

uArm Swift Pro

Developer Guide
V1.0.6

Contents

SAFETY INSTRUCTIONS	4
GENERAL INFORMATION	4
1.Source file	4
2.Reference frame.....	5
3.Coordinate	6
4.Mounting.....	8
5. Buttons & indicator lights.....	10
6.Extension description.....	11
SPECIFICATIONS.....	13
APPLICATION INFORMATION.....	16
1.Send command over usb cable.....	16
2.Send command over bluetooth	19
3.The 2nd UART.....	23
4.Arduino	25
5.Ros & Python.....	27
6.Openmv demo	27
7.RECOVER FROM THE WRONG CODE.....	27
PROTOCOL.....	28

1.Introduction.....	28
2.Example.....	28
3.Commands(tbd).....	28
UARM COMMUNITY.....	40
RELEASE NOTE.....	40

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.

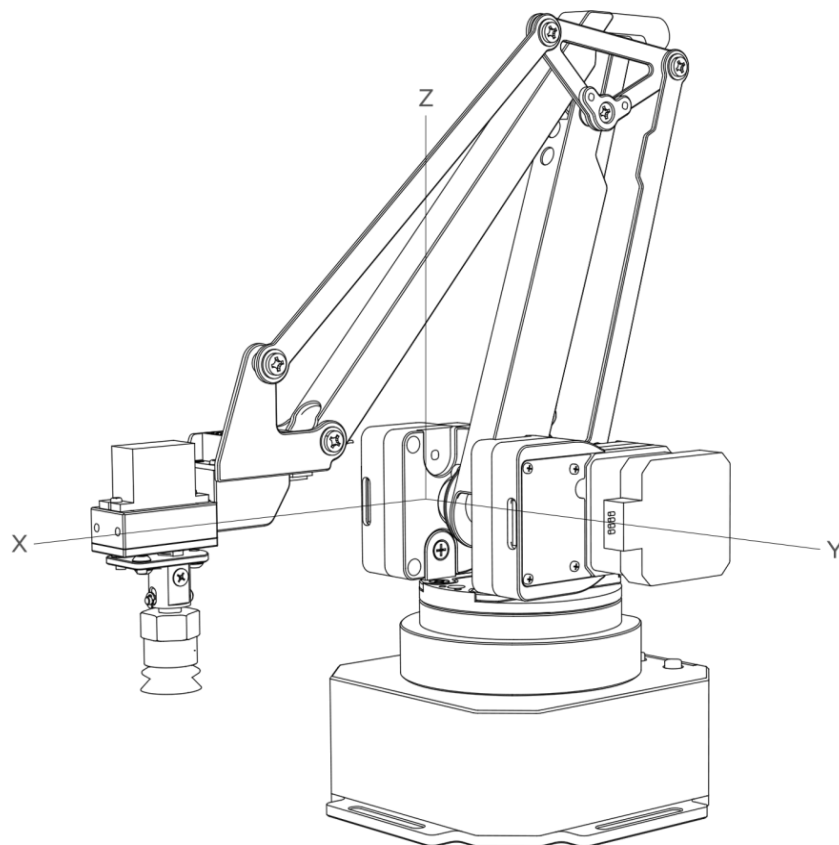
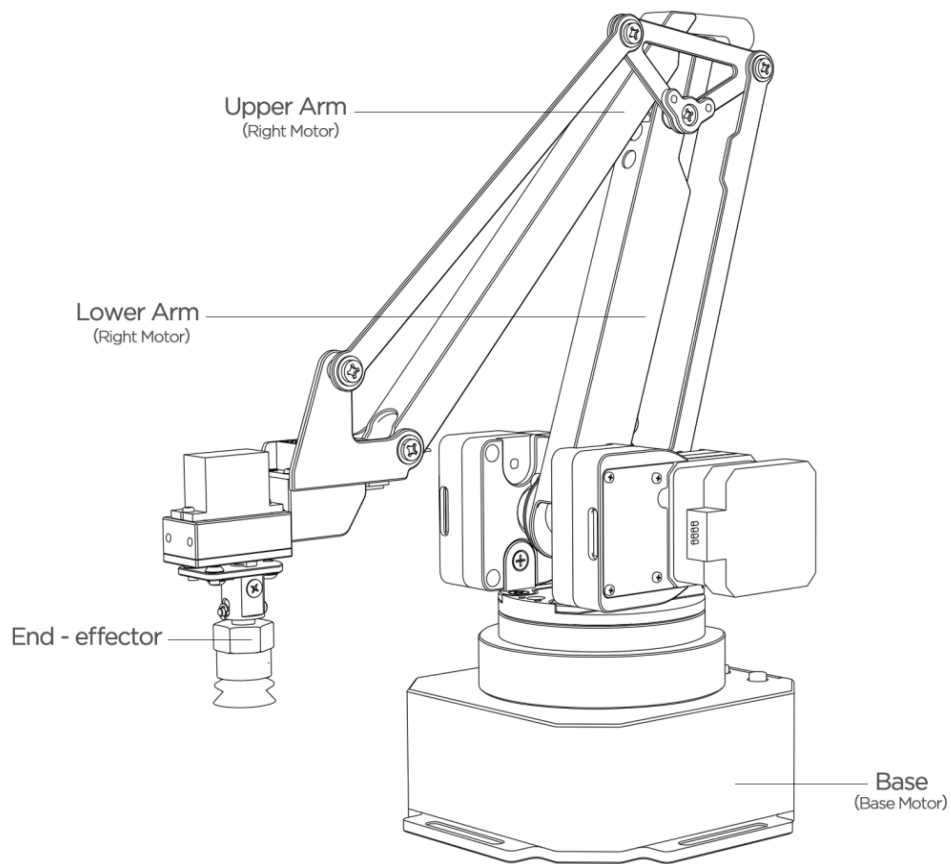
General Information

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

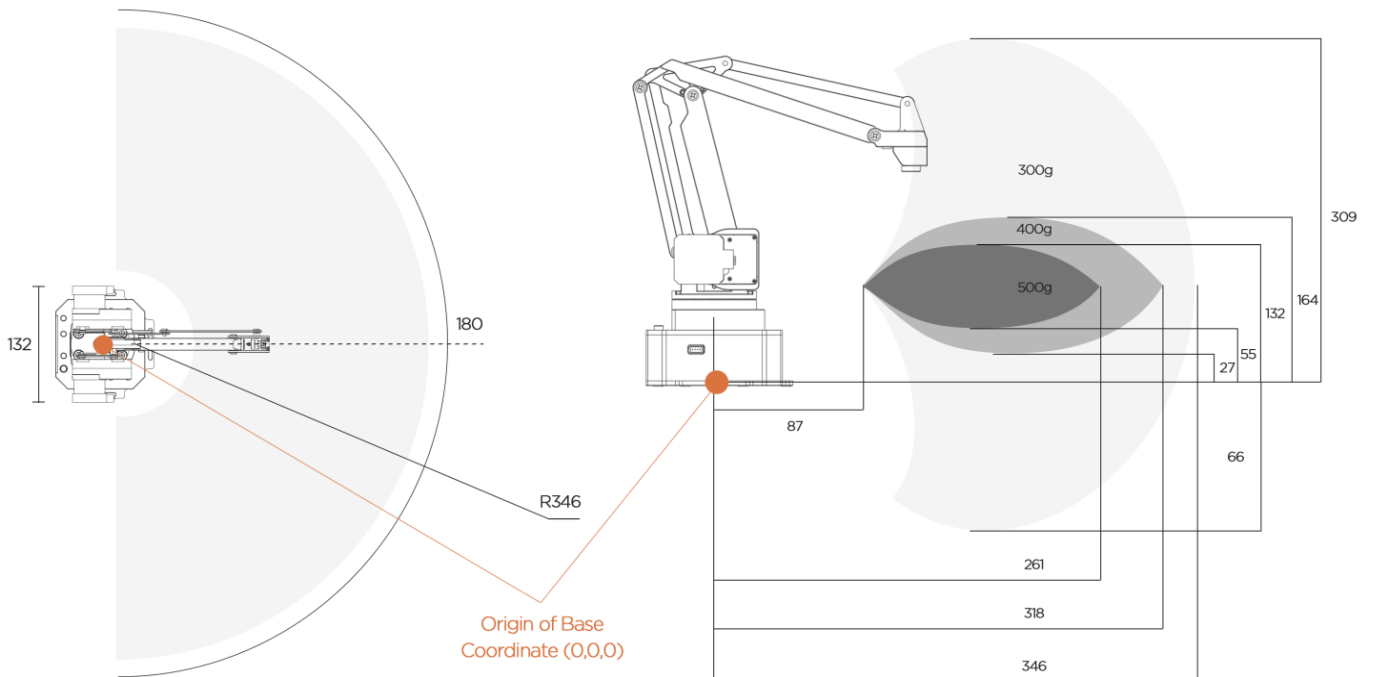
1. Source file

- 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

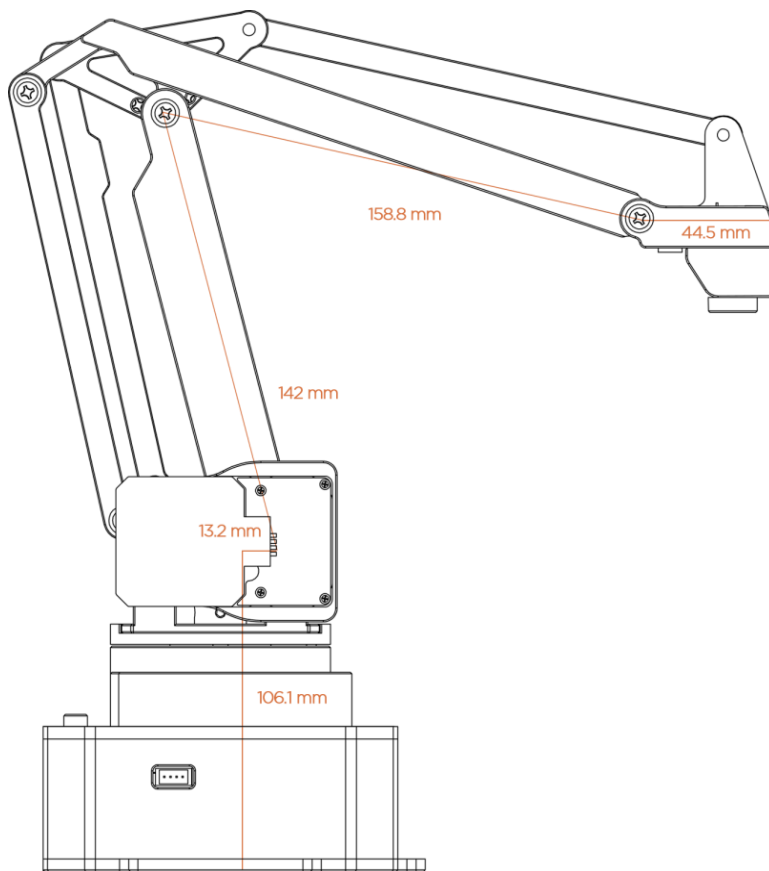


3.Coordinate



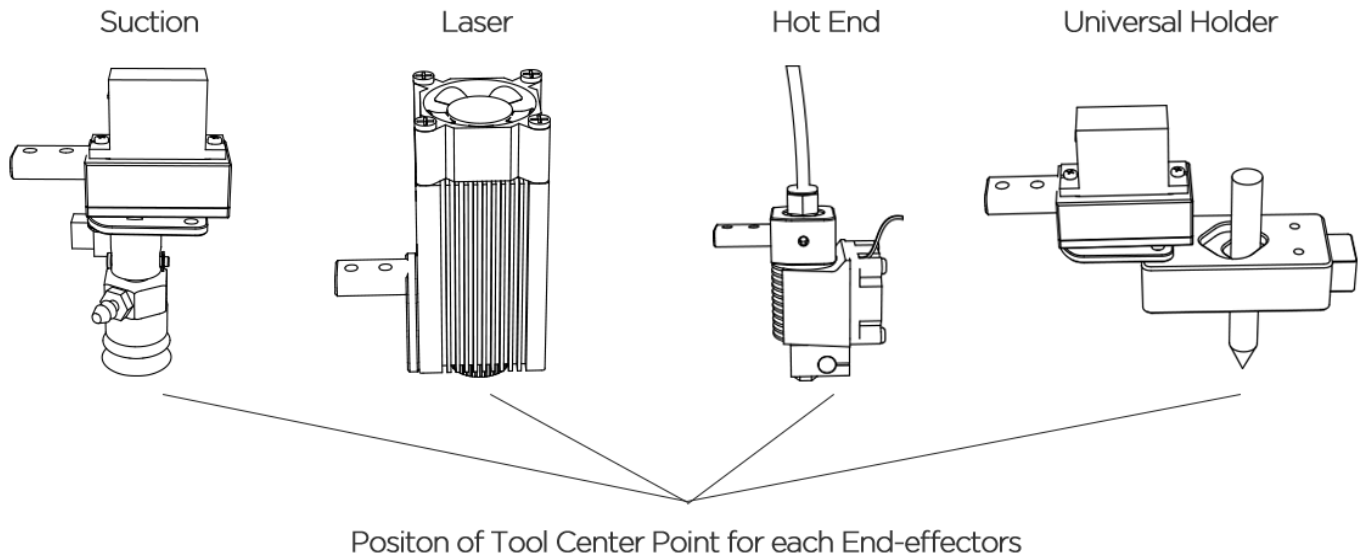
The picture in the right also shows the dynamic payload range of uArm.

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.



Currently we offer 4 kinds of mode:

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

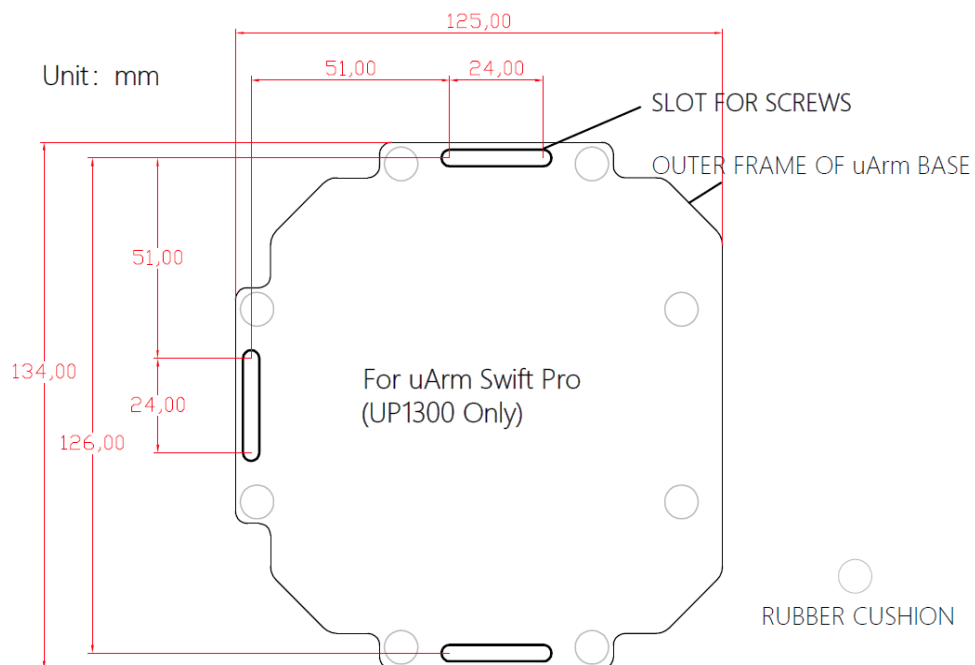
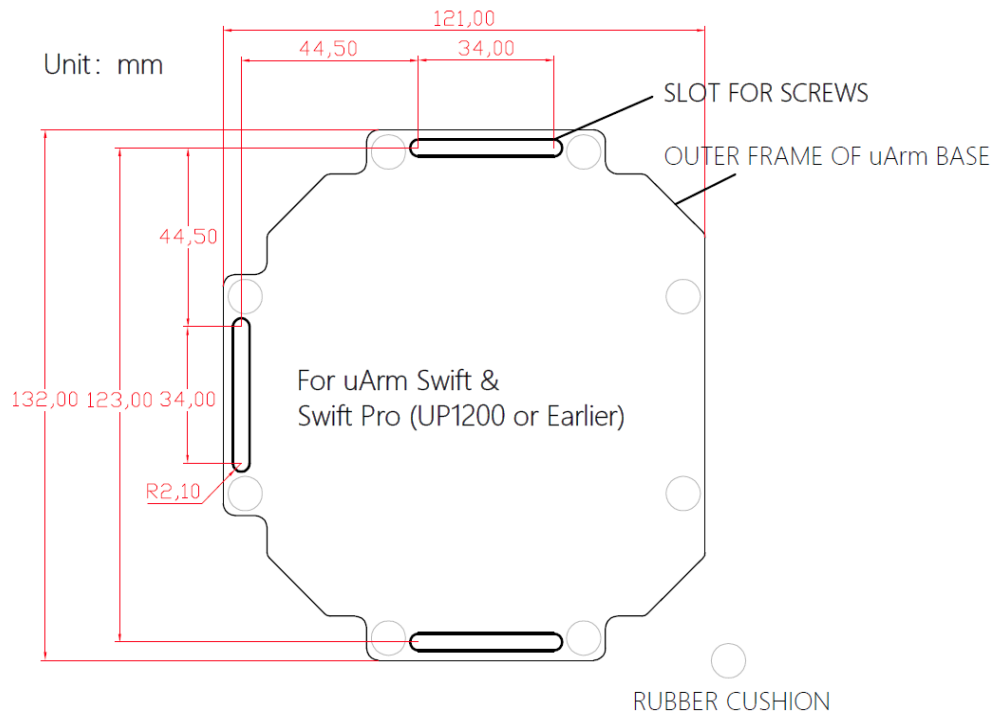
M2400 S1: Laser mode (end-effector tools: laser)

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

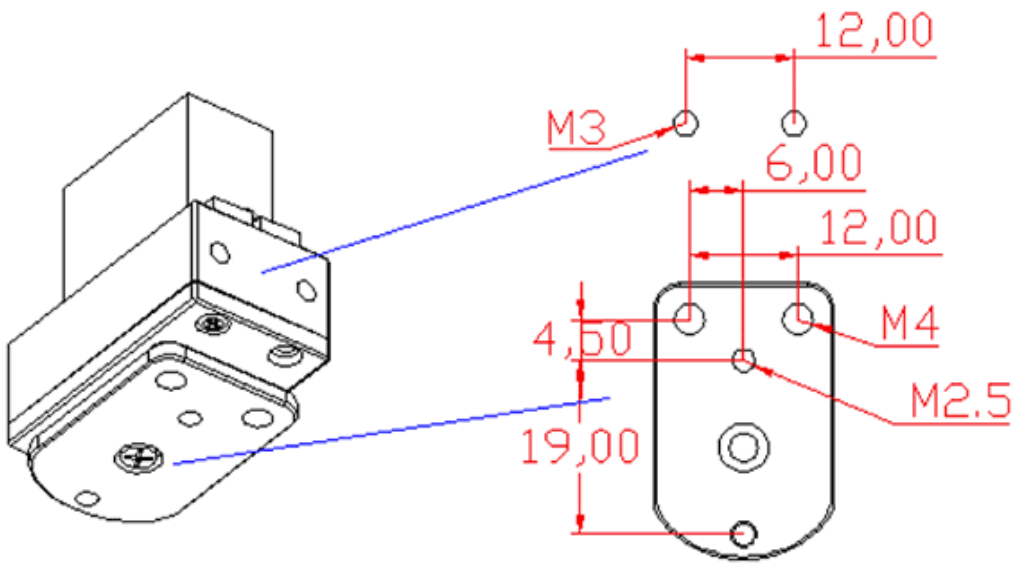
M2400 S3: Universal holder mode (end-effector tools: universal holder)

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

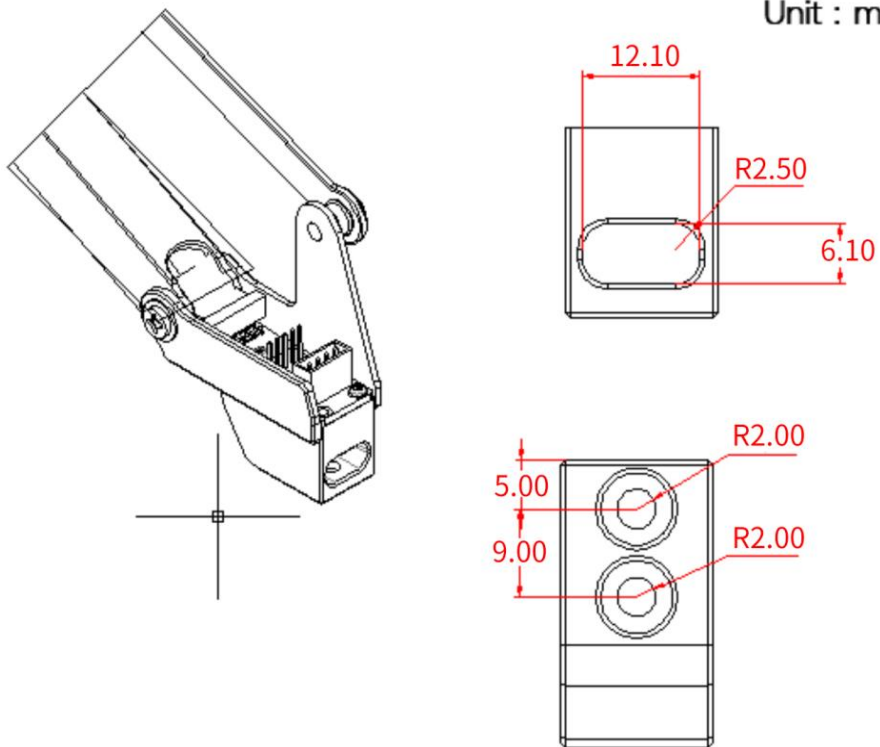
4. Mounting



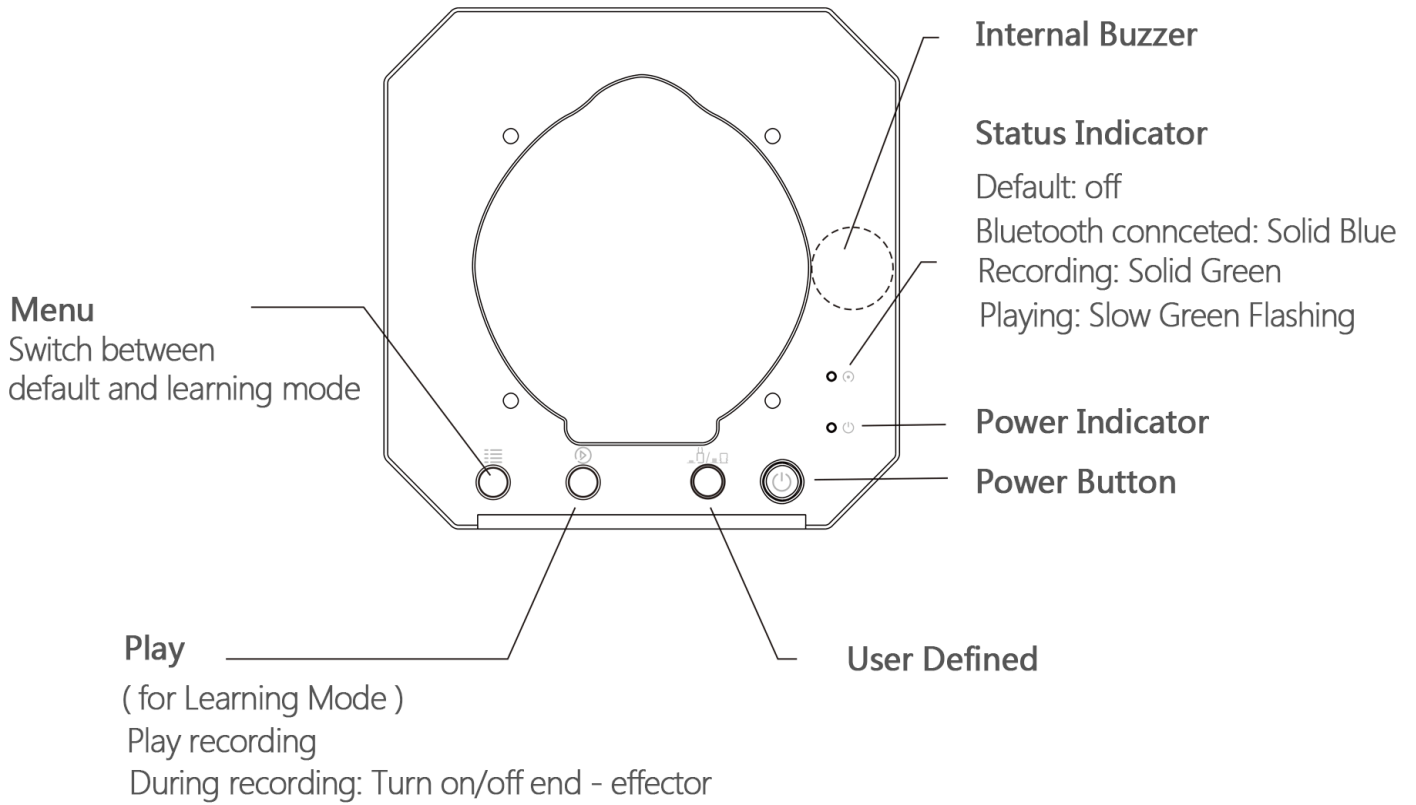
Unit: mm



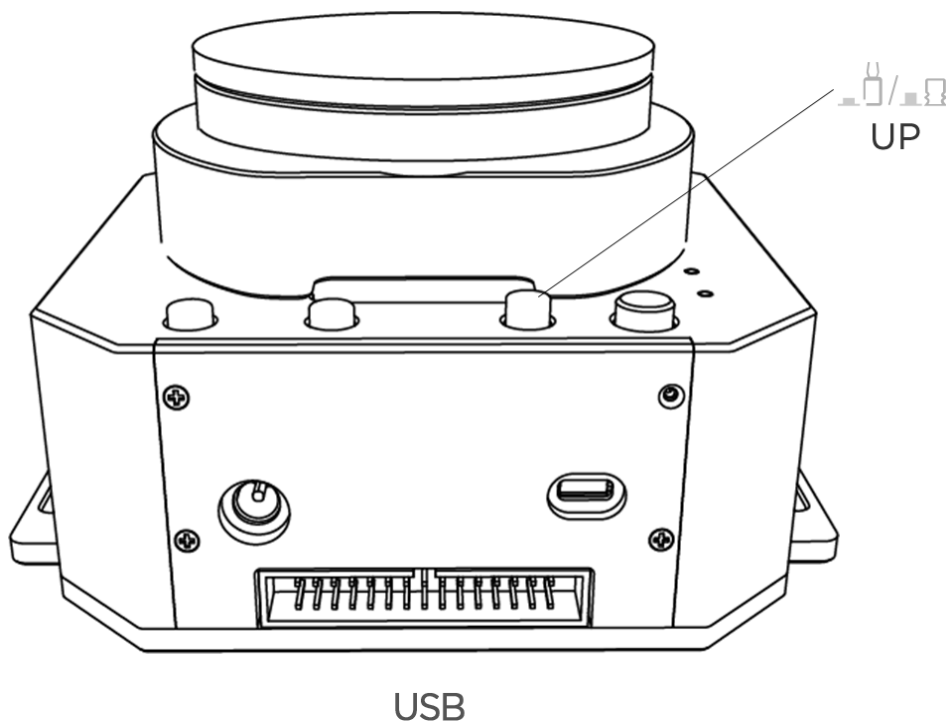
Unit : mm

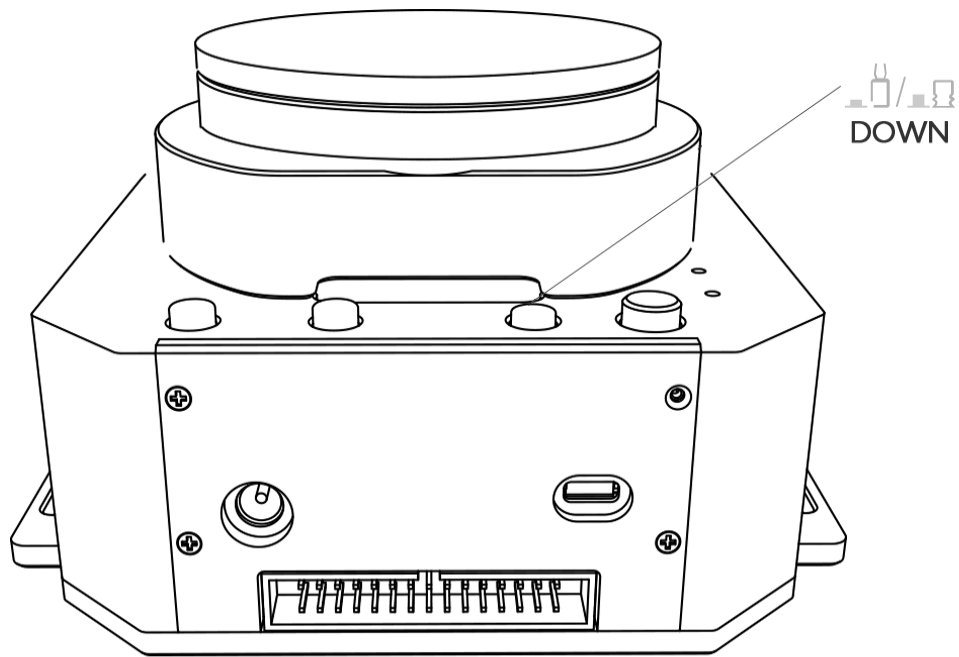


5. Buttons & Indicator Lights



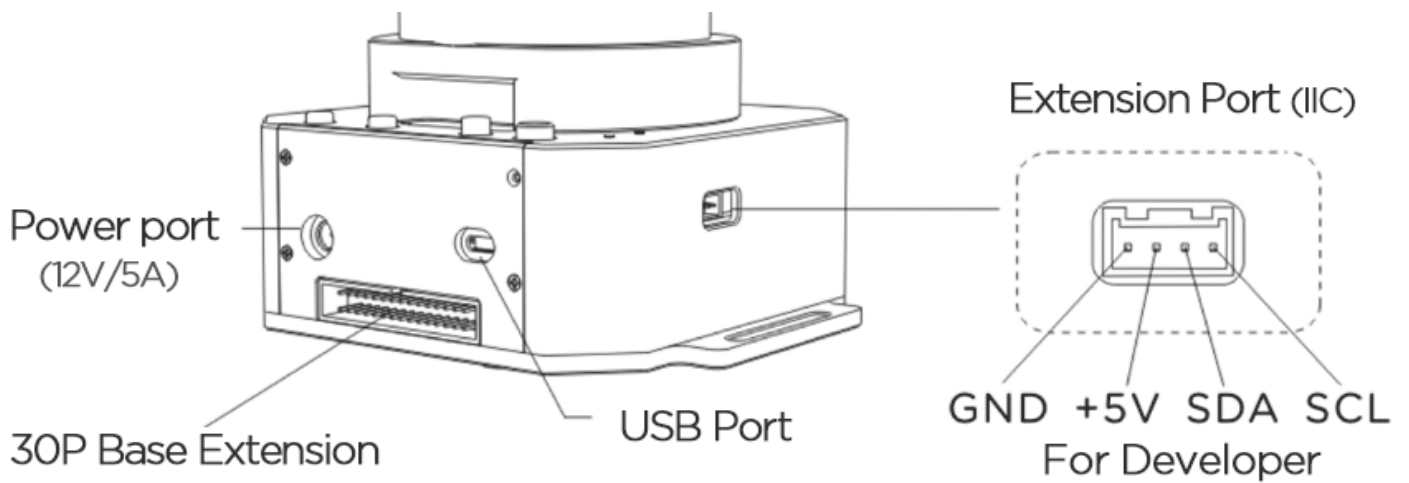
Caution: By default, the user defined button is for switching between Bluetooth and USB mode. Please ensure the button is UP while communicating with uArm via USB.

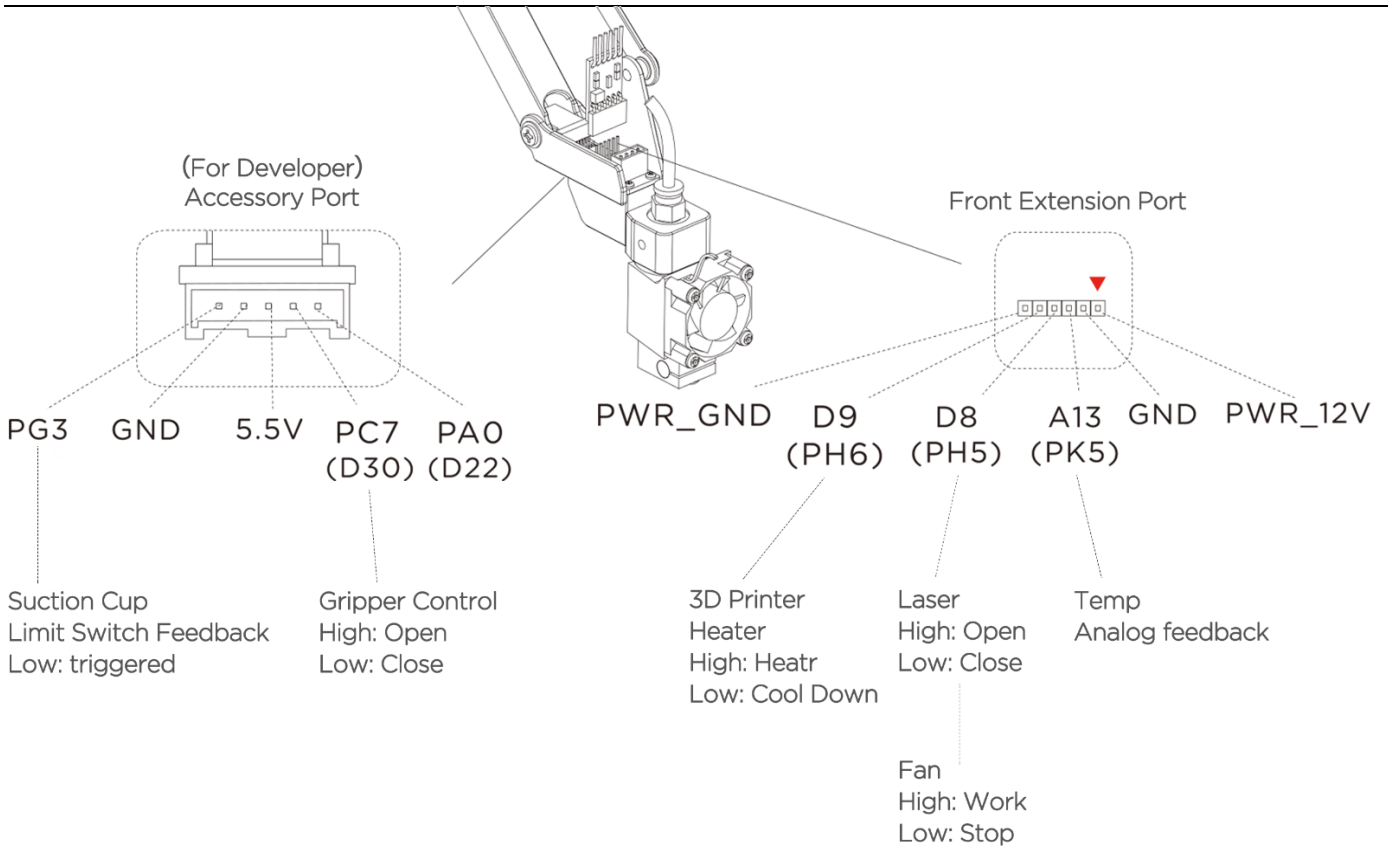




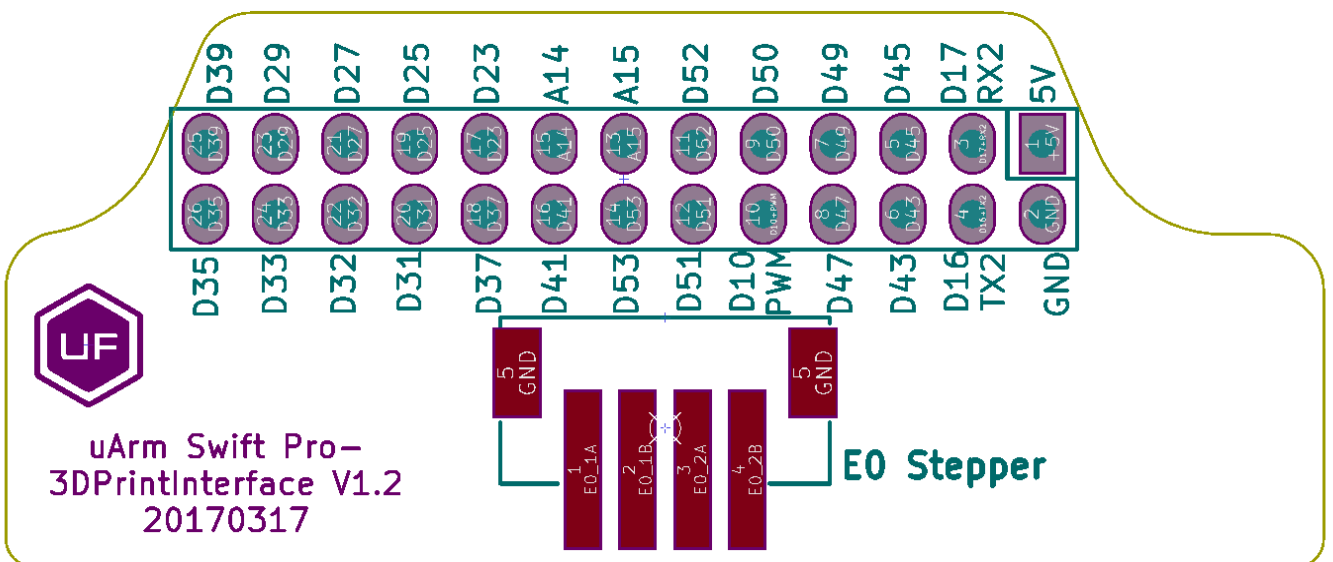
Bluetooth

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°C-35°C 30%RH-80%RH noncondensing
Storage Temperature & Humidity	-20°C-60°C 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	

	Angle	Speed	Lifetime	Torque
Base Motor	0° ~ 180°	40°/s	>3000h	12kg · cm
Left Motor	0° ~ 130°	40°/s	>3000h	12kg · cm
Right Motor	0° ~ 106°	40°/s	>3000h	12kg · cm
End-effector Motor	0° ~ 180°	60°/s	500h	2kg · cm

Accessories		
Integrated Pump	Suction Diameter	5mm ~ 10mm
	Max. Pressure	33kPa
	Max. Lifting Weight	1000g
	Feature	With feedback
Metallic Universal Holder	Weight	36g
	Dimension(L*W*H)	62mm*25mm*15mm
	Material	Aluminum
	Holder Diameter	14mm
Metallic 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
Integrated Bluetooth	Frequency Band(3)	100M
	Operating frequency/frequencies	2400Mhz – 2483.5MHz

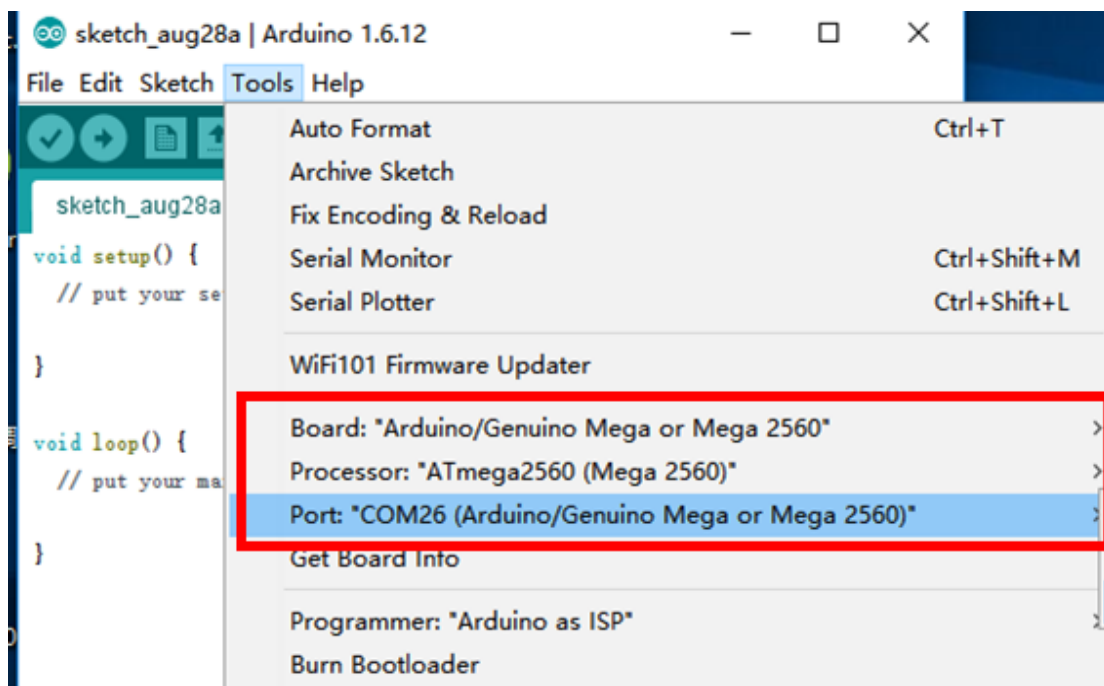
	Channel spacing / bandwidth	2MHz
	Transmit power	0dBm
	Mode of operation	GFSK
	(simplex / duplex)	simplex
	Duty cycle	100ms
	Access Protocol (if applicable)	Bluetooth Low Energy 4.0
	Transmitter Class type	CLASS 2
3D Printing Kit (Only for Pro)	Type	E3D v6
	Hot End Dimension(L*W*H)	37mm*31mm*76mm
	Extruder Dimension(L*W*H)	61mm*52*66mm
	Consumption	35W
	Nozzle Diameter	0.4mm
	Max Temp	270 °C
	Material	PLA
	Max. Printing Speed	20mm/s
	File Format	.Gcode
	Printing Size(L*W*H)	10mm*10mm*10mm
Laser Engraving Kit (Only for Pro)	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

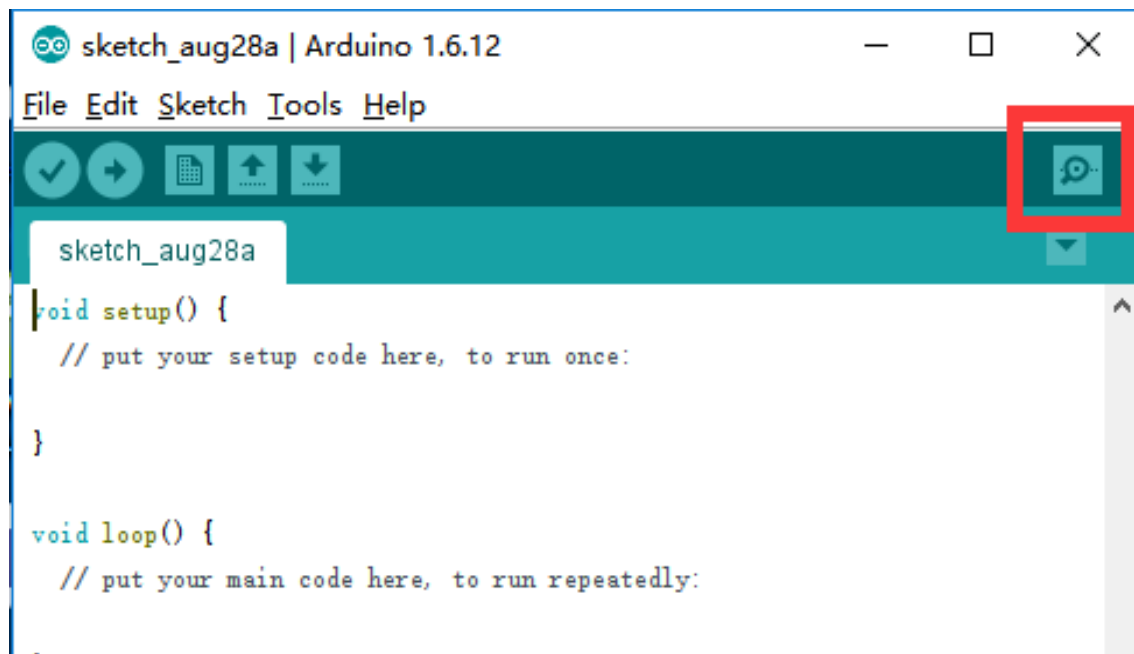
We would introduce several ways to play with the robot arm in different platform.

1.Send Command over USB Cable

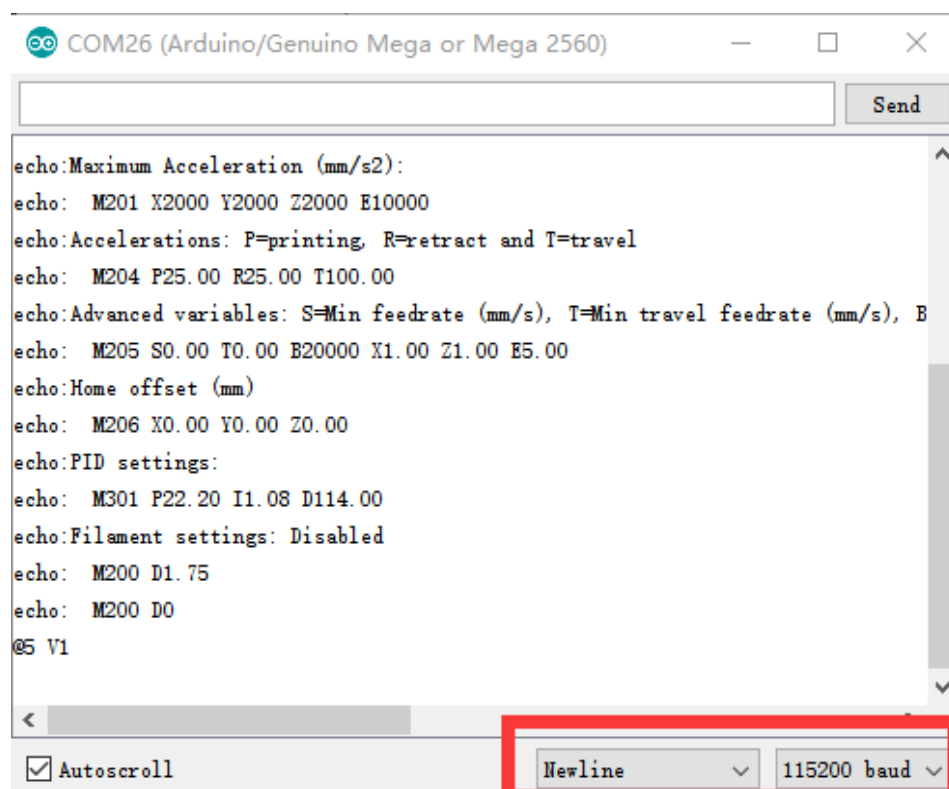
Power on the uArm and open the Arduino IDE. And setting the board like the picture below. Please make sure the port you are choosing is the correct port of uArm.



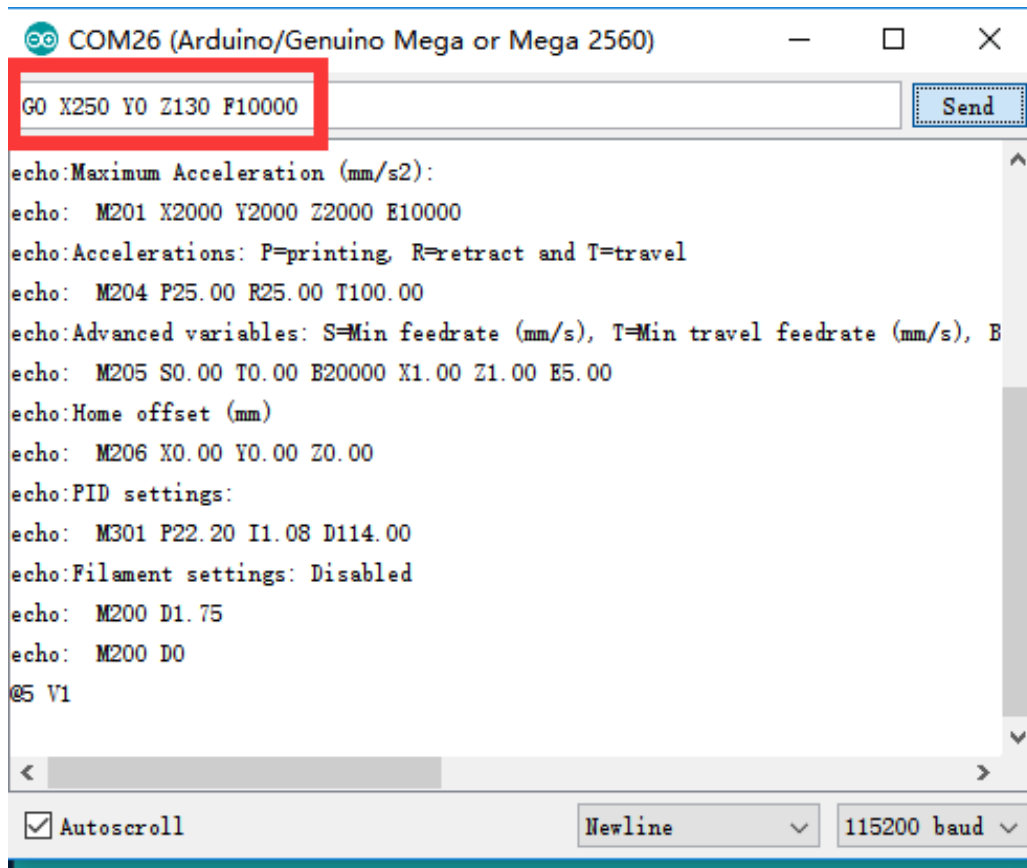
Open the serial monitor in the right side of Arduino IDE. After clicking, and you could hear a beep which means the uArm is connected.



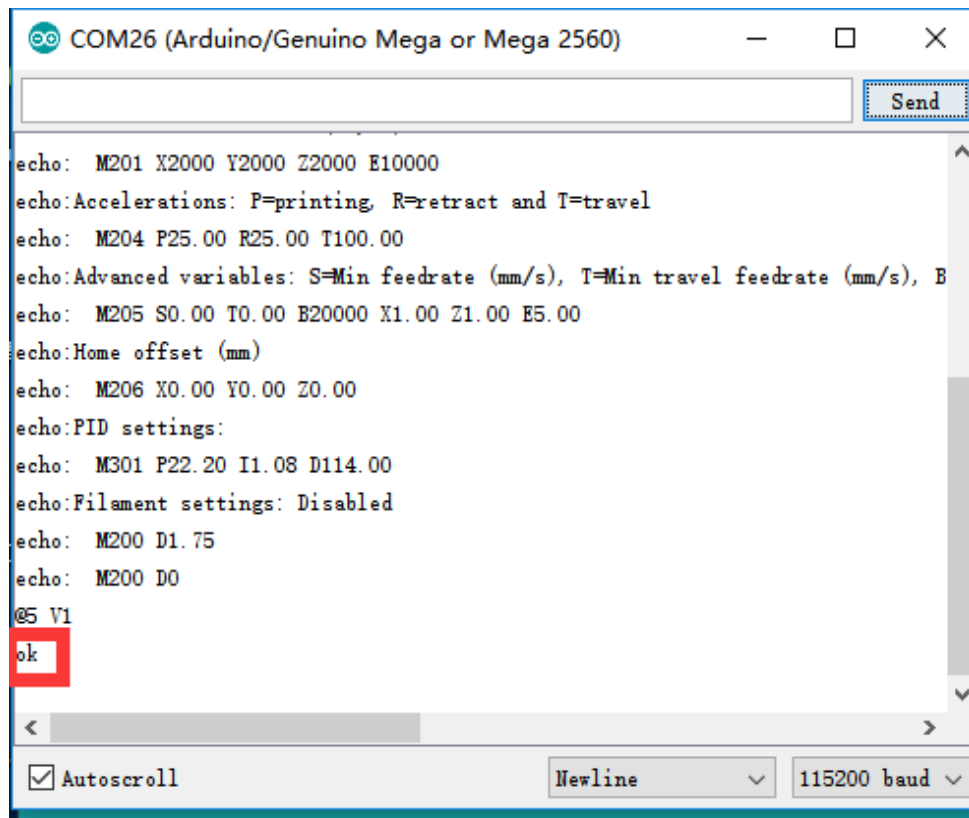
Set the parameter of serial monitor in the right bottom. If the setting is correct, you would receive the detail information from uArm like the picture below.



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



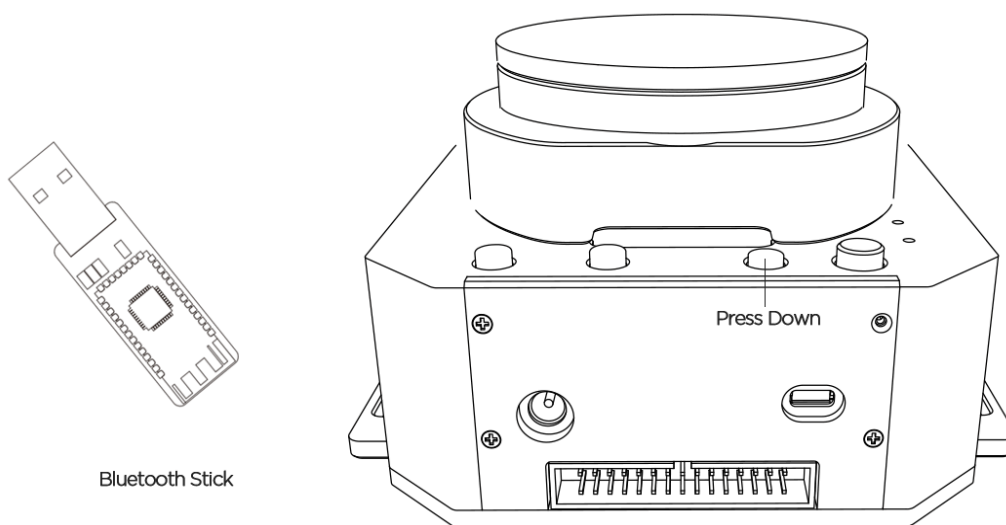
If uArm finishes the movement, it replies "ok" .



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

2.Send Command over Bluetooth

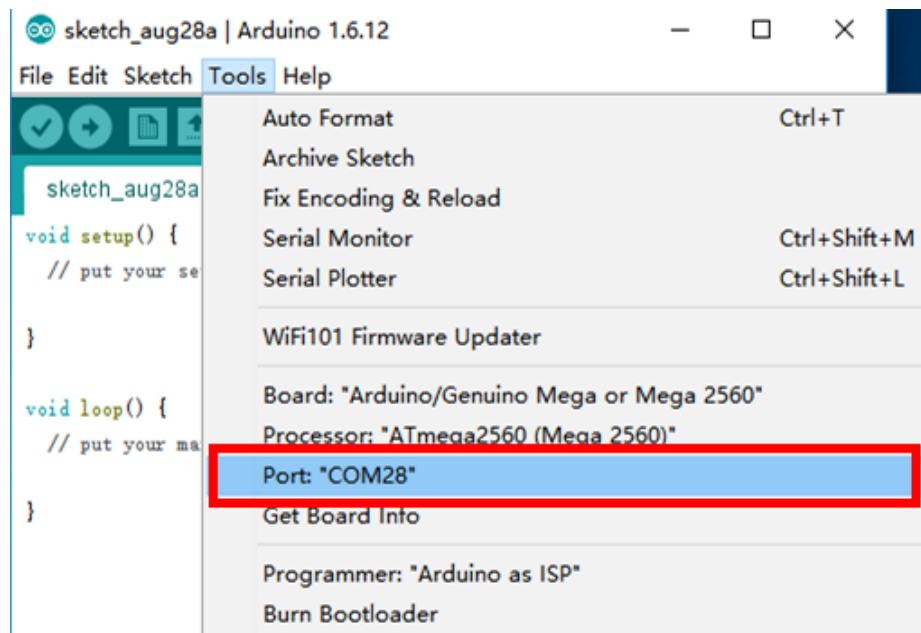
Plug in the Bluetooth stick, and press down the button beside the power button.



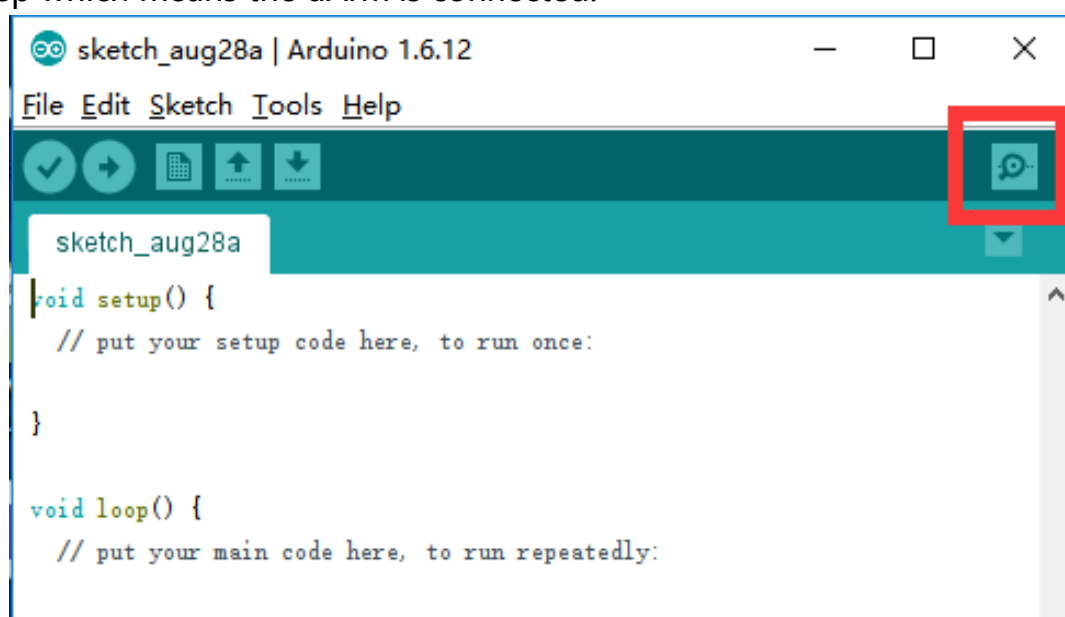
Power on the uArm. When the Bluetooth stick is searching, the blue indicator keeps blink until the wireless connection is built up between stick and uArm. And the blue indicators in both stick and uArm become solid.

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.

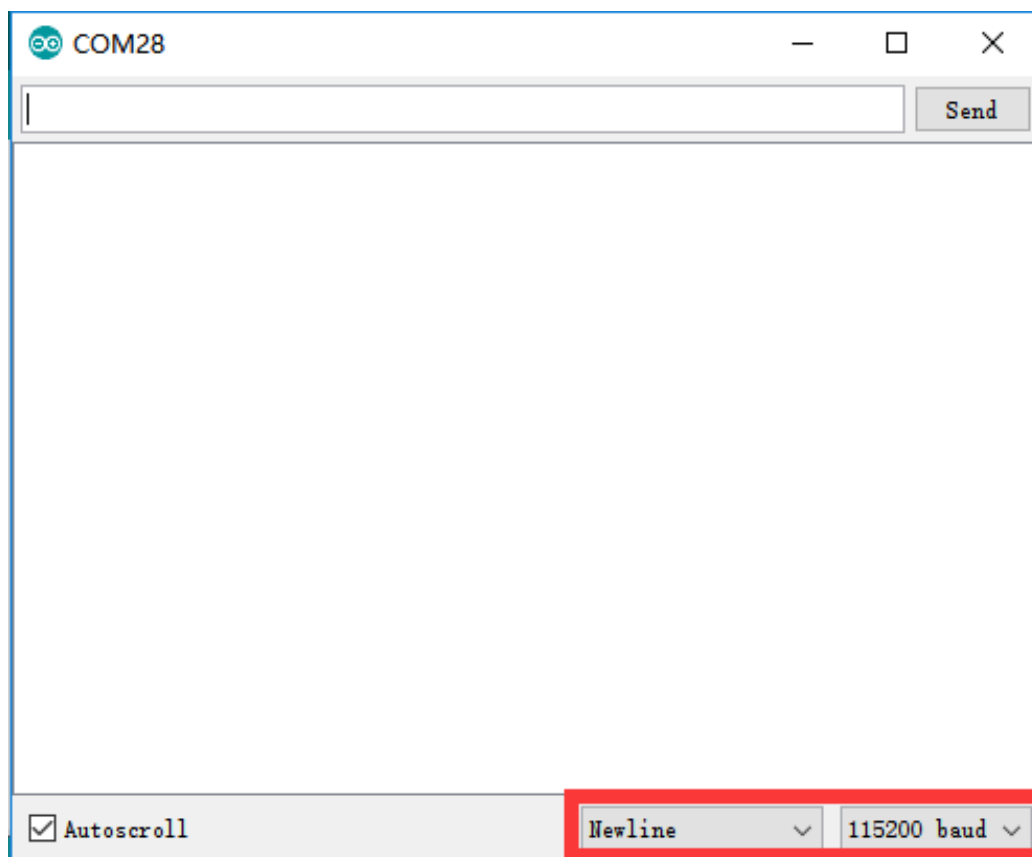
(Driver of stick: <http://www.ftdichip.com/Drivers/VCP.htm>)



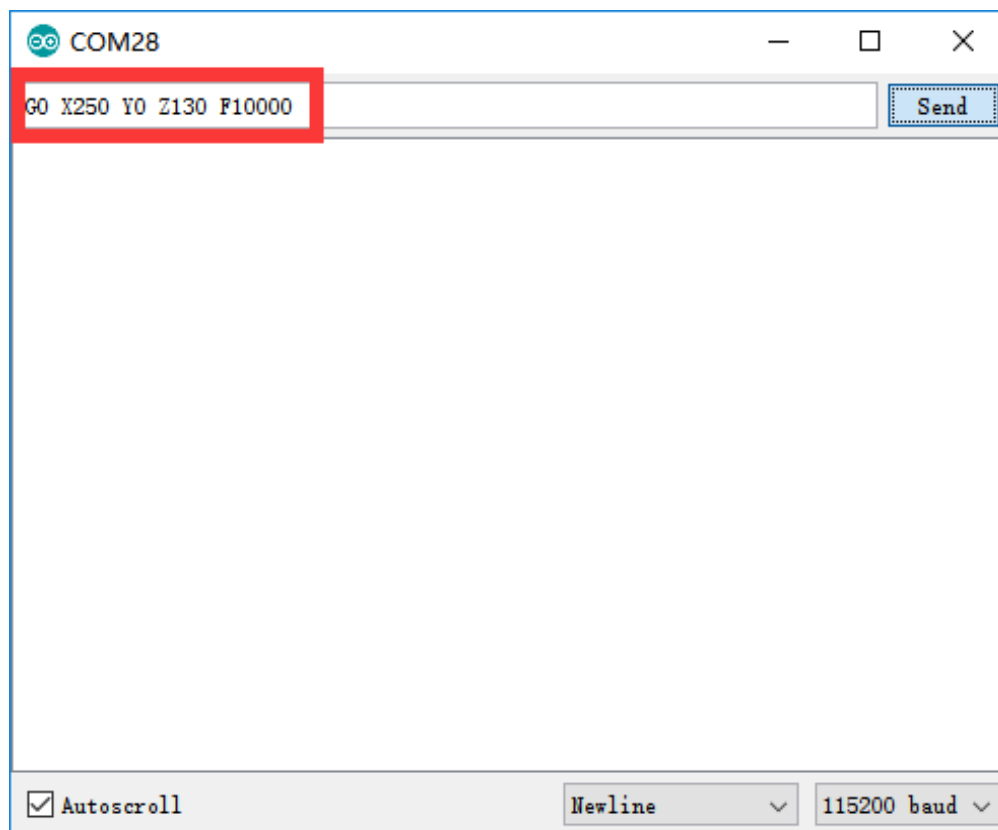
Open the serial monitor in the right side of Arduino IDE. After clicking, and you could hear a beep which means the uArm is connected.



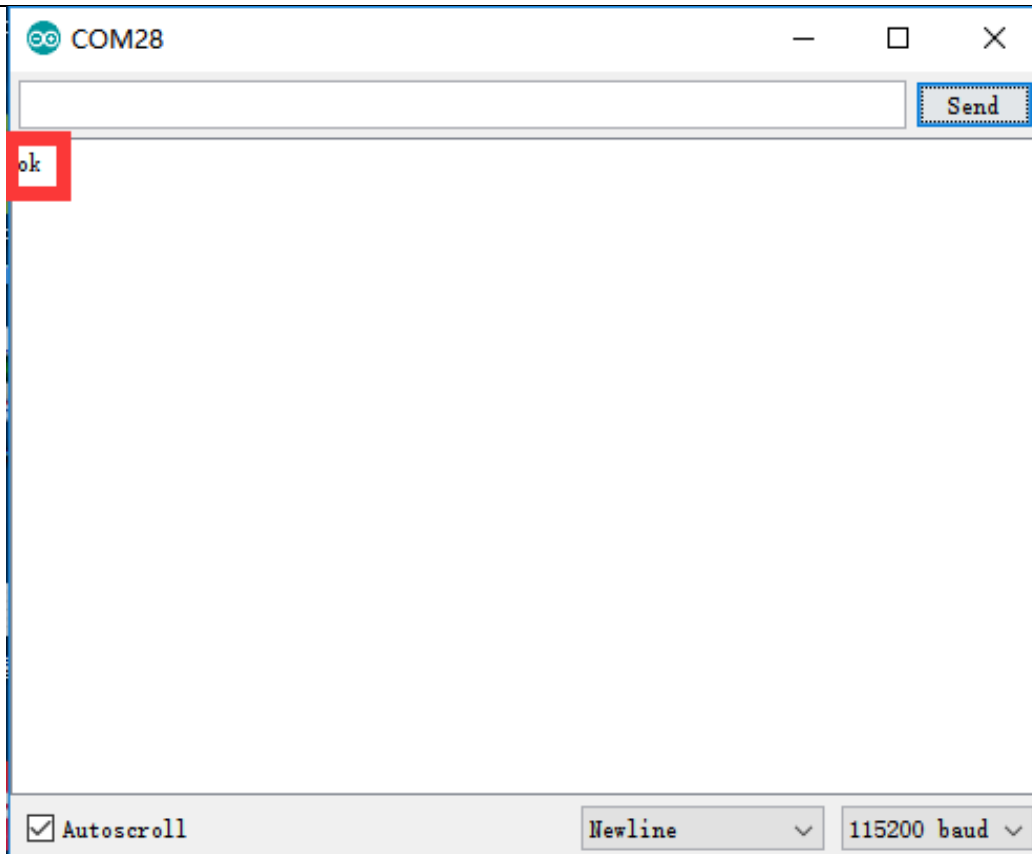
Set the parameter of serial monitor in the right bottom like the picture below.



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" .



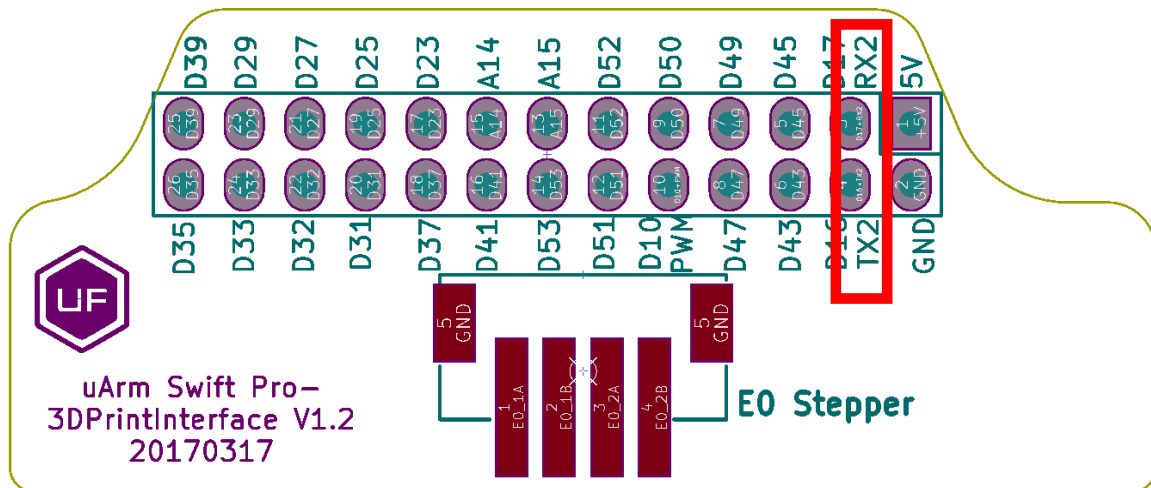
Please check the chapter of Protocol (Page 20) in this guide to test more commands.

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.

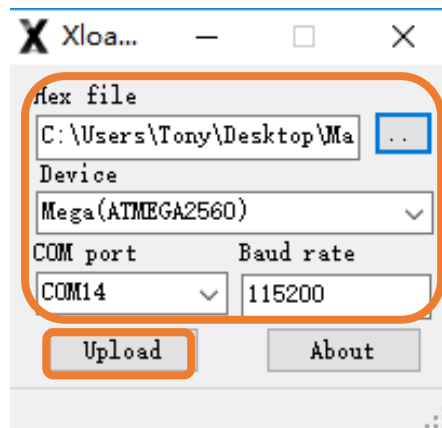
All the pins of extension board are connected with the Arduino MEGA 2560 directly so it's TTL level. And voltage above 5V might burn the IO out.



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.

Switch to 2nd UART communication

Connect uArm Swift Pro to PC. Open XLoader (xloader.russeotto.com/), load uArmSwiftPro_2ndUART.hex (<https://github.com/TonyLeheng/Vision-Pick-and-Place>) Click "Upload" to upload the code to uArm Swift Pro.



Switch to USB communication

Connect uArm Swift Pro to PC, open XLoader (xloader.russeotto.com/), load uArmSwiftPro_Standard.hex (<https://github.com/TonyLeheng/Vision-Pick-and-Place>). Click "Upload" to upload the code to uArm Swift Pro.

4.Arduino

The main code is written by Arduino IDE. Please check the link below:

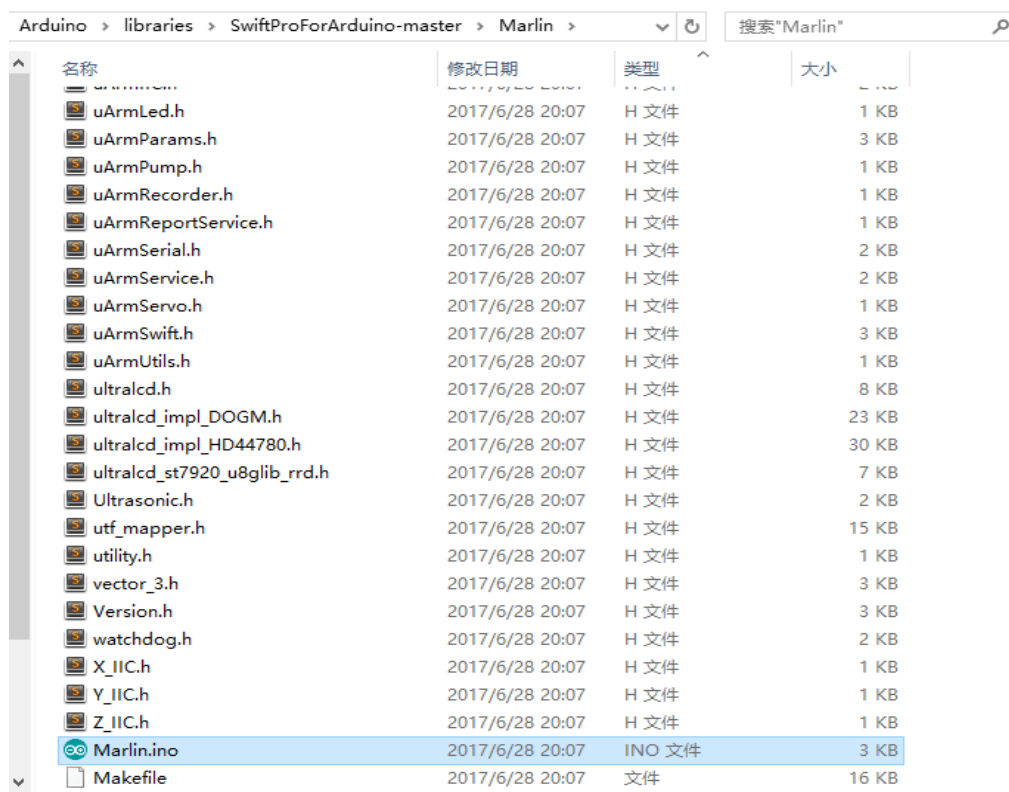
<https://github.com/uArm-Developer/SwiftProForArduino>

How to compile and upload the file?

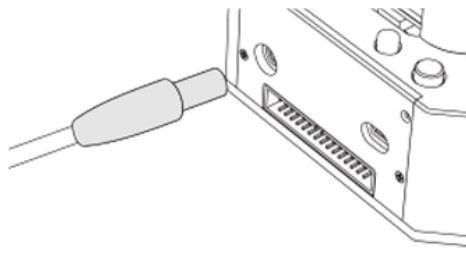
1. Download the code and extract it to your Arduino libraries folder (normally it' s in C:/Users/ufactory/documents/Arduino/libraries/)



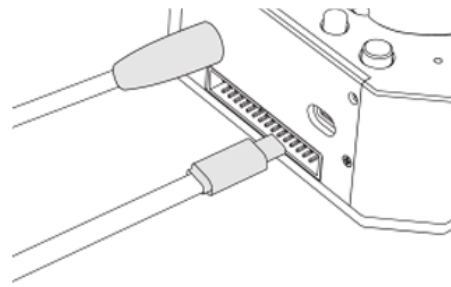
2. Find the file named Marlin.ino in the Marlin folder and open it



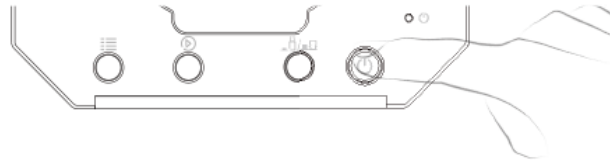
3. Plug in the USB cable and power port then turn on the power button



Plug in the power port

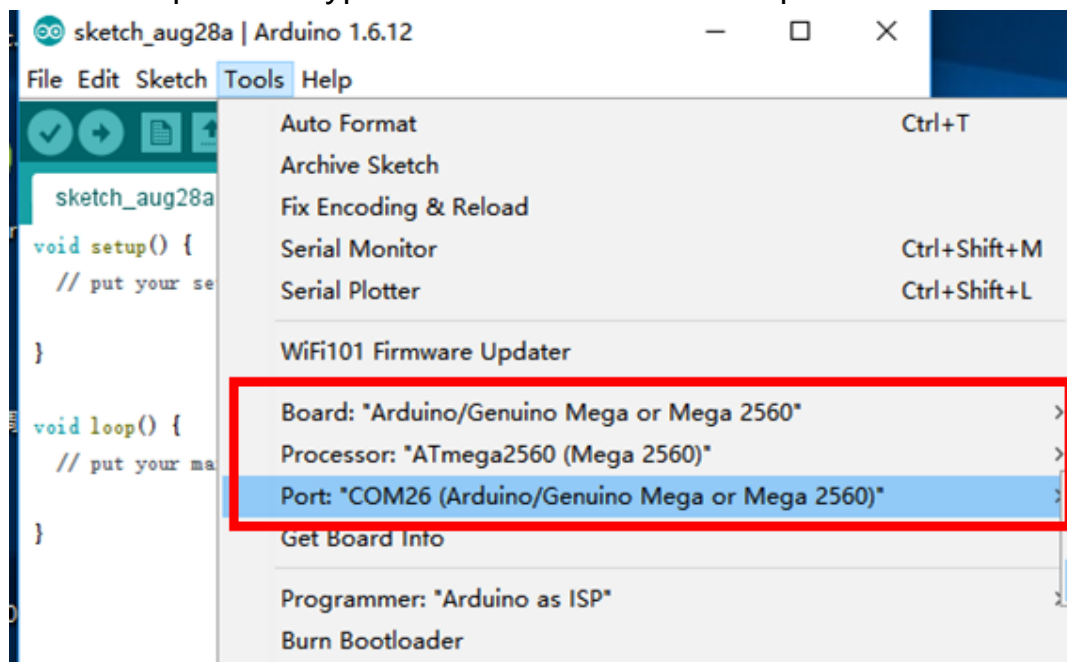


Connect the uArm with USB cable

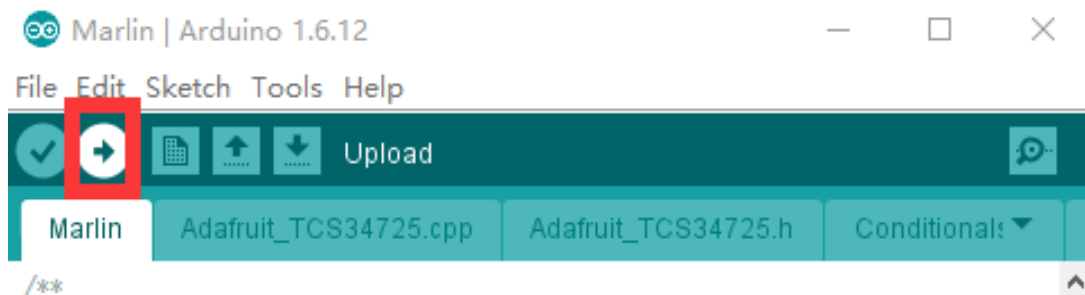


Turn on the power button

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.

7.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

- uArm 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
<p>1. #<u>n</u> is used for the debug, if you don' t want to use it please remove it directly. (For Example: G2202 N<u>0</u>V<u>90</u>\n)</p> <p>2. '\n' is the symbol of line feed.</p>		
Moving Command (parameters are in underline)		
# <u>n</u> G0 X <u>100</u> Y <u>100</u> Z <u>100</u> F <u>1000</u> \n	Move to XYZ(mm), F is speed(mm/min)	\$ <u>n</u> ok \n or \$ <u>n</u> E <u>x</u> \n (refer to Err output)
# <u>n</u> G1 X <u>100</u> Y <u>100</u> Z <u>100</u> F <u>1000</u> \n	After entering the laser mode (M2400 S1), command G1 means laser on, G0 means off.	\$ <u>n</u> ok \n or \$ <u>n</u> E <u>x</u> \n (refer to Err output)
# <u>n</u> G2004 P <u>1000</u> \n	Delay microsecond	\$ <u>n</u> ok \n
# <u>n</u> G2201 S <u>100</u> R <u>90</u> H <u>80</u> F <u>1000</u> \n	Polar coordinates, S is stretch(mm), R is rotation(degree),H is height(mm), F is speed(mm/min)	\$ <u>n</u> ok \n or \$ <u>n</u> E <u>x</u> \n (refer to Err output)
# <u>n</u> G2202 N <u>0</u> V <u>90</u> \n	Move the motor to the position ,N is ID of joints(0~3),V is angle(0~180)	\$ <u>n</u> ok \n or \$ <u>n</u> E <u>x</u> \n (refer to Err output)
# <u>n</u> G2204 X <u>10</u> Y <u>10</u> Z <u>10</u> F <u>1000</u> \n	Relative displacement	\$ <u>n</u> ok \n or \$ <u>n</u> E <u>x</u> \n (refer to Err output)

# <u>n</u> G2205 S <u>10</u> R <u>10</u> H <u>10</u> F <u>1000</u> \n	Polar coordinates for relative displacement	\$ <u>n</u> ok \n or \$ <u>n</u> E <u>x</u> \n (refer to Err output)
Setting Command (parameters are in underline)		
# <u>n</u> M17\n	Attach all the joint motors	\$ <u>n</u> ok \n
#n M204 P200 T200 R200\n	Set accelerations and save P = Printing moves R = Extruder only (no X, Y, Z) moves T =Hot End Travel (non printing) moves	\$ <u>n</u> ok \n
# <u>n</u> M2019\n	Detach all the joint motors	\$ <u>n</u> ok \n
# <u>n</u> M2120 V <u>0.2</u> \n	Set time cycle of feedback, return Cartesian coordinates, V is time(seconds)	@3 X <u>154.71</u> Y <u>194.91</u> Z <u>10.21</u> \n
# <u>n</u> M2122 V <u>1</u> \n	Report (@9 V0) when stop. V1: Enable V0: Disable	\$ <u>n</u> ok \n
# <u>n</u> M2201 N <u>0</u> \n	attach motor, N is ID of joints(0~3)	\$ <u>n</u> ok \n or \$ <u>n</u> E <u>x</u> \n (refer to Err output)
# <u>n</u> M2202 N <u>0</u> \n	Detach motor, N is ID of joints(0~3)	\$ <u>n</u> ok \n or \$ <u>n</u> E <u>x</u> \n (refer to Err output)
# <u>n</u> M2203 N <u>0</u> \n	Check if the motor is attached, N is ID of joints(0~3)	\$ <u>n</u> ok V <u>1</u> \n (1 attached,0 detached)
# <u>n</u> M2210 F <u>1000</u> T <u>200</u> \n	buzzer,F is frequency, T is time (ms)	\$ <u>n</u> ok \n or \$ <u>n</u> E <u>x</u> \n (refer to Err output)

# <u>n</u> M2211 N <u>0</u> A <u>200</u> T <u>1</u> \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)	\$ <u>n</u> ok V <u>10</u> \n
# <u>n</u> M2212 N <u>0</u> A <u>200</u> T <u>1</u> V <u>10</u> \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	\$ <u>n</u> ok\n
# <u>n</u> M2213 V <u>0</u> \n	Default function of base buttons (0 false, 1 true)	\$ <u>n</u> ok\n
# <u>n</u> M2220 X <u>100</u> Y <u>100</u> Z <u>100</u> \n	Convert coordinates to angle of joints	\$ <u>n</u> ok B <u>50</u> L <u>50</u> R <u>50</u> \n (B joint 0,L joint 1,R joints 2, 0~180)
# <u>n</u> M2221 B <u>0</u> L <u>50</u> R <u>50</u> \n	Convert angle of joints to coordinates	\$ <u>n</u> ok X <u>100</u> Y <u>100</u> Z <u>100</u> \n
# <u>n</u> M2222 X <u>100</u> Y <u>100</u> Z <u>100</u> P <u>0</u> \n	Check if it can reach,P1 polar, P0 Cartesian coordinates	\$ <u>n</u> ok V <u>1</u> \n (1 reachable, 0 unreachable)
# <u>n</u> M2231 V <u>1</u> \n	pump V1 working, V0 stop	\$ <u>n</u> ok \n or \$ <u>n</u> Ex\n (refer to Err output)
# <u>n</u> M2232 V <u>1</u> \n	gripper V1 close, V0 open	\$ <u>n</u> ok \n or \$ <u>n</u> Ex\n (refer to Err output)
# <u>n</u> M2234 V <u>1</u> \n	Enable/disable Bluetooth (1:enable, 0:disable)	\$ <u>n</u> ok\n
# <u>n</u> M2240 N <u>1</u> V <u>1</u> \n	Set the digital IO output	\$ <u>n</u> ok \n or \$ <u>n</u> Ex\n (refer to Err output)

# <u>n</u> M2241 N <u>1</u> V <u>1</u> \n	Set the digital IO direction (V1 Output; V0 Input;)	\$ <u>n</u> ok \n
# <u>n</u> M2245 V <u>btname</u> \n	Set the name of Bluetooth, 11 letters limited (Do not add # <u>n</u> in this command)	\$ <u>n</u> ok \n
# <u>n</u> M2304 P <u>0</u> \n	Please check the Grove modules below	
# <u>n</u> M2305 P <u>0</u> N <u>1</u> \n	Please check the Grove modules below	
# <u>n</u> M2306 P <u>0</u> V <u>1000</u> \n	Please check the Grove modules below	
# <u>n</u> M2307 P <u>0</u> V <u>1</u> \n	Please check the Grove modules below	
# <u>n</u> M2400 S <u>0</u> \n	Set the mode of arm (0:Normal 1:Laser 2:3D printing 3:Universal holder)	\$ <u>n</u> ok \n
# <u>n</u> M2401\n	Set the current position into the reference position	\$ <u>n</u> ok \n
# <u>n</u> M2410\n	Set the height zero point	\$ <u>n</u> ok \n
# <u>n</u> M2411 S <u>100</u> \n	Set the offset of end-effector (mm)	\$ <u>n</u> ok \n
# <u>n</u> M2500\n	Switch the uart0 to uart2 for external TTL uart communication (For example OpenMV)	\$ <u>n</u> ok \n
Querying Command (parameters are in underline)		
# <u>n</u> P2200\n	Get the current angle of	\$ <u>n</u> ok B <u>50</u> L <u>50</u> R <u>50</u> \n

	joints	
# <u>n</u> P2201\n	Get the device name	\$ <u>n</u> ok V <u>3.2</u> \n
# <u>n</u> P2202\n	Get the hardware version	\$ <u>n</u> ok V <u>1.2</u> \n
# <u>n</u> P2203\n	Get the software version	\$ <u>n</u> ok V <u>3.2</u> \n
# <u>n</u> P2204\n	Get the API version	\$ <u>n</u> ok V <u>3.2</u> \n
# <u>n</u> P2205\n	Get the UID	\$ <u>n</u> ok V <u>0123456789AB</u> \n
# <u>n</u> P2206 N <u>0</u> \n	Get the angle of number 0 joint (0~2)	\$n ok V <u>80</u> \n
# <u>n</u> P2220\n	Get current coordinates	\$ <u>n</u> ok X <u>100</u> Y <u>100</u> Z <u>100</u> \n
# <u>n</u> P2221\n	Get current polar coordinates	\$ <u>n</u> ok S <u>100</u> R <u>90</u> H <u>80</u> \n
# <u>n</u> P2231\n	Get the status of pump	\$ <u>n</u> ok V <u>1</u> \n (0 stop, 1 working, 2 grabbing things)
# <u>n</u> P2232\n	Get the status of gripper	\$ <u>n</u> ok V <u>1</u> \n (0 stop, 1 working, 2 grabbing things)
# <u>n</u> P2233\n	Get the status of limited switch	\$ <u>n</u> ok V <u>1</u> (1 triggered, 0 untriggered)
# <u>n</u> P2234\n	Get the status of power connection	\$ <u>n</u> ok V <u>1</u> (1 connected, 0 unconnected)
# <u>n</u> P2240 N <u>1</u> \n	Get the status of digital IO	\$ <u>n</u> ok V <u>1</u> \n (1 High, 0 Low)
# <u>n</u> P2241 N <u>1</u> \n	Get the status of analog IO	\$ <u>n</u> ok V <u>295</u> \n (return the data of ADC)
# <u>n</u> P2242\n	Get the default value of AS5600 in each joint	\$ <u>n</u> ok B <u>2401</u> L <u>344</u> R <u>1048</u> \n
# <u>n</u> P2400\n	Check current status	\$ <u>n</u> ok V <u>1</u> \n (0: normal; 1: laser;

		2: 3D printing; 3: Universal holder;)
Ticking feedback		
@1	Ready	
@3	Timed feedback , " M2120"	
@4 N <u>0</u> V <u>1</u> \n	Report the button event. N: 0 = Menu button, 1 = Play button V: 1 = Click, 2 = Long Press	
@5 V <u>1</u> \n	Report event of power connection	
@6 N <u>0</u> V <u>1</u> \n	Report event of limit switch in end-effector	
@7 temp error	Temperature error in 3D printing	
@9 V0\n	Stop movement	
Err Output		
E20	Command not exist	
E21	Parameter error	
E22	Address out of range	
E23	Command buffer full	
E24	Power unconnected	
E25	Operation failure	

Grove No.	Module	Commands	Description	Support Ports	Return
1	Chainable RGB LED	# <u>n</u> M2304 P <u>3</u> \n	Deinit	3, 4, 5	\$ <u>n</u> ok\n
		# <u>n</u> M2305 P <u>3</u> N <u>1</u> V <u>2</u> \n	Init Module 1 in Port <u>3</u> . V is the number of LEDs chained.	3, 4, 5	\$ <u>n</u> ok\n or E25 init fail
		# <u>n</u> M2307 P <u>3</u> V <u>0</u> R228 G128 B100\n	Set the color of <u>0</u> th LED	3, 4, 5	\$ <u>n</u> ok\n
2	Button	# <u>n</u> M2304 P <u>3</u> \n	Deinit	3, 4, 5	
		# <u>n</u> M2305 P <u>3</u> N <u>2</u> \n	Init Module 2 in Port <u>3</u>	3, 4, 5	
		Press down			@11 P <u>3</u> N <u>2</u> V <u>0</u> \n
		Click			@11 P <u>3</u> N <u>2</u> V <u>1</u> \n
		Long pressed			@11 P <u>3</u> N <u>2</u> V <u>2</u> \n
3	Slide Potentiometer	# <u>n</u> M2304 P <u>1</u> \n	Deinit	1, 2	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>1</u> N <u>3</u> \n	Init Module 3 in Port1	1, 2	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>1</u> V <u>1000</u> \n	Set report interval (ms)	1, 2	@11 P <u>1</u> N <u>3</u> V583\n
4	Vibration Motor	# <u>n</u> M2304 P <u>3</u> \n	Deinit	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>3</u> N <u>4</u> \n	Init Module 4 in Port <u>3</u>	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2307 P <u>3</u> V <u>1</u> \n	V1: turn on; V0: turn off	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
5	Light Sensor	# <u>n</u> M2304 P <u>1</u> \n	Deinit	1, 2, 13	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>1</u> N <u>5</u> \n	Init Module 5 in Port <u>1</u>	1, 2, 13	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>1</u> V <u>1000</u> \n	Set report interval (ms)	1, 2, 13	@11 P <u>1</u> N <u>5</u> V583\n
6		# <u>n</u> M2304 P <u>1</u> \n	Deinit	1, 2, 13	\$ <u>n</u> ok \n

	Angle Sensor	# <u>n</u> M2305 P <u>1</u> N6\n	Init Module 6 in Port <u>1</u>	1, 2, 13	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>1</u> V <u>1000</u> \n	Set report interval (ms)	1, 2, 13	@11 P <u>1</u> N6 V583\n
7	Air Quality Sensor	# <u>n</u> M2304 P <u>1</u> \n	Deinit	1, 2, 13	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>1</u> N7\n	Init Module 7 in Port <u>1</u>	1, 2, 13	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>1</u> V <u>1000</u> \n	Set report interval (ms)	1, 2, 13	@11 P <u>1</u> N7 V583\n
8	Sound Sensor	# <u>n</u> M2304 P <u>1</u> \n	Deinit	1, 2, 13	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>1</u> N8\n	Init Module 8 in Port <u>1</u>	1, 2, 13	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>1</u> V <u>1000</u> \n	Set report interval (ms)	1, 2, 13	@11 P <u>1</u> N8 V583\n
9	6-Axis Accelerometer & Compass	# <u>n</u> M2304 P <u>0</u> \n	Deinit	0	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>0</u> N9\n	Init Module 9 in Port <u>0</u>	0	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>0</u> V <u>1000</u> \n	Set report interval (ms) XYZ is the rotation angle of each axis. H is the clockwise angle between the magnetic north and x-axis T is the clockwise angle between the magnetic north and the projection of the positive x-axis in the horizontal plane	0	@11 P <u>0</u> N9 X2.0 Y2.0 Z2.0 H2.0 T2.0\n
10	Color Sensor	# <u>n</u> M2304 P <u>0</u> \n	Deinit	0	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>0</u> N10\n	Init Module 10 in Port <u>0</u>	0	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>0</u> V <u>1000</u> \n	Set report interval (ms)	0	@11 P <u>0</u> N10 R218 G31 B128\n

11	Gesture Sensor	# <u>n</u> M2304 P <u>0</u> \n	Deinit	0	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>0</u> N11\n	Init Module 11 in Port <u>0</u>	0	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>0</u> V <u>1000</u> \n	Set report interval (ms)	0	@11 P <u>0</u> N11 V1\n 1: right 2: left 4: up 8: down 16: forward 32: backward 64: clockwise 128:counter clockwise
12	Ultrasonic	# <u>n</u> M2304 P <u>3</u> \n	Deinit	4, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>3</u> N12\n	Init Module 12 in Port <u>3</u>	4, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>3</u> V1000\n	Set report interval (ms)	4, 8, 9	@11 P <u>3</u> N12 V4\n Value in cm
13	Fan	# <u>n</u> M2304 P <u>4</u> \n	Deinit	4, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>4</u> N13\n	Init Module 13 in Port <u>4</u>	4, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2307 P <u>4</u> V120\n	Set Fan speed(0~255)	4, 8, 9	\$ <u>n</u> ok \n
14	Electromagnet	# <u>n</u> M2304 P <u>3</u> \n	Deinit	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>3</u> N14\n	Init Module 14 in Port <u>3</u>	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2307 P <u>3</u> V1\n	1:turn on 0: turn off	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
15		# <u>n</u> M2304 P <u>0</u> \n	Deinit	0	\$ <u>n</u> ok \n

	Temperature & Humidity	# <u>n</u> M2305 P <u>0</u> N15\n	Init Module 15 in Port <u>0</u>	0	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>0</u> V <u>1000</u> \n	Set report interval (ms)	0	@11 P <u>0</u> N15 T23.3 H82.2\n
16	PIR Sensor	# <u>n</u> M2304 P <u>3</u> \n	Deinit	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>3</u> N16\n	Init Module 16 in Port <u>3</u>	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>3</u> V1000\n	Set report interval (ms)	3, 4, 5, 8, 9	@11 P <u>3</u> N16 V1\n 1: Motion detected 0: no motion detected
17	1602 LCD	# <u>n</u> M2304 P <u>0</u> \n	Deinit	0	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>0</u> N17\n	Init Module 17 in Port <u>0</u>	0	\$ <u>n</u> ok \n
		# <u>n</u> M2307 P <u>0</u> R128 G120 B10\n	Set backlight color	0	\$ <u>n</u> ok \n
		# <u>n</u> M2307 P <u>0</u> T <u>0</u> \n	0: turn off display 1: turn on display 2: clear	0	\$ <u>n</u> ok \n
		# <u>n</u> M2307 P <u>0</u> V <u>0</u> S <u>Test</u> \n	V(0~1): row selected S: the display string	0	\$ <u>n</u> ok \n
18	Line Finder	# <u>n</u> M2304 P <u>3</u> \n	Deinit	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>3</u> N18\n	Init Module 18 in Port <u>3</u>	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>3</u> V1000\n	Set report interval (ms)	3, 4, 5, 8, 9	@11 P <u>3</u> N18 V1\n 0: object detected

					1: no object detected
19	Infrared Reflective Sensor	# <u>n</u> M2304 P <u>3</u> \n	Deinit	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>3</u> N19\n	Init Module 19 in Port <u>3</u>	3, 4, 5, 8, 9	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>3</u> V1000\n	Set report interval (ms)	3, 4, 5, 8, 9	@11 P <u>3</u> N19 V1\n 1: object detected 0: no object detected
20	EMG Detector	# <u>n</u> M2304 P <u>1</u> \n	Deinit	1, 2, 13	\$ <u>n</u> ok \n
		# <u>n</u> M2305 P <u>1</u> N20\n	Init Module 20 in Port <u>1</u>	1, 2, 13	\$ <u>n</u> ok \n
		# <u>n</u> M2306 P <u>1</u> V <u>1000</u> \n	Set report interval (ms)	1, 2, 13	@11 P <u>1</u> N20 V583\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](#)

[uArm Technical Support](#)

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 Add detail steps of Arduino upload	Tony
1.0.3	Add the relationship of left/right motor with the upper and lower arm	Tony
1.0.4	Modify several commands of Gcode	David
1.0.5	Modify several commands of Gcode	Daniel
1.0.6	Modify the content of switching to 2nd UART Add the dimensions of the base of uArm Swift Pro V1.1	Daniel