CS-279 Intelligent Robotics Report Car

Park Navigation

Team Name: Group 3

Date: 07/03/2024 Team

Members:

1. Carl
2. Kieran
3. Pele
4. Bushra
5. Omar

CS-279 Intelligent Robotics

Report

Synopsis

The report seeks to holistically provide an insight into the design and implementation approach that our team used when programming our task with the mBot2. We were tasked with designing a parking lot and implementing an autonomous parking system. Navigation, parking management, and enhanced car functionalities within the simulated environment all were essential features when solving our task.

Our approach is structured into two main design components, hardware, and software. The integrated sensors of the mBot2 allowed us to effectively simulate a car in a parking environment. There were four stages to our approach: line following, parking, standard car functionality, and the integration of different robots for different functions.

From the software side, we thoroughly explored and tested all sensors, developing the programming logic to execute our task efficiently, including line-following algorithms, exit identification algorithms, parking space assessment algorithms, timed parking management, implementation of other car functionality algorithms such as alarms, lights, sensors, and more.

Some challenges we faced included sensor calibration and algorithmic optimisation, as well as the integration of advanced functionalities and how to utilise them. Thus, we carefully

constructed a testing and validation framework that ensured programmed behaviours are indeed reliable and accurate.

The insights from this report regarding the complexities involved in designing and implementing a robotic system (interfacing both hardware and software for seamless operations toward the desired functionality) could be used as a stepping-stone to aid in future iterations of autonomous vehicles.

Task Statement

Vehicle autonomy is a new and exciting field of research in the car industry, but presents its own set of issues, specifically parking in crowded areas. Our automated parking robot aims to solve this by creating an accurate car park simulation with its own set of needs and restrictions made evident in the solution:

Needs:

- Ultrasonic Sensor for Object Detection
- Quad-RGB sensor for line/colour detection
- Light & Motion Sensor
- Speakers/Microphone
- Robust Code Algorithms

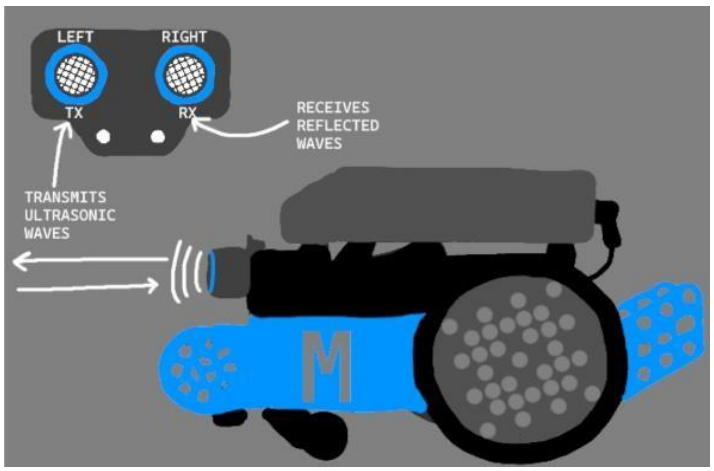
Restrictions:

- Inaccurate Turning
- Poor Sensing
- DIY Restrictions.
- Sound Recording.
- Communication With Other Bots

Hardware Usage

Our task utilises the full suite of mBot2 sensors to explore the robot's capabilities and simulate real-world automotive functionalities and environmental interactions, demonstrating the robot's ability to adapt to dynamic scenarios.

Ultrasonic Sensor



The ultrasonic sensor measures distance by transmitting ultrasonic waves from the left probe and receiving them on the right. The reflected waves are measured to determine distance, which helps in automotive tasks, such as checking space availability and detecting obstructions or pedestrians for safety and efficiency.

Figure 1: Ultrasonic Sensor Demonstration.

Quad-RGB Sensor

The Quad-RGB sensor detects and measures all the colour characteristics of a surface. The sensor works by projecting light onto the object; the reflected light from the object is then sent back at various wavelengths of light corresponding to different colours to detect the colour on the surface with reasonable precision. Precise sensing was initially challenging but improved with testing and calibration techniques. The sensor, equipped with four probes, enhances line following and space detection for the autonomous parking system. The outer pair carries out the fine-tuning while the inner pair does larger adjustments to enhance the accuracy, which was at the forefront of our design for our autonomous parking system for line following and space detection.

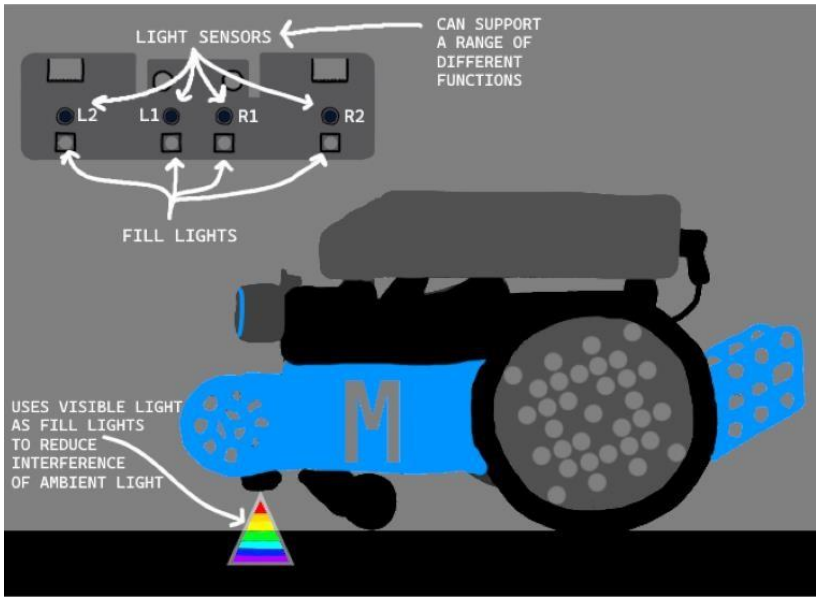
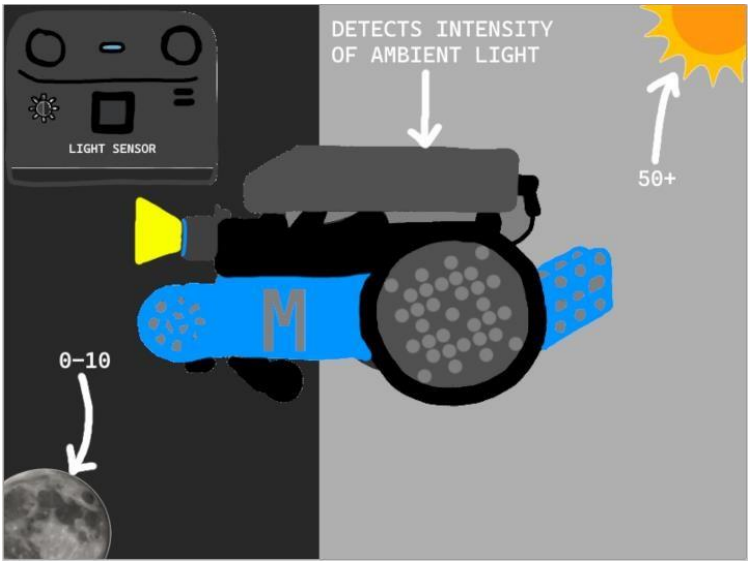


Figure 2: Quad-RGB Sensor Demonstration.

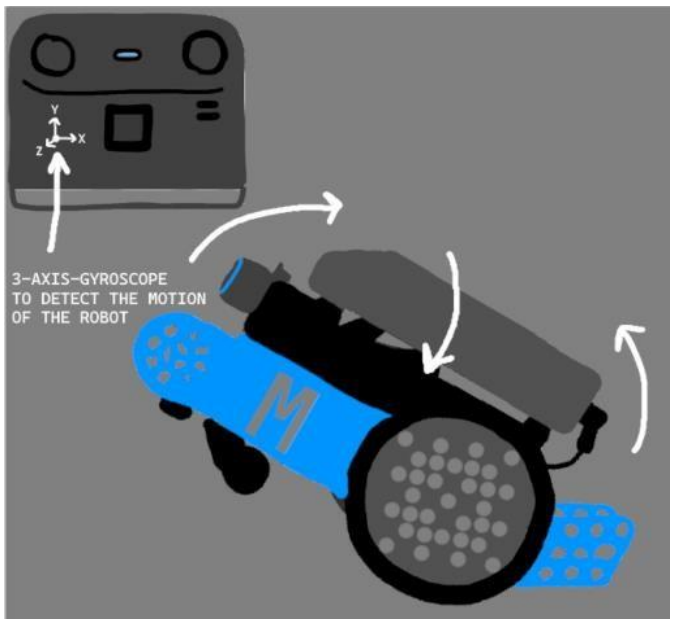
Light Sensor



For the light sensor to detect ambient light, it uses a photoresistor as its core component. This has the ability to change the resistance according to the surrounding light intensity, which can then be detected by the microcontroller, which therefore determines the light intensity in the surrounding environment (Makeblock Education, 2022). We used this to simulate headlights in a dark setting.

Figure 3: Light Sensor Demonstration.

Motion Sensor



The gyroscope in the motion sensor is used to detect motion. It uses a rotor at the axis to resist the change of direction when the gyroscope begins to rotate. We used this to detect a potential car hijacking by detecting excessive motion.

Wi-Fi and LAN Connectivity:

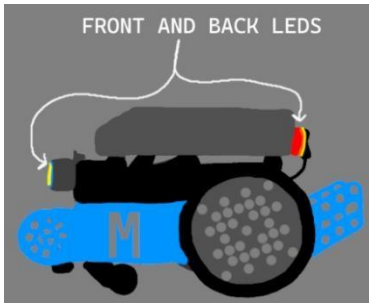
The robot has Wi-Fi capabilities that allow it to connect to the internet using radio waves to transmit data. It can use this to communicate with other mBots on the internet via LAN. This was used to create a payment authentication system for priority parking.

Figure 4: Motion Sensor Demonstration.

Speaker & Microphone

The robot comes equipped with a speaker to project sound by converting gathered electrical energy into mechanical energy, either pre-recorded or used-recorded, via the built-in microphone, capturing sound and converts it into an electrical current to be interpreted as a string, triggering a response from the robot. We applied these features for emergency and car alarms.

LEDs & Charging Port



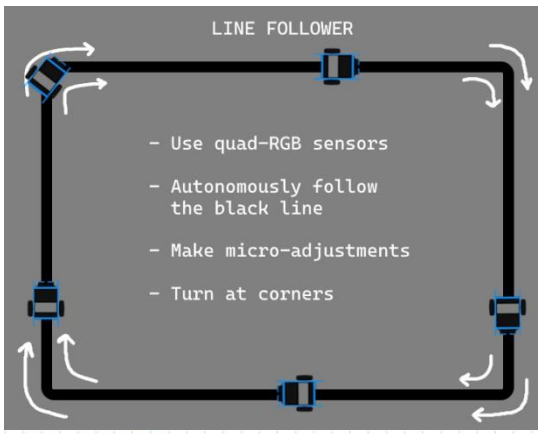
The robot features 5 rear LEDs that could be individually controlled, and 2 front LEDs for the ultrasonic sensor, allowing us to implement various lights typical of a standard car. It also comes with a charging port to charge the robot used to simulate recharging in electric-only parking spots.

Figure 5: LED Demonstration.

Software Analysis and Design

The software side utilises different sensors working together in parallel to seamlessly simulate an autonomous parking system designed to behave as close to a real car as possible.

Line Following



We used t he quad-RGB sensors to create a line - following algorithm to traverse the car park by detecting the black road with all four probes on the sensor, allowing the car to adjust if one of the probes stops detecting black.

Figure 6: Line Follower Algorithm Overview.

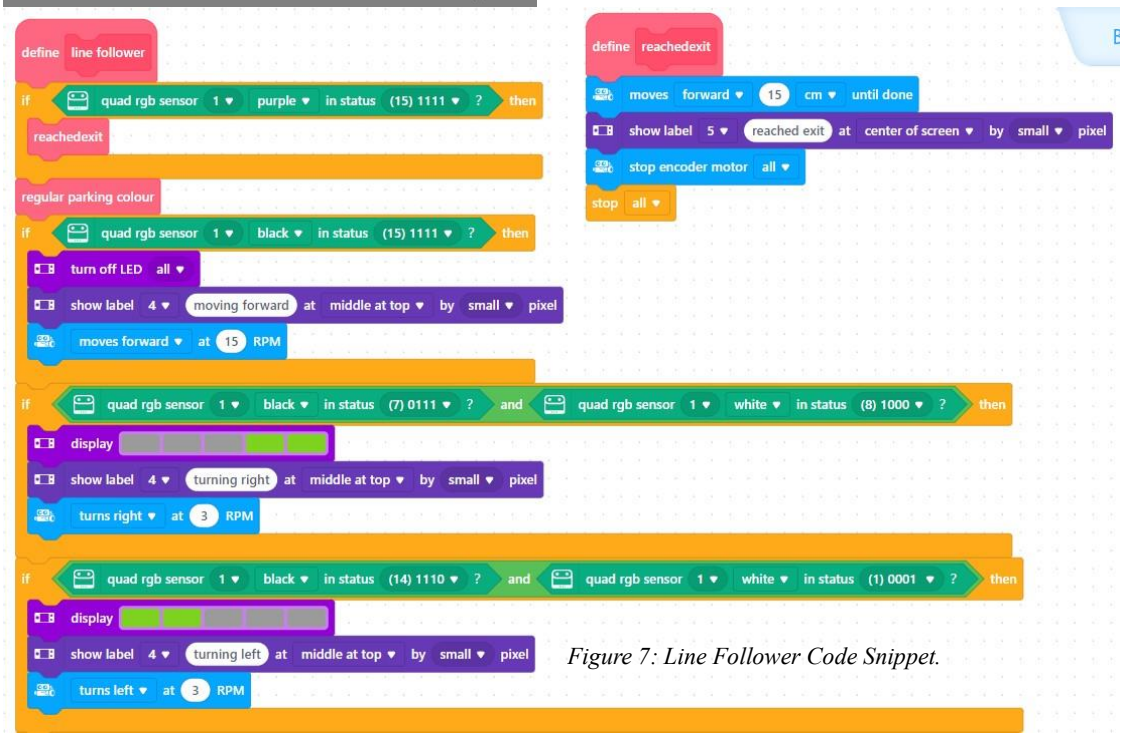


Figure 7: Line Follower Code Snippet.

Figure 8: Line Follower Code Snippet.

Figure 9: Line Follower Code Snippet.

This approach allowed us to make micro-adjustments where necessary, using the outer sensors for sharper turns, and the inner sensors for gradual ones. We also integrated calibration fixes, based on our rigorous testing with labels to display the detected colour of each probe to the CyberPi.

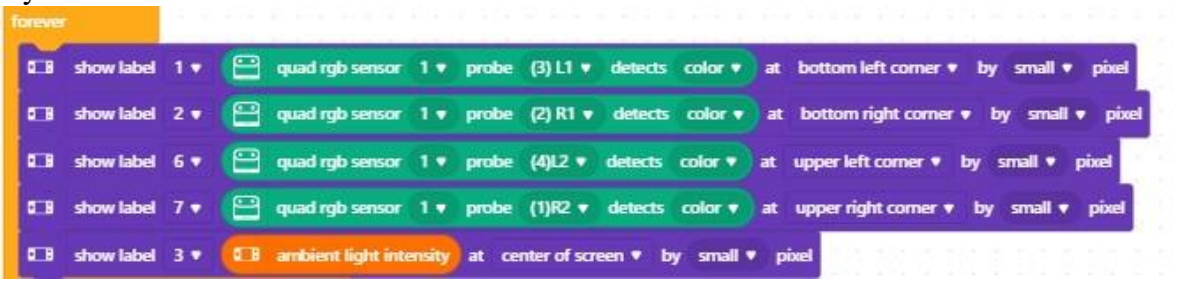


Figure 10: Testing Functions for Each Individual Probe on The RGB Sensor.

Parking

The parking algorithm utilises both the quad-RGB and ultrasonic sensors. When the probes detect a colour deviation, the bot will turn, using the ultrasonic sensor to check for a space. If there is an available spot, the bot will park, and if not, it will continue along the line.

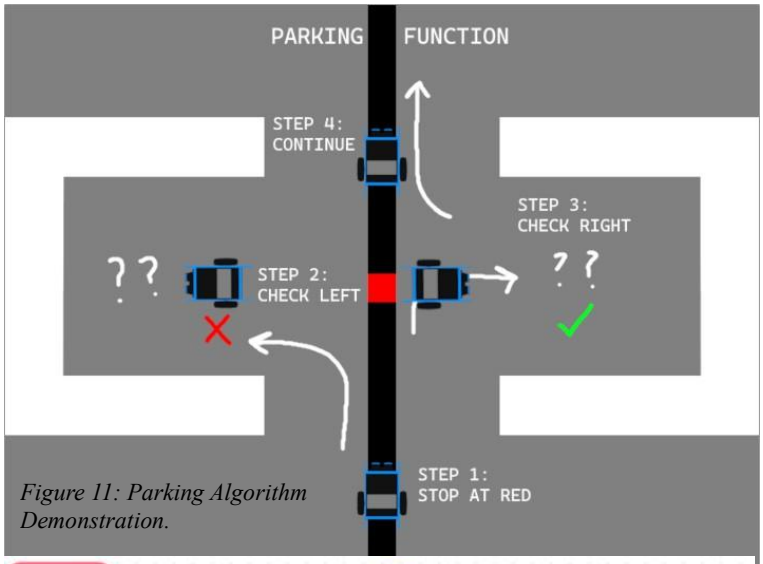


Figure 11: Parking Algorithm Demonstration.

Figure 12: Parking Code Snippet.

This chosen method was the most effective way for the robot to know if a space was occupied.

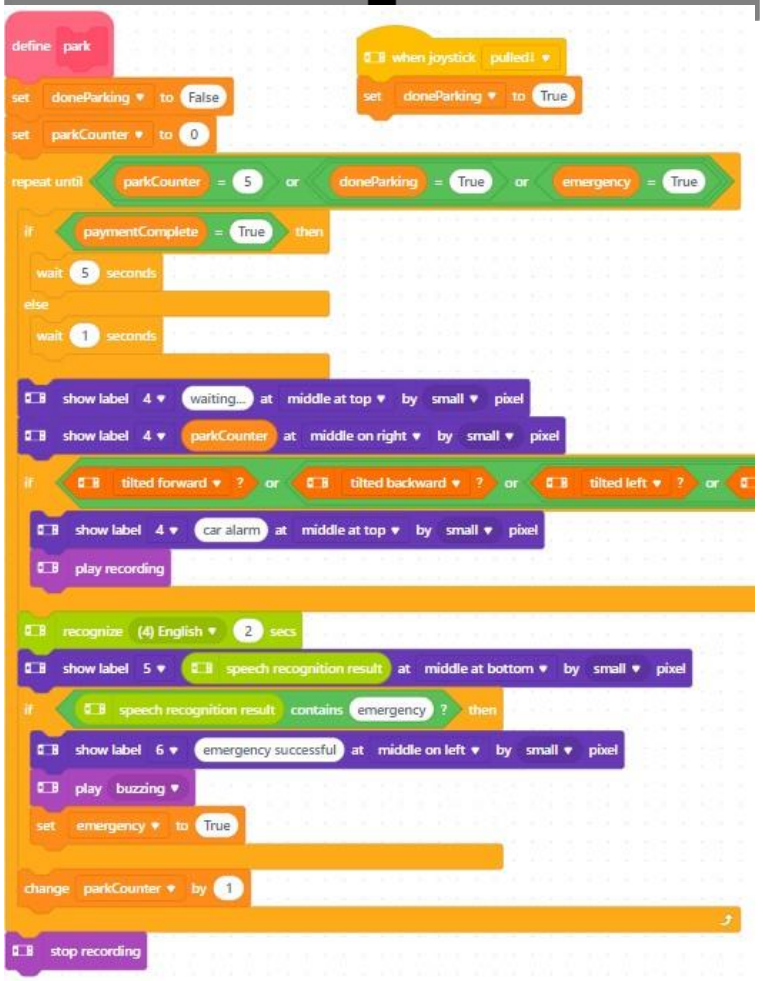


Figure 13: Parking Code Snippet.

Figure 14: Parking Code Snippet.

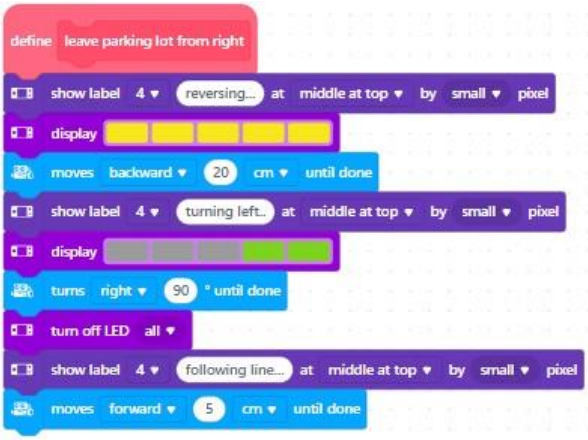


Figure 15: Parking Code Snippet.

Figure 16: Parking Code Snippet.

Additional Features

We implemented different algorithms to ensure the robot behaves similarly to a car; indicators activating the back LEDs, reversing and brake lights, headlights, a car horn for obstructions, and alarms for both the car and emergency situations.

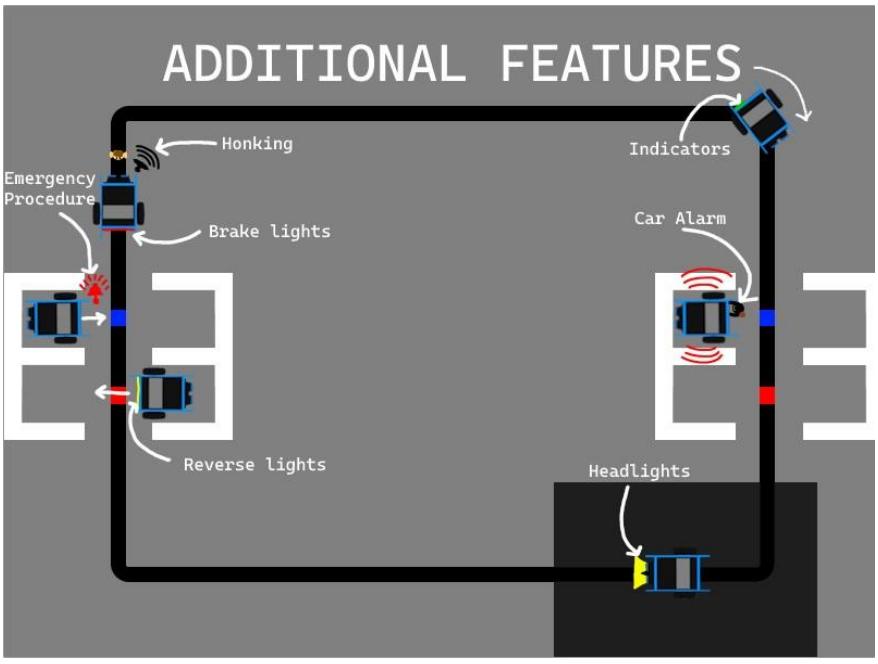


Figure 17: Additional Features Demonstration.

Indicators

Figure 18: Indicator Code.

Brake Lights

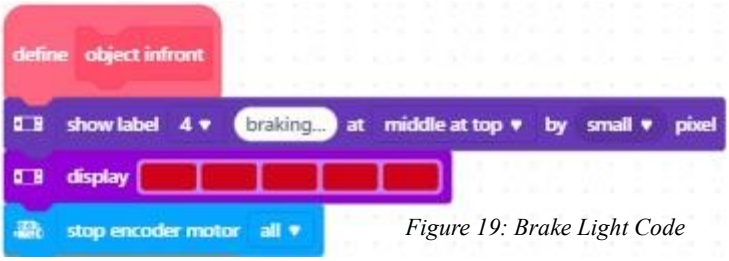


Figure 19: Brake Light Code

Reverse Lights

Figure 20: Reverse Light Code

Headlights

Figure 21: Headlight Code.

The use of lights was straightforward, but effective. The front and back LEDs activated in various colours to enhance the bot's realism and limiting the indicator lights to just 2 of the 5 rear LED made it clear which direction the bot was turning.

Car Horn

We used our own counter variable to play a horn every second after 5 seconds. This was an alternative to making the robot wait 5 seconds as it could not perform other operations within that time, highlighting another limitation of the robot.

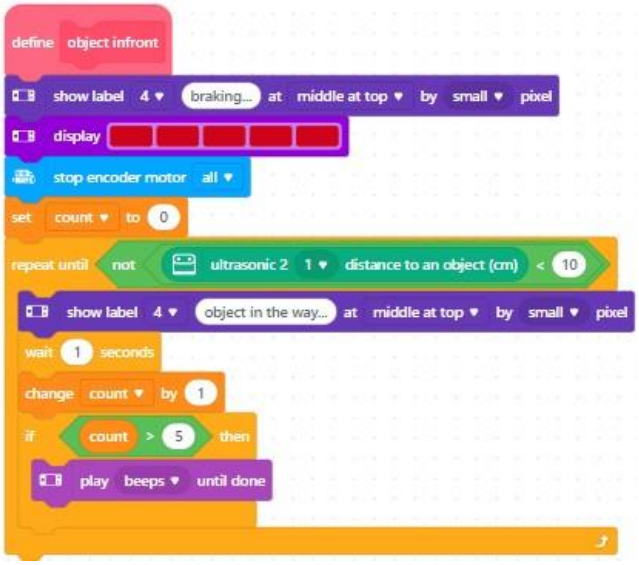


Figure 22: Car Horn Code.

Car Emergency

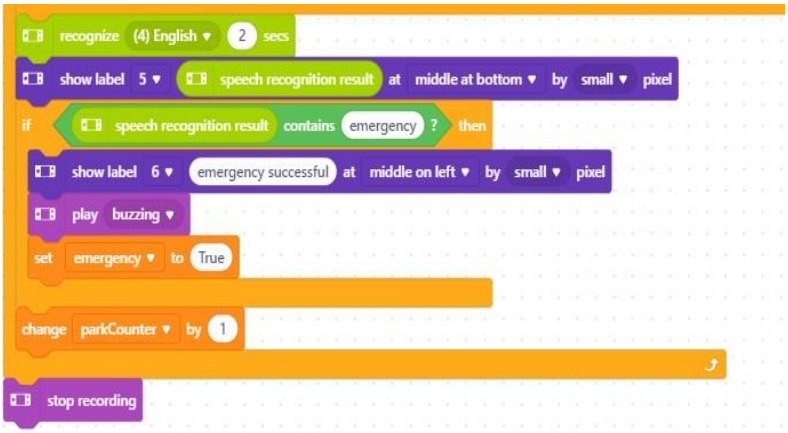


Figure 23: Car Emergency Code.

The robot uses the microphone to record audio from the surrounding environment, which is translated and stored as a string. We detected phrases with the keyword "emergency" to make the robot leave the lot faster than usual, utilising similar line following procedures. However, this highlighted another limitation - it could not perform other actions whilst recording, similar to the wait function.

Car Alarm

Figure 24: Car Alarm Code Snippet.

We used the microphone, speaker, and motion features for this implementation. The bot will record an alarm sound and store it, which can later be used to detect excessive motion in any direction during parking.

Figure 25: Car Alarm Code Snippet.

Different Parking Variations

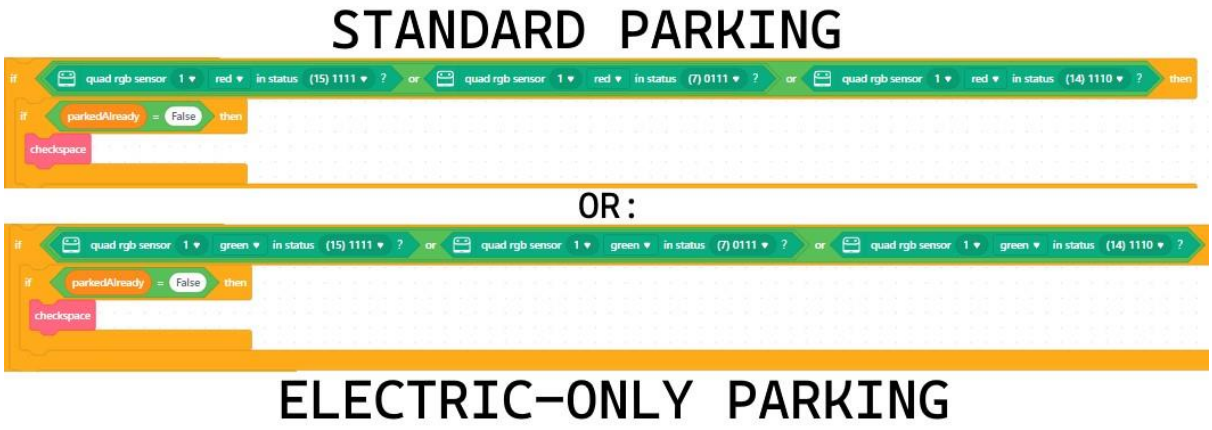


Figure 26: Standard VS. Electric-Only Parking Comparison.

Priority Parking

The bot will communicate over LAN for a payment request and will extend the parking-time if it is authenticated.

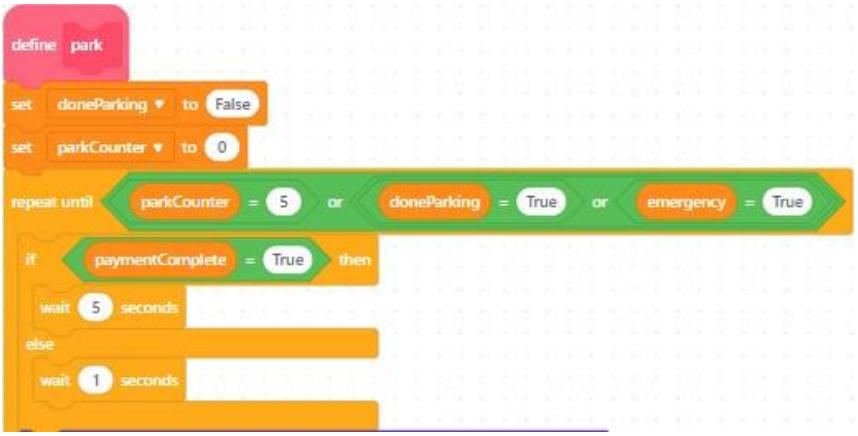


Figure 27: Priority Parking Code Snippet.

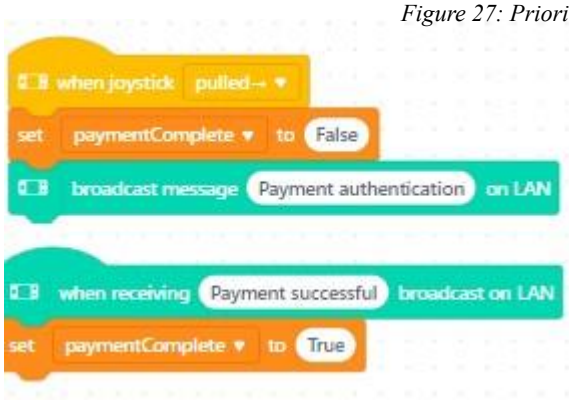


Figure 28: Priority Parking Request.

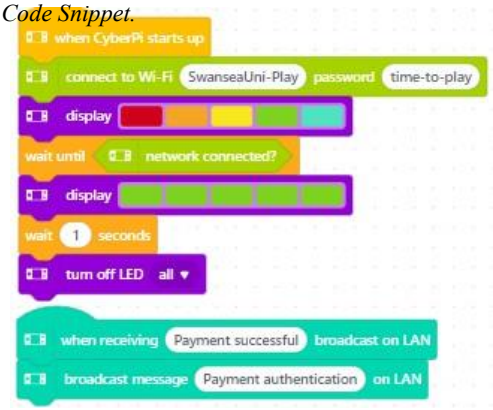


Figure 29: Priority Parking Authenticator.

Testing and Debugging

Testing and debugging were made simple using the provided display features that allowed us to monitor all sensory data, shown in the previous code snippets.

Integration and Testing

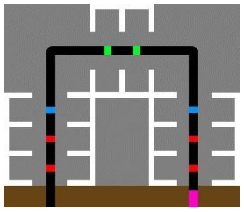
As previously mentioned, the system operates similarly to a real parking scenario, carrying out a standard parking procedure, with a variety of additional features due to the initial simplicity. We designed this because, while autonomous vehicles are prevalent in the car industry, they often lack solid parking capabilities.

Our integration was structured around a 4-stage plan to ensure successful project progression:

1. Line follower.
2. Parking.
3. Car functionality.
4. Different parking robots.

We chose this approach to maintain robot autonomy, allowing it to adapt to various parking lot layouts without hard-coded paths.

Map Layout



The map design leveraged the robot's features without relying on "hard coding" paths, allowing us to utilise the majority of the mBot2's designed functions.

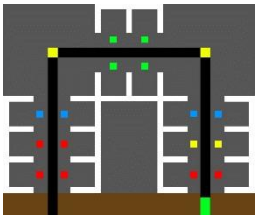


Figure 30: Our final map design.

Figure 31: Our Initial map design.

Line Follower

The line follower was a crucial element and carefully implemented, using quad-RGB sensors to stay on track. If the robot veered off course, it adjusted its path based on sensor responses.

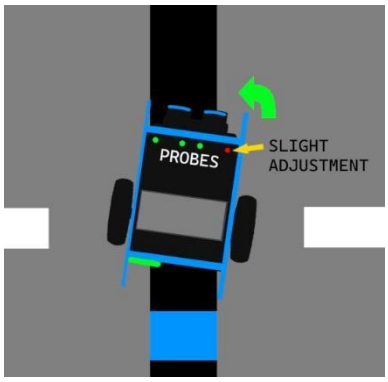


Figure 32: Line Follower Demonstration.

Parking

For parking, our method followed these steps that we had previously planned: • Stop and turn left on the colour-coded spot.

- Check for an obstruction; if detected, turn 180 degrees to the other side.
- If both spaces are full, continue along the road.
- If a space is empty, move forward and park.

We used ultrasonic and quad-RGB sensors to navigate and park to showcase the key capabilities of the mBot2, using specific coding blocks to line following, parking, and other features.

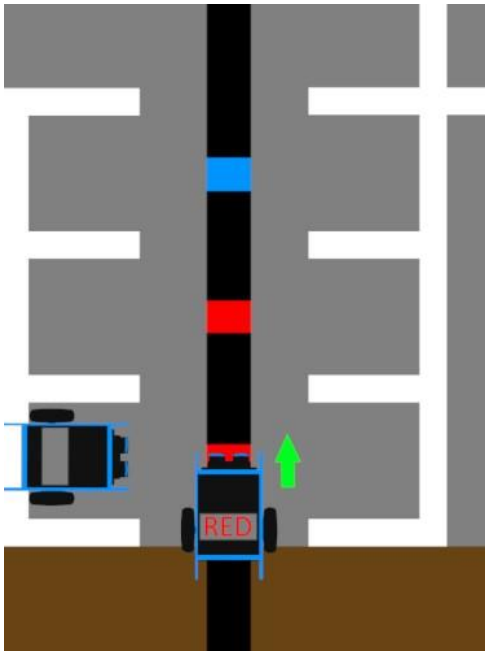


Figure 33: Traversing the parking lot.

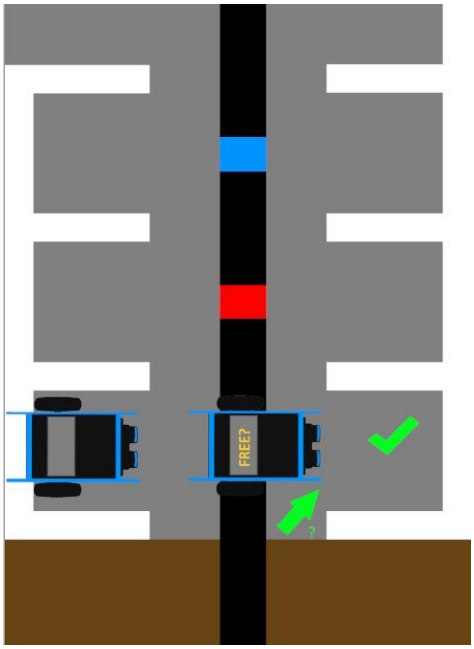


Figure 35: Entering available space.

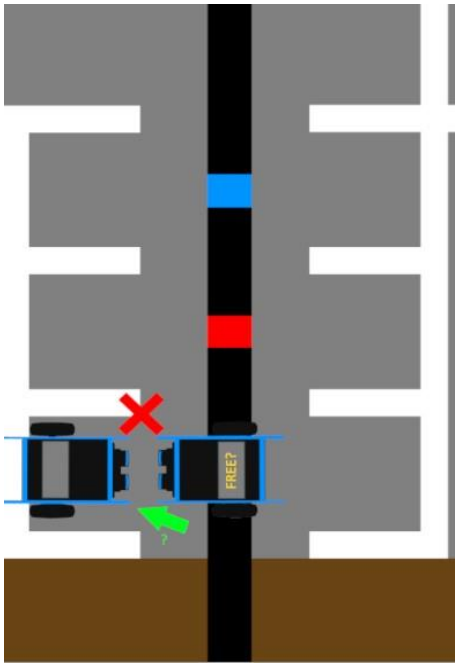


Figure 34: Checking space availability.

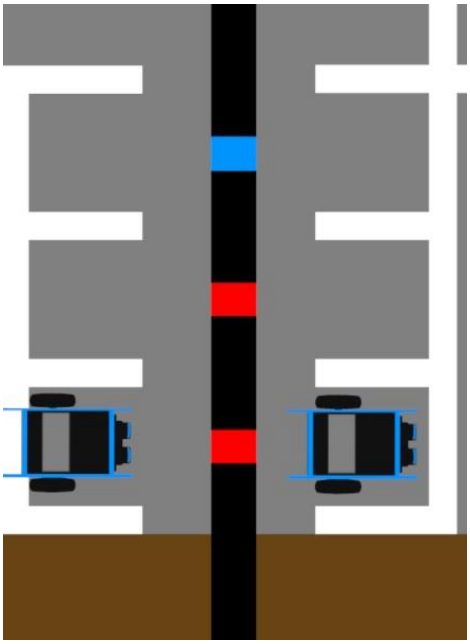
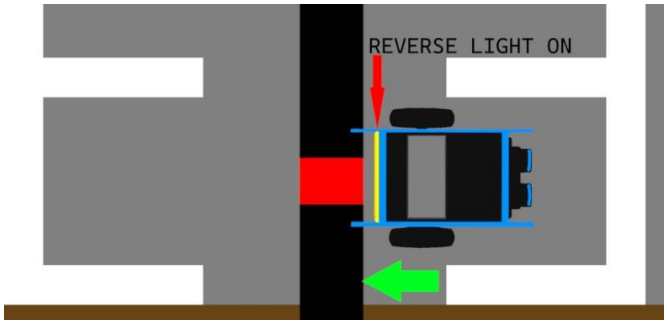


Figure 36 : Entering available space.

Car Functionality

We aimed to replicate regular car functions using various features.

Reverse Lights



The bot's back LEDs will turn yellow when reversing.

Figure 37: Reverse Light Demonstration.

Brake Lights

When the bot brakes for other cars, pedestrians or stop and exit gates, the LEDs will turn red.

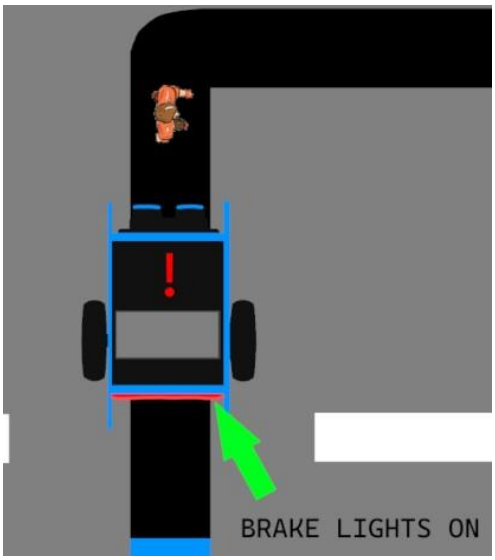
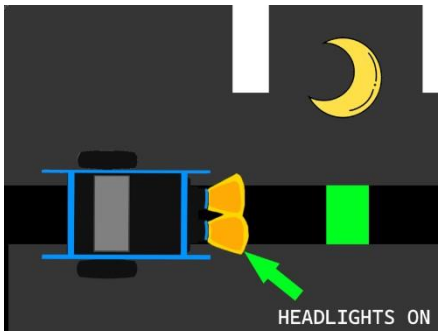


Figure 38: Brake Light Demonstration.

Headlights



The light sensor activates the front lights in low-light conditions.

Figure 39: Headlight Demonstration

Indicators

The bot will indicate to the side it is turning using the back LEDs no matter how small/large the turn is.

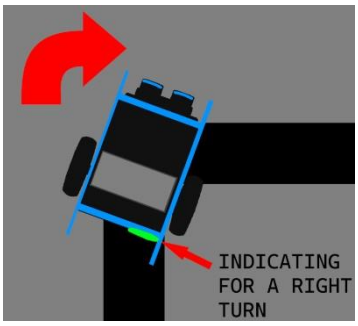
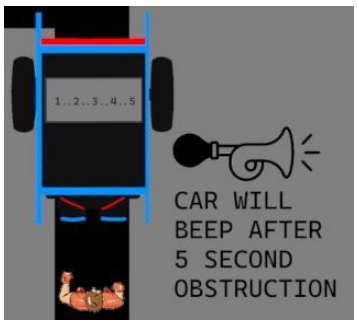


Figure 40: Indicator Demonstration.

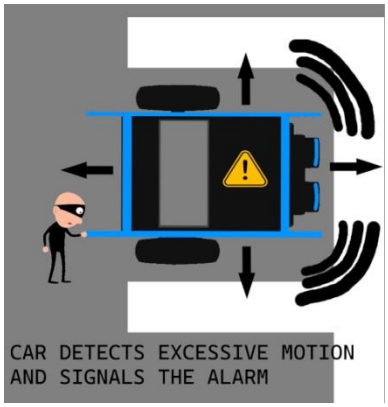
Object Detection & Car Horn



If an object is detected in front of the mBot's ultrasonic sensors for an extended time, the bot will beep until the obstruction clears.

Figure 41: Car Detection and Horn Demonstration.

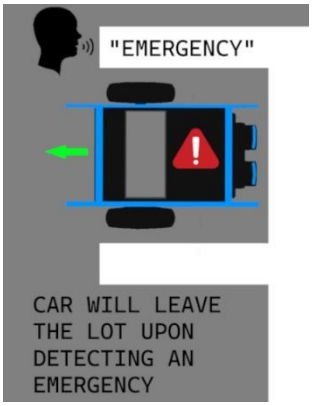
Car alarm



To simulate car theft, the motion sensor triggers a previously recorded audio alarm if excessive motion is detected while parked.

Figure 42: Car alarm Demonstration.

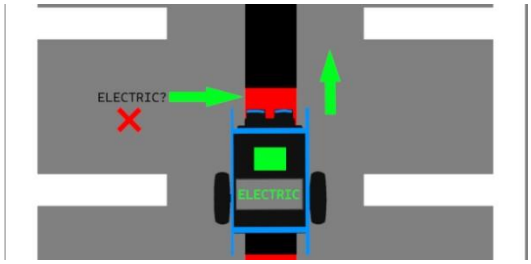
Emergency Alarm



The bot will initiate the emergency procedure when detecting the "emergency" keyword, reversing and exiting the parking lot using its microphone and speaker functionality.

Figure 43: Emergency Procedure Demonstration.

Parking variations



The robot's quad-RGB sensor allows parking in specific spots based on the chosen mode (standard, disabled or electric).

Figure 44: Parking Variation Demonstration.

Priority parking

Priority mode enables an extended parking duration if a payment request is authenticated over LAN.

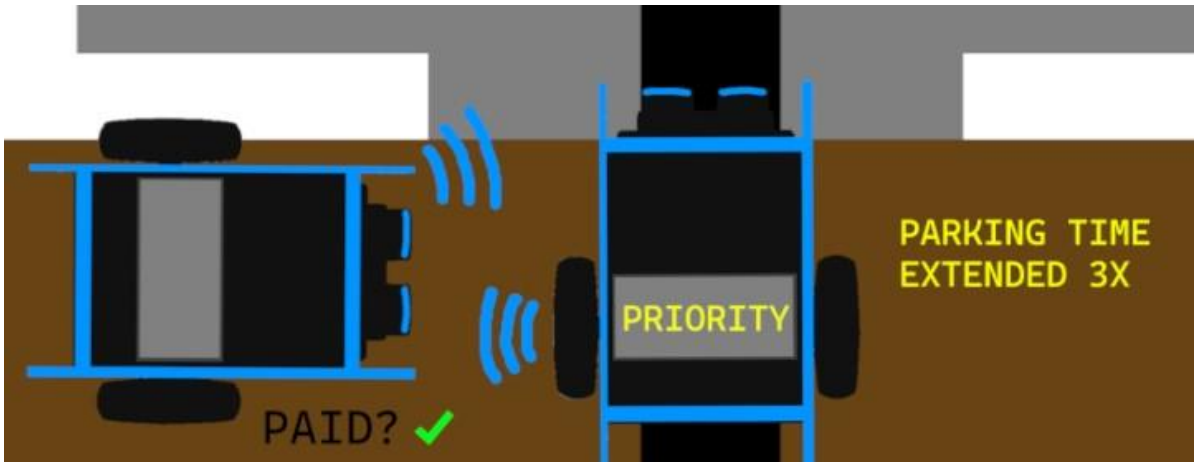
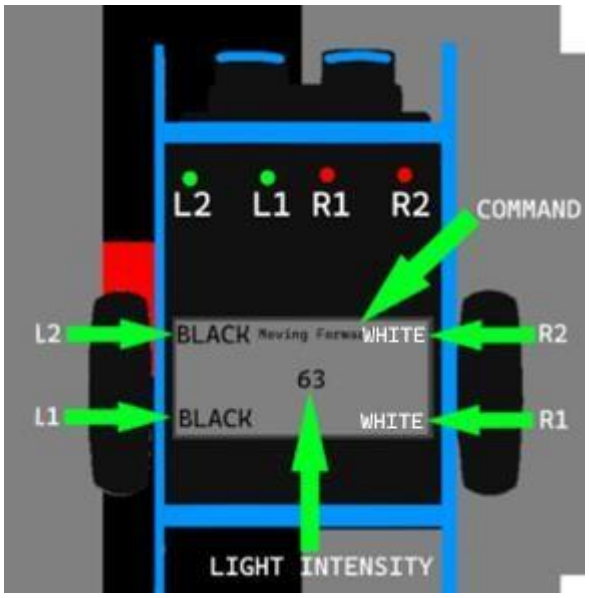


Figure 45: Priority Parking Demonstration.

Testing



Testing and debugging were essential to identify and fix code bugs. We had used this initially to notice that our quad-RGB sensor had poor calibration, and thus created a debugger to monitor sensor data and ensure accurate calibration.

Figure 46: Testing and Debugging Demonstration.

Conclusion

To conclude, we designed, tested, and implemented numerous features provided by the mBot2 to fulfil our task. Although some aspects of our project were challenging, it was rewarding to learn and construct a fully autonomous parking robot at every incremental step.

We utilised many hardware features that the bot had to offer, such as the ultrasonic sensor, the quad-RGB sensor, the LEDs, the speaker, the microphone, the motion sensor, and the light sensor, all having specific software integration to simulate a self-parking car.

Testing was a vital strategy to resolve any issues we faced along the way, using the display methodologies to consistently preview what our code was doing and how the robot was responding. Without this, we would have never detected certain issues, for example poor calibration with the RGB sensor.

There are many lessons to take away from this project. For most of the group, it was our first real introduction into robotics in general, and we found it fascinating to learn and test a lot of the features that are also prevalent in the real world. Teamwork and communication were also a valuable lesson, which in hindsight could be a future improvement.

In the future, as previously mentioned, we would like to design an autonomous parking lot, which would work conjunctively with all of our designed robots. The parking lot would use load/weight sensors to detect the presence of vehicles and would communicate to the robots which spots are free and which aren't. If there are no spots available, the robot will move to another parking lot.

References

Makeblock Education. (2022, April 29). *Light Sensor*. Makeblock Education. Retrieved May 3, 2024, from https://education.makeblock.com/help/