ARI2202 - Assignment

Robotics Assignment

Roman Országh Jr., Nathan Bonavia Zammit, Ben Spiteri

Bachelor of Science in

Information Technology (Honours)

(Artificial Intelligence)

May 2022

Contents

GOALS

We were assigned with the task to get a robot through an obstacle course. Said course consisted of 5 black lines which were 2 tiles wide (approx 1.2 m total width), and a number of different placed obstacles which will differ for every time the course is run. In addition, each robot will be placed at a different orientation in the beginning and we were tasked with correcting said orientation. The end goal would be to reach the end of the course in the most time efficient way possible. To reach this goal we used multiple different tools that came with the robot set, these include using the ultrasonic sensor in order to be able to avoid the objects in the course, the gyroscope to set the robot's position in order to always move in the correct forward direction. The line tracking module in order to identify whenever the robot is on a black line. An ultrasonic sensor to detect the obstacles ahead of the robot. Combining all of these modules would allow us to reach the end goal as required. During the testing and building of the code we encountered multiple different errors and inaccuracies which we needed to alter and correct, however as we found out, certain inaccuracies simply came down to hardware limitations from the robot itself. Certain alterations were implemented in order for said limitations however some of these were out of our hands, or were discovered at a late time and as such time constraints simply did not allow us to find any possible fixes.

RELEVANT LITERATURE

We were presented with the Elegoo Smart Robot Car v4.0 for our assignment. This robot is assembled with multiple different components such as a gyroscope, ultrasonic sensor, camera and much more. Said components were all connected through a Elegoo UNO R3 Board along with an I/O Expansion Board. An Elegoo UNO R3 Board is exactly the same as an Arduino UNO R3 Board however the difference being that it is made by the company Elegoo, and it doesn't support the Arduino Project and does not include the Arduino IDE [1]. An I/O Expansion Board is often used to expand the number of pins available to most microcontrollers as they usually come with a very limited number of pins, and hence a limited number of interfaces with the outside world. An I/O Expansion Board takes a few of those pins and then fans them to more pins [2]. Through the use of these boards, we were able to connect all the required modules together. An example of these modules would be the GY-521 Module, the Ultrasonic Sensor Module, the Camera Module, and the Line-Tracking Module. All these modules combined allowed us to tackle the task with which we were presented with, except for the Camera Module as that wasn't needed or used at any point throughout our work.

The GY-521 is a breakout board for the MPU-6050 MEMS (Microelectromechanical systems) that features a 3-axis gyroscope, a 3-axis accelerometer, a digital motion processor (DMP) and a temperature sensor. The DMP can be used to process complex algorithms that turn the raw values from the sensors into stable position data. The DMP processes algorithms tha turn the raw values from the sensors into stable position data. The sensor values are retrieved by using the I2C serial data bus, which requires only 2 wires, SCL and SDA [3].

The GY-521 breakout has eight pins [3]:

- VCC (The breakout board has a voltage regulator. Therefore, you can connect the board to 3.3V and 5V sources.)
- GND
- SCL (Serial Clock Line of the I2C protocol.)
- SDA (Serial Data Line of the I2C protocol.)
- XDA (Auxiliary data => I2C master serial data for connecting the module to external sensors.)
- XCL (Auxiliary clock => I2C master serial clock for connecting the module to external sensors.)
- AD0 (If this pin is LOW, the I2C address of the board will be 0x68. Otherwise, if the pin is HIGH, the address will be 0x69.)

- INT (Interrupt digital output)

Regarding the Elegoo Smart Robot car v4.0 which we used, all wires and pins come prepared from beforehand and all we had to do was simply connect them to the boards as was asked of us in the manual which was provided.

Our Elegoo Smart Robot car v4.0 came with an Ultrasonic Sensor HC-SR04 which is a sensor which can measure distance. It emits an ultrasound at 40,000 Hz which travels through the air and if an object or obstacle is its path, it will bounce back to the module. Considering the travel time and the speed of the sound, one would be able to use these to calculate the distance [4].

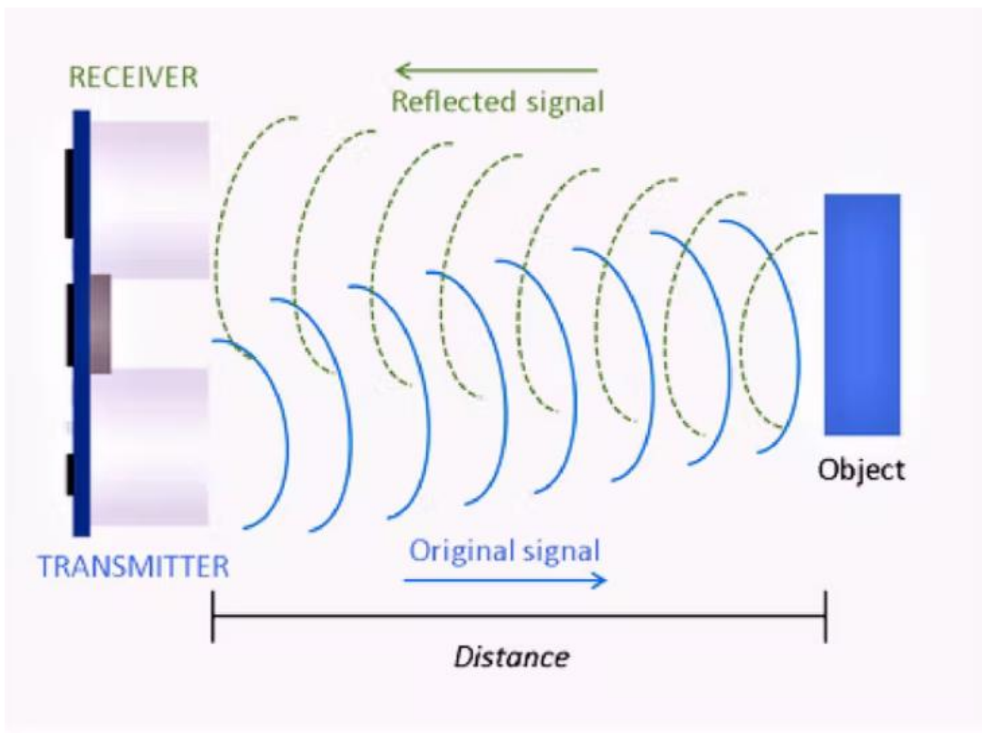


Fig 1: Ultrasonic Sensor HC-SR04 Principle

The configuration pin of HC-SR04 is VCC (1), TRIG (2), ECHO (3), and GND (4). The supply voltage of VCC is +5V and you can attach TRIG and ECHO pin to any Digital I/O in your Arduino Board [4].

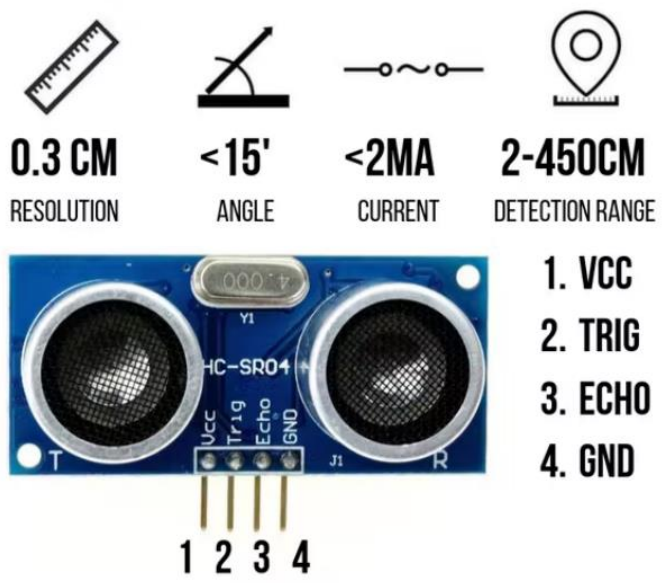


Fig 2: Ultrasonic Sensor HC-SR04 Configuration and Specification

The servo motor that positions the ultrasonic sensor is the SG90. A servo motor is a DC motor that has an internal "servomechanism", a method of controlling the position of the motor shaft with an internal feedback mechanism. Most servo motors, including the SG90 used in the kit are restricted to 180°. In order to control the servo motor, a pulse is sent to it. The width of the pulse determines the desired position of the servo. The servo motors internal controller receives the pulse and positions the motor shaft in the appropriate position [5].

When it came to the Line-tracking Module, LEDs and light sensors were used for the robot car. The light from the LEDs is beamed down at the floor and then reflected back. A dark surface would reflect very little light, whilst a light surface will reflect the light fairly efficiently. The sensors are tuned to a threshold such that the dark surface can be distinguished from a light one [5].

OVERVIEW OF IMPLEMENTATION & SOLUTION

Line Detection

When developing our algorithm, we opted to start from the end, determining when we would arrive at the start location. For this we looked at the course layout, and determined that the only 2 viable options would be either the use of some sort of timer or, the better approach that we chose, is using the light sensor available on the robot and counting the number of black lines it crossed. Hence, we implemented a counter function in our algorithm, that while navigating the course would count the number of black lines that it would have crossed and when reaching the fifth black line, terminate the process as the goal would have been reached. To implement our concept accurately, we first took measurements in the same environment at the course to find the calibration of your light sensors as variance is inevitable. After, we set a threshold for our light sensors, and when a majority of the sensors read values above the threshold the likelihood is that a line is being crossed. To eliminate double readings we are making use of a bool flag value that once set true, would require a non-black reading to reset to false.

Starting Position

Next we approach the start position problem. For this we used the PID Controller (GY-521 Module), from which we used one axis - the yaw angle. This angle helped us determine the calibration we needed to adjust our robot position accurately to start the course, which in this case we needed perpendicular to the start line. Given that in the assignment briefing we knew that the robot would start at an angle centered on the start line. We determined we could take advantage of the PID control and the light sensor. At the start the robot will rotate in one direction continuously to find the black tape using the light sensors. From there our robot will drive forward to aid in horizontal calibration, and then, given we have determined the robot is parallel to the start line, we rotate using the yaw to rotate the robot 90 degrees making him perpendicular to the course lines. The process for this is we initialize the PID Controller after the car finds the black tape, so the yaw angle will be 0, and then we just rotate the car until the yaw is 90 degrees.

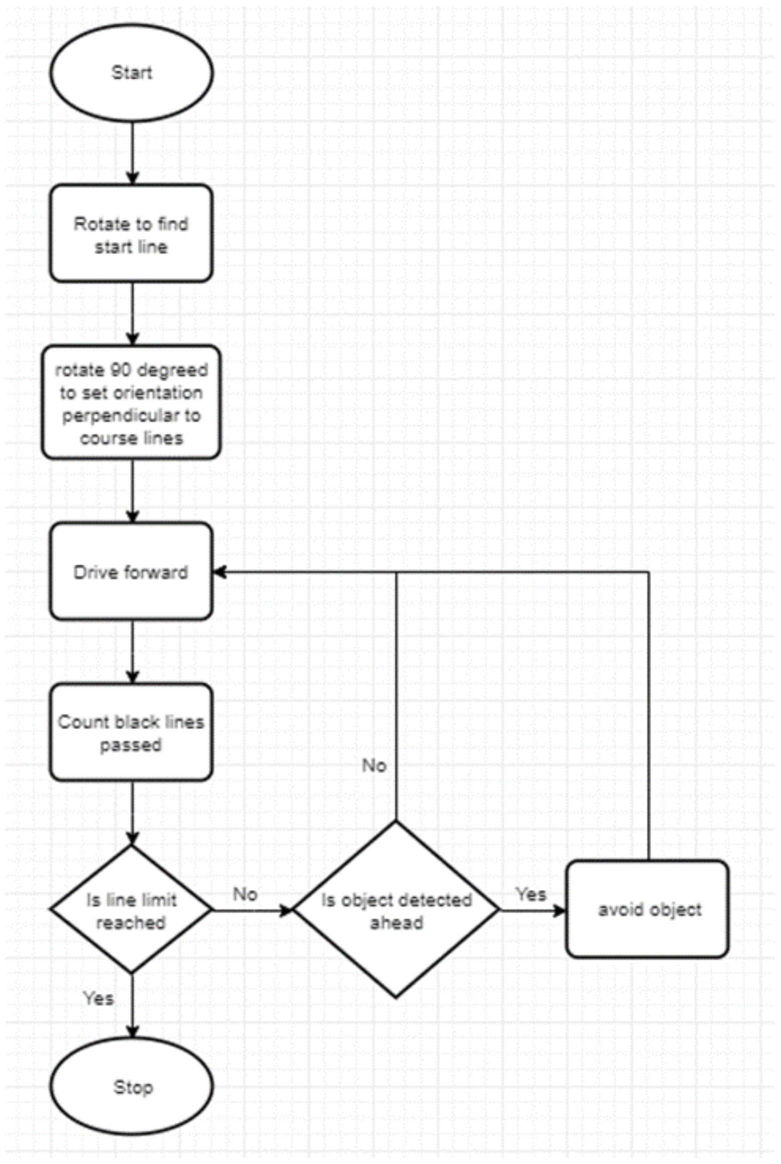
Object Avoidance

For the final and main problem that needed to be overcome for this project, we further split it up into 3 parts. The task of object avoidance and recalibration of position was split into, detection of the object, calculating the turn angle, and then finally determining how to adjust our position back on course. Object detection was our first priority, using the robot's ultrasonic sensor we could determine if an object was directly in front of the robot within a narrow angle. To make our lookahead more ideal, we took advantage of the servo controller. Using the servo we would take the readings over the full 180 degrees range in front of our robot upon an object being sensed directly ahead. Once this process was working, we could move on to determining the angle our robot needed to turn. Our first intuition was to rotate until the ultrasonic sensor no longer detected the object, however due to the very limited field of view this was not viable. Instead we took advantage of some simple calculations using the speed the robot is turning at, the angle at which the object was detected and the time it would take to turn a certain angle. Through this we would use the angle of object detection to rotate for a certain amount of milliseconds clearing the object. Then we would drive forward and readjust our course using the previous time of rotation.

DESCRIPTION OF THE CODE

Going over our project from a more technical aspect, a brief explanation of the code and some included snippets of particular value are discussed below.

Flowchart



Code Overview

Going over our code implementation in a more detailed manner, we created a variety of different functions along with using inbuilt header files from elegoo. The structure of our projects works by in the setup function, initializing all the variables and pins, calibrating the sensors and adjusting the servo angle to have the ultrasonic sensor facing forward.

From the setup function we also call the set_straight function, this function takes care of setting the robot position as required.

After the setup function, the program moves to the loop function (in the flowchart this would be form the drive forward process block onwards). Within the loop function we begin by taking the light sensor values, we check the line count, and if still below the set limit, we set the robot moving forward until an obstacle is found ahead. Once an obstacle is located, the avoid_obstacle function is called. This function further calls functions to take various ultrasonic position readings, calculates the required turn angle, moves around the object and readjusts the position.

TESTING & RESULTS

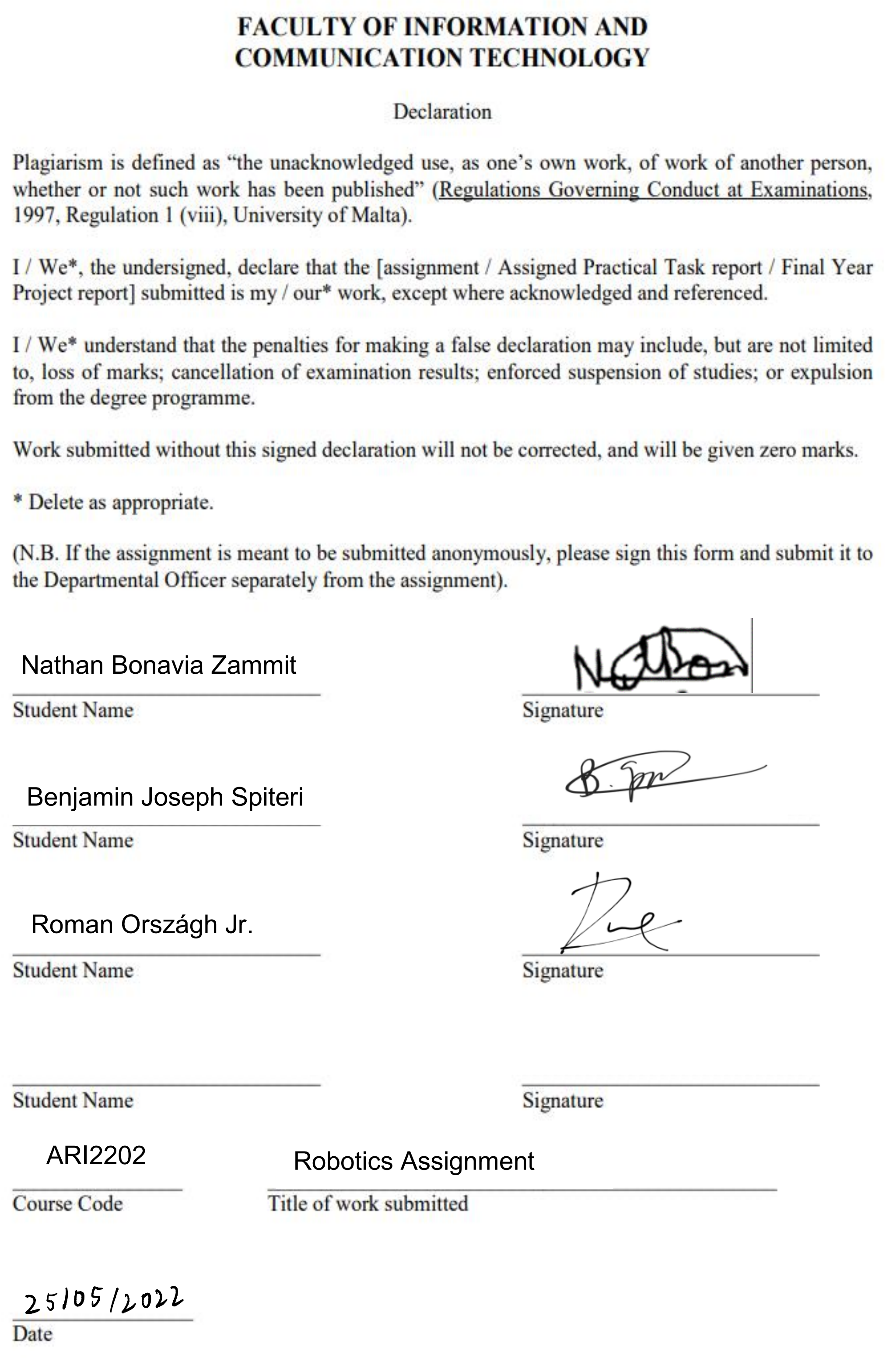
Given our approach to completing this project, the testing process was conducted simultaneously with the development of our algorithms. The first round of testing was done on the individual sensors using the code provided with the robot kit, this was done so we can find the sensitivities and thresholds of the sensors in order to familiarize ourselves with them before developing our own program.

At the start of this process the testing did not begin on a good note as we could not manage the algorithm for PID Controller. We kept finding mistakes and inaccuracies between the general code provided and our specific unit, which were making refactoring the official code inefficient and unproductive. For the light sensor the process was much more hassle free, with the required information being able to be collected quickly. For the ultrasonic sensor, some unexpected readings were found, such as the really narrow field of view, but no issues were encountered.

When combining the different portions of the algorithm, the testing approach taken was to build a model of the testing environment and constantly have the robot run the course after each modification or alteration to the algorithm. While this led to some misidentification of problems, the inbuilt log on the arduino ide helped us in correctly finding our mistakes.

Overall from the testing we conducted on the final algorithm, we identified some limitations of our algorithm, that while not crucial, could lead to mistakes being done during the course. The main issue we identified are, the suspension of the line counting module while the robot is changing angle to avoid an object.

PLAGIARISM FORM



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own work, of work of another person, whether or not such work has been published" (Regulations Governing Conduct at Examinations, 1997, Regulation 1 (viii), University of Malta).

I / We*, the undersigned, declare that the [assignment / Assigned Practical Task report / Final Year Project report] submitted is my / our* work, except where acknowledged and referenced.

I / We* understand that the penalties for making a false declaration may include, but are not limited to, loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

* Delete as appropriate.

(N.B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Nathan Bonavia Zammit	[illegible]
Student Name	Signature
Benjamin Joseph Spiteri	[illegible]
Student Name	Signature
Roman Országh Jr.	[illegible]
Student Name	Signature
Student Name	Signature

ARI2202	Robotics Assignment
Course Code	Title of work submitted

25/05/2022
Date

DISTRIBUTION OF WORK

Name	Code	Documentation	Average	Signature
Nathan Bonavia Zammit	100%	100%	100%	[illegible]
Roman Országh Jr.	100%	100%	100%	[illegible]
Benjamin Joseph Spiteri	100%	100%	100%	[illegible]

REFERENCES

[1] - A. Ghoshal, “Elegoo vs Arduino”, educvs.com, https://www.educba.com/elegoo-vs-arduino/ (4/05/22)

[2] - Arduino Forum, https://forum.arduino.cc/t/what-is-an-i-o-expansion-board/106832

[3]- M. Schoeffler, “Tutorial: How to use the GY-521 module (MPU-6050 breakout board) with the Arduino Uno”, mshoeffler.com, https://mschoeffler.com/2017/10/05/tutorial-how-to-use-the-gy-521-module-mpu-6050-breakout-board-with-the-arduino-uno/ (7/05/22)

[4]- Arduino Project Hub, https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6

[5] - https://dronebotworkshop.com/elegoo-robot-car-part-3/(4/05/22)