Mars Rover Lab Report

Jack Turk, Joel Odugbesan, Reshika Raghavendra
Date: 12/15/2025

Poster
Video: IMG_7877.mov

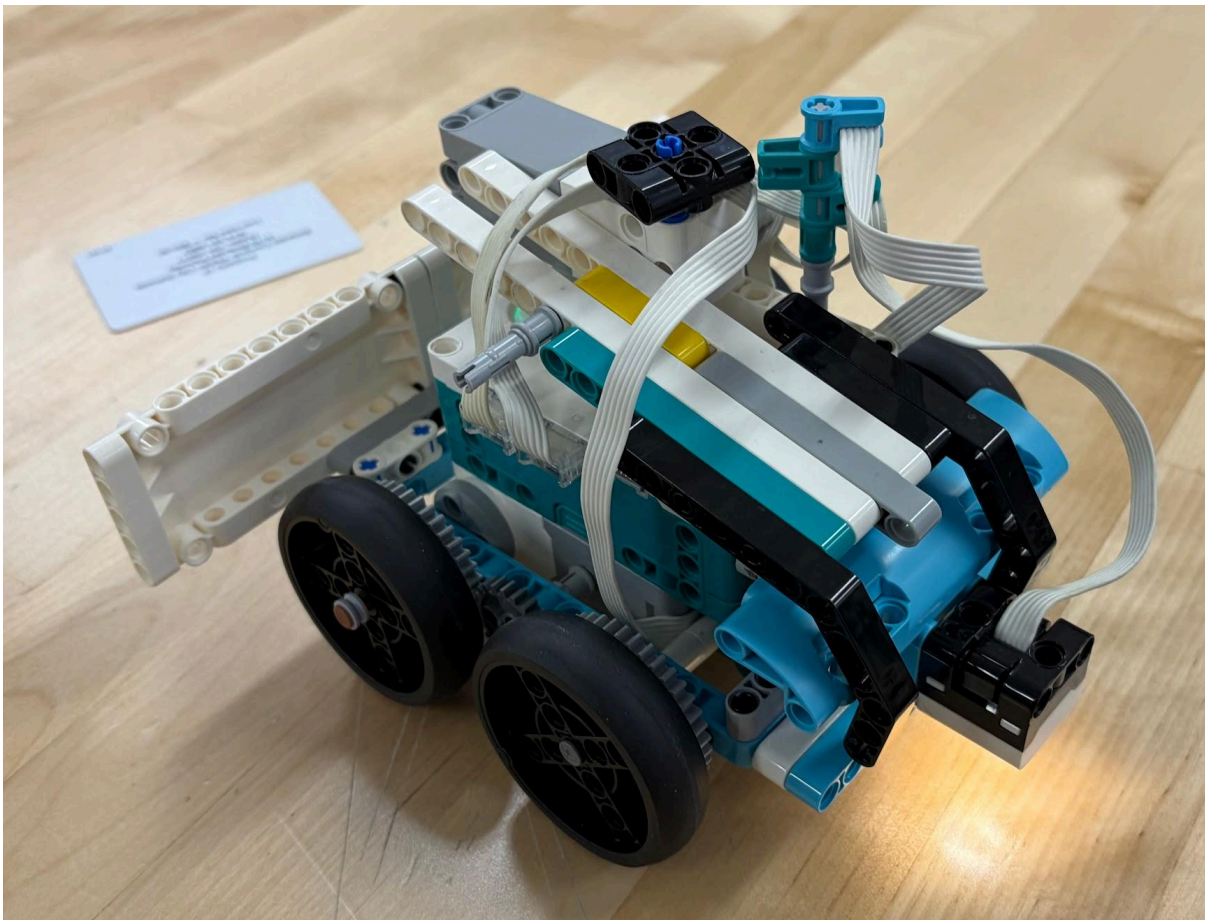
Introduction / Empathize:

Objective:
For the Mars rover mission, our team designed a robot that could complete as many tasks as possible within the ten-minute time limit. The robot was tasked to complete a series of tasks in a certain order. These tasks included placing a fuel core into the reactor, collecting the rock samples and dropping them off in the collecting area, going down and up the ramp, triggering the communications array, collecting the soil sample, going down the ramp towards the dry river bed, and coming up on the platform through a different ramp. Because we were only allowed to see the robot running every other minute, our main strategy was to code consistent movements for the robot to follow so the robot could operate without any monitoring needed.

User Needs and criteria:
To successfully complete the mission, we had to meet several user needs and constraints:

- Accurate sensors
- Controlled speed
- Rover must complete all the tasks within 10 minutes
- The robot must fit within the black starting box

These requirements allowed us to decide on our designs, which also include the mechanical arm to place the fuel core inside the reactor. We tested our design multiple times to improve our overall design.

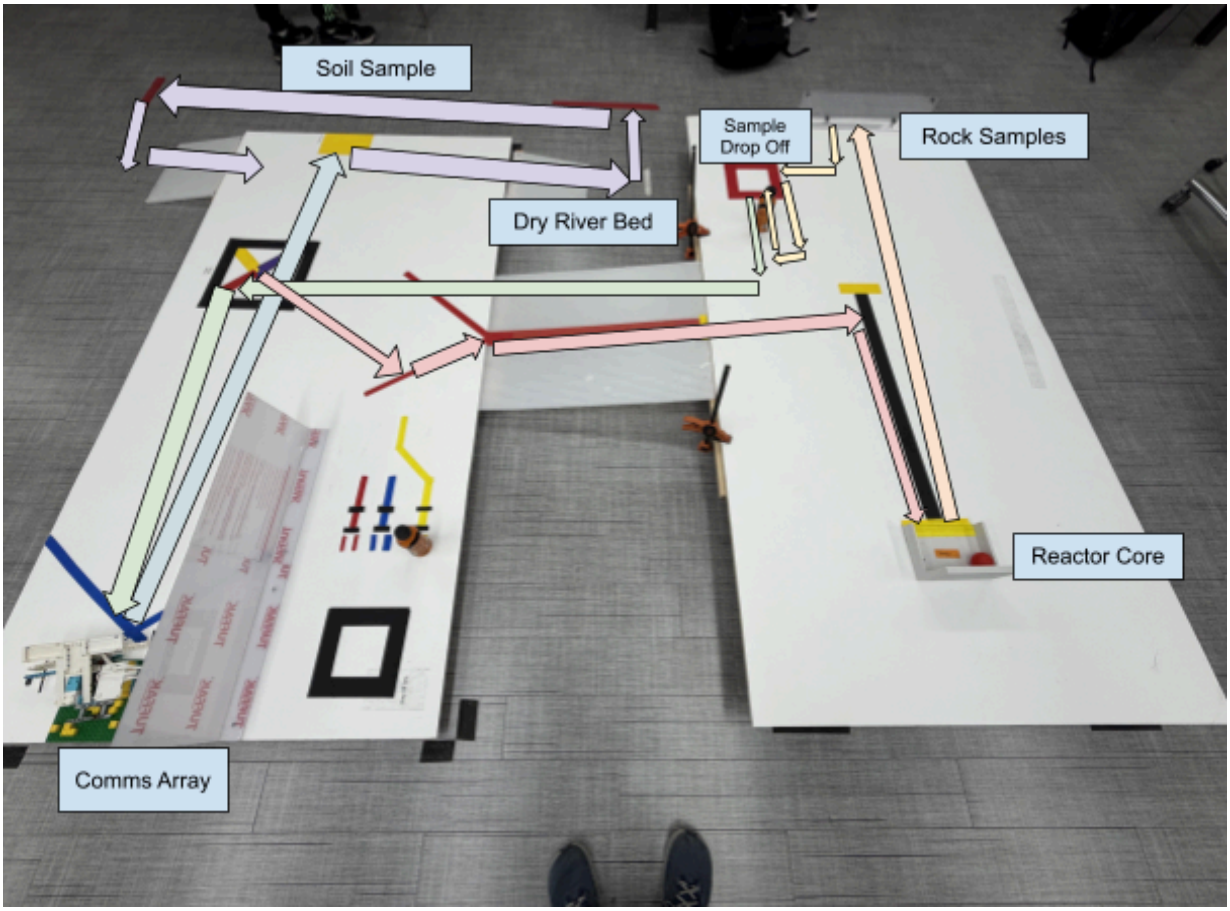


Define:

This was our team's task order:

- Placing the fuel core in the reactor
- Collecting the rock sample (the robot stays near the rock sample area for 5 seconds)
- Dropping off the sample containers in the drop-off zone
- Going down the ramp to the communications array to activate the communications array
- Collecting the soil sample
- Descending through the small ramp to the Dry river bed for 5 seconds
- Coming up on the platform using the other ramp.

Success Metrics:
We didn't really have any exact success metrics other than accuracy. The time limit is long enough that speed is not a worry. The only main focus is that the code will work as close to 100% of the time as possible. We used this metric to include fail-safes where the code didn't always work.

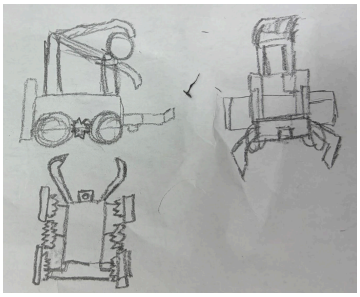


Ideate:

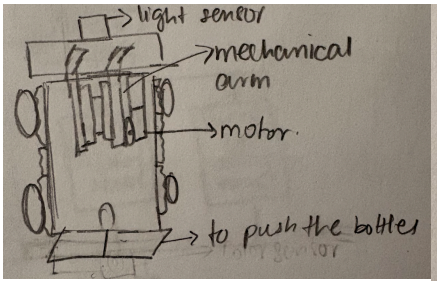
Team strategy:

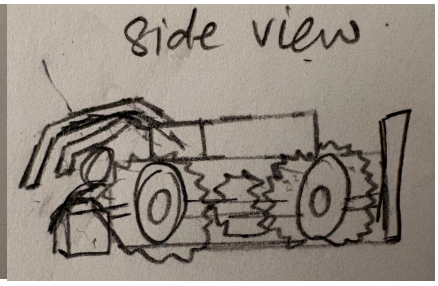
During this process, our team focused on developing a clear and efficient strategy to complete tasks on time. We began by talking about the overall task order for the 10-minute course. Once the task order was finalized, we then divided the responsibilities evenly among the team members, with each person focusing on specific tasks. Since we were only able to see the robot every other minute, the flow diagrams helped us predict what the robot should be doing while it was out of view and quickly fix any problems when we could view it.

Rover Designs:



The first Design we came up with had two distinct features. First, it had a very tall ramp on rails with a claw for the reactor core drop. This design made sure that the ball (reactor core) would always make it into the reactor without getting stuck on our rover. This design, however, had two inherent flaws. First, the tall contraption raised our rover's center of mass and lowered its stability. This, unfortunately, made our robot very prone to falling over, which would immediately end our campaign. The second problem was that the high starting position of the ball gave it a high potential energy, and often would cause the ball to bounce in the reactor and roll out. The other distinct feature we had was a funnel out front instead of a wall-like pusher. This also had two problems. The former being that if our robot was not centered on the sample drop-off zone, the funnel would bring the sample further away from the drop-off zone and ultimately miss the zone. The latter was the fact that the claw made it hard to press up against the wall and center our rover up. These problems led us to rethink our design and ultimately create our final design.



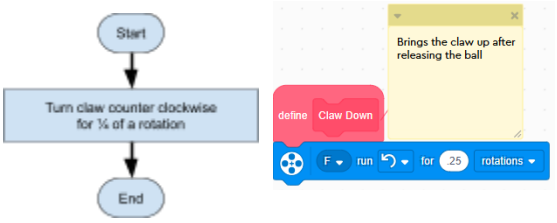


For our final design, we made a lower ball drop mechanism to increase our stability. Also, we swapped out our front claw for a flat pusher to negate any variability in our position relative to the drop-off zone.

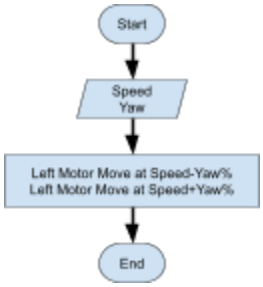
Code Notes/Flow Diagrams for sensors and motors:
During this time, we focused on getting our base functions completed. These functions included our drive straight code, line follower code, turn by angle code, and more. Here are the flow diagrams for those.

Claw Up:

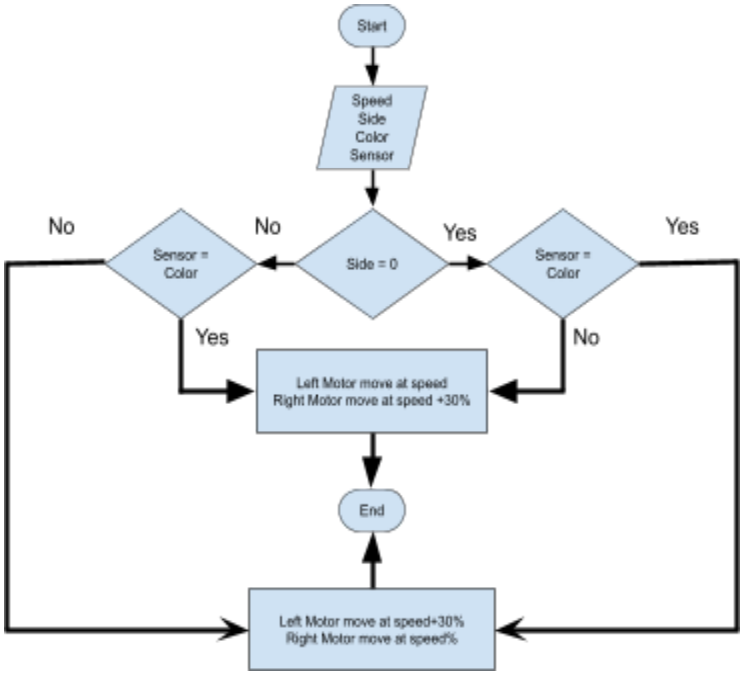
Claw Down:

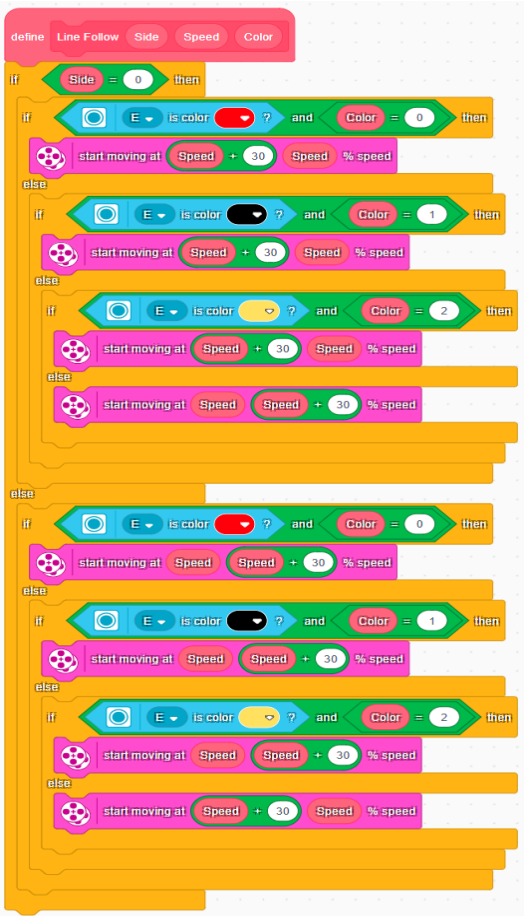


DriveStraight:

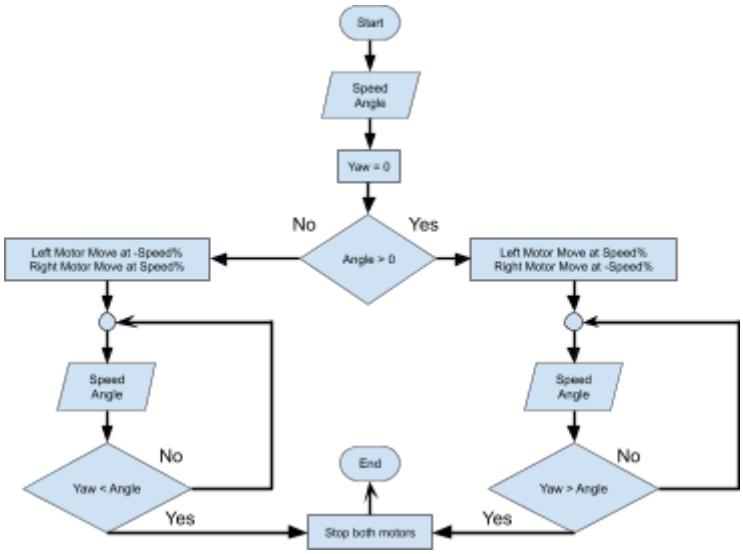


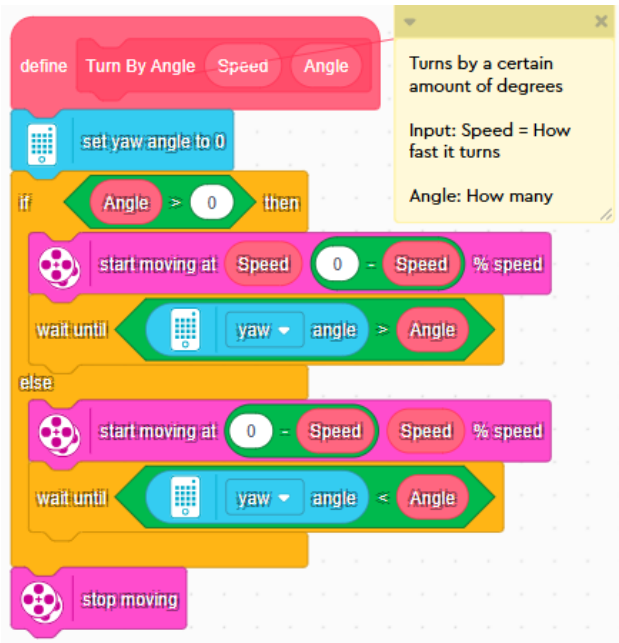
Line Follow:



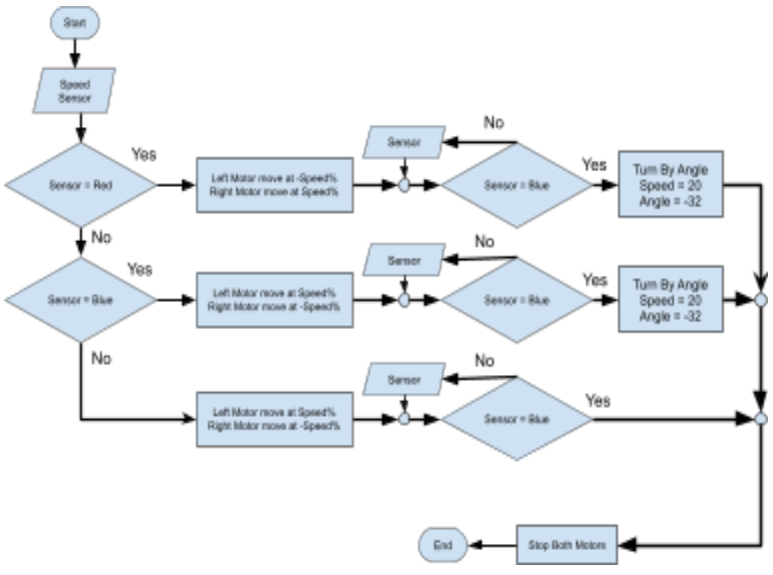


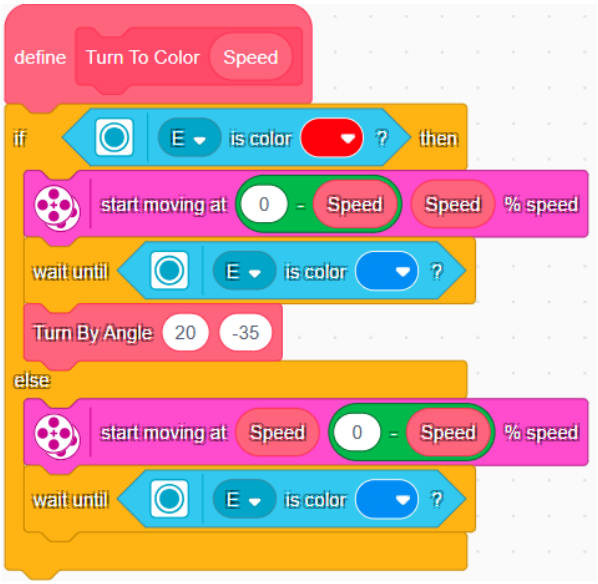
Turn By Angle:





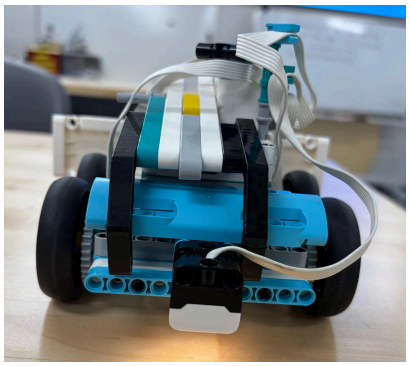
Turn to color:





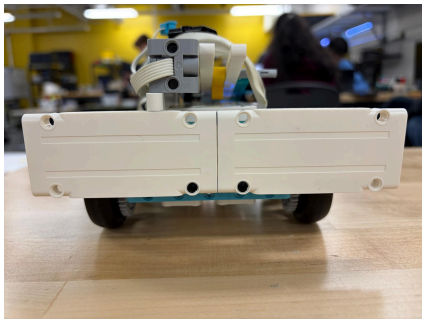
Prototype:

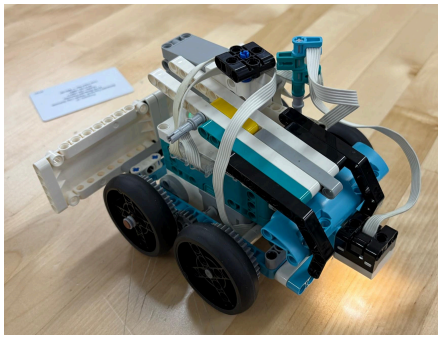
Rover Design explanation:
The robot includes three main features in order for it to complete all of the tasks on time.



The mechanical arm attached in front of our robot is built to drop the fuel core (ball) into the reactor. This made it easier for us to code the mechanical arm to just lift up and down.

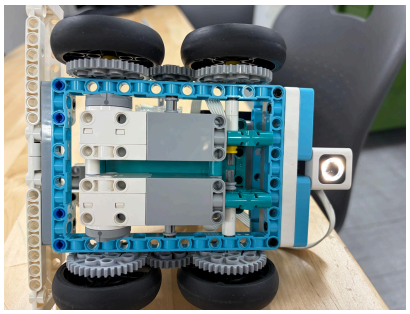
At the rear end, a straight pushing plate is attached to create a wide and even contact surface. This allows the robot to push bottles into the drop-off box without them slipping or rotating away during contact.



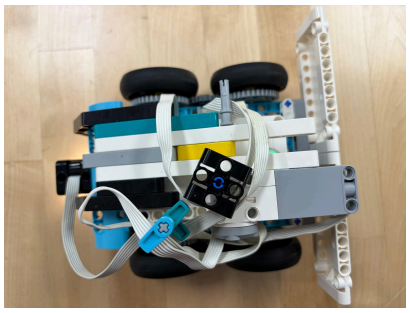


The light sensor is attached at the front under the mechanical arm, positioned close to the ground, which allows no room for sensor errors on the test day by maximizing accuracy. Since it's so close to the surface, it reduces interference from ambient lighting.

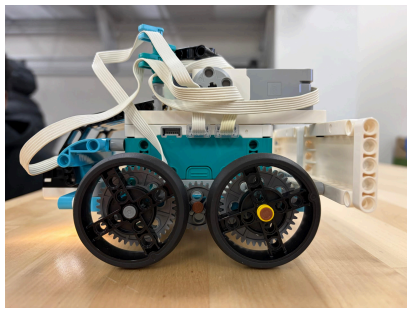
Bottom View:



Top View:



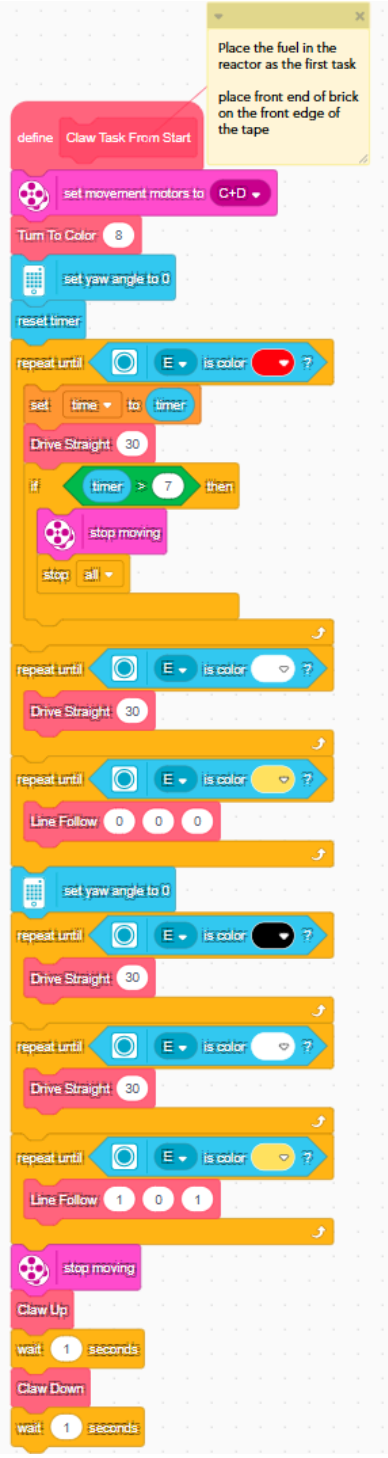
Side View:

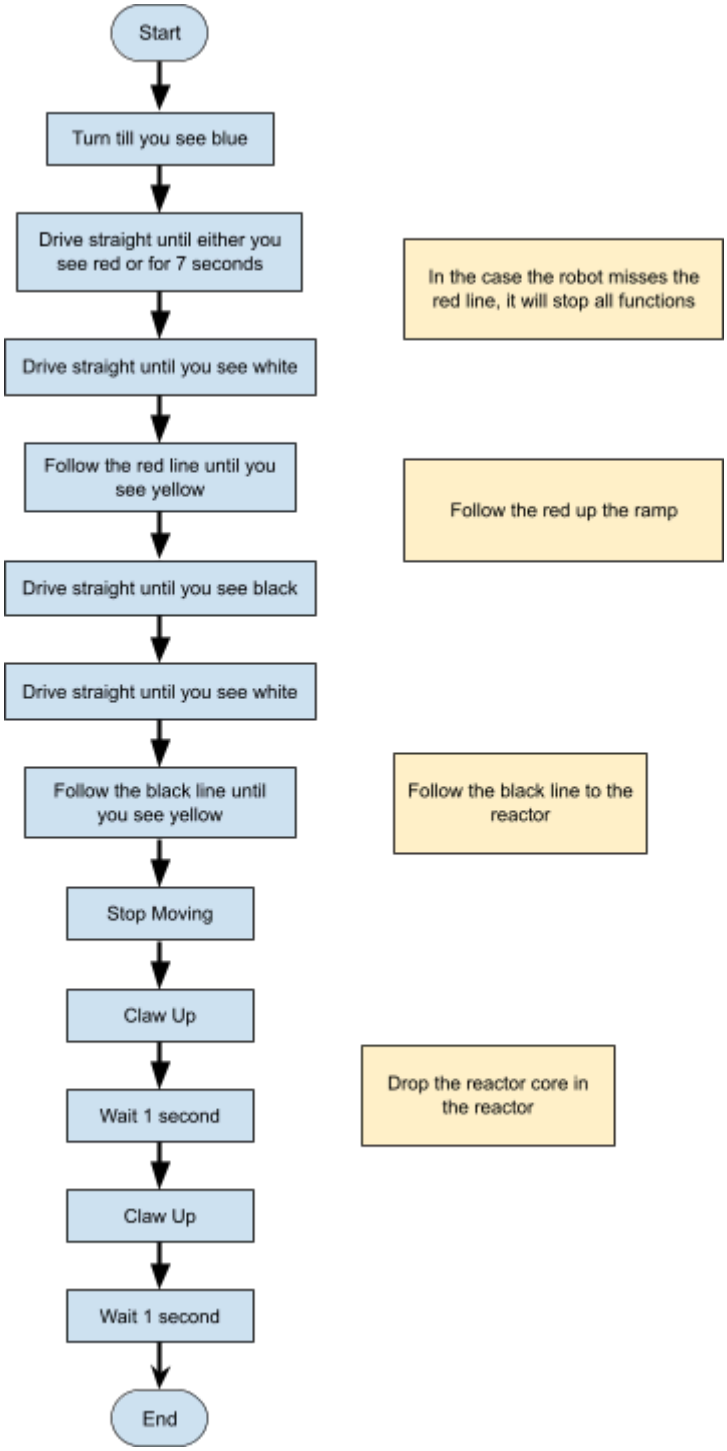


Code notes/Flow Diagrams for working tasks code:

- Reactor Core:

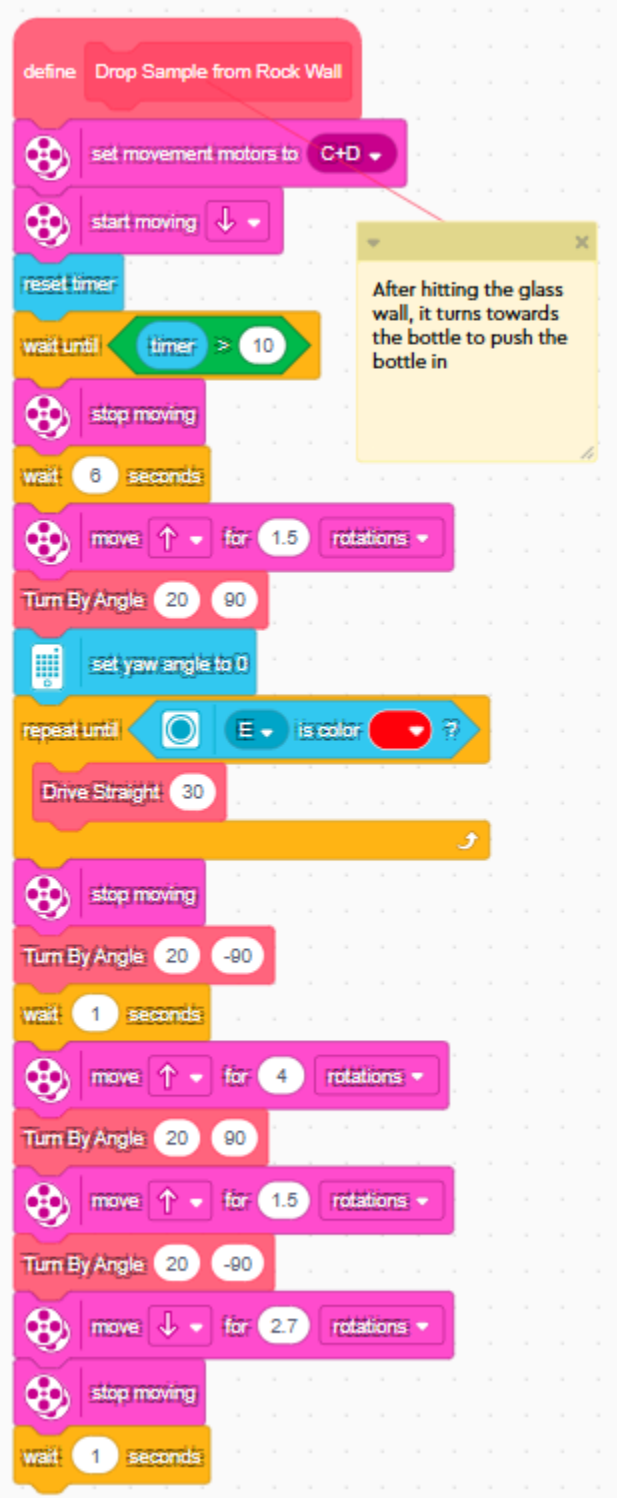
This is the first task completed. The robot starts in the starting square, goes up the ramp, and drops the reactor core in the reactor.

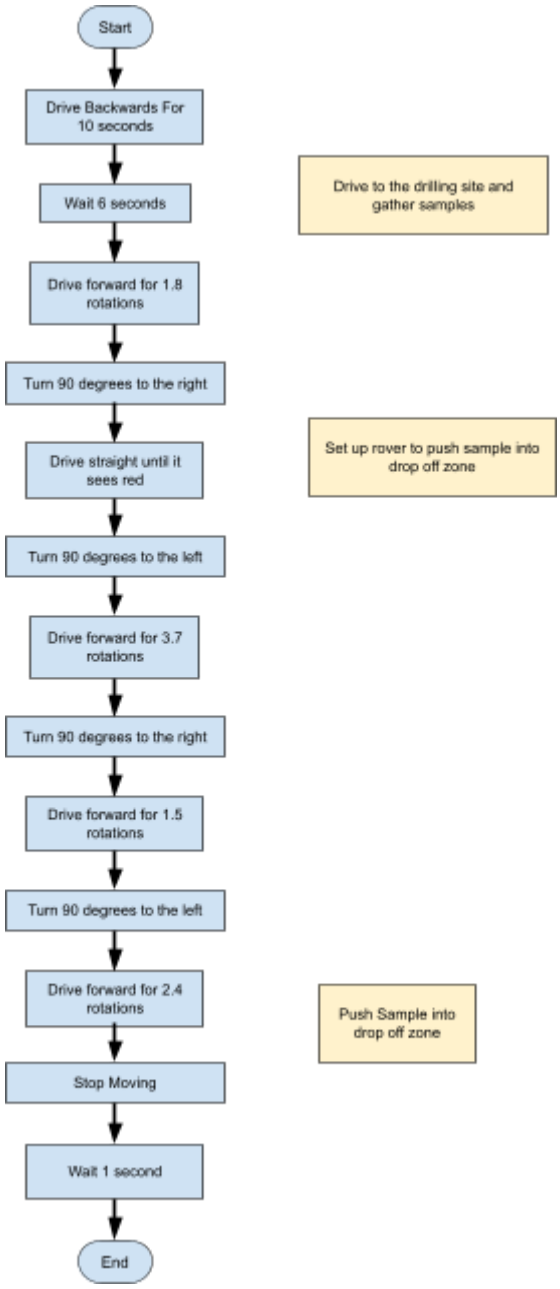




- Rock Samples and Same Drop Off:

After dropping off the reactor core, the rover drives to the drilling site. Then it traverses its way to the sample drop-off area and pushes the sample into the drop-off zone.

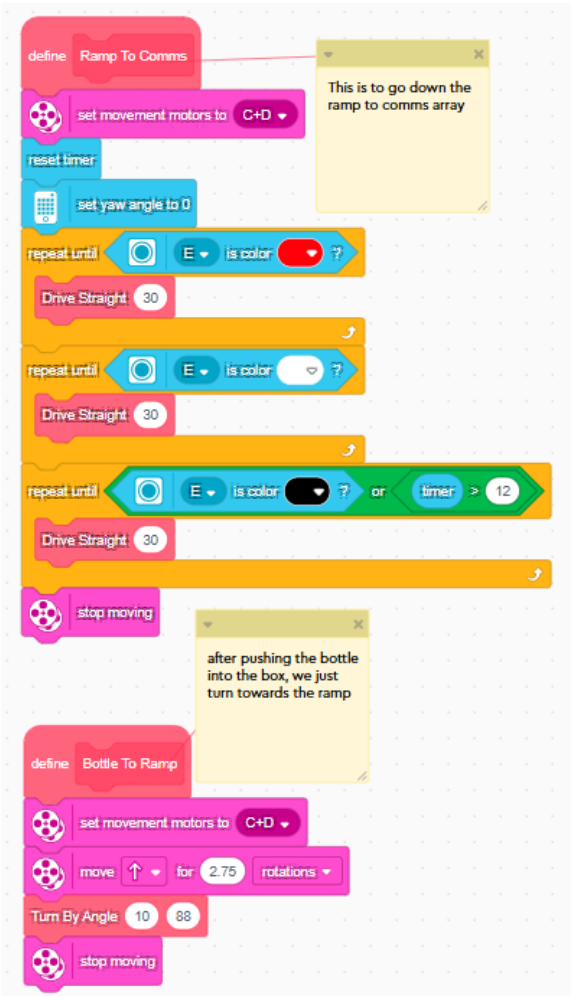


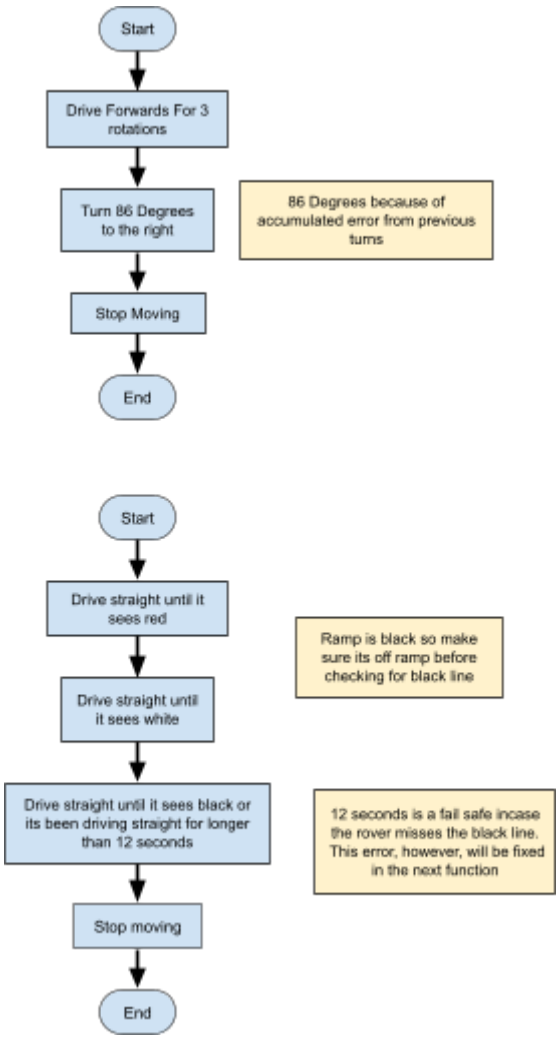


-

- Sample Drop Off to Comms Array:

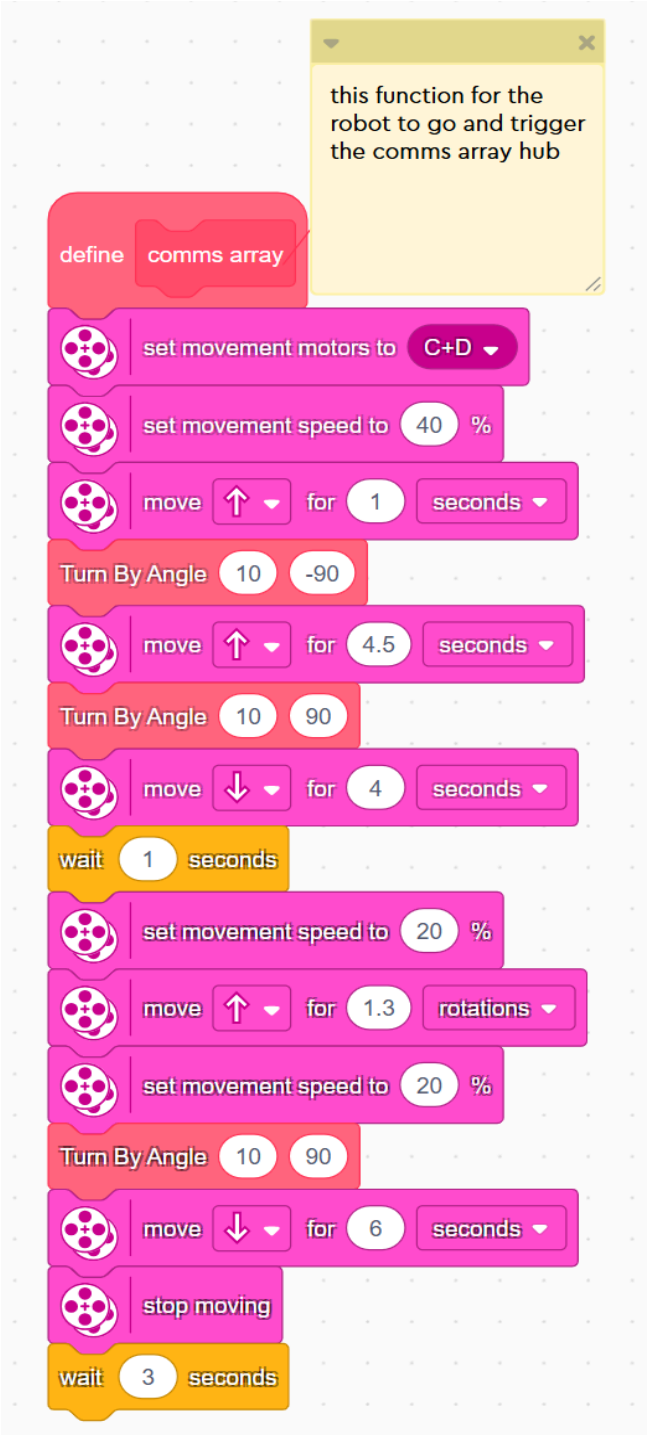
This step in the program consists of two functions: Drop off to ramp, and ramp to comms. The reason for this is that right before the rover descends the ramp, we will make sure that the rover is correctly oriented and that it will not drive off the ramp. These two programs position the rover right around the starting area for the Comms array function to take over.

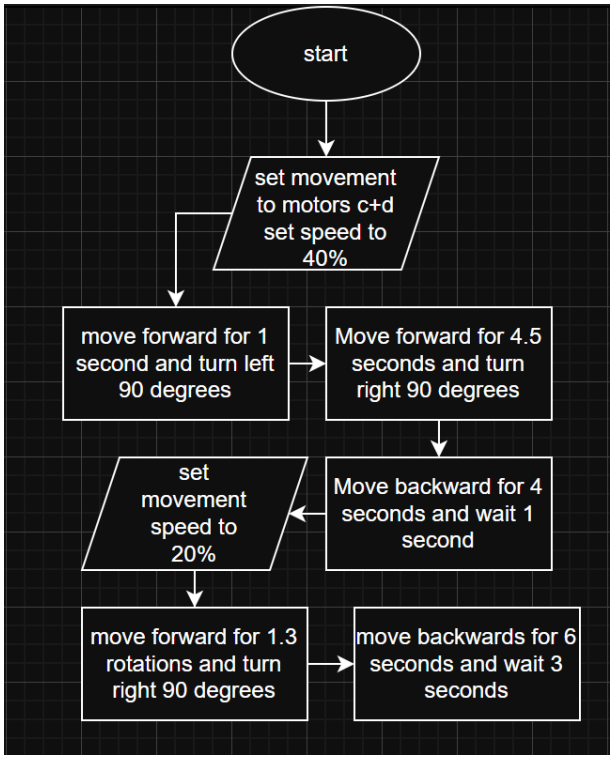




- Communications array:

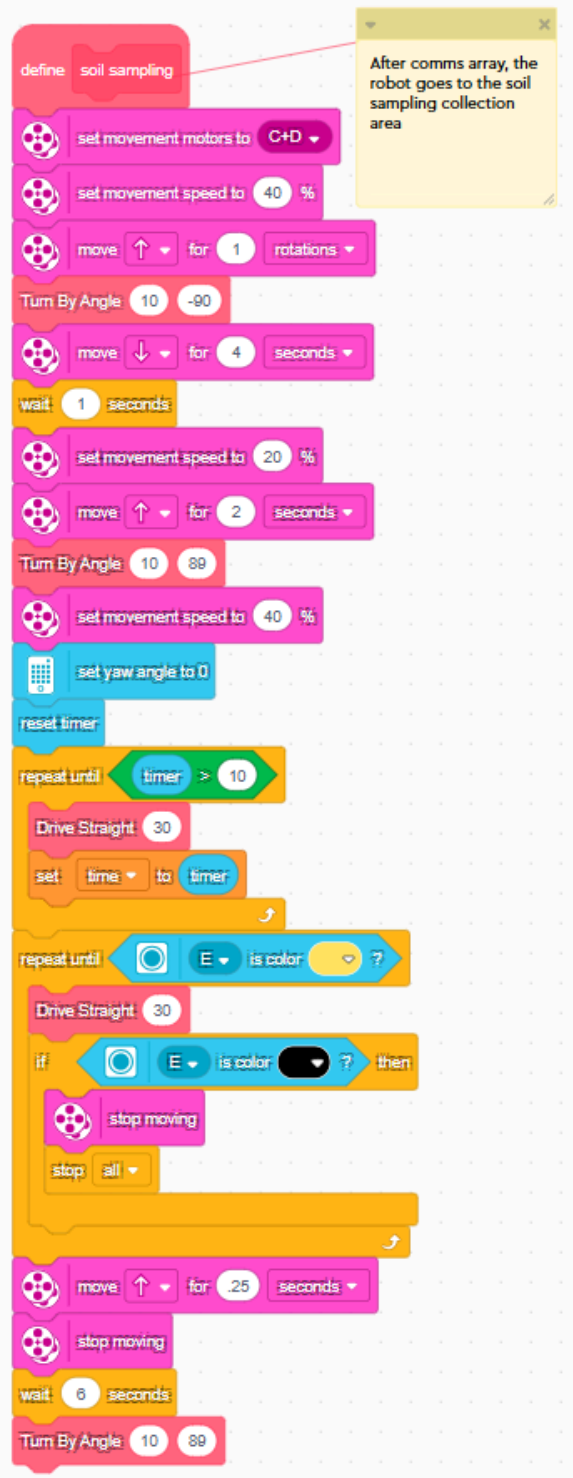
This task is performed right after pushing the bottle into the drop-off box, where the robot goes down the ramp and turns right towards the communications array, and goes forward enough to send a signal through the touch sensor to activate the communications array.

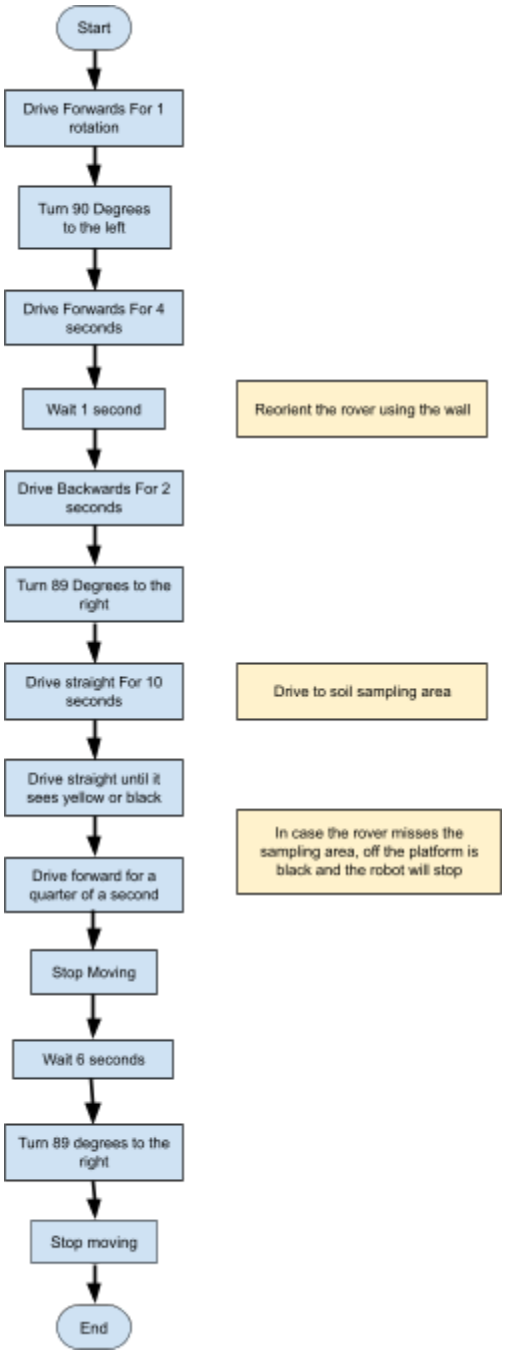




- Soil Sampling:

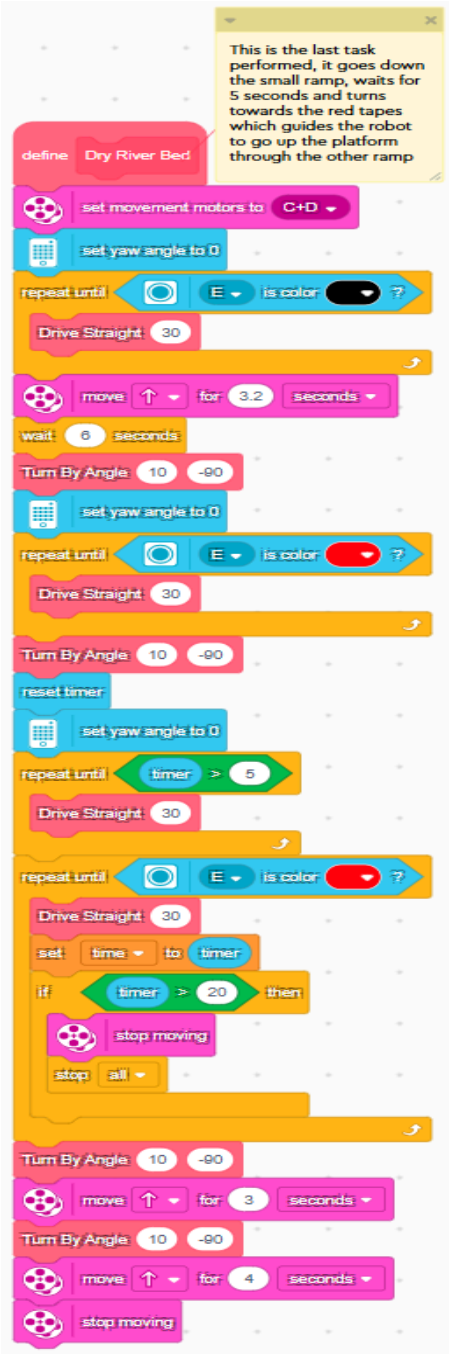
The rover collects its soil samples as its penultimate task. The task begins at the comms array, and the robot quickly uses the wall to reorient itself before driving straight to the soil area. The program ends with the rover turning right to set up the final task.

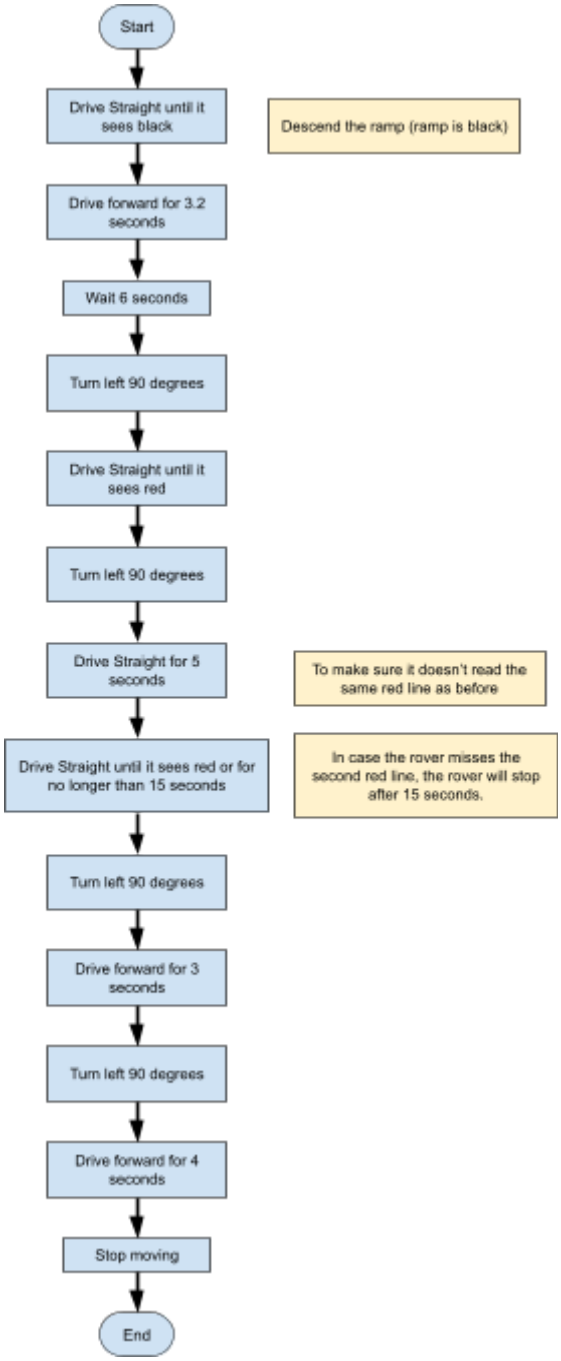




- Dry River Bed:

The rover's final task. The rover descends into the dry river bed, stops for 5 seconds, and makes its way around the platform before ascending back onto the platform using the other ramp.





Test:

Timing and Data (Thick Black Line = Stopping Point inbetween):

Task	Time To Complete	Time From Start	Frequent Issues	Solution to the Problem
Reactor Core	40 sec	40 sec		
Rock Samples	20 sec	1 min	Missing the wall	Stop moving before raising a claw
Sample Drop Off	20 sec	1 min 20 sec		
Comms Array	50 sec	2 min 10 sec	Falling off the Ramp	Stop to check alignment before descending the ramp
Soil Sample	30 sec	2 min 40 sec		
Dry River Bed	50 sec	3 min 30 sec	Missing the second red line	Failsafe in place in case it misses

Initial Runs, Problems, and Solutions:

1. Early on, we discovered that turning on the rover was very inconsistent. Each turn had a couple of degrees of margin of error that would compound over time. This accumulated error would throw off our robot for the later tasks to a point where they couldn't be completed. To combat this, we came up with two solutions. First, whenever possible, we would back up against a flat wall with our pusher to orient ourselves. This allowed us to be certain of our yaw angle as well as our distance in that direction. The other solution was breaking up the code into segments. Because we see the robot every 60 seconds, we can run a part of the program that we know works while we can't see it. We can then check the orientation of the robot after those 60 seconds, and if it’s necessary, we can turn our robot a couple of degrees either way to re-center our rover.

2. During our test run on Sol 8, our rover drove off the platform while driving from the reactor to the rock samples. It had happened a few times before, but it was during the test run that we figured out why. Once our rover reached the reactor, it didn't stop its motors until after the claw was lifted up. This meant that any turning left over from the line follower code would compound while the claw was lifting up and would leave our robot off-center by 30+ degrees. As a safety precaution, this is the first point at which we will stop our rover to check its orientation. The first task takes 40 seconds, so we will only need to wait 20 seconds to make sure the rover is good to keep going.

3. Throughout the code, there were commands that involved driving straight until the light sensor saw a color. This would work really well, as you knew for certain your distance in the forward direction after this. The only issue was that if the rover missed the color line entirely, then it would go forward forever. To counteract this, we introduced failsafes into our code. During error-prone drive to color commands, we would introduce a timer and only go forward for a certain amount of time. This would ensure that if the rover missed the colored line, it would only continue for a couple more seconds before ending all programs.

Implement:

Mission Objectives vs. Actual Performance:

Going into our final mission, we planned on doing everything apart from the second sample container. We followed through on our plan of doing the reactor core blind, and everything else while being able to see the robot, to have a good trade-off between our time constraints and our success rate. In our final mission, we finished all of our pre-planned objectives with 5 minutes to spare and ended up manually traversing the rover to the final sample container to complete an extra objective.

Which improvements made the biggest difference.

The improvement that made the biggest difference was our redesigned back wall. Early on in our rover's development, we had a claw/funnel design to push the samples into the drop off zone. We later switched to a flat back wall in case our robot was ever not lined up correctly and this proved significant as during our run our robot was off to the left of the drop off zone and still managed to complete the drop off.

Suggesting improvements for future missions

As we completed every objective we attempted, there's not much that we could've improved on. However, one major thing that we could've improved on was our initial robot design. The wheels we used weren't the most straight wheels ever and made our turning clunky. Also, putting the pusher and the light sensor on opposite sides of the rover made the first sample drop off difficult as we didn't know when to stop pushing the bottle. Overall there is not much else we would change.

Personal Reflection:

- *Reshika:*

  Throughout the Mars rover project, I played a key role in coding during the second half of the course, focusing on tasks such as the Communications Array, Soil Sampling, and Dry River Bed. I primarily timed the robot to determine the duration needed to travel between points, and I programmed the communications array based on these measurements. For the Soil Sampling component, I instructed the robot to reverse for several seconds and stop when its sensor detected a yellow color, then pause for five seconds. The success of this project relied heavily on selecting the optimal algorithms to ensure the robot operated efficiently. During testing, we also decided to deposit the second sample into the drop-off box. As a group, I believe we mainly improved on coding for the robot. We encountered several challenges while coding for the dry riverbed area. As a Team, we divided our tasks equally to work efficiently on time. Our communication among the team was pretty strong, which helped us fix any bugs on time. One of the mistakes I made while coding for the communications array was not resetting the times and the yaw angle of the robot. I then fixed it with the help of my teammate. In the future, I would probably collect more media and sketch designs to see what we need to do, rather than just getting straight to building.

- *Jack:*
  My main role in the mars rover project was getting everything set up for my teammates. I was the only group member with prior programming experience, so I went ahead and created all of our base functions. These included our turning by a certain angle, drive straight, and line follower code. Also, I created the programs for the rover to start from the starting square and drop of the reactor core. Other than that, we all helped build the rover and comms array, so I had a part in those, and I just helped my groupmates when I could. One skill that I learned was to focus on the repeatability of code. I had to program the robot with the thinking that it needed to work every single time and not just one time. Another skill I improved on was writing my code legibly with comments so that my groupmates could understand my code and use it/improve it. Early our group had some conflicts with what tasks we would complete and who in the group would complete each task. These conflicts, however, were solved without any problems. One mistake I had was in the starting square. I never accounted for the robot starting in the blue, and thus the first time I tried it starting in the blue the rover missed the red lines. Luckily I caught this mistake because on our final mission our rover started in blue. One thing I would do differently would be plan more ahead. Our robot's first stage was very sloppy and the build was all over the place. Our second design was much better and I wish it was our first design.