Chapter-4
Analysis and Design

4.1 Use – Case Diagram

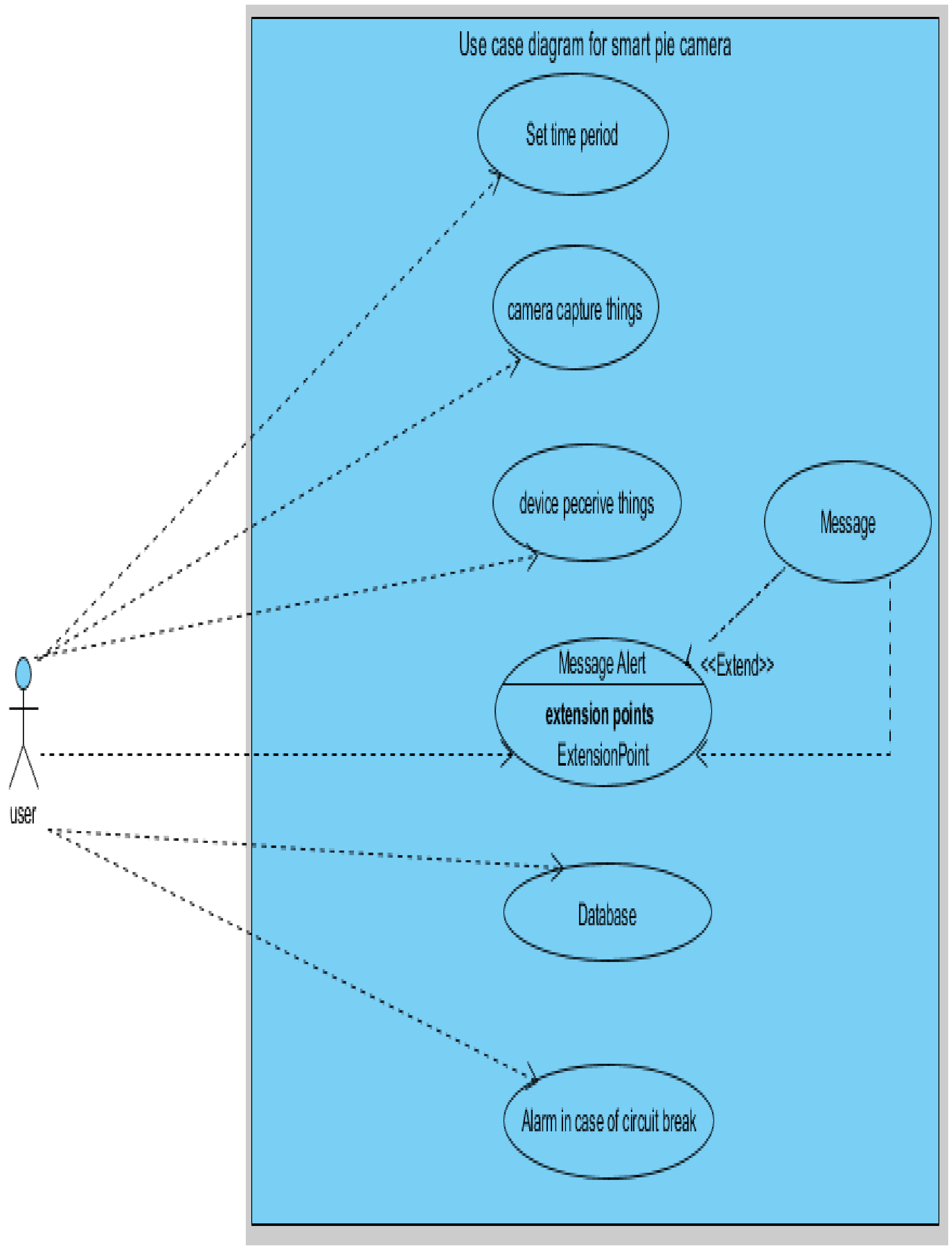

Figure 4.1 Use –Case Diagram

4.2 Activity Diagram

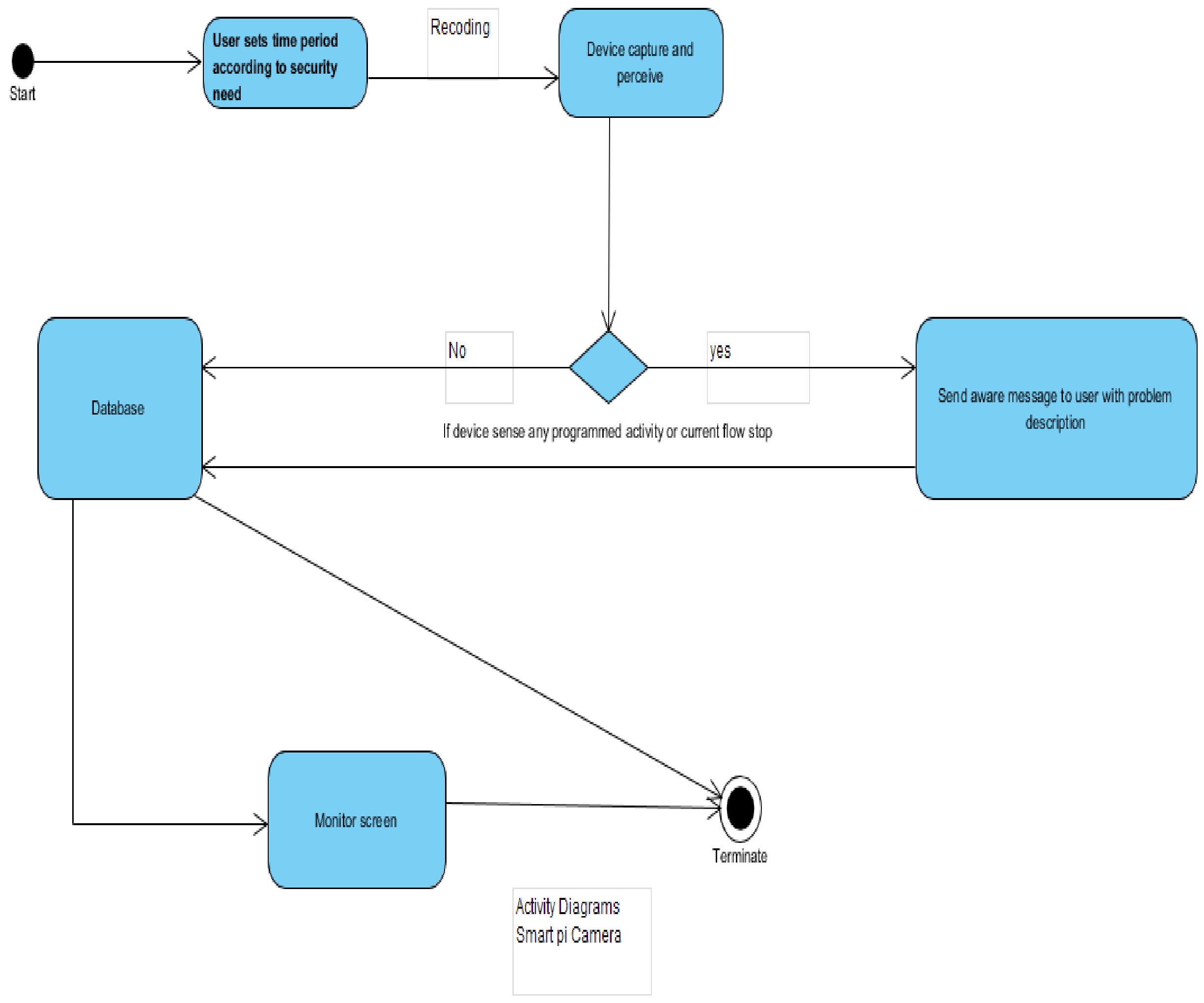
Start
User sets time period according to security need
Recoding
Device capture and perceive
Database
No
yes
If device sense any programmed activity or current flow stop
Send aware message to user with problem description
Monitor screen
Terminate
Activity Diagrams
Smart pi Camera

Figure 4.2 Activity Diagram

4.3 Sequence Diagram

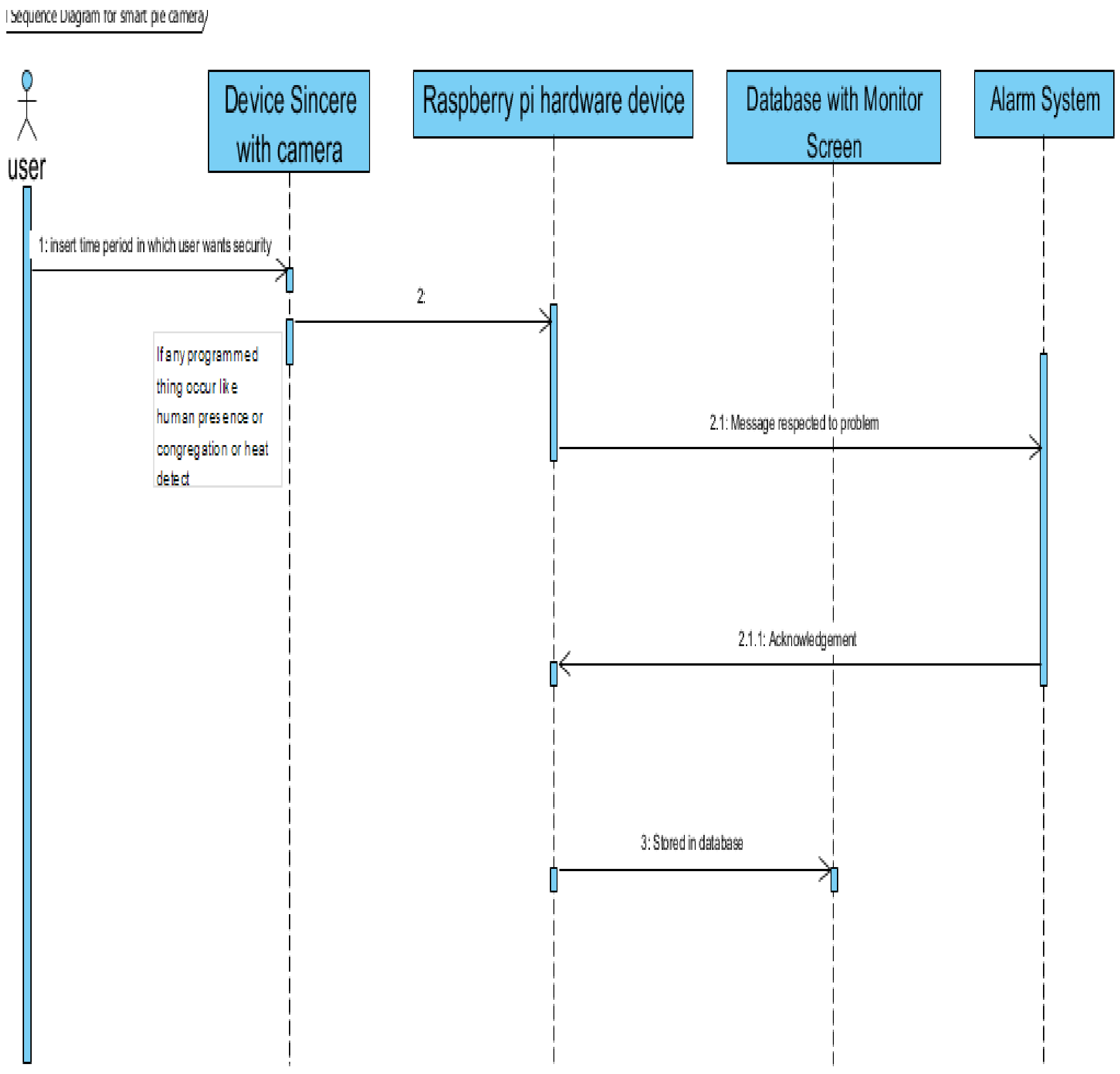
Sequence Diagram for smart pie camera
user
Device Sincere with camera
Raspberry pi hardware device
Database with Monitor Screen
Alarm System
1: insert time period in which user wants security
2:
If any programmed thing occur like human presence or congregation or heat detect
2.1: Message respected to problem
2.1.1: Acknowledgement
3: Stored in database

Figure 4.3 Sequence Diagram

4.4 Dataflow Diagram

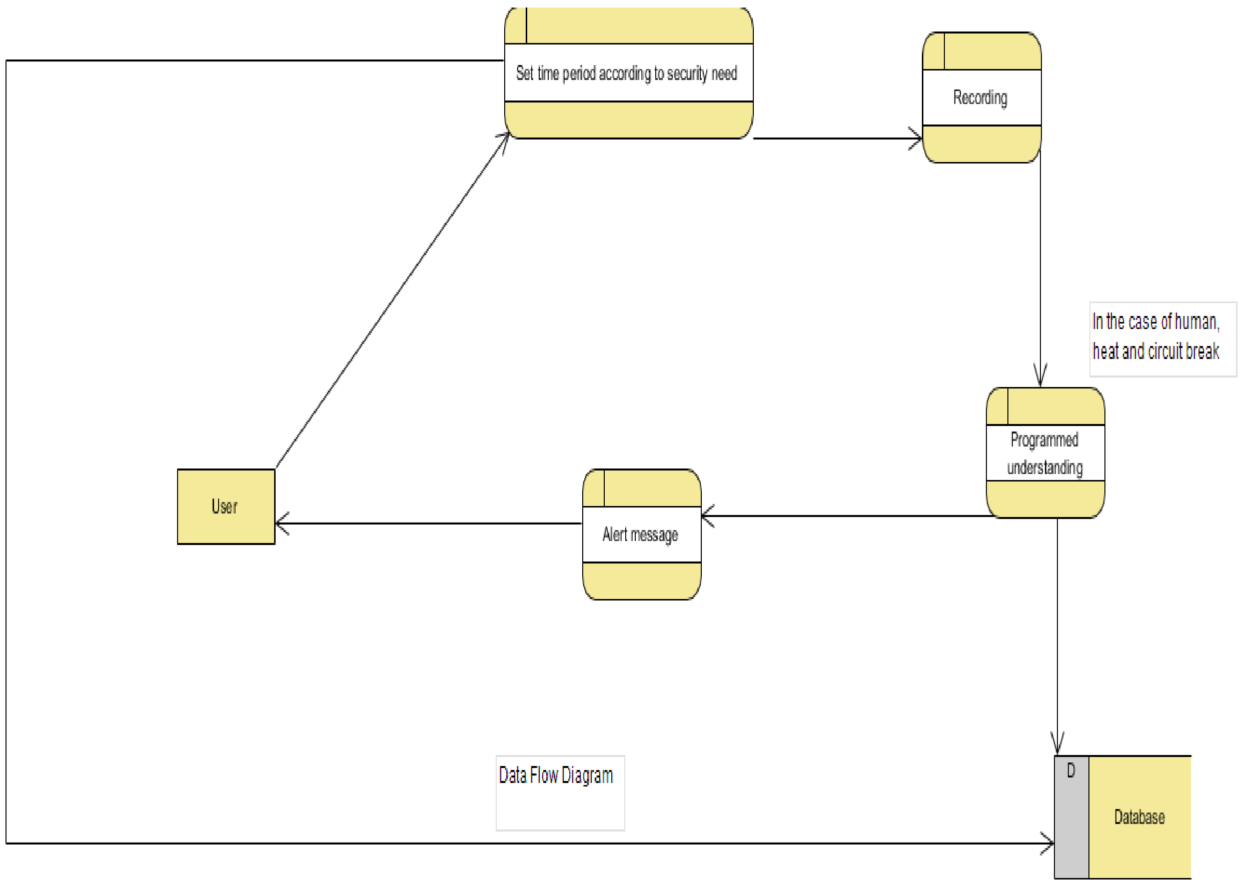
Set time period according to security need
Recording
In the case of human, heat and circuit break
Programmed understanding
Alert message
User
Data Flow Diagram
D
Database

Figure 4.4 Dataflow Diagram

.

4.5 Resources:

A security system is a real time embedded system utilized for motion detection and alerting the user and neighbors by messages, calls and alarms. Therefore, basically it will protect home from robbers and burglars. The implementation of the following function will be implemented using hardware Raspberry pi, here we are going to use Raspberry pi 3 B+ model for implementing the project. The pi-based security system is composed of mainly two parts. These are: design software & design hardware.

4.6 Design Hardware:

The module Raspberry Pi 3 B consist of parts namely:

- Raspberry Pi Model B controller,
- RJ45 connector,
- Micro SD card,
- HDMI cable,
- Bread board,
- Connecting wires for GPIO,
- PIR motion sensor,
- PI camera module.

4.6.1 Raspberry Pi 3 Model B

The model is used for implementing the project. The model is a 1.4 GHz 64-bit quad-core processor, dual band wireless LAN. Bluetooth 4.2/BLE, faster Ethernet and Power over Ethernet support with separate PoE HAT, 1 GB RAM, GPIO pins. The model has advantage over other models in terms of increased number of USB ports and large number of GPIO pins.

4.6.1.1 Booting up the Pi Model

Raspbian, is recommended OS for working on Raspberry Pi, is a free open source operating system, is the foundation's official supported operating system based on Debian. Raspbian 'Wheezy' image was written into the 4GB Micro SD card. Raspbian 'Wheezy' image was written into the 4GB Micro SD card. After slotting in the Micro SD card and connecting RJ45 Ethernet cable to the Pi and, the personal computer with VnC Connector software (using IP configuration through ethernet we will remotely access pi from computer running on windows platform), the pi gets ready to get configured. This was to allow it communicate with the Raspberry Pi.

4.6.1.2 Setting Up internet connection on the Pi

Internet was necessary in so that the Pi can communicate over network protocols and thus allow for installation of necessary Python packages. The architectural diagram below drawn is used to achieve that. Pi needed internet for communication and also for installing python packages. The architectural diagram below is used to achieve that.

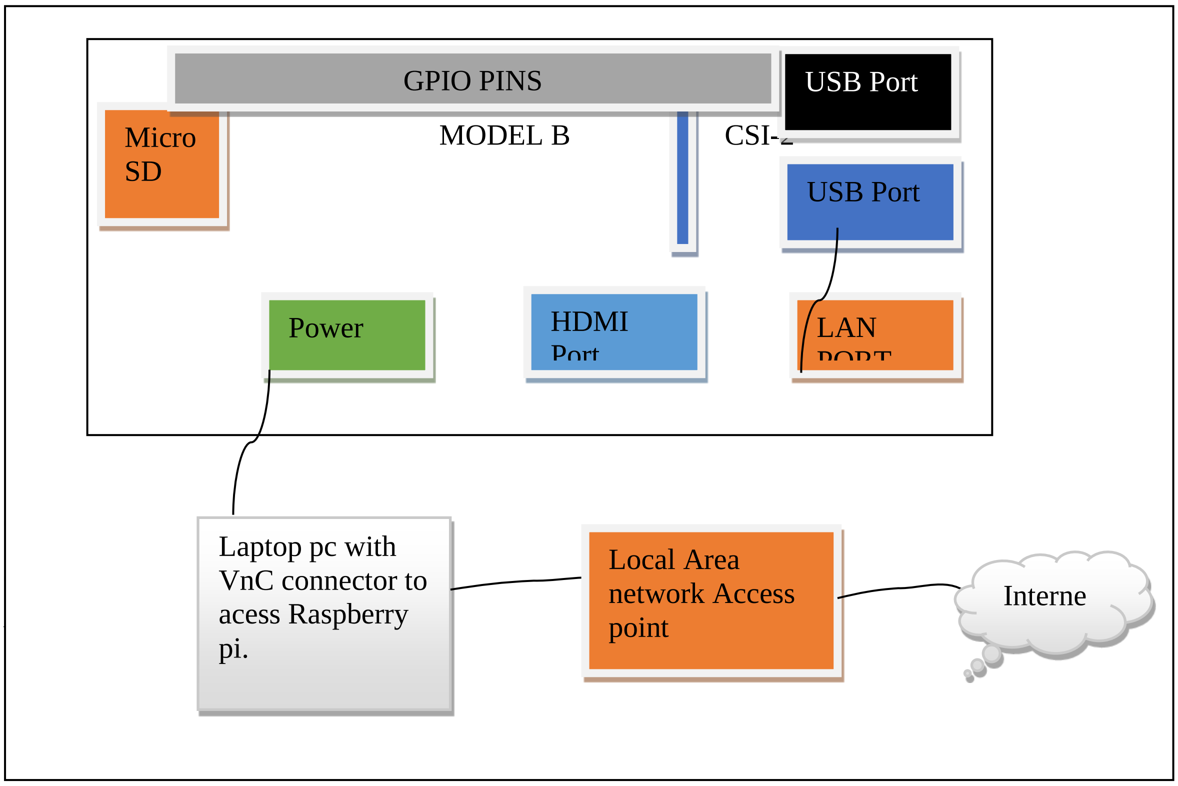
GPIO PINS
USB Port
Micro SD
MODEL B
CSI-2
USB Port
Power
HDMI Port
LAN PORT
Laptop pc with VnC connector to acess Raspberry pi.
Local Area network Access point
Interne

Figure 4.5: Internet set up architecture

Since the broadcast router uses Dynamic Host Configuration Protocol (DHCP) to dish out IP addresses to devices connected to it, it was necessary to change the IP address of the Pi from static to dynamic. This was done by editing the network interfaces file using the command:

```
Sudo nano /etc/network/interfaces
```

4.6.1.3 Enabling the Pi Camera

The camera is especially designed for the Raspberry pi. Camera is hooked to the raspberry pi through CSI-2 , electrical port ,an extremely fast port. To configure and enable the camera, the following commands were executed at the CLI of the raspberry pi:

```
Sudo apt_get update
Sudo apt_get upgrade
Sudo raspi-config
```

After these configuration settings, the system was rebooted. This was done to ensure that the camera was allocated enough space in memory. The camera takes 5MP image and has a resolution of 1080 by 890. And to ensure that the camera was well configured and functional, the following command was executed.

```
Sudo raspistill—o image.png
```

This by default this command takes a three second image and save it in a file called image.png.

4.6.2 Setting Up the Passive Infrared Sensor

This is formed the prime motion sensor. It was used to control the entire system. The device used here was HC501SR passive infrared sensor. The detection range is 7 meters by 140(degrees) coning angles. It has a delay time of 16 seconds but adjustable. The ambient temperature is 253K-323K. It was powered directly from the Pi through the 5V dc supply pin. Its output was connected as the input to the programmable GPIO pin.

4.7 Hardware Architecture

The entire system modules were interfaced together as shown below.

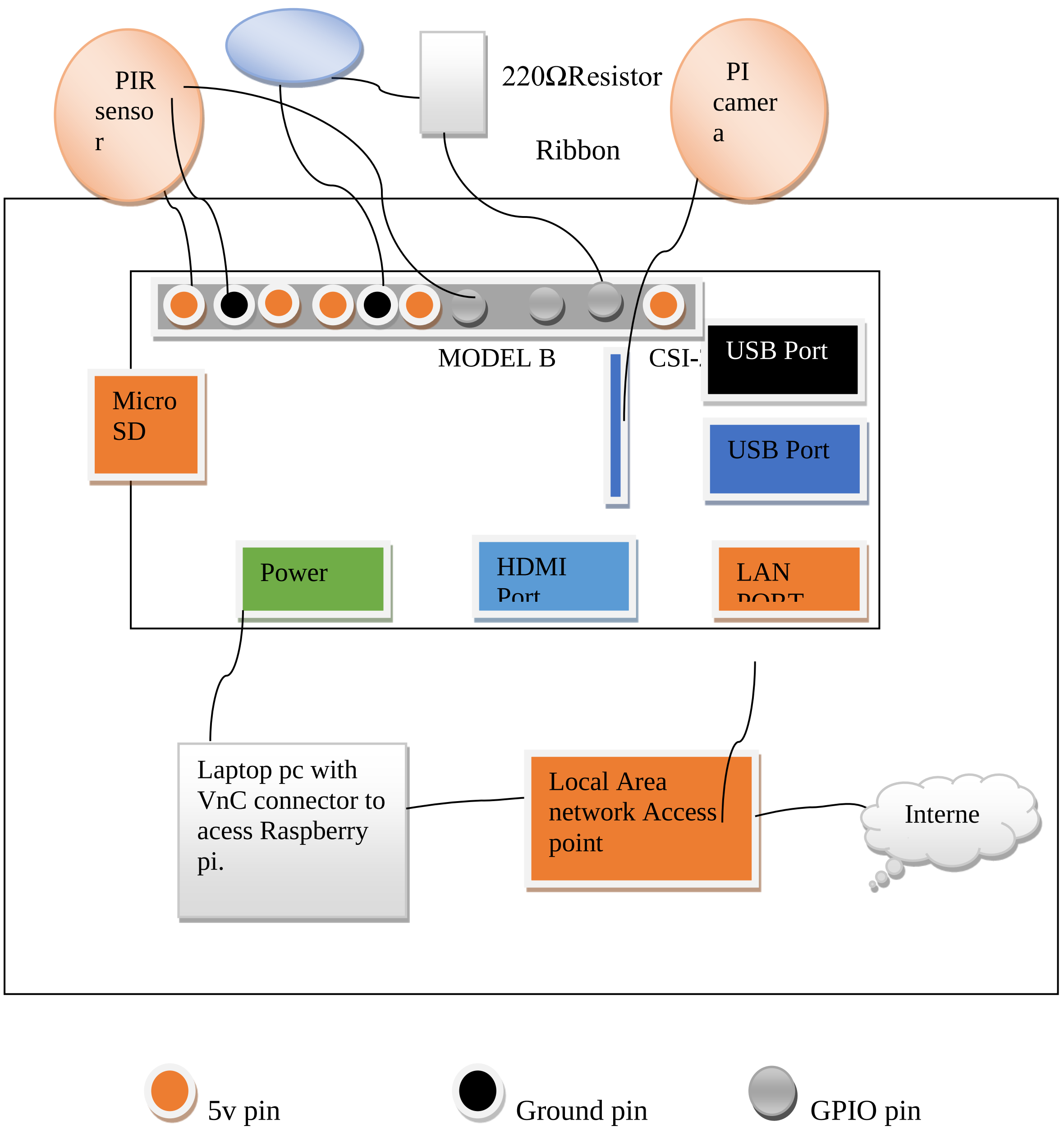

Figure 4.6:Hardware Architecture of the security Systems Based on Raspberry pi

4.8.1 Design Software

4.8.1 The flowchart of the Raspberry Pi Based Security system:

The data flow diagram is use to show the design of security system. It shows the event starting from the stage of detecting a motion to sending of message or alert. This algorithm was implemented using a python script.

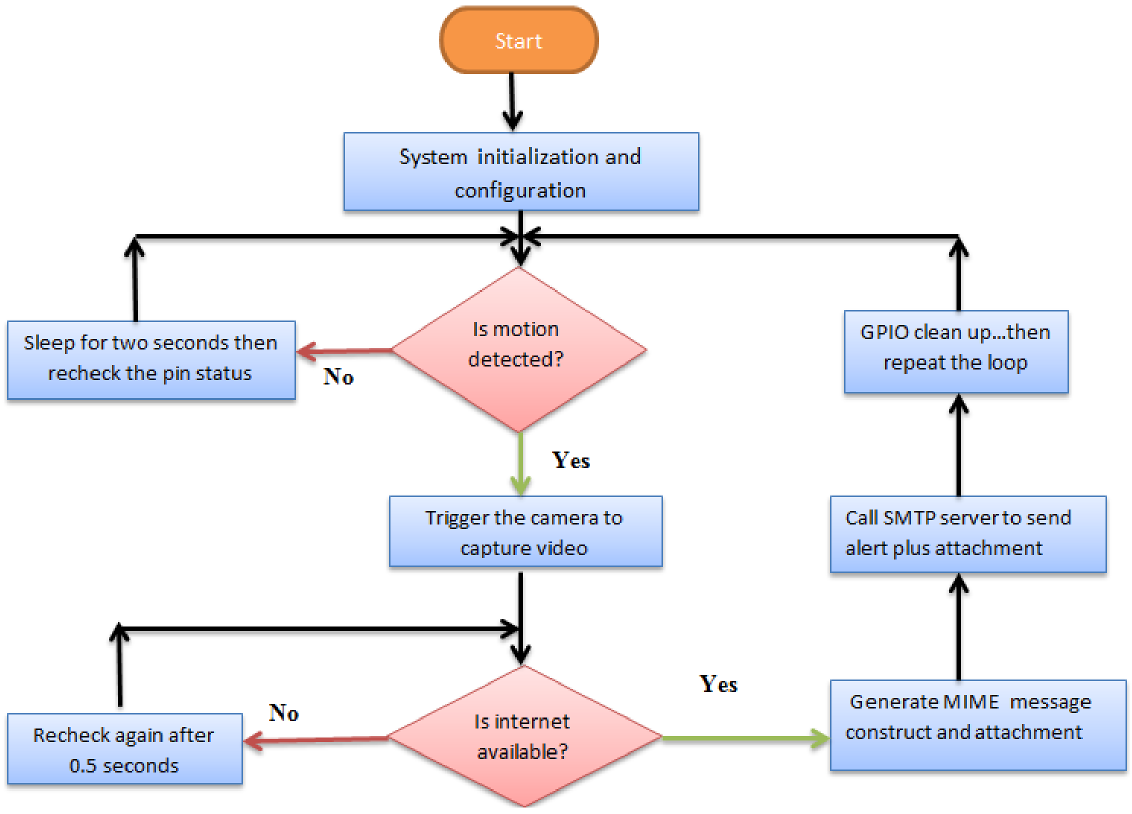
Start
System initialization and configuration
Is motion detected?
No
Sleep for two seconds then recheck the pin status
Yes
Trigger the camera to capture video
Is internet available?
No
Recheck again after 0.5 seconds
Yes
Generate MIME message construct and attachment
Call SMTP server to send alert plus attachment
GPIO clean up...then repeat the loop

Figure 4.7: Flowchart implementation of the security system

4.8.1.1 Initialization and Configuration of System

This involved the following tasks:

- Importing Python libraries and packages. These libraries are predefined and help in making the interfaced modules work properly.
- Pi Camera setting and configuration.
- GPIO settings and pin initialization: (the channel was set using the BCM channel numbering. Passive infrared pin channel was set to read mode while the led channel was set to drive/write mode.

a) Read a Channel

In order to read the value of any GPIO pin, simply type; GPIO.input(channel) .

b) Drive a channel

In order to drive a channel of GPIO pin, type; GPIO.output(channel, status).

This sequence of events can be elaborated well using the block diagram below :-

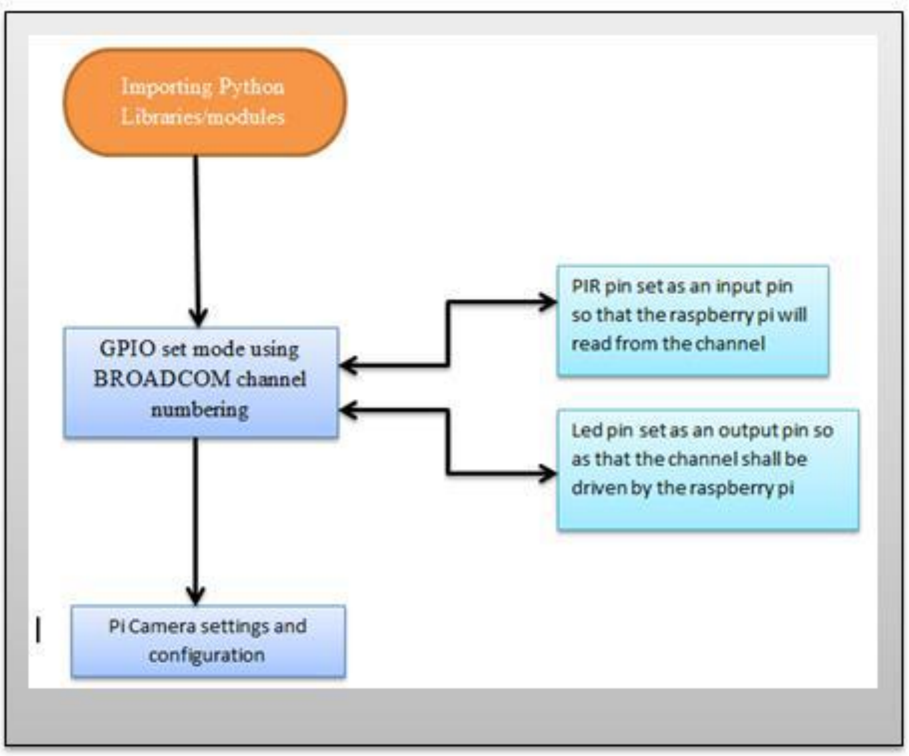
Importing Python Libraries/modules
GPIO set mode using BROADCOM channel numbering
PIR pin set as an input pin so that the raspberry pi will read from the channel
Led pin set as an output pin so as that the channel shall be driven by the raspberry pi
Pi Camera settings and configuration

Figure 4.8 : System Initialization and configuration

4.8.1.2 E-mail Generation and sending process.

After the configuring the system to send an alert to the user, we can use Multipurpose Internet Mail Extension (MIME) package for the generation of the attachment. It supports ASCII characters and non-texts attachments (audio, video etc).For the delivery of the e-mail we used protocol name Simple Mail Transfer Protocol (SMTP) from the Raspberry Pi to the predefined user. This can be summarized using the blocks below.

4.8.1.3 Pseudo code

```
1. Upon restart of the system, send out email with boot IP assigned to a mail host.
```

2. Check the status of the GPIO pin. If the pin is LOW, GPIO output pin 13 should remain LOW and the system is idle. Else if the pin suddenly goes HIGH. Interpret this as an interrupt event
3. While the value of the input GPIO pin is HIGH (interrupt event), set pin 13 to be HIGH. This instance blinks the LED. Call the function that starts the Pi Camera.
4. Camera takes a 10 seconds video or capture a image as user wants and save it in a file.
5. The system checks whether the internet is enabled on the Raspberry Pi.
6. If internet, send email to a prescribed mail host. If no internet, wait for 5 seconds then check again.
7. Reset the PIR sensor pin to LOW and recheck again the status after 2 seconds. This should return the program to the main loop.

4.8.1.4 Developing the Full Code Listing.

To be able to develop the Python script that executes the algorithm defined in the flowchart, the following were done at the CLI of the Raspberry Pi:

- The Pi was started and a directory was created using **mkdir** command
- Inside the directory, a file was created using the **touch** command and made executable using **sudo chmod +x (filename).**
- The **nano** command was then used to open the editor and the full Python code was written there. The script was executed using the following command:

```
Sudo python filename.py
```

4.8.2 OpenCv – Python Video Processing

OpenCv is a very powerful tool used to analyze images and video files. The basic processing procedure to be followed is detailed in the flowchart below. Thresholding as a technique of image processing was chosen for the implementation of motion detection and tracking in video streams. The choice to script using OpenCv – Python was because Python on its own does not support video processing. There is so far no video processing library in Python. To achieve image processing, open CV is needed. The following flowchart was used for this implementation.

4.8.2.1 Flowchart

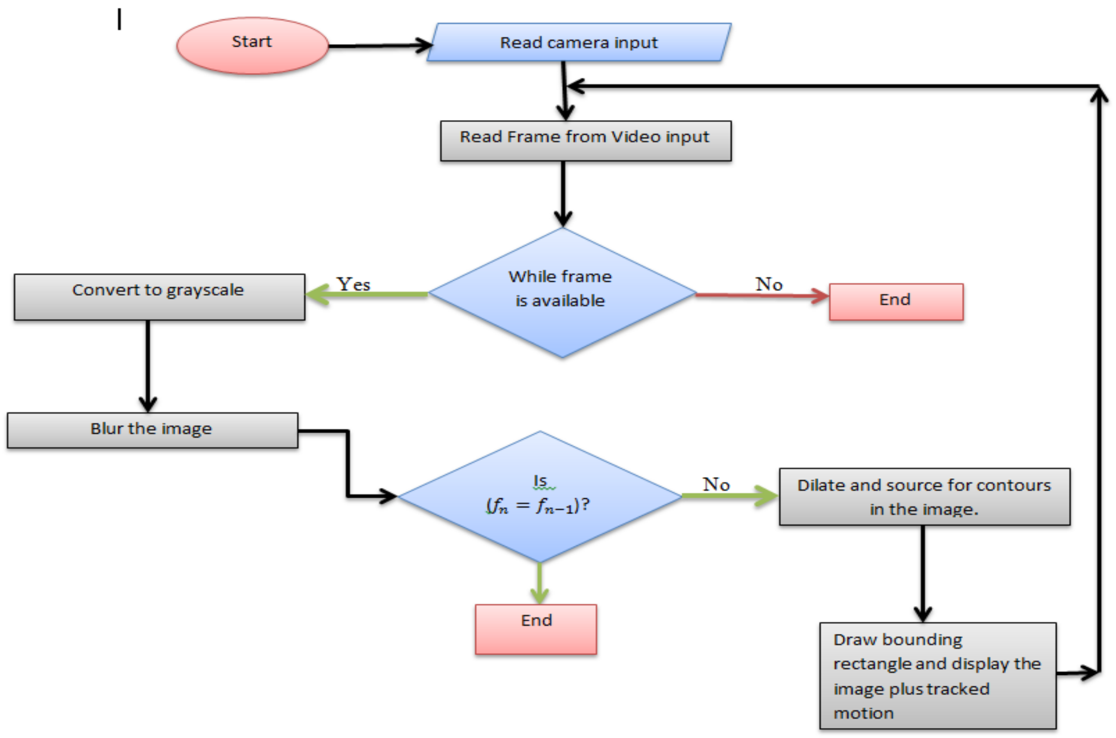
Start
Read camera input
Read Frame from Video input
While frame is available
Yes
No
Convert to grayscale
End
Blur the image
Is $(f_n = f_{n-1})$?
No
Dilate and source for contours in the image.
End
Draw bounding rectangle and display the image plus tracked motion

Figure 4.9 : Flowchart of motion detection and tracking algorithm

4.8.2.2 Pseudo code

1. Start the camera and set to capture video stream.
2. Grab a frame from the video stream. If frame is grabbed, continue with the process. Else stop. Initialize the frame as the current frame
3. Convert the captured frame to gray scale. Then do Gaussian blurring to remove noise in the gray image.
4. Capture another frame and repeat step two above.
5. Check for pixel threshold if enough to call motion detected.
6. Draw a rectangle around the region where motion was detected.