Road To Smart LNMIIT Using IoT

Project report submitted in partial fulfillment
of the requirements for the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

by

Deepika Singhal - Y13UC082
Kapil Matani - Y13UC132
Prachi Agarwal - Y13UC195
Sonal Sharma - Y13UC293

Under Guidance of
Dr. Santosh Shah

Department of Electronics and Communication Engineering
The LNM Institute of Information Technology, Jaipur

April 2017

The LNM Institute of Information Technology
Jaipur, India

CERTIFICATE

This is to certify that the project entitled "Road To Smart LNMIIT Using IoT", submitted by Deepika Singhal (Y13UC082), Kapil Matani (Y13UC132), Prachi Agarwal (Y13UC195) and Sonal Sharma (Y13UC293) in partial fulfillment of the requirement of degree in Bachelor of Technology (B. Tech), is a bonafide record of work carried out by them at the Department of Electronics and Communication Engineering, The LNM Institute of Information Technology, Jaipur, (Rajasthan) India, during the academic session 2016-2017 under my supervision and guidance and the same has not been submitted elsewhere for award of any other degree. In my/our opinion, this thesis is of standard required for the award of the degree of Bachelor of Technology (B. Tech).

Date

Adviser: Dr. Santosh Shah

Dedicated to LNMIIT: our Alma Mater

Acknowledgments

We take this opportunity to express our profound gratitude and deep regards to our guide Dr. Santosh Shah for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. His cordial support, valuable information and guidance helped us in completing this task through various stages. The blessing, help and guidance given by him, time to time, shall carry us a long way in the journey of life on which we are about to embark.

We are obliged to staff members of LNMIIT, Jaipur for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our project.

Abstract

This project investigates an automation system where different sensors will be taking the values from a lecture hall and after processing the information received, give commands to switch on/off the device or adjust their speeds. Lecture Hall automation system design is a smart solution to save time, money and man power with power saving and electricity. The proposed system needs to be set with the threshold values to various appliances in a lecture hall. The acquired sensor nodes will be used to send real-time sensor values onto a central node/ a gateway which will then display and give commands to different devices accordingly. It is cost effective but yet flexible, robust, adaptable and secure.

Contents

List of Figures

List of Tables

Chapter *1*

Introduction

The popularity of smart automated halls has been increasing greatly in recent years due to much higher affordability and simplicity through electronics gadgets connectivity like smartphone and tablet. The concept of the ”Internet of Things” has tied in closely with the popularization of automation.
Automation is a term used to describe the working together of all amenities and appliances. For example, a centrally-controlled LCD panel can have the capability to control everything from heating, air conditioning, security systems, audio systems, video systems, lighting, kitchen appliances, and home theatre installations. It is essential that the different controllable appliances be interconnected and communicate with each other. The basic aim of automation of lecture hall is to control or monitor signals from different appliances, or basic services. A central processing unit or web browser can be used to control or monitor the hall automation system.

1.1 Proposed Ideas For A Smart Campus

- Student Smart Card
- Smart Attendance System
- Energy Management System
- Water Management System
- Security Management System
- **Student Smart Card**

  Monetary card system can be used for payments at places like canteen, bus trips and overdue books in library.

  Implementation:

  LNMIIT is already using RFID card system which are currently being used for attendance purpose in classes. Certain change in algorithm of RFID can make it useful for monetary payments. Different RFID machines can be installed in respective canteen, college gate and library. Student can recharge their RFID cards and payments can be made by punching at machines.

- **Smart Attendance System**

  Implementation:

  Passive Wi-Fi transmissions can be decoded on any Wi-Fi device including routers, mobile phones and tablets. Students would be asked to register their smart phones for their registered courses. As soon as they enter in the lecture hall, their MAC address would be automatically be detected passive Wifi devices.

- **Energy Management System**

  Automatic turning off the lights in a room as soon as the person leaves.

  Implementation:

  Two Modules will be there. First one is known as "Digital Visitor counter" and second module is known as "Automatic room light controller". Main concept behind this project is known as "Visitor counter" which measures the number of persons entering in any room like hostel room, classroom. This function is implemented using a pair of Infrared sensors. LCD display placed outside the room displays this value of person count. This person count will be incremented if somebody enters inside the room and at that time lights are turned on. And in reverse way, person count will be decremented if somebody leaves the room. When number of persons inside the room is zero, lights inside the room are turned off using a relay interface. In this way Relay does the operation of "Automatic room light controller". Since this idea uses 2 infrared sensors, it can be used as Bidirectional person counter as well.

- **Water Management System**

  This system is sensor based automatic water tap. The tap starts whenever anyone put his/her hand in front of sensor which is below the water tap.

  Implementation:

  The IR sensor is made using 555 timer, used as a stable multivibrator, vibrating at around 38 KHz. and another 555 timer IC is used as monostable multivibrator for tap circuit.

  1. The Sensor Circuit:

     The sensor circuit is based on the 555 timer IC in astable mode. 555 timer IC vibrates at 38KHz and that waves are transmitted using IR led and if any obstacle is in front of IR led the rays reflects and the TSOP1738 IR receiver modules receives and sends signal.

  2. The Main Circuit:

     In main circuit 555 timer IC is used in monostable mode hence when it receives any signal from IR sensor then automatically enables the solenoid valve for some time and then automatically turn off the output current in BT136 Triac.

- **Security Management System**

  Access cards can be implemented in library and labs where students being registered for that lab course can easily have an access in lab at any point of time according to his/her own use.

Implementation:
LNMIIT is already using RFID card system which are currently being used for attendance purpose in classes. Certain change in algorithm of RFID can make it useful for access in labs. Different RFID machines can be installed in labs.

1.2 The Area of Work

Implementations in our project:

1. Controlling Temperature of the lecture hall
2. Controlling Lights of the lecture hall
3. Turning On and Off electrical appliances present

Our aim through this project is to make a smart lecture hall in our campus. We are applying this to one of the lecture hall and then it would be implemented in others also. In this, we will be controlling different appliances of the hall which includes switching on/off the lights and fans.

1.3 Problem Addressed

1. To design and implement prototype of smart hall system to minimize human efforts and conserve energy.
2. To design low cost system with user friendly functionality.

1.4 Motivation

1. Smart hall systems: convenience, security, energy management and connectivity.
2. Smart hall systems are not yet fully integrated into our society.
3. The cost of installing a smart hall is very high.
4. Our main motivation behind this project is Energy Management at an affordable cost.

1.5 Existing System And Disadvantages

- Categories:
    1. Internet based monitoring using Servers, GPRS modem
    2. Monitoring using wireless sensor network

3. Wireless monitoring using Bluetooth, Wi-Fi, Zigbee and RF.

- Disadvantages:

1. Cater towards upper class
2. Difficult to install and troubleshoot
3. Focuses on one way appliance
4. Proprietary and closed.

Chapter *2*

Literature Survey

2.1 Introduction

For this B.Tech project, We carried out a Literature Survey after which we were able to understand the functioning and architecture of a Sensor node and Sensor Network and what are the components would be best to use in a sensor network. We visited several blogs/websites of different companies who provide equipment for the purpose of Internet of things like Raspberry Pi foundation,Texas Instruments, Arduino. By Looking at their efficiency, specifications and their availability in market, we had chosen the components to use for this project.

2.2 Internet Of Things (IOT)

The Internet of Things is closely identified with RFID as the method of communication, although it also may include other sensor technologies, wireless technologies or QR codes.[1]It is basically a computing concept that describes a future where every object will be connected to the Internet and be able to identify themselves to other devices.

Through IOT, no longer does the object relate just to you, but is now connected to surrounding objects and database data this is why IoT is significant because an object that can represent itself digitally becomes something greater than the object by itself and when many objects act in unison, they are known as having ”ambient intelligence.”

Figure 2.1 Basic architecture of IOT

Different Applications for IoT Technology:

- Construction and Infrastructure
- Energy Saving
- Environmental Monitoring
- Smart Wearable
- Smart City
- Medical and Health Care
- Home Automation
- Smart Attendance System
- Industrial IoT
- Transportation

2.3 Sensor Node

A sensor node, is a node in a wired/wireless sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network.[2]

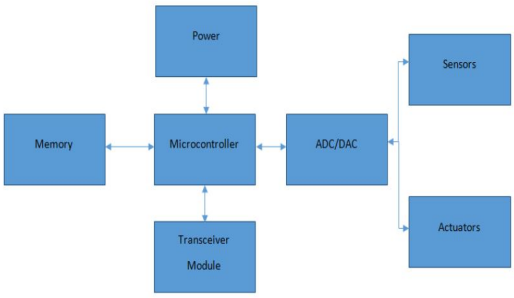

Figure 2.2 Basic Architecture of a sensor node

A Sensor node is generally composed of the following parts:

1. **Controller**
   The controller performs tasks, processes data and controls the functionality of other components in the sensor node.The most common controller is a microcontroller.

2. **Transceiver**
   It is used for sending and receiving information wirelessly. Most transceivers operating in idle mode have a power consumption almost equal to the power consumed in receive mode.

3. **External Memory**
   Two categories of memory based on the purpose of storage are: user memory used for storing application related or personal data, and program memory used for programming the device.

4. **Power source**
   Generally NiCd or Lithium Ion batteries.

5. **Sensors**
   Sensors are used by wireless sensor nodes to capture data from their environment. They are hardware devices that produce a measurable response to a change in a physical condition like temperature or pressure. Sensors measure physical data of the parameter to be monitored and have specific characteristics such as accuracy, sensitivity etc.

2.4 Sensor Network

A Sensor Network is like a distributed network that comprises of a different number of embedded systems which are connected to each other through some specific set of wireless or wired architecture. It also consists of gateway device which helps to connect our wired or wireless node distribution to the internet. Embeddeed systems here, are referred as Sensor nodes.

Parameters which are monitored like power-line voltage, chemical concentrations, pollutant levels and vital body functions, temperature, humidity, pressure, wind direction and speed, illumination intensity, vibration intensity and sound intensity,

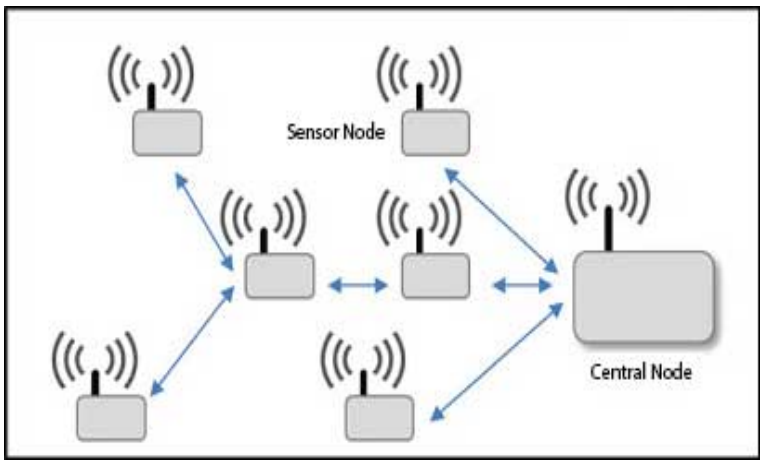

Figure 2.3 Proposed sensor network

2.5 Bluetooth Low Energy

Bluetooth Low Energy or Bluetooth Smart is a wireless PAN(personal area network) technology designed and marketed by Bluetooth Special Interest Group(BSIG).[3]

Bluetooth Low Energy(BLE) will be very useful in a TI SensorTag as through BLE we can easily connect with iOS, Android, Windows Phone and BlackBerry, as well as OS X, Linux and Windows 8.

Compared to the classic Bluetooth technology, Bluetooth Smart is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range.

Low Energy Profile makes use of ATT as its application protocol. Its attribute is composed by 3 elements:

1. A 16 bit handle
2. An UUID which defines the attribute type
3. A value of a certain length

2.6 Arduino Uno

The Arduino Uno is a microcontroller board based on the ATmega328 which contains 14 digital input/output pins of which 6 can be used as PWM(Pulse width modulation) outputs, 6 analog inputs, a power jack, an ICSP header, and a reset button 16 MHz crystal oscillator and a USB connection.

Simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started[4]. Therefore, it contains everything that is needed to support the micro-controller

It features the Atmega8U2 programmed as a USB-to-serial converter.

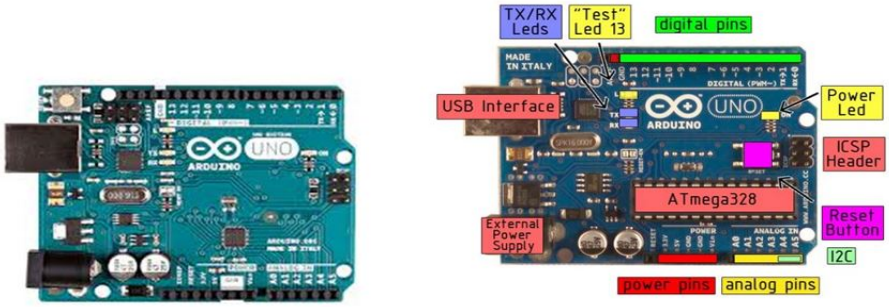

Figure 2.4 Arduino Uno and its pin diagram

Features:

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6 DC
- Current per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB of which 0.5 KB
- SRAM: 2 KB, EEPROM: 1 KB, Clock Speed: 16 MHz

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically

2.7 Raspberry Pi V2

The Raspberry Pi 2 is a micro-controller which is second generation Raspberry Pi and will be used as the gateway or as central node in our Sensor Node Network.[5]
Compared to the Raspberry Pi 1 it has:

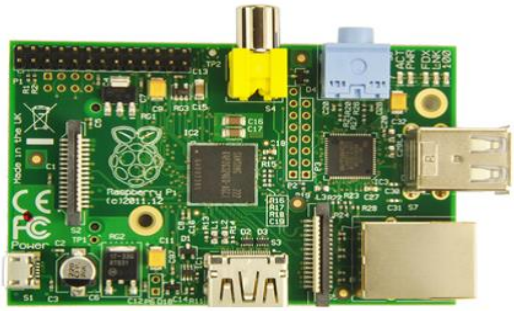

Figure 2.5 Raspberry Pi V2

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM

Like the (Pi 1) Model B+, it also has:

- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

Because it has an ARMv7 processor, it can run the full range of ARM GNU/Linux distributions as well as Microsoft Windows 10.

2.8 Sensor Tag from TI

The TI Sensor Tag contains CC2650[8] which is wireless MCU having ARM Cortex M3 as main Central Processing Unit.
The TI Sensor Tag[6] is a product Texas Instruments which is launched some years back.

The TI E2E[9] community is also very supportive and highly informative. TI[7] has made SensorTag an open source (GitHub) and has provided with schematics and reference designs for the same.

Figure 2.6 Sensor Tag

Why TI sensor Tag ?

- The components are connected and the hardware is already made, we can directly start to code the CC2650 to take values from different sensors and send it wirelessly. The SensorTag already has a built-in PCB antenna for RF communications for customization.
- The cost for each SensorTag is only $29 which is really inexpensive and affordable as compared to the MSP430f1611 which has $40 for only the development board or embed LPC1768 which is ARM Cortex M3 Development board for $49.
- There are some sensors already installed in the hardware of the Sensor Tag which measure:
  (a) Temperature
  (b) Humidity
  (c) Light
  (d) Motion Sensing
  (e) Pressure/Altitude
  (f) The Sensor Tag also has GPIO pins for further expansion with other sensors or actuators.
- There are three wireless communication stack that can be used in Sensor Tag:
  (a) ZigBee

(b) 6LowPAN

(c) Bluetooth Low Energy

2.9 Voice Recognition Module V3

The proposed method for controlling podium lights uses a Voice Recognition Module V3 for voice processing and recognition.
It can support maximum 80 voice commands each of 1500ms and maximum 7 voice commands are effective at the same time [8] which are stored in container from flash memory.In this module the Voice Recognition V3 library of Arduino is imported. There are two steps to load a command in voice module:

1. **Train** – the process of recording your voice commands using microphone, stored in flash memory(number from 0 to 79).

2. **Load** – copy trained voice to recognizer.

Recognizer is a memory container which stores acting voice commands. Max 7 voice commands can be loaded at a time. It is core part of voice recognition module. Group helps to manage recorded voices, each group has maximum 7 records.

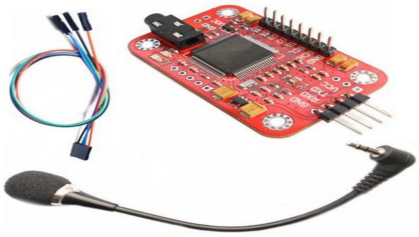

Figure 2.7 Voice Recognition Module V3

Chapter *3*

Proposed Work

3.1 Basic Functional Model

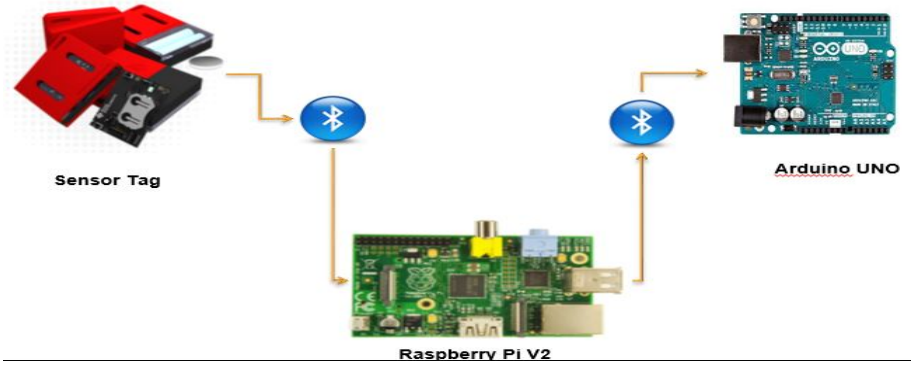

Figure 3.1 Basic Function Model of Project

3.2 Architecture

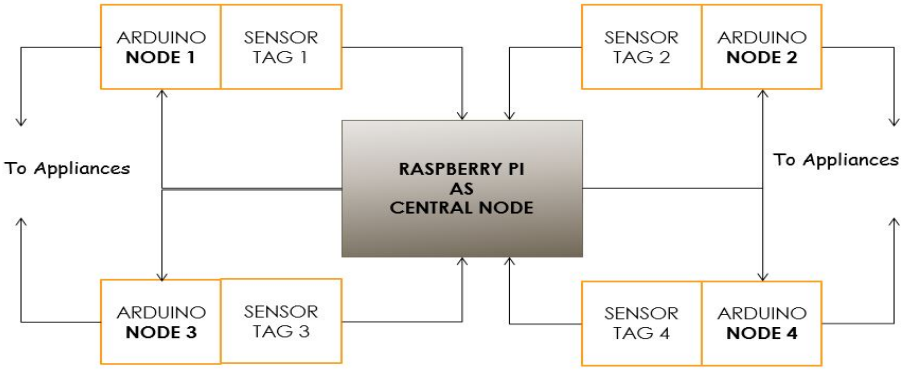

Figure 3.2 Architecture

3.3 Visual Representation

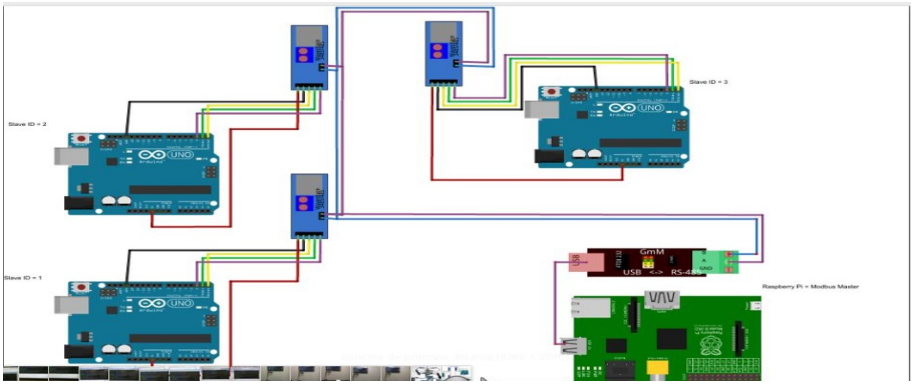

Figure 3.3 Connection Diagram between sensor nodes and central node

3.4 Arduino As Actuator

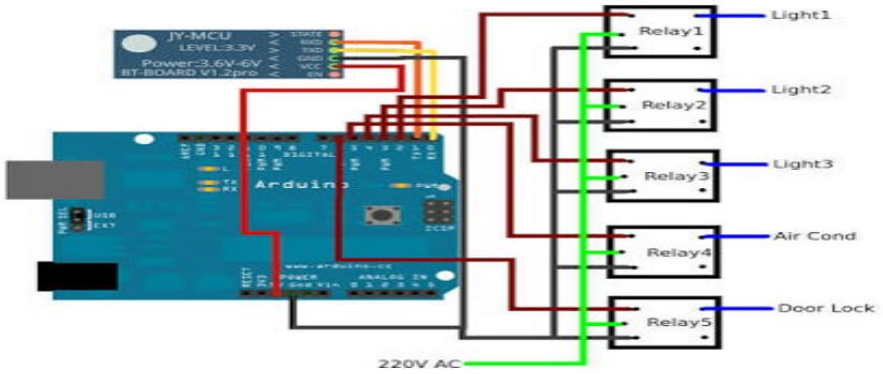

Figure 3.4 Connections between Arduino and devices

3.5 Architecture for voice controlled podium lights

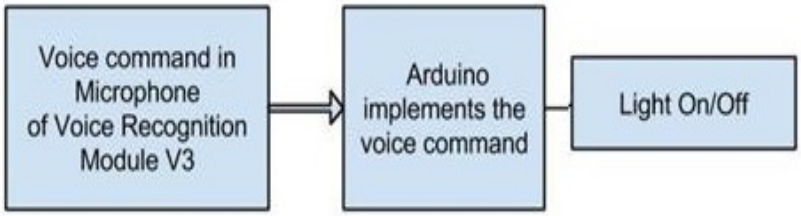

Figure 3.5 Architecture for voice controlled podium lights

3.6 Connection of V3 module with Arduino

Arduino		VR Module
5V	---->	5V
2	---->	TX
3	---->	RX
GND	---->	GND

Figure 3.6 Basic connections

Chapter *4*

Simulation and Results

4.1 Setting Up the Development Environment

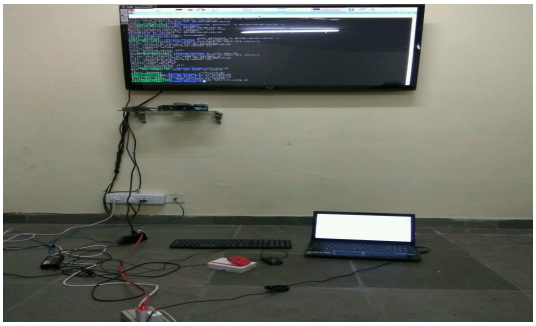

Figure 4.1 Basic architecture of IOT

4.2 Sensing Values from the Sensor Tag

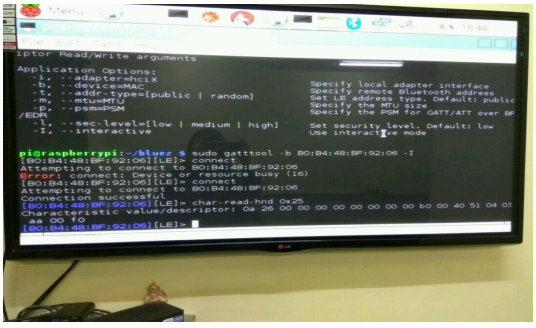

Figure 4.2 Sensor Tag values

4.3 Data Collection

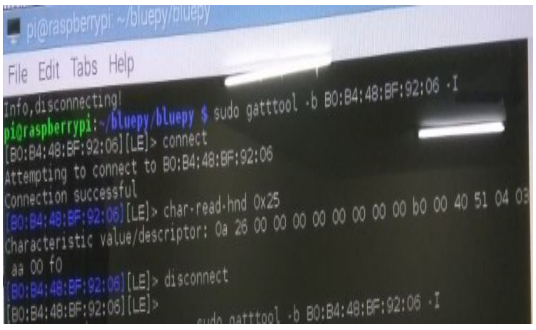

Figure 4.3 Manually Taking Values

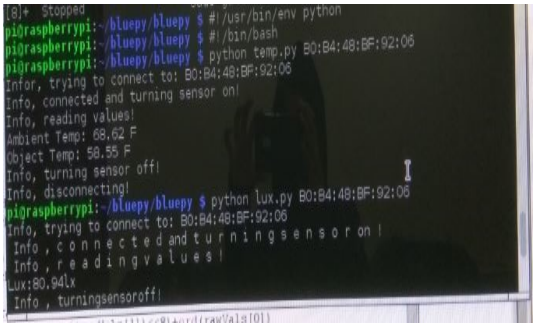

Figure 4.4 Automating the Data Collection Process

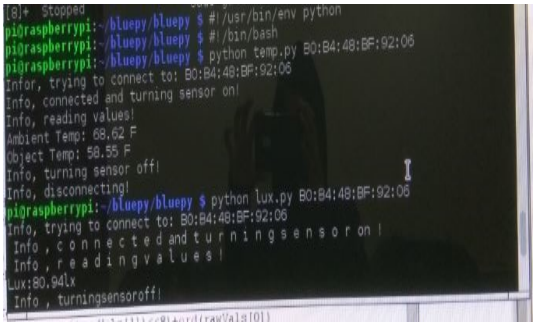

Figure 4.5 Serially Taking Sensor Data from Multiple Sensor Tags

4.4 Taking Sensor Data from Multiple Sensor Tags

Here we have connected 2 different sensor tags to the central node. The central node decides from which sensor tag the value has to be read.

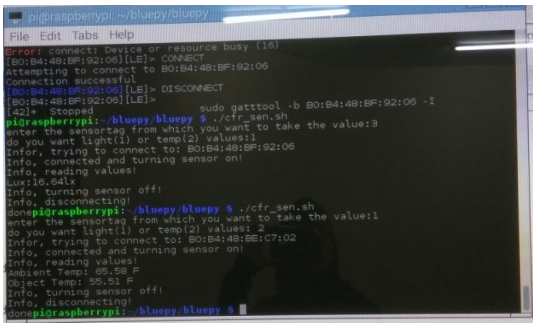

Figure 4.6 Multiple Sensor Tag

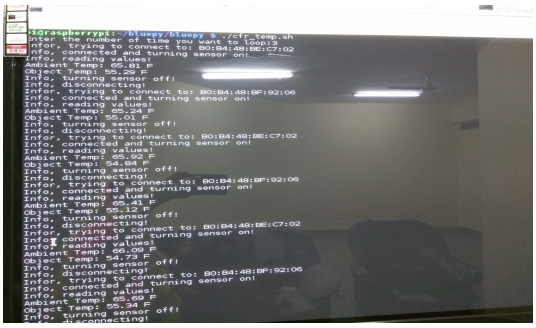

Figure 4.7 Temperature values from 2 sensors

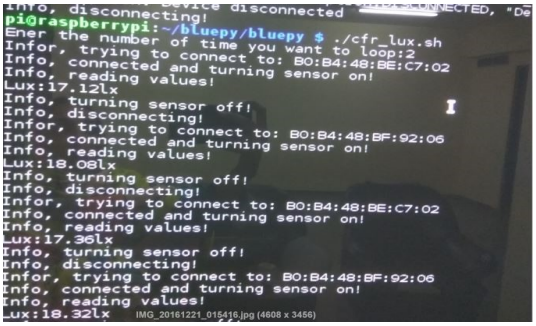

Figure 4.8 Lux values from 2 sensors

4.5 Finding the threshold value of light and temperature for a lecture hall

For calculating threshold value, we have done a survey by taking a group of 10 students. Each students experiences different temperature and light sensitivity. The value for each student is noted and the average of the 10 values is taken as threshold.

Values from survey	
Temperature values	Lux values
65.80	17.26
66.02	17.12
62.98	18.32
63.32	17.08
65.24	17.76
65.42	18.08
63.77	18.38
64.38	17.76
66.42	18.28
65.54	17.36
64.88	17.72

Table 4.1 Threshold value calculation

4.6 Comparison with the threshold values and controlling the devices

1. In this, central node decides from which sensor tag the value has to be read from different sensor tags connected to the node.

2. Central node compares the data with the threshold values and controls the led according to the result.

4.7 When the central node (RPi) will send information to the corresponding nodes (arduino)

1. Here, we have taken the example of only one child node.

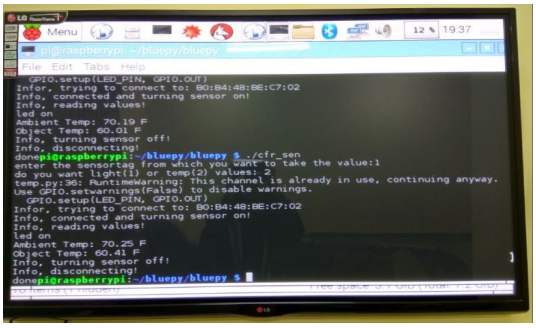

Figure 4.9 Comparing sensor tag values with threshold

2. We have connected Arduino with Raspberry Pi where Pi will send the information to Arduino and controls the devices.

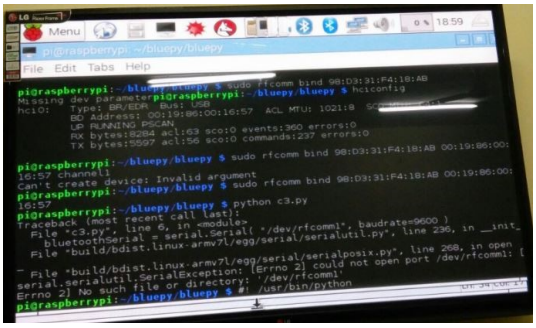

Figure 4.10 Connection between Arduino and RPi

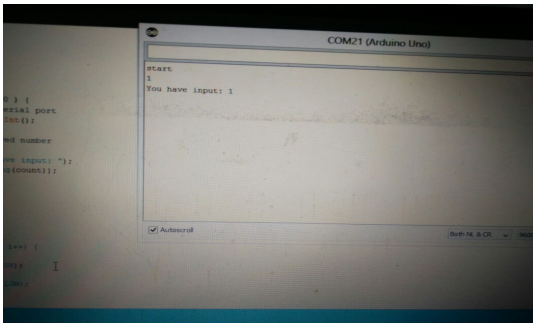

Figure 4.11 Output screen showing result at Arduino

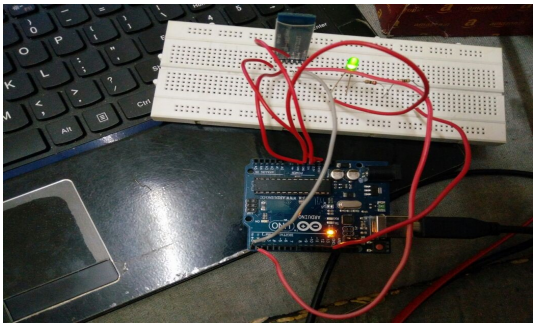

Figure 4.12 Led showing the output

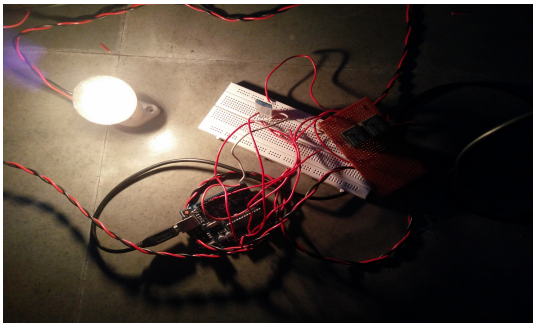

Figure 4.13 Real-time implementation-Light on

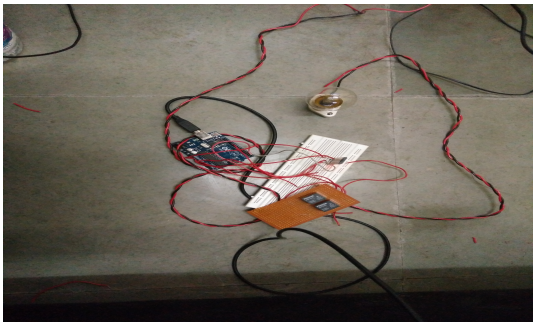

Figure 4.14 Real-time implementation-Light off

4.8 Recording voice over V3 Module

In the voice recognition part of the system the voice is captured by the microphone and given to the Voice Recognition Module V3. To train a command we have to follow these steps

1. Upload code in Arduino through Arduino IDE.
2. Open Serial Monitor. Set baud rate 115200, set send with Newline or Both NL and CR.
3. In order to train record 0 with signature ”On”, send “sigtrain 0 On” command. When Serial Monitor prints ”Speak now”, you need to speak your voice and when Serial Monitor prints ”Speak again”, speak it again. If these two voice are matched, Serial Monitor prints ”Success”, and ”record 0” is trained, or if are not matched, repeat speaking until success.
4. Send “load 0 1” command to load voice in recognizer and speak the command in microphone to check whether your voice is recognised or not. If recognised, then VR Index, Group, Record Number and Signature details are shown.

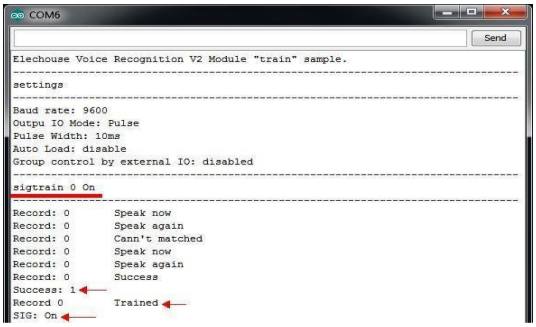

Figure 4.15 Recording voice

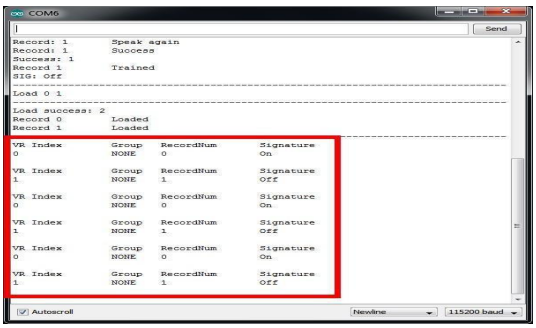

Figure 4.16 Voice recorded successfully

4.9 Calculating average number of attempts required to record voice successfully and also the distance from which it must be recorded

For calculating these values, we have done a survey by taking a group of 10 students. Each students speaks differently from various distances. The value for each student is noted and the average of the 10 values is taken as optimum value.

Values from survey	
Best distance(cm)	Number of attempts(avg)
5	1
5.5	1
6.2	2
5.1	1
6.0	3
4.4	1
7.0	2
6.5	1
4.5	1
5.2	2
5.5	1.4

Table 4.2 Data for calculating optimum distance and average number of attempts.

Chapter *5*

Conclusions and Future Work

In this B.Tech project,we have an extensive study and effort to connect the components of a Sensor node network through Bluetooth module and Bluetooth Low Energy Module.

Real time sensor values have been serially logged and process on central node from various child nodes in implemented sensor node network. Sensor node network is a star topology network.

Controlled lights of podium through voice recognition module.The speech commands are loaded and recognised by Voice Recognition Module V3 and implemented on different appliances using micro-controller (Arduino-UNO). Experiments were conducted with different speakers (male/female voice) while varying the distance between mic and mouth. Different commands were prompted and responses are analysed on the above parameters.

5.1 Scope of Further Work

The real time values which was obtained through sensor nodes can be used in:

- Uploading these sensor values on cloud for Data Analytics and central control
- Setting different threshold Values for different Classrooms according to time of the day/season
- By broadcasting the sensor values from using individual sensors for every purpose, the process of obtaining values from sensor tags can be fastened as compared to obatining values serially which is a slow process.
- The current firmware on the TI Sensor Tag is not that power efficient, hence the firmware can also be updated.
- Voice module could be used to control other appliances also in the lecture room.

Bibliography

[1] Internet Of Things (TI),
http://www.ti.com/ww/en/internet of things/iot-overview.html

[2] Sensor Node (Wikipedia),
https://en.wikipedia.org/wiki/Sensor node

[3] Bluetooth Low Energy(BLE),
https://en.wikipedia.org/wiki/Bluetooth low energy

[4] Arduino Uno Website,
https://forum.arduino.cc/

[5] Raspberry Pi V2,
https://www.raspberrypi.org/

[6] Sensor Tag (TI),
http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/index.html

[7] Texas Instruments Website,
http://www.ti.com/

[8] CC2650 MCU,
http://www.ti.com/product/cc2650

[9] TI E2E Community,
https://e2e.ti.com/

[10] GitHub- Ian Harvey,
https://github.com/IanHarvey/bluepy

Chapter *6*

Appendix

#Python Code for reading temperature

```
#!/usr/bin/env python
import struct, sys, traceback
from btle import UUID, Peripheral, BTLEException


def TI_UUID(val):
        return UUID("%08X-0451-4000-b000-000000000000" % (0xF0000000+val))

config_uuid= TI_UUID(0xAA02)
data_uuid=TI_UUID(0xAA01)

sensorOn= struct.pack("B",0x01)
sensorOff= struct.pack("B",0x00)

if len(sys.argv)!=2:
        print "Fatal, must pass device address:", sys.argv[0], "<device address>"
        quit()
try:
        print "Infor, trying to connect to:", sys.argv[1]
        p = Peripheral(sys.argv[1])

except BTLEException:
        print "Fatal, unable to connect!"

except:
        print "Fatal, unexpected error!"
        traceback.print_exc()
```

```
        raise

else:

        try:
                print "Info, connected and turning sensor on!"
                ch=p.getCharacteristics(uuid=config_uuid)[0]
                ch.write(sensorOn, withResponse=True)

                print "Info, reading values!"
                ch=p.getCharacteristics(uuid=data_uuid)[0]
                rawVals=ch.read()
                rawVal =(ord(rawVals[1])<<8)+ord(rawVals[0])


                mantissa = rawVal & 0x0FFF;
                exponent=(rawVal>>12) & 0xFF ;
                magnitude = pow(2.0 , exponent) ;
                output = (mantissa* magnitude ) ;
                print "Lux:%4.2flx"%(output/100.0)

                print "Info, turning sensor off!"
                ch=p.getCharacteristics(uuid=config_uuid)[0]
                ch.write(sensorOff, withResponse=True)

        except:
                print "Fatal, unexpected error!"
                traceback.print_exc()
                raise

        finally:
                print "Info, disconnecting!"
                p.disconnect()


finally:
        quit()
```

#Python Code for reading temperature

```
#!/usr/bin/env python
import struct, sys, traceback
from btle import UUID, Peripheral, BTLEException


def TI_UUID(val):
        return UUID("%08X-0451-4000-b000-000000000000" % (0xF0000000+val))

config_uuid= TI_UUID(0xAA02)
data_uuid=TI_UUID(0xAA01)

sensorOn= struct.pack("B",0x01)
sensorOff= struct.pack("B",0x00)

if len(sys.argv)!=2:
        print "Fatal, must pass device address:", sys.argv[0], "<device address>"
        quit()
try:
        print "Infor, trying to connect to:", sys.argv[1]
        p = Peripheral(sys.argv[1])

except BTLEException:
        print "Fatal, unable to connect!"

except:
        print "Fatal, unexpected error!"
        traceback.print_exc()
        raise

else:

        try:
                print "Info, connected and turning sensor on!"
                ch=p.getCharacteristics(uuid=config_uuid)[0]
                ch.write(sensorOn, withResponse=True)

                print "Info, reading values!"
```

```
                ch=p.getCharacteristics(uuid=data_uuid)[0]
                rawVals=ch.read()

                rawVobj = (ord(rawVals[1])<<8)+ord(rawVals[0])
                rawTamb = (ord(rawVals[3])<<8)+ord(rawVals[2])



                print "Ambient Temp: %.2f F" % float((rawTamb*0.25*0.03125)*1.8+3
                print "Object Temp: %.2f F" % float((rawVobj*0.25*0.03125)*1.8+32

                print "Info, turning sensor off!"
                ch=p.getCharacteristics(uuid=config_uuid)[0]
                ch.write(sensorOff, withResponse=True)

        except:
                print "Fatal, unexpected error!"
                traceback.print_exc()
                raise

        finally:
                print "Info, disconnecting!"
                p.disconnect()


finally:
        quit()
```

#Shell script for reading lux values

```
#!/bin/bash
echo -n "Ener the number of time you want to loop:"
read n
for((i=0;i<n;i++))
do
        python lux.py B0:B4:48:BE:C7:02 #|grep -o '# .*'|
#       tee -a /home/pi/Lux_values/tag2.txt
        python lux.py B0:B4:48:BF:92:06 #|grep -o '# .*'|
 #       tee -a /home/pi/Lux_values/tag4.txt

Done
```

#Shell script for reading temperature values

```
#!/bin/bash
echo -n "Enter the number of time you want to loop:"
read n
for((i=0;i<n;i++))
do
        python temp.py B0:B4:48:BE:C7:02
        python temp.py B0:B4:48:BF:92:06
done
echo -n "done"
```

#Shell script for reading from multiple sensor tags

```
#!/bin/bash
echo -n "enter the sensortag from which you want to take the value:"
read m
if [[ m -eq "1" ]];
then
        echo -n "do you want light(1) or temp(2) values: "
        read o1
        if [[ o1 -eq "1" ]];
        then
                python lux.py B0:B4:48:BE:C7:02
        elif [[ o1 -eq "2" ]];
        then
                python temp.py B0:B4:48:BE:C7:02
        fi
```

```
elif [[ m -eq "2" ]];then
        echo -n "do you want light(1) or temp(2) values:"
        read o2
        if [[ o2 -eq "1" ]];then
                python lux.py B0:B4:48:CF:FB:02
        elif [[ o2 -eq "2" ]] ;then
                python temp.py B0:B4:48:CF:FB:02
        fi
elif [[ m -eq "3" ]];then
        echo -n "do you want light(1) or temp(2) values:"
        read o3
        if [[ o3 -eq "1" ]] ;then
                python lux.py B0:B4:48:BF:92:06
        elif [[ o3 -eq "2" ]] ;then
                python temp.py B0:B4:48:BF:92:06
        fi
fi
echo -n "done"
```

#Python Code for reading temperature, comparing with its threshold values and getting LED ON/OFF

```
#!/usr/bin/env python
import struct, sys, traceback
import time
import os
import RPi.GPIO as GPIO
from btle import UUID, Peripheral, BTLEException
import serial
from time import sleep


def TI_UUID(val):
        return UUID("%08X-0451-4000-b000-000000000000" % (0xF0000000+val))


config_uuid= TI_UUID(0xAA02)
data_uuid=TI_UUID(0xAA01)

sensorOn= struct.pack("B",0x01)
sensorOff= struct.pack("B",0x00)
```

```
LED_PIN = 3
amb_thresh = 30.00
obj_thresh = 25.00

def GPIOsetup():
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(LED_PIN, GPIO.OUT)
        GPIO.setwarnings(False)

def ledON():
        GPIO.output(LED_PIN,1)
        print "led on"
        return()


def ledOFF():
        GPIO.output(LED_PIN,0)
        print "led off"
        return()

def ardu(val):
        bluetoothSerial = serial.Serial( "/dev/rfcomm0", baudrate=9600 )
        print val


        bluetoothSerial.write( str(val) )
        print bluetoothSerial.readline()

        return()

def getTemp():

        if len(sys.argv)!=2:
                print "Fatal, must pass device address:", sys.argv[0], "<device a
                quit()
        try:
                print "Infor, trying to connect to:", sys.argv[1]
```

```
        p = Peripheral(sys.argv[1])

except BTLEException:
        print "Fatal, unable to connect!"

except:
        print "Fatal, unexpected error!"
        traceback.print_exc()
        raise

else:

        try:
                print "Info, connected and turning sensor on!"
                ch=p.getCharacteristics(uuid=config_uuid)[0]
                ch.write(sensorOn, withResponse=True)

                print "Info, reading values!"
                ch=p.getCharacteristics(uuid=data_uuid)[0]
                rawVals=ch.read()

                rawVobj = (ord(rawVals[1])<<8)+ord(rawVals[0])
                rawTamb = (ord(rawVals[3])<<8)+ord(rawVals[2])

                amb = float((rawTamb*0.25*0.03125)*1.8+32)
                obj = float((rawVobj*0.25*0.03125)*1.8+32)

                if amb>amb_thresh:
                        ledON()
                        ardu(1)
                else:
                        ledOFF()
                        ardu(0)


                print "Ambient Temp: %.2f F" % float((rawTamb*0.25*0.0312

                print "Object Temp: %.2f F" % float((rawVobj*0.25*0.03125
```

```
                print "Info, turning sensor off!"
                ch=p.getCharacteristics(uuid=config_uuid)[0]
                ch.write(sensorOff, withResponse=True)

            except:
                print "Fatal, unexpected error!"
                traceback.print_exc()
                raise

            finally:
                print "Info, disconnecting!"
                p.disconnect()


    finally:
        quit()

    return()

GPIOsetup()
getTemp()
```

#Code for Controlling Light through Voice

```
#include <SoftwareSerial.h>
#include "VoiceRecognitionV3.h"
#include<Servo.h>

/**
  Connection
  Arduino    VoiceRecognitionModule
   2   ------->     TX
   3   ------->     RX
*/
VR myVR(2,3);     // 2:RX 3:TX, you can choose your favourite pins.
uint8_t records[7]; // save record
uint8_t buf[64];

int light = 3;
```

```
#define onRecord    (0)
#define offRecord   (1)

/**
  @brief    Print signature, if the character is invisible,
            print hexible value instead.
  @param    buf     --> command length
            len     --> number of parameters
*/
void printSignature(uint8_t *buf, int len)
{
  int i;
  for(i=0; i<len; i++){
    if(buf[i]>0x19 && buf[i]<0x7F){
      Serial.write(buf[i]);
    }
    else{
      Serial.print("[");
      Serial.print(buf[i], HEX);
      Serial.print("]");
    }
  }
}

/**
  @brief    Print signature, if the character is invisible,
            print hexible value instead.
  @param    buf  -->  VR module return value when voice is recognized.
              buf[0]  -->  Group mode(FF: None Group, 0x8n: User, 0x0n:System
              buf[1]  -->  number of record which is recognized.
              buf[2]  -->  Recognizer index(position) value of the recognized reco
              buf[3]  -->  Signature length
              buf[4]~buf[n] --> Signature
*/
void printVR(uint8_t *buf)
{
  Serial.println("VR Index\tGroup\tRecordNum\tSignature");
```

```
  Serial.print(buf[2], DEC);
  Serial.print("\t\t");

  if(buf[0] == 0xFF){
    Serial.print("NONE");
  }
  else if(buf[0]&0x80){
    Serial.print("UG ");
    Serial.print(buf[0]&(~0x80), DEC);
  }
  else{
    Serial.print("SG ");
    Serial.print(buf[0], DEC);
  }
  Serial.print("\t");

  Serial.print(buf[1], DEC);
  Serial.print("\t\t");
  if(buf[3]>0){
    printSignature(buf+4, buf[3]);
  }
  else{
    Serial.print("NONE");
  }
  Serial.println("\r\n");
}

void setup()
{
  /** initialize */
  myVR.begin(9600);

  Serial.begin(115200);
  Serial.println("Electrohouse Voice Recognition V3 Module\r\nControl LED sample"

  pinMode(led1, OUTPUT);
```

```
  if(myVR.clear() == 0){
    Serial.println("Recognizer cleared.");
  }else{
    Serial.println("Not find VoiceRecognitionModule.");
    Serial.println("Please check connection and restart Arduino.");
    while(1);
  }

  if(myVR.load((uint8_t)onRecord) >= 0){
    Serial.println("onRecord loaded");
  }

  if(myVR.load((uint8_t)offRecord) >= 0){
    Serial.println("offRecord loaded");
  }
}

void loop()
{
  int ret;
  ret = myVR.recognize(buf, 50);
  if(ret>0){
    switch(buf[1]){
      case onRecord:
        /** turn on Light */
        digitalWrite(light, HIGH);
        break;
      case offRecord:
        /** turn off Light*/
        digitalWrite(light, LOW);
        break;
      default:
        Serial.println("Record function undefined");
        break;
    }
    /** voice recognized */
    printVR(buf);
  }
```

```
}
```