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()
```