```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Mar  7 00:00:39 2018

@author: JasonGrillo
"""

import pyb
import micropython

class Turret_Hub_Task:
    ''' This defines the task function method for a nerf turret hub.
    '''

    def __init__(self, pan_position, tilt_angle, pan_coords, tilt_coords,
        ''' Construct a turret hub task function by initilizing any share
            variables and objects
            @param pan_position The shared variable for the pan position
            @param tilt_angle The shared variable for the tilt position
            @param pan_coords The queue of coordinates for the pan axis
            @param tilt_coords The queue of coordinates for the tilt axis
            @param FEED_BULLETS The shared variable flag for the nerf gun
            @param WINDUP_GUN The shared variable flag for the nerf gun m
        '''
        self.pan_position = pan_position
        self.tilt_angle = tilt_angle
        self.pan_coords = pan_coords
        self.tilt_coords = tilt_coords
        self.FEED_BULLETS = FEED_BULLETS
        self.WINDUP_GUN = WINDUP_GUN

        self.TARGET_CMD = False

        self.CALIBRATION_FLG = False

        self.pan_centroids = [0.0, 0.0, 0.0, 0.0, 0.0]
        self.tilt_centroids = [0.0, 0.0, 0.0, 0.0, 0.0]

# ----------------------------------------------------------------------
# ----------------------------------------------------------------------

    def read_GUI(self):
        ''' Reads the serial port for incoming commands and executes the
        '''
        if self.vcp.any():
```

```
            self.GUI_input = float(self.vcp.read(2).decode('UTF-8'))
            self.GUI_Lookup_Table(self.GUI_input)

# ------------------------------------------------------------------------
# ------------------------------------------------------------------------

    def turret_hub_fun(self):
        ''' Defines the task function method for a turret hub object.
        '''
        self.vcp = pyb.USB_VCP ()

        STATE_0 = micropython.const(0)
        STATE_1 = micropython.const(1)
        STATE_2 = micropython.const(2)
        STATE_3 = micropython.const(3)
        STATE_4 = micropython.const(4)
        STATE_5 = micropython.const(5)
        STATE_6 = micropython.const(6)
        STATE_7 = micropython.const(7)
        STATE_8 = micropython.const(8)
        STATE_9 = micropython.const(9)
        STATE_10 = micropython.const(10)
        STATE_11 = micropython.const(11)

        self.state = STATE_0

        self.pan_coords.put(0)
        self.tilt_coords.put(0)

        while True:

            ##################################

            ## STATE 0: CALIBRATE POINT A
            if self.state == STATE_0:
                self.read_GUI()
                yield (self.state)

                if self.CALIBRATION_FLG:
                    # input location A into pan centroids
                    self.calibrate_point(0, self.pan_position.get(), self
                    self.CALIBRATION_FLG = False
                    self.state = STATE_1

            ##################################
```

```
        ## STATE 1: CALIBRATE POINT B
        elif self.state == STATE_1:
            self.read_GUI()
            yield (self.state)

            if self.CALIBRATION_FLG:
                # input location B into pan centroids
                self.calibrate_point(1, self.pan_position.get(), self
                self.CALIBRATION_FLG = False
                self.state = STATE_2

        ##################################

        ## STATE 2: CALIBRATE POINT C
        elif self.state == STATE_2:
            self.read_GUI()
            yield (self.state)

            if self.CALIBRATION_FLG:
                # input location C into pan centroids
                self.calibrate_point(2, self.pan_position.get(), self
                self.CALIBRATION_FLG = False
                self.state = STATE_3

        ##################################

        ## STATE 3: CALIBRATE POINT D
        elif self.state == STATE_3:
            self.read_GUI()
            yield (self.state)

            if self.CALIBRATION_FLG:
                # input location D into pan centroids
                self.calibrate_point(3, self.pan_position.get(), self
                self.CALIBRATION_FLG = False
                self.state = STATE_4

        ##################################

        ## STATE 4: CALIBRATE POINT E & 1
        elif self.state == STATE_4:
            self.read_GUI()
            yield (self.state)

            if self.CALIBRATION_FLG:
                # input location E into pan centroids
```

```
            self.calibrate_point(4, self.pan_position.get(), self
            # input location 1 into tilt centroids
            self.calibrate_point(0, self.pan_position.get(), self
            self.CALIBRATION_FLG = False
            self.state = STATE_5

    ##################################

    ## STATE 5: CALIBRATE POINT 2
    elif self.state == STATE_5:
        self.read_GUI()
        yield (self.state)

        if self.CALIBRATION_FLG:
            # input location 2 into tilt centroids
            self.calibrate_point(1, self.pan_position.get(), self
            self.CALIBRATION_FLG = False
            self.state = STATE_6

    ##################################

    ## STATE 6: CALIBRATE POINT 3
    elif self.state == STATE_6:
        self.read_GUI()
        yield (self.state)

        if self.CALIBRATION_FLG:
            # input location 3 into tilt centroids
            self.calibrate_point(2, self.pan_position.get(), self
            self.CALIBRATION_FLG = False
            self.state = STATE_7

    ##################################

    ## STATE 7: CALIBRATE POINT 4
    elif self.state == STATE_7:
        self.read_GUI()
        yield (self.state)

        if self.CALIBRATION_FLG:
            # input location 4 into tilt centroids
            self.calibrate_point(3, self.pan_position.get(), self
            self.CALIBRATION_FLG = False
            self.state = STATE_8

    ##################################
```

```
## STATE 8: CALIBRATE POINT 5
elif self.state == STATE_8:
    self.read_GUI()
    yield (self.state)

    if self.CALIBRATION_FLG:
        # input location 5 into tilt centroids
        self.calibrate_point(4, self.pan_position.get(), self
        print('Calibration complete.')
        self.state = STATE_9

###################################

## STATE 9: STOPPED, NOT SHOOTING
elif self.state == STATE_9:
    self.read_GUI()
    yield (self.state)

    if self.TARGET_CMD:
        if self.WINDUP_GUN.get():
            self.FEED_BULLETS.put(1)
            self.state = STATE_10
        else:
            print('Windup the Gun!!')

###################################

## STATE 10: MOVING, SHOOTING
elif self.state == STATE_10:
    # clear the target cmd flag for state 9 next time
    self.TARGET_CMD = False
    self.state = STATE_11

###################################

## STATE 11: STOPPED, SHOOTING
elif self.state == STATE_11:
    self.read_GUI()
    yield (self.state)

    if not self.FEED_BULLETS.get():
        self.state = STATE_9

###################################
```

```
            yield (self.state)

# ------------------------------------------------------------------------
# ------------------------------------------------------------------------

    def target_cmd(self, pan, tilt, target_cmd = True):
        ''' Defines what to do when target cmd is entered through the GUI
        self.pan_coords.put(pan)
        self.tilt_coords.put(tilt)
        print(pan)
        print(tilt)
        if target_cmd:
            self.TARGET_CMD = True
        else:
            self.TARGET_CMD = False

# ------------------------------------------------------------------------
# ------------------------------------------------------------------------

    def calibrate_point(self, index, pan_coor, tilt_coor, pan = False, ti
        ''' enters the calibrated point into the proper centroid list.
        @param index The index of the point in the centroid list
        @param pan_coor The pan coordinate of the point
        @param tilt_coor The tilt coordinate of the point
        @param pan Indicate if it's a pan calibration point
        @param tilt Indicate if it's a tilt calibration point
        '''
        if pan:
            self.pan_centroids[index] = pan_coor - 700
            self.pan_coords.put(pan_coor)

        if tilt:
            self.tilt_centroids[index] = tilt_coor + 3.5
            self.tilt_coords.put(tilt_coor)

# ------------------------------------------------------------------------
# ------------------------------------------------------------------------

    def GUI_Lookup_Table(self, command):
        ''' Decodes GUI commands based on a defined list of commands

        GUI Layout:
         _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
        | A1  B1  C1  D1  E1   Wind_on    Up       Calibration  |

        | A2  B2  C2  D2  E2   Feed_on    Down                  |
```

```
        | A3  B3  C3  D3  E3   Wind_off   Left                  |
        | A4  B4  C4  D4  E4   Feed_off   Right                 |
        | A5  B5  C5  D5  E5     Home                           |
         - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         GUI Command Numbers:
         _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
        | 1   6   11  16  21      26      31          36        |
        | 2   7   12  17  22      27      32                    |
        | 3   8   13  18  23      28      33                    |
        | 4   9   14  19  24      29      34                    |
        | 5   10  15  20  25      30                            |
         - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        @param command The incoming GUI command to decode
        '''

# --- A TARGETS ---

       # A1 Target
        if(command == 1):
            self.target_cmd(self.pan_centroids[0], self.tilt_centroids[0]

        # A2 Target
        elif(command == 2):
            self.target_cmd(self.pan_centroids[0], self.tilt_centroids[1]

        # A3 Target
        elif(command == 3):
            self.target_cmd(self.pan_centroids[0], self.tilt_centroids[2]

        # A4 Target
        elif(command == 4):
            self.target_cmd(self.pan_centroids[0], self.tilt_centroids[3]

        # A5 Target
        elif(command == 5):
            self.target_cmd(self.pan_centroids[0], self.tilt_centroids[4]
```

```
# --- B TARGETS ---

        # B1 Target
        elif(command == 6):
            self.target_cmd(self.pan_centroids[1], self.tilt_centroids[0]

        # B2 Target
        elif(command == 7):
            self.target_cmd(self.pan_centroids[1], self.tilt_centroids[1]

        # B3 Target
        elif(command == 8):
            self.target_cmd(self.pan_centroids[1], self.tilt_centroids[2]

        # B4 Target
        elif(command == 9):
            self.target_cmd(self.pan_centroids[1], self.tilt_centroids[3]

        # B5 Target
        elif(command == 10):
            self.target_cmd(self.pan_centroids[1], self.tilt_centroids[4]

# --- C TARGETS ---

        # C1 Target
        elif(command == 11):
            self.target_cmd(self.pan_centroids[2], self.tilt_centroids[0]

        # C2 Target
        elif(command == 12):
            self.target_cmd(self.pan_centroids[2], self.tilt_centroids[1]

        # C3 Target
        elif(command == 13):
            self.target_cmd(self.pan_centroids[2], self.tilt_centroids[2]

        # C4 Target
        elif(command == 14):
            self.target_cmd(self.pan_centroids[2], self.tilt_centroids[3]

        # C5 Target
        elif(command == 15):
            self.target_cmd(self.pan_centroids[2], self.tilt_centroids[4]

# --- D TARGETS ---
```

```
        # D1 Target
        elif(command == 16):
            self.target_cmd(self.pan_centroids[3], self.tilt_centroids[0]

        # D2 Target
        elif(command == 17):
            self.target_cmd(self.pan_centroids[3], self.tilt_centroids[1]

        # D3 Target
        elif(command == 18):
            self.target_cmd(self.pan_centroids[3], self.tilt_centroids[2]

        # D4 Target
        elif(command == 19):
            self.target_cmd(self.pan_centroids[3], self.tilt_centroids[3]

        # D5 Target
        elif(command == 20):
            self.target_cmd(self.pan_centroids[3], self.tilt_centroids[4]

    # --- E TARGETS ---

        # E1 Target
        elif(command == 21):
            self.target_cmd(self.pan_centroids[4], self.tilt_centroids[0]

        # E2 Target
        elif(command == 22):
            self.target_cmd(self.pan_centroids[4], self.tilt_centroids[1]

        # E3 Target
        elif(command == 23):
            self.target_cmd(self.pan_centroids[4], self.tilt_centroids[2]

        # E4 Target
        elif(command == 24):
            self.target_cmd(self.pan_centroids[4], self.tilt_centroids[3]

        # E5 Target
        elif(command == 25):
            self.target_cmd(self.pan_centroids[4], self.tilt_centroids[4]

    # --- SHOOT ---

        # WINDUP ON
        elif(command == 26):
```

```
            self.WINDUP_GUN.put(1)

        # FEED ON
        elif(command == 27):
            self.FEED_BULLETS.put(1)

        # WINDUP OFF
        elif(command == 28):
            self.WINDUP_GUN.put(0)

        # FEED OFF
        elif(command == 29):
            self.FEED_BULLETS.put(0)

# --- MOVE ---

        # UP
        elif(command == 31):
            self.tilt_coords.put(self.tilt_angle.get() + 1)

        # DOWN
        elif(command == 32):
            self.tilt_coords.put(self.tilt_angle.get() - 1)

        # LEFT
        elif(command == 33):
            self.pan_coords.put(self.pan_position.get() + 100)

        # RIGHT
        elif(command == 34):
            self.pan_coords.put(self.pan_position.get() - 100)

# --- CALIBRATE LOCATIONS ---

        # CALIBRATION POINT
        elif(command == 36):
            self.CALIBRATION_FLG = True


# --- Home Button ---

        # HOME
        elif(command == 30):
            self.target_cmd(self.pan_centroids[2], self.tilt_centroids[0]
```