**TITLE: Home Automation**

**PROBLEM DEFINITION**

Customer satisfaction and convenience is ultimate goal of any business venture. Here we aimed at developing a system which can fulfill these above-mentioned goals in cost-effective manner and reasonable response time. The system works in two modes : auto and manual. Auto mode monitors whether someone entered room and then turns on or turns off light and fan. For manual mode, user gives input for turning on/off of light and fan. System is deployed locally but can be accessed remotely which is striking feature of system. For demonstrating green and yellow LED’s were used and for all test cases output is verified.

**CONTENTS**

1.SYSTEM DESIGN METHODOLOGY

1.1 Requirement Specification

1.2 Process model specification

1.3 Domain model specification

1.4 Information model specification

1.5 Service level specification

1.6 IoT level specification

1.7 Functional view specification

1.8 Operational view specification

1.9 Device and component integration

1.10 Application Development

2.SOURCE CODE

3.DEPLOYMENT

4.RESULTS

5.CONCLUSION

**1. SYSTEM DESIGN METHODOLOGY**

1.1 Requirement specification

This is first step in IoT design methodology. This step is to define purpose and requirements of the system. In this step purpose, behavior and requirements are captured.

For specified system following requirements are captured

1) Purpose: A home automation system that allows controlling of the lights in home remotely through web application

2) Behavior: The home automation system should behave in two modes: auto and manual.

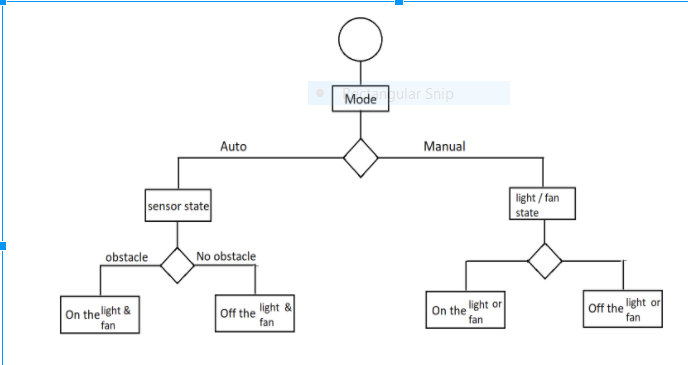
3)System management requirements: The system should provide remote monitoring and control functions.

4)Data Analysis requirements: System should perform local analysis of data.

5)Application deployment requirements: The application should be deployed locally but accessible remotely.

1.2 Process model specification

In this step **use cases** of IoT system are formally described using requirements obtained from previous step.



1.3 Domain model specification

In this step **entities, objects and classes** are identified.

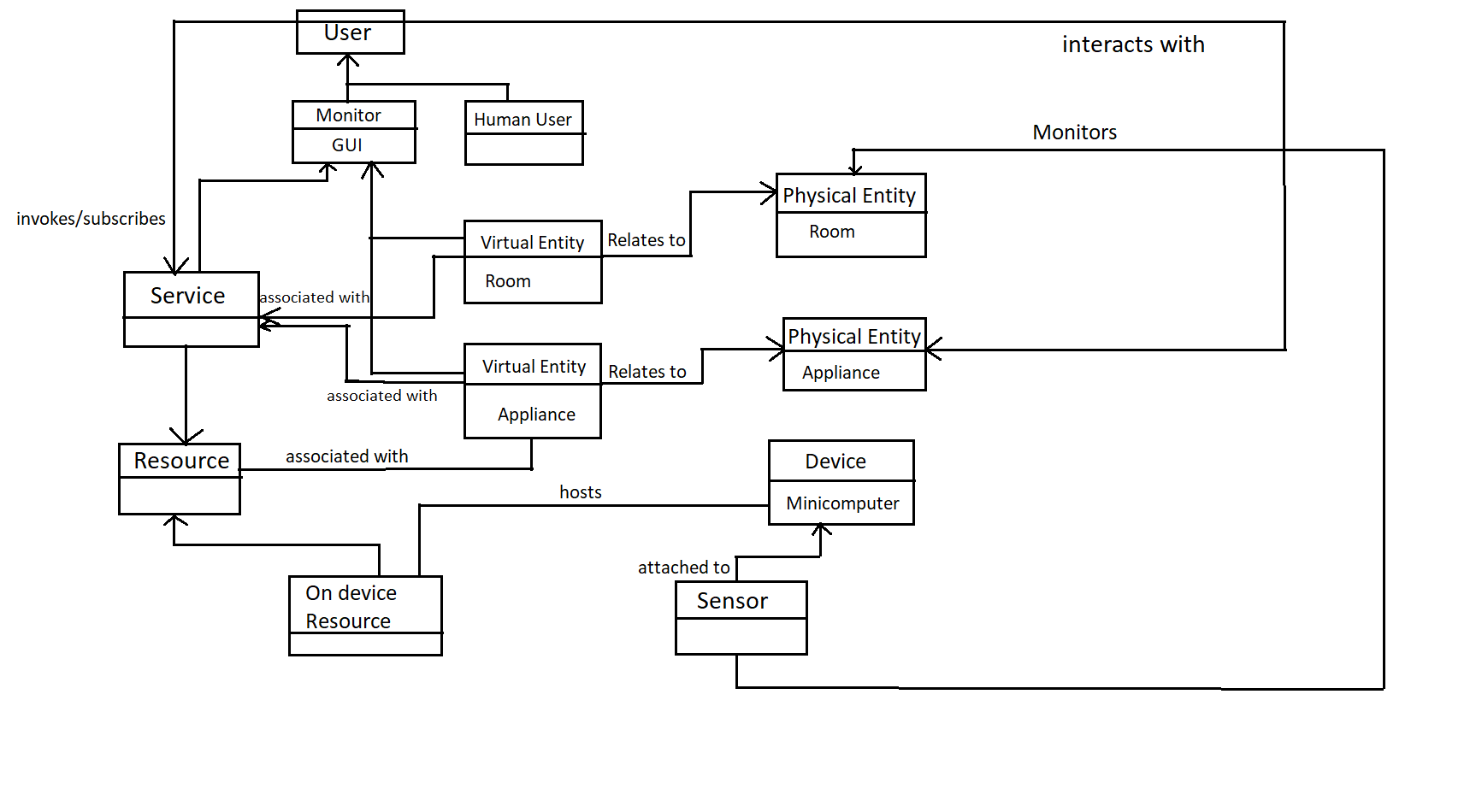
a. Physical entity: Room, light appliance

b. Virtual entity: Room, light appliance

c. Device: IR sensor

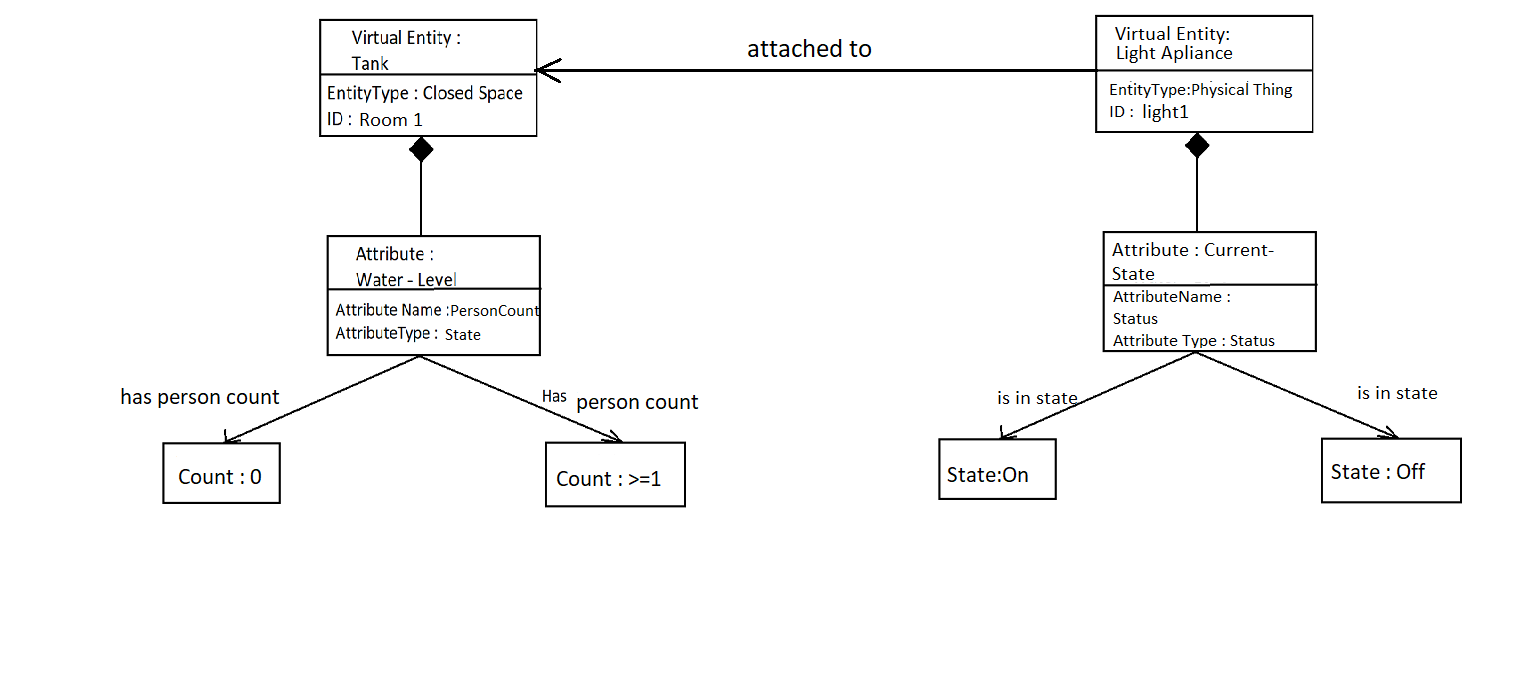
d. Resource: python script, circuit wires

e. Service: mode selection, actuation



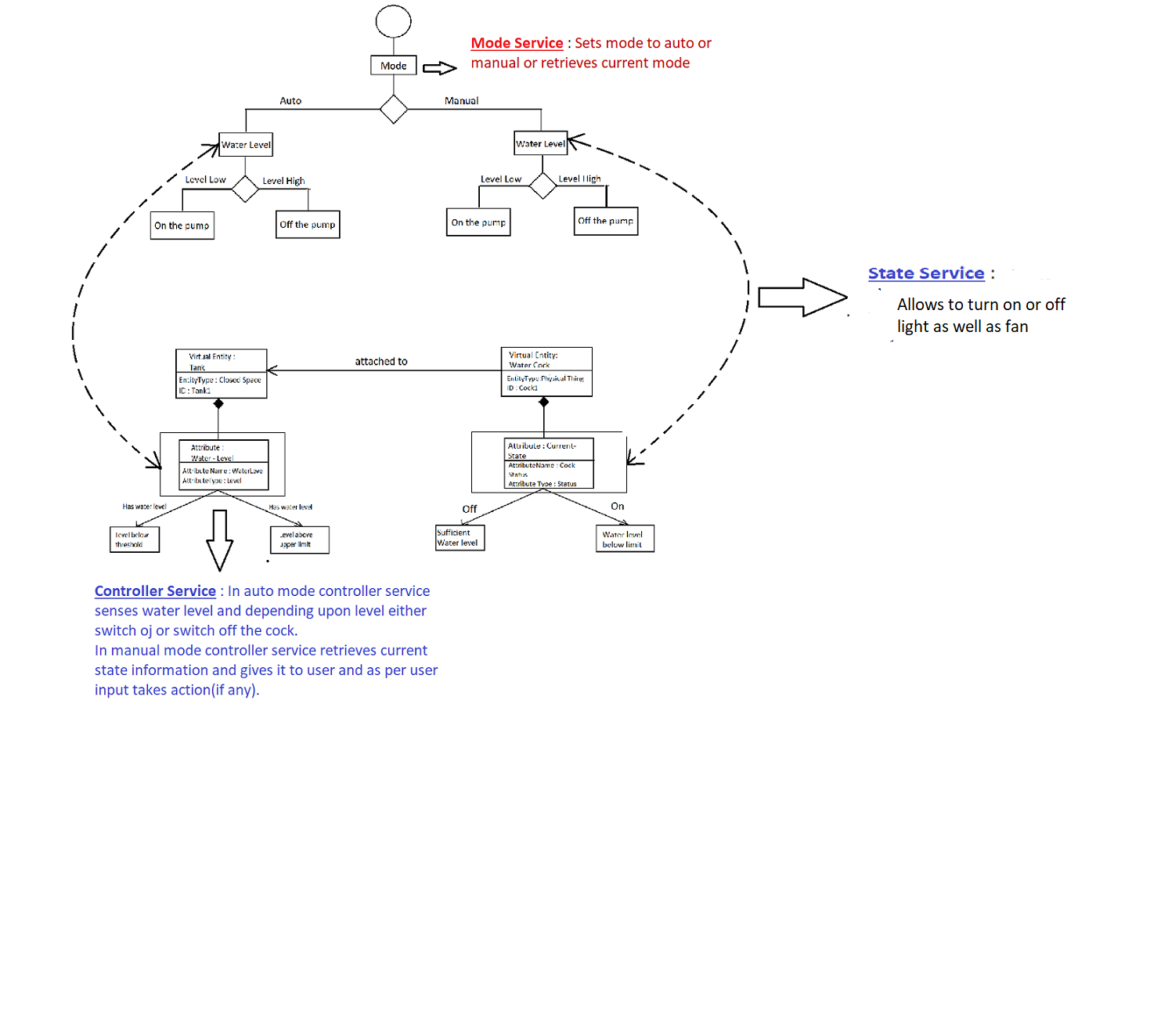
1.4 Information model specification

Information model defines structure of all information in IoT system. For example, attributes of virtual entities, relations etc. It does not defines how information is stored or displayed. Information model for Home automation is given below: To define the information model, we first list the Virtual Entities defined in the Domain Model. Information model adds more details to the Virtual Entities by defining their attributes and relations.



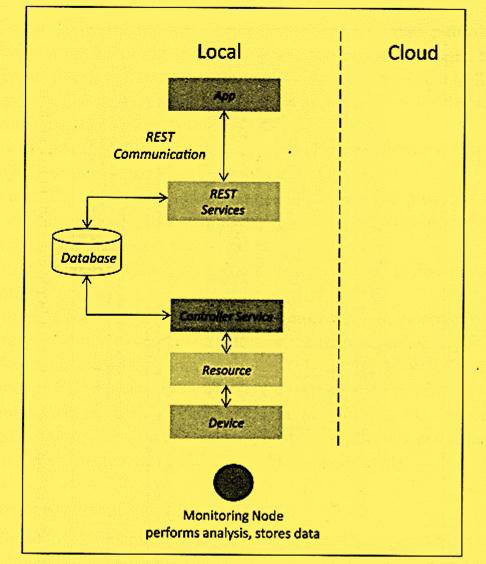
1.5 Service level specification:

From the process specification and information model, we identify the states and attributes. For each state and attribute, we define a service. These services either change the state or attribute values or retrieve the current values.



1.6 IoT level specification

The sixth step in the IoT design methodology is to define the IoT level for the system.



1.7 Functional view specification:

The Functional Groups (FG) included in a Functional View include:

Device: The device FG contains devices for monitoring and control. In the home automation example. the device FG includes a single board mini-computer, a light sensor and relay switch(actuator).

Communication: The communication FG handles the communication for the IoT system. The communication FG includes the communication protocols that form the backbone of IoT systems and enable network connectivity.

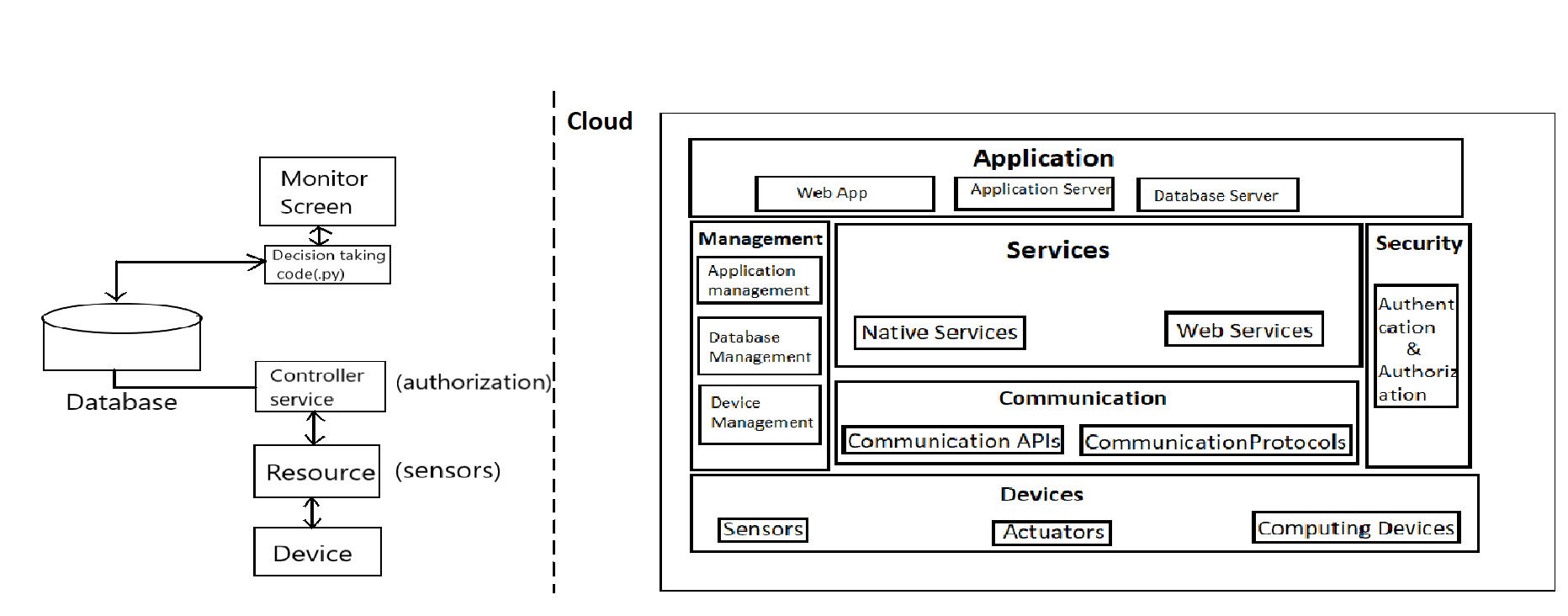
The communication FG also includes the communication AP is (such as REST and WebSocket) that are used by the services and applications to exchange data over the network.

•Services: The service FG includes various services involved in the IoT system such as services for device monitoring, device control services, data publishing services and services for device discovery.

•Management: The management FG includes all functionalities that are needed to configure and manage the loT system .

•Security: The security FG includes security mechanisms for the lot system such as authentication, authorization, data security, etc.

•Application: The application FG includes applications that provide an interface to the users to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and the processed data.



1.8 Operational view specification

In this step, various options pertaining to the lot system deployment and operation are defined, such as, service hosting options, storage options, device options, application hosting options, etc.

Operational View specifications for the home automation example are as follows:

•Devices: Computing device (Raspberry Pi), IR(sensor), LEDs.

•Communication APls: REST AP is

•Communication Protocols: Link Layer -802.11, Network Layer -1Pv4/1Pv6, Application -HTTP.

Application:

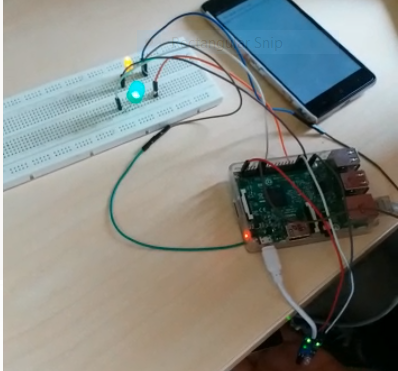
•Web Application -Django Web Application

Services:

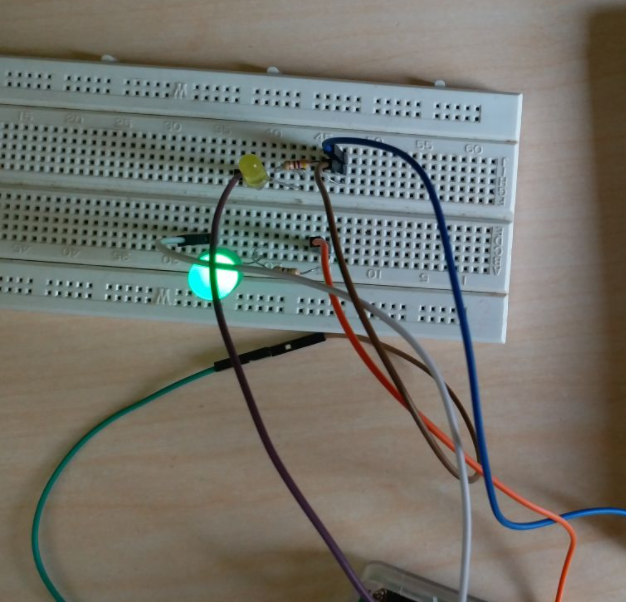
Controller Service -Hosted on device, implemented in Python and run as a native service.

•Mode service - hosted on device, implemented with Django-REST Framework.

1.9 Device and component integration:



1.10 Application Development



**2.SOURCE CODE**

**IOT/settings.py**

ALLOWED\_HOSTS = [‘192.168.43.3’,]

**Urls.py**

from django.conf.urls import patterns, include, url

# Uncomment the next two lines to enable the admin:

# from django.contrib import admin

# admin.autodiscover()

from control.views import auto\_mode,manual\_mode,light,turn\_light\_on,turn\_light\_off,turn\_ac\_on,turn\_ac\_off,change\_mode

urlpatterns = patterns[

# Examples:

# url(r'^$', 'IOT.views.home', name='home'),

# url(r'^IOT/', include('IOT.foo.urls')),

# Uncomment the admin/doc line below to enable admin documentation:

# url(r'^admin/doc/', include('django.contrib.admindocs.urls')),

# Uncomment the next line to enable the admin:

# url(r'^admin/', include(admin.site.urls)),

url(r'^auto/$',auto\_mode),

url(r'^manual/$',manual\_mode),

url(r'^light/$',light),

url(r'^lo/$',turn\_light\_on),

url(r'^loff/$',turn\_light\_off),

url(r'^ao/$',turn\_ac\_on),

url(r'^aoff/$',turn\_ac\_off),

url(r'^cm/$',change\_mode),

]

**Views.py**

# Create your views here.

from \_\_future\_\_ import unicode\_literals

from django.shortcuts import render

import time

import RPi.GPIO as GPIO

from django.http import HttpResponse

def auto\_mode(request):

sensor = 7

GPIO.setmode(GPIO.BCM)

GPIO.setup(sensor,GPIO.IN)

GPIO.setup(13,GPIO.OUT)

GPIO.setup(17,GPIO.OUT)

cnt=1

while True:

if GPIO.input(sensor):

GPIO.output(13,GPIO.HIGH) #Light

GPIO.output(17,GPIO.HIGH) #AC

if cnt == 1:

return HttpResponse('someone is in the room so turned on light and AC')

cnt=2;

else:

GPIO.output(13,GPIO.LOW)

GPIO.output(17,GPIO.LOW)

if cnt == 2:

return HttpResponse('no one is in the room so turned off light and AC')

cnt=3;

def manual\_mode(request):

return render\_to\_response('abcd/menu.html')

def turn\_light\_on(request):

GPIO.setmode(GPIO.BCM)

GPIO.setup(13,GPIO.OUT)

GPIO.output(13,GPIO.HIGH)

def turn\_light\_off(request):

GPIO.setmode(GPIO.BCM)

GPIO.setup(13,GPIO.OUT)

GPIO.output(13,GPIO.LOW)

def turn\_ac\_on(request):

GPIO.setmode(GPIO.BCM)

GPIO.setup(13,GPIO.OUT)

GPIO.output(13,GPIO.HIGH)

def turn\_ac\_off(request):

GPIO.setmode(GPIO.BCM)

GPIO.setup(17,GPIO.OUT)

GPIO.output(17,GPIO.LOW)

def cm(request):

return render\_to\_response('templates/index.html')

def light(request):

GPIO.setmode(GPIO.BCM)

GPIO.setup(12,GPIO.OUT)

GPIO.setup(16,GPIO.OUT)

GPIO.setup(20,GPIO.OUT)

GPIO.setup(21,GPIO.OUT)

while True:

for i in range(0,16):

x = str(bin(i))

if x[0] == 1:

GPIO.setup(12,GPIO.HIGH)

else:

GPIO.setup(12,GPIO.LOW)

if x[1] == 1:

GPIO.setup(16,GPIO.HIGH)

else:

GPIO.setup(16,GPIO.LOW)

if x[2] == 1:

GPIO.setup(20,GPIO.HIGH)

else:

GPIO.setup(20,GPIO.LOW)

if x[3] == 1:

GPIO.setup(21,GPIO.HIGH)

else:

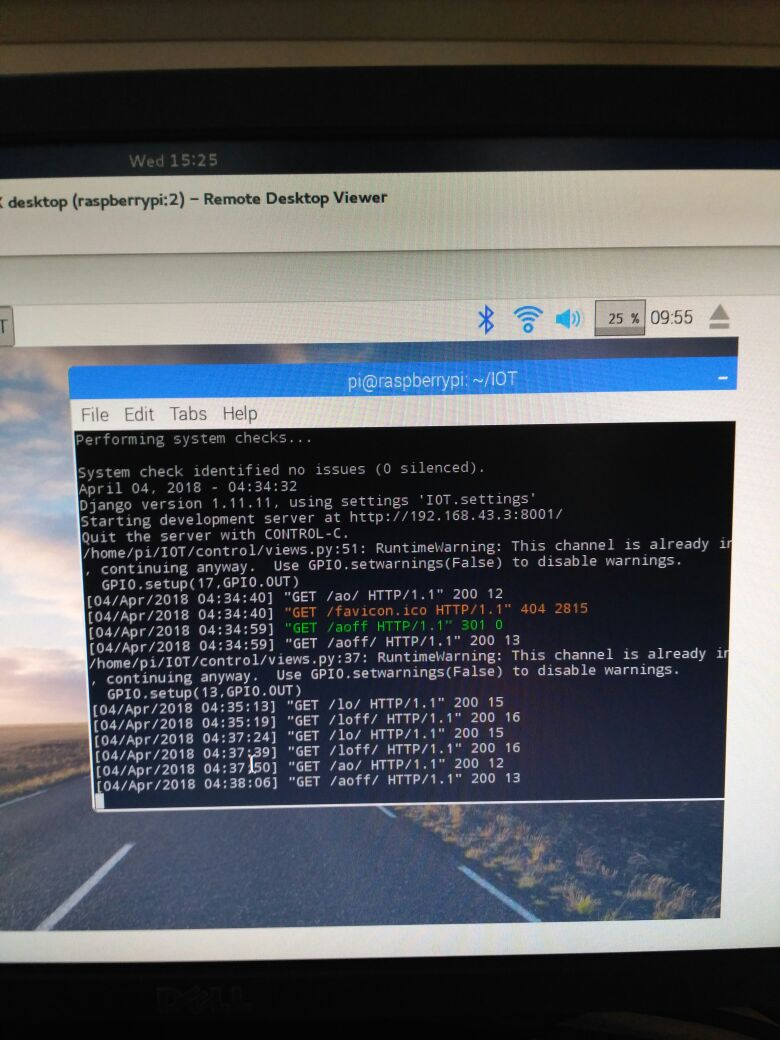
GPIO.setup(21,GPIO.LOW)

time.sleep(10)

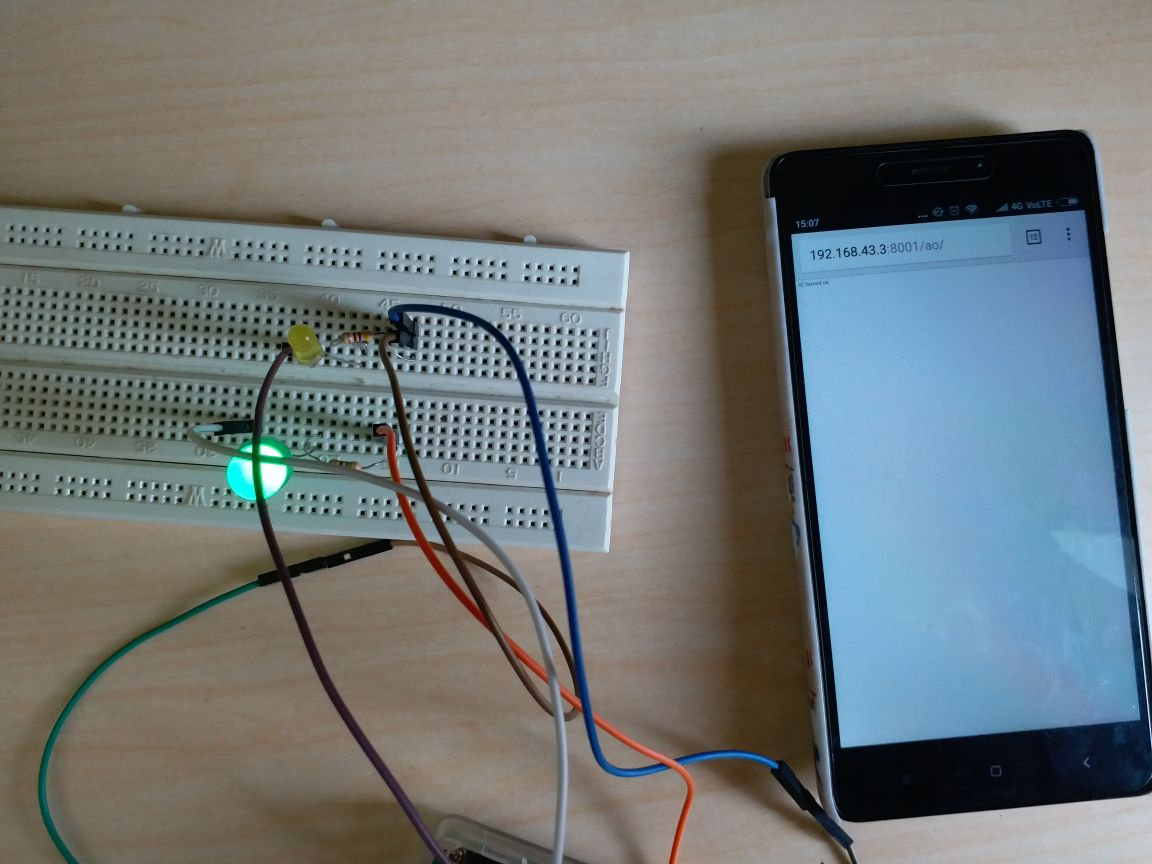
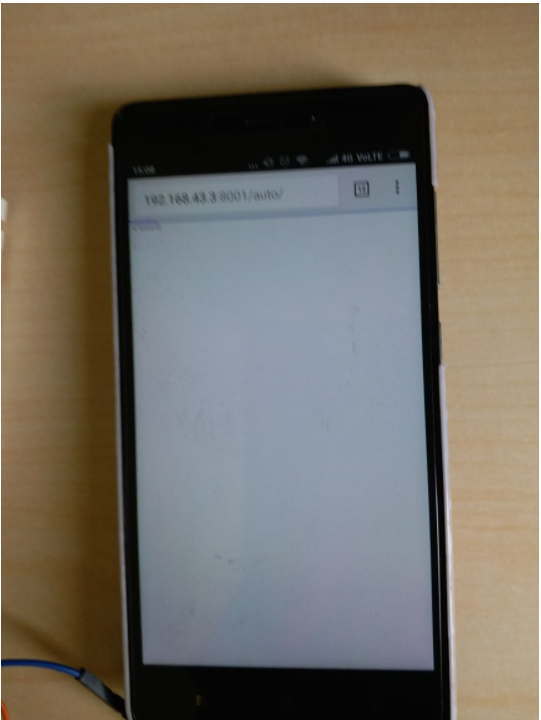
**3.DEPLOYMENT**

Deploy the project by firing following command:

python manage.py run server 192.168.43.3



**4.RESULTS**

****

****

**5.CONCLUSION**

The above developed application is created using raspberry pi, LEDs and Django framework. It worked on local system and one can access it remotely using mobile phone. It has been tested for various test cases and thus verified.