






# CSCI944

## Perception & Planning

### Assessment 3

Name	Student Number	Contribution	Signature
Md Abu Sayed Khan Faisal	8376311	100%	
Arabinda Saha Partha	8198378	100%	Arabinda Saha Partha
Farhan Javid	8154429	100%	
Sean Chang	8081591	100%	
Hassan Anis	8125612	100%	
Aleksandra Zaskalkina	7602509	100%	

**Abstract:** This report provides a comprehensive overview of the design, construction, and programming of our LEGO MINDSTORMS EV3 robot developed for Assignment 3. The robot is designed to complete two distinct tasks: a penalty shootout (Task 1) and a ball finding and goal-scoring task (Task 2). Using a combination of motors, sensors, and custom software, the robot autonomously navigates the field, detects and collects balls, and scores goals as specified in the assignment requirements.

# 1. Introduction

While we have had much experience in programming robots to perform tasks in a virtual environment, this is the first time we must design a physical robot that will execute tasks in the real world. Our robot was created from the LEGO MINDSTORMS EV3 kit. We programmed our robot using the LEGO MINDSTORMS EV3 education software [1] to execute the two tasks assigned.

## 2. Background

LEGO MINDSTORMS are a line of educational robotics kits that provide a central hub and battery pack, several motors and sensors, and structural pieces that utilize the LEGO connection system [2]. The LEGO MINDSTORMS kit is widely recognized as one of the best robotics education kits for entry level students [3] and is what is used to create our own robot to execute the tasks provided.

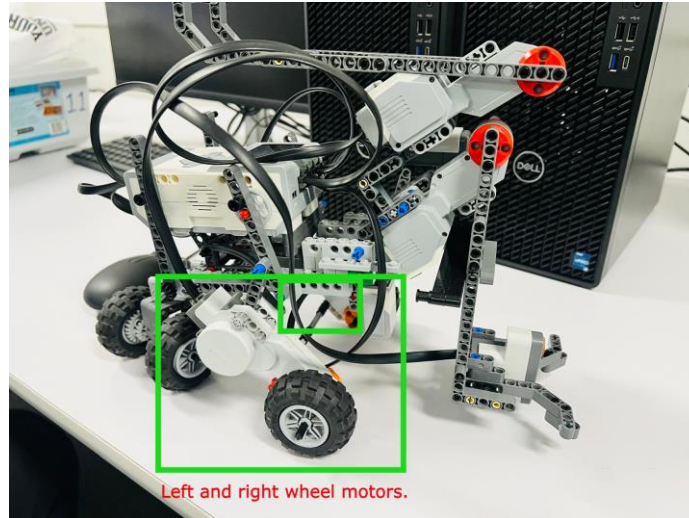
The tasks assigned are based on ball detection and the general idea of playing soccer. Programming robots to play soccer has long been a way to test how they perform tasks and make decisions in a common environment [4]. Essentially, the thought is that if a robot can identify an object as a ball and make the correct decision on what to do with that ball to score a goal, then similar logic can be applied in other areas and with other objectives. Especially so if the robot is expected to play both with and against a team of robots.

## 3. Hardware Design

Our robot is built using the LEGO MINDSTORMS EV3 kit and includes the following key hardware components

### 3.1. Motors

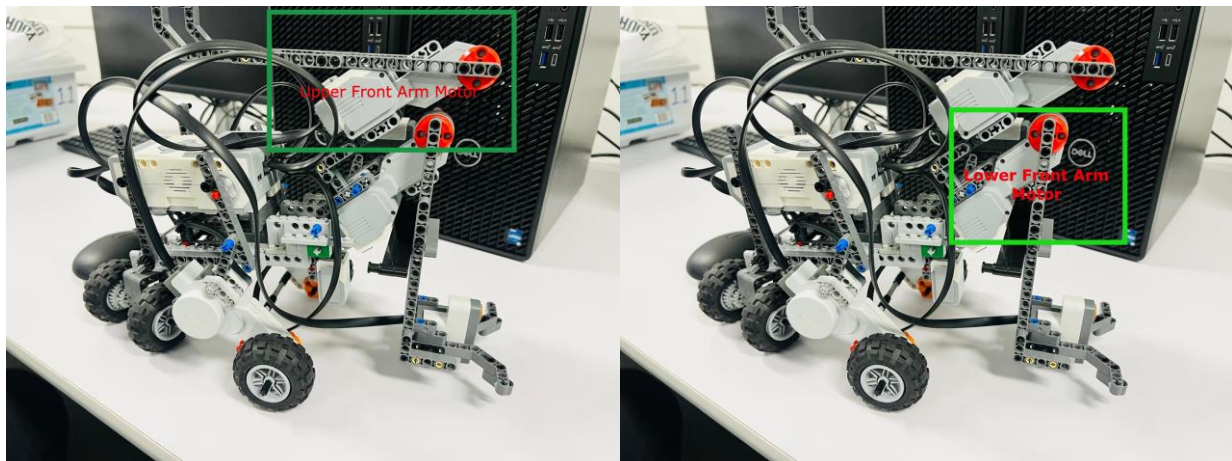
The robot is equipped with four motors. Two large motors are responsible for the movement of the robot (Figure 1). The left motor controls the left wheel, and the right motor controls the right wheel. The robot uses these two large motors to control its movement, enabling precise differential steering. This design allows the robot to navigate the field with accuracy, whether it's positioning itself for a penalty shot or searching for balls.



**Figure 1:** Wheel Motors

Two additional motors control the robot's two front arms (Figure 2):

1. **Upper Front Arm:** This arm is used to collect balls during Task 2. It lowers to grip the ball securely and lifts the ball for transport toward the goal. Upon arriving at the goal, the arm raises to let go of the ball. The estimated range of its grabbing capability is approximately within 10 cm.
2. **Lower Front Arm:** This arm functions as a kicking mechanism during Task 1, delivering a powerful kick to shoot the ball into the goal from the penalty line.

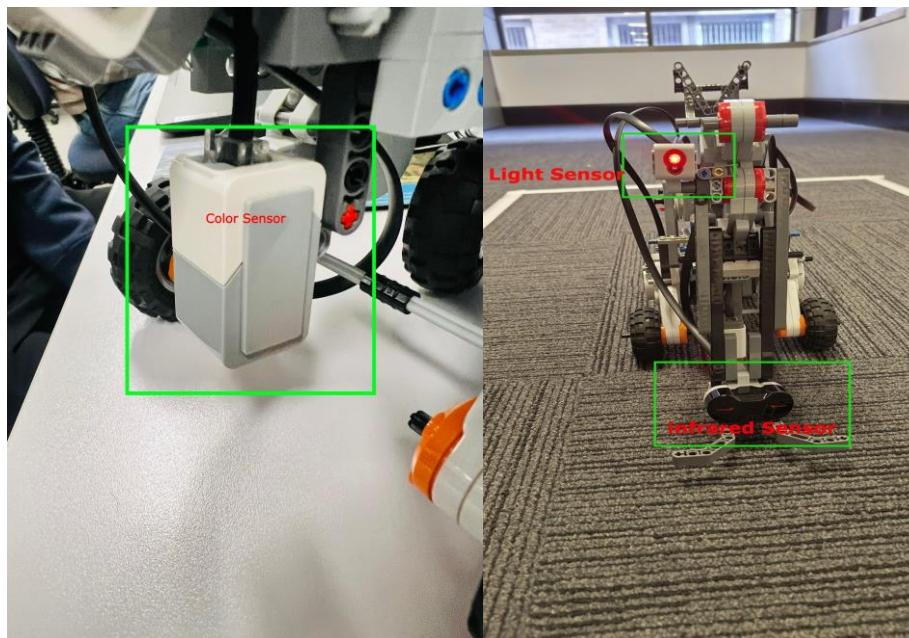


**Figure 2:** Arm Motors

### 3.2. Sensors

The robot is configured with three sensors to execute the two assigned tasks (Figure 3):

1. **Colour Sensor:** The colour sensor is located at the front bottom of the robot, pointing towards the ground. This sensor is capable of detecting either light or colour, depending on its coded configuration. Its role is to detect the boundary lines on the field and to keep the robot within the play area. This was achieved by detecting reflected colour intensity above 40%, which contrasted the dark floors. Additionally, it is utilized for identifying the penalty line in Task 1.
2. **Infrared Sensor:** The infrared sensor is positioned at the front of the robot and is aligned parallel to the ground. It is responsible for detecting objects, like balls, the tasks. This sensor returns a percentage value, where a smaller percentage indicates closer proximity or an object to the sensor. Through practical testing, the grabbing arm's maximum range of 10cm was found to correspond to a proximity of 18%, which was set as the threshold for the grabbing function to activate.
3. **Light Sensor:** The light sensor is located at the front of the robot and is positioned higher than the infrared sensor. It is the same type of sensor as the colour sensor and is responsible for detecting a guiding flashlight that is utilized to align the robot with the goal. The sensor's output was configured to detect ambient light intensity and provided readings in terms of percentage, where a higher percentage indicated greater light intensity. A threshold of 15% ambient light intensity was used to allow the robot to distinguish between the flashlight and external room lighting. This sensor played a crucial role in both tasks by assisting the robot in adjusting its position prior to kicking or pushing the ball.



**Figure 3:** Sensor Placement

## 4. Software Design

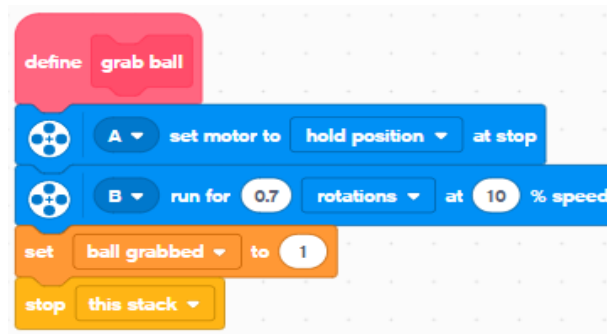
The LEGO MINDSTORMS EV3 environment is used to program the robot, enabling it to control its motors and carry out the designated tasks based on sensor input. Four distinct programs were created to address the specific demands of each challenge, with one light-guided and one autonomous program dedicated to each task. The light-guided solutions proved to be more robust, but they require continuous user input to navigate, collect balls, and score goals using the light sensor. In contrast, autonomous solutions only need user input for alignment when scoring goals, relying on the light sensor for this specific task. The following is an in-depth overview of the execution states, control flow and software architecture for both tasks.

For the successful execution of both tasks, specific states were monitored to support the algorithm's functionality, as outlined below:

Configuration	States
Task 1 light-guided	["ball grabbed", "turn speed"]
Task 1 autonomous	["scan speed", "approaching", "ball grabbed", "ball proximity"]
Task 2 light-guided	["ball grabbed", "turn speed"]
Task 2 autonomous	["scan speed", "approaching", "ball grabbed", "ball proximity"]

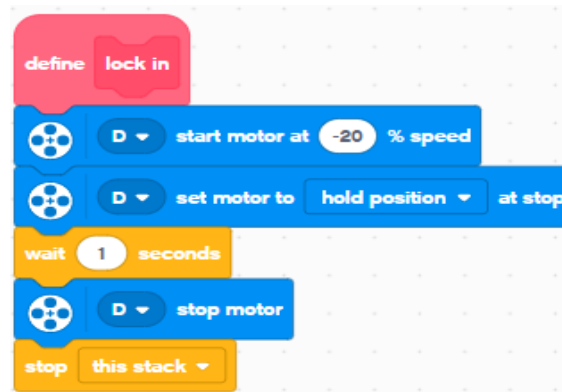
The following functions were universally applied in all configurations:

**Grab ball:** This function actuates the grabber arm to sweep downwards and hold its position, locking the ball in place relative to the robot. It also sets the "ball grabbed" variable to 1.



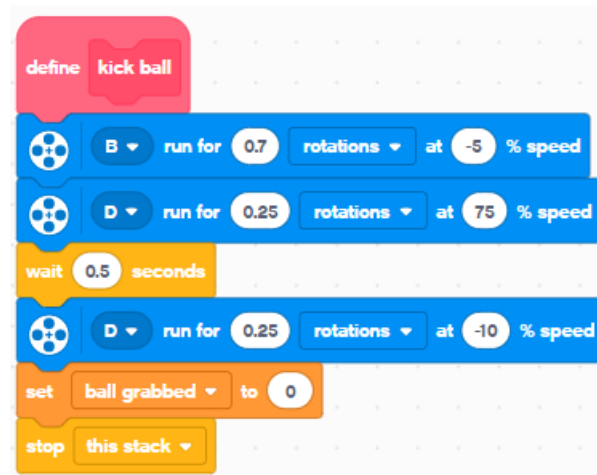
**Figure 4:** Blocks to Grab the Ball

**Lock in:** This function locks the kicker arm in place against a support to ensure the infrared sensor maintains a stable position while scanning for balls.



**Figure 5:** Blocks to locks the kicker arm

**Kick ball:** This function first moves the grabber arm out of the way, then moves the kicker arm rapidly to kick the ball. The kicker arm is then retracted to its original position. The “ball grabbed” variable is set to 0.



**Figure 6:** Blocks to kick the ball with arm

## 4.1 Task 1: Penalty Shootout

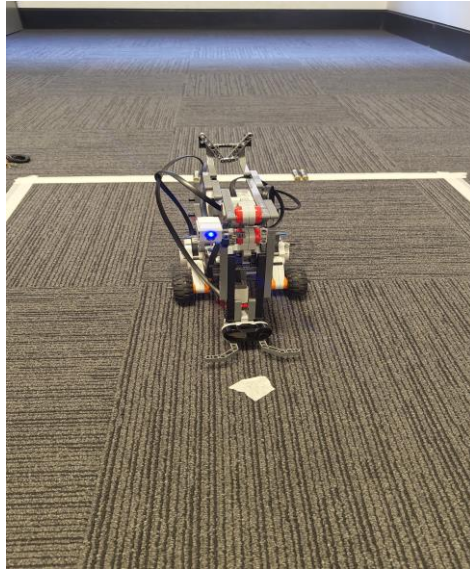
### 4.1.1 Robot Execution

In this task, the robot is expected to shoot a ball toward the goal from the penalty line after moving toward the back wall and turning around. The code for the light-guided and autonomous solutions can be found in appendices B and E respectively. The software is designed to handle the following steps:

**Move Toward the Back of the Field:** When activated, the robot utilizes its two large motors to propel itself forward. Positioned at the bottom of the robot, the color sensor constantly scans the ground for boundary lines. If it detects a boundary, the robot corrects its path to remain inside the



designated area. The robot persists in its movement until it identifies the back of the field, signaled by the color sensor when it crosses the rear boundary line.

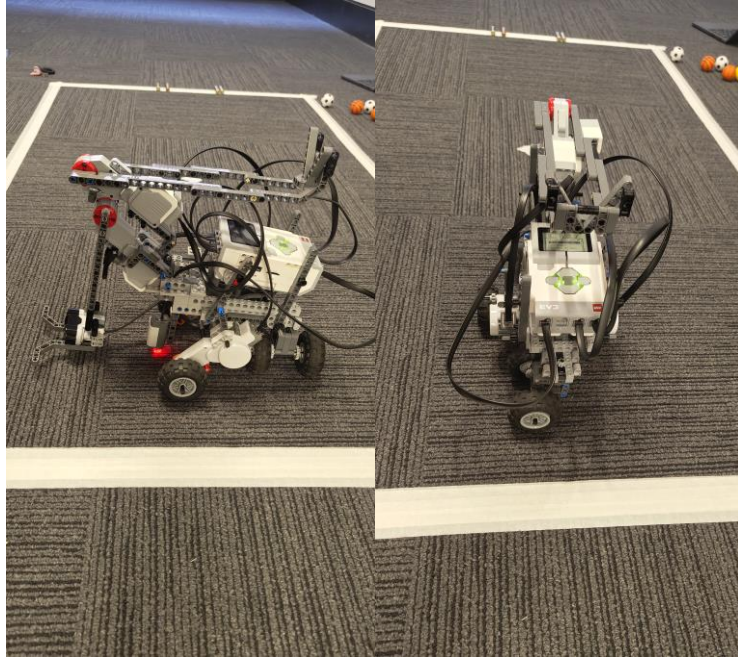


**Figure 7:** Moving toward the back of the field

**Turn Around and Move to the Penalty Line:** After reaching the back of the field, the robot reverses slightly before performing a 180-degree differential turn by actuating the wheels in opposite directions.

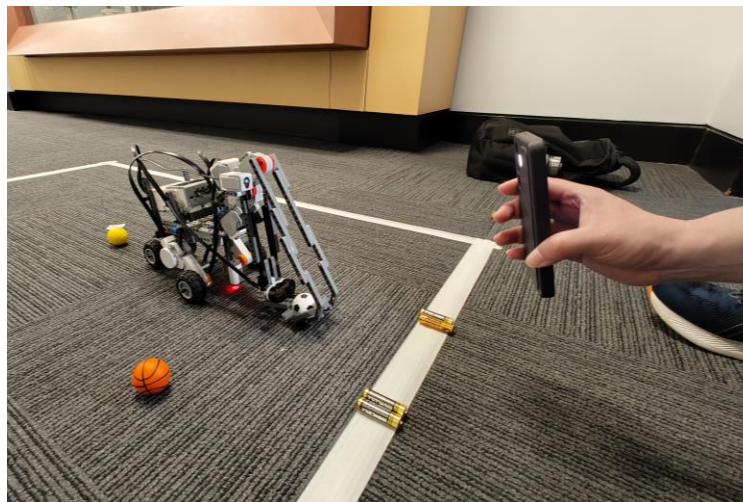
In the case of the autonomous solution, the robot then moves a fixed distance forward until it is in the estimated general vicinity of the ball. It then proceeds to autonomously rotate on the spot via differential turning while using the infrared sensor to attempt to detect its proximity to the ball. It then moves a short distance towards the closest detected object before sweeping again. This process is repeated until the detected proximity is within the 10cm range, and the ball grabbing function is used.

For the light-guided solution, the robot keeps moving forward unless it detects an input to the light sensor, in which case it rotates on the spot via differential turning before proceeding to move forward in its new direction. The robot initially rotates clockwise, then alternates its rotation direction with each subsequent activation of the function by the flashlight.



**Figure 8:** Turning around towards the goal

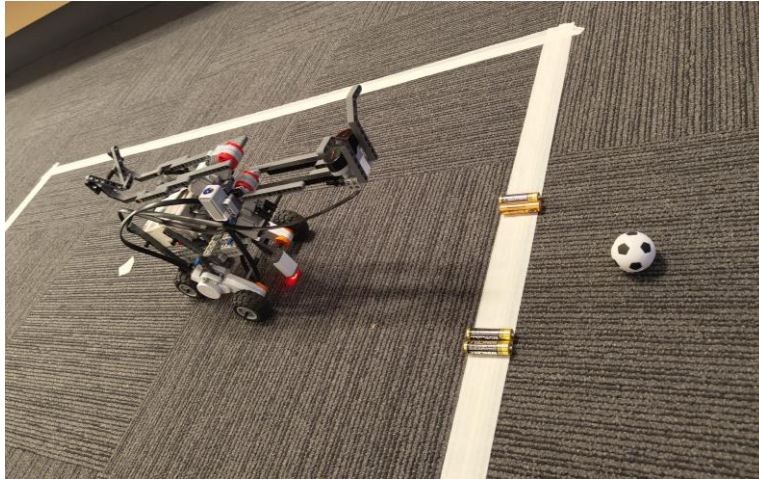
**Align with the Goal Using the Light Sensor:** This step only applies for the autonomous solution. Once the ball has been grabbed, the robot proceeds to rotate on the spot until the flashlight is detected, upon which the robot will stop moving. This is used to align the robot's aim with the goal. This step is unnecessary for the light-guided solution since the user's earlier guiding of the robot towards the ball also serves to align its aim with the goal.



**Figure 9:** Robot being guided by a torch

**Kick the Ball:** Upon reaching the correct alignment with the goal, the robot engages the lower front arm, which is linked to a motor, to propel the ball. The arm is specifically engineered to deliver sufficient force for shooting the ball into the goal. This sequence is repeated for all 10 penalty shots, with the robot making adjustments and kicking each ball one after the other.





**Figure 10:** Kicking mechanism

## 4.2. Task 2: Ball Finding and Goal Scoring

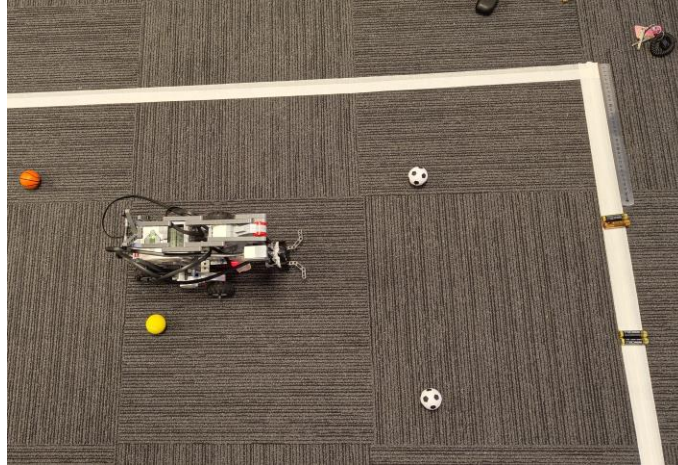
### 4.2.1 Robot Execution

In this task, the robot must find and collect balls placed randomly on the field, then push them into the goal within a 5-minute time frame. The code for the light-guided and autonomous solutions can be found in appendices C and F respectively. The software is designed to handle the steps below.

**Ball Detection Using the Infrared Sensor:** In the case of the autonomous solution, the robot would begin by performing a scanning sweep while rotating 360 degrees on the spot. If it detects an object within its proximity, it moves slightly in that direction and performs another scanning sweep. This is repeated until the detected ball is close enough such that the ball grabbing function is called. In the case where no nearby objects are detected, the robot continues to move forward in its original direction.

In the case of the light guided solution, the robot will start moving forward. When the flashlight is used to activate the light sensor, the robot will stop and rotate clockwise on the spot before continuing forward in its new direction. Every subsequent activation will cause the robot to rotate in the direction opposite to the last rotation. Thus, the user can use the flashlight to direct the robot to nearby balls.

In both cases, the robot stops moving and the grabbing function is automatically called once an object is detected within 10cm proximity to the infrared sensor.



**Figure 8:** Ball detection using infrared sensor

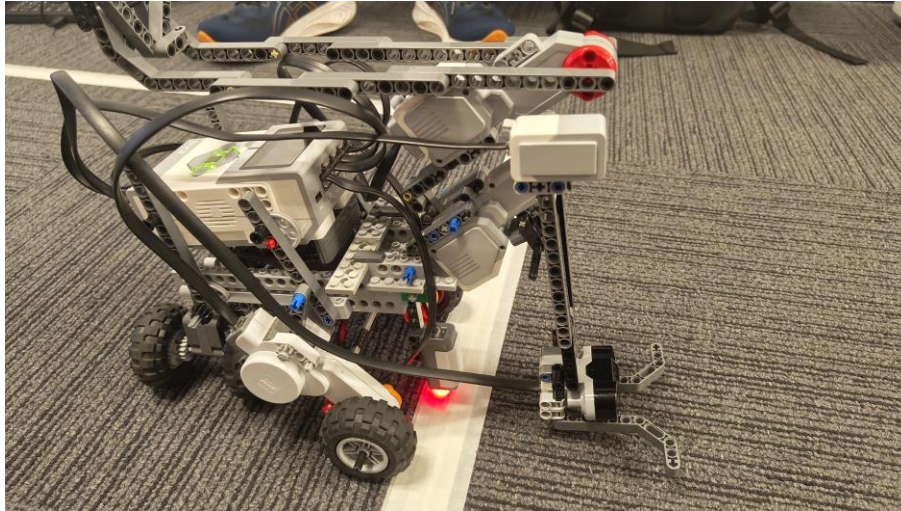
**Ball Collection with the Upper Front Arm:** Once the ball is detected, the robot activates the grabber arm, which lowers to grip the ball. The concave shape of the grabber locks the ball in place and allows the robot to easily bring it along as it moves. Both the kicker and grabber arm are set to lock in place while the ball is grabbed, rigidly securing the ball during movement.



**Figure 9:** Ball collection with upper front arm

**Boundary Detection:** In both cases while the robot is searching for balls, the colour sensor detects if the robot is passing over a boundary line. This triggers the robot will reverse slightly, perform a 60 degrees clockwise turn, then proceed with moving straight in its new direction.

In the case of the autonomous solution, this mechanism allows the robot to change its direction and eventually navigate the whole course until a ball is detected nearby.



**Figure 10:** Detecting the boundary

**Move Toward the Goal:** In the case of the autonomous solution, the ball will await two adjustments. The first is indicated by a flashlight input which will cause the robot to rotate clockwise until the light is removed. It will then move forward in this new direction until a second flashlight input causes it to stop and rotate anticlockwise. These two rotations allow the user to direct the robot's goal path.

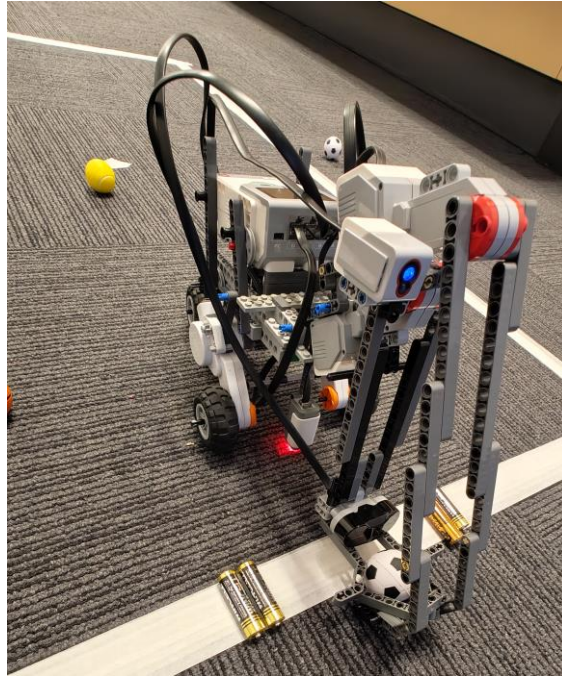
In the case of the light guided solution, the robot will proceed to navigate as in the previous step, with the user directing it to rotate via the flashlight.



**Figure 11:** Move toward the goal

**Push the Ball into the Goal:** Using the colour sensor, the robot detects when it has passed the goal line in both cases. The ball is then kicked out to free the kicker and grabber arms, and their positions are reset. This is followed by a short reverse and a 180 degree rotation by the robot, which then loops back to the beginning to search for more balls.





**Figure 12:** Push ball into goal

### 4.3 Autonomous ball finding details

The logic and control flow that enable the robot to autonomously locate balls is more intricate. The following explanation of the code is detailed in appendix D.

The "Ball scan" function manages the robot's repeated scanning sweeps and allows it to autonomously home in on a nearby ball. This function operates in a loop, utilizing the variable "approaching" to monitor the robot's progress toward the ball, and "ball grabbed" to confirm when the robot has successfully secured the ball.

There are three conditions for detection on the infrared sensor in this function:

- If the sensor detects proximity greater than 50%, the robot moves for 0.5 rotations of its wheels
- If the sensor detects proximity from 18% to 50%, the robot moves for 0.2 rotations of its wheels
- If the sensor detects proximity within 18%, the robot grabs the ball

"Ball approach" is the function which is called to allow the robot to detect and align itself with the closest detected proximity to itself. This works by setting 70% proximity as a bare minimum for detection. The robot then rotates on the spot in small increments and starts recording the reading of the infrared sensor once the "ball proximity" threshold of closer than 70% is achieved. This value is updated as long as each successive increment indicates an object in closer proximity to the sensor. Once the sensor indicates a further proximity value, the scanning loop ends and the last rotation increment is reversed, ensuring that the robot will home in on a local minimum.

## 5. Challenges and Solutions

During development, we faced several challenges in the building of the robot and the execution of the tasks in code.

**Ball Detection:** Ball detection was the most significant challenge for both autonomous solutions. With the infrared sensor's maximum range of 255 cm, our initial approach involved having the robot perform a 360-degree sweep of the field, detecting and moving toward the nearest object. However, this strategy didn't account for the difficulty in detecting smaller objects, like the balls placed on the floor. To implement this sweeping scan, the infrared sensor had to be positioned parallel to the ground, but due to the small size of the balls, the sensor had to be placed very close to the floor. This resulted in partial detection of the ground, which significantly reduced the sensor's effective range. Additionally, the natural fluctuations in sensor readings, caused by vibrations from the robot's movements, further compromised its precision during operation. As a result, the autonomous solutions required the robot to make smaller movements and perform frequent scanning sweeps with a much reduced detection range of 20cm to search for nearby balls, due to the sensors' significantly reduced range. Additionally, relying solely on the infrared sensor's proximity readings proved less reliable than the light-guided solutions, as the sensor occasionally failed to detect nearby balls during sweeps. Although reducing the sweep rotation speed and breaking up the sweep into smaller intermittent rotations helped to partially mitigate this issue, it still occurred intermittently.

**Steering Stability:** In our initial robot design, it had three wheels: two at the front, controlled by motors for movement, and a third at the back for balance. However, this setup caused the robot to tip over when turning too quickly. Our first attempt to solve this was by reducing the turning speed, but this significantly slowed down the robot's overall performance. Ultimately, we resolved the issue by adding a second wheel at the back to improve stability during turns.

**Infrared Sensor Stability:** The infrared sensor was mounted on the kicker arm and was therefore not secured rigidly to the robot's body. An additional support was built behind the arm and the "lock in" function was implemented so that the kicker arm motor would lock the kicker arm against this support while the robot was searching for balls. This provided sufficient support to the sensor such that proximity readings would not fluctuate wildly during operation.

**Ball Collection:** Initially, the ball collection mechanism had a risk of causing balls to roll out of the collection zone if they weren't perfectly aligned in the center during the grabbing process. To address this, the heads of the grabbing and kicking arms were redesigned with a concave shape, allowing the grabbed ball to be funneled toward the center. This modification greatly improved the success rate of the ball grabbing function.

**Boundary Detection:** The initial height of the color sensor occasionally caused inaccurate detection of floor colors. Manual testing was conducted to find an optimal distance between the



sensor and the floor, ensuring reliable and accurate detection of floor markings while avoiding being too close, which could cause the sensor to malfunction.

## 6. Recommendations for Improvement

While the current robot design performs well in both tasks, there are several areas that could be optimized to enhance its overall performance, accuracy, and efficiency.

The limited effective range of the infrared sensor significantly hindered the robot's detection capabilities. Implementing a more precise sensor could enable the robot to cover larger areas during sweeps and improve ball detection accuracy. Alternatively, adding a second infrared sensor could enable the implementation of triangulation functions, improving the robot's ability to locate the ball and increasing the success rate.

Although the current concave design of the kicking and grabbing heads effectively funnels the ball toward the center for locking between the arms, there were still occasional instances where the ball slipped out due to being positioned too far to one side of the robot's center. This issue could be minimized by widening the grabber and kicker heads to cover a larger area, improving overall control.

The current implementation of the light sensor for guiding and steering the robot, activated by a user with a flashlight, faced operational challenges, as the light sensor occasionally failed to detect the flashlight input. The most likely reasons for this included sudden changes in light intensity caused by the robot rotating away from the light source, as well as the user not consistently following the rotation, which led the intensity value to drop below the minimum threshold. Additionally, the light sensor could have had internal defects. Regardless of the cause, replacing the light sensor system with a dedicated signal-emitting beacon would enhance the robot's reliability.

From a software standpoint, integrating an internal mapping system to monitor the robot's relative position on the track would enable more intelligent navigation decisions during autonomous operation for Task 2. This improvement would result in shorter paths, faster completion times, and overall enhanced performance.

## 7. Conclusion

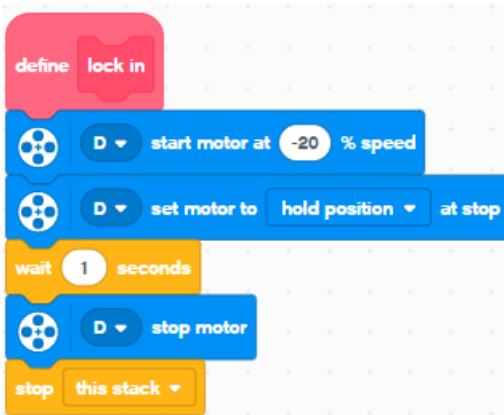
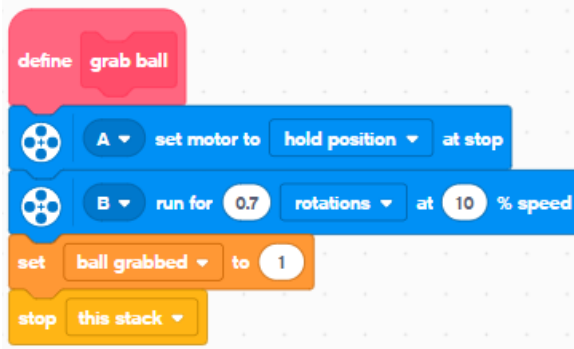
The robot was designed to complete both tasks with precision and efficiency. The dual-arm design, combined with the integration of multiple sensors, allowed the robot to perform complex behaviors autonomously. The modular software design ensured that each task was handled effectively, resulting in a robot that could successfully perform the penalty shootout and the ball finding and goal scoring tasks.

## 8. References

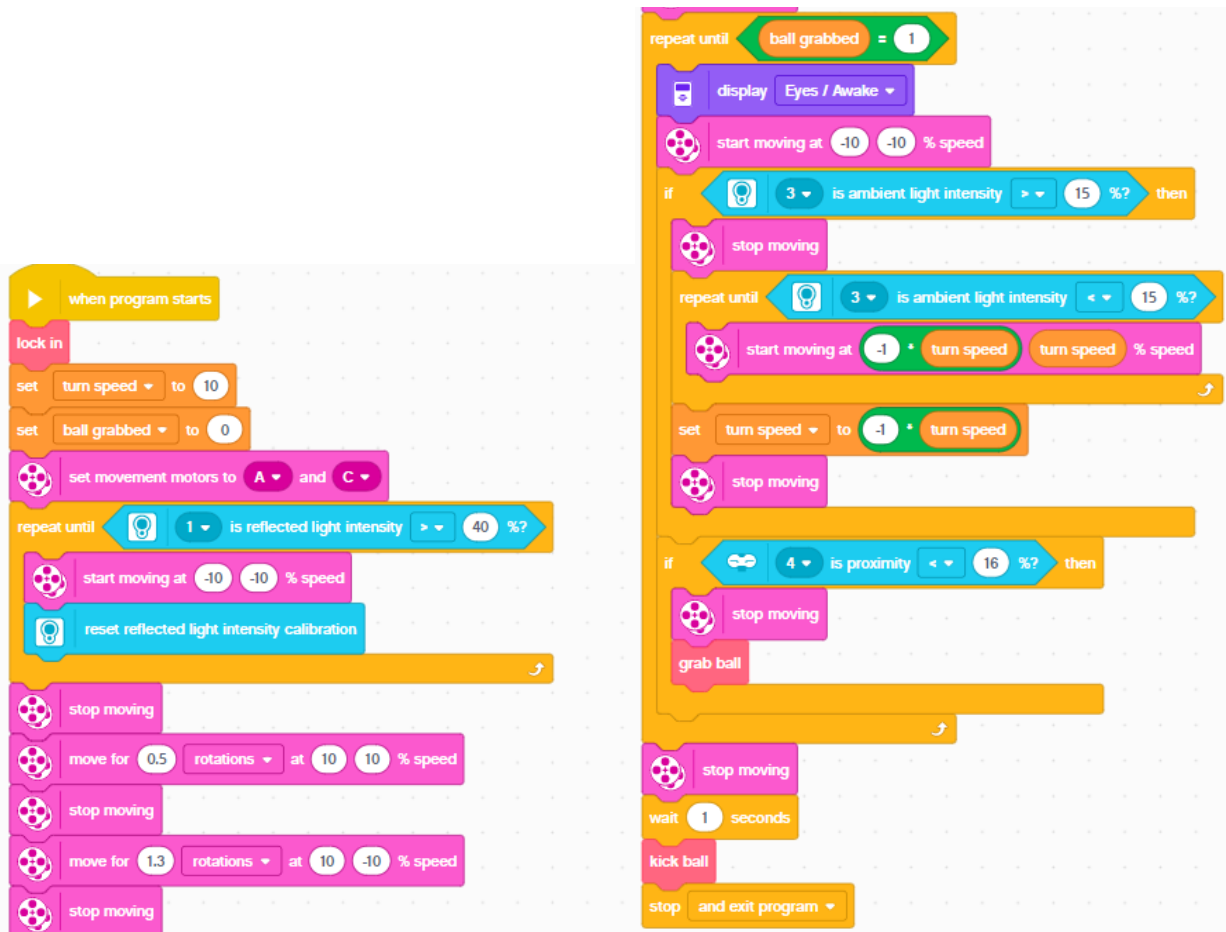
- [1] “MINDSTORMS EV3 downloads – LEGO Education,” LEGO® Education.  
<https://education.lego.com/en-us/downloads/mindstorms-ev3/software/> (accessed Oct. 13, 2024).
- [2] “What is LEGO® MINDSTORMS®? | Official LEGO® Shop AU,” Lego.com, 2024.  
[https://www.lego.com/en-au/themes/mindstorms/how-it-works?icmp=LP-SHH-Standard-Mindstorms\\_About\\_HB\\_Let\\_Get\\_Technical-TH-MD-LIXHJUYDDN](https://www.lego.com/en-au/themes/mindstorms/how-it-works?icmp=LP-SHH-Standard-Mindstorms_About_HB_Let_Get_Technical-TH-MD-LIXHJUYDDN) (accessed Oct. 13, 2024).
- [3] L. Fortunati, A. M. Manganelli, and G. Ferrin, “Arts and crafts robots or LEGO® MINDSTORMS robots? A comparative study in educational robotics,” *International Journal of Technology and Design Education*, Jul. 2020, doi:  
<https://doi.org/10.1007/s10798-020-09609-7>.
- [4] J. Ruiz-del-Solar, R. Verschae, M. Arenas and P. Loncomilla, "Play Ball!," in *IEEE Robotics & Automation Magazine*, vol. 17, no. 4, pp. 43-53, Dec. 2010, doi:  
10.1109/MRA.2010.938840.

## 9. Appendices

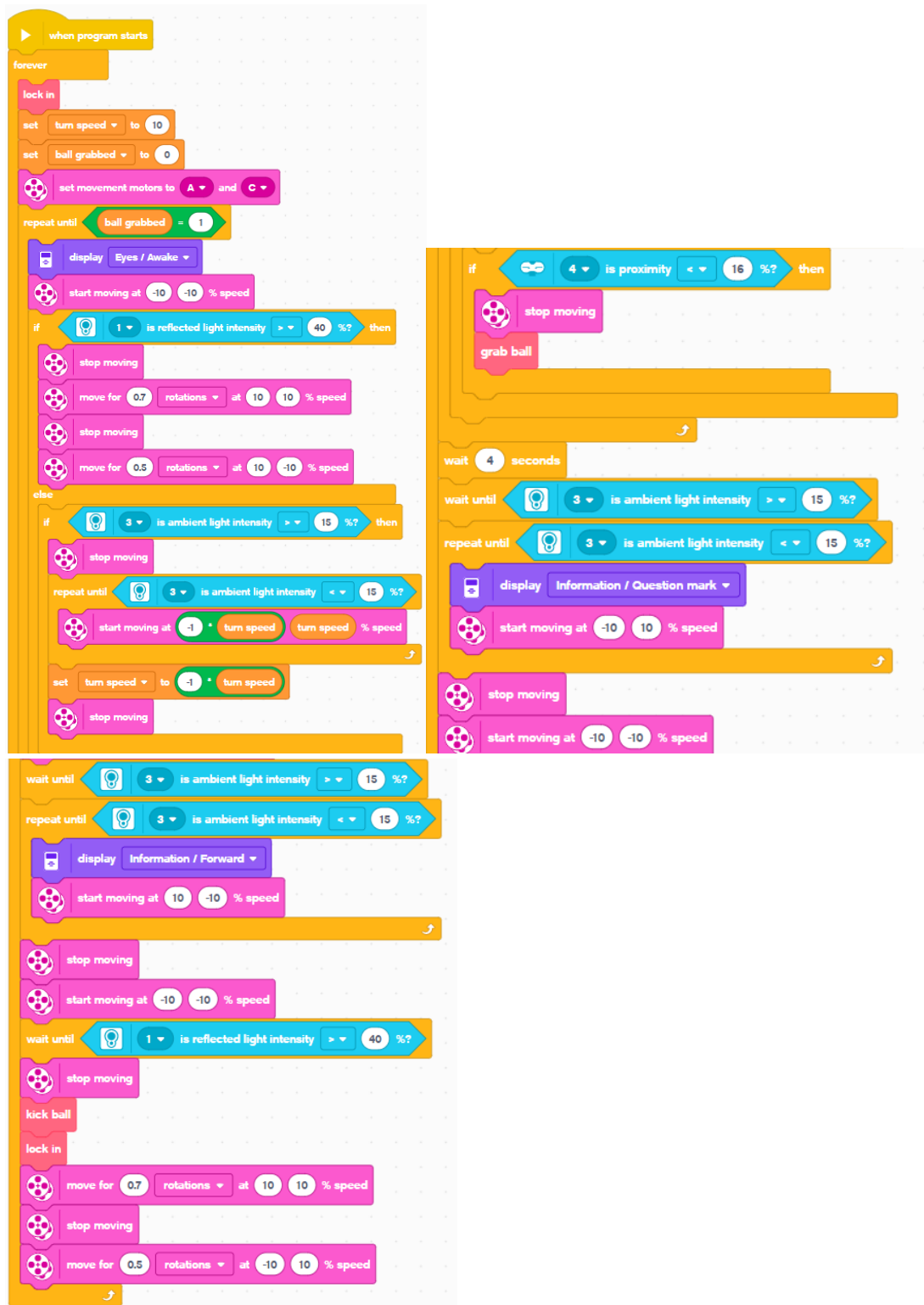
### Appendix A: Universal functions



## Appendix B: Block code for Task 1 light-guided solution

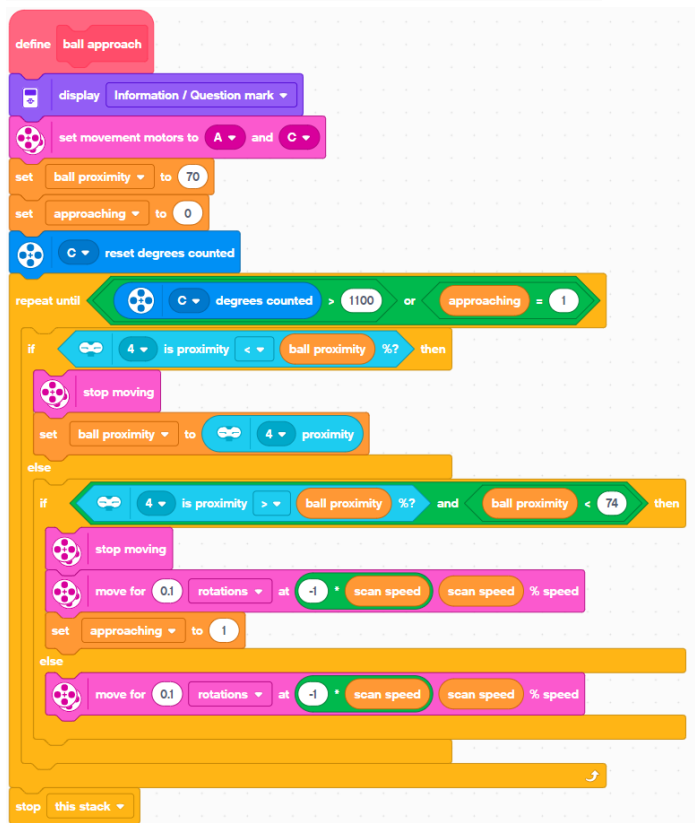
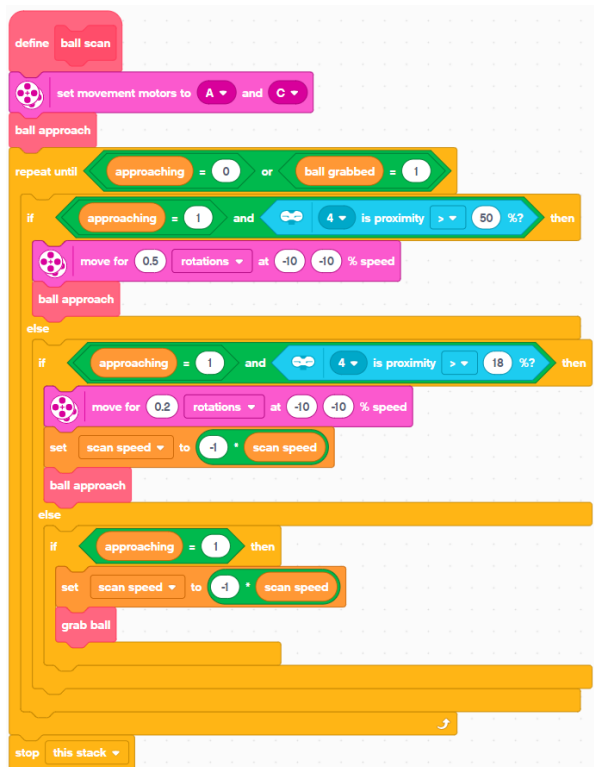


## Appendix C: Block code for Task 2 light-guided solution

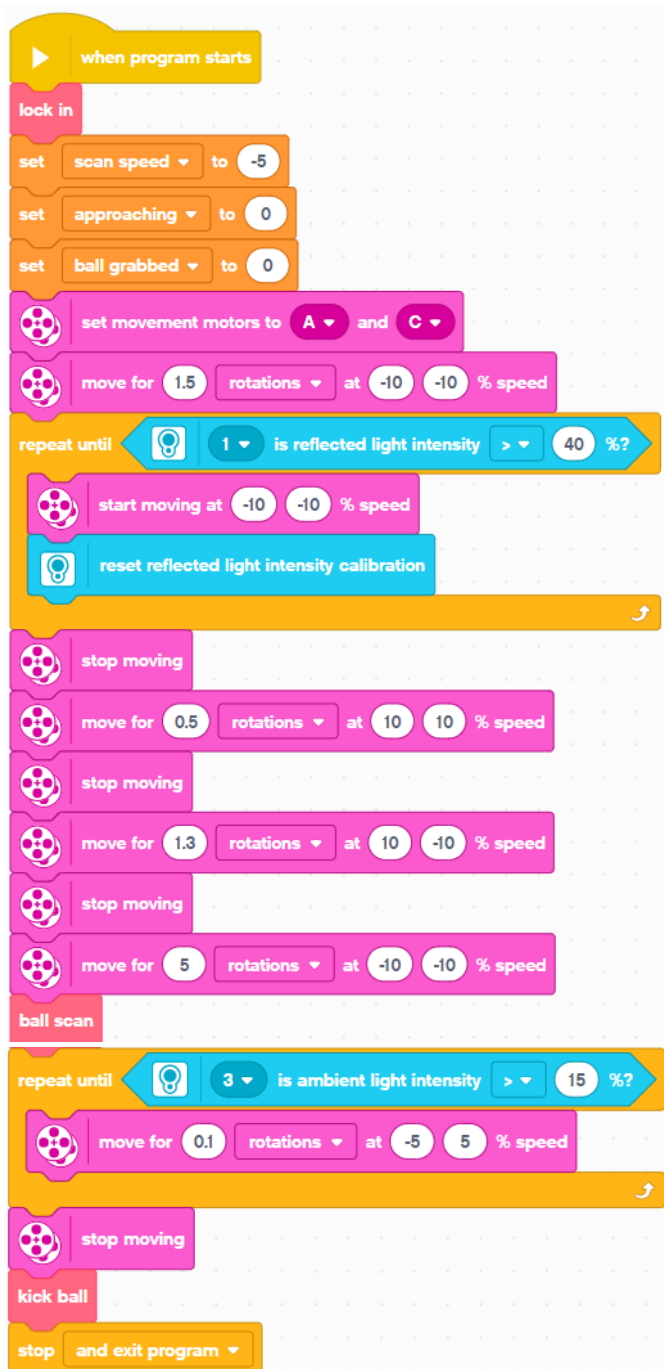




## Appendix D: Sweep scanning function for autonomous solutions



## Appendix E: Block code for Task 1 autonomous solution



## Appendix F: Block code for Task 2 autonomous solution

