```
#Library Imports
from tritium.control import ControlFunction
from tritium.arbitration import BidType, Precedence, arbitration_by, arbitration_check
from socketserver import UDPServer, ThreadingMixIn, BaseRequestHandler
from threading import Thread
from queue import Queue

#Base Class
class UDPSequenceControlRequestHandler(BaseRequestHandler):

   command_queue = None

   def handle(self):
      data = self.request[0].strip()
      cmd = data.decode('utf-8')
      self.command_queue.put(cmd)


class ThreadedUDPServer(ThreadingMixIn, UDPServer):
   allow_reuse_address = True

# Creation of UDP Connection
class UDPSequenceControlServer(ControlFunction):

   arbitration = { #For robots with arbitration only
            'controls' : 'TTS',
            'precedence': Precedence.USER_MAXIMUM,
            'type': BidType.ON_DEMAND
   }

   HOST = ""
   PORT = 9999

   def on_activate(self):
      self.server = s = ThreadedUDPServer((self.HOST, self.PORT),
UDPSequenceControlRequestHandler)
      UDPSequenceControlRequestHandler.command_queue = self.command_queue = Queue()

      self.server_thread = t = Thread(target=s.serve_forever)
      t.daemon = True
      t.start()

   def on_deactivate(self):
      self.robot.stop_all_sequences()
      self.robot.play_sequence('rest_pose')
      self.server.shutdown()

   def on_tick(self):
      q = self.command_queue
      while not q.empty():
         cmd = q.get_nowait()
```

```
        self.debug_out('command: {}'.format(cmd))
        self.handle_command(cmd)

    def handle_command(self, cmd):

        r = self.robot
        left_ind = self.robot['Flex Left Index'].demand

        # Stop Everything
        if cmd.startswith('stop'):
            self.debug_out('STOP')
            r.stop_all_sequences()

        # Stop Everything and Reset
        elif cmd.startswith('Reset'):
            self.debug_out('RESET')
            self.relax()

        # Play Sequence
        elif cmd.startswith('play:'):
            sn = cmd[5:]
            self.debug_out('PLAY {0}'.format(sn))
            r.play_sequence(sn)

        # Fist Controls
        # Left Fist
        elif cmd.startswith('LFist:'):
            sn = cmd[6:]
            if(sn=="0"):
                #No Fist, Open Left Fist
                r['Thumb Pitch Left'].demand = 0
                r['Thumb Roll Left'].demand = 0
                r['Thumb Flex Left'].demand = 0
                r['Flex Left Index'].demand = 0
                r['Flex Left Middle'].demand = 0
                r['Flex Left Pinky'].demand = 0
                r['Flex Left Ring'].demand = 0
            else:
                #Open Left Fist
                r['Thumb Pitch Left'].demand = 4800
                r['Thumb Roll Left'].demand = 4800
                r['Thumb Flex Left'].demand = 4800
                r['Flex Left Index'].demand = 4800
                r['Flex Left Middle'].demand = 4800
                r['Flex Left Pinky'].demand = 4800
                r['Flex Left Ring'].demand = 4800
        # Right Fist
        elif cmd.startswith('RFist:'):
            sn = cmd[6:]
            if(sn=="0"):
                #No Fist, Open Right Fist
```

```
            r['Thumb Pitch Right'].demand = 0
            r['Thumb Roll Right'].demand = 0
            r['Thumb Flex Right'].demand = 0
            r['Flex Right Index'].demand = 0
            r['Flex Right Middle'].demand = 0
            #r['Flex Right Pinky'].demand = 0
            r['Flex Right Ring'].demand = 0
        else:
            #Open Right Fist
            r['Thumb Pitch Right'].demand = 4800
            r['Thumb Roll Right'].demand = 4800
            r['Thumb Flex Right'].demand = 4800
            r['Flex Right Index'].demand = 4800
            r['Flex Right Middle'].demand = 4800
            #r['Flex Right Pinky'].demand = 4800
            #r['Flex Right Ring'].demand = 4800

    # Torso Control
    # Input: Torso:-10+10
    elif cmd.startswith('Torso:'):
        sn = cmd[6:]
        self.debug_out('Torso {0} {1}'.format(sn[:3],sn[-3:]))
        #L Arm Up [-10,10]: Local Rotation Z
        r['Torso Pitch'].demand = int(sn[:3])
        #L Arm Twist [-15,15]: Local Rotation X
        r['Torso Yaw'].demand = int(sn[-3:])

    # Left Upper Arm Control
    # Input: LArm:-10-20+30
    elif cmd.startswith('LArm:'):
        sn = cmd[5:]
        self.debug_out('Left Arm Up {0} {1} {2}'.format(sn[:3],sn[3:6],sn[-3:]))
        #L Arm Up [-80,0]: Local Rotation Z
        r['Shoulder Pitch Left'].demand = int(sn[:3])
        #L Arm Out [-80,-20]: Local Rotation Y
        r['Shoulder Roll Left'].demand = int(sn[3:6])
        #L Arm Twist [-25,35]: Local Rotation X
        r['Shoulder Yaw Left'].demand = int(sn[-3:])

    # Left Forearm Control
    elif cmd.startswith('LForearm:'):
        sn = cmd[9:]
        self.debug_out('Left Forearm Up {0} {1}'.format(sn[:3],sn[-3:]))
        #L Arm Up [-80,10]: Local Rotation Z
        r['Elbow Pitch Left'].demand = int(sn[:3])
        #L Wrist Roll [-80,80]: Local Rotation Y
        r['Wrist Roll Left'].demand = int(sn[-3:])

    # Right Upper Arm Control
    # Input: RArm:-10-20+30
    elif cmd.startswith('RArm:'):
```

```
            sn = cmd[5:]
            self.debug_out('Right Arm Up {0} {1} {2}'.format(sn[:3],sn[3:6],sn[-3:]))
            #L Arm Up [-80,-10]: Local Rotation Z
            r['Shoulder Pitch Right'].demand = int(sn[:3])
            #L Arm Out [-85,-35]: Local Rotation Y
            r['Shoulder Roll Right'].demand = int(sn[3:6])
            #L Arm Twist [-25,35]: Local Rotation X
            r['Shoulder Yaw Right'].demand = int(sn[-3:])

        # Right Forearm Control
        elif cmd.startswith('RForearm:'):
            sn = cmd[9:]
            self.debug_out('Right Forearm Up {0} {1}'.format(sn[:3],sn[-3:]))
            #R Arm Up [-80,10]: Local Rotation Z
            r['Elbow Pitch Right'].demand = int(sn[:3])
            #R Wrist Roll [-80,10]: Local Rotation Y
            r['Wrist Roll Right'].demand = int(sn[-3:])

        else:
            self.debug_out('WARNING unrecognised command: {0}'.format(cmd))

    def relax(self):
            self.robot.stop_all_sequences()
            self.robot['Shoulder Pitch Left'].demand = -80
            self.robot['Shoulder Roll Left'].demand = -85
            self.robot['Shoulder Yaw Left'].demand = 5
            self.robot['Elbow Pitch Left'].demand = -80
            self.robot['Wrist Pitch Left'].demand = 0
            self.robot['Wrist Roll Left'].demand = 10

            self.robot['Shoulder Pitch Right'].demand = -80
            self.robot['Shoulder Roll Right'].demand = -85
            self.robot['Shoulder Yaw Right'].demand = 5
            self.robot['Elbow Pitch Right'].demand = -80
            self.robot['Wrist Pitch Right'].demand = 0
            self.robot['Wrist Roll Right'].demand = 10


            self.robot['Torso Pitch'].demand = 0
            self.robot['Torso Roll'].demand = 0
            self.robot['Torso Yaw'].demand = 0

            self.robot['Head Pitch'].demand = 0
            self.robot['Head Roll'].demand = 0
            self.robot['Head Yaw'].demand = 0

            self.robot['Flex Right Index'].demand = 0
            self.robot['Flex Right Middle'].demand = 0
            self.robot['Flex Right Ring'].demand = 0
            self.robot['Flex Right Pinky'].demand = 0
            self.robot['Thumb Flex Right'].demand = 2500
```

```
self.robot['Thumb Pitch Right'].demand = 2500
self.robot['Thumb Roll Right'].demand = 4800
self.robot['Finger Spread Right'].demand = 2500

self.robot['Flex Left Index'].demand = 0
self.robot['Flex Left Middle'].demand = 0
self.robot['Flex Left Ring'].demand = 0
self.robot['Flex Left Pinky'].demand = 0
self.robot['Thumb Flex Left'].demand = 2500
self.robot['Thumb Pitch Left'].demand = 2500
self.robot['Thumb Roll Left'].demand = 4800
self.robot['Finger Spread Left'].demand = 2500

self.robot.play_sequence('RestPose')
```