Student Name: Kim

Assignment: Manual Rover

Notes: does the manual stuff and checks for color

Project Name: VEXcode Project

Project Type: Python

Date: Fri Jun 16 2023

```
#region VEXcode Generated Robot Configuration
from vex import *
import urandom

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

# Robot configuration code
motor_group_3_motor_a = Motor(Ports.PORT3, GearSetting.RATIO_18_1, True)
motor_group_3_motor_b = Motor(Ports.PORT4, GearSetting.RATIO_18_1, False)
motor_group_3 = MotorGroup(motor_group_3_motor_a, motor_group_3_motor_b)
optical_2 = Optical(Ports.PORT2)
controller_1 = Controller(PRIMARY)


# wait for rotation sensor to fully initialize
wait(30, MSEC)


def play_vexcode_sound(sound_name):
    # Helper to make playing sounds from the V5 in VEXcode easier and
    # keeps the code cleaner by making it clear what is happening.
    print("VEXPlaySound:" + sound_name)
    wait(5, MSEC)

# add a small delay to make sure we don't print in the middle of the REPL header
wait(200, MSEC)
# clear the console to make sure we don't have the REPL in the console
print("\033[2J")



# define variables used for controlling motors based on controller inputs
controller_1_up_down_buttons_control_motors_stopped = True

# define a task that will handle monitoring inputs from controller_1
def rc_auto_loop_function_controller_1():
    global controller_1_up_down_buttons_control_motors_stopped, remote_control_cod
e_enabled
    # process the controller input every 20 milliseconds
    # update the motors based on the input values
    while True:
        if remote_control_code_enabled:
            # check the buttonUp/buttonDown status
            # to control motor_group_3
            if controller_1.buttonUp.pressing():
                motor_group_3.spin(FORWARD)
                controller_1_up_down_buttons_control_motors_stopped = False
            elif controller_1.buttonDown.pressing():
                motor_group_3.spin(REVERSE)
                controller_1_up_down_buttons_control_motors_stopped = False
            elif not controller_1_up_down_buttons_control_motors_stopped:
                motor_group_3.stop()
                # set the toggle so that we don't constantly tell the motor to sto
p when
```

```
                    # the buttons are released
                    controller_1_up_down_buttons_control_motors_stopped = True
            # wait before repeating the process
            wait(20, MSEC)

    # define variable for remote controller enable/disable
    remote_control_code_enabled = True

    rc_auto_loop_thread_controller_1 = Thread(rc_auto_loop_function_controller_1)

    #endregion VEXcode Generated Robot Configuration

    # ------------------------------------------
    #
    #   Project:      rover thingy
    #   Author:       group
    #   Created:      TODAY
    #   Description:  we made rover it's cool and awsome
    #
    # ------------------------------------------

    # Library imports
    from vex import *

    # Begin project code

    #make variables HERE
    blue = 0
    orange = 0
    brown = 0
    green = 0
    gray = 0
    red = 0
    line = 1

    #functions start HERE

    optical_2.set_light_power(50)
    brain.screen.print("Now operational! ^_^")
    def testdrive(): # this function was just a test and does nothing, isn't called an
  d is just here as a reminder for where we started
        #motor_group_3.spin(REVERSE)
        wait(250)
        #motor_group_3.stop()
        newlinemsg("finished 2 second drive...")

    #testdrive()

    def constantcheck(): # this function also does nothing and was used for debug to c
  onstantly check the output underneath our sensor
        newlinemsg("now checking for color")

        while True:
            wait(1000)
```

```
106             colornum = checkcolor(optical_2.hue(), optical_2.brightness(), True)
107             if not colornum == 0:
108                 newlinemsg(colornum)
109
110
111
112     def checkcolor(chue, cbright, increment): # this method compares the hue and brigh
    tness outputs to detect which color it sees
113         global blue, orange, brown, green, gray, red
114         if (chue <= 20 and chue >= 16) and (cbright <= 12 and cbright >= 10): # this i
    s red
115             if increment: red += 1
116             return ("red " + str(red))
117         elif (chue <= 27 and chue >= 21) and (cbright <= 20 and cbright >= 11): # thi
    s is orange
118             if increment: orange += 1
119             return ("orange " + str(orange))
120         elif (chue <= 41 and chue >= 34) and (cbright <= 10 and cbright >= 5): # this
    is brown
121             if increment: brown += 1
122             return ("brown " + str(brown))
123         elif (chue <= 48 and chue >= 44) and (cbright <= 28 and cbright >= 20): # thi
    s is gray
124             if increment: gray += 1
125             return ("gray " + str(gray))
126         elif (chue <= 92 and chue >= 84) and (cbright <= 8 and cbright >= 7): # this i
    s green
127             if increment: green += 1
128             return ("green " + str(green))
129         elif (chue <= 233 and chue >= 224) and (cbright <= 10 and cbright >= 7): # thi
    s is gray
130             if increment: blue += 1
131             return ("blue " + str(blue))
132         elif (chue < 45 and chue >= 43) and (cbright < 20): # this is the FLOOR
133             newlinemsg("THIS IS THE FLOOR.")
134             return 0
135         else:
136             newlinemsg("no color detected, hue: " + str(chue) + " br: " + str(cbrigh
    t))
137             return 0
138
139     def newlinemsg(text): # this function makes a quick workaround to printing new lin
    es for messages, and erasing old ones
140         global line
141         line += 1
142         if line > 10:
143             line = 1
144             brain.screen.clear_screen()
145         brain.screen.set_cursor(line, 1)
146         brain.screen.print(text)
147
148     #constantcheck()
149
150     def AButtonPress(): # function for when the A button is pressed
151
```

```
152         returntext = checkcolor(optical_2.hue(), optical_2.brightness(), True)
153         if not returntext == 0:
154             newlinemsg(returntext)
155 
156     def BButtonPress(): # function for when the B button is pressed (same as A)
157         returntext = checkcolor(optical_2.hue(), optical_2.brightness(), False)
158         if not returntext == 0:
159             newlinemsg(returntext)
160 
161     def turnside(direction, ontype): # this function is triggered by 4 different call
    s, and each correspond to what the behavior will be
162         if direction:
163             if ontype:
164                 motor_group_3_motor_a.spin(REVERSE)
165                 motor_group_3_motor_b.spin(FORWARD)
166             else:
167                 motor_group_3.stop()
168         else:
169             if ontype:
170                 motor_group_3_motor_a.spin(FORWARD)
171                 motor_group_3_motor_b.spin(REVERSE)
172             else:
173                 motor_group_3.stop()
174 
175     def buttonleftpressed(): # button press that calls the turnside() function with pa
    rameters
176         turnside(True, True)
177 
178     def buttonleftrelease(): # button press that calls the turnside() function with pa
    rameters
179         turnside(True, False)
180 
181     def buttonrightpressed(): # button press that calls the turnside() function with p
    arameters
182         turnside(False, True)
183 
184     def buttonrightrelease(): # button press that calls the turnside() function with p
    arameters
185         turnside(False, False)
186 
187     def buttonycheck(): # the final method, prints all colors and how many we've seen
    to that point, and also shows the total
188         global blue, orange, brown, green, gray, red, line
189         line = 4
190         brain.screen.clear_screen()
191         brain.screen.set_cursor(1, 1)
192         brain.screen.print("red: " + str(red) + "   | orange: " + str(orange))
193         brain.screen.set_cursor(2, 1)
194         brain.screen.print("brown: " + str(brown) + " | gray: " + str(gray))
195         brain.screen.set_cursor(3, 1)
196         brain.screen.print("green: " + str(green) + " | blue: " + str(blue))
197         brain.screen.set_cursor(4, 1)
198         brain.screen.print("total: " + str(red + orange + brown + green + gray + blu
    e))
199 
```

```
200
201     controller_1.buttonA.pressed(AButtonPress)
202     controller_1.buttonB.pressed(BButtonPress)
203     controller_1.buttonLeft.pressed(buttonleftpressed)
204     controller_1.buttonLeft.released(buttonleftrelease)
205     controller_1.buttonRight.pressed(buttonrightpressed)
206     controller_1.buttonRight.released(buttonrightrelease)
207     controller_1.buttonY.pressed(buttonycheck)
```