2013 ESE 519 Project

PhanTom

Gesture-based Control System

Chenyang Zhu

Hanfei Sun

Chao Liu

CONTENT

1. Introduction

Nowadays wearable devices are more accessible to human beings, with wearable devices, human beings can do things more easily and effortlessly. This project, PhanTom, is to design a gesture-based control system which lets users to use the movements and motions of hands to effortlessly control digital devices. We built a master system which integrates EMG signal as well as IMU signal to get gesture and motion control to seamlessly interpret what the hands and fingers doing. Then we transmit the control signals over wireless channels, to the slave systems which can execute correspond actions according to the instructions. We implement multiple slave systems, including vehicle, cannon, quadrotor and some software on PC. And in the final demo, we set two slave systems, attack and defend system respectively. The attack system consisted of a vehicle and a canon, the vehicle could move around with the canon. And the canon could also aim around and shoot. The defend system consisted of a canon which is much bigger and can aim around and shoot. Two users wear our devices can fight with each other. With some simple and intuitive gestures and hand motions, people can play the game easily.

2. System Architecture

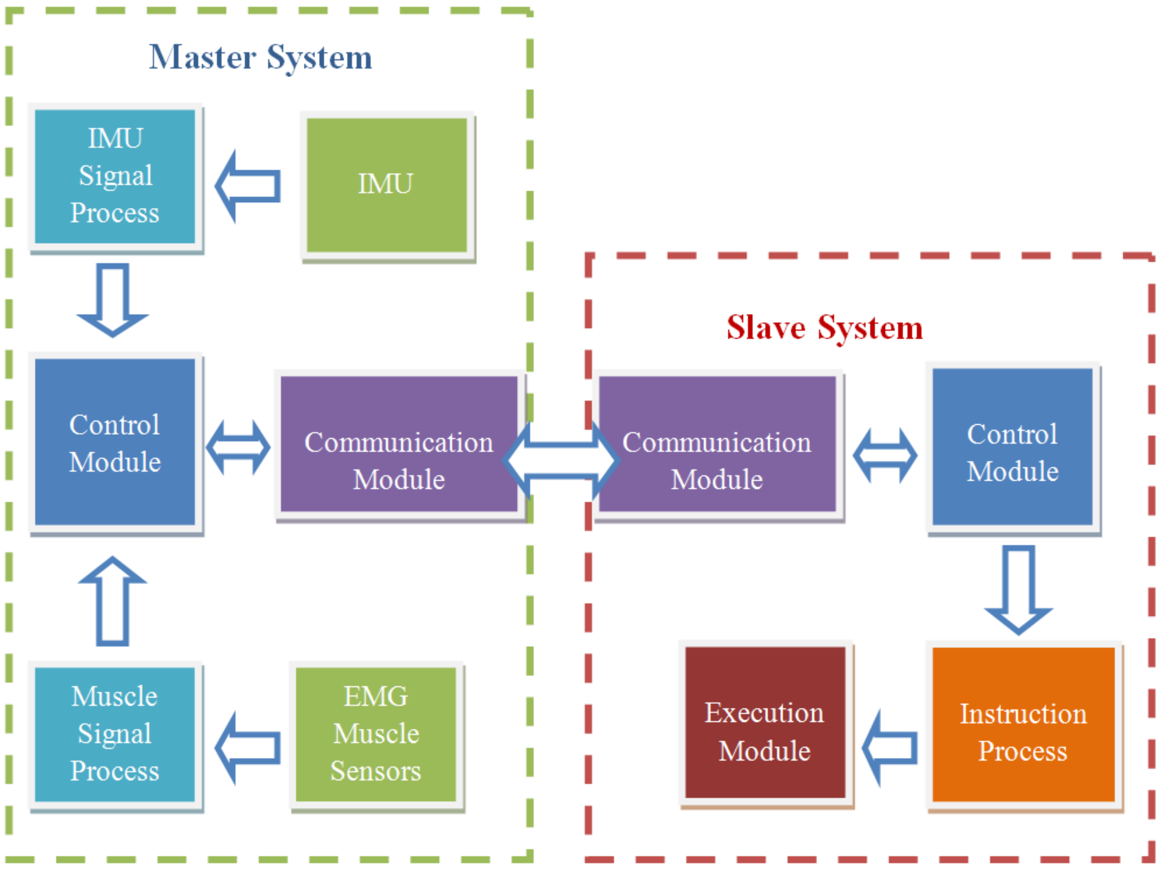
2.1 Overview

PhanTom consists of two parts: one master system with general interface and one slave system which can be used for different purposes.

In master system, EMG sensors and IMU will be located on the arm of user. Then the muscle signals and angle values will be processed by related circuits and sent to control module. Control module will use machine learning algorithm to recognize current gesture and send corresponding instruction to slave system via communication module.

The slave system has matched communication module to receive the order. Then its control module will control the execution module via peripheral circuits according to the order.

PhanTom is a general purpose control system, so users can develop various execution modules with different usages according to their demands.

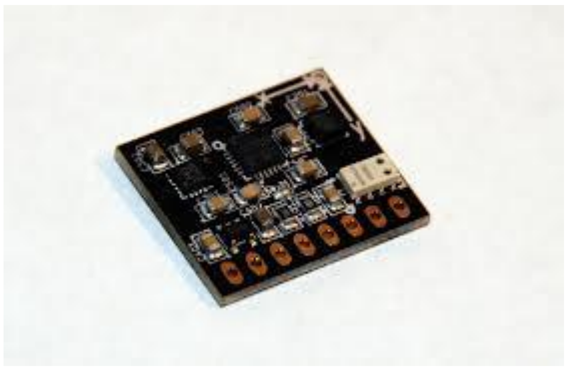


PhanTom Architecture

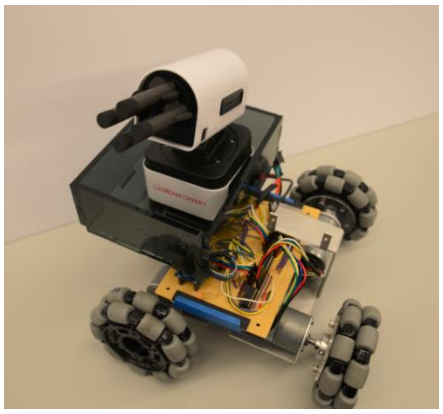
2.2 Core components

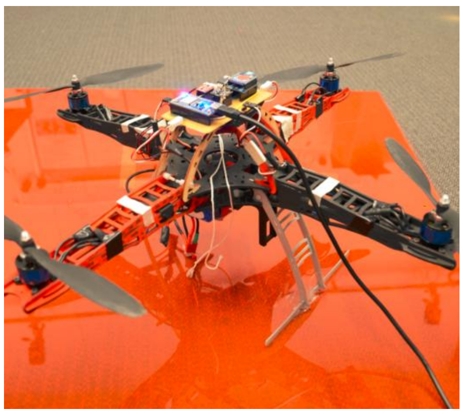
Currently, we use mbed LPC 1768 as the control systems, and the Xbee as the communication systems.

And following are the EMG sensors and IMU.



In slave systems, we build up a vehicle and a quardrotor.

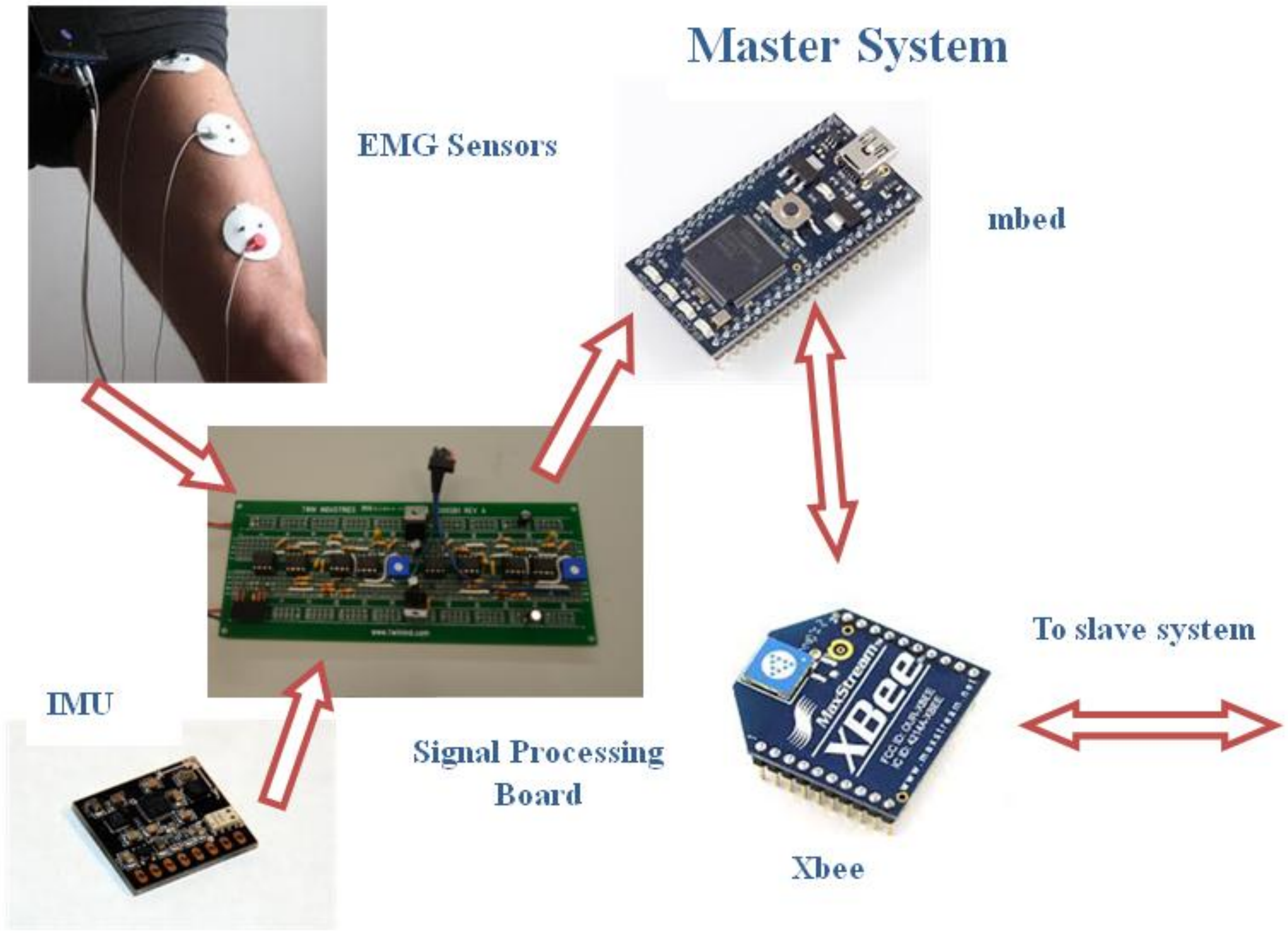




We also hacked and changed two toy cannons as the "weapons".

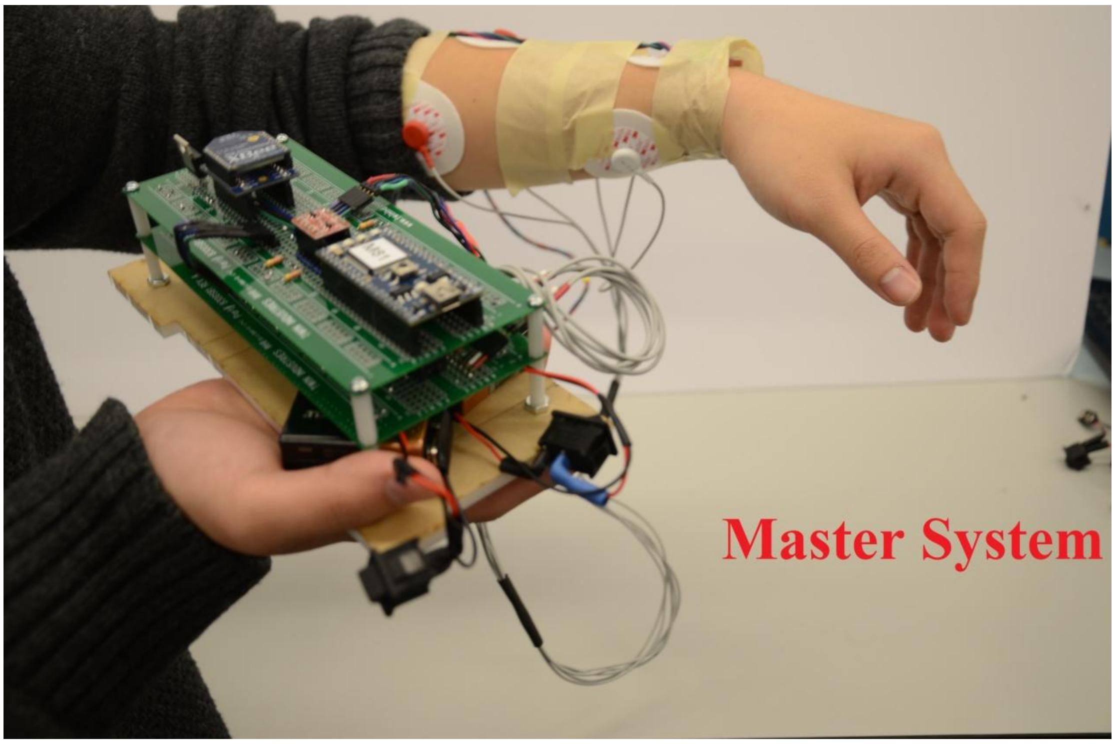
3. Mater System

3.1 Structure



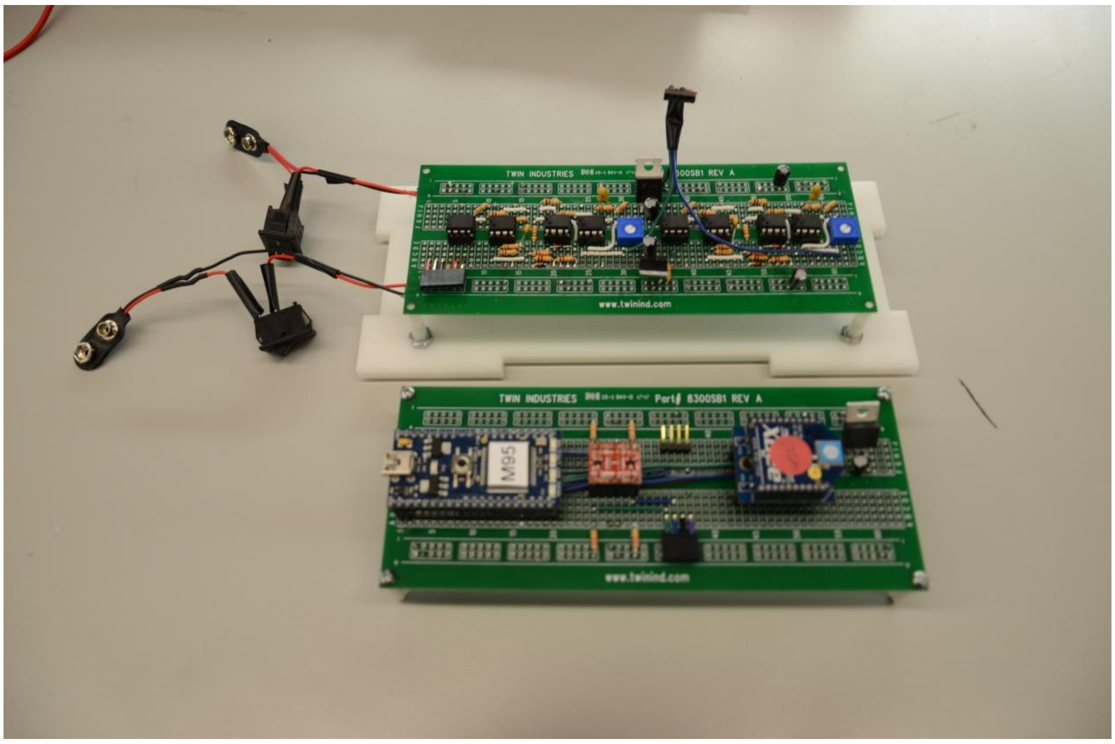
Master System

We have developed a portable master system with two 9V batteries. Five EMG sensors are attached to the user's left arm, they can sense the motions of two muscles. And the IMU located on the back of forearm will calculate the related angle. After sensed data collection, a two-stack board integrates all the following processes such as signal amplification, gesture recognition and instruction transmission.

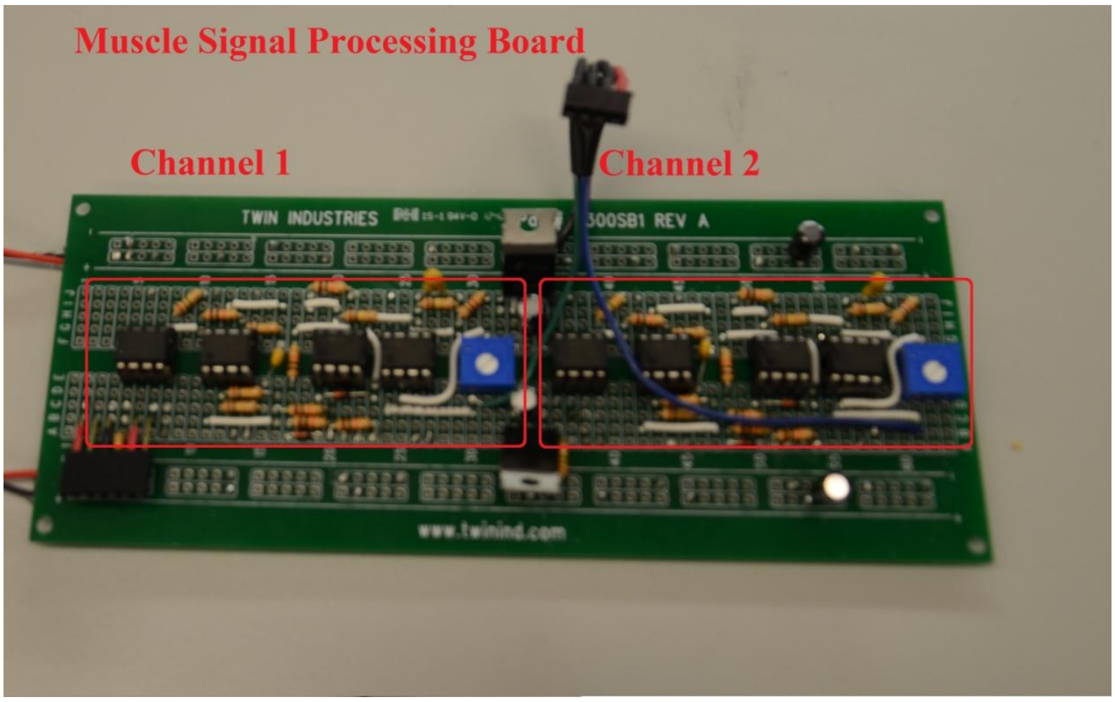


3.2 Hardware

The core of master system hardware is the two-stage circuit board.
The upper stack consists of the mbed, Xbee and a level shifter used to translate the data from IMU. The lower stack is used to process muscle signals. The sensors are collected to the board through several headers.

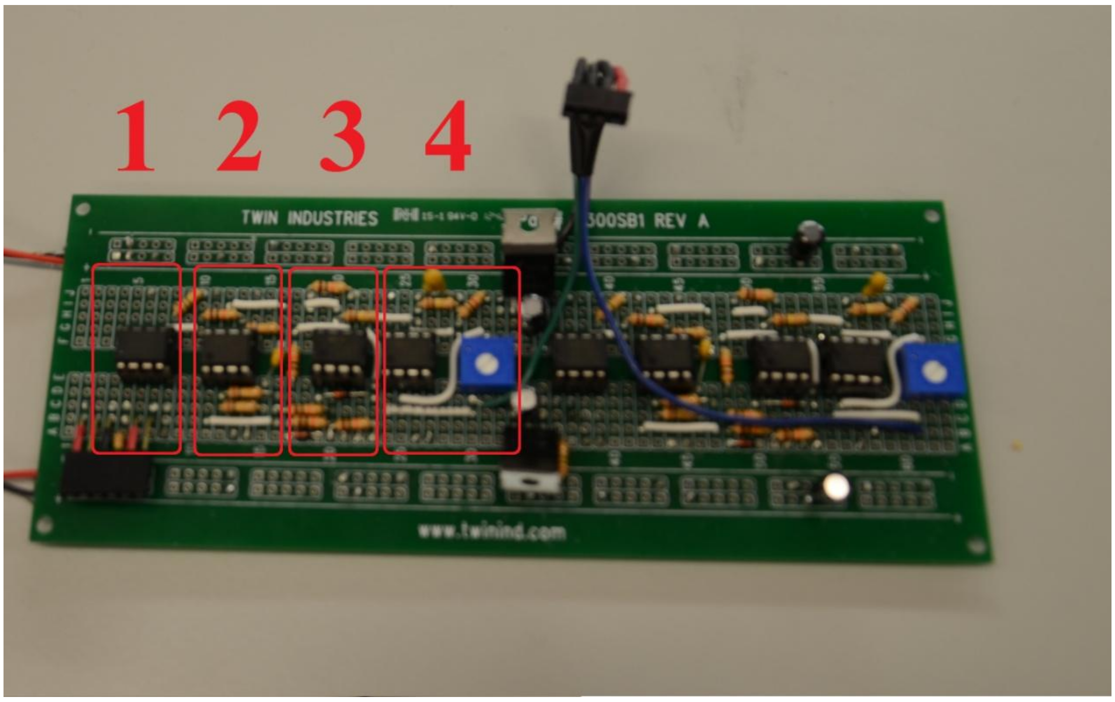

Master System Circuit Board

As shown below, the muscle signal processing board has two parallel channels, so it can recognize the statues of two muscles.



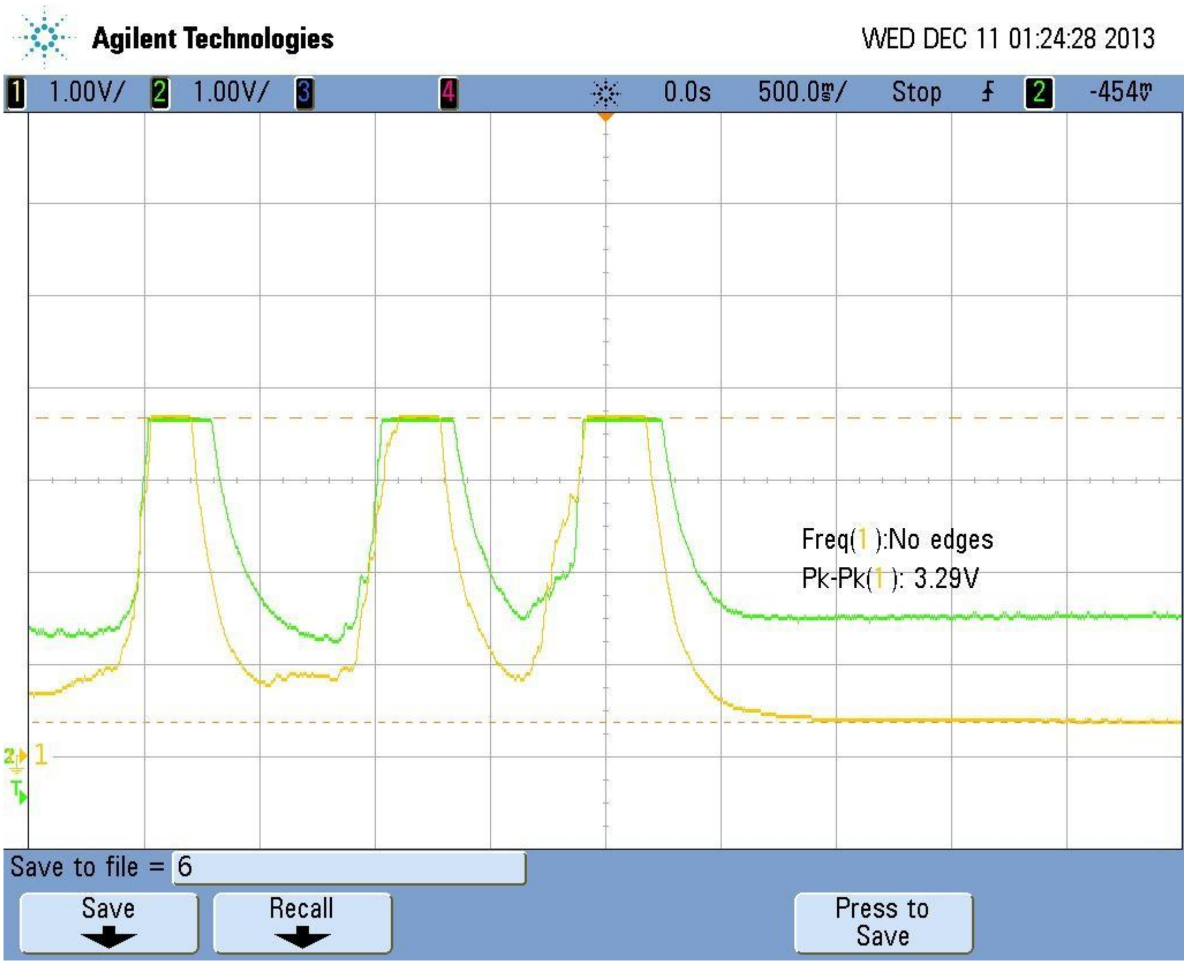
Muscle Sensor Processing Board

In each channel, the circuit can be divided into four parts. Part 1 is an instrument amplifier, it can extract the 500 uV muscle signal from the noises. Part 2 is a high pass filter and part 3 can regulate the waveform. At last part 4 can amplify the signal according to the value of adjustable resistance.



3.3 Software

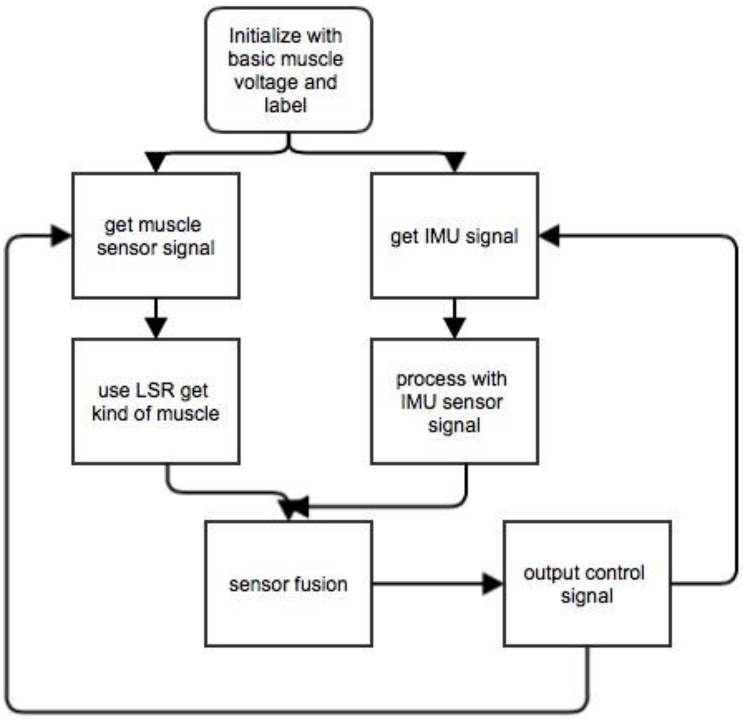
After processing the electric signal from muscle, we got the processed signal as follows:



This is the signal that we got. Then we needed to transform the signals into actionable data. As different people have different muscle voltage, and the same gestures may have different waveforms, but it must have the same trend. So we planned to use Machine Learning method to train the data and applied this to different people. Also we could get the features of some gestures so that we could cluster the gestures with the same trend. At first we know that by simply set a threshold for the signal would not work. As different people have different muscle voltage input, also during the testing process, there may be a lot of noise. If we applied a linear algorithm, the bias would be very big. But if we match the data too much, the variance would be too much. The following are the algorithm that we've thought about:

a. K-Means clustering. This is an unsupervised learning. Firstly, we should randomly choose K clusters as K gestures, then we calculate the distance between the gesture and the K gesture we choose and cluster this gesture to the one which is the closest. And then we got the new K clusters by calculating the average of the gestures which is clustered into one cluster. And then we do the above part again and again to recognize the gestures. However, this method consumes a lot of computation of the MCU, also the effect is bad as gestures are unknown. As a result, we can't use unsupervised method for the gesture learning. However, we could adapt the method that the gestures around some point can be clustered into one gesture, as they have the same trend. And this could avoid the noise that the same gestures may have different values.

b. Regression. We have also adapted linear regression for this system. For each gesture, we use a sampling frequency of 20Hz to take the analog value of the muscle voltage. Then we used a queue structure to capture the moving analog value. Firstly we initialize some labels for the signal. Then for the moving input, we calculated the moving queue with the labels of different gestures by using the least square error method and consider it as the one with the least square error. By using this method, we could recognize some gestures precisely, and sent out the consistent control signal.


Initialize with basic muscle voltage and label
get muscle sensor signal
get IMU signal
use LSR get kind of muscle
process with IMU sensor signal
sensor fusion
output control signal

Software Architecture

For the IMU part, we use complementary filter to get the angles. For the angle in x-axis and the angle in y-axis, apply low-pass filter on the accelerometer raw data and apply high-pass filter on the gyro raw data, and then combine them together to get the accurate angles. We need to change the parameter to make the IMU result response as fast as possible so that our device can be very sensitive to the change of the angle. For the angle in z-axis, since the accelerometer cannot contribute anything, we can only integrate the raw data of the gyro to get the angle which is not very accurate. We need to get rid of the bias to make the angle in x-axis more accurate and stable.

And the software architecture is as the diagram above:

Firstly, we initialize the system with the basic muscle voltage of the user and create the labels for different gestures. Then we consistently got the value of muscle sensor and IMU sensor. And the method is mentioned above. After we got the muscle sensor signal and IMU signal, we have to use sensor fusion method to combine these two signals to make it a better recognize the gestures and hand motions. As to sensor fusion, we design logics to make this two sensors integrate with each other perfectly. And the logic is as follows:

a) If gesture is recognized as no movement of hands, then it’s recognized as idle.
b) If gesture is recognized as put up hands, and x-axis of IMU is recognized as between +-30, then it’s recognized as forwarding.
c) If gesture is recognized as put down hands, and x-axis of IMU is recognized as between +-30, then it’s recognized as backward.
d) If gesture is recognized as put up hands, and the x-axis of IMU is recognized as minus -30, then it’s recognized as turning left. And the speed of turning left is calculated according to the angle of x-axis. The larger the absolute value of x-axis, the larger the speed.
e) If the gesture is recognized as put down hand, and the x-axis of IMU is recognized as minus -30, then it’s recognized as turning right. And the speed of turning right is calculated according to the angle of x-axis. The larger the absolute value of x-axis, the larger the speed.
f) If the gesture is recognized as put up hand, and the x-axis of IMU is recognized as above 30, then it’s recognized as shooting.
g) If the gesture is recognized as finger clipping, and the y-axis of IMU is recognized as above 60, then it’s recognized as changing mode.

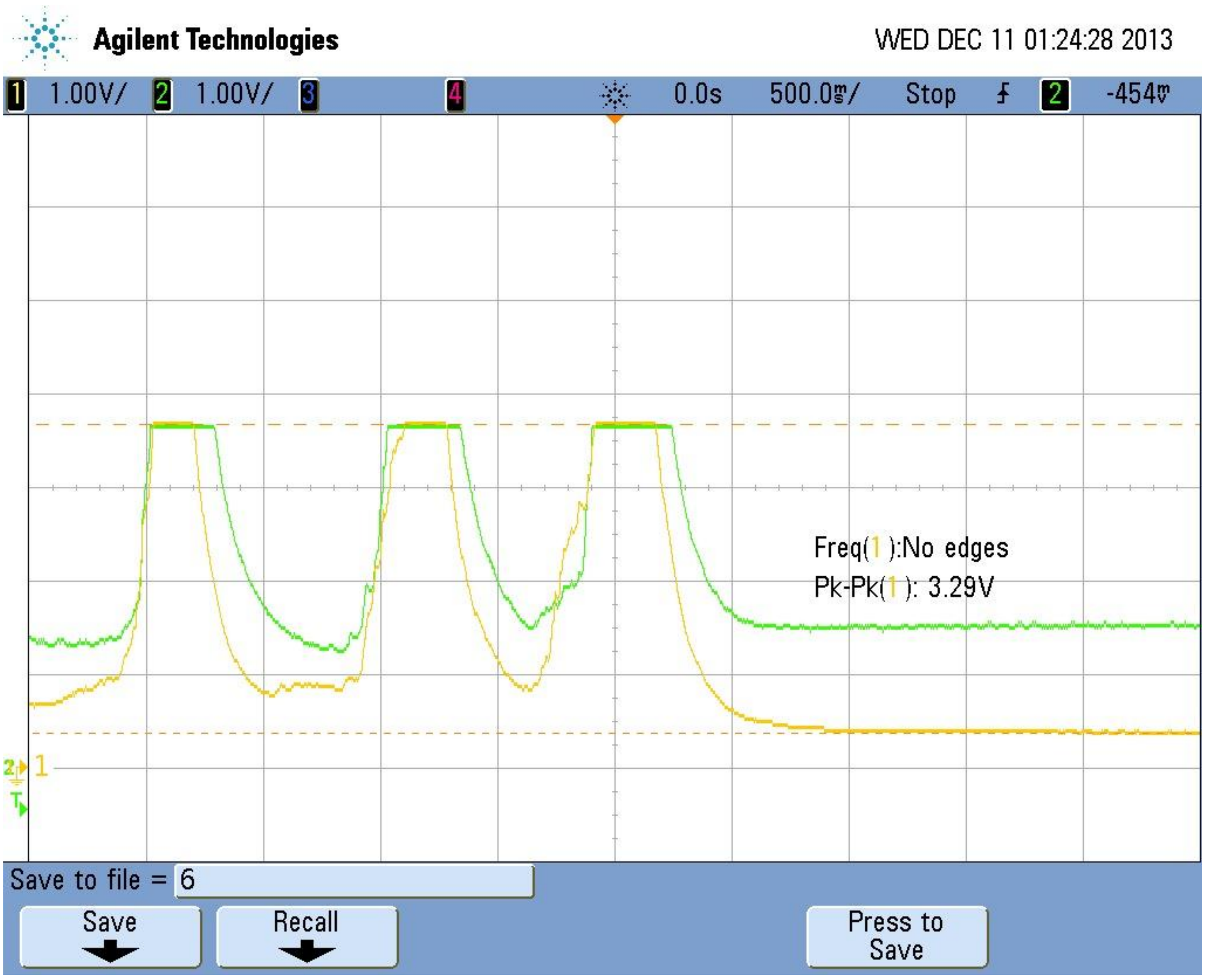
And after this simple sensor fusion, we could perfectly get some typical gestures.

3.4 Typical Gestures

We designed 7 typical gestures for the system.

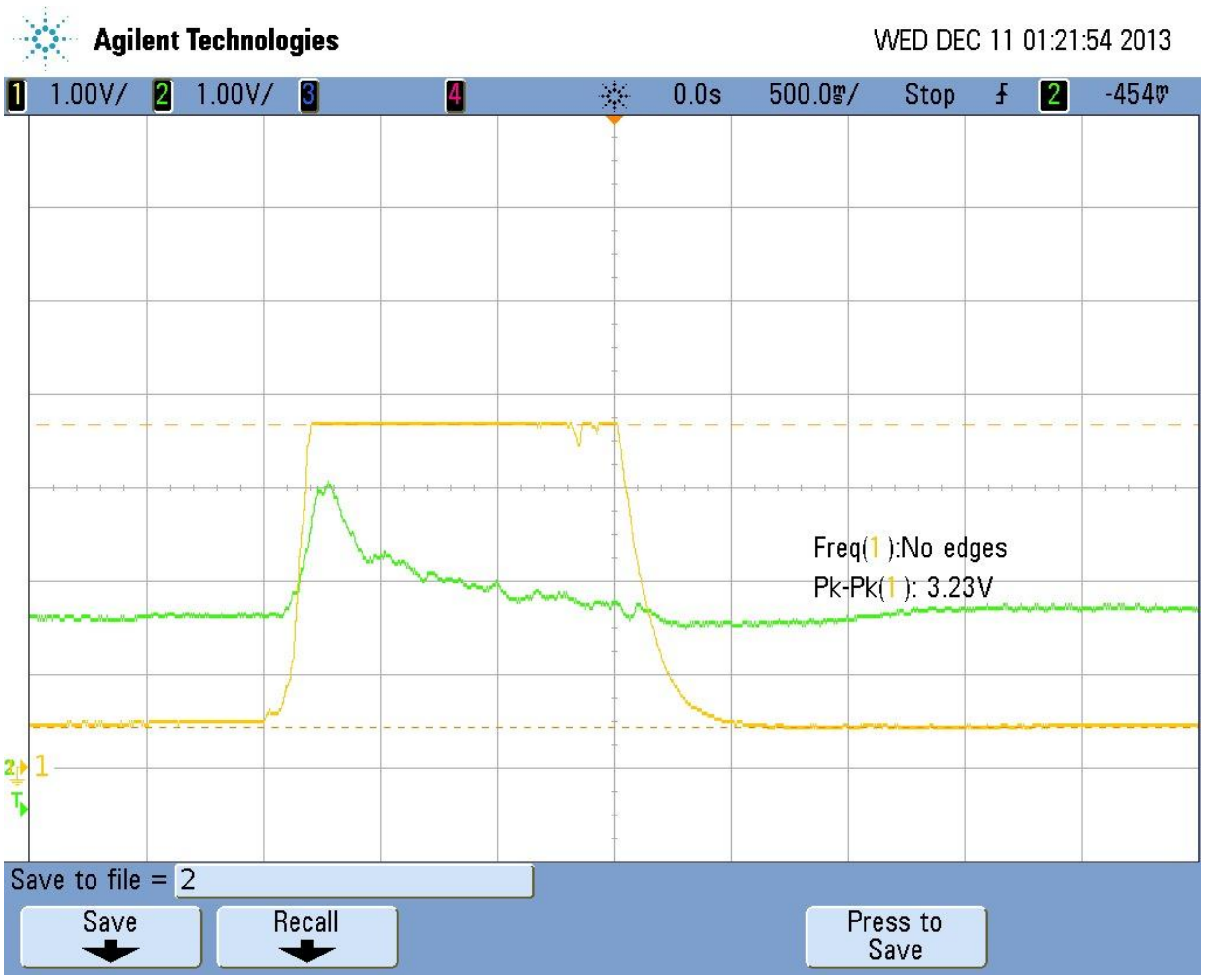
3.4.1 Change mode

We used a single finger clip as the gesture the change the mode. When we clip our finger, the mode changed to another one. Now we only have two modes for the attack system. One mode is the vehicle mode, which control the car to move around. Another mode is the canon mode, which we use gestures and hand motions to control the canon move up and down, turn left and right and shoot. The waveform of mode change is as follows:



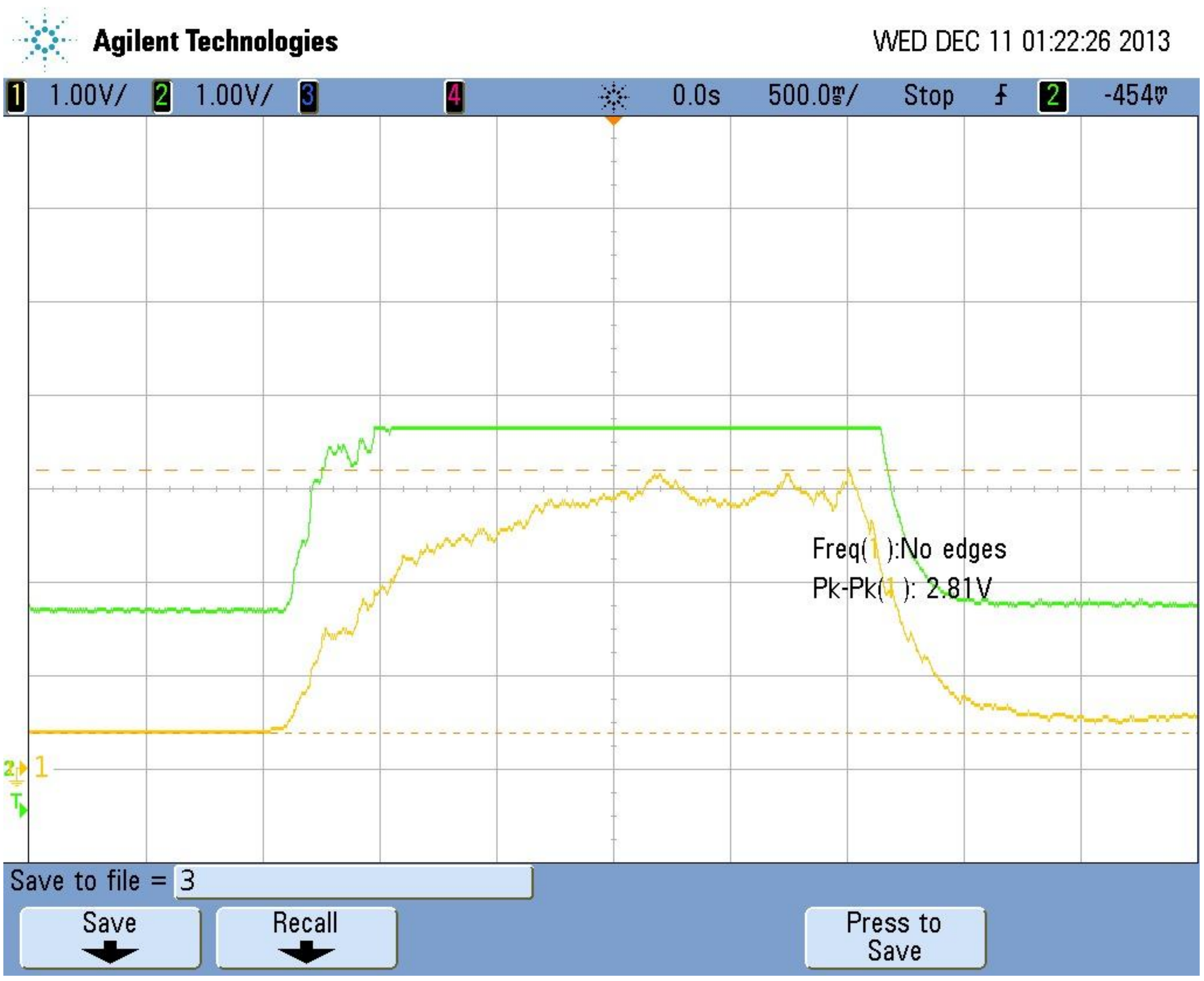
3.4.2 Moving forward/ Canon Up

We designed a gesture to control the car to move forward. When we stretch out our arm and put up our hand, then it's recognized as moving forward in the vehicle mode or canon up in the canon mode. And the waveform of this gesture is as follows:



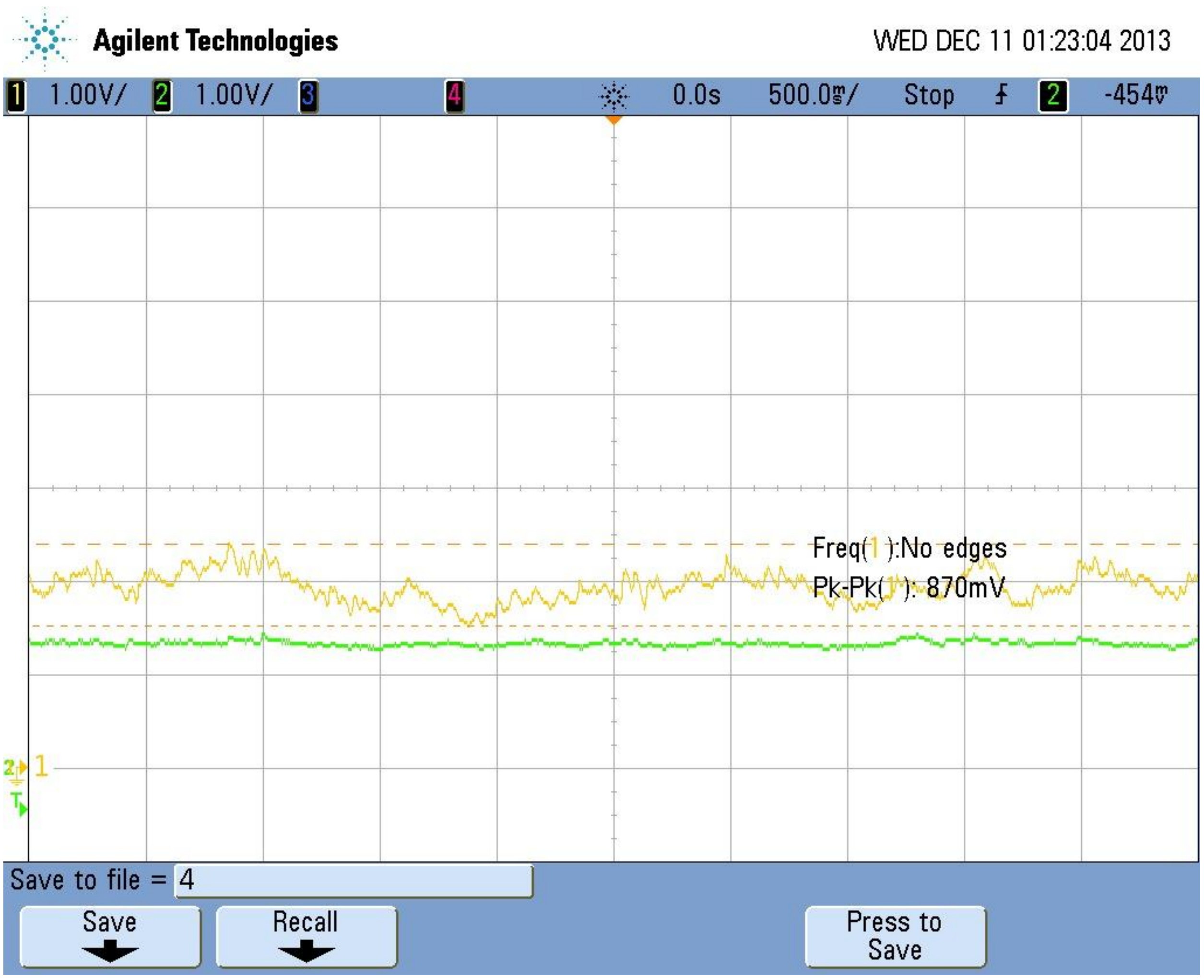
3.4.3 Moving backward/ Canon Down

We also designed a gesture to control the car to move backward. When we stretch our arm and put down our hand, then it's recognized as moving backward in the vehicle mode or canon down in the canon mode. And the waveform of this gesture is as follows:



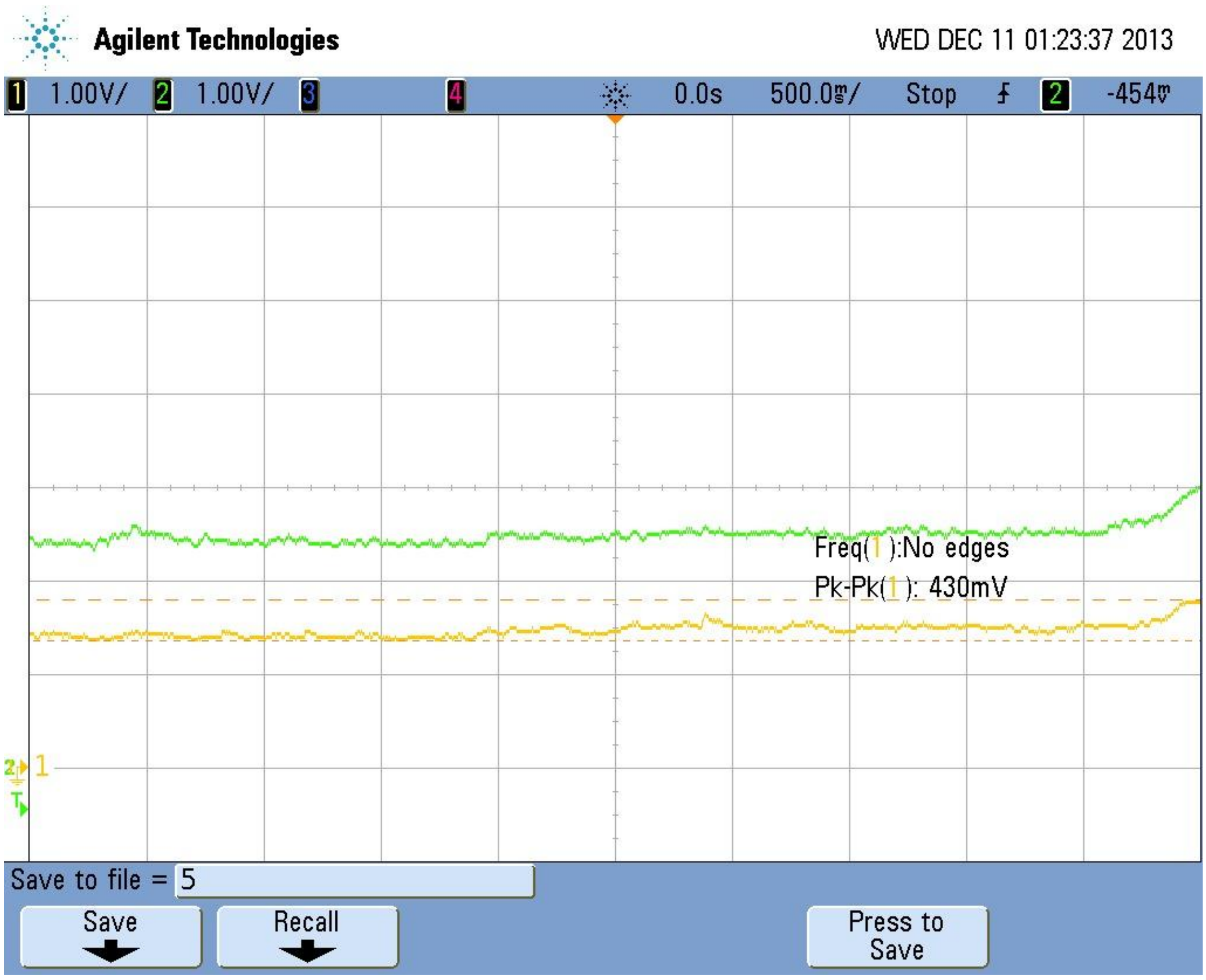
3.4.4 Turning Left / Canon Left

Then this is the gesture to control the car to turn left. When we stretch our arm and twist the hand to the left, then it's recognized as turning left in the vehicle mode or canon left in the canon mode. To make it smoother, we designed turning speed for different angles in the x-axis which is detected by the IMU on hand. For example, when the angle in x is less then 30, then the vehicle turn left in a slower speed. And we set three levels for turning left. When the hand is totally pointing left, and then the turning speed is the fastest.



3.4.5 Turning Right/ Canon Right

There’re also gestures for turning right. Same with turning left above, there’re also three levels for turning right.

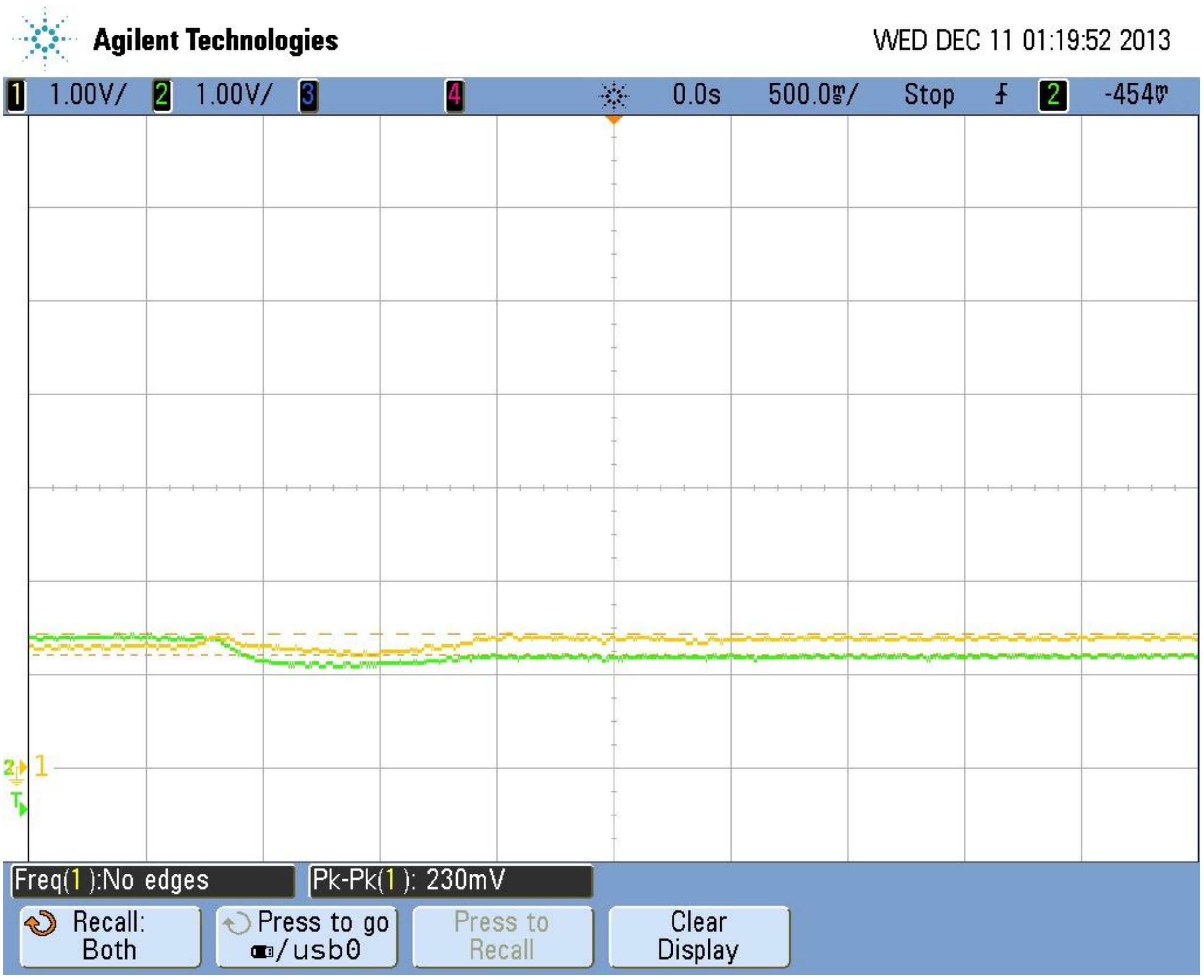


3.4.6 Shoot

Then we designed a gesture for the shooting. When we stretch our hand and make the hand turn left, then it’s the shooting gesture. It’s almost the same gesture with turning left, but we only set one level for it. Also, we use IMU to decide the direction of the hands. If hand is pointing right, then it’s shooting. And the waveform is the same with turning left.

3.4.7 Idle

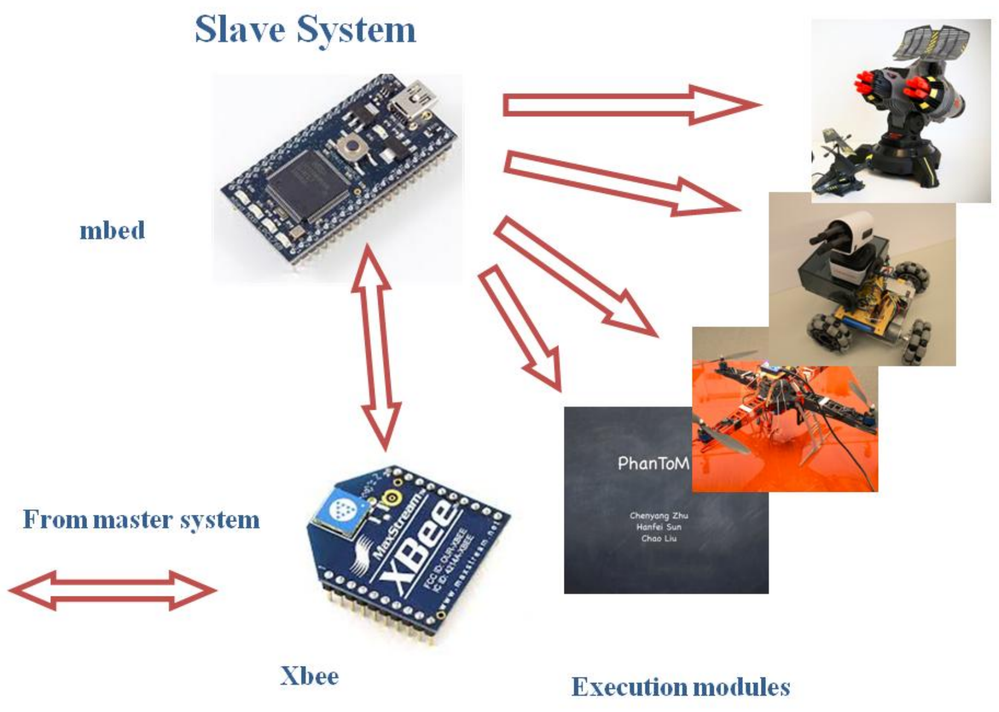
When we do nothing, the gesture is in the Idle mode. In this mode, nothing will happen to the vehicle canon. And we have tested some small movements, and they won't be recognized as the gestures that we specified above. Our system could recognize the Idle situation perfectly.



4. Slave System

4.1 Structure

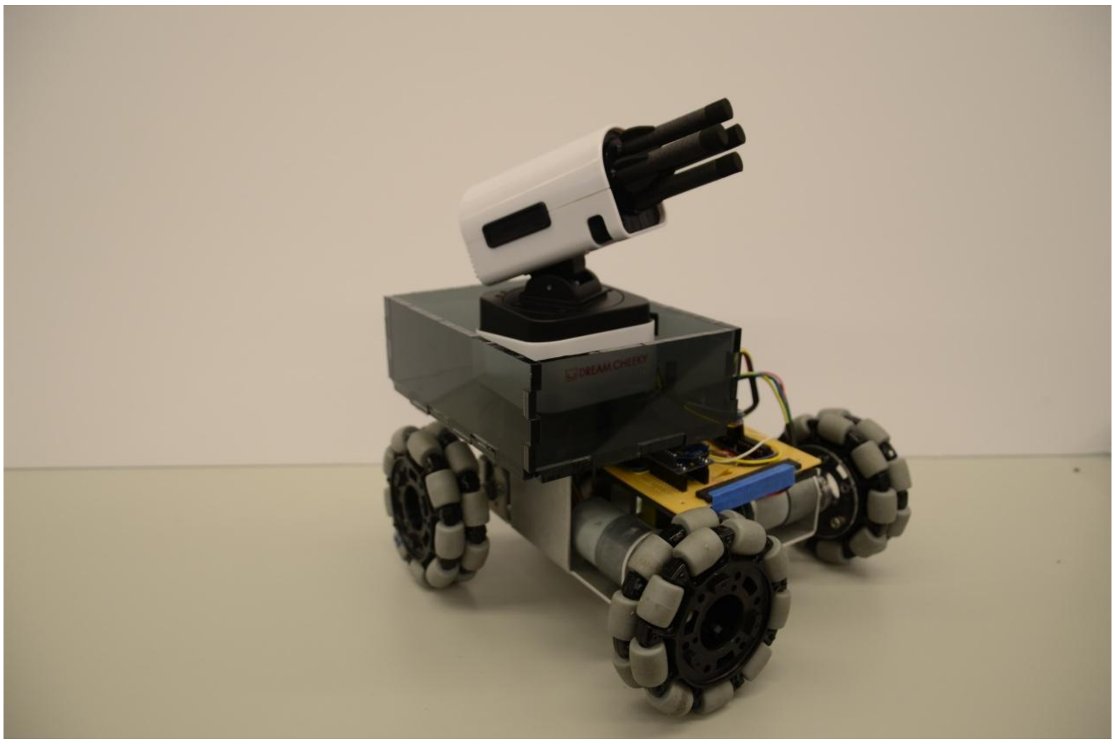
Similar to that in master system, all the slave systems uses mbed as the control module and Xbee as the communication module. But the execution systems and corresponding circuits are different among the slave systems according to the specific usage. We already developed four different slave systems: self-propelled gun, stationary cannon, quardrotor and gesture controlled slides.



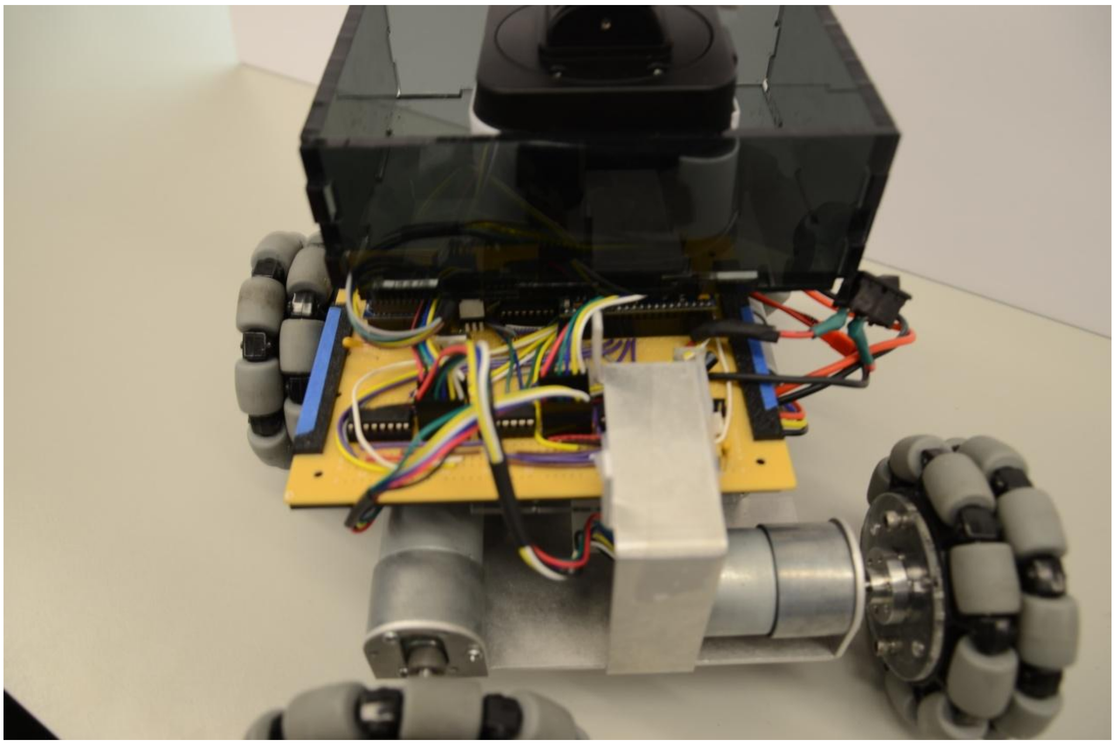
Slave System

4.2 Implementation 1: Self-propelled Gun

In this part, we built a battery supplied vehicle and a hacked toy gun. This system has two modes: vehicle mode and gun mode. A snap gesture can change between these modes. In vehicle mode, the car can move forwards, move back, move left straightly, move right straightly, turn left, turn right, clockwise rotate and anti-clockwise rotate according to the gestures. And in gun mode, the gun can aim up, down, left, right or shoot.

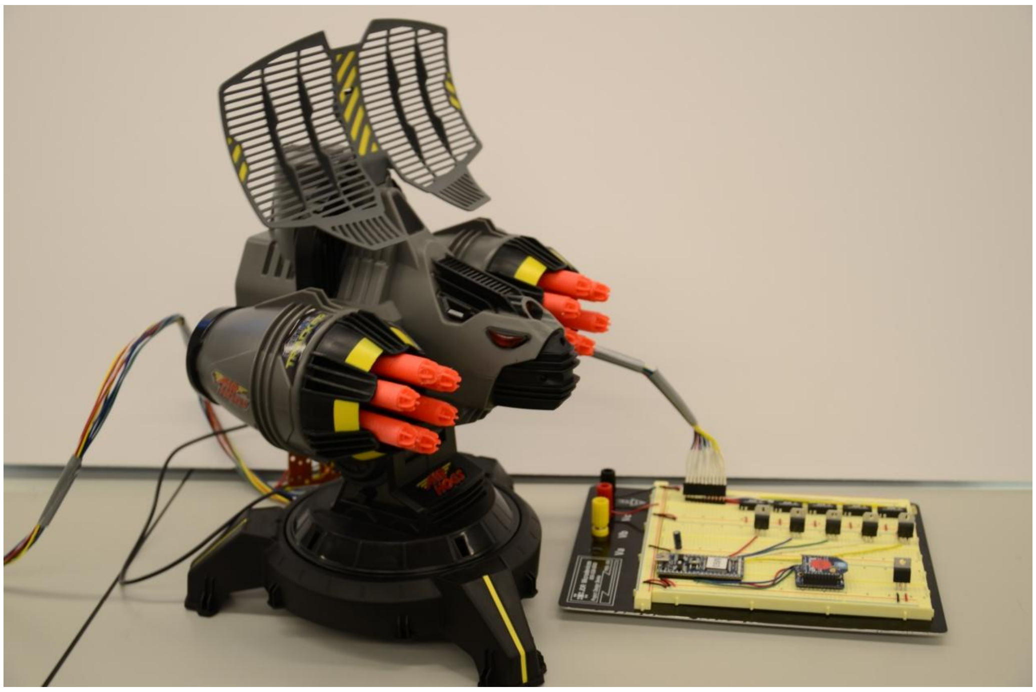


The car has four universal wheels and four motors. They are controlled by two H-bridges on the circuit board according to the master's instructions. And another H-bridge is used to control the gun. The board also contains mbed and Xbee as control and communication modules. A 11.1V battery is used to drive all the system.



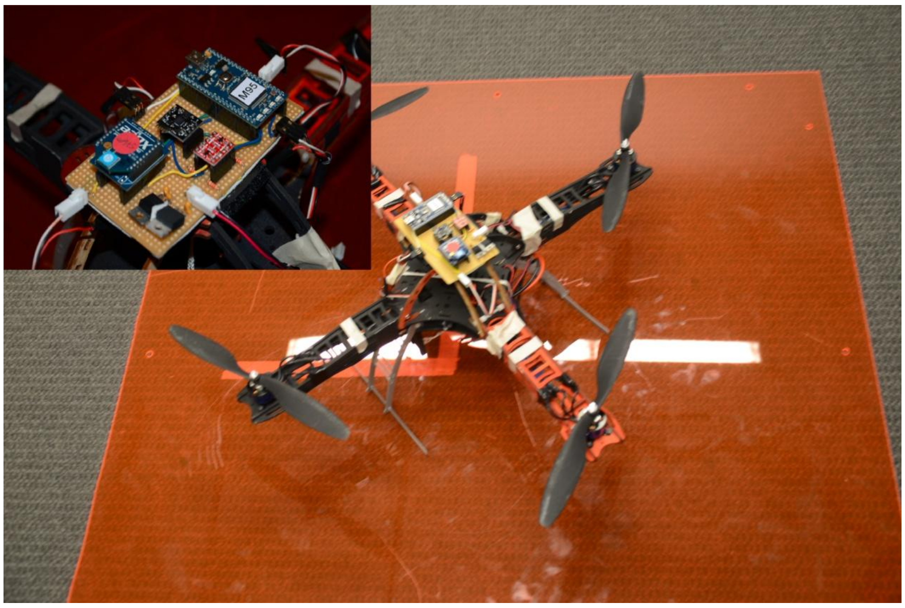
4.3 Implementation 2: Stationary Cannon

In this part, we hack a toy cannon as the defender in the game. We open the control panel and observe the signals when the machine is working, then replace the buttons with another circuit including mbed and Xbee. So the cannon can turn left, right, up and down according to the user's gesture.



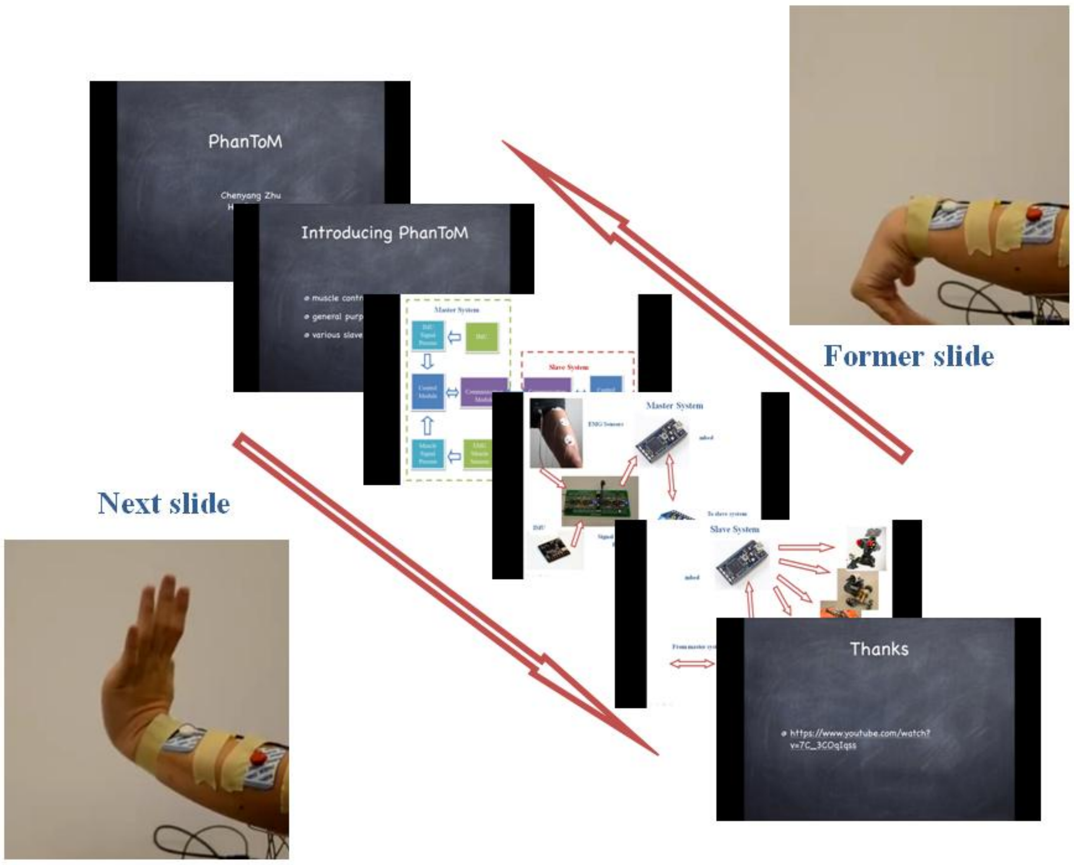
4.4 Implementation 3: Quardrotor

In this part, we built a quardrotor with existed motors and frames. On the circuit board, besides the Xbee and mbed used to communicate and control motors, we also integrate an IMU to calculate the angles during flying, which is used for PID control. The quardrotor is still under testing now.



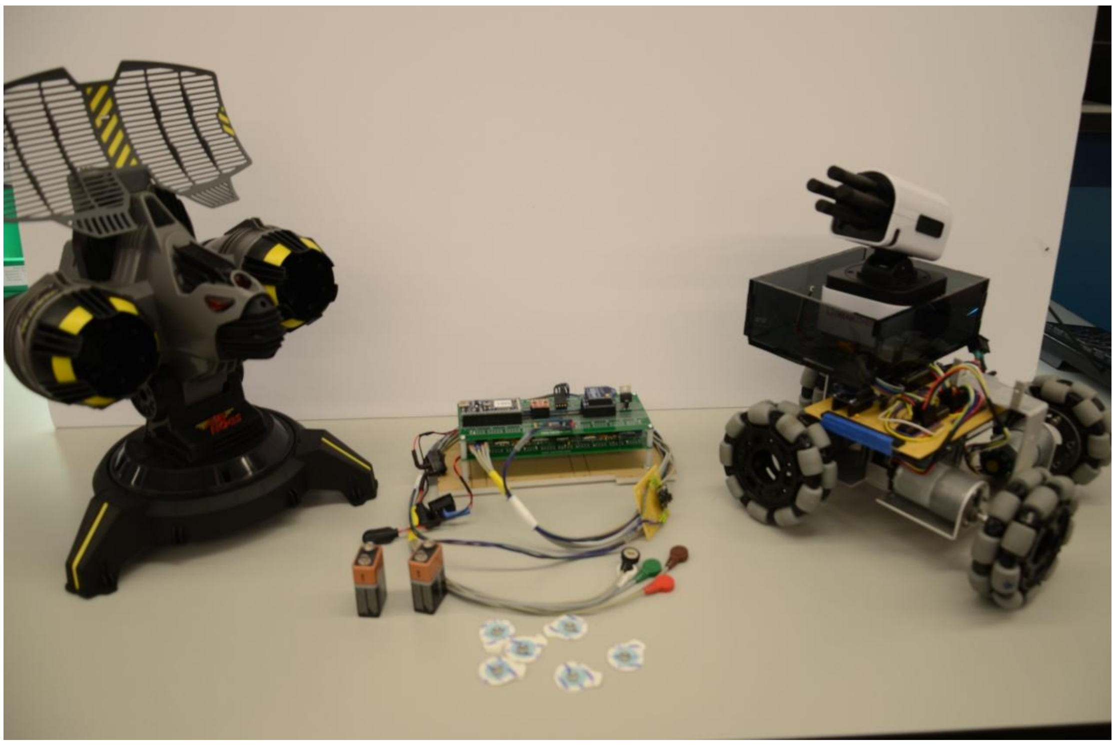
4.5 Implementation 4: Gesture Controlled Slides

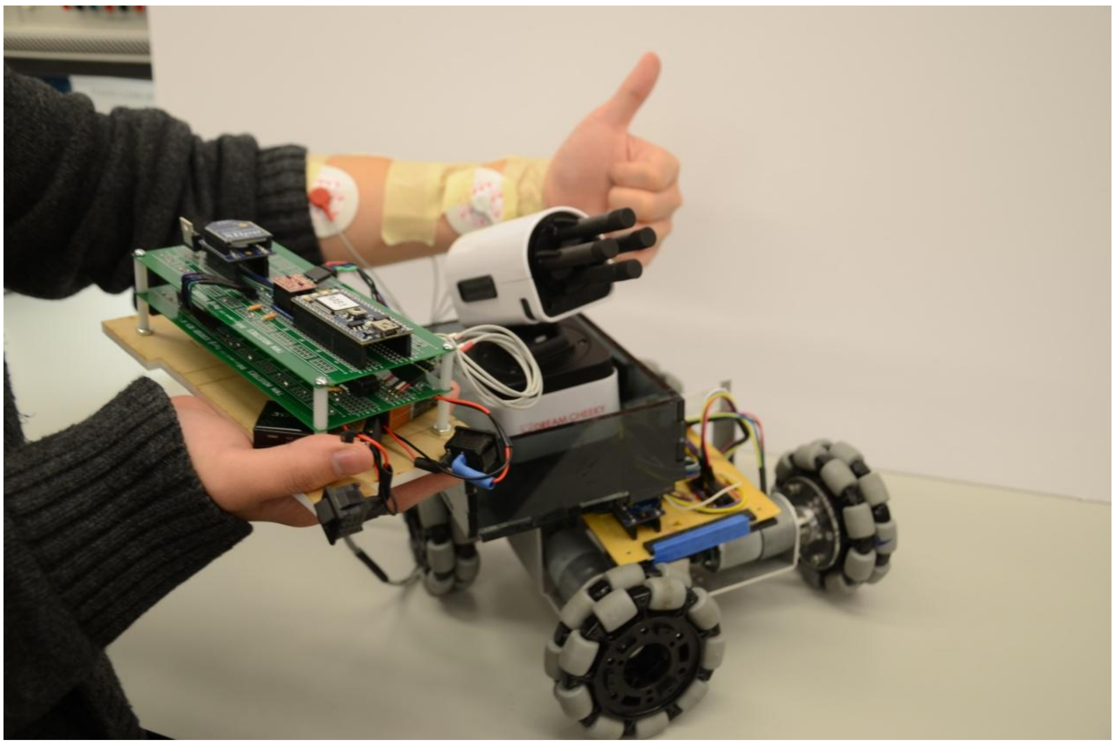
For this part, we used Matlab to play our presentation slides which can listen to our PhanToM system via xBee plugin in the computer. We use 2 gestures to control the slides to move forward or backward which is showed in the figure below.



5. Final Demo

In our final demo, it is a game between two players with two PhanTom systems. One player can control the self-propelled gun and another can handle the stationary cannon, then the two players can use their gestures to control their toys to shoot at each other.





6. More Information

Team members:

Chenyang Zhu

Second year master student, major in EMBS in Department of Computer and Information Science of University of Pennsylvania. Good at Software Engineering, Machine Learning, Algorithm, NLP.

Hanfei Sun

Second year master student, major in Electronic Engineering of University of Pennsylvania.   Good at Integrated Circuit design and PCB design.

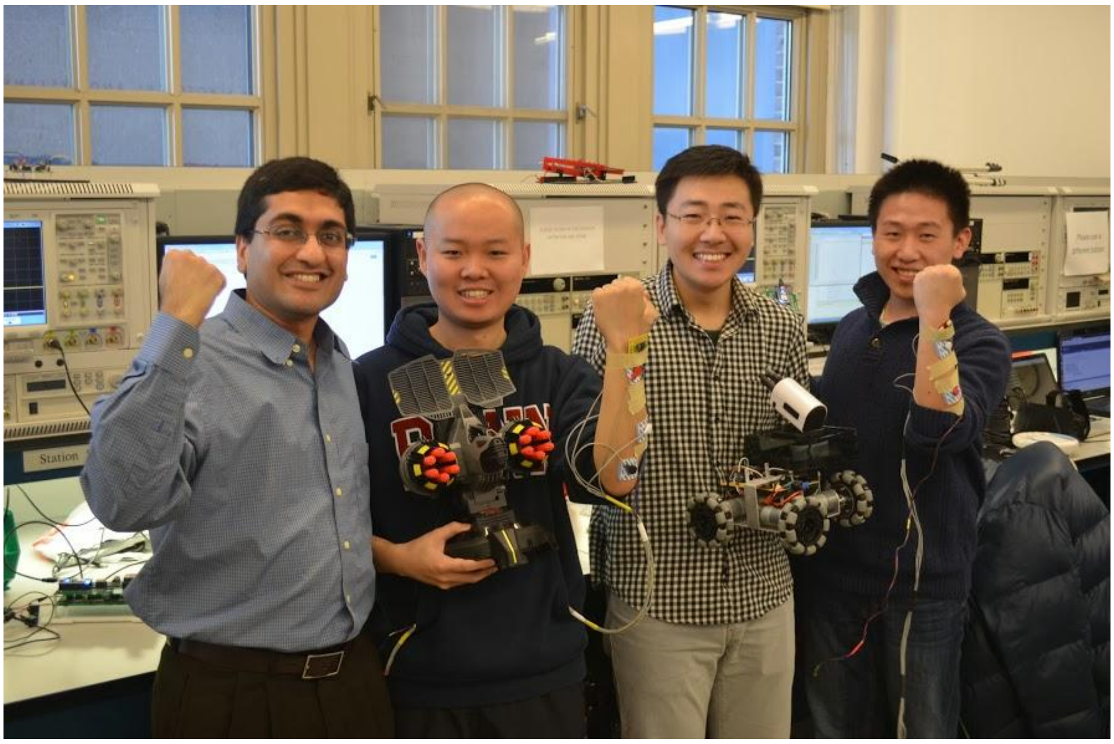
Chao Liu

Second year master student, major in Robotics in Department of Computer and Information Science in University of Pennsylvania. I vision myself being a competent engineer able to generate employable solutions with my multifunctional expertise. With stressful training via a lot of innovative projects including hardware, software and also mechanical design, I am confident that I can be a competent engineer.

Advisor:

Professor Rahul Mangharam

Rahul Mangharam is the Stephen J. Angelo Term Chair Associate Professor in the Department of Electrical and Systems Engineeringat the University of Pennsylvania. He also holds a secondary appointment in the Department of Computer and Information Sciences and is a founding member of the PRECISE Center. He directs the mLAB- Real-Time and Embedded Systems Lab at Penn. His interests are in real-time scheduling algorithms for networked embedded systems with applications in energy-efficient buildings, automotive systems, medical devices and industrial wireless control networks.



Please find more information on our website:
https://sites.google.com/site/phantomupenn2013/home