به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

سیستم های نهفته و بی درنگ

گزارش پروژه اول

ملیکا مرافق 810197581

نازنین یوسفیان 810197610

سینا سلیمیان 810197528

حمیدرضا خدادادی 810197499

بهار 1401

Main-Board

در ابتدا جداکننده های داده های ارسالی و دریافتی برای دما و رطوبت که از طریق بلوتوث ارسال می شوند را مشخص می کنیم. حداکثر سرعت موتور DC را هم 256 تعریف می کنیم. برای کار با LCD کتابخانه LiquidCrystal.h را include می کنیم. سپس LCD را با شماره پین هایی که به وسیله آن به arduino متصل است، مشخص می کنیم.

mainBoard §

```
#include <LiquidCrystal.h>

#define MAX_SPEED 256

float humidity, temperature;
int wateringRate;
int PWM;

bool readSerial();
void showOnLCD();

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

در تابع setup، در ابتدا یک LCD با ابعاد 20 * 4 (تعداد سطر و ستون های LCD) می سازیم. همچنین ارتباط سریال را برای ارتباط با بلوتوث شروع می کنیم. و همچنین سرعت اولیه موتور را هم برابر صفر قرار می دهیم.

تابع loop، تابعی است که همواره در حال اجرا است. در این تابع در ابتدای کارش، تابع readSerial را صدا می زنیم. این تابع را در ادامه توضیح خواهیم داد. این تابع readSerial یک bool به خروجی می دهد که اگر برابر true باشد، یعنی که داده ای که را از بلوتوث دریافت کرده ایم، بزرگ تر از 4 بایت است و می توانیم ادامه عملیات را انجام دهیم. که نشان دهنده این است که ما 2 بایت برای رطوبت (یک جداکننده 'H' و یک عدد float به عنوان مقدار) خوانده ایم و 2 بایت هم برای گرما (یک جداکننده 'T' و یک عدد float به عنوان مقدار) خوانده ایم. در نتیجه اگر وارد شرط بشویم، تابع setWateringRate را صدا می زنیم که دستورات

جدید موتور تنظیم شود. تابع showOnLCD هم برای نمایش دما و رطوبت بدست آمده از سنسور صدا زده می شود.

```
void setup() {
    PWM = 0;
    lcd.begin(20, 4);
    Serial.begin(9600);
}

void loop() {
     bool rData = readSerial();
     showOnLCD();
     if (rData) {
       setWateringRate();
     }
}
```

همانطور که بالاتر گفته شد، در تابع readSerial تعداد بایت های موجود برای خواندن از پورت سریال را بررسی می کنیم. اگر این مقدار بیش از ۴ بایت بود (۲ بایت رطوبت و ۲ بایت دما به علاوه delimiter ها) بایت اول را به صورت کاراکتر می خوانیم. اگر برابر 'H' بود یعنی داده بعدی که دریافت می کنیم رطوبت است و می دانیم که این مقدار float بوده پس آن را با parseFloat می خوانیم و آن را ذخیره می کنیم. به طریقی مشابه دما را نیز ذخیره می کنیم.

```
bool readSerial(){
  char incomingByte;
  bool rData = false;

  if(Serial.available() >= 4){// 2 byte H , 2byte ? check
    rData = true;
    incomingByte = Serial.read();
    if (incomingByte == 'H')
      humidity = Serial.parseFloat();
    incomingByte = Serial.read();
    if (incomingByte == 'T')
      temperature = Serial.parseFloat();
  }
  return rData;
}
```

تابع setWateringRate برای مشخص کردن دستورات مناسب و ارسال آن ها به گره غیر مرکزی استفاده می شود. در این تابع بدین صورت عمل می شود که اگر رطوبت بالای 50 درصد بود، آبیاری صورت نمی گیرد. برای این کار، مقدار wateringRate و PWM که سرعت موتور است را برابر صفر قرار می دهیم. و اگر رطوبت کمتر از 20 درصد بود، آبیاری با نرخ 20 سی سی بر دقیقه صورت می گیرد. برای این کار، مقدار wateringRate و PWM که سرعت موتور است را به ترتیب برابر 20 و 256/4 (پالس با duty cycle 25%) قرار می دهیم. و اگر رطوبت بیشتر از 20 درصد و کمتر از 50 درصد بود دو حالت داریم:

- اگر دما کمتر از 25 درجه سلیسیوس بود، آبیاری صورت نمی گیرد. برای این کار، مقدار wateringRate و PWM که سرعت موتور است را برابر صفر قرار می دهیم.
- و اگر دما بیشتر از 25 درجه سلیسیوس بود، آبیاری با نرخ 10 سی سی بر دقیقه صورت می گیرد. برای این کار، مقدار wateringRate و PWM که سرعت موتور است را به ترتیب برابر 10 و 256/10 (پالس با duty cycle 10%) قرار می دهیم.

mainBoard §

```
void setWateringRate() {
    if(humidity > 50){
        wateringRate = 0;
        PWM = 0;
    }
    else if(humidity < 20) {
        wateringRate = 20; //20cc
        PWM = MAX_SPEED / 4;
    }
    else if (20 <= humidity && humidity <= 50) {
        if (temperature < 25){
           wateringRate = 0;
           PWM = 0;
        }
        else {
           wateringRate = 10; //10cc
           PWM = MAX_SPEED / 10;
        }

    }
    Serial.println(PWM);
}
```

در تابع showOnLCD دما و رطوبت دریافتی از TH-Board و از طریق بلوتوث را نمایش می دهیم. در اولین سطر از LCD (گوشه، سمت چپ، بالا) مقدار صحیح دما را نمایش می دهیم. سپس با جا به جا کردن cursor، در خط دوم و در ابتدای سطر رطوبت را نمایش می دهیم. در نهایت نرخ آبیاری را در سطر سوم نمایش می دهیم.

```
void showOnLCD(){
  lcd.setCursor(0, 0);
  lcd.print("Temperature: ");
  lcd.print(int(temperature));

  lcd.setCursor(0, 1);
  lcd.print("Humidity: ");
  lcd.print(humidity);

  lcd.setCursor(0,2);
  lcd.print("Water: ");
  lcd.print(wateringRate);
  lcd.print("cc");
}
```

TH-Board

ابتدا کتابخانه Wire.h را که برای ارتباط I2C است را include می کنیم. آدرس I2C سنسور SHT25 که برابر 0x40 است را تعریف می کنیم. پین های یک و دو TH board که برابر 10 و 9 است را هم define می کنیم.

تابع setMotorPinMode بدین صورت عمل می کند که دو تا پین TH board که باید به موتور متصل شود را برابر OUTPUT ست می کند.

thBoard §

```
#include <Wire.h>

#define SensorAddr 0x40
#define motorPin1 10 //pin2 L293D
#define motorPin2 9 //pin7 L293D

unsigned long myTime;

void setMotorPinMode(){
    pinMode(motorPin1, OUTPUT);
    pinMode(motorPin2, OUTPUT);
}
```

در تابع setup، با Wire.begin() به عنوان کنترلر به bus جوین می شویم. همچنین ارتباط سریال (UART) را با مقدار baud rate پیش فرض ۹۶۰۰ شروع می کنیم. همچنین تابع setMotorPinMode را هم که بالاتر توضیح دادیم، صدا می زنیم.

تابع loop، تابعی است که همواره در حال اجرا است. در تابع loop لازم است که همواره رطوبت و دما را از سنسور دریافت کنیم. این کار را با استفاده از توابع تعریف شده getHumidityFromSensor و getTemperatureFromSensor انجام می دهیم. سپس لازم است که اطلاعات به دست آمده (دما و رطوبت) از سنسور را از طریق بلوتوث به برد مرکزی بفرستیم.

TH-Board از طریق ارتباط سریال UART به بلوتوث متصل است. اطلاعات دما و رطوبت را بر روی پورت سریال می نویسیم تا به ماژول بلوتوث ارسال شود.

سپس تا زمانی که داده جدیدی روی سیم موجود نباشد، روی ارتباط سریال بلاک می شویم و صبر می کنیم. و زمانی که دیتای جدید را دریافت کردیم که همان سرعت جدید موتور خواهد بود، آن را به یک پین موتور منتقل می کنیم و پین دیگر موتور را هم برابر صفر قرار می دهیم. سپس 4 ثانیه دیلی در این فرایند خواهیم داشت. (هر کدام از دو تابع دریافت دما و رطوبت، 0.5 ثانیه دیلی دارند و در مجموع 5 ثانیه دیلی خواهیم داشت.)

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

}

void loop() {
  float humidity = getHumidityFromSensor();
  float celsiusTemperature = getTemperatureFromSensor();
  Serial.println("H"+String(humidity)+"T"+String(celsiusTemperature));

  while (Serial.available() < 1) {}

  int PWM = Serial.parseInt();
  analogWrite(motorPin1, PWM);
  analogWrite(motorPin2, 0);

  delay(4000); // 5 - 1s delay to getT,getH
}
```

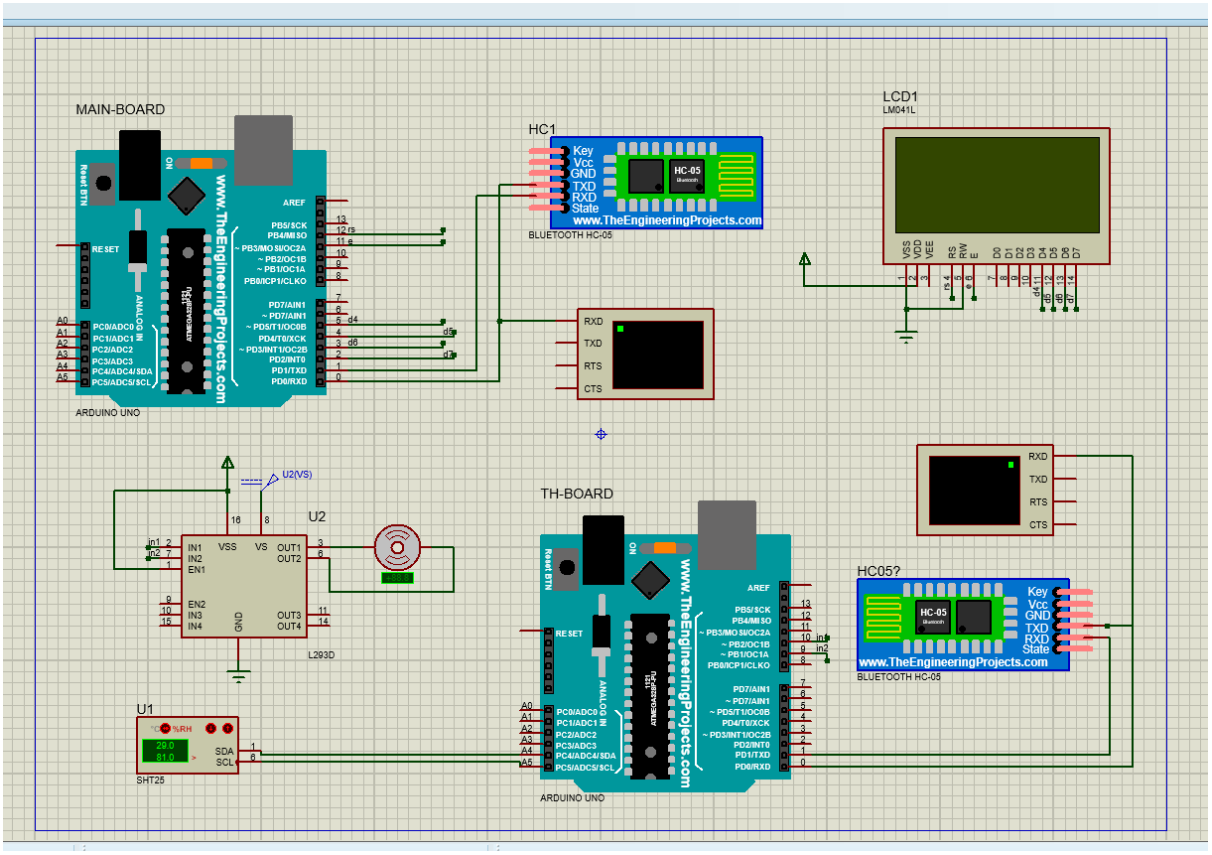
در تابع getHumidityFromSensor در ابتدا ATmega به عنوان I2C master مخابره را آغاز می کند و آدرس سنسور (slave) را میفرستد. سپس دستور اندازه گیری رطوبت (0xF5) را به سنسور می دهد و در نهایت مخابره را پایان می دهد. ATmega درخواست ۲ بایت از slave می کند و سپس بررسی می کند که آیا در بافرش دو بایت داده را دارد یا خیر. (یعنی دو بایتی را که درخواست کرده بود دریافت کرده یا نه.) در صورت وجود داده آن را می خواند و در نهایت رطوبت را بر می گرداند.

```
float getHumidityFromSensor(){

  unsigned int data[2];
  Wire.beginTransmission(SensorAddr);// Start I2C transmission
  Wire.write(0xF5); // Send humidity measurement command, NO HOLD master
  Wire.endTransmission(); // Stop I2C transmission
  delay(500);

  Wire.requestFrom(SensorAddr, 2);

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

    return  (((data[0] * 256.0 + data[1]) * 125.0) / 65536.0) - 6;
  }
}
```

تابع getTemperatureFromSensor مشابه تابع دریافت رطوبت است با این تفاوت که نوع دستور داده شده به سنسور، اندازه گیری دما است (0xF3). و در نهایت دما را برحسب درجه سلسیوس باز می گردانیم.

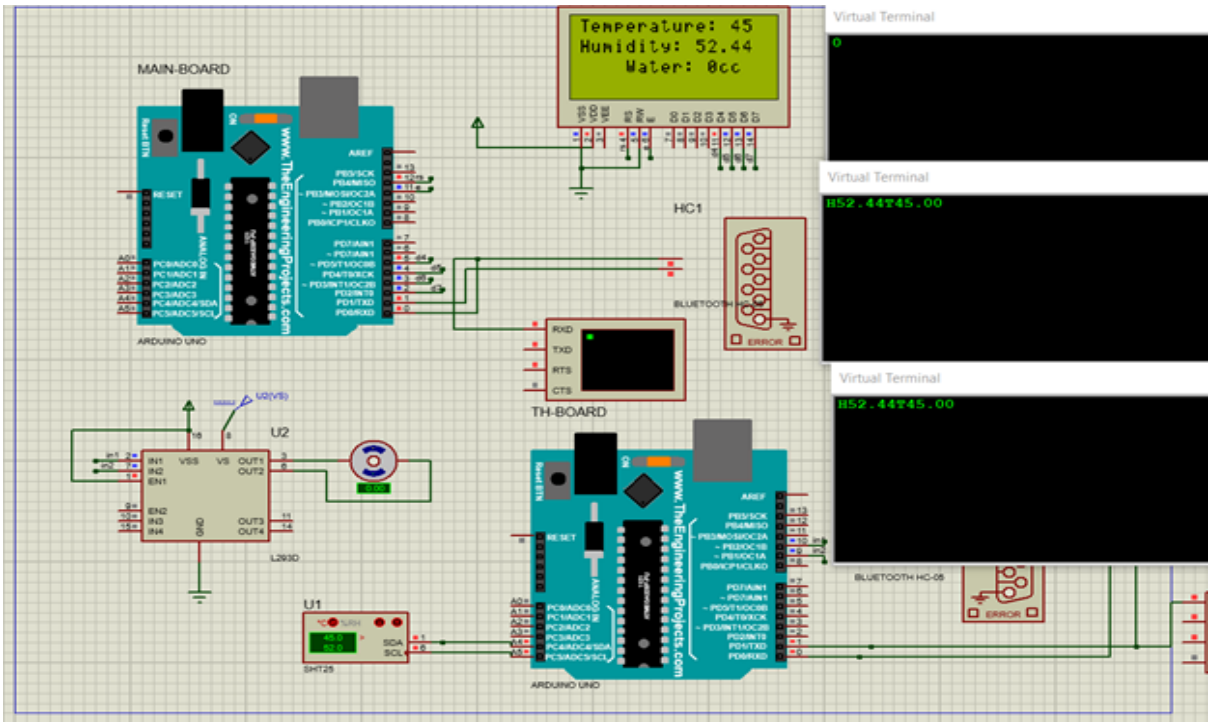
```
float getTemperatureFromSensor(){
    unsigned int data[2];
    Wire.beginTransmission(SensorAddr);
    Wire.write(0xF3);  // Send temperature measurement command, NO HOLD master
    Wire.endTransmission();  // Stop I2C transmission
    delay(500);
    Wire.requestFrom(SensorAddr, 2);// Request 2 bytes of data

    if(Wire.available() == 2)  {// Read 2 bytes of data  temp msb, temp lsb
      data[0] = Wire.read();
      data[1] = Wire.read();
      float celsiusTemperature = (((data[0] * 256.0 + data[1]) * 175.72) / 65536.0) - 46.85;
      return celsiusTemperature;
  }
}
```

شبیه سازی در Proteus:

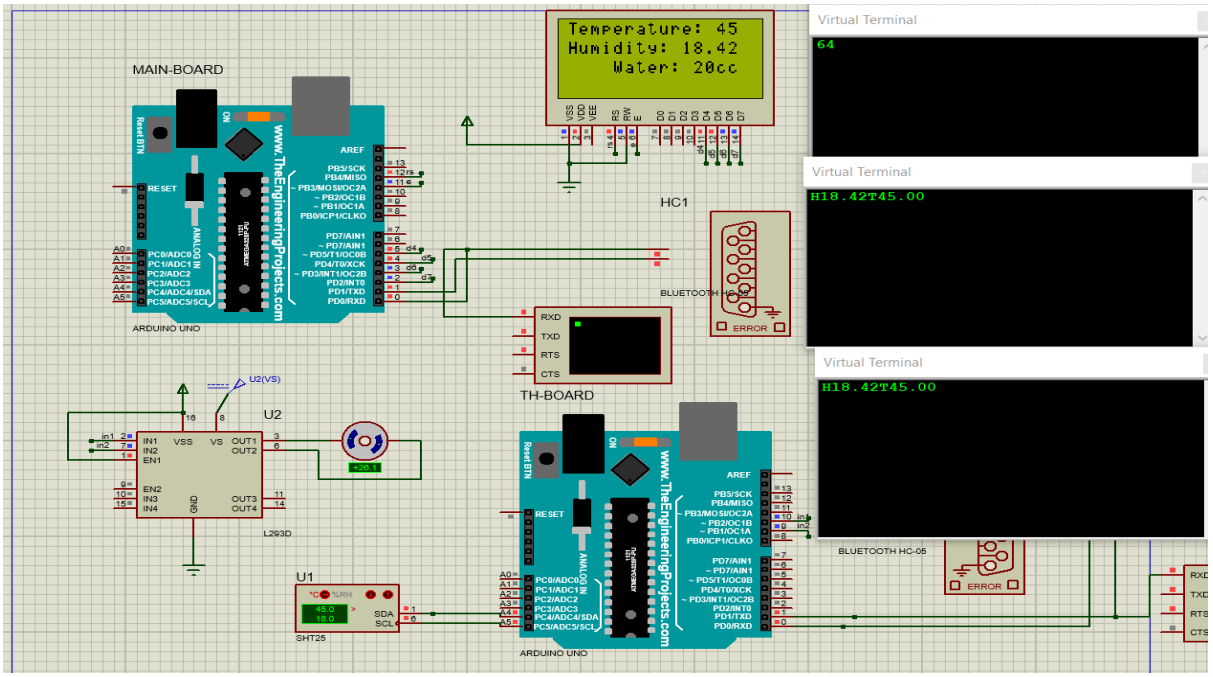


شکل کلی مدار بسته شده به صورت بالا است. سیم های اتصالات، به خاطر تمیزی مدار به صورت لیبل گذاری شده متصل شده اند. دو بورد mainBoard و THBoard از نوع Arduino Uno داریم. کد های هر یک از بورد ها را در ادیتور Arduino کامپایل می کنیم و سپس آدرس فایل هگز آن ها را در بورد ها قرار می دهیم. یک سنسور اندازه گیری دما و رطوبت به بورد TH وصل می کنیم. هر کدام از این دو بورد، به ماژول بلوتوث متصل است و COM3 و COM4 این دو بلوتوث با هم pair شده اند. بورد TH دیتای دما و رطوبت را از سنسور دریافت می کند و از طریق بلوتوث به بورد Main منتقل می کند. این بورد، داده دریافتی را در LCD ای که به آن متصل است، نمایش می دهد. سپس بر اساس منطق پیاده سازی شده، با توجه به مقدار دما و رطوبت، سرعت موتور و wateringRate را تنظیم می کند و سرعت موتور را از طریق بلوتوث به بورد TH بر می گرداند. سپس این بورد TH که از طریق پین های آنالوگش به موتور متصل است، دیتا را به موتور می فرستد و موتور می چرخد.

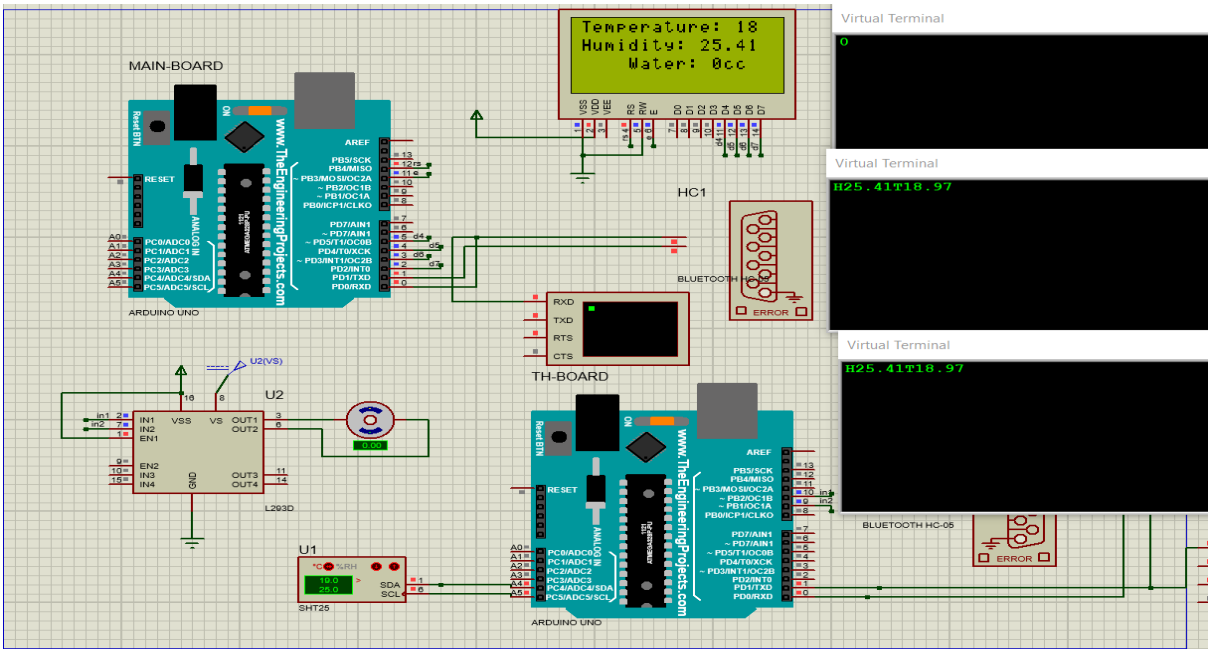
در حالت زیر، رطوبت (نمایش داده شده در LCD) بالای 50 درصد است. در نتیجه نرخ wateringRate برابر صفر می شود و PWM یا سرعت موتور هم صفر می شود و موتور نمی چرخد.



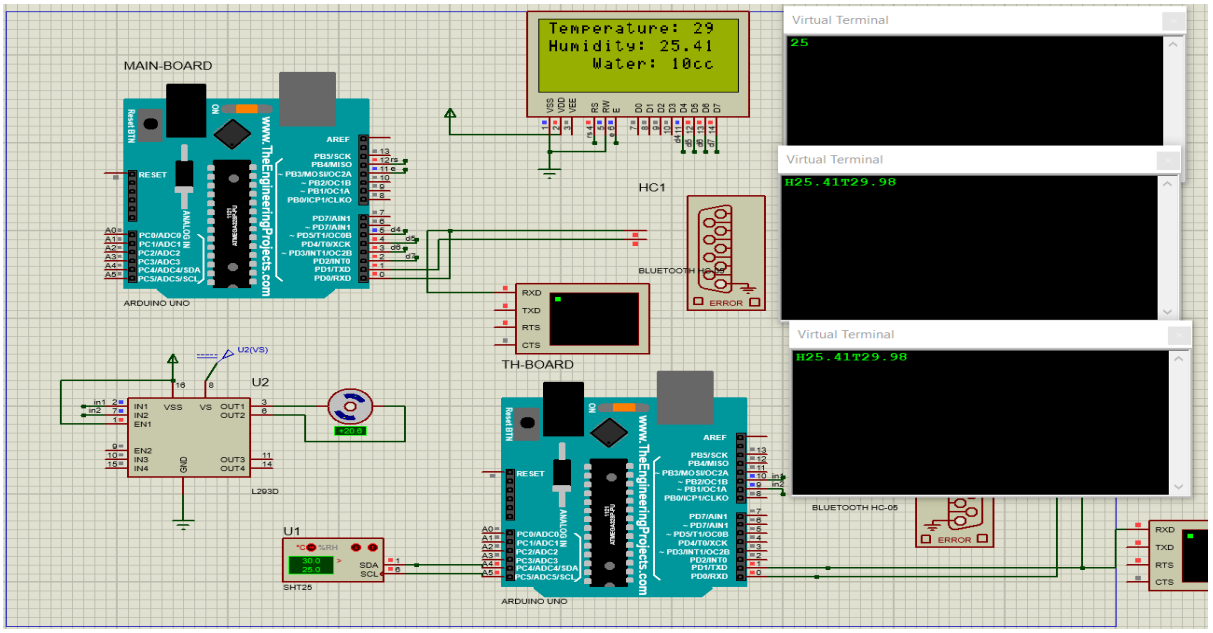
در حالت زیر، رطوبت (نمایش داده شده در LCD) کمتر از 20 درصد است. در نتیجه نرخ wateringRate برابر 20 سی سی بر دقیقه می شود و PWM یا سرعت موتور هم 64 یا 256 * 0.25 می شود و موتور می چرخد.



در حالت زیر، رطوبت (نمایش داده شده در LCD) بین 20 و 50 درصد و دما کمتر از 25 درجه سلیسیوس است. در نتیجه نرخ wateringRate برابر صفر می شود و PWM یا سرعت موتور هم صفر می شود و موتور نمی چرخد.



در حالت زیر، رطوبت (نمایش داده شده در LCD) بین 20 و 50 درصد و دما بیشتر از 25 درجه سلیسیوس است. در نتیجه نرخ wateringRate برابر 10 می شود و PWM یا سرعت موتور هم 25 یا 256 * 0.10 می شود و موتور می چرخد.



پرسش ها:

1. تکنولوژی بلوتوث از امواج رادیویی استفاده می کند. امواج رادیویی بین 30Hz تا 300GHz هستند. هرچه فرکانس پایین تر باشد، نرخ داده انتقالی کمتر خواهد بود. پس درواقع 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 (2400 تا 2483.5 مگاهرتز). باندهای رادیویی ISM بخش هایی از طیف رادیویی هستند که در سطح بین المللی برای اهداف صنعتی، علمی و پزشکی (ISM) رزرو شده اند و نیاز به مجوز دولتی ندارند.

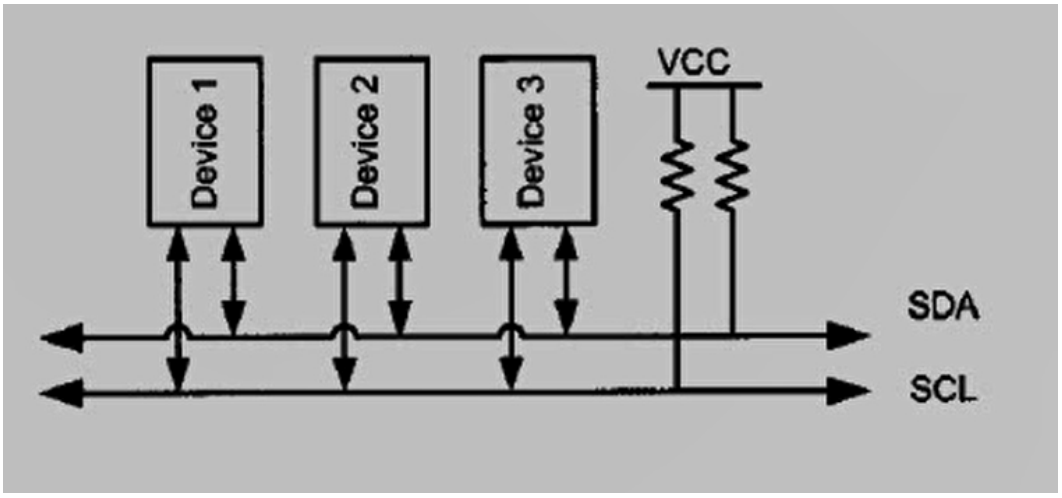
فناوری بلوتوث روش های مختلفی را برای کاهش تداخل داده ها به کار می گیرد. از جمله استفاده از بسته های داده کوچک و سریع و روش frequency hopping.

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 (40 در مورد بلوتوث کم انرژی) 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 مدت طولانی نخواهد بود.

2. بله. می دانیم که I2C یک bus است و می توان تا ۱۱۹ دستگاه را به آن متصل کرد.

از مدل Wired-and استفاده می شود به این شکل که خطوط SDA و SCL را همواره یک نگه می داریم. اگر دستگاهی خواست یک بر روی خط بفرستد کاری را لازم نیست انجام دهد. اما اگر کسی خواست صفر را بر روی خط بفرستد خط را باید به زمین وصل کند.



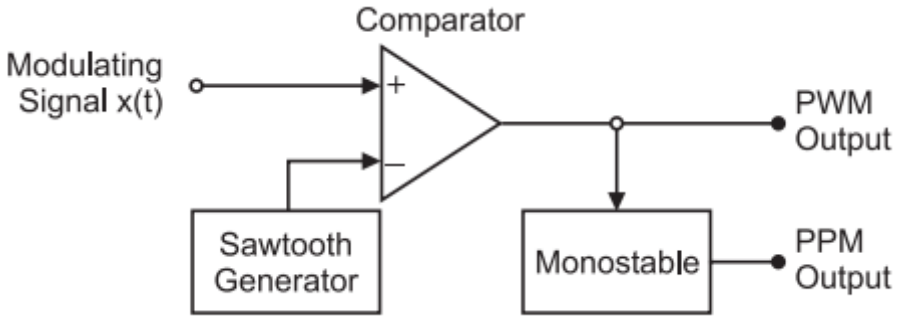
چون منبع تغذیه (vcc) از طریق مقاومت به زمین وصل است یک ضعیف را روی خط می فرستد و با وصل کردن خط به زمین و ایجاد صفر قوی صفر بر روی خط قرار می گیرد و به این صورت تضمین می شود که خط نمی سوزد. برای حل مشکل تداخل داده ها باید از مکانیزم داوری استفاده کنیم.

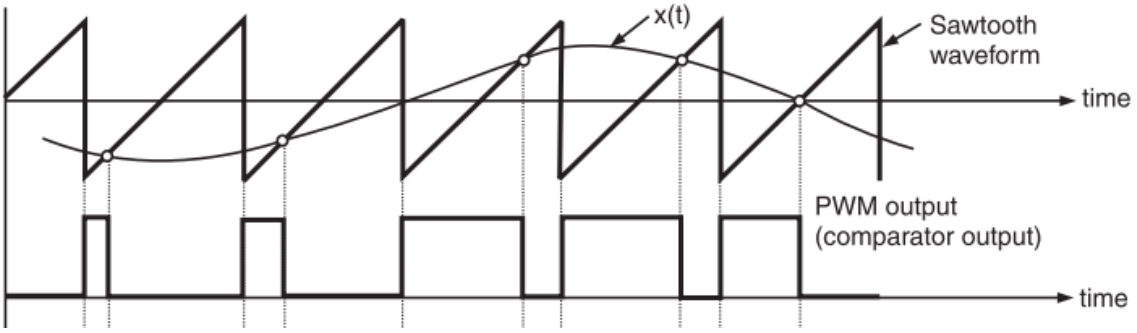
اگر دو یا چند master همزمان بخواهند داده ای را بر روی خط قرار دهند در زمانی که SCL در وضعیت High قرار دارد، داوری بر روی SDA انجام می شود. هر master بررسی می کند که سیگنال SDA بر روی bus با سیگنال SDA ی که تولید کرده برابر است یا نه. در صورت عدم تطابق master داوری را می بازد و باید تا زمان آزاد شدن خط صبر کند.

3. سیگنال PWM را می توان با استفاده از یک مقایسه کننده تولید کرد.

یک سیگنال دندان اره ای با فرکانس fs توسط sawtooth generator تولید می شود و به ترمینال معکوس کننده مقایسه کننده اعمال می شود. این سیگنال دندان اره ای به عنوان سیگنال نمونه گیری استفاده می شود. سیگنال تعدیل کننده (x (t modulating signal به ترمینال دیگر همان مقایسه کننده اعمال می شود. مقایسه کننده دو سیگنال را با هم مقایسه می کند و یک سیگنال PWM را به عنوان شکل موج خروجی تولید می کند.

هنگامی که مقدار لحظه ای x(t) بیشتر از سیگنال دندان اره ای باشد، سیگنال خروجی PWM در حالت High می ماند و در غیر این صورت در حالت Low است.

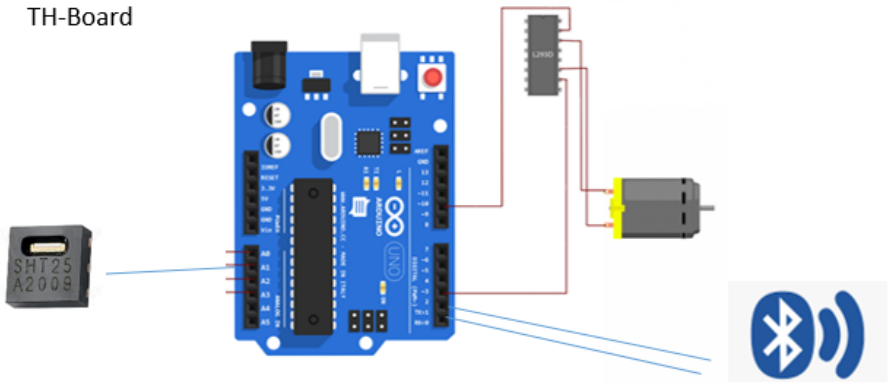




از شکل می توان دریافت که لبه ی بالا رونده‌ی PWM با لبه ی پایین رونده ی سیگنال دندان اره ای منطبق است. بنابراین لبه ی بالارونده ی PWM همواره در بازه زمانی های ثابتی تولید می شود. با این حال وقوع لبه ی پایین رونده PWM به مقدار لحظه ای x(t) بستگی دارد.

.4

TH-Board



Main-Board