گزارش کار
تمرین کامپیوتری اول

اعضای گروه: نگین مریوانی
بهاران خاتمی
ملیکه احقاقی

پاسخ سوالات

سوال اول

زبان برنامه نویسی آردوینو استفاده از PWM را آسان می کند. به سادگی(analogWrite(pin, dutyCycle را فراخوانی کنید.به طوری که dutyCycle یک مقدار از ۰ تا ۲۵۵ است وpin یکی از پین های PWM است (۳، ۵، ۶، ۹، ۱۰، یا ۱۱). تابع analogWrite رابط کاربری ساده ای را برای PWM سخت افزاری فراهم می کند، اما هیچ کنترلی روی فرکانس را فراهم نمی کند. (توجه داشته باشید که برخلاف نام تابع، خروجی یک سیگنال دیجیتال است که اغلب به عنوان موج مربعی نامیده می شود.)

روش های دیگری جهت ساخت موج PWM در arduino وجود دارد:

Bit-banging PWM:

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delayMicroseconds(100); // Approximately 10% duty cycle @ 1KHz
  digitalWrite(13, LOW);
  delayMicroseconds(1000 - 100);
}
```

این روش به ما این اجازه را می دهد که از پین دلخواه استفاده کنیم و dutycycle را به شکل دلخواه به دست آوریم. اما در این جا این مشکل وجود دارد که هر گونه interrupt، timing مدار را تحت شعاع قرار خواهد داد.تصمیم گیری مناسب در مورد constant ها دشوار است که به فرکانس و duty cycle مطلوب بتوان رسید. هر چند دقت بالای در محاسبه ی آن به کار رود.

Using the ATmega PWM registers directly:

در این روش در مود های مختلف می توان با کمک تایمر میکروکنترلر موجود روی بورد آردوینو سیگنال PWM را تولید کرد.

سوال دوم

موتورهای DC موتور های دو سیمه هستند (power و ground) که در این نوع از موتور ها، موتور دائما در حال چرخش است(تا زمانی که power اعمال می شود).سرعت چرخش موتور به وسیله PWM کنترل میشود به طوری که به صورت سریع power صفر و یک می شود و نسبت مقدار یک بودن پالس به کل دوره تناوب که به آن duty cycle گفته میشود نسبت سرعت موتور نسبت به حداکثر مقدار آن را مشخص می کند.مثلا اگر duty cycle ٪۵۰ باشد موتور با نصف سرعت حداکثری خود می چرخد. این صفر و یک شدن انقدر سریع اتفاق میفتد که به نظر می آید موتور پیوسته در حال حرکت است.

موتور های servo موتور های سه سیمه هستند(power،ground و control). این موتورها دائماً در حال چرخش نیستند بلکه زاویه چرخش آنها که بین صفر تا ۱۸۰ درجه است به وسیله سیگنال کنترلی که مشخص کننده استیت خروجی موتور است و به وسیله مدار کنترل کننده و سنسور position مشخص میگردد.PWM به عنوان سیگنال کنترلی این موتور ها استفاده می شود. در این موتور ها طول یک بودن سیگنال مشخص کننده موقعیت شفت در این موتورهاست. بسته به نوع موتور پالس به طول مشخصی شفت موتور را در وضعیت میانی نگه می دارد و طول پالس بزرگتر یا کوچکتر منجر به چرخش موتور در جهت عقربه ساعت یا خلاف آن می گردد.

سوال سوم

استپر موتور تعداد زیادی قطب دارد؛ (معمولا ۵۰ تا ۱۰۰ قطب) . در مقایسه سروو موتور قطب های کمتری دارد (معمولا بین ۴-۱۲ قطب). برای هر قطب یک نقطه استپ برای شفت موتور وجود دارد. تعداد بیشتر قطب ها به استپر موتور اجازه می دهد تا بین هر قطب دقیق تر و صحیح تر حرکت کند، بنابراین به استپر موتور اجازه می دهد که بدون استفاده از فیدبک برای بسیاری از کاربردها مورد استفاده قرار گیرد. در مقابل، سروو موتورها اختلاف بین موقعیت کنونی انکودر و موقعیتی که به آنها فرمان حرکت به آن موقعیت داده شده است را می خوانند و جریانی که نیاز است به موقعیت دلخواه برسد ارسال می کنند. با شرایط موجود امروزه، کنترل استپر موتورها به مراتب ساده تر از سروو موتور های می باشد. ولی دقت سروو موتور های بیشتر است .برای کاربردهایی که سرعت بالا و گشتاور بالا مورد نیاز می باشد، سروو موتور فوق العاده است. سرعت استپر موتورها در محدوده ۲۰۰۰ rpm می باشد، در حالی که سروو موتورها بسیار سریع تر هستند. به اضافه سروو موتورها کم صدا هستند، برای درایوهای AC و DC موجود هستند و بخاطر رزونانس در سیستم دچار لرزش نمی شوند.

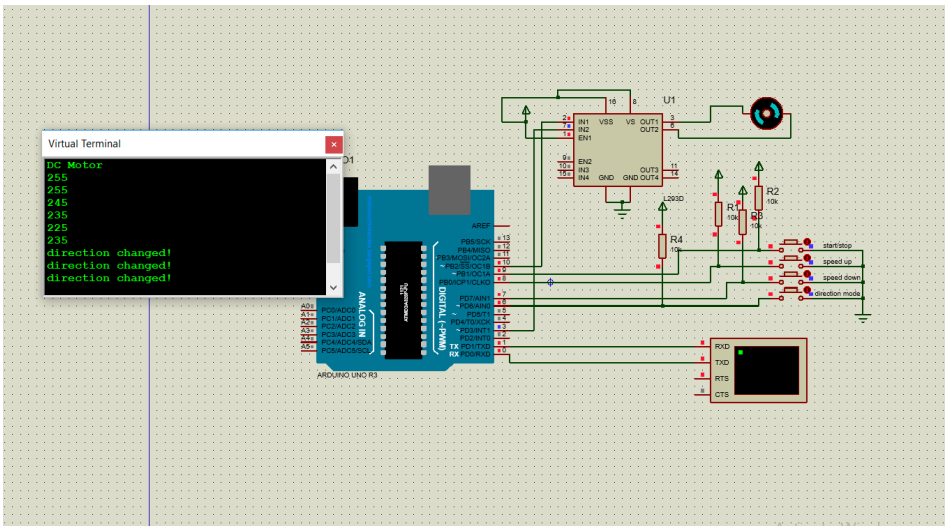
نتیجه:

سروو موتورها برای سرعت های بالا، گشتاور های بالا مناسب می باشد در حالی که استپر موتور برای شتاب های کاری پایین و گشتاورهای نگهدارنده بالا بهتر می باشد.

نمونه استفاده از servo motor : ماشین های صنعتی مثل cutting machine و CD/DVD players , packing machines

نمونه استفاده از stepper motor : اسباب بازی ها ،تلسکوپ و انتن ها

پیاده سازی موتور DC




```
DC
//poles to driver
int motorPin1 = 10;
int motorPin2 = 3;

//speed of DC motor moves controlled by PWM pulse
int curSpeed = 255;

//flags to handle functional modes
int isOn = 1;
int speedUp = 0;
int speedDown = 0;
int isClockwise = 1;

//input pins controlled by Buttons
const int stopPin = 9;
const int speedUpPin = 8;
const int speedDownPin = 7;
const int directionPin = 6;

//to handle current state of motor
int stopPinState = 1;
int speedUpPinState = 1;
int speedDownPinState = 1;
int directionPinState = 1;

//initialization
void setup()
{
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);

  pinMode(stopPin, INPUT);
  pinMode(speedUpPin, INPUT);
  pinMode(speedDownPin, INPUT);
  pinMode(directionPin, INPUT);

  curSpeed = 255;
  analogWrite(motorPin1, curSpeed);
  analogWrite(motorPin2, 0);

  Serial.begin(9600);// rate at which the arduino communicates
  Serial.println("DC Motor");
}
```

با کمک پایه های ۳ و۱۰ که پایه های PWM هستند حرکت موتور کنترل می شود. بدین جهت flag هایی تعیین شده است که مود حرکتی آن را معین می کند. بدین صورت که از ۴ پین ورودی ۶ تا ۹ مود حرکتی مشخص می شود.state ها بدین جهت استفاده می شود که تا زمانی که کلید فشرده است اثر آن اعمال نشود و در صورت رها کردن آن اثر آن اعمال شود. در بخش setup نوع پین ها که INPUT یا OUTPUT هستند مشخص شده است و مود اولیه به صورت دیفالت بر روی DC motor پیاده سازی شده است. تابع analogWrite موج PWM ای ایجاد می کند که dutyCycle آن وابسته به اختلاف پتانسیل دو پین دو سر است.

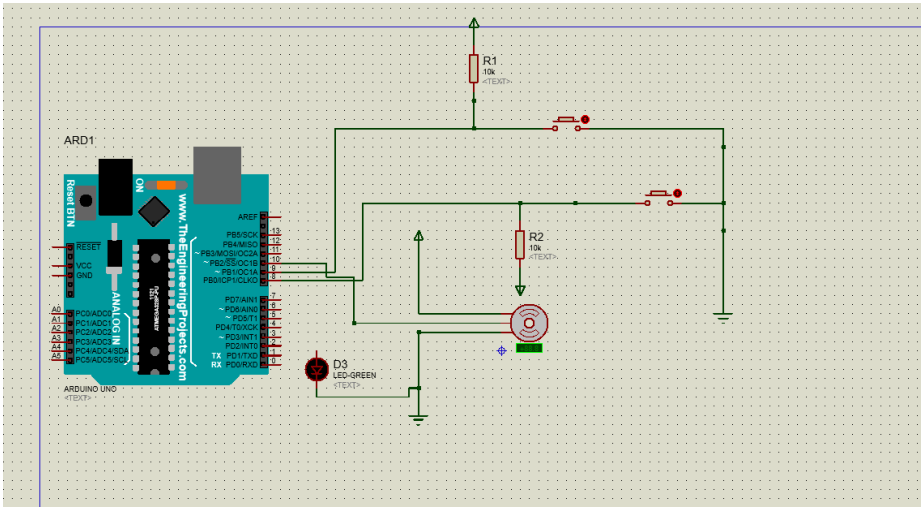
جهت دیباگ مدار از serial استفاده شده است که وضعیت حرکتی موتور را با اتصال سریال به virtual terminal بر روی آن نمایش دهد.

```
  //implementation in  main loop
  void loop()
  {
    stopPinState = digitalRead(stopPin);
    speedUpPinState = digitalRead(speedUpPin);
    speedDownPinState = digitalRead(speedDownPin);
    directionPinState = digitalRead(directionPin);

    if (stopPinState == LOW)
    {
      while(stopPinState == LOW) {
        stopPinState = digitalRead(stopPin);
      }
      isOn = 1 - isOn;
      if(isOn)
        curSpeed = 255;
      else
        curSpeed = 0;
    }
    if (speedUpPinState == LOW)
    {
      while(speedUpPinState == LOW) {
        speedUpPinState = digitalRead(speedUpPin);
      }
      speedUp = 1;
    }
    if (speedDownPinState == LOW)
    {
      while(speedDownPinState == LOW) {
        speedDownPinState = digitalRead(speedDownPin);
      }
      speedDown = 1;
    }
    if (directionPinState == LOW)
    {
      while(directionPinState == LOW) {
        directionPinState = digitalRead(directionPin);
      }
      Serial.println("direction changed!");//print on the serial
      isClockwise = 1 - isClockwise;
    }


  if(isOn)
  {
    if(speedUp)
    {
      speedUp = 0;
      curSpeed  = curSpeed + 10;
      if(curSpeed > 255)
        curSpeed = 255;
      Serial.println(curSpeed);//print on the serial
    }
    else if (speedDown)
    {
      speedDown = 0;
      curSpeed = curSpeed - 10;
      if(curSpeed < 0)
        curSpeed = 0;
      Serial.println(curSpeed);//print on the serial
    }
  }
  else
  {
    curSpeed = 0;
  }


  if(isClockwise)
  {
    analogWrite(motorPin1, curSpeed);
    analogWrite(motorPin2, 0);
  }
  else
  {

    analogWrite(motorPin1, 0);
    analogWrite(motorPin2, curSpeed);
  }

}
```

Done compiling.

به طور پیوسته در یک حلقه state کلید ها دریافت می شود و در صورت LOW بودن اثر آن ها اعمال می شود. توجه شود که در افزایش سرعت در صورت افزایش سرعت به بالاتر از 255 و یا کاهش آن به زیر ۰ مقادیر ان ها ماکسیمم و مینیمم ممکن ست می شود.
هم چنین جهت تعیین جهت حرکت کافیست عدد اعمال شده روی motorPin ها معکوس شود.

پیاده سازی موتور Servo



sketch_feb13a

```
#include <Servo.h>

Servo myservo;
int pos = 1500;
int clockwise = 8;
int anticlockwise = 9;
int deg4 = 200/9;

void setup()
{
  myservo.attach(10);
  pinMode(clockwise, INPUT);
  pinMode(anticlockwise, INPUT);
}

void loop()
{
  if(digitalRead(clockwise) == LOW)
  {
     pos = pos + deg4;
     myservo.writeMicroseconds(pos);
     while(digitalRead(clockwise) == LOW){}
  }
  if(digitalRead(anticlockwise) == LOW)
  {
    pos = pos - deg4;
    myservo.writeMicroseconds(pos);
    while(digitalRead(anticlockwise) == LOW){}
  }

}
```

در پیاده سازی این بخش از کتابخانه ی Servo.h استفاده شده است. بدین صورت که در setup اولیه موتور سروو روی پایه ی PWM شماره ۱۰ ست شده است. همچنین جهت حرکت آن توسط دو پین ساعتگرد و پادساعتگرد کنترل می شود. به ازای روشن شدن هر کلید در جهت مورد نظر موتور ۴ درجه خواهد چرخید که بدین منظور در صورت LOW شدن state آن کلید پس از رها شدن آن کلید با کمک تابع writeMicroseconds(pos)، position مطلوب ست می شود.

پیاده سازی موتور Stepper