

Precision Farming Using Autonomous Vehicle

Group A1:

Jaouaher Belgacem

Jasmeet Singh Matta

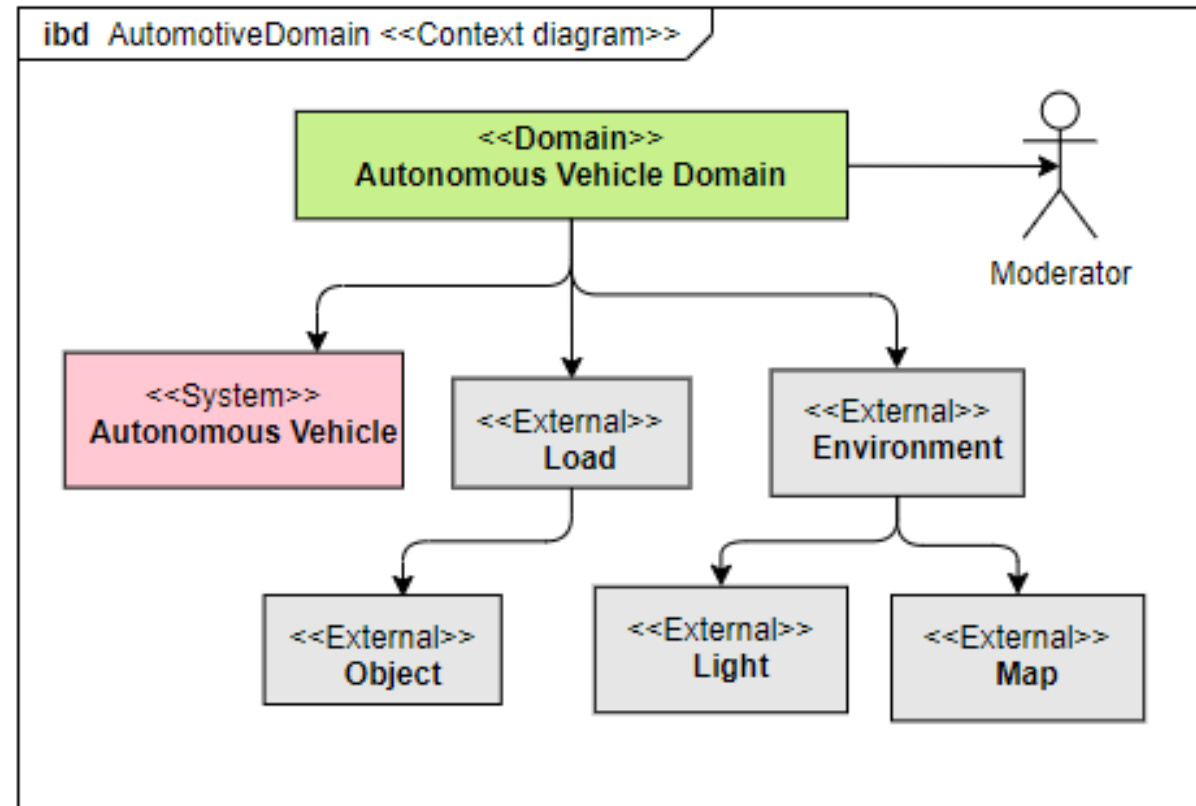
Arsany Girgis



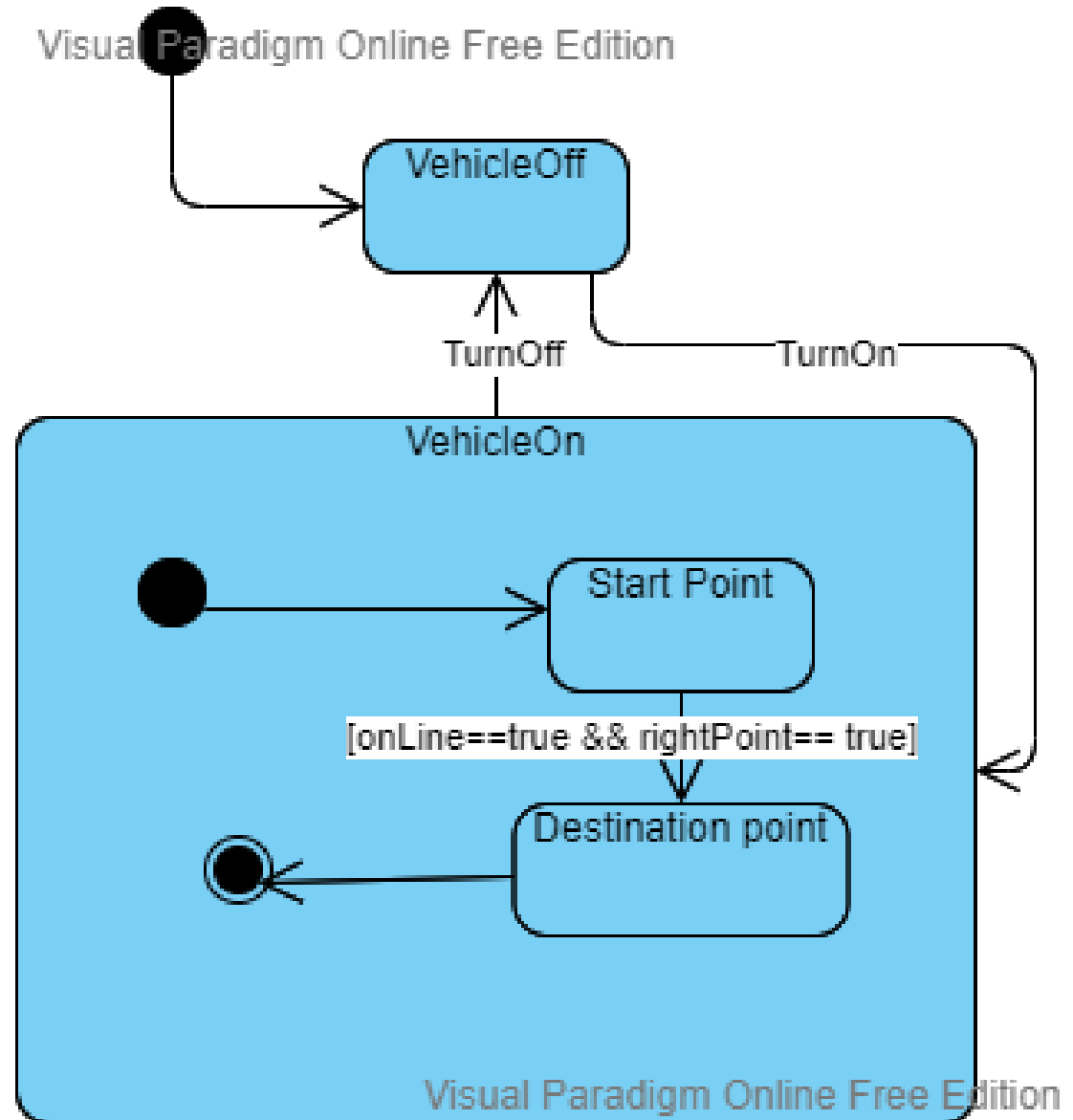
[1]

The Software Modelling

Context Diagram

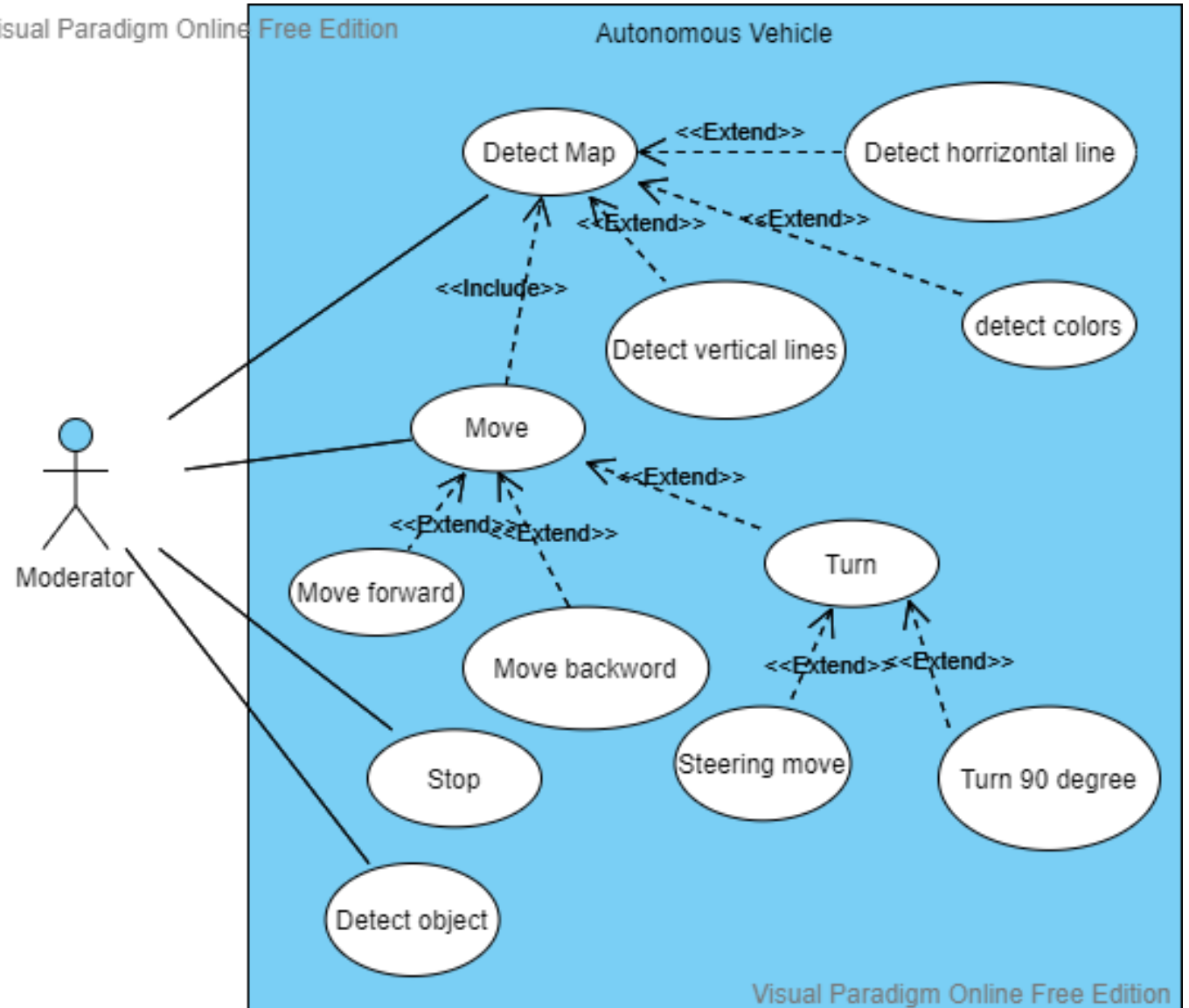


State Machine Diagram

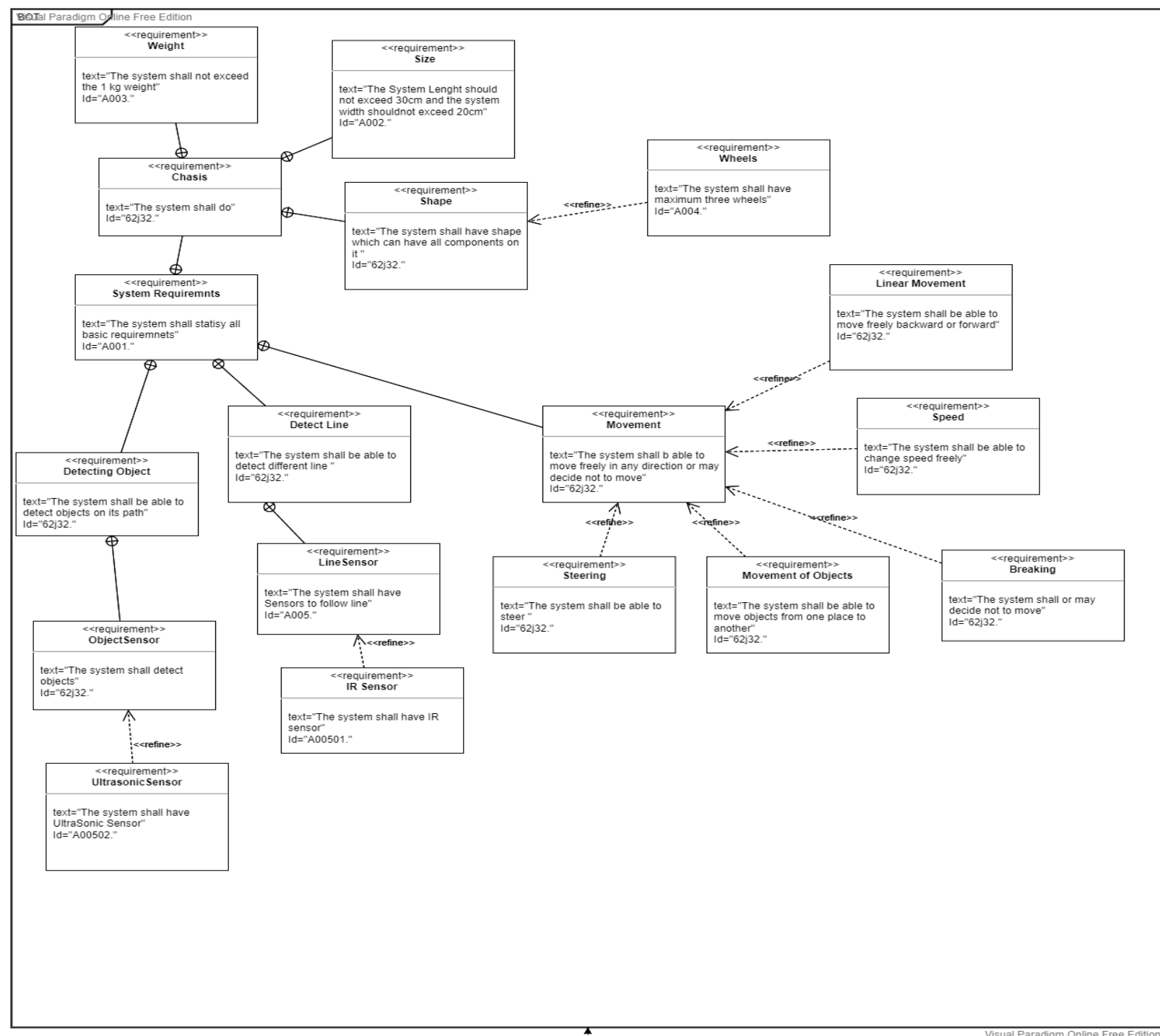


Use Case Diagram

Visual Paradigm Online Free Edition

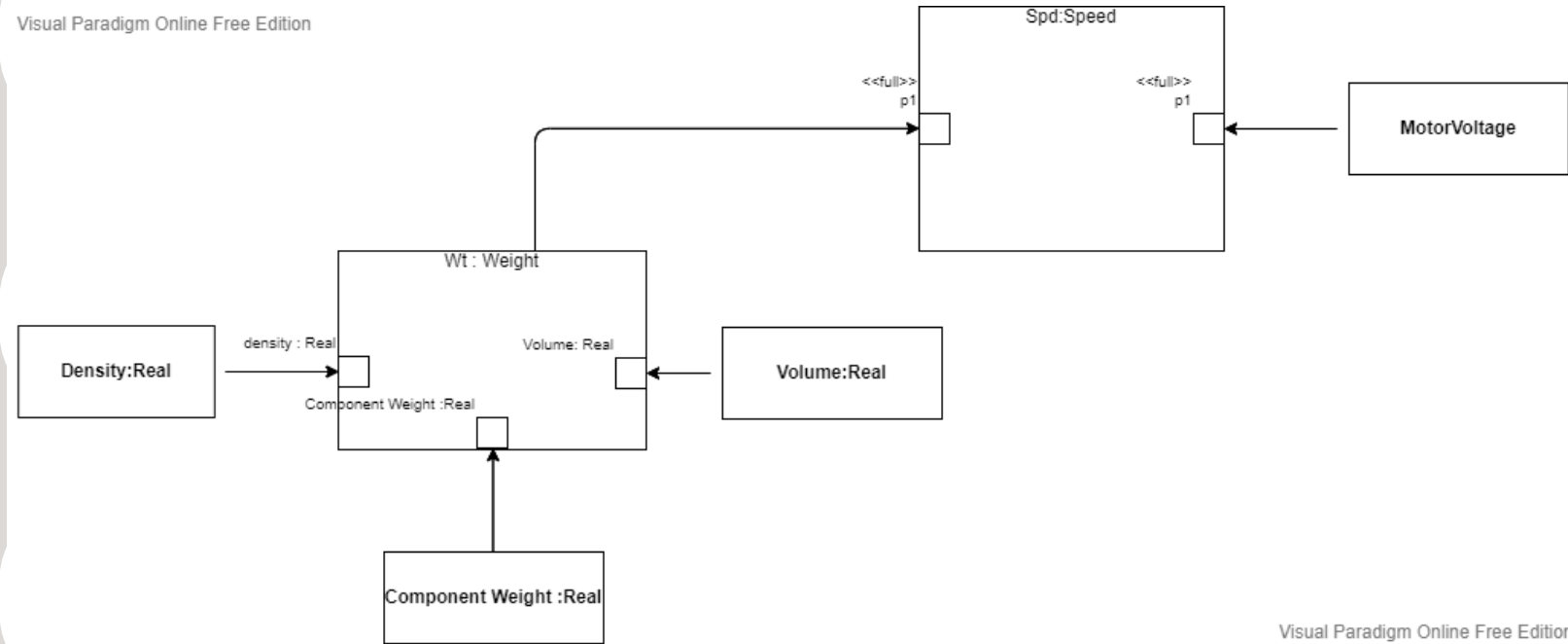


Requirement Diagram



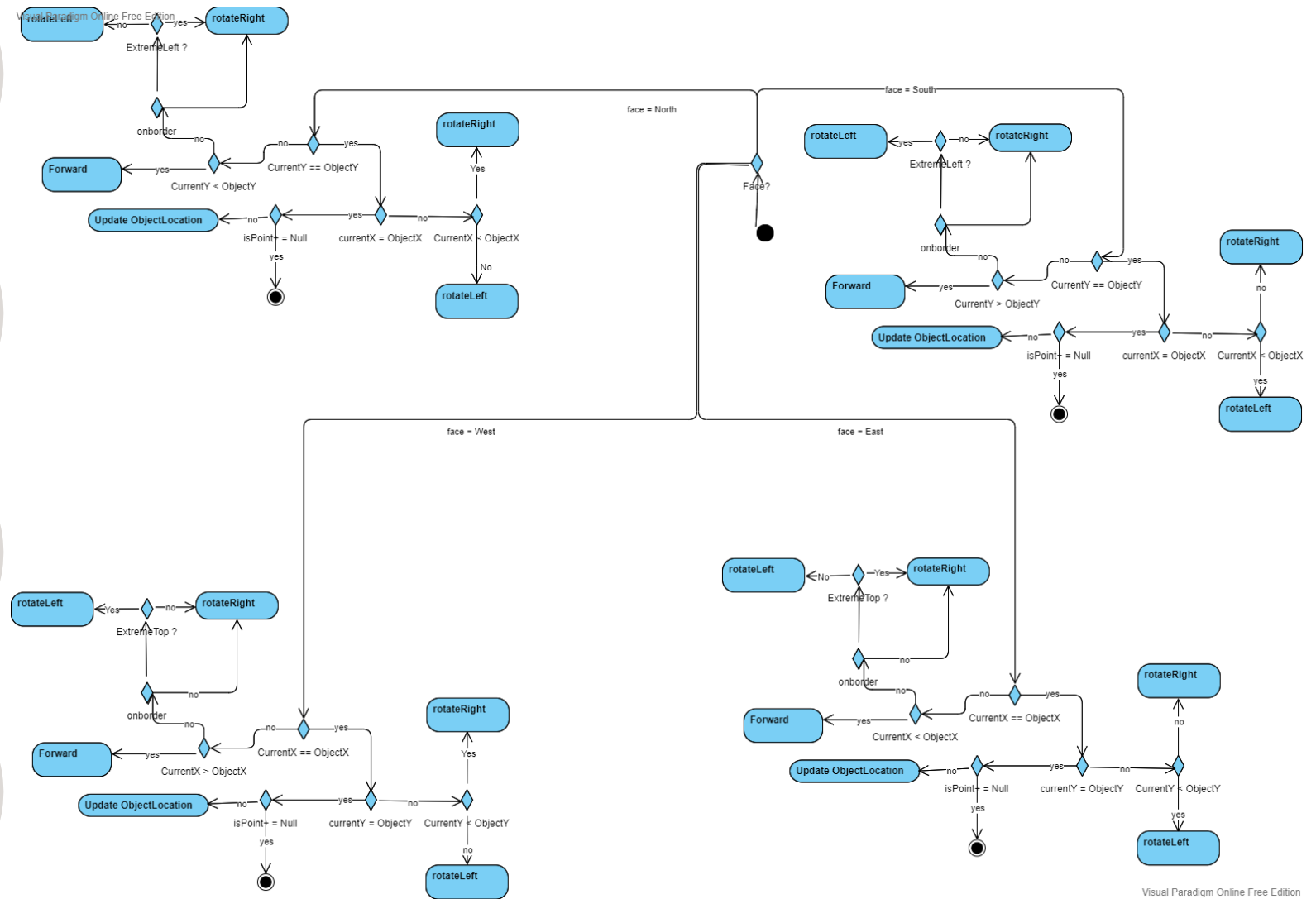
Parametric Diagram

Visual Paradigm Online Free Edition



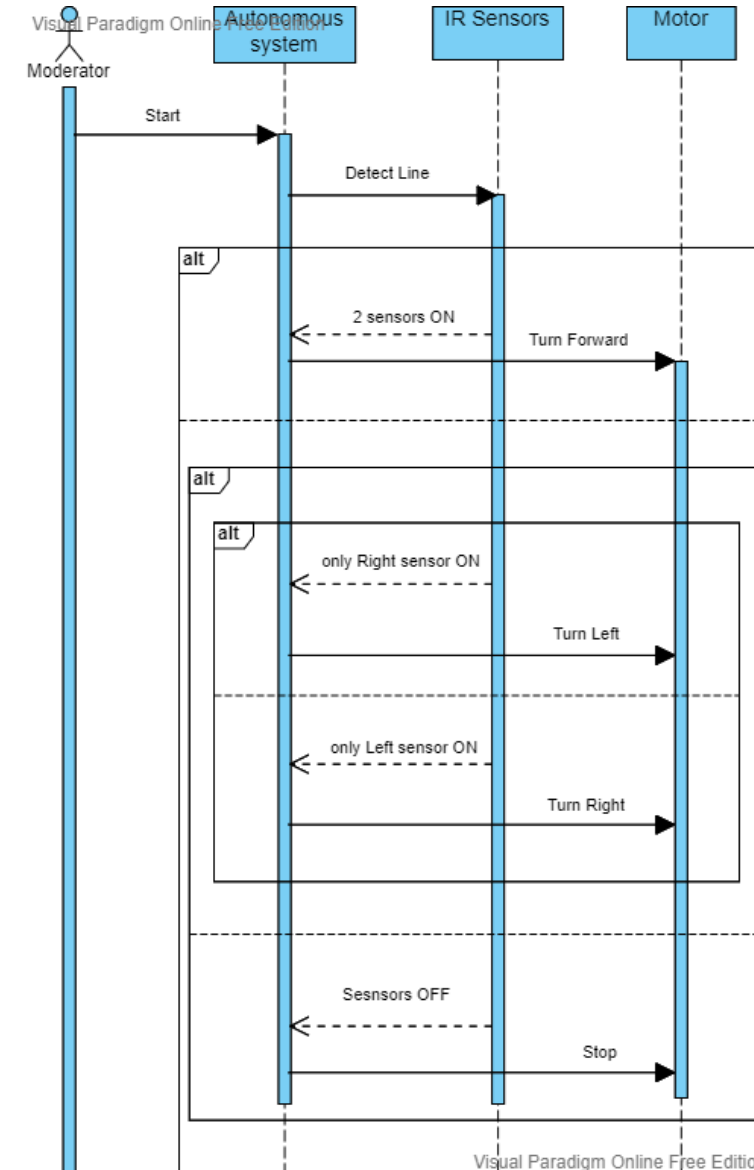
Visual Paradigm Online Free Edition

Activity Diagram



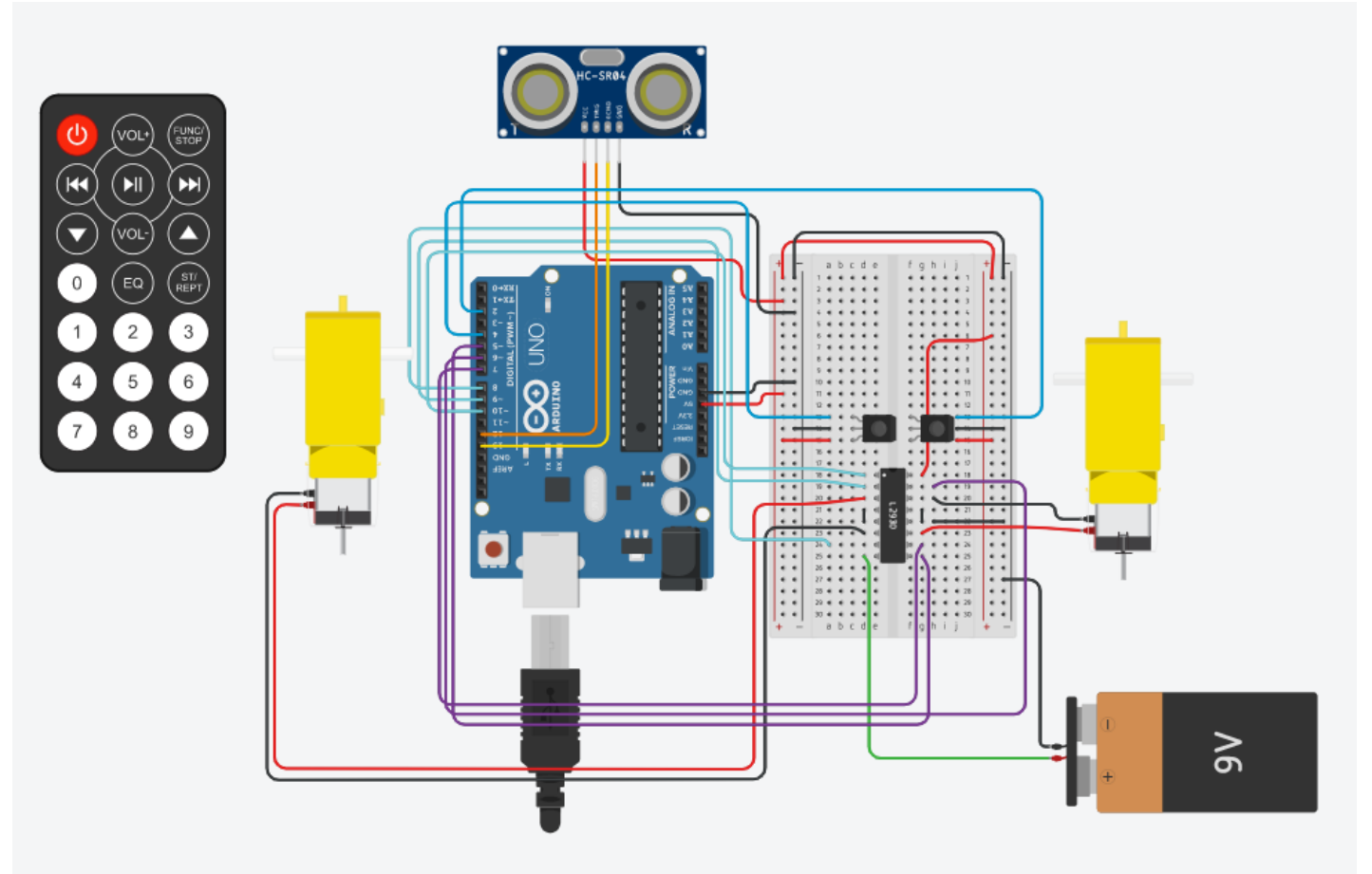
Visual Paradigm Online Free Edition

Sequence Diagram



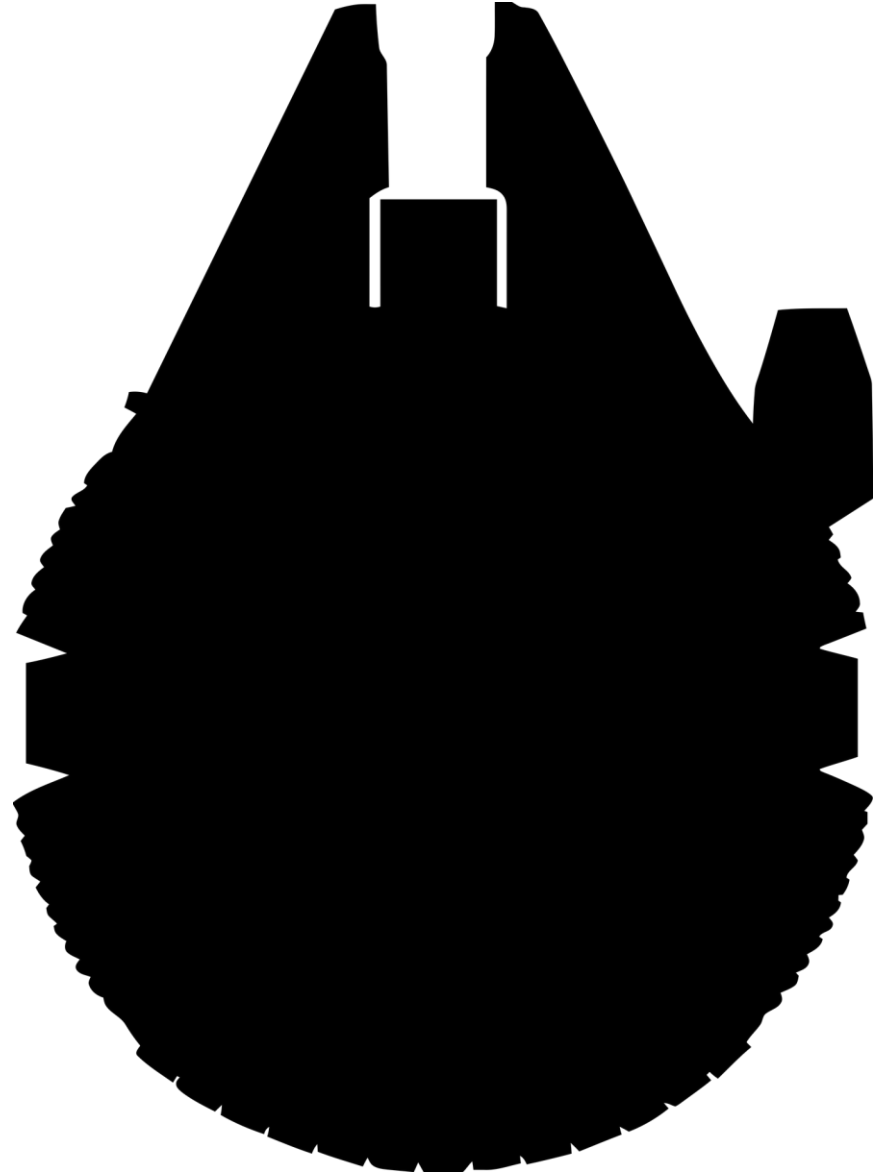
The Simulation

TinkerCad simulation

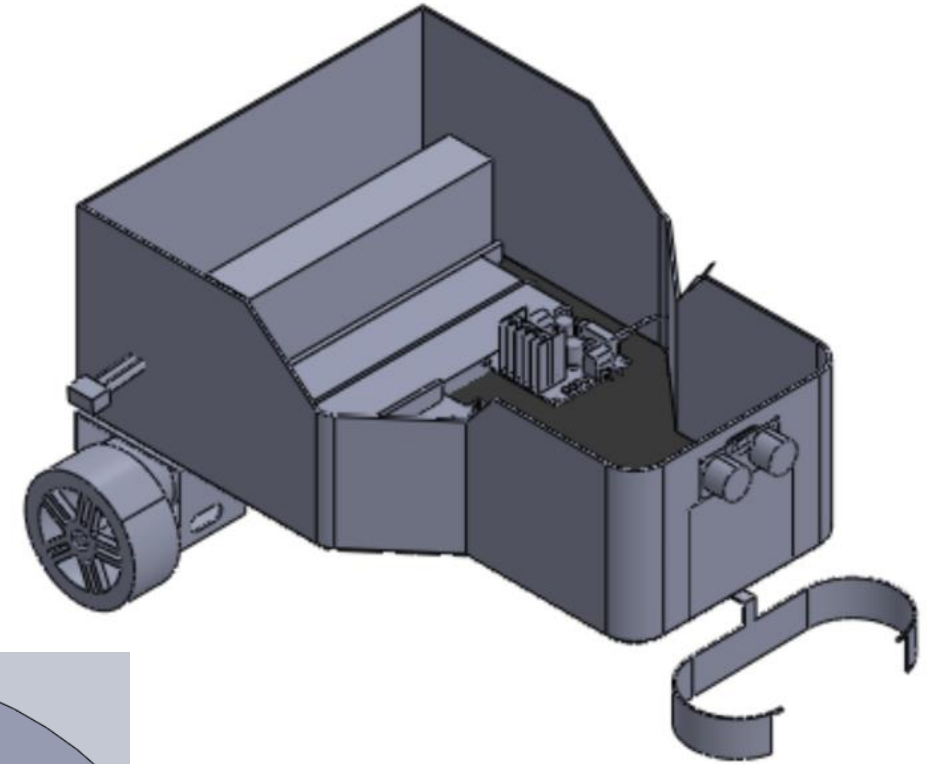
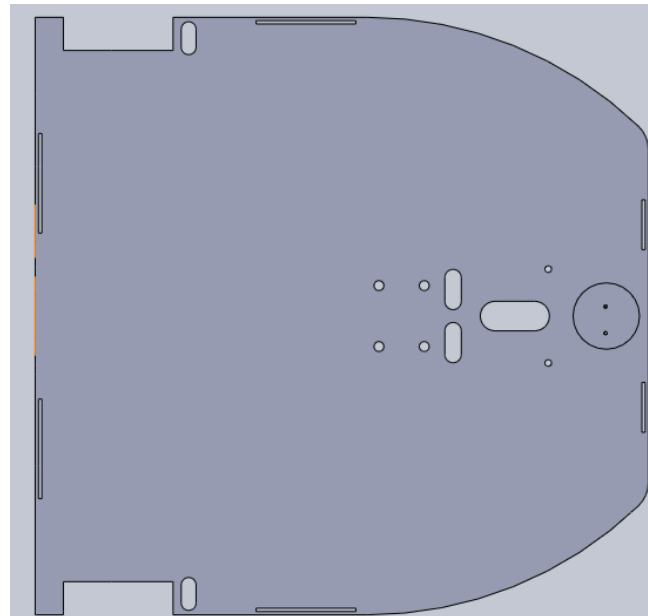


The Hardware Modelling

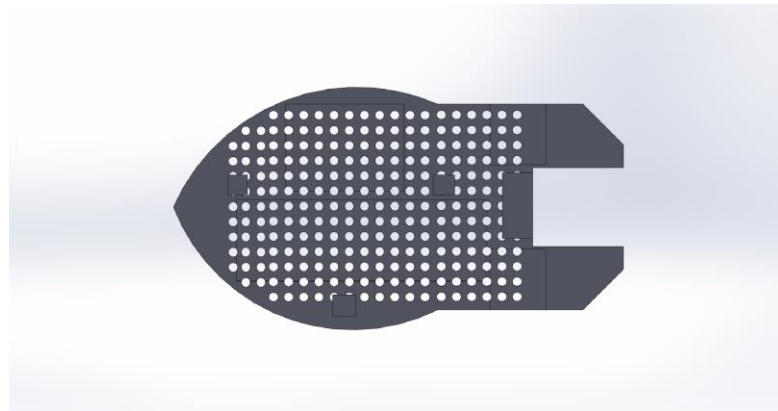
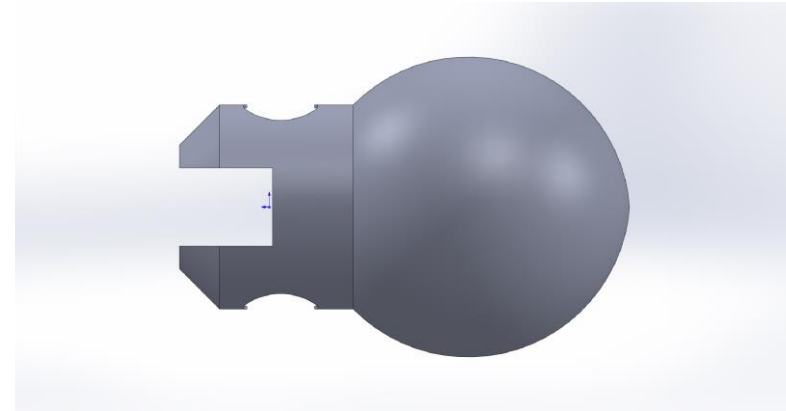
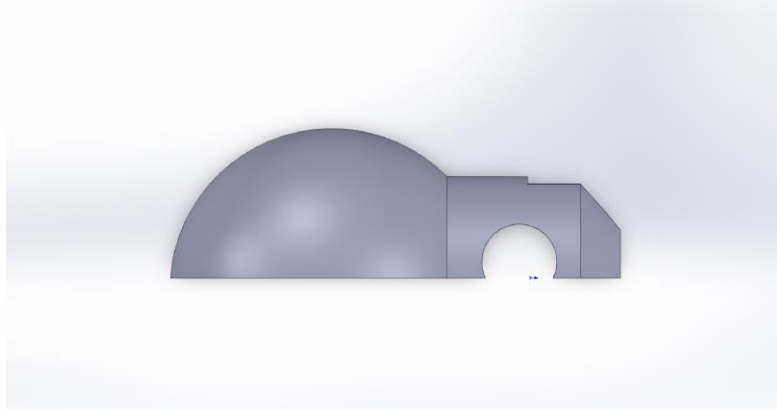
Design Inspiration



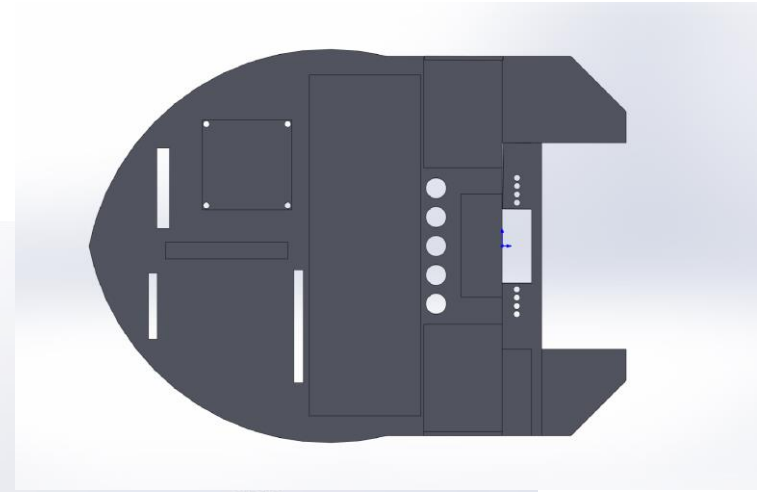
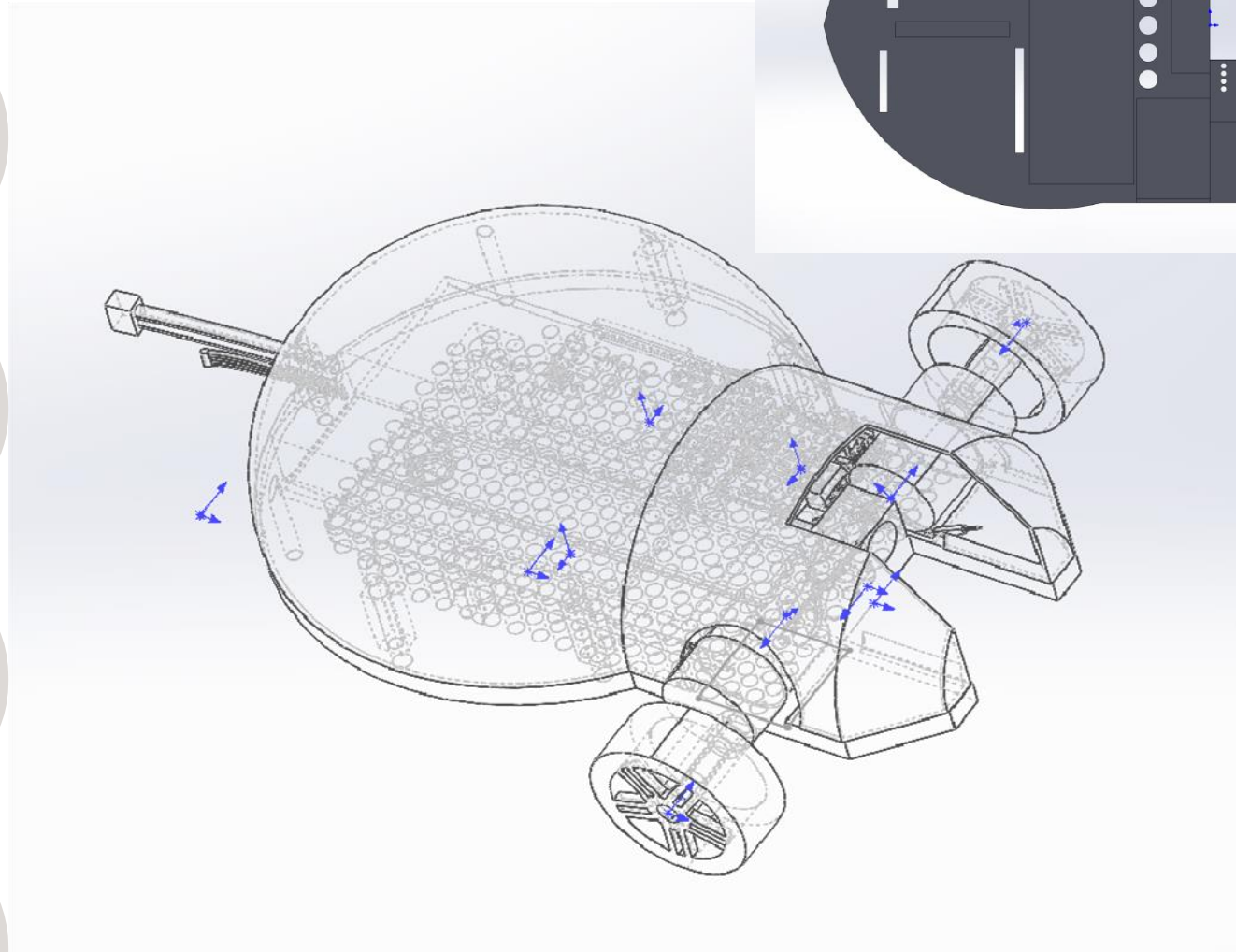
Design1



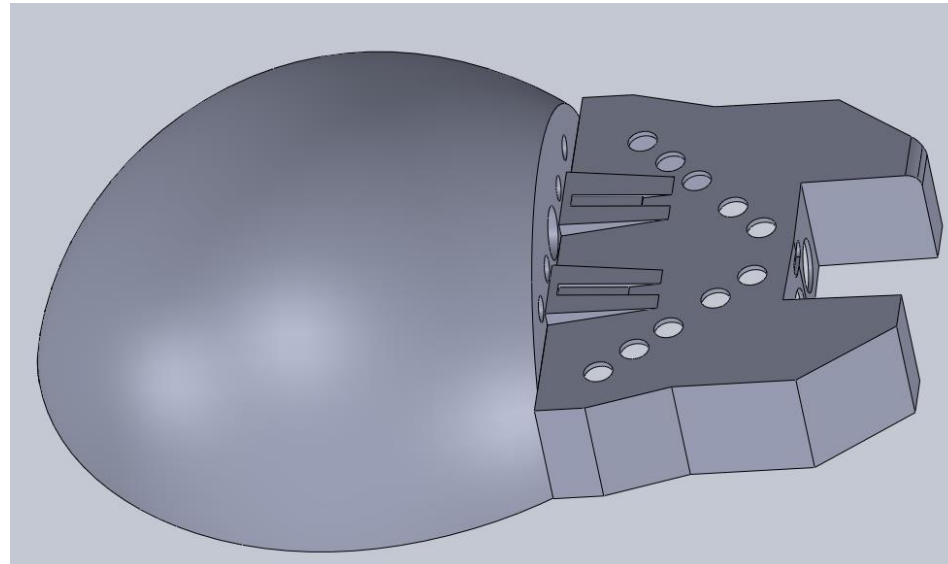
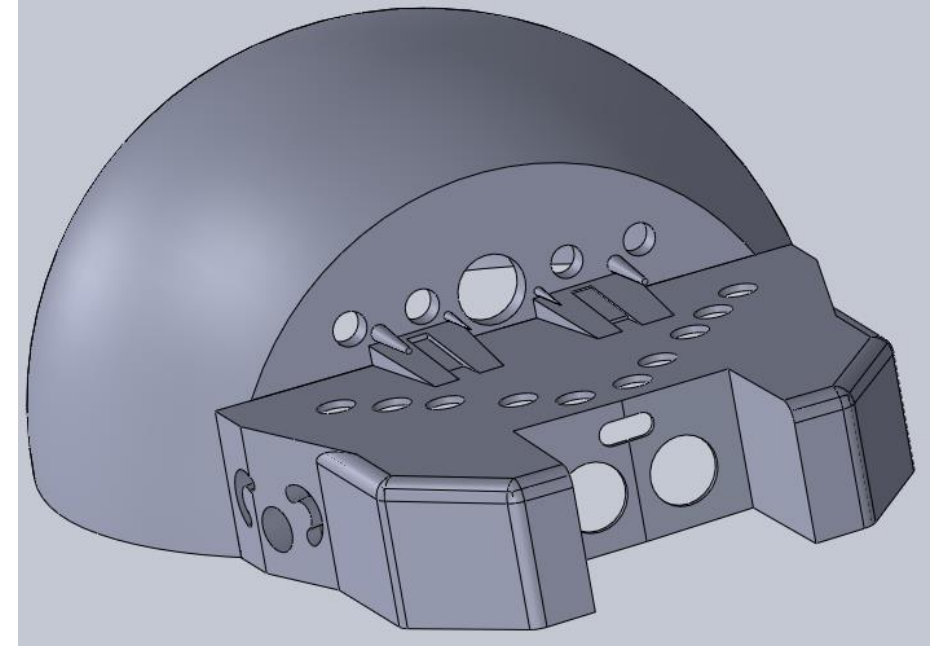
Design2



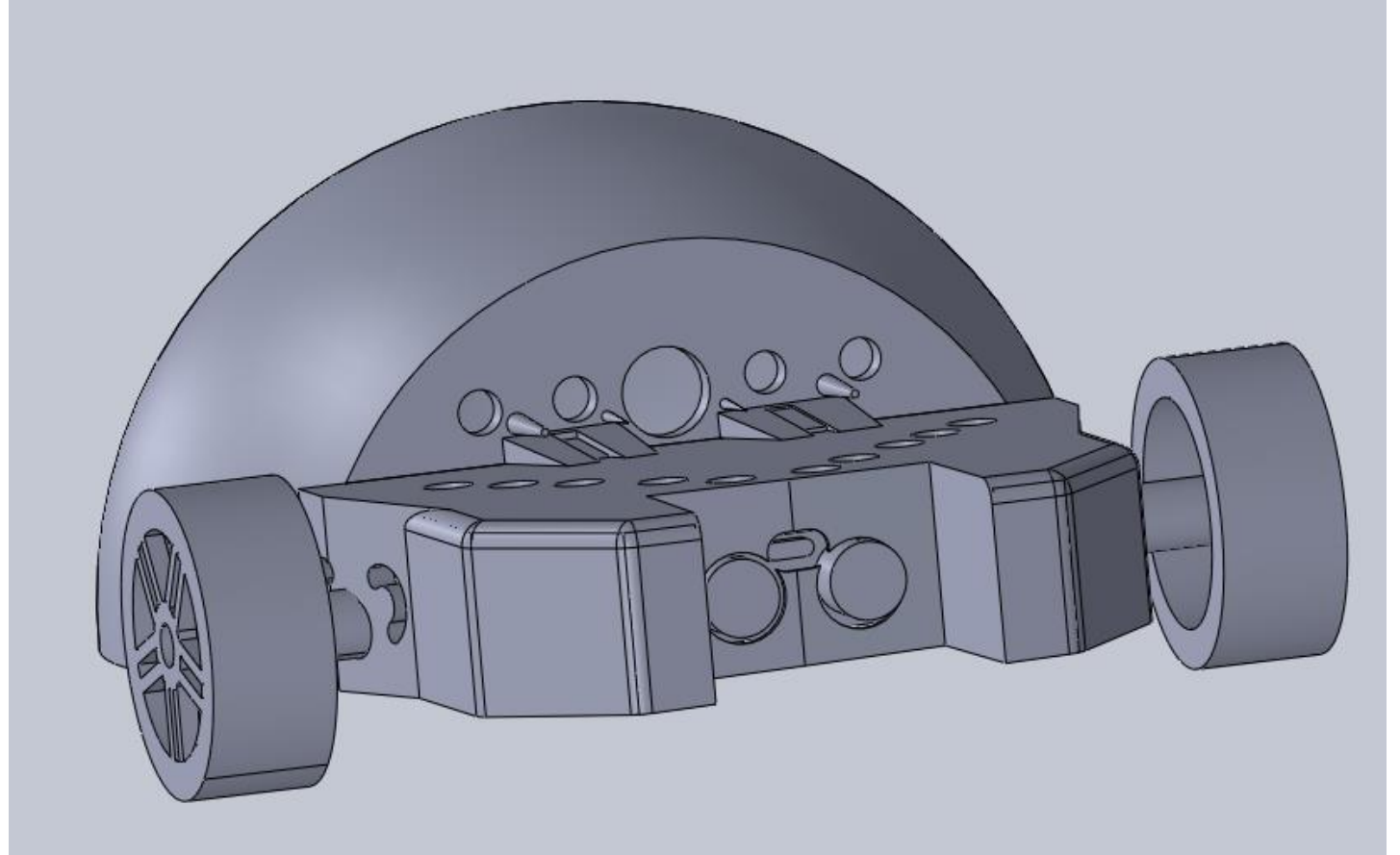
Design2



Final Design



Final Design



Code Implementation

Color Detection

```
//This is to get Green using filter  s2 s3
```

```
int getGreenPW() {  
    digitalWrite(S2,HIGH);  
    digitalWrite(S3,HIGH);  
    int PW;  
    PW = pulseIn(sensorOut, LOW);  
    return PW;  
}
```

```
//This is to get Blue using filter  s2 s3
```

```
int getBluePW(){  
    digitalWrite(S2, LOW);  
    digitalWrite(S3, HIGH);  
    int PW;  
    PW = pulseIn(sensorOut, LOW);  
    return PW;  
}
```

```
void loop(){  
    Serial.println(currentColor);  
}
```

```
// This is to get red using filter  s2 s3
```

```
int getRedPW()  
{  
    digitalWrite(S2, LOW);  
    digitalWrite(S3, LOW);  
    int PW;  
    PW = pulseIn(sensorOut, LOW);  
    return PW;  
}
```

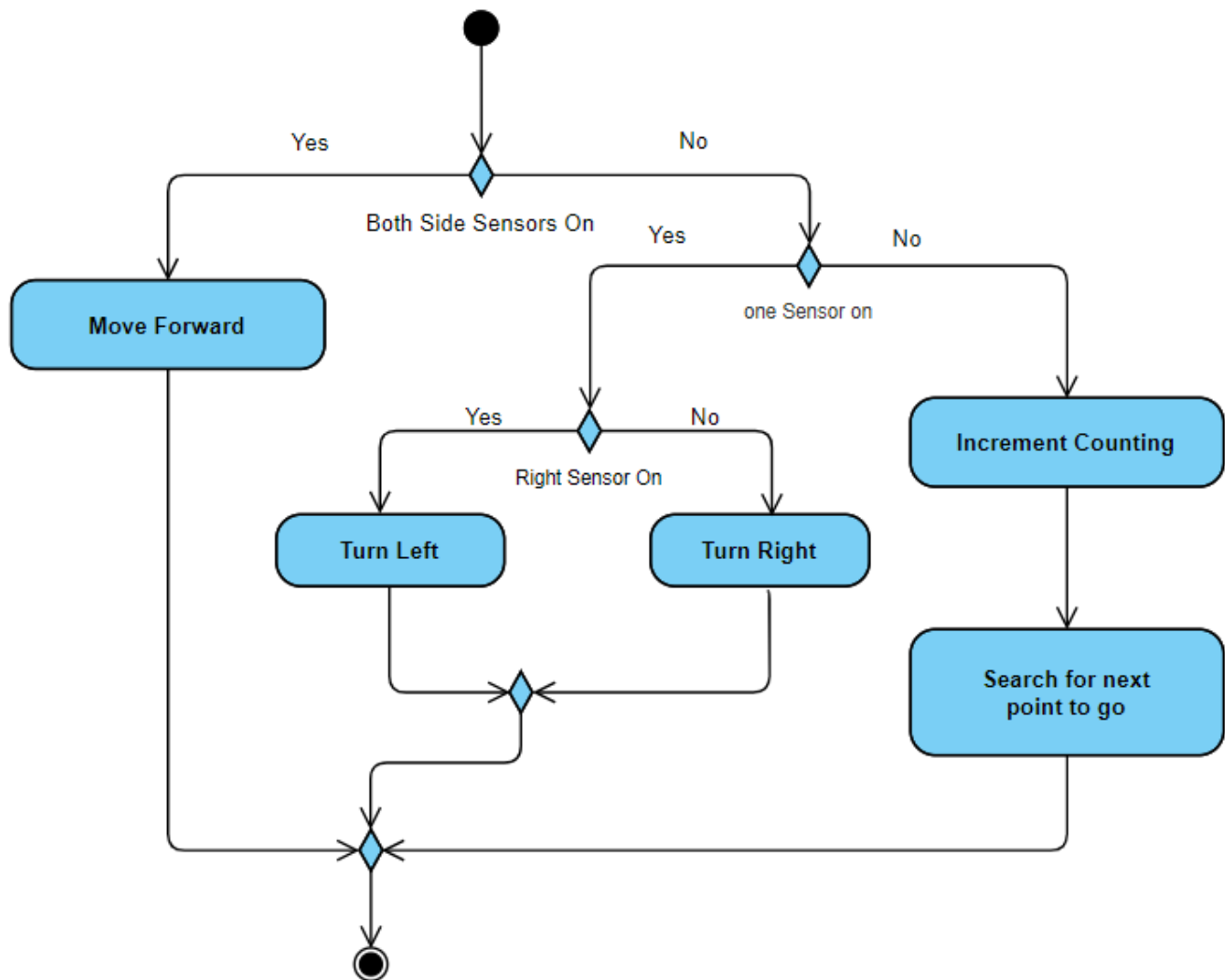
```
// This code is to get current color
```

```
void colorCall()  
{  
    redPW = getRedPW();  
    bluePW = getBluePW();  
    greenPW = getGreenPW();  
    getColor();  
}
```

Color Detection

```
//This is to update the color variable and also to calibrate the readings
void getColor(){
    if ((redPW >= 114 && redPW <= 150) && (greenPW >= 324 && greenPW <= 350) && (bluePW >= 364 && bluePW <= 395))
    {
        currentColor = 0;} // orange
    else if ((redPW >= 320 && redPW <= 340) && (greenPW >= 505 && greenPW <= 520) && (bluePW >= 235 && bluePW <= 355))
    {
        currentColor = 3;} // purple
    else if ((redPW >= 174 && redPW <= 230) && (greenPW >= 190 && greenPW <= 210) && (bluePW >= 320 && bluePW <= 335))
    {
        currentColor = 2;} // green
    else if ((redPW >= 232 && redPW <= 290) && (greenPW >= 170 && greenPW <= 190) && (bluePW >= 130 && bluePW <= 160))
    {
        currentColor = 1;} // cyan
    else if ((redPW >= 90 && redPW <= 108) && (greenPW >= 95 && greenPW <= 115) && (bluePW >= 80 && bluePW <= 95))
    {
        currentColor = 4;} // white
    else if (((redPW >= 900 && redPW <= 960) && (greenPW >= 1000 && greenPW <= 1065) && (bluePW >= 800 && bluePW <= 860))
    || ((redPW >= 622 && redPW <= 750) && (greenPW >= 710 && greenPW <= 810) && (bluePW >= 500 && bluePW <= 677)))
    {
        currentColor = 5;} // black
}
```

Line Follower Logic



Motor Pins Configuration

```
//This is to rotate motor
```

```
void switchState(){
```

```
switch(state){
```

```
case Stop:
```

```
digitalWrite(motorA1,LOW);  
digitalWrite(motorB1,LOW);  
digitalWrite(motorA2,LOW);  
digitalWrite(motorB2,LOW);  
break;
```

```
case Forward:
```

```
digitalWrite(motorA1,HIGH);  
digitalWrite(motorB1,HIGH);  
digitalWrite(motorA2,LOW);  
digitalWrite(motorB2,LOW);  
break;
```

```
case Right:
```

```
digitalWrite(motorA1,HIGH);  
digitalWrite(motorB1,LOW);  
digitalWrite(motorA2,LOW);  
digitalWrite(motorB2,HIGH);  
break;
```

```
case Left:
```

```
digitalWrite(motorA1,LOW);  
digitalWrite(motorB1,HIGH);  
digitalWrite(motorA2,HIGH);  
digitalWrite(motorB2,LOW);  
break;
```

```
case Backward:
```

```
digitalWrite(motorA1,LOW);  
digitalWrite(motorB1,LOW);  
digitalWrite(motorA2,HIGH);  
digitalWrite(motorB2,HIGH);  
break;
```

```
}
```

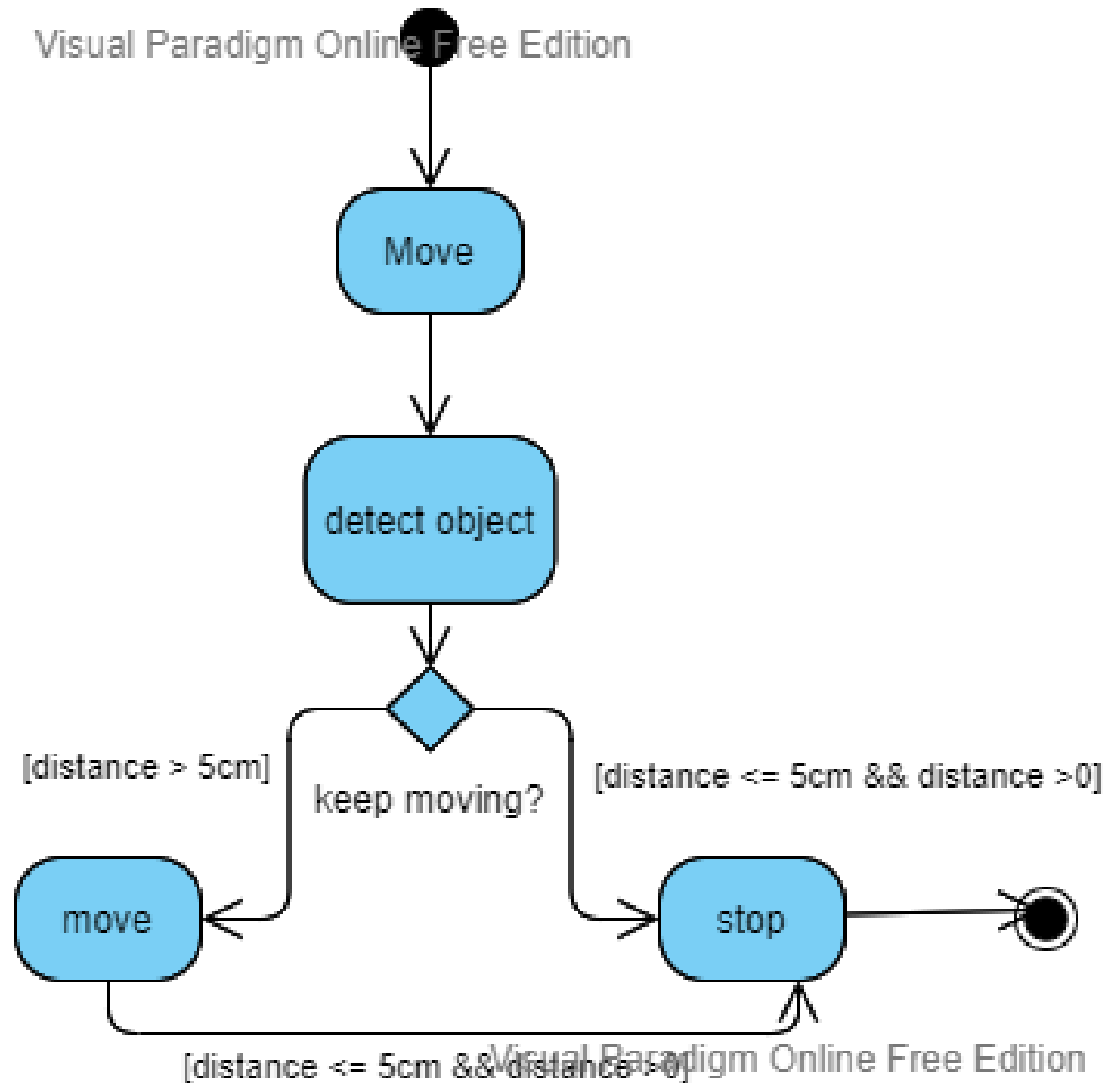
Line Follower Logic

```
// This is to keep bot allign  
void onLine()  
{  
    if(digitalRead(ln1) == HIGH && digitalRead (ln2) == HIGH)  
    {  
        change = true;  
        state = Forward;  
    }  
    if(digitalRead(ln1)==HIGH && digitalRead(ln2)==LOW)  
    {  
        state = Left;  
    }  
    if(digitalRead(ln1)==LOW && digitalRead(ln2)==HIGH)  
    {  
        state = Right;}  
}
```


Line Follower Logic

```
if (digitalRead (ln1)== LOW && digitalRead(ln2)== LOW)
{
    if (change == true){
        change = false;
        mapp(); // Here we are updating the position on map
    }
    state = Stop;
    switchState();
    Movement();
}
switchState();
}
```

Object Detection

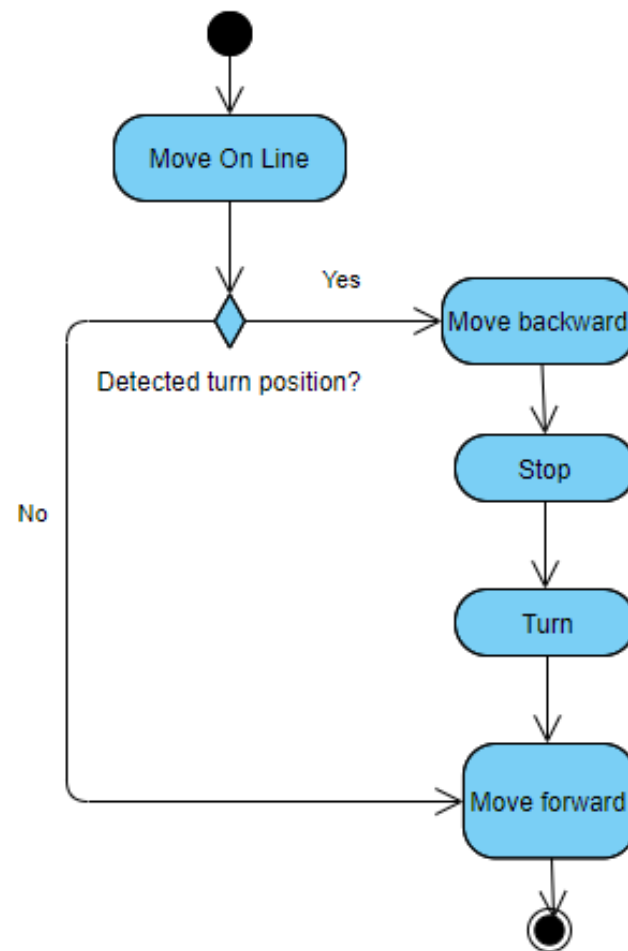


Object Detection

```
#define sonicTrig 12
#define sonicEcho 11
volatile int duration, distance;
volatile bool objectDetected= false;
void setup() {
    pinMode(sonicTrig,OUTPUT);
    pinMode(sonicEcho,INPUT);
    Serial.begin(9600);
}

void loop() {
void ultraSonic()
{
    digitalWrite(sonicTrig,HIGH);
    digitalWrite(sonicTrig,LOW);
    duration = pulseIn(sonicEcho,HIGH);
    distance = duration*0.0343/2;
    Serial.print("The Distance: ");
    Serial.println(distance);
/* the distance should be positive because at some points
 * the ultrasonic detects with negative numbers*/
*/
    if (distance <=50 && distance >= 0)
    {
        Serial.println ("Object detected" );
        Serial.print ("Distance of the object from the car is " );
        Serial.print ( distance);
        Serial.println ( " cm");// print out the distance in cm.
        objectDetected = true;}
    else{ objectDetected = false;}
}
```

90° Turn



90° Turn

```
case Backward:  
  //analogWrite(motorPowerA,180);  
  //analogWrite(motorPowerB,100);
```

```
digitalWrite(motorA1,LOW);  
digitalWrite(motorB1,LOW);  
digitalWrite(motorA2,HIGH);  
digitalWrite(motorB2,HIGH);  
break;
```

```
void rotateLeft()  
{  
  
  state = Backward;  
  switchState();  
  state = Stop;  
  digitalWrite(motorA1,LOW);  
  digitalWrite(motorB1,HIGH);  
  digitalWrite(motorA2,HIGH);  
  digitalWrite(motorB2,LOW);  
  delay(500);  
}
```

90° Turn

```
void rotateRight()  
{
```

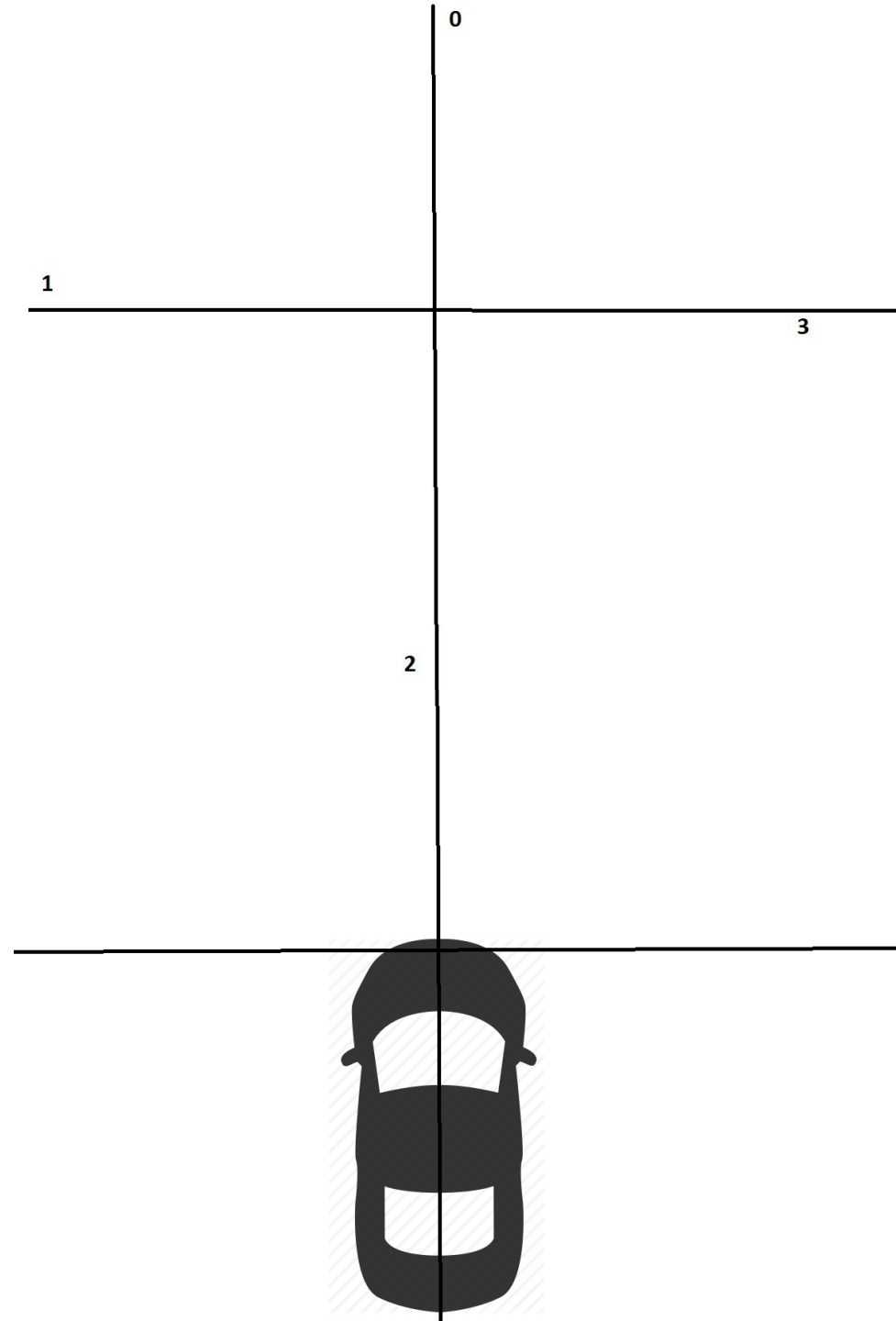
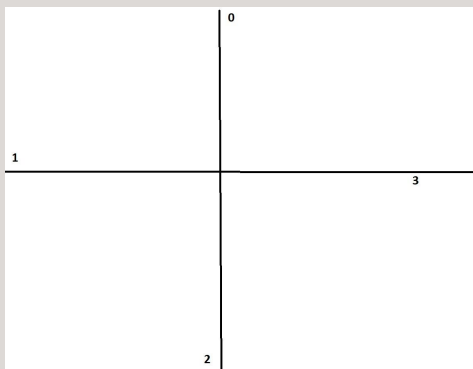
```
    state = Backward;  
    switchState();  
    state = Stop;  
    digitalWrite(motorA1,HIGH);  
    digitalWrite(motorB1,LOW);  
    digitalWrite(motorA2,LOW);  
    digitalWrite(motorB2,HIGH);  
    delay(500);
```

```
}
```

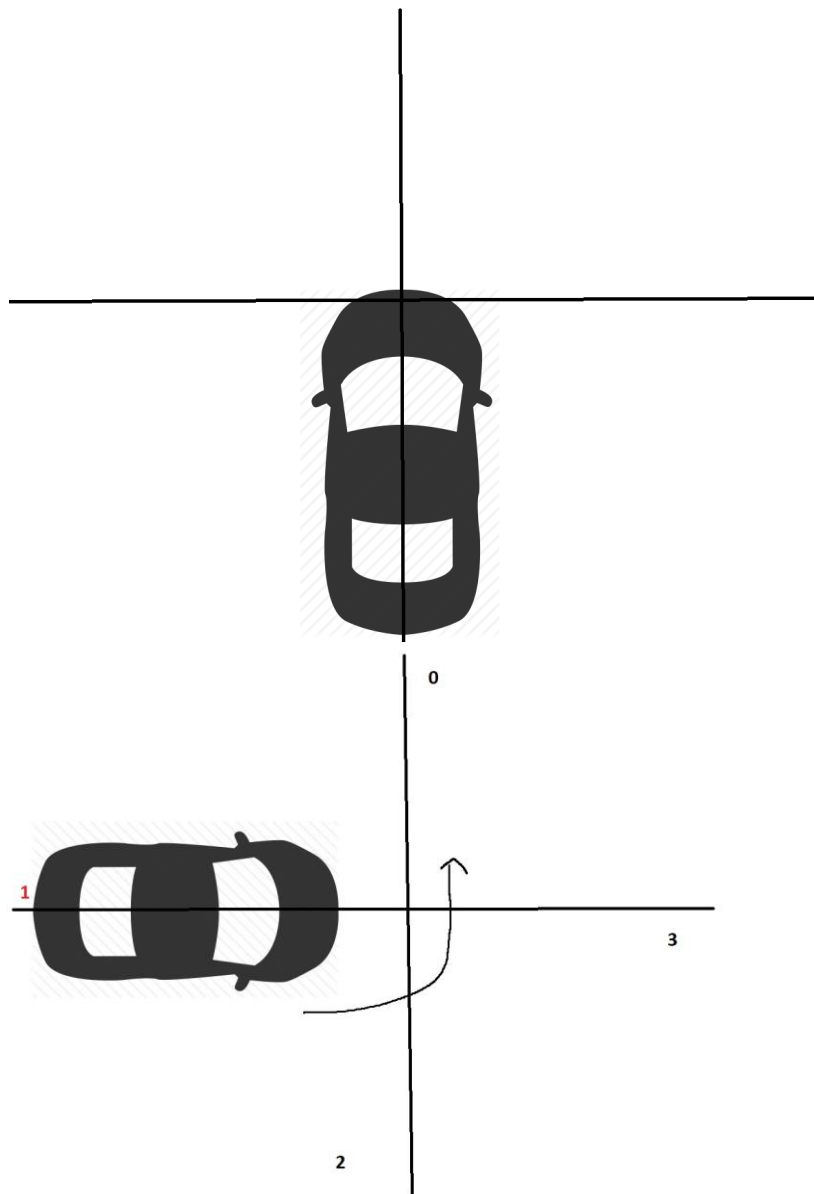
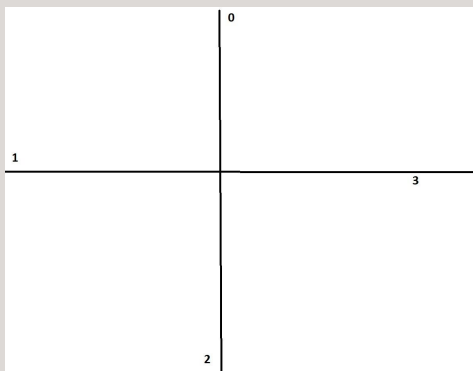
```
if (digitalRead (ln1)== LOW && digitalRead(ln2)== LOW)  
{  
    rotateRight();  
    state= Forward;  
    switchState();  
}
```

```
.. ..
```

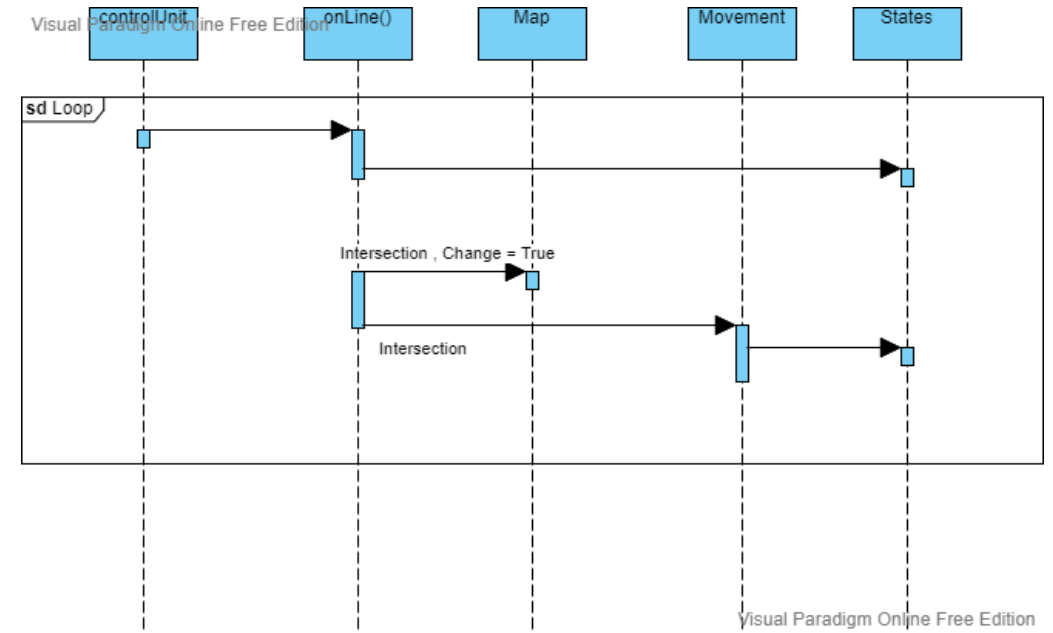
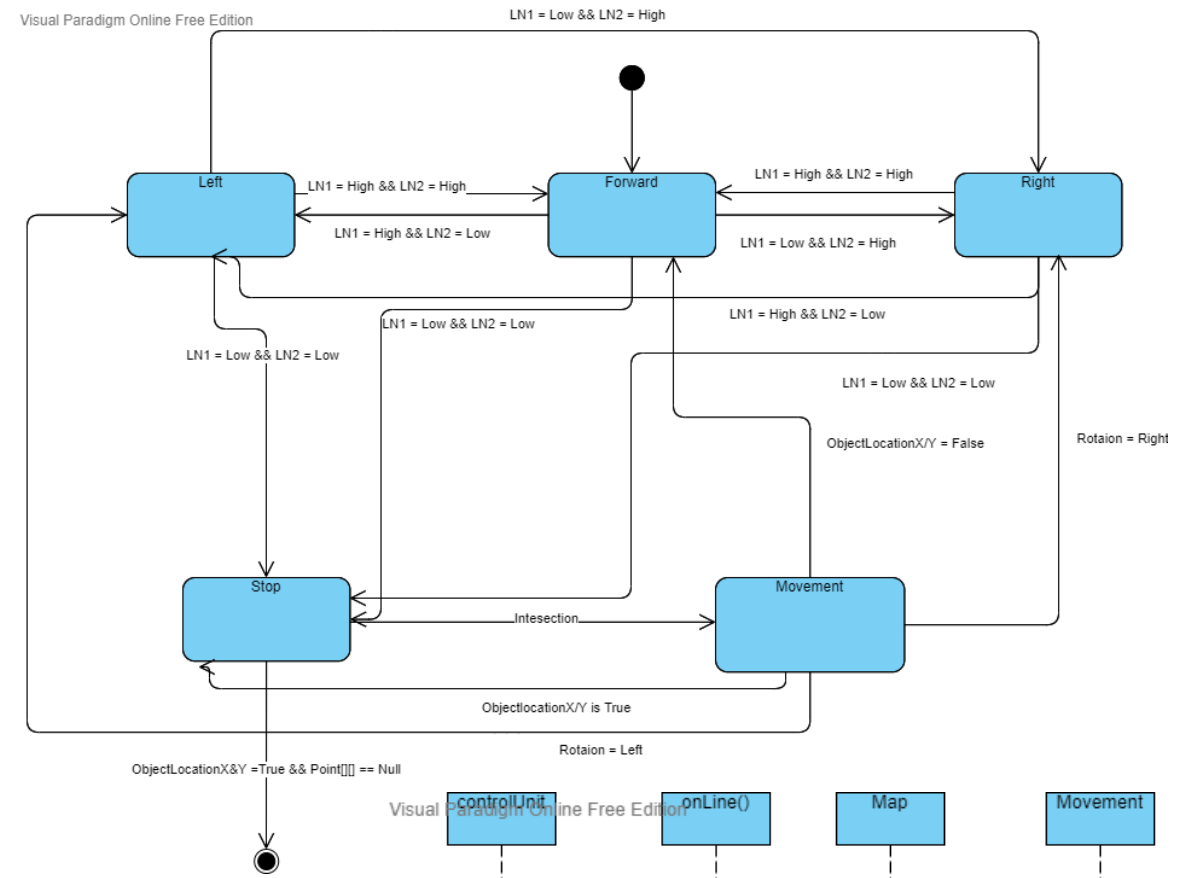
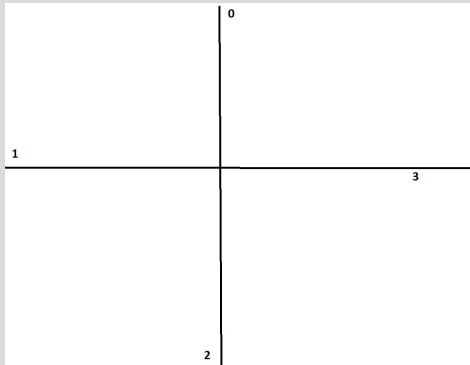
Mapping



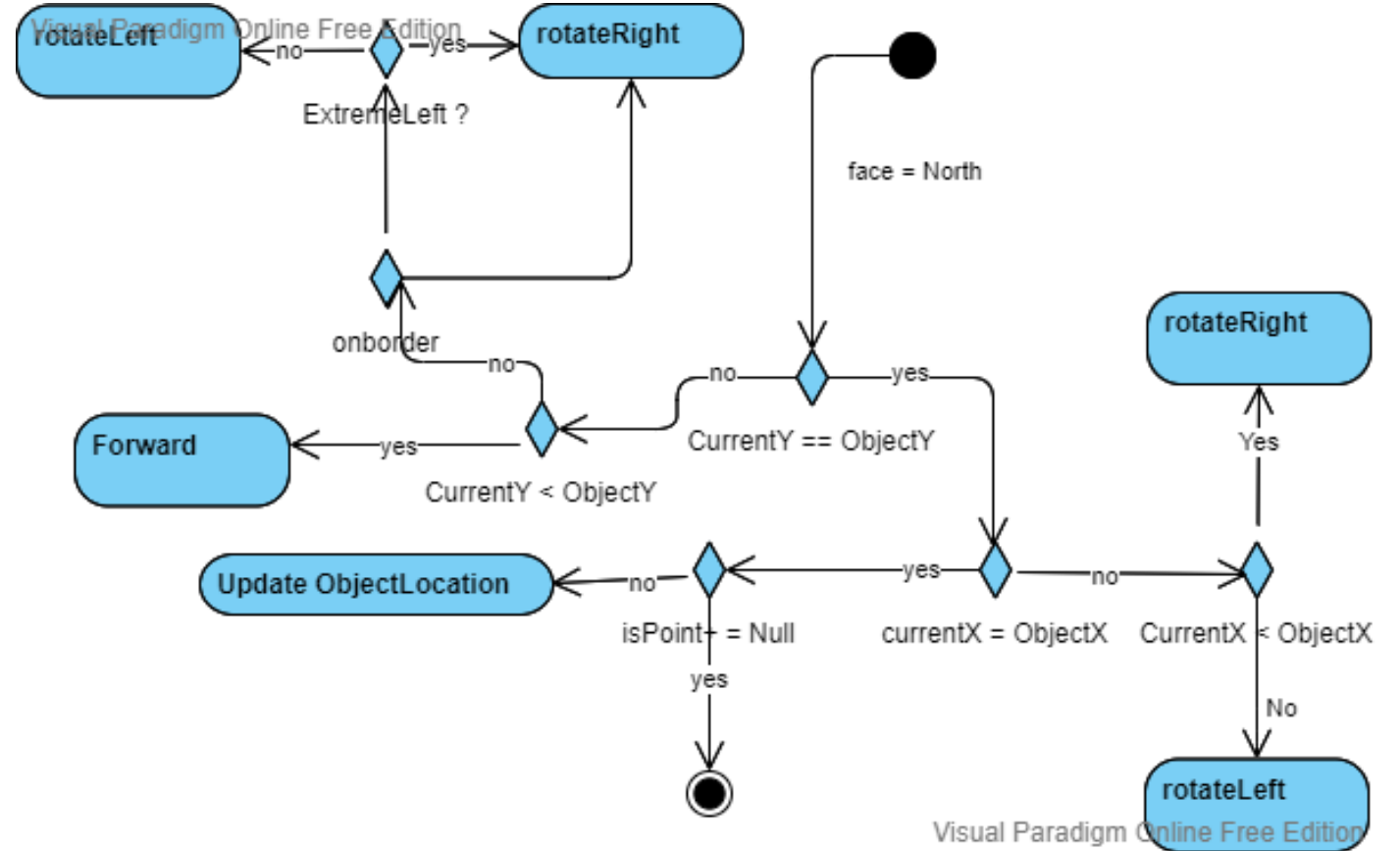
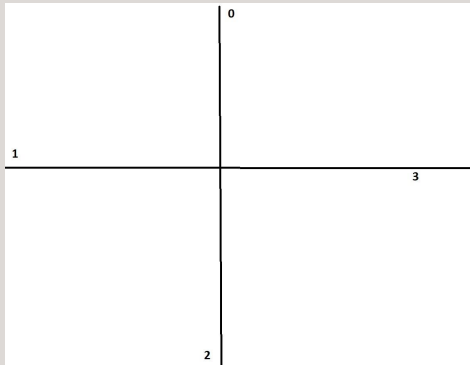
Algorithm



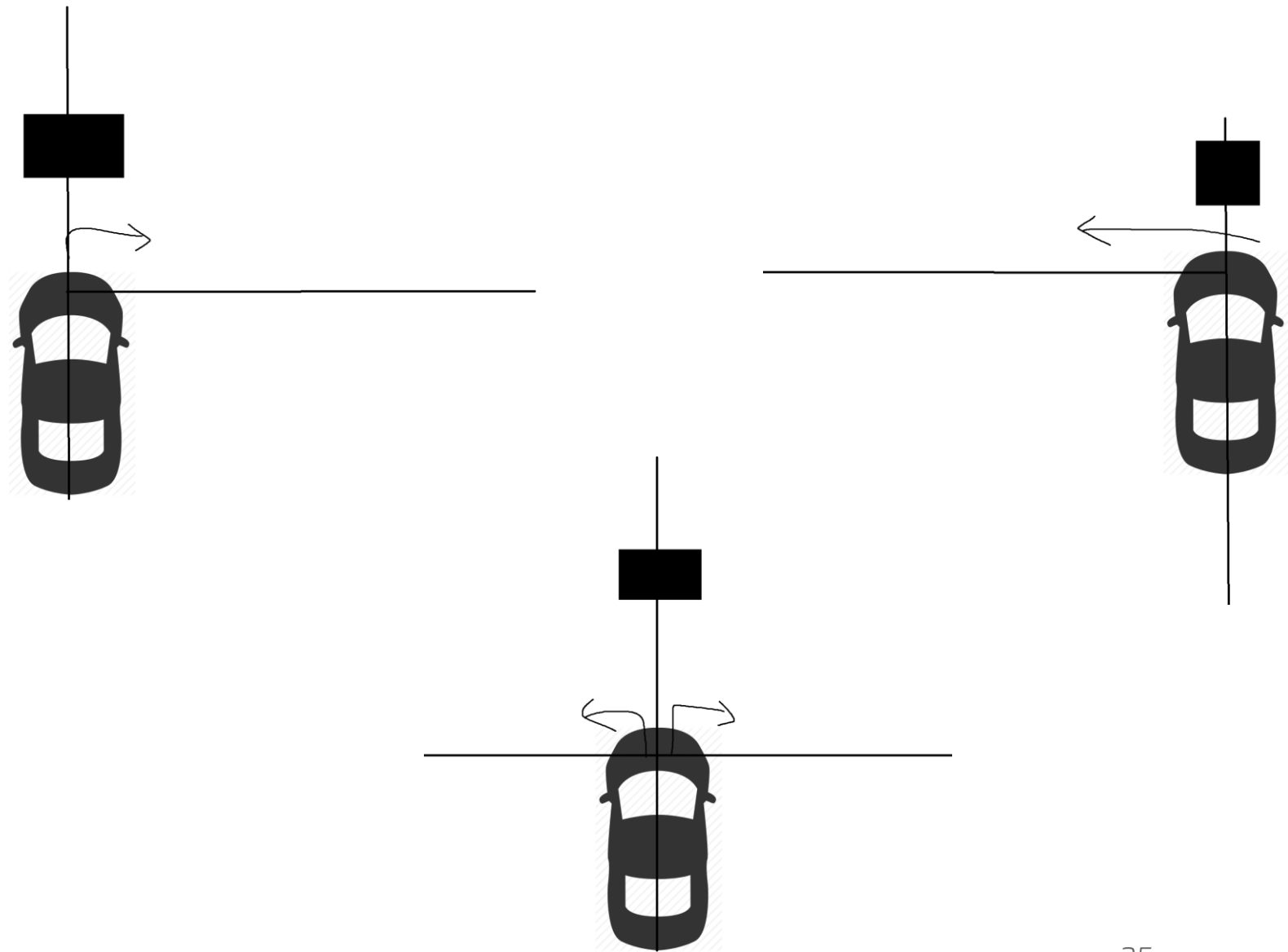
Algorithm



Algorithm



Obstacle Avoidance



Sources

1: pixabay.com

THANK YOU FOR YOUR
ATTENTION