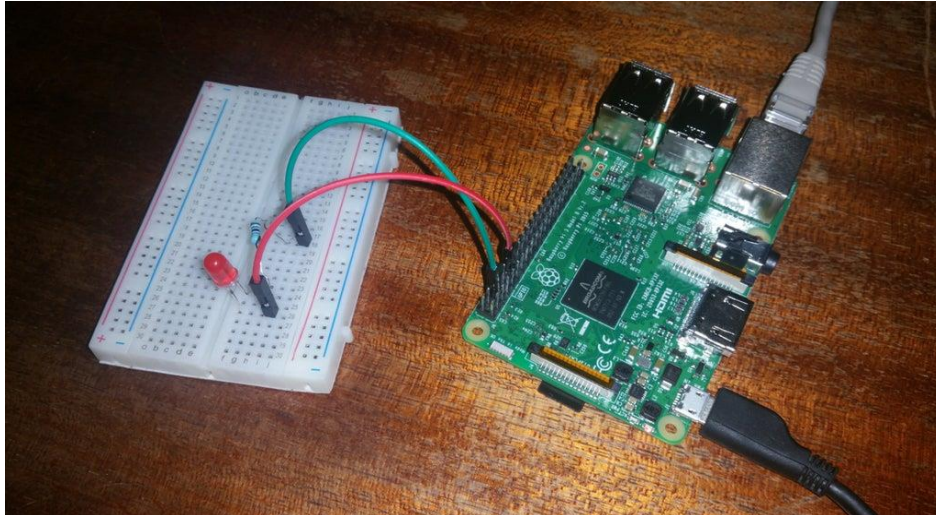


Ex.no.1 Blink and Pattern using Arduino or Raspberry Pi

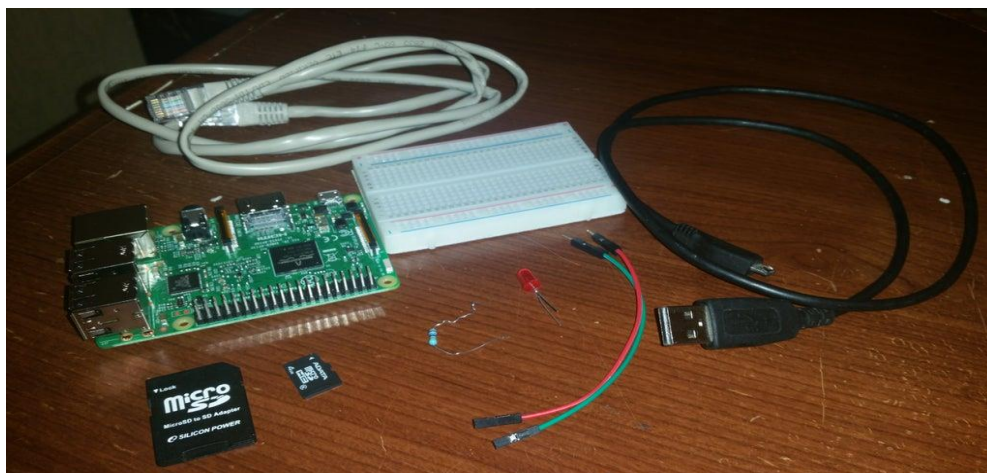
Aim

To Develop an application for LED Blink and Pattern using Arduino or Raspberry Pi

Procedure

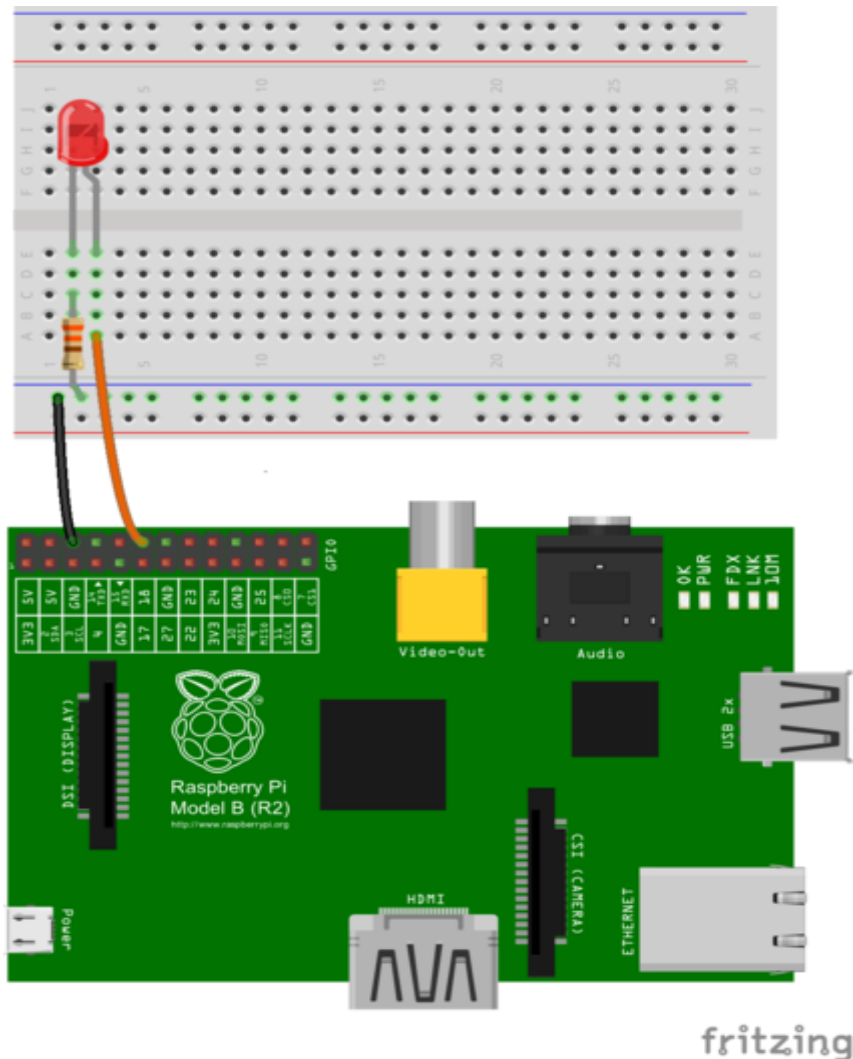


Step 1: Needed Components



- 1 x Raspberry Pi
- 1 x USB cable
- 1 x LED
- 1 x Breadboard
- 1 x SD Card and adapter (4GB minimum)
- 1 x LAN cable
- 1 x 50-ohm resistor
- 2 x Jumper wires

Step 2: Building the Circuit



Every LED has two sides - one **negative** and one **positive**. Choose the negative one and using the resistor, connect it up to GND (**pin 6**). The other end goes to **pin 18**. Feel free to use the picture as a reference.

Step 3: Setting Up the Raspberry

```
pi@raspberrypi:~ $ sudo apt-get install python3
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.5.3-1).
python3 set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $ sudo apt-get install python
Reading package lists... Done
Building dependency tree
Reading state information... Done
python is already the newest version (2.7.13-2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Now, we need to install Python or Python 3.

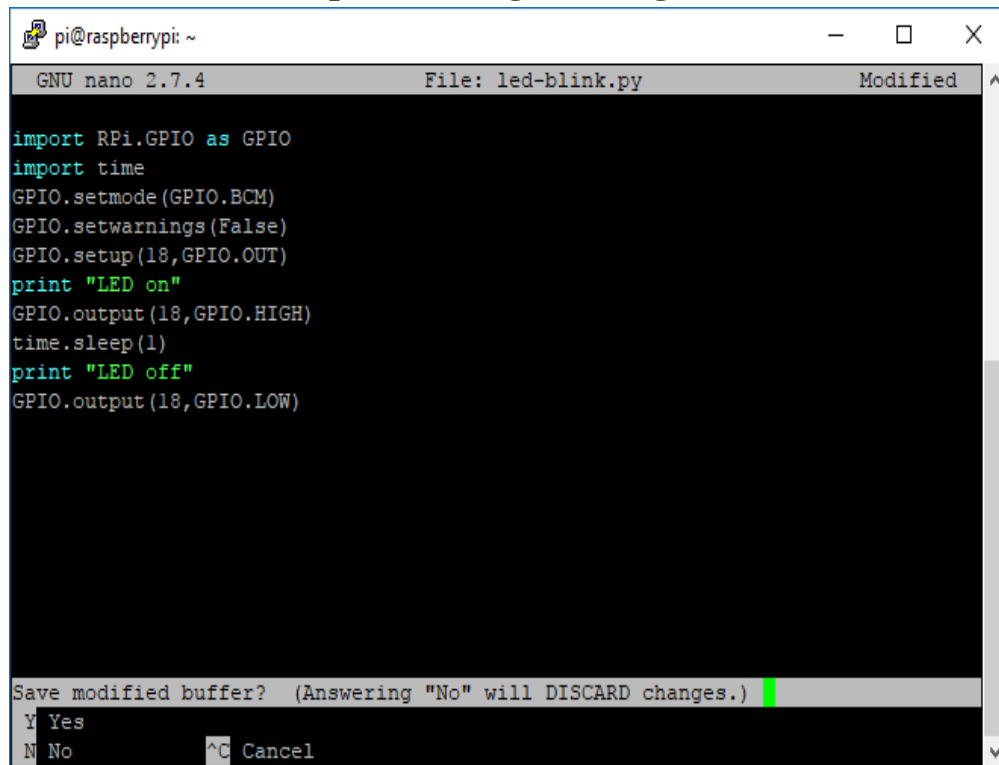
Enter the following command:

sudo apt-get install python

or

sudo apt-get install python3

Step 4: Writing the Program



```
pi@raspberrypi: ~
GNU nano 2.7.4      File: led-blink.py      Modified
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
print "LED on"
GPIO.output(18,GPIO.HIGH)
time.sleep(1)
print "LED off"
GPIO.output(18,GPIO.LOW)

Save modified buffer? (Answering "No" will DISCARD changes.)
Y Yes
N No      ^C Cancel
```

Paste the following code in the newly-create file:

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(18,GPIO.OUT)

print "LED on"

GPIO.output(18,GPIO.HIGH)

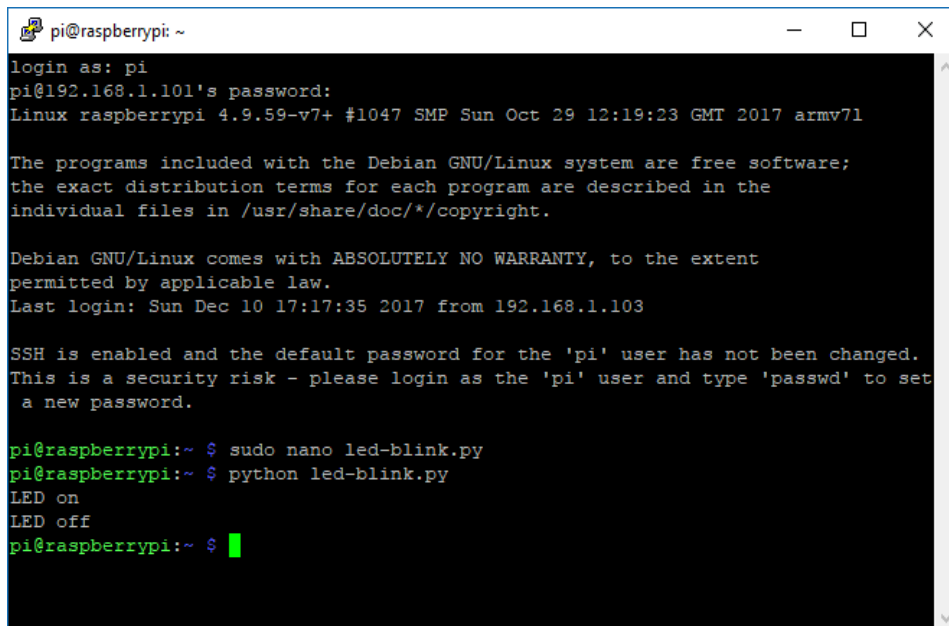
time.sleep(1)

print "LED off"

GPIO.output(18,GPIO.LOW)

Save the file and go back to the console.

Step 5: Running the Program

A terminal window titled 'pi@raspberrypi: ~' with standard window controls. The terminal output shows a login sequence for the 'pi' user at IP 192.168.1.101, followed by system information and a security warning about SSH. The user then runs 'sudo nano led-blink.py' and 'python led-blink.py', which outputs 'LED on' and 'LED off' respectively. The prompt returns to the user.

```
pi@raspberrypi: ~
login as: pi
pi@192.168.1.101's password:
Linux raspberrypi 4.9.59-v7+ #1047 SMP Sun Oct 29 12:19:23 GMT 2017 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Dec 10 17:17:35 2017 from 192.168.1.103

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $ sudo nano led-blink.py
pi@raspberrypi:~ $ python led-blink.py
LED on
LED off
pi@raspberrypi:~ $
```

Result

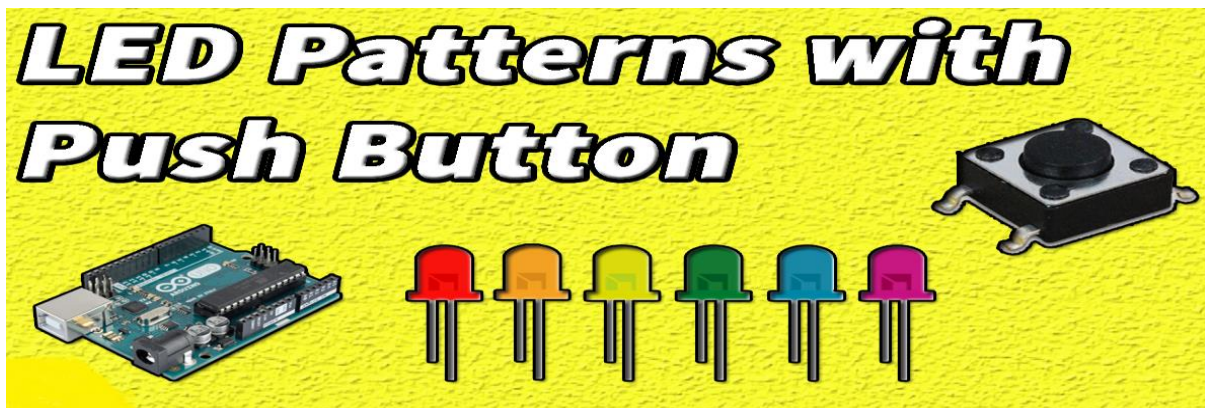
Thus an application for LED Blink and Pattern using Arduino or Raspberry Pi was created and run successfully

Ex.No.2 LED Pattern with Push button using Arduino

Aim

To Develop an application for LED Pattern with Push Button Control using Arduino or Raspberry Pi

Procedure



This project is an advancement of simple Led to multiple LEDs with Arduino and a push-button which switches them at a very fast rate. Whenever you press a push-button it generates a unique pattern that is represented with the help of LEDs. This is a very interesting project Follow the steps below to make it.

Hardware Components

Following are the necessary hardware components:

S.NO	Components	Qty
1.	Arduino Uno R3	1
2.	USB Cable A/B	1
3.	Push Button	1
4.	Resistors (10k,470 ohm)	1,7
5.	5mm LED	7
6.	Breadboard	1
7.	Connecting Wires	1

Working Explanation

As mentioned earlier as well that this project of an LED pattern is an advancement of simple LEDs to multiple LEDs. After making a simple Led project of turning it on and turning it off with the help of Arduino you can make several other projects and learn new coding with it. In Led pattern project we are using 7 LEDs connected to digital pins of Arduino and a push-button is used to generate a unique pattern on LEDs. Every time a push-button is pressed, it switches the lighting pattern from previous values to the new outputs which are the main idea of the project. This project LED pattern is a great step of practicing your programming skills and learning some new projects.

Connection

- STEP # 1 (Make Push Button Connections)
Pin1 to 5V of Arduino.
Resistor 10k B/w Pin2 of Push Btn & GND of Arduino
Pin2 is also connected to D6 of Arduino
- STEP # 2 (Make LED Connections)
Connect All -VE of LED To GND to Arduino
- STEP # 3 (Make Resistors Connections)
All Resistor's to +VE of LED and then D7,D8,D9,D10,D11,D12,D13 of Arduino
- STEP # 4 (Upload Code)

Circuit Diagram

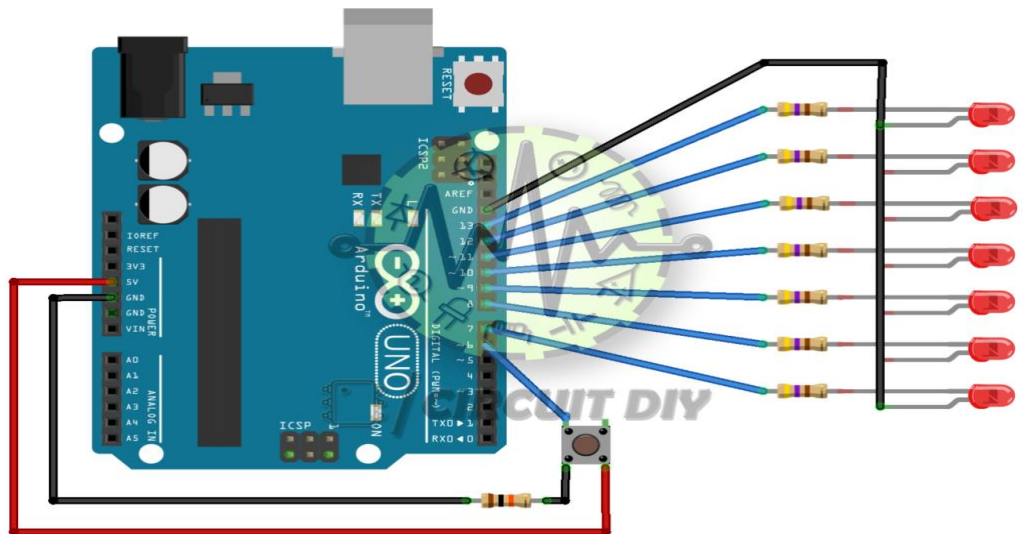


Fig - LED patterns circuit

Code

```
int L1 = 13;
int L2 = 12;
int L3 = 11;
int L4 = 10;
int L5 = 9;
int L6 = 8;
int L7 = 7; //7 LED pin

int buttonPin = 6; //the number of the pushbutton pin

int de=50; // delay time

int p=0; // variable for pattem

int buttonState = 0; // variable for reading the pushbutton status

void setup() {

pinMode(L1, OUTPUT);
pinMode(L2, OUTPUT);
pinMode(L3, OUTPUT);
pinMode(L4, OUTPUT);
```



```

pinMode(L5, OUTPUT);
pinMode(L6, OUTPUT);
pinMode(L7, OUTPUT);

pinMode(buttonPin, INPUT);

}

void loop()
{
buttonState = digitalRead(buttonPin);

if (buttonState == HIGH)

    {
    p++;
    delay(2000);
    }

if(p==1)
{
digitalWrite(L1,1);
digitalWrite(L2,0);
digitalWrite(L3,0);
digitalWrite(L4,0);
digitalWrite(L5,0);
digitalWrite(L6,0);
digitalWrite(L7,0); //1
delay(de);

digitalWrite(L1,0);
digitalWrite(L2,1);
digitalWrite(L3,0);
digitalWrite(L4,0);
digitalWrite(L5,0);
digitalWrite(L6,0);
digitalWrite(L7,0); //2
delay(de);

digitalWrite(L1,0);
digitalWrite(L2,0);
digitalWrite(L3,1);
digitalWrite(L4,0);
digitalWrite(L5,0);

```

```
digitalWrite(L6,0);  
digitalWrite(L7,0); //3  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,1);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //4  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,1);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //5  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,1);  
digitalWrite(L7,0); //6  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,1); //7  
delay(de);  
}
```

```
if(p==2)
{
digitalWrite(L1,0);
digitalWrite(L2,0);
digitalWrite(L3,0);
digitalWrite(L4,0);
digitalWrite(L5,0);
digitalWrite(L6,0);
digitalWrite(L7,1); //7
delay(de);
```

```
digitalWrite(L1,0);
digitalWrite(L2,0);
digitalWrite(L3,0);
digitalWrite(L4,0);
digitalWrite(L5,0);
digitalWrite(L6,1);
digitalWrite(L7,0); //6
delay(de);
```

```
digitalWrite(L1,0);
digitalWrite(L2,0);
digitalWrite(L3,0);
digitalWrite(L4,0);
digitalWrite(L5,1);
digitalWrite(L6,0);
digitalWrite(L7,0); //5
delay(de);
```

```
digitalWrite(L1,0);
digitalWrite(L2,0);
digitalWrite(L3,0);
digitalWrite(L4,1);
digitalWrite(L5,0);
digitalWrite(L6,0);
digitalWrite(L7,0); //4
delay(de);
```

```
digitalWrite(L1,0);
digitalWrite(L2,0);
digitalWrite(L3,1);
digitalWrite(L4,0);
digitalWrite(L5,0);
digitalWrite(L6,0);
```

```
digitalWrite(L7,0); //3  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,1);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //2  
delay(de);
```

```
digitalWrite(L1,1);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //1  
delay(de);
```

```
}
```

```
if(p==3)  
{  
digitalWrite(L1,1);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //1  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,1);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //2  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,1);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //3  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,1);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //4  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,1);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //5  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,1);  
digitalWrite(L7,0); //6  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,1); //7  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,1);  
digitalWrite(L7,0); //6  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,1);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //5  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,0);  
digitalWrite(L4,1);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //4  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,0);  
digitalWrite(L3,1);  
digitalWrite(L4,0);  
digitalWrite(L5,0);  
digitalWrite(L6,0);  
digitalWrite(L7,0); //3  
delay(de);
```

```
digitalWrite(L1,0);  
digitalWrite(L2,1);  
digitalWrite(L3,0);  
digitalWrite(L4,0);  
digitalWrite(L5,0);
```

```
digitalWrite(L6,0);
digitalWrite(L7,0); //2
delay(de);
}
```

```
if(p==4)
{
digitalWrite(L1,1);
digitalWrite(L2,0);
digitalWrite(L3,0);
digitalWrite(L4,0);
digitalWrite(L5,0);
digitalWrite(L6,0);
digitalWrite(L7,1); //1,7
delay(de);
```

```
digitalWrite(L1,0);
digitalWrite(L2,1);
digitalWrite(L3,0);
digitalWrite(L4,0);
digitalWrite(L5,0);
digitalWrite(L6,1);
digitalWrite(L7,0); //2,6
delay(de);
```

```
digitalWrite(L1,0);
digitalWrite(L2,0);
digitalWrite(L3,1);
digitalWrite(L4,0);
digitalWrite(L5,1);
digitalWrite(L6,0);
digitalWrite(L7,0); //3,5
delay(de);
```

```
digitalWrite(L1,0);
digitalWrite(L2,0);
digitalWrite(L3,0);
digitalWrite(L4,1);
digitalWrite(L5,0);
digitalWrite(L6,0);
digitalWrite(L7,0); //4
```



```
delay(de);
```

```
}
```

```
if(p==5)
```

```
{
```

```
digitalWrite(L1,0);
```

```
digitalWrite(L2,0);
```

```
digitalWrite(L3,0);
```

```
digitalWrite(L4,1);
```

```
digitalWrite(L5,0);
```

```
digitalWrite(L6,0);
```

```
digitalWrite(L7,0); //4
```

```
delay(de);
```

```
digitalWrite(L1,0);
```

```
digitalWrite(L2,0);
```

```
digitalWrite(L3,1);
```

```
digitalWrite(L4,0);
```

```
digitalWrite(L5,1);
```

```
digitalWrite(L6,0);
```

```
digitalWrite(L7,0); //3,5
```

```
delay(de);
```

```
digitalWrite(L1,0);
```

```
digitalWrite(L2,1);
```

```
digitalWrite(L3,0);
```

```
digitalWrite(L4,0);
```

```
digitalWrite(L5,0);
```

```
digitalWrite(L6,1);
```

```
digitalWrite(L7,0); //2,6
```

```
delay(de);
```

```
digitalWrite(L1,1);
```

```
digitalWrite(L2,0);
```

```
digitalWrite(L3,0);
```

```
digitalWrite(L4,0);
```

```
digitalWrite(L5,0);
```

```
digitalWrite(L6,0);
```

```
digitalWrite(L7,1); //1,7
```

```
delay(de);
```

```

    }

    if(p==6)
    {
        digitalWrite(L1,1);
        delay(de);
        digitalWrite(L2,1);
        delay(de);
        digitalWrite(L3,1);
        delay(de);
        digitalWrite(L4,1);
        delay(de);
        digitalWrite(L5,1);
        delay(de);
        digitalWrite(L6,1);
        delay(de);
        digitalWrite(L7,1); //1,7
        delay(de);
        digitalWrite(L7,0); //1,7
        delay(de);
        digitalWrite(L6,0);
        delay(de);
        digitalWrite(L5,0);
        delay(de);
        digitalWrite(L4,0);
        delay(de);
        digitalWrite(L3,0);
        delay(de);
        digitalWrite(L2,0);
        delay(de);
        digitalWrite(L1,0);
        delay(de);

    }

    if(p==7)
    {
        digitalWrite(L1,0);
        digitalWrite(L2,0);
        digitalWrite(L3,0);
        digitalWrite(L4,0);
        digitalWrite(L5,0);
        digitalWrite(L6,0);
    }

```

```
digitalWrite(L7,0); //1,7  
p=0;  
}  
}
```

Application

- This project can be used as an indication of something like errors
- With the help of this project, you can make your secret lock pattern

Result

Thus an application for LED Pattern with Push Button Control using Arduino or Raspberry Pi was created and run successfully

Ex.No.3 Arduino Temperature Sensor Using LM35

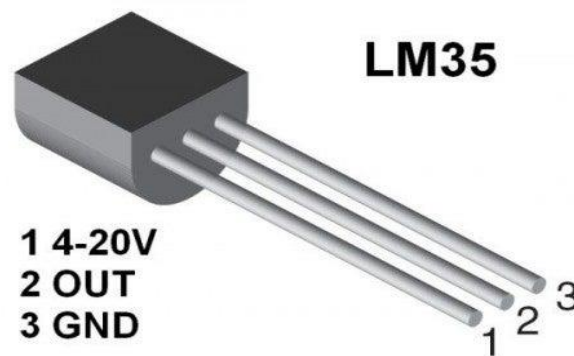
Aim

To Develop an application for LM35 Temperature Sensor to display temperature values using arduino or Raspberry Pi

Procedure

Introduction

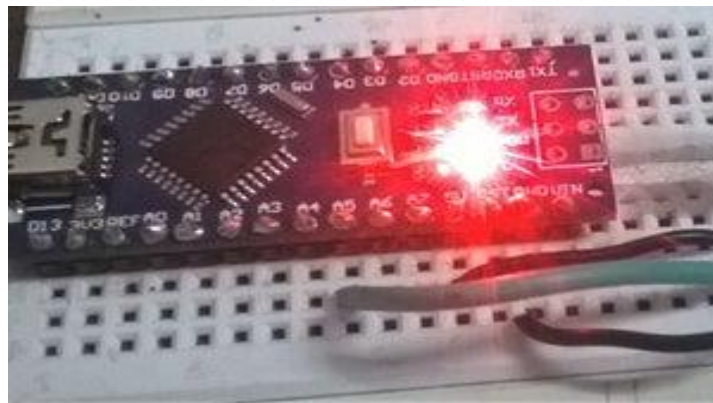
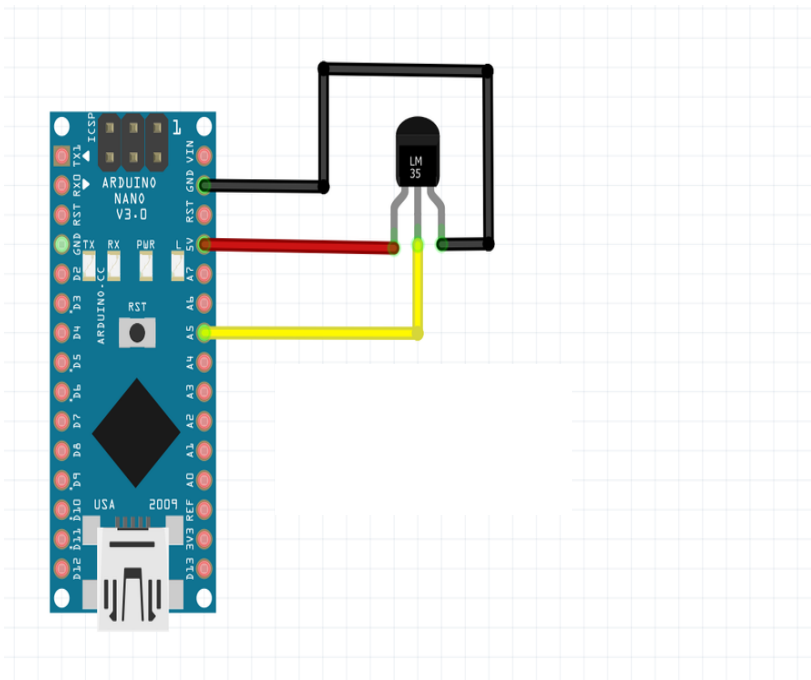
The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly proportional to the Centigrade temperature. LM35 is three terminal linear temperature sensor from National semiconductors. It can measure temperature from -55 degree Celsius to +150 degree Celsius. The voltage output of the LM35 increases 10mV per degree Celsius rise in temperature. LM35 can be operated from a 5V supply and the stand by current is less than 60uA.



Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full -55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates from 4 V to 30 V
- Less than 60-μA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only $\pm 1/4^\circ\text{C}$ Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

Step 1: Hardware Required and Circuit Diagram



- Arduino Board (Any)
- LM35 Sensor
- Bread Board

Connect the Circuit as shown in image and upload the following code.

Step 2: Programming the Arduino

```
const int sensor=A5; // Assigning analog pin A5 to variable 'sensor'
float tempc; //variable to store temperature in degree Celsius
float tempf; //variable to store temperature in Fahrenheit
float vout; //temporary variable to hold sensor reading
void setup() {
  pinMode(sensor,INPUT); // Configuring sensor pin as input
  Serial.begin(9600);
}
```

```

void loop() {
  vout=analogRead(sensor); //Reading the value from sensor
  vout=(vout*500)/1023;
  tempc=vout; // Storing value in Degree Celsius
  tempf=(vout*1.8)+32; // Converting to Fahrenheit
  Serial.print("in DegreeC=");
  Serial.print("\t");
  Serial.print(tempc);
  Serial.print(" ");
  Serial.print("in Fahrenheit=");
  Serial.print("\t");
  Serial.print(tempf);
  Serial.println();
  delay(500); //Delay of 1 second for ease of viewing }

```

Step 3: Output Result

The screenshot shows the Arduino IDE interface. On the left, the code from the previous block is displayed. The status bar at the bottom indicates 'Arduino Nano, ATmega328P on COM4'. On the right, the serial monitor shows the output of the code. The output consists of multiple lines, each containing two temperature readings: one in degrees Celsius and one in degrees Fahrenheit, separated by a tab and a space. The values are approximately 25.90°C (78.63°F) and 26.39°C (79.51°F).

```

in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.42 in Fahrenheit= 77.75
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 26.39 in Fahrenheit= 79.51
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 26.39 in Fahrenheit= 79.51
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 26.39 in Fahrenheit= 79.51
in DegreeC= 26.39 in Fahrenheit= 79.51
in DegreeC= 26.39 in Fahrenheit= 79.51
in DegreeC= 26.39 in Fahrenheit= 79.51
in DegreeC= 26.39 in Fahrenheit= 79.51
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63
in DegreeC= 25.90 in Fahrenheit= 78.63

```


Result

Thus an application for LM35 Temperature Sensor to display temperature values using arduino or Raspberry Pi

Ex.No.4 Forest fire detection system using IoT sensor network

Aim

To Develop an application for Forest fire detection end node using Raspberry Pi device and sensor

Procedure

Introduction

we will be detecting a forest fire using an IoT sensor network and making a portable device with battery operation. Getting information about a fire hazard in a forest in time can prevent the forest fire from spreading. The sensor network can give the particular location of the fire as we already know where the sensors are installed.

Purpose

1. Detect- Sensing the forest fire
2. Report – Report to the server
3. Locate – Locate the location of fire occurrence
4. React – Alerting the forest fire team

Architecture

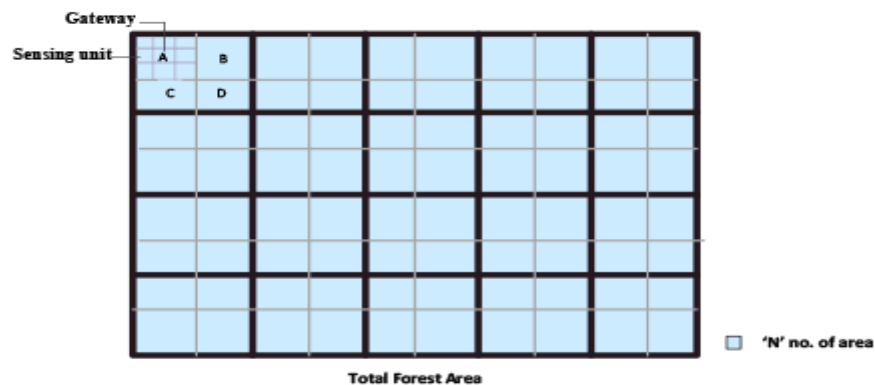


Fig 1

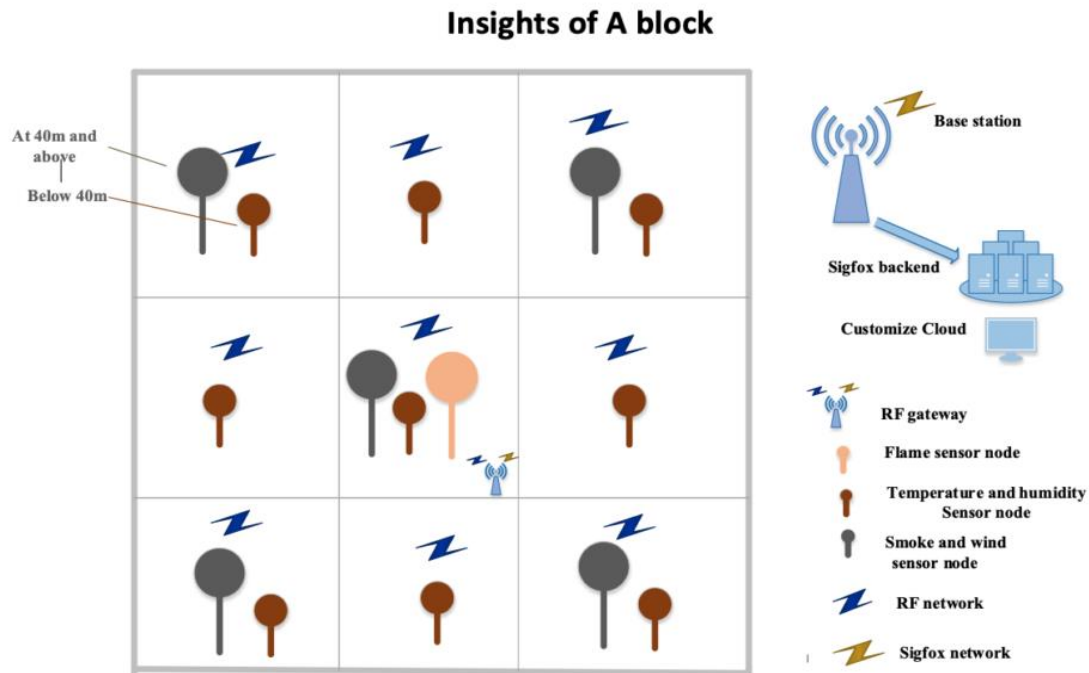


Fig 2

Strategy

1. We can divide the forest in 'N' no. of areas as seen in Fig 1.
2. Each area is again divided into four cells- A, B, C, and D. Each cell will have several sensing units, which is shown in fig 2. The sensors are not evenly distributed; some sensors are in more quantity than another sensor.

The purpose of using the uneven amount of sensors:

Some environmental parameters do not change rapidly and can be sensed by fewer sensors in a large area.

3. The sensor unit consists of mainly three parts:

- Sensor unit
- RF (Radio frequency) Unit with (Inbuilt controller)
- Battery management unit

The sensor unit consists of several sensor nodes, as described in fig.2, and each node will have an RF module and a battery for a power source.

RF unit collects the data from the sensor nodes and transmits it to the Gateway. The Gateway listens for the incoming data from pre-registered nodes and forwards those data to the SIGFOX base stations. From those base stations, the data gets transferred to the SIGFOX backend, and in this above architecture, we are using our customized cloud. All the data will be shifted to the cloud via the SIGFOX call back function. Our customize cloud is supposed to do the major processing at the cloud end only; the cloud will make the decisions and provide the alerts to the team in case of fire detection.

IGFOX module selection

The module is selected for the RCZ1 region, which includes European countries and covers the Middle East. The operating frequency band for these countries is 868MHz.

We can take **AX-SIGFOX-API** trans-receiver (up-link and downlink both) SoC which will

fulfill our application requirement with the additional functionality.

Operating Voltage	1.8V – 3.6V
Transmit Power	14 dB with TX current consumption: 49mA @3V
Receiver Sensitivity	-126 dB with RX current consumption: 10mA @3V
Standby current	0.5mA
Sleep current	1.3uA
Cost	\$1.67
Operating Mode	AT command Mode and API mode

Different sensor units with causes of false alarm and their prevention technique

Sensing Unit

We can detect the fire by using the following sensing unit.

1) Smoke and gas detector

Smoke detector – For early detection of fire, we can use two types of smoke detectors

-Ionization smoke detector – Fast response for open flame fire (high energy) as this type of fire produce small smoke particles.

-Photoelectric smoke detector – Fast response to a smoldering fire.

Gas detector The sensors have to be sensitive enough to detect even very low smoke concentrations. For this reason, gas sensors or a combination of gas sensors together with an aspiration system should be used. It consists of majorly CO₂, CO, NO₂, CH₂, H₂ gas sensors.

Causes of false alarm by smoke detector

Fog and cloud, Non-smoke objects like plants and animals.

Solution

We can also reduce the false detection rate by considering the result of the gas sensor and other sensors (heat and humidity).

2) Thermal detector

The smoldering and open flame raises the temperature of the surrounding. By using a temperature sensor, we can detect the change in the environmental condition.

We have two modes of temperature sensing

-Rate of rise – This will respond quickly to the high flame fire

-Fixed- temperature – This will respond to the slowly increasing smoldering or ground fire when the temperature reaches a pre-defined threshold level.

Cause of false alarm- Sunlight

Solution

By analyzing the temperature of the whole forest- On a sunny day, the temperature will rise uniformly all over the forest region, all the sensors will give an approximately same reading.

By considering the season of that area.
By implementing the thermal detector in a shady area

3) Flame detector

Flame detectors optically sense either the ultraviolet (UV) and infrared (IR) radiation flames give off. Flame detectors are “line of sight” devices, and they are subject to being blocked by objects placed in front of them.

Mode of flame detector

We can use our detector in scanning mode to prevent the flame detector from being covered by any obstacle. In this mode, the device will rotate by 360°, and it stops when the signal is received. The detector will only alert when the signal persists for a specific set of times.

Causes of a false alarm: Sunlight and non-smoke object

Solution

By using such filters, which blocks the solar radiation.
In the case of non-smoke objects, we can consider the result of other sensors also.

4) Wind speed detector and Humidity detector

Temperature affects the humidity as well as wind speed. By measuring both parameters, we can detect fire in that area.

Detection algorithm

The algorithm plays a crucial role in minimizing the false alarm rate in our system. Instead of relying on one or two sensor data, we will analyze the data from all the sensors, resulting in fewer chances of false alarms.

Gas and temperature detector data

S gas, Temp is equal to

$$[(NH_2, \text{filtered } KH_2 + NCO, \text{filtered } KCO + NCH, \text{filtered } KCH + NCO, \text{filtered } KCO + NCO_2, \text{filtered } KCO_2)(1 + NTemp, \text{filtered } KTemp)]$$

S gas, Temp \geq Threshold value \Rightarrow Activate temperature and gas sensor boolean

Humidity and smoke detector data

S Humidity, Smoke = (Humidity \geq TH) (Smoke \geq Ts) \Rightarrow Activate Humidity and smoke sensor boolean

TH = Threshold level of humidity

Ts = Threshold level of smoke

Flame detector data

S flame = (Flame data \geq Tt) \Rightarrow Activate flame sensor boolean

Tt = Threshold time for flame detector

The measured sensor data for smoke, gas, temperature, humidity, and flame will transmit to our Gateway and feed into the detection algorithm/customized cloud.

The decision of “active alarm/no alarm” results from data from all sensors compared to the threshold value.

False alarm caused by Human activity

– Camping and stubble fire

Solution

Permission from the forest department authority

Other techniques to reduce false alarm

I. FWI system (Fire Weather Index system)

This system will measure the risk of a forest fire or determine the possibility of a forest fire. It is composed of six components that individually and collectively account for the occurrence of fire.

If FWI is measuring the high risk of forest fire, then there are fewer chances of false alarms, and in the case of low fire risk, we can double-check the trustworthiness of the alarm.

II. Animal as Biological Sensor – Two different detection methods can be implemented.

These methods are the Thermal Detection (TD) method to measure instant temperature changes and the Animal Behavior Classification (ABC) method to classify sudden animal changes.

Decision-making unit

Customized Cloud

The personal cloud will accept the data from the SIGFOX backend through the callback APIs. The cloud will majorly perform these operations:

1. The cloud will collect the sensor data and log the data with the device ID, time, location, and RSSI strength.
2. The acceptable sensing range will be stored in the cloud, and the real-time sensing value will be compared and processed onto the cloud only.
3. Based on the backend calculations at the cloud, the cloud will alert the team in case of fire detection.
4. The cloud will be highly secured with TLS security enabled.

Maximizing Long life

Enclosure

We will use the ‘Ingress protection (IP) rating system to protect environmental effects like dust, dirt, wind, etc. Our enclosure is made of polymer, which can sustain high temperatures and other environmental extremes.

-Use of Faraday cage for blocking electromagnetic fields.

-The detector units are prone to dust, corrosion, and environmental extremes; for them, we can make a protective cap made of sintered metal which prevents them from soiling with dust and humidity

Sleep time

As the SIGFOX network can accept only 140 packets per day, we don’t need to sample the sensor data every second or minute. We will be enabling cyclic sleep or pin sleep on the sensor nodes. This will significantly reduce the power consumption of the device; this will maximize the device’s lifecycle.

Customized Cloud

We are doing all the processing at our cloud end only; the cloud will decide the case of fire occurrence and send an alert. This will significantly reduce the power consumption of our device.

Data transmission and battery life

As our SIGFOX device can only receive 140 messages per day, we have divided the total forest area in the 'N' section in our proposed design. Each section is then further divided into nine cells; each cell consists of 9 nodes. If we take an average, we can send 14 messages or alerts from each node per day.

-Per day, each node will only get active about 14 times and consumes more power; the rest of the time, it will remain in sleep mode.

-If the total time for transmission of data from one node to RF module is approx. 200ms.

Then each node will remain active for about 3s (14*200ms) per day.

Active time of one node

-Per day-active time – 3s

-10 years active time – 3h approx.

-As per theoretical calculation, the average power consumption of each node is approx. 1.6W per day.

Calculation of battery capacity and discharge time.

For instance, Rated voltage of battery = 3.6V

The rated capacity of battery = 2000mAh

Discharge rate = 0.25C or 500mA (discharge rate as per each node current consumption)

Total discharge time in hrs. = Battery capacity(mAh)/Discharge current(mA).

$T = 2000/500mA = 4h$

So we can use a battery of 2000mAh or above, which suits best to our design. Now we will discuss the chemistry/type of the battery, which should have a long shelf life.

Battery selection

The suitable battery for our system is primary batteries as they have a long shelf life, extensive temperature range, and low self-discharge rate and cost. As our device's maximum operating voltage is not more than 4v – 5V, we can use lithium-ion-based primary batteries as the Li-ion battery has a high energy density and long life.

Battery type

I. LiSOC12

II. LiMnO2

III. XOL series batteries

All the above batteries have a shelf life of up to 10 years with a low self-discharge current and extensive temperature range.

Other technique

Energy harvesting system

To increase battery life, we can also adopt the energy harvesting system as a primary battery operates our system, so we cannot charge them by energy harvesting system. But we can operate our device directly through an energy harvesting system.

Below is the list of a few available systems that can be installed as per the available source of energy

- i. Solar system – This system needs solar light to generate sufficient power.
- ii. Wind system – Can be installed where frequent wind flow is there
- iii. Electromagnetic energy system – Can be installed near transformer system which produces EM wave.

In brief description

How it works

For instance, if we take an electromagnetic (EM) energy system, some transducers convert the EM wave into an electrical current. We need to use the copper panel with the transducer for sensing the EM wave from the surroundings. The converted energy from the transducer is then used to power our device.

For another energy source like wind or solar energy, we need a different sensing system that will capture the solar energy/wind energy.

EM wave source

The source of energy for EM waves can be a transformer which emits EM waves. If we place our device near the transformer line, the energy harvesting system can capture the emitted EM wave from the transformer.

Switching unit

For powering our device by energy harvesting system, we need a switching circuit. We can use an electronics switch (transistor) or electromechanical switch (relay), which suits our design best. This will switch our device from the battery to the energy harvesting system. The switching will take place only when our energy harvesting system stores sufficient energy to power our device.

Combination of energy harvesting system

For more power storage, we can use two or three energy harvesting systems simultaneously, like a combination of solar and wind systems or two EM systems.

This is how we can detect fire in forests using an IoT sensor network.

Result

Thus an application for Forest fire detection end node using Raspberry Pi device and sensor was created and run successfully

Ex.No.5

Home intrusion detection system

Aim

To Develop an application for home intrusion detection web application

Procedure

Introduction

Designing the IOT system is complex as it involves interactions between various components. So I am going to show the step by step procedure to implement my project which is home intrusion detection system.

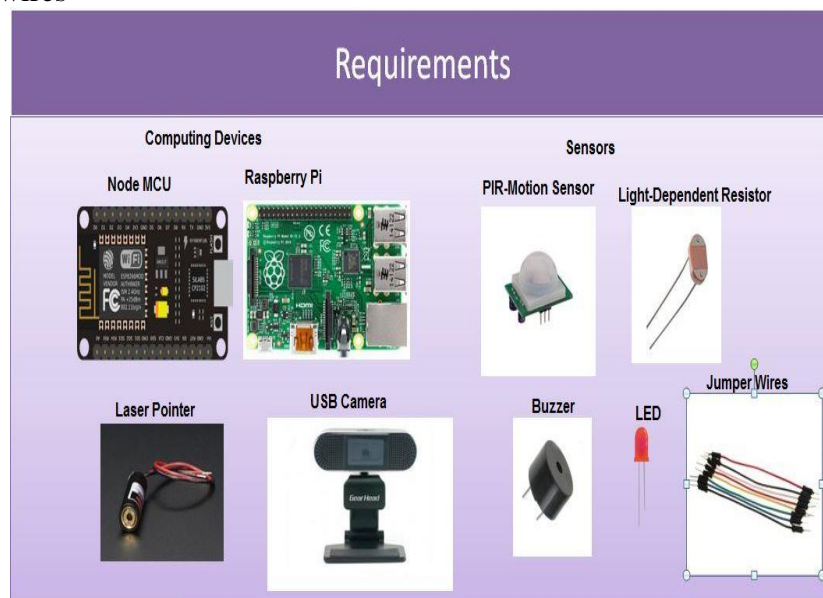
Step 1: Purpose and Requirement Specification

Purpose:

The purpose of home intrusion detection system is to detect intrusions using sensors and raise alerts, if necessary.

Requirements:

- ESP12E Node MCU board
- Raspberry Pi
- PIR sensor
- Light dependent Resistor
- Laser pointer
- Web cam
- Buzzer
- Some wires



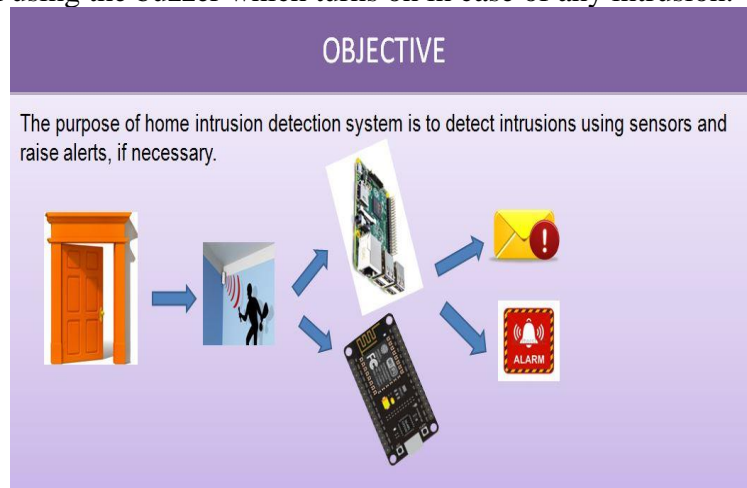
Behavior:

In case of any intrusion, I intend to capture a picture of the intruder, mail the image to the respective end users and alert them. I would like to use an alarm which goes on in case of an intrusion.

Work-Flow:

Using all the above requirements, I am going to detect the intrusion that is:

- With the help of ***Light dependent resistor and PIR motion sensor***, I am going to detect the motions in the room.
- If a motion is detected, I intend to capture the image with the help of a ***webCam*** and store locally
- Now the alerts are sent to the user with the captured image.
- Also I am using the buzzer which turns on in case of any intrusion.

**Data Analysis Requirement:**

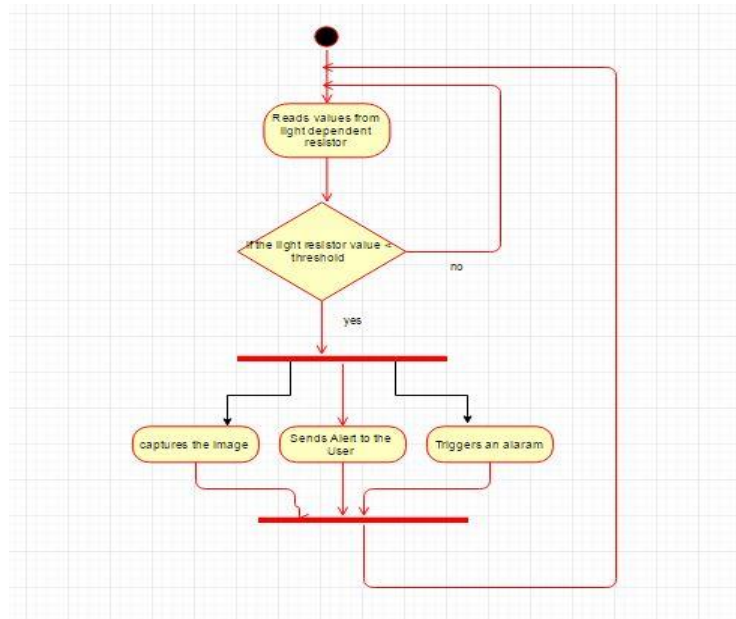
The light dependent resistor data is analyzed locally. If the sensor reading falls below the threshold value, an image is captured and send as an alert to the user.

Application Deployment Requirement:

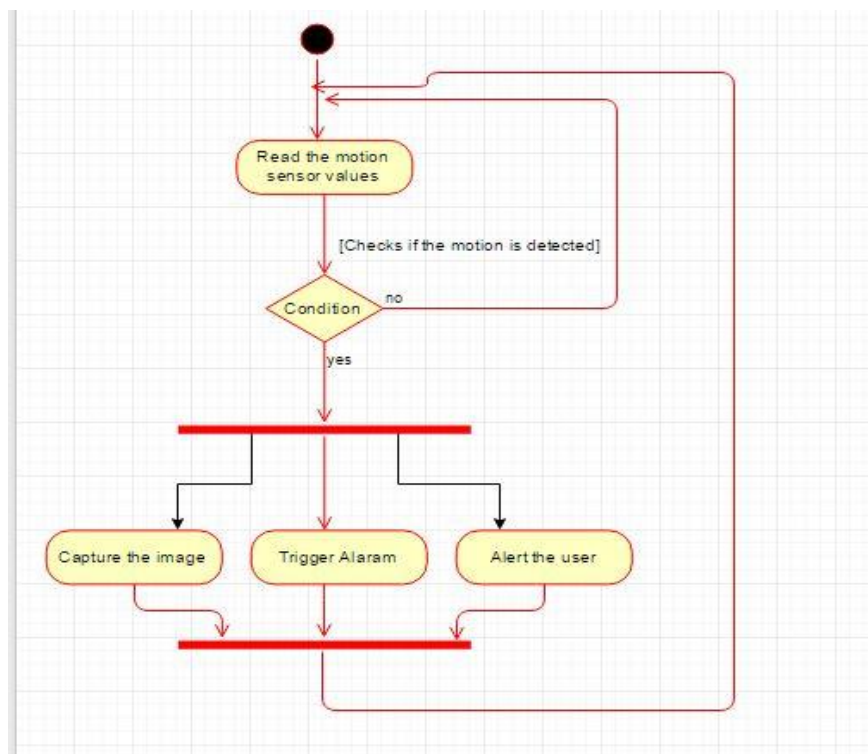
The application is deployed locally on the device and can be accessible from anywhere via node-red.

STEP 2: Process Specification

The process diagrams for home intrusion detection system are as shown below:



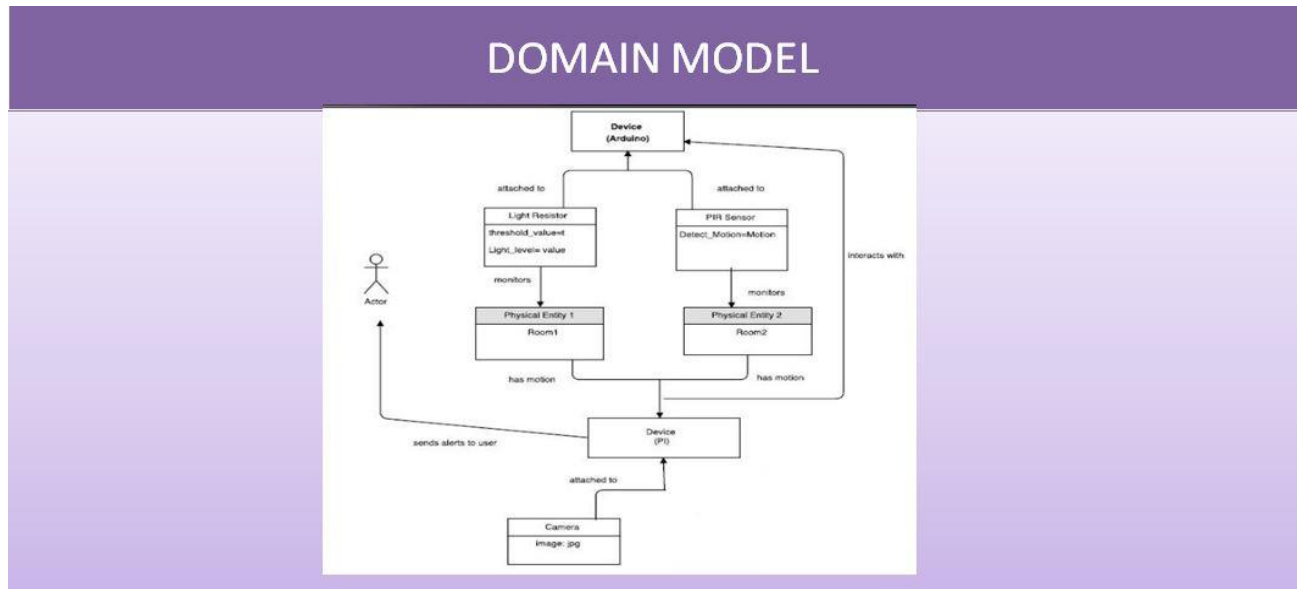
From the above process diagram, we can see that the process starts at the circle. The light dependent resistor which is shown in the rectangular box in the above figure is placed in a room which reads the light level. Here I am going to use a Laser pointer and I am going to fix it to the Light dependent resistor on the board. Now when the laser beam is breached we can see that there is a gradual fall on light level value. If the light level falls below the threshold value, an image is captured and an alert will be sent to the user. And also it turns on the buzzer. This decision (yes/no) to raise an alert is shown in the diamond box in the above figure.



From the above process diagram, we can see that the process starts at the circle. The PIR motion sensor which is shown in the rectangular box in the above figure, is placed in a room. It detects the motion in the room. If there is a motion then an image is captured and an alert will be sent to the user. And also it turns on the buzzer. This decision (yes/no) to raise an alert is shown in the diamond box in the above figure.

STEP 3: Domain Model Specification

The domain model specification of home intrusion detection system is as shown below:

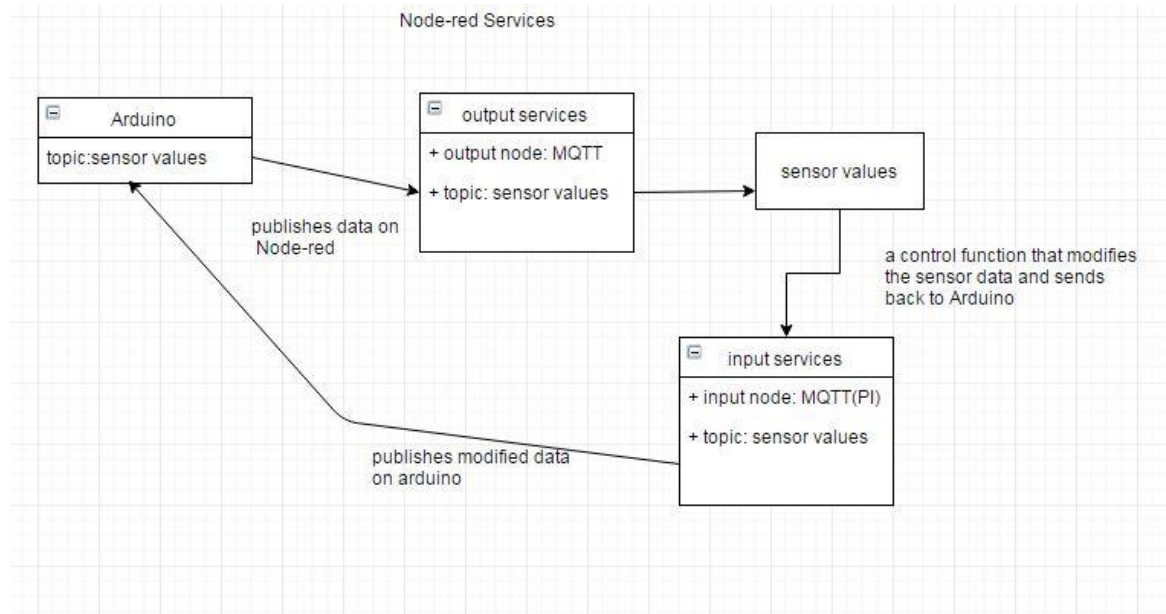


Step 4: Service Specifications:

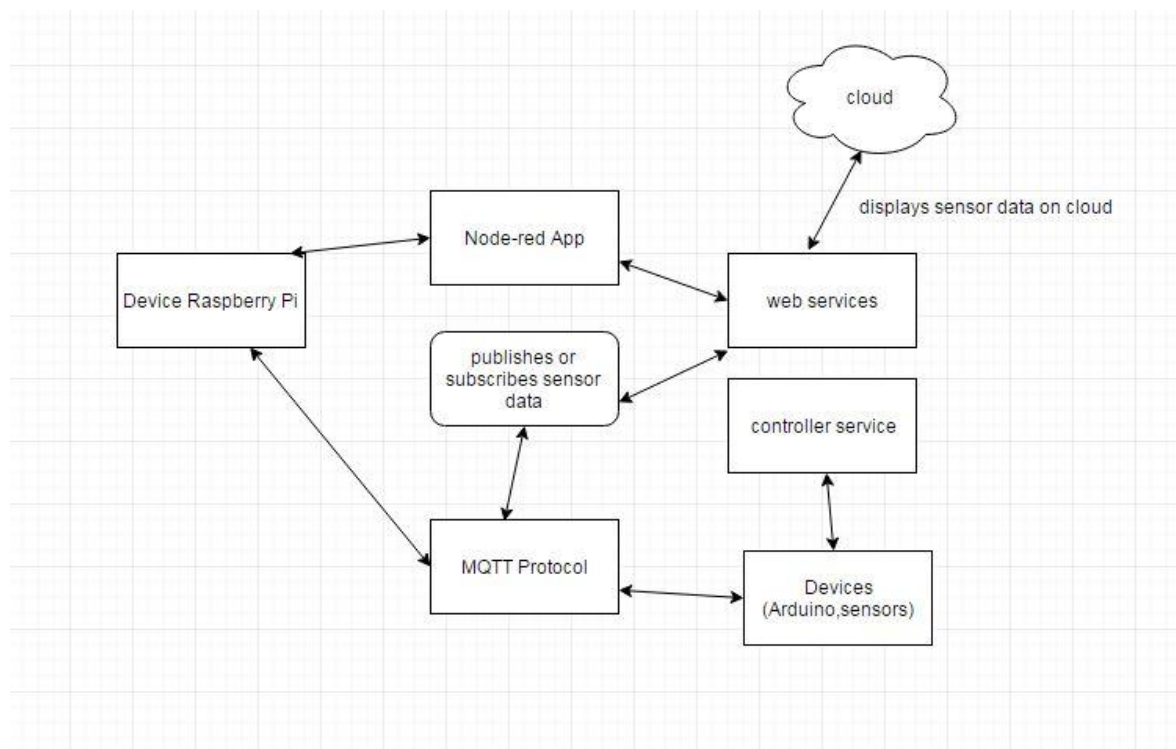
Service specifications defines the services in the IOT system, service types, service inputs/output, service endpoints, service schedules, service productions and service effects.

In my home-intrusion detection system, I am using node-red which provides a browser-based UI for creating flows of events and deploying them to its light-weight runtime. With built in node.js, it can be run at the edge of the network or in the cloud. The node package manager (npm) ecosystem can be used to easily extend the palette of nodes available, *enabling connections to new devices and services*.

Node-RED define flows where either incoming sensor data from 'things' is handled, e.g. stored in databases, or where commands are sent to devices. The service specifications of my home intrusion detection detection system are as shown in the below diagram:



Reference: I have drawn this diagram by taking a reference from the book Internet of things A Hands-On Approach by Arshadeep Bahga and Vijay Madisetti
Block diagram for service specifications for my system:

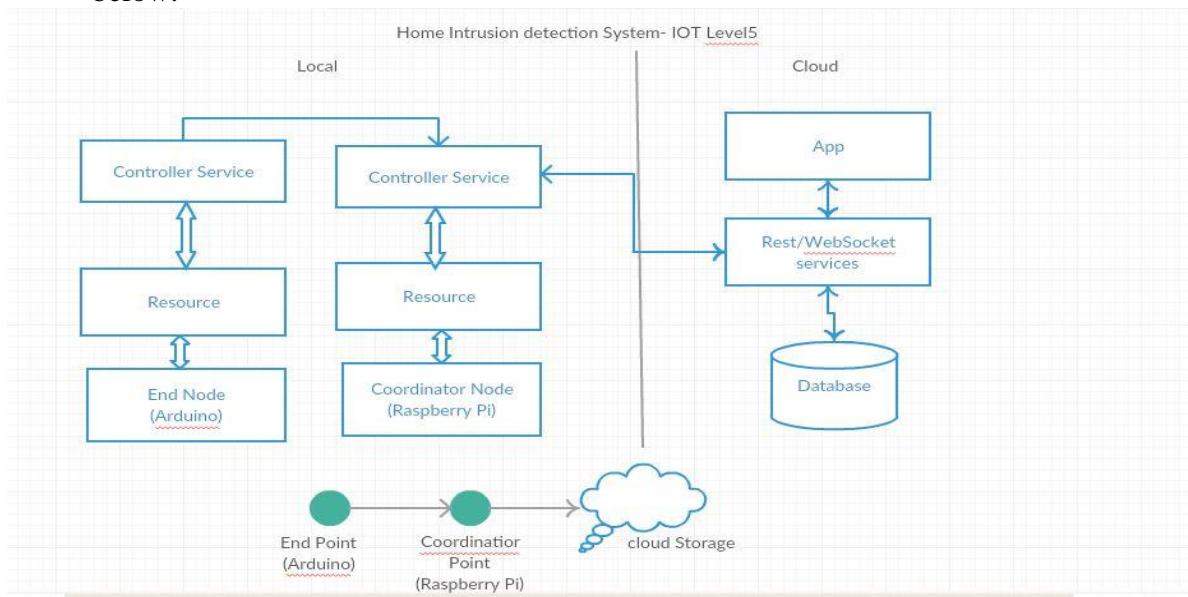


STEP 5: IOT Level Specification

I am going to implement my IOT system that is “**Home intrusion detection system**” in **level 5**. The IOT level 5 system has multiple nodes and one coordinator node. Coordinator node collects the data from the end nodes and sends to the cloud.
In my system for “Home Intrusion Detection”

- The **end node** is “**Arduino**” equipped with PIR and light dependent resistor.

- The **coordinator node** is “**Raspberry Pi**” collects the data from arduino and sends it to the cloud. The analysis is done locally in raspberry pi.
- The deployment design of my home intrusion detection Level-5 system is as shown below:

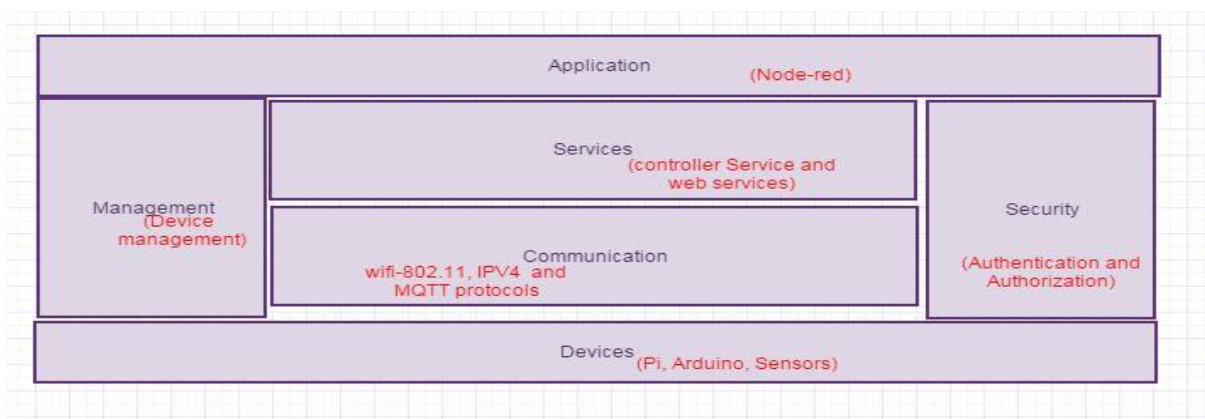


In the above figure we can observe that from our end node Arduino, the data is sent to Raspberry Pi via MQTT. The local analysis is done on the coordinator node which is Pi in my case and the data is sent to cloud via Rest/web socket services.

Step 6: Functional View Specification

The IOT Functional Model identifies groups of functionalities, of which most are grounded in key concepts of the IoT Domain Model. The functional view defines the functions of the IOT systems grouped into various functional groups (FGs). A number of these Functionality Groups (FG) build on each other, following the relations identified in the IoT Domain Model. The Functionality Groups provide the functionalities for interacting with the instances of these concepts or managing the information related to the concepts, e.g. information about Virtual Entities or descriptions of IoT Services. The functionalities of the FGs that manage information use the IoT Information Model as the basis for structuring their information.

The functional groups involved in an functional view are as shown in below figure:

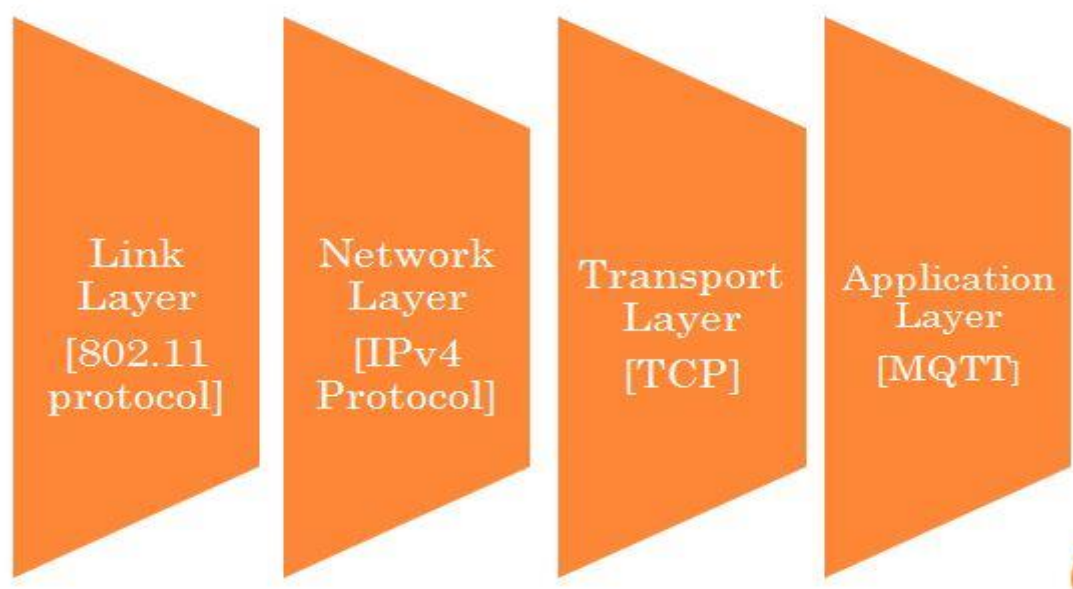


Devices: In my “home intrusion detection system” the computing devices I am going to use are *Raspberry Pi and Arduino*. And the sensors that I am going to use to implement my system are *PIR motion sensor, Laser pointer and the light dependent resistor*.

Communication: The communication block handles the communication for the IOT system. The communication protocols allow devices to exchange data over network. I am going to use following protocols to develop my system:

- **802.11 wifi** protocol to determine how the data is physically sent over the networks physical layer.
- **IPV4 protocol** to transmit the IP datagrams from the source network to the destination network.
- **MQTT** a light weight messaging protocol to publish data to the applications.

PROTOCOLS



Services: The service functional group includes various services involved in the IoT system such as services for device monitoring, device control services and data publishing services. In my system I am going to use native service called the controller service and the web services.

Application: The application functional group provides an interface to the users to control and monitor various aspects of the IOT system. I am going to use *node-red* application to control the sensor values.

Security: To ensure security for my node-red web application, I am going to use the IP address that was generated when my Pi is connected to the network . The IP address is unique and generates a new IP when a new connection is established to the Pi

STEP 7: Operational view Specification

Operation view is very important to address how actual system can be realized by selecting technologies and making them communicate and operate in a comprehensive way. The operational view specifications for home intrusion detection system are as follows:

Devices: The computing devices I am going to use are *Raspberry Pi and Arduino*. And the sensors that I am going to use are *PIR motion sensor, Laser pointer and the light dependent resistor*.

Communication Protocols: Link layer 802.11, network layer-IPV4, application layer MQTT. I have mentioned all these protocols in the above figure.

Services: Controller service that is hosted on my device runs as a native service.

Application: Node-red web application.

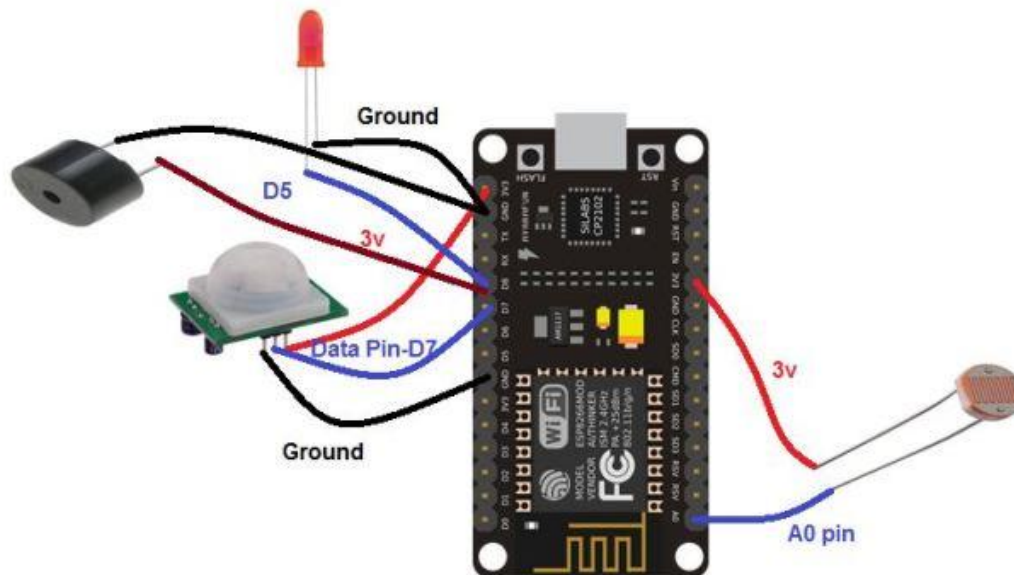
Security: Authentication and authorization.

Management: Raspberry Pi device management.

The mapping of operational view and functional view specifications are shown in the above figure in the functional view block where I have mentioned the operations in red.

STEP 8: Device and Component Integration

In this step we have to integrate the devices and components. The devices and components used in my home intrusion detection system are raspberry Pi, arduino, PIR sensor, laser pointer, light dependent resistor, web cam and a buzzer. The schematic diagram of my home intrusion detection system is as shown in the below figure:

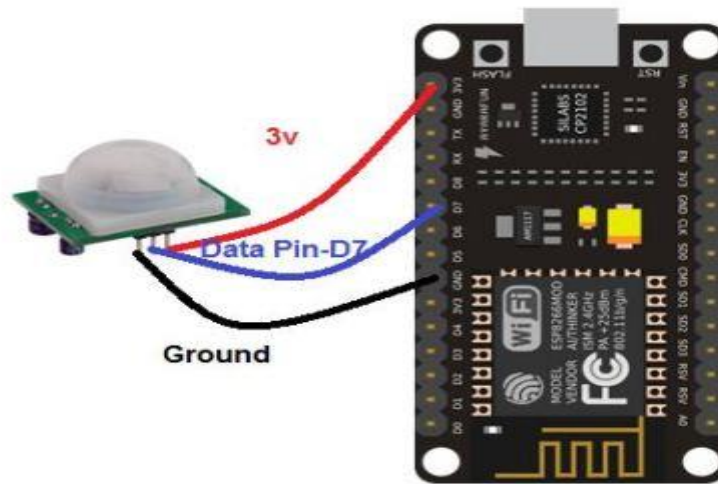


STEP 9. Application Development

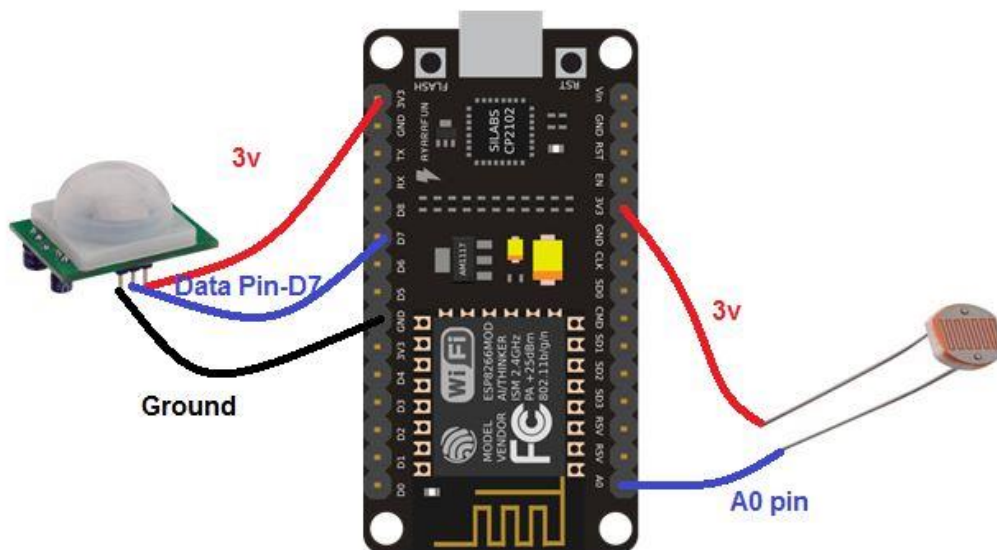
(a) Circuit Connections:

- First the node-mcu is pushed into the bread board.
- To this PIR sensor is connected. PIR sensor has 3 pins:
 - **5v Pin** – This pin is connected to 3v pin of node-mcu
 - **output Pin** – This pin is connected to any data pin of node-mcu. Here in my circuit I connected it to D7 data pin.
 - **Ground pin** – This pin is connected to ground pin of node mcu.

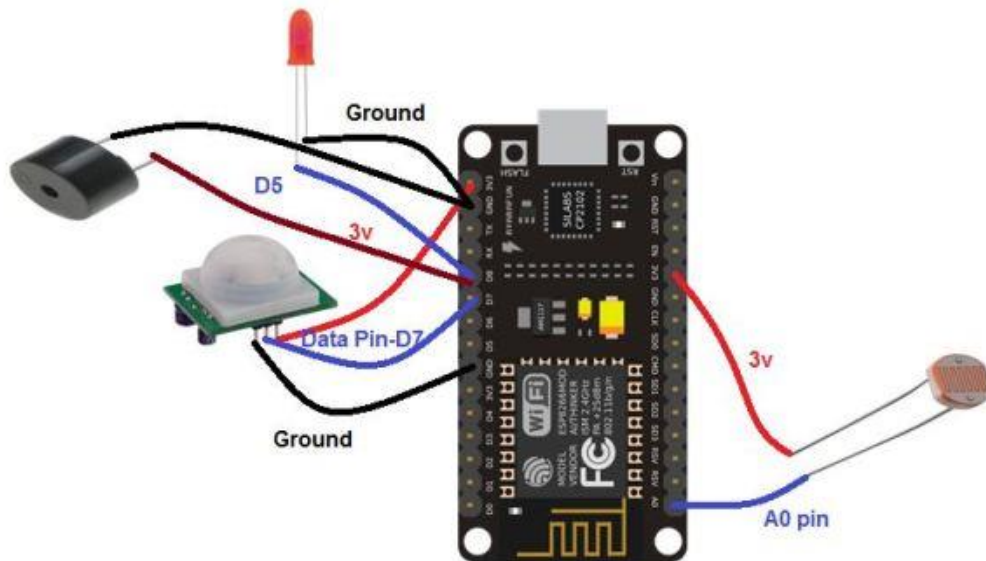
The connections of PIR sensor are as shown below:



Next a light dependent resistor(LDR) is pushed into the breadboard. One leg of LDR is connected to 3v pin of nodemcu and the other leg is connected to analog pin. This is because the readings of LDR are analog and hence to read the analog values we need to connect it to A0 pin of nodemcu. The connection is as follows:

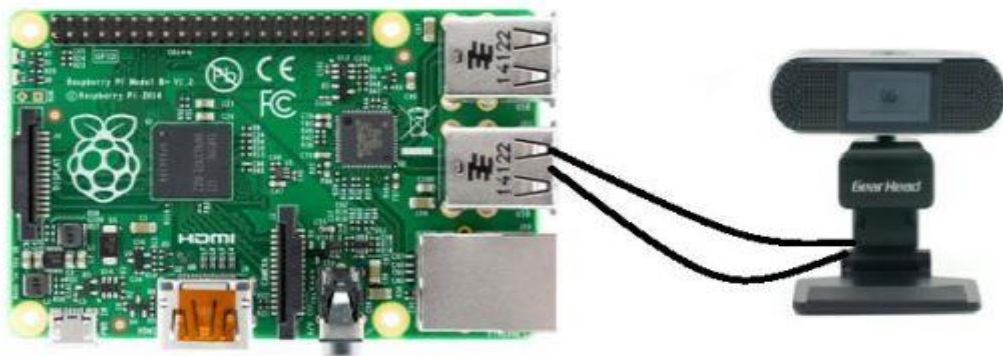


- I have used an LED and a buzzer to alert the user. I have connected one leg of LED and a buzzer to the data pin and the other legs are connected to the ground pin. So in case of intrusion the data pin is set to high and then the buzzer is triggered, LED is turned on alerting the user. The connections for the same is as shown below:

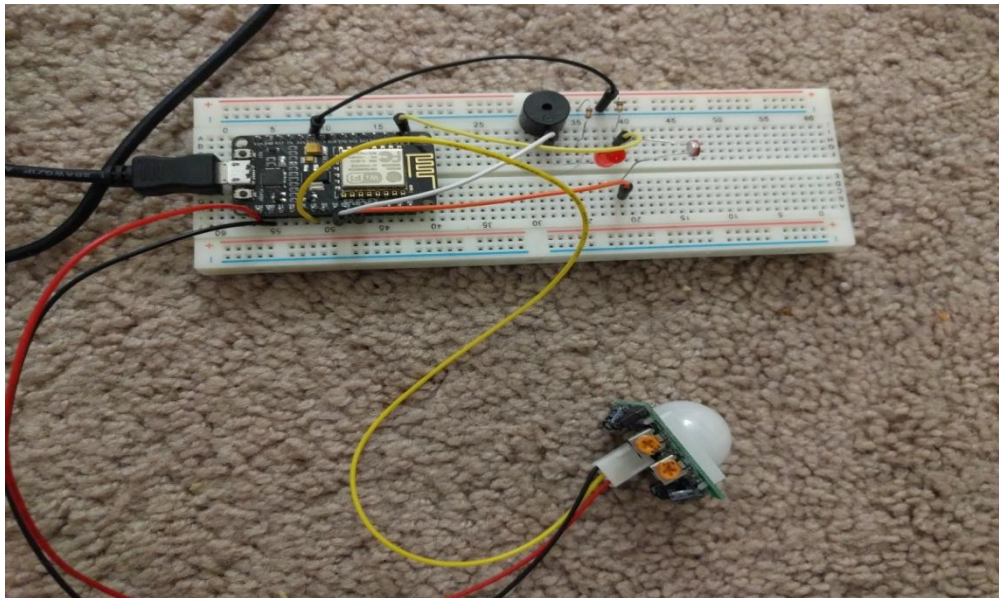
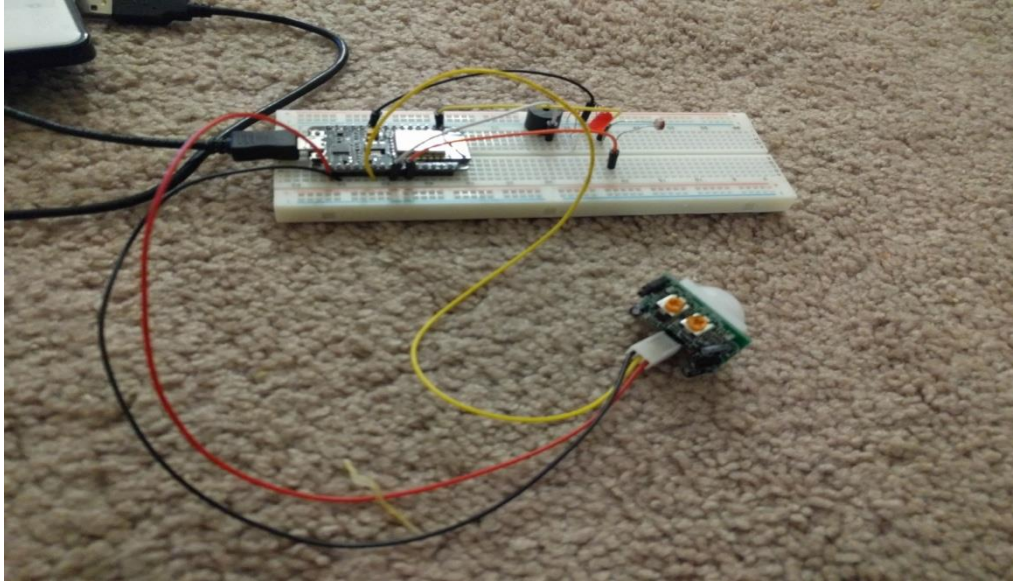


(b)Methodology:

In nodemcu I will be collecting the sensor data and I am going to publish this data onto the pi. If there is a motion PI initiates a command to capture the image and sends this as an alert message to the user. So to accomplish this a uSB camera should be connected to Pi to capture the image. The connection is as shown below:



Circuit Wiring:



We are ready with all the devices and made the circuit connections. Now to implement the above methodology we need to install required software's and libraries to accomplish our goal.

(c) Installation of Required software's and libraries:

- **Nodemcu:** Our goal is to collect sensor values from nodemcu and publish them to the PI. So to accomplish we need to install the following:
 - Install **Arduino IDE** to upload the code into the board. The installation procedure is explained in my previous post
- To publish the data we need **MQTT libraries** on nodemcu. I have already explained the installation procedure of MQTT libraries on nodemcu in my previous post.
 - **Raspberry Pi:** To receive the sensor data from nodemcu, we need to have **MQTT broker and node-red** on Pi. I have already explained the installation procedure of node-red and MQTT on PI in my previous post and the link for it

- If there is a intrusion then PI will be capturing the image of the intruder and sends it to the user. So to capture an image we need to install the following USB cam libraries onto the pi.
 - First install the fswebcam package with the help of following command:
sudo apt-get install fswebcam
 - To check whether the fswebcam installed properly, test it with the following command:
fswebcam test.jpg
 - This command captures the image and stores in your /home/pi location.
- To start the Webcam surveillance we need to install the following libraries on to the Pi.

```
sudo apt-get install motion
mkdir /home/pi/motion
sudo cp /etc/motion/motion.conf /home/pi/motion
mkdir /home/pi/motion/media
```

We are done with all the software and libraries installation on both Arduino and Raspberry Pi. Now we can start writing the code to collect the sensor data and publishing it on Pi. Also for image capturing and sending an email alert

(d) Collecting the sensor values on nodemcu and publishing it on Pi:

- To collect the sensor values of PIR and LDR, first give power supply to the nodemcu with the help of USB cable.
- Write the below code on Arduino IDE and upload it on nodemcu.

Code Reference:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
int ledPin = 14; // choose the pin for the LED
int inputPin = 13;
int LDRValue = 0;
// choose the input pin (for PIR sensor)
int val = 0; // variable for reading the pin status
const char* ssid = "Cisco48681";
const char* password = "villanova";
const char* mqtt_server = "153.104.206.108";
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[150];
int value = 0;
char msg1[150];
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
```

```

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(". ");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  Serial.print("came into callback. The published message from Pi is ");
  Serial.println();
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(14, LOW); // Turn the LED on (Note that LOW is the voltage level
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
  } else {
    digitalWrite(14, HIGH); // Turn the LED off by making the voltage HIGH
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT); // declare LED as output
  pinMode(inputPin, INPUT); // declare sensor as input
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
    }
  }
}

```

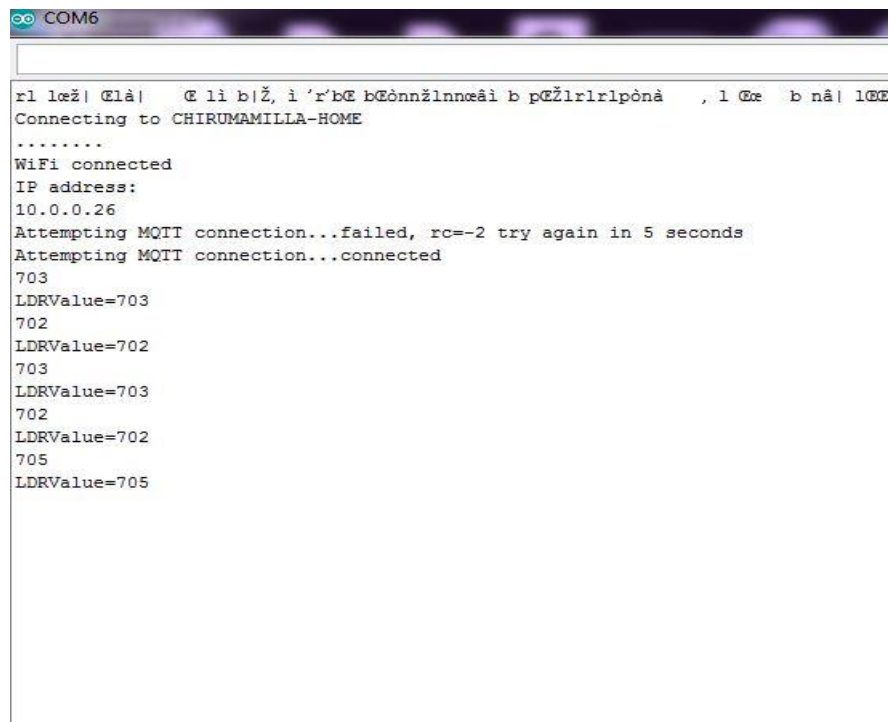


```

// Once connected, publish an announcement...
client.publish("outTopic", "hello world");
// ... and resubscribe
client.subscribe("inTopic");
} else {
Serial.print("failed, rc=");
Serial.print(client.state());
Serial.println(" try again in 5 seconds");
// Wait 5 seconds before retrying
delay(5000);
}
}
}
void loop() {
// put your main code here, to run repeatedly:
if (!client.connected()) {
reconnect();
}
client.loop();
LDRValue=analogRead(A0);
delay(1000);
Serial.println(LDRValue);
String ldr="LDRValue="+String(LDRValue);
Serial.println(ldr);
String disp=ldr;
disp.toCharArray(msg1,disp.length());
snprintf(msg,150,msg1,value);
client.publish("ldrvalue",msg);
val = digitalRead(inputPin); // read input value
if (val == HIGH or LDRValue < 1000) { // check if the input is HIGH
digitalWrite(ledPin, HIGH); // turn LED ON
// we have just turned on
client.publish("outTopic", "Motion detected");
delay(2000);
// We only want to print on the output change, not state
}
else {
digitalWrite(ledPin, LOW); // turn LED OFF
// we have just turned of
//client.publish("outTopic", "Motion ended!");
// We only want to print on the output change, not state
}
}
}

```

We can observe in the serial monitor that it tries to establish MQTT connection and publish the collected sensor data. The output is as shown below:



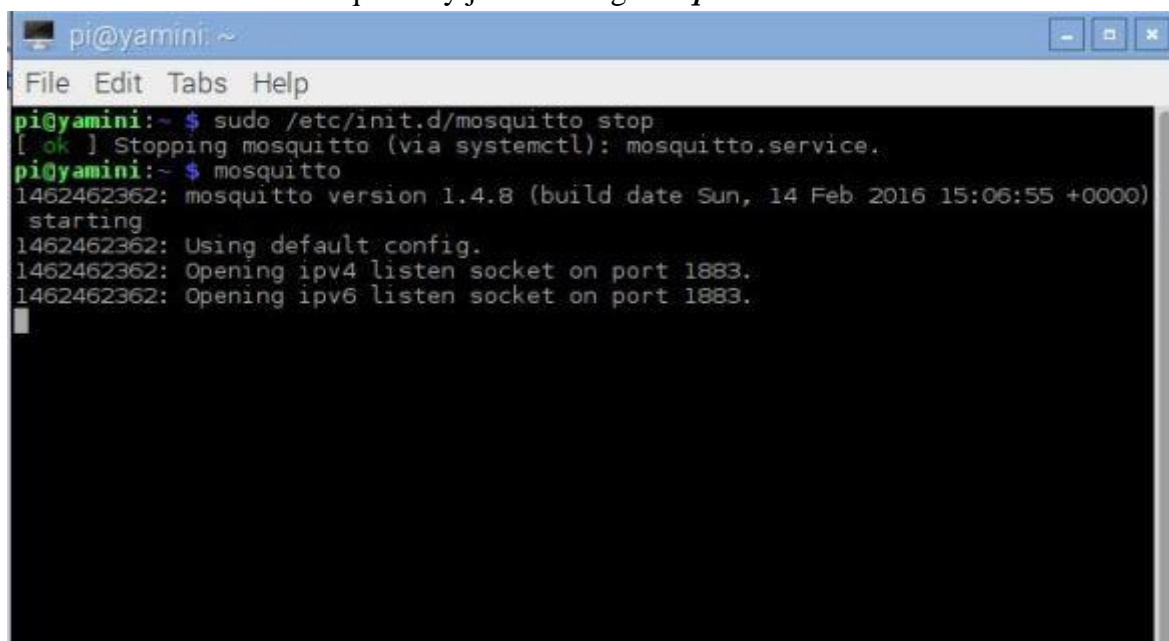
```

COM6
Connecting to CHIRUMAMILLA-HOME
.....
WiFi connected
IP address:
10.0.0.26
Attempting MQTT connection...failed, rc=-2 try again in 5 seconds
Attempting MQTT connection...connected
703
LDRValue=703
702
LDRValue=702
703
LDRValue=703
702
LDRValue=702
705
LDRValue=705

```

(e) Pi collecting the Sensor data from nodemcu:

- Open LXTerminal and start the node-red by executing the command *node-red-start*.
- Open another LXTerminal and first stop the MQTT service with the below command *sudo /etc/init.d/mosquitto stop*
- Now restart the mosquitto by just entering *mosquitto* command on LXTerminal.



```

pi@yamini: ~
File Edit Tabs Help
pi@yamini:~ $ sudo /etc/init.d/mosquitto stop
[ ok ] Stopping mosquitto (via systemctl): mosquitto.service.
pi@yamini:~ $ mosquitto
1462462362: mosquitto version 1.4.8 (build date Sun, 14 Feb 2016 15:06:55 +0000)
starting
1462462362: Using default config.
1462462362: Opening ipv4 listen socket on port 1883.
1462462362: Opening ipv6 listen socket on port 1883.

```

```

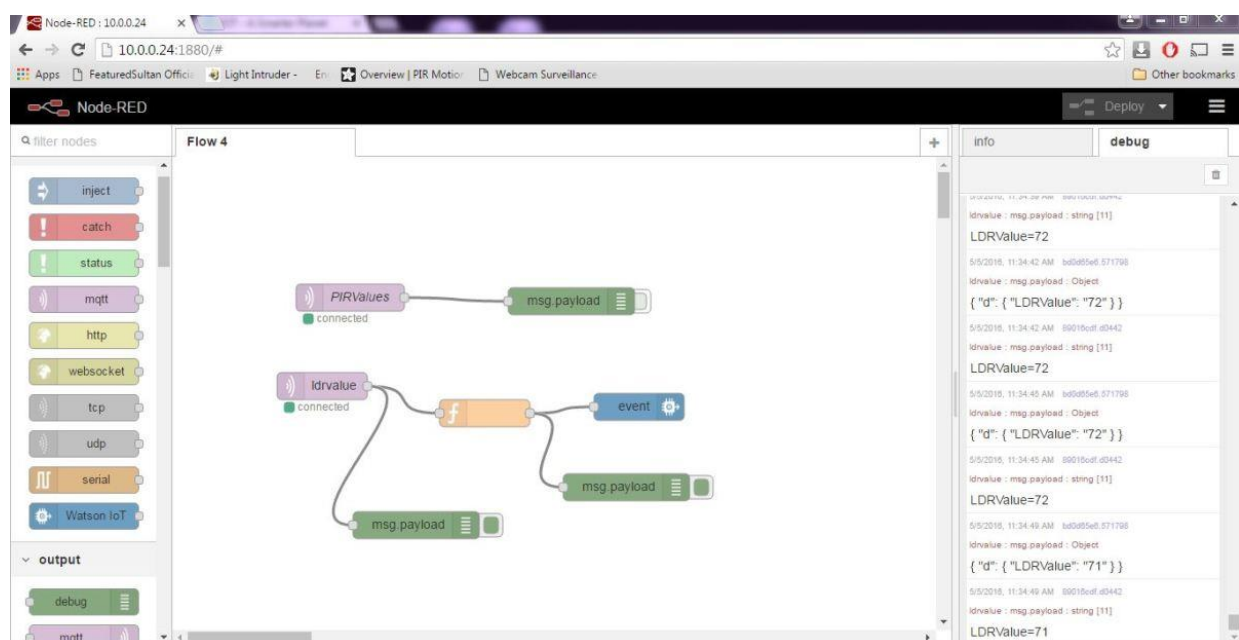
Node-RED console
File Edit Tabs Help
pi@yamini:~ $ node-red-stop
Stop Node-RED
Use node-red-start to start Node-RED again
pi@yamini:~ $ node-red-start
Start Node-RED
Once Node-RED has started, point a browser at http://10.0.0.24:1880
On Pi Node-RED works better with the Iceweasel browser
Use node-red-stop to stop Node-RED
Use node-red-start to start Node-RED again
Use sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use sudo systemctl disable nodered.service to disable autostart on boot
To find more nodes and example flows - go to http://flows.nodered.org
You may also need to install and upgrade npm
sudo apt-get install npm
sudo npm i -g npm@2.x

```

(f) Interaction between nodemcu and Pi:

Start the flows in the node-red. I have already explained in my previous post about the nodes and how to create them and wire them together.

Wiring and node-diagrams of my system are as shown below:

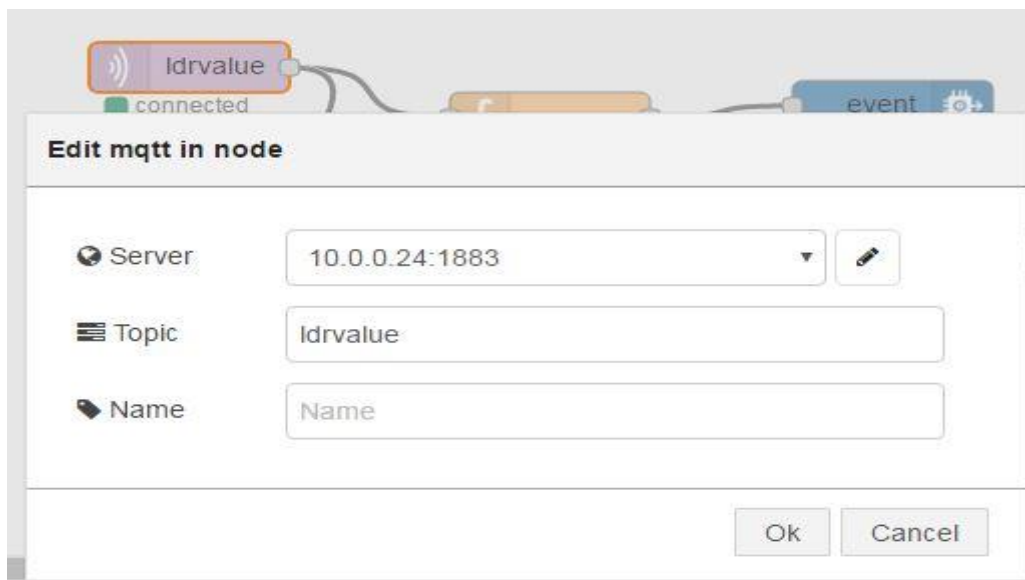


As I am using PIR sensor and Light dependent resistor to monitor the intrusions, I need to publish both PIR and LDR values on to the Pi.

For collecting PIR values – Select a MQTT input node and do the configuration as shown below. To this node connect the timestamp node.



For collecting the LDR values, select another MQTT node and do the below configuration:



From nodemcu, we are receiving the values ***LDRvalue: 789***, we can modify it and send back only the value say 789 to Arduino. I am just testing to see whether Pi can communicate with arduino or not. To accomplish this I have created a function node and written the following code in it. So when I deployed and executed it, we can see arduino is receiving the values from Pi. Thus, two-way communication is successfully established. Code in function node is as follows

***msg.payload={ 'd': { 'LDRValue': msg.payload.replace("LDRValue=", "") } }
return msg;***

The screenshot shows the Node-RED interface. On the left, a flow diagram includes an 'ldrvalue' input node, a function node (orange box with 'f'), a 'msg.payload' output node, and a 'timestamp' node. The function node is selected, and its configuration is shown in the 'Edit function node' panel on the right. The panel has a 'Name' field, a 'Function' text area containing the code:

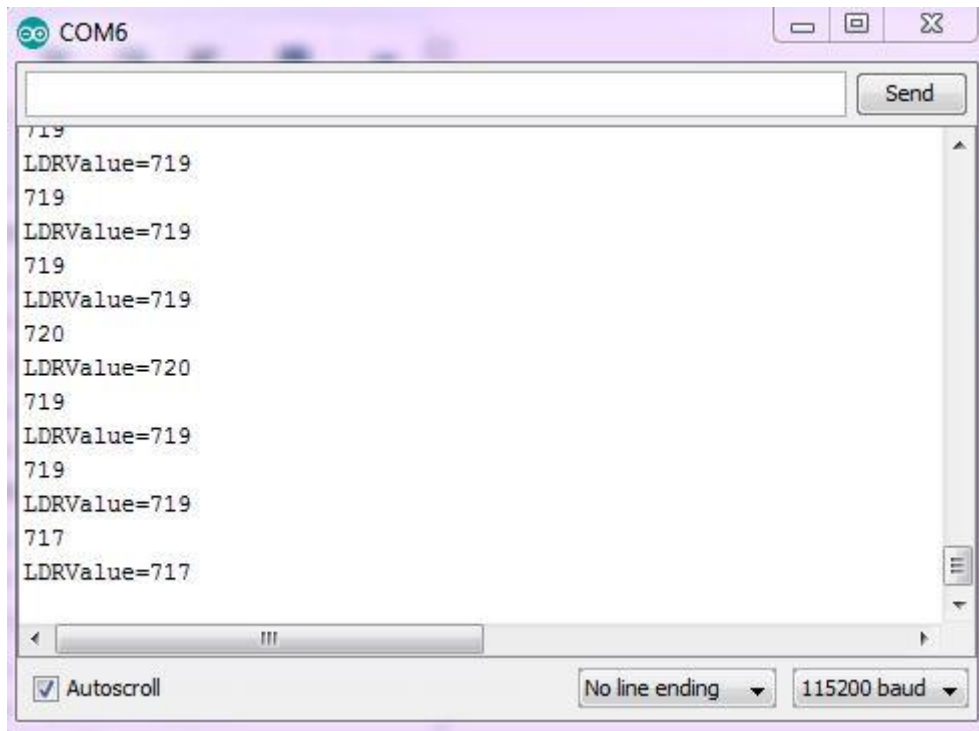

```
i 1 msg.payload={ 'd': { 'LDRValue': msg.payload.replace("LDRValue=", "") } }
2 return msg;
```

 and an 'Outputs' dropdown set to '1'. Below the text area is a yellow tip: 'See the Info tab for help writing functions.' At the bottom right of the panel are 'Ok' and 'Cancel' buttons.

Node-red debug column:

The screenshot shows the 'debug' tab of the Node-RED debug console. It displays a list of messages. Each message entry includes a timestamp, a unique ID, and the message content. The messages are as follows:

Timestamp	ID	Message
5/5/2016, 11:55:45 AM	bd0d85e6.571798	ldrvalue : msg.payload : string [11] LDRValue=71
5/5/2016, 11:55:45 AM	89016cdf.d0442	ldrvalue : msg.payload : Object { "d": { "LDRValue": "71" } }
5/5/2016, 11:55:49 AM	bd0d85e6.571798	ldrvalue : msg.payload : string [11] LDRValue=71
5/5/2016, 11:55:49 AM	89016cdf.d0442	ldrvalue : msg.payload : Object { "d": { "LDRValue": "71" } }
5/5/2016, 11:55:52 AM	bd0d85e6.571798	ldrvalue : msg.payload : string [11] LDRValue=71
5/5/2016, 11:55:52 AM	89016cdf.d0442	ldrvalue : msg.payload : Object { "d": { "LDRValue": "71" } }
5/5/2016, 11:55:55 AM	bd0d85e6.571798	ldrvalue : msg.payload : string [11] LDRValue=71



Two-way communication is successfully established.

(f)Experiments and Results:

If there is a intrusion, Pi should capture the image of the intruder and send the image to the user. To accomplish this we need to execute the below code in the Pi:

- To read the values from node-red on Pi, we need to install the paho-library onto the pi.
- The SMTP library helps to send the email to the user from raspberry Pi. So in the code we need to import this library.

Code

```
import paho.mqtt.client as mqtt
import time
import smtplib
from subprocess import call
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email.Utils import COMMASPACE, formatdate
from email import Encoders
import os
USERNAME = "iotexperimentz@gmail.com"
PASSWORD = "XXXXXXXXXX"
def sendMail(to, subject, text, files=[]):
    assert type(to)==list
    assert type(files)==list
    msg = MIMEMultipart()
    msg['From'] = USERNAME
```

```

msg['To'] = COMMASPACE.join(to)
msg['Date'] = formatdate(localtime=True)
msg['Subject'] = subject
msg.attach( MIMEText(text) )
for file in files:
part = MIMEBase('application', "octet-stream")
part.set_payload( open(file,"rb").read() )
Encoders.encode_base64(part)
part.add_header('Content-Disposition', 'attachment; filename="%s"'
% os.path.basename(file))
msg.attach(part)
server = smtplib.SMTP('smtp.gmail.com:587')
server.ehlo_or_helo_if_needed()
server.starttls()
server.ehlo_or_helo_if_needed()
server.login(USERNAME,PASSWORD)
server.sendmail(USERNAME, to, msg.as_string())
server.quit()
# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, rc):
print("Connected with result code "+str(rc))
# Subscribing in on_connect() means that if we lose the connection and
# reconnect then subscriptions will be renewed.
client.subscribe("outTopic/#")

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
print(msg.topic+" "+str(msg.payload))
call(["fswebcam", "-d", "/dev/video0", "-r", "1280*720", "-no-banner", "test1.jpg"])
time.sleep(2)
sendMail( ["yamini.tella@gmail.com"],
"Detected Intrusion",
"Someone broke into your room, picture attached",
["/home/pi/test1.jpg"] )
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect("10.0.0.24", 1883, 60)
# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a
# manual interface.
client.loop_forever()

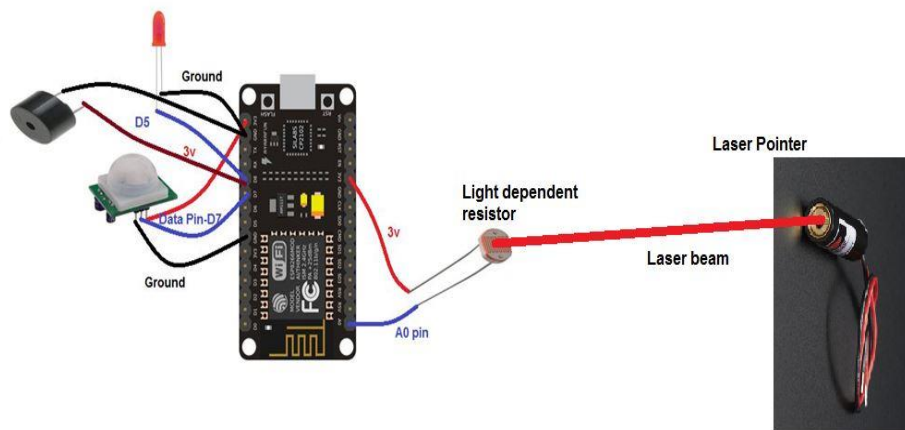
```

By executing the above code, if there is an intrusion then an image will be captured and send as an email alert to the user.

Experiments and Results:

- I will be pointing a laser beam on to the Light dependent resistor.
- I will note down the light level when laser beam is pointed. I will set this value as a threshold value.
- Now if there is an obstacle in the laser beam the light level gradually falls down.i.e. the value will be less than threshold value. This means there is an intrusion and this message is sent to Pi to capture the image of the intruder and alert the user.

Work Flow of this scenario is as shown below:



In case of intrusion:

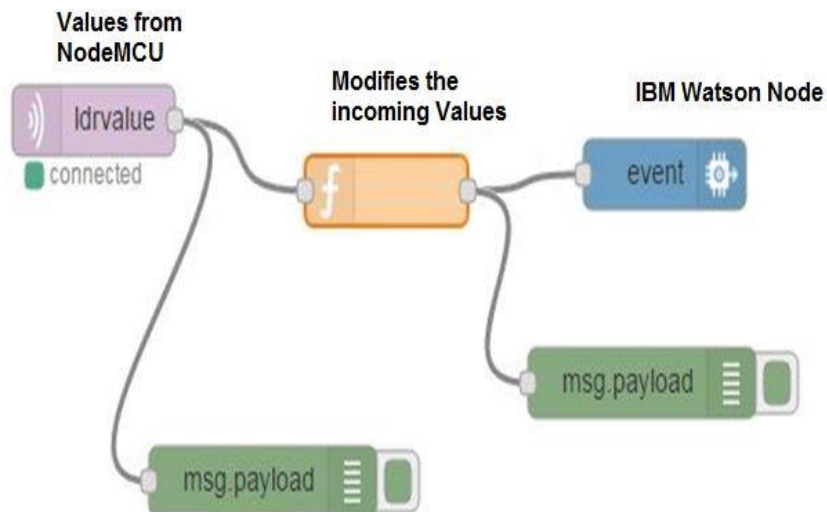


- We can observe that in the above figure if someone intrudes, there will be a breach in the laser beam and the light level will be less than threshold value.
- This message will be sent to Pi
- Pi receives the message and captures the image of the intruder and send the alert email to the user.

Graphical representation of LDR values in IBM Watson cloud:

- I have used IBM watson cloud to graphically represent my data.

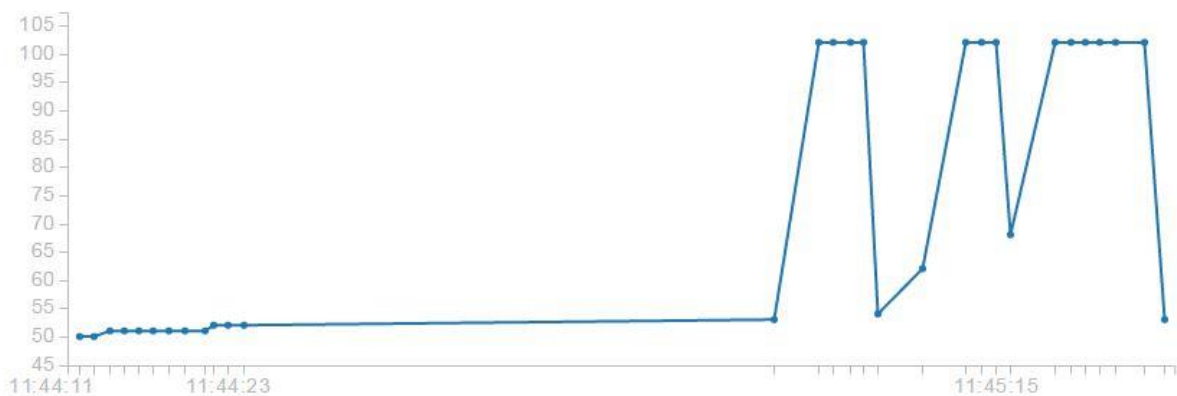
- To send the values to the cloud I have used Watson IOT node in Node-red application.
- Node-red has an inbuilt node for IBM watson. Select that node and connect it to the LDR MQTT node as shown below:



- When we click on the event node, it will redirect to the watson cloud where we can see the graphical representation of our LDR values. It is as hown below:

afea8726.4d4e08

event.LDRValue



Event	Datapoint	Value	Time Received
event	LDRValue	73	May 2, 2016 12:24:54 PM

Pi captures the image of the intruder and sends it as an Email alert to the user and it is as shown in the below figure:

Captured Image:



Result

Thus an application for home intrusion detection web application was developed successfully

Ex.No.6 Smart parking application using python and Django for web application

Aim

To Develop an application for Smart parking application using python and Django for web application

Procedure

Introduction

we are going to Create a Vehicle Parking Management in Python. This project helps maintain the details of the vehicle owner, number, type, date, and the amount of time the vehicle is parked in the area. Accordingly, the bill generates for the particular vehicle parked in the area.

1. Import Function and Initializing the variables

```
#Import Time
import time

Vehicle_Number=['XXXX-XX-XXXX']
Vehicle_Type=['Bike']
vehicle_Name=['Intruder']
Owner_Name=['Unknown']
Date=['22-22-3636']
Time=['22:22:22']
bikes=100
cars=250
bicycles=78
```

In this code block, we are importing the time module to implement its methods and function in the project. We have initialized the variables vehicle number, vehicle type, vehicle name, owner name date, and time to some default value. As well as bikes, cars, and bicycles with some initial value.

2. Create a while loop block to display the options in Vehicle Parking Management Project

```
def main():
global bikes,cars,bicycles
try:
while True:
print("-----")
print("\t\tParking Management System")
print("-----")
print("1.Vehicle Entry")
print("2.Remove Entry" )
```

```

print("3.View Parked Vehicle ")
print("4.View Left Parking Space ")
print("5.Amount Details ")
print("6.Bill")
print("7.Close Programme ")
print("+-----+")
ch=int(input("\tSelect option:"))

```

In this code block, we have initialized the bikes, cars, and bicycles as global variables. They are accessible through the entire main block. Here we are providing the options to choose the service options from the list, for the vehicle parking management system.

Code for vehicle number entry

```

if ch==1:
no=True
while no==True:
    Vno=input("\tEnter vehicle number (XXXX-XX-XXXX) - ").upper()
if Vno=="":
print("##### Enter Vehicle No. #####")
elif Vno in Vehicle_Number:
print("##### Vehicle Number Already Exists")
elif len(Vno)==12:
no=not True
    Vehicle_Number.append(Vno)
else:
print("##### Enter Valid Vehicle Number #####")

```

Ch is for choice, Once we select the ch option as 1 which is for vehicle entry number, then we provide the while loop. while the number(no is True). We will store the vehicle number in Vno. If the vno is empty i.e vno=="". The user asks to enter the vehicle number, else If the vno entered is already present in the vehicle number then it prints the vehicle number already exists. Else if len(vno)==12, It will ask to append the info to the vehicle number variable.

Code to enter the vehicle type

```

typee=True
while typee==True:
    Vtype=str(input("\tEnter vehicle type(Bicycle=A/Bike=B/Car=C):")).lower()
if Vtype=="":
print("##### Enter Vehicle Type #####")
elif Vtype=="a":
    Vehicle_Type.append("Bicycle")
bicycles-=1
typee=not True
elif Vtype=="b":
    Vehicle_Type.append("Bike")
bikes-=1
typee=not True
elif Vtype=="c":

```

```

    Vehicle_Type.append("Car")
cars-=1
typee=not True
else:
print("##### Please Enter Valid Option #####")

```

Here we have to initialize the typee variable to true. While the condition is True, the system asks to enter the vehicle type i.e a,b, or c which will accept the input in the lower case. Here A is for bicycle, B is for Bike and C is for Car. Any vehicle type you enter is stored in the variable Vtype. If the Vtype==""(empty). It will ask to enter the vehicle type. According to the type of variable you enter the vehicle type will be stored in the variable and typee variable is set to not True.

Code to enter the vehicle name

```

name=True
while name==True:
vname=input("\nEnter vehicle name - ")
if vname=="":
print("#####Please Enter Vehicle Name #####")
else:
    vehicle_Name.append(vname)
name=not True

```

Here we have set the name== True. While the name == True i.e until we enter the name.vname store the value i.e. vehicle name. if the vname is empty system asks to enter the vehicle name, else it will store the name using the append function to the vehicle name variable The name variable is initialized to not True.

Code to enter the owners name

```

o=True
while o==True:
    OName=input("\nEnter owner name - ")
if OName=="":
print("##### Please Enter Owner Name #####")
else:
    Owner_Name.append(OName)
o=not True

```

O is initialized to True. While the condition satisfies the Owner's name is stored in the OName variable. If the OName is empty it system asks to enter the owner name else it will store the Owner name in the owner name variable. O is now initialized to Not True.

Code enter the date and time

```

d=True
while d==True:
date=input("\nEnter Date (DD-MM-YYYY) - ")
if date=="":
print("##### Enter Date #####")
elif len(date)!=10:

```

```

print("##### Enter Valid Date #####")
else:
    Date.append(date)
    d=not True
t=True
while t==True:
    time=input("\tEnter Time (HH:MM:SS) - ")
    if t=="":
        print("##### Enter Time #####")
    elif len(time)!=8:
        print("##### Please Enter Valid Time #####")
    else:
        Time.append(time)
        t=not True
print("\n.....Record")

```

Similarly, we have to create a while loop to enter the date and time initializing d and t to 0. Date variable stores the date and the time-variable stores time. The date and time variable checks the condition and accordingly execute further.

Code to remove the entry from the register

```

elif ch==2:
    no=True
    while no==True:
        Vno=input("\tEnter vehicle number to Delete(XXXX-XX-XXXX) - ").upper()
        if Vno=="":
            print("##### Enter Vehicle No. #####")
        elif len(Vno)==12:
            if Vno in Vehicle_Number:
                i=Vehicle_Number.index(Vno)
                Vehicle_Number.pop(i)
                Vehicle_Type.pop(i)
                vehicle_Name.pop(i)
                Owner_Name.pop(i)
            Date.pop(i)
            Time.pop(i)
            no=not True
            print("\n.....
            Removed Sucessfully.....")
        elif Vno not in Vehicle_Number:
            print("##### No Such Entry #####")
        else:
            print("Error")
        else:
            print("##### Enter Valid Vehicle Number #####")

```

In this code block, we are writing the code to remove the particular entry of a vehicle from the database. Users have entered a valid vehicle number to Create a Vehicle Parking Management in Python. If the vehicle number is present in our database and if the length of the vehicle number is 12 then it uses a pop function to remove the particular entry. Else if the

vehicle number is not present in the database it will print “No such Entry” and asks to enter a valid vehicle number.

Code to display the vehicles present in the parking area

```
elif ch==3:
count=0
print("-----")
print("\t\t\tParked Vehicle")
print("-----")
print("Vehicle No.\tVehicle Type\t\tVehicle Name\t\tOwner Name\t\tDate\t\tTime")
print("-----")
for i in range(len(Vehicle_Number)):
count+=1
print(Vehicle_Number[i],"\t",Vehicle_Type[i],"\t\t\t",vehicle_Name[i],"\t\t\t",Owner_Name[i],"\t\t\t",Date[i],"\t\t\t",Time[i])
print("-----")
print("----- Total Records - ",count,"-----")
print("-----")
```

Here ch==3 is for displaying the parked vehicles in the parking area. For this, we have to use the for loop function. It counts the length of the vehicle number. This will display the whole information of the vehicles.

Code for spaces left in the parking area

```
elif ch==4:
print("-----")
print("\t\t\tSpaces Left For Parking")
print("-----")
print("\tSpaces Available for Bicycle - ",bicycles)
print("\tSpaces Available for Bike - ",bikes)
print("\tSpaces Available for Car - ",cars)
print("-----")
```

This block of code displays the spaces left for parking in the parking area.

Code for displaying the parking rate

```
elif ch==5:
print("-----")
print("\t\t\tParking Rate")
print("-----")
print("*1.Bicycle\t\tRs20 / Hour")
```



```

print("*2.Bike      Rs40/ Hour")
print("*3.Car      Rs60/ Hour")
print("-----")

```

It displays the parking rate of different types of vehicles.

Code to generate bills for different types of vehicles parked

```

elif ch==6:
print("..... Generating Bill
.....")
no=True
while no==True:
    Vno=input("\tEnter vehicle number to Delete(XXXX-XX-XXXX) - ").upper()
    if Vno=="":
        print("##### Enter Vehicle No. #####")
    elif len(Vno)==12:
        if Vno in Vehicle_Number:
            i=Vehicle_Number.index(Vno)
            no=not True
        elif Vno not in Vehicle_Number:
            print("##### No Such Entry #####")
        else:
            print("Error")
        else:
            print("##### Enter Valid Vehicle Number #####")
            print("\tVehicle Check in time - ",Time[i])
            print("\tVehicle Check in Date - ",Date[i])
            print("\tVehicle Type - ",Vehicle_Type[i])
            inp=True
            amt=0
            while inp==True:
                hr=input("\tEnter No. of Hours Vehicle Parked - ").lower()
                if hr=="":
                    print("##### Please Enter Hours #####")
                elif int(hr)==0 and Vehicle_Type[i]=="Bicycle":
                    amt=20
                    inp=not True
                elif int(hr)==0 and Vehicle_Type[i]=="Bike":
                    amt=40
                    inp=not True
                elif int(hr)==0 and Vehicle_Type[i]=="Car":
                    amt=60
                    inp=not True
                elif int(hr)>=1:
                    if Vehicle_Type[i]=="Bicycle":
                        amt=int(hr)*int(20)
                        inp=not True
                    elif Vehicle_Type[i]=="Bike":
                        amt=int(hr)*int(40)
                        inp=not True

```

```

elif Vehicle_Type[i]=="Car":
amt=int(hr)*int(60)
inp=not True
print("\t Parking Charge - ",amt)
ac=18/100*int(amt)
print("\tAdd. charge 18 % - ",ac)
print("\tTotal Charge - ",int(amt)+int(ac))
print(".....Thank you for using our
service.....")
a=input("\tPress Any Key to Proceed - ")
elif ch==7:
print(".....Thank you for using our
service.....")
print("*****(: Bye Bye :)*")
break
quit

```

The billing section generates the bill for the vehicle parked. We have to enter the correct vehicle number and check the length of the vehicle number and the vehicle number present in the database. After this check the time and the type of vehicle. Depending upon the vehicle type the system calculates the charges. The last choice `ch==7` is to come out of the service options and quit the program.

Output

```

1.Vehicle Entry
2.Remove Entry
3.View Parked Vehicle
4.View Left Parking Space
5.Amount Details
6.Bill
7.Close Programme
+-----+
Select option:1
Enter vehicle number (XXXX-XX-XXXX) - KH12 ST 3646
Enter vehicle type(Bicycle=A/Bike=B/Car=C):C
Enter vehicle name - Altos
Enter owner name - Ravi
Enter Date (DD-MM-YYYY) - 12 09 2017
Enter Time (HH:MM:SS) - 7 12 60
##### Please Enter Valid Date #####
Enter Time (HH:MM:SS) - 07 12 60

.....Record detail saved.....

```

Result

Thus an application for Smart parking application using python and Django for web was developed successfully