## Interrupt version

I'm making a GEM telescope mount that automatically targets and tracks stars using a nema 17 motor with 1.8 degree step size, each with a 27:1 planetary gear reduction and a 30:1 Cycloidal gear reduction. I can microstep 32 times per step. I use tetra3 to see where the telescope is currently pointed

I’d like to write code for my automatic telescope mount.   
There will be three scripts.  
1) Sensor.py (on my laptop) will be continuously running, and using Tetra3 and a camera, will provide where the telescope is currently pointing. It will make a RA and Dec observation every so often (about every 1-3 seconds), and interrupt main.py to tell it a new observation whenever it does. It will also give the desired RA and Dec of the object we want to view. (you can assume the actual solving/camera interfacing to be blackboxed)  
It will be working as fast as it can. As soon as it has an observation, it will tell main.py via a socket connection (which you should implement). There is a camera that will take a picture and save it every 3 seconds. As soon as the picture is saved, it can begin to solve for the new observation

2) Main.py (on my laptop) will tell the controller.ino what to do. Whenever it gets a desired and current RA and Dec from sensor.py (through a socket, which you should implement) it will calculate number of microsteps (in each axis, microsteps = angular error in degrees/((1.8/32)/810)), and send it to controller.ino via a serial comport connection, which you should implement. If the error is less than 10 arcseconds, it will send an instruction of 0 steps.  
  
3) Controller.ino will run on the ESP32 and will get serial communication from the main.py script. It will do the number of steps, until interrupted by a new instruction, in which case it will forget the old instruction. If it receives an instruction of 0 steps, the Dec motor should stop moving, and the RA motor will track the sidereal rate, with 16621.24 miscroseconds between each microstep.  
  
  
I’m not sure what driver I have, but here is sample code for the driver. Only use this code for reference to try to figure out how to work with the microstep driver  
  
//defines pins

const int stepPin = 6; //PUL -Pulse

const int dirPin = 7; //DIR -Direction

const int enPin = 8; //ENA -Enable

void setup(){

//Sets the pins as Outputs

pinMode(stepPin,OUTPUT);

pinMode(dirPin,OUTPUT);

pinMode(enPin,OUTPUT);

digitalWrite(enPin,LOW);

}

void loop(){

//Enables the motor direction to move

digitalWrite(dirPin,HIGH);

//Makes 200 Pulses for making one full cycle rotation

for(int x = 0; x < 200; x++){

digitalWrite(stepPin,HIGH);

delayMicroseconds(500);

digitalWrite(stepPin,LOW);

delayMicroseconds(500);

}

//One second delay

delay(1000);

//Changes the rotations direction

digitalWrite(dirPin,LOW);

// Makes 200 pulses for making one full cycle rotation

for(int x = 0; x < 200; x++) {

digitalWrite(stepPin,HIGH);

delayMicroseconds(500);

digitalWrite(stepPin,LOW);

delayMicroseconds(500);

}

//One second delay

delay(1000);

## Complete Each Movement Version

c

write sensor and main now, and wait until my next prompt to write controller

## Argparser

I want to create an argparser.  
  
Essentially, you can call astrid in two ways.  
  
you can

Ok, now write me a .py .ino pair where the py file prompts the user for a number of microsteps for the .ino to turn a stepper. Then, using serial, it tells the esp32 how many steps. As soon as esp32 is done, it asks python for another user input , at which point python will prompt the user again.  
  
Between commands, the motor will be in an idle state. If it’s idling, it will drift at 32 microsteps a second. While idling, it will ask python for another user input every 5 seconds. Until it gets an input, it will continue idling.

(no header files btw)