

انتقال بیسیم دما و رطوبت بین رزبری پای و NodeMCU



مهم‌ترین قسمت یک پروژه‌ی IOT چیست؟ انتقال اطلاعات (و البته مصرف انرژی!). به‌طوری که اطلاعات از بخش‌های مختلف دریافت و به یک سرور یا پردازشگر مرکزی ارسال می‌شوند. همچنین در طرف دیگر هم، پاسخ و دستور مناسب از پردازشگر به عملگرها (Actuators) منتقل می‌گردد. بنابراین اگر قصد دارید در این حوزه‌ی مهم و آینده‌دار قدم بردارید، باید با انواع ماژول‌ها، بردها، پروتکل‌ها و روش‌های استفاده از آن‌ها آشنا باشید. به همین دلیل در این بخش از آموزش جامع رزبری پای، قصد داریم نحوه‌ی انتقال داده بین رزبری پای و Esp8266 را باهم بررسی کنیم. اما چه داده‌ای را منتقل کنیم؟ در کنار آموزش این ارتباط، می‌خواهم نحوه‌ی راه‌اندازی یکی از سنسورهای بسیار بسیار کاربردی یعنی حسگر دما و رطوبت DHT را نیز با شما به اشتراک بگذارم. بنابراین با من همراه باشید تا در ادامه، نحوه‌ی ارسال و دریافت داده‌های حسگر دما و رطوبت DHT توسط NodeMCU و رزبری پای را گام به گام، عملی و خیلی ساده در طی بخش‌های زیر ببینیم:

- معرفی و راه‌اندازی سنسور دما و رطوبت DHT11 و DHT22 با رزبری پای
- معرفی ماژول Esp8266 و نحوه‌ی راه‌اندازی آن
- معرفی و نصب MQTT روی رزبری پای
- آموزش نحوه‌ی انتقال داده (Local) به‌صورت مستقیم و بی‌سیم بین رزبری پای و NodeMCU با MQTT

قطعات مورد نیاز:

برد رزبری پای ۴

سنسور دما و رطوبت DHT22

Breadboard

سیم

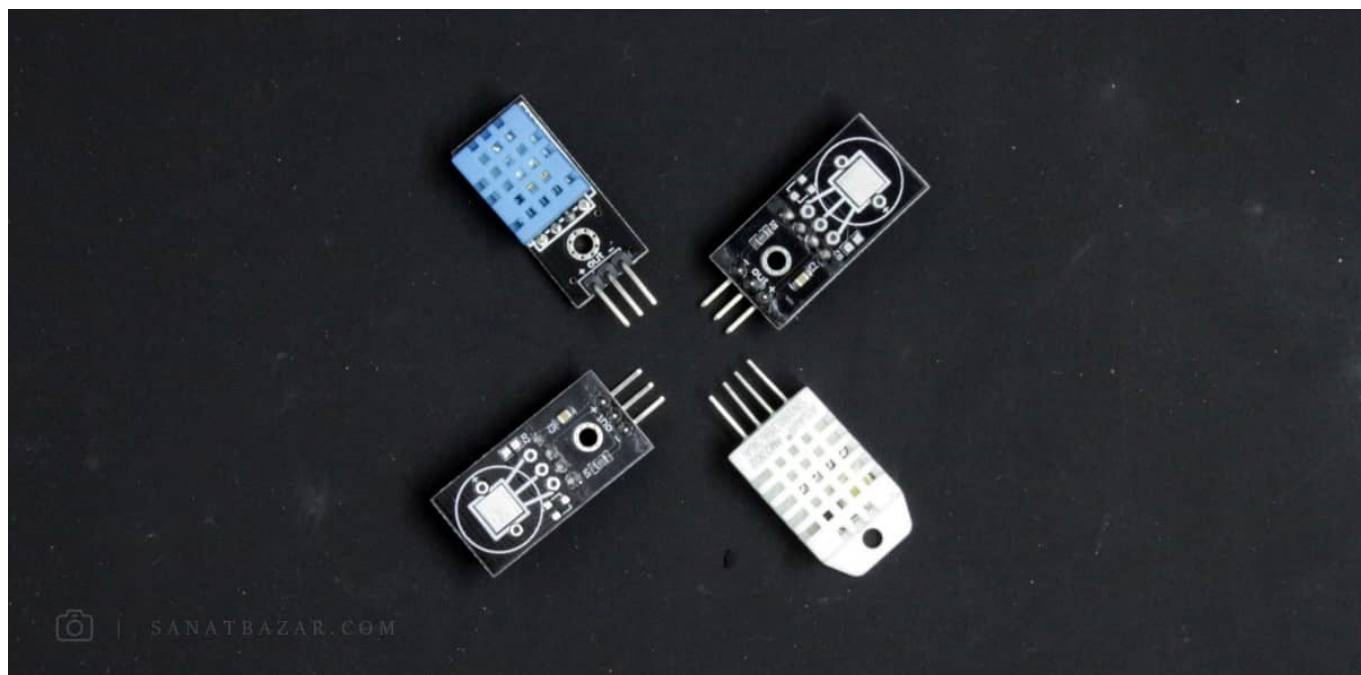
برد NodeMCU

مقاومت 10K

LED (دو عدد)

SD Card با حداقل حافظه 8GB

حسگر دما و رطوبت DHT: ساده و مناسب!



حسگرهای دما و رطوبت با توجه به قیمت مناسب و راهاندازی بسیار آسان، همیشه از سنسورهای مورد توجه علاقه‌مندان به پروژه‌های DIY (Do It Yourself) بوده‌اند. این قطعه توانایی اندازه‌گیری دما و رطوبت را به صورت همزمان داشته و به یکی از اجزای ضروری و مهم گلدان‌های هوشمند تبدیل شده است. در بین حسگرهای دما سنسورهای DHT11 و DHT22 نسبت به سایر مدل‌ها شناخته شده‌ترند و به همین دلیل در این بخش به آموزش راهاندازی این دو سنسور مازول می‌پردازیم. اگرچه هر دو ظاهر و پین‌های یکسانی دارند، اما مشخصات آن‌ها کمی با یکدیگر متفاوت است. در هر دو سنسور از سه بخش ترمیستور (مقاومت متغیر با دما)، لایه‌های حساس به رطوبت و مدار تبدیل سیگنال آنالوگ به دیجیتال استفاده شده است. در جدول زیر به بخشی از ویژگی‌های این دو قطعه اشاره می‌کنیم:

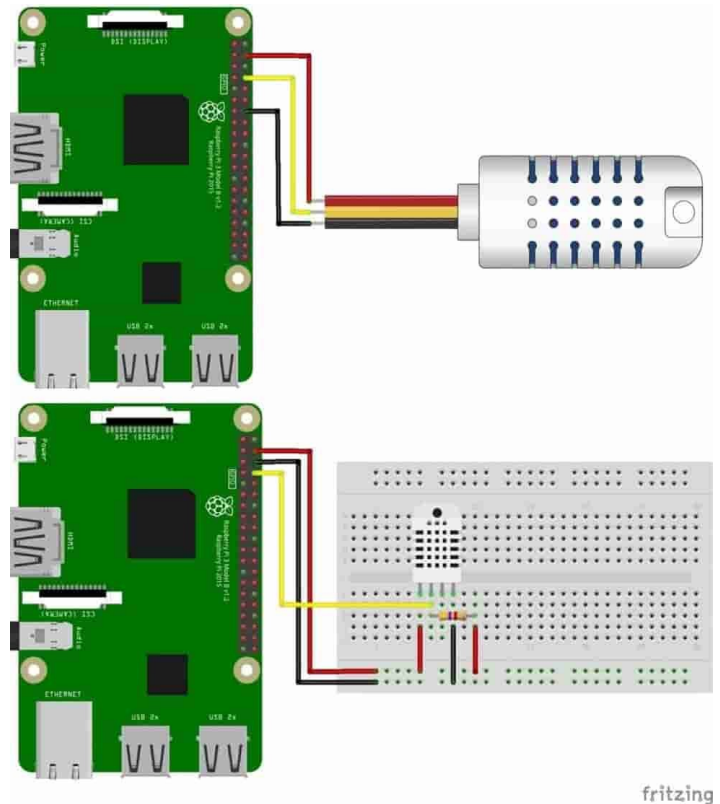
سنسور	DHT 11	DHT 22
ولتاژ ورودی و خروجی	۳ تا ۵ ولت	۳ تا ۵ ولت
حد اکثر جریان	۵.۲ میلی آمپر	۵.۲ میلی آمپر
میزان اندازه‌گیری رطوبت	۲۰٪ تا ۸۰٪ اندازه‌گیری با خطای ۵٪	۰٪ تا ۱۰۰٪ اندازه‌گیری با خطای ۲٪ تا ۵٪
میزان اندازه‌گیری دما	۰٪ تا ۵۰٪ اندازه‌گیری با خطای ۲٪	۰٪ تا ۴۰٪ اندازه‌گیری با خطای ۰.۵٪
نرخ نمونه‌برداری	۱ Hz (یک نمونه در هر ثانیه)	۵.۰ Hz (یک نمونه در هر دو ثانیه)
اندازه	۵.۱۵ mm * ۱۲ mm * ۵.۵ mm	۱.۱۵ mm * ۲۵ mm * ۷.۷ mm
تعداد پایه‌ها	۳ یا ۴ (دیجیتال)	۳ یا ۴ (دیجیتال)

همانطور که مشاهده می‌کنید، مدل DHT22 از دقت، نرخ و محدوده‌ی اندازه‌گیری بیشتری برخوردار است و شما می‌توانید متناسب با حساسیت و نیاز خود می‌توانید یکی از این سنسورها را انتخاب کنید. این دو سنسور هم به صورت ۴ پایه و هم به صورت ۳ پایه با برد الکترونیکی تولید می‌شوند. در ادامه نحوه‌ی راهاندازی آن‌ها را بررسی می‌کنیم.

برای راهاندازی حسگرهای ۴ پایه لازم است پایه‌ی Data را با مقاومت Pull Up کنید. در نوع ۳ پایه، این پایه به صورت داخلی Pull Up شده است.

راهاندازی سنسور دما و رطوبت DHT11 و DHT22 با رزبری پای

خب چون اینجا صفحه مربوط به رزبری پای است، پس آموزش را ابتدا با این برد شروع می‌کنیم. برای این کار فرقی نمی‌کند از DHT 11 استفاده کنید یا DHT 22. پس بدون معطلی قطعات را مانند شکل زیر به برد متصل کنید:



fritzing

- پایه‌ی مثبت حسگر را به پایه‌ی تغذیه‌ی رزبری پای (پین ۲)
- پایه‌ی منفی حسگر را به پایه‌ی GND رزبری پای (پین ۶)
- پایه‌ی data حسگر را به GPIO15 رزبری پای (پین ۱۴)

برای DHT با ۴ پایه، کافیسیت علاوه‌بر Pull Up کردن پایه‌ی data سنسور با مقاومت 10K Ω ، یک تغییر جزئی در کد پایتون خود ایجاد کنید. ابتدا مدار خود را به شکل زیر تغییر دهید. (پایه‌ی NC را لازم نیست به جایی وصل کنید).

- پایه‌ی مثبت حسگر را به پایه‌ی تغذیه‌ی رزبری پای (پین ۲)
- پایه‌ی منفی حسگر را به پایه‌ی GND رزبری پای (پین ۶)
- پایه‌ی data حسگر را با مقاومت Pull Up 10K Ω و به GPIO 14 (پین ۸)

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.

یکی از ویژگی‌های مثبت این سنسورها، وجود کتابخانه‌های آماده جهت استفاده و راهاندازی بسیار آسان آن‌ها است. در اینجا ما از کتابخانه‌ی Adatfruit که به زبان پایتون نوشته شده، استفاده می‌کنیم. برای این کار ابتدا رزبری پای خود را روشن و این کتابخانه را با اجرای دستور زیر در ترمینال لینوکس دانلود کنید:

```
$ pip3 install Adafruit_DHT
```

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

در قدم بعدی توسط دستور mkdir پوشه‌ای مناسب برای این پروژه ایجاد کنید. سپس با دستور nano یک فایل متنی با نام DHT11.py ساخته و کد پایتون زیر را در آن وارد کنید:

```
# SanatBazar
```

```
# Raspberry Pi Tutorial Series
# Author: Arvin Ghahremani
# Website: www.sanatzbazar.com

import RPi.GPIO as GPIO
import Adafruit_DHT as DHT
import time

data = 14
GPIO.setmode(GPIO.BCM)
GPIO.setup(data, GPIO.IN)

while True:
    try:
        humidity, temperature = DHT.read_retry(DHT.DHT11, data)
#        For DHT22:
#        humidity, temperature = DHT.read_retry(DHT.DHT22, data)

        print('Temperature = ', temperature)
        print('Humidity = ', humidity)
        time.sleep(5)
    except KeyboardInterrupt :
        break

GPIO.cleanup()
```

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

پس از ذخیره، کافیت با دستور زیر برنامه‌ی خود را اجرا کنید:

```
$ python3 DHT11.py
```

همانطور که مشاهده می‌کنید هر ۵ ثانیه، مقادیر دما و رطوبت نمایش داده می‌شود. برای متوقف کردن برنامه می‌توانید از کلید Ctrl+C استفاده کنید.

```

pi@SanatBazar: ~/projects/DHT_ESP8266_01
pi@SanatBazar:~/projects/DHT_ESP8266_01 $ python3 DHT11.py
Temperature = 30.0
Humidity = 15.0

Temperature = 31.0
Humidity = 14.0

^Cpi@SanatBazar:~/projects/DHT_ESP8266_01 $

```

 | SANATBAZAR.COM

خب این از رزبری پای. حالا فرض می‌کنیم که رزبری پای گیرنده و NodeMCU فرستنده است. پس در ادامه می‌خواهیم همین کارها را با برد NodeMCU هم انجام بدیم.

راه‌اندازی سنسور دما و رطوبت DHT11 و DHT22 با NodeMCU



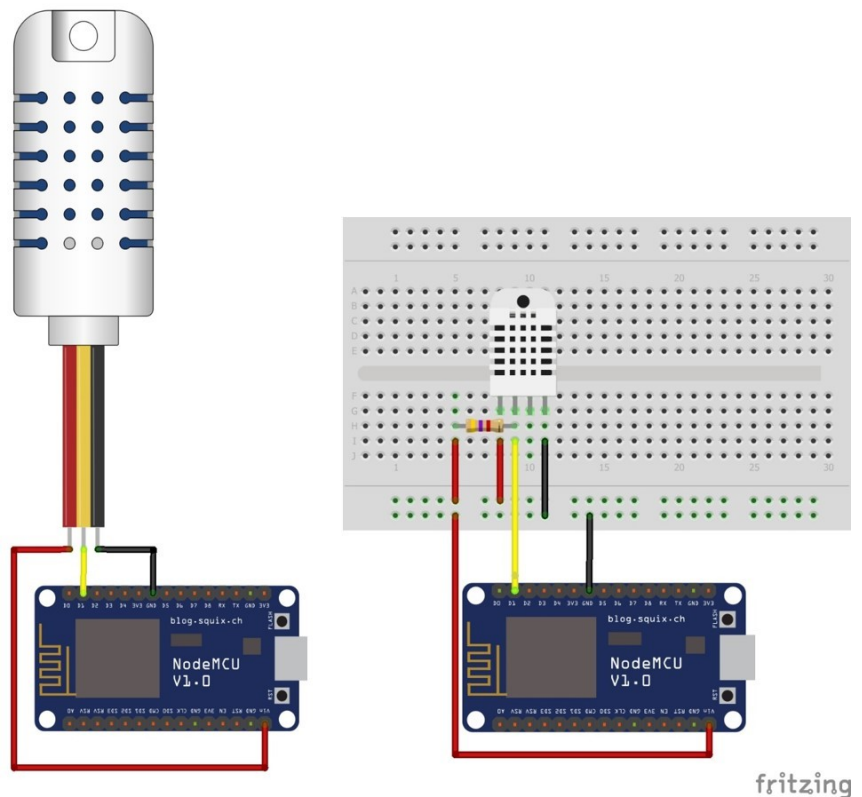
TOUT	ADC 0	A0	D0	GPIO 16	USER	WAKE
	GND	G	D1	GPIO 5		
	5 V	RSV	D2	GPIO 4		
SDD 3	GPIO 10	SD3	D3	GPIO 3	FLASH	
SDD 2	GPIO 9	SD2	D4	GPIO 2		
SDD 1	MOSI	SD1	3V3	3V3		
SDCMD	CS	CMD	GND	GND		
SDD 0	MISO	SD0	D5	GPIO 14	HSCLK	
SDCLK	SCLK	CLK	D6	GPIO 12	HMISO	
	GND	GND	D7	GPIO 13	HMOSI	RXD1
	3V3	3V3	D8	GPIO 15	HCS	TXD1
	EN	EN	D9	GPIO 3		RXD0
	RST	RST	D10	GPIO 1		TXD0
	GND	GND	GND	GND		
	5 V	VIN	3V3	3V3		

قبل از اینکه بریم سراغ سنسور و نصب و راه‌اندازی، اول ببینیم این NodeMCU چی هست اصلاً؟ اگر تا حالا با این برد کار نکردید، مطمئن باشید بعد از خواندن این مطلب، NodeMCU به یکی از گزینه‌های شما برای انجام پروژه‌های مختلف، به‌خصوص IoT تبدیل خواهد شد. این برد را می‌توان ترکیبی از آردوینو و تراشه‌های Esp8266 که برای ارتباط و انتقال بیسیم استفاده می‌شوند، دانست. NodeMCU از ۱۲ پایه ورودی-خروجی دیجیتال، یک پایه آنالوگ، رابط‌های ارتباط UART، I2C، SPI و ماژول Esp8266 برای ارتباط Wi-Fi برخوردار است. نکته‌ی خیلی مهم دیگر درباره‌ی این برد، قابلیت و شباهت کامل برنامه‌نویسی آن با آردوینو است. بنابراین اگر با Arduino IDE و نحوه‌ی برنامه‌نویسی در آن آشنا هستید، برای کار با NodeMCU نیاز به هیچ چیز دیگری ندارید. در اینجا هم چون قصد ما انتقال داده‌های بیسیم است، می‌خواهیم از این برد برای انتقال مقادیر دما و رطوبت به رزبری پای استفاده کنیم. قبل از آن اجازه بدید اول سنسور DHT را با این میکروکنترلر راه‌اندازی کنیم، سپس کتابخانه و پلتفرم لازم برای این انتقال را به شما معرفی می‌کنم.

برای مطالعه‌ی بیشتر درباره‌ی بردهای NodeMCU و Esp8266، می‌توانید به صفحه‌ی معرفی و راه‌اندازی انواع ماژول وایرلس با آردوینو مراجعه کنید.

پایه‌های ورودی خروجی این برد در شکل بالا نمایش داده شده است. دقت کنید که نام‌گذاری پایه‌ها روی برد با چیزی که در Arduino IDE می‌نویسید متفاوت است. مطابق شکل زیر، مدار اندازه‌گیری دما و رطوبت DHT11 و DHT22 را می‌بینیم:

پایه Vin در NodeMCU برای تغذیه برد توسط باتری در نظر گرفته شده است. اما از آنجایی که این پایه به صورت مستقیم به مدار تغذیه USB متصل است، در صورت تغذیه برد توسط کابل USB می‌توانید از این پین به عنوان خروجی 5V استفاده کنید. البته به استثنای نسخه NodeMCU V3!



fritzing

مطابق این شکل برای DHT با سه پایه:

- پایه منفی را به پین G (Ground)
- پایه مثبت را به پین VIN (5V)
- پایه Out را به پین D2 (GPIO 4)

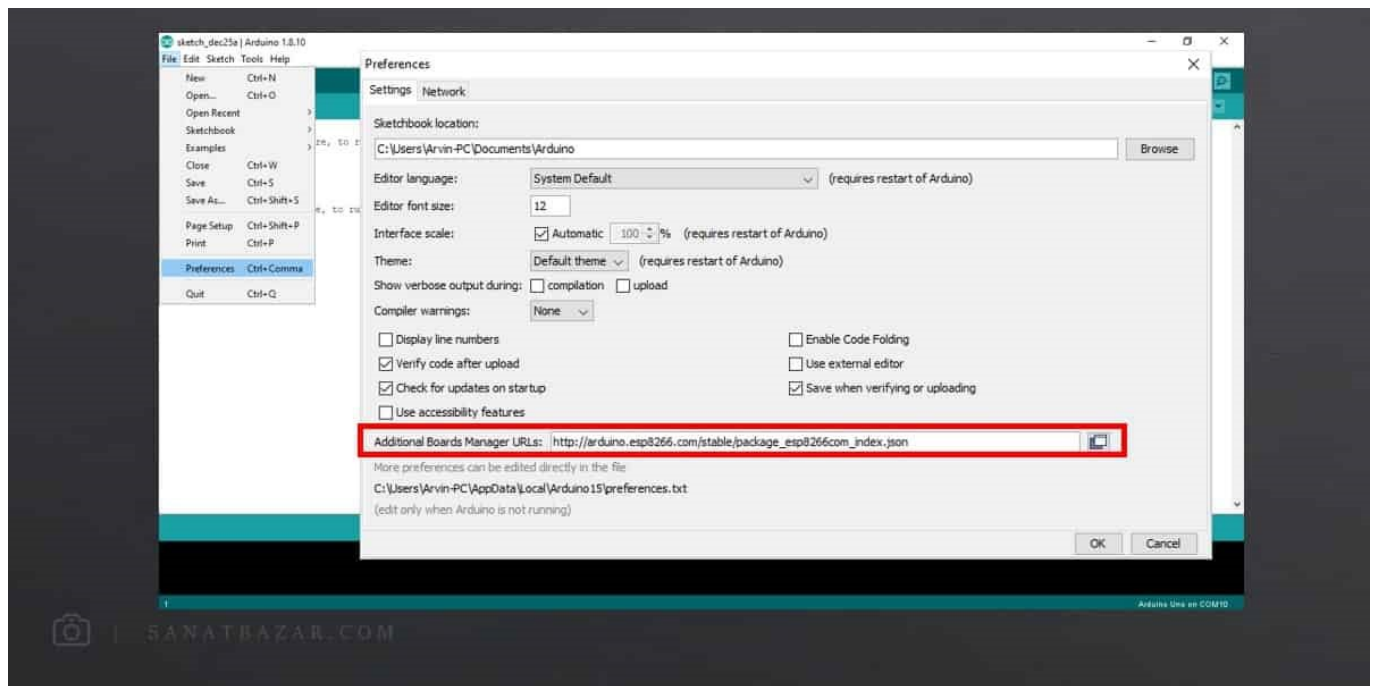
و برای DHT با چهار پایه:

- پایه منفی را به پین G (Ground)
- پایه مثبت را به پین VIN (5V)
- پایه data حسگر را با مقاومت 10K، Pull Up و به پین D2 (GPIO 4)

با توجه به این که جای برخی پایه‌ها در نسخه‌های مختلف NodeMCU متفاوت است، هنگام نصب مدار به نام پایه دقت کنید. در اینجا ما از NodeMCU Ver 0.1 استفاده می‌کنیم.

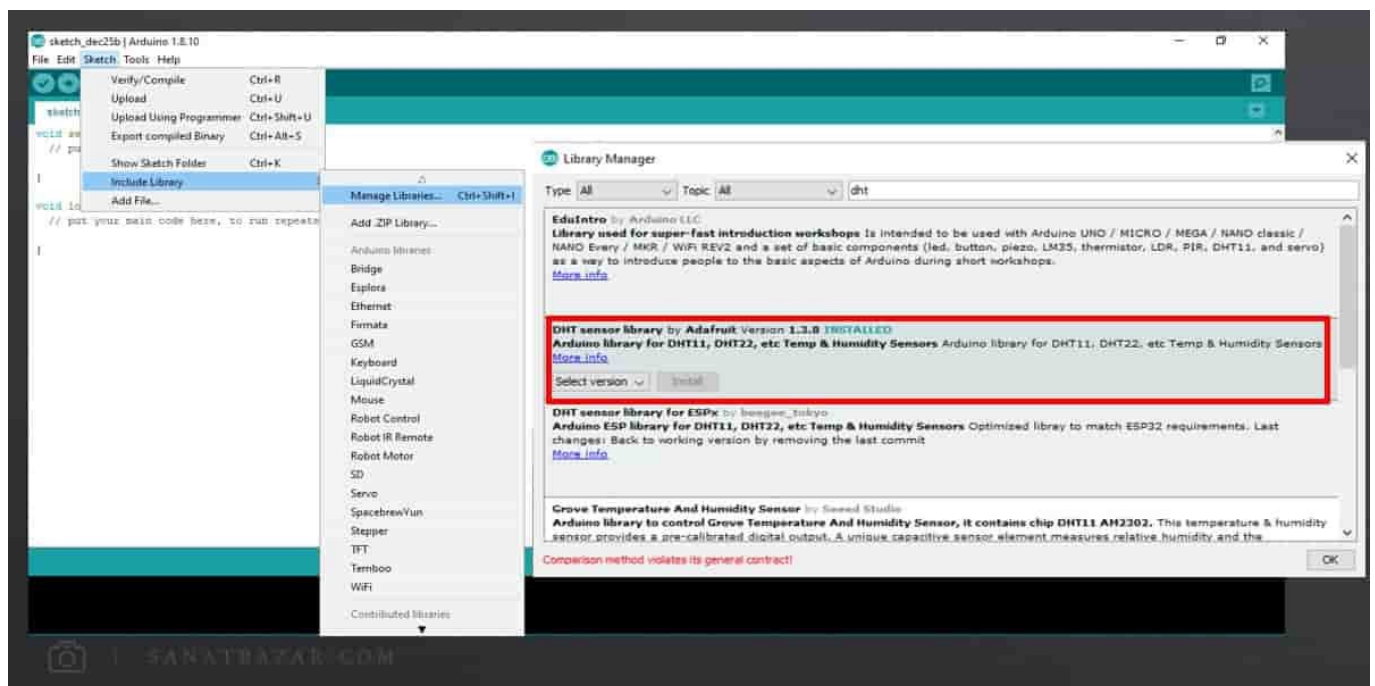
متصل می‌کنیم. پس از انجام این کار نوبت به تنظیمات و کدنویسی نرم‌افزاری می‌رسد. همانطور که گفتیم، برای Program کردن این برد از نرم‌افزار Arduino IDE استفاده می‌شود. اما برای شناسایی برد توسط نرم‌افزار و راه‌اندازی آن، لازم است تغییرات کوچکی را در IDE ایجاد و کتابخانه لازم برای کار با برد را نصب کنید. برای این کار از قسمت File، Preferences و انتخاب و مشابه تصویر، آدرس زیر را در بخش Additional Boards Manager URLs وارد کنید:

```
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```



تا اینجا برد NodeMCU را به بردهای برنامه اضافه کردید. در قدم بعدی ابتدا فایل کتابخانه esp8266 را از بخش پایین دانلود و از مسیر Sketch ▢ Add Library ▢ Manage Libraries به برنامه معرفی کنید. سپس در Sketch ▢ Add Library ▢ Manage Libraries به جستجو و نصب کنید.

فایل کتابخانه DHT



پس از این که همه چیز آماده شد، کد لازم برای راهاندازی سنسور را در برنامه می‌نویسیم.

```
#include <DHT.h>

#include <ESP8266WiFi.h>

#define data 4

DHT dht(data , DHT11);

void setup() {
```

```

Serial.begin(115200);

dht.begin();

}

void loop() {

    float h = dht.readHumidity();

    float t = dht.readTemperature();

    Serial.print("Temperature = ");

    Serial.print(t);

    Serial.print(" C");

    Serial.print(" ----- ");

    Serial.print("Humidity = ");

    Serial.print(h);

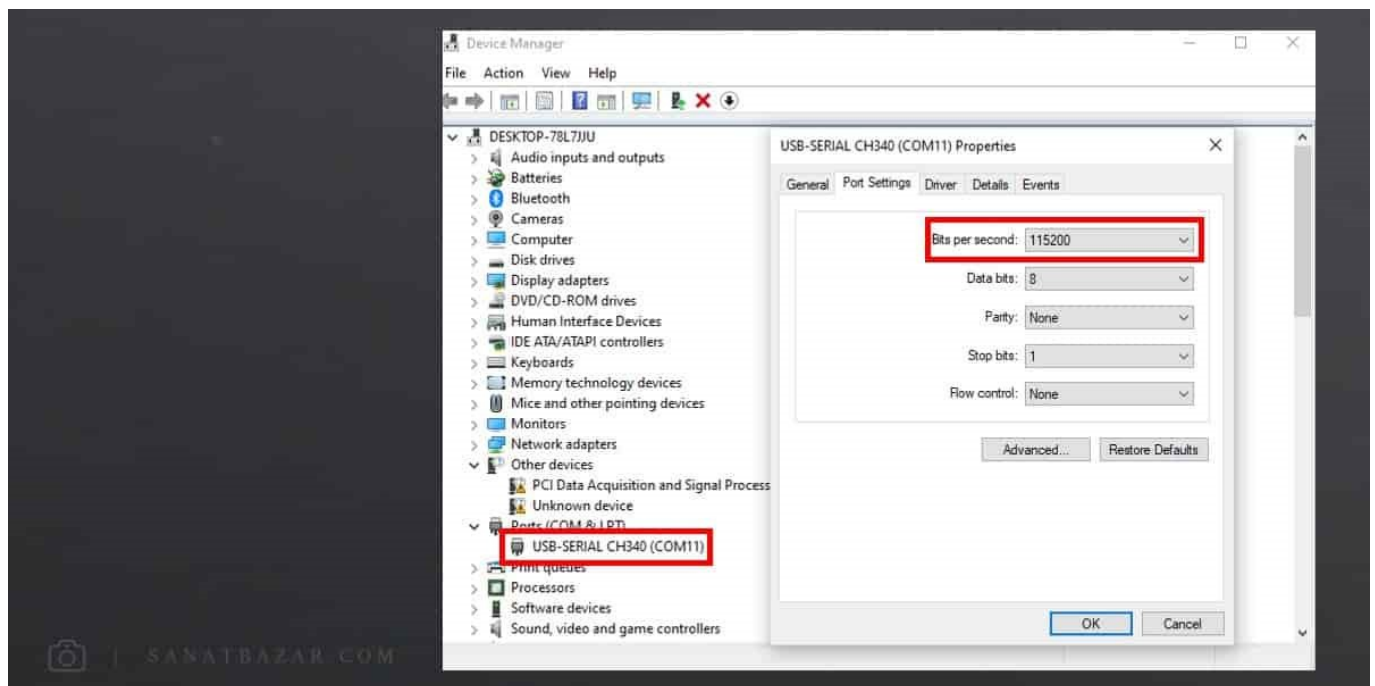
    Serial.println(" %");

    delay(2000);

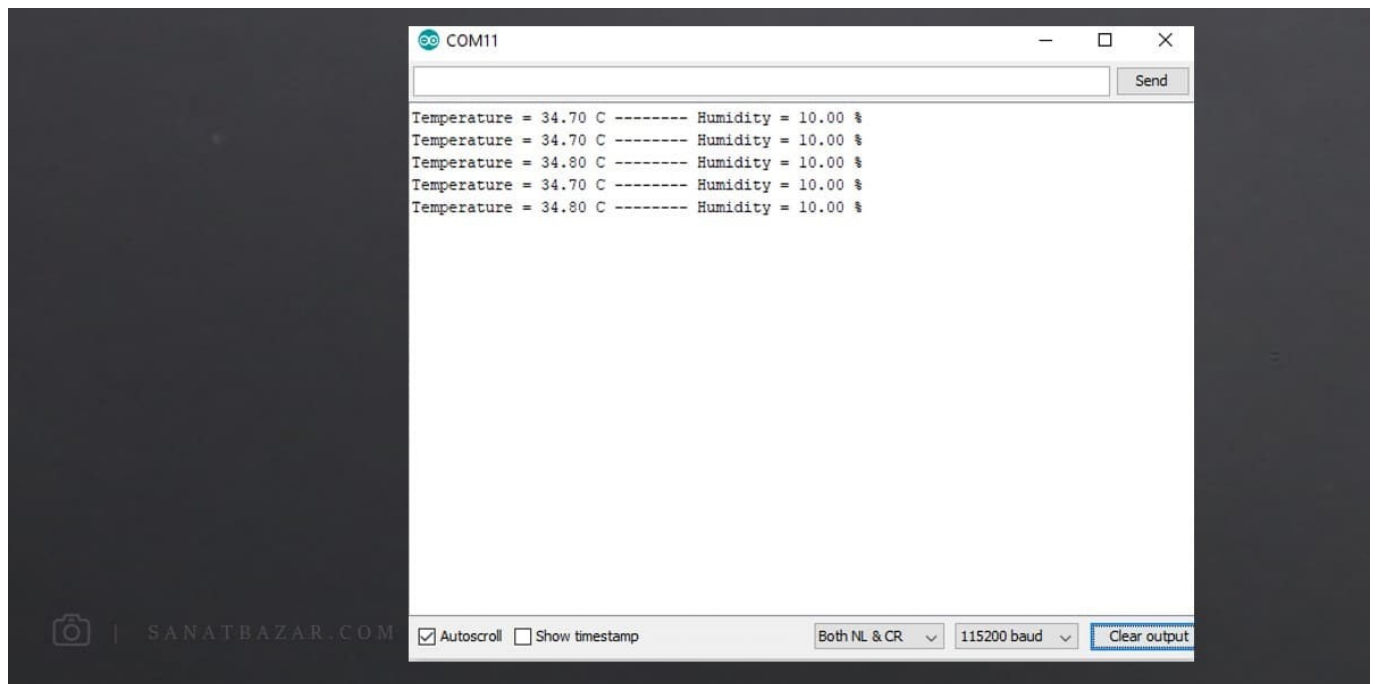
}

```

حالا NodeMCU را از طریق USB به کامپیوتر خود وصل کنید. پس از شناسایی برد، از در منوی Tools، Borad را انتخاب کرده و آن را روی Generic ESP8266 تنظیم کنید. سپس از بخش Search ویندوز، Device Manager را جست‌وجو کرده تا تنظیمات پورت اختصاص داده شده به برد را به انجام بدهیم. مطابق تصویر، این مقدار برای من COM11 در نظر گرفته شده است. روی آن کلیک کرده و از بخش Bit Per Second، Port Setting، 115200 تنظیم کنید.



در نهایت از منوی Tools نرم‌افزار Port، Arduino IDE، داده شده را انتخاب و برنامه را Upload کنید. نتایج را می‌توانید از طریق Serial Monitor مشاهده کنید. برای نمایش صحیح، آن را روی Baud 115200 قرار دهید:



MQTT | Mosquitte Telemetry Transport: چیست؟ چطور آن را نصب کنم؟



خب پس از این که با مقدمات کار آشنا شدیم، حالا می‌خواهیم یک پروژه‌ی جالب و آموزشی انجام بدیم. قصد داریم اطلاعات دما و رطوبت را با NodeMCU اندازه‌گیری و به صورت بیسیم به رزبری پای ارسال کنیم. اگر دما و رطوبت از حد معینی بیشتر بود، رزبری پای دستور روشن کردن LED های مربوط به آن‌ها را به NodeMCU ارسال می‌کند. این فقط یک پروژه‌ی آموزشی است و شما می‌توانید به جای این کار ساده، داده‌های سنسورهای مختلف را توسط رزبری پای دریافت کرده و پس از تحلیل آن‌ها، فرمان مناسب را به NodeMCU بفرستید. اما گفتیم می‌خواهیم این کار را با MQTT انجام بدیم. پس اول ببینیم MQTT چیست؟

MQTT یک پروتکل ارتباطی است که توسط آن چندین دستگاه می‌توانند به هم متصل شده و داده‌ها را انتقال دهند. اخیراً این پروتکل به دلیل سادگی و کاربرد مناسب، در پروژه‌های IoT مورد توجه قرار گرفته است. پس اگر قصد ورود به این حوزه دارید، جای درستی آمده‌اید. اما این پروتکل چطوری کار می‌کند؟ انتقال اطلاعات در بستر MQTT توسط سه بخش اصلی انجام می‌شود. Publisher، Subscriber و MQTT Broker. در واقع اطلاعات توسط Publisher (فرستنده) روی یک Topic (موضوع) به یک سیم تصور کنید البته این ارتباط بیسیمه ولی برای این که بفهمید نقش Topic چیست، فرض کنید یک سیم مجازی که اطلاعات رو از مبدا خاصی به مقصد خاصی می‌بره) ریخته شده و توسط MQTT Broker به چندین Subscriber (گیرنده) ارسال می‌شود. در اینجا ما می‌خواهیم اطلاعات سنسور DHT را توسط MQTT روی یک Topic به رزبری پای به عنوان Publish، MQTT Broker، Subscriber کنیم. پس برای این کار اول رزبری پای خود را روشن و MQTT را همراه با من روی آن نصب کنید. در

قدم دستورات زیر را برای نصب پکیج مورد نظر، وارد کنید:

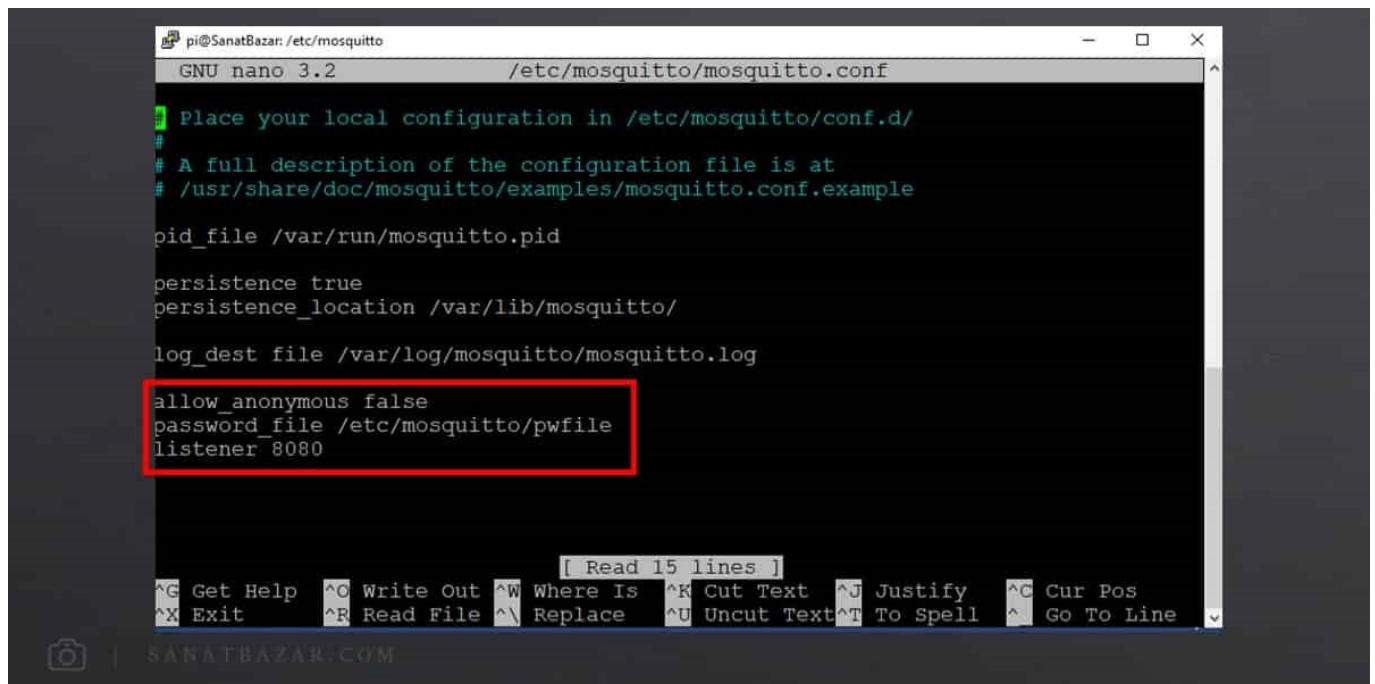
```
$ sudo apt-get update
$ sudo apt-get install mosquito -y
$ sudo apt-get install mosquito-clients -y
```

پس از انجام دستورات فوق برای پیکربندی پروتکل، فایل پیکربندی زیر را تغییر دهید:

```
$ sudo nano /etc/mosquitto/mosquitto.conf
```

در این فایل عبارت `include-dir /etc/mosquitto/conf.d` را پاک و دستورات زیر را جایگزین کنید. با این کار تعیین کردیم که از دسترسی افراد متفرقه، بدون ورود رمز و نام کاربری به اطلاعات Topic جلوگیری شود (اگر داشتن Username و Password برای شما مهم نیست می‌توانید موارد زیر را انجام ندهید:

```
allow_anonymous false
password_file /etc/mosquitto/pwfile
listener 1883
```



پورت ارتباطی به صورت پیش‌فرض 1883 است اما شما می‌توانید در صورت نیاز آن را تغییر دهید. برای تعیین Username و Password، دستور زیر را اجرا کنید. در این دستور به جای Username، نام مورد نظر خود را وارد کنید و پس از اجرای دستور Password از شما درخواست می‌شود. بنابراین پس از اجرا، رمز دلخواه خود را وارد کنید.

```
$ sudo mosquitto_passwd -c /etc/mosquitto/pwfile username
```

در نهایت، لازم است که برد خود را یک بار راه‌اندازی مجدد کنید. پس از این مرحله، برای این که مطمئن شویم همه‌ی کارها را به درستی انجام داده‌ایم، دو ترمینال را همزمان باز می‌کنیم (دو تا پنجره‌ی Putty را باز به رزبری پای متصل شوید). در پنجره‌ی اول دستور زیر را وارد کنید:

```
$ mosquitto_sub -d -u username -P password -t test
```

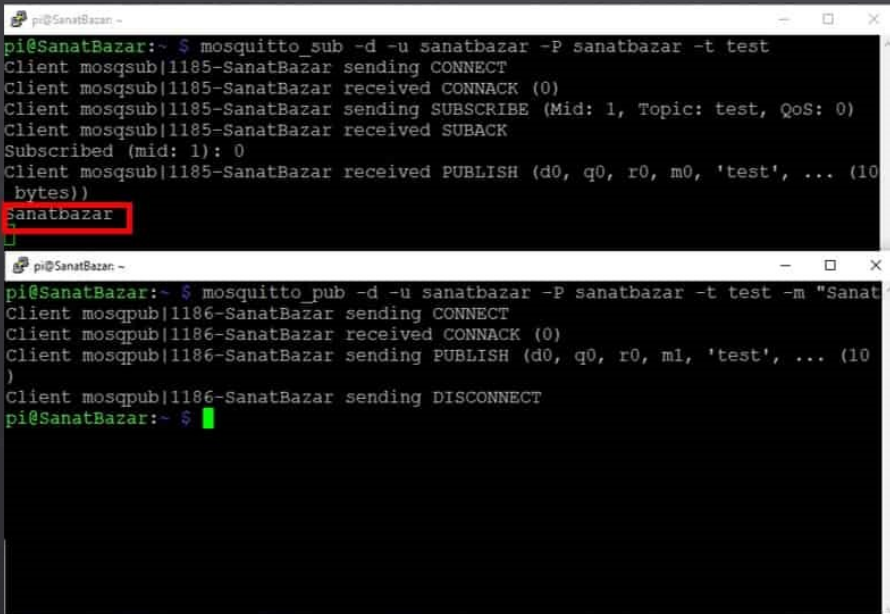
با این دستور یک Subscriber تعریف کردیم که پیام ارسال شده روی تاپیک test با نام کاربری و رمز مورد نظر ارسال را دریافت کند. (سویچ d جزئیات ارسال را

نمایش می‌دهد.) شما باید به جای username و password مقادیر خود را جایگزین کنید. اگر در ابتدا برای MQTT رمز و نام کاربری ایجاد نکردید، دستور زیر را به جای قبلی اجرا کنید:

```
$ mosquitto_sub -d -t test -m "Sanatbazar"
```

حالا در پنجره‌ی دیگر Publisher را به شکل زیر ایجاد کنید. با اجرای دستور باید Sanatbazar را در Subscriber مشاهده کنید:

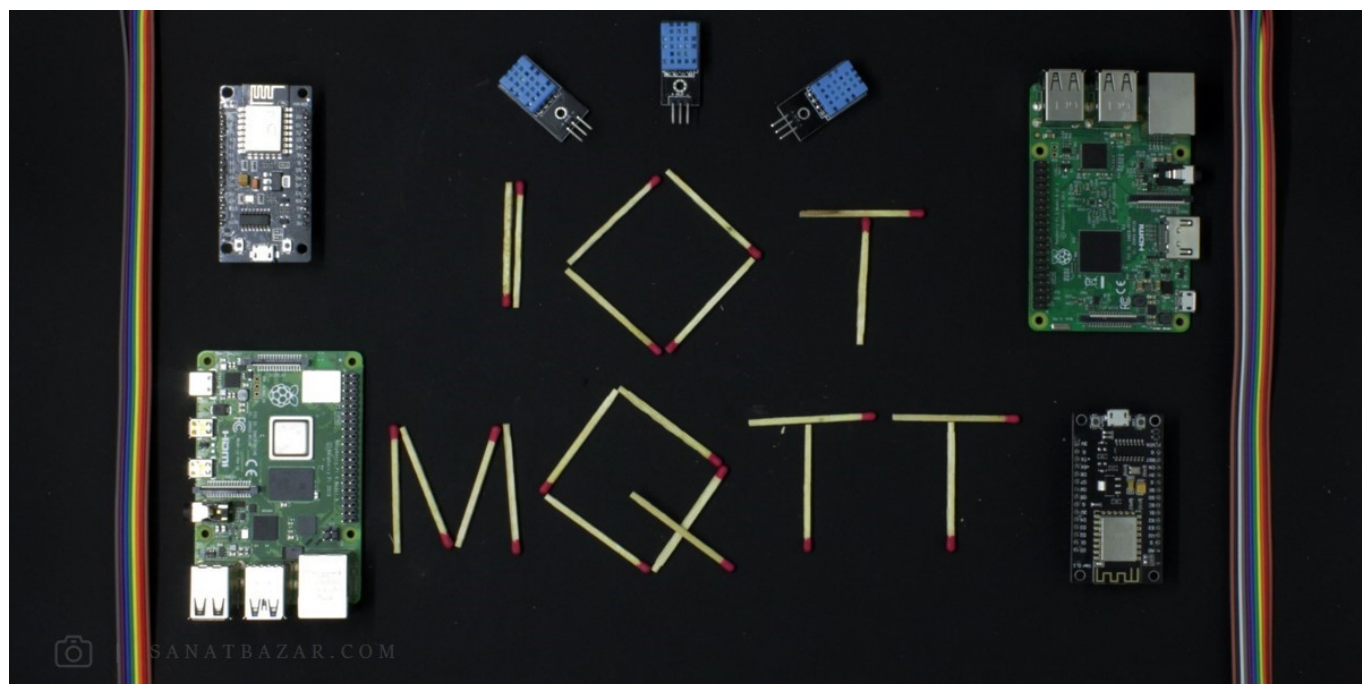
```
$ mosquitto_pub -d -u username -P password -t test -m "Sanatbazar"
```



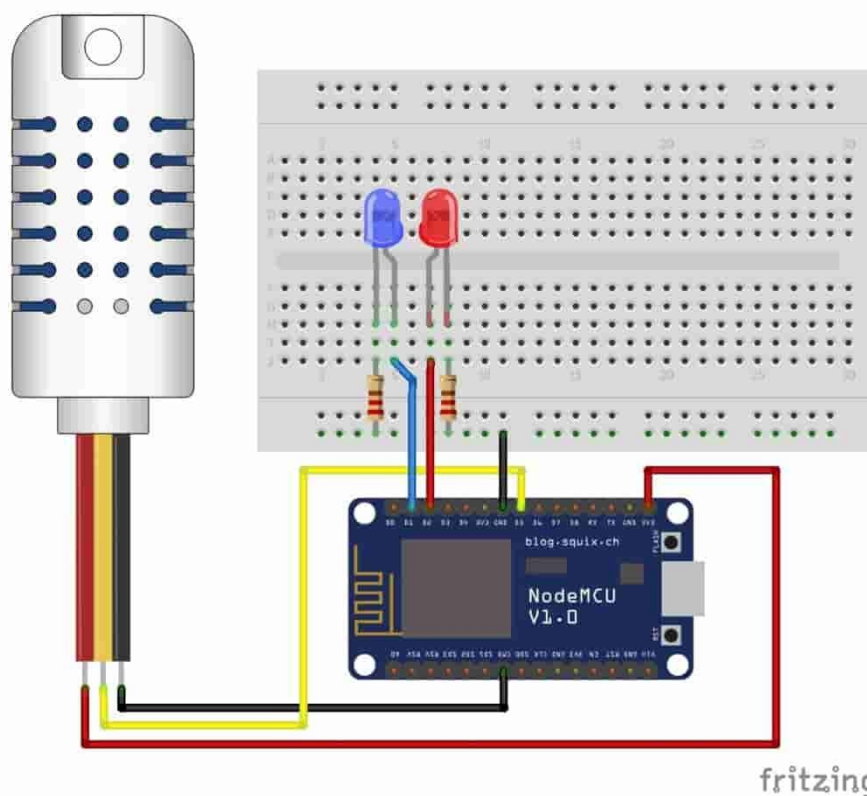
```
pi@SanatBazar:~$ mosquitto_sub -d -u sanatbazar -P sanatbazar -t test
Client mosqsub|1185-SanatBazar sending CONNECT
Client mosqsub|1185-SanatBazar received CONNACK (0)
Client mosqsub|1185-SanatBazar sending SUBSCRIBE (Mid: 1, Topic: test, QoS: 0)
Client mosqsub|1185-SanatBazar received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|1185-SanatBazar received PUBLISH (d0, q0, r0, m0, 'test', ... (10
bytes))
sanatbazar

pi@SanatBazar:~$ mosquitto_pub -d -u sanatbazar -P sanatbazar -t test -m "Sanat
Client mosqpub|1186-SanatBazar sending CONNECT
Client mosqpub|1186-SanatBazar received CONNACK (0)
Client mosqpub|1186-SanatBazar sending PUBLISH (d0, q0, r0, m1, 'test', ... (10
)
Client mosqpub|1186-SanatBazar sending DISCONNECT
pi@SanatBazar:~$
```

آموزش انتقال داده‌های رطوبت و دما از NodeMCU به رزبری پای (سیستم تهویه‌ی هوشمند)



خب تا اینجا MQTT را روی رزبری پای نصب و امتحان کردیم. حالا می‌خواهیم با استفاده از این پروتکل، یک پروژه‌ی کاربردی IoT انجام بدیم! من اسمشو گذاشتم سیستم تهویه‌ی هوشمند! چرا؟ چون قراره اطلاعات رطوبت و دما از یک نقطه توسط NodeMCU جمع‌آوری و به رزبری پای ارسال شود. سپس رزبری پای با توجه به این مقادیر، دستور روشن و خاموش کردن LED های متصل به NodeMCU را ارسال می‌کند. شاید بگید عجب سیستم تهویه‌ی مسخره‌ای!! ولی انقدر هم بد نیست. شما با توجه به امکانات و توانایی‌های خودتون می‌توانید داده‌ها را در یک پایگاه داده ذخیره و به جای LED از رله‌های متصل به یک سیستم تهویه استفاده کنید. کد انتقال داده آماده شده و فقط کافی‌ه هرکس با توجه به نیاز خودش کمی آن را تغییر دهد. نگران نباشید در ادامه به شما می‌گم کجاها را باید عوض کنید! در این پروژه ما رزبری پای را هم Broker و هم Subscriber تعریف می‌کنیم اما شما می‌توانید با توجه به نیاز خود، رزبری پای را سرور و چندین NodeMCU دیگر را به‌صورت End point (فرستنده یا گیرنده) تنظیم کنید. حالا که سرور ما آماده شده، ابتدا مدار را مطابق شکل زیر وصل و سپس کد NodeMCU را به شکل زیر می‌نویسیم:



من در اینجا:

- LED قرمز را به پین D1 (GPIO 4)

- LED آبی را به پایه‌ی (GPIO 5 D2)
- پین Data سنسور DHT11 را به (GPIO 14 D5)
- زمین مدار را به G
- VCC مدار را به 3V

وصل می‌کنم. برای کاهش جریان عبوری از LED، پایه منفی آن‌ها را توسط مقاومت 220Ω به زمین مدار وصل کنید.

حالا بریم سراغ کدنویسی! در ابتدا کتابخانه و مشخصات Broker را معرفی می‌کنیم:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
```

در این بخش متناسب با نوع سنسوری که استفاده می‌کنید، دستور مناسب را بنویسید:

```
#define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
```

در این قسمت Username و Password که در بخش ساخت Broker معرفی کردید را وارد نمایید:

```
#define username "*****"
#define pass "*****"
```

در این بخش هم مشخصات مودم و روتر خود را به درستی وارد کنید:

```
const char* ssid = "*****";
const char* password = "*****";
```

برای برقراری ارتباط MQTT لازم است IP دستگاه Broker به تمامی Node ها معرفی شود:

```
// Change the variable to your Raspberry Pi IP address, so it connects to your MQTT broker
const char* mqtt_server = "192.168.0.2";
```

در این مرحله، پایه‌های ورودی خروجی مورد نظر خود را تعیین کنید.

```
const int ledGPIO5 = 5;
const int ledGPIO4 = 4;

String data;

// DHT Sensor
const int DHTPin = 14;
// Initialize DHT sensor.
DHT dht(DHTPin, DHTTYPE);

// Timers auxiliar variables
long now = millis();
```

```
long lastMeasure = 0;
```

در ادامه دستورات لازم برای اتصال NodeMCU به روتر را می‌نویسیم:

```
void setup_wifi() {
    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(".");
    }
    Serial.println("");
    Serial.print("WiFi connected - ESP IP address: ");
    Serial.println(WiFi.localIP());
}
```

خب کم‌کم رسیدیم به بخش تخصصی کار. برای خواندن مقادیر Subscribe شده از تابع callback استفاده می‌شود. این بخش را بدون اعمال تغییر در کد خود کپی کنید:

```
void callback(String topic, byte* message, unsigned int length) {
    Serial.print("Message arrived on topic: ");
    Serial.print(topic);
    Serial.print(". Message: ");
    String messageTemp;
    : نمایش بدیم Serial Monitor حالا می‌خواهیم مقدار دریافت شده را در
    for (int i = 0; i < length; i++) {
        Serial.print((char)message[i]);
        messageTemp += (char)message[i];
    }
    Serial.println();
}
```

در اینجا تعیین می‌کنم که با توجه به مقدار خوانده شده از هر Topic، چه اقدامی صورت بگیرد؟ من می‌خواستم با توجه به مقدار دما و رطوبت دو تا LED را خاموش و روشن کنم. اما این دستور از سمت سرور یعنی رزبری پای قرار بود ارسال شود که در ادامه کد آن قسمت هم بررسی می‌کنیم. پس مقدار روشن یا خاموش شدن را روی هر Topic مخصوص می‌فرستم، در NodeMCU می‌خوانم و با توجه به آن LED را روشن یا خاموش می‌کنم. من این مقادیر را صفر و یک در نظر گرفتم، شما می‌توانید هر چیز دیگری مثلاً ON و OFF در نظر بگیرید. این فقط یک پروژه‌ی آموزشی است! به جای روشن و خاموش کردن LED می‌توانید یک رله یا مازول خاص را راه‌اندازی کنید. کافیت کد آن را در همین قسمت جایگزین LED ها کنید. نام Topic مربوط به LED اول /esp8266/4 و نام Topic برای LED دوم /esp8266/5 است.

```
if (topic == "/esp8266/4") {
    Serial.print("Changing GPIO 4 to ");
}
```

```

        if(messageTemp == "1"){
            digitalWrite(ledGPIO4, HIGH);
            Serial.print("On");
        }
        else if(messageTemp == "0"){
            digitalWrite(ledGPIO4, LOW);
            Serial.print("Off");
        }
    }
    {
        if(topic=="/esp8266/5"){
            Serial.print("Changing GPIO 5 to ");
            if(messageTemp == "1"){
                digitalWrite(ledGPIO5, HIGH);
                Serial.print("On");
            }
            else if(messageTemp == "0"){
                digitalWrite(ledGPIO5, LOW);
                Serial.print("Off");
            }
        }
        {
            Serial.println();
        }
    }

```

در اینجا تا زمانی که ارتباط MQTT برقرار نشده باشد (به هر دلیلی!) برنامه سعی می‌کند تا زمان برقراری، ارتباط را reconnect کند:

```

void reconnect(){
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect("ESP8266Client")) {
            Serial.println("connected");
            client.subscribe("/esp8266/4");
            client.subscribe("/esp8266/5");
        }
        else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}

```

}

در ادامه هم تنظیمات setup شامل ورودی خروجی کردن و Baudrate ارتباط را تعیین می‌کنیم:

```
void setup(){
    dht.begin();

    pinMode(ledGPIO4, OUTPUT);
    pinMode(ledGPIO5, OUTPUT);

    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

MQTT: برقراری ارتباط
void loop(){
    if (!client.connected()) {
        reconnect();
    }

    if(!client.loop())
        client.connect("ESP8266Client",username,pass);
}
```

در نهایت مقادیر دما و رطوبت را خوانده، به Char تبدیل و به سرور روی Topic های مشخص (/temperature/esp8266 برای دما و /humidity/esp8266 برای رطوبت) ارسال می‌کنیم. این کار را هر ۱۰ ثانیه یکبار انجام می‌دهیم:

```
now = millis();

if (now - lastMeasure > 10000) {
    lastMeasure = now;
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    if (isnan(h) || isnan(t)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    static char temperature[7];
    dtostrf(t, 3, 2, temperature);

    static char humidity[7];
    dtostrf(h, 3, 2, humidity);

    client.publish("/esp8266/temperature", temperature);
}
```

```

        client.publish("/esp8266/humidity", humidity);

        data="Temprature :";
        data+=temperature;
        data+=" ";
        data+="*C";
        data+=" ";
        data+="Humidity :";
        data+=humidity;
        data+=" ";
        data+="%";
        Serial.println(data);
    }
}

```

کد نهایی به صورت زیر خواهد بود:

```

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

// Uncomment one of the lines bellow for whatever DHT sensor type you're using!
#define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

#define username "*****"
#define pass "*****"

// Change the credentials below, so your ESP8266 connects to your router
const char* ssid = "*****";
const char* password = "*****";

// Change the variable to your Raspberry Pi IP address, so it connects to your MQTT broker
const char* mqtt_server = "192.168.0.2";

// Initializes the espClient
WiFiClient espClient;
PubSubClient client(espClient);

// Connect an LED to each GPIO of your ESP8266
const int ledGPIO5 = 5;
const int ledGPIO4 = 4;

String data;

```

```
// DHT Sensor
const int DHTPin = 14;

// Initialize DHT sensor.
DHT dht(DHTPin, DHTTYPE);

// Timers auxiliar variables
long now = millis();
long lastMeasure = 0;

// Don't change the function below. This functions connects your ESP8266 to your
router
void setup_wifi() {
    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(".");
    }
    Serial.println("");
    Serial.print("WiFi connected - ESP IP address: ");
    Serial.println(WiFi.localIP());
}

// This functions is executed when some device publishes a message to a topic that
your ESP8266 is subscribed to
void callback(String topic, byte* message, unsigned int length) {
    Serial.print("Message arrived on topic: ");
    Serial.print(topic);
    Serial.print(". Message: ");
    String messageTemp;

    for (int i = 0; i < length; i++) {
        Serial.print((char)message[i]);
        messageTemp += (char)message[i];
    }
    Serial.println();
}
```

```
// Feel free to add more if statements to control more GPIOs with MQTT

if(topic=="/esp8266/4"){
    Serial.print("Changing GPIO 4 to ");
    if(messageTemp == "1"){
        digitalWrite(ledGPIO4, HIGH);
        Serial.print("On");
    }
    else if(messageTemp == "0"){
        digitalWrite(ledGPIO4, LOW);
        Serial.print("Off");
    }
}

if(topic=="/esp8266/5"){
    Serial.print("Changing GPIO 5 to ");
    if(messageTemp == "1"){
        digitalWrite(ledGPIO5, HIGH);
        Serial.print("On");
    }
    else if(messageTemp == "0"){
        digitalWrite(ledGPIO5, LOW);
        Serial.print("Off");
    }
}

Serial.println();
}

// This functions reconnects your ESP8266 to your MQTT broker
// Change the function below if you want to subscribe to more topics with your
ESP8266

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect("ESP8266Client")) {
            Serial.println("connected");
            // Subscribe or resubscribe to a topic
            // You can subscribe to more topics (to control more LEDs in this example)
            client.subscribe("/esp8266/4");
            client.subscribe("/esp8266/5");
        } else {

```

```
        Serial.print("failed, rc=");
        Serial.print(client.state());
        Serial.println(" try again in 5 seconds");
        // Wait 5 seconds before retrying
        delay(5000);
    }
}

// The setup function sets your ESP GPIOs to Outputs, starts the serial communication
// at a baud rate of 115200
// Sets your mqtt broker and sets the callback function
// The callback function is what receives messages and actually controls the LEDs
void setup() {
    dht.begin();
    pinMode(ledGPIO4, OUTPUT);
    pinMode(ledGPIO5, OUTPUT);

    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

// For this project, you don't need to change anything in the loop function.
// Basically it ensures that you ESP is connected to your broker
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    if(!client.loop())
        client.connect("ESP8266Client",username,pass);

    now = millis();
    // Publishes new temperature and humidity every 10 seconds
    if (now - lastMeasure > 10000) {
        lastMeasure = now;
        // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
        float h = dht.readHumidity();
        // Read temperature as Celsius (the default)
        float t = dht.readTemperature();
    }
}
```

```
// Check if any reads failed and exit early (to try again).
if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
}

static char temperature[7];
dtostrf(t, 3, 2, temperature);

static char humidity[7];
dtostrf(h, 3, 2, humidity);

// Publishes Temperature and Humidity values
client.publish("/esp8266/temperature", temperature);
client.publish("/esp8266/humidity", humidity);

data="Temprature :";
data+=temperature;
data+=" ";
data+="*C";
data+=" ";
data+="Humidity :";
data+=humidity;
data+=" ";
data+="%";
Serial.println(data);
}
}
```

خب این از NodeMCU! بریم سراغ رزبری پای. قبل از هر چیز با استفاده از دستور nano یک فایل برای ذخیره‌ی کدهایتان ایجاد کنید. من نام این فایل را mqtt.py انتخاب می‌کنم. سپس کد زیر را در این فایل کپی کنید:

مشابه برنامه‌ی قبلی، در ابتدا کتابخانه‌ها و متغیرهای برنامه را تعریف می‌کنیم. در اینجا user و passwd، همان username و password تعریف شده برای Broker هستند. با توجه به این که رزبری پای از سیستم‌عامل برخوردار است، خودتان به صورت دستی برد را به روتر مشترک با NodeMCU وصل کنید.

```
import paho.mqtt.client as mqtt
import time

temp=0
hum=0

user="*****"

passwd="*****"
```

در ادامه دو تابع `on_message` و `on_connect` را تعریف و در آن وظایفی که باید پس از اتصال و دریافت پیام‌های NodeMCU از Topic های تعریف شده انجام شود را تعیین می‌کنیم:

```
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

    client.subscribe("/esp8266/temperature")
    client.subscribe("/esp8266/humidity")

def on_message(client, userdata, message):
    print("Received message '" + " ' on topic '"
          + message.topic + "' with QoS " + str(message.qos))

    if message.topic == "/esp8266/temperature":
        global temp
        temp=float(message.payload)

    if message.topic == "/esp8266/humidity":
        global hum
        hum=float(message.payload)
```

همانطور که مشاهده کردید، مقادیر دما و رطوبت در دو متغیر `temp` و `hum` قرار داده می‌شود. همچنین با توجه به این که پیام‌ها به صورت کارکتری ارسال می‌شوند، برای انجام عملیات و تحلیل آن‌ها در ادامه، این مقادیر را به `Float` تبدیل کردیم. سپس دستورات لازم برای برقراری اتصال رزبری پای به Broker (که خودشه!) را می‌نویسیم:

```
mqttc=mqtt.Client()
mqttc.username_pw_set(username=user,password=passwd)
mqttc.on_connect = on_connect
mqttc.on_message = on_message
mqttc.connect("localhost",1883,60)
mqttc.loop_start()

ها LED ،تعیین کردیم که اگر دما و رطوبت از حدی بیشتر باشند While در نهایت، درون حلقه‌ی
ها خاموش شود. شما می‌توانید متناسب با نیاز خود، دستورات LED ،روشن و اگر کمتر باشند
بیشتری را به این بخش اضافه کنید:
```

```
while(True):
    global temp
    print("temprature= ",temp)
    print("humidity= ",hum)

    if temp>30:
        print('Its Hot! Turn the LED ON')
        (rc, mid) = mqttc.publish("/esp8266/4", '1',qos=1)
    else:
        print('Its Cold! Turn the LED OFF')
        (rc, mid) = mqttc.publish("/esp8266/4", '0',qos=1)
```

```

global hum

if hum>20:

    print('Humidity is High! Turn the LED ON')

    (rc, mid) = mqttc.publish("/esp8266/5", '1', qos=1)

else:

    print('Humidity is Low! Turn the LED OFF')

    (rc, mid) = mqttc.publish("/esp8266/5", '0', qos=1)

time.sleep(10)

```

مقدار qos سه حالت زیر را برای ارسال داده ایجاد می‌کند:

qos=0: داده‌ها بدون تضمین دریافت در سمت گیرنده، پشت سر هم ارسال شوند.

qos=1: داده‌ی بعدی پس از تایید شدن دریافت داده‌ی قبلی از طرف Broker، ارسال می‌شود.

qos=2: داده‌ی بعدی پس از تایید شدن دریافت داده‌ی قبلی از طرف Subscriber، ارسال می‌شود.

کد نهایی به شکل زیر خواهد بود:

```

import paho.mqtt.client as mqtt
import time

temp=0
hum=0
user="*****"
passwd="*****"

# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):

    print("Connected with result code "+str(rc))

    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.

    client.subscribe("/esp8266/temperature")

    client.subscribe("/esp8266/humidity")

# The callback for when a PUBLISH message is received from the ESP8266.
def on_message(client, userdata, message):

    print("Received message '" + str(message.topic) + "' on topic '"
          + message.topic + "' with QoS " + str(message.qos))

    if message.topic == "/esp8266/temperature":

        global temp

        temp=float(message.payload)

    if message.topic == "/esp8266/humidity":

        global hum

        hum=float(message.payload)

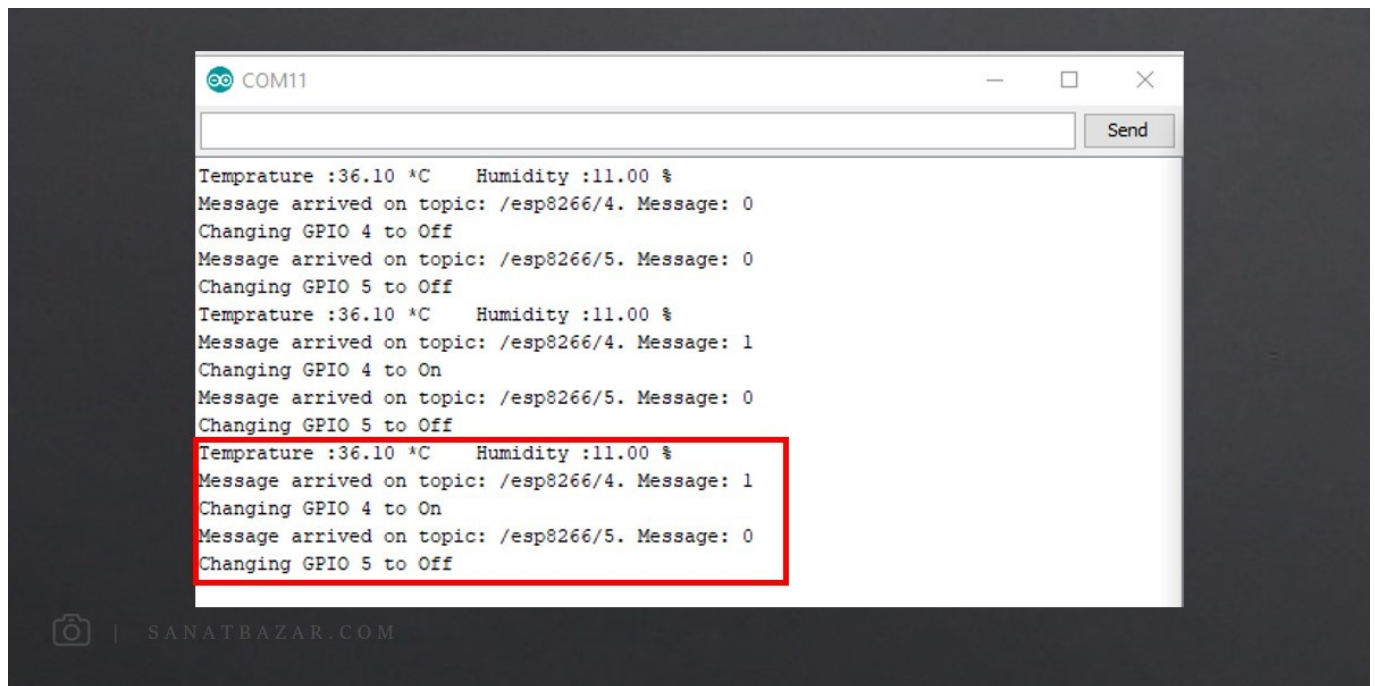
```

```
mqttc=mqtt.Client()
mqttc.username_pw_set(username=user,password=passwd)
mqttc.on_connect = on_connect
mqttc.on_message = on_message
mqttc.connect("localhost",1883,60)
mqttc.loop_start()

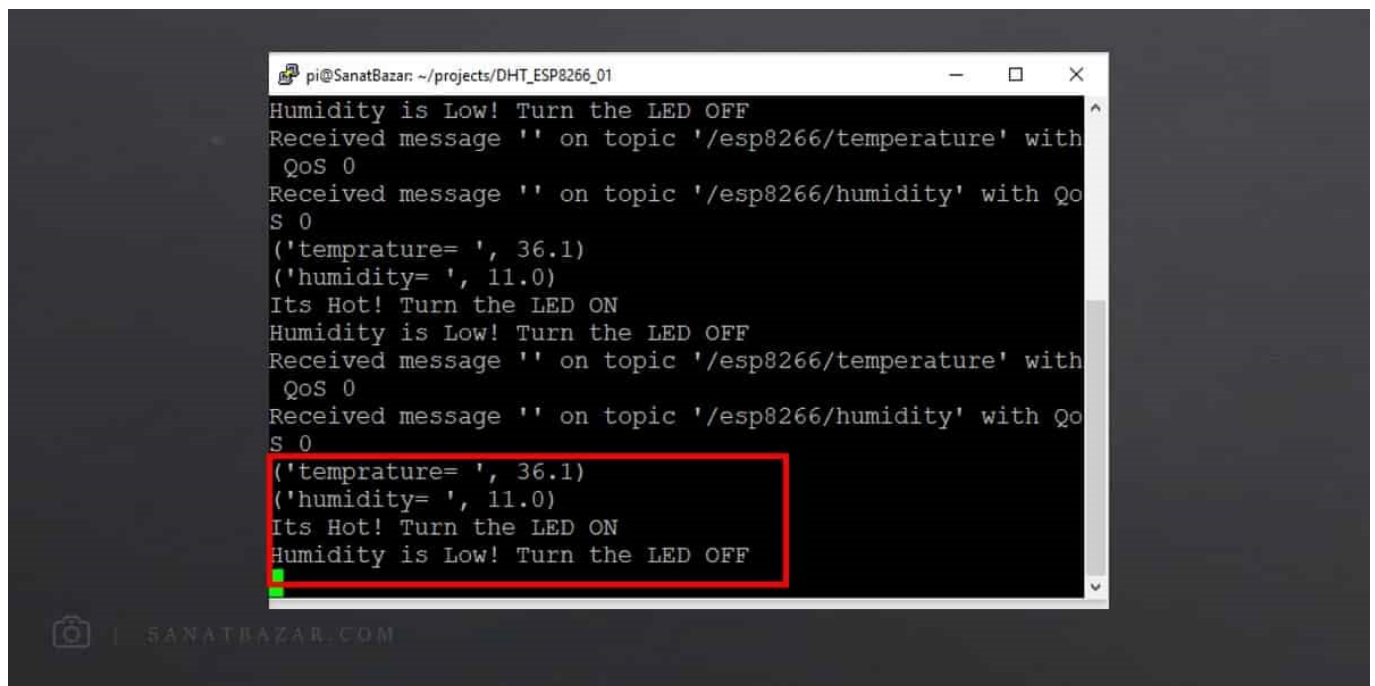
while(True):
    global temp
    print("temprature= ",temp)
    print("humidity= ",hum)
    if temp>30:
        print('Its Hot! Turn the LED ON')
        (rc, mid) = mqttc.publish("/esp8266/4", '1',qos=1)
    else:
        print('Its Cold! Turn the LED OFF')
        (rc, mid) = mqttc.publish("/esp8266/4", '0',qos=1)

    global hum
    if hum>20:
        print('Humidity is High! Turn the LED ON')
        (rc, mid) = mqttc.publish("/esp8266/5", '1',qos=1)
    else:
        print('Humidity is Low! Turn the LED OFF')
        (rc, mid) = mqttc.publish("/esp8266/5", '0',qos=1)
    time.sleep(10)
```

پس از ذخیره و اجرای برنامه‌ها در NodeMCU و رزبری پای، نتایج زیر را مشاهده خواهید کرد. در سمت NodeMCU:



و در بخش رزبری پای:



نتیجه گیری

در این قسمت از آموزش رزبری پای و IoT، اطلاعات سنسور دما و رطوبت DHT را توسط NodeMCU اندازه گیری و توسط MQTT به رزبری پای فرستادیم. از طرف دیگر، در رزبری پای پس از مشاهده داده ها، فرمان مناسب برای فعال شدن عملگرها را به طور مشابه با MQTT به NodeMCU ارسال کردیم. اگر توجه کرده باشید، این ارتباط به صورت داخلی برقرار بود. یعنی در صورتی که رزبری پای و NodeMCU به یک روتر مشترک متصل نباشند (از هم دور باشند) امکان پیاده سازی این پروژه وجود ندارد. بنابراین با من همراه باشید تا در قسمت بعدی این انتقال داده را در قالب یک پروژه ی جدید و از طریق اینترنت و ThingSpeak قدم به قدم انجام بدیم. با این کار می توانید داده ها را از هرجای دنیا با اینترنت ارسال و از جای دیگر دریافت کنید!

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.