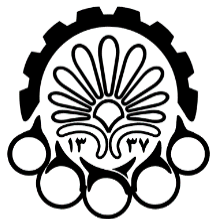


به نام خدا  
دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)  
دانشکده مهندسی کامپیوتر



دانشگاه صنعتی امیرکبیر  
( پلی تکنیک تهران )

آزمایشگاه ریزپردازنده

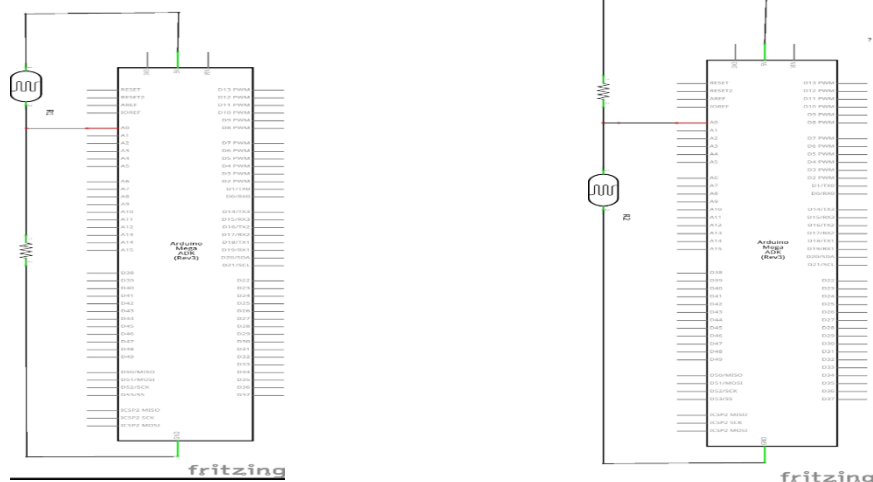
گزارش کار آزمایش پنجم

استاد درس: مهندس معصوم زاده

محمد امین رضائی / ۹۷۳۱۰۲۴

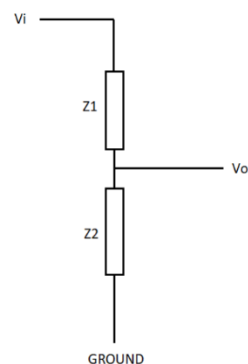
مهدی رحمانی / ۹۷۳۱۷۰۱

**سوال 1:** در مورد تفاوت دو مدار فوق تحقیق کنید. میزان ولتاژ خروجی هر کدام با تغییرات نور چگونه تغییر می‌کند.



باتوجه به مدار های بالا و توضیحات موجود در ویدیوی مربوط به آزمایش در کانال تلگرام آزمایشگاه، میتوان گفت شکل شماتیک مدار به صورت زیر میباشد و فرمول ولتاژ خروجی از رابطه‌ی زیر به دست می‌آید:

$$V_o = Z_2 \times \frac{V_i}{(Z_1 + Z_2)}$$



میدانیم که فوتوسل خودش یک مقاومت متغیر میباشد و در این شکل میتواند  $Z_1$  یا  $Z_2$  باشد. همچنین میدانیم که شدت نور با میزان مقاومت فوتوسل رابطه عکس دارد. حال هر کدام از حالت ها را بررسی میکنیم و رابطه‌ی

میزان ولتاژ خروجی را ابتدا با مقدار مقاومت فوتوسل و سپس شدت نور به دست می آوریم:  
حالت اول: Z1 فوتوسل باشد(شکل سمت چپ):

$$V_o = Z_2 \times \frac{V_i}{(Z_1 + Z_2)} \rightarrow Z_1 + Z_2 = Z_2 \times \frac{V_i}{V_o} \rightarrow Z_1 = Z_2 \times \frac{V_i}{V_o} - Z_2$$

$$\rightarrow \begin{cases} Z_1 \uparrow \rightarrow \text{Light} \downarrow \rightarrow V_o \downarrow \\ Z_1 \downarrow \rightarrow \text{Light} \uparrow \rightarrow V_o \uparrow \end{cases}$$

در این حالت بین روشنایی و ولتاژ رابطه مستقیم است.

حالت اول: Z2 فوتوسل باشد(شکل سمت راست):

$$V_o = Z_2 \times \frac{V_i}{(Z_1 + Z_2)} \rightarrow Z_1 + Z_2 = Z_2 \times \frac{V_i}{V_o} \rightarrow Z_1 = Z_2 \times \left( \frac{V_i}{V_o} - 1 \right)$$

$$Z_2 = \frac{Z_1}{\left( \frac{V_i}{V_o} - 1 \right)} \rightarrow Z_2 = \frac{Z_1 \times V_o}{V_i - V_o} \rightarrow Z_2 = Z_1 \times \left( -1 - \frac{V_i}{V_o - V_i} \right)$$

$$\rightarrow \begin{cases} Z_2 \uparrow \rightarrow \text{Light} \downarrow \rightarrow V_o \uparrow \\ Z_2 \downarrow \rightarrow \text{Light} \uparrow \rightarrow V_o \downarrow \end{cases}$$

در این حالت رابطه عکس داریم.

**سوال 2:** در مورد پایه‌های سنسور دما و همینطور نحوه تبدیل ولتاژ خروجی به میزان دما تحقیق کنید.

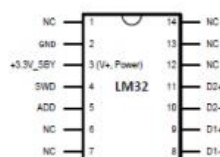


Figure 1. TSSOP-14

#### Pin Functions

Pin Descriptions			
Pin Number	Pin Name	Description	Typical Connection
1, 6, 7, 12, 13, 14	NC	No Connect	May be tied to V+, GND or left floating
2	GND	Ground	System ground
3	V+/ $\pm$ 3.3V_SBY	Positive power supply pin	Connected system 3.3 V standby power and to a 0.1 $\mu$ F bypass capacitor in parallel with 100 pF. A bulk capacitance of approximately 10 $\mu$ F needs to be in the near vicinity of the LM32.
4	SWD	SensorPath Bus line; Open-drain output	Super I/O, Pull-up resistor, 1.6k
5	ADD	Digital input - device number select input for the serial bus device number	Pull-up to 3.3 V or pull-down to GND resistor, 10k; must never be left floating
8, 10	D1-, D2-	Thermal diode analog voltage output and negative monitoring input	Remote Thermal Diode cathode (THERM_DC) - Diode 1 should always be connected to the processor thermal diode. Diode 2 may be connected to an MMBT3904 or GPU thermal diode. A 100 pF capacitor should be connected between respective D- and D+ for noise filtering.
9, 11	D1+, D2+	Thermal diode analog current output and positive monitoring input	Remote Thermal Diode anode (THERM_DA) - Diode 1 should always be connected to the processor thermal diode. Diode 2 may be connected to an MMBT3904 or GPU thermal diode. A 100 pF capacitor should be connected between respective D- and D+ for noise filtering.

با هر 10 میلی ولت تغییر ولتاژ ، دمای خروجی به اندازه یک درجه سانتی گراد افزایش میابد.

هنگامی که سنسور ولتاژ 500 میلی ولت را تولید می کند ، دما در 50 درجه سانتیگراد است

دمای خروجی 400 میلی ولت 40 درجه سانتیگراد است.

برای 600 mv دما 60 درجه سانتیگراد است ....

**سوال 3:** در مورد پایه‌های MOSI، SCLK، MISO در آردوینو Mega تحقیق کنید. پایه‌ی پیشفرض برای

SS کدام پایه است؟

MOSI دارای شماره پین 51 است و اگر مستر باشیم خروجی ما هست و اگر اسلیو باشیم ورودی ما است  
MISO دارای شماره پین 50 است و اگر مستر باشیم ورودی ما هست و اگر اسلیو باشیم خروجی ما است.  
SLCK دارای شماره پین 52 است و اگر مستر باشیم خروجی کلاک ما هست و اگر اسلیو باشیم ورودی کلاک  
ما است.

SS هم دارای شماره پین 53 هست (به صورت پیشفرض) که بصورت Active Low است ، اگر مستر باشیم  
خروجی ما هست و اگر اسلیو باشیم ورودی (فعال کننده) ما است.

**سوال 4:** در مورد نحوه‌ی انتخاب برد Slave توسط SS تحقیق نموده و نحوه پیاده‌سازی برنامه را برای این که برد مرکزی بتواند به ترتیب و در هر ثانیه برای یکی از بردهای Slave داده ارسال کند، شرح دهید. ( برای این کار بهتر است نمونه کدهایی که برای ارتباط بین دو آردوینو از طریق پروتکل SPI در اینترنت موجود است را بررسی نمایید. )

کافیست به ازای هر اسلیو یک پین SS در نظر بگیریم (البته اگر تعداد اسلیو ها از 4 تا بیشتر است می توان از دیگر استفاده کرد ). هر زمان که می خواهیم با اسلیو مورد نظر ارتباط برقرار کنیم کافیست که SS آن را LOW کنیم

```
void loop() {
    if (turn == 1) {
        sendData(val);
        turn = 2; delay(800);
    } else if (turn == 2) {
        sendData(val);
        turn = 1; delay(800);
    }
}

void sendData (int8_t data, int ss) {
    if (ss == 1)
        digitalWrite(slaveSelectPin1, LOW);
    else if (ss == 2)
        digitalWrite(slaveSelectPin2, LOW);
    delay(100);
    SPI.transfer(data);
    delay(100);
    if (ss == 1)
        digitalWrite(slaveSelectPin1, HIGH);
    else if (ss == 2)
        digitalWrite(slaveSelectPin2, HIGH);
}
```

با توجه به کد می توانیم در هر سری متغیر turn را تغییر دهیم و آن را به عنوان ورودی SS تابع sendData پاس دهیم.

**سوال 5:** مقدار کلاک توسط Master تعیین می‌شود یا Slave ؟

مقدار کلاک توسط Master تعیین میشود.

**سوال 6:** هر یک از تابع‌های نوشته شده را از راه لینک کتابخانه Wire، در مستندات آردوینو بررسی کنید.

- **begin()**

با تنظیم SCK، MOSI و SS بر روی خروجی‌ها، پول آپ کردن SCK و Low کردن MOSI و high کردن SS، گذرگاه SPI را آغاز می‌کند.

- **setClockDivider()**

تقسیم کننده کلاک SPI را نسبت به کلاک سیستم تنظیم می‌کند. در برده ای مبتنی بر AVR، تقسیم کننده های موجود 2، 4، 8، 16، 32، 64 یا 128 است. تنظیمات پیش فرض SPI\_CLOCK\_DIV4 است که کلاک SPI را به یک چهارم فرکانس کلاک سیستم تنظیم می‌کند.

- **transfer ()**

هم داده را ارسال می‌کند و هم داده دریافتی را در خروجی بر میگرداند.

- **attachInterrupt()**

اسلیو را برای دریافت interrupt فعال می‌کند (در واقع با آمدن داده، interrupt به اسلیو داده می‌شود تا آن را دریافت کند)

**سوال 7:** دستور مورد نیاز تا آردوینو در حالت Slave قرار گیرد را نوشته و در مورد کارایی آن تحقیق نمایید.

```
pinMode(SS, INPUT_PULLUP);  
pinMode(MOSI, INPUT);  
pinMode(SCK, INPUT);  
SPCR |= _BV(SPE);  
SPI.attachInterrupt();
```

را در setup نوشته و سپس تابعی را با فرمت زیر می نویسیم . که در آن می توانیم از رجیستر SPDR بایت دریافت شده را بخوانیم.

```
ISR (SPI_STC_vect)  
{  
  
    .....  
  
}
```

**سوال 8:** تابع ISR در کد Slave به چه منظور استفاده می شود؟ رجیستر مربوط به بایت دریافتی چیست ؟

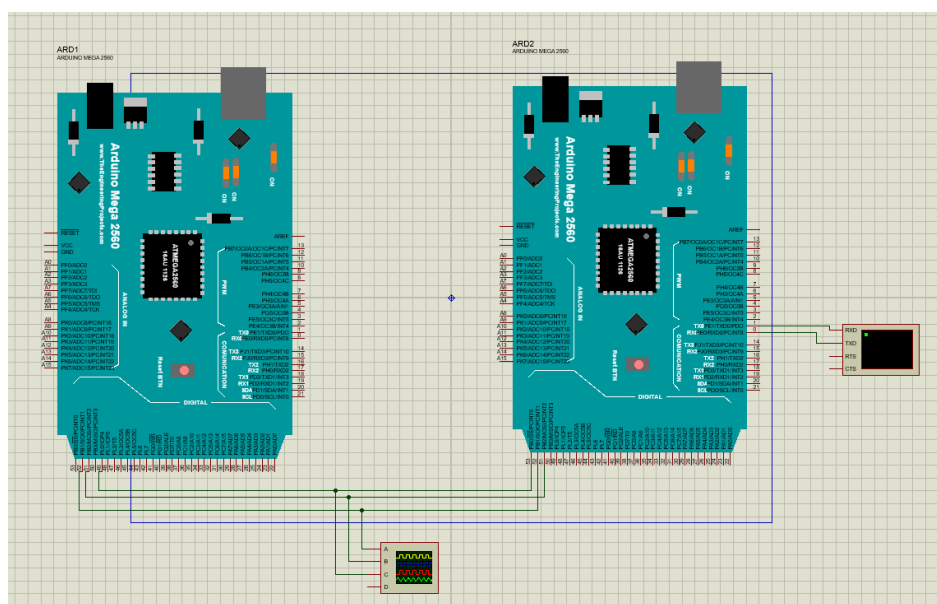
در واقع این تابع همان interrupt service routine ای است که پس از این که اسلیو داده ای را دریافت کرد باید صدا زده شود و رجیستری که آخرین بایت دریافت شده را در خود نگه می دارد SPDR است



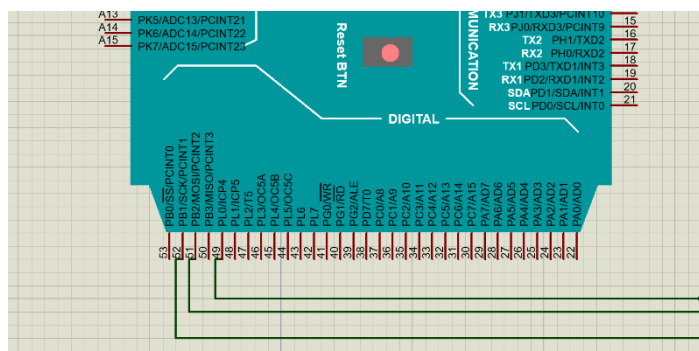
## گزارش کار

**سوال اول)** ارتباط میان دو دستگاه آردوینو از طریق SPI برقرار نمایید. بدین منظور دو برنامه یکی برای دریافت اطلاعات توسط برد Slave و دیگری برای ارسال اطلاعات از طریق برد master بنویسید. لازم به توضیح است master هر ثانیه کلمه اسم و شماره دانشجویی شما را برای برد Slave ارسال می کند. حتما پایه SS در آردوینو master را پایه‌ای به جز پایه پیش فرض آردوینو قرار دهید.

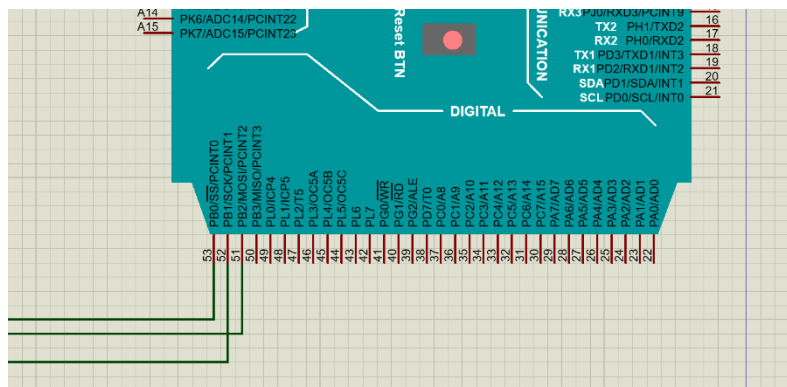
ابتدا مدار مورد نظر را به صورت زیر می‌بینید:



در این مدار آردوینوی سمت چپ نقش Master را دارد و آردوینوی سمت راست نقش Slave را دارد. در شکل زیر پین‌های مربوط به آردینوی Master را می‌بینید که در آن پین شماره‌ی 52 پین SCK یا کلاک ما می‌باشد و پین شماره‌ی 51 پین MOSI و پین شماره‌ی 49 پین SS یا سلکت ما می‌باشد. (طبق خواسته‌ی سوال پایه‌ی پیشفرض آردوینو را استفاده نکردیم).



در شکل زیر پین های مربوط به آردینوی Slave را میبینید که در آن پین شماره‌ی 52 پین SCK یا کلاک ما میباشد و پین شماره‌ی 51 پین MOSI و پین شماره‌ی 53 پین SS یا سلکت ما میباشد. در اینجا چون صورت سوال چیزی راجع به پایه‌ی select قابل استفاده در slave نگفته است میتوان همان پایه‌ی شماره 53 را گرفت.



در اینجا چون فقط میخواستیم به کمک Master روی Slave داده بنویسیم بنابراین از پایه MISO استفاده نکردیم.

حال به سراغ کد میرویم. کد نیز دارای دو بخش Master و Slave میباشد. کد Master ابتدا بررسی میکنیم:

```

master | Arduino 1.8.13
File Edit Sketch Tools Help

master
#include <SPI.h>

const int slaveSelectPin1 = 49;

void setup() {
    pinMode(slaveSelectPin1, OUTPUT);
    digitalWrite(slaveSelectPin1, HIGH);

    SPI.begin();
}

void loop() {
    sendData("Amin9731024/Mahdi9731701", 24);
    delay(200);
}

```

ابتدا باید کتابخانه‌ی مربوط به SPI را include کنیم. سپس چون از ما خواسته شده از پایه‌ی پیشفرض برای SS استفاده نکنیم بنابراین از پایه‌ی 49 استفاده میکنیم و آن را به عنوان slaveSelectPin1 تعریف میکنیم. در قسمت Setup باید تنظیمات مربوط به این پایه‌ی SS را تعیین کنیم. باید بگوییم هم یک پایه‌ی خروجی هست هم اینکه باید در ابتدای کار مقدار HIGH داشته باشد. (چرا که میدانیم این پایه به صورت Active Low میباشد و در ابتدای کار میخواهیم غیرفعال باشد). همچنین باید SPI.begin() را بگوییم تا با تنظیم SCK، MOSI و SS بر روی خروجی ها، پول آپ کردن SCK و Low کردن MOSI، گذرگاه SPI را آغاز کند. در قسمت loop نیز تابع sendData را صدا زدیم که دو آرگومان ورودی میگیرد. یکی آرایه‌ی ای از کاراکترها و

استرینگی که میخواهیم بفرستیم و دیگری سائز این آرایه میباشد.

در زیر کد مربوط به این تابع را میبینید:

```
void sendData (char data[], int _size)
{
    digitalWrite(slaveSelectPin1, LOW);

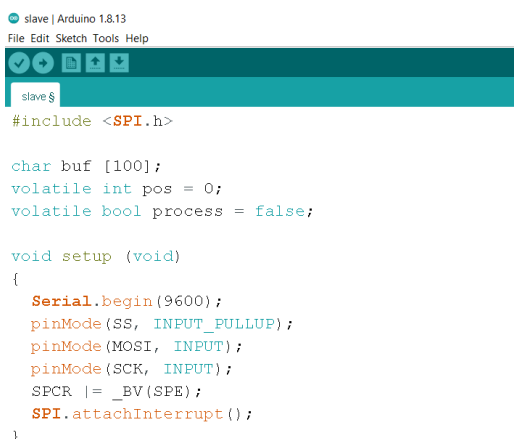
    delay(100);
    for (int i = 0; i < _size; i++)
    {
        SPI.transfer(data[i]);
        delay(25);
    }
    SPI.transfer('z');
    delay(100);

    digitalWrite(slaveSelectPin1, HIGH);
}
```

در اینجا نیز در مرتبه‌ی اول اومدیم پایه‌ی مربوط به SS را LOW کردیم تا Slave مورد نظر فعال شود و داده را دریافت کند.

سپس به کمک یک حلقه و تابع transfer داده را کاراکتر به کاراکتر ارسال میکنیم. همچنین از ما خواسته شده که در هر یک ثانیه اسم و شماره دانشجویی ما را ارسال کند. بنابراین مجموع delay ها برابر با 1 ثانیه میباشد.

در آخر نیز باید پایه‌ی مربوط به SS را HIGH کنیم تا Slave مورد نظر غیرفعال شود. حال به سراغ کد slave میرویم:



```
slave $
#include <SPI.h>

char buf [100];
volatile int pos = 0;
volatile bool process = false;

void setup (void)
{
    Serial.begin(9600);
    pinMode(SS, INPUT_PULLUP);
    pinMode(MOSI, INPUT);
    pinMode(SCK, INPUT);
    SPCR |= _BV(SPE);
    SPI.attachInterrupt();
}
```

در اینجا نیز ابتدا باید کتابخانه SPI را include کنیم. سپس یک بافر دریافت تعریف میکنیم. دومتغیر volatile به نام های pos و process تعریف میکنیم که در ادامه کارایی آن ها را میبینیم. در تابع setup باید برخلاف Master یک سری تنظیمات پایه ها را دستی انجام دهیم. یعنی بگوییم SS به صورت ورودی و pull up باشد. همچنین MOSI و SCK ما نیز باید به صورت ورودی باشند.

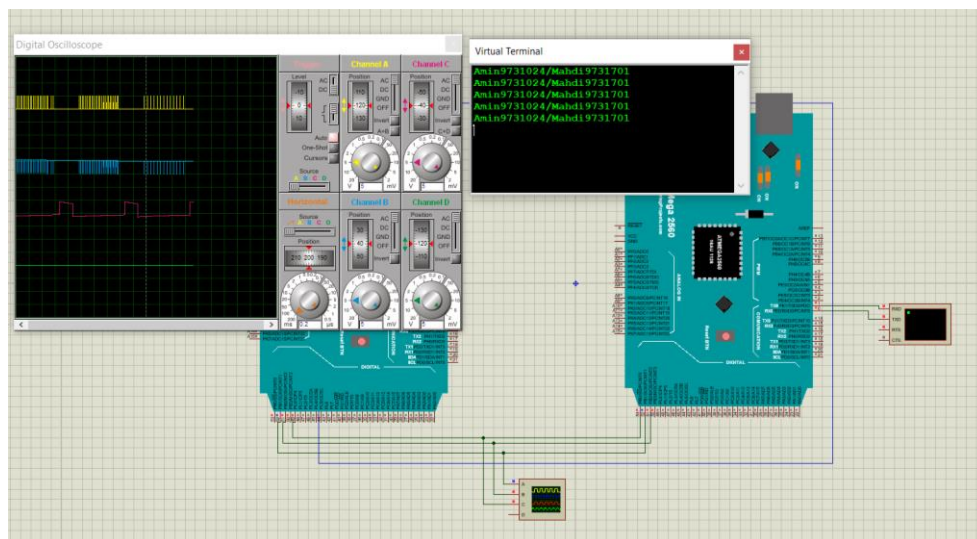
در ضمن برای کار با ترمینال باید Serial.begin را صدا بزنیم. تابع SPI.attachInterrupt هم که توضیحش در پیشگزارش آمد را هم باید صدا بزنیم.

```
ISR (SPI_STC_vect)
{
    char c = SPDR;
    if (pos < 100 && process == false)
    {
        if (c == 'z')
            process = true;
        else
        {
            buf[pos] = SPDR;
            pos++;
        }
    }
}

void loop (void)
{
    if (process)
    {
        buf[pos] = 0;
        Serial.println (buf);
        pos = 0;
        process = false;
    }
}
```

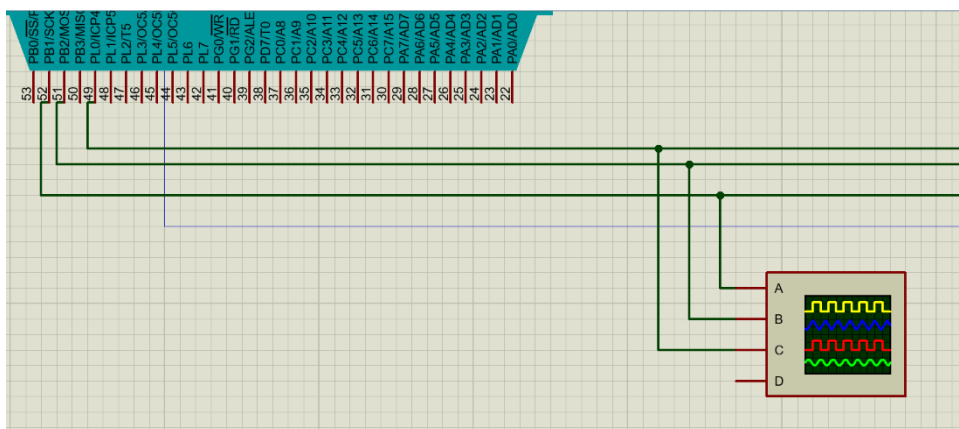
در بالا کد مربوط به ISR آمده. وقتی کاراکتر به کاراکتر ورودی ها را دریافت میکنیم چک میکنیم اگر بافر پر نشده باشد و هنوز به حرف Z نرسیده باشیم آن ها را دونه دونه در بافر ذخیره میکنیم و کانتر بافر را یکی زیاد میکنیم. وقتی یک خط دریافت میشود و به حرف Z میرسیم مقدار process را true میکنیم. بعد از آن در قسمت loop وقتی میبیند که مقدار process برابر true شد مقدار بافر را در ترمینال چاپ میکنیم و pos را برابر با 0 و مقدار process را برابر false قرار میدهیم تا دوباره خط بعدی را در بافر از اول دریافت کند.

خروجی به صورت زیر میباشد:



**سوال دوم) موج خروجی سه پایه SS، MOSI، SCLK را برای سه مقدار Clock توسط اسیلوسکوپ مشاهده کنید.**

ابتدا پایه های اسلیسکوپ را بررسی میکنیم که به چه پین هایی متصل شده اند:

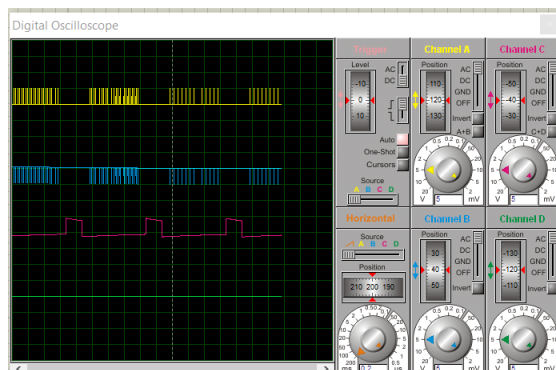


پایه ی A به پین 52 که برای کلاک یا همان SCK هست، وصل شده است.  
پایه ی B به پین 51 که برای MOSI میباشد وصل شده است.  
پایه ی C به پین 43 که برای SS هست، وصل شده است.

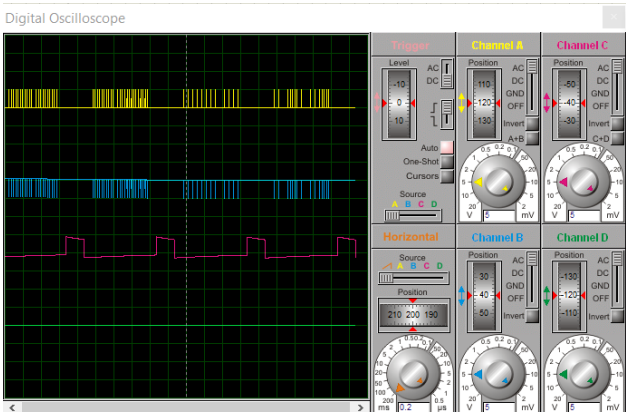
حال برای ایجاد کلاک های مختلف میتوان یک خط زیر را در قسمت setup اضافه کرد:

```
void setup() {  
    pinMode(slaveSelectPin1, OUTPUT);  
    digitalWrite(slaveSelectPin1, HIGH);  
    SPI.setClockDivider(SPI_CLOCK_DIV2);  
    SPI.begin();  
}
```

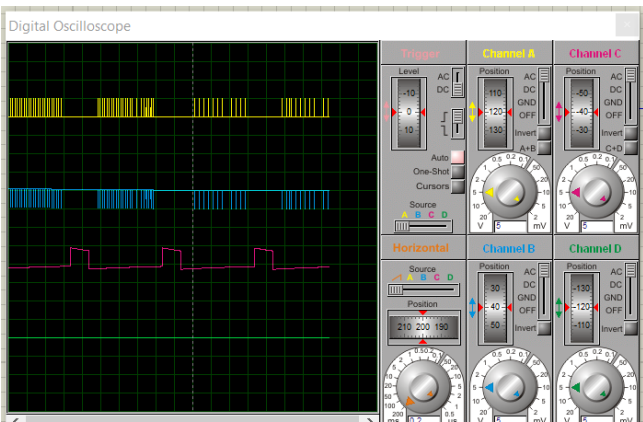
برای  $SPI\_CLOCK\_DIV = 2$  :



برای  $\text{SPI\_CLOCK\_DIV} = 8$ :



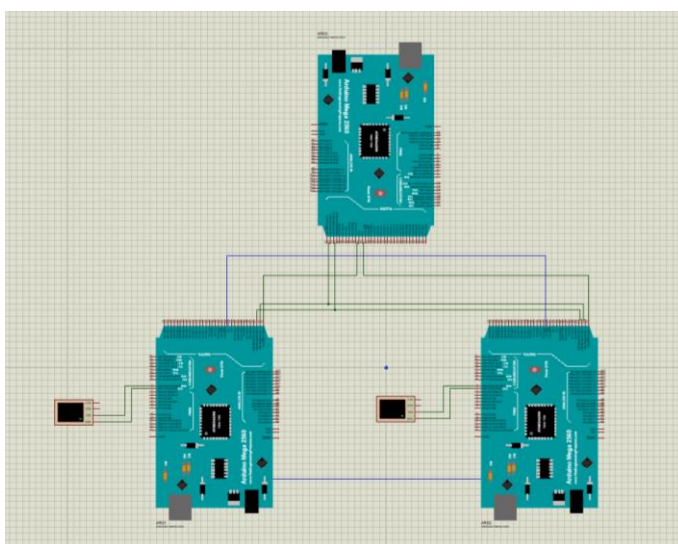
برای  $\text{SPI\_CLOCK\_DIV} = 32$ :



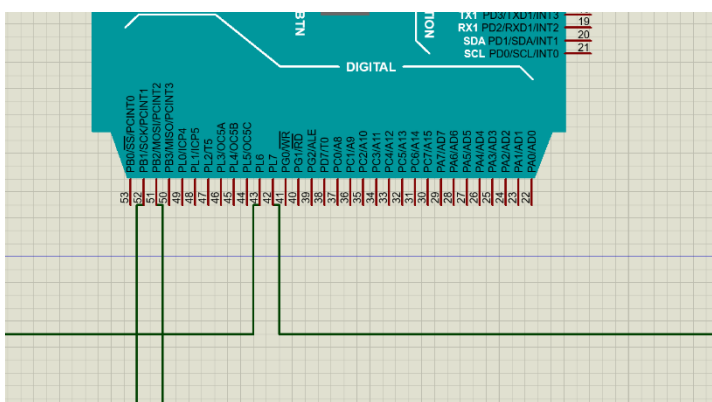
ابته در شکل ها به خوبی میزان تراکم کلاک ها در خط اول قابل تشخیص نیست.

**سوال سوم)** کد قسمت Master و اتصالات آن را به گونه ای تغییر دهید که بعد از اضافه کردن یک Slave و قرار دادن کد مربوط به برد Slave، به طور متناوب و هر ثانیه به آردوینو دوم اسم شما ارسال شود و در ثانیه بعدی آردوینو اول کلمه "Hello" your name را دریافت کند.

ابتدا مدار مربوطه را به صورت زیر میبندیم:



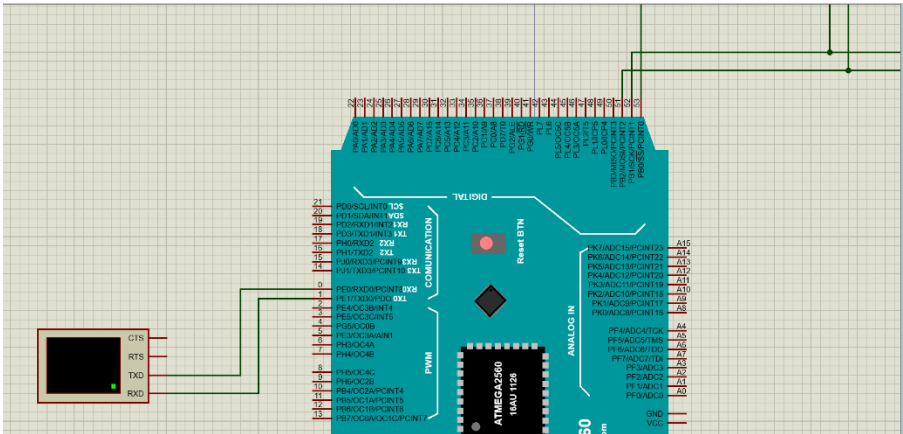
ما ، آردوینوی بالایی میباشد. دو آردوینوی پایینی Slave های ما میباشند. به هرکدام از آن ها یک Terminal برای نمایش داده ای که به آن ها توسط Master ارسال شده است، متصل میکنیم. پایه های مربوط به Master به صورت زیر میباشند:



پین شماره 52 پین SCK یا کلاک ما میباشد و پین شماره 51 پین MOSI و پین های شماره 42 و 43 پین های SS یا سلکت ما میباشند. شماره 42 مربوط به SS آردوینوی Slave سمت راستی و شماره 43

مربوط به SS آردوینوی Slave سمت چپی میباشد.

در زیر پین های مربوط به Slave سمت چپی میباشد:



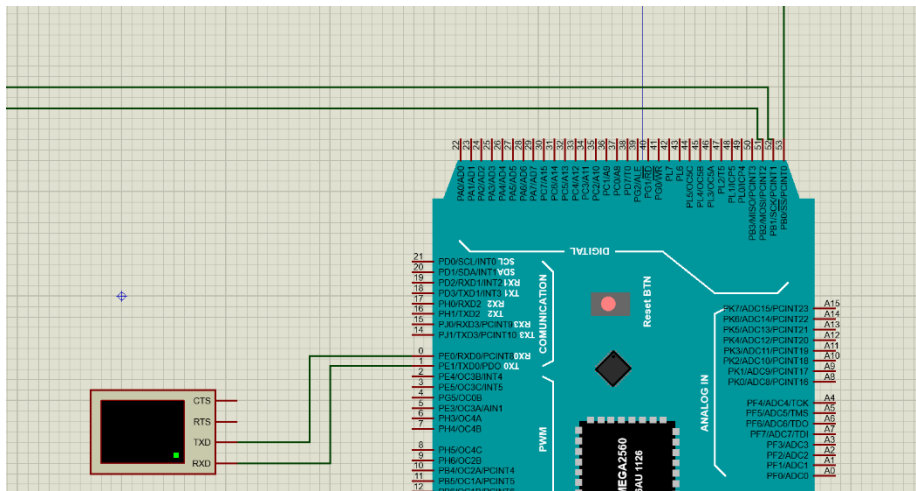
پین شماره‌ی 52 پین SCK یا کلاک ما می‌باشد و پین شماره‌ی 51 پین MOSI و پین‌های شماره‌ی 53 پین

SS یا سلکت ما می باشد که به 43 ی Master وصل می باشد.

همچنین برای ترمینال مجازی پایه TXD آن را به RXDی آردوینو و پایه ی RXD آن را به TXD آردوینو

وصل میکنیم.

در زیر پین های مربوط به Slave سمت راستی میباشد:



شبيه Slave سمت چپ ميباشد. پين شماره 52 پين SCK يا كلاک ما ميباشد و پين شماره 51 پين MOSI

و پین‌های شماره ی 53 پین SS یا سلکت ما می باشد که به 42 ی Master وصل می باشد.

همچنین برای ترمینال مجازی پایه TXD آن را به RXDی آردوینو و پایه ی RXD آن را به TXD آردوینو

وصل میکنیم.



حال به سراغ کدها میرویم:  
در ابتدا به کد مربوط به Master میپردازیم که در آن تغییرات داشتیم:

```
master
#include <SPI.h>

const int slaveSelectPin1 = 43;
const int slaveSelectPin2 = 42;

int turn = 1;

void setup() {
    pinMode(slaveSelectPin1, OUTPUT);
    digitalWrite(slaveSelectPin1, HIGH);

    pinMode(slaveSelectPin2, OUTPUT);
    digitalWrite(slaveSelectPin2, HIGH);

    SPI.begin();
}

void loop() {
    if (turn == 1) {
        {
            sendData("Hello Amin & Mahdi", 18, turn);
            turn = 2;
            delay(200);
        } else if (turn == 2) {
        {
            sendData("Amin/Mahdi", 10, turn);
            turn = 1;
            delay(300);
        }
    }
}

void sendData (char data[], int _size, int ss)
{
    if (ss == 1)
        digitalWrite(slaveSelectPin1, LOW);
    else if (ss == 2)
        digitalWrite(slaveSelectPin2, LOW);

    delay(100);
    for (int i = 0; i < _size; i++)
    {
        SPI.transfer(data[i]);
        delay(25);
    }
    SPI.transfer('z');
    delay(200);

    if (ss == 1)
        digitalWrite(slaveSelectPin1, HIGH);
    else if (ss == 2)
        digitalWrite(slaveSelectPin2, HIGH);
}
```

در قسمت قبل بخش های مختلف و جزئیات را توضیح دادیم. در این قسمت نیز فقط برخی قسمت های کد تغییر کرده است.

در ابتدا پس از include کردن کتابخانه SPI ، چون دوتا Slave داریم برای آن ها دوتا پین SS مجزا تعریف میکنیم. سپس در setup تنظیمات آن ها را انجام میدهیم و میگوییم که خروجی و pull up باشند. یک متغیر turn تعریف کردیم که به کمک آن میتوانیم بگوییم در هر نوبت کدام Slave را فعال کن. وقتی turn برابر با 1 باشد نوبت Slave اول میباشد و باید Hello Amin & Mahdi را چاپ کند و بعد متغیر turn را برابر با 2 قرار دهد تا دفعه ی بعد نوبت Slave دوم ود و آن Amin/Mahdi را چاپ کند. تابع sendData هم مانند قبل میباشد فقط یک متغیر SS به عنوان ورودی میگیرد که همان نوبت slave ها میباشد. Slave مربوطه را فعال میکنیم و بعد با یک حلقه کاراکتر به کاراکتر استرینگ را به Slave مورد نظر میفرستیم.

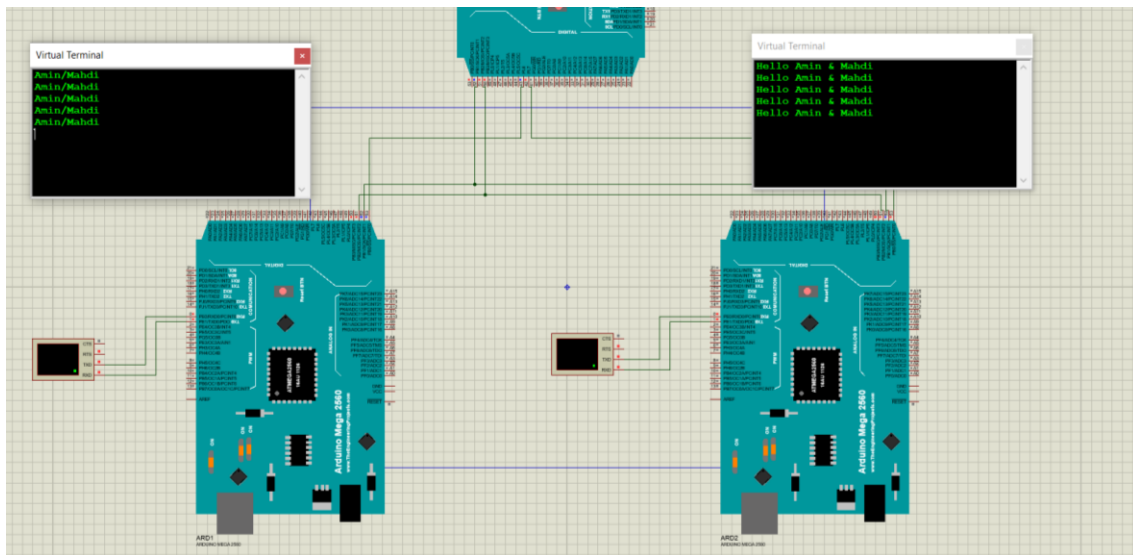
کد مربوط به slave ها مانند سری قبل میباشد و تغییری در آن ها ندادیم:

```
slave $
#include <SPI.h>

char buf [100];
volatile int pos = 0;
volatile bool process = false;

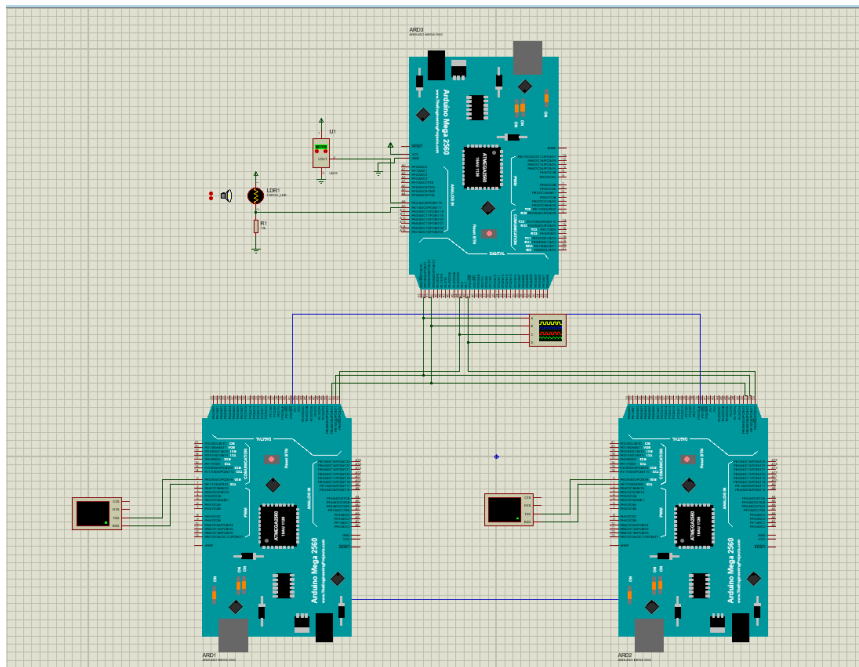
void setup (void)
{
  Serial.begin(9600);
  pinMode(SS, INPUT_PULLUP);
  pinMode(MOSI, INPUT);
  pinMode(SCK, INPUT);
  SPCR |= _BV(SPE);
  SPI.attachInterrupt();
}
ISR (SPI_STC_vect)
{
  char c = SPDR;
  if (pos < 100 && process == false)
  {
    if (c == 'z')
      process = true;
    else
    {
      buf[pos] = SPDR;
      pos++;
    }
  }
}
void loop (void)
{
  if (process)
  {
    buf[pos] = 0;
    Serial.println (buf);
    pos = 0;
    process = false;
  }
}
```

همچنین خروجی این قسمت به صورت زیر میباشد:

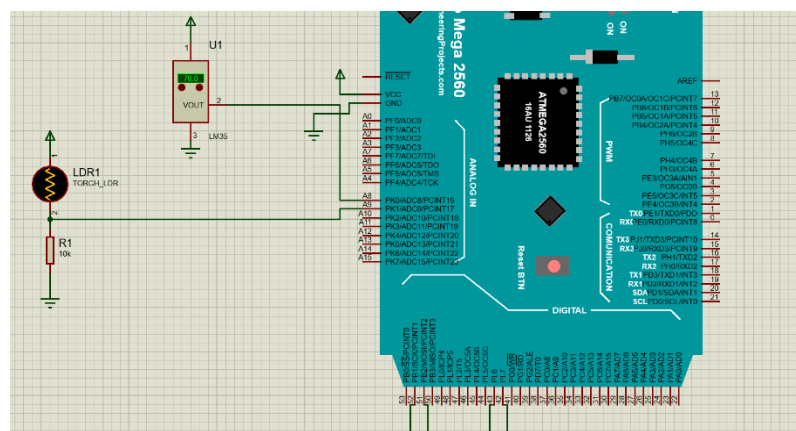


**سوال چهارم)** داده‌های Hello world و Hi را با اطلاعات مربوط به دو سنسور دما و نور جای‌گزین نمایید. برای خواندن ولتاژ خروجی هر یک از سنسورها کافیست از دستور analogRead استفاده نمایید. سپس عدد به دست آمده را به بازه مناسب map نمایید.

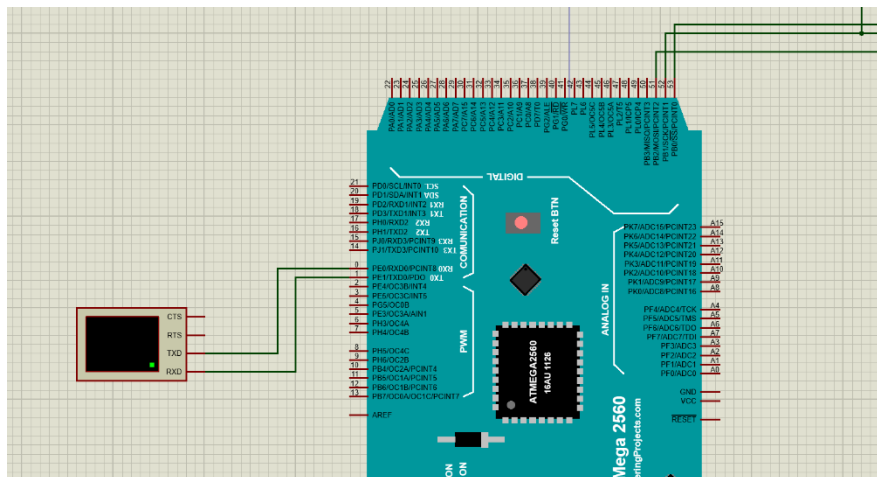
مدار موردنظر به صورت زیر میباشد:



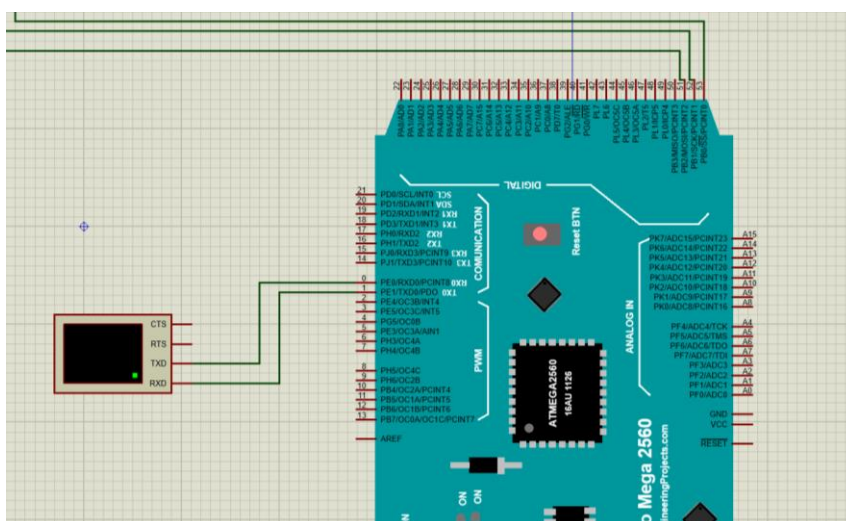
مانند مثال قبل آردوینوی بالایی آردوینوی Master میباشد. آردوینوی سمت چپ Slave میباشد و دما را به آن میفرستیم تا روی ترمینال مجازی نشان دهد. آردوینوی سمت راست Slave میباشد و مقدار درصد روشنایی را به آن میفرستیم تا روی ترمینال نشان دهد. پورت‌های ورودی خروجی Master به صورت زیر میباشد:



پین شماره‌ی 52 پین SCK یا کلاک ما میباشد و پین شماره‌ی 51 پین MOSI و پین‌های شماره‌ی 42 و 43 پین‌های SS یا سلکت ما میباشد. شماره‌ی 42 مربوط به SS آردینوی Slave سمت راستی و شماره‌ی 43 مربوط به SS آردینوی Slave سمت چپی میباشد. همچنین VCC و GND آن را نیز به منبع تغذیه و زمین وصل کردیم. پین A8 را به سنسور دما و پین A9 را به سنسور نور ( فوتوسل) متصل کردیم. در تصویر زیر پین‌های مربوط به Slave مربوط به دما میباشد:



پین شماره‌ی 52 پین SCK یا کلاک ما میباشد و پین شماره‌ی 51 پین MOSI و پین‌های شماره‌ی 53 پین SS یا سلکت ما میباشد که به 43 ی Master وصل میباشد. همچنین برای ترمینال مجازی پایه TXD آن را به RXD آردینو و پایه RXD آن را به TXD آردینو وصل میکنیم. در تصویر زیر پین‌های مربوط به Slave مربوط به نور میباشد:



شبه Slave سمت چپ میباشد. پین شماره‌ی 52 پین SCK یا کلاک ما میباشد و پین شماره‌ی 51 پین MOSI و پین‌های شماره‌ی 53 پین SS یا سلکت ما میباشد که به 42 ی Master وصل میباشد. همچنین برای ترمینال مجازی پایه TXD آن را به RXD ی آردوینو و پایه‌ی RXD آن را به TXD آردوینو وصل میکنیم.

حال به سراغ کدهای این بخش میرویم:  
کد مربوط به Master به صورت زیر میباشد:

master | Arduino 1.8.13  
File Edit Sketch Tools Help

```
master $  
#include <SPI.h>  
  
const int slaveSelectPin1 = 43;  
const int slaveSelectPin2 = 42;  
const int tempPIN = A8;  
const int lightPIN = A9;  
  
int analogValue;  
uint8_t mapped;  
int turn = 1;  
  
void setup() {  
  
  pinMode(slaveSelectPin1, OUTPUT);  
  digitalWrite(slaveSelectPin1, HIGH);  
  
  pinMode(slaveSelectPin2, OUTPUT);  
  digitalWrite(slaveSelectPin2, HIGH);  
  
  pinMode(lightPIN, INPUT);  
  pinMode(tempPIN, INPUT);  
  
  SPI.begin();  
}
```

```
master $  
  
void loop() {  
  if (turn == 1)  
  {  
    delay(400);  
    analogValue = analogRead(tempPIN);  
    mapped = map(analogValue, 0, 306, 0, 150);  
    sendData(mapped, turn);  
    turn = 2;  
    delay(200);  
  } else if (turn == 2) {  
    delay(200);  
    analogValue = analogRead(lightPIN);  
    mapped = map(analogValue, 0, 1023, 0, 100);  
    sendData(mapped, turn);  
    turn = 1;  
    delay(200);  
  }  
}  
void sendData (uint8_t data, int ss)  
{  
  if (ss == 1)  
    digitalWrite(slaveSelectPin1, LOW);  
  else if (ss == 2)  
    digitalWrite(slaveSelectPin2, LOW);  
  delay(100);  
  SPI.transfer(data);  
  delay(100);  
  if (ss == 1)  
    digitalWrite(slaveSelectPin1, HIGH);  
  else if (ss == 2)  
    digitalWrite(slaveSelectPin2, HIGH);  
}
```

همانطور که مشاهده میشود بخش setup و library و متغیرهای تعریف شده مانند قبل است با این تفاوت که دو متغیر tempPin و lightPin که مربوط به ورودی‌های آنالوگ ما هستند، را هم جداگانه تعریف کردیم و در setup به صورت ورودی تنظیمات آن را درست کردیم. در قسمت loop یک سری تغییرات داریم. از جمله اینکه این مقدار آنالوگی که در ورودی میگیریم را باید به کمک تابع map به یک مقدار دلخواهی map کنیم. این کار را با اندازه گیری ها هم برای light هم temperature انجام میدهیم. در نهایت مقدار mapped شده را به تابع sendData میدهیم تا برای به سمت Slave های موردنظر که نوبتشان هست ارسال کند.

حال به سراغ کدهای Slave ها میرویم:

کد سمت چپ برای slave مربوط به نمایش روشنایی و کد سمت راست برای slave مربوط به نمایش دما می باشد:

```
slave | Arduino 1.8.13
File Edit Sketch Tools Help

slave
#include <SPI.h>
int value;
volatile bool process = false;

void setup (void)
{
    Serial.begin(9600);
    pinMode(SS, INPUT_PULLUP);
    pinMode(MOSI, INPUT);
    pinMode(SCK, INPUT);
    SPCR |= _BV(SPE);
    SPI.attachInterrupt();
}

ISR (SPI_STC_vect)
{
    byte number = SPDR;
    value = (uint8_t)number;
    process = true;
}

void loop (void)
{
    if (process)
    {
        Serial.print("Light: ");
        Serial.print(value);
        Serial.println("%");
        process = false;
    }
}
```

```
slave | Arduino 1.8.13
File Edit Sketch Tools Help

slave §
#include <SPI.h>

int value;
volatile bool process = false;

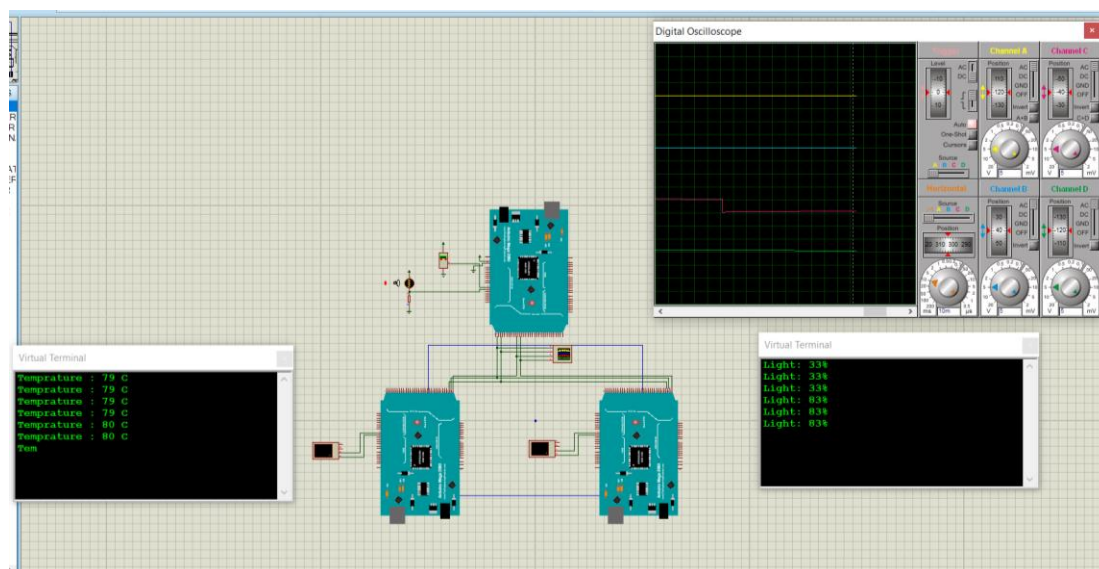
void setup (void)
{
    Serial.begin(9600);
    pinMode(SS, INPUT_PULLUP);
    pinMode(MOSI, INPUT);
    pinMode(SCK, INPUT);
    SPCR |= _BV(SPE);
    SPI.attachInterrupt();
}

ISR (SPI_STC_vect)
{
    byte number = SPDR;
    value = (uint8_t)number;
    process = true;
}

void loop (void)
{
    if (process)
    {
        Serial.print("Temperature : ");
        Serial.print(value);
        Serial.println(" C");
        process = false;
    }
}
```

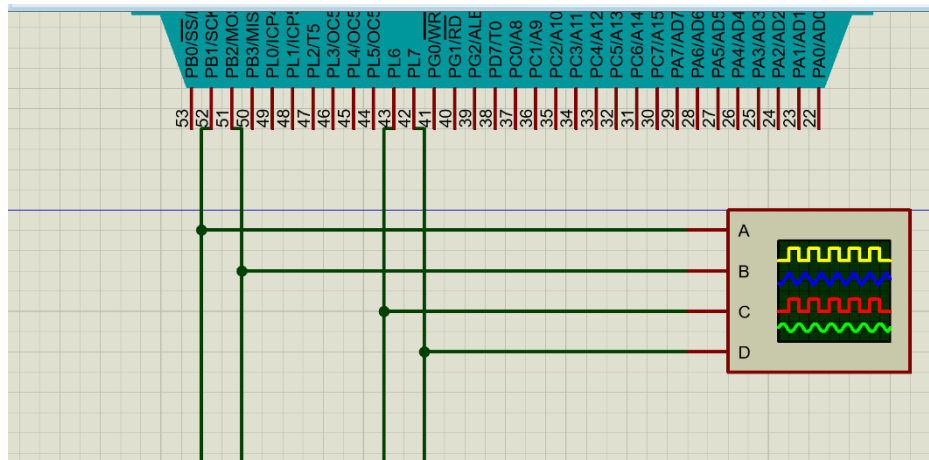
همانطور که مشاهده میشود کدهای مربوط به اسلیو ها تغییر خاصی نکرده اند جز اینکه در قسمت loop برای نمایش بهتر و مشخص تر دما و روشنایی در ترمینال رشته ها و پیام هایی را هم اضافه تر به همراه داده ای که از Master میگیرند چاپ میکنیم.

خروجی این بخش به صورت زیر می باشد:



**سوال پنجم)** موج خروجی را برای چهار پایه SS1, SS2, SCLK, MOSI توسط اسیلوسکوپ مشاهده نمایید.

ابتدا پایه های اسلیسکوپ را بررسی میکنیم که به چه پین هایی متصل شده اند:



پایه ی A به پین 52 که برای کلاک یا همان SCK هست، وصل شده است.

پایه ی B به پین 51 که برای MOSI میباشد وصل شده است.

پایه ی C به پین 43 که برای SS مربوط به اسلیو نمایش دهنده ی دما (سمت چپی) هست، وصل شده است.

پایه ی D به پین 42 که برای SS مربوط به اسلیو نمایش دهنده ی روشنایی (سمت راستی) هست، وصل شده است.

موج خروجی به صورت زیر میباشد:

