Harry in the Maze

*

Arduino 1 – Harry

A – Setup

We just set the pinMode for the three pins connected to the IR sensors.

B – Functions

- #### existRight()

This function reads the first IR sensor to check if there is a line on the right of the robot. It returns an integer (0 for no, 1 for yes).

- #### existLeft()

This function reads the second IR sensor to check if there is a line on the left of the robot. It returns an integer (0 for no, 1 for yes).

- #### existForward()

This function reads the third IR sensor to check if there is a line on the left of the robot. It returns an integer (0 for no, 1 for yes).

C – Loop

- #### Left Hand on Wall Algorithm

This technique (by Patrick McCabe) ensures that the robot reaches the target (end of maze). This technique uses the fact that if, in a maze, you keep your left hand on the edge of the wall at all times, you would get out of a non-looping maze.

Steps of the algorithm:

- If you can turn left**,** if **existLeft()** returns true, turn left → **rotate90left();**
- else if you can continue going forward, if **existForward()** returns true, then continue → **moveForward();**

- else if you can turn right, if **existRight()** returns true, then turn right → **rotate90right();**
- else, if you can't do all of the above, this means you are at a dead end, and you need to turn around, and make your robot face the opposite direction. So, we rotate right (or left, it doesn't matter) 2 times.

*

Arduino 2 – The Safe

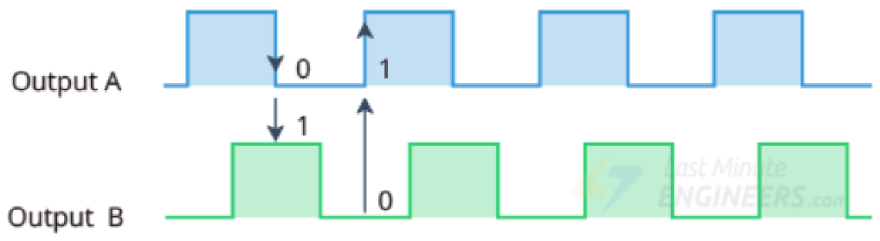
A – Setup

- ### Setting Interrupt Pins

We set the pinMode and attachInterrupt for the 6 interrupt pins that will be connected to the three encoders (2 for each encoder).

B – Functions

The following functions are repeated three times for each encoder using a different counter to measure each angle:



- ### ISR_A

When A changes, if A != B, then we're turning clockwise (incrementing counter), and if A == B, then we're turning counter-clockwise (decrementing counter).

- ISR_B

When B changes, if A == B, then we’re turning clockwise (incrementing counter), and if A != B, then we’re turning counter-clockwise (decrementing counter).

C - Loop

During the loop, we continuously calculate the three angle values measured by each encoder. If the angle values measured match the desired values (37, 10, 54), then the greenLED is turned on.

Angles are calculated by the function:

Angle = (Counter * 360 / PPR * decodeNumber)

PPR → pulses per rotation
decodeNumber → 4 since we have for each encoder 2 interrupt pins and they receive value on CHANGE
counter → counter produced by the interrupt function

(Remark: I got the modulus of the previous angle value by 360, just to make sure that if we spin more than one spin, we can still get accurate readings.).