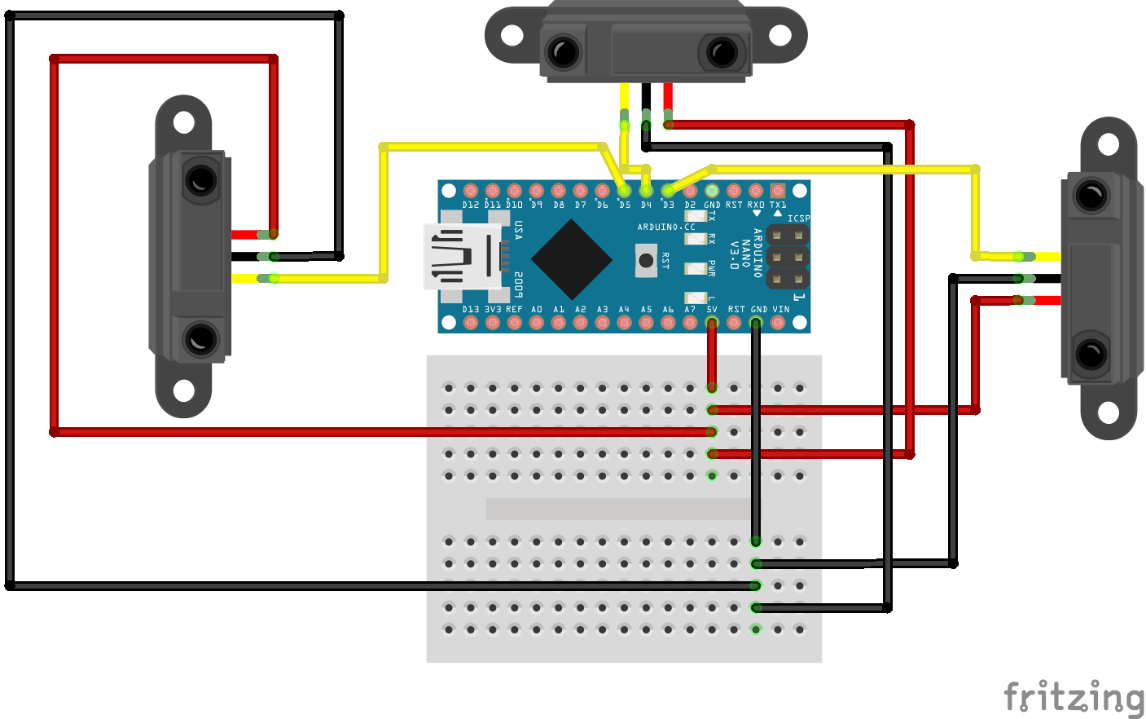
**Harry in the maze**

**Part 1: The maze:**

**The circuit:**



We used a simple circuit that utilizes 3 IR sensors to check whether the red line is there. All three sensors are connected to 5V VCC and ground. Each sensor is also connected to a pin on the Arduino to read the signal.

**The code:**

For this task, it was very important to note that this isn’t straight forward and there are a lot of situations where a decision will have to be made on which direction to go. So a coordinate system was very important for this task.

Text

Description automatically generated

A simple coordinate class would be very helpful in storing the coordinates data.

A screenshot of a computer

Description automatically generated with medium confidence

These three lines are very important in this task as they tell us which direction we are currently facing. The coordinates variable holds our current position in the maze. The sign variable holds our orientation. And the I variable holds the axis that we are going on. Using the sign and I variables together we are able to tell which direction we are heading and adding that to the coordinates implements a very strong direction system that can tell us where we currently are.



Another very important aspect is that at a crossroad we must check each road to find the correct one, which could cause an infinite loop if done incorrectly. Here we have a linked list that stores previously visited coordinates. However, if we were to store every single coordinate, this could get very memory intensive. So, we only store one coordinate of a potential path at a crossroad.

Text

Description automatically generated

This is a very simple linear search function that checks if a given coordinate has been visited yet.

Text

Description automatically generated

The first part of the loop is straightforward. As long as forward is the only direction to go, keep moving forward. Line 57 is updating the coordinates by the current direction where if we are moving on the x direction, I would be equal to 0, if we were moving in the y direction, I would be 1. If we were going in the positive direction, sing[i] would be 1 and if we were going in the negative direction, sign[i] would be -1.

Text

Description automatically generated

If there is nowhere to go, simply turn back.

Text

Description automatically generated

This Boolean checks if there are two or more roads available to take as it will be very important to check later on.

Text

Description automatically generated

If we have more than one path to take, first add the path we just came from to the visited list so that in case we get back to this point again we know not to take this road again. Line 74 is adding the last point of the path instead of the crossroad point itself.

Text

Description automatically generated

In this case we don’t have to check if it’s a crossroad because if it wasn’t either the earlier loop wouldn’t have ended or this if condition wouldn’t be true. Before taking the road, we need to check if it has been visited before. We check the first coordinate of the path as that would’ve been the one added to the visited list. If it hasn’t been visited then we move forward.

Text

Description automatically generated

In the case of the left direction being available we first need to parse the direction to know where we would be heading after turning left. We store it in temporary variables as the road might not be suitable so we don’t need to change the main direction yet.

Text

Description automatically generated

In the case of a crossroad, similar to the forward direction, we check the first coordinate of that road. If it hasn’t been visited we change the main direction and visit that road. If it isn’t a crossroad, this is our only option so we just change direction without any other checks.

Text

Description automatically generated

Text

Description automatically generated

The right direction is the exact same as the left direction with simple changes in the rotate functions.

**Part 2: The safe:**

A picture containing text

Description automatically generated

Some definitions used for easy access. Each of these encoder pins are defined to interrupt pins which is the reason this task cannot be done without the use of an Arduino mega with 6 interrupt pins.



A counter array that holds a counter for each encoder.

Text

Description automatically generated

Interrupt functions that adjust the counter according to the direction of movement. This has been done for each of the encoders.

Text

Description automatically generated

Setting up the pin modes as well as attaching the interrupts to the encoder pins.

Text

Description automatically generated

In the loop, for each of the encoders, we calculate the relative degree since the start of the program by dividing the count by 4 then dividing it by the selected PPR then multiplying by 360 to get the degrees moved. Whenever that degree reaches the correct number for each encoder, turn on the green led, otherwise keep it off.