HOME AUTOMATION USING RASPBERRY PI

MINI PROJECT REPORT

Submitted by

Deepak J Puthukkaden (ECU 122 20)

Giridhar A K (ECU 122 31)

Govindh B (ECU 122 32)

Rohit Sreekumar (ECU 122 53)

Shine Ali (ECU 122 59)

in partial fulfillment for the award of the degree

of

Bachelor of Technology

in

ELECTRONICS AND COMMUNICATION ENGINEERING

of

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS ENGINEERING

MODEL ENGINEERING COLLEGE

COCHIN 682 021

MODEL ENGINEERING COLLEGE

THRIKKAKARA, KOCHI-21

DEPARTMENT OF ELECTRONICS ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that the mini project report entitled

HOME AUTOMATION USING RASPBERRY PI

Submitted by

. . . . . .

. . . . . .

is a bonafide account of the work done by him/her under our supervision

Mrs. Laila D	***Mrs. Shiji T P***	***Mr.Rashid M.E***
Head of Department	***Mini Project Coordinator***	***Project Guide***

ABSTRACT

Home automation is one of the most researched topics in the present technological scenario. It is basically an application of Internet of Things or IoT. Internet of things is an interconnection of all the appliances and devices that we use in our everyday life; it may be coffee vending machines, smart phones, medical monitors to huge weather stations. In the approach of internet of things, all devices work in a synchronized manner, i.e. they are interdependent.

This system consists of a main processing unit (Raspberry Pi), microcontroller to interface the different modules, and the independent modules. In our project, the area which we focus on is home automation that works based on real time data. It does the following functions:

- *Adaptive internal lighting and automatic external lighting system*
- *Automatic water pumping system*
- *Password enabled door lock using keypad*
- *Interactive Music player*
- *Logging of all the control events*

ACKNOWLEDGEMENT

At this moment of accomplishment, we are presenting our work with great pride and pleasure, we would like to express our sincere gratitude to all those who helped us in the successful completion of our venture. First of all, we would like to thank our Principal **Prof. V.P Devassia** who provided us with all facilities and amenities for the development of our project. We would like to thank our HOD, **Mrs.Laila D** for helping us in the successful accomplishment of our project. We are exceedingly grateful to our project coordinator Assistant Professor, **Mrs.Shiji T.P** for her timely and valuable suggestions. We would also like to thank our project guide Senior Lecturer **Mr.Rashid M.E.** who gave us constant guidance and support throughout this journey of turbulence.We also sincerely thank **Mr.Sujith**, Lab Technician Department of Electronics and Communication for his constant support and encouragement for our project. We would also like to thank our parents and friends for their over-whelming and whole hearted encouragement and support without which this would not have been successful. Above all we thank God almighty for constantly motivating us with His love, and giving courage to move forward with confidence and self-belief.

Table of Contents

List of Figures

List of Abbreviations

RISC	Reduced Instruction Set Computing
RPi	Raspberry Pi
IC	Integrated Circuit
LED	Light Emitting Diode
RTC	Real Time Clock
PIR	Passive Infra Red
LDR	Light Dependent Resistor
SoC	System On a Chip
GPU	Graphics Processing Unit
UART	Universal Asynchronous Receiver/Transmitter
I2C	Inter-Integrated Circuit
GND	Ground
LCD	Liquid Crystal Display
PCB	Printed Circuit Board

Chapter 1

PROJECT OVERVIEW

1. 1 Introduction

The proposed idea for our Mini project for the Fifth Semester (2014-2015) is Home Automation based on Raspberry Pi. Automation is a broad concept in engineering and home automation is but a small subsidiary of this vast idea. Recently, the Internet Of Things has gathered momentum and many interesting applications have surfaced making many of the daily chores simple and efficient. We intend to implement a small scale home automation project using the Raspberry Pi. Raspberry Pi is a powerful credit card sized computer capable of running as a normal computer and also capable of running the functions of a microcontroller.

Due to the hardware limitations (limited no of input and output pins) of the Raspberry Pi, we are forced to interface additional microcontrollers to control the various sensors involved in the automation.

1. 2 Objectives

The system comprises automation of some daily tasks in our home. The different functions of the system are:

- Adaptive Internal Lighting and automatic external lighting System
- Automatic Water Pumping System
- Password Enabled Door Lock using keypad
- Interactive Music Player
- Data logging and temperature measurement

1.3 System Specifications

- Raspberry Pi
- Atmega 328P (2)
- Serial Interface
- Power supply 5V, 2A
- Passive InfraRed (PIR)
- Real Time Clock (RTC)

Chapter 2

SYSTEM OVERVIEW

2.1 Modules

We use 4 individual modules in our project namely Raspberry Pi, Microcontroller (Atmega 328P), PIR sensor and an RTC (DS1307).

2.1.1 Raspberry Pi

Raspberry Pi is a single board computer. It has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor, VideoCore IV GPU with 256MB of RAM. But it does not come with a Real-Time Clock, though it can be interfaced with DS1307 through I2C.

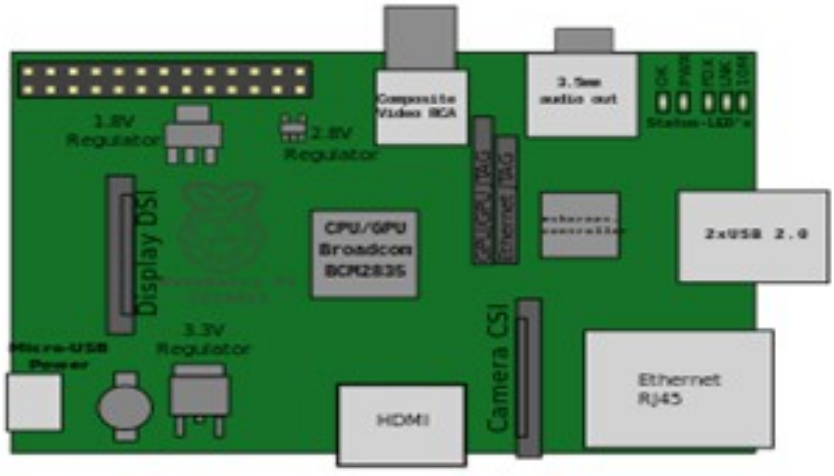

Figure 2.1:-Raspberry Pi[3]

2.1.2 Microcontroller Atmega328P

The high-performance Atmel picoPower 8-bit AVR RISC-based microcontroller combines 32KB ISP flash memory with read-while-write capabilities, 1024B EEPROM, 2KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, a 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts.

By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed.

Figure 2.2:- Atmega328P[2]

2.1.3 Passive Infrared (PIR) sensor

A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. An individual PIR sensor detects changes in the amount of infrared radiation impinging upon it, which varies depending on the temperature and surface characteristics of the objects in front of the sensor. When an object, such as a human, passes in front of the background, such as a wall, the temperature at that point in the sensor's field of view will rise from room temperature to body temperature, and then back again. The sensor converts the resulting change in the incoming infrared radiation into a change in the output voltage, and this triggers the detection. Moving objects of similar temperature to the background but different surface characteristics may also have a different infrared emission pattern, and thus sometimes trigger the detector.

Figure 2.3:- Passive Infra Red (PIR) Sensor[7]

2.1.4 LM35 Temperature Sensor

LM35 is a precision IC temperature sensor with its output proportional to the temperature (in ºC). The sensor circuitry is sealed and therefore it is not subjected to oxidation and other processes. With LM35, temperature can be measured more accurately than with a thermistor. It also possess low self heating and does not cause more than 0.1 ºC temperature rise in still air.

The operating temperature range is from -55°C to 150°C. The output voltage varies by 10mV in response to every ºC rise/fall in ambient temperature, *i.e.,* its scale factor is 0.01V/ ºC.

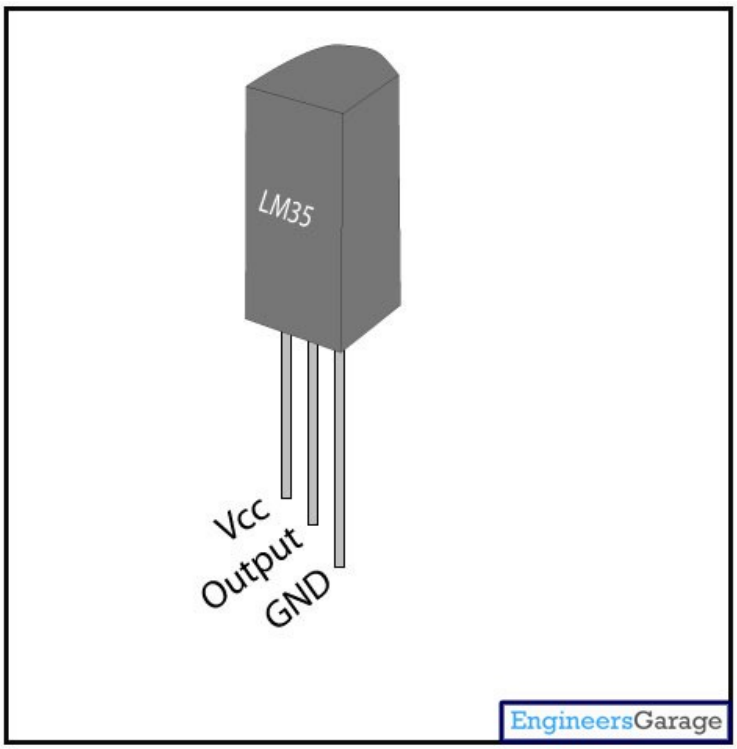

Figure 2.4:- LM35 Temperature Sensor[6]

2.1.4 Real Time Clock (RTC)

A real-time clock (RTC) is a computer clock that keeps track of the current time.

Main advantages are:-

1. Low Power Consumption

2. Frees the main system from critical time related tasks

3. More accurate than other methods

Figure 2.5:- Real Time Clock (DS1307)[8]

2.2 Systems

There are 6 systems or individual modules in our project. They are:

2.2.1 Adaptive Internal Lighting System

The Adaptive Internal Lighting System controls the amount of illumination in a room by controlling a light source, based on the amount of light present in the room. The sensor involved is a light dependant resistor (LDR) which varies it's resistance depending on the amount of light falling on it. The change in resistance will result in a change in the voltage drop across the resistance (a varying analog signal). For demonstration purposes we intend to use an LED, the brightness of which is controlled by Pulse Width Modulation.

2.2.2 Automatic External Lighting System

This feature of our project controls an External Lighting System based on the amount of external light. When the light outside falls below a specific value , the light is turned on. It also makes use of a LDR.

2.2.3 Automatic Water Pumping System

Through this system we intend to control the water pump based on the water level inside the water tank. This prevents the overflowing and wastage of water which is a pristine and scarce resource. The water level is determined by two sensors and the output is used by Raspberry Pi to drive a relay that switches the water pumping system.

2.2.4 Password Enabled Door Lock System

This feature adds to the security of the smart home. A 4*3 keypad is interfaced with the microcontroller which is then connected to a door lock system. The door lock gets unlocked when the correct password is entered.

2.2.5 Interactive Music System

This part of the project adds to the entertainment in the smart home. A simple music player is integrated with the Raspberry Pi which can be controlled using small buttons for functions like play, pause etc.

2.2.6 Data logging and Temperature Measurement

This feature comprises of the data logging and event tracking feature of our project. Data logging includes the temperature measurement and event logging. It can be used to analyze the working of the home. Temperature measurement is done by lm35 and logged in Raspberry Pi.

2.3 Block Diagram

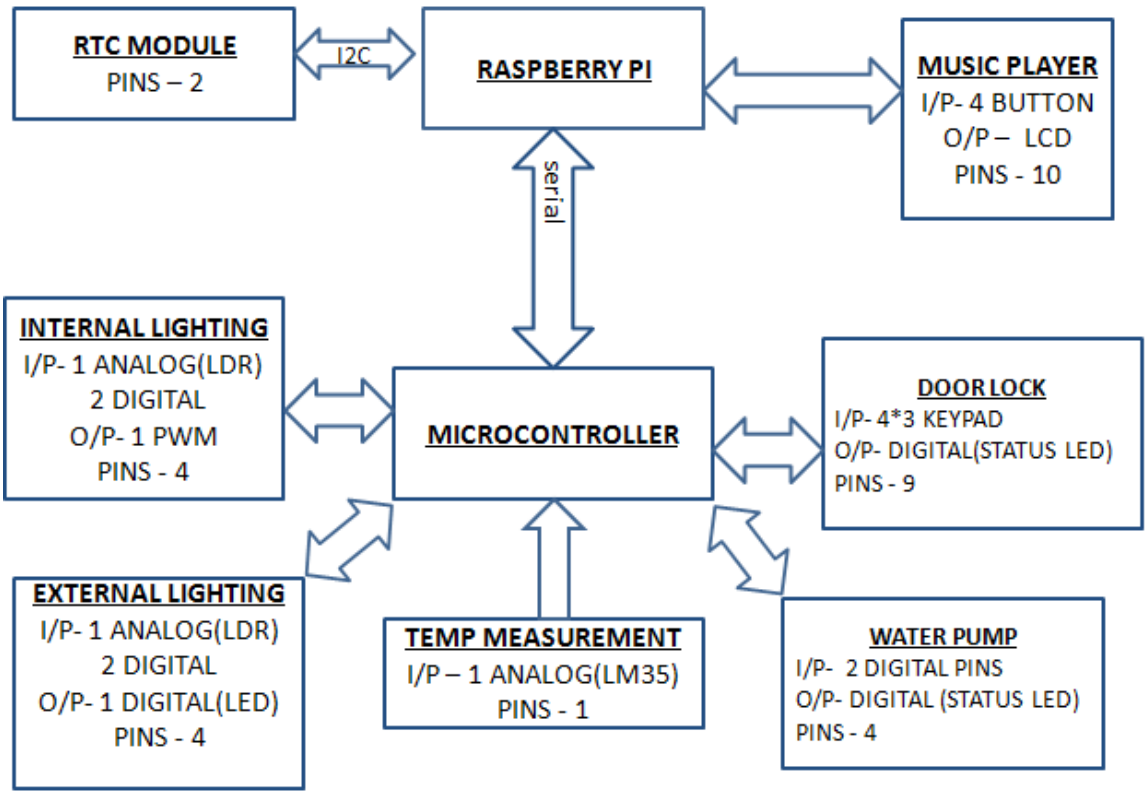

Figure 2.6:- Block Diagram

2.4 Serial Communication

The universal asynchronous receiver/transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires. Communication may be simplex (in one direction only, with no provision for the receiving device to send information back to the transmitting device), full duplex (both devices send and receive at the same time) or half duplex (devices take turns transmitting and receiving).

There are two modes of Serial communication - the synchronous and asynchronous mode. In synchronous mode, both the transmitter and the receiver work in synchronization with each other. In Asynchronous mode, the transmitter and the receiver are not synchronized. The speed of transmission of data is given by the baud rate. The baud rate represents the number of bits sent per second. The standard baud rates used in serial communication are 4800, 9600, 115200 etc.

In synchronous mode, the clock signal is recovered from the data stream and both the transmitter and the receiver work in synchronization with each other. In the case of asynchronous transmission, there is a start and stop bit that denotes the starting and stopping of the communication. The start and stop bits are sent in addition to the 8 data bits.

The mode that we are using is UART (asynchronous mode) with a baud rate of 9600.

Chapter 3

SYSTEM DESIGN

3.1 Adaptive Lighting System

It takes in the analog input from the LDR and the corresponding voltage is transmitted into the microcontroller.

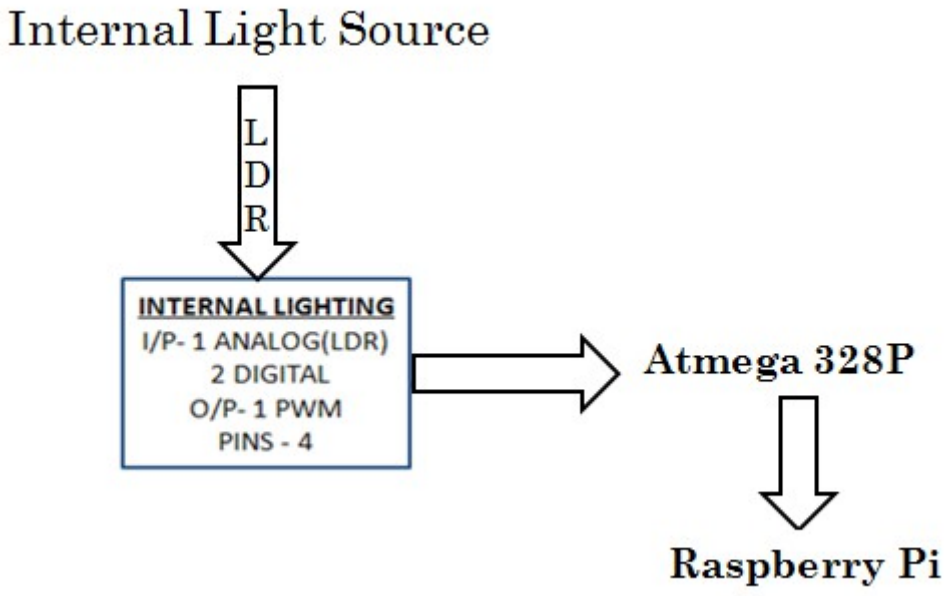

Figure 3.1:- Flowchart of Adaptive Internal Lighting System

Hardware Requirements:-

- LDR
- LED
- Power Supply- 3.3V

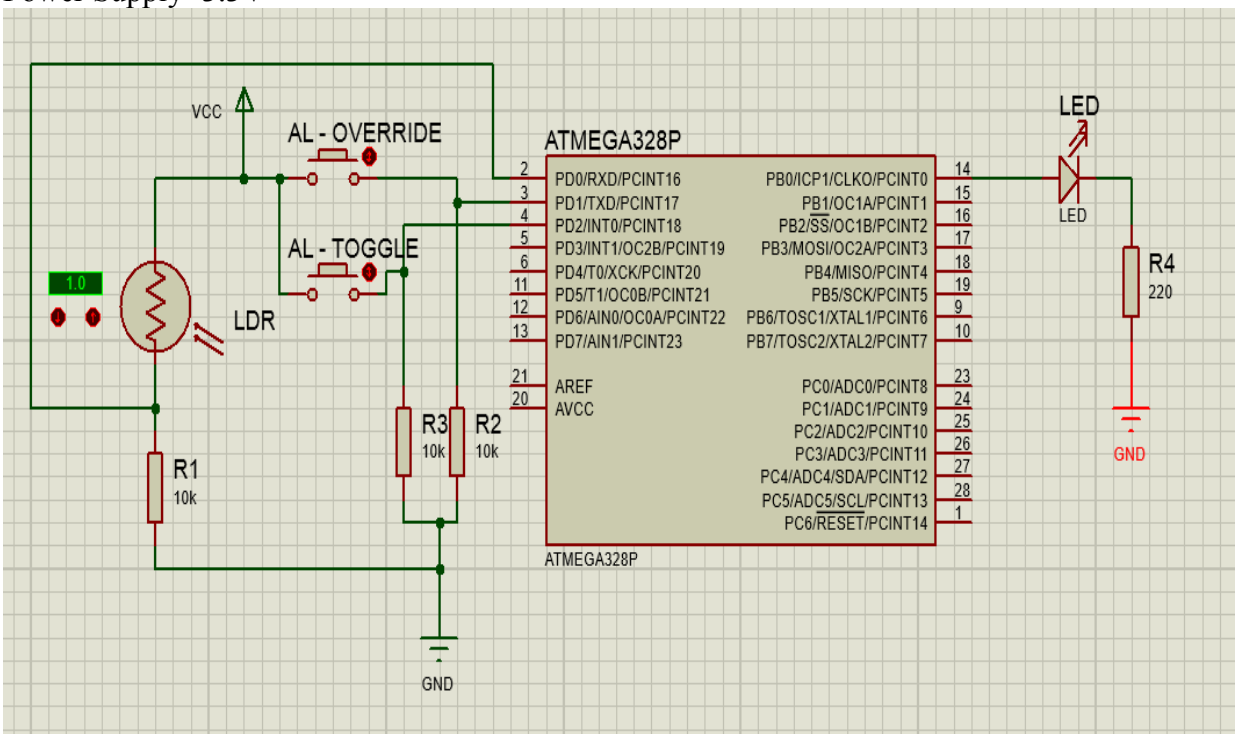

Figure 3.2:- Adaptive Lighting System Design

3.2 Automatic External Lighting System

It takes in the analog input from the LDR and the corresponding voltage is transmitted into the microcontroller.

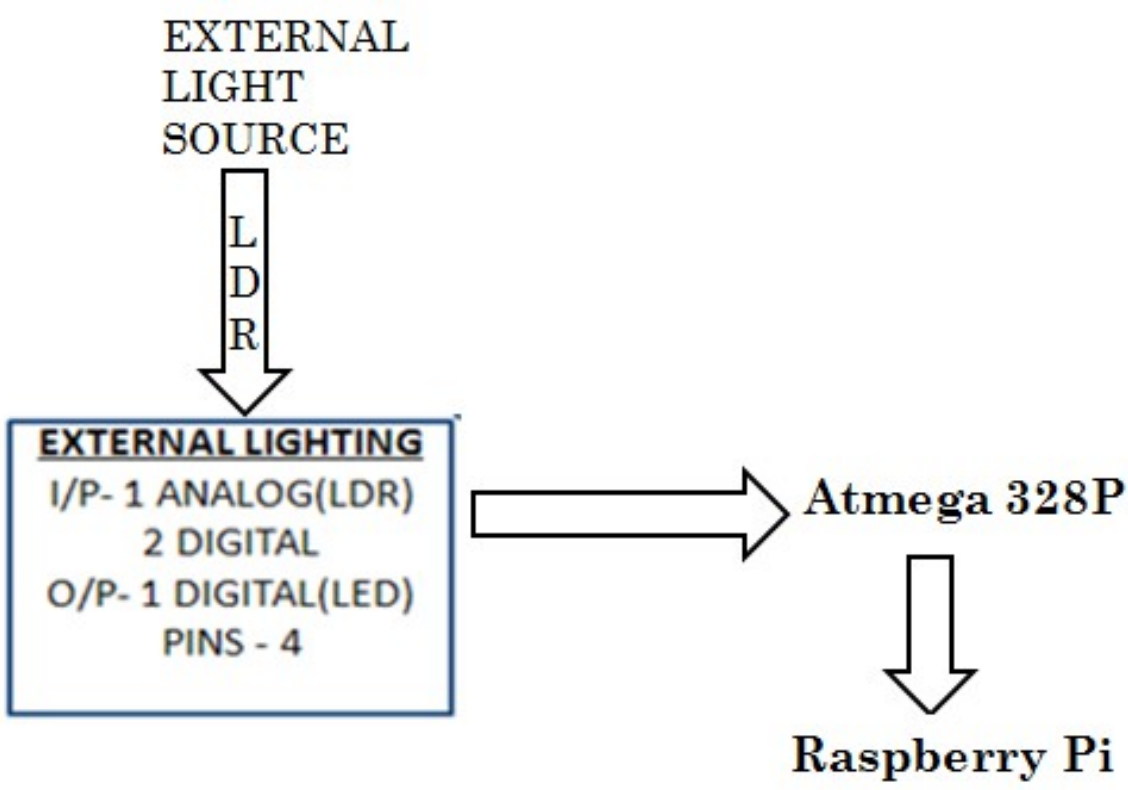

Figure 3.3:- Flowchart of Automatic External Lighting System

Hardware Requirements:-

- LDR
- LED
- Power Supply- 3.3V

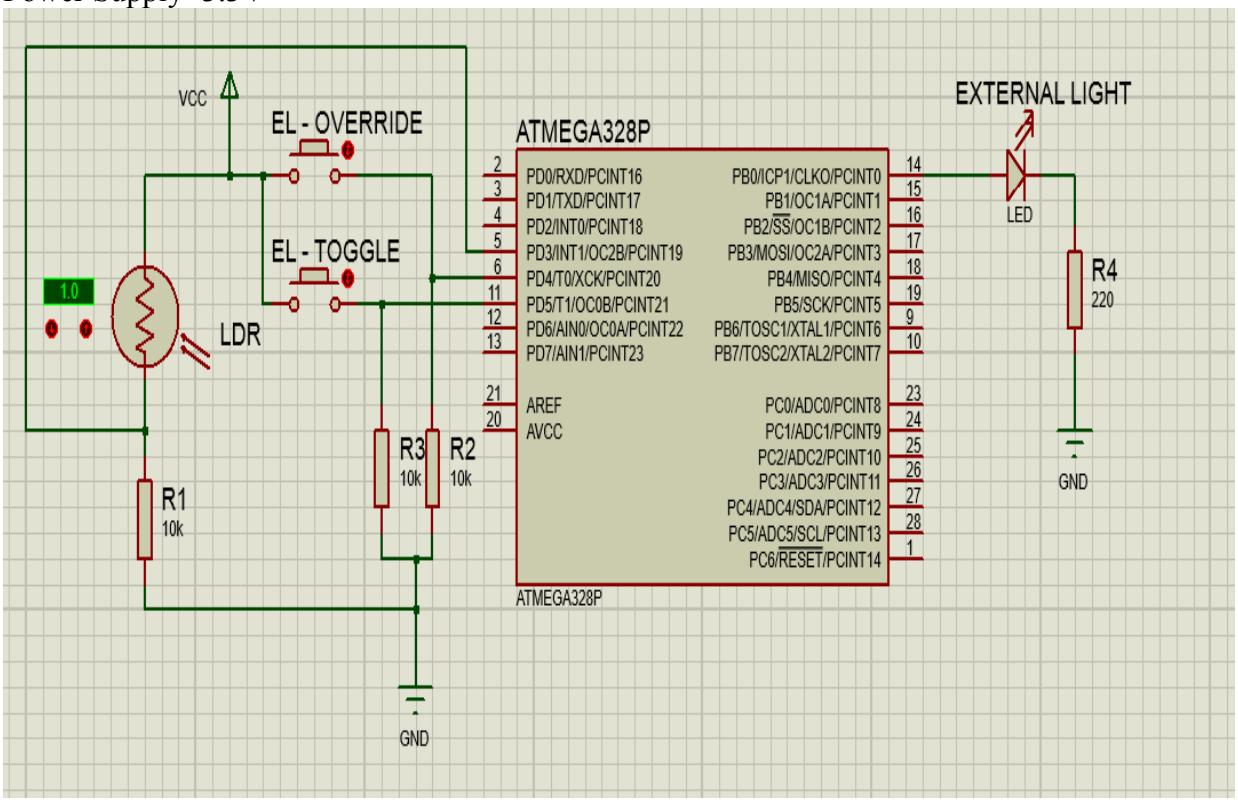

Figure 3.4:- Automatic Lighting System Design

3.3 Automatic Water Pump System

It uses 3 sensors namely for the bottom tank, low level of top tank and high level of top tank. The water is pumped from the bottom tank to the top tank. The sensors send the output of the water level from the top tank to the microcontroller and the microcontroller controls the motor accordingly.

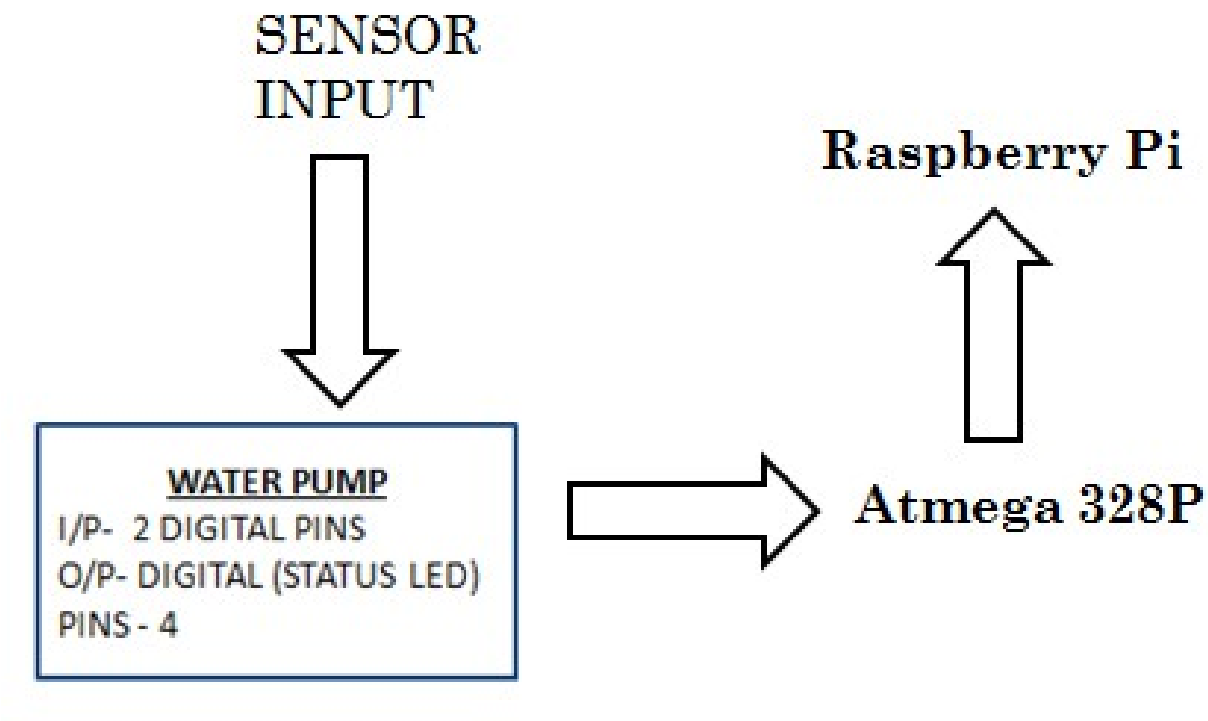

Figure 3.5:- Flowchart of Automatic Water Pump System

Hardware Requirements:-

- Relay
- L293D
- Power Supply-5V

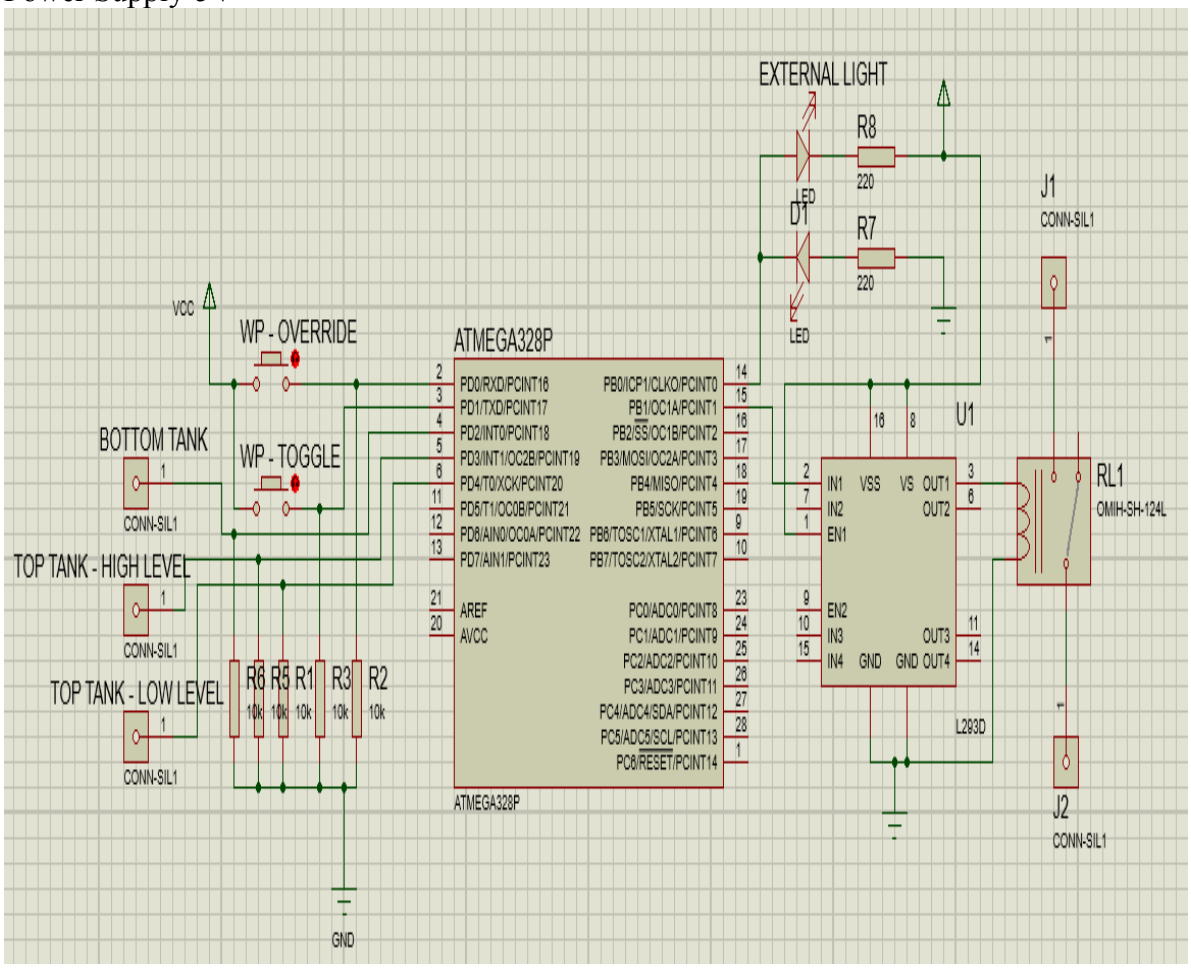

Figure 3.6:- Automatic Water Pump System Design

3.4 Password Enabled Door Lock

The system takes in the input password from the keypad, the microcontroller checks it with the preset password and controls the motor accordingly.

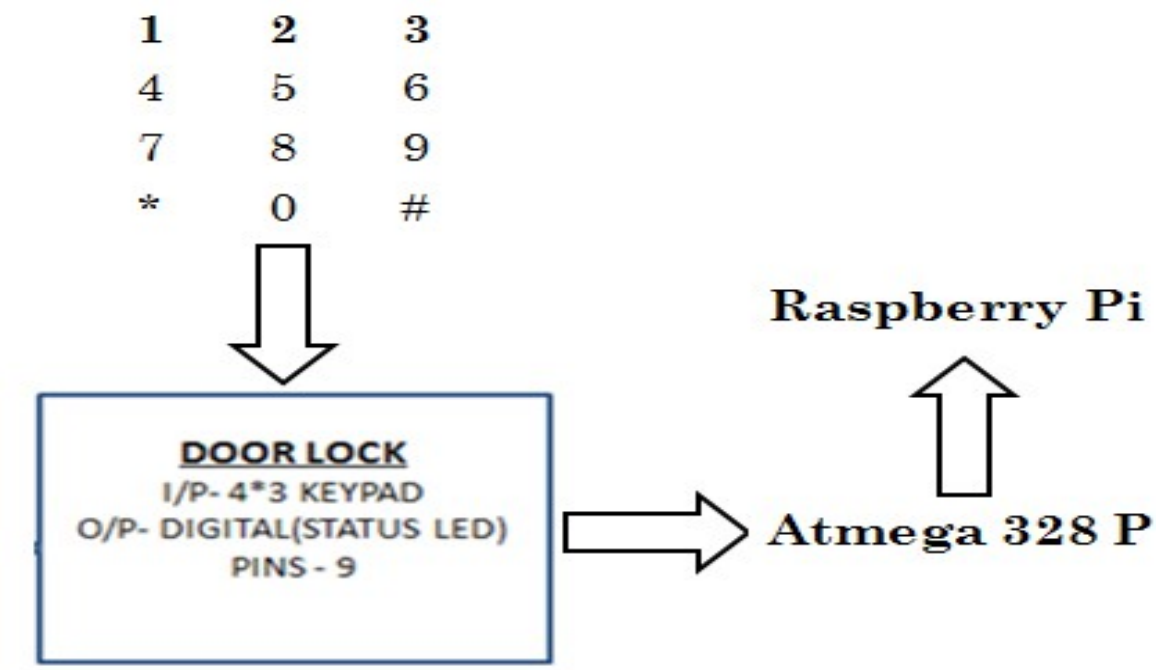

Figure 3.7:- Flowchart of Password Enabled Door-lock System

Hardware Requirements:-

- Relay
- Keypad
- L293D
- Power Supply-5V

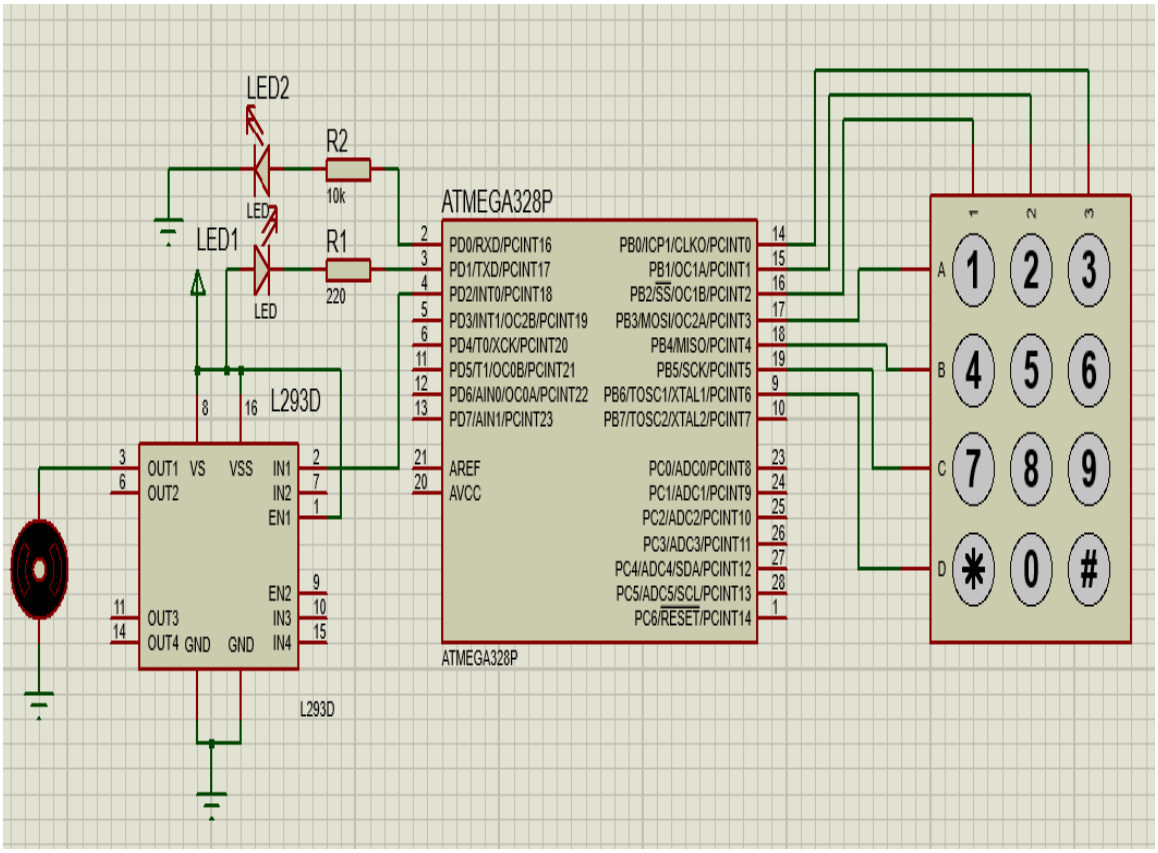

Figure 3.8:- Password Enabled Door Lock Design

3.5 Interactive Music Player System

There are 4 buttons namely PLAY,PAUSE,NEXT SONG,PREVIOUS SONG which are connected to the raspberry pi. Also an LCD is interfaced with the raspberry pi which displays the current title of the song being played.

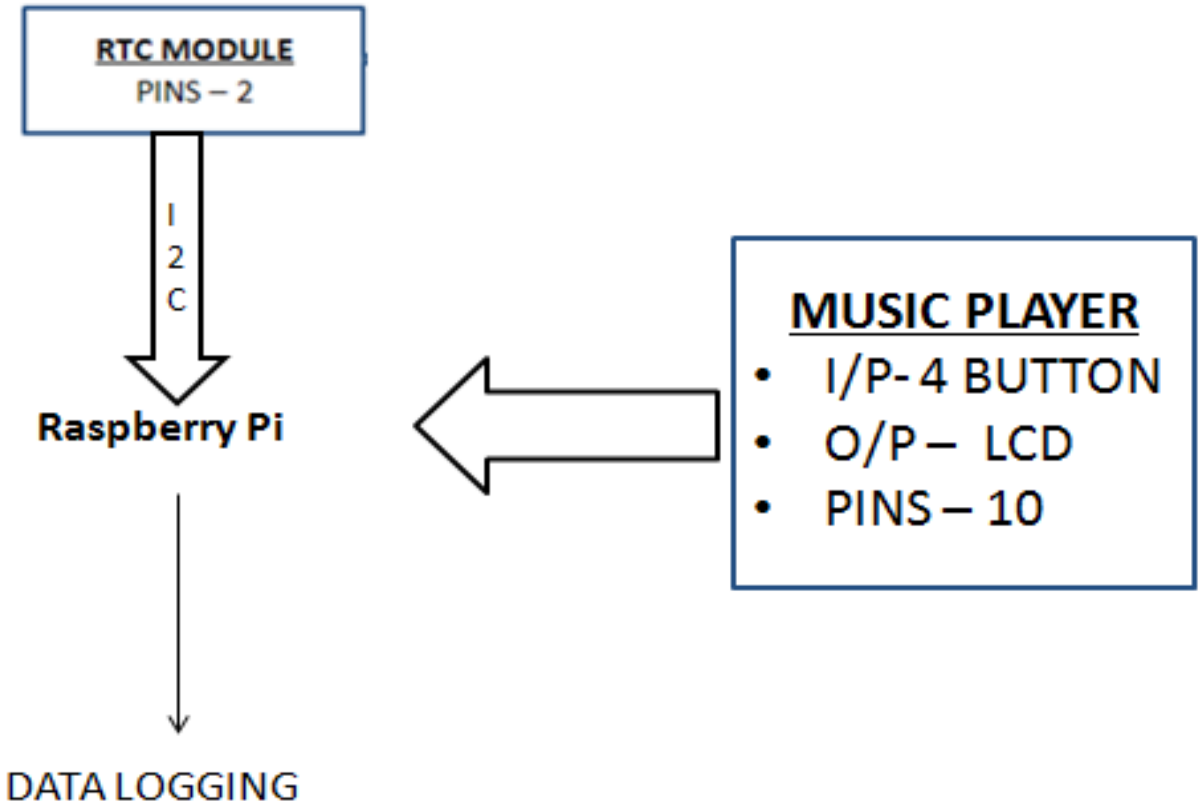

Figure 3.9:- Flowchart of Interactive Music Player System

Hardware Requirements:-

- Push Buttons
- Speaker (headphone)
- LCD
- Power Supply-5V

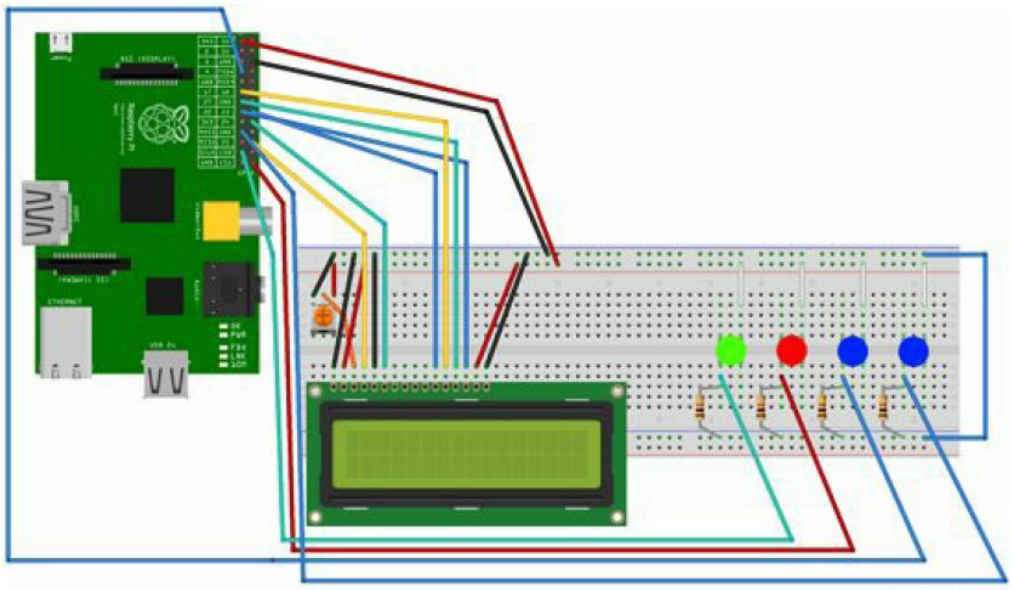

Figure 3.10:- Interactive Music Player System Design

3.6 Power Supply

The power supply generates 5V output which is required to drive the L293D and also the DS1307.

Specifications:

- 5V, 2A

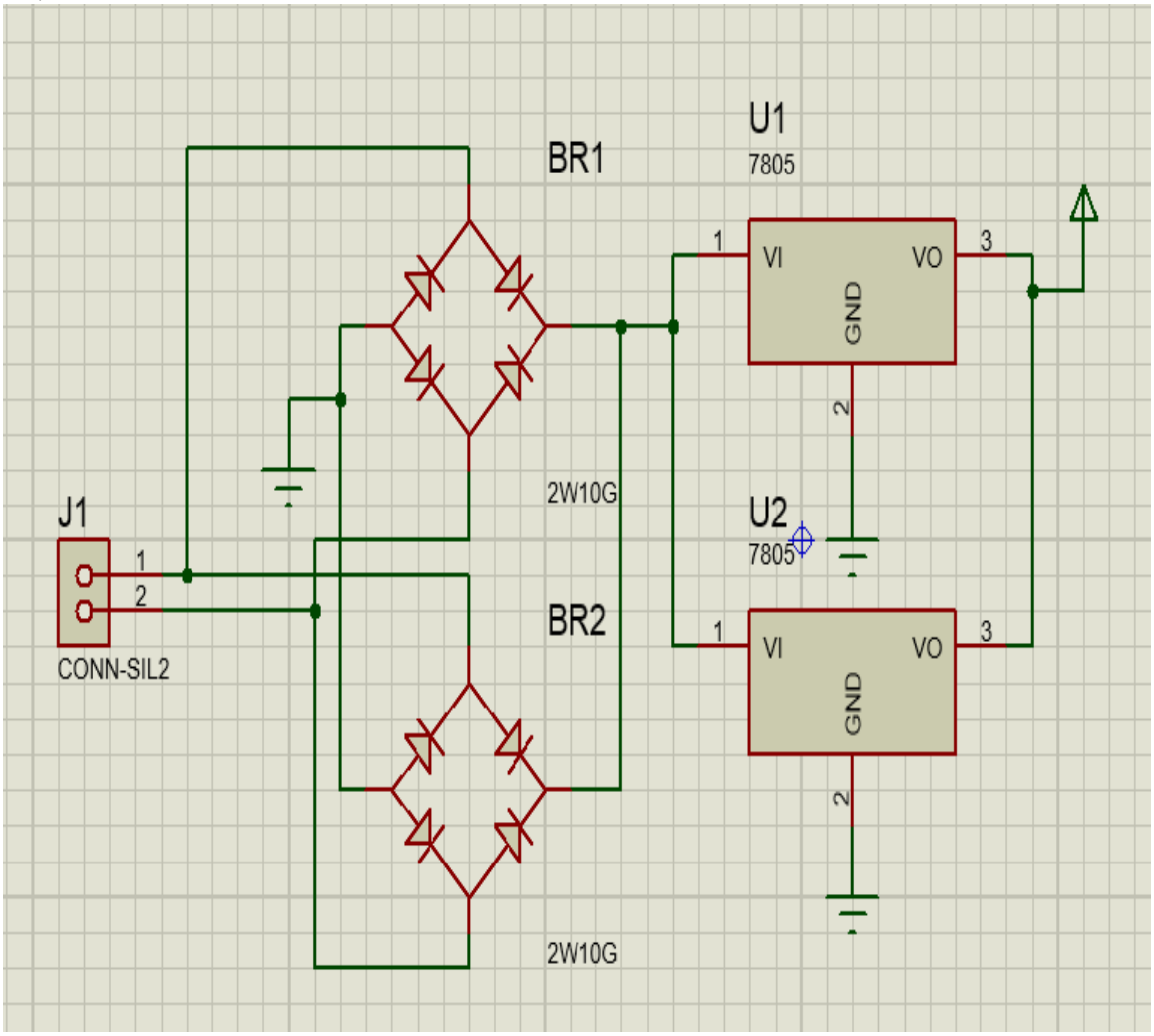

Figure 3.11:- Power Supply Design

3.7 Real Time Clock (RTC)

The DS1307 provides the real time for the system. It aims at low power consumption.

It oscillates with a crystal frequency of 32.768kHz.

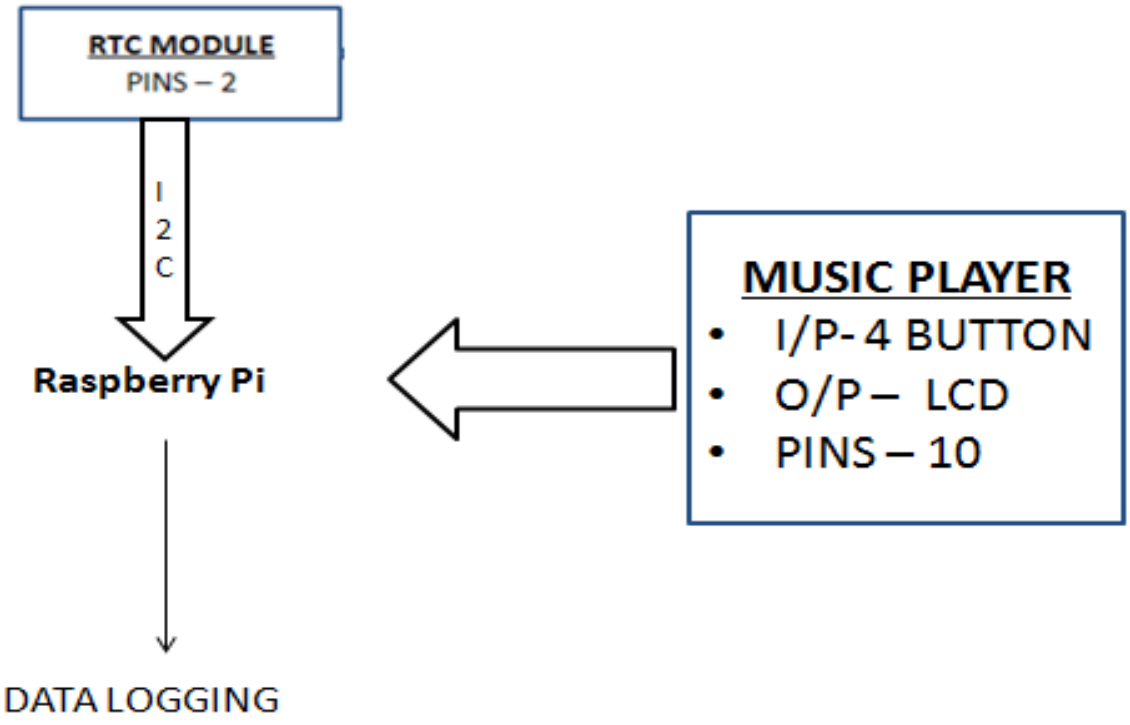

Figure 3.12:- Flowchart of RTC System

Hardware Requirements:-

- DS1307 RTC chip - A low power, full binary coded decimal serial real time clock.

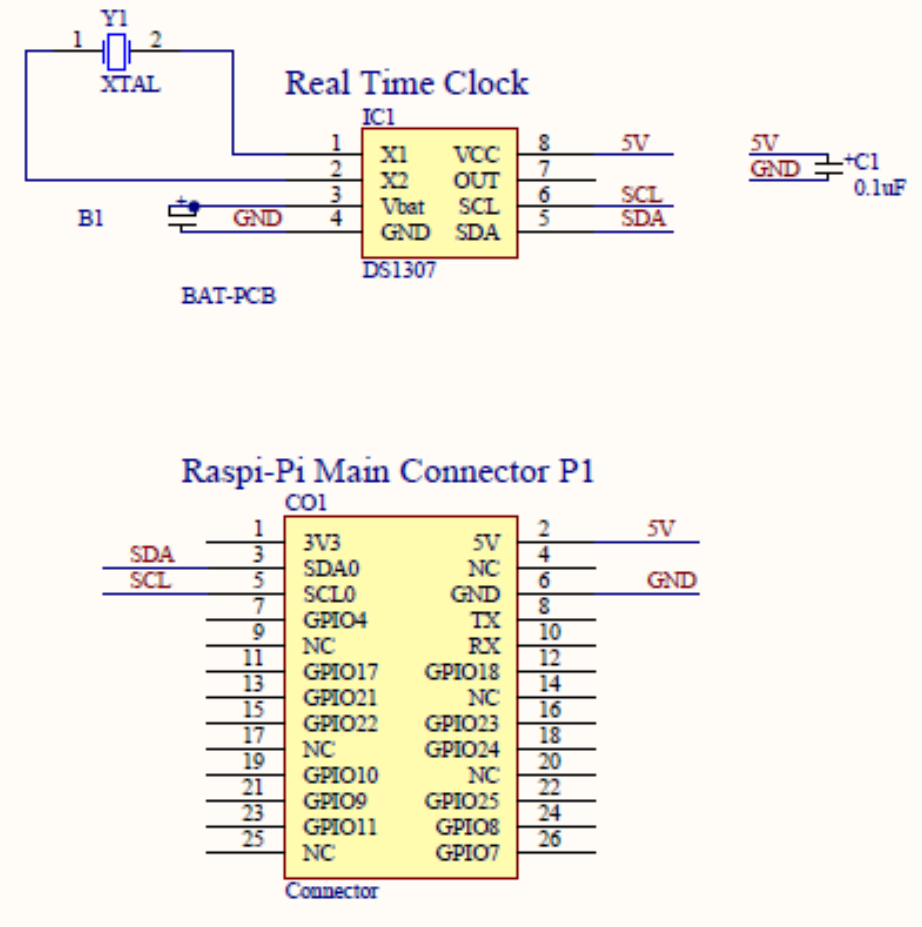

Figure 3.13:- Real Time Clock Module Design

Chapter 4

PCB FABRICATION AND SOLDERING

4.1 PCB Fabrication

The materials required for PCB fabrication are copper clad sheet, little paint drilling machine and ferric chloride solution.

The PCB fabrication involves the following steps:

4.1.1 Preparation of the PCB layout

The layout is commonly prepared in the scale of *2:1*. It offers a reasonable compromised below accuracy gain and handling convenience as *2:1* part of art work has the actual PCB area. Grid systems are commonly used for preparation of the layout. The use of the grid sheet gives more convenience in placement of components and conductors. The grid system based on 0.1 is found to be too coursing, a grid equidistant of *0.1mm* is recommended.

Procedure:

• Each and every PCB is viewed from component side.

• The designing of the layout is started with an absolutely clear component list and circuit diagram.

• The large components are placed first and the space in between is filled with smaller area.

• In the designing of a PCB layout, it is very important to divide circuit into sub units. Each of these sub units are realized in the defined portion of the board.

• The components are placed in the print sheet tanning and standard length and width.

• The punched component layout is circled to taking the standard size of the land pads.

4.1.2 Pattern transfer

After the film is processed the film master are obtained. The transfer of the conductor which is on the film master on to the copper clad base material is done by two methods mainly photo printing and screen printing. Photo printing is extremely accurate which is also applied to the fabrication of semiconductors. Screen printing is comparatively cheap and simple method for pattern transfer although less precise than photo printing. But this is less costly, so this method is commonly used.

4.1.3 Screen printing

In screen printing, the processing is very simple. A screen fabric with uniform meshes and opening stretched and fixed on a solid frame of metal or wood. The circuit pattern is photographically transferred on to queue through the opening master onto the surface of the material to be printed. The light sensitive material is coated on to the screen and using film master the pattern is transferred to the screen. Then using ink the pattern is transferred to the copper clad sheet. Two methods are used for screened printing pattern into screen.

1. Direct method
2. Indirect method

In direct method the photographically sensitive film is transferred to screen. The film is exposed and then pasted to the screen. The pattern is then transferred to screen using the links and squeegee.

4.1.4. Etching

The removal of unwanted copper from copper clad sheet is known as etching.

For these 4 types of tanks are used.

1. Ferric Chloride
2. Cupric Chloride
3. Chromic acid
4 .Alkaline ammonia

Among these Ferric Chloride is cheap and also suited for home and industrial applications. The high corrosive power of FeCl leads to short etching time and little under etching. Ferric chloride matches well photo and screen printed resists.

4.1.5 Drilling

Drilling of components, mounting holes into the PCBs is by the most important mechanical machining operation in PCB production process. The importance of hole drilling on PCB has further group with electronic component miniaturization and its need for smaller hole diameters and higher packing density where hole punching is practically routed out.

Four types of drilling are commonly used.

1. Drilling by direct sight
2. Drilling by optical sight
3. Jig drilling
4. NC drilling

4.1.6 Component mounting

Component is basically mounted on one side of the board. On polarized two lead components are mounted to give the marking or the orientation throughout the board. The component orientation can be both horizontal as well as vertical but uniformly, direction is placed. The uniformity in orientation of polarized component is determined during design of PCB.

4.2 Soldering and Desoldering

4.2.1 Soldering

Soldering is a process of joining two or more dissimilar metals by melting another metal having low melting points.

4.2.2 Soldering Flux

In order to make the surface accept the solder readily, the component terminals should be free from oxides and other obstructing films. Soldering flux cleans the oxides from the surface of metal. The leads should be cleaned chemically or by scrapping using blade. Small amount of lead should be coated on the portion of leads and the bits of soldering. This process is called tinning. Zinc Chloride, Aluminum Chloride and rosin are the most commonly used fluxes.

4.2.3 Solder

It is an alloy of tin and lead, typically 60% tin and 40% lead. It melts at a temperature about 473K. Coating a surface with solder is called tinning. Solder used for electronic purposes contain tiny cores of flux, like the wires inside the main flux. The flux is corrosive like an acid and it clears the metal surface as the solder melts. This is why we melt solder on the joints, not on the iron tip.

4.2.4 Soldering Tips

It is the tool used to melt and apply solder at the joints in the circuit. It operates in 230V main supply. The normal power ratings of the soldering iron are 10W, 25W, 35W, 65W, 125W.

4.2.5 Preparing the Soldering Iron

- Place the soldering iron in its stand and plug it in. The iron will take a few minutes to reach its operating temperature of about 673K.
- Dampen the sponge in the stand.

- Wait for a few minutes for the soldering iron to warm up.
- Wipe the top of the iron on damped sponge.
- Melt little solder on the tip of the iron.

4.3 PCB Design

We have incorporated our entire components into a single PCB, in order to sell it as a product in the market

Figure 4.1:- PCB Layout

Chapter 5

SOFTWARE

5.1 Microcontroller 1- Program

Microcontroller 1 controls the internal adaptive lighting system, automatic external lighting system and temperature measurement system. The analog input is taken from the LDR of the lighting system and converted to a DC value which is compared to a preset threshold values and the light intensity is controlled using it. The temperature is measured using LM35 sensor.

5.2 Microcontroller 2- Program

Microcontroller 2 controls the Water Pump and Door Lock System. There are sensors located in the water tank at the top and bottom, which controls the motor pump. The keypad used is 3*4 which has 7 output lines connected to the microcontroller. A password is stored in the EPROM of the microcontroller, the user entered password is checked with the pre stored password. “ * ” when pressed within the allocated time is used to change the preset password in the microcontroller.

5.3 Raspberry Pi Programs

There are 2 programs stored in the Raspberry Pi. They are:-

5.3.1 Music Player Program

There are 4 buttons namely PLAY,PAUSE,NEXT SONG,PREVIOUS SONG which are connected to the raspberry pi. Also an LCD is interfaced with the raspberry pi which displays the current title of the song being played.

5.3.2 Data Logging Program

Initially a protocol is established for each systems and the serial communication and data logging is programmed using this protocol. A 2 byte data is serially transmitted to the raspberry pi from the microcontrollers, the first byte represents the system and the second byte represents the status of the system. The raspberry pi compares the received bytes with the preset protocol and logs the corresponding data. The temperature from the LM35 as such is stored in the Raspberry Pi.

Chapter 6

CONCLUSION

6.1 Observation

- In this work, we have implemented Home Automation using Raspberry Pi.
- The two Atmega Microcontrollers were programmed successfully.
- The design and implementation of the individual systems was successful.
- Serial communication between various devices was successful.
- Data was successfully logged in the Raspberry Pi.

6.2 Future Scope

The scope for improvement is immense. The data log from sensors can be shared over the internet and algorithms be implemented to take intelligent decisions regarding the power consumption etc. It could also be used to monitor the national average etc. We could also incorporate wireless sensor networks (WSN) for efficient power management. Raspberry Pi can be configured to act as a radio transmitter. This acts as an emergency radio service. The keypad can be replaced by a voice recognition system. Also we could integrate voice control to almost all the features. The various sensors can be connected to the central microcontroller unit via Zigbee Modules.

REFERENCES AND LINKS

[1] "*Home automation",* http://en.wikipedia.org/wiki/Home_automation

[2] "*ATmega 328P Data sheet*" from, http://www.atmel.com/images/atmel-8271-8-bit-avr- microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet.pdf

[3] *"Raspberry Pi"* from http://www.raspberrypi.org/

[4] *"Serial Communication"* , http://forum.arduino.cc/index.php/topic,219394.0.html

[5] Atmega 328P programming available from http://arduino.serial.programming/atmega328p

[6] LM35 temperature sensor available at https://www.ti.com/products/sensors/lm35.pdf

[7] PIR available at http://en.wikipedia.org/wiki/Passive_Infrared_Sensor

[8] DS1307 available at www.instructables.com/rpiandds1307

APPENDIX 1

Cost Description

Serial No:-	Materials	Quantity	Cost
1.	Raspberry Pi	1	3200
2.	Atmega 328P	2	400
3.	LM35	1	Sampled
4.	LDR	2	5
5.	DS1307	1	Sampled
6.	LCD Display	1	Salvaged from old
7.	PCB	1	300
8.	Board(Casing)	1	150
9.	Other Components	1	800
TOTAL		**4855**	

APPENDIX 2

MICROCONTROLLER 1-PROGRAM

```
/*
Main Program
==========


adaptive lighting
al-adaptive lighting

external lighting
el-external lighting

temperature mesuring system
tm-temperature measurement

Atmega1:
PIR:
i/p - A0

*/

#include <EEPROM.h>
//preprosessor definitions
```

```
#define pirinput 9
#define alinput A1
#define aloutput 10
#define aloverride 12
#define altoggle 13
#define althresholdlow 500
#define althresholdmedium 650
#define althresholdhigh 900
#define aleeprom 1
#define elinput A2
#define eloutput 7
#define threshold 500
#define eloverride 5
#define eltoggle 6
#define eleeprom 2

#define tminput A3

//variable defenitions

int serial_input;

int pirstate = LOW; // we start, assuming no motion detected
int pirvalue = 0; // variable for reading the pin status
int pir_status=0;

int alstate,alsensor;

int elstate,elsensor;

int tm;
```

```
//functions
int pir();
void adaptive_internal_lighting_system(); //adaptive internal lighting system function
void external_lighting_system(); //external lighting system function
void temperature_measurement(); //temperature measurement function
void alstatus();
void elstatus();

//setup function
void setup()
{
  pinMode(pirinput, INPUT); // declare sensor as input
  pinMode(aloverride,INPUT);
  pinMode(altoggle,INPUT);
  alstate=EEPROM.read(aleeprom);

  pinMode(eloutput,OUTPUT);
  pinMode(eloverride,INPUT);
  pinMode(eltoggle,INPUT);
  elstate=EEPROM.read(eleeprom);

  delay(1000);
  Serial.begin(9600);

  delay(1000);
}

//cloop function
void loop()
{
```

```
  adaptive_internal_lighting_system(); //call the adaptive internal lighting system function
  external_lighting_system(); //call the external lighting system function
  temperature_measurement();//call the temperature measurement function
  delay(100);
}

void serialEvent()
{
  if(Serial.available()==1)
  { serial_input=Serial.read();
    if(serial_input==20)
     alstatus();
    else if(serial_input==30)
     elstatus();
    else if(serial_input==60)
     temperature_measurement();
  }
}



int pir()
{
pirvalue = digitalRead(pirinput); // read input value
if (pirvalue == HIGH)
    {
     // check if the input is HIGH
     if (pirstate == LOW)
       {
         // we have just turned on
         // We only want to print on the output change, not state
         pirstate = HIGH;
```

```
        return pirstate;
       }
    }
else
  {
        if (pirstate == HIGH)
       {
        // we have just turned of
        pirstate = LOW;
        return pirstate;
       }
  }
}


//adaptive internal lighting system function
void adaptive_internal_lighting_system()
{
  pir_status=pir();
  if(pir_status==0&&alstate!=5)
  { Serial.print(25);
    alstate=5;
    EEPROM.write(aleeprom,alstate);
  }
  if(pir_status==0&&alstate==5)
   return;
 if(!digitalRead(aloverride))
 { alsensor = analogRead(alinput); //read the sensor value
   if(alsensor>=althresholdlow)
   { analogWrite(aloutput,20); //output the PWM to the LED
     if(alsensor>=althresholdmedium&&alsensor<althresholdhigh)
```

```
     analogWrite(aloutput,127);
    else if(alsensor>=althresholdhigh)
     analogWrite(aloutput,255);
    if(alstate!=2)
    {
      Serial.print(22);
      alstate=2;
      EEPROM.write(aleeprom,alstate);
    }
  }
  else
  { if(alstate!=1)
    { analogWrite(aloutput,0);
      alstate=1;
      Serial.print(21);
      EEPROM.write(aleeprom,alstate);
    }
  }
}
else //override function
{ if(digitalRead(altoggle)) //toggle - ON
  { analogWrite(aloutput,255);
    if(alstate!=4)
    { alstate=4;
      Serial.print(24);
      EEPROM.write(aleeprom,alstate);
    }
  }
  else //toggle - OFF
  { analogWrite(aloutput,0);
    if(alstate!=3)
```

```
    { alstate=3;
      Serial.print(23);
      EEPROM.write(aleeprom,alstate);
    }
   }
 }
 return;
}

void alstatus()
{
  if(alstate==1)
   Serial.print(21);
  else if(alstate==2)
   Serial.print(22);
  else if(alstate==3)
   Serial.print(23);
  else if(alstate==4)
   Serial.print(24);
  else
   Serial.print(25);
  return;
}

//external lighting system function
void external_lighting_system()
{
  if(!digitalRead(eloverride))
  { elsensor=analogRead(elinput);
    if(elsensor>threshold)
    { digitalWrite(eloutput,HIGH);
```

```
   if(elstate!=1)
   { Serial.print(32);
     elstate=1;
     EEPROM.write(eleeprom,elstate);
   }
  }
  else if(elsensor<threshold)
  { digitalWrite(eloutput,LOW);
   { if(elstate!=0)
     { Serial.print(31);
       elstate=0;
       EEPROM.write(eleeprom,elstate);
     }
   }
  }
}
else //override function
{ if(digitalRead(eltoggle))
  { digitalWrite(eloutput,HIGH);
   if(elstate!=3)
   { Serial.print(34);
     elstate=3;
     EEPROM.write(eleeprom,elstate);
   }
  }
  else
  { digitalWrite(eloutput,LOW);
   if(elstate!=2)
   { Serial.print(33);
     elstate=2;
     EEPROM.write(eleeprom,elstate);
```

```
      }
    }
  }
  return;
}

void elstatus()
{
  if(elstate==1)
    Serial.print(31);
  else if(elstate==2)
    Serial.print(32);
  else if(elstate==3)
    Serial.print(33);
  else
    Serial.print(34);
  return;
}
//temperature measurement system
void temperature_measurement()
{
  tm=analogRead(tminput);//read the temperature value
  //tmvoltage = tm * (5 / 1024);
  Serial.print(61);
  Serial.print(tm);
  return;
}
```

MICROCONTROLLER 2 –PROGRAM

```
/*
Atmega 2
======= =

water pump system
pc-pumpcontrol

*/



//preprosessor definitions

#include <Keypad.h>
#include<String.h>
#include<EEPROM.h>




#define pcinput A2
#define pcoutput 11
#define pcoverride 6
#define pctoggle 7

#define pcinput11 A0
#define pcinput21 A1
#define pcinput22 A2
#define pcoutput A3
#define pcled A4
#define pcoverride A5
#define pctoggle 13
```

```
#define pceeprom 6


const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
char t;

byte rowPins[ROWS] = {5, 4, 3, 2}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {8, 7, 6}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
byte ledPin = 13;
byte motorpin1=12;
byte motorpin2=11;
boolean blink = false;
boolean ledPin_state;
int addr=0,i,c[5];
char b[5],key;

//variable defenitions

int pcstate;

int serial_input;
```

```
//functions
void pump_control_system(); //automatic pump control system function
void KEYPAD();
void motorup();
void motordown();
void keypadEvent(KeypadEvent key);




//setup function
void setup()
{
 pinMode(pcinput11,INPUT);
 pinMode(pcinput21,INPUT);
 pinMode(pcinput22,INPUT);
 pinMode(pcoverride,INPUT);
 pinMode(pctoggle,INPUT);
 pinMode(pcoutput,OUTPUT);
 pinMode(pcled,OUTPUT);
 pcstate=EEPROM.read(pceeprom);

 pinMode(ledPin, OUTPUT);             // Sets the digital pin as output.
 digitalWrite(ledPin, HIGH);          // Turn the LED on.
 ledPin_state = digitalRead(ledPin);   // Store initial LED state. HIGH when LED is on.
 keypad.addEventListener(keypadEvent); // Add an event listener for this keypad
 for(i=0,addr=0;i<4;i++,addr++)
 {
  c[i]=EEPROM.read(addr);
  b[i]=c[i]+48;
 }
```

```
  addr=0;

  delay(1000);
  Serial.begin(9600);
  delay(1000);
}


//loop function
void loop()
{ pump_control_system(); //call the automatic pump control system function
  KEYPAD();
  delay(1);
}


void serialEvent()
{
  if(Serial.available()==1)
  { serial_input=Serial.read();
    if(serial_input==40)
     pcstatus();
  }
}


//pump control system
void pump_control_system()
{
  if(digitalRead(pcinput11==1)&&pcstate!=5)
  { Serial.print(45);
```

```
  pcstate=5;
  EEPROM.write(pceeprom,pcstate);
  digitalWrite(pcled,HIGH);
  delay(500);
  digitalWrite(pcled,LOW);
}
if(pcstate==5)
 return;
pcstate=1;
if(!digitalRead(pcoverride))
{ int a,b;
  a=digitalRead(pcinput21);
  b=digitalRead(pcinput22);
  if(a==1&&b==1&&pcstate!=2)
  { digitalWrite(pcoutput,HIGH);
    Serial.print(42);
    pcstate = 2;
    EEPROM.write(pceeprom,pcstate);
  }
  else if(a==0&&b==0&&pcstate!=1)
  { digitalWrite(pcoutput,LOW);
    Serial.print(41);
    pcstate = 1;
    EEPROM.write(pceeprom,pcstate);
  }
}
else //override function
{ if(digitalRead(pctoggle)==1&&pcstate!=4)
  { digitalWrite(pcoutput,HIGH);
    Serial.print(44);
    pcstate = 4;
```

```
      EEPROM.write(pceeprom,pcstate);
    }
    else if(pctoggle==0&&pcstate!=3)
    { digitalWrite(pcoutput,LOW);
      Serial.print(43);
      pcstate = 3;
      EEPROM.write(pceeprom,pcstate);
    }
  }
  return;
}

void pcstatus()
{
  if(pcstate==1)
   Serial.print(31);
  else if(pcstate==2)
   Serial.print(32);
  else if(pcstate==3)
   Serial.print(23);
  else
   Serial.print(24);
  return;
}

void KEYPAD()
{ int i,j,k,l;
  //Serial.println("Enter password:");
  char a[5];
  for(i=0;i<4;i++)
  { char key = keypad.waitForKey();
```

```
  //if (key)
  //{
  //    Serial.println(key);
  //}
  a[i]=key;
}
a[i]='\0';
if(!strcmp(a,b))
{ //Serial.println("Password is correct\n");
  Serial.print(53);
  for(j=0;j<100;j++)
  { for(k=0;k<2000;k++)
    { if(keypad.getKey()=='*')
      { //Serial.println("Enter new password");
        for(l=0,addr=0;l<4;l++)
        { b[l]=keypad.waitForKey();
          c[l]=b[l]-48;
          EEPROM.write(addr,c[l]);
          //Serial.println(b[l]);
          addr++;
        }
        Serial.print(54);
      }
    }
  }
  //Serial.println("Delay over");
  motorup();
  key = keypad.waitForKey();
  if(key=='#')
   motordown();
}
```

```
  else
   Serial.print(53);
  addr=0;
  if (blink)
    { digitalWrite(ledPin,!digitalRead(ledPin));       // Change the ledPin from Hi2Lo or
Lo2Hi.
    delay(100);
  }
}

void motorup()
{
  digitalWrite(motorpin1,HIGH);
  digitalWrite(motorpin2,LOW);
  delay(2500);
  digitalWrite(motorpin1,LOW);
}
void motordown()
{
  digitalWrite(motorpin2,HIGH);
  digitalWrite(motorpin1,LOW);
  delay(2500);
  digitalWrite(motorpin2,LOW);
}
// Taking care of some special events.
void keypadEvent(KeypadEvent key)
{ switch (keypad.getState()){
  case PRESSED:
      if (key == '#') {
          digitalWrite(ledPin,!digitalRead(ledPin));
          ledPin_state = digitalRead(ledPin);        // Remember LED state, lit or unlit.
```

```
        }
        break;

    case RELEASED:
        if (key == '*') {
            digitalWrite(ledPin,ledPin_state);     // Restore LED state from before it started blinking.
            blink = false;
        }
        break;

    case HOLD:
        if (key == '*') {
            blink = true;    // Blink the LED when holding the * key.
        }
        break;
    }
}
```

```
#import libraries
import glob, random, sys, vlc, time  #glob-load the mp3 files names
                        #random-shuffle the tracks
                        #sys-for exit()
                        #vlc-music player
import RPi.GPIO as GPIO  #gpio buttons
from Adafruit_CharLCD import *

#cli arguments check
if len(sys.argv) <= 1:  #to exit if no input folder is present
        print("Please specify a folder with mp3 files")
        sys.exit(1)

folder = sys.argv[1]
files = glob.glob(folder+"/*.mp3")
if len(files) == 0:  #checks for mp3 file are present or not
        print("No mp3 files in directory", folder, "..exiting")
        sys.exit(1)

random.shuffle(files)

#vlc setup
player = vlc.MediaPlayer()
medialist = vlc.MediaList(files) #medialist-playlist player
mlplayer = vlc.MediaListPlayer()
mlplayer.set_media_player(player)
mlplayer.set_media_list(medialist)

#gpio setup
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
```

RASPBERRY PI-MUSIC PLAYER PROGRAM

```
PLAY_BUTTON=11
STOP_BUTTON=7
BACK_BUTTON=4
FORWARD_BUTTON=10
GPIO.setup(PLAY_BUTTON, GPIO.IN)
GPIO.setup(STOP_BUTTON, GPIO.IN)
GPIO.setup(BACK_BUTTON, GPIO.IN)
GPIO.setup(FORWARD_BUTTON, GPIO.IN)

#lcd setup
lcd = Adafruit_CharLCD()
lcd.clear()
lcd.message("Hit play!")

def handle_changed_track(event, player):
        media = player.get_media()
        media.parse()
        artist = media.get_meta(vlc.Meta.Artist) or "Unknown artist"
        title = media.get_meta(vlc.Meta.Title) or "Unknown song title"
        album = media.get_meta(vlc.Meta.Title) or "Unknown song"
        lcd.clear()
        lcd.message(title+"\n"+artist+"-"+album)
        playerem = player.event_manager()

playerm.event_attach(vlc.EventType.MediaPlayerMediaChanged,handle_changed_track,
player)

#while loop
while True:
```

```
#button = input("Hit a button")
if GPIO.input(PLAY_BUTTON):
        print("Pressed play button")
        if mlplayer.is_playing():
                mlplayer.pause()
        else:
                mlplayer.play()
elif GPIO.input(STOP_BUTTON):
        print("Pressed stop button")
        mlplayer.stop()
        random.shuffle(files)
        medialist = vlc.MediaList(files)
        mlplayer.set_media_list(medialist)
elif GPIO.input(BACK_BUTTON):
        print("Pressed back button")
        mlplayer.previous()
elif GPIO.input(FORWARD_BUTTON):
        print("Pressed forward button")
        mlplayer.next()
#else:
#       print("Unrecognised input")
time.sleep(0.3)
lcd.scrollDisplayLeft()
```

RASPBERRY PI-DATA LOGGING PROGRAM

```
import time
import serial
ser = serial.Serial('/dev/ttyAMA0', 9600, timeout=1)
ser.open()
def log(x):
   u=time.strftime("%d-%m-%Y")
   t=time.strftime("%H:%M:%S")
   path = "/home/pi/test/1/dj/rpi/" + u +".txt"
   rpilog=open(path,'a')
   t=time.strftime("%H:%M:%S")
   rpilog.write('\n'+repr(t)+'\t')
   rpilog.write(x)
   rpilog.close()
   try:
   while 1:
      response = ser.read(2)
      print response
                  if response[0]=="2":
                  if response[1]=="1":
                        log("Adaptive Internal Lighting system - OFF - Auto Mode")
                  elif response[1]=="2":
                        log("Adaptive Internal Lighting system - ON - Auto Mode")
                  elif response[1]=="3":
                        log("Adaptive Internal Lighting system - OFF - Manual Mode")
                  elif response[1]=="4":
                        log("Adaptive Internal Lighting system - ON - Manual Mode")
                  elif response[0]=="3":
                  if response[1]=="1":
                        log("External Lighting system - OFF - Auto Mode")
```

```
                elif response[1]=="2":
                        log("External Lighting system - ON - Auto Mode")
                elif response[1]=="3":
                        log("External Lighting system - OFF - Manual Mode")
                elif response[1]=="4":
                        log("External Lighting system - ON - Manual Mode")
                elif response[0]=="4":
                if response[1]=="1":
                        log("Water pumping system - OFF - Auto Mode")
                elif response[1]=="2":
                        log("Water pumping system - ON - Auto Mode")
                elif response[1]=="3":
                        log("Water pumping system - OFF - Manual Mode")
                elif response[1]=="4":
                        log("Water pumping system - ON - Manual Mode")
                elif response[0]=="5":
                if response[1]=="1":
                        log("Door is open")
                elif response[1]=="2":
                        log("Door is closed")
                elif response[1]=="3":
                        log("Password enterned incorrect")
                elif response[0]=="6":
                if response[1]=="1":
                       temp1 = ser.read(2)
                      temp2 = ser.read(2)
                      adc=temp2[1]+temp2[0]+temp1[1]+temp1[0]
                log(str(adc))
#       time.sleep(0.2)
except KeyboardInterrupt:
       ser.close()
```