About the BluBLOC Arduino UNO Powered Self Driving Car

The Arduino powered self driving car which the BluBLOC team representatives designed for the PRO Future Engineers “Self Driving Car” competition has several modules and Arduino UNO components designed to replicate the fundamentals of autonomous driving.

The Arduino UNO board connects with different sensors and motors on the car. These parts work together to help the vehicle move forward, backward, steer, and detect surroundings/obstacles. The main components we used include a servo motor (SG90 servo) for steering, a TT DC Gearbox Motor for driving, three ultrasonic sensors (HC-SRO4) for obstacle detection, and a color sensor (TCS3200) for recognizing the colors of the pieces on the board. Lastly, a Motor Driver Shield (Dual Channel DC Motor Driver) connected directly onto the Arduino was used to send data for the movement of the DC motors.

How it Works

The team designed their own chassis for this project, attaching the main circuit on a breadboard with its wires in a snug external holder for stability and ease of access. Above was the Arduino UNO Module, the Motor Driver Shield modifier, and the two “18650 Li-ion batteries” with a barrel jack (7.4 V) to power the entire mechanism. Underneath the upper chassis was where a servo motor and the gearmotor modules for the steering and movement of the car were screwed into place. Around the frame, the team also secured the four distance sensors in all directions, while the color sensor was placed in front. This entire schematic was constructed from scratch by the members of the team, with the base and structural components being made out of foam board and 3D printed materials, while the mechanical parts such as the steering and wheel holders were 3D printed. LEGO Technic pieces were used in the steering mechanism, along with a dowel (0.7cm) on the drive wheel system.

First, we used the servo motor connected to analog pin A0. This motor helps steer the wheels left and right. The code tells the servo, which acts as a lever, to move to a certain angle to move a “rack” left or right where LEGO Technic parts are connected to a series of mechanisms that turn hinge-like parts connected to another “rack” parallel to the first. The hinge-like part, which acts as the steering arm where the wheel is connected, is free to rotate through this system. As the rack moves, the hinge twists slightly, causing the front wheels which are mounted on the hinge to turn left or right. This setup, commonly referred to as a parallel steering system, allows the servo to indirectly steer the wheels by converting its rotational motion into a side to side motion, and then angular turning.

The DC motor, controlled through the Out A and Out B pins on the arduino shield, was responsible for making the car go forward or backward. The code adjusts the motor speed and directions which are depending on the sensor reading around the vehicle. A simple gearbox system was put under the chassis of the car, where a small gear is directly connected to the motor. A bigger gear is meshed to this small gear to add more torque. The bigger gear is connected to an axle which powers the drive wheels at the back.

The ultrasonic sensors help the car know how close it is to walls or obstacles. The car has three sensors: one in the front, one on the left side, and one on the right side. The echo pins of the left, right, and front are connected to pin 3, 4, and 7 respectively. Their trig pins are connected to pins A1, 5, and 6 in the same order. These sensors work by sending out sound waves and measuring how long it takes for them to bounce back. From this, Arduino can calculate the distance of the nearest object. All sensors activate when something is around 23 cm away. In the code, if the front sensor detects something close, the car moves backward. If something is close to the right or left side, the car steers in the opposite direction to avoid it.

The color sensor was stationed in front of the car in a way that it will recognize the colors of the map without obstruction of the main chassis and proper distancing from the obstacles. As the rules describe, this will be the main recognition method in the obstacle round, where the car will turn left when the obstacle is green, and right if red. The team also plans to apply the color recognition to the magenta walls that will guide the car to its parallel parking sequence. Upon the completion of one lap, when the car recognizes the magenta walls, it will repeat its driving sequence until it completes three laps. Only when it completes the three laps would it begin the parallel parking sequence.

The code is uploaded into the Arduino Uno through an Arduino Uno USB cable connecting the system to the team’s laptop. Here, the team could verify and upload the code.

Movement in the “Driving Open” Challenge

The Driving Open Challenge presents a space closed within a square area, and in the middle of that space is another smaller square area, making a closed looping track. The teams have to ensure that their robot would be able to do one lap on the track, and that the robot would be able to stop on its original starting point.

In order to pass the first challenge, the team designed the robot so it can hug the wall or base its movement around it. To be able to move around the center square, the robot will utilize its left and right ultrasonic sensor to move in different directions, as if one of the sensors detects one of the sides of the center square the robot will turn and move in the opposite direction of where the sensor is located. In case if the robot turns too much to the point the front of it will bump into the sides of the center square, the front sensor will detect the wall and will make the robot steer away from the obstacle. If the robot steers too far towards the outer square, the sensor corresponding to its direction will turn it the opposite way, in order for the robot to not bump onto the wall. After a set delay, the robot will stop on the same section it started in.

Movement in the “Obstacle” Challenge

The Obstacle Challenge presents a new level of difficulty because of the addition of the traffic signs and the parking lot. The vehicle is supposed to take note of the direction it should go in response to a traffic light, perform three laps, and be able to parallel park at the end of those three laps.

In order to perform such a challenge, the team designed the robot to respond to the additional pieces presented in the playing field. The robot would perform as it would in the first challenge until it meets a colored block, or a “traffic light.” To be able to dodge the traffic lights, the robot uses its color sensor to detect the color of the obstacle. If it is the color green, it’ll respond by steering to the left, while if it is the color red, it’ll respond by steering to the right. This steering takes priority over the obstacle detection of the ultrasonic sensors. After a fixed amount of time, the ultrasonic sensors take priority again in order to move the robot away from possibly bumping onto the walls.

While going around the course, the robot will count how many magenta walls it has seen. Every time it detects one, the robot adds a count within its code pertaining to the amount of laps. Once this count has reached three laps, it initiates its parking sequence. It will attempt to perform a parallel park by approaching the magenta wall, steering right, and moving backward. After a short delay, it will move left then backward.

Build Development

The robot underwent numerous changes in order to optimize its design. The team decided to categorize these into three prototypes, each of which were improved based on concepts the team learned on the way and the issues that were addressed.

On the first days of the project while the team was working for their robot in the 25th Philippine Robotics Olympiad, there were a few weeks dedicated to planning and consolidating the resources we had at our disposal. Considering our capabilities, resources, and own aspirations to design a car from scratch utilizing our academic knowledge, the team decided on creating a self-driving car designed using mainly 3d parts and programmed utilizing Arduino Uno compatible parts.

The first prototype was composed of a completely 3d printed chassis and body, composed of a "shelf-like" structure with the breadboard and circuitry, including ultrasonic sensors facing front, left, right, and behind the vehicle were placed inside the shelf while the Arduino Board and two daisy chained 9v batteries, connected via jumper cables for power, were stored above along with an arduino color sensor angled downwards towards the immediate forward area of the car. A simple transmission made out of two gears connected in series to a TT DC motor was fastened underneath for propulsion, while a servo motor on the front underside was screwed on to control a rack and pinion system controlled by a gear screwed onto the servo head.

This initial prototype was found to be too heavy due to the 3d printed parts, and certain high stress parts like the axles holders for the various sensors were observed to break easily. Additionally, the dual 9v batteries in series power source was found to be incredibly unstable and after further research, was found to be dangerous in application. In relation to the mechanics of the gear systems in place, the gears were found to not mesh consistently in both the rack and pinion system and the transmission system. This made it difficult for the car to even move, and efforts to get it to run in the two main challenges were difficult. Hence, these were revised for the second prototype prior to the 24th Philippine Robotics Olympiad.

The second prototype retained the propulsion system, reinforcing the fastened motor instead with a proper containment box to lock it into place using wooden sticks as a pass-through fastener. The chassis was moved around and lightened through carpentry capabilities, and the starring system was changed from a rack and pinion system into a system similar to an RC car, featuring a static beam with two swivel points on each end where the "rack" of the wheels were put on, rotating with the servo instead of being driven by the rack and pinion system of the first prototype. These changes largely increased the consistency of both the steering and transmission system. Additionally, the upper body was removed, and replaced by a sintra board platform to decrease weight. The arduino components were moved to the platform, and the battery was changed from a dual 9v battery source into two individual 9v batteries no longer connected in series, which alleviated the risk while still preserving the high voltage we believed we needed.

This second prototype was used as our main submission into the 24th Philippine Robotics Olympiad, and while it may not have completed the challenges, it did prove that the sensor and code setup we were utilizing were functioning and reacting to its surroundings, encouraging further research and engineering on the car.

Moving onwards into the Asia Pacific Open Championship for the World Robotics Olympiad, the team was able to get time to reflect on the car and the days leading up to the prior competition. Spotting points of improvement, the team moved on to create the third and final robot. Retaining the sintra board platform and 3d printed underside chassis, the third prototype of the car was simplified, removing the rear sensor entirely, finding it redundant; changing the steering system into one akin to an RC car through more durable lego technic parts, changing the transmission axle into wood and sticking the wheels using proper adhesive to ensure durability; and other improvements in the overall structural design and electronics management. These improvements also made the former dual 9v battery requirement unnecessary, allowing the team to instead utilize a safer and more consistent 3.3v lithium battery pack for power and better accessibility.

Code-wise, there was a big methodology shift in the programming of the vehicle. For both challenges, the strategy was moved from utilizing straight-styled driving to zigzag driving, riding on the inconsistencies of steering straight as observed by the arduino servo motor. This gives it its signature “zigzag” style of navigation.

With this robot, the team was incredibly satisfied, and while there are still points of improvement, the team finds that the car is a great study on the capability of a simple electronic resource like Arduino UNO in application. The time creating the robot also allowed the team to interact with various groups within and outside the school, expanded each member’s knowledge, capability, and even made connections in the fields of engineering and beyond, fulfilling not just the commitments of the members to BluBLOC, but also our commitment into becoming future engineers of the Philippines.

Conclusion

Overall, this project created by the BluBLOC varsity team of the Ateneo de Manila involves combining sensor modules with motors using code to create a basic self-driving car that can steer, avoid obstacles, and keep track of laps for a parallel parking at the end. The code is designed to run in real time and constantly read data from its surroundings. By reacting to this data, the car can move around the course on its own. This shows how programming and electronics, even done using the most basic components of Arduino, can work together to make a robot car that behaves like a smart vehicle, being able to do complex tasks such as avoiding obstacles and performing a parallel park.