```
/*

This program establishes the communication between the arduino and the computer.

  It is constantly looking to recieve a sequence of movements of the form:

    'direction_linear_movement/steps_linear_movement/direction_rotation/steps_rotation/*'

    The first 2 instructions are to be transmitted to the stepper motor connected to the linear feedthrough.
    The last 2 instructions are to be transmitted to the stepper motor connected to the rotatory platform.
    The '/' is an end marker that sinalizes an instruction as been recieved.
    The '*' is an end marker that sinalizes the end of a group of 4 instructions: the information is
    only transmitted to the stepper motors once a full sequence of movement (the 4 instructions) has been
    recieved.

    The direction instructions can only be 1 or 0.
    The steps instructions can be any integer.

*/

const int numChars = 6;      // max number of chars per sequence of movement
char receivedChars[numChars];  // array to store the chars of the received data

boolean newData = false;       // check if full sequence of numbers as been recieved

const int l = 4;              // max number of instructions per sequence of movements
long data[l];                   // array to store the recieved data converted to integers

// Linear feedthrough (motor 1) pins
const int LINEAR_STEPS = 3;           // Arduino pin connected to steps of X axis in the gecko motor driver
const int LINEAR_DIR = 4;             // Arduino pin connected to dir of X axis in gecko motor driver
// Rotatory platform (motor 2) pins
const int ROTATION_STEPS = 5;         // Arduino pin connected to steps of Y axis in gecko motor driver
const int ROTATION_DIR = 6;           // Arduino pin connected to dir of Y axis in gecko motor driver

long i;
int j;
bool stopRotation = false;
int t = 200;

void setup() {
    Serial.begin(115200);
    pinMode(LINEAR_STEPS, OUTPUT);
    pinMode(LINEAR_DIR, OUTPUT);
    pinMode(ROTATION_STEPS, OUTPUT);
    pinMode(ROTATION_DIR, OUTPUT);
}

void loop() {
    recvMoveSeq();
    exeMovement();
}

void recvMoveSeq(){

    static byte ndx = 0;
    static byte n = 0;
    char endMarker1 = '/'; // end each steps and dir: /
    char endMarker2 = '*'; // end sequence of movements: *
    char rc;

    if (Serial.available() > 0){

        rc = Serial.read();

        if(rc == endMarker2){
            n = 0;
            newData = true;
        }else if(rc == endMarker1){
            receivedChars[ndx] = '\0'; // terminate the string
            ndx = 0;
            data[n] = atol(receivedChars);
            n++;
        }else{
            receivedChars[ndx] = rc;
            ndx++;
        }
    }
}
```

```
void exeMovement() {

    if (newData == true) {

        Serial.println('v'); // send "v" (of "vertical") to Python code

        // send motors a constant positive signal to indicate the movement is down
        if(data[0] == 1){
            digitalWrite(LINEAR_DIR, HIGH);
            Serial.println('d'); // send "d" (of "down") to Python code
        }

        // send motors a constant positive signal to indicate the movement is up
        if(data[0] == 0){
            digitalWrite(LINEAR_DIR, LOW);
            Serial.println('u'); // send "u" (of "up") to Python code
        }

        if(data[1] > 0){

            // send motors a pulse of 400us for each step
            for(i = 1; i <= data[1]; ++i){
                if(Serial.read() == 's'){stopRotation = true; break;}
                digitalWrite(LINEAR_STEPS, HIGH);
                delayMicroseconds(t);
                digitalWrite(LINEAR_STEPS, LOW);
                delayMicroseconds(t);
                if(i % 10 == 0){Serial.println(10);} // send Python code "10" indicating 10 steps were taken
                if(i == data[1]){Serial.println(i % 10);} // send Python the number of steps from penultimate
                                                          // to last step
            }

            // if the movement is up, erase the mechanical gap by moving half a motor turn up and then down
            if(data[0] == 0){
                for(j = 1; j <= 1000; ++j){
                    digitalWrite(LINEAR_STEPS, HIGH);
                    delayMicroseconds(t);
                    digitalWrite(LINEAR_STEPS, LOW);
                    delayMicroseconds(t);
                    if(j % 10 == 0){Serial.println(10);}
                }

                digitalWrite(LINEAR_DIR, HIGH);
                Serial.println('d');

                for(j = 1; j <= 1000; ++j){
                    digitalWrite(LINEAR_STEPS, HIGH);
                    delayMicroseconds(t);
                    digitalWrite(LINEAR_STEPS, LOW);
                    delayMicroseconds(t);
                    if(j % 10 == 0){Serial.println(10);}
               }
            }
         }

        Serial.println('r'); // send "r" (of "rotation") to Python code

        if(data[2] == 1){
            digitalWrite(ROTATION_DIR, HIGH);
            Serial.println('n'); // send "n" (of "negative direction (360º -> 0º)") to Python code
        }

        if(data[2] == 0){
            digitalWrite(ROTATION_DIR, LOW);
            Serial.println('p'); // send "p" (of "positive direction (0º -> 360º)") to Python code
        }

        if(data[3] > 0){

            for(i = 1; i <= data[3]; ++i){
                if(Serial.read() == 's' or stopRotation == true){break;}
                digitalWrite(ROTATION_STEPS, HIGH);
                delayMicroseconds(t);
                digitalWrite(ROTATION_STEPS, LOW);
                delayMicroseconds(t);
                if(i % 10 == 0){Serial.println(10);}
                if(i == data[1]){Serial.println(i % 10);}
            }

            // if the movement is in the negative direction, erase the mechanical gap by moving half
            // a motor turn in the negative direction and then in the positive direction
            if(data[2] == 1 and stopRotation == false){
                for(j = 1; j <= 1000; ++j){
                    digitalWrite(ROTATION_STEPS, HIGH);
                    delayMicroseconds(t);
                    digitalWrite(ROTATION_STEPS, LOW);
                    delayMicroseconds(t);
                    if(j % 10 == 0){Serial.println(10);}
                }

                digitalWrite(ROTATION_DIR, LOW);
                Serial.println('p');

                for(j = 1; j <= 1000; ++j){
                    digitalWrite(ROTATION_STEPS, HIGH);
                    delayMicroseconds(t);
                    digitalWrite(ROTATION_STEPS, LOW);
                    delayMicroseconds(t);
                    if(j % 10 == 0){Serial.println(10);}
               }
            }
         }

        newData = false;
        stopRotation = false;
        Serial.println('f'); // send "f" (of "finished") to Python code
    }
}
```