نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

ارسال میزان دما و نور با SPI

استاد: **آقای معصوم‌زاده**
درس: **آزمایشگاه ریزپردازنده و زبان اسمبلی**

- ## هدف آزمایش

آشنایی با پروتکلSPI ، تحلیل موج خروجی آردوینوی master و راه اندازی حسگر نور و دما

- ## وسایل مورد نیاز

  - ### برد آردوییِنو مگا

برنامه های slave و master روی این برد ها صورت میگیرد.

  - ### مقاومت متغیر فتوسل یا LDR

این مقاومت با تغییرات میزان نور تغییر میکند. این تغییرات با میزان نور رابطه عکس دارد. برای تبدیل تغییرات مقاومت این قطعه به تغییرات ولتاژ میتوان از مدار تقسیم ولتاژ بهره برد.

  - ### سنسور دما

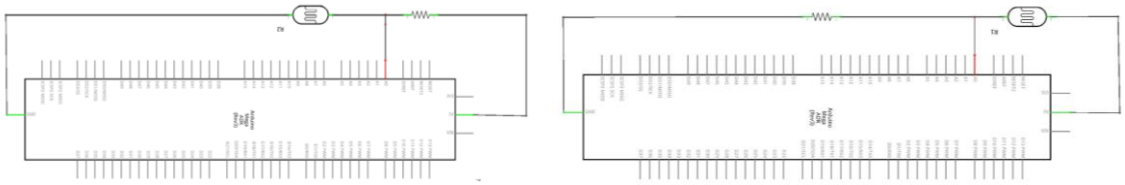
این سنسور میزان دمای محیط را بر حسب درجه سانتیگراد به ولتاژ آنالوگ تبدیل میکند.

- ## توابع جدید

begin()	شروع ارتباط و ست کردن باس های ارتباطی (SCK و MOSI بصورت pull-down و SS بصورت pull-up)
setClockDivider()	تعیین مقدار فرکانس کلاک. تنظیمات پیشفرض آن ۱/۴ است. یعنی میزان فرکانس SPI برابر ۱/۴ فرکانس سیستم میشود.
transfer()	دیتا را از slave در پاسخ به درخواست master میفرستد. یا اینکه بایت ها را برای انتقال از master به slave در صف میگذارد.
attachInterrupt()	فعال سازی وقفه ها در SPI

- ## سوالات تئوری

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



*در شکل **چپ**: افزایش نور ← کاهش مقاومت ←کاهش ولتاژ*
*در شکل **راست**: افزایش نور ← کاهش مقاومت ←افزایش ولتاژ*

  - **در مورد پایه های سنسور دما و همینطور نحوه تبدیل ولتاژ خروجی به میزان دما تحقیق کنید.**

سه پایه دارد:
VCC
GND
ولتاژ خروجی (تبدیل تغییرات دما (۵۵- تا ۱۵۰+ درجه سلسیوس) به ولتاژ آنالوگ)

- **در مورد پایه های SCLK، MISO، MOSI در آردوینو Mega تحقیق کنید. پایه ی پیشفرض برای SS کدام پایه است؟ برای مشاهده آن میتوانید به محل نصب آردوینو رفته، مسیر زیر را دنبال نمایید و در انتها فایل داخل پوشه را باز نمایید: hardware -> arduino -> avr -> variants -> mega**

```
#define PIN_SPI_SS    (53)
#define PIN_SPI_MOSI  (51)
#define PIN_SPI_MISO  (50)
#define PIN_SPI_SCK   (52)
```

- **در مورد نحوهی انتخاب بورد Slave توسط SS تحقیق نموده و نحوه پیاده سازی برنامه را برای اینکه بورد مرکزی بتواند به ترتیب و در هر ثانیه برای یکی از بردهای Slave داده ارسال کند، شرح دهید. (برای اینکار بهتر است نمونه کدهایی که برای ارتباط بین دو آردوینو از طریق پروتکل SPI در اینترنت موجود است را بررسی نمایید.)**

برای انتخاب slave و شروع ارتباط با آن از سمت master پایه ss متصل به slave را در حالت LOW قرار میدهیم. حال پس از اتمام ارتباط مقدار گفته شده را HIGH میکنیم. (این پایه active-low میباشد)

- **مقدار کلاک توسط Master تعیین میشود یا Slave ؟**

توسط master

- **هر یک از تابع های نوشته شده را از راه لینک کتابخانه Wire ،در مستندات آردوینو بررسی کنید.**

*جواب در قسمت **توابع جدید***

- **دستور مورد نیاز برای اینکه آردوینو در حالت Slave قرار گیرد، را نوشته و در مورد کارایی آن تحقیق نمایید.**

از دستور زیر استفاده میکنیم:

```
SPCR |= _BV(SPE);
```

که اگر داده ای از master دریافت شود، تابع ISR فرخوانی شده و داده از رجیستر SPDR خوانده میشود.

- **تابع ISR در کد Slave به چه منظور استفاده میشود؟ رجیستر مربوط به بایت دریافتی چیست ؟**

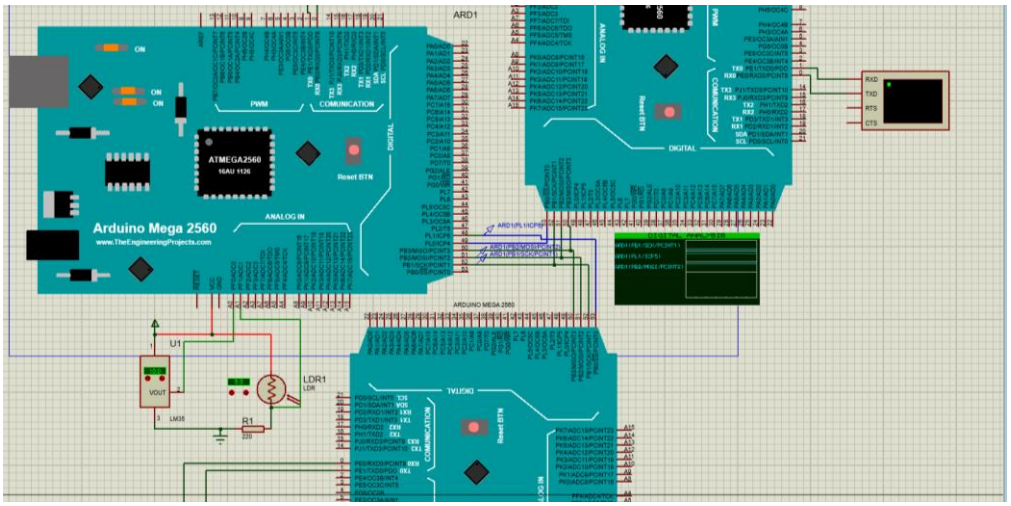
هنگامی که در پروتکل SPI از سمت master به slave داده ای ارسال میشود، این عمل در slave یک وقفه بوجود می آورد و ما در این تابع مشخص میکنیم که با این وقفه که دارای داده ای است که محتوای آن در ثبات SPDR ذخیره شده باید چه کنیم.

نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

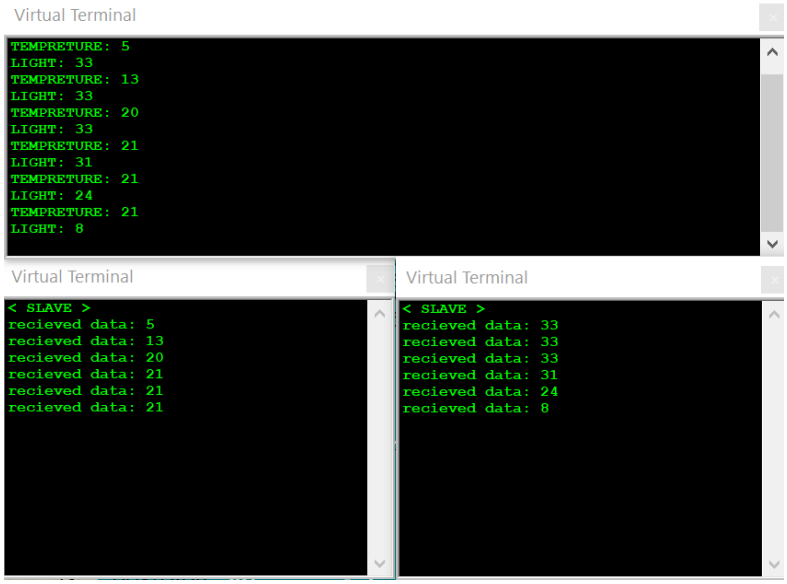
ارسال میزان دما و نور با SPI

استاد: **آقای معصوم‌زاده**
درس: **آزمایشگاه ریزپردازنده و زبان اسمبلی**

- **مدار**



- **نمونه خروجی**



- **کد** (master)

```
//MASTER
#include <SPI.h>
#define ss1 49
#define ss2 48
#define PIN_TEMPRETURE A0
#define PIN_LIGHT A1

String tempreture_value ;
```

نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

ارسال میزان دما و نور با SPI

استاد: **آقای معصوم‌زاده**
درس: **آزمایشگاه ریزپردازنده و زبان اسمبلی**

```
String light_value;

void transferStringToSlave(int port){

    // tempreture or light...
    // what do we want to transfer?...
    bool is_temp;
    if (port == ss1){
      is_temp = true;
      Serial.print("TEMPRETURE: ");
    }
    else if (port == ss2) {
      is_temp = false;
      Serial.print("LIGHT:      ");
    }
    else return;

    // activation of the slave:
    digitalWrite(port,LOW);

    // sending the data byte-to-byte:
    if (is_temp ){
      for (int i = 0 ; i < sizeof tempreture_value ; i++ ){
        SPI.transfer((char)tempreture_value[i]);
        Serial.print(tempreture_value[i]);
        delay(1);
      }
    }else{
      for (int i = 0 ; i < sizeof light_value ; i++ ){
        SPI.transfer(light_value[i]);
        Serial.print(light_value[i]);
        delay(1);
      }
    }

    // sending '\r' → end of the transmition
    SPI.transfer('\r');

    // deactivation of the slave:
    digitalWrite(port,HIGH);

    // go to the next line
    Serial.println();
}
```

```
void setup()
  {
    pinMode(PIN_LIGHT, INPUT);
    pinMode(PIN_TEMPRETURE, INPUT);
    pinMode(ss1,OUTPUT);
    pinMode(ss2,OUTPUT);
    Serial.begin(9600);

    // active-low slave selects
    digitalWrite(ss1,HIGH);
    digitalWrite(ss2,HIGH);
    SPI.begin();
    SPI.setClockDivider(SPI_CLOCK_DIV8);
    delay(100);
    Serial.println("< MASTER >");
   }

void loop(){

  // get analog values from the outside
  int lght = analogRead(PIN_LIGHT);
  int tmp = analogRead(PIN_TEMPRETURE);

  // map inputs into percentages
  lght = map(lght, 0, 200, 0, 100);
  tmp = map(tmp, 0, 400, 0, 100);

  // convert them into Strings
  light_value = (String)lght + '%';
  tempreture_value = (String)tmp + '%';

  transferStringToSlave(ss1);
  delay(500);
  transferStringToSlave(ss2);
  delay(500);

  Serial.println("---------------");
}
```

- **کد** (slave)

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

نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

ارسال میزان دما و نور با SPI

استاد: **آقای معصوم‌زاده**
درس: **آزمایشگاه ریزپردازنده و زبان اسمبلی**

```
char buff [100];
volatile byte index;
volatile boolean process_is_done;
volatile boolean start;

void setup(){
  Serial.begin(9600);
  SPCR |= _BV(SPE);
  index = 0;
  process_is_done = false;
  start = true;

  //allow SPI interrupts
  SPI.attachInterrupt();
  Serial.println("< SLAVE >");
}

void loop(void){

  // character '\r' is recieved → the process is done
  if (process_is_done) {
    process_is_done = false; //reset the process
    index= 0; //reset the index to zero
  }

  delay(5);
}

// Interrupt Service Routine
ISR (SPI_STC_vect) {
  // new input data recieved → the process is not done yet
  process_is_done = false;

  // print the begenning of each line
  if (start){
    Serial.print("recieved data: ");
    start = false;
  }

  // read byte from SPI Data Register in each interrupt
  byte c= SPDR;
  if (index < sizeof buff) {

    // save data in the array buff
```

```
    buff [index++] = c;

    // check if the data is finished or not
    if (c != '\r')
      Serial.print((char)c); // print each recieved character

    // done...
    else {
      process_is_done = true;
      start = true;
      Serial.println();
    }
  }
}
```