آزمایش ۴ ریزپردازنده

سپهر توکلی - ۹۸۳۱۱۱۱

محمد چوپان - ۹۸۳۱۱۲۵

مفهوم PWMو استفاده های آن:

اگر بخواهیم یک عدد یا مقدار را روی یک پین مشخص کنیم با توجه به آن که فقط دو حالت صفر یا یک داریم، به جای آنکه عدد مورد نظر را به صورت تغییرات آنی ولتاژ تعیین کنیم، به صورت تغییراتی(موج مربعی) در طول زمان آن را نشان می دهیم. پس با توجه به آنکه در هر duty cycle در یک موج مربعی، چه کسری از زمان سیگنال مقدار یک دارد و چه کسری مقدار صفر، میتوان عدد مورد نظر را نشان داد.

استفاده های PWM:

ارسال دیتا به سروو موتور

کنترل سرعت موتور

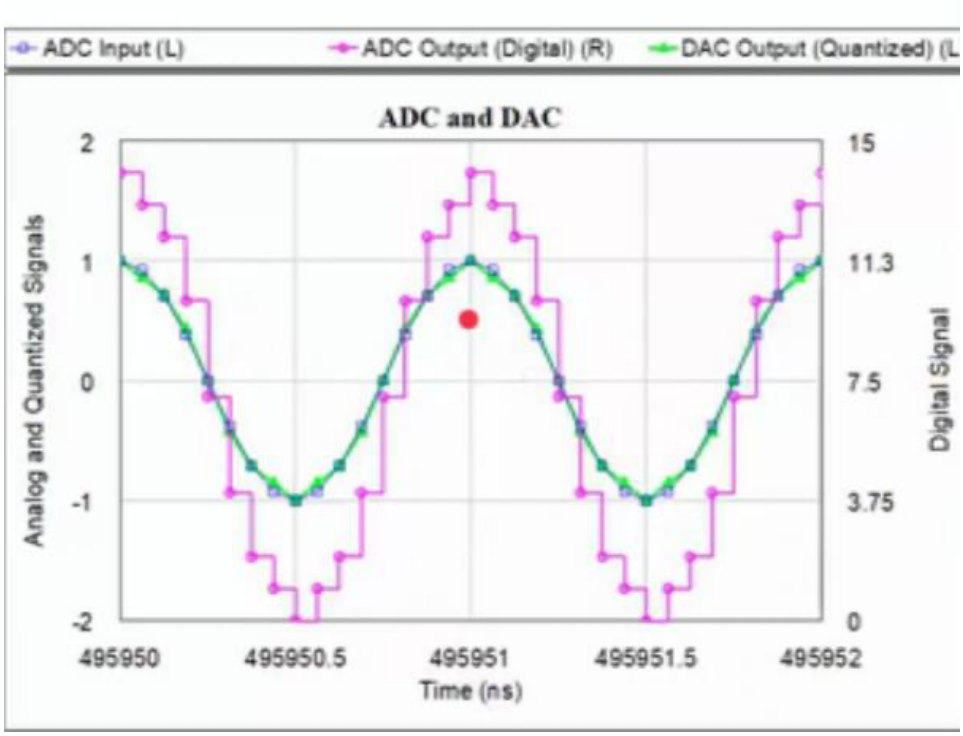
کنترل نور یا LEDبه کمک میکروکنترلر

کاربردهای سروو موتور:

عموما برای این استفاده میشوند که یک زاویهی کمی مثلا بین صفر تا ۱۸۰ درجه را بچرخند. مثلا چرخاندن پیچ یک دستگاه، بلند کردن یک وزنه. چون به خاطر جعبه دندهای که دارند زور آنها در رساندن موتور به یک زاویه خاص، می تواند خیلی زیاد باشد. به طور کلی در ساخت رباتها، در اتوموبیل ها برای ثابت نگه داشتن سرعت، سیستم های ردیابی خورشیدی، دستگاه های پزشکی و... کاربرد دارند.

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

در شکل زیر موج سبزرنگ یک ورودی آنالوگ از و بین مقادیر ۱ تا منفی ۱ ولت نوسان میکند. اگر این موج به همین صورت توسط یک دستگاه دیجیتال خوانده شود کل مقادیر در طی تمام زمانها، صفر خوانده می شود. برای آنکه این داده ها از بین نروند و به درستی خوانده شوند نیاز به یک مبدل آنالوگ به دیجیتال داریم. تا این موج را به صورت موج قرمز رنگ در شکل روبرو، در بیاورد .



analogRead()

مقدار را از یک پین آنالوگ می خواند و موج ورودی آنالوگ را به مقادیر دیجیتال تبدیل میکند. یعنی مقادیر بین 0 تا 5 ولت را به مقادیر صحیح بین 0تا 1023 نگاشت می کند.

Syntax

```
analogRead(pin)
```

Parameters

`pin`: the name of the analog input pin to read from (A0 to A5 on most boards, A0 to A6 on MKR boards, A0 to A7 on the Mini and Nano, A0 to A15 on the Mega).

Returns

The analog reading on the pin. Although it is limited to the resolution of the analog to digital converter (0-1023 for 10 bits or 0-4095 for 12 bits). Data type: `int`.

توابع مورد نیاز از کتابخانه servo.h :

attach()

سروو را به یک پین (که دیتای یک موج مربعی روی آن است) وصل می کند. در آردوینوهای قدیمی تر فقط پین های ۹ و ۱۰ این ویژگی را ساپورت می کردند.

Description

Attach the Servo variable to a pin. Note that in Arduino 0016 and earlier, the Servo library supports only servos on only two pins: 9 and 10.

Syntax

servo.attach(pin)
servo.attach(pin, min, max)

Parameters

servo: a variable of type Servo

pin: the number of the pin that the servo is attached to

min (optional): the pulse width, in microseconds, corresponding to the minimum (0-degree) angle on the servo (defaults to 544)

max (optional): the pulse width, in microseconds, corresponding to the maximum (180-degree) angle on the servo (defaults to 2400)

Write()

در یک سروو استاندارد (۱۸۰ درجه ای) زاویه را Set می کند و در یک سروو ۳۶۰ درجه سرعت چرخش را Set می کند. که ۰ به معنی حداکثر سرعت در یک جهت، ۱۸۰ به معنی حداکثر سرعت در جهت مخالف و ۹۰ به معنی ساکن بودن است .

Description

Writes a value to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft (in degrees), moving the shaft to that orientation. On a continuous rotation servo, this will set the speed of the servo (with 0 being full-speed in one direction, 180 being full speed in the other, and a value near 90 being no movement).

Syntax

servo.write(angle)

Parameters

servo: a variable of type Servo

angle: the value to write to the servo, from 0 to 180

read() :

زاویه کنونی سروو را می خواند (آخرین مقداری که در write) پاس داده شده است.

Description

Read the current angle of the servo (the value passed to the last call to write()).

Syntax

servo.read()

Parameters

servo: a variable of type Servo

Returns

The angle of the servo, from 0 to 180 degrees.

writeMicroseconds() :

مقداری که ورودی می دهیم، 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 سروو‌های استاندارد، مقدار ۱۰۰۰ کاملاً خلاف جهت عقربه های ساعت ، ۲۰۰۰ کاملاً در جهت عقربه های ساعت و ۱۵۰۰ در وسط قرار دارد . چون در برخی موتور ها ممکن است کاملا استاندارد نباشند و دقیق قرار نگیرند، محدوده را ۷۰۰ تا ۲۳۰۰ را می توان به عنوان ورودی داد. دقت اندازه گیری که اینجا می توانیم داشته باشیم از write بیشتر است . در سروو‌های ۳۶۰ درجه، این تابع همانند تابع writeاست .

Description

Writes a value in microseconds (uS) to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft. On standard servos a parameter value of 1000 is fully counter-clockwise, 2000 is fully clockwise, and 1500 is in the middle.

Note that some manufactures do not follow this standard very closely so that servos often respond to values between 700 and 2300. Feel free to increase these endpoints until the servo no longer continues to increase its range. Note however that attempting to drive a servo past its endpoints (often indicated by a growling sound) is a high-current state, and should be avoided.

Continuous-rotation servos will respond to the writeMicrosecond function in an analogous manner to the write function.

Syntax

servo.writeMicroseconds(uS)

Parameters

servo: a variable of type Servo

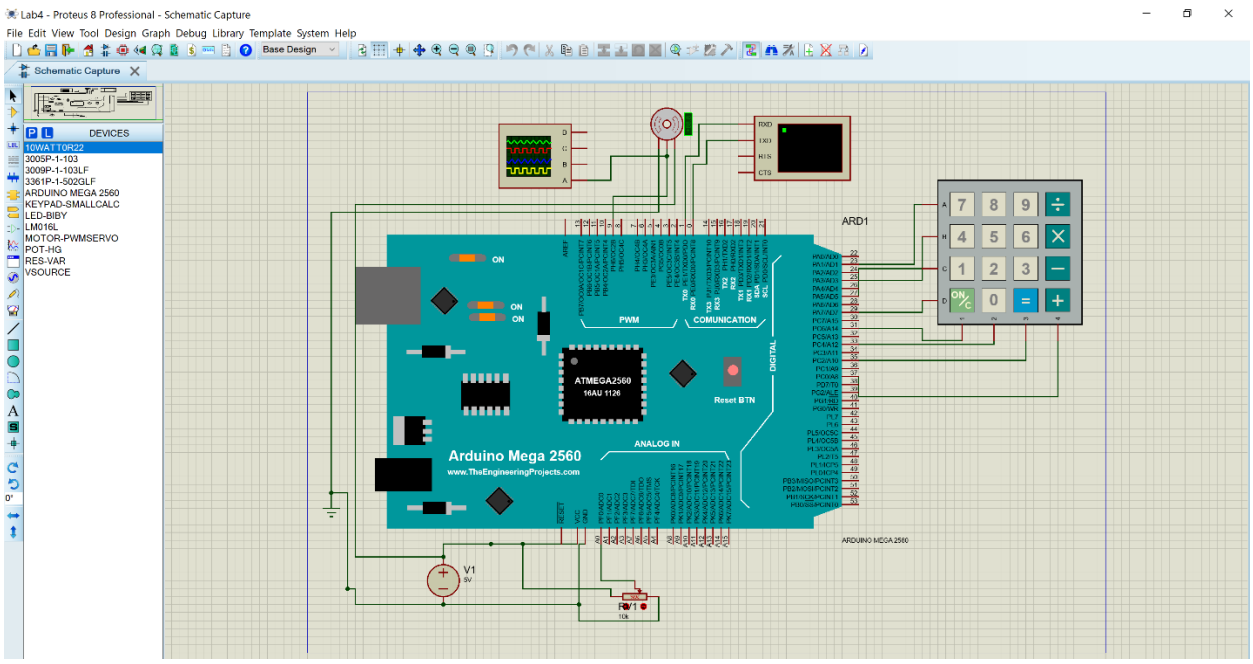
uS: the value of the parameter in microseconds (*int*)

readMicroseconds ():

آخرین مقدار نوشته شده توسط writeMicrosecondsرا با واحد میکروثانیه برمیگرداند .

شرح آزمایش :

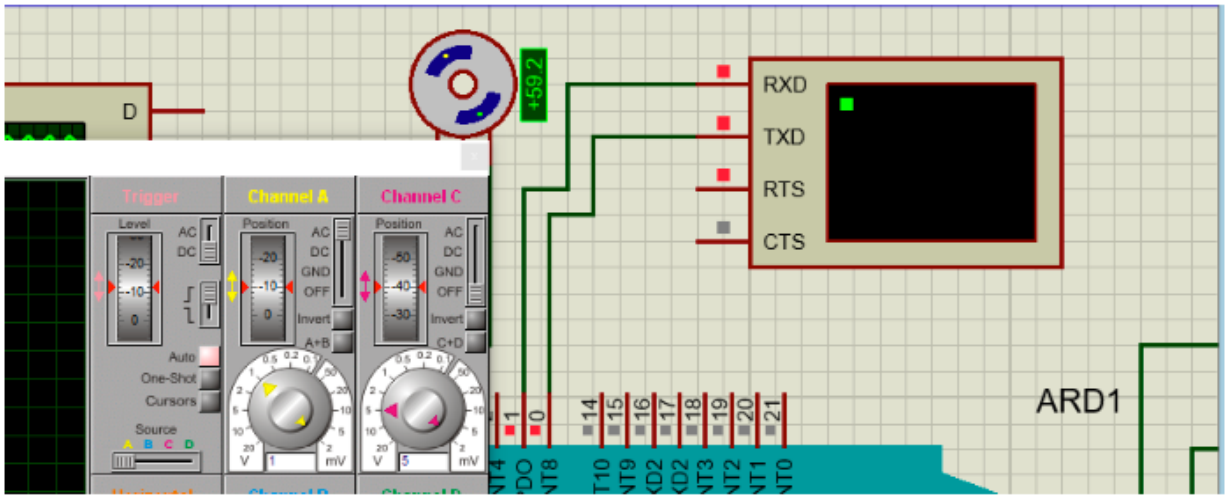
در ابتدا مدار مورد نیاز را به صورت زیر می بندیم :

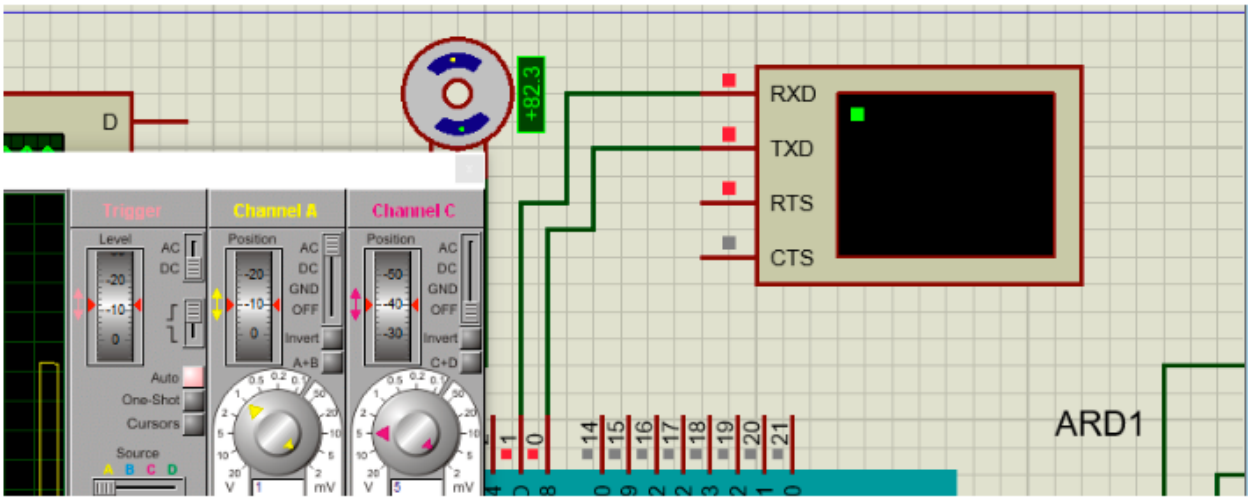


۱) برنامهای بنویسید که به صورت خودکار سروو از زاویه 0تا 90درجه تغییر کند و سپس از زاویه 90به 0 بازگردد . سپس به صورت متناوب این حرکت را تکرار کند .

در setup، ابتدا degreeرا 0می کنیم، و stepsرا هم 1در نظر گرفتیم. با توجه به کد همواره مقدار Degreeیکی زیاد میشود تا به 90برسد، در این حالت 1-می شود و هر بار اجرا شدن تابع، degree یکی کم میشود تا به 0برسد. این پروسه همواره تکرار می شود .

```
#include <Servo.h>

Servo myservo;

String input;
int degree;



int steps = 1;

void setup() {
  myservo.attach(9,1000,2000);
  Serial.begin(9600);
  degree = 0;
  input = "";
}

void loop() {
  myservo.write(degree);
  delay(30);
  if(degree == 90)
    steps = -1;
  if(degree == 0)
    steps = 1;
  degree = degree + steps ;

}
```

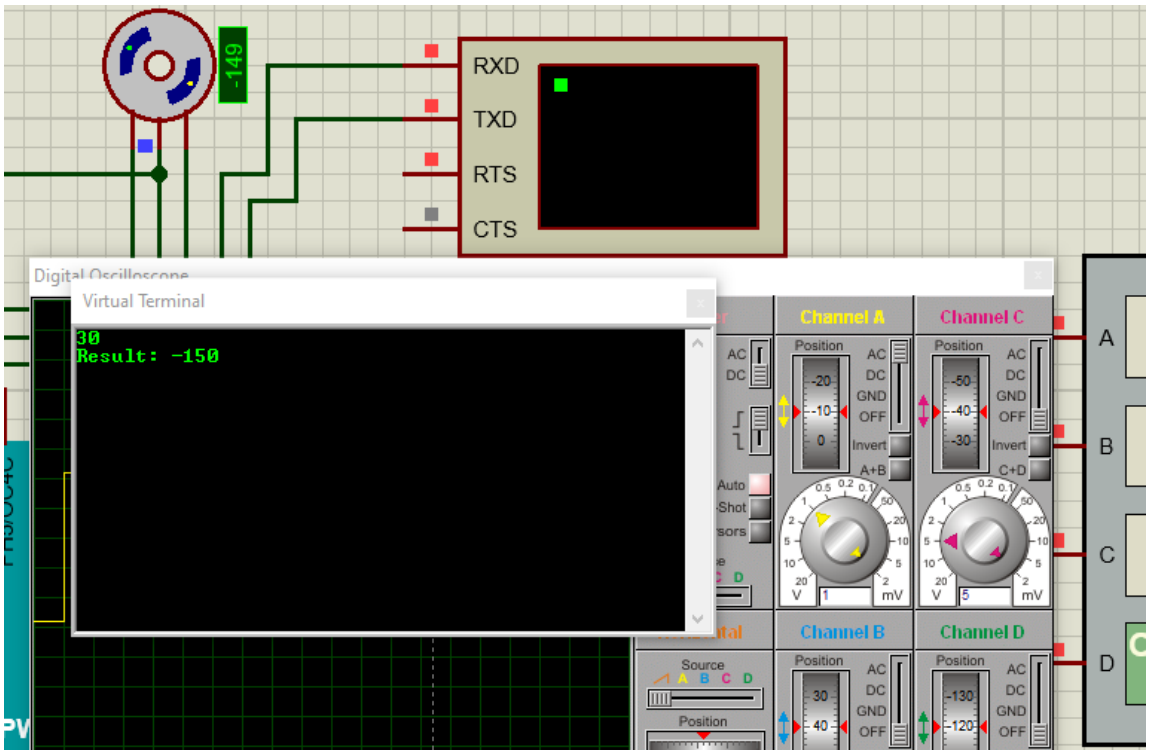

D
+59.2
RXD
TXD
RTS
CTS
Trigger
Channel A
Channel C
Level
Position
AC
DC
GND
OFF
Invert
A+B
C+D
Auto
One-Shot
Cursors
Source
ARD1

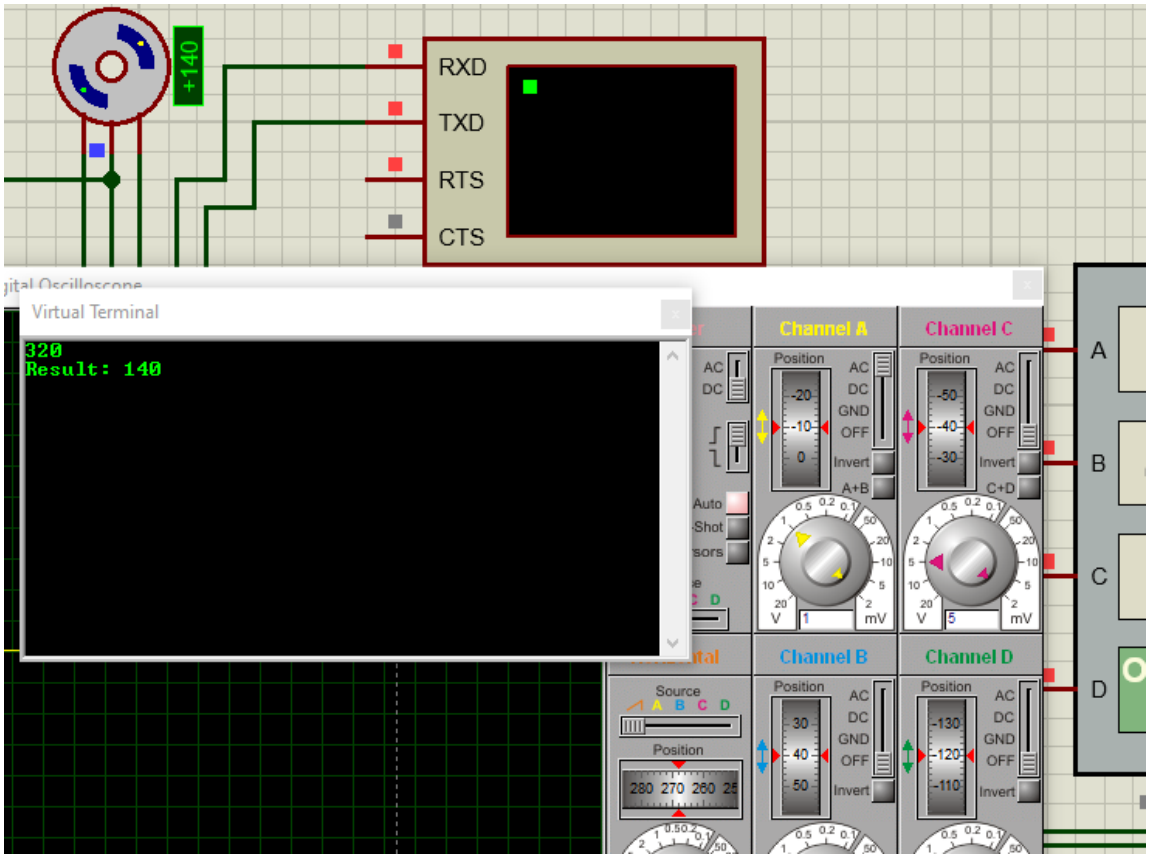

D
+82.3
RXD
TXD
RTS
CTS
Trigger
Channel A
Channel C
Level
Position
AC
DC
GND
OFF
Invert
A+B
C+D
Auto
One-Shot
Cursors
Source
ARD1

۲) برنامه ای بنویسید که کاربر با کیبورد، یک عدد بین 0و 360انتخاب کرده و سروو موتور آن را بین 180- درجه و 180 درجه نشان دهد .

درکد تابع این قسمت، اگر دکمه ای درکیپد فشار داده شده بود آن را خوانده و اگر کاراکترش بین 0تا 9بود به inputاضافه میکنیم. اگر کلید = فشرده شد یعنی ورودی آماده است. چون موتور سروو بین 0تا 180میتواند مقدار بگیرد و ورودی ما تا 360است، پس از تبدیل ورودی از رشته به عدد، آن را بر 2تقسیم میکنیم تا در بازه مورد نظر ما قرار گیرد. برای دیدن نتیجه نهایی و زاویه ای که قرار است موتور چاپ کند آن را در ترمینال چاپ میکنیم. پس از آن با تابع writeمقدار به دست آمده را به موتور میدهیم. در این حالت لازم است در مدار، با دابل کلیک روی موتور، بازه زاویه آن را از 0تا 180 درجه به 180- تا 180تغییر دهیم تا خود موتور زاویه را با مقدار مورد نظر ما نشان دهد. با انجام شدن تبدیل مورد نظر، زاویه خروجی روی موتور باید 180درجه از زاویه ورودی کمتر باشد. (مثلا اگر 0وارد کردیم، 180-درجه در موتور نمایش داده شود و به همین ترتیب) .

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

const byte rows = 4; //four rows
const byte cols = 4; //four columns
char keys[rows][cols] = {
  {'7','8','9','/'},
  {'4','5','6', '*'},
  {'1','2','3', '-'},
  {'c', '0', '=', '+'}
};

byte rowPins[rows] = {23, 25, 27, 29}; //connect to the row pinouts of the keypad
byte colPins[cols] = {31, 33, 35, 37}; //connect to the column pinouts of the keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols );

Servo myservo;
String input;
int degree;
String serialInput = "";
int serialDegree = 0;

int potpin = 0;
int val;

int steps = 1;

void setup() {
  myservo.attach(9,1000,2000);
  Serial.begin(9600);
  keypad.addEventListener(keypadEvent);
  degree = 0;
  input = "";
}

void loop() {
  char key = keypad.getKey();
}



void keypadEvent(KeypadEvent key){
  if(keypad.getState() == PRESSED){
    if(key >= '0' && key <= '9'){
      Serial.print(key);
      input += key;
    }
    if(key == '='){
      degree = input.toInt();
      Serial.println();
      Serial.print("Result: ");
      Serial.println(degree - 180);
      degree = degree/2;
      myservo.write(degree);
      Serial.println();
      input = "";
    }
  }
}
```

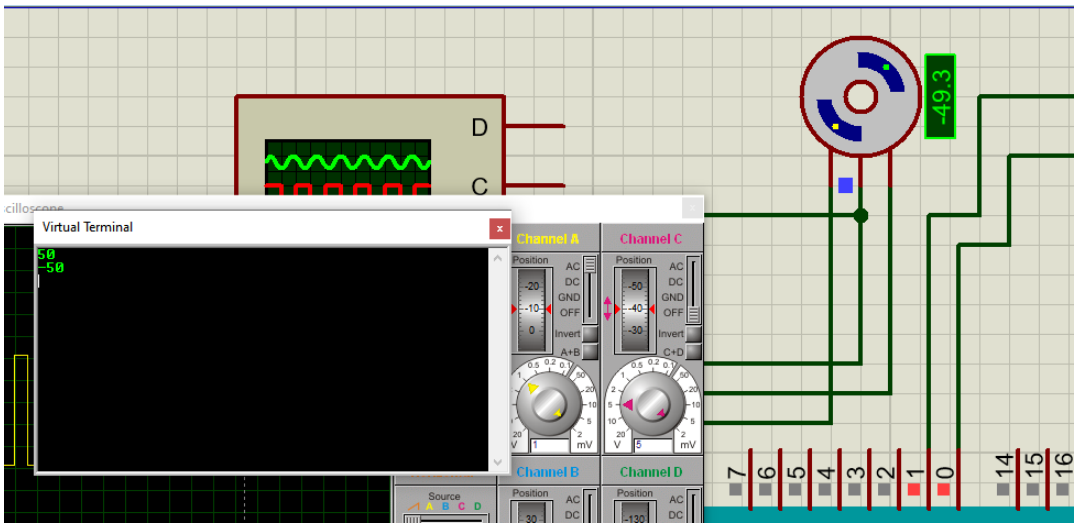

-149
RXD
TXD
RTS
CTS
Virtual Terminal
30
Result: -150
Digital Oscilloscope
Channel A
Channel B
Channel C
Channel D
A
B
C
D

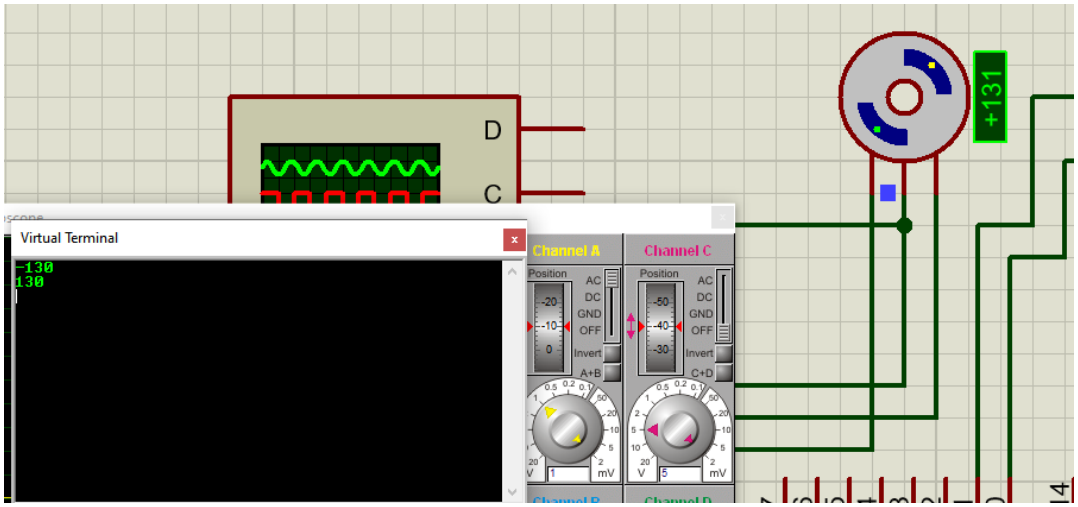

+140
RXD
TXD
RTS
CTS
Virtual Terminal
320
Result: 140
Channel A
Channel B
Channel C
Channel D
A
B
C
D

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

برای تابع این قسمت، ابتدا چک میکنیم که داده وارد ترمینال شده باشد، در این صورت رشته را خوانده و آن را به عدد تبدیل میکنیم. این بار هم چون با وارد کردن زاویه منفی، از محدوده 0تا 180 خارج میشویم، ورودی را از 180کم کرده و تقسیم بر 2میکنیم. محدوده سروو را نیز مانند قسمت قبل به 180– تا 180تغییر میدهیم. خود ورودی و نتیجه ای که قرار است موتور نشان دهد را در ترمینال چاپ میکنیم. سپس با writeنتیجه را به موتور منتقل میکنیم .

```
#include <Servo.h>

Servo myservo;
String input;
int degree;
String serialInput = "";
int serialDegree = 0;

int potpin = 0;
int val;

int steps = 1;

void setup() {
  myservo.attach(9,1000,2000);
  Serial.begin(9600);
  degree = 0;
  input = "";
}

void loop() {
if(Serial.available() > 0){
    serialInput = Serial.readString();
    Serial.println(serialInput.toInt());
    serialDegree = (180 - serialInput.toInt())/2;
    Serial.println(serialInput.toInt()* (-1));
  }
  myservo.write(serialDegree);
}
```


D
C
-49.3
Virtual Terminal
50
-50
Channel A
Channel C
Channel B
Channel D
7
6
5
4
3
2
1
0
14
15
16

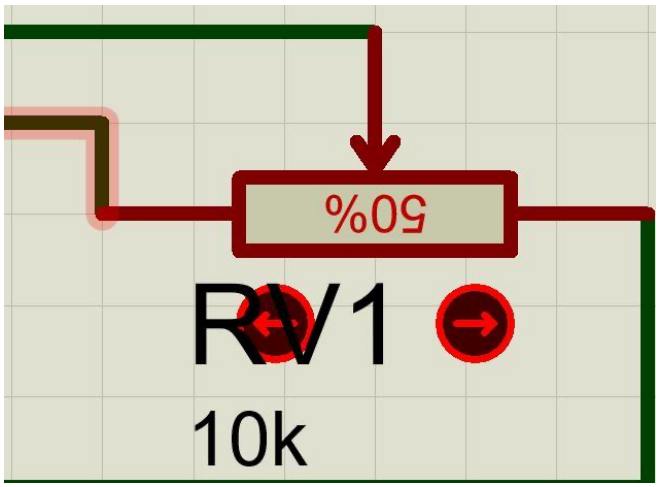

D
C
+131
Virtual Terminal
-130
130
Channel A
Channel C

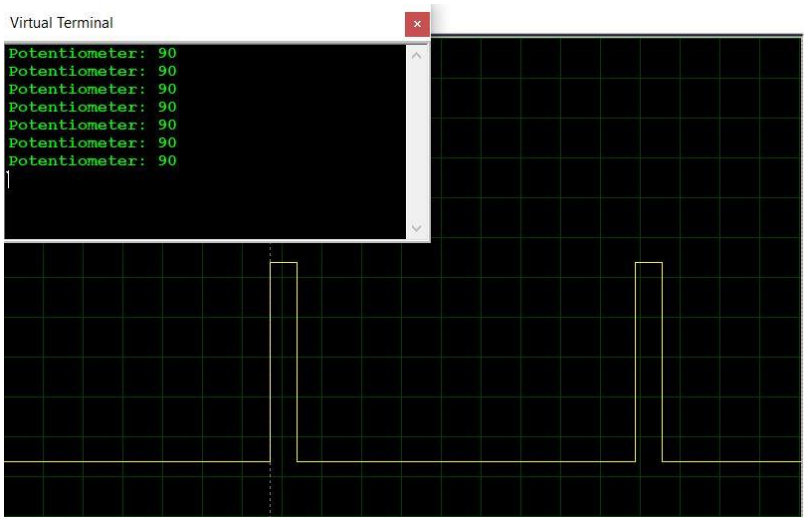
۴) برنامه باید به گونهای باشد که با تغییر مقدار پتانسیومتر که به یک پایه آنالوگ برد متصل است، زایه سروو موتور تغییر کند .

مانند شکل زیر پورت مربوط به پتانسیومتر را تعریف کردیم، مقدار آن را خواندیم و مپ کردیم. سپس روی صفحه نمایش نشان دادیم .

ابتدا با ضرب درصد پتانسیومتر که ۱۰کیلو اهم تعریف شده است مقدار فعلی مقاومت را محاسبه میکنیم، عددی نمایش داده شده باید حاصل ضرب آن در ۱۸۰ تقسیم بر ۱۰۲۴ باشد .

```
#include <Servo.h>

Servo myservo;
String input;
int degree;
String serialInput = "";
int serialDegree = 0;

int potpin = 0;
int val;

int steps = 1;

void setup() {
  myservo.attach(9,1000,2000);
  Serial.begin(9600);
  degree = 0;
  input = "";
}

void loop() {
    Serial.print("Potentiometer: ");
    val = analogRead(A0);            // reads the value of the potentiometer (value between 0 and 1023)
    val = map(val, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0 and 180)
    Serial.println(val);
    myservo.write(val);
    delay(1000);
}
```

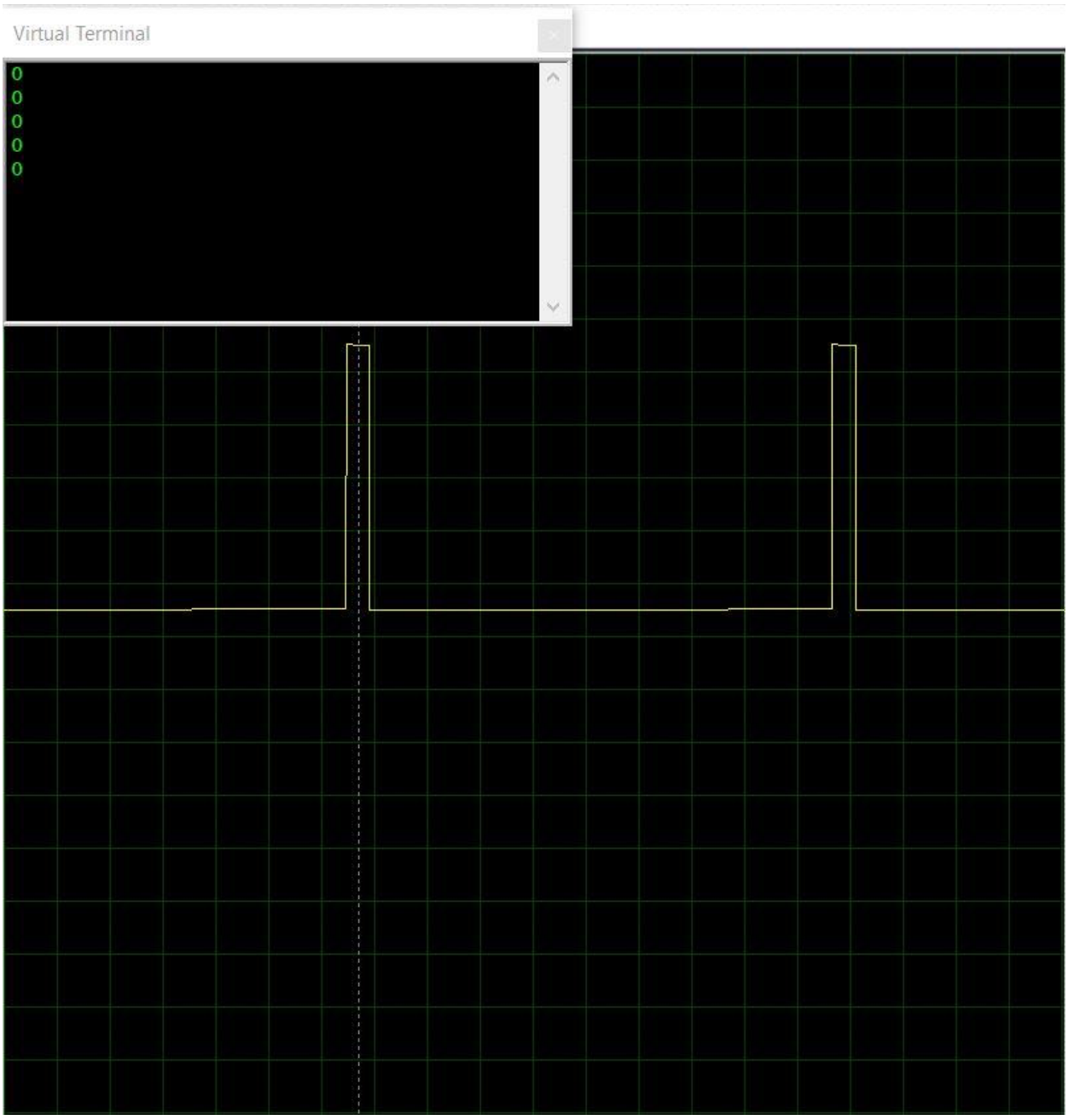

50%
RV1
10k

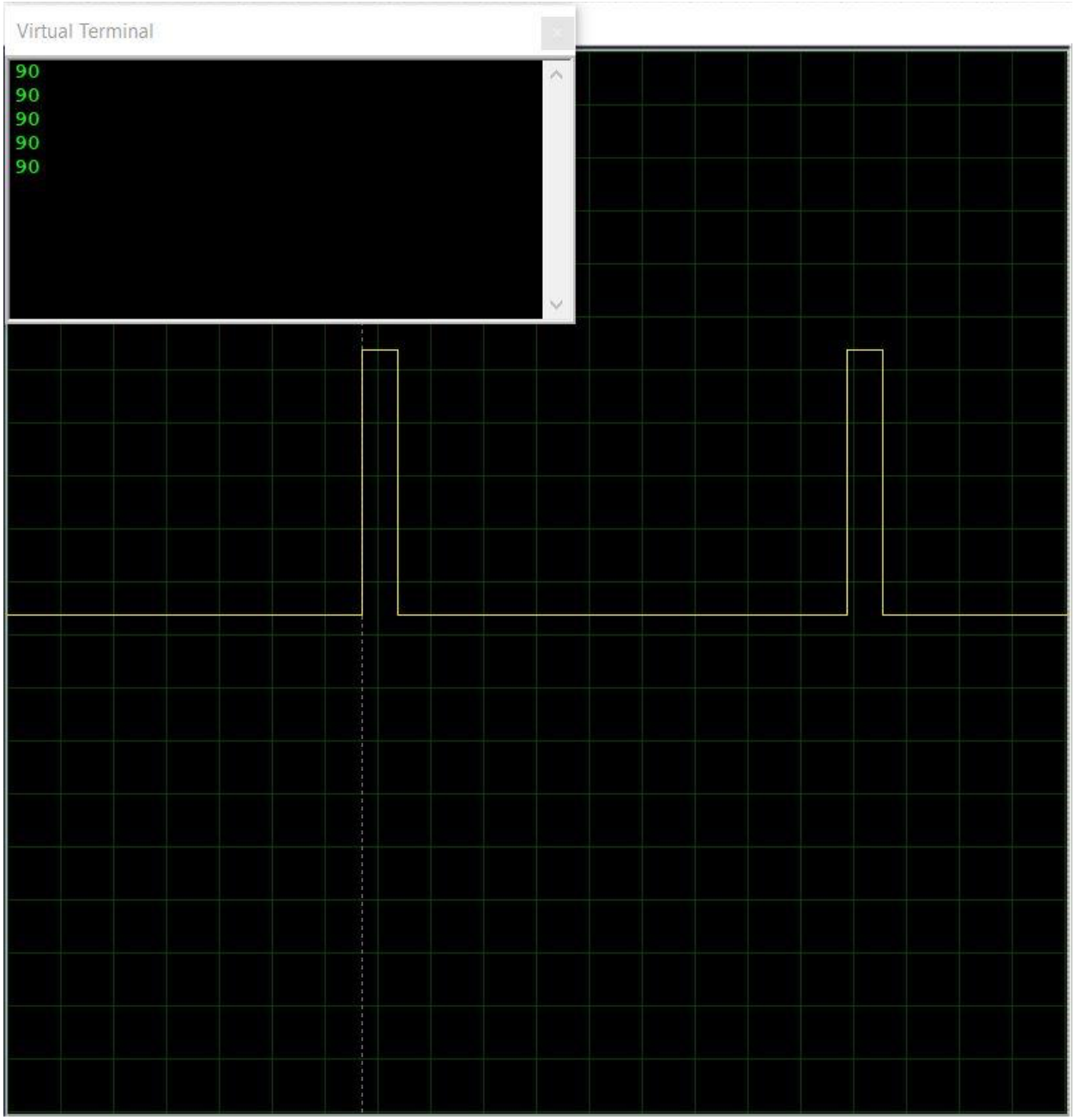

Virtual Terminal
Potentiometer: 90
Potentiometer: 90
Potentiometer: 90
Potentiometer: 90
Potentiometer: 90
Potentiometer: 90
Potentiometer: 90

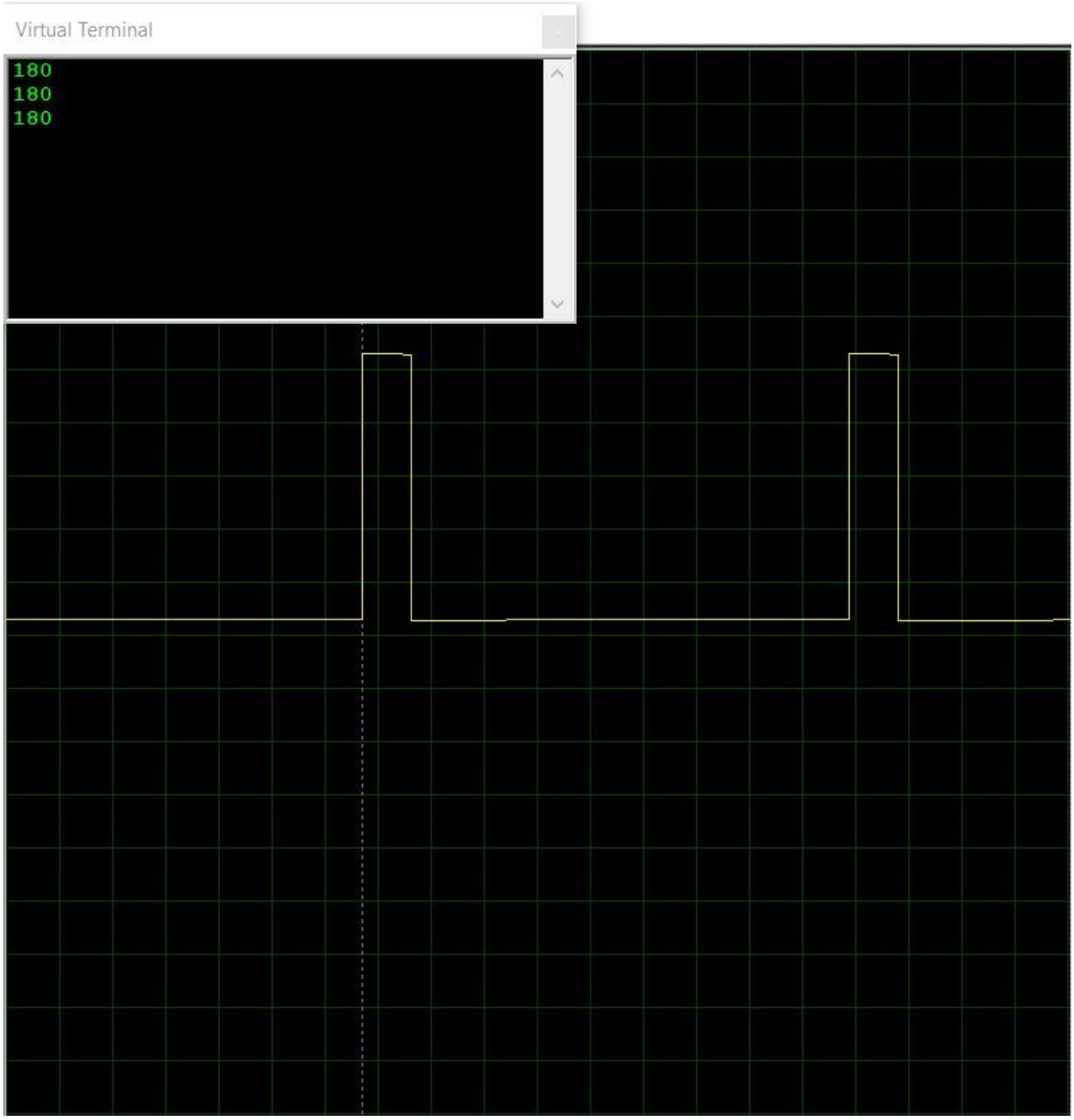
۵) اسیلوسکوپ را به خط سروو موتور متصل کنید. چه چیزی متوجه میشوید؟ آیا میتوانید دوره پایه و همچنین دوره کاری موج را به ازای زاویه های گردش مختلف سروو موتور به دست آورید؟

مانند کد مقدارهای مختلف را امتحان کرده و نتایج زیر را به دست آوردیم .

```
val = 0;
Serial.println(val);
servo.write(val);
delay(1000);
```


Virtual Terminal
0
0
0
0
0


Virtual Terminal
90
90
90
90
90



همانطور که در تصاویر مشخص است با بزرگتر شدن مقدار دوره ی کاری افزایش مییابد، چون تاخیرها یکسان است مدت زمانی که طول میکشد تا پالس بعدی دیده شود یکسان است در نتیجه با زیادتر شدن زاویه، دورهای که کاری انجام داده نمیشود و پالس نداریم، کاهش مییابد .