LAB4-MC

PWM and Servo Motors

CMP(N)211

Spring 2021

Lab Objectives

- ✔ Understand PWM
- ✔ Control Servo motors
- ✔ Seven segments decoders

Recap

Till Now we have learnt the following:

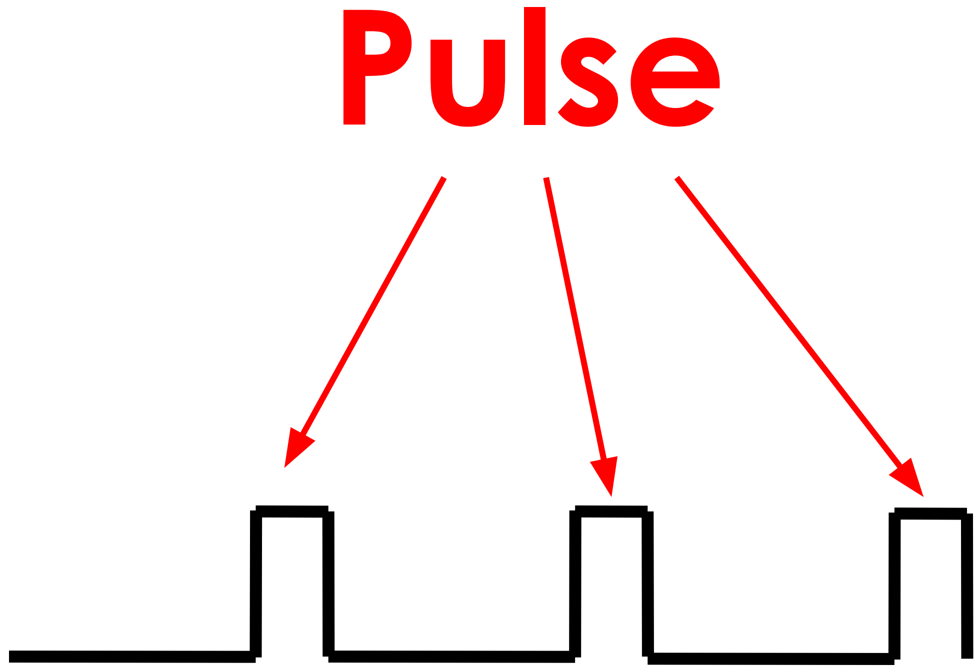
- ✔ Read analog Signals
- ✔ Read and write Digital Signals
- ✔ Control Dc motors
- ✔ Control Stepper motors
- ✔ Temperature Sensors
- ✔ Now it’s time to learn how to Write analog like signals and control the Servo motors

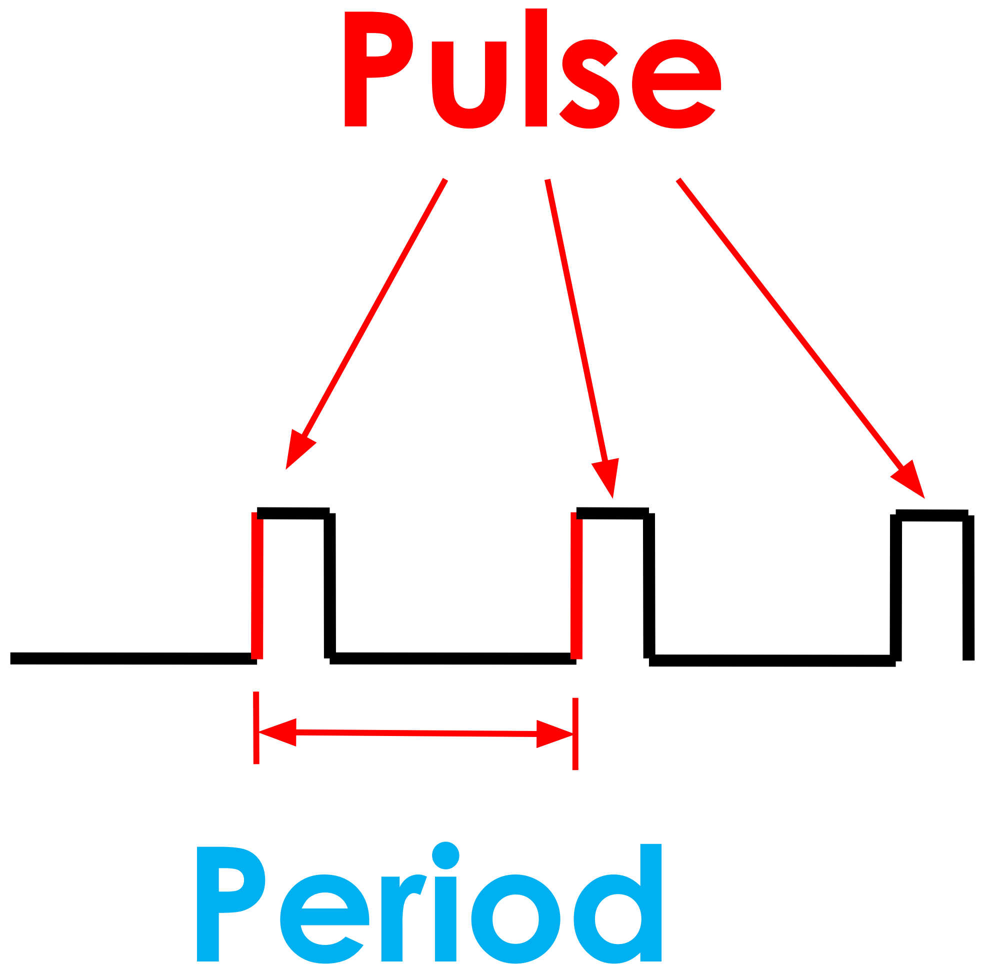
PWM

Pulse Width Modulation

Used for :

- ✔ Dimming LEDs
- ✔ Out sound
- ✔ Controlling motors

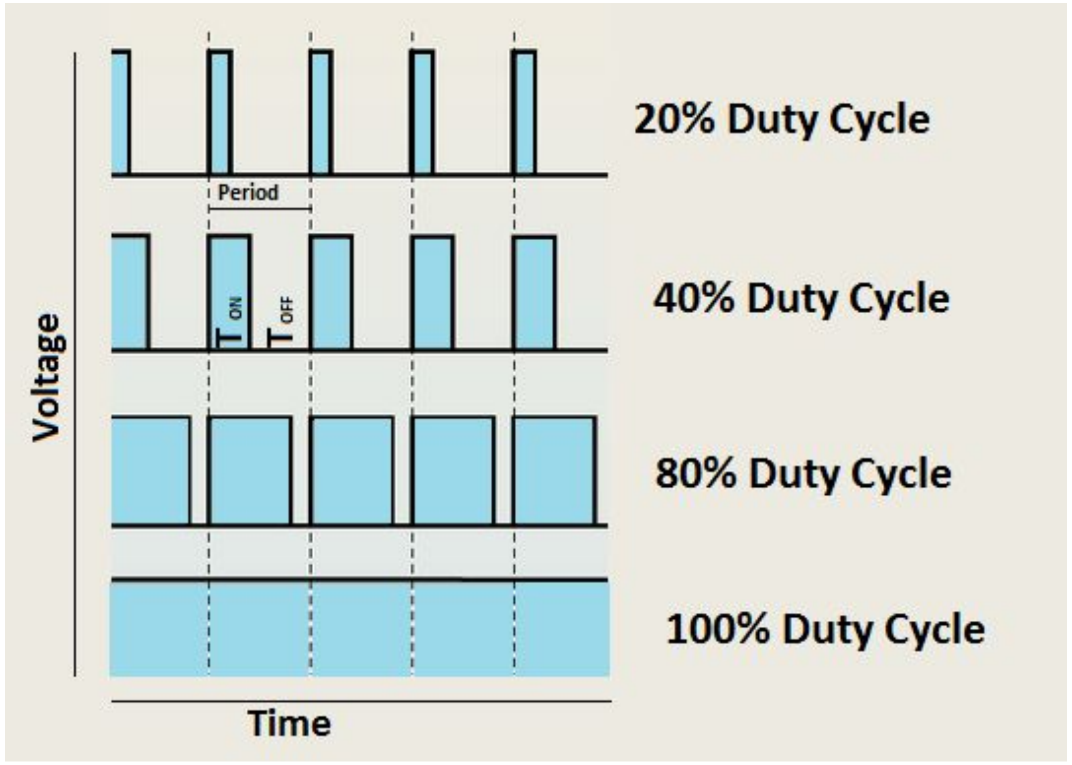

Pulse


Pulse
Period

Duty Cycle

Duty Cycle: It is the percentage of time when the signal was high during the time of period.

So at 50% duty cycle and 1Hz frequency, the led will be high for half a second and will be low for the other half second. If we increase the frequency to 50Hz (50 times ON and OFF per second), then the led will be seen glowing at half brightness by the human eye.

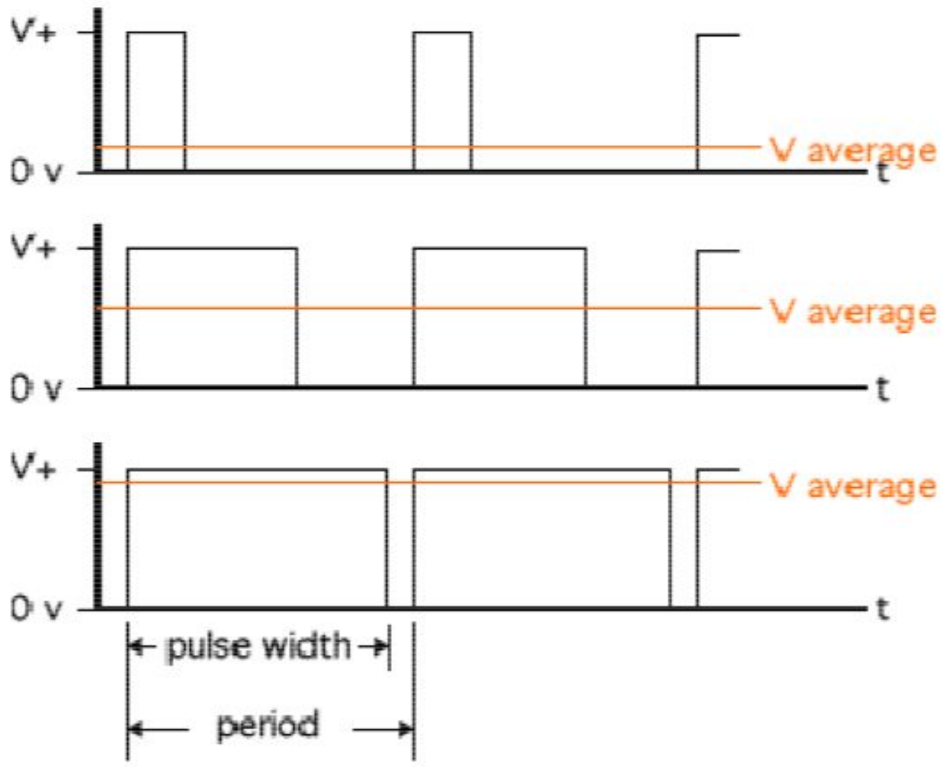


PWM

- ✔ A Pulse Width Modulation (PWM) Signal is a method for generating an analog signal using a digital source.
- ✔ Pulse width modulation is basically, a square wave with a varying high and low time.
- ✔ PWM does this by changing the pulse width, which in turn, changes the duty cycle of a square wave to alter how much power is supplied to the attached component.
- ✔ The average voltage can be calculated by taking the digital high voltage multiplied by the duty cycle

Duty Cycle x High Voltage Level = Average Voltage

PWM



Duty Cycle x High Voltage Level = Average Voltage

PWM

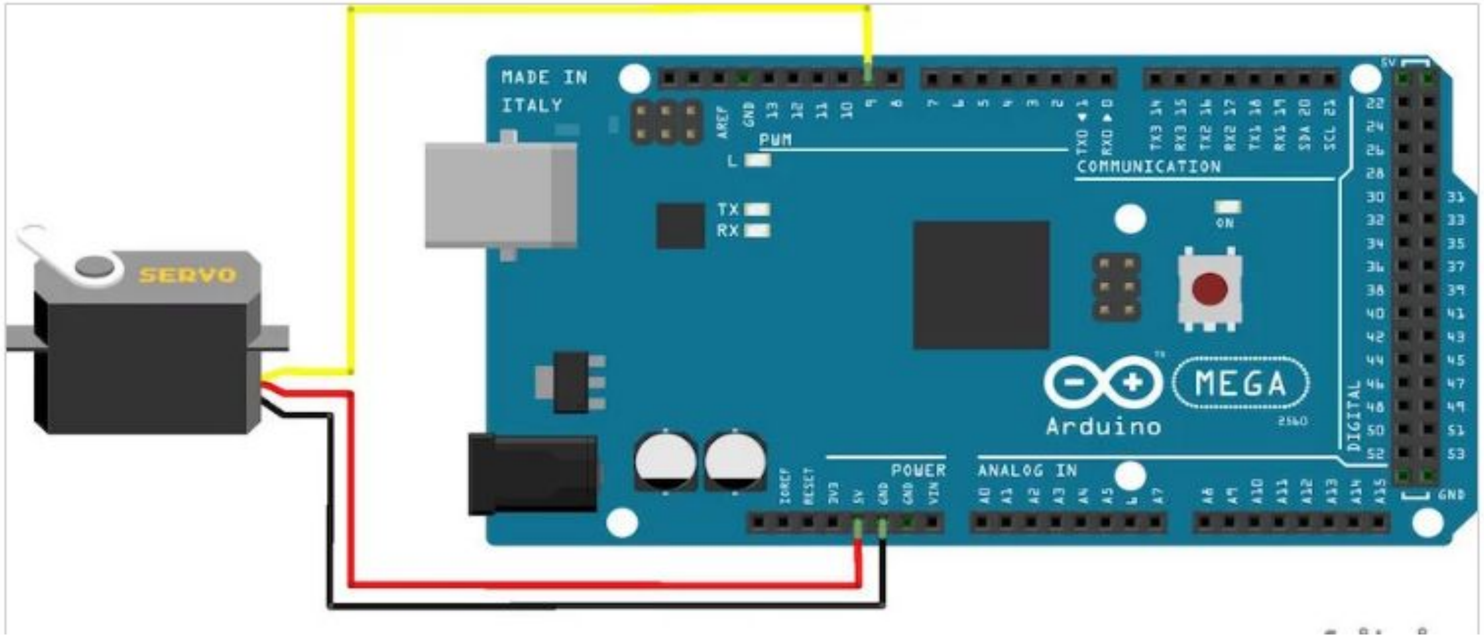
- ✔ The Arduino has six pins that can already do PWM.
- ✔ On Arduino Uno, the PWM pins are 3, 5, 6, 9, 10 and 11.
- ✔ The frequency of PWM signal on pins 5 and 6 will be about 980Hz and on other pins will be 490Hz. The PWM pins are labeled with ~ sign.
- ✔ The Arduino IDE has a built in function “analogWrite()” which can be used to generate a PWM signal.
- ✔ The function needs a pin and a value to set the duty cycle.
- ✔ The value that gets input for this function has to be between 0 and 255.
- ✔ A duty cycle of 100% occurs if the value is set at 255 and a value of 0 gives a duty cycle of 0%.

Duty cycle *255= analogwrite value

Servo Motors

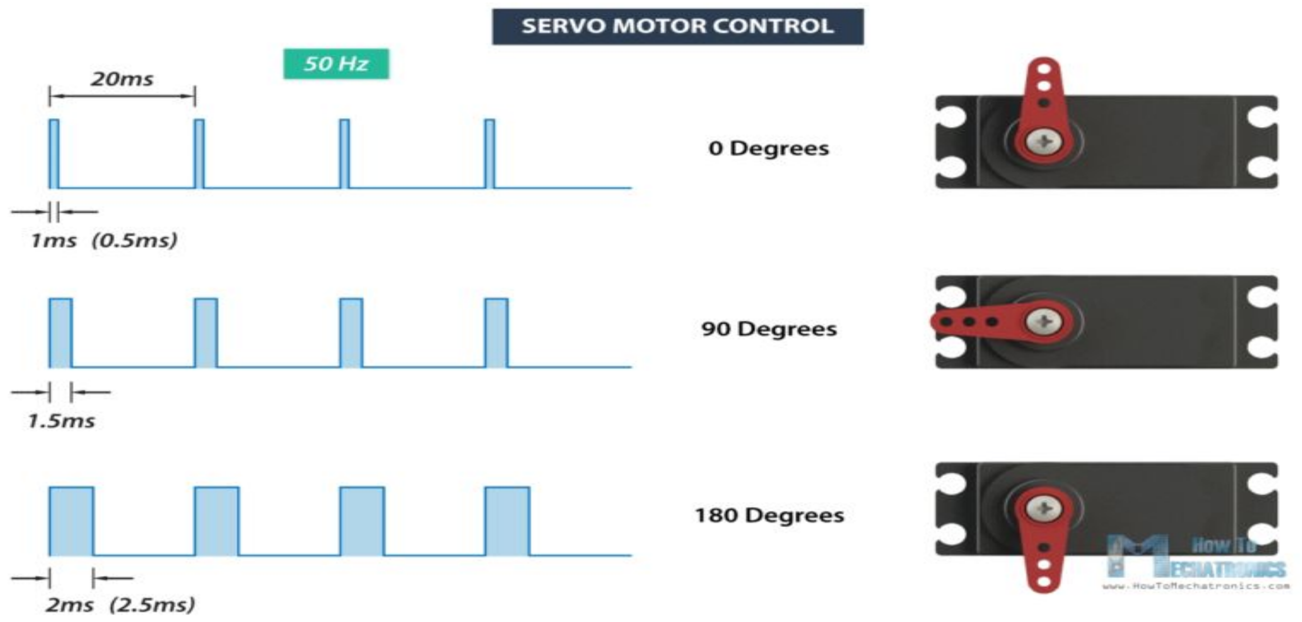
Servo Motors

- Servo red wire – 5V pin Arduino
- Servo brown wire – Ground pin Arduino
- Servo yellow wire – PWM digital pin Arduino

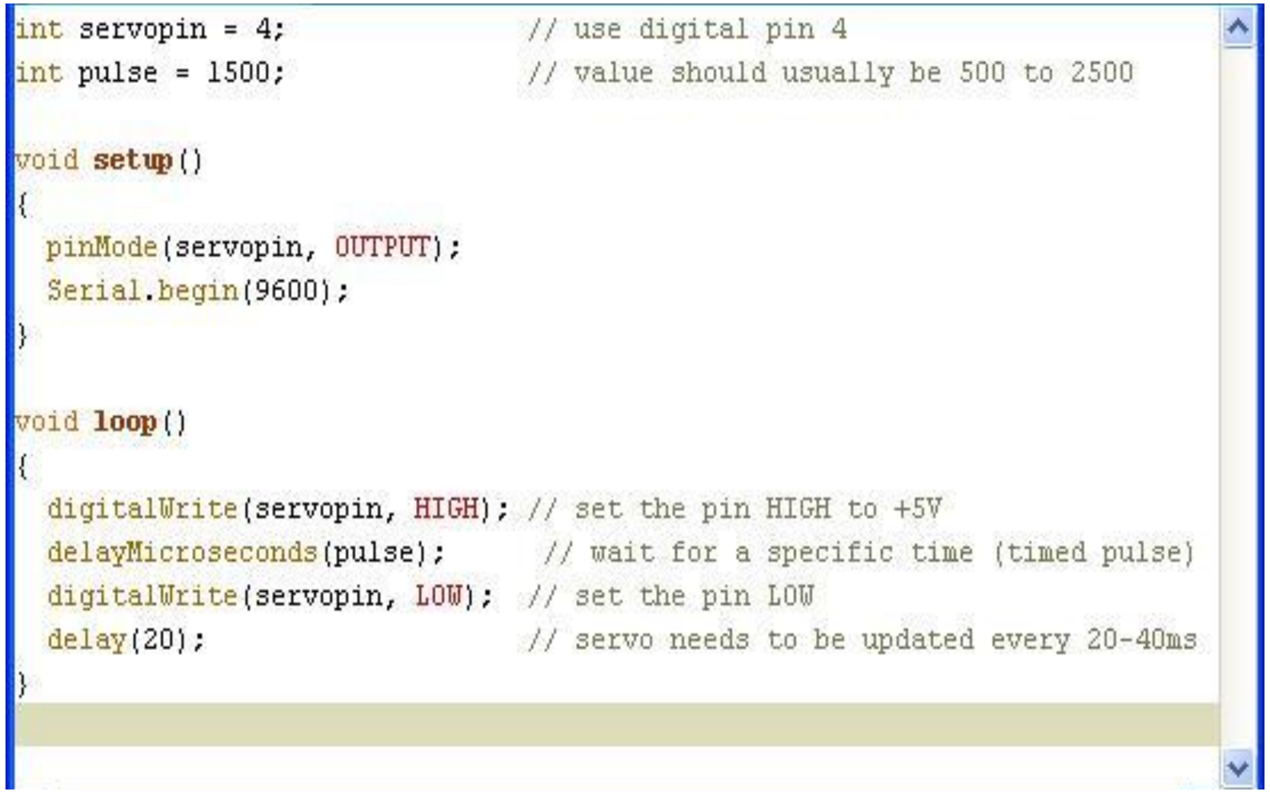


How does it work ?

- A servo motor is controlled by sending a series of pulses through the signal line, a pulse should occur every 20ms.
- The duty cycle determines angular position of the servo and it rotates up to 180 degrees Max.



You can control it by making your own pulses



```
int servopin = 4;                  // use digital pin 4
int pulse = 1500;                  // value should usually be 500 to 2500

void setup()
{
  pinMode(servopin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  digitalWrite(servopin, HIGH); // set the pin HIGH to +5V
  delayMicroseconds(pulse);       // wait for a specific time (timed pulse)
  digitalWrite(servopin, LOW);  // set the pin LOW
  delay(20);                     // servo needs to be updated every 20-40ms
}
```

Yet this is another simpler method!

Arduino Servo Library

```
// Include the Servo library
#include <Servo.h>
// Declare the Servo (PWM) pin
int servoPin = 3;
// Create a servo object
Servo Servo1;
void setup() {
   // We need to attach the servo to
the used pin number
Servo1.attach(servoPin,1000,2000);
}
```

minimum and maximum pulse width in microseconds

```
void loop(){
   // Make servo go to 0 degrees
   Servo1.write(0);
   delay(1000);
   // Make servo go to 90 degrees
   Servo1.write(90);
   delay(1000);
   // Make servo go to 180 degrees
   Servo1.write(180);
   delay(1000);
}
```

Exercise

- Use the Servo, Tact Switches, 7segments and the Serial Monitor to Perform the following requirement.

1. The Arduino gets a BCD character from **Four** switches.
2. The four switches can produce characters from [0,1,…, 9,A,..F].
3. This BCD character is printed on a 7Segments display.
4. On the same time ,The Arduino gets this BCD character as input from the digital bins.

Exercise

- Use the Servo, Tact Switches, 7segments and the Serial Monitor to Perform the following requirement.

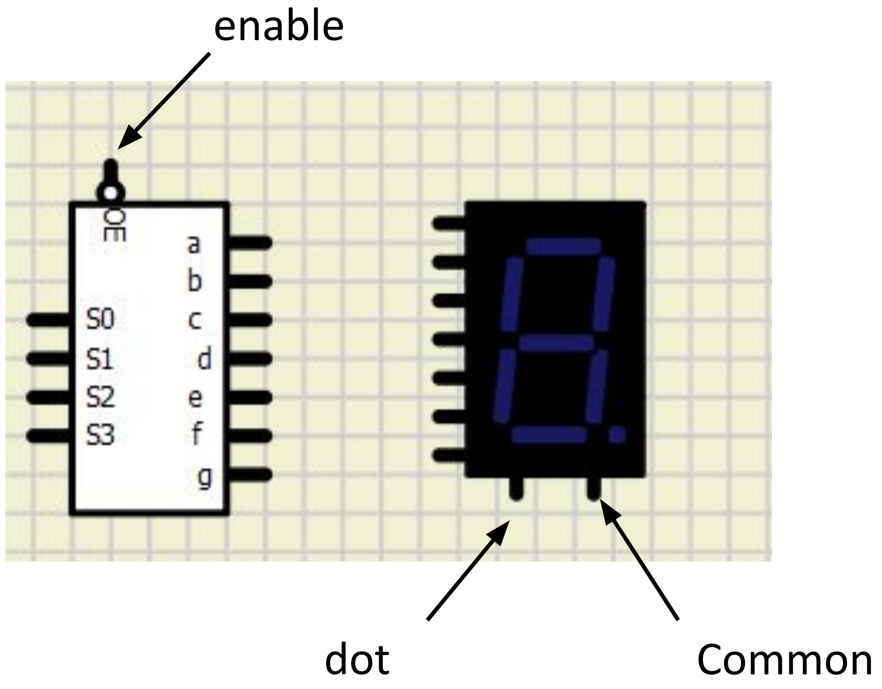
5. The Arduino should map the input digits and move the connected servo to the corresponding angle [0 ,20 ,….,180].

BCD char	**0**	**1**	**..**	**9**	**A**	**…**	**F**
Angle	0	20	….	180	180	180	180

6. The Arduino should print to the serial an error message for invalid inputs (above 9).

Exercise

- For the 7Segment display you will use single digit.
- And you can use a BCD to 7Segment decoder .
- The $\overline{\text{OE}}$ enable pin should be connected to GND to work.
- For the 7Segments if it is working in the Common Cathode Mode, then the last pin should be connected to GND as well.



Exercise

- For extra knowledge about the 7segment and it's decoder , use these two tutorials
- https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html
- https://www.electronics-tutorials.ws/combination/comb_6.html