1. Describe your core routine

Our core routine is repeatedly executed according to a specific cycle. In a cycle, our core routine do these jobs sequentially.

First, check whether each IR sensor detect color as white or black. In these part, these executions are executed sequentially :

- 1. Charges a capacitor.
- 2. Wait for fully charged.
- 3. 10000us times are required to check.
  For every 1 use, we check whether each IR sensor value is 0 or 1. And with time counter, we classify whether sensor is detecting white or black.

Second, with information about classified color value, our device choose how to act.
In basic logic, the speed of the motor varies according to the degree of bias of black.
For example, if black is biased to left, we should increase left motor speed, and decrease right motor speed.
If black is biased to right, we should increase right motor speed, and decrease left motor speed.
By this logic, we can make our device line tracing.

In addition, there is case that, even if the device is trying to trace the line, device can fail to trace the line because of rapid line change. In this case, our device rotates 90 degrees, to catch up line. This case occurs when ((1. There is no detected sensor) or (2. Only one sensor at the far end is turned on)) with (in recent past, rapid line change is detected.). With line change direction, our device chooses to rotate left or right.

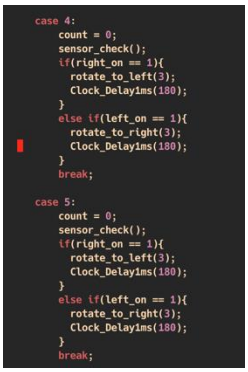
For example:

1. 11111000 -> 00000000 => turn left.
2. 11110000 -> 10000000 => turn left.
3. 00011111 -> 00000000 => turn right.
4. 00011111 -> 10000000 => turn right.
5. 11111000 -> 00011000 => just go straight!

2. Describe your noise handling algorithm

Our main noise handling algorithm was noise-rotation.

Noise rotation routine snippet:

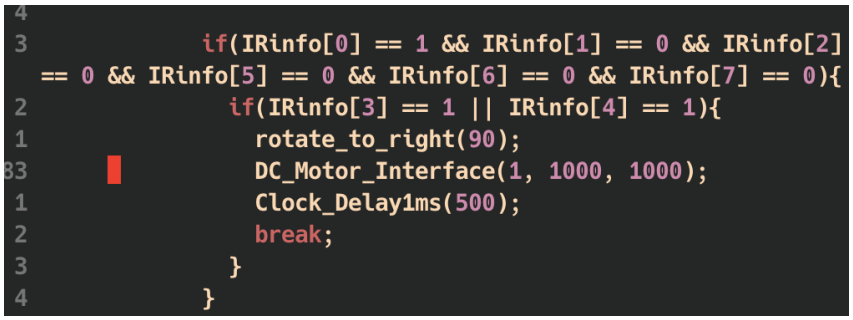


```
case 4:
    count = 0;
    sensor_check();
    if(right_on == 1){
      rotate_to_left(3);
      Clock_Delay1ms(180);
    }
    else if(left_on == 1){
      rotate_to_right(3);
      Clock_Delay1ms(180);
    }
    break;

case 5:
    count = 0;
    sensor_check();
    if(right_on == 1){
      rotate_to_left(3);
      Clock_Delay1ms(180);
    }
    else if(left_on == 1){
      rotate_to_right(3);
      Clock_Delay1ms(180);
    }
    break;
```

Noise rotation, as the name suggests, was to rotate our machine whenever noise was detected. This allowed for our machine to find its optimal route despite the track's noise density. The rotation rate was kept at a minimum (3 degrees) so that such rotation caused by noise would not hinder the machine's track search.

Specific areas of the track had to be handled through hardcoding. Our main routine sequence could not handle rather weird overlaps of track.
Hardcode snippet:



```
            if(IRinfo[0] == 1 && IRinfo[1] == 0 && IRinfo[2]
  == 0 && IRinfo[5] == 0 && IRinfo[6] == 0 && IRinfo[7] == 0){
              if(IRinfo[3] == 1 || IRinfo[4] == 1){
                rotate_to_right(90);
                DC_Motor_Interface(1, 1000, 1000);
                Clock_Delay1ms(500);
                break;
              }
            }
```

This routine allowed for our machine to escape the overlaps occurring at the bottom left of the track. When specific sensors were on, we made sure our machine escape that area.

3. peer assessment

Name: 이재욱 Contribution: 100%

Name: 정승연 Contribution: 100%

Thank you for reading.