**CHAPTER I**

**INTRODUCTION**

**1. INTRODUCTION**

The **Internet of things (IoT)** is the network of physical devices, vehicles, home appliances and other items [embedded](https://en.wikipedia.org/wiki/Embedded_system) with [electronics](https://en.wikipedia.org/wiki/Electronics), [software](https://en.wikipedia.org/wiki/Software), [sensors](https://en.wikipedia.org/wiki/Sensor), [actuators](https://en.wikipedia.org/wiki/Actuator), and [network connectivity](https://en.wikipedia.org/wiki/Internet_access) which enables these objects to connect and exchange [data](https://en.wikipedia.org/wiki/Data). Each thing is uniquely identifiable through its embedded computing system but is able to inter-operate within the existing [Internet](https://en.wikipedia.org/wiki/Internet) infrastructure.

**Some changes that IoT will happen in the future:**

* Cheaper Home Security Cameras
* Less Expensive Surveillance
* Better Home Control

Monitoring facilities will be necessary and useful for our daily life, because it is very important for us to think about our security. This work developed system, which is organized with an integrated web server, highly secured cameras, Wi-Fi devices are connected to the internet. A specific server is located in the center of our system, which is called Integrated Server, which periodically obtains videos from some security cameras through the private network. Such videos are transmitted from the cameras to the server. The Integrated Server requires network cameras to transmit video at a sampling rate and compresses the video to MPEG then accumulates a series of them in the storage. The system captures information and transmits the live video streams via Wi-Fi wireless transceiver for IoT module to a Smart phone personal application by using the web.

Home automation or Domestic[s is building automation for](https://en.wikipedia.org/wiki/Building_automation) a home, called a smart home or smart house. A home automation system will control lighting, entertainment systems, and appliances. It may also include home security such as access control and alarm systems. When connected with the Internet, home devices are an important constituent of the Internet of Things.

A home automation system typically connects controlled devices to a central hub or "gateway". The user interface for control of the system uses either wall-mounted terminals, table or desktop computers, a mobile phone application, or a Web interface, that may also be accessible off-site through the Internet.

**1.1Home Automation**

Home automation is the process of controlling home appliances automatically using various control system techniques. The electrical and electronic appliances in the home such as fan, lights, outdoor lights, fire alarm, kitchen timer, etc., can be controlled using various control techniques.

Smart-n-Secure Home controls home appliances such as IOT based home automation over the cloud, home automation under Wi-Fi through android apps from any Smartphone, Arduino based home automation, and home automation by android application based remote control, home automation using digital control, RF based home automation system.

## 1.2 Benefits of Home Automation

1. **Security**

Tap your finger to turn on the lights when you get home so you worried about what’s hiding in the shadows, or in your pathways. Or automate to turn on when you aren’t home to look like you are to ward off potential robbers. Door locks are another automated home product that can increase your home security.

1. **Energy Efficiency**

Increase your home’s energy efficiency by remotely powering off systems and appliances when they aren’t in use. In addition to the standard home automation products that give you active control, some products actively monitor systems and arm the homeowner with knowledge, insight and guidance to achieve greater control and energy efficiency.

**iii. Savings**

Home automation literally pays off. When you are able to use home systems and appliances only when needed, the savings will be apparent in the first utility bill. No more wasting money on lights left on when you aren’t home, or spending money on gas to drive home because you forgot to lock the door. Monetary savings are apparent, but you’ll also be saving time. No wasted trips home, no running through the house turning everything off, no time spent worrying about what was or wasn’t turned off.

1. **Convenience**

Don’t you hate having to rely on neighbors to watch your house when you’re gone? With home automation, convenient control of your home is at your fingertips. You don’t have to trust someone else with your most valued possessions.

1. **Comfort**

Connected home products like the Sensi™ Wi-Fi Thermostats let you conveniently adjust your home temperature from the mobile app so your family is always comfortable.

1. **Peace of Mind**

One of the biggest hidden benefits that comes with home automation is peace of mind. No more worrying if you turned off the lights, locked the door or turned off the television. For people who have a lot on their plates, being able to easily check these items off the to-do list and stop the obsessive worrying, home automation is reassuring and definitely worth the investment.

**1.3 HOME SECURITY AUTOMATION**

Connected homes are the wave of the future, and automated home security systems are no exception. In terms of home monitoring or deterring crime, combining home automation and security is the way to go: you can program, monitor and control your security system remotely or by using an in-home dashboard. These systems include equipment like high-decibel alarms, certain types of security cameras, home automation locks (smart locks), motion sensors, control panels, and apps. If you want to invest in a home automation security system, you first need to decide how your system will be installed

**1.4 Advantages of Home Security Automation**

* **You could save money.** By choosing a  home security system, and installing your security system’s equipment and control panel yourself, you could drastically cut down the initial cost of your automated home security system.
* **You have more control.** With Home Security Automation, you can customize and adjust your system as needed. Add sensors or cameras as your needs change. You may also be able to integrate home security equipment with your existing smart home products.
* **You can do things on your own time.** No appointments needed, and no need to be home during normal business hours. With Home Security Automation, you can install your system on your schedule and at your pace, be it at sunrise, sunset or anytime in between—seven days a week.

**1.5 Equipments Needed In Home Security Automation**

Home automation security requires a few additions to standard items—such as the control panel, alarm and yard signs—but these will transform how you manage your security system and might even introduce energy savings in your home. Most devices offer the capability to remotely manage your home security system (e.g., a mobile application and a light-control module). Others include the following:

* Home Security Camera (Raspberry pi camera)
* Video Surveillance Door Bell (Buzzer)
* Security Sensors (PIR Motion Detector)

## 1.6 Home Security Cameras

A security camera is one of the best ways to both identify and catch burglars, especially because it’s more likely that a burglary will take place when no one is home. There are several types of security cameras to choose from, and they all have various features and benefits.

**1.7 Benefits of Home Security Cameras**

There are many benefits of home security cameras. And if you have a security camera in your home automation security system, the advantages increase.

* **They deter crime.** If criminals spot a camera, they just might change their minds about breaking in.
* **You can remotely access from anywhere.** Wi-Fi-enabled security cameras can alert you via your smart device no matter where you are.
* **They give you more connectivity. Security cameras can interface with your hub and help inform you when to use your other automated devices like smart thermostats or light switches.**
* **You can have peace of mind.** Even if it’s only to check up on pets while you’re away, a security camera integrated into your smart home will keep anxiety at bay.

**1.8 Features of Home Security Camera (Rpi Cam)**

* **Wi-Fi capability.** A Wi-Fi-connected camera sends notifications to your smart device and allows you to keep an eye on what’s happening at home without having to invest in a fully automated home security system (a great choice if you live in a small apartment).
* **Outdoor or indoor.** Indoor and outdoor security cameras differ based on the types of external factors they can withstand. They come in similar styles and have similar features, but outdoor cameras must be able to withstand all types of weather and light conditions.
* **Day/night use.** Most types of security cameras can observe and record what’s happening during the day, but the distance from which your camera can record images in low to no light is directly related to effectiveness, because criminal activity often happens at night. Some cameras boast night vision up to 30 feet and others to 100, so be sure to check the specifications.
* **Video storage.** Your footage likely won’t be saved on the camera itself. Most types of security cameras store information on the cloud (software used through network connection), and some models have SD card slots so that you can physically pull the video when you want to review it.
* **Color or black-and-white video.** Color video gives you a better chance to identify a perpetrator based on something like clothing; however, color security cameras don’t perform as well in low-lighting conditions. Black-and-white security cameras can perform in even the dimmest of conditions and are often cheaper.
* **Recording modes.** Continuous (24-hour) recording catches everything, even if it’s not helpful. Motion-detecting recording doesn’t capture unnecessary events, which saves bandwidth and storage. Scheduled recording also saves bandwidth as you can program your camera to specifically record only during certain times.
* **Video resolution.** The higher the resolution, the more visual information a security camera can capture, resulting in better clarity of the footage, especially when zooming in.
* **Wires or wireless.** Some types of security cameras must still be connected to a power source and require video cables to transmit footage to devices such as a computer or television. But a wireless security camera transmits the video (and audio, if included) over the Internet to a receiver that connects to your device.
* **Motion detecting.** Most Wi-Fi-connected cameras have built-in motion- and sound-detecting sensors and will alert you when those sensors are triggered.
* **Field of vision.** Field of vision—how wide an angle the camera can record—is important because it affects how many cameras you need and where to place them. Some newer types of security cameras have a 360-degree field of vision.

**1.9 Home Security Motion Sensor (PIR Motion Sensor)**

A motion sensor, or motion detector, is a critical part of a home security system. It’s the primary piece of equipment that detects an intruder.

**1.9.1 Working of Motion Sensor**

A motion sensor uses various types of technology to detect movement in an area. If movement trips a sensor, your security system’s control panel receives a signal, which in turn alerts you to a potential threat in your home. In addition, motion-sensing lights are a great addition to home security, as they detect movement and then signal a light to turn on, deterring would-be criminals. Remember that security sensors are different from light-switch sensors, but you can learn more about the home energy savings of indoor light switch motion sensors here.

### 1.9.2 Benefits of Security Sensors

variety of benefits that make them excellent crime deterrents and great additions to your automated From motion detectors to motion-sensing lights, these security measures have a home security system.

* **Security is heightened.** The core purpose of a security sensor—and its biggest benefit—is securing your home against burglars. If your system is breached without alerting you to a door or window being opened, your security sensors will detect an intruder and sound the alarm. This gives you time to call for assistance.
* **False alarms are a thing of the past.** Security sensors are so advanced that you don’t need to worry about pets setting them off.  
  Security sensors easily disabled. You can easily disable a security sensor if you feel the urge to grab a midnight snack.
* **They catch almost everything.** Security sensors can cover nearly the entire area of a room.
* **Having sensors reduce overall crime.** Motion-sensing lights in particular deter crime: they leave would-be burglars with no place to hide, and they give you more time to investigate or call the police.

**CHAPTER II**

**RASPBERRY PI 3**

**AND**

**ARCHITECTURE**

2.1**Raspberry pi 3**

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries.

Raspberry Pi feature a Broadcom system on a chip (SoC), which includes an hARM compatible central processing unit (CPU) and an on-chip graphics processing unit (GPU, a videocore IV). CPU speed ranges from 700 MHz to 1.2 GHz for the Pi 3 and on board memory range from 256 MB to 1 GB RAM. Secure Digital (SD) cards are used to store the operating system and program memory in either the SDHC or MicroSDHC sizes. Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phono jack for audio. Lower level output is provided by a number of GPIO pins which support common protocols like I²C. The B-models have an 8P8C Ethernet port and the Pi 3 and Pi Zero W have on board Wi-Fi 802.11n and Bluetooth.

###### 

Figure

**2.2SUMMARY**

|  |  |
| --- | --- |
| Raspberry pi 3 | Model b |
| OperatingVoltage | 5V |
| Input Voltage | 5V |
| Input Voltage (limits) | 5.5-5.75V |
| DigitalI/O Pins | 40 (of which 17 provideinput output) |
| AnalogInput Pins | 4 |
| DC Current perI/O Pin | 50 Ma |
| DC Current for3.3V Pin | 50 Ma |
| Operating System | Raspbian |
| Memory | 1GB LPBR2 RAM at 900MHz |
| Storage | MicroSDHC slot (using 16 GB) |
| Clock Speed | 1.2GHz |

* **Power supply**

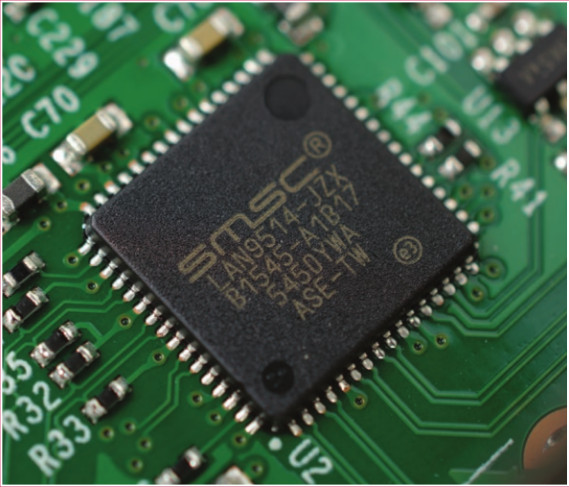
A Raspberry pi 3 is powered by the +5.1V micro USB supply.

* **DSI CONNECTOR**

The Raspberry Pi display is an LCD display which connects to the Raspberry Pi through the DSI connector. In some situations, it allows for the use of both the HDMI and LCD displays at the same time (this requires software support).

## 2.3 USB chip

The Raspberry Pi 3 shares the same SMSC LAN9514 chip as its predecessor, the Raspberry Pi 2, adding 10/100 Ethernet connectivity and four USB channels to the board. As before, the SMSC chip connects to the SoC via a single USB channel, acting as a USB-to-Ethernet adaptor and USB hub



Figure

## 2.4 Antenna

There’s no need to connect an external antenna to the Raspberry Pi 3. Its radios are connected to this chip antenna soldered directly to the board, in order to keep the size of the device to a minimum. Despite its diminutive stature, this antenna should be more than capable of picking up wireless LAN and Bluetooth signals – even through walls.

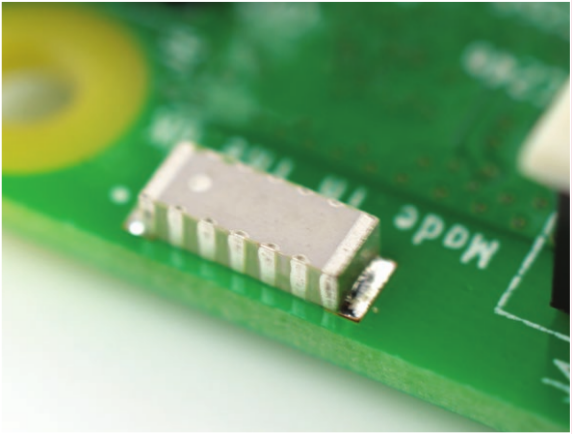


Figure 3

## 2.5 SoC

Built specifically for the new Pi 3, the Broadcom BCM2837 system-on-chip (SoC) includes four high-performance ARM Cortex-A53 processing cores running at 1.2GHz with 32kB Level 1 and 512kB Level 2 cache memory, a VideoCore IV graphics processor, and is linked to a 1GB LPDDR2 memory module on the rear of the board.



Figure 4

## 2.6 GPIO

The Raspberry Pi 3 features the same 40-pin general-purpose input-output (GPIO) header as all the Pis going back to the Model B+ and Model A+. Any existing GPIO hardware will work without modification; the only change is a switch to which UART is exposed on the GPIO’s pins, but that’s handled internally by the operating system.

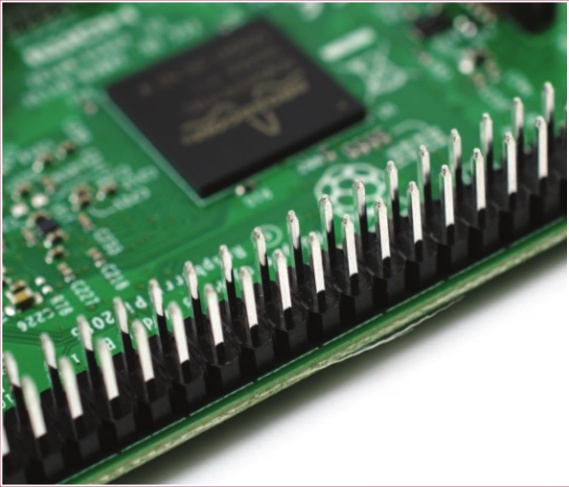


Figure 5

**2.7 Wireless radio**

So small, its markings can only be properly seen through a microscope or magnifying glass, the Broadcom BCM43438 chip provides 2.4GHz 802.11n wireless LAN, Bluetooth Low Energy, and Bluetooth 4.1 Classic radio support. Cleverly built directly onto the board to keep costs down, rather than the more common fully qualified module approach, its only unused feature is a disconnected FM radio receiver.

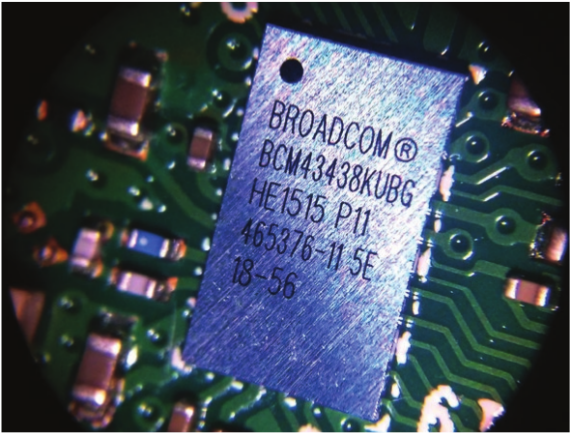


Figure 6

**2.8 GPIO PINS**

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Pi Zero and Pi Zero W). Prior to the Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header.

Any of the GPIO pins can be designated (in software) as an input or output pin and used for a wide range of purposes.

* **voltage**

Two 5V pins and two 3V3 pins are present on the board, as well as a number of ground pins (0V), which are unconfigurable. The remaining pins are all general purpose 3V3 pins, meaning outputs are set to 3V3 and inputs are 3V3-tolerant.

* **Outputs**

A GPIO pin designated as an output pin can be set to high (3V3) or low (0V).

Inputs

A GPIO pin designated as an input pin can be read as high (3V3) or low (0V). This is made easier with the use of internal pull-up or pull-down resistors. Pins GPIO2 and GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software.

* **More**

As well as simple input and output devices, the GPIO pins can be used with a variety of alternative functions, some are available on all pins, others on specific pins.

PWM (pulse-width modulation)

Software PWM available on all pins

Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19

* **SPI**

SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)

SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)

I2C

Data: (GPIO2); Clock (GPIO3)

EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)

* **Serial**

TX (GPIO14); RX (GPIO15)

GPIO pinout

It's important to be aware of which pin is which. Some people use pin labels (like the [RasPiOPortsplus](http://rasp.io/portsplus/) PCB, or the printable [Raspberry Leaf](https://github.com/splitbrain/rpibplusleaf)).

**Table 1: PIN CONFIGURATION**

|  |  |
| --- | --- |
| Pin number | Pin name |
|  | 3.3V DC power |
|  | 5V DC power |
|  | GPIO 02(SDA1,I2C) |
|  | 5V DC power |
|  | GPIO03(SCL1,I2C) |
|  | GROUND |
|  | GPIO04(GPIO\_GCLK) |
|  | GPIO14(TXD0) |
|  | GROUND |
|  | GPIO15(RXD0) |
|  | GPIO17(GPIO\_GEN0) |
|  | GPIO18(GPIO\_GEN1) |
|  | GPIO27(GPIO\_GEN2) |
|  | GROUND |
|  | GPIO22(GPIO\_GEN3) |
|  | GPIO23(GPIO\_GEN4) |
|  | 3.3V DC POWER |
|  | GPIO24(GPIO\_GEN5) |
|  | GPIO10(SPI\_MOSI) |
|  | GROUND |
|  | GPIO09(SPI\_MISO) |
|  | GPIO25(GPIO\_GEN6) |
|  | GPIO11(SPI\_CLK) |
|  | GPIO08(SPI\_CE0\_N) |
|  | GROUND |
|  | GPIO07(SPI\_CE1\_N) |
|  | ID\_SD(12C ID EEPROM) |
|  | ID\_SC(12C ID EEPROM) |
|  | GPIO05 |
|  | GROUND |
|  | GPIO06 |
|  | GPIO12 |
|  | GPIO13 |
|  | GROUND |
|  | GPIO19 |
|  | GPIO16 |
|  | GPIO26 |
|  | GPIO20 |
|  | GROUND |
|  | GPIO21 |

* **Key benefits of Raspberry pi**
* Low cost
* Cross platform
* Simple clear programming environment
* Open source

**CHAPTER III**

**Devices**

### 3.1Passive infrared motion sensors

A passive infrared sensor (PIR sensors) electronic sensors that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors. Infrared radiation enters through the front of the sensor, known as the 'sensor face'. At the core of a PIR sensor is a solid state sensor or set of sensors, made from pyroelectric materials—materials which generate energy when exposed to heat. Typically, the sensors are approximately 1/4 inch square (40 mm2), and take the form of a thin film. Materials commonly used in PIR sensors include gallium nitride (GaN), caesium nitrate (CsNO3), polyvinyl fluorides, derivatives of phenylpyridine, and cobalt phthalocyanine. The sensor is often manufactured as part of an integrated circuit. An individual PIR sensor detects changes in the amount of infrared radiation impinging upon it, which varies depending on the temperature and surface characteristics of the objects in front of the sensor. When an object, such as a human, passes in front of the background, such as a wall, the temperature at that point in the sensor's field of view will rise from room temperature to body temperature, and then back again. The sensor converts the resulting change in the incoming infrared radiation into a change in the output voltage, and this triggers the detection. Objects of similar temperature but different surface characteristics may also have a different infrared emission pattern, and thus moving them with respect to the background may trigger the detector as well.



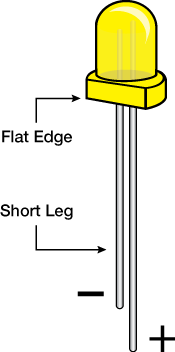
Figure 7: PIR Motion Sensor

Table 1: PIR motion sensors specification

|  |  |
| --- | --- |
| **Specification** |  |
| Operating Voltage | 12VDC(Power Up by Solar Panel) |
| Detection Speed | 0.3m/5~3.2m/s |
| Transmitting Distance | Minimum 100m(in the open area) |
| Dimensions | 160\*65\*50.5mm(H\*W\*D) |
| Relative Humidity | 95%(25 `C) |

### 3.2 Light Emitting diode

An LED is a small light (it stands for "light emitting diode") that works with relatively little power. LEDs have polarity, which means they will only light up if you orient the legs properly. The long leg is typically positive, and should connect to a digital pin on the Arduino board. The short leg goes to GND; the bulb of the LED will also typically have a flat edge on this side. In order to protect the LED, you will also need use a resistor "in series" with the LED. If the LED doesn't light up, trying reversing the legs (you won't hurt the LED if you plug it in backwards for a short period of time). LEDs are small, powerful lights that are used in many different applications. To start, we will work on blinking an LED, the Hello World of microcontrollers. It is as simple as turning a light on and off. Establishing this important baseline will give you a solid foundation as we work towards experiments that are more complex. To find out the polarity of an LED, look at it closely. The shorter of the two legs, towards the flat edge of the bulb indicates the negative terminal.



* **Figure 8: LED**

### 3.3 Buzzer

Piezo buzzer is an electronic device commonly used to produce sound. Piezo buzzer is based on the inverse principle of piezo electricity discovered in 1880 by Jacques and Pierre Curie. It is the phenomena of generating electricity when mechanical pressure is applied to certain materials and the vice versa is also true. Such materials are called piezo electric materials. Piezo electric materials are either naturally available or manmade. Piezoceramic is class of manmade material, which poses piezo electric effect and is widely used to make disc, the heart of piezo buzzer. When subjected to an alternating electric eld they stretch or compress, in accordance with the frequency of the signal thereby producing sound. Piezo buzzers are used for making beeps alarms and tones. They can be used in alarm systems, for keypad feedback, or some games. Light weight, simple construction and low price make it usable in various applications like car/truck reversing indicator, computers, call bells etc. Also they are fun to play around with. While technological advancements have caused buzzers to be impractical and undesirable, there are still instances in which buzzers and similar circuits may be used. Typical uses of buzzers and beepers include [alarm devices](https://en.wikipedia.org/wiki/Alarm_devices), [timers](https://en.wikipedia.org/wiki/Timer), and confirmation of user input such as a mouse click or keystroke.



**Figure 9: Buzzer**

Table 3: Buzzer specification

|  |  |
| --- | --- |
| **Specification** |  |
| Voltage | 9DCV-15DCV |
| Rated Voltage | 12DCV |
| Power cost | Max 0.5W |
| Respond distance | 0.3m-2.0m |
| Alarm distance | 0m-2.0m |
| Display mode | LCD display |
| Sensors diameter | 22mm |
| Install mode | Drill |
| Sensors angle | X,Y,60℃ |
| Work temperature | -30℃…..+80℃ |
| Alarm type | Buzzer (Bi Bi….) |
| Alarm volume | 65dB |
| Alarm frequency | 0.5-3Hz(commonly) |

**3.4 Raspberry pi Camera**

The Raspberry Pi Camera Module v2 replaced the original Camera Module in April 2016. The v2 Camera Module has a Sony IMX219 8-megapixel sensor (compared to the 5-megapixel OmniVision OV5647 sensor of the original camera).

The Camera Module can be used to take high-definition video, as well as stills photographs. It’s easy to use for beginners, but has plenty to offer advanced users if you’re looking to expand your knowledge. There are lots of examples online of people using it for time-lapse, slow-motion, and other video cleverness. You can also use the libraries we bundle with the camera to create effects.

You can read all the gory details about IMX219 and the Exmor R back-illuminated sensor architecture on Sony’s website, but suffice to say this is more than just a resolution upgrade: it’s a leap forward in image quality, colour fidelity, and low-light performance. It supports 1080p30, 720p60 and VGA90 video modes, as well as still capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.

The camera works with all models of Raspberry Pi 1, 2, and 3. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Picamera Python library. See the Getting Started with Picamera resource to learn how to use it.

The camera module is very popular in home security applications, and in wildlife camera traps.

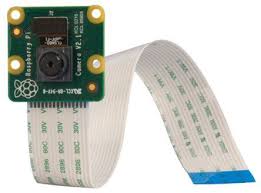


Figure 10:Raspberry pi camera

The camera may come with a small piece of translucent blue plastic film covering the lens. This is only present to protect the lens while it is being mailed to you, and needs to be removed by gently peeling it off.

* **Enabling the camera**

Open the **raspi-config**tool from the **Terminal :**

**sudo raspi-config:**

Select **Enable camera** and hit**Enter**, then go to**Finish** and you'll be prompted to reboot.

* **Using the camera**

Libraries for using the camera are available in:

* Shell (Linux command line)
* Python

|  |  |  |
| --- | --- | --- |
| Table 4 |  | SOFTWARE FEATURES OF RASPBERRY PI CAMERA |
| Picture formats |  | JPEG (accelerated), JPEG + RAW, GIF, BMP, PNG, YUV420, RGB888 |
| Video formats |  | raw h.264 (accelerated) |
| Effects |  | negative, solarise, posterize, whiteboard, blackboard, sketch, denoise, emboss, oilpaint, hatch, gpen, pastel, watercolour, film, blur, saturation |
| Exposure modes |  | auto, night, nightpreview, backlight, spotlight, sports, snow, beach, verylong, fixedfps, antishake, fireworks |
| Metering modes |  | average, spot, backlit, matrix |
| Automatic white  balance modes |  | off, auto, sun, cloud, shade, tungsten, fluorescent, incandescent, flash, horizon |
| Triggers |  | Keypress, UNIX signal, timeout |
| Extra modes |  | demo, burst/timelapse, circular buffer, video with motion vectors, segmented video, live preview on 3D models |

**CHAPTER IV**

**Technical Specifications**

## 4.1 Hardware Environment

Table 5: Hardware requirements

|  |  |  |
| --- | --- | --- |
| **Server side** |  | **Client side** |
|  |  |  |
| Raspberry pi 3, pi camera, | Pc, | Smartphone | . |
| PIR Motion Sensor, Buzzer,  LED. |  |  |  |
|  |  |  |
|  |  |
|  |
|  | |  |

## Software Environment

Table 6: Software Requirements

|  |  |
| --- | --- |
| Server side | Client side |
|  |  |
| Python . | Browser e.g. Google Chrome, Mozilla |
|  |
|  |
| Python django for website | Firefox etc. |
|  |  |

**4.2.1 Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

**4.2.2Python django**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django is an extremely popular and fully featured server-side web framework, written in Python. The module shows you why Django is one of the most popular web server frameworks, how to set up a development environment, and how to start using it to create your own web applications.

* **Prerequisites**

Before starting this module you don't need to have any knowledge of Django. Ideally, you would need to understand what server-side web programming and web frameworks are by reading the topics in our Server-side website programming first steps module.

A general knowledge of programming concepts and Python is recommended, but is not essential to understanding the core concepts.

Note: Python is one of the easiest programming languages for novices to read and understand. That said, if you want to understand this module better, there are numerous free books and tutorials available on the Internet to help you out. (new programmers might want to check out the Python for Non Programmers page on the python.org wiki).

**4.3 Django introduction**

In this first Django article we answer the question "What is Django?" and give you an overview of what makes this web framework special. We'll outline the main features, including some advanced functionality that we won't have time to cover in detail in this module. We'll also show you some of the main building blocks of a Django application, to give you an idea of what it can do before you set it up and start playing.

* **Setting up a Django development environment**

Now that you know what Django is for, we'll show you how to setup and test a Django development environment on Windows, Linux (Ubuntu), and Mac OS X — whatever common operating system you are using, this article should give you what you need to be able to start developing Django apps.

* **The Local Library website**

The first article in our practical tutorial series explains what you'll learn, and provides an overview of the "local library" — an example website we'll be working through and evolving in subsequent articles.

* **Creating a skeleton website**

This article shows how you can create a "skeleton" website project as a basis, which you can then go on to populate with site-specific settings, urls, models, views, and templates.

* **Using models**

This article shows how to define models for the LocalLibrary website — models represent the data structures we want to store our app's data in, and also allow Django to store data in a database for us (and modify it later on). It explains what a model is, how it is declared, and some of the main field types. It also briefly shows a few of the main ways you can access model data

* **Django admin site**

Now that we've created models for the LocalLibrary website, we'll use the Django Admin site to add some "real" book data. First we'll show you how to register the models with the admin site, then we'll show you how to login and create some data. At the end we show some ways in which you can further improve the presentation of the admin site.

* **Creating our home page**

We're now ready to add the code to display our first full page — a home page for the LocalLibrary that shows how many records we have of each model type and provides sidebar navigation links to our other pages. Along the way we'll gain practical experience in writing basic URL maps and views, getting records from the database, and using templates.

* **Generic list and detail views**

This tutorial extends our LocalLibrary website, adding list and detail pages for books and authors. Here we'll learn about generic class-based views, and show how they can reduce the amount of code you have to write for common use cases. We'll also go into URL handling in greater detail, showing how to perform basic pattern matching.

* **Sessions framework**

This tutorial extends our LocalLibrary website, adding a session-based visit-counter to the home page. This is a relatively simple example, but it does show how you can use the session framework to provide persistent behaviour for anonymous users in your own sites.

* **User authentication and permissions**

In this tutorial we'll show you how to allow users to login to your site with their own accounts, and how to control what they can do and see based on whether or not they are logged in and their permissions. As part of this demonstration we'll extend the LocalLibrary website, adding login and logout pages, and user- and staff-specific pages for viewing books that have been borrowed.

* **Working with forms**

In this tutorial we'll show you how to work with HTML Forms in Django, and in particular the easiest way to write forms to create, update, and delete model instances. As part of this demonstration we'll extend the LocalLibrary website so that librarians can renew books, create, update, and delete authors using our own forms (rather than using the admin application).

* **Testing a Django web application**

As websites grow they become harder to test manually — not only is there more to test, but, as the interactions between components become more complex, a small change in one area can require many additional tests to verify its impact on other areas. One way to mitigate these problems is to write automated tests, which can easily and reliably be run every time you make a change. This tutorial shows how to automate unit testing of your website using Django's test framework.

* **Deploying Django to production**

Now you've created (and tested) an awesome LocalLibrary website, you're going to want to install it on a public web server so that it can be accessed by library staff and members over the Internet. This article provides an overview of how you might go about finding a host to deploy your website, and what you need to do in order to get your site ready for production.

* **Django web application security**

Protecting user data is an essential part of any website design. We previously explained some of the more common security threats in the article Web security — this article provides a practical demonstration of how Django's in-built protections handle such threats.

**4.4 Features of django**

* **Ridiculously fast.**

Django was designed to help developers take applications from concept to completion as quickly as possible.

* **Reassuringly secure.**

Django takes security seriously and helps developers avoid many common security mistakes.

* **Exceedingly scalable.**

Some of the busiest sites on the Web leverage Django’s ability to quickly and flexibly scale.

**CHAPTER V**

**IMPLEMENTATION**

# 5.1 Implementation

Home Security Automation is Connected homes are the wave of the future, and automated home security systems are no exception. In terms of home monitoring or deterring crime, combining home automation and security is the way to go: you can program, monitor and control your security system remotely or by using an in-home dashboard. These systems include equipment like high-decibel alarms, certain types of security cameras, home automation locks (smart locks), motion sensors, control panels, and apps. If you want to invest in a home automation security system, you first need to decide how your system will be installed.

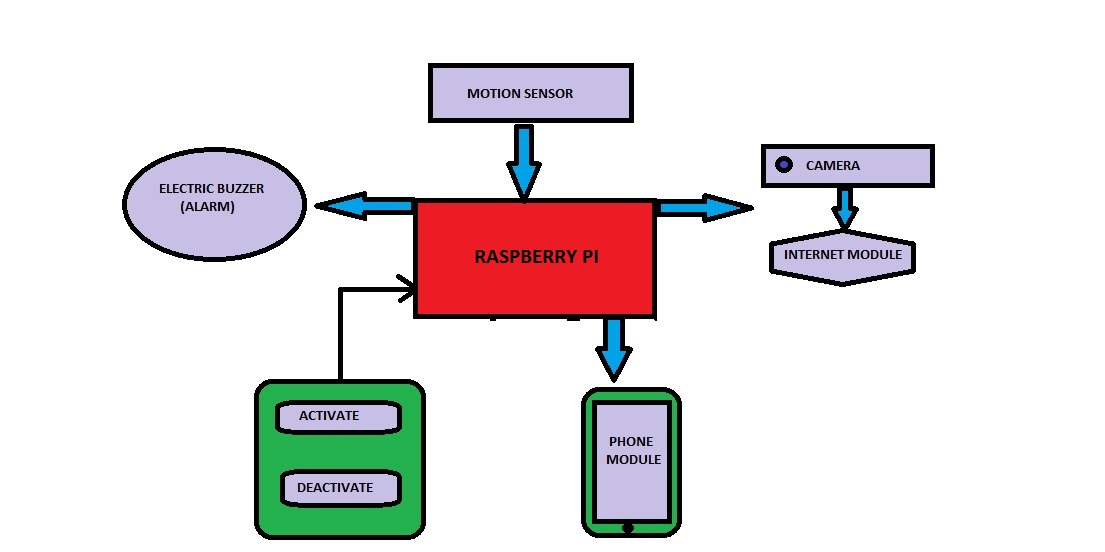
The following are the hardware devices which are needed to implement the home security automation:

* Raspberry pi
* PIR Motion Sensor.
* Raspberry pi camera
* Buzzer.
* Light Emitting Diode (LED).
* Push button.

When the intruder wants to enter the house, the PIR motion sensor detects the motion and takes the motion as input to the raspberry pi which in turn activates the camera , the camera then clicks the pictures or records the motion of intruder and sends it as the notifications to house owner either in his email address or he can watch it on webpage provided to him and also the buzzer installed in the house activates .

**CHAPTER VI**

**Block Diagram**

****

**Block diagram Home Security Automation**

**CHAPTER VII**

**CODE**

**7.1 Code for Push Button**

import RPi.GPIO as gpio

from time import sleep

gpio.setwarnings(False)

gpio.setmode(gpio.BOARD)

btn = 16

gpio.setup(btn,gpio.IN,pull\_up\_down=gpio.PUD\_UP)

buz = 3

gpio.setup(buz,gpio.OUT)

def buzz():

gpio.output(buz,True)

try:

while(1):

if gpio.input(btn)==0:

print("Button Pressed..!")

buzz()

sleep(.5)

else:

gpio.output(buz,False)

sleep(.5)

finally:

gpio.output(buz,0)

gpio.cleanup()

**7.2 Code for Raspberry pi Cam (video , picture capturing and email sending)**

import sys

import io

import logging

import socketserver

from threading import Condition

from http import server

import RPi.GPIO as gpio

import picamera

import time

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.base import MIMEBase

from email import encoders

from email.mime.image import MIMEImage

fromaddr = "msameer234@gmail.com"

toaddr = "msameer234@gmail.com"

mail = MIMEMultipart()

mail["From"] = fromaddr

mail["To"] = toaddr

mail["Subject"] = "Attactment"

body = "Please find the attachment"

gpio.setwarnings(False)

gpio.setmode(gpio.BOARD)

led = 18

gpio.setup(led,gpio.OUT)

btn = 16

gpio.setup(btn,gpio.IN,pull\_up\_down=gpio.PUD\_UP)

pir = 7

gpio.setup(pir,gpio.IN)

buz = 3

gpio.setup(buz,gpio.OUT)

HIGH = 1

LOW = 0

data = ""

gpio.output(led,0)

gpio.output(buz,0)

def sendMail(data):

mail.attach(MIMEText(body,'plain'))

print(data)

dat = '%s.jpg'%data

print(dat)

attachment = open(cam/dat,'rb')

image = MIMEImage(attachment.read())

attachment.close()

mail.attach(image)

server = smtplib.SMTP('smtp.gmail.com',587)

server.starttls()

server.login(fromaddr,"Probook000")

text = mail.as\_string()

server.sendmail(fromaddr,toaddr,text)

print("Senddddd..")

server.quit()

def capture\_image():

camera = picamera.PiCamera()

data = time.strftime("cam/%d\_%b\_%Y|%H:%M:%S")

camera.start\_preview()

time.sleep(1)

print(data)

camera.capture('%s.jpg'%data)

print('Image Captured')

camera.stop\_preview()

print('camera preview stopped')

time.sleep(0.5)

print("Sending image to Email")

## sendMail(data)

print('Closing camera')

time.sleep(1)

camera.close()

print("Image Sent..!!!")

print("Press CTRL+C to stop monitoring\n\n")

print("------------------------------------------")

def record\_video():

camera = picamera.PiCamera()

data = time.strftime("videos/%d\_%b\_%Y|%H:%M:%S")

camera.start\_preview()

time.sleep(1)

print(data)

duration = int(input("Video Duration(In Seconds): "))

camera.start\_recording('%s.h264'%data)

print('[-] Recording Video')

time.sleep(duration)

camera.stop\_recording()

print('[-] Video recorded successfully')

camera.stop\_preview()

print('camera preview stopped')

time.sleep(0.5)

## print("Sending image to Email")

## sendMail(data)

print('Closing camera')

time.sleep(1)

camera.close()

## print("Image Sent..!!!")

## print("Press CTRL+C to stop monitoring\n\n")

print("------------------------------------------")

def startMonitoring():

time.sleep(2)

print("------------------------------------------")

print("Monitoring Mode")

monitoring = True

print("Press CTRL+C to stop monitoring")

while monitoring:

if gpio.input(pir) == 1:

print('Motion Detected')

gpio.output(led,1)

gpio.output(buz,1)

capture\_image()

while(gpio.input(pir)):

time.sleep(1)

elif gpio.input(btn) == 0:

print('Button Pressed')

gpio.output(led,1)

gpio.output(buz,1)

capture\_image()

else:

gpio.output(led,0)

gpio.output(buz,0)

time.sleep(0.1)

PAGE="""\

<html>

<head>

<title>Raspberry Pi - Surveillance Camera</title>

</head>

<body>

<center><h1>Raspberry Pi - Surveillance Camera</h1></center>

<center><img src="stream.mjpg" width="640" height="480"></center>

</body>

</html>

"""

class StreamingOutput(object):

def \_\_init\_\_(self):

self.frame = None

self.buffer = io.BytesIO()

self.condition = Condition()

def write(self, buf):

if buf.startswith(b'\xff\xd8'):

# New frame, copy the existing buffer's content and notify all

# clients it's available

self.buffer.truncate()

with self.condition:

self.frame = self.buffer.getvalue()

self.condition.notify\_all()

self.buffer.seek(0)

return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):

def do\_GET(self):

if self.path == '/':

self.send\_response(301)

self.send\_header('Location', '/index.html')

self.end\_headers()

elif self.path == '/index.html':

content = PAGE.encode('utf-8')

self.send\_response(200)

self.send\_header('Content-Type', 'text/html')

self.send\_header('Content-Length', len(content))

self.end\_headers()

self.wfile.write(content)

elif self.path == '/stream.mjpg':

self.send\_response(200)

self.send\_header('Age', 0)

self.send\_header('Cache-Control', 'no-cache, private')

self.send\_header('Pragma', 'no-cache')

self.send\_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')

self.end\_headers()

try:

while True:

with output.condition:

output.condition.wait()

frame = output.frame

self.wfile.write(b'--FRAME\r\n')

self.send\_header('Content-Type', 'image/jpeg')

self.send\_header('Content-Length', len(frame))

self.end\_headers()

self.wfile.write(frame)

self.wfile.write(b'\r\n')

except Exception as e:

logging.warning(

'Removed streaming client %s: %s',

self.client\_address, str(e))

else:

self.send\_error(404)

self.end\_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):

allow\_reuse\_address = True

daemon\_threads = True

output = StreamingOutput()

def main():

print("\nWelcome to HOME SECURITY AUTOMATION\n")

options = """ OPTIONS:

1. START MONITORING

2. CAPTURE IMAGE

3. RECORD VIDEO

4. STREAM LIVE

PRESS CTRL+C TO EXIT"""

try:

print(options)

choice = int(input("Select your option(1-5): "))

if choice == 1:

try:

startMonitoring()

finally:

print("Monitoring Stopped.\n")

gpio.output(led,0)

gpio.output(buz,0)

main()

elif choice == 2:

capture\_image()

elif choice == 3:

try:

record\_video()

finally:

gpio.output(led,0)

gpio.output(buz,0)

main()

elif choice == 4:

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:

#Uncomment the next line to change your Pi's Camera rotation (in degrees)

#camera.rotation = 90

camera.start\_recording(output, format='mjpeg')

try:

address = ('', 8000)

server = StreamingServer(address, StreamingHandler)

server.serve\_forever()

finally:

camera.stop\_recording()

gpio.cleanup()

main()

else:

main()

except:

print('Exit Successfully')

finally:

gpio.output(led,0)

gpio.output(buz,0)

gpio.cleanup()

main()

**7.3 Code for Buzzer**

import RPi.GPIO as gpio

from time import sleep

gpio.setmode(gpio.BOARD)

buz = 3

gpio.setup(buz,gpio.OUT)

try:

while 1:

gpio.output(buz,1)

sleep(0.1)

gpio.output(buz,0)

sleep(0.1)

finally:

gpio.output(buz,0)

gpio.cleanup()

**7.4 Code for PIR Motion Sensor**

import RPi.GPIO as gpio

from picamera import PiCamera

import time

from time import sleep

gpio.setmode (gpio.BOARD)

led = 18

pir = 7

gpio.setup(led,gpio.OUT)

gpio.setup(pir,gpio.IN)

try:

sleep(0.5)

while True:

pirinput = gpio.input(pir)

if pirinput == 1:

print("Detected..!!")

## gpio.output(led,True)

## sleep(0.2)

while pirinput == 1:

gpio.output(led,True)

sleep(0.1)

gpio.output(led,False)

sleep(0.1)

pirinput = gpio.input(pir)

## sleep(0.01)

sleep(0.1)

else:

print("Not detected..!!")

gpio.output(led,False)

sleep(0.1)

sleep(0.01)

except:

gpio.cleanup()

gpio.output(led,False)

##camera = PiCamera()

##

##camera.start\_preview()

##print("Camera turned On")

##sleep(10)

##camera.stop\_preview()

##print("Camera turned Off")

##camera.close()

**7.5 Code for Web Streaming**

# Web streaming example

# Source code from the official PiCamera package

# http://picamera.readthedocs.io/en/latest/recipes2.html#web-streaming

import io

import picamera

import logging

import socketserver

from threading import Condition

from http import server

PAGE="""\

<html>

<head>

<title>Raspberry Pi - Surveillance Camera</title>

</head>

<body>

<center><h1>Raspberry Pi - Surveillance Camera</h1></center>

<center><img src="stream.mjpg" width="640" height="480"></center>

</body>

</html>

"""

class StreamingOutput(object):

def \_\_init\_\_(self):

self.frame = None

self.buffer = io.BytesIO()

self.condition = Condition()

def write(self, buf):

if buf.startswith(b'\xff\xd8'):

# New frame, copy the existing buffer's content and notify all

# clients it's available

self.buffer.truncate()

with self.condition:

self.frame = self.buffer.getvalue()

self.condition.notify\_all()

self.buffer.seek(0)

return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):

def do\_GET(self):

if self.path == '/':

self.send\_response(301)

self.send\_header('Location', '/index.html')

self.end\_headers()

elif self.path == '/index.html':

content = PAGE.encode('utf-8')

self.send\_response(200)

self.send\_header('Content-Type', 'text/html')

self.send\_header('Content-Length', len(content))

self.end\_headers()

self.wfile.write(content)

elif self.path == '/stream.mjpg':

self.send\_response(200)

self.send\_header('Age', 0)

self.send\_header('Cache-Control', 'no-cache, private')

self.send\_header('Pragma', 'no-cache')

self.send\_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')

self.end\_headers()

try:

while True:

with output.condition:

output.condition.wait()

frame = output.frame

self.wfile.write(b'--FRAME\r\n')

self.send\_header('Content-Type', 'image/jpeg')

self.send\_header('Content-Length', len(frame))

self.end\_headers()

self.wfile.write(frame)

self.wfile.write(b'\r\n')

except Exception as e:

logging.warning(

'Removed streaming client %s: %s',

self.client\_address, str(e))

else:

self.send\_error(404)

self.end\_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):

allow\_reuse\_address = True

daemon\_threads = True

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:

output = StreamingOutput()

#Uncomment the next line to change your Pi's Camera rotation (in degrees)

#camera.rotation = 90

camera.start\_recording(output, format='mjpeg')

try:

address = ('', 8000)

server = StreamingServer(address, StreamingHandler)

server.serve\_forever()

finally:

camera.stop\_recording()

**7.6 Code for django Web Frame work**

{% load staticfiles %}

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Home Security Automation</title>

</head>

<body>

<div role="main">

{% block head %}

<h2 text-align="center">HOME SECURITY AUTOMATION</h2>

{% endblock %}

{% block content %}

<form method="POST" action=''>{% csrf\_token %}

<div align="center">

<ul>

{% url 'turnOn' as turnOn %}

{% url 'turnOff' as turnOff %}

<li>

<a href="{% url 'turnOn' %}">Turn On</a>

</li>

<li>

<a href="{% url 'turnOff' %}">Turn Off</a>

</li>

<li>

<a href="{% url 'Monitoring' %}">Monitoring</a>

</li>

<li>

<a href="{% url 'stopMonitoring' %}">Stop Monitoring</a>

</li>

</ul>

</div>

</form>

{% endblock %}

{% block footer %}

{% endblock %}

</div>

{% block script %}

{% endblock %}

</body>

</html>

**7.7 code for Start Moitoring**

{% load staticfiles %}

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Home Security Automation</title>

</head>

<body method="POST" action=''>

<div method="POST" action='' role="main">

{% block head %}

<h2 text-align="center">HOME SECURITY AUTOMATION</h2>

<h4 text-align="center">Monitoring</h4>

{% endblock %}

{% block content %}

<div method="POST" action='' style="height:50px; width:300px; backgroung-color:blue; align="center">

<form method="POST" action=''>{% csrf\_token %}

<div>

Monitoring Mode:

<input type="radio" name="mmradio" id="mmradio1" value="On" selected="True">ON

<input type="radio" name="mmradio" id="mmradio2" value="Off">OFF

<input type="submit" name"submit" value="Submit">

</div>

</form>

</div>

<div>

</div>

{% endblock %}

{% block footer %}

{% endblock %}

</div>

{% block script %}

{% endblock %}

</body>

</html>

**CHAPTER VIII**

**Conclusion And Recommendation**

**Conclusion and Future Scope**

Home Security Automation demonstrates the collaboration of IOT with home automation, which provides the facility of obtaining the real-time data regarding Home security, movement detection remotely. This ability is attained by an online dashboard which are secured by providing access control. The access control is provided by online web. The registering of the legitimate user is done by an administrator of the system. Our system consists, online dashboard, and the hardware Ethernet Shield for the management, security, storage and real-time availability of data availability of data. The User Interface of application is user-friendly which provides an abstraction of the complexity of the system.

The Home Security Automation based on IOT can be annexed in future by introducing extra sensors concerning greater security and automation. Addition of HD camera will increase security and will provide 24-hour surveillance from anywhere. Gas detection module and vibration sensor module can be combined to the system which will provide gas leakage alerts and seismic alerts, respectively. Also, a cryptosystem can be added resulting in further enhanced security and access control mechanism. In this fashion, there are vast areas where home security automation based on IOT can be extended.

## References

1. Hill, Jim (12 September 2015). [*"The smart home: a glossary guide for the perplexed"*](http://www.t3.com/features/the-smart-home-guide). T3. Retrieved *27 March* 2017.
2. [IEEE 802.11-2007: “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications"](https://standards.ieee.org/getieee802/802.11.html).
3. <https://www.quora.com/Who-coined-the-term-internet-of-things>.
4. ["Molluscan eye"](http://molluscan-eye.epoc.u-bordeaux1.fr/index.php?rubrique=accueil&lang=en/). Retrieved 26 June 2015.
5. Erlich, Yaniv (2015). ["A vision for ubiquitous sequencing"](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4579324). *Genome Research*. **25** (10): 1411–1416. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1101/gr.191692.115](https://doi.org/10.1101%2Fgr.191692.115). [ISSN](https://en.wikipedia.org/wiki/International_Standard_Serial_Number) [1088-9051](https://www.worldcat.org/issn/1088-9051). [PMC](https://en.wikipedia.org/wiki/PubMed_Central) [4579324](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4579324) . [PMID](https://en.wikipedia.org/wiki/PubMed_Identifier) [26430149](https://www.ncbi.nlm.nih.gov/pubmed/26430149).
6. Wigmore, I. (June 2014). ["Internet of Things (IoT)"](http://whatis.techtarget.com/definition/Internet-of-Things). *TechTarget*.
7. Noto La Diego, Guido; Walden, Ian (1 February 2016). "Contracting for the 'Internet of Things': Looking into the Nest". Queen Mary School of Law Legal Studies Research Paper No. 219/2016.
8. http://www.afcdud.com/fr/smart-city/422-how-the-history-of-smart-homes.html
9. <https://www.lifewire.com/what-do-http-and-https-stand-for-3482375>
10. *Raspberry pi.org*