CHAPTER 1

INTRODUCTION

This chapter gives brief description about the objectives of project. System overview and contribution are discussed. General idea of the project is presented in this chapter.

1.1 Introduction to autonomous security

With the advent of new equipment, the house breaching capabilities of intruders have vastly increased. As a countermeasure, passive security systems have been implemented by home owners and corporate offices. While these systems provide adequate security, they fail to differentiate between potential intruders and permitted individuals. Another area commercial systems lack is reporting the activity of area under surveillance in real time to the user, when the user is remotely located away from the property.

The aim of this project is to design an automated security system which leverages the combined capabilities of available security modules and computer vision libraries. The motivation of this project is that privacy and security have become the cornerstone of peaceful life in urban areas, which still needs a greater level of development and tighter integration of available modules. In this project, the status of the security system will be reflected in email being sent to the property owner and other concerning individuals. A Raspberry Pi will be used for handling image processing and password authentication part of the security system. An ATmega 328 microcontroller is responsible for Radio-frequency identity card(RFID) authentication and controlling the state of ambient lights. The system will get information about human presence from passive infra-red(PIR) module. A comprehensive storage structure has been implemented to store entry logs.

An automatic presence detection system comprised of passive infra-red sensor and one or more camera modules has been implemented. The processing circuits have switching devices for handshaking with each other through general purpose input output pins. A display monitor displays data about the overall system status. An email-implementation generates alert messages in case the system detects a breach or potential harm to the existing setup.

1.2 Main objectives

- To ensure security of house through authentication procedures
- To maintain log of accesses and breaches using computer vision and image processing
- To improve property owner's awareness about activity in and around his/her property

1.3 Thesis Organization

This thesis is based on computer vision and physical actuation circuits. The Chapter 1 Introduction gives brief description about the objectives of project. System overview and contribution are discussed. General idea of the project is presented in this chapter. Chapter 2 Review of literature gives brief description about the survey related to the area, security and various key management techniques that are studied by many researchers. Chapter 3 Work Done gives brief description about the design of the project i.e. full project view and modules and it also describes the overall implementation of each module. Chapter 4 Results and Discussion gives overview of the results achieved by the system, and suggests ways and means for further development. Chapter 5 gives a summary of the project and a brief conclusion derived from the outcomes. Chapter 6 Appendix presents the information, inclusion of which, in the prior chapters would have otherwise broken the flow of data being presented. Chapter 7 Literature Cited lists the various books and papers referred while designing the system.

CHAPTER 2

REVIEW OF LITERATURE

This chapter gives brief description about the survey related to the area, security and various key management techniques that are studied by many researchers.

2.1 Need of the System

In the real estate sector, security plays a pivotal role in determining value of property as well as level of safety available to the user. The security solutions available involve continuous storing of large amounts of data, most of which is never required. Also, most of the solutions available in the domestic market lack decision making skills when the prime owner is away. This presents a need for a smarter system that can think and make decisions by itself and store and provide filtered and curated data regarding the property to the owners and concerned security services like guards, police etc operating in the region. This requires tight integration and well defined interlinking of physical identity checking devices and proper image processing on the incoming camera feed, which is found to be lacking in available systems and has been implemented in the project. As a solution to the mentioned problems a new system has been devised and implemented.

2.2 Background

2.2.1 Field of invention

The system relates to an automatic home monitoring system, particularly, but not exclusively, for monitoring the house or any designated building to provide an indication or alarm when the security of an area is breached based on constraints determined by the state of sensors placed on or under doors, windows etc. and area scanned and monitored by the passive infrared sensor array. It is pointed out that the automated security system of present invention can include a plurality of presence sensing passive infrared sensors and photo capturing as well as video recording cameras associated with a plurality of buildings requiring security, and is not necessarily restricted for use with only buildings.

2.3 Description of Prior Act

Home security systems are known but have various disadvantages which the presented system overcomes or substantially reduces. They require large hard drives for storing the continuously recorded video footage. There is no available method for the system to know when to start recording and when to stop. The user, in case of breach, has to filter out a lot of image and video data to get to the relevant parts of it. Due to the large size of recorded data, and it's continuous nature, it's not possible to report breaches to the owner when situated at a remote place. Thus, due to lack of cross-checking window, there can be false alarm which can cause inconvenience to the owners as well as emergency response services. Also, due to the continuous operating nature of the systems, the power consumption is quite high. The systems consume a lot of power and can't be sustained for longer periods of time on batteries in case of power failures.

Due to a lack of a proper reporting system, owners can't be truly worry free regarding the security of premises. In case of forced gunpoint entry, the owner has no means of quietly reporting the incident to those present nearby and to emergency response teams without raising alarm for the intruder. Furthermore, known prior act does not provide sufficient warning time for the response mediums to react in time and help the individuals at risk.

2.4 Summary of Emergence of the System

It is a feature of the present invention to overcome all of above-mentioned disadvantages of the prior act. Another feature of the present invention is to automatically store cropped faces of individuals visiting the property locally as well as keep it's redundant copy on the cloud. This database can be used for facial training of the system. The data logs over a period of 24 hours can be transmitted by batch directly to intended authorities if required.

According to above features, from a broad aspect, the present invention provides an automated real time security system which comprises a central image processing module and a secondary physical authentication module, along with image capturing modules

connected in parallel if required in multiple numbers. Each image capturing module progressively scans the area for faces to be tracked and stores. Access output means is provided through virtual desktop for locally accessing stored data if required. Display means is provided through a connectable monitor port. Alarm means is provided an email server or through input/output pins to which alarm device can be easily connected.

CHAPTER 3

WORK DONE

This chapter gives brief description about the design of the project i.e. full project view and modules and it also describes the overall implementation of each module.

3.1 Design and implementation

The hardware design is made by using for modules aling with some allied circuitry. The main modules are:

- Module 1: Image processor
- Module 2: Radio frequency identity card reader and checker
- Module 3: Password entry membrane keyboard module
- Module 4: Image capturing device

These circuits are installed in the system as shown.

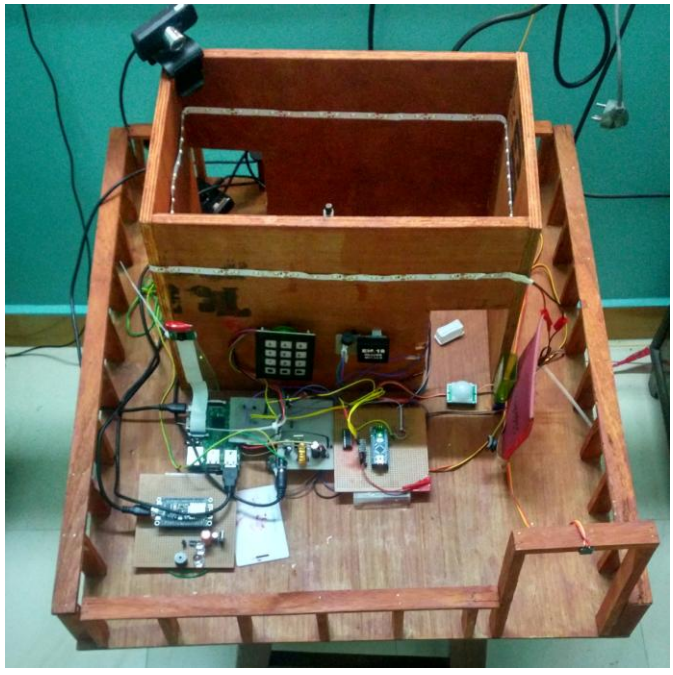


Fig. 3.1.1 Overall System View

The image processor controls 2 cameras as a time, which can be quickly increased as per requirement. It is due to the use of Raspberry Pi Model 3B.

3.2 Logic Flowchart

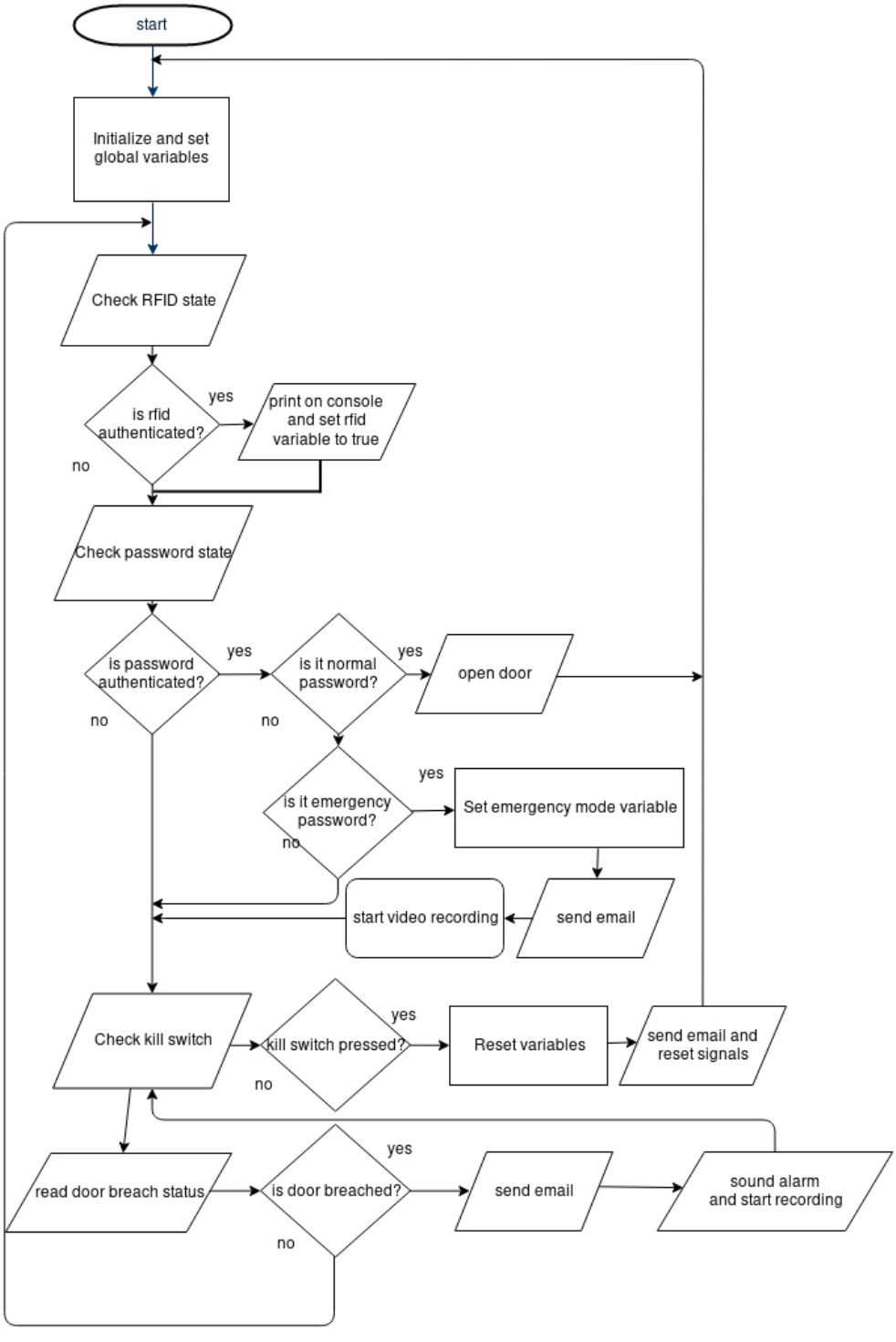


Fig. 3.1.2 Raspberry Pi Logic Flowchart

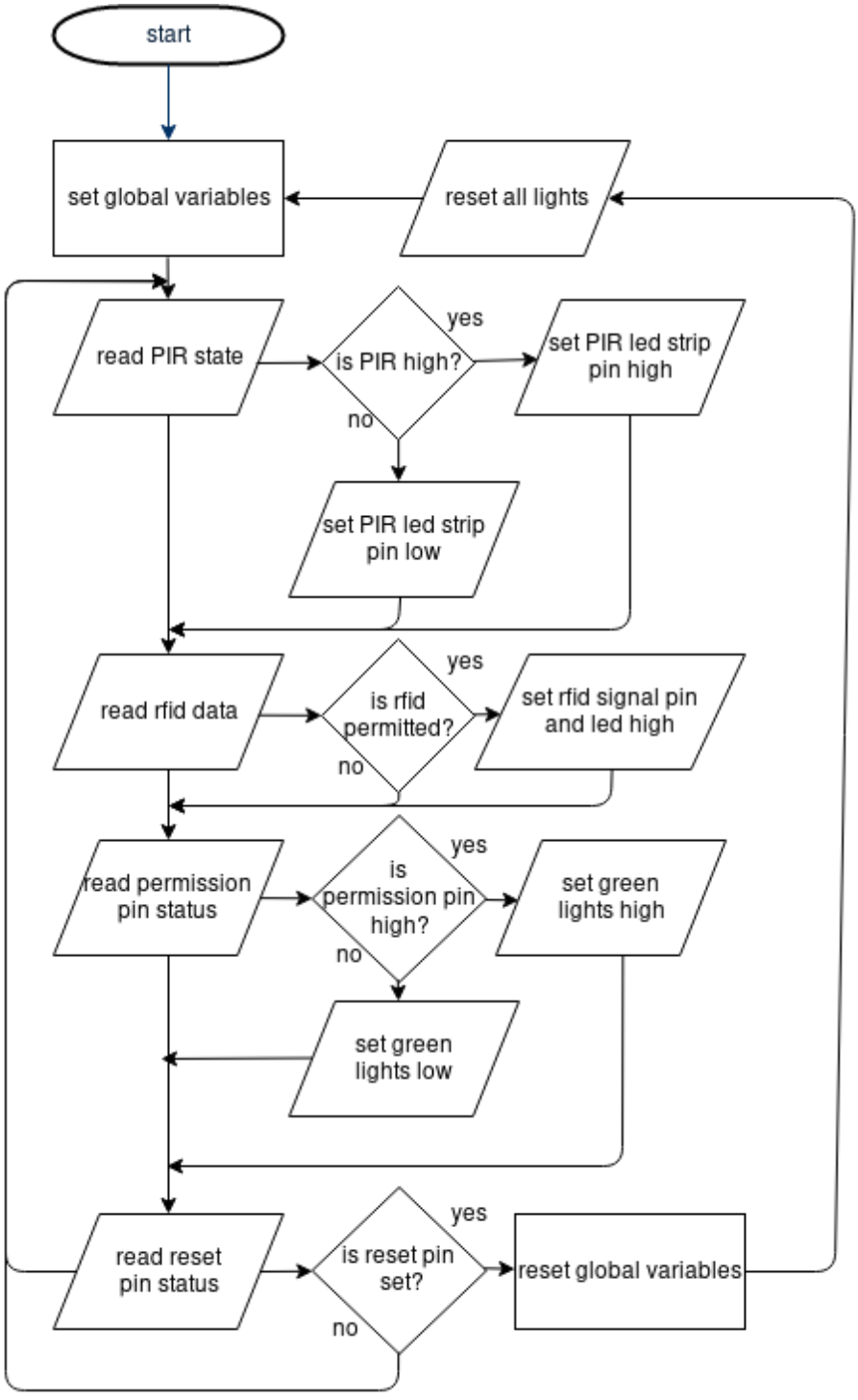


Fig. 3.1.3 ATmega 328 Logic Flowchart

3.3 Design Platform

3.3.1 Layout planning

The layout of printed circuit board has to incorporate all information on the board. The planning procedure depends upon many factors. These factors include operation of circuits, behavior of components in different situations and environmental situations and temperatures. The layout of the design should be prepared keeping in mind final output of the circuit and actual installation space on the printed circuit board.

3.3.2 Layout Designing Procedure

First rule is not to start design of layout unless an absolutely clear circuit diagram is available with the component list. Another important rule is to prepare over having every printed circuit board layout from the component side, this will minimize any further complications. While mounting the components, large ones should be placed first and placed in such a number that de-soldering of other components is not necessary in case of replacement. Following precautions should be taken while designing:

- Diameter of components holes usually in a given printed circuit board are in range of 3.8 mm to 4.8 mm in transistor pads. Minimized spacing between conducting lines must be produced. When the supply lines are to be put on the layout, extreme care should be taken, width of such lines and patterns in which they are distributed is of high importance to stability of circuit voltage, on suitable supply voltage of system can be done. Similarly maximum possible current on printed circuit board should not disturb the design of conducting path and width.
- Special attention should be given to design on 3.3v based circuit boards. Special keys are introduced during transition from one voltage level to other. Space required by components should be carefully rated and accounted for.
- A high package density printed circuit board looks very attractive and gives the impression of optimum design, but in most cases a high package density is less reliable and costly. Switching and fault finding becomes complicated if the package density is very high. After completion of design, a thorough check should be carried out.

3.3.3 Schematics

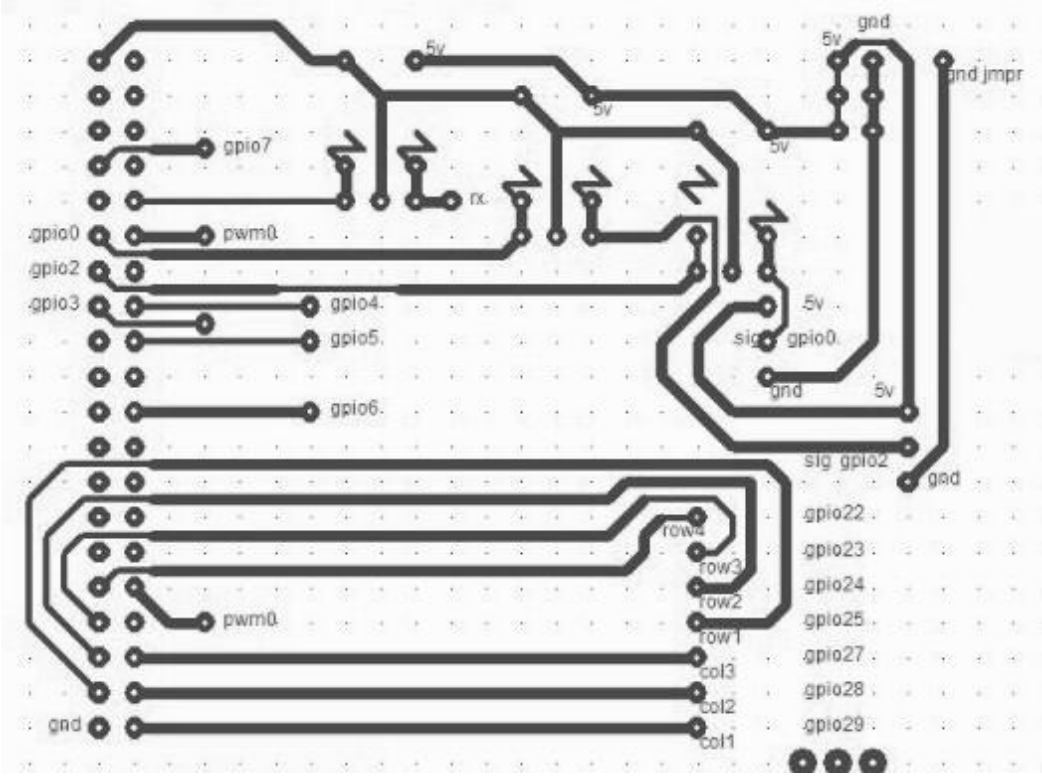


Fig. 3.2.3.1 Raspberry Pi Breakout Board Schematic

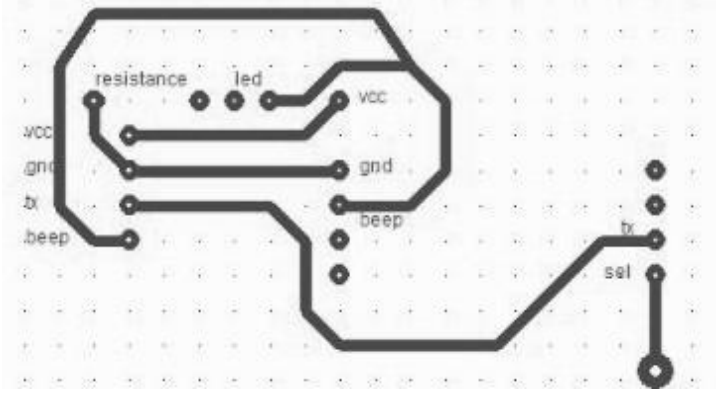


Fig. 3.2.3.2 EM18 Schematic

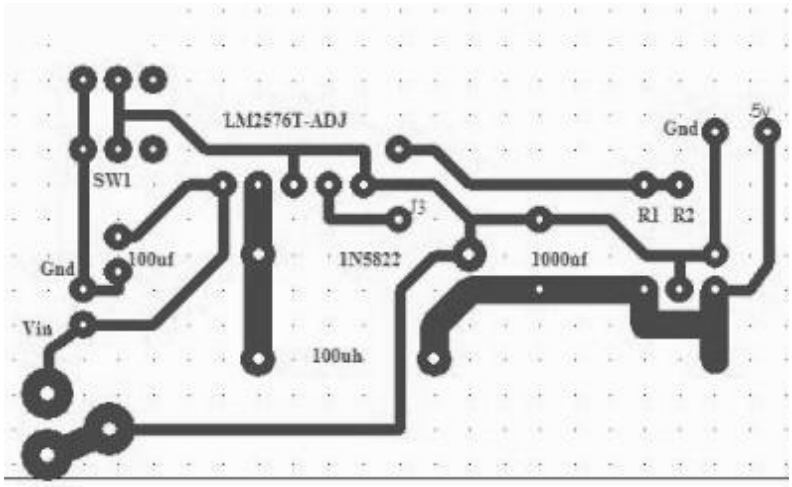


Fig. 3.2.3.3 Buck Converter Schematic

3.3.4 Etching and Rising

The printed circuit board design of circuit is printed on surface of printed circuit board by using screen printing technique. The printed circuit board surface is completely cleaned for any carbon content which may cause dry soldering of components. Then the printed circuit board is put in Ferric Chloride solution. The printed circuit board is dipped completely in etching solution and heated at the same time.

After etching is over, Ferric chloride contaminated surface must be cleaned, the usual practice followed is the use of water ring lib of 1%. Then oxalic acid is poured upon the printed circuit board followed by immediate rinsing with water.

3.3.5 Drilling

Drilling of mechanical holes for mounting is most important operation. The printed circuit board surface offers laminated resistance after etching. Hence drilling by all professional grade manufacturers and labs to compensate for the laminated surface is done with a 0.05mm drill bit. The drill bits may become blunt if used for glass epoxy printed circuit boards. Hence drill made of tingsten carbide are widely used for printed circuit board drilling operation, even for paper phenol lamination.

3.3.6 Component mounting

Component mounting is a very critical process. Performance and reliability are very much related to the proper maintaining of components. Following are the consideration of components:

- While bending, leads must be taken to ensure minimum strain on solder joints. Bending must not cause damage to component of lead.
- Hot mounted resistance must not touch board to avoid strain on solder joint.
- Vertical mounted resistance should touch the board to avoid touch strain on solder joints.
- Copper must be insulated to avoid possible short circuit.
- Component dispatching more heat should be provided with heat sink. This will avoid thermal strain on laminate and ensure cooling of components.

3.3.7 Component soldering

Components are soldered by supplying heat to the soldering material at the leads of components. The soldering iron consists of insulating handle connected via a metal shrank to the bit. The function is to store the heat to convey to the components to store and deliver molter solder and flux. To remove surplus solder from joint soldering iron with low, 20w, 30w, 50w are available. For electronintegrated circuits, 25w iron is used generally with soldering wire of Sn=60% and Pb= 40% must be used. There are basically two types of soldering techniques viz. Manual soldering iron and mass soldering with solder droplet dispensing machines.

3.4 Methodology

3.4.1 Microprocessor Raspberry Pi Model 3B

Fig. 3.4.1.1 Raspberry Pi Model 3B

Figure 3.3.1 Raspberry Pi model 3BThe raspberry pi is the heart of proposed embedded system. It performs image processing over the OpenCV computing platform. Microprocessors are used in autonomous devises so they can perform complex processing tasks in short amount of time.

Features of Raspberry Pi Model 3B are:

Specifications:

- **Processor**: Broadcom BCM2387 chipset. 1.2GHz Quad-Core ARM Cortex-A53 802.11 b/g/n Wireless LAN and Bluetooth 4.1
- **GPU** Dual Core VideoCore IV® Multimedia Co-Processor
- **Memory** 1GB LPDDR2
- **OS** Boots from Micro SD card, running a version of the Linux
- **Power** Micro USB socket 5V1, 2.5A

Connectors

- **Ethernet** 10/100 BaseT Ethernet socket
- **Video Output** HDMI (rev 1.3 & 1.4 Composite RCA
- **Audio Output** Audio Output 3.5mm jack, HDMI USB 4 x USB v2.0 Connector
- **GPIO** 40-pin 2.54 mm (100 mil) expansion header: 2x20 strip
- **Camera** 15-pin MIPI Camera Serial Interface (CSI-2)
- **Display** Display Serial Interface (DSI) 15 way flat flex cable
- **Memory** Push/pull Micro SDIO

- I/O 48x general purpose ports
- Bus 2x I2C bus
- 2x SPI lanes
- Serial port 2x UART
- Storage 2x SD/SDIO
- Usb 1x USB2 HOST/OTG
- 1x 4-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 2-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 4-lane DSI Display Interface (up to 1Gbps per lane)
- 1x 2-lane DSI Display Interface (up to 1Gbps per lane)
- ARMv6 (CM1) or ARMv7 (CM3, CM3L) Instruction Set

3.4.2 Microcontroller ATmega 328

The ATmega 328 is the second most important processing device of the proposed system. Micro controllers are used in automatically controlled products and devices such as automobile engines control systems, medical devices , office machines, power tools etc. By reducing the size and cost compared to a design that uses a separate microprocessor, memory and input/output devices, micro controllers make it economical to digitally control even more devices and processes.

Some micro controllers may use four bit words and operate at clock frequencies as low as 4 kHz, for low power consumption in order of milli-watts or micro-watts. They have the ability to retain functionality while waiting for an event such as a button press or other interrupt. Power consumption when sleeping may be in just nano-watts, making them well suited for long lasting battery applications.

At present some major manufactures are Microchip(PIC microcontrollers), Atmel (publication: AVR microcontrollers), Hitachi, Phillips, Maxim, NXP, Intel etc. Our intrest is in ATmega 328. It belongs to Atmel's AVR series micro controller family. Its features are:

Advanced RISC Architecture

- – 131 Powerful Instructions
- – Most Single Clock Cycle Execution
- – 32 x 8 General Purpose Working Registers
- – Fully Static Operation
- – Up to 20 MIPS Throughput at 20MHz
- – On-chip 2-cycle Multiplier

High Endurance Non-volatile Memory Segments

- – 32KBytes of In-System Self-Programmable Flash program
- Memory
- – 1KBytes EEPROM

- – 2KBytes Internal SRAM
- – Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
- – Data Retention: 20 years at 85°C/100 years at 25°C (1)
- – Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
- – Programming Lock for Software Security

Atmel ® QTouch ® Library Support

- – Capacitive Touch Buttons, Sliders and Wheels
- – QTouch and QMatrix ® Acquisition
- – Up to 64 sense channels

Peripheral Features

- – Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
- – One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- – Real Time Counter with Separate Oscillator
- – Six PWM Channels
- – 8-channel 10-bit ADC in TQFP and QFN/MLF package
- • Temperature Measurement
- – 6-channel 10-bit ADC in PDIP Package
- • Temperature Measurement
- – Two Master/Slave SPI Serial Interface
- – One Programmable Serial USART
- – One Byte-oriented 2-wire Serial Interface (Philips I 2 C compatible)
- – Programmable Watchdog Timer with Separate On-chip Oscillator
- – One On-chip Analog Comparator
- – Interrupt and Wake-up on Pin Change

Special Microcontroller Features

- – Power-on Reset and Programmable Brown-out Detection
- – Internal Calibrated Oscillator

- – External and Internal Interrupt Sources
- – Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and
- Extended Standby

I/O and Packages

- – 23 Programmable I/O Lines
- – 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

Operating Voltage

- 1.8-5.5 volts

Temperature Range:

- -40°C to 105°C

Speed Grade:

- – 0 - 4MHz @ 1.8 - 5.5V
- – 0 - 10MHz @ 2.7 - 5.5V
- – 0 - 20MHz @ 4.5 – 5.5V

Power Consumption at 1MHz

- 1.8V, 25°C
- – Active Mode: 0.2mA
- – Power-down Mode: 0.1µA
- – Power-save Mode: 0.75µA (Including 32kHz RTC)

Pin Count: The ATmega328 has 24 pins. Two for power(pin number 24: +1.8 or 5v, pin number 23: ground), two for oscillator, one for reset (pin 22), three for providing necessary power and reference voltage to its internal analog to digital converter and 12 normal input-output pins.

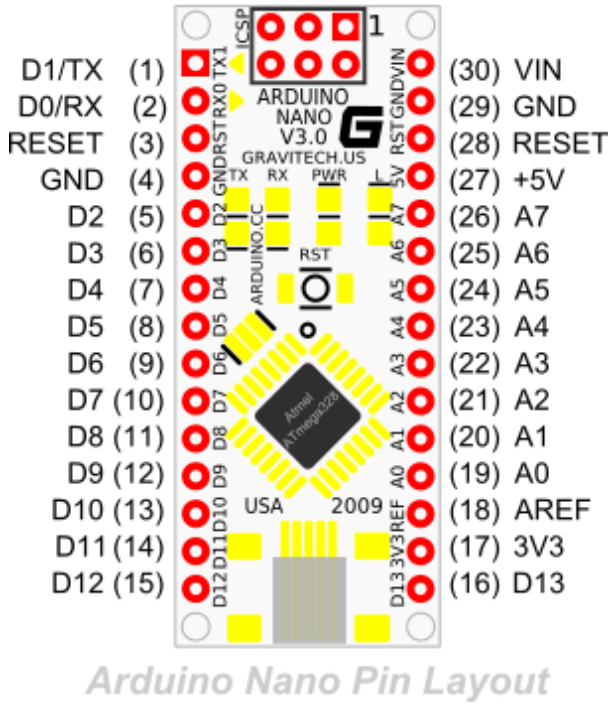


Fig. 3.4.2.1 Pin diagram of ATmega 328

About input/output pins: The ATmega328 is capable of handling analogue inputs. Pins 23 to 28 can be used as a single input channel to the internal analog to digital converter, plus a pair of pins AREF, AVCC, GND togeather can make an analog to digital converter channel.

No pins can perform and serve for two purposes at the same time. It's programmer's responsibility to resolve the conflict in the circuitry and the program. Programmers are advised to have a look to priority tables and internal configuration from the datasheet.

Analog to digital converter: It has 8 successive approximation type analog to digital converter in which total 8 single channels are selectable. Referance is selectable, wither an external reference can be used or internal 2.56v reference can be brought into action. The external reference can be connected to the AREF pin.

Communication options: ATmega 328 has two data transfer modules embedded in it. They are:

- Two wire interface
- USART (Universal Synchronous/Asynchronous Receiver/Transmitter)

3.4.3 Power Supply

A buck converter (step-down converter) is a Dc to Dc power converter which steps down voltage (while stepping up current) from its input (supply) to its output (load). It is a class of Switch mode power supply(SMPS) typically containing at least two semiconductors (a diode and a transistor although modern buck converters frequently replace the diode with a second transistor used for synchronous rectification) and at least one energy storage element, capacitor, inductor, or the two in combination. To reduce voltage ripple, filters made of capacitors (sometimes in combination with inductors) are normally added to such a converter's output (load-side filter) and input (supply-side filter)

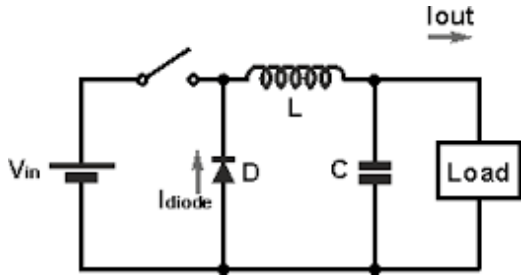
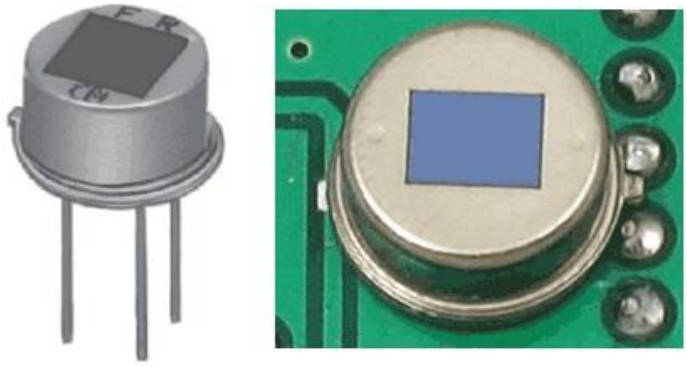


Fig. 3.4.3.1 Buck regulator

Switching converters (such as buck converters) provide much greater power efficiency as DC-to-DC converters than linear regulators, which are simpler circuits that lower voltages by dissipating power as heat, but do not step up output current.

Buck converters can be remarkably efficient (often higher than 90%), making them useful for tasks such as converting a computer's main (bulk) supply voltage (often 12V) down to lower voltages needed by microcontorollers, central processing units etc(1.8V or less), etc.



3.4.4 Passive Infrared Sensor

Fig. 3.4.4.1 Passive Infrared sensor

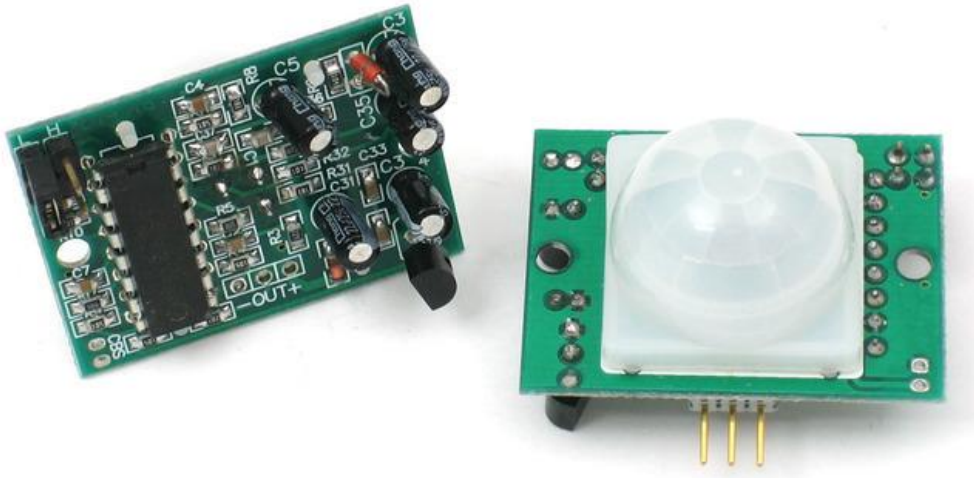


Fig. 3.4.4.2 Passive infrared sensor module overview

Sensor : The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR. The lens used converges radion upon the sensor so that the two

slots can 'see' out past some distance (basically the sensitivity of the sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.

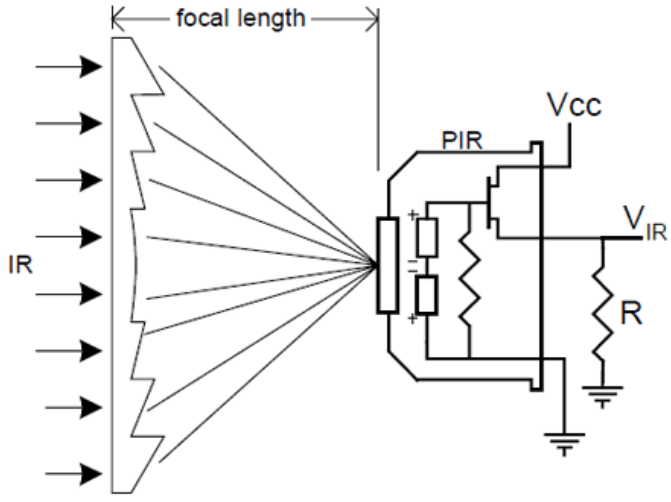


Fig. 3.4.4.3 Passive Infrared Sensor Lens

Lens : Passive Infrared sensors are rather generic and for the most part vary only in price and sensitivity. Most of the real work happens with the optintegrated circuits. This is a pretty good idea for manufacturing: the Passive Infrared sensor and circuitry is fixed and costs a few dollars. The lens can change the breadth, range, sensing pattern, very easily.

Operation: The module operates at +5v Vcc and can detect motion in a conical range of 0.3 to 7 at a 120 degree angle. The delay time of the sensor can be adjusted from 3 seconds to 5 minutes.

3.4.5 Raspberry pi Camera

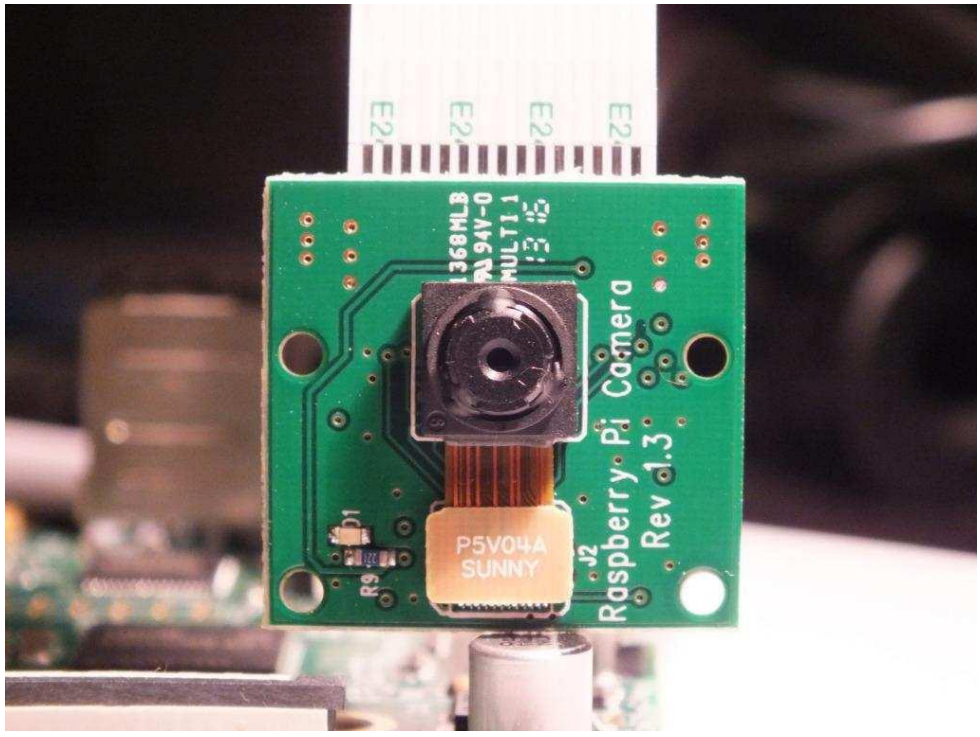


Fig. 3.4.5.1 Raspberry Pi Camera Module

Fig. 3.4.5.1 Raspberry pi camera moduleThe Raspberry Pi has had a connector on it to attach a camera to the GPU (the VideoCore 4 Graphintegrated circuits Processing Unit on the Raspberry Pi). This connection uses the CSI- 2 electrical protocol and is a standard used in most mobile phones. It is an extremely fast connection, which on the Raspberry Pi is capable of sending 1 080p sized images (1 920x1 080 x1 0bpp) at 30 frames per second, or lower resolution at even higher frame rates. It had always been intended at some point to release a camera module that could use this connection, as the ability to stream high speed video data through the GPU without any interaction with the ARM processor would always make the camera much more efficient than any USB attached webcam. It would also enable the use of the GPU's ability to encode H264 video, or JPEG images in hardware.

Hardware specification:

- **Size** Around 25 × 24 × 9 mm
- **Weight** 3g
- **Still resolution** 5 Megapixels
- **Video modes** 1080p30, 720p60 and 640 × 480p60/90

- **Linux integration** V4L2 driver available
- **C programming API** OpenMAX IL and others available
- **Sensor** OmniVision OV5647
- **Sensor resolution** 2592 × 1944 pixels
- **Sensor image area** 3.76 × 2.74 mm
- **Pixel size** 1.4 µm × 1.4 µm
- **Optical size** ¼"
- **Full-frame SLR lens equivalent** 35 mm
- **S/N ratio** 36 dB
- **Dynamic range** 67 dB @ 8x gain
- **Sensitivity** 680 mV/lux-sec
- **Dark current** 16 mV/sec @ 60 C
- **Well capacity** 4.3 Ke-
- **Fixed focus** 1 m to infinity
- **Focal length** 3.60 mm +/- 0.01
- **Horizontal field of view** 53.50 +/- 0.13 degrees
- **Vertical field of view** 41.41 +/- 0.11 degrees
- **Focal ratio (F-Stop)** 2.9

3.4.6 Em18 Radio Frequency Identity Module

Fig. 3.4.6.1 Em18 Module

Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags attached to objects. The tags contain electronically stored information. Passive tags collect energy from a nearby RFID reader's interrogating radio waves.

The EM-18 RFID Reader module operating at 125kHz is an inexpensive solution for your RFID based application. The Reader module comes with an on-chip antenna and can be powered up with a 5V power supply. Power-up the module and connect the transmit pin of the module to recieve pin of your microcontroller. Show your card within the reading distance and the card number is thrown at the output. Optionally the module can be configured for also a weigand output.

Specifications:

- 5VDC through USB (External 5V supply will boost range of the module)
- Current: <50mA
- Operating Frequency: 125Khz
- Read Distance: 10cm
- Size of RFID reader module: 32mm(length) * 32mm(width) * 8mm(height)

3.4.7 Membrane Keyboard

Fig. 3.4.7.1 Membrane Keyboard

Key Specifications

- Maximum Rating: 24 VDC, 30 mAInterface: 8-pin access to 4x4 matrix
- Operating temperature: 32 to 122 °F (0 to 50°C)

Dimensions

- Keypad, 2.7 x 3.0 in (6.9 x 7.6 cm)
- Cable: 0.78 x 3.5 in (2.0 x 8.8 cm)

Matrix keypads use a combination of four rows and four columns to provide button states to the host device, typically a microcontroller. Underneath each key is a pushbutton, with one end connected to one row, and the other end connected to one column. These connections are shown in **Fig. 3.4.7.2**

Membrane keyboards are prone to bouncing when the button is released, which has to be accounted for in programming the input to the microcontroller. This is done bu setting a delay to wait for the switch bouncing to stop. Generally, this delay is set to a minimum duration of 30 milliseconds.

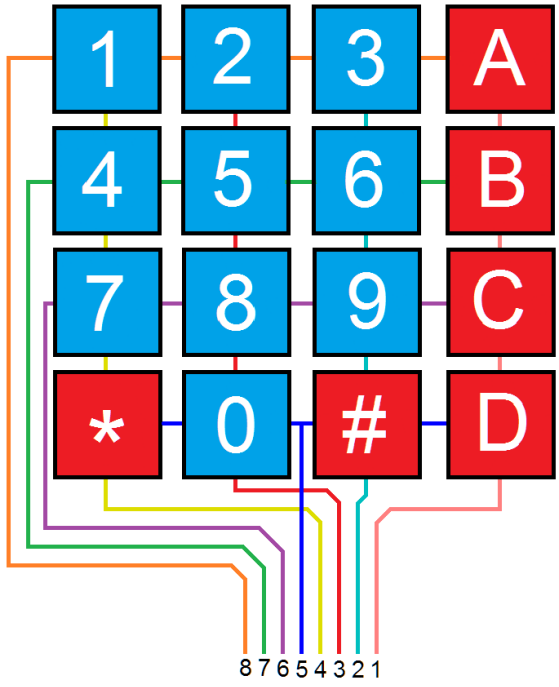


Fig. 3.4.7.2 Matrix Keypad Connections

In order for the microcontroller to determine which button is pressed, it first needs to pull each of the four columns (pins 1-4) either low or high one at a time, and then poll the states of the four rows (pins 5-8). Depending on the states of the columns, the microcontroller can tell which button is pressed. For example, say your program pulls all four columns low and then pulls the first row high. It then reads the input states of each column, and reads pin 1 high. This means that a contact has been made between column 4 and row 1, so button 'A' has been pressed.

3.5 Software Support

3.5.1 Express PCB

This program was used for systematic designing of the printed circuit board layout. Express PCB is easy to use Windows application for laying out printed circuit boards. There are two parts to Express PCB, Cad software and board manufacturing service. The CAD software includes Express SCH for drawing express schematintegrated circuits and Express PCB for designing printed circuit boards.

3.5.1.1 Placing components

The procedure to be followed for placing components is as follows:

- It's best to place parts only on top side of the board. When placing components, make sure snap to grid is turned on. Usually, a value of 0.025'' for the snap grid is best for this job. Place all the components that need to be in specific locations. This includes connectors, switches, mounting holes, heat sinks or any other item that mounts to an external location.
- Give careful thought when placing component to maximize trace lengths. Out parts next to each other that connect to each other. Doing a good job here will make laying traces much easier. Arrange integrated circuits in only one or two orientations: up and down, or right and left. Align each ic so that pin one is in same place for each orientation, usually on top or left sides.
- Position polarized parts (i.e diodes and electrolytic capacitors) with the positive lleads all having the same orientation. Also use a square pad to mark the positive leads of these components.
- One can save a lot of time by leaving generous space between integrated circuits for traces. Frequently the beginner runs out of room when routing traces. Leave 0.350'' to 0.500'' between integrated circuits. For large integrated circuits allow even more.
- Parts not found in the component library can be made by placing a series of individual pads and then grouping them together. Place one pad for each lead of the component. It is very important to measure the pin spacing and pin diameters as accurately as possible. Typically, dial or digital calipers are used for this job.

- After placing all components, print a copy of the layout. Place each component on top of the layout. Check to ensure that you have allowed enough space for every part to the rest without touching each other.

3.5.1.2 Placing Power and Ground Traces

After the components are placed, next step is to place power and ground traces. It is essential when working with integrated circuits to have solid power and ground lines, using wide traces that connect to common rails for each supply. It is very important to avoid snaking or daisy chaining power lines from part-to-part. One common configuration has been used in the project. The bottom layer is filled with ground plane for best results. Large traces feeding from a single rail are used for positive supply.

3.5.1.3 Checking Work

After all traces are placed, it is best to double check the routing of every signal to verify that nothing is missing or incorrectly wired. This is done by running through the schematic, one wire at a time. Carefully follow the path of each trace on your pc layout to verify that it is same as on your schematic. After each trace is confirmed, mark that signal on the schematic with a yellow highlighter. Inspect your layout, both top and bottom, to ensure that the gap between every item (pad to pad, trace to trace) is 0.007'' or greater. Use the pad information tool to determine the diameters of pads that make up a component. Check for missing via. Express PCB will automatically insert a via when changing layers as a series of traces are placed. Users often forget that via are not automatically inserted otherwise. An easy awy to check for missing via is to first print the top layer, then print the bottom. Visually inspect each side for traces that don't connect anything. When a missing via is found, one is inserted. This is done by by clicking on the pad in side toolbar , select a via from drop down list box and click on the layout where the via is missing. Metal components such as heat sinks, crystals, switches, batteries and connectors can cause shorts if they are placed over traces on the top layer. Inspect for these shorts by placing all the metal components on a printout of the top layer. Then look for traces that run below the metal components.

3.6 Design Platform

The software code is partly written in C language and partly in python. Simplicity of both languages make them easier for compiler to generate efficient machine-learning instructions. C offers efficient bit manipulation using bit wise operators. Python offers readability which helps in implementing the complex logical algorithms.

The Atmega 328 performs following operations in infinite loop:

The Raspberry Pi 328 performs following operations in infinite loop:

3.7 Algorithms and Implementation

3.7.1 Raspberry Pi main algorithm

The raspberry pi code consists of main sections as follows:

- Initialize all ports and variables
- Check status of conditionals and take decision
- Display status of ongoing operations on console

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import time
import random
import re
import os
import curses
import cv2
import smtplib
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(32, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(22, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

```
GPIO.setup(11, GPIO.OUT)
GPIO.setup(18, GPIO.IN)
GPIO.setup(16, GPIO.OUT)

ROW         = [31,33,35,37]
COL      = [36,38,40]
print('#################GPIO SET')
GPIO.setup(11, GPIO.LOW)
GPIO.setup(16, GPIO.LOW)
MATRIX = [  ['6','5','4'],
            ['3','2','1'],
            ['9','8','7'],
            ['','0','']     ]
print('#################INITIAL SETUP SUCCESSFUL')
normalPassword = '111';
emergencyPassword = '555';
inputPassword = '0000'
passwordAuthentication = False
faceAuthentication = False
rfidAuthentication = False
pirAuthentication = False
doorBreach = False
parametersPassed = 0
print('#################GLOBALS SET')

def readPassword():
    #reinitialize gpio
    for j in range(3):
        GPIO.setup(COL[j], GPIO.OUT)
        GPIO.output(COL[j], 1)
    for i in range(4):
```

```
        GPIO.setup(ROW[i], GPIO.IN, pull_up_down = GPIO.PUD_UP)
    for j in range(3):
            GPIO.output(COL[j], 0)
            for i in range(4):
                if GPIO.input(ROW[i]) == 0:
                    global inputPassword
                    print (MATRIX[i][j])
                    print('row is' + str(i) + 'col is' + str(j))
                    inputPassword = inputPassword + MATRIX[i][j]
                    print(MATRIX[i][j])
                    while (GPIO.input(ROW[i]) == 0):
                        pass
                        time.sleep(0.2)
                    return (MATRIX[i][j])
                    print('debouncing')
            GPIO.output(COL[j], 1)

def checkNormalPassword(inputPassword):
    global passwordAuthentication
    if normalPassword in inputPassword:
        print('Password matched')
        passwordAuthentication = True

def checkEmergencyPassword(inputPassword):
    if emergencyPassword in inputPassword:
        print('!!!!!!!!Emergency mode!!!!!!!!')
        sendEmergencyAlert()
        validateEmergency()
        sendEmergencyAlert()

def checkRfid():
```

```
    global rfidAuthentication
    rfid = GPIO.input(18)
    if rfid == 0:
        rfidAuthentication = False
        pass
    elif rfid == 1:
        time.sleep(0.2)
        rfid = GPIO.input(18)
        if rfid == 1:
            print('###########RFID FINALIZED')
            rfidAuthentication = True

def openDoor():
    print('door opened')

def closeDoor():
    print('door closed')
    try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login("emailid@provider.com", "emailPassword")

        msg = "Intruder ALERT at house!"
        server.sendmail("emailid@provider.com", "emailid@provider.com", msg)
        server.quit()
        print('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!')
        print('ALERT SENT VIA EMAIL')
        print('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!')
    except:
        print('smtp error in sendAlert()')
```

```
def sendEmergencyAlert():
    try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login("emailid@provider.com", "emailPassword")

        msg = "EMERGENCY EMERGENCY AT HOUSE NUMBER 10!"
        server.sendmail("emailid@provider.com", "emailid@provider.com", msg)
        server.quit()
        print('eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee')
        print('EMERGENCY ALERT SENT VIA EMAIL')
        print('eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee')
    except:
        print('smtp error in sendEmergencyAlert()')

def disableAlert():
    try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login("emailid@provider.com", "emailPassword")

        msg = "killSwitch used!"
        server.sendmail("emailid@provider.com", "emailid@provider.com", msg)
        server.quit()
        print('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!')
        print('ALERT HAS BEEN DISABLED')
        print('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!')
    except:
        print('smtp error in disableAlert()')

def killSwitch():                   #once inside the house
```

```
    global passwordAuthentication       #use global variable
    global faceAuthentication           #use global variable
    global rfidAuthentication           #use global variable
    global pirAuthentication            #use global variable
    global inputPassword                #use global variable
    global doorBreach                   #use global variable

    closeDoor()
    killSignal()

    passwordAuthentication = False      #reset variables
    faceAuthentication = False
    rfidAuthentication = False
    pirAuthentication = False
    doorBreach = False
    inputPassword = '0'
    disableAlert()
    os.system('clear')
    print('########')
    print('NEW ITERATION########')
    print('########')

def killSignal():
    GPIO.setup(11, GPIO.HIGH)       #kill signal to nano
    print('----------------------NANO SYNC DELAY')
    time.sleep(3)                   #sync time
    GPIO.setup(11, GPIO.LOW)
    print('----------------------NANO REACTIVATION DELAY')
    time.sleep(5)                   #nano activation time
```

```
def checkKillSwitch():
    killSwitchStatus = GPIO.input(22)
    if killSwitchStatus == 1:
        pass

    elif killSwitchStatus == 0:
        time.sleep(0.5)
        killSwitchStatus = GPIO.input(22)

        if killSwitchStatus == 0:
            print('!!!KILL SWITCH PRESSED!!!')
            GPIO.setup(11, GPIO.HIGH)
            time.sleep(3)
            killSwitch()
            GPIO.setup(11, GPIO.LOW)

def checkDoorBreach():
    doorBreachStatus = GPIO.input(32)
    if doorBreachStatus == 0:
        pass

    elif doorBreachStatus == 1:
        time.sleep(0.5)
        doorBreachStatus = GPIO.input(32)

        if doorBreachStatus == 1:
            GPIO.setup(16, GPIO.HIGH)
            time.sleep(0.2)
            GPIO.setup(16, GPIO.LOW)
            time.sleep(0.2)
            print('BREACH BREACH BREACH BREACH')
```

```
GPIO.setup(16, GPIO.HIGH)
time.sleep(0.2)
GPIO.setup(16, GPIO.LOW)
time.sleep(0.2)
print('BREACH BREACH BREACH BREACH')
GPIO.setup(16, GPIO.HIGH)
time.sleep(0.2)
GPIO.setup(16, GPIO.LOW)
print('BREACH BREACH BREACH BREACH')
time.sleep(0.2)
print(' ')
print(' ')
print(' ')
print(' ')
print(' ')
print(' ')
print(' ')
print(' ')
print(' ')
print(' ')
sendBreachAlert()
GPIO.setup(16, GPIO.HIGH)
time.sleep(0.2)
GPIO.setup(16, GPIO.LOW)
time.sleep(0.2)
print('checking faces')
os.system('python breachFacechopEmail.py')
print('done checking faces')
print('@@@@@@@@@@@CAPTURING VIDEO@@@@@@@@@@@')
captureVideo()
```

```
            print('+++++++++CAPTURING VIDEO DONE +++++++++++++++++')

            print('breachSLeep')
            time.sleep(5)
            print('breachSLeep ENDS')
            os.system('clear')
            print('########')
            print('NEW ITERATION########')
            print('########')

def captureVideo():
    timestr = time.strftime("%Y%m%d-%H%M%S")
    fileName = timestr + '.avi'
    print (timestr)
    cap = cv2.VideoCapture(0)
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter(fileName,fourcc, 20.0, (640,480))

    testInput = '0'

    while(cap.isOpened()):
        ret, frame = cap.read()
        if ret==True:
            frame = cv2.flip(frame,0)
            out.write(frame)
            cv2.imshow('frame',frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        else:
            break
        killSwitchStatus = GPIO.input(22)
```

```
        if killSwitchStatus == 1:
            pass

        elif killSwitchStatus == 0:
            time.sleep(0.5)
            killSwitchStatus = GPIO.input(22)

            if killSwitchStatus == 0:
                print('!!!KILL SWITCH PRESSED!!!')
                GPIO.setup(11, GPIO.HIGH)
                time.sleep(3)
                killSwitch()
                break
                GPIO.setup(11, GPIO.LOW)
    cap.release()
    out.release()
    cv2.destroyAllWindows()

def sendBreachAlert():
    try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login("emailid@provider.com", "emailPassword")

        msg = "Intruder ALERT at house!"
        server.sendmail("emailid@provider.com", "emailid@provider.com", msg)
        server.quit()
        print('ALERT ALERT ALERT ALERT')
        print('BREACH ALERT SENT VIA EMAIL')
        print('BREACH ALERT SENT VIA EMAIL')
    except:
```

```
        print('smtp error in sendBreachAlert()')

def validateEmergency():
    openDoor()
    print('all ok!Going for the kill')
    print('green zone notice')
    GPIO.setup(16, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.LOW)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.LOW)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.LOW)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.LOW)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.LOW)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.setup(16, GPIO.LOW)
    time.sleep(0.5)
```

```
    print('checking faces')
    os.system('python emergencyFacechopEmail.py')
    print('done checking faces')
    openDoor()
    GPIO.setup(16, GPIO.HIGH)
    print('green zone')
    time.sleep(5)
    print('green zone ends')
    GPIO.setup(16, GPIO.LOW)
    killSwitch()
    closeDoor()
    print('done EMERGENCY ENTRY')

def validateParameters():
    global passwordAuthentication
    global rfidAuthentication
    parametersPassed = 0
    if(passwordAuthentication == True):
        parametersPassed += 1
    if(rfidAuthentication == True):
        parametersPassed += 1

    if(parametersPassed >= 2):
        openDoor()
        print('all ok!Going for the kill')
        print('green zone notice')
        GPIO.setup(16, GPIO.HIGH)
        time.sleep(0.5)
        GPIO.setup(16, GPIO.LOW)
        time.sleep(0.5)
        GPIO.setup(16, GPIO.HIGH)
```

```
        time.sleep(0.5)
        GPIO.setup(16, GPIO.LOW)
        time.sleep(0.5)
        GPIO.setup(16, GPIO.HIGH)
        time.sleep(0.5)
        GPIO.setup(16, GPIO.LOW)
        time.sleep(0.5)
        GPIO.setup(16, GPIO.HIGH)
        time.sleep(0.5)
        GPIO.setup(16, GPIO.LOW)
        time.sleep(0.5)
        GPIO.setup(16, GPIO.HIGH)
        time.sleep(0.5)
        GPIO.setup(16, GPIO.LOW)
        time.sleep(0.5)
        GPIO.setup(16, GPIO.HIGH)
        time.sleep(0.5)
        GPIO.setup(16, GPIO.LOW)
        time.sleep(0.5)

        openDoor()
        GPIO.setup(16, GPIO.HIGH)
        print('green zone')
        time.sleep(5)
        print('green zone ends')
        GPIO.setup(16, GPIO.LOW)
        killSwitch()
        closeDoor()
        print('done validate-killing')
time.sleep(1)
print('W E L C O M E')
```

```
print('T O')
print('A R I S E')
print('########BEGIN########')
while True:

    #working
    checkRfid()
    checkKillSwitch()
    readPassword()
    checkNormalPassword(inputPassword)
    checkEmergencyPassword(inputPassword)
    checkDoorBreach()
    #print('done')
    #inProgress
    validateParameters()
```

3.7.2 Breach State Face Crop and Email algorithm

In case of breach, the following code takes snapshot from the primary cameras, detects faces, crops out faces and emails it to the owner:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import cv2
import os
import smtplib
import time
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart


def breachFacechopEmail(image):
    print('############################################################')
    print('@breachFacechopEmail taking EMERGENCY snap')
    timestr = time.strftime("%Y%m%d-%H%M%S")
    print (timestr)
    imageName = timestr +'.jpg'
    os.system('raspistill -w 1920 -h 1080 -n -q 20 -o '+ timestr +'.jpg')
    facedata = "haarcascades/haarcascade_frontalface_default.xml"
    cascade = cv2.CascadeClassifier(facedata)


    img = cv2.imread(imageName)
    print('@breachFacechopEmail image loaded')
```

```
minisize = (img.shape[1],img.shape[0])
miniframe = cv2.resize(img, minisize)

faces = cascade.detectMultiScale(miniframe)
msg = MIMEMultipart()
msg['Subject'] = 'BREACH AT HOUSE WAS DETECHED'
msg['From'] = 'emailid@provider.com'
msg['To'] = 'emailid@provider.com'
text = MIMEText("During breach, following faces were detected")
msg.attach(text)
#attach base image
face_file_name = imageName
print ('attached-' +imageName)
img_data = open(face_file_name, 'rb').read()
image = MIMEImage(img_data, name=os.path.basename(face_file_name))
msg.attach(image)
print('@breachFacechopEmail detecting faces')
for f in faces:
    x, y, w, h = [ v for v in f ]
    cv2.rectangle(img, (x,y), (x+w,y+h), (255,255,255))

    sub_face = img[y:y+h, x:x+w]

    face_file_name = "face_" + str(y) + ".jpg"
    cv2.imwrite(face_file_name, sub_face)
```

```
        img_data = open(face_file_name, 'rb').read()
        image = MIMEImage(img_data, name=os.path.basename(face_file_name))
        msg.attach(image)
        print('@breachFacechopEmail detection was successful, images were saved')

    try:
    #smtp part
            print('@breachFacechopEmail compiling email')
            s = smtplib.SMTP('smtp.gmail.com', 587)
            s.ehlo()
            s.starttls()
            s.ehlo()
            s.login('emailid@provider.com', 'emailPassword')
            s.sendmail('emailid@provider.com', 'emailid@provider.com', msg.as_string())
            s.quit()
            print('@breachFacechopEmail email sent')
    except:
                print('error sending image email')
    print('@breachFacechopEmail EXITING')
    exit()
    #cv2.imshow(image, img)

    return
if __name__  == '__main__':
```

```
breachFacechopEmail("faces.jpg")


while(True):
    key = cv2.waitKey(20)
    if key in [27, ord('Q'), ord('q')]:
        break
```

3.7.3 Emergency state face crop and email algorithm

In case of emergency mode entry, the following code takes snapshot from the primary cameras, detects faces, crops out faces and emails it to the owner:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import cv2
import os
import smtplib
import time
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
def emergencyFacechopEmail(image):
    print('########')
    print('@facechop taking EMERGENCY snap')
    timestr = time.strftime("%Y%m%d-%H%M%S")
    print (timestr)
    imageName = timestr +'.jpg'
    os.system('raspistill -w 1920 -h 1080 -n -q 20 -o '+ timestr +'.jpg')
    facedata = "haarcascades/haarcascade_frontalface_default.xml"
    cascade = cv2.CascadeClassifier(facedata)
    img = cv2.imread(imageName)
    print('@emergencyFacechopEmail image loaded')
    minisize = (img.shape[1],img.shape[0])
    miniframe = cv2.resize(img, minisize)
```

```
faces = cascade.detectMultiScale(miniframe)


msg = MIMEMultipart()
msg['Subject'] = 'EMERGENCY ENTRY OCCURED AT HOUSE'
msg['From'] = 'emailid@provider.com'
msg['To'] = 'emailid@provider.com'
text = MIMEText("following faces were detected")
msg.attach(text)
#attach base image
face_file_name = imageName
print ('attached-' +imageName)
img_data = open(face_file_name, 'rb').read()
image = MIMEImage(img_data, name=os.path.basename(face_file_name))
msg.attach(image)
print('@emergencyFacechopEmail detecting faces')
for f in faces:
    x, y, w, h = [ v for v in f ]
    cv2.rectangle(img, (x,y), (x+w,y+h), (255,255,255))
    sub_face = img[y:y+h, x:x+w]
    face_file_name = "face_" + str(y) + ".jpg"
    cv2.imwrite(face_file_name, sub_face)
    img_data = open(face_file_name, 'rb').read()
    image = MIMEImage(img_data, name=os.path.basename(face_file_name))
    msg.attach(image)
    print('@emergencyFacechopEmail detection was successful, images were saved')
```

```
    try:
            print('@emergencyFacechopEmail compiling email')
            s = smtplib.SMTP('smtp.gmail.com', 587)
            s.ehlo()
            s.starttls()
            s.ehlo()
            s.login('emailid@provider.com', 'emailPassword')
            s.sendmail('emailid@provider.com', 'emailid@provider.com', msg.as_string())
            s.quit()
            print('@emergencyFacechopEmail email sent')
    except:
                print('error sending image email')
    print('@emergencyFacechopEmail EXITING')
    exit()


    return
if __name__  == '__main__':
    emergencyFacechopEmail("faces.jpg")
    while(True):
        key = cv2.waitKey(20)
        if key in [27, ord('Q'), ord('q')]:
            break
```

3.7.4 ATmega 328 algorithm

The ATmega328 microcontroller continously polls for radio frequenct identity module input as well as input from designated handshaking pins from the Raspberry Pi and produces output accordingly.

```
#define GREEN_LED 3
#define PIR_LED 4
#define RESET_INPUT 5
#define RFID_OUT 6
#define GREEN_INPUT 7
#define PIR_INPUT 9
#define BUILTIN_LED 13
#define WARNLEDPIN 12
char tag[] ="4C006E92C373";
char input[12];
int count = 0;
boolean flag = 0;
int pirState = 0;
int greenState = 0;
int resetState = 0;
void setup()
{
  Serial.begin(9600);
  pinMode(GREEN_LED,OUTPUT);
  pinMode(PIR_LED,OUTPUT);
  pinMode(RESET_INPUT,INPUT);
  pinMode(RFID_OUT,OUTPUT);
  pinMode(GREEN_INPUT,INPUT);
    pinMode(PIR_INPUT,INPUT);
  pinMode(BUILTIN_LED,OUTPUT);
  digitalWrite(GREEN_LED, HIGH);
  digitalWrite(PIR_LED, HIGH);
```

```
  delay(100);
  digitalWrite(GREEN_LED, LOW);
  digitalWrite(PIR_LED, LOW);
  digitalWrite(RFID_OUT, LOW);
  delay(100);
}
void loop()
{
  pir();
  rfid();
  green();
  reset();
}
void pir()
{
  pirState = digitalRead(PIR_INPUT);
  if (pirState == HIGH)
    {
      digitalWrite(PIR_LED, HIGH);
    } else {
     digitalWrite(PIR_LED, LOW);
    }
}
void green()
{
  greenState = digitalRead(GREEN_INPUT);
  if (greenState == HIGH)
    {
     digitalWrite(GREEN_LED, HIGH);
    } else {
     digitalWrite(GREEN_LED, LOW);
```

```
    }
}
void reset()
{
  resetState = digitalRead(RESET_INPUT);
  if (resetState == HIGH)
    {
      digitalWrite(GREEN_LED, LOW);
      digitalWrite(PIR_LED, LOW);
      digitalWrite(BUILTIN_LED, LOW);
      digitalWrite(RFID_OUT, LOW);
      delay(5000);
    }
}
void rfid()
{
if(Serial.available())
  {
    count = 0;
    while(Serial.available() && count < 12)
    {
      input[count] = Serial.read(); // Read 1 Byte of data and store it in the input[] variable
      count++;
      delay(5);
    }
    if(count == 12)
    {
      count =0;
      flag = 1;
       while(count<12 && flag !=0)
      {
```

```
      if(input[count]==tag[count])
      flag = 1;
      else
      flag= 0;
      count++; // increment i
     }
   }
   if(flag == 1)
   {
    digitalWrite(BUILTIN_LED, HIGH);
    Serial.println("Access Allowed!");
    digitalWrite(RFID_OUT,HIGH);
    delay(100);
   }
   else
   {
    digitalWrite(BUILTIN_LED, LOW);
    Serial.println("Access Denied"); // Incorrect Tag Message
   }
   for(count=0; count<12; count++)
  {
  /count = 0;
  }
 }
}
```

3.8 Software Details

3.8.1 IDLE

IDLE is Python's Integrated Development and Learning Environment (IDLE)

IDLE has the following features:

- coded in 100% pure Python, using the tkinter GUI toolkit
- cross-platform: works mostly the same on Windows, Unix, and Mac OS X
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- configuration, browsers, and other dialogs

IDLE has two main window types, the Shell window and the Editor window. It is possible to have multiple editor windows simultaneously. Output windows, such as used for Edit / Find in Files, are a subtype of edit window. They currently have the same top menu as Editor windows but a different default title and context menu.IDLE's menus dynamically change based on which window is currently selected. Each menu documented below indicates which window type it is associated with.

CHAPTER 4

RESULTS AND DISCUSSION

This chapter gives overview of the results achieved by the system, and suggests ways and means for further development.

4.1 Snapshots

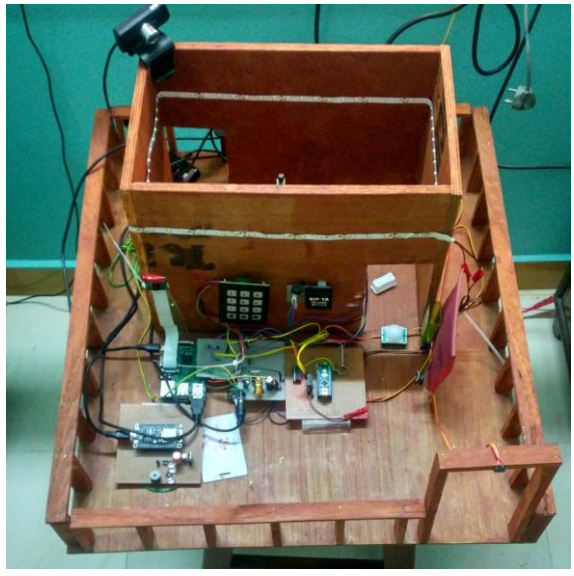


Fig. 4.1.1 Overall System View

Fig. 4.1.2 Activity Detected State

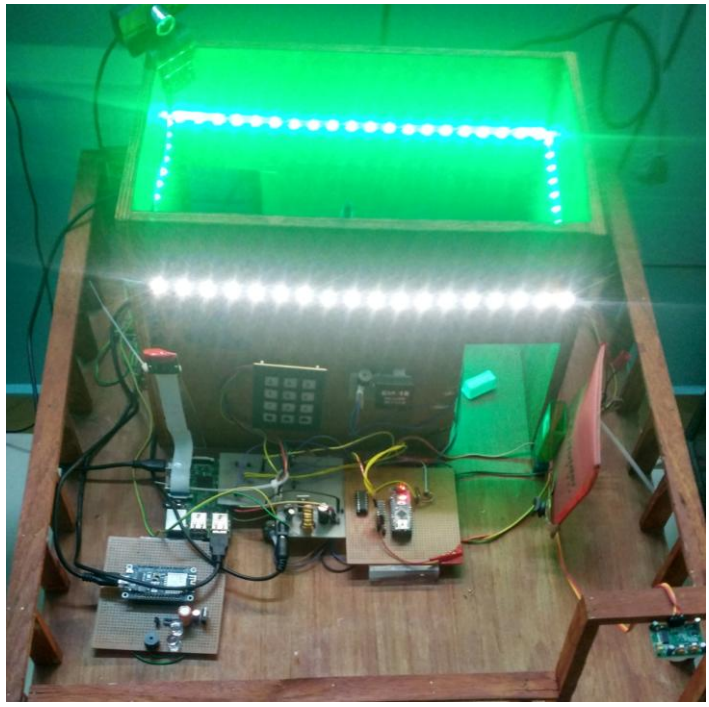


Fig. 4.1.3 Entry Permitted State

Fig. 4.1.3 Entry Permitted State

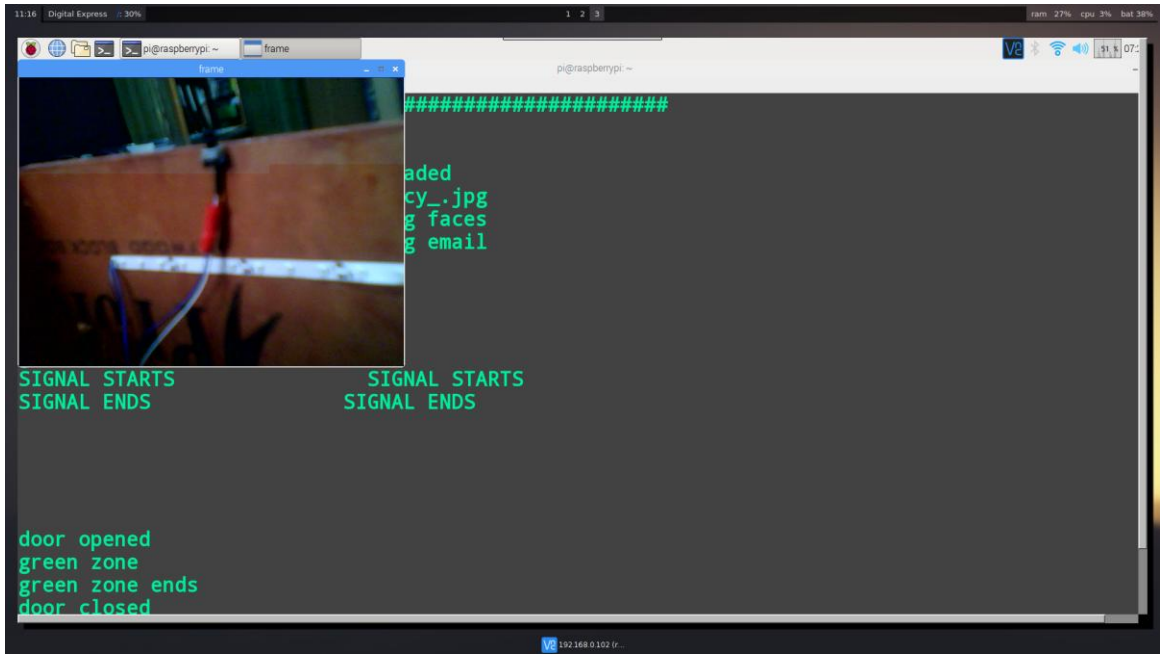


Fig. 4.1.4 Breach Detected State Console Output

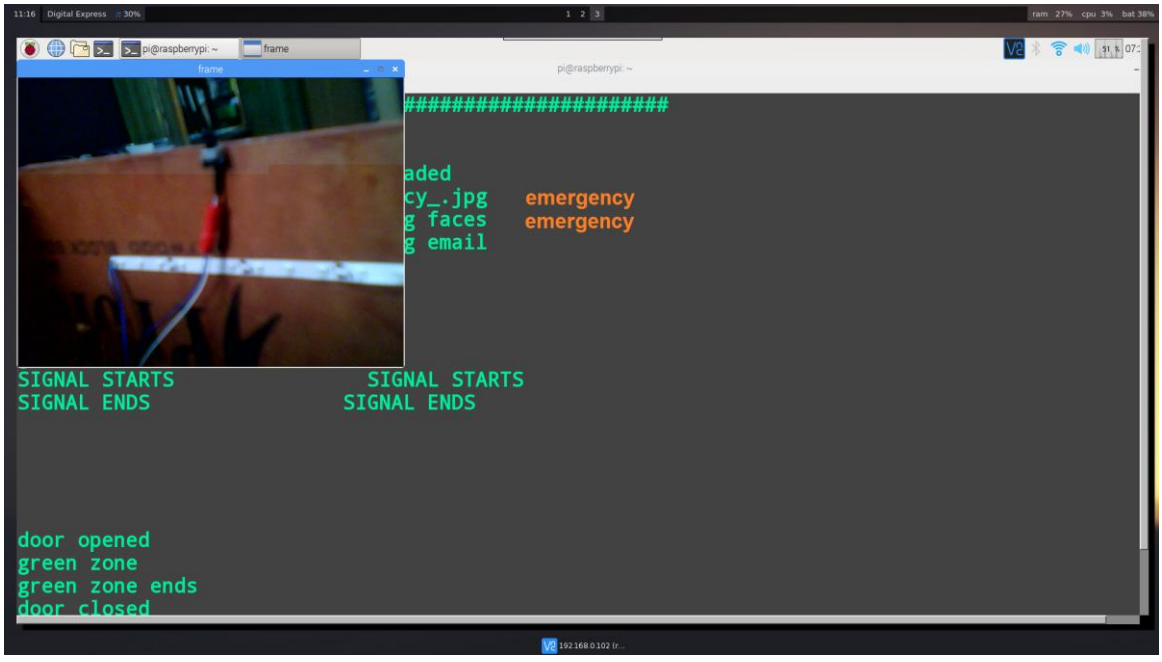


Fig. 4.1.5 Emergency entry Console Output

4.2 Discussion

The average face detection response time on the Raspberry Pi Model 3B using single core was found to average around 3.5 seconds at a resolution of 1920x1080. The video capture frame rates achieved were 40 frames per second with a single camera and 21 frames per second for 2 cameras recording simultaneously.

The performance can be further enhanced by using more cores available on the Raspberry Pi Model 3B through parallel processing. Results can also be enhanced by working completely on gray scale, in which case, frame rates of over 65 frames per second for single recording camera and over 35 fps for two cameras recording simultaneously were achieved.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

This chapter gives a summary of the project and a brief conclusion derived from the outcomes

5.1 Conclusion

The project designed and implemented the microcontroller based system for monitoring and regulating security of an area wherever required Raspberry Pi Model 3B and ATmega 328 microcontroller and various camera modules were utilized.

The system can be used to monitor and prevent intrusion and breaches into buildings if the incoming individual does not fulfill predefined criteria.

5.1 Future Scope

With the use f more processing power, the average detection rate can be increased to obtain smoother operation in practical scenarios. With further use of networking, face databases can be shared with systems in other houses for tracking activity of individuals.

REFERENCES

This section lists the various books and papers referred while designing the system.

[1] G Bradski, A Kaehler, "Learning OpenCV: Computer vision with the OpenCV library" O'Reilly Media, Inc.

[2] Paul Viola, Michael J. Jones, "Robust Real-Time Face Detection",International Journal of Computer Vision 57(2), 137–154, 2004

[3] Face Recognition Data, University of Essex, UK, Face 94, http://cswww.essex.ac.uk/mv/all faces/faces94.html.

[4] Prof. (Dr.) Khanna SamratVivekanand Omprakash, "Wireless home security system with mobile", International Journal of Advanced Engineering Technology E-ISSN 0976-3945

[5] Jignesh B Jadav, Dr.K.H.Wandra, Mr.Rohit Dabhi,"Innovative Automobile Security System Using Various Security Modules", International Journal on Recent and Innovation Trends in Computing and Communication Volume: 3 Issue: 2 ISSN: 2321-8169 583 – 586

[6] P.Vigneswari, V.Indhu, R.R.Narmatha, A.Sathinisha and J.M.Subashini "Automated Security System using Surveillance" International Journal of Current Engineering and Technology, E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

[7] K.S.Shilpashree, Lokesha.H, Hadimani Shivkumar, "Implementation of Image Processing on Raspberry Pi" International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 5, May 2015, ISSN (Online) 2278-1021 ISSN (Print) 2319-5940

[8] Vedang Ratan Vatsa, Gopal Singh, "Raspberry Pi based Implementation of Internet of Things using Mobile Messaging Application - 'Telegram' ", International Journal of Computer Applications (0975 – 8887) Volume 145 – No.14, July 2016

APPENDIX

This section presents the information, inclusion of which, in the prior chapters would have otherwise broken the flow of data being presented.

Frequently asked questions

- **How will the system alert owners if internet connection is not available?**
  Pins have been allocated for attaching alarm devices like buzzers, which can be easily used as a signal source for other alarm modules as desired.

- **How many passive infra-red sensors can be used for input?**
  By setting up the sensors in parallel configuration and using a simple current limiter circuit, as many sensors as needed can be attached to the general purpose input output pin of the microcontroller.

- **Is there any way to force stop the entire system if needed?**
  In the system design, a kill switch has been provided which can be used to reset the microcontroller and pi module to their initial states. However use of kill switch raises a brief alarm and sends email alerts for security purposes.
- **Is there any way to send video footage to the owner when located at come remote place?**
  The video footage generally exceeds the maximum permitted file size for email, however, it can be easily attached using the google dive application programmer interface and the uploaded file's link can be sent via email.