Smart Board Eraser Robot

Team Battleship

EE 209 AS

Robotics: Design, Manufacture, and Control

Fall 2016

Boyang Cai 304330123
Manni Chen
Zhuoqi Li 504135743
Weining Zou 004855607

Table of Content

1 Abstract

The goal of our project is to design a smart board erasing robot. We notice that people always spend lots of time erasing board. We hope to design a robot that helps people do this task so people can save the time and do something more meaningful. Also, we want to add features to make it humanization and intellectualization. Our smart board eraser has a pre-installed camera which can take a picture in front of the board. With software development, the system can extract the board area from the environment and project it into a touch screen. The user can select a rectangular area on the screen and request the system to erase it. There are also other buttons that assist the user to redo selection, save content and update board area information. We use RPI to control the camera taking pictures, make board detection and create user interface. For the hardware control, we use the Arduino MEGA 2560. The hardware is a large frame attached outside of the board. A board eraser attached to the frame can move both across the board and upward and downward to erase certain area.The wireless communication between RPI and Arduino uses RF24 communication. There are lots of problems occurred during the installation but after 5 weeks of working, the robot has the features we desired and accomplishes the task as we expected.

2 Background Introduction

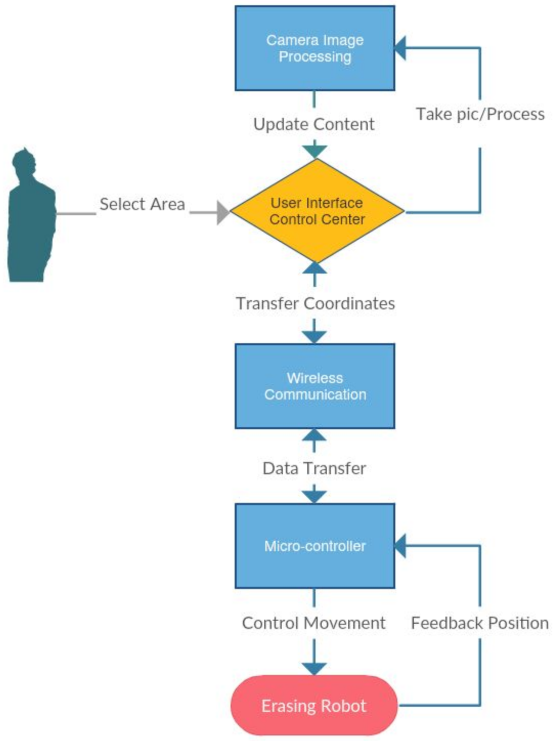
Board erasing is an unavoidable situation for most professors and students. However, erasing board can take lots of time during lectures and the dust caused during the action can affect breath. Meanwhile, after the board erased, instructors and students won't have any track to check whether their notes are the same as what written in the lecture. These kinds of problems have troubled us for many years and people are keeping finding solutions. Some mechanical students invented “auto board eraser”, a large mechanical structure to erase the entire board. However, the structure takes too much space and cannot clean the base by section. Some schools hire note takers to collect notes but the correctness cannot be guaranteed. Also, some classes may provide video recording but it costs lots of resources and has requirement on the classroom and the given lectures. Therefore, we decided to create a new design to solve these problems.

Our design, “Smart board eraser”, is aimed to intelligently and automatically erase the board based on request. It can also snapshot the board and save pictures for further reference. Our robot will go through the following process to achieve board erasing. First, a camera in front of the board will capture a picture. Based on the photo, the system can recognize the board area from the environment and project it on the touch screen. The user interface on the touch screen also contains some buttons. The user can select a small area from projection and click the erase button to ask the robot to erase that area on the real board. If the user select the wrong area, he can also click the reselect button to redo the selection. When the user wants to erase another area after the robot erasing previous area, he can click the update button to update the board image on the touch screen. There is also a save button which can save the current board content to the local directory for user to view later. To erase the board, the system sends the request information to the mechanical control section. The mechanical control first translates the coordinate from the screen size to the real board dimension and then starts erasing the board. A metal frame is fixed outside of the board. The X axis is attached to the board and its Y axis can move freely along the X axis. The XY axes correspond to the board's length and width. A eraser is attached on the Y axis and can be moved up and down. This is our Z axis which prevents the eraser removing reserved sections.

The auto erasing function can be further divided into three main features. First, with the camera assistant, the robot achieve auto recognition on the board area and can project it onto a touch screen. In this way, the user can easily select an area on the touch screen and determine if he wants to erase this part. Also the coordinate shown on the touch screen has been matched with the real board so the mechanical parts can erase the corresponding section based on the selected section on the touch screen. Second, the recognition and erasing area can cover the whole board so there's no dead angle that can not be detect. The motor can also move fast and forcefully to clean the board efficiently. Last, instead of having heavy mechanical parts like other "auto board eraser", our smart board eraser is light weighted. The wireless communication between the recognition part and the mechanical part avoids the long wire setup and guarantees the data transmission speed and stability. In the following sections, we will detailedly discuss the methods applied to each feature and procedures to integrate them into a whole part. We will also test and evaluate by sections and check whether our applications satisfy the requirement.

3 System Design

3.1 System Overview



The flowing chart above presents how we accomplish the board erasing robot. First, a pre-installed camera takes a picture in front of the board and sends it to our center control, Raspberry pi. Using edge detection and transformation, raspberry pi can recognize the board area, extract it from the environment and adjust any skew or distortion. After completing the recognition, raspberry pi projects the board area onto a connected touch screen and the user can draw a range within the area. The user interface on the touch screen also contains several buttons which allows the user to update the board(recapture the board image and projection), send selected area information, redo selection and save content information. Users can select a rectangular area on the displayed board area and ask the robot to erase it on the real board. When the raspberry pi received erasing signal, it will directly send the data to the mechanical control Arduino through wireless communication. Arduino will transform the original touch screen coordinate to the real board coordinate and guide the mechanical parts move to erase the board. After completing the task, the eraser will move to a feedback position for calibration.

3.2 Detailed Technical Approach

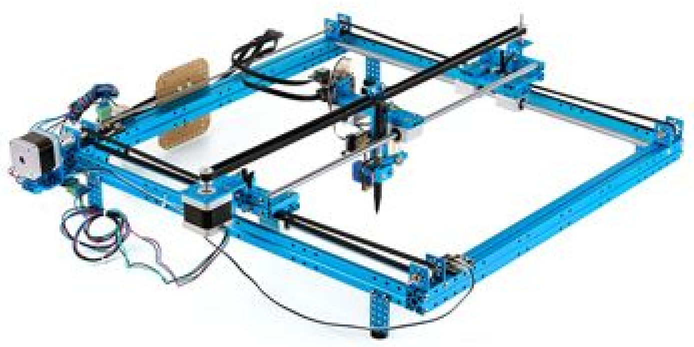
In this section, We will talk about detailed technical approaches from hardware assembling to controller algorithms.

Here is our project Github repository link:

https://github.com/lonelygoddady/EE209AS_Smart_blackboard_eraser.git

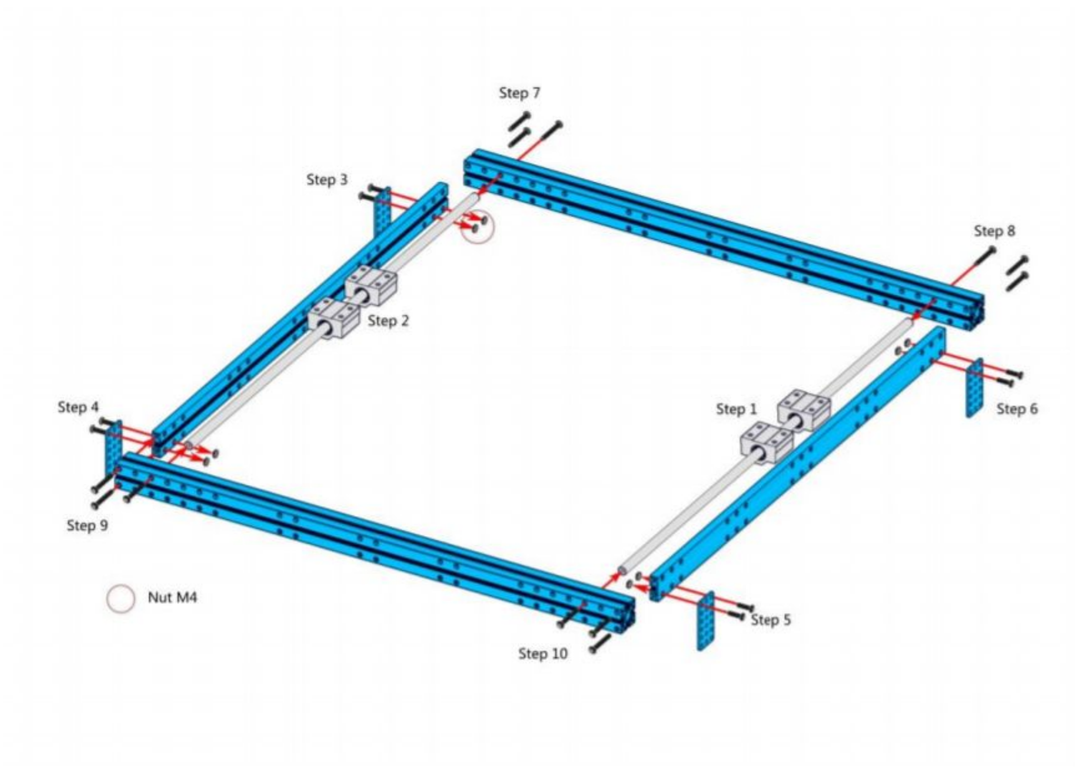
3.2.1 Hardware Assembly

At the very beginning of this project, we were thinking about DIY the XY plotter for our project. However, after researching for a little bit on the internet, we found DIY the robot may be too costly and time consuming. Therefore, we decided to purchase a well-established product and finally we reach to the XY plotter robot from the Makeblock.com [1]. This robot has three degrees of freedom: X, Y and Z in a limited rectangular space and is used for drawing. It X and Y axis are driven by two stepper motors with belts and pulley; the Z axis is driven by a small servo. Although the robot design does fit our the purpose of our project, its size is way too small (0.5 m X 0.5 m) and some modification is needed. We realized that the XY plotted robot can be assembled and all of its parts are available on the Makeblock.com. Therefore, to extend its size to meet our requirement, we simply need to order one more piece of the component and connect them with screws. Also we only bought the ‘without electronics’ version of this robot as we would like to customize our robot with our microcontrollers and electronics and we use two 2-A 42 BYG stepper motor to drive the motion.



Original Makeblock XY plotter robot

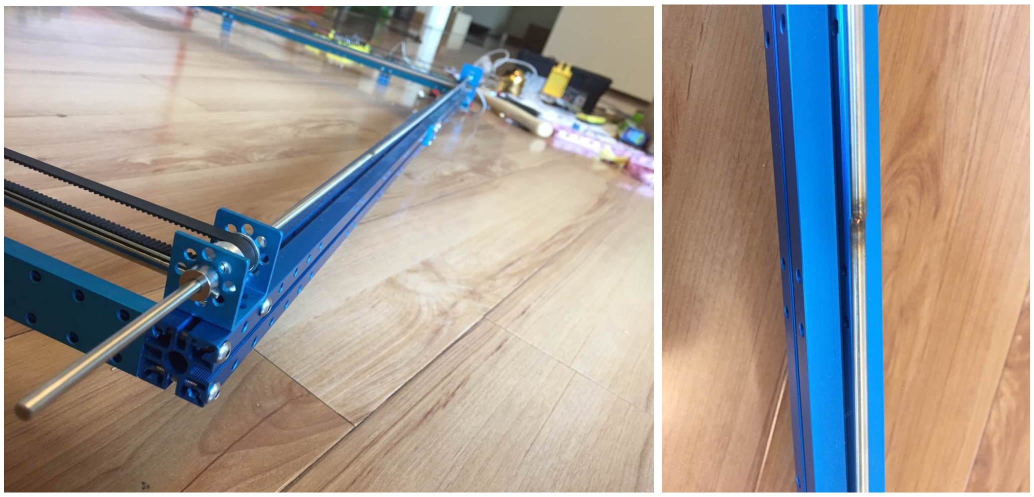
We decided that our robot should be as large as 1m x 1m in size, and this is four times larger than the original robot. Since the robot can be customized, we could simply extend this robot by connecting one more frame bar on each side of the frame and connect each linear motion shaft with additional shaft. Beyond that, we also needs to replace original belts with one that has twice the length and extent the driving shaft connected to the motor twice the size as well.



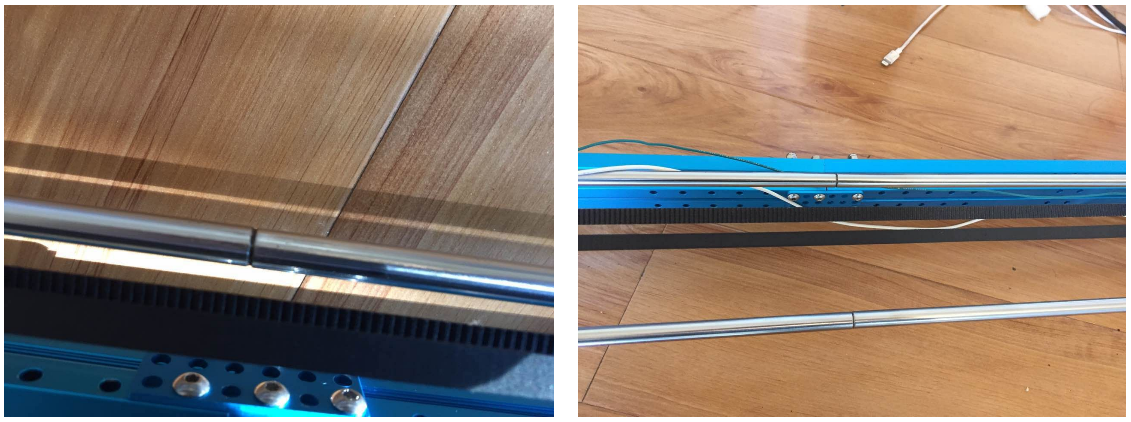
Frame assembling and customization

Components to be extended

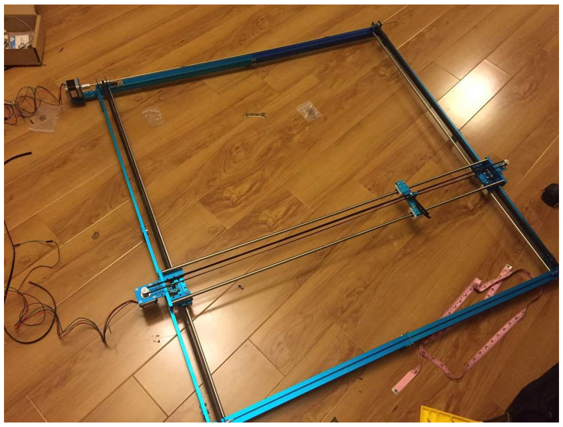
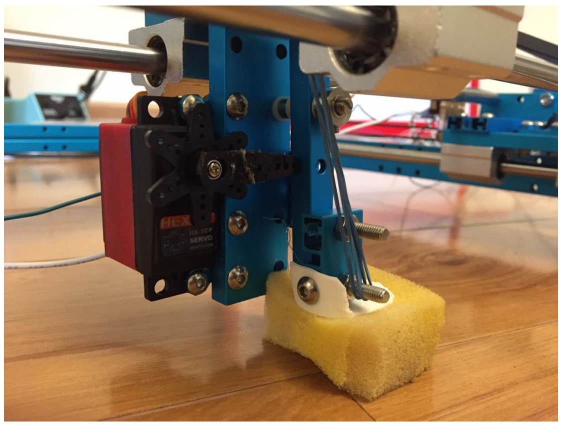
Components shown above can be easily extended by connecting two pieces together with M4 screws, and they compose the basic frame of our robot. However, there is one component that cannot be simply extended this way: the linear shaft connected with the motor as the driving axis. Because it is basically a solid metal stick, there is no way we could simply connect two pieces with screws. To solve this problem, we took three metal shafts to the UCLA welding shop and combined three pieces of shaft into one.


Motor driving shaft and welding point

After installing the slider and stepper motors onto the frame, mechanics of our robot is basically finished. However, Since this robot is not designed for this size and due to the imperfectness of our customization, it causes a few problems in the movement of the eraser end effector. For example, although we connect two pieces of slider shaft with headless M4 screw, there is still a slight gap at the connection point. This little gap sometimes will get the slider stoked when it is passing through. To alleviate the effect of this gap, we tried with multiple method including lubricating the shaft with engine oil and filling the gap with melted wax and it finally gets a little better than it was.

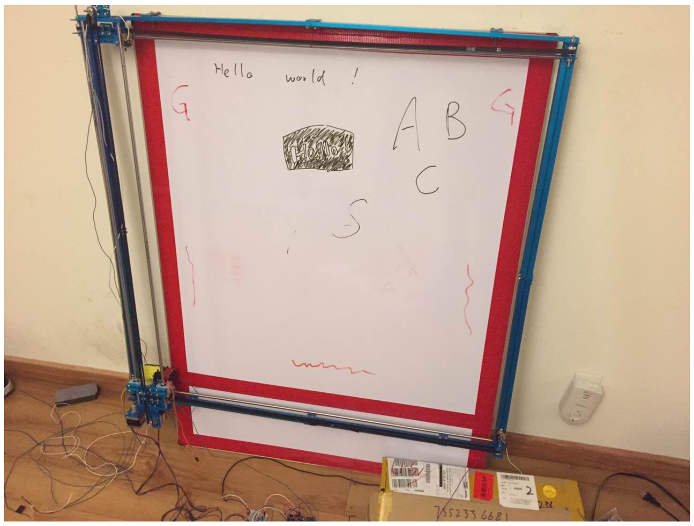

Connection gap of the slider shaft

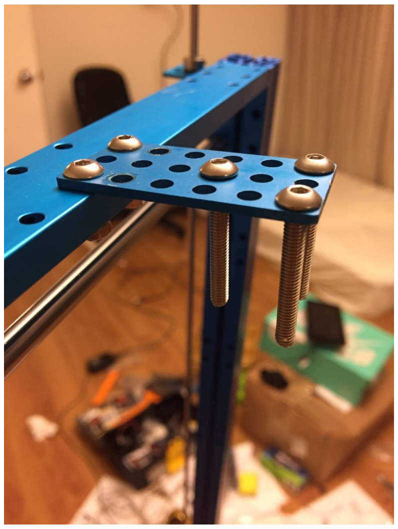
The picture below shows how the robot looks like after our customization. Next we installed the Z axis end effector onto the robot. Z axis movement is enabled by a servo motor. The eraser at first was a dish-washing sponge, later we replaced with a shower sponge later to reduce erasing friction.

XY robot after customization and End Effector

Since the whiteboard to be erased stands against the wall, we also need to mount our robot onto the whiteboard and perform erasing in order to mimic real-life scenario. In order to do this we installed two hooks made out of long M4 screws at both end of one frame beam, and then clip the edge of the board with both hooks. Since the robot is very heavy, this causes a little deformation of the frame beam but It is not a problem when we went through the tested. The orientation of the robot huang on the board is very critical. As displayed in the photo below, we choose this orientation so that the Y axis can move horizontally. If we choose the orientation that x axis moves vertically, the stepper motor and the belt may not generate enough torque and friction to fight against the gravity of the whole axis.





Robot installed on the whiteboard and holding hook

3.2.2 Board Recognition and Projection

Since we need to tell the exact erasing position to the Eraser, we need to map only the board area to the screen and match the position on the screen with the position on the actual board. We used a Logitech C910 as our board recognition camera. We uses openCV and edge detection to adjust and find the edges of the board. First, we make the whole photo to be the grayscale and increase its contract. This will make it easy to detect the edge of the pic. Then we apply gaussian blur to eliminate noise and find the edges using cv2.Canny.

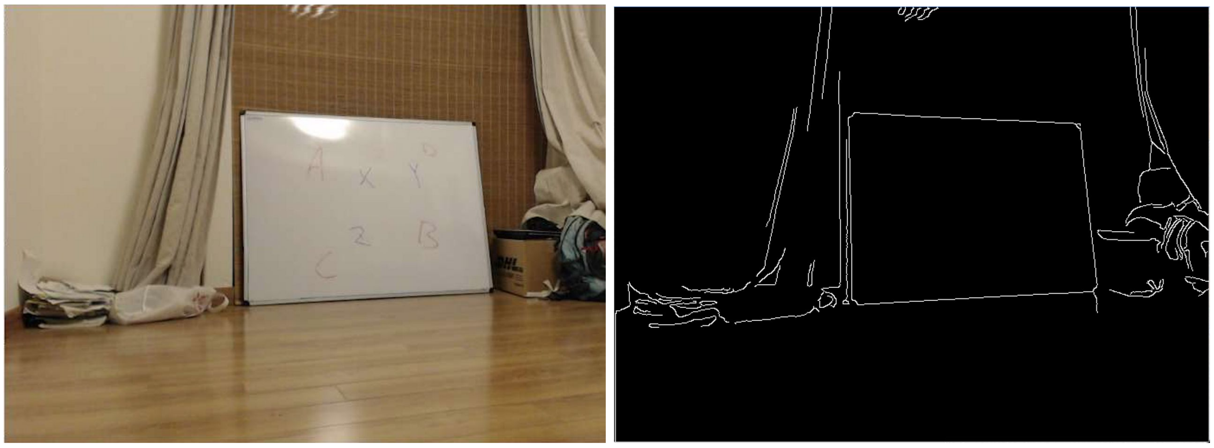


Figure: The original picture(left) and the edged picture(right)

After we find the edged picture, we start to find contours using cv2.findContours(). This function will give us all the Contours in the edged picture. We then filter the contours file by the following steps:

1. sort the contours from the biggest to the lowest since the board should be the biggest contour in the image
2. loop over the contour and see if these contours can result in a closed polygonal curve.
3. for each polygonal curve, find the one that has only four contours.

The above transformation is set up on the assumption that the board is the main part of the picture and it is rectangle shape and it is always true in our case.

Once we found the vertex of the board, we can then apply the transformation matrix to make the sheared picture back to the rectangle shape.

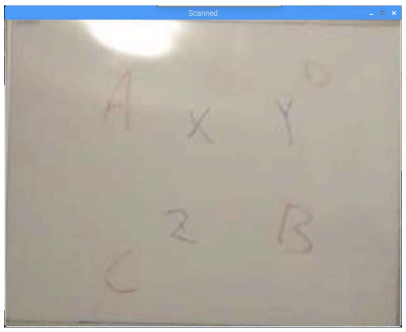


Figure: The final picture that will show on the screen

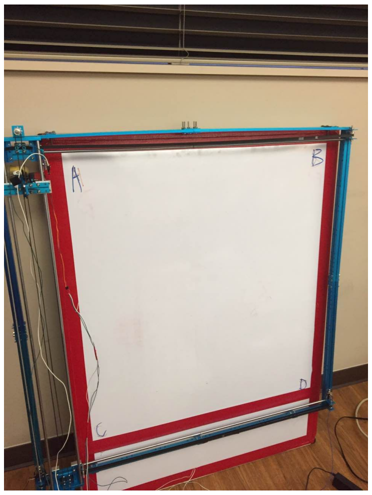
3.2.3 User interface

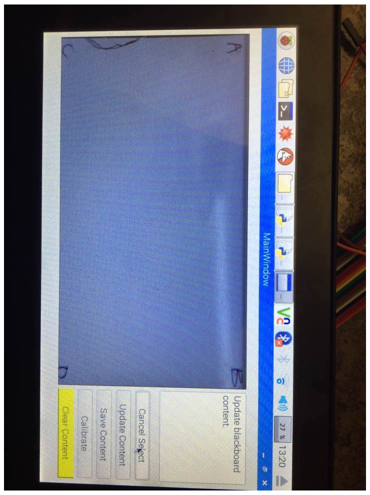
The user interface displayed on a 7-inch screen mounted on raspberry Pi is used for the user to choose erasing area and also for the system to display messages. It also serves as the ‘command center’ of our system which handles data transfer and image processing. Our UI is developed with the open source software QT designer. After using this software to create interface window and control widgets such as function buttons, we transfer the generated .ui file into python file and develop functions of included widgets using PYQT. The development of the user interface went through 8 versions. The following chart shows features of each version developed:

Version Number	Features
Version 1	Basic Board content display and buttons
Version 2	Finish widgets interaction logic and basic functions
Version 3	Board content split into four clickable areas
Version 4	Finalize user interface layout and enable rectangular area selection
Version 5	Facilitate area selection and coordinates mapping
Version 6	Integrate Communication and image processing function
Version 7	Add image calibration button and finalize design

The picture below is the finalized version (V8) of the user interface design. Mapped board content will be projected and fit into the rectangular area window. The smaller rectangular window on the right is the system message display. Since the resolution of the touch screen is only 800X480, the area to display the board content is actually about 600X320. This value also defines the resolution of our robot as we try to map the display area coordinate with the coordinate of actual erasing area on the whiteboard. However, since our robot has a square erasing area, this means it has more resolution in the x axis than the y axis. Fortunately because our project has large resolution tolerance, this is not really a problem. And this problem will vanish if we use a larger screen.

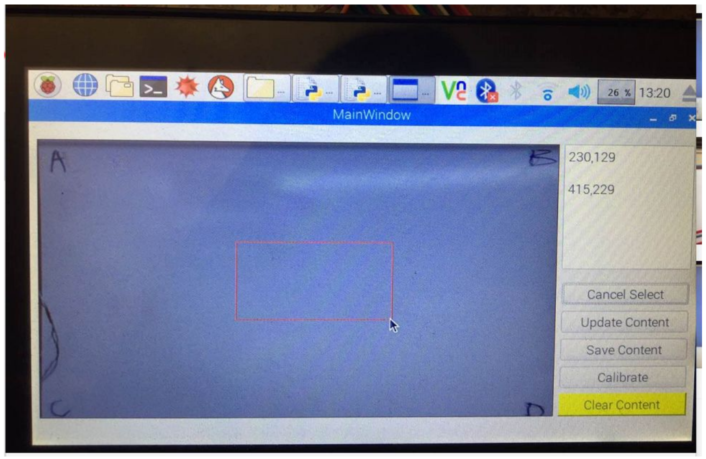
To perform erasing, first user needs to press the *'update content'* button which will let the web camera to take a photo and display it onto the board content window. After user makes sure nothing is blocking the camera and the photo took does cover the whole board, user can click the *'calibrate button*' so the image processing algorithm will find the board area, calibrate leaning angle offset and project board content onto the display window. Next, user could select the rectangular area to be erased by clicking two points on the screen. After user chooses these two points, a red rectangular frame will be drawn using these two points as diagonal points defining a rectangle. Seeing this area, user could either cancel by clicking *'cancel selected area'* or send signal to microcontroller to perform the erasing by clicking '*clear content'*. Next time when the user wants to erase new content on the board, he/she can simply press the '*update content*' button and repeat the process again. Note that the user does not need to click the '*calibrate*' button as long as the camera position stays unchanged.





Actual board content and projected board content on touch screen

The selected area information will be sent to the robot controller by passing transformed coordinates of two points selected by the user. The origin of coordinate system on the screen and on the board locates at the top left corner. Originally we were thinking about let user to draw their customized area on the screen. However, soon we realize it is actually really hard to draw on the coarse surface of our cheap touch screen. Also, since our eraser has a rectangle shape with only three degree of freedom (X,Y,Z) and is unable to rotate, it could only erase area in rectangular grids. It is basically impossible for it to erase an irregular area without large errors, not to mention the low resolution of the projection window. Therefore, due to limited time and budget, at current stage of our prototype we try to keep it simple and easy for user to use. George has already come up with an algorithm to break irregular areas into rectangular grids to perform the erasing.



230,129

415,229

Area highlight after selection

3.2.4 Wireless Communication

Wires between Arduino controller and Raspberry Pi can affect appearance and long wire can be easily damaged or loose in accident. Therefore, we decided to set up wireless communications between Arduino and RPI. Initially, we wanted to use Bluetooth to transmit data but after two weeks of trial, we found the original Bluetooth module in RPI is broken. Therefore, we switched to use RF24 module. Although we didn't find any RF24 library directly applied to RPI, using RF24 to communicate between Arduinos is quite successful and stable. Finally, we determined to add an additional Arduino for data transmission. RPI will have USB serial connection to the transmit Arduino which use RF24 to wirelessly communicate with Controller Arduino.

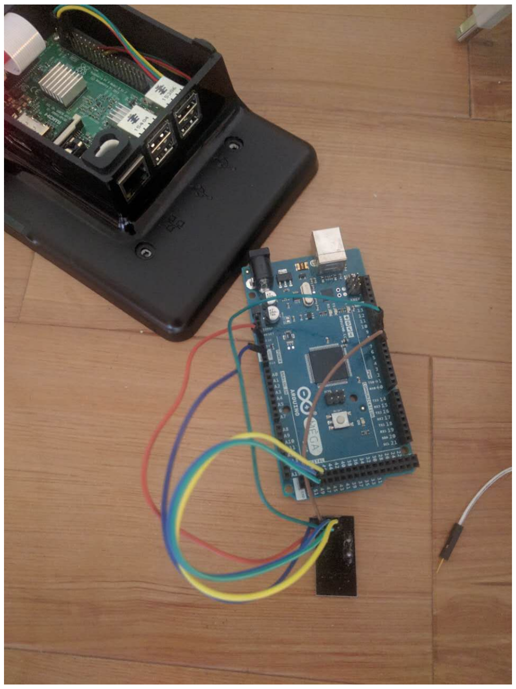
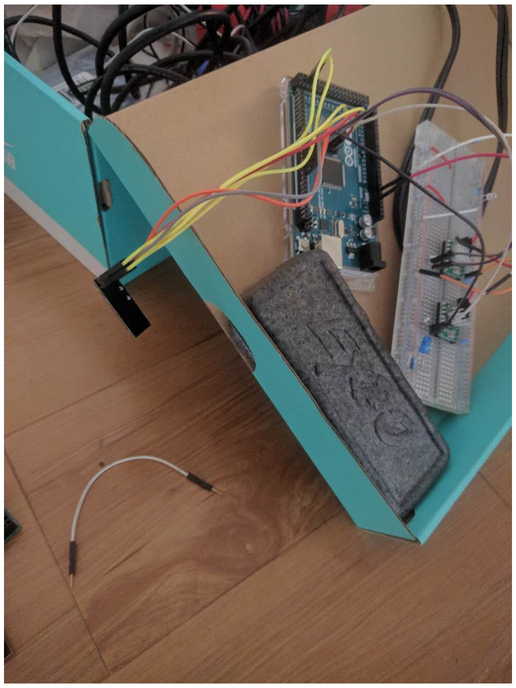
Brief Summary of Bluetooth Trial

Since RPI has on-board Bluetooth and we have a Bluetooth module for Arduino in hand, we first planned to use Bluetooth to transmit data between RPI and Arduino. There are plenty of tutorials for Bluetooth communication between these boards. We thought the communication installation could be fast. However, there were lots of challenges and problems during setup. First, this kind of communication has been many years. Most of the tutorials are written in 2015 or earlier. However, the Bluetooth package for RPI just updated few months ago and lots of commands have been changed. We took a while to find out and study the updated tutorial. Then we found that even the Bluetooth on RPI could successfully pair with the Arduino Bluetooth module, the connection was always broken. After a few tests, we found the Arduino Bluetooth module works properly since it could connect to our cellphone and receive signals from it. However, when we tried a lot of times to connect the Bluetooth on RPI with any Bluetooth devices, none of them succeed. All auto-setup or manual installation instructions to connect Bluetooth on RPI with other devices didn't work. Therefore, we believed that the on-board Bluetooth on RPI was broken and gave up the Bluetooth trial.

Application of RF24 with Serial Communication

We found that many projects using optimized high speed NRF24L01+ drivers to do wireless communication between Arduinos. For each Arduino using RF24 communication, it has an address containing two parts: the writing pipe and the reading pipe. If Arduino A wants to communicate with Arduino B. Their addresses should be matched that the writing pipe address for Arduino A is the same as the reading pipe address for Arduino B, and vice versa. Although it shows that the driver can also be used on RPIs, we couldn't install RF24 library on RPIs. In order to save time, we decided to add one Arduino for communication purpose, which is called transmit Arduino later.

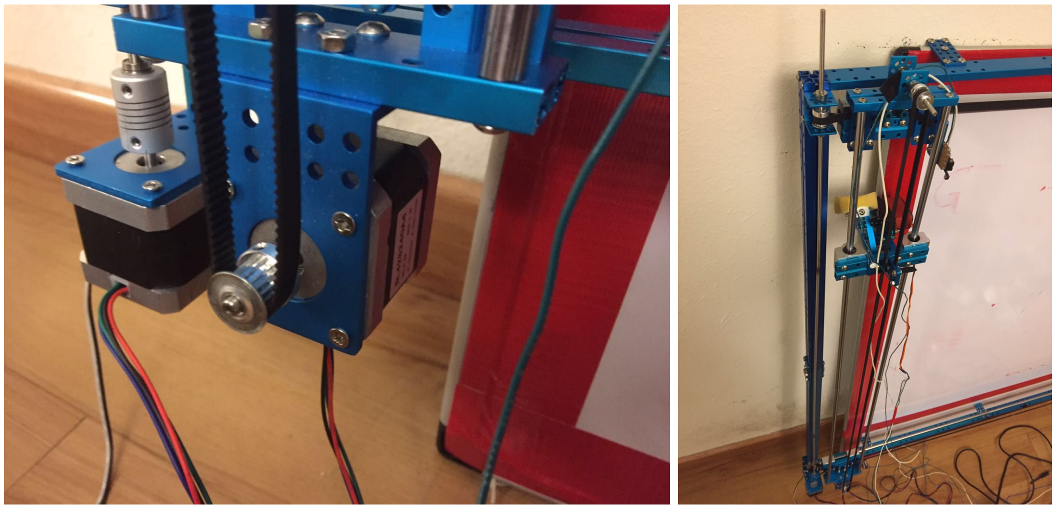
First we connected RPI with transmit Arduino with a USB cable. When RPI wants to transfer data, it will open the serial gate, send the signal and close the gate. RPI has to open and close the serial at each time to change the serial availability state in transmit Arduino. After the transmit Arduino receives the data from RPI, it starts the RF24 communication with the controller Arduino. The controller Arduino always listen to the transmit Arduino till it receives the data. Then it will stop listening and write to transmit Arduino to check if it receives the correct information. For the controller Arduino, once it receives the data from RPI, it will stop listening and send data to the controller Arduino. Then it will listen to the feedback from controller Arduino to check if data has been successfully transmitted.

3.2.5 Eraser Robot Control

XY axis movement

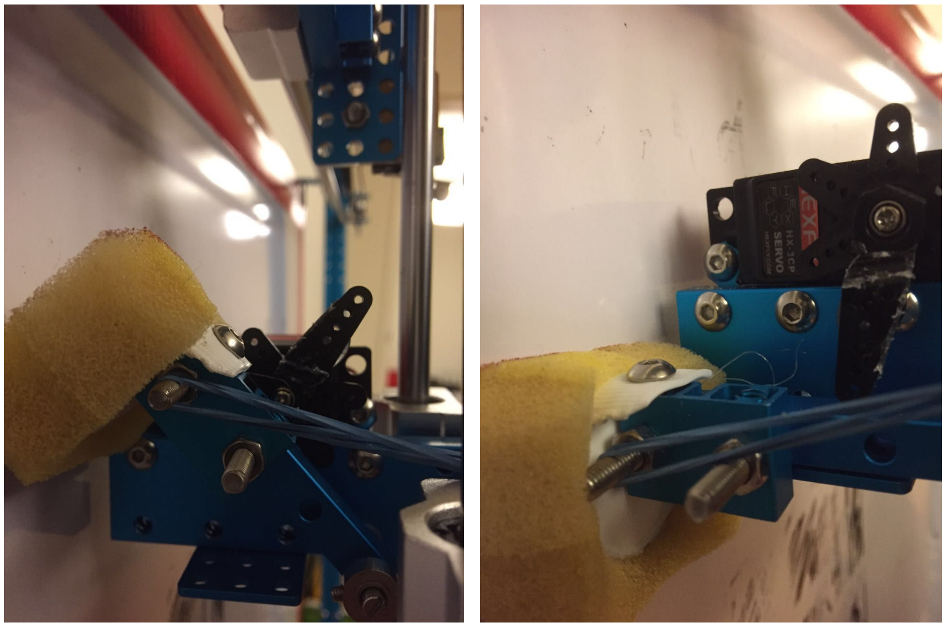
As mentioned previously, the robot has three degree of freedom (X,Y,Z) limited in the frame, motion in the X and Y axis is dominated by two 42-BYG stepper motors. This motor we use is a 2-phase motor that can provide 40N.cm Holding torque and 52G.cm2 rotor torque with 1.7 A/phase [5]. These motors come with the original XY plotter robot package from the Makeblock.com and should be strong enough to drive our slider and maintain slider's position when it is moving in the y axis. The rotational torque of our motor is transferred into linear motion driving force with the use of pulley and belts. In order to drive the motion in the X axis, driving torque is promised at both the bottom and the top of the robot through the motor shaft. The second motor is mounted on the Y axis and the end effector is driven by the belt in a similar fashion. When the end effector is moving in the X axis, the whole Y axis, including the slider, slider motion shaft and the motor will also move along. This causes the X axis motor to have a much greater load and leads to some trouble in the motion. For example, it gets jammed occasionally when Y axis is moving pass the shaft connection gap. In comparison, movement in the Y axis is much more smoother, and gravity of the end effector turns out to not a big problem.


Stepper motor torque transmission structure

As mentioned in the user interface section, limited by the touch screen resolution, the actual resolution of the projected content is around 600X320. At the very beginning, we were a little worried about whether the motor has enough resolution, however, to our surprise, the motor proves to be way too good - it takes both motor 5,000 steps to reach at the end of the axis. So technically our motor will fit as long as the screen has a resolution lower than 5000X5000.

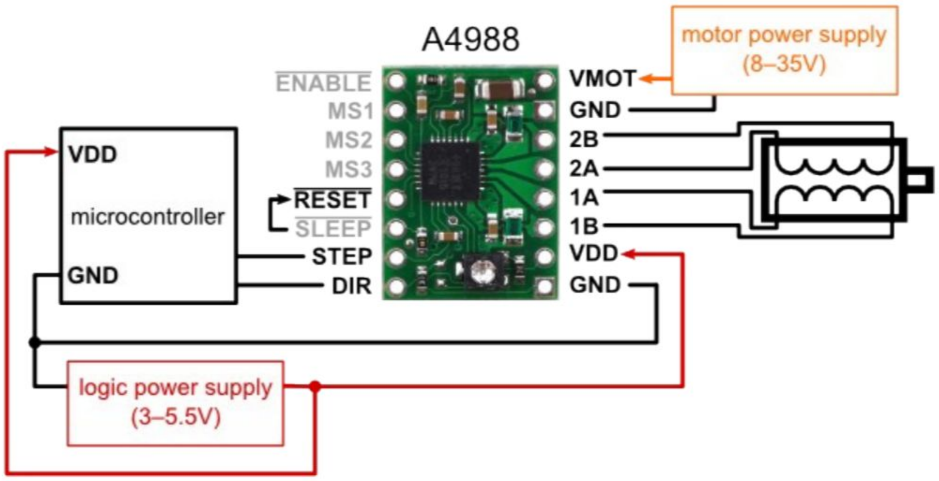
End effector (Z axis) movement

The end effector movement is enabled by a servo motor. Whenever erasing is needed, the servo leaf will rotate to 180 degrees and push the sponge to touch the whiteboard. When erasing is performed, the servo leaf will maintain at 180 degree locking the movable structure and prevent the sponge from getting lifted by friction while moving. After erasing is done, the servo will rotate back to 0 degree, dragging the movable structure up with rubber band and therefore release the sponge. At the very beginning, we were using a SG90 9g servo as the actuator, but soon we realize it is too weak to push the eraser against the board. Therefore, we replace it with a 3KG servo which can provide twice as much torque.


End Effector push and release

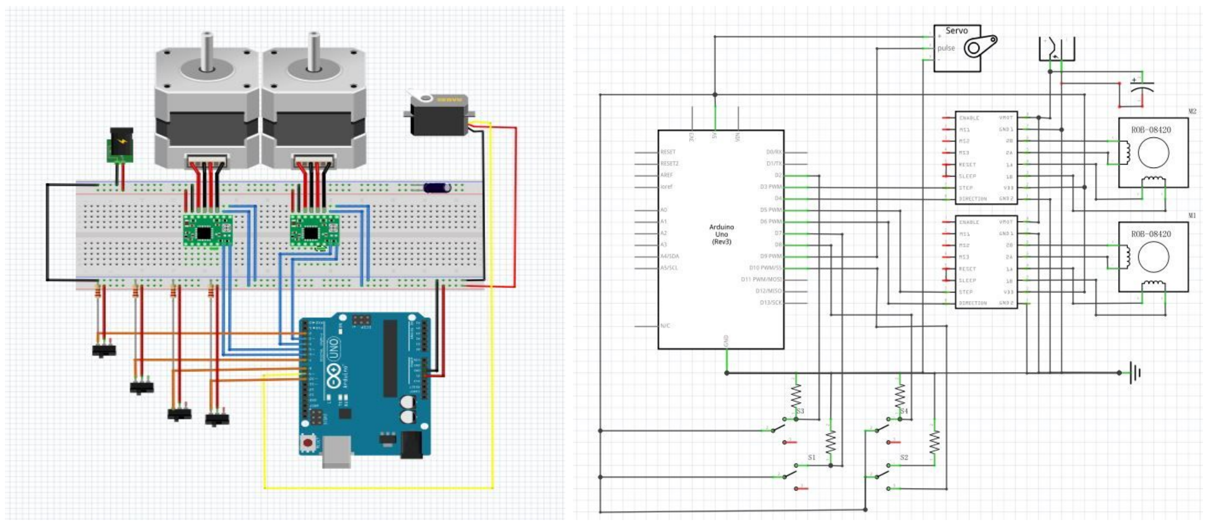
Actuator Control

To drive two stepper motors, we bought the A4988 stepper motor driver, which is a cheap standard driver used in 3-D printers and is able to output 2.0 A current at max. The following diagram shows how to connect the driver and motor:



Step and DIR pins are logic output signal from the microcontroller to control the motor movement, we use the default full step mode as it already provides far more than enough resolution. Despite that this connection looks standard and simple, we spent most of our time in this project to work on this part. More details will be revealed in the following challenge and failure section. For the power supply to drive the stepper motor, we find a 12V FPGA power supply which able to provide 2.0 A. We cut its head and takes out the ground and VDD wire, hooking them into our circuitry. Furthermore, the servo is driven by 5-V output from the arduino and control by a PWM pin from the arduino.

As for the controller, we choose a arduino UNO board. We also install four slider switches on four end of the frame. Whenever the slider moves to the end and trigger the switch, the robot will stop moving and knows it reaches to the end. The following is our circuitry diagram and schematic:



Robot Control Circuitry

Erase Algorithm

When two coordinates defining the erasing rectangle get transferred, first they will be transform from the touch screen coordinate to the actual coordinate on the whiteboard. After coordinates transformation is done, four values of two coordinates will be fed into the erasing function. Our erasing algorithm is very straight-forward. First the end effector will go to the first diagonal point, drop down the eraser and then perform the erasing by moving to the Y value of the second diagonal point and then come back. Next, decrement or increment the X value of the end effector by the size of eraser and repeat last step. After the whole area is erased, the reset function will be called and the end effector will returns to the origin with eraser retrieves. Two end switches get triggered and the robot knows its current position is at origin and update its own position. The controller will also send a feedback signal back to RPI to notify the erasing task has finished.

Note that while the robot is moving, although it will keep updating its position, however, it has no real-world feedback to correct its localization prediction. Therefore, whenever the movement is not as smooth as expected, error between the actual position of the end effector and the predicted position will occur. Sometimes when the movement gets jammed because of friction, this error will become very large and the robot becomes extremely inaccurate. For now, due to budget and time limit, the only position feedback mechanism we have on the system is the end switch. Whenever end switch toggles, the robot will correct its position accordingly. Since position error, either large or small, will occur during erasing because of friction, unflattened board surface and belt slipping, reset function needs to be called after each execution of erasing to correct position error. In the future, we could add potentiometer to feedback end effector position during movement.

3.3 Challenges and Failures

"You will run into the avenue of problems."

- *Professor Mehta*

Exactly as professor mehta predicted when we first proposed this project idea, we encountered tons of challenges and run straight into the "Blvd of problems". However, much to our surprise, most of troubles confronted come from those places where we considered to be least likely to yield any issues.

Bluetooth module malfunction

As mentioned in the previous function, in the very beginning, we tried to realize the wireless communication via RPI embedded bluetooth module. However, after dozens of trials and more than one week of work, we finally found the internal bluetooth module is not working. While it can be detected and paired, it could not connect to any devices including cellphones and transfer data. We have tried firmware re-installation, upgrading driver and even reflashing RPI OS but none of them works. We also consulted another team which also utilizes RPI bluetooth module. We learned that they also encountered this problem and end up solving it by replacing with a brand new RPI. However, we are unable to adopt this solution due to limited budget.

Signal pin conflicts between communication and hardware

Initially, we used Arduino UNO as the controller Arduino. When our group members complete parts separately, both communication and hardware control succeed. However, after we merge the codes and wire connection together, we found that the controller Arduino receives lots of false data through RF24. We did a bunch of testing and found out that the digital pins (pin 11, pin 12, pin 13) used in hardware installation are also communication signal pins (SCK, MISO, MOSI). Therefore, when RF24 wants to get signals from SCK, MISO and MOSI, it will read the information from both the pins connected directly to the RF24 driver and the digital pins that are actually used for hardware. Also, when we attached the servo to the PWM pins, it will also influence the RF24 communication. To solve the problem, we changed to use Arduino MEGA2560 as the controller Arduino. MEGA2560 has enough pins that we avoid using pins have multiple usages. We also stopped attaching servo from the setup. Because when the controller Arduino sends feedback signal, the listening process is disabled, we attach the servo at this time and complete the hardware movement. In this way, the signal from servo won't affect the RF24 communication. We also detach the servo every time it completes the erasing task and then start RF24 listening. After applying these methods, there are no more conflictions and the communication process works well with the hardware movement.

Imperfect Robot Mechanics

Another challenge we faced that seriously affect the accuracy and smoothness of our system arises from the imperfect robot structure mechanics. For example, it requires a lot of torque while performing erasing. Although our motor proves to be strong enough to do it, the torque delivery structure, the use of pulley and belt, does not meet this goal. Sometimes the slider would be jammed by the connection gap or unflattened board surface because there is not enough friction between the touch surface of the belt and pulleys, causing the motor to rotate just by itself and belt slipping. Limited by the material we have, there is basically nothing we could do. One way to fix this problem is to replace the use of pulley and belt with chains.

Beyond that, as mentioned in previous section, we also have problems such as connection gap, unflatten board surface and shape deformation when hang.

Circuitry bugs and hardware failure

We encounter most of our troubles in the motor controlling and had spent almost 60% of our project time in hardware debugging and tuning. Since Weinning could not figure it out himself, George had to finish the UI development early and help him on this part. As described the and motor controller section, stepper motors are controlled by two drivers with a rather straight-forward standard circuit. We planned to spend only one week on this part but it turns out we totally underestimate its difficulty. Due to initial messy circuitry connection, we burnt two arduino microcontroller board (Mega 2560 and Uno) and one Macbook Air by accidentally short the 12-V power source with the arduino 5-V output. This causes everything in the circuit connected with 5-V output gets pull up to 12 V, including the USB port connect to arduino board from laptop motherboard.

The biggest challenge we faced in this project lays in the controlling of the stepper motor. Although the connection is rather simple, the stepper motor often does not operate as we expected. Furthermore, out of 20 stepper motor driver we bought, we damaged and burnt 16 of them. Since we have checked the arduino board and motor are intact, the problem must lay in the driver. After searching on the internet for awhile, we finally learned that although it claims our driver could provide a Max of 2.0 A current, without proper cooling, it could at most output 0.7 A [2]. Since the motor could draw as much as 2.0 A current, with not current limit set on the driver, the chip would be fried in as short as 15 seconds. This is the reason why our drivers were so easily damaged.

However, after we set the current limit on the driver, another problem arises and took us a long time to figure out. Sometimes it is unable to drive the motor at all because the current provided is too low considering our load! This put us in a very embarrassing situation: we cannot either let the current to be too high or limit the current too much. We have to find a perfect range that the current is just adequate for the motor to work and also not too much to overheat the driver. This windows turns out to be very narrow: around 0.7 ~ 0.9 A and it varies from chip to chip. In order to find this range, we damaged a few drivers through the process.

Probably because we have run through so many incorrect trials and failed attempts, finally our stepper motor for X-axis stops to work on Monday of the final's week. Luckily we have already record a demo video before that.

4 Team

4.1 Duty assignment

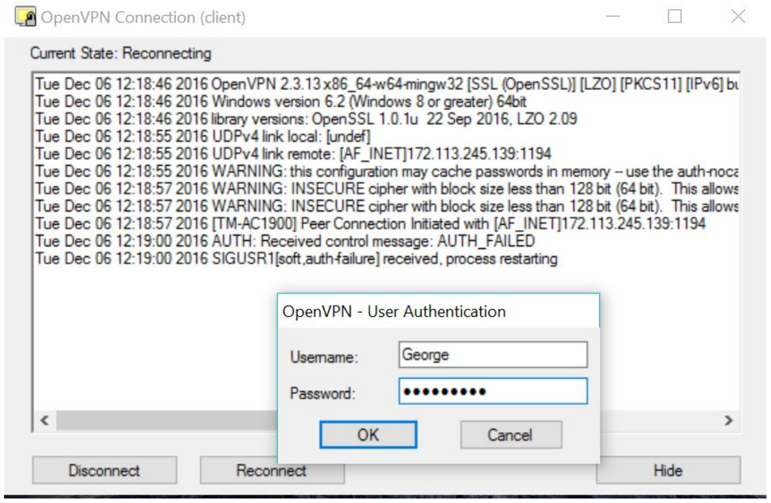
As described in previous sections, we divided our project into five parts: hardware assembly, Image processing, user interface, wireless communication and robot control. For hardware assembly, all team members involved in this section. For the following four parts, each of team member is in charge of one part but also cooperated with other members in process to complete the whole project.

Boyang Cai is in charge of the board area recognition and correction. Weining Zou works on mechanical controllers to erase board. Zhuoqi(George) Li creates the user interface on touch screen and helps Weining Zou with eraser robot control. Manni Chen sets up the wireless communication between raspberry pi and Arduino. We integrated each member's work at the very end.

4.2 Work coordination

Since our robot is too huge to carry around, we set up the robot in Manni and Boyong's apartment. After settling meeting time on social media chat group, Zhuoqi (George) and Weining would come to work on the project each week.

Three out of five parts of our system require members to work on the raspberry pi. However, we only have one pi available. In order to make it possible for each member to do their work simultaneously, we built multiple VNC servers on Raspberry Pi so that each team member can SSH into the machine and work on independent sessions. Boyang also built a OpenVPN server on his router so we can remotely connect to the RPI set up in his apartment from anywhere as OpenVPN client. These measures considerably enhance our work efficiency.



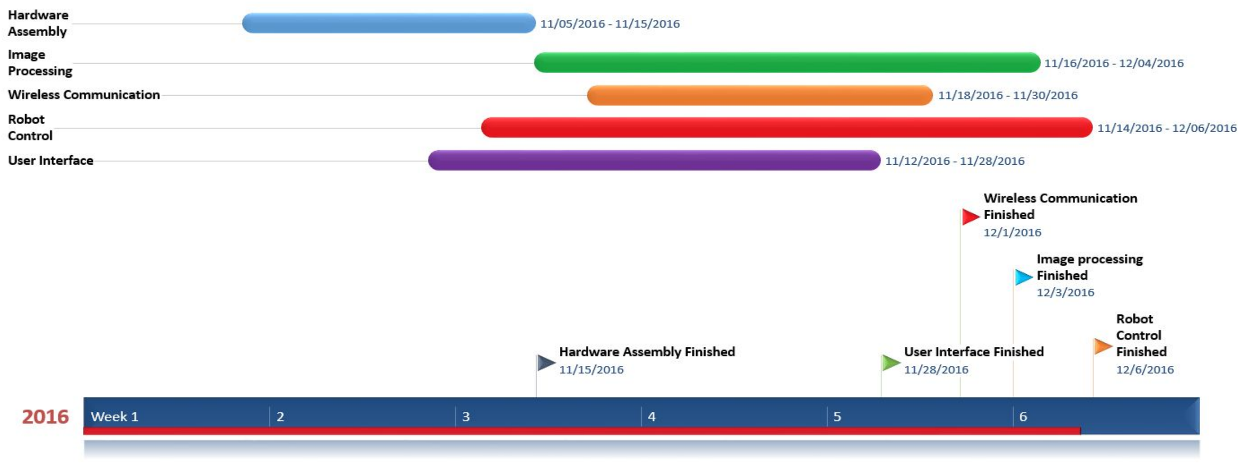
```
Tue Dec 06 12:18:46 2016 OpenVPN 2.3.13 x86_64-w64-mingw32 [SSL (OpenSSL)] [LZO] [PKCS11] [IPv6] bu
Tue Dec 06 12:18:46 2016 Windows version 6.2 (Windows 8 or greater) 64bit
Tue Dec 06 12:18:46 2016 library versions: OpenSSL 1.0.1u  22 Sep 2016, LZO 2.09
Tue Dec 06 12:18:55 2016 UDPv4 link local: [undef]
Tue Dec 06 12:18:55 2016 UDPv4 link remote: [AF_INET]172.113.245.139:1194
Tue Dec 06 12:18:55 2016 WARNING: this configuration may cache passwords in memory -- use the auth-noca
Tue Dec 06 12:18:57 2016 WARNING: INSECURE cipher with block size less than 128 bit (64 bit).  This allows
Tue Dec 06 12:18:57 2016 WARNING: INSECURE cipher with block size less than 128 bit (64 bit).  This allows
Tue Dec 06 12:18:57 2016 [TM-AC1900] Peer Connection Initiated with [AF_INET]172.113.245.139:1194
Tue Dec 06 12:19:00 2016 AUTH: Received control message: AUTH_FAILED
Tue Dec 06 12:19:00 2016 SIGUSR1[soft,auth-failure] received, process restarting
```

OpenVPN - User Authentication
Username: George
Password: ●●●●●●●●●

OpenVPN client connection login

5 Implementation Schedule

The following gantt chart reveals our system implementation schedule and parts completion date. As shown from the chart, the robot control part is not finished until the final week. As mentioned in the previous sections, this happens because we encountered countless hardware failure and mechanical problems. Despite that the robot control part falls beyond schedule, other parts are finished on time.



6 Test and Evaluation Results

During the controller building process, some experiments are done in order to make sure the controller works properly. The first and the most important test we did is the movement test. Two functions called move_x and move_y were used to test the movement of the eraser. The two functions are designed to let eraser always remember its current locations. We set the left-bottom point as the origin. We gave x coordinate to move_x function as well as y coordinate to move_y function to see if it can go to the right position. We kept doing this to make sure it can reach the correct position from any point. After the eraser can move to every coordinate according to x, y inputs, we started to test switches making the eraser stop at the edge. The test failed and the eraser still kept moving when it hit the edge. We found switches gave floating input instead of 1 for trigger and 0 for release. The connection of our circuitry was wrong. Resistors were needed between switches and ground. We fixed this problem by adding 4 resistors between the 4 switches and ground. After deal with X,Y axis, we tested the servo on Z axis to see if it is strong enough to press the eraser on the board. Because of the uneven surface of our white board, the servo sometimes could not press the eraser on the board. Thus, we changed to a more powerful servo to make sure the eraser will be stay on the board when erasing. When everything was ready, we combined all the part together and started to test the erasing and reset function based on the diagonal vertices we gave. The eraser went to the rectangular area we selected and performed erasing. It came back to the origin when the erasing was done. The result means our controller is successful.

As the controller is just a prototype, there are some defects. The controller circuitry is now on a breadboard as the circuitry plot shows. These wires can be annoying when the eraser is moving. Thus, a PCB based on the schematic showed before need to be used replacing the breadboard. The erasing speed is not satisfied even the step-motors reach their maximum speed. So, there may be a more efficient algorithm that we can use to replace the present one. The function now is also limited. The erasing function can only erase one rectangular area at one time. For the future work, we want to make our erasing function can erase multiple rectangular areas at one time. Furthermore, we will try to perform irregular erasing, which means erase an area whose boundaries are formed by curves. To do this, we may need to add an extra servo at Z axis to let the eraser can rotate when necessary.

When the eraser moves, the slider of X, Y axis may jam occasionally as mentioned before. To improve this, we will try to find some ways to improve the physical condition of the sliders, such as make them smoother. However, some kind of small jam may be unavoidable. So, we need to find a way to let eraser know jams occurs and fixes errors due to those jams like adding some other sensors as detectors.

We can easily detect the whole board area and project it onto the user interface. Sometimes when the motor didn’t go back to the original place or a person stand in front of the board, the edge detection may fail because it can’t find the complete 4 edges of the board. But for most of the times, it works perfectly.

We record a video to show our project results and post it on YouTube.

https://youtu.be/OeyT_Mt9cQI

7 Conclusion

Building a robot is interesting but also difficult. Edge detection is a completely new area for us to explore but we are excited to see it works. We learnt lots of methods to set up communication between RPI and Arduinos. We also spent lots of time to learn how the mechanical parts work, trouble shooting and figure out solutions. Although the final result is not ideal as we original expect, it still gives us an inspiration on robot development.

Besides the knowledge in building up robots, we also learnt a lot on teamwork skills, research and testing strategies. Although we divided the projects into individual parts, but we always scheduled to work together. When we found a new application, we always consulted other team members to check how it will affect their work. When one met difficult problems, others were always willing to help and give advice to solve it. Meanwhile, we also became familiar with lots of blogs and websites that guide us to complete the problems and provide solutions for problems we met. Besides, we concluded some methods for troubleshooting while testing. For example, when we combined communication with the hardware control and found irregular activity from the communication, we disabled the whole hardware control and gradually added commands and tested repeatedly to see what affects the communication.

The smart board eraser is not only a simple robot that can solve problems in board erasing. With a few changes, we may also apply it to other areas. If we change the eraser to a painting brush and enlarge the whole frame dimension, we may use the robot to scan a picture and paint it on the wall. In a library, we may ask the robot to scan the barcode on the book and grab and sends the book to the coordinate shelf. With other end effectors, we believe the robot can accomplish lots of job and make our life easier.

8 References

1. http://www.makeblock.com/xy-plotter-robot-kit/
2. https://arduinodabbler.wordpress.com/2012/07/13/heatsinking-a-pololu-a4988-a4983-stepper-driver/
3. http://www.robotshop.com/media/files/pdf2/xy-plotter-v2.0-user-guide-mdraw-version-1.0.pdf
4. http://forum.arduino.cc/index.php?topic=408764.0

5. http://www.makeblock.com/42byg-stepper-motor/
6. https://tmrh20.github.io/RF24/classRF24.html#a048a20c73c7d9b2e02dcbae6fb9c4ba8
7. http://www.instructables.com/id/Raspberry-Pi-Arduino-Serial-Communication/
8. http://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/