پلتفرم اختصاصی

مهندسی کنترل

IoT BASED PROJECTS

Realization with Raspberry Pi, NodeMCU and Arduino

DR. RAJESH SINGH
DR. ANITA GEHLOT
DR. LOVI RAJ GUPTA
NAVJOT RATHOUR
MAHENDRA SWAIN
BHUPENDRA SINGH

bpb

IoT BASED PROJECTS

Realization with Raspberry Pi, NodeMCU and Arduino

DR. RAJESH SINGH
DR. ANITA GEHLOT
DR. LOVI RAJ GUPTA
NAVJOT RATHOUR
MAHENDRA SWAIN
BHUPENDRA SINGH

bpb

Iot Based Projects

Realization with Raspberry Pi, NodeMCU and Arduino

by

Dr. Rajesh Singh
Professor, Lovely Professional University

Dr. Anita Gehlot
Associate Professor, Lovely Professional University

Dr. Lovi Raj Gupta
Executive Dean, Lovely Professional University

Navjot Rathour
Assistant Professor, Lovely Professional University

Mahendra Swain
Assistant Professor, Quantum University

Bhupendra Singh
Managing Director, Schematics Microelectronics

FIRST EDITION 2020

ISBN: 978-93-89328-523

Distributors:

BPB PUBLICATIONS

20, Ansari Road, Darya Ganj

New Delhi-110002

Ph: 23254990/23254991

MICRO MEDIA

Shop No. 5, Mahendra Chambers,

150 DN Rd. Next to Capital Cinema,

V.T. (C.S.T.) Station, MUMBAI-400 001

Ph: 22078296/22078297

DECCAN AGENCIES

4-3-329, Bank Street,

Hyderabad-500195

Ph: 24756967/24756400

BPB BOOK CENTRE

376 Old Lajpat Rai Market,

Delhi-110006

Ph: 23861747

Published by Manish Jain for BPB Publications, 20 Ansari Road, Darya Ganj, New Delhi-110002
and Printed by him at Repro India Ltd, Mumbai

About the Authors

Dr. Rajesh Singh is currently associated with *Lovely Professional University* as a Professor with more than 16years of experience in academics. He has been awarded a gold medal in M.Tech from RGPV, Bhopal (M.P) India and honors in his B.E from Dr B.R. Ambedkar University, Agra (U.P) India. His area of expertise includes embedded systems, robotics, wireless sensor networks and IoTs. He has organized and conducted many workshops, summer internships, and expert lectures for students as well as faculty. He has been honored as a keynote speaker and session chair to international/national conferences, faculty development programs, and workshops. He has 27 patents in his account. He has published more than 100 research papers in referred journals/conferences and 18 books in the area of Embedded Systems and Internet of Things with reputed publishers like CRC/Taylor & Francis, Narosa, GBS, IRP, NIPA, River Publishers, Bentham Science, and RI publication. He is an editor to a special issue published by AISC book series, Springer in 2017 and 2018 and IGI global in 2019.

Under his mentorship, students have participated in national/international competitions, including *Innovative Design Challenge competition* by Texas and DST and Laureate award of excellence in robotics engineering in Madrid, Spain in 2014 and 2015. His team has been the winner of *Smart India Hackathon-2019* hardware version conducted by MHRD, Government of India for the problem statement of Mahindra & Mahindra. Under his mentorship, the student team got the *InSc award 2019* under the students' projects program. He has been awarded the *Gandhian Young Technological Innovation (GYTI) Award* as a mentor to *On-Board Diagnostic Data Analysis System-OBDAS*, appreciated under *Cutting Edge Innovation* during the *Festival of Innovation and Entrepreneurship* at the Rashtrapati Bhavan, India in 2018.

He has been awarded the *Best Researcher award* at *MANTHAN* (*Awards of Excellence*), organized by University of Petroleum and Energy Studies,

2017 and *Research appreciation award-2019* for his contribution in research from Lovely Professional University. Twice in four years, he has been awarded the *Certificate of Appreciation* from University of Petroleum and Energy Studies for his exemplary work for conducting multi-disciplinary workshops for the students. He got the *Certificate of Appreciation* for mentoring the projects submitted to the Texas Instruments Innovation challenge India design contest, from Texas Instruments in 2015 and 2018. He has been honored with the *Certificate of Excellence* from third faculty branding awards15, organized by the EET CRS research wing for excellence in professional education and Industry for the category *Award for Excellence in Research*, 2015 and young investigator award at the International Conference on Science and Information in 2012.

Dr. Anita Gehlot is currently associated with *Lovely Professional University* as an Associate Professor with more than 12 years of experience in academics. Her area of expertise includes embedded systems, wireless sensor networks and IoT. She has organized and conducted many workshops, summer internships and expert lectures for students as well as faculty. She has been honored as a keynote speaker and session chair to international/national conferences, faculty development programs and workshops. She has 24 patents in her account. She has published more than 70 research papers in referred journals/conferences and 18 books in the area of Embedded Systems and Internet of Things with reputed publishers like CRC/Taylor & Francis, Narosa, GBS, IRP, NIPA, River Publishers, Bentham Science, and RI publication. She is the editor to a special issue published by AISC book series, Springer in 2018, and IGI global in 2019.

She hasbeen awarded *Certificate of Appreciation* from University of Petroleum and Energy Studies for her exemplary work. Under her mentorship, the student team got the *InSc award 2019* under the students' projects program. She has been awarded *Gandhian Young Technological Innovation (GYTI) Award* as a mentor to *On Board Diagnostic Data Analysis System-OBDAS*, appreciated under *Cutting Edge Innovation* during the *Festival of Innovation and Entrepreneurship* at Rashtrapati Bhavan, India in 2018.

Dr. Lovi Raj Gupta is the Executive Dean, Faculty of Technology and Sciences, Lovely Professional University. He is a leading light in the field of technical and higher education in the country. His research-focused approach and an insightful innovative intervention of technology in education have won him much accolades and laurels.

He holds a PhD degree in Bioinformatics. He did his M.Tech. in Computer aided Design and Interactive Graphics from IIT, Kanpur and B.E. (Hons.) from MITS, Gwalior. Having a flair for endless learning, he has done more than 20 plus certifications and specializations online on the Internet of Things (IoT), Augmented Reality, and Gamification, from University of California at Irvine, Yonsei University, South Korea, Wharton School, University of Pennsylvania, and Google Machine Learning Group. His research interests are in the areas of Robotics, Mechatronics, Bioinformatics, IoT, AI and ML using Tensor Flow (CMLE), and Gamification.

In 2001, he was appointed as an Assistant Controller (Technology), Ministry of IT, Govt. of India by the Honorable President of India in the Office of the Controller of Certifying Authorities (CCA). In 2013, he was accorded the role in the National Advisory Board for What Can I Give Mission - Kalam Foundation of Dr APJ Abdul Kalam. In 2011, he received the MIT Technology Review Grand Challenge Award followed by the coveted Infosys InfyMakers Award in the year 2016. He has 10 patents to his account.

Ms Navjot Rathour is associated with Lovely Professional University as an Assistant Professor with more than eight years of experience in academics. She is pursuing her PhD in Electronics and Communication Engineering from Lovely Professional University. She has one patent in her account. She has published seven research papers in referred journals and conference. She has organized many summer internship and expert lectures for students. She

was awarded *Academic Honor* from Lovely Professional University in her Masters for being the University Topper.

Mahendra Swain is a PhD scholar at Lovely Professional University, Jalandhar, Punjab. He has completed his B.Tech in ECE from Centurion University of Technology and Management, Bhubaneswar, M.Tech from Lovely professional University. He has published various research articles and attended National and international conferences in the field of Embedded System and IoTs.

Bhupendra Singh is the Managing Director of Schematics Microelectronics and provides Product design and R&D support to industries and Universities. He has completed BCA, PGDCA, M.Sc. (CS), M.Tech and has more than 11 years of experience in the field of Computer Networking and Embedded systems. He has published 18 books in the area of Embedded Systems and IoT with reputed publishers like CRC/Taylor & Francis, Bentham Science, Narosa, GBS, IRP, NIPA, River Publisher, and RI publication.

Acknowledgements

We acknowledge the support from nuttyengineer.com, which provided its products in order to demonstrate and explain how the systems worked. We would like to thank the publisher for encouraging our ideas in this book and their support in helping us to manage this project efficiently.

We are grateful to the honorable Chancellor (Lovely Professional University) Ashok Mittal, Mrs. Rashmi Mittal (Pro Chancellor, LPU), and Dr. Ramesh Kanwar (Vice Chancellor, LPU) for their support and constant encouragement. We are also thankful to our family, friends, relatives, colleagues, and students for their moral support and blessings.

Dr. Rajesh Singh
Dr. Anita Gehlot
Dr. Lovi Raj Gupta
Navjot Rathour
Mahendra Swain
Bhupendra Singh

Preface

The objective of this book is to discuss the various projects based on **Internet of Things** (**IoTs**). This book comprises a total of fourteen chapters. This text is beneficial for people who want to get started with hardware-based projects using IoT.

This book is entirely based on the practical experience of the authors while undergoing projects with students and industries.

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors if any, occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Table of Contents

1. ESP8266-based Wireless Web Server
- Introduction
- Circuit diagram
- Program
- Local server/web display
- Conclusion

2. Air Pollution Meter Using Raspberry Pi
- Introduction
- System description
- Circuit diagram and connection
- Application/data logger
- Conclusion

3. Smart Garage Door
- Introduction
- Circuit diagram
- Blynkapp
- Programs
- Conclusion

4. Baggage Tracker
- Introduction
- System description
- Circuit diagram and connection
- Application/data logger
- Conclusion

5. Smart Trash Collector
- Introduction
- System description
- Circuit diagram and connection

Application/data logger
Conclusion

6. Car Parking System
Introduction
System description
Circuit diagram and connection
Application/data logger
Conclusion

7. Home Automation
Introduction
Circuit diagram
Program
Blynk app
Conclusion

8. Environmental Parameter Monitoring
IoT-based greenhouse effect monitoring system
Circuit diagram
Program
Arduino Mini program for the local server
NodeMCU program for the greenhouse receiver
ThingViewapp
Conclusion

9. Intelligent System for the Blind
Introduction
Circuit diagram and connection
Program
Conclusion

10. Sign-in to Speech Using the IoTs
Introduction
Installing dependencies on the controller platform
Creating a gesture
Conclusion

11. Windows 10 on Raspberry
- Introduction
- Raspberry Pi
- Linux kernel architecture
- Windows 10 on Raspberry Pi
  - *SD card formatter*
- Performance testing and evaluation
- Conclusion

12. Wireless Video Surveillance Robot Using Raspberry Pi
- Introduction
- Circuit diagram and connection
- Application/data logger
- Raspberry Pi and its installation
- Writing an SD card with NOOBS
- Blynkapp
- Conclusion

13. IoT-based Smart Camera
- Introduction
- Circuit diagram and connection
- Application/data logger
- Conclusion

14. IoT-based Air Quality Monitoring System Using NodeMCU
- Introduction
- Circuit diagram
- Program
- Virtuino app
- Conclusion

CHAPTER 1

ESP8266-based Wireless Web Server

The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack manufactured in 2014. This chapter discusses the open source platform, NodeMCU with ESP8299 WiFi SoC. The circuit diagram and interfacing steps are discussed to understand the working of NodeMCU to control the home appliances with the web server.

Introduction

To understand the interfacing of NodeMCU, a system is designed to control the home appliances with the local server. NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC. NodeMCU, as shown in *Figure 1.1*, provides access to the GPIO (General Purpose Input/Output):

Figure 1.1: *NodeMCU*

For developing purposes, the pin description is shown in following *Table 1.1*:

IO index	0	1	2	3	4	5	6	7	8	9	10	11	12
ESP8266 pin	GPIO16	GPIO5	GPIO4	GPIO0	GPIO2	GPIO14	GPIO12	GPIO13	GPIO15	GPIO3	GPIO1	GPIO9	GPIO10

Table 1.1: *ESP8266 pin description*

The home appliances can be controlled with NODEMCU through relays. *Figure 1.2* shows the detailed block diagram of the system and *Table 1.2* shows the component list required for the system. It shows the block diagram of the system. It comprises NodeMCU, LCD, power supply, and electrical appliances:

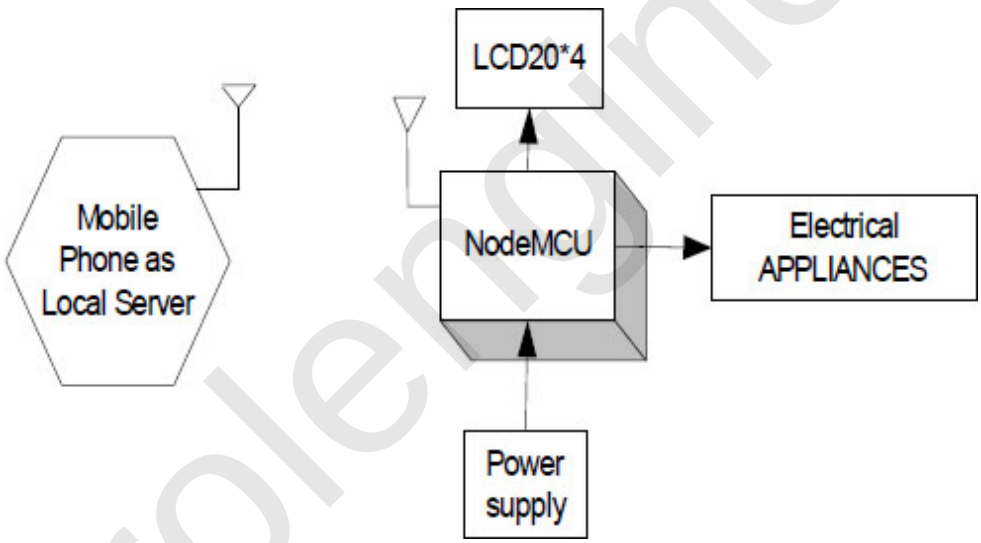

Figure 1.2: *Block Diagram*

Table 1.2 shows the list of components required to design the system:

Component/Specification	Quantity
Power supply 12V/1Amp	1
Node MCU	1
Solid state relay board	4
Extension board for four appliances	4
Power supply extension	1

ISP programmer	1
LCD16*2	1
LCD patch	1
+5V power supply	1

Table 1.2: *Components List*

Circuit diagram

The circuit diagram of the system is shown in *Figure 1.3* and the description of the system is as follows:

- NodeMCU D0 pin is attached to RS pin of LCD
- RW pin of LCD is connected to Ground
- NodeMCU D1 pin is attached with E pin of LCD
- NodeMCU D2 pin is attached with D4 pin of LCD
- NodeMCU D3 is attached with D5pin of LCD
- NodeMCU D4 pin is attached with D6 pin of LCD
- NodeMCU D5 pin is attached with D7 pin of LCD
- Pin 1 and pin 16 of LCD is connected with Ground
- Pin 2 and pin 15 of LCD is connected with +Vcc

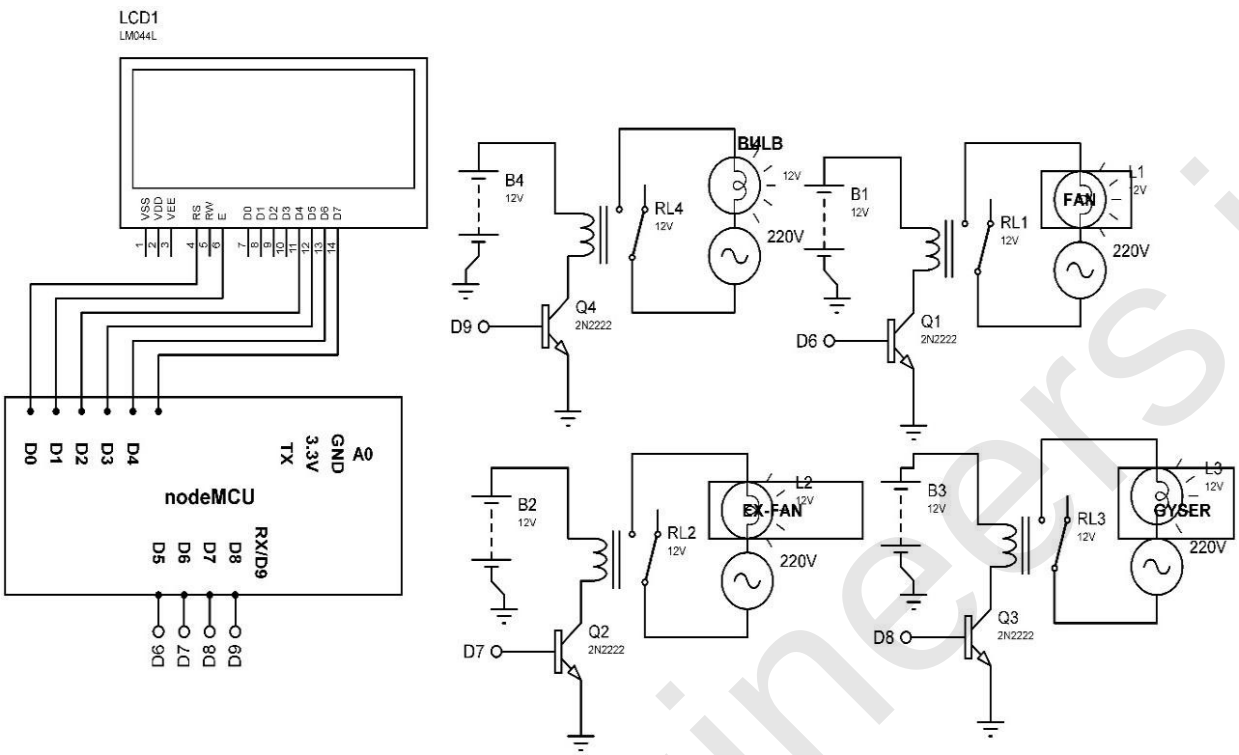

Figure 1.3: *Circuit diagram*

Program

Let's see the following program:

```
//for hot spot
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
int Analog = A0;

#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(D0, D1, D2, D3, D4, D5);

//for hotspot
MDNSResponder mdns;
const char* ssid = "ESPServer_RAJ"; // add your credentials here
const char* password = "RAJ@12345";
String webString="";
```

```
ESP8266WebServer server(80);
String webPage = "";
String web="";
int pin1 = D6; // assign integer to pin D6
int pin2 = D7; // assign integer to pin D7
int pin3 = D8; // assign integer to pin D8
int pin4 = D0; // assign integer to pin D0
int TEMP_level=0; // assume variable
void setup()
{
 lcd.begin(20, 4); // initialize LCD
 lcd.print("robot Monitoring"); // print string on LCD
 webPage +="<h2>ESP8266 Web Server new</h2><p>TEMP METER <a
 href=\"TEMP\"><button> TEMPERATURE (oC)</button></a></p>";//
 for temperature
 webPage += "<p>BULB-STATUS <a href=\"BULBON\">
 <button>ON</button></a> <a href=\"BULBOFF\">
 <button>OFF</button></a></p>";
 webPage += "<p>FAN-STATUS <a href=\"FANON\"><button>ON</button>
 </a> <a href=\"FANOFF\"><button>OFF</button></a></p>";
 webPage += "<p>EXHAUST FAN-STATUS <a href=\"EXHAUSTFANON\">
 <button>ON</button></a> <a href=\"EXHAUSTFANOFF\">
 <button>OFF</button></a></p>";
 webPage += "<p>GYSER-STATUS <a href=\"GYSERON\">
 <button>ON</button></a> <a href=\"GYSEROFF\">
 <button>OFF</button></a></p>";
 webPage += "<p>ALLOFF-STATUS <a href=\"GYSERON\">
 <button>ON</button></a> <a href=\"ALLOFF\">
 <button>OFF</button></a></p>";

 // preparing GPIOs
 pinMode(pin1, OUTPUT);// assign pin as an OUTPUT
 digitalWrite(pin1, LOW); // make pin to LOW
 pinMode(pin2, OUTPUT); // assign pin as an OUTPUT
 digitalWrite(pin2, LOW); // make pin to LOW
 pinMode(pin3, OUTPUT); // assign pin as an OUTPUT
 digitalWrite(pin3, LOW); // make pin to LOW
 pinMode(pin4, OUTPUT); //assign pin as an OUTPUT
```

```
digitalWrite(pin4, LOW); // make pin to LOW
delay(1000); // delay of 1000 mSec
Serial.begin(115200); // Initialize serial communication
WiFi.begin(ssid, password); // initialize Wi-Fi
Serial.println("”); // print string on serial

// Wait for connection
while (WiFi.status() != WL_CONNECTED)
{
 delay(500); // delay of 500mSec
 Serial.print(“.”); // print string on Serial
}
Serial.println(“”);// print string on Serial
Serial.print(“Connected to “); // print string on Serial
Serial.println(ssid);// print ssid on serial
Serial.print(“IP address: “); // print string on Serial
Serial.println(WiFi.localIP()); // print IP on serial
if (mdns.begin(“esp8266”, WiFi.localIP()))
{
 Serial.println(“MDNS responder started”); // print string on
 Serial
}
server.on(“/”, []()
{
 server.send(200, “text/html”, webPage);
}

/*************************************************************
************************/

server.on(“/TEMP”, []()
{
 get_TEMP();
 webString=”TEMPERATURE: “+String((float)TEMP_level)+”oC”;
 server.send(200, “text/plain”, webString); // send to
 someone’s browser when asked
}
server.on(“/BULBON”, []()
{
```

```
 server.send(200, "text/html", webPage);
 digitalWrite(pin1, HIGH); // make pin to HIGH
 digitalWrite(pin2, LOW); // make pin to LOW
 digitalWrite(pin3, LOW); // make pin to LOW
 digitalWrite(pin4, LOW); // make pin to LOW
 lcd.clear(); // clear the contents of LCD
 lcd.setCursor(0, 1); // set cursor on LCD
 lcd.print("BULB ON "); // print string on LCD
 delay(1000); // delay of 1000mSec
}
server.on("/BULBOFF", []()
{
 server.send(200, "text/html", webPage);
 digitalWrite(pin1, LOW); // make pin to LOW
 digitalWrite(pin2, LOW); // make pin to LOW
 digitalWrite(pin3,LOW); // make pin to LOW
 digitalWrite(pin4, LOW); // make pin to LOW
 lcd.clear(); // clear contents of LCD
 lcd.setCursor(0, 1); // set cursor on LCD
 lcd.print("BULB OFF"); // print string on LCD
 delay(1000); // delay of 1000 mSec
}
server.on("/FANON", []()
{
 server.send(200, "text/html", webPage);
 digitalWrite(pin1, LOW); // make pin to LOW
 digitalWrite(pin2, HIGH); // make pin to HIGH
 digitalWrite(pin3,LOW); // make pin to LOW
 digitalWrite(pin4, LOW); // make pin to LOW
 lcd.clear(); // clear the contents of LCD
 lcd.setCursor(0, 1); // set cursor on LCD
 lcd.print("FAN ON "); // print string on LCD
 delay(1000); //delay of 1000 mSec
}
server.on("/FANOFF", []()
{
 server.send(200, "text/html", webPage);
```

```
 digitalWrite(pin1, LOW); // make pin to LOW
 digitalWrite(pin2, LOW); // make pin to LOW
 digitalWrite(pin3,LOW); // make pin to LOW
 digitalWrite(pin4, LOW); // make pin to LOW
 lcd.clear();// clear the contents of LCD
 lcd.setCursor(0, 1); // set cursor on LCD
 lcd.print("FAN OFF "); // print string on LCD
 delay(1000); //delay of 1000 mSec
}
server.on("/EXHAUSTFANON", []()
{
 server.send(200, "text/html", webPage);
 digitalWrite(pin1, LOW); // make pin to LOW
 digitalWrite(pin2, LOW); // make pin to LOW
 digitalWrite(pin3,HIGH); // make pin to HIGH
 digitalWrite(pin4, LOW); // make pin to LOW
 lcd.clear(); // clear the contents of LCD
 lcd.setCursor(0, 1); // set cursor on LCD
 lcd.print("EXHAUST FAN ON "); // print string on LCD
 delay(1000); // delay of 1000 mSec
}
server.on("/EXHAUSTFANOFF", []()
{
 server.send(200, "text/html", webPage);
digitalWrite(pin1, LOW); // make pin to LOW
 digitalWrite(pin2, LOW); // make pin to LOW
 digitalWrite(pin3,LOW); // make pin to LOW
 digitalWrite(pin4, LOW); // make pin to LOW
 lcd.clear();// clear the contents of LCD
 lcd.setCursor(0, 1); // set cursor on LCD
 lcd.print("EXHAUST FAN OFF "); // print string on LCD
 delay(1000); // delay of 1000 mSec
}
server.on("/GYSERON", []()
{
 server.send(200, "text/html", webPage);
 digitalWrite(pin1, LOW); // make pin to LOW
```

```
    digitalWrite(pin2, LOW); // make pin to LOW
    digitalWrite(pin3,LOW); // make pin to LOW
    digitalWrite(pin4, HIGH); // make pin to HIGH
    lcd.clear(); // clear the contents of LCD
    lcd.setCursor(0, 1); // set cursor on LCD
    lcd.print("GYSER ON "); // print string on LCD
    delay(1000); // delay 1000 mSec
  }
  server.on("/GYSEROFF", []()
  {
    server.send(200, "text/html", webPage);
  digitalWrite(pin1, LOW); // make pin to LOW
    digitalWrite(pin2, LOW); // make pin to LOW
    digitalWrite(pin3,LOW); // make pin to LOW
    digitalWrite(pin4, LOW); // make pin to LOW
    lcd.clear();// clear the contents of LCD
    lcd.setCursor(0, 1); // set cursor on LCD
    lcd.print("GYSER OFF"); // print string on LCD
    delay(1000); // delay 1000 mSec
  }
  server.begin(); // initialize server
  Serial.println("Congrats Boss, Your HTTP server started"); //
  print string on Serial
}

void loop()
{
  server.handleClient();
  get_TEMP(); // call function
  lcd.clear(); // clear the contents of LCD
  lcd.setCursor(0, 0); // set cursor on LCD
  lcd.print(TEMP_level); // print the value on LCD
  delay(500); // delay of 500 mSec
}

void get_TEMP()
{
  int TEMP_level1= analogRead(Analog); // read analog value
```

```
 TEMP_level=TEMP_level1/2; // scale factor to get temperature

}
```

Local server/web display

Upload the program mentioned in the preceding section and check the serial port for its IP address. Open the IP address and it will show the webpage, as shown in *Figure 1.4*:

Figure 1.4: *Application view*

Conclusion

This chapter concludes the basics of NodeMCU and explains the circuit diagram and program to control home appliances.

CHAPTER 2

Air Pollution Meter Using Raspberry Pi

Air quality is a global challenge for governments, regulators, city administrators, and citizens. Multi-billion dollar sums are invested by governments in policies and solutions all around the world to improve the air quality and they are empowering cities to tackle air pollution. In order to implement effective policies and interventions, there is an increasing focus on understanding the levels and causes of air pollution.

As of today, air quality monitoring is performed by large, expensive scientific instruments, installed and maintained professionally, at a relatively small number of fixed locations. For example, in India, there are around 570 monitoring stations, which boils down to 5766.67 square km per station on average, and this distribution is not even uniform [1].

According to the Indian government report, cities that share similar air quality as Delhi are Faridabad, Jodhpur, Lucknow, Moradabad, Muzaffarnagar, Pali, and Varanasi; out of which Lucknow has four air quality monitoring stations and rest all have only one station [3].

For air quality monitoring, the **Air Quality Index** (**AQI**) is important. It is an index prepared by government institutions to categories risk of air pollution. Lower AQI value represents pleasant air quality [10]. The various algorithms and hardware descriptionsare used to read the various gas sensors to monitor and control air quality [11-14].

Alison J. Greig did an extensive study on the relationship between the count and mass of air born particles. He successfully pointed out that on an average composition of particles, fine particles (particle size 2.5 µm or less) amount to a large count but their total mass is very low; however, coarse particles (particle size 2.5µm to 10µm) have very less count but amount to a very large total mass [4]. The extent to which a person is exposed to pollutants and what kind of pollutants depends a lot on the activity one does and the ventilation system of the building. A person who spends more time in an indoor environment is less likely to be affected by outside pollutants, as it's difficult to penetrate through the closed setup and these pollutants end up decaying and deposit well in time. People who spend more time indoors are exposed to a different set of pollutants which pose a similar threat to health like CO and smoke from the improper cooking stove,

alcohol vapors from perfumes and toiletries and more [5]. *Alexandra Schieweck* and *Erik Uhde* proposed a way to create smart homes which can help control air quality inside it. Their idea is to install sensors outside and inside the house, and using the data from both sources, home ventilation is controlled [6]. Another take on this to use a random forest approach for predicting air quality using meteorology data, real-time traffic information, road information [7]. Development of boards such as AIRQino, which is essentially an Arduino shield fully equipped with air quality sensors which can monitor parameters such as temperature, relative humidity, CO, CO2, NO2, O3, VOC, PM2.5 and PM10 [8]. Akira Tiele et al. discussed a very efficient and low-cost indoor environment quality management unit which is portable and displays the recorded **Indoor Environment Quality** (**IEQ**) parameters on an OLED display present on the device. Table1 shows the IEQ index scoring system for various sensor like a light sensor, CO2 and CO sensor, temperature and humidity sensor, sound level sensor, and dust sensor [9].

IEQ Parameter	Good	Average	Poor	Bad
Humidity	40-50%	50-60%	60-70%	>70%
Temperature	20-24°C	16-20°C	24-26°C	>26°C
PM2.5	$(0\text{-}10)10^{-6}g/m^3$	$(10\text{-}15)10^{-6}g/m^3$	$(15\text{-}35)10^{-6}g/m^3$	$(35)10^{-6}g/m^3$
PM10	$(0\text{-}50)10^{-6}g/m^3$	$(50\text{-}80)10^{-6}g/m^3$	$(80\text{-}150)10^{-6}g/m^3$	$(150)10^{-6}g/m^3$
VOC	0-200 ppb	200-350 ppb	350-500 ppb	>500 ppb
CO_2	350-500 ppm	500-1000 ppm	1000-5000 ppm	>5000 ppm
CO	0-3 ppm	3-8 ppm	8-10 ppm	>10 ppm

Table 2.1: *IEQ index scoring chart*

Introduction

In today's fast-paced and industrialized society, air quality is becoming a topic of prominent concern. In India, about 12 million people are affected by **Chronic Obstructive Pulmonary Disease** (**COPD**) and about 1.5 million people in India die each year due to diseases caused or fostered by air pollution, that is, one-sixth of all Indian deaths. Even now, when the numbers are so high, there is no cheap and simple solution to monitor the indoor air quality and this idea is also not popular. This section proposes a simple Raspberry Pi controlled device which records quality of air present indoor. These similar devices can be distributed all

across the house and we can upload the data to an online server. Such a network of smart devices will result in complete control of air quality present inside the house for the user to analyze and enable him to take appropriate action when any parameter reaches an alarming level.

System description

Figure 2.1 shows the Raspberry Pi 3 Model B that serves well to implement the air quality sensor node with its quad. It has core 900Mhz process with Broadcom BCM2836:

- CPU: 4× Cortex-A53900 MHz
- USB port: 4
- On-Board Network: Ethernet, 802.11n wireless
- Bluetooth 4.1
- GPIO: 17

Figure 2.1: *Raspberry Pi*

The characteristics of Raspberry Pi are as follows and its comparison with other SBCs is shown in *Table 2.2*:

Parameters/Board	Raspberry Pi 3	Arduino	Intel Galileo	Beagle Bone
CPU	900 MHz quad core Boradcom BCM2836	ATMEGA 8, ATMEGA 1280	400 MHz single core Intel Quark X1000	AM3359 1GHz ARM Cortex -A8
RAM	1 GB	16 to 32 KB	SRAM: 512 KB DRAM: 256 MB	512MB DDR3 RAM
POWER	10W	5W	15W	15W

Operating System	Raspbian, Pidora, ARCH Linux ARM, Ubuntu MATE	NA	Arduino Linux Distribution for Galileo	Android, Debian, Fedora, Ubuntu, Yocto

Table 2.2: *Comparison of Raspberry Pi with other SBCs*

The preceding table shows the comparison among various boards with respect to parameters like CPU, RAM, power, and OS. The brief description of the components used in the system is as follows:

- **Arduino**: The Raspberry Pi doesn't come with any analogue pins. To read analogue sensors, we need to integrate Arduino UNO with Raspberry Pi using Pyfirmata. It is an ATMEGA 328P run microcontroller with 16MHz clock speed.
- **MQ7 sensor**: It is a simple **Carbon Monoxide** (**CO**) sensor, which is capable of detecting COgas concentrations anywhere from 10 to 10000ppm. It is an analogue sensor with the operating temperature range of -10 to 50°C.
- **MQ3**: This sensor is useful in sensing gas leakages. It works well for detecting Alcohol vapors, Benzine, Methane, and Hexane. Its sensitivity can be easily adjusted using the potentiometer present on board.
- **MQ2**: This is a common smoke sensor, and is also used for gas detection for gases such as LPG. Like MQ3, its sensitivity can also be adjusted using the onboard potentiometer.
- **DHT11 sensor**: It is a very basic and cheap digital sensor that measures temperature and humidity in the air around it. It houses a capacitive humidity sensor and thermistor to measure temperature around it.
- **GP2Y1030AU0F sensor**: It is a dust sensor. It utilizes an optical sensing technique, that is, it detects the light reflected off the dust in the air and due to its high sensitivity, it calculates the amount of dust present in the air. The operating temperature ranges from -10° to +65° and the operating voltage ranges from 4.5V to 5.5V. It is also capable of distinguishing PM2.5 from PM10.

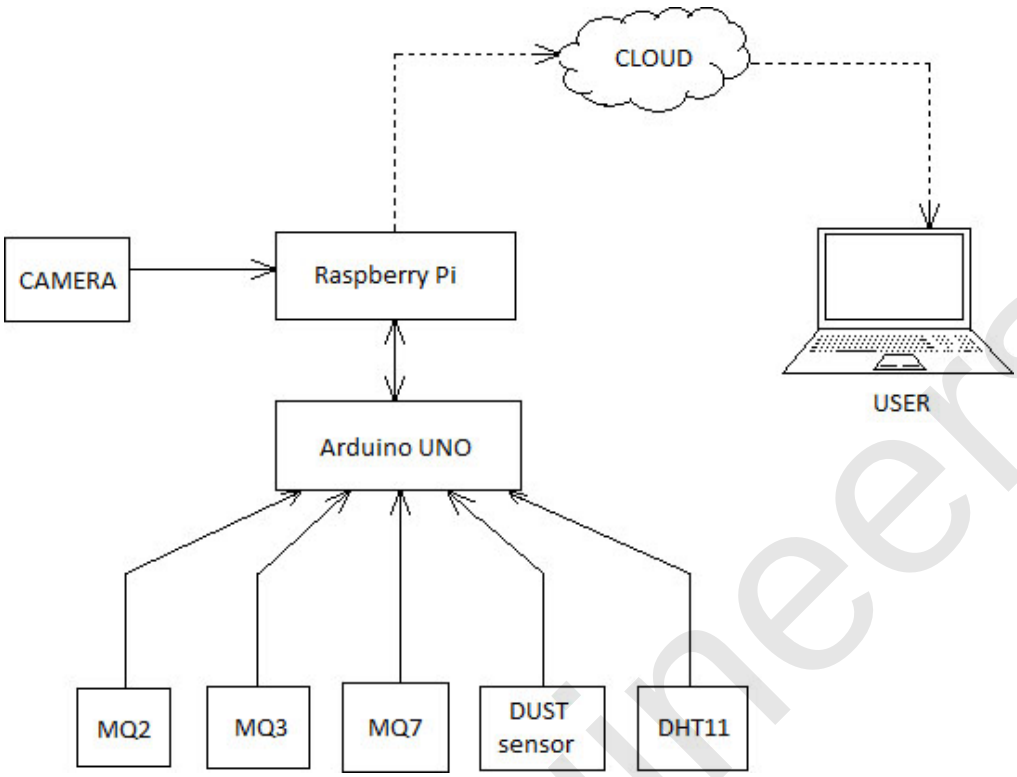

Figure 2.2: *Block diagram of air quality monitoring Node*

The *Figure 2.2* shows the air quality monitoring node which has Arduino and Raspberry Pi used as computing platforms using the Pyfirmata protocol and access the sensors using Arduino because the analogue data obtained from different sensors can easily be interpreted by Arduino. MQ2 will sense the amount of smoke present in the air and MQ3 will record the amount of alcohol vapor while MQ7 will sense the amount of CO present in the air and send it to Arduino. Similarly, DHT11 will measure the temperature and humidity present in the air and the dust sensor will measure the amount of dust present in the air and send it to Arduino. Raspberry Pi will request and collect data from Arduino. Pi will also take pictures of the room when the dust level crosses a threshold level. All the data collected by Pi using Arduino is transferred to the cloud from where the status can be accessed via any mobile device [15].

Circuit diagram and connection

Figure 2.3 shows the circuit diagram of the air quality monitoring sensor node and which comprises AC supply which is converted to 12V DC. Further, 12V DC is converted to 5V DC to supply power to Raspberry Pi and other peripherals. One camera module is connected to Pi through the designated camera port on Raspberry Pi that will take pictures of the room if the dust amount sensed by the

dust sensor crosses the threshold. The Arduino UNO microcontroller is connected to Pi via one USB port available on Pi to facilitate the analogue input for Pi:

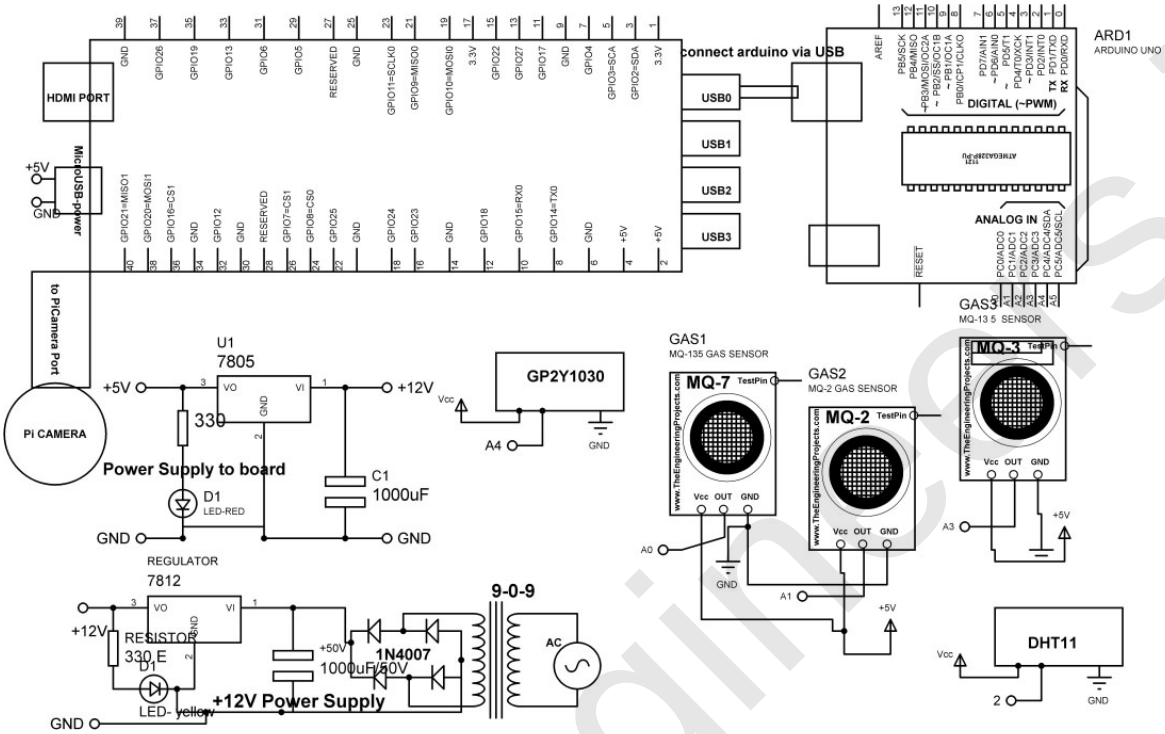

Figure 2.3: *Circuit diagram of the air quality monitoring sensor node*

Application/data logger

```
import sys#import Library
import urllib2# import Library
import pyfirmata# import Library
import Adafruit_DHT# import Library
from time import sleep# import Library
import RPi.GPIO as GPIO# import Library
from picamera import PiCamera# import Library

board = pyfirmata.Arduino('/dev/ttyUSB0')#recognize the serial
communication port
ledPower = board.get_pin('d:12:o')# assign digital pin 12 to an
output
measurePin = board.get_pin('a:5:i')#assign digital pin A5 to an
input
coPin = board.get_pin('a:0:i')# assign digital pin A0 to an input
it = pyfirmata.util.Iterator(board)# use iterator
```

```
it.start()# start iterator
measurePin.enable_reporting()#enable pin
coPin.enable_reporting() # enable pin

camera = PiCamera()# call camera
photoCount = 0 # assume variable
timer = 5 #assume variable
previousDustValue = 0 #assume variable

samplingTime = 0.000280
deltaTime = 0.000040
sleepTime = 0.009680
coMeasured = 0
voMeasured = 0
calcVoltage = 0
dustDensity = 0
RelativeHumidity = 0
Temperature = 0
myAPI = 'SRWU0GUE90JLEOP2' # write API key from thingspeak.com
baseURL = 'https://api.thingspeak.com/update?api_key=%s' % myAPI #
URL of thingspeak
def getTempHum(): # function for DHT11
RelativeHumidity, Temperature =
Adafruit_DHT.read_retry(Adafruit_DHT.DHT11,17)
def getDustData(): # function for dust sampler
 ledPower.write(0) # make pin to LOW
 sleep(samplingTime) # delay
 voMeasured = measurePin.read()
 sleep(deltaTime) # make pin to LOW
 ledPower.write(1) # make pin to HIGH
 sleep(sleepTime) # make pin to LOW

 calcVoltage = voMeasured * (5.0) # scaling
 dustDensity = (0.17) * calcVoltage - (0.1)

 if (dustDensity < 0):
  dustDensity = 0.00

def takePicture(): # function to take picture
 if (dustDensity != previousDustValue):
  camera.start_preview() # start camera preview
  sleep(2) # delay
```

```
    camera.capture('/home/pi/Desktop/image%s.jpg' % photoCount) #
    capture image
    camera.stop_preview() # stop camera preview
    photoCount += 1
    previousDustValue = dustDensity
    timer = 5
  else:
    if( timer == 0):
      camera.capture('/home/pi/Desktop/image%s.jpg' % photoCount) #
      capture image
    photoCount += 1
    timer = 5
  else:
    timer -= 1

def getCo():
  coMeasured = coPin.read() # read sensor pin
  coMeasured = coMeasured * 1000

def getStringFormat(): # call function
    return(str(Temperature), str(RelativeHumidity), str(dustDensity),
    str(coMeasured)))

while True:
  getTempHum() # call function
  getCo() # call function
  getDustData() # call function

  if (dustDensity >= 0.5):
    takePicture() # call function to take picture

  T,RH,D,CO = getStringFormat()
  f=urllib2.urlopen(baseURL+'&field1=%s&field2=%s&field3=%s&field4=%s
  ' % (T,RH,D,CO))
  print f.read() # read function
  f.close() # close function
  sleep(15) # delay of 15 sec
```

The main server is **ThingSpeak.com**. Follow the given steps to create an account and check the data on cloud. *Figure 2.4* shows the window of ThingSpeak and *Figure 2.5* shows the channel created inside ThingSpeak.

Steps to create a channel:

1. Sign in to ThingSpeak by creating a new MathWorks account.
2. Click on **Channels| My Channels**.

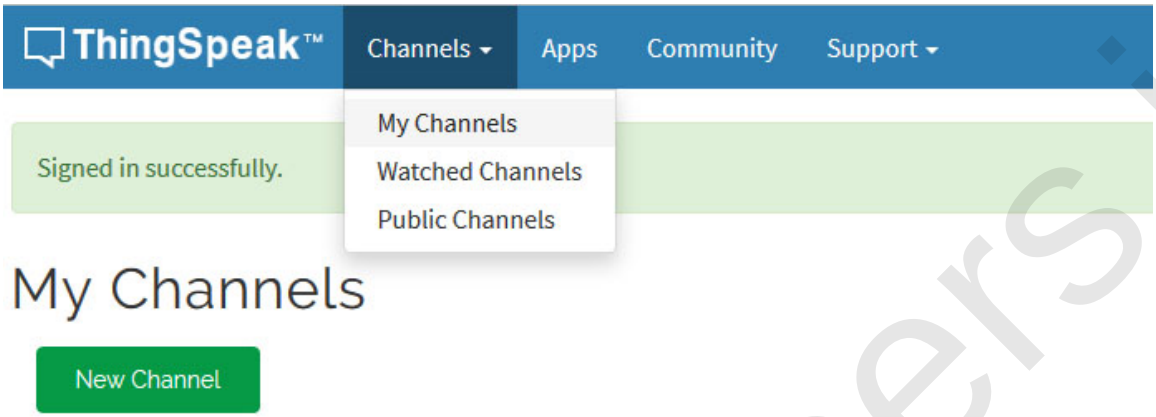

Figure 2.4: *Window for ThingSpeak*

3. Click on **New Channel**.

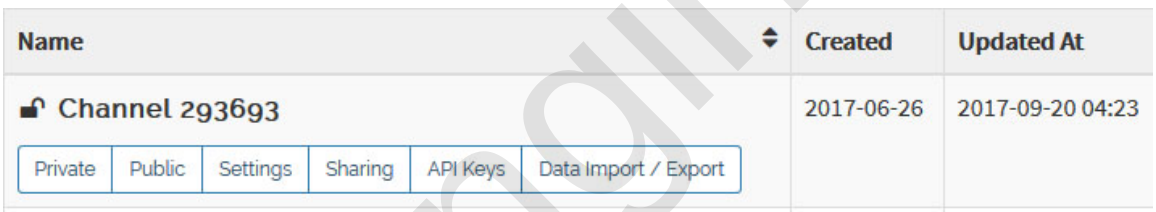

Figure 2.5: *New channel in my channels*

4. Check the boxes next to Fields 1–1. Enter the channel setting and click on **Save Channel** at the bottom of the settings.
5. Check the API write key (this key needs to be written in the program for the local server).

The system is able to measure temperature, humidity, CO, VOC (volatile organic compounds) and also dust particles in the air, PM2.5, PM10 as shown in *Figure 2.6* and *Figure 2.7*. The data from sensors will be displayed on the screen with real-time updating and the results will be uploaded to the cloud through the internet using Raspberry Pi 3 model B+:

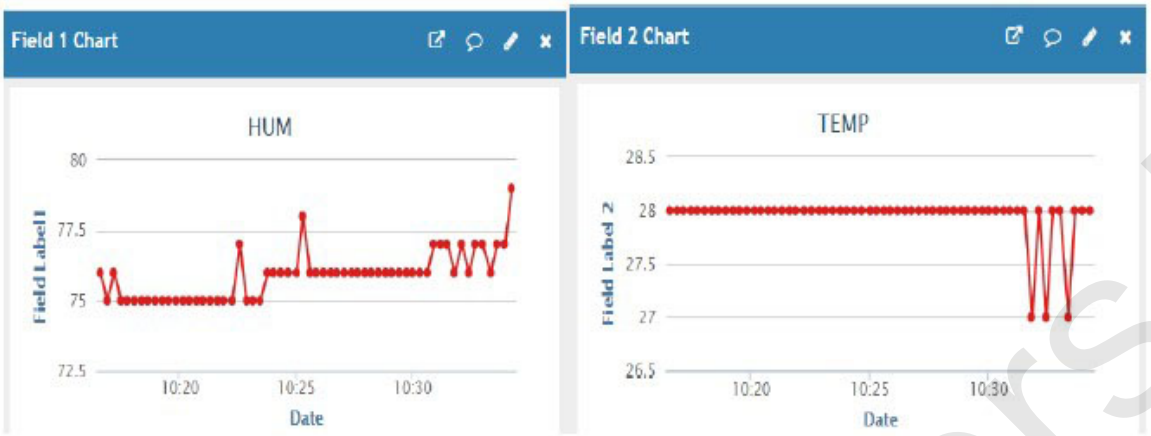

Figure 2.6: *Humidity and temperature sensors data on the cloud server*

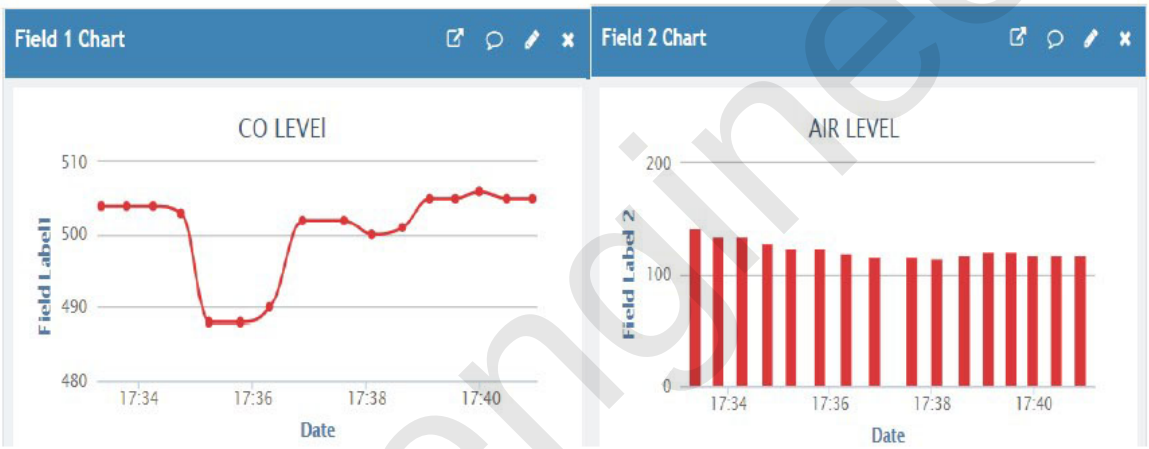

Figure 2.7: *View of CO and PM10 sensors data on the cloud server*

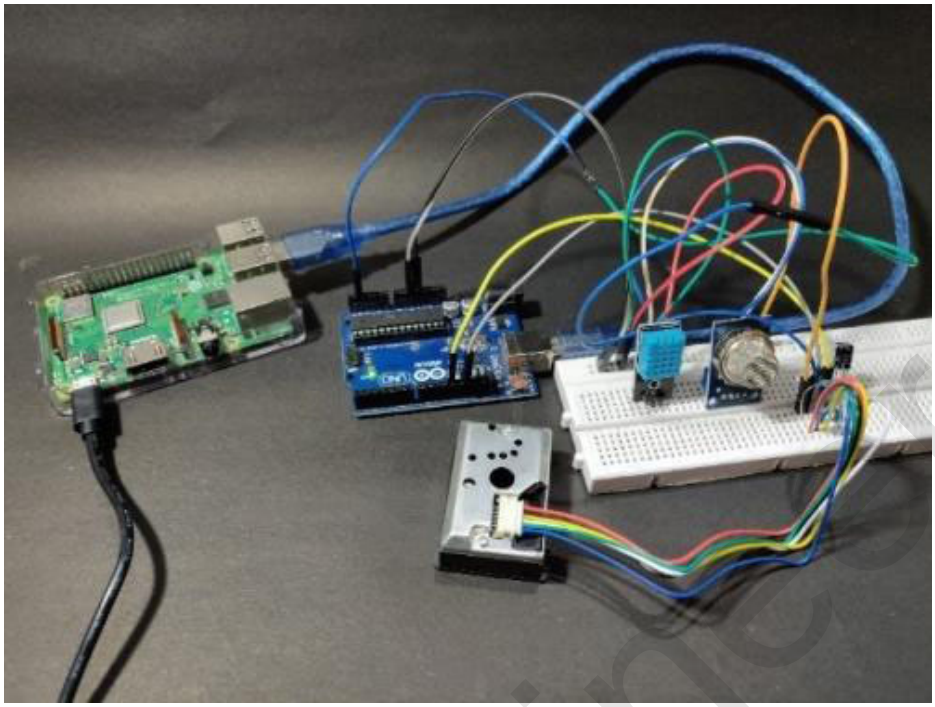

Figure 2.8: *View of the setup of the air quality sensor node*

Note-remove one figure. No lead is required. All figures are our own figures.

Figure 2.8 shows the hardware setup of the air quality sensor node. The stored data is very useful for comparing statistics with the ideal AQI and IEQ index set forth by the government. It also helps inhabitants to realize what actions or activities escalated the level of harmful gases and how it varies with temp and humidity. Nowadays, many companies are producing air purifiers for indoor systems to make them efficient and cost effective. For these products, this system helps a lot to identify how much purification is needed and how much energy needs to be consumed to purify the air.

Conclusion

This chapter concludes the discussion on the air quality monitoring system with the help of sensors. The system is designed with the help of Raspberry Pi and Arduino. It helps to understand the interfacing of Raspberry Pi and Arduino to design an application.

CHAPTER 3

Smart Garage Door

The Garage doors are used more like main doors of the house. The poor layout of the house tracks enough amount of dirt via the main entrance. The scenario gets worse in the western coastal regions of Canada when it comes to the rainy season. It would be better to have a remote control that is connected with Wi-Fi. Such remote controls can be used remotely if connected with the server. An Android app can be used that communicates with the ESP8266-based customized board.

Introduction

The project comprises the breakout board for NodeMCU, NodeMCU, power supply adaptor, motion sensor, relays, and display devices. The main objective is to provide the switching for the door locks and control the switching also. There are two door locks which are being controlled by machine 1 and machine 2 of the garage remotely. *Figure 3.1* shows the block diagram of the entire system that will help to monitor the locks via the smart Android app and shows the communication also:

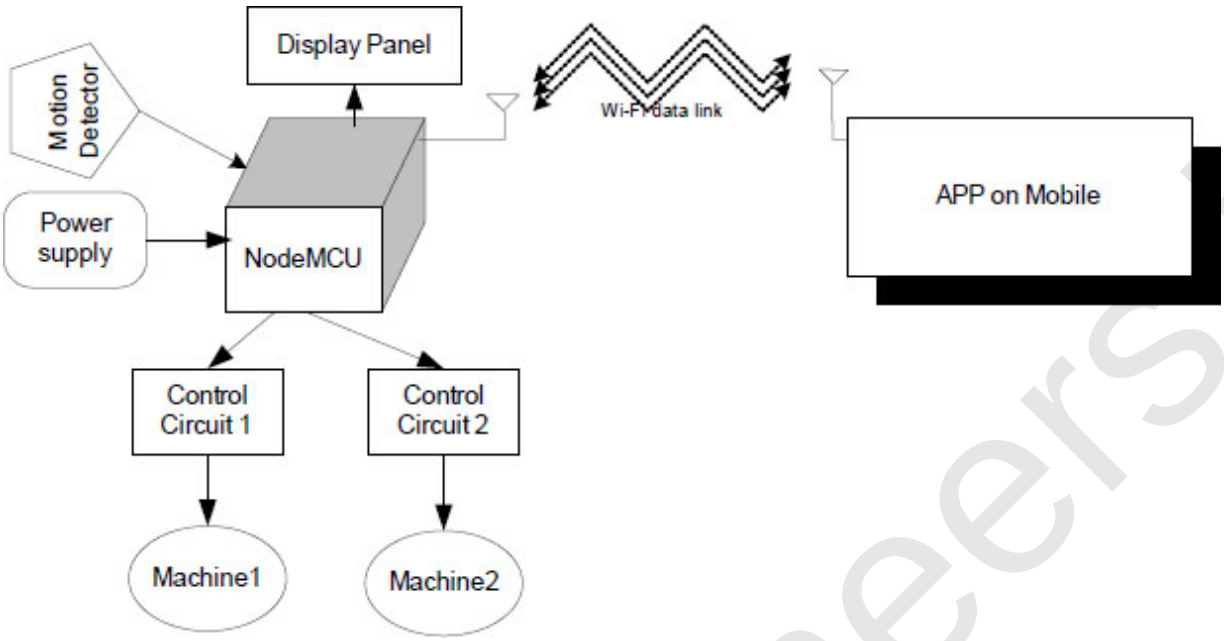

Figure 3.1: *Block diagram of the system*

The list of components required to build the system is shown in *Table 3.1*:

Component	**Quantity**
Power supply 12V/1Amp	1
NodeMCU	1
Jumper wire M-M	20
Jumper wire M-F	20
Jumper wire F-F	20
Power supply extension (To get more +5V and GND)	1
Level converter to 12V to 5V,3.3V	1
Motion sensor	1
Two relay board	1
LCD20*4	1
LCD breakout board/patch	1

Table 3.1: *Components list*

Note: All components are available at www.nuttyengineer.com.

Circuit diagram

To develop the hardware design, the components need to be connected as per the given description:

1. +5V and GND pins of NodeMCU are connected to +5V and GND pins of the power supply.
2. Pin 1 and pin 16 of LCD are connected to GND of the power supply.
3. Pin 2 and pin 15 of LCD are connected to +5V of the power supply.
4. Fixed terminals of 10K POT are connected to +5V and GND of the power supply and the variable terminal to pin3 of LCD.
5. Pin D1, GND, and pin D2 of NodeMCU are connected to pin 4 (RS), pin 5(RW), and pin 6(E) of LCD, respectively.
6. Pin D3, pin D4, pin D5, and pin D6 of NodeMCU are connected to pin 11 (D4), pin 12(D5), pin 13 (D6), and pin 14(D7) of LCD, respectively.
7. +Vcc, GND, and OUT pins of the motion sensor are connected to +5V, GND, and D7 pins of NodeMCU.
8. D7 and D8 pins of NodeMCU are connected to provide input to the solid-state relay.

Figure 3.2 shows the circuit diagram of the system:

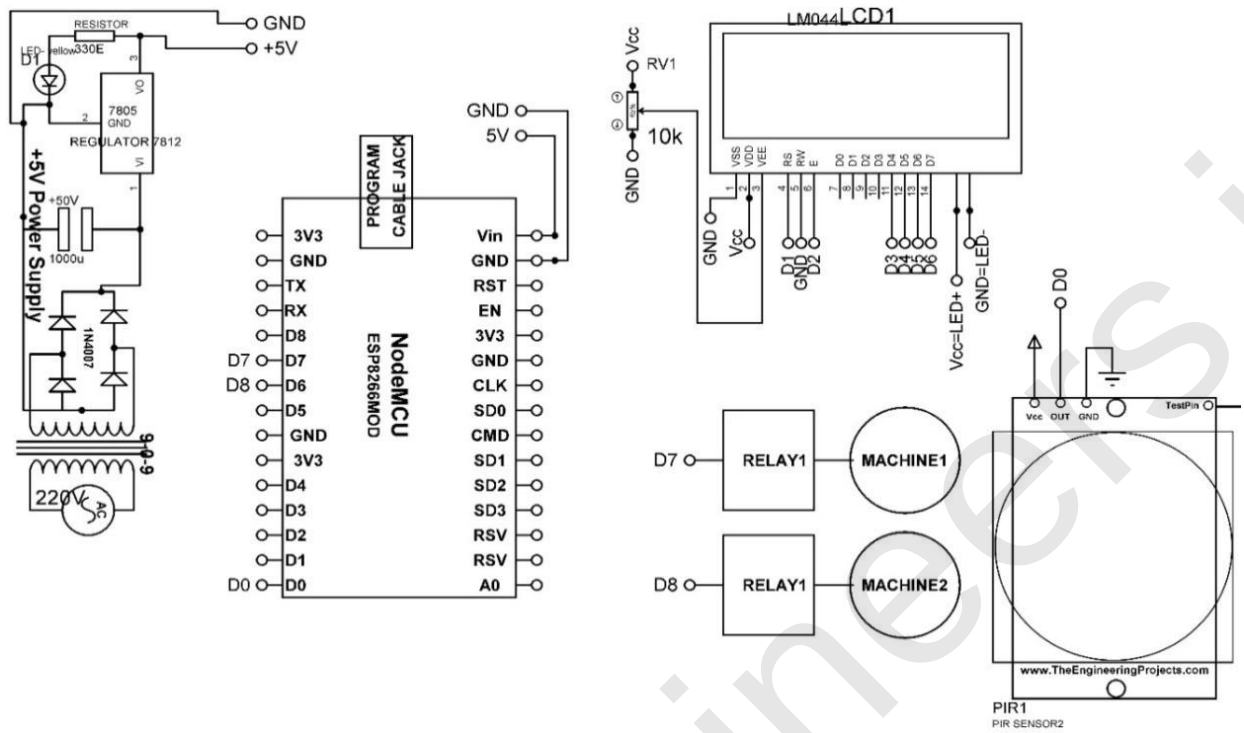

Figure 3.2: *Circuit diagram of the system*

[Figure 3.3](#) shows the circuit diagram of the solid state relay connections:

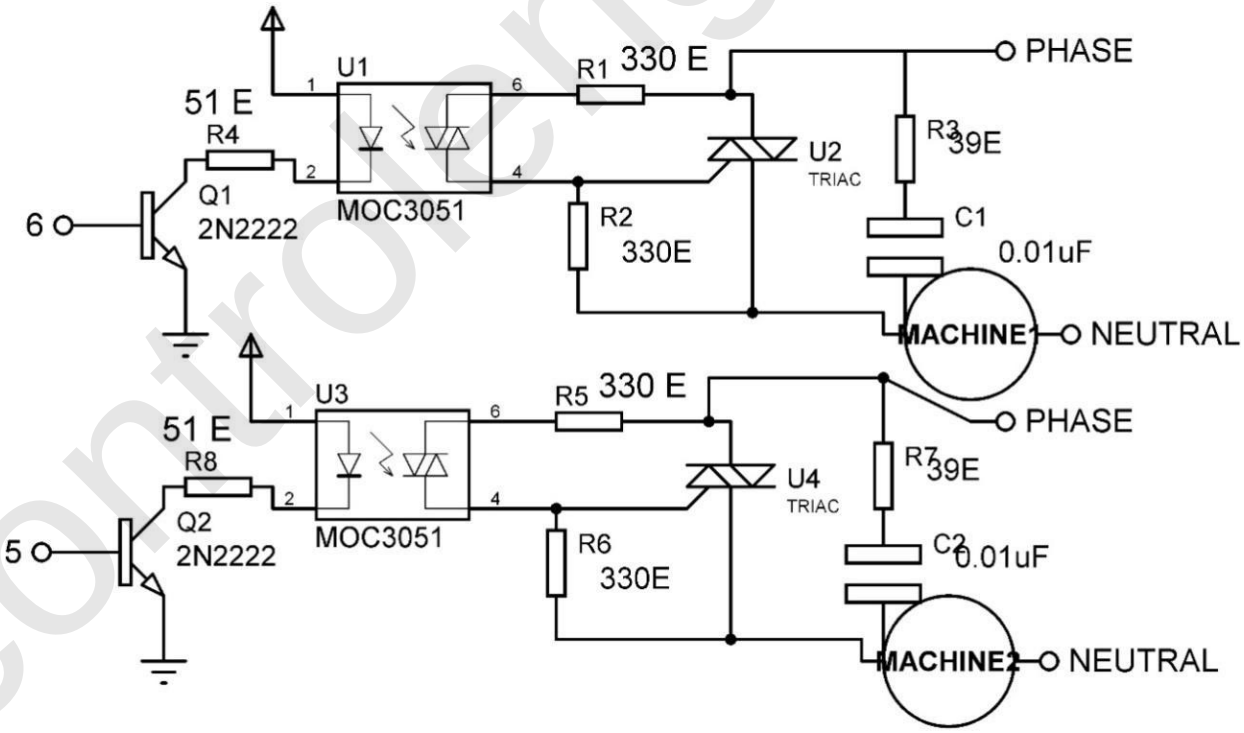

Figure 3.3: *Circuit diagram of the solid state relay (SSR)*

Blynk App

Blynk is an iOS and Android platform used to design a mobile app. The following steps are used to design the Blynk app:

1. Download and install the Blynk app for your mobile Android or iPhone from http://www.blynk.cc/getting-started/.
2. Create a Blynk account.
3. Create a new project. Click on +to createa new project and choose the theme dark (black background) or light (white background) and click on create.
4. Auth token is a unique identifier which will be received on the email address of the user, provided at the time of making the account. Save this token as this is required to be copied in the main program of the receiver section.
5. Select the device to which the smartphone needs to communicate, for example, ESP8266 (NodeMCU).
6. Open the widget box and select the components required for the project.
7. Tap on the widget to get its settings, and select virtual terminals as V1, V2, V3, V4, etc. for each buttons, which need to be defined later in the program.
8. After completing the widget settings, run the **`project.Front`** end of app for the system, as shown in *Figure 3.4*:

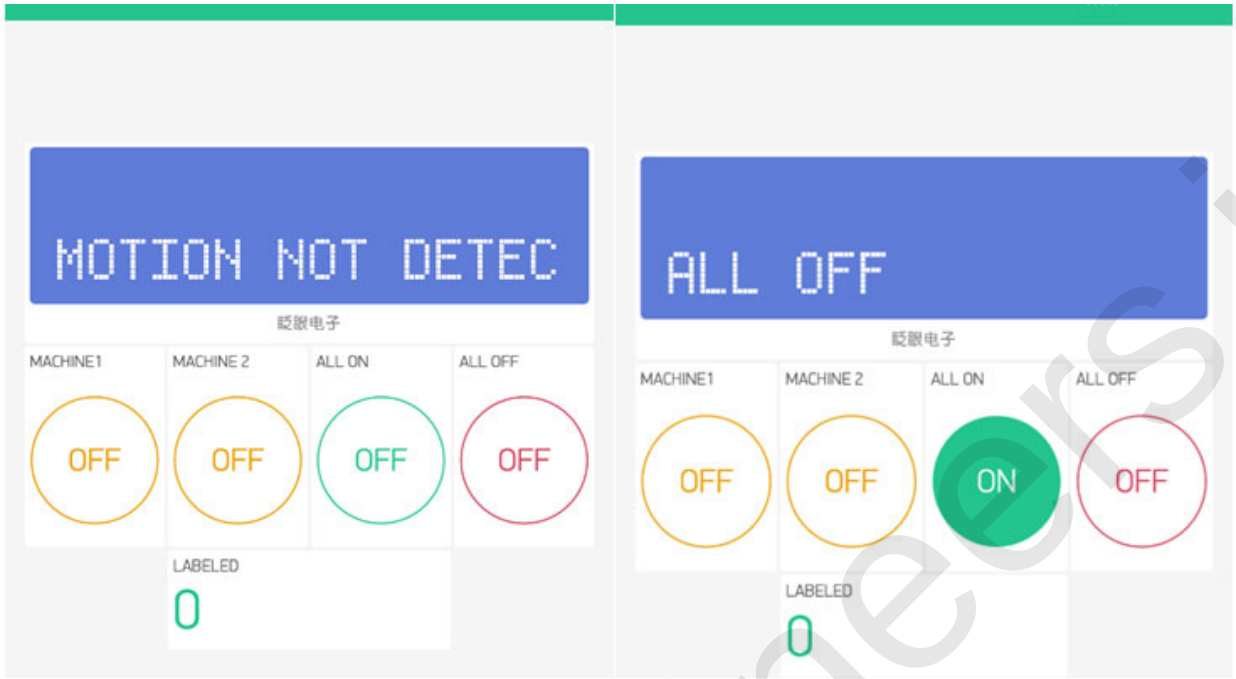

Figure 3.4: *Blynk application*

Programs

The following is the program for NodeMCU:

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <LiquidCrystal.h>

const int rs = D1, en = D2, d4 =D3, d5 = D4, d6 = D5, d7 = D6;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
BlynkTimer timer;
char auth[] = "5c8e33bf09a04b03b2fa153928b075c5"; // add auth.
token
char ssid[] = "ESPServer_RAJ"; // add user name
char pass[] = "RAJ@12345"; // add password
int RELAY1=D7; // assign integer to pin D7
int RELAY2=D8; // assign integer to pin D8
WidgetLCD blynkDISPLAY(V1); # connect blynk LCD on virtual pin
V1

BLYNK_WRITE (V2) # connect button to virtual pin V2 of blynk APP
```

```
{
 int MACHINE1_VAL = param.asInt(); // assigning incoming value
 from pin V1 to a variable
 if(MACHINE1_VAL==HIGH)
 {
  blynkDISPLAY.clear(); // clear the contents of blynk LCD
  Serial.write(10); // write data on serial
  digitalWrite(RELAY1,HIGH); # make relay pin to HIGH
  blynkDISPLAY.print(0,1,"MACHINE1 ON"); // print string on
  Blynk LCD
  lcd.setCursor(0,2); // set cursor on hardware connected LCD
  lcd.print("MACHINE1 ONN "); # print string on LCD
  delay(20); // delay of 20 mSec
 }
}

BLYNK_WRITE(V3) // connect button to virtual pin V3 of blynk APP
{
 int MACHINE2_VAL = param.asInt();
 if(MACHINE2_VAL==HIGH)
 {
  blynkDISPLAY.clear(); // clear the contents of blynk LCD
  Serial.write(10); // print data on serial
  digitalWrite(RELAY2,HIGH); # make relay pin to HIGH
  blynkDISPLAY.print(0,1,"MACHINE2 ON");
  lcd.setCursor(0,2); // set cursor on LCD
  lcd.print("MACHINE2 ONN "); // print string on LCD
  delay(20); // delay of 20 mSec
 }
}

BLYNK_WRITE(V4) // connect button to virtual pin V4 of blynk APP
{
 int ALL_OFF_VAL = param.asInt(); // assigning incoming value
 from pin V1 to a variable
 if(ALL_OFF_VAL ==HIGH)
 {
  blynkDISPLAY.clear(); // clear the contents of Blynk LCD
```

```
    Serial.write(30); // send data on serial
    digitalWrite(RELAY1,LOW); // make relay1 pin to LOW
    digitalWrite(RELAY2,LOW); // make relay2 pin to LOW
    blynkDISPLAY.print(0,1,"ALL OFF "); // print string blynk LCD
    lcd.setCursor(0,2); // set cursor on Hardware connected LCD
    lcd.print("ALL MACHINE OFF"); // print string on LCD
    delay(20); // delay of 20 mSec
  }
}

BLYNK_WRITE(V5) // connect button to virtual pin V5 of blynk APP
{
  int ALL_ONN_VAL = param.asInt(); // assigning incoming value
  from pin V1 to a variable
  if(ALL_ONN_VAL ==HIGH)
  {
    blynkDISPLAY.clear(); // clear the contents of blynk LCD
    Serial.write(40); // write data on serial
    digitalWrite(RELAY1,HIGH); // Make relay1 pin to HIGH
    digitalWrite(RELAY2,HIGH); // Make relay2 pin to HIGH
    blynkDISPLAY.print(0,1,"ALL ON "); // print string on blynk
    LCD
    lcd.setCursor(0,2); // set cursor on LCD
    lcd.print("ALL MACHINE ONN"); // print string on LCD
    delay(20); // delay of 20 msec
  }
}

void READ_SENSOR()
{
  int X =digitalRead(D0); // read digital pin
  if(X==HIGH)
  {
    Blynk.virtualWrite(V6, X); // print data on blynk virtual pin
    V6
    blynkDISPLAY.print(0,1,"MOTION DETECTED "); // print string
    on blynk LCD
    lcd.setCursor(0,1); // set cursor on LCD
```

```
   lcd.print("MOTION DETECTED "); // print string on LCD
  }
  else
  {
   Blynk.virtualWrite(V6, X); // print data on blynk virtual pin
   V6
   blynkDISPLAY.print(0,1,"MOTION NOT DETECTED"); // print
   string on blynk LCD
   lcd.setCursor(0,1); // set cursor on LCD
   lcd.print("MOTION NOT DETECTED"); // print string on LCD
  }
}

void setup()
{
 Serial.begin(9600); // initialize serial communication
 Blynk.begin(auth, ssid, pass); // initialize blynk app
 lcd.begin(20,4); // initialize LCD
 pinMode(RELAY1,OUTPUT); // assign relay1 as an OUTPUT
 pinMode(RELAY2,OUTPUT); // assign relay2 as an OUTPUT
 timer.setInterval(5000L, READ_SENSOR); // read function after
 5000 mSec
}

void loop()
{
 Blynk.run(); // start bynk app
 timer.run(); // Initiates BlynkTimer

}
```

Conclusion

This chapter concludes the garage door mechanism and its control with the Blynk app. The complete description is discussed with the help of the circuit diagram and program in detail.

CHAPTER 4

Baggage Tracker

The theft of luggage from public places and other areas is a troublesome situation for anyone. Majority of the work that has been done in the recent years is mainly based on the video surveillance. In such systems, the monitoring is done with the help of human beings for the detection of suspicious bags or activities. Few techniques of monitoring the bags for checking the things inside the bags are also developed. Some of them are discussed here.

- Konas Bags Konas: It developed the bag packs and luggage that can be tracked with the help of a smartphone. The tracking device made these bag packs and luggage trackable with the help of smartphone apps that is installed in your cell phones [16].

  The following are the disadvantages:

  - Such systems are not able to provide the safety and security because this system can only track the bag packs and luggage after being lost rather than being stolen.
  - Such systems are not reliable because the owner is hardly aware of the things being lost or stolen and by the time, the owner comes to know the possibility that the bag and the belongings may have travelled a long distance.

- **Trackable Bag Tags**: A number of trackable tags are available in market which can be tied with the bag. Such tags have inbuilt GPS in them that will help to track your bag. Various features are available in such tags like Robot Check, LugLoc and dynotag, and more [17].

  The following are the disadvantages:

  - Firstly, such tags will notify the owner once the bag is stolen or lost. The tracking is only possible after it is lost or stolen only.

- Secondly, such tags are not able to provide the visualization interface on a laptop, mobile, or computer to see the actual location of the bag. The owner is dependent on the tag that ownsthe company to track the actual location of the bag and belongings.

- **Trackdot Luggage Tracker**: Another device called Trackdot is a ground-based cellular technology-enabled microelectronics device which can be kept inside the luggage for tracking purpose. This device automatically enters into the airplane mode when kept inside the airplane duringtakeoff and automatically enters into the normal mode when landing brakes are applied [18].

  The following are the disadvantages:

  - The cost of this system is very high and approximately $56.
  - Availability is in the western countries only.
  - An instant notification system to inform stolen or lost is not available in the system.
  - This device is not available in India.

Introduction

In order to tackle theft of a luggage, a tracking system is designed. This system has an alarm that is connected to the Arduino Uno board and to track the location, a GPS module is also used. As soon as the bag goes beyond the predefined range, the alarm will start ringing. Moreover, for tracking purpose, a map has been created that will track the location of the bag as it moves. The markers created on the mobile app will give the actual location of the bag as it goes away from the owner.

System description

The system uses **IoTs** (**Internet of Things**) in order to track the luggage and bag packs. This system is designed with the help of the Arduino board and GPS module, as shown in *Figure 4.1*. An alarm has been connected to the system in order to generate an alert. As soon as the bag getsstolen or is lost, the alarm starts alerting the owner. This system is provided with the map that is synchronized with the system in order to track the actual location of the

bag and belongings with the help of markers. In addition to this, the bag moves out of a particular range like 10m, 20m, or 30 m and the owner gets a notification in the form of a message also.

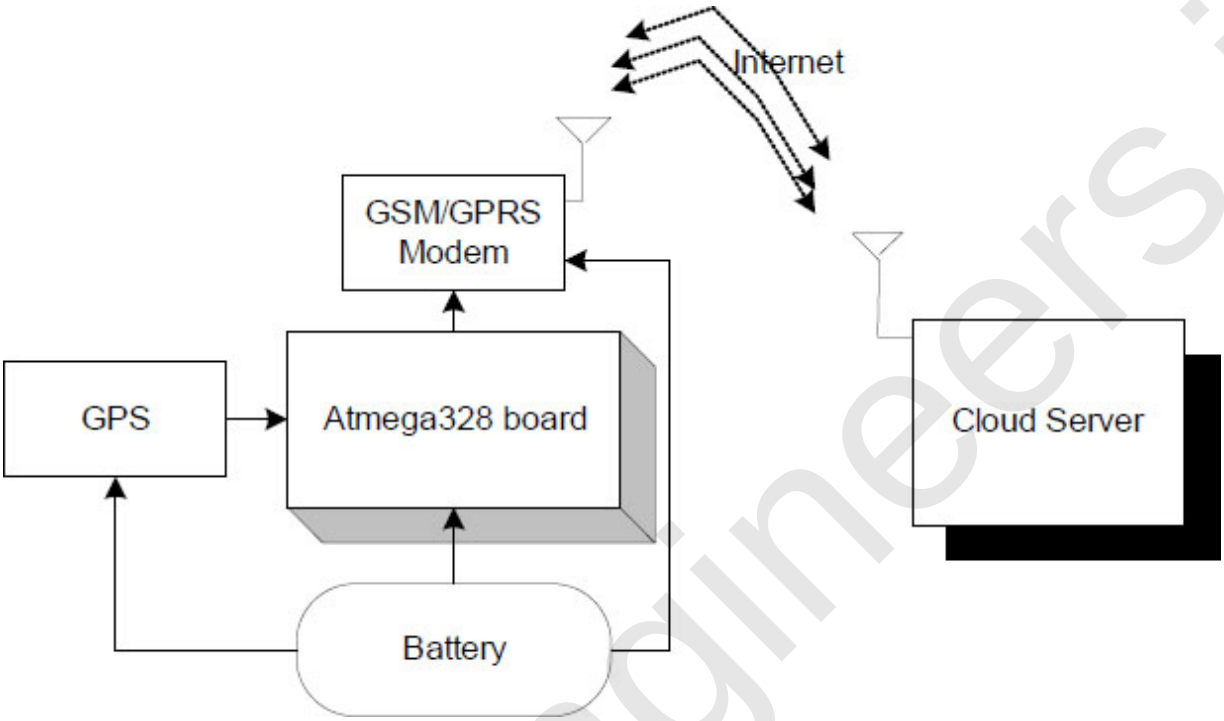

Figure 4.1: *Block diagram of the system*

Circuit diagram and connection

The following is the description of the connections:

1. +5V pin and GND of the power supply are connected to Vcc pin and GND pin of the customized Atmega328 board with NuttyFi.
2. +5V, GND, RX and TX pins of the GPRS modem connected to +5V, GND, PD7 and PD6 pins of the Atmega328 board.
3. +5V, GND, RX and TX pins of the GPS connected to +5V, GND, PD9 and PD8 pins of the Atmega328 board.

The circuit diagram of the proposed system is shown in *Figure 4.2*:

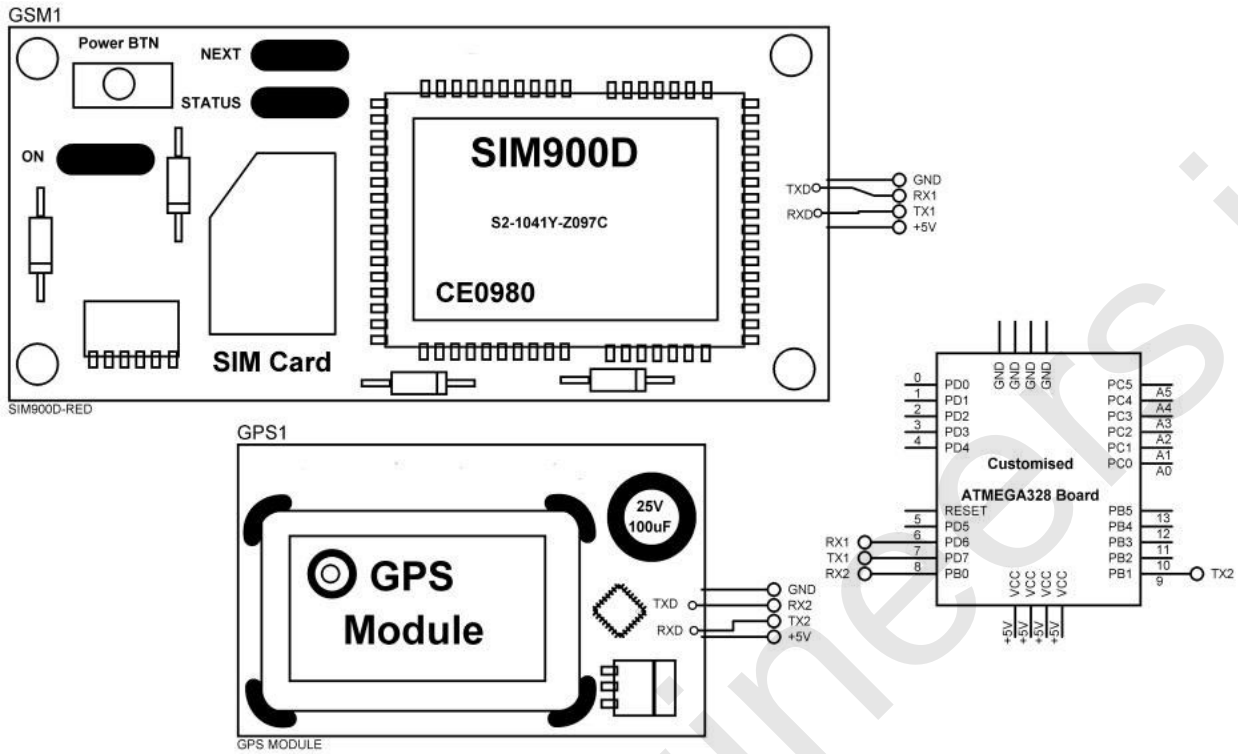

Figure 4.2: *Connections of the circuit*

Application/data logger

Let's take a look at the following program:

```
#include <SoftwareSerial.h>
#include <String.h>
#include <TinyGPS.h>

SoftwareSerial MyGPRS(6, 7); // use soft serial to make RX, TX
for GPRS
SoftwareSerial MyGPS(8, 9); // use soft serial to make RX, TX
for GPS
TinyGPS gps;
void getgps(TinyGPS &gps);
char thingSpeakAddress[] = "api.thingspeak.com";
int8_t answer;
float latitude, longitude;

void getgps(TinyGPS &gps)
{
```

```
  //float latitude, longitude;
  gps.f_get_position(&latitude, &longitude);
  Serial.print(latitude); // print data on serial
  Serial.print(" "); // print data on serial
  Serial.print(longitude); print data on serial
  Serial.println(" "); print data on serial
  delay(3000); // delay of 3000msec
}

void CallGPRS()
{
  byte a;
  if ( MyGPS.available() > 0 ) // if there is data coming into
  the serial line
  {
    a = MyGPS.read(); // get the byte of data
    if(gps.encode(a)) // if there is valid GPS data…
    {
      getgps(gps); // grab the data and display it on the LCD
    }
  }
  gprspwr_on();
  answer = sendATcommand("AT+CGATT?","OK",5,2000);
  answer = sendATcommand("AT+CSTT=\"CMNET\"","OK",3,2000);
  answer = sendATcommand("AT+CIICR","OK",3,2000);
  answer = sendATcommand("AT+CIFSR","OK",3,2000);
  answer = sendATcommand("AT+CIPSPRT=0","OK",3,2000);

  //connect gprs to thingspeak
  answer =
  sendATcommand("AT+CIPSTART=\"tcp\",\"api.thingspeak.com\",\"80\
  "","CONNECT OK",5,2000);
  answer = senddata1(latitude,longitude);
  delay(3000); // delay 3000 mSec
  gprspwr_off(); // make gps power off

  //put arduino to sleep?
  for (int i=0; i<60; i++)
  {
```

```
   delay(150); // delay of 150 mSec
  }
}

void setup()
{
  // put your setup code here, to run once:
  MyGPRS.begin(9600); // initialize soft serial communication
  Serial.begin(9600); // initialize serial communication
  MyGPS.begin(9600); // initialize soft serial communication
  delay(1000); // delay 1000 mSec
}

void loop()
{
  byte l;
  CallGPRS(); // call GPRS function
  delay(500);// delay of 500 mSec
}

/*****************************************************************
***********/
//int8_t senddata1(int data,int data1,int data2,int data3,int
data4,int data5,int data6,int data7)
float senddata1(float data,float data1)
{
  MyGPRS.println("AT+CIPSEND");
  while( MyGPRS.available() > 0) MyGPRS.read(); // Clean the
  input buffer
  delay(500);
  MyGPRS.println("POST /update HTTP/1.1"); // Send the AT command
  while( MyGPRS.available() > 0) MyGPRS.read(); // Clean the
  input buffer
  delay(500);
  MyGPRS.println("Host: api.thingspeak.com"); // Send the AT
  command
  while( MyGPRS.available() > 0) MyGPRS.read(); // Clean the
  input buffer
  delay(500);
```

```
MyGPRS.println("Connection: close"); // Send the AT command

while( MyGPRS.available() > 0)MyGPRS.read(); // Clean the input
buffer
delay(500);
MyGPRS.println("X-THINGSPEAKAPIKEY:MUCGLRMVDNCNPJGH");
//T1GIUPBKKRDPMWRX");

while( Serial2.available() > 0) Serial2.read(); // Clean the
input buffer
delay(500);
MyGPRS.println("Content-Type: application/x-www-form-
urlencoded"); // Send the AT command
while( MyGPRS.available() > 0) MyGPRS.read(); // Clean the
input buffer
delay(500);
MyGPRS.println("Content-Length:92"); // Send the AT command

while( MyGPRS.available() > 0) MyGPRS.read(); // Clean the
input buffer
delay(500);
MyGPRS.println(""); // Send the AT command

while( MyGPRS.available() > 0) MyGPRS.read(); // Clean the
input buffer
delay(500); // delay of 500 mSec
MyGPRS.print("&field1="); // Send the AT command
MyGPRS.print(data);
MyGPRS.print("&field2="); // Send the AT command
MyGPRS.print(data1);
while( MyGPRS.available() > 0) MyGPRS.read(); // Clean the
input buffer
delay(500); // delay of 500 mSec

MyGPRS.println((char)26);
delay(500); // delay of 500 mSec
while( MyGPRS.available() > 0) MyGPRS.read(); // Clean the
input buffer
delay(500); // delay of 500 mSec
MyGPRS.print("&field1="); // Send the AT command
```

```
 Serial.print(data); // print data on serial
 Serial.print("&field2="); // Send the AT command
 Serial.print(data1); // print data on serial
 answer = 0;
 return answer;
}

void gprspwr_on()
{
 pinMode(5, OUTPUT); // assign pin as an output
 digitalWrite(5,LOW); // make pin to LOW
 delay(1000); // delay of 1000 mSec
 digitalWrite(5,HIGH); // make pin to HIGH
 delay(2000); // delay of 1000 mSec
 digitalWrite(5,LOW); // make pin to LOW
 readATcommand("Call Ready",6,10000);
 if (answer == 1)
 {
 }
}

void gprspwr_off()
{
 pinMode(5, OUTPUT); // assign pin as OUTPUT
 digitalWrite(5,LOW); // make pin to LOW
 delay(1000); // delay 1000 mSec
 digitalWrite(5,HIGH); // make pin to HIGH
 delay(2000); // delay 2000 mSec
 digitalWrite(5,LOW); // make pin to LOW
 answer = readATcommand("NORMAL POWER DOWN",2,2000);
 if (answer == 1)
 {
 }
}

boolean gprspwr_status()
{
 answer = sendATcommand("AT", "OK", 2, 2000);
 if (answer == 0)
```

```
  {
  }
  else if (answer == 1)
  {
  }
  return answer;
}

int8_t readATcommand(char* expected_answer1, unsigned int
expected_answers, unsigned int timeout)
{
  uint8_t x=0, answer=0;
  boolean complete = 0;
  char a;
  char response[100];
  unsigned long previous;
  String incomingdata;
  boolean first;
  previous = millis();

  for(int i = 0; i < expected_answers; i++)
  {
    x = 0;
    complete = 0;
    a = 0;
    first = 0;
    memset(response, '\0', 100); // Initialize the string
    do
    {
      if(MyGPRS.available() != 0)
      {
        a = MyGPRS.read(); // Read GPRS
        if (a == 13)
        {
          a = MyGPRS.read(); // read GPRS
      if (a == 10)
        {
          if (first == 0)
```

```
      {
       //keep going, just ignore it
      }
      else
      {
       complete = 1;
      }
     }
    }
    else if(a == 0)
    {
    }
    else
    {
     response[x] = a;
     x++;
     first = 1;
    }
    if(strstr(response, expected_answer1) != NULL)
    {
     answer = 1;
     complete = 1;
     return answer;
    }
    else if(strstr(response, "ERROR") != NULL)
    {
     answer = 2;
    }
   }
  }
  while((complete == 0) && ((millis() - previous) < timeout));
  }
  return answer;
}
int8_t sendATcommand(char* ATcommand, char* expected_answer1,
unsigned int expected_answers, unsigned int timeout)
{
```

```
 uint8_t x=0, answer=0;
 boolean complete = 0, first = 0;
 char a;
 char response[100];
 unsigned long previous;
 String incomingdata;
 delay(100); // delay of 100 mSec
 while( MyGPRS.available() > 0) MyGPRS.read(); // Clean the
 input buffer
 MyGPRS.println(ATcommand); // Send the AT command
 previous = millis();

for(int i = 0; i < expected_answers; i++){
   x = 0;
   complete = 0;
   a = 0;
   first = 0;
   memset(response, '\0', 100); // Initialize the string
   do{
    if(MyGPRS.available() != 0)
    {
   a = MyGPRS.read(); // read GPRS
   if (a == 13)
   {
a = MyGPRS.read();// read GPRS
if (a == 10){
    if (first == 0)
    {
     //keep going, just ignore it
    }
    else
    {
     complete = 1;
    }
    }
   }
   else if(a == 0)
   {
```

```
    }
    else
    {
      response[x] = a;
      x++;
      first = 1;
    }
    if (strstr(response, expected_answer1) != NULL)
    {
      answer = 1;
      complete = 1;
    }
    else if(strstr(response, "ERROR") != NULL)
    {
      answer = 2;
      complete = 1;
    }
      }
    }
    while((complete == 0) && ((millis() - previous) < timeout));
  }
  return answer

}
```

Conclusion

This chapter discusses the language tracking system with the help of circuit diagram and program.

CHAPTER 5

Smart Trash Collector

One of the major problems of the developing countries is Trash in the current scenario. Countries like India are facing this serious problem. The main reason behind this problem is the lack of standards for the management of trash. Majority of the people don't care about the trash and the associated impact that causes big problems. Researchers identified the problems and developed a system known as *Smart Garbage Based on IoTs*. The main reason behind this system is management of trash in a proper and efficient way.

Introduction

The system consists of an ultrasonic sensor and NodeMCU with IoT to identify the garbage. The system is designed with a graphical user interface for a desktop and a mobile phone can also be used for the continuous monitoring of garbage. *Figure 5.1* presents the waste management architecture which functions on the nRF module and LoRa network:

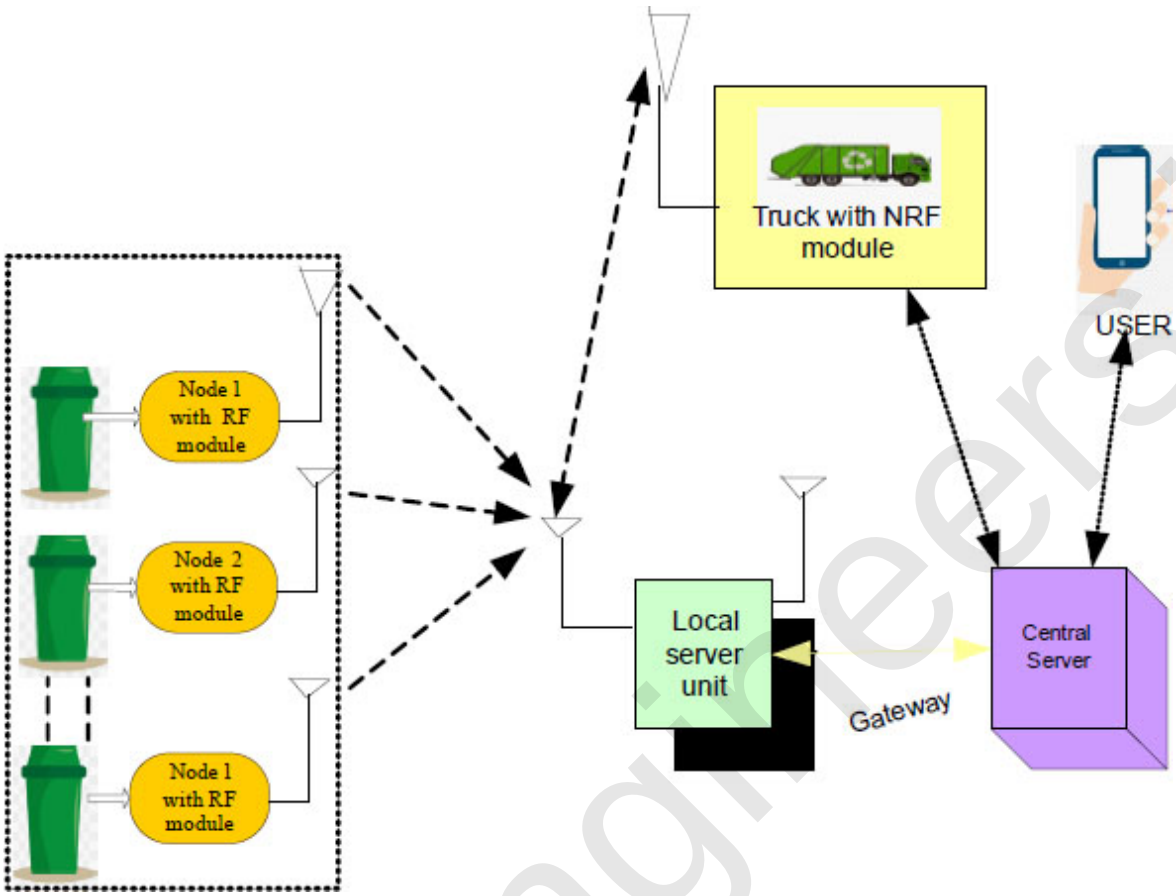

Figure 5.1: *Waste management architecture*

Here, the RF module transmits the sensory data from the bins to the local server unit, and the local server unit communicates the data to the cloud server. Through the nRF module, the data regarding the quantity and the level of waste in the bin is transmitted to the local server unit. A cloud-based mobile application is developed for the user to monitor the container and transfer the money to the municipal authority. With regards to the quantity of waste, the user of the container will pay charges to the municipal wallet for collection of garbage.

System description

Figure 5.2 presents the RF module-based sensor mote, and this is the main component of the architecture because the functioning of the whole architecture is initiated from the sensor mote end. With regards to level measurement and load cell, the microcontroller processes the information of the quantity of waste and level of waste through the nRF module. The nRF

module is useful for bi-directional communication within a range of 100 meters. A level sensor helps to measure the threshold level of garbage in the garbage bin. The load cell will help to measure the weight of the garbage in the container:

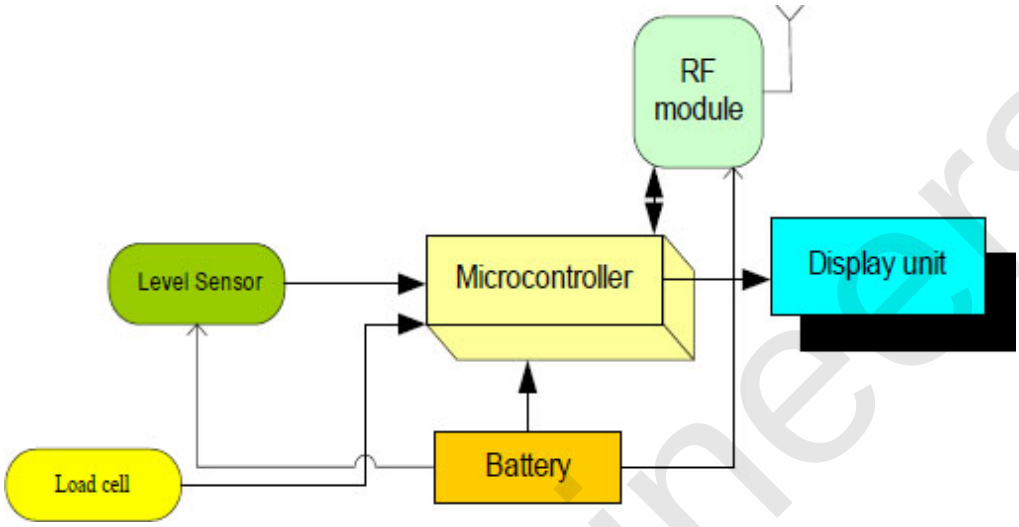

Figure 5.2: *Architecture of the nRF module-based sensor mote*

Figure 5.3 depicts the architecture of the local server unit. The local server unit is implemented in this architecture to promote both short and longrange communication. The RF module is useful for short communication, and LoRA is helpful for longdistance communication. Initially, the RF module transmits the information from the garbage bins and trucks to the local server. From the local server, the LoRA module sends the information to the central server which it receives from the RF module through the NodeMCU gateway. As usual, the microcontroller helps to control the devices that are connecting to it, and storage is also a part of this architecture for backing up the data:

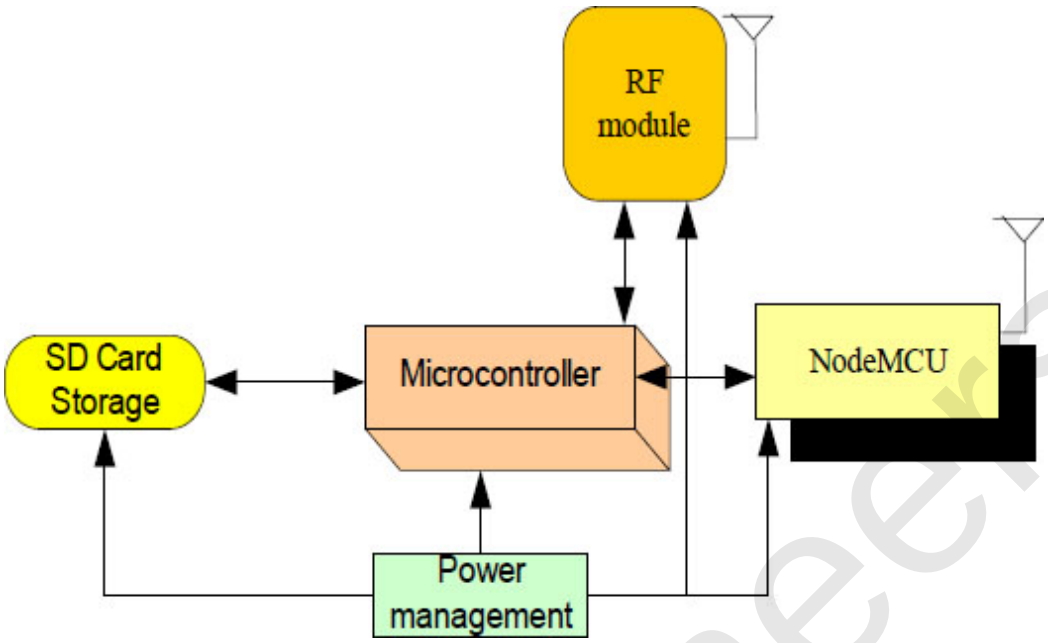

Figure 5.3: *Local server architecture*

Circuit diagram and connection

To design the system, connect the components as follows:

1. +5V pin and GND of the power supply are connected to Vcc pin and GND pin of the customized **atmega328** board with NuttyFi.
2. Pins 1 and 16 of the LCD are connected to GND of the power supply.
3. Pins 2 and 15 of the LCD are connected to +5V of the power supply.
4. Two fixed terminals of the POT are connected to +5V and GND of the LCD and the variable terminal of the POT is connected to pin 3 of the LCD.
5. RS, RW, and E pins of the LCD are connected to pins 13, GND and 12 of the customized atmega328 board.
6. D4, D5, D6, and D7 pins of the LCD are connected to pins 11, 10, 9 and 8 of the customized atmega328 board.
7. +5V and GND pin of the fire sensor are connected to +5V and GND pins of the power supply.
8. OUT pin of the fire sensor is connected to pin 5 of the customized atmega328 board.

9. +5V, GND, Trigger and echo pins of the ultrasonic sensor are connected to +5V, GND, 2 and 3 pins of the Atmega328 board.
10. +5V, GND, RX and TX pins of the RF modem are connected to +5V, GND, D7, and D6 pins of the Atmega328 board.

The connection of a smart bin is shown in *Figure 5.4*:

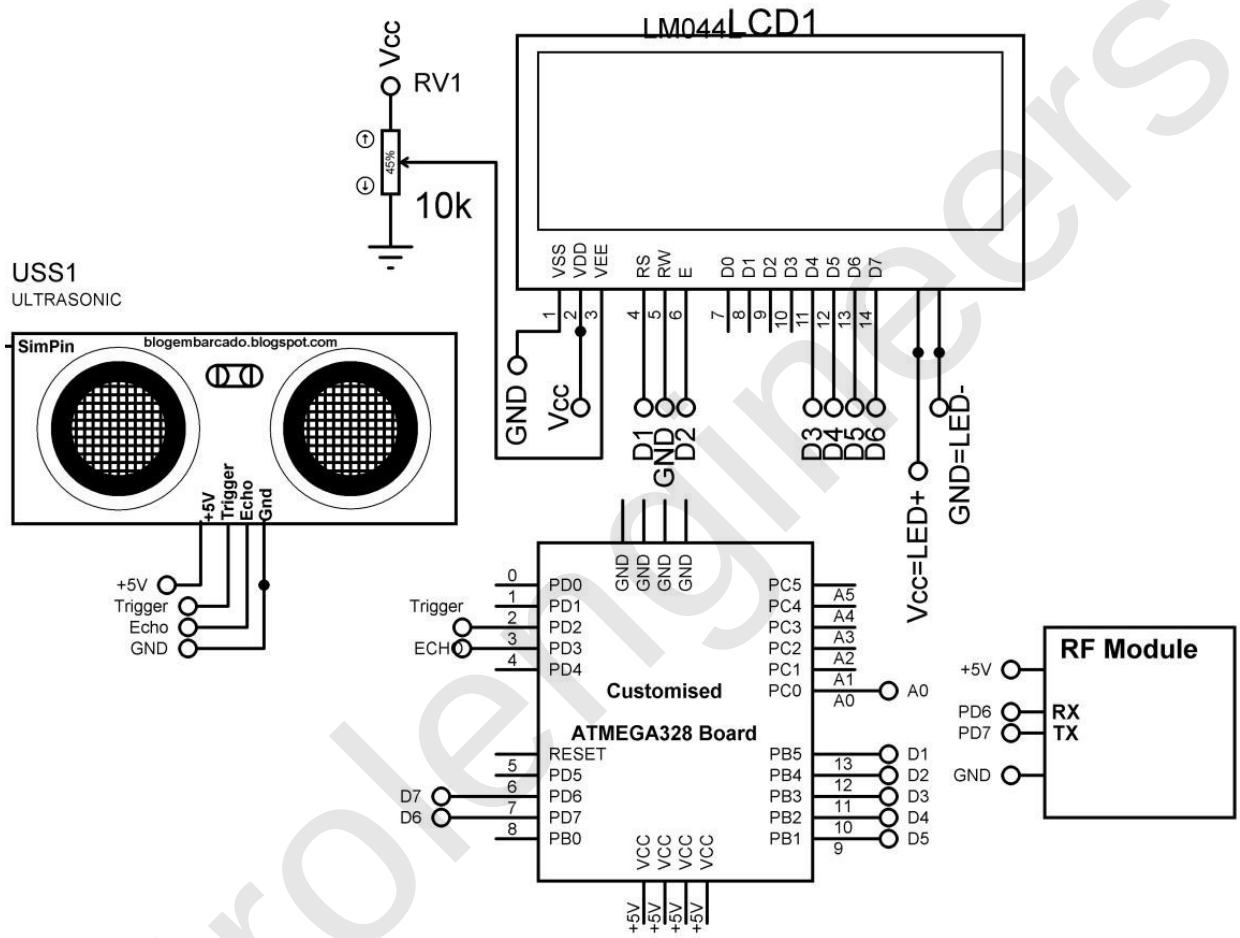

Figure 5.4: *Circuit diagram of a smart bin*

The connection of the local server is shown in *Figure 5.5* and the details are as follows:

1. +5V pin and GND of the power supply are connected to Vcc pin and GND pin of the customized atmega328 board with NuttyFi.
2. +5V, GND, RX and TX pins of the RF modem are connected to +5V, GND, PD0 and PD1 pins of the Atmega328 board.
3. +5V, GND, RX, and TX pins of NuttyFi are connected to +5V, GND, PD7, and PD6 pins of the Atmega328 board.

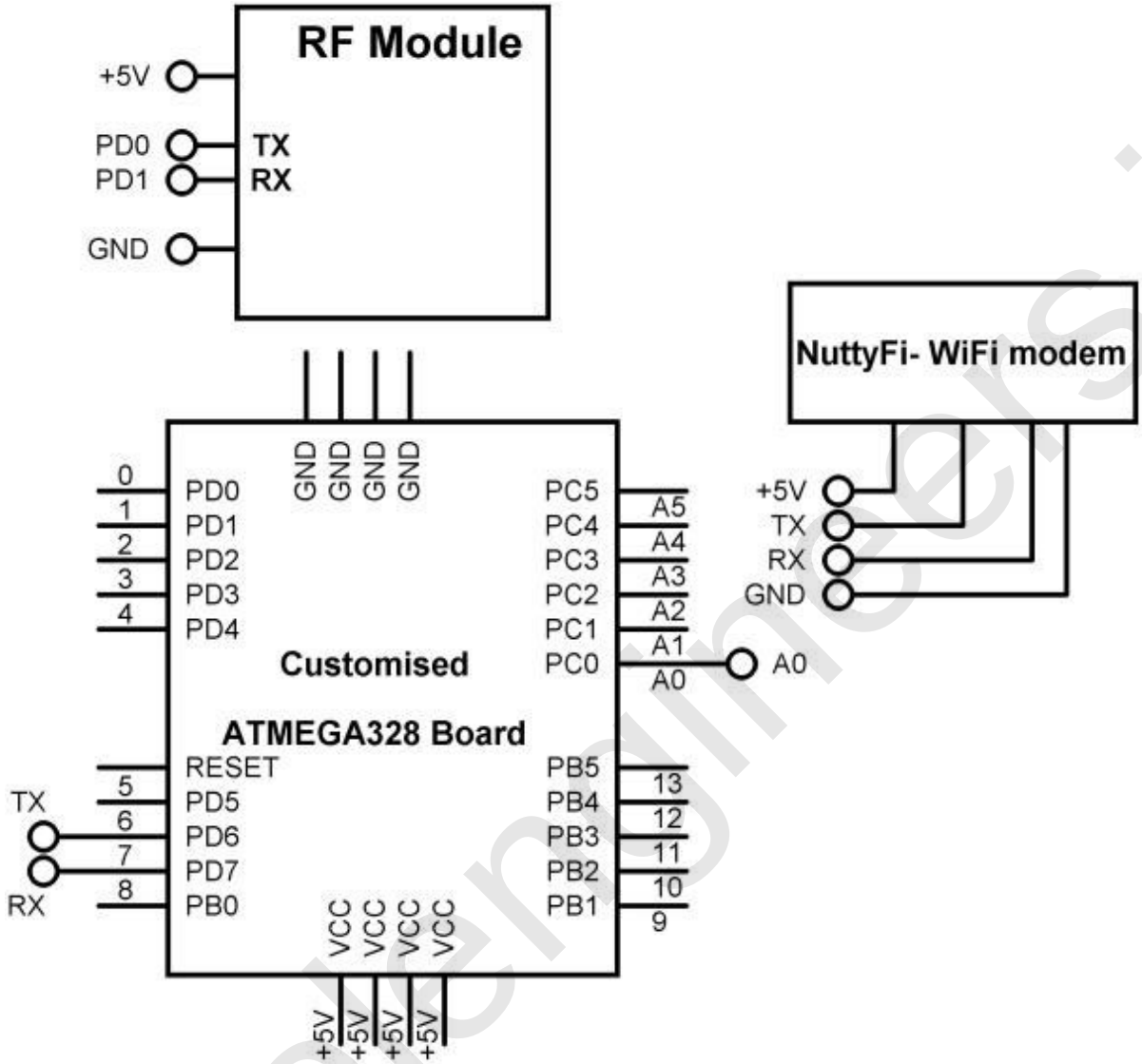

Figure 5.5: *Circuit diagram of a local server*

Application/data logger

The following code snippet is for a smart bin:

```
#include<SoftwareSerial.h>
SoftwareSerial rajSerial(6, 7);
#include <LiquidCrystal.h>// add library of LCD

LiquidCrystal lcd(13,12, 11, 10, 9, 8); // connect LCD pins with
Atmega328
```

```
const int trigger_Pin = 2; //assign integer to pin 2 (Trigger
Pin of Ultrasonic Sensor)
const int echo_Pin = 3; // assign integer to pin 3 (Echo Pin of
Ultrasonic Sensor)
long duration, inches, cm;

void setup()
{
 Serial.begin(9600); // initialize serial communication
 rajSerial.begin(9600); // initialize soft serial communication
 lcd.begin(16,2); // initialize LCD
 pinMode(trigger_Pin, OUTPUT); // set pin 10 as an output
 pinMode(echo_Pin, INPUT); // set pin 9 as an input
 delay(1000); // wait for 1 Sec
}

void loop()
{
 digitalWrite(trigger_Pin, LOW); // make Pin 10 pin to LOW
 delayMicroseconds(2); // wait for 2 uSec
 digitalWrite(trigger_Pin, HIGH); // make Pin 10 to HIGH
 delayMicroseconds(10); // wait for 10 uSec
 digitalWrite(trigger_Pin, LOW); // make Pin 10 pin to LOW
 duration = pulseIn(echo_Pin, HIGH); // make Pin 9 to HIGH
 inches = microsecondsToInches(duration); // record inches
 cm = microsecondsToCentimeters(duration); // record cm
 Serial.print("Distance:"); // print string on serial
 Serial.print(cm); // print value on serial
 Serial.print("cm"); // print string on serial
 Serial.println(); // print '\r\n'
 Serial.print("Distance:"); // print string on serial
 Serial.print(inches); // print value on serial
 Serial.print("inches"); // print string on serial
 Serial.println();// print '\r\n'
 delay(2000); // wait for 2 Sec
 rajSerial.print('\r'); // print char on LCD
 rajSerial.print(cm); // print data on LCD
 rajSerial.print('|'); // print char on LCD
```

```
 rajSerial.print(cm); // print data on LCD
 rajSerial.print('\n'); // print char on LCD
}

long microsecondsToInches(long microseconds)
{
 return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds)
{
 return microseconds / 29 / 2;

}
```

The following code snippet is for the local server:

```
#include<SoftwareSerial.h>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

SoftwareSerial rajSerial(6, 7);
// WiFi credentials
//char auth[] = "5c8e33bf09a04b03b2fa153928b075c5";// MY rsingh
char auth[] = "xxxxxxxxxxxxxxxxxxx";// add KEY here
char ssid[] = "ESPServer";
char pass[] = "RAJ@123";
WidgetLCD LCD_BLYNK(V0);
BlynkTimer timer;

void send_data()
{
 if (rajSerial.available()<1)
 return; // check serial and return if value is less than 1
 char g=rajSerial.read(); // record serial data on RX line
 if (g!='\r')
 return; // return if value is not equal to '\r'
 int cm=rajSerial.parseInt(); // use function to record first
 byte as fire sensor state data
```

```
 LCD_BLYNK.clear(); // clear the contents of blynk LCD
 Blynk.virtualWrite(V4, cm); // write data on V4 pin of blynk
 LCD_BLYNK.print(0,0,"DIS:"); // write string on of blynk LCD
 LCD_BLYNK.print(5,0,cm); // write data on of blynk LCD
}

void setup()
{
 Serial.begin(9600); // initialize serial communication
 rajSerial.begin(9600); // initialize soft serial communication
 Blynk.begin(auth, ssid, pass); // initialize blynk
 timer.setInterval(5000L, send_data); // call function sfter
 5000 msec
 }

void loop()
{
 Blynk.run(); // run blynk APP
 timer.run(); // run blynk timer

}
```

Blynk is a platform with iOS and Android apps used to control Arduino, Raspberry Pi and the likes over the Internet. It can easily build graphic interfaces for all your projects by simply dragging and dropping widgets. You can download the latest Blynk library from https://github.com/blynkkk/blynk-library/releases/latest.

Design the Blynk app and upload the program and check for the availability of data:

Figure 5.6: *Snapshot of the Blynk app for a system*

Conclusion

This chapter concludes the architecture of a smart garbage monitoring system with the Blynk app. The complete system is described with the help of a circuit diagram and program.

CHAPTER 6

Car Parking System

With the advancement of technology, millions and billions of devices can communicate with each other. One of the problems that most of the developing nations are facing is about parking of cars. Majority of cities have enormous cars but do not have enough space to park those cars. There are so many issues that are still challenges like finding the nearest unoccupied space for parking and the nearest road traffic congestion for parking.

Introduction

The industry of smart parking has witnessed majority of innovations in terms of smart cameras, automation, smart management, smart payment, smart energy, and **ANPR** (**Automatic Number Plate Recognition**). A similar approach is used with anultrasonic sensor to detect the vehicle and to open the gate automatically. NodeMCU is also used that acts as a main controller to control the peripherals.

System description

The system comprises n number of nodes and one server node. Each local node comprises the customized Atmega328 board, RF modem, IR sensor and the battery. server node comprises the customized Atmega238 board, Wi-Fi modem, RF modem, display, and battery. Local nodes communicate data through the RF modem and then communicate to the cloud server through the Wi-Fi modem. The RF modem is used to communicate in the black zone (where no internet connection is available). When the internet signal is available, the sensory data is communicated to the cloud. *Figure 6.1* shows the block diagram of the system:

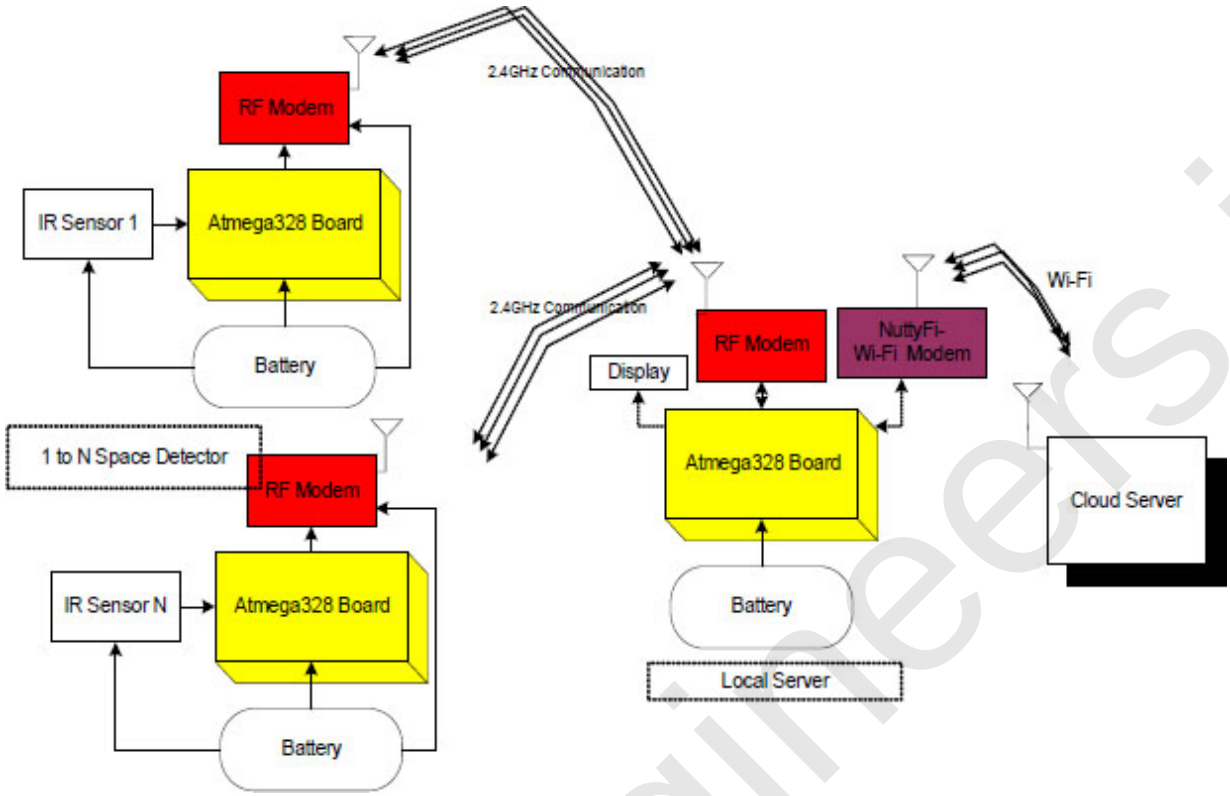

Figure 6.1: *Block diagram of the system*

Table 6.1 and Table 6.2 show the detailed component list for the greenhouse effect monitoring system and list for the local server:

Component/Specification	Quantity
Power supply 12V/1Amp	1
Jumper wire M-M	20
Jumper wire M-F	20
Jumper wire F-F	20
Power supply extension (To get more +5V and GND)	1
+12V to +5V converter	1
IR sensor	1
Atmega328 board	1
RF Modem	1
breakout board for RF Modem	1

Table 6.1: *Component list for the greenhouse effect monitoring system in the black zone*

Component/Specification	Quantity
Power supply 12V/1Amp	1
Jumper wire M-M	20
Jumper wire M-F	20
Jumper wire F-F	20
Power supply extension (To get more +5V and GND)	1
+12V to +5V converter	1
NuttyFi breakout board	1
NuttyFi	1
Atmega328 board	1
RF Modem	1
RF modem breakout board	1

Table 6.2: *Components list for local Server*

Note: All components are available at www.nuttyengineer.com.

Circuit diagram and connection

The circuit diagram of the parking space detector with the pin description is shown in *Figure 6.2* and for the local server is shown in *Figure 6.3*. The details are explained as follows:

1. Connect +12V/1A power supply DC jack to DC jack of the Atmega328 board.
2. Connect the IR sensor output pin to pin 5 of the Atmega328 board.
3. Connect TX, RX, +Vcc, and GND pins of the RF Modem to pins 0, 1, +5V, and GND of the Atmega328 board.

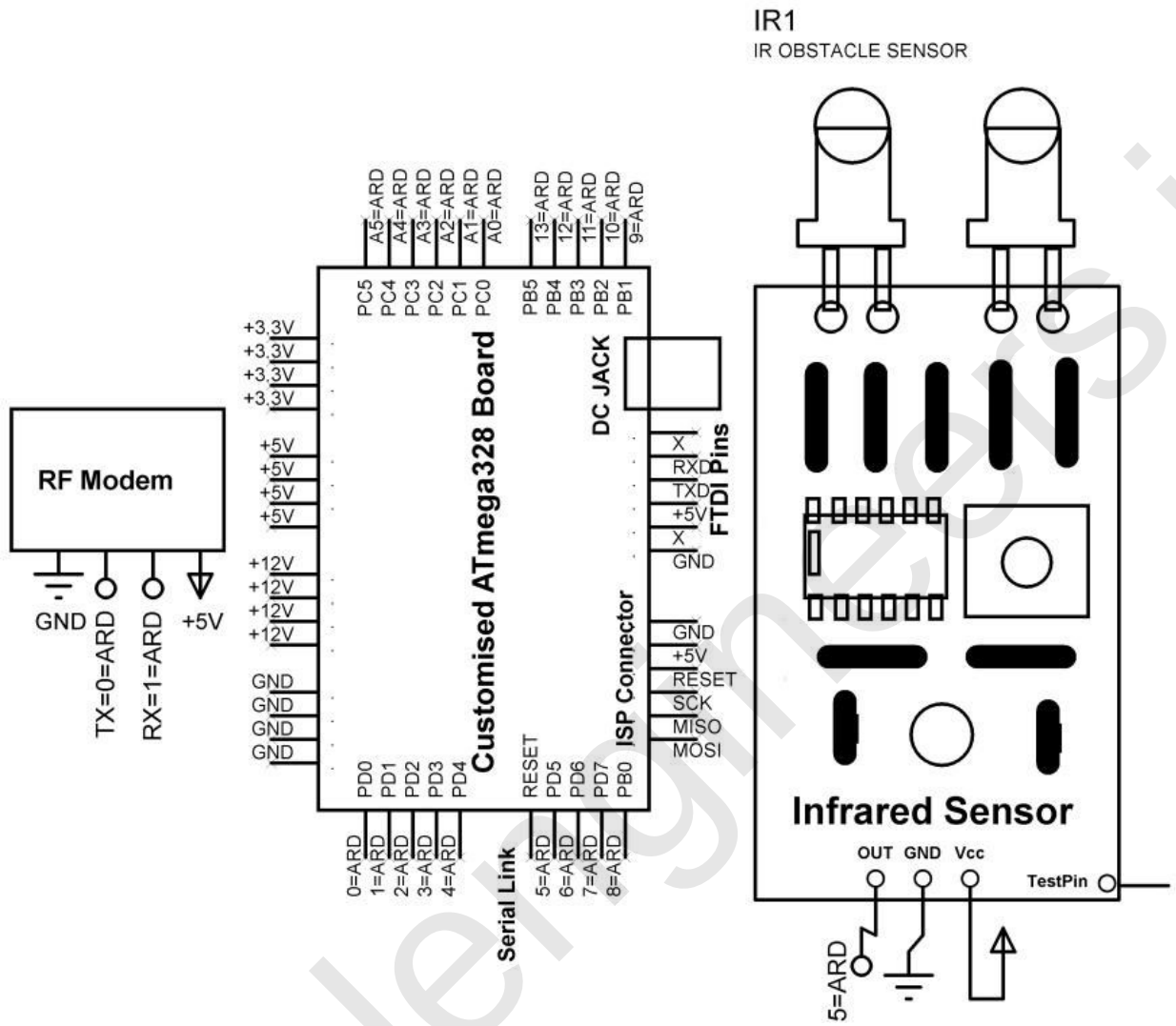

Figure 6.2: *Circuit diagram of the parking space detector*

The circuit connections for the local server are as follows:

1. Connect +12V/1A power supply DC jack to DC jack of the Atmega328 board.
2. Connect TX, RX, +Vcc, and GND pins of the RF modem to pins 0, 1, +5V, and GND of the Atmega328 board.
3. Connect TX(7), RX(6), +Vcc, and GND pins of NuttyFi to pins PD6, PD7, +5V, and GND of the Atmega328 board.

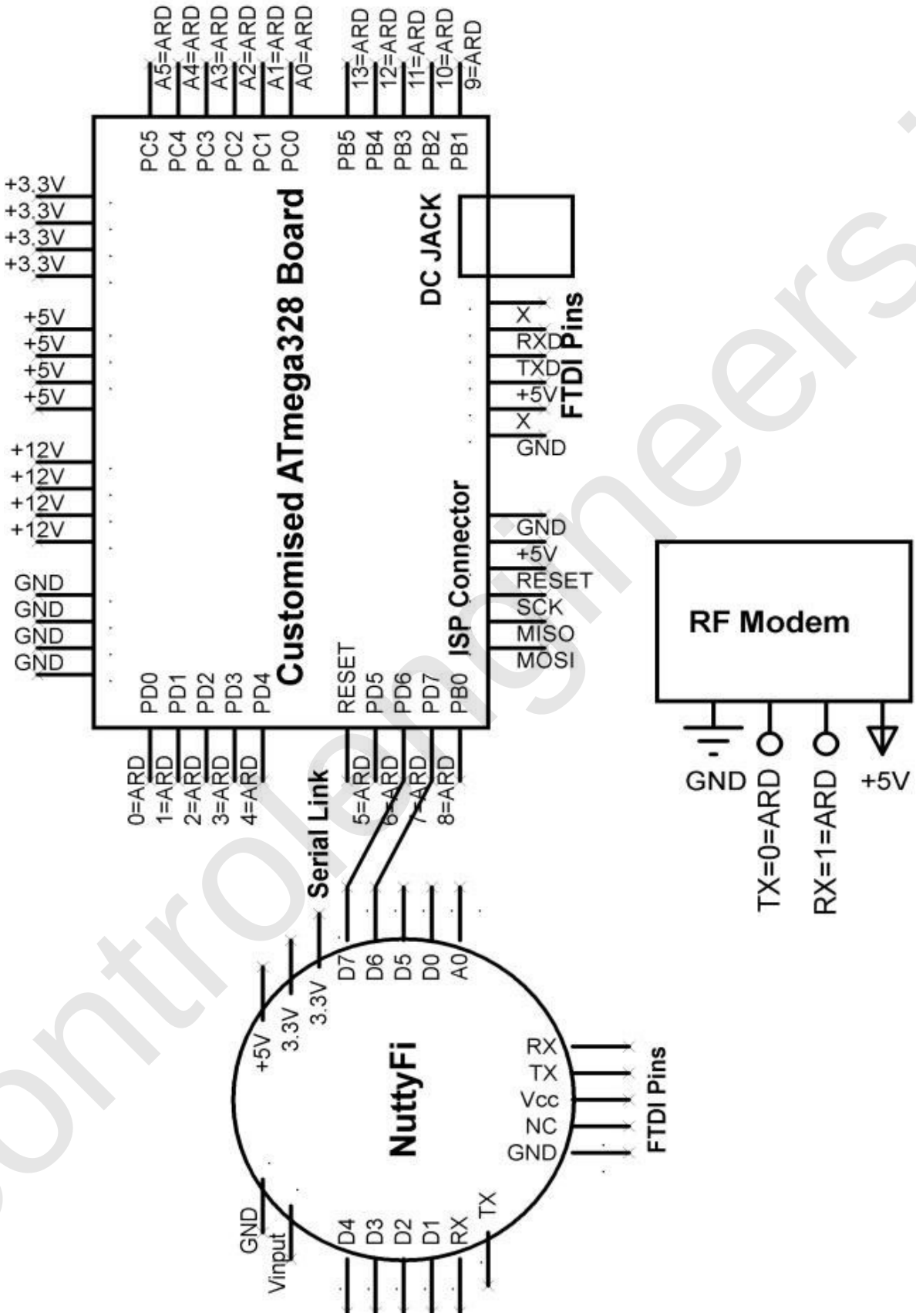

Figure 6.3: *Circuit diagram of the local server*

Application/data logger

The following code snippetis forthe parking space detector:

```
int IR_sensor_pin=5; // give integer to the pin
int IR_sensor_state=0; // give integer

void setup()
{
 Serial.begin(9600); // initialize serial communication
 pinMode(IR_sensor_pin, INPUT); // define sensor pin as an input
}

void loop()
{
 int IR_sensor_state=digitalRead(IR_sensor_pin); // read sensor
 if (IR_sensor_state== HIGH)
 {
  Serial.write(10); // send data on serial
  delay(100); // delay 100 mSec
 }
 else
 {
  Serial.write(20); // send data on serial
  delay(100); // delay 100 mSec
 }

}
```

The following code snippets are for the local servers. First, we will see the code snippet for the Atmega328 board:

```
void setup()
{
 lcd.begin(16,2); // initialize serial communication
 Serial.begin(9600); // initialize serial communication
}
```

```
void loop()
{
 int IR_sensor_state=Serial.read(); // read derail data on RX
 pin
 if (IR_sensor_state==10)
 {
  Serial.print('\r'); // write char on serial
  Serial.print(10);// send data on serial
  Serial.print("|");// send data on serial
  Serial.print(10);// send data on serial
  Serial.print('\n');// send data on serial
  delay(20);// delay 20 mSec
 }
 else
 {
  Serial.print('\r');// send data on serial
  Serial.print(20);// send data on serial
  Serial.print("|");// send data on serial
  Serial.print(20);// send data on serial
  Serial.print('\n'); // send data on serial
  delay(20);// delay 20 mSec
 }

}
```

Next, we will see the code snippet for NuttyFi:

```
#define CAYENNE_PRINT Serial
#include <CayenneMQTTESP8266.h>
#include<SoftwareSerial.h>
SoftwareSerial rajSerial(D6, D7);

char ssid[] = "ESPServer"; // add id of hotspot ID
char wifiPassword[] = "RAJ@12"; // add password here

char username[] = "b51fe780-ef06-11e9-ba7c-716e7f5ba423";
char password[] = "3b2c472fb996e4f1a12b8a1686e4538eb4b7a4f5";
char clientID[] = "f4079290-ef06-11e9-8221-599f77add412";

unsigned long lastMillis = 0;
```

```
void send_data()
{
if (rajSerial.available()<1)
 return; // check serial and return if value is less than 1
char g=rajSerial.read(); // record serial data on RX line
if (g!='\r')
 return; // return if valur is not equal to '\r'
int IR1_data=rajSerial.parseInt(); // use function to record
first byte as fire sensor state data
}

void setup()
{
 Serial.begin(9600); // initialize serial communication
 rajSerial.begin(9600); // add initialize soft serial
 communication
 Cayenne.begin(username, password, clientID, ssid,
 wifiPassword);
}

void loop()
{
 Cayenne.loop();
 //Publish data every 10 seconds (10000 milliseconds). Change
 this value to publish at a different interval.
 if (millis() - lastMillis > 10000)
 {
  lastMillis = millis();
  //Write data to Cayenne here. This example just sends the
  current uptime in milliseconds.
  send_data(); // call function here
  Cayenne.virtualWrite(0, IR1_data); // write data on channel 0
  of cayenne APP
 }
}

CAYENNE_IN_DEFAULT()
{
```

```
  CAYENNE_LOG("CAYENNE_IN_DEFAULT(%u) - %s, %s", request.channel,
  getValue.getId(), getValue.asString());
  //Process message here. If there is an error set an error
  message using getValue.setError(), e.g getValue.setError("Error
  message");

}
```

Cayenne APP for Data logging

The following are the steps to add NodeMCU in the cayenne cloud:

1. Install the Arduino IDE and add Cayenne MQTT Library to the Arduino IDE.
2. Install the ESP8266 board package inthe Arduino IDE.
3. Install the required USB driver on the computer to program ESP8266.
4. Connect ESP8266 to PC/Mac via the data-capable USB cable.
5. In the Arduino IDE, go to the tools menu; select the board, and now the port ESP8266 is connected to.
6. Use the MQTT username, MQTT password, client ID as well as **`ssid[]`** and **`wifiPassord[]`** in the Arduino IDE to write the code:

```
//#define CAYENNE_DEBUG
#define CAYENNE_PRINT Serial
#include <CayenneMQTTESP8266.h>
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(D7,D8,false,256);
// WiFi network info.
char ssid[] = "ESPServer_RAJ";
char wifiPassword[] = "RAJ@12345";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "fac81bb0-7283-11e7-85a3-9540e9f7b5aa";
char password[] = "3745eb389f4e035711428158f7cdc1adc0475946";
char clientID[] = "386b86f0-7284-11e7-b0bc-87cd67a1f8c7";
```

Figure 6.4: *Snapshot showing username, password to MQTT*

7. Burn the code in Arduino and NodeMCU. Then, a window will open; *Figure 6.4* shows the snapshots for the developed mobile app after burning the program.

Conclusion

This chapter concludes the car parking system with the help of sensors and RF communication. The RF module is used for the black zone communication where no internet signal is available. The detailsare also communicated to the cloud with the Wi-Fi module, from the location, where the internet signal is available. The detailed steps to develop the Cayenne server are also discussed in this chapter.

CHAPTER 7

Home Automation

A home automation system is a solution which enables automating the electronic and electrical devices within a home. It uses a combination of hardware and software technologies which control the appliances and devices within a home.

Introduction

The complete working of the system can be understood with the help of a system, which can control home appliances with the mobile App. *Figure 7.1* shows the block diagram of the system. The system comprises DC 12V/1Amp adaptor, 12V to 5V, 3.3V converter, liquid crystal display, relay control unit, and AC loads. The objective of the system is to display the information of ON/OFF of respective appliances on the LCD. The ON/OFF home appliances which are connected to NodeMCU/ESP8266/Wi-Fi modem from the cloud server using APP are shown in the following diagram:

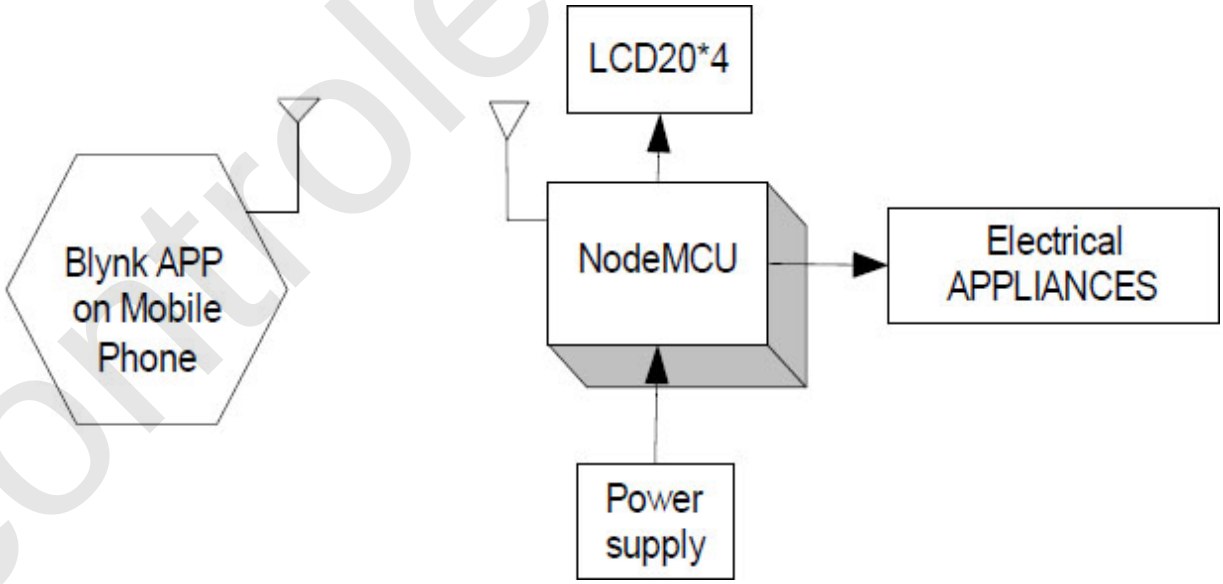

Figure 7.1: *Block diagram of the system*

Table 7.1 shows the list of components required to fabricate the system:

Component/Specification	Quantity
Power supply 12V/1Amp	1
Node MCU	1
Solid state relay board	4
Extension board for four appliances	4
Power supply extension	1
ISP programmer	1
LCD16*2	1
LCD patch	1
+5V power supply	1

Table 7.1: *Component list*

Circuit diagram

The connections of the external devices with NodeMCU areas follows:

1. +5V pin of the power supply is connected to Vcc pin of NodeMCU.
2. GND pin of the power supply is connected to GND pin of NodeMCU.
3. Pins 1 and16 of LCD are connected to GND of the power supply.
4. Pins 2 and15 of LCD are connected to +Vcc of the power supply.
5. Two fixed lags of POT are connected to +5V and GND of LCD and the variable lag of POT is connected to pin 3 of LCD.
6. RS, RW, and E pins of LCD are connected to pins D0, GND, and D1 of NodeMCU.
7. D4, D5, D6, and D7 pins of LCD are connected to pins D2, D3, D4, and D5 of NodeMCU.
8. Connect the input of the relay board to D6 pin NodeMCU.
9. Connect the input of the relay board to D7 pin NodeMCU.
10. Connect the input of the relay board to D8 pin NodeMCU.
11. Connect the input of the relay board to D9 pin NodeMCU.

12. Connect the output pin (NOs and COMs) of the relay to the AC load.

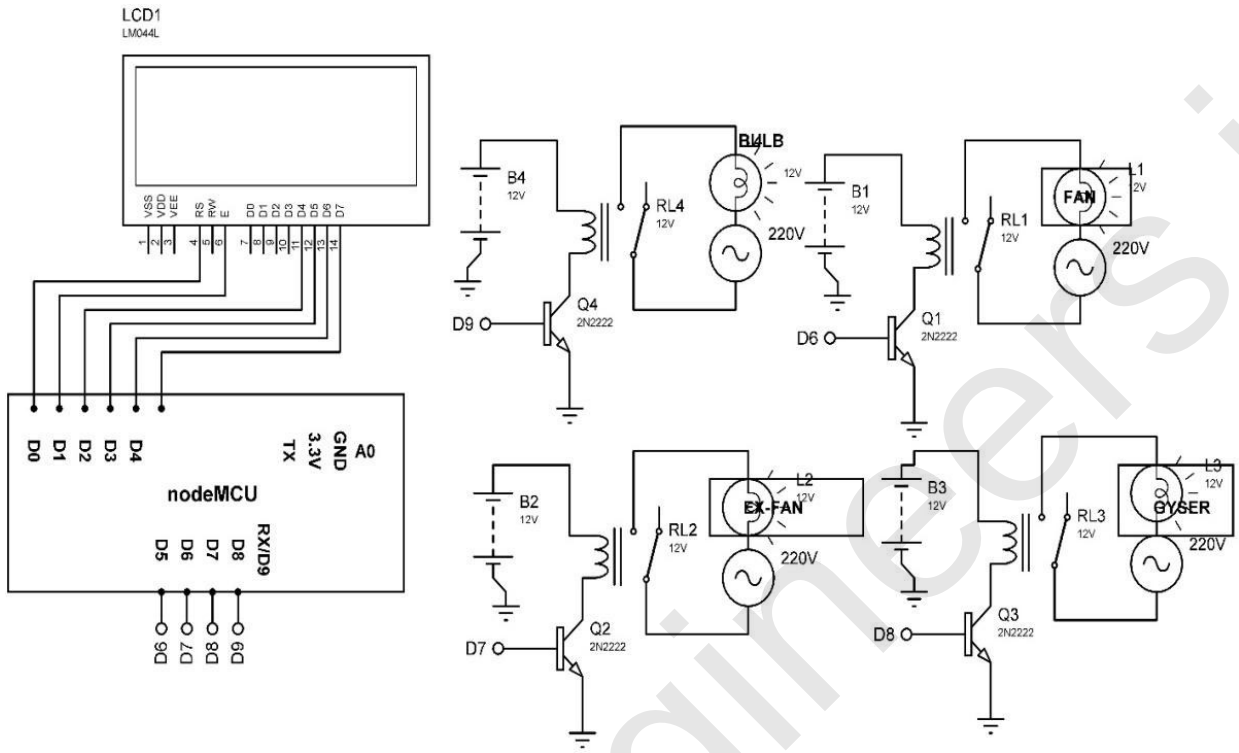

Figure 7.2: *Circuit diagram*

Figure 7.2 shows the circuit diagram of the system with NodeMCU.

Program

This program is developed to control the home appliances with the Blynk app:

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(D0, D1, D2, D3, D4, D5);
char auth[] = "8507cac915f04a1bb4b00987e420afa0"; // add auth.
Token here
char ssid[] = "ESPServer_RAJ"; // add hotspot id here
char pass[] = "RAJ@12345"; // add password here

int BULB=12; // define pin to integer
```

```
int FAN=13; // define pin to integer
int EX_FAN=15; // define pin to integer
int GYSER=3; // define pin to integer
BLYNK_WRITE(V1)
{
 int BULB_VAL = param.asInt(); // assigning incoming value from
 pin V1 to a variable
 if(BULB_VAL==HIGH)
 {
  lcd.clear(); // clear the contents of LCD
  digitalWrite(BULB,HIGH); // make pin to HIGH
  digitalWrite(FAN,LOW); // make pin to LOW
  digitalWrite(EX_FAN,LOW); // make pin to LOW
  digitalWrite(GYSER,LOW); // make pin to LOW
  lcd.setCursor(0,0);// set cursor on LCD
  lcd.print("BULB ON"); // print string on serial
  delay(10); // delay 10 mSec
 }
}

BLYNK_WRITE(V2)
{
 int FAN_VAL = param.asInt(); // assigning incoming value from
 pin V1 to a variable
 if(FAN_VAL==HIGH)
 {
  lcd.clear(); // clear the contents of LCD
  digitalWrite(BULB,LOW); // make pin to LOW
  digitalWrite(FAN,HIGH); // make pin to HIGH
  digitalWrite(EX_FAN,LOW); // make pin to LOW
  digitalWrite(GYSER,LOW); // make pin to LOW
  lcd.setCursor(0,1); // set cursor on LCD
  lcd.print("FAN ON"); // print string on LCD
 delay(10); // delay 10 mSec
 }
}

BLYNK_WRITE(V3)
```

```
{
 int EX_FAN_VAL = param.asInt(); // assigning incoming value
 from pin V1 to a variable
 if(EX_FAN_VAL==HIGH)
 {
  lcd.clear();// clear the contents of LCD
  digitalWrite(BULB,LOW); // make pin to LOW
  digitalWrite(FAN,LOW); // make pin to LOW
  digitalWrite(EX_FAN,HIGH); // make pin to HIGH
  digitalWrite(GYSER,LOW); // make pin to LOW
  lcd.setCursor(0,2); // set cursor on LCD
  lcd.print("EX FAN ON");// print string on LCD
  delay(10); // delay 10 mSec
 }
}

BLYNK_WRITE(V4)
{
 int GYSER_VAL = param.asInt(); // assigning incoming value from
 pin V1 to a variable
 if(GYSER_VAL=HIGH)
 {
  lcd.clear();
  digitalWrite(BULB,LOW); // make pin to LOW
  digitalWrite(FAN,LOW); // make pin to LOW
  digitalWrite(EX_FAN,LOW); // make pin to LOW
  digitalWrite(GYSER,HIGH); // make pin to HIGH
  lcd.setCursor(0,3); // set cursor on LCD
  lcd.print("GYSER ON"); // print string on LCD
  delay(10); // delay 10 m Sec
 }
}

BLYNK_WRITE(V5)
{
 int ALLOFF_VAL = param.asInt(); // assigning incoming value
 from pin V1 to a variable
 if(ALLOFF_VAL =HIGH)
```

```
  {
   lcd.clear();
   digitalWrite(BULB,LOW); // make pin to LOW
   digitalWrite(FAN,LOW); // make pin to LOW
   digitalWrite(EX_FAN,LOW); // make pin to LOW
   digitalWrite(GYSER,LOW); // make pin to LOW
   lcd.setCursor(0,0); // set cursor on LCD
   lcd.print("ALL OFF"); // print string on LCD
   delay(10); // delay 10 m Sec
  }
}

void setup()
{
 Serial.begin(9600); // initialize serial communication
 lcd.begin(20,4); // initialize LCD
 Blynk.begin(auth, ssid, pass); // initialize blynk
 pinMode(BULB,OUTPUT);//D6 pin of NodeMCU
 pinMode(FAN,OUTPUT);//D7 pin of NodeMCU
 pinMode(EX_FAN,OUTPUT);//D8 pin of NodeMCU
 pinMode(GYSER,OUTPUT);//D9 pin of NodeMCU
}

void loop()
{
 Blynk.run(); // run blynk APP

}
```

Blynk app

Follow the steps to design the Blynk app, described in section 3.4.

Figure 7.3 show a snapshot of the application used to control the home appliances like bulb, heater, fan, and geyser.

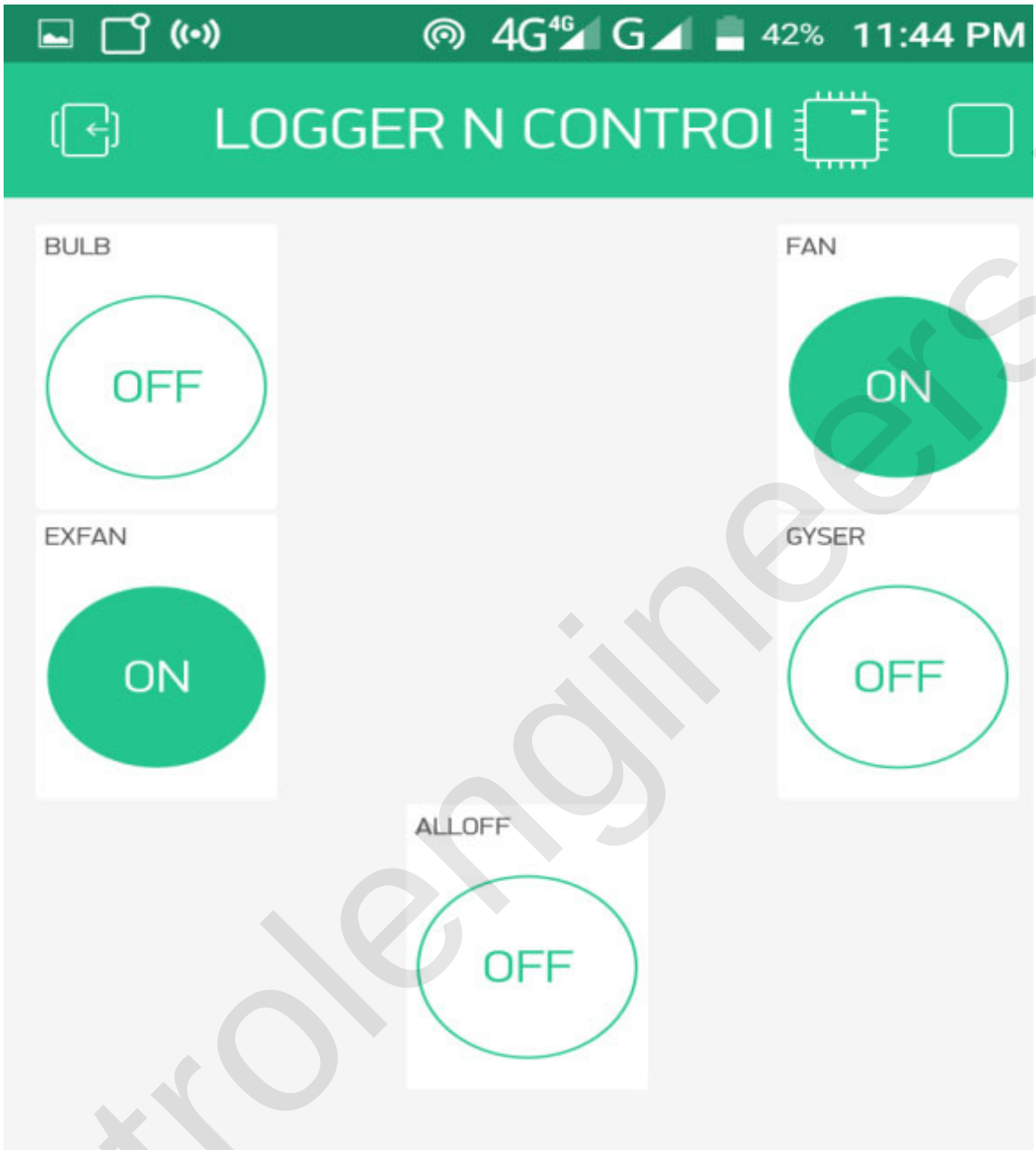

Figure 7.3: *Snapshot of the app*

Conclusion

This chapter concludes the concept of home automation. The complete system is described with the help of circuit connections and programs. The Blynk app is designed to control the home appliances.

CHAPTER 8

Environmental Parameter Monitoring

This chapter describes an IoT-based environment parameter monitoring system. Wireless Sensor Network is developed using NodeMCU, Arduino Uno, and ZigBee. A network is required to deploy sensor nodes and collect data from various locations.

The major areas of environment monitoring can be identified as follows:

- Monitoring air for quality, carbon dioxide and smog-like gases, carbon monoxide in confined areas, and indoor ozone levels.
- Monitoring water for quality, pollutants, thermal contaminants, chemical leakages, the presence of lead, and flood water levels.
- Monitoring soil for moisture and vibration levels in order to detect and prevent landslides.
- Monitoring forests and protected land for forest fires.
- Monitoring for natural disasters like earthquakes and tsunami warnings.
- Monitoring fisheries for both animal health and poaching.
- Monitoring snowfall levels at ski resorts and in national forests for weather tracking and avalanche prevention.
- Monitoring data centers for air temperature and humidity.

IoT-based greenhouse effect monitoring system

This section discusses the greenhouse monitoring system using Arduino Mini, XBee, and WiFi modem. *Figure 8.1* shows the block diagram greenhouse effect monitoring system in the environment for the black zone (without internet connectivity). The communication between nodes placed in the black zone is through XBee, and then all this information is communicated to the main server through the local server with internet connectivity. The system comprises Arduino mini, DHT11, UV index sensor,

BMP180 sensor, power supply adaptor 12V/1A and 12V to 5V converter, LCD as display unit, XBee and shield for XBee. The objective of the system is to read the sensors like DHT11 (temperature and humidity sensor), UV index sensor, and BMP180 sensor using Arduino mini and display the information on LCD. The packet of sensory data is transferred wirelessly through XBee to the local server:

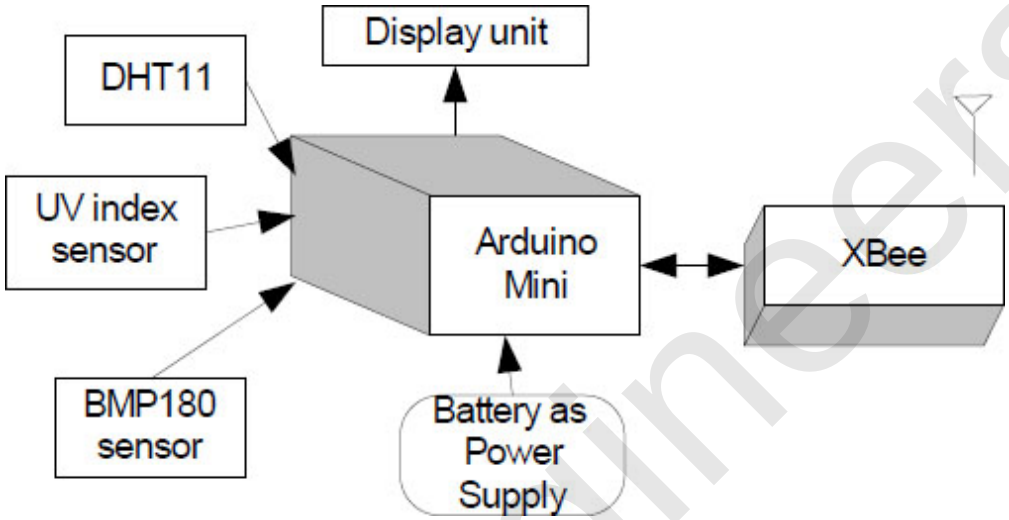

Figure 8.1: *Block diagram for greenhouse effect monitoring system in the black zone*

Figure 8.2 shows the block diagram for the local server. The system comprises Arduino mini, NodeMCU, fire sensor, smoke sensor, temperature sensor, power supply adaptor 12V/1A, the 12V to 5V convertor, and LCD as the display unit, XBee and shield for XBee:

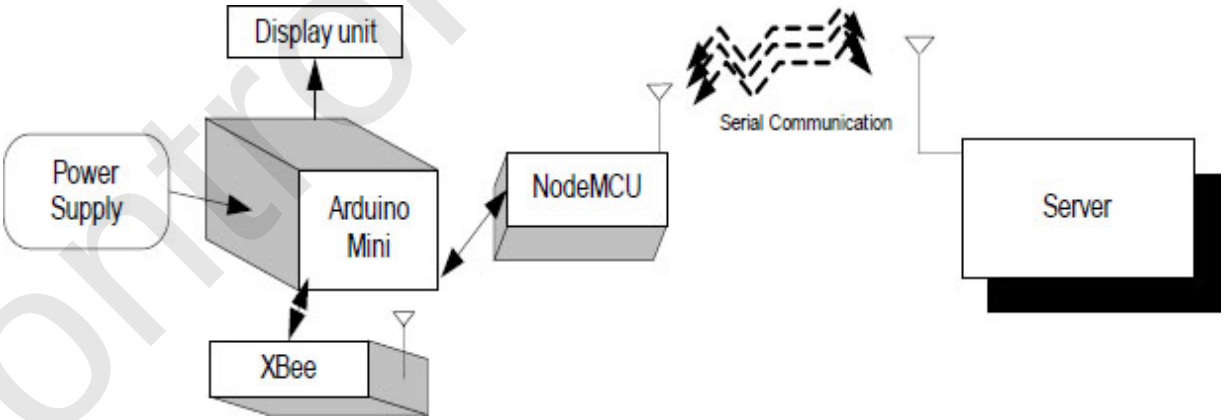

Figure 8.2: *Block diagram for the local Server*

The objective of the system is to read the sensor data packet using Arduino mini and display the information on the LCD. Then, the packet of sensory

data is transferred through the wireless WiFi link using NodeMCU to the main server or cloud server.

Table 8.1and Table 8.2 show the component list to develop the nodes for the black zone and local server, respectively:

Component/Specification	Quantity
Power supply 12V/1Amp	1
Jumper wire M-M	20
Jumper wire M-F	20
Jumper wire F-F	20
Power supply extension (To get more +5V and GND)	1
+12V to +5V converter	1
LCD20*4	1
LCD patch/explorer board	1
Fire sensor	1
Smoke sensor	1
Temperature sensor	1
Arduino UNO	1
XBee	1
XBee explorer/breakout board	1

Table 8.1: *Component list for the greenhouse effect monitoring system in the black zone*

Component/Specification	Quantity
Power supply 12V/1Amp	1
Jumper wire M-M	20
Jumper wire M-F	20
Jumper wire F-F	20
Power supply extension (To get more +5V and GND)	1
+12V to +5V converter	1
LCD20*4	1
LCD patch/explorer board	1

NodeMCU patch	1
NodeMCU	1
Arduino UNO	1
XBee	1
XBee explorer/breakout board	1

Table 8.2: *Component list for the local server*

Note: All components are available at www.nuttyengineer.com.

Circuit diagram

The following sections describe the diagram of the connections of Arduino mini and NodeMCU with external devices.

Connection of the greenhouse parameter transmitter for the black zone

The connection transmitter for the greenhouse effect monitoring system is as follows:

1. Connect DHT11 sensor output pin OUTPUT_SS to pin 2 of Arduino Mini.
2. Connect +Vcc and GND pins of DHT11 sensor to +5V and GND of the power supply.
3. Connect UV index sensor output pin to pin A1 of Arduino Mini.
4. Connect +Vcc and GND pins of UV index sensor to +5V and GND of the power supply.
5. Connect BMP180 sensor output pins SCL and SDA to pin A4 and A5 of Arduino Mini.
6. Connect +Vcc and GND pins of BMP180 sensor to +5V and GND of the power supply.
7. Connect +12V/1A power supply DC jack to DC jack of NodeMCU.
8. Connect +12V/1A power supply DC jack to DC jack of Arduino Mini.
9. Pins RS, RW, and E of LCD are connected to pins 12, GND, and 11 of Arduino Mini.

10. Pins D4, D5, D6, and D7of LCD are connected to pins 10, 9, 8, and 7 of Arduino Mini.
11. Pins 1,3, and 16 of LCD are connected to GND of the power supply using the power supply patch.
12. Pins 2 and 15 of LCD are connected to +5V of the power supply using the power supply patch.
13. RX (0), TX (1), Vcc, and GND pin of Arduino mini are connected to TX, RX, +5V, and GND pin of the XBee breakout board.

Figure 8.3 shows the circuit diagram for the greenhouse effect monitoring system in the black zone:

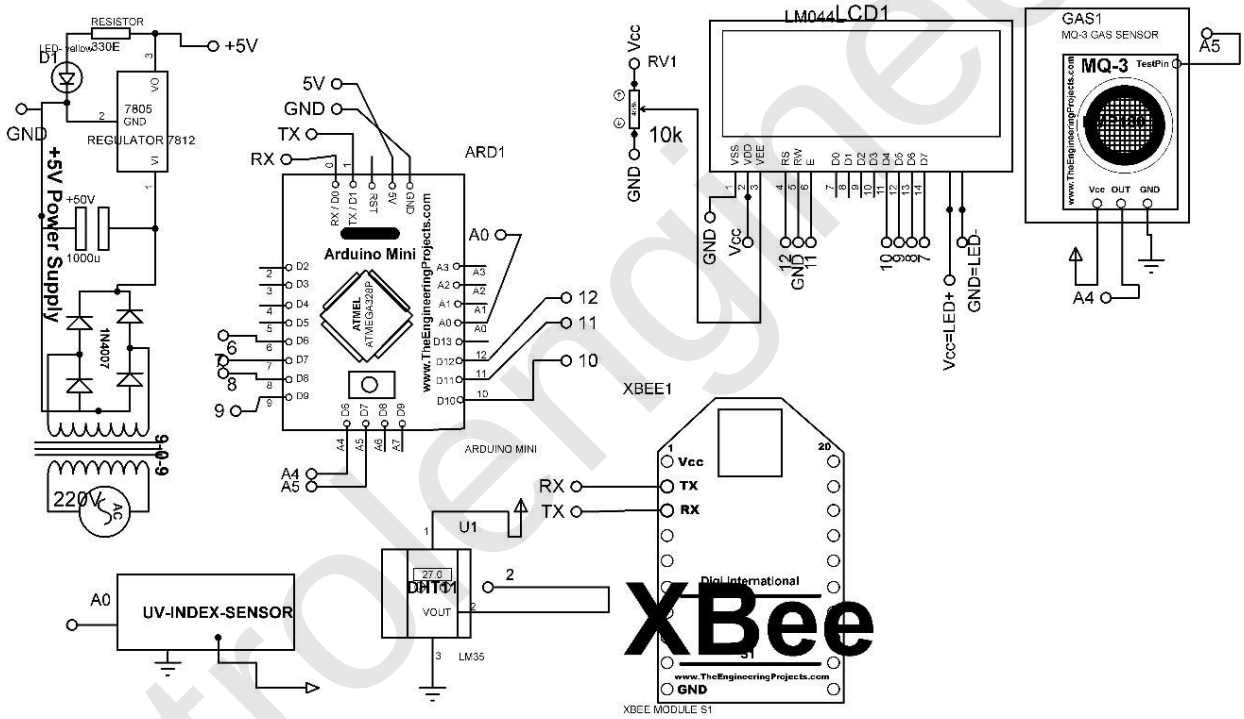

Figure 8.3: *Circuit diagram for the greenhouse effect monitoring system in the black zone*

Connections of the local server

The connections for local server are as follows:

1. Connect +12V/1A power supply DC jack to DC jack of NodeMCU.
2. Connect +12V/1A power supply DC jack to DC jack of Arduino Mini.
3. Pins RS, RW and E of LCD is connected to pins 12, GND, and 11 of Arduino Mini.

4. Pins D4, D5, D6, and D7of LCD are connected to pins 10, 9, 8, and 7 of Arduino Mini.
5. Pins 1,3, and 16 of LCD are connected to GND of the power supply using the power supply patch.
6. Pins 2 and 15 of LCD are connected to +5V of the power supply using the power supply patch.
7. RX (0), TX (1), Vcc, and GND pin of Arduino Mini is connected to RX, TX, +5V, and GND pin of the XBee breakout board.
8. TX, RX, Vcc, and GND pin of NodeMCU are connected to RX(4), TX(5), +5V, and GND pin of Arduino Mini.

Figure 8.4 shows the circuit diagram of the local server:

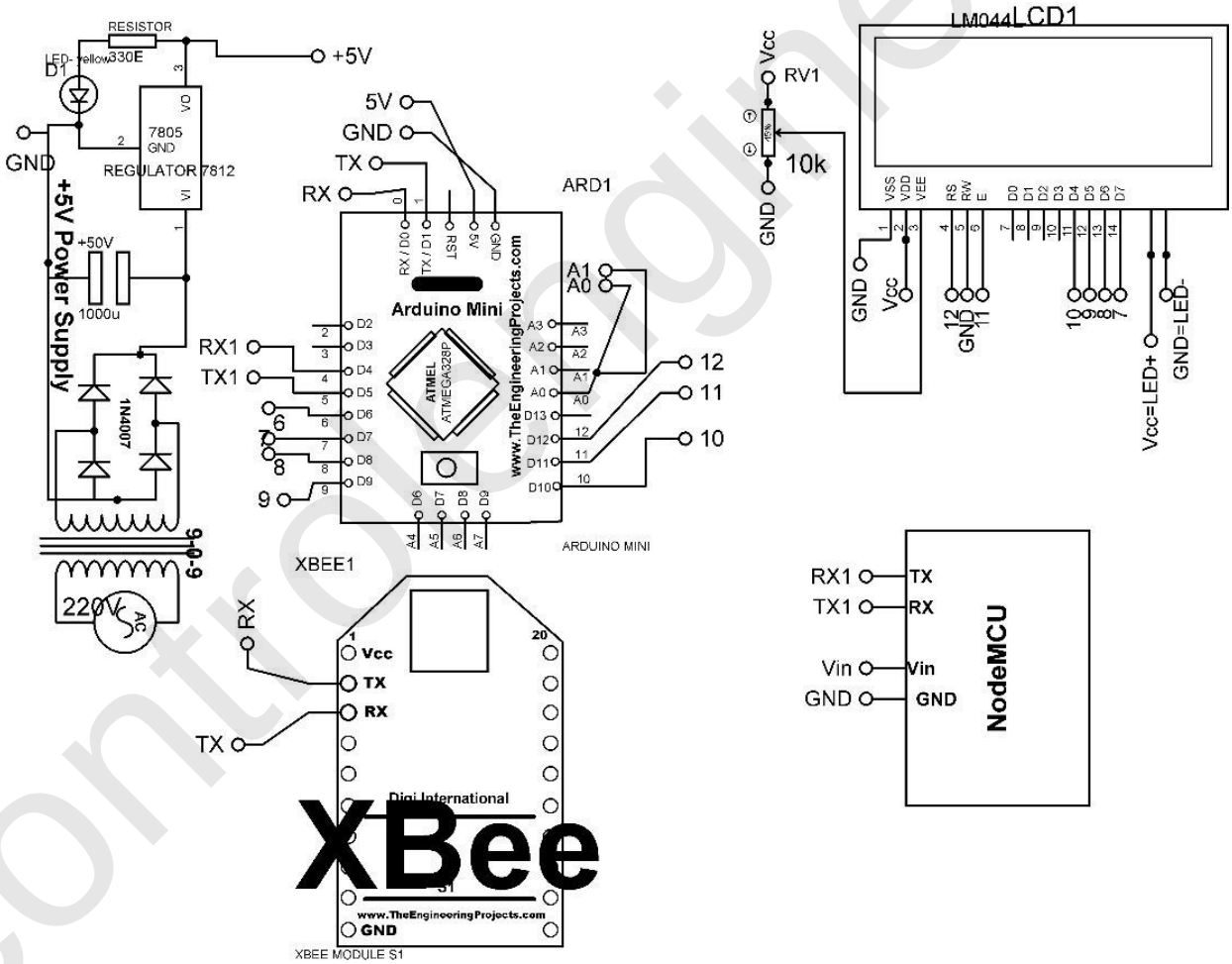

Figure 8.4: *Circuit diagram of the local server*

Program

The following sections show the program for the greenhouse effect transmitter and local server, respectively.

Arduino Mini program for the greenhouse effect transmitter

Refer to the following code:

```
// library for BMP185
#include <Wire.h>
#include <Adafruit_BMP085.h>
Adafruit_BMP085 bmp;

// library for DHT11
#include <dht.h>
dht DHT;
#define DHT11_PIN 2

// library for LCD
#include <LiquidCrystal.h>
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

// libray for Softserial
#include <SoftwareSerial.h>
SoftwareSerial mySerial(6,7);// 6 rx /7 tx

void setup()
{
 Serial.begin(9600); // initialize serial communication
 mySerial.begin(9600); // initialize soft serial communication
 lcd.begin(20, 4); // initialize LCD
 bmp.begin(); // initialize BMP sensor
}

void loop()
{
 lcd.clear(); // clear the contents of LCD
 int chk = DHT.read11(DHT11_PIN); // check DHT sensor
 float TEMP=DHT.temperature; // read temperature value
 float HUM=DHT.humidity; // read humidity value
 float PRESS=bmp.readPressure(); // read pressure value
```

```
float ALT=bmp.readAltitude(); // read altitude
int UV=analogRead(A0); // read analog pin
lcd.setCursor(0,0); // set cursor on LCD
lcd.print("TEMP:"); // print string on LCD
lcd.print(TEMP); // print value on LCD
lcd.setCursor(0,1); // set cursor on LCD
lcd.print("HUM:"); // print string on LCD
lcd.print(HUM); // print value on LCD

// read and display BMP185 data
lcd.setCursor(0,2); // set cursor on LCD
lcd.print("P0:"); // print string on LCD
lcd.print(PRESS); // print value on LCD
lcd.print("Pa"); // print string on LCD

lcd.setCursor(0,3); // Calculate altitude assuming 'standard'
barometric & pressure of 1013.25 millibar = 101325 Pascal
lcd.print("A0:"); // print string on LCD
lcd.print(ALT); // print value on LCD
lcd.print("m"); // print string on LCD

lcd.setCursor(10,0); // Calculate altitude assuming 'standard'
barometric & pressure of 1013.25 millibar = 101325 Pascal
lcd.print("UV_Indx:"); // print string on LCD
lcd.print(UV); // print value on LCD

Serial.print(TEMP); // serial print the value on serial
Serial.print(","); // print comma on serial
Serial.print(HUM); // serial print the value on serial
Serial.print(",");// print comma on serial
Serial.print(PRESS); // serial print the value on serial
Serial.print(",");// print comma on serial
Serial.print(ALT); // serial print the value on serial
Serial.print(",");// print comma on serial
Serial.print(UV); // serial print the value on serial
Serial.print('\n'); // print char on serial
delay(30); // delay 30 mSec

}
```

Arduino Mini program for the local server

Refer to the following program:

```
// library for LCD
#include <LiquidCrystal.h>
LiquidCrystal DISPLAY(12,11,10,9, 8, 7);
String TEMP,SMOKE,FIRE_STATUS;
String inputString_NODEMCU = "”; // a string to hold incoming
data

void setup(void)
{
 Serial.begin(9600); // initialize serial communication
 DISPLAY.begin(20, 4); // initialize LCD
 DISPLAY.setCursor(0,0); // set cursor on LCD
 DISPLAY.print("forest fire Det..”); // print string on LCD
 delay(4000); // delay of 4000 mSec
 lcd.clear(); // clear the contents of LCD
}

void loop(void)
{
 serialEvent_NODEMCU(); // call function
 delay(50); // delay of 50 mSec
 DISPLAY.setCursor(0,0); // set cursor on LCD
 DISPLAY.print("TEMP:”); // print string on LCD
 DISPLAY.setCursor(5,0); // set cursor on LCD
 DISPLAY.print(TEMP); // print value on LCD
 DISPLAY.setCursor(0,1); // set cursor on LCD
 DISPLAY.print("HUM:”); // print string on LCD
 DISPLAY.setCursor(5,1); // set cursor on LCD
 DISPLAY.print(HUM); // print value on LCD
 DISPLAY.setCursor(0,2); // set cursor on LCD
 DISPLAY.print("PRESS:”); // print string on LCD
 DISPLAY.setCursor(6,2); // set cursor on LCD
 DISPLAY.print(PRESS); // print value on LCD
 DISPLAY.setCursor(0,3); // set cursor on LCD
 DISPLAY.print("ALT:”); // print string on LCD
```

```
 DISPLAY.setCursor(5,3); // set cursor on LCD
 DISPLAY.print(ALT); // print value on LCD
 DISPLAY.setCursor(10,0); // set cursor on LCD
 DISPLAY.print("UV:"); // print string on LCD
 DISPLAY.setCursor(15,0); // set cursor on LCD
 DISPLAY.print(UV); // print value on LCD

 Serial.print(TEMP); // print value on serial
 Serial.print(","); // print comma on serial
 Serial.print(HUM); // print value on serial
 Serial.print(","); // print comma on serial
 Serial.print(PRESS); // print value on serial
 Serial.print(","); // print comma on serial
 Serial.print(ALT); // print value on serial
 Serial.print(","); // print comma on serial
 Serial.print(UV); // print value on serial
 Serial.print('\n'); // print char on serial
 delay(20); // delay 20 mSec
}

void serialEvent_NODEMCU()
{
 while (Serial.available()>0)
 {
  inputString_NODEMCU = Serial.readStringUntil('\n');// Get
  serial input
  StringSplitter *splitter = new
  StringSplitter(inputString_NODEMCU, ',',5); // new
  StringSplitter(string_to_split, delimiter, limit)
 int itemCount = splitter->getItemCount();
  for(int i = 0; i < itemCount; i++)
  {
  String item = splitter->getItemAtIndex(i);
  TEMP= splitter->getItemAtIndex(0); // extract sensor value
  from main string
  HUM= splitter->getItemAtIndex(1); // extract sensor value
  from main string
```

```
  PRESS= splitter->getItemAtIndex(2); // extract sensor value
  from main string
  ALT= splitter->getItemAtIndex(3); // extract sensor value
  from main string
  UV= splitter->getItemAtIndex(4); // extract sensor value from
  main string
  }
 inputString_NODEMCU = "”; // make string empty
 delay(200); // delay 200 msec
 }

}
```

NodeMCU program for the greenhouse receiver

Refer to the following program:

```
// for Softserial lib and string splitter
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include "StringSplitter.h"

SoftwareSerial rajSerial(D7,D8,false,256);
String apiKey1 = "R2ACMZBH7IV8B0KH"; // add api key
const char* ssid = "ESPServer_RAJ"; // add ID
const char* password = "12345678"; // add password
const char* server = "api.thingspeak.com";
WiFiClient client;
String TEMP,HUM,PRESS,ALT,UV;
String inputString_NODEMCU = "”; // a string to hold incoming
data

void setup()
{
 Serial.begin(9600); // initialize serial communication
 rajSerial.begin(9600); // initialize soft serial communication
 inputString_NODEMCU.reserve(200);
 delay(10); // delay 10 Msec
 WiFi.begin(ssid, password);// initialize Wi-Fi
```

```
 Serial.println(); // serial print nothing
 Serial.println();// serial print nothing
 Serial.print("Connecting to "); // print string on serial
 Serial.println(ssid); // print ssid on serial
 while (WiFi.status() != WL_CONNECTED)
 {
  delay(500); // delay 500 mSec
  Serial.print("."); // print serial
 }
 Serial.println(""); // print serial nothing
 Serial.println("WiFi connected"); // print string on serial
}

void loop()
{
 if (client.connect(server,80))
 {
  serialEvent_NODEMCU(); // call function
  send1_GREENHOUSE_HEALTH_PARA();
 }
 client.stop(); // make client stop
 Serial.println("Waiting"); // print string on serial
 delay(20000);// thingspeak needs minimum 15 sec delay between
 updates
}

void send1_GREENHOUSE_HEALTH_PARA()
{
 String postStr = apiKey1;
 postStr +="&field1=";
 postStr += String(TEMP);
 postStr +="&field2=";
 postStr += String(HUM);
 postStr +="&field3=";
 postStr += String(PRESS);
 postStr +="&field4=";
 postStr += String(ALT);
 postStr +="&field5=";
```

```
 postStr += String(UV);
 postStr += "\r\n\r\n";

 client.print("POST /update HTTP/1.1\n");
 client.print("Host: api.thingspeak.com\n");
 client.print("Connection: close\n");
 client.print("X-THINGSPEAKAPIKEY: "+apiKey1+"\n");
 client.print("Content-Type: application/x-www-form-
 urlencoded\n");
 client.print("Content-Length: ");
 client.print(postStr.length());
 client.print("\n\n");
 client.print(postStr);
 Serial.print("Send data to channel-1 "); // print string on
 Serial
 Serial.print("Content-Length: "); // print string on Serial
 Serial.print(postStr.length());
 Serial.print("Field-1: "); // print string on Serial
 Serial.print(TEMP); // print value on Serial
 Serial.print("Field-2: "); // print string on Serial
 Serial.print(HUM); // print value on Serial
 Serial.print("Field-3: "); // print string on Serial
 Serial.print(PRESS); // print value on Serial
 Serial.print("Field-4: "); // print string on Serial
 Serial.print(ALT); // print value on Serial
 Serial.print("Field-5: "); // print string on Serial
 Serial.print(UV); // print value on Serial
 Serial.println(" data send"); // print string on Serial
}

void serialEvent_NODEMCU()
{
 while (Serial.available()>0)
 {
  inputString_NODEMCU = Serial.readStringUntil('\n');// Get
  serial input
  StringSplitter *splitter = new
  StringSplitter(inputString_NODEMCU, ',',5); // new
```

```
  StringSplitter(string_to_split, delimiter, limit)
  int itemCount = splitter->getItemCount();
  for(int i = 0; i < itemCount; i++)
  {
   String item = splitter->getItemAtIndex(i);
   TEMP= splitter->getItemAtIndex(0); // extract sensor value
   from main string
   HUM= splitter->getItemAtIndex(1); // extract sensor value
   from main string
   PRESS= splitter->getItemAtIndex(2); // extract sensor value
   from main string
   ALT= splitter->getItemAtIndex(3); // extract sensor value
   from main string
   UV= splitter->getItemAtIndex(4); // extract sensor value
   from main string
  }
 inputString_NODEMCU = “”; // make string empty
 delay(200); // delay 200 mSec
 }

}
```

ThingViewapp

Follow the steps described in section 3.1.4 and receive the data on thingspeak.com. The same data can also be checked on the mobile app Thing View. *Figure 8.5* shows the snapshots for the data received on the mobile app:

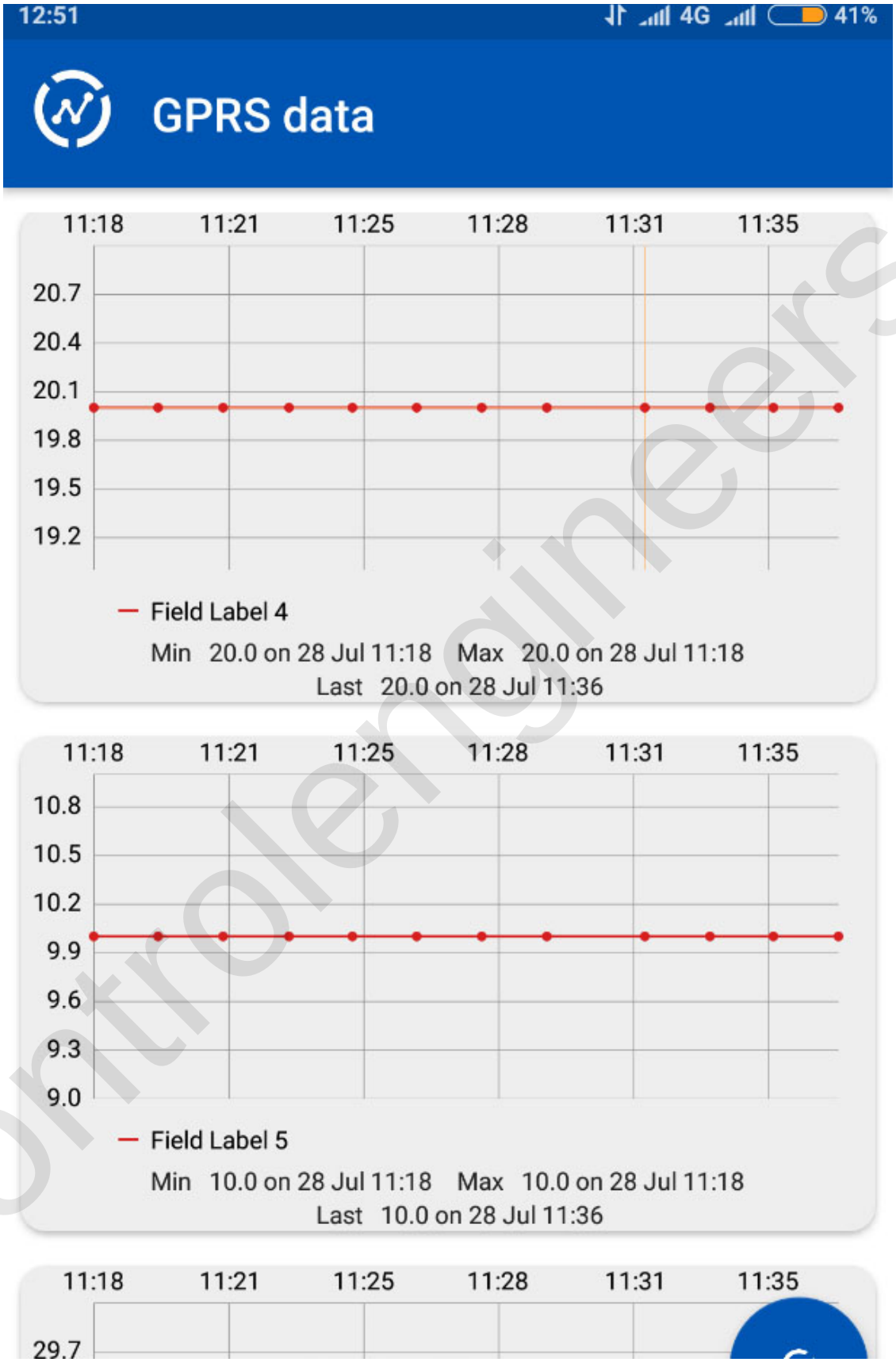

Figure 8.5: *Snapshots for the data received on the mobile app*

Conclusion

This chapter concludes the green house monitoring system. The system can be understood with the help of the circuit diagram and mobile app.

CHAPTER 9

Intelligent System for the Blind

Humans are not disabled. A person can never be broken. This chapter describes the system for blind people.

Introduction

A simple Arduino board and sensors are used for designing a stick for the blind. This smart stick is equipped with an ultrasonic sensor for sensing the obstacle, the RF remote that will locate the stick for a blind person and LDR for sensing the condition of light. All the feedback is given to the person using this stick with the help of a buzzer. *Figure 9.1* shows the block diagram of the system:

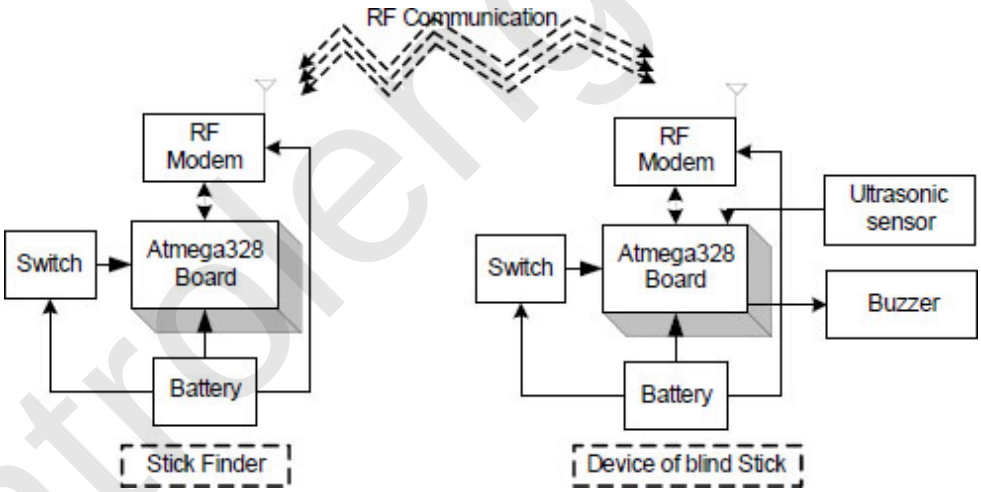

Figure 9.1: *Block diagram of the system*

Table 9.1 and *Table 9.2* gives the details of components of the required stick finder and complete system:

Component/Specification	Quantity
Battery of 9V/200 mA	1
Jumper wire M-M	20

Jumper wire M-F	20
Jumper wire F-F	20
+9V to +5V converter	1
Button	1
Atmega328 board	1
RF Modem	1
breakout board for RF Modem	1

Table 9.1: *Components for stick finder*

Component/Specification	**Quantity**
Battery of 9V/200 mA	1
Jumper wire M-M	20
Jumper wire M-F	20
Jumper wire F-F	20
+9V to +5V converter	1
Atmega328 breakout board	1
Atmega328 board	1
Atmega328 board	1
RF Modem	1
RF modem breakout board	1
Ultrasonic sensor	1
LDR sensor	1

Table 9.2: *Components for complete system*

Circuit diagram and connection

The circuit diagram of the stick finder is shown in *Figure 9.2* and the details of the device on the blind stick are shown in *Figure 9.3*:

1. Connect battery to the +5V and GND pin of the Atmega328 board.
2. Connect the button to pin5 of the Atmega328 board.

3. Connect Vcc, GND, RX, and TX pins of the RF modem to +5V, GND, 1(TX) and 0(RX) pins of the Atmega328 board.

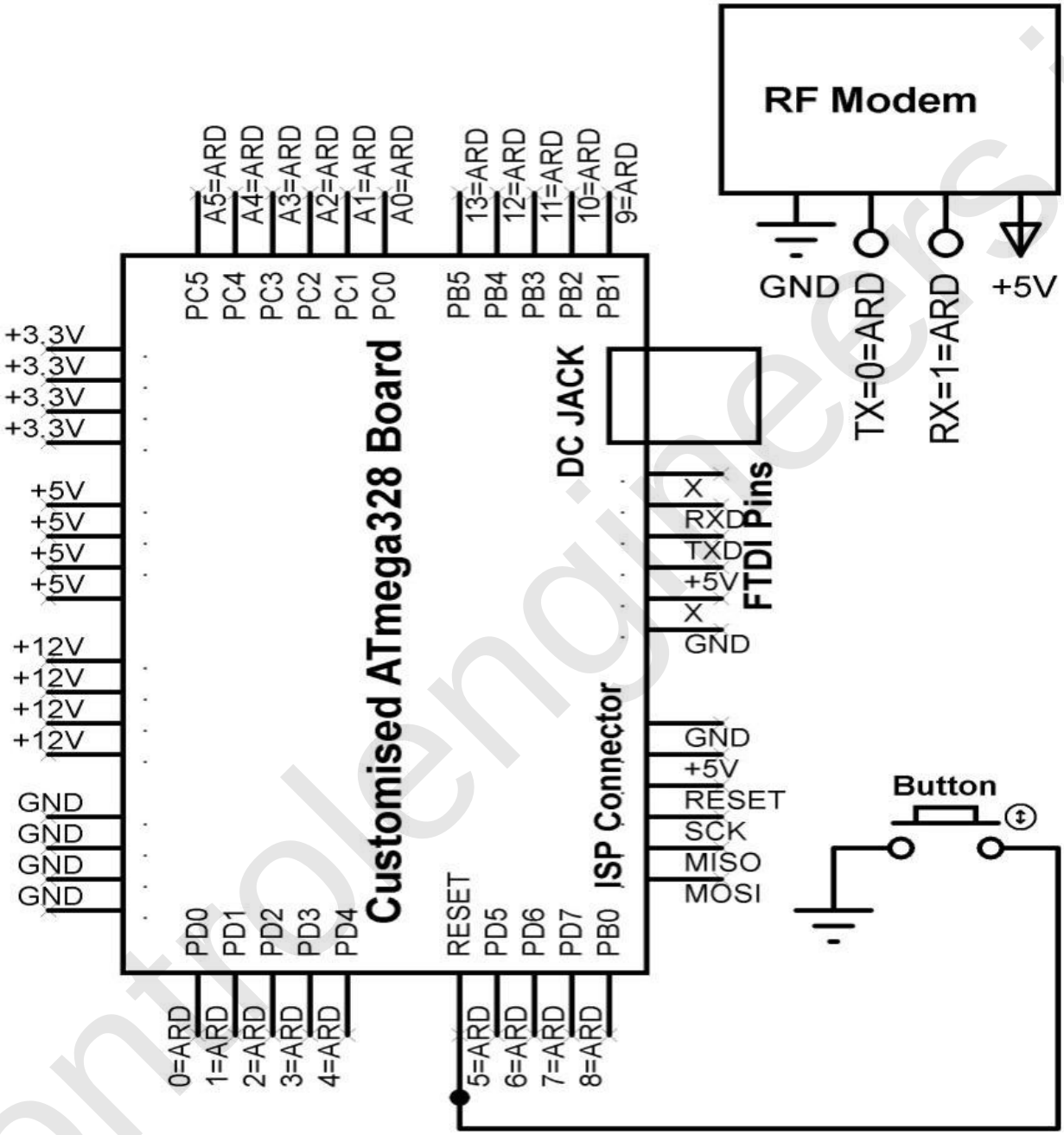

Figure 9.2: *Stick finder*

The following steps need to be followed to provide connections for the stick:

1. Connect the battery to the +5V and GND pins of the atmega328 board.
2. Connect the buzzer to pin 13 of the atmega328 board.

3. Connect Vcc, GND, RX, TX pins of the RF modem to +5V, GND, 1(TX) and 0(RX) pins of the atmega328 board.
4. Connect Vcc, GND, trigger and echo pins of the ultrasonic sensor to +5V, GND, 2 and 3 pins of the atmega328 board.
5. Connect Vcc, GND, and OUT pins of LDR to +5V, GND, and A0 pins of the atmega328 board.

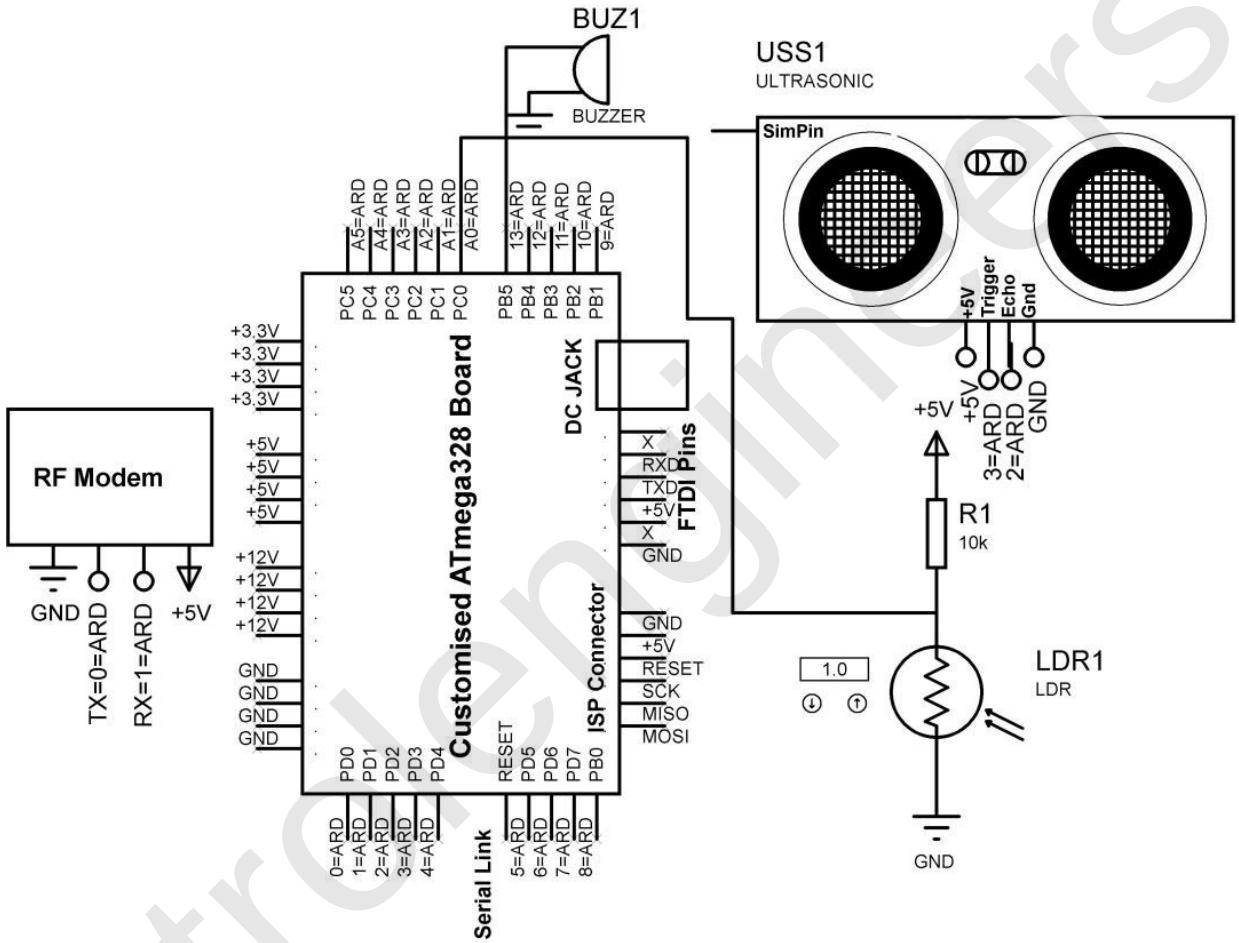

Figure 9.3: *Device on the blind stick*

Program

This section describes the programs to design the system including the program code for the stick finder and blind stick.

Code for stick finder

Refer to the following code:

```
int switch_pin=5;
int switch_state=0;
void setup()
{
 Serial.begin(9600); // initialize serial communication
 pinMode(switch_pin, INPUT); // assign pin as an input
}

void loop()
{
 int switch_state=digitalRead(switch_pin); // read switch
 if (switch_state== HIGH)
{
 Serial.write(100); // write value serial
 delay(20); // delay 20 mSec
}
else
{
 Serial.write(200); // write value serial
 delay(20); // delay 20 mSec
}
}
```

Code for blind stick

```
const int trigger = 3; //Trigger pin of ultrasonic Sesnor
const int echo = 2; //Echo pin of ultrasonic Sesnor
const int Buzz = 13; //attach buzzer to pin 13
const int Light = A0; //attach ligjht sensor to A0 pin

long time_taken;
int dist;
int Signal;
int Intens;

void setup()
{
Serial.begin(9600); // initialize serial communication
pinMode(Buzz,OUTPUT); // assign pin as an OUTPUT
digitalWrite(Buzz,LOW); // make pin to LOW
```

```
pinMode(trigger, OUTPUT); // assign pin as an OUTPUT
pinMode(echo, INPUT); // assign pin as an INPUT
}
void calculate_distance(int trigger, int echo)
{
digitalWrite(trigger, LOW); // make pin to LOW
delayMicroseconds(2); // delay of 2 uSec
digitalWrite(trigger, HIGH); // make pin to HIGH
delayMicroseconds(10); // delay of 10 uSec
digitalWrite(trigger, LOW); // make pin to LOW

time_taken = pulseIn(echo, HIGH); // calculate time
dist= time_taken*0.034/2;
if (dist>300)
dist=300;
}

void loop()
{
calculate_distance(trigger,echo);
Intens = analogRead(Light); // read analog pin
int check_from_remote=Serial.read(); // read serial data
if (check_from_remote == 100) //If remote pressed
{
 Serial.print(similar_count); // print value on serial
 Serial.println("Remote Pressed"); // print string on serial
 digitalWrite(Buzz,HIGH); // make pin to HIGH
 delay(3000); // delay 3000 m Sec
 digitalWrite(Buzz,LOW); // make pin to LOW
}
if (Intens<200)
{
 Serial.print(Intens);
 Serial.println("Bright Light");
 digitalWrite(Buzz,HIGH); // make pin to HIGH
 delay(200); // delay 200 m Sec
 digitalWrite(Buzz,LOW); // make pin to LOW
 delay(200); // delay 200 m Sec
```

```
 digitalWrite(Buzz,HIGH); // make pin to HIGH
 delay(200); // delay 200 m Sec
 digitalWrite(Buzz,LOW); // make pin to LOW
 delay(200); // delay 200 m Sec
 delay(500); // delay 500 m Sec
}

if (Intens>800)//If very bright
{
 Serial.print(Intens);
 Serial.println("Low Light");
 digitalWrite(Buzz,HIGH); // make pin to HIGH
 delay(500); // delay 500 mSec
 digitalWrite(Buzz,LOW); // make pin to LOW
 delay(500); // delay 500 mSec
 digitalWrite(Buzz,HIGH); // make pin to HIGH
 delay(500); // delay 500 mSec
 digitalWrite(Buzz,LOW); // make pin to LOW
 delay(500); // delay 500 mSec
}

if (dist<50)
{
 Serial.print(dist); // print value on serial
 Serial.println("Object Alert"); // print string on serial
 digitalWrite(Buzz,HIGH); // make pin to HIGH
 for (int i=dist; i>0; i--)
 delay(10); // delay 10 mSec
 digitalWrite(Buzz,LOW); // make pin to LOW
 for (int i=dist; i>0; i--)
 delay(10); // delay 10 mSec
}

}
```

Conclusion

This chapter concludes an intelligent system for the blind. The system is described with the help ofa circuit diagram and program.

CHAPTER 10

Sign-in to Speech Using the IoTs

The main objective is to translate the sign language to text/speech. This chapter illustrates the sign to text conversion. This step-by-step process helps the user to configure on the X86 and ARM platform. The methodology for the sign feature extraction and pre-processing of information is discussed.sed and implement

Introduction

The framework presents productive and quick procedures for distinguishing proof of the handmotion speaking to letters in order of the Sign Language.

Figure 10.1 shows the flow diagram to convert sign to speech recognition:

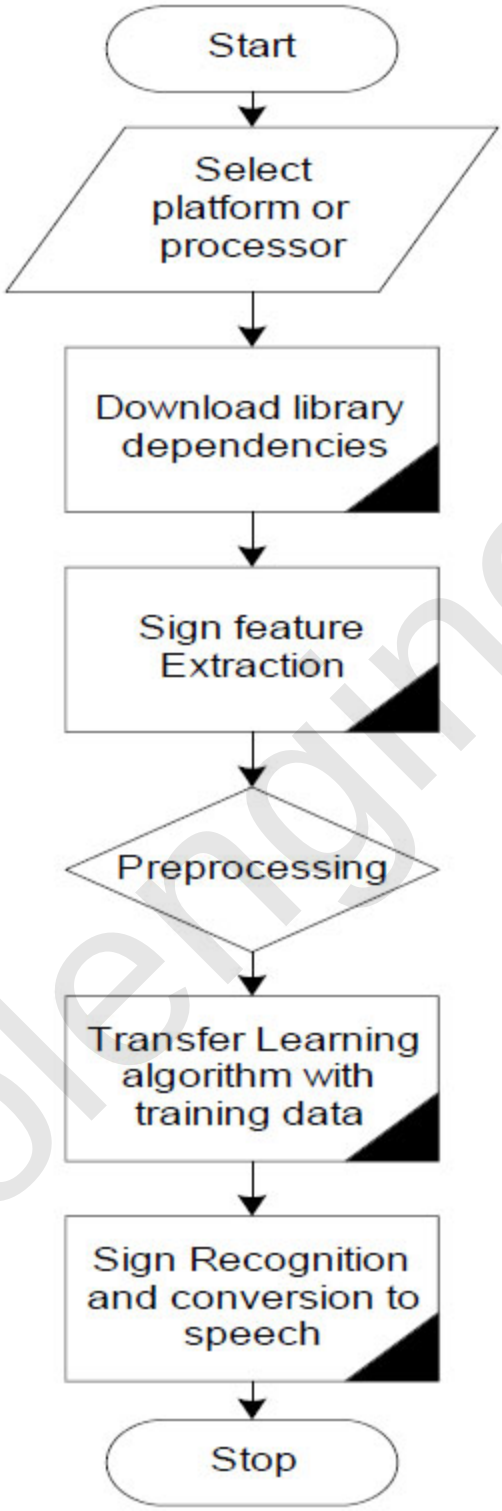

Figure 10.1: *Flow diagram for sign to speech or text conversion*

Figure 10.2 shows the steps to convert sign to speech which involves pre-processing, the CNN algorithm and post processing:

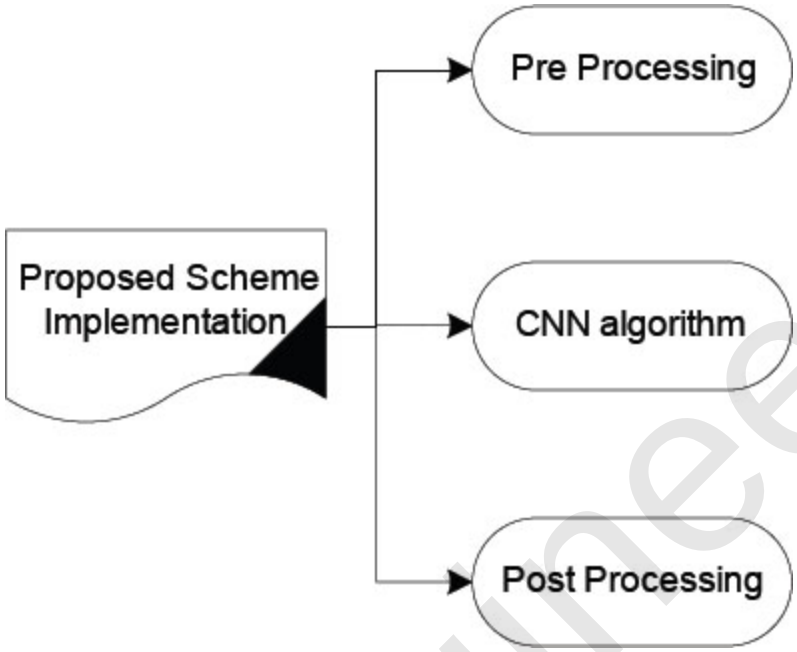

Figure 10.2: *Steps involved in the process*

Installing dependencies on the controller platform

The code is designed and developed with Python 3.5. Before running the main.py file library, dependencies need to be installed on the IDE:

1. Installation of OpenCV 3.1.0 by writing **`pip install OpenCV 3.1.0.`**
2. Installation of Keras 2.0.8.
3. Installation of Tensorflow 1.11 (CPU version); it is also compatible with GPU version.
4. Installation of numpy 1.15.2 (**`pip install numpy 1.15.2`**).
5. Installation of joblib 0.10.3 (**`pip install joblib 0.10.3`**).

The sign feature extraction uses the pre-trained models in Keras was run on an AWS EC2 p2.8xlarge instance with the Bitfusion Ubuntu 14 TensorFlow-2017 AMI. Packages updation from Python 2 to Python 3 has been done during execution.

Before running the final script, training data needs to be updated:

```
$ python live_demo.py --help
```

Installing the requirements on X86:

1. Installation of OpenCV 3.1.0 by writing `pip install OpenCV 3.1.0`.
2. Installation of Keras 2.0.8.
3. Installation of Tensorflow 1.11 (CPU version); it is also compatible with GPU version.
4. Installation of numpy 1.15.2 (`pip install numpy 1.15.2`).
5. Installation of joblib 0.10.3 (`pip install joblib 0.10.3`).
6. Go to terminal of cmd depending on your OS.
7. If you have an NVidia GPU, then make sure you have the prerequisites for the Tensorflow GPU installation (Refer to the official site). Then, use this command:

   ```
   pip install -r requirements_gpu.txt
   ```

8. If you do not have a GPU, then use this command:

   ```
   pip install -r requirements_cpu.txt
   ```

Creating a gesture

1. `python set_hand_hist.py`
   - A windows **Set hand histogram** will appear.
   - **Set hand histogram** will have 50 squares (5x10).
   - Put your hand in those squares. Make sure your hand covers all the squares.
   - Press *c*. 1 and the other window will appear **Thresh**.
   - On pressing *c,* only white patches corresponding to the parts of the image which has your skin color should appear on the **Thresh** window.
   - Make sure all the squares are covered by your hand.
   - If you are not successful, then move your hand a little bit and press *c* again. Repeat this until you get a good histogram.
   - After you get a good histogram, press *s* to save the histogram. All the windows close.

2. Then, an OpenCV window called **Capturing gestures** which will appear. In the webcam feed, you will see a green window (inside which you will have to do your gesture) and a counter that counts the number of pictures stored.

   `python create_gestures.py`

3. `python flip_images.py`
4. Load the images by giving the command: `python load_images.py.`
5. The following gestures will be shown:

Evaluation parameters	Advantages	disadvantages
High capacity	High	Low
Robustness	High	Low
Perceptual precision	High	Low
Temper resistance	High	Low
Computational complexity	Low	High

Table 10.1: *Evaluation parameters*

The preceding table illustrates the performance of the sign to speech conversion in both the X86 and ARM platforms. The evaluation parameters are mentioned in the table. From the preceding table, it is concluded that the performance on the ARM platform is better than X86.

Conclusion

This chapter gives a brief idea about the process of sign to speech conversion. In this chapter, the step-by-step process from installation of dependency library and execution of the main code has been explained. The methodology gives the overall idea about the process of execution of sign to text. The preceding experiment is conducted for ISL andASL. The performance analysis of both the architectures, that is, X86 and ARM are illustrated.

CHAPTER 11

Windows 10 on Raspberry

Nowadays, the Linux operating system is the choice for almost all new embedded device projects. Linux provides a powerful, flexible kernel, and open runtime platform which is being improved by the Linux community to support new processors, buses, devices, and protocols. At the same time,Windows 10 does have its own advantage. It is reliable with the architecture infrastructure, application prototyping, optimization, and deployment. This chapter demonstrates the use of Windows 10 OS on.

Introduction

Multi-functional smartphones are being dramatically increased in the wireless phone market [1]. Mobile communication technology is undoubtedly one of the most significant developments in the electronic system field in recent years. To provide mobile data transmission, where people can transmit and receive information wherever they are and whenever they like to, mobility is at the heart of these wireless systems. The size and the user interface of mobile devices are the main concerns in the design of mobile devices.

Raspberry Pi

Raspberry Pi is a little, multifunctional credit size PC working in Linux OS. The major and most evident contrast between the Pi and PC is that the work area will have a processor from either Intel or AMD at its center, though Pi has an ARM-based CPU. The CPU is the part that really runs the project. The following diagram shows Raspberry Pi and its components:

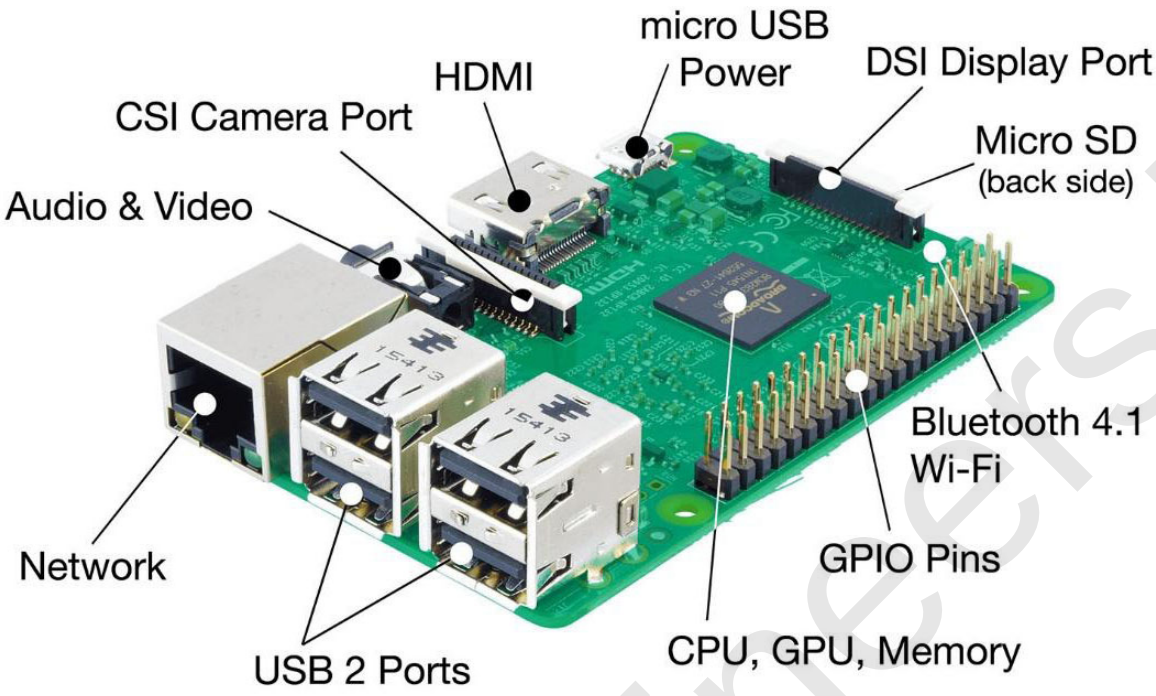

Figure 11.1: *Raspberry Pi with components*

Before a CPU can run a program, it must be translated into a language that the CPU can understand. The model B Pi has 512MB of RAM. It supports the Ethernet cable to access the network and the USB cable to interface with peripheral devices. RCA video (works with older TVs), 3.5mm audio standard headphone socket, HDMI audio andvideo (works with modern TVs and DVI monitors) are the peripheral devices.

Linux kernel architecture

There are three different layers in the Linux kernel. At the top level, SCI (system call interface), the significance of this layer is to read and write instructions and socket calls. Then, there are architecture dependent and architecture independent layers.

The device drivers have the source codes for the Linux kernel. The arch subdirectory is architecture-dependent and contains subdirectories for various architectures of the machine.

The cross compiler provides the platform to generate and execute codes for a target in which the compiler is running. Cross compilation environments support **Application Binary Interface** (**ABI**) and **Embedded Application Binary interface** (**EABI**). The ABI represents higher level language to machine level

language. For different targets, the Linux kernel gets updated with tool chains for different applications.

The process of installing Windows 10 on a Raspberry Pi involves downloading the installer, drivers, and the OS itself from a variety of unofficial sources. Everything from the **Start** menu to the **Edge** browser operates in slow motion. Installing it will probably take at least an hour, and some blue screens may be experienced. After compilation of Windows 10 on Raspberry Pi, and if the Ethernet connection is available, web can be surfed on it.

Windows 10 on Raspberry Pi

Hardware required:

1. Raspberry Pi 3, 3B or 3B+.
2. High-speed, class 10 microSD card of at least 16GB, preferably A1 speed.
3. A good 2.5-amp power adapter for the Pi.

Software required:

1. SD card formatter.
2. The WoA (Windows on Arm) installer for Raspberry Pi.
3. The latest core package.
4. Windows 10 Arm image (tested with version 17134).
5. Latest Ethernet driver.

SD card formatter

It is used to format the SD card. It will erase all the previous data stored on it. It is an open source software that can be easily downloaded.

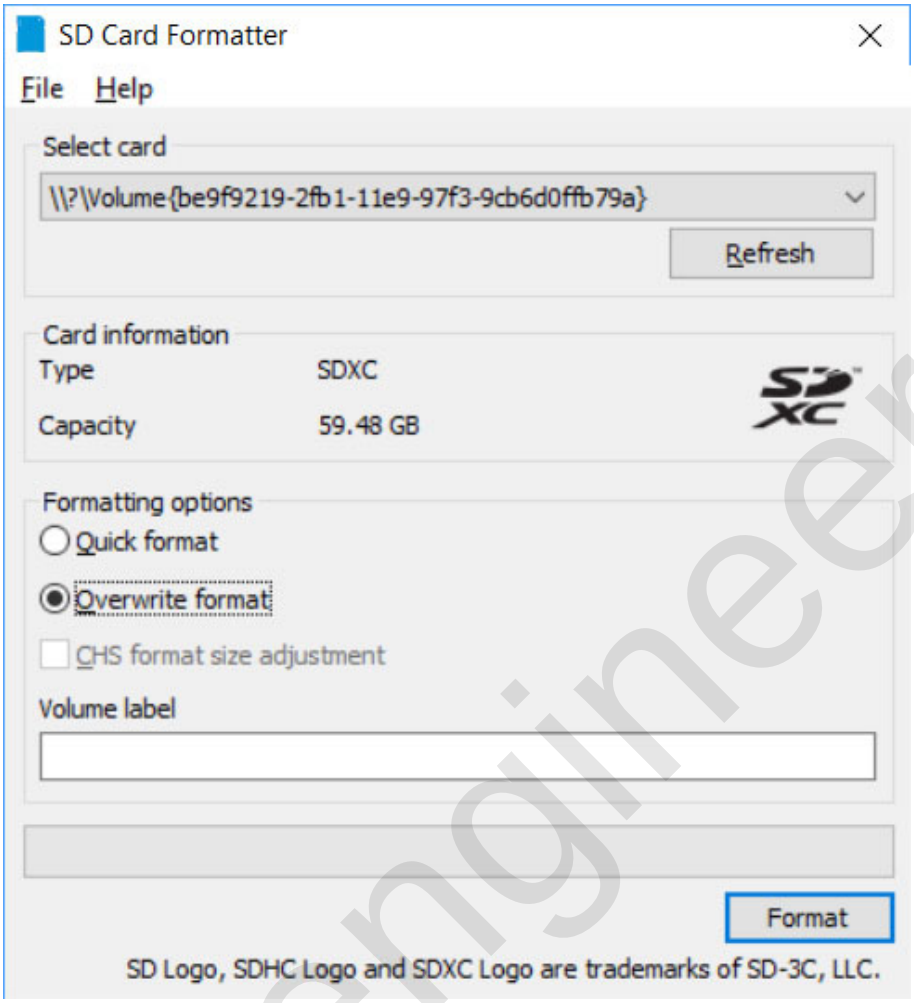

Figure 11.2: *SD card formatting software*

1. Downloading of the Windows image: The link to download the window image for Raspberry Pi (https://uupdump.ml/) is mentioned. While downloading, makes sure the image is compatible with the ARM platform of 64 bit. Both the new and old version can be downloaded. Preference could be given as per individual. In *Figure 11.3*, the web link interface is shown as follows:

Figure 11.3: *Downloading Windows images*

2. Extracting the Z**IPf**ile: After downloading the file, extract the file into the folder. Make sure all the files have been extracted properly. During extracting the file, you may get a notification or warning **not to run the script.**Then, click on run to run it. It may take several minutes from 15 to 20 minutes. Relax and when it is done, it will create the ISO file.

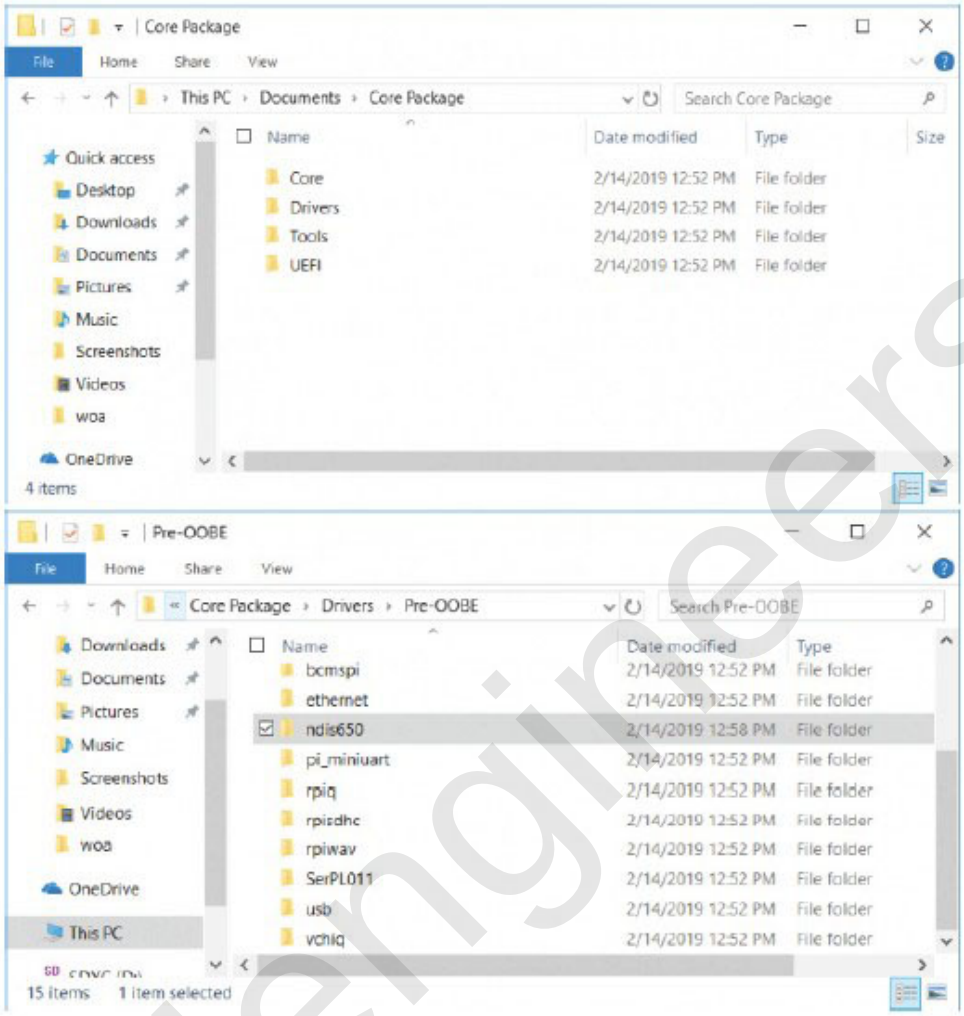

Figure 11.4: *During extraction of file*

3. Now, unzip the core files into the same folder.
4. Download the ZIP file of the Ethernet driver and extract the file to the same folder.
5. Copy the **`ndis650`** folder from the Ethernet driver into the core package's **`/drivers/Pre-OOBE`** subfolder.
6. Edit the **`config.txt`** file in the core package's **`UEFI`** subfolder, and add **`disable_overscan=1`** to a line at the end if it's not already there.
7. Create a new ZIP file from the core package folder (for example, **`CorePackage1.4.zip`**).
8. Rightclick on the ISO file that the script created for you in step 3, and select Mount. This will assign the ISO file a drive letter (for example, **`G:`**), so you can later browse it from within the WoA app.

9. Download, unzip, and run the WoA installer. The Windows smart screen will warn you not to run it. Click on **More Info** and then **Run Anyway**.

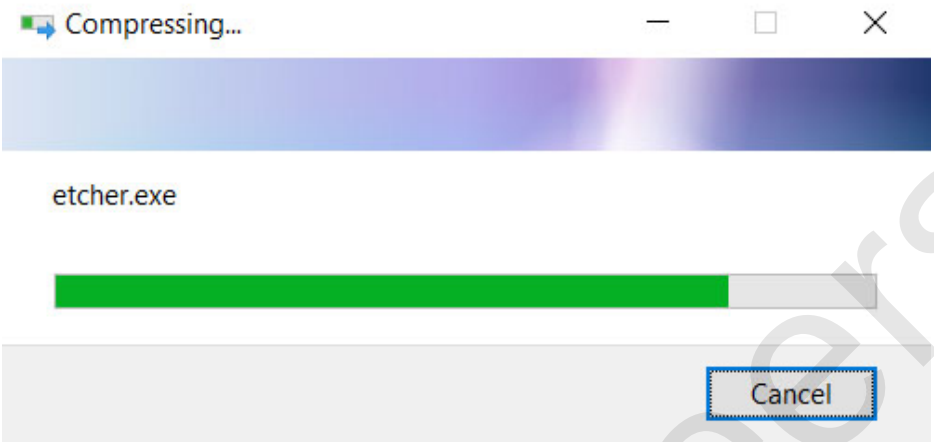

Figure 11.5: *Generating .exe file*

10. Go to the **Advanced** tab, and click on **Import Core package** and then **select the Core Package zip file**.
11. Under the **Windows Deployment**tab, click on **Browse** and then navigate to the ISO file youmounted to select `install.wim` from the `/sources` subfolder.
12. Click on **Deploy**. This may take several minutes.

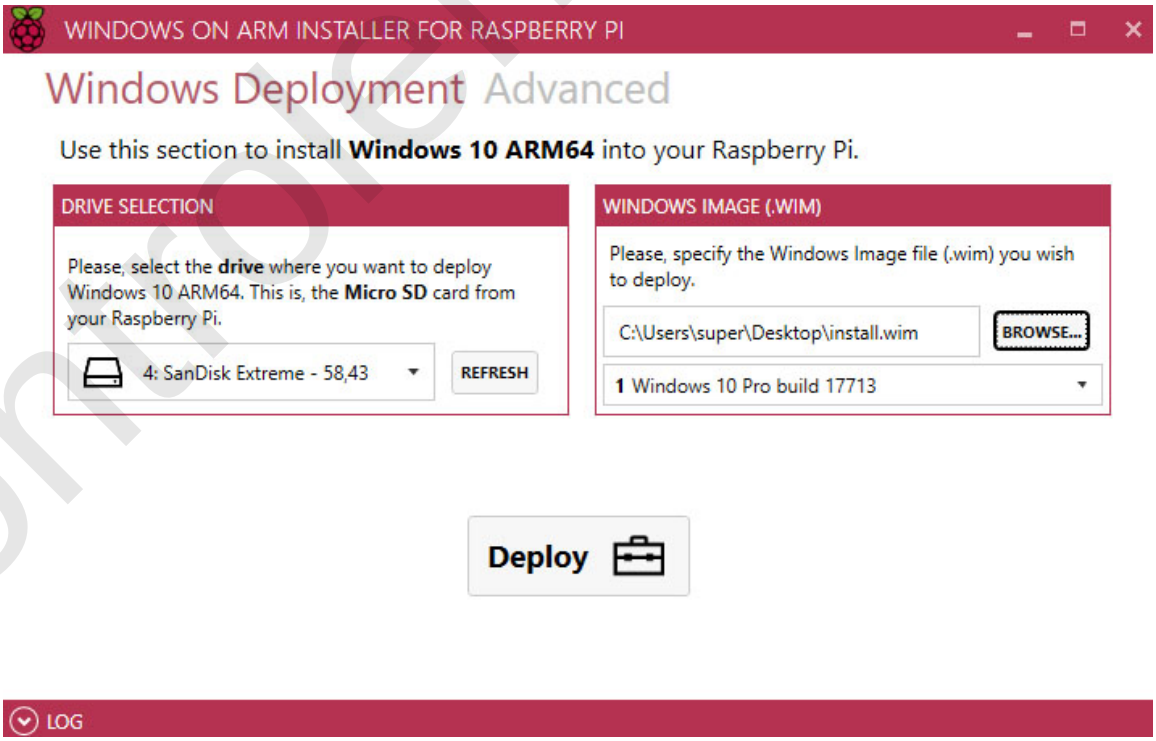

Figure 11.6: *Windows deployment*

13. Insert the microSD card into your Raspberry Pi and then boot it with a keyboard and mouse attached.
14. Type `exit` at the Shell prompt, and hit *Enter*. This will take you to the **BIOS** menu. You need to do this because at the first boot, your Pi goes to this UEFI shell instead of booting off the Windows 10 partition.
15. Select **Device Manager**, then **Rasperry PI Configuration**, and then **HypDxe Configuration**.
16. Make sure **System Boot Mode** is setto **Boot in EL1**. That may be the default.
17. Navigate to the **Device Manager** | **Raspberry Pi Configuration** | **Chipset Configuration** menu, and select **Max** from the **CPU clock** menu.
18. Press *ESC* a couple of times, and select *Y* to save your changes. Then, hit *ESC* a couple of times to return to the home screen.
19. Navigate to the **Boot Maintenance Manager**| **Boot Options** | **Change Boot Order** menu.
20. Hit *Enter* and use the - key to move **SD/MMC** on **Broadcom SDHOST** to the top. Hit *Enter*, then **Commit Changes** and exit. This will make sure your computer always boot straight to Windows and not to the UEFI shell.
21. From the home screen, navigate to **Boot Manager**, and select **SD/MMC** on **Broadcom SDHOST**. You may be asked to hit *Enter* to reset. Now, your Pi will boot off of the Windows partition.
22. Wait a really long time while Windows installation may take several minutes. Wait until the installation gets completed [19].
23. Complete the Windows 10 install process. The installation .process will navigate with various options. Choose it correctly and proceed. Finally, after several minutes, the Windows 10 interface will be appear on the screen.

The entire activity of installing Windows 10 on the SD card might take 1:30 hours to 2 hours. Make sure you have connected the Raspberry Pi to the internet via Ethernet. After the installation, now Pi is ready to use. It has been seen that it is slower compared to Raspbian and PiLFS.

Performance testing and evaluation

To check the performance of Windows 10 on Raspberry, three algorithms have been used. Performance evaluation can be done on the custom kernel for the following details to identify the execution seed:

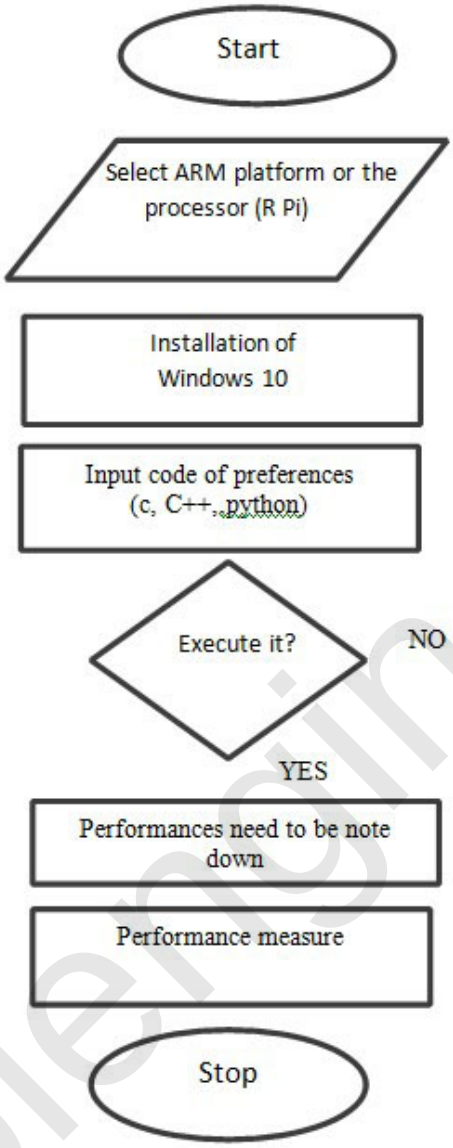

Figure 11.7: *Methodology of evaluation of Windows 10 OS on ARM architecture*

The results are shown below in *Table 11.1* and *Table 11.2*:

Evaluation parameters	Bubble sort	Binary search	Merge sort
CPU cycles used	2.2x1018	3.2x1018	2.5x1018
Context switch time in ms	1245	1324	1367
Task clock cycle	3456	3589	3678
Cache hit time in ms	976	976	945
Overall performance in percentage	75	69	79

Table 11.1: *Using Python on Raspbian*

Evaluation parameters	Bubble sort	Binary search	Merge sort
CPU cycles used	2.1x1018	2.9x1018	2.3x1018
Context switch time in ms	1189	1201	1235
Task clock cycle	3879	3987	3794
Cache hit time in ms	1125	1192	1232
Overall performance in percentage	84	73	65

Table 11.2: *Using Python on Windows 10*

Conclusion

In this chapter, a detailed installation of Windows 10 on the ARM platform is illustrated. Further, a performance measure is conducted to verify the coding efficiency and CPU clock cycles uses. A comparative analysis is carried out.

CHAPTER 12

Wireless Video Surveillance Robot Using Raspberry Pi

Live streaming and video surveillance plays an important role in terms of safety and security. If a compact system is designed, which is capable of doing video surveillance and live streaming, then it would be a great help in various fields. A robot is designed and implemented that is capable of streaming the live video and is very compact in size also. The major role is being played by RaspberryPi which is a credit card sized motherboard. It runs on the Raspbian operating system. The camera used in this project is Pi-cam that can further tilt and rotate. The technology used in this robot is Wi-Fi 802.11 for controlling the robot and camera module.

Introduction

This section describes the design of a robot with raspberry Pi and cloud with a live video stream that is obtained over an**IP** (**Internet Protocol**) address.

Figure 12.1 shows the detailed block diagram of a surveillance robot. The complete system has two parts: robot and cloud server. The robot comprises Raspberry Pi, battery, camera, NuttyFi, motor driver, and motors.

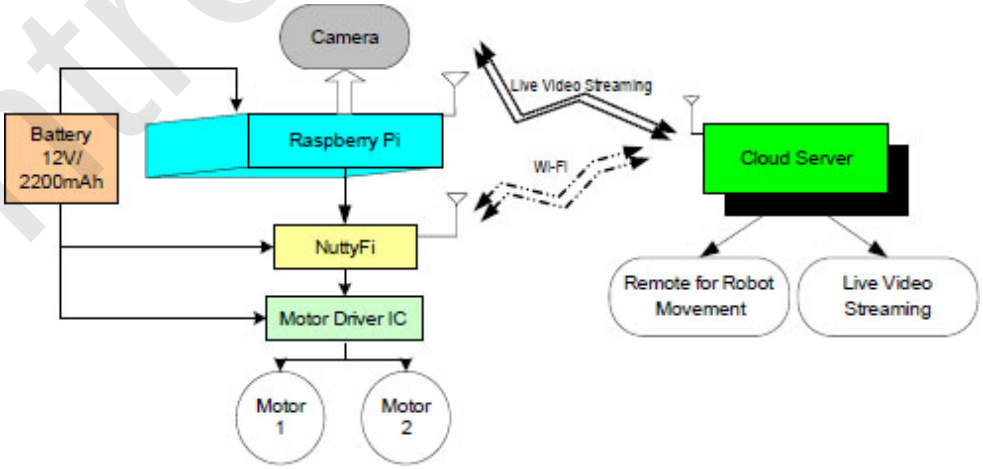

Figure 12.1: *Block diagram of a video surveillance robot*

To begin with, the operating system needs to be uploaded on RaspberryPi. It can be Raspbian Jessie or NOOBS using an SD card. Once the OS is loaded, the system needs to be booted by connecting it to the monitor with the help of the HDMI cable and adding devices like the USB keyboard, Pi-Cam, and mouse with the system via USB ports that are available on RaspberryPi. Once the system is booted, it enables the camera module from the terminal. Now, use the LX terminal to give commands to type the commands and all the commands carefully give permissions to the files which need to be downloaded.

The motors are connected with the PWM pins of RaspberryPi. With the help of this, the quality of video streaming is improved for the robot and the number of frames is transmitted per second. *Table 12.1* gives the detailed list of components that is required for making this robot:

Component/Specification	Quantity
12V/1A adaptor	1
Jumper wire M-M	20
Jumper wire M-F	20
Jumper wire F-F	20
+12V to +5V converter	1
Raspberry Pi 3B+	1
NuttyFi	1
Breakout board	1
USB camera	1

Table 12.1: *Component list*

Circuit diagram and connection

1. Connect power supply 12V/1A to Raspberry Pi and NuttyFi.
2. Connect 4,5,12 and 12 pins of L293D to GND.
3. Connect 1,9 and 16 pins to +5V.

4. Connect 2, 7, 10 and 15 pins of L293D to D1, D2, D3 and D4 pins of NuttyFi.
5. Connect 3 and 6 pins of L29D to two pins of the first DC motor.
6. Connect 11 and 14 pins of L29D to two pins of the second DC motor.
7. Connect the camera to the port of camera on Raspberry Pi.

Figure 12.2 gives the detailed circuit diagram of the robot:

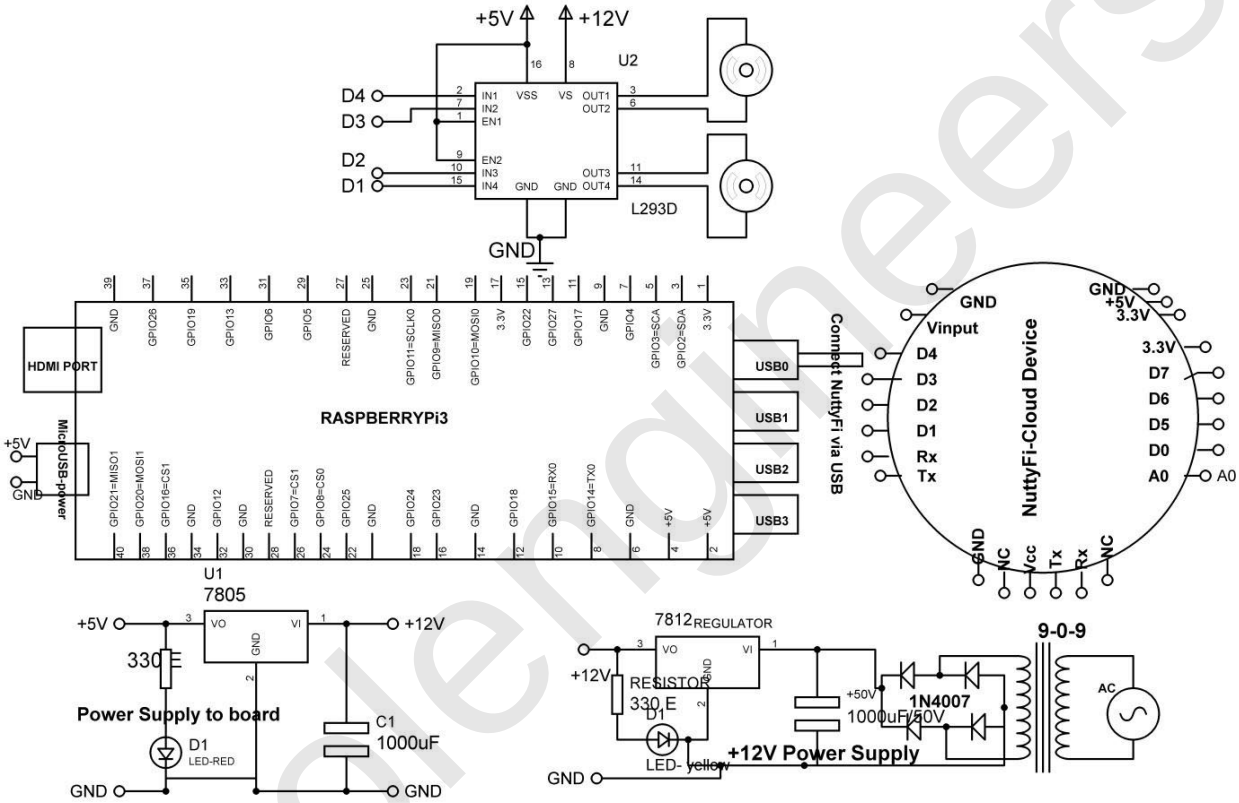

Figure 12.2: *Circuit diagram of a video surveillance robot*

Application/data logger

The following program is the code for NuttyFi:

```
#include <ESP8266WiFi.h> // add ESP library
#include <BlynkSimpleEsp8266.h>// add blynk library

char auth[] = "8507cac915f04a1bb4b00987e420afa0";//add token
here
char ssid[] = "SchematicsMicroelectronics"; // add hotspot
address
```

```
char pass[] = "XXXXXXX"; // add hotspot password
BlynkTimer timer; // add timer in blynk
int M1_POSITIVE=D1;//D1 as pin 1 of motor driver
int M1_NEGATIVE=D2;//D2 as pin 2of motor driver
int M2_POSITIVE=D3;//D3 as pin 3 of motor driver
int M2_NEGATIVE=D4;//D4 as pin 4 of motor driver
WidgetLCD LCD_BLYNK(V0); // assign virtual pin V0 to blynk LCD
BLYNK_WRITE(V1)
{
 int FORWARD = param.asInt(); // read V1
 if(FORWARD==HIGH) // check condition
 {
  digitalWrite(M1_POSITIVE,HIGH); // make pin1 to HIGH
  digitalWrite(M1_NEGATIVE,LOW); // make pin2 to LOW
  digitalWrite(M2_POSITIVE,HIGH); // make pin3 to HIGH
  digitalWrite(M2_NEGATIVE,LOW); // make pin4 to Low
  LCD_BLYNK.print(0,0,"ROBO CNTRL SYS"); // print string on
  Blynk LCD
  LCD_BLYNK.print(0,1,"FORWARD"); // print string on Blynk LCD
  delay(10); // wait for 10 mSec
 }
}

BLYNK_WRITE(V2)
{
 int REVERSE = param.asInt(); // read virtual pin V2
 if(REVERSE==HIGH) // check condition
 {
  digitalWrite(M1_POSITIVE,LOW); // make pin1 to LOW
  digitalWrite(M1_NEGATIVE,HIGH); // make pin2 to HIGH
  digitalWrite(M2_POSITIVE,LOW); // make pin3 to LOW
  digitalWrite(M2_NEGATIVE,HIGH); // make pin4 to HIGH
  LCD_BLYNK.print(0,0,"ROBO CNTRL SYS"); // print string on
  Blynk LCD
  LCD_BLYNK.print(0,1,"REVERSE"); // print string on Blynk LCD
  delay(10); // wait for 10 mSec
 }
}
```

```
BLYNK_WRITE(V3)
{
 int LEFT = param.asInt(); // read virtual pin V2
 if(LEFT==HIGH) // check state
 {
  digitalWrite(M1_POSITIVE,HIGH); // make pin1 to HIGH
  digitalWrite(M1_NEGATIVE,LOW); // make pin2 to LOW
  digitalWrite(M2_POSITIVE,LOW); // make pin3 to LOW
  digitalWrite(M2_NEGATIVE,LOW); // make pin4 to LOW
  LCD_BLYNK.print(0,0,"ROBO CNTRL SYS"); // print string on
  Blynk LCD
  LCD_BLYNK.print(0,1,"LEFT "); // print string on Blynk LCD
  delay(10); // wait for 10 mSec
 }
}

BLYNK_WRITE(V4)
{
 int RIGHT = param.asInt(); // read V4 pin
 if(RIGHT==HIGH) // check state
 {
  digitalWrite(M1_POSITIVE,LOW); // make pin1 to LOW
  digitalWrite(M1_NEGATIVE,LOW); // make pin2 to LOW
  digitalWrite(M2_POSITIVE,HIGH); // make pin3 to HIGH
  digitalWrite(M2_NEGATIVE,LOW); // make pin4 to LOW
  LCD_BLYNK.print(0,0,"ROBO CNTRL SYS"); // print string on
  BLYNK APP
  LCD_BLYNK.print(0,1,"RIGHT "); // print string on BLYNK APP
  delay(10); // wait for 10 mSec
 }
}

BLYNK_WRITE(V5)
{
 int STOP = param.asInt();
 if(STOP==HIGH)
 {
  digitalWrite(M1_POSITIVE,LOW); // make pin1 to LOW
```

```
   digitalWrite(M1_NEGATIVE,LOW); // make pin2 to LOW
   digitalWrite(M2_POSITIVE,LOW); // make pin3 to LOW
   digitalWrite(M2_NEGATIVE,LOW); // make pin4 to LOW
   LCD_BLYNK.print(0,0,"ROBO CNTRL SYS"); // print string on
   BLYNK APP
   LCD_BLYNK.print(0,1,"STOP "); // print string on BLYNK APP
   delay(10); // wait for 10 mSec
  }
}

void setup()
{
 Serial.begin(9600); // initialize serial communication
 Blynk.begin(auth, ssid, pass); // begin BLYNK
 pinMode(M1_POSITIVE,OUTPUT); // pin1 of motor driver assign as
 output
 pinMode(M1_NEGATIVE,OUTPUT); // pin2 of motor driver assign as
 output
 pinMode(M2_POSITIVE,OUTPUT); // pin3 of motor driver assign as
 output
 pinMode(M2_NEGATIVE,OUTPUT); // pin4 of motor driver assign as
 output
 }

void loop()
{
 Blynk.run(); // run BLYNK app
 timer.run(); // Initiates BlynkTimer

}
```

Raspberry Pi and its installation

Raspberry Pi is a powerful minicomputer which is specifically designed by the researchers to make work easier. It has got a large number of components that are extremely useful for computer-based projects like the Ethernet cable, SD card slot, Wi-Fi antenna ports, USB ports, and many more. *Figure 12.3* shows the basic requirement of a Raspberry Pi-based system to keep it running:

Figure 12.3: *Raspberry Pi, power supply adaptor, and SD card*

- Monitor or TV with HDMI or DVI input
- Video link from the Pi's HDMI port to the screen's information port
- USB console and mouse
- Ethernet link**Power supply**: Raspberry Pi works with a MicroUSB cable which has the capability of supplying at least 700mA at 5V. In order to select the power supply, one can see the details mentioned on the supply.
- **SD card**: The SD card used for the system should always be class 4 and above and minimum 4 GB and above. The good brand of SD cards also improves the efficiency of the system. Class 10 SD cards with 16 GB and above are always suggested for better performance.
- **Screen**: The screen is also suggested to be connected and this screen can be any television or monitor connected with the help of HDMI or DVI input. Further, the laptop can also be connected and the laptop screen can also be used.
- **Keyboard and mouse**: USB peripherals ought to be lowcontrol under 0.1A each. Most consoles and mice are fine, yet maybe a modest number (especially those with brilliant LEDs) ought to be connected to a different controlled USB center point, which is connected to Pi.

Writing an SD card with NOOBS

1. Download the NOOBS document record from http://www.raspberrypi.org/downloads, separate it, and spot it on a SD card.
2. Once you have downloaded the NOOBS document record, separate it and duplicate the folder contents onto the SD card.
3. Put the SD card containing the separated NOOBS records into your Raspberry Pi and afterwards power up your Raspberry Pi. When it boots, a window appears. *Figure 12.4* shows the Raspbian window:

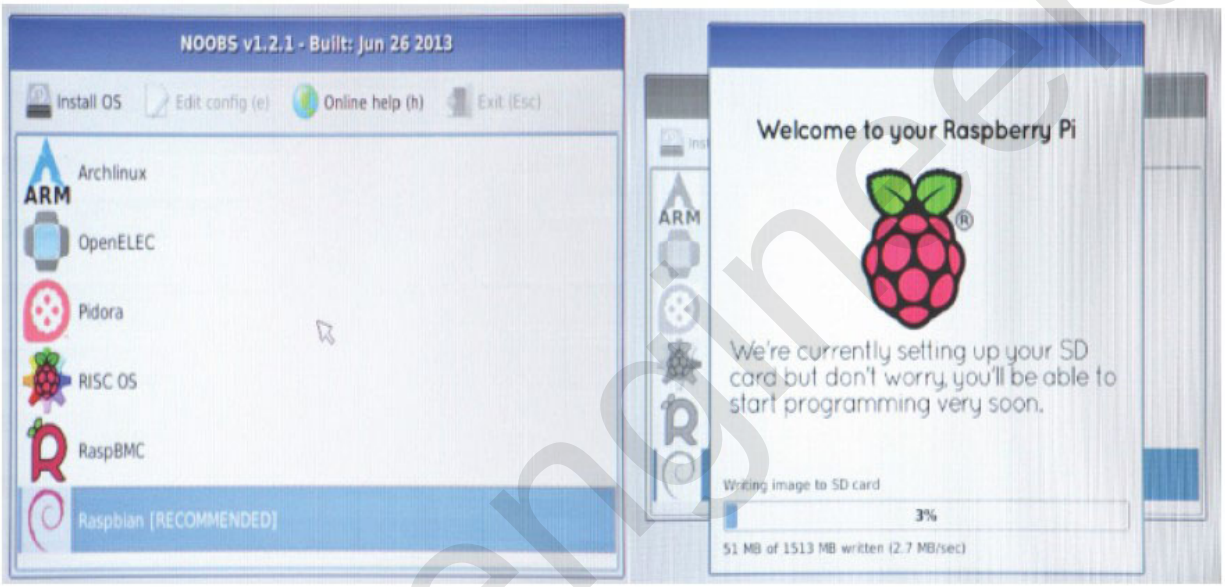

Figure 12.4: *View of Raspbian window*

Few terminal commands for Raspberry Pi are as follows:

- Update package lists

  `sudo apt-get update`
- Download and install updated packages

  `sudo apt-get upgrade`
- Clean old package files:

  `sudo apt-get clean`
- The Raspberry Pi configuration tool:

  `sudo raspi-config`
- List directory contents:

```
ls
```

- Change directories:

```
cd
```

- Create a directory:

```
mkdir
```

- Remove a directory:

```
rmdir
```

- Move a file:

```
mv
```

- Show a tree of directories:

```
tree -d
```

- Show the current directory:

```
pwd
```

- Clearing the terminal window:

```
clear
```

- Shut down the Raspberry Pi:

```
sudo halt
```

The programming of Raspberry-Pi is done with the help of command lines in the LX terminal of the Raspbian OS. It includes the directions for downloading the server from the web and making it executable and auto start during booting to acquire the live stream video through the raspberry camera.

1. Install Raspbian and update it and make sure the camera is enabled:

```
sudo apt-get update
sudo apt-get upgrade
```

2. Install `vlc`:

```
sudo apt-get install vlc
```

3. Create a script to start the stream with the following content, or run the following command:

```
sudo nano myscript.sh
```

```
raspivid -o - -t 0 -hf -w 640 -h 360 -fps 25 | cvlc -vvv
stream:///dev/stdin     --sout     '#rtp{sdp=rtsp://:8554}'
:demux=h264
```

4. Make the script runnable:

```
sudo chmod +x myscript.sh
```

5. If you want to start the stream automatically, then you have to add the script to crontab. To make this work, I had to make another script run by `cron` (OBS! VLC can't be run as `sudo`, so make sure you're in the right `cron`). `sudo nano myscript2.sh`:

```
#!/bin/bash

/path/to/myscript.sh

Then:

sudo chmod +x myscript2.sh

crontab -e

@reboot /path/to/myscript2.sh
```

6. To watch the video stream, open VLC on a computer on the same network as the raspberry pi you are using for streaming. **Press Media|Open Network** stream and paste the following in the field:

```
rtsp://192.168.1.78:8554/
```

If you don't care about FPS (frames per second) and don't want any delay, you could use MJPEG.

Blynkapp

Follow the steps to design the Blynk app. The front end of the app for the proposed system is shown in *Figure 12.5*:

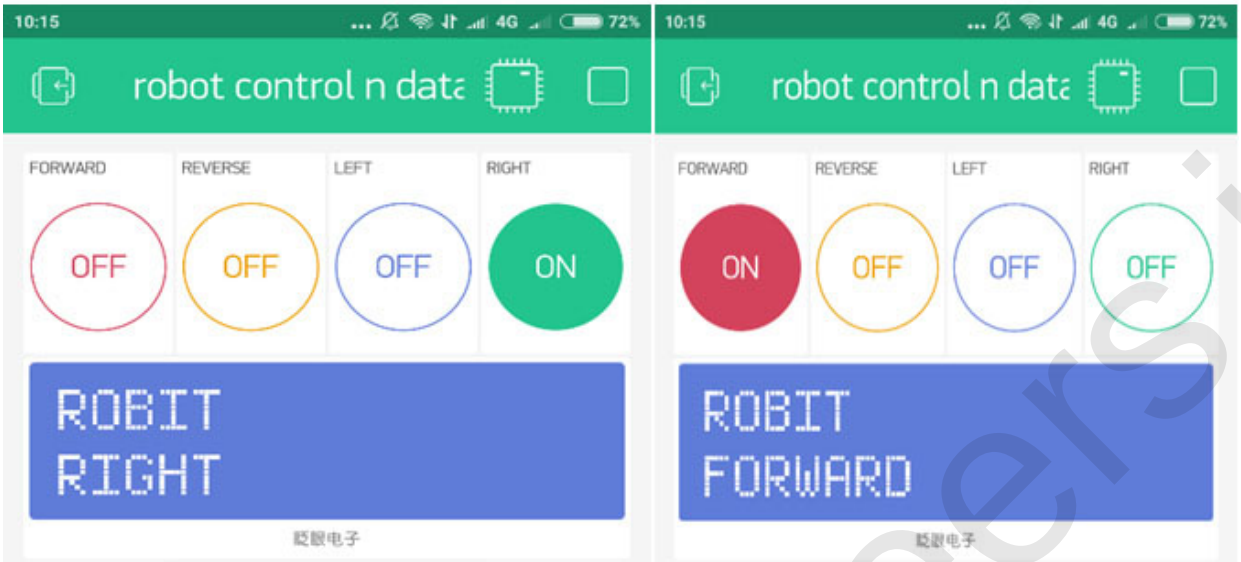

Figure 12.5: *View of the Blynk app*

Conclusion

This chapter concludes the steps to control the robot with RaspberryPi and the Blynk app. The complete system is described with the help of a circuit diagram and program.

CHAPTER 13

IoT-based Smart Camera

A camera is an optical instrument which is used to capture the image or to record the video. The data is stored in a digital system or on a photographic film. A camera consists of a lens which focuses light from the scene.

Introduction

A system is designed in such a way that it will activate the camera module only when the person is detected and will click the photo that is further sent as an attachment in the email to the registered emailID.

Figure 13.1 shows the block diagram of the IoT-based smart camera system. It comprises Raspberry Pi, PIR sensor, SD card, Pi camera, and power supply. A passive infrared or PIR or IR motion sensor is widely used for sensing the motion. The sensor detects the change in IR levels and senses the presence of human beings. The range of the PIR sensor is approximately 7 meters and 110° x 70° angle. The PIR sensor is having three pins: Ground, Digital Out, and 3-5 V DC. When no motion is detected, the digital out pin remains at low state and will go high when motion is detected and which is further used by RaspberryPi to sense it:

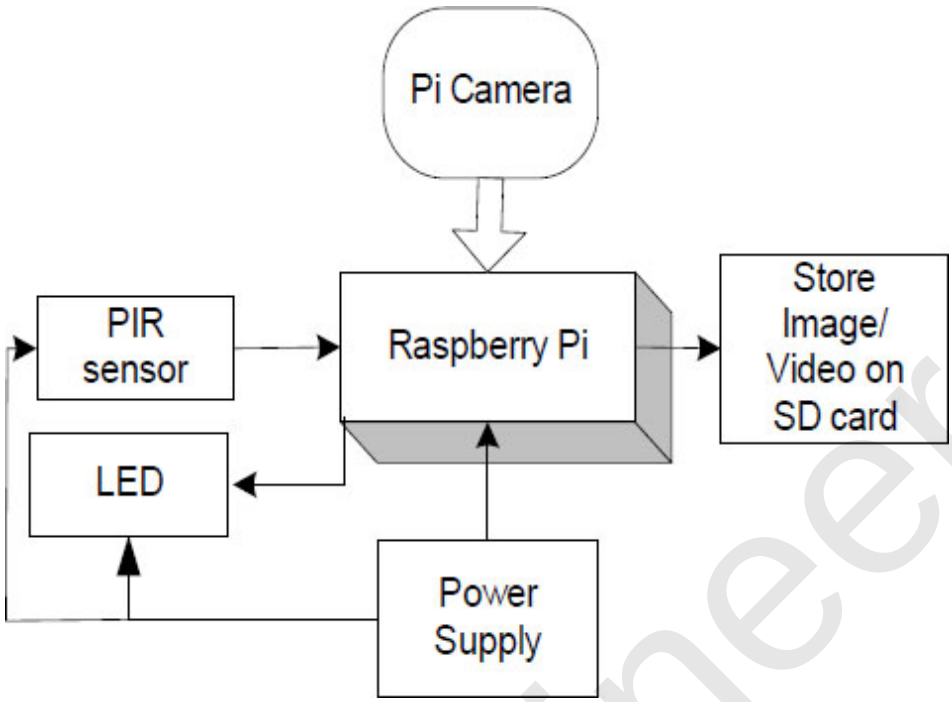

Figure 13.1: *Block diagram of the system*

One needs an inbuilt Pi-camera library to control the Pi camera:

1. Start the camera.
2. Click the picture when motion is detected with the help of the PIR sensor.
3. Save all the pictures in the folder that is located on your desktop with a time stamp.
4. Name and arrange the photos in the order they are clicked with the timestamp.
5. Exit the camera preview when the push button is pressed in order to exit the camera.

Anyone can use this PIR motion sensor along with RaspberryPi and Pi-camera for variousapplications like the home security system and after the detection of the intruder, the captured images can be sent immediately to the owner. The list of components required for this project is shown in *Table 13.1*:

Component/Specification	Quantity
12V/1A adaptor	1

Jumper wire M-M	20
Jumper wire M-F	20
Jumper wire F-F	20
+12V to +5V converter	1
Raspberry Pi 3B+	1
NuttyFi	1
Breakout board	1
USB camera	1

Table 13.1: *Component list*

Circuit diagram and connection

The detailed description of a circuit diagram is as follows:

1. Connect the power supply 12V/1A to Raspberry Pi and NuttyFi.
2. Connect the camera to the port of camera on Raspberry Pi.
3. Connect Vcc, GND, and OUT pin of the PIR sensor to +5V, GND, and GPIO21 of Raspberry Pi.
4. Connect the LED with the resistor to GPIO 3 of Raspberry Pi.

The detailed circuit diagram is shown in *Figure 13.2*:

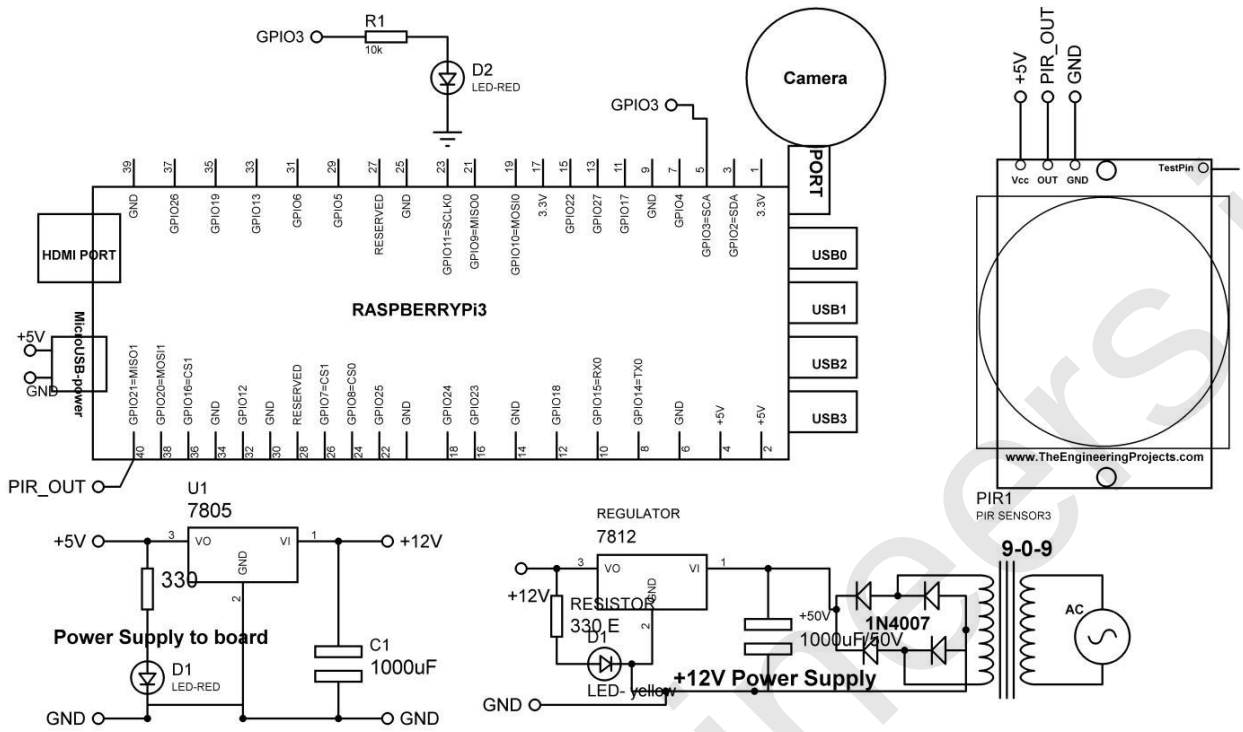

Figure 13.2: *Circuit diagram of the system*

Application/data logger

The following is the program to store the photo on the SD card:

```
import RPi.GPIO as GPIO // import library
import time // import library

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM) // set mode in BCM
GPIO.setup(3,GPIO.OUT) # connect Buzzer on GPIO3
GPIO.setup(21,GPIO.IN) # connect Motion sensor on GPIO21

import picamera // import library
with picamera.PiCamera() as camera:
camera.resolution = (1024, 768) // set camera resolution
try:
 while 1:
  x=GPIO.input(21) // read GPIO
  if(x==1):
   print("object detected") # print string on terminal
   GPIO.output(3,True) # make pin to HIGH
```

```
    camera.start_preview() // starty camera preview
    camera.start_recording('/home/pi/video.h264')
    sleep(10) // delay 10 Sec
    camera.stop_recording() # start recording
    camera.stop_preview() # stop camera preview
    time.sleep(1) # delay 1 Sec
   else:
    print("no object detected") #print string on terminal
    GPIO.output(3,False) #make pin to low
    time.sleep(1) // sleep for 1 sec
except KeyboardInterrupt:
   print("user interrupted") # print string on terminal
   GPIO.cleanup() # clean GPIO
```

The following is the program for capturing and sending the images to email ID:

```
import RPi.GPIO as gpio
import picamera
import time
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
from email.mime.image import MIMEImage

fromaddr = "rapperpranav@gmail.com" # change the email address
accordingly
toaddr = "96rakeshpandey@gmail.com"
mail = MIMEMultipart()
mail['From'] = fromaddr
mail['To'] = toaddr
mail['Subject'] = "Attachment"
body = "Please find the attachment"

led=17
pir=18
HIGH=1
LOW=0
```

```
gpio.setwarnings(False)
gpio.setmode(gpio.BCM) # set pin mode as BCM
gpio.setup(led, gpio.OUT) # assign pin as an OUTPUT
gpio.setup(pir, gpio.IN) # assign pin as an INPUT
data=""

def sendMail(data):
 mail.attach(MIMEText(body, 'plain'))
 print data # print data on terminal
 dat='%s.jpg'%data
 print (data) # print data on terminal
 attachment = open(dat, 'rb')
 image=MIMEImage(attachment.read())
 attachment.close()
 mail.attach(image)
 server = smtplib.SMTP('smtp.gmail.com', 587)
 server.starttls()
 server.login(fromaddr, "Pra@3lee")
 text = mail.as_string()
 server.sendmail(fromaddr, toaddr, text)
 server.quit() # quit from server
def capture_image():
 data= time.strftime("%d_%b_%Y|%H:%M:%S")
 camera.start_preview() # start camera preview
 time.sleep(5) # delay of 5 Sec
 print data # print dat on terminal
 camera.capture('%s.jpg'%data)
 camera.stop_preview()
 time.sleep(1) # delay of 1 Sec
 sendMail(data)# mail data

gpio.output(led, 0) # make pin tom LOW
camera = picamera.PiCamera()
camera.rotation=180 # set rotation 180
camera.awb_mode= 'auto'
camera.brightness=55 # set brightness of camera
while 1:
 if gpio.input(pir)==1: # read sensor
```

```
  gpio.output(led, HIGH) # make pin to HIGH
  capture_image() # capture image
  while(gpio.input(pir)==1):
     time.sleep(1) # delay 1 Sec
else:
  gpio.output(led, LOW) # make pin to LOW
  time.sleep(0.01) # delay 0.01 Sec
```

Conclusion

This chapter concludes the introduction to the camera and its use in the security application. A system to capture the image and share it on the authorized email address is discussed in detail with the help of circuit connections and programs.

CHAPTER 14

IoT-based Air Quality Monitoring System Using NodeMCU

This section describes the air quality monitoring system using Arduino Mini and WiFi modem (NodeMCU). Air quality is one of the important parameters to be observed for a healthy lifestyle. Even in mines and underground constructions, it is important to detect the harmful gases on time to save the human lives.

Introduction

Figure 14.1 shows the block diagram of the system. The system comprises Arduino mini, DC 12V/1Amp adaptor, 12V to 5V, 3.3V converter, BMP180 sensor, MQ135sensor, dust sampler, liquid crystal display, and NodeMCU. The objective of the system is to display the information of the BMP180 sensor, air quality using MQ135, and dust sampler of air on the liquid crystal. The sensors are interfaced to Arduino Mini. The data packet is formed with Arduino Mini, which contains all sensory information. The data packet from Arduino Mini is transferred serially to NodeMCU. The NodeMCU/WiFi modem transfers the packet information to the cloud APP/cloud server:

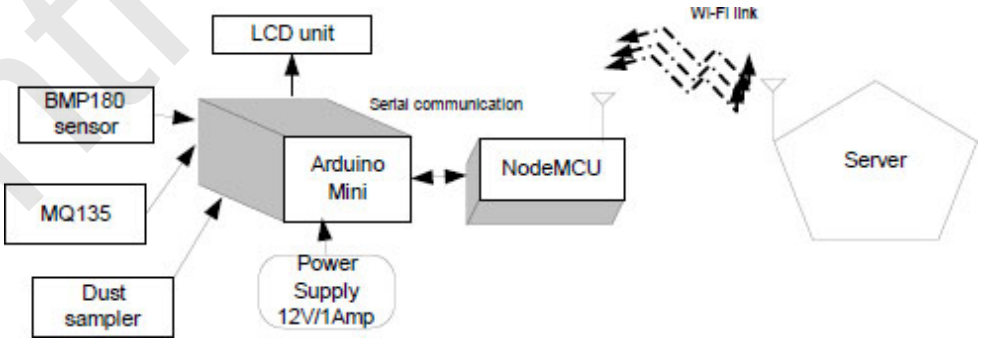

Figure 14.1: *Circuit diagram of the system*

Table 14.1 shows the component list required to develop the system:

Components	Quantity
LCD20*4	1
LCD20*4 patch	1
DC 12V/1Amp adaptor	1
12V to 5V, 3.3V converter	1
LED with 330 Ohm resistor	1
Jumper wire M to M	20
Jumper wire M to F	20
Jumper wire F to F	20
BMP180 sensor	1
MQ135sensor	1
Dust sampler	1
Arduino mini	1
NodeMCU	1
NodeMCU breakout board/Patch	1

Table 14.1: *Components list*

Note: All components are available at www.nuttyengineer.com.

Circuit diagram

The following is the detailed description of a circuit diagram which is as follows:

1. +5V pin of the power supply is connected to Vcc pin of Arduino Mini.
2. GND pin of the power supply is connected to the GND pin of Arduino Mini.
3. Pins 1 and16 of the LCD are connected to GND of the power supply.
4. Pins 2 and15 of the LCD are connected to +Vcc of the power supply.
5. Two fixed lags of POT are connected to +5V and GND of the LCD and the variable lag of POT is connected to pin 3 of the LCD.

6. RS, RW, and E pins of the LCD are connected to pins D1=12, GND and D2=11 of Ti Launch PAD.
7. D4, D5, D6, and D7 pins of the LCD are connected to pins D3=10, D4=9, D5=8 and D6=7 of Arduino Mini.
8. +5V and GND pin of the PH sensor and TDS sensor are connected to +5V and GND pins of the power supply, respectively.
9. OUT pin of the MQ135 sensor is connected to pin A1 of Arduino Mini.
10. SCL and SDA pins of the BMP180 sensor are connected to pin A4 and A5 pins of Arduino Mini.
11. OUT pin of the dust sampler sensor is connected to pin A2of Arduino Mini.
12. Connect TX(1), RX(0), +VCC, and GND of Arduino Mini to TX, RX, +VCC and GND of NodeMCU.

Figure 14.2 shows the detailed circuit diagram of the system:

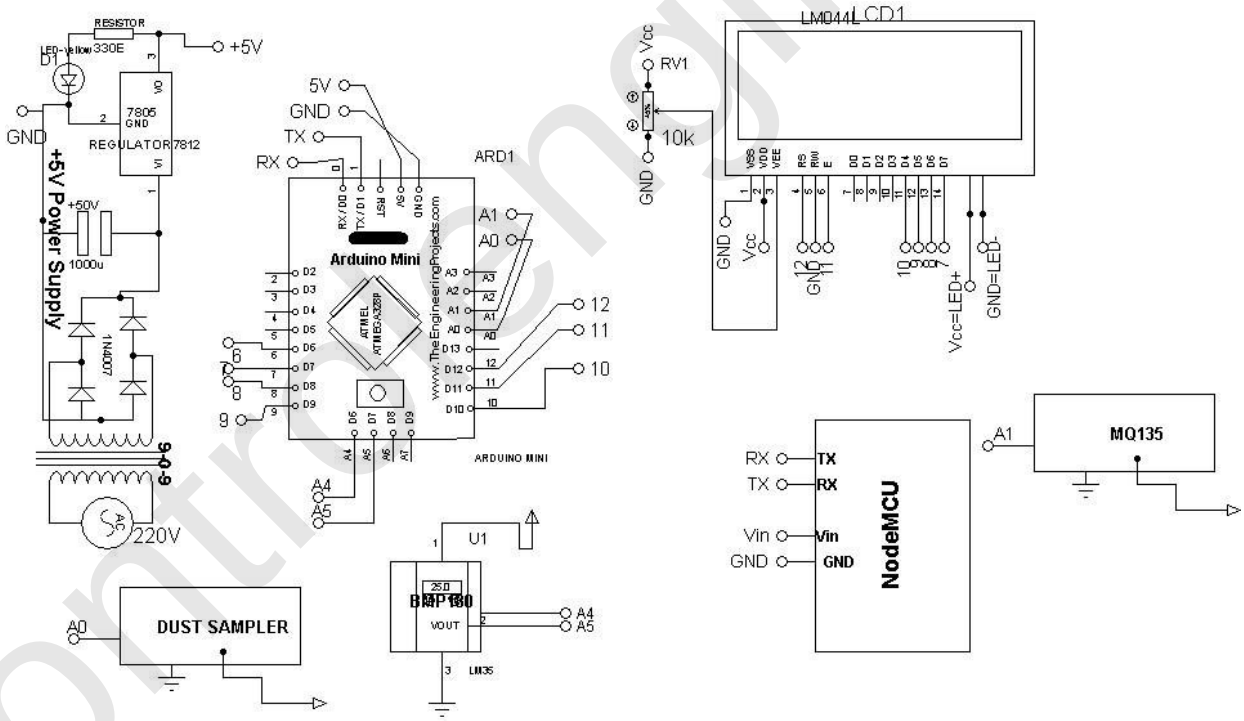

Figure 14.2: *Circuit diagram of the system*

Program

The program for Arduino Mini is as follows:

```
// library for BMP185
#include <Wire.h>
#include <Adafruit_BMP085.h>
Adafruit_BMP085 bmp;

// library for LCD
#include <LiquidCrystal.h>
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

// libray for Softserial
#include <SoftwareSerial.h>
SoftwareSerial mySerial(6,7);// 6 rx /7 tx

void setup()
{
 Serial.begin(9600); // initialize serial communication
 mySerial.begin(9600); // initialize soft serial communication
 lcd.begin(20, 4); // initialize LCD
 bmp.begin(); // initialize BMP sensor
}

void loop()
{
lcd.clear(); // clear the contents of LCD
int chk = DHT.read11(DHT11_PIN); // read DHT11
float PRESS=bmp.readPressure(); // record pressure
float ALT=bmp.readAltitude(); // record altitude
int DUST_SAMPLER=analogRead(A0); // read dust sampler
int MQ135=analogRead(A1);// read MQ135 sensor

// read and display BMP185 data
 lcd.setCursor(0,0); // set cursor on LCD
 lcd.print("P0:"); // print string on LCD
 lcd.print(PRESS); // print value on LCD
 lcd.print("Pa"); // print string on LCD

 lcd.setCursor(0,1); // Calculate altitude assuming 'standard'
 barometric & pressure of 1013.25 millibar = 101325 Pascal
 lcd.print("A0:"); // print string on LCD
 lcd.print(ALT); // print value on LCD
```

```
 lcd.print("m"); // print string on LCD

 lcd.setCursor(0,2); // set cursor on LCD
 lcd.print("DUST_sample:"); // print string on LCD
 lcd.print(DUST_SAMPLER); // print value on LCD
 lcd.setCursor(0,3); // set cursor on LCD
 lcd.print("MQ135:"); // print string on LCD
 lcd.print(MQ135); // print value on LCD
 lcd.setCursor(10,0); // set cursor on LCD
 // Calculate altitude assuming 'standard' barometric & pressure
 of 1013.25 millibar = 101325 Pascal
 lcd.print("UV_Indx:"); // print string on LCD
 lcd.print(UV); // print value on LCD
 Serial.print(PRESS); // print value on serial
 Serial.print(","); // print comma on serial
 Serial.print(ALT); // print value on serial
 Serial.print(",");// print comma on serial
 Serial.print(DUST_SAMPLER); // print value on serial
 Serial.print(",");// print comma on serial
 Serial.print(MQ135); // print value on serial
 Serial.print('\n'); // print comma on serial
 delay(30); // delay of 30 mSec

}
```

The program for NodeMCU is as follows:

```
#include <ESP8266WiFi.h>
#include "Virtuino_ESP_WifiServer.h"
const char* ssid = "ESPServer_RAJ";
const char* password = "RAJ@12345";
WiFiServer server(8000); // Server port
Virtuino_ESP_WifiServer virtuino(&server);
int storedValue=0;
int counter =0;
long storedTime=0;
String PRESS,ALT,DUST,MQ135;
String inputString_NODEMCU = "";
void setup()
```

```
{
 virtuino.DEBUG=true; // set this value TRUE to enable the
 serial monitor status
 virtuino.password="1234"; // Set a password to your web server
 for more protection
 Serial.begin(9600); // Enable this line only if DEBUG=true
 delay(10);
 pinMode(2, OUTPUT);
 digitalWrite(2, 0); // make pin to LOW
 Serial.println("Connecting to "+String(ssid));
 WiFi.mode(WIFI_STA); // Config module as station only.
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED)
 {
 delay(500); // delay 500 msec
 Serial.print("."); // print serial
 }
 Serial.println("");// print serial
 Serial.println("WiFi connected");// print string on serial
 Serial.println(WiFi.localIP()); // print IP on serial
server.begin(); // initialize server
Serial.println("Server started"); // print string on serial
}

void loop()
{
 virtuino.run(); // run virtuino APP
 serialEvent_NODEMCU();
 int v1=virtuino.vDigitalMemoryRead(0); // Read virtual memory 0
 from Virtuino app
 nt v2=virtuino.vDigitalMemoryRead(1); // Read virtual memory 0
 from Virtuino app
 if (v1!=storedValue)
 {
 Serial.println("-------Virtual pin DV0 is changed
 to="+String(v1));
 if (v1==1)
digitalWrite(D4,0); // make pin to LOW
```

```
 else
digitalWrite(D4,1); // make pin to HIGH
storedValue=v1;
 }
 if (v2!=storedValue)
 {
 Serial.println("-------Virtual pin DV0 is changed
 to="+String(v2));
 if (v2==1)
digitalWrite(D4,0); // make pin to LOW
 else
digitalWrite(D4,1); // make pin to HIGH
 storedValue=v2;
}
 serialEvent_NODEMCU(); // call function for sensor value
 virtuino.vMemoryWrite(2,PRESS); // write data on channel to of
 virtuino
 virtuino.vMemoryWrite(3,ALT); // write data on channel to of
 virtuino
 virtuino.vMemoryWrite(4,DUST); // write data on channel to of
 virtuino
 virtuino.vMemoryWrite(5,MQ135); // write data on channel to of
 virtuino
 long t= millis();
 if (t>storedTime+5000)
 {
 counter++;
 if (counter>20) counter=0; // limit = 20
  storedTime = t;
  virtuino.vMemoryWrite(12,counter); // write counter to
  virtual pin V12
 }
}
void serialEvent_NODEMCU()
{
 while (Serial.available()>0)
 {
```

```
inputString_NODEMCU = Serial.readStringUntil('\n');// Get serial
input
StringSplitter *splitter = new
StringSplitter(inputString_NODEMCU, ',',5); // new
StringSplitter(string_to_split, delimiter, limit)
int itemCount = splitter->getItemCount();
for(int i = 0; i < itemCount; i++)
{
String item = splitter->getItemAtIndex(i);
PRESS= splitter->getItemAtIndex(0); // extract the value of
sensor from main string
ALT= splitter->getItemAtIndex(1); // extract the value of sensor
from main string
DUST= splitter->getItemAtIndex(2); // extract the value of
sensor from main string
MQ135= splitter->getItemAtIndex(3); // extract the value of
sensor from main string
}
inputString_NODEMCU = ""; // make string empty
delay(200); // delay of 200 msec
}

}
```

Virtuino app

Figure 14.3 shows the Virtuino app for the system. The Virtuino application is a human machine interface platform on cloud. The Virtuino application can be controlled through Bluetooth, WiFi, GPRS, and ThingSpeak. It is a freely published application given by Ilias Lamprou.

The following are the steps to develop the app:

1. Make the connection with Arduino uno, NodeMCU and external devices.
2. Follow the following step:
   1) Click to download the Virtuino Library ver 1.1.
   2) Run to add the library via the Arduino IDE.

3) Burn the hex code in NodeMCU.

3. Add WiFi settings with the Android device.
4. Make the application in the Virtuino app and run it to interact with ESP8266/NodeMCU:

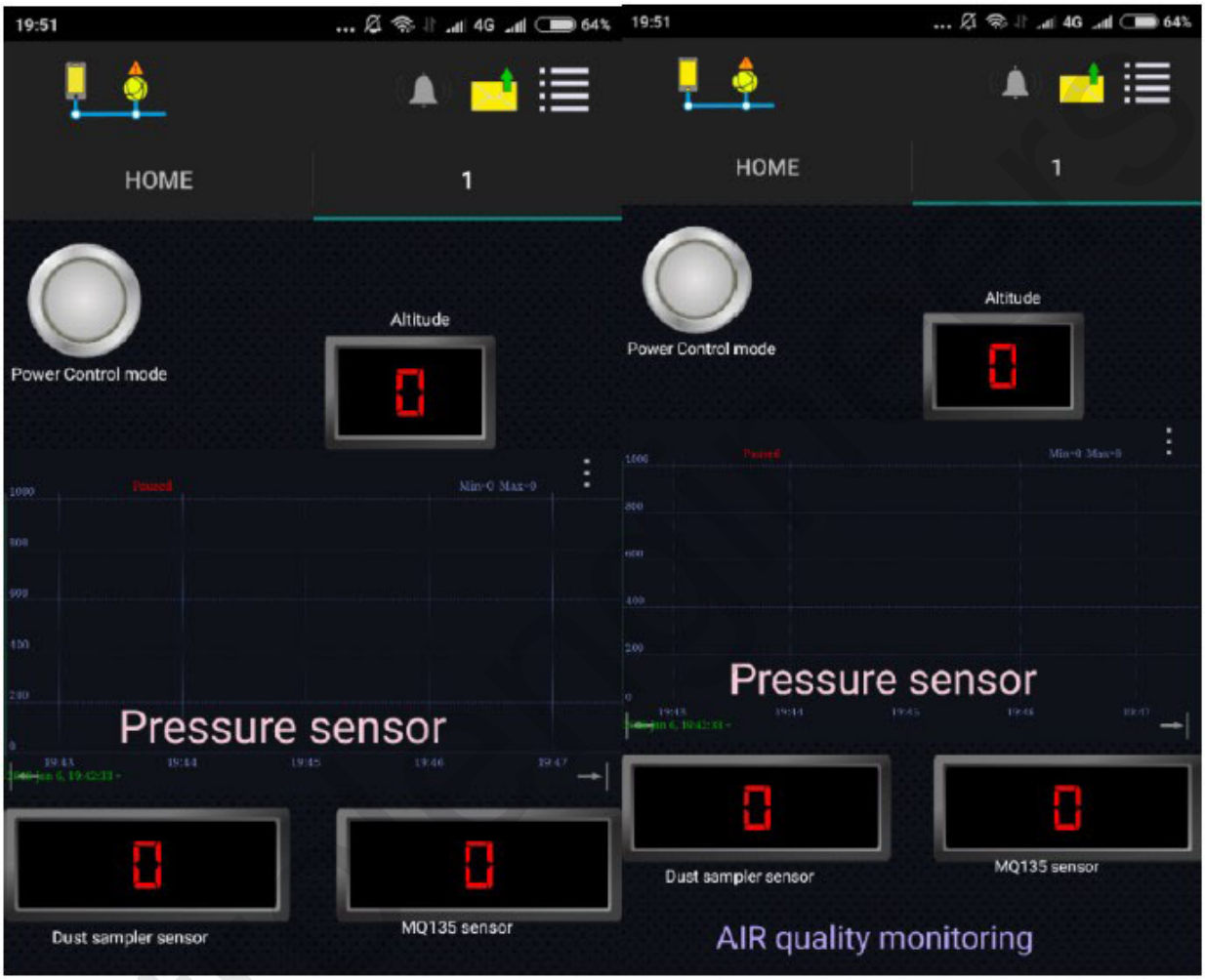

Figure 14.3: *Virtuino app*

Conclusion

This chapter concludes the air quality monitoring system with the help of sensors. A complete system design and development is discussed along with the circuit connections and programs. The Virtuino app is also discussed to display the sensory data.