Team 7

Assignment name: Capstone Project Report

Date: April 16, 2021

Team members: Abdulrahman Alzimam: asa635@nau.edu

Quincy Ahuja: qka2@nau.edu

Siqi Peng: sp2286@nau.edu

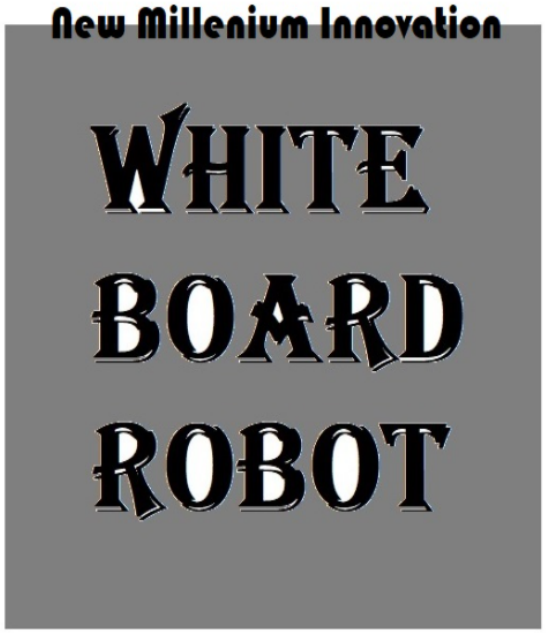


Table of Contents

Introduction

For this project, clients want to work on a robot that will control the robot, it will read and write over the whiteboard and can transfer the whiteboard through the scanning to the website servers that will be available across the globe. The main purpose of this project is to design such a whiteboard which can act as a robot and perform different tasks which can help in distance learning or in online courses. With the advancement in technology, new inventions are developing in the world and people's needs are changing over the time. Now people want to have as much facilities as they can get, and therefore the demand for robots has increased. Now robots are working in every field of life and soon they are taking places of humans in many working fields. Along with that robots can help in the educational field. In this field robots can do many types of things like from teaching to other physical activities. One of the options is to use the robot for operating the white board, as white board uses in all types of classrooms, and teachers write on the white board to explain the things to the students. This whole process can transform into a digital world where a robot will write on the white board. Now coming to another feature, that is removing or cleaning the white board and this is one of most tedious work to do as a human, so if a robot will do it would be better, therefore writing on the white board and cleaning the white board are two things that can be done by the robot.

This a new project and even though some sort of white board cleaner and writers have made before but this project is based on its own new concept. It is solving the problem for scanning of board, available on the white board for learning programs from distances. The client Dr. Winfree has some requirements to fulfill in this project by the robot, the machine must be able to write on a whiteboard and the robot must not make too much noise while writing. The robot may make slightly more noise than a whisper so the dB noise output of the system must be below 40 dB 1.2. Needs controllable marker placement that the user can specify. The robot must control marker location within an acceptable error of less than 1 cm in either direction 1.3 The machine must be able to use at least one marker at a time. The machine will prioritize accuracy over speed. The machine must be able to erase the whiteboard. Must be able to erase specific areas, not just all of it at once. The machine must be able to scan and save the whiteboard. Scan must not capture surroundings of the white board. Scans must be accessible for users to download and save as bitmap files through the web application. All functions must be accessible remotely to the client through an online web application. The machine must have a stable preferably ethernet connection. The web app will have a virtual draw & erase tool, scan and save tool, and can check to see what is already on the whiteboard. The power supply for the robot must be capable of doing AC-DC conversion. The power supply must be able to plug directly into the outlet. All connections must be covered for safety purposes. The machine must not get in the way of someone trying to locally access the whiteboard. The machine will be fixed to the wall and border of the whiteboard. The cost of developing the machine must not exceed $500. The machine must be compatible with multiple size whiteboards, including 4’x4’ and 4’x8’ sized

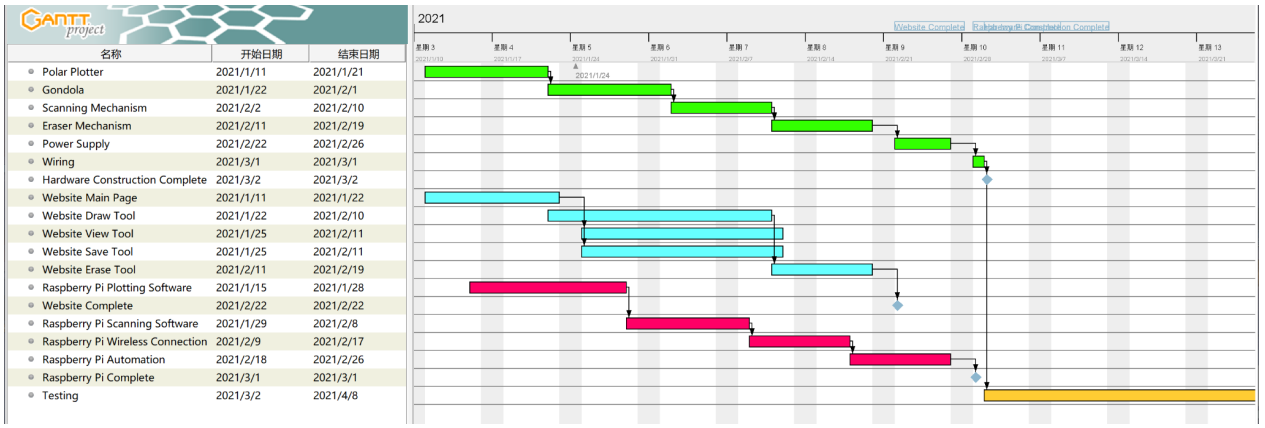
whiteboards.

Design Process

Overall Design Process

After we received the customer's work requirements, first of all, we discussed with the customer about the general demand direction, understood the tasks to be completed by the project, and understood the general technical requirements, project structure and project background of the project. Team members are familiar with each other and understand the technical solution of the project, and prepare the corresponding development plan according to the progress of the project and customer demand time.

The team members then conduct research related to the project and investigate the current research background. Through technical weekly, magazine, or paper, understand the current research results of predecessors, and according to the existing information, decide where to start the project work. Once we had a clear direction, we started weekly team meetings to make sure the details of the team's execution were agreed on, and to communicate the current status of the project.



名称	开始日期	结束日期
Polar Plotter	2021/1/11	2021/1/21
Gondola	2021/1/22	2021/2/1
Scanning Mechanism	2021/2/2	2021/2/10
Eraser Mechanism	2021/2/11	2021/2/19
Power Supply	2021/2/22	2021/2/26
Wiring	2021/3/1	2021/3/1
Hardware Construction Complete	2021/3/2	2021/3/2
Website Main Page	2021/1/11	2021/1/22
Website Draw Tool	2021/1/22	2021/2/10
Website View Tool	2021/1/25	2021/2/11
Website Save Tool	2021/1/25	2021/2/11
Website Erase Tool	2021/2/11	2021/2/19
Raspberry Pi Plotting Software	2021/1/15	2021/1/28
Website Complete	2021/2/22	2021/2/22
Raspberry Pi Scanning Software	2021/1/29	2021/2/8
Raspberry Pi Wireless Connection	2021/2/9	2021/2/17
Raspberry Pi Automation	2021/2/18	2021/2/26
Raspberry Pi Complete	2021/3/1	2021/3/1
Testing	2021/3/2	2021/4/8

Figure 1: Our initial Gantt Chart schedule

The team communicated and reported the progress of the project every week, coordinated the contents responsible by the team members according to the product requirements and planning improvement, and started to create the design prototype as planned.

After we design the prototype, we start to do the technical assessment, risk assessment, to ensure that the customer's requirements are met, and the project design is as beautiful and brief as possible. After confirming that there was no problem with the prototype design, we started to build the prototype, purchased the required materials, and started to write the required codes and complete the required circuit design.

Integrate the built prototype into subsystems, and start testing the subsystems. We will feed back the results of the test to the customer. And according to the customer's wishes and test results, to the best of our ability, redesigned and updated our engineering system.

Functional Decomposition

The base of our design uses a polar plotter that moves a gondola suspended between two motors around the whiteboard. The gondola contains the marker and eraser, as well as electronics to control the positioning of them against the whiteboard. The marker position against the whiteboard is controlled by a servo, while the eraser position is controlled by an electromagnet embedded in the eraser. The polar plotter motors are operated by a Ramps 1.4 motor control board controlled by a Raspberry Pi.

The Raspberry Pi has an open-source polar plotting software, Makelangelo, installed to perform image processing and send commands to the polar plotter. An HD webcam is also plugged into the Raspberry Pi and mounted on a servo-controlled arm above the whiteboard to extend and retract the camera when needed. The Raspberry Pi is remotely accessible anywhere via VNC Viewer, allowing users to use the robot from anywhere.

So the function of the finished project can be divided into：

- Write on whiteboard
- Erase on whiteboard
- Operate remotely via internet
- View whiteboard remotely
- Save images of whiteboard
- Scan the whiteboard
- Accurate within 1cm
- Does not interfere with local use of white board
- Work on any size whiteboard

Prototype findings

The prototype of the polar plotter section, which was the key to the entire design, was also the most complex mechanical part. The prototype we completed can control the two stepper motors needed to move the Gondola. This prototype reduces the risk of the project by identifying the most difficult and critical aspects of the project (mechanical and electrical). However, after we finished the prototype, we encountered the problem of a damaged stepper motor driver. In the final product design, we replaced and upgraded the stepper motor so that it could work normally

In the prototype of the web page, most of what we wanted to achieve was achieved. The web page had written and erased functions. At the same time, we expect to be able to directly upload our drawn picture files to Makelangelo, so as to

reduce the step of secondary selection through Makelangelo and facilitate operation. Unfortunately, in the subsequent design, we failed to do this. The web page could not be called and requested by the Makelangelo like a mobile application, and could not always be hung in the background waiting for the Makelangelo to send a picture application to the background of the web page. We tried to modify the Java program to do this, but failed.

Final Design

We modified the prototype many times, integrated the prototype into a subsystem, tested it many times, and used the feedback we got from presenting the results to our customers to modify our project many times. For example, we initially planned to control the robot via the website, but due to the complexity of the operation required to connect the website to the robot, we finally decided to switch to remote control via the VNC viewer.

Our final design is as follows:

For the whiteboard robot, all robot functions are controlled by Raspberry Pi. Our customers can remotely control the robot by accessing the Raspberry Pi through a VNC viewer. The polar plotter is the basis of our design, which uses two stepper motors mounted in the top corner of the whiteboard and a gondola suspended between them to achieve polar plotting. The gondola is mounted with a marker and an electromagnet eraser and the gondola is propelled by a motor, which is controlled by a RAMPS 1.4 motor control board, to track the image path. The image that needs to be drawn is drawn on the website, saved to the local, and then uploaded and processed into G code through Makelangelo. The image is drawn on the whiteboard by Raspberry Pi. As for the camera, which takes pictures and uploads them to the web for viewing, is mounted on a retractable arm above the whiteboard.

Our whiteboard robot can be extended to any size whiteboard.

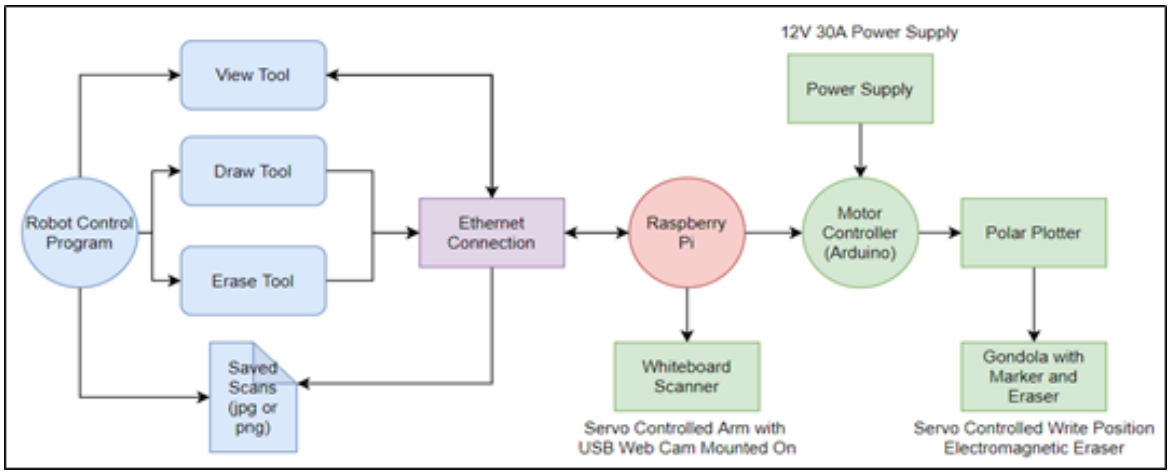


Figure 2: Flowchart

On the whole, the project is divided into two parts. One part is the software content, which controls the writing content of the whiteboard robot on the computer, and the other part is the hardware part, that is, the whiteboard robot writes on the whiteboard.

These two sections are subdivided into nine different subsections. We call the whiteboard robot Raspberry Pi. For the hardware part, the technical points that need to be achieved are as follows: Power Supply, Motor Controller, Polar Plotter and Marker/Eraser, Whiteboard Scanner.

By using coordinates and a robotic arm to achieve the function of drawing on the whiteboard, we have a Gondola suspended between two motors moving around the whiteboard. Therefore, for the realization of this part of functions, the first thing we need to consider is the power supply. Only after the power system that can guarantee the operation of Raspberry Pi can we control the motor and provide the basis for the flexible rotation and writing of the manipulator arm.

A cable car mounted on the robotic arm is fitted with a marker and an electromagnetic eraser, and an electronic device that controls their position on the whiteboard.

The position of the marker on the whiteboard is controlled by the server, and the position of the eraser is controlled by an electromagnet embedded in the eraser. For this part, we need to provide a Polar plotter. Only with this coordinate and position information can Raspberry Pi write and draw or erase normally. In addition, an important technical point is that we need a camera that can transmit the existing content on the whiteboard to the network control side. So our Raspberry Pi also has a high-definition webcam mounted on a servo-control arm above the whiteboard, which can be retractable if needed to better identify the whiteboard content.

For the software part, the technical points that need to be done are view tool, canvas draw tool, erase tool and saved scan tool. The software is divided into two parts. The first part is the software that controls Raspberry Pi, called Makelangelo. The second part allows us to draw and save graphics on computers or mobile phones, and then upload them to Makelangelo, and then Raspberry Pi draws patterns.

Raspberry Pi has an open source polar coordinate drawing software, Makelangelo, which has many functions. We can select the pictures we have drawn through this software, upload them to the software, perform image processing, convert the pictures into G code and save them into SVG files. Then send the command to the polar plotter, so that the Raspberry Pi accepts the command and draws.

At the same time, we need a third terminal, which serves as a bridge between the operator and the whiteboard robot, to realize the functions of Canvas Draw Tool, Erase Tool and Saved Scans Tool. We chose to use web pages as the bridge because web pages are more adaptable and do not differentiate between required devices.

Results

After the completion of the project, we have a total of eight-point test items, subdivided the content of the twenty-one-point test. The tests were carried out in different fields, such as the robot's hardware precision, functional integrity, control difficulty level and external influence factors of the robot design.

We have 11 inspection tests, 3UTS tests, 2UTM tests and 4 integrated tests, estimated use at least two weeks to test all of them.

Below is a figure of our system architecture. The red rectangles represent the subsystems we tested.

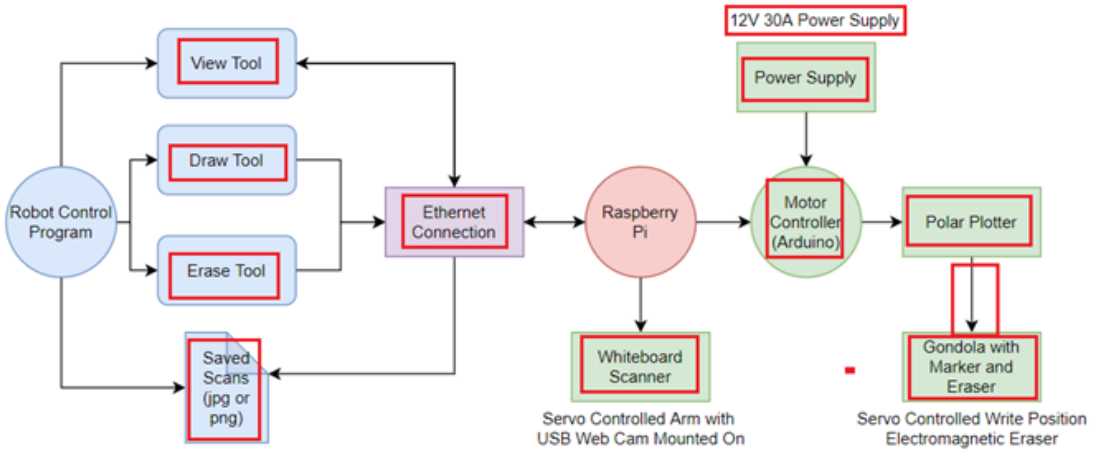


Figure 3: Design Architecture

Figure 4 below shows a list of all of the requirements, and their testing status. Green indicates a pass, yellow indicates a partial pass, and red indicates a failure. The most important requirements are marked with an asterisk.

	A	B	C	D
1			Instructions: List all of your requirements, and use a numbering system.	
2				
3	Type of Test	Status	Req #	Requirement
4				
5	Integrate	*	1	The machine must be able to write on a whiteboard
6	Inspect		1.1	The robot must not make too much noise while writing
7	UTS		1.1.1	The robot noise output of the system must be below 60 dB
8	Inspect		1.2	Needs controllable marker placement that the user can specify
9	UTS		1.2.1	The robot must control marker location within an acceptable error of less than 1 cm in each direction, and should prioritze accuracy over speed
10	Inspect		1.3	The machine must be able to use at least one marker at a time
11	Integrate	*	2	The machine must be able to erase the whiteboard
12	UTM		2.1	Must be able to erase specific areas not just all of it at once
13	Integrate	*	3	The machine must be able to scan and save the whiteboard
14	Inspect		3.1	Scan must not capture surroundings of the white board
15	UTS		3.2	Scans must be accessible for users to download and save as picture files
16	Integrate	*	4	All functions must be accessible remotely to the client though the internet
17	Inspect		4.1	The machine must have a stable preferably ethernet connection
18	Inspect		5	The power supply for the robot must be capable of doing AC-DC conversion
19	Inspect		5.1	The power supply must be able to plug directly into the outlet
20	Inspect		5.1.1	All connections must be covered for safety purposes
21	Inspect		6	The machine must not get in the way of someone trying to locally access the whiteboard
22	Inspect		6.1	The machine will be fixed to the wall and/or border of the whiteboard
23	Inspect		7	The cost of developing the machine must not exceed $500
24	UTM		8	The machine must compatible with multiple size whiteboards
25				
26				

Figure 4: Requirements, status, type of test

Marked with an asterisk are:

1. The machine must be able to write on a whiteboard. This is the most basic part that our robot needs to have. Only if it can write, can it have the additional function behind. If this is not possible, our design is a complete failure.

2. The machine must be able to erase the whiteboard. This is an important additional function of our robot. When we finish writing a whiteboard and need to write new content, this function is necessary. This feature would better allow our whiteboard to be used remotely, without the need to wipe the whiteboard.

3. The machine must be able to scan and save the whiteboard. This function allows our robot to have the ability see what is on the whiteboard and save it. This let the user know what is already there before starting a drawing and to save it.

4. All functions must be accessible remotely to the client through the internet. The main feature of the robot is it can be used anywhere from the internet. This helps with remote learning and convenience.

Type of Tests

We have 4 types of tests that we performed on the robot. The first type is a unit matrix test (UTM). We used this type of testing when we were testing for the same types of variables and inputs. Each input was expected to yield a certain output, and changing the input should yield a predictable output. We chose UTMs to test the polar

plotter positioning, eraser positioning, and testing multiple sized whiteboards. The 2nd type of testing we did is Step by Step testing (UTS). We chose UTS when it made sense to test for different subsystem path execution. We chose UTSs to test the accuracy of the robot, the noise level, and to test the scanning/saving subsystems. The 3rd type of tests we did were inspection tests. We did these tests on requirements that could be visually completed without any procedures. We inspected the cost, the robot mount, the accessibility to the whiteboard, the safety, the power supply, the ethernet connection, the scan captures, the marker usability, having controllable marker placement, and excessive noise. The final type of testing we did was integration testing. We used this type of testing to make sure the subsystems work together the way they are supposed to. We used integration testing to test being able to write on the whiteboard, being able to erase the whiteboard, being able to scan and save the whiteboard, and testing that all functions are usable through the internet.

Major Tests and Analysis of Results

One major test we did was the polar plotter accuracy test. This test is to validate the requirement that the robot draws accurately within 1 cm of the picture. This test was done by generating a 10x10 cm square for the robot to draw. The square was then measured to see if it was within the required accuracy range. The test failed, as the sides of the square were not drawn straight within 1 cm (see appendix B for result). This is due to the motors being out of sync. We also noticed that drawing a larger image usually resulted in greater accuracy. Because of this, we suspect the power supply does not always supply enough power.

Another major test we did was the noise test. While drawing a sample image, we placed a phone with a noise dB meter 3 feet from the robot. This tested the requirement that the robot not be louder than 60 dB. The robot passed the test, as the loudest recorded noise was 59.3 dB, with the average noise being 53.2 dB.

Another major test we did was the integration test of requirements 1, 2, 3, and 4. This tested that all the subsystems worked together so that the robot could write on the whiteboard, scan and save the whiteboard, and erase the whiteboard through the internet. This was done by connecting to the Raspberry Pi through VNC Viewer, uploading a sample image, and drawing it with the robot.

Once finished, we scanned and saved the whiteboard, and then erased it. All of these procedures worked except the eraser. Unfortunately, the eraser does not make enough contact with the whiteboard despite having an electromagnet inside. This could be fixed with a stronger electromagnet or other device to increase contact with the whiteboard. Overall, the project performed decently under the testing. Some of the requirements were not met exactly, especially the accuracy of the robot and the erasing mechanism. However we feel that both of these issues could be fixed by using a stronger power supply and redesigning our eraser mechanism.

Conclusion of Capstone Project

The project was to develop a whiteboard robot that will be remotely controlled, and it will be able to scan the whiteboard through the cameras and upload the scanned files to the user and it can also read and write over the whiteboard as well. The project has been completed and the design has been implemented with a polar plotter that can move the marker around the whiteboard to draw images. The system has a camera scanner that is fitted above the whiteboard to scan the complete board which the user can save if they desire.

The remote control is done by controlling the Raspberry Pi with VNC Viewer from anywhere. The robot is fully operational from the Raspberry Pi, allowing users to control all functions of the robot remotely..

The design is operational and it has been implemented. Most of the project requirements were met, and for the most part the robot is able to be used as the client intended.The learning obtained from this course regarding the designing is quite useful for all the team members and this learning can help in future in professional work.

Unfortunately, several things went wrong during this project. The power supply we used was not sufficient to properly power the motors, and resulted in accuracy issues. The eraser mechanism was also ineffective. We could improve the project by getting a better power supply and redesigning the eraser mechanism. We also changed several requirements to make them more reasonable. For example, we raised the max noise from 40 dB to 60 dB. We also changed requirements regarding the internet controllability. We originally intended to have an internet browser application to control the robot. However, this turned out to be not necessary and too difficult to implement. We also learned that testing is useful in design, as it shows the flaws that need to be changed.

User Manual

Introduction

Thank you again for choosing New Millenium Innovation to design and create a product that will hopefully help you and others more effectively educate other students. We have confidence that our system will allow for greater utilization of whiteboards by enabling the remote monitoring and use of them in classrooms. These features include:

- Remotely drawing pictures and text on your whiteboard
- Remotely viewing your whiteboard
- Easily saving your whiteboard contents for future viewing

This user manual is for you to reference as needed in order to properly set the White Board Robot up, connect to the robot, and utilize the robot. We hope you can benefit from it for many years to come!

The focus of this project was to create a device that would allow our client to easily and effectively use his whiteboard remotely and autonomously.The device should also not inhibit the local use of the whiteboard when not in use. Finally, the device should be adaptable to all common sized whiteboards. These were the main criteria that influenced our design of the robot.

The design itself is comprised of 5 main subsystems; a polar plotter, a gondola, a retractable camera, polar plotting software, and robot controller. The polar plotter is the foundation of our design, and consists of 2 NEMA 17 stepper motors mounted on the top corners of the whiteboard, two counterweights each weighing half the weight of the gondola, and a motor controller. The motor controller is responsible for handling all the control signals and power to the motors. The gondola is suspended between the two motors using GT2 2 mm pitch timing belts, allowing the motors to precisely move the position of the gondola around the whiteboard. The writing marker is housed on the gondola, along with the eraser and electronics (a servo motor for the marker and an electromagnet for the eraser) to control their position against the whiteboard. In order to view the whiteboard, an HD webcam is mounted above the whiteboard on a retractable rotating arm. The arm is rotated on command by the user via a servo motor, ensuring the camera can view the whiteboard when desired and remain out of the way when not in use. In order for the robot to actually draw images, the polar plotter must move the gondola around the whiteboard in the shape of the image, and the gondola-mounted servo must move the marker against and away from the whiteboard as necessary. To achieve this, we used an open-source software called Makelangelo to perform the necessary image processing and translation into G-code for the motor controller to understand. Finally, we used a Raspberry Pi to control all functions of the robot remotely. The Pi has all the command scripts to control the robot, as well as Makelangelo installed. Users are able to remotely access the Pi desktop via VNC Viewer and use the robot as desired.

Coming up with our design involved several steps. We first conducted research on drawing machines, and found there were already similar devices, though none that fulfilled all of our requirements. We evaluated several types of drawing machines and decided to implement a polar plotter as our basis, as it was the most

mechanically simple drawing machine to construct and control. This required a motor control board, and there were several to choose from. After more research and much trial and error, we ended up settling on an Arduino MEGA with a RAMPS 1.4 motor control shield attached, as it interfaced well with the control software and provided all the pins we needed. We then began designing the control system and the scanning mechanism. Initially, we intended the robot to be fully controlled from a website browser application. However, the complexity of creating such a website and automating everything on the Raspberry Pi soon became apparent, and we opted to implement remote control with VNC Viewer and easily accessible and usable scripts on the Pi. We also initially wished to mount the camera on the gondola itself and scan automatically scan the whiteboard frame by frame. But again, the complexity of implementing this was too difficult given our time and other focuses. So instead we settled on a retractable arm above the whiteboard with a webcam mounted on the end to capture the whiteboard.

Installation

Below are the general steps for installing the robot on a whiteboard, powering up the robot, and getting set up for remote use. This is assuming that all subsystems are assembled and wired but not attached to the whiteboard or each other. See appendix C for pictures of the complete installation of the robot. See appendix D for proper wiring of stepper motors and servos.

1. Install the NEMA 17 stepper motors on the top corner of the whiteboard. Ideally, you would want the motors placed at least 6 inches above and 6 inches out from the edges of the whiteboard. Once mounted, suspend the gondola between the two motors; feel free to pull on the belts to bring the gondola and counterweights to an even height.
2. Mount the servo controlled arm above the whiteboard, with the servo arm facing upwards. Ensure you are placing in the right orientation, as the arm rotates counterclockwise to extend and clockwise to retract. Mount the camera on the end of the arm if it is not already so that it will be viewing the whiteboard at an optimal angle when the arm is extended.
3. Mount the motor control board above the camera arm. Connect the stepper motor connectors to the motors, then connect the gondola servo to the servo first set of servo pins on the motor control board, and connect the camera arm servo to the corresponding connectors. See appendix B for proper connection pictures.
4. Mount Raspberry Pi such that the motor controller, webcam, and servo control pin can be connected, then connect them. Plug the Pi power supply into the Pi, and connect ethernet cable to the Pi.
5. Connect the 5 V power supply connector to the Vcc servo pin on the motor controller (See appendix D). Power on the power supply.
6. In order to properly set up the robot, you must know the dimensions of the polar plotter and the whiteboard in mm. For the polar plotter width, measure from the inside of the one stepper motor to the inside of the other. For the

height, measure from the bottom of a stepper motor to the bottom of the whiteboard. These measurements will be needed in step 5 below.

This completes all of the hardware setup of the robot. These next steps detail how to set up the remote connection with the Pi to control the robot and set up Makelangelo.

1. VNC Viewer is required to control the robot. Go to https://www.realvnc.com/en/connect/download/viewer/ and download the proper installation for your device, and follow the instructions to install the program.
2. Once VNC Viewer is installed, you must sign in with the proper credentials to access the robot.
3. Upon sign in, simply double click "WhiteboardRobot" and it will automatically connect to the Pi, allowing you to interact with the Pi desktop to control the robot.
4. Using VNC Viewer, double click **launch_makelangelo.sh** and then **Execute in Terminal.** This will launch Makelangleo. Once it is open click **Connect** near the upper right corner and then **OK** to connect Makelangelo to the robot. NOTE: If prompted to update the software by Makelangelo **do not update!**
5. Once the robot is connected, click **Settings** under the **Connect** button (which should now say **Disconnect**). In the settings window, ensure that the model selected in the dropdown box is **Makelangelo (custom).** On the **Machine** tab, input the correct dimension of the polar plotter as measured in step 6 of the hardware setup, followed by the correct dimensions of the whiteboard on the **Paper** tab. Click **OK** to save and close the settings window.

In order for the robot to draw accurately, the gondola must be positioned near the bottom center of the whiteboard before starting the drawing (this is called the "home position". However, when not in use, the robot will be stored above the whiteboard. So, the gondola must be manually positioned correctly before each use. This can be easily done using the **Manual Driving** commands in Makelangelo. But on initial setup, you will need to measure how many times to push the proper button to position the gondola. Follow these steps to take the measurement:

1. Turn off the power supply powering the motor controller.
2. Pull on the counterweight side of the timing belts until the gondola is centered at the top of the whiteboard in its stored position.
3. Click the **Set Home** button in Makelangelo. Note the position of the gondola on the virtual Makelangelo whiteboard. This is where the gondola needs to be when starting a drawing.
4. Click the arrow next **Manual Driving** to show the manual driving commands. Click the **-100** button to lower the gondola until it is in the same position as previously shown. Note how many times you press **-100** as you will need to do this before starting every drawing.
5. You may wish to press the **Set Home** button after every **-100** move, as this will reset the position on the virtual whiteboard for you to reference.

NOTE: Since this is an approximate way to set the correct home position, it is recommended to set the paper size as at least 6 inches smaller in both dimensions of the whiteboard (not the machine!) than you measured to account for error in the above method.

Configuration and Use

The following steps outline how to use the robot once it is properly installed. This covers how to upload images and draw them, generate art and text, view the whiteboard, and scan and save the whiteboard.

1.
2. Uploading Images and Drawing Images
    - 2.1. Most common image file formats are supported (Note: All testing was done with .jpg and .png file types).
    - 2.2. Once you have an image (created by you or downloaded), save it on your local device to your preferred location. Open VNC Viewer, and hover your mouse near the top center of the viewing window until a toolbar pops out. In the toolbar, click on the “Transfer Files” button, then click on the “Send Files” button in the popup. Navigate to your picture and double click it. This will copy the image to the Pi desktop.
    - 2.3. If not already open, double click “launch_makelangelo.sh” on the Pi desktop to launch Makelangelo.
    - 2.4. Click “Open File…” go to /home/Pi/Desktop/ and double click the picture you uploaded.
    - 2.5. The picture will be loaded into Makelangelo, and a “Conversion Options” popup window will appear. This lists the image conversion options available in Makelangelo. Select the conversion type you desire in the dropdown box. Once selected, it will generate a preview of the whiteboard (it may take a moment), and give you more options to manipulate the conversion. Once you are happy with the conversion, click “OK”
    - 2.6. Set the gondola to the right position. Use the **Manual Driving** commands to set the gondola to its known home measured in the installation. If you wish, you can verify this using the viewing camera (see viewing instructions below).
    - 2.7. Click **Start**. The robot will draw the image. If you wish to store the gondola when it's done, use the manual driving commands.

3. Generating Text and Art
    - 3.1. Makelangelo is also capable of generating its own art and text. To do this select the **Generate Art** button.
    - 3.2. A **Generate Art** popup will appear. Select your desired type in the dropdown box. Once selected, it will generate a preview, and you can alter the images using the parameters shown in the popup.

3.3. To write text with the robot, select **Your Message Here** from the dropdown box. You can type text into the text box and edit it using the given parameters. **This is the recommended way to display information using the robot**.

3.4. Once you are satisfied with your text or art, close the **Generate Art** box and follow steps 1.6 and 1.7.

4. Viewing and Saving the Whiteboard

4.1. To view the whiteboard, navigate to the Raspberry Pi desktop in VNC Viewer.

4.2. Double click on **scan_whiteboard.sh.** This extends the camera, takes a picture, and then retracts the camera. The message "Capture complete" will appear in the terminal once complete. Close the terminal.

4.3. The picture is saved as **whiteboard_capture.jpg** on the desktop. Double click to view the picture.

4.4. If you wish to save the picture, navigate to the VNC Server icon in the status bar on the top right of the Pi Desktop. Right-click the icon and select **File Tranfer…** Then select **Send Files** on the popup. Navigate to home/pi/Desktop/ and double click the picture file. By default this will download the picture to your local desktop.

Maintenance

While we aimed to create a robust and reliable robot, it is certainly possible that maintenance will be required for upkeep. Currently, most of the wiring is done with jumper wires. Thus it is possible for wires to come loose during operation or over periods of time. We recommend periodically checking the wiring for any loose connections.

Another point of weakness is the servo controlling the camera arm. Since the servo is mounted at the center of the rotation of the arm, a lot of stress is put on it during operation of the camera. It is likely that it will need to be replaced in the future, or perhaps even upgraded to a stronger servo. The current servo we are using can be found here.

Since this is a whiteboard drawing robot, it requires markers. Unfortunately, the robot is not able to replace marker caps, leaving them prone to drying out. On top of this, drawing pictures usually requires a large amount of ink. This means that markers will need to be replaced frequently, and it is recommended to manually uncap and cap markers after each use.

It is also recommended to delete uploaded pictures from the Raspberry Pi after you are finished with them. The Raspberry Pi has a relatively limited memory, and it could fill up quickly.

The robot also relies heavily on Makelangelo. Newer versions of the software have been released, but after testing several newer versions, we settled on the currently used version due to its reliability with the motor controller being used, compatibility

with Raspberry Pi, and the ability to generate text. **Please do not update the software, even if prompted that newer versions are available.**

Troubleshooting

Below is a list of common problems we encountered while constructing the robot and potential solutions, as well as other possible problems with possible fixes.

Makelangelo will not connect with the robot

Ensure motor controller is plugged in to Raspberry Pi via USB cable. It is possible that when you connect, it will not be the default port shown in the dropdown. Try the other ports shown in the dropdown.

The robot will not draw/move even though it shows it is connected

Ensure the motor controller power supply is powered on and connected properly. Makelangelo will connect to the motor controller through the USB cable even if it is not receiving power from the motor controller power supply.

The drawing is crooked/stretched/deformed

Ensure you have input the correct machine dimensions as described in the installation. Sometimes Makelangelo will revert its settings to default, so it is recommended to check your machine dimensions every time you connect Makelangelo to the robot.

The image generated by Makelangelo is not what I wanted

Unfortunately, some images can be tricky to be effectively processed and drawn by Makelangelo. Go through each image conversion type as described in step 1.5 of Configuration and Use. It may take some experimenting to get what you want. Change the parameters around to see if you can make it more desirable. Otherwise, you will have to find, create, or generate different images.

The image I want to draw is multicolored; how do I do that?

Our robot does not currently support more than one marker at a time. If you really want too, some Makelangelo conversion options allow you to manually swap the marker for the correct color.

Conclusion

We hope this user manual was helpful and informative, and wish you the best with your new robot. It has been a pleasure to work with you in completing this project. While we are all moving on to professional careers, we would be happy to answer questions in the coming months to help you get the product deployed and operating optimally. We hope our robot serves to help advance the education of others, and also as a basis for even better versions of the robot in the coming years.

Appendix A

This appendix documents the code created by the team for the project. All code was implemented on the Raspberry Pi for users to use while controlling the robot. We created 4 simple control programs, 2 shell scripts and 2 python programs, documented below with a brief description for each.

launch_makelangelo.sh:

```
java -jar /home/pi/Downloads/Makelangelo/Makelangelo-7.23.0-with-dependencies.j$
```

This executable script simply launches Makelangelo from the Pi Desktop.

scan_whiteboard1.sh

```
#!/bin/sh
echo "Extending camera arm..."
python /home/pi/Pictures/move_cam.py
echo "Capturing image...:
fswebcam /home/pi/Desktop/whiteboard_scan.jpg
fswebcam /home/pi/Desktop/whiteboard_scan.jpg
echo "Retrating arm..."
python /home/pi/Pictures/retract_cam.py
echo "Finished"
```

This executable script runs the two python programs below, and also issues the commands to capture images with the camera.

Move_Cam.Py:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,GPIO.OUT)
camServo = GPIO.PWM(11,50)
camServo.start(0)
time.sleep(2)
duty = 8
camServo.ChangeDutyCycle(duty)
time.sleep(2)
camServo.stop()
GPIO.cleanup()
```

This python program controls the servo to extend the camera arm.

Retract_Cam.py:

```
import RPi.GPIO as GPIO
import time
```

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,GPIO.OUT)
camServo = GPIO.PWM(11,50)
camServo.start(0)
time.sleep(2)
duty = 0.5
camServo.ChangeDutyCycle(duty)
time.sleep(2)
camServo.stop()
GPIO.cleanup()
```

This python program controls the servo to retract the camera arm.

Appendix B

This appendix documents the results of our initial testing.

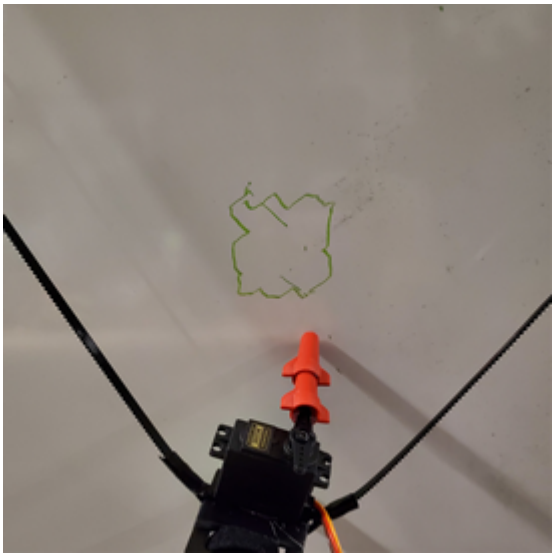


Figure 5: Accuracy Test Results. As demonstrated above, the robot was inaccurate due to the stepper motors undergoing irregular movement.

Appendix C

This appendix shows images of the completely setup device. Note that the eraser is not installed as it failed our requirements and is ineffective.

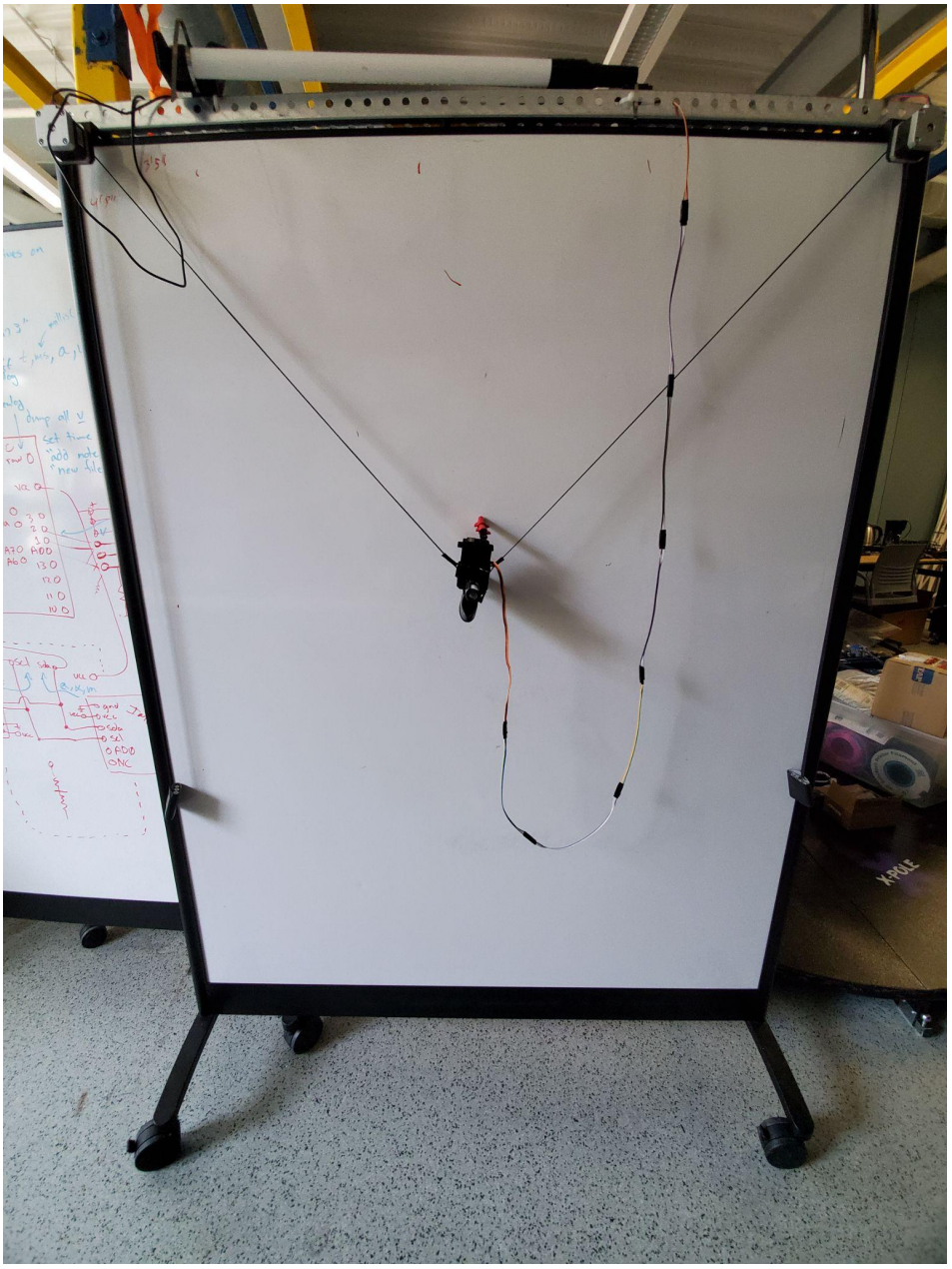


Figure 6: Complete setup of whiteboard robot, front view.

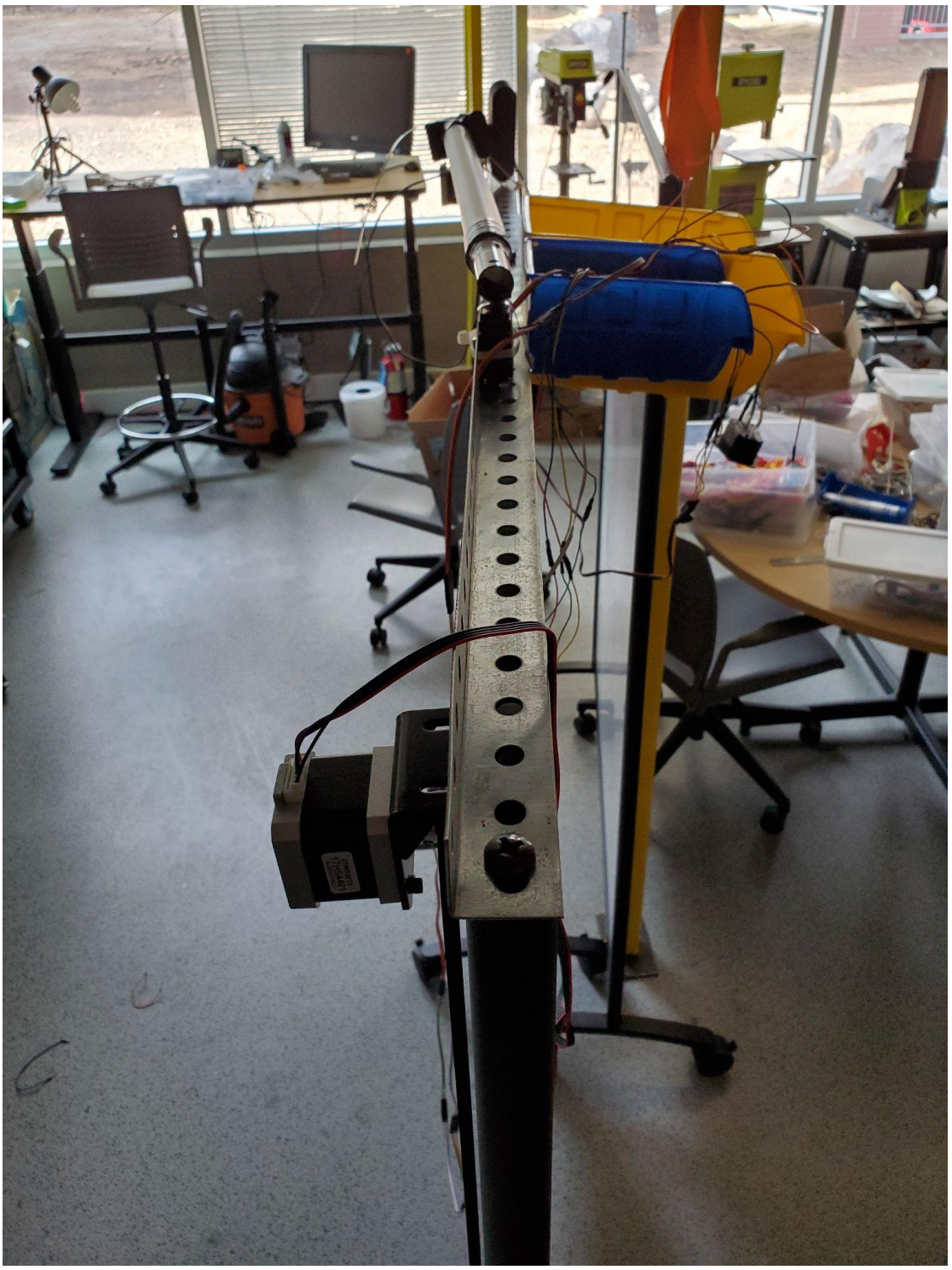


Figure 7: Complete whiteboard robot setup, top-side view

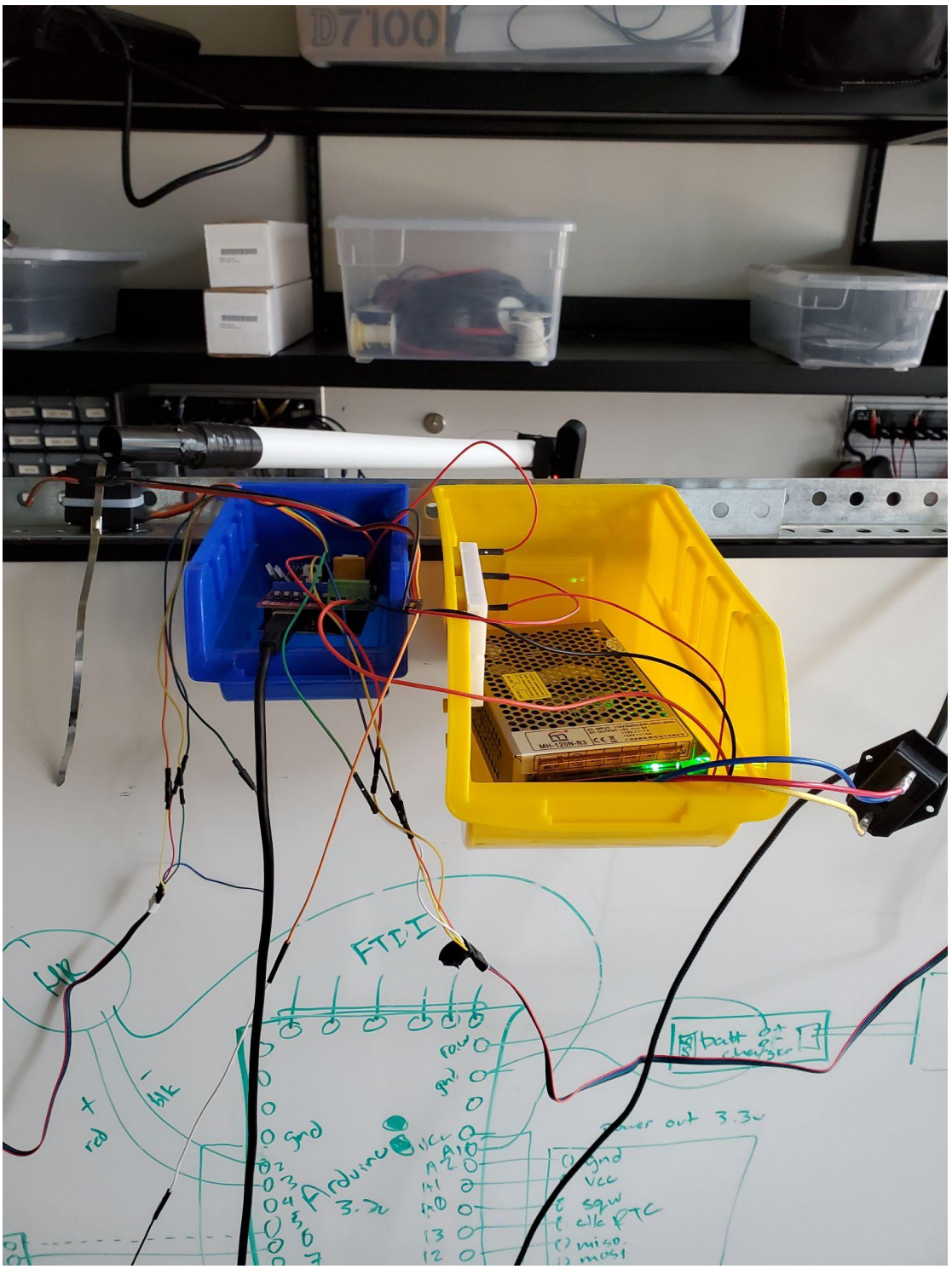


Figure 8: Complete setup of whiteboard robot, motor controller, camera and camera arm, and power supply. Note the Raspberry Pi is not mounted on top of the whiteboard in this setup.

Appendix D

This appendix shows the proper wiring of the subsystems. There are additional tips in the captions of each image.

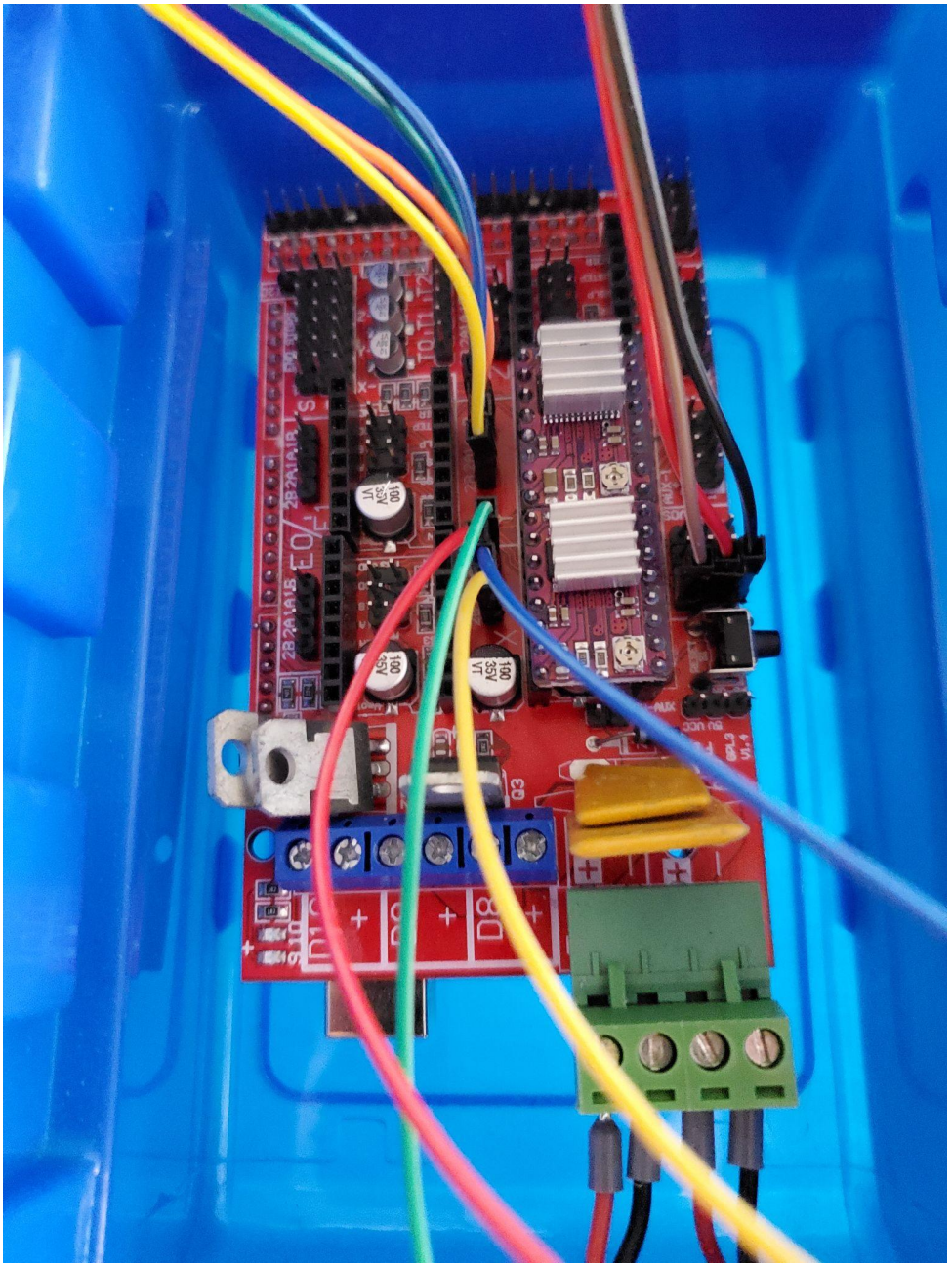


Figure 9: Proper wiring of the motor to motor controllers. Note the power wires on the bottom right connector will only have the 2 right-most wires connected to power. The 2 seen on the left were for testing purposes only.

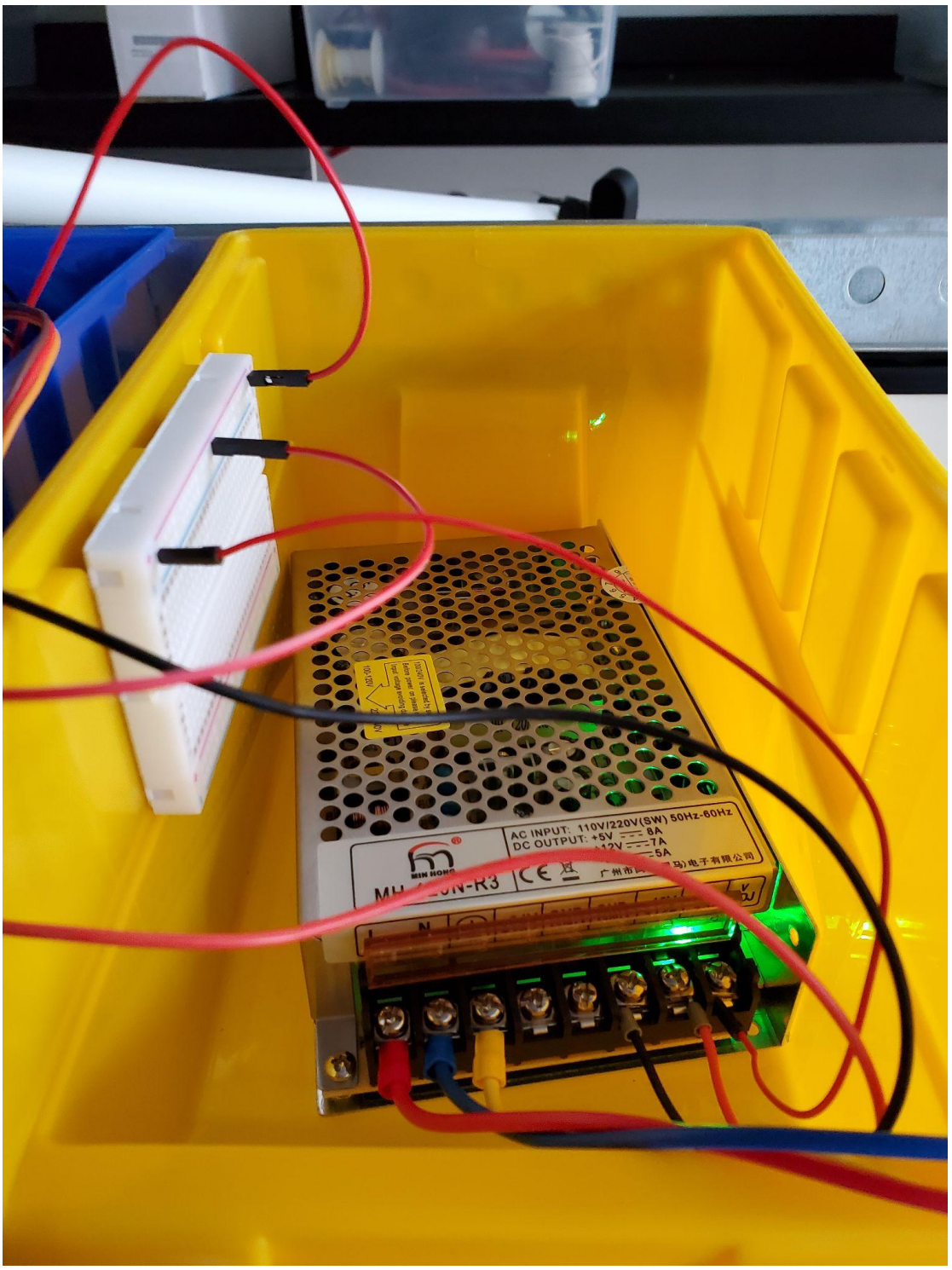


Figure 10: Power supply setup. The 12 V and GND terminals are connected to the green + and - terminals on the motor controller shown in the bottom right of figure 9. The 5 V terminal connected to the breadboard is used to provide the 5 V power to the servo motors.

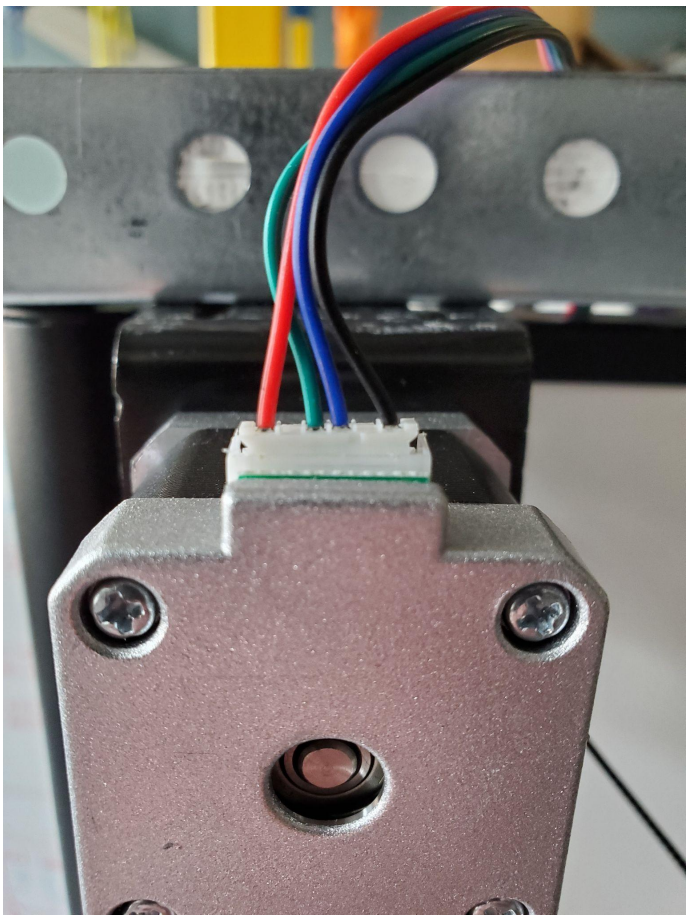


Figure 11: Stepper motor wire connection. The connector shown should be connected to the motor controller as follows (reference figure 9). When facing the whiteboard, the top left motor should be wired to the X axis inputs so that:
Red -> 1A
Green -> 1B
Blue -> 2B
Black -> 2A

The right motor should be wired to the Y axis inputs so that:
Red -> 2A
Green -> 2B
Blue -> 1B
Black -> 1A

Each connector pin is labeled on the motor controller.

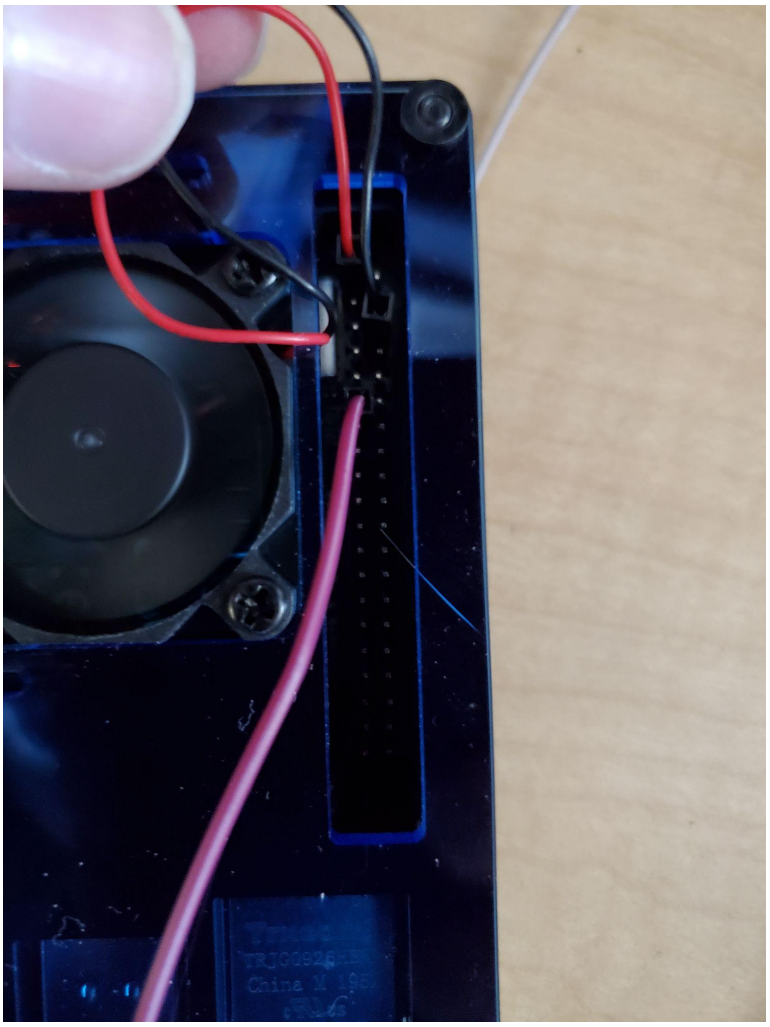


Figure 12: Proper connection pin for camera arm servo control wire.

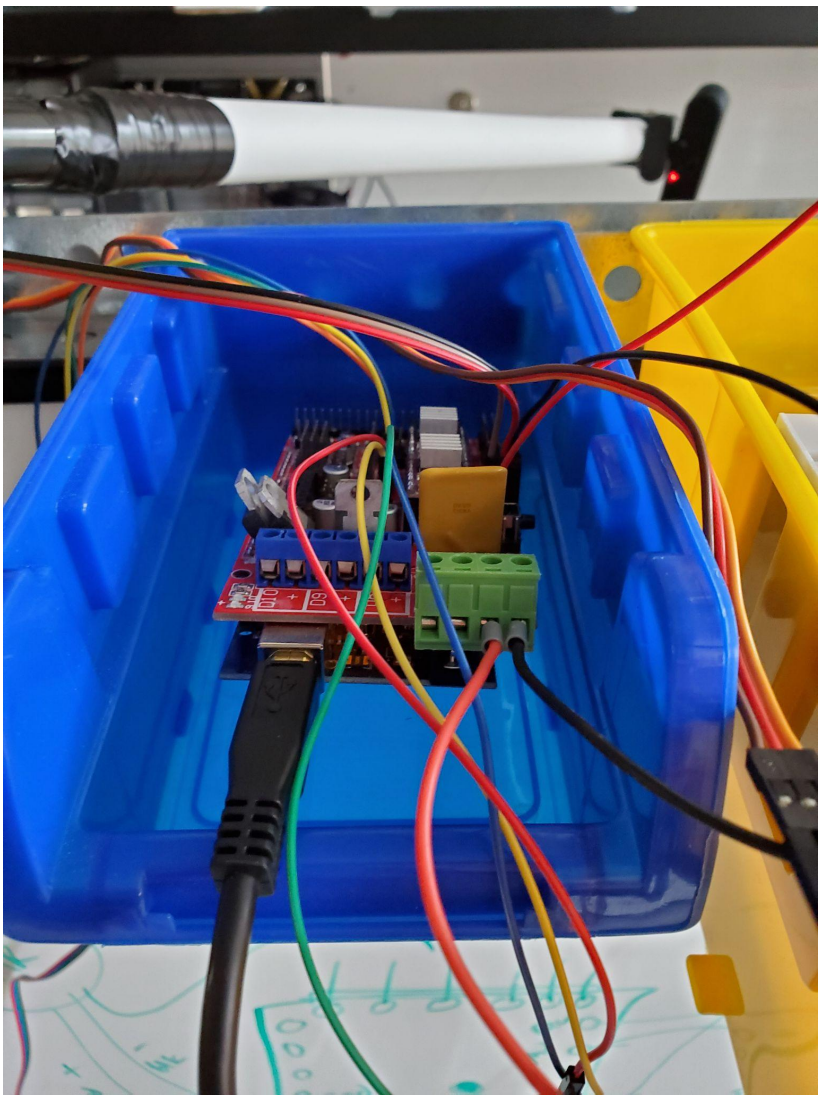


Figure 13: Final complete setup of motor controller. Note that power from the 5 V terminal is connected to the servo Vcc pin on the motor control board.