RoboCupJunior Rescue Line 2023

Team Description Paper

Zonda IITA Salta

Joaquín Rodríguez – Joaquín Argañaraz

Abstract – We are taking part in the Rescue Line category, where the robot needs to follow a line, avoiding any obstacles that may appear and turning left or right according to green tapes on the floor, until it reaches a rescue area. In this part, the robot will have to identify and classify alive and dead victims and rescue them, leaving them in a red or green corner located in the square. Our robot is able to do this efficiently and in record-breaking time (compared to previous versions of our project). This year, we took things way further and decided to use a Lego Spike kit, though we developed a communication protocol between the Spike and an Arduino Uno. This way, we couldn't use Lego pieces only, and our robot features servos, 3D pieces and Time of Flight sensors, something never seen before.

I. INTRODUCTION

A. Team

Our team is rather small: it consists of only two people: Joaquín Rodriguez and Joaquín Argañaraz. We were both born in Salta and have been taking robotics classes since 2018 and 2017 respectively. Joaquín A. has taken part in RoboCup Junior 2019, in Australia, and has vast experience in educative robotics, as he's been improving his skills and techniques over the years. He took part in Roboliga Nacional 2017 – Rescue Line, in Roboliga Nacional 2018 – Rescue Line, which took place in San Luis, Argentina, resulting in the first place, in Roboliga Nacional 2019 and in Roboliga Nacional 2022 – Rescue Line, with Zonda Team, which resulted in the first place and was able to represent Argentina in RoboCup Junior 2023. Joaquín Rodríguez, on the other hand, took part in Roboliga Nacional 2021 – Rescue Simulation, in RoboCup 2022 – Rescue Simulation, resulting in the 6th place and obtaining a Prize for the Best Teamwork. He also took part in Roboliga Nacional 2022, with Zonda Team, resulting in the first place, which lets the team represent our country in this competition.

II. PROJECT PLANNING

A. Overall Project Plan

Our aim for the competition is to improve the robot we've been building since September 2022. That original version of our project was able to navigate the whole scenario smoothly and with no complications at all. Nonetheless, we wanted to take things to the next level. We believe there was much more job to get done, so that is what we did. Our plan was to keep using Lego Spike for the main structure of the robot, yet we figured that some of its features didn't live up to our expectations; we wanted a more precise distance sensor and a more professional claw. Keeping those things in mind, we realized that, even though Spike could perform such tasks, a dedicated tech pieces would do a better job. Therefore, we 3D printed a claw that would fit

perfectly in our robot and bought Time of Flight sensors to replace the original Lego Spike distance sensor. To control all this gear, we initially intended to use a Teensy 4.1, which was impossible due to time issues. Therefore, we used an Arduino Nano instead.

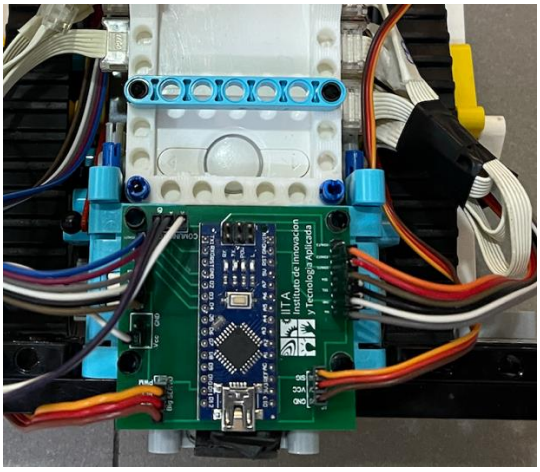


Fig.1: Arduino Nano fixed to the board.

Our plan was to use Lego Spike and Arduino Nano all together, integrated with each other, communicating both boards with a communication protocol of our own. Using a Lego minicomputer along with another board is something that none of us has ever seen before and we're sure that, this could open a whole new world of possible things for our robot.

Apart from integrating all those improvements to the project, there were some issues in the PID controller the robot uses to follow the line, so it was important to fix that as soon as possible.

Even though we are conscious the task the robot has to go over is rather complex and requires time and preparation, when we decided how to face this exercise, we chose to be ambitious. Time is not on our side and, in fact, is becoming a problem there is not enough time to implement everything we wanted on our robot. To begin with, we are choosing the first victims corner option, which is 0.5 cm tall instead of the second option, with 6cm of height. The original plan was to choose the second one, and the hardware selection was made according to that initial decision. Nonetheless, the robot would have needed a new color sensor to detect the wall and identify if it was red or green, which, even though was a good idea, couldn't be implemented, as there are no free ports and there was literally no space for a new sensor.

Another important thing regarding our strategy is the rescue kit the robot has to pick up from the side of the line. We are going to have it put next to the line instead of carrying from the beginning of the task. Despite still not working properly and having very little time to fix it, our robot contains the necessary hardware to identify the cube and its claw is designed in a way that allows it to pick up the ball. Carrying the rescue kit from the beginning sounds logic and might be, in fact, a much cleverer idea. However, many pieces of the robot were designed specifically to be able to solve these kinds of situations and we want to show that.

B. Integration Plan

Our robot could be organized in various aspects: electronics, software, and hardware. The integration between all these areas is seamless and is barely noticeable, as we have organized everything in such a way that by pressing just a single button and connecting a single cable, it's possible to get everything operating together.

We are working with a Lego Spike and an Arduino Nano, which means we have to establish some sort of connection between the two. To do so, after some time researching, we found out that it is possible to connect a Lego device to an Arduino board using a distance or

color sensor, being the Lego Spike the host of the connection and the Arduino the slave, which was perfect for what we had planned. To establish such connection, we had to tear down a Lego Spike Distance Sensor and connect one side of the cable to the Arduino. Then we connected the Distance Sensor cable to the Spike. This way, the Arduino is able to send and receive messages to and from the Spike. Even though it is possible for the buffer to fill up and cause a lag between what the Spike is sending and what the Arduino is reading, this did not happen due to the implementation of a function that makes each board send messages only once, and not many times.

III. HARDWARE

A. Mechanical Design and Manufacturing

Lego Spike:

Our robot is built on a chassis made by Lego Spike pieces. From the caterpillars to the supports for the Spike Hub and the Arduino Nano, we have used the Lego Spike normal pieces that come in the box when it is first purchased, as it simplifies the building process, while still providing a reliable and solid base on which the other parts of the robot can be held.

3D Pieces

Apart from using the Lego Spike pieces that came with the robotics kit, we designed and printed many 3D pieces in our attempt to improve the claw mechanism.

Firstly, this robot features a claw (Fig. 2) that allows it to pick the victims and the rescue kit with no problems at all, thanks to its big lip at the bottom. This claw was not made by us; we found the design online (ap. 1) and decided to use it. however, the claw on the robot isn't the same as the one on the Internet. In fact, it looks like a completely different piece, as we've modified it in order to make it more functional to our needs. The changes include: 1) a lip that helps it pick up the rescue kit, 2) an increase in the height of the upper part of the claw (the claw wasn't long enough, and it didn't reach he ground, which made the process of picking the ball and the cube up a lot more complicated. Therefore, even though we could've made the claw a little bit taller, we decided to add some inches in the upper part, as it was the one that would mean the least increase in weight) and 3) reinforcements of certain pieces that are likely to break, like structural slim sticks and the part of the claw where we screw it to the main structure, as it was too tall, but too thin.

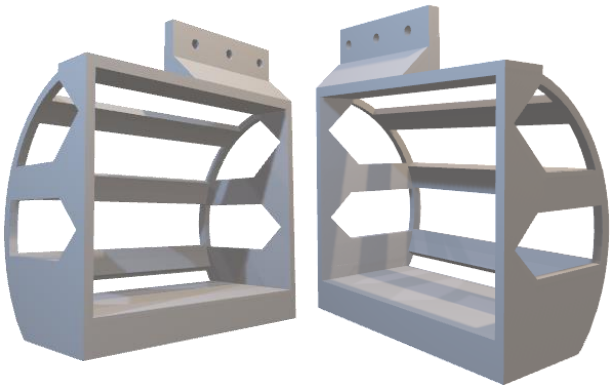


Fig. 2: The completely redesigned claw the robot features. It has a big lip at the bottom. It is 9 mm height, which, according to our measurements, is the biggest the down lip can be in order to be able to pick the ball up smoothly while still being able to lift the rescue kit: the servo makes pressure, and the down lips provide with enough surface for the cube to be lifted while being kept tight between the two claws. When the claw is lifting, the rescue kit eventually falls into the claws.

The claw is mounted on a 3D-designed support that does not only support the claw, but also the servo that operates the whole lifting and lowering mechanism. This piece was completely designed by us, and, even though exact measurements were taken, many

versions had to be made in order to find the perfect size for the servo box.

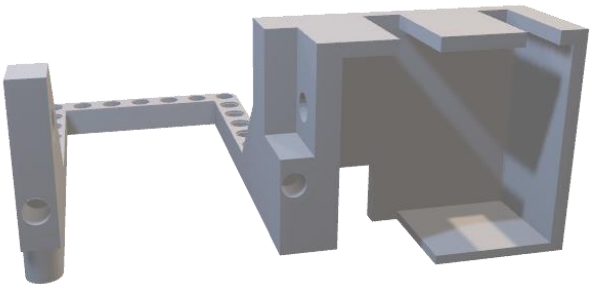


Fig. 3: Standard Servo Support holds the standard servo, which is responsible of controlling the lifting and lowering of the claw. Hence, the piece has space for a horizontal stick that will receive the rotation of the servo and help with the movement of the claw.

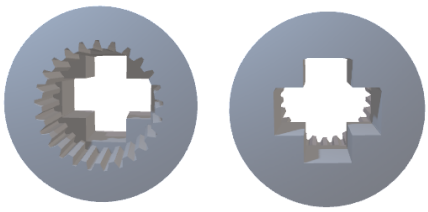


Fig. 4: Adaptor that connects the horizontal stick to the servo.

We needed a piece to hold the servo that opens and closes the claw, as well as the claw itself. This piece needed to be strong, yet not too heavy, as it had to be able to be lifted and lowered by the standard servo. Keeping this in mind, we designed a piece that would have these requirements. This piece is rather long, helping the claw be a little separated from the robot's structure when lowered, and contains a small box that holds the servo, as well as interlocking joints to place the gears that help with the claw's movement.

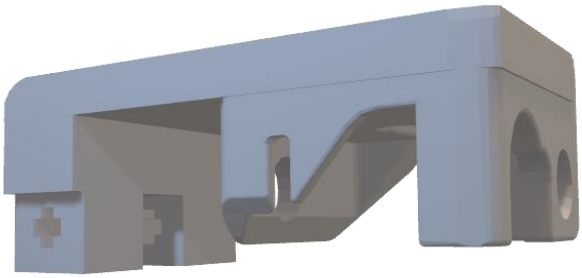


Fig. 5: Small Servo Support holds the small servo in a personalized servo box that keeps the small digital servo tight and ready to operate the claws. It also has little reliefs at the back so that the gears can fit in perfectly.

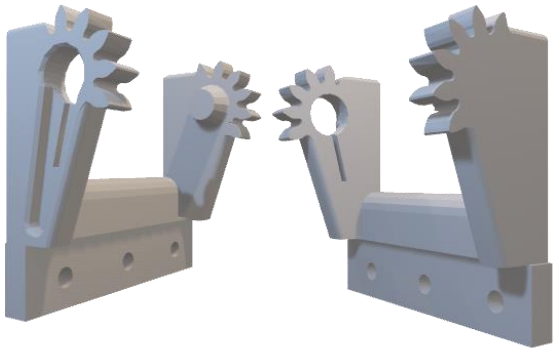


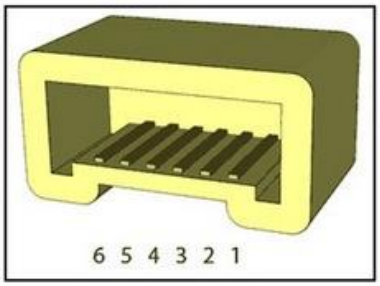
Fig. 6: The claw gear mechanism that allow it to open and close.

B. Electronic Design and Manufacturing

We are using two main types of sensors: three Lego Color Sensors and four Time of Flight sensors. Despite being used simultaneously, they are not controlled by the same board. The Lego Spike controls the Color Sensors and the Gyroscope and the Arduino Nano controls the ToFs sensors. In order for these two boards to work together, we needed to establish an Universal Asynchronous Receiver-Transmitter (UART) connection between them, which is possible after connecting Spike TX to Arduino RX and vice versa.

Lego Spike

Spike is the main board we are using and the host of the connection Spike-Arduino. The Lego Spike receives and sends information to the Arduino using one of its sensor cables. We had to cut a Distance Sensor cable in order to have access to the pins.



Pin	Label	Function
1	M1	Motor power lead 1 (PWM controlled)
2	M2	Motor power lead 2 (PWM controlled)
3	GND	Ground (0V)
4	VCC	Power for device electronics (3.3V)
5	ID1	Analog identification line 1 / Serial data (hub -> device)
6	ID2	Analog identification line 2 / Serial data (device -> hub)

Fig. 7: Lego cable pinout

The pins include two PWM pins, two alimentation (VCC – GND) pins and two communication pins. The pins we are using for communication are RX, TX and GND. The pins in the cable are connected to Arduino RX, TX and GND (switching RX in the Lego cable to TX in the Arduino and RX in the Arduino to TX in the Lego cable). By doing this, we are able to send and receive information from and to the Arduino and Spike without clutter, as we've implemented a message request – answer system that we will explain in detail later.

Also, aiming to create a robot that isn't completely full of disorganized cables and non-safe connections, we designed a board to hold the Arduino Nano and facilitate the connection with the Lego cable.

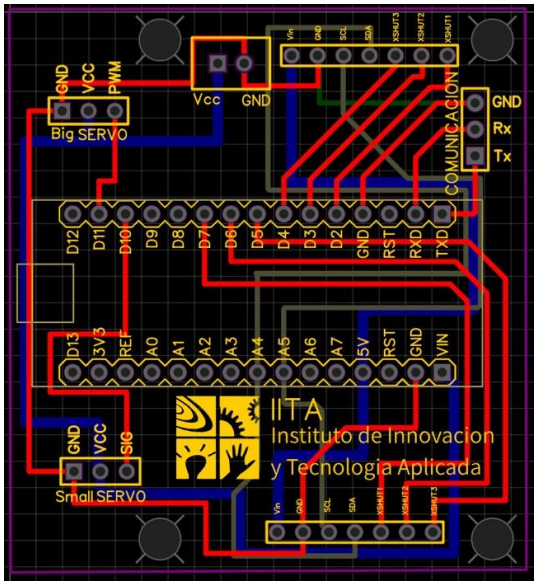


Fig. 8: The board that holds the Arduino in the middle and receives the cable connections on the sides.

Regarding sensors, we are using three Lego color sensors, which are symmetrically arranged under the robot.

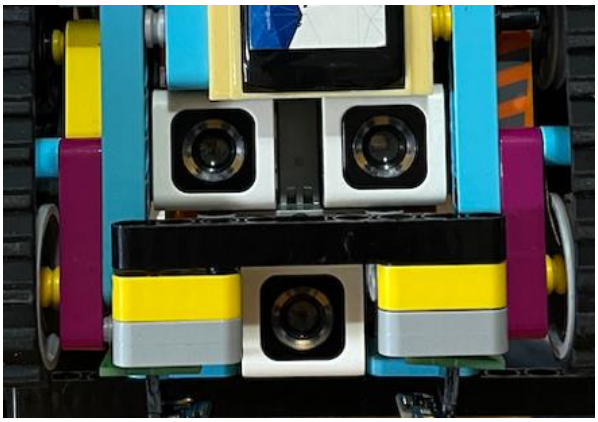


Fig. 9: Color sensors under the robot.

To detect things, we implemented a new technology for us, which is Time of Flight sensors (ToFs). They are fixed to a personalized board located in the front part of our robot.

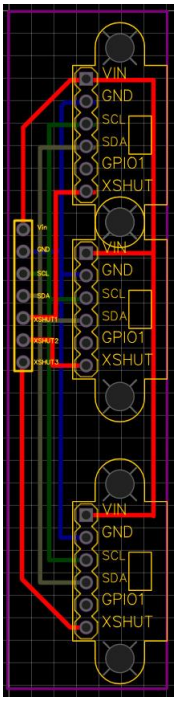


Fig. 10: Board that holds the Tine of Flight sensors. There are three spots for a ToF sensor, yet we are only using two and planning to add a third one in the future.

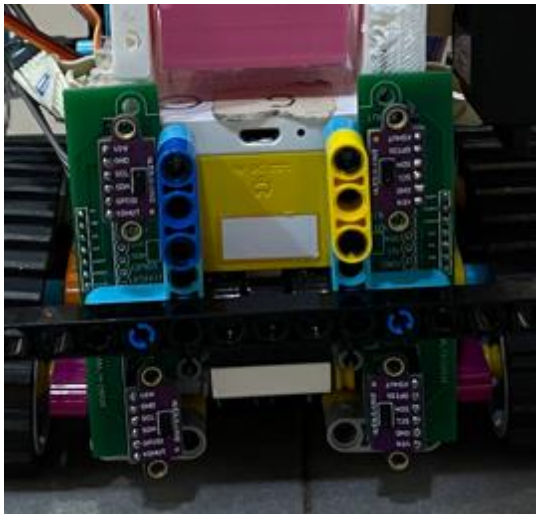


Fig. 11: Time of Flight boards located at the front of our robot.

IV. SOFTWARE

A. General Software Architecture

We have coded our robot in two different languages: we used Python for the Lego Spike and C++ for the Arduino Nano.

As we mentioned earlier, our robot consists of two different boards that need to communicate wirily. Hence, codes should be able to read the buffers, so as to know if there is anything it needs to know, and also send messages to the other board using a specific communication protocol. This communication protocol is explained in detail in the diagram below:

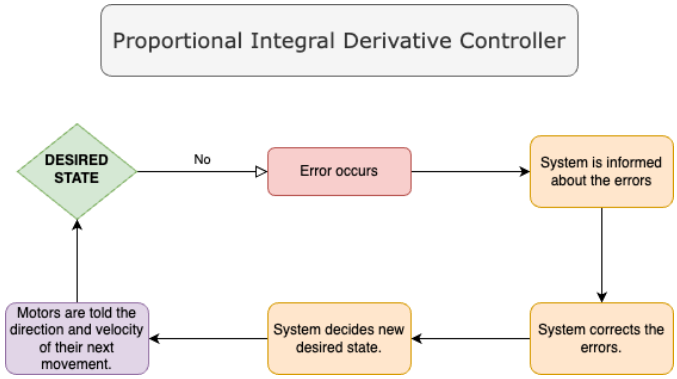


Diagram 1: Communication Protocol

Python Code

Our Python code is used in the Lego Spike Hub, which controls the motors and receives and process information from three Color Sensors and is responsible for the whole line-following part of the exercise, except for the obstacle, that needs the Time-of-Flight sensors. The algorithm used to follow the line is the Proportional-Integral-Derivative Controller (PID Controller) (Appendix, 1)

C++ Code

The C++ code operates the servo and receives and process the Time-of-Flight data. This code reads the state the Lego sends via UART messages and starts sending the Spike letter, depending on the state, according to what the sensors are seeing, (STATE = 'line-follower' → a: 'go', f: 'stop') (STATE = 'obstacle' → N: 'nule', R: 'super close', C: 'close', M: 'very far', L: 'far') (STATE = 'rescue' → U: 'nule', N: 'near', F: 'far'). It also reads if the Spike has sent an order and,

depending on the message the Lego Hub sends, lifts, lowers, opens, or closes the claw.

B. Innovative Solutions

UART Communication Protocol Arduino - Spike

A communication protocol between two completely different microcontrollers was implemented. Keeping in mind that it was necessary to connect those boards seamlessly, establishing a clean connection that would not clutter neither the Arduino nor the Spike buffers, an UART communication protocol was developed. As it is known, UART connections use serial ports to receive and send information. Therefore, knowing that both boards had to be capable of sending and receiving info, so as to maintain a human-like conversion, though not only using complete phrases, sentences and paragraphs, but uppercase and lowercase letters in order to transmit instructions or states. As it has been stated before, the two boards are connecting via serial ports using RX, TX and GND pins. The communication protocol is possible because of a Lego Distance Sensor cable that has been cut, so that the connector that goes right in the Lego Spike Hub could be on one side of the cable (allowing it to connect to the hub like a normal sensor) and the pins could be ready to be plugged in the Arduino Nano. This way, the Arduino functions almost like a normal sensor (or at least that is how the Spike understands the information it sends), except for the fact that, of course, must be coded in a different language and that it is not, by all means, a normal sensor. It could be considered a 'mega-Lego-sensor', as it is in charge of many important tasks: operating everything related to the claw (opening, closing, lowering and lifting it) and checking the robot does not crash into obstacles or walls by controlling the data collected by Time-of-Flight sensors. This communication protocol is also effective due to the highly-debugged code, which has a function that prevents buffers from cluttering with messages.

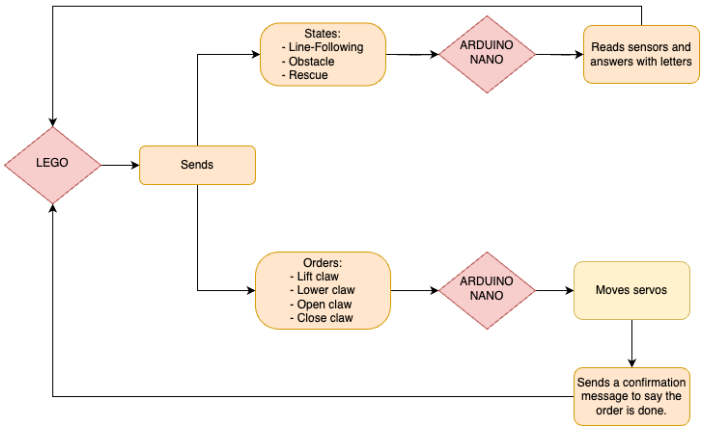
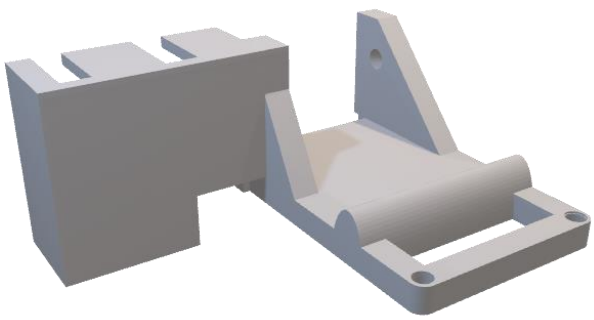
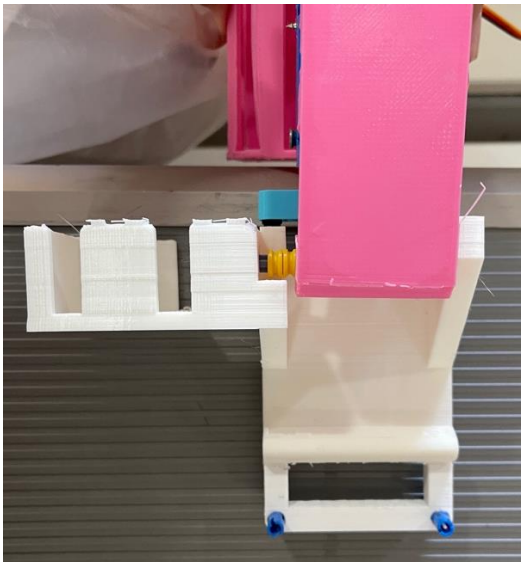


Diagram 2: Communication Protocol.

Integration

In order to make this integration even better, the Arduino is not working alone; many Time-Of-Flight sensors depend on it, as well as two servos located above and at the side of the claw. For us to integrate those non-Lego pieces into the design and aesthetics of the main structure, it was necessary to adapt some pieces. Therefore, some crucial parts of the robot, such as the claw mount, are fixed to the Lego Brick because they have been 3D-Printed, and the design met Lego measurements and sizes.





Figs. 12 & 13: The claw mount that is fixed to the Lego Brick

V. PERFORMANCE EVALUATION

Our robot is able to follow the line slowly, though precisely and never steps out of line. Obstacles, ramps, seesaws, speedbumps are solved. It is also able to deposit the Rescue Kit in the rescue area in a decent way. Even though it sometimes struggles with ramps, it is sure that the robot will overcome it. The most complicated part is, for sure, the rescue area, which is still under testing. Zonda IITA Salta robot is able to identify victims, rescue and deposit them in a corner with no complications. However, we did not add anything so as for the robot to be able to know whether it is an alive or a dead victim. Nonetheless, it recognizes red and green corners and it chooses where to deposit one victim or the other using a `random()` function.

VI. CONCLUSION

To sum up everything that has been stated so far, Zonda IITA Salta robot is able to follow the line seamlessly and with no complications, except for the ramp, sometimes. This robot is innovating by including a Lego – Arduino communication protocol which allows both boards to work together by sending and receiving messages that command states or orders. Also, we are implementing a new technology for us, which are Time of Flight sensors. In order to integrate them to the general structure of the robot, we designed boards, which are not exclusive for ToFs. There is also a custom board for the Arduino Nano. These boards help with cable management, as there are not disorganized cables around the robot, and also with safety, due to the fact that these cables connected to the boards will not fall off, causing problems.

This is, overall, a good, reliable robot that might be a little slow sometimes, but it is surely going to perform decently.

REFERENCES

- https://scholar.google.com/scholar_url?url=https://ieeexplore.ieee.org/abstract/document/1372551/&hl=es&sa=T&oi=gsb&ct=res&cd=0&d=3288845489934275248&ei=AWOPZL_3EYrGsQK0rZDoCg&scisig=AGlGAw9ssZtQ0dxlGTAoKT7h0kUD
- https://en.wikipedia.org/wiki/PID_controller
- http://ethesis.nitrkl.ac.in/5405/1/109EI0326.pdf
- https://www.ni.com/es-cr/shop/labview/pid-theory-explained.html
- https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html
- https://www.circuitbasics.com/basics-uart-communication/
- https://www.geeksforgeeks.org/universal-asynchronous-receiver-transmitter-uart-protocol/