VEXcode VR Skills Challenge - Sensory Programming

Author
Ishaan

Co-Authors
Nayan, Shravan

Editors
Vu-Lam

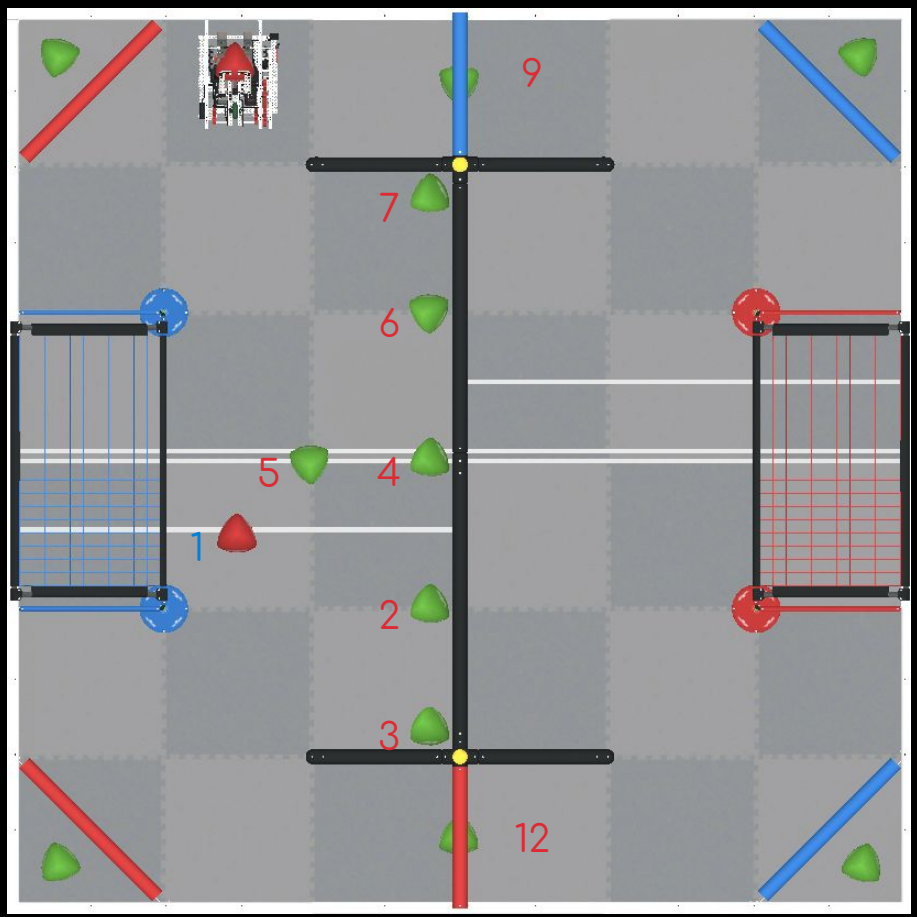
Team
Spectra - 1816X

Location
St. Albans, UK

Contents

Introduction to our VR Skills Programme

Our team, 1816X, has been very successful in this year's VR Skills for Over Under, currently ranking 21st in the world in Middle School with 70 points.



In our programme, we first collect and score the tri-balls in the numerical order as shown above, with the red tri-balls being scored in the blue goal and the other ones in the red. Then we go to the matchload zone in the top left, and shoot two triballs towards the other side. Then we collect the remaining tri-balls and score it into the red goal before time runs out.

Setting all the Variables

We have split our programming routine into 3 parts, so we can easily navigate our way around the code and make adjustments to separate parts of the code.

```
# Setting all events and variables
scoreFieldTriballs = Event()
useMatchloads = Event()
scoreAllianceTriball = Event()
numberOfMatchloads = 0
```

This variable is to keep track of how many times we have used a tri-ball from the matchload stations.

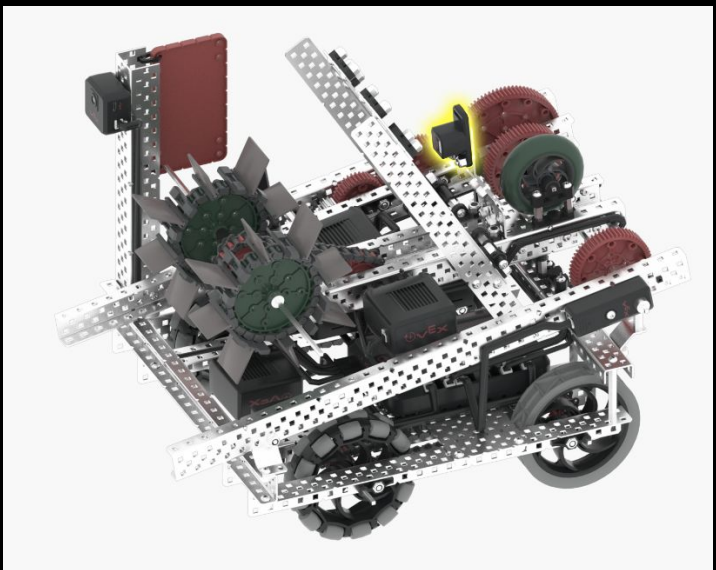
Creating the Functions

In the programme, we have utilised the sensory readings of two sensors in 3 different functions, one for driving towards the matchload bar and two for spinning the arm up or down.

```
def driveToMatchloadBar():
    global scoreAllianceTriball, message1, scoreFieldTriballs, useMatchloads, numberOfMatchloads
    drivetrain.drive(FORWARD)
    while not optical.is_near_object():
        wait(5, MSEC)
    drivetrain.stop()
    numberOfMatchloads = numberOfMatchloads + 1
    brain.screen.print(numberOfMatchloads)
```

We analysed where the optical sensor was on the hero bot, and saw how it is placed behind the intake such that it can identify whether a tri-ball is there or not. It can also be used to detect the matchload bar.

We can utilise the drive forward command as it may be at a varied distance to the matchload bar. This function guarantees the fact that the robot will be in a position to intake the tri-ball from the corner.



We can constantly update the number of times we have went to the matchload bar.

Creating the Functions 2

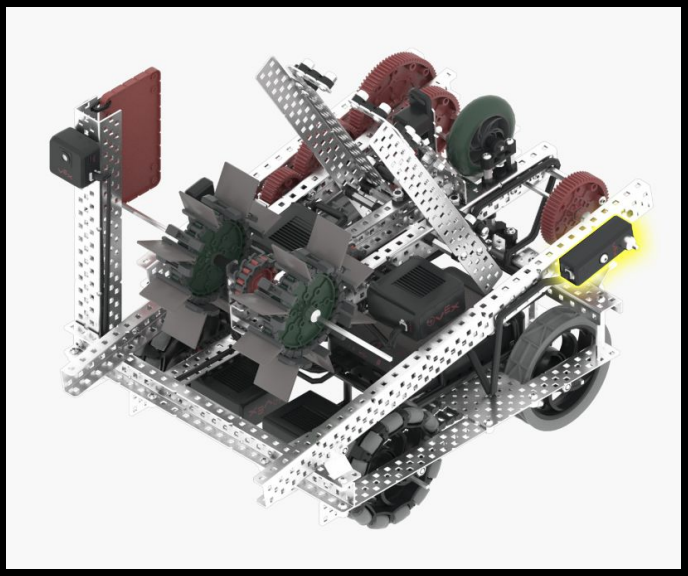
```
def spinArmUp(degrees):
    global scoreAllianceTriball, message1, scoreFieldTriballs, useMatchloads, numberOfMatchloads
    arm_motor.spin_for(REVERSE, (degrees * 7.14), DEGREES, wait=False)

def spinArmDown(degrees):
    global scoreAllianceTriball, message1, scoreFieldTriballs, useMatchloads, numberOfMatchloads
    arm_motor.spin_for(FORWARD, (degrees * 7.14), DEGREES, wait=False)
```

We have used two separate functions for spinning the arm so that it could be easier to differentiate when it is moving up and down.

We have used a local variable of 'degrees' to see how much the arm motor will spin and which position it will be in.

We found out that the maximum the Arm Motor span in degrees is 1200. We also found out that the maximum the rotation sensor span was 168 degrees. Since the rotation sensor's rotations were much more accurate to how much the Arm Motor actually moved, we decided that it would be better to use those values. However, we could not just place those values in the spin for command as it was not the same. So we found out that the ratio of the rotation sensor to the motor's values are 1 : 7.14. We multiplied this to get our desired value of rotation.



Setting all of the Velocities and Values

It is essential that we set velocities for all of the motors that we use as it changes many factors such as how quick we can score points.

```
# Setting all values and velocities before starting
def main():
    global scoreFieldTriballs, useMatchloads, scoreAllianceTriball, numberOfMatchloads
    numberOfMatchloads = 0
    rotation.set_position(0, DEGREES)
    arm_motor.set_velocity(100, PERCENT)
    intake_motor.set_velocity(100, PERCENT)
    drivetrain.set_heading(0, DEGREES)
    drivetrain.set_drive_velocity(100, PERCENT)
    drivetrain.set_turn_velocity(100, PERCENT)
    scoreAllianceTriball.broadcast()
```

Towards the end of the programme, my position and heading varies due to the fact that we have used lots of conditional logic. This means that we can go to specific headings at the end and our programme can still be accurate.

This signals that we can go to the next part of the programme.

It is essential for the intake and the arm motor to be at 100% velocity as it means that it can shoot and move at the fastest rates. In our programme, speed is of the most important thing and we need everything to be fast.

Scoring the Alliance Tri-balls

In this part of the programme, we score our alliance preload and the second alliance tri-ball into the blue goal. This is because we can score them in any goal and it would be more efficient if it was in the blue goal.

```
def onScoreAllianceTriball():
    global scoreFieldTriballs, useMatchloads, scoreAllianceTriball, numberOfMatchloads
    arm_motor.spin(FORWARD)
    drivetrain.drive_for(FORWARD, 1100, MM)
    drivetrain.turn_for(RIGHT, 45, DEGREES)
    wait(0.38, SECONDS)
    intake_motor.spin(REVERSE)
    wait(0.57, SECONDS)
    drivetrain.turn_for(LEFT, 45, DEGREES)
    intake_motor.spin(FORWARD)
    drivetrain.drive_for(FORWARD, 650, MM)
    intake_motor.spin(REVERSE)
    wait(0.3, SECONDS)
    drivetrain.turn_for(RIGHT, 90, DEGREES)
    drivetrain.turn_for(LEFT, 90, DEGREES)
    drivetrain.drive_for(FORWARD, 200, MM)
    drivetrain.turn_for(LEFT, 90, DEGREES)
    scoreFieldTriballs.broadcast()
```

This signals that we can go to the next part of the programme.

We can easily score both triballs in around 5 seconds.

We hard-coded the start as we knew our starting position and it is very easy to get the exact values you need.

Scoring the Tri-balls on the Field

In this part of the programme, we score all of the tri-balls which are on the field.

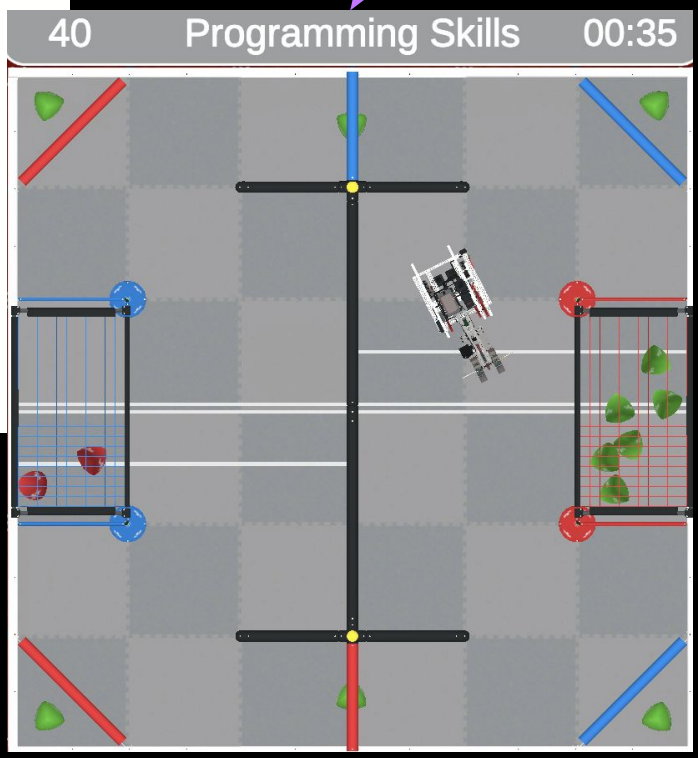
```
def onScoreFieldTriballs():
    global scoreFieldTriballs, useMatchloads, scoreAllianceTriball, numberOfMatchloads
    intake_motor.spin(FORWARD)
    drivetrain.drive_for(FORWARD, 1350, MM)
    intake_motor.spin(REVERSE)
    drivetrain.turn_for(LEFT, 15, DEGREES)
    wait(0.5, SECONDS)
    drivetrain.turn_for(RIGHT, 150, DEGREES)
    drivetrain.drive_for(FORWARD, 240, MM)
    intake_motor.spin(FORWARD)
    drivetrain.drive_for(FORWARD, 295, MM)
    drivetrain.drive_for(REVERSE, 450, MM)
    intake_motor.spin(REVERSE)
    drivetrain.turn_for(LEFT, 150, DEGREES)
    drivetrain.drive_for(FORWARD, 240, MM)
    drivetrain.drive_for(REVERSE, 240, MM)
    drivetrain.turn_for(RIGHT, 105, DEGREES)
    intake_motor.spin(FORWARD)
    drivetrain.drive_for(REVERSE, 500, MM)
    drivetrain.turn_for(RIGHT, 90, DEGREES)
    drivetrain.drive_for(FORWARD, 250, MM)
    drivetrain.drive_for(REVERSE, 140, MM)
    intake_motor.spin(REVERSE)
    drivetrain.turn_for(RIGHT, 180, DEGREES)
    drivetrain.drive_for(FORWARD, 300, MM)
    wait(0.3, SECONDS)
    intake_motor.spin(FORWARD)
    drivetrain.turn_for(RIGHT, 180, DEGREES)
    drivetrain.drive_for(FORWARD, 900, MM)
    drivetrain.turn_for(RIGHT, 180, DEGREES)
    drivetrain.drive_for(FORWARD, 400, MM)
    intake_motor.spin(REVERSE)
    drivetrain.drive_for(FORWARD, 520, MM)
    drivetrain.drive_for(REVERSE, 400, MM)
    drivetrain.turn_for(LEFT, 15, DEGREES)
    drivetrain.drive_for(FORWARD, 700, MM)
    drivetrain.turn_for(LEFT, 90, DEGREES)
    drivetrain.drive_for(FORWARD, 420, MM)
    intake_motor.spin(FORWARD)
    drivetrain.drive_for(FORWARD, 200, MM)
    drivetrain.turn_for(LEFT, 230, DEGREES)
    intake_motor.spin(REVERSE)
    drivetrain.drive_for(FORWARD, 650, MM)
    drivetrain.turn_for(LEFT, 130, DEGREES)
    intake_motor.spin(FORWARD)
    drivetrain.drive_for(FORWARD, 400, MM)
    drivetrain.turn_for(LEFT, 55, DEGREES)
    drivetrain.drive_for(FORWARD, 550, MM)
    drivetrain.drive_for(REVERSE, 50, MM)
    drivetrain.turn_for(RIGHT, 180, DEGREES)
    intake_motor.spin(REVERSE)
    drivetrain.drive_for(FORWARD, 600, MM)
    drivetrain.drive_for(REVERSE, 600, MM)
    drivetrain.turn_for(RIGHT, 20, DEGREES)
    drivetrain.drive_for(FORWARD, 495, MM)
    drivetrain.turn_for(RIGHT, 139, DEGREES)
    useMatchloads.broadcast()
```

Intaking the Tri-balls into the Robot

Scoring Tri-balls into the Goal

We decided to hard code this entire section because all of the tri-balls had predetermined positions.

This means that we have scored 6 tri-balls into the red goal and 2 alliance tri-balls into the blue goal, equating to 40 points. This has been achieved in 25 seconds.



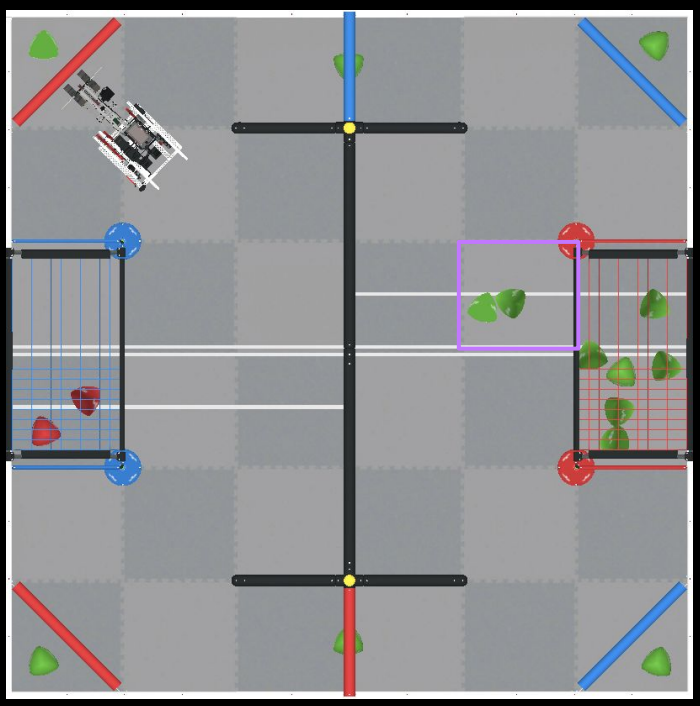
This signals that we can go to the next part of the programme.

Using the Matchloads and Scoring

In this part of the programme, we utilise the matchloads and we use our functions to score the highest number of points.

```
def onUseMatchloads():
    global scoreFieldTriballs, useMatchloads, scoreAllianceTriball, numberOfMatchloads
    intake_motor.spin(FORWARD)
    drivetrain.drive_for(FORWARD, 1100, MM)
    drivetrain.turn_for(RIGHT, 45, DEGREES)
    driveToMatchloadBar()
    drivetrain.drive_for(REVERSE, 700, MM, wait=False)
    spinArmUp(160)
    wait(1.7, SECONDS)
    drivetrain.turn_for(LEFT, 25, DEGREES)
    intake_motor.spin(REVERSE)
    drivetrain.drive_for(REVERSE, 550, MM)
    wait(0.5, SECONDS)
    drivetrain.drive_for(FORWARD, 550, MM)
    intake_motor.spin(FORWARD)
    drivetrain.turn_for(RIGHT, 25, DEGREES)
    spinArmDown(160)
    wait(1.7, SECONDS)
    driveToMatchloadBar()
    drivetrain.drive_for(REVERSE, 700, MM, wait=False)
    spinArmUp(160)
    wait(2, SECONDS)
    drivetrain.turn_for(LEFT, 25, DEGREES)
    intake_motor.spin(REVERSE)
    drivetrain.drive_for(REVERSE, 550, MM)
    wait(0.5, SECONDS)
    spinArmDown(160)
```

One cycle of matchloading



We have utilised two cycles of matchloads, this is because this cycle is very time-taking, with one cycle taking approximately 5 seconds. The matchloads consistently land in the tile which has been highlighted. This allows us to later score them when we get to that side of the field.

Using the Matchloads and Scoring 2

```
intake_motor.spin(FORWARD)
drivetrain.drive_for(FORWARD, 550, MM)
drivetrain.turn_for(RIGHT, 25, DEGREES)
drivetrain.drive_for(FORWARD, 500, MM)
drivetrain.turn_for(RIGHT, 90, DEGREES)
drivetrain.drive_for(FORWARD, 350, MM)
drivetrain.turn_for(RIGHT, 45, DEGREES)
drivetrain.drive_for(FORWARD, 1500, MM)
drivetrain.turn_for(RIGHT, 5, DEGREES)
drivetrain.drive_for(FORWARD, 300, MM)
drivetrain.turn_for(RIGHT, 45, DEGREES)
intake_motor.spin(REVERSE)
drivetrain.drive_for(FORWARD, 400, MM)
while not not optical.is_near_object():
    wait(5, MSEC)
wait(0.2, SECONDS)
drivetrain.turn_for(LEFT, 90, DEGREES)
intake_motor.spin(FORWARD)
driveToMatchloadBar()
drivetrain.turn_for(RIGHT, 125, DEGREES)
intake_motor.spin(REVERSE)
drivetrain.drive_for(FORWARD, 250, MM)
while not not optical.is_near_object():
    wait(5, MSEC)
wait(0.2, SECONDS)
drivetrain.turn_for(RIGHT, 65, DEGREES)
drivetrain.drive_for(FORWARD, 400, MM)
intake_motor.spin(FORWARD)
drivetrain.turn_for(LEFT, 43, DEGREES)
drivetrain.drive(FORWARD)
while not optical.is_near_object():
    wait(5, MSEC)
drivetrain.stop()
drivetrain.drive_for(FORWARD, 220, MM)
drivetrain.turn_for(LEFT, 90, DEGREES)
intake_motor.spin(REVERSE)
wait(0.5, SECONDS)
```

Navigating the field while intaking a tri-ball and getting ready to shoot it.

Waiting until I have fully released the tri-ball before getting the tri-ball from the corner.

Driving towards the matchloaded tri-balls and trying to score them.

The different positions are visible. These normally affect results.

The tri-balls that I had previously matchloaded, while consistently being in that one tile, still had different placements. The better these placements are, the more points I score.

Using the Matchloads and Scoring 3

The remainder of this programme ends it by scoring two more tri-balls into the goal. One is from the matchload section and one is from underneath the elevation bar.

```
drivetrain.turn_to_heading(0, DEGREES)
drivetrain.drive_for(FORWARD, 1260, MM)
intake_motor.spin(FORWARD)
drivetrain.turn_to_heading(-45, DEGREES)
driveToMatchloadBar()
drivetrain.drive_for(REVERSE, 200, MM)
drivetrain.turn_to_heading(-150, DEGREES)
intake_motor.spin(REVERSE)
drivetrain.drive_for(FORWARD, 200, MM)
wait(0.3, SECONDS)
drivetrain.turn_to_heading(45, DEGREES)
drivetrain.drive_for(FORWARD, 600, MM)
drivetrain.turn_to_heading(90, DEGREES)
intake_motor.spin(FORWARD)
drivetrain.drive_for(FORWARD, 635, MM)
drivetrain.drive_for(REVERSE, 600, MM)
drivetrain.turn_to_heading(180, DEGREES)
drivetrain.drive_for(FORWARD, 1500, MM)
drivetrain.turn_for(RIGHT, 90, DEGREES)
intake_motor.spin(REVERSE)
```

I have used the turn to heading command here because the conditional logic prior to this does not guarantee the heading the robot will be in.

On our best run, we managed 70 points, where every single tri-ball ended up being scored for 5 points.