	به نام خدا	

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

پروژه اول درس سیستم‌های نهفته بی‌درنگ

نام و نام‌خانوادگی	**سارا رضایی‌منش-نرگس غلامی-آوا کاظمی‌نژاد-ریحانه احمدپور**
شماره دانشجویی	**810198494-810197675-810198447-810198576**
تاریخ ارسال گزارش	

منطق کلی برنامه

کل پروژه به دو بخش تقسیم می‌شود، طراحی دستگاه‌ها و نوشتن کد برای آن‌ها.

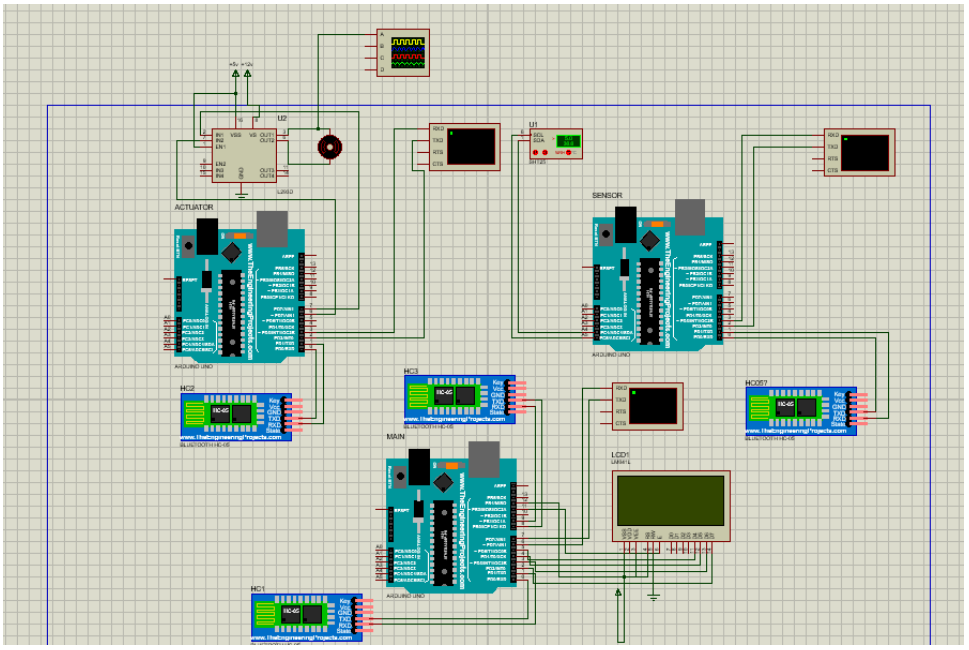
برای شروع طراحی ماژول‌های گفته شده را از اینترنت برمی‌داریم و به کمک راهنمایی‌های موجود در اینترنت آنها را به هم وصل می‌کنیم.

برای این پروژه ما سه ماژول کلی داریم که برای هر کدام در برنامه یک بورد آردوئینو در نظر گرفته شده است:

۱. cps: که همان گره‌ی مرکزی است. به این گره دو بلوتوث متصل است که یکی از آن‌ها با سنسور و دیگری با عملگر در ارتباط است. (در وسط طراحی قرار گرفته است.)

۲. sensor: گره‌ی حس کننده‌ی دما می‌باشد که اطلاعات لازم را از طریق بلوتوث برای گره‌ی مرکزی ارسال می‌کند.(سمت راست طراحی قرار دارد)

۳. actuator: گره‌ی عملگر می‌باشد که با توجه به دستوری که از سمت cps و با استفاده از بلوتوث به او می‌رسد، عمل مورد نظر را انجام می‌دهد. (سمت چپ طراحی قرار دارد)



برای هر کدام از این بوردهای آردوئینو یک کد نوشته شده است که شامل دو بخش setup و loop هست.

setup تعیین می‌کند که برای ارتباط با محیط طراحی چه کارهایی از قبل باید انجام شود و loop نیز کارهایی را که در هر دور در اجرای برنامه باید انجام شود را مشخص می‌کند.

هر کدام از این کدها یک پروژه‌ی platform IO هستند و خروجی هگز را در لوکیشن بیس بعد از build کردن تولید می‌کنند. سپس از این خروجی در بخش طراحی پروتئوس استفاده می‌کنیم، و می‌توانیم برنامه پروتئوس خود را ران کنیم.

برای کار کردن بلوتوث‌ها هم به یک برنامه دیگری نیاز داریم به نام virtual serial port driver که نام کامپوننت‌ها را برای اتصال می‌گیرد و نام هر دو جفتی که باید به هم وصل شوند را می‌گیرد و اتصال را برقرار می‌کند.

CPS

```
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>
#include <AltSoftSerial.h>
#include <Wire.h>
```

این همان Main است و کتابخانه اول برای ویرچوال مانیتور استفاده می‌شود که همان ترمینال سیاه پروتئوس می‌شود، دومی برای ال‌سی‌دی main است که قرار است رطوبت و دما و دیوتی سایکل را نشان دهد، سومی برای این است که بتواند داده‌ها را از کامپوننت‌های دیگر بخواند و بگیرد(که این برای ارتباط با سنسور هم هست)، کتابخانه‌ی wire برای این بخش استفاده نمی‌شود بلکه برای سنسور استفاده می‌شود.

```
#define HUMIDITY 0
#define TEMPERATURE 1
float info[2];// input format == humidity-temperature-dutycycle$
$
```

این آرایه دوتایی برای ذخیره‌ی داده‌های سنسور و سپس بررسی آنها و بدست آوردن نرخ آبیاری است که سپس در فرمت تعریف شده قرار می‌گیرد و بعد ارسال به اکچویتور را داریم، به این شکل که ایندکس صفرم طبق دیفاین همان رطوبت است و ایندکس اول همان دما می‌شود، هر دوی اینها از سنسور گرفته می‌شوند.

همین‌طور که در کامنت دیده می‌شود استایل فرستادن داده به عملگر به صورت زیر است:

humidity-temperature-duty cycle$

که داده‌ی اول و دوم(رطوبت و دما) همان‌طور که می‌دانیم از سنسور گرفته می‌شود، سپس بعد از بررسی نرخ آبیاری(که داخل آرایه info نیست، زیرا صرفا نرخ ارسالش از همان داده‌های گرفته شده از سنسور بررسی می‌شود و سپس در استرینگ ارسالی قرار می‌گیرد)، با «-» میان داده‌ها و «$» برای نشان پایان ارسال به عملگر اکچویتور فرستاده می‌شود.

ست‌آپ سیستم را داریم:

```
#define RXPIN 8
#define TXPIN 9
AltSoftSerial BTSerial(RXPIN, TXPIN);


#define ANALOG_WRITE_INTERVAL 255
SoftwareSerial virtualMonitor(7, 6); // RX | TX
int outputPin = 6;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  pinMode(outputPin, OUTPUT);
  Serial.begin(9600);
  virtualMonitor.begin(9600); // debug
  BTSerial.begin(9600);
  pinMode(RXPIN, INPUT);
  pinMode(TXPIN, OUTPUT);


  lcd.begin(16, 2);
  lcd.print("MAIN");
}
```

طبق RXPINوTXPIN که داریم، به ترتیب پین‌های گیرنده و ارسال ماژول پروتئوس را مشخص می‌کنیم.

AltSoftSerial BTSerial(RXPIN, TXPIN); این خط سنسور گیرنده‌ی پین‌ها برای خواندن از سنسور بلوتوث را فعال می‌کند.

SoftwareSerial virtualMonitor(7, 6); // RX | TX این خط هم سنسورهای دریافت ترمینالی را فعال می‌کند.

طبق فیلم یوتیوب صفحه نمایش main را تشکیل می‌دهیم.

این .begin(9600) به ترتیب برای خود مین و ترمینال مین و بلوتوث هست که تنظیمات فرکانس دریافتی دستگاه‌های استفاده شده است و مقداردهی سریالی ارتباطات است، که نرخش طبق لینک گفته شده به ۹۶۰۰ تنظیم شده است.

همچنین pinMode ها به ترتیب برای ترمینال و بلوتوث هستن، که ترمینال کامپوننت اصلی از آنجایی که نیازی نداشت به اینپوتی وصل باشد و صرفا خروجی متن ماژول است ازش یکی داریم.

تا به الان تنظیمات ابتدایی کامپوننت را دیدیم، حال بریم کارایی سیستم و نحوه حل مسئله‌اش را ببینیم:

```
void loop()
{
  char c = '\n';
  if (Serial.available()){
    c = Serial.read();
    entry += c;
  }
  if (c == '$'){
    virtualMonitor.println("MAIN: Received from sensor: ");
    float *sensorData = sensorInputProccess(entry);
    sendDataToActuator(sensorData);
    entry = "";
  }
}
```

به کمک متغیر c کاراکتر به کاراکتر از سنسور می‌خونیم تا جایی که به پایان اینپوت فرستادن سنسور که میشه دما و رطوبت برسیم، و در entry کل ورودی را ذخیره می‌کنیم. بعد دیدن $ یعنی ورودی تمام شده، پس در تابع sensorInputProccess داده‌ها را برای حساب کردن نرخ آبیاری بدست می‌آوریم، سپس با توجه به آرایه info نرخ آبیاری را محاسبه می‌کنیم و داده‌ها را به فرمتی که بالاتر توضیح دادیم درمی‌آوریم و به عملگر می‌فرستیم.

```
float *sensorInputProccess(const String &data)
{
  int splitIndex = 0;
  for (unsigned int i = 0; i < data.length(); i++)
    if (data[i] == '-')
      splitIndex = i;

  info[HUMIDITY] = data.substring(0, splitIndex).toFloat();
  lcd.setCursor(0, 1);
  lcd.println(info[HUMIDITY]);
  info[TEMPERATURE] = data.substring(splitIndex + 1,
data.length()).toFloat();
  lcd.println(info[TEMPERATURE]);
```

```
  return info;
}
```

اینجا در ابتدا تا اولین «-» پیش می‌رود تا مکانی که داده رطوبت قبل آن قرار دارد را بیابد، سپس ورودی را به قبل و بعدش تقسیم می‌کند که قبلش میشه رطوبت و بعدش میشه دما.

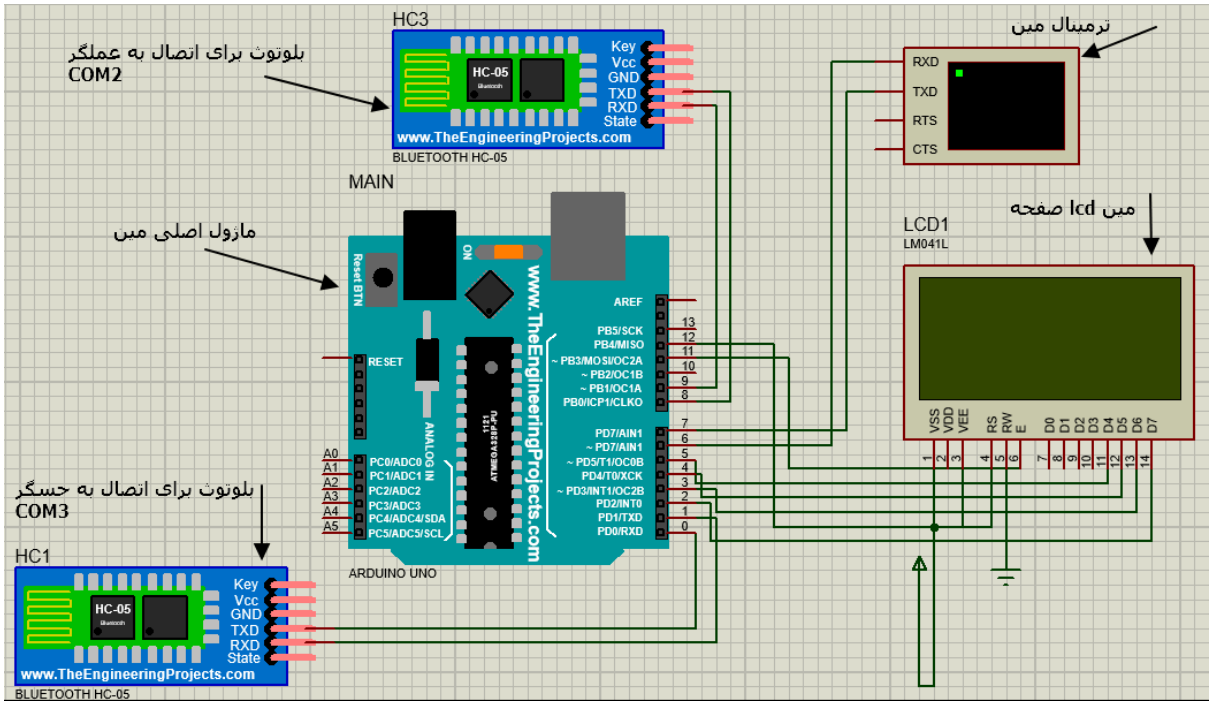
```
void sendDataToActuator(int dutyCycle){
  lcd.println(dutyCycle);
  BTSerial.print(info[HUMIDITY]);
  BTSerial.print("-");
  delay(100);
  BTSerial.print(info[TEMPERATURE]);
  BTSerial.print("-");
  delay(100);
  BTSerial.print(dutyCycle);
  BTSerial.print("$");
  delay(100);
  virtualMonitor.println("MAIN: sent to actuator!");
}

void sendDataToActuator(float* sensorData)
{
  if (info[HUMIDITY] < 10)
    sendDataToActuator(25);
  else if (info[HUMIDITY] >= 10 && info[HUMIDITY] < 20)
    sendDataToActuator(20);
  else if (info[HUMIDITY] >= 20 && info[HUMIDITY] < 30)
    if (info[TEMPERATURE] > 25)
      sendDataToActuator(10);
    else
      sendDataToActuator(0);
  else
      sendDataToActuator(0);
}
```

آرایه‌ای که بالاتر درست شد اینجا به عنوان ورودی استفاده می‌شود و شرایط گفته شده با توجه به رطوبت و دما برای محاسبه نرخ آبیاری اعمال می‌شود و در نهایت در تابع بالاتر داده‌ها به کمک بلوتوث ست شده برای اکچویتور برای عملگر فرستاده می‌شود.

تابع پایین که بررسی می‌کند اگر رطوبت کمتر از ۱۰ بود نرخ ۲۵، اگر بین ۱۰ تا ۲۰ بود نرخ ۲۰، اگر بین ۲۰ تا ۳۰ بود و دما بیشتر از ۲۵ درجه سلسیوس بود نرخ ۱۰ درصد شود غیر این حالت‌ها آبیاری صورت نگیرد که یعنی نرخ ۰ شود.

تابع بالایی که فرمت می‌کند داده‌ها را به بسته‌ای که عملگر آن را تحلیل کند، از طریق بلوتوث متصله به عملگر که COM2 هست و به COM4 عملگر می‌فرستد.



Sensor

کتابخانه‌ها توضیحاتی مشابه cps دارن.

کتابخانه wire.h که در این بخش کارایی دارد برای این است که بتواند نقش فرستنده بودن برای ارسال به شکل I2C را پیاده کند.

کتابخانه Arduino.h هم برای پیاده سازی دیلی‌ها به کار گرفته شده.

```
#define Addr 0x40

SoftwareSerial virtualMonitor(2, 3); // RX | TX

#define HUMIDITY 0
#define TEMPERATURE 1

float lastHumidity = 0;

void setup() {
  Wire.begin();
  Serial.begin(9600);
  virtualMonitor.begin(9600);
}
```

این آدرس دیفاینی تعریف استاندارد SHT25 برای I2C آدرس هست که برای ۶۴ بیتی این مقدار را دارد، این داده‌ها با توجه به لینک‌ها معرفی شده در صورت پروژه و جست‌وجو یافت شد.

تعریف virtualMonitor، دو دیفاین رطوبت و دما و نرخ ارتباطات مشابه است.

متغیر lastHumidity به دلیل شرط در صورت پروژه‌ست که گفته شده «گره حسگر پیوسته وضعیت گلدان را بررسی می‌کند و زمانی که رطوبتش به اندازه‌ی ۵ درصد آخرین رطوبت ارسال شده به مرکزی تغییر کرد مقادیر ارسال شود».

خط Wire.begin(); مقداردهی اولیه I2C برای ارتباطاتش است که به عنوان منبع شناخته شود.

```
bool isChanged(float newHumidity){
  return abs(lastHumidity-newHumidity) > 0.05*lastHumidity;
}
```

```
void loop() {
  float* sensorData = extractDataForMain();

  if(isChanged(sensorData[HUMIDITY])){
    sendToMain(sensorData);
    lastHumidity = sensorData[HUMIDITY];
  }
  delay(500);
}
```

طبق lastHumidity ای که توضیح داده شد، در تابع isChanged بررسی می‌شود اگر نسبت به پنج صدم درصد تغییرات داشتیم، باید به مین گفته شود.

در لوپ هم داده را به کمک I2C از برنامه می‌گیریم و با توجه به تحلیل‌ها و تبدیل‌ها آنها را به دیتاهای مورد تحلیل سلسیوسی/رطوبتی در می‌آوریم، و به مین می‌فرستیم.

تابعی که داده‌ی تحلیلی برای مین را می‌سازد extractDataForMain است، و تابعی که مشابه بخش قبل فرمت می‌کند داده‌های تحلیلی را به استرینگی که داده را بتواند انتقال دهد sendToMain است که مطابق فرمت :

humidity-temperature$

lastHumidity داده را به مین می‌فرستد و برای اینکه تغییرات را هم بررسی کند همان‌طور که گفته شد به آخرین رطوبت آپدیت می‌شود.

```
float calHumidity(unsigned int I2CInput1, unsigned int
I2CInput2){
  return (((I2CInput1 << 8 | I2CInput2) * 125.0) / 65536.0) - 6;
}

float calTemperature(unsigned int I2CInput1, unsigned int
I2CInput2){
  return (((I2CInput1 << 8 | I2CInput2) * 175.72) / 65536.0) -
46.85;
}
```

به ترتیب برای بدست آوردن رطوبت و دما هستن SHT25 این دو تابع محاسباتی طبق لینک استاندارد.

```
float* extractDataForMain() {
  float* result = new float[2];
  unsigned int I2CData[2];
```

```
  Wire.beginTransmission(Addr);
  Wire.write(0xF5);
  Wire.endTransmission();
  delay(500);

  Wire.requestFrom(Addr, 2);
   if(Wire.available() == 2) {
    I2CData[0] = Wire.read();
    I2CData[1] = Wire.read();
    result[HUMIDITY] = calHumidity(I2CData[0], I2CData[1]);
    virtualMonitor.print("Humidity is:");
    virtualMonitor.print(result[HUMIDITY] );
    virtualMonitor.println(" %RH");
  }
  Wire.beginTransmission(Addr);
  Wire.write(0xF3);
  Wire.endTransmission();
  delay(500);

  Wire.requestFrom(Addr, 2);

  if(Wire.available() == 2) {
    I2CData[0] = Wire.read();
    I2CData[1] = Wire.read();

    result[TEMPERATURE] = calTemperature(I2CData[0], I2CData[1]);

    virtualMonitor.print("Temperature is:");
    virtualMonitor.print(result[TEMPERATURE]);
    virtualMonitor.println(" C");
  }
  return result;
}
```

در Wire.beginTransmission(Addr); شروع انتقال داده توسط I2C را خواهیم داشت.

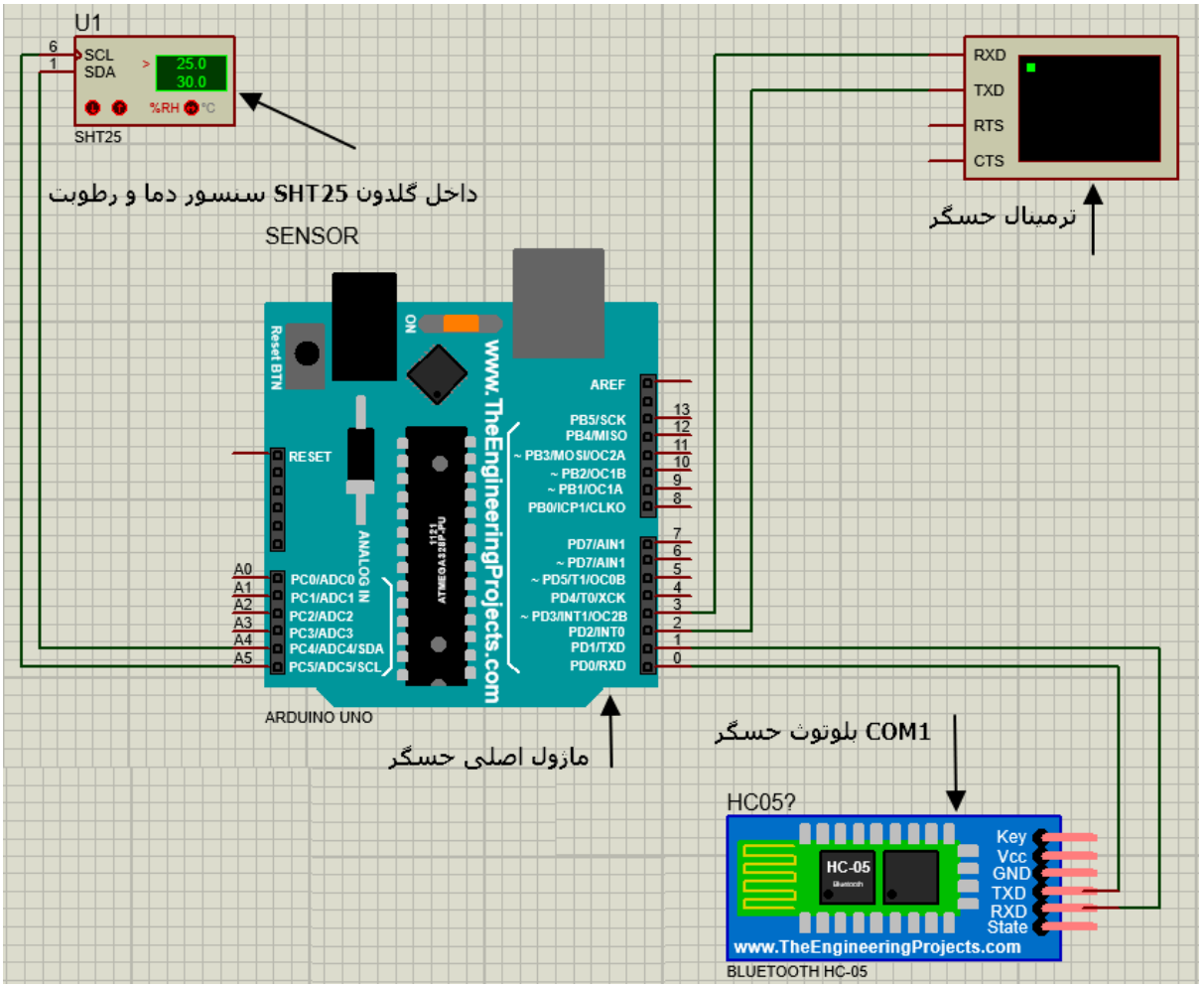
در Wire.write(${destination}); با توجه به اینکه یکبار برای رطوبت و یکبار برای دما استفاده می‌شود برای هر کدام آدرسی که باید اندازه‌گیری‌ها را بفرستد ست می‌کند.

سپس با Wire.endTransmission(); پایان انتقال را اعلام می‌کند. و برای هر کدام هم ۵۰۰ میلی ثانیه دیلی را می‌گذارد.

سپس از آدرس استانداردی که قبلا اشاره شده بود، شروع به خواندن می‌کند، به این شکل که داده‌ها را دو بایتی می‌خواند که بایت اول msB هستن (I2CData[۰]) یعنی با ارزش مکانی بیشتر و بایت دوم (I2CData[۱]) خوانده شده کم ارزش‌تر یا به عبارتی lsB هستن، به همین علت در کانورتورها و لینک گفته شده می‌بینیم که یکی شیفت ۸ تایی به راست داره.

برای هر دو(رطوبت و دما) ابتدا می‌بینیم آیا I2C داده‌ای برای خواندن بهمون میده یا خیر، که در صورت دادن، از وایر می‌خوانیم و سپس با توجه به کارایی‌ای که می‌خواهیم داده‌ی I2C را به داده‌ی قابل تحلیل سلسیوس/رطوبت در می‌آوریم.

این virtualMonitor.print ها هم برای این هست که خروجی داده را روی سریال ویرچوال ماشین بگذاریم که روی ترمینال پروتئوس ماژول مربوطه نشان داده شود.


U1
SCL
SDA
25.0
30.0
%RH °C
SHT25
داخل گلدون SHT25 سنسور دما و رطوبت
SENSOR
RXD
TXD
RTS
CTS
ترمینال حسگر
Reset BTN
ON
www.TheEngineeringProjects.com
AREF
RESET
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD7/AIN1
~ PD5/T1/OC0B
PD4/T0/XCK
~ PD3/INT1/OC2B
PD2/INT0
PD1/TXD
PD0/RXD
ANALOG IN
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
ARDUINO UNO
مازول اصلی حسگر
بلوتوث حسگر COM1
HC05?
HC-05
Key
Vcc
GND
TXD
RXD
State
www.TheEngineeringProjects.com
BLUETOOTH HC-05

Actuator

```
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>
#include <Wire.h>


String entry = "";
SoftwareSerial virtualMonitor(2, 3); // RX | TX
#define M1 7
#define M2 6

#define HUMIDITY 0
#define TEMPERATURE 1
#define DUTY_CYCLE 2
// input format == humidity-temperature$

void setup() {
  Serial.begin(9600);
  virtualMonitor.begin(9600);
  pinMode(M1,OUTPUT) ;
  pinMode(M2,OUTPUT) ;
}
```

پورت‌های M1, M2 .کد بالا تماما مشابه بخش‌ها قبل است، بنابراین تفاوت‌ها را توضیح می‌دهیم

هم همان نرخ آبیاری است که مین تولید می‌کند DUTY_CYCLE، است stepper motor متصله به

بعد از اینکه داده‌های حسگر را تحلیل کرد.

```
char c = '\n';
int dutyCycle = 100;
bool isPrint = false;

void loop() {
  if (Serial.available()){
    c = Serial.read();
    entry += c;
    isPrint = true;
```

```
  }
  if (c == '$'){
    if(isPrint){
      virtualMonitor.print("ACTUATOR: Received from MAIN: ");
      virtualMonitor.println(entry);
      dutyCycle = mainInputProccess(entry);
      isPrint = false;
    }
    entry = "";
    digitalWrite(M1,HIGH);
    digitalWrite(M2,LOW);
    delay(10*dutyCycle);
    digitalWrite(M1,LOW);
    digitalWrite(M2,HIGH);
    delay(1000 - 10*dutyCycle);
  }

}
```

ایف اول که مشابه همان چیزی بود که داشتیم فقط با تفاوت فلگ نوشتن و عدم نوشتن، که برای این است اگر دستور از مین آمد و خواندیم، ما نیاز بود موتورهای عملگر هم کار خود را انجام بدن و دیلی‌شون اعمال بشه بدون اینکه کل ماژول به مشکل بخوره، این راه پیاده شده.

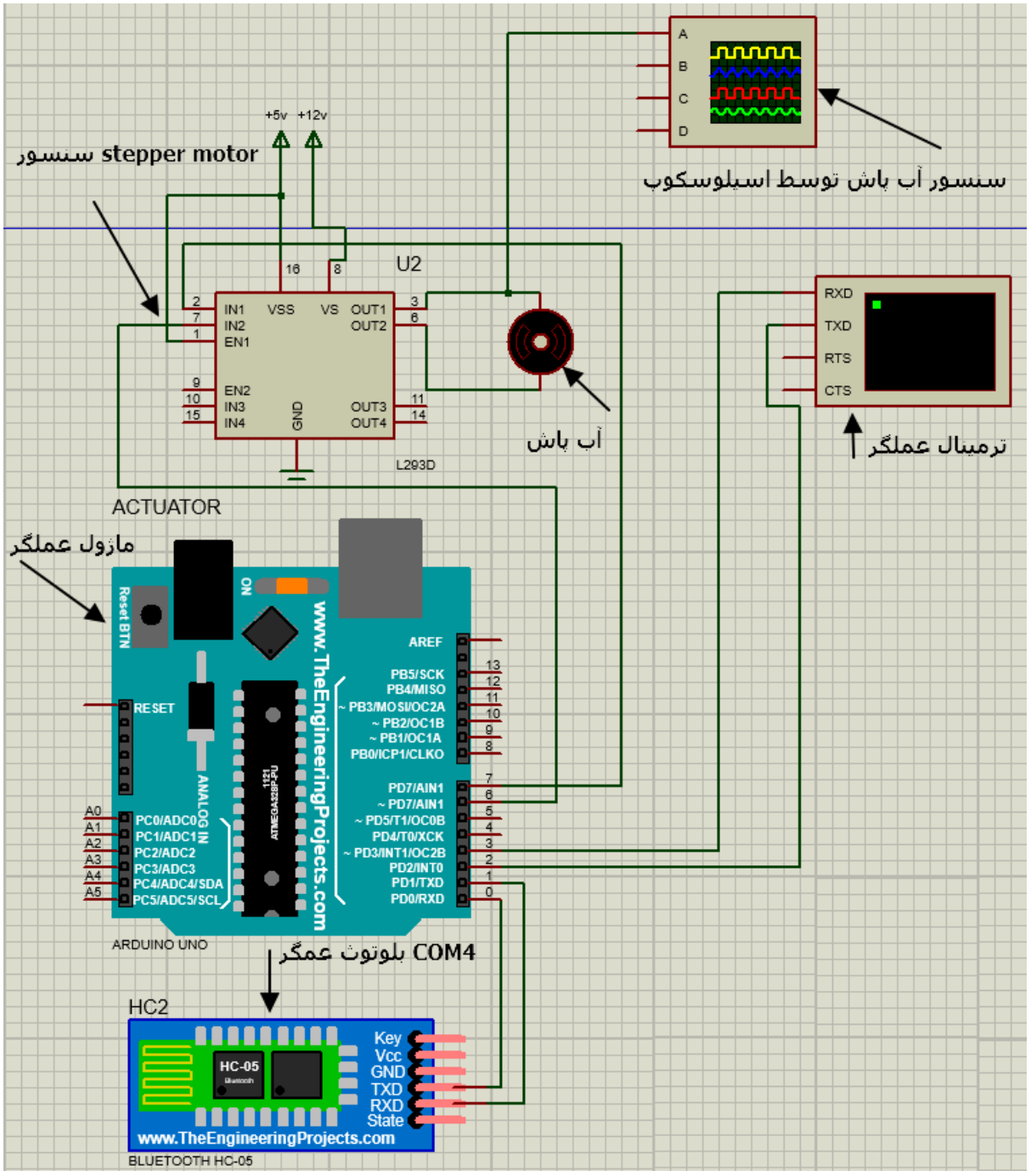
اگر دستور ورودی از طرف main بود و باید روی ترمینال ذکر شود که عملگر فعال شده ، در آن صورت ورودی‌های Main گرفته می‌شود و روی ترمینال چاپ می‌شود با توجه به تابع mainInputProccess، در غیر این هم یعنی از قبل دستورات از main گرفته شده، و الان کنترل بشه که در زمان گفته شده آبیاری بشود و در بقیه زمان ها آبیاری نشود.

```
60      digitalWrite(M1,HIGH);
61      digitalWrite(M2,LOW);
62      delay(10*dutyCycle);
63      digitalWrite(M1,LOW);
64      digitalWrite(M2,HIGH);
65      delay(1000 - 10*dutyCycle);
```

این خطوط قطع و وصل شدن و شدت هم که از main می‌آید را کنترل می‌کنن.

```
int mainInputProccess(const String &data)
{
  int splitIndex[2] = {0, 0};
  int j = 0;
```

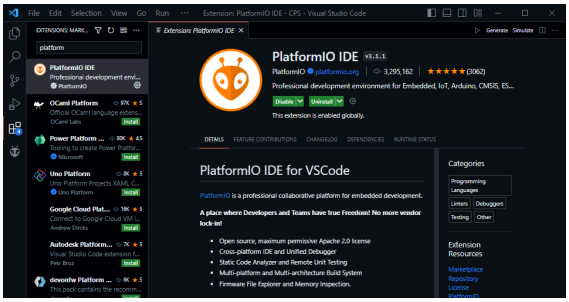
```
  for (unsigned int i = 0; i < data.length(); i++)
    if (data[i] == '-'){
      splitIndex[j] = i;
      j++;
    }
  virtualMonitor.print("Humidity: ");
  virtualMonitor.print(data.substring(0,
splitIndex[0]).toFloat());
  virtualMonitor.print("Temperature");
  virtualMonitor.print(data.substring(splitIndex[0] + 1,
splitIndex[1]).toFloat());
  int dutyCycle = data.substring(splitIndex[1] + 1,
data.length()).toInt();
  virtualMonitor.print(dutyCycle);
  return dutyCycle;
}
```

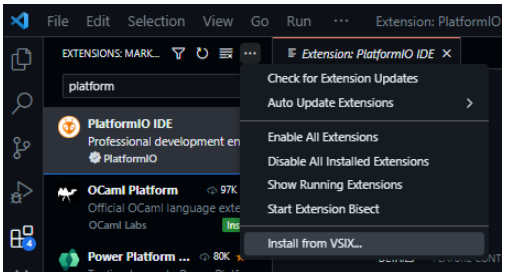

سنسور stepper motor
+5v +12v
U2
IN1
IN2
EN1
EN2
IN3
IN4
VSS
VS
GND
OUT1
OUT2
OUT3
OUT4
L293D
ACTUATOR
سنسور آب باش توسط اسیلوسکوپ
A
B
C
D
آب باش
RXD
TXD
RTS
CTS
ترمینال عملگر
ماژول عملگر
Reset BTN
ON
RESET
ANALOG IN
www.TheEngineeringProjects.com
AREF
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD7/AIN1
~ PD5/T1/OC0B
PD4/T0/XCK
~ PD3/INT1/OC2B
PD2/INT0
PD1/TXD
PD0/RXD
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
ARDUINO UNO
بلوتوث عمگر COM4
HC2
HC-05
Key
Vcc
GND
TXD
RXD
State
www.TheEngineeringProjects.com
BLUETOOTH HC-05

اجرای برنامه

ابتدا باید سه کار نصبی را انجام دهیم:

۱. نصب اکستنشن platform IO در vscode برای اینکه بستر کار کدها در این خواهد بود، یا از طریق خود سرچ وی‌اس‌کد یا از طریق دانلود vsix.

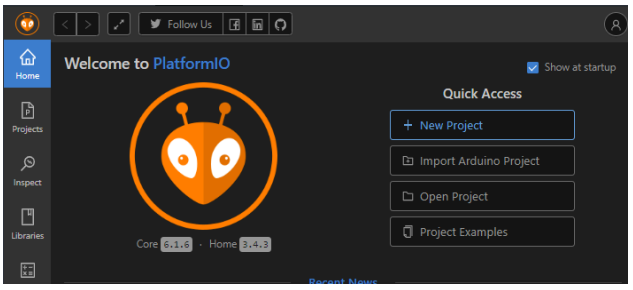




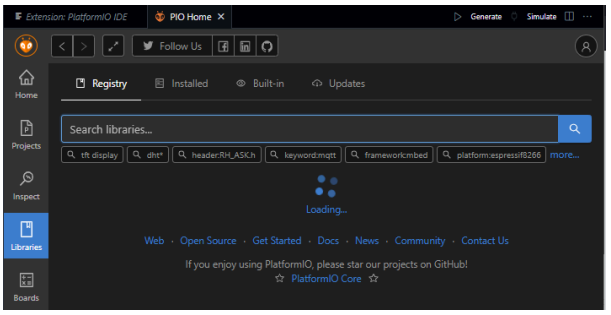
۲. نصب برنامه‌ی proteus 8.15 که بخش طراحی درش انجام می‌شود.

۳. نصب برنامه‌ی virtual serial port که برای کار بلوتوث‌ها به آن نیاز داریم.

برای هر بخش کد باید پروژه جدید زد که لوکیشن این پروژه جدید باید در بیس پروژه‌ی اصلی باشد، که ساخت هر پروژه جدید هم اینگونه‌ست:



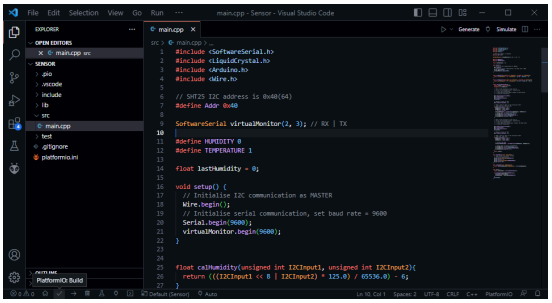
برای هر موردی که کدش را می‌زنیم کتابخانه‌های مربوط از platform IO اضافه می‌شوند:

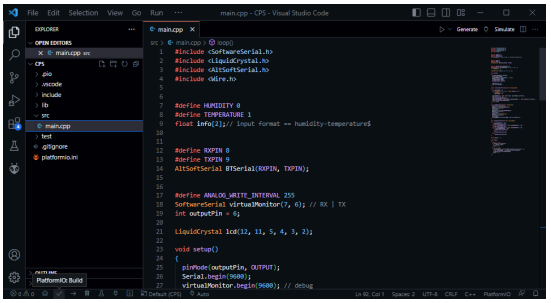


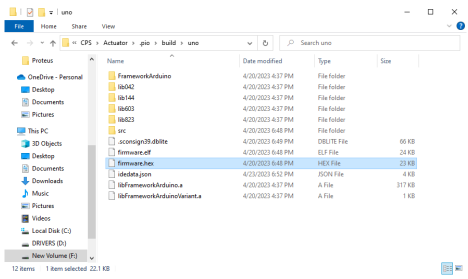
هر کتابخانه‌ای که انتخاب می‌کنیم، در فایل platformio.ini اضافه می‌شود که در لوکیشن اصلی پروژه قرار دارد.

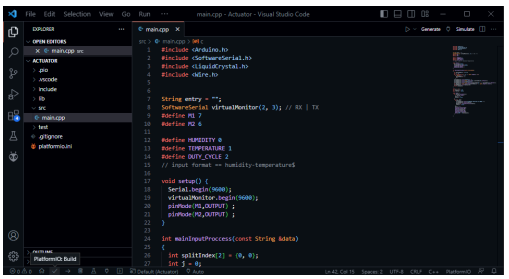
بنابراین در نهایت داریم:

پس از اینکه کدهای هر بخش را زدیم، به ترتیب همه را ران می‌کنیم، تا فایل هگز تولید شود و برای پروتئوس استفاده کنیم:









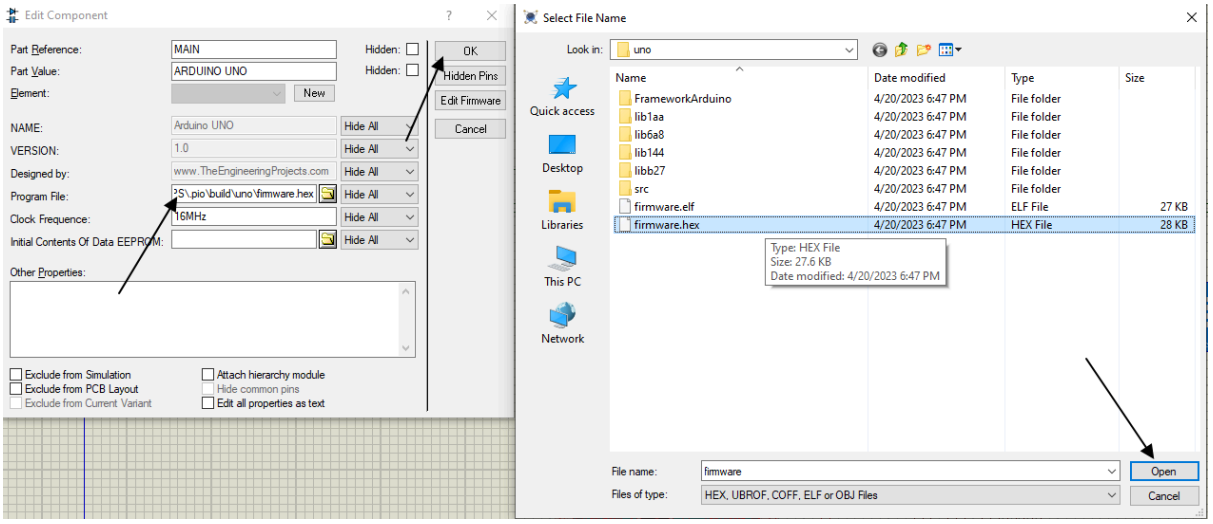
خط لوکیشن‌هایی که فایل هگز تولید می‌شود:

\CPS\${part}\.pio\build\uno\firmware.hex

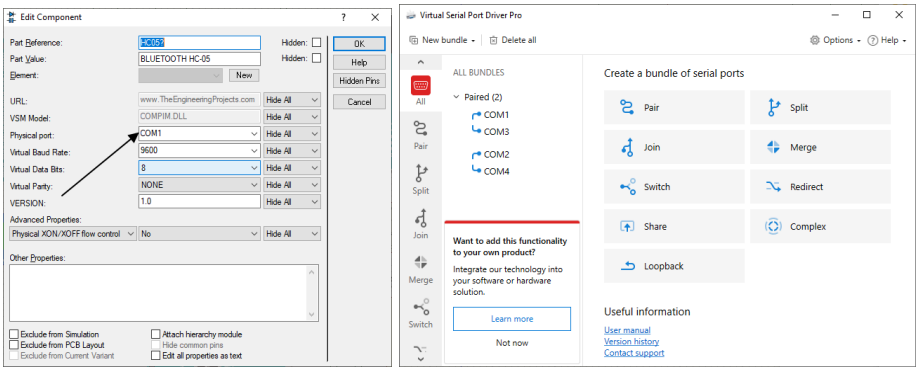
ابتدا پروژه‌ی پروتئوس را باز می‌کنیم، که یا از فایل‌هایی که قبلا تیم روی گیت گذاشته استفاده می‌کنیم یا یکی جدید می‌سازیم و کامپونت‌ها را از سرچ بار درمی‌آوریم یا از اینترنت پیدا می‌کنیم و فایلش را کپی می‌کنیم که قابل وارد شدن باشه.

بعد از تکمیل طراحی، حال باید هر کدام از هگزهای تولیدی را در پروتئوس استفاده کنیم:

به ترتیب روی هر سه کامپونت cps/sensor/actuator دابل کلیک می‌کنیم و هگز تولید شده را آپلود می‌کنیم:



حال بلوتوث‌ها رو هم فعال می‌کنیم:



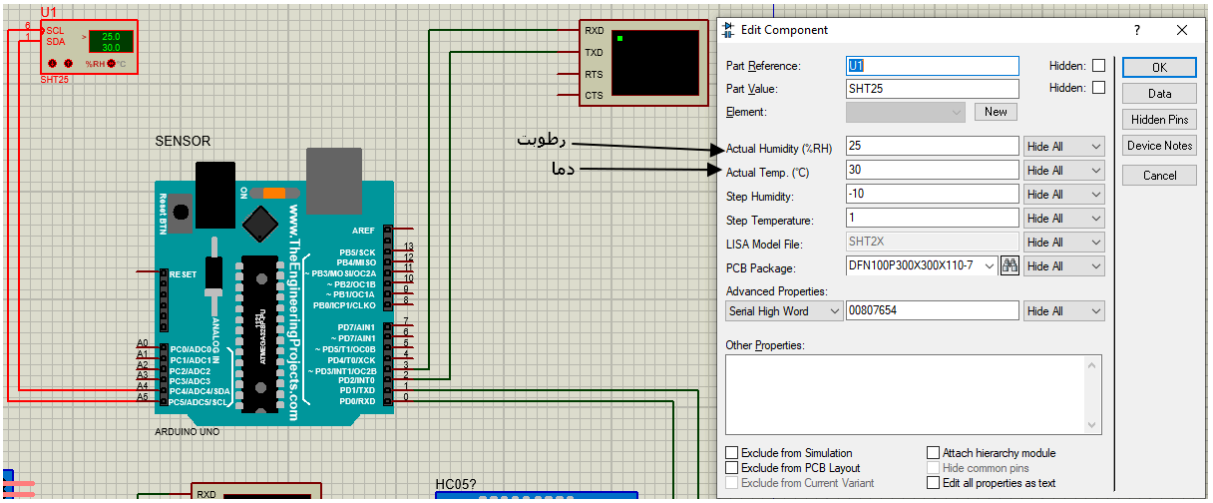
هر جفتی که می‌زنیم نام کامپونت‌ها را از پروتئوس می‌گیریم و بدون سیم بهم متصل می‌کنیم.

همان‌طور که توضیح داده شده، COM1,3 برای اتصال سنسور و مین هست و COM2,4 برای اتصال مین و اکچویتور است.

حال ران شدن کلی را خواهیم داشت:

شرایط گفته شده در مسئله را با مقدار داخل موتور حسگر می‌توانیم بدست آوریم:

۹



۱. اگر رطوبت بالای 30 درصد بود، آبیاری صورت نگیرد.

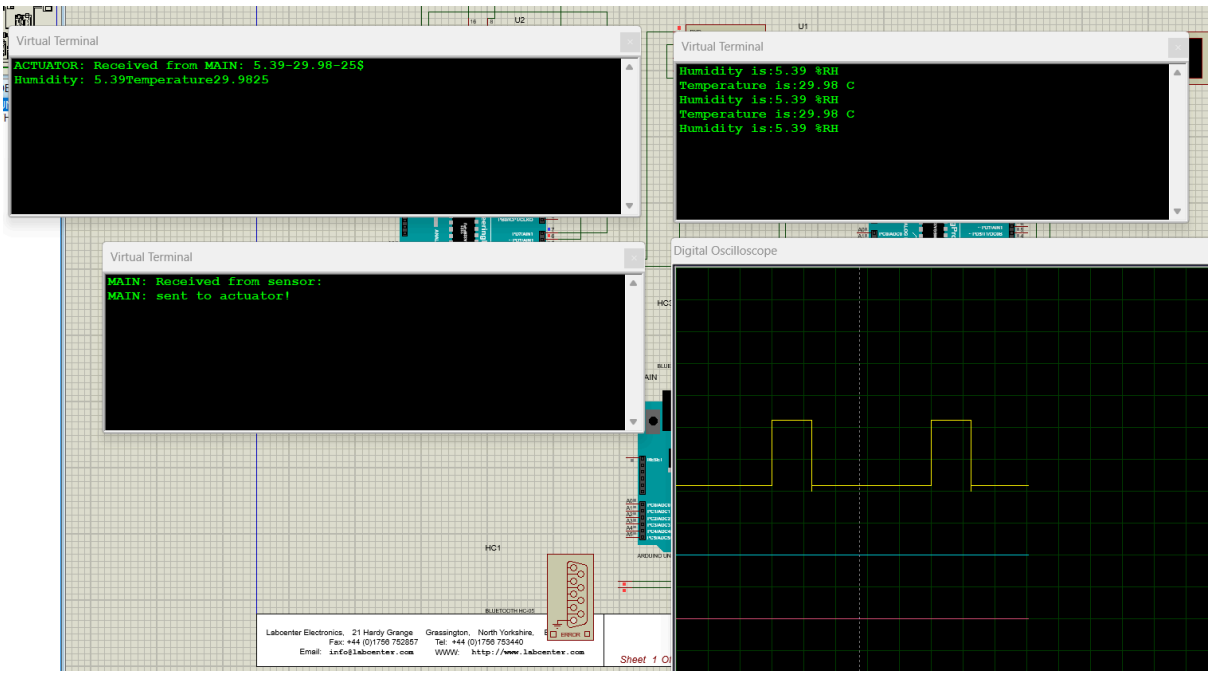
۲. اگر رطوبت بین 20 تا 30 درصد بود نیز دو حالت رخ می‌دهد. اگر دما کم‌تر از ۲۵ درجه‌ی سلسیوس بود، آبیاری لازم نیست. اگر دما بیش‌تر از ۲۵ درجه‌ی سلسیوس بود، آبیاری با نرخ ۱۰ سی سی بر دقیقه صورت گیرد (duty cycle پالس ارسالی به موتور: 10٪).

۳. اگر رطوبت بین 10 تا 20 درصد بود ، آبیاری با نرخ 15 سی‌سی بر دقیقه انجام گیرد. برای اینکار باید پالس هایی با duty cycle ٪20 به موتور ارسال گردد.

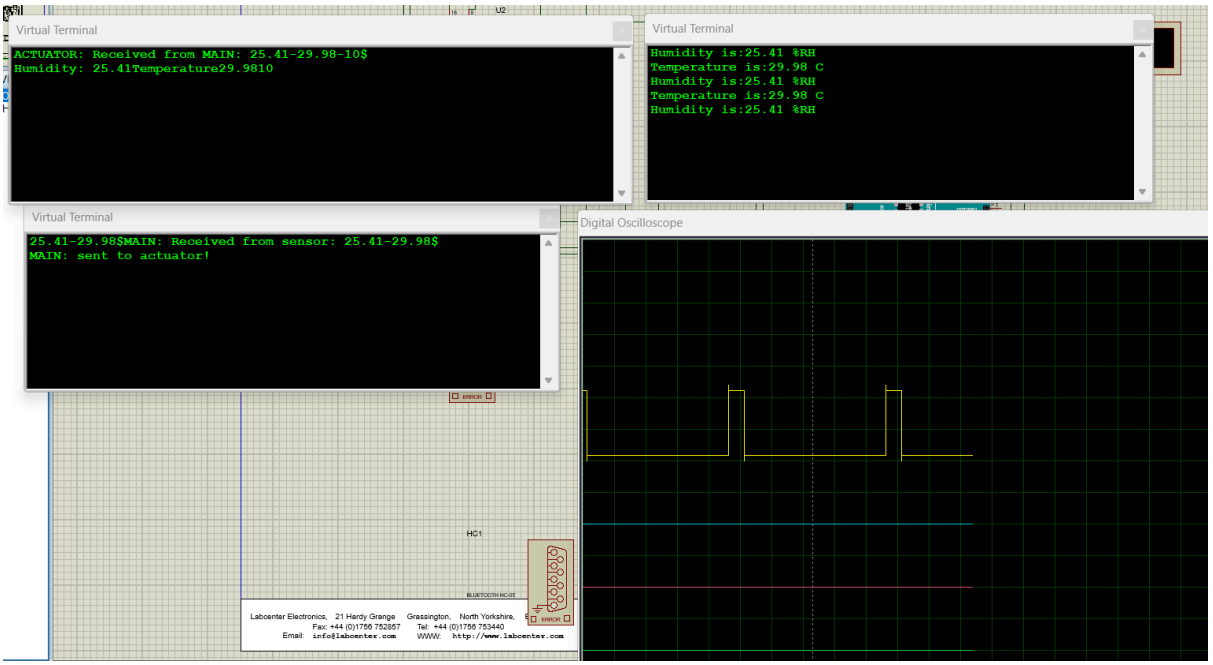
۴. اگر رطوبت کم‌تر از 10 درصد بود، آبیاری با نرخ20 سی‌سی بر دقیقه انجام گیرد. برای اینکار باید پالس هایی با duty cycle ٪25 به موتور ارسال گردد.

خروجی

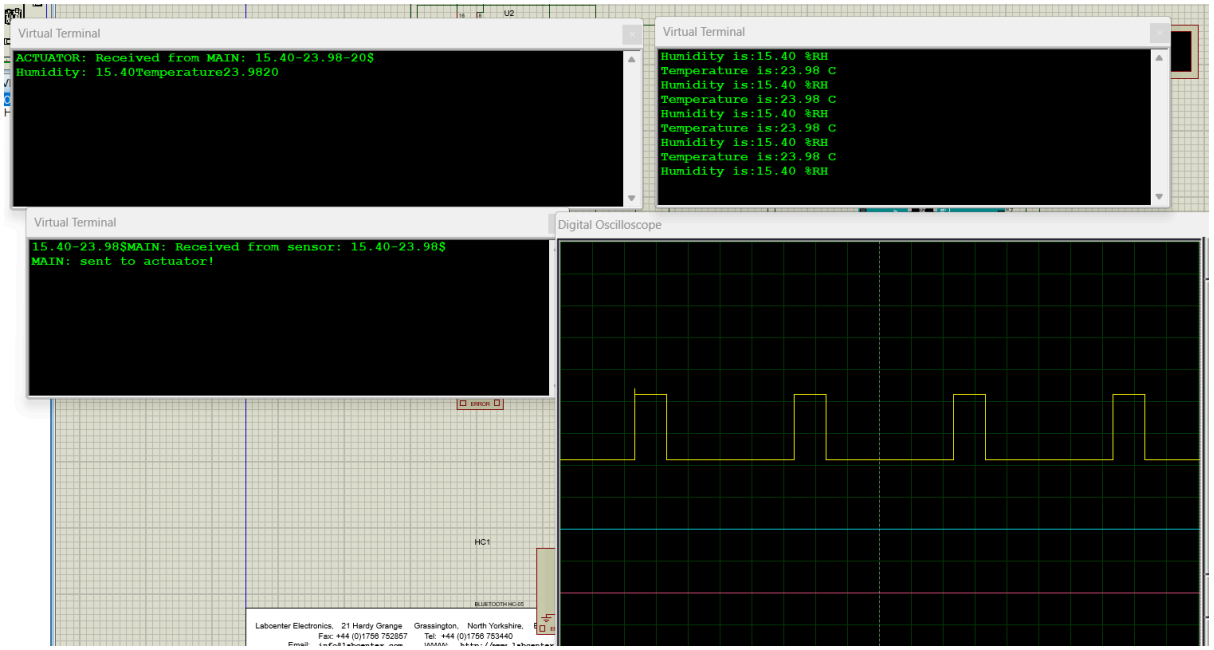
حالت ۱:



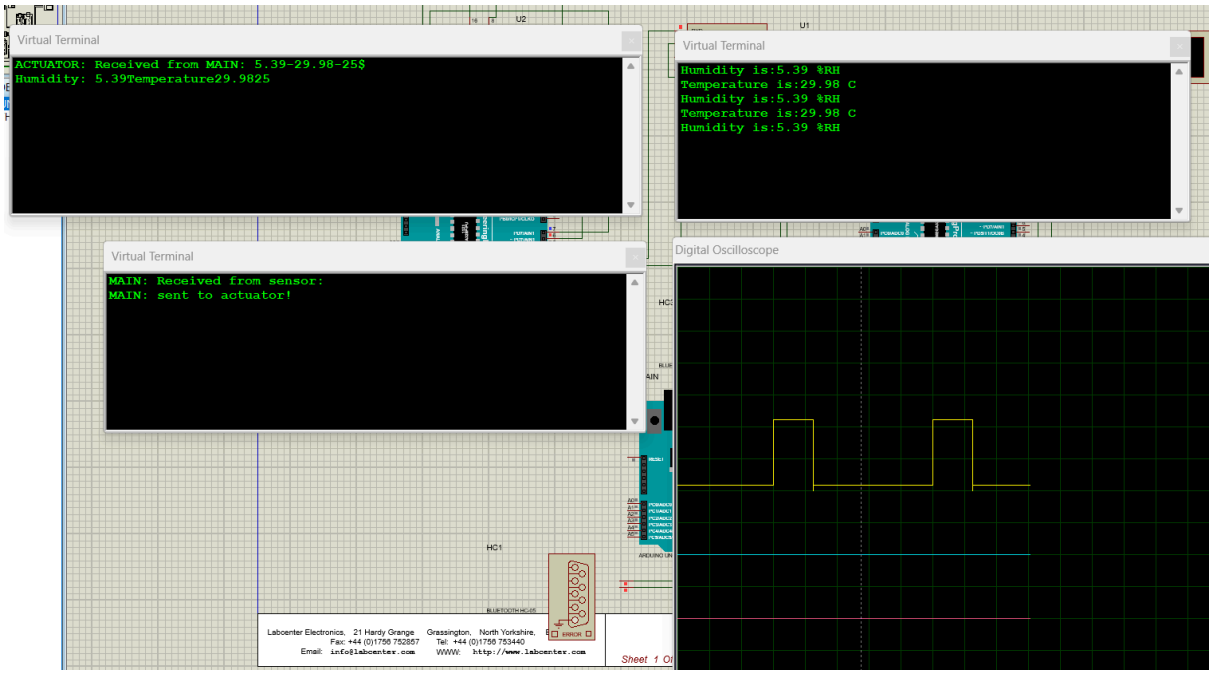
حالت ۲:



حالت ۳:


Virtual Terminal
ACTUATOR: Received from MAIN: 15.40-23.98-20$
Humidity: 15.40Temperature23.9820
Virtual Terminal
Humidity is:15.40 %RH
Temperature is:23.98 C
Humidity is:15.40 %RH
Temperature is:23.98 C
Humidity is:15.40 %RH
Temperature is:23.98 C
Humidity is:15.40 %RH
Temperature is:23.98 C
Humidity is:15.40 %RH
Virtual Terminal
15.40-23.98$MAIN: Received from sensor: 15.40-23.98$
MAIN: sent to actuator!
Digital Oscilloscope

حالت ۴:


Virtual Terminal
ACTUATOR: Received from MAIN: 5.39-29.98-25$
Humidity: 5.39Temperature29.9825
Virtual Terminal
Humidity is:5.39 %RH
Temperature is:29.98 C
Humidity is:5.39 %RH
Temperature is:29.98 C
Humidity is:5.39 %RH
Virtual Terminal
MAIN: Received from sensor:
MAIN: sent to actuator!
Digital Oscilloscope

پاسخ سوالات

پاسخ سوال ۱:

تکنولوژی بلوتوث از امواج رادیویی استفاده میکند. امواج رادیویی بیم 30Hz تا 300GHz هستند. هر چند فرکانس پایین تر باشد، نرخ داده انتقالی کمتر خواهد بود. پس در واقع انتخاب یک طیف رادیویی باعث ایجاد trade-off بین برد و نرخ داده خواهد شد. بلوتوث از فرکانس 2.4GHz امواج رادیویی ISM استفاده می کند(2400MHz تا 2483MHz). باندهای رادیویی ISM بخش هایی از طیف رادیویی هستند که در سطح بین المللی برلی اهداف صنعتی، علمی و پزشکی(ISM) رزرو شده اند و نیاز به مجوز دولتی ندارند.

فناوری بلوتوث روش های مختلفی را برای کاهش تداخل داده ها به کار می گیرد. از جمله استفاده از بسته های داده کوچک و سریع و روش frequency hopping.

داشتن بسته های کوتاهی که به سرعت انتقال داده می شوند، به این معنی است که احتمال برخورد آنها با بسته های ارسال شده توسط سایر دستگاه های نزدیک کمتر است.

روش frequency hopping به این صورت است که طیف را به چندین کانال مجزا تقسیم می کند و سپس هنگام ارسال بسته ها بین آن کانال ها پرش می کند. در هر ثانیه 1600 پرش انجام می شود و باعث می شود احتمال اینکه دو دستگاه با هم تداخل پیدا کنند بسیار کم شود و اگر تداخل پیدا کنند نیز برای مدتی طولانی نخواهد بود.

پاسخ سوال ۲:

باس I2C از تکنیک داوری برای جلوگیری از برخورد بین دستگاه ها زمان انتقال داده ها از طریق باس استفاده می کند. در داوری، به هر دستگاه متصل به باس، یک آدرس منحصر به فرد اختصاص داده می شود که آم وا در باس شناسایی می کند. زمانی که یک دستگاه می خواهد داده ها را به وسیله باس منتقل کند، ابتدا یک شرط شروع و ادرس دستگاه مورد نظر را ارسال می کند. همه دستگاه های موجود در باس این آدرس را دریافت می

کنند، ولی فقط دستگاهی که آدرس منطبق را دارد با ارسال یک سیگنال تایید(ACK) به فرستنده پاسخ می دهد.

اگر دو یا چند دستگاه به طور همزمان سعی برای انتقال داده داشته باشند، خطر احتمال برخورد در باس وجود دارد. برای پیشگیری از این مسئله، هر دستگاه قبل از انتقال داده، وضعیت باس را بررسی می کند. اگر باس مشغول باشد، دستگاه منتظر می ماند تا آزاد شود. اگر همزمان دو دستگاه تلاش برای انتقال داده ها داشته باشند، دستگاهی که آدرس پایین تر را دارد در داوری برنده می شود و می تواند به انتقال داده ادامه بدهد، اما دستگاه دیگر انتقال را متوقف می کند تا بعدا دوباره امتحان کند.

این نوع از داوری تضمین می کند که تنها یک دستگاه می تواند داده ها را در یک زمان باس 12C منتقل کند و از برخورد بین دستگاه ها جلوگیری می کند.

پاسخ سوال ۲:

:DC Motor

همانطور که از نام آن پیداست، موتور DC نوعی موتور الکتریکی است که انرژی الکتریکی را به صورت جریان مستقیم (DC) به انرژی مکانیکی (چرخش شفت) تبدیل می کند.

موتورهای DC دارای دو نوع ساختار هستند که عبارتند از موتور DC برس دار و موتور DC بدون جاروبک. اما، وقتی از کلمه کلیدی موتور DC استفاده می کنیم، به سادگی به این معنی است که ما در مورد موتور DC برس خورده صحبت می کنیم.

یک موتور DC از یک قسمت استاتور تشکیل شده است که سیستم میدان مغناطیسی را تشکیل می دهد و یک قسمت روتور که به عنوان آرمیچر موتور عمل می کند. یک کموتاتور نیز روی شفت روتور نصب شده است. بخش های کموتاتور به انتهای سیم پیچ آرمیچر متصل می شوند. از برس های کربنی برای برقراری تماس الکتریکی با کموتاتور و آرمیچر استفاده می شود. بنابراین، برای راه اندازی موتور DC، جریان مستقیم از کموتاتور و برس ها به سیم پیچ آرمیچر عبور می کند.

جریان الکتریکی که از سیم‌پیچ آرمیچر عبور می‌کند، میدان مغناطیسی ایجاد می‌کند که با میدان مغناطیسی استاتور تعامل می‌کند و گشتاور روی روتور تولید می‌کند. با توجه به این گشتاور، روتور برای تولید انرژی مکانیکی خروجی در شفت می چرخد.

کاربرد:

موتورهای DC معمولاً در دستگاه های الکترونیکی، جرثقیل ها، اسباب بازی ها، آسانسورها و بالابرها، ابزارهای برقی و در بسیاری از وسایل دیگر استفاده می شوند.

:Servo Motor

نوعی موتور الکتریکی که به عنوان یک محرک چرخشی عمل می کند و امکان کنترل دقیق سرعت، موقعیت و شتاب زاویه ای را فراهم می کند به عنوان موتور سروو شناخته می شود.

سروو موتورها شامل یک سنسور برای بازخورد موقعیت روی روتور خود هستند تا موقعیت و سرعت آن را همیشه کنترل کنند. بنابراین، موتورهای سروو به طور کلی با طراحی سیستم کنترل حلقه بسته خود مشخص می شوند.

کاربرد:

سروو موتورها می توانند در دو نوع سروو موتورهای AC و سروو موتورهای DC باشند. جایی که سروو موتورهای AC در کاربردهای بزرگ مورد استفاده قرار می گیرند

:Stepper Motor

نوعی موتور الکتریکی DC بدون جاروبک که چرخش کامل روتور برای آن به مراحل مساوی تقسیم می شود به عنوان موتور پله ای شناخته می شود. موتورهای پله ای از تعداد زیادی سیم پیچی با فاصله مساوی روی استاتور تشکیل شده اند، این سیم پیچ ها در هنگام تامین برق به عنوان قطب های مغناطیسی عمل می کنند. روتور استپر موتور از جفت آهنربای دائمی به شکل دنده ساخته شده است.

موتور پله ای مجهز به یک کنترل کننده موتور الکترونیکی است که جریان را به هر سیم پیچ متوالی استاتور تغییر می دهد تا روتور را به صورت مغناطیسی از یک قطب به قطب دیگر حرکت دهد.همانطور که از بحث بالا مشخص می شود که اپراتور می تواند چرخش روتور را به طور موثر برای حرکات دورانی دقیق و پلکانی کنترل کند، به همین دلیل این موتورها را موتورهای پله ای می نامند.

یکی از مزیت‌های بزرگ موتورهای پله‌ای این است که می‌توانند یک گشتاور نگهدارنده، یعنی یک گشتاور مثبت در سرعت صفر ارائه دهند، که این موتور را برای کاربردهای موقعیت‌یابی مختلف مانند روباتیک مناسب می‌کند. با این حال، موتورهای پله ای کارایی کمتری نسبت به موتورهای DC معمولی دارند.

کاربرد:

موتورهای پله ای به طور گسترده در هارد دیسک ها، چاپگرها و بسیاری از سیستم های کنترلی دیگر و غیره استفاده می شوند

تفاوت ها:

Servo Motor	Stepper Motor	DC Motor	فاکتور ها
سرعت سروو موتورها معمولاً از 1000 RPM تا 6000 RPM زیاد است.	سرعت استپر موتور کم است، حدود 200 تا 2000 دور در دقیقه.	موتورهای DC بسته به نوع موتور محدوده سرعت متوسطی دارند.	سرعت
سروو موتورها دارای مکانیسم کنترل پیچیده ای هستند. با این حال، آنها بسیار قابل کنترل اند	مکانیزم کنترل استپر موتور نیاز به میکروکنترلر دارد.	مکانیزم کنترل موتورهای DC ساده است و نیازی به وسایل اضافی مانند میکروکنترلر نیست.	مکانیزم کنترل
راندمان سروو موتورها بالاست.	راندمان موتورهای استپر پایین است.	موتورهای DC دارای راندمان بالایی در حدود 85٪ هستند.	کارایی