Contents

[Introduction of “ThisArm” Firmware 2](#_Toc456632539)

[Hardware Requirement of this Firmware 2](#_Toc456632540)

[Optional Hardware 2](#_Toc456632541)

[Default pin assignment 2](#_Toc456632542)

[Source Code Files Explained 3](#_Toc456632543)

[Calibration 4](#_Toc456632544)

[Calibration of servo angle 5](#_Toc456632545)

[Create mapping between servo output and angle 7](#_Toc456632546)

[Calibration of Robot Arm Hardware: 9](#_Toc456632547)

[G-Code Control Guide: 10](#_Toc456632548)

[G-Code: 11](#_Toc456632549)

[M-Code: 11](#_Toc456632550)

[Explanation on transforming Cartesian coordinate to RC servo input waveforms (setpoint (double NewX, double NewY, double NewZ)). 12](#_Toc456632551)

[Explanation of Encoder control interface: 13](#_Toc456632552)

[Reference: 13](#_Toc456632553)

## Introduction of “ThisArm” Firmware

I put this hobby project open-source with a view to change the close source nature of Robot Arm firmware.

This Firmware aim to provide a higher level of automation and user interaction with hobby level robot arm that replicate kinematic of ABB IRB460. Existing example in the market/open source world include:

* MeArm
* EEZYbotArm
* EEZYbotArm MK2
* And a lot more in E-bay, Taobao……

Feature in this version include:

* Utilize the calibration data, convert human-understanding angle to RC-servo signal
* Invert kinematic to calculate desirable angle of Left, Right Servo from shoulder and elbow angle
* Interaction with robot arm with Mechanical Encoder with push button.
* Control of Robot arm by 4 potentialmeter
* G-code interpreter
  + Able to automate motion of arm by any G-code senders in the internet:
    - <https://chrome.google.com/webstore/detail/gcode-sender/ngncibnakmabjlfpadjagnbdjbhoelom>

## Hardware Requirement of this Firmware

* Computer with Arduino IDE set-up
* Arduino UNO (or any higher grade model like Arduino Mega Uno)
  + This is the brain
* Encoder with Push button
  + This is a good human computer interaction interface
  + This would make your life much easier
* RC-servo based Robot arm share the same kinematic of ABB IRB460

## Optional Hardware

* Arduino Sensor shield V5 (save you a lot of time when wiring the servo motors and others)
* Female Dupont Cable (save you from soldering)
* Linear Potentialmeter (4 of them, each for one dimension)
  + Beware **do not** use logarithmic potentiometer

## Default pin assignment

You can change the pin assignment as you want after you understand the code. The following is the default pin assignment:

|  |  |  |
| --- | --- | --- |
| Pin | Function of pin | Defined location |
| 10 | Base servo (a.k.a. middle servo) | Procedure “AngularServoInit()” in “AngularServo.h” |
| 9 | Left servo | Procedure “AngularServoInit()” in “AngularServo.h” |
| 6 | Right servo | Procedure “AngularServoInit()” in “AngularServo.h” |
| 5 | Claw servo | Procedure “AngularServoInit()” in “AngularServo.h” |
| 12 | Encoder Channel A (Encoder1\_A) | “ThisArm.ino” |
| 13 | Encoder Channel B (Encoder1\_B) | “ThisArm.ino” |
| 11 | Encoder Push button (Encoder1\_Click) | “ThisArm.ino” |
| A2 | Potentiometer for Base | Procedure “initJointsControl()” of “AnalogControl.h” |
| A1 | Potentiometer for Shoulder | Procedure “initJointsControl()” of “AnalogControl.h” |
| A0 | Potentiometer for Elbow | Procedure “initJointsControl()” of “AnalogControl.h” |
| A3 | Potentiometer for Claw | Procedure “initJointsControl()” of “AnalogControl.h” |

## Source Code Files Explained

This part aim to deliver a brief view on structure and usage of different files. Skip this part if you just want to set the arm up without more understanding.

|  |  |
| --- | --- |
| File Name | Function |
| AnalogControl.h | Contain Structure that convert potential meter input to corresponding angle |
| AngularServo.h | Contain Logic that convert input angle to Servo signal |
| ControlFunctions.h | Header for ControlFunctions.ino |
| ControlFunctions.ino | Contains function for different system stage that supposed to put into LOOP() for getting user’s input and deliver servo output |
| CoordinateControl.h | Contains coordinate management and invert kinematic logics. |
| DisplayFunctions.h | Header for DisplayFunctions.ino |
| DisplayFunctions.ino | Contains functions for different system stage that deliver debug/calibration assistance output via Serial interface |
| GCode.h | Header for GCode.ino |
| GCode.ino | Contains functions for taking G-code and convert it to arm output. |
| InitFunctions.h | Header for InitFunctions.ino |
| InitFunctions.ino | Contains Initialize functions for different machine stage |
| ThisArm.ino | The Main file of the This Arm software |
| StageContainer.h | Contains structures that help to manage different stages of the firmware. |
| Transformation.h | Contains functions for Linear linear interpolation |

## Calibration

This firmware is designed to fit in different robot arm hardware and servos motors. Since different servo output different angle on the same signal, Calibration is important for proper operation. (Even you are using the same MeArm as me)

The following are units used by calibration:

* Unit of length: mm
* Unit of Angle: Degree
* Unit of feed rate: mm per second

### Calibration of servo angle

The goal of this calibration is to ensure the servo motors deliver the exact angle we want.

The logic of calibration is based on linear interpolation, which utilize two known points to create a mapping between two dimensions.

Parameter of calibration is as follow:

|  |  |
| --- | --- |
| Parameter | Purpose |
| MIN | The minimum raw value the program would output |
| MAX | The maximum raw value the program would output |
| raw\_1 | The first raw value |
| ang\_1 | The first angle use for calibration |
| raw\_2 | The second raw value |
| ang\_2 | The second angle use for calibration |

#### Preparation:

As the first step, open the “StageContainer.h” source file and modify the first line to “**CurrentStage=0;**” from “**CurrentStage=7;**”

This would ensure the machine boot up with configuration that you can use the encoder to control the raw value of servo motor. (a.k.a. Micro second of “on” part of signal)

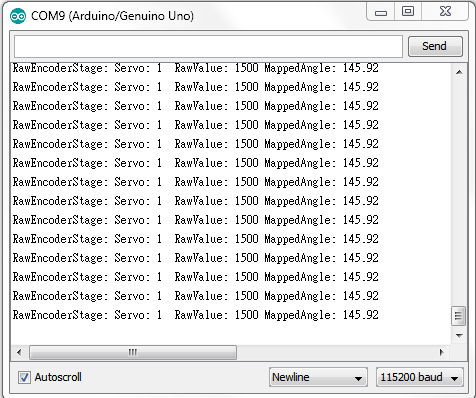
To utilize all available range of servo , go to “AngularServo.h” and do the following modification:

1. Change all MIN value to 600
2. Change all MAX value to 2400

Compile the source and upload to the Arduino.

After the compilation, the serial monitor should show something like below:

* “Raw Value” is the raw output of servo
* “Mapped Angle” is meaningless without calibration, can be ignored now.



Test and ensure everything works fine by turning the encoder and click the button.

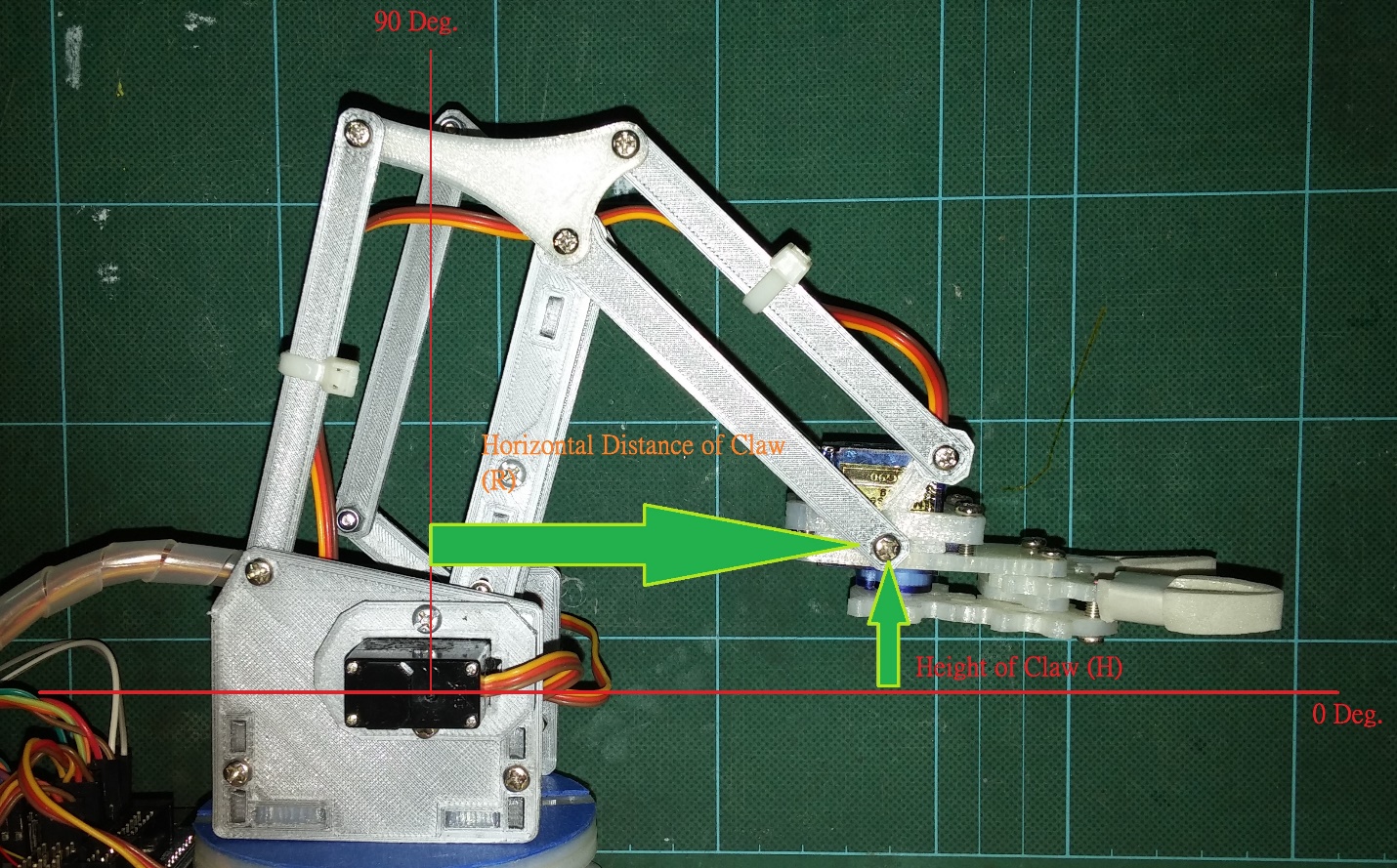
Supposed result is:

* When turning the encoder, one of the 4 servos change its angle
* When turning the encoder when the push button is pushed down, one of the 4 servos change its angle in a larger step
* When push the button without turning it, you would switch to control to other servo.
* Do not do any double click action yet, it is too early for this stage.

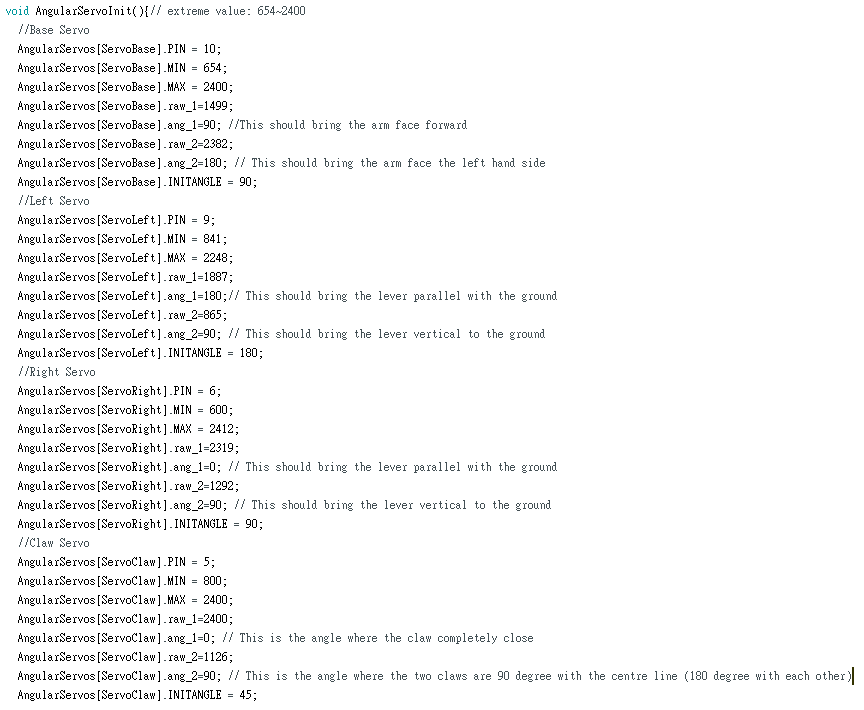
### Create mapping between servo output and angle

Prepare your screw drivers and prepare to disassembly a part of your arm, this will make your life easier.

The angle of the firmware is defined by the following picture:



The following task you need to do is use the encoder to move the servo to angles specified in the source code, and update the RAW value.



Compile and upload the new source code after that.

You can verify the result by change the stage of the software to “AnguleEncoderStage”, it can be done by double click the push button. (Two clicks within 300 milliseconds). In this stage, the encoder change the angle of each servo motor rather than raw value.

**Optional:**  Modify the MIN and MAX value for each servo so that the firmware would not output extreme value that break the physical structure of your arm.

### Calibration of Robot Arm Hardware:

The invert kinematic of robot arm is closely related to the hardware structure of arm, to ensure it give acceptable accuracy, dimension of arm should be input into the firmware.

The following is the dimensions you need to input (CoordinateControl.h ):

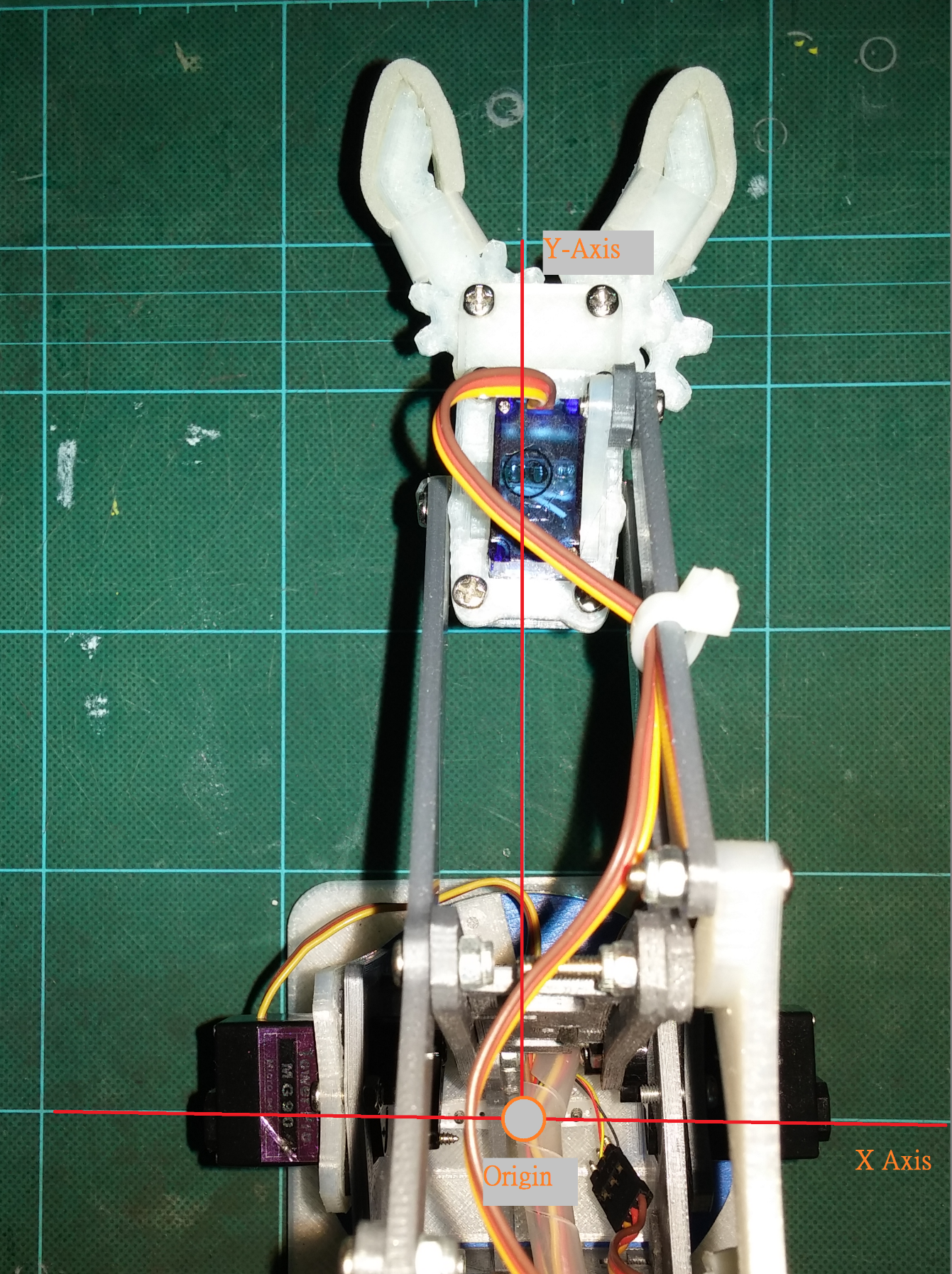
|  |  |
| --- | --- |
| Parameter | Description |
| BackArm\_mm | Length in mm of the back arm |
| ForeArm\_mm | Length in mm of the fore arm |
| d\_Min | Minimum distance between the claw and the base |
| d\_Max | Maximum distance between the claw and the base |
| R\_offset\_mm | Length in mm that the Left and Right servo place outward than rotation center. |

After input all those parameters, you should able to move the arm vertically and horizontally by the encoder in the “PolarEncoderControlStage”. (Use double Click to make the stage change)

The result should looks like this: <https://youtu.be/xu5G9OfJZHc>

## G-Code Control Guide:

This firmware use XYZ coordinates system to control the arm. Axis assignment was explained in the following picture:



The following G-code and M-code are available for automation and communicate with other system.

### G-Code:

|  |  |
| --- | --- |
| G0 [X(x-coordinate) Y(y-coordinate) Z(z-coordinate)] | Jump to a specific coordinate |
| G1 [X(x-coordinate) Y(y-coordinate) Z(z-coordinate) F(Feed Rate)] | Move to a specific coordinate with provided feed rate |
| G4 P(Millisecond to pause) | Pause for a specified time |
| G90 | Switch to absolute mode |
| G91 | Switch to relative mode |

### M-Code:

|  |  |
| --- | --- |
| M100 | Display help |
| M106 S(Claw Angle) | Set the angle of Claw |
| M114 | Report current position and feed rate |

## Explanation on transforming Cartesian coordinate to RC servo input waveforms (setpoint (double NewX, double NewY, double NewZ)).

Transform Cartesian coordinate to Polar Coordinate   
C2P (double x, double y, double z, double &R, double &H, double &A)

Transform desirable output angle of servo to raw value for Servo Library.  
ASet(AngularServo AServo, float angle)

\*\*This step need Calibration data

Transform Angle of different joint to desirable output angle of each servo.  
Angle of Right Servo=Shoulder  
Angle of Left Servo= Shoulder+Elbow  
Angle of Middle Servo=Angle of Base

Transform Polar Coordinate to angle of different logical angle.  
InverseKinematicsTransform (double R, double H, double &Shoulder, double &Elbow)

\*\*This step need Calibration data

## Explanation of Encoder control interface:

Available action:

* Turn
  + Increase/Decrease the parameter selected.
* Pushdown and Turn
  + Increase/Decrease the parameter selected in a larger step
* Click
  + Switch selected item (Except G-code and Analog Control Stage)
* Double Click
  + Button was clicked twice within “DoubleClickInteral\_ms” (10ms by default)
  + Execute DoubleClick Routine (For change of machine Stage)

## Reference

* GCode CNC Demo by MarginallyClever
  + <https://github.com/MarginallyClever/GcodeCNCDemo>