Team 4 Documentation
Whiteboard Bot

Table of Contents

Introduction

The Whiteboard Bot allows a person to write or draw on their touchscreen phones, and have it written or drawn on an actual whiteboard using an Android app and a robot. We came up with this idea while brainstorming cool and convenient things that we could do with robots, and out of all our ideas this one was the most practically applicable. For teachers, our project would make writing on a whiteboard more convenient, since they would not have to stand up to do so. The whiteboard robot also opens up possibilities for the physically handicapped and disabled, since it allows them to control things they would normally not be able to control bedridden or confined to a wheelchair. The core idea of this project-making menial tasks easier and more convenient for humans to do through robotics-can be applied to many problems in the world.

Materials

Components

- STM32F103C8T6 Microcontroller
- 2 x Nema 17 Stepper Motor
- 2 x A4988 Step Motor Driver
- 1 x Small Servo Motor
- Timing Belt & Pulley
- HC-05 Bluetooth Module
- 12v/6000mAh Battery
- 6.5'' x ⅜'' Wooden Dowel
- Meter Stick

Tools

- Computer
- Soldering iron & solder
- 3D Printer
- USB to Serial Adapter
- Assorted Screw / Nuts

Software

- Arduino IDE
- STM32 Flash Loader Demonstrator

Instructions

The first step to creating the Whiteboard Bot is to program the STM32 microcontroller. The microcontroller is what controls the Whiteboard bot to draw on a whiteboard. It listens for instructions from the Android app via bluetooth.

Burning the STM32duino Bootloader

With an Arduino, all we need to do to program it is to connect it to a computer using a USB cable and launch the Arduino IDE. However, this is not normally possible on a STM microcontroller since it lacks the capability of being programmed via USB by the default bootloader it comes with. Instead, everytime we want to program the microcontroller, we must we must move the jumper to the BOOT1 pin and then use a serial to USB cable to program the microcontroller. Furthermore, we must also use a program called STM32CubeMX and an IDE to program the microcontroller, which are not very beginner-friendly. Fortunately, there is a solution to this problem. We can install a special bootloader that brings the ability to use an Arduino IDE along with a regular USB cable to program the STM32 microcontroller, similar to programming any regular Arduino

To begin we must first download/install the following:

- STM32 Flash Loader Demonstrator - https://www.st.com/en/development-tools/flasher-stm32.html
- STM32 Bootloader (pick the one for your board) https://github.com/rogerclarkmelbourne/STM32duino-bootloader/tree/master/bootloader_only_binaries

Then move the jumper on the microcontroller to BOOT1 and connect the STM32 microcontroller to your computer using a serial to usb adapter.

STM32 Microcontroller		**Serial Adapter**
5V/3.3V	---->	5V/3.3V
GND	---->	GND
RX (A10)	---->	TX
TX (A9)	---->	RX

Now you can launch the STM32 Flash Loader program and follow the instructions on screen. When prompted, select the bootloader file you downloaded earlier. Once you have successfully burned the STM32duino bootloader, you may now disconnect the microcontroller from your computer, remove the serial adapter, and move the jumper back to BOOT0. From now on you may use a micro usb cable and the Arduino IDE to program the microcontroller.

Explaining the Whiteboard Bot Controller Code

Before we upload the code to the microcontroller, let's first understand how it works. If you have worked with an Arduino before or any basic microcontroller, you likely know that a microcontroller can do only one thing at a time. This limitation makes programming the microcontroller more difficult, especially when we begin doing complicated tasks. For the Whiteboard Bot, the microcontroller has to be able to do all of the following tasks at the same time:

- Move two stepper motors, possibly in different directions and amount of steps.
- Move a servo motor.
- Read data from the Bluetooth adapter as well as send data via Bluetooth.

As we can see, accomplishing all these tasks will be difficult without the ability to do multiple things at the same time. In laptops and desktop computers, we usually have multiple cores inside the processor, so that we are able to do multiple things at the same time. However, even then we do not have enough cores for everything we want to do at the same time. Instead, processors switch back and forth between tasks to make it seem like their doing things at the same time, when in reality they're not. Each separate task is called a thread. Thus, when we have numerous tasks that we want to do at the same time, it is called multithreading. The STM32 has only one core and no true support for multithreading. However we are able to do something similar known as protothreading.

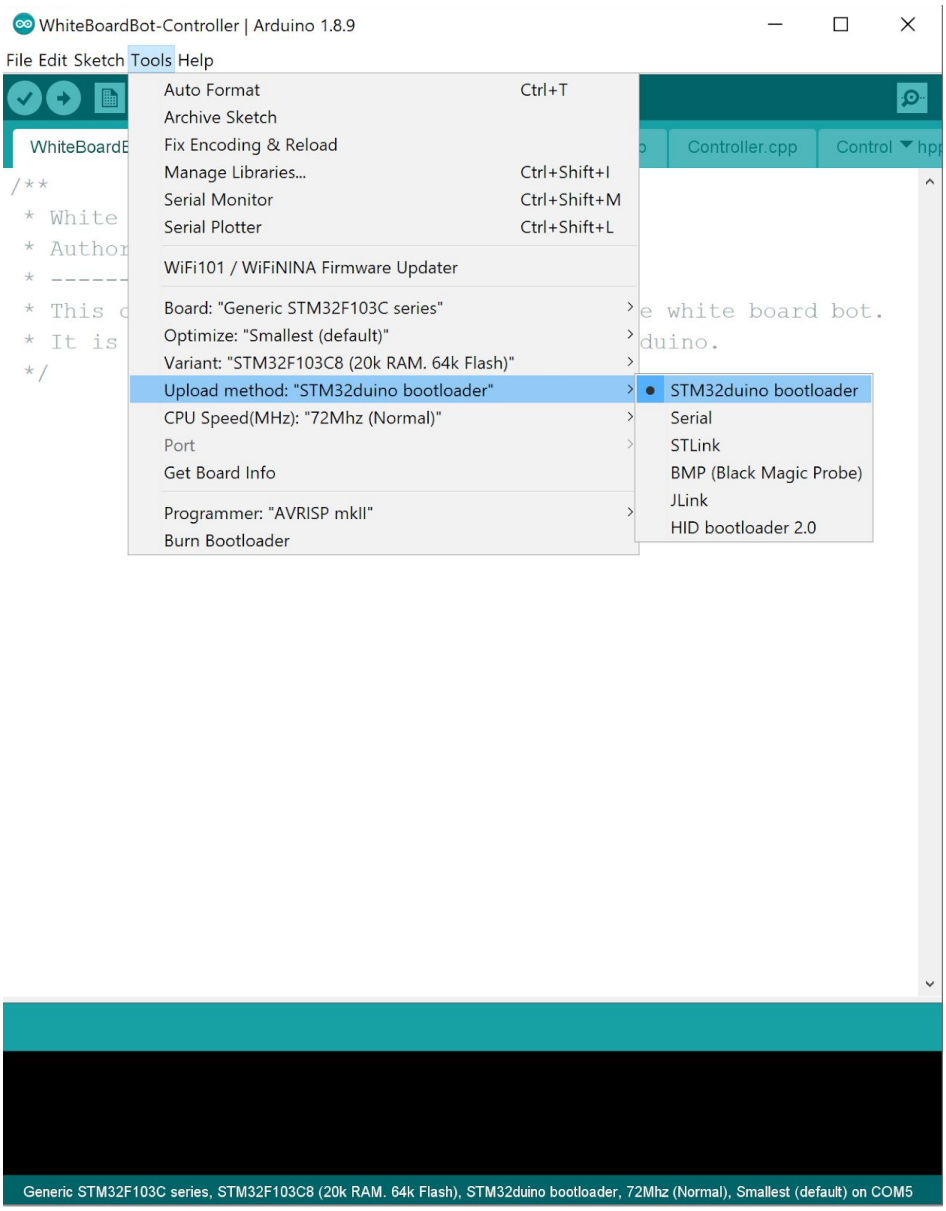
The code for Whiteboard Bot is divided up into "modules". Each module is a class that extends from an abstract class named Module. In the Arduino loop() method we simply call module.run() on all modules. For all modules, run() simply checks whether a boolean named pause is set to true or not and if it is not, it calls module.runTasks(). The idea is that runTasks() will contain all the things the module has to do at a certain point in time. Once the module is finished, we continue on processing the other modules' tasks. With this, we no longer have to do delay when we have nothing to do. We have included a class named Protothreading that allows us to pause a module for a certain amount of time and also invoke a callback once that time is up.

To demonstrate protothreading, we have also included an LED blinker module which constantly blinks the onboard LED every second at all times.

Uploading the Whiteboard Bot Code to the Microcontroller

The code for the Whiteboard Bot is published in the following repository: https://github.com/enlilodisho/WhiteBoardBot-Controller.

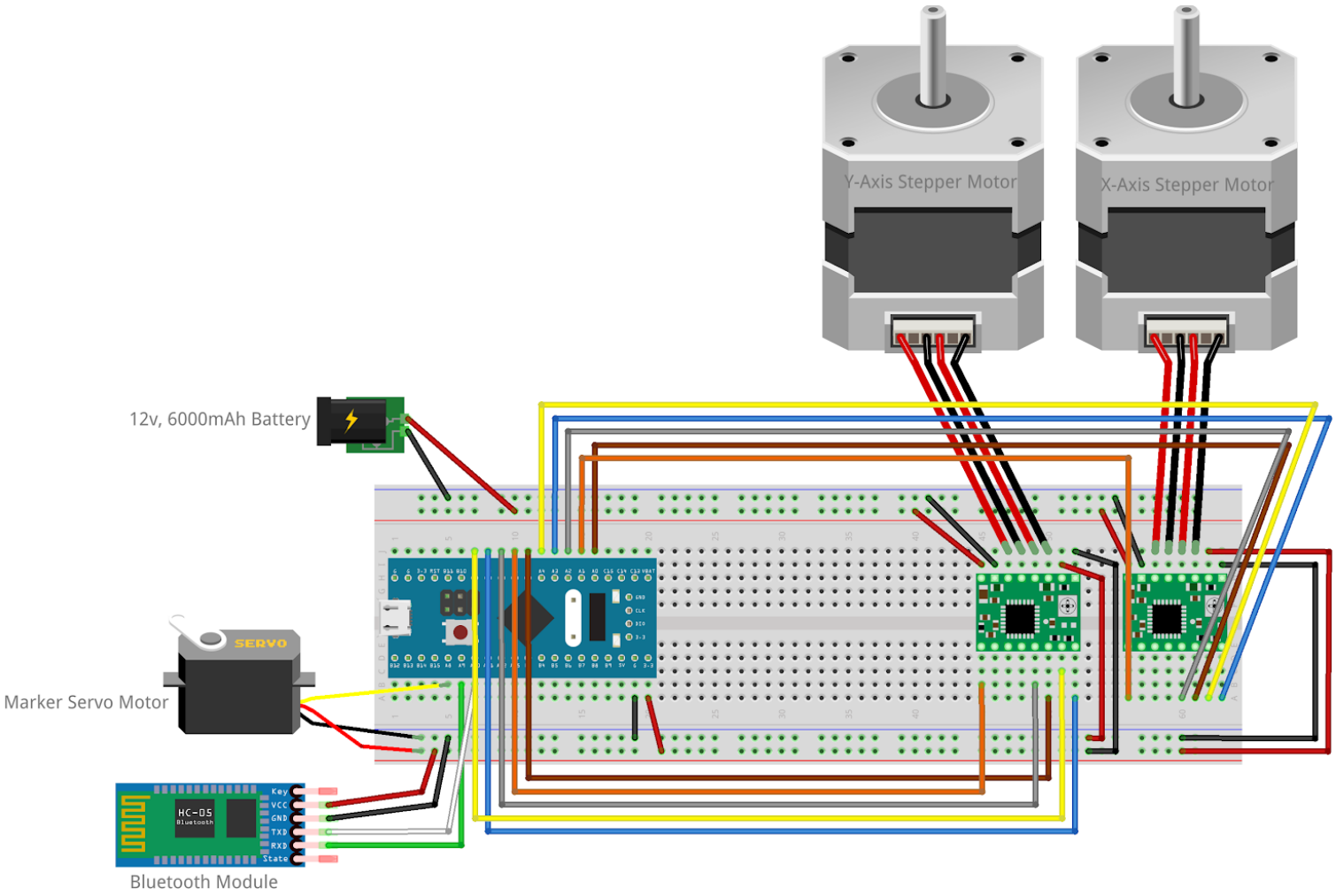
Download the code or clone the repository to your computer. Then open the WhiteBoardBot-Controller.ino file into the Arduino IDE. Navigate to Tools and make sure all the settings match the picture below.



Then upload the code to the board!

Wiring the Whiteboard Bot

Wire everything as shown in the schematic below:



The default pins for the marker servo motor is A8. The pins for the motor driver and STM32 microcontroller are listed below. If desired, any of the pins may be modified by changing the value in the config file.

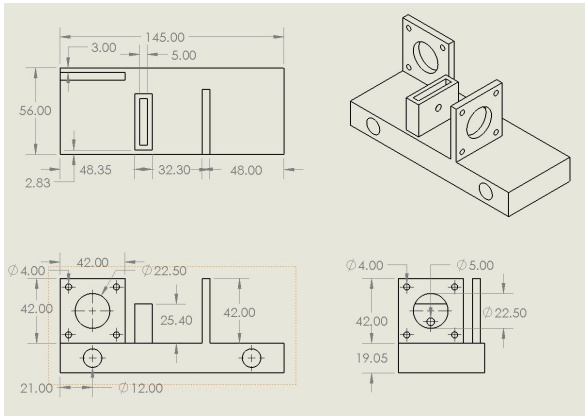
A4988 Motor Driver for X-Axis Motor		**STM32 Microcontroller**
Direction	---->	A3
Step	---->	A4
Sleep	---->	A0
Reset	---->	A2
Enable	---->	A1

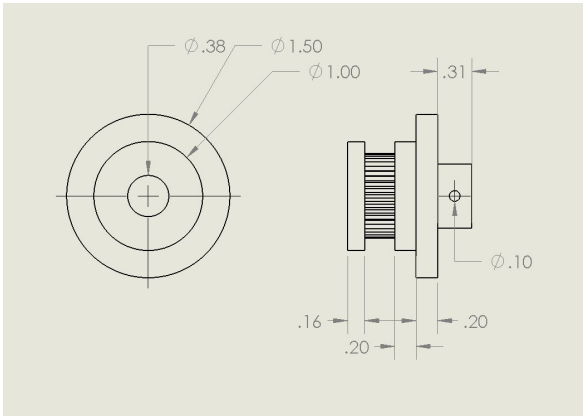
A4988 Motor Driver for Y-Axis Motor		**STM32 Microcontroller**
Direction	---->	B0
Step	---->	B1
Sleep	---->	A5
Reset	---->	A7
Enable	---->	A6

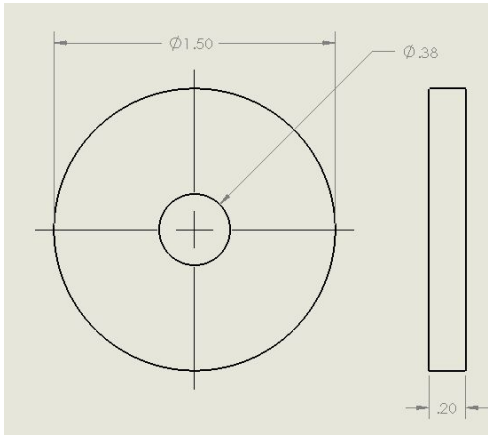
Building the Robot

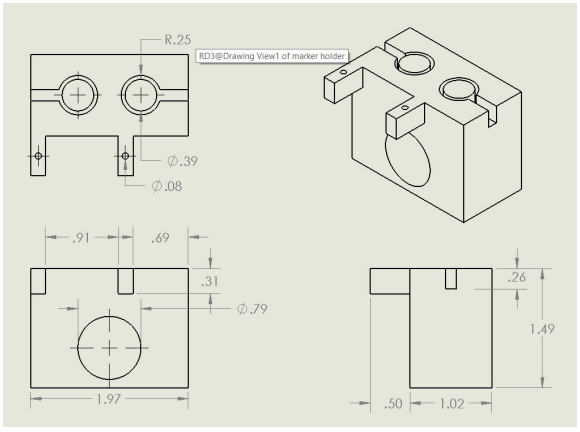
The robot mainly consists of 3D printed material. To start off, copy drawings below into a 3D modeling program such as solid works. Upload the 3D models onto a 3D printer and print.

**NOTE: These prints should be done in high quality to ensure that motors and meter stick fit into their respective spots.
**NOTE: you will need to print 3 passive wheels (bottom left)

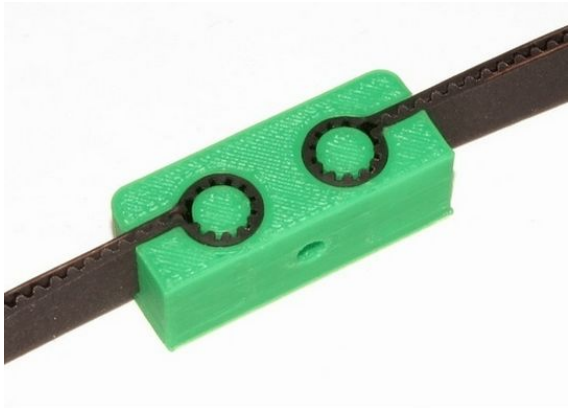






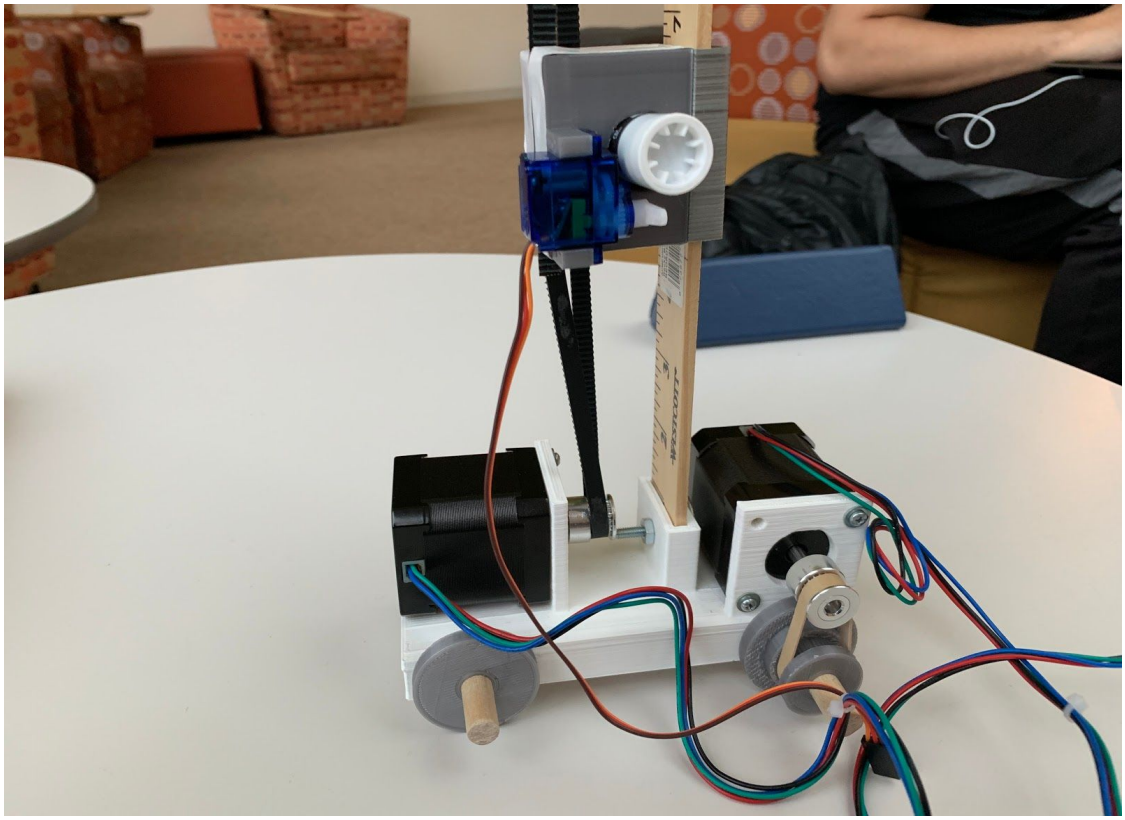


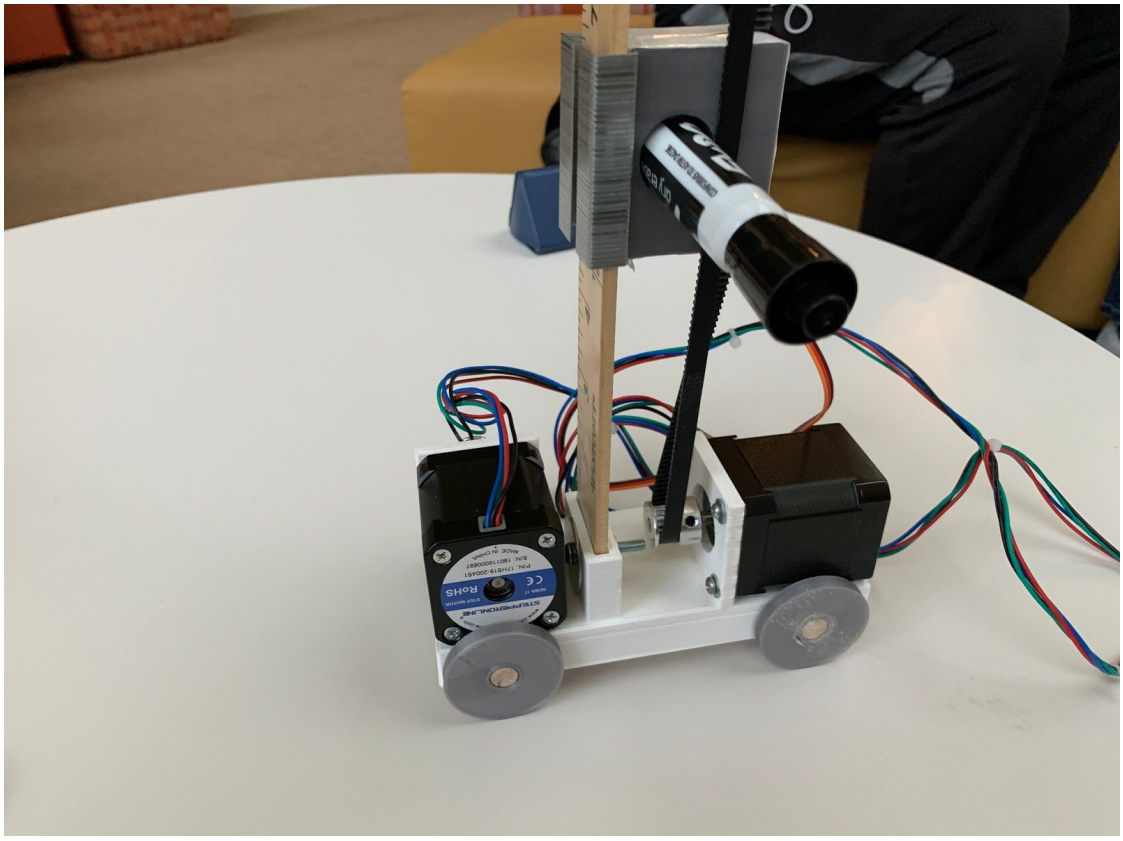
After printing, cut the wooden dowel into 3'' and 3 ½'' segments. These will be used as axles for the wheels. Attach the front of the stepper motors to the face of the motor plate and screw in to secure. Place the dowles through the 12mm hole in the base. The larger dowel should be on the side with the motor axle sticking out. Friction fit the wheels on the ends of the dowels so the largest one is right next to the motor axle. Place the meter stick into the middle slot and secure with a screw at the bottom. Drill a hole at the top of the meter stick and attach screw at the top hole.



Attach pulleys to both motors and the screw at the top of the meter stick. Measure out a little more than 2 meters of timing belt. Loop the timing belt around the two pulleys going along the meter stick and join the ends in the marker holder as shown on the right. Attach small servo to marker holder so the rotating head is below the .79 mm hole.

After these steps have been completed, the robot should look similar to the following pictures.





Android Application Overview

Class Descriptions:

PathDrawView.java:
Responsible for converting user input to a drawing on the phone screen.

Line2D.java:
Used in *PathDrawView.java* to draw lines on the phone screen.

MainActivity.java:
The main class for the android app. Connects the device to the robot via bluetooth and sends drawing to the robot in the form of two integers as coordinates on the whiteboard.

WBBController.java:
Controls bluetooth communication with the Whiteboard Bot. Used in MainActivity.java.

The application always starts in path mode. A path can be drawn directly on the screen with a finger or a stylus. When a Bluetooth connection is established and a satisfactory path has been drawn, tap `SEND' to send instructions to the bot. The app communicates with the bot by sending the x coordinate followed by the y coordinate over the bluetooth connection. The Whiteboard bot then moves to these coordinates and draws the path.

To delete a path simply begin drawing a new one. The old path will be discarded and the new one will be shown instead. The Whiteboard Bot is currently unable to erase a whiteboard automatically, so it is required to manually clear the whiteboard of any old drawings.

The user interface is composed of 3 buttons “Connect”, “Send”, “Reset”, and a white canvas to draw in.

The following are descriptions for what each button does:

CONNECT: When pressed, this button will connect to the robot. If pairing is required, it will be done automatically without any user interaction. When connected or when a failure occurs, the app will briefly flash a message on the screen. Once connected to the robot, the app will be able to send data and read commands from the robot.

SEND: The button sends a set of pairs of integers which represents coordinates to the robot when connected. Upon receiving this data, the robot should begin drawing the path sent.

RESET: This button clears all drawings from the drawing region on the app. This will not clear the actual whiteboard of any past drawings. Erasing is not yet implemented in the Whiteboard Bot, thus the user will have to manually clear the whiteboard if desired.

Conclusion

If we had more time, we would have added extra functionality to our project. The first thing we'd add would be the ability to erase from the whiteboard and switch markers to change colors. Another feature we wanted to add, but did not have the time to do so was to implement a method to auto-detect the whiteboard size. Currently, our robot has the size hard-coded in the code on the microcontroller. We were also considering adding whiteboard games such as tic-tac-toe and hangman to the robot. This functionality would require adding a camera to provide a way for the robot to see what is on the board. Our greatest challenge was programming the android app and getting it to communicate properly with the robot, since none of us had much experience with Android.