#!/usr/bin/env pybricks-micropython

from pybricks.robotics import DriveBase

from pybricks.hubs import EV3Brick

from pybricks.ev3devices import Motor

from pybricks.parameters import Port, Direction

from pixycamev3.pixy2 import Pixy2, MainFeatures

from pybricks.messaging import BluetoothMailboxServer, TextMailbox

from time import sleep

def main():

    motor\_a = Motor(Port.A, positive\_direction=Direction.CLOCKWISE)

    motor\_d = Motor(Port.D, positive\_direction=Direction.CLOCKWISE)

    drive = DriveBase(motor\_a, motor\_d, wheel\_diameter=42, axle\_track=143)

    cannon = Motor(Port.B, positive\_direction=Direction.CLOCKWISE, gears=[12, 36])

    ev3 = EV3Brick()

    pixy2 = Pixy2(port=1, i2c\_address=0x54)

    data=MainFeatures()

    basic\_speed = 500

    GAIN = 10

    server = BluetoothMailboxServer()

    mbox = TextMailbox('greeting', server)

    next = True

    ev3.screen.print('waiting for connection...')

    server.wait\_for\_connection()

    ev3.screen.print('connected!')

    drive.straight(170)

    drive.turn(180)

    drive.stop()

    while True:

        response = coms(ev3, mbox)

        if response == 'start':

            next = True

            break

        else:

            sleep(1)

    while next:

        pixy2.set\_lamp(1, 0)

        ev3.speaker.say("GO")

        ev3.screen.print("GOING")

        line\_follow(brick=ev3, cam=pixy2,

                    cannon=cannon,

                    data=data,

                    motor\_L=motor\_a,

                    motor\_R=motor\_d,

                    basic\_speed=basic\_speed,

                    GAIN=GAIN,

                    wheel\_base=drive)

        ev3.screen.print("Line follow finished")

        ev3.speaker.say("STOP")

        ev3.screen.print("Stopped")

        pixy2.set\_lamp(0, 0)

        next = False

def shoot(brick, cannon):

    """Takes the cannon motor and rotates based on gearing for one revolution"""

    cannon.run\_target(1000, 360, wait=True)

    cannon.reset\_angle(0)

def line\_follow(brick, cam, cannon, data, motor\_L, motor\_R, basic\_speed, GAIN, wheel\_base):

    """Takes all variables needed for Pixy Cam line follow.  Checks for barcodes for actions.

       wheel\_base.turn activates the DriveBase module, which MUST be turned off by wheel\_base.stop

       in order to go back to independant motor control for line follow."""

    X\_REF=39

    active = True

    good\_count = 0

    bad\_count = 0

    while active:

        if brick.buttons.pressed():

            cam.set\_lamp(0, 0)

        if good\_count + bad\_count == 5:

            q\_stop(motor\_L, motor\_R)

            active = False

        # Get linetracking data from pixy2

        data = cam.get\_linetracking\_data()

        # Process data

        if data.error:

            pass

        else:

            if data.number\_of\_barcodes > 0:

                # Barcode(s) found

                for i in range(0, data.number\_of\_barcodes):

                    if data.barcodes[i].code == 5:

                        q\_stop(motor\_L, motor\_R)

                        brick.speaker.say('sector clear')

                        wheel\_base.turn(180)

                        wheel\_base.stop()

                    elif data.barcodes[i].code==13:

                        q\_stop(motor\_L, motor\_R)

                        brick.speaker.say('enemy spotted')

                        shoot(brick, cannon)

                        brick.speaker.say('bogie down')

                        bad\_count += 1

                        wheel\_base.turn(180)

                        wheel\_base.stop()

                    elif data.barcodes[i].code==14:

                        q\_stop(motor\_L, motor\_R)

                        brick.speaker.say('friendly')

                        good\_count += 1

                        wheel\_base.turn(180)

                        wheel\_base.stop()

        if data.number\_of\_vectors > 0:

            dx = X\_REF - data.vectors[0].x1

            move(motor\_L, motor\_R, basic\_speed, dx, GAIN)

        else:

            # No vector data, stop robot

            q\_stop(motor\_L, motor\_R)

            #wheel\_base.turn(10)

            #wheel\_base.stop()

            #sleep(0.25)

        # Clear data for reading next loop

        data.clear()

def move(motor\_L, motor\_R, basic\_speed, speed\_x, GAIN):

    """Uses individual motors to correct for line follow based on Pixy Cam vector data"""

    speed\_x \*= GAIN

    speed\_L = limit\_speed(basic\_speed - speed\_x)

    speed\_R = limit\_speed(basic\_speed + speed\_x)

    motor\_L.run(speed\_L)

    motor\_R.run(speed\_R)

def limit\_speed(speed):

  """Limit speed in range [-1000,1000]."""

  if speed > 1000:

    speed = 1000

  elif speed < -1000:

    speed = -1000

  return speed

def q\_stop(motor\_L, motor\_R):

    """Stops each motor when driving with independant control for line follow"""

    motor\_L.stop()

    motor\_R.stop()

# Before running this program, make sure the client and server EV3 bricks are

# paired using Bluetooth, but do NOT connect them. The program will take care

# of establishing the connection.

# The server must be started before the client!

def coms(brick, mbox):

    mbox.wait()

    return (mbox.read())

if \_\_name\_\_ == '\_\_main\_\_':

    main()

#!/usr/bin/env pybricks-micropython

# Before running this program, make sure the client and server EV3 bricks are

# paired using Bluetooth, but do NOT connect them. The program will take care

# of establishing the connection.

# The server must be started before the client!

from pybricks.messaging import BluetoothMailboxClient, TextMailbox

from pybricks.hubs import EV3Brick

from time import sleep

from pybricks.ev3devices import Motor, TouchSensor

from pybricks.parameters import Port, Direction

def main():

    SERVER = 'ev3dev' # This is the name of the remote EV3 or PC we are connecting to.

    scorebot = EV3Brick()

    med\_motor = Motor(Port.A, positive\_direction=Direction.CLOCKWISE)

    button = TouchSensor(Port.S1)

    client = BluetoothMailboxClient()

    mbox = TextMailbox('greeting', client)

    scorebot.screen.print('establishing connection...')

    client.connect(SERVER)

    scorebot.screen.print('connected!')

    active = True

    while active:

        if button.pressed():

            scorebot.speaker.say('scanning')

            med\_motor.run(200)

            scorebot.speaker.play\_file('684783\_\_muray\_\_sonar\_830hz\_synthesized.wav')

            med\_motor.stop()

            scorebot.speaker.say('scan complete')

            active = False

    # In this program, the client sends the first message and then waits for the

    # server to reply.

    mbox.send('start')

if \_\_name\_\_ == '\_\_main\_\_':

    main()