Baby Shark

Documentation and Operating Instructions

11/28/2022

Version 1.1

Table of Contents

Introduction

This animatronic device consists of a linear actuator that bounces up and down. It is controlled using an Arduino, stepper motor drive, and SPDT sensors.

Parts List

- **Fuyu FSL40 700mm stroke linear guide rail motor drive**
  Documentation:
  - https://www.fuyumotion.com/linear-2-axis-guide-rail-motor-drive-ball-screw-xy-stage-product/
  - https://components101.com/motors/nema-23-stepper-motor-datasheet-spec
- **DM556T stepper motor driver**
  Documentation:
  - https://www.omc-stepperonline.com/digital-stepper-driver-1-8-5-6a-20-50vdc-for-nema-23-24-34-stepper-motor-dm556t
- **Arduino Mega**
- **SPDT Switch**

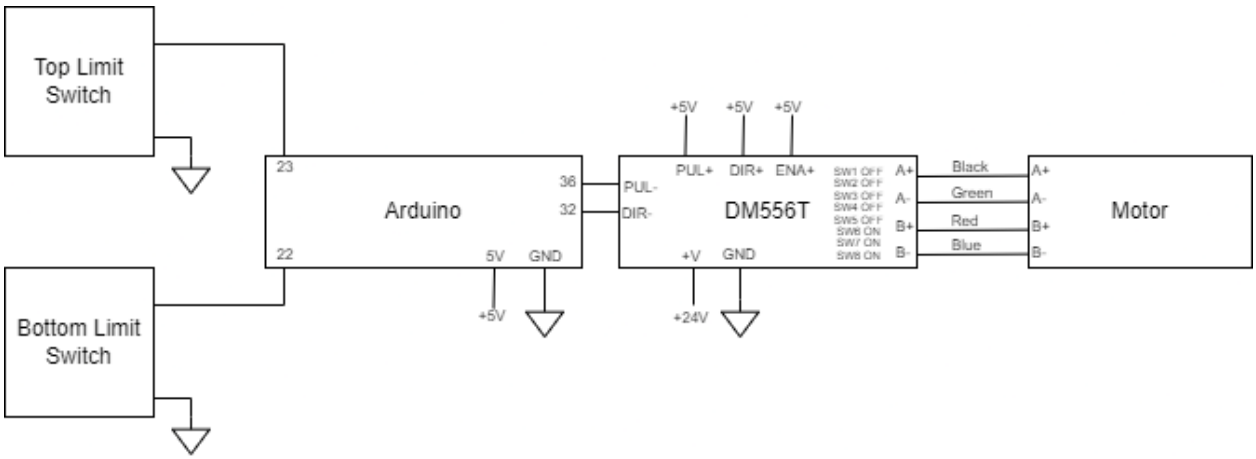
Description

The actuator is driven by a DC stepper motor controlled using a microcontroller and a motor driver. Limit switches are placed at either end of the actuator's rail to calibrate the distance of travel of the actuator and to prevent the actuator from over-travelling.

The actuator travels up and down at a programmed speed and acceleration. Hold times are programmed at the top and bottom position.

Upon startup, the actuator will complete a calibration cycle. It will move down until hitting the bottom limit switch, then move up until hitting the top limit switch. It will then move regularly down and up without touching the switches.

Block Diagram



Code

Code for this project is available on Github. This project uses the AccelStepper library for DC stepper motor control.

Pin Assignments

These parameters define the pin assignments on the Arduino. Digital pins are used for inputs and outputs. Switches use internal pullup resistors. **It is not recommended to change these values.**

```
/* PIN DEFINITIONS */
#define PUL             36
#define DIR             32
#define BOTTOM_TRIGGER  22
#define TOP_TRIGGER     23
```

The Arduino is also used as a voltage regulator to 5V, but these pin assignments are not listed in the code. **The program for the** `TOP_TRIGGER` **has been disabled.**

Speed and Hold Time Parameters

These parameters control the speed and acceleration of movement as well as the hold times at the top and bottom of movement during which time the actuator is not moving. **These values can be changed to adjust speed and hold times.**

```
 9   /* SPEED PARAMETERS */
10   #define MAXSPEED     1500  // steps per second
11   #define ACCELERATION 2000  // steps per second per second
12
13   /* HOLD TIMES */
14   #define TOP_WAIT            8000    //millisceonds
15   #define BOTTOM_WAIT         3000    //milliseconds
```

Speed is defined by the number of pulses of the DC motor per second, i.e. (steps/second). Acceleration is defined by (steps/second/second). Upon running, the motor will accelerate at the rate defined by `ACCELERATION` to the maximum speed defined by `MAXSPEED`. The motor operates at 400 (pulses/revolution), thus the speed is 3.75 (revolutions/second).

Hold times determine the amount of time the motor spends waiting at the top or bottom. Time is defined in milliseconds.

It should be noted that faster speeds will cause the motor to heat up faster. To allow the motor time to cool down, it is recommended to increase hold times.

Other Parameters

These parameters define control logic used by the code for calibration and direction. **It is not recommended to change these values.**

```
/* OTHER MAGIC NUMBERS */
#define START_DELAY         1000
#define ACTUATOR_CALIBRATE  26000   //Max height of travel
#define START_CALIBRATE     100000  //Distance beyond actual actuator length for calibration purposes
#define INCH                1000    //Approx 3000 steps per inch
#define INIT_DIRECTION      -1      //-1 is DOWN, 1 is UP
#define UP_DIR              1
#define DOWN_DIR            -1
```

`START_DELAY` is the amount of time in milliseconds between powering the Arduino and the initial run of the program. `START_CALIBRATE` is an arbitrary number of pulses of the DC motor, approximately equivalent to the amount of pulses required to span the length of the actuator's stroke. `ACTUATOR_CALIBRATE` defines the maximum height of travel. `INCH` is the number of pulses required to span one inch. `INIT_DIRECTION` defines the initial direction of travel. Other parameters are for control logic.

Changing the Code

1. Clone the Github repository or download the code.
2. Install the Arduino IDE
3. Install the AccelStepper library
4. Open the project.
5. Verify and upload the code to the Arduino

Many of the parameters in the code can be easily reprogrammed. Simply change the number value of the parameter you wish to redefine.

Operating Instructions

Turning on

1. Plug in the 24V power supply for the actuator.
2. Plug in the 5V power supply for the Arduino.

Turning off

1. Unplug the 24V power supply.
2. Unplug the 5V power supply.

Troubleshooting

If the actuator does not move upon plugging in

1. Ensure the Arduino is powered.
2. Press the Reset button on the Arduino to restart the program.
3. Check wire connections to Arduino and DM556T driver.

If the actuator gets stuck at the bottom

1. Unplug the actuator and Arduino.
2. Manually turn the motor to move the actuator away from the bottom.
3. Check if switches are intact.
4. Ensure that the base of the motor is flush with the baseboard.

   If switches are not broken:
   - Change the DIP switch settings to increase the pulses/revolution.

     OR
   - Decrease `MAXSPEED` in programming.

If the actuator gets stuck at the top

1. Unplug the actuator and Arduino.
2. Ensure that the base of the motor is flush with the baseboard.
3. Restart the Arduino and actuator.

If the actuator does not reach the bottom

1. Press the reset button on the arduino

   If problem persists:
   1. Unplug the actuator and the Arduino
   2. Check if the bottom switch is intact

If the actuator does not reach the top

1. Unplug the actuator and Arduino
2. Cut the wires attached to the top switch. The maximum height is defined in the programming and the top switch is not strictly necessary. DO NOT REMOVE THE BOTTOM SWITCH

If the actuator moves UP instead of DOWN on startup

1. Press reset button on Arduino