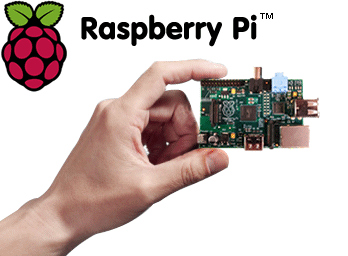
Tianqi Huang O’Dell

Coury Ferson

Michael Mcbride

Tianqi Huang

Christian Kirk



The Home Defense System – Team 5

**Contents**

**Section 1 - Project Description**

*1.1 Project*

*1.2 Description*

**Section 2 - Overview**

*2.1 Internal Software Data Structure*

*2.2 Global Data Structure*

*2.3 Temp Data Structure*

*2.4 Database Description*

**Section 3 - System Architecture**

*3.1 System Structure*

*3.2 Description for Pi*

*3.3 Description for Azure Cloud*

*3.4 Dynamic Behavior for Home Defense System*

**Section 4 – User Interface Design**

*4.1 Description of the User Interface*

*4.2 Interface Design Rules*

*4.3 Components Available*

**Section 5 – Restrictions, Limitations, and Constraints**

**Section 6 – Testing Issues**

*6.1 Classes of Tests*

*6.2 Expected software Response*

*6.3 Performance Bounds*

*6.4 Identification of Critical Components*

**Section 1 – Project Description**

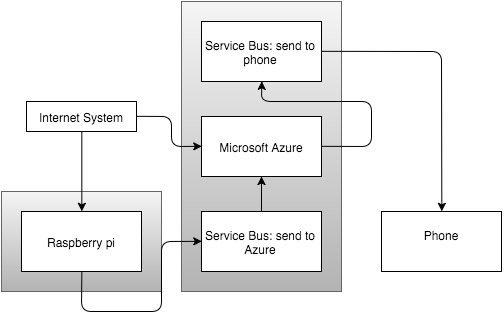
* 1. **The Home Defense System**

**1.2**

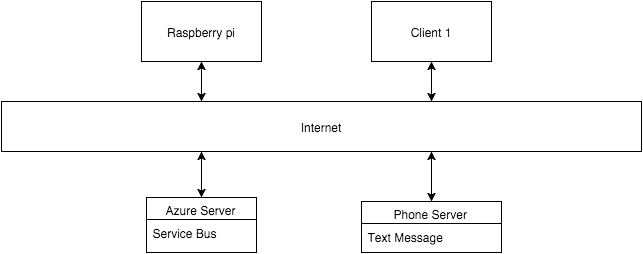
The Home Defense System is an alarm system using the Raspberry Pi and the cloud. By using this system and triggering special alarms for the user we are able to detect motion, take pictures of the motion detection and send it currently to Drop Box, and also we are able to retake the temperature every minute to make sure there is not a fire or a pipe burst. By having a system that is a fair price and easy to use we are able to give the middle class consumer some home security without having to break the bank. Our consumer will be able to protect their home and their loved ones by simply taking a brief moment to set up the Pi.

**Software Design**

We have a few different systems we will be using in our project. We use an internet service, the raspberry pi, and Microsoft Azure. All of the systems must communicate with each other in their own way. The pi must talk to the cloud, and the pi cloud must talk to the users phone. In order for the systems to be able to send and receive all of their data they must have internet or the fetch will fail.



Our system will be following the Client-Server architecture. We rely on our services to communicate with each other via the internet so it fits our needs the best.



2. Data design

2.1 Internal software data structure

The Home Defense System internal data structure is divided into two parts: The raspberry pi temperature trigger and raspberry pi motion detection trigger.

1. In the raspberry pi local memory, motion detection trigger will set as Boolean type data. It will send true when the camera detection detect a motion happened. After cloud receive trigger, the cloud will decide to send the break-in alarm to the user or not.
2. In the raspberry pi local memory, temperature detection trigger will set as Boolean type data. It will send true when the temperature detection detect the temperature is higher than user set value. After cloud receive trigger, the cloud will decide to send the fire alarm to the user or not.

2.2 Global data structure

Once the detection detect a motion or aberrant temperature, it will create a data file which contain the motion frame or temperature log. Then this file will send to cloud and finally send to user device.

2.3 Temporary data structure

The temperature and frame capture by detection each second are temporary data structure which the data will disappear after the algorithm analyze them.

For the snap of frame which detection think it is motion. The home defense system will create a frame file at local pi memory, then it will send to cloud and destroy at local pi memory.

2.4 Database description

In this project, we don’t use database.

3. Architectural and component-level design

3.1 System structure

The Home Defense System is broken up into two major components: a raspberry pi that data collection and analyze device, the cloud that policy engine and alarm sender system.

The cloud is a system that can receive pi’s trigger and put the trigger to policy engine. Then, decide send the alarm to the user device or not.

3.1.1 Architecture diagram

Home Defense System

Raspberry Pi

Azure Cloud

User Device

3.2 Description for Raspberry Pi

3.2.1 Processing narrative (PSPEC) for Raspberry Pi

The raspberry pi is a device that include two major detection, motion detection and temperature detection. Both detection will collection data and analyze data. After analyze, if there are motion indeed happened it will send the motion detection trigger to cloud. And if temperature aberrant, it will send the temperature trigger to cloud.

* + 1. Raspberry Pi interface description

The input interface of pi will divided by two parts. The one is the video capture by pi’s camera and another is the temperature collect by pi’s temperature detection.

The output interface of pi is the trigger of motion detect and temperature detect. Furthermore, if motion detect find motion happened, the snap frame of motion detection will send to cloud.

* + 1. Raspberry Pi processing detail

Firstly, Raspberry Pi should running under the cv virtual environment. Secondly, the Pi should connection the cloud. Thirdly, the motion detection python program and temperature detection c program should run. Finally, after everything run correctly, the Pi should send temperature trigger per second and motion trigger when motion was detected to cloud.

* + - 1. Design Class hierarchy for Raspberry Pi

Raspberry Pi

Motion Detection

Temperature Detection

Get Video

Send trigger

Get Tem.

Send trigger

Analyze Video

Analyze Tem.

s

s

s

* + - 1. Restrictions/limitations for Raspberry Pi

1. The Raspberry Pi only work under internet connected.
2. The Raspberry Pi need professional set up before user use.
   * + 1. Performance issues for Raspberry Pi

The Raspberry Pi will send trigger to cloud per second and motion trigger will only send to cloud when motion detected. If motion detected by detection, the FPS on live screen will lower 20 and frame will have delay.

* + - 1. Design constraints for Raspberry Pi

1. The Raspberry Pi’s camera is not night vision, thus the Pi only can install light controlled place.
2. The Raspberry Pi’s CPU is not good enough, thus the Pi will slower execution when large motion detected.
3. Slow WiFi network card may cause trigger send lag. Internet is suggested.
4. The Temperature Detection sometime has bad data collection. But still reliable at most time.
   * + 1. Processing detail for Motion Detection of Raspberry Pi

The Motion Detection is the core function of the Home Defense System.

3.2.3.5.1 Processing narrative for Motion Detection

The detection will get video by Raspberry Pi camera, and analyze each frame then detect weather the motion happened. Once motion detected is positive, then send trigger to the cloud with the snap picture file will created at Pi local memory, and send immediately to the cloud. After file sent, the local temporary file will deleted.

* + - * 1. Algorithmic model for Motion Detection

The background subtraction is the main algorithm of motion detection. The video will capture by Raspberry Pi camera, the pi\_surveillance.py python program will check each frame which capture by camera, program take the weighted mean of previous frames along with the current frame. Thus the background will dynamically adjust by lighting condition and static motion (like put a cup in front camera, the cup will detect as motion, but after cup stand there a while, the cup is not count motion anymore).

After motion detected by motion detection, a Boolean value will send as trigger to the cloud policy engine.

* + - 1. Processing detail for Temperature Detection of Raspberry Pi

The Temperature Detection can help user avoid fire hazard.

* + - * 1. Processing narrative for Temperature Detection

The Temperature Detection is built by DHT11 Humiture Sensor. This sensor can collection temperature and humidness per second. During the detection running, a temp\_log.txt file also write as temperature log for reference and back up. If the temperature higher than 150 degree in Fahrenheit, the temperature trigger will send to cloud.

* + - * 1. Algorithmic model for Temperature Detection

The Temperature Detection program write by c program. It provided by DHT11 sensor provider. The temperature and humidity will show on screen per second. Once the temperature higher than 150 degree, a Boolean value will send as trigger to cloud policy engine.

3.3 Description for Azure Cloud

3.3.1 Processing narrative (PSPEC) for Azure Cloud

Microsoft Azure is online software used to create various applications. We are choosing to use Azure for its Service Bus application. The Service Bus is used to fetch information from the pi and send a message to the user.

* + 1. Azure Cloud interface description

The interface is very user-friendly. Azure comes with a fully loaded dashboard that displays all of the users services, allowing them to easily select, modify, or delete anything they choose

* + 1. Azure Cloud processing detail
       1. Design Class hierarchy for Azure Cloud
       2. Restrictions/limitations for Azure Cloud

In order for the cloud to successfully get the information from the pi it must be connected to an Internet source. Also, the cloud costs money every time you fetch data. So the more data being fetched, the more money will be required to keep the service active.

* + - 1. Performance issues for Azure Cloud

Azure is very efficient. Although the fetch requests don’t get to Azure immediately, there is only a 5-10 minute delay, which is not of much concern.

* + - 1. Design constraints for Azure Cloud
      2. Processing detail for Operation 1 of Azure Cloud
         1. Processing narrative for Operation 1

When the Service Bus is created it gives the user a key for each operation. In order to successfully send data to the cloud the user must use those keys in a desired program to link to the cloud. Our system uses a python program to send a message to that key every time our test check is met.

* + - * 1. Algorithmic model for Operation 1

The Algorithm used for the sending operation is very simple. It consists of a for – loop that sends the current data to the cloud every time a check is failed. It doesn’t take much processing for the program to run these checks.

* + - 1. Processing detail for Operation 2 of Azure Cloud
         1. Processing narrative for Operation 2

After the cloud receives a message from the pi it will then be required to notify the user of the information gathered. In our case the cloud will send a text message to the user with a send operation.

* + - * 1. Algorithmic model for Operation 2

Much like the first operation, the algorithm needed to send the user a text message is very simple. It uses a while loop to and once the loop becomes true, it will send a text message to the user. The user however must have a valid phone number or the message cannot be sent.

* 1. Dynamic Behavior for Home Defense System

The Raspberry Pi interact Azure Cloud by trigger, and Azure Cloud interact Pi by service bus. And the Azure Cloud will send alarm to user device by internet.

3.4.1 Interaction Diagrams

Raspberry Pi

Microsoft’s Azure cloud

User Device

[ Motion\_Detection\_Trigger == True ]

Motion\_Alarm\_on()

Data\_analyse()

allt

Alarm\_reveived\_confirmation

Send\_snap(snap\_frame)

[ Temp\_Detection\_Trigger == True ]

allt

Temp\_Alarm\_on()

Alarm\_reveived\_confirmation

Send\_Trigger ()

Loop

4.0 User interface design

The user will be able to choose from a range of temperature values so that they can specifically set a value that when exceeded, the user will receive an SMS message. The user can also set a certain time for the Pi to look for motion detection.

(i.e. If temp > 90 send SMS)

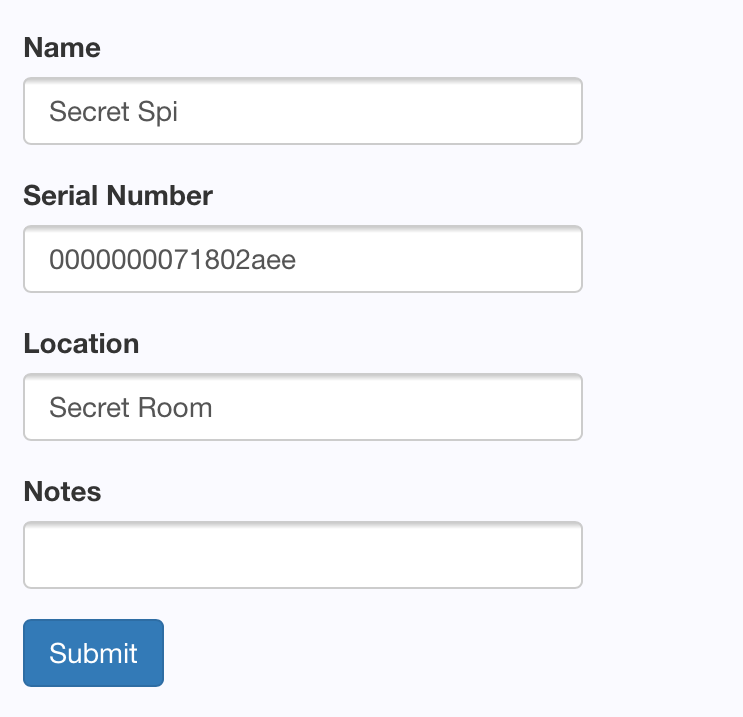
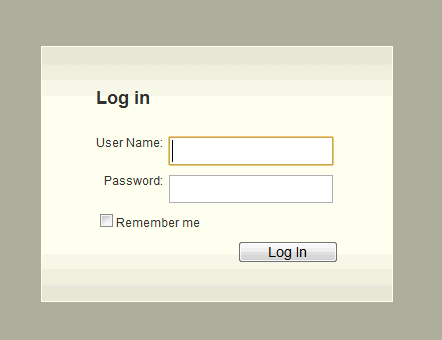
(i.e. RecordTime = 11:00 a.m. – 1:00p.m.)

4.1 Description of the user interface

The user will be able to log on to the Cloud service and set values for temperature and time.

4.1.1 Screen images

Standard Log on GUI. We also provide an interface to register your Raspberry Pi.



4.1.2 Objects and actions

The user will be required to sign up for the Home Defense System. After signup they will be taken to a page where they can register their Pi, and from there they can modify the settings for when the Pi will record video and when the Pi will record temperature. They can also modify the criterion that needs to be met in order to receive an SMS alert.

4.2 Interface design rules

The user will *not* be able to enter non-sensible data for the temperature and motion detection values. (i.e. Temperature = 2000) (Time = 1 p.m. – 12 p.m.)

4.3 components available

The main GUI component will be hosted through Microsoft Azure.

4.4 UIDS description

*N/A*

5.0 Restrictions, limitations, and constraints

Some hardware limitations that we have to work around is the limited processing power of the Raspberry Pi itself, along with the limited memory. While the memory is more than sufficient enough for our programs to run on, the picture data generated will have to be continuously deleted to make room for new data to be stored.

We are constrained by using Microsoft Azure for the cloud services, which means that the Raspberry Pi also requires a constant internet connection.

6.0 Testing Issues

We will mainly focus on testing the policy engine and the ability for the user to define policies. There will be preliminary tests about data transfer from the Raspberry Pi to the cloud and the cloud to the user, data storage, and data deletion after a specified time.

6.1 Classes of tests

In terms of black-box testing, we will test the ability of the system as a whole to transfer data and alert the user when necessary. The inputs will be either motion or temperature detected and then alerting the user. White box testing will be done, mainly with making sure that our system can handle any errors that bad input from the camera or temperature sensor would give. Also, making sure that our logic checking is correct for the user defined policies.

6.2 Expected software response

We expect the software to correctly alert the user if motion or temperature triggers are sent during the times that the user wishes to be alerted, as per their policies. Otherwise, the system should not falsely alert the user. If the user tries to enter a logically invalid policy, the system should inform them and allow them to change the policy.

6.3 Performance bounds

After motion is detected or temperature is detected outside of its acceptable values and the policy conditions are met, the user should be notified within 10 seconds.

6.4 Identification of critical components

The input devices on the actual Pi itself will need testing and checking to ensure that erroneous values are properly handled. The only other critical part is checking the user defined policies to validate them to avoid false alerts being sent.

**Task 3**

We chose the MIT license because we wanted our software to be open source. Open source gives the opportunity for programmers to make better programs and that’s exactly what our team wants to see. If someone can make our product better, we’re all for it. As a team we want to share what we’ve learned with the programming community, that’s why this license works best for us.

**----------------------------------------------------------**

**The MIT License (MIT)**

The MIT License (MIT)

Copyright (c) <year> <HomeDefenseSystem >

Permission is hereby granted, free of charge, to any person obtaining a copy  
of this software and associated documentation files (the "Software"), to deal  
in the Software without restriction, including without limitation the rights  
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
copies of the Software, and to permit persons to whom the Software is  
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in  
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,  
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE SOFTWARE.