# SMART PARKING SYSTEM

**AIM**: Design a prototype of the parking system which keeps the track of empty parking spots and informs the drivers entering a parking spot.

**HARDWARE REQUIREMENTS**:

➢ Node MCU ESP8266

➢ IR Sensor

➢ LCD Display

➢ Jumper wires

**SOFTWARE REQUIREMENTS**:

➢ Arduino IDE

**DESCRIPTION:**  The goal of parking system project is to reserve parking spot for a car/vehicle before it arrives. One of the most problems that driver faces is finding a free parking spot, so many drivers stopping their cars at the edges of the street. Therefore, we choose this to prevent the frustration of finding a parking spot and they can reserve a spot when they stay at home. It is challenging to drive on congested roads, and what is even worse is finding a parking slot. The search for an available parking space takes up potentially productive time.Using this smart parking system, we are able find our spot without any wastage of time. We can see the slots which are available at a particular location by using an application which is specially designed our iOS and android mobile phones for this smart parking system, in this application. We will be able to see the parking slots, when user books a slot in this application automatically the route map to the parking slot location will start in our using navigation through the maps provided by google.

**CIRCUIT DIAGRAM**:

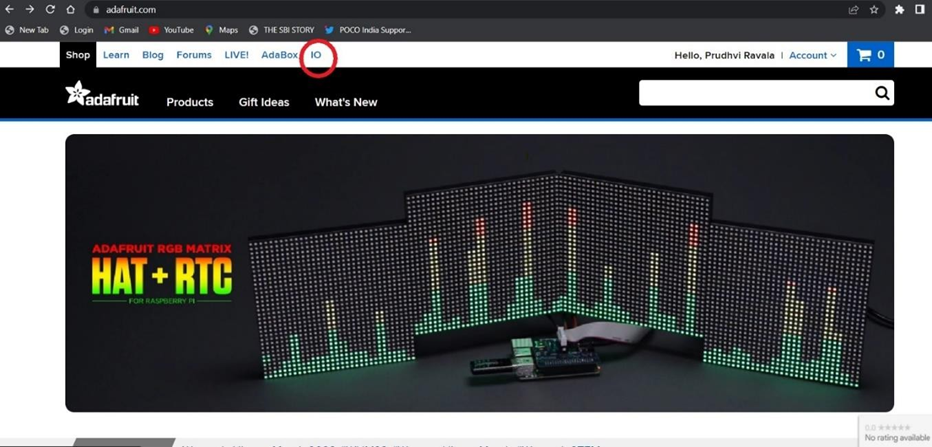


**PROCEDURE:**

1. First open your browser then go to the website https://www.adafruit.com.

2. Login (or) signup into the website using your E-mail id.

3. After logging into the website, we can see on the top of web page named as IO.

4. Click on that IO menu.

5. After clicking on that IO, we can see a new webpage displayed on our window.

6. On that webpage we can see Feeds, click on the feeds tab.

7. Now create 4 feeds naming as

• PARKING 1

• PARKING 2

• PARKING 3

• MAPS

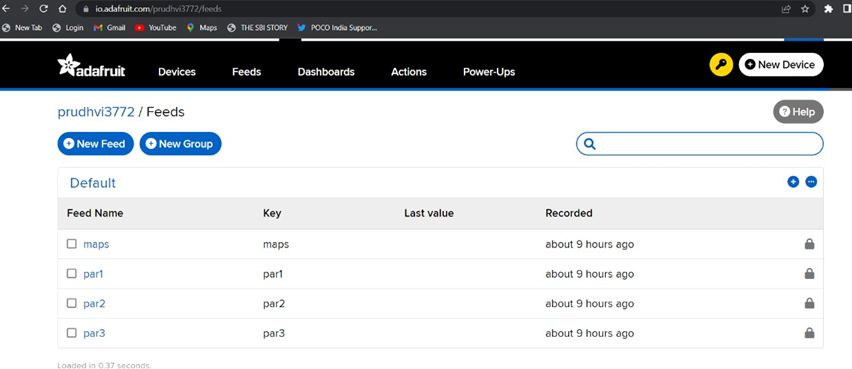
8. In that same webpage we can see another menu named as dashboard, go to the dashboards, and create a new dashboard naming as smart parking system.

9. After creating dashboard, go to the dashboard settings and select create a new block.

10. We can see various type of inputs, now select indicators from it.

11. Connect the feed to the parking 1 from the feed menu, now connect the remaining feeds (parking 2, parking 3) using same process.

12. Then go to the settings again and click on edit block.



13. Modify every indicator block and give the condition and test values inputs as 1.

14. Now go to the dashboard settings and create a new input as map from the menu.

15. Edit the map block, give the title name as location and select map type of view as satellite imagery.

16. After creation of the four blocks go to the Arduino IDE in choose File  Examples  Adafruit IO Arduino  AdafruitIO-00-publish

17. In that we have two files naming as AdafruitIO-00-publish and config

18. We can type the code in AdafruitIO-00-publish.ino file and in that code we will modify the feed names as park 1, park 2, park 3 and maps.

19. Next go to config.h file in that we can give the key token from the Adafruit also give SSID and password of your mobile personal hotspot.

20. Then compile the code after that we can upload the code into Node MCU.

21. After uploading, give the proper connections carefully.

22. Finally upload the code to the serial monitor.

23. Now we can see the output in Adafruit IO dashboard and in LCD display.

24. If the slot is filled it shows red indication otherwise green.

25. Note down all the observations.

**CODE:**

#include "config.h"

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

double lat = 14.913190;

double lon = 79.98295;

double ele = 0;

int value = 0;

//14.913190009548426, 79.98295658367893

// digital pin 5

#define Sensor1 14

#define Sensor2 12

#define Sensor3 13

// button state

bool current1 = false;

bool last1 = false;

bool current2 = false;

bool last2 = false;

bool current3 = false;

bool last3 = false;

int s1,s2,s3,s;

LiquidCrystal\_I2C lcd(0x27, 16, 2);

AdafruitIO\_Feed \*Slot1 = io.feed("counter");

AdafruitIO\_Feed \*Slot2 = io.feed("Humidity");

AdafruitIO\_Feed \*Slot3 = io.feed("Temperature");

AdafruitIO\_Feed \*location = io.feed("stream");

void setup() {

lcd.begin(16,2);

// Turn on the blacklight and print a message.

lcd.backlight();

lcd.setCursor(2,0);

lcd.print("Smart Parking ");

lcd.setCursor(5,1);

lcd.print("System");

delay(2000);

// set button pin as an input

pinMode(Sensor1, INPUT);

pinMode(Sensor2, INPUT);

pinMode(Sensor3, INPUT);

// start the serial connection

Serial.begin(115200);

// wait for serial monitor to open

while(! Serial);

// connect to io.adafruit.com

Serial.print("Connecting to Adafruit IO");

io.connect();

// wait for a connection

while(io.status() < AIO\_CONNECTED) {

Serial.print(".");

delay(500);

}

// we are connected

Serial.println();

Serial.println(io.statusText());

lcd.clear();

}

void loop() {

// io.run(); is required for all sketches.

// it should always be present at the top of your loop

// function. it keeps the client connected to

// io.adafruit.com, and processes any incoming data.

io.run();

lcd.backlight();

lcd.setCursor(0,0);

lcd.print("Available:");

lcd.setCursor(11,0);

lcd.print(s);

lcd.setCursor(0,1);

lcd.print("S1:");

lcd.setCursor(3,1);

lcd.print(digitalRead(Sensor1));

lcd.setCursor(4,1);

lcd.print(" S2: ");

lcd.setCursor(8,1);

lcd.print(digitalRead(Sensor2));

lcd.setCursor(11,1);

lcd.print(" S3:");

lcd.setCursor(15,1);

lcd.print(digitalRead(Sensor3));

delay(2000);

// grab the current state of the button.

// we have to flip the logic because we are

// using a pullup resistor.

if(digitalRead(Sensor1) == LOW){

current1 = true;

s1=0;

}

else

{

current1 = false;

s1=1;

}

if(digitalRead(Sensor2) == LOW){

current2 = true;

s2=0;

}

else{

s2=1;

current2 = false;

}

if(digitalRead(Sensor3) == LOW){

current3 = true;

s3=0;

}

else{

current3 = false;

s3=1;

}

s=s1+s2+s3;

// return if the value hasn't changed

if((current1 == last1) && (current2 == last2) && (current3 == last3))

return;

Serial.println("----- sending -----");

Serial.print("value: ");

Serial.println(value);

Serial.print("lat: ");

Serial.println(lat, 6);

Serial.print("lon: ");

Serial.println(lon, 6);

Serial.print("ele: ");

Serial.println(ele, 2);

// save the current state to the 'digital' feed on adafruit io

Serial.print("sending button -> ");

Serial.println(current1);

Serial.println(current2);

Serial.println(current3);

Slot1->save(current1);

Slot2->save(current2);

Slot3->save(current3);

location->save(value, lat, lon, ele);

value += 1;

//lat -= 0.01;

//lon += 0.02;

//ele += 1;

// store last button state

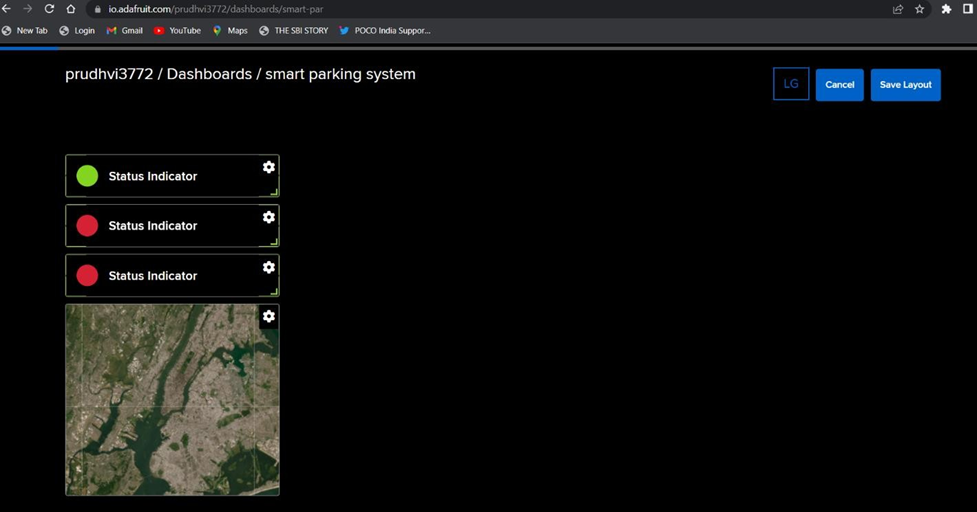
last1 = current1;

last2 = current2;

last3 = current3;

delay(10000);

**OUTPUT**:



# ENVIRONMENT MONITORING SYSTEM

**Aim**: DESIGN AN ENVIRONMENT MONITORING SYSTEM USING IOT .

**Hardware requirements**:

* Node MCU
* DHT11 Sensor
* Wind Sensor
* Buzzer
* LCD Display
* Jumper Wires

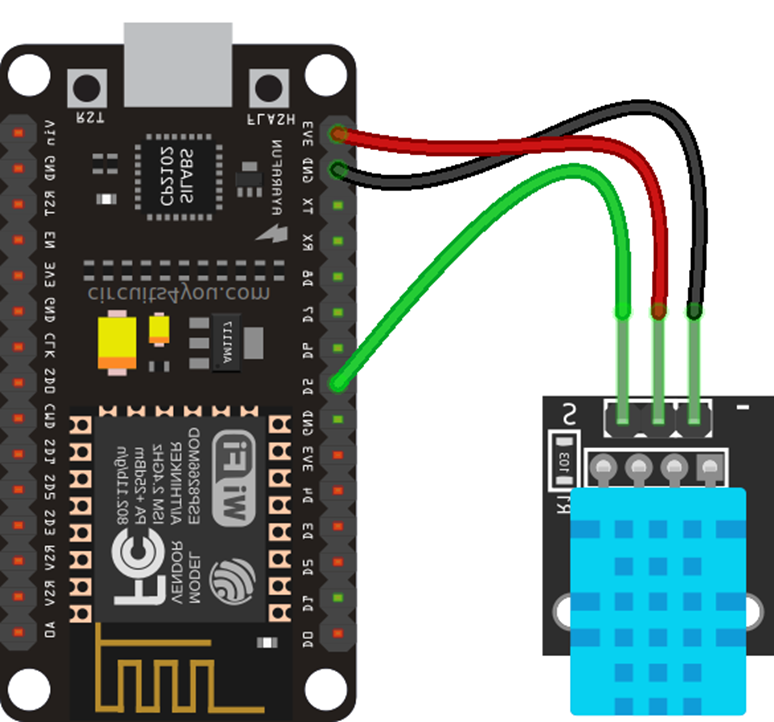
**SOFTWARE REQUIREMENTS**:

* Adafruit cloud
* Arduino software

**DESCRIPTION:**

Temperature and humidity sensors are among the most commonly used environmental sensors. Humidity sensors are also sometimes referred to as hygrometers. These devices are used to provide the actual humidity condition within the air at any given point or in any given place. Humidity is the presence of water within the air. The amount of water vapor that is present in the air can affect not only personal comfort but can also affect various manufacturing processes within industrial applications. Measuring humidity within the environment can be critical due to the fact that the higher the humidity, the warmer it may seem. Temperature and Humidity sensor is devices that can convert temperature and humidity into electrical signals that can easily measure temperature and humidity.

**CIRCUIT DIAGRAM:**



**PROCEDURE:**

1. First to write the code we have to go to file->examples-> Adafruit IO Arduino- >Arduino\_subscribe then we have to write the record regarding our project.

2. While writing the code then we have to give names of feed as of we have provided in the cloud(adafruit.com).

3. After implement the code we have to compile it.

4. To upload the code to Node MCU firstly we have to connect the system to the Node MCU using a data cable.

5. Then we should connect the sensor to the kit using the cables and should provide power supply.

6. Then after all the connections are done correctly then we have to select the feed (which we provide in the code) in the cloud platform (Adafruit.com)

7. Then after selecting the feed we have to select the dashboard of the name Temperature and Humidity. Then we have to create the new block and select the gauge for showing the output of humidity and temperature.

8.After selecting the gauge we have to provide the lower limit for the gauges we have selected.

**SOURCE CODE:**

#include "config.h"

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include "DHT.h"

#define DHTPIN 14

#define Wind A0

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

int Buzzer = 12; //D6

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example Starts Here \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// this int will hold the current count for our sketch

int count = 0;

int count1 = 0;

LiquidCrystal\_I2C lcd(0x27, 16, 2);

// set up the 'counter' feed

AdafruitIO\_Feed \*Temperature = io.feed("Temperature");

AdafruitIO\_Feed \*Humidity = io.feed("Humidity");

AdafruitIO\_Feed \*Windspeed = io.feed("Wind");

void setup() {

// start the serial connection

Serial.begin(115200);

Serial.println(F("DHTxx test!"));

pinMode(Wind,INPUT);

pinMode(Buzzer,OUTPUT);

dht.begin();

lcd.begin(16,2);

// Turn on the blacklight and print a message.

lcd.backlight();

lcd.setCursor(1,0);

lcd.print("Environmental ");

lcd.setCursor(0,1);

lcd.print("Monitoring SYS");

delay(2000);

// wait for serial monitor to open

while(! Serial);

Serial.print("Connecting to Adafruit IO");

// connect to io.adafruit.com

io.connect();

// wait for a connection

while(io.status() < AIO\_CONNECTED) {

Serial.print(".");

delay(500);

}

// we are connected

Serial.println();

Serial.println(io.statusText());

lcd.clear();

}

void loop() {

// io.run(); is required for all sketches.

// it should always be present at the top of your loop

// function. it keeps the client connected to

// io.adafruit.com, and processes any incoming data.

io.run();

int h = dht.readHumidity();

int t = dht.readTemperature();

int wsd= analogRead(Wind);

int ws = map(wsd,0,700,0,150);

if (isnan(h) || isnan(t) ) {

Serial.println(F("Failed to read from DHT sensor!"));

return;

}

float hic = dht.computeHeatIndex(t, h, false);

Serial.print(F("Humidity: "));

Serial.print(h);

Serial.print(F("% Temperature: "));

Serial.print(t);

Serial.print(F("°C "));

Serial.print(F("Wind speed: "));

Serial.print(ws);

Serial.print(F("Km/h"));

lcd.setCursor(0,0);

lcd.print("Wind:");

lcd.setCursor(5,0);

lcd.print(ws);

lcd.setCursor(8,0);

lcd.print(" Km/h");

lcd.setCursor(0,1);

lcd.print("HUM:");

lcd.setCursor(4,1);

lcd.print(h);

lcd.setCursor(6,1);

lcd.print("% TEM:");

lcd.setCursor(12,1);

lcd.print(t);

lcd.setCursor(14,1);

lcd.print("°C");

delay(2000);

if(ws > 80){

digitalWrite(Buzzer,HIGH);

delay(100);

lcd.clear();

lcd.setCursor(3,0);

lcd.print("Hurricane ");

lcd.setCursor(5,1);

lcd.print("Alert!! ");

delay(2000);

digitalWrite(Buzzer,LOW);

delay(100);

lcd.clear();

}

if (count == 0){

// save count to the 'counter' feed on Adafruit IO

Serial.println("sending data -> ");

//Serial.println(count);

Temperature->save(t);

Humidity->save(h);

Windspeed->save(ws);

count1 = count++;

}

if(count1>0){

count1++;

if (count1 >=5){

count=0;

count1=0;

}

}

// increment the count by 1

//count++;

// Adafruit IO is rate limited for publishing, so a delay is required in

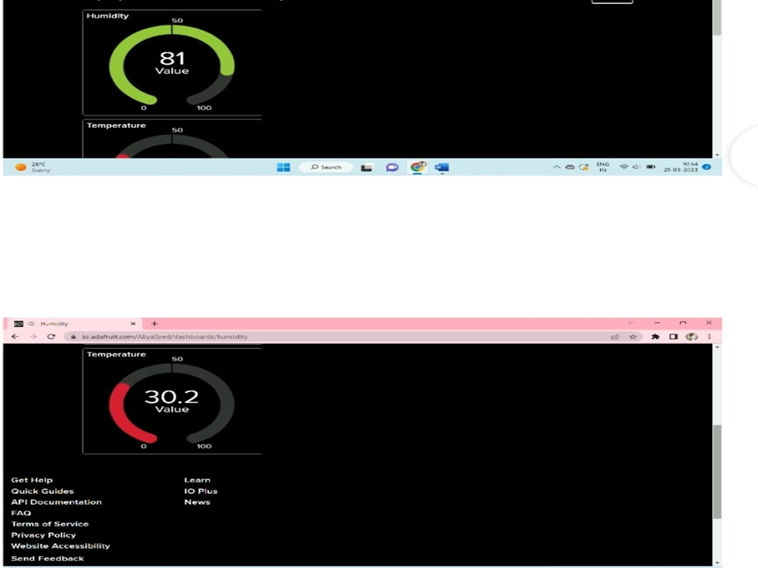
// between feed->save events. In this example, we will wait three seconds

// (1000 milliseconds == 1 second) during each loop.

//delay(30000);

}

**OUTPUT:**



MEDICAL ALERT SYSTEM

**AIM:** To design a medical alert system which alerts the elderly patients whenever it time to take medicine. Particular medicine and other value added information may also be provided.

**HARDWARE REQUIREMENTS**:

• RTC module

• LCD display

• Buzzer

• Node MCU

• Jumper Wires

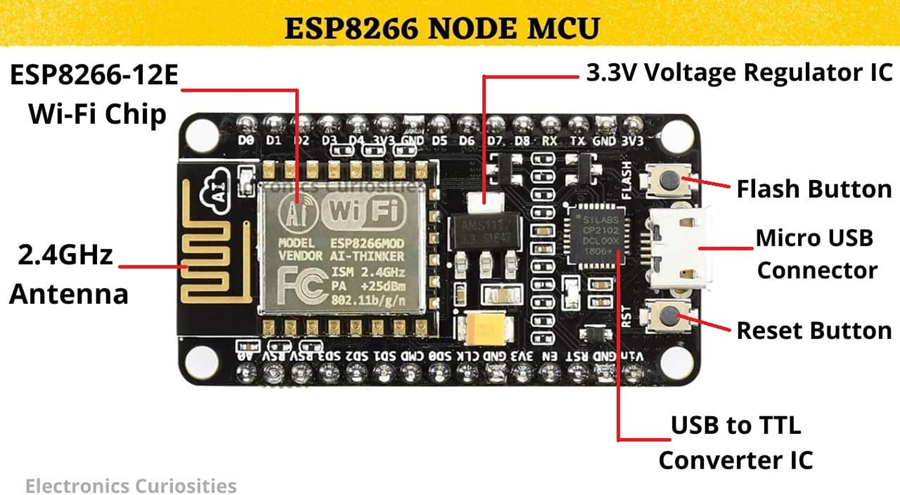
**SOFTWARE REQUIREMENTS**:

* Adafruit cloud
* Arduino software

**DESCRIPTION:**

medical alert system is a device or service that allows individuals with medical conditions or disabilities to quickly call for help in case of an emergency. These systems are designed to help individuals who may be at risk of falling, experiencing a medical emergency, or needing assistance due to a disability or other health condition. The system typically includes a wearable device, such as a pendant or bracelet, that is equipped with a panic button or call button. When the button is pressed, the device sends an alert to a monitoring center staffed by trained operators who can assess the situation and contact emergency services or a designated caregiver or family member. Most medical alert systems use wireless technology to transmit the alert signal, allowing for easy use and mobility. Some systems may also include additional features such as fall detection, GPS tracking, medication reminders, and home automation integration.

Medical alert systems can be purchased or rented and typically require a monthly fee for monitoring services. They can be used by individuals of any age who may need assistance due to a medical condition or disability, but are particularly useful for seniors who live alone, have limited mobility, or have chronic medical conditions.



***RTC Module:***

RTC is an electronic device in the form of an Integrated Chip (IC) available in various packaging options. The purpose of an RTC or a real-time clock is to provide precise time and date which can be used for various applications. We have DS3231, DS1307, DS1302, SeeedStudio RTC module available. The RTC module is ideal in any time-critical applications like attendance systems, digital cameras, automated sprinkler systems, or a time and calendar display.

***Buzzer:***

An audio signaling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.

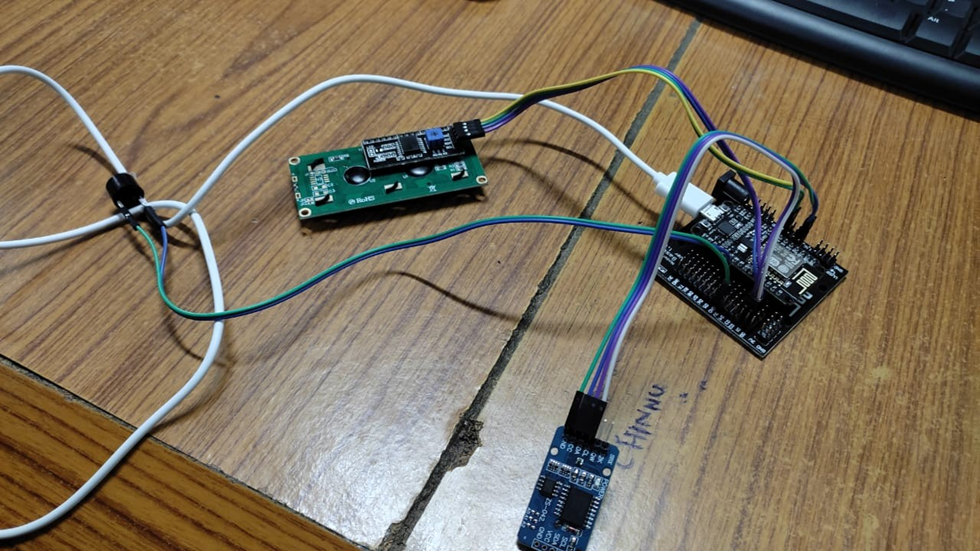
***LCD Display:***

LCD (Liquid Crystal Display) is a type of flat panel display which uses liquid crystals in its primary form of operation. LEDs have a large and varying set of use cases for consumers and businesses, as they can be commonly found in smartphones, televisions, computer monitors and instrument panels.

***NODE MCU:***

The Node MCU (Node Micro Controller Unit) is an open source software and hardware development environment that is built around a very inexpensive System-on-a-Chip (SoC) called the Node MCU.

**CIRCUIT DIAGRAM**:

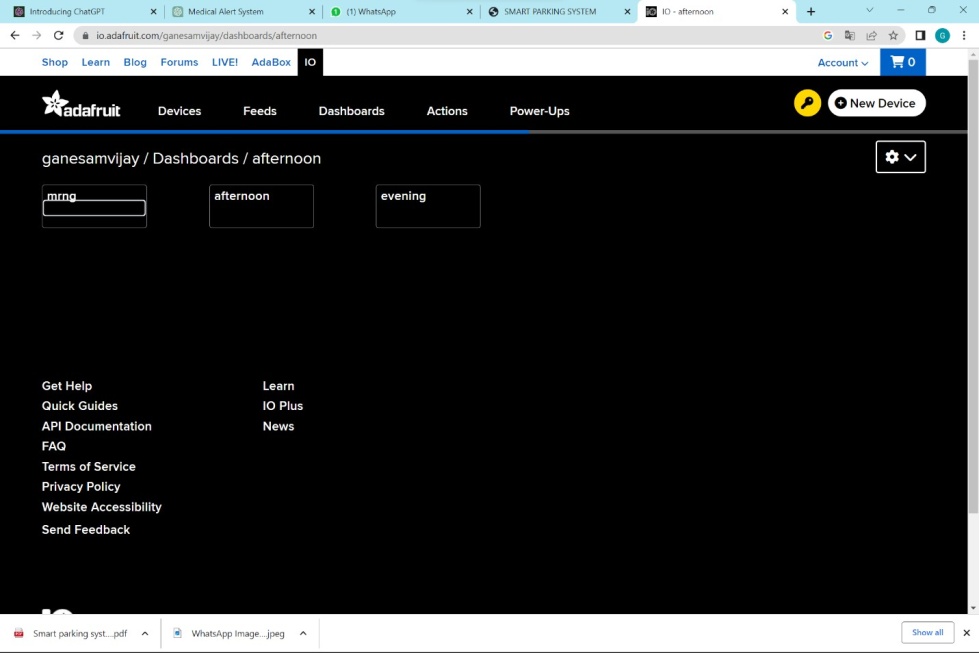


**PROCEDURE:**

1. First open your browser then gothewebsitehttps://www.adafruit.com

2. Login (or) signup into the website using your E-mail id.

3. After logging into the website, we can see on the top of web page named as IO.



5. Clicking on that IO, we can see a new webpage displayed on our window.

6. On that webpage we can see Feeds, click on the feeds tab.

7. Now create one feed After naming as relay

8. In that same webpage we can see another menu named as dashboard, go to the dashboards and create a new dashboard naming as medical alert system.

9. After creating dashboard, go to the dashboard settings and select create a new block.

10. We can see various type of inputs, now select text from it.

11. Connect the feed to the relay from the feed menu, now connect the fee

name to dash board using same process.

12. Then go to the settings again and click on edit block.

13. Modify every text block and give the condition and test values inputs as

16. After creation of the four blocks go to the Arduino IDE in choose File 

Examples  Adafruit IO Arduino  AdafruitIO-00-publish.

17. In that we have two files naming as AdafruitIO-00-publish and config.

18. We can type the code in AdafruitIO-00-publish.ino file and in that code

we will modify the feed name as relay.

19. Next go to config.h file in that we can give the key token from the

Adafruit also give SSID and password of your mobile personal hotspot.

20. Then compile the code after that we can upload the code into Node

MCU.

21.After upload the code into nodemcu comment this line of source code.

//rtc.adjust(DateTime(2023, 3, 4, 19, 15, 0));

22. Go to the source code and copy the assigned time and paste it in Adafruit

23.After pasting in AdaFruit, we need to set the time when we want the

medicine alert to sound.M1=09:00+M2=12:00+M3=20:0024. After uploading, give the proper connections carefully.

25. Finally upload the code to the serial monitor.

26. Now we can see the output in Adafruit IO dashboard and in LCD

display.

27.Then finally buzzer wiil rang like a medicine remainder

**SOURCE CODE:**

/ medicine reminder//

#include "config.h"

#include "RTClib.h"

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

RTC\_DS1307 rtc;

LiquidCrystal\_I2C lcd(0x27, 16, 2);

int h1,m1,h2,m2,h3,m3;

String hs1,ms1,hs2,ms2,hs3,ms3;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example Starts Here\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Buzzer = 12; //d6

// Relay is connected to PyPortal's D3 connector

#define RELAY\_POWER\_PIN 12 //D6

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

// Set up the 'relay feed'

AdafruitIO\_Feed \*relay = io.feed("Medicine");

void setup() {

lcd.begin(16,2);

// Turn on the blacklight and print a message.

lcd.backlight();

lcd.setCursor(2,0);

lcd.print("Smart Medicine ");

lcd.setCursor(5,1);

lcd.print("Reminder");

delay(2000);

pinMode(Buzzer,OUTPUT);

// start the serial connection

Serial.begin(115200);

pinMode(RELAY\_POWER\_PIN, OUTPUT);

// wait for serial monitor to open

while(! Serial);

if (! rtc.begin()) {

Serial.println("Couldn't find RTC");

Serial.flush();

while (1) delay(10);

}

if (! rtc.isrunning()) {

Serial.println("RTC is NOT running, let's set the time!");

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

// This line sets the RTC with an explicit date & time, for example to set

// January 21, 2014 at 3am you would call:

}

// When time needs to be re-set on a previously configured device, the

// following line sets the RTC to the date & time this sketch was compiled

// rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

// This line sets the RTC with an explicit date & time, for example to set

// January 21, 2014 at 3am you would call:

//rtc.adjust(DateTime(2023, 3, 4, 19, 15, 0));

Serial.print("Connecting to Adafruit IO");

// connect to io.adafruit.com

io.connect();

// set up a message handler for the 'relay' feed.

// the handleMessage function (defined below)

// will be called whenever a message is

// received from adafruit io

relay->onMessage(handleMessage);

// wait for a connection

while(io.status() < AIO\_CONNECTED) {

Serial.print(".");

delay(500);

}

// we are connected

Serial.println();

Serial.println(io.statusText());

// Get the last known value from the feed

relay->get();

lcd.clear();

}

void loop() {

// io.run(); is required for all sketches.

// it should always be present at the top of your loop

// function. it keeps the client connected to

// io.adafruit.com, and processes any incoming data.

io.run();

DateTime now = rtc.now();

Serial.print(now.year(), DEC);

Serial.print('/');

Serial.print(now.month(), DEC);

Serial.print('/');

Serial.print(now.day(), DEC);

Serial.print(" (");

Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);

Serial.print(") ");

Serial.print(now.hour(), DEC);

Serial.print(':');

Serial.print(now.minute(), DEC);

Serial.print(':');

Serial.print(now.second(), DEC);

Serial.println();

lcd.setCursor(0,0);

lcd.print(now.hour(), DEC);

lcd.setCursor(2,0);

lcd.print(":");

lcd.setCursor(3,0);

lcd.print(now.minute(), DEC);

lcd.setCursor(5,0);

lcd.print(" ");

lcd.setCursor(6,0);

lcd.print(now.day(), DEC);

lcd.setCursor(8,0);

lcd.print("/");

lcd.setCursor(9,0);

lcd.print(now.month(), DEC);

lcd.setCursor(11,0);

lcd.print("/");

lcd.setCursor(12,0);

lcd.print(now.year(), DEC);

lcd.setCursor(2,1);

lcd.print(daysOfTheWeek[now.dayOfTheWeek()]);

delay(1000);

// Check to see if the morning scheduled trigger has executed

if ((now.hour() == h1)&& (now.minute() == m1)) {

Serial.println("Medicen 1 reminder ON");

lcd.setCursor(0,0);

lcd.print("Medicen1 Reminder");

lcd.setCursor(0,1);

lcd.print(" !!Alert!! ");

digitalWrite(RELAY\_POWER\_PIN, HIGH);

delay(500);

lcd.clear();

Serial.println("Medicen1 reminder OFF");

digitalWrite(RELAY\_POWER\_PIN, LOW);

delay(50);

}

if (((now.hour() == h2)&& (now.minute() == m2))) {

Serial.println("Medicen 2 reminder ON");

lcd.setCursor(0,0);

lcd.print("Medicen2 Reminder");

lcd.setCursor(0,1);

lcd.print(" !!Alert!! ");

digitalWrite(RELAY\_POWER\_PIN, HIGH);

delay(500);

lcd.clear();

Serial.println("Medicen2 reminder OFF");

digitalWrite(RELAY\_POWER\_PIN, LOW);

delay(50);

}

if ((now.hour() == h3)&& (now.minute() == m3)) {

Serial.println("Medicen3 reminder ON");

lcd.setCursor(0,0);

lcd.print("Medicen3 Reminder");

lcd.setCursor(0,1);

lcd.print(" !!Alert!! ");

digitalWrite(RELAY\_POWER\_PIN, HIGH);

delay(500);

lcd.clear();

Serial.println("Medicen3 reminder OFF");

digitalWrite(RELAY\_POWER\_PIN, LOW);

delay(50);

}

//delay(1000);

}

// this function is called whenever an 'relay' feed message

// is received from Adafruit IO. it was attached to

// the 'relay' feed in the setup() function above.

void handleMessage(AdafruitIO\_Data \*data) {

Serial.print("feed received new data <- ");

Serial.println(data->toChar());

String newdata= data->toChar();

Serial.println(newdata); // M1=09:00+M2=12:00+M3=20:00

hs1= newdata.substring(3, 5); //save 3,4 location

ms1= newdata.substring(6, 8);

hs2= newdata.substring(12, 14);

ms2= newdata.substring(15, 17);

hs3= newdata.substring(21, 23);

ms3= newdata.substring(24, 27);

h1 = hs1.toInt();

Serial.println(h1);

m1 = ms1.toInt();

Serial.println(m1);

h2 = hs2.toInt();

Serial.println(h2);

m2 = ms2.toInt();

Serial.println(m2);

h3 = hs3.toInt();

Serial.println(h3);

m3 = ms3.toInt();

Serial.println(m3);

}

**CODE FOR CONFIG:**

* #defineIO\_USERNAME"gurrammaneesh444"
* #defineIO\_KEY "aio\_FIpd54lO7Zbt5kcJPwNuly5TYAdq"



* #defineWIFI\_SSID"MyHeart"
* #defineWIFI\_PASS"123456789"



* #include"AdafruitIO\_WiFi.h"
* #ifdefined(USE\_AIRLIFT)||defined(ADAFRUIT\_METRO\_M4\_AIRLIFT\_LITE)

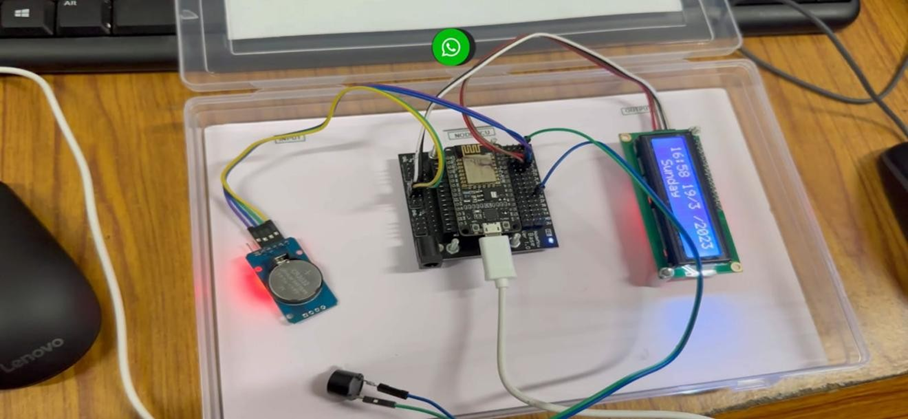
|| \

* defined(ADAFRUIT\_PYPORTAL)



* #if!defined(SPIWIFI\_SS
* #defineSPIWIFISPI
* #defineSPIWIFI\_SS10
* #defineNINA\_ACK9
* #defineNINA\_RESETN6
* #defineNINA\_GPIO0-1
* #endif
* AdafruitIO\_WiFiio(IO\_USERNAME, IO\_KEY, WIFI\_SSID, WIFI\_PASS,SPIWIFI\_SS,
* NINA\_ACK,NINA\_RESETN,NINA\_GPIO0,&SPIWIFI);
* #else
* AdafruitIO\_WiFiio(IO\_USERNAME,IO\_KEY,WIFI\_SSID,WIFI\_PASS);
* #endif
* We have to change adafruit key and wi fi credentials in the above code.

**OUTPUT:**



# SMART STREAM LIGHTING SYSTEM

**Aim**: Design an intelligent stream lighting system.

**Hardware Components:**

* 4 LED lights
* USB Cable
* Bread board
* Jumper wires

**Software Components:**

* Arduino IDE
* ESP8266BoardModule

#### Aadafruit IO Arduino Library

#### **Description:**

* 1. An intelligent stream lighting system is a system that uses advanced technologies to provide optimal lighting conditions for various environments. The system is designed to automatically adjust the lighting based on the environment's conditions and the user's preferences.
  2. The intelligent stream lighting system can also have energy-saving features, such as automatically turning off lights in empty rooms or reducing brightness during low-traffic periods.

**Procedure:** We want to design 3 steps to build an intelligent stream lighting system

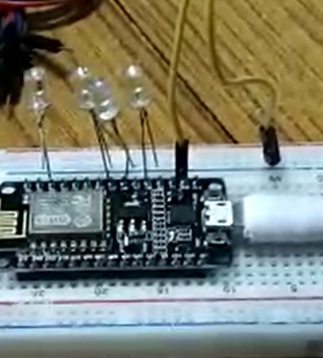
**Step-1:** We have to create dashboard and feed in Adafruit platform

* 1. We want to go to google and search for <https://www.adafruit.com/> and go to Adafruit IO.
  2. Click on feeds, create an new feed as lights and click on dashboard.
  3. Create a new dashboard as lights and click on the name of lights.
  4. Click on settings option in top right side and click on Create New Block.
  5. Then you will then be presented with a list of block and select numbers list and it will open.
  6. Go to Adafruit IO and click on feeds.
  7. Create connect a feed box
  8. Select lights as a feed, then click on“ next step” and set block title, minimum and maximum value and click on “create block”.

**Step-2**: Hardware design.

* Collect all hardware requirement components.
* Take NodeMCU board and upload above code to the Node MCU with the help of USB cable.
* And take 4 LEDs and connect the LED positive pin respectively to D2,D5,D6,D8.
* Then connect all negative pins to GND.

**CIRCUIT DIAGRAM:**

****

**Step-3:**

Arduino program mode for smart streaming lights.

* Open “Arduino IDE” and open file, select examples and select Adafruit IO Arduino, then click on adafruit\_00\_publish.
* It will open two Arduino files as publish and config.h
* In publish we want to copy below code.

**Source Code:**

#include "config.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Example Starts Here \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// this should correspond to a pin with PWM capability

#define LED\_PIN1 4

#define LED\_PIN2 14

#define LED\_PIN3 12

#define LED\_PIN4 15

// set up the 'analog' feed

AdafruitIO\_Feed \*analog = io.feed("counter");

void setup() {

// set up led pin as an analog output

pinMode(LED\_PIN1, OUTPUT);

pinMode(LED\_PIN2, OUTPUT);

pinMode(LED\_PIN3, OUTPUT);

pinMode(LED\_PIN4, OUTPUT);

// start the serial connection

Serial.begin(115200);

// wait for serial monitor to open

while(! Serial);

// connect to io.adafruit.com

Serial.print("Connecting to Adafruit IO");

io.connect();

// set up a message handler for the 'analog' feed.

// the handleMessage function (defined below)

// will be called whenever a message is

// received from adafruit io.

analog->onMessage(handleMessage);

// wait for a connection

while(io.status() < AIO\_CONNECTED) {

Serial.print(".");

delay(500);

}

// we are connected

Serial.println();

Serial.println(io.statusText());

analog->get();

}

void loop() {

// io.run(); is required for all sketches.

// it should always be present at the top of your loop

// function. it keeps the client connected to

// io.adafruit.com, and processes any incoming data.

io.run();

}

// this function is called whenever an 'analog' message

// is received from Adafruit IO. it was attached to

// the analog feed in the setup() function above.

void handleMessage(AdafruitIO\_Data \*data) {

// convert the data to integer

int reading = data->toInt();

Serial.print("received <- ");

Serial.println(reading);

if(reading == 0){

analogWrite(LED\_PIN1, 0);

analogWrite(LED\_PIN2, 0);

analogWrite(LED\_PIN3, 0);

analogWrite(LED\_PIN4, 0);

delay(100);

}

if((reading >0 ) &&(reading <= 25 )){

analogWrite(LED\_PIN1, 50);

analogWrite(LED\_PIN2, 50);

analogWrite(LED\_PIN3, 50);

analogWrite(LED\_PIN4, 50);

delay(100);

}

if((reading >25 ) &&(reading <= 50 )){

analogWrite(LED\_PIN1, 100);

analogWrite(LED\_PIN2, 100);

analogWrite(LED\_PIN3, 100);

analogWrite(LED\_PIN4, 100);

delay(100);

}

if((reading >50 ) &&(reading <= 75 )){

analogWrite(LED\_PIN1, 150);

analogWrite(LED\_PIN2, 150);

analogWrite(LED\_PIN3, 150);

analogWrite(LED\_PIN4, 150);

delay(100);

}

if((reading >75 ) &&(reading <= 100 )){

analogWrite(LED\_PIN1, 250);

analogWrite(LED\_PIN2, 250);

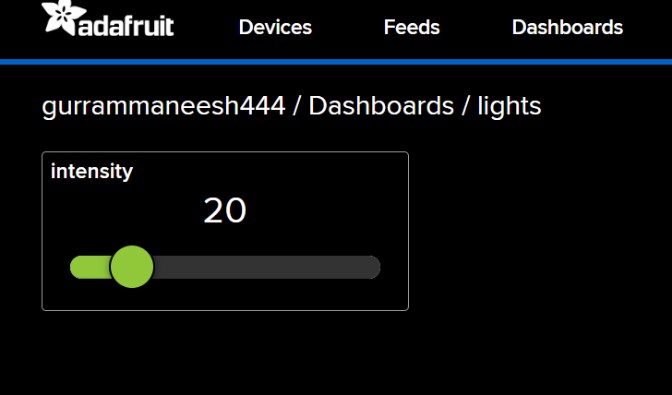
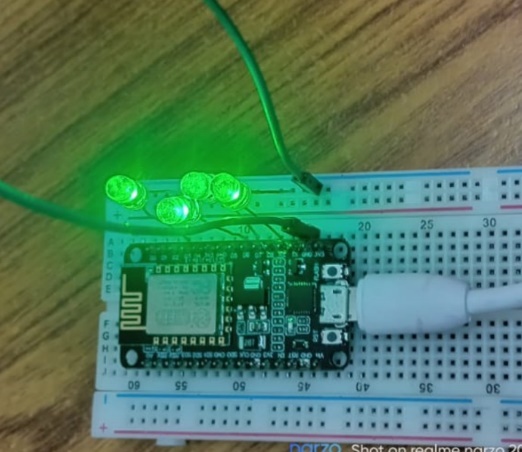
analogWrite(LED\_PIN3, 250);

analogWrite(LED\_PIN4, 250);

delay(1000);

}}

**Output:**



# PID Based Room Temperature Controlling System.

**AIM:** To design the PID Based Room Temperature Controlling System.

**Hardware Components:**

* 1. Node MCU
  2. 4-pin Lcd Display
  3. Relay
  4. MAX6675
  5. HW-131

**Software Components:**

1. Adafruit
2. Arduino IDE
3. Adafruit IO Libraries

**Description:**

* A PID controller is an instrumental used in industrial control applications to regulate temperature, flow, pressure, speed and other process variables.
* PID (Proportional Integral Derivative) controllers use a control loop feedback mechanism to control process variables and are the most accurate and stable controller.

**Procedure:**

* To control your room’s temperature, we can build a small temperature controller. In this case, we use a PID(Proportional-Integral-Derivative) controller.
* When you set a certain temperature, a PID controller will change the temperature by turning either cooler or hotter.

**Connections:**

**RELAY:**

* Input-HW131(5v)
* GND-HW131(GND)
* VCC-Node MCU (D3)

LCD:

* GND-HW131(GND)
* Vcc-HW131(5v)
* SDA-Node MCU(D2)
* SCL-Node MCU(D1)

MAX6675:

* GND-Node MCU (GND)
* Vcc- Node MCU(3v)
* SCK-Node MCU(D5)
* CS-Node MCU(D8SO-NodeMCU(D6)

**Adafruit IO setup for PID based room temperature controlling system.**

* Step1: Create an account in adafruit IO.
* Step2: Click on IO, create a Feed as Relay.
* Step3: Create a new Dashboard.
* Step 4: Now, Copy Username and Key in Adafruit IO.

**Programming Node MCU for PID Based room controlling System**

⮚To program Node MCU with Arduino IDE go to File🡪preferences🡪settings

* Enter https;//Arduino esp8266.com/stable/package\_esp8266.com\_index.json into the Additional Board Manager URL field and click ‘OK’.
* Now go to Tools🡪Board🡪Board Manager.
* In Board manager Window, Type esp in the search box, esp8266 will be listed
* There below. Now select the Latest version of the board and click on install.
* After installation is complete, go to Tools🡪Board🡪and select NodeMCU1.0(ESP-12E Module). Now you can program Node MCU with Arduino IDE.

**SOURCE CODE:**

#include "config.h"

#include "max6675.h"

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Set the LCD address to 0x27 for a 16 chars and 2 line display

LiquidCrystal\_I2C lcd(0x27, 16, 2);

int thermoDO = 12; //12 D6so

int thermoCS = 15; //15 D8cs

int thermoCLK = 14; //14 D5clck

MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);

int PWM\_pin = 2;

//Variables

float temperature\_read = 0.0;

float set\_temperature = 50;

float PID\_error = 0;

float previous\_error = 0;

float elapsedTime, Time, timePrev;

int PID\_value = 0;

//PID constants

int kp = 9.1; int ki = 0.3; int kd = 1.8;

int PID\_p = 0; int PID\_i = 0; int PID\_d = 0;

AdafruitIO\_Feed \*relay = io.feed("Relay");

void setup() {

lcd.begin(16,2);

lcd.setCursor(2,0);

lcd.print("PID SYSTEM FOR");

lcd.setCursor(5,1);

lcd.print("Room Temperature");

delay(2000);

Serial.begin(115200);

pinMode(PWM\_pin,OUTPUT);

Time = millis();

Serial.println("MAX6675 test");

// wait for MAX chip to stabilize

delay(500);

io.connect();

// received from adafruit io

relay->onMessage(handleMessage);

// wait for a connection

while(io.status() < AIO\_CONNECTED) {

Serial.print(".");

delay(500);

}

// we are connected

Serial.println();

Serial.println(io.statusText());

// Get the last known value from the feed

relay->get();

lcd.clear();

}

void loop() {

// basic readout test, just print the current temp

io.run();

temperature\_read = thermocouple.readCelsius();

Serial.print("C = ");

Serial.println(temperature\_read);

//Next we calculate the error between the setpoint and the real value

PID\_error = set\_temperature - temperature\_read;

//Calculate the P value

PID\_p = kp \* PID\_error;

//Calculate the I value in a range on +-3

if(-3 < PID\_error <3)

{

PID\_i = PID\_i + (ki \* PID\_error);

}

//For derivative we need real time to calculate speed change rate

timePrev = Time; // the previous time is stored before the actual time read

Time = millis(); // actual time read

elapsedTime = (Time - timePrev) / 1000;

//Now we can calculate the D calue

PID\_d = kd\*((PID\_error - previous\_error)/elapsedTime);

//Final total PID value is the sum of P + I + D

//Serial.println("P="+String(PID\_p)+"P="+String(PID\_i)+"P="+String(PID\_d));

PID\_value = PID\_p + PID\_i + PID\_d;

//Serial.println("PID="+String(PID\_value));

//We define PWM range between 0 and 255

if(PID\_value < 0)

{ PID\_value = 0; }

if(PID\_value > 255)

{ PID\_value = 255; }

//Now we can write the PWM signal to the mosfet on digital pin D3

int trig = 255-PID\_value;

//analogWrite(PWM\_pin,trig);

if(trig >= 250){

digitalWrite(PWM\_pin,LOW);

}

if(trig < 250){

digitalWrite(PWM\_pin,HIGH);

}

Serial.println("PWM="+String(trig));

previous\_error = PID\_error; //Remember to store the previous error for next loop.

delay(300);

lcd.setCursor(0,0);

lcd.print("PID TEMP control");

lcd.setCursor(0,1);

lcd.print("S:");

lcd.setCursor(2,1);

lcd.print(set\_temperature);

lcd.setCursor(9,1);

lcd.print("R:");

lcd.setCursor(11,1);

lcd.print(temperature\_read);

}

void handleMessage(AdafruitIO\_Data \*data) {

Serial.print("feed received new data <- ");

//Serial.println(data->toint());

int newdata= data->toInt();

Serial.println(newdata);

set\_temperature = newdata;

}

OUTPUT:

