User Manual For 2D Cartesian Belt Drive

Author: Tianyu Wang, Jiayin Qu

Johns Hopkins University

Special Thanks to Dr. Kriger and Dr. Vagvolgyi for technical support

Catalog

Components

1. 3 * 57 Stepper Motor
2. 3* Motor Controller
3. 2* Heavy Duty Servo
4. 2* 500 mm belt drive (Forward-backward direction)
5. 1* 3400 mm belt drive (Left-right direction)
6. 1* 2000 mm belt drive (Up-down direction)
7. Arduino Mega 2560 Board
8. Arduino Mega 2560 Extension Board
9. Power Supply for all
10. Wires

Assembly

The overall assembly of the system using the above components are shown below.

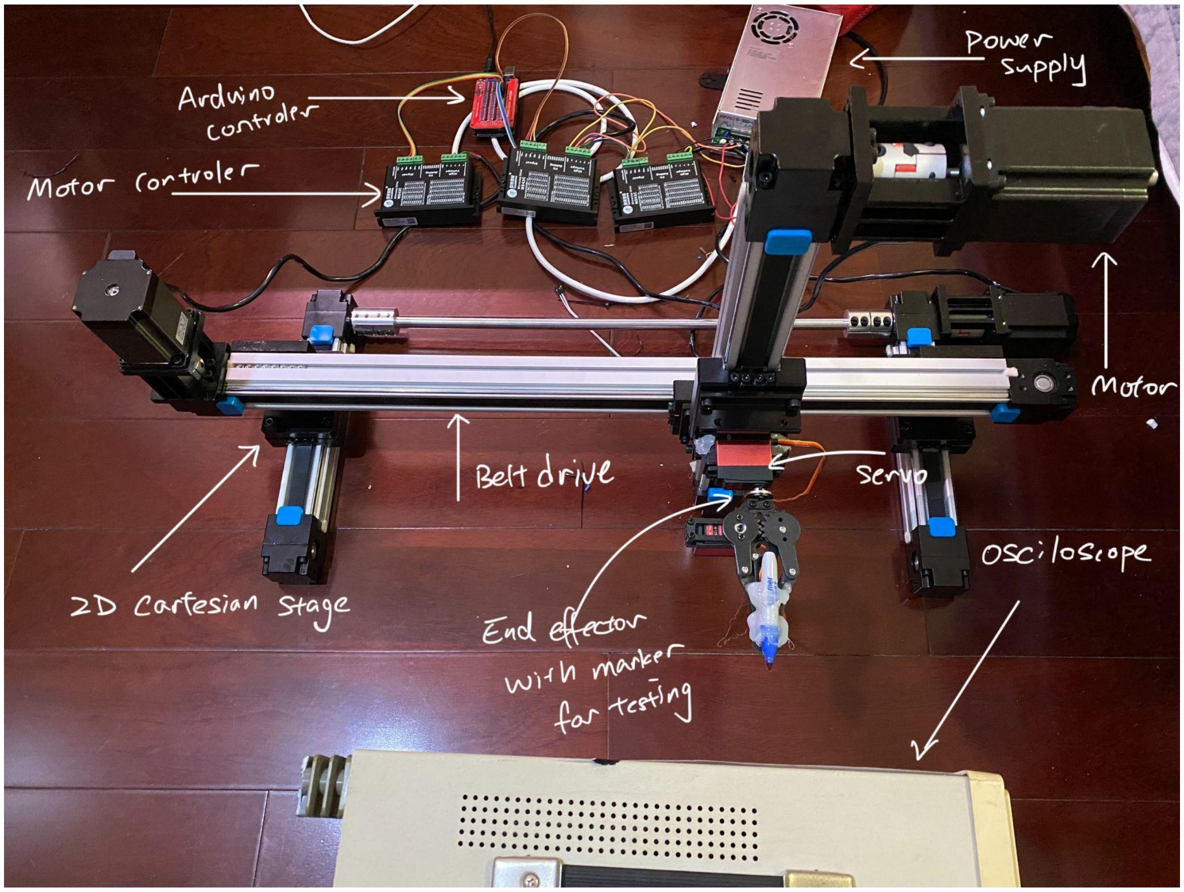


Fig 1. Assembly of the whole system.

Here we showcase some of the close up assembly and wiring choices.

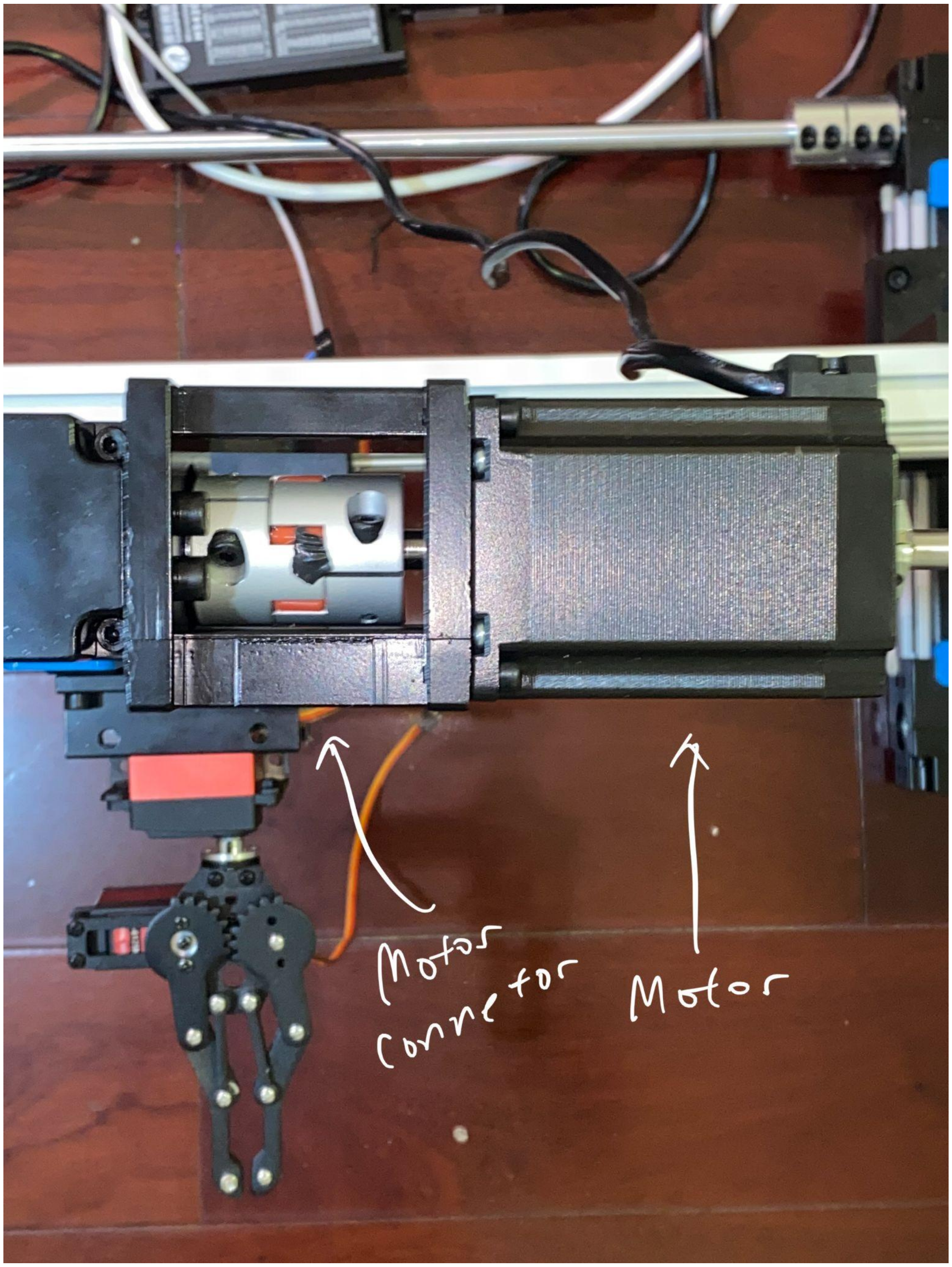


Fig2. Motor Assembly

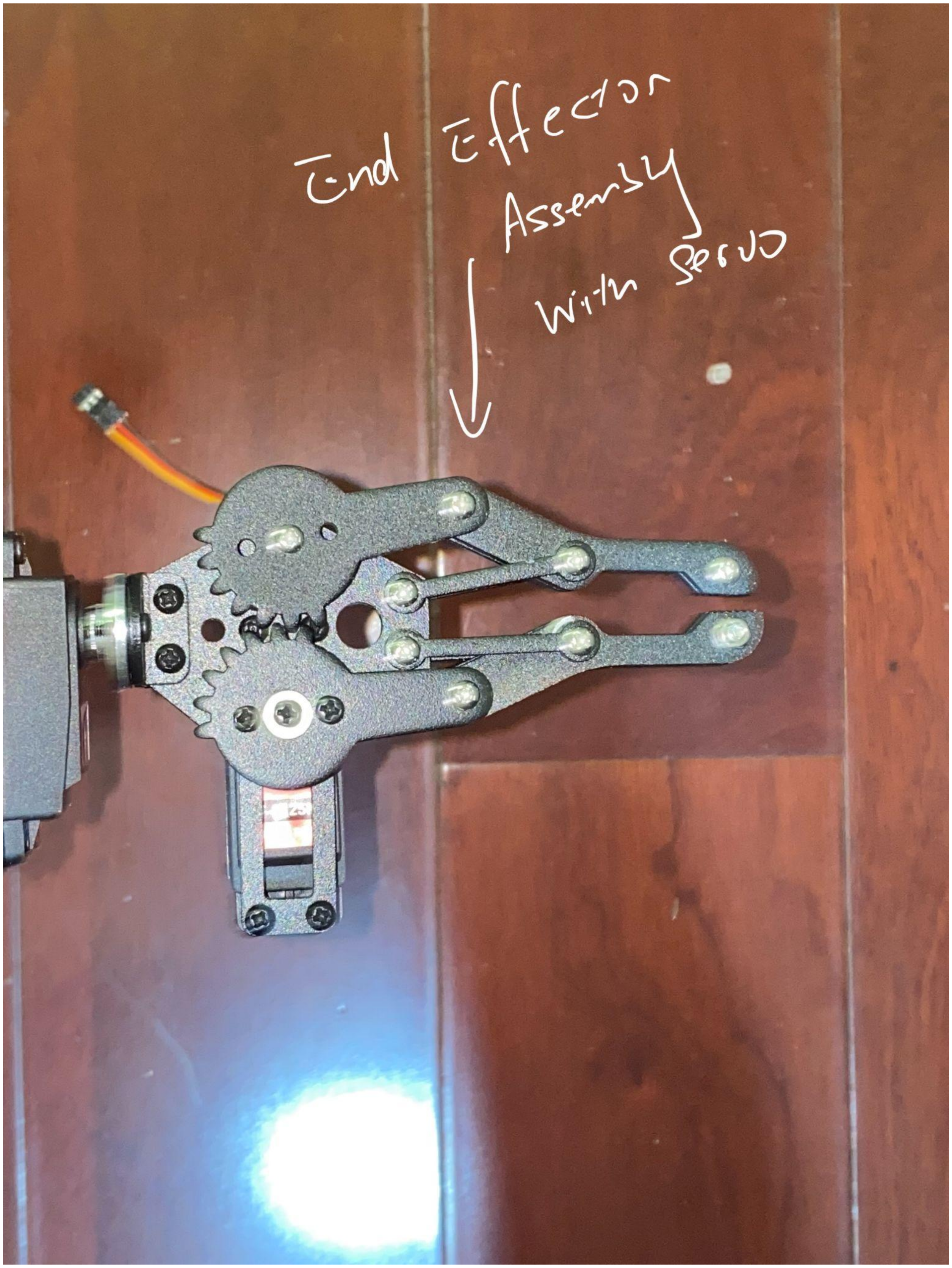


Fig 3. End-effector Assembly

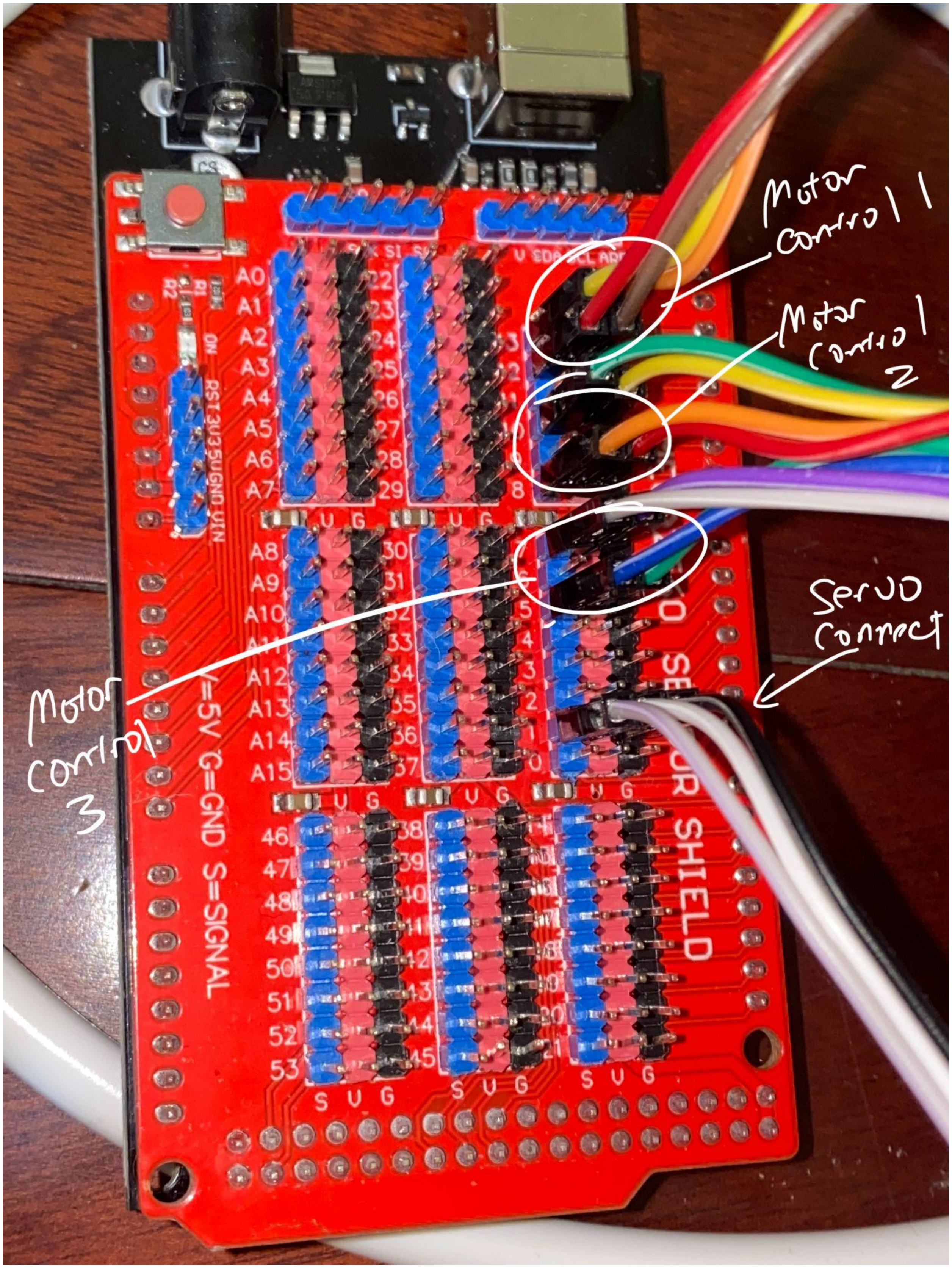


Fig 3. Arduino Board Wiring

Note here that motor controller 1, 2,3 is respectively left-right, up-down , and front-back direction movement motor. However, you can choose to mount them differently as long as they are connected on PWM supported pins of the arduino. Notice also here that you should connect the dir+ and pulse+ to 5 V (red) and dir- and pulse - to the signal pin(blue).

For servo connection, you simply align the three pin to the correct alignment of signal, 5V and ground.

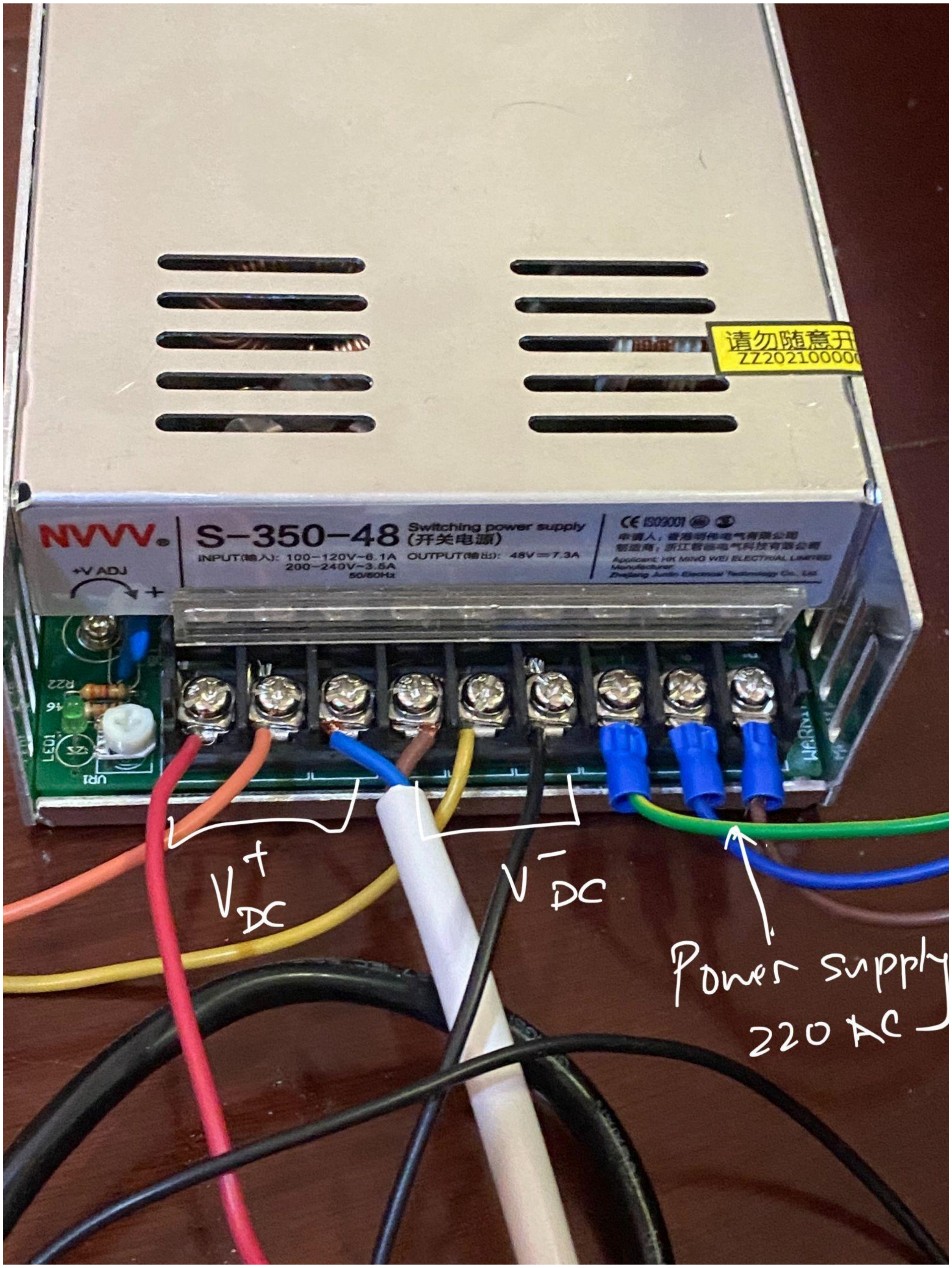
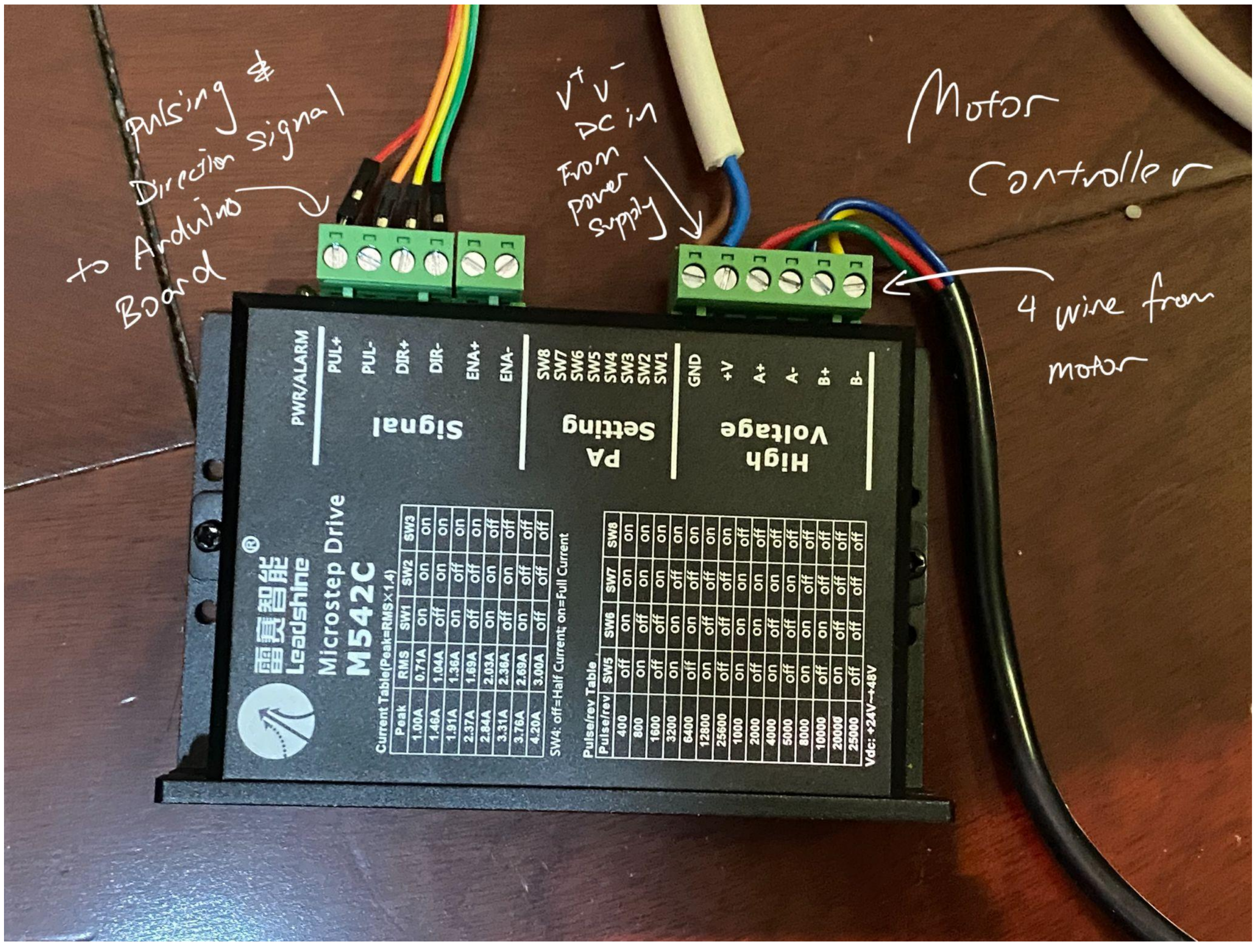


Fig 4. Power Supply.

Note Here that the power supply may vary depending on your supplier. But the general truth is that you should take the AC in from the right and connect respectively V+ V- and Ground. Then on the left, you connect all the corresponding V+ DC out to the V+ of the motor controller and V- DC to the V- / Ground of the motor controller.

Before we get started with the experiments it would be a good idea to set the current switches to suit your stepper motor by reading its manual and adjusting setting on the side as shown by the picture below.



Current Table(Peak=RMS×1.4)

Peak	RMS	SW1	SW2	SW3
1.00A	0.71A	on	on	on
1.46A	1.04A	off	on	on
1.91A	1.36A	on	off	on
2.37A	1.69A	off	off	on
2.84A	2.03A	on	on	off
3.31A	2.36A	off	on	off
3.76A	2.69A	on	off	off
4.20A	3.00A	off	off	off

SW4: off=Half Current; on=Full Current

Pulse/rev Table

Pulse/rev	SW5	SW6	SW7	SW8
400	off	on	on	on
800	on	off	on	on
1600	off	off	on	on
3200	on	on	off	on
6400	off	on	off	on
12800	on	off	off	on
25600	off	off	off	on
1000	on	on	on	off
2000	off	on	on	off
4000	on	off	on	off
5000	off	off	on	off
8000	on	on	off	off
10000	off	on	off	off
20000	on	off	off	off
25000	off	off	off	off

Vdc: +24V~+48V

Fig 6. Sample motor controller config from implementation.

Here is index for the different connector

- PUL – This is the Pulse that steps the motor.
- DIR – This is a logic signal to set the motor Direction.
- ENA – This is an Enable signal
- We ignore the ENA (Enable) connections and let module always being enabled.

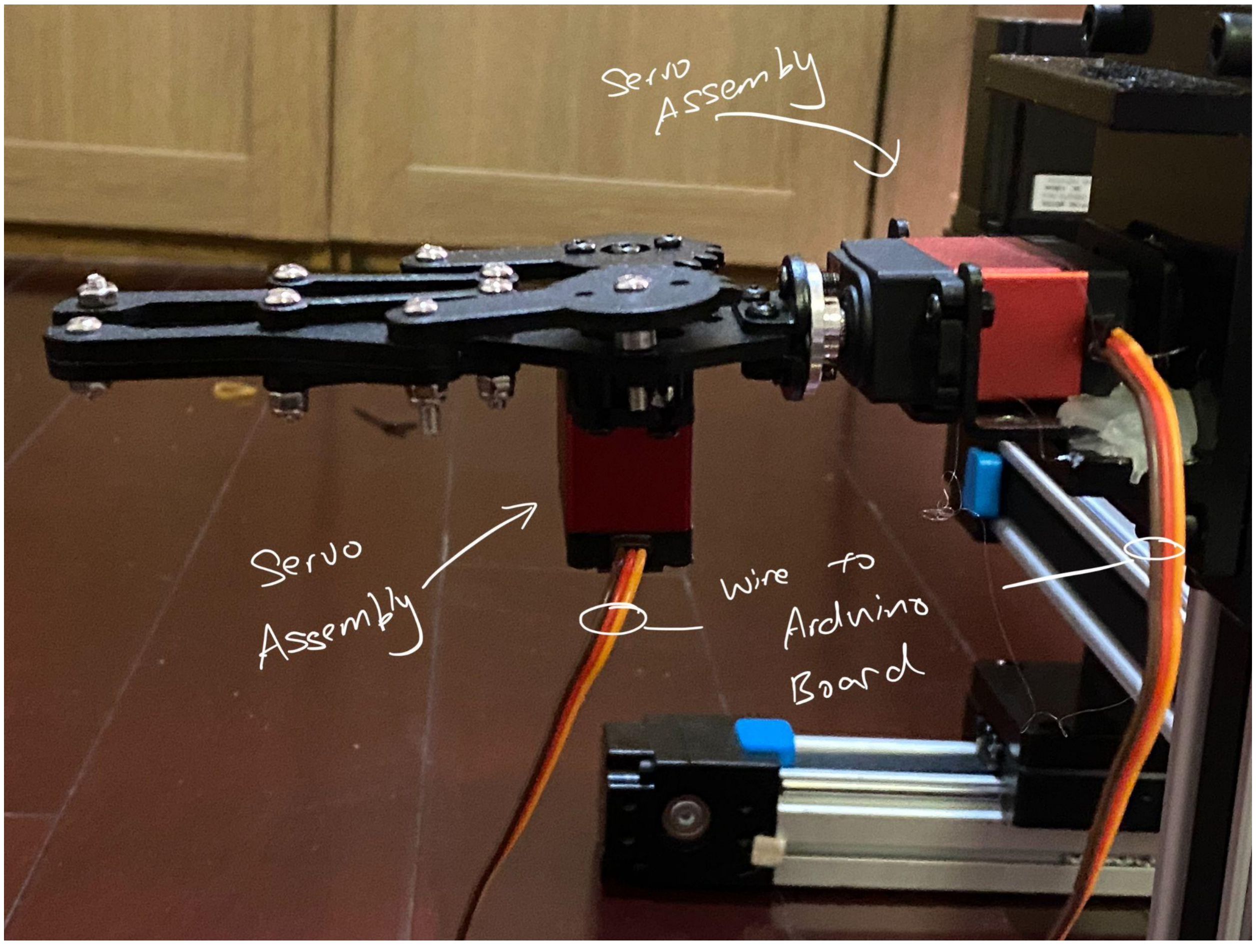


Fig 7. Assembly of Servo

Arduino Demo

Now that we have assembled the whole set-up. We can begin to control our robot. Code snippet shown below showcases what we used for our testing experiment. Note that in this case we did not connect the servo.

Sample purpose: Draw vertical and horizontal line, then at the starting position draw a rotated circle.

<u>All the function used are arduino functions and please refer to { Stepper.h } library for comments and more details.</u>

CODE:

```
// Include the Arduino Stepper Library
#include <Stepper.h>

// Define Constants

// Number of steps per output rotation
const int STEPS_PER_REV = 800;

// Create Instance of Stepper Class
Stepper stepper_ud(STEPS_PER_REV, 8, 9, 10, 11); // Note here we only care about location
1,3 in this (1,2,3,4) input since 2, 4 are just 5V and it does not matter where we put it
Stepper stepper_lr(STEPS_PER_REV, 12, 3, 13, 4);// ud - up and down (pin 8 dir, 10 puls) ; lr -
left right (pin 12 dir, 13 puls); fb - front back (pin 5 dir, 7 puls)
Stepper stepper_fb(STEPS_PER_REV, 5, 1, 7, 2);
void setup() {
  // nothing to do inside the setup
}

void loop() {

  stepper_ud.setSpeed(100);// set speed for all motors, input is rpm.
  stepper_lr.setSpeed(100);// ex. set left right motor to 100 rpm
  stepper_fb.setSpeed(50);

  stepper_fb.step(-800); // move forward 800 steps
  delay(100);// we like to allow some delay between command to prevent jerking
  stepper_lr.step(800); // move right 800 steps
  delay(100);
  stepper_lr.step(-1600);// move left 1600 step
  delay(100);
  stepper_lr.step(800); // move right 800 steps
  delay(1000);
  stepper_ud.step(800); // move up 800 steps
  delay(100);
  stepper_ud.step(-1600); // move down 800 steps
  delay(100);
  stepper_ud.step(800); // move up 800 steps
  delay(1000);

  stepper_lr.step(5000); // move right 5000 steps
  delay(100);
  stepper_lr.step(-5000);  // move left 5000 steps
  delay(100);
  stepper_ud.step(3000); // move up 5000 steps
  delay(100);
  stepper_ud.step(-3000); // move down 5000 steps
  delay(100);

  // Draw a square from center
  for (int x = 0; x < (STEPS_PER_REV * 3); x++ ) {
    stepper_ud.step(1);
    stepper_lr.step(1);
  }
  delay(100);
  for (int x = 0; x < (STEPS_PER_REV * 3); x++ ) {
    stepper_ud.step(1);
    stepper_lr.step(-1);
  }
  delay(100);
  for (int x = 0; x < (STEPS_PER_REV * 3); x++ ) {
    stepper_ud.step(-1);
    stepper_lr.step(-1);
  }
  delay(100);
  for (int x = 0; x < (STEPS_PER_REV * 3); x++ ) {
    stepper_ud.step(-1);
    stepper_lr.step(1);
  }
  delay(100);
  stepper_fb.step(800); // move back 800 steps
  delay(10000);
}
```

References

All the code and pictures are from me, therefore no references are needed for this document.