

بسم الله الرحمن الرحيم

گزارش بخش عملی پروژه ابزار دقیق

موضوع پروژه: کنترل و مراقبت از گیاهان آپارتمانی

استاد: دکتر افشار

اعضای گروه:

ریحانه آهنی 9823009

فاطمه رفیعی 9823039

سمیرا سلجوقی 9823048

گروه 7

فهرست

فصل اول: گزارش انتخاب یا ساخت المان ها و اجزای سخت افزاری سیستم.....	4
1-1 مشخصات اصلی اجزای مدار:	4
1-1-1 سنسور نور (LDR MLG5516):	4
2-1-1 سنسور دما (LM335):	5
3-1-1 سنسور رطوبت (YL-69):	6
4-1-1 برد آردوینو (Arduino UNO):	9
5-1-1 ماژول وای فای (ESP8266-01):	12
2-1-2 بروشور و دیتاشیت های المان های مدار:	14
3-1-2 عکس از اجزاء سیستم:	15
4-1-2 گزارش ساخت سیستم:	16
5-1-2 گزارش تست سیستم:	19
فصل دوم: گزارش نرم افزار های نگارش و استفاده شده	20
2-1 اصل و توضیح بخش های مختلف کد به همراه عکس صفحات کاربر:	20
2-1-2 ارتباط میان آردوینو و سنسور ها:	20
2-2-2 ارتباط میان سنسور YL-69:	21
3-1-2 ارتباط میان سنسور نور LDR:	21
4-1-2 ارتباط آردوینو با تایمر داخلی آن:	22
5-1-2 ارتباط ESP و آردوینو:	24
6-1-2 ارتباط میان سرور و ESP:	26
7-1-2 ارتباط میان سرور و دیتابیس:	27
8-1-2 ارتباط میان سرور و محیط کاربر:	28
2-2 پارامتر های مورد نیاز سرور:	35
3-2 توضیحات کد به صورت جزئی تر:	36
4-2 نرم افزار های برای استفاده از کد:	42
5-2 نحوه تست عملکرد نرم افزارها:	43
6-2 مرجع و لینک کد:	45

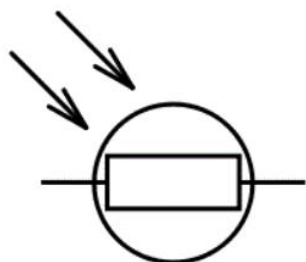
46	فصل سوم: طراحی و شبیه سازی در پروتئوس
46	3-1 طراحی مدار:
47	3-2 شبیه سازی مدار:
48	3-3 مرجع کد و فایل شبیه سازی:
49	فصل چهارم: لینک کلیپ های تهیه شده

فصل اول: گزارش انتخاب یا ساخت المان ها و اجزای سخت افزاری سیستم

1-1 مشخصات اصلی اجزای مدار:

1-1-1 سنسور نور (LDR MLG5516):

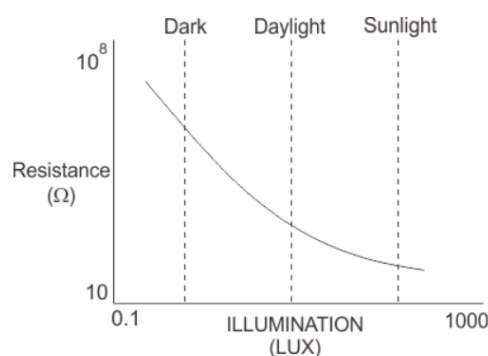
مقاومت های نوری (LDR) ابزاری است که مقاومت آن تابعی از تابش الکترومغناطیسی تصادفی است. از این رو، آن ها دستگاه های حساس به نور هستند. آن ها همچنین به عنوان رسانا های نوری، سلول های رسانای نوری یا بطور ساده تر سلول های نوری نامیده می شوند LDR. مخفف Light Dependent Resistor یا همان مقاومت حساس به نور است. آن ها از مواد نیمه هادی ساخته شده اند که مقاومت الکتریکی بالایی دارند. علامت های مختلفی برای نشان دادن مقاومت نوری یا LDR وجود دارد، یک نماد رایج در شکل زیر نشان داده شده است. فلش نشان دهنده سقوط نور بر روی آن است.



نماد LDR در مدار



مقاومت های نوری (LDR) وسیله های وابسته به نور هستند که با تابش نور بر روی آن ها مقاومت آن ها کاهش می یابد و در تاریکی افزایش می یابد. شکل زیر منحنی مقاومت در برابر روشنایی را برای یک LDR خاص نشان می دهد.



فوتوسل ها یا LDR ها ابزار های غیرخطی هستند. حساسیت آن ها با طول موج تابش نور روی آن ها متفاوت است. برخی از سلول های نوری ممکن است به طیف خاصی از طول موج پاسخ ندهند. بر اساس ماده ای که برای سلول های مختلف استفاده می شود منحنی های واکنش طیفی متفاوت دارند. همچنین حساسیت LDR نسبت به فوتودیود ها و ترانزیستور های نوری کمتر است. دسترسی پذیری این سنسور بالا است و به راحتی و به قیمت مناسب قابل تهیه است.

شرکت سازنده این سنسور Mobicon Electronic Component است و قیمت آن حدود 8000 ریال است.

مشخصات مهم سنسور نوری استفاده شده در پروژه در جدول زیر نمایش داده شده است:

Model NO.	Dimension	Voltage applied	Power dissipation	Ambient temperature	Light resistance		Dark resistance	Peak sensitivity wavelength	Rise response time	Fall response time
	mm	Vdc max	mW max	°C	10 Lux(kΩ)	100 Lux(kΩ)	MΩ	nm	ms	ms
MLG5516	5	100	50	90	5-10	1-2	0.5	540	30	40

1-2-1 سنسور دما (LM335):

سنسور دما ابزاری ساده است که درجه گرما یا برودت را اندازه گیری کرده و آن را به یک واحد قابل خواندن تبدیل می نماید ، سنسور دما دستگاهی است که به طور معمول، یک ترموکوپل یا یک آشکارساز دمای مقاومت است که اندازه گیری دما را به شکل قابل خواندن از طریق سیگنال الکتریکی فراهم می کند . کارکرد یک دما سنج به ولتاژ دیود بستگی دارد. تغییر دما مستقیماً با مقاومت دیود متناسب است. هرچه دما سردتر باشد، مقاومت آن کمتر خواهد بود و بالعکس. مقاومت در برابر دیود اندازه گیری شده و به واحد قابل خواندن دما (فارنهایت، سانتیگراد و غیره) تبدیل می شود و به صورت عددی بر روی واحدهای بازخوانی نمایش داده می شود. در زمینه نظارت ژئوتکنیکی، این سنسورهای دما برای اندازه گیری دمای داخلی سازه ها مانند پل ها، سدها، ساختمان ها، نیروگاه ها و غیره استفاده می شوند.



در این پروژه برای اندازه گیری دما از سنسور LM335 استفاده کردیم که دارای امپدانس دینامیکی خروجی خیلی کم است (کمتر از 1 اهم) و رنج کاری 40- تا 100+ درجه دارد. این سنسور یک سنسور دمای غیر خطی با دقت بالاست که به راحتی کالیبره می شود. ولتاژ شکست این سنسور متناسب با دمای مطلق در 10 میلی ولت بر درجه کلون است. در مدار این سنسور یک مقاومت دینامیکی کمتر از یک اهم وجود دارد که با یک رنج جریان 0.45 تا 5 میلی آمپر بدون نوسان یا تغییر در مشخصات سنسور کار می کند. همچنین دسترسی پذیری این سنسور بالا است و به راحتی و به قیمت مناسب قابل تهیه است.

شرکت سازنده این سنسور SGS-THOMSON microelectronics است و قیمت آن حدود 180000 ریال است.

مشخصات مهم سنسور دما استفاده شده در پروژه در جدول زیر نمایش داده شده است:

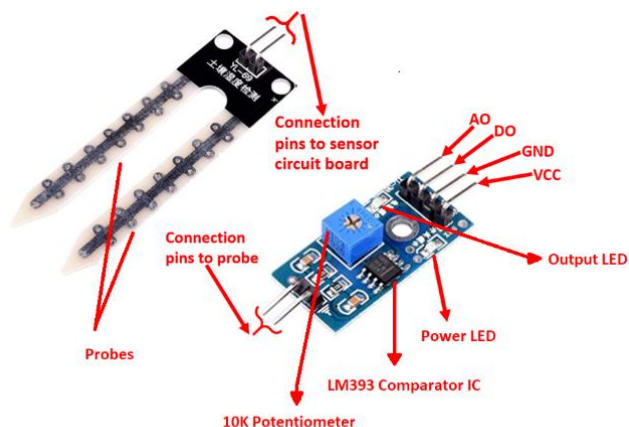
Model NO.	Forward current	Reverse current	Operating temperature range	Operating output voltage(V)			Non-linearity (°C)		Dynamic impedance	Output voltage temperature drift
	mA	mA	°C	Min.	Typ.	Max.	Typ.	Max.	Ω	mV/°C
LM335	10	15	-40 to +100	2.92	2.98	3.04	0.3	1.5	0.6	+10

1-1-3 سنسور رطوبت (YL-69):

از سنسور رطوبت سنج خاک مدل YL-69 ، می توان برای کنترل رطوبت خاک گیاهان و ساخت یک سیستم آبیاری خودکار بهره گرفت. این سنسور متشکل از یک مازول الکترونیکی و یک پراب چنگالی شکل برای تشخیص میزان آب موجود در خاک است. سنسور رطوبت سنج خاک دارای دو خروجی آنالوگ و دیجیتال است، بنابراین در دو مد دیجیتال و آنالوگ قابل استفاده می باشد.

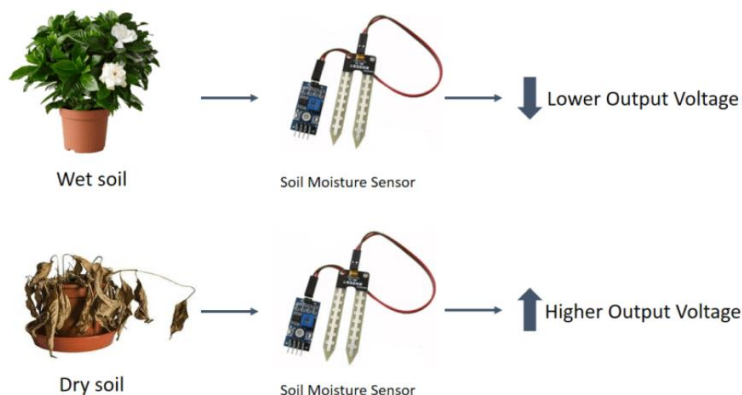
پراب چنگالی شکل سنسور دارای دو رسانای بدون پوشش است که درون خاک و یا هر چیز دیگری که قصد اندازه گیری آب داخل آن را داشته باشید، قرار می گیرد. این پراب شبیه به یک مقاومت متغیر عمل می کند به طوری که مقدار این مقاومت بسته به میزان رطوبت موجود در خاک تغییر خواهد کرد. آب بیشتر باعث می شود که خاک به راحتی جریان الکتریسیته را هدایت کند (مقاومت کمتر)، در حالی که خاک خشک جریان الکتریکی ضعیفی را هدایت می کند (مقاومت بیشتر). ولتاژ خروجی سنسور متناسب با محتوای آب در خاک تغییر می کند. به این صورت که زمانی که خاک خشک است، ولتاژ کاهش می یابد و برعکس.

مقدار این ولتاژ در پایه‌ی خروجی آنالوگ ماژول (AO) قابل دسترس است. LM393 که یک مقایسه کننده با دقت بالا است، سیگنال را به مقدار دیجیتال تبدیل می‌کند که از طریق پایه‌ی DO ماژول در دسترس قرار می‌گیرد. بر روی این ماژول یک پتانسیومتر برای تنظیم حساسیت خروجی دیجیتال (DO) قرار داده شده است. بنابراین حد آستانه رطوبت را می‌توان با استفاده از این پتانسیومتر تنظیم نمود.



خروجی می‌تواند به صورت دیجیتال DO باشد به نحوی که اگر مقدار رطوبت از ترشلد سنسور فراتر رود خروجی به صورت LOW و در غیر این صورت HIGH خواهد بود. همچنین خروجی می‌تواند به صورت آنالوگ AO (بین صفر تا 1023) باشد.

از مشکلات معمول سنسورهای رطوبت‌سنج خاک می‌توان به طول عمر کوتاه آن‌ها به دلیل قرارگرفتن در معرض محیط‌های مرطوب اشاره کرد. یکی از راه‌های افزایش طول عمر این سنسور این است که تنها در زمان خواندن مقدار رطوبت خاک آن را روشن کنید، به عبارت دیگر از اتصال دائم تغذیه به سنسور اجتناب کنید، در غیر این صورت سرعت خوردگی به طور قابل ملاحظه‌ای افزایش می‌یابد. اتصال پایه‌ی VCC ماژول سنسور به یک پایه‌ی دیجیتال آردوینو و تنظیم آن بصورت High یا Low براساس نیاز، یکی از روش‌های آسان انجام این کار است.



مشخصات مهم سنسور رطوبت خاک:

- حساسیت قابل تنظیم از طریق پتانسیومتر آبی رنگ موجود بر روی برد

- ولتاژ عملیاتی بین 3.3 تا 5 ولت

- دارای خروجی دیجیتال و آنالوگ

- طراحی فیزیکی مناسب برای نصب راحت و آسان

- اندازه کوچک (3 در 1.6 سانتیمتر)

- LED نماینگر پاور (قرمز) و سوئیچینگ خروجی دیجیتال (سبز)

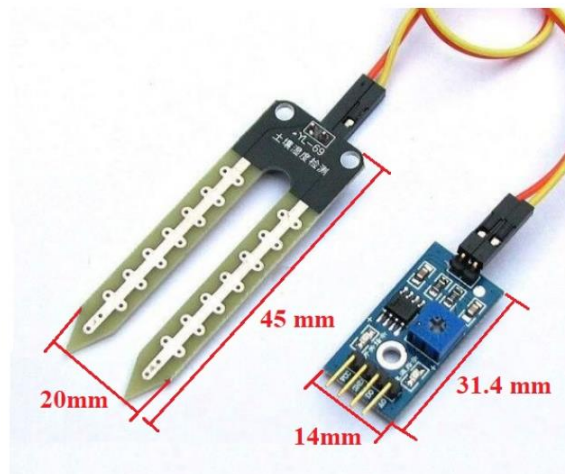
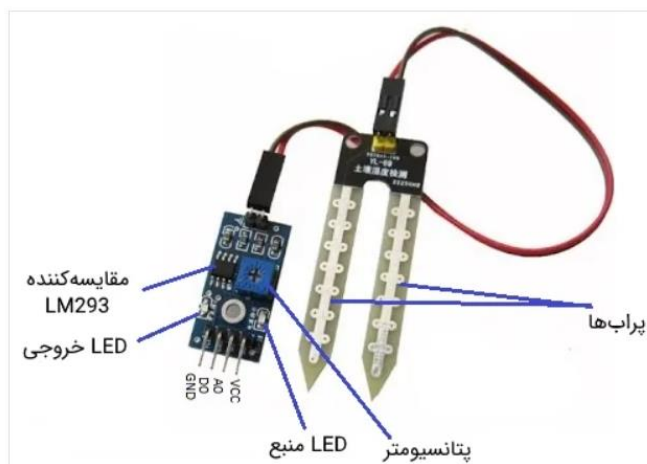
- چیپ مقایسه گر LM393 با پایداری بالا

- VCC اتصال به منبع تغذیه 3.3 تا 5 ولتی

- DO- خروجی دیجیتال (0 یا یک)

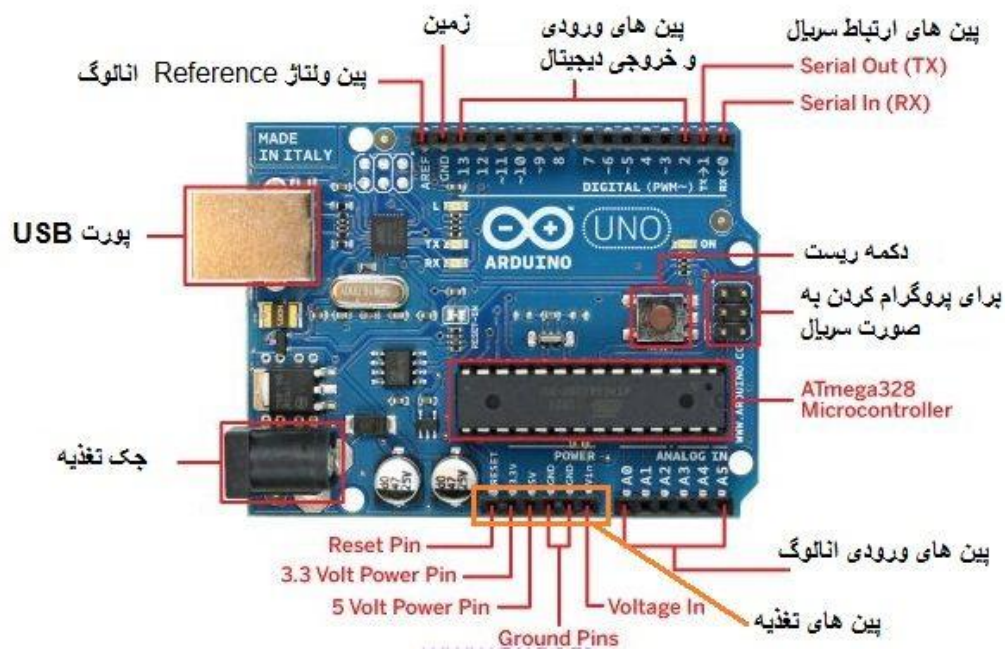
- AO- خروجی آنالوگ (0-4.2 ولت)

برند این سنسور Flying Fish Sensors است و قیمت آن حدود 190000 ریال است.



1-1-4 برد آردوینو (Arduino UNO):

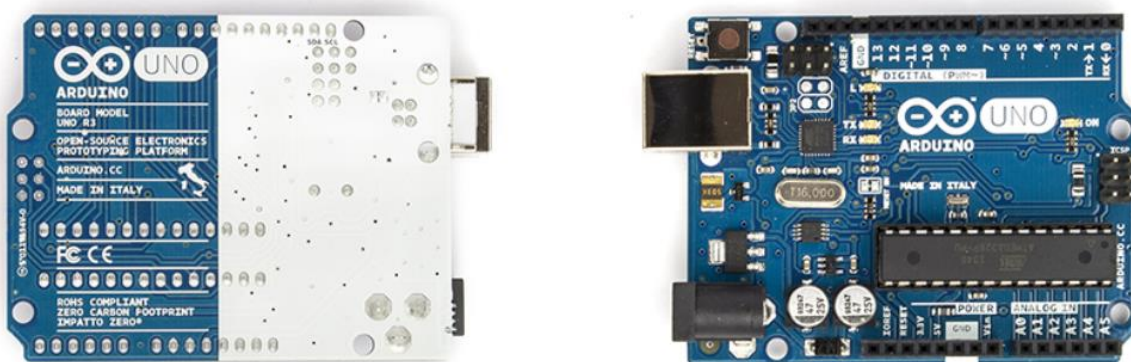
برد Arduino UNO بر پایه ATmega328P می باشد. این برد دارای 14 پین ورودی و خروجی دیجیتال (I/O) می باشد (که 6 عدد آن می تواند به عنوان خروجی PWM استفاده نمود این پین ها با علامت ~ بر روی برد مشخص شده اند) و 6 پین ورودی آنالوگ (Analog Input A0-A5) می باشد. همچنین برد آردوینو UNO دارای کریستال 16MHz، اتصال USB، ورودی تغذیه مجزا، کانکتور ICSP و کلید ریست می باشد که در تصویر زیر مشاهده می کنید.



کانکتور ICSP برای برنامه ریزی برد آردوینو استفاده می شود. البته به طور معمول یک برد آردوینو از برنامه بوت لودر برای پروگرام کردن استفاده می کند اما اگر بوت لودر آسیب دیده باشد یا از بین رفته باشد در اون زمان کانکتور ICSP می تواند جایگزین مناسبی باشد. همچنین می تواند برای برگرداندن بوت لودر آسیب دیده هم استفاده بشود.

این پلت فرم ضمن داشتن سخت افزار، بخش نرم افزاری نیز دارد. این بخش خود شامل دو قسمت می باشد که بخش اول آن بوت لودر یا Boot loader و بخش دوم آن به نام IDE و یا محیط یکپارچه توسعه می باشد.

بوت لودر که یک بخش از **آردوینو** است، در واقع نوعی کد بسیار کوچک در درون میکروکنترلر می باشد که کنترلر را خاص و ویژه کرده و در واقع بین IDE و برد هماهنگی و یکپارچگی ایجاد می کند. در صورتی که شما بوت لودر را حذف نمایید، هر چقدر تلاش کنید IDE را برنامه ریزی نمایید، کنترلر اطلاعاتی دریافت نمی کند و چیزی متوجه نمی شود. به عبارت دیگر بوت لودر نوعی مترجم یا رابط بین IDE و کنترلر است و هماهنگی های لازم را ایجاد می کند. پیش از اینکه بوت لودر شروع به هماهنگی کند، باید آن را در قسمت کنترلر سیو کنید و یا از یک پروگرامر دیگری استفاده نمایید.



نحوه اتصال برد آردوینو Uno به رایانه:

برای اتصال بردها به کامپیوتر از مبدل USB به Serial استفاده شده است. در برد آردوینو نوع SMD از آئسی CH340 برای این مبدل و در آردوینو Uno نوع DIP از ATmega16 استفاده شده است. این موضوع یکی از دلایلی است که آردوینو smd Uno به نسبت برد آردوینو Uno R3 ارزانتر می باشد نکته ای که در باره برد SMD وجود دارد این است که درایور ارتباط USB به سریال آن به طور مجزا باید نصب شود تا برنامه قادر به شناسایی برد باشد.

مشخصات مهم برد آردوینو در جداول زیر نمایش داده شده است:

Board	Name	Arduino UNO R3	Power	I/O Voltage	5V
	SKU	A000066		Input voltage (nominal)	7-12V
Microcontroller	ATmega328P			DC Current per I/O Pin	20 mA
USB connector	USB-B			Power Supply Connector	Barrel Plug
Pins	Built-in LED Pin	13	Clock speed	Main Processor	ATmega328P 16 MHz
	Digital I/O Pins	14		USB-Serial Processor	ATmega16U2 16 MHz
	Analog input pins	6	Memory	ATmega328P	2KB SRAM, 32KB FLASH, 1KB EEPROM
	PWM pins	6		Dimensions	Weight
Communication	UART	Yes	Width		53.4 mm
	I2C	Yes	Length		68.6 mm
	SPI	Yes			

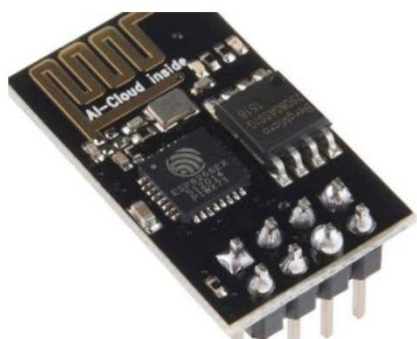
مشخصات فنی (ARDUINO UNO):

ATMega328P	میکروکنترلر
5 Volt	ولتاژ کاری میکروکنترلر
7~12VDC	ولتاژ ورودی برد (ولتاژ توصیه شده)
6~20VDC	محدوده ولتاژ ورودی برد
14 (که 6 عدد از آنها می تواند به عنوان PWM استفاده نمود)	پین های ورودی و خروجی (I/O)
6	خروجی PWM
6	ورودی آنالوگ
20mA	ماکزیمم جریان هر پین در حالت خروجی
50mA	جریان ماکزیمم برای پین 3.3 ولتی
32 کیلو بایت که 0.5KB آن برای Bootloader اشغال شده است	حافظه قابل برنامه ریزی (Flash Memory)
2KB	SRAM
1KB	EEPROM
16MHz	فرکانس کاری پردازنده (Clock Speed)
68.6mm	ابعاد برد: طول
53.4mm	ابعاد برد: عرض
25gr	وزن

1-1-5 مازول وای فای (ESP8266-01):

از این مازول برای اتصال برد آردوینو به سیستم Wi-Fi استفاده می کنیم که یکی از یکپارچه ترین تراشه های وای فای در صنعت است و قیمت مناسبی دارد و به آسانی نیز یافت می شود.

ESP8266 یک SOC (سیستم روی تراشه) وای فای است که توسط Espressif Systems تولید شده است. این یک تراشه بسیار یکپارچه است که برای ارائه اتصال کامل به اینترنت در یک بسته کوچک طراحی شده است. این مازول دارای 11 پین GPIO (پین های ورودی/خروجی عمومی) و یک ورودی آنالوگ نیز می باشد. این بدان معناست که شما می توانید آن را مانند هر آردوینو معمولی یا میکروکنترلر دیگری برنامه ریزی کنید. و علاوه بر آن، شما ارتباط Wi-Fi را دریافت می کنید، بنابراین می توانید از آن برای اتصال به شبکه Wi-Fi خود، اتصال به اینترنت، میزبانی وب سرور با صفحات وب واقعی استفاده کنید.



ESP8266 را می توان به عنوان یک مازول Wi-Fi خارجی، با استفاده از Firmware استاندارد AT Command set و با اتصال آن به هر میکروکنترلر دارای ارتباط سریال استفاده کرد. همچنین میتوان ESP8266 را مستقیماً به عنوان یک میکروکنترلر فعال Wifi، با برنامه ریزی یک سیستم عامل جدید با استفاده از SDK استفاده کرد. مازول ESP8266 علاوه بر پین های ورودی خروجی دیجیتال (I/O) دارای PWM، SPI، I2C و ... است.

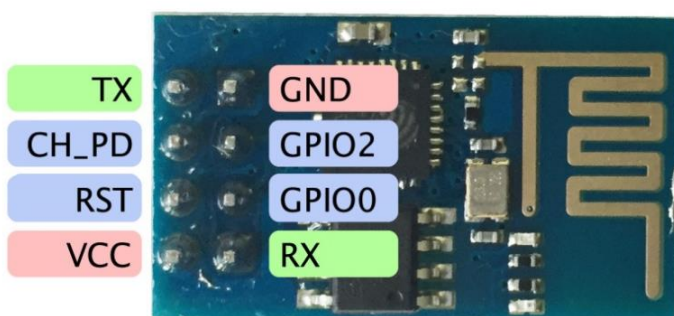
چند سال است که این برد وجود دارد و بیشتر در زمینه های اینترنت اشیا استفاده می شود، جایی که می خواهیم اتصال به اینترنت را مثلاً به پروژه آردوینو اضافه کنیم. قیمت این برد کم است و باعث شده است به انتخابی ایده آل برای پروژه های مبتنی بر اینترنت اشیا یا Wifi تبدیل شود.

ESP8266 از چندین فروشنده و با بردهای متفاوت در دسترس است. اکثر بردها از نظر تعداد پین های موجود، مقدار حافظه فلش برای ذخیره برنامه و داده ها، و محافظ روی SOC متفاوت هستند. برخی از بردها همچنین از کانکتور خارجی آنتن uFL و همچنین آنتن داخلی روی تراشه پشتیبانی می کنند.

رایج ترین انواع عبارتند از:

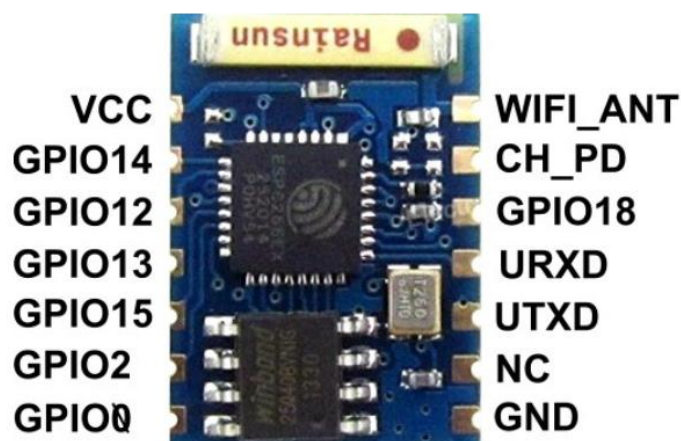
ESP-01-

این اولین و ساده ترین برد با استفاده از ESP8266 است. این ماژول امکان اتصال خطوط سریال را فراهم می کند و فقط دو پین GPIO را برای استفاده برجسته کرده است. این ماژول ارزان ترین ماژول مبتنی بر ESP8266 است و می توان آن را از بسیاری از تامین کنندگان چینی با 2.5 دلار خریداری کرد.



ESP-03-

این برد نسل دوم است که پین های GPIO بیشتری را برجسته کرده است و از یک آنتن متفاوت به اضافه یک کانکتور آنتن خارجی استفاده می کند.



ESP8266 کاربردهای زیادی در مورد اینترنت اشیا دارد. در اینجا فقط برخی از عملکردهایی که تراشه برای آنها استفاده می شود آورده شده است:

- شبکه: آنتن Wi-Fi ماژول دستگاه های تعبیه شده را قادر می سازد به روترها متصل شوند و داده ها را انتقال دهند.

- پردازش داده ها: شامل پردازش ورودی های پایه از حسگرهای آنالوگ و دیجیتال برای محاسبات بسیار پیچیده تر با RTOS یا غیر سیستم عامل SDK است.

- اتصال P2P: با استفاده از اتصال P2P اینترنت اشیا، ارتباط مستقیم بین ESP ها و سایر دستگاه ها ایجاد کنید.

- وب سرور: به صفحات نوشته شده در HTML یا زبان های توسعه دسترسی داشته باشید.

1-2 بروشور و دیتاشیت های المان های مدار:

دیتاشیت های المان های مدار به ترتیب زیر لینک شده اند:

✓ [دیتاشیت سنسور نور LDR MLG5516](#)

✓ [دیتاشیت سنسور دما LM335](#)

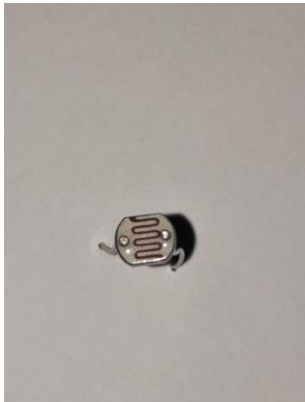
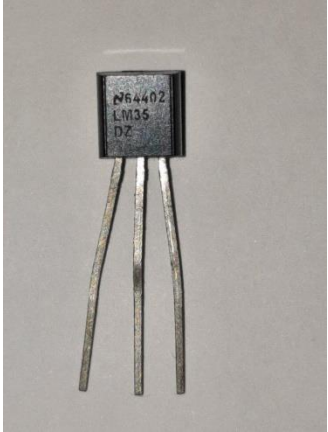
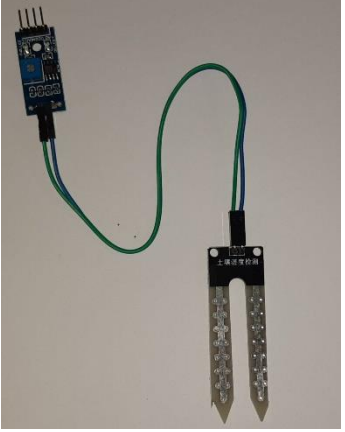

✓ [دیتاشیت سنسور رطوبت خاک YL-69](#)

✓ [دیتاشیت برد آردوینو Arduino UNO](#)

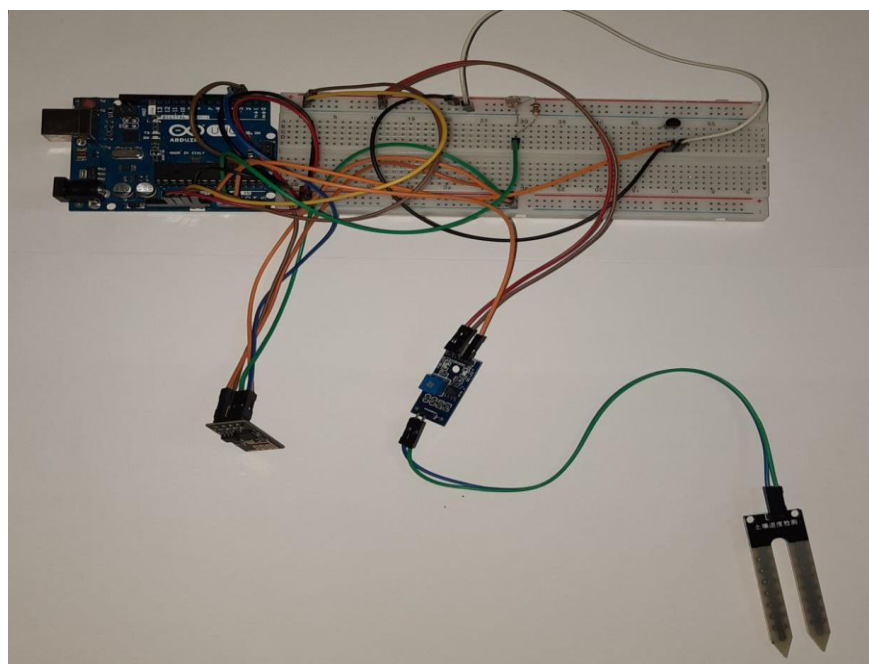
✓ [دیتاشیت ماژول ESP8266-01](#)

3-1 عکس از اجزاء سیستم:

عکس از المان های مدار به صورت جدول زیر ارائه شده است:

سنسور نور	سنسور دما	سنسور رطوبت	ماژول Wi-Fi
			

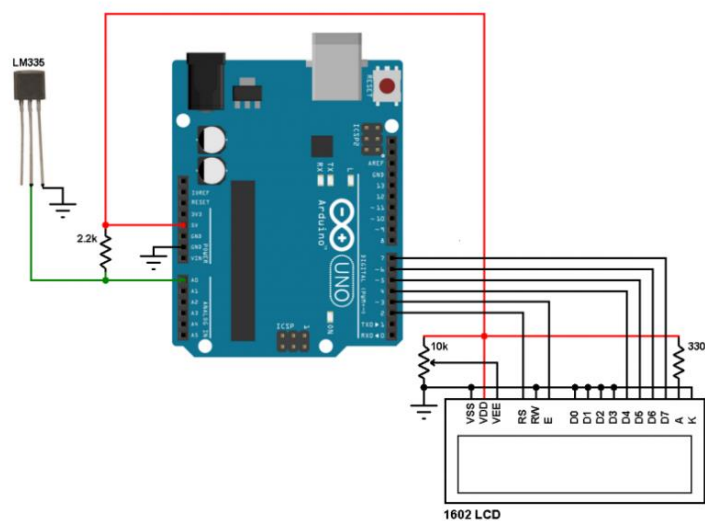
و عکس از مدار کلی ساخته شده:



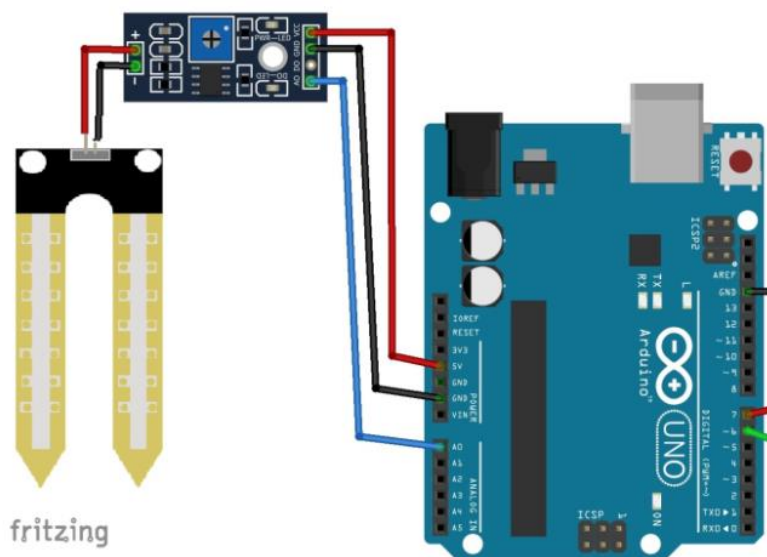
4-1 گزارش ساخت سیستم:

پس از خرید و تهیه قطعات مورد نیاز برای پروژه که شامل برد Arduino UNO ، سنسور رطوبت خاک-YL-69 ، سنسور دما LM335 ، مقاومت نوری LDR MLG5516 ، ماژول ESP8266-01 ، مقاومت کربنی ، برد و سیم های جامپر نری به نری و نری به مادگی می شود، با استفاده از دیتاشیت ها و بررسی در سایت های مختلف نحوه اتصالات را به دست می آوریم.

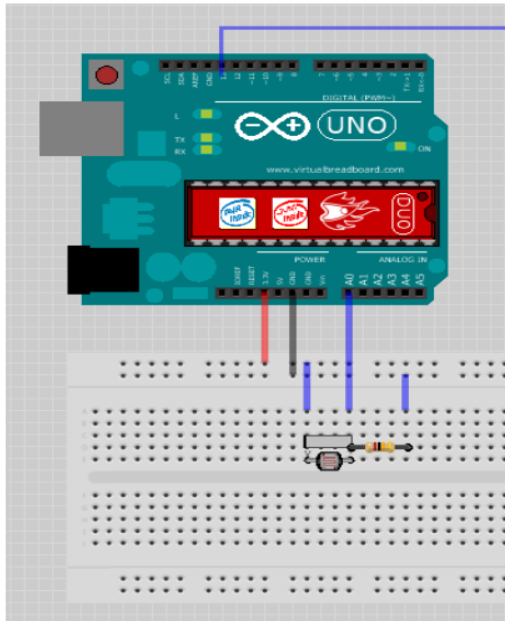
نحوه اتصال LM335 به برد:



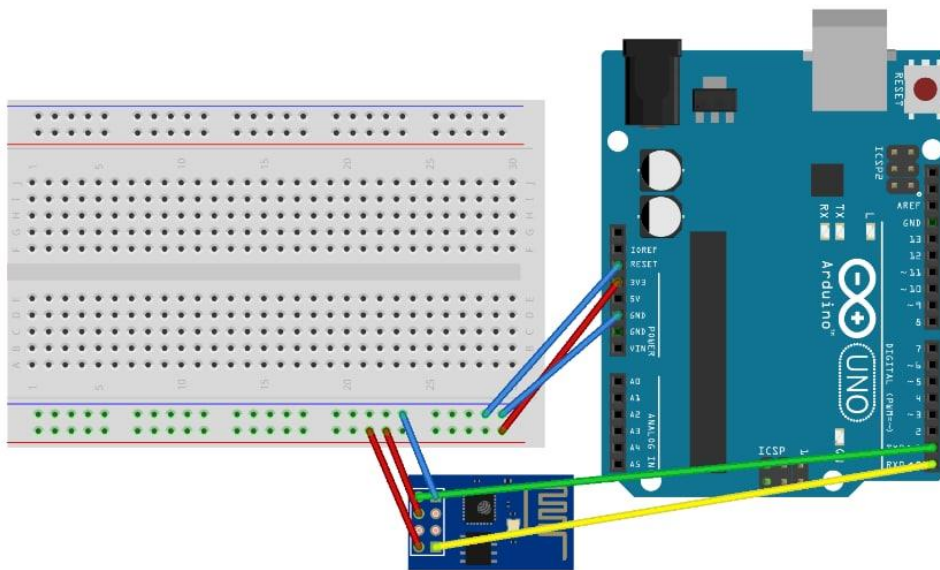
نحوه اتصال YL-69 به برد:



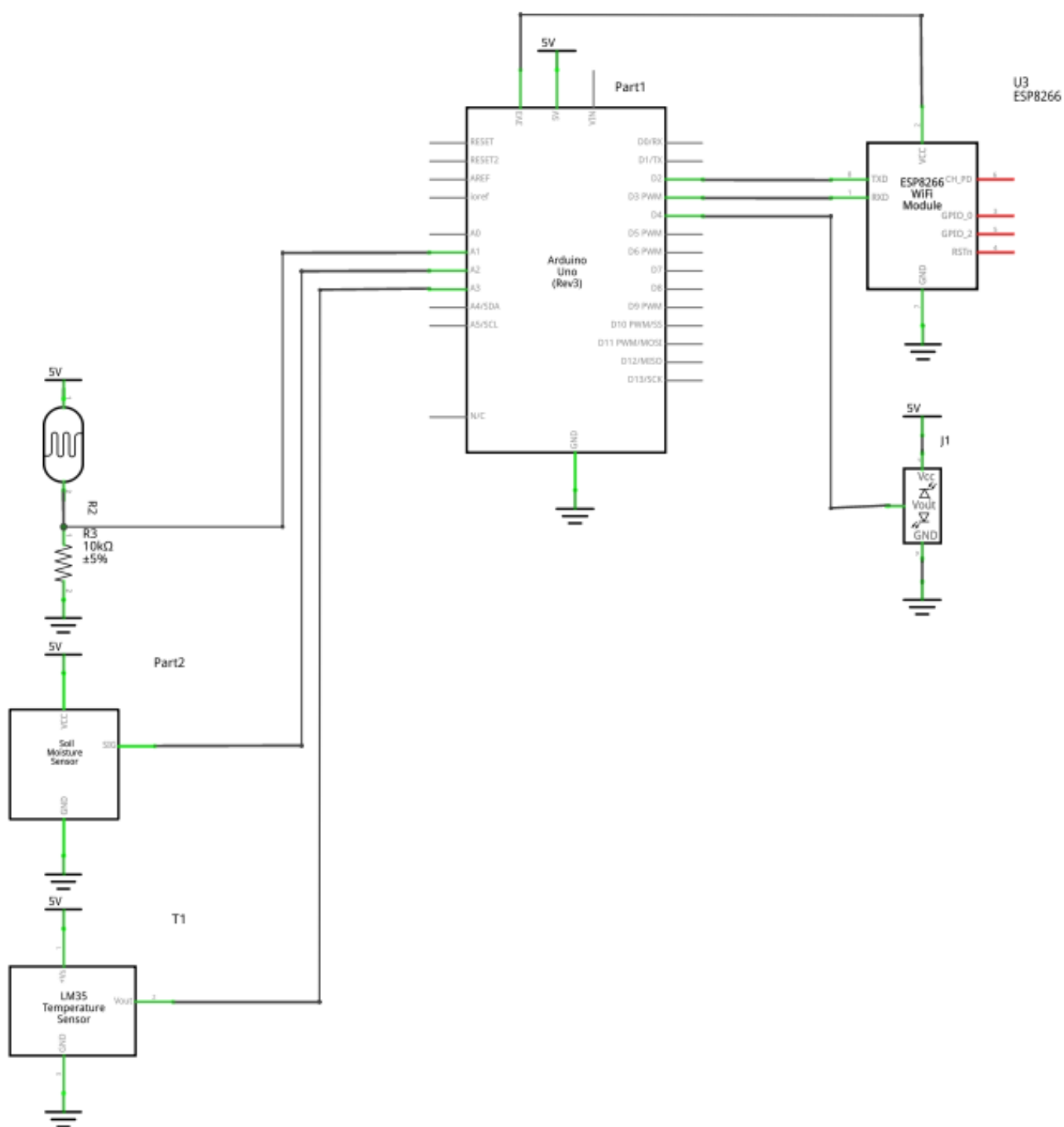
نحوه اتصال LDR MLG5516 به برد:



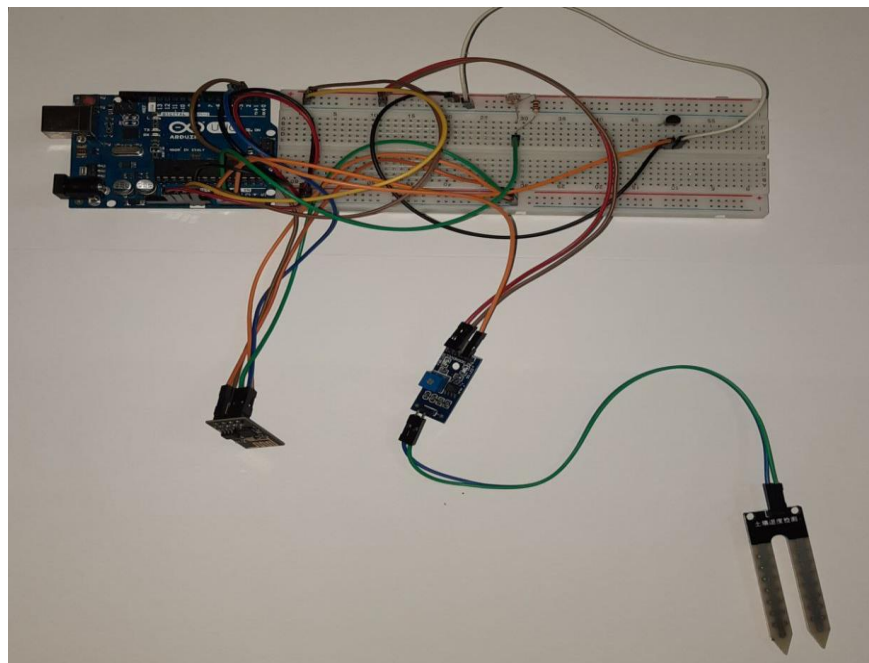
نحوه اتصال ESP8266-01 به برد:



شمای کلی مدار:



به دلیل اینکه از پایه های Vcc و GND برد استفاده زیادی می شود و به جهت راحتی اتصالات، قطعات را بر روی برد متصل می کنیم:



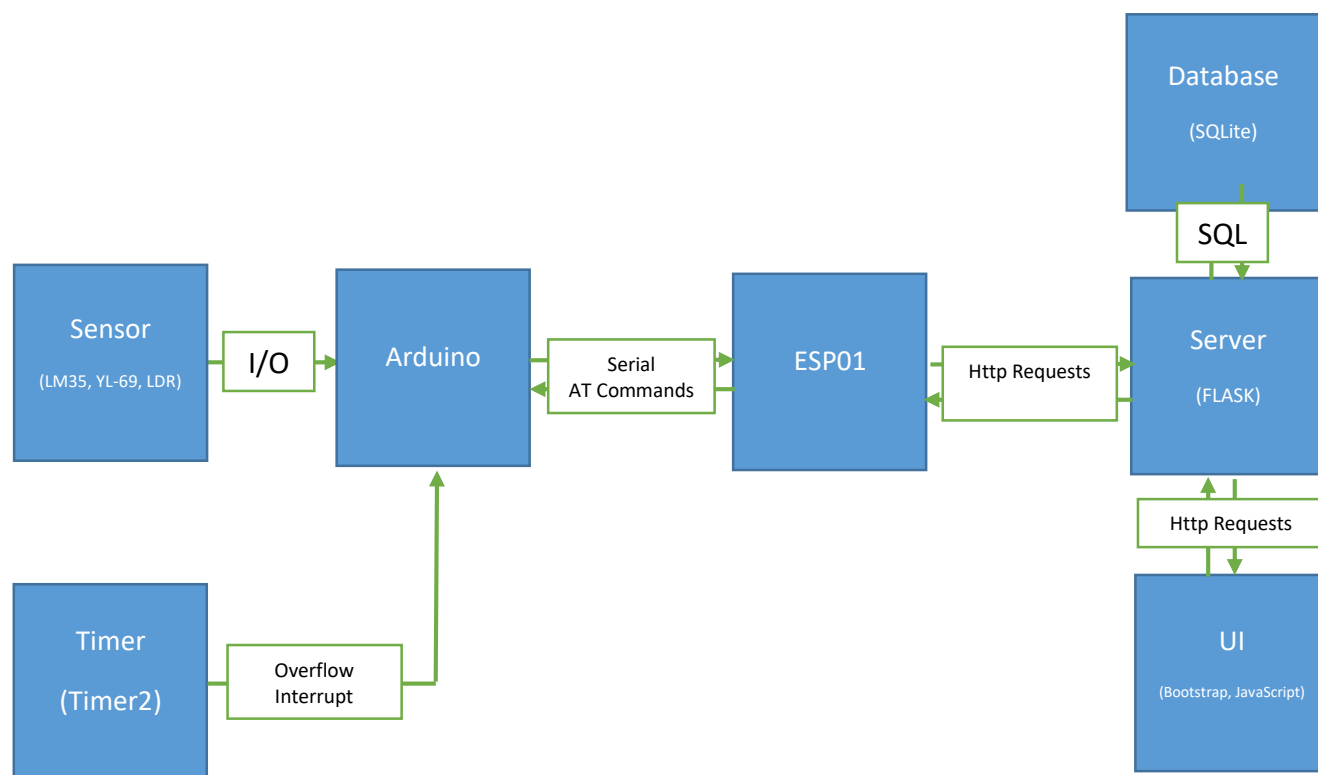
1-5 گزارش تست سیستم:

کلیپ تست سیستم به همراه توضیحات کامل از مدار و تست تک تک سنسورها با اشاره به کد آردوینو در فصل چهارم قرار داده شده است.

فصل دوم: گزارش نرم افزار های نگارش و استفاده شده

1-2 اصل و توضیح بخش های مختلف کد به همراه عکس صفحات کاربر:

اصل کلی کد پروژه بطور خلاصه به شکل زیر است:



1-1-2 ارتباط میان آردوینو و سنسور ها:

این ارتباط توسط ورودی ها و خروجی های دیجیتال و آنالوگ آردوینو انجام میشود.

سنسور LM35 از ولتاژ خود برای نمایش میزان دما استفاده میکند. LM35 یک دماسنج آنالوگ 3 پین است که در بسته بندی TO-92 بسته بندی شده است. خروجی یک ولتاژ خطی است که با $10.0 \text{ mV}/^{\circ}\text{C}$ مطابقت دارد، بنابراین خروجی مستقیماً بر حسب درجه سانتیگراد با استفاده از یک پورت آنالوگ روی یک میکروکنترلر قابل خواندن است. دقت معمولاً در 0.5 درجه سانتیگراد است.

از آنجایی که محدوده ولتاژ خروجی سنسور به 1.5 ولت محدود شده است، در صورت تمایل به وضوح بیشتر، ولتاژ مرجع آنالوگ در uC را می توان از 5 ولت پیش فرض به چیزی نزدیک تر به 1.5 ولت تغییر داد که اندازه گام ولتاژ را کاهش می دهد.

یک جایگزین برای اندازه گیری دما، استفاده از سنسور دما دیجیتال مانند DS18B20 است که در همان بسته ارائه می شود. سنسورهای دما دیجیتال معمولاً ایمنی بهتری نسبت به نوپز دارند که زمانی مفید است که سنسور در فاصله ای دورتر قرار گیرد یا در یک محیط پر سر و صدا قرار گیرد، اما برای اجرای پروتکل ارتباطی به یک کتابخانه نرم افزاری نیاز دارند. کتابخانه ها برای اکثر میکروکنترلرهای رایج مانند آردوینو به راحتی در دسترس هستند و استفاده از آنها بسیار ساده است. اما ما در این پروژه از سنسور LM35 استفاده کرده ایم.

این برنامه از این خط برای تبدیل خروجی LM35 به درجه سانتیگراد استفاده می کند.

`temperature = ((analogRead(LM335Pin) * 5.0 / 1024.0) * 100;`

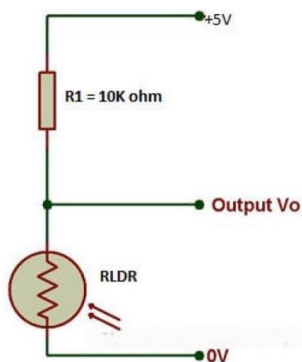
این خط به سادگی خروجی آنالوگ خام را می گیرد و آن را در 5.0 ضرب می کند زیرا ولتاژ مرجع 5 ADC ولت است. سپس آن عدد را بر 1024 تقسیم می کند زیرا آردوینو دارای یک 10 ADC بیتی است که 1024 مرحله وضوح را ارائه می دهد. نتیجه آن بیت ریاضی ولتاژی را که در حال اندازه گیری از LM35 که در محدوده 0 تا 1.5 ولت خواهد بود به ما می دهد. برای رسیدن به درجه سانتیگراد از ولتاژ به دما، آن عدد را در 100 ضرب می کنیم زیرا خروجی 0.010 ولت در هر درجه است.

2-1-2 ارتباط میان سنسور YL-69:

سنسور YL-69 یک سنسور مقاومتی است. ولتاژ خروجی این سنسور با خشکی خاک افزایش و با رطوبت آن کاهش میابد. خروجی آنالوگ آن توسط ADC آردوینو خوانده و برای تبدیل آن به مقداری که با رطوبت افزایش و با خشکی کاهش میابد، از خروجی آن 1024 را کم می کنیم.

`1023 - analogRead(MoistureSignal);`

2-3-1-2 ارتباط میان سنسور نور LDR:



مقاومت وابسته به نور LDR از قطعه ای از مواد نیمه هادی در معرض نور مانند سولفید کادمیوم ساخته شده است که با ایجاد جفت حفره-الکترون در تاریکی، مقاومت الکتریکی خود را از چندین هزار اهم در تاریکی به تنها چند صد اهم تغییر می دهد. در این سنسور با افزایش نور مقاومت کاهش و با کاهش نور مقاومت افزایش می یابد. برای بررسی این تغییر مقاومت توسط آردوینو از مدار روبه رو استفاده

میکنیم. خروجی ولتاژ این مدار با کاهش نور افزایش و با افزایش نور کاهش می یابد و نسبتی معکوس دارد. برای تبدیل آن به نسبت مستقیم از خروجی آن 1024 کم می کنیم.

```
light = 1023 - analogRead(LDRPin);
```

2-1-4 ارتباط آردوینو با تایمر داخلی آن:

در پروژه ما آردوینو هر 15 دقیقه اطلاعات را از سنسور ها جمع آوری و سپس توسط ESP01 به سرور ارسال میکند. برای اینجا این تاخیر 15 دقیقه، ما از تایمر های داخلی آردوینو و میکروکنترلر داخل آن برای ایجاد تاخیر یک دقیقه استفاده کرده ایم، سپس هر 15 دقیقه کدهای ما اجرا میشوند. آردوینو Uno از میکروکنترلر ATmega328P استفاده می کند که دارای 3 تایمر است. ما از تایمر 8 بیتی 2 استفاده میکنیم. برای تنظیم آن لازم است از ریجسترها و ثبات های این تایمر در کد استفاده کنیم.

```
cli();
```

برای اینکار باید تمام وقفه ها (Interrupt) ها را غیرفعال کنیم، زیرا وجود وقفه در میان تنظیم تایمر ها ممکن است باعث خرابی تنظیمات و اعمال نشدن آنها شود.

```
TCCR2A = (0 << COM2A1) | (0 << COM2A0);
```

در ادامه باید منبع تایمر را انتخاب کنیم. تایمر ها میتوانند به عنوان شمارنده یا تایمر استفاده شوند، استفاده از حالت شمارنده می تواند در مواردی مانند استفاده از سنسور آلتراسونیک برای تشخیص فاصله کاربردی باشد، اما در این پروژه همچین استفاده ایی از تایمر ها نداریم و از منبع کلاک خود آردوینو که 16 مگاهرتز است استفاده میکنیم.

```
TCCR2B = (1 << CS22) | (1 << CS21) | (1 << CS20);
```

سپس مقسم منبع کلاک برای ایجاد تاخیر بیشتر استفاده میکنیم. در این حالت ما از مقسم 1024 استفاده کردیم، به این معنا که هر 1024 سیکل کلاک (که 16 مگاهرتز است) کلاک تایمر یک سیکل کلاک را دریافت میکند (که 15625 هرتز است).

```
TCNT2 = 28;
```

در این بخش میزان باقی مانده تایمر را تنظیم می کنیم. برای برخی از تاخیرها نیاز است تایمر از مقدار خاصی شروع کند تا تاخیر تاحد ممکن دقیق باشد. برای محاسبه آن می توان از محاسبه گر های آنلاین استفاده کرد (در بخش منابع وبسایت استفاده شده قرار داده شده است)

```
TIMSK2 = (1 << TOIE2);
```

در اینجا تایمر را برای ایجاد وقفه در هر سرریز تایمر تنظیم میکنیم، با اینکار آردوینو نیازی ندارد که تایمر را برای سرریز شدن بررسی کند و تایمر زمانی که سرریز رخ داد، آردوینو را مطلع می سازد.

```
sei(); // Set interrupts
```

در اینجا نیز وقفه ها دوباره فعال میشوند.

```
volatile unsigned long ovf_count = 0, min_ovf = 0;
```

به علت سرعت بالای آردوینو، نیاز است که چندین بار سرریز تایمر رخ دهد تا تاخیر مورد نیاز ایجاد شود، برای نگه داری تعداد این سرریزها از متغیر `ovf_count` استفاده کرده ایم. این متغیر به علت استفاده شدن در وقفه بطور `volatile` تعریف شده است، این کار برای جلوگیری از `Data race` و خرابی مقدار آن در صورت ایجاد وقفه در زمان تغییر مقدار آن انجام شده است.

```
ISR(TIMER2_OVF_vect) {  
}
```

وقفه ها در آردوینو مانند توابع تعریف میشوند، اما برخلاف آنها از سینکس `ISR` (Interrupt Service Routine) برای تعریف آن استفاده میکنیم. در اینجا `ISR` برای سرریز تایمر 2 تعریف شده است.

```
TCNT2 = 28;
```

در این بخش میزان باقی مانده تایمر را تنظیم می کنیم. برای برخی از تاخیرها نیاز است تایمر از مقدار خاصی شروع کند تا تاخیر تاحد ممکن دقیق باشد. برای محاسبه آن می توان از محاسبه گر های آنلاین استفاده کرد (در بخش منابع وبسایت استفاده شده قرار داده شده است)

```
ovf_count++;
```

در اینجا تعداد سرریز ها رخ داده بیشتر میشود (زیرا برای ایجاد تاخیر های بیشتر به تعداد سرریز بیشتری نیاز است)

```
if (ovf_count == 3662) {  
    min_ovf++;  
    ovf_count = 0;  
}
```

با بررسی اینکه تعداد سرریز ها به 3662 سرریز رسیده است میتوانیم متوجه شویم یک دقیقه تاخیر انجام شده یا نشده است. برای محاسبه این مقدار میتوان از وبسایتی که در منابع قرار داده شده است استفاده کرد. در صورتی که این تعداد سرریز انجام شد، تعداد سرریز های یک دقیقه افزایش می باشد.

و در کد با بررسی متغیر min_ovf میتوان متوجه شد از زمان اجرا چند دقیقه گذشته است.

2-1-5 ارتباط ESP و آردوینو:

برای ارتباط میان ESP و آردوینو از کتابخانه Software Serial استفاده میکنیم. این کتابخانه یک ارتباط سریال مجازی میان ESP و آردوینو ایجاد میکند، زیرا ارتباط اصلی سریال آردوینو برای مخابره و اتصال با کامپیوتر است. آردوینو با استفاده از دستورات AT که در Firmware ماژول ESP موجود است استفاده میشود.

در ابتدا آردوینو وجود و درستی ESP را با ارسال دستور AT بررسی میکند. در صورتی که پاسخ از طرف ماژول OK باشد، ماژول سالم است و آردوینو بقیه مراحل را ادامه میدهد.

در بخش بعدی آردوینو دستور AT+RST را به ماژول ارسال می کند، با اینکار ماژول ریست می شود و در صورتی که مشکلی از نظر اتصالاتی موجود باشد، تا حدودی رفع میشود. همچنین در صورتی که ماژول به اینترنت متصل باشد، خروجی GOT IP را نیز با خروجی ها دیگر مانند نسخه ماژول و... ارسال میکند. در صورتی که ماژول به اینترنت متصل نباشد، آردوینو دستورات تابع ConnectToWiFi را اجرا میکند (در ادامه در خصوص آنها توضیح داده میشود)

در مرحله بعدی با دستور AT+CWMODE=3 ماژول در حالت سرور و کلاینت قرار میگیرد. با اینکار میتواند با سرور ارتباط داشته باشد.

در بخش بعدی با اجرای دستور AT+CIPMUX=1 تعداد حداکثر اتصالات Http ماژول به 1 تغییر میکند (برای راحتی دستورات در ادامه) و با اینکار مراحل تنظیمات اولیه ESP به پایان میرسد.

همانطور که در قسمت های قبل ذکر شد، در صورتی که ESP به اینترنت متصل نباشد، دستورات تابع ConnectToWiFi اجرا میشود. این تابع در ابتدا با اجرای دستور AT+CWLAP لیست تمام Access Point ها یا اتصالات وای فای که ماژول میتواند به آنها متصل شود را دریافت میکند و به Serial Monitor ارسال می کند. سپس از کاربر SSID و رمز عبور AP مورد نظر را دریافت و با دستور AT+CWLAP= را به آن AP متصل میکند. در صورتی که اتصال با موفقیت انجام شود ماژول با OK پاسخ می دهد و خروجی تابع true میشود، در صورتی که اتصال با موفقیت نباشد خروجی تابع false میشود.

سپس در محلی که تابع ConnectToWiFi فراخوانی میشود، توسط یک حلقه while تا زمانی که خروجی true نباشد اجرا میشود (به این معنا که این تابع تا زمانی که اتصال به وای فای ممکن نشده است اجرا میشود)

ارسال داده های سنسور به ESP سپس به سرور نیز توسط دستورات AT انجام میشود. در مرحله اول با کامند `AT+CIPSTART=0,"TCP","SERVER_ADDR",SERVER_PORT` ماژول به سرور متصل میشود. آدرس سرور و پورت اتصال توسط دو متغیر `SERVER_ADDR` و `SERVER_PORT` به ESP اعلام میشود. برای اینکار از تابع `sprintf` برای ایجاد این دستور با توجه به `SERVER_ADDR` و `SERVER_PORT` استفاده میشود. در صورتی که اتصال به سرور با مشکل مواجه شود، ماژول با `ERROR` پاسخ میدهد.

در مرحله بعدی مراحل Login کردن در سرور انجام میشود. برای اینکار باید دو مقدار `username` و `password` به سرور ارسال شود. شیوه ارسال داده یک درخواست `HTTP` به شیوه `POST` است. درخواست های `HTTP` میان سرور و ماژول در ادامه بطور کامل توضیح داده میشود.

در ادامه پس از ارسال این درخواست به سرور، `sessionCookie` از پاسخ سرور (که توسط ESP به آردوینو ارسال میشود) استخراج میشود. این مقدار مشخص کننده این است که ما به سرور Login کرده ایم و با هر درخواست که نیاز به Login دارد این مقدار باید ارسال شود. علت وجود `Login` و `sessionCookie` برای افزایش امنیت سرور و همچنین جلوگیری از دسترسی دیگر افراد بدون ثبت نام در سرور است.

در مرحله بعدی مراحل ارسال داده ها به سرور انجام میشود. برای اینکار باید مقادیر سنسور ها به سرور ارسال شود. شیوه ارسال داده یک درخواست `HTTP` به شیوه `POST` است. درخواست های `HTTP` میان سرور و ماژول در ادامه بطور کامل توضیح داده میشود. همچنین در این درخواست باید `sessionCookie` نیز همراه با درخواست `HTTP` ارسال شود.

2-1-6 ارتباط میان سرور و ESP:

این ارتباط توسط درخواست های HTTP به شیوه های GET و POST انجام میشود.

این جدول تمام درخواست ها و عملکرد های آنها را نشان می دهد:

عملکرد	درخواست، شیوه و پارامتر ها
ورود به سرور با توجه به پارامتر های username و password	POST /auth/login?username=&password=
ثبت نام در سرور با توجه به پارامتر های username و password	POST /auth/register?username=&password=
خروج از سرور	POST /auth/logout
تغییر نام کاربری با توجه به پارامتر های username و password	POST /auth/change_username?username=&password=
ثبت داده در سرور با توجه به پارامتر های سنسور	GET /api/insert_record?light=&moisture=&tempeature
دریافت کل کاربران و دستگاه های سرور	GET /api/get_devices
دریافت n سطر از داده های سنسورها	GET /api/fetch_recent_record?limit=n

برای اجرای این دستورات، ESP مراحل زیر را طی میکند:

AT+CIPSTART=0,"TCP",**SERVER_ADDR**,**SERVER_PORT**

AT+CIPSEND=0,**REQUEST_LENGTH**

REQUEST

AT+CIPCLOSE

در اینجا **SERVER_ADDR** آدرس سرور، **SERVER_PORT** پورتهی که سرور روی آن اجرا شده است، **REQUEST_LENGTH** طول درخواست به همراه اطلاعات (تعداد کاراکترهای آن) و **REQUEST** خود درخواست است.

بخش های مختلف سرور توسط قسمت هایی به نام Blueprint از کتابخانه flask برای راحتی کار و جداسازی بخش های مختلف از یکدیگر در فایل های مختلف قرار گرفته اند. این Blueprint ها همچنین باعث ایجاد بخش هایی مانند /auth یا /api در درخواست های http و همچنین افزایش امنیت سرور با جداسازی بخش های نیازمند ورودی به وبسایت از بخش های بدون نیاز آنها میشود.

2-1-7 ارتباط میان سرور و دیتابیس:

ما در این پروژه از دیتابیس SQLite برای ذخیره و خواندن داده ها استفاده کردیم. این دیتابیس با زبان برنامه نویسی SQL بر روی جدول ها تغییراتی و دستوراتی اجرا میکند. سرور ما دارای دو جدول است، یک جدول برای کاربرها و جدولی دیگر برای داده های ارسالی توسط ESP. این شیوه از ایجاد دیتابیس و ذخیره داده ها باعث میشود که چندین گلدان را نیز در موقعیت های مختلف کنترل و بررسی کرد.

جدول کاربران	
id	مشخصه منحصر به فرد
username	نام کاربری
password	گذرواژه
جدول اطلاعات	
id	مشخصه منحصر به فرد داده
device_id	مشخصه منحصر به فرد ارسال کننده داده
light	میزان نور
moisture	میزان رطوبت
temperature	میزان دما
created	تاریخ و زمان ثبت داده

اطلاعات این جداول توسط کتابخانه sqlite در پایتون خوانده و به JSON تبدیل میشوند. سپس این اطلاعات در پاسخ درخواست های HTTP مشخص (به عنوان مثال داده های سنسور در درخواست fetch_recent_record) به درخواست کننده ارسال میشوند.

2-1-8 ارتباط میان سرور و محیط کاربر:

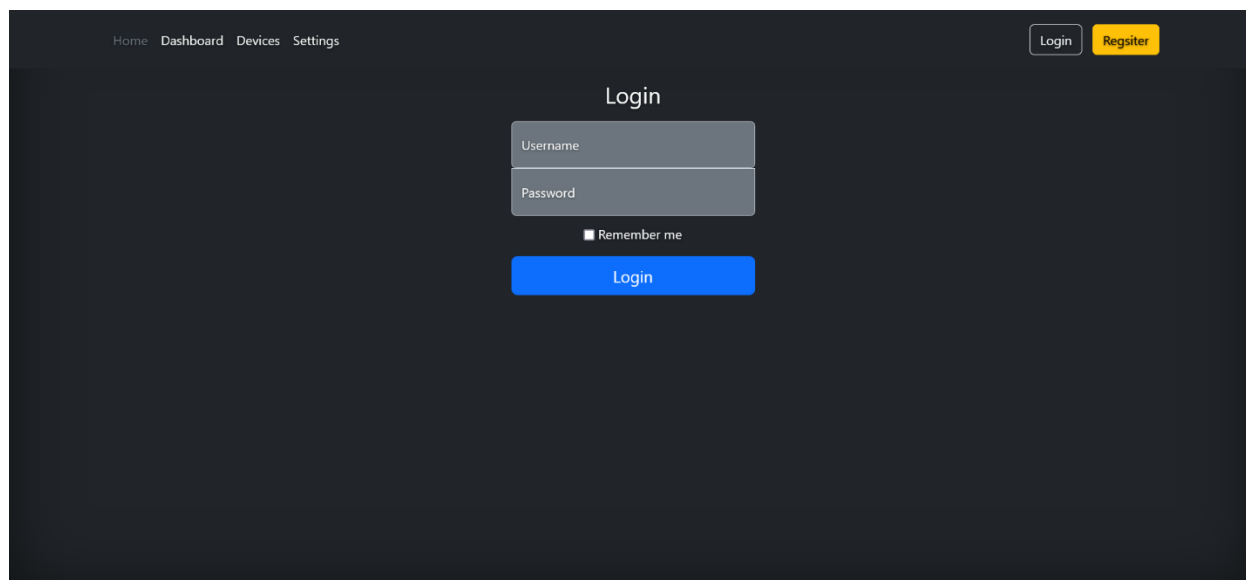
محیط کلی کاربری، با استفاده از جاوا اسکریپت و بوت استرپ نوشته شده است. از بوت استرپ برای طراحی قالب کلی و محیط کاربری استفاده شده است و جاوا اسکریپت با استفاده از `fetch` داده ها را با استفاده از درخواست های `Http` از سرور دریافت می کند، این داده ها سپس توسط جاوا اسکریپت به صفحه اضافه میشود. همچنین با استفاده از جاوا اسکریپت مواردی مانند ثبت نام و ورودی به وبسایت انجام میشود. کل صفحات وبسایت کاملاً واکنش گرا و قابل استفاده در هر مرورگر بروز و جدیدی است.

–صفحه اصلی:



در این صفحه اطلاعاتی در خصوص خود سرور، سازنده و... آن موجود است. همچنین دارای دکمه ای برای ورودی به وبسایت است. از منوی وبسایت نیز میتوان دسترسی به صفحه اصلی، صفحه ورودی و صفحه ثبت نام داشت.

–صفحه ورود:



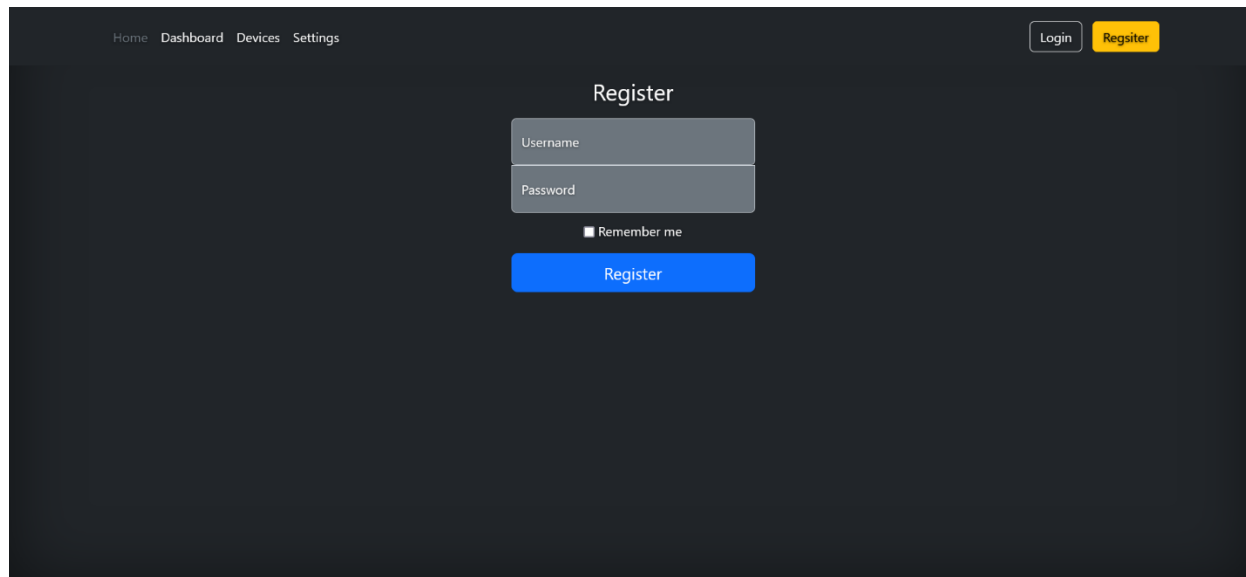
این صفحه نیز دارای لینک ها و دکمه هایی به صفحات داشبورد، تنظیمات و دستگاه ها دارد. همچنین در بالا سمت راست لینک برای ثبت نام و ورودی نیز دارد.

در وسط صفحه فرم ورودی اطلاعات است. در صورت وقوع خطا و مشکلی، در بخش بالایی فرم خطا نمایش داده میشود. همچنین این فرم قابلیت ذخیره اطلاعات نام کاربری را دارد که با انتخاب گزینه Remember me فعال میشود.

پس از ورود موفقیت آمیز، کاربر به صفحه داشبرد انتقال میابد.

مراحل ورودی به وبسایت توسط جاوا اسکریپت و توسط تابع fetch انجام میشود. با کلیک بر دکمه login جاوا اسکریپت به سرور درخواست ورود می دهد و درصورت تقبل درخواست از طرف سرور، پس از یک ثانیه کاربر به صفحه داشبرد انتقال پیدا می کند. در صورت رد درخواست از طرف سرور، علت رد درخواست با JSON به جاوا اسکریپت ارسال میشود و جاوا اسکریپت با استفاده از این اطلاعات علت رد خواست را به کاربر نشان میدهد. در این روش مرورگر کاربری تازه سازی نمیشود و علت رد درخواست بطور بلادرنگ به کاربر نشان داده میشود.

–صفحه ثبت نام:



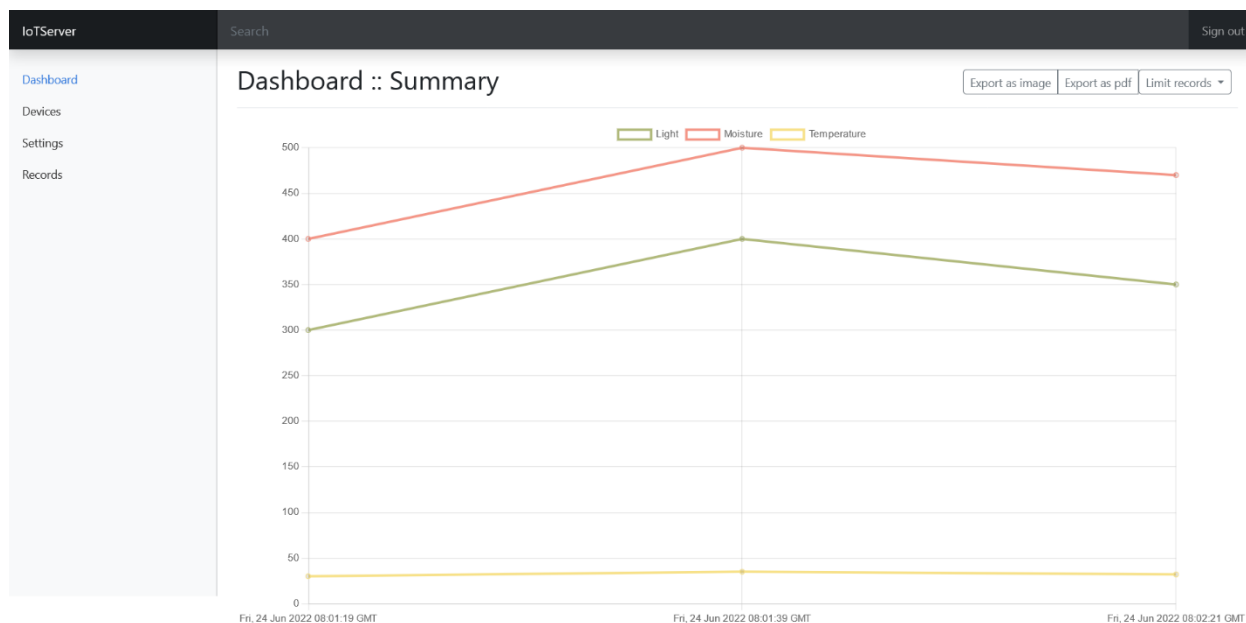
این صفحه نیز دارای لینک ها و دکمه هایی به صفحات داشبورد، تنظیمات و دستگاه ها دارد. همچنین در بالا سمت راست لینک برای ثبت نام و ورودی نیز دارد.

در وسط صفحه فرم ثبت نام است. در صورت وقوع خطا و مشکلی، در بخش بالایی فرم خطا نمایش داده میشود. همچنین این فرم قابلیت ذخیره اطلاعات نام کاربری را دارد که با انتخاب گزینه Remember me فعال میشود. پس از ثبت نام موفقیت آمیز، کاربر به صفحه ورود انتقال میابد.

مراحل ثبت نام به وبسایت توسط جاوا اسکریپت و توسط تابع fetch انجام میشود. با کلیک بر دکمه Register جاوا اسکریپت به سرور درخواست ثبت نام می دهد و در صورت تقبل درخواست از طرف سرور، پس از یک ثانیه کاربر به صفحه ورود انتقال پیدا می کند. در صورت رد درخواست از طرف سرور، علت رد درخواست با JSON به جاوا اسکریپت ارسال میشود و جاوا اسکریپت با استفاده از این اطلاعات علت رد خواست را به کاربر نشان میدهد. در این روش مرورگر کاربری تازه سازی نمیشود و علت رد درخواست بطور بلادرنگ به کاربر نشان داده میشود.

همچنین در این صفحه جاوا اسکریپت با استفاده از Regex خوبی رمز و وجود کاراکتر های غیرمجاز بررسی میشود، به عنوان مثال وجود حروف الفبای فارسی در رمز و نام کاربری مشکل ساز است و توسط Regex و جاوا اسکریپت از وقوع این مشکل جلوگیری میشود.

–صفحه داشبرد:



در این صفحه خلاصه ای از داده های دریافت شده از سنسورها به شیوه نمودار خطی نشان داده میشود. همچنین در محور X زمان ایجاد درخواست ها نیز نمایش داده میشود. هریک از خطوط را نیز میتوان غیر فعال کرد. برای اینکار کافی است روی برجسب خط مورد نظر کلیک کرد و با اینکار از نمودار محو میشود، همچنین محور Y نیز با این تغییر کمتر یا بیشتر میشود تا خطوط دیگر با دقت بیشتری نمایش داده شوند. همچنین میتوان با بردن اشاره گر بر روی هر یک از نقطه های داده میتوان مقدار دقیق آن را نیز بررسی کرد. نمودار موجود در این صفحه هر 15 دقیقه بطور خودکار تازه سازی میشود و با اینکار داده های جدیدتر از سرور دریافت میشود.

این صفحه تنظیمات و قابلیت های دیگری نیز دارد. با استفاده از دکمه های Export در بالا سمت راست میتوان نمودار را به شکل PDF یا تصویر روی رایانه یا تلفن همراه خود ذخیره کرد.

با استفاده از دکمه Limit Records میتوان تعداد داده های نمایش داده شده روی نمودار را میتوان تغییر داد. موارد قابل انتخاب آن 10 داده آخر، 20 داده آخر و یا همه داده ها است.

در بالا سمت راست نیز میتوان با استفاده از دکمه Sign Out از وبسایت خروج کرد. با اینکار نشست کاربری شما به پایان می رسد و به صفحه اصلی بازگردانده میشود.

در منوی سمت چپ نیز می توانید به بخش های مختلف داشبرد مانند صفحه داشبرد، تنظیمات، دستگاه ها و رکورد های داده بروید.

IoTServer

Search

Sign out

Dashboard

Devices

Settings

Records

Dashboard :: Devices

Export as CSV

#	Username
1	reyhaneh

در این صفحه دستگاه ها و کاربران ثبت نام کرده در وبسایت نمایش داده میشود. در بالا سمت راست نیز با استفاده از دکمه `Export as csv` میتوان لیست تمام این کاربران را بصورت فایل اکسل ذخیره کرد. برای دریافت داده از سرور از جاوا اسکریپت و `fetch` استفاده شده است.

IoTServer Search Sign out

Dashboard
Devices
Settings
Records

Dashboard :: Settings

Username

Current password

Save settings

در این صفحه میتوان نام کاربری خود را تغییر داد. برای اینکار باید نام کاربری دلخواه خود را در ورودی Username وارد کرد و سپس رمز فعلی خود را در ورودی Current Password وارد کرد. با کلیک بر دکمه Save settings تغییر اعمال میشود و شما به صفحه ورود انتقال پیدا میکنید تا دوباره وارد وبسایت شوید. در صورت وقوع هر گونه مشکل علت مشکل در بخش بالایی فرم نمایش داده میشود.

–صفحه رکورد ها:

IoTServer

Search

Sign out

Dashboard

Devices

Settings

Records

Dashboard :: Records

Export as CSVLimit records

#	Device ID	Created	Light	Moisture	Temperature
1	1	Fri, 24 Jun 2022 08:01:19 GMT	300	400	30
2	1	Fri, 24 Jun 2022 08:01:39 GMT	400	500	35
3	1	Fri, 24 Jun 2022 08:02:21 GMT	350	470	32

در این صفحه داده ها به شکل جدول نمایش داده میشود. همانند صفحه داشبرد، اطلاعات این صفحه هر 15 دقیقه بطور اتوماتیک توسط جاوا اسکریپت تازه سازی میشوند و نیازی به تازه سازی مرورگر توسط کاربرد نیست. همچنین در این صفحه مشابه صفحه داشبرد میتوان با استفاده از دکمه Limit Records میتوان داده تعداد داده های نمایش داده شده روی نمودار را میتوان تغییر داد. موارد قابل انتخاب آن 10 داده آخر، 20 داده آخر و یا همه داده ها است. در بالا سمت راست نیز با استفاده از دکمه Export as csv میتوان تمام داده ها را بصورت فایل اکسل ذخیره کرد. برای دریافت داده از سرور از جاوا اسکریپت و fetch استفاده شده است.

2-2 پارامتر های مورد نیاز سرور:

پارامتر های مورد نیاز سرور	
آدرس IP سرور	برای تنظیم آن هنگام اجرای سرور با Flask باید از آپشن host استفاده کرد.
پورت سرور	برای تنظیم آن هنگام اجرای سرور با Flask باید از آپشن port استفاده کرد.
پارامتر های مورد نیاز آردوینو	
آدرس IP سرور	در کد متغیر server را تغییر دهید.
آدرس پورت سرور	در کد متغیر port را تغییر دهید.
نام کاربری جهت استفاده توسط آردوینو	در کد متغیر username را تغییر دهید.
رمز عبور جهت استفاده توسط آردوینو	در کد متغیر password را تغییر دهید.
پارامتر های مورد نیاز آردوینو در هنگام استفاده از ThingsBoard	
آدرس IP سرور	در کد متغیر server را تغییر دهید.
آدرس پورت سرور	در کد متغیر port را تغییر دهید.
Access Token	در کد متغیر token را تغییر دهید.

2-3 توضیحات کد به صورت جزئی تر:

در ابتدای کار باید کتابخانه SoftwareSerial اضافه گردد. کتابخانه SoftwareSerial با استفاده از نرم افزار برای تکرار عملکرد، امکان برقراری ارتباط سریال روی سایر پین های دیجیتال برد آردوینو را فراهم می کند. امکان داشتن چندین پورت سریال نرم افزار با سرعت 115200 bps وجود دارد. برای سهولت کار در ادامه نیز چند مورد را define می کنیم. دقت داریم که baud rate ماژول ESP را پس از تنظیم روی 9600 ثابت قرار دادیم.

```
#include <SoftwareSerial.h>

#define ArduinoRX 3 // Wire this to Tx Pin of ESP
#define ArduinoTX 2 // Wire this to Rx Pin of ESP
#define ESPBaud 9600
```

از آن جا که LDR به پین A1 آردوینو متصل می شود، آن را با همین نام define می کنیم و به همین ترتیب سنسور moisture هم به پین A2 و LM335 هم به پین A0.

```
#define LDRPin A1
#define MoistureSignal A2
#define LM335Pin A0
```

در این قسمت یوزرنیم و پسورد مربوط به سرور local را به صورت const وارد می کنیم. همچنین IP و Port ای که از قبل اختصاص دادیم را به نام server و port ذخیره می کنیم.

```
const char* username = "reyhaneh";
const char* password = "13791379";
const char* server = "192.168.1.102";
const int port = 5000;
```

با این دستور، از یک software serial برای اتصال به ESP استفاده می کنیم.

```
SoftwareSerial ESP(ArduinoRX, ArduinoTX);
```

سپس یک تابع boolean تعریف کردیم. در داخل آن توسط دستور AT+CWLAP لیست همه‌ی access point ها یا همان نقاط دسترسی را به دست آورده و با serial.println آن را نمایش می‌دهیم. سپس در یک حلقه‌ی while چک می‌کنیم که تا زمانی که EPS فعال باشد، دقیقاً همان چیزی که دریافت می‌کند را به عنوان خروجی نشان دهد.

```
bool ConnectToWiFi() {
    ESP.print("AT+CWLAP\r\n");
    delay(2000);

    Serial.println("Available access points (WiFi connections): ");

    while (ESP.available()) {
        Serial.write(ESP.read());
    }
}
```

در ادامه، می‌خواهیم تمام SSID های موجود را نمایش دهیم و بعد از اینکه پیدا شد، آن را در یک رشته قرار می‌دهیم و توسط دستور Serial.readString آن را دریافت می‌کنیم. البته برای زمانی که هنوز هیچ ورودی‌ای وجود ندارد یک حلقه‌ی while گذاشته شده که در آن بماند.

```
Serial.println("SSID: ");
while (!Serial.available())
    ;
String ssid = Serial.readString();
```

در مورد پسورد نیز به همین ترتیب عمل کردیم، یعنی اگر پسورد را وارد کرده باشیم می‌توانیم از while عبور کنیم و با یک string مقدار آن را ذخیره کنیم.

```
Serial.println("Password: ");
while (!Serial.available())
    ;
String password = Serial.readString();
```

این جا هم با ESP.print اسم شبکه network به همراه password را می نویسیم.

```
ESP.print("AT+CWJAP=");  
ESP.print(ssid);  
ESP.print(",");  
ESP.print(password);  
ESP.print("\r\n");
```

در نهایت در این تابع بولین، مقدار تاخیر را 2 ثانیه در نظر می گیریم و چک می کنیم که آیا ESP در جواب AT command به ما OK را برگردانده یا خیر. (چون می دانیم در این صورت ماژول آماده به کار است.) با Serial.println این موضوع را پرینت می کنیم تا از وضعیت مطلع باشیم و در نتیجه تابع ما مقدار true را برمی گرداند ولی در غیر این صورت false را می فرستد به این مفهوم که تلاش برای اتصال به شبکه ناموفق بوده است.

```
delay(2000);  
if (ESP.find("OK")) {  
    Serial.println("Ready.");  
    return true;  
} else {  
    Serial.println("Could not connect, try again");  
    return false;  
}
```

در مرحله بعد برای initialize کردن ESP یک تابع void داریم. در این تابع شروع کار با مقدار دهی baud rate به ESP است. البته در این مرحله نیز باید وجود ماژول و قطعیت از آماده بودن آن دوباره مورد تست قرار گیرد. (با AT)

```
inline void InitESP() {  
    ESP.begin(ESPBAUD);  
    // Check for module existence  
    ESP.print("AT\r\n");  
    delay(1000);  
  
    if (ESP.find("OK")) {  
        Serial.println("Ready.");  
    } else {  
        Serial.println("ESP not found");  
        return;  
    }  
}
```

در نهایت اگر ESP توانست IP را شناسایی کند، پیامی مبنی بر "اتصال به WIFI" نمایش داده می‌شود و تا زمانی هم که به WIFI متصل نشده در همین حالت باقی می‌ماند چرا که حلقه‌ی while را ترک نمی‌کند.

```
if (ESP.find("GOT IP")) {  
    Serial.println("Connected to WIFI");  
} else {  
    Serial.println("Could not connect to WIFI");  
  
    while (!ConnectToWIFI())  
        ;  
}
```

در اینجا با دستور AT+CWMODE=3 ماژول در حالت سرور و کلاینت قرار می‌گیرد و با این کار می‌تواند با سرور ارتباط داشته باشد. و اگر باز هم خروجی OK را برگرداند یعنی این mode مورد نظر تنظیم شده است.

```
ESP.print("AT+CWMODE=3\r\n");  
delay(500);
```

```
if (ESP.find("OK")) {  
    Serial.println("Mode set.");  
}
```

در بخش بعدی با اجرای دستور AT+CIPMUX=1 تعداد حداکثر اتصالات Http ماژول به 1 تغییر می‌کند. چرا که دستور CIPMUX در واقع دستگاه را برای یک اتصال single یا multiple تنظیم می‌کند.

```
ESP.print("AT+CIPMUX=1\r\n");  
delay(500);  
if (ESP.find("OK")) {  
    Serial.println("Connections set.");  
}
```

اگر تا این جا تمام مراحل به درستی طی شده باشند چنین پیامی به نمایش درمی‌آید:

```
Serial.println("ESP initialization comepleted successfully");
```

این قسمت مربوط به خود سنسورها و دریافت اطلاعات آنها می‌باشد. به این ترتیب که طبق اتصالات ما در آردوینو، مقدار سنسور LDR را به صورت اختلاف آن با max خودش یعنی 1023 می‌خوانیم تا رابطه‌ی دما و روشنایی مستقیم شود. برای Lm335 که سنسور دما است، برای تبدیل ولتاژ خروجی به دما باید آن را در یک نسبت تبدیل ضرب کنیم.

```
unsigned long light = 0, moisture = 0, temperature = 0;

void readSensors() {
    temperature = ((analogRead(LM335Pin) / 1024.0) * 5000) / 10;
    light = 1023 - analogRead(LDRPin);
    moisture = 1023 - analogRead(MoistureSignal);
}
```

در این بخش اطلاعاتی که از سنسورها جمع آوری شده بود به ESP ارسال می‌شوند.

دوباره یک تابع بولین تعریف می‌کنیم و برای مقادیری که سنسورها قرار است ارسال کنند، متغیرهای اولیه را تعریف می‌کنیم.

```
void ESPFlush() {
    while (ESP.available() > 0) {
        char t = ESP.read();
    }
}
```

```
bool sendSensorValues() {
    char data[50];
    char request[300];
    char command[50];

    ESPFlush();
```

دستور AT+CIPSTART یک اتصال UDP یا TCP را برقرار می‌کند.

```
sprintf(command, "AT+CIPSTART=0,\"TCP\", \"%s\", %d\r\n", server, port);
ESP.print(command);
delay(500);
```


در مرحله بعدی مراحل Login کردن در سرور انجام میشود. برای اینکار باید دو مقدار username و password به سرور ارسال شود. شیوه ارسال داده یک درخواست HTTP به شیوه POST است.

```
/* Login */
sprintf(data, "username=%s&password=%s", username, password);
sprintf(request, "POST /auth/login HTTP/1.0\r\nContent-Length:%d\r\nContent-Type: application/x-www-form-urlencoded\r\n\r\n%s", strlen(data), data);
sprintf(command, "AT+CIPSEND=0,%d\r\n", strlen(request));
Serial.print(command);
ESP.print(command);

delay(500);

if (ESP.find(">")) {
  Serial.println("Ready to send data");
} else {
  Serial.println("Could not send data");
  return false;
}

ESP.print(request);

ESP.print("AT+CIPCLOSE \r\n");
```

در ادامه پس از ارسال این درخواست به سرور، sessionCookie از پاسخ سرور (که توسط ESP به آردوینو ارسال میشود) استخراج میشود. این مقدار مشخص کننده این است که ما به سرور Login کرده ایم و با هر درخواست که نیاز به Login دارد این مقدار باید ارسال شود

```
/* Register data */
String res;

if (ESP.find("Set-Cookie: session=")) {
  res = ESP.readString();
  int cookieIndex = res.indexOf(";");
  Serial.print(cookieIndex);
  res = res.substring(0, cookieIndex);

  Serial.println("Logged in successfully, session cookie: ");
  Serial.println(res);
}

sprintf(command, "AT+CIPSTART=0,\"TCP\", \"%s\", %d\r\n", server, port);
ESP.print(command);

sprintf(data, "i=0&light=%lu&moisture=%lu&temperature=%lu", light, moisture, temperature);
sprintf(request, "GET /api/insert_record HTTP/1.0\r\nContent-Length:%d\r\nContent-Type: application/x-www-form-urlencoded\r\nCookie: username=rejhaneh; session=%s\r\n\r\n%s", strlen(data), res.c_str(), data);
sprintf(command, "AT+CIPSEND=0,%d\r\n", strlen(request));

delay(500);

if (ESP.find("ERROR")) {
  Serial.print("Could not connect to server");
  return false;
}
```

در مرحله بعدی مراحل ارسال داده ها به سرور انجام میشود. برای اینکار باید مقادیر سنسور ها به سرور ارسال شود. شیوه ارسال داده یک درخواست HTTP به شیوه POST است. همچنین در این درخواست باید sessionCookie نیز همراه با درخواست HTTP ارسال شود.

```
delay(100);
ESP.print(command);

if (ESP.find(">")) {
  Serial.println("Ready to send data");
} else {
  Serial.println("Could not send data");
  return false;
}

ESP.print(request);

if (ESP.find("200")) {
  Serial.println("Sent data successfully");
} else {
  Serial.println("Sent data successfully");
  return false;
}

ESP.print("AT+CIPCLOSE \r\n");
}
```

2-4 نرم افزار های برای استفاده از کد:

نرم افزار های مورد نیاز برای استفاده از کد آردوینو: نرم افزار آردوینو قابل دانلود از وبسایت arduino.cc

نرم افزار های مورد نیاز برای استفاده از کد سرور: مفسر پایتون و نصب کتابخانه های (این لیست بطور آماده در پوشه کد سرور به نام requirements.txt موجود است و با استفاده از کامند `pip install -r requirements.txt` میتوان آنها را بطور یکجا نصب کرد):

```
attrs==21.4.0
autopep8==1.6.0
click==8.1.3
coverage==6.4.1
Flask==2.1.2
importlib-metadata==4.11.4
iniconfig==1.1.1
itsdangerous==2.1.2
Jinja2==3.1.2
MarkupSafe==2.1.1
packaging==21.3
pluggy==1.0.0
py==1.11.0
pycodestyle==2.8.0
pyparsing==3.0.9
pytest==7.1.2
toml==0.10.2
tomli==2.0.1
Werkzeug==2.1.2
zipp==3.8.0
```

2-5 نحوه تست عملکرد نرم افزارها:

برای تست سنسور ها، همانطور که در بند الف ذکر شد، سنسور ها را تک تک به شیوه توضیح داده شده به آردوینو متصل و تست کردیم. برای تست سرور ما از یک فریم ورک تست به نام `pytest` استفاده کردیم. برای استفاده از این فریم ورک تست در پوشه `tests` فایل های پایتون ایجاد کردیم که هر یک از حالت های درخواست های `http` را می سنجد. به عنوان مثال مثال برای تست `insert_record` حالت های زیر بررسی میشود:

- نبود یکی از پارامتر های ورودی
- اشتباه بودن پارامتر های ورودی
- بررسی صحت اطلاعات ثبت شده در دیتابیس
- بررسی وجود نداشتن داده های اشتباه و درخواست های رد شده در دیتابیس
- بررسی دستیابی به این درخواست بدون لاگین
- بررسی دستیابی به این درخواست با شیوه اشتباه (`insert_record` فقط با شیوه و روش `POST` به درستی کار میکند و روش `GET` را باید رد کند)

برای بررسی اتوماتیک همه این تست ها، در پوشه `tests` باید کامند زیر را اجرا کرد:

```
python -m pytest -vv
```

که به ما نشان میدهد کدام یک از تست ها به درستی پاس شده و کدام تست ها رد شده اند.

تمامی مواردی که توسط این روش تست میشوند، در جدول زیر ذکر شده اند.

مورد تست	تست های انجام شده
<code>api/insert_record</code>	<ul style="list-style-type: none">• نبود یکی از پارامتر های ورودی• اشتباه بودن پارامتر های ورودی• بررسی صحت اطلاعات ثبت شده در دیتابیس• بررسی وجود نداشتن داده های اشتباه و درخواست های رد شده در دیتابیس• بررسی دستیابی به این درخواست بدون لاگین• بررسی دستیابی به این درخواست با شیوه اشتباه• بررسی درستی ارور بازگردانده شده به کاربر
<code>api/fetch_recent_record</code>	<ul style="list-style-type: none">• نبود یکی از پارامتر های ورودی• اشتباه بودن پارامتر های ورودی• بررسی دستیابی به این درخواست بدون لاگین• بررسی دستیابی به این درخواست با شیوه اشتباه

	<ul style="list-style-type: none"> • بررسی صحت داده های بازگردانده شده
auth/register	<ul style="list-style-type: none"> • نبود یکی از پارامتر های ورودی • اشتباه بودن پارامتر های ورودی • ضعف پسورد • وجود کاراکتر های غیرمجاز در پارامترها • بررسی صحت ارور بازگردانده شده به کاربر • بررسی ایجاد کاربر تکراری • بررسی تغییر نکردن رمز کاربر تکراری در صورت سعی ایجاد کاربر تکراری • بررسی دستیابی به این درخواست با شیوه اشتباه • بررسی واکنش سرور به ایجاد کاربر توسط کاربر ورود کرده
auth/login	<ul style="list-style-type: none"> • نبود یکی از پارامتر های ورودی • اشتباه بودن پارامتر های ورودی • وجود کاراکتر های غیرمجاز در پارامترها • بررسی صحت ارور بازگردانده شده به کاربر • بررسی سعی ورود با اطلاعات نادرست • بررسی دستیابی به این درخواست با شیوه اشتباه • بررسی واکنش سرور به ورود مجدد به سرور
auth/logout	<ul style="list-style-type: none"> • بررسی دستیابی به این درخواست با شیوه اشتباه • بررسی دستیابی به این درخواست بدون لاگین • بررسی پاک شدن نشست کاربر پس از Logout
DATABASE	<ul style="list-style-type: none"> • بررسی پاک شدن اتصال دیتابیس پس از خاموشی سرور • بررسی کامند ایجاد کننده دیتابیس

برای بررسی آردوینو ما از فریم ورک یا کتابخانه خاصی استفاده نکردیم و آن را در شرایط مختلف تست کردیم:

مورد تست	موارد تست شده
ESP	<ul style="list-style-type: none"> • نبود ESP • فعال نبودن ESP • بودن ESP در deep sleep • اتصال به نقاط اتصال مختلف (Hotspot، Router و ...) • خاموشی سرور • در دسترس نبودن سرور • نبود دیتابیس در سرور
LM335	<ul style="list-style-type: none"> • بررسی کارکرد در حالت کالیبره شده و نشده • بررسی خطای سنسور مقابل چندین سنسور دیگر • بررسی با منابع تغذیه مختلف
LDR	<ul style="list-style-type: none"> • بررسی درستی افزایش و کاهش ولتاژ و خروجی نسبت به نور • بررسی با منابع تغذیه مختلف • بررسی با مقاومت های مختلف در مدار
YL - 69	<ul style="list-style-type: none"> • بررسی با منابع تغذیه مختلف • بررسی واکنش در رطوبت بسیار بالا

2-6 مرجع و لینک کد:

<https://github.com/ReyhanehAhani/IoTServer>

<https://github.com/ReyhanehAhani/ArduinoIoTClient>

<https://github.com/ReyhanehAhani/ThingsBoardArduino>

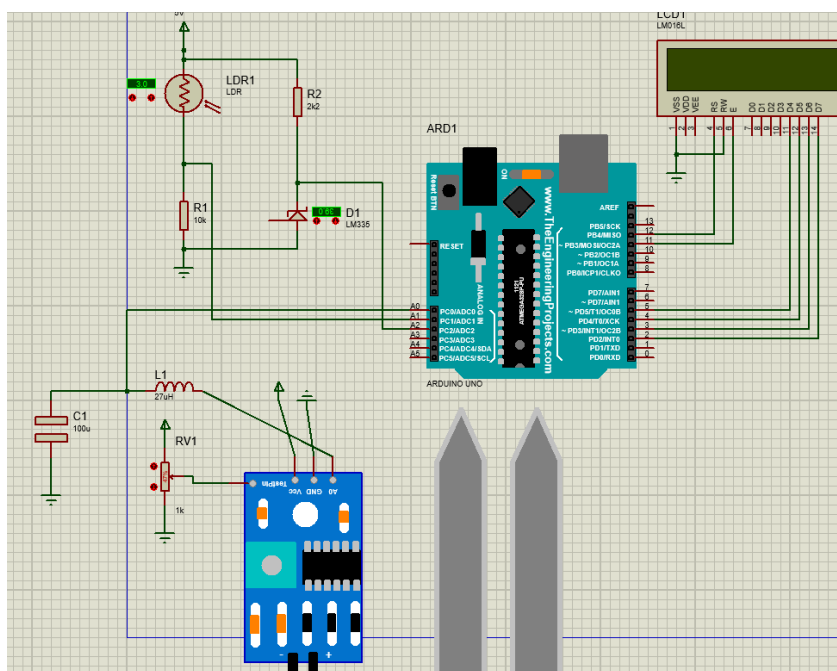
سازنده ریحانه آهنی

فصل سوم: طراحی و شبیه سازی در پروتئوس

3-1 طراحی مدار:

می خواهیم در این بخش مدار و اتصالات سنسور ها به آردوینو را شبیه سازی کنیم. در این شبیه سازی به برد آردوینو، سنسور رطوبت YL-69، سنسور نور LDR MLG5516 و سنسور دما LM335 و LCD نیاز داریم که به غیر از برد آردوینو و سنسور رطوبت، سایر موارد را کتابخانه های پروتئوس موجود دارند. بنابراین در ابتدا کتابخانه های (library) آردوینو و سنسور رطوبت را دانلود کرده و اضافه می کنیم.

سپس با سرچ نام المان های مدنظر آن ها را بر روی صفحه آورده و اتصالات را مطابق مدار بهسازی انجام می دهیم:



همانطور که از مدار مشخص است برای سنسور های دما و نور از یک منبع ولتاژ dc به اندازه 5 ولت و مقاومت های متناسب استفاده کردیم و به ترتیب به پایه های A2 و A1 برد آردوینو متصل کردیم. برای سنسور رطوبت برای اینکه بتوانیم مقدار رطوبت را تغییر دهیم باید مقدار ولتاژ پایه testpin را تغییر داد که به این منظور از یک پتانسیومتر استفاده می کنیم و همچنین در حالت شبیه سازی برای اتصال پایه سنسور به پایه برد (A0) به یک فیلتر LC نیاز داریم. اتصالات LCD را نیز مطابق با دیتاشیت انجام می دهیم به این گونه که پین R/W را به زمین وصل کرده چون فقط قصد نوشتن بر روی LCD را داریم و پین های reset و enable و همچنین 4 پین d4-d7 را به آردوینو متصل می کنیم.

2-3 شبیه سازی مدار:

در ادامه در نرم افزار Arduino IDE کد مدنظر را برای خواندن اطلاعات سنسور ها و نمایش آن بر روی LCD می نویسیم. در ابتدا کتابخانه مربوط به LCD را اضافه میکنیم و سپس پین های مدنظر که شامل 4 پین خروجی d4-d7 و پین های reset و enable را فعال می کنیم. همچنین پایه های آردوینو متصل به سنسورها را مطابق مدار پروتئوس تعریف می کنیم.

در قسمت setup توسط دستور lcd.begin(cols,rows) ، LCD را به همراه تعداد سطر ها و ستون هایش به آردوینو معرفی می کنیم. همچنین با دستور Serial.begin(9600) سرعت (baud rate) آردوینو را مشخص میکنیم. و در آخر نیز پین های متصل به سنسور ها را به حالت ورودی قرار می دهیم.

```
#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int ldrPin = A1;
int Moisture_signal = A0;
int LM335_pin = A2;

void setup() {
  lcd.begin(16, 2);
  Serial.begin(9600);
  pinMode(ldrPin, INPUT);
  pinMode(Moisture_signal, INPUT);
  pinMode(LM335_pin, INPUT);
}
```

در قسمت loop ابتدا با دستور lcd.setCursor(0,0) موقعیت مکان نما را به سطر و ستون صفر میبریم و در ادامه با خواندن مقادیر سنسور ها (با دستور analogRead) مقادیر هر سنسور را با دستور lcd.print روی LCD نمایش می دهیم.

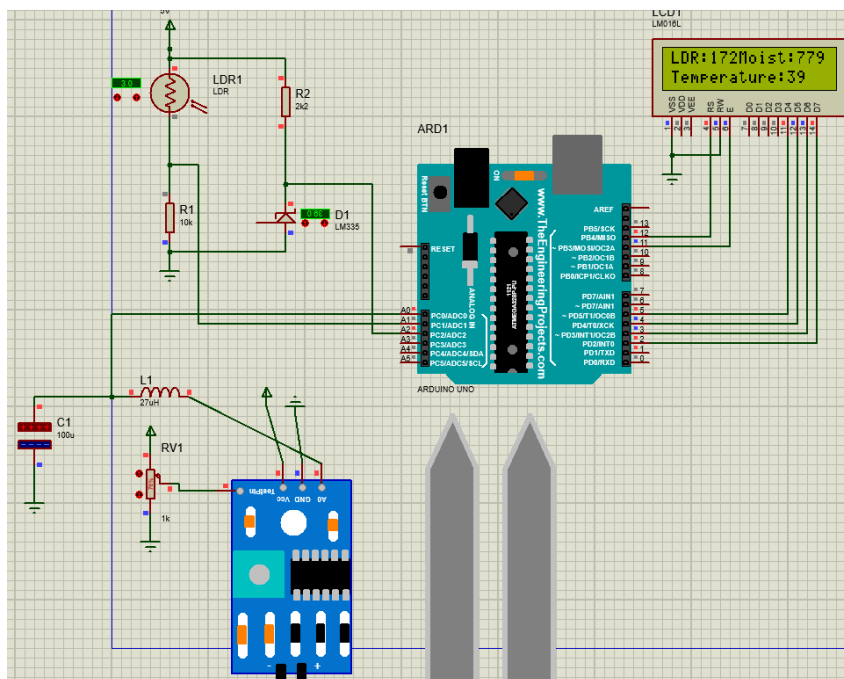
```
void loop() {
  lcd.setCursor(0, 0);
  int ldrStatus = analogRead(ldrPin);
  lcd.print("LDR:");
  lcd.print(ldrStatus);

  int Moisture = analogRead(Moisture_signal);
  lcd.print("Moist:");
  lcd.print(Moisture);

  int Kelvin = analogRead(LM335_pin) * 0.489; // Read analog voltage and convert it to Kelvin (0.489 = 500/1023)
  int Celsius = Kelvin - 273;
  lcd.setCursor(0, 1);
  lcd.print("Temperature:");
  lcd.print(Celsius);
  |
  delay(1000);
}
```

سپس کد را کامپایل کرد و با کلیک بر روی `export compile binary` فایل `sketch>>hex` را ذخیره کرده و با دابل کلیک بر روی آردوینو، کد را بر روی برد می ریزیم.

پس از اجرا برنامه مشاهده می شود که مقادیر خوانده شده توسط سنسور ها روی LCD نمایش داده می شود که با تغییر مقادیر سنسور ها نیز مقادیر روی LCD تغییر می کنند.



3-3 مرجع کد و فایل شبیه سازی:

[لینک کد آردوینو](#)

[لینک فایل پروتئوس](#)

سازنده فاطمه رفیعی و سمیرا سلجوقی

فصل چهارم: لینک کلیپ های تهیه شده

[فیلم تست سخت افزار و نرم افزار پروژه](#)

[فیلم توضیحات بخش نرم افزار پروژه \(وب اپلیکیشن\)](#)

[فیلم تست سنسور ها به صورت جداگانه به همراه کد](#)