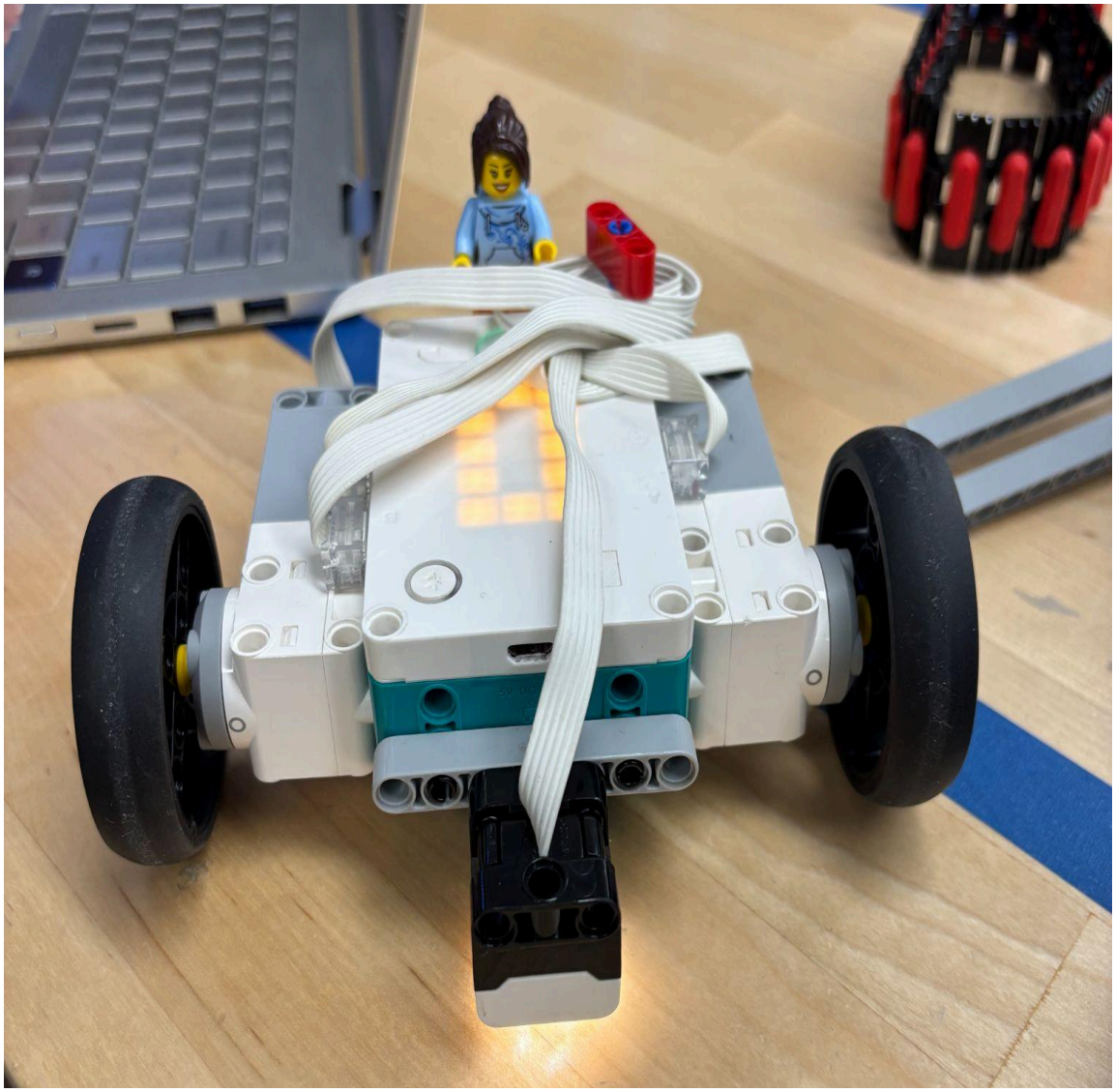
# Lab Report

*Line Following Robot Lab*



**Mary Piper Blayney, Jack Turk, and Christian Niese**

10.28.2025

Engineering Academy

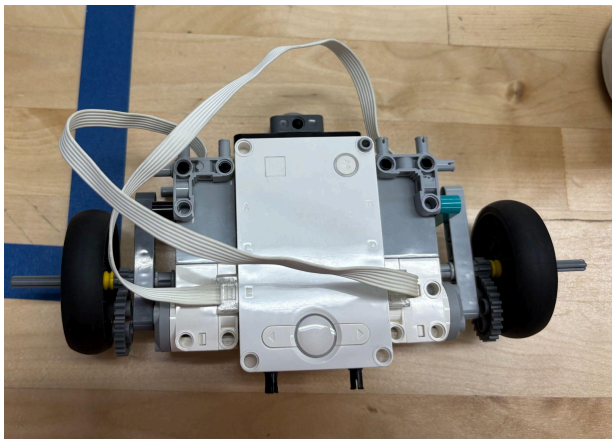
## Introduction

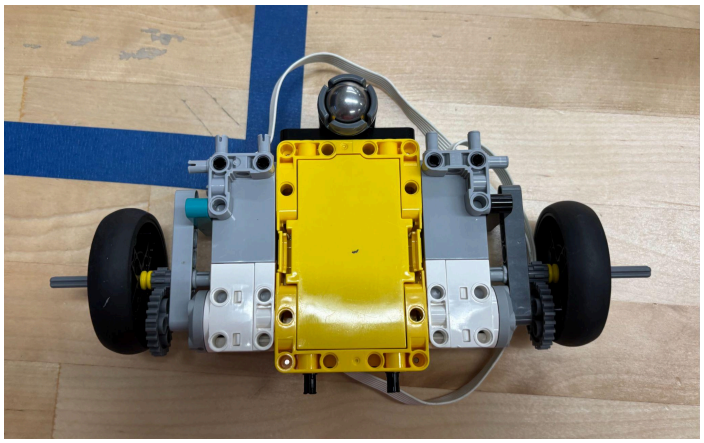
In this project, we were tasked with making a robot that could follow a lined trail through a light sensor. We had the choice of following a red or black line, which meant we had to choose coding with reflected light or color coded code. Our group chose to use reflected light and to follow the black light. To follow the line, we needed to make a robot that could drive and turn well to follow. Once all the groups in our class had made robots and coded them, we ran our robots against each other and timed to see whos was the fastest. The people who worked on this project include Mary Piper Blayney, Jack Turk, and Christian Niese.

## Brainstorming

Once we started talking about what we wanted to do with this project. Our initial plans include having our back wheels be powered and independent, and a ball wheel that makes turning a lot easier than trying to deal with another wheel on the front. We also knew we wanted our robot to be smaller and as simple as possible to help keep things moving and help us not overcomplicate the design process.Designing the robot

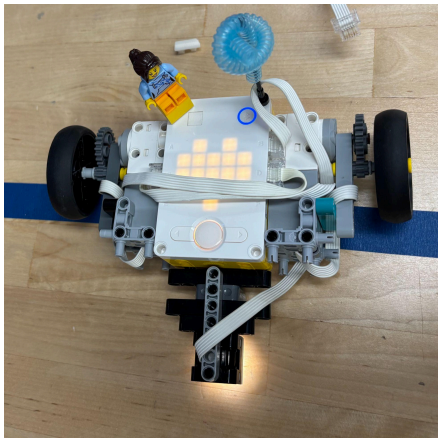
### Iteration 1

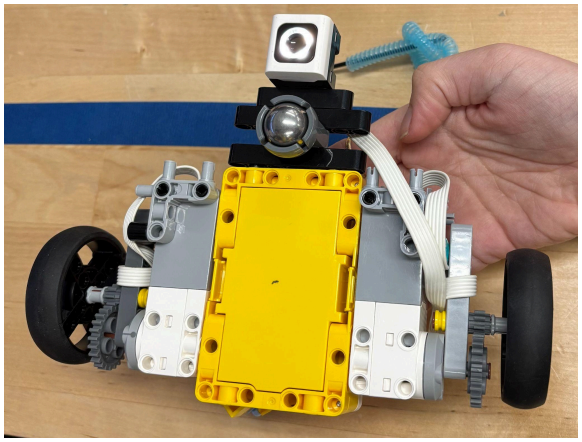




This is the top and bottom of our group's first iteration. As you can see, we used two separate motors and the ball wheel to optimize turning. We also used smaller wheels and attempted to make our robot on the smaller and more simple side. Another thing we tried with this iteration includes a gear ratio on our wheels to try to make our robot even faster. While these pictures do not include it, we had our light sensor on the back in this iteration. This iteration was a good starting point for us to improve on.

## Iteration 2





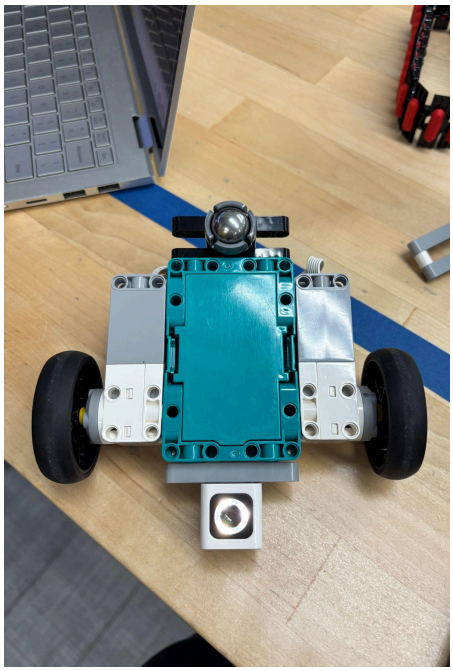
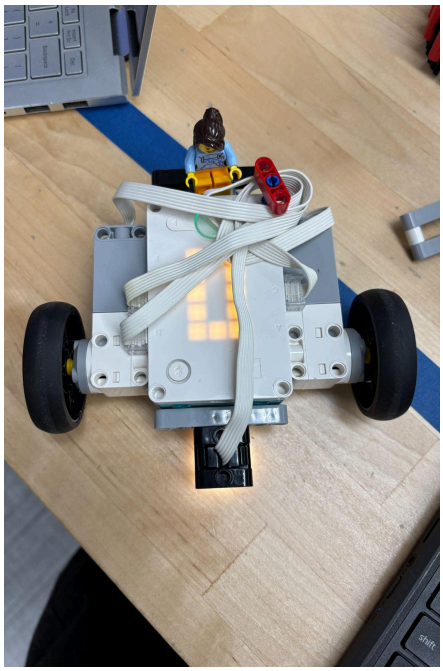
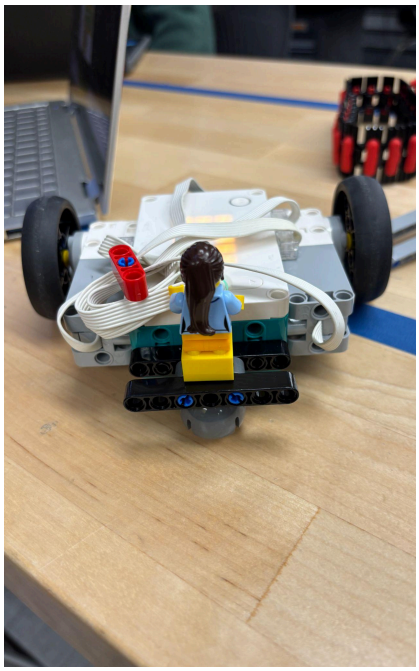
This is the top and bottom of our group's second iteration. Some differences from iteration one include where the light sensor goes (was in the back) plus added line management and a lego person. While this iteration worked ok, the front was very clunky and our gear ratio was not very helpful for this project. We kept the ball wheel and the smaller wheels, as turning was ok and the smaller wheels plus motors on the side were helpful to keep the light sensor close to the ground for readings.

### Testing Iteration 2

While in this design stage, we took a video of our robots movement to record progress. This video shows how all the pieces were working together and a baseline of what our code looked like at the time.

Click to watch video

## Iteration 3

These pictures show the top, back, and bottom of our third iteration. What we changed from the second iteration includes the placement of the light sensor (again), the placement of the line management, and the placement of the lego girl. We also got rid of the gear ratio, as we found it unhelpful and it was much easier to get rid of it. For our new line management, all cords were taken, twisted to shorten, and then placed on a rod meant to stick blocks together that was stuck into a hole on the top, and then capped with a 3 holed block. This definitely was better than having the cords wrapped around and keeps them all in one place, but there is definitely still room for improvement there.

### Testing Iteration 3

While in this design stage, we took a video of our robots movement to record progress. This video shows how all the pieces were working together and a baseline of what our code looked like at the time.

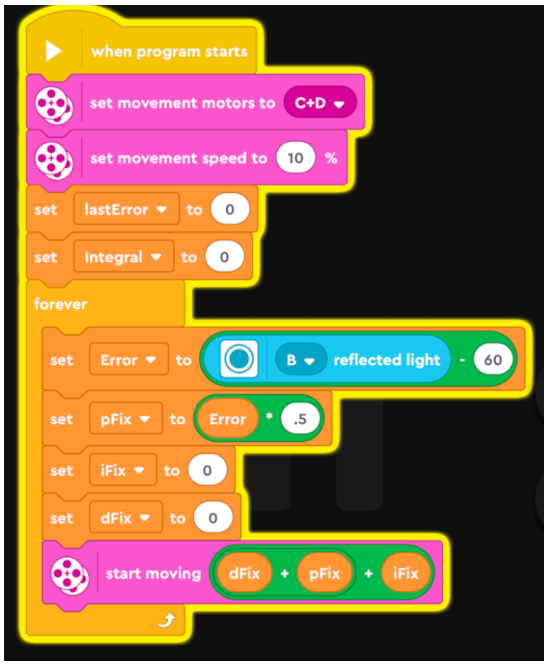
Click to watch video

## Coding the robot

Right from the start we wanted to use a PID controller for our robot to follow the line. We believed that most groups would do a standard two case algorithm, which would give us a competitive advantage. Although it had major upsides in terms of speed and smoothness, using a PID controller had a massive downside in its complexity. We had to spend the first couple of days just trying to understand what a PID controller was, how to implement it, and how to tune it. Although it didn't go as smoothly as planned, we are happy that we tried it.

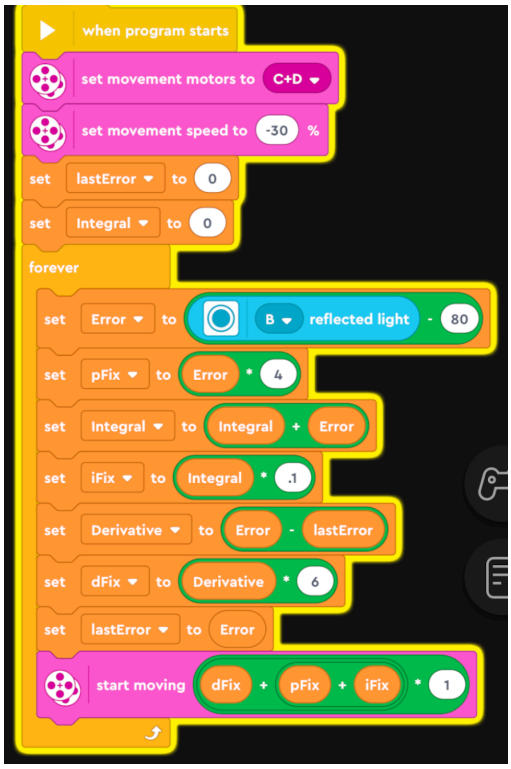
### Stage 1

We began implementing our PID controller on a simple rectangle at our table. Our first iterations just implemented a P component, and were able to navigate the course. It wasn't ideal, however, as there was no dampening of the turning and thus the robot would constantly be correcting, overcorrecting, and/or recorrecting itself to the line.



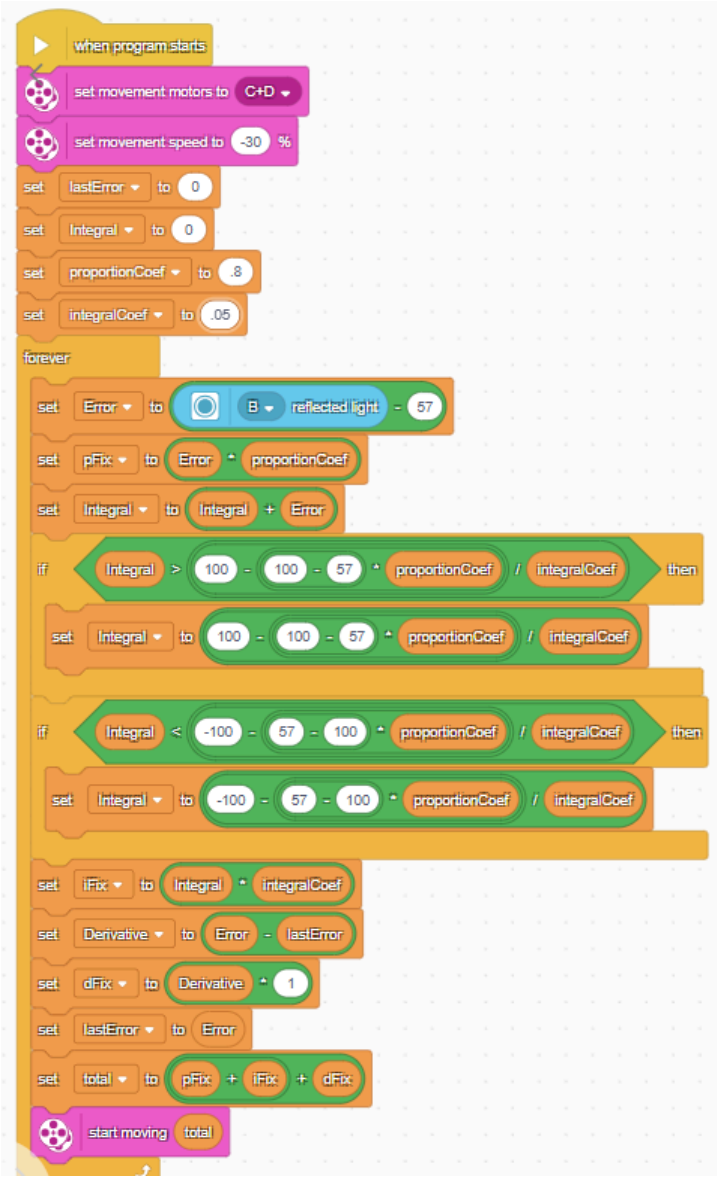
## Stage 2

Once we achieved our first working implementation of the PID controller, the real course was built and we graduated to it. We began the process of implementing both the I and D components, as well as tuning the overall algorithm. This is where our struggle set in. We spent 3 whole class periods tuning, and returning with no improvement. Sometimes the robot would increasingly oscillate, and other times it just wouldn't turn sharp enough. Looking back on it, it is very obvious why the robot struggled, but at the time we still didn't have the greatest conceptual understanding on why the robot was failing.




## Stage 3

After more research and a better understanding of PID controllers, we were able to spot our problem and fix it. It turned out that pretty much every value was too high, but especially our P component. Our P component was as high as 4 at one point, and we brought it down to .6-.8. We also increased the I component a bit to help on the sharper turns, as well as continuously tuned the D component when other components were changed. Finally, we added an upper limit on the integral to the point that the max integral + max proportion would equal the max turning rate. This helped prevent overshoot which was a common problem.

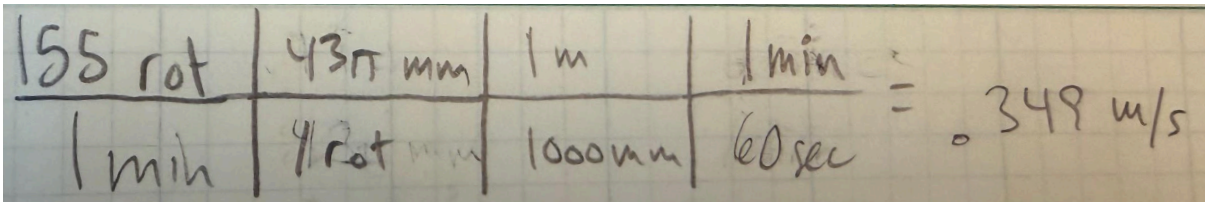



## Stage 4

Once succeeding at 30% speed, we tried to increase to 40% speed and later 50%. Unfortunately, we once again struggled to successfully tune the PID controller at these new speeds and ultimately had to stay at 30% speed so that we could complete the course. Although it may not be the fastest, it'll be smoother than 90% of the other robots.

## Calculation of the top speed of the robot

Through online research we know that the lego motors run at 155 rotations per minute. We then used dimensional analysis to convert the speed from rotations per minute to meters per second and found the top speed to be .349 m/s.




## Final run against classmates

When we competed against our teammates, we had two trials that we did. Our teacher picked a random number on the board for us to start at and if we were going clockwise or counterclockwise for the first time and the second time he picked those two but we were able to change one. The first time, we started at 4 and went counterclockwise and the second time, we started at 6 and went counterclockwise. On our first trial our time was 72.6 seconds. On our second trial our time was 71.4 seconds, which is one second better. We got fourth place out of the people in our class and we had one the least amount of oscillations during our runs.

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| Team | Trial #1 Start | Time | Trial #2 Start | Time | Fastest |
| Juliet | 2, ccw | 43.2 | 1, cw | 33.5 | 33.5 |
| Hotel | 2, ccw | 69.5 | 2, ccw | 52.2 | 52.2 |
| Charlie | 5, cw | 62.3 | 6, cw | 62.8 | 62.3 |
| Alpha | 4, cw | 72.6 | 6, cw | 71.4 | 71.4 |
| India | 5, ccw | 74.7 | 6, cw | 73.6 | 73.6 |
| Bravo | 3, cw | 133.1 | 2, cw | 121.2 | 121.2 |

## Conclusion

In this project, we created a robot that ran with a PID and a light sensor that we got to work very well with no oscillations other than while turning. When we competed against our classmates, we had the fourth fastest time in our class at 71.4 seconds. When racing, we chose to only race on the black tape which was longer but thats the one our code was best on. If we were to redo this project, we would try to get our robot faster or maybe try to color coded line too instead of just doing the black line.