Making a Line Follower

Analog Output

Analog Output in Arduino

- In the Arduino, analog output can only be given by the means of PWM.
- PWM stands for Pulse Width Modulation.
- PWM works by pulsating DC current, and varying the amount of time that each pulse stays ‘on’ to control the amount of current that flows to a device such as a motor or a LED.

Why do we need Analog Output

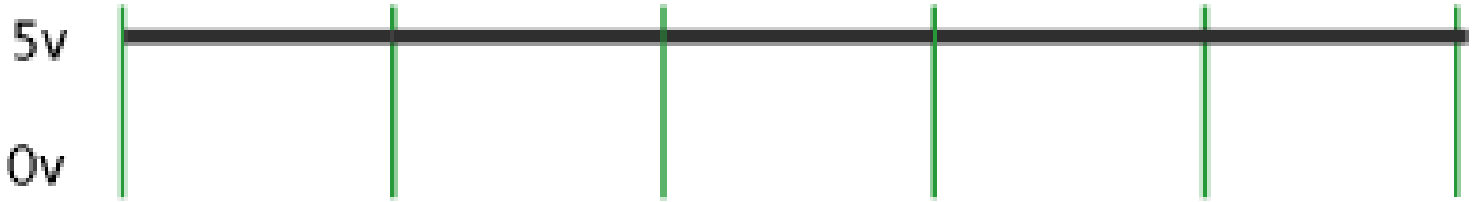
- Imagine, if your Line follower Robot has several sensors, (5-7). It is capable of detecting whether a turn is a soft turn or a sharp (hairpin-like) turn.

- For a soft turn, instead of turning off the opposite motor, if the robot reduced the speed of the opposite motor instead,

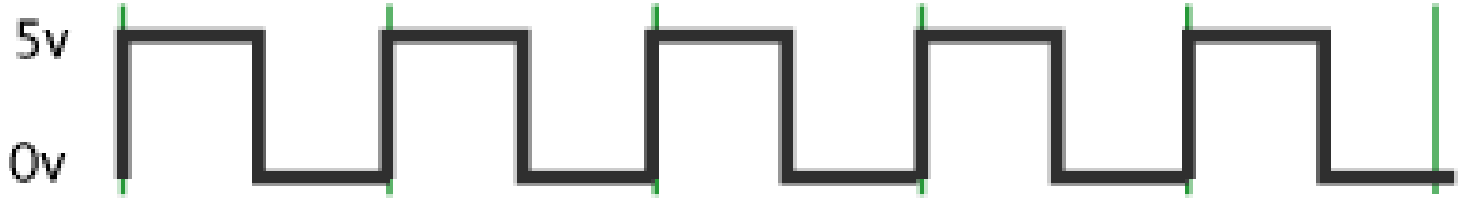
- Wouldn't the turn be taken more smoothly ?

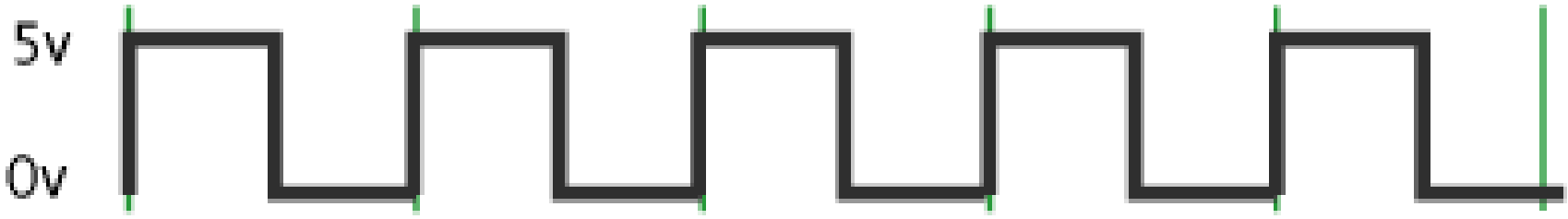
So how does PWM work ?

- Currently, ‘digitalWrite’ gives a 5V output.



- But what if I wanted to control the speed of the motor to suppose 2.5V ?’



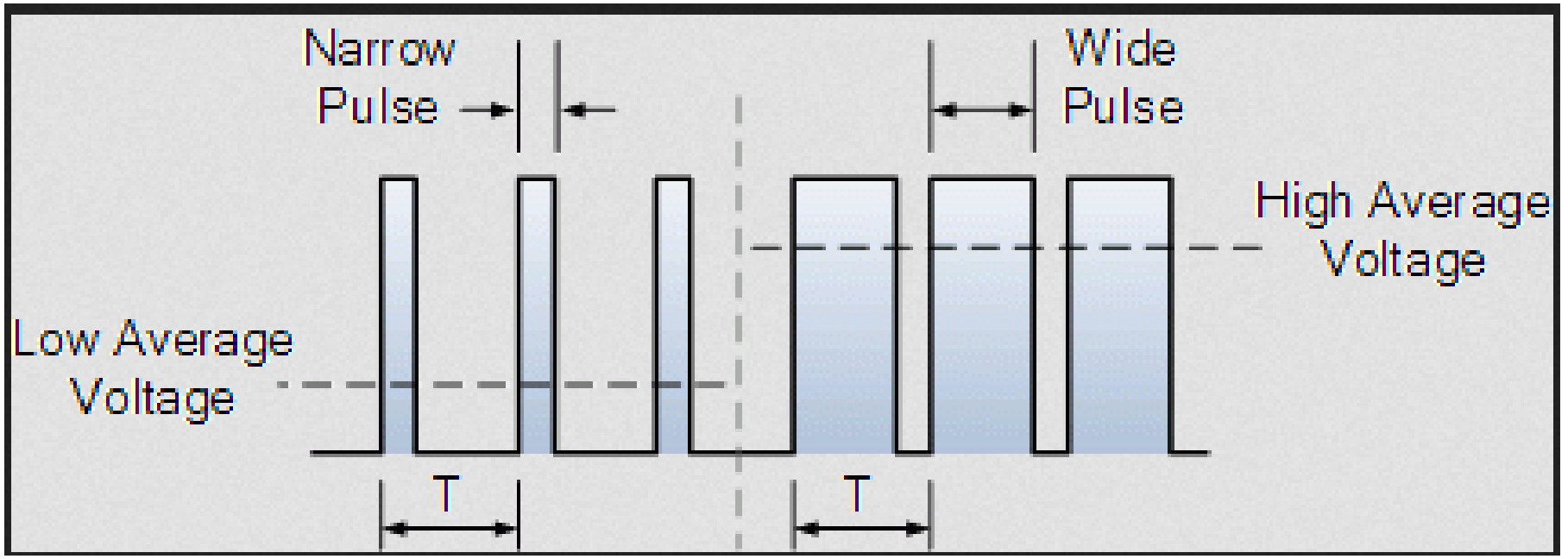


- So instead of a constant 5V, if square pulses of 5V are supplied to the motor,

- In terms of constant voltage, the motor would receive the average value.

- The net effective voltage received by the motor would only be 2.5V,

Varying Average Voltage with PWM



So, how to use PWM with Arduino?

- The register size of the Arduino is 8-bits.
- The maximum value in a 8-bit digital system would be : (0 0 0 0 0 0 0 0) (binary)
- The maximum value in a 8-bit digital system would be : (1 1 1 1 1 1 1 1) (binary)

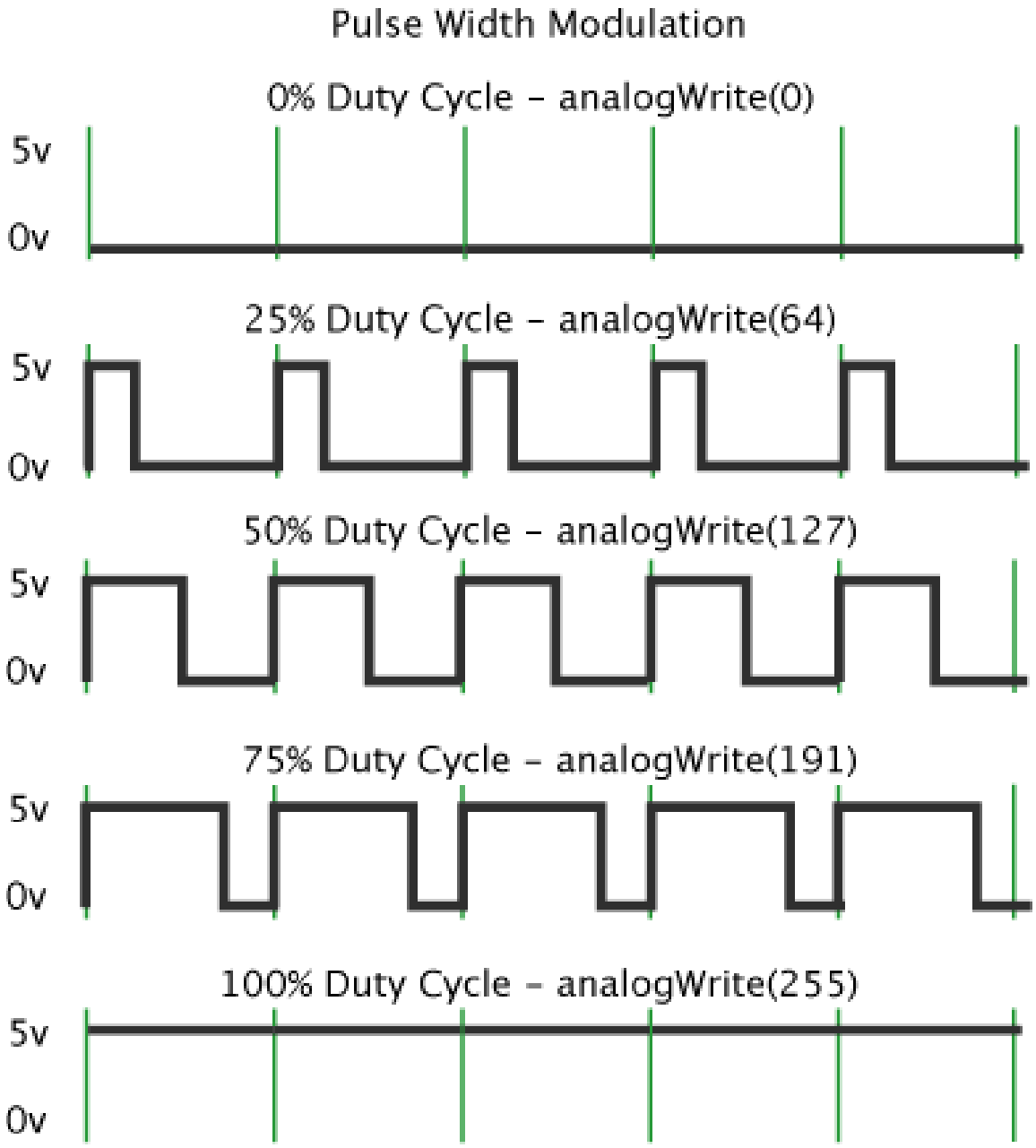
- In decimal Equivalent, the minimum and maximum values would be 0 and 255.

Hence,

- The voltage range 0-5V is divided into a range of 0-255.
- To code PWM, the following syntax is used: “analogWrite”

For example:

```
analogWrite(m1, 0);                    ( 0  Speed of Motor )
                .
                .
                .
analogWrite(m1, 255);                  ( Full Speed of Motor )
```



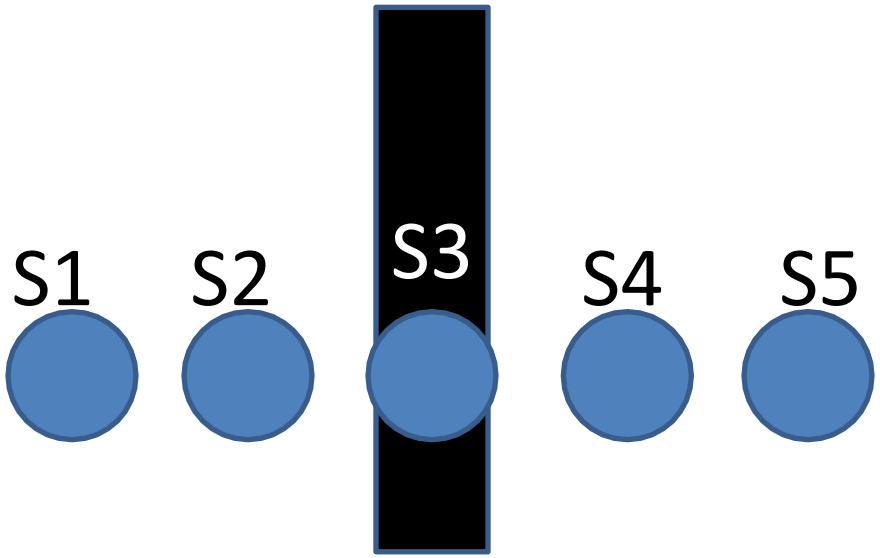
- 0 V
- 1.25 V
- 2.5 V
- 3.75 V
- 5 V

An important Note,

- The PWM pulses are only given to the Motor Driver and do not actually go to the motor directly.

- The Motor Driver Amplifies the Pulses to a range of 0-12V and then the Pulses are Supplied to the Motors.

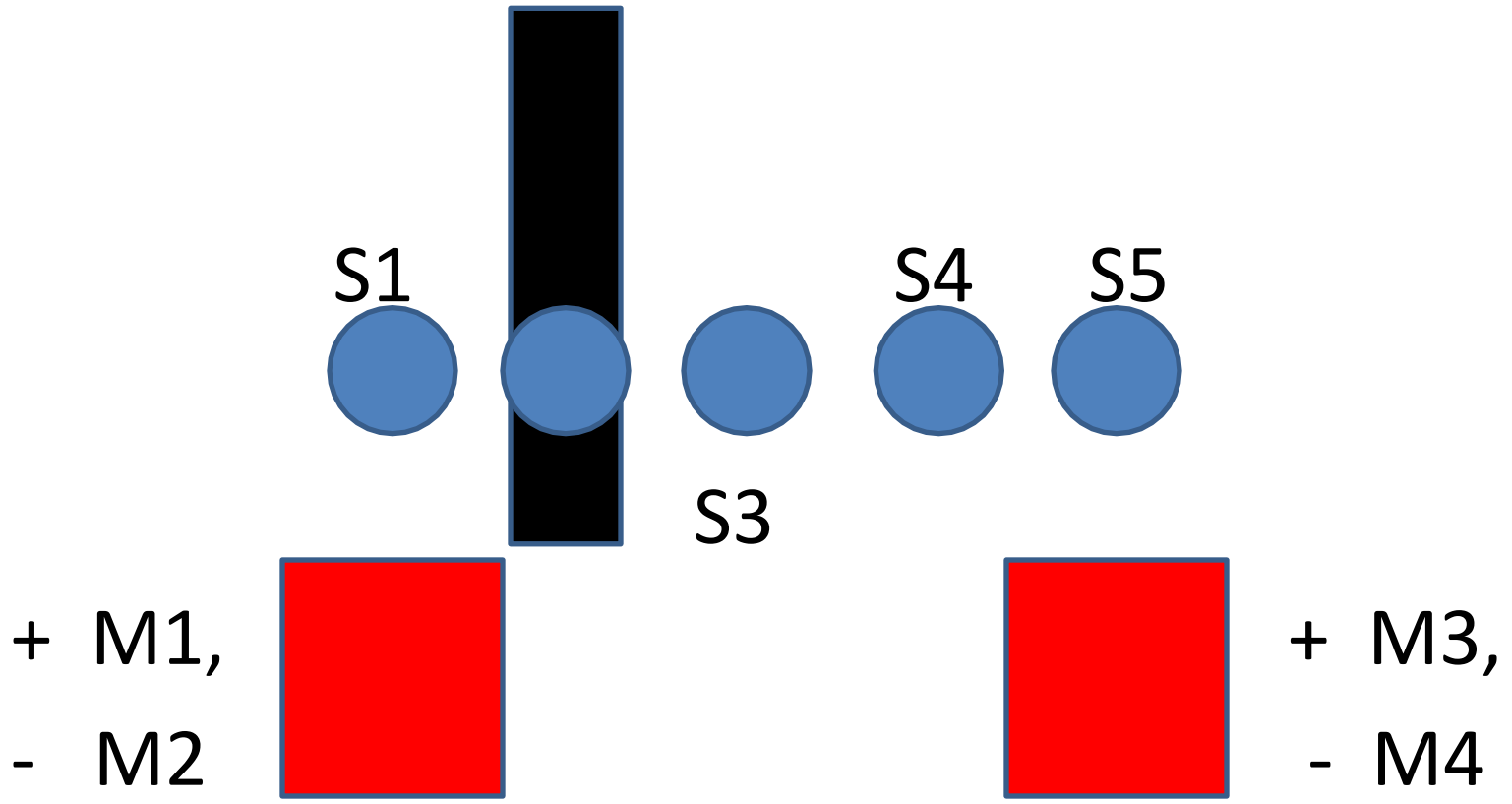
Straight Line



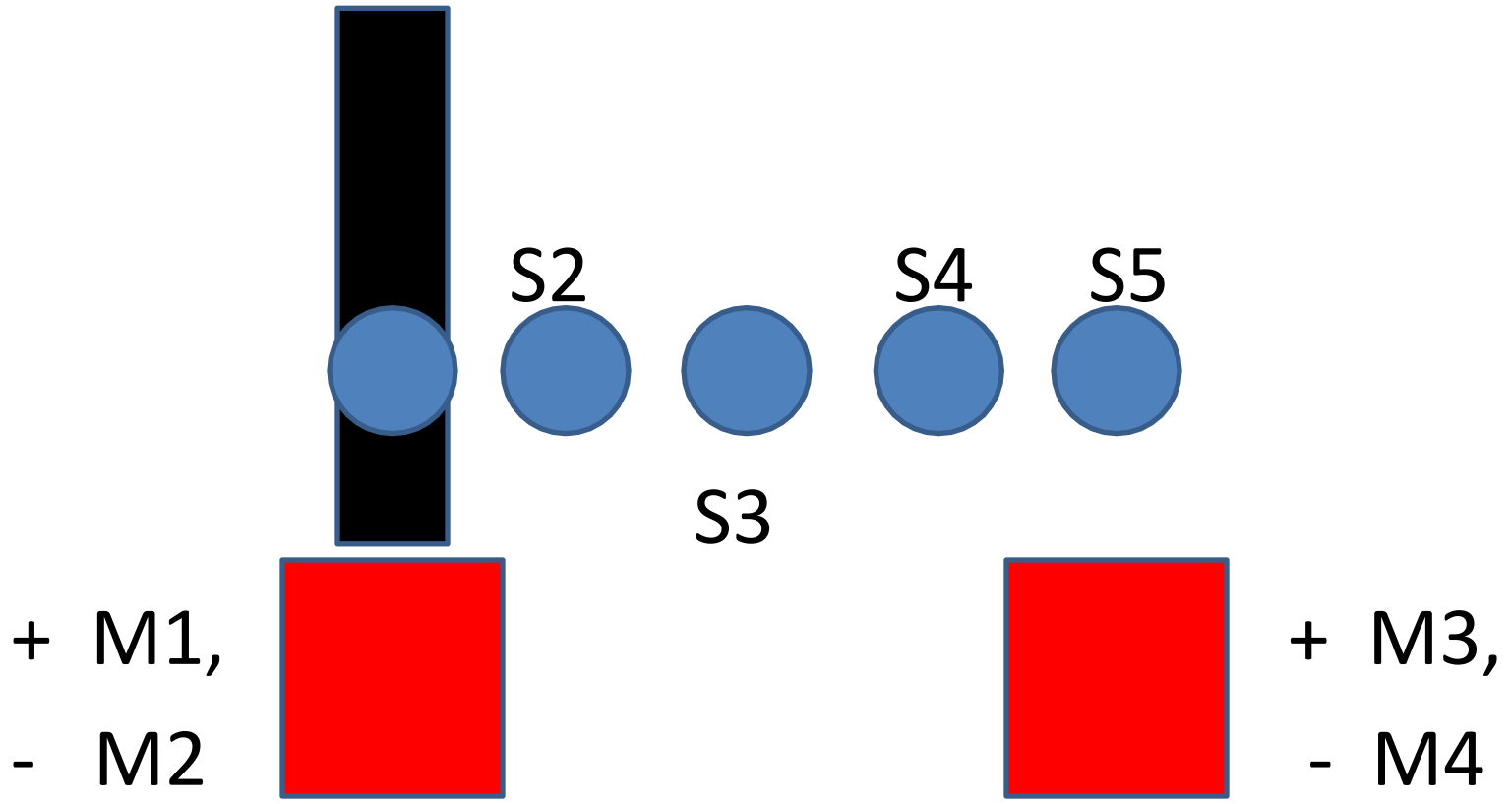
+ M1,
- M2

+ M3,
- M4

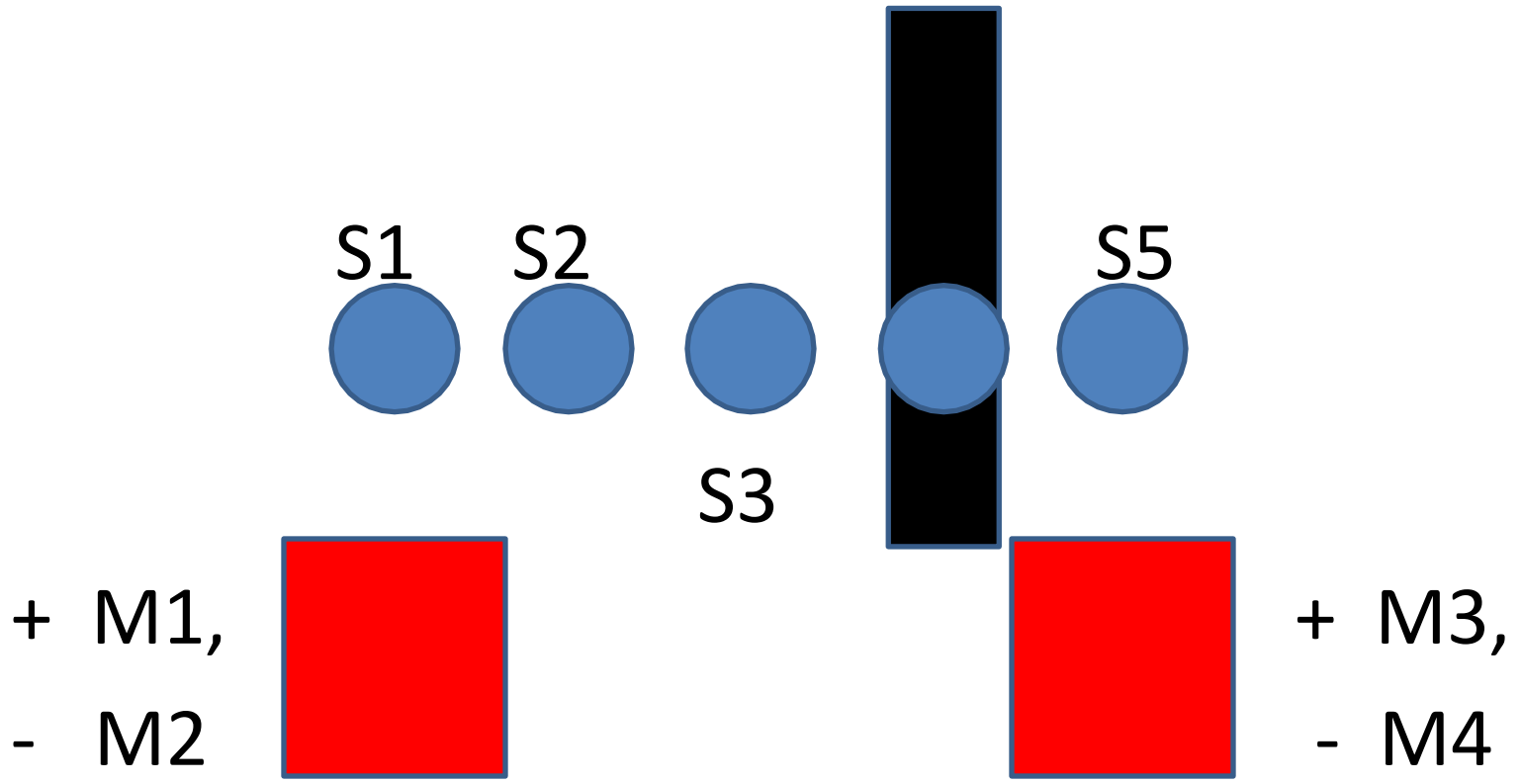
Minor Left Turn



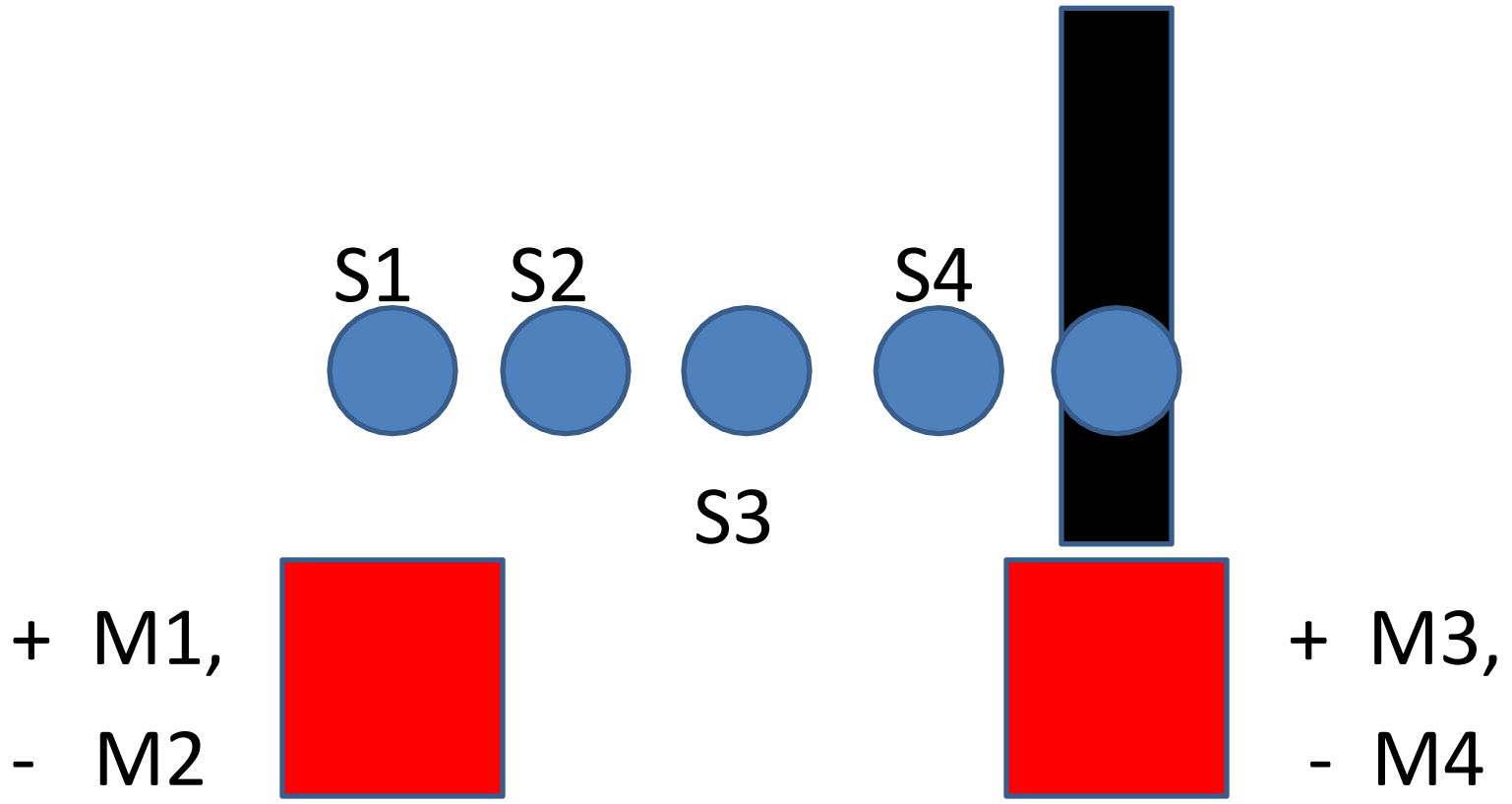
Major Left Turn



Minor Right Turn



Major Right Turn



S1 Input (Corner Left)	S2 Input (Center Left)	S3 Input (Center)	S4 Input (Center Right	S5 Input (Corner Right)	M1 Output (+)	M2 Output (-)	M3 Output (+)	M4 Output (-)
LOW	LOW	HIGH	LOW	LOW	255	0	255	0
LOW	HIGH	LOW	LOW	LOW	~100	LOW	255	LOW
HIGH	LOW	LOW	LOW	LOW	LOW	~100	255	LOW
LOW	LOW	LOW	HIGH	LOW	255	LOW	~100	LOW
LOW	LOW	LOW	LOW	HIGH	255	LOW	LOW	~100

Variable Declaration

```
int s1 = 2, s2 = 3, s3 = 4, s4 = 5, s5 = 7;
int m1 = 6, m2 = 9, m3 = 10, m4 = 11;
int a,b,c,d,e;
```

Void Setup

```
pinMode(s1, INPUT);
pinMode(s2, INPUT);
pinMode(s3, INPUT);
pinMode(s4, INPUT);
pinMode(s5, INPUT);

pinMode(m1, OUTPUT);
pinMode(m2, OUTPUT);
pinMode(m3, OUTPUT);
pinMode(m4, OUTPUT);
```

Void Loop

```
Void loop()
{
a = digitalRead(s1);
b = digitalRead(s2);
c = digitalRead(s3);
d = digitalRead(s4);
e = digitalRead(s5);

If ( a == LOW && b == LOW  &&  c == HIGH  &&  d == LOW  &&  e == LOW)
{
 analogWrite(m1, 255);
 analogWrite(m2, 0);
 analogWrite(m3, 255);
 analogWrite(m4, 0);
}
```

Contd….

```
If ( a == LOW && b == HIGH  &&  c == LOW  &&  d == LOW  &&  e == LOW)
{
 analogWrite(m1, 100);
 analogWrite(m2, 0);
 analogWrite(m3, 255);
 analogWrite(m4, 0);
}
If ( a == HIGH && b == LOW  &&  c == LOW  &&  d == LOW  &&  e == LOW)
{
 analogWrite(m1, 0);
 analogWrite(m2, 100);
 analogWrite(m3, 255);
 analogWrite(m4, 0);
}
```

```
If ( a == LOW && b == LOW  &&  c == LOW &&  d == HIGH &&  e == LOW)
{
 analogWrite(m1, 255);
 analogWrite(m2, 0);
 analogWrite(m3, 100);
 analogWrite(m4, 0);
}
If ( a == LOW && b == LOW  &&  c == LOW  &&  d == LOW  &&  e == HIGH)
{
 analogWrite(m1, 255);
 analogWrite(m2, 0);
 analogWrite(m3, 0);
 analogWrite(m4, 100);
}
```