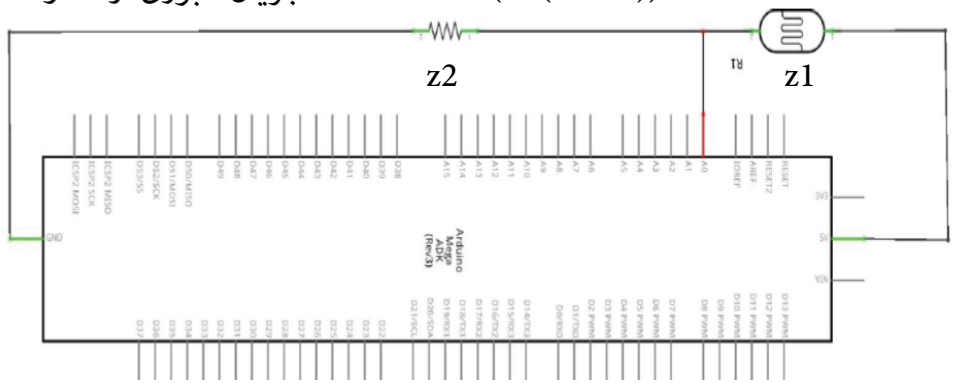
پیش‌گزارش آزمایشگاه _ آز8: آشنایی با پروتکل SPI و تحلیل موج خروجی مستر، راه اندازی حسگر دما و نور

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

فتوسل یک قطعه مقاومتی است که مقاومت آن با تابش نور تغییر می کند و با آن نسبت عکس دارد. از میان دو مقاومت که یکی فتوسل و دیگری مقاومت معمولی است می توان vout را از پورت آنالوگ خواند که عددی بین 0 تا 1023 است و ولتاژ را بدست می آوریم. جریانی که از پایه vout خارج می شود 0 است(جریان نمی کشد) پس جریان بین مقاومت ها تقسیم می شود. با استفاده از خواندن از پورت آنالوگ و محاسبات تقسیم ولتاژ می توان روابط میان نور و مقاومت فتوسل را دریافت.

جریان عبوری از مقاومت ها=Vin/(z1 + z2)

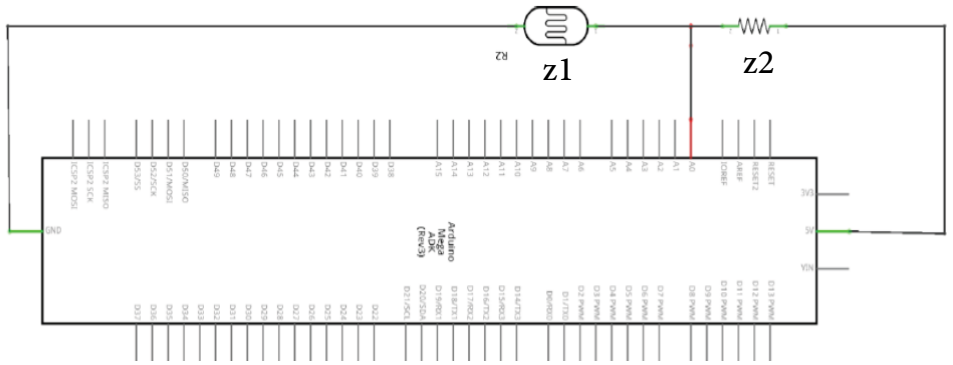
جریان عبوری از مقاومت = 0+IR =z2*(vi/(z1+z2))



در صورت افزایش نور مقاومت فتوسل کم شده ولتاژ خروجی افزایش می یابد
در صورت کاهش نور مقاومت فتوسل زیاد شده و مخرج کسر افزایش یافته و ولتاژ خروخی کاهش می یابد

جریان عبوری از مقاومت ها=Vin/(z1 + z2)

جریان عبوری از فتوسل = 0+IR =z1*(vi/(z1+z2))



در صورت افزایش نور ولتاژ مقاومت فتوسل کاهش یافته و ولتاژ خروجی کاهش می یابد
در صورت کاهش نور مقاومت فتوسل افزایش یافته ولتاژ خروخی افزایش می یابد

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

سنسور میزان دما lm35محیط برحسب درجه سانتی گراد است. این سنسور دارای سه پایه زمین، 5ولت و خروجی است که به یک پورت آنالوگ مپ شده و در آنجا با نمونه برداری و استفاده از قواعد ریمان تبدیل به مقادیر دیجیتال میشود.

- در مورد پایه هایMOSI، MISO، SCLKدر آردوینو Megaتحقیق کنید. پایه ی پیشفرض برایSS کدام پایه است؟ مشاهده آن می توانید به محل نصب آردوینو رفته، مسیر زیر را دنبال نمایید و در انتها فایل داخل پوشه را باز نمایید:
-> hardware -> arduino -> avr -> variants -> mega

پین های mosi,miso,sclk پین های ثابتی روی برد هستند چرا که سخت افزار خاص خود را دارند ولی پین ss می تواند پین دیجیتال دلخواه باشد ما برای آسانی کار پینی روی برد و در کتابخانه برای آن در نظر گرفته شده است.

در sclk در برد آردینو در صورتی که polarity(CPOL)(تعیین کننده مقدار اولیه کلاک) مقدار 0 یا 1 داده شده باشد، با phase (CPHA) تعیین می کنیم که کلاک در لبه پایین رونده تغییر دیتا دهد یا در لبه بالا رونده.
برای پایان ارتباط یا شروع آن کلاک باید در حالت پلاریته خود باشد.

پایین رونده	CPOL=0 CPHA=1	بالارونده	CPOL=0 CPHA=0

یایه پیش فرض برای ss یک پایه output پین است که در حالت pull up (high) قرار دارد. زمانی که این پین را low میکند تا ارتباط را شروع کند نیاز به یک دیلی دارد تا پیام به مقصد برسد.

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

static const uint8_t SS   = PIN_SPI_SS;
static const uint8_t MOSI = PIN_SPI_MOSI;
static const uint8_t MISO = PIN_SPI_MISO;
static const uint8_t SCK  = PIN_SPI_SCK;
```

- در مورد نحوه ی انتخاب بورد **slave**توسط **ss** تحقیق نموده و نحوه پیاده سازی برنامه را برای این که بورد مرکزی بتواند به ترتیب و در هر ثانیه برای یکی از اسلیو ها داده ارسال کند، شرح دهید. (برای این کار بهتر است نمونه کد هایی که برای ارتباط بین دو آردواینو از طریق پروتکل spi در اینترنت موجود است را بررسی نمایید.)

هر slave داری 4 سیم است که دو تای آن انتقال داده، یکی کلاک و دیگری ss (slave select)که مشخصه خود اسلیو برای بیدار شدن است. نحوه انتخاب بودر اسلیو مدنظر بدین صورت است که پین ss که یک پین active low است(به این معنی که 1 و غیر فعال است تا زمانی که 0 شده و فعال شود) با 0 شدن بیدار می شود و شروع به گوش دادن به مستر می کند و زمانی که اینتراپتی رخ دهد داده را در یک آرایه در رجیستر بیت به بیت ذخیره کرده و آن را پردازش می کند.

```
#include <SPI.h>
//--------------------------------------------------------------------------------
#define MESSAGE   "Hello, world!\r"
//--------------------------------------------------------------------------------
void setup (void) {
   Serial.begin(9600); //set baud rate to 115200 for usart
   digitalWrite(SS, HIGH); // disable Slave Select
   //SPI.beginTransaction(SPISettings(14000000, MSBFIRST, SPI_MODE0));
   SPI.begin();
   SPI.setClockDivider(SPI_CLOCK_DIV8);//divide the clock by 8
}

void loop (void) {
   char c;
   digitalWrite(SS, LOW); // enable Slave Select
   // send test string
   for (const char * p = MESSAGE ; c = *p; p++)
   {
      SPI.transfer (c);
      Serial.print(c);
   }
   digitalWrite(SS, HIGH); // disable Slave Select
   delay(2000);
}
```

```
void setup (void) {
   Serial.begin (9600);
   pinMode(MISO, OUTPUT); // have to send on master in so it set as output
   SPCR |= _BV(SPE); // turn on SPI in slave mode
   indx = 0; // buffer empty
   process = false;
   SPI.attachInterrupt(); // turn on interrupt
}

ISR (SPI_STC_vect) // SPI interrupt routine
{
   byte c = SPDR; // read byte from SPI Data Register
   if (indx < sizeof buff) {
      buff [indx++] = c; // save data in the next index in the array buff
      if (c == '\r') //check for the end of the word
      process = true;
   }
}

void loop (void) {
   if (process) {
      process = false; //reset the process
      Serial.println (buff); //print the array on serial monitor
      indx= 0; //reset button to zero
   }
}
```

- مقدار کلاک توسط **Master** تعیین می شود یا **Slave** ؟

مقدار کلاک توسط مستر تعیین می شود و دلیل نام گذاری آن هم همین است. اسلیو ها با توجه به کلاکی از جانب مستر به آنها فرستاده می شوند واکنش نشان داده و تابع هستند و مستر با تعیین کلاک مشخص می کند که سرعت ارتباط چقدر باشد و داده چه زمانی ارسال یا دریافت شود

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

توابع کتابخانه SPI :

transfer(address)

Syntax

```
receivedVal = SPI.transfer(val)
receivedVal16 = SPI.transfer16(val16)
SPI.transfer(buffer, size)
```

براساس ارسال و دریافت همزمان انجام SPI انتقال receiveVal می شود: داده های دریافت شده در بازگردانده می شود. در صورت (receiveVal16 یا) انتقال بافر ، داده های دریافت شده در بافر در محل ذخیره می شوند (داده های قدیمی با داده های دریافت شده جایگزین می شوند)

---

attachInrterrupt()

Syntax

```
SPI.attachInrterrrupt()
```

برای ایجاد اینتراپت زمانی که ورودی داده ای وارد می شود و ذخیره یک باید در رجیستر مربوط به خود در بخش سخت افزار آردینو

begin()

Syntax

```
SPI.begin()
```

با تنظیم SCK ، MOSIو SS بر روی خروجی ها ، کشیدن SCK و MOSI به پایینlow و SS زیادhigh ، گذرگاه SPI را آغاز می کند .

---

setClockDriver()

Syntax

```
SPI.setClockDivider(divider)
```

تقسیم کننده ساعت SPI را نسبت به ساعت سیستم تنظیم می کندپیش فرض SPI_CLOCK_DIV4 است که ساعت SPI را روی یک چهارم فرکانس ساعت سیستم تنظیم می کند

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

از مشکلاتی که در اسلو ها وجود دارد که موجب ناکارایی آن می شود عدم وجود ack برای تایید ارتباط در انتقال داده است و در صورت نیاز به ack می توان با یک پرینت با اکو کردن مشابه آن را ساخت به طوری که بعد از یک بایت یا زمانی نباشد که داده گم شد است اما به خودی خود این بیت را ندارد. مسئله دیگر سیم ها بسیار و تاثیر الکترومغناطیس روی سیم ها موجب نویز پذیری آن می شود. برای رفع این مشکل نیاز است روی تک تک سیم ها یک روکش gnd تابانده شود تا اختلاف پتانسیلی که برد از روی سیم می خواند ثابت بماند. کارایی آن در زمانی است که می خواهیم در هزنیه صرف جویی کنیم و نیاز به ارتباط سریع در فاصله کوتاه داریم.

در ابتدا زمانی که تابع begin را صدا می زنیم برد به حالت مستر اینیشیال می شود. برای بردن آن به حالت مستر یکی از راه ها آسان آن است که پین ها از ابتدا خودمان تعریف کنیم.

SS pin -> Input & Pullup
MOSI pin -> Input
MISO pin -> Output
SCK pin -> Input
SPCR|=_BV(SPE); را در setup می نویسیم
Define ISR (SPI_STC_vect){…SPDR…}
Check for new data in loop()!

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

زمانیکه یک دیتا در ورودی رسیو می شود isr را با ورودی spi_stc_vect فراخوانی شده و وقفه ای در سیستم ایجا می کند. این تابه دارای رجیستر SPDR است که یک بایت را که در آن لحظه با اینتراپت خوانده و در بافر ورودی است را باز می گرداند. با یک آرایه از بایت به طور گلوبال در کد اسلیو می توان هر بار ورودی داده را با زیاد کردن i خواند و ذخیره کرد و در یک حلقه بررسی می کنیم که آیا آن i زیاد شده است با خیر.