

# گزارش کار پروژه اینترنت اشیا

استاد : اصغر کریم پور گمش آبادی.

گروه دانشجویان :

1. یاسین ابراهیم نژادیان

2. امید اعظامی

3. آرش حسین پور

موضوع: پروژه شماره 1 کنترل سرعت فن بر اساس مقدار سنسور رطوبت و نمایش

ان در LCD و سرور MQTT

تاریخ: 17/10/1403

دانشگاه : ملی مهارت تبریز

## فهرست

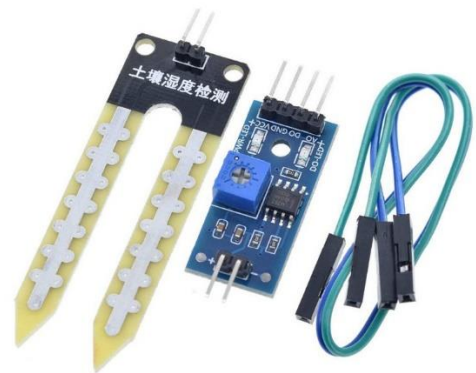
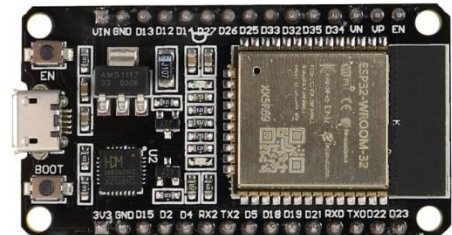
5	قدم 0: پیش نیاز ها.....
5	وسایل بکار رفته .....
7	شکل کامل مدار .....
7	کد های کامل برنامه.....
7	توابع loop و setup .....
8	چطور برد رو ریست کنیم.....
8	اضافه کردن برد esp32 به Arduino IDE .....
10	کدام برد رو انتخاب کنیم.....
11	قدم 1: خواندن اطلاعات سنسور رطوبت .....
11	پایه های سنسور رطوبت.....
12	سیم کشی .....
12	کد خواندن اطلاعات سنسور رطوبت .....
13	توضیحات کوتاه درباره کد .....
14	قدم 2: تنظیم سرعت فن .....
14	توضیحات کوتاه درباره فن ۴ سیم.....
15	سیم کشی .....
16	کد کنترل فن .....
17	توضیحات ساده درباره عملکرد کد .....
18	قدم 3: تنظیم سرعت فن بر اساس میزان رطوبت .....

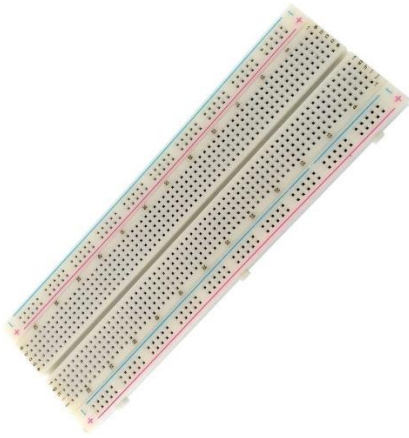
18.....	تابع تنظیم سرعت
18.....	تابع loop تا اینجا کار
19.....	تابع getActualFanSpeed
<b>20 .....</b>	<b>قدم 4: نمایش اطلاعات روی LCD</b>
20.....	سیم کشی پایه های مثبت LCD
20.....	سیم کشی پایه های منفی LCD
21.....	سیم کشی پایه های متصل به برد
21.....	کد مربوط به کنترل LCD
22.....	پارامتر های LiquidCrystal
22.....	نمایش اطلاعات روی LCD
22.....	تابع refreshLCD
23.....	ارور ها یا مشکلات LCD که حل کردیم
<b>23 .....</b>	<b>قدم 5: ارسال اطلاعات با MQTT</b>
23.....	اتصال به WiFi
23.....	اضافه کردن لایبرری PubSubClient
24.....	چطور فایل zip لایبرری رو وارد برنامه کنیم
25.....	اتصال به سرور یا بروکر MQTT
26.....	ارسال اطلاعات سنسور ها در MQTT
27.....	تابع publishMqttData
28.....	!!! تابع loop تا اینجا کار
29.....	راه اندازی برنامه MQTT Dash



قدم 0: پیش نیاز ها

وسایل بکار رفته

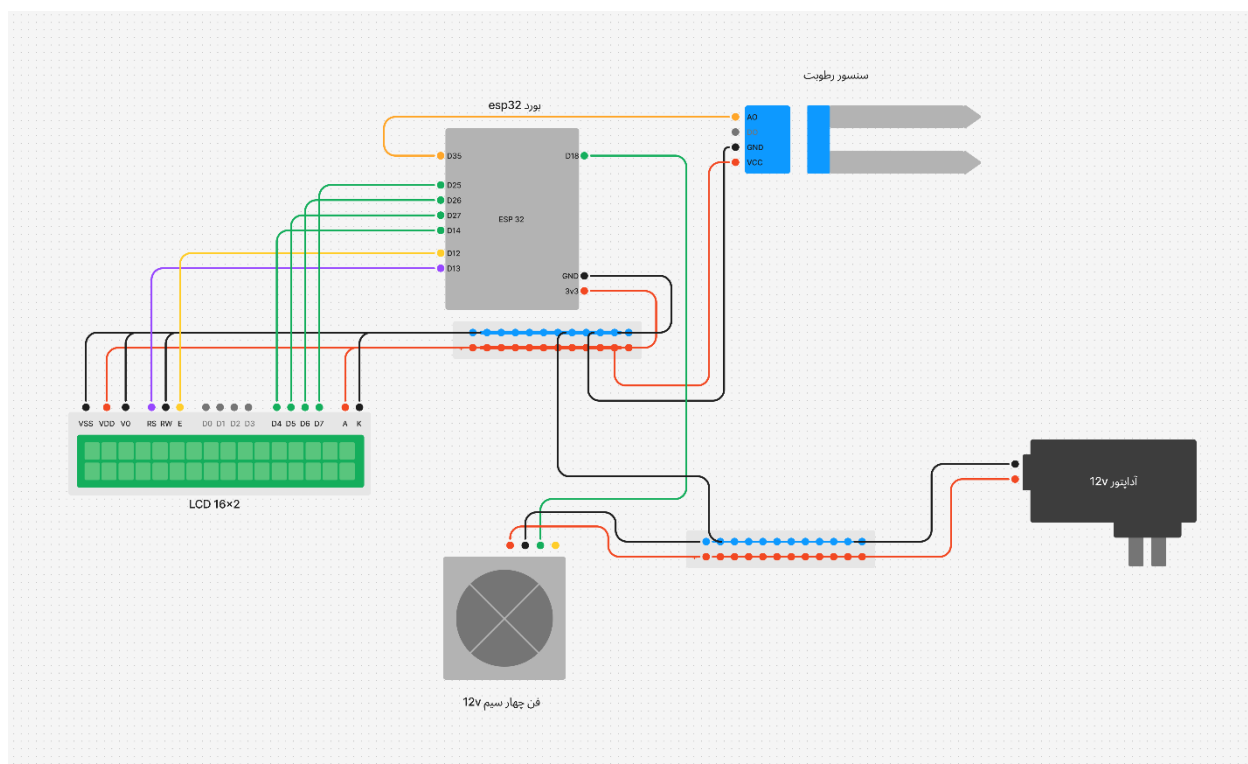




لیست لوازم:

1. برد esp32 wroom 30 pin
2. Lcd 16x2 بدون i2c
3. سنسور رطوبت خاک یا soil moisture
4. فن 4 سیم 12V brushless
5. آداپتور 12 V
6. سیم جامپر
7. برد مورد

## شکل کامل مدار



کد های کامل برنامه.

کد های کامل و تمام فایل ها در ریپازیتوری زیر قرار دارند.

<https://github.com/nezhadian/IOT2024>

## توابع loop و setup

### 1. تابع: setup

**کاربرد:** این تابع یک بار در ابتدای اجرای برنامه فراخوانی می‌شود.

**هدف:** برای انجام تنظیمات اولیه و آماده‌سازی سخت‌افزار و نرم‌افزار استفاده می‌شود.

### ۲. تابع: loop

**کاربرد:** این تابع به طور مداوم و بی‌پایان تکرار می‌شود.

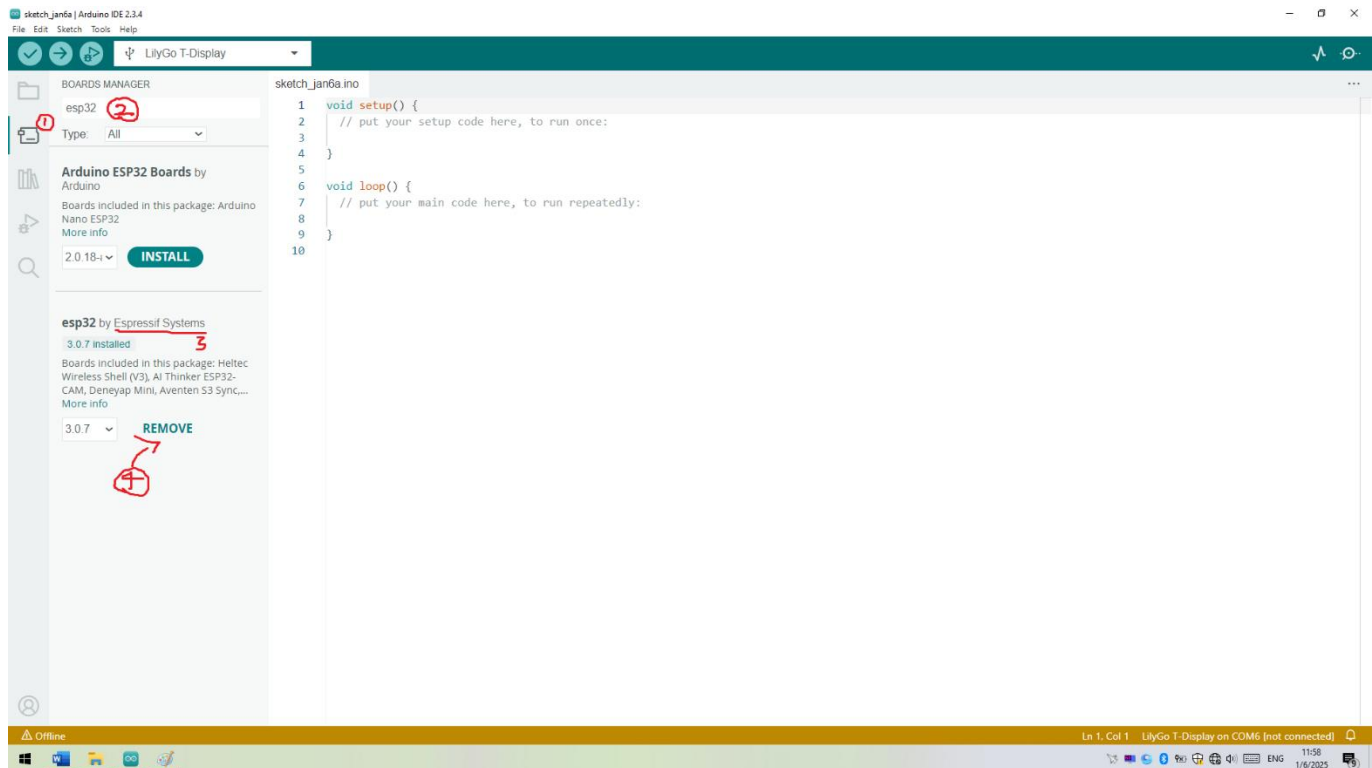
**هدف:** برای اجرای کدهای اصلی برنامه (مانند خواندن سنسورها، کنترل موتورها، ارسال داده‌ها و ...) استفاده می‌شود.

چطور برد رو ریست کنیم.



با استفاده از این دکمه روی برد میتوانیم برد رو ریست کنیم. با اینکار روند اجرای برنامه از اول شروع میشه.

## اضافه کردن برد esp32 به Arduino IDE



برای استفاده از برد esp32 لازمه که برد اون رو در Arduino IDE نصب کنیم.

1. روی Boards Manager کلیک میکنیم.

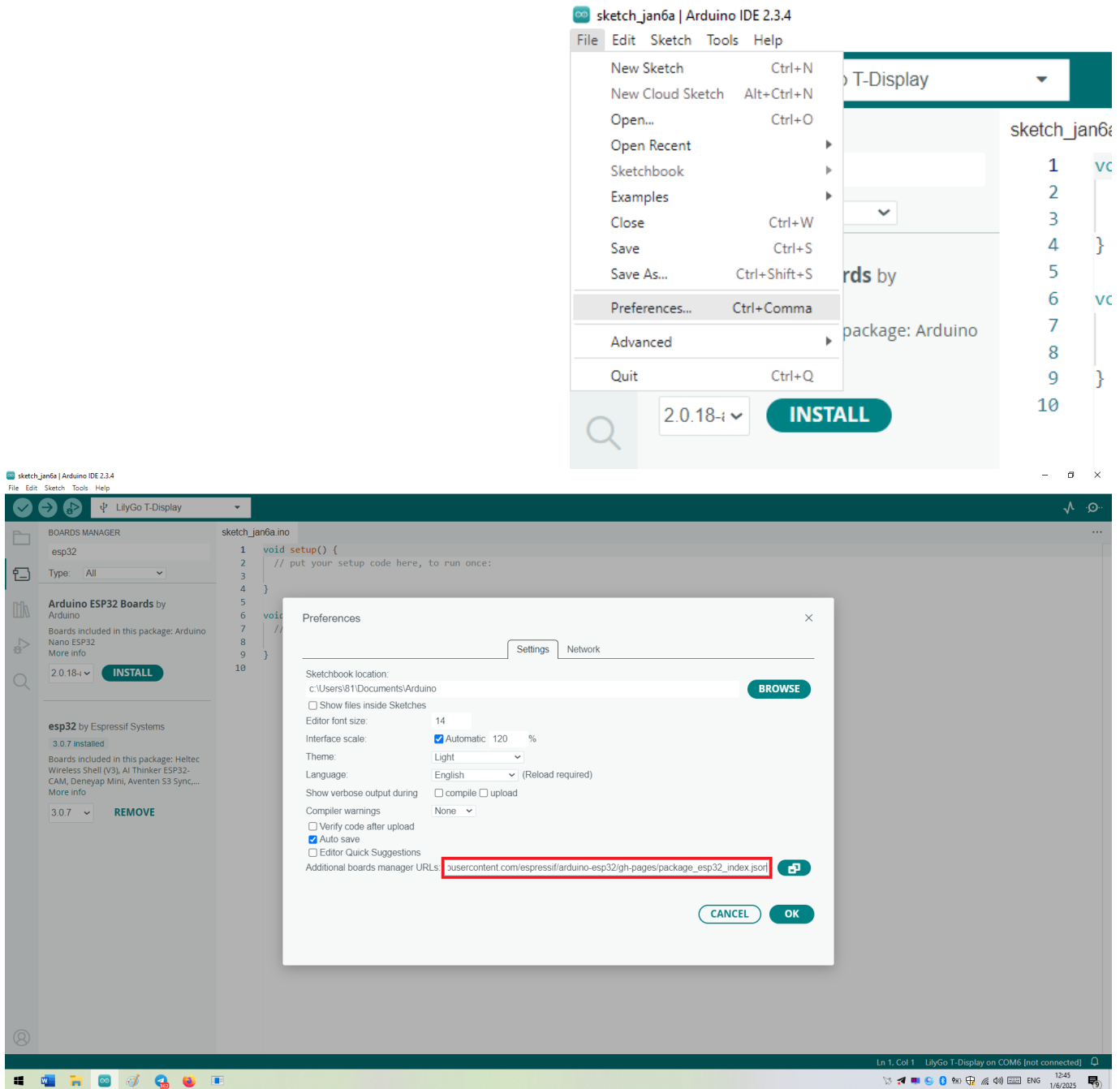
2. Esp32 رو سرچ میکنیم.



3. مطمئن میشیم که برد مورد نظر از طرف Espressif Systems است.

4. اون رو نصب میکنیم.

اگر برد esp32 در Boards Manager موجود نبود چیکار کنیم؟

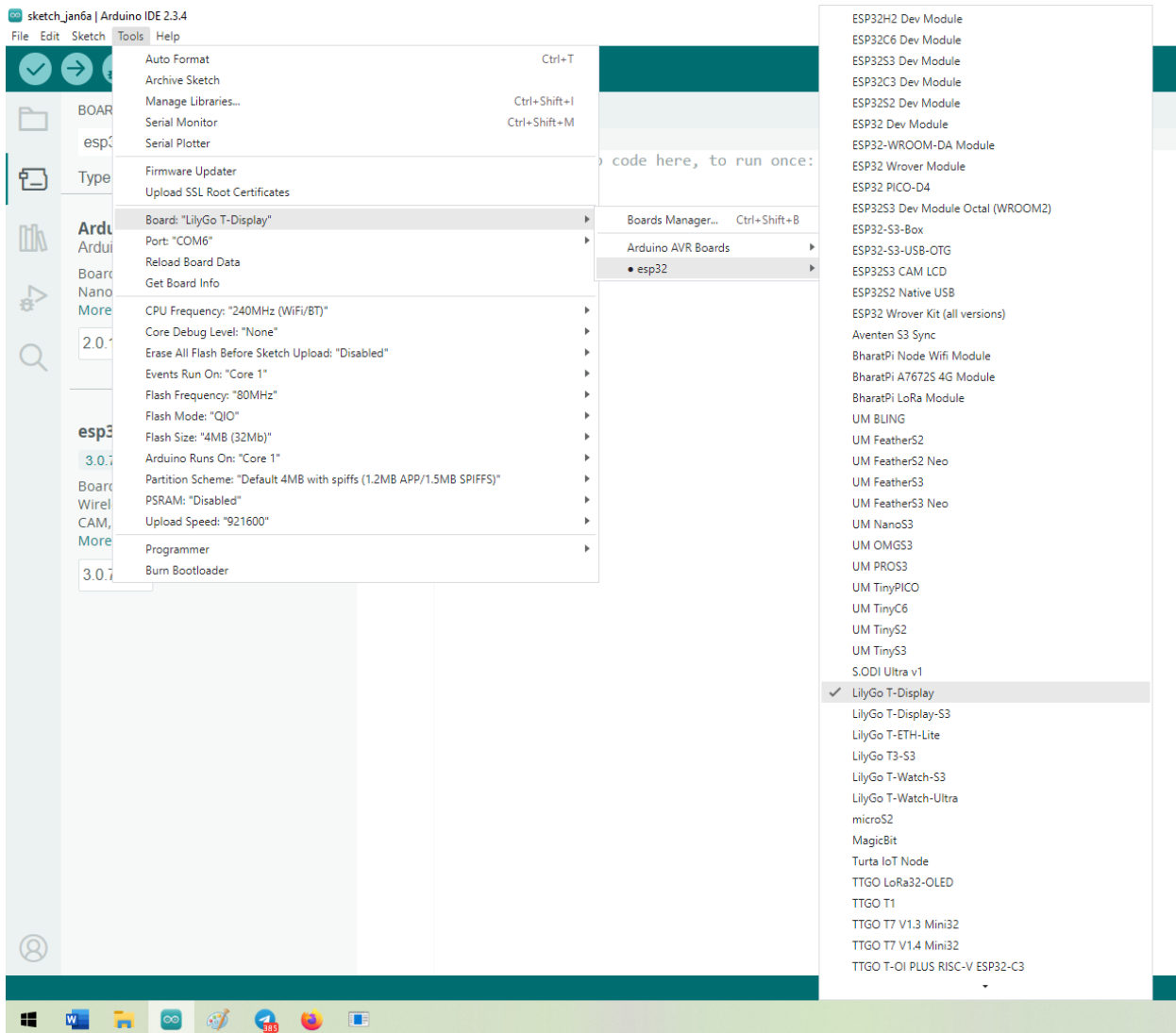


و این ادرس رو مینویسیم تا برنامه برد را پیدا کند.

[https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json)

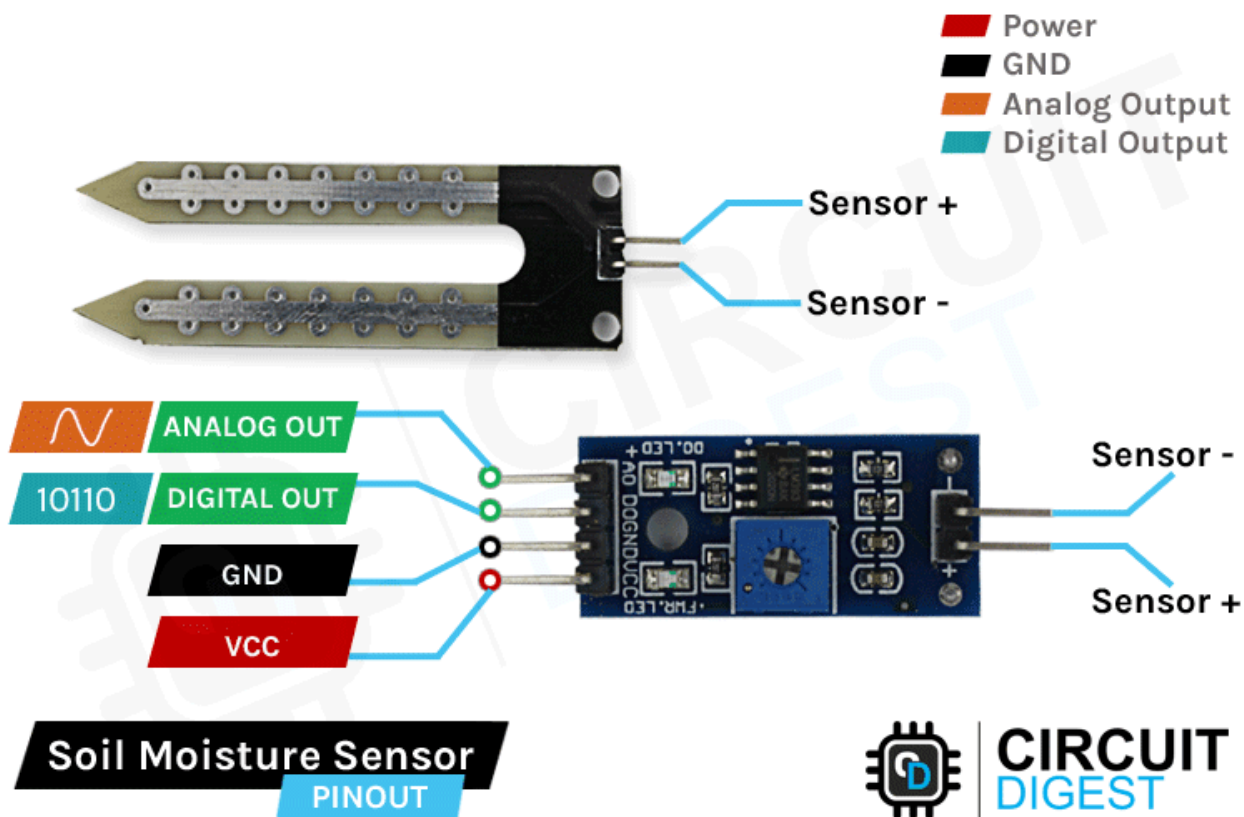
## کدام برد رو انتخاب کنیم.

برای این برد ما گزینه ی LilyGo T-Display سازگار است.



# قدم 1: خواندن اطلاعات سنسور رطوبت

پایه های سنسور رطوبت



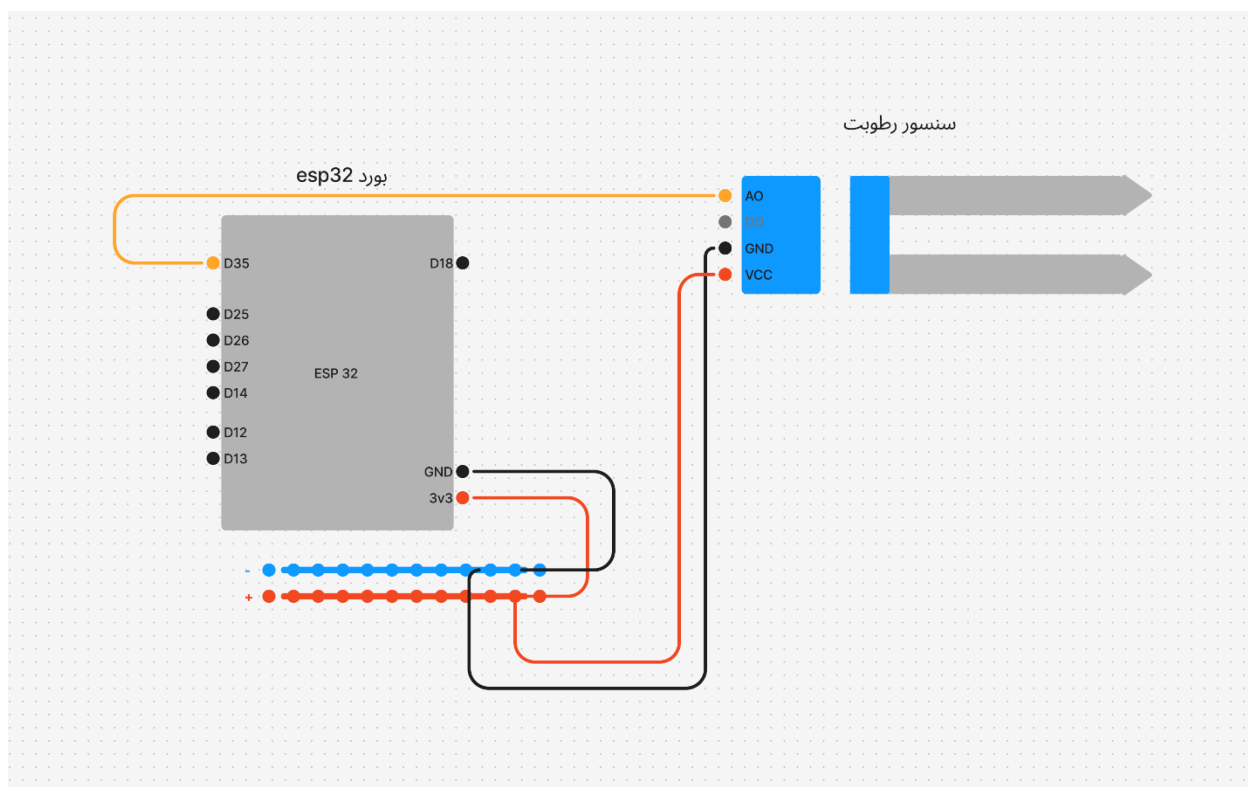
**VCC:** پایه مثبت ولتاژ معمولا 3.3v تا 5v

**GND:** پایه منفی

**DO:** خروجی دیجیتال که در صورت رسیدن به آستانه رطوبت، سیگنال HIGH یا LOW می دهد.

**AO:** خروجی آنالوگ که مقدار دقیق رطوبت خاک را ارائه می دهد.

## سیم کشی



بر اساس تصویر سیم کشی را انجام میدهیم.

## کد خواندن اطلاعات سنسور رطوبت

```
// تعریف پایه سنسور رطوبت خاک
const int soilPin = 35;

void setup() {
  // شروع ارتباط سریال برای نمایش اطلاعات
  Serial.begin(115200);

  // تنظیم پایه سنسور به عنوان ورودی
  pinMode(soilPin, INPUT);
}

// تابع خواندن مقدار رطوبت خاک
int readSoilMoistureValue() {
  // خواندن مقدار آنالوگ از سنسور
  int analog = analogRead(soilPin);

  // تبدیل مقدار آنالوگ به درصد رطوبت (۰ تا ۱۰۰)
  int soilValue = 100 - map(analog, 0, 4095, 0, 100);
}
```

```

// نمایش مقدار آنالوگ و درصد رطوبت در سریال مانیتور
Serial.print("Debug: soil analog ");
Serial.println(analog);
Serial.print("Debug: soil value ");
Serial.println(soilValue);

return soilValue;
}

void loop() {
// خواندن مقدار رطوبت خاک
int soilValue = readSoilMoistureValue();

// نمایش مقدار رطوبت خاک در سریال مانیتور
Serial.print("Sensor: Soil ");
Serial.println(soilValue);

// تاخیر ۱ ثانیه‌ای قبل از خواندن مجدد
delay(1000);
}

```

## توضیحات کوتاه درباره کد

### 1. تابع analogRead

- این تابع مقدار آنالوگ سنسور رطوبت خاک را می‌خواند و یک عدد بین ۰ تا ۴۰۹۵ برمی‌گرداند.
- نشان‌دهنده حداکثر رطوبت (خاک کاملاً خیس).
- ۴۰۹۵ نشان‌دهنده حداقل رطوبت (خاک کاملاً خشک).

### 2. دستور map

- این تابع یک عدد از یک بازه (مثلاً ۰ تا ۴۰۹۵) را به بازه‌ی دیگری (مثلاً ۰ تا ۱۰۰) تبدیل می‌کند.
- در این کد، map(analog, 0, 4095, 0, 100) مقدار آنالوگ را به یک درصد (۰ تا ۱۰۰) تبدیل می‌کند.

### 3. چرا خروجی map را منهای ۱۰۰ کرده‌ایم؟

- چون مقدار سنسور با رطوبت رابطه معکوس دارد (هرچه خاک خشک‌تر باشد، عدد بزرگ‌تر است).
- با منهای کردن از ۱۰۰، این رابطه معکوس می‌شود.
  - خاک کاملاً خشک (۰٪ رطوبت).
  - ۱۰۰ خاک کاملاً خیس (۱۰۰٪ رطوبت).
- این کار باعث می‌شود خروجی قابل‌درک‌تر باشد و نشان‌دهنده درصد واقعی رطوبت خاک باشد.

## قدم 2: تنظیم سرعت فن



### توضیحات کوتاه درباره فن ۴ سیم

#### سیم قرمز (+)

این سیم به منبع تغذیه ۱۲ ولت متصل می‌شود تا برق مورد نیاز فن تأمین شود. این سیم قرمز به هیچ وجه نباید با برد ارتباط داشته باشد. چون درجا برد رو میسوزاند.

#### سیم مشکی (GND)

این سیم به زمین (GND) مدار متصل می‌شود تا مدار کامل شود.

#### سیم سبز (PWM)

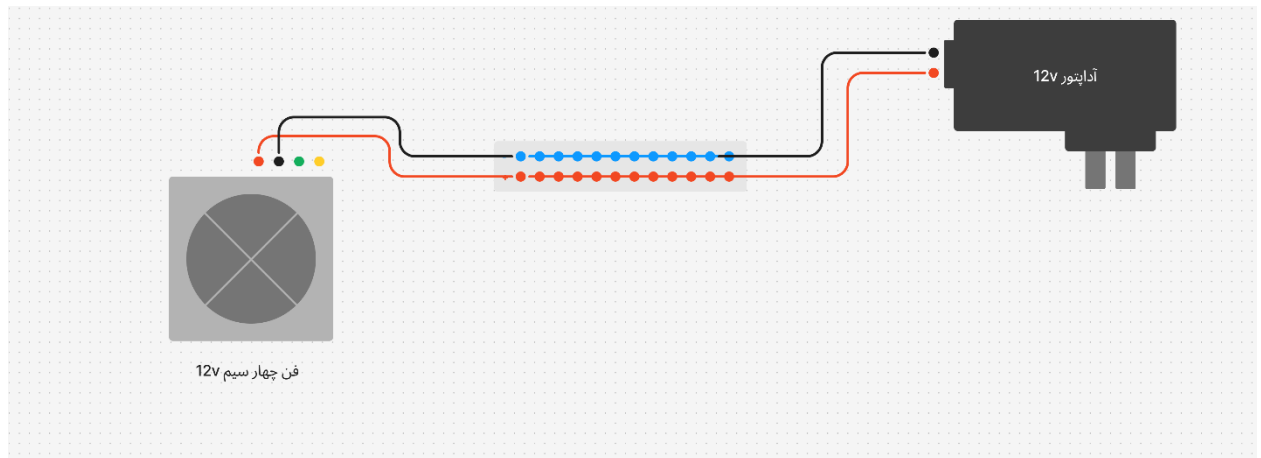
این سیم برای کنترل سرعت فن استفاده می‌شود. با ارسال سیگنال PWM (مدولاسیون عرض پالس) به این

سیم، می‌توان سرعت فن را تنظیم کرد.  
فرکانس معمول PWM برای فن‌های ۴ سیم، حدود ۲۵ کیلوهرتز است.

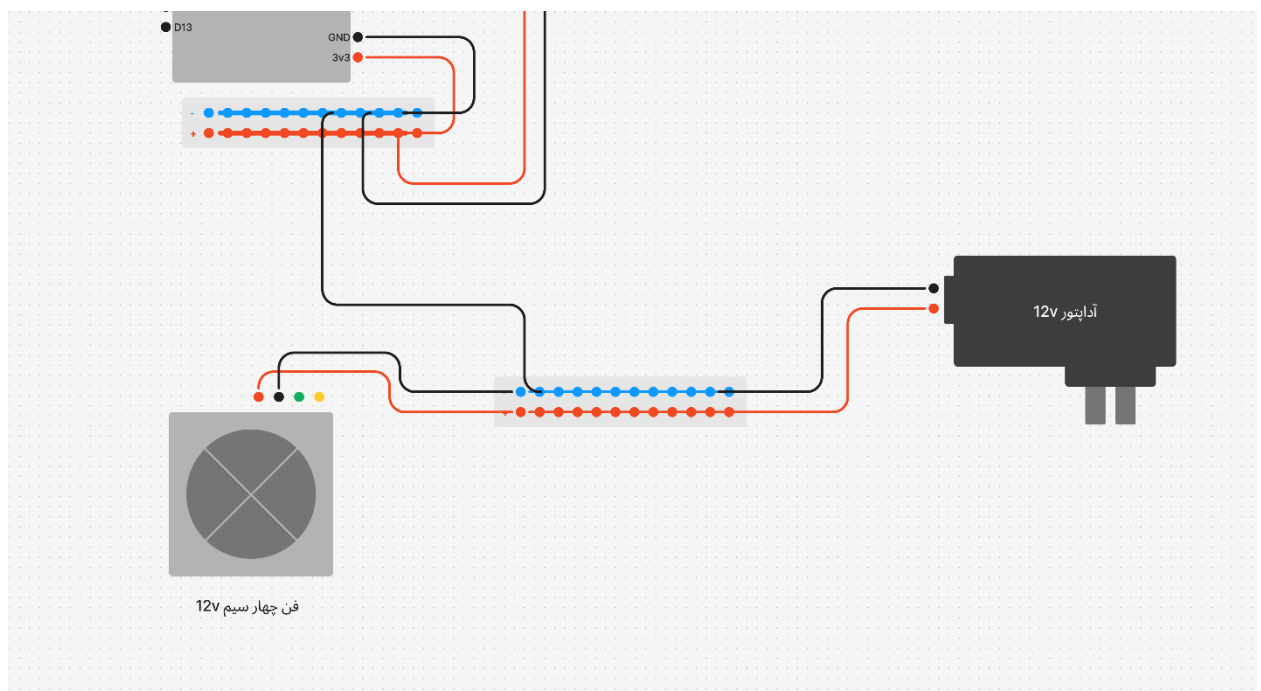
### سیم زرد (Tachometer)

این سیم برای خواندن سرعت فن استفاده می‌شود. که ما ازش استفاده نمی‌کنیم.

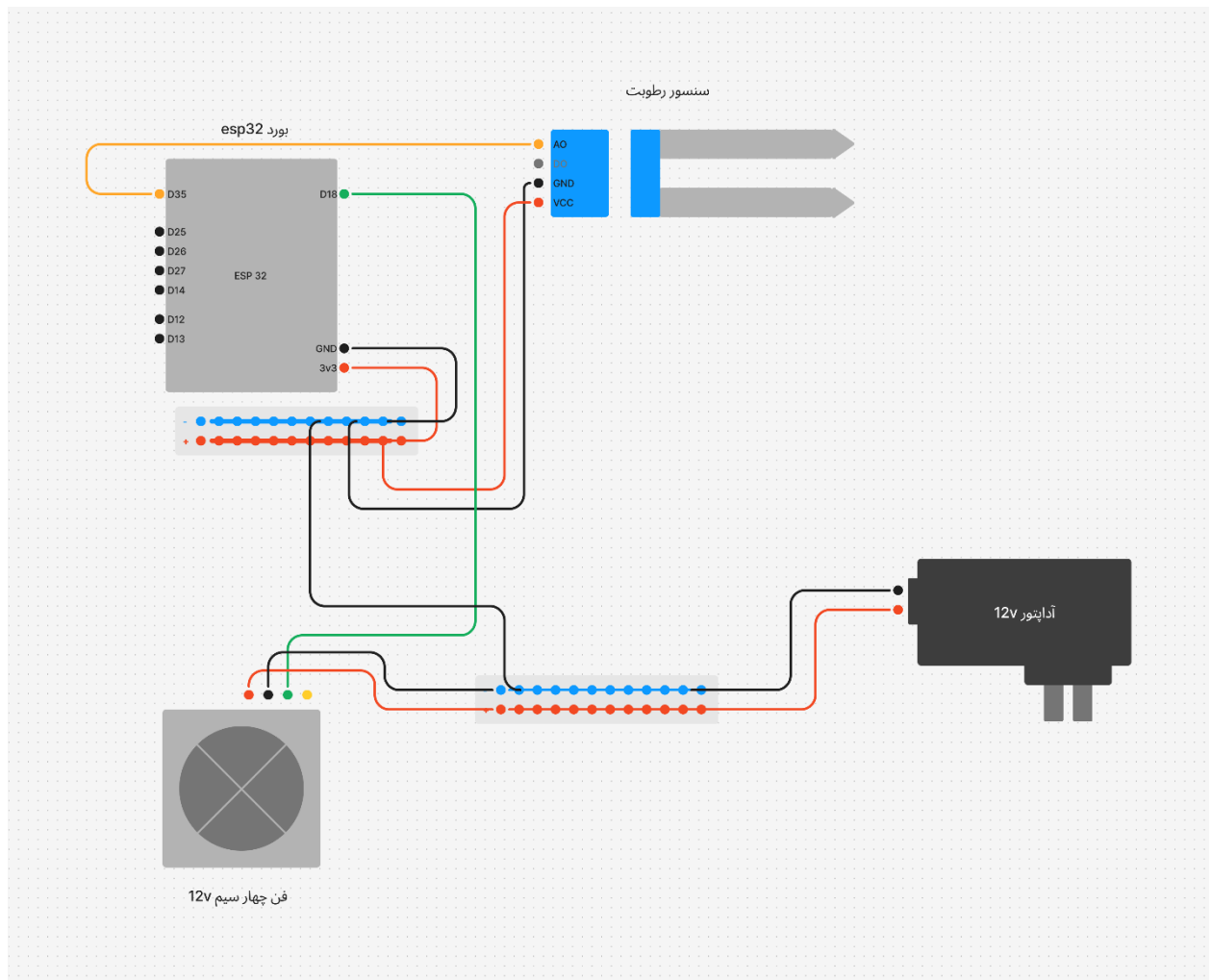
### سیم کشی



فن باید به این شیوه به اداپتور 12v وصل شود.



**نکته خیلی مهم:** GND یا پایه منفی اداپتور باید به پایه GND برد متصل شود در غیر اینصورت سرعت فن کنترل نمیشود.



در آخر هم سیم سبز رو به پایه‌ی D18 وصل میکنیم.

## کد کنترل فن

```
// تعریف پایه فن
const int fanPin = 18;
long curFanSpeed = 0;

void setup() {
  // شروع ارتباط سریال برای نمایش اطلاعات
  Serial.begin(115200);

  // برای کنترل سرعت فن PWM تنظیم
  const int pwmFreq = 20000; // کیلوهرتز ۲۰ PWM فرکانس
  const int pwmChannel = 0; // کانال PWM
  const int pwmResolution = 8; // (بیت = مقادیر ۰ تا ۲۵۵) PWM رزولوشن
  ledcAttachChannel(fanPin, pwmFreq, pwmResolution, 8); // PWM اتصال پایه فن به کانال

  // تنظیم سرعت اولیه فن
  setFanSpeed(255); // فن خاموش
}
```



```
// تابع تنظیم سرعت فن
void setFanSpeed(int speed) {
    ledcWrite(fanPin, speed); // PWM تنظیم سرعت فن با استفاده از
    curFanSpeed = speed;      // ذخیره سرعت فعلی فن
}

void loop() {
    // تغییر سرعت فن به صورت تستی
    setFanSpeed(0); // سرعت حداکثر
    delay(2000);    // تاخیر ۲ ثانیه
    setFanSpeed(128); // سرعت متوسط
    delay(2000);    // تاخیر ۲ ثانیه
    setFanSpeed(255); // توقف فن
    delay(2000);    // تاخیر ۲ ثانیه
}
```

## توضیحات ساده درباره عملکرد کد

۱. تابع `ledcAttachChannel`

**کاربرد:** این تابع پایه‌ی مورد نظر (`fanPin`) را به یک کانال PWM متصل می‌کند. و آن را روی فرکانس 20k hz تنظیم می‌کند. این مقدار بر اساس اطلاعات فن تنظیم شده است.

۲. تابع `setFanSpeed`

**کاربرد:** این تابع سرعت فن را تنظیم می‌کند.

**پارامترها:**

`Speed`: مقدار سرعت فن (بین ۰ تا ۲۵۵).

اگر `speed = 0` باشد، فن با حداکثر سرعت می‌چرخد.

اگر `speed = 255` باشد، فن خاموش می‌شود.

**عملکرد:**

با استفاده از `ledcWrite`، مقدار `speed` را به کانال PWM اعمال می‌کند.

مقدار `speed` را در متغیر `curFanSpeed` ذخیره می‌کند.

## قدم 3: تنظیم سرعت فن بر اساس میزان رطوبت

### تابع تنظیم سرعت

اگر بخواهیم بر اساس چیزی که در توضیحات پروژه نوشته شده سرعت فن را تنظیم کنیم باید به همین کدی بنویسیم.

```
void setFanSpeedBasedOnSoilMoisture(int soilValue){  
    if(soilValue > 30)  
        setFanSpeed(0);  
    else if(soilValue < 24)  
        setFanSpeed(255);  
}
```

اما چون این فن ما پتانسیل بیشتری داشت. ما کد هوشمندانه تری رو برایش نوشتیم.

```
void setFanSpeedBasedOnSoilMoisture(int soilValue){  
    if(soilValue < 30)  
        setFanSpeed(255);  
    else if(soilValue < 70)  
        setFanSpeed(128);  
    else if(soilValue <= 100)  
        setFanSpeed(0);  
    else  
        setFanSpeed(255);  
}
```



### تابع loop تا اینجا کار

در آخر این کد ما برای تابع loop است که مقدار رطوبت رو میخونه و بر اساس اون سرعت فن رو تنظیم میکنه.

```
void loop() {  
    // خواندن مقدار رطوبت خاک از سنسور  
    float soilValue = readSoilMoistureValue();  
  
    // نمایش مقدار رطوبت خاک در سریال مانیتور  
    Serial.print("Sensor: Soil ");  
    Serial.println(soilValue);  
}
```

```
// تنظیم سرعت فن بر اساس مقدار رطوبت خاک
setFanSpeedBasedOnSoilMoisture(soilValue);

// فعلی PWM از مقدار (RPM بر اساس) محاسبه سرعت واقعی فن
int actualFanSpeed = getActualFanSpeed(curFanSpeed);

// نمایش سرعت فن در سریال مانیتور
Serial.print("Sensor: Fan speed ");
Serial.println(actualFanSpeed);

// تاخیر ۵۰۰ میلی ثانیه قبل از تکرار حلقه
delay(500);
}
```

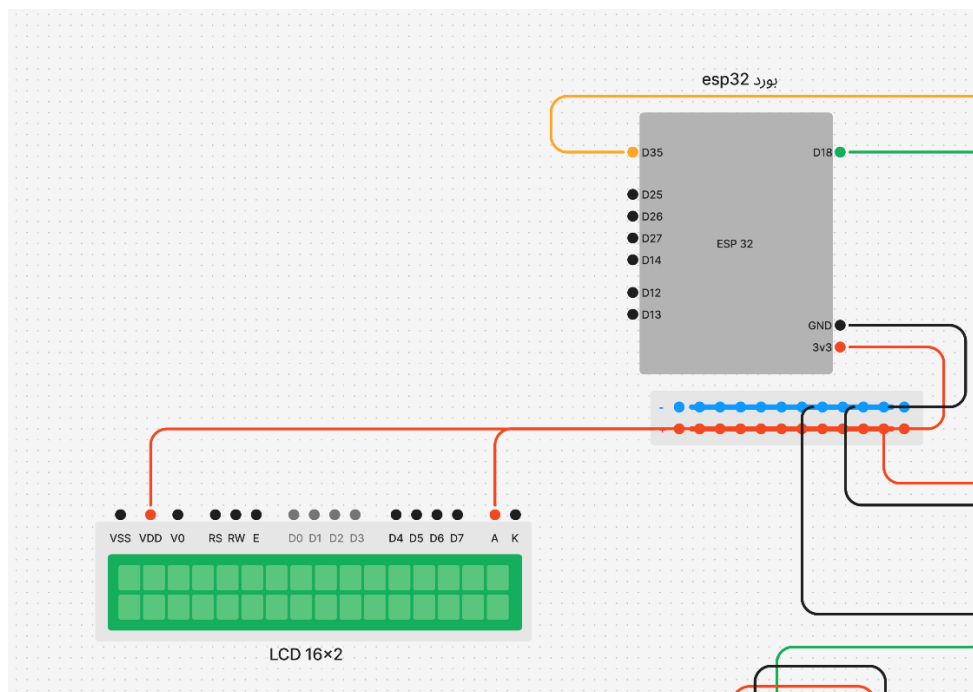
## تابع getActualFanSpeed

```
int getActualFanSpeed(int value){
    return 12000 - map(value,0,255,0,12000);
}
```

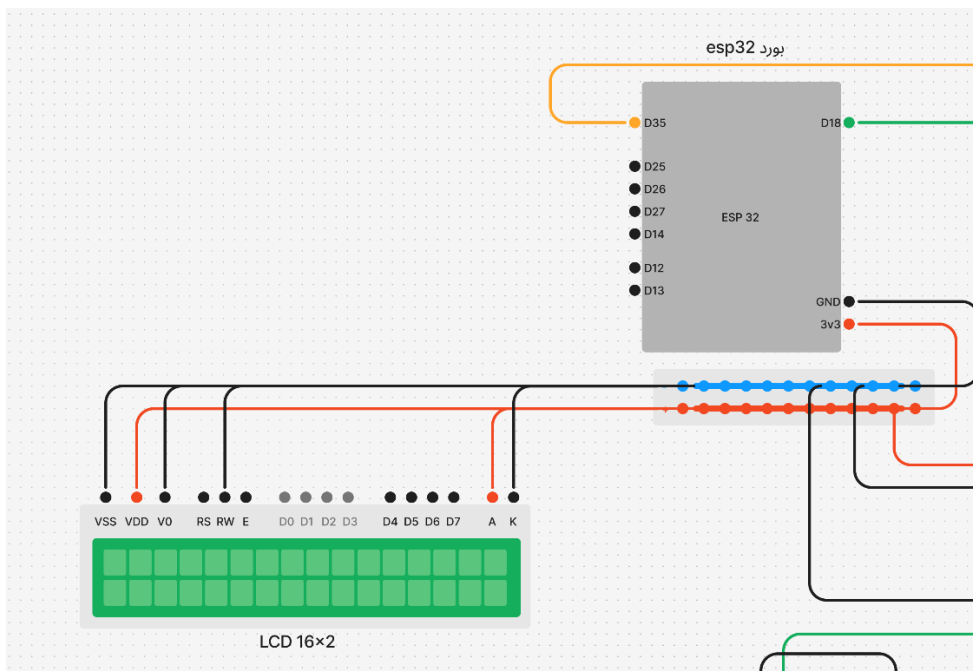
این تابع سرعت واقعی فن رو بر اساس PWM محاسبه میکند. که توی مستندات اصلی آن سرعت آن 12000 دور در دقیقه است که ما بر اساس نسبت PWM اون رو محاسبه میکنیم. البته این مقدار ممکنه با سرعت واقعی چرخش فن متفاوت باشه اما در حالت عادی تقریبا درسته.

## قدم 4: نمایش اطلاعات روی LCD

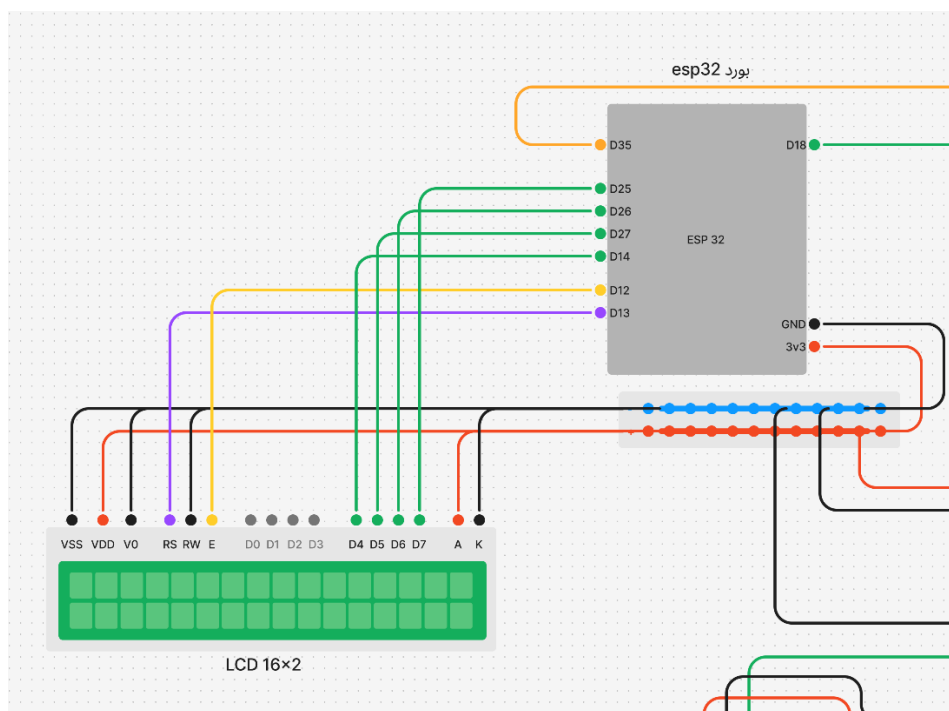
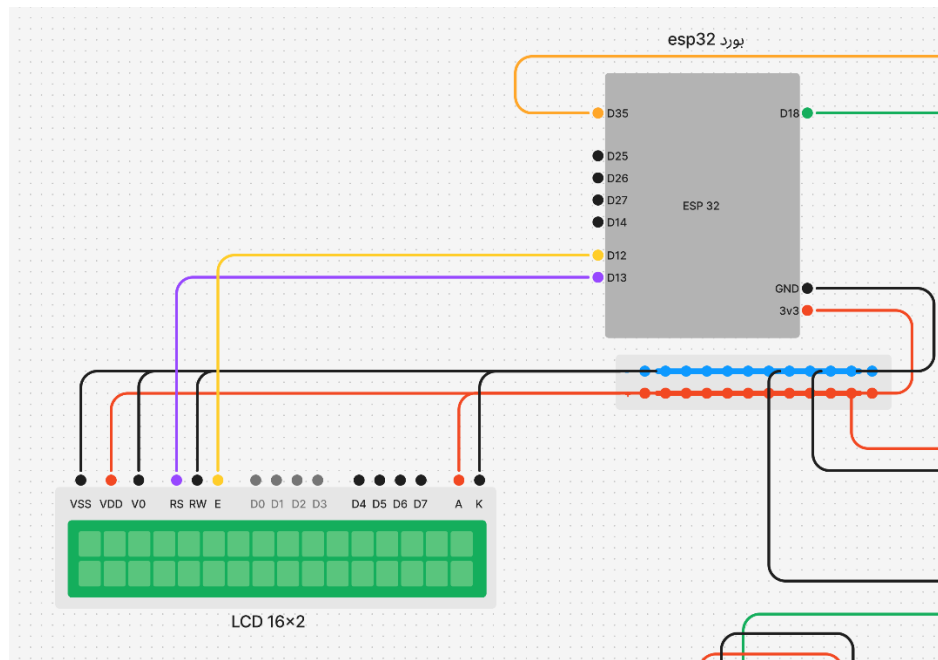
### سیم کشی پایه های مثبت LCD



### سیم کشی پایه های منفی LCD



## سیم‌کشی پایه‌های متصل به برد



## کد مربوط به کنترل LCD

```
// اضافه کردن کتابخانه
#include <LiquidCrystal.h>

// پیکربندی پایه‌های LCD
LiquidCrystal lcd(13, 12, 14, 27, 26, 25);

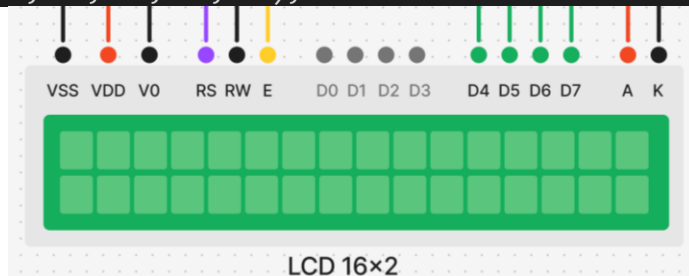
void setup() {
    // که برای ما LCD تنظیم ابعاد
    // سطر روی هم هستند 2
}
```

```
// و هر سطر 16 خانه دارد
lcd.begin(16, 2);
}
```

## پارامترهای LiquidCrystal

هر کدوم از این پارامترها مربوط به این پین‌ها روی LCD میباشند.

```
LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
```



## نمایش اطلاعات روی LCD

تا اینجا کار تابع loop به این شیوه درآمده است اول مقدار رطوبت خوانده میشود بعد سرعت فن بر اساس مقدار رطوبت تنظیم میشود و حالا باید سرعت فن و مقدار رطوبت روی LCD نمایش داده شود.

```
void loop() {
    float soilValue = readSoilMoistureValue();
    Serial.print("Sensor: Soil ");
    Serial.println(soilValue);

    setFanSpeedBasedOnSoilMoisture(soilValue);
    int actualFanSpeed = getActualFanSpeed(curFanSpeed);
    Serial.print("Sensor: Fan speed ");
    Serial.println(actualFanSpeed);

    refreshLCD(soilValue, actualFanSpeed);
    delay(500);
}
```

## تابع refreshLCD

```
void refreshLCD(int soilValue, int speed) {
    lcd.clear(); // پاک کردن صفحه LCD
    lcd.setCursor(0, 0); // قرار دادن کursor در ستون ۰ و سطر ۰ (سطر اول)
    lcd.print("Soil "); // نمایش متن "Soil " روی LCD
    lcd.print(soilValue); // نمایش مقدار رطوبت خاک
    lcd.setCursor(0, 1); // قرار دادن کursor در ستون ۰ و سطر ۱ (سطر دوم)
    lcd.print("RPM "); // نمایش متن "RPM " روی LCD
    lcd.print(speed); // نمایش مقدار سرعت فن
}
```

```
}
```

ارور ها یا مشکلات LCD که حل کردیم.

1. نمایشگر فقط مربع نشون میداد => پایه ولتاژ + به VIN وصل شده باید به 3v3 وصل میشد.
2. نمایشگر کاراکتر های چرت و پرت چاپ میکرد => یکی از سیم های اطلاعات D4,D5,D6,D7 به خوبی وصل نشده بود.

## قدم 5:ارسال اطلاعات با MQTT

### اتصال به WiFi

برای اتصال به WiFi از لایبرری WiFi استفاده میکنیم.

```
#include <WiFi.h>
```

بعد از وارد کردن لایبرری نام و رمز WiFi را تعریف میکنیم.

```
const char* ssid = "Galaxy Note 12";  
const char* password = "12345678";
```

و حالا با استفاده از تابع begin برنامه شروع به جستجوی شبکه مدنظر میکند و هر زمان در محدوده قرار بگیرد. به ان وصل میشود.

```
void setup() {  
  
    WiFi.begin(ssid, password);  
  
}
```

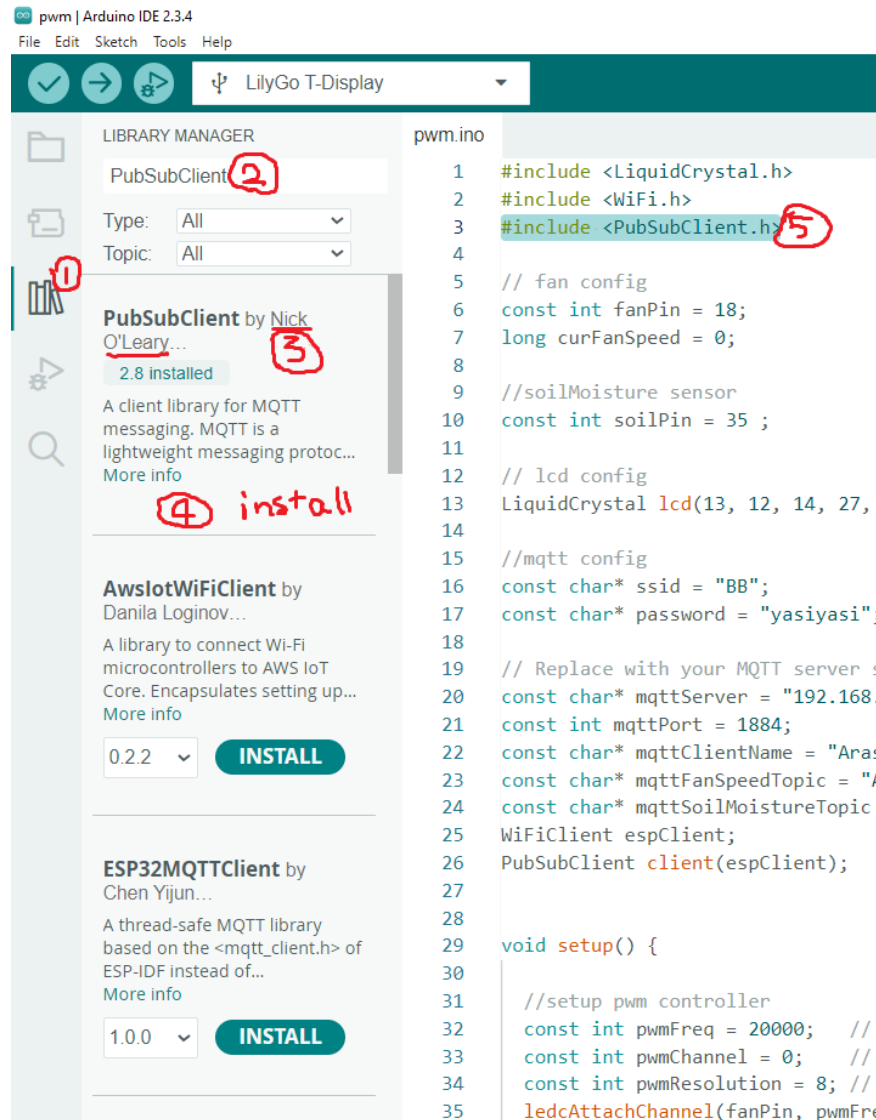
حالا چطور بفهمیم WiFi وصل شده یا ن؟ با استفاده از این کد.

```
if(WiFi.status() != WL_CONNECTED){  
    Serial.println("MQTT: Wifi is not connected");  
    return;  
}
```

تابع status() وضعیت فعلی اتصال را بر میگردداند که اگر برابر WL\_CONNECTED باشد. یعنی WiFi وصل است  
قطعه کد بالا قطع بودن اتصال رو بررسی میکند که در ادامه از آن نیز استفاده میکنیم.

### اضافه کردن لایبرری PubSubClient

برای استفاده از پروتکل MQTT ما از لایبرری PubSubClient استفاده میکنیم در ادامه نحوه اضافه کردن آن توضیح داده شده.



اول به قسمت Library Manager میرویم.

اسم کتابخانه PubSubClient رو مینویسم.

مطمئن میشیم که توسعه دهنده آن Nick O'Leary هست.

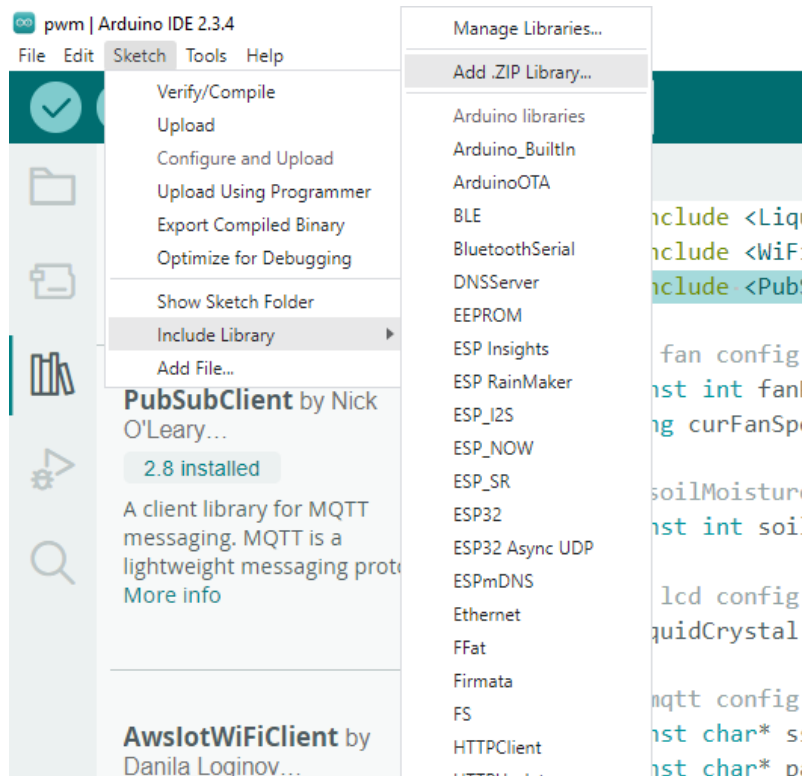
و دکمه install رو میزنیم.

بعد میتونیم با دستور include ازش استفاده کنیم.

چطور فایل zip لایبرری رو وارد برنامه کنیم.

گاهی اوقات لایبرری مورد نظر ما در لیست Library Manger پیدا نمیشود. و به جاش فایل zip انرا از وبسایت اصلی دانلود میکنیم. حالا چطور میتونیم این فایل zip رو وارد Arduino IDE کنیم.





به این مسیر میرویم. Sketch>Include Library>Add .ZIP Library... و بعد در پنجره ای که باز میشه. فایل zip که دانلود کردیم رو میتونیم اضافه کنیم.

## اتصال به سرور یا بروکر MQTT

بعد از اتصال به WiFi این بار نوبت آن میرسد که به سرور متصل بشیم. اول لایبرری رو وارد برنامه میکنیم.

```
#include <PubSubClient.h>
```

حالا اطلاعات اولیه رو تعریف میکنیم.

```
const char* mqttServer = "192.168.4.1";
const int mqttPort = 1883;
const char* mqttClientName = "ArashYasinOmid";
```

:mqttServer

همان ادرس سرور ماست. که در این مورد از IP سرور خود کلاس دانشگاه استفاده کردیم. همچنین میتونیم از test.mosquitto.org هم استفاده کنیم.

:mqttPort

معمولا ثابت روی 1883 هست.

:mqttClientName

اسمیه که باهاش خومون رو به سرور معرفی میکنیم.

```
WiFiClient espClient;  
PubSubClient client(espClient);
```

حالا باید یک کلاينت pubSub تعریف کنیم که با استفاده از آن اطلاعات رو به سرور میفرستیم.

و در تابع setup هم اون رو پیکربندی میکنیم.

```
void setup() {  
    client.setServer(mqttServer, mqttPort);  
}
```

حالا بعد از تعریف Client باید از وصل بودن به سرور مطمئن شویم.

```
if(!client.connected()){  
    client.connect(mqttClientName);  
    Serial.println("MQTT: Attempting to connect to broker...");  
    return;  
}
```

این کد چک میکنه که آیا به سرور وصل هستیم که اگر وصل نباشد. سعی میکنه وصل بشه.

:client.connected()

این تابع وضعیت اتصال رو بر میگردونه.

:client.connect(mqttClientName)

این تابع سعی میکنه. به سرور وصل بشه.

در ادامه از این کد استفاده میکنیم.

## ارسال اطلاعات سنسور ها در MQTT

بعد از اطمینان از اتصال به WiFi و سرور نوبت آن میرسد که اطلاعات سنسور رطوبت و سرعت فعلی فن را به سرور ارسال کنیم برای اینکار ما به topic نیاز داریم topic ها مسیر های یکتایی بودن که به سنسور مورد نظر اشاره میکنند.

اول تاپیک هارو تعریف میکنیم. این تاپیک ها بعدا در برنامه MQTT Dash هم استفاده خواهند شد.

```
const char* mqttFanSpeedTopic = "AYO/fan";//تایپیک سرعت فن
const char* mqttSoilMoistureTopic = "AYO/soil";//تایپیک سنسور رطوبت
```

## تابع publishMqttData

در کد برنامه این تابع برای ارسال اطلاعات سنسور رطوبت و سرعت فن به سرور MQTT استفاده میشود.

```
void publishMQTTData(int soil, int fanSpeed) {
    // بررسی اتصال به Wi-Fi
    if (WiFi.status() != WL_CONNECTED) {
        Serial.println("MQTT: Wifi is not connected"); // چاپ پیام خطا اگر
        return; // خروج از تابع
    }

    // بررسی اتصال به سرور MQTT
    if (!client.connected()) {
        client.connect(mqttClientName); // تلاش برای اتصال به سرور
        Serial.println("MQTT: Attempting to connect to broker..."); // چاپ پیام در حال اتصال
        return; // خروج از تابع
    }

    // تبدیل مقادیر سرعت فن و رطوبت خاک به رشته
    char* fanSpeedPtr = intToCharPtr(fanSpeed); // تبدیل سرعت فن به رشته
    char* soilSpeedPtr = intToCharPtr(soil); // تبدیل رطوبت خاک به رشته

    // MQTT ارسال داده‌ها به سرور
    client.publish(mqttFanSpeedTopic, fanSpeedPtr); // ارسال سرعت فن
    client.publish(mqttSoilMoistureTopic, soilSpeedPtr); // ارسال رطوبت خاک

    // آزاد کردن حافظه اختصاص داده شده برای رشته‌ها
    free(fanSpeedPtr); // آزاد کردن حافظه سرعت فن
    free(soilSpeedPtr); // آزاد کردن حافظه رطوبت خاک

    // چاپ پیام موفقیت‌آمیز بودن ارسال داده‌ها
    Serial.println("MQTT: data published.");
}
```

## عملکرد:

1. اتصال به WiFi رو چک میکنه و اگر وصل نبود از تابع بیرون میاد و به حلقه اصلی برنامه بر میگردد.
2. اتصال به سرور رو چک میکنه. اگر وصل نبود یکبار سعی میکنه وصل بشه. و بعد از تابع بیرون میاد.
3. حالا اگر WiFi وصل بود و سرور هم وصل بود. اطلاعات رو به سرور میفرسته. برای اینکار اول باید متغیر های soil و fanSpeed که از نوع int هستند. به string تبدیل شوند. چون تابع client.publish فقط

string رو قبول میکنه. این کار تبدیل int به string با استفاده از تابع IntToCharPtr انجام میشود. و بعد از ارسال اطلاعات تابع free متغیر های string رو از حافظه آزاد میکنه.

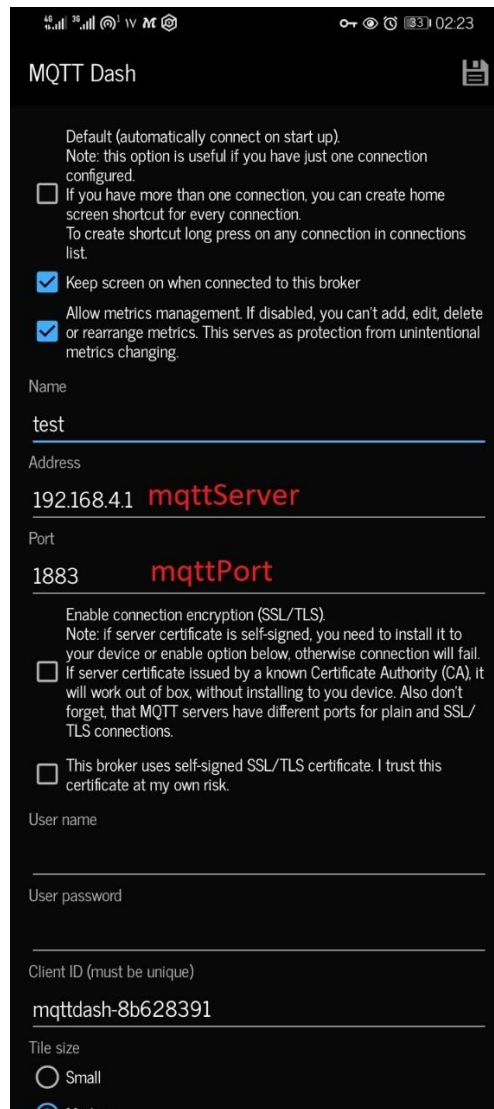
## !!! تابع loop تا اینجا کار

حالا تا اینجا کار تابع loop به این شیوه درآمده است.

نخست مقدار سنسور رطوبت رو میخونه بر اساس اون سرعت فن رو تنظیم میکنه. و بعد مقدار سنسور رطوبت و سرعت فن رو توی LCD نمایش میده و بعد از اون اونهارو به سرور MQTT ارسال میکنه.

```
void loop() {  
    float soilValue = readSoilMoistureValue();  
    Serial.print("Sensor: Soil ");  
    Serial.println(soilValue);  
  
    setFanSpeedBasedOnSoilMoisture(soilValue);  
    int actualFanSpeed = getActualFanSpeed(curFanSpeed);  
    Serial.print("Sensor: Fan speed ");  
    Serial.println(actualFanSpeed);  
  
    refreshLCD(soilValue, actualFanSpeed);  
    publishMQTTData(soilValue, actualFanSpeed);  
    delay(500);  
}
```

## راه اندازی برنامه MQTT Dash



The screenshot shows the MQTT Dash app interface on a mobile device. At the top, there's a status bar with signal, battery, and time (02:23). The app title 'MQTT Dash' is at the top left, and a save icon is at the top right. Below the title, there's a section for default settings with a note: 'Default (automatically connect on start up). Note: this option is useful if you have just one connection configured.' There are three checkboxes: the first is unchecked with a note about creating home screen shortcuts; the second is checked for 'Keep screen on when connected to this broker'; the third is checked for 'Allow metrics management'. Below this is a 'Name' field with the value 'test'. Then an 'Address' field with '192.168.41' and a red label 'mqttServer'. A 'Port' field with '1883' and a red label 'mqttPort'. A section for 'Enable connection encryption (SSL/TLS)' with a note about certificates and two unchecked checkboxes. Then 'User name' and 'User password' fields. A 'Client ID (must be unique)' field with the value 'mqttdash-8b628391'. Finally, a 'Tile size' section with 'Small' and 'Medium' radio buttons, where 'Medium' is selected.

MQTT Dash

Default (automatically connect on start up).  
Note: this option is useful if you have just one connection configured.

☐ If you have more than one connection, you can create home screen shortcut for every connection.  
To create shortcut long press on any connection in connections list.

☒ Keep screen on when connected to this broker

☒ Allow metrics management. If disabled, you can't add, edit, delete or rearrange metrics. This serves as protection from unintentional metrics changing.

Name

test

Address

192.168.41 mqttServer

Port

1883 mqttPort

Enable connection encryption (SSL/TLS).  
Note: if server certificate is self-signed, you need to install it to your device or enable option below, otherwise connection will fail.

☐ If server certificate issued by a known Certificate Authority (CA), it will work out of box, without installing to you device. Also don't forget, that MQTT servers have different ports for plain and SSL/TLS connections.

☐ This broker uses self-signed SSL/TLS certificate. I trust this certificate at my own risk.

User name

User password

Client ID (must be unique)

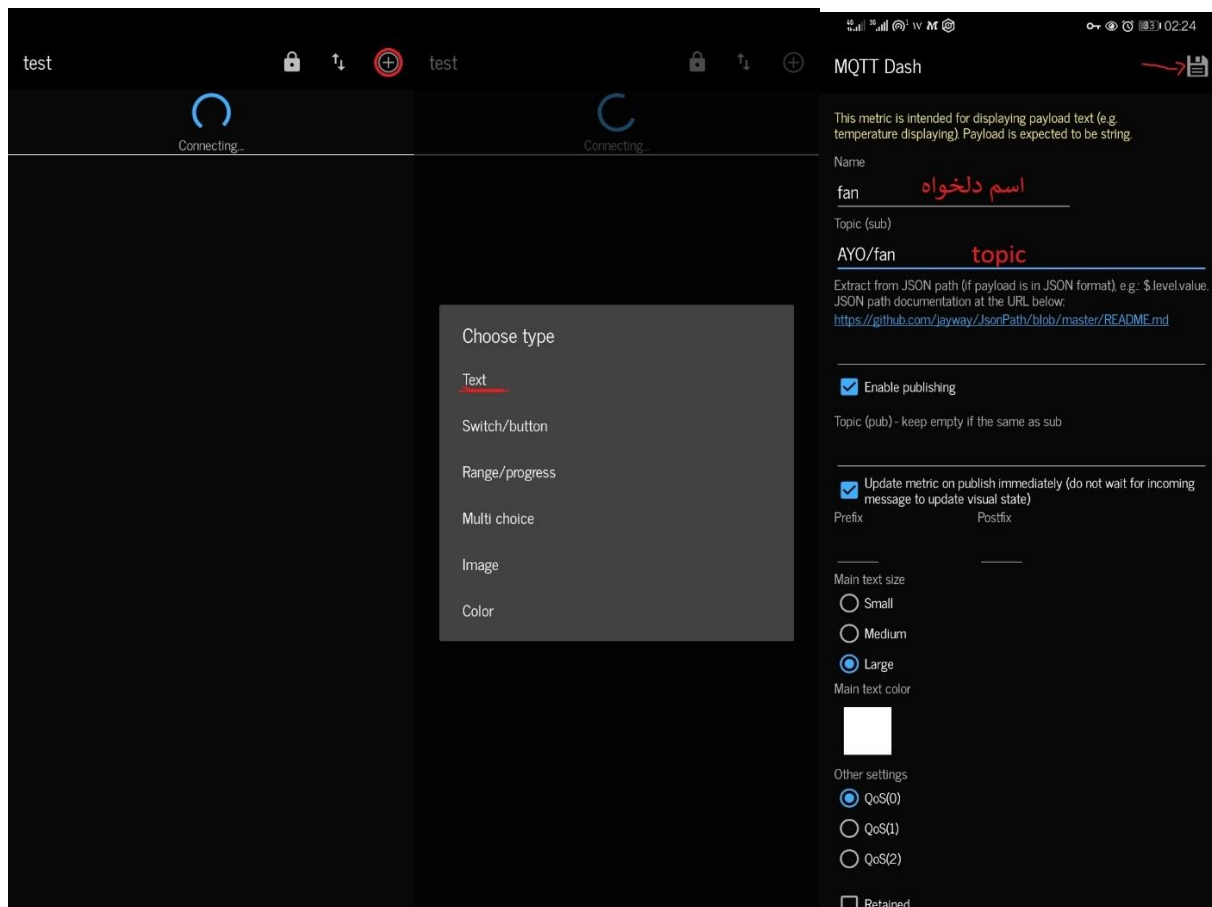
mqttdash-8b628391

Tile size

☐ Small

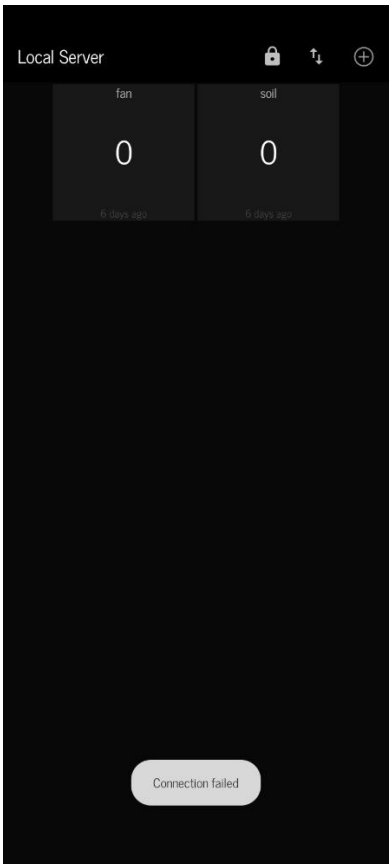
☒ Medium

اول ادرس سرور که توی کد مشخص کرده بودیم + پورت موردنظر رو وارد میکنیم.



بعد از اتصال به سرور باید یک topic از نوع Text بسازیم و از همون topic که در کد نوشتیم در اینجا هم بنویسیم به دو topic نیاز داریم یکی برای fan و یکی برای soil

```
const char* mqttFanSpeedTopic = "AYO/fan";  
const char* mqttSoilMoistureTopic = "AYO/soil";
```



# با تشکر از توجه شما.

استاد : اصغر کریم پور گمش آبادی.

گروه دانشجویان :

1. یاسین ابراهیم نژادیان

2. امید اعظامی

3. آرش حسین پور

موضوع: پروژه شماره 1 کنترل سرعت فن بر اساس مقدار سنسور رطوبت و نمایش

ان در LCD و سرور MQTT

تاریخ: 17/10/1403

دانشگاه : ملی مهارت تبریز