CSE4360-01 Project 3 – Fall '23 Team 4

Members: Akshay Kumar Paluri , Daniel Thomas Palma, Edward Alkire, Hyun Ho Kim

Intro

Our project was Simplified Soccer. In Simplified Soccer there are two teams, one robot each. There is a defensive zone on each size demarcated by a red and blue line respectively, and a central zone between them, with a green stripe across the middle of the playing field. The rules are that each robot may not cross into the opponent's defensive zone. A goal is achieved when the ball rolls anywhere across, all the way down the field. Rolling into, but not past the defensive zone is not sufficient for a goal. When a goal is scored each team's robot is reset to their starting location at the goal line, with the losing team getting a 1s head start. Notably, this project centers around the use of an IR emitting ball, requiring robots to use IR seeker sensors to track the ball, which output direction and intensity of detected IR signals.

Initial Planning

We understood the greatest difficulty was going to come from the sensors, whether it be from poor accuracy, or errored readings based on experience in previous projects. Knowing this, an optimal strategy should rely on sensors only as necessary and might incorporate a variety of sensors to reduce reliance on any single sensor and enable fallback procedures.

Our initial analysis considered that since the goal area is multiple times the size of our robot footprint limitation (.75ft by .75ft), and that we only have one robot, even with accurate sensors it wouldn't be fruitful to focus on a defensive strategy. Therefore, we should optimize for an aggressive offensive strategy. Such a strategy could excel against opponents who did not also optimize for offense.

If our robot could adjust itself to face the ball at the start, then drive faster than the competition and beat them to the ball, it would minimize reliance on accurate sensor readings, only requiring one good reading, and eliminate the need for complexity in managing edge cases, defensive procedures, and accurate ball tracking. Such a strategy could only be assured if we made efforts to maximize the speed of our robot beyond that expected of the competition. The most immediate design choice would be to implement gear ratios to increase the ratio and therefore the speed at which the motor may drive the wheel. The robot should be as light to increase speed, it should be low to the ground and have a low center of mass to reduce risk of flipping over (or being flipped by the opponent), and the weight should be distributed over and around the driven wheels to increase traction.

Ideally the robot might incorporate four motors instead of the two normally used to increase achievable speed. If two were used per wheel in a two-wheel design, we would most likely break or significantly damage our flimsy LEGO axle. If we used one motor each in a four-wheel design, this might exceed the footprint limitations, especially if we will want a forward-facing u-shaped bumper to control the ball.

Since using four motors would not be practical, it would be difficult to make our robot decisively the fastest among what would be expected of competitors. An offensive but more adaptable strategy would be required.

Physical Design

The robot was designed to be nimble and robust, with supports along the lower foundation, with the wheels and motors configured in a high gear ratio. The robot has a simple front facing u-shaped bumper to improve control of the ball.

There are a variety of sensors equipped to enable better contextual awareness of the external environment. Most essentially, there is the IR seeker, used to detect the ball location. We chose to angle it downward to improve consistency of readings, and to better detect when the ball has been caught. The color sensor is configured low to the ground, pointing down to detect the colored tape, enabling the robot to alter its priority based on the regional indicators. Most importantly the color sensor enables the robot to respect the boundary of the opponent's defensive zone and enables our robot to switch into a defensive strategy when it detects its own defensive zone.

Other notable features are a sonic sensor to detect when the robot is driving into a wall, a gyro to help the robot to know which team's goal it's facing. Zip ties were utilized to stabilize the cables.

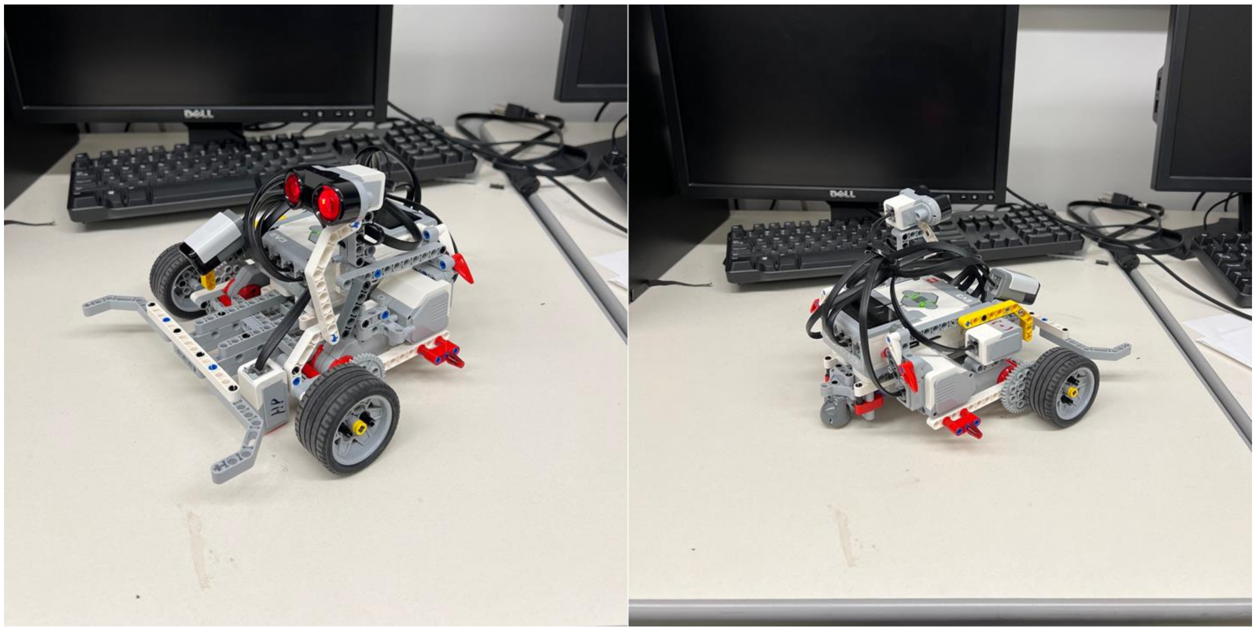
Figure 1: view of the robot

Software Design

The robot was designed to prioritize locating and moving towards an IR signal, using input from the IR seeker. When the system doesn't sense an IR signal, it switches into wandering mode that randomly rotates the robot in place, then moves forward to scan at a new location.

When the robot is first started, it keeps track of which color of line it first passes, either red or blue, designating it as the home side. After this, the robot is only allowed to cross back into the home side when it senses an IR signal within it, where a procedure is programmed to gather the ball from the home defensive zone and turn it around. Entering the home defensive zone had to be treated carefully to avoid leaving the map. Otherwise, it will remain within the middle of the map.

The way our robot follows the IR ball is by extracting the index and intensity from an array created by the IR sensor function, with the highest value in the array determining the index. The robot can then adjust the speed of the right and left wheels based on this index and the strength. We have established a criterion for determining whether the robot has recognized the ball or not based on this index value.

Wall detection is only used as a means of preventing the robot from becoming stuck. When the robot detects a wall, it will either turn in place if it detects the ball or reverse for a period away from the wall if no ball is detected, this as to help keep control of the ball. The gyroscope is used in the same way except that it has the added use of determining which direction the robot should turn based off where it is facing in respect to the defensive side.

There could be several situations where the robot doesn't detect the ball. This requires the robot to turn in random movements until it detects a ball. To implement this, we used a loop which will function when the best strength for IR sensor is found is 0, and there are several if and while statements which will cause the robot to turn in a different direction. The function returns an integer that indicates how the robot will turn. The rand integer indicates how the robot has turned previously, and if it doesn't detect the ball, it will change the number, causing another if statement in the function to trigger. This will cause another time bound movement to run in a different direction. By having time bound while statements in the function, we can stop the movements when it detects the ball, mitigating false readings or abrupt stops from stray IR detections.

Experience with The System

When we started this project, our primary focus was being first to reach the ball, as quickly as possible. Therefore, we believed that applying a gear ratio to the motors, setting the robot body low, and minimizing weight would allow us to create the fastest robot. However, due to limitations in robot components such as poor tire traction, we could not fully utilize the motor's performance. Subsequently, we shifted our attention to how the robot should react in various situations to handle the inevitability of not being first.

With four sensors—color sensor, gyroscope sensor, IR sensor, and ultrasonic sensor—our robot responded well to diverse situations. The color sensor effectively distinguished red and blue lines in most scenarios. However, the gyroscope sensor gradually deviated from our set angle as the match duration increased. The IR sensor performed better than expected, quickly detecting the ball even during wandering. Occasionally, the ultrasonic sensor struggled to accurately measure distance when the robot and the wall were diagonally aligned, leading to situations where the robot continuously pushed the ball against the wall. We resolved this issue by having the robot retreat if its position remained unchanged (as measured by the sonic sensor). Additionally, we dealt with various uncontrollable variables, such as the bumper getting stuck between walls or uneven flooring.

The most challenging aspect of the project was the defensive mode. Carefully maneuvering the robot to push the ball in the opposite direction without accidentally scoring an own goal proved more difficult than anticipated. We decided to slow down the robot as it approached the ball using the IR sensor's intensity and utilized the gyroscope sensor to determine our position. While there were not many instances where the robot needed to switch to defense mode, we foresaw the possibility and achieved some success in extracting the ball from the home area.

Overall, the robot moved well according to the game's objectives, but sensor errors, especially gyroscopic errors caused by slipping or obstacles in the field, led to some unmitigatable unexpected behaviors in certain situations.

Instructions

To run the program, load program on to robot designed similarly to ours with the sonic sensor at a 45-degree angle. Upload the file using the LEGO Mindstorms EV3 MicroPython VS Code extension. Next, run by going into file browser and selecting with center button of the EV3 device.