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 ........................................3

Product Overview........................................4

End-Effectors Installation.........................6

Offline Learning Mode ..............................10

Software: uArm Studio (Win/Mac)..........11

For Developers .................................17

uArm Community ..............................20
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

- **Internal Buzzer**
- **Status Indicator**
  - Default: off
  - Bluetooth connected: solid blue
  - Recording: solid green
  - Playing: slow green flashing
- **Power Indicator**
- **Power Button**
- **Menu**
  - Switch between default and learning mode
- **Play**
  - (for Learning Mode)
  - Play recording
  - During recording: Turn on/off end-effector
- **End-effector Switch**
  - Down: end-effector except suction cup
  - Up: suction cup
3. Extension Description

Power Port (6V5A)

USB Port

Extension Port (IIC)
- GND
- +5V
- SDA
- SCL
- For Developer

Accessory Port
- PA6 (D28)
- GND
- 6V
- PD3 (D18)
- PD2 (D19)
- For Developer

Front Extension Port
- PJ0 (D15)
- PJ1 (D14)
- 6V
- GND
- PJ2
- For Developer
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.

1. Start Recording
2. While recording, switch on/off the suction cup/gripper.
3. Stop Recording
4. Play Recording

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”.

![uArm Swift connected status](image.png)
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.


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;
1. 
   `#n` is used for the debug, if you don’t want to use it please remove it directly.
   (For Example: G2202 N0 V90\n)
2. ‘\n’ is the symbol of line feed.

<table>
<thead>
<tr>
<th>GCodeCommand</th>
<th>Description</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>#n G0 X100 Y100 Z100 F1000\n</td>
<td>Move to XYZ(mm), F is speed(mm/min)</td>
<td>$n$ OK \n or $n$ Ex \n (refer to Err output)</td>
</tr>
<tr>
<td>#n G2201 S100 R90 H80 F1000\n</td>
<td>Polar coordinates, S is stretch(mm), R is rotation(degree), H is height(mm), F is speed(mm/min)</td>
<td>$n$ OK \n or $n$ Ex \n (refer to Err output)</td>
</tr>
<tr>
<td>#n G2202 N0 V90\n</td>
<td>Move the motor to the position, N is ID of joints(0<del>3), V is angle(0</del>180)</td>
<td>$n$ OK \n or $n$ Ex \n (refer to Err output)</td>
</tr>
<tr>
<td>#n G2204 X10 Y10 Z10 F1000\n</td>
<td>Relative displacement</td>
<td>$n$ OK \n or $n$ Ex \n (refer to Err output)</td>
</tr>
<tr>
<td>#n G2205 S10 R10 H10 F1000\n</td>
<td>Polar coordinates for relative displacement</td>
<td>$n$ OK \n or $n$ Ex \n (refer to Err output)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setting Command</th>
<th>Parameters are in underline</th>
</tr>
</thead>
<tbody>
<tr>
<td>#n M17\n</td>
<td>Attach all the joint motors</td>
</tr>
<tr>
<td>#n M2019\n</td>
<td>Detach all the joint motors</td>
</tr>
<tr>
<td>#n M2120 V0.2\n</td>
<td>Set time cycle of feedback, return Cartesian coordinates, V is time(seconds)</td>
</tr>
<tr>
<td>#n M2200\n</td>
<td>Check if uArm is moving</td>
</tr>
<tr>
<td>#n M2201 N0\n</td>
<td>attach motor, Nis ID of joints(0~3)</td>
</tr>
<tr>
<td>#n M2202 N0\n</td>
<td>Detach motor, Nis ID of joints(0~3)</td>
</tr>
<tr>
<td>#n M2203 N0\n</td>
<td>Check if the motor is attached, Nis ID of joints(0~3)</td>
</tr>
<tr>
<td>#n M2210 F1000 T200\n</td>
<td>buzzer, F is frequency, Tis time (ms)</td>
</tr>
<tr>
<td>#n M2211 N0 A200 T\n</td>
<td>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)</td>
</tr>
<tr>
<td>#n M2212 N0 A200 T1 V10\n</td>
<td>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</td>
</tr>
<tr>
<td>#n M2213 V0\n</td>
<td>Default function of base buttons (0 false, 1 true)</td>
</tr>
<tr>
<td>#n M2220 X100 Y100 Z100\n</td>
<td>Convert coordinates to angle of joints</td>
</tr>
<tr>
<td>#n M2221 B0 L50 R50\n</td>
<td>Convert angle of joints to coordinates</td>
</tr>
<tr>
<td>#n M2222 X100 Y100 Z100 P0\n</td>
<td>Check if it can reach, P1polar, P0Cartesian coordinates</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>#n M2231 V \n</td>
<td>pump V1 working, V0 stop</td>
</tr>
<tr>
<td>#n M2232 V \n</td>
<td>gripper V1 close, V0 open</td>
</tr>
<tr>
<td>#n M2234 V \n</td>
<td>Enable/disable Bluetooth (1: enable, 0: disable)</td>
</tr>
<tr>
<td>#n M2240 N J V \n</td>
<td>Set the digital IO output</td>
</tr>
<tr>
<td>M2245 \n btname \n</td>
<td>Set the name of Bluetooth, 11 letters limited (Do not add #n in this command)</td>
</tr>
</tbody>
</table>

### Querying Command (parameters are in underline)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>#n P2200\n</td>
<td>Get the current angle of joints</td>
<td>$n OK B50 L50 R50\n</td>
</tr>
<tr>
<td>#n P2201\n</td>
<td>Get the device name</td>
<td>$n OK V3.2\n</td>
</tr>
<tr>
<td>#n P2202\n</td>
<td>Get the hardware version</td>
<td>$n OK V1.2\n</td>
</tr>
<tr>
<td>#n P2203\n</td>
<td>Get the software version</td>
<td>$n OK V3.2\n</td>
</tr>
<tr>
<td>#n P2204\n</td>
<td>Get the API version</td>
<td>$n OK V3.2\n</td>
</tr>
<tr>
<td>#n P2205\n</td>
<td>Get the UID</td>
<td>$n OK V0123456789AB\n</td>
</tr>
<tr>
<td>#n P2206 N J V \n</td>
<td>Get the angle of number 0 joint (0~2)</td>
<td>$n OK V80\n</td>
</tr>
<tr>
<td>#n P2220\n</td>
<td>Get current coordinates</td>
<td>$n OK X100 Y100 Z100\n</td>
</tr>
<tr>
<td>#n P2221\n</td>
<td>Get current polar coordinates</td>
<td>$n OK S100 R90 H80\n</td>
</tr>
<tr>
<td>#n P2231\n</td>
<td>Get the status of pump</td>
<td>$n OK V1\n (0 stop, 1 working, 2 grabbing things)</td>
</tr>
<tr>
<td>#n P2232\n</td>
<td>Get the status of gripper</td>
<td>$n OK V1\n (0 stop, 1 working, 2 grabbing things)</td>
</tr>
<tr>
<td>#n P2233\n</td>
<td>Get the status of limited switch</td>
<td>$n OK V1 (1 triggered, 0 untriggered)</td>
</tr>
<tr>
<td>#n P2234\n</td>
<td>Get the status of power connection</td>
<td>$n OK V1 (1 connected, 0 unconnected)</td>
</tr>
<tr>
<td>#n P2240 N J V \n</td>
<td>Get the status of digital IO</td>
<td>$n OK V1 (1 High, 0 Low)</td>
</tr>
<tr>
<td>#n P2241 N J V \n</td>
<td>Get the status of analog IO</td>
<td>$n OK V295\n (return the data of ADC)</td>
</tr>
</tbody>
</table>

### Ticking (Tip Sensor of Suction Cup) feedback

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@1</td>
<td>Ready</td>
</tr>
<tr>
<td>@3</td>
<td>Time cycle of feedback, &quot;M2120&quot;</td>
</tr>
<tr>
<td>@4 N J V \n</td>
<td>Report the button event. N: 0 = Menu button, 1 = Play button V: 1 = Click, 2 = Long Press</td>
</tr>
<tr>
<td>@5 V \n</td>
<td>Report event of power connection</td>
</tr>
<tr>
<td>@6 N J V \n</td>
<td>Report event of limit switch in end-effector</td>
</tr>
</tbody>
</table>

### Err Output
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E20</td>
<td>Command not exist</td>
</tr>
<tr>
<td>E21</td>
<td>Parameter error</td>
</tr>
<tr>
<td>E22</td>
<td>Address out of range</td>
</tr>
<tr>
<td>E23</td>
<td>Command buffer zone is full</td>
</tr>
<tr>
<td>E24</td>
<td>Power unconnected</td>
</tr>
<tr>
<td>E25</td>
<td>Operation failure</td>
</tr>
</tbody>
</table>

# uArm Community

[UFACTORY Official Forum](#)

[uArm User Facebook Group](#)

[Ask for Help](#)