**The Ball Balancer Build Guide**

***By Eeman Aleem and Jake Jo***

***WATiMake Co-op Report Spring 2024***

Table of Contents

[Introduction 2](#_Toc175053854)

[CAD Model 2](#_Toc175053855)

[Electrical Setup 3](#_Toc175053856)

[Software 6](#_Toc175053857)

[References 7](#_Toc175053858)

# Introduction

This project uses a 3-RPS Parallel Axis Manipulator, computer vision with OpenCV, and PID controls to move a platform and prevent a ball from falling off of it. Notable hardware includes 3D printed/laser-cut pieces for the 3RPS manipulator and camera stand along with motors+drivers, a webcam, an Arduino, and Raspberry Pi to perform the ball tracking, PID control, and motor movement.

# CAD Model

**3RPS Parallel Manipulator**

The design for the robot is loosely modified from another ball balancer by Aaed Musa from as seen in [1]. The 3D printed parts of this 3RPS manipulator consists of:

* + **1 x** base plate
  + **3 x** link 1
  + **3 x** link 2
  + **1 x** platform frame
  + **3 x** motor spacer
  + **6 x** tie rod spacers

Students can choose the size and colour of the top platform to match their needs and it can be glued or taped directly onto the platform frame. A full assembly can be found [here](https://github.com/EemanAleem/The-Great-Ball-Entrapment/blob/main/CAD/Parallel%20Manipulator%20Assembly.zip).

**Camera Stand**

The 3D printed parts consist of

* + **2 x** base pieces
  + **2 x** top pieces
  + **1 x**  Slider for C270 camera

The laser-cut pieces consist of

* + **4 x** basswood piece 1
  + **2 x** basswood piece 2

A full assembly can be found [here](https://github.com/EemanAleem/The-Great-Ball-Entrapment/tree/main/CAD/Camera%20Stand/sldprt%20files)

# Electrical Setup

The following is the electrical equipment we used, along with their schematics:

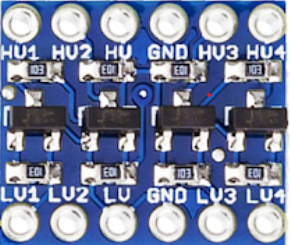
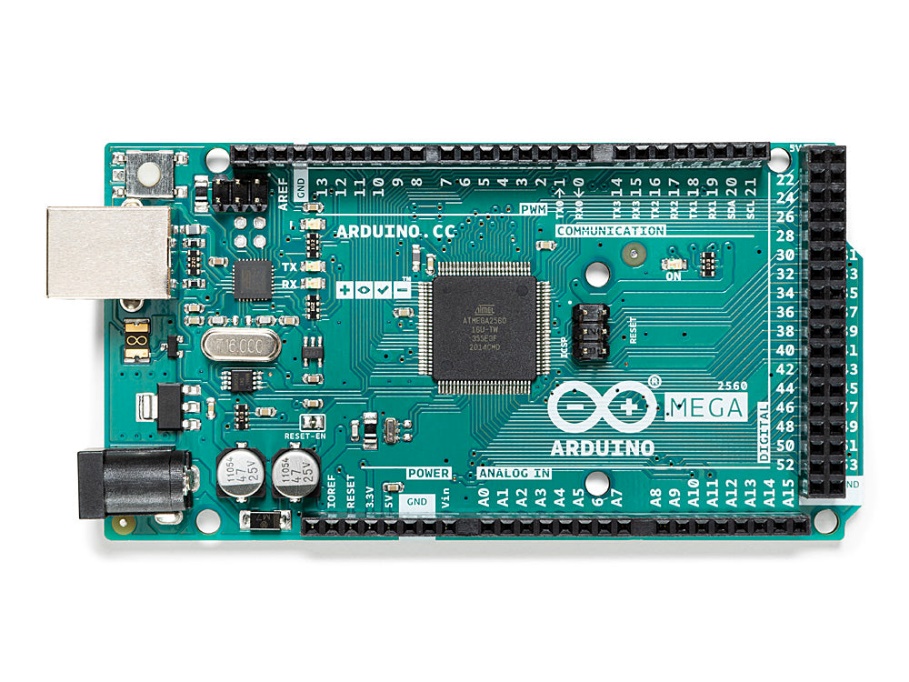
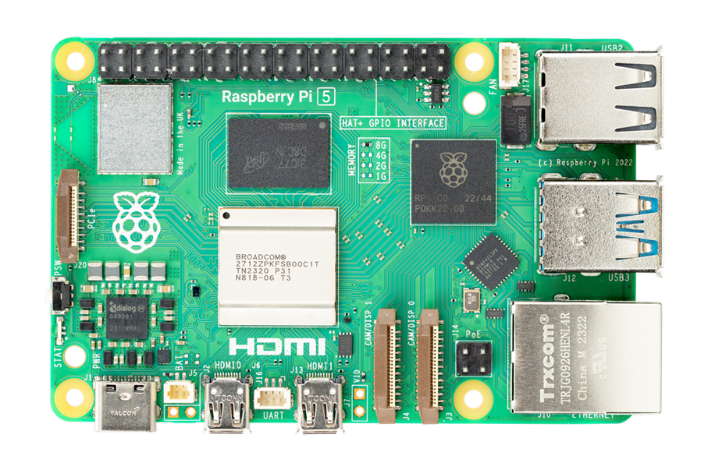
* + One Raspberry Pi 5 with 4GB RAM
  + One 12GB MicroSD card
  + One power supply cable + HDMI cable for the Pi 5
  + One Arduino Mega 2560 Rev3
  + One USB cable for the Mega
  + One Logitech Webcam
  + Three TB6600 Micro-Stepping Motor Drivers
  + Three NEMA-17s (any version will suffice)
  + One 3.3V-5V 4-channel bidirectional logic level shifter
  + One breadboard
  + Four female-male jumper wires
  + Thirty-Four male-male jumper wires
  + One keyboard
  + One mouse

Below are the schematics to connect an Arduino to a Pi with the level shifter and how to connect a motor to the TB6600 driver to the Arduino

|  |  |
| --- | --- |
| 3.3V-5V LLS | Raspberry Pi 5 |
| LV2 | GPIO 3 (SDA) |
| LV1 | GPIO 4 (SCL) |
| LV | GPIO 1 (3.3V) |
| GND | GPIO 20 (GND) |

|  |  |
| --- | --- |
| Arduino Mega 2560 | |
| D20 (SDA) | HV2 on LLS |
| D21 (SCL) | HV1 on LLS |
| 5V | HV on LLS |
| GND | GND on LLS |

Here, the 3.3V-5V Logic Level Shifter is referred to as a ‘LLS’.



Logitech Webcam

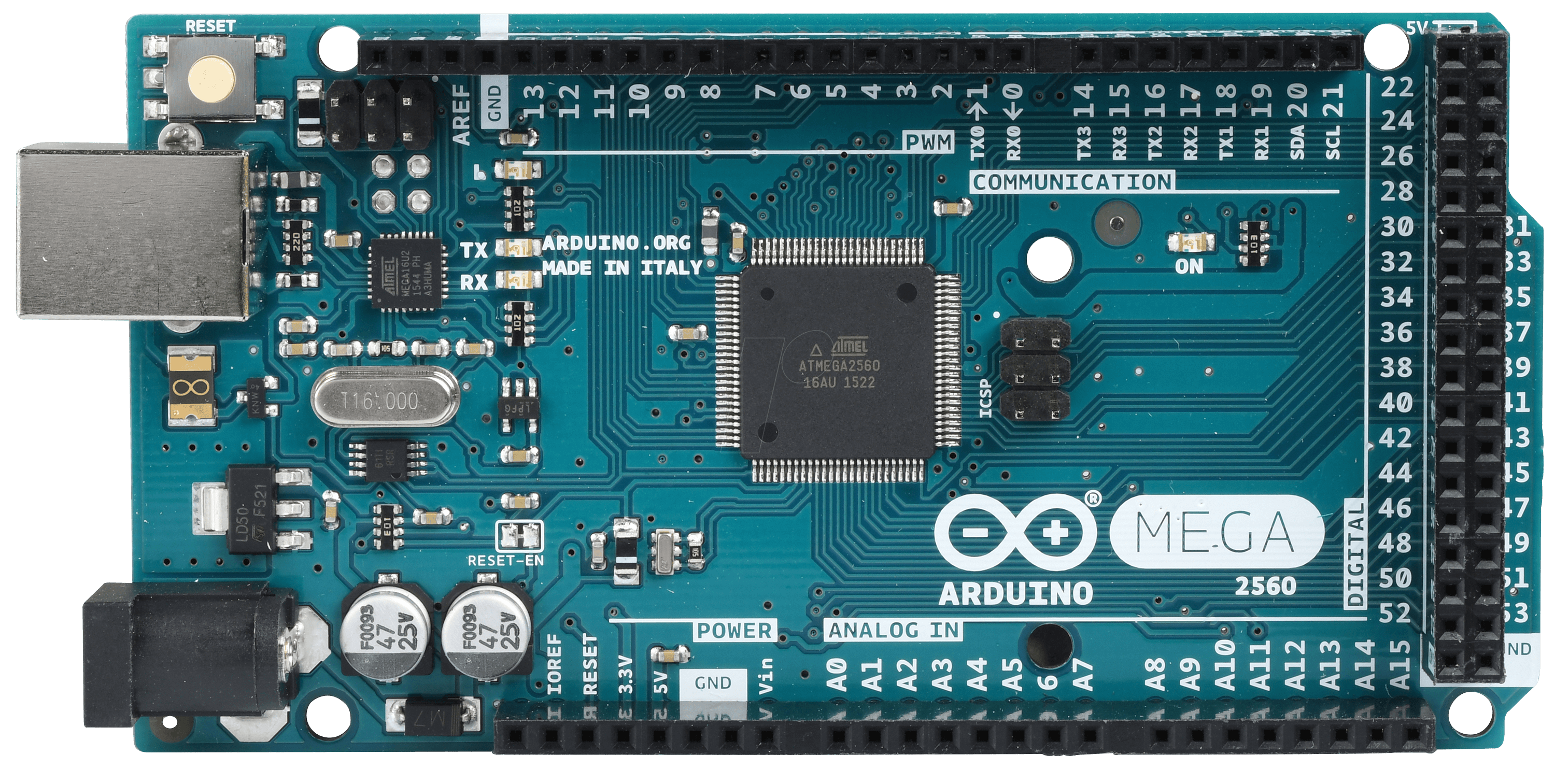
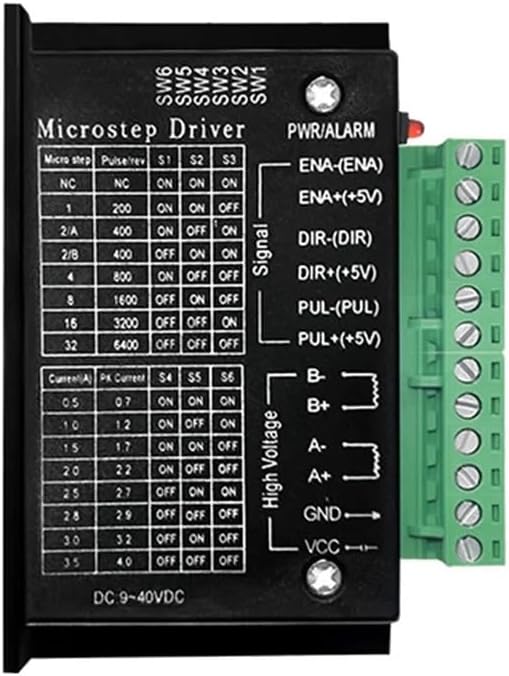
Mouse & Keyboard

Power Supply

HDMI Cable

PC

Repeat the following schematic for each motor but use different Arduino pins to connect the Arduino to the TB6600 driver



|  |  |
| --- | --- |
| Arduino Mega 2560 | |
| D2 | TB6600 DIR+ pin |
| D3 | TB6600 PUL+ pin |
| D4 | TB6600 ENA+ pin |
| GND | TB6600 ENA- pin |

|  |  |
| --- | --- |
| TB6600 Motor Driver | |
| GND | Use multiple wires to connect PUL-, DIR-, ENA-, and Arduino GND pin |
| ENA+ | Arduino D4 pin |
| DIR+ | Arduino D2 pin |
| PUL+ | Arduino D3 pinr |
| B- | NEMA 17 Coil 2 (6th pin) |
| B+ | NEMA 17 Coil 2 (4th pin) |
| A- | NEMA 17 Coil 1 (2nd pin) |
| A+ | NEMA 17 coil 1 (1st pin) |
| GND | PSU GND terminal |
| VCC | 9-42 VDC |

The 6 slots on the motor are referred to (from right to left) as pins 1 through 6 in the table on the right.

# Software

This project uses 2 programs, one on the Raspberry Pi, and one with the Arduino. Fully commented code can be found [here](https://github.com/EemanAleem/The-Great-Ball-Entrapment/tree/main/Integration) but as a summary, here is a simplified version of what each program does:

**The Python Program on the Raspberry Pi:**

1. Uses the webcam to detect the yellow ball's center coordinates in the x-y plane in reference to the balancing platform. This is done in the 'detect\_yellow\_ball' function.
2. Uses PID control to calculate the error between the ball's current position and its target position in both the x and y direction, done in the 'PID' function.
3. Along with a desired height to keep the platform (in our case it is 4.5 inches), a 3D vector to move the ball is generated.
4. To match this vector, inverse kinematic calculations are implemented in the 'theta' function to output each stepper motor leg positions. For reference, the calculations are also taken from [1] and slightly modified to fit the dimensions of this balancer.
5. The position data for each of the stepper motors is then sent via I2C to the Arduino Mega, byte by byte. This is done in the 'SendData' function.
6. The processing only stops when the yellow ball is no longer present. The program can be ended by pressing on lowercase 'q'.

**The Arduino File:**

1. Performs setup including setting up I2C comms (and itself as a client/slave) and homes the motors.
2. Once a data transmission is received by the Pi, the receiveEvent function is called. Once two bytes are received, byte merging / byte concatenation is performed to get the original two-byte integers.
3. Data is then stored in their respective indexes in the pos[3] array which stores the stepper motor positions.
4. The speed of each motor is calculated depending on how far the motor currently is from its target.
5. The motors then move according to their received positions until data transmissions are received again.
6. This code is looped.

# References

[1] “Ball Balancer,” *Aaed Musa*. https://www.aaedmusa.com/projects/project-three-sng7y-gaslp

‌