به نام خدا

گزارش آزمایش 4

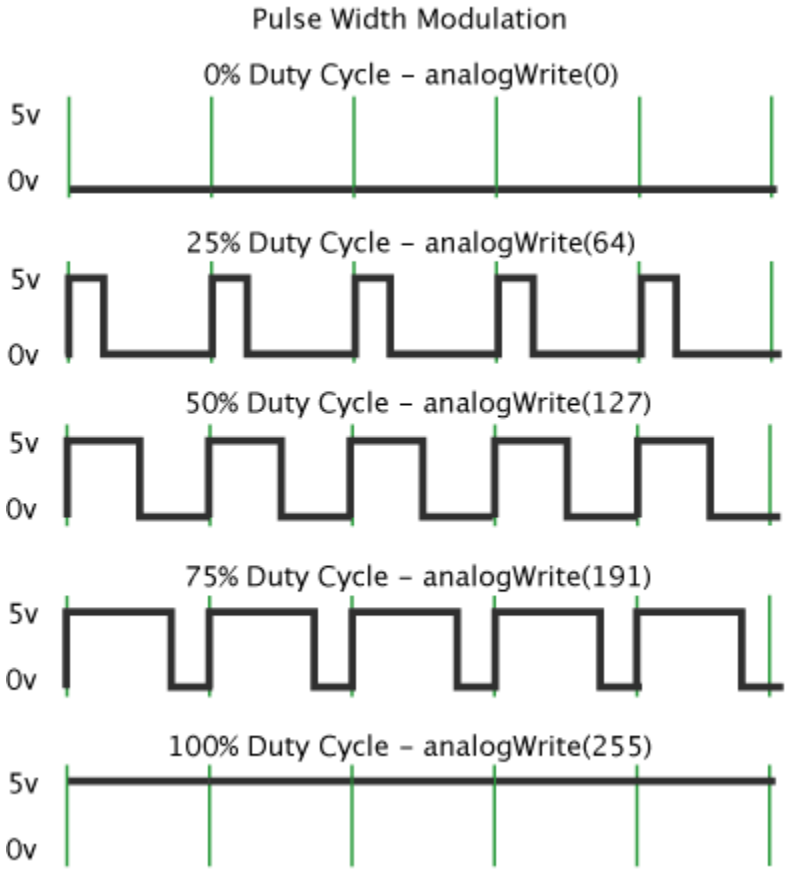
آزمایشگاه ریزپردازنده و زبان اسمبلی

محمد جواد زندیه، ابوالفضل بکیاسای کیوی

دانشگاه صنعتی امیرکبیر، دانشکده مهندسی کامپیوتر

- **مفهوم PWM و کاربرد ها**

PWM یا Pulse With Modulation فرآیندی است که برای انتقال داده آنالوگ فقط با یک پین دیجیتال انجام میگیرد. به طوری که یک دوره تناوب اصلی (Fundamental Period) در نظر گرفته شده و در هر دوره، مقدار پین از ابتدای دوره تا یک نسبت از طول دوره مساوی 1 و از آنجا تا انتهای دوره مساوی صفر قرار داده میشود. از روی اینکه در کل چه نسبتی از دوره پین مورد نظر مساوی 1 بوده، مقدار آنالوگ بدست می آید.



اینکه چه درصدی از یک دوره برابر 1 بوده در واقع Duty Cycle نامیده میشود.

- **کاربرد های سروو موتور ها**

سروو موتور ها در واقع موتور های چرخشی دقیقی هستند که در بسیاری از وسایل که نیاز به حرکت چرخشی کنترل شده دارند استفاده میشوند. همچون دوربین ها (پایه دوربین)، تلسکوپ ها، آنتن ها و ربات های کوچک و حتی ربات های بزرگ در سطح کارخانه جات صنعتی (همچون خودرو سازی).

- **توضیح در مورد ورودی آنالوگ**

برد آردوینو میتواند مقدار آنالوگ ولتاژ یک پین را دریافت کند به صورتی که برد مگا، 5 ولت را ماکسیمم در نظر گرفته و با دقت 10 بیت، میتواند مقدار 0 ولت تا 5 ولت را در بازه 0 تا 1023 جداسازی کند. تابع ()analogRead این کار را برای ما انجام میدهد، به طوری که در یک شماره پین به عنوان پارامتر گرفته و ولتاژ آنالوگ آن پین را به شکل عددی صحیح از 0 تا 1023 برمیگرداند.

- **تعریف توابع Servo.h:**

تابع attach یک پین را به عوان پین کنترلی برای موتور سروو در نظر میگیرد و میتوان برای آن حداقل و حداکثر طول پالس (duty cycle) را در واحد زمانی میکرو ثانیه مشخص کرد (مقادیر اولیه آن در صورتی که مقدار دهی نکنیم، 544 و 2400 میکرو ثانیه میباشند).

تابع write، یک مقدار درجه را گرفته و وضعیت سروو رو روی آن زاویه قرار میدهد.
تابع read، زاویه کنونی موتور را از 0 تا 180 برمیگرداند.
تابع writeMicroseconds، یک مقدار برای طول duty cycle را در واحد زمانی میکرو ثانیه گرفته و پالس ارسالی به موتور را روی آن مقدار تنظیم میکند.
تابع readMicroseconds، در document خود کتابخانه تعریف نشده ولی از روی اسمش میتوان کارایی آن را حدس زد به طوری که طول duty cycle ای که در حال حاضر به موتور ارسال میشود را در واحد زمانی میکرو ثانیه برمیگرداند.

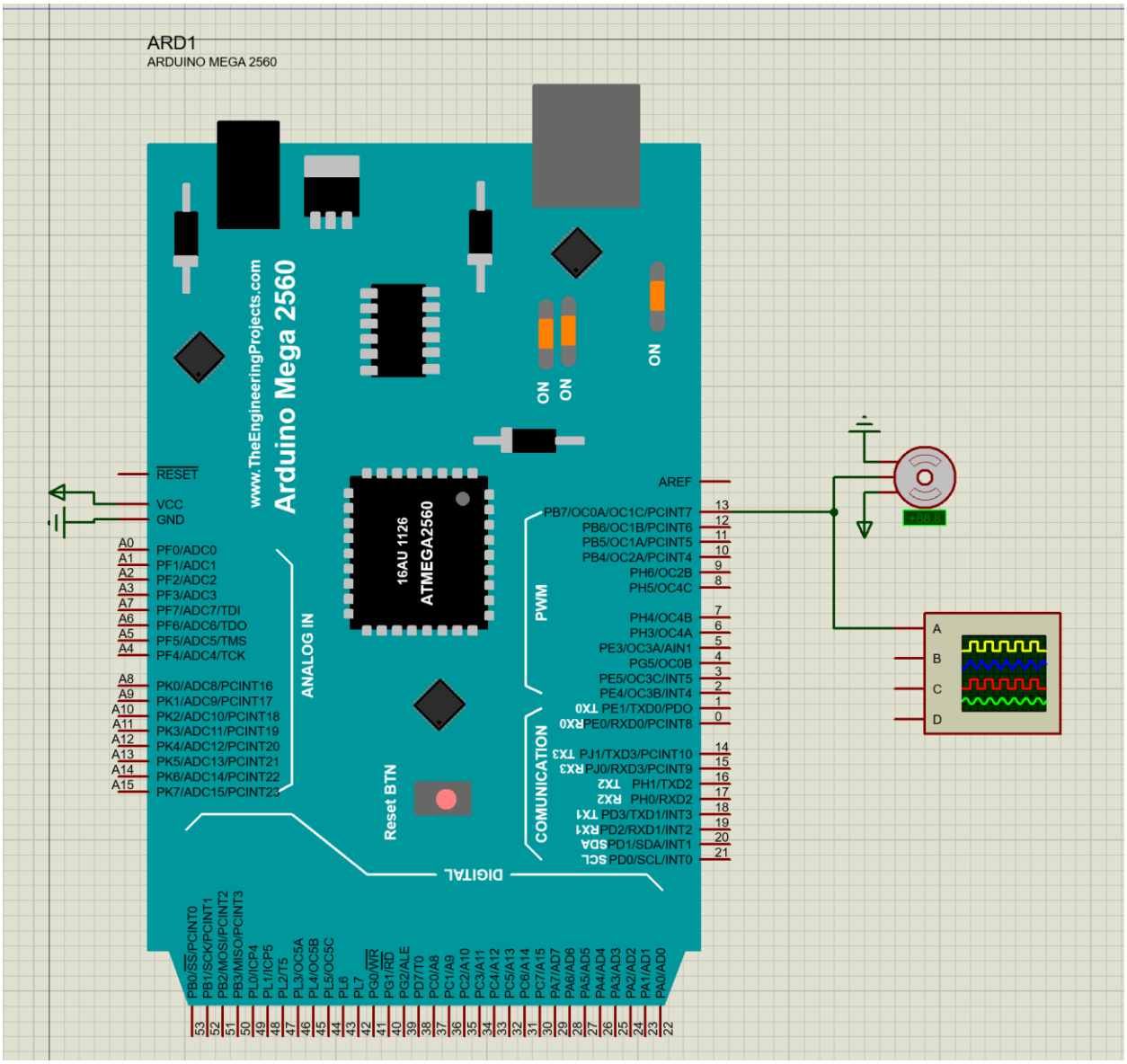
- **Stepper موتور ها**

این موتور ها تا حد زیادی همانند Servo موتور ها کار میکنند ولی تفاوت هایی وجود دارد که باعث میشود مصرف کنندگان روی انتخاب بین servo و stepper ها تامل کنند. برای مقایسه به طور خلاصه servo موتور ها برای سیستم هایی با سرعت های بالاتر، شتاب بیشتر و دقت بالا مناسب هستند و در ازای آن قیمت و پیچیدگی بیشتری نسبت به stepper ها دارند، در مقابل، stepper موتور ها برای سیستم های با سرعت و شتاب کمتر و بدون نیاز به دقت بالا مناسب هستند. چرا که اگر این موارد مورد نیاز نباشند، این موتور ها خوبی هایی مثل اندازه‌ی کوچک تر و هزینه و پیچیدگی پایین‌تری نیز دارند.

- **پرسش Fundamental Period و Duty Cycle**

این موارد در قسمت PWM به طور کامل توضیح داده شدند.

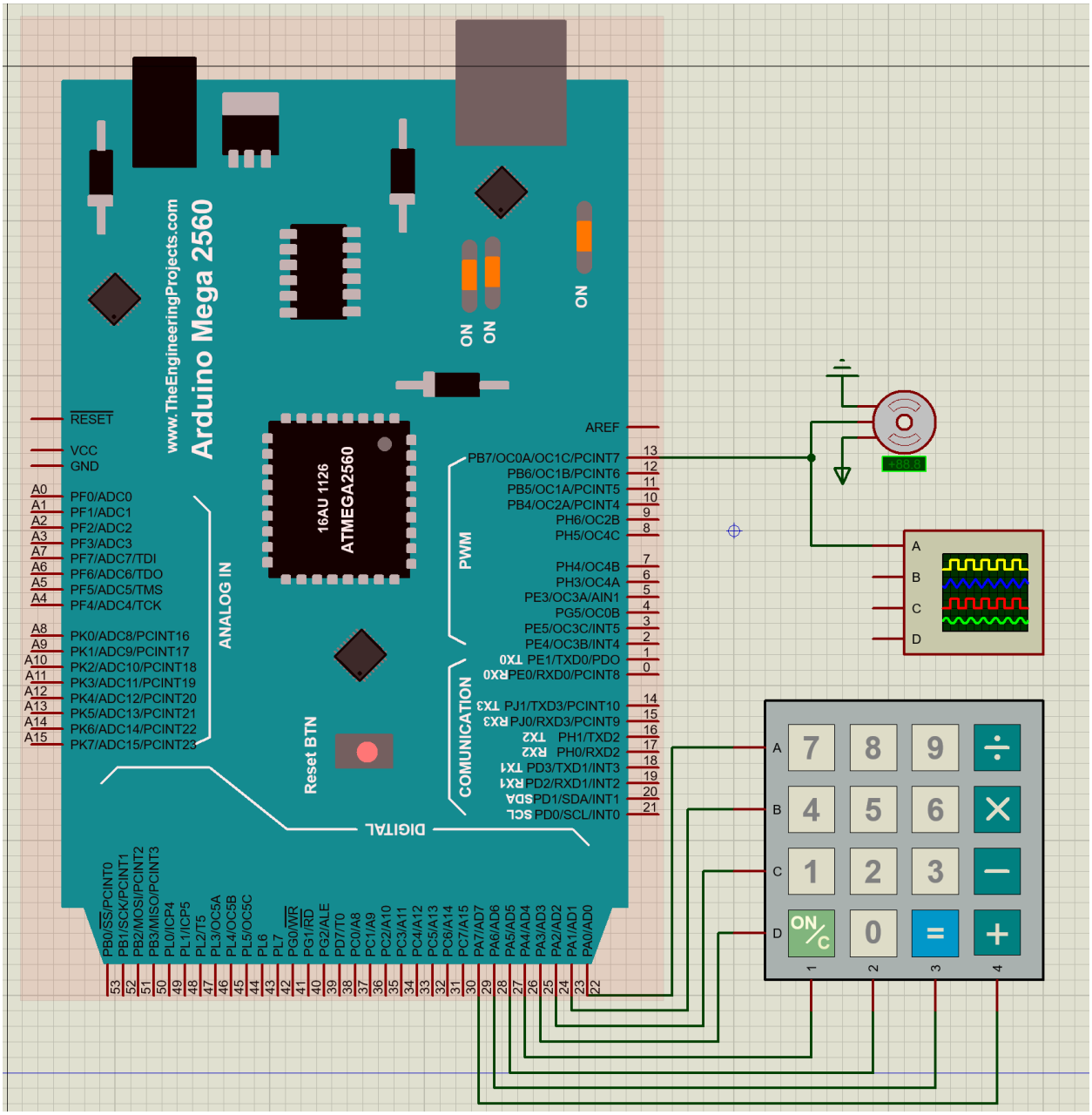
Part 1:




```
#include <Servo.h>
#define SERVO_PIN 13
Servo servo;
void setup() {
  // put your setup code here, to run once:
  servo.attach(SERVO_PIN);
}
```

```
int val(int x) {
  //return map(x, 0, 90, 93, 144);
  return map(x, -90, 90, 0, 180);
}
void loop() {
  for(int i=0;i<=90;i++) {
    servo.write(val(i));
    delay(20);
  }
  for(int i=89;i>0;i--) {
    servo.write(val(i));
    delay(20);
  }
}
```

Part 2:

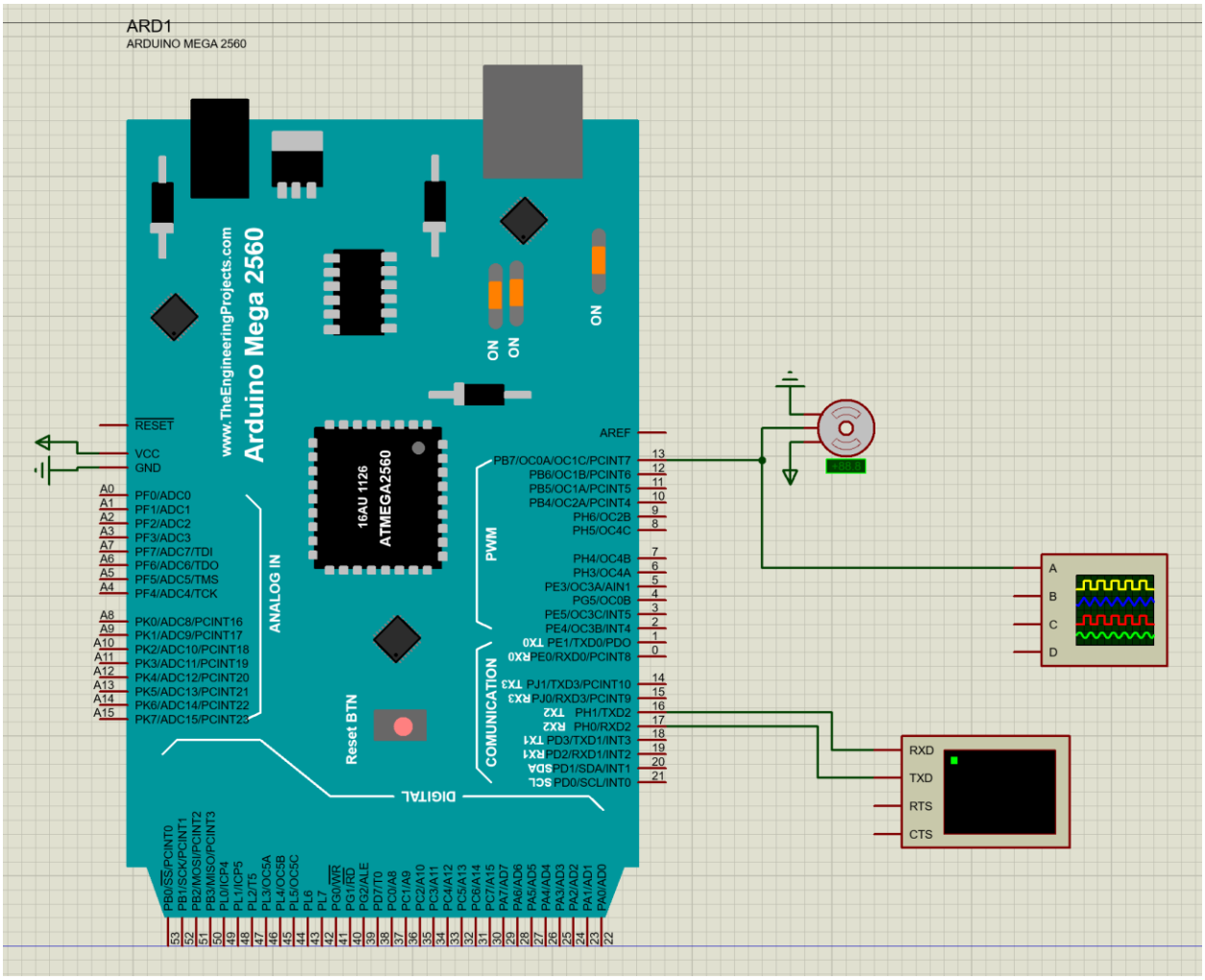



```
#include <Servo.h>
#include <Keypad.h>
#define SERVO_PIN 13
```

```
//keypad object initializing
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'7','8','9','/'},
  {'4','5','6','*'},
  {'1','2','3','-'},
  {'C','0','=','+'}
};
byte rowPins[ROWS] = {22, 23, 24, 25};
byte colPins[COLS] = {26, 27, 28, 29};

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

Servo servo;

String number_string = "";

void setup() {
  servo.attach(SERVO_PIN);
  keypad.addEventListener(keypadEvent);
}

void loop() {
  // put your main code here, to run repeatedly:
  char key = keypad.getKey();
}
```

```
void keypadEvent(KeypadEvent key) {
  if(keypad.getState() == PRESSED){
    if(key == 'C'){
      number_string = "";
    } else if((key >= '0') && (key <= '9')){
      // 0 < key < 9
      number_string += key;
      if(number_string.length() > 3)
        number_string = "";
    } else if(key == '='){
      int v = number_string.toInt();
      number_string = "";
      if(v > 360)
        return;
      int d = map(v, 0, 360, 0, 180);
      servo.write(d);
    }
  }
}
```

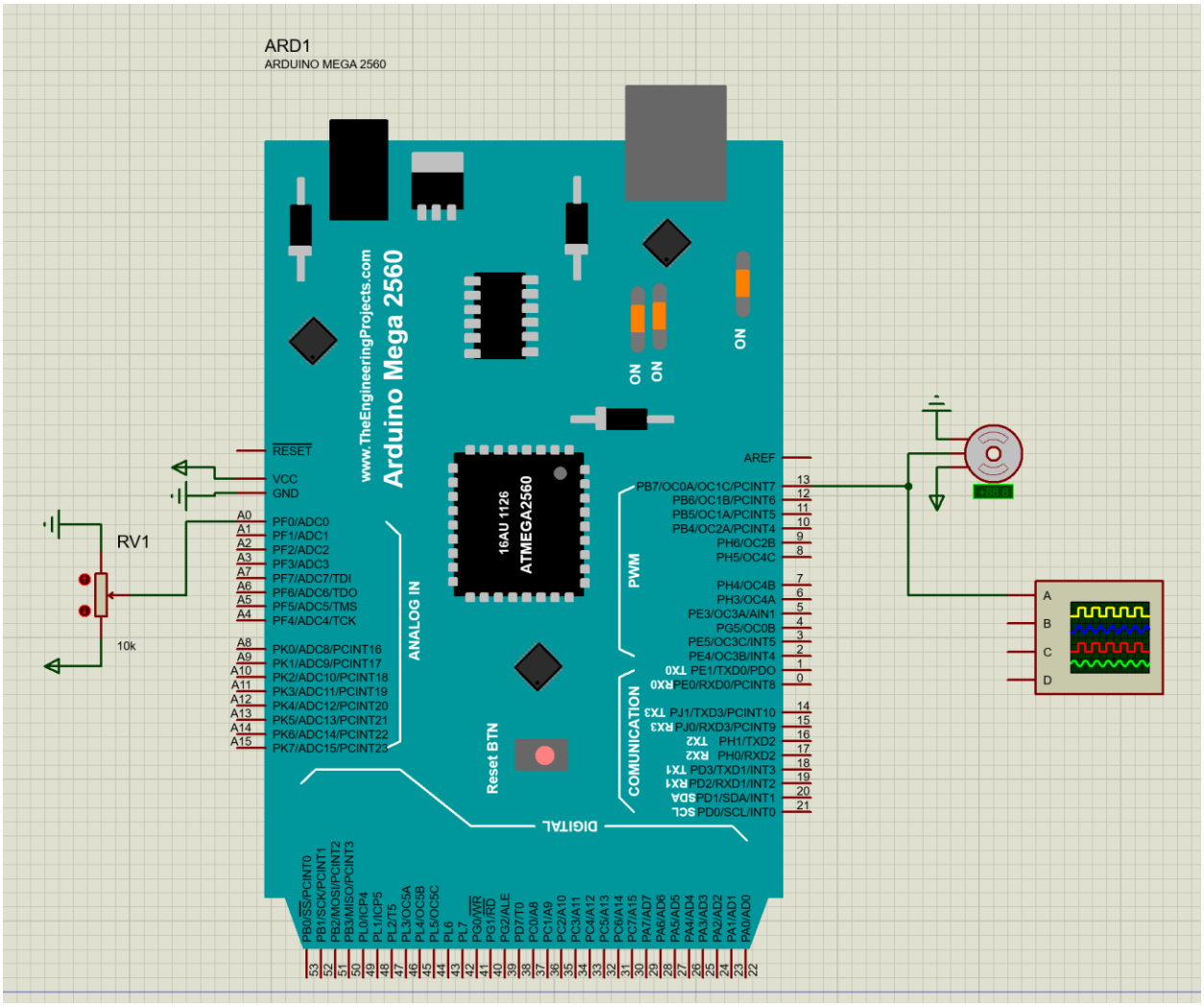
Part 3:




```
#include <Servo.h>
#define SERVO_PIN 13
Servo servo;
int deg = 180;

void setup() {
  servo.attach(SERVO_PIN);
  Serial2.begin(9600);
}
```

```
void loop(){
  if(Serial2.available() > 0) {
    int num = Serial2.parseInt();
    Serial2.println(num);
    num = (-num + 360) % 360;
    deg = (deg + num) % 360;
  }
  int t = map(deg, 0, 360, 0, 180);
  servo.write(t);
}
```

Part 4:




```
#include <Servo.h>

#define SERVO_PIN 13
#define POT_PIN A0

Servo servo;

int tmp;
```

```
void setup() {
  servo.attach(SERVO_PIN);
  pinMode(POT_PIN, INPUT);
}

void loop(){
  tmp = analogRead(POT_PIN);
  servo.write(map(tmp, 0, 1023, 0, 180));
}
```