

# گزارش پروژه کارشناسی

عنوان پروژه: ربات گلدان مکان یاب نور خورشید

اعضای گروه:

محسن احمدی ۹۳۳۳۲۸۵

فاطمه چنگیزیان ۹۳۳۲۶۸۲

تاریخ ارائه: ۹۷/۸/۲۰

استاد راهنما: مهندس شناور

استاد ناظر: دکتر یثربی

## فهرست عناوین:

۳	.....مقدمه
۴	.....بخش نرم افزاری
۲۰	.....بخش سخت افزاری
۳۱	.....منابع
۳۳	.....تخمین هزینه های ساخت
۳۴	.....تصاویر واقعی از ربات

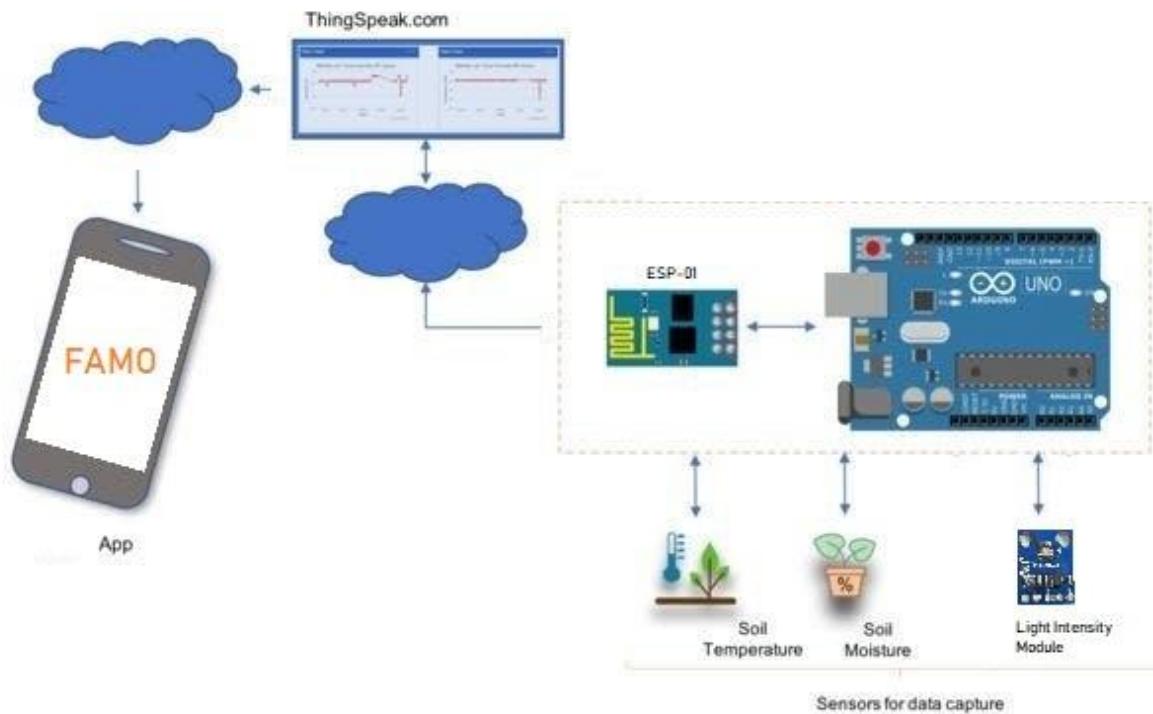
## مقدمه

با گسترش آپارتمان نشینی و علاقه‌ی افراد به نگهداری گل و گیاه در منزل، نگهداری صحیح از این گیاهان عاملی مهم در سلامت و طول عمر گیاه محسوب می‌شود. یکی از مهمترین نیازهای یک گیاه برای رشد و طول عمر، جذب نور خورشید به مقدار مناسب و مدت کافی است. از طرفی مکان خورشید در طول روز تغییر می‌کند و شدت نور آن روی گیاهان پایدار نیست به همین دلیل گیاهان آپارتمانی غالباً به دلیل مکان نامناسب برای جذب نور یا جذب نور بیش از حد، عمر کوتاهی دارند. در کنار دیگر وسایل هوشمند منزل، نیاز به یک گلدان هوشمند که مسئله‌ی جذب نور مورد نیاز را حل کند احساس می‌شود. هدف از ساخت این دستگاه نیز، رفع مشکل جذب نور توسط یک ربات هوشمند است که مکان نور خورشید را پیدا می‌کند، به سمت آن می‌رود و مقدار کافی از آن را جذب می‌کند و سپس به سایه بازمی‌گردد.

به منظور رفع مشکل نورسانی به گیاه در زمان‌های مختلف مثل هنگامی که فرد در خانه حضور ندارد یا مکان نور خورشید مرتب تغییر می‌کند، اگر بتوان رباتی را طراحی کرد که با تشخیص مکان نور خورشید به سمت آن برود و به مقدار مناسبی در نور بماند و سپس به سایه بازگردد، مشکل جذب نور و رشد مناسب گیاه در مکان‌های سرپوشیده برطرف می‌شود.

این دستگاه هم یک ربات چرخدار است که بعنوان گلدان استفاده می‌شود، این گلدان متحرک با استفاده از مدارهای حساس به نور که روی آن تعییه شده است از نور اطراف خود نمونه برداری می‌کند، جهتی که بیشترین نور را دارد انتخاب کرده و به سمت آن حرکت می‌کند. در مسیر رسیدن به منبع نور با استفاده از مدار تشخیص مانع از برخورد به موانع پیش رو جلوگیری کرده و به مقصد خود که نور مستقیم خورشید است میرسد. پس از گذشت مدتی معلوم در نور خورشید، ربات حرکت کرده و به سمت سایه می‌رود. اطلاعات مربوط به \* \* \* مقدار و مدت زمان نور دریافتی از طریق مدار ارتباطی بیسیم قابل ارسال به کاربر است. این ربات از دو موتور DC برای حرکت خود استفاده می‌کند که توسط یک درایور L298N برای کنترل سرعت و جهت حرکت فرمان می‌گیرند. پردازش‌های مربوط به حرکت‌ها و دریافت اطلاعات مربوط به نور محیط توسط پردازنده‌ی AVR Atmega328 انجام می‌شود. به منظور عدم برخورد به مانع در بین راه از سنسورهای اولتراسونیک srf04 استفاده شده است. این سنسورها به ارسال امواج فرماصوت و دریافت انعکاس آنها و محاسبه‌ی زمان رفت و برگشت موج با توجه به سرعت آن می‌توانند فاصله موانع مقابل خود را محاسبه کنند.

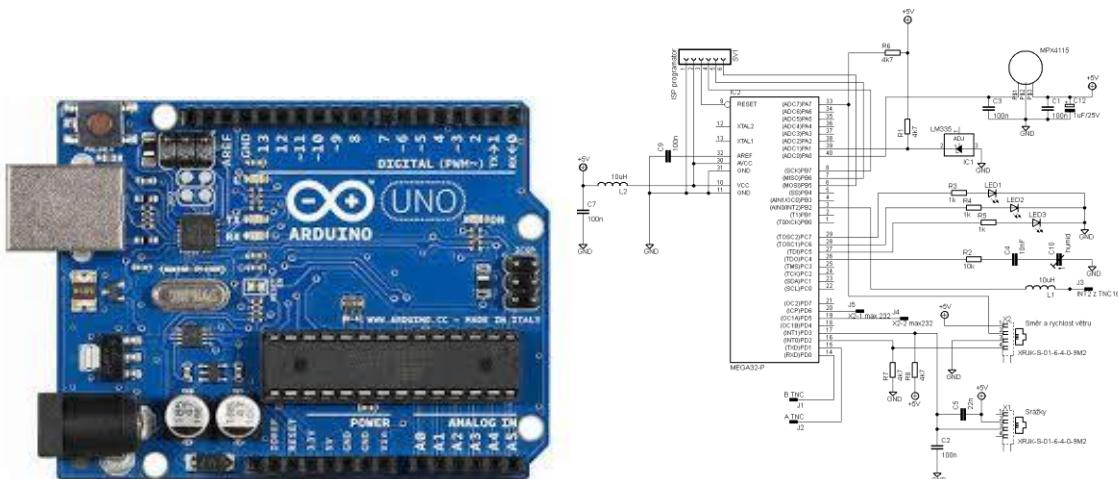
## بخش نرم افزاری:



## گام اول: ابزار و قطعات مورد نیاز

ربات ذکر شده شامل قسمت های زیر است:

(Microcontroller) – Arduino UNO •



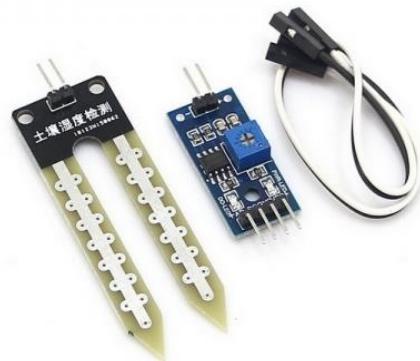
(Communication Module) – ESP8266-01 •



(1-Wire Digital Temperature sensor for use on soil) – DS18B20 •



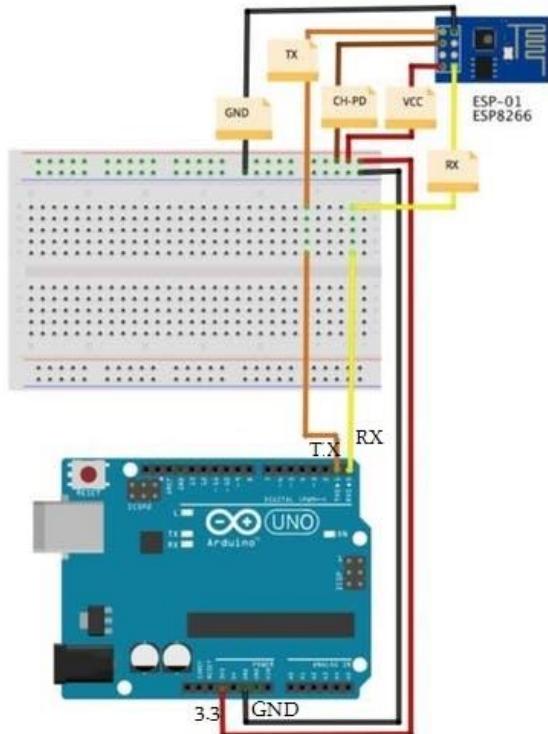
(Soil Humidity sensor) – YL-69 + LM393 •



1 x 4K7 ohm resistor (to be used with DS18B20) •

## قدم اول : راه اندازی اولیه ESP8266-01

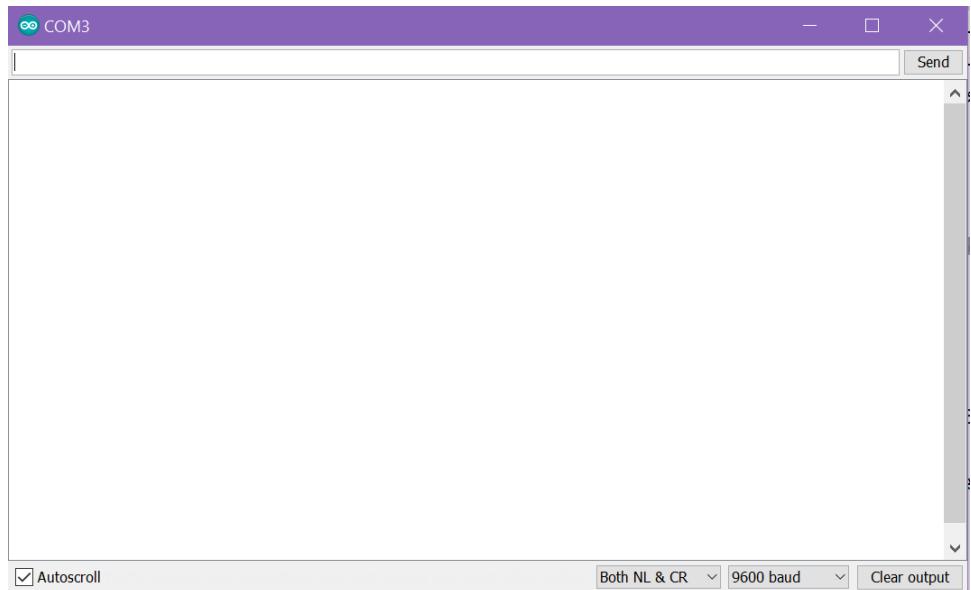
به عنوان یک پل سریالی استفاده می شود، بدین معنی است که ما آن را با استفاده از دستورات ATCOMMAND برنامه ریزی می کنیم. اولین چیزی که باید به آن بپردازیم این است که ESP-01 در سرعت Baud Rate صحیح باشد. در مورد پروژه ما، 9600 است. معمولاً ESP-01 از کارخانه با 115200 9600 تغییر دهیم. برای اینکار مازول خود را به صورت زیر به آردوینو متصل میکنیم:



سپس آردوینو را به کامپیوتر متصل میکنیم و پس از باز کردن نرم افزار آردوینو مثال مورد نظر را از مسیر File> Examples> 01.Basics> BareMinimum: زیر لود میکنیم. این مثال عملایک کد خالی است که به ما اطمینان می دهد بین آردوینو و مازول ESP هیچگونه ارتباطی برقرار نباشد.

در واقع دلیل اصلی این کار این است که قبلاً از ارتباط آردینو با ماژول ESP اطمینان حاصل کنیم که آردینو از هیچ ارتباط سریالی (TX,RX) دیگری استفاده نمیکند این کار برای ایجاد یک ارتباط مناسب و صحیح با ماژول ESP ضروری است.

اگر بر روی کامپیوتر Serial Monitor آردینو را باز کنیم و سرعت آن را بر روی 115,200 baud تنظیم کنیم سپس دستور "AT" را ارسال کنیم ماژول ESP-01 باید به ما کلمه را "OK" بازگرداند.



- سریال مانتیور آردینو

سپس سرعت BAUD RATE را به کمک دستور زیر تغییر می دهیم:  
**"AT + CIOBAUD = 9600"**

یا از دستور زیر استفاده میکنیم:

AT+UART\_DEF=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>

9600 baud / 8 data bits / 1 stop bits / none parity / flow control: به عنوان مثال

**"AT + UART\_DEF = 9600,8,1,0,0"**

در این حالت در صفحه سریال مانتیور در باکس پایین آن سرعت را به 9600 baud تغییر می دهیم سپس دوباره در بالای صفحه دستور AT را ارسال و جواب OK را می بینیم حال باید ماژول خود را در مود Client شبکه وایفای خود قرار بدھیم. از دستور زیر استفاده میکنیم:

### "AT + CWMODE = 1"

حال باید ماژول را به شبکه وایفای خود وصل کنیم.

برای اینکار با استفاده از دستور زیر "network\_name" نام شبکه وایفای خود و پسورد شبکه را جایگذاری میکنیم

### "AT + CWJAP = "network\_name", "network\_password""

اگر پاسخ زیر را مشاهده کردید بدین معنی است که کانکشن شما به درستی انجام شده است:

### WIFI CONNECTED WIFI GOT IP

برای یافتن IP ماژول خود از دستور زیر بهره میبریم دقت شود که این IP address را یادداشت کرده زیرا در آینده بدان نیاز خواهیم داشت

### "AT + CIFSR"

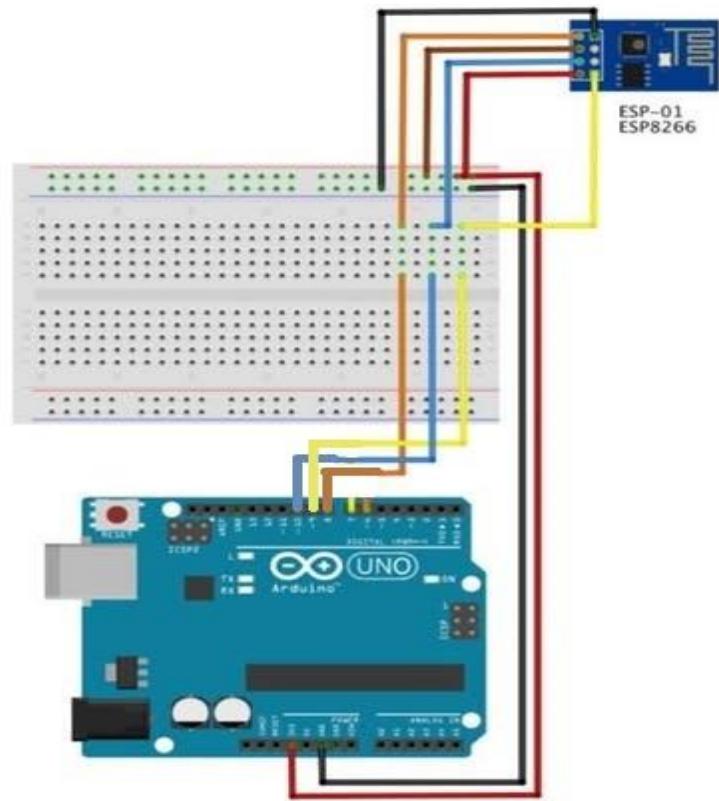
در ماژول کاربردی ما:

+CIFSR:STAIP,"192.168.1.234"  
+CIFSR:STAMAC,"84:f3:eb:90:bf:d0"

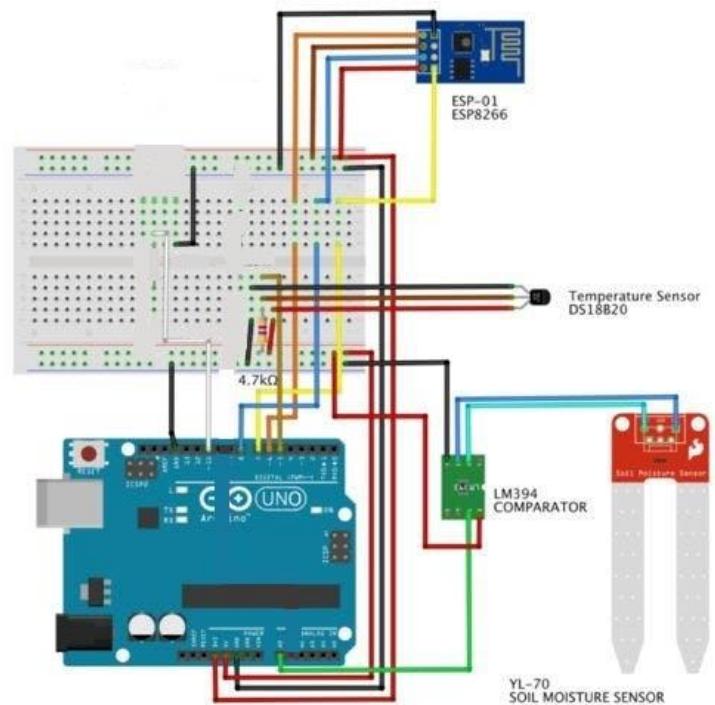
## قدم دوم: تست ماژول ESP-01

پس از تکمیل قدم اول باید مدار نهایی ماژول را ببندیم بدین منظورم سیم کشی ها را به صورت زیر تغییر می دهیم:

- ESP-01 RX to UNO Pin D8 •
- ESP-01 TX to UNO Pin D9 •
- ESP-01 Ch-Pd to Vcc (3.3V) •
- ESP-01 Reset to UNO Pin D10 •
- ESP-01 Vcc to 3.3V •
- ESP-01 Gnd to UNO GND •

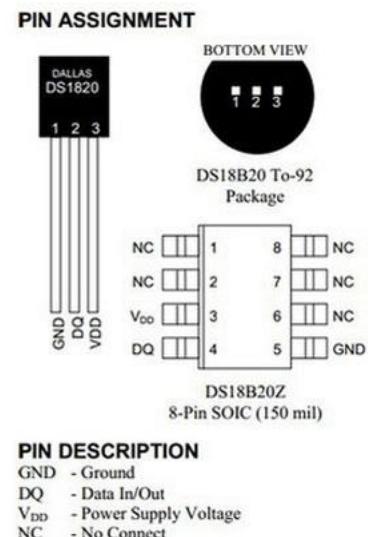
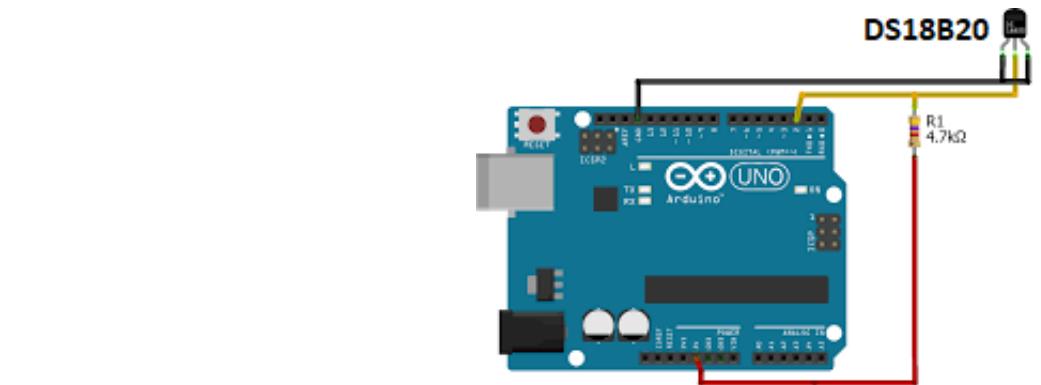


**قدم سوم: اتصال سنسورها و مازول ESP-01**



## سنسور دما:

برای اندازه گیری دمای خاک گیاه از سنسور دما سنج DS18b20 ضد آب با پوشش استیل ضد زنگ استفاده می کنیم. سنسور دما DS18b20 دارای خروجی دیجیتال است و به راحتی تنها با یک مقاومت 4.7 کیلو می توان آنرا به آردوینو متصل کرد. خروجی به صورت دیجیتال و با دقت 12 بیت قابل دسترس است. بنابراین میتوان دما را با دقت 0.0625 درجه سانتیگراد اندازه گیری کرد. سنسور را به صورت زیر به آردوینو متصل میکنیم:



## سنسور رطوبت سنج به همراه چیپ مقایسه گر LM393

همانطور که از نام این سنسور مشخص است ، برای اندازه گیری میزان رطوبت و یا آب موجود در خاک و زمین استفاده می گردد. این سنسور در اصل میزان تشنجی گیاه شما را مشخص می نماید، خروجی آنalog این سنسور در زمانی که میزان رطوبت خاک کم باشد ، مقدار خروجی بالا و زمانی که رطوبت بالا باشد مقدار پایین تری را به پین آنalog آردوینو ارسال می نماید. برای تعیین میزان رطوبت ، خروجی AO باید به ورودی آنalog آردوینو متصل گردد.

سنسور را به صورت زیر به آردوینو متصل میکنیم:

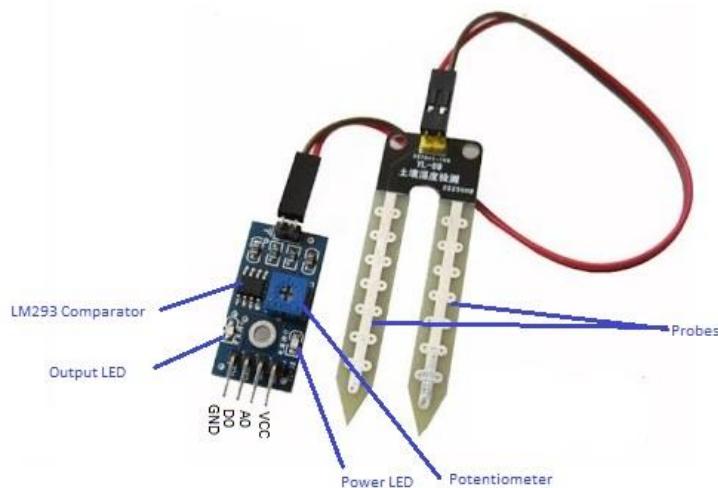
اتصالات:

VCC : اتصال به منبع تغذیه 3.3 تا 5 ولتی

GND : اتصال به GND

DO : خروجی دیجیتال ( 0 یا یک )

AO: خروجی آنالوگ



#### قدم چهارم: ThingSpeak platform

یکی از مهمترین بخش های پروژه ما **ThingSpeak** است که به ما اجازه می دهد جمع آوری، تجزیه و تحلیل داده ها را انجام دهیم. ابتدا وارد آن میشویم و یک اکانت میسازیم سپس یک کانال جدید ایجاد می کنیم که در آن ما 3 سنسور و یک وضعیت اضافی (**spare field status**) داریم:

## Sunlight Tracking Flower Pot

Channel ID: 596372  
Author: fifichh  
Access: Public

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

### Channel Settings

Percentage complete 30%

Channel ID 596372

Name	Sunlight Tracking Flower Pot
Description	
Field 1	light(%)
Field 2	soil humidity
Field 3	soil temperature
Field 4	spare
Field 5	
Field 6	
Field 7	
Field 8	
Metadata	

### Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

### Channel Settings

- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
  - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.
- **Video URL:** If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.
- **Link to GitHub:** If you store your ThingSpeak code on GitHub®, specify the GitHub repository URL.

### Using the Channel

You can get data into a channel from a device, website, or another ThingSpeak channel. You can then visualize data and transform it using [ThingSpeak Apps](#).

See [Tutorial: ThingSpeak and MATLAB](#) for an example of measuring dew point from a weather station that acquires data from an Arduino® device.

[Learn More](#)

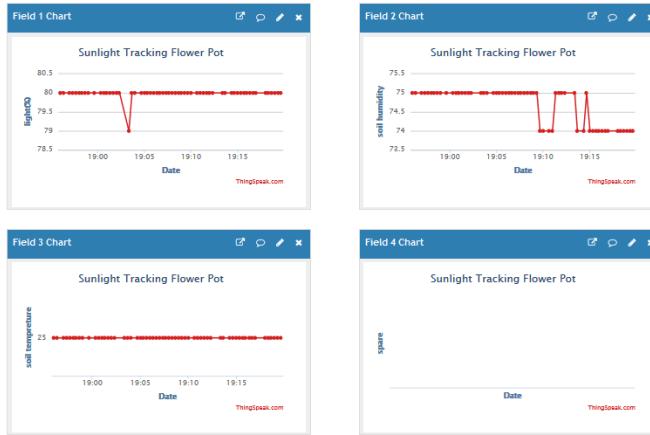
## Sunlight Tracking Flower Pot

Channel ID: 596372  
Author: fifichh  
Access: Public

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export | Add Visualizations | Add Widgets | Export recent data | MATLAB Analysis | MATLAB Visualization

### Channel Stats

Created: 27 days ago  
Updated: 23 days ago  
Last entry: 22 days ago  
Entries: 276



Field 1: Light in %

Field 2: Soil Humidity in %

Field 3: Soil Temperature in oC

Field 4: Battery Charge in %

Write API Key

Key 1Q...

Generate New Write API Key

Read API Keys

Key 0E...

Note

Save Note Delete API Key

Generate New Read API Key

## Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

## API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click [Generate New Write API Key](#).
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click [Generate New Read API Key](#) to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

## API Requests

### Update a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=1Q...&field1=123
```

### Get a Channel Feed

```
GET https://api.thingspeak.com/channels/596372/feeds.json?results=2
```

### Get a Channel Field

```
GET https://api.thingspeak.com/channels/596372/fields/1.json?results=1
```

### Get Channel Status Updates

```
GET https://api.thingspeak.com/channels/596372/status.json
```

## قدم پنجم: کد آردوینو

ارسال اطلاعات سنسورها از آردوینو به پلت فرم **Thingspeak** از طریق مازول وایفای(**Serial Monitor**) آردوینو  
در این مرحله، سرویس **Cloud** ما در دسترس است و همچنین سنسورها اطلاعات را از گیاه دریافت میکنند. هدف ما این است که این اطلاعات را به **ThingSpeak** پلت فرم ارسال کنیم. برای اینکار، باید یک رشته **GET** ارسال شود. که در 3 بخش انجام داده میشود:

1-ابتدا "Start cmd" را ارسال میکنیم:

```
AT+CIPSTART="TCP","184.106.153.149",80
```

این دستور متغیرهایی که در آن استفاده میشوند را به نمایش در میآورد:

[AT+CIPSTART,(“type”),(“ip address”),(port)]

پارامترها:

type : نوع پروتکل که TCP باشد یا UDP

addr : آی پی سرور

port : شماره پورت دستگاهی که می خواهد به ماژول متصل شود.

2- به دنبال آن:

**AT+CIPSEND=116**

مشخص کردن طول دادهای که می خواهیم ارسال کنیم:

AT+CIPSEND=length

3- سپس رشته داده را دریافت میکند و آن را در فیلد مناسب با آن در ThinkSpeak قرار میدهد:

**GET/update?api\_key=YOUR\_WRITE\_KEY\_HERE&field1 light = &field2  
soilHum=0&field3 soilTemp = &field4 spare = <br**

در ادامه کد کلی آردوینو را مشاهده میکنید :

توضیح جزییات به صورت کامنت و روی تصویر قرار گرفته است.

```

1// Thingspeak
2String statusChWriteKey = "10[REDACTED]"; // Status Channel id: 596372
3#include <SoftwareSerial.h>
4SoftwareSerial EspSerial(8, 8); // Rx, Tx
5#define HARDWARE_RESET 10
6// DS18B20
7#include <OneWire.h>
8#include <DallasTemperature.h>
9#define ONE_WIRE_BUS 2 // DS18B20 on pin D2
10OneWire oneWire(ONE_WIRE_BUS);
11DallasTemperature DS18B20(&oneWire);
12int soilTemp = 0;
13#include <stdlib.h>
14// LDR (Light)
15#define ldrPIN 1
16//int light = 0;
17// Soil humidity
18#define soilHumPIN 0
19int soilHum = 0;
20
21// Variables to be used with timers
22long writeTimingSeconds = 17; // ==> Define Sample time in seconds to send data
23long startWriteTiming = 0;
24long elapsedWriteTime = 0;
25
26int spare = 0;
27boolean error;
28void setup()
29{
30  Serial.begin(9600);
31
32  pinMode(HARDWARE_RESET, OUTPUT);
33
34  digitalWrite(HARDWARE_RESET, HIGH);
35
36  DS18B20.begin();
37
38  EspSerial.begin(9600); // Communicacao com Modulo WiFi
39  EspHardwareReset(); //Reset do Modulo WiFi
40  startWriteTiming = millis(); // starting the "program clock"
41}
42void loop()
43{
44  start: //label
45  error=0;
46
47  elapsedWriteTime = millis()-startWriteTiming;
48
49  if (elapsedWriteTime > (writeTimingSeconds*1000))
50  {
51    readSensors();
52    writeThingSpeak();
53    startWriteTiming = millis();
54  }
55
56  if (error==1) //Resend if transmission is not completed
57  {
58    Serial.println(" <<< ERROR >>>");
59    delay (2000);
60    goto start; //go to label "start"
61  }
62
63***** Read Sensors value *****/
64void readSensors(void)
65{
66
67
68 //light = map(analogRead(ldrPIN), 1023, 0, 0, 100); //LDRDark:0 ==> light 100%
69 soilHum = map(analogRead(soilHumPIN), 1023, 0, 0, 100);
70 DS18B20.requestTemperatures();
71 soilTemp = DS18B20.getTempCByIndex(0); // Sensor 0 will capture Soil Temp in Celcius
72
73
74 int val = analogRead(batteryPin); // read the value from the sensor
75 float volts = (val / 1023.0) * referenceVolts; // calculate the ratio Serial.
76 batterycharge = map(volts, 0, 5, 0, 100);

```

 battery charge part

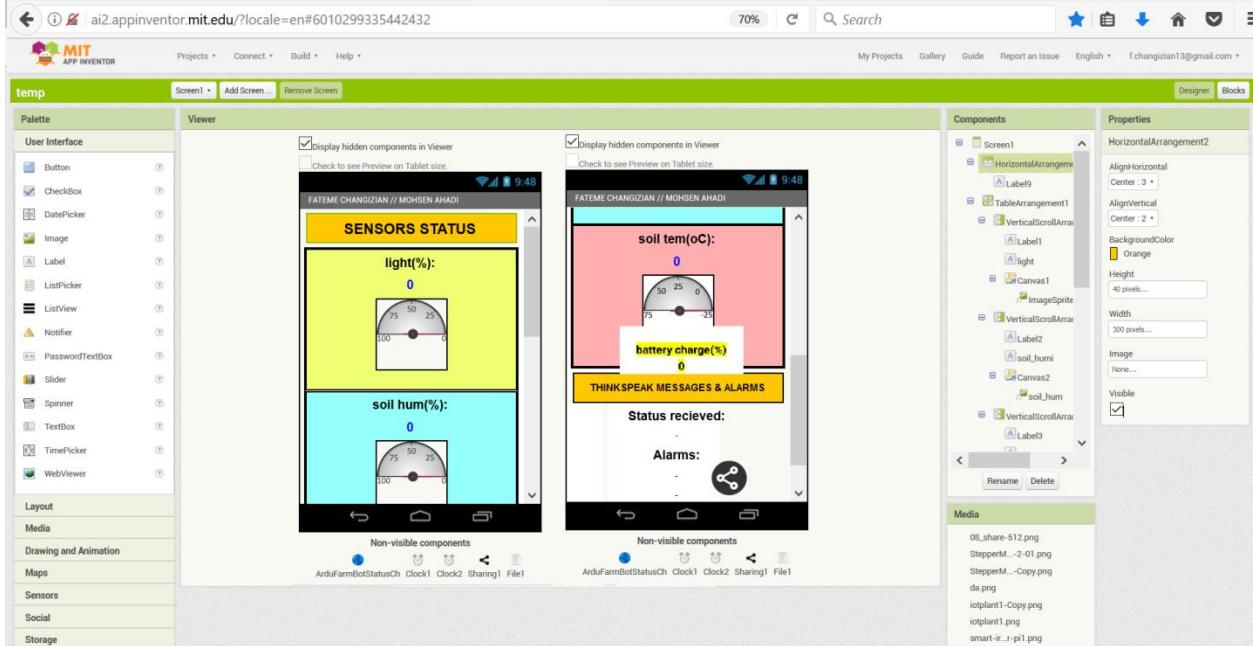
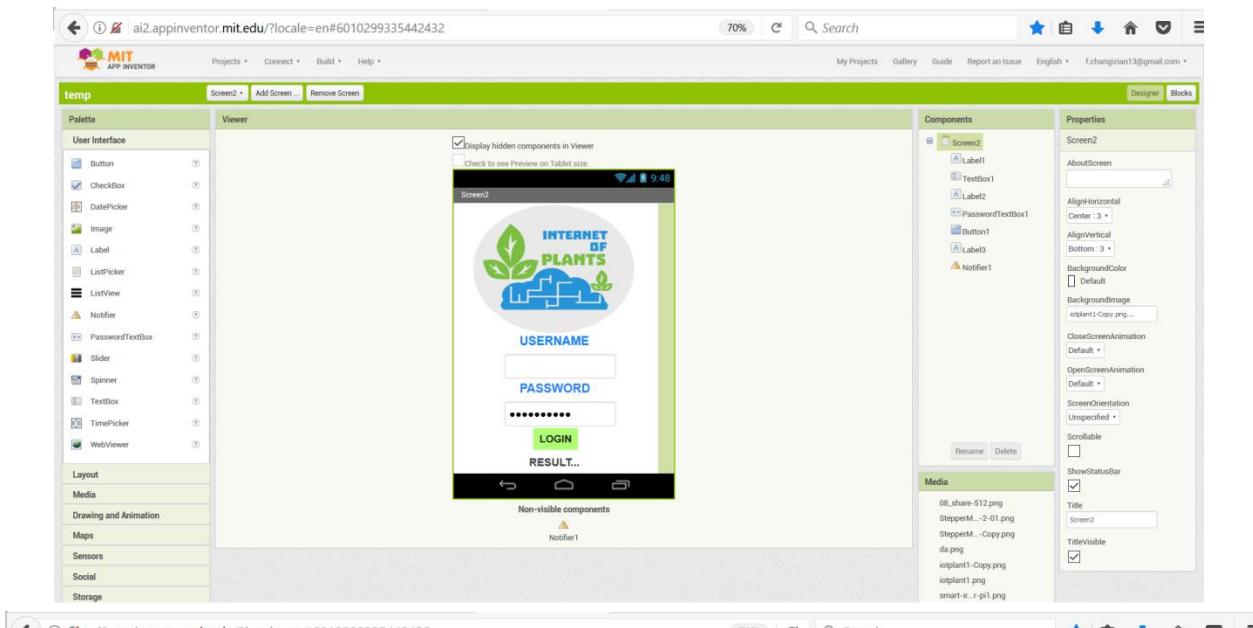
```

73***** Conexao com TCP com Thingspeak *****/
74void writeThingSpeak(void)
75{
76  startThingSpeakCmd();
77  // preparacao da string GET
78  String getStr = "GET /update?api_key=";
79  getStr += statusChWriteKey;
80  //getStr += "field1=";
81  //getStr += String(light);
82  getStr += "field2=";
83  getStr += String(soilHum);
84  getStr += "field3=";
85  getStr += String(soilTemp);
86  getStr += "\r\n\r\n";
87  sendThingSpeakGetCmd(getStr);
88
89***** Reset ESP *****/
90void EspHardwareReset(void)
91{
92  Serial.println("Resetting.....");
93  digitalWrite(HARDWARE_RESET, LOW);
94  delay(500);
95  digitalWrite(HARDWARE_RESET, HIGH);
96  delay(8000); // Time needed to start reading
97  Serial.println("RESET");
98
99***** Start communication with ThingSpeak*****
100void startThingSpeakCmd(void)
101{
102  EspSerial.flush(); // clears the buffer before starting to record
103
104  String cmd = "AT+CIPSTART=\\"TCP\\", \\""; 
105  cmd += "184.106.153.149"; // IP address of api.thingspeak.com
106  cmd += "\", 80";
107  EspSerial.println(cmd);
108  Serial.print("enviado ==> Start cmd: ");
109
110  Serial.println(cmd);
111  if(EspSerial.find("Error"))
112  {
113    Serial.println("AT+CIPSTART error");
114    return;
115  }
116***** send a GET cmd to ThingSpeak *****/
117String sendThingSpeakGetCmd(String getStr)
118{
119  String cmd = "AT+CIPSEND="; 
120  cmd += String(getStr.length());
121  EspSerial.println(cmd);
122  Serial.print("enviado ==> lenght cmd: ");
123  Serial.println(cmd);
124  if(EspSerial.find((char *)">"))
125  {
126    EspSerial.print(getStr);
127
128    Serial.print("enviado ==> getStr: ");
129    Serial.println(getStr);
130    delay(500); // time to process the GET.
131    String messageBody = "";
132    while (EspSerial.available())
133    {
134      String line = EspSerial.readStringUntil('\n');
135      if (line.length() == 1)
136      { //actual content starts after empty line (that has length 1)
137        messageBody = EspSerial.readStringUntil('\n');
138      }
139      Serial.print("MessageBody received: ");
140      Serial.println(messageBody);
141      return messageBody;
142    }
143    else
144    {
145      EspSerial.println("AT+CIPCLOSE"); // alert user close connection
146      Serial.println("ESP8266 CIPSEND ERROR: RESENDING"); // Resend...
147      spare = spare + 1;
148      error=1;
149      return "error";
150    }
151}

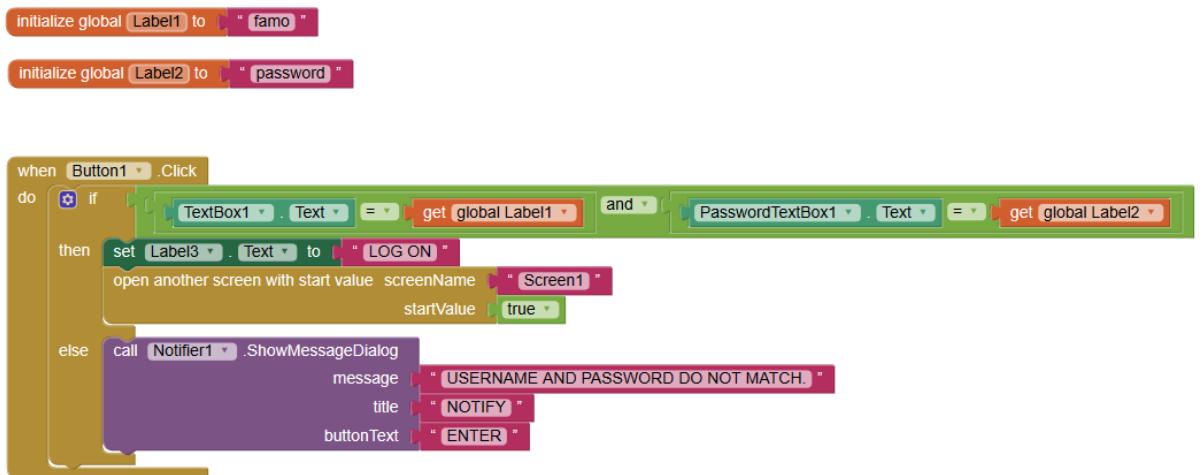
```

## قدم ششم: اپلیکیشن (For Sensor Status Monitoring )FAMO

ابتدا USER INTERFACE و یا رابط کاربری اپلیکیشن را طراحی میکنیم اپلیکیشن بدین صورت است که در ابتدای وارد شدن وارد صفحه LOGIN میشویم و در صورت ورود اطلاعات صحیح وارد صفحه اصلی (نمایش اطلاعات سنسورها) میشویم. پس از ورود به صفحه دوم اطلاعات سه سنسور به صورت عددی و همچنین بر روی یک گیج درجه بندی شده نمایش داده میشود در انتهای وضعیت دریافت داده و دو نوع اخطار یا آلام نمایش داده میشود که توضیحات آن در ادامه گفته میشود سپس در آخر میتوانیم تمام اطلاعات سنسور ها در طول کل کارکرد ربات را در قالب یک فایل تکست به اشتراک بگذاریم.



حال به کد نویسی و بلوک گذاری اپلیکیشن میپردازیم:

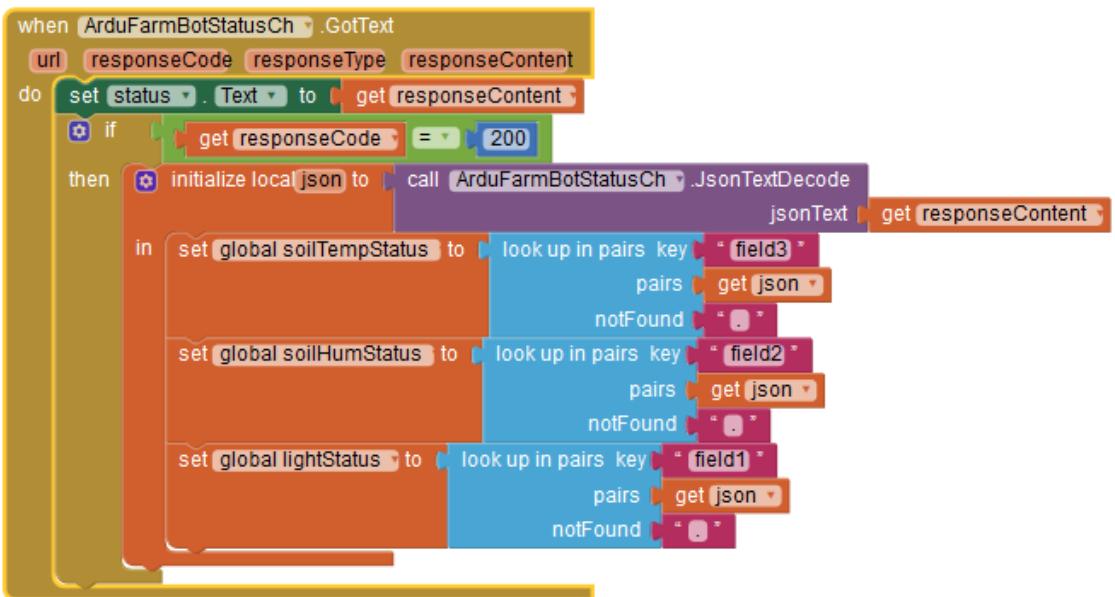


در تصویر بالا ابتدا password و username مورد نظر خودمان را ست میکنیم و با قرار دادن بلوک ها اگر رمز صحیح باشد وارد صفحه بعدی میشویم در غیر این صورت پیام username and password do not match نمایش داده میشود و باید دوباره امتحان کنیم.

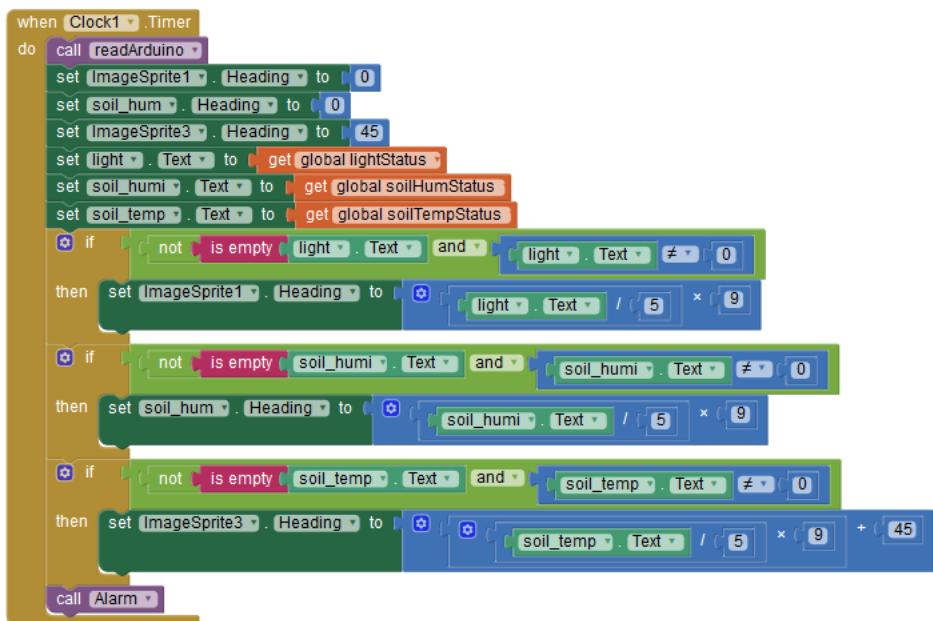
حال پس از ورود به صفحه اصلی :



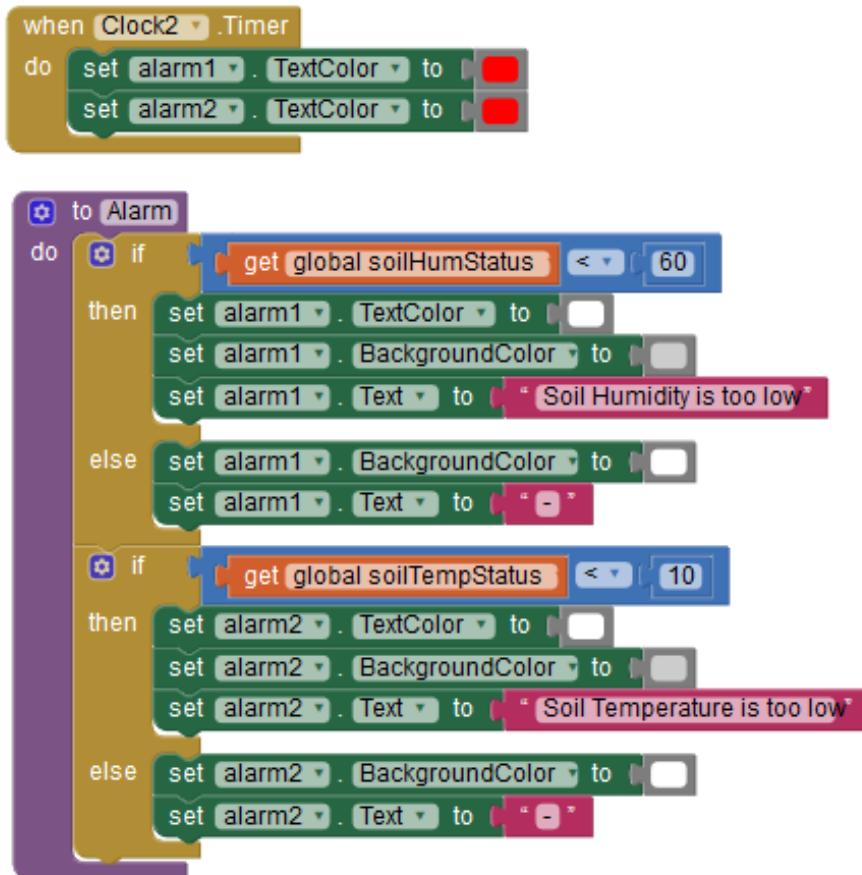
مقادیر سنسور ها را در ابتدا صفر میکنیم و اطلاعات کانال ThingSpeak(READ CHANNEL) را در اختیار app قرار میدهیم. سپس اطلاعات هر فیلد مورد نظر را در اختیار متغیر های تعریف شده در اپلیکیشن قرار میدهیم.



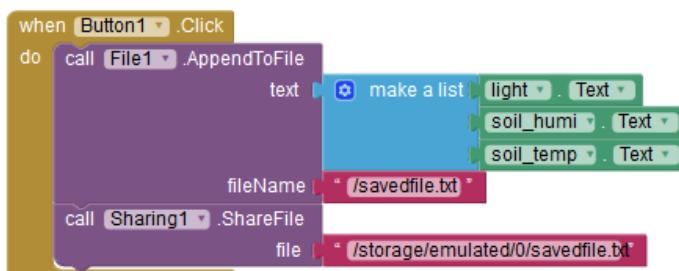
در تصویر زیر ابتدا توسط clock1 تعیین میکنیم که هر چند مدت یکبار اطلاعات سنسور ها را بخوانیم سپس عقره گیج ها بر روی مقادیر صفرشان تنظیم میکنیم و سپس ارتباط بین عدد و عقره گیج را به گونه ای تنظیم میکنیم که درست خود قرار گیرد.



سپس توسط یک کلاک کاری میکنیم که پیام اخطار به صورت چشمک زن نمایش داده شود پیام های آلام عبارتند از پیام های مرتبط با رطوبت خاک و دمای خاک که اگر مقدار آنها از یک مقدار معین کمتر باشد این پیام نمایش داده میشود. (..... is too low)



با فشردن کلید share اطلاعات سنسورها به صورت عددی و در قالب یک فایل تکست(فایل تکست در گوشی موبایل ذخیره شده است) قابل ارسال و به اشتراک گذاری است.



## بخش ساخت افزاری:

در این قسمت از گزارش به توضیح قطعات استفاده شده و کدهای برنامه اردوینو میپردازیم.

### برد اردوینو uno:

مرکر اصلی ارسال، دریافت و محاسبات مربوط به نور محیط، فاصله‌ی مواعن و فرمان‌های مربوط به چرخش ربات در این قسمت از سیستم انجام می‌گیرد. این پردازنده دارای ۳۲ رجیستر ۸ بیتی است که اطلاعات را در خود ذخیره می‌کنند. سرعت پردازش اردوینو UNO ۲۰ مگاهرتر است که به منظور فرمان‌های خواندن و نوشتن این ربات فرکانس مناسبی است.



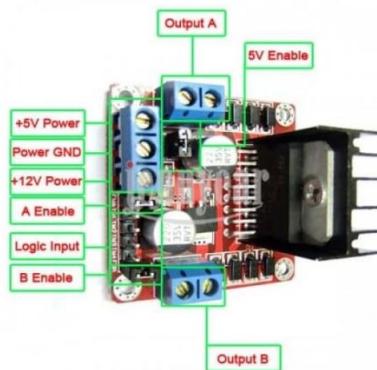
### موتورهای DC و چرخ ها:

برای حرکت ربات از موتورهای DC گیربکس دار استفاده شده که با اتصال به منبع تغذیه با واسطه‌ی موتور درایور قادر هستند ربات را با سرعت‌ها و درجهت‌های دلخواه حرکت دهند.



## موتور درایور : L298N

این کنترل کننده به منظور فرمان های تنظیم سرعت و جهت چرخش در حرکات و توقف های موتور های دی سی است. این درایور دارای دو خروجی برای فرمان به دو موتور دی سی ، ورودی تغذیه ی ۱۲۵ ولتی، فرمان ۵ ولتی و یک ورودی زمین است. با برنامه نویسی روی خروجی های enA و enB طبق عددی بین ۰ تا ۲۵۵ که نشانگر ۸ بیت میباشد میتوان سرعت هر کدام از موتور ها را کنترل کرد.



## موتور servو :

این نوع موتور که در زاویه ای بین ۰ تا ۱۸۰ درجه چرخش میکند برای چرخاندن سنسور اولتراسونیک به اطراف برای نمونه برداری از فواصل اطراف استفاده میشود.



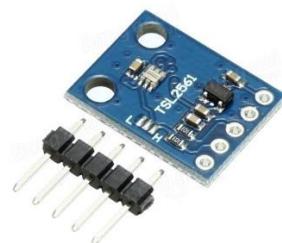
## سنسور فاصله سنج فراصوت:

این فرستنده و گیرنده با استفاده از خاصیت انعکاس امواج فراصوت قادر به اندازه گیری فاصله ی موانع از ربات است. این سنسور با نصب روی سرورو و چرخش به اطراف میتواند از وجود مانع در روبرو، چپ و راست خود مطلع شود.



### سنسور نورسنج :TSL2561

به منظور اندازه گیری دقیق شدت نور محیط بر حسب لوکس از مژول زیر استفاده شده که قابلیت تنظیم فرکانس خواندن نور و بهره‌ی اندازه گیری نور را نیز دارد. ربات در ابتدای کار خود با چرخش‌های ۹۰ درجه‌ای و اندازه گیری مقادیر این سنسور مقصده را برای حرکت خود انتخاب می‌کند.



### بخش برنامه نویسی:

در این قسمت بخش‌های مختلف برنامه و الگوریتم‌های استفاده شده را توضیح خواهیم داد:  
در خطوط اول کتابخانه‌های مربوط به موتور درایور، متغیرهای مورد استفاده در برنامه و مقادیر اولیه آنها تعریف شده که معنی هر کدام از آنها بصورت خط به خط کامنت شده است.

```

1 #include <L298N.h> //adding L298 motor driver lib
2 #include <SparkFunTSL2561.h> //adding TSL2561 lux meter lib
3 #include <Wire.h> //
4 #include <NewPing.h> //adding ultra-sonic lib
5 #include <Servo.h> //adding SG90 servo motor lib
6 #include <SoftwareSerial.h> //adding esp wireless module lib
7 #include <OneWire.h>
8 #include <DallasTemperature.h> //adding temp sesor lib
9 #define ONE_WIRE_BUS 2 // DS18B20 on pin D5
10 #define maxdistance 200 //maximum distance that the ultra-son can read
11 #include <stdlib.h>
12 const int lightThresh=3000; //the amount of sun lux
13 const int distanceThresh=20; //the threshold of distance of an obstacle
14 int distanceTemp=0; //the initial value of temperature
15 // Thingspeak
16 String statusChWriteKey = "1QQ8D4FPB9DA75PA"; // Status Channel id: 596372
17 SoftwareSerial EspSerial(9,8); // Rx, Tx
18 #define HARDWARE_RESET 10
19 // DS18B20
20 // Soil humidity
21 #define soilHumPIN 0
22 int soilHum = 0;
23 OneWire oneWire(ONE_WIRE_BUS);
24 DallasTemperature DS18B20(&oneWire);
25 int soilTemp = 0; //
26 int maxlight; //an integer value to keep the destination's lux number
27 // defines pins numbers
28 int trigPin=A1; //an analog pin used to send ultra-sound
29 int echoPin=A2; //an analog pin to recieve the reflect of the ultra-sound
30 // defines variables
31 long duration;
32 int distance = 100; //the initial value of the forward distance from an obstacle
33 int distanceR; // longest distance from the right side
34 int distanceL; //longest distance from the left side
35 //*****

```

در ادامه توابع سروو موتور و سنسور فراصوت تعریف شده اند و همچنین پین های متصل به موتور دی سی از اردوینو مشخص گردیده اند. جزئیات هر کدام از خطوط مشخص شده است.

```

35 //*****
36
37 // Variables to be used with timers
38 long writeTimingSeconds = 60; // ==> Define Sample time in seconds to send data
39 long startWriteTiming = 0;
40 long elapsedWriteTime = 0;
41
42 int spare = 0;
43 boolean error;
44 *****
45 NewPing sonar(trigPin, echoPin, maxdistance); //defining the ultrasonic function
46 Servo myservo; //defining sg90 servo motor function
47
48 // Create an SFE_TSL2561 object, here called "light":
49
50 SFE_TSL2561 light; //defining the luxmeter function
51
52 // Global variables:
53
54 boolean gain; // Gain setting, 0 = X1, 1 = X16;
55 unsigned int ms; // Integration ("shutter") time in milliseconds
56 double lux; // Resulting lux value
57 int luxnum;
58 //motor variables:
59 const int EnableA = 11; //enableA pin of the L298 connects to digital pin 11 of arduino
60 const int RightMotorForward = 4; //analog pin 4 arduino connects to the right dc motor
61 const int RightMotorBackward = 5; //analog pin 5 arduino connects to the right dc motor
62 const int LeftMotorForward = 7; //analog pin 7 arduino connects to the left dc motor
63 const int LeftMotorBackward = 6; //analog pin 5 arduino connects to the left dc motor
64 const int EnableB = 3; //pin 3 controls the other dc motor as enableB in L298 driver
65
66 L298N motorA(EnableA, RightMotorForward, RightMotorBackward); //difining L298 function
67 L298N motorB(EnableB, LeftMotorForward, LeftMotorBackward);
68
69 *****

```

در ادامه‌ی کد که قسمت `setup` برنامه است هر فرمانی که نوشته شود فقط یک بار بعد از روشن شدن ربات اجرا خواهد شد. در ابتدا قسمت مانیتورینگ اردوینو بعنوان `serial print` راه اندازی و سپس پین‌هایی از اردوینو که به موتور از طریق درایور متصل شده‌اند به عنوان خروجی مشخص می‌شوند. بعد از آن موتور سرورو در زاویه‌ی ۹۰ درجه یعنی حالتی که سنسور اولتراسونیک نصب شده روی آن به رویرو نگاه کند تنظیم شده و سنسور اقدام به خواندن فاصله‌ی روبروی خود می‌کند. برای رسیدن به مقدار دقیق تر و بدون خطایی از فاصله، مقدار سنسور را ۴ بار خوانده و سپس در رجیستر مربوطه ذخیره می‌کنیم. در خطوط بعدی دو پین انالوگ اردوینو که بعنوان گیرنده و فرستنده‌ی اولتراسونیک مشخص شده‌اند را بعنوان ورودی و خروجی تعریف کردیم. پس از آن فرکانس نمونه برداری سنسور نورسنج را تنظیم کرده و تابع `()turnAround` را اجرا می‌کنیم. این تابع که بعدا با جزئیات توضیح داده خواهد شد وظیفه‌ی چرخش به ۴ سمت و نمونه برداری از نور اطراف خود و رفتن به سمت بیشترین نور را بر عهده دارد.

```

69 void setup() {
70     Serial.begin(9600);
71     //
72     pinMode(HARDWARE_RESET, OUTPUT);
73
74     digitalWrite(HARDWARE_RESET, HIGH);
75
76     DS18B20.begin();
77
78     EspSerial.begin(9600); // Comunicacao com Modulo WiFi
79     EspHardwareReset(); //Reset do Modulo WiFi
80     startWriteTiming = millis(); // starting the "program clock"
81     //motor setup
82     pinMode (RightMotorForward, OUTPUT);
83     pinMode (RightMotorBackward, OUTPUT);
84     pinMode (LeftMotorForward, OUTPUT);
85     pinMode (LeftMotorBackward, OUTPUT);
86     pinMode (EnableA, OUTPUT);
87     pinMode (EnableB, OUTPUT);
88     motorA.setSpeed(175); // an integer between 0 and 255
89     motorB.setSpeed(175); // an integer between 0 and 255
90     myservo.attach(12);
91     myservo.write(90);
92     distance = readPing();
93     delay(100);
94     distance = readPing();
95     delay(100);
96     distance = readPing();
97     delay(100);
98     distance = readPing();
99     delay(100);
100
101    pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
102    pinMode(echoPin, INPUT); // Sets the echoPin as an Input
103

```

```

103
104     Serial.begin(9600);
105     //Serial.println("TSL2561 example sketch");
106     light.begin();
107     gain = 0;
108     // If time = 0, integration will be 13.7ms
109     // If time = 1, integration will be 101ms
110     // If time = 2, integration will be 402ms
111     // If time = 3, use manual start / stop to perform your own integration
112     unsigned char time = 1;
113     //Serial.println("Set timing...");
114     light.setTiming(gain,time,ms);
115     // To start taking measurements, power up the sensor:
116     Serial.println("Powerup...");
117     light.setPowerUp();
118     //
119     //
120     turnAround() ;
121
122 }
123
124
125 //*****

```

قسمت بعدی مهم برنامه که دائما تکرار میشود قسمت `loop()` است که دارای کد مربوط به ارسال داده و تابع `avoidObstacle()` است که بعد از این توضیح داده خواهد شد. این تابع وظیفه‌ی تشخیص وجود مانع بر سر راه ربات و دور زدن آن مانع را برعهده دارد.

```

126
127 void loop() {
128     start: //label
129     error=0;
130
131     elapsedWriteTime = millis()-startWriteTiming;
132
133     if (elapsedWriteTime > (writeTimingSeconds*1000))
134     {
135         readSensors();
136         writeThingSpeak();
137         startWriteTiming = millis();
138     }
139
140     if (error==1) //Resend if transmission is not completed
141     {
142         Serial.println(" <<< ERROR >>>");
143         delay (2000);
144         goto start; //go to label "start"
145     }
146
147     //
148     avoidObstacle();
149
150     if(luxnum >= lightThresh){
151
152         moveStop();
153         delay(10000);
154         turnAround();
155     }
156     else{}
157
158 }
159 //*****

```

پس از این توابع مختلف حرکات موتور دی سی و موتور سرورو بصورت جدا تعریف شده و در تابع دور زدن مانع استفاده شده اند. برای توضیحات مربوط به این توابع خطوط برنامه کامنت شده اند و در این گزارش به تعریف توابع مهمتر و الگوریتم‌های استفاده شده میپردازیم

## turnAround()

این تابع به این صورت عمل میکند که ابتدا با دور زدن به ۴ طرف خود با زاویه‌ی ۹۰ درجه میچرخد و هر بار عددی از نور محیط را بر میگرداند. سپس با استفاده از تابعی اماده در نرم افزار اردوینو بیشترین مقدار از این ۴ متغیر را پیدا میکند و جهت این نور را بعنوان مقصد خود بر میگزیند. بعد از آن ربات به سمت این مقصد شروع به حرکت میکند.

```
357 void turnAround(){
358     int LUX1, LUX2, LUX3, LUX4 ;           393 //fourth Turn
359     //First Turn                         394 turnRight();
360     motorA.forward();                   395 delay(600);
361     motorB.forward();                   396 moveStop();
362     delay(1);                           397 delay(2000);
363     turnRight();                        398 getLight();
364     delay(600);                         399 LUX4=luxnum;
365     moveStop();                         400 Serial.println("LUX4: ");
366     delay(2000);                        401 Serial.println(LUX4);
367     getLight();                          402 //
368     LUX1= luxnum;                      403 int maxlight = max(max(LUX1,LUX2), max(LUX3,LUX4));
369     Serial.println("LUX1: ");            404 Serial.print("Maxlight is ");
370     Serial.println(LUX1);               405 Serial.println(maxlight);
371                                         406 //
372     //second turn                       407
373     turnRight();                        408 if (maxlight==LUX1){
374     delay(600);                         409     turnRight();
375     moveStop();                         410     delay(600);
376     delay(2000);                        411     moveStop();
377     getLight();                          412     delay(100);
378     LUX2=luxnum;                      413     moveForward();
379     Serial.println("LUX2: ");            414 }
380     Serial.println(LUX2);               415 //
381                                         416 else if (maxlight==LUX2){
382     //third turn                        417     turnRight();
383     turnRight();                        418     delay(1200);
384     delay(600);                         419     moveStop();
385     moveStop();                         420     delay(100);
386     delay(2000);                        421     moveForward();
387     getLight();                          422 }
388     LUX3=luxnum;                      423 //
389     Serial.println("LUX3: ");            424 else if (maxlight==LUX3){
390     Serial.println(LUX3);               425     turnRight();
391                                         426     delay(1800);
392                                         427     moveStop();
```

## avoidObstacle()

این تابع که وظیفه‌ی تشخیص و عبور از مانع را برعهده دارد ابتدا در صورت مشاهده‌ی هر گونه مانع با استفاده از سنسور اولتراسونیک در فاصله‌ی حداقل ۲۰ سانتی‌متری خود، دستور به توقف موتور‌ها میدهد. پس از آن سنسور فاصله‌ی سنج به چپ و راست می‌چرخد تا فاصله‌ی اطراف خود را نیز به دست آورد. در این قسمت سه وجود خواهد داشت که عبارتند از: ۱-فاصله‌ی چپ از راست بیشتر باشد، ۲-فاصله‌ی راست از چپ بیشتر باشد، ۳-فاصله‌ی چپ و راست مساوی باشد

اگر ربات در هر کدام از این حالت‌های fsm قرار گیرد، با انتخاب فاصله‌ی بیشتر به سمت آن می‌چرخد اما سنسور اولتراسونیک در خلاف جهت موتور‌ها می‌چرخد و شروع به اندازه گیری فاصله‌ی تا مانع می‌کند.

تا زمانی که فاصله‌ی ربات از مانع کمتر از فاصله‌ی تعريف شده بعنوان threshold باقی بماند، دستور به حرکت ربات در راستای مانع میدهیم، به محض اینکه ربات ما مانع را رد کند، سنسور اولتراسونیک فاصله‌ای بیشتر از مقدار حداقلی را اندازه می‌گیرد. بعد از آن دستور داده می‌شود که ربات ۹۰ درجه در جهت خلاف قبل (یعنی به سمت اولیه که مقصد نهایی بوده) بچرخد و به حرکت خود ادامه دهد.

```

517 void avoidObstacle(){
518     distance=readPing();
519     Serial.println("distanceForward: ");
520     Serial.println(distance);
521     if (distance <= distanceThresh){
522         moveStop();
523         delay(300);
524         distanceR=lookRight();
525         delay(500);
526         Serial.println("distanceRight: ");
527         Serial.println(distanceR);
528         distanceL=lookLeft();
529         delay(500);
530         Serial.println("distanceLeft: ");
531         Serial.println(distanceL);
532
533         if(distanceR > distanceL){
534             turnRight();
535             delay(600);
536             moveStop();
537             delay(300);
538             distanceTemp=lookLeft();
539             delay(500);
540             Serial.println("distanceL: ");
541             Serial.println(distanceTemp);
542             if(distanceTemp <= 30){
543                 moveForward();
544                 distanceTemp=lookLeft();
545                 delay(500);
546             }
547             if(distanceTemp > 30){
548                 moveStop();
549                 delay(300);
550                 myservo.write(90);
551             }
552         }
553         turnLeft();
554         delay(600);
555         moveStop();
556         delay(300);
557         moveForward();
558     }
559
560     else if (distanceL > distanceR){
561         turnLeft();
562         delay(600);
563         moveStop();
564         delay(300);
565         distanceTemp=lookRight();
566         delay(500);
567         Serial.println("distanceR: ");
568         Serial.println(distanceTemp);
569         if(distanceTemp <= 30){
570             moveForward();
571             distanceTemp=lookRight();
572             delay(200);
573         }
574         if(distanceTemp > 30){
575             moveStop();
576             delay(300);
577             myservo.write(90);
578             turnRight();
579             delay(600);
580             moveStop();
581             delay(300);
582             moveForward();
583         }
584     }
585 }

```

```

586 }  

587     else if(distanceR == distanceL){  

588         turnLeft();  

589         delay(600);  

590         moveStop();  

591         delay(300);  

592         distanceTemp=lookRight();  

593         delay(200);  

594         Serial.println("distanceR: ");  

595         Serial.println(distanceTemp);  

596         if(distanceTemp <= 30){  

597             moveForward();  

598             distanceTemp=lookRight();  

599             delay(200);  

600         }  

601         else if(distanceTemp > 30){  

602             moveStop();  

603             delay(300);  

604             myservo.write(90);  

605             turnRight();  

606             delay(600);  

607             moveStop();  

608             delay(300);  

609             moveForward();  

610         }  

611     }  

612  

613  

614  

615 }  

616     moveForward();  

617 }  

618 }  

619 }  


```

## منابع استفاده شده در برنامه:

منبع توابع و کتابخانه های استفاده شده در این بخش از لینک های زیر قابل دسترسی هستند:

۱- کتابخانه موتور درایور <L298.h> :

<https://www.arduinolibraries.info/libraries/l298-n>

۲- کتابخانه تابع سنسور دما <dallastemperature.h> و <onewire.h>

<https://www.arduinolibraries.info/libraries/one-wire>

<https://www.arduinolibraries.info/libraries/dallas-temperature>

۳- کتابخانه ی تابع سنسور نور <newping.h>

<https://playground.arduino.cc/Code/NewPing>

۴- کتابخانه ی ماژول esp8266 <softwareserial.h>

<https://www.arduino.cc/en/Reference/SoftwareSerial>

۵- کتابخانه ی توابع سرو موتور <servo.h>

<https://www.arduinolibraries.info/libraries/servo>

۶- راهنمای تابع max()

<https://www.arduino.cc/reference/en/language/functions/math/max/>

۷- راهنمای تابع Serial.begin()

<https://www.arduino.cc/en/Serial/Begin>

۸- راهنمای تابع goto()

<https://www.arduino.cc/reference/en/language/structure/control-structure/goto/>

۹- سایت app inverter برای طراحی اپلیکیشن اندروید:

<http://appinventor.mit.edu/explore/>

۱۰- سایت پلتفرم thingspeak

<https://thingspeak.com>

۱۱- آموزش کار با ماژول esp8266-01

<http://digispark.ir>

۱۲- کتاب آموزش کار با اردوینو

<http://libgen.io/book/index.php?md5=DE225E18C9BEB55C0984081F7FF0A140>

۱۳- کتاب آموزش کار با برنامه‌ی app inverter

<http://libgen.io/book/index.php?md5=3A351B5DBBC836DED4A9E0E469717A8C>

## تخمین هزینه های ساخت ربات:

قطعات استفاده شده در این ربات به همراه قیمت انها به شرح زیر میباشد:

نام قطعه	مدل قطعه	قیمت(تومان)
شاسی ربات و دو موتور		۴۶۸۰۰
بورد اردوینو	UNO-R3	۵۷۰۰۰
ماژول نورسنج	TSL2561	۱۴۷۰۰
موتور درایور	L298N	۲۱۲۰۰
ماژول وایفای	ESP8266-01	۳۶۰۰۰
ماژول اولتراسونیک	SRF-04	۳۶۰۰۰
منبع تغذیه	POWER BANK REMAX	۳۲۵۰۰
سیم های جامپر	MM/FM/FF	۱۴,۰۰۰
پیچ و مهره و پایه		۷۲۰۰
سنسور دما با شیلد عایق	DS18B20	۳۰۰۰۰
سنسور رطوبت	YL-69 + LM393	۱۵۰۰۰
سرво موتور	SG90	۱۹۵۰۰
طبقات مشبک پلاستیکی		۷۰۰۰
برد بورد کوچک		۵۰۰۰
<b>جمع کل</b>	.....	<b>۳۴۱۹۰۰</b>

تصاویر واقعی از ربات:

