به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

آزمایشگاه ریزپردازنده

گزارش‌کار آزمایش چهارم

استاد درس: مهندس معصوم زاده

مهدی رحمانی/ ۹۷۳۱۷۰۱

محمّد امین رضائی/ ۹۷۳۱۰۲۴

«پیش گزارش»

مفهوم PWM و استفاده‌های آن

PWM مخفف کلمه‌ی Pulse Width Modulation و به معنای "مدولاسیون پهنای پالس" است PWM . تکنیکی برای کنترل ولتاژ است. با استفاده از این تکنیک ما می توانیم ولتاژ را کنترل کنیم. در PWM ما باید زمان یک شدن و صفر شدن یک سیکل از موج مربعی را تعیین کنیم. هر چقدر مقدار زمان یک نسبت به صفر بیشتر باشد مقدار duty cycle نیز بیشتر و به طبع آن ولتاژ خروجی نیز بیشتر خواهد شد. در واقع زمانی که بخواهیم مقادیر میانی در خروجی داشته باشیم از آن استفاده می کنیم ، برای مثال اگر بخواهیم شدت نور یک lcdو همچنین روشن یا خاموش بودن آن را فقط با 1 سیگنال مشخص کنیم از این روش استفاده می کنیم . کاربرد : کنترل کردن سرعت و همچنین زاویه موتور ، کنترل کردن مقدار روشنایی لامپ و....

کاربردهای سروو موتور

در ابتدا سروو موتور را معرفی میکنیم:

سروو موتور عملگر دورانی یا عملگر خطی است که امکان کنترل دقیق موقعیت زاویه ای یا خطی، سرعت و شتاب را فراهم می‌کند. سروو موتورها شامل یک موتور مناسب به همراه یک سنسور خاص برای بازخورد موقعیت (Position Feedback)هستند. سروو موتورها همچنین شامل یک کنترلر تقریباً پیچیده هستند، که معمولاً خود یک واحد مجزای طراحی شده برای آنها می‌باشد.

سروو موتورها کلاس خاصی از موتورها نیستند، با اینکه معمولاً از عبارت *servomotor* برای اشاره به موتورهایی استفاده می‌شود که برای استفاده در سیستم‌های کنترل حلقه-بسته مناسب هستند.

توان نامی این موتورها بین چند دهم وات تا چندصد وات متغیر است.

سروموتورها در دو نوع ساخته می‌شوند:

1. سروموتورهای جریان مستقیم (دی‌سی)
2. سروموتورهای جریان متناوب (ای‌سی)

همچنین سرو موتور ها در انواع گیربکس دار و بدون گیر بکس وجود دارند و در اندازه های خیلی کوچک تا اندازه های بزرگ تولید می شوند

در تعداد زیادی از کاربردها که در آن‌ها کنترل موقعیت بسیار مهم هست ، استفاده می شود. ما نمی توانیم از موتور DC برای کنترل دم هواپیما استفاده کنیم زیرا کنترل موتور DC برای قرار گیری دقیق نه تنها دشوار بلکه تقریبا غیرممکن است. در کاربرد هایی مانند این ، ما برای انجام صحیح کار به یک سروو موتور متکی هستیم.

از سرو موتور ها در هر جایی که الکترو موتور ها کاربرد دارد می توان استفاده کرد اما به دلیل قیمت بالای آن نسب به الکترو موتور ها، ما تنها در جاهایی از سرو موتور استفاده می کنیم که عملا با الکترو موتور های

معمولی کار به خوبی انجام نمی شود و نیاز به دقت و سرعت عمل بالایی داریم . برخی از دستگاه ها و مکان هایی که سرو موتور در آنها استفاده می شود:

ماشین آالت نساجی ، ماشین آالت چاپ ، سینما ۵ بعدی و ۳ بعدی ، دستگاه های تزریق پالستیک ، دستگاه های پزشکی ، دستگاه های CNC فلزات،چوب،طال ماشین آالت و دستگاه های تولید قطعات الکترونیکی و...

توضیح در مورد ورودی آنالوگ و تحلیل آن در آردوینو و تابع مورد استفاده این آزمایش:

- **analogRead()**

سیگنال ورودی را تبدیل به سیگنال پله پله ای کرده در بازه 0 تا 1023 و در نهایت آن را به عنوان ورودی در نظر می گیرد . و سپس به ما معادل زاویه را عددی بین 0 تا 1023 می دهد و ما آن را به 0 تا 180 تبدیل می‌کنیم.

: analogRead()

مقدار را از پین آنالوگ مشخص شده می خواند. بردهای آردوینو دارای مبدل چند کاناله 10 بیتی آنالوگ به دیجیتال هستند. این بدان معنی است که ولتاژهای ورودی بین 0 و ولتاژ عملیاتی (5 ولت یا 3.3 ولت) را به مقادیر صحیح بین 0 و 1023 ترسیم می کند.

تعریف مختصر توابع مورد نیاز از کتابخانه Servo.h مانند:

تابع attach() :

متغیر Servo را به یک پین وصل می کند .

attach(pin)

البته نسخه دیگری هم دارد که بصورت زی میباشد:

attach(pin, min , max)

که در آن:

Min به معنای عرض پالس در میکرو ثانیه ، مربوط به حداقل زاویه روی سروو

Max به معنای عرض پالس در میکرو ثانیه ، مربوط به حداکثر زاویه روی سروو

تابع write() :

برای سروو مقداری را می نویسد ، شفت را بر این اساس کنترل می کند. در یک سروو استاندارد ، این زاویه شفت را (بر حسب درجه) تنظیم می کند (بین 0 تا 180)و شفت را به آن جهت می برد. در سروو چرخش مداوم ، این سرعت سروو را تنظیم می کند(درصورتی که 0 در یک جهت تمام سرعت می چرخد ، 180 در جهت دیگر تمام سرعت می چرخد و 90 بدون حرکت است). پس به این صورت است:

write(angle)

تابع read() :

زاویه فعلی سروو را می خواند در واقع مقدار پارامتر پاس داده شده به آخرین صدا زدن تابع write() را میخواند.

تابع writeMicroseconds() :

مقدار میکروثانیه (uS)برای سروو می نویسد و شفت را بر این اساس کنترل می کند. در سروو استاندارد ، این زاویه شفت را تنظیم می کند. در سروهای استاندارد مقدار پارامتر 1000 کاملا خلاف جهت عقربه های ساعت ، 2000 کاملاً در جهت عقربه های ساعت و 1500 در وسط قرار دارد. پس به صورت زیر است:

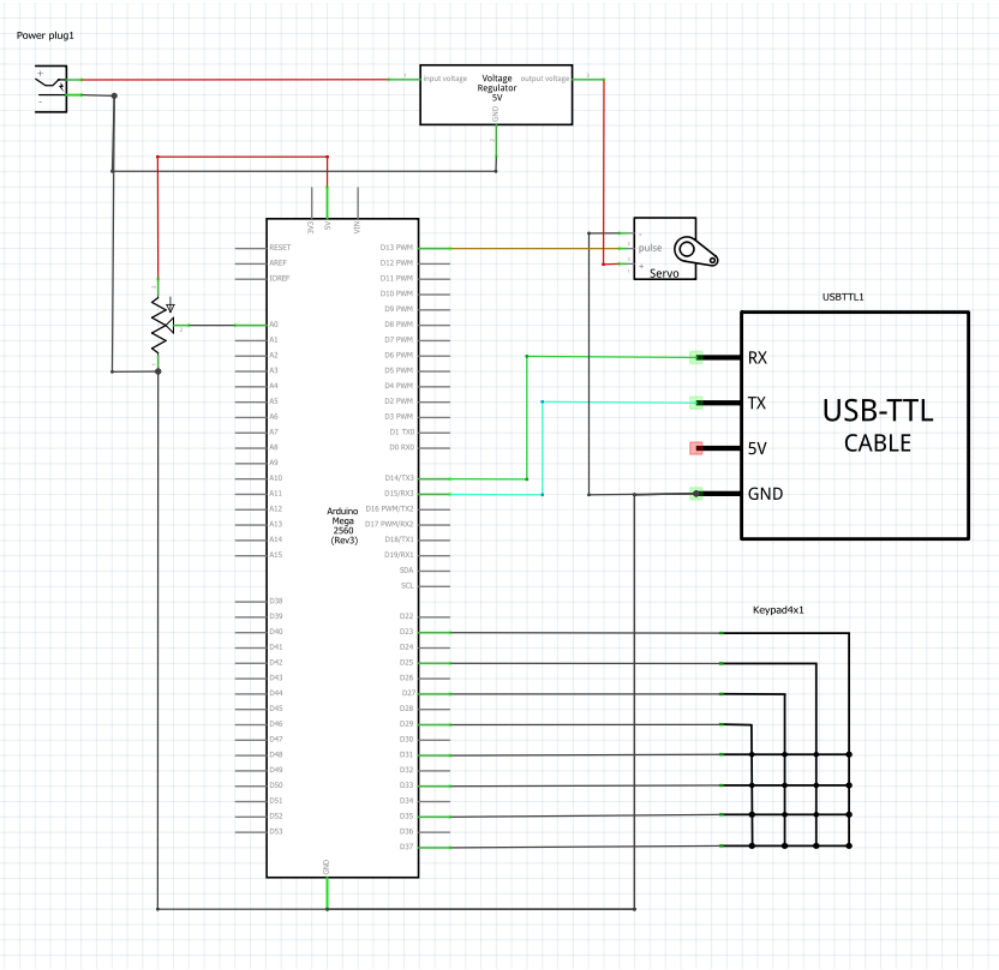
writeMicroseconds(uS)

تابع readMicroseconds() :

مقدار uS فعلی سروو را میخواند. درواقع مقدار پرامتر پاس داده شده به آخرین صدا زدن تابع witeMicrosecods() میباشد.

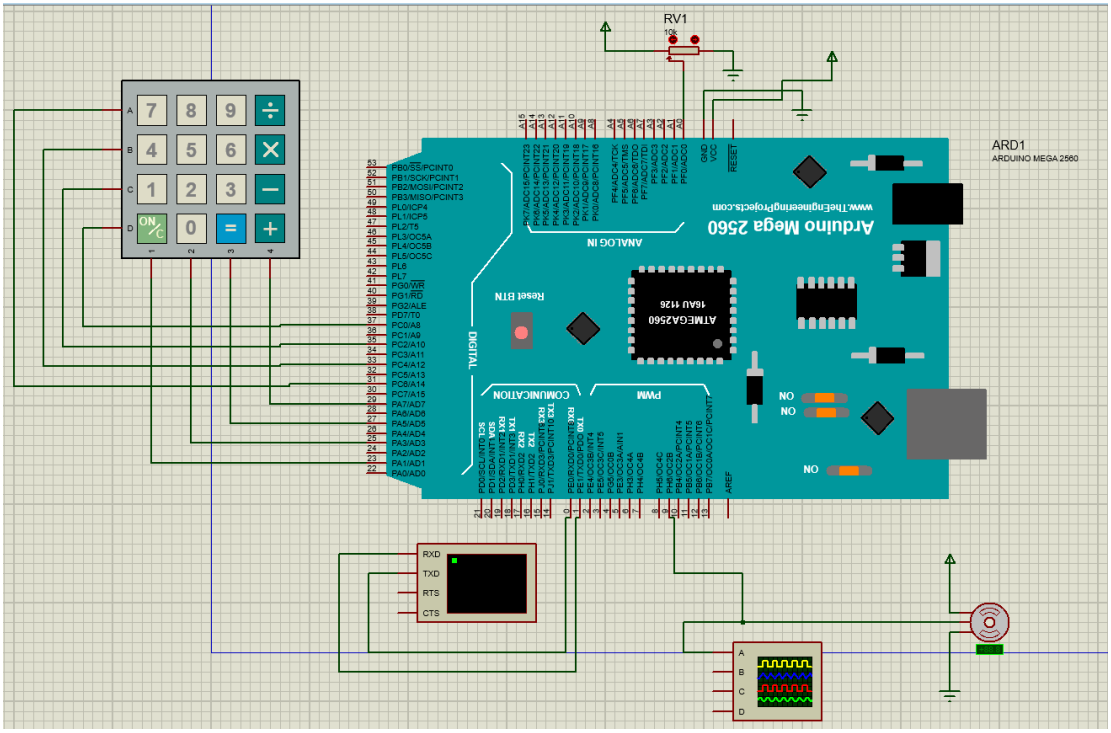
«گزارش کار»

ابتدا مدار زیر را بررسی میکنیم:



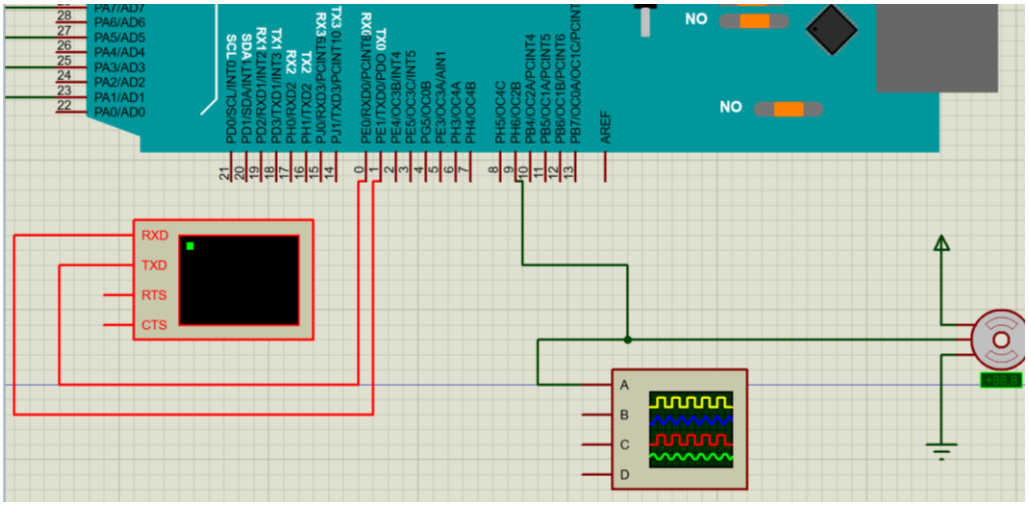
برای بستن مدار بالا در پروتئوس، به یک صفحه کلید keypad x44،یک برد Arduino2560،
یک پتانسیومتر با مقاومت 10 کیلو اهم ، یک سروو موتور و یک اوسیلوسکپ و یک ترمینال مجازی احتیاج داریم. طبق دستور کار قطعات گفته شده را به پورت های مورد نظر در برد وصل می‌کنیم.

مدار مربوط به این آزمایش را در پروتئوس به صورت زیر میبندیم:



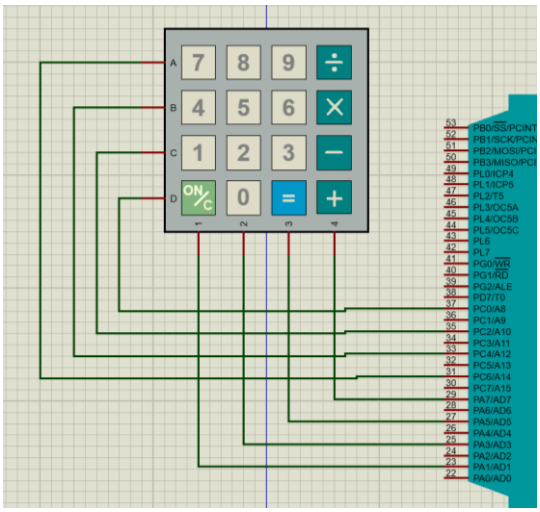
حال پین های خروجی را به صورت دقیق تری بررسی میکنیم:

پین های خروجی از بورد که به اسیلوسکوپ و ترمینال مجازی و سروو موتور وصل میباشد به صورت زیر است:



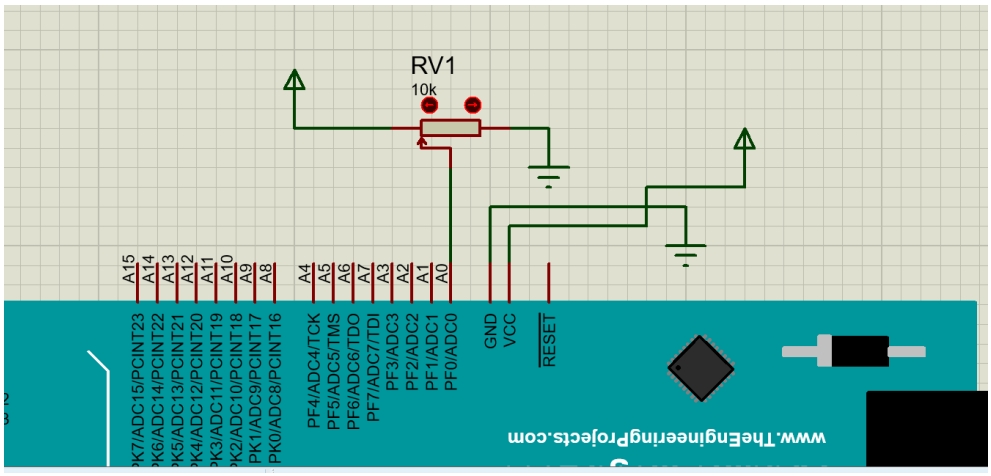
همانطور که مشاهده میشود پین شماره‌ی 9 بورد به سروو و پایه‌ی A از اسیلوسکوپ متصل میباشد. یک پایه‌ی سروو نیز به VCC و پایه‌ی دیگر آن نیز به GND متصل میباشد.

پین های خروجی از بورد که صفحه کلید وصل میباشد به صورت زیر اســت:



پین‌های خروجی بورد با شماره‌های 23 و 25و 27 و29 به ستون‌های صفحه کلید و پین‌های 31 و 33 و 35 و 37 بورد را نیز به سطرهای A و B و C و D صفحه کلید وصل میکنیم.

پین های خروجی از بورد که به پتانسیومتر و زمین و VCC وصل میباشد به صورت زیر اســت:



پین آنالوگ A0 را به پتانسیومتر متصل میکنیم. همچنین یک سر پتانسیومتر را به زمین و سر دیگر آن را به VCC وصل میکنیم. همچنین VCC و GND بورد آردوینو را هم به VCC و GND وصل میکنیم.

بخش اول) برنامه‌ای بنویسید که به صورت خودکار سروو از زاویه 0 تا 90 درجه تغییر کند و سپس از زاویه 90 به 0 بازگردد. سپس به صورت متناوب این حرکت را تکرار کند.

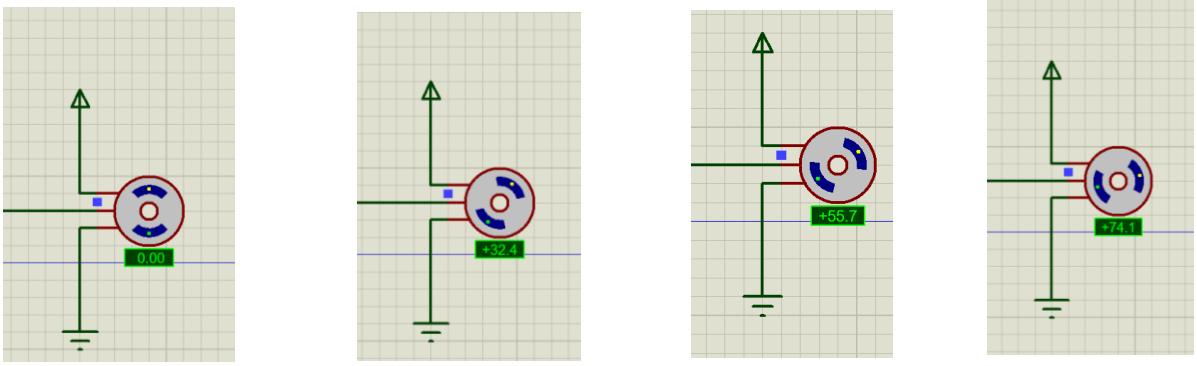
کد برنامه مربوط به این قسمت به صورت زیر میباشد:

code1

```
#include <Servo.h>

Servo myservo;
int i = 0;

void setup() {
  myservo.attach(9);
}

void loop() {
  for (i = 0; i <= 90; i++) {
    myservo.write(i);
    delay(15);
  }
  for (i = 90; i >= 0; i--) {
    myservo.write(i);
    delay(15);
  }
}
```

همانطور که مشاهده میشود ابتدا کتابخانه‌ی servo.h را لازم است تا include کنیم. سپس یک شیء از آن میسازیم. در قسمت setup باید به کمک attach پین شماره‌ی 9 را به عنوان خروجی برای سروو تنظیم کنیم. سپس در قسمت loop با کمک دو حلقه‌ی for یک بار از 0 تا 90 را به عنوان زاویه‌ی ورودی سروو write میکنیم. سپس وقتی به زاویه‌ی 90 درجه رسید به کمک حلقه‌ی دوم بار دیگر به پایین برمیگردیم و زاویه‌ها را یک درجه یک درجه کم میکنیم. این فرآیند به همین ترتیب به صورت متناوب تکرار میشود.

نمونه‌هایی از لحظات خروجی را در پروتئوس به صورت زیر میتوانید ببینید:



بخش دوم) برنامه‌ای بنویسید که کاربر با کیبورد یک عدد بین 0 تا 360 انتخاب کرده و سروو موتور آن را بین 180- درجه و 180+ درجه نشان دهد.

کد برنامه مربوط به این قسمت به صورت زیر میباشد:

code2

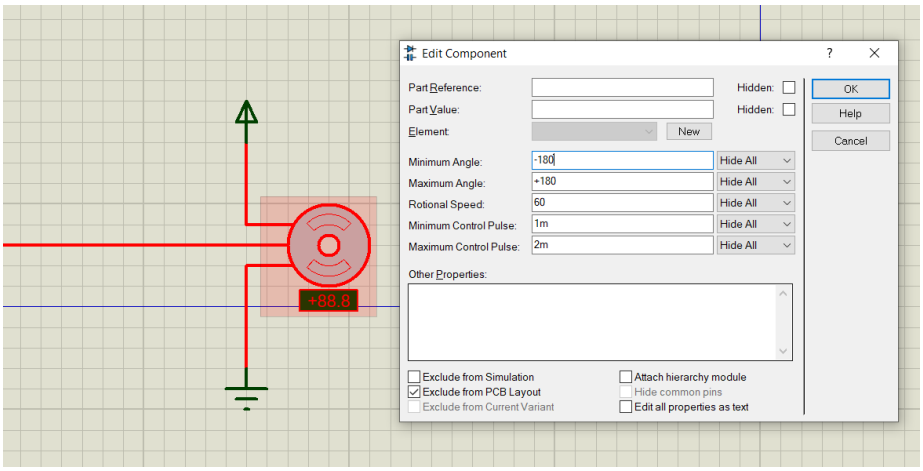
```
#include <Servo.h>
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 4; //three columns

char keys[ROWS][COLS] = {
  {'7','8','9','/'},
  {'4','5','6','*'},
  {'1','2','3','-'},
  {'s','0','=','+'}
};

byte rowPins[ROWS] = {31, 33, 35, 37};
byte colPins[COLS] = {23, 25, 27, 29};

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

Servo myservo;
int number = 0;
void setup() {
  myservo.attach(9);
  myservo.writeMicroseconds(700);
}
```

```
void loop() {
  char recieve = keypad.getKey();
  if (recieve)
  {
    if (recieve >= '0' && recieve <= '9')
    {
      number = (number * 10) + ((int)recieve - 48) ;
    }
    else if (recieve == '=')
    {
      if (number > 360)
        number = myservo.read();

      myservo.write(number);
      number = 0;
    }
  }
}
```

همانطور که مشاهده میشود چون در این قسمت کار با کیبورد داریم لازم است تا کتابخانه Keypad.h را نیز include کنیم. مانند آزمایش هایی که قبلا داشتیم باید ابتدا یک سی متغیر تعریف کنیم و تعداد سطر و ستون کیبورد را مشخص کنیم. یک آرایه دو بعدی بسازیم و فرم کلی کاراکترهای موجود در کیبورد را به آن بدهیم.
همچنین یک آرایه شامل شماره‌ی پین‌های خروجی بورد که به سطر های کیبورد متصل اند و یک آرایه شامل شماره‌ی پین‌های خروجی بورد که به ستون های کیبورد متصل اند را تعریف میکنیم.
سپس یک object از Keypad میسازیم و این متغیر ها را به عنوان ورودی به کانستراکتور آن میدهیم.
در مرحله‌ی بعد لازم است تا در setup پس از تعریف یک شیء از Servo تنظیمات مربوط به آن را نیز انجام دهیم.
در عکس سمت راست که مربوط به کد قسمت loop میباشد اینگونه گفتیم که اگر کاربر در کیبورد گزینه‌ای را فشار داد بیاییم بررسی کنیم که آیا آن یک کاراکتر عددی بین 0 تا 9 است یا = میباشد. اگر کاراکتر 0 تا 9 بود باید عدد مدنظر را با کمک کد اسکی این کاراکترها و ضربدر 10 کردن عدد در هرمرحله و جمع کردن با مقدار عدد جدید وارد شده را بسازیم.
اگر کاربر= را فشار داد و عدد بزرگ تر از 360 بود چون غیرمجاز است باید در همین موقعیتی که هست بماند.پس مقدار متغیر number را برابر با موقعیت فعلی میذاریم. در نهایت باید این مقدار را به کمک write به عنوان

ورودی سروو دهد. و سپس مقدار number را برای دفعات بعدی 0 کند.

برای آن که سروو عدد را بین 180- تا 180+ نشان دهد در پروتئوس به صورت زیر با کلیک کردن روی سروو آن را تنظیم میکنیم:



بخش سوم) برنامه‌ای بنویسید که با استفاده از سریال مانیتور، مقدار زاویه مورد نظر را وارد کنیم و سروو موتور به اندازه ی قرینه ی آن عدد تغییر زاویه دهد.

کد برنامه مربوط به این قسمت به صورت زیر میباشد:

cod3

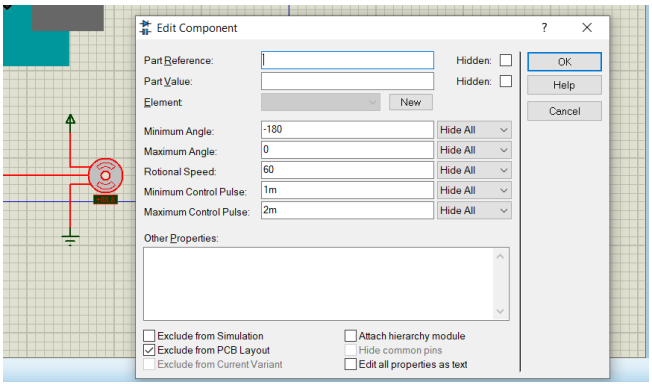
```
#include <Servo.h>

Servo myservo;
int number = 0;
int first = 1;
void setup() {
  Serial.begin(9600);
  myservo.attach(9, 1000, 2000);
  myservo.writeMicroseconds(700);
  delay(15);
}
```

```
void loop() {
  char recieve = Serial.read();

  if (recieve)
  {
    if (recieve >= '0' && recieve <= '9')
    {
      if (first == 1 && recieve == '0')
        return;
      number = (number * 10) + recieve - 48;
      Serial.write(recieve);
      first = 0;
    }
    else if (recieve == '=')
    {
      if (number > 180)
        number = myservo.read();
        myservo.write(180 - number);
      Serial.write(12);
      number = 0;
      first = 1;
    }
  }

}
```

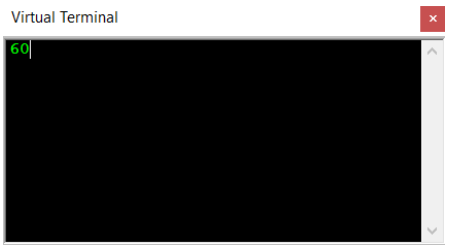
در این کد ابتدا یک شیء از servo میسازیم و آن را در بخش setup تنظیم میکنیم. تعیین min و max در تابع attach و همچنین کمک گرفتن از تابع writeMicroseconds باعث میشوند سروو ما کمی دقیق تر عمل کند.

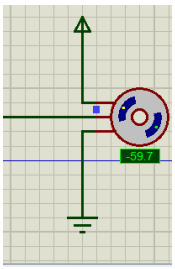
همچنین چون در این قسمت میخواهیم از ترمینال مجازی استفاده کنیم لازم است که برای ارتباط با آن Serial.begin را استفاده کنیم.

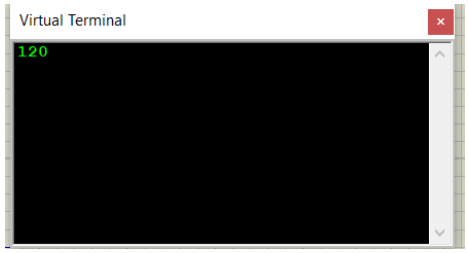
در قسمت loop هم قرار است ابتدا کاراکتر وارد شده در ترمینال را دریافت کنیم و بقیه ی مراحل مانند قسمت قبل میباشد. لازم به ذکر است که عدد وارد شده در این قسمت باید بین 0 تا 180 باشد سپس آن را از 180 کم میکنیم و از طرفی ر تنظیمات سروو در پروتئوس مینیمم و ماکسیمم را روی 180- تا 0 تنظیم میکنیم:

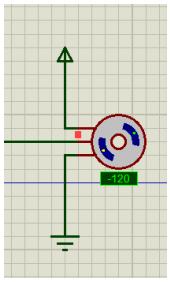


حال به این ترتیب اگر عددی در ترمینال زده شود و سپس مساوی را بزنیم در سروو به مقدار قرینه‌ی آن تغییر زاویه میدهد. نمونه هایی از خروجی را در پروتئوس میتوانید در زیر مشاهده کنید:









بخش چهارم) برنامه باید به گونه‌ای نوشته شود که با تغییر مقدار پتانسیومتر که به یک پایه آنالوگ برد متصل است، زاویه سروو موتور تغییر کند.

کد برنامه مربوط به این قسمت به صورت زیر میباشد:

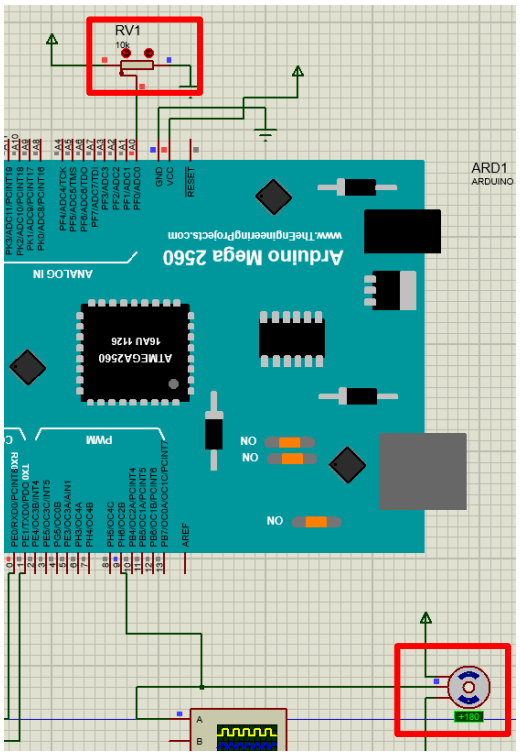
code4

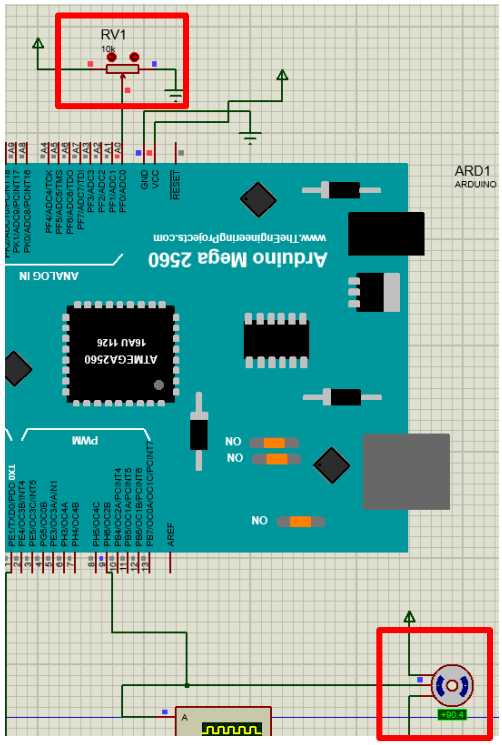
```
#include <Servo.h>

Servo myservo;


int val;
int potpin = A0;

void setup() {
  myservo.attach(9, 1000, 2000);
}

void loop() {
  val = analogRead(potpin);
  val = map(val, 0, 1023, 0, 180);
  myservo.write(val);
  delay(15);
}
```

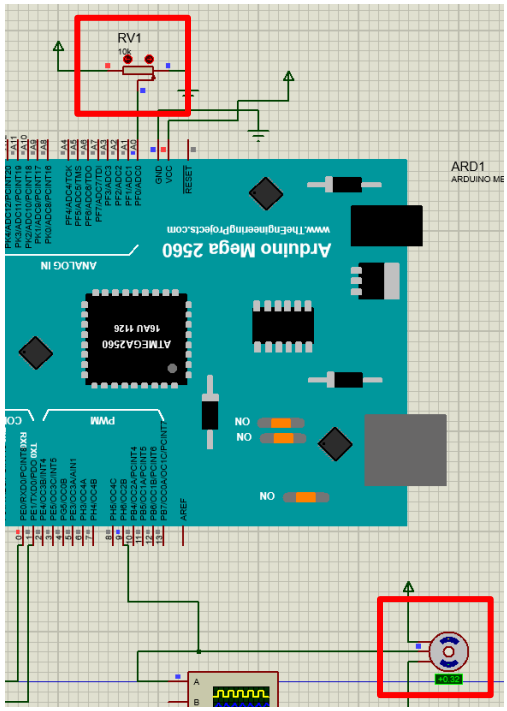
پس از اینکه کتابخانه‌ی Servo.h را include کردیم و یک object از آن ساختیم، دو متغیر تعریف میکنیم. یکی val که مقدار پتانسیومتر را در آن نگهداری میکنیم و یکی هم potpin که همان پین آنالوگ از آردوینو میباشد که به پتانسیومتر متصل میباشد یعنی همان A0.

پس از صدا زدن تابع attach در بخش setup و تنظیم کردن سروو به سراغ بخش loop میرویم که در آن باید به کمک تابع analogRead مقداری که به عنوان ورودی از پتانسیومتر به آن پین A0 می آید را در val ذخیره کنیم. سپس چون خروجی آن یک عدد بین 0تا 1023 میباشد به کمک تابع map آن را به 0 تا 180 درجه assign میکنیم.

عکس هایی از خروجی را میتونید در زیر مشاهده کنید:





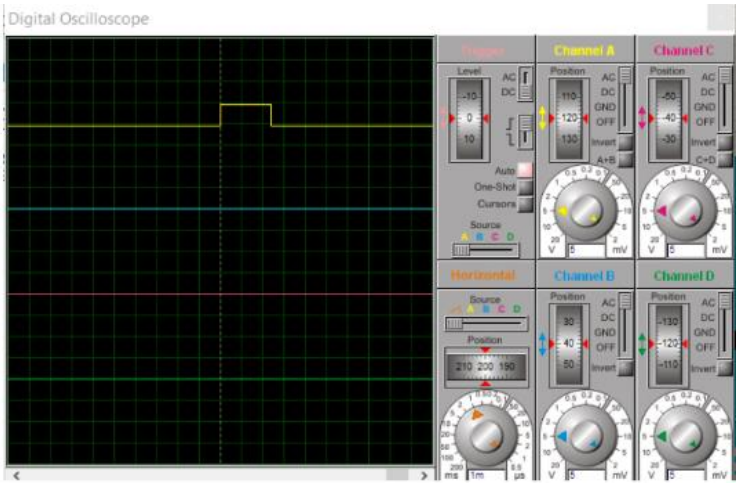


بخش پنجم) اسیلوسکوپ را به خط سروو موتور متصل کنید. چه چیزی متوجه می‌شوید؟ آیا می‌توانید دوره پایه (Fundamental Period) و همچنین دوره کاری (Duty Cycle) موج را به ازای زاویه‌های گردش مختلف سرو موتور به دست آورید؟

اگر زاویه را روی 180 درجه بگذاریم ، منطقا در این حالت ٪ 100 duty cycle داریم که طول این موج مقدار دوره پایه را به ما می دهد ، و همچنین به ازای هر زاویه میبینیم که اندازه Duty cycle تغییر می کند . و مقدار آن برابر است با :$\frac{PW}{T} \times 100$

PW: برابر است با طول زمانی که به ازای اون زاویه ، مقدار سیگنال 1 هست.

برای مثال برای 180 درجه اگر به اسیلوسکوپ نگاه کنیم:



همچنین برای 50 درجه به صورت زیر میباشد: