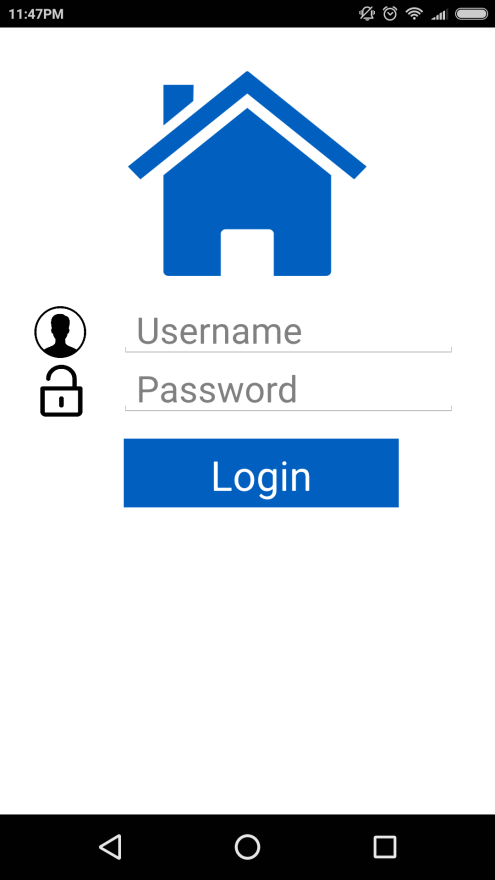
**On HSS-App:**

In order to load the taken photo from server, HSS-App accesses it via url:

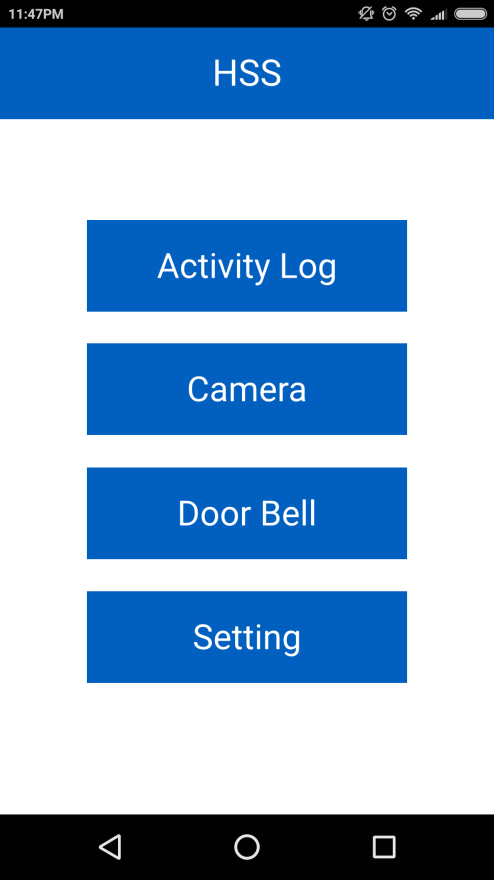
http://<IPAdress>/door.jpg

## 4.5. HSS-App User Interface Design

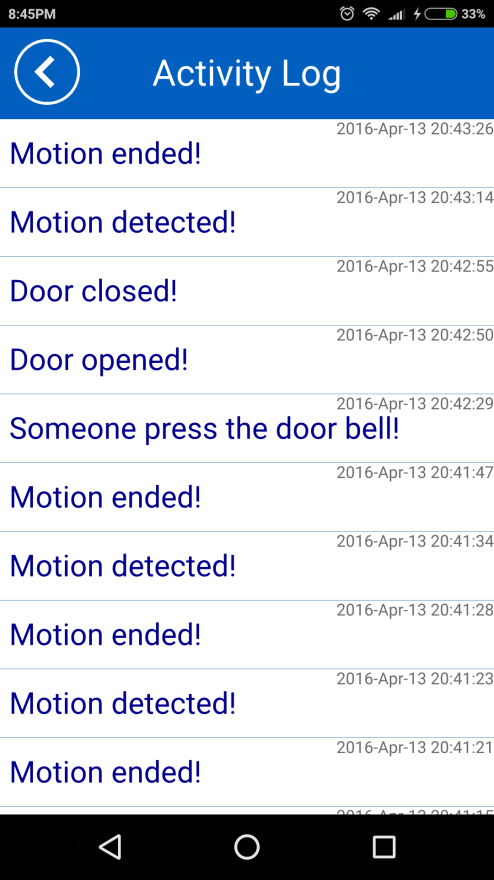
### 4.5.1. Login Screen

  
*Figure 28: Login Screen*

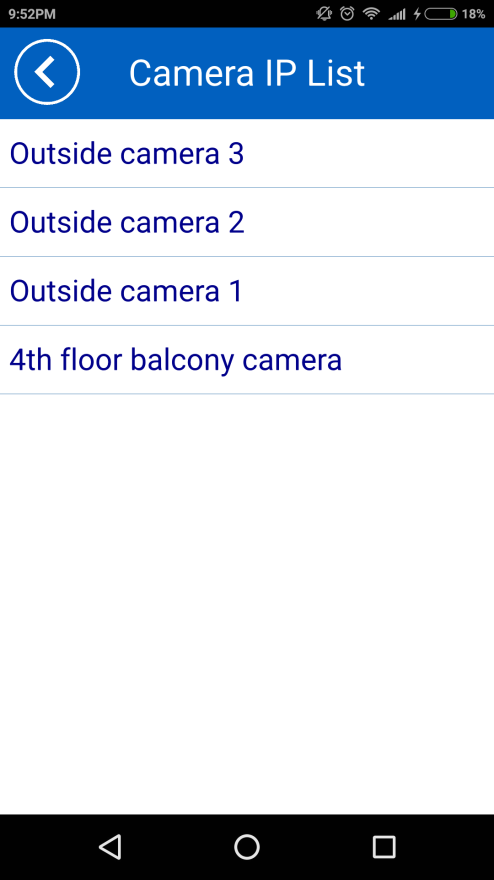
### 4.5.2. Home Screen

  
*Figure 29: Home screen*

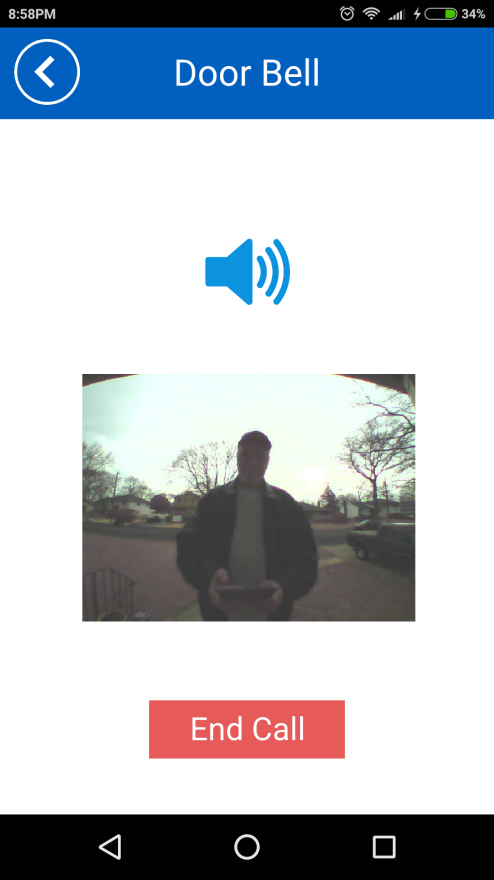
### 4.5.3. Activity Log Screen

  
*Figure 27: Activity Log Screen*

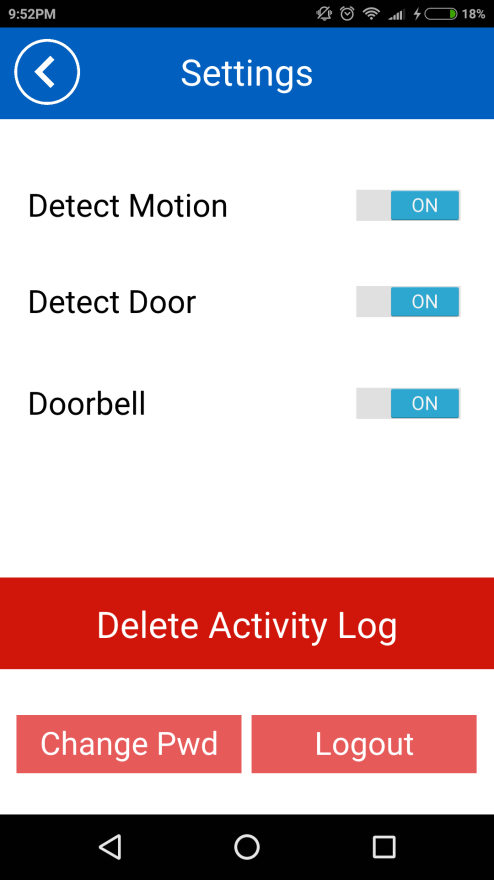
### 4.5.4. Camera IP Screen

  
*Figure 28: Camera IP Screen*

### 4.5.5. Doorbell Screen

*  
Figure 29: Doorbell Screen*

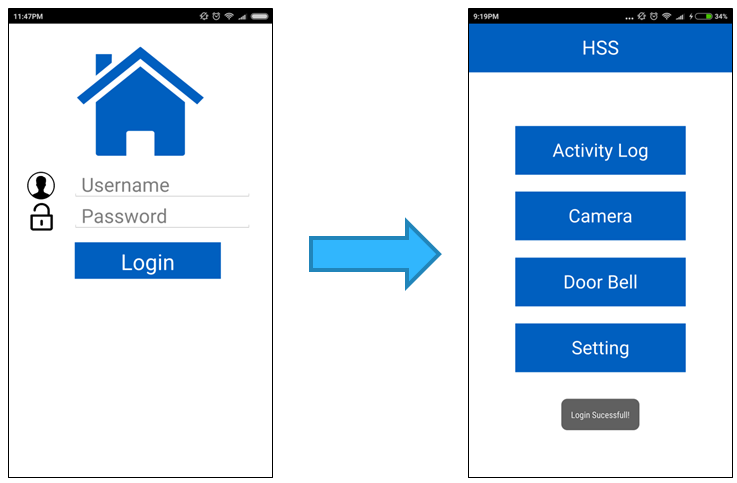
### 4.5.6. Setting Screen

*  
Figure 30: Setting Screen*

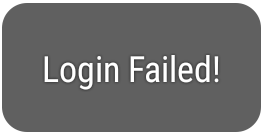
# CHAPTER 5 – IMPLEMENTATION

## 5.1. Login

After logging in successfully, application passes Login Screen and shows Home Screen and a message popup: “Login Successful”

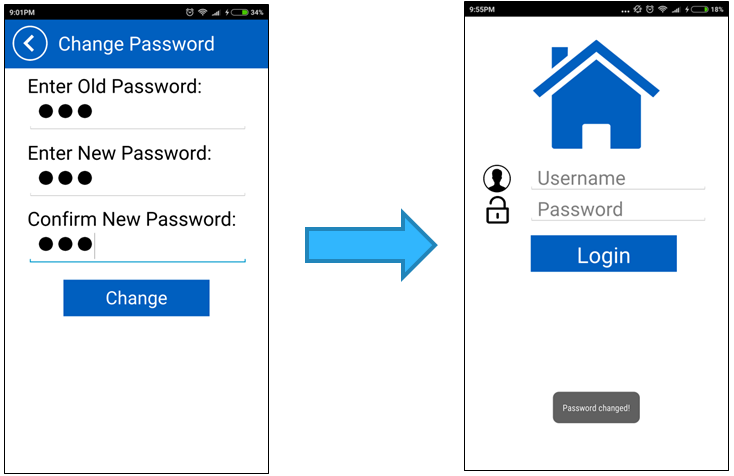


When login failed, a message popup “Login Failed” is shown on screen



## 5.2. Change Password

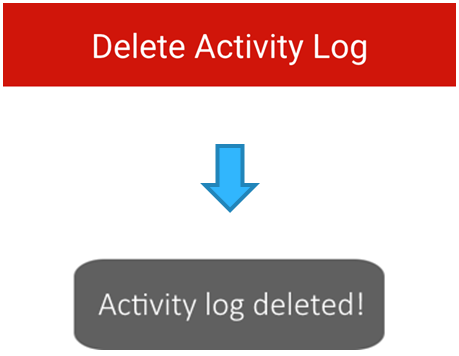
After changing password successfully, application will show Login screen with popup message “Password changed!”. Application requires username and new password to login again

  
*Figure : Changing password successfully*

When changing password process is fail, a message popup “Password unchanged!” is shown on screen. User is kept on Change Password screen.

## 5.3. Delete Activity Log

User click Delete Activity Log button in Setting Screen. A popup message will be shown: “Acvitity log deleted!” and Activity log screen will be empty from now on (synchronized with the data on the server).

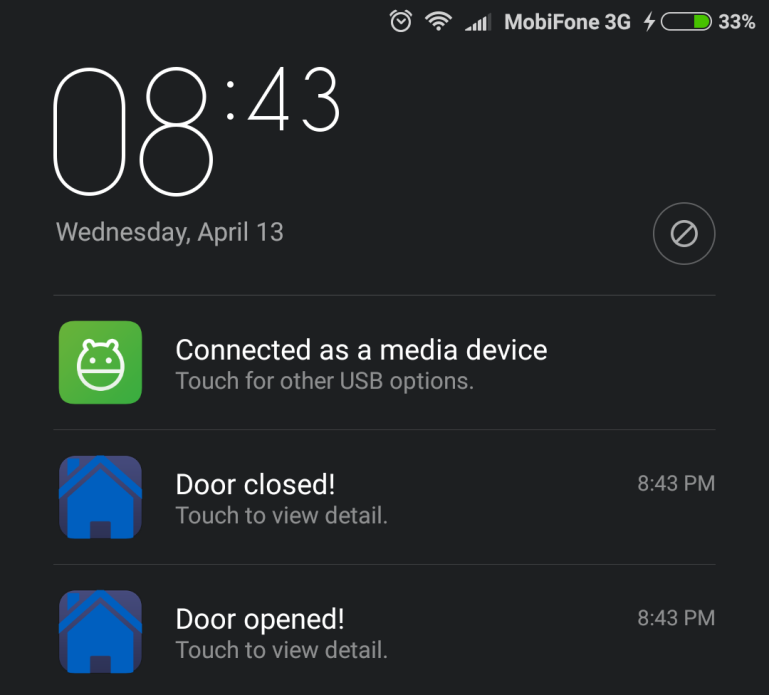


Android application only sends delete command unidirectional. User has ability to access Setting screen to delete the log, also means the delete command will be always transmitted and executed. Therefore, no error message should be returned in any case.

  
*Figure : Activity Log screen after deleting successfully*

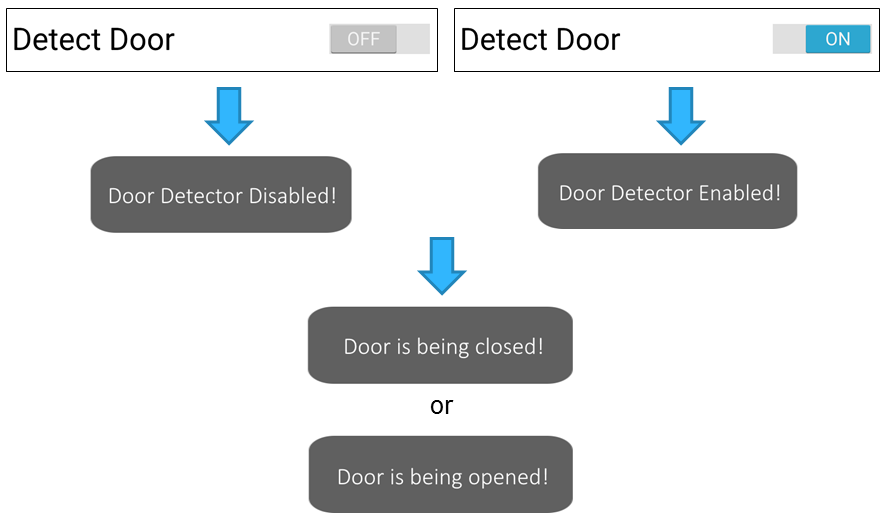
## 5.4. Pushing Notification

Pushing notification via Pusher is executed in background, so no any error or successful message will be shown on screen when triggered an event, except the final notifications on Android.

  
*Figure : Final notifications on Android*

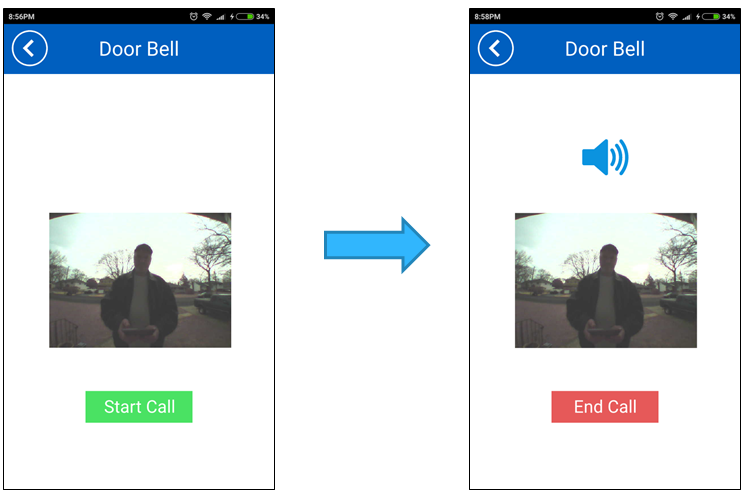
User can check the success of sending message to Pusher server by go to Activity log screen. If an event was recorded, it means that message had been sent to Pusher. The rest work is checking internet connection of smartphone and in user house.

## 5.5. Control Pushing Notification

  
*Figure : Response the current state of door sensor*

When user change Detect Door switch option to On or Off, application will show corresponding popup message to inform. Then another popup will be shown to alarm what current state of the door is, being opened or being closed.

## 5.6. Calling Doorbell

 *Figure : Calling Doorbell Testing*

When user click Start Call button, the button will change to End Call and a speaker shown on screen. No any error message will be shown. If calling got problem, we should check internet connection in both 2 clients side.

Default, when accessing Door Bell screen, application will load automatically the photo that taken by the time visitor press the doorbell button. If taking photo process get any problem (it means no any photo was taken), it still will load the most recent photo saved on server.

# CHAPTER 6 – FUTURE WORKS

This is the last chapter of Home Security System’s Documentation. We will discuss about future works to enhance the performance of system.

Firstly, adding more sensors is the most important things. The needs of user is very diversity, not only door sensor and motion detector. They can be CO detector sensor, glass break sensor, etc… This could be done easily because Arduino has a lot of digital PINs, and almost of sensors have the same operation: trigger the signal and when reaching some thresholds.

It’s glad to hear that with Linphone SDK, we can integrate video calling feature, not only voice over IP. But the lack of time had prevented us to build this in this moment. When that feature is integrated, taking photo process outside the door is not needed, so that will release a lot of system resources for using other tasks (no fswebcam package and Apache HTTP server).

To make a business with this project, we should integrate a procedure for automatically configuration at first use. In our project, IP address of Raspberry Pi is entered manual. But in the future, it should be this operation:

1. If HSS-Board is connected to router via wireless connection, a feature called “Auto Wifi” will be deployed. It means user can request the Raspberry Pi access any available wireless network you want via smartphone. If HSS-Board is connected by RJ45 cable, user can ignore this.
2. HSS-Board and HSS-App are in the same network. The application will scan the network to recognize HSS-Board’s IP (ping to broadcast address and then do reverse ARP process to find IP corresponding MAC address). So user doesn’t need to figure out HSS-Board’s IP and enter it manual anymore.

[2] Morgan Stanley, Research “Technology and Internet Trends”, 2015

1. Karen Rose, Scott Eldridge, Lyman Chapin, “The Internet of Things: An Overview. Understanding the Issues and Challenges of a More Connected World”, October 2015 [↑](#footnote-ref-1)
2. Morgan Stanley, Research “Technology and Internet Trends”, 2015 [↑](#footnote-ref-2)
3. Linphone overview at <http://www.linphone.org/technical-corner/liblinphone/overview> [↑](#footnote-ref-3)
4. Pusher Documentation at <https://pusher.com/docs> [↑](#footnote-ref-4)