ELEC 299 Technical Report

Winter 2015

Group # 20

April 6th 2015

Maytha Nassor (10085970)

Johan Cornelissen (10098176)

Muhammad Usman Majeed (10086980)

# Project Overview:

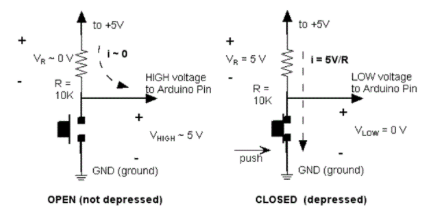
## Goals and Objectives:

The first goal of our team was to analyze the problem which was to come up with most efficient solution to drop 7 balls as fast as possible. Second was to complete the project with the scheduled timetable, which meant that team had to do everything possible to drive the project to the end and stay on time. Also to avoid incompetence in the planning of the scope so as to have a reasonable time schedule. We were forced to always keep the plan up to date, recording actual vs planned progress. The team identified any deviations from plan and fixed them quickly. Third was that the project was to be completed with a high level of quality, making sure that each requirement of the project was fulfilled. Our last goal was to keep the team happy by rewarding and recognizing them for their success, which meant assigning them with work that complements their strengths.

In terms of the objectives of the project, the team had to use the IR communication at the start of the match, which was considered as a constraint to the project design. The match length was 2 minutes, so that had to be taken into consideration before building any algorithms for the robot, so it will be possible for the robot to finish it on time. Throughout the preparation phase, the team made sure to meet the judge’s needs and make the robot as appealing as possible.

# Approach / Strategy

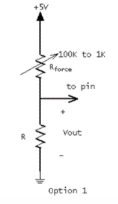
## Sensors and Circuits used:

* **Bumper sensors**

The pushbutton circuit located on the right was used to implement the bumper sensor circuit. Instead of the push button seen above, a wire goes to the bump sensor and another wire comes back from the bump sensor, which acts as a switch. 10 kilo ohms pull-up resistor is used so that when the bump sensor is pressed a LOW voltage is provided to the Arduino pin. When the sensor is not pressed a HIGH voltage is provided is provided to the Arduino pin. In the code provided, the left bumper sensor circuit was attached to digital pin 11, and the right bumper sensor circuit was attached to digital pin 12. Bumper sensors were used to detect when the robot hit the wall by each ball location and also used to determine when the robot hit the goal and is able to release the ball.

* **IR receiver**

The IR receiver was used to be able to find where each ball would be located. Each ball had an infrared transmitter above its location to allow the robot to detect its respective location. As explained in more detail below, the robot determined which location was transmitting a beacon by rotating the servo arm with the attached IR receiver. The IR receiver was attached to the +5V supply of the Arduino, the ground on the Arduino and digital pin 9.

* **Force sensor**

The circuit located on the left was implemented to allow a force sensor to be used on the gripper arm. This allowed our team to initiate gripper closing until the force sensor is activated when picking up a ball. In the circuit the +5V supply was connected to the appropriate supply on the Arduino, and the ground was also connected respectively. The “to pin” connection on the circuit above was connected to pin 10 as seen in the code. When the force sensor is not sensing a object between the grippers, a LOW voltage (seen as 0 in Arduino) is provided to pin 10. In the contrary when the sensor senses the ball in-between the gripper opening then a HIGH voltage (seen as 1 in Arduino) is provided to pin 10.

* **IR sensors for line detection**

The IR sensors used for line following was the only analog sensors used in our robot design. The reason analog is used is to provide the Arduino software and code with varying values depending on the darkness of the surface under the sensors. If digital pins were used then only a HIGH or LOW output could be provided by the sensors. The left IR sensor was attached to A0, right IR sensor attached to A2 and the middle IR sensor was respectively attached to A1. A threshold was set for each sensor separately because in some lighting conditions the sensors had slight internal differences. For the competition course the black line threshold was set to 480 for all three IR sensors.

## Software Control Systems

Figure 1: Waterfall diagram for software control systems

## Highlights of important control functions

* **Line Follow**

The line follow code is an integral part of our robot’s execution. The robot follows a black line to each ball location and also follows a black line to the goal to be able to score. The line follow code consists of three if-statements inside of a while loop. The while loop condition ensures that when all three IR sensors are detecting a value above the threshold (indicating the area is black) that the robot will stop. This condition is used to make the robot stop at the centre after scoring, retrieving the ball etc. until further instructions are given. The inner if-statements consist of turn on the right motor is the left threshold is not reached, turning on the left motor if the right threshold is not reach. Additionally a third conditional statement is used to make the robot continue driving straight if the middle IR sensors sense black tape.

* **Beacon Sensing**

The beacon sensing code is what allowed the robot to be able to determine the location of each ball and provide positional awareness. Beacon sensing consists of two functions in the code, the first function is named caseReturn. This function is responsible for calling a function named beaconFinder and turning the servo arm if the beaconFinder function is unsuccessful. The beaconFinder code checks 6 times to see if there is an IR signal received that is not 0. If a non-zero value is seen then the function returns true to the caseReturn function. The case return function remembers the location at which the servo arm was located when the beaconFinder returns a true and returns a case value based on this position. For example, if a non-zero value were found when the servo arm was pointing straight infront of the robot, then the function would tell the overall program to use case 1;

* **Ball Pickup**

Ball pickup was a simplified process because line follow was used to bring the robot to the ball. In each case specific function the robot would line follow until its bumper sensors were activated, indicating the robot had hit the wall. The robot would then backup for a short delay and lower its servo arm. Since line follow was used to keep the robot straight and the servo arm was kept at the straight position, the robot could lower its arm always perfectly without missing the ball. The code then initiated a loop that slowly closes the gripper until the force sensor was activated. Once the ball was in the gripper’s hands the robot would lift the servo, and turn around until a black line is sensed. Line follow is initiated till the robot reaches the centre and the scoring code is initiated.

* **Scoring**

The scoring code consisted of calling the line follow function and proceeding to the goal until the bump sensors are activated. At this point the robot is aware that it is in front of the goal and the gripper releases the ball so that it falls into the goal (the servo arm is already in an upper ward angle position). After releasing the gripper, the robot backs up for a small delay, turns 180 degrees until the black line and line follows. The line follow code stops at the centre of the court and then the code loops again so the beacon sensing code is initiated to restart the process.

# Performance Analysis

## Performance and Strategy

In terms of the goals that the team had initially established at the start of the project, our robot performance met our expectations. Due to the difficulty of achieving reliable hardware and software components on the robots, we prepared for unexpected errors in the robots performance. This was evident throughout testing, and competition day for many teams. Our outlined strategy relied heavily on line-following code, which we deemed a reliable method to navigate around the course. This strategy combined with relying on other sensory feedback rather than hard-coded routines was extremely successful due to the high variance in performance based on the robots battery life.

Prior to competition day, the robot was able to reliably pick up 7 balls within 2 minutes and 23 seconds. Despite the competition rules stating that the first team to obtain 7 balls under two minutes will win the match, the team weighted reliability over speed.

On competition day, we tested our robot once before an official marked run. Using the rubric below, the team scored the robots performance for each of the four matches we competed in.

Table 1: Robot performance rubric

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Functionality | Outstanding – 4 | Proficient – 3 | Poor – 2 | Unsatisfactory – 1 |
| Beacon Finding | * IR sensor was able to locate beacon on first scan | * IR sensor was able to locate beacon with two scans | * IR sensor was able to locate beacon after more than two scans | * IR sensor was unable to locate any beacons |
| Line Following | * Line following algorithm was continuously reliable for navigation | * Line following algorithm experienced problems but robot was able to re-navigate on track | * Components of line following algorithm were present however robot experienced difficulty navigating | * Line following algorithm completely failed |
| Sensor Feedback   * Touch * Force | * Sensor feedback was always triggered as expected and robot reacted accordingly | * Sensor feedback required multiple triggers before sensing | * Sensor feedback failed multiple times | * Sensor feedback was not visible |
| Servo Movement | * Servo movement was always smooth and accurate | * Servo movement was smooth but slightly off target | * Servo movement was unsynchronized but on target | * Servo movement was choppy and off target |

The results were as follows:

Table 2: Robot scores

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Match | Beacon Finding | Line Following | Sensor Feedback | Servo Movement | Total |
| 1 | 4 | 4 | 1 | 3 | **12** |
| 2 | 4 | 4 | 4 | 3 | **15** |
| 3 | 3 | 4 | 4 | 3 | **14** |
| 4 | 4 | 4 | 4 | 3 | **15** |

During the first match, the team noticed that the bump sensors were not being reactive at all, trickling the effect through the rest of our routine. As a result of this, we were only able to retrieve one ball. After returning to our lab bench, we realized that the circuitry for the bump sensors was wrong, as one wire had been plugged out and randomly placed in the wrong port. For the other three matches, the performance score received a 3/4 for servo movement as the servo that would vertically rotate to grab the ball would sometimes swing down too fast, altering the position that we anticipated it to have for grabbing the ball. However, due to the reliability of the force sensor, this was not detrimental to the performance of the robot. To our surprise, the robot was able to locate the beacons on the first scan each time, except for during match three, where one of the positions required two scans before it was recognized. As this only happened once or twice during the span of the day, the team ruled this functionality of the robot to be outstanding.

In conclusion, the performance of our robot was what we expected. The performance of the robot could have definitely been improved by one factor: the speed at which is executed tasks, which will be further discussed in the next section.

## Achievements, problems, and improvements

Looking back to the semester working up to the competition day, the team accomplished significant programming, team building and time management skills. All team members came into ELEC 299 saying that they had done minimal Arduino programming for the APSC – 200 project. We established that we wanted to use the course lab times productively to learn the course material. By working together, we would try different methods to tackle the lab modules. Using one computer monitor, we would each take turns fixing things that we thought were wrong with the code. Through many unsuccessful compilation errors, we learnt the software components that would and would not work with Arduino. When it came to hardware, the team would always verbally explain the circuit schematics before trying to build it. This way, we ensured that we all understood the hardware components of the robot as well. The success of our teamwork helped us build effective team work skills, teaching us how to constructively criticize each other’s work. When it came to time management, all three team members preferred finishing assigned modules prior to the marking lab period. This allowed us to always be one week ahead in terms of modules, giving us more time for the difficult ones. The reward of this has taught us that time management skills are vital for success. In conclusion, this course proved successful in teaching us the fundamentals of Arduino programming.

When dealing with electrical hardware and software, running into anticipated and unexpected problems is inevitable. When it came to hardware, for the touch sensor module, the team blew approximately 6 fuses before realizing that our built circuit was wrong. We had short circuited in parallel with a resistor, so the circuit was not even seeing the resistance from that resistor. This brought our attention to the tedious nature of building a circuit carefully, component by component. As previously mentioned, on competition day, one of the wires in our touch sensor circuit was plugged out, teaching us to make sure that wires are perhaps taped down once correct so that they do not become unplugged. In terms of software, the code we wrote was reliable every run, however there were a few dependencies that the robot had on hard-coded delays. For example, when completing a 90 degree turn, the robot would initially have a delay of 200 mseconds to ensure the middle sensor is on white, and then it would continue turning until the middle sensor saw black again. With varying battery levels, we noticed that the delay would be too short or long, risking the behavior of the robot. Another reoccurring problem experienced with the structure of Arduino programming was the use of the void Loop(). Many times the team would only want something to be executed once, but would find it hard integrating it into the loop and finding a condition that would halt the program. With more experience, we realized that the use of external functions proved helpful with splitting up the functionality of the program, with different conditions lengthening the loop function so that it didn’t continuously loop quickly.

In terms of functionality and strategy of our robot, there are many changes we could have made to enhance its performance. There were two main changes that would have had a significant contribution to our success. Firstly, after grabbing the ball, the robot would make a 180 degree turn, line follow back to the starting position before making the appropriate turn to the goal. To shorten the routine time, it would be faster to take the hypotenuse route to the goal instead of taking the two shorter legs. The speed of the line follow code would have meant optimizing the conditions for the IR sensor. Secondly, the team could have used Bluetooth to tell the robot the ball locations, which would exclude all the time taken to search for the beacons. Due to the Bluetooth module giving us trouble, we deemed that it was too risky to rely on Bluetooth, however it would have been a lot more reliable. In terms of software, coding with a reliability on sensors is a lot better than using delays. This would have allowed the variable battery levels to have less of an impact on the performance of the robot.

These changes would have helped achieve the optimal performance of the robot, however, with the results achieved the team is more than satisfied, and will take many vital skills away from this course and project.