用户手册

BY-E140 智能夹持器

Version 2.2

2021 年 9 月

SUMMARY OF DOCUMENT REVISIONS			
Rev. No.	**Date Revised**	**Section Revised**	**Revision Description**
0	2019/4	xww	Drafted
1	2020/3	xsx	2d drawings updated
2	2021/2	JZL	Update UR interface
3	2021/9	JZL	Update elite robots interface

1.0 前言

1.1 一般事项

感谢您使用BY-E140产品，本产品适配各种工业机器人和其他运动控制设备，用于食品包装、零部件抓取、科研院所实验等各种应用。使用本产品之前，请注意下面提示：

- 在使用之前，请仔细阅读该手册，在充分理解的基础上正确操作。
- 请妥善保管该手册，以便随时取阅。
- 关于该装置的基本操作，另有相关资料记载，请在使用之前一并认真阅读并理解后正确操作。

1.2 安全事项

不合规的使用可能引起潜在的危险，因此要求BY-E140的用户使用时务必评估安全风险。这些潜在的安全风险因素包括；

- 由于设置力不当引起的加紧
- 高速运行时的夹持器手指的撞击
- 安装在机器人或者移动设备上的潜在撞击

不合规的应用类型也可能引起潜在的危险。在涉及下面类型的应用时，本公司不建议使用，请谨慎评估。

- 在户外使用、涉及到有潜在化学污染或电子干扰用途、或是产品目录及手册中规定之外的用途。
- 核电动力设备、焚烧系统、铁路航空、交通工具、医疗器械、娱乐设施、安全设备以 及服从于行政机关或独立产业条例的其他设施。
- 危及生命财产安全的系统、器械、装置。
- 水电煤气供给系统、或 24小时连续运转的系统等高可靠性要求设备。

1.3 特定用语

序号	名称	接受
1	夹持器	指 BY-E140 智能夹持器件
2	控制盒	指 BY-E140 微控制单元
3	Elite	指 Elite 机器人
4	URCap	指 UR 机器人配套软件包
5	PolyScope	指 UR 机器人软件系统

2.0 硬件安装

2.1 设备明细

当收到BY-E140产品时，本产品标准包装内已经涵盖所有适配机器人对应部件。打开本产品的标准包装，开箱后如图 1开箱图所示：

1

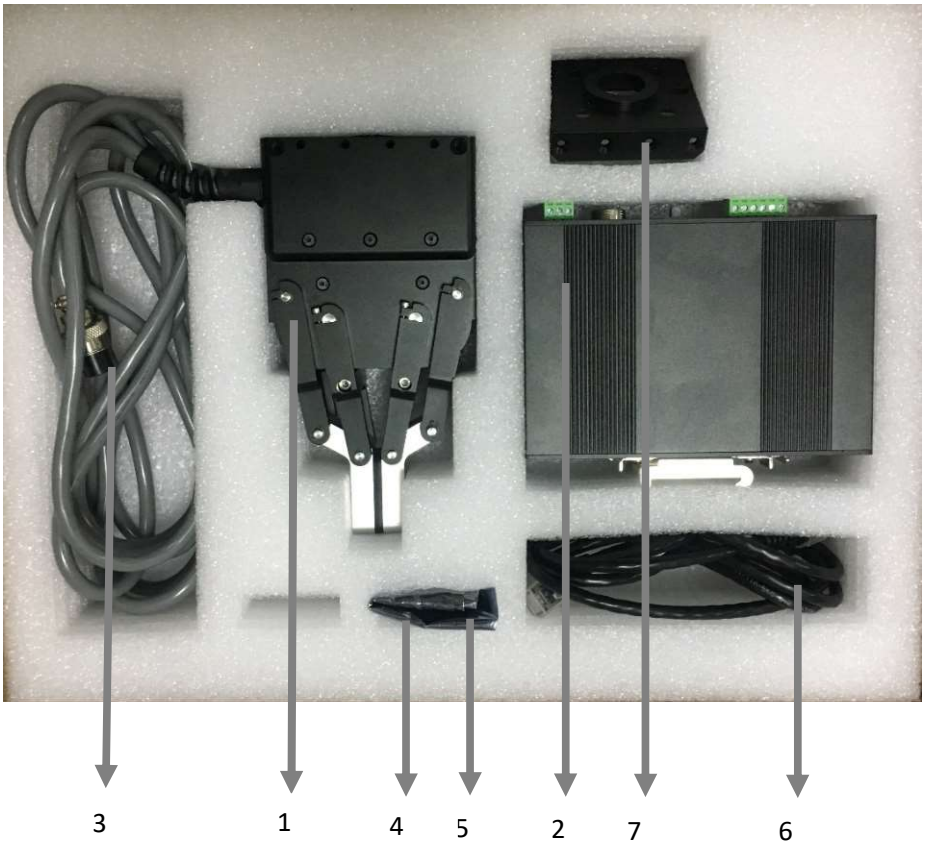

图 1 开箱图

产品明细：

序号	名称	描述	数量
1	夹持器	BY-E140 本体	1
2	控制盒	夹持器和机器人控制单元	1
3	通信电缆	用于夹持器和控制盒直接连接	1
4	螺丝	用于安置 BY-E140 到机器人末端法兰	若干
5	USB 盘	URCap 软件及用户手册	1
6	网线	控制器连接	1
7	末端法兰	用于安装至机器人手臂	1

2.2 外形图

BY-E140智能夹持器，是末端手指平行类夹持器。本体采用紧凑型设计，以最大程度的减少本体的重量，同时提高加持力，特别适合机器人末端夹持物体类使用。夹持器外形图和外形尺寸图如图 2 BY-E140外形图和图 3 BY-E140外形尺寸图所示：

图 2 BY-E140 外形图

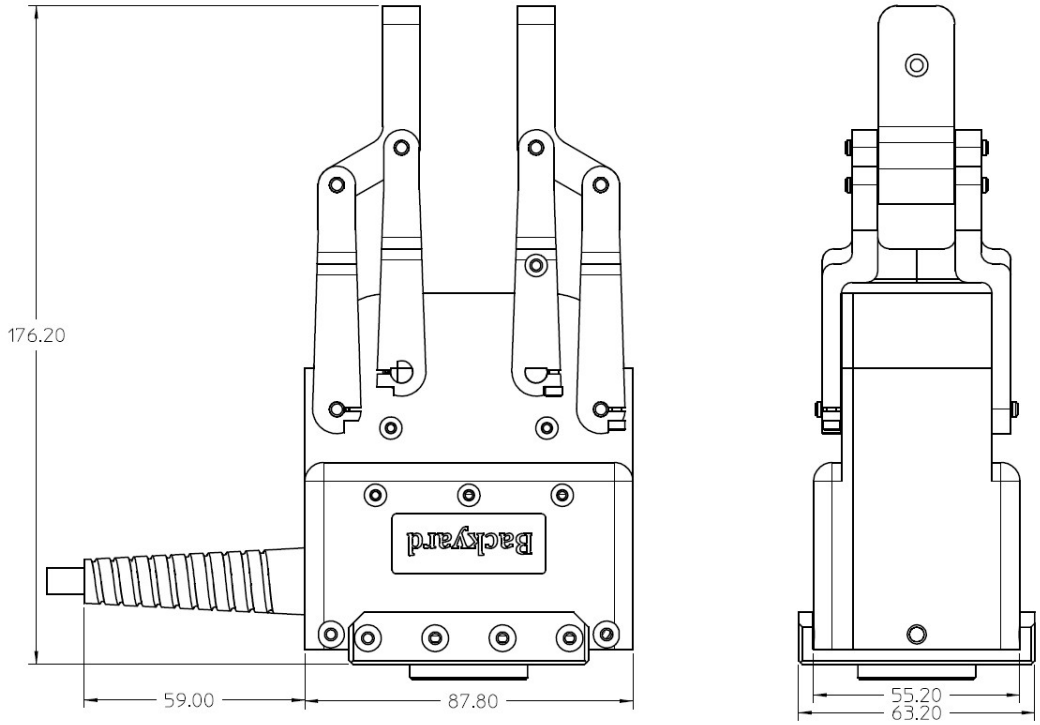

图 3 BY-E140 外形尺寸图

2.3 详细尺寸

夹持器本身已经预留法兰连接孔和配备法兰转接板。夹持器本身底部和侧面安装尺寸参考图 4 BY-E140底部和侧面安装尺寸图（自定义法兰板）。法兰转接板尺寸参考图 5 BY-E140安装法兰图。

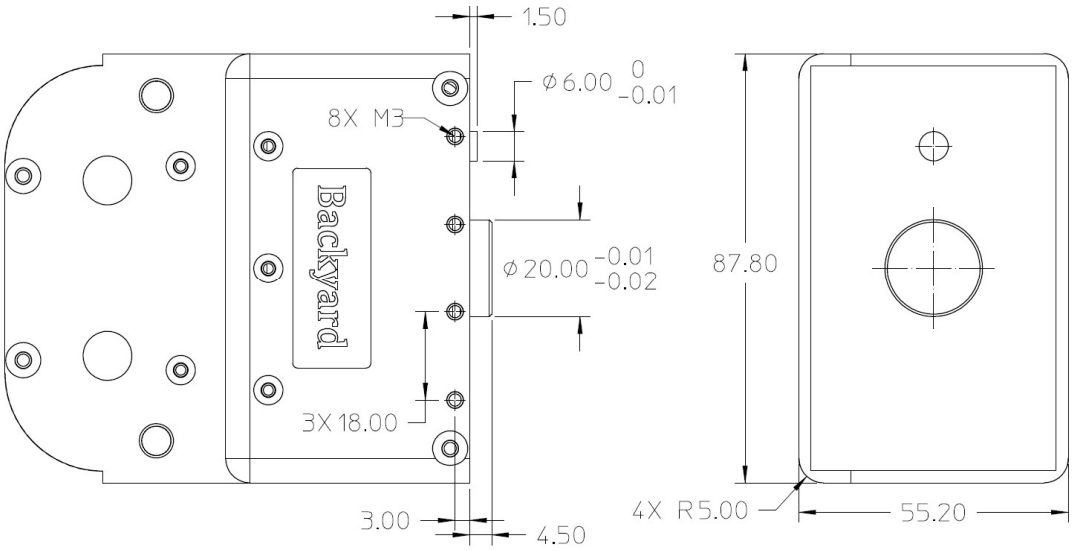

图 4 BY-E140 底部和侧面安装尺寸图（自定义法兰板）

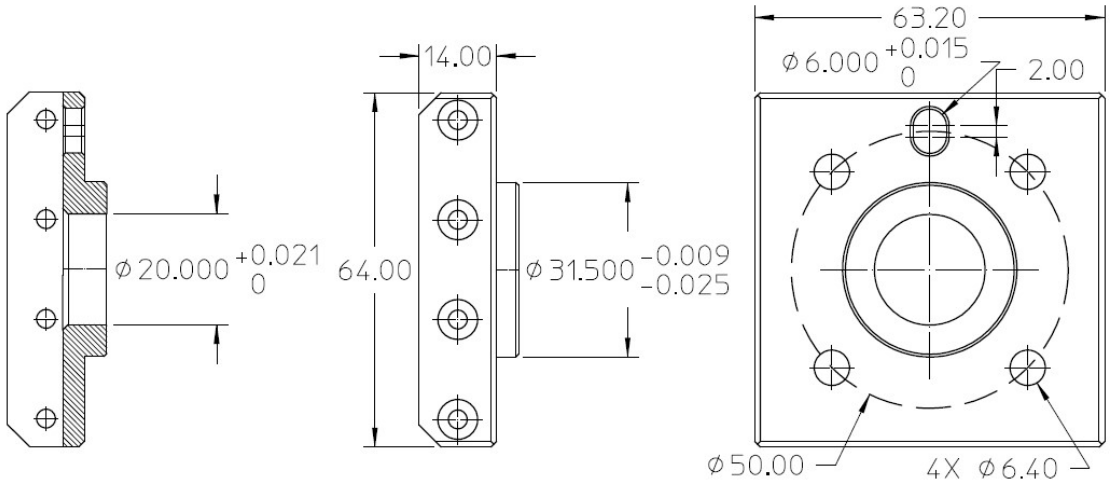

图 5 BY-E140 安装法兰图

夹持器手指安装孔已经预留多个安装孔，根据需要可以灵活调整安装位置。手指法兰尺寸参考图 6 BY-E140手指法兰图

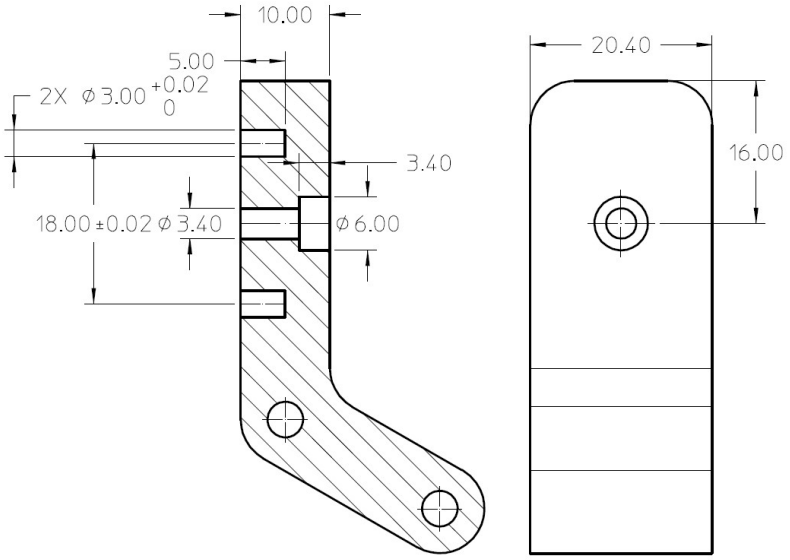

图 6 BY-E140 手指法兰图

2.4 硬件连接

2.4.1 夹持器安装

夹持器安装时，要首先把机器人位置调整到容易安装的位置。如下图机器人法兰朝上，或者侧面也同样可以。

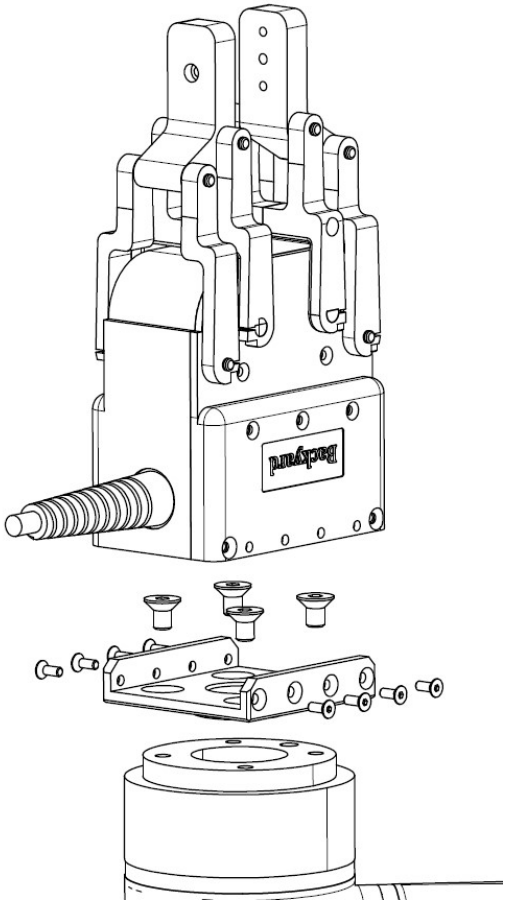

图 7 机器人安装初始位置示例

夹持器安装时分为两个步骤。

- 第一步，安装夹持器法兰

夹持器本体上已经适配了配套机器人的法兰，只需要对准机器人法兰的螺纹孔，拧紧4个螺丝，BY-E140适配的螺丝为M6x8沉头螺丝。

- 第二步，安装夹持器

夹持器法兰安装后，把夹持器本体连接在夹持器法兰上。夹持器和夹持器法兰对准后，在前后各拧紧4个M3x8沉头螺丝，夹持器可以固定好。

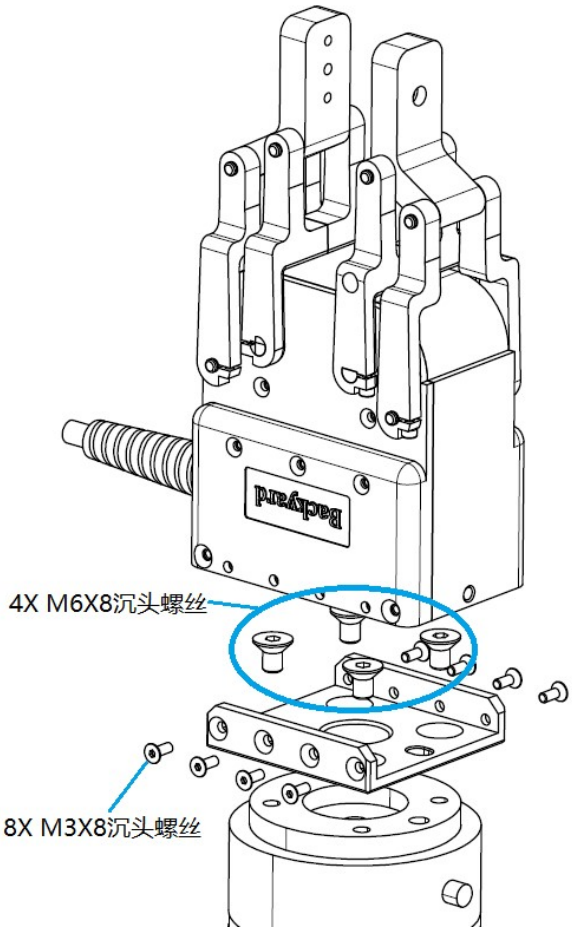

图 8 夹持器法兰装配图

安全事项：

- 确保螺丝正确并安全地安装到位。
- 确保使用工具安全装配，装配时不会有零件意外坠落造成危险。
- 确保夹持器正确对准法兰，不会有夹手的风险

夹持器安装后以后，在机器人的状态如下图。

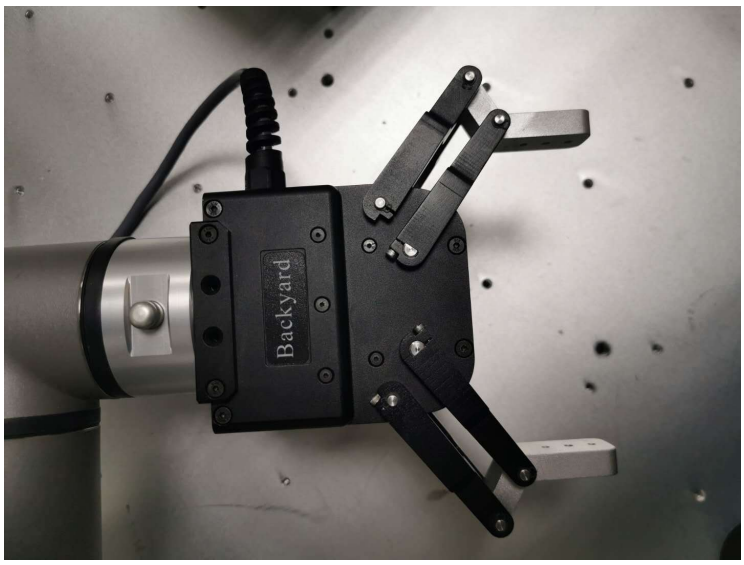

图 9 夹持器安装图

2.4.2 控制盒连接

控制盒可以安装在标准T35导轨上，参考安装方式如下，

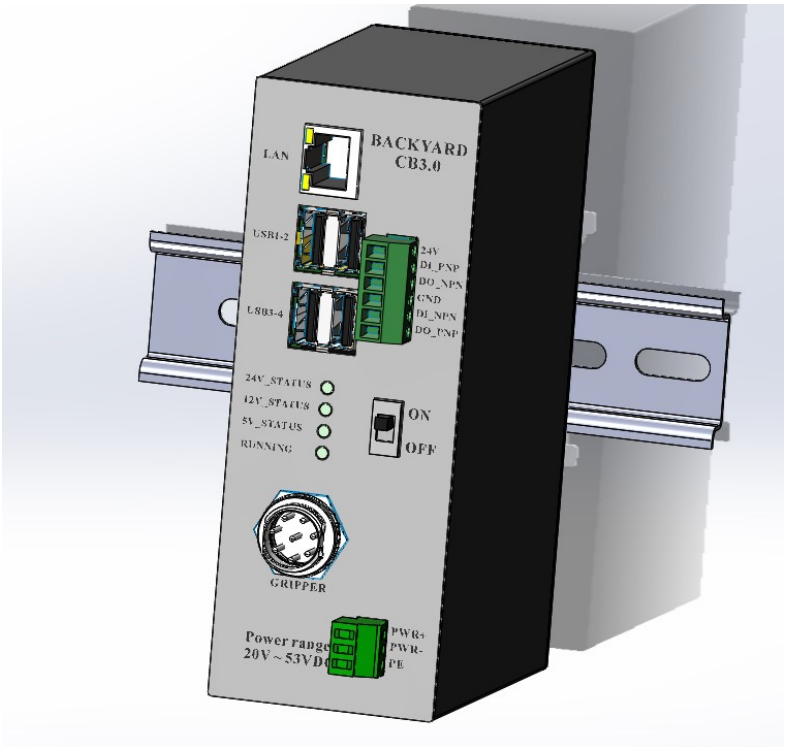

图 10 控制盒安装图

控制盒连接时需要做的工作是连接三个电缆插头。

第一， 连接夹持器的控制电缆。夹持器安装好以后，控制电缆可以沿机械臂本体，使用扎带或者专用电缆固定导槽固定。控制电缆的末端接入控制盒，如图 10 控制盒安装图左上方连接。

第二， 电源连接。控制盒内本身内置了DC-DC的电源模块，供电需要把控制盒直流输入端子接入电源，电源电源允许范围24VDC~48VDC。如图 10 控制盒安装图右下方连接。

第三， 以太网连接。机器人和控制盒直接通信基于以太网连接。使用标准RJ45接头的网线，连接控制盒和机器人控制柜。如图 10 控制盒安装图左下方连接。

安全事项：

1. 确保控制盒、夹持器和电缆不接触液体。潮湿的夹持器可导致死亡。
2. 控制盒和夹持器不得暴露在灰尘或超出IP20 等级的潮湿环境密切注意存在传导性灰尘的环境。

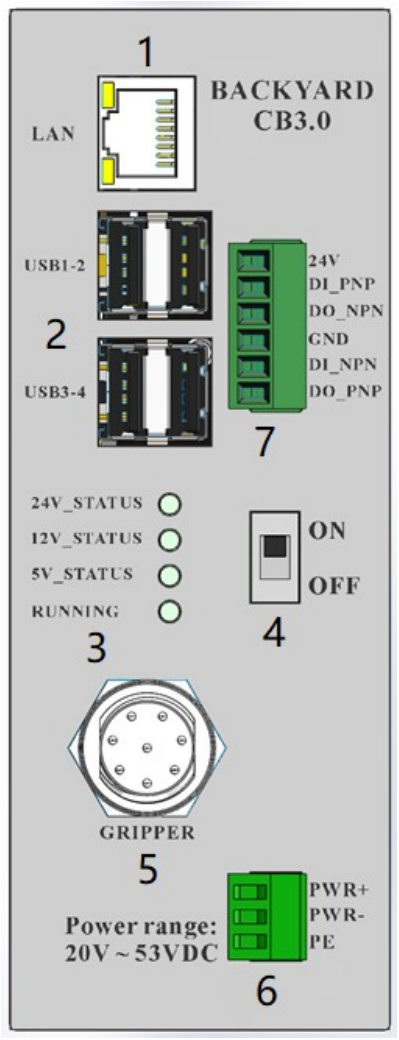

图 11 控制盒外形图

明细：

序号	名称	描述
1	网口	以太网连接，用于连接机器人或 PC
2	USB	用于连接 USB 设备
3	状态灯	状态指示
4	开关	控制盒开关，用于上电断电
5	夹持器电缆	连接夹持器和控制盒
6	供电端子	控制盒供电，电源范围 24VDC~48VDC
7	IO 端子	IO 输入输出连接

图 12 实物连接图

3.0 软件配置

3.1 夹持器的配置与升级

3.2.1 夹持器网络连接和设置

BY-E140的夹持器配置和升级通过网页进行，用户无需安装专门的软件，电脑或者手机连接夹持器，通过浏览器即可完成。

有两种方式可以连接BY-E140:

1、有线连接，通过网线控制盒Lan口；

2、无线WiFi(SSID：Backyardxxxx,”xxxx”为序列号， 密码：WelcomeBYWorld2018)。

有线连接时，在浏览器上输入192.168.137.9即可登录夹持器配置界面。

无线连接时，在浏览器上输入192.168.0.1，即可登录夹持器配置界面。

有线连接时，通过控制盒Lan口，因注意修改本机电脑IP至夹持器同一个网段，夹持器出厂默认IP为192.168.137.9，那么本机对应网卡的IP则可以修改为192.168.137.xxx，其中xxx为非9的，1-255之间的数字，子网掩码：255.255.255.0。

在浏览器上输入192.168.137.9也可登录夹持器配置界面。

3.2.2 夹持器主界面

通过WiFi连接夹持器成功后，打开浏览器输入浏览器上输入192.168.0.1或者192.168.137.9，首先登录夹持器配置界面。

BY-E140夹持器的网页界面右上方有两个电源选项：

重新启动和关闭。

此功能是为了重新启动和关闭夹持器控制盒控制软件，以减小直接上下电对于控制系统的冲击。建议在每次关闭夹持器电源之前，先点关闭，大约25s之后再关闭夹持器控制盒电源。

BY-E140夹持器登录界面如图 13 BY-E140夹持器主界面，包含夹持器的当前位置，速度，力和温度。

首次使用或者刚刚启动时，夹持器处于未标定状态（Dashboard页面左下角呈现红色未标定），此时需要进入手动模式界面，图 13 BY-E140夹持器主界面。点击Calibrate，进行上电标定，上电标定完成之后即可通过手动模式操作夹爪。

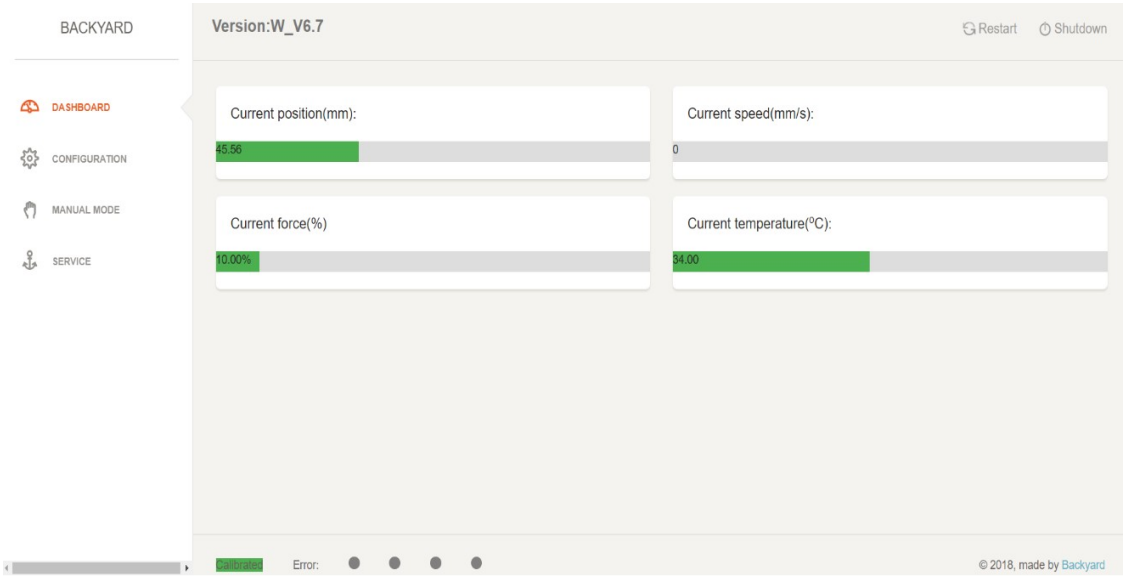

图 13 BY-E140 夹持器主界面

3.2.3 夹持器设置界面

BY-E140夹持器的设置界面包含网络设置和接口服务。

网络设置是指根据客户的使用条件去修改控制盒Lan口的IP和网关，设置完成后点击设置(需用网线连接该端口)，即可自动重启控制盒完成IP的设置。BY-E140提供四种接口：XMLRPC, ModbusTCP, TCP/IP, IO，其中一种接口服务默认自动启动，如果控制盒上信号灯不闪烁，那么接口服务未能正常启动。

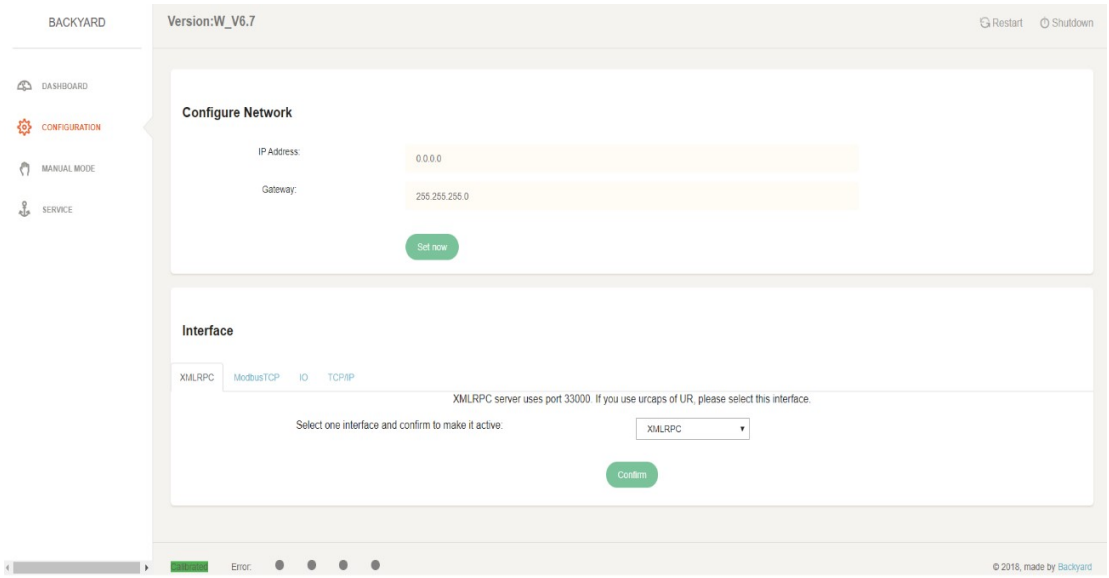

图 14 BY-E140 配置界面

3.2.4 夹持器手动模式

手动模式，提供了手动操作夹持器的方法。

上电之后，夹持器会处于未标定状态，点击Calibrate完成夹持器的上电标定，此时夹持器会自动完成一闭一张动作，识别出当前的夹持器行程。通过滑动条操作速度、加速度和力之后，调节位置滑动条，即可以以上述参数去控制夹持器到达指定位置。

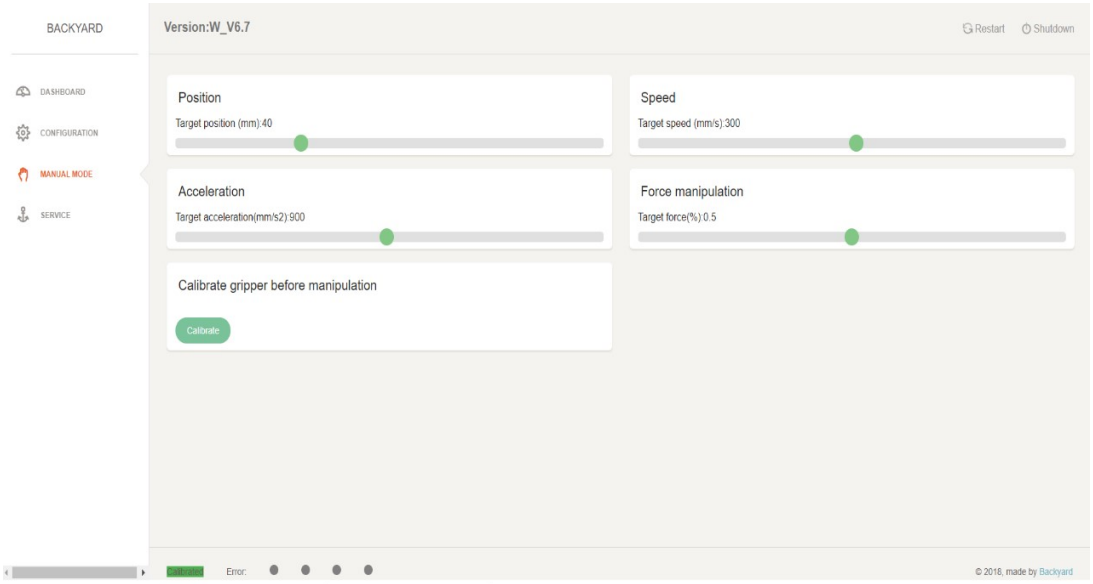

图 15 BY-E140 手动模式

3.2.5 夹持器软件升级

BY-E140夹持器的软件升级通过图 16页面完成。

选择升级文件（后缀.byup），点击Confirm to upgrade即可自动完成软件的升级和重启。重启之后在浏览器地址栏重新输入夹持器的IP地址或者192.168.0.1(WiFi连接)，即可观察到软件版本的变化(Version：Vxx)。

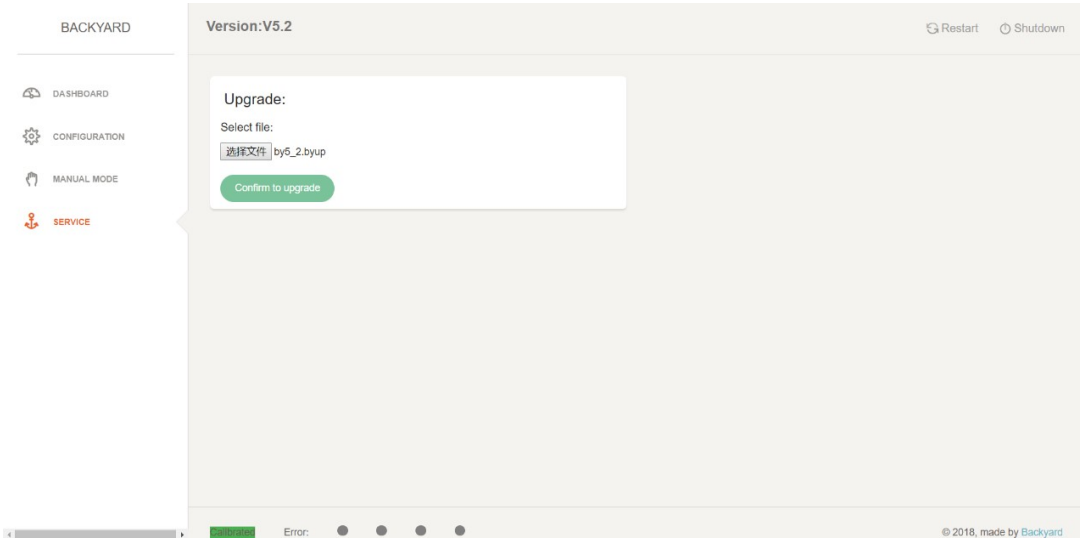

图 16 BY-E140 升级维护

3.2 通讯接口

3. 4. 1 ModbusTCP

BY-E140 夹爪提供 ModbusTCP server/slave 接口，激活方式如图 17，切记点击 confirm 确认，之前选择的接口则变为不可用。

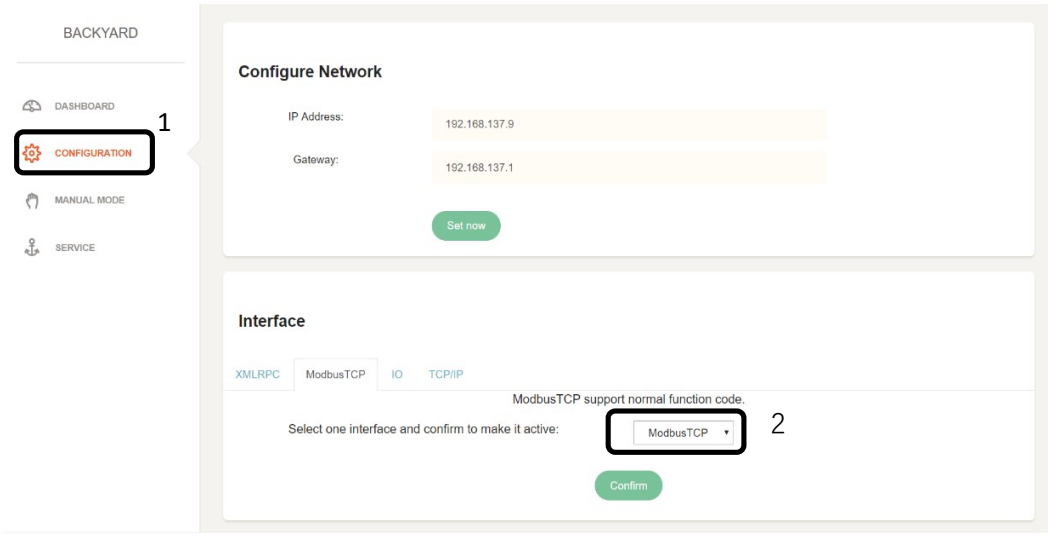

图 17 ModbusTCP 接口设置

表 1 ModbusTCP 地址表

地址	访问权限	名称	范围	备注
0x32 (50)	R/W	目标位置		目标位置（mm）的 100 倍，40mm 即为 4000。设置 CMD 后生效

0x33 (51)	R/W	目标速度		目标速度(mm/s)的 10 倍，100mm/s 即为 1000。设置 CMD 后生效
0x34 (52)	R/W	目标加速度		目标加速度(mm/s²)的 10 倍，1000 mm/s² 即为 10000。设置 CMD 后生效
0x35 (53)	R/W	目标力	0 - 1	目标力矩的 1000 倍，0.5 即为 500。设置 CMD 后生效
0x36 (54)	R/W	CMD 指令		指令执行之后需要恢复到 0 才能输入新指令， 10:calibrate,1: moveTo, 111: restart,222: shutdown
0x3C (60)	R	Error		零：OK
0x3D (61)	R	当前位置		当前位置（mm）的 100 倍，4000 即为 40mm
0x3E (62)	R	当前速度		当前速度(mm/s)的 10 倍，1000 即为 100mm/s
0x3F (63)	R	当前力		当前负载的 1000 倍。500 即为 0.5
0x40 (64)	R	电机电压		当前电机电压的 10 倍，120 即 12V
0x41 (65)	R	当前温度		当前电机温度，单位 ºC
0x42 (66)	R	指令完成		1：指令完成；0：指令正在执行
0x43 (67)	R	标定状态		1：夹爪已标定，0：夹爪未标定，moveTo 指令之前必须先标定夹爪

所有寄存器均使用-1 补码，即 65535 代表-1(65535-65536)。

示例程序(python)：

```python
#使用 pyModbusTCp 库，https://github.com/sourceperl/pyModbusTCP
from pyModbusTCP.client import ModbusClient
import time

#获取夹爪的当前状态
def getStatus(bygripper):
    retv=bygripper.read_holding_registers(60,8)
    if(retv):
        return retv
    else:
        None
#发送命令至夹爪
def setCMD(bygripper,cmd,pos,spd,acc,torq):
    pos=round(pos*100)
    spd=round(spd*10)
    acc=round(acc*10)
    torq=round(torq*1000)
    r0=bygripper.write_multiple_registers(50,[pos,spd,acc,torq,cmd])
    return r0
#创建名为 bygripper 的 ModbusTCP 对象，其中 host 为夹爪 IP 地址，其中 unit_id 为从设备
#地址，必须为 1.
bygripper= ModbusClient(host="192.168.137.9", port=502, unit_id=1, auto_open=True, auto_close=True)
bygripper.open()     #连接夹爪
setCMD(bygripper,0,0,0,0,0)      #重置命令寄存器
if(setCMD(bygripper,10,0,0,0,0)):    #标定夹爪
    while(getStatus(bygripper)[7]!=1):   #等待标定完成
        time.sleep(0.01)
    setCMD(bygripper,0,0,0,0,0)   #重置命令寄存器
    setCMD(bygripper,1,0,50,700,0.15)   #发送运动指令
    while(getStatus(bygripper)[6]!=1):  #等待运动指令完成
        time.sleep(0.01)
    setCMD(bygripper,0,0,0,0,0)         #重置命令寄存器
    setCMD(bygripper,1,60,50,700,0.15)  #发送运动指令
```

```
    while(getStatus(bygripper)[6]!=1):  #等待运动指令完成
        time.sleep(0.01)
bygripper.close()  #关闭 modbusTCP 连接。
```

代码 1 ModbusTCP 示例代码

3.4.2 TCP/IP

夹爪可配置成 TCP/IP 服务器模式，端口设定参考图 18，端口或者接口修改后，切记点击 confirm 确认，之前选择的接口则变为不可用。

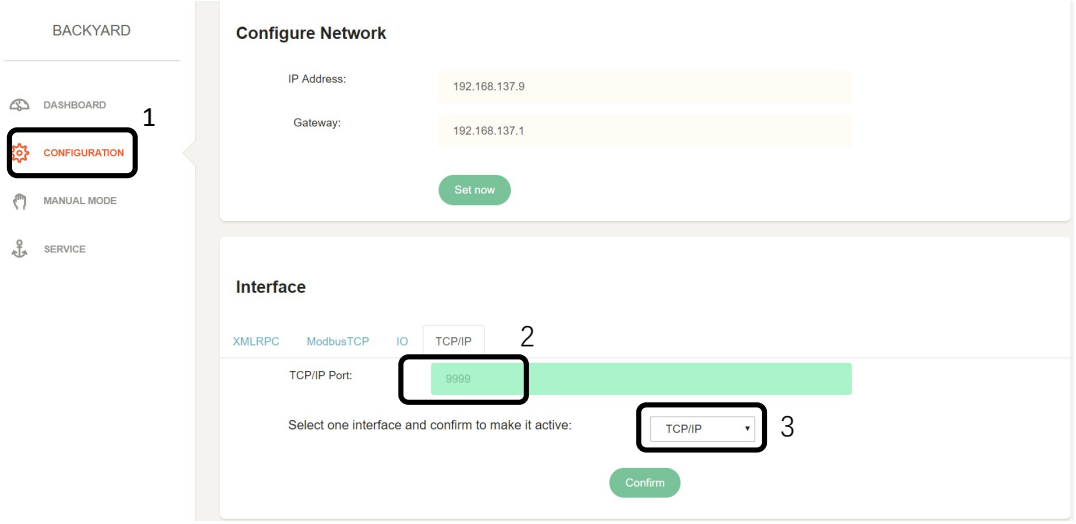

图 18 TCP/IP 服务器设置

TCP/IP 指令的发送格式为：“CMD(Arg1,Arg2,…)\n”，其中 CMD 为指令，Arg1-ArgN 为参数，如果无传入参数，则为空，即发送“CMD()\n”。返回值为数字字符串，支持的指令(CMD)如下：

表 2 TCP/IP 接口支持指令集

指令	说明	返回值
calibrateGripper()	夹爪标定	数字字符串
moveTo(pos,spd,acc,torque,tolerance=90,waitFlag=True) pos:0-140, spd:0-150, acc:0-750, torque:0-1	夹爪运动	数字字符串
getStatus()	状态获取	“[com,err,pos,spd,force, temperature,calibrated,stroke]”
getCalibrated()	标定状态	数字字符串

shutdown()	关闭控制器	数字字符串
restart()	重启控制器	数字字符串

getStatus()返回的列表字符串意义：

com：底层通讯状态，33 为正常，255 为异常；err：电机状态，0 为正常，非 0 为异常；

pos：当前位置(mm)，spd:当前速度(mm/s), force:当前力(0-100%)，temperature:当前电机温度，calibrated:夹爪是否已经标定，1 为已标定，0 为未标定，stroke：标定后夹爪的行程范围。

表 3 返回值数字字符串

数字字符串	说明	可能的恢复办法
“0“	指令正常执行	
“-1“	指令执行失败	夹爪异常或者软件 bug，联系服务商
“-2”	参数类型不对	检查输入参数的类型，如应该为 True，输入却是 1
“-3”	参数数量不对	输入参数的数量不对，过多或者必需参数没有
“-4”	指令不支持	检查输入指令的拼写
“-10”	非法字符串	检查输入字符串的格式和拼写

示例程序(python)：

```
1.  import socket
2.  import sys
3.  import time
4.
5.  HOST, PORT = "192.168.137.9", 9999  #夹爪 IP 和 TCP/IP 服务器端口号
6.  #定义 moveTo 函数
7.  def moveTo(sock,position,speed,acceleration,torque,tolerance=90,waitFlag=True):
8.      cmd="moveTo"
9.      cmd_args=cmd+"({0},{1},{2},{3},{4},{5})".format(position,speed,acceleration,torque,tolera
    nce,waitFlag)+"\n"
10.     sock.sendall(bytes(cmd_args,"utf-8"))
11.     return int(str(sock.recv(1024), "utf-8"))
12.
13.  #定义夹爪标定函数
14. def calibrateGripper(sock):
15.     cmd_args="calibrateGripper()"+"\n"
16.     sock.sendall(bytes(cmd_args,"utf-8"))
17.     return int(str(sock.recv(1024), "utf-8"))
18.  #获取夹爪当前状态
```

```
def getStatus(sock):
    cmd_args="getStatus()"+"\n"
    sock.sendall(bytes(cmd_args,"utf-8"))
    return str(sock.recv(1024), "utf-8")
#获取夹爪是否已经标定
def getCalibrated(sock):
    cmd_args="getCalibrated()"+"\n"
    sock.sendall(bytes(cmd_args,"utf-8"))
    return str(sock.recv(1024), "utf-8")
#关闭夹爪控制器
def shutdown(sock):
    cmd_args="shutdown()"+"\n"
    sock.sendall(bytes(cmd_args,"utf-8"))
    return str(sock.recv(1024), "utf-8")

 #重启夹爪控制器
def restart(sock):
    cmd_args="restart()"+"\n"
    sock.sendall(bytes(cmd_args,"utf-8"))
    return str(sock.recv(1024), "utf-8")
 #创建 socket 对象
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    #连接夹爪
    sock.connect((HOST, PORT))
    if(calibrateGripper(sock)==0):  #返回 0 则，标定成功
        print(getCalibrated(sock))
        print(moveTo(sock,60,150,700,0.5))  #运动到 60mm 位置
        print(getStatus(sock))          #获取当前状态
        time.sleep(1)
        print(moveTo(sock,10,150,700,0.5))   #运动到 10mm 位置
        print(getStatus(sock))
        print(moveTo(sock,65,150,700,0.5))
        print(moveTo(sock,0,150,700,0.5))
except Exception as e:
    print(e)

finally:
    sock.close()            #关闭 socket 释放资源
```

3.4.3 I/O

夹爪可配置成 IO 模式，BY-E140 支持一对 PNP 输入输出和一对 NPN 输入输出接口，使用 IO 接口不能动态的调整夹持范围，速度等参数，但是可以通过网页进行预编程，这样触发 IO 后，夹爪会按照预先的设定进行夹取。接口或者参数修改后，切记点击 confirm 确认，之前选择的接口则变为不可用。

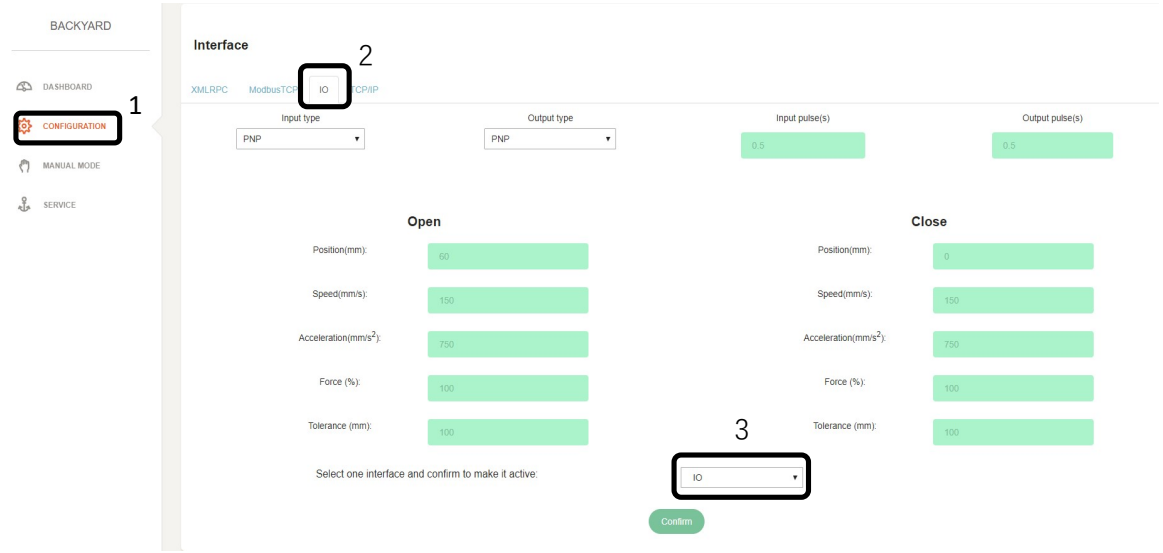

图 19 IO 接口设置

输入输出类型可分别设置为 PNP 或者 NPN 类型；Input pulse 则设定输入电平的最短时间，动作成功执行之后输出 Output pulse 时长的脉冲。Open 和 Close 列分别设置夹爪打开和关闭的运动参数。输入处于 On 状态时，夹爪打开，Off 状态时，夹爪关闭；同时在正常执行完打开和关闭后输出 On 状态脉冲，详细参考表 4。

表 4 输入输出不同配置的 IO 功能及输出

	输入 PNP	输入 NPN
输出 PNP （输出脉冲长度 Output pulse）	输入端高电平，夹爪打开，输出高电平脉冲 输入端浮动，夹爪关闭，输出高电平脉冲	输入端低电平，夹爪打开，输出高电平脉冲 输入浮动，夹爪关闭，输出高电平脉冲
输出 NPN (输出脉冲长度 Output pulse)	输入端高电平，夹爪打开，输出低电平脉冲 输入端浮动，夹爪关闭，输出低电平脉冲	输入端低电平，夹爪打开，输出低电平脉冲 输入端浮动，夹爪关闭，输出低电平脉冲

IO 模式启动之后，如若夹爪未处于已标定模式，那么需要进入网页进行手动标定，详细步骤参考图 20。

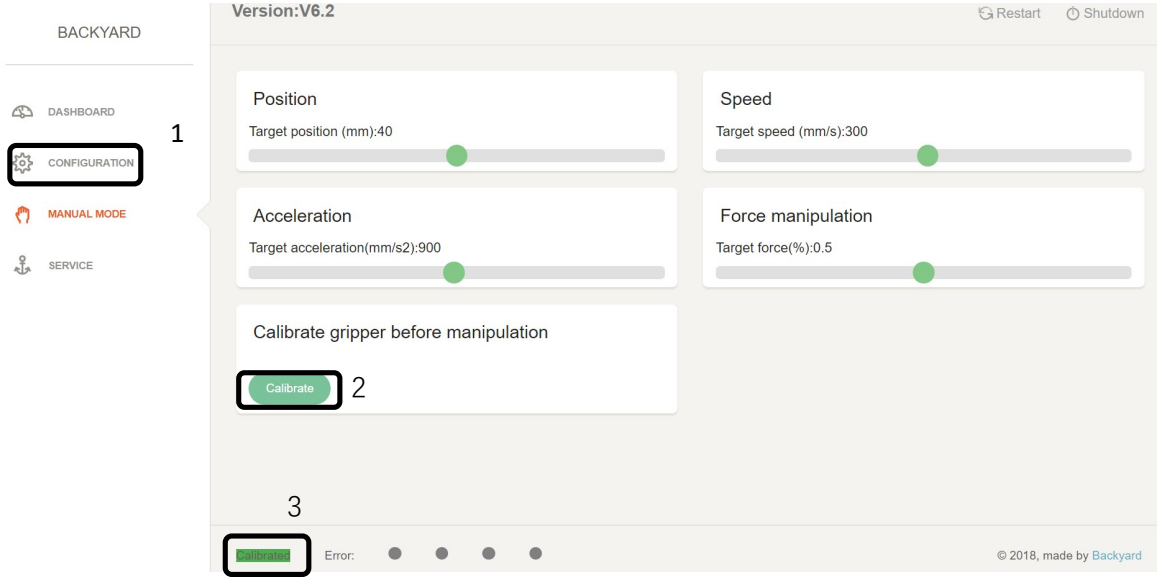

图 20 IO 模式下需手动标定（标定成功则 3 处显示 Calibrated）

图 21 为控制盒电路 IO 接口，pDx 为 PNP 数字输入输出，nDx 为 NPN 数字输入输出。24V 为控制盒 24V 电压输出，最大电流 1A。

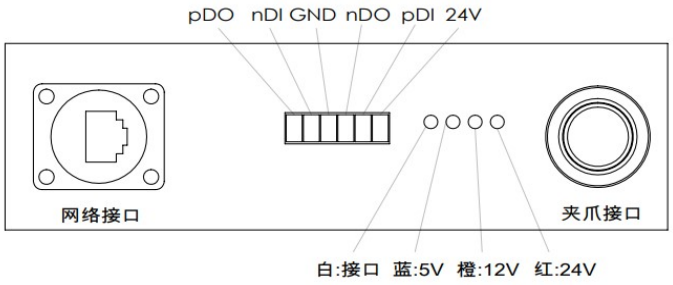

图 21 控制盒 IO 接口

3. 4. 4 XMLRPC 接口

XMLRPC 一种采用 xml 格式进行编码，HTTP 作为传输的远程调用协议。BY-E140 夹爪支持 XMLRPC 接口，端口号 33000，激活方式如图 22。切记点击 confirm 确认，之前选择的接口则变为不可用。

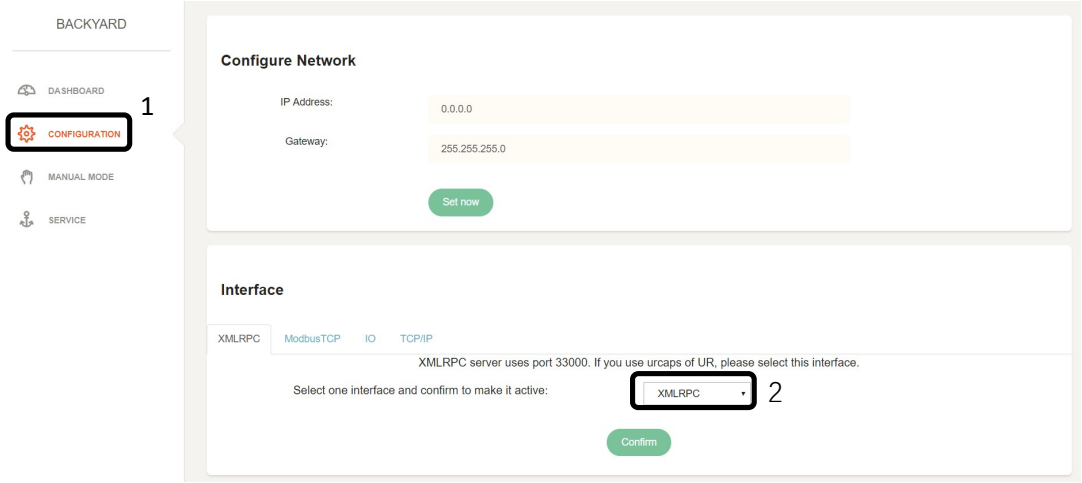

图 22 XMLRPC 接口

表 5 XMLRPC 接口支持指令集

指令	说明	返回值
calibrateGripper()	夹爪标定	bool 类型
moveTo(pos,spd,acc,torque,tolerance=90,waitFlag=True) pos:0-140, spd:0-150, acc:0-750, torque:0-1	夹爪运动	bool 类型
getStatus()	状态获取	List 列表或者数组类型，[com,err,pos,spd,force, temperature,calibrated,stroke]
getCalibrated()	标定状态	bool 类型
shutdown()	关闭控制器	null 或者 None
restart()	重启控制器	null 或者 None

示例程序(python)：

```
import xmlrpc.client
import time
#连接夹爪 xmlrpc 接口，192.168.137.9 为夹爪 ip，33000 为 xmlrpc 服务器端口
with xmlrpc.client.ServerProxy("http://192.168.137.9:33000/") as bygripper:
    if(bygripper.calibrateGripper()):       #标定夹爪
        print(bygripper.getCalibrated())    #获取夹爪标定状态
        bygripper.moveTo(20,100,500,0.5)    #使夹爪运动到 20mm 处
        time.sleep(1)
        bygripper.moveTo(60,100,500,0.5)    #使夹爪运动到 60mm 处
        for ii in range(0,4):
            # 获取夹爪状态，例如：[33, 0, 59.86, 0.0, 0.031, 12.0, 31, 1, 82.55875122212385]
```

```
            print(bygripper.getStatus())
            time.sleep(0.5)
        print("end")
    else:
        print("calibrate failed")
```

3.3 URCAP的安装、卸载

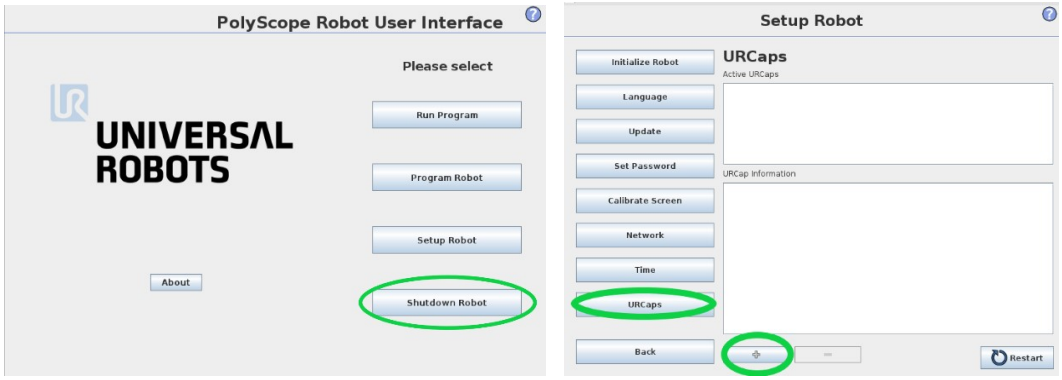

图 23 URCap 安装

BY-E140夹持器的URCap需在PolyScope3.7（或者PolyScope5.1）及之上的版本使用，安装流程如图 23和图 24。

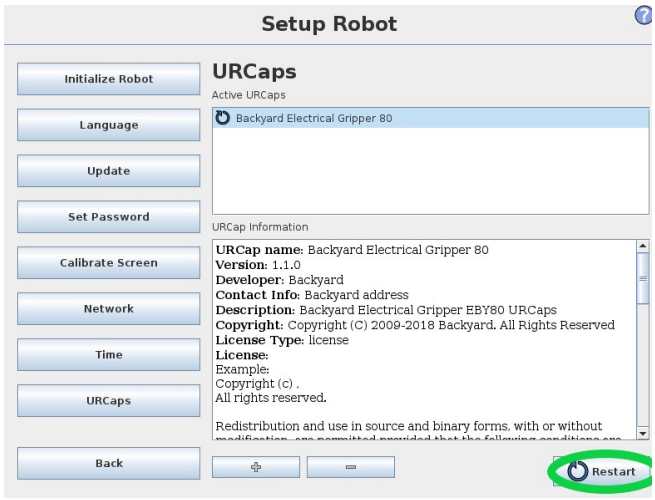

图 24 选择夹持器.urcap 文件，并重启机器人

URCap的卸载同安装很类似，只需进入URCap设置界面，点击-号，然后重启机器人即可，如图 25。

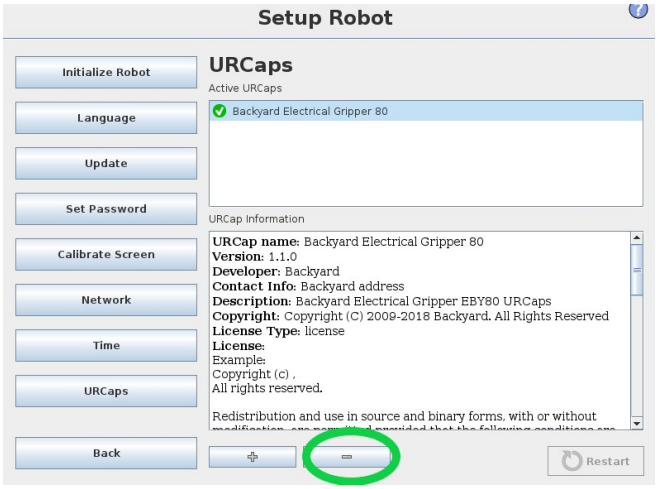

图 25 URCap 的卸载

3.4 UR机器人编程示例

BY-E140兼容UR机器人CB系列和E系列，本例以CB系列机器人为例介绍夹持器的操作和使用，E系列操作类似。

配合UR机器人使用，请确认在夹持器的网页配置中，选择XMLRPC模式，参考图 18 TCP/IP 服务器设置，**并点击confirm确认**。

3.5.1 机器人控制器和夹持器 IP 配置

为了能够让机器人与BY-E140夹持器通讯，需要将机器人与夹持器设置在同一个网段，如夹持器的IP是192.168.1.4，那么机器人的IP需要配置成192.168.1.xxx（xxx为非4的0-254之间的数）。然后进入安装设置界面BY-E140 Gripper，将夹持器的IP填入IP地址框中，点击连接，连接成功则Status行会显示已连接，否则请检查网络的连接和IP设置。再点击Calibrate完成夹持器的上电标定，如图 26。

3.5.2 机器人安装栏设置

默认参数栏是后续编程过程中添加夹持器节点时一些默认的参数设置。

- 位置（position）表示夹持器需要到达的位置。
- 速度（speed）则表示夹持器的运行速度。

- 加速度（acceleration）表示夹持器运动过程中的加减速。
- 力（Force）代表夹持器手指遇到阻力之后最大的作用力也即夹持力，取值范围0-1,1代表最大夹持力。
- 误差(Tolerance)表示夹持器伺服时位置的最大偏差，比如夹取的物体宽度是30mm，误差设置为3mm，那么如果程序在运行时，检测到该次夹取实际宽度是20mm，那么会停止程序的运行以防止误夹。

电源选项（Power options）可以实现BY-E140控制器的重启和关闭。每次将夹持器断电之前，建议使用shutdown将夹持器先行关闭，大约25s之后再关闭电源。

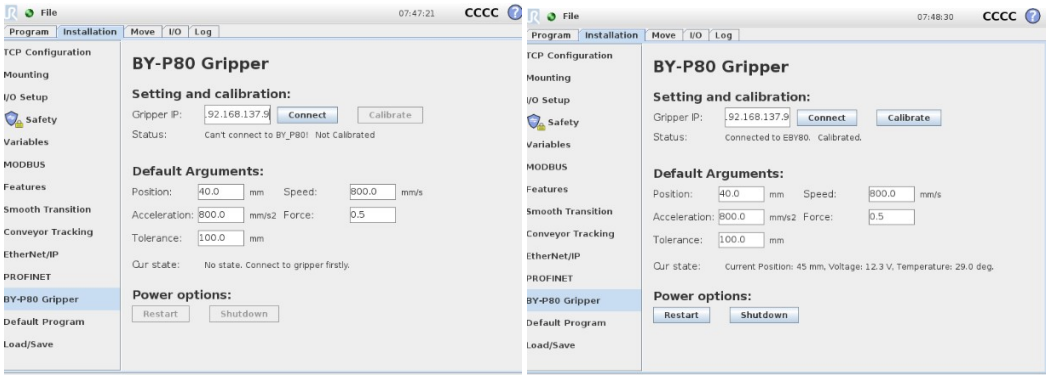

图 26 BY-E140 安装设置界面

3.5.3 机器人程序

完成夹持器的安装设置之后，即可正常使用夹持器。

进入程序选项页，通过URCaps选项页添加夹持器的程序节点，选中夹持器节点之后，通过命令选项页即可看到每一个程序节点对应的参数，图 27，WaitOrNot选项框默认勾选表示机器人要等到夹持器到达指定位置或者夹住物体之后才会执行后续指令，否则夹持器动作和机器人的后续指令执行并行进行，这在一些特定的应用中可以用于节约时间。Perform immediately可用于将最新的设置参数应用到夹持器。

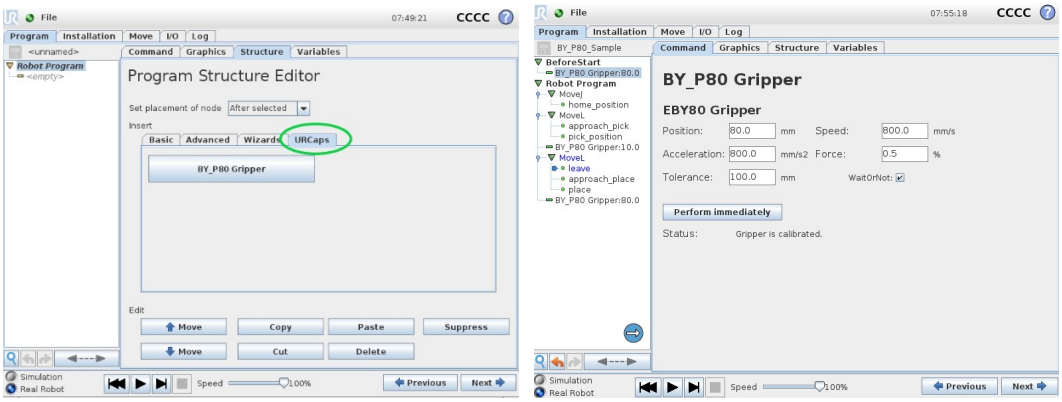

图 27 BY-E140 编程示例

前面提到的WaitOrNot选项框可以让夹持器的运动与机器人呢后续指令的执行并行，那么通常也需要在特定的时刻查看夹持器的位置等信息，然后才能执行后续指令，图 28 ，getBYGripperStatus()函数返回一个列表[错误，位置，速度，电压，温度]。

- 错误：为0，则代表没有夹持器错误；
- 位置：当前的夹持器位置，单位mm；
- 速度：当前的夹持器速度，单位mm/s；
- 电压：当前的工作电压，单位V；
- 温度：当前的电机温度，单位oC。

BY-E140电机的最高允许工作温度是78摄氏度。如果超过此温度，那么控制器会释放电机的扭矩以保护电机，此时请间隔20分钟后再使用。

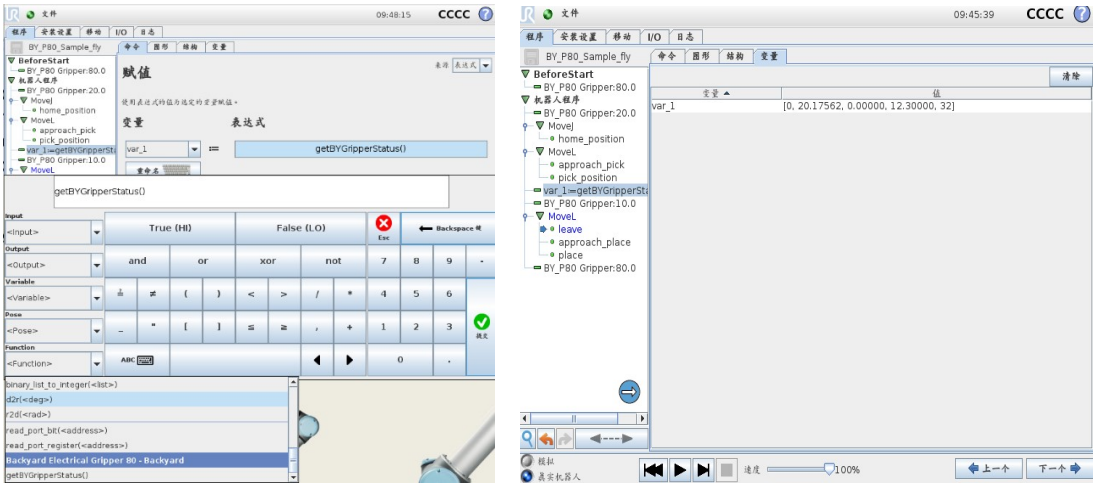

图 28 读取夹持器状态

3.5 艾利特机器人编程示例

3.6.1 驱动脚本导入

1， 将 BYSocket.lua 文件放到 rbctrl/luadir 文件夹目录下，然后把文件夹放到 U 盘里。
2， U 盘插入机器人控制器
3， 在机器人示教器上左上角选择系统->U 盘到本地->脚本升级
4， 在机器人示教器右上角选择脚本，打开 BYSocket。

3.6.2 驱动使用说明

1， 设置 D0 变量选择夹爪运行的模式
D0==1， 夹爪校准
D0==2， 夹爪移动
D0==3， 测试打开闭合
D0==4， 获取夹爪状态，见表 3，并将返回值放入 D10-D15 变量中去
D0==5， 重启夹爪
D0==6， 关闭夹爪

表 3 返回值数字字符串

数字字符串	说明	可能的恢复办法
“0 “	指令正常执行	
“-1 “	指令执行失败	夹爪异常或者软件 bug，联系服务商
“-2”	参数类型不对	检查输入参数的类型，如应该为 True，输入却是 1
“-3”	参数数量不对	输入参数的数量不对，过多或者必需参数没有
“-4”	指令不支持	检查输入指令的拼写
“-10”	非法字符串	检查输入字符串的格式和拼写

3.6.3 夹爪运动模式设置

当设置夹爪 D0==2 时，要同时设置夹爪的目标位置，速度，加速度和加持力这 4 个参数。

实现这四个参数的设置，只需要修改机器人的 D1-D4 变量。

D1 对应的是目标位置，范围 0-80 mm

D2 对应的是速度， 范围是 0-500 mm/s

D3 对应的是加速度，范围是 0-1600 mm/s^2

D4 对应的是加持力 范围是 0-1 最大加持的的百分比

3.6 ROS接口

3.6.1 ROS 接口设置与内容

BYE140 自 6.4（包含 6.4）版本往上支持 ROS 接口，相应的 ROS 包目前还未在代码托管网站公开，可与购买商联系获得。ROS 包基于 Kinetic 版本开发，主要包含一个 publisher 节点（bygripper_status）广播名为 by_status 的 topic，该 topic 包含当前夹爪的状态和一个服务节点(bygripper_driver)包含一些可以控制夹爪的 service（服务）。本文假设使用 catkin 工具，ROS 放置在 catkin_ws/src 文件夹中。

确保 1、夹爪通讯接口设置在 ModbusTCP 模式，参考 3.2.1 节设置，

2、byp80 ROS 包具有可执行权限(*chmod +x aaa*，其中 aaa 为文件夹名)，

3、回到 catkin_ws 目录，执行 *catkin_make*，并 *source ./devel/setup.sh*，

4、启动 roscore，并在另一个终端执行 *roslaunch byp80 byp80.launch ip:=”192.168.137.9”*

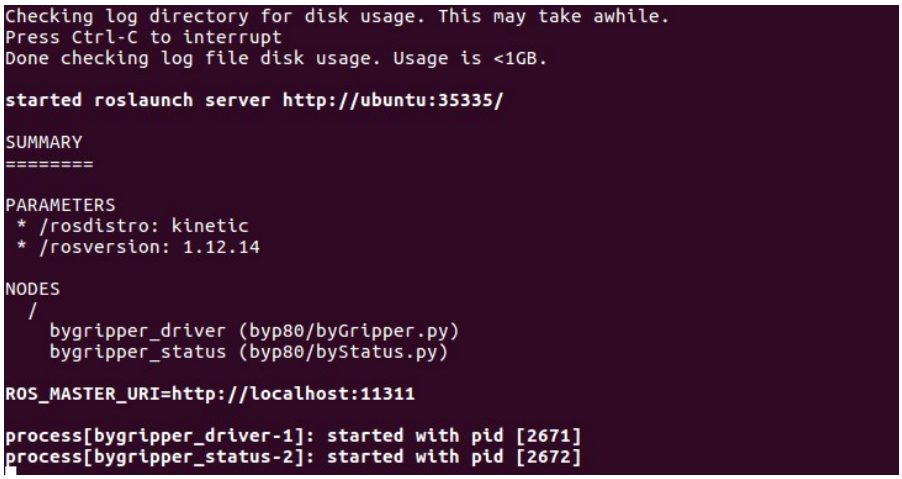

图 29 roslaunch 成功之后的输出

如果运行 roslaunch 之后得到类似图 29 的输出，那么代表 byp80 的 ROS 包启动正常，其中 ip 为夹爪的 ip 地址。

表 6 by_status topic 发布的消息 ByStatus 的格式定义

消息内容	类型	备注
communication_error	float32	通讯错误，0：正常；1：与夹爪通讯有问题；检查连接或者夹爪 ip 设置
motor_error	float32	电机故障，0：正常；非 0 则代表异常，请联系购买商
current_position	float32	当前手指的位置，单位 mm
current_speed	float32	当前手指的速度，单位 mm/s
current_force	float32	当前驱动力，范围 0-1,1 代表最大驱动力
current_voltage	float32	当前工作电压，单位为伏特
current_temperature	float32	当前温度，单位为摄氏度
completed	bool	当前指令是否完成
calibrated	bool	夹爪是否已经标定，使用之前必须先标定

bygripper_driver 服务节点提供的服务如表 7，

表 7 bygripper_driver 节点提供的服务

服务名称	参数与返回值	备注
CalibrateGripper()	参数：无；返回值：bool	标定成功为 True，否则为 False
GetCalibrated()	参数：无；返回值：bool	已标定为 True，否则为 False
GetStatus	参数：无；返回值：(communication_error，motor_error,current_position, current_speed, current_force, current_voltage, current_temperature, completed,calibrated)	Communication_error:通讯错误，检查连接和 ip 设置；Motor_error：电机状态异常，断电重启或者联系购买商；Current_position：当前位置，单位 mm；current_speed:当前速度，单位 mms/s；current_force：当前输出力，0-1,1 对应最大力输出；current_voltage：当前工作电压，单位伏特；current_temperature：电机当前温度，单位摄氏度；completed:指令完成；calibrated:类型 bool，夹爪是否标定
MoveTo	参数：(position, speed, acceleration, torque, tolerance, waitFlag)；返回值：bool	Position:目标位置，单位 mm；speed：目标速度，单位 mm/s；acceleration:目标加速度，单位 mm/s2；torque：目标夹持力，0-1,1 为输出最大力；tolerance：目标位置与实际位置最大误差，单位 mm，超过该误差则返回 Flase；waitFlag：类型 bool, 表示程序是否阻塞，直至夹爪到达目标位置或者夹住物体。
RestartGripper	参数：无；返回值：successful，类型 bool	重新启动夹爪，成功则返回 True，否则 False
ShutdownGripper	参数：无；返回值：successful，类型 bool	关闭夹爪，成功则返回 True，否则 False

3.6.2 ROS 接口调用例子

Client.py 调用 ROS 接口服务。

```
1.  #!/usr/bin/env python
2.
3.  import sys
4.  import rospy
5.  from byp80.srv import *
6.  from byp80.msg import *
7.
8.  def test():
9.      rospy.wait_for_service('shutdown_gripper')    #等待 shutdown service 可用
10.     try:
11.         #设置服务代理
12.         calib_gripper = rospy.ServiceProxy('calibrate_gripper', CalibrateGripper)
13.         moveTo = rospy.ServiceProxy('move_to', MoveTo)
14.         getStatus=rospy.ServiceProxy('get_status',GetStatus)
15.         restart=rospy.ServiceProxy('restart_gripper',RestartGripper)
16.         shutdown=rospy.ServiceProxy('shutdown_gripper',ShutdownGripper)
17.         #调用标定服务，并在成功后发出运动指令
18.         if(calib_gripper().successful):
19.             moveTo(5,200,500,1,100,True)
20.             moveTo(50,200,500,1,100,True)
21.         print(getStatus())
22.         #print(restart())
```

```
        #print(shutdown())
    except rospy.ServiceException, e:
        print "Service call failed: %s"%e

if __name__ == "__main__":
    test()
```

listener.py 订阅夹爪状态 topic。

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import String
from byp80.msg import *

def callback(data):
    rospy.loginfo(rospy.get_caller_id()+"I heard %s",'Status:({0},{1},{2},{3},{4},{5},{6},{7},{8})'.format(data.communication_error,data.motor_error,data.current_position,data.current_speed,
                                                                data.current_force,data.current_voltage,data.current_temperature,data.completed,data.calibrated))

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # name are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)    #初始化节点

    rospy.Subscriber('by_status', ByStatus, callback)     #订阅 by_status 这个夹爪状态 topic

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```