RIDERS
71832P

87

POINTS

ASUNCIÓN
PARAGUAY

VEX CODE VR

Participants

Facundo

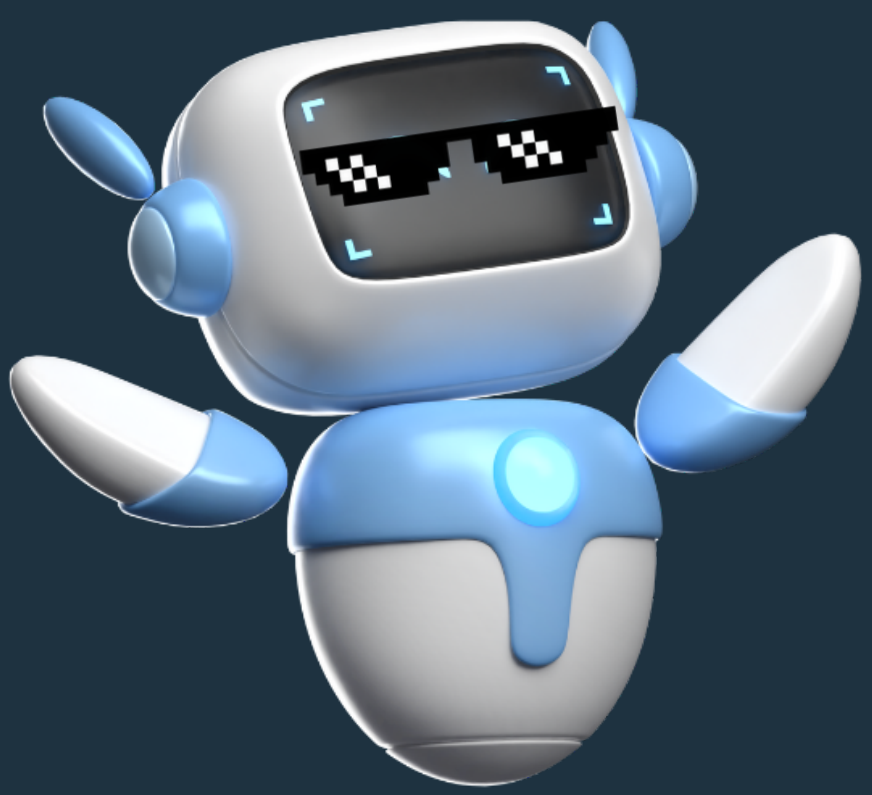
High-quality programmer and leader. With notable achievements and experience in VRC, our team engineer, notebooker, programmer and four-time VEX Worlds competitor, steps up to this challenge to redefine the boundaries of programming. Nothing is too much for these two friends at the computer.

Two heads are better than one. The driver, engineer, programmer, and five-time VEX Worlds competitor is ready to compete in this challenge. Alongside his teammate, they aim to once again solidify the title of the world's strongest team: Riders.

Santino

How did we run the program?

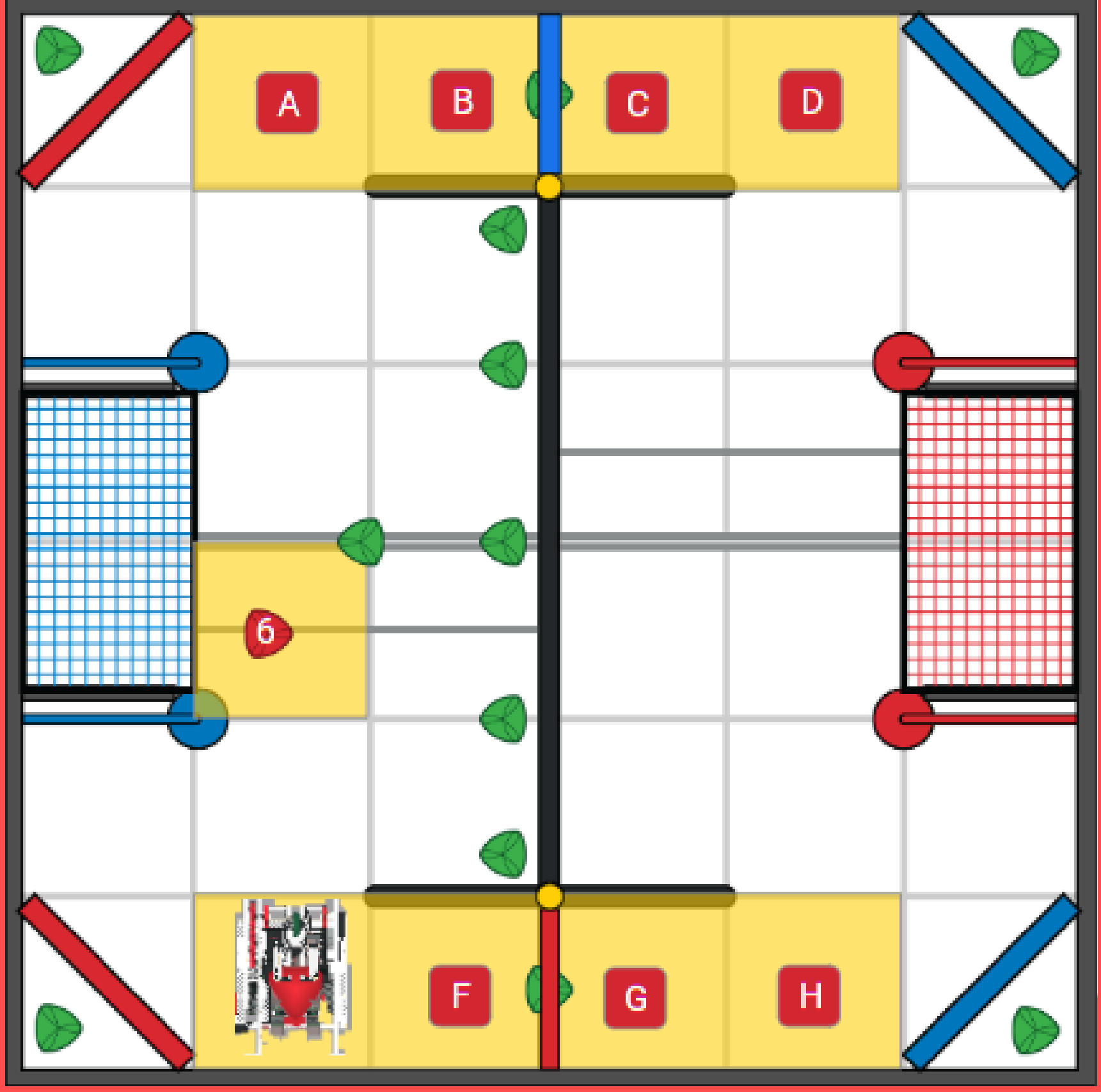
- We used a powerfull dedicated GPU (NVIDIA RTX 3060)
- We disabled web browser VSYNC
- We kept the FPS from 150 to 300
- We kept the laptop in a good temperature



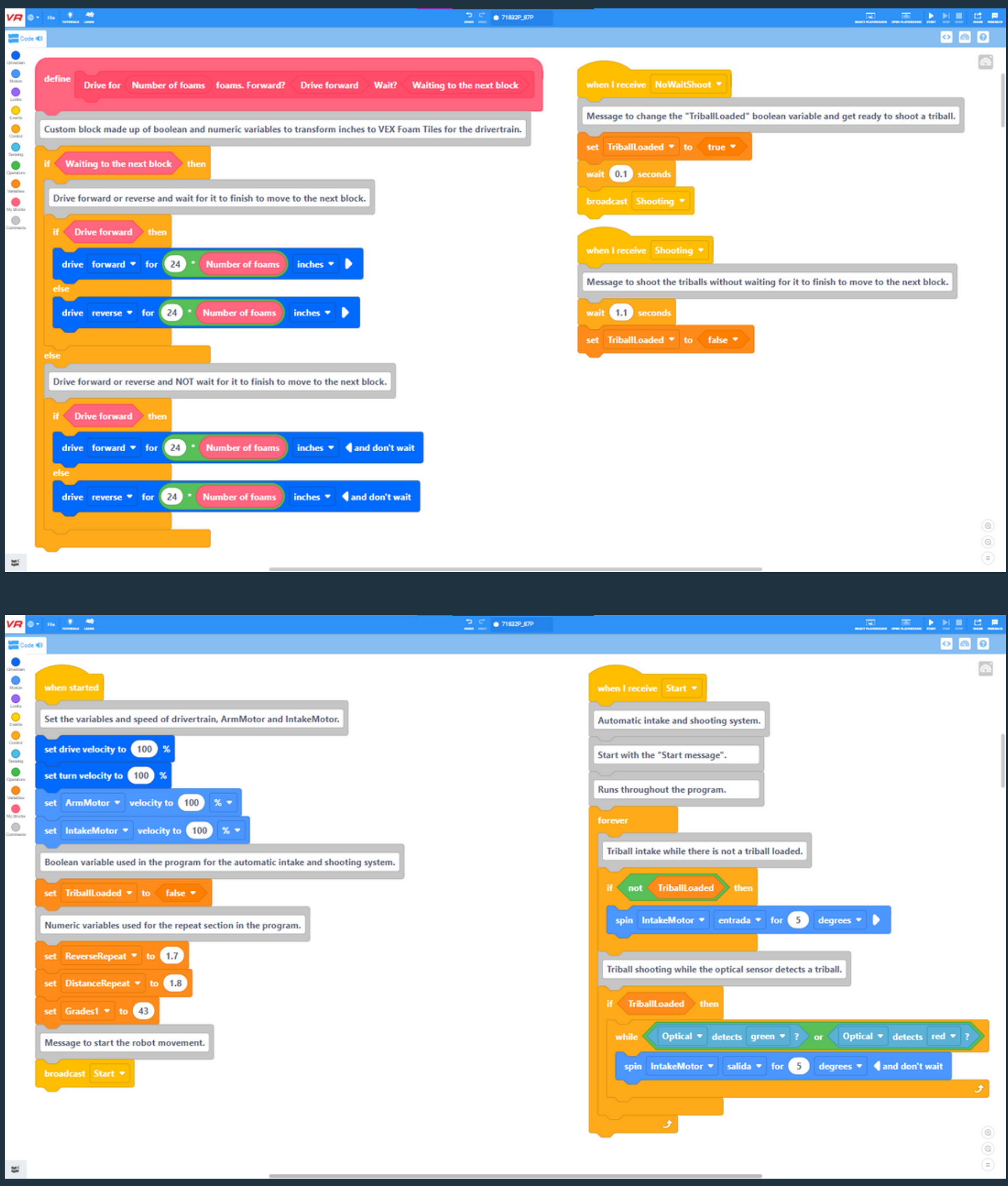
These are the conditions under which we programmed and ran the simulator. Our score and the proper functioning of the code were obtained under these conditions, so it's possible that with different parameters, the simulation may exhibit variations in its performance.

Pre-Match Checklist

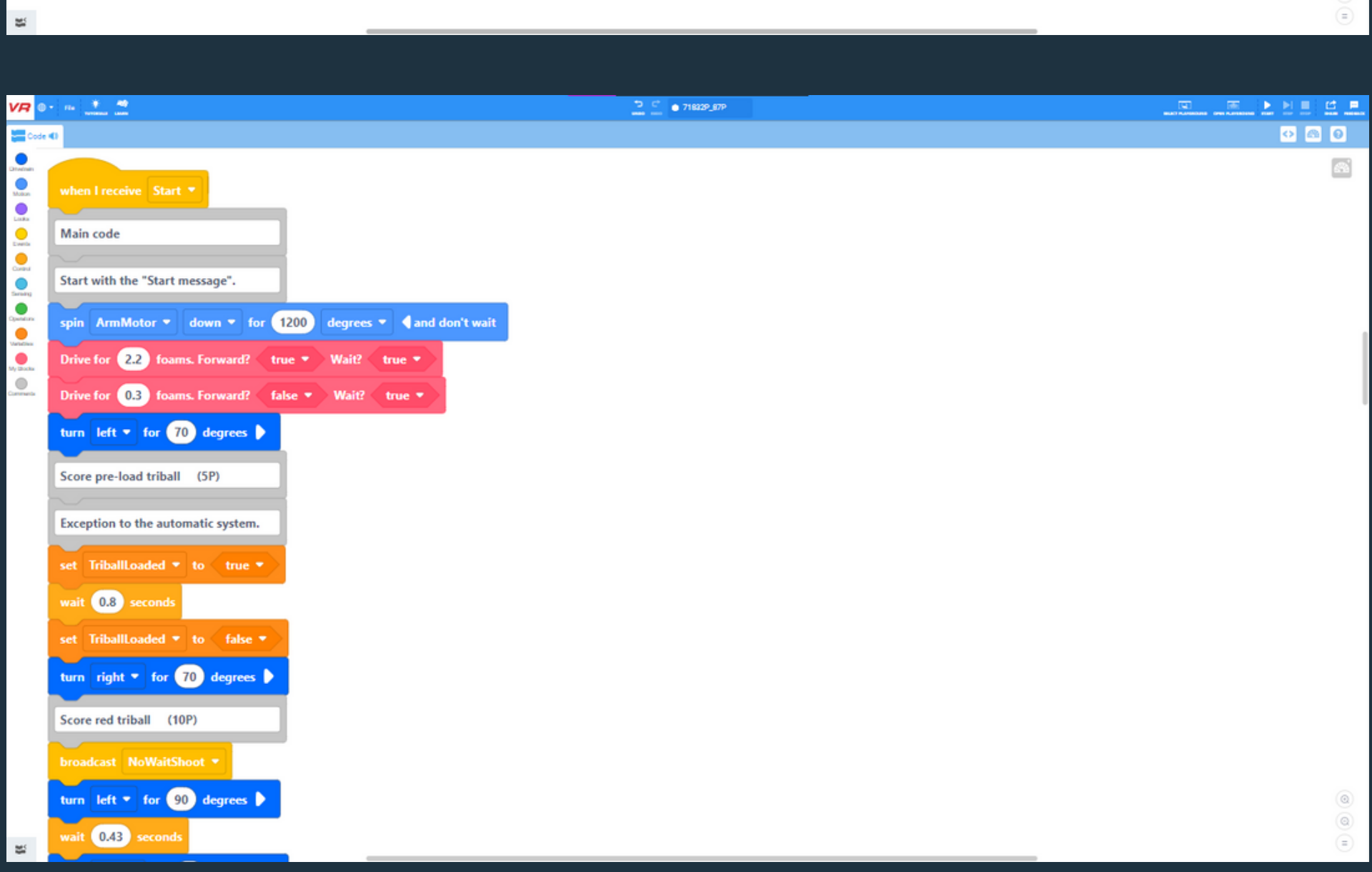
START LOCATION	DIRECTION	PRE-LOAD	FIELD PRE-LOAD
E	N		6



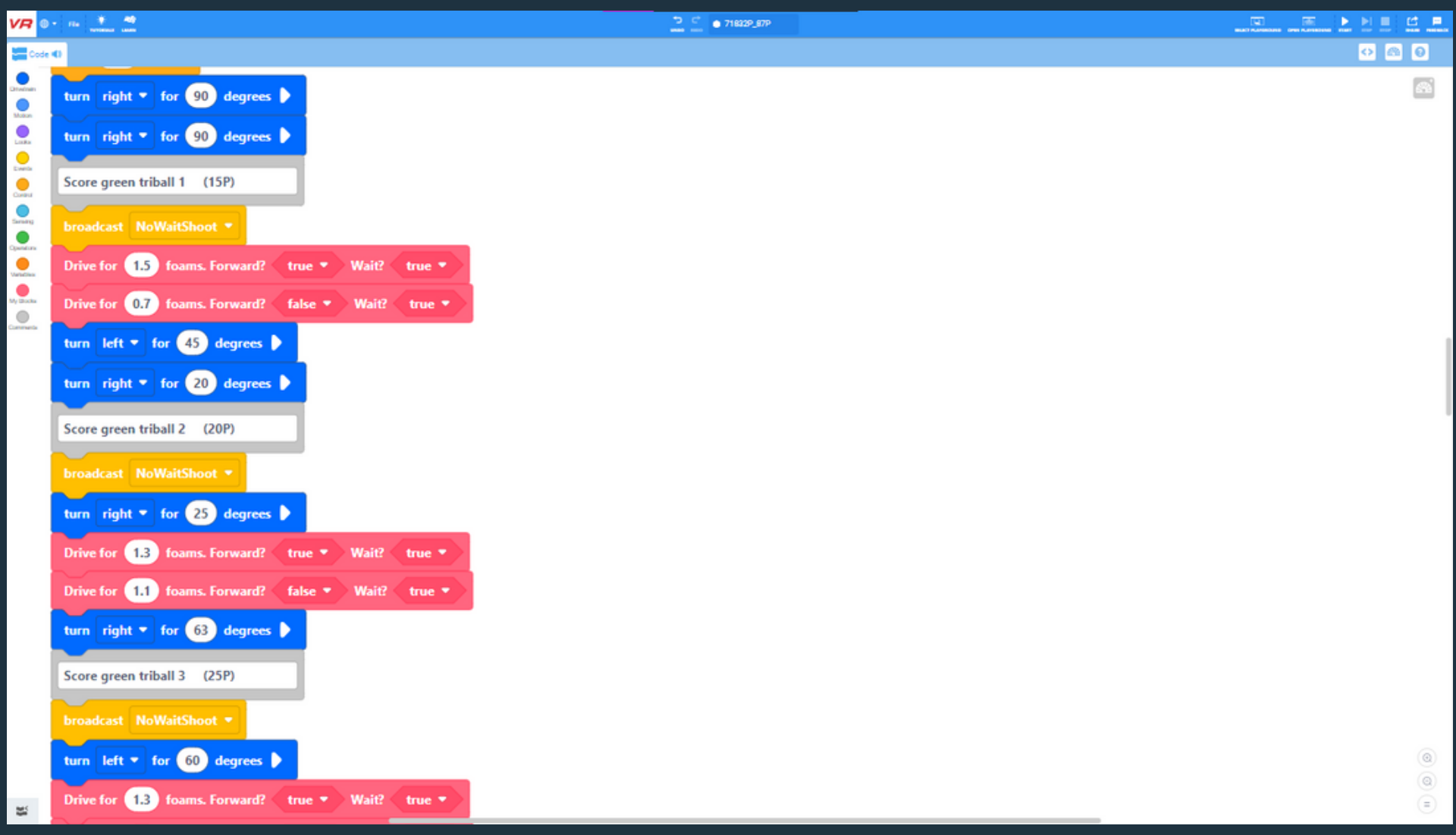
Full Code (1)

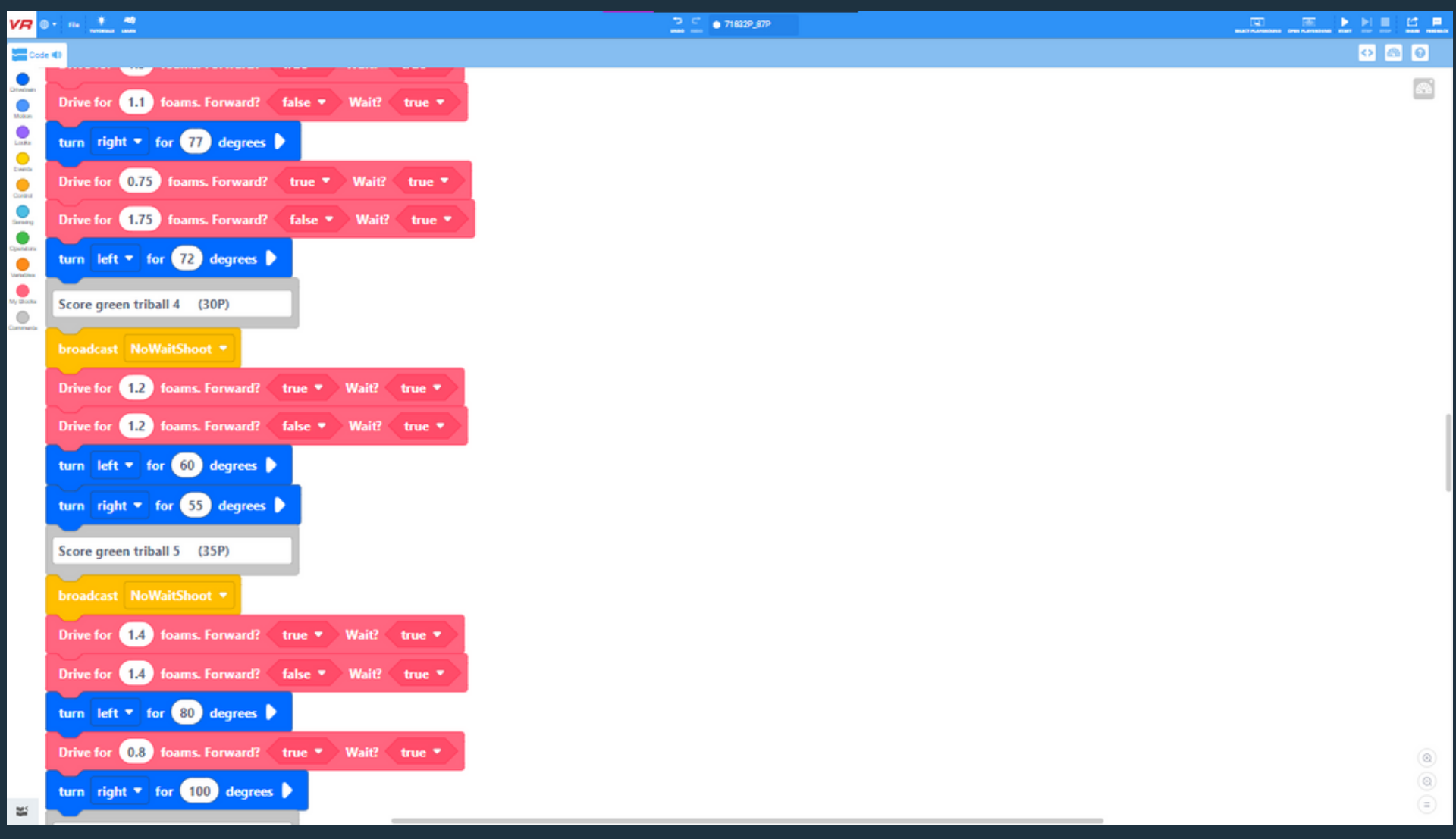

define Drive for Number of foams foams. Forward? Drive forward Wait? Waiting to the next block
Custom block made up of boolean and numeric variables to transform inches to VEX Foam Tiles for the drivertrain.
if Waiting to the next block then
Drive forward or reverse and wait for it to finish to move to the next block.
if Drive forward then
drive forward for 24 * Number of foams inches
else
drive reverse for 24 * Number of foams inches
else
Drive forward or reverse and NOT wait for it to finish to move to the next block.
if Drive forward then
drive forward for 24 * Number of foams inches and don't wait
else
drive reverse for 24 * Number of foams inches and don't wait
when I receive NoWaitShoot
Message to change the "TriballLoaded" boolean variable and get ready to shoot a triball.
set TriballLoaded to true
wait 0.1 seconds
broadcast Shooting
when I receive Shooting
Message to shoot the triballs without waiting for it to finish to move to the next block.
wait 1.1 seconds
set TriballLoaded to false

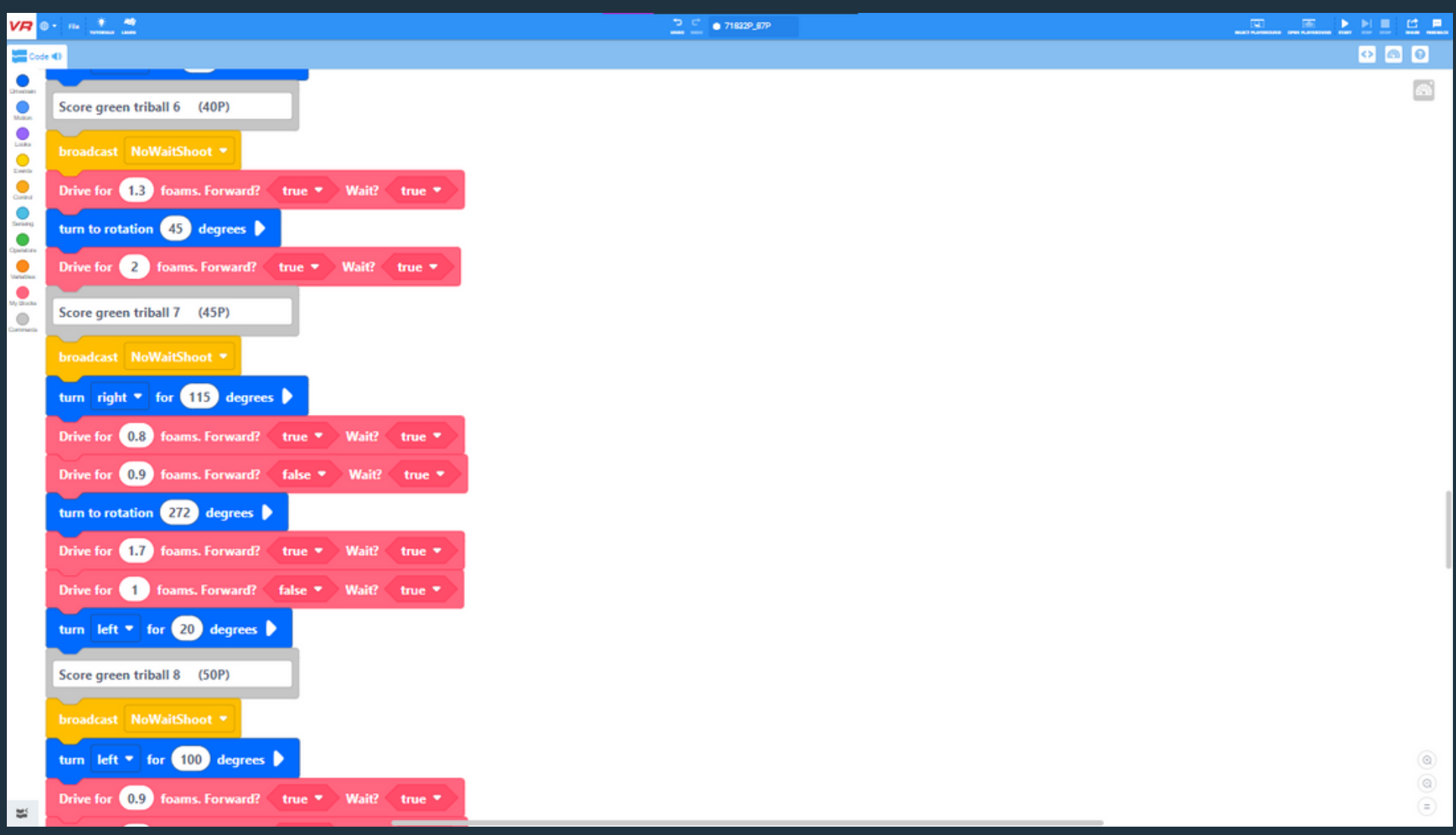
when started
Set the variables and speed of drivertrain, ArmMotor and IntakeMotor.
set drive velocity to 100 %
set turn velocity to 100 %
set ArmMotor velocity to 100 %
set IntakeMotor velocity to 100 %
Boolean variable used in the program for the automatic intake and shooting system.
set TriballLoaded to false
Numeric variables used for the repeat section in the program.
set ReverseRepeat to 1.7
set DistanceRepeat to 1.8
set Grades1 to 43
Message to start the robot movement.
broadcast Start
when I receive Start
Automatic intake and shooting system.
Start with the "Start message".
Runs throughout the program.
forever
Triball intake while there is not a triball loaded.
if not TriballLoaded then
spin IntakeMotor entrada for 5 degrees
Triball shooting while the optical sensor detects a triball.
if TriballLoaded then
while Optical detects green ? or Optical detects red ?
spin IntakeMotor salida for 5 degrees and don't wait


when I receive Start
Main code
Start with the "Start message".
spin ArmMotor down for 1200 degrees and don't wait
Drive for 2.2 foams. Forward? true Wait? true
Drive for 0.3 foams. Forward? false Wait? true
turn left for 70 degrees
Score pre-load triball (5P)
Exception to the automatic system.
set TriballLoaded to true
wait 0.8 seconds
set TriballLoaded to false
turn right for 70 degrees
Score red triball (10P)
broadcast NoWaitShoot
turn left for 90 degrees
wait 0.43 seconds

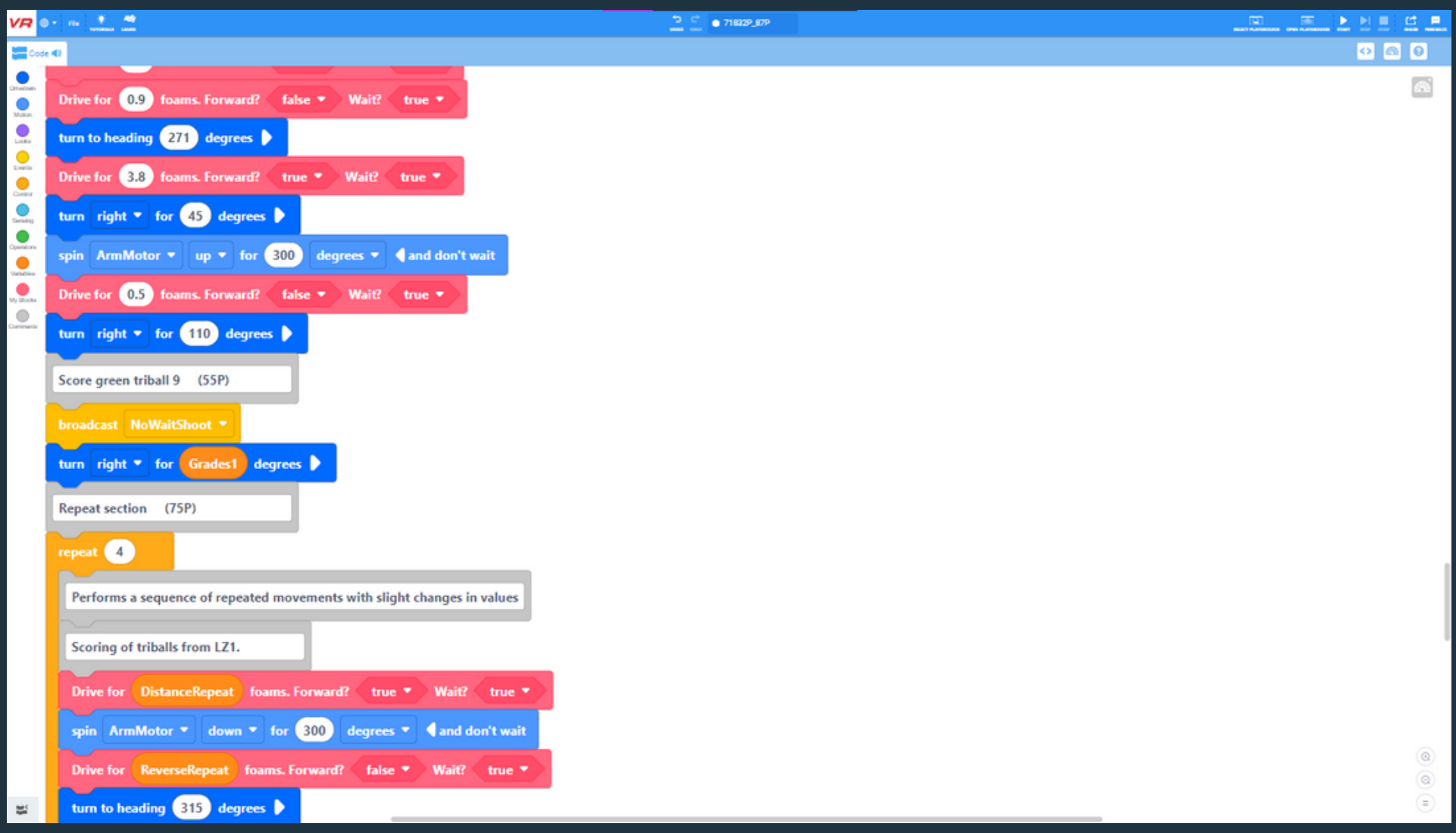
Full Code (2)

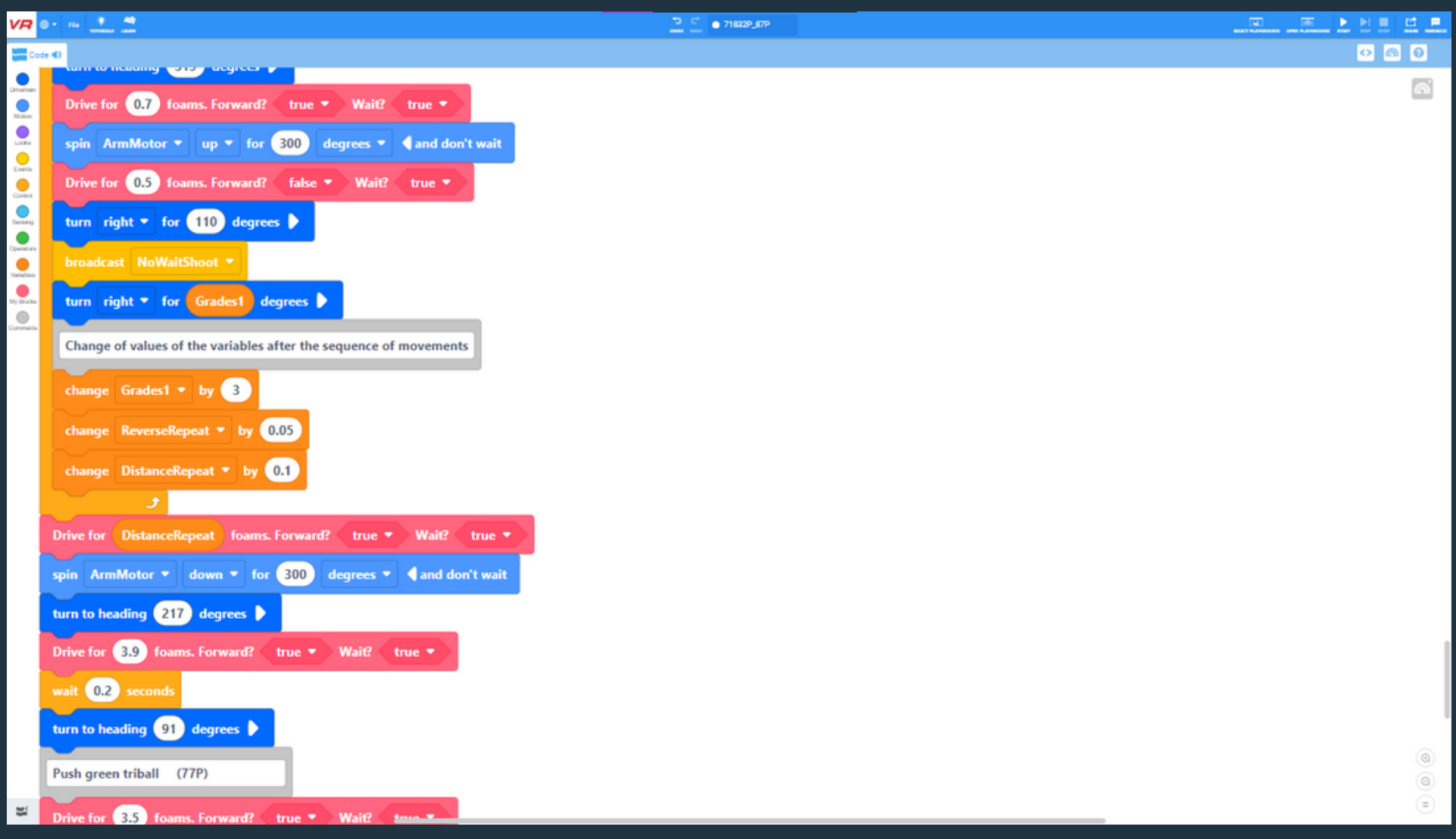

turn right for 90 degrees
turn right for 90 degrees
Score green triball 1 (15P)
broadcast NoWaitShoot
Drive for 1.5 foams. Forward? true Wait? true
Drive for 0.7 foams. Forward? false Wait? true
turn left for 45 degrees
turn right for 20 degrees
Score green triball 2 (20P)
broadcast NoWaitShoot
turn right for 25 degrees
Drive for 1.3 foams. Forward? true Wait? true
Drive for 1.1 foams. Forward? false Wait? true
turn right for 63 degrees
Score green triball 3 (25P)
broadcast NoWaitShoot
turn left for 60 degrees
Drive for 1.3 foams. Forward? true Wait? true

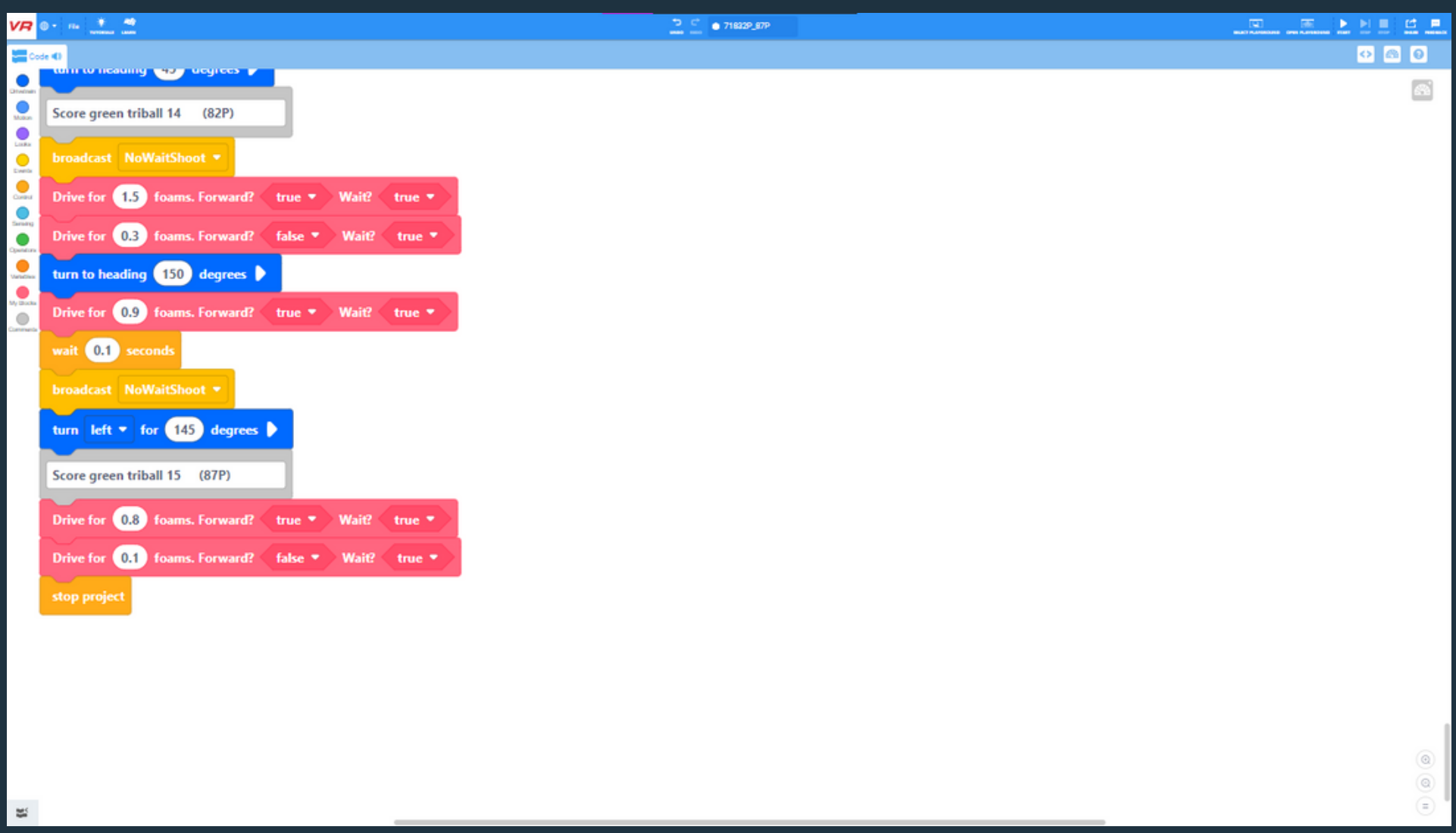

Drive for 1.1 foams. Forward? false Wait? true
turn right for 77 degrees
Drive for 0.75 foams. Forward? true Wait? true
Drive for 1.75 foams. Forward? false Wait? true
turn left for 72 degrees
Score green triball 4 (30P)
broadcast NoWaitShoot
Drive for 1.2 foams. Forward? true Wait? true
Drive for 1.2 foams. Forward? false Wait? true
turn left for 60 degrees
turn right for 55 degrees
Score green triball 5 (35P)
broadcast NoWaitShoot
Drive for 1.4 foams. Forward? true Wait? true
Drive for 1.4 foams. Forward? false Wait? true
turn left for 80 degrees
Drive for 0.8 foams. Forward? true Wait? true
turn right for 100 degrees


Score green triball 6 (40P)
broadcast NoWaitShoot
Drive for 1.3 foams. Forward? true Wait? true
turn to rotation 45 degrees
Drive for 2 foams. Forward? true Wait? true
Score green triball 7 (45P)
broadcast NoWaitShoot
turn right for 115 degrees
Drive for 0.8 foams. Forward? true Wait? true
Drive for 0.9 foams. Forward? false Wait? true
turn to rotation 272 degrees
Drive for 1.7 foams. Forward? true Wait? true
Drive for 1 foams. Forward? false Wait? true
turn left for 20 degrees
Score green triball 8 (50P)
broadcast NoWaitShoot
turn left for 100 degrees
Drive for 0.9 foams. Forward? true Wait? true

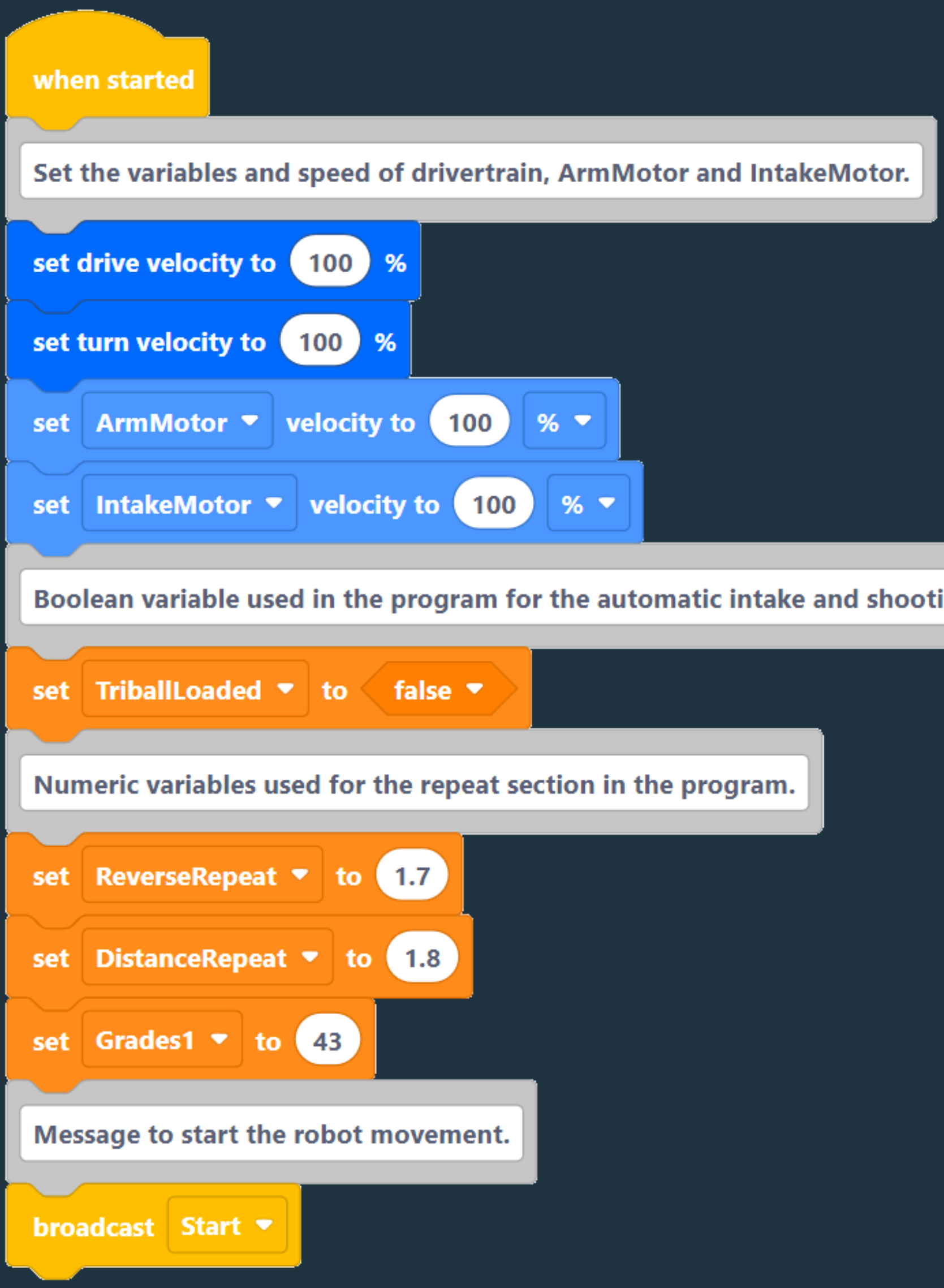
Full Code (3)


Drive for 0.9 foams. Forward? false Wait? true
turn to heading 271 degrees
Drive for 3.8 foams. Forward? true Wait? true
turn right for 45 degrees
spin ArmMotor up for 300 degrees and don't wait
Drive for 0.5 foams. Forward? false Wait? true
turn right for 110 degrees
Score green triball 9 (55P)
broadcast NoWaitShoot
turn right for Grades1 degrees
Repeat section (75P)
repeat 4
Performs a sequence of repeated movements with slight changes in values
Scoring of triballs from LZ1.
Drive for DistanceRepeat foams. Forward? true Wait? true
spin ArmMotor down for 300 degrees and don't wait
Drive for ReverseRepeat foams. Forward? false Wait? true
turn to heading 315 degrees


Drive for 0.7 foams. Forward? true Wait? true
spin ArmMotor up for 300 degrees and don't wait
Drive for 0.5 foams. Forward? false Wait? true
turn right for 110 degrees
broadcast NoWaitShoot
turn right for Grades1 degrees
Change of values of the variables after the sequence of movements
change Grades1 by 3
change ReverseRepeat by 0.05
change DistanceRepeat by 0.1
Drive for DistanceRepeat foams. Forward? true Wait? true
spin ArmMotor down for 300 degrees and don't wait
turn to heading 217 degrees
Drive for 3.9 foams. Forward? true Wait? true
wait 0.2 seconds
turn to heading 91 degrees
Push green triball (77P)
Drive for 3.5 foams. Forward? true Wait?


Score green triball 14 (82P)
broadcast NoWaitShoot
Drive for 1.5 foams. Forward? true Wait? true
Drive for 0.3 foams. Forward? false Wait? true
turn to heading 150 degrees
Drive for 0.9 foams. Forward? true Wait? true
wait 0.1 seconds
broadcast NoWaitShoot
turn left for 145 degrees
Score green triball 15 (87P)
Drive for 0.8 foams. Forward? true Wait? true
Drive for 0.1 foams. Forward? false Wait? true
stop project

Initial Setup

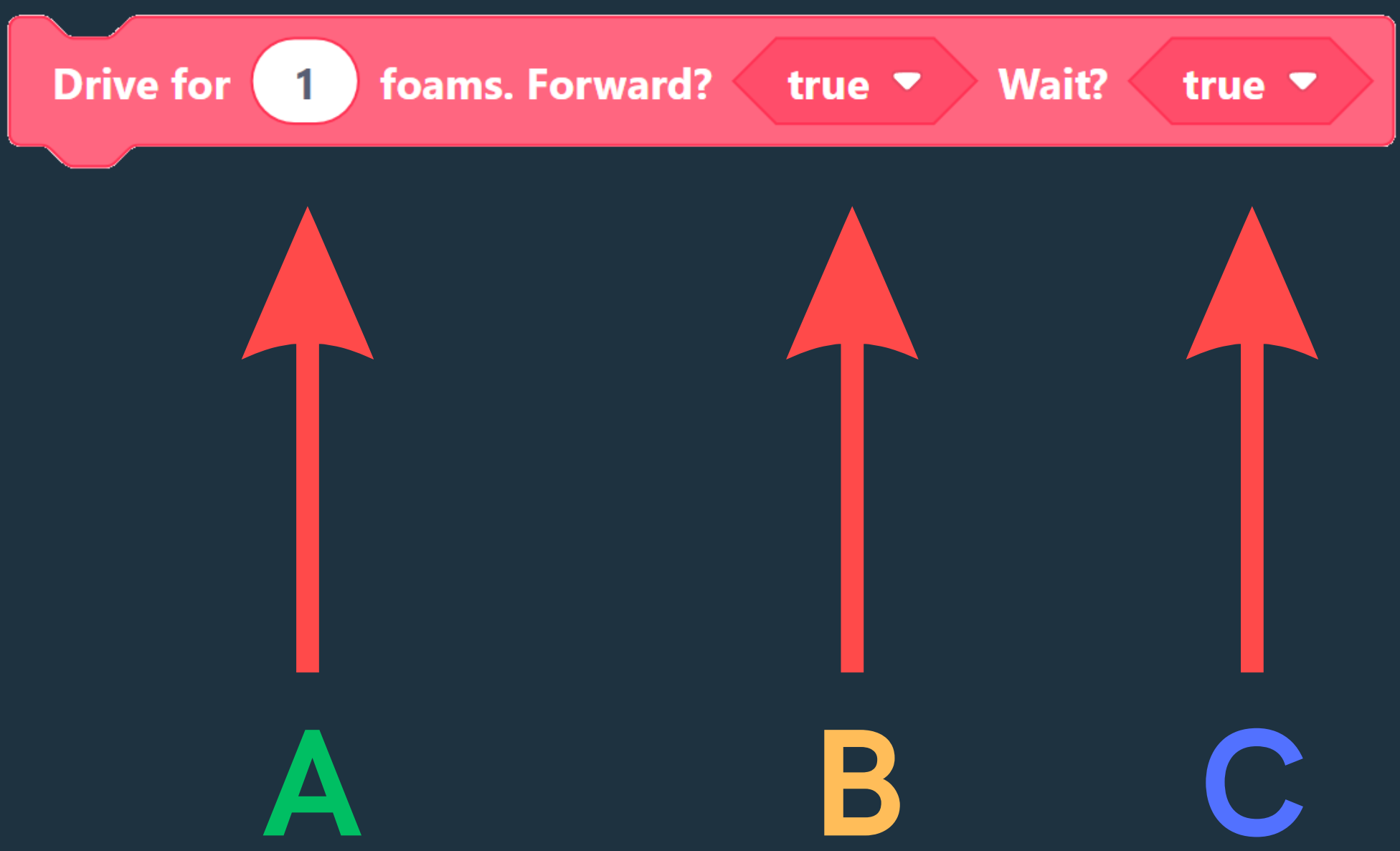


These blocks are used to set all speeds, variables, and configurations before the robot and its mechanisms are set in motion to avoid bugs. We use the message block as a signal to trigger various simultaneous processes, such as movement and the automatic system. This does not cause a delay for the robot to start moving.

Foams as Measuring System

Milimiters? Inches? Nah.. We use a custom block for the robot to handle distances in foams. In this way, we have a more visual division of the field that makes it easier for us to calculate distances when programming. This block works by numerical and boolean values, as well as having all the capabilities of a normal driving block. It's a simple, effective, and functional way to make our robot move.

How the block works?



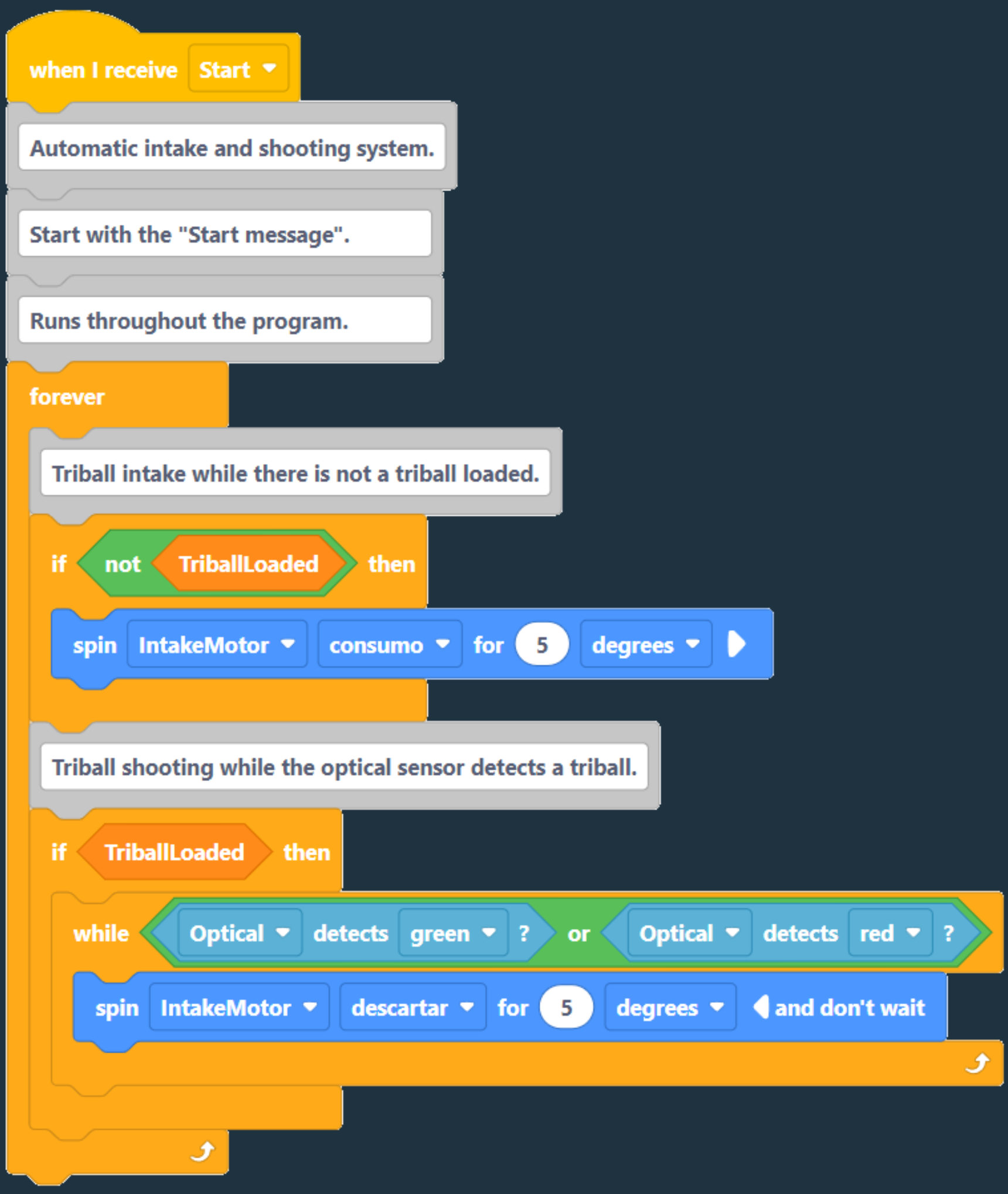
A. Numeric variable used to indicate how many foams the robot will traverse. (Supports decimal quantities)

B. Boolean variable used to indicate whether the robot should drive forward or in reverse. (If true, forward. Else, reverse)

C. Boolean variable used to indicate whether the block should finish its process before moving on to the next one. (If true, wait for the next block)

In the example the robot will drive forward for 1 foam, and will wait.

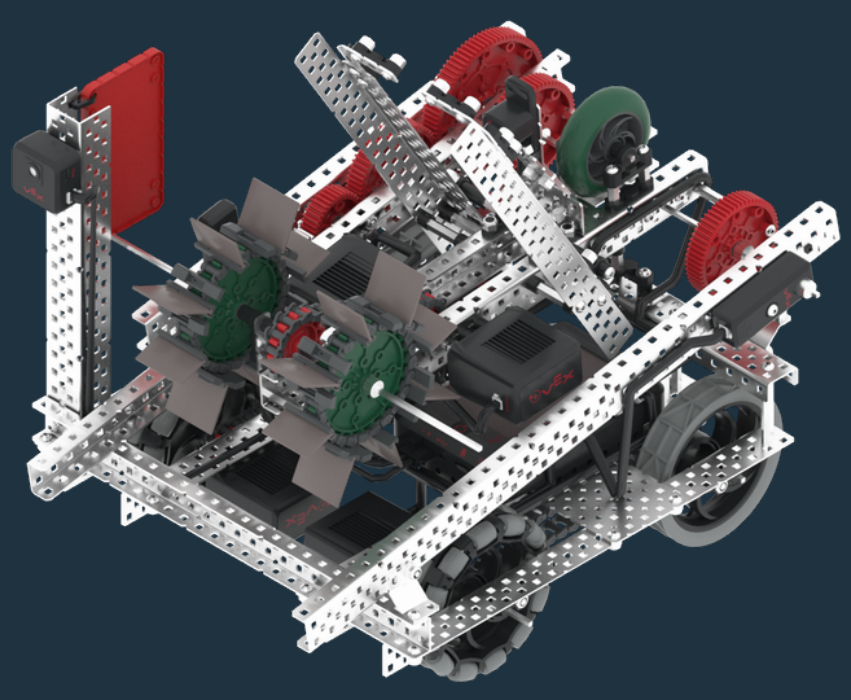
Automatic Intake and Shooting system



This automatic system is activated by the message block “Start” and works forever using two conditionals. It works this way: if the “Triball Loaded” variable is false, the IntakeMotor will start consuming. But if this variable is set to True, and while the optical sensor is detecting a green or red object (Triball), the IntakeMotor will start discarding, droping or shooting triballs. Also this allow our program to run correctly, because when we use the intake block some bugs may occur making the robot unable to shoot or drop a triball.

Why we should use an automatic Intake system?

- To save blocks in the code
- To not waste time in robot movement
- To make our program cleaner
- To use just one block to shoot or drop.
- To use in a creative way the VEXcode tools we have
- Why not? It’s so cool!

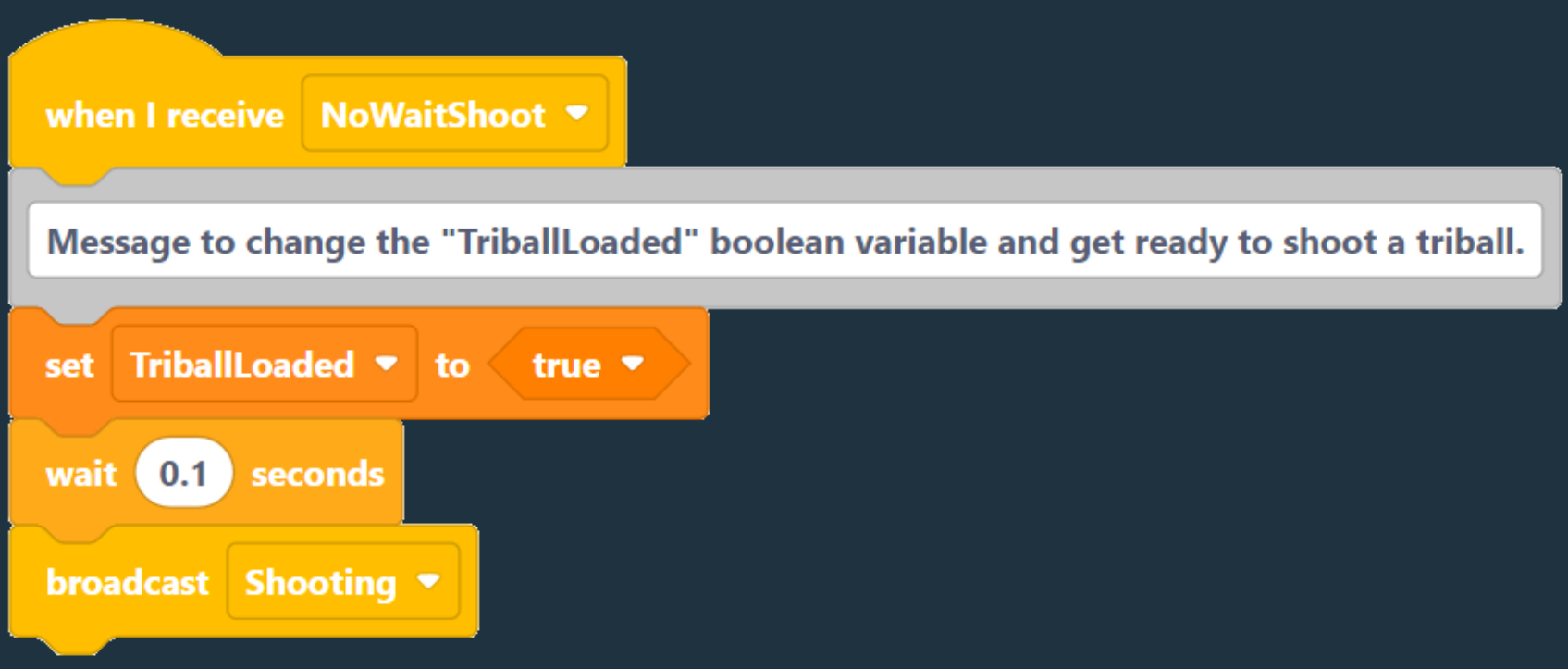


We use 'if' blocks inside a 'forever' block to keep the system active throughout the entire program. The optical sensor is used to stop discarding when it no longer detects triballs. Why? Because the shooting process works more accurately if the IntakeMotor stops once the optical sensor no longer detects the triball. This is because, if it doesn't stop, it will overpropel the shoot of the triball and may go above the goal.

Shooting System

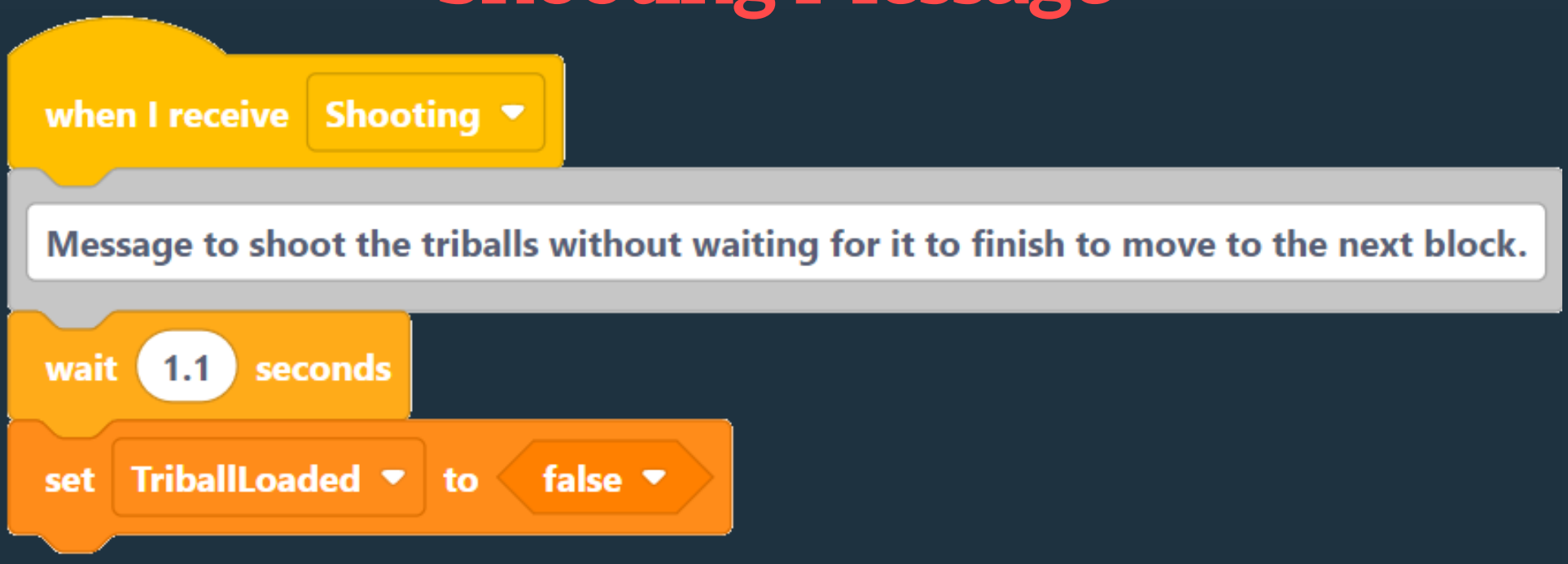
We decided to use messages blocks instead of custom blocks, so that the shoot occurs simultaneously with the robot's movement, avoiding the need for blocks to finish their processes before moving on to the next, in order to shoot while the robot is moving.

NoWaitShoot Message



When this message is received, the “TriballLoaded” variable is set to true, causing the robot to start discarding the triball. Then, the “Shooting” message is broadcast.

Shooting Message



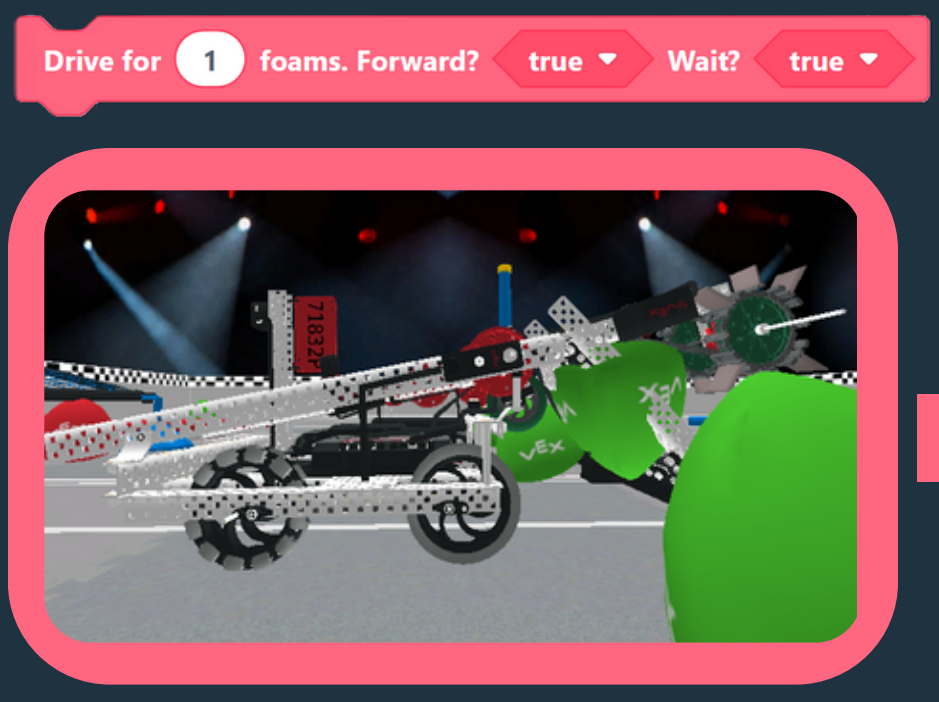
When this message is received, it waits for the necessary time for a triball to be shot with power and sets the variable back to false. After, the automatic Intake system is reactivated.

How to shoot a triball?

To make the robot shoot the triball you need to use the NoWaitShoot message, but you can use more blocks to shoot in diferent ways.

1st block

Start shooting message

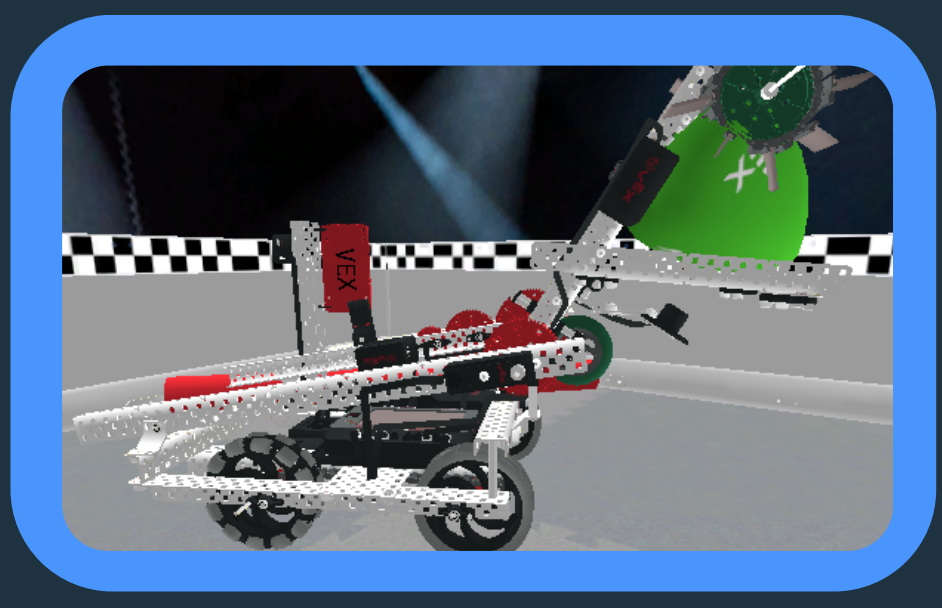


2nd block

Start movement (shoot farther)

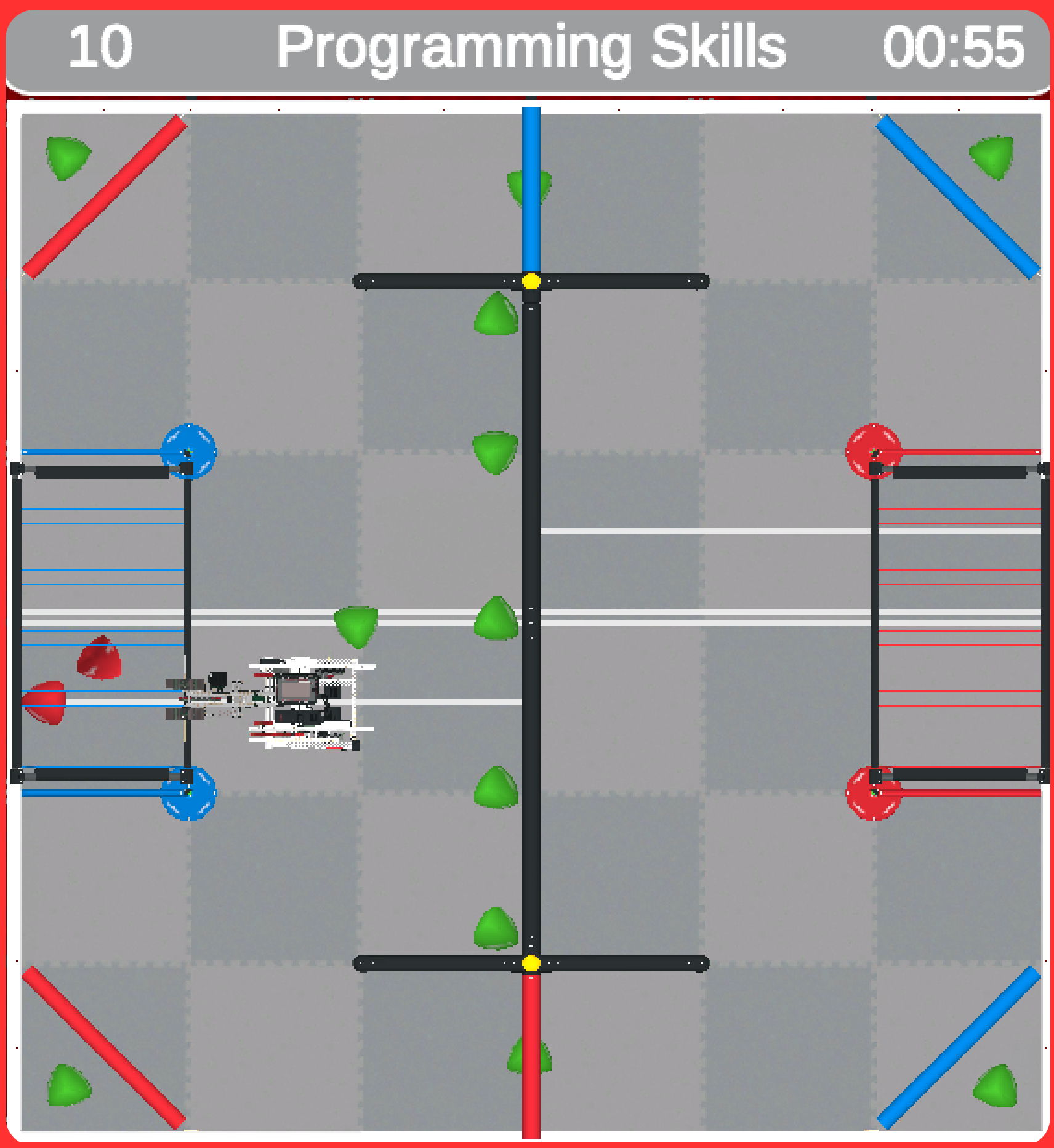
3rd block

Raise your arm (shoot even farther)



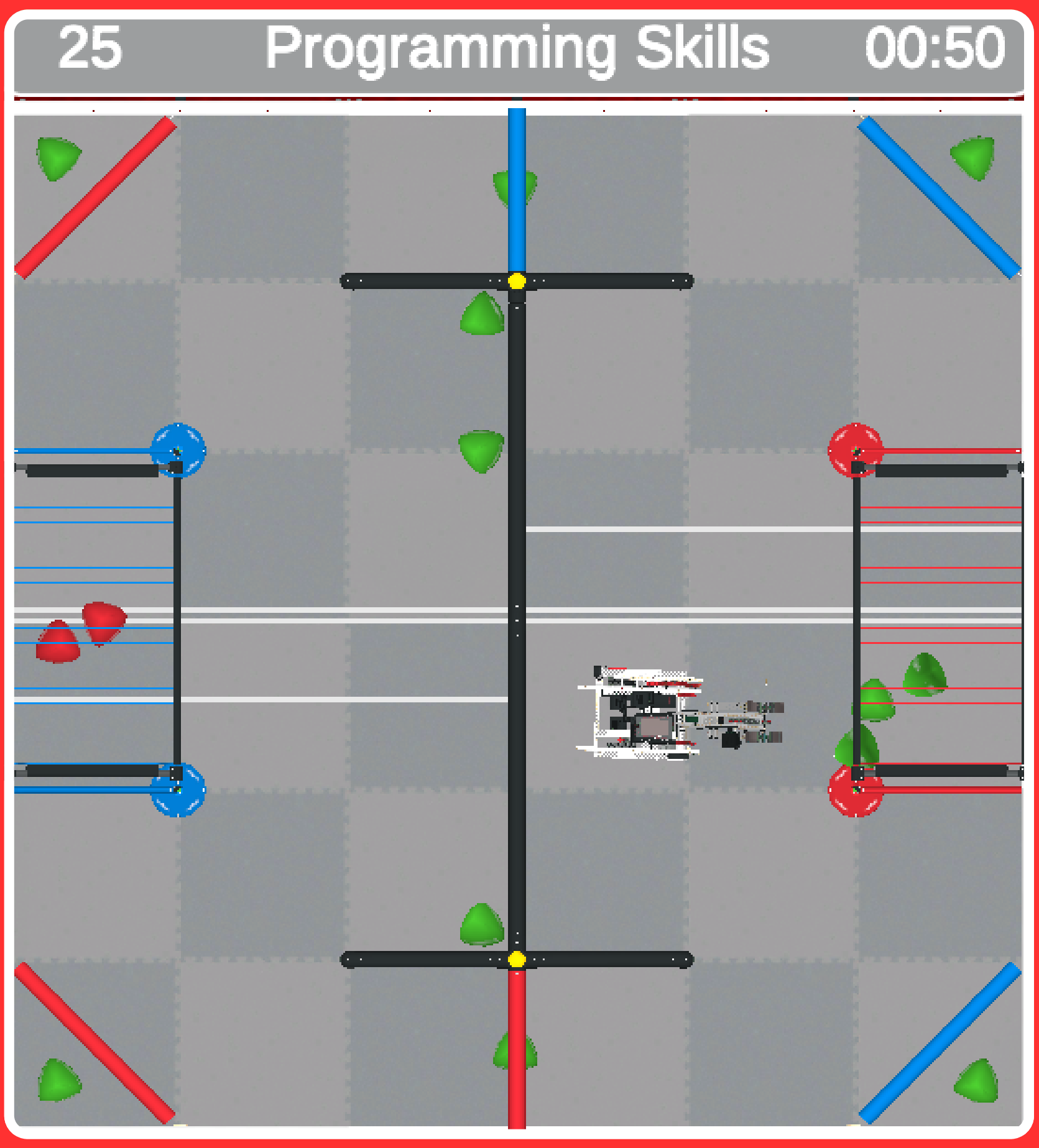
The following section is the main part of the program, where the robot executes movements to score points. To keep the order, the code attached in this document is divided into several parts in an organized and sequential way. All the parts togheter is our functional code.

Scoring 2 preloads (1st part)



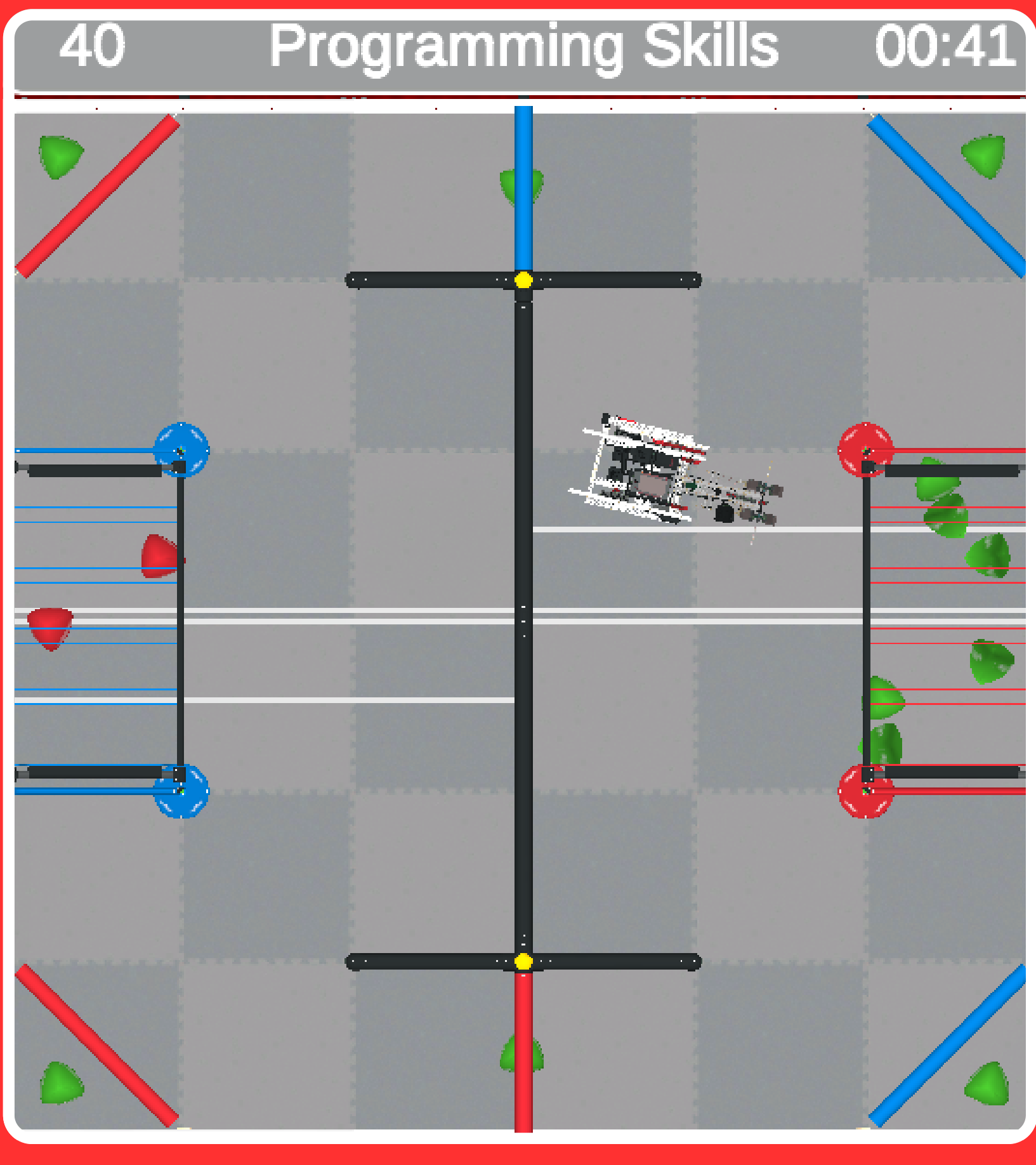
The robot scored the pre-load triballs on the blue goal. (The exception of the automatic system before is just to drop the first pre-load faster)

Scoring 3 triballs (2nd part)



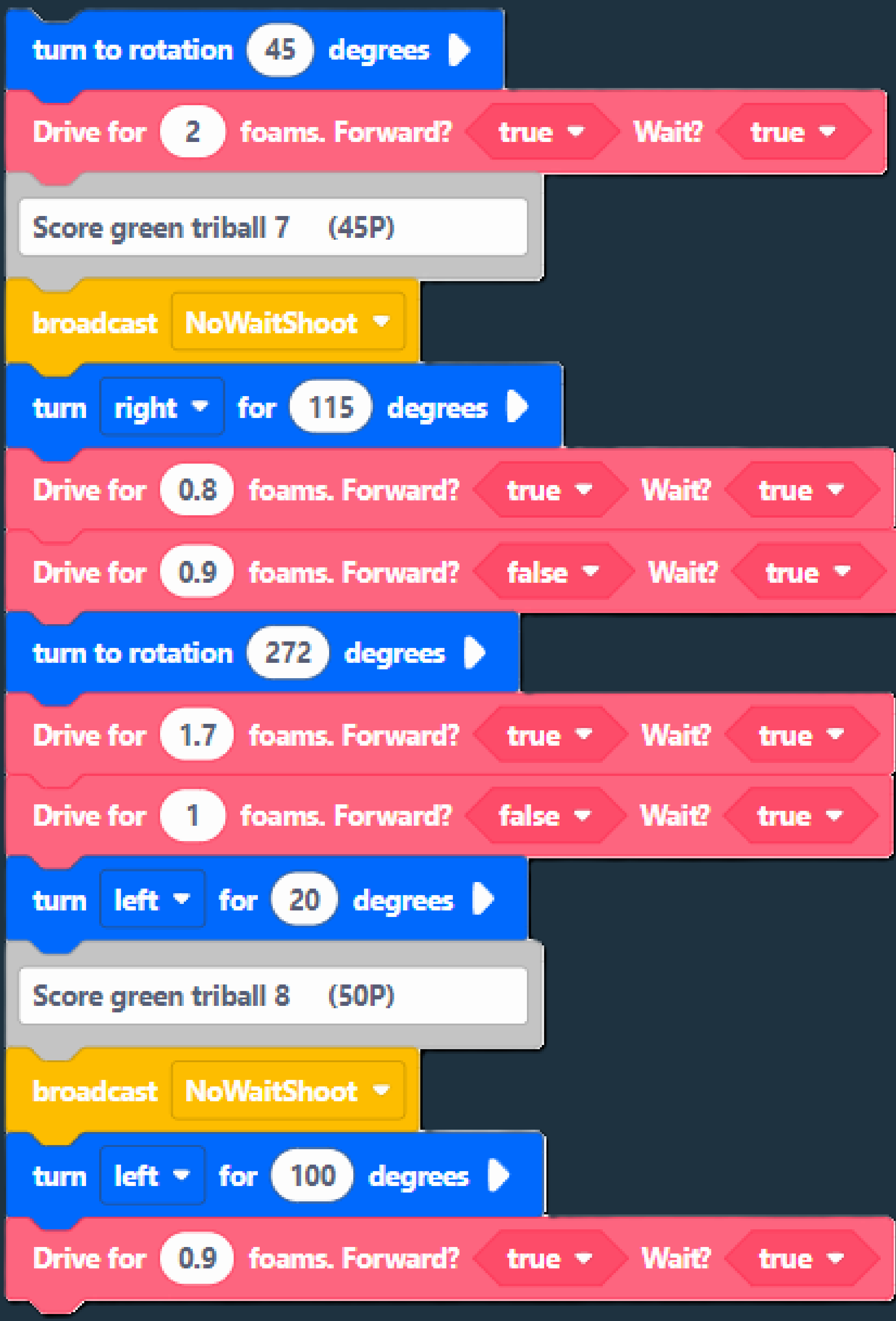
The robot took 3 green triballs and shot them on the goal.

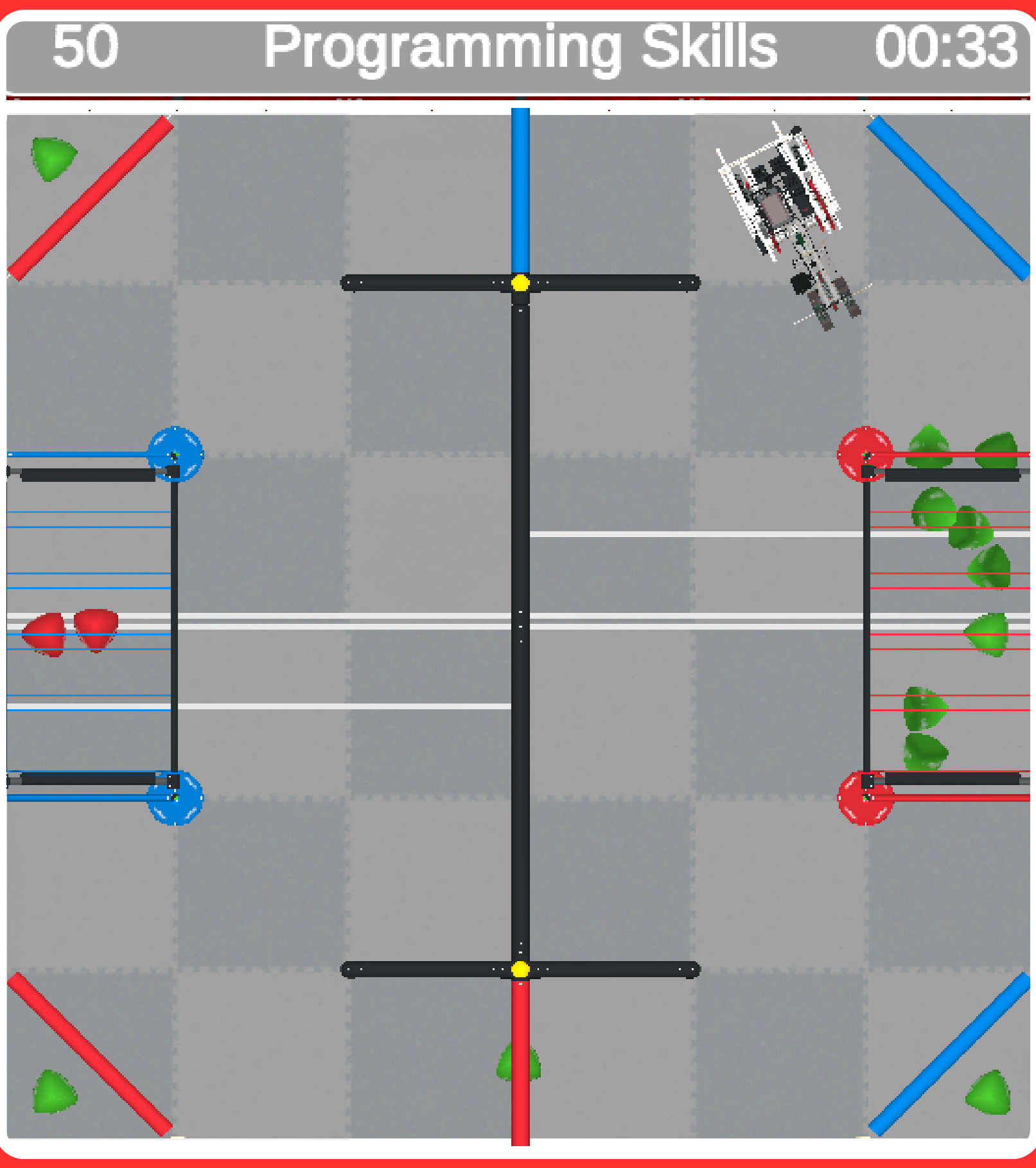
Scoring 3 triballs (3rd part)



The robot took the 3 triballs left on the barrier and headed to the other side of the field scoring them.

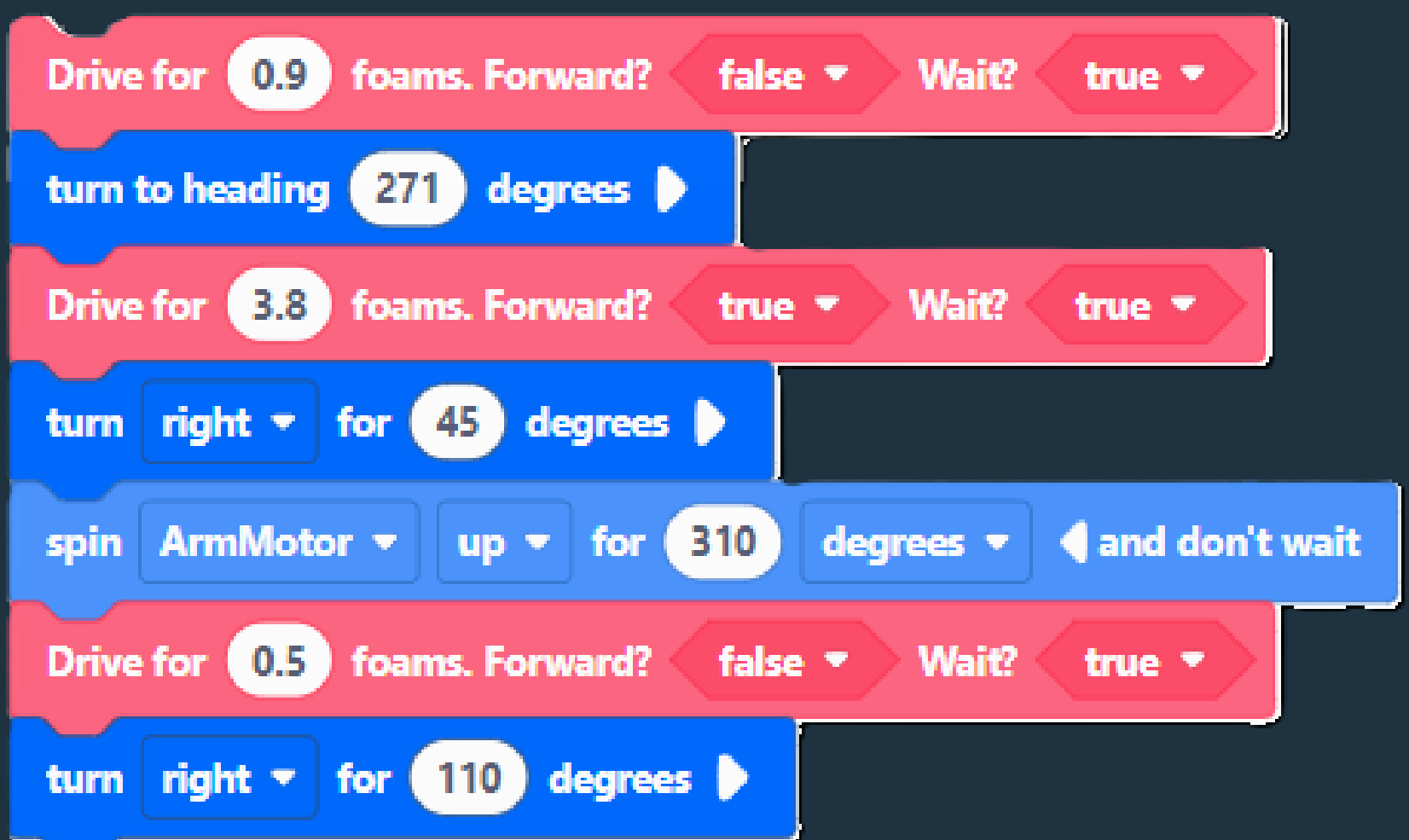
Scoring 2 triballs (4th part)

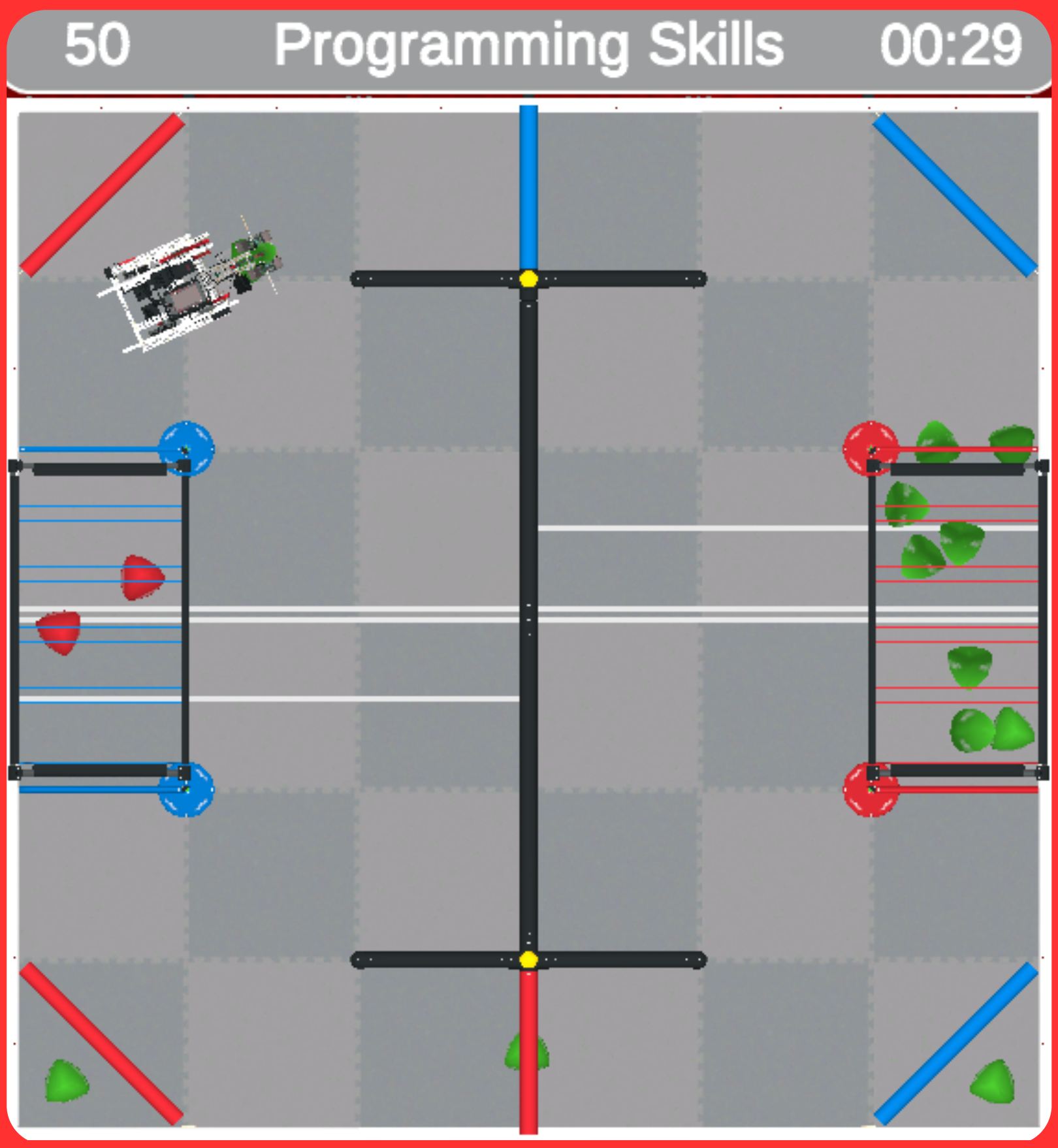




The robot took and scored the triballs on the blue load zone and under the blue bar.

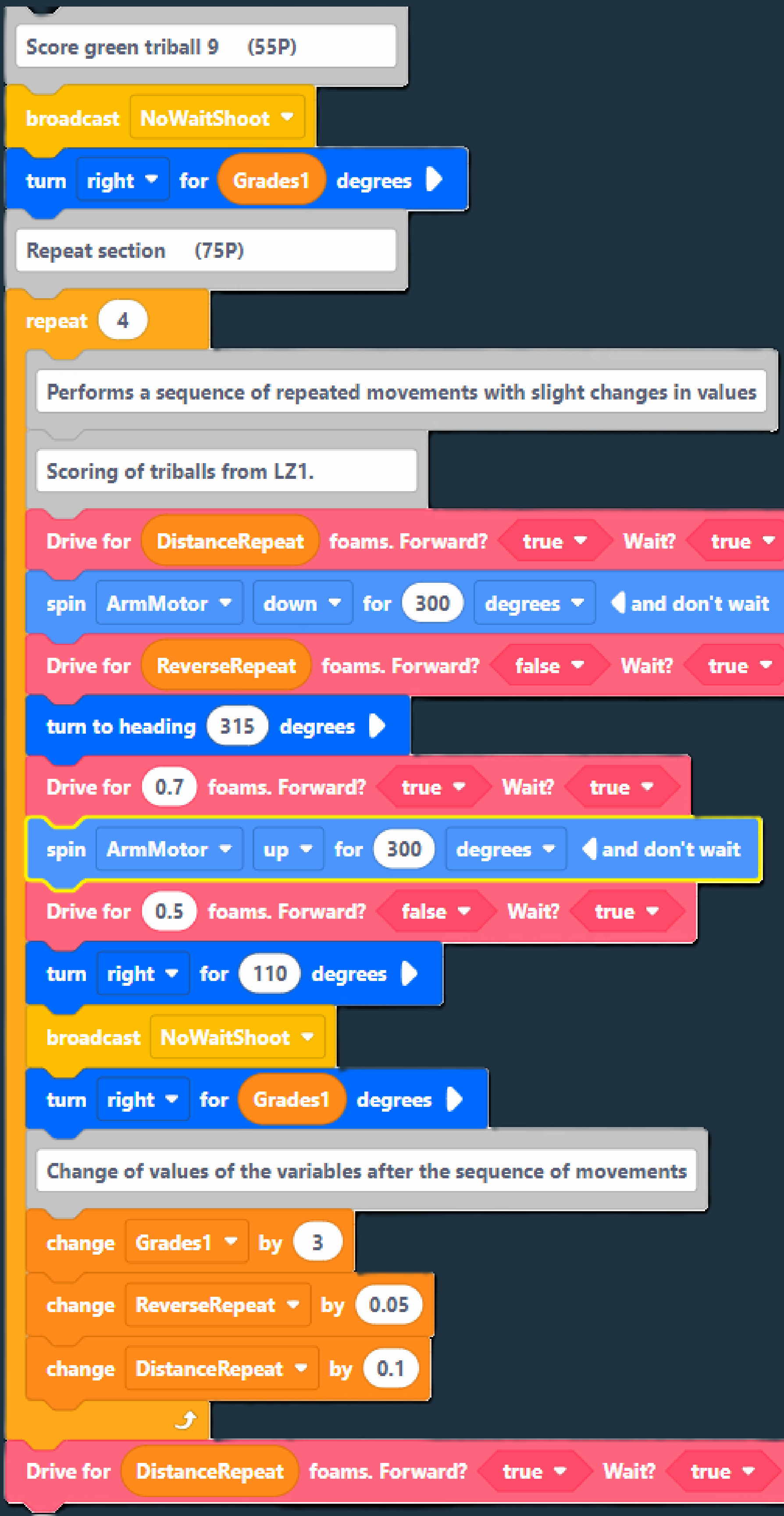
Taking 1 triball before repeat (5th part)

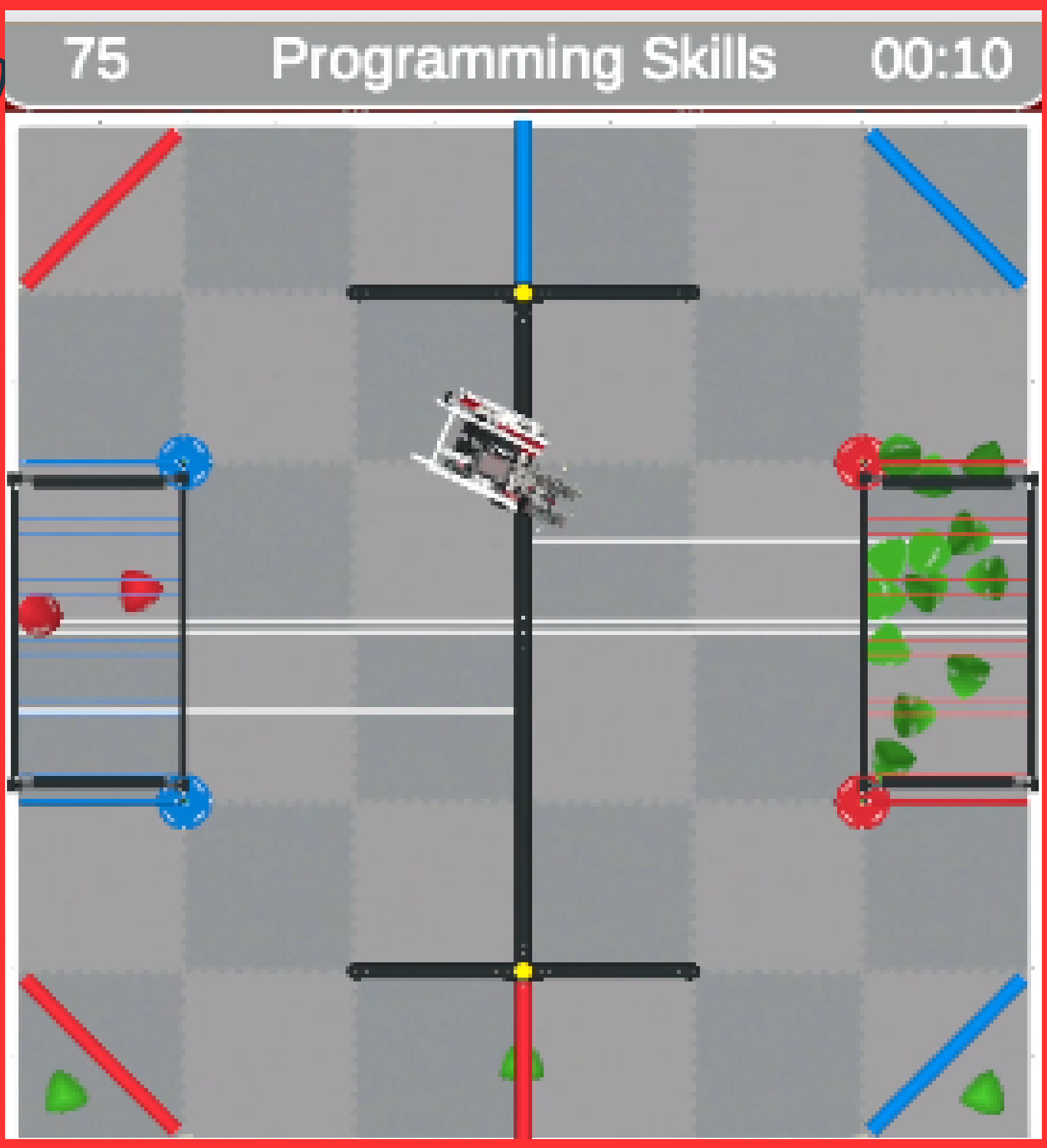




The robot headed to the LoadZone1 and is getting ready to the Repeat section. Here on the repeat section the arm raises to shoot farther.

Repeat and score 5 triballs (6th part)

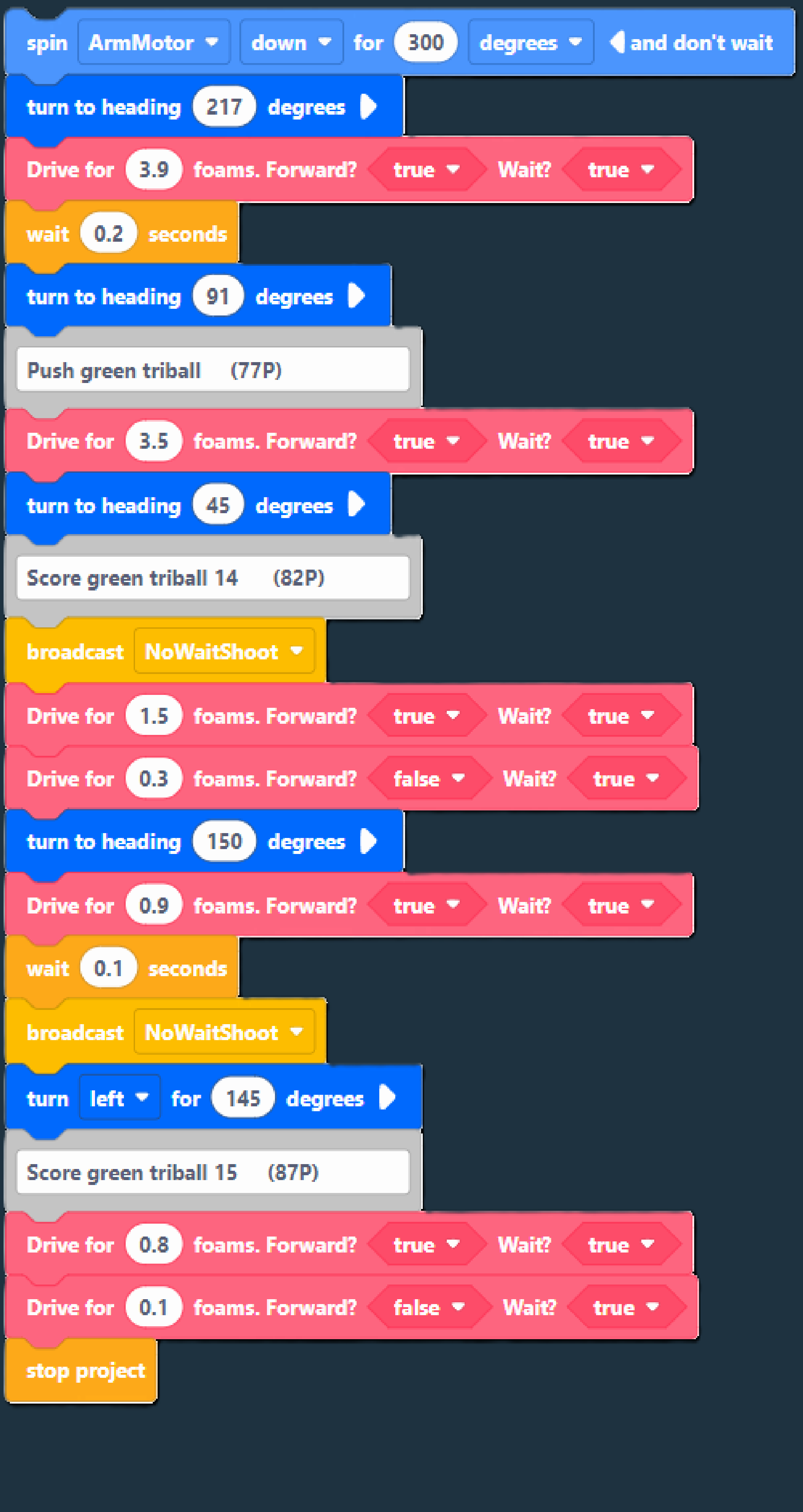


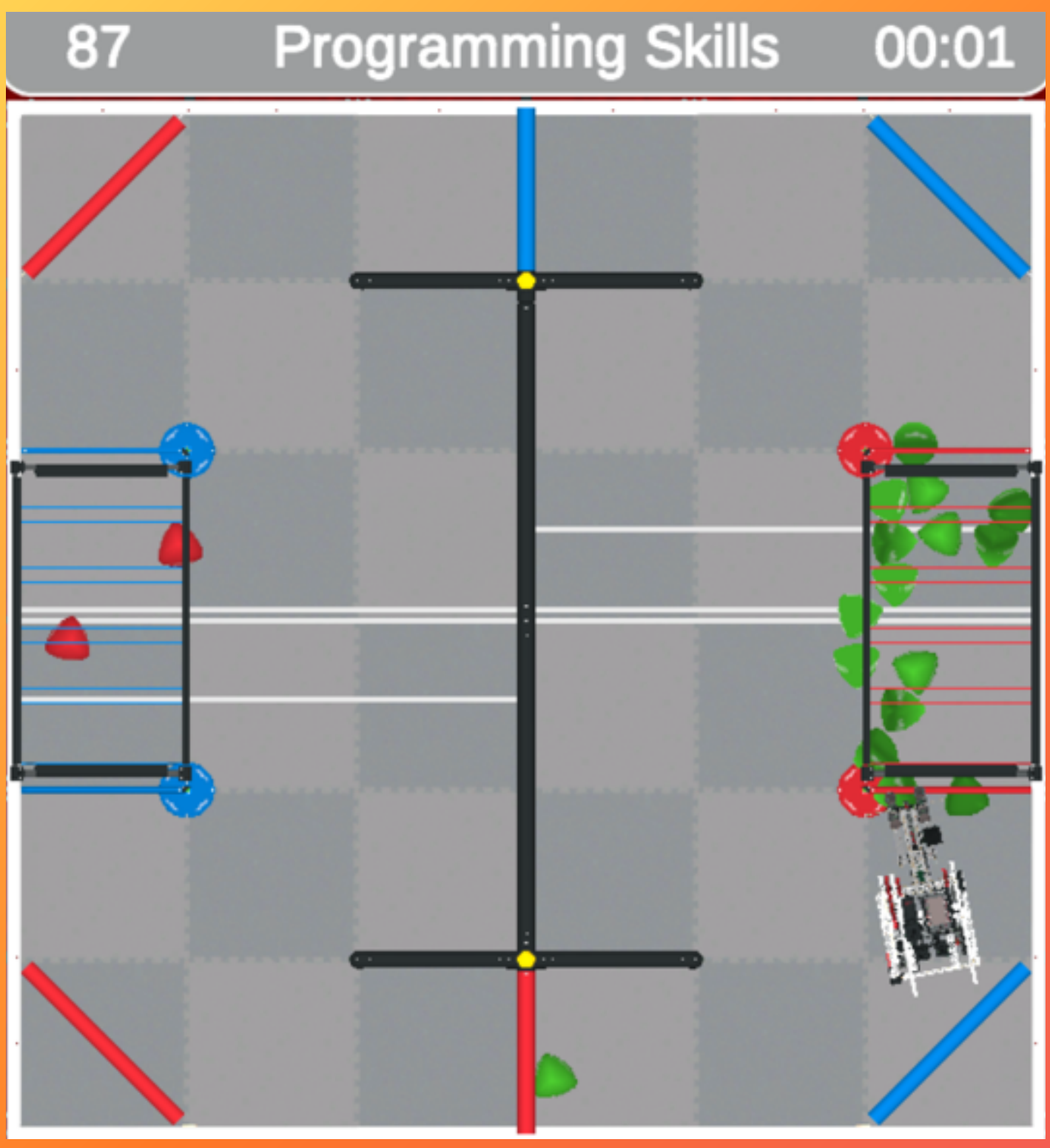


The robot loaded 5 triballs from the LoadZone1 and scored them.

Our code makes the robot shoot triballs at the goal with style. We use a set of blocks in a loop, doing it four times for five awesome shots side by side. The trick is in tweaking variables for the robot's angle and distance each time the blocks are repeated, so the triballs smoothly enter the goal without crashing or hanging in the air.

Scoring 3 triballs (7th part)





The robot score 2 triballs and push 1 to the red offensive zone, finishing the program with 1 second left and 87 points.

All the hours of programming and work have brought us to this point, and this is consequently the culmination of our project. We are very proud of what we have achieved as a team and delighted to have participated in this challenge. Before officially closing our work, we want to share the significance and the lessons we have gained from this experience...

Our team chose the VEXcode VR online challenge to enhance our coding and problem-solving skills. This challenge allows us to delve into the world of programming and strengthen our competencies in STEM areas.

From the outset, we were determined to succeed. Drawing from our previous years' experience, employing techniques we had used, and with time and dedication, we managed to delve into, understand, and apply the knogledge we adquire, for example: custom blocks, boolean variables, complex control blocks, etc.

Along the way, we discovered that the use of sensors can provide greater precision in our codes, whether intended for a physical or virtual robot. It's also important to highlight the efficiency of conditionals and how their proper implementation is essential for effective code.

The acquired knowledge is of great value in competitions, offering us insight into the importance not only of the effectiveness in using sensors and variables but also the significance of creativity, perseverance, and problem-solving ability.

Our solutions can be a great asset to our community. We aim to introduce othre people to this robotics program and have them become part of our community, the VEX community, and share the same passion for technology as we do.