```
"""
Created on Fri Feb 16 16:53:17 2018

@privatesection - Stuff in this file doesn't need to be Doxygen-ed

@author: JasonGrillo
"""

import pyb
import micropython
import gc

import encoder_task_func
import IMU_task_func
import motor_task_func
import nerf_task_func
import turret_hub_task_func

import cotask
import task_share



# Allocate memory so that exceptions raised in interrupt service routines
# generate useful diagnostic printouts
micropython.alloc_emergency_exception_buf (100)


# ============================================================================
# ======================== Run the Turret Code ===============================
# ============================================================================

if __name__ == "__main__":

    ############################################################################
    ############################## VARIABLES ###################################
    ############################################################################

    #Pan Coordinates Queue is used to deliver target pan encoder value to
    #Pan Motor Task from Turret Hub Task
    pan_coords = task_share.Queue ('f', 2, thread_protect = False,
                                   overwrite = False, name = "Pan_Coords"

    #Tilt Coordinates Queue is used to deliver target tilt IMU value to
    #Tilt Motor Task from Turret Hub Task
    tilt_coords = task_share.Queue ('f', 2, thread_protect = False,
```

```
                                    overwrite = False, name = "Tilt_Coord

#Pan Position Share is used to deliver current encoder value to
#to the Turret Hub and Pan Motor tasks from the Encoder Task
pan_position = task_share.Share ('f', thread_protect = False,
                                     name = "Pan_Position")

#Tilt Angle Share is used to deliver current IMU pitch value to
#to the Turret Hub and Tilt Motor tasks from the Encoder Task
tilt_angle = task_share.Share ('f', thread_protect = False,
                                    name = "Tilt_Position")

#Share Sent from Turret Hub Task to Nerf Gun Task to Start Feeding Bu
FEED_BULLETS = task_share.Share('f', thread_protect = False,
                                     name = "Feed_Bullets")

#Share sent from Turret Hub Task to Nerf Gun Task to
WINDUP_GUN = task_share.Share ('f', thread_protect = False,
                                    name = "Windup_Gun")

######################################################################
########################### TASK OBJECTS #############################
######################################################################

pan_encoder = encoder_task_func.Encoder_Task(pan_position, 4,
                                     pyb.Pin.board.PB6, pyb.Pin.board.PB7

tilt_IMU = IMU_task_func.IMU_Task(tilt_angle) #what to put here 0 for

#0.02
pan_motor = motor_task_func.Motor_Task(pan_position,
                                       pan_coords, 3,
                                       pyb.Pin.board.PA10,
                                       pyb.Pin.board.PB4,
                                       pyb.Pin.board.PB5, 0.01, .0125
#1.2
tilt_motor = motor_task_func.Motor_Task(tilt_angle,
                                        tilt_coords, 5,
                                        pyb.Pin.board.PC1,
                                        pyb.Pin.board.PA0,
                                        pyb.Pin.board.PA1, 2.0, 0.75,

turret_hub = turret_hub_task_func.Turret_Hub_Task(pan_position, tilt_
                                                  tilt_coords, FEED_B

nerf_gun = nerf_task_func.Nerf_Task(WINDUP_GUN, FEED_BULLETS, pyb.Pin
```

```
##############################################################
############################## TASKS ##########################
##############################################################

#Turret Hub Timing => Timing: 100 ms, Priority 1 (Lowest)
task0 = cotask.Task (turret_hub.turret_hub_fun, name = 'Task_0', prio
                     period = 100, profile = True, trace = False)
#Pan Encoder => Timing 2 ms, Priority 5(Highest)
task1 = cotask.Task (pan_encoder.enc_fun, name = 'Task_1', priority =
                     period = 2, profile = True, trace = False)
#Tilt IMU => Timing 5 ms (minimum 10 ms, applied 2x SF), Priority 5(H
task2 = cotask.Task (tilt_IMU.IMU_fun, name = 'Task_2', priority = 5,
                     period = 5, profile = True, trace = False)
#Pan Motor => Timing 20 ms, Priority 3 (Medium)
task3 = cotask.Task (pan_motor.mot_fun, name = 'Task_3', priority = 3
                     period = 20, profile = True, trace = False)
#Tilt Motor => Timing 20 ms, Priority 3 (Medium)
task4 = cotask.Task (tilt_motor.mot_fun, name = 'Task_4', priority =
                     period = 20, profile = True, trace = False)
#Nerf Gun => Timing 200 ms, Priority 1 (Lowest)
task5 = cotask.Task (nerf_gun.gun_fun, name = 'Task_5', priority = 1,
                     period = 200, profile = True, trace = False)

cotask.task_list.append (task0)
cotask.task_list.append (task1)
cotask.task_list.append (task2)
cotask.task_list.append (task3)
cotask.task_list.append (task4)
cotask.task_list.append (task5)

# Run the memory garbage collector to ensure memory is as defragmente
# possible before the real-time scheduler is started
gc.collect ()

##############################################################
############################## RUN ############################
##############################################################

# Run the scheduler with the chosen scheduling algorithm
while True:
    cotask.task_list.pri_sched()
```