5XIB0 - DBL Venus Exploration

Final Report

Team 23

Full Name	Student ID
Adam Lasota	1793411
Ana Sirbu	1829858
Bastiaan Schaap	1499386
Daniel Tyukov	1819283
Jiahui Que	1886339
Ioanna Panagiotopoulou	1785729
Malik Weren	1788701
Martijn Oosterhuis	1860615
Stefano Bracciali	1880330
Steven Zollinger	1889818

Hand-in deadline: June 25, 2023

Eindhoven, July 9, 2023

Contents

1 Problem Statement

Robots equipped with multiple sensors and actuators are expected to explore the planet Venus and navigate through a maze of steep hills and craters to find three hidden rock samples and bring them back to the predefined laboratory while staying within the boundaries of the map. The robot has to avoid the hills and the craters at all costs, or else the robot will be lost. This exploration mission comes with several challenging tasks. The key challenges can be formulated as follows:

- Navigate around the planet without colliding with any obstacles (hills and craters).
- Detect and collect the rock samples.
- Navigate back to the laboratory with the rock samples, again without colliding with any obstacles.

In order to overcome these main challenges, the robot is equipped with two servo motors that drive the wheels, a servo motor that operates the gripper (up and down, open and close), and an ultrasound sensor as a baseline. Additional sensors and actuators can be added to improve the functionality of the robot. The added components will consist of infrared sensors, which will detect the black lines around the craters, and a main beacon at the center of the laboratory, combined with smaller beacons on top of the robots to get them back to the laboratory.
This way, the hills can be detected by the ultrasound sensor. The craters, the boundaries of the map, and the rock samples can be detected by the infrared sensors. The beacon on the laboratory, combined with the infrared sensor on top of the robot, will make sure that the robots can navigate back to the laboratory.

2 System Overview

The robots Fig. 2.2 are both equipped with an Arduino Uno micro-controller board that will receive all the inputs of the sensors and control the according actuators. Both robots will sense the environment with:

- 6 infrared sensors
- 1 ultrasound sensor

The infrared sensors will consist of three short-range sensors to sense the ground environment and one long-range sensor to detect the laboratory.
The actuators on the robots are:

- 2 wheel servos
- 1 gripper servo

These actuators will drive the robots around the map and collect the rock samples when found.

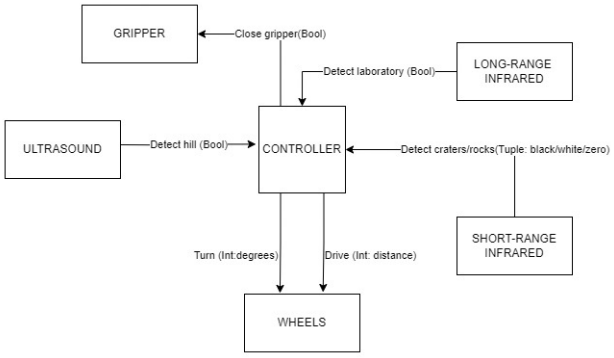


Figure 2.1: System diagram of the interconnected sub-systems.

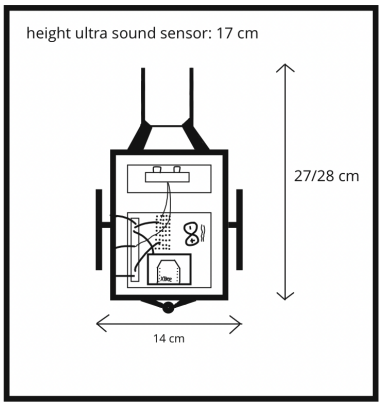


Figure 2.2: robot diagram

Figure 7.1 clearly shows the initial system that was devised and the interaction between all different sensors and actuators connected to the Arduino Uno controller. Based on this diagram, the following sub-modules were created:

- Ultrasound Detection
- Infrared Detection
- Laboratory
- Movement and Rock Sampling
- Communication
- Camera Detection (Extra)

All sub-modules were tested and completed separately before they were integrated into the final strategy. In the following sections, the working of all sub-modules will be concisely explained, as well as the tests that were performed. Section **??** will discuss the final strategy of the system and the trade-offs that were made between the different sub-modules. It will also describe all the issues that were encountered during the final integration of the sub-modules and how they were solved.
The Camera Detection sub-module is an additional idea that came up during the project. For this idea, a camera connected to a Raspberry Pi would be mounted on one of the robots and use OpenCV library functions to sense the environment and act accordingly. However, this would not comply with the assignment rules, so for that reason, the decision was made to add it as an additional sub-module and explore the possibilities of this idea. Section 8 will elaborate on the findings of this additional solution.

3 Ultrasound Detection

3.1 Component Description

The robot is mounted with a PING ultrasonic sensor, the purpose of which is to detect the mountains on the planet in order to avoid them. The ultrasound sensor is connected to the robot's Arduino Uno microcontroller through three pins (+5V, GND, SIG(signal)) and attached to a servo motor that enables it to rotate with a range of 180 degrees. The sensor detects the distance from the closest object by sending an ultrasonic pulse (in the range of 40kHz) and by receiving the reflected pulse. The time it takes for the pulse to hit the closest object in its range and to bounce back is then converted into centimeters using the speed of sound (340m/s). Moreover, the sensor is programmed to constantly rotate, spanning a 105° angle in order to detect the obstacles that are not just right in front of the robot. This is executed with a rotation angle of 60° to the left of the robot and a rotation angle of 45° to the right of the robot, with respect to position 0, which is straight ahead. This choice of movement for the ultrasound servo motor was decided based on the skewed position of the sensor on top of the robot.

3.2 Testing

Testing the ultrasound sensor consisted of pushing the boundaries of its detection capabilities, such as range in length and field of 'view', in order to figure out how walls/mountains are perceived when encountered by the robot. Because the field of view is more narrow than expected, the servo on which the ultrasound is mounted has to provide a wider range of detection.
For the frontal detection of the ultrasound sensor, a threshold of 980 microseconds, or approximately 16 centimeters from the sensor itself, has been set to allow the robot and its grippers enough space to recalculate and adjust directions. The tests consisted of simulating situations in which the robot drove toward an obstacle located at different angles. By checking the serial monitor in Arduino IDE it was possible to check whether the sensor would return an obstacle.

Figure 3.1: Ultrasound Sensor

4 Infrared Detection

4.1 Component Description

The infrared sensor consists of two LEDs (emitters), a photo-transistor in between and resistors before each component. While the LEDs emit infrared light, the photo-transistor receives the reflected light and changes its current which sequentially changes the output voltage, depending on the intensity of the reflected rays. The darker the color of the tape, the more rays are absorbed and respectively, the brighter the color, the more rays are reflected back.

The infrared sensors consist of three pins (+5V, GND, A/D) which are connected to the Arduino Uno microcontroller. Three infrared sensors are placed on the front of the robot facing down. These sensors are responsible for rock sample detection. Another infrared sensor is placed facing forward in order to detect the laboratory and finally, two other sensors are responsible for detecting borders and they are placed on the sides of the robot, a few centimeters in front of the wheels.

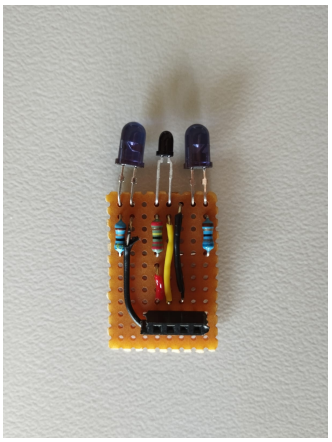
The infrared sensors' analog signal is the Arduino input, which converts the analog signal to a digital integer. This integer is a value from 0 to 1023, which is equivalent to a voltage between 0 and 5. A selected threshold divides the range into two regions. If the value given by the ADC is in one area it is defined as true and the other region will be defined as false. In addition to a single threshold, multiple thresholds can be assigned to the sensor, allowing it to define multiple regions. Instead of a binary true or false, the sensor can provide an enumerated value corresponding to a specific color within each region.

4.2 Testing

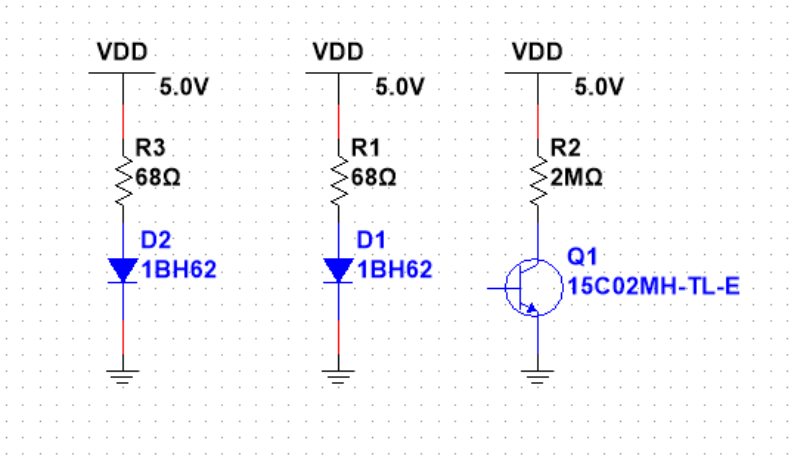
For the infrared receiver part of the sensor, three different components could have been used. One of the alternatives is a photo-resistor, but the sensitivity is too slow for the purpose of the sensors and it would be

affected too much by temperature. The other alternatives are a photo-diode and a photo-transistor, which are both viable options, but the sensitivity of the photo-diode is lower than the photo-transistor. So, it was decided to use a photo-transistor, which has a good response time and the circuit is simple to build.

The first trial was done using a breadboard. The components used are the THUS5400 IR emitter and the SFH309FA-4 photo-transistor. With the datasheet of the components, an appropriate resistance was calculated. The calculated resistance for the IR emitter of 100 Ω was tested. The circuit was tested with a single IR emitter and photo-transistor. After testing it was decided it would be better to obtain brighter light, so it was replaced with 68 Ω, and added an extra IR emitter to obtain better output values received by the photo-transistor. A large resistor was used at the collector pin of the transistor to obtain a larger voltage drop and thereby pulling the voltage into its ideal operating region. After testing with different values a resistor of 2 MΩ was the best option.



(a) Circuit design of the IR sensor in real life



(b) Circuit design of the IR sensor in Multisim

For the software part, a function was made that takes the rolling average of 50 readings. However, extensive testing revealed that this approach caused delays in the robot's actions due to the system's layout. To improve response time and eliminate any potential delays, it was ultimately decided to forego the rolling average calculation. This decision was made after thorough testing to ensure the reliability of the sensors. The calibration of the threshold values was accomplished after mounting the sensors and examining the values given by the sensors. A suitable threshold was determined and set accordingly.

During extensive testing of the function with multiple sensors, it was discovered that interference occurred between the sensors. Various methods were attempted to isolate and identify the source of the problem, but unfortunately, the root cause could not be determined. Despite this, it was decided that the data obtained from the sensors remained reliable enough to serve its intended purpose.

5 Laboratory

5.1 Strategy

The objective of the following sub-module is to get the robot back to the lab after picking up the block.

5.1.1 Chosen strategy

The robot will have a mounted infrared sensor as a tower, which will be used to detect the PWM signal of infrared light from the laboratory beacon. The robot will rotate until the lab is detected, and then drive forward, toward the lab. When the robot detects the walls of the lab, the robot will have a predefined flowchart-based set of actions based on the input from the infrared sensors.

5.1.2 Alternative strategy

An alternative strategy that was considered for the project was introducing the concept of a virtual map, where all the movements of the robot are registered in a database. When the robot picks up the block, the reverse sequence of movements will be executed to return the robot to its original position in the lab. After some group discussions, the lab beacon and IR strategy with a flowchart was chosen as that method is more reliable as it doesn't require very precise movements from the robot, which was realized to be very hard to accomplish with the robot in use.

5.2 Component Description

5.2.1 Lab Beacon

The beacon consists of three IR LEDs mounted on a 60cm tower. As seen in Figure 5.1(a), the IR LEDs are put in a series configuration and are supplied with an input of 5V and 38 KHz PWM signal. The choice of using a 38KHz PWM signal as opposed to a continuous digital signal was since 38KHz is the industry standard for IR emitters in order to differentiate IR light from surrounding sun IR light. As for the resistance, the design choice was to rely on the internal resistance of the Arduino Uno as the input resistance of the circuit to drive good current to the circuit. The design choice of putting them in series was for the main purpose that the Arduino had enough voltage to provide good voltage drop through the components, and this allowed to drive good current through the LEDs resulting in maximum power usage. Maximum power was essential to achieve higher intensity in order for IR receiver to pick up the signal over long distances.

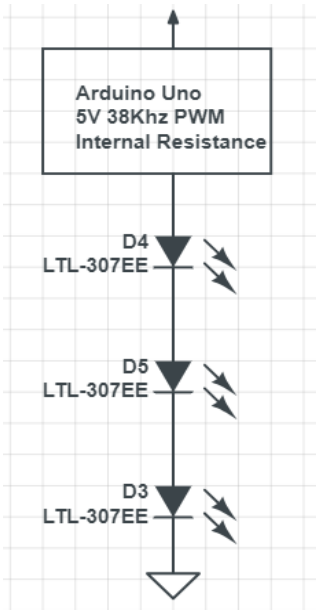
5.2.2 IR reciever

The IR receiver utilized in this submodule is an infrared phototransistor HX-1838. The following transistor has an IR filter, which makes it only receive signals of 38KHz. As seen in Figure 5.1(b), it is a standard transistor circuit supplied with 5V and outputs a digital signal according to the IR signal received.

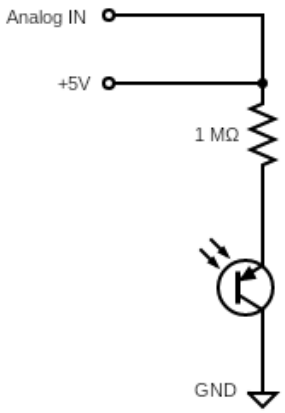
5.3 Testing

The first test for testing the functionality of the submodule was testing if it is able to emit infrared light. This was tested by pointing a phone camera into the diode and seeing if the diode is pink. The second test was done using two Arduino and breadboards with one supplying a 5V PWM signal to the IR emitter and the other Arduino to check if the IR 38Khz signal is received by the IR receiver, and outputting a boolean value. The following was tested with simulated situations. When the lab beacon tower and IR emitter tower were built, this was tested by letting the robot turn and seeing if the robot stops turning and drives forward when it was pointing in the direction of the lab.

The final test was to integrate the system with the infrared detection sub-module for the detection of the color of the laboratory walls. This testing will require setting different thresholds for different values of infrared radiation values received from different colors, and observing if it follows the algorithmic flowchart made. This was done by simulating situations and observing if it turns in the correct direction. This required mostly software testing and creating hardware placement adjustments on the robot.



(a) Circuit diagram of the Beacon.



(b) Circuit diagram of the robot-mounted towers.

Figure 5.1: Circuit diagrams of the laboratory setup.

6 Movement and Rock Sampling

6.1 Component Description

The wheel servos are responsible for driving around the map and the gripper servo is used to pick up the rock samples when found. The wheel servos are equipped with digital encoders, that are used to control the movement of the wheels. The wheel servos can be calibrated, so that the wheels are idle at a value of 90 degrees, which is the midpoint between 0 and 180 degrees.
The servo of the gripper only has two states: open and closed. The gripper will be activated as soon as the rock sample is sensed by the infrared sensor and deactivated when the robot returns to the laboratory.
The input of the software function for the straight line driving is distance in centimeters. For the turning function, the input is degrees.

6.2 Testing

The testing of the wheel servos and the gripper was straightforward and only required little research beforehand. The test that was performed on the wheel servos and encoders was to test whether the servos are correctly calibrated and turn accordingly. The obtained results showed that the servos were poorly calibrated and had to be recalibrated to the midpoint of 90. Once this was done, the wheels behaved as they were supposed to be.
The test on the gripper was performed following the wheel servo test. Software was written to close and raise the gripper repeatedly. This did not work initially, as the metal cable pulling on the gripper was connected to the wrong arm on the servo. Once this was fixed, the gripper behaved accordingly.

7 Communication

7.1 Component Description

The communication system consists of two Zigbee modules that are each mounted to an Arduino Uno board on the robots. These modules allow for wireless communication between both robots.
Both robots are required to be initialized so that they communicate on the same corresponding channel and with the correct ID. The initialization can either be done through XCTU or initialized in the Arduino script. Once the modules are initialized, data can be written to the serial. The receiving robot will check if there is any data transmitted/available. If so, it will store it in an array and act based on the data in this array.
In the overall strategy that was defined, communication would be omitted initially and only added if it could improve the overall system. Communication could overcomplicate the integration of all sub-modules and was not necessarily needed. Because of this, the Zigbee modules would only transmit Boolean values or individual characters to keep the communication low-level, just to show the working of the sub-module.

7.2 Testing

In order to get the communication working, some simple tests were performed. Firstly, both Zigbee modules had to be initialized in order for both of them to communicate. This was done by setting them up in XCTU. Once initialized, some simple commands were sent, like blinking the LED on pin 13 and receiving the correct characters that were sent. This was working relatively quickly, which allowed for more extensive testing.
Next up, the communication was tested by turning the servo of the ultrasound servo on the robots. This was tested by sending a Boolean value to the other robot, which in its turn would read this value and act accordingly by turning the ultrasound sensor.
The communication was completed within a week and ready to be implemented. Only during the integration of all sub-modules, the decision was made to not use the communication in the overall final system.

Figure 7.1: XBee system

8 Camera Detection (Extra)

In an attempt to expand upon the scope of the project, another robot design was proposed. This alternative robot would use higher amounts of processing power to process a live camera feed.

8.1 Component Description

This alternative robot was equipped with a webcam and a Raspberry Pi 4B, using real-time computer vision library 'OpenCV' to process the image into data that could be transmitted to the Arduino already on the robot. With this additional data, the infrared sensors would be obsolete, but the ultrasound sensor would still be required to detect the mountains.

8.2 Obstacle detection

The image from the camera was transformed to the HSL color space, an alternative to the RGB color space image that was obtained from the camera. Now, each pixel was defined by a position and an array containing values for hue, saturation, and lightness. This lightness value could be compared with a certain threshold value to detect dark-colored objects, such as the boundaries and craters, or light-colored objects, such as the rock samples. Finally, a small section of the image close to the robot was cropped out, one on each side, and the average lightness values for all these pixels were computed. If this average was above a certain threshold, the robot would know there was a boundary or crater, and direct the Arduino to rotate, depending on the side on which danger was detected, away from the danger.

8.3 Lab detection

For detection of the location and orientation of the lab, the OpenCV library was utilized to scan printed ArUco codes which were attached to the sides of the laboratory. These codes were easy to detect, their position on the x-axis was used to determine where the robot had to move, and the codes had information encoded in them to tell the robot which side of the lab it was on. To finalize robot movement into the laboratory, one idea was to include a compass module. By logging the absolute rotation at the start of the movement, the rotation of the laboratory can be saved. Combining this information with the data from the ArUco codes would give enough information to plan a path into the laboratory.

Figure 8.1: Examples of different ArUco codes

8.4 Testing

The webcam was mounted to one of the robots, and connected to the Raspberry Pi. Code was run on the Raspberry Pi over an SSH connection, enabling remote troubleshooting. The image captured by the Pi could be seen through X-11 Forwarding.

The code was able to detect blocks and boundaries, as can be seen in the following figures.

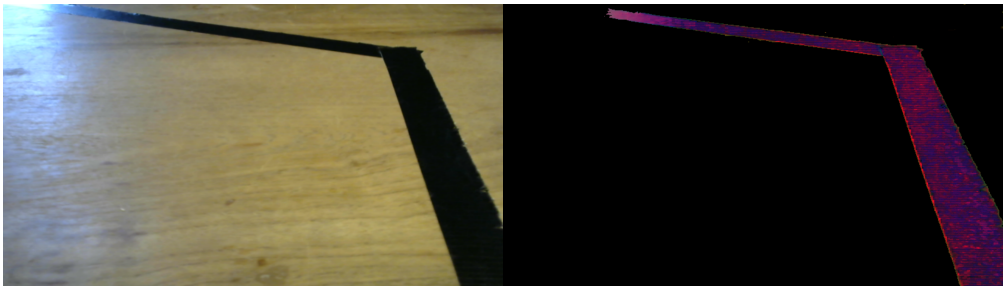


Figure 8.2: Camera feed and boundary threshold

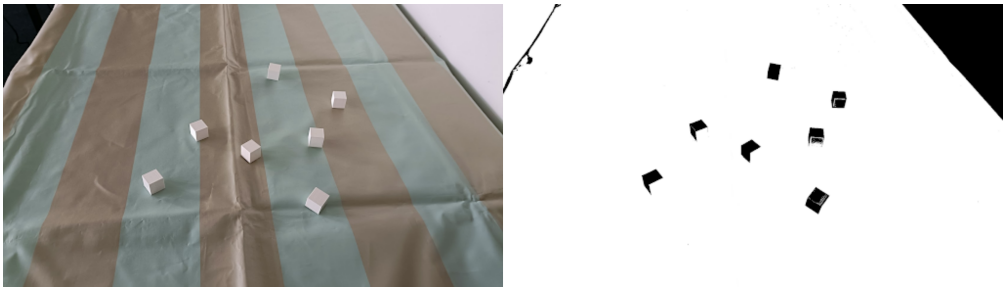


Figure 8.3: Camera feed and boundary threshold

9 Final System and Strategy

9.1 Description of algorithmic Strategy

A function-based approach for algorithm integration was chosen. The algorithm was divided into two main types of functions, which are actuator functions for movement and sensor functions for obstacle detection and navigation. The choice of using this approach as opposed to using the standard object-oriented approach allowed for optimal hardware data use, as there were not many variables but in a lot of ways those variables were utilized. This provided the software with more structure and easier logic implementation.

Two different flowcharts were made: one flowchart would control the initial strategy and the second would incorporate the camera strategy. Figure 9.1 shows a simplification of the final algorithm. Each subsystem has been given a different color, which is zoomed in in figure 9.2.

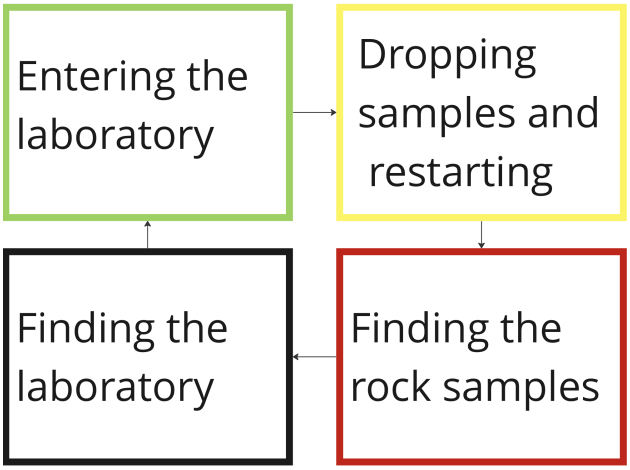


Figure 9.1: Flowchart of the basic algorithm.

As seen in Figure 9.1, shows how all subsystems work and how they are integrated into the final strategy.

The algorithm starts with figure 9.2a, where the robot starts searching for a rock sample. It will stay in this subsystem until it has found a sample while avoiding all obstacles. Once found, it will move to the next subsystem.

The second subsystem, depicted in figure 9.2b is responsible to get the robot with the sample back to the laboratory. Here it would start by trying to find the beacon that is attached to the lab and avoid the obstacles to get back to the laboratory. In the end, the robot will check for the laboratory wall and from there enter the next part of the algorithm.

Figure 9.2c is the most complex part of the algorithm. The robot will first scan on which side of the lab it is located. This would be determined based on the color of the side of the wall. From there, it would rotate and drive parallel to the lab until the entrance of the lab is found.

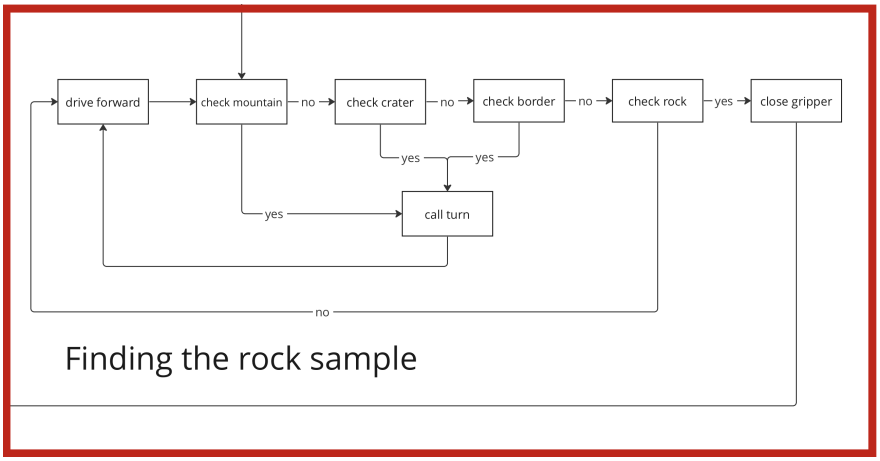
Once the robot is straight aligned with the entrance ramp of the lab, it will enter the final part of the algorithm where it would drop the sample in the laboratory. This part is shown in figure 9.2d. The ramp at the entrance of the wall is marked on the sides by black tape to avoid it from driving off. Near the end of the ramp, there is a black line that marks the stopping point of the robot, where it has to drop the rock sample. Once this line is detected by the front infrared sensors, the gripper will open and the rock sample is dropped inside the lab.

9.2 Testing of Final System

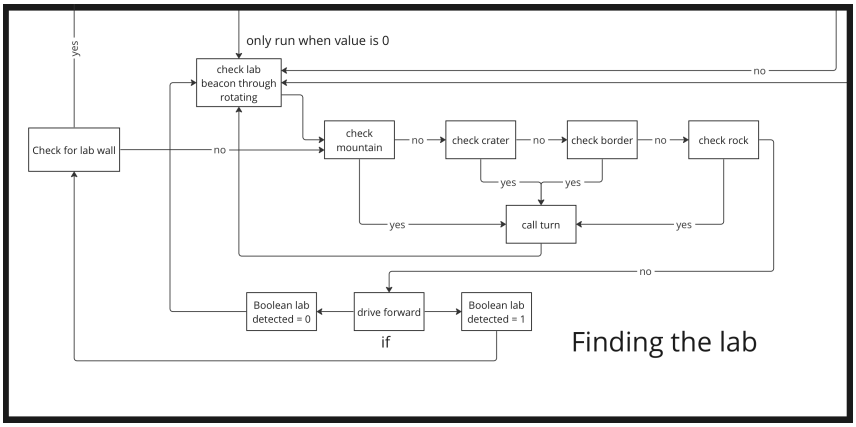
During the process of the final integration, the team made use of an incremental approach to be able to properly validate the logic behind the algorithm and to observe how well the robot handles its commands. The testing initially started by putting a loop for the robot to move and check for obstacles, such as craters, borders, and mountains. The expected outcome was that the robot will not crash into an obstacle and navigate for the block properly. There were two main problems that arose from testing the movement and sensors:
The first problem was that the code went really fast not providing time for the robot to adapt sensor information by performing actuator tasks. This issue was solved by using cardboard extended at the front of the robot. This allowed us to make use of more IR sensors on the robot to prevent fault situations, and since the sensors were further from the wheel, the robot had more time to do its turns. Delays were also implemented in the code to ensure the performance and reliability of the robot.

The second problem was the initial code only turned in one direction when an obstacle was detected. This was improved by making more case-based functions, where the robot would turn left when the right sensor detects and turns right when the left sensor detects. As a result, this improved the performance of the robot as it wouldn't get stuck as much in corners and congested areas.

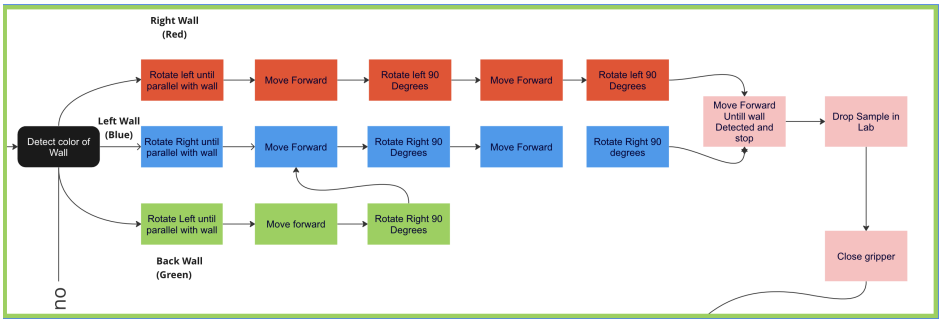
As for when the block is collected, a sequence of steps occurs to get the robot back to the lab as previously explained. The testing for that in the final system was to run code for the robot to turn and check for IR light. The expected outcome was for the robot to drive toward the lab when pointing in its direction. Then, the algorithmic flowchart for when it reaches the lab wall was tested. However, the logic was there but the implementation was very difficult when tests were done. This was mostly a hardware issue as the robot didn't have accurate movements making it hard to follow the flowchart's path.



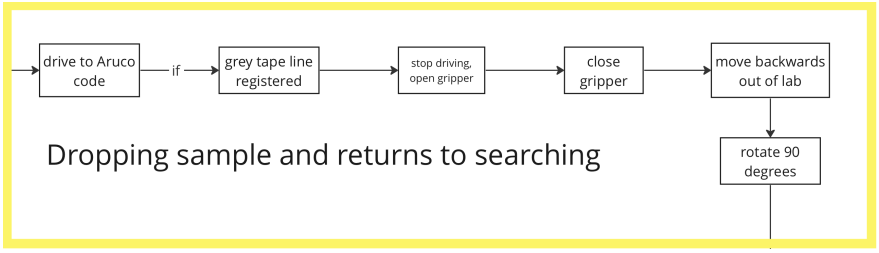
(a) Flowchart of the algorithm to find the rock samples.



(b) Flowchart of the algorithm to find the laboratory.



(c) Flowchart of the algorithm to enter the laboratory.



(d) Flowchart of the algorithm to drop the rock sample.

Figure 9.2: All subsystems of the total final algorithm.

10 Summary

10.1 Conclusion

In conclusion, most sub-modules fulfilled their tasks when engaged separately, but they did not work as expected when tested collectively. In the start, the team faced some difficulties when having to integrate everything together, but after some changes, a lot of things were fixed. Although, the team did not have time to complete the whole procedure of returning all the rock samples properly back to the lab. Firstly, the ultrasound sensor worked properly when integrated into the system and enabled the robot to detect the mountains during its path. Although, sometimes during the final tests, the robot failed to detect the obstacles when they were located at certain critical angles or when they were located too close to a crater. The infrared sensors succeeded to detect the boundaries, the craters, and the rock samples, according to the difference they were detected from the reflected light. During the final testing, the robot had some problems in situations where the craters were very close to each other or too close to a border such that it would get stuck. Moreover, sometimes a sample would not be detected by the robot if positioned in a small gap without a sensor. This happened because we did not have enough time to think of a better solution for positioning the sensors. The movement functions worked properly, when called during the entire process, enabling the robot to drive, stop and turn. The rock sampling worked only in certain situations because the robot struggled with aligning the gripper with the rock sample. By moving the robot in and out of the locker multiple times, the position of the rock-detecting sensors was slightly changed, making the alignment and the tuning very difficult. As far as the laboratory part is concerned, it worked properly when tested independently and it did not have a lot of troubles during integration. Finally, the communication module was not implemented in the final system since the team ended up using just one robot.

10.2 Reflection

The robot managed to find the direction of the lab and reach its proximity. So successfully, the robot managed to safely cruise Venus and collect rock samples but unfortunately, it did not complete the route of going back to the lab and depositing the sample. In order to complete the task and make the robot find the laboratory, further testing, and debugging are needed. In order to increase the accuracy of the system it is necessary to better tune the sensors and to optimize their thresholds. The algorithm for going back to the lab could have been further improved by considering design alternatives, which allowed for more accurate movements. One such example would have been to use decoders, along with a virtual map, programmed into the main function. This might not be as optimal as using an IR beacon in terms of performance, but would be easier and more reliable to implement in this project.