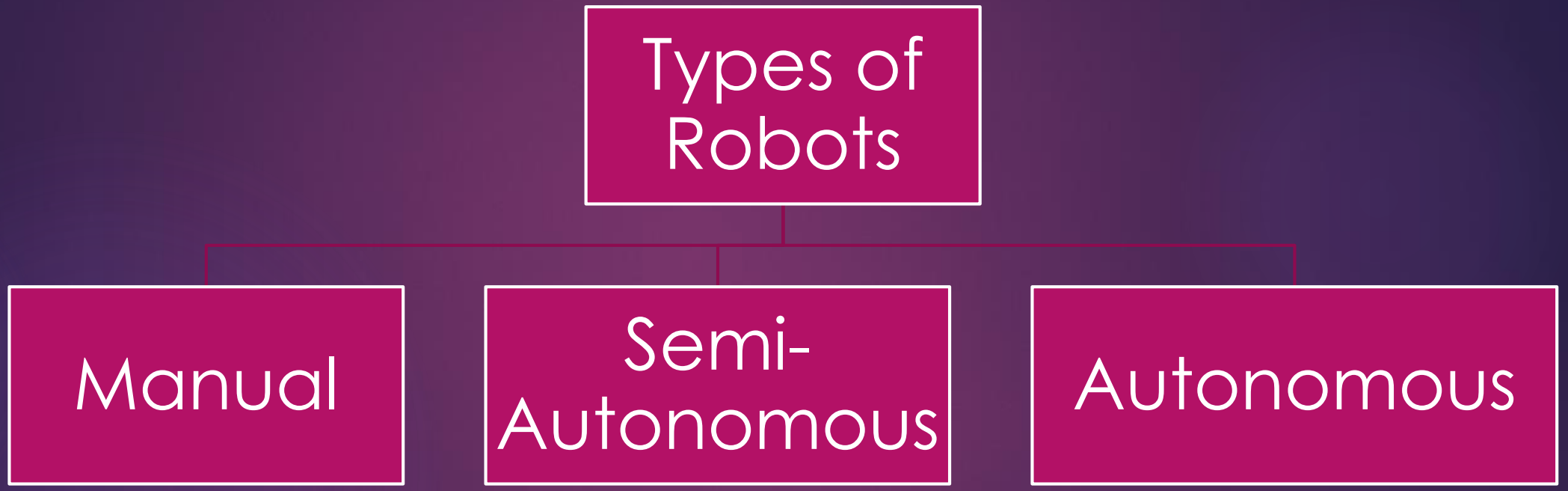
MAZE EXPLORER

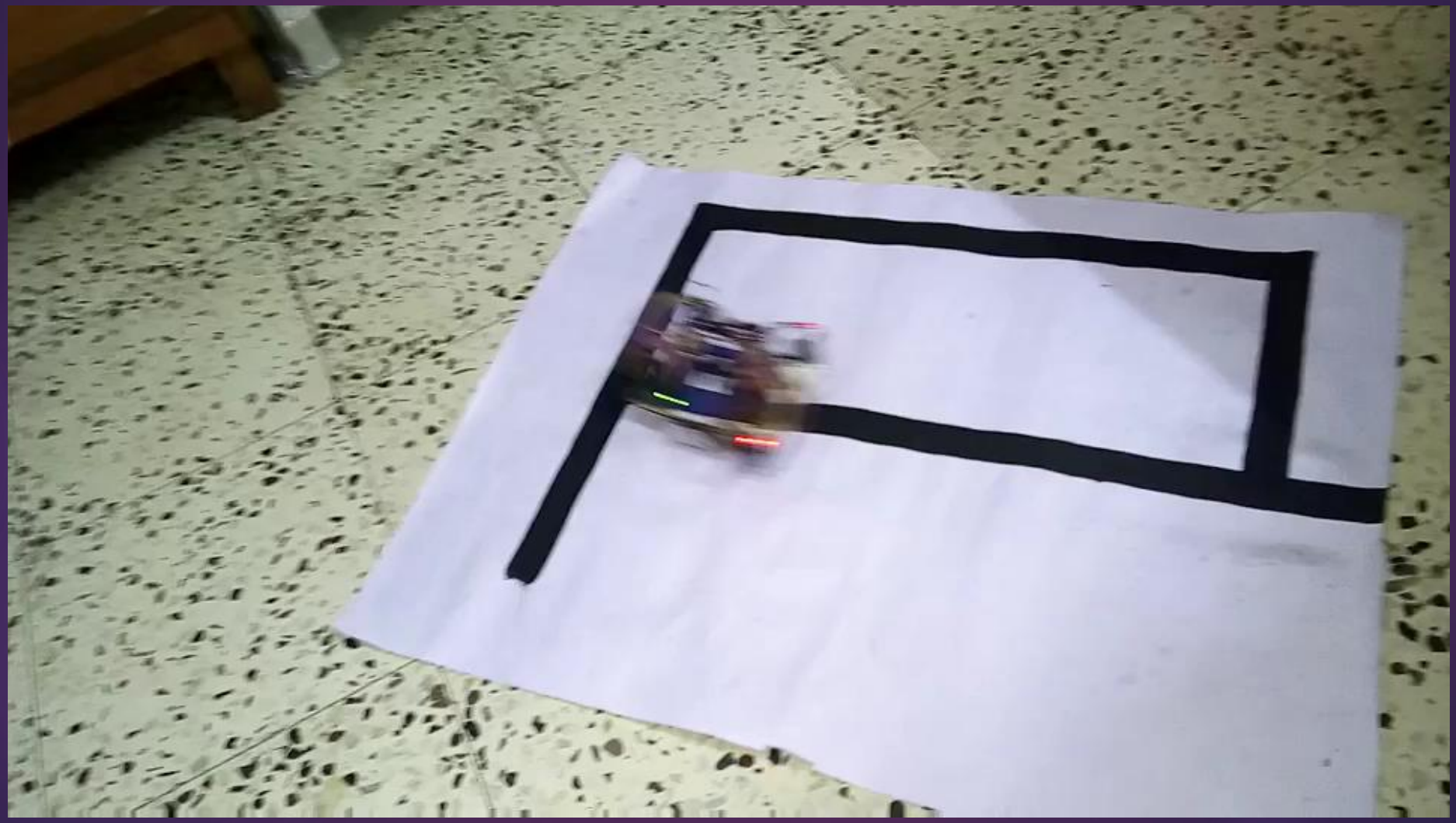

Types of Robots
Manual
Semi-Autonomous
Autonomous

Autonomous Robots

- Autonomous robots are intelligent machines capable of performing tasks in the world by themselves, without explicit human control.
- Obstacle avoider robot, Line follower robots, Maze Solver robots, shortest path finder robots.

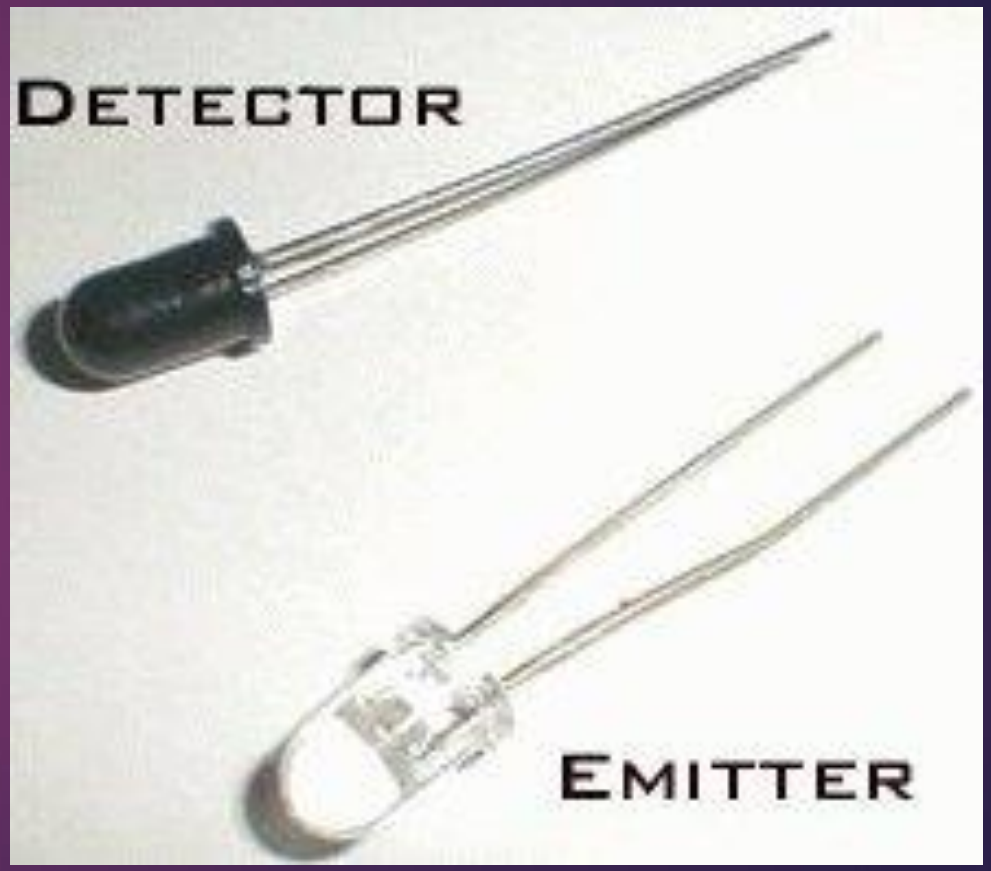
  Example: Roomba, ASIMO

Line Follower



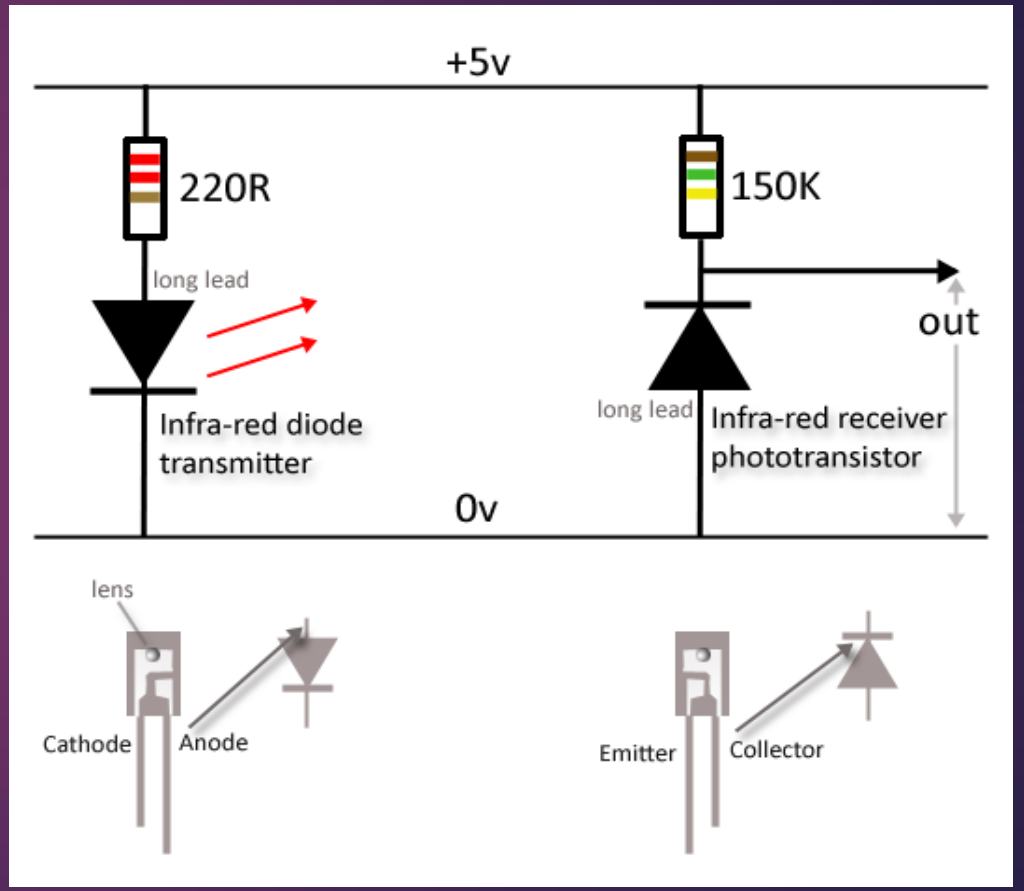
IR Receiver & Transmitter

- The IR transmitter sends an IR radiation(in the infrared wavelength region),which is reflected of a surface and falls upon a receiver.
- Due to the falling of light on the receiver a potential difference is created across the ends.
- This potential difference is recognized by a microcontroller.



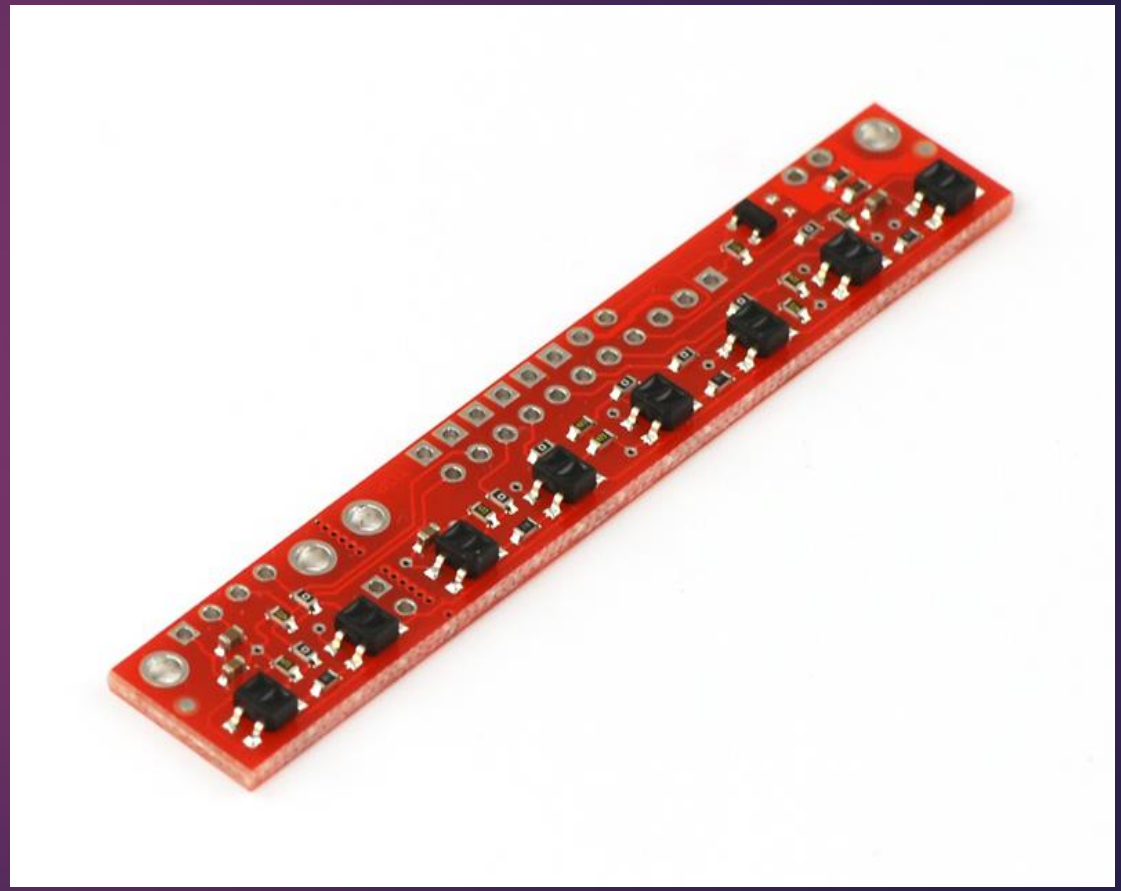
IR Receiver & Transmitter

- Transmitter (IR LED): Forward Biased (Anode (Longer Terminal to Positive))
- Receiver (Photo Diode): Reverse Biased (Cathode (Shorter Terminal to Positive))
- Receiver acts as a variable resistance
- Resistance decreases when IR light falls on it.



Digital Sensors

- QTR Reflectance sensor arrays
- Digital I/O-measurable output
- No analog-to-digital converter (ADC) is required
- Auto Calibration possible using libraries



analogRead()

- Syntax: analogRead(pin_number);
- Data type: Returns an int value;
- Example Code: int a = analogRead(A0);
- If a voltage of 2.5 V is being received at A0, then it will be converted to it's digital value **511** (2.5 * 2^10 / 5) and analogRead will return 511 and store it in **a**

analogWrite()

- Syntax: analogWrite(pin_number, ADC value);
- Purpose: To give a voltage other than 5V
- Example Code: analogWrite(A0, 127);
- If we want to give a voltage of 2.5V at pin A0 then we use the above command in the given manner. Also used to give digital PWM (Pulse Width Modulation).

Line Following in Arduino

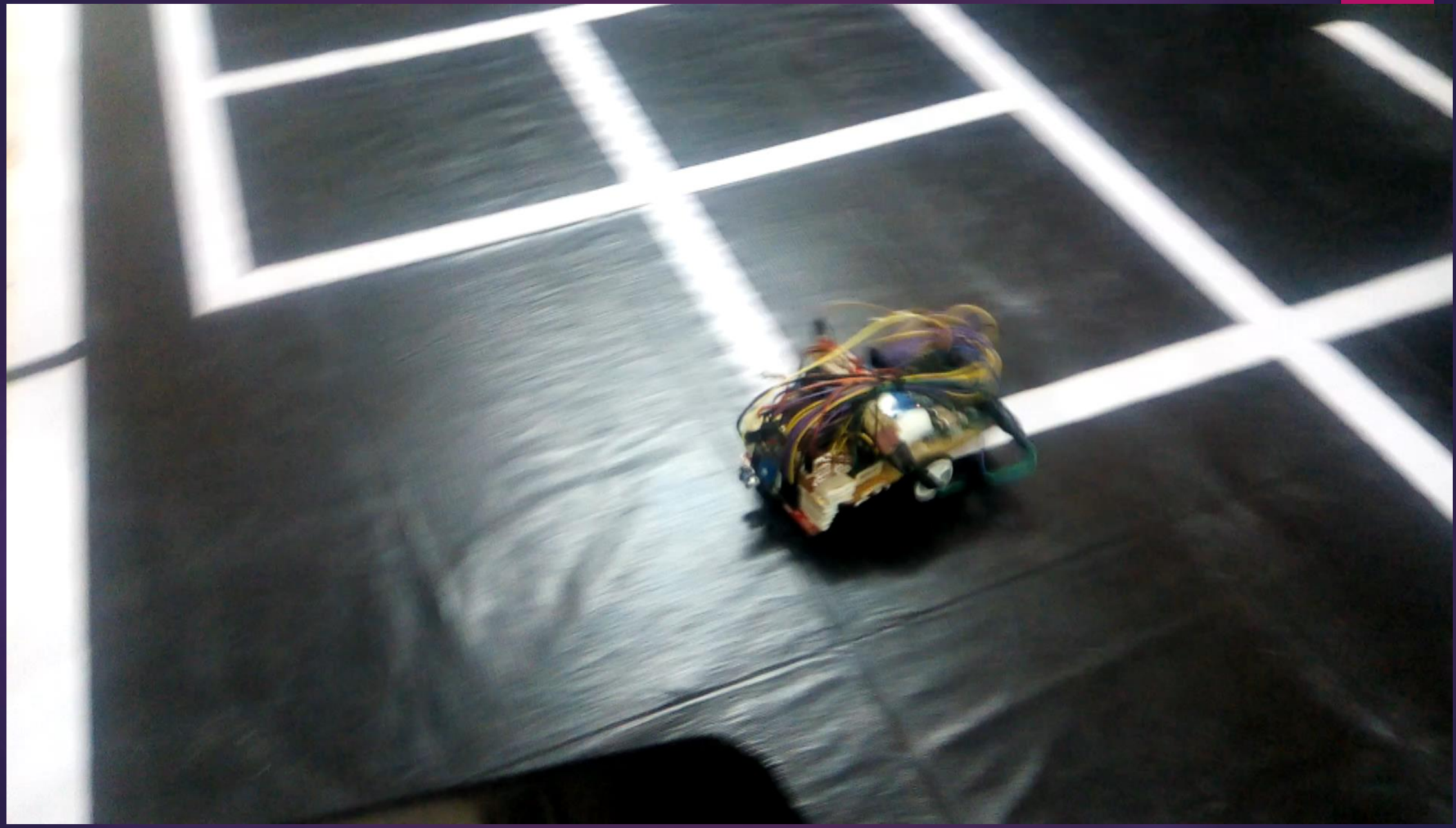
```
int s1=A0,s2=A1,s3=A2,s4=A3,s5=A4; //declaration and initialization of sensors
Int m1=6,m2=7,m3=8,m4=9;           //declaration and initialization of motors
void setup()
{
pinMode(6,OUTPUT);  //left motor
pinMode(7,OUTPUT);  //left motor
pinMode(8,OUTPUT);  //right motor
pinMode(9,OUTPUT);  //right motor
}

void loop()
{
s1v = analogRead(s1); //left most
s2v = analogRead(s2); //left
s3v = analogRead(s3); //middle
s4v = analogRead(s4); //right
s5v = analogRead(s5); //right most
```

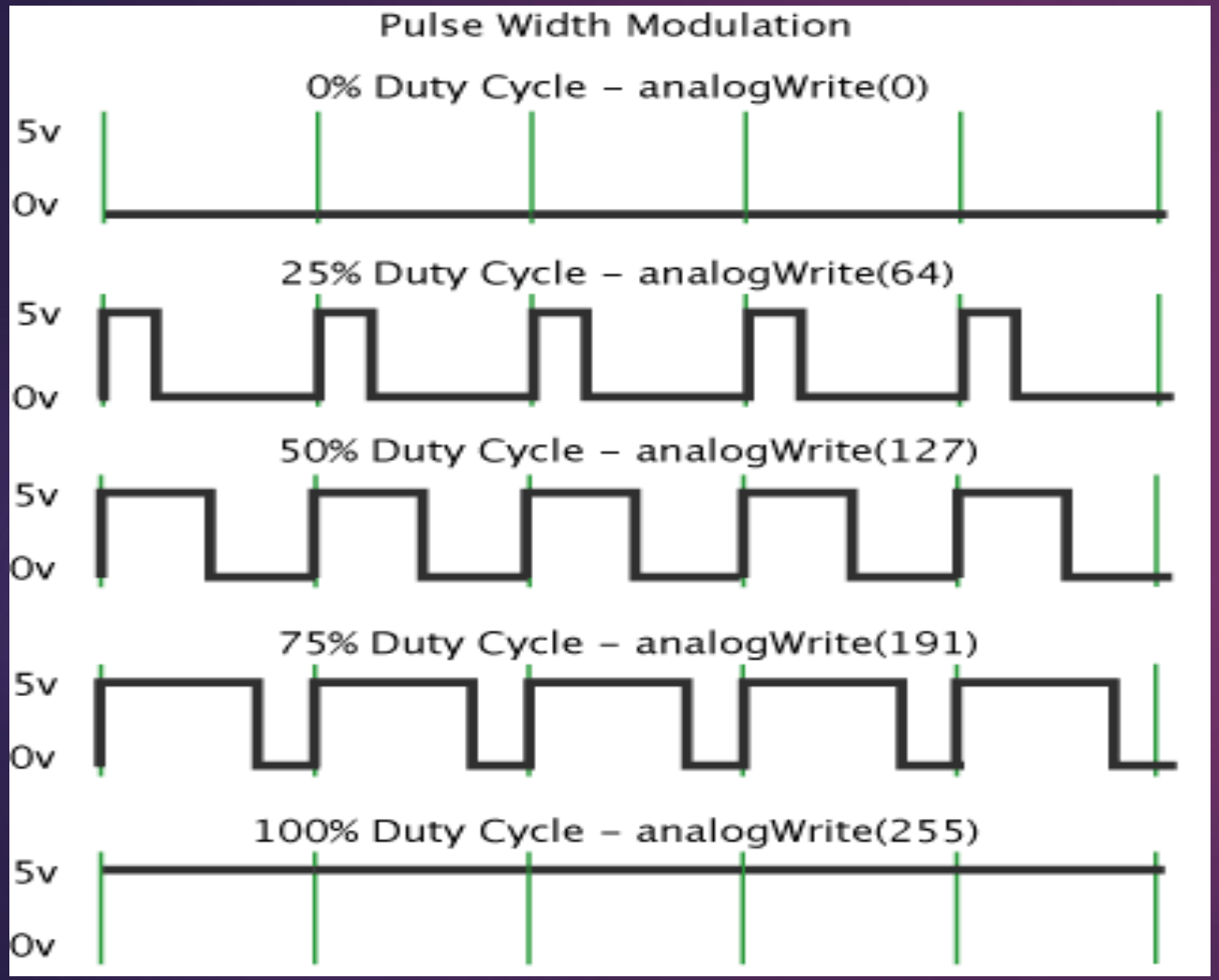
Line Following in Arduino

```
// ADC values vary b/w 700 - 800 when sensor is on black strip
// ADC values vary b/w 300 - 400  when sensor is on white strip

if ( s1v < 300 && s2v < 300 && s3v > 700 && s4v < 300 && s5v < 300)
{
move_forward();
}
else if ( s1v > 700 && s2v < 700 && s3v > 700 && s4v < 300 && s5v < 300)
{
move_left();
}
else if ( s1v < 300 && s2v < 300 && s3v > 700 && s4v > 700 && s5v > 700)
{
move_right();
}
```



PWM



- So a PWM signal is characterized by two parameters:
  1. Frequency: 1/T
  2. Duty Cycle: T_{on}/T
- Total Time Period (T): $T_{on} + T_{off}$
- Output (V_{out}): $(T_{on}/T) \times V_{in}$
- By varying T_{on} (keep T constant), we obtain different output voltages and speeds

WHY PWM?

- The speed control of DC motor is significant in applications where precision and protection are of importance.
- *Pulse width modulation* is a great method of controlling the amount of power delivered to a load without dissipating any wasted power.
- We can definitely control Speed of a motor with a resistor, but this wastes power and energy in the form of heat across the resistor.
- If we use PWM, we you do not have a resistor in series, meaning no power is dissipated in the form of heat. We just shuttle the Motor between ON & OFF, and the average gives us the voltage. So , no power is wasted.

Code without PWM

```
void move_left()
{
 digitalWrite(m1,LOW);
 digitalWrite(m2,HIGH);
 digitalWrite(m3,HIGH);
 digitalWrite(m4,LOW);
}
```

```
void move_right()
{
 digitalWrite(m1,HIGH);
 digitalWrite(m2,LOW);
 digitalWrite(m3,LOW);
 digitalWrite(m4,HIGH);
}
```

Code with PWM

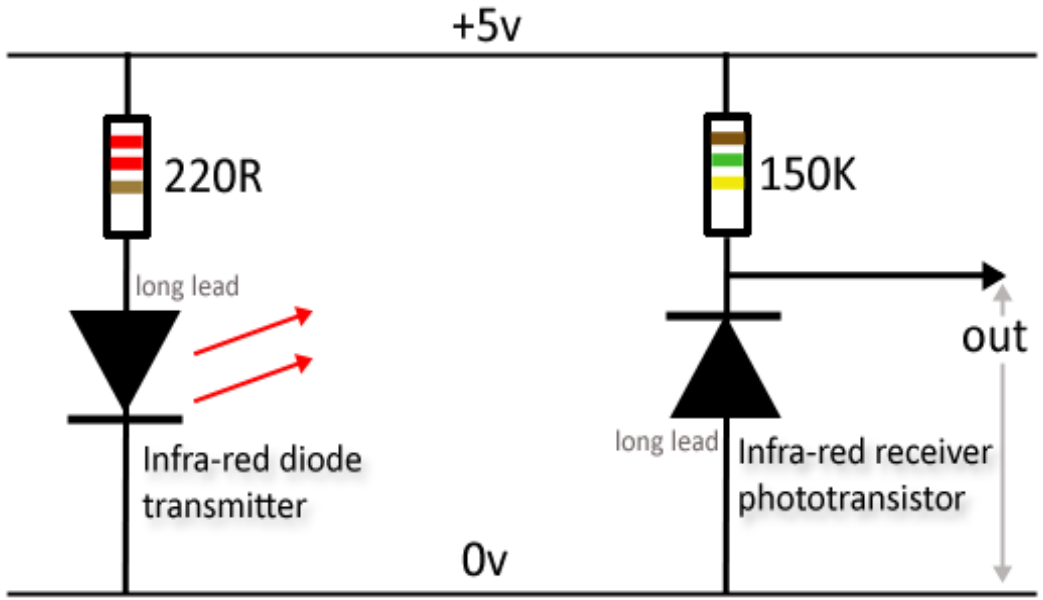
```
void move_left()
{
analogWrite(aoutleft,127);
analogWrite(aoutright,127);
digitalWrite(m1,LOW);
digitalWrite(m2,HIGH);
digitalWrite(m3,HIGH);
digitalWrite(m4,LOW);
}
```

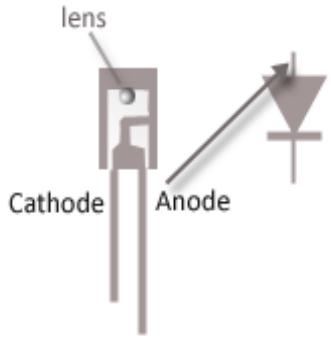
```
void move_right()
{
analogWrite(aoutleft,127);
analogWrite(aoutright,127);
digitalWrite(m1,HIGH);
digitalWrite(m2,LOW);
digitalWrite(m3,LOW);
digitalWrite(m4,HIGH);
}
```

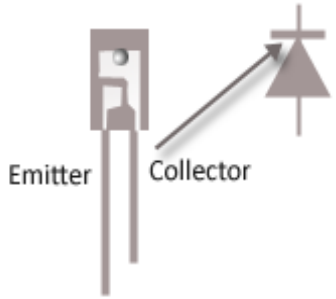
What's next?

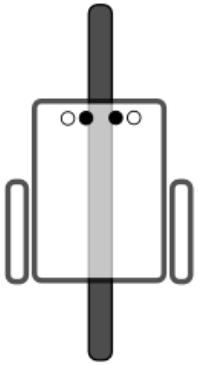
- Review of sensor mechanism
- Traversing a Grid
- Maze Solving Basics
- Curved Line Following (PID)

Review

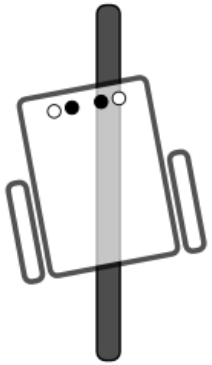








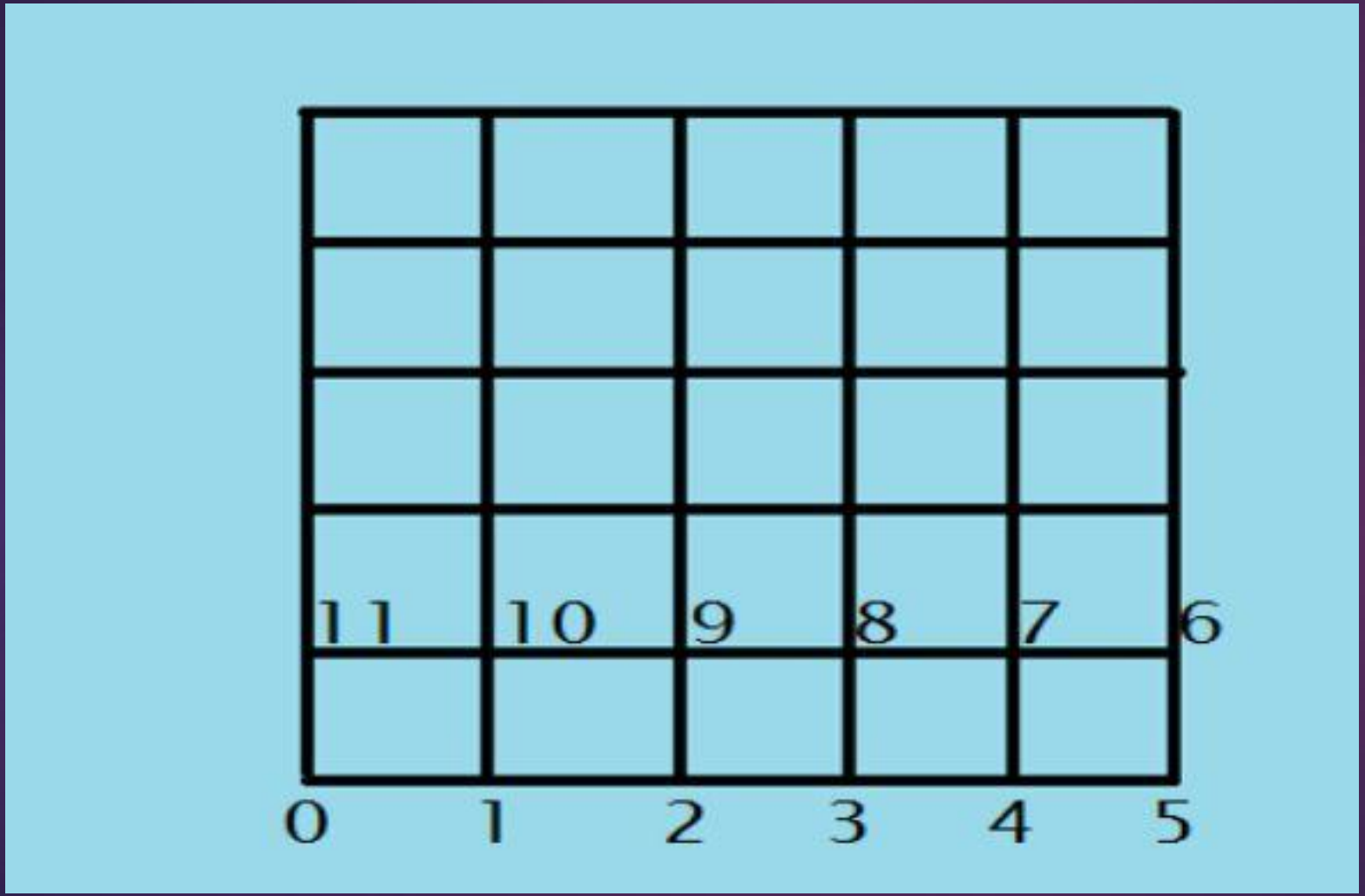
Both the sensors detect the line.
Hence the robot moves forward



The right sensor detects the line.
Hence the robot moves to the right.

The left sensor detects the line.
Hence the robot moves to the left.

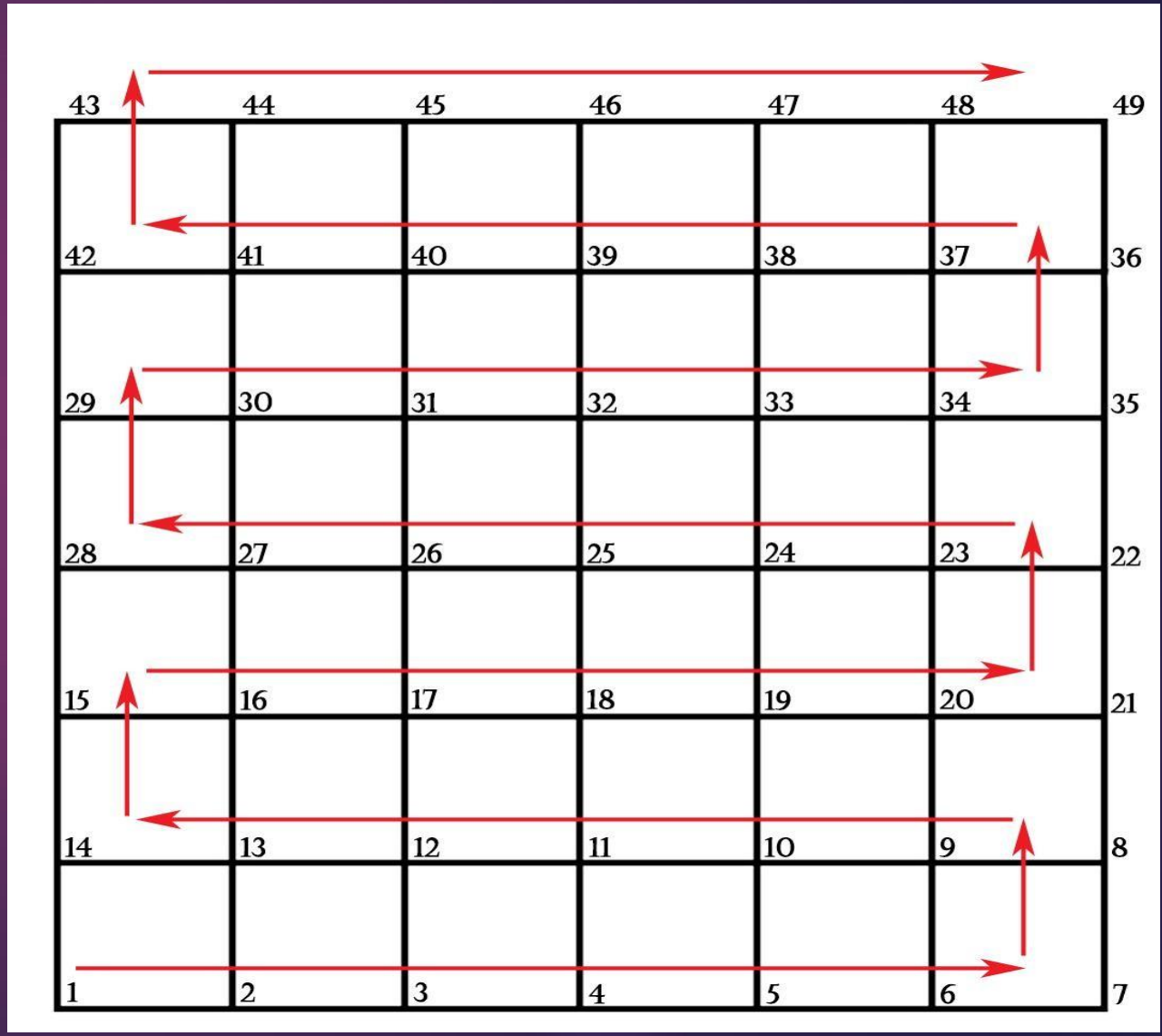
Numbering Convention in Grid

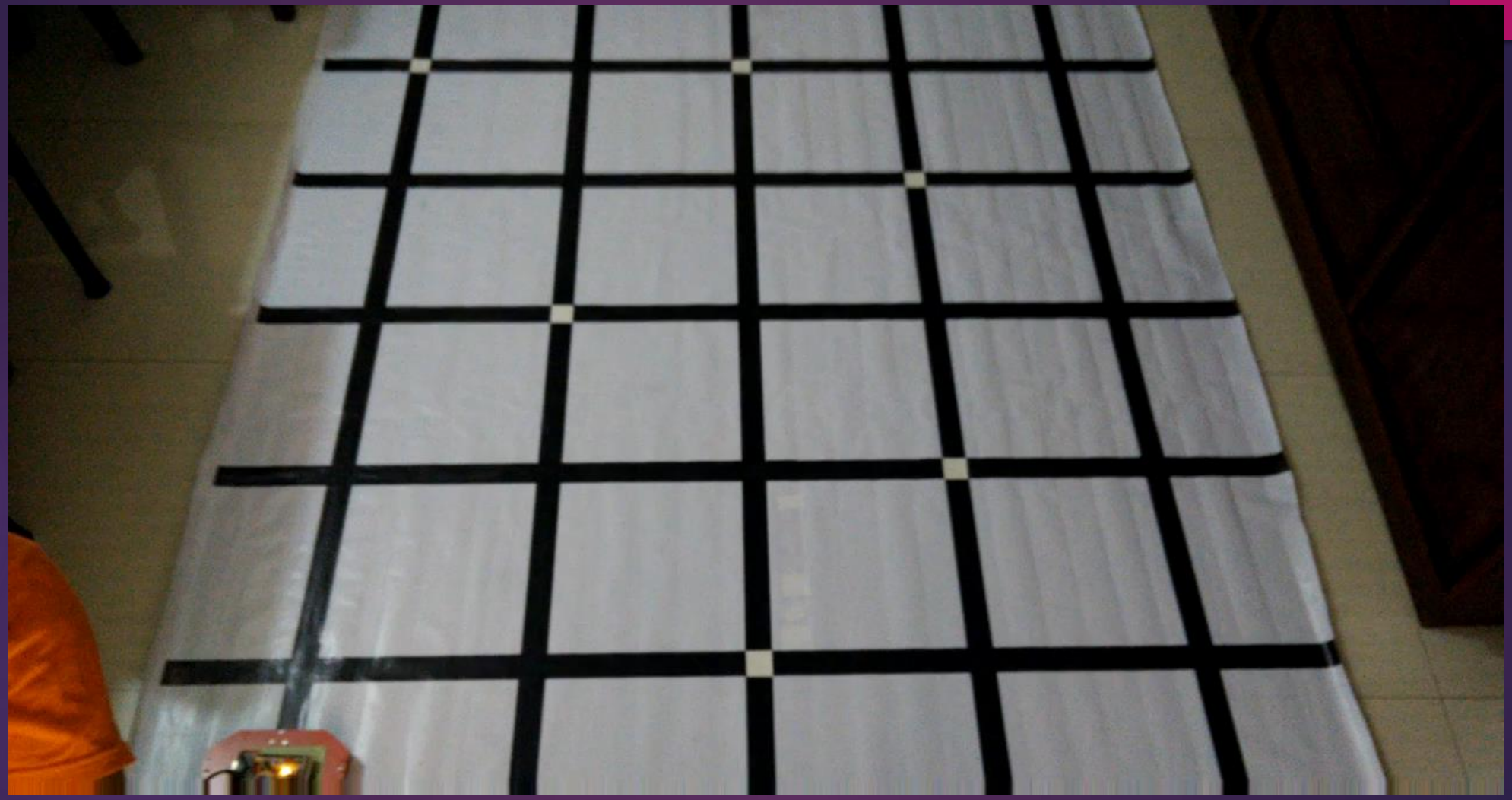


Grid Following

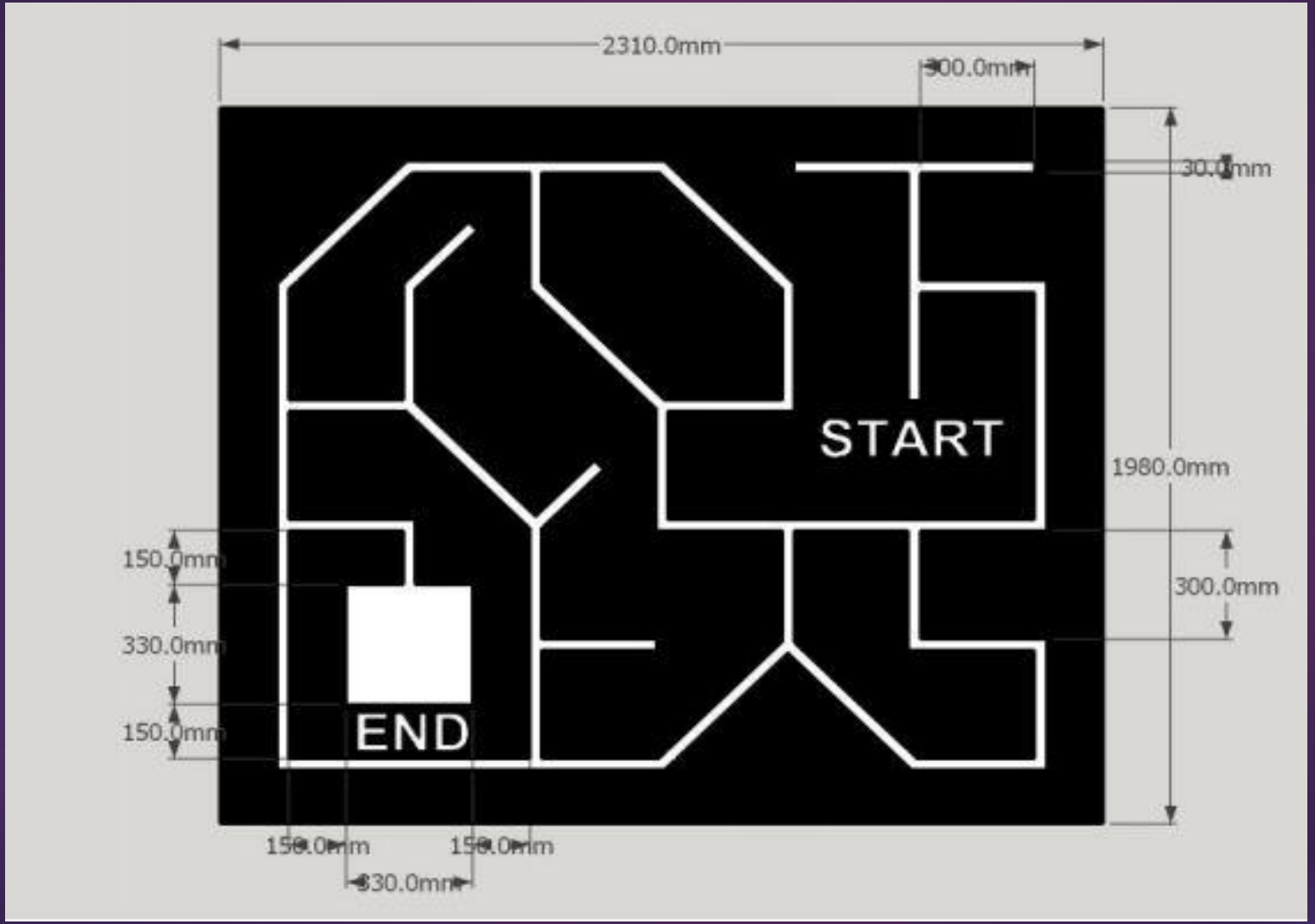
- Declare count variable
- Initialize to 1
- Increment as a new crossing is reached
- if (count%7 == 0 || count%7 == 1)

```
{
  temp = int (count/7)
  {
    if(temp%2 = 0)
    {
      Right_turn();
    }
    else
   {
      Left_turn();
   }
   }
   }
```





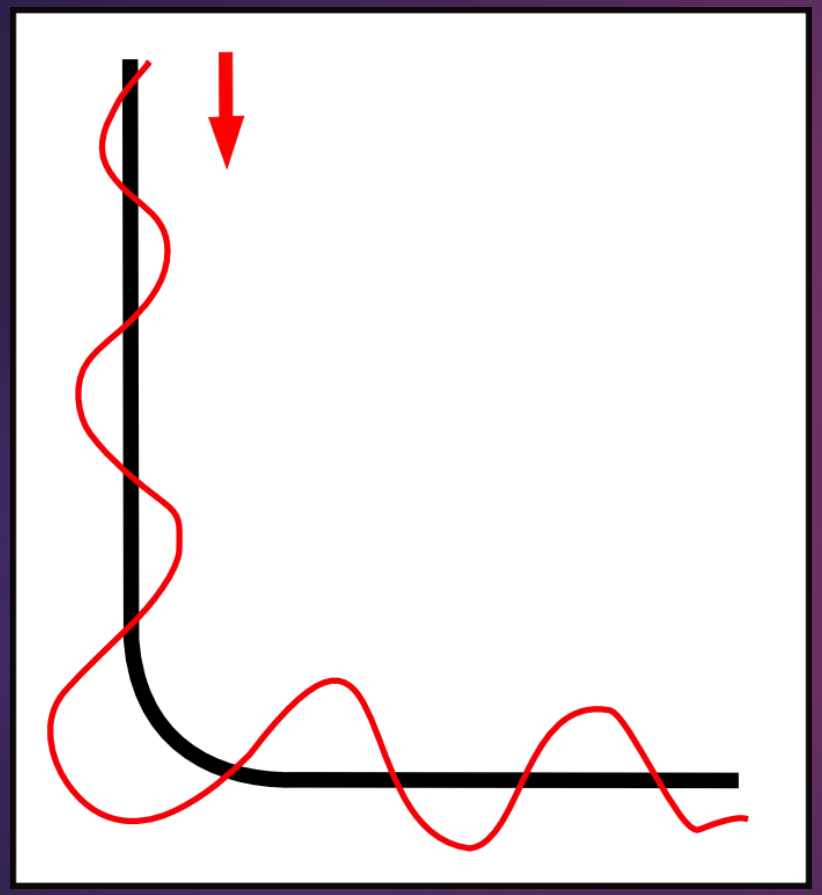
Maze Solving Basics



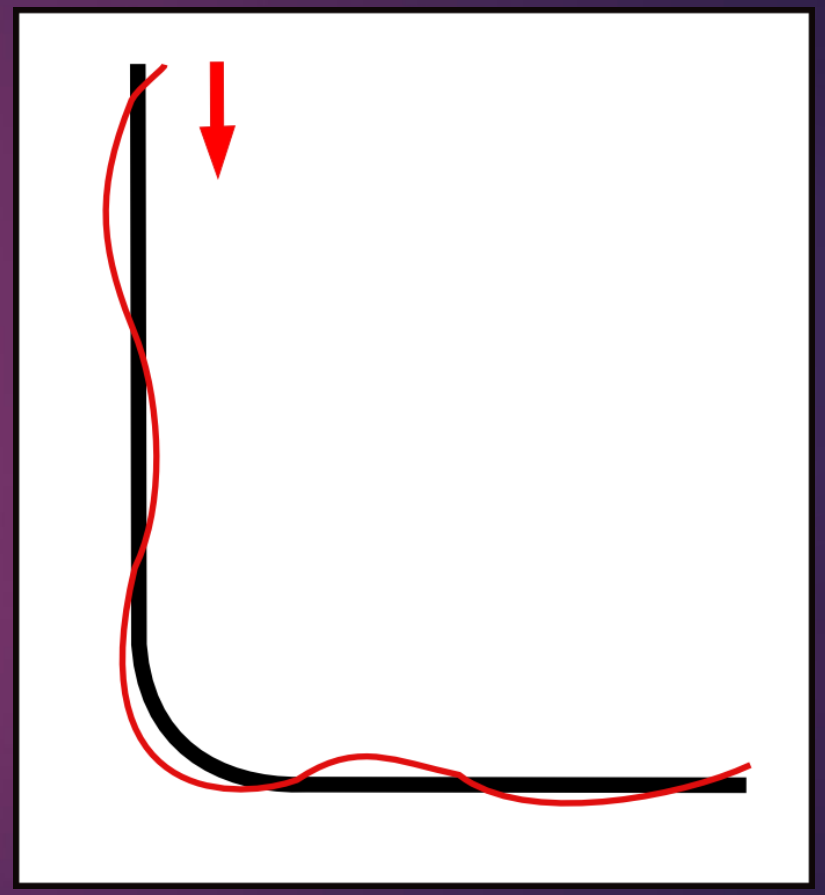
PID Control

- “PID” is an acronym for Proportional Integral Derivative.
- Main task of the PID controller is to minimize the error of whatever we are controlling.
- It takes in input, calculates the deviation from the intended behaviour and accordingly adjusts the output so that deviation from the intended behaviour is minimized and greater accuracy obtained.

Why PID?



Without PID



With PID

General Terminology

- Error - The error is the amount at which a device isn't doing something right.
- Proportional (P) - The proportional term is directly proportional to the error at present.
- Integral (I) - The integral term depends on the cumulative error made over a period of time (t).
- Derivative (D) - The derivative term depends rate of change of error.
- Constant (factor)- Each term (P, I, D) will need to be tweaked. Kp, Ki and Kd are the constants used to vary the effect of Proportional, Integral and Derivative terms respectively.

Error measurement

- In order to measure the error from the set position, i.e. the centre we can use the weighted values method. Suppose we are using a 5 sensor array to take the position input of the robot. The input obtained can be weighted depending on the possible combinations of input. The weight values assigned would be such that the error in position is defined both exactly and relatively.

00001	4
00011	3
00010	2
00110	1
00100	0
01100	-1
01000	-2
11000	-3
10000	-4

Line Following

- The **proportional** value is approximately proportional to your robot's position with respect to the line. We calculate the error based on the current position.
- The **integral** value records the history of your robot's motion: it is a sum of all of the values of the proportional term that were recorded since the robot started running.
- The **derivative** is the rate of change of the proportional value. To reduce the oscillating effect.
- The **set point value** is the reading that corresponds to the "perfect" placement of sensors on top of the lines.
- **The first task is to calculate the error.**

  Error = set point value – current value

PID formula

- Proportional = Kp * (Error)
- Derivative = Kd * (Error - LastError)
- Integral = Ki * (Integral + Error) //generally ignored
- Control value used to adjust the robot's motion= (Proportional) + (Integral) + (Derivative)
- RightMotorSpeed = RightBaseSpeed + Control Value
- LeftMotorSpeed = LeftBaseSpeed – Control Value
- These speeds are given to the motors using **PWM**

Tweaking & Tuning

- How to find **Kp,Kd,Ki**?
- **Kp -** The goal is to get the robot to follow the line even if it is very wobbly. If the robot overshoots and loses the line, reduce the Kp value. If the robot cannot navigate a turn or seems sluggish, increase the Kp value.
- **Kd -** Start with a small value. Try increasing this value until you see lesser amount of wobbling.
- **Ki -** Start with small or zero value. If the Ki value is too high, the robot will jerk left and right quickly. If it is too low, you won't see any perceivable difference. Since Integral is cumulative, the Ki value has a significant impact. You may end up adjusting it by .01 increments.

The END??

- PID? Is that the name of a super hero or a mega villain? It's kinda both. If you manage to tune the parameters perfectly, you'll be fascinated by the results that follow, however, at the same time, you're gonna have some sleepless nights if you get it completely wrong.