

ELEG498 Final Project Report
05/21/2017

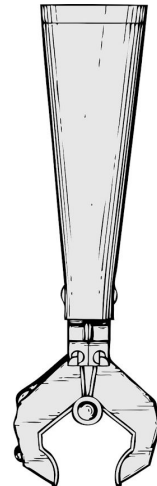
Project Title: Remote Controlled Quadcopter with Robotic Arm

Project Name: AMDRONE

Team members:

Ryan(Haoyuan) Wang
Jianbo Pei

The objective of this project was to design and build an a mechanical arm on a quadcopter with and camera to analyze and capture the given subject through the computer. The finalized



quadcopter would be able to catch a given subject and deliver it accurately and intact to a given location. It can be used in many occasions, such as carrier packages delivery, removing dangerous subjects, etc. Our original plan received a feedback commenting it “ambitious”; meanwhile, we discovered huge workload we are facing within a short school year, we decided to reduce the project to only building a remote controlled quadcopter with robotic arm which can be controlled grabbing objects and moving it to a different location.

The original plan was targeting carrier companies such as UPS, Fedex and USPS. Due to the cheap cost and fast speed of flying a quadcopter to a desirable location for short distance deliveries, this quadcopter plan can make tremendously saving on the fuel and employee cost

of carrier companies. And the other target customer is the police/army to utilize the robotic arm to remove dangerous/hazardous subjects. Because of the advantage of the flying, they can get much more closer to many places by the quadcopter that people normally cannot reach.

Because of the reduce workload of the project and we are currently building a prototype of the product, our final product is hoped to be easy to keep up with our original plan with minor modification and a little enhancement.

Our goal is to successfully build a robotic arm controlled by Arduino Pro Mini and embed it into a quadcopter. This goal involves an important parts for us doing research, designing and building: the robotic arm and quadcopter. The first step of success is the built robotic arm can be controlled wirelessly and running without glitch. The constraint is the mechanical structure of the robotic arm would be an obstacle for us to discover, and the arm is required to be light and nimble. Otherwise, it will be a huge issue for us later when we assembled the arm on the quadcopter and discovered the quadcopter couldn't lift the weight of the robotic arm. The building of the robotic arm would require many mechanical engineering skills. To build a quadcopter, we need lots of quadcopter experience. The biggest challenge is to test it. To successfully the quadcopter, we need to subtly adjust all parameters. When testing it, some minor errors could cause the quadcopter flip it over or damage it.

Quadcopter:

---Fall Semester

The other important part for this semester is to test the pre-programmed quadcopter. The pre-programmed quadcopter



we are currently using is Parrot AR. Drone 2.0 from the university's senior design lab. It is controlled by a fully designed and pre-programmed application from smartphones like iPhone through WIFI. The drone itself already has a HD camera which enables us to see horizontally and vertically, so after the robotic arm is successfully embedded into the Parrot quadcopter, we will use the camera to test our robotic arm to grab the desired object. Initially, the quadcopter couldn't be started and we assumed it might be some mechanical problems of the Parrot quadcopter. But later on we noticed that every time when we tried to charge the battery, there is a red light blinking on the charger all the time. We conclude it must be the problem of the battery that results the failure of starting the quadcopter. Later, we purchased a pair of new batteries and we successfully flew and tested the pre-programmed quadcopter.

After the battery problem was successfully solved, we started to test and learn the flying techniques of the pre-programmed quadcopter. The quadcopter could be used indoors or outdoors. The package of the drone includes a foam shell. When flying the drone indoors, the drone needs to be



equipped with the foam shell. Otherwise, the propellers are easy to break accidentally. The whole setup was pretty simple. First, we need to download an application from app store called AR. FreeFlight. After installing the software, we need to connect the battery to the drone. Once the battery is connected, the drone will automatically check the status itself. This process is supposed to take about two or three minutes. Sometimes, some leds did not light up. So, we

reconnected the battery until all the green leds light up. When all four the green leds under propellers light up, it means that the drone is ready to connect with smartphone. Then, open the wifi setting on the smartphone, if all steps were correct before, we will find an unlocked wifi which is the signal from the AR. Drone. Click it and the phone will successfully connect to the drone. After wifi is

connected, it is time to fly the drone. The drone can launch or land itself by clicking the icon on the application. The driving mode in the application has two virtual

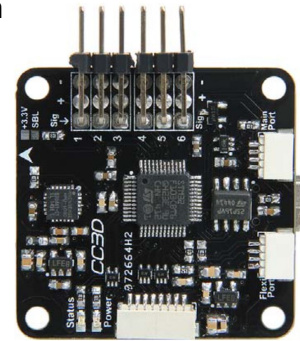


control pads. The left pad is for up and down. The right pad is for turning. Forward and backward are controlled by rotating the phone respectively. After flying the drone indoors and outdoors, we encountered some problems with it. First problem is that flying the drone indoors requires too much space. We tested it in the living room. The living room is about 20 meter square. However, the drone was very easy to crash. Second problem is that the drone was not steady outdoors. It was very difficult to fly it from one position directly to another position. The drone was very hard to control when it is windy. Moreover, we noticed that there was no switch on the drone to turn on/off each time finishing flying. Because our main goal is to assemble the robotic arm on the drone, we used different objects to test the weight that the drone is able to carry steadily. We first measure the weight for all the objects. After attaching different objects on the drone and testing it, we concluded that the drone is able to carry about 200 grams steadily at maximum. Therefore, we decided to focus on the arm only for now.

---Spring Semester

The problem we were having with the pre-built quadcopter Parrot AR. Drone 2.0 made us shifting to build our own quadcopter in the spring semester. The focus we have in the spring semester is heavily laid on the construction and testing of the self-build quadcopter. We started our work with an undone quadcopter that was left from last year's senior design class. The quadcopter was only left with the frame, 4 propellers, 4 motors and 4 ESCs.

The first thing we have to think of is what central controller we can use to direct signals to the quadcopter. Later on, Prof. Cotton recommended us to use a microcontroller called CC3D which is a flight microcontroller that used on most self-build quadcopters for beginners. It's a 32-bit microcontroller that can run at 72MHz, and it equipped with a 3-axis gyroscope and a 3-axis accelerometer. It has 6 channels with different pins on board, and everything can be programmed by the LibrePilot Ground Control Station software. The microcontroller can directly connect to the radio receiver that gether control signals from the radio transmitter and process the signal to drive the quadcopter with proper instructions.



The second equipment we have to acquire is then the radio transmitter and receiver. What we have is a FS-CT6A radio transmitter and receiver. They are both at 2.4GHz frequency band and have 6 channels. However, at the first, we couldn't connect these two devices together, but later we learnt a technique called transmitter and receiver binding. It requires a match line that can short the signal pin and negative pin on the receiver. We power

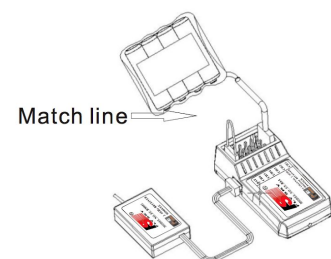
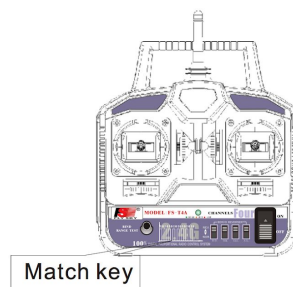


Figure 1

the transmitter and receiver and press the match key on the transmitter, then we have two connected transmitter and receiver. So the control signal can be sent from transmitter all the way to the microcontroller. When we tried to pair the control signal sending from transmitter to the microcontroller through the LibrePilot software, we were stuck at a point that the software requires us to have a flight mode switch on the transmitter, but the microcontroller doesn't respond to the transmitter's physical switch. Eventually, we upgraded the LibrePilot software and the new software allowed us to skip the pairing process, so we don't have to have a flight mode control function.

When everything has set up already, we test to fly the quadcopter that tights with two long thin sticks to prevent it from flip over on taking off. But the first problem we met is that one of the 4 motor's shaft broke and the propeller flew off. We had to change a new Turnigy D2830-11 brushless outrunner



motor and calibrate the motor again. Also we used a glue to stick the propeller's holders and adapters incase they flew off again. When we were actual testing the quadcopter, it flight not very steady with the long light sticks but it can

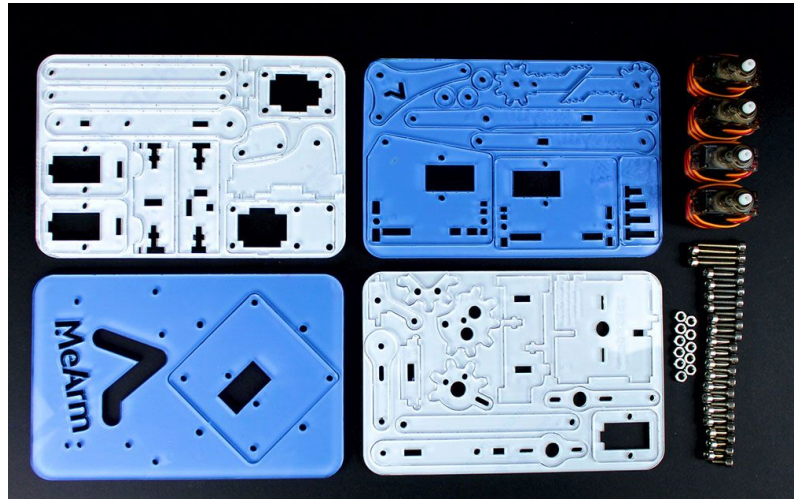


be controlled. But later on we took away the sticks, the quadcopter kept flipping over on taking off. We figured it is the motor output configuration problem. But we calibrate the motors many times, the motor output are still not as desired as we wanted.

Robotic arm:

---Fall Semester

From the hardware perspective, our approach is to work on assembling the robotic arm first. The robotic arm we are using is MeArm Nuka. The reason we chose it as our robotic arm is because of its light weight. The whole arm only weighs about 230 grams. Due to the



lift power of our current pre-programmed quadcopter is very limited, and it needs to handle the weight of both robotic arm and targeted project, we decided to choose a lightweight robotic arm as light as possible. In addition, it is an open source robot arm kit with a collection of laser cut acrylic parts and 4 hobby servos which made us assemble the robotic arm much easier. Besides the advantages mentioned above, the robotic arm has a 4 degree of freedom feature. It includes the base rotating, up and down moving, forward and retrieve, and grabbing or loosening. When integrating the robotic arm on quadcopter, we need to consider the directions carefully because the robotic arm is intended to sit on the table. However, when mounting it on the quadcopter, the directions we programmed are going to reverse. Also, the power distributed to the Arduino which controls the robotic arm needs to be regulated carefully. Another challenge is the integration of Arduino and Raspberry Pi. Arduino as a subsystem needs to receive signal from Raspberry Pi, so the signal transfer between Arduino and Raspberry Pi has to be prompt and accurate. If we achieved this goal by building a lightweight robotic arm and integrated it

onto the Parrot quadcopter earlier than we expected, we will be starting to work on designing and building our own quadcopter with more lifting power.

During the assembling process of the MeArm, we found that the robotic arm is too light and the material is very fragile. Therefore, we had a small incident during the assembly process: we accidentally broke the left support arm. The assembling process was not that easy as we expected. The arm we purchased is version 1.1. However,

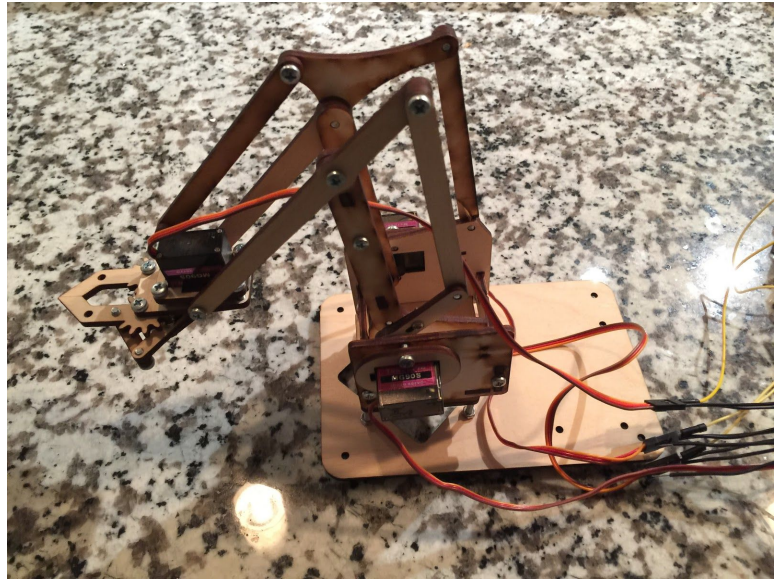


there was something wrong with the website where we purchased the arm. We couldn't find the instructions for the version 1.1. There was only instructions for version 1.0. Therefore, we followed the instructions for version 1.0 and started to assemble the arm. Although the only tool we need is screwdriver, because the parts are fragile and the screws are small, it was very hard to assemble them together. We didn't noticed that the parts have front and back side in the beginning. So, sometimes we used force to connect the parts tightly. Because some parts of the version 1.1 arm are different from the version 1.0 arm, we need to think the correct way ourselves. When assembling the parts as shown above, because the holes didn't match each other, we used force to install them. Because the parts are fragile, the part was accidentally broken. We tried to use tape to stick them together, but the broken place which is the rotating point didn't work with tapes on it. Therefore, we decided to ignore it first and continue the

assembling process. Eventually, the arm was successfully assembled. It took much longer time than we expected.

---Spring Semester

Because the robotic arm we constructed before broke in the left support arm, we invested a new robotic arm. Considering the weight than the quadcopter can lift, we chose a similar robotic arm kit. The robotic arm kit is basically the same as before. It has four motors and four degree of freedom. However,



the material is not the same. The new robotic arm is made of hard wooden material, which increases the durability and strongness. The weight of new robotic arm is 50 gram larger than the plastic one, which is 280 gram. The new arm could lift up to 100 gram things.


The power supply we needed for this robotic arm is the same as before. Because the quadcopter could provide the needed voltage for robotic arm, we don't need the battery attached to it any more, which reduced lots of weight. Also, we used a breadboard before to connect the wires. Because the breadboard contributed a lot weight for the robotic arm package, we welded needed wires together, which helped us get rid of the breadboard. Without the battery pack and breadboard, the weight of robotic arm package has been reduced a lot.

Robotic arm coding:

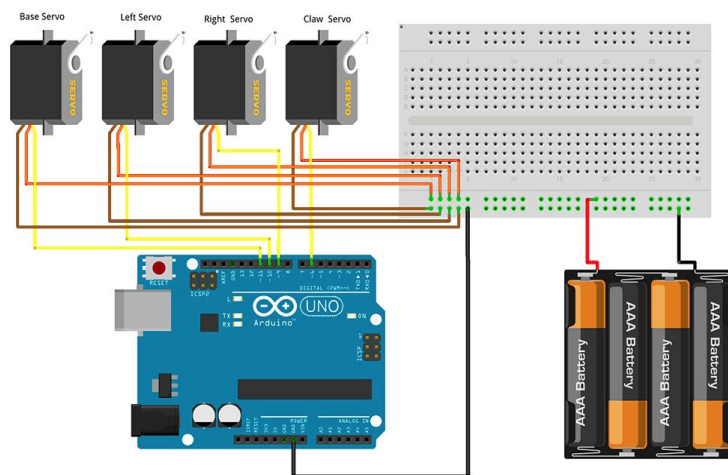
---Fall Semester

From the software aspect of the robotic arm, we first have had to acquaint ourselves with programming it with Arduino microcontroller. At the beginning, we had to have Arduino IDE software installed so we can transfer source code to the Arduino via a USB cable. We found some already written C code that can generate correct PWM signal from Arduino Uno to the MeArm servo motors. We have just started to experiment and test the robotic arm with the source code we have found, and we will modify the code to adjust the robotic arm for a better performance later. Once we finished our experiment on robotic arm and everything has set up, the robotic arm is ready to accept separate movement commands. We plan to use MeCon MeArm motion control software. It is integrated with position control capability and multiple servos control panel. It would be a great control system on our desktop end.

At the beginning of programming the MeArm robotic arm, we downloaded the Arduino IDE. The IDE has a very clear layout that we can easily input our C code into the coding area. We

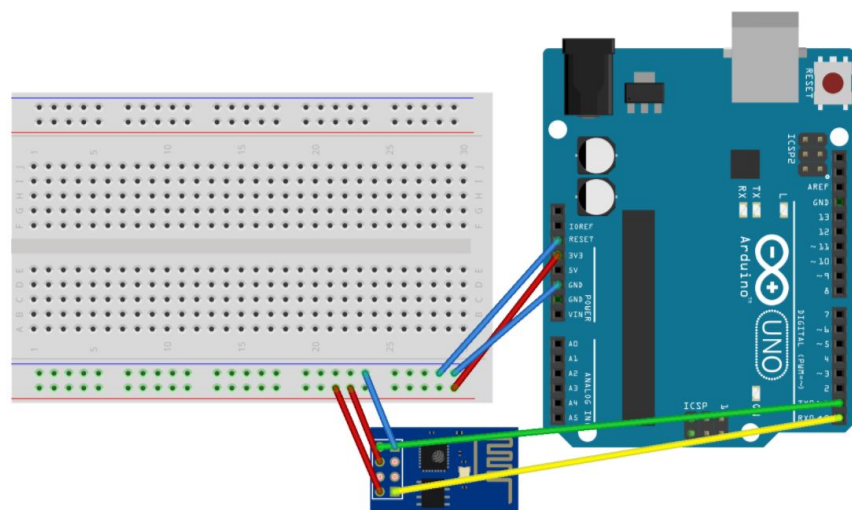


```
1 #include<Servo.h>
2 Servo servoC; //25-90 release-grab
3 Servo servoR;
4 Servo servoB; //Base: 80-110 right-left
5 Servo servoH;
6
7 void setup(){
8   servoC.attach(3);
9   servoR.attach(5);
10  servoB.attach(6);
11  // servoH.attach(9);
12 }
13
14 void loop(){
15   servoB.write(150);
16   servoR.write(10);
17   servoC.write(25);
18
19   //grab
20   for (int i = 25; i<120; i++){
21
22     servoC.write(i);
23     delay(50);
24   }
25 }
26
```

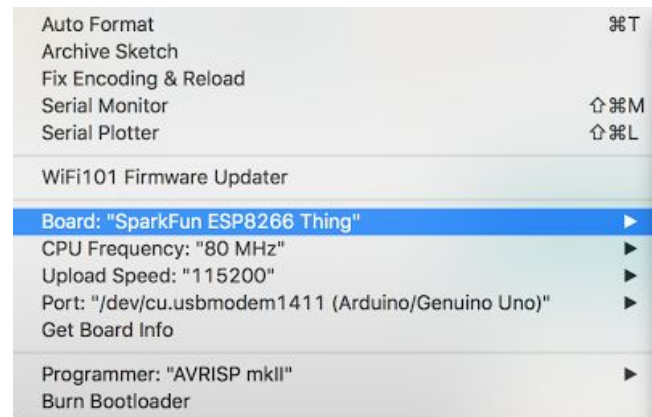


started to define 4 servo pins to separately control the 4 different servos, and we used pin 3, 5, 6 and 10 which all can generate PWM signals to send control signals to the servos. A breadboard was used to connect 4 AA batteries to the servos to supply enough power. After everything was set up, we started to test the extreme case of each servo to test what integer degree should be the maximum, and we got the base servo: 0(clockwise 90) - 180(counter-clockwise 90), left: ? (one of the arms to support vertical direction is accidentally broken as stated above, the effect of the broken arm might result the left servo doesn't work), right: 10-100, claw: 80(widest)- 140(grab) - as the value increases, the claw will grab harder. We built a simple loop code to let the robotic arm grab a box and lift it, then move to another place and drop it to test the arm's function. The code is perfectly working on the arm, and it can grab a certain size stuff from one fixed place to another fixed place. The video is uploaded to the Youtube, and next we figured we should be able to control the arm instantly. We looked up and found that we can control the arm with Python instructions. So we installed the python 2.7.12 and Pyserial 2.7. We learned that the Pyserial is used to support compatibility with serial ports in Arduino, so the Python code can directly send instructions to the arm, and we successfully controlled the arm with instant code instruction from Python.

After setting up the software interface that we can directly control the robotic arm with wiring to the computer, we decided to move to remotely control the robotic arm from a

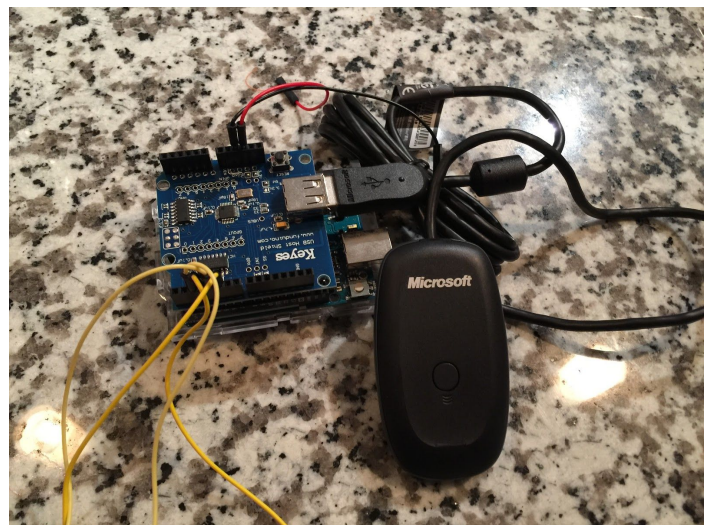


wireless remote controller. The first idea came to us was to use WiFi signal to control the robotic arm. We borrowed a ESP8266 WiFi module which compatible with Arduino. And B=before programming the ESP8266 WiFi module, we installed the support driver for ESP8266 from Boards manager in Arduino IDE. Based on the instructions for setting up the module, we built it up with every instruction on the manual. And as shown in the graph the red LED is on means it has successfully integrated on Arduino. Finally, we have successfully get WiFi module connect to our laptop WiFi.



---Spring Semester

During fall semester, we successfully controlled the robotic arm with Arduino through Pyserial. However, the transmitted signal was sent through USB from computer, which was wired. Although the WiFi module could transmit signal, it is still not very convenient to carry a computer all the time. Therefore, we changed our signal transmitting method from using a computer to a more portable device, Xbox 360 controller.



We did some research online and found that someone successfully configured a Xbox 360 controller to control a robotic arm through Arduino. So, we decided to adopt this method.

The required components are Xbox 360 controller, wireless receiver, and USB shield for Arduino. The coding platform went back to Arduino IDE. Because Arduino IDE already has built-in functions for Xbox 360 controller, we could directly call the function for Xbox 360 controller to configure it. After we did some research for tutorial to teach how to configure Xbox 360 using Arduino IDE, we started to write code step by step. Initially we configured four controls for four motors. Left/Right bumpers are for base rotation. Y axis moving for Left joystick is to control reaching and retrieving. Y axis moving for Right joystick is to control moving up and down. D-pad is to control opening and closing the claw. Also, the start button is to initialize the robotic arm with starting state. Later, we added two buttons. B button is for detach all motors. Considering when users don't want to move the robotic arm unintentionally, especially when flying the quadcopter with grabbing target, button B could prevent unintentional touch happening. If users would like to take the control back, they could press A button to reattach all motors. The code for configuring the robotic arm is in our github.

The photo below is the whole package for the wireless controlled robotic arm:



Conclusion:

---Fall Semester

To conclude what we have done so far for the fall semester, we tested a pre-programmed quadcopter Parrot AR. Drone 2.0, and found it is not stable enough to carry a certain weight we wished to have. We built a robotic arm called MeArm and had a small accident which we tried to fix later. Then, we used Arduino and Python to successfully control the arm by computer. The broken arm was fixed using tapes but affected our later experiment with the lifting ability of the robotic arm. So, the data that we collected for the arm was not that precise. We aim to be able to rebuild a robotic arm with fully function of rotating, lifting and grabbing. The next semester's goal will be successfully build a quadcopter with our own parts and fully integrate the robotic arm onto the quadcopter.

---Spring Semester

The spring semester is the final phase of our senior design project. We rebuilt a new wooden robotic MeArm and implemented it with a new control method. We used both Microsoft XBOX controller and receiver on the robotic arm and programmed it with Arduino Uno + USB shield. The quadcopter is our main



concentration this semester. We kept working on the pre-build quadcopter and added the CC3D microcontroller to add the brain for the quadcopter. We tested the radio transmitter and receiver to make sure they both paired with the microcontroller. After several small incidents with testing quadcopter, we started to fly the quadcopter but the result was not that good. Last, we

integrated the robotic arm and quadcopter together, and we have our final prototype of our project.

Reference

[1] *Control your MeArm from Arduino*

<http://learn.mime.co.uk/docs/control-your-mearm-from-arduino/>

[2] *MeArm Servo Motor Arduino Wiring Schematics and Source Code*

<http://microbotlabs.com/armuno-arduino-schematic.html>

[3] *MEARM (II) – TESTING THE SERVOS*

<http://www.silvinopresa.com/how-to/arduino/mearm-ii-testing-the-servos/>

[4] *Xbox 360 Robotic Arm [Arduino]: Axiom Arm*

<http://www.instructables.com/id/AXIOM-ARM/>

APPENDIX

Equipment used:

---Fall semester

1. Arduino Uno
2. MeArm Robotic Arm
3. Parrot A.R. Drone 2.0
4. ESP8266 WiFi Module
5. Breadboard
6. Jump wires
7. Battery holder

---Spring semester

1. Arduino Uno

2. MeArm Robotic Arm
3. Self-built quadcopter
 - a. 4 propellers (2CW + 2CCW)
 - b. 4 Turnigy D2830-11 Brushless Outrunner
 - c. Frame
 - d. Turnigy 1500mAh 3S 20C Lipo Pack
4. CC3D flight controller
5. FS-CT6A Radio transmitter and receiver
6. Two long wooden sticks

Components (hardware, software, etc.):

1. Arduino IDE 1.6.13
2. Python 2.7.12
3. Pyserial 2.7
4. LibrePilot Ground Control Station (GCS)