

راه اندازی و انتقال داده های سنسور گاز با رزبری پای از طریق اینترنت



در این آموزش می‌خواهیم اطلاعات یک سنسور تشخیص گاز را از یک محل بخوانیم و از طریق اینترنت و ThingSpeak که یک پلتفرم شناخته شده برای IoT هست به رزبری پای بفرستیم. همچنین رزبری پای با توجه به مقادیر ارسالی، دستور لازم برای فعال سازی یک Device (در اینجا برای سادگی LED گذاشتیم اما شما می‌توانید هر چیز دیگری بگذارید) را به طور مشابه ارسال می‌کند. پس در این قسمت از آموزش رزبری پای با من همراه باشید تا مباحث زیر را یاد بگیریم:

- آشنایی با سنسور تشخیص گاز MQ2
- آموزش نحوه‌ی راه‌اندازی سنسور تشخیص گاز MQ2 با NodeMCU
- آموزش ساخت حساب کاربری، Channel و Field در com
- آموزش انتقال و دریافت داده از Thingspeak توسط رزبری پای و NodeMCU در قالب مثال سیستم تشخیص آلودگی و غلظت گاز

قطعات مورد نیاز:

رزبری پای 4

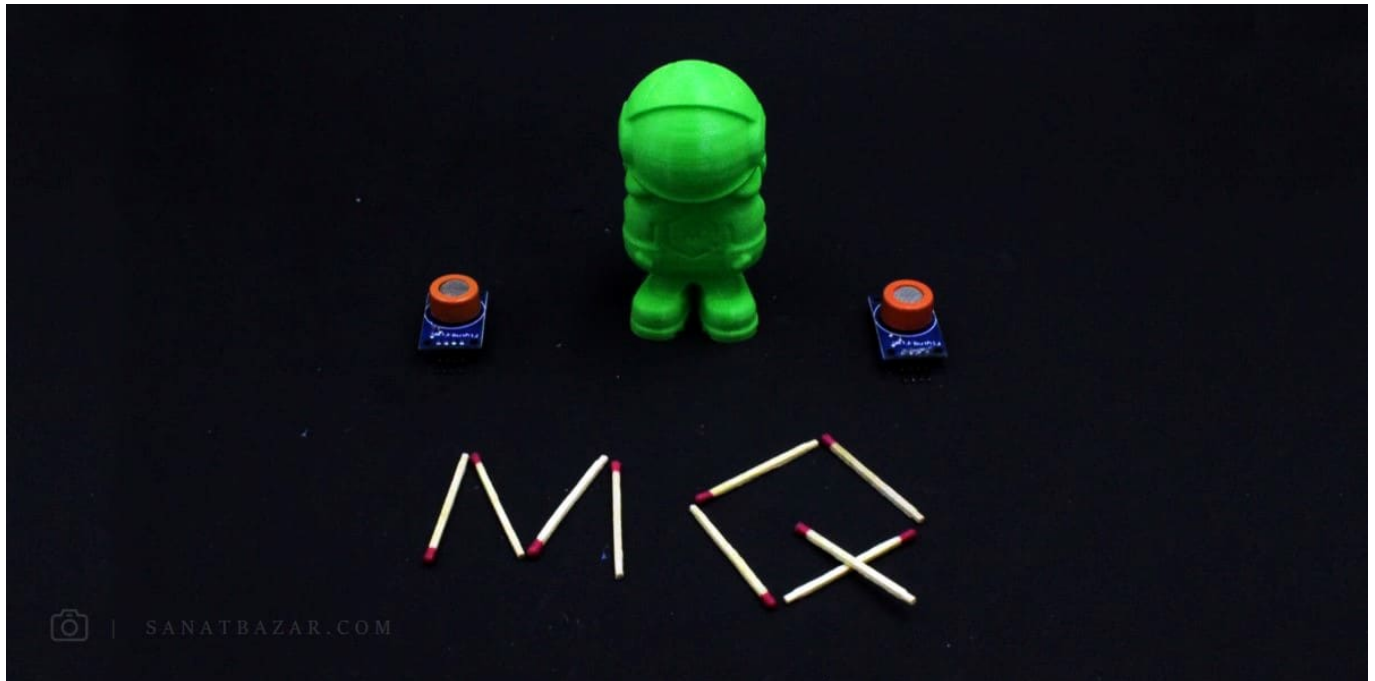
NodeMCU

سنسور تشخیص گاز MQ2

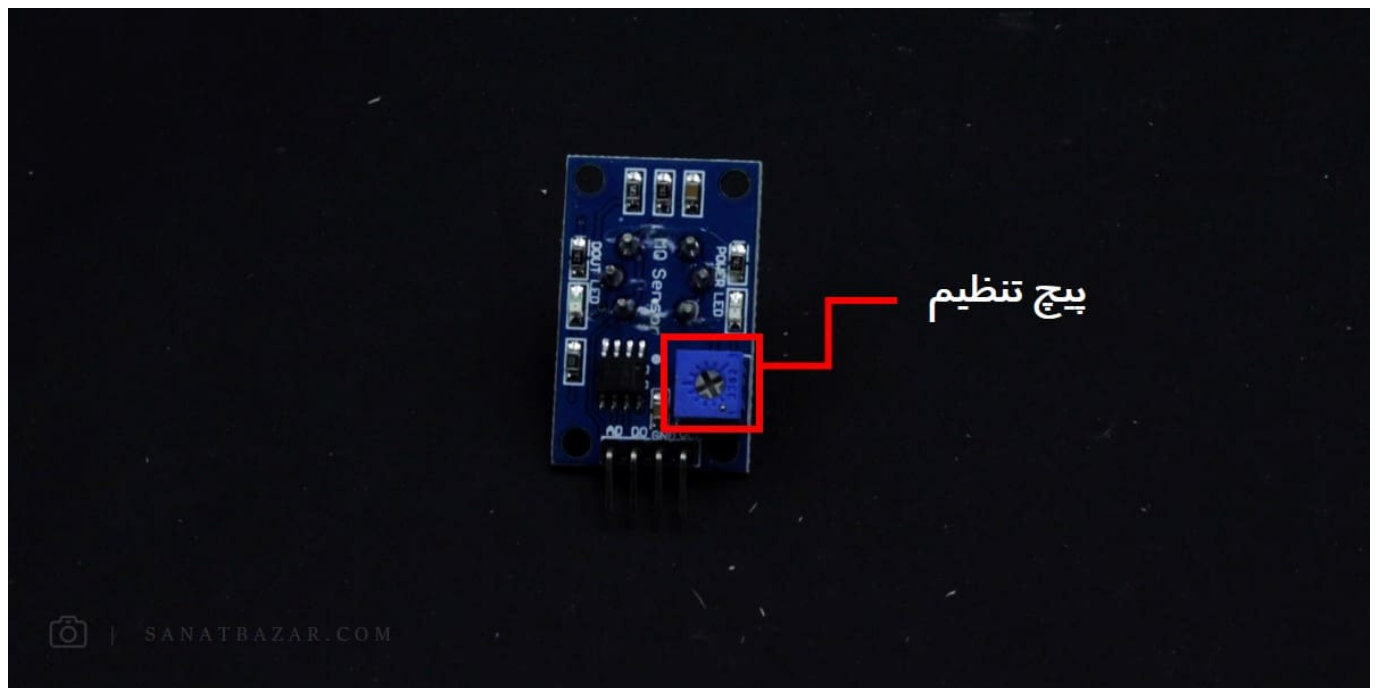
سیم

برد برد

مقدمه: سنسورهای MQ2

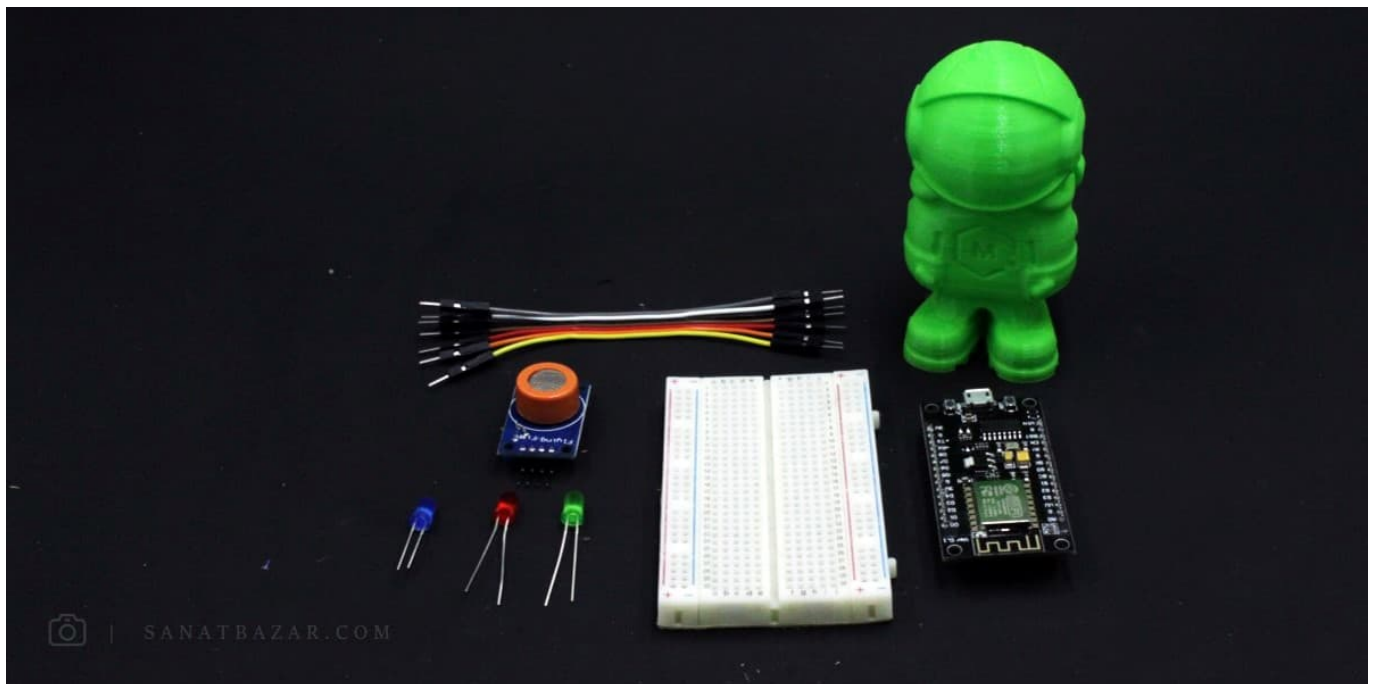


سنسور MQ2 یک حسگر حساس به LPG (گاز مایع)، CO (مونوکسیدکربن)، دود، متان، پروپان و الکل است. انواع دیگری از این سنسور مانند MQ4 و MQ8 نیز در بازار موجود است که از آن‌ها نیز می‌توانید برای تشخیص گاز شهری، هیدروژن و سایر مشتقات گازی و نفتی استفاده کنید. پس با استفاده از این سنسور می‌توانید به راحتی یک دستگاه آنالیز آلودگی هوا برای خودتان بسازید. حتی در صورت تمایل با آموزشی که در ادامه می‌بینیم، می‌توانید مقدار این آلودگی را از هرجایی و از طریق اینترنت ببینید. برای راحتی کاربران، این سنسورها به صورت ماژول و آماده نیز تولید می‌شوند که برای راه‌اندازی آن‌ها کفایت پایه‌های ماژول را به میکروکنترلر خود متصل کرده و از کتابخانه‌های آماده و ساده‌ی موجود برای آن‌ها استفاده کنید. در اینجا ما از ماژول MQ2 برای تشخیص سه نوع گاز CO، LPG و دود استفاده خواهیم کرد. ماژول‌های MQ در واقع یک مقاومت شیمیایی‌اند که با توجه به غلظت گازهای اطراف، مقدار ولتاژ خروجی آن‌ها تغییر می‌کند. مقدار این مقاومت برای هر نوع گاز متفاوت است. بنابراین برای سنجش میزان غلظت آن، کفایت با استفاده از یک تقسیم ولتاژ ساده از مقدار خروجی، ولتاژ متناسب با مقاومت آن گاز را محاسبه کرد. خروجی این ماژول آنالوگ بوده و برای تغذیه به ولتاژ 5V نیاز دارد. همچنین میزان حساسیت این سنسور به گاز مورد نظر شما قابل تنظیم است.



برای کالیبره کردن آن کافیتست منبع گاز (مثلاً دود، فندک یا کبریت در حال اشتعال) را در مقابل آن گرفته و پیچ پتانسیومتر روی ماژول را تا زمانی که (DOUT LED) روشن شود بچرخانید. در ادامه با نحوه‌ی راه‌اندازی آن آشنا خواهیم شد.

راه‌اندازی سنسور MQ2 با NodeMCU

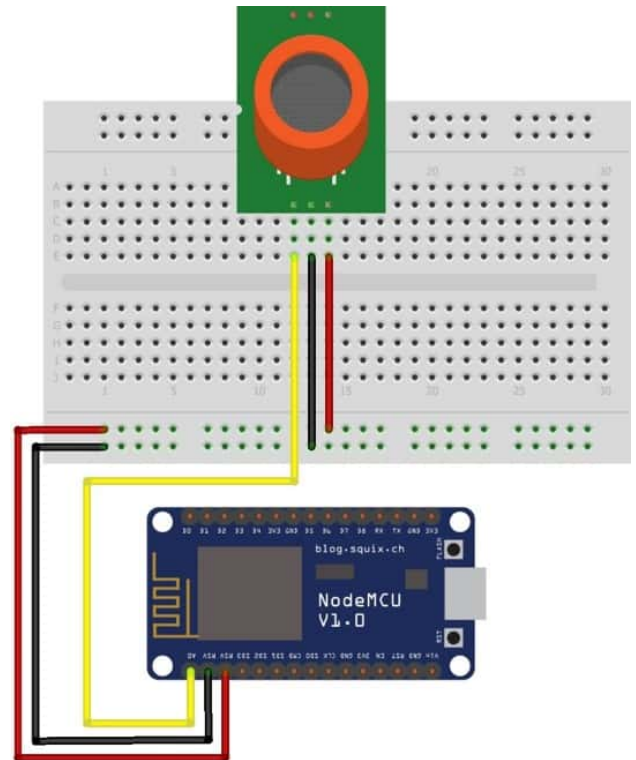


از آنجایی که استفاده از برد رزبری پای برای خواندن داده‌های یک سنسور ساده اصلاً منطقی نیست، در اینجا قصد داریم این حسگر را توسط NodeMCU راه‌اندازی کنیم. اگر با بردهای NodeMCU و ESP آشنا باشید، حتماً می‌دانید که این دسته از میکروکنترلرها به دلیل برخورداری از ماژول Wi-Fi داخلی، برای استفاده در پروژه‌ها انتقال بیسیم داده نسبت به آردوینو برتری دارند. از آنجایی که ما هم در این آموزش قصد داریم داده‌های خروجی از طریق اینترنت به رزبری پای انتقال دهیم، از این برد استفاده می‌کنیم. نحوه‌ی کدنویسی و اجرای دستورات در NodeMCU توسط نرم‌افزار Arduino IDE در بخش انتقال داده‌های حسگر دما و رطوبت DHT بین NodeMCU و رزبری پای از طریق MQTT مفصل توضیح داده شده است. بنابراین در اینجا فرض می‌کنیم تنظیمات اولیه نرم‌افزار برای شناسایی برد NodeMCU را انجام دادید. در غیر این صورت می‌توانید به آموزش ذکر شده مراجعه کنید. پس اگر کتابخانه برد NodeMCU را در نرم‌افزار Arduino نصب کردید، بدون معطلی مدار زیر را ببندید. در غیر این صورت برید نصب کنید و برگردید تا ادامه‌ی آموزش را باهم پیش ببریم. برای اتصال MQ2 به برد مطابق شکل زیر:

- پین A0 (خروجی آنالوگ) سنسور MQ2 را به پین A0 برد NodeMCU
- پین GND سنسور MQ2 را به پایه ی G برد NodeMCU
- پین VCC سنسور MQ2 را به پایه ی (+5V) (VU)

وصل می کنیم. (اگر ماژول شما دارای ۴ پایه بود، پین D0 را آزاد گذاشته و به جایی وصل نکنید.)

با توجه به این که جای برخی پایه ها در نسخه های مختلف NodeMCU متفاوت است، هنگام نصب مدار به نام پایه دقت کنید. در اینجا ما از NodeMCU Ver 0.1 استفاده می کنیم.



fritzing

پس از نصب سخت افزار، ابتدا کتابخانه ی MQ2 را از لینک زیر دانلود و به نرم افزار خود معرفی کنید. سپس کافیسست کد زیر را اجرا کنید:

```
#include <MQ2.h>
#include <ESP8266WiFi.h>

//I2C pins declaration
int Analog_Input = A0;
int lpg, co, smoke;
String LPG,SMOKE,C0;
MQ2 mq2(Analog_Input);

void setup(){
    Serial.begin(115200);

    mq2.begin();
}

void loop(){
```

```

lpg = mq2.readLPG();

co = mq2.readCO();

smoke = mq2.readSmoke();


LPG="LPG: ";

LPG+=lpg;

SMOKE="SMOKE: ";

SMOKE+=smoke;

CO="CO: ";

CO+=co;

Serial.println(LPG);

Serial.println(CO);

Serial.println(SMOKE);

Serial.println(" ");

delay(5000);

}

```

پس از Program کردن برد، با قرار دادن کبریت یا فندک در مقابل سنسور، می‌توانید نتایج کد خود را ببینید. برای من نتایج به شکل زیر است:

در حالت عادی

پس از قرار دادن فندک در مقابل سنسور

Sensor	Value
LPG	4196
CO	425237
SMOKE	26992
LPG	5510
CO	653930
SMOKE	39239
LPG	8429
CO	1115323
SMOKE	60499
LPG	18156
CO	3221569
SMOKE	131786
LPG	15576
CO	2606537
SMOKE	109619
LPG	13621
CO	2165440

COM11

Send

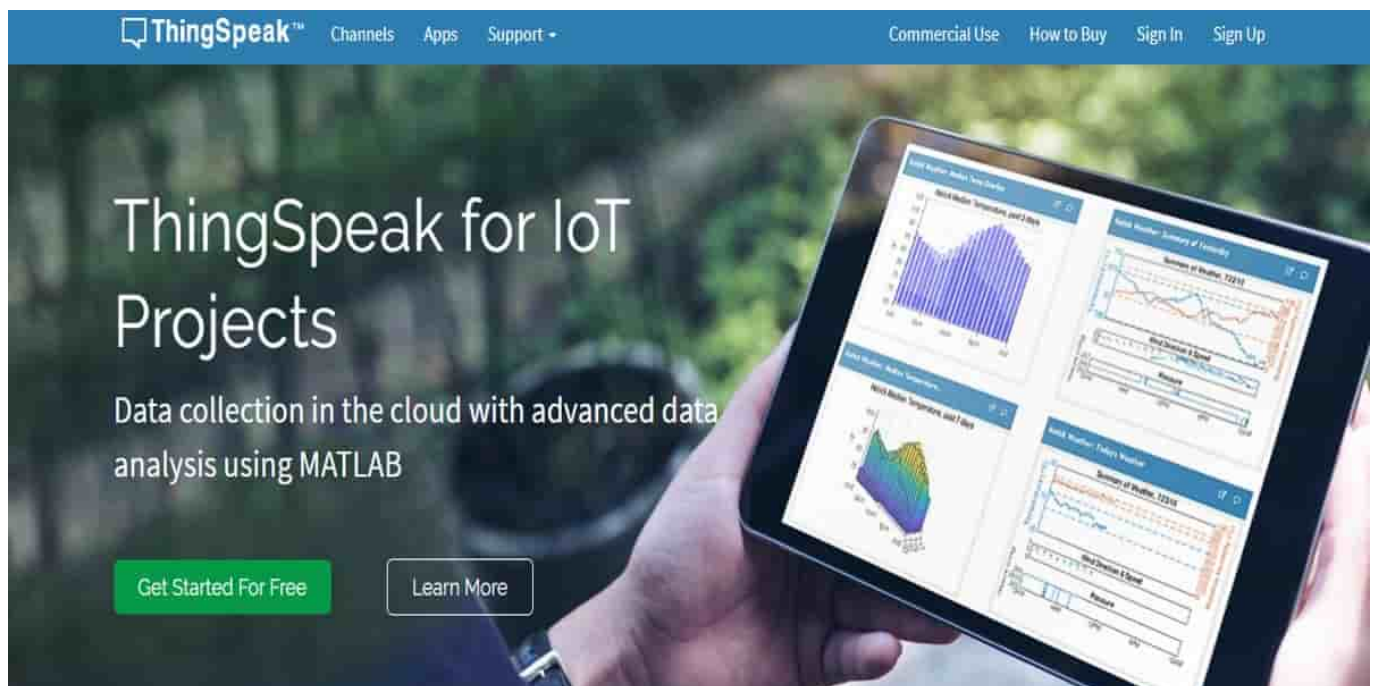
Autoscroll Show timestamp Both NL & CR 115200 baud Clear output

SANATBAZAR.COM

همانطوری که مشاهده می‌کنید، پس از قرار دادن فندک در مقابل سنسور، مقادیر به سمت بالا جهش پیدا کرده و بعد از خاموش کردن آن، مقدار داده‌ها کمتر شده است. خب تا اینجا گام اول پروژه یعنی آماده کردن داده‌ها برای بارگذاری و انتقال بر بستر اینترنت را انجام دادیم. بریم سراغ مرحله‌ی بعد یعنی ThingSpeak!

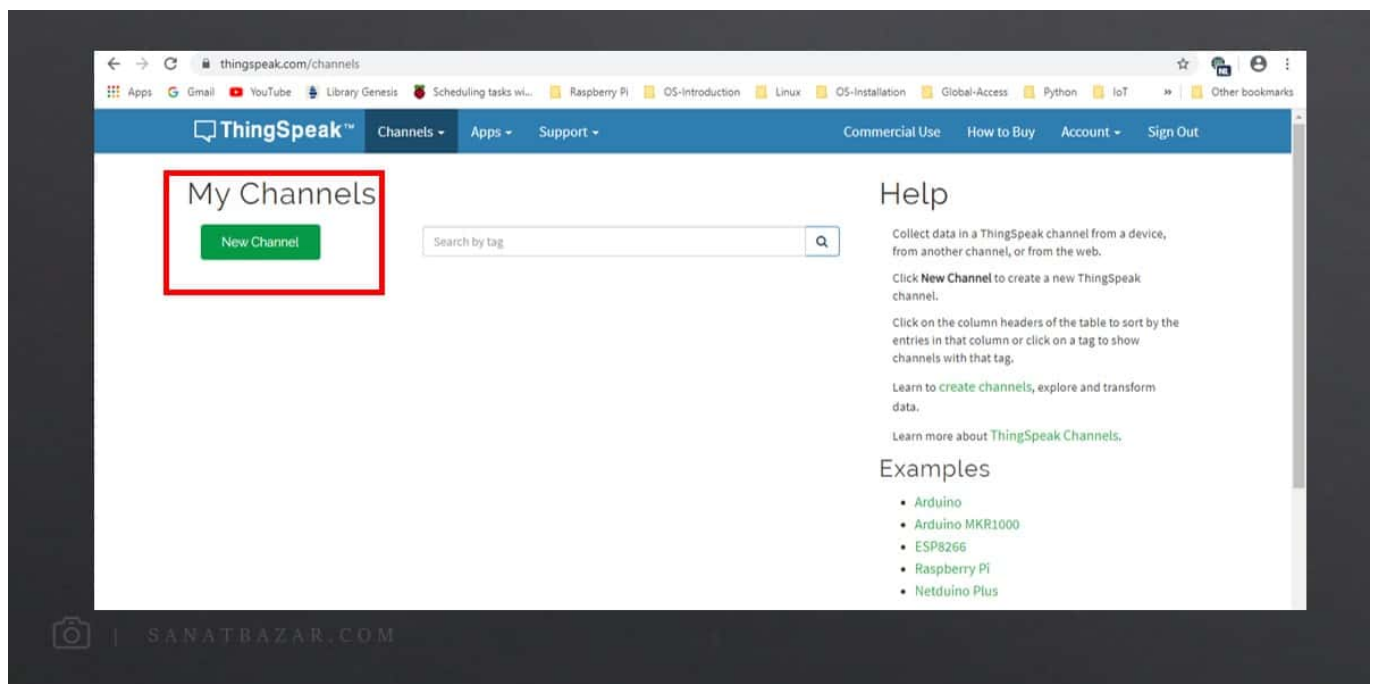
در این آموزش فرض شده که رزبری پای شما از قبل دارای سیستم‌عامل است. در غیر این صورت برای نصب سیستم‌عامل می‌توانید به آموزش راه‌اندازی رزبری پای ۴ با نصب سیستم‌عامل رزبین مراجعه کنید.

آموزش ThingSpeak: مناسب برای مشاهده‌ی داده‌های سنسوری با اینترنت



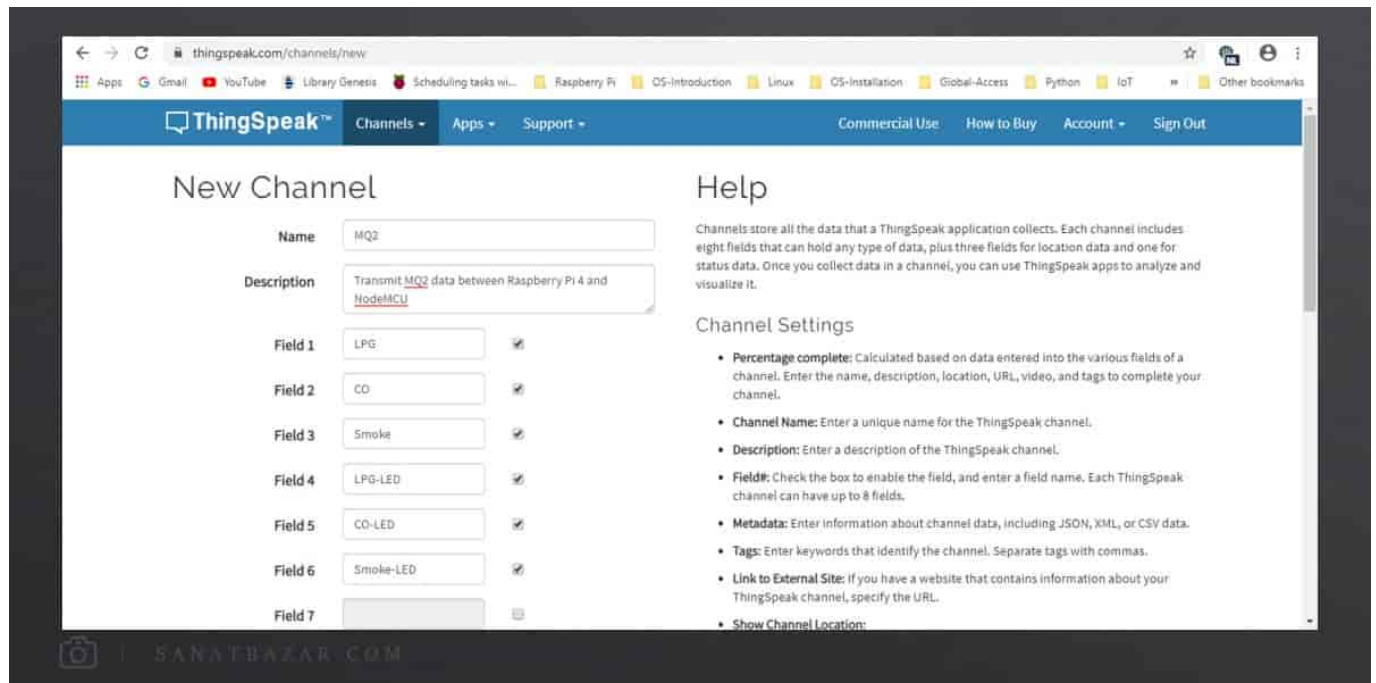
Thingspeak یک پلتفرم IoT است که امکان جمع‌آوری، مشاهده و آنالیز داده‌ها در یک سرور ابری را فراهم می‌کند. بنابراین، با استفاده از این سرویس می‌توانید داده‌های سنسوری خود را در یک Cloud ذخیره، مشاهده و پردازش کنید. از مزیت‌های این سرویس می‌توان به ارتباط مناسب آن با نرم‌افزار MATLAB اشاره کرد. به‌طوری که می‌توانید به‌صورت آنلاین دیتاهای خود را در MATLAB بارگذاری کرده و با استفاده از توابع قدرتمند آن، پردازش‌های مورد نظر خود را انجام دهید. اما از کجا شروع کنیم؟

در ابتدا برای استفاده از ThingSpeak باید به سایت thingspeak.com مراجعه کرده و یک حساب کاربری برای خود ایجاد کنید. (اگر در MATLAB حساب کاربری دارید، می‌توانید از آن استفاده کنید) برای این کار از نوار بالا روی Sign Up کلیک کرده و مشخصات خود را وارد کنید. (متأسفانه کشور ایران در این سایت تریف نشده، برای همین مکان خود را کشور دیگری تعریف کنید) پس از اینکه حساب خود را ساختید و وارد شدید، در صفحه‌ی زیر روی New Channel کلیک کنید. در واقع برای هر کدام از پروژه‌هایتان باید یک کانال مجزا بسازید تا دیتاها در این کانال ذخیره شوند.

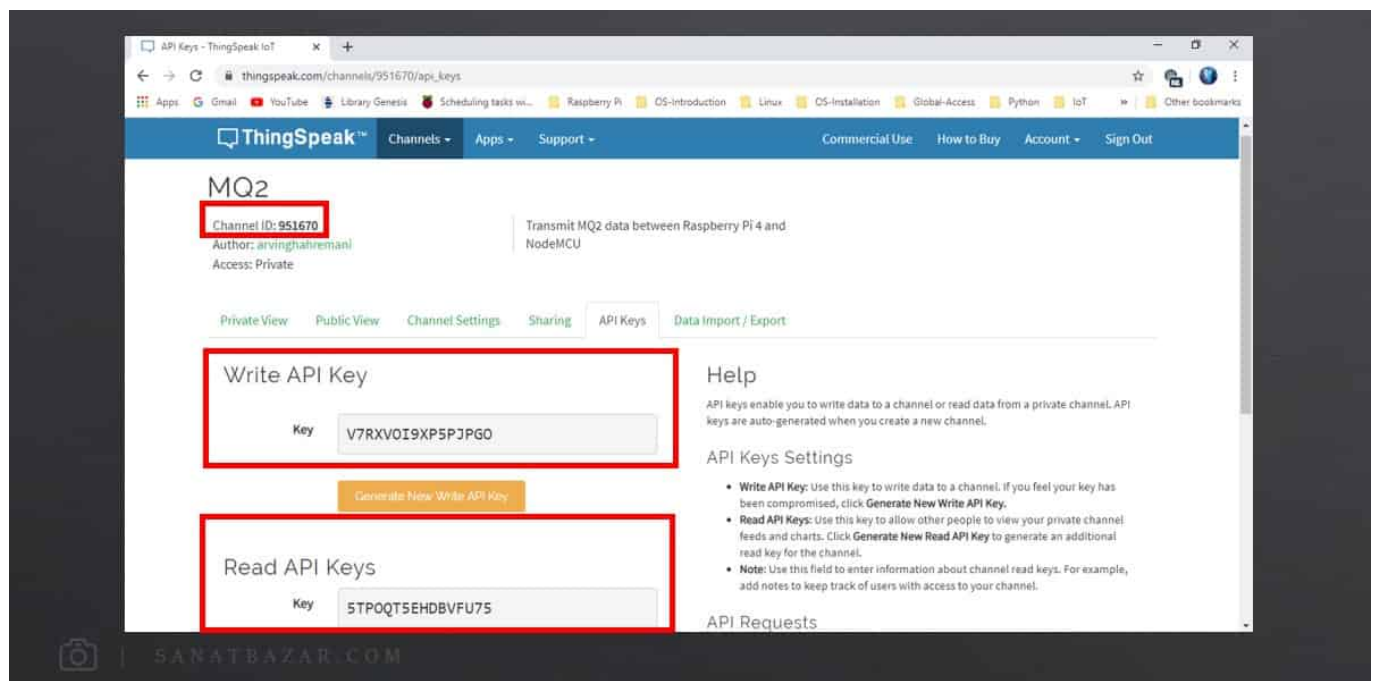


در صفحه‌ی جدید ابتدا برای پروژه و کانال خود نامی انتخاب کنید. من این نام را MQ2 قرار دادم. سپس می‌توانید توضیحاتی را درباره این پروژه در قسمت

Description بنویسید. در قدم بعدی به تعداد متغیرهای که می خواهید بر بستر ای پلتفرم بارگذاری و ارسال شوند، Field ایجاد کنید. در این پروژه من می خواهم ۳ مقدار CO، LPG و Smoke را برای رزبری پای ارسال و دستورات خاموش و روشن کردن LED های مرتبط با آنها را در NodeMCU دریافت کنم. پس به ۶ Field نیاز دارم:



بقیه ی قسمت ها را در صورت نیاز می توانید خودتان تکمیل کنید. در غیر این صورت خای گذاشته و در انتهای صفحه روی Save Channel کلیک کنید. پس از این کار، در صفحه ی جدید به تعداد متغیرهای تعریف شده نمودار مشاهده خواهید کرد. زمانی که داده های خود را روی ThingSpeak قرار دهید این نمودارها مقادیر داده های شما را نمایش خواهند داد. برای این که بتوانی داده ها را به حساب کاربری خود بفراستی به Channel ID و API Key نیاز داریم. بنابراین به بخش API Keys بروید و این مقادیر را یادداشت کنید:



دیگه تنظیمات ThingSpeak تموم شد. حالا وقتشه که کد قبلی را کمی تغییر بدیم تا داده ها روی ThingSpeak دیده شوند. برای این کار کد زیر را در Arduino IDE وارد کنید.

در ابتدا متغیرها و کتابخانه های لازم را تعریف و معرفی می کنیم. همچنین برای اتصال به اینترنت SSID و Password روتر خود را وارد کنید:

```
#include <ESP8266WiFi.h>;
```

```
#include <WiFiClient.h>;
#include <ThingSpeak.h>;
#include <MQ2.h>

const char* ssid = "*****"; //Your Network SSID
const char* password = "*****"; //Your Network Password
int statusCode = 0;
String led_s, led_c, led_l;

#define LED_S 16
#define LED_C 5
#define LED_L 4

//I2C pins declaration
int Analog_Input = A0;
float lpg, co, smoke;
String LPG, SMOKE, CO;
MQ2 mq2(Analog_Input);

WiFiClient client;
```

در بخش بعدی، شماره‌ی شناسایی کانال (Read API Key و Write API Key خود را برای اتصال و خواندن و نوشتن روی ThingSpeak وارد کنید (Channel ID نیازی به " ندارد اما API ها را در " بنویسید!):

```
unsigned long myChannelNumber = *****; //Your Channel Number (Without Brackets)

const char * myWriteAPIKey = "*****"; //Your Write API Key
const char * myReadAPIKey = "*****"; //Your Write API Key

void setup()

{

    Serial.begin(115200);

    // Connect to WiFi network
    delay(10);
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
```



```

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    Serial.print(".");

    ThingSpeak.begin(client);

    mq2.begin();

}

```

پس از برقراری ارتباط با روتر و ThingSpeak در این بخش پین های مورد نظر برای اتصال LED به NodeMCU را تعریف می کنیم.

```

pinMode(LED_S, OUTPUT);

pinMode(LED_C, OUTPUT);

pinMode(LED_L, OUTPUT);

}

```

سپس در حلقه ی اصلی برنامه مقادیر سنسور MQ2 را می خوانیم:

```

void loop()
{

    lpg = mq2.readLPG();
    co = mq2.readCO();
    smoke = mq2.readSmoke();

    LPG = "LPG: ";
    LPG += lpg;

    SMOKE = "SMOKE: ";
    SMOKE += smoke;

    CO = "CO: ";
    CO += co;

    Serial.println(LPG);
    Serial.println(CO);
    Serial.println(SMOKE);
    Serial.println(" ");
}

```

در این بخش مقادیر سنسور را روی Field های مربوطه که در سایت ThingSpeak تعریف کردیم می ریزیم. یعنی LPG در Co، Field 1 در Field 2 و Smoke در Field 3. (چون مقدار داده ها خیلی زیاد بود برای مشاهده ی بهتر و پیوسته تر نمودار مقادیر را بر ۱۰۰ تقسیم کردیم. شما می توانید این کار را نکنید)

```

ThingSpeak.setField(1, lpg / 100);

ThingSpeak.setField(2, co / 100);

ThingSpeak.setField(3, smoke / 100);

ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey); //Update in ThingSpeak

```

حالا در این قسمت می‌خواهیم مقادیر ارسال شده از رزبری پای روی ThingSpeak را بخوانیم. این که چطوری با رزبری پای دیتا بفرستیم را در بخش بعدی به شما خواهیم گفت. در اینجا فرض کنید این دیتا ها روی ThingSpeak هست و شما می‌خواهید آن را بخوانید. برای LED را تعیین کردیم اگر عدد دریافتی ۱ بود، LED روشن و اگر صفر بود، LED خاموش شود.

```
led_s = "Smoke LED is: ";
int LED_smoke = ThingSpeak.readLongField(myChannelNumber, 6, myReadAPIKey);
statusCode = ThingSpeak.getLastReadStatus();
if (statusCode == 200)
{
    if (LED_smoke == 1) {
        led_s += "ON";
        Serial.println(led_s);
        digitalWrite(LED_S, HIGH);
    }
    else
    {
        led_s += "OFF";
        Serial.println(led_s);
        digitalWrite(LED_S, LOW);
    }
}

led_c = "CO LED is: ";
int LED_co = ThingSpeak.readLongField(myChannelNumber, 5, myReadAPIKey);
statusCode = ThingSpeak.getLastReadStatus();
if (statusCode == 200)
{
    if (LED_co == 1) {
        led_c += "ON";
        Serial.println(led_c);
        digitalWrite(LED_C, HIGH);
    }
    else
    {
        led_c += "OFF";
        Serial.println(led_c);
        digitalWrite(LED_C, LOW);
    }
}

led_l = "LPG LED is: ";
int LED_lpg = ThingSpeak.readLongField(myChannelNumber, 4, myReadAPIKey);
statusCode = ThingSpeak.getLastReadStatus();
```

```

if (statusCode == 200)
{
    if (LED_lpg == 1) {
        led_l += "ON";
        Serial.println(led_l);
        digitalWrite(LED_L, HIGH);
    }
    else
    {
        led_l += "OFF";
        Serial.println(led_l);
        digitalWrite(LED_L, LOW);
    }
}

delay(15000);

Serial.println("*****");
}

```

: به شکل زیر خواهد بود NodeMCU کد نهایی برای

```

#include <ESP8266WiFi.h>;
#include <WiFiClient.h>;
#include <ThingSpeak.h>;
#include <MQ2.h>

const char* ssid = "*****"; //Your Network SSID
const char* password = "*****"; //Your Network Password
int statusCode = 0;
String led_s, led_c, led_l;

#define LED_S 16
#define LED_C 5
#define LED_L 4

//I2C pins declaration
int Analog_Input = A0;
float lpg, co, smoke;
String LPG, SMOKE, CO;
MQ2 mq2(Analog_Input);

WiFiClient client;

unsigned long myChannelNumber = *****; //Your Channel Number (Without Brackets)

```

```
const char * myWriteAPIKey = "*****"; //Your Write API Key
const char * myReadAPIKey = "*****"; //Your Write API Key

void setup()

{

    Serial.begin(115200);

    // Connect to WiFi network
    delay(10);
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
        ThingSpeak.begin(client);
        mq2.begin();

    }
    pinMode(LED_S,OUTPUT);
    pinMode(LED_C,OUTPUT);
    pinMode(LED_L,OUTPUT);

}

void loop()

{

    lpg = mq2.readLPG();
    co = mq2.readC0();
    smoke = mq2.readSmoke();
    LPG = "LPG: ";
    LPG += lpg;
    SMOKE = "SMOKE: ";
    SMOKE += smoke;
    C0 = "C0: ";
```

```
C0 += co;

Serial.println(LPG);

Serial.println(C0);

Serial.println(SMOKE);

Serial.println(" ");

ThingSpeak.setField(1, lpg / 100);

ThingSpeak.setField(2, co / 100);

ThingSpeak.setField(3, smoke / 100);

ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey); //Update in ThingSpeak


led_s = "Smoke LED is: ";

int LED_smoke = ThingSpeak.readLongField(myChannelNumber, 6, myReadAPIKey);

statusCode = ThingSpeak.getLastReadStatus();

if (statusCode == 200)

{

    if (LED_smoke == 1) {

        led_s += "ON";

        Serial.println(led_s);

        digitalWrite(LED_S, HIGH);

    }

    else

    {

        led_s += "OFF";

        Serial.println(led_s);

        digitalWrite(LED_S, LOW);

    }

}

led_c = "CO LED is: ";

int LED_co = ThingSpeak.readLongField(myChannelNumber, 5, myReadAPIKey);

statusCode = ThingSpeak.getLastReadStatus();

if (statusCode == 200)

{

    if (LED_co == 1) {

        led_c += "ON";

        Serial.println(led_c);

        digitalWrite(LED_C, HIGH);

    }

    else

    {

        led_c += "OFF";
```

```

        Serial.println(led_c);

        digitalWrite(LED_C, LOW);

    }

}

led_l = "LPG LED is: ";

int LED_lpg = ThingSpeak.readLongField(myChannelNumber, 4, myReadAPIKey);

statusCode = ThingSpeak.getLastReadStatus();

if (statusCode == 200)
{
    if (LED_lpg == 1) {
        led_l += "ON";
        Serial.println(led_l);
        digitalWrite(LED_L, HIGH);
    }
    else
    {
        led_l += "OFF";
        Serial.println(led_l);
        digitalWrite(LED_L, LOW);
    }
}

delay(15000);

Serial.println("*****");

}

```

نتایج برای من به شکل زیر است. همانطور که می بینید، در ساعت 16:54 مقادیر سنسوری جهش داشته و این یعنی در آن زمان شعله ای تشخیص داده شده است:



توجه داشته باشید که داده ها در ThingSpeak هر ۱۵ ثانیه یکبار قابل به روز رسانی اند! خب کار ما با nodeMCU تمام شد. بریم سراغ رزبری پای. می خواهیم با رزبری پای هم بخوانیم هم بنویسیم. ابتدا رزبری پای را به روتر مشترک با NodeMCU وصل کنید. سپس مشابه پروژه های قبلی ابتدا با دستور nano یک فایل متنی برای کدهای خود ایجاد کنید. من نام این فایل را thingspeak.py انتخاب کردم.

اگر با لینوکس آشنا نیستید، نگران نباشید. برای یادگیری لینوکس کافیسیت به بخش آموزش لینوکس برای کار با رزبری پای مراجعه کنید.

در این فایل کافیسیت کد پایتون زیر را ذخیره کنید. در بخش اول مشابه NodeMCU باید کتابخانه و ID و API Key ها را وارد کنیم. (در اینجا API ها و ID را در “ “ قرار می دهیم)

```
import urllib2
import json
import time

WRITE_API_KEY="*****"

READ_API_KEY="*****"

CHANNEL_ID= "*****"
```

سپس در یک حلقه ی بی نهایت، ابتدا مقادیر را از ThingSpeak می خوانیم:

```
while True:
    try:
        #Read from channel

        TS =
        urllib2.urlopen("http://api.thingspeak.com/channels/%s/feeds/last.json?api_key=%s" \
                        % (CHANNEL_ID,READ_API_KEY))

        response = TS.read()

        data=json.loads(response)
```

مقادیر سنسوری در قالب json قابل دریافت اند. بنابراین با دستورات زیر، مقدار هر Field را در متغیر مربوط به خودش می ریزیم:

```
Time = data['created_at']
lpg = data['field1']
co = data['field2']
smoke = data['field3']

print(Time + " " + "Smoke: " + smoke + " " + "CO: " + co + " " + "LPG: " + lpg)

TS.close()
```

سپس با صحیح و خطا تعیین می کنیم هر کدام از پارامترها در چه محدوده ای خطرناک و در چه محدوده ای سالم اند:

```
# write to channel

if float(smoke)>1000:

    smoke_led=1

    smoke_LED="ON"

else:

    smoke_led=0
```

```

smoke_LED="OFF"

if float(co)>20000:
    co_led=1
    co_LED="ON"
else:
    co_led=0
    co_LED="OFF"

if float(lpg)>80:
    lpg_led=1
    lpg_LED="ON"
else:
    lpg_led=0
    lpg_LED="OFF"

```

در نهایت دستور لازم برای روشن و خاموش کردن LED مربوط به هر نوع گاز را در Field مربوط به خودش قرار می دهیم:

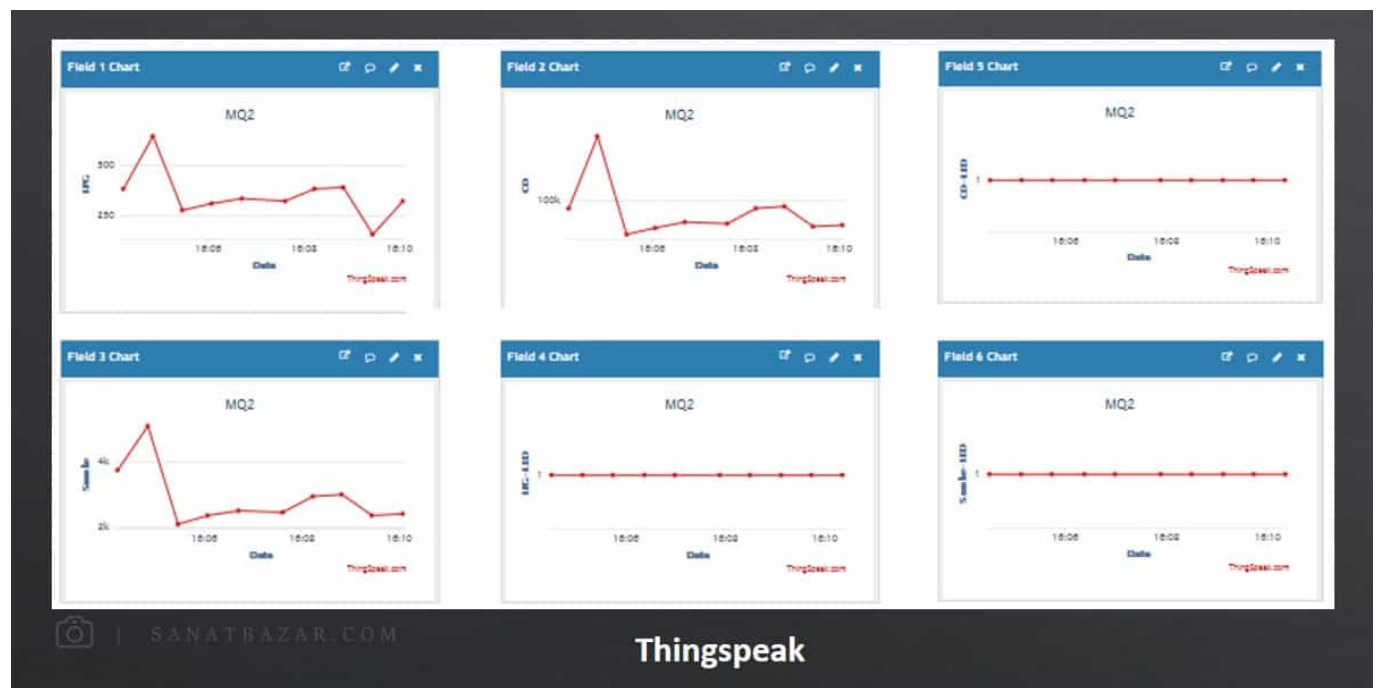
```

baseUrl = 'https://api.thingspeak.com/update?api_key=%s' % WRITE_API_KEY
f= urllib2.urlopen(baseUrl +
                    "&field4=%s&field5=%s&field6=%s" %
                    (lpg_led,co_led,smoke_led))
    print("          "+"Smoke_LED: "+smoke_LED+ "          " +"CO_LED:
    "+co_LED+ "          " +"LPG_LED: "+lpg_LED)
    f.close()
    time.sleep(15)

except TypeError:
    pass

```

در نهایت نتایج در هر برد به شکل زیر خواهد بود:



```

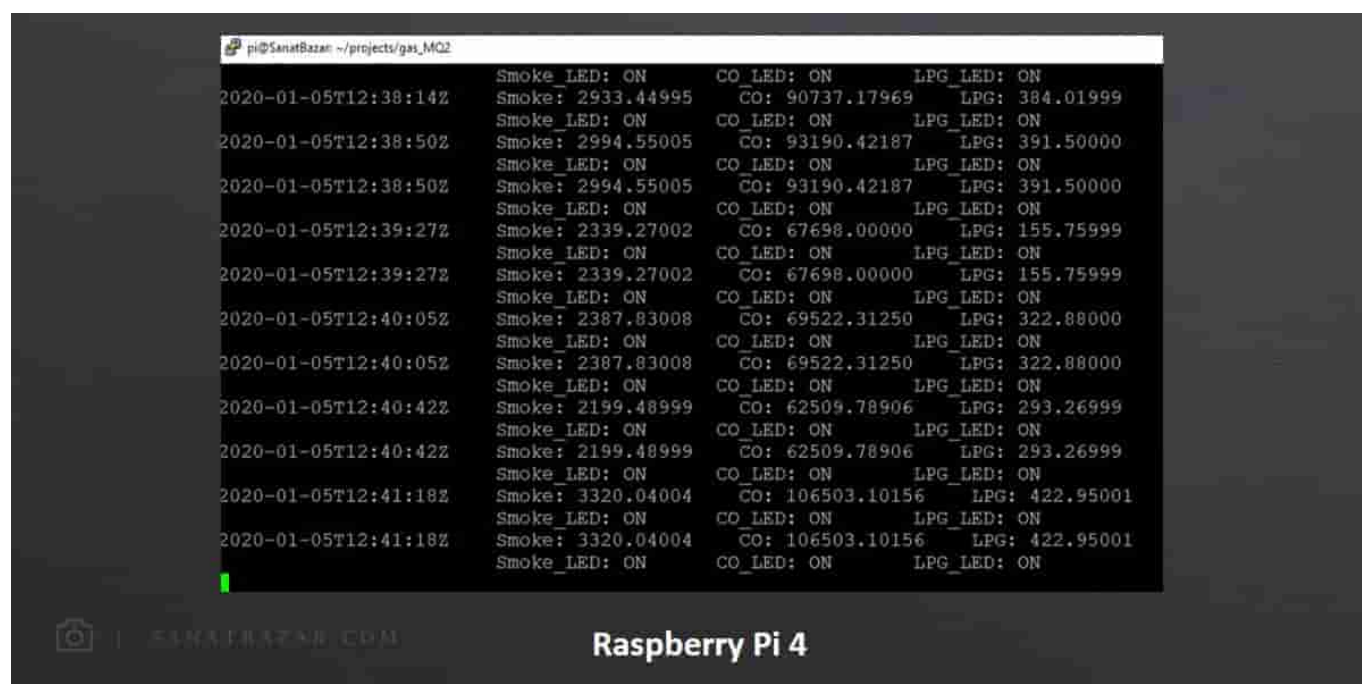
COM11
Send

Smoke LED is: ON
CO LED is: ON
LPG LED is: ON
*****
LPG: 431.20
CO: 106503.10
SMOKE: 3320.04

Smoke LED is: ON
CO LED is: ON
LPG LED is: ON
*****
LPG: 439.62
CO: 109388.98
SMOKE: 3389.34

Smoke LED is: ON
CO LED is: ON
LPG LED is: ON
*****
LPG: 439.62
CO: 109388.98
SMOKE: 3320.04

☒ Autoscroll ☐ Show timestamp Both NL & CR 115200 baud Clear output
  
```



نتیجه گیری

در این پروژه نحوه ی خواندن داده های سنسور پرکاربرد گازی MQ توسط NodeMCU را با هم دیدیم. سپس با پلتفرم ThingSpeak و نحوه ی استفاده از آن آشنا شدیم. این داده ها را از طریق ThingSpeak در اینترنت مشاهده، در رزبری پای پردازش و پاسخ را در برد NodeMCU دریافت کردیم. همانطور که دیدید تمامی این مراحل از طریق اینترنت انجام شده و محدودیت مکانی برای انجام این کار وجود ندارد. شما می توانید به جای داده های MQ هر نوع اطلاعاتی که خواستید بارگذاری و پردازش کنید و پاسخ مناسب را به Node مربوطه بفرستید.

نظرات شما باعث بهبود محتوای آموزشی ما می شود. اگر این آموزش را دوست داشتید، همین طور اگر سوالی در مورد آن دارید، از شنیدن نظراتتان خوشحال خواهیم شد.