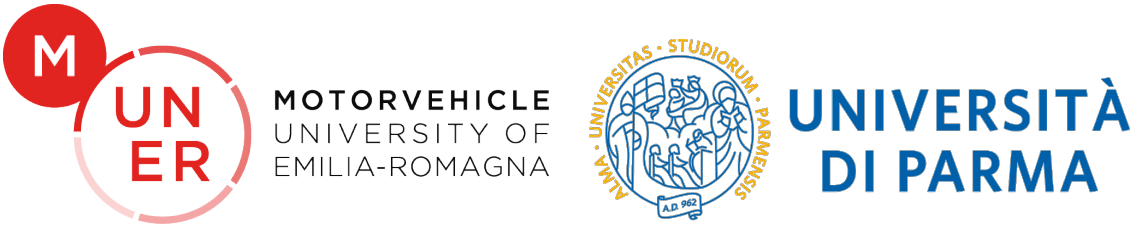


Electronics for Automotive System Robocar Report

Valerio Tiri and Federico Rovighi

October 26, 2024

Contents

List of Figures

Chapter 1

Introduction

1.1 Project purposes

The purpose of this project is to gain familiarity with the development of a Hardware-in-the-Loop (HiL) process, in which a Robocar is equipped with a suite of sensors that enable it to accomplish a variety of simple tasks. The car is outfitted with an *STMF411RE* microcontroller, which runs C code generated within Simulink. The development environment thus consists of Simulink, with sensors and actuators initially mapped using the *STM CUBE IDE* (Figure 1.1). The microcontroller's mapped pins are then used in Simulink to acquire sensor inputs and send outputs to the actuators, through the development of control strategies and feedback loops, enabling the car to perform the required tasks

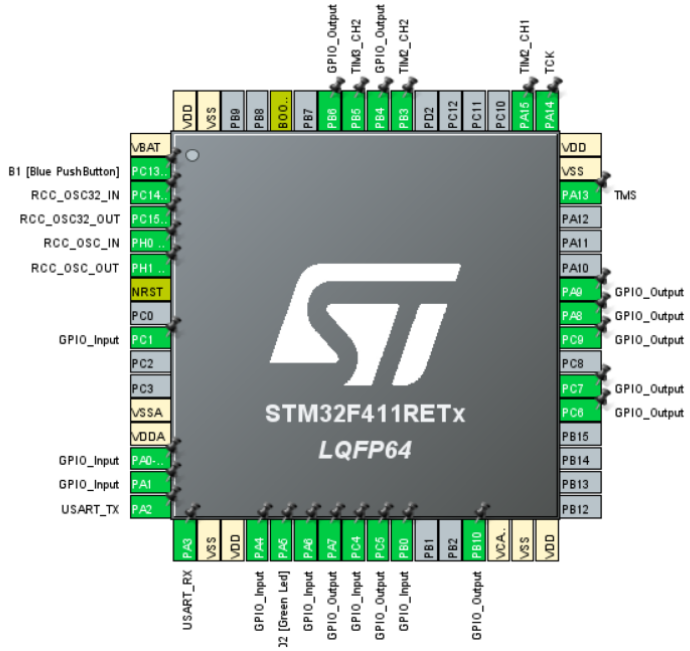


Figure 1.1: STM CUBE IDE.

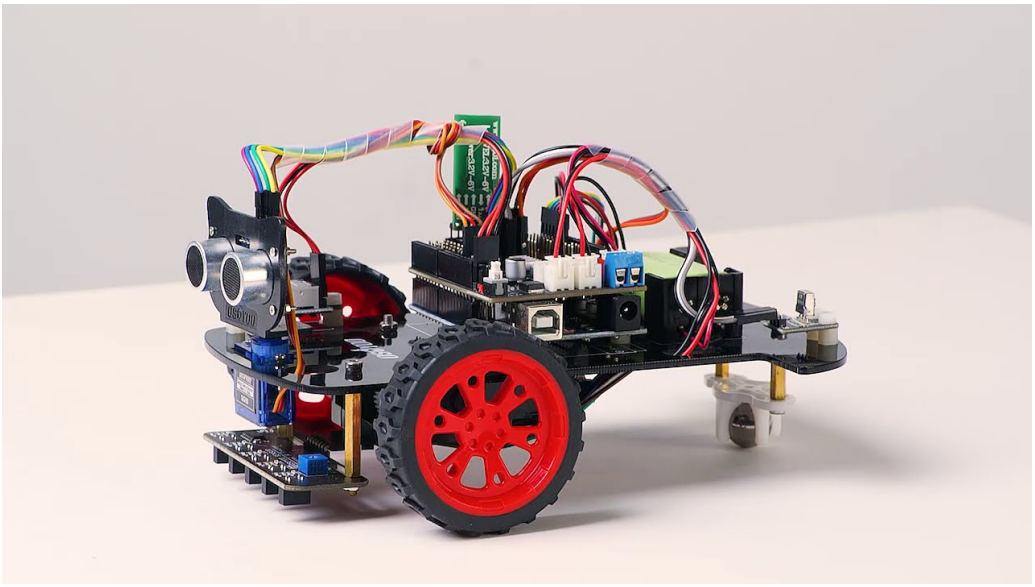


Figure 1.2: Osoyoo Model-3 V2.0 Robocar

1.2 Robocar

The Robocar used for this project is the *Osoyoo Model-3 V2.0 Robot Learning Kit* (Figure 1.2). This product is designed primarily for educational and science projects at the college level. It comes as a kit consisting of various components and devices (Figure 1.3), along with the necessary screws and nuts, neatly organized in a box with dedicated storage spaces for key parts. The first step is to carefully assemble all the components with the aid of the provided user manual and instructional video available on the product's website.

From Figure 1.3 is possible to see all the components comprehending the learning kit, which encompasses, starting from the top-left side of the Figure:

1. IR receiver,
2. 3-digit seven segment display,
3. Acoustic buzzer,
4. Servo-motor,
5. Bluetooth (BT) receiver-transmitter module,
6. pair of ultra-sonic sensors (receiver, transmitter) mounted on the appropriate support,
7. Array of 5 IR senors,
8. Remote for IR sensor,
9. Osoyoo user manual,
10. Car's chassis,
11. Arduino-compatible micro-controller,
12. Osoyoo motor shield,

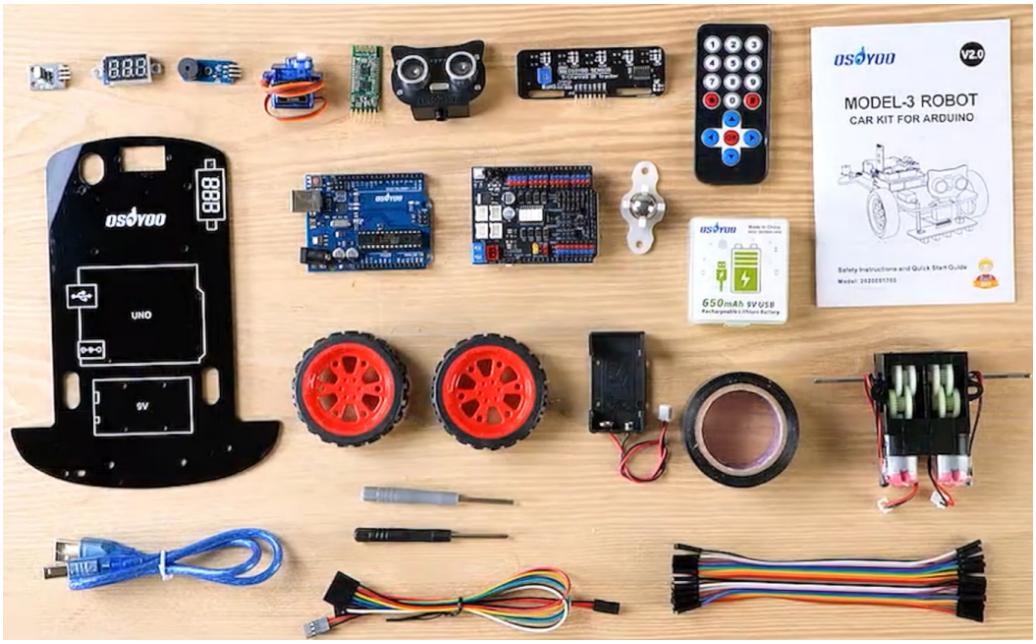


Figure 1.3: Parts of the Robocar inside the kit

13. Omnidirectional steel ball with support,
14. 9V battery case with inside 9V battery and charging cable,
15. 2 car wheels,
16. Battery support,
17. Black tape,
18. Motor couple,
19. USB cable,
20. Screwdrivers,
21. Female-female interconnection cables.

The chassis serves as the structural foundation, onto which the other components are mounted. Beneath the chassis, the 5 IR sensor array is installed, along with a pair of motors with wheels attached to the ends, and a steel ball acting as a third wheel. On top of the chassis, the microcontroller—replaced in this project with the STM32F411RE—is mounted, on it is positioned the motor shield. Some additional components, such as the IR receiver and Bluetooth module, are directly connected to the motor shield, while others, like the buzzer and ultrasonic sensors, are mounted on the chassis and connected to the board via cables. The two motors are plugged into designated sockets on the motor shield.

In the purpose of the exam, before starting the actual development, a Simulink software is provided. This software makes the car able to follow a black line drawn on the ground by the use of the IR sensor array. The car then stops when all the sensors detect dark or light ground simultaneously.

Chapter 2

Implemented Functions

This chapter is dedicated to the practical implementation of the features added upon the standard pre-defined one by the use of the given sensors and actuators.

2.1 Smoother turning behaviour

The simplest and most immediate modification implemented was to adjust the behavior of the existing function in the provided Simulink project to achieve smoother motor response when the front-mounted IR sensor array detects a turn. In the original setup, when the vehicle detects a turn, the function reverses the rotation of the inner wheel, causing it to rotate in the opposite direction to the direction of travel. This behavior is useful for navigating tight corners, allowing the robot to make sharper turns. However, when the corner is more gradual, this approach leads to a jerky movement characterized by frequent stops and starts.

To address this issue and improve the vehicle's performance on wider curves, the pre-defined model was modified. Now, when only the central sensor and its immediate neighbor detect the black line—indicating a less sharp turn—the inner wheel is either slowed down or stopped, rather than reversing direction. This adjustment allows the car to rotate smoothly, resulting in more fluid direction changes and a more refined driving. When the vehicle is traveling without problems along the way three LEDs will blink in succession.

2.1.1 Introduced Modules

In the development of this function the only modules used are the **IR array** (Figure 2.1) and the **drive motor**.

IR tracking array

The sensor array consists of 5 infrared optical sensors configured to read as digital bits. These sensors are adaptable to various complex environments thanks to an onboard

sensitivity potentiometer (blue square in Figure), which allows users to adjust the sensor's behavior according to the specific situation—such as variations in the angle of a track's curve or changes in the trigger threshold. The 5-channel infrared tracking probes can simultaneously detect the center track line as well as the left and right edges of the track.

The infrared reflective sensor utilizes ITR9909 (a 5-channel reflective infrared optical sensor mounted in line) to detect both color and distance. The working principle relies on the differing reflectivity of infrared light based on surface color, converting the reflected signal's intensity into a current signal. The sensitivity potentiometer allows the sensor to effectively adjust for variations in surface reflectivity, particularly for detecting black lines that indicate the vehicle's intended path. During detection, the sensor signals LOW when black is detected and HIGH when white is detected. The detection height is adjustable between 0 and 3 cm.

Equipped with an onboard hex inverter to provide clean digital output, this sensor is essential to our project. Its 5 integrated infrared probes—equivalent to 5 individual tracking sensors—work together to ensure smoother and more efficient line tracking. The sensor operates within a voltage range of 3 to 3.5 volts and is sensitive to dark colors and infrared light. Each of the 5 probes acts as a digital bit, transmitting a LOW output when an object is detected (such as the black track) and a HIGH output when no object is detected. If three or more probes simultaneously detect the black track, the combined digital output is LOW. This behavior can also be observed through onboard LED indicators on the sensor.

In summary, the presence of 5 tracking probes significantly enhances the tracking performance, providing smooth and reliable navigation for the robot car.

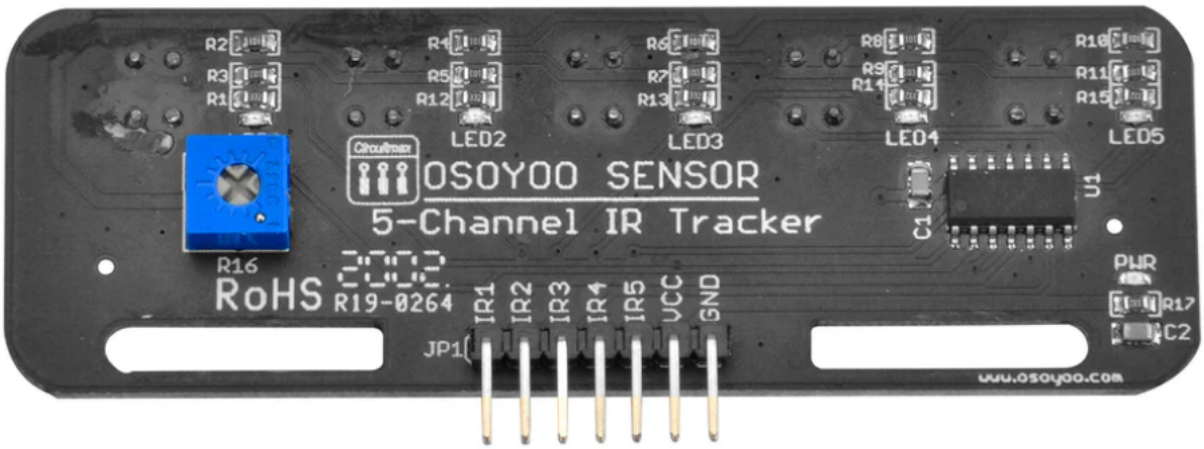


Figure 2.1: IR sensor array.

Drive Motor

The main driving motor assembly consists of two DC motors connected to gears that drive two shafts, to which the wheels are mounted. One key advantage of this configuration is the ability to control the speed of each wheel independently, enabling more advanced driving dynamics for the car. Unfortunately, due to the poor production quality, these motors are not so reliable and may happen that one of the DC motors exhibited different

rotational characteristics compared to the other, or does not rotate at all when low PWM values are provided for fine motion regulations.

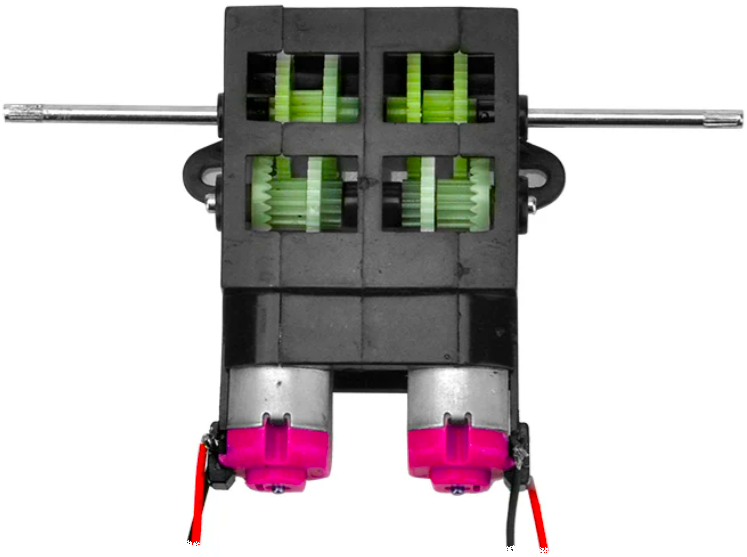


Figure 2.2: Drive motor module.

LEDs

From this initial simple function, three different LEDs have been implemented to provide the user with clear feedback on the car's status. Each function is associated with a unique LED lighting combination. These LEDs are basic in design, each powered by a 5V Vcc supply and connected in series with a resistor (Figure 2.3) to limit the current flow. In this project, the LEDs are configured using negative logic. That's because the Vcc is applied to the LED's anode, while the cathode is connected to a microcontroller pin. As a result, the LED turns on when the pin's output is set to zero, and turns off when the pin's output is set to one, achieving the negative logic configuration.

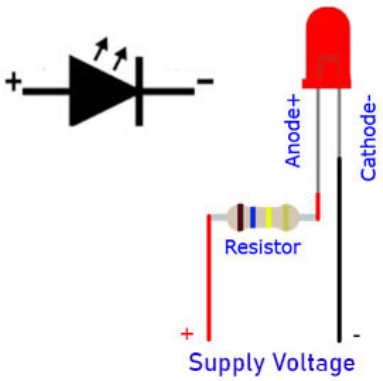


Figure 2.3: LED and resistor in series.

2.1.2 Simulink code

The logic implemented in Simulink for executing the line-following task and achieving smoother behavior is detailed across Figures 2.4 to 2.9. In Figure 2.4, the logic for moving the car forward is depicted. This happens when both the center IR tracker sensor (C) and one of the adjacent sensors (either R1 or L1) detect the black line, while the outer sensors (R2 and L2) do not. This configuration ensures the car stays aligned with the path.

Figure 2.5 Figure 2.6 illustrates the logic for executing slight turns to the left or right. The vehicle adjusts its trajectory when the sensors detect that the black line is no longer centered, causing a small directional change.

In Figure 2.7. the full-turning logic is presented. This logic comes into play when the vehicle needs to perform more significant directional adjustments, such as when encountering sharp turns on the path.

Finally, the output from these logic blocks feeds into a *MATLAB function* block, shown in Figure 2.8. This block is responsible for determining the appropriate Pulse Width Modulation (PWM) signal and direction to be applied to each motor, ensuring that the vehicle performs the intended movement—whether it's moving forward, making slight adjustments, or turning fully. Into this function is also implemented the motor control for the other function involving modifications in motor behaviour.

The LED lighting procedure is illustrated in Figure 2.9, where all the three LEDs are going to be lighten in succession.

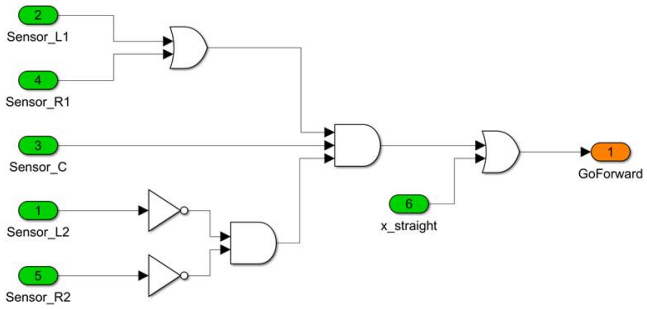


Figure 2.4: Forward line-following logic.

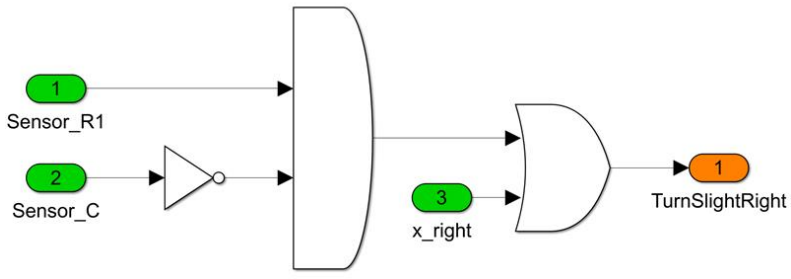


Figure 2.5: Slight turn right line-following logic.

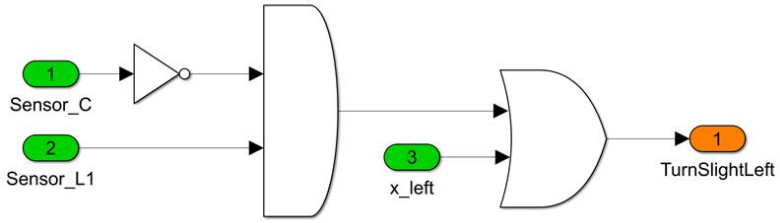


Figure 2.6: Slight turn left line-following logic.

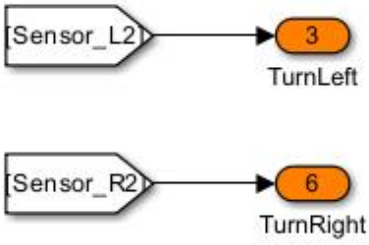


Figure 2.7: Turning line-following logic.

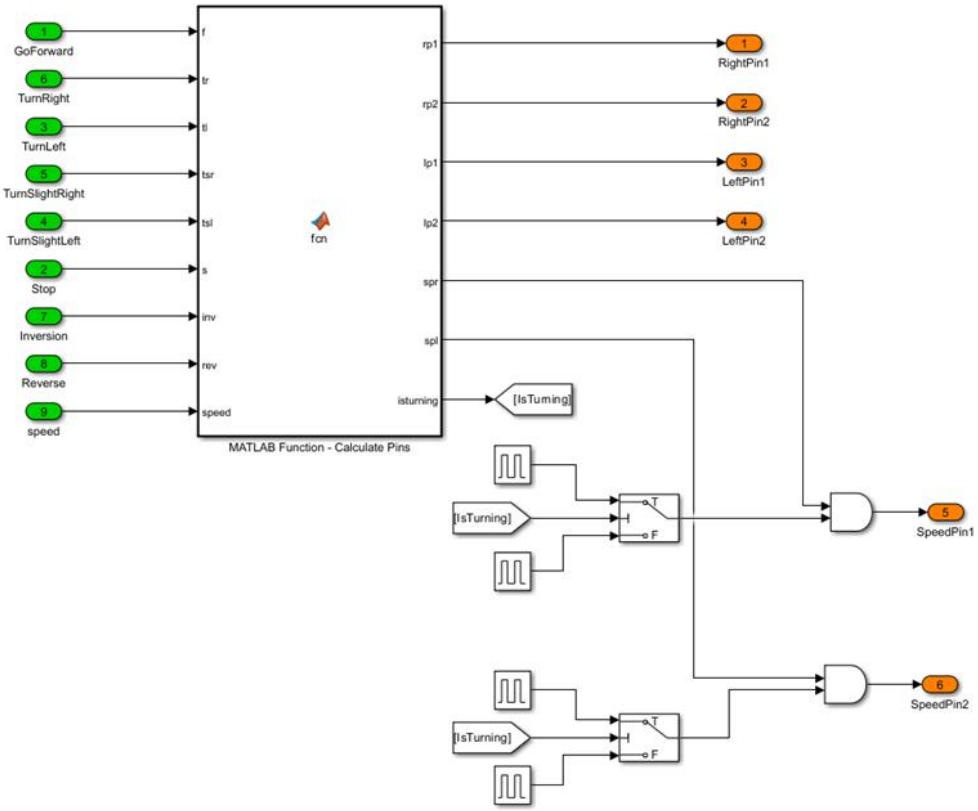


Figure 2.8: Motors movement logic.

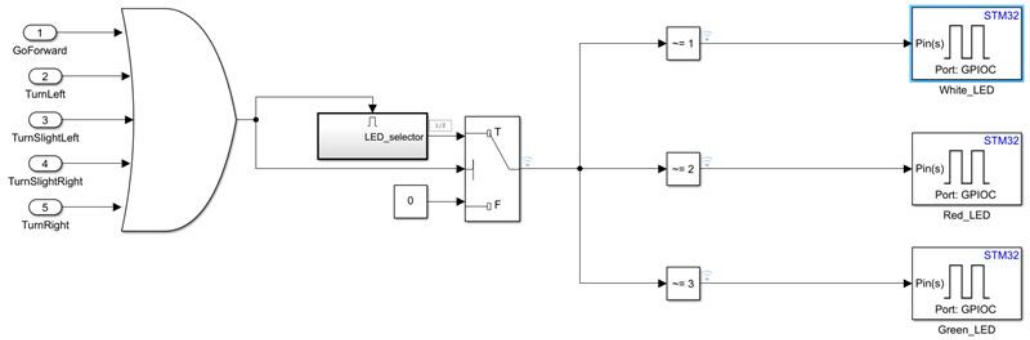


Figure 2.9: LED lighting procedure.

2.2 No road left

The second simplest function implemented involves modifying the existing model once again. As mentioned earlier, the car was programmed to stop when all 5 sensors in the IR tracker array detected either a light or dark surface simultaneously. This behavior has now been changed. In the newly developed logic, if the IR tracker array detects an entirely dark surface, it will no longer signal a stop; instead, it will indicate that the car has reached a T-crossing, triggering a more complex function that will be explained in Section 2.6.

On the other hand, when the sensor array detects an entirely light surface, it signifies that the road has reached a dead end, and the car can no longer proceed. If this condition persists for a specified minimum amount of time, the buzzer installed on the vehicle will activate, alerting the user that the car has reached the end of the road. Once the buzzer ends to ring, the vehicle will perform a 180 degree turn on the spot in a way that returns aligned again with the black strip and the execution of the line-following task can resume. When there is no road left, also the red LED will light up for signaling the status change.

2.2.1 Introduced Modules

Buzzer

In the development of this working pipeline of this function the two modules used are the IR tracking array and the Buzzer (Figure 2.10). The YL-44 buzzer module is a compact yet impactful addition to the robot car's sensory system, designed to improve situational awareness through audible cues. Operating within the audible frequency range of approximately 2 kHz, this active buzzer is straightforward and convenient, producing sound autonomously when connected to VCC and GND. Unlike its passive counterpart, the YL-44 requires no external frequency generator, making integration into the robot car's control system effortless. The buzzer is activated by setting the I/O pin to LOW, initiating sound emission, while setting the pin to HIGH or leaving it open deactivates the buzzer.

Additionally, the buzzer can be modulated using Pulse Width Modulation (PWM), allowing for nuanced control over sound intensity. By leveraging Simulink, PWM signals

are generated through a *pulse genertor* block, enabling precise timing control to simulate the sound of a real car indicator. The YL-44 buzzer serves as an auditory signal, alerting nearby entities to the robot car's presence and signaling upcoming maneuvers, thus enhancing safety and communication in dynamic environments.

Figure 2.10: Buzzer module.

2.2.2 Simulink code

As previously mentioned, the expected behavior of the car in this function is to stop when no road remains. The complete stop logic is illustrated in Figure 2.11. This figure also contains logic for other functions, which will be explained later. For the purpose of this function, the relevant part is located in the lower section of the diagram, where the system instructs the car to stop when none of the IR tracker sensors detect the black line on the road.

Once this condition is met, the function triggers the output signal to activate the *buzzer*, which then flows into an *OR gate* that collects all stop-related logic signals. Additionally, from this figure, the implementation of the red LED lighting is visible, indicating the vehicle's status whenever the no-road-left logic is activated.

Finally, is also possible to see that this stop condition also activates the *inversion* routine after a certain amount of time, which gives the motor control function the indication to perform the 180 degree rotation movement, for the time needed for the rotation to be completed.

The Buzzer rings with three different tones, following the logic implemented in Figure 2.12 which takes in input the *buzzer* signals coming from the stop logic. The higher part of the Figure is dedicated to the current function, it is used to submit the buzzer three different frequencies in succession. The inside of the *Buzz-routine* subsystem is reported in Figure 2.13, which implements the logic with the timings needed for the generation of the three sounds in succession.

2.3 Emergency braking System

The first key feature implemented in this car project is the **Emergency Braking** function. While the effect of this feature is straightforward, its practical implementation is more

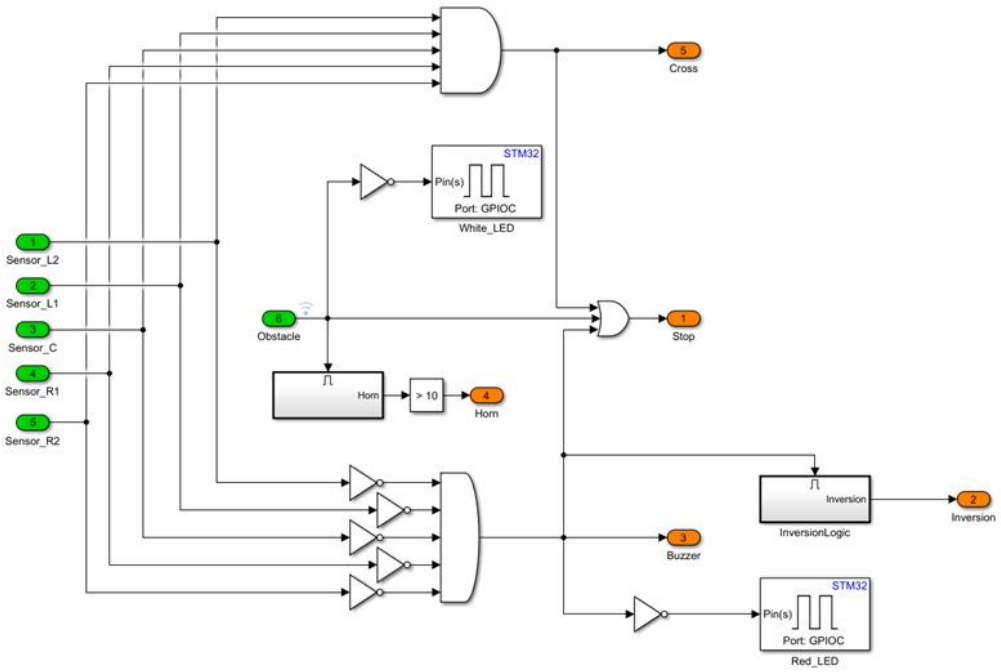


Figure 2.11: Stop logic.

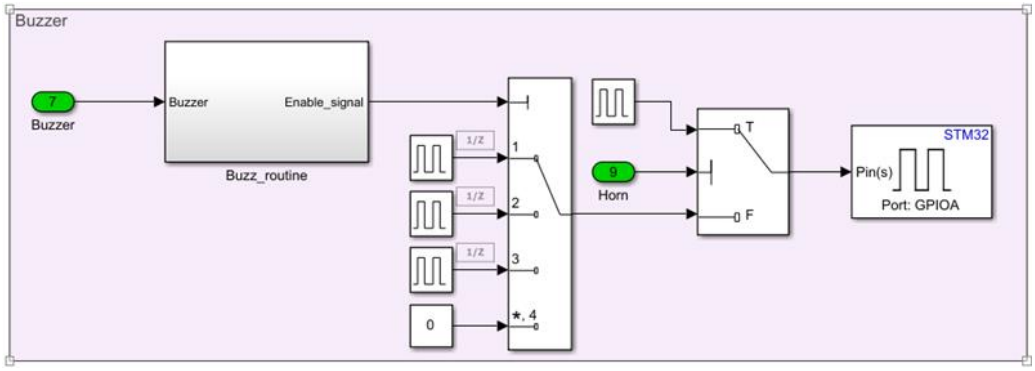


Figure 2.12: Buzzer logic.

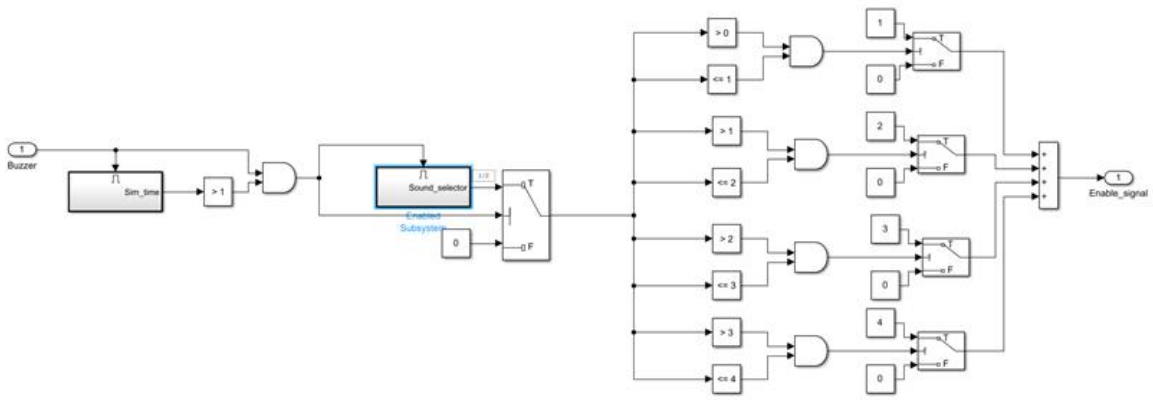


Figure 2.13: Inside of Buzz-routine subsystem.

complex. This function utilizes the ultrasonic sensor module (UT) to detect obstacles within a specified distance. When an obstacle is detected within this range, the car

immediately stops moving until the obstruction is cleared. If the obstacle remains for more than five seconds, the car activates the horn (simulated by a low sound from the buzzer), in true Italian style. During this stop, a white LED lights up, signaling the car's current status.

2.3.1 Introduced Modules

UT sensor

The most difficult part of this feature is the use of the UT sensor, since it is composed by two parts that must interact perfectly in order to receive the desired signal. The ultrasonic HC-SR04 distance measuring transducer sensor (Figure 2.14) is a key component consisting of an ultrasonic transmitter, receiver, and control unit. Known for its affordability and ease of integration, the HC-SR04 provides a measuring range from 2 cm to 400 cm, making it highly effective for obstacle detection, navigation, and even cruise control functionalities.

Figure 2.14: UT module.

The sensor operates similarly to SONAR systems used in submarines, emitting ultrasonic sound pulses and detecting their reflections to calculate the distance to nearby objects. Equipped with two ultrasonic transducers—a transmitter and a receiver—the HC-SR04 utilizes a simple four-pin interface, making it easy to connect with the STM32 microcontroller board. The VCC and GND pins supply power, while the Trig and Echo pins connect to digital GPIO pins, allowing for signal control and data reception. When the Trig pin is set to high, the sensor emits ultrasonic waves, and the Echo pin receives the reflected signals, enabling accurate distance calculations.

This functionality is based on the principle of echolocation: the time taken for the ultrasound pulse to travel to and from an object allows the sensor to compute the distance. The HC-SR04 sensor thus plays a crucial role in improving the robot-car's spatial awareness, enhancing its navigation capabilities.

One of the main challenges in using the HC-SR04 with the STM32 microcontroller was generating the correct trigger signal and accurately reading the reflected pulse. Achieving precise timing for these signals was critical for reliable distance measurements. Despite this, the sensor's inclusion significantly enhances the robot's ability to detect and avoid obstacles, making it an indispensable part of the project's navigation system.

2.3.2 Simulink code

The effectiveness of this function hinges on the precise control of the UT sensor's over the generation and detection of ultrasound signals. The process begins by setting the Trig pin to high for 10 microseconds, triggering an 8-cycle ultrasonic burst at 40 kHz, traveling at the speed of sound. The Echo pin then transitions to high, signaling that it is listening for the reflected wave. If no object is detected, the Echo pin returns to low after 38 milliseconds, timing out. However, if an object reflects the pulse back to the sensor, the Echo pin returns to low before the timeout, allowing for the calculation of the distance using the formula $distance = speed \times time$.

The Simulink models used for implementing these statements is reported in Figure 2.15, in which is possible to see the ultrasonic signal generation on the top-right, while on the left the USART CH1 ports takes in input the signals coming from the Echo pin on the UT module. This signals is then manipulated and filtered to extract the distance in centimeters from the obstacle. The resulting signal is then compared to a threshold (Figure 2.16).

If the condition is satisfied the *obstacle* variable goes high, which then enters in the stop logic in Figure 2.11 an then into the motor control function (Figure 2.8), resulting in the car stopping. The LED implementation is similar to the one depicted in Figure 2.11. When the related variable goes high, a *NOT* port is placed before the GPIO block because of the negative logic in the LED implementation.

As said before, if the object is detected for more than 2 second the buzzer should ring. To do this the lower branch of Figure 2.12 is used to make the buzzer able to produce the predefined sound for the time necessary.

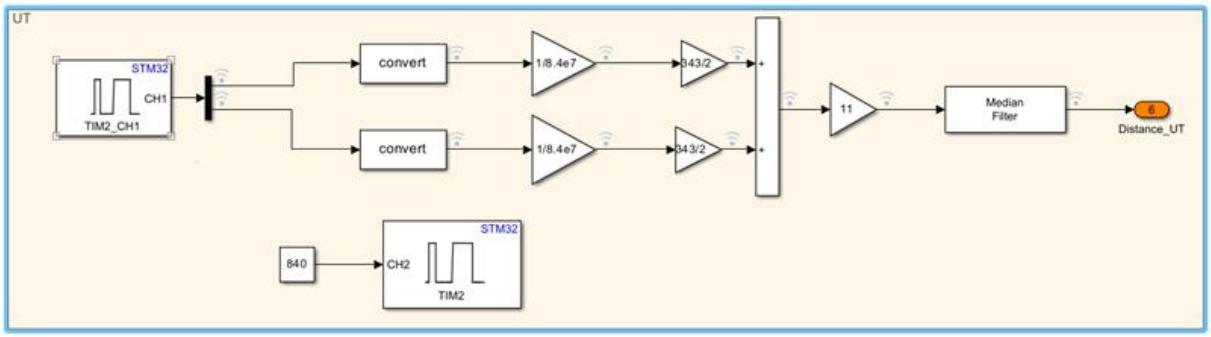


Figure 2.15: UT signal generation and collection.

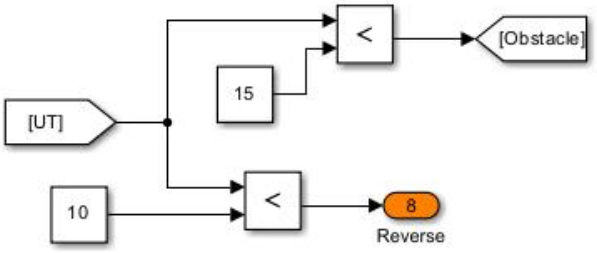


Figure 2.16: UT threshold comparison.

2.4 No contact

This new simple function, called **No contact** uses the UT module to understand if an object is closer than a certain threshold, if this happens the car will retreat along the way in order to avoid the contact with the approaching object. No LED will light up during the execution of this function.

2.4.1 Simulink code

This function primarily utilizes the logic implemented in Figure 2.15, with modifications to suit its unique purpose. This time, the output of the manipulations is compared to a lower threshold than the one used in the previous function (see the lower part of Figure 2.16). When the condition is met—indicating the presence of an object closer than the required distance—the *Reverse* variable is triggered, signaling that the object is too close. This variable is then fed into the motor control function, which causes the car to reverse, moving away from the approaching object.

In this instance, the ultrasonic sensor (UT) is employed dynamically rather than statically, adding complexity to its use. When the car is moving backward, the sensor struggles to accurately determine the distance to the object. As a result, the sensor produces fluctuating signals, as shown in Figure 2.17, where the blue line represents the computed distance and the red line represents the value of the Reverse variable. The up-and-down behavior in the blue line reflects inconsistent distance measurements and so the red line, leading to a fragmented, jerky movement of the car. Although this behavior is not ideal, no alternative methods were found to effectively implement this function.

2.5 Parking maneuver

Another important feature implemented in this project is the **parking maneuver**. This function leverages the use of the *IR receiver* module and its *remote control*, allowing the car to simulate parking when traveling along a straight section of the black path. The parking maneuver can be activated via the IR remote. Upon receiving the input signal, the car will pull over to the side of the path, mimicking the behavior of parking.

When the car receives another input from the remote, it will exit the parked state and return to the path, resuming the line-following task. To accomplish this, two distinct motor maneuver strategies are required—one for pulling over and another for returning to the path.

Throughout the entire parking task, the green LED will light up, signaling the car's change in status, indicating that it is in the parking mode. This visual cue helps users easily track the state of the car during the operation.

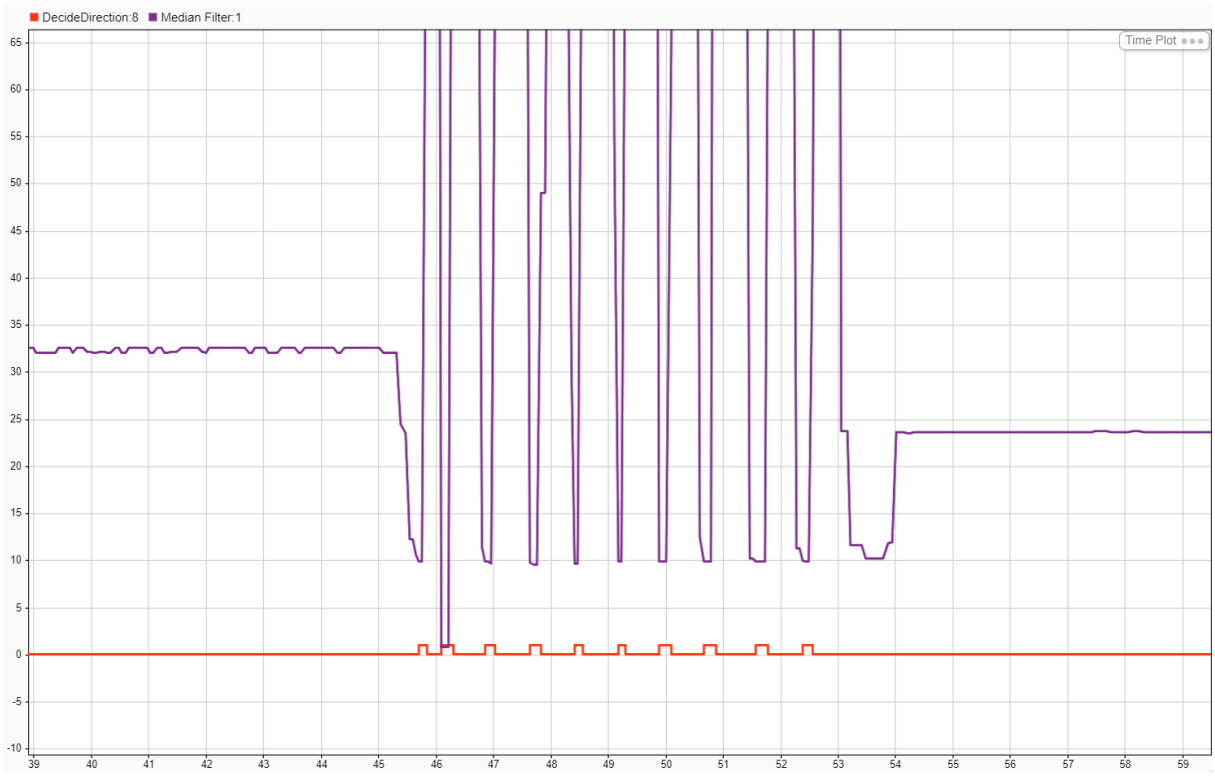


Figure 2.17: Scope on No Contact.

2.5.1 Introduced Modules

IR remote and receiver

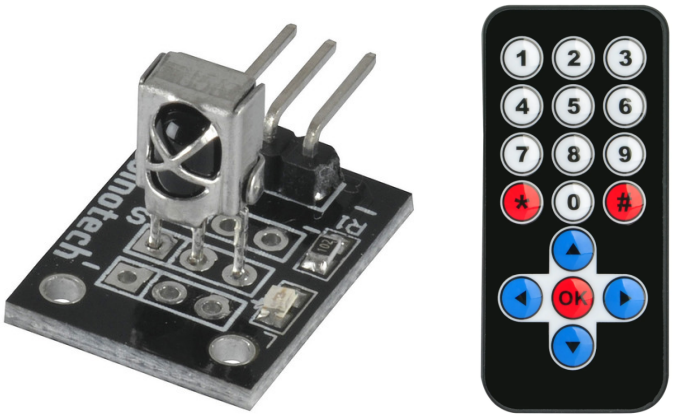
The **IR receiver HC02** is a compact, highly efficient sensor designed to detect infrared (IR) signals emitted from its **remote controller** (Figure 2.18). It converts the incoming modulated IR signals into a digital signal, allowing them to be processed by a microcontroller. In essence, the remote emits a series of infrared light pulses, and the HC02 acts as a photodiode, detecting these incoming signals.

One of the key features of the HC02 is its ability to filter out unwanted ambient infrared light and noise, achieved through a band-pass filter tuned specifically to the modulation frequency of the remote control—typically around 38 kHz for most commercial remotes. This filter ensures that the sensor is responsive only to the signals sent by the remote control, ignoring other sources of infrared radiation in the environment.

Once the HC02 receives a valid signal, it outputs a digital pulse train corresponding to the data transmitted by the remote control. This signal can be read by a microcontroller via one of its GPIO pins, which interprets the pulses to decode the transmitted commands. The HC02 sensor connects to the microcontroller using three pins:

1. VCC – Provides power to the sensor, usually 5V.
2. GND – Ground connection.
3. DATA – Signal pin that delivers the digital output to the microcontroller's GPIO pin.

Although the HC02 is quite effective at filtering and detecting IR signals, it does have some limitations. The range is relatively short, and the remote control must be properly



(a) IR receiver (b) IR remote

Figure 2.18: IR reciver module and Remote

aimed directly at the IR receiver for the sensor to function correctly. This is because the sensor's reception area is narrow, and off-axis signals may not be detected too reliably. Despite this, the HC02 provides a simple and effective way to interface remote controls with microcontrollers for various applications, making it an essential component in the control system of this robot-car project.

2.5.2 Simulink code

This function integrates the use of the IR remote and IR receiver to control the car's actions. While the car is following the black path, pressing a button on the IR remote triggers the **parking maneuver** function. It starts with the acquisition of the steps signals coming from the IR remote with a GPIO input port (on the left in Figure 2.19), its input then goes into the function the contains the logic aimed to filter the signal and output the number of negative edges sent by the IR remote (Figure 2.20). The output then consists of an integer number set as *IR Number* variable. The IR-number variable is crucial for determining when to initiate the parking maneuver and when to resume following the road. Practically, the function works by counting the number of positive edges, and depending on whether this count falls between two predefined thresholds, it signals the start of the parking maneuver. Once the number of edges surpasses the higher threshold, the car returns to the road and resumes its original path-following behavior.

This variable goes then into the function dedicated to the motor control for the parking maneuver (Figure 2.21), which control the motors speeds and direction to let the car perform the correct movements.

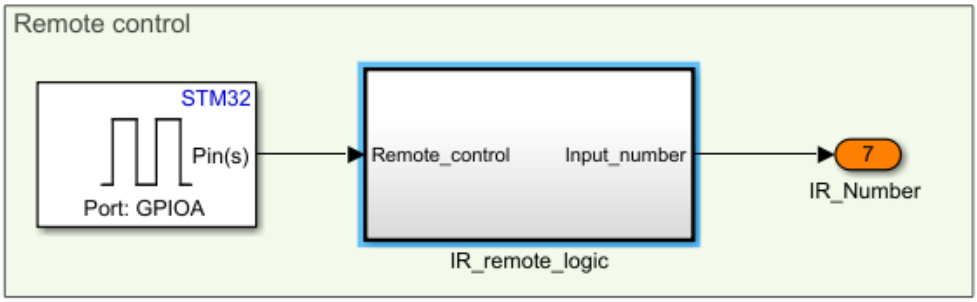


Figure 2.19: Scope on No Contact.

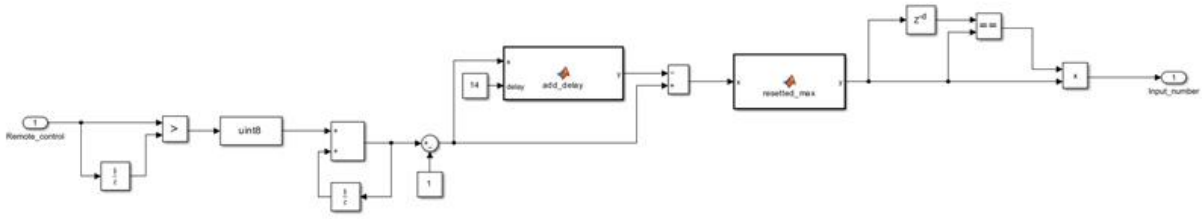


Figure 2.20: IR receiver falling edges counter.

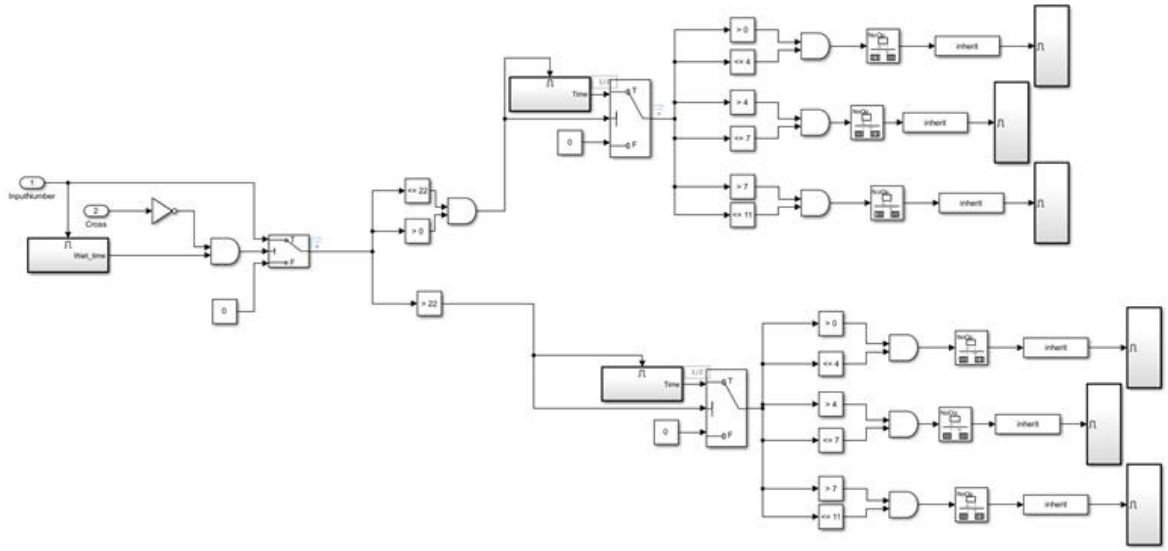


Figure 2.21: Parking function motor control.

2.6 X Crossroad

X crossroad is the name given to the last implemented features, the most complicated, that integrates a series of actuation in order to be completed. The feature start is triggered when the car approaches a crossroad, being it at T o X. By doing that all the IR tracking sensors will see dark ground, letting the car stop on the crossroad.

The vehicle will then receive an impulse train signal used to select the chosen direction. Moreover, before selecting its direction the vehicle will scan its surrounding thanks to the

UT sensor and the motion of the servomotor, by performing a rotation from -45 degrees to 45 degrees.

2.6.1 Introduced Modules

Servomotor

The OSOYOO SG90 micro servo motor (Figure 2.22) plays a crucial role in the robot-car's architecture, primarily serving as the mounting platform for the ultrasonic sensor. This integration allows the sensor to scan its environment over a 180° range, from right to left, enhancing the vehicle's ability to detect obstacles from multiple angles. Known for its compact size and high torque output, the SG90 servo is versatile and reliable, making it a popular choice in robotics and modeling applications.

Figure 2.22: Servo motor.

With a rotational range of up to 180°, the servo motor provides precise movement control, which is vital for directing the ultrasonic sensor to various positions, thereby increasing the car's capacity to detect objects across different orientations. The motor operates based on pulse width modulation (PWM), interpreting signals to correspond with specific angular positions. This makes the motor highly suitable for seamless integration within the robot-car's control system.

2.6.2 Simulink code

In order to correcly use the servomor PWM signals are required. To generate it, TIMER 1, a 16-bit timer, was utilized. A pre-scaler was set to 83, dividing the frequency by 84 to achieve a frequency of 1 MHz, ensuring optimal performance while maintaining energy efficiency. The servo motor's rotation angles were finely tuned through the calibration of counts provided by the Simulink models. Through empirical testing, it was determined that 800 counts correspond to a 45-degree clockwise rotation, 2350 counts achieve the opposite counter-clockwise rotation, 1575 counts position the servo motor in alignment with the vehicle's orientation, facilitating optimal functionality within the robot-car system.

This precise control of the servo motor allows the ultrasonic sensor to effectively scan the surroundings and adjust its position dynamically based on the car's movement. Figure 2.23 shows the motor PWM selection, while Figure 2.24 shows the inside of the *Turn routine*, dedicated to decide the rotation direction for the servo-motor.

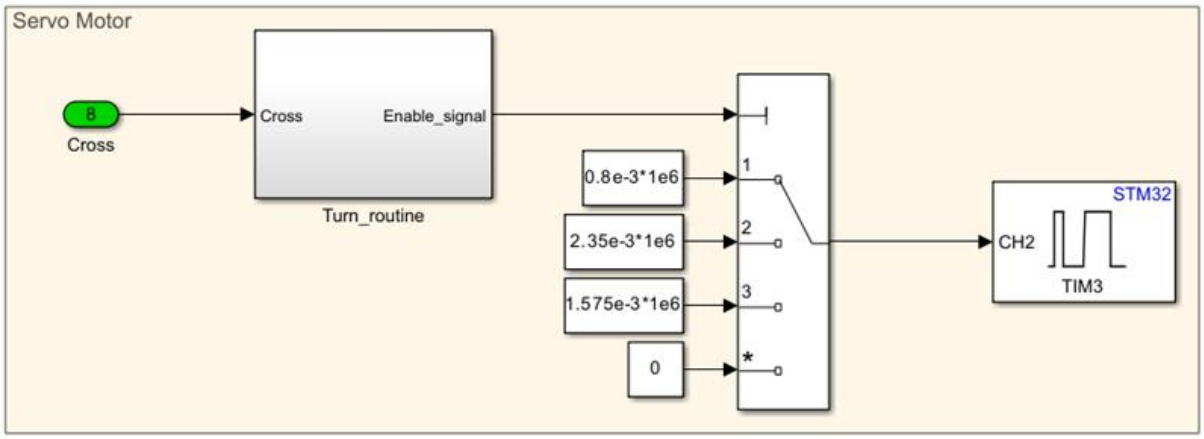


Figure 2.23: Servo motor direction selection.

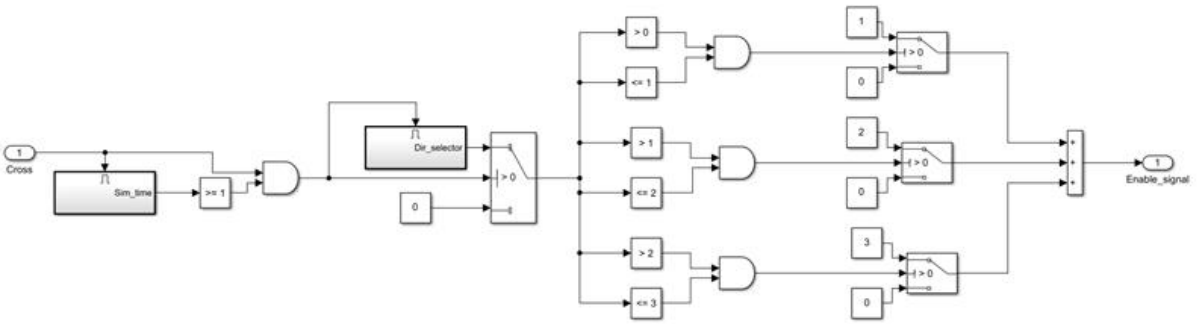


Figure 2.24: Servo motor routine subsystem.

Chapter 3

QFD and FMEA

For completing the exam is also necessary to present the Quality Function Deployment (QFD) and the Fault Mode and Effect Analysis (FMEA) for the developed project.

3.1 QFD

Quality Function Deployment (QFD) is a structured methodology used in product and service development to align the final product with customer needs and expectations. It serves as a bridge between customer demands (often termed the "voice of the customer") and the technical or design specifications that guide the development process. The primary objective of QFD is to ensure quality at every stage of production, reducing the risk of misalignment between customer desires and the engineering output.

The process is typically structured into a series of matrices, sometimes called the "House of Quality," where customer requirements are mapped against technical specifications to highlight areas for improvement and to address potential trade-offs. Here's how the QFD process generally unfolds:

1. Gathering Customer Input: The process begins by collecting detailed feedback from customers to identify their needs, desires, and expectations. These are referred to as customer attributes or "what's". Each attribute is evaluated by the customer in terms of its importance, typically on a scale from 1 to 6, reflecting how crucial that feature is to the customer.

2. Evaluating Customer Perception: After assigning importance values to each "what," a customer perception score between 1 and 6 is assigned for both the current project and a competitor's product for each customer requirement. This allows a comparative analysis of how well the current project meets customer expectations versus the competition.

3. Calculating the Importance Score: The difference between the competitor's best score and the score of the developed project is calculated for each "what." This difference is then multiplied by the customer importance number, resulting in an absolute importance score for each customer requirement. This step helps prioritize areas where the product is falling behind in customer expectations and needs improvement.

4. Translating Requirements into Specifications: The next phase involves converting the customer requirements (the "what's") into measurable engineering or technical specifications (the "how's"). This is done by identifying key features or technical characteristics that can meet the customer's needs, allowing the engineering team to focus on specific, actionable areas for development.

By systematically analyzing and mapping customer needs against product specifications, QFD ensures that customer priorities are at the forefront of the development process. This method helps minimize misunderstandings and ensures that the engineering team delivers a product that aligns with customer expectations, enhancing overall satisfaction and product quality. The developed QFD for this project is reported in Figure 3.1.

3.2 FMEA

Fault Mode and Effect Analysis (FMEA) is a structured and systematic methodology used to identify, analyze, and prioritize potential failure modes in a system, process, or product. Originally developed for the aerospace industry, FMEA is now widely adopted in various sectors such as automotive, manufacturing, and healthcare due to its effectiveness in improving reliability, safety, and product quality. The core aim of FMEA is to proactively detect potential failure points and mitigate risks before they occur, ensuring that issues are addressed early in the design or process phase.

By analyzing each component, subsystem, or process step, FMEA allows teams to anticipate faults, assess their impact, and prioritize corrective actions. The analysis is typically guided by three key factors:

Severity (S) This measures the seriousness of the effect of a failure mode on the overall system. It is assigned a score between 1 and 10, where higher scores indicate more severe consequences.

Occurrence (O) This assesses the likelihood of a failure occurring due to a specific cause. Like severity, this is also rated on a scale from 1 to 10, with higher scores indicating a greater likelihood of occurrence.

Detection (D) This evaluates the effectiveness of current controls in detecting potential failure modes before they affect the product or process. A score from 1 to 10 is assigned, where higher scores reflect a lower probability of detection before the issue impacts the product.

These three scores are multiplied to calculate a Risk Priority Number (RPN) for each potential failure mode:

$$RPN = S \times O \times D \tag{3.1}$$

The RPN provides a quantitative way to prioritize risks, where a higher score indicates a higher-priority risk that needs to be addressed. FMEA tables typically organize information into columns for listing functions, potential failure modes, effects, causes, current controls, and their associated severity, occurrence, and detection scores.

3.2.1 Steps is FMEA

1. List Functions and Potential Failure Modes: In the first column, list all the functions of the system, product, or process. In the next column, identify potential failure modes for each function.

2. Identify Effects and Assign Severity: For each failure mode, describe its potential effects on the system's performance and assign a severity score (S) from 1 to 10, where higher numbers represent more critical effects.

3. Determine Causes and Occurrence: Next, identify the potential causes for each failure mode and assign an occurrence score (O), indicating the likelihood of that failure occurring.

4. Evaluate Detection Controls and Assign Detection Score: List current prevention and detection measures in place to address each failure mode and assign a detection score (D) based on the effectiveness of these controls.

5. Calculate RPN: Multiply the three scores (S, O, and D) for each failure mode to determine its RPN. This will guide the prioritization of corrective actions.

6. Take Corrective Action: For failure modes with high RPN scores, initiate corrective actions aimed at reducing one or more of the contributing factors (severity, occurrence, or detection) to bring down the RPN. A post-corrective action FMEA is then conducted to reassess the RPN after improvements.

3.2.2 Key Objectives of FMEA

- Minimizing Risk: By identifying and addressing high-RPN failure modes, FMEA reduces the likelihood of defects and improves product quality.

- Improving System Robustness: A thorough FMEA ensures that the system is more resilient against potential failures, leading to a reduced RPN score in future evaluations.

- Enhancing Design and Process Optimization: FMEA insights help in fine-tuning designs and processes to prevent failures from occurring in the first place.

In conclusion, FMEA is an invaluable tool for improving reliability and quality by systematically identifying and addressing potential risks in design and processes. It not only aids in reducing the likelihood of failure but also helps teams focus on areas for improvement, ensuring robust performance and safety in the final product. The developed QFD for this project is reported in Figure 3.2.

	Traking Sensors	\															
	UT distance measurement	3	\														
	Motors spin differentiation	4	1	\													
	IR signal receiving	1	2	3	\												
	LED lighting	0	1	1	2	\											
	Buzzer acoustic signaling	0	1	1	2	4	\										
	7-segment voltmeter	0	0	0	0	0	0	\									

▲ no relation
▭ balanced
● strong

Target		Hows							Customer perception					
Whats	Importance	Traking Sensors	UT distance measurement	Motors spin differentiation	IR signal receiving	LED lighting	Buzzer acoustic signaling	7-segment voltmeter	our product	competitor 1	competitor 2	best	GAP	Absolute importance
Forward Motion	6	● 54	▲ 0	▭ 18	▲ 0	▲ 0	▲ 0	▲ 0	6	6	4	6	1	6
Follow trajectory	6	● 54	▲ 0	● 54	▲ 0	▲ 0	▲ 0	▲ 0	6	6	5	6	1	6
Crossroad reliable	5	● 45	▭ 15	● 45	▭ 15	▲ 0	▲ 0	▲ 0	5	3	6	6	1,2	7,2
Safety	6	▲ 0	● 54	▲ 0	▲ 0	▭ 18	▭ 18	▲ 0	6	5	3	6	1	6
Ease of parking	4	▲ 0	▲ 0	● 36	▭ 12	▲ 0	▲ 0	▲ 0	4	4	5	5	1,25	6,25
Status signaling	2	▭ 18	▭ 18	▲ 0	▲ 0	● 18	● 18	▲ 0	2	1	4	4	2	8
Battery output	1	▲ 0	▲ 0	▲ 0	▲ 0	▲ 0	▲ 0	● 9	1	1	2	2	2	4
Remote control	3	▲ 0	▲ 0	▲ 0	● 27	▲ 0	▲ 0	▲ 0	3	5	6	6	2	12
Absolute index		171	87	153	54	36	36	9						
Relative index		31,32%	15,93%	28,02%	9,89%	6,59%	6,59%	1,65%						

Figure 3.1: QFD.

							Current Control				
S. No.	Function	Potential failure mode	Potential Effect of failure	Severity (S)	Causes	Potential Causes of Failure	Occurrence (O)	Prevention	Detection	Detection (D)	R.P.N.
1	Smoother turning behaviour	Motors do not spin	Car does not move	10		Damage, incorrect wiring	1	\	\	10	100
		IR tracker not working	Car does not follow the trajectory	10		Incorrect wiring	2	Assemble the kit carefully	Check led on sensor	4	80
		LED not lighting	No status signaling	3		Incorrect wiring	4	Assemble the kit carefully	Check port Vcc output	3	36
2	No road left	Motors do not spin	No 180 degree rotation	10		Damage, incorrect wiring	1	\	\	10	100
		IR tracker not working	No road end awareness	10		Incorrect wiring	2	Assemble the kit carefully	Check led on sensor	4	80
		LED not lighting	No status signaling	2		Incorrect wiring	4	Assemble the kit carefully	Check port Vcc output	3	24
		Buzzer not working	No status signaling	3		Overheating	2	\	Check port Vcc output	3	18
3	Emergency braking System	UT sensor not working	No obstacle awareness	10		Damage	2	\	\	10	200
		LED not lighting	No status signaling	2		Incorrect wiring	4	Assemble the kit carefully	Check port Vcc output	3	24
		Buzzer not working	No status signaling	3		Overheating	2	\	Check port Vcc output	3	18
4	Parking manoeuvre	IR receiver not working	No remote control	8		Dead battery, out of range	3	Charge batteries before use	\	10	240
		Motors do not spin	Car does not move	9		Damage, incorrect wiring	1	\	\	10	90
		LED not lighting	No status signaling	3		Incorrect wiring	4	Assemble the kit carefully	Check port Vcc output	3	36
5	X Crossroad	IR receiver not working	No remote control	8		Dead battery, out of range	3	Charge batteries before use	Check led on sensor	10	240
		UT sensor not working	No obstacle awareness	6		Damage	2	\	\	10	120
		Servo Motor not spinning	No surrouding view	7		Damage, overheating	2	\	\	10	140
		IR tracker not working	No crossroad awareness	10		Incorrect wiring	2	Assemble the kit carefully	Check led on sensor	4	80
		LED not lighting	No status signaling	3		Incorrect wiring	4	Assemble the kit carefully	Check port Vcc output	3	36
6	No contact	Motors do not spin	Car cannot retreat	7		Damage, incorrect wiring	1	\	\	10	70
		UT sensor not working	No object approaching awareness	7		Damage	2	\	\	10	140

Figure 3.2: FMEA.

Chapter 4

Conclusions

In conclusion, the development of this project required substantial time and effort. At the outset, the primary objective was to define the essential functions of the system and devise an effective implementation plan. Having a clear outline of tasks and execution strategies was crucial, as it kept us focused on our goals and ensured that we maintained a consistent direction throughout the project.

The first hands-on task involved assembling the robot kit-car, which, while relatively straightforward in hindsight, was a significant milestone. Despite being one of the easier tasks, it played a pivotal role in building the foundational structure upon which all subsequent tasks were based.

One of the major challenges we faced involved developing the system model in Simulink and working with the STM Cube IDE for the first time. None had prior experience with programming an STM Nucleo board, and our familiarity with Simulink tools was limited. This process encountered obstacles such as integrating sensors, configuring timers, and troubleshooting complex code.

Looking back, the results have been deeply rewarding. Each feature—from line-following to obstacle detection and emergency braking—was meticulously implemented to ensure thier working in every conditions, without cross-correlations with other pieces of code, resulting in bugs and malfunctions. These functionalities showcase the culmination of our efforts and reflect the depth of the learning throughout the exam preparation period.

The car now navigates smoothly along its path, effectively detects and avoids obstacles, and executes emergency stops with the activation of the buzzer sensor when necessary. Moreover, the implementation of functions with remote control opened new horizions to this type of projects.

In conclusion, the project provided valuable insights and practical experience in robotics, embedded systems, and model-based design. Taking pride in not only achieving prefixed goals but also laying the groundwork for future improvements and expansions, ensuring that this project has the potential for further development and upgrades.

[1]

[1] A tribute to all the coffee drank by the authors during the development of this project.

Bibliography

[1] Osoyoo. (2020, May 22). Osoyoo Model-3 V2.0 Robot Learning Kit Model2020001700. https://osoyoo.com/2020/05/22/osoyoo-model-3-v2-0-robot-learningkit/

[2] STMicroelectronics. (n.d.). STM32 Nucleo-64 boards (MB1136) User Manual. https://www.st.com/resource/en/user_manual/um1724-stm32nucleo64-boards-mb1136-stmicroelectronics.pdf

[3] STMicroelectronics. (n.d.). STM32F411RE Specifica ons. https://www.st.com/en/microcontrollers-microprocessors/stm32f411re.html

[4] STMicroelectronics. (n.d.). STM32F411RE Datasheet. https://www.st.com/resource/en/datasheet/stm32f411re.pdf

[5] SparkFun Electronics. (n.d.). Ultrasonic Ranging Module HC-SR04 Datasheet. https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[6] TowerPro. (n.d.). SG90 9g Micro Servo Datasheet. https://datasheetspdf.com/mobile/791970/TowerPro/SG90/1

[7] Osoyoo. IR receiver module, https://osoyoo.com/2017/08/15/ir-receive-module-and-ir-remote-controller/