EECE 381 Module 2 Report

The Security System with Android Device

Instructor: Steve Wilton

Section: L1B

Group: #2

Group members:

Xu Chen	#19502137
Peter Wang	#69736080
Yuxin Xu	#37909090
Michael Zheng	#23410111

Table of Contents

1. Introduction

Security camera is a very common product that many people use for a variety of purposes. Normally, security cameras have their own unique equipment and require special installation for it to be working. Their prices range from a few hundred to a few thousand dollars. But they have one common function: detect intruders and trigger alarm.

Our security camera system works the same way as a traditional security system. However, you only need two android devices for it to work. It is cheap, user friendly and does not require any special installation at all.

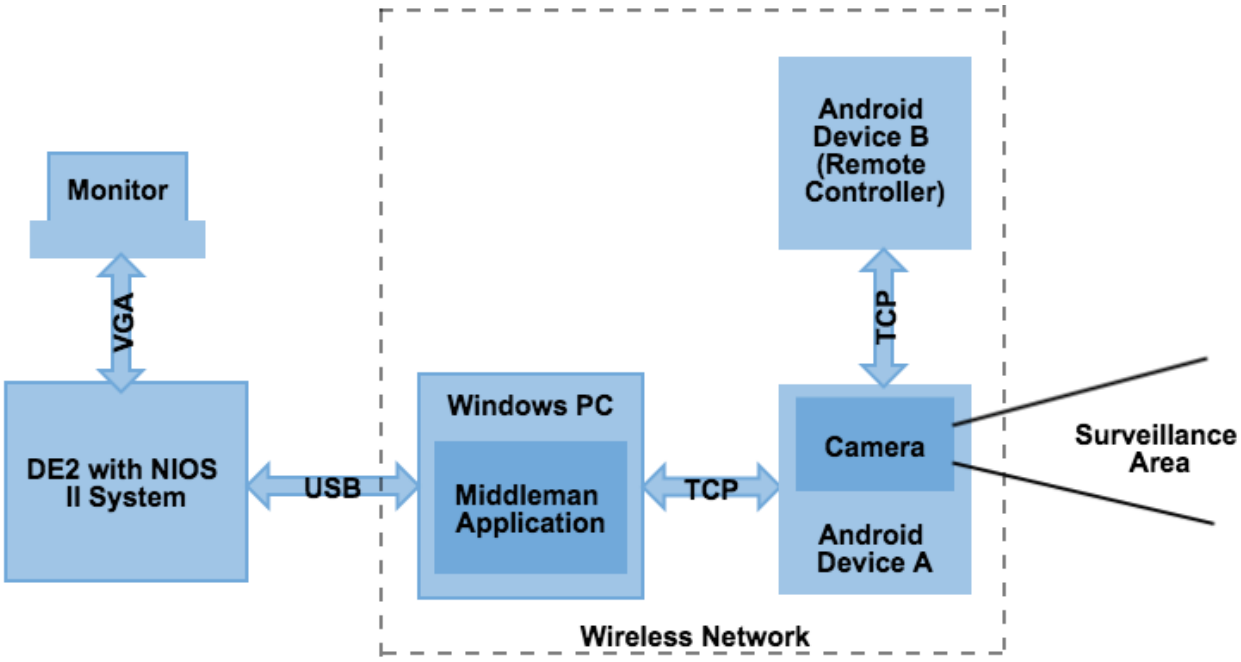

Figure 1 System Structure

As shown in Figure 1, our system consists of three major parts. Android device A acts as a camera which faces the surveillance area, such as a door or a window. Android device B stands for the owner's cell phone. It is used as a remote that turn on/off the camera or change its settings. Also it allows the user to see the real-time pictures delivered from camera when the alarm has been triggered. The third part is a DE2 board which is connected to a PC and a monitor. Every

time when an alarm is triggered, the camera send a site picture to the PC and display this picture on the monitor through DE2 board. In our design, this part is put in a security office for the constant surveillance.

2. Development Process

2.1 Market and Requirement Analysis

As the first step of our project, market and requirement analysis costs us much time. We begin with collecting information of some mature security systems on the market. Based on these data, we set up our goals and market orientation.

Our security camera system targets at common families who would like to have a basic camera system for their house or facility security, but are not willing to pay too much money for it. Their requirement includes real-time photo display, audible alarm, long-distance message or picture reminds, etc.

2.2 High Level Design

Based on requirement analysis and tutorials, we set up the top level design (especially the system structure as shown in Figure 1) in a short time. For the requirement of economy, users only need an old Android cell phone that is no longer in use, since our app could run on any Android device as long as it has a camera.

In this step, our major challenge was that we did not know if the picture transfer speed could meet our requirement or not. The performance of picture comparison algorithm was another concern. Fortunately, in later steps these concerns did not trouble us too much.

2.3 Detailed Design and Coding

We were able to set up communication between the software on DE2 board and the app on an emulated Android device. We also implemented the camera API so that the device could take picture every interval of time. We did not made much progress until this point, but we learned how to use the Middle Man software and how to work with the Android SDK, therefore we were able to finish our project on time during the second half of the schedule.

2.4 Testing and Final Version

Test includes module test operated by the coder, and system overall test operated by users (friends and classmates). In module test, we used the debug mode on Eclipse. Single step debugging made errors and some undiscovered bug clear in this stage. After fixing them, we proceeded into the overall test. The group members and some other students were invited to simulate the break-in process and saw the performance of our system. During this stage, we adjusted some improper settings and fixed some trivial bugs until the final version.

3. Work Accomplished

3.1 Camera Control

We have enabled the Camera to take a photo continuously and the intervals could be set to as short as 3 seconds. We just keep two continuous photos for comparison.

3.2 Picture Comparison

Picture comparison is one of the most important part in our code. The system compares two continuous photos. If the comparison shows that there exists a significant change between them, the alarming process will be triggered

3.3 Communication between Android Devices and DE2 Board

Once an alarming process was triggered, the camera will send the picture to both DE2 board and the remote (android device B in Figure 1) via wireless network. We implemented the communications using socket programming.

3.4 Real-Time Picture Display on Monitor

The system displays the real-time picture delivered from the camera on a monitor. We also activated switch 2 on DE2 board for clearing the screen.

3.5 Save/Load on SD Card

When an alarm was triggered, the site picture will be saved to SD card automatically. By toggling switch 1 on DE2 board, we could easily load and display the saved picture.

3.6 Sound Effect on Android Devices

When an alarm is triggered, there will be a sharp sound played on both camera and remote to remind the owner and scare off the intruders.

4. Detailed Design

4.1 Task Allocation

As described in chapter 1, our system consists of three major parts. The task allocation is also based on this distribution.

We use an Android device as a camera which monitors the customer's house by continuously taking photos and comparing them. Therefore, the photo taking, format

modification and photo comparison algorithms are running on this device. Also, the module maintains a socket channel for sending pictures and receiving setting instructions. A sound class is also needed for playing the alarm sound. This module is an Android app developed by javascript.

The DE2 board and connected PC work as a server in the security office of a building or a residence. Once an alarm was triggered, it receives and displays the site picture delivered from camera. So a socket server and VGA access functions are indispensable. It also contains the SD access code for saving and loading pictures. This module is developed by C on Windows 7 platform.

Usually the house owner's cell phone acts as a remote, which enable users turn on/off and change settings of camera like a TV remote. It also receives the site pictures when alarms are triggered. The Android app running on this device needs to provide an user interface and maintains a socket channel for two-way transmission like that on camera.

4.2 Hardware Introduction

Our system hardware consists of two Android devices (with camera), one PC and one NIOS2 system emulated by QSYS on DE2 board. The Android device could be a tablet or a cell phone. During development, we use Samsung Galaxy Tab 4.

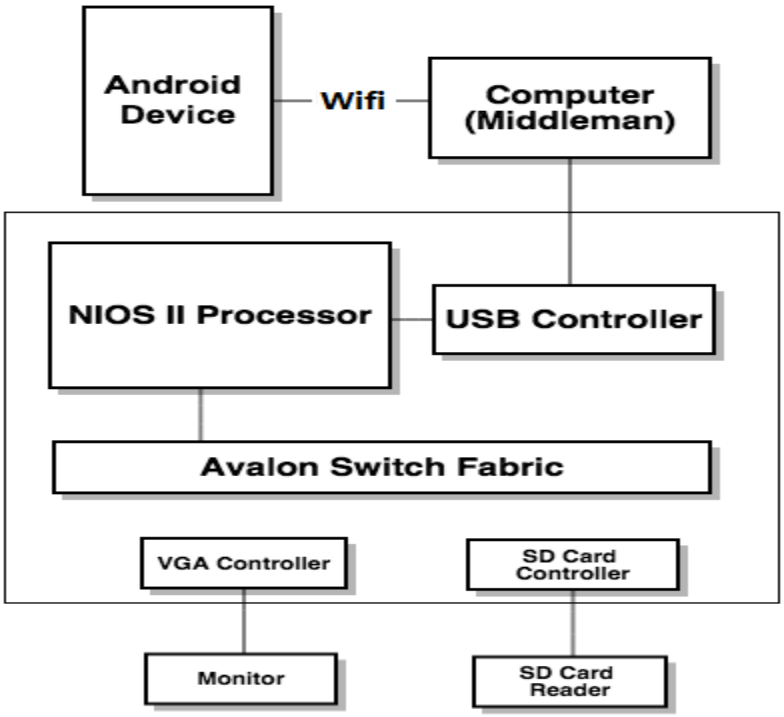

Figure 2 NIOS2 System Block Diagram

In NIOS2 system, besides the processor (NIOS II/s) and on-chip memory, other major hardware components are:

- USB controller

  This module is used to control the USB communication.

- SD Card Interface

  The SD Card Interface module functions as an interface between the SD card reader on the DE2 and the NIOS system, it allows the system to store and retrieve large files in an SD Card.

- Graphic related modules

  There are several modules for VGA Graphics. Together, they allow the NIOS system to control a VGA output device.

4.3 Software Introduction

According to the hardware structure, our software modules are also distributed on three different devices: camera, DE2 board and remote android device.

4.3.1 Software module on Camera

On the camera device, when the user starts our app, the main activity pops up first. It contains the code for creating socket and making test connection with the Middle Man software wirelessly.

Then enter the camera activity by pressing the "camera" button. The onCreate function is overridden for creating a camera instance, a preview window of the camera on the screen, an alarm sound, and a timer task which runs every 3 seconds.

In the timer task, a photo is taken using "*takePicture*()". It is saved in Bitmap and resized down to 320*240 by using "*createScaledBitmap*()". The variables "*photo1*" and "*photo2*" are used to store two newest photos. All bitmaps are converted to a pixel array in RGB565 format.

Afterwards, photo1 and photo2 are passed down to a comparison algorithm (introduced in section 4.3.5). If their similarity is less than 95%, the alarm routine will be called. Otherwise, the whole process will halt until the next timer task begins. The alarm routine contains the code for sound effect and sending picture to DE2 board and owner's Android device.

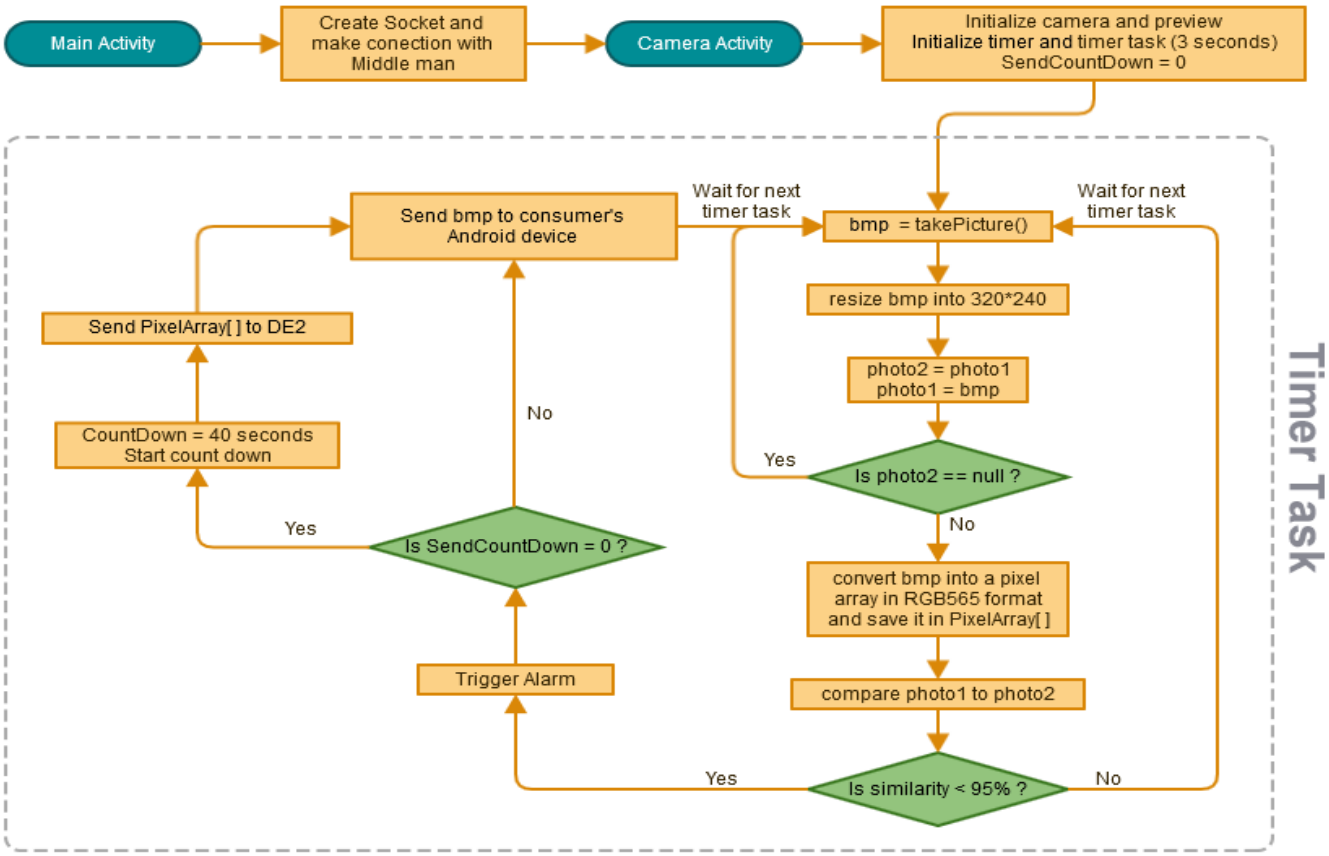

Figure 3 Flow Chart of Module on Camera(Android system)

We also define a counter named "*CountDown"* to avoid traffic congestions between camera and DE2. Because the picture transfer will take approximately 35 seconds, we stop the picture comparison temporarily for the same length of time (35 seconds), so that there will not be large bulk of data flush into the socket buffer.

4.3.2 Software Module on DE2 Board

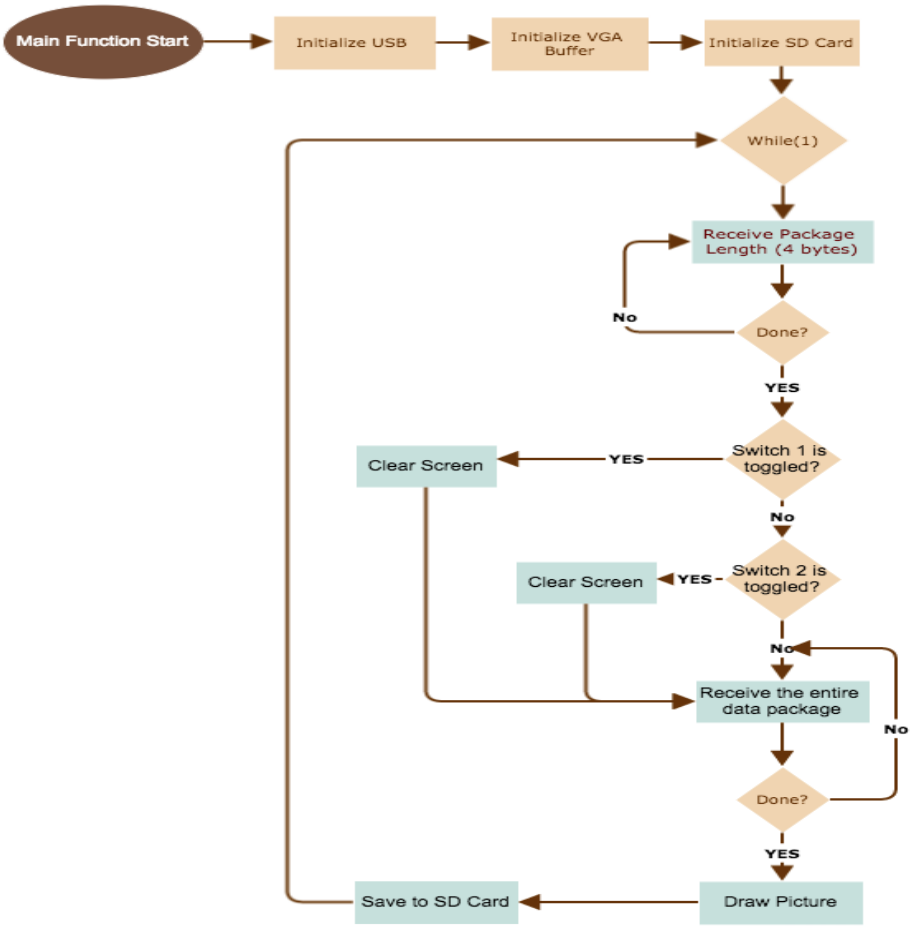

Figure 4 Flow Chart of Module on DE2 Board(Nios II system)

The major task of this module is to receive photos from camera (actually forwarded by middleman) and to display these photos on the monitor. After initializing the USB controller and VGA drawer, it starts an infinite loop to keep waiting for new photo. Once a photo transfer finished, it will be displayed on the monitor and saved to SD card automatically.

4.3.3 Software Module on Remote

The app on user's mobile phone receives and displays the photo from the Camera device. The program begins with the Phone Activity, which contains nothing but a button to enter Phone Activity. In this activity, an ImageView is initialized for displaying photo received, and the

ServerThread is initialized. In the ServerThread, a socket is created to establish the connection between this device and the Camera device. Afterwards, it keeps listening to the port it binds and receives data when there is a photo transfer. If the *Display* button is pressed, the received photo will be displayed in the *ImageView* window.

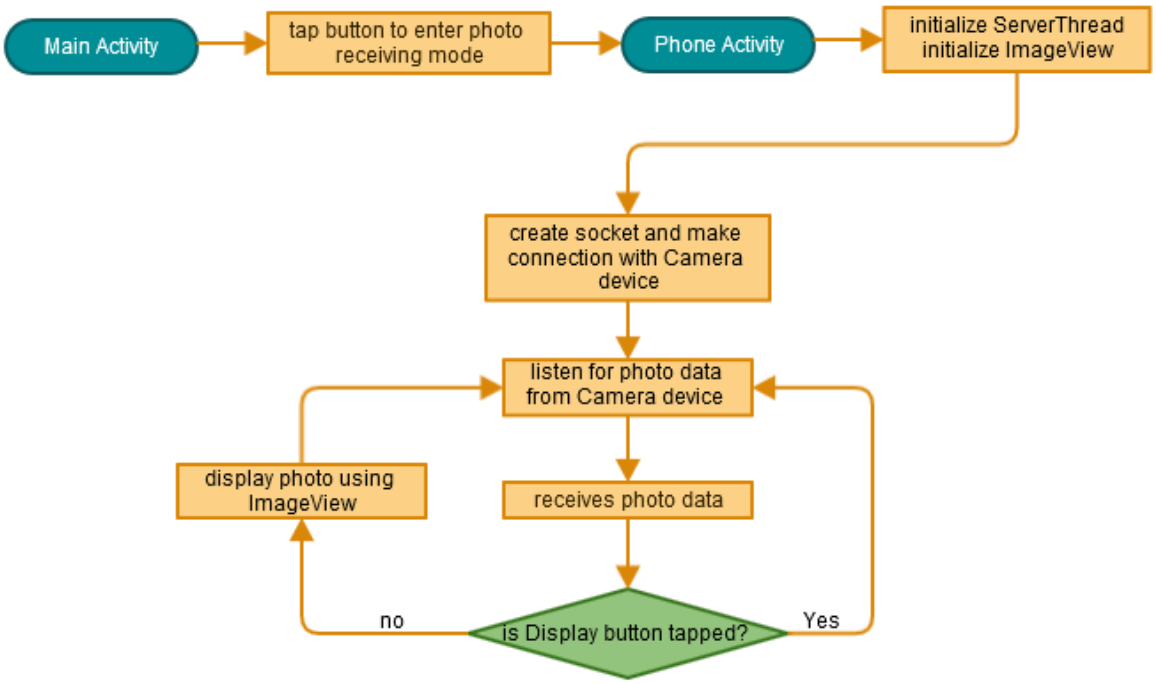

Figure 5 Flow Chart of Module on User's mobile device(Android system)

4.3.4 Data Transmission Between Android Devices and DE2

Image transfer is crucial in our system. It could be classified into two types: Android to Android, Android to DE2. Socket programming is used no matter what type it is. Client/Server mode make all data transfer simple and safe.

1. Android device to Android device via wireless network

We came up with 3 ways to accomplish this task : bluetooth, NFC and Wifi-direct P2P. After considering the range that the two devices will be placed at, we decided to use Wifi-direct

P2P. Wifi-direct allows two android devices to be connected wirelessly and initiate file transfer without first connecting to a hotspot. The team manages to connect two device together, but is unsuccessful in initiating a file transfer. Therefore, we had to create a local network firstly, and make two devices join the network. Then we used socket programming to set up a channel, and initiate file transfer through the ports we defined after setting up the client and server.

2. Android device to DE2 board

Since we can not make them exchange data directly, a computer (with a wireless network adapter) has to be used as a relay station. The Android device communicate with the computer, which then forward what it received to DE2 board. A software "middleman", which is running on the computer, is responsible for the communication between PC and DE2 board over USB. The socket programming is similar to Android-to-Android type.

4.3.5 Picture Snapshot and Comparison

We have enabled the Camera to take a photo continuously. The intervals could be set to as short as 3 seconds. Considering the transmission speed, all photos will be trimmed to a bitmap in the size of 320*240. We just keep two continuous photos for comparison. If the comparison shows that there exists a significant change between them, the alarm will be triggered and the photo will be converted into a data array for transmission to DE2 board and the remote.

Picture comparison is one of the most important part in our code, and we have tried two methods. The first one compresses each photo into a black and white 10*8 image. Then it compares these images and produces an integer that represents the number of different pixels.

The second method extracts the RGB values from pixels in two photos. A difference larger than the preselected value will be marked as a distinction. This process is done repeatedly

for every pixel in photos and we get a double-type variable which represents the percentage of similar pixels.

Finally we decided to use the second method, since it gives us a more precisely result .

4.3.6 Sound effect on Android Devices

To implement sound effects on an android device, we used the android media manager API. We firstly created a soundpool to store a mono mp3.file and activate the hardware buttons to control the sound. Then, we assigned the mp3.file with ID number and load the sound from the soundpool. Finally, call a play function to play the sound and an auto pause to disable the sound.

5. Solution Assessment

Our system has almost all the features that we initially wanted to embrace.

- Taking photos continuously
- Photo comparison
- Photo transfer
- Photo display on monitor and remote Android device
- Save and load photos on SD card

The only feature that we did not implement successfully is:

- Remote configuration and control of the camera

The final version of our system ran without any problem during the demo. Our system successfully detected the break-ins and took and displayed the photos of break-ins. This is all due to the efforts we made in testing.

Due to the limited time and experience, our system is hard to be as robust as the security system on the market. There are probably still many hidden bugs in our app, they are hard to find unless we run a series of complete tests on it.

6. Integration of Standards

Our app follows some of the Android App standards and design principles.

- Our app has an action bar and the Back button on the bar brings back main activity to the front when a sub-activity is running.
- We used the Holo Light theme as the main theme for our app so that it looks consistent with the default Android system theme.
- Our app supports lower versions of Android system. We tested our app on an API level 14 Android system which is older than which we have targeted (API level 19), the result was satisfactory.
- Our app also supports different devices with different hardware standards, we ran the app on an HTC cellphone and it work without any problem.

This security camera system does not need much user interference. Once it starts running, it will do everything by itself, therefore we did not spend much time on improving the GUI.

We believe that, to some extent, imposing Android App standard helps the evolution of technology. It is important to design apps that behave in a consistent, predictable fashion. If all apps follow this standard, the GUI of all apps would be consistent, and users would easily grasp the use of it. Better user experience means more customers and bigger market, and this would then lead to faster evolution in Android System and Android Apps. A fixed pattern certainly has

negative effect on technique evolution. But it can not exceed the advantages it brings to both developers and users.

7. Conclusions

Our camera security system meets almost all high level goals and is able to detect suspicious activities happen in the surveillance area, and report them to the security server and the user's cell phone.

Still, we have proposed more features that could be added into this system in the future development.

- Sound detection

We can add sound detection into the system by activating the microphone on the Android device to pick up sound waves, this would help the system to detect a break-in more accurately.

- Remotely configure and control the camera

It's the only feature which we proposed in the top level design but did not really implement. It requires a two-way socket connection between the user's Android device and the Camera device. Given enough time, we definitely could implement this feature.

- Support multiple camera device

Supporting several cameras could significantly enhance the monitoring area and security level. To make them work properly, we need to maintain a camera queue at the server side and solve the conflict problem caused by the concurrent alarms on different camera.

8. Source Code Link

Data