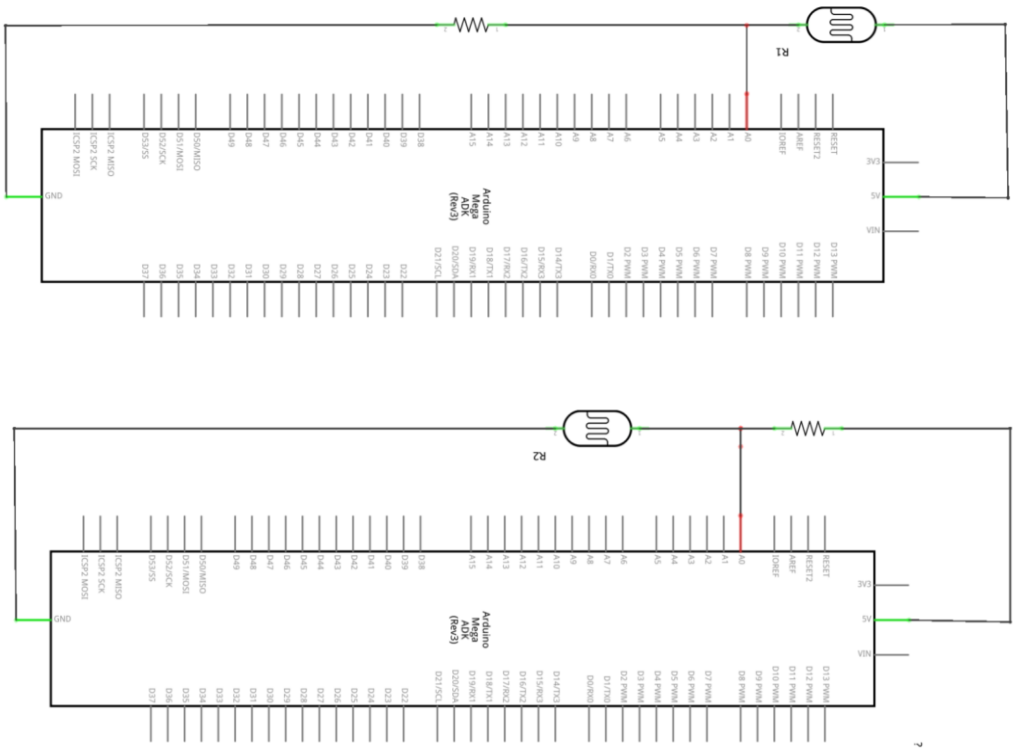
به نام خدا

3/12/2021

گزارش آزمایش شماره 8

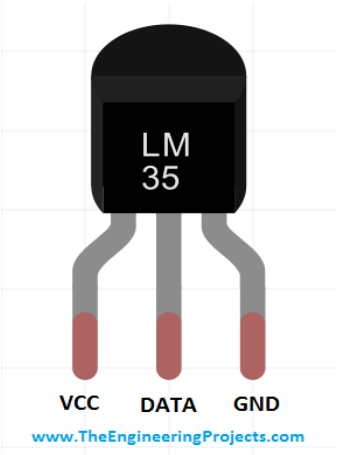
آزمایشگاه ریزپردازنده و زبان اسمبلی

محمد جواد زندیه , ابوالفضل بکیاسای کیوی
دانشگاه صنعتی امیرکبیر دانشکده مهندسی کامپیوتر



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

در این دو مدار ولتاژ به نسبت میزان مقاومت تقسیم میشود. از آنجا که با افزایش میزان نور، مقاومت LDR کاهش می یابد، در مدار اول ورودی آنالوگ افزایش یافته و در مدار دوم ورودی آنالوگ کاهش می یابد.



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

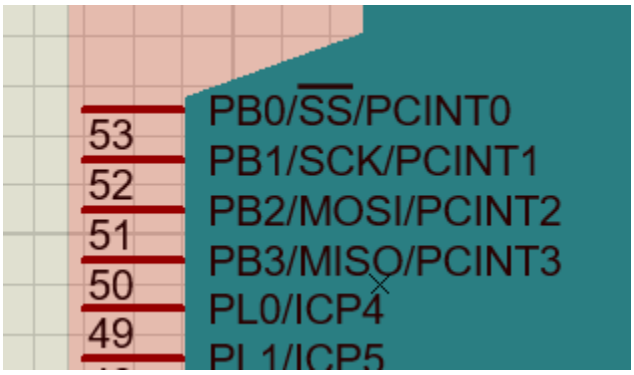
سنسور LM35، دارای سه پایه است. دو پایه ی کناری برای ground و power بوده و پایه وسط خروجی ولتاژ آنالوگ سنسور میباشد. این سنسور بر مبنای درجه سلسیوس کار میکند و به ازای هر درجه سانتیگراد، 10mV ولتاژ بیشتری را خروجی میدهد. پس با ضرب کردن ولتاژ خروجی آن در 100 میتوان به درجه سانتیگراد دست یافت.

پرسش: آیا در پروتکل SPI امکان حضور چند master وجود دارد ؟ چه پروتکلی این امکان را به ما می دهد؟

خیر در پروتکل SPI امکان حضور چند Master را نداریم، تنها یک master و چندین slave میتوان داشت. ولی در پروتکل I2C میتوان چندین Master و چندین Slave داشت، و محدودیت آن این است که در هر زمان ارتباط بین یک Master و یک Slave میتواند برقرار شود.

پرسش: در رابطه با ارتباط duplex-full تحقیق کنید. آیا پروتکل SPI از این امکان بهره مند است؟

ارتباط full duplex، ارتباطی است که داده میتواند به صورت همزمان به دو جهت منتقل شود. در پروتوکل SPI نیز ما برای انتقال داده از master به slave یک سیم داریم و برای انتقال از slave به master هم یک سیم، پس انتقال دو طرفه امکان پذیر بوده و full duplex میباشد.



پرسش: در مورد پایه های SCLK، MISO،MOSI در آردوینو Mega تحقیق کنید. پایه ی پیشفرض برای SS کدام پایه است؟

پایه پیشفرض برای MISO = 50، برای MOSI = 51، برای SCLK = 52 و برای SS = 53 میباشد.
محتوای فایل پین های پیشفرض:

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

پرسش : در مورد نحوه ی انتخاب برد Slave توسط SS تحقیق نموده و نحوه پیاده سازی برنامه را برای اینکه برد مرکزی بتواند به ترتیب و در هر ثانیه برای یکی از بردهای Slave داده ارسال کند، شرح دهید.

برای هر slave روی برد master یک پین SS در نظر گرفته میشود. هرگاه master بخواهد به هر کدارم از slave ها داده منتقل کند، پین SS مربوط به آن slave را active میکند (active low) و شروع به فرستادن داده می نماید. برای ارسال داده در هر ثانیه به یک slave، کافیست در هر ثانیه پین SS آن slave را LOW کنیم.

پرسش : مقدار کلاک توسط Master تعیین میشود یا Slave ؟

برد master همواره کلاک همه slave ها را تعیین میکند تا sync باشند.

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

begin()

پایه های SCK و MOSI و SS را برای ارتباط SPI به عنوان خروجی مشخص میکند و pullup و pulldown مورد نیاز را انجام میدهد.

setClockDivider()

مقدار کلاک SPI را بر حسب کلاک اصلی برد مشخص میکند. مقادیر 2,4,8,16,32,64,128 را میتواند به عنوان ورودی بگیرد. (مقدار اولیه آن SPI_CLOCK_DIV4 یا همان 4 است). اگر مقدار x به آن ورودی داده شود فرکانس کلاک SPI را برابر 1/x ضرب در کلاک اصلی برد قرار میدهد. (برای برد 16MHz برابر 4MHz قرار میدهد.)

transfer()

یک مقدار را send کرده و به طور همزمان یک مقدار را دریافت کرده و بر میگرداند.

attachInterrupt()

مشخص میکند که پروتکل SPI باید با وقفه انجام شود، یعنی وقتی یک slave سلکت میشود و داده ای به آن ارسال میشود، برایش یک وقفه بوجود آمده تا بتواند خود را برای ارتباط آماده کند و داده را دریافت کند. بعد از رخداد وقفه، ISR مشخص شده در کد Slave اجرا میشود.

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

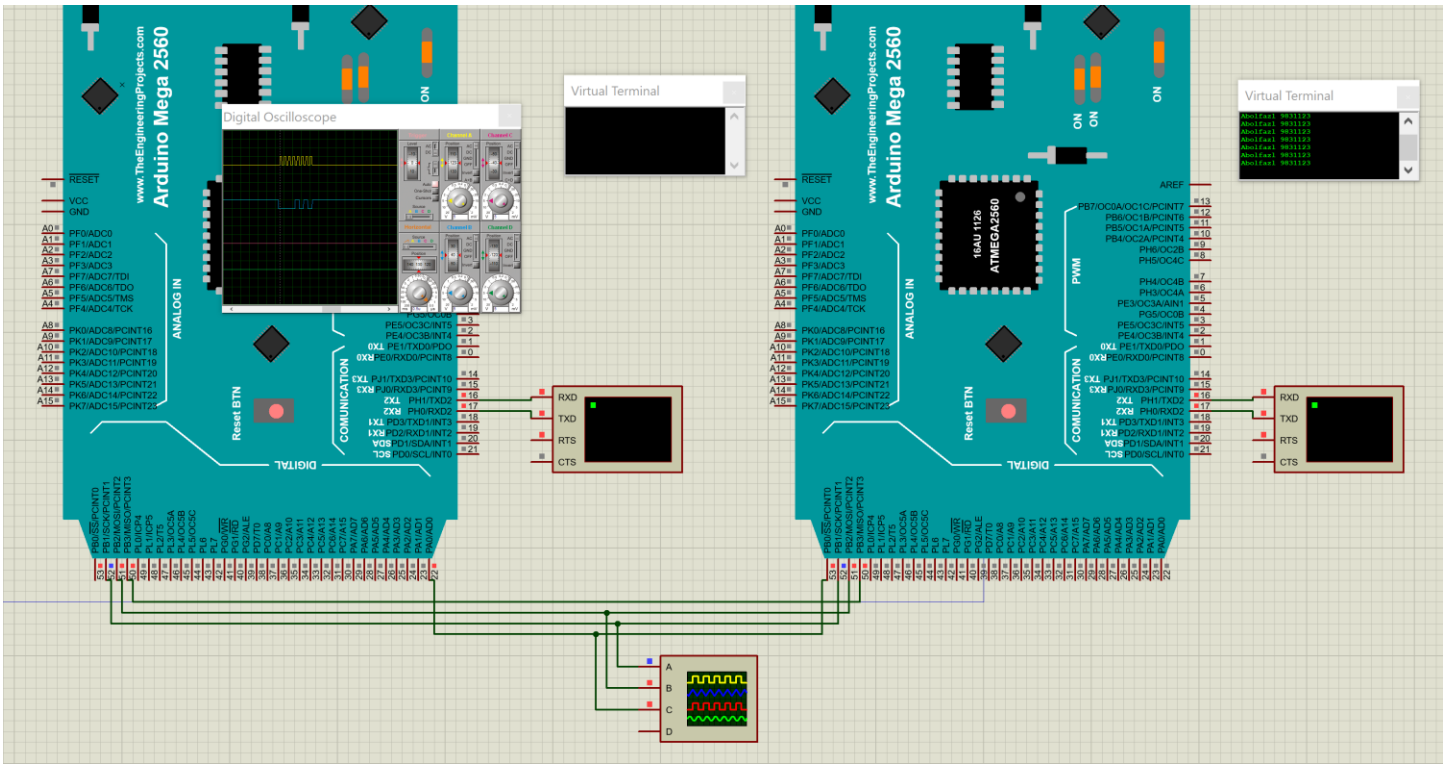
SPCR |= _BV(SPE);

این دستور بیت مشخص کردن Slave بودن را در کنترل رجیستر فعال میکند. SPE جایگاه بیت مورد نظر میباشد و _BV(SPE) معادل (1 << SPE) بوده و یک بایت تمام صفر با تنها بیت SPE ام 1 را تحویل میدهد که این مقدار را با SPCR، OR میکنیم و در SPCR قرار میدهیم تا بیت Slave یک شود.

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

تابع ISR وقتی اجرا میشود که آن Slave سلکت شده باشد و یک بایت داده به آن ارسال شود. رجیستری با نام SPDR یا همان Data Register برای انتقال داده استفاده میشود (هم بایت دریافتی، هم بایت ارسالی)

Part1:




```
#include <SPI.h>

#define SS1 22

const String message = "Abolfazl 9831123\n";

volatile byte data_get, data_send;

void setup(){
  Serial2.begin(9600);
  pinMode(SS1, OUTPUT);
  SPI.begin();
  SPI.setClockDivider(SPI_CLOCK_DIV4);
  digitalWrite(SS1, HIGH);
}

void loop(){

  digitalWrite(SS1, LOW);
```

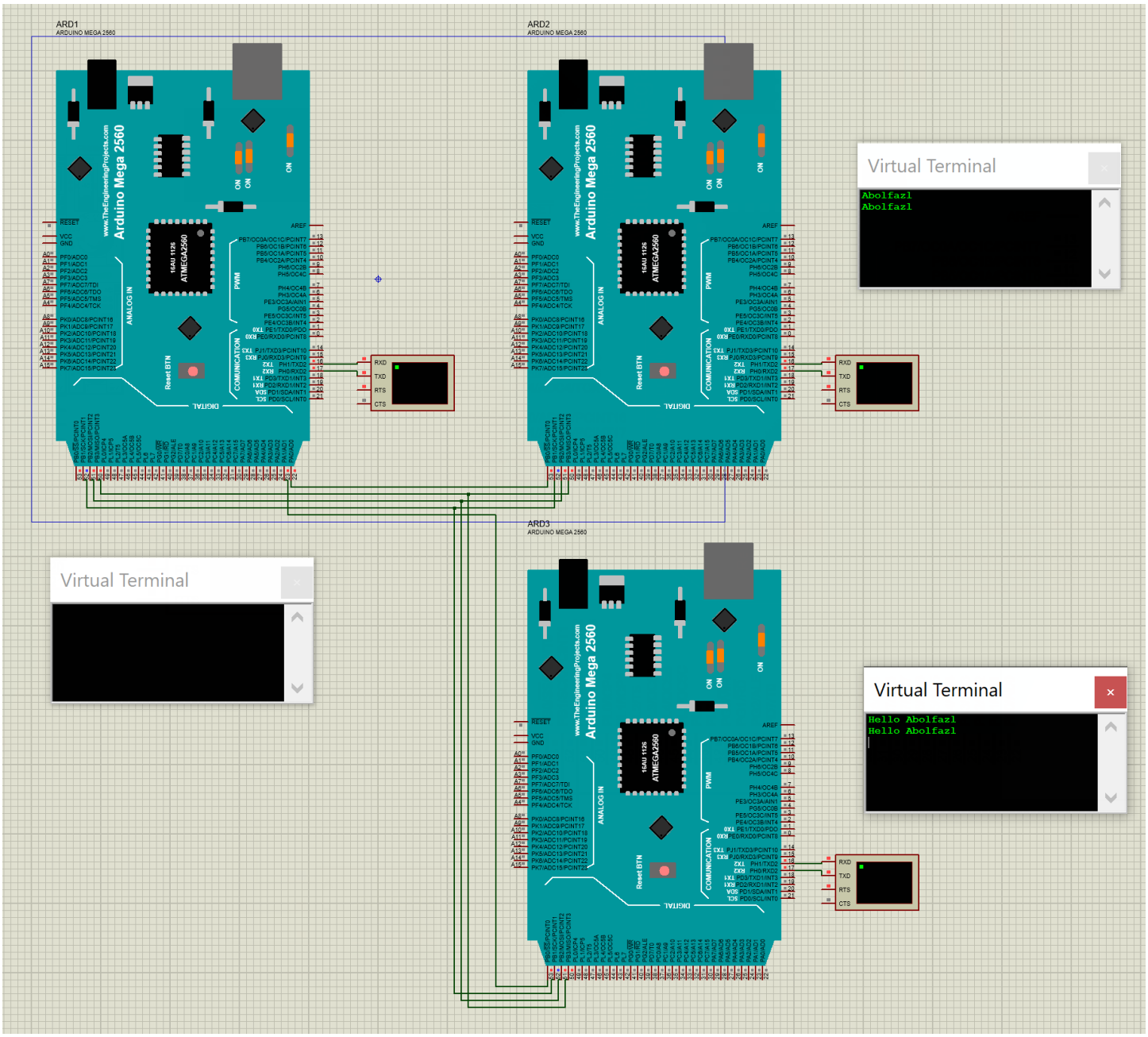
```
  for(int i=0;i<message.length();i++) {
    data_send = message.charAt(i);
    data_get = SPI.transfer(data_send);
    delayMicroseconds(20);
  }

  digitalWrite(SS1, HIGH);
  delay(1000);
}
```

Slave:

```
#include<SPI.h>

volatile boolean received;
volatile byte data_get, data_send;

void setup()
{
  Serial2.begin(9600);
  pinMode(MISO,OUTPUT);
  pinMode(SS,INPUT);

  SPCR |= _BV(SPE); // Setting the control register Slave bit (SPE is
the bit that specifies being Slave)    Same as (1 << SPE)
  received = false;

  SPI.attachInterrupt();
}

ISR (SPI_STC_vect)
{
  data_get = SPDR;  //  Saving Data Register into variable
  received = true;
```

```
}

char last = '\n', c;
String tmp = "";

void loop()
{
  if(received)
  {
    c = data_get;
    if(c!='\n')
      tmp += c;

    //SPDR = data_send; // Setting Data Register for sending data
    received = false;

    if(c == '\n') {
      Serial2.println(tmp);
      tmp = "";
    }
  }
}
```

Part2:



Master:

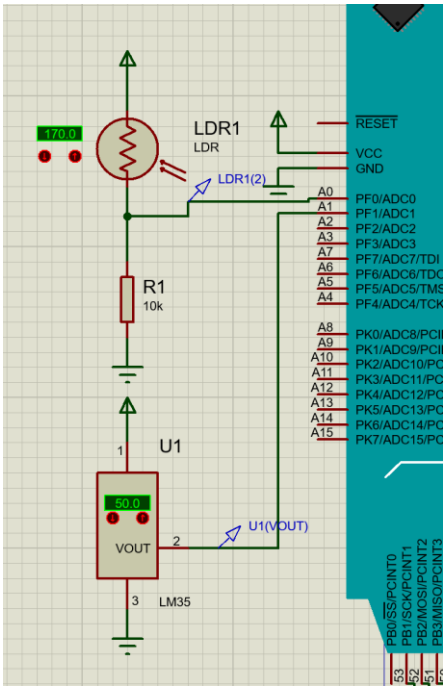
```
#include <SPI.h>

#define SS1 22
#define SS2 23

const String message1 = "Abolfazl\n";
const String message2 = "Hello Abolfazl\n";
```

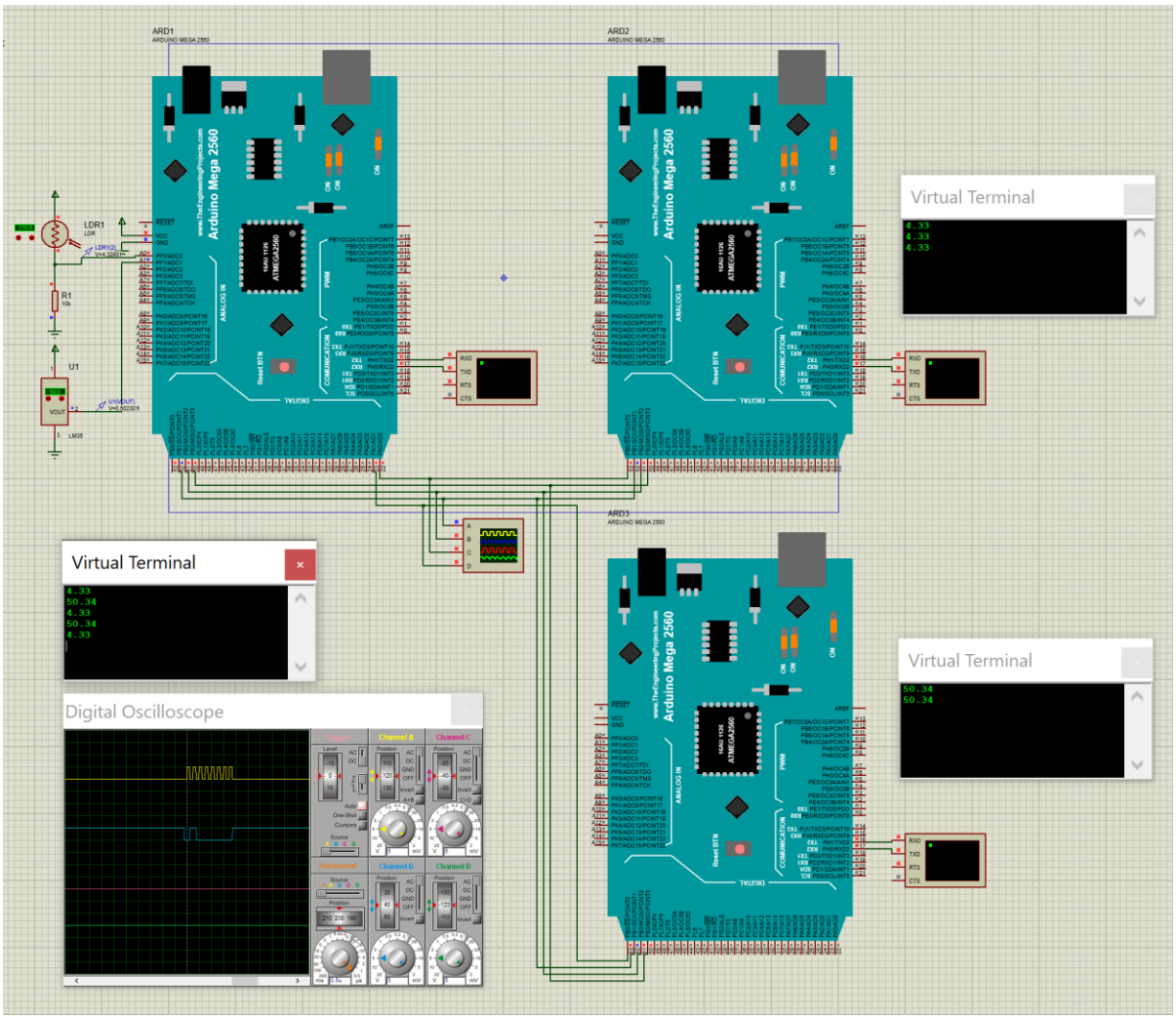
```
volatile byte data_get, data_send;

void setup(){
  Serial2.begin(9600);
  pinMode(SS1, OUTPUT);
  pinMode(SS2, OUTPUT);
  SPI.begin();
  SPI.setClockDivider(SPI_CLOCK_DIV4);
  digitalWrite(SS1, HIGH);
  digitalWrite(SS2, HIGH);
}

void loop(){

  digitalWrite(SS1, LOW);

  for(int i=0;i<message1.length();i++) {
    data_send = message1.charAt(i);
    data_get = SPI.transfer(data_send);
    delayMicroseconds(30);
  }

  digitalWrite(SS1, HIGH);
  delay(1000);


  digitalWrite(SS2, LOW);

  for(int i=0;i<message2.length();i++) {
    data_send = message2.charAt(i);
    data_get = SPI.transfer(data_send);
    delayMicroseconds(30);
  }
```

```
  digitalWrite(SS2, HIGH);
  delay(1000);
}


Slave:

#include<SPI.h>

volatile boolean received;
volatile byte data_get, data_send;

void setup()
{
  Serial2.begin(9600);
  pinMode(MISO,OUTPUT);
  pinMode(SS,INPUT);

  SPCR |= _BV(SPE); // Setting the control register Slave bit (SPE is
the bit that specifies being Slave)    Same as (1 << SPE)
  received = false;

  SPI.attachInterrupt();
}

ISR (SPI_STC_vect)
{
  data_get = SPDR;  //  Saving Data Register into variable
  received = true;
}

char last = '\n', c;
String tmp = "";

void loop()
{
```

```
  if(received)
  {
    c = data_get;
    if(c!='\n')
      tmp += c;

    //SPDR = data_send; // Setting Data Register for sending data
    received = false;

    if(c == '\n') {
      Serial2.println(tmp);
      tmp = "";
    }
  }
}
```

Part3:


170.0
LDR1
LDR
LDR1(2)
R1
10k
U1
50.0
VOUT
U1(VOUT)
LM35
RESET
VCC
GND
A0
A1
A2
A3
A7
A6
A5
A4
A8
A9
A10
A11
A12
A13
A14
A15
PF0/ADC0
PF1/ADC1
PF2/ADC2
PF3/ADC3
PF7/ADC7/TDI
PF6/ADC6/TDO
PF5/ADC5/TMS
PF4/ADC4/TCK
PK0/ADC8/PCI
PK1/ADC9/PCI
PK2/ADC10/PC
PK3/ADC11/PC
PK4/ADC12/PC
PK5/ADC13/PC
PK6/ADC14/PC
PK7/ADC15/PC
PB0/SS/PCINT0
PB1/SCK/PCINT1
PB2/MOSI/PCINT2
PB3/MISO/PCINT3
53
52
51


ARD1
ARDUINO MEGA 2560
ARD2
ARDUINO MEGA 2560
ARD3
ARDUINO MEGA 2560
Arduino Mega 2560
LDR1
R1
U1
LM35
Virtual Terminal
4.33
4.33
4.33
Virtual Terminal
4.33
50.34
4.33
50.34
4.33
Virtual Terminal
50.34
50.34
Digital Oscilloscope

Master:

```
#include <SPI.h>

#define LDR A0
#define LM35 A1

#define SS1 22
#define SS2 23

volatile byte data_get, data_send;

void setup(){
  Serial2.begin(9600);
  pinMode(SS1, OUTPUT);
  pinMode(SS2, OUTPUT);
  SPI.begin();
  SPI.setClockDivider(SPI_CLOCK_DIV4);
  digitalWrite(SS1, HIGH);
  digitalWrite(SS2, HIGH);
}

float fdata;

void loop(){

  int light_read = analogRead(LDR);
  float light_v = 5 * light_read / 1023.0;
  //float RLDR = (10000.0 * (5 - Vout))/Vout;
  //float lux = 500 / RLDR;
  Serial2.println(light_v);

  fdata = light_v;
  digitalWrite(SS1, LOW);

  for(int i=0;i<sizeof(float);i++) {
```

```
    data_send = *(((byte*)(&fdata))+i);
    data_get = SPI.transfer(data_send);
    delayMicroseconds(30);
  }

  digitalWrite(SS1, HIGH);
  delay(1000);



  int temp_read = analogRead(LM35);
  float temp_v = 5 * temp_read / 1023.0;
  float temp = 100 * temp_v;
  Serial2.println(temp);

  fdata = temp;
  digitalWrite(SS2, LOW);

  for(int i=0;i<sizeof(float);i++) {
    data_send = *(((byte*)(&fdata))+i);
    data_get = SPI.transfer(data_send);
    delayMicroseconds(30);
  }

  digitalWrite(SS2, HIGH);
  delay(1000);
}
```

Slave:

```
#include<SPI.h>

volatile boolean received;
volatile byte data_get, data_send;
```

```
void setup()
{
  Serial2.begin(9600);
  pinMode(MISO,OUTPUT);
  pinMode(SS,INPUT);

  SPCR |= _BV(SPE); // Setting the control register Slave bit (SPE is
the bit that specifies being Slave)    Same as (1 << SPE)
  received = false;

  SPI.attachInterrupt();
}

ISR (SPI_STC_vect)
{
  data_get = SPDR;  //  Saving Data Register into variable
  received = true;
}

int pos = 0;
float fdata = 0;

void loop()
{
  if(received)
  {
    *(((byte*)(&fdata))+pos) = data_get;
    pos = (pos + 1) % sizeof(float);
    received = false;
    if(pos == 0)
      Serial2.println(fdata);
  }
}
```