باسمه تعالی

دانشکده مهندسی مکانیک

پردیس دانشکده‌های فنی

دانشگاه تهران

گزارش تکالیف

جلسه دوم

درس سیستم‌های اندازه‌گیری کارشناسی

دکتر صدیقی

مهدی عبداله چالکی (۸۱۰۶۹۶۲۶۸)

نیم‌سال دوم

سال تحصیلی ۰۰-۹۹

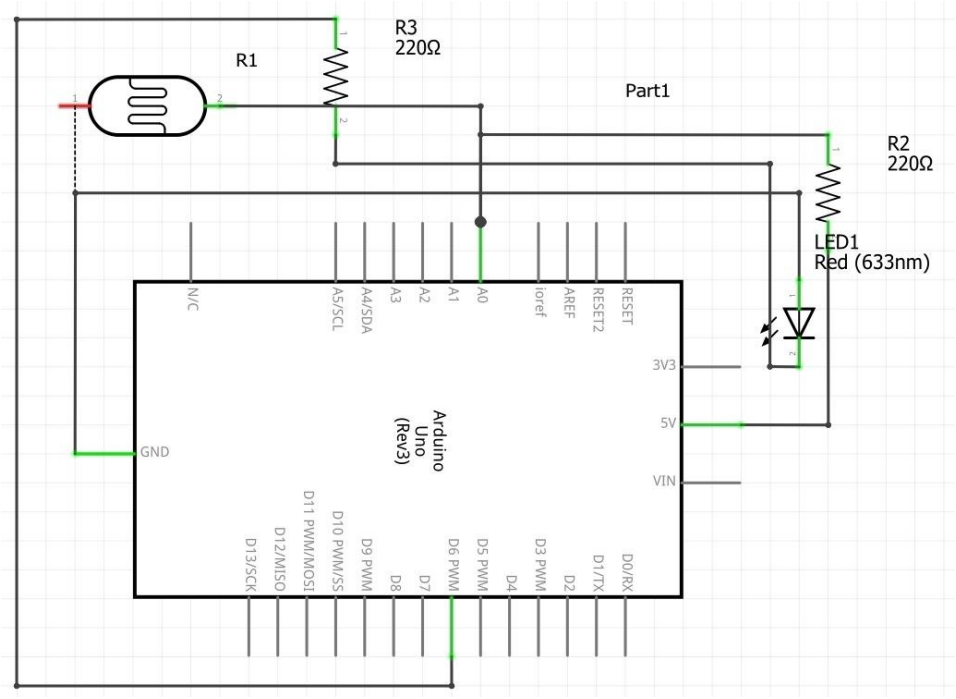
۱- تکلیف اول

در این تمرین، باید دو مدار سری اول تمرین‌های آزمایشگاه‌را با یکدیگر ترکیب کنیم. بدین صورت که یک دیود نوری هر یک ثانیه یک بار چشک بزند و همزمان، یک سنسور فوتوسل نور محیط را دریافت کرده و متناسب با آن، نور دیود نوری دیگری را کنترل کند؛ اگر نور محیط کم بود، نور آن دیود نوری بیشتر شده و اگر نور محیط زیاد شد، نور دیود نوری کم شود.

برای دیود نوری چشمک‌زن، از خود آردوینو کمک می‌گیریم و از دیودنوری موجود در آن استفاده می‌کنیم. تایمر ۱ را با دوره‌ی تناوب ۱ ثانیه راه‌اندازی کرده و هر ثانیه خاموش و روشن می‌کنیم. برای اینکه این بخش برنامه دقیقا هر ثانیه اجرا شود و به اجرای بقیه‌ی بخش‌ها کاری نداشته باشد، از دستور attachInterrupt استفاده می‌کنیم.

برای بخش LDR نیز همانند تمرین گذشته، ابتدا با داده برداری در هر ۱۰ میلی ثانیه و میانگین‌گیری روی هر بیست عدد داده، نویز را کاهش می‌دهیم. تابع درون‌یابی تعریف می‌کنیم و به ازای بازه‌ای که نور محیط در آن قرار دارد، مقدار مناسب نور محیط را محاسبه می‌کنیم. محاسبات مقاومت و ولتاژ همانند تمرین قبلی است و نکته‌ی جدیدی ندارد.

مدار شماتیک این تمرین، به صورت زیر است:

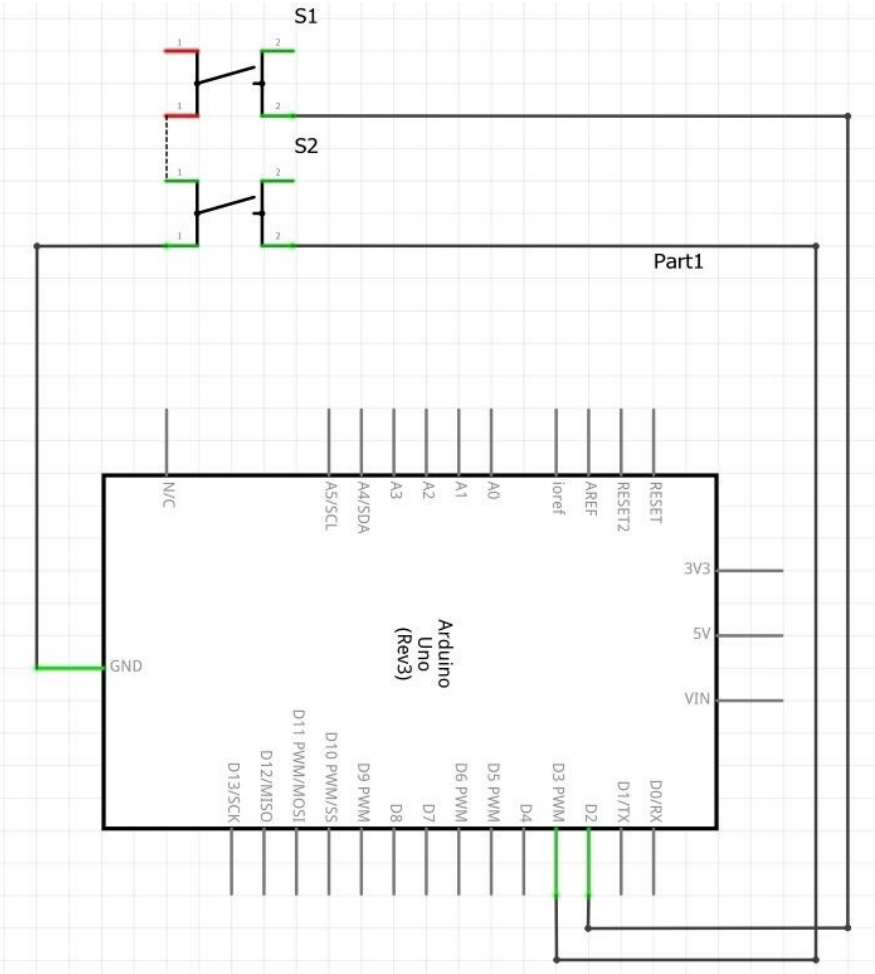


شکل ۱: شماتیک مدار تمرین اول

۲- تکلیف دوم

در تمرین دوم، قصد داریم با کمک دو عدد کلید فشاری، یک کورنومتر بسازیم. یک دکمه برای شروع و توقف زمان اندازه‌گیری شده و دکمه‌ی دیگر برای شروع مجدد زمان از صفر استفاده می‌شوند. همچنین یک دیود نوری - بدون توجه به عملکرد کورنومتر - هر یک ثانیه یک بار چشمک می‌زند.

شکل شماتیک این مدار به صورت زیر است:



شکل ۲: شماتیک مدار تمرین دوم

ابتدا برای دیود نوری چشمک زن، مشابه بخش قبلی عمل کرده و از دیود نوری خود آردوینو کمک می‌گیریم و با استفاده از کتابخانه‌ی TimerOne، و راه اندازی تایمر ۱ با دوره‌ی تناوب یک ثانیه، این عمل صورت می‌گیرد.

برای بخش کرنومتر، از مفهوم پنجره‌های زمانی استفاده می‌کنیم. حرف s شروع یک پنجره‌ی زمانی، حرف t برای حاصل تفریق زمان فعلی از زمان شروع شدن یک پنجره و حرف f نیز برای زمان تجمعی (جمع زمان پنجره‌های قبلی با زمان سپری شده در پنجره‌ی فعلی) به کار می‌رود. پس از هر بار فشردن دکمه ریست، این

زمانی تجمعی صفر می‌شود و کار ادامه می‌یابد. در صورت فشردن دکمه شروع، ابتدا زمان فعلی در متغیر s ذخیره می‌شود تا پنجره شروع شود، متغیر t در حلقه اصلی، حاصل تفریق زمان شروع پنجره از زمان حال را محاسبه می‌کند. پس از فشردن دکمه توقف، این زمان (که برابر است با پنجره‌ی بسته شده) با زمان پنجره‌های قبلی جمع می‌شود و در خروجی چاپ می‌گردد.

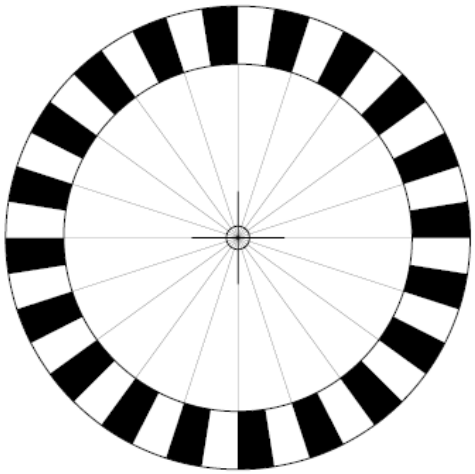
همچنین برای خنثی کردن اثر برخورد فلزهای سنسور با یکدیگر در مدت زمان کوتاه و مختل شدن عملکرد دکمه‌های فشاری، از مفهوم debouncing استفاده می‌شود. بدین صورت که اگر زمان میان دو تغییر در وضعیت یک دکمه کمتر از زمان تعیین شده (در اینجا ۲۰۰ میلی‌ثانیه) بود، بار دوم را در نظر نمی‌گیریم.

در نهایت نیز زمان اندازه‌گیری شده که بر حسب میلی‌ثانیه است را با کمک مفاهیم تقسیم صحیح و باقی مانده، به دقیقه و ثانیه و میلی‌ثانیه تبدیل می‌کنیم.

۳- تمرین سوم

در تمرین سوم، هدف ما ساختن یک انکودر و گزارش جابجایی و سرعت لحظه‌ای آن با استفاده از یک سنسور فرستنده – دریافت کننده‌ی نوری است.

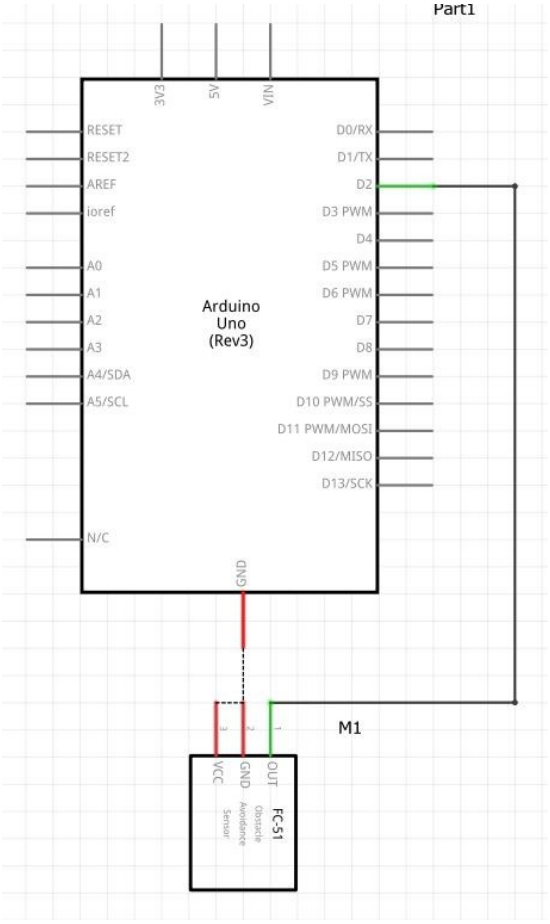
در گام اول، نیاز به طرح انکودر است تا بر روی کاغذ چاپ شود. برای این کار از نرم‌افزار LaTex کمک گرفته شده است:



شکل ۳: انکودر طراحی شده در LaTex

این انکودر دارای ۴۰ درجه است و در نتیجه هر ۹ درجه را بر روی دایره می‌تواند تشخیص دهد.

حال به سراغ مدار می‌رویم. سنسور دارای سه پایه را وصل می‌کنیم.



شکل ۴: شماتیک مدار تمرین سوم

این سنسور به عنوان ورودی تعریف می‌شود. برای اینکه بتوانیم هر بار سنسور از لبه‌ی شکافی عبور کرد تغییرات را متوجه شویم، attachInterrupt را تنظیم می‌کنیم تا با تغییر سطح ولتاژ کار کند. از دیود نوری خود آردوینو کمک گرفته و به ازای هربار عبور شکاف از جلو سنسور، ۲ میلی‌ثانیه آن را روشن و خاموش می‌کنیم.

برای اندازه‌گیری جابجایی، یک شمارنده تعریف کرده تا هر عبور شکاف را ثبت کند. می‌دانیم اختلاف هر شکاف با بعدی، ۹ درجه است. پس تعداد شمارنده در ۹ ضرب شده و مقدار نهایی به درجه بیان می‌شود.

برای اندازه‌گیری سرعت زاویه‌ای، زمان شکاف جدید ثبت شده، و قبلی از آن کم می‌شود. با تقسیم ۹ درجه بر این زمان، سرعت لحظه‌ای (که تقریبی است) بدست می‌آید.

۴- پاسخ به سوالات

سوال۱: چرا برای سرعت‌های پایین‌تر، از تعداد سوراخ بیشتری استفاده می‌شود و برای سرعت بالاتر، از دیسک با سوراخ کمتر؟

پاسخ: سنسور برای آنکه بتواند به مقدار نهایی خود برسد، به زمان نیاز دارد. ثابت زمانی سنسور هر قدر هم اندک باشد، در سرعت‌های بسیار بالا اگر فاصله شکاف‌ها و در نتیجه زمان سنسور برای تغییر دادن ولتاژ خروجی خود کم باشد، عملکرد آن مختل می‌شود. اما در سرعت‌های پایین چنین مشکلی وجود ندارد و سنسور زمان کافی دارد تا به پایخ نهایی خود برسد.

سوال۲: در سرعت‌های بسیار بالا، چه اتفاقی خواهد افتاد؟

پاسخ: در سرعت‌های بسیار بالا، سنسور وقتی با تابع پله تحریک می‌شود، بسته به ثابت زمانی خود زمان نیاز دارد تا سطح خروجی خود را به مقدار نهایی برساند. اگر زمان کافی نداشته باشد (موضوع مورد بحث)، آنگاه نمی‌تواند به مقدار نهایی خود برسد و عملکردش مختل خواهد شد و در نتیجه مقدار زاویه‌ی درستی زا گزارش نخواهد کرد (احتمالا تعدادی سوراخ را رد می‌کند).

۵- ضمائم

تمرین اول:

```
#include <TimerOne.h>
volatile bool ledState=HIGH;      // defining ledState boolean to keep
the record of LED's state


/********* LDR *********/
#define NUM_SAMPLES 20            // Number of samples used for
averaging


int LDR = A0;            // Pin A0 has an LDR attached to it.
int led_pin = 6;         // Digital pin 6 has an LED attached to it.


int sum = 0;                      // sum of samples taken
unsigned char sample_count = 0;   // current sample number
float V_out = 0.0;                // calculated V_out
float Resistance = 0.0;           // calculated Resistance
float Illumination = 0.0;         // calculated Illumination
float LED;                        // output to be sent to the LED


/********* Setup *********/
void setup() {


  // using built-in LED
  pinMode(LED_BUILTIN, OUTPUT);


  // initialize timer1 with a period of 1s
  Timer1.initialize(1000000);


  // blinkLED to run once overy seconds
  Timer1.attachInterrupt(blinkLED);
```

```
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);

  // Define led_pin as an output
  pinMode(led_pin, OUTPUT);
}



/********* Main Loop *********/
void loop() {

  // This loop is used for averaging and decreasing the noise
while (sample_count < NUM_SAMPLES) {
        sum += analogRead(LDR);
        sample_count++;
        delay(10);
}

// Map output voltage  to 0-5 scale
V_out = ((float)sum / (float)NUM_SAMPLES) * (5.0 / 1024.0);

// Calculate Resistance of the LDR
Resistance = (((float)V_out*220.0)/(5.0-V_out));

// Map output voltage to 0-255 scale to control LED light
LED = ((float)sum / (float)NUM_SAMPLES) * 255.0 / 1023.0;

analogWrite(led_pin, LED);
```

```
// Reset values
sample_count = 0;
sum = 0;



// Look up for the right category
if (Resistance<1.5e7 && Resistance>1e6) {
  Illumination = interp(1e6,1.5e7,0.1,0.001,Resistance);
  Serial.print(Illumination);
  Serial.println (" LUX");
}

else if (Resistance<1e6 && Resistance>7.5e4) {
  Illumination = interp(7.5e4,1e6,35,0.1,Resistance);
  Serial.print(Illumination);
  Serial.println (" LUX");
}

else if (Resistance<7.5e4 && Resistance>3e4) {
  Illumination = interp(3e4,7.5e4,72,35,Resistance);
  Serial.print(Illumination);
  Serial.println (" LUX");
}

else if (Resistance<3e4 && Resistance>1.7e4) {
  Illumination = interp(1.7e4,3e4,157,72,Resistance);
  Serial.print(Illumination);
  Serial.println (" LUX");
}
```

```
else if (Resistance<1.7e4 && Resistance>1.5e4) {
  Illumination = interp(1.5e4,1.7e4,200,157,Resistance);
  Serial.print(Illumination);
  Serial.println (" LUX");
}

else if (Resistance<1.5e4 && Resistance>3.8e3) {
  Illumination = interp(3.8e3,1.5e4,1380,200,Resistance);
  Serial.print(Illumination);
  Serial.println (" LUX");
}

else if (Resistance<3.8e3 && Resistance>2.8e3) {
  Illumination = interp(2.8e3,3.8e3,3700,1380,Resistance);
  Serial.print(Illumination);
  Serial.println (" LUX");
}

else if (Resistance<2.8e3 && Resistance>1.4e3) {
  Illumination = interp(1.4e3,2.8e3,7300,3700,Resistance);
  Serial.print(Illumination);
  Serial.println (" LUX");
}

else if (Resistance<1.4e3 && Resistance>47) {
  Illumination = interp(47,1.4e3,2.3e4,7300,Resistance);
  Serial.print(Illumination);
  Serial.println (" LUX");
}
```

```
else if (Resistance<47 && Resistance>10) {
  Illumination = interp(10,47,1e5,2.3e4,Resistance);
  Serial.print(Illumination);
  Serial.println (" LUX");
}

else {
  Illumination = 0;
  Serial.print(Illumination);
  Serial.println (" LUX");
}



}



/********* ISR *********/
void blinkLED() {
  digitalWrite(LED_BUILTIN, ledState);    // toggle LED
  ledState=!ledState;                     // keep the record of LED's
state
}



// Float interpolation function
float interp(float A_1, float A_2, float B_1, float B_2, float x){
  return (((x-A_1)/(A_2-A_1))*(B_2-B_1))+B_1;
}
```

تمرین دوم:

```
#include <TimerOne.h>

volatile bool ledState=HIGH;    // defining ledState boolean to keep
the record of LED's state


int start_stop = 2;             // digital pin 2 has a pushbutton
attached to it.

int reset_button = 3;           // digital pin 2 has a pushbutton
attached to it.

volatile bool state = false;    // defining state boolean as a global
variable

volatile long t = 0;            // defining t as long to keep the time

int debouncingTime = 200;       // defining debouncing time as
milliseconds

volatile long f = 0;            // cumulative time

volatile long d = 0;            // used for debouncing

volatile long s = 0;            // save starting point of the current
window

int second = 0;                 // save passed seconds

int milli = 0;                  // save passed miliseconds

int mins = 0;                   // save passed minutes


void setup() {


  // initialize serial communication at 9600 bits per second:

  Serial.begin(9600);

  // use built-in LED

  pinMode(LED_BUILTIN, OUTPUT);

  // initialize timer1 with a period of 1s

  Timer1.initialize(1000000);

  // blinkLED to run once overy seconds

  Timer1.attachInterrupt(blinkLED);
```

```
  // declaring start_stop pin as digital input
  pinMode (start_stop, INPUT_PULLUP);

  // declaring reset pin as digital input
  pinMode (reset_button, INPUT_PULLUP);

  // declaring start_stop pin as hardware interrupt
  attachInterrupt (digitalPinToInterrupt(start_stop), start_stopISR,
FALLING);

  // declaring reset pin as hardware interrupt
  attachInterrupt (digitalPinToInterrupt(reset_button), ResetISR,
FALLING);
}

void loop() {

  // calculate passed time in the currrent window
  t = millis() - s;

}

/****** start_stopISR ******/
void start_stopISR() {
  if (millis()-d > debouncingTime) {

    // toggling state boolean in case button is pushed
    state = !state;
```

```
    if (state) {

      // set new starting point each time the chronometer starts
      s = millis();
    }

  else if (state == false) {

      // sum up total passed time each time the chronometer stops
      f = f + t;

      // change from miliseconds to "mins:seconds:miliseconds"
      mins = (f/(60000))%60;
      second = (f/1000)%60;
      milli = f % 1000;

      // print elapsed time
      Serial.print(mins);
      Serial.print(" : ");
      Serial.print(second);
      Serial.print(" : ");
      Serial.println(milli);
    }

    // used foe debouncing
    d = millis();
  }
}
```

```
/****** ResetISR ******/
void ResetISR() {
  if (millis()-d>debouncingTime) {

    // set elapsed time to zero
    f = 0;

    // start new time window
    s = millis();

    // used foe debouncing
    d = millis();
  }
}



/****** blinkLED ******/
void blinkLED() {

  // toggle LED
  digitalWrite(LED_BUILTIN, ledState);

  // keep the record of LED's state
  ledState=!ledState;
}
```

تمرین سوم:

```
int sensor = 2;               // Pin 2 has a sensor attached to it.
int v = 0;                    // voltage of the sensor
int slots = 40;               // nuber of slots on the encoder
int counter = 0;              // counts number of changes
int x = 0;                    // position
int res;                      // resolution of the encoder
volatile long t = 0;          // prevoius time
volatile long d = 0;          // current time
int debouncingTime = 5;       // used to avoid multiple output
float omega;                  // angular velocity of the disk


/****** Setup ******/
void setup() {

  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);

  // Define sensor as an input
  pinMode(sensor, INPUT);

  // Define builtin LED as an output
  pinMode(LED_BUILTIN, OUTPUT);

  // declaring sensor pin as hardware interrupt
  attachInterrupt (digitalPinToInterrupt(sensor), sensorISR, CHANGE);

  // reolution of the encoder
  res = 360 / slots;
}
```

```
/****** Main Loop ******/

void loop() {
}

/****** sensorISR ******/

void sensorISR() {
  if (millis()-t > debouncingTime) {

    // count state changes
    counter ++;

    // calculate and print position
    x = counter * res;
    Serial.print("Position: ");
    Serial.print(x);
    Serial.println(" Deg");

    // get current time
    d = millis();

    // calculate and print angular velocity
    omega = 1000 * res / (d - t);
    Serial.print("Ang Velocity: ");
    Serial.print(omega);
    Serial.println(" Deg/s");

    // save time for the next loop
    t = millis();
```

```
    // blink LED for every pulse
    digitalWrite(LED_BUILTIN, 1);
    delay(2);
    digitalWrite(LED_BUILTIN, 0);
  }
}
```