آشنایی با پروتکل های ارتباطی انتقال داده و سنسور ها

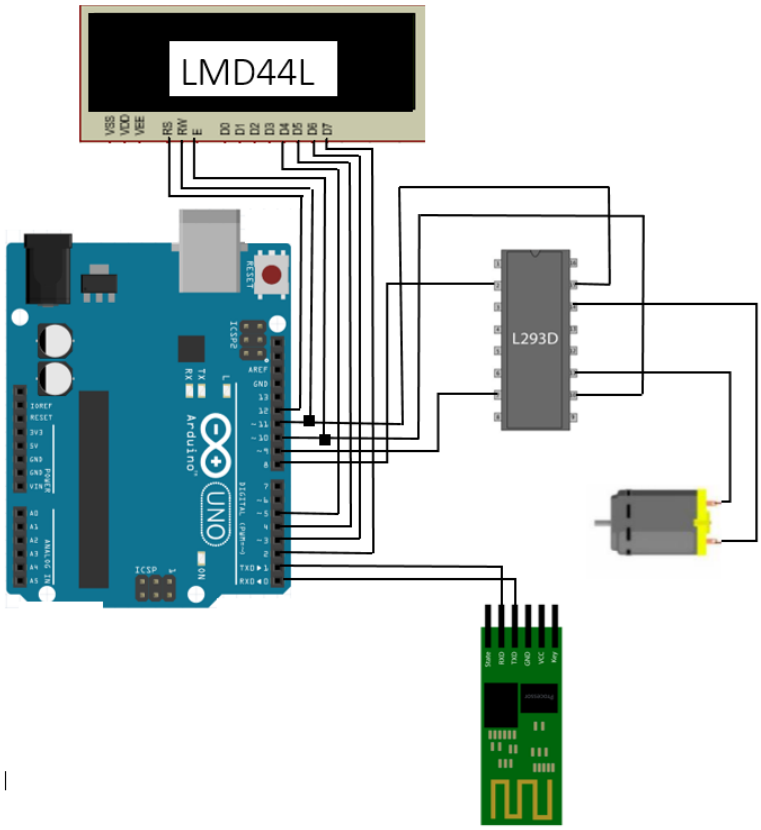
ایمان مرادی 810196560
نسترن علی پور 810196515
پارسا صدری سینکی 810195526
رامین فریاد 810195447

طراحی مفهومی

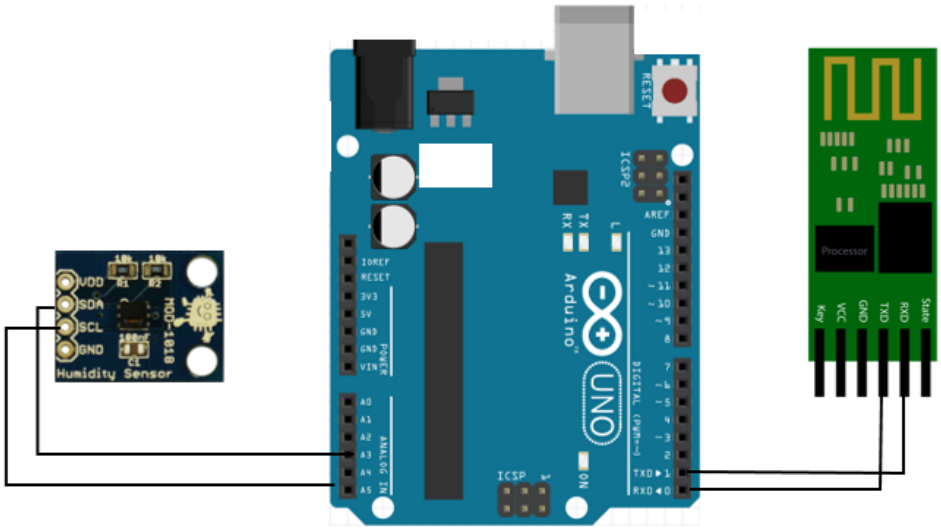
در ابتدا ، یک طراحی مفهومی از سیستم تعریف شده، ایجاد میکنیم. این طراحی به ما کمک میکند تا سیستم را بهتر درک کنیم.

سیستم شامل دو برد است و هر برد شامل یک آردوینو میشود. قصد داریم سیستمی طراحی کنیم که آردوینو از برد master ، آبرسانی گلدان را کنترل کند. کنترل آبرسانی از طریق یک موتور DC تحت کنترل این آردوینو فراهم می گردد. به این صورت که آردوینو master اطلاعات دما و رطوبت را از برد slave دریافت میکند و با توجه به این داده ها، سرعت موتور را تنظیم میکند و همچنین داده ها را در یک LCD نمایش میدهد.

بنابر این در برد master به یک آردوینو ، یک LCD متصل به آن ، موتور DC (برای کنترل این موتور ، یک درایور (L293d) و یک قطعه بلوتوث از مدل HC-05 احتیاج داریم. صفحه ی LCD که از مدل LMD44L است ، به همراه بلوتوث ، مستقیما به آردوینو متصل میشوند تا ورودی های لازم به آنها از خروجی های آردوینو ، ارسال شود و تحت کنترل آردوینو باشند. به این منظور، پورت های ۴،۵،۶،۱۱،۱۲،۱۳،۱۴ از LCD را به پورت های ۱۲،۱۱،۱۰،۵،۴،۳،۲ آردوینو متصل میکنیم. پورت های TXD, RXD را نیز به ترتیب به پورت های خروجی RXD, TXD آردوینو متصل میکنیم. برای کنترل موتور DC از یک L293d مشابه پروژه قبل استفاده میکنیم. به این صورت که پورت enable درایور L293d را به پورت 10 که از پورت های pwm آردوینو است، متصل میکنیم و input1, input2 را به پورت ۶،۷ آردوینو متصل میکنیم و خروجی های درایور را به موتور متصل می کنیم. طراحی مفهومی برد master به صورت زیر است:



برد slave نیز دارای یک آردوینو ، یک قطعه بلوتوث HC-05 و یک سنسور رطوبت و دمای SHT25 است. در این برد با دریافت درخواست برد master مبنی بر دریافت اطلاعات دما و رطوبت گلدان، سنسور SHT25 را به آردوینو از طریق رابط I2C متصل میکنیم و آردوینو این اطلاعات را در پورت های RXD, TXD خروجی میدهد. قطعه ی بلوتوث به قطعه ی بلوتوث دیگر در برد master , متصل میشود تا اطلاعات را ارسال کند. بنابراین نیاز است تا پورت های RXD, TXD آردوینو به پورت های TXD, RXD متصل شوند. طراحی مفهومی برد slave به صورت زیر است:



Slave Board

SHT25

یکی از کامپوننت های به کار رفته در این شبیه سازی سنسور sht25 است که با استفاده از آن دما و رطوبت یک محیط را میتوان اندازه گیری کرد. برای کار با این سنسور به مستندات آن رجوع کردیم. این سنسور میتواند دما و رطوبت را اندازه گیری کرده و به مدار دیگری که با استفاده از آن قصد دارد اقدام خاصی (مانند کم کردن درجه ی حرارت و ...) را انجام دهد بفرستد. خروجی این سنسور برای تبدیل شدن به مقادیر مورد نیاز باید در فرمول خاصی قرار بگیرند:

Relative Humidity Conversion:

$$RH = -6 + 125 * \frac{S_{RH}}{2^{16}}$$

Temperature Conversion:

$$T = -46.85 + 175.72 * \frac{S_T}{2^{16}}$$

بنابراین نیاز است که پس از خواندن داده های ورودی آمده در آردوینو این تبدیل را انجام دهیم تا به مقدار رطوبت و دما (بر حسب درجه ی سلسیوس) برسیم.

Arduino Slave

مدار آردوینو ی TH Board از دو قسمت تشکیل شده است. اولین قسمت وظیفه ی خواندن داده ها از سنسور و ذخیره آنها را برعهده دارد و قسمت دوم وظیفه ی دریافت دستور ارسال داده ها و رتباط با برد اصلی و فرستادن داده های ذخیره شده ی دما و رطوبت با استفاده از بلوتوث است.

ارتباط با سنسور:

برای ارتباط با سنسور از کتابخانه ی Wire استفاده میکنیم. این ماژول از پروتکل سریال I2C برای ارتباط استفاده میکند. برای setup این ماژول، پس از include کردن کتابخانه ی آن در فایل، تابع begin آن را در تابع setup اجرا میکنیم. حال با استفاده از تابع Wire.beginTransmission میتوان در برد آردوینو به سنسور وصل شد. آدرس ثبت شده برای وصل شدن به این برد در Arduino، آدرس 0x40 است. بنابراین باید این تابع را به صورت زیر فراخوانی کرد:

```
Wire.beginTransmission(0x40)
```

فراخوانی این تابع داده ای ارسال نمیکند و برخی از داده های کتابخانه ی Wire را برای state آن تغییر میدهد. حال میتوان با استفاده از تابع write از سنسور درخواست کرد که مقدار داده ی رطوبت و یا دما را ارسال کند. برای درخواست ارسال رطوبت تابع write را با مقدار 0xF5 فراخوانی می کنیم. سپس Wire.endTranmision را فراخوانی کرده تا این درخواست ارسال شود. مشابه تابع beginTransmision که برای آغاز ارسال داده ها به یک کامپوننت با آدرس خاص استفاده میشود، از تابع requestFrom برای خواندن داده ها از یک آدرس(آدرس کامپوننت) به کار میرود. این تابع آدرس کامپوننتی که میخواهیم داده ها از آن خوانده شوند و تعداد دیتا فریم هایی که باید خوانده شوند را گرفته داده های دریافتی را ذخیره میکند. لازم به ذکر است که این تابع و تابع write در واقع یک تابع ساده نیستند و در هر کدام از آنها توابع سطح پایین تری اجرا و پردازش می شوند تا ارسال داده ها به شکل گفته شده طبق پروتکل I2C انجام پذیرد. (به عنوان مثال خود تابع requestFrom تابع beginTransmision را فراخوانی می کند)

با استفاده از تابع Wire.available می توان تعداد فریم های درخواستی را به دست آورد و متوجه شد که آیا تعداد فریم های دریافت شده با تعداد فریم های رسیده مطابقت دارد یا خیر. با استفاده از تابع Wire.read هم میتوان از بافری که در کتابخانه ی Wire قرار داده شده و داده های دریافتی را در آن ذخیره میکند، داده ها را گرفت.

حال برای گرفتن مقدار رطوبت باید تابع

```
Wire.requestFrom(0x40, 2)
```

را فراخوانی کرد (قبل از آن یک beginTransmision با همین آدرس و یک write با مقدار 0xf5 صورت گرفته است که برای مشخص کردن این است که سنسور می بایست مقدار رطوبت را برگرداند). 0x40 آدرس سنسور sht25 و مقدار 2 مشخص کننده ی تعداد فریم های ارسالی است. حال با استفاده تابع Wire.read فریم های ارسالی را دریافت کرده و از آنجا که داده ها به ترتیب از most significant bit به least significant bit ارسال شده اند می توان با ضرب فریم اول در 2^8 و جمع آن با فریم دوم به مقدار رطوبت ذخیره شده در سنسور رسید حال کافی است که این عدد را در فرمول ذکر شده قرار داد تا رطوبت را به دست آوریم. این مقدار را در متغیری ذخیره میکنیم.

مشابه همین دستورات برای گرفتن دما از سنسور به کار میرود.
برای درخواست ارسال مقدار دما همین تابع را با مقدار 0xF3 اجرا میکنیم. بنابراین از توضیح مجدد صرف نظر میکنیم. تنها تفاوت فرمول محاسبه ی دما است.

قسمت دیگر برد slave مربوط به ارتباط با برد اصلی است. طراحی به این صورت است که برد اصلی دستور دریافت دما و رطوبت را به برد slave میدهد. برد اسلیو دستور دریافتی را پردازش میکند تا نوع دستور را تشخیص دهد(هر چند در این پروژه تنها یک دستور داریم ولی باز هم این کار هم از لحاظ کد نویسی و هم از لحاظ تشخیص خطا لازم است) بعد از خواندن دستور که در تابع recvCommand انجام میشود تابع processCommand فراخوانی میشود تا دستور دریافتی را پردازش کرده و پاسخ مناسب را برگرداند. ابتدا پاسخ مناسب را تولید کرده و سپس پاسخ آماده شده به فرمت زیر را به برد اصلی میفرستیم:

Response: <T:H>

که T دما و H رطوبت است. ارسال داده ها در این قسمت با استفاده از کتابخانه ی Serial اتفاق می افتد و در مدار هم با استفاده از کامپوننت HC-05 داده ها بین دو برد اصلی و slave رد و منتقل می شوند. این انتقال با استفاده از پروتکل UART انجام می شود. برای استفاده از کتابخانه ی Serial کافی است که در تابع begin برد اسلیو Serial.begin را فراخوانی کرده و برای خواندن از آن از Serial.read که یک ورودی از سر بافر برمیگرداند و برای نوشتن از Serial.write که برای نوشتن یک آرایه از کاراکتر ها (یک استرینگ)، استفاده می شود. همچنین برای فهمیدن مقدار داده های موجود در بافر میتوان از Serial.available استفاده کرد.

Main Board

برد اصلی هر 5 ثانیه یک بار به TH Board توسط بلوتوث یک درخواست می فرستد تا داده های دما و رطوبت را دریافت کند.
نحوه ارسال یا دریافت توسط کامپوننت HC-05 مشابه TH Board می باشد و با استفاده از دستورات Serial.read و Serial.write می باشد. ارتباط بین دو ماژول بلوتوث توسط ساخت و اتصال دو پورت مجازی COM3 و COM4 صورت گرفت با استفاده از برنامه com0com.

برای حل این مشکل که یک گره از نیمه یک پیام را دریافت کند ما ابتدا و انتهای پیام های ارسالی توسط ماژول بلوتوث را با < و > مشخص کرده ایم تا دما و رطوبت بصورت ناقص خوانده نشوند.

در مرحله بعد بعد از دریافت داده های دما و رطوبت برد اصلی باید این رشته ها را به اعداد اعشاری تبدیل کند و تصمیم بگیرد که با چه سرعتی باید آبیاری صورت بگیرد. با توجه به چهار حالت ذکر شده در صورت پروژه برد سرعت موتور را تنظیم می کند و پیام مناسب را بر روی LCD نمایش می دهد.

تنظیم سرعت برد مشابه پروژه قبل و با استفاده از یک درایور L293d انجام شد. به این صورت که برای تولید duty cycle 25% از عدد 64 و برای تولید duty cycle 10% از عدد 25 روی پین enable درایور استفاده شد.

آخرین قسمت برد اصلی بخش نمایش روی LCD است که با استفاده از کتابخانه LiquidCrystal انجام می شود. ابتدا باید پین های استفاده شده آردوینو برای نمایش روی LCD را مشخص کنیم. سپس در قسمت setup باید سایز LCD را به تابع lcd.begin بدهیم.

برای نوشتن روی LCD کافیست از تابع print روی متغیر lcd صدا بزنیم همراه با آرگومان مورد نظر برای چاپ. برای مثال برای نوشتن hello world کد مقابل را استفاده می کنیم.

```
lcd.println("hello world");
```

برای مشخص کرد این که در سطر اول یا سطر دوم LCD می نویسیم هم از تابع مقابل استفاده می کنیم. که در اینجا 0 نشان دهنده ستون و 1 نشان دهنده سطر مورد نظر هست.

```
lcd.setCursor(0, 1);
```

برای این که نوشته های قبلی پاک شوند و با نوشته های جدید تداخلی نداشته باشند هنگام چاپ نوشته ای جدید ابتدا تابع زیر را فراخوانی می کنیم.

```
lcd.clear();
```

پرسش ها

1- بلوتوث از رنج فرکانسی 2402 تا 2481 MHz برای ارتباط بی سیم استفاده می کند که مشابه فرکانس Wifi هست.

بلوتوث در رنج فرکانسی مشخص شده، دارای 79 کانال 1 MHz است که کافیست هر دو بلوتوث درحال مکالمه از یکی از این کانال ها استفاده کنند تا تداخلی صورت نگیرد.

برای این که از یک کانال مشابه دو دست دستگاه به طور اتفاقی استفاده نکنند و باعث تداخل نشوند بلوتوث از تکنیکی به نام spread-spectrum frequency hopping استفاده می کند. این تکنیک به این صورت هست که با نرخ 1600 بار در ثانیه کانال مورد استفاده را تغییر می دهد تا میزان زمان تداخل با سیگنال های دیگر را به حداقل برساند. تداخل در کل زمان ارتباط بی سیم تنها در حالتی می تواند اتفاق بیفتد که یک دستگاه بلوتوث دقیقا با همان پترن بین کانال ها جابجا شود که احتمال این اتفاق بسیار ناچیز می باشد.

2- بله امکانش وجود دارد. چون هر سنسور آدرس منحصر بفرد خودش را دارد می توانند همه به یک پورت وصل شوند و هر دستگاه فقط پکت های با آدرس خودش را دریافت کند.

هر دستگاهی که بخواهد از bus استفاده می کند تا وقتی سیگنال Stop را ندیده است یعنی bus در اختیار کسی دیگر است و نمی تواند از آن استفاده کند.

مشکل زمانی بوجود می آید که دستگاهی سیگنال Start را ندیده باشد و فکر کند bus آزاد هست و بخواهد از آن استفاده کند. برای همین هر فرستنده ای روی bus باید داده ای را که روی bus قرار داده با آن جه روی آن هست را همیشه مقایسه کند و در صورتی که متفاوت باشد دیگر bus در اختیار او نیست و باید منتظر سیگنال Stop بماند.

علت این اتفاق این است که چون طراحی باس I2C به صورت wired-and هست اگر یک دستگاه آن را روی صفر بگذارد روی صفر می ماند و دستگاه دیگری بخواهد روی bus مقدار غیر از صفر بگذار، bus تغییر نمی کند.