Titantron - 38535A - 2024

VEX VR Online Challenge

Caspar Chen, Matthew Yam

Team # 38535A

Glenview, Illinois

```
#region VEXcode Generated Robot Configuration
import math
import random
from vexcode_vrc import *
from vexcode_vrc.events import get_Task_func

# Brain should be defined by default
brain=Brain()

drivetrain = Drivetrain("drivetrain", 0)
arm_motor = Motor("ArmMotor", 3)
rotation = Rotation("Rotation", 7)
intake_motor = Motor("IntakeMotor", 8)
optical = Optical("Optical", 11)
gps = GPS("GPS", 20)

#endregion VEXcode Generated Robot Configuration

#
------------------------------------------------------------------
------------
#
#    Starting Location:  B
#    Starting Direction: West
#    Robot Preload:      Yes
#    Preload Location:   6
#
#
------------------------------------------------------------------
------------
armFetch = 1300
armLaunch = 175
```

```
def setup():
    drivetrain.set_heading(-90,DEGREES) #targetAngle() won't work
without this
    drivetrain.set_drive_velocity(100, PERCENT) #Set all motors
to 100 percent
    drivetrain.set_turn_velocity(100,PERCENT)
    arm_motor.set_velocity(100, PERCENT)
    intake_motor.set_velocity(100, PERCENT)
    arm_motor.spin_to_position(armFetch, DEGREES, wait=False)
#Arm motor to position for intaking balls
def hitAndRun(): #Backs up into the ball for 2 points
    startPos = (-897,1435)
    drivetrain.drive_for(REVERSE,200,MM)
    goToTarget(startPos)
def preloads():
    goalAngle = -90
    goalPos = (-897,0)
    goToTarget(goalPos) #Scores robot preload
    drivetrain.turn_to_heading(goalAngle,DEGREES)
    release()
    fetch(clockwise=False) #Grabs second preload and scores
    drivetrain.turn_to_heading(goalAngle,DEGREES)
    release()
def doubleScore(): #Intakes 2 green balls, goes to goal on other
side, scores
    intake_motor.spin(FORWARD)
    drivetrain.set_drive_velocity(80,PERCENT)
    goToTarget((1000, 0))
    drivetrain.set_drive_velocity(100,PERCENT)
    release()
def barballs():
    ballCoords = [(-100,-1100),(-125,600),(-125,1075)]
```

```
    goal = (1000,0)
    for x in ballCoords: #Goes to every coordinate and scores
        goToTarget(x,getBall=True)
        goToTarget(goal)
        release()
    lastOne = (-100,-600) #Launches last ball along bar while
going to bottom left corner
    launchPoint = (-200,-200)
    goToTarget(lastOne,getBall=True)
    launchSet()
    goToTarget(launchPoint)
    drivetrain.drive_for(REVERSE,100,MM)
    launch()
def finalBall(): #Bottom right corner; last ball scored in
program
    arm_motor.spin_to_position(1300,DEGREES,wait=False)
    goToTarget((1600,-1600),getBall=True)
    goToTarget((1500,-800))
    release()
    drivetrain.drive_for(REVERSE,200,MM)


def targetAngle(coords):
    targetX = coords[0]
    targetY = coords[1]
    xDiff = targetX-gps.x_position(MM)
    yDiff = targetY-gps.y_position(MM)
    if xDiff != 0: #Avoid division by zero
        angle = abs(math.degrees(math.atan(yDiff/xDiff)))
#Absolute value of the reverse tangent of coordinates, converted
from radians to degrees
```

```
        if yDiff>=0 and xDiff>0: #The heading depends on where
the ball is relative to the robot
            turnAngle = 90-angle
        elif yDiff>=0 and xDiff<0:
            turnAngle = -90+angle
        elif yDiff<=0 and xDiff<0:
            turnAngle = -90-angle
        elif yDiff<=0 and xDiff>0:
            turnAngle = 90+angle
        return turnAngle
    elif yDiff>0: #If ball is directly above or directly below
the robot
        return 0
    elif yDiff<0:
        return 180
    else:
        return gps.heading()
def targetDistance(coords): #Distance formula to find how far
robot needs to go
    targetX = coords[0]
    targetY = coords[1]
    distance =
math.sqrt(pow(targetX-gps.x_position(MM),2)+pow(targetY-gps.y_po
sition(MM),2))
    return distance
def goToTarget(coords, getBall=False):
    drivetrain.turn_to_heading(targetAngle(coords),DEGREES) #Use
target angle to turn to ball
    if getBall: #Accounts for arm length to grab ball
        armsReach = 300
        intake_motor.spin(FORWARD)
```

```
        distance = targetDistance(coords)-armsReach #Target
distance drives to ball
        drivetrain.drive_for(FORWARD,distance, MM)
    else:
        distance = targetDistance(coords)
        drivetrain.drive_for(FORWARD,distance, MM)
def reverseTarget(coords,getBall=False): #Reverse version of
goToTarget()
    drivetrain.turn_to_heading(targetAngle(coords)+180,DEGREES)
#Turns another 180 degrees
    if getBall:
        armsReach = 300
        intake_motor.spin(FORWARD)
        distance = targetDistance(coords)-armsReach
    else:
        distance = targetDistance(coords)
    drivetrain.drive_for(REVERSE,distance, MM)

def fetch(clockwise): #Spins around to find ball
    intake_motor.spin(FORWARD)
    degreesTurned = 0 #Stops spinning if robot has turn 360
degrees
    if clockwise: #Parameter to spin clockwise or
counterclockwise
        while (not optical.is_near_object()) and (not
degreesTurned>=360): #Stops spinning when object detected with
optical sensor
            drivetrain.turn_for(RIGHT,1,DEGREES,wait=False)
            degreesTurned+=1
    else:
        while (not optical.is_near_object()) and (not
degreesTurned>=360):
```

```
            drivetrain.turn_for(LEFT,1,DEGREES,wait=False)
            degreesTurned+=1

def release(): #Function for shooting the ball
    intake_motor.spin(REVERSE)
    wait(0.8,SECONDS)

def launching():
    global launchAngle
    launchAngle = -81
    hit = False #Bumps ball on the bottom first for 2 points
    for x in range(5): #Launches 5 balls
        reloadBalls()
        if hit == False:
            reverseTarget((0,-1500))
            drivetrain.drive_for(FORWARD,400,MM)
            hit = True
        launchSet()
        launch()
def reloadBalls(): #Goes to bottom left corner and gets a ball
    intake_motor.spin(FORWARD)
    arm_motor.spin_to_position(armFetch,DEGREES,wait=False)
    cornerCoords = (-1600,-1600)
    goToTarget(cornerCoords,getBall=True)
    wait(0.2,SECONDS)
def launchSet(): #Positions arm motor for launching and drives
backwards towards launch position
    arm_motor.spin_to_position(armLaunch,DEGREES,wait=False)
    launchPos = (-100,-200)
    reverseTarget(launchPos)
launchAngle = -90
def launch(): #Releases ball
```

```
    global launchAngle
    drivetrain.turn_to_heading(launchAngle,DEGREES)
    launchAngle-=8 #Decrements shooting angle to prevents balls
blocking each other
    intake_motor.spin(REVERSE)
    wait(0.8,SECONDS)

def main(): #Functions divide entire program into parts
    setup()
    hitAndRun()
    preloads()
    doubleScore()
    barballs()
    launching()
    finalBall()

    stop_project()

vr_thread(main)
```