UFACTORY

# UARM Swift Pro

# Developer Guide

V1.0.3

Sep. 2017

# Contents

# Safety Instructions

1. Please don’t put your hands between the arms when uArm is moving.
2. Please use the official power supply for safety reasons.
3. Please clear a space for uArm, in case of knocking down anything.

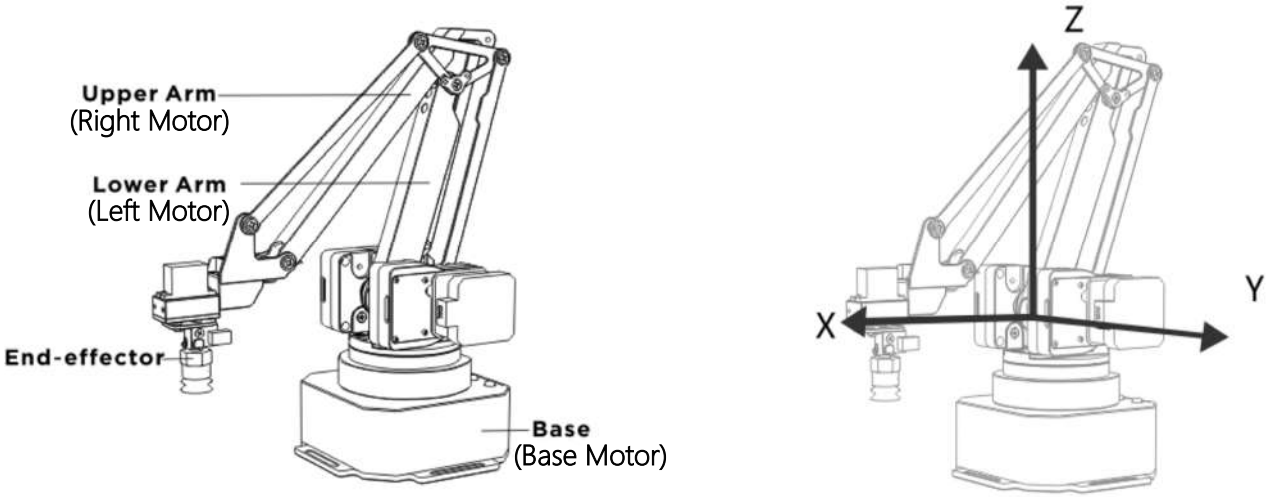
# General Information

General information for the robot arm, and it's good to know before developing.

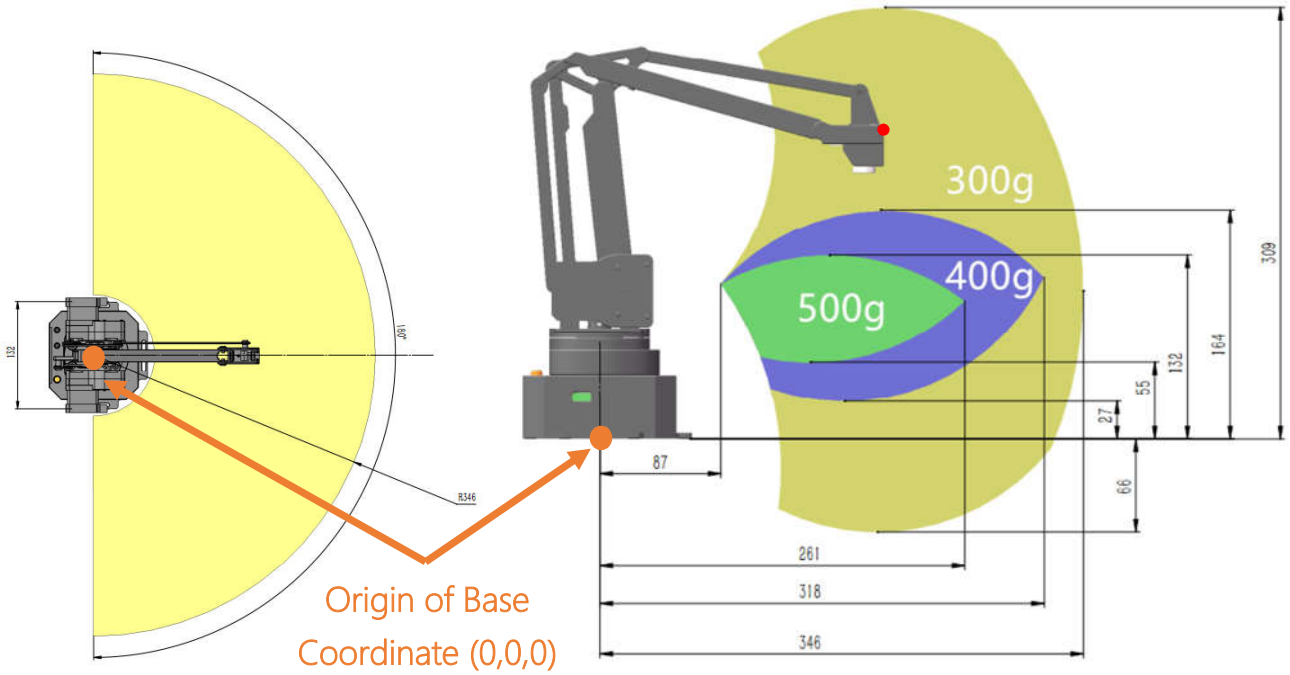
## 1.Source file

- Source code of Firmware for Swift Pro:
  https://github.com/uArm-Developer/SwiftProForArduino
- Source code of ROS for Swift Pro:
  https://github.com/uArm-Developer/SwiftproForROS
- Python library for Swift Pro:
  https://github.com/uArm-Developer/pyuf
- OpenMV example for tracking:
  https://github.com/uArm-Developer/OpenMV-Examples
- To be continued...(Arduino, C++, Raspberry Pi)

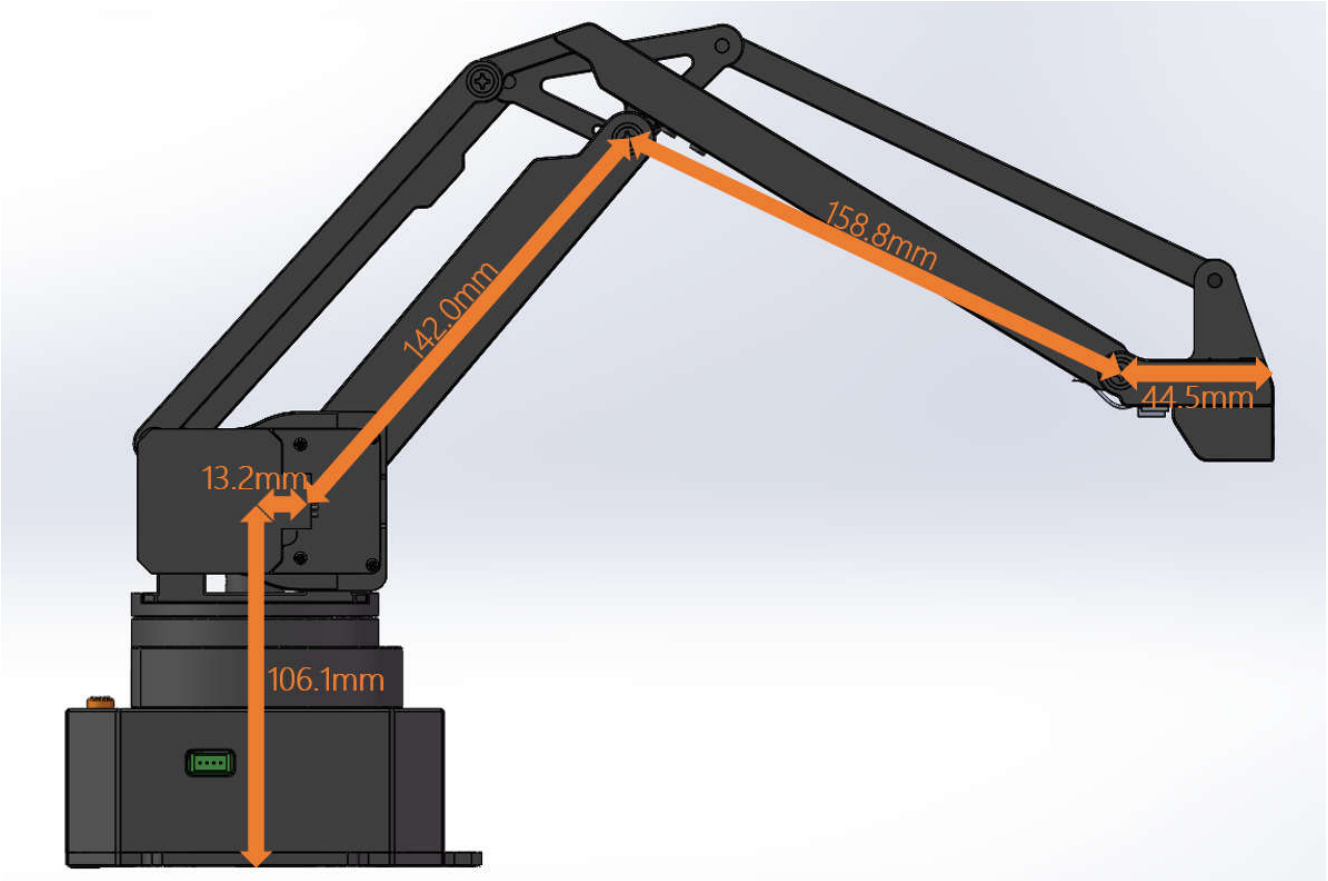
## 2.Reference Frame



# 3.Coordinate

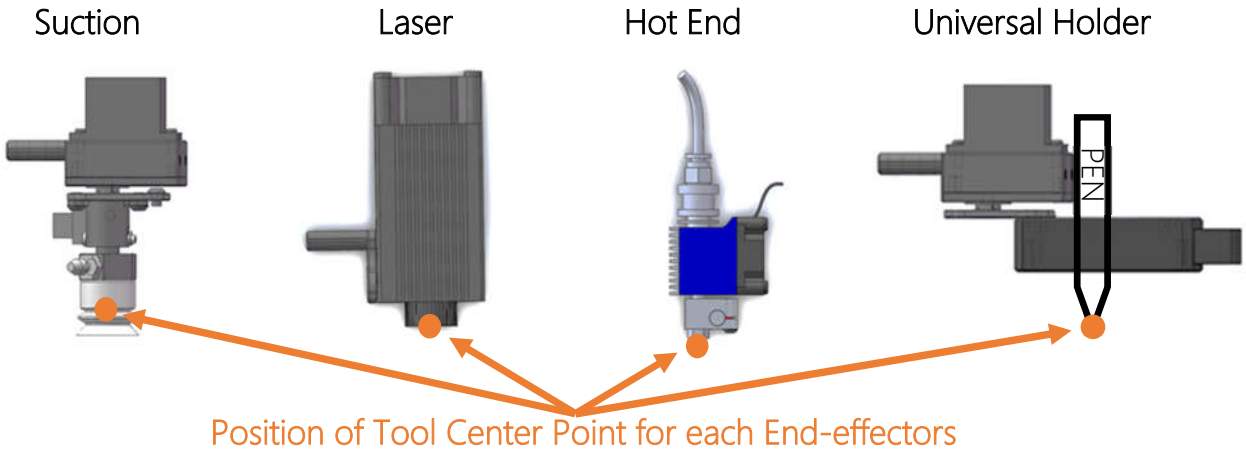



( The picture in the right also shows the dynamic payload range of uArm.
Test condition: G2202 F1000; Red point is the Tool Center Point. )




Detail size of each arm and base.

The origin of base coordinate is in the center of the base. But the tool center point is different for different end-effectors. And we also offer the different commands for different usages.




Position of Tool Center Point for each End-effectors

Currently we offer 4 kinds of mode:

M2400 S0 : Normal mode (end-effector tools: suction)

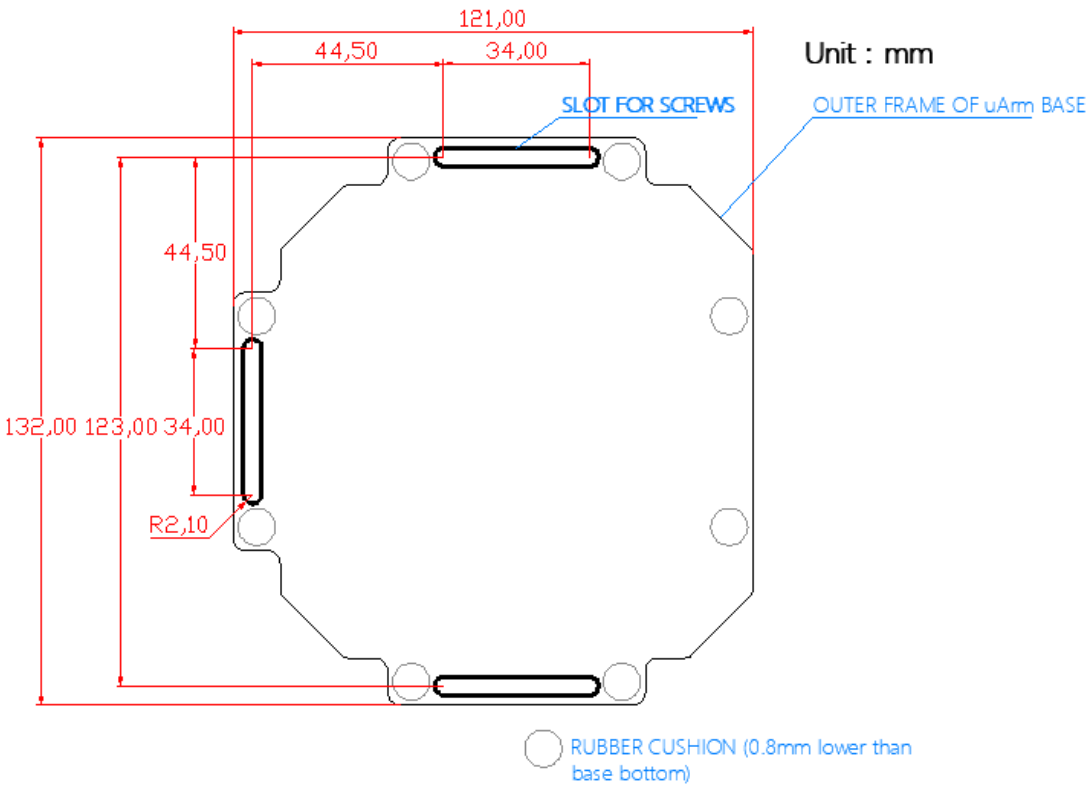
M2400 S1 : Laser mode (end-effector tools: laser)

M2400 S2 : 3D printing mode (end-effector tools: Hot End)

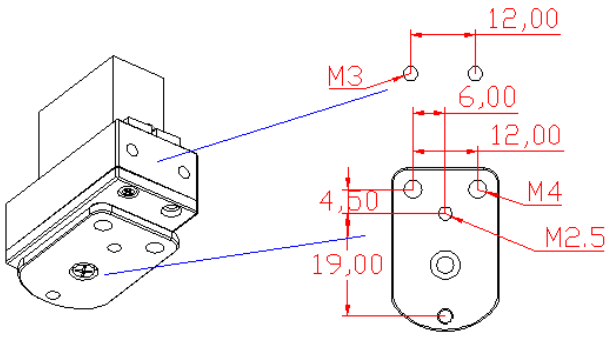
M2400 S3 : Universal holder mode (end-effector tools: universal holder)

For the gripper, there is no special mode since gripper has the fingers and can rotate horizontally.

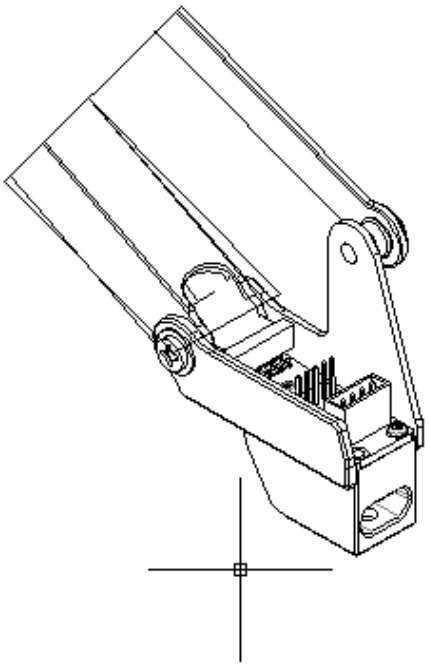
## 4.Mounting

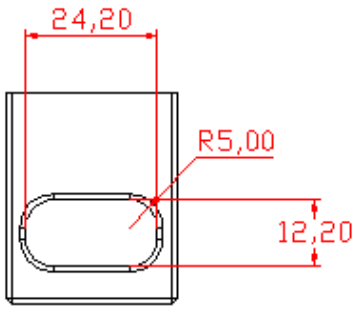


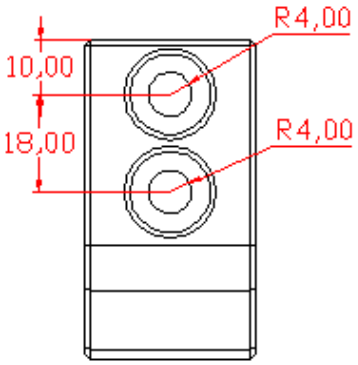
Unit : mm



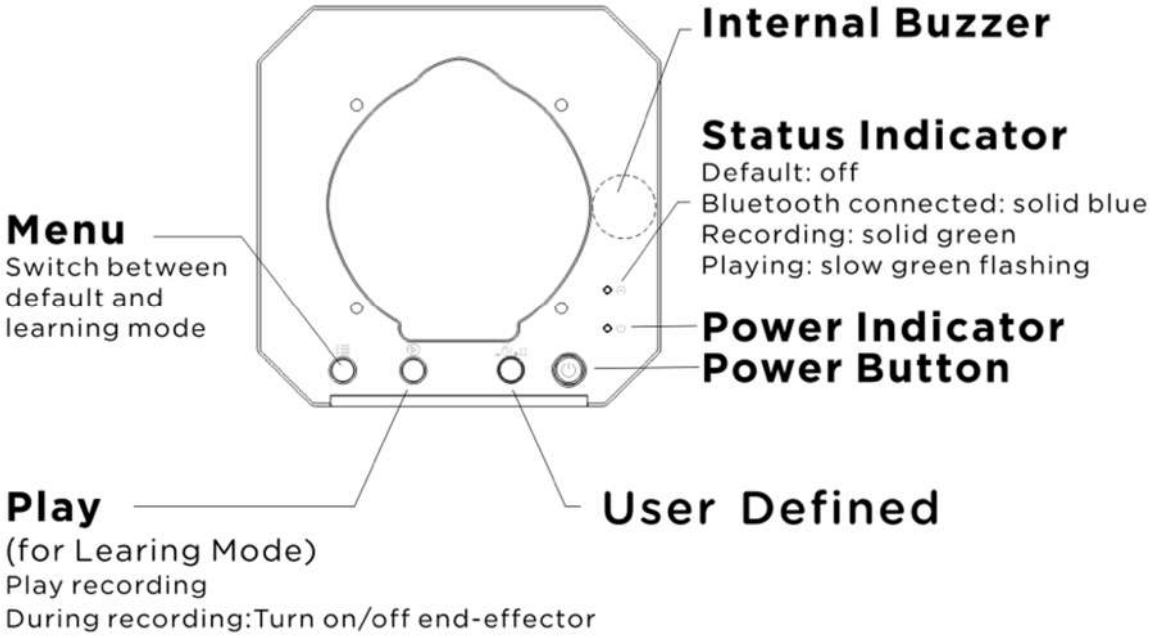

Unit : mm



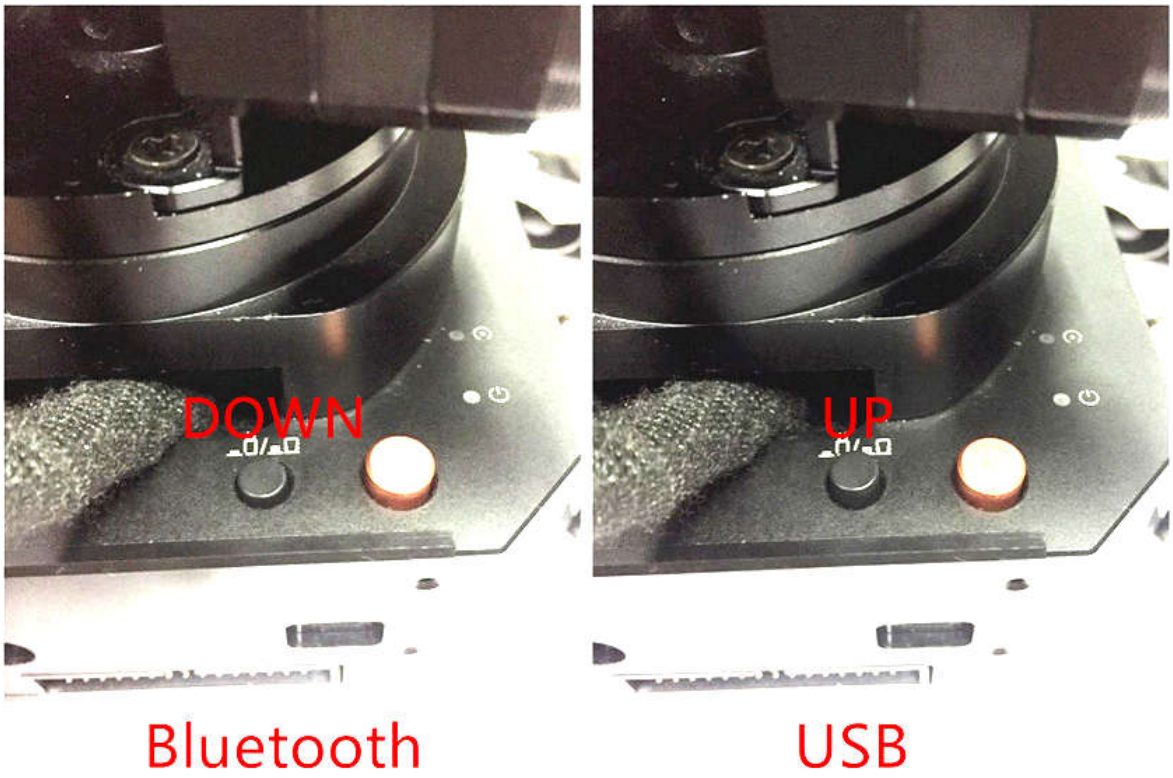


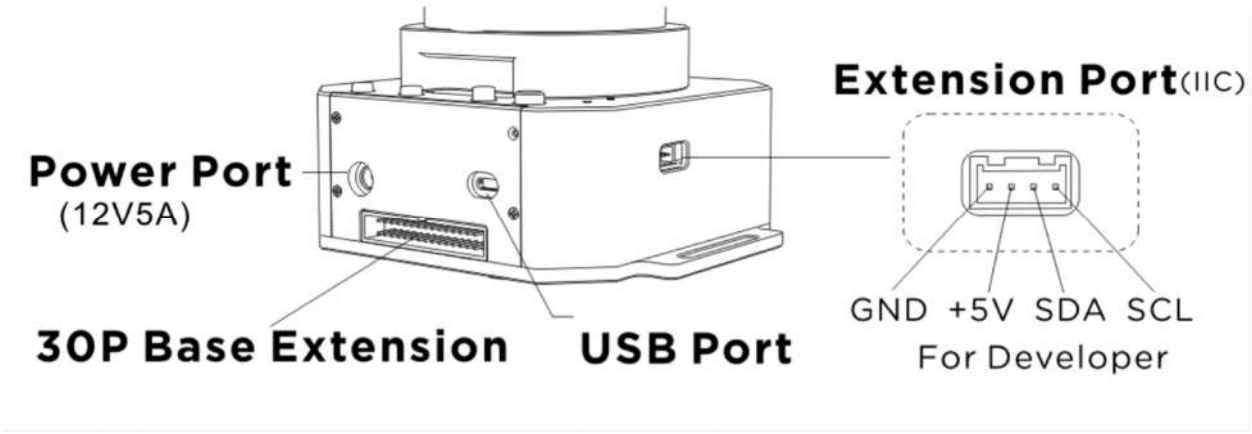
# 5. Buttons & Indicator Lights

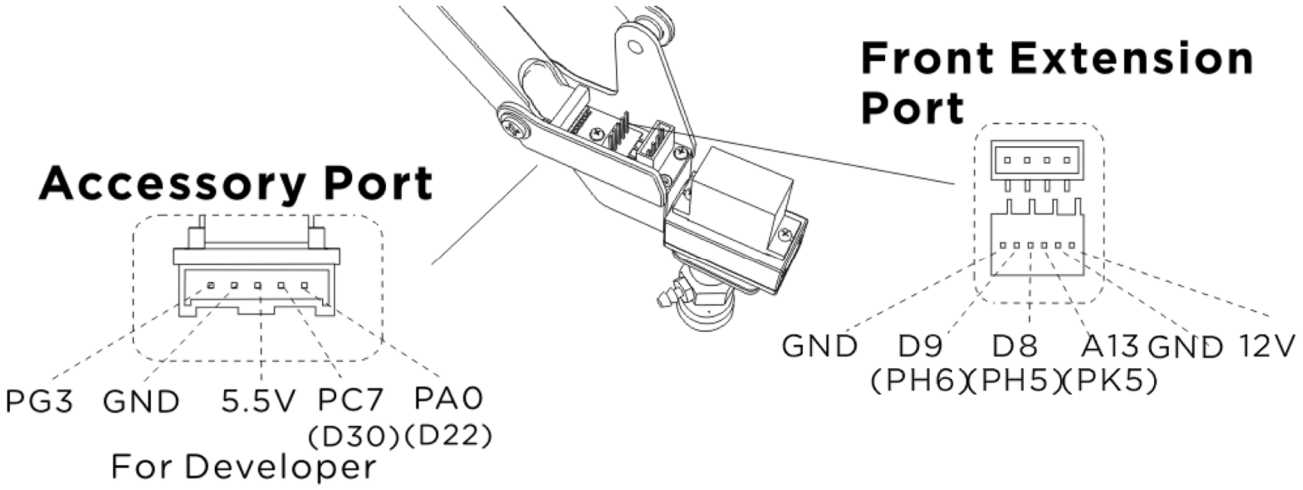



Caution: By default, the user defined button is for switching between Bluetooth and USB mode. Please ensure the button is UP while communicating with uArm via USB.

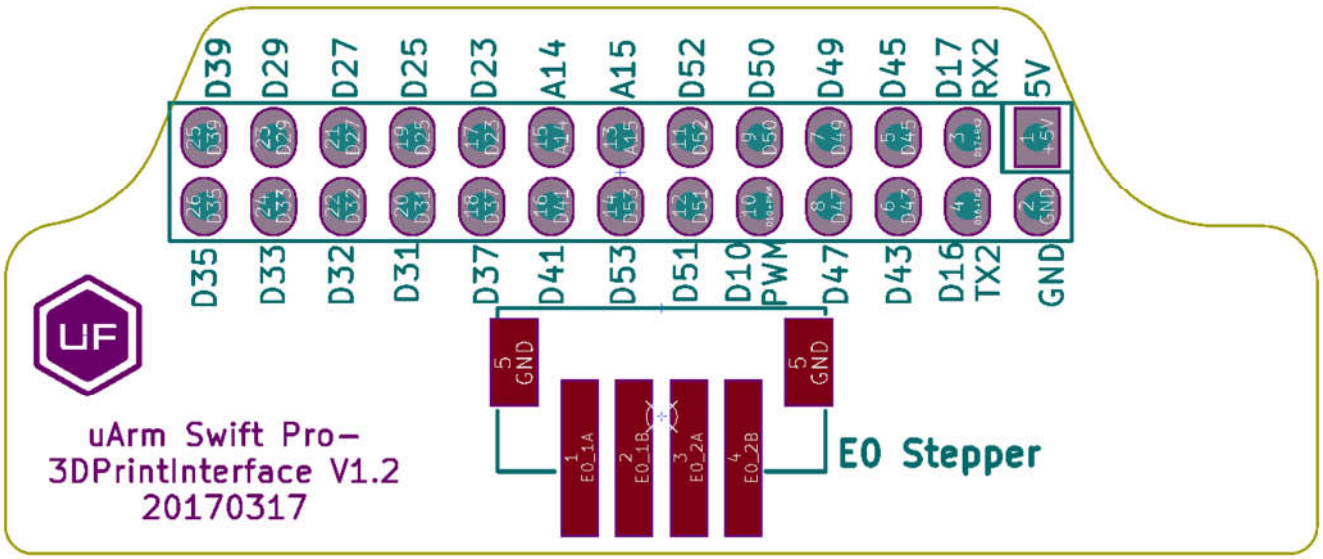


# 6.Extension Description




## Details of 30P Base Extension



# Specifications

| Specifications | | | |
|---|---|---|---|
| Weight | 2.2kg | | |
| Degrees of Freedom | 4 | | |
| Repeatability | 0.2mm | | |
| Max. Payload | 500g | | |
| Working Range | 50mm ~ 320mm | | |
| Max. Speed | 100mm/s | | |
| Connector | Micro USB | | |
| Wireless | Bluetooth 4.0 | | |
| Input Voltage | DC 12V | | |
| Power Adapter | Input:100 ~ 240V 50/60Hz; Output: 12V5A 60W | | |
| Operation Temperature & Humidity | 0℃-35℃ 30%RH-80%RH noncondensing | | |
| Storage Temperature & Humidity | -20℃-60℃ 30%RH-80%RH noncondensing | | |
| Hardware | | | |
| Joint Type | Customized Gearbox + Stepper | | |
| Position Feedback | 12 bit Encoder | | |
| Reducer | Customized ultra-thin Gearbox | | |
| Dimension(L*W*H) | 150mm*140mm*281mm | | |
| Mother Board | Arduino MEGA 2560 | | |
| Material | Aluminum | | |
| Baud Rate | 115200bps | | |
| Extendable I/O Interface | I/O *27 , IIC *1 , 5V*1 , 12V*1 , Stepper*1 | | |
| Software | | | |
| PC Control | uArm Studio | | |
| App Control | uArm Play | | |
| For Developer | Python/Arduino/ROS | | |
| Feature | Open Source | | |
| Joint Speed & Torque | | | |
| | Speed | Lifetime | Torque |
| Base Motor | 40°/s | >3000h | 12kg·cm |
| Left Motor | 40°/s | >3000h | 12kg·cm |
| Right Motor | 40°/s | >3000h | 12kg·cm |
| End-effector Motor | 60°/s | 500h | 2kg·cm |

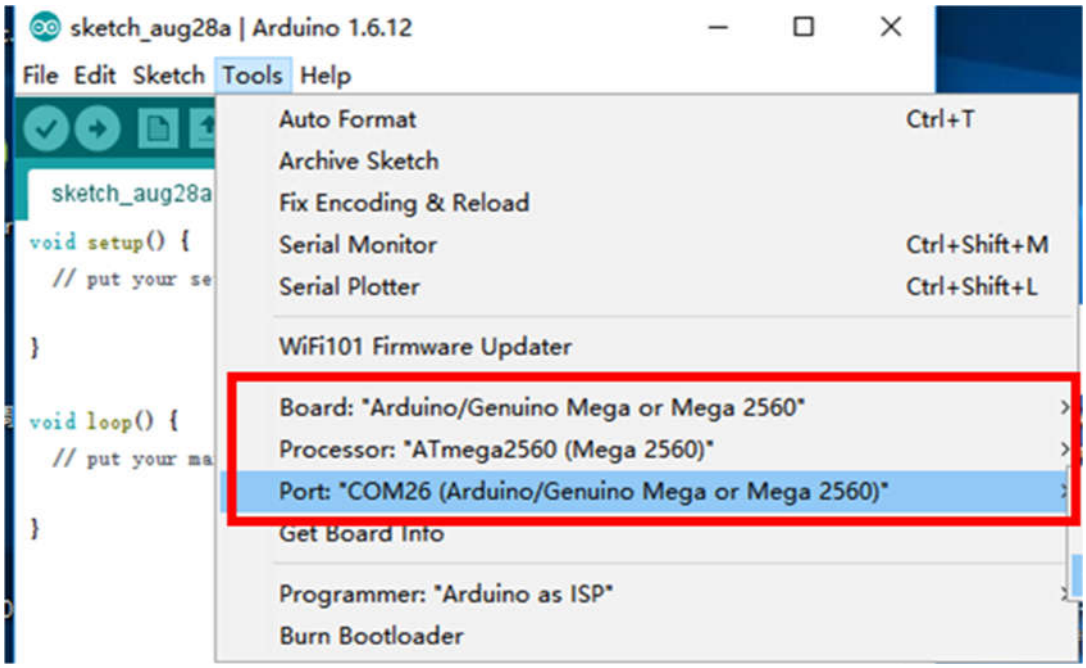
| Accessories | | |
|---|---|---|
| Pump | Suction Diameter | 5mm ~ 10mm |
| | Max. Pressure | 33kPa |
| | Max. Lifting Weight | 1000g |
| | Feature | With feedback |
| Universal Holder | Weight | 36g |
| | Dimension(L*W*H) | 62mm*25mm*15mm |
| | Material | Aluminum |
| | Holder Diameter | 14mm |
| Gripper | Weight | 58g |
| | Dimension(L*W*H) | 92mm*50mm*18mm |
| | Material | Aluminum |
| | Max. Force | 750~800g |
| | Max. Size of Object | 40mm |
| | Max. Speed | 20mm/s |
| | Drive Mode | Electrically-driven |
| | Working Voltage/Current | 6V/300mA |
| OpenMV Camera | Focal Length | 2.8mm |
| | FOV | 115° |
| | F-number | f2.0 |
| | Programmable Method | Micro Python |
| | Weight | 16g |
| | Dimension(L*W*H) | 45mm*36mm*30mm |
| 3D Printing Kit | Type | E3D v6 |
| | Consumption | 35W |
| | Nozzle Diameter | 0.4mm |
| | Max Temp | 270 ℃ |
| | Material | PLA |
| | Max. Printing Speed | 20mm/s |
| | File Format | .gcode |
| | Printing Size(L*W*H) | 10mm*10mm*10mm |
| Laser Engraving Kit | Laser Power | 500mW |
| | Working Voltage/Current | 12V/5A |
| | Wave Length | 405nm |
| | Weight | 140g |
| | Dimension(L*W*H) | 55mm*33mm*88mm |
| | Materials to Engrave | Wood, Plastic, Leather, Feather, Paper, etc. |

# Application Information

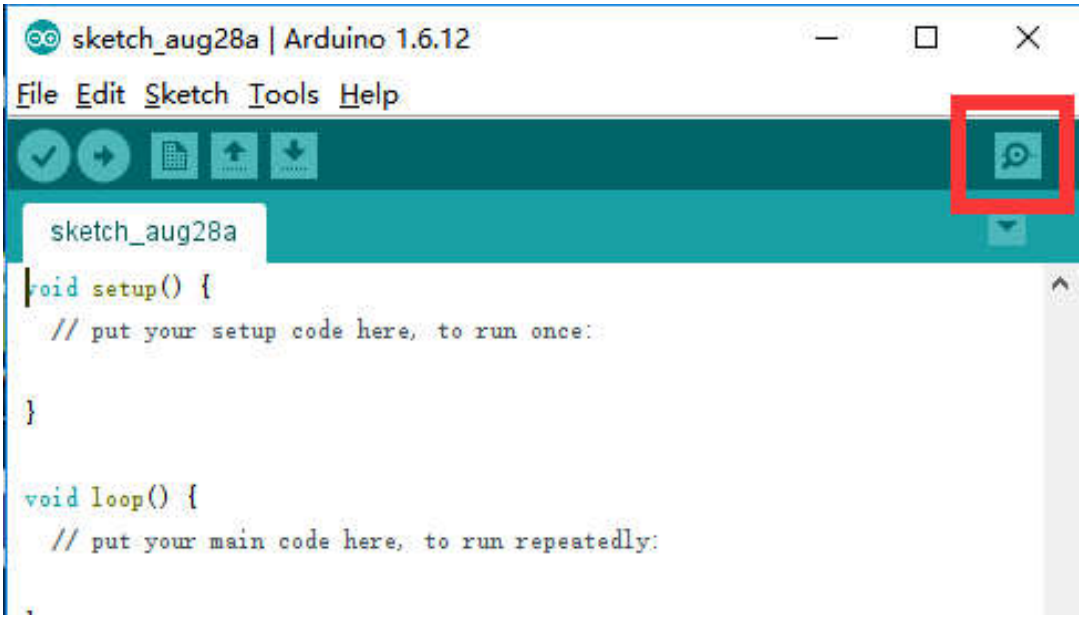
We would introduce several ways to play with the robot arm in different platform.

## 1.Send Command over USB Cable

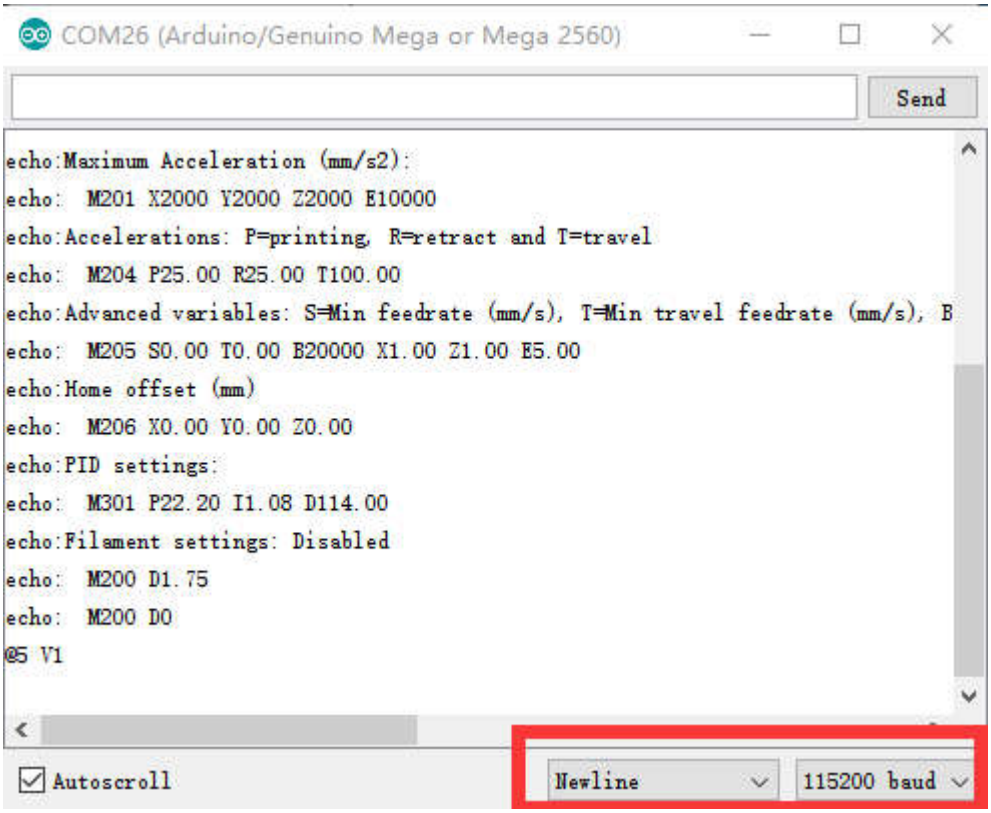
Power on the uArm and open the Arduino IDE. And setting the board like the picture below. Please make sure the port you are choosing is the correct port of uArm.



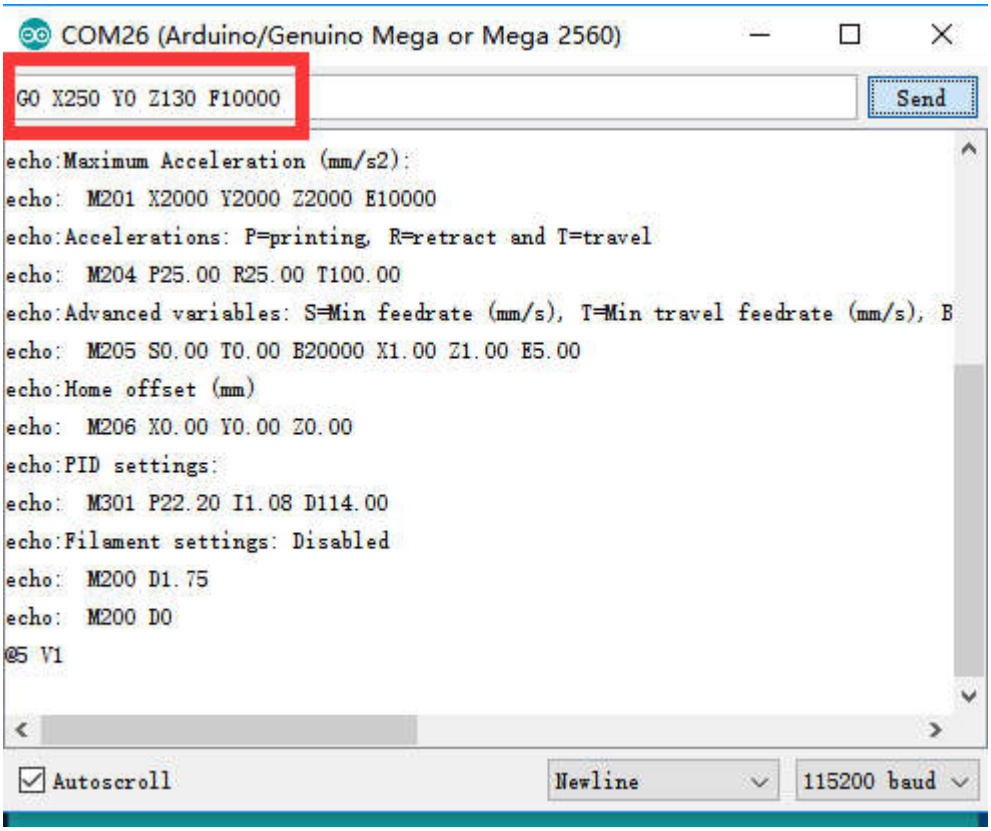

Open the serial monitor in the right side of Arduino IDE. After clicking, and you could hear a beep which means the uArm is connected.



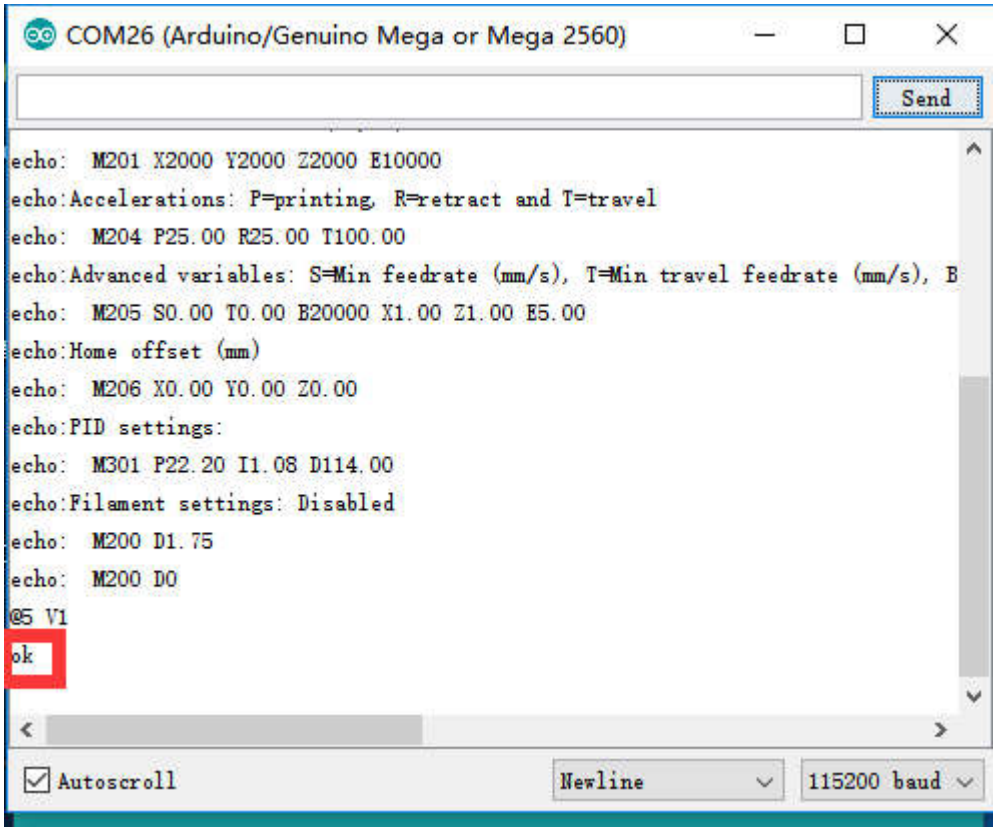
Set the parameter of serial monitor in the right bottom. If the setting is correct, you would receive the detail information from uArm like the picture below.




Now, you are able to send the command to the uArm. Let's send "G0 X250 Y0 Z130 F10000".



If uArm finishes the movement, it replies “ok”.




Please check the chapter of Protocol (Page 20) in this guide to test more commands.

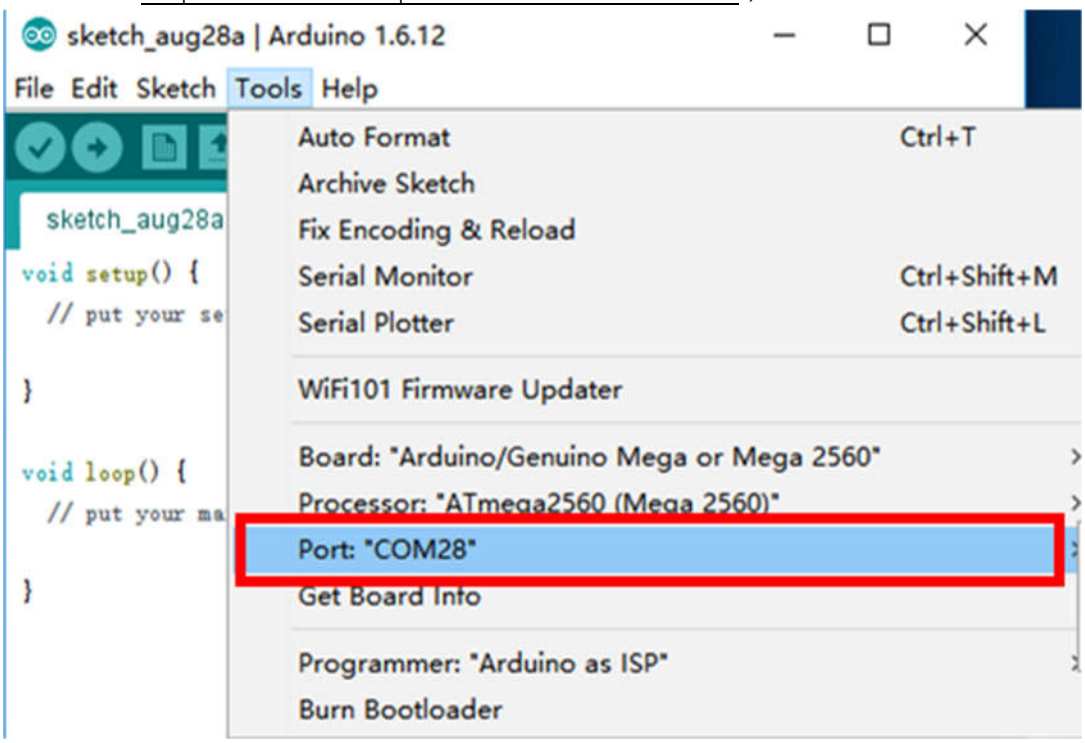
## 2.Send Command over Bluetooth

Plug in the Bluetooth stick. And press down the button beside the power button.

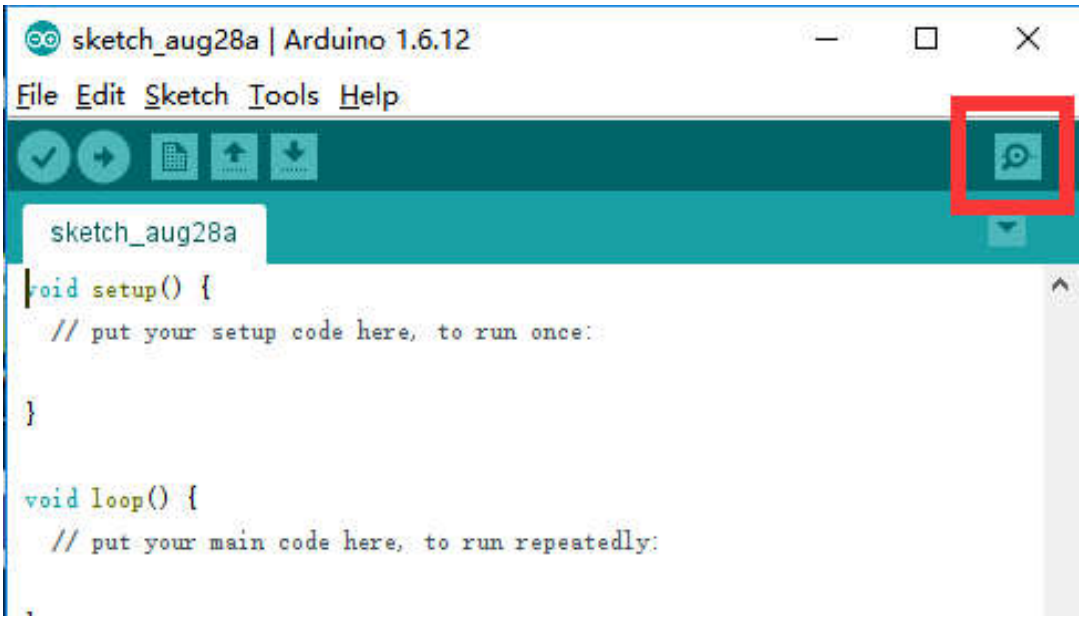

Power on the uArm. When the Bluetooth stick is searching, the blue indicator keeps blink until the wireless connection is built up between stick and uArm. And the blue indicators in both stick and uArm become solid.

Open the Arduino IDE. And setting the COM port like the picture below. Please make sure the port you are choosing is the correct port of Bluetooth stick.

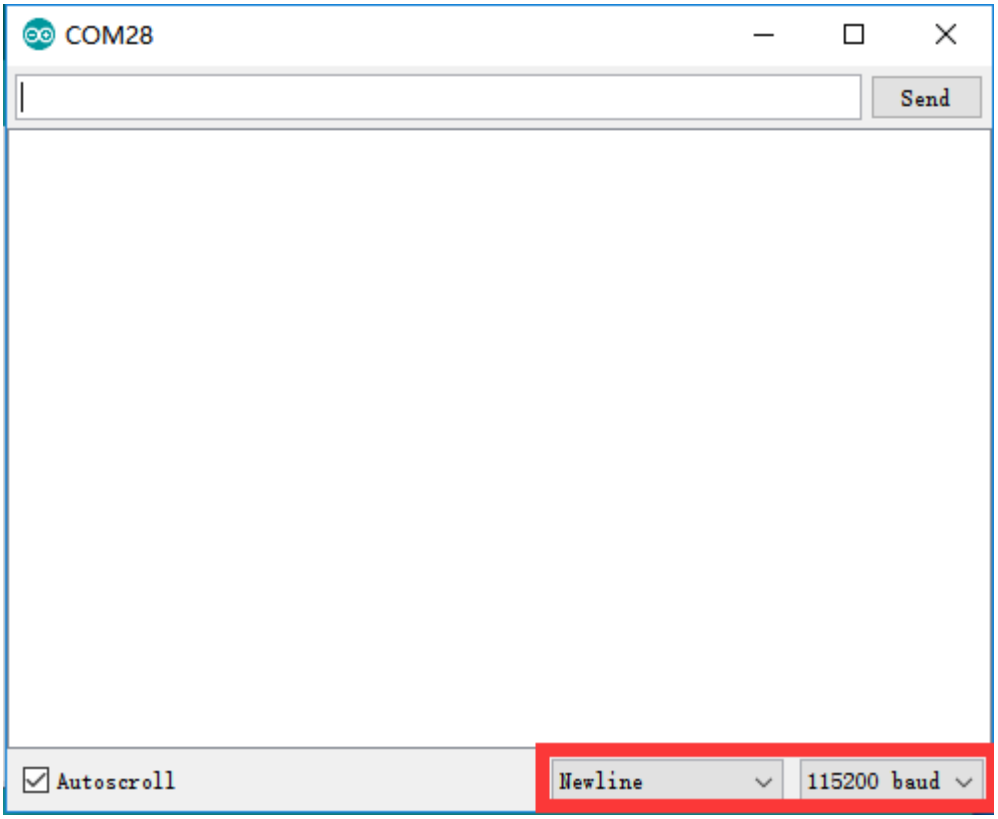
(Driver of stick: http://www.ftdichip.com/Drivers/VCP.htm )



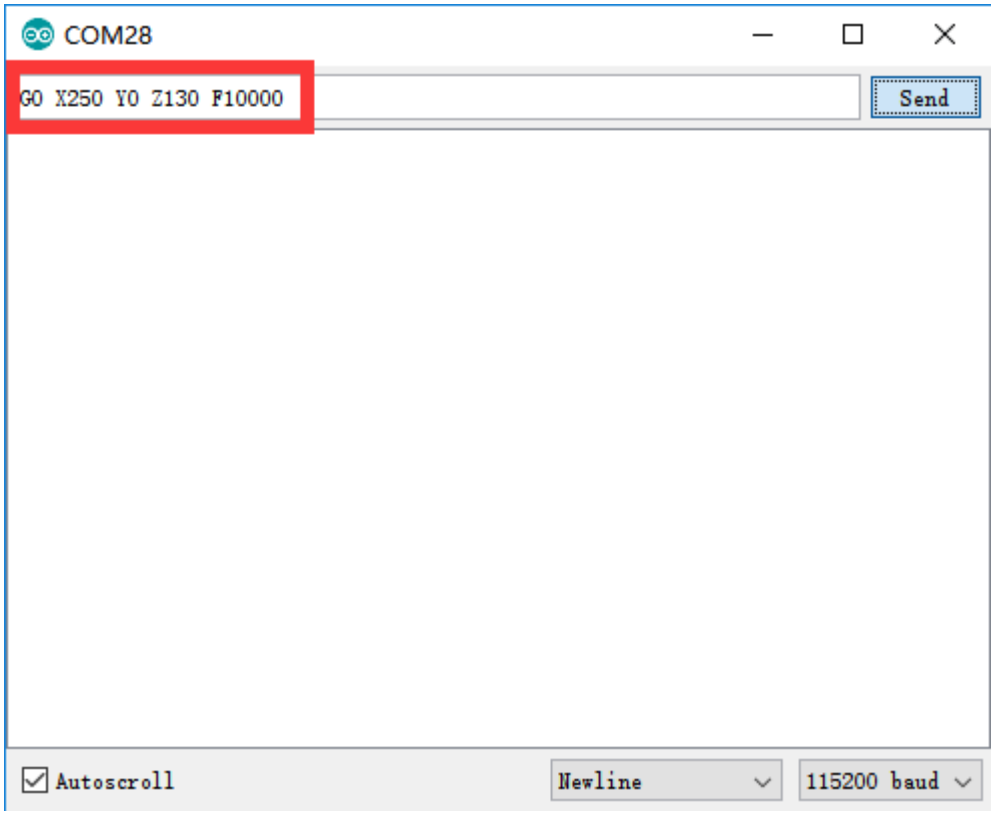

Open the serial monitor in the right side of Arduino IDE. After clicking, and you could hear a beep which means the uArm is connected.



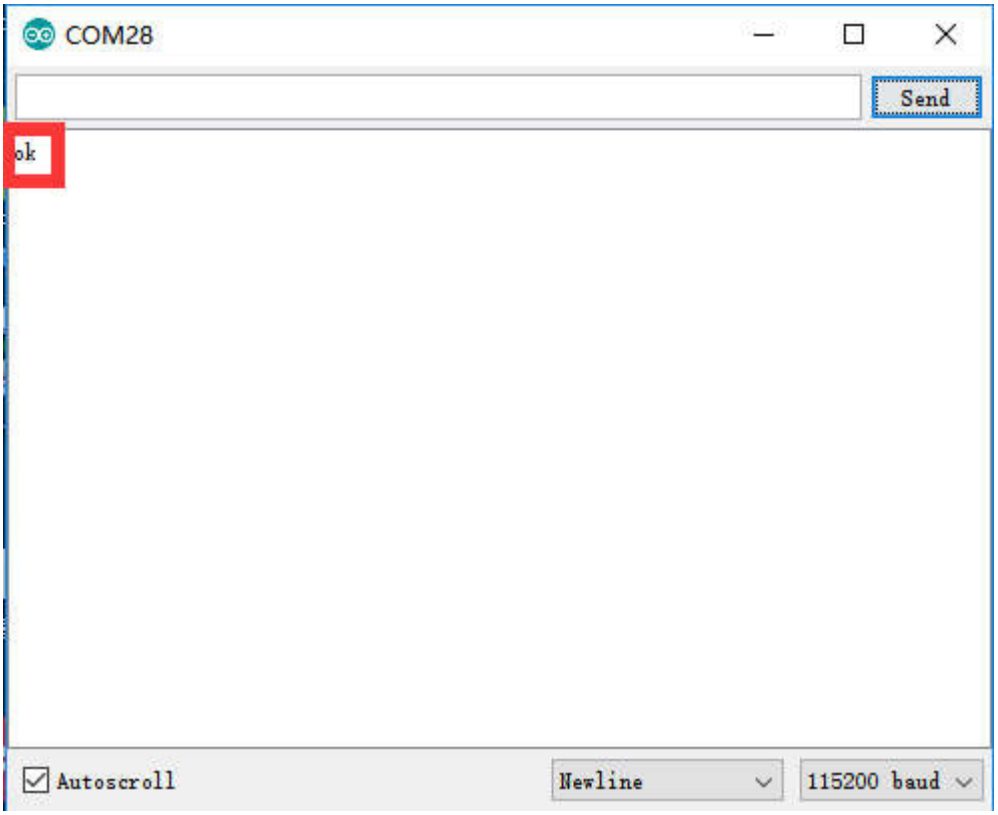
Set the parameter of serial monitor in the right bottom like the picture below.




Now, you are able to send the command to the uArm. Let's send "G0 X250 Y0 Z130 F10000".



If uArm finishes the movement, it replies “ok”.



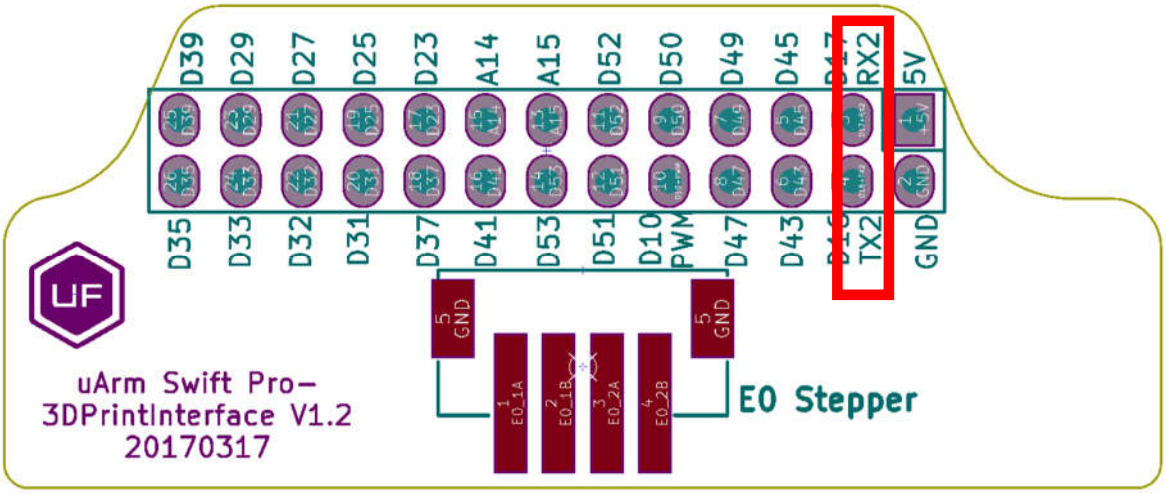

Please check the chapter of Protocol (Page 20) in this guide to test more commands.

## 3.The 2nd UART

Sometimes the 2nd UART is important for our project, for example you want another Arduino to communicate with uArm.

During the design, we have had it in mind. There is the 2nd UART in the 30P base extension.

All the pins of extension board are connected with the Arduino MEGA 2560 directly so it’s TTL level. And voltage above 5V might burn the IO out.



So wiring the UART with the jump wire and also the GND. Then the hardware set up is finished. Then we have to change the main communication port from USB to the 2nd UART port, since the code only supports one port to deal with the command.

Sending "#0 M2500" command over USB cable to switch the port, and there are several point you should know:

1. The port will be switched immediately (both ports receive the reply "ok"), and the USB port can not be used as the communication port any more, only 2nd UART port would work for sending Gcode.
2. There is no way to switch port any more, the only way to use USB port is reset the system by power button.

If it's not convenient for your project, please try to modify the source code following the steps below:

1. Download the Arduino source file in Github.
2. Find the file named **uArmSerial.cpp** and modify the code in line 16 from **_serial=&serial;** to **_serial=&serial2**; .
3. Find the file named **uArmService.cpp** and modify the code in line 693 from **commSerial.setSerialPort(&Serial);** to **commSerial.setSerialPort(&Serial2);** .
4. Rebuild the files and download the code to uArm.

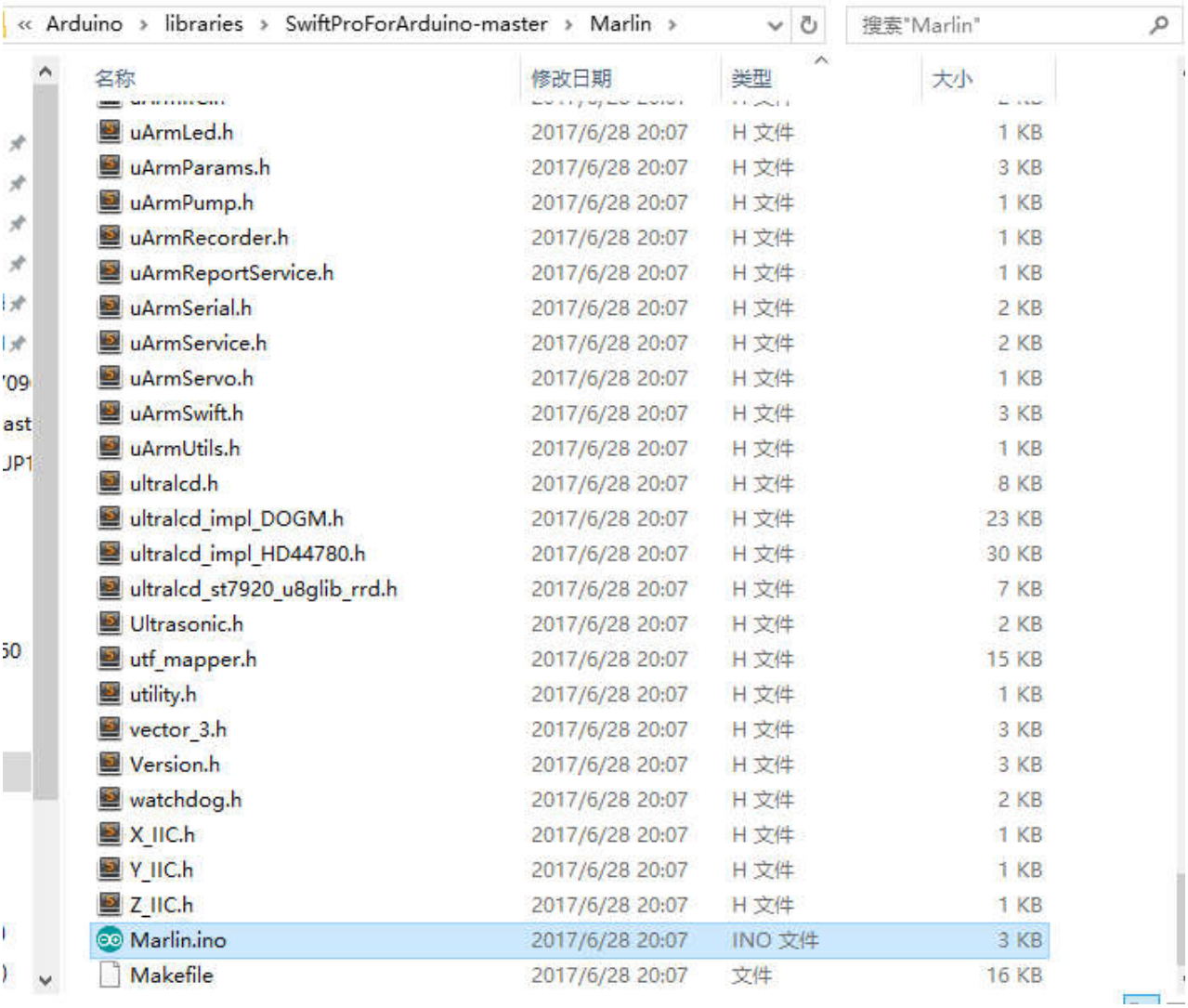
# 4.Arduino

The main code is written by Arduino IDE. Please check the link below:
https://github.com/uArm-Developer/SwiftProForArduino

### How to compile and upload the file ?

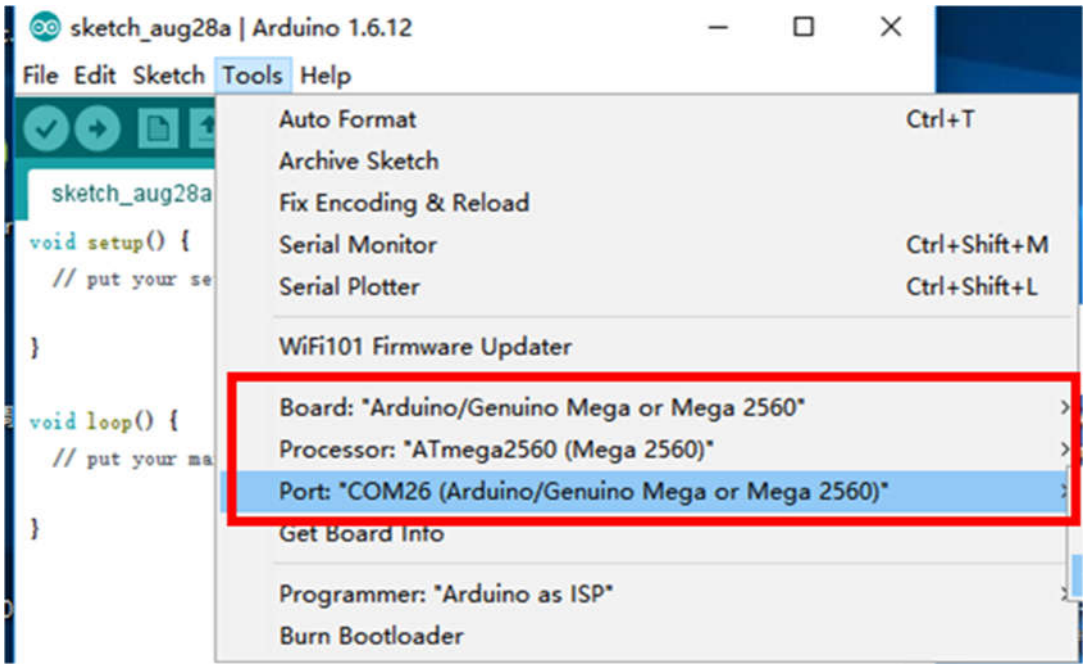
1. Download the code and extract it to your Arduino libraries folder (normally it's in C:/Users/ufactory/documents/Arduino/libraries/)

2. Find the file named Marlin.ino in the Marlin folder and open it

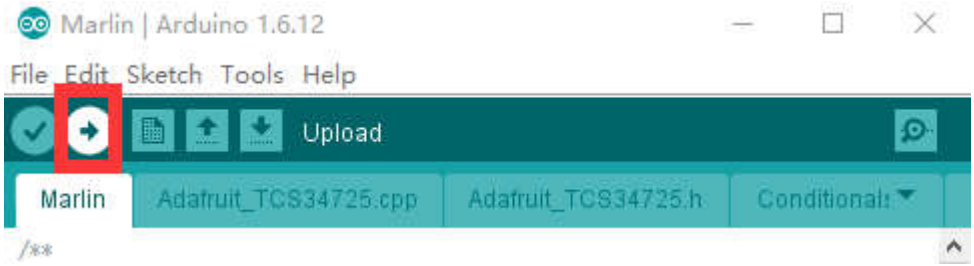



3. Plug in the USB cable and power port then turn on the power button

4. Select the correct port and type of Arduino board like the picture below




5. Click the upload button to finish the uploading




# 5.ROS & Python

Currently we released the library of Python and ROS. For more information please check the link below.

Source code of ROS for Swift Pro:

https://github.com/uArm-Developer/SwiftproForROS

Python library for Swift Pro:

https://github.com/uArm-Developer/pyuf

# 6.OpenMV Demo

And also the demo of OpenMV:

https://github.com/uArm-Developer/OpenMV-Examples

You could find the details steps in quick start guide.

# 6.Recover from the Wrong Code

Sometimes you might want to go back to the official firmware and it's too complicated to download the Arduino source code and download it. Or you flashed bad code to the uArm and you can't even run it. Please try the offline flash tool here : https://drive.google.com/drive/u/0/folders/0B-L-tCvknXU9dDhfSGJwT1JDY1U

# Protocol

## 1.Introduction

- *u*Arm Gcode is an important part of the uArm software.
- Based on the standard gCode protocol, we add a new protocol head in front of the gCode so that it can be more easily to use and debug.
- What's more, it is designed to be compatible with the standard gCode. (We offer the code of decode the standard gCode)

## 2.Example

- Sending command from PC

"#25 G0 X180 Y0 Z150 F5000"

//move to [180,0,150] with the speed 5000mm/min

- Reply from uArm "$25 ok"

## 3.Commands(TBD)·

Command can be divided into two parts:

Command with underline: it's the new added protocol head.

- The command from PC starts with '#', while the command from uArm starts with'$'.
- And the data following the symbol decided by the PC, and the reply from

the uArm should have the same data which indicates it finish the command. (In the example above, PC sends the command with '#25' and uArm replies the command with '$25')

Command without the underline: it's the standard gCode.

Caution

1.There should be blank space between each parameter;
2.The letters in the command should be capitalized;

| GCode Command (v1.2) | Description | Feedback |
|---|---|---|
| **1.** #***n*** is used for the debug, if you don't want to use it please remove it directly. (For Example: G2202 N***0*** V***90***\n)<br>2. '\n' is the symbol of line feed. | | |
| Moving Command (parameters are in underline) | | |
| #***n*** G0 X***100*** Y***100*** Z***100*** F***1000***\n | Move to XYZ(mm), F is speed(mm/min) | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| #***n*** G1 X***100*** Y***100*** Z***100*** F***1000***\n | **After entering the laser mode (M2400 S1)**, command **G1** means laser on, **G0** means off. | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| #***n*** G2004 P***1000***\n | Delay microsecond | $***n*** ok \n |
| #***n*** G2201 S***100*** R***90*** H***80*** F***1000***\n | Polar coordinates, S is stretch(mm), R is rotation(degree),H is height(mm), F is speed(mm/min) | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| #***n*** G2202 N***0*** V***90***\n | Move the motor to the position ,N is ID of joints(0~3),V is angle(0~180) | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| #***n*** G2203\n | Stop moving | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| #***n*** G2204 X***10*** Y***10*** Z***10*** F***1000***\n | Relative displacement | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| #***n*** G2205 S***10*** R***10*** H***10*** F***1000***\n | Polar coordinates for relative displacement | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| Setting Command (parameters are in underline) | | |
| #***n*** M17\n | Attach all the joint motors | $***n*** ok \n |
| #***n*** M2019\n | Detach all the joint motors | $***n*** ok \n |
| #***n*** M2120 V***0.2***\n | Set time cycle of feedback, return Cartesian coordinates, V is time(seconds) | @3 X***154.71*** Y***194.91*** Z***10.21***\n |
| #***n*** M2200\n | Check if uArm is moving | $***n*** ok V***1***\n (1 moving,0 stop) |
| #***n*** M2201 N***0***\n | attach motor, N is ID of joints(0~3) | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| #***n*** M2202 N***0***\n | Detach motor, N is ID of joints(0~3) | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |

| | | |
|---|---|---|
| #***n*** M2203 N***0***\n | Check if the motor is attached, N is ID of joints(0~3) | $***n*** ok V***1***\n (1 attached,0 detached) |
| #***n*** M2210 F***1000*** T***200***\n | buzzer,F is frequency, T is time (ms) | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| #***n*** M2211 N***0*** A***200*** T***1***\n | Read EEPROM N(0~2,0 is internal EEPROM,1 is USR_E2PROM, 2 is SYS_E2PROM), A is address, T is type (1 char,2 int,4 float) | $***n*** ok V***10***\n |
| #***n*** M2212 N***0*** A***200*** T***1*** V***10***\n | Write EEPROM N(0~2,0 is internal EEPROM,1 is USR_E2PROM, 2 is SYS_E2PROM), A is address, T is type (1 char,2 int,4 float)V is the input data | $***n*** ok\n |
| #***n*** M2213 V***0***\n | Default function of base buttons (0 false, 1 true) | $***n*** ok\n |
| #***n*** M2220 X***100*** Y***100*** Z***100***\n | Convert coordinates to angle of joints | $***n*** ok B***50*** L***50*** R***50***\n (B joint 0,L joint 1,R joints 2, 0~180) |
| #***n*** M2221 B***0*** L***50*** R***50***\n | Convert angle of joints to coordinates | $***n*** ok X***100*** Y***100*** Z***100***\n |
| #***n*** M2222 X***100*** Y***100*** Z***100*** P***0***\n | Check if it can reach,P1 polar, P0 Cartesian coordinates | $***n*** ok V***1***\n (1 reachable, 0 unreachable) |
| #***n*** M2231 V***1***\n | pump V1 working, V0 stop | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| #***n*** M2232 V***1***\n | gripper V1 close, V0 open | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| #***n*** M2234 V***1***\n | Enable/disable Bluetooth (1:enable, 0:disable) | $***n*** ok\n |
| #***n*** M2240 N***1*** V***1***\n | Set the digital IO output | $***n*** ok \n or $***n*** E***x*** \n (refer to Err output) |
| M2245 V***btname***\n | Set the name of Bluetooth, 11 letters limited (Do not add #***n*** in this command) | ok \n |
| M2246\n | Rewrite UUID | ok\n |
| #***n*** M2300 N***10***\n | Please check the Grove modules & OpenMV below | |

| | | |
|---|---|---|
| #***n*** M2301 N***10*** V***1000***\n | Please check the Grove modules & OpenMV below | |
| #***n*** M2302 N***10*** V***1***\n | Please check the Grove modules & OpenMV below | |
| #***n*** M2303 N***17*** T***0*** V***0***\n | Please check the Grove modules & OpenMV below | |
| #***n*** M2400 S***0***\n | Set the mode of arm (0:Normal 1:Laser 2:3D printing 3:Universal holder) | $***n*** ok \n |
| #***n*** M2401\n | Set the current position into the reference position | $***n*** ok \n |
| #***n*** M2410\n | Set the height zero point | $***n*** ok \n |
| #***n*** M2411 S***100***\n | Set the offset of end-effector (mm) | $***n*** ok \n |
| #***n*** M2500\n | Please check the Grove modules & OpenMV below | |
| Querying Command (parameters are in underline) | | |
| #***n*** P2200\n | Get the current angle of joints | $***n*** ok B***50*** L***50*** R***50***\n |
| #***n*** P2201\n | Get the device name | $***n*** ok V***3.2***\n |
| #***n*** P2202\n | Get the hardware version | $***n*** ok V***1.2***\n |
| #***n*** P2203\n | Get the software version | $***n*** ok V***3.2***\n |
| #***n*** P2204\n | Get the API version | $***n*** ok V***3.2***\n |
| #***n*** P2205\n | Get the UID | $***n*** ok V***0123456789AB***\n |
| #***n*** P2206 N***0***\n | Get the angle of number 0 joint (0~2) | $n ok V***80***\n |
| #***n*** P2220\n | Get current coordinates | $***n*** ok X***100*** Y***100*** Z***100***\n |
| #***n*** P2221\n | Get current polar coordinates | $***n*** ok S***100*** R***90*** H***80***\n |
| #***n*** P2231\n | Get the status of pump | $***n*** ok V***1***\n (0 stop, 1 working, 2 grabbing things) |
| #***n*** P2232\n | Get the status of gripper | $***n*** ok V***1***\n (0 stop, 1 working, 2 grabbing things) |
| #***n*** P2233\n | Get the status of limited switch | $***n*** ok V***1*** (1 triggered, 0 untriggered) |

| | | |
|---|---|---|
| #***n*** P2234\n | Get the status of power connection | $***n*** ok V***1*** (1 connected, 0 unconnected) |
| #***n*** P2240 N***1***\n | Get the status of digital IO | $***n*** ok V***1***\n (1 High, 0 Low) |
| #***n*** P2241 N***1***\n | Get the status of analog IO | $***n*** ok V***295***\n (return the data of ADC) |
| #***n*** P2242\n | Get the default value of AS5600 in each joint | $***n*** ok B***2401*** L***344*** R***1048***\n |
| #***n*** P2400\n | Check current status | $***n*** ok V***1***\n (0: normal; 1: laser; 2: 3D printing; 3: Universal holder;) |
| Ticking feedback | | |
| @1 | Ready | |
| @3 | Timed feedback , ”M2120” | |
| @4 N***0*** V***1***\n | Report the button event.<br>N: 0 = Menu button, 1 = Play button<br>V: 1 = Click, 2 = Long Press | |
| @5 V***1***\n | Report event of power connection | |
| @6 N***0*** V***1***\n | Report event of limit switch in end-effector | |
| @7 temp error | Temperature error in 3D printing | |
| Err Output | | |
| E20 | Command not exist | |
| E21 | Parameter error | |
| E22 | Address out of range | |
| E23 | Command buffer ssssfull | |
| E24 | Power unconnected | |
| E25 | Operation failure | |
| Grove modules & OpenMV | | |
| N is the ID of each grove modules:<br>***10***: Color sensor; ***11***: Gesture sensor; ***12***: Ultrasonic; ***13***: Fan; ***14***: Electromagnet; ***15***: Temperature & Humidity; ***16***: PIR Motion; ***17***: RGB LCD; | | |
| #***n*** M2300 N***10***\n | Initialize the Grove modules, N is the ID of each module | |
| #***n*** M2301 N10 V***1000***\n | Auto report time for color | @10 N10 R***20*** G***10*** B***255***\n |

| | | |
|---|---|---|
| | sensor, V is the time (microsecond) | (RGB value) |
| #***n*** M2301 N11 V***1000***\n | Auto report time for gesture sensor, V is the time (microsecond) | @10 N11 V***16***\n<br>(***1***: right; ***2***: left; ***4***: up; ***8***: down; ***16***: forward; ***32***: backward; ***64***: CW; ***128***: CCW;) |
| #***n*** M2301 N12 V***1000***\n | Auto report time for ultrasonic, V is the time (microsecond) | @10 N12 V***27***\n<br>(The distance value in cm) |
| #***n*** M2301 N15 V***1000***\n | Auto report time for T&H, V is the time (microsecond) | @10 N15 T***32.12*** H***76.5***\n<br>(temperature in ℃, humidity in %) |
| #***n*** M2301 N16 V***1000***\n | Auto report time for PIR motion, V is the time (microsecond) | @10 N16 V***1***\n<br>(1: motion; 0: no motion;) |
| #***n*** M2302 N13 V***128***\n | Fan setting, V is the duty cycle from 0-255 | $***n*** ok\n |
| #***n*** M2302 N14 V***1***\n | Electromagnet setting, 0 is off, 1 is on | $***n*** ok\n |
| #***n*** M2303 N17 T***0***\n | Turn off/on display (0 is off, 1 is on, 2 is clear) | $***n*** ok\n |
| #***n*** M2303 N17 R***25*** G***25*** B***25***\n | Change the rgb value of backlight | $***n*** ok\n |
| #***n*** M2303 S***1*** V***Text***\n | S is the line (1 or 2), V is the text content<br>For example:<br>M2303 S***1*** V***ufactory*** | $***n*** ok\n |
| #***n*** M2500\n | Switch the uart0 to uart2 for external TTL uart communication (For example OpenMV) | $***n*** ok \n |

## d. Different modes for uArm Swift Pro

Since different types of the end-effectors have different length and height, so we designed the command M2400, which could help us to fit the uArm into different situations easily. With this command, there is no need to concern about how to adjust the parameters for different situations.

Currently we offer 4 kinds of mode:
M2400 S0 : Normal mode (end-effector tools: suction)
M2400 S1 : Laser mode (end-effector tools: laser)
M2400 S2 : 3D printing mode (end-effector tools: hot end)

M2400 S3 : Universal holder mode (end-effector tools: universal holder)

For the gripper, there is no special mode since gripper has the fingers and can rotate horizontally.

# uArm Community

UFACTORY Official Forum

uArm User Facebook Group

Ask for Help

# Release Note

| Version | Note | |
|---|---|---|
| 1.0.0 | Setup the document | Tony |
| 1.0.1 | Update the working range | Tony |
| 1.0.2 | Add the mounting and detail size of each part<br>Add detail steps of Arduino upload | Tony |
| 1.0.3 | Add the relationship of left/right motor with the upper and lower arm | Tony |
| | | |
| | | |
| | | |
| | | |
| | | |