UFACTORY

UARM Swift

Quick-Start Guide

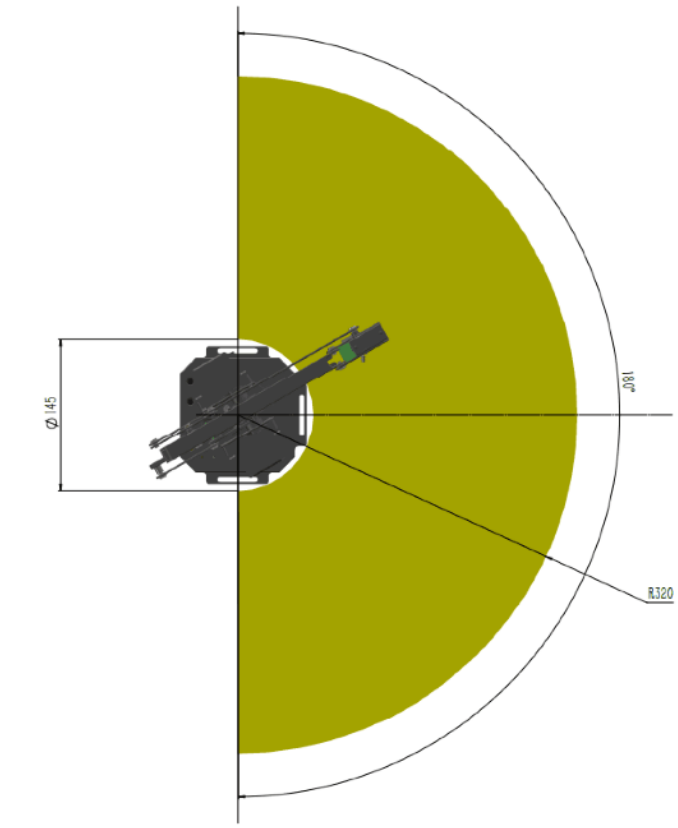
V1.0
May. 2017

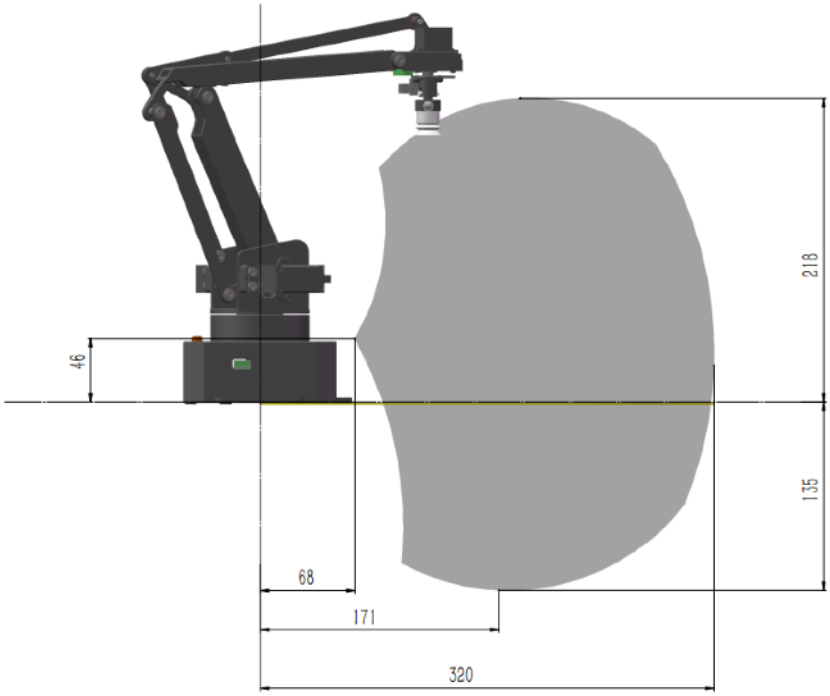
⚠ This is the quick-start guide for **uArm Swift.**
The quick-start of using **uArm Swift Pro** will be explained in a separate document.

Contents

Safety Instructions

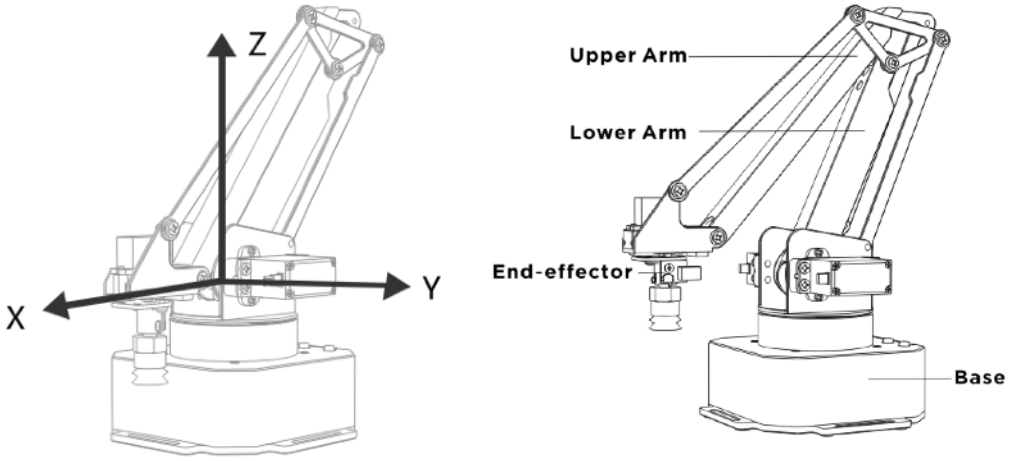
1. Please **don't put your hands between the arms** when uArm is moving.
2. Please use the **official power supply** for safety reasons.
3. Please **clear a space** for uArm, in case of knocking down anything.



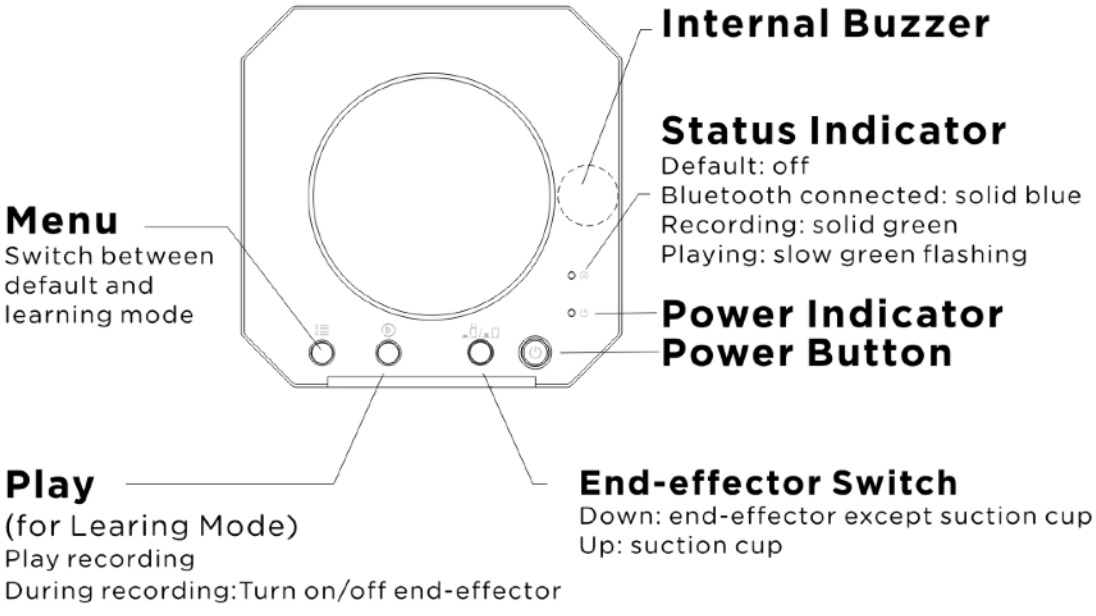


Product Overview

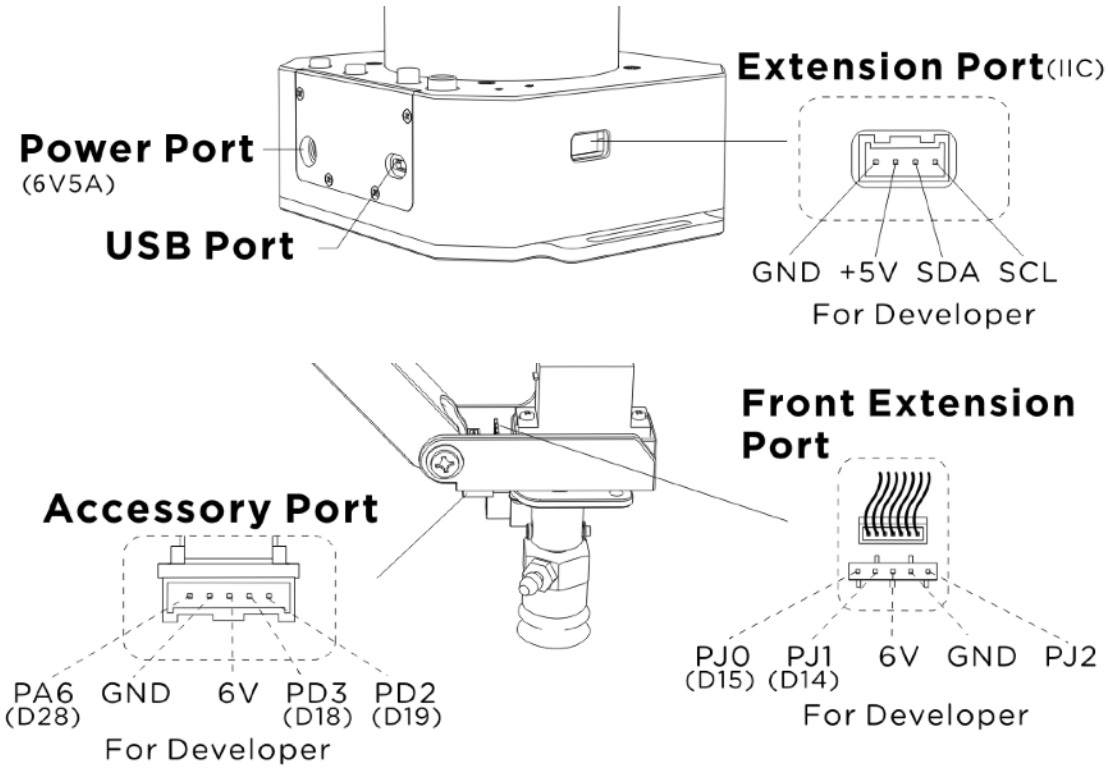
1. Reference Frame



2. Buttons & Indicator Lights



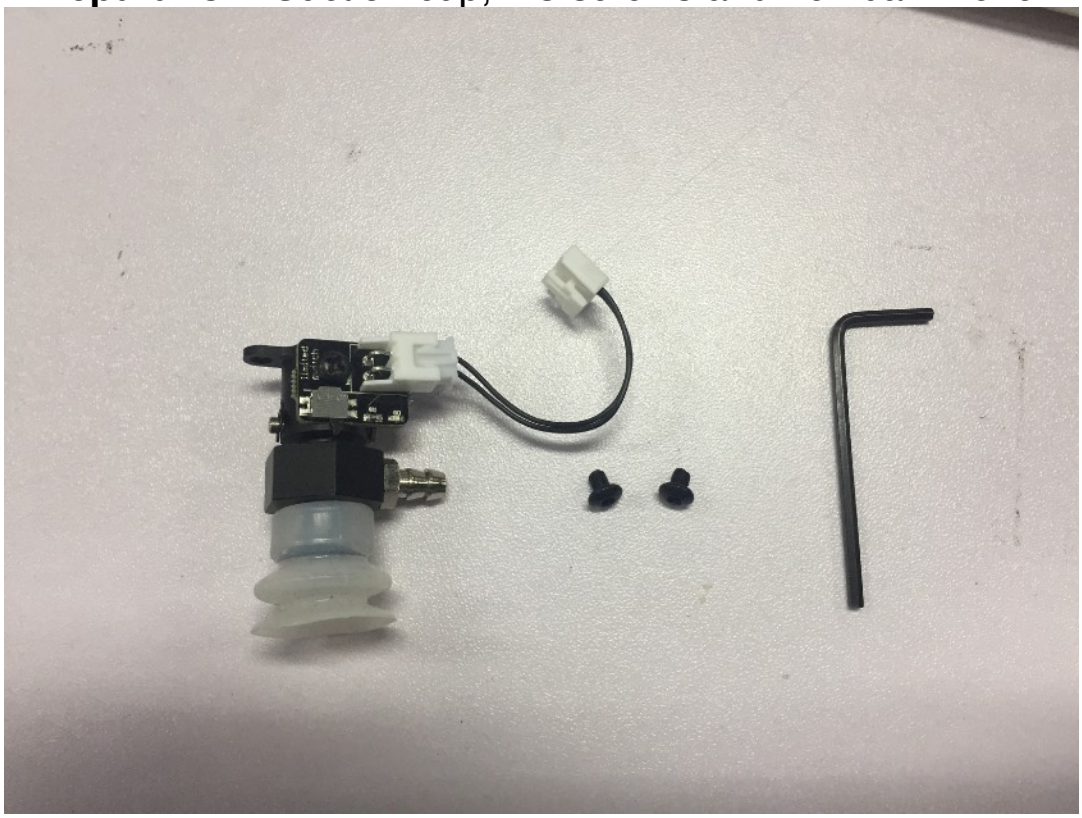
3. Extension Description



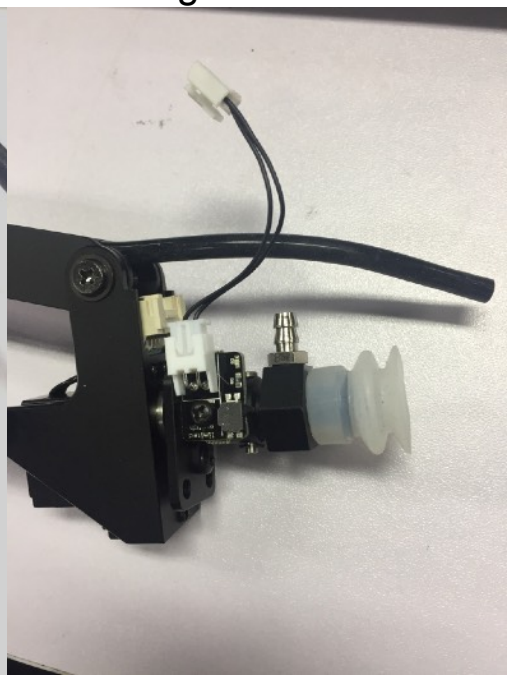
End-Effectors Installation

1. Suction Cup (Default)

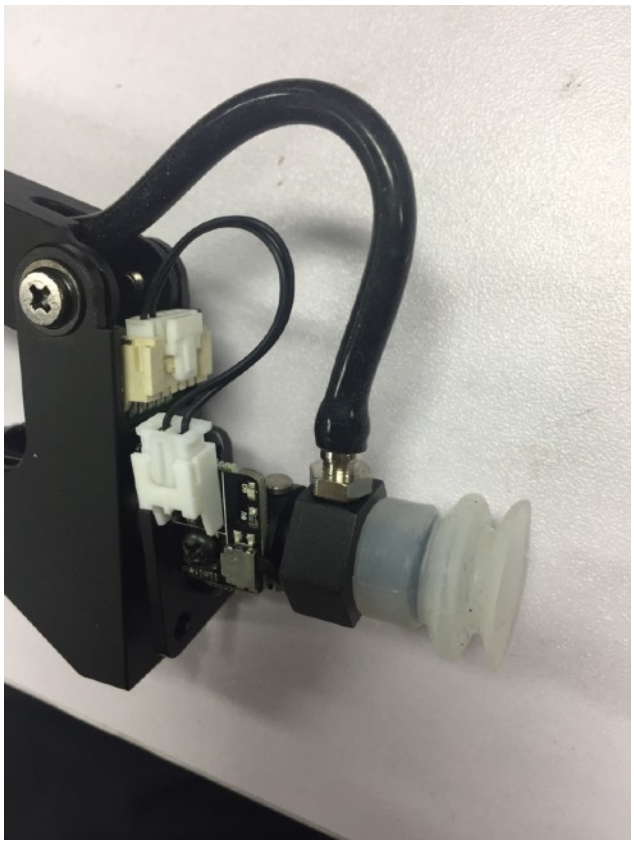
Preparation: Suction cup, M3 screws and hex bar wrench



Step 1: Fix the suction cup to the front mounting block with the M3 screws.



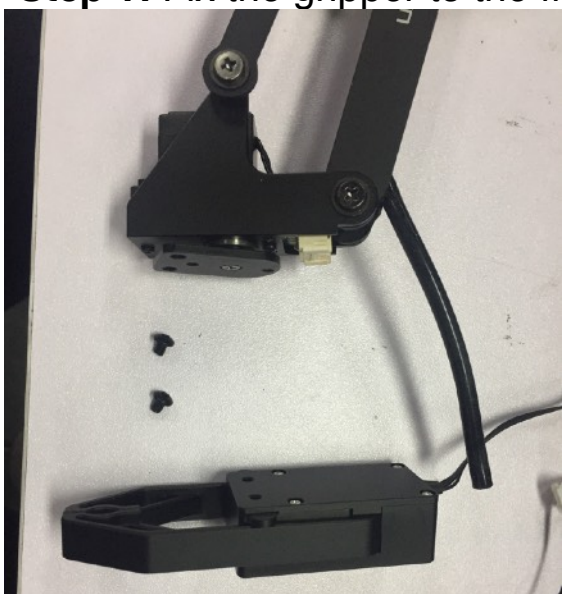
Step 2: Wiring the limited switch and silicon tube



Note: Before unplugging the wire, press the locker of connector and then unplug it.

2. Swift Gripper

Step 1: Fix the gripper to the front mounting block with 2 screws



Step 2: Plug the wire of gripper

Note: Because there is no need to use the silicon tube for suction cup, we could use the velcro to fix the tube with the upper arm.

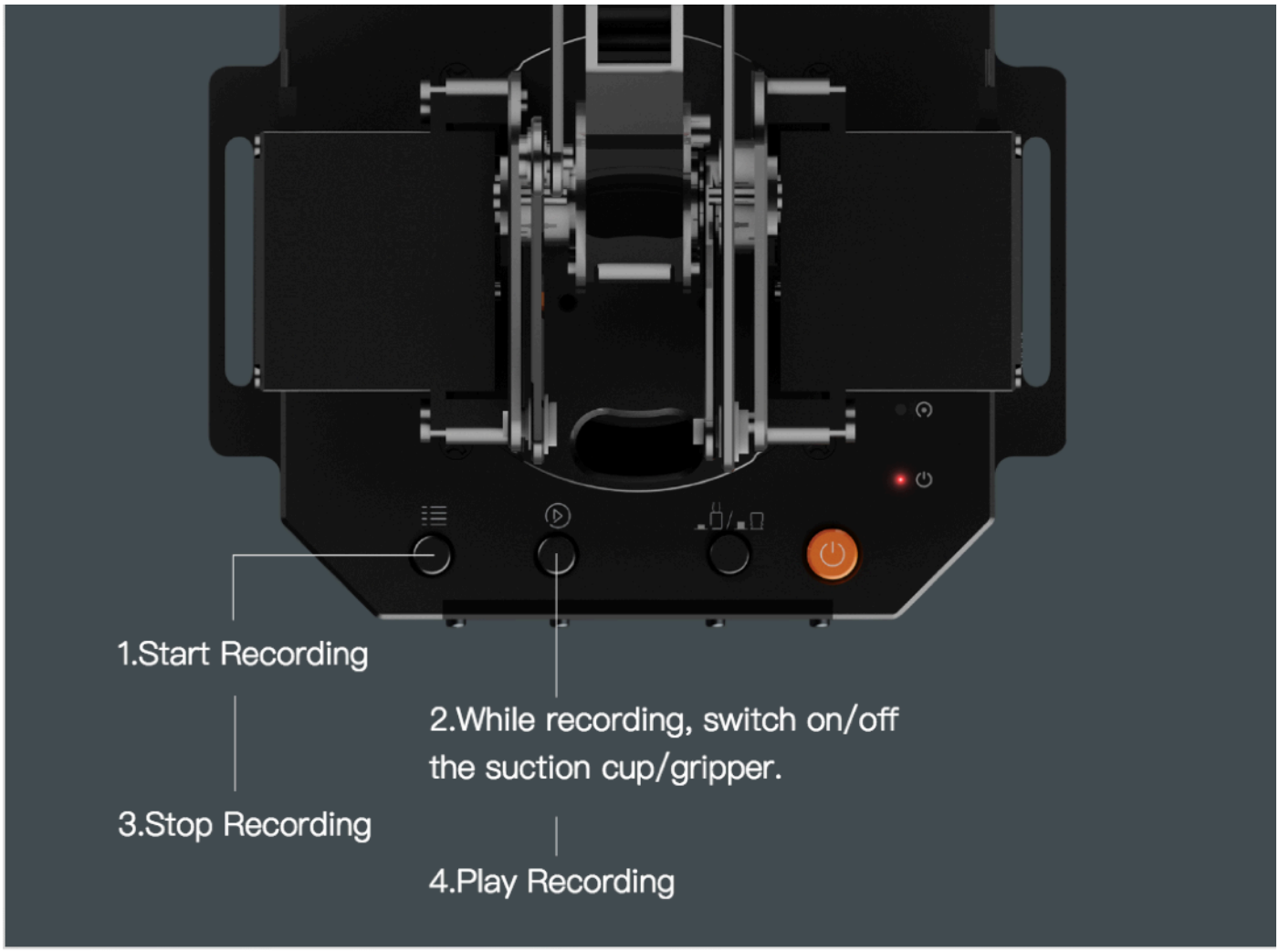
3. Swift Universal Holder

Step1 : Fix the universal holder to the front mounting block.

Step2: Install the pen to the holder.

Offline Learning Mode

Use buttons on the base to “teach” uArm by hand.



Teach:

1.Start learning mode. Press the ☰ once ,and the status indicator turns green.

2.Teach the robot manually. Press the ▷ once to turn on the end-effector, again to turn off.

3.Finish the learning process. Press ☰ once, and the status indicator turns off.

Play:

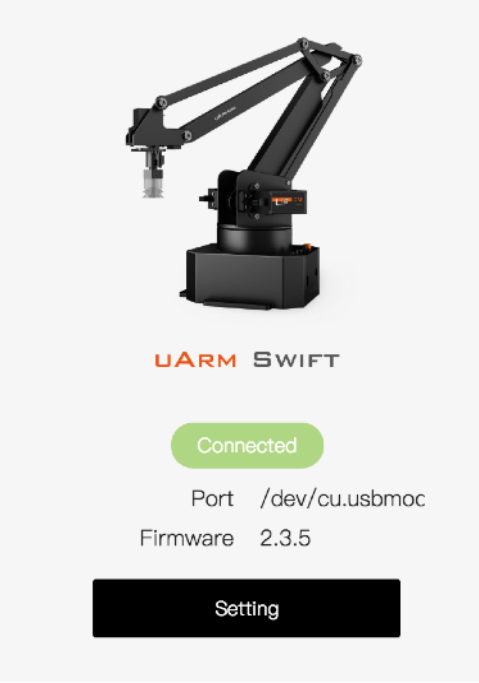
1.One-time playback : Press ▷ once, or Loop playback: press ▷ & hold for 2 seconds.

2.The status indicator starts flashing green slowly.

3.Press ▷ once to stop playing.

Software: uArm Studio (Win/Mac)

1. **Download uArm Studio from:**
   http://www.ufactory.cc/#/en/support/



2. **Device Connection**

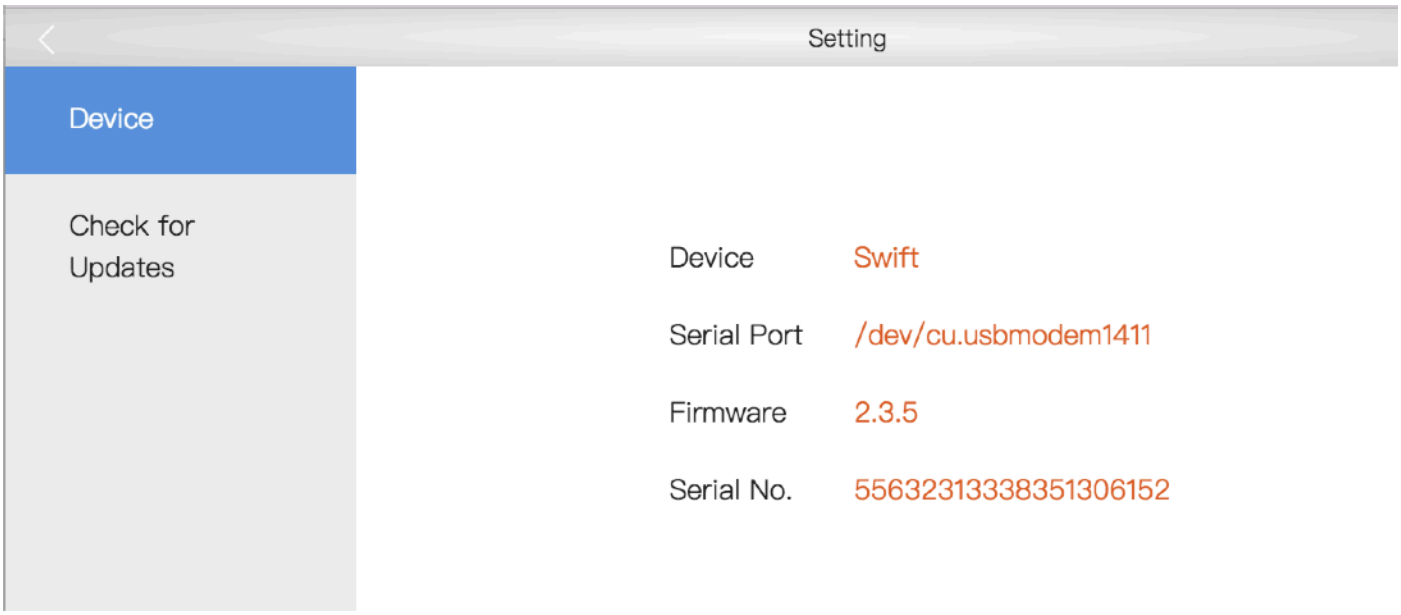
1）Plug in the power cable.

2）Press down the power button.

3）Connect uArm to your computer via USB.

Once connected, you'll see the connected status on home page.

More info is displayed in "Setting".



3. Teach & Play: Learning Mode

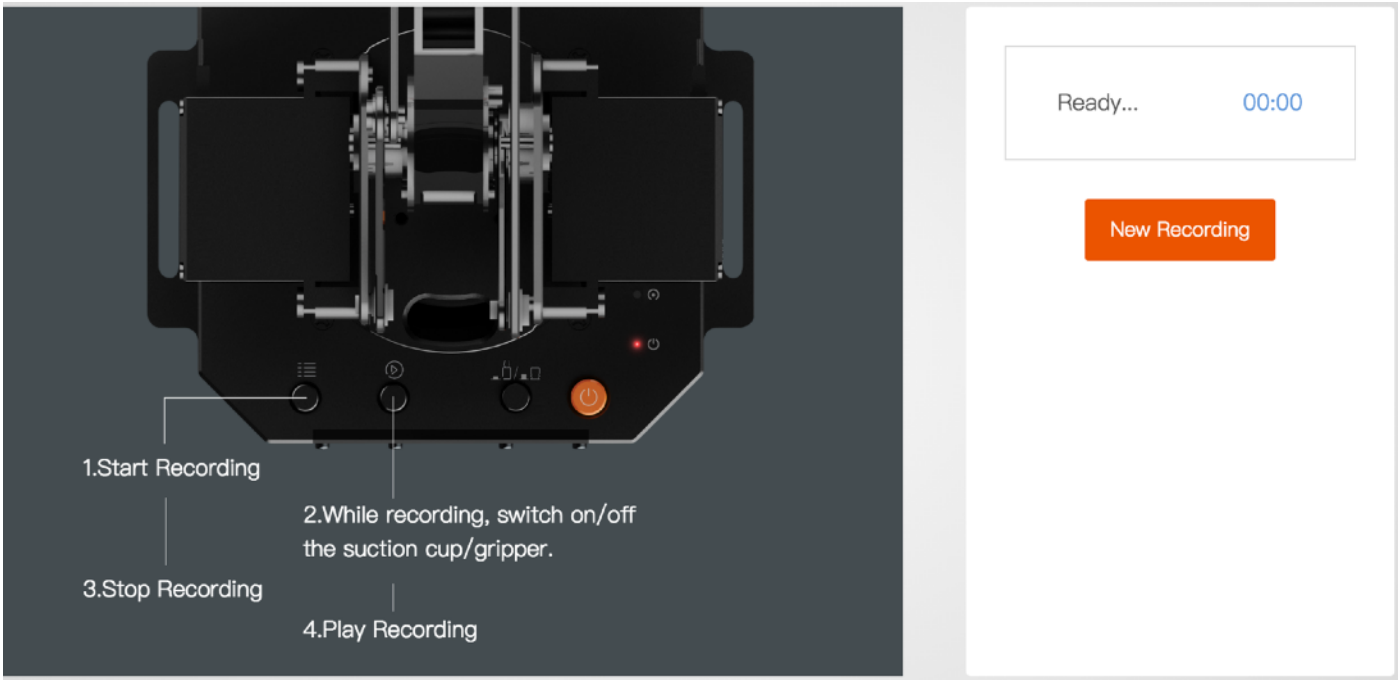
What is Teach & Play?

Teach uArm by hand, and then replay the recording anytime.

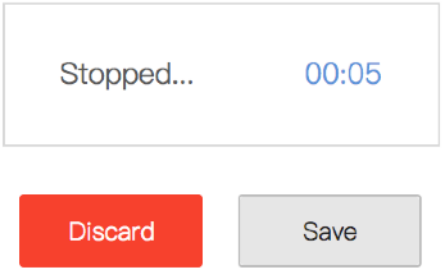
How?

1) Make a recording

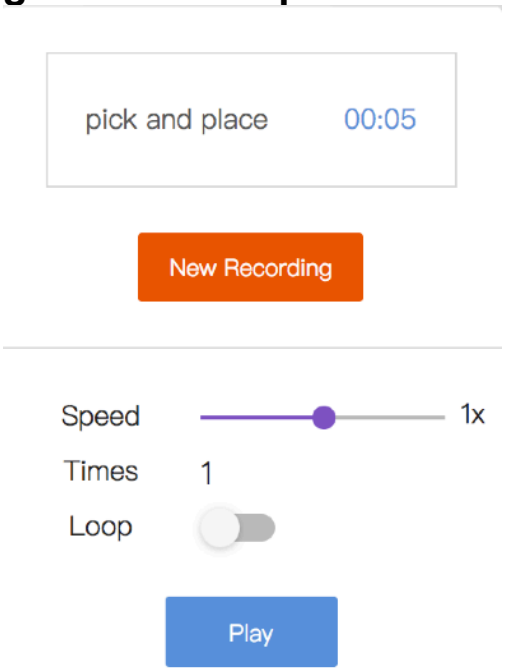
- Click the “New Recording” button to start “teaching”, OR,
- Use the buttons on the base (usage of the buttons is the same as that under “*Offline Learning Mode*”).



2) Save your recording



3) Replay the recording in different speed and times



What makes **"Teach & Play"** different from **"Offline Learning Mode"**?

1) No time limit while "teaching" with uArm Studio.
2) You may save, export your recordings and import recordings made by others.
3) You may apply your recording in Blockly (visual programming interface, which is explained up next).

4. Blockly: Visual Programming

What is Blockly?

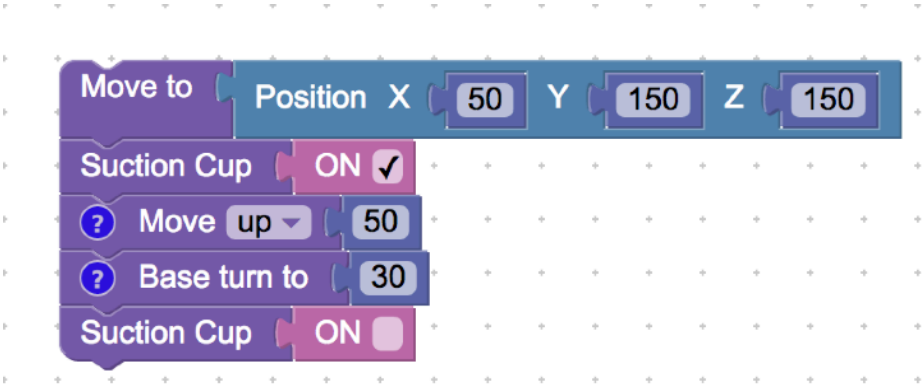
Blockly in uArm Studio is a visual programming interface specially designed for controlling uArm.

Getting Started

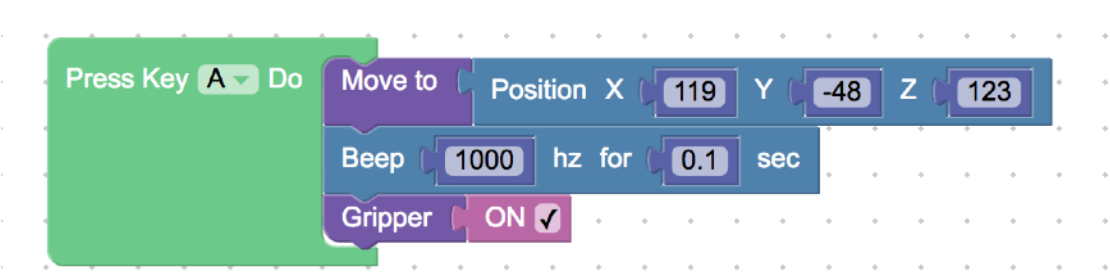
Three **"missions"** are prepared to get you through Blockly quickly. Please try them out!

What can you do with Blockly?

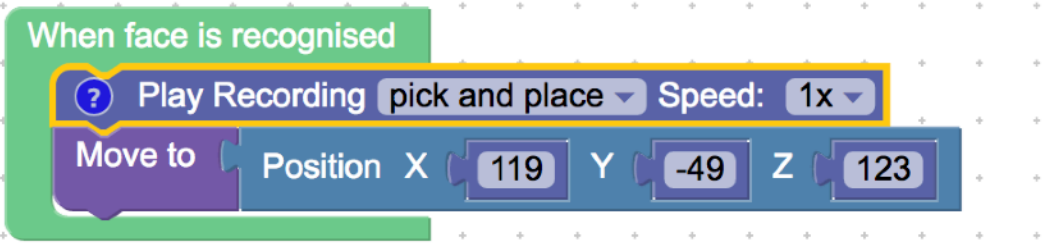
1) Control uArm's basic movements



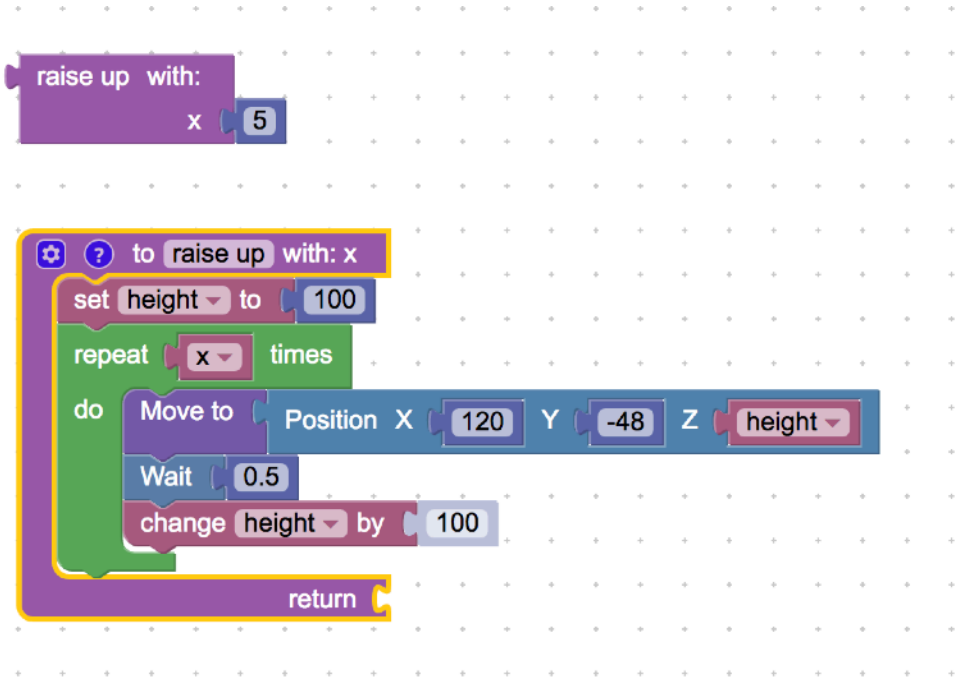
2) Change events (i.e. how you trigger commands)



3) Apply recorded movements



4) Dig deeper into programming (functions, variables, etc.)



5. Mouse & Keyboard Control

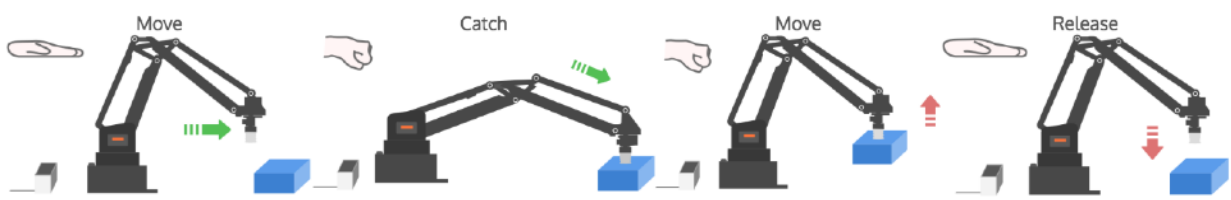
This section is for real-time controlling with mouse and keyboard.
You may use your mouse and keyboard shortcuts at the same time.

6. Gesture Control: Leap Motion

Control uArm with your hand motion, via Leap Motion, a third-party device for hand tracking.

If you want to try it out, you will need:
1) a Leap Motion Controller
2) Driver for Leap Motion Controller installed

Start your real-time control with hand motion:



For Developers

1. Library

uArm Swift - Arduino Library

* Libraries for Python & ROS users will be released soon.

2. Communication 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;

GCodeCommand	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) 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)	$*n* OK \nor$*n* E*x* \n(refer to Err output)
#*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)	$*n* OK \nor$*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)	$*n* OK \nor$*n* E*x* \n(refer to Err output)
#*n* G2204 X*10* Y*10* Z*10* F*1000*\n	Relative displacement	$*n* OK \nor$*n* E*x* \n(refer to Err output)
#*n* G2205 S*10* R*10* H*10* F*1000*\n	Polar coordinates for relative displacement	$*n* OK \nor$*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*154.714* Y*194.915* Z*10.217*\n
#*n* M2200\n	Check if uArm is moving	$*n* OK V*1*\n(1moving,0 stop)
#*n* M2201 N*0*\n	attach motor, Nis ID of joints(0~3)	$*n* OK \nor$*n* E*x* \n(refer to Err output)
#*n* M2202 N*0*\n	Detach motor, Nis ID of joints(0~3)	$*n* OK \nor$*n* E*x* \n(refer to Err output)
#*n* M2203 N*0*\n	Check if the motor is attached, Nis 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, Tis time (ms)	$*n* OK \nor$*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), Ais 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 (Bjoint 0,Ljoint 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,P1polar,P0Cartesian coordinates	$*n* OK V*1*\n (1 reachable,0 unreachable)

#*n* M2231 V*1*\n	pump V1working, V0stop	**$*n* OK \n**or**$*n* E*x* \n**(refer to Err output)
#*n* M2232 V*1*\n	gripper V1close, V0open	**$*n* OK \n**or**$*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**$*n* E*x* \n**(refer to Err output)
M2245 V*btname*\n	Set the name of Bluetooth, 11 letters limited (Do not add **#*n*** in this command)	**OK \n**
Querying Command(parameters are in underline)		
#*n* P2200\n	Get the current angle of joints	**$*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	**$*n* OK V*1*** (1 triggered, 0untriggered)
#*n* P2234\n	Get the status of power connection	**$*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)
Ticking (Tip Sensor of Suction Cup) feedback		
@1	Ready	
@3	Time cycle of feedback , ”M2120”	
@4 N*0* V*1*\n	Report the button event. N: 0 = Menu button, 1 = Play button V: 1 = Click, 2 = Long Press	
@5 V*1*\n	Reportevent of power connection	
@6 N*0* V*1*\n	Report event of limit switch in end-effector	
Err Output		

E20	Command not exist	
E21	Parameter error	
E22	Address out of range	
E23	Command buffer zone is full	
E24	Power unconnected	
E25	Operation failure	

uArm Community

UFACTORY Official Forum

uArm User Facebook Group

Ask for Help