epicsEdgeRoboArm Documentation

Release 1.0

Jeff Gebhardt, Pete Jemian

May 17, 2016

Contents

EPICS support for the OWI Edge Robotic Arm over USB

maintainer Pete R. Jemian

email jemian@anl.gov

docs http://epicsEdgeRoboArm.readthedocs.org

git https://github.com/bcda-aps/epicsEdgeRoboArm.git

CHAPTER 1

Contents

1.1 Overview

Fig. 1.1: OWI-535 Edge Robotic Arm

The OWI-535 Edge Robotic Arm is a child's toy that is built from a kit. [1] The arm has some interesting specifications:

- four motorized rotary axes: base, shoulder, elbow, wrist
- one motorized grip
- LED
- hand-held switch box
- maximum 100g load

It's a fun learning device. The robot arm has its limitations that make it useless for any practical robotics implementation:

[1] official site: http://www.owirobot.com/robotic-arm-edge-1/

- no twist axis for the wrist
- no limit switches
- no encoders
- DC motor speed depends on power available from batteries

An optional USB interface [2] is available, providing a Windows application to operate the robot. The USB command protocol was deciphered [3] and posted online by a third party, enabling communication from a Linux computer.

1.1.1 USB protocol

Device appears on Linux as:

```
Bus 005 Device 007: ID 1267:0000 Logic3 / SpectraVideo plc
```

A simple USB vendor control transfer of three bytes appears to be the entire control method. The bits in these bytes appear to directly control the physical lines of the microcontroller. Effectively the microcontroller is behaving as nothing more than a USB attached I/O expander.

- Bits 0-7 control the LED (0 for off 0xff for on)
- Bits 8-15 turn a motor output on (direction is just done by having two switches per motor)

The bits in the motor bytes are used in pairs as inputs to ST1152 motor controllers. The truth table for these controllers is:

```
 Input | Motor
 A | B | Output
---+---+-------
 0 | 0 | Idle
 0 | 1 | Forwards
 1 | 0 | Backwards
 1 | 1 | Brake
```

The windows software only ever uses 00, 01 and 10 i.e. it never applies a brake signal. To summarize, bits 0 and 1 control the first motor, bits 2 and 3 the second and so on for all five motors. This leaves bits 10-15 unused.

1.1.2 2012 ANL Energy Showcase - First Demo of EPICS IOC

Fig. 1.2: Robotic Arm demo at 2012 ANL Energy Showcase

In preparation for the 2012 Argonne National Laboratory Energy Showcase (an open house for the community [4]), the BCDA group [5] created linux-based EPICS controls [6] for the robot arm to simulate how robots install samples into

[2] USB kit: https://www.owirobot.com/products/USB-Interface-for-Robotic-Arm-Edge.html
[3] libusb program for Linux : http://www.kyllikki.org/rbtarm.c
[4] 2012 ANL Energy Showcase: https://www.flickr.com/photos/argonne/7996170862/in/album-72157631558448229
[5] BCDA: http://www.aps.anl.gov/bcda
[6] First IOC was created by Jeff Gebhardt, APS BCDA group

X-ray detectors at several of the APS experiment area beamlines. The robots allow for faster sample loading and enable scientists to use the APS while at their home institutions.

Using a Raspberry Pi as the Linux IOC host and EPICS, this hands-on IOC demonstrates how modestly a "complete" control system might be constructed. A GUI can be added on the network for alternative control of the robot.

1.1.3 Demo System with joystick

Recently, USB joystick control was added to the IOC [7] which enables truly headless (no GUI needed) operations. In the photo here, a wooden marble tree instrument [8] is used to provide an interesting target for the robot arm actions.

photo

1.2 EPICS IOC

The IOC must be run as root. EPICS base 3.14.12.1 (or higher) is required. The support is provided by modifying synApps v5.6 [1], removing modules that are not used, and adding support where appropriate.

1.2.1 USB

The Linux host must provide a libusb support library. USB communications must be performed by root, so the IOC must run as root.

uses *drvAsynUSBPort.c* : Asyn device support using local usb interface

asyn configuration of USB communications in IOC's `st.cmd` file

```
drvAsynUSBPortConfigure("USB1", "Robotic Arm", 0x1267, 0, 0, 0, 0, 1)
asynOctetConnect("USB1", "USB1")
```

1.2.2 Databases

All actions of the robot are provided through a single EPICS database: `edgeRoboArmIOC/support/ip-2-13/ipApp/Db/roboArm.db`

The USB communication is controlled by *asyn* through a single PV:

USB communication through *asyn*

[7] EPICS IOC joystick control added to the IOC by Keenan Lang, APS, BCDA group
[8] marble tree: http://www.berea.com/appalachian-fireside-gallery/
[1] synApps: http://www.aps.anl.gov/bcda/synApps/

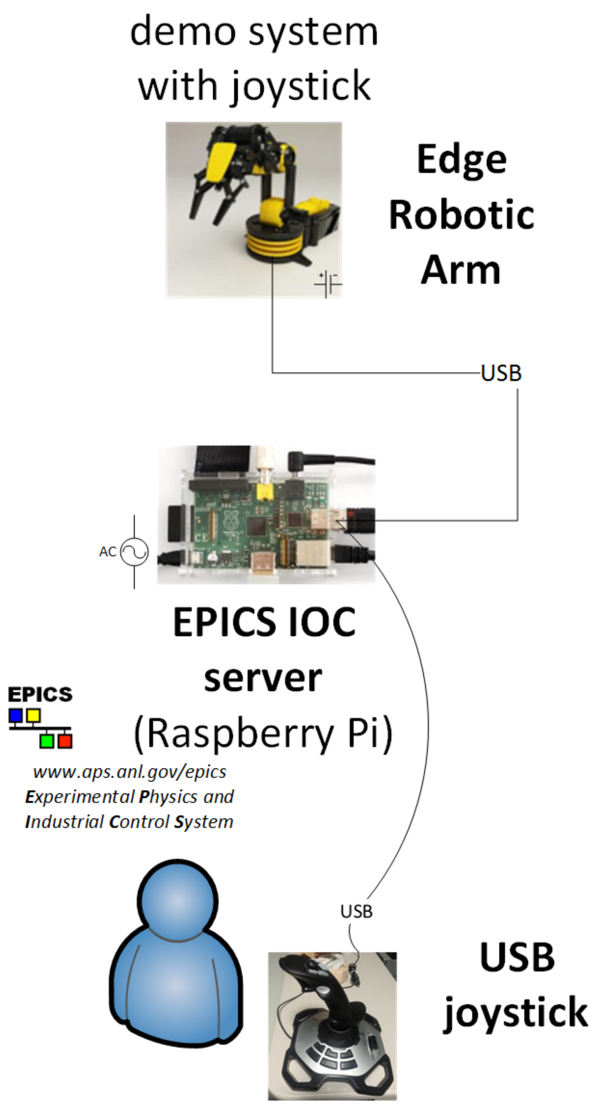

Fig. 1.4: Basic connection schematic for headless IOC and operation using a joystick

```
record(stringout, "$(P)$(A)send_cmd_str") {
  field(DESC, "send the motion command string")
  field(DTYP, "asyn robo stringParm")
  field(OUT, "@asyn($(PORT))")
}
```

The bit position of each motion axis is encoded in the database, such as:

bit command of each axis is encoded: elbow UP=16, DOWN=32, STOP=0

```
record(mbbo,  "$(P)$(A)elbow_move") {
  field(DESC, "elbow motion")
  field(DTYP, "Raw Soft Channel")
  field(ZRST, "STOP")
  field(ZRVL, "0")
  field(ONST, "UP")
  field(ONVL, "16")
  field(TWST, "DOWN")
  field(TWVL, "32")
  field(FLNK, "$(P)$(A)send_cmd.PROC  PP MS")
}
```

Commands for all axes are aggregated in these two records:

USB command assembled in two records

```
record(calc,  "$(P)$(A)send_cmd") {
  field(DESC, "send the motion command")
  field(INPA, "$(P)$(A)grip_move.RVAL      NPP NMS")
  field(INPB, "$(P)$(A)wrist_move.RVAL     NPP NMS")
  field(INPC, "$(P)$(A)elbow_move.RVAL     NPP NMS")
  field(INPD, "$(P)$(A)shoulder_move.RVAL  NPP NMS")
  field(CALC, "A+B+C+D")
  field(FLNK, "$(P)$(A)send_cmd_2.PROC      PP  MS")
}

record(scalcout, "$(P)$(A)send_cmd_2") {
  field(DESC, "send the motion command")
  field(INPA, "$(P)$(A)send_cmd.VAL        NPP NMS")
  field(INPB, "$(P)$(A)base_move.RVAL      NPP NMS")
  field(INPC, "$(P)$(A)led_onoff.VAL       NPP NMS")
  field(CALC, "STR(A)+' '+STR(B)+' '+STR(C)")
  field(OUT,  "$(P)$(A)send_cmd_str.VAL     PP  MS")
}
```

1.2.3 IOC startup

A standard *xxx* IOC from synApps was used to create the IOC for the robot. All configuration details are provided in the *st.cmd* and related scripts. The IOC is started by running the bash script `edgeRoboArmIOC/support/xxx-5-6/iocBoot/iocLinux/run`. An additional script is provided to run the IOC in a detached *screen* session: `in-screen.sh`.

1.2.4 cron task

A bash script was created to be run as a periodic (once a minute) *cron* task, checking to see if the IOC is not running. If not running, it checks if the robot's USB connection is detected and then tries to start the IOC. With this task running, the EPICS IOC starts automatically after the Linux OS is booted and the robot arm is connected by USB. The file is stored in the startup directory: `edgeRoboArmIOC/support/xxx-5-6/iocBoot/iocLinux/restart_ioc_check.sh`

restart_ioc_check.sh

```bash
#!/bin/bash

# restart_ioc_check.sh
# must run as root to use USB support

# run by  crontab -e
#      * * * * *  /root/restart_ioc_check.sh 2>&1 /dev/null
#              field          allowed values
#              -----          --------------
#              minute         0-59
#              hour           0-23
#              day of month   1-31
#              month          1-12 (or names, see below)
#          day of week    0-7 (0 or 7 is Sun, or use names)
#
#   # auto-start the robotic arm IOC
#   * * * * *  /root/restart_ioc_check.sh 2>&1 /dev/null
#
#------------
#
# also, as root (do these steps BEFORE enabling the cron job):
#   cd /root
#   ln -s ${ST_CMD_DIR} ./ioc
#   ln -s ioc/restart_ioc_check.sh ./
#   ln -s ioc/is_ioc_up.py ./


export EPICS_HOST_ARCH=`/usr/local/epics/base/startup/EpicsHostArch`
export EPICS_BASE_BIN=/usr/local/epics/base/bin/${EPICS_HOST_ARCH}
export ROBOT_DIR=/usr/local/epics/epicsEdgeRoboArm
export IOC_TOP=${ROBOT_DIR}/edgeRoboArmIOC/support/xxx-5-6
export ST_CMD_DIR=${IOC_TOP}/iocBoot/iocLinux


# ID 1267:0000 Logic3 / SpectraVideo plc
export usb_connect=`lsusb | grep  "ID 1267:0000 Logic3 / SpectraVideo plc"`

if [ "${usb_connect}" != "" ]; then
  #echo "<${usb_connect}>"
  export ioc_off=`/root/is_ioc_up.py`
  if [ "${ioc_off}" != "" ]; then
    #echo "IOC is not running"
    ${EPICS_BASE_BIN}/caRepeater &

    cd  ${ST_CMD_DIR}
    ./in-screen.sh
```

```
  fi
fi
```

1.2.5 SNL state program (optional)

In an attempt to automate the actions of the robot arm in a programmed sequence, Jeff Gebhardt wrote a state notation language sequence program (and accompanying database). The automation allows for move sequences up to five steps. This support can be found in:

- `edgeRoboArmIOC/support/ip-2-13/ipApp/Db/roboArmSeq.db`
- `edgeRoboArmIOC/support/ip-2-13/ipApp/Db/roboArmSeq_settings.req`
- `edgeRoboArmIOC/support/ip-2-13/ipApp/src/RoboArm.st`

A movie was created showing the robot locating, grasping, and lifting a toy block, then dropping it into a nearby coffee cup.

To accomplish this, the batteries were new and the robot, block, and coffee cup were placed in a known starting position.

Moves were programmed based on elapsed time. Due to lack of feedback encoding, backlash and windup of the motor gears, and unreliable positioning based on battery power available for a given time of movement, it is not realistic to program any sequence of more than 5 waypoints.

In short, we were lucky to get a good video. Took some careful work to be that lucky.

1.2.6 GUI support

Initial user interfaces created were:

- CSS BOY
- MEDM

Screens are provided for each.

> *Interesting to note the first "user" at the 2012 ANL Energy Showcase was a six-year old child who wanted to press the CSS BOY screen button directly with her finger, completely ignoring the offered mouse interface to the GUI.*

(Now, with touch-screen laptops, the CSS BOY interface can be tested for multi-touch compatibility.)

Later, a Python GUI was created to work on the Raspberry Pi. This interface allowed the use of keyboard bindings to each of the GUI buttons. From this keyboard binding interface, a true multitouch capability was added.

1.2.7 Joystick support

See the section *Joystick - IOC support (not really a client)* for more details.

Now, the LED feature on the robot arm becomes useful! Verify the IOC is running by pulsing the LED with the programmed button on the joystick. Once that works, the joystick is now ready to be used.

1.3 User Interfaces

Since the robot arm does not have encoders, position limit switches, or other position sensors, there is no ability to determine position. The controls for the robot are provided through these 6 PVs:

EPICS PV name	PV RTYP	values
`xxx:A1:led_onoff`	bo	OFF=0, ON=1
`xxx:A1:base_move`	mbbo	STOP=0, CW=1, CCW=2
`xxx:A1:shoulder_move`	mbbo	STOP=0, UP=1, DOWN=2
`xxx:A1:elbow_move`	mbbo	STOP=0, UP=1, DOWN=2
`xxx:A1:wrist_move`	mbbo	STOP=0, UP=1, DOWN=2
`xxx:A1:grip_move`	mbbo	STOP=0, CLOSE=1, OPEN=2

These client interfaces have been demonstrated:

1.3.1 CSS BOY client

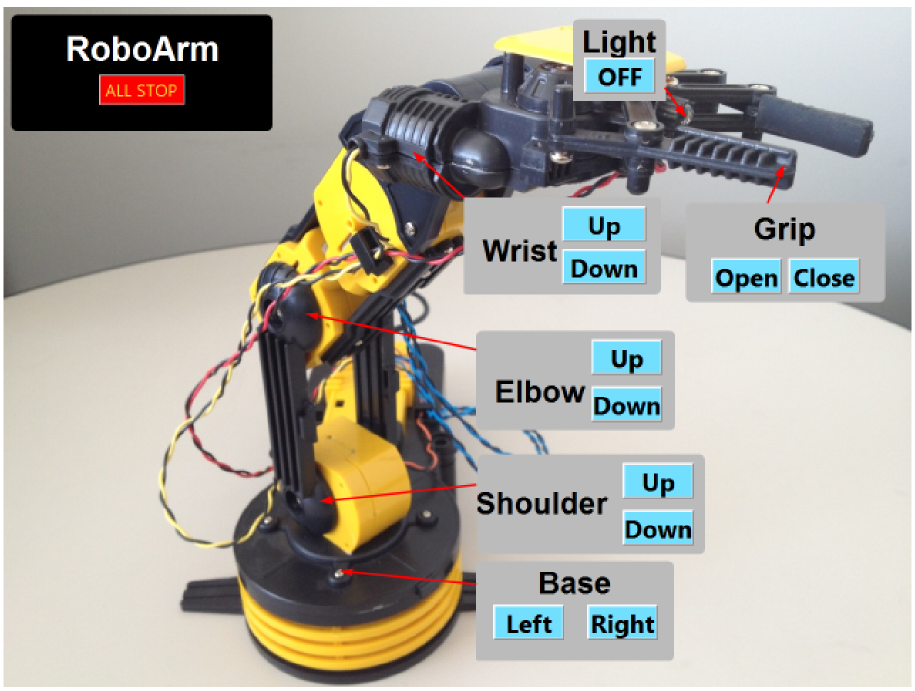

Fig. 1.5: CSS BOY control screen

Basic controls of the robot axes and LED are provided by buttons on the CSS BOY screen. Each action will happen as long as the button is held down.

1.3.2 Python - PyEpics and PyQt4 client

A Python Graphical User Interface client (using PyEpics and PyQt4) was created to provide a button interface to the robot controls. This was especially useful since the hand-held controller broke.

The GUI screen is rather basic, it provides buttons for all robot arm actions. Additional keyboard equivalents were assigned. With the key press bindings, it was then possible to control more than one axis of the robot arm at the same time. The success of any multitouch interface to this robot is limited by available battery power.

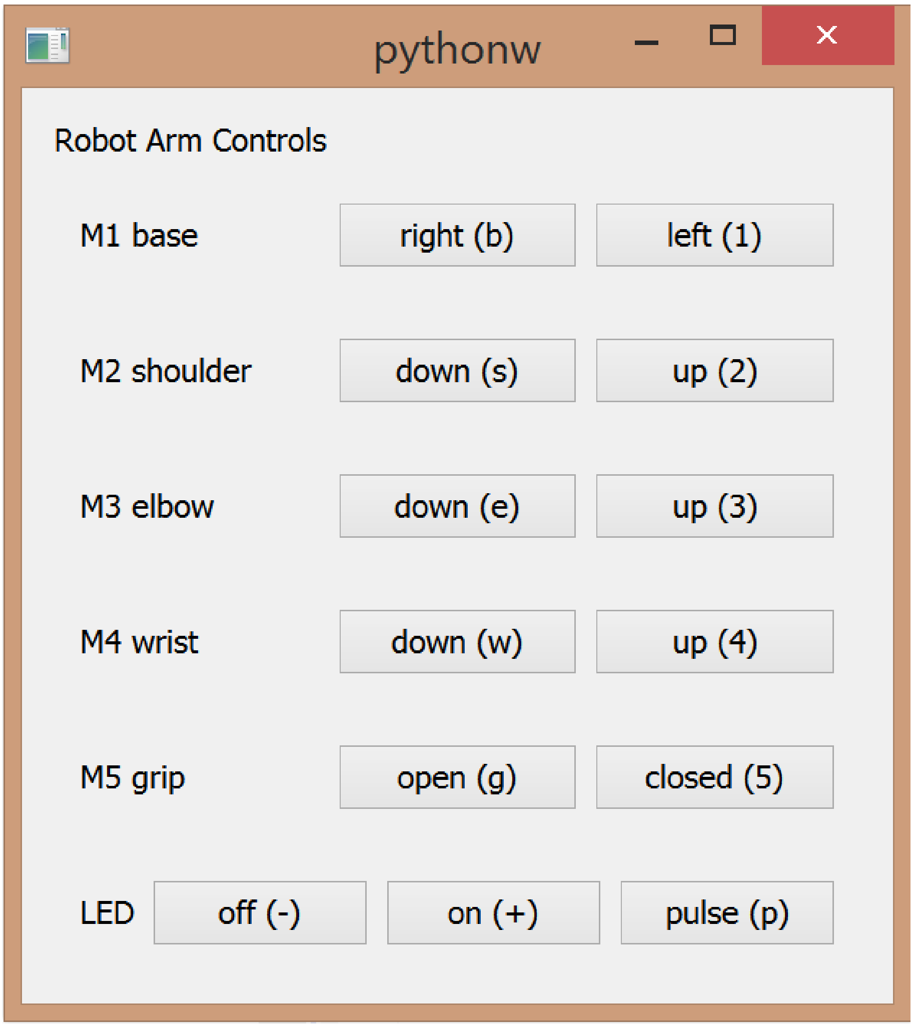

Fig. 1.6: The Python Graphical User Interface.

The control is provided in two Python modules:

robot.py interfaces with EPICS, converts move commands to PVs, provides basic workout, no GUI

gui_robot.py interfaces with *robot* module, provides the GUI

source code documentation

Source code of the Python client is provided below.

gui_robot

robot

1.3.3 Joystick - IOC support (not really a client)

The joystick is an operator interface. Controls for this interface have been implemented here within the IOC.

Keenan Lang, APS BCDA group, had developed an HMI (human-machine interface) module to allow human-machine interface devices such as mice, keyboards, and joysticks (and other) to communicate directly into an EPICS IOC. In a few hours, he added that support to the robot IOC project so that a particular joystick can be used to control the robot arm directly within the IOC.

With added joystick control in the IOC, it is not necessary to require a KVM GUI (video screen + keyboard + mouse) to operate the robot.

button	action
trigger	close grip
thumb	open grip
twist	rotate base in same direction
lever	turns on/off base rotation (useful when trying to grasp objects)
joystick	shoulder axis: forward=down, back=up
elbow	two buttons, forward and back
wrist	knob, forward=down, back=up

The joystick buttons are described in file:

- `edgeRoboArmIOC/support/usb-1-0/usbApp/Db/LogitechExtreme3DPro.in`

The actions are mapped to buttons in a database file:

- `edgeRoboArmIOC/support/xxx-5-6/xxxApp/Db/roboArm.db`
- (includes all of `edgeRoboArmIOC/support/ip-2-13/ipApp/Db/roboArm.db`)

EPICS IOC startup commands to support the joystick.

```
usbCreateDriver("JOYSTICK", "$(USB)/usbApp/Db/LogitechExtreme3DPro.in")
usbConnectDevice("JOYSTICK", 0, 0x046D, 0xC215)
dbLoadRecords("../../xxxApp/Db/roboArm.db", "P=xxx:, A=A1:, INPORT=JOYSTICK, OUTPORT=USB1")
```

The database provides the mapping between EPICS records and joystick buttons.

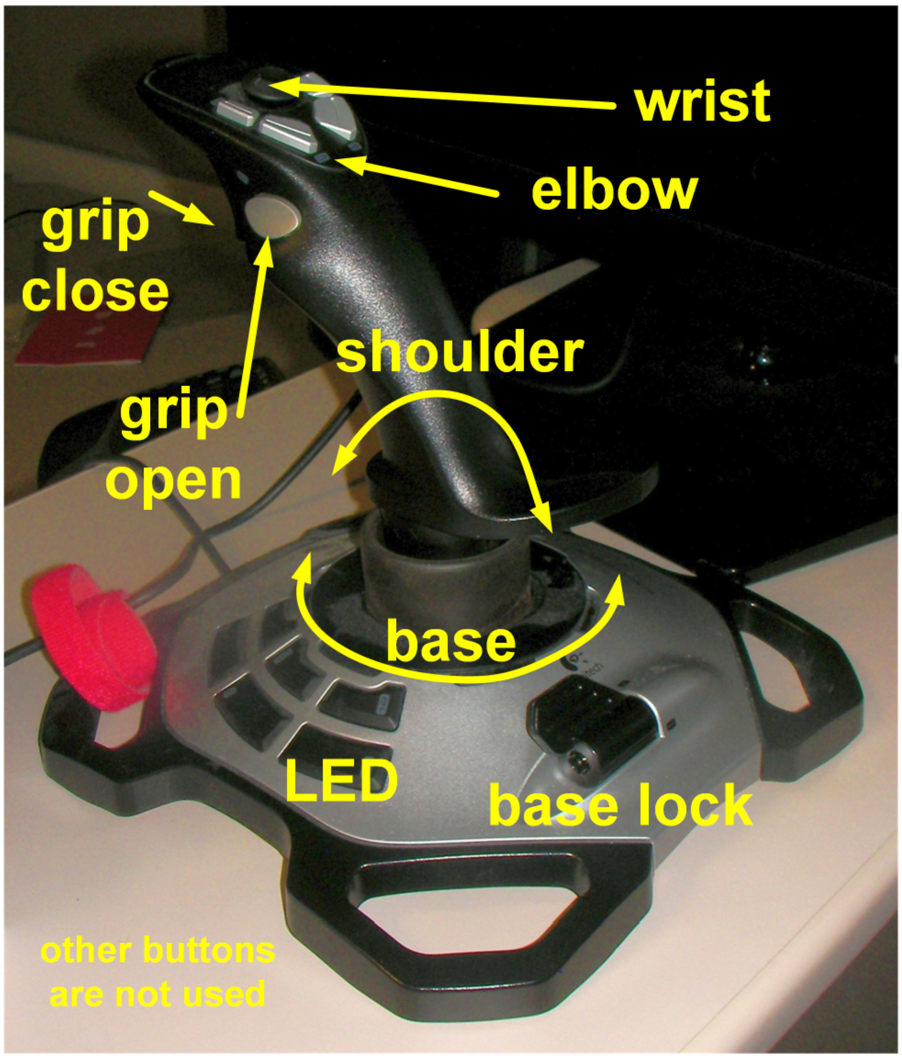

Fig. 1.7: This is the joystick we will use. (model: Logitech Extreme 3D Pro) It is right-handed and has a twist action (vertical axis). We'll use that for the base rotation.

The joystick grip buttons are mapped in a *calcout* record.

```
record(calcout, "$(P)$(A)grip_calc")
{
   field(INPA, "$(P)$(A)Trigger_State.VAL NPP")
   field(INPB, "$(P)$(A)LButton_State.VAL NPP")
   field(CALC, "2 * B + A")
   field(OUT, "$(P)$(A)grip_move PP")
}
record(bi, "$(P)$(A)Trigger_State")
{
   field(DTYP, "asynInt32")
   field(SCAN, "I/O Intr")
   field(INP, "@asyn($(INPORT), 0, 0)TRIGGER_PRESSED")
   field(FLNK, "$(P)$(A)grip_calc")
}
record(bi, "$(P)$(A)LButton_State")
{
   field(DTYP, "asynInt32")
   field(SCAN, "I/O Intr")
   field(INP, "@asyn($(INPORT), 0, 0)LBUTTON_PRESSED")
   field(FLNK, "$(P)$(A)grip_calc")
}
```

Note: To use a different joystick, you'll need to create a new file todescribe the buttons on the joystick and the values used by USB communications: `$(USB)/usbApp/Db/<new_joystick>.in`

Then, you'll need to modify the `../../xxxApp/Db/roboArm.db` file for the names of the new buttons. You might also need to update the calculation logic in the database to match your new joystick.

1.4 Examples

There are two examples to demonstrate the EPICS control of the OWI Edge Robotic Arm. For comparison, an additional example shows the sample changing robot at Advanced Photon Source beam line 11-BM. That robot, also under EPICS control, is several axes in common with the OWI Edge Robotic Arm. But, the 11-BM robot is far more advanced, including more rotation axes such as wrist twist.

1.4.1 Get the ball rolling: The Marble Tree

The robot arm can lift small objects and move them under visual control. Using a joystick and some practice, the control can become intuitive.

A great example of the robot would be to place a ball in a maze, pick it up at the end and repeat. With that in mind, a marble tree (wooden musical instrument and coffee table amusement) is ideal. The marble tree shown in the pictures was purchased in Berea, KY. [1]

Example

This system uses the robot arm, a Raspberry Pi to run the IOC, and a joystick that runs in the IOC. No GUI is necessary. It is useful to place the robot on a plinth so that it can reach top of the marble tree, as well as pick up the marble from

[1] marble tree: http://www.berea.com/appalachian-fireside-gallery/

the bin at the bottom.

Fig. 1.8: System for marble tree example

See the section *Joystick - IOC support (not really a client)* for details about the mapping of controls on the joystick.

Once the Raspberry Pi has been connected to the joystick and robot arm and the Linux system is started up, the EPICS IOC should start within two minutes. (Otherwise something is wrong. Check all the connections.) Keep in mind that the Raspberry Pi is very sensitive to changes in electrical power demand. It is best to plug everything in **before** plugging in the electrical power to the Raspberry Pi.

Pulse the LED button to ensure the IOC is operating.

Step 1

Step 2

Step 3

Step 4

Step 5

Step 6

1.4.2 Programmable Sequence

In preparation for the 2012 Argonne National Laboratory Energy Showcase (an open house for the community [1]), the BCDA group [2] created linux-based EPICS controls [3] for the robot arm to simulate how robots install samples into

[2] http://www.barryrhodes.com/2012/01/addressing-ball.html
[1] 2012 ANL Energy Showcase: https://www.flickr.com/photos/argonne/7996170862/in/album-72157631558448229
[2] BCDA: http://www.aps.anl.gov/bcda
[3] First IOC was created by Jeff Gebhardt, APS BCDA group

Fig. 1.9: Move arm into place to pick up marble. Be sure to clear all the wooden leaves!

Fig. 1.10: Address the ball. [2]

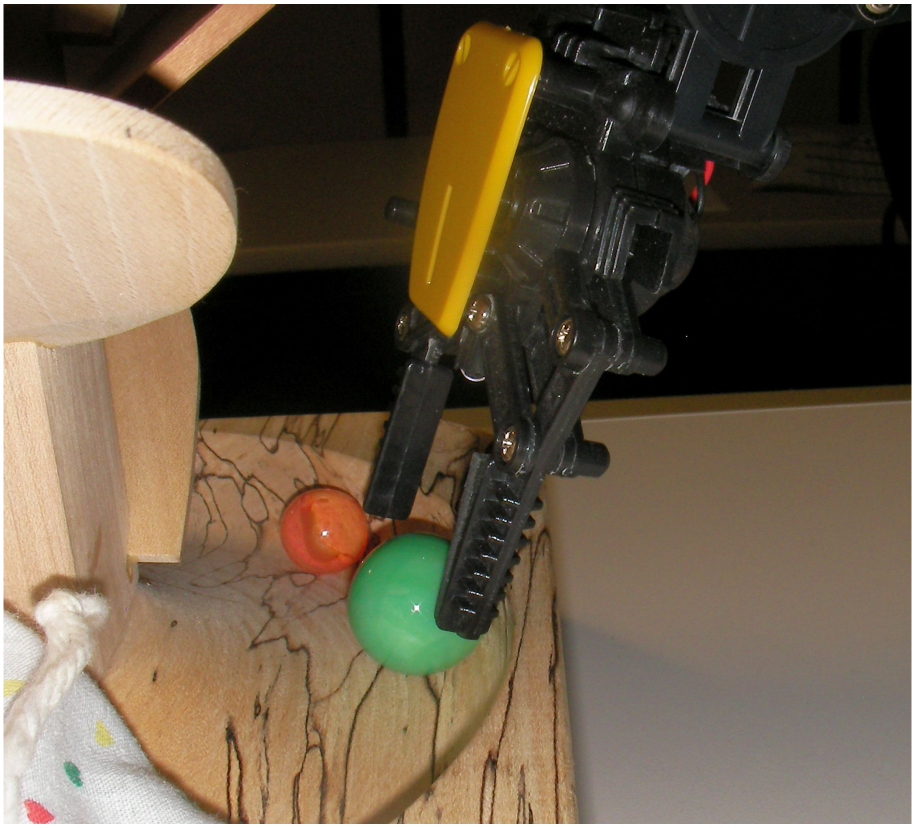

Fig. 1.11: Approach the ball with the grips open. It may help to turn on the LED to verify alignment.

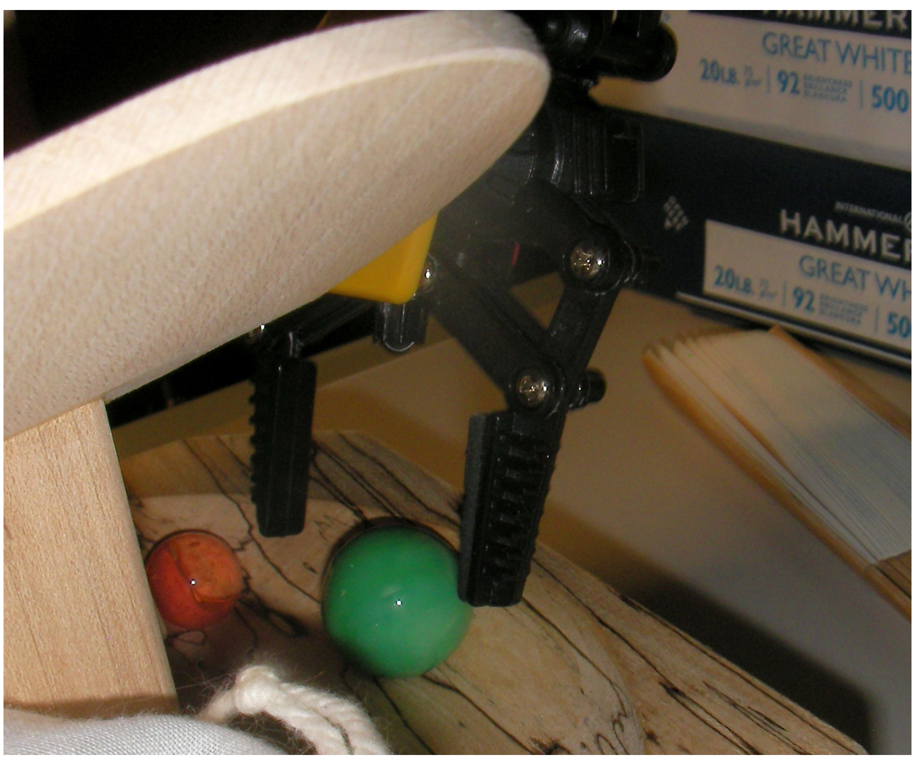

Fig. 1.12: It may be needed to nudge the ball to using the base to pick it up with the grips.

Fig. 1.13: Grip the ball until the motor stops.

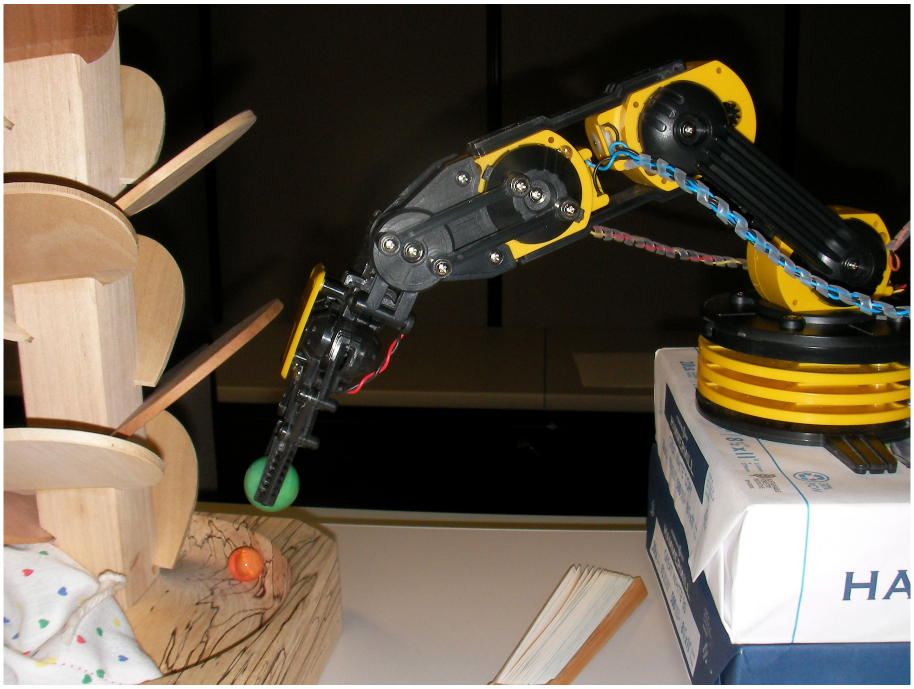

Fig. 1.14: Carefully, raise the shoulder a bit, without banging the wooden leaves. Don't knock the ball out of the grips.

Move back until the arm can clear all the leaves.

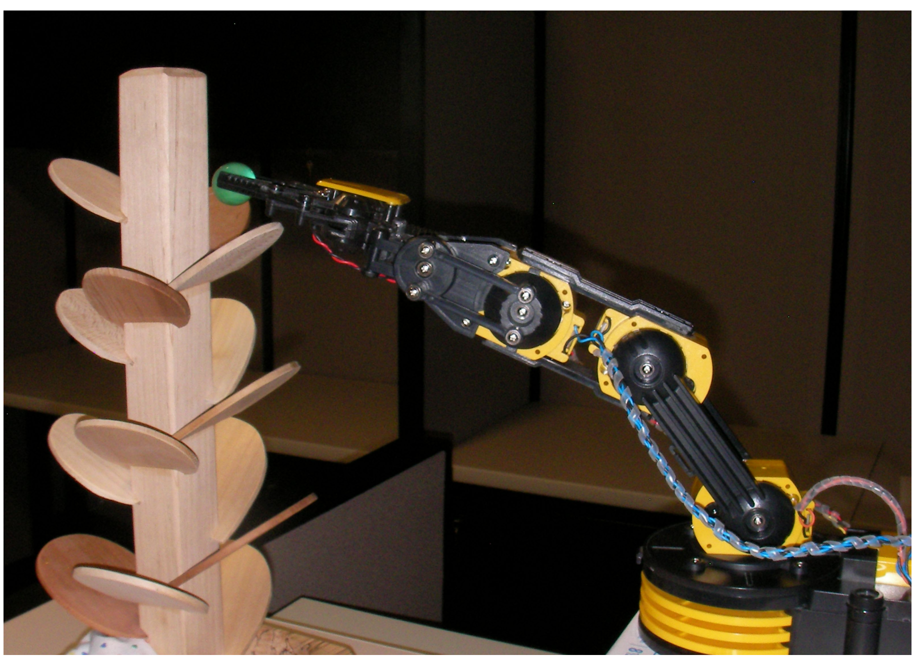

Fig. 1.15: Raise and lengthen the arm to position the ball at the top of the marble tree.

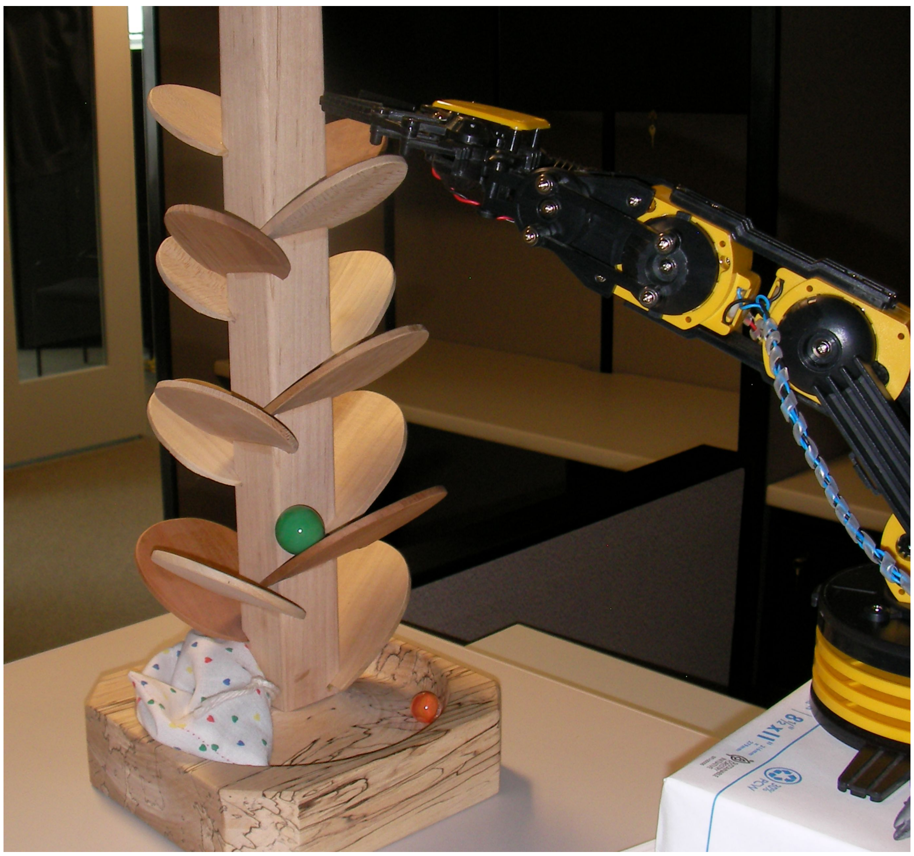

Fig. 1.16: Open the grips to release the ball. Listen as the ball moves downward.

X-ray detectors at several of the APS experiment area beamlines. The robots allow for faster sample loading and enable scientists to use the APS while at their home institutions.

Using a Raspberry Pi as the Linux IOC host and EPICS, this hands-on IOC demonstrates how modestly a "complete" control system might be constructed. A GUI can be added on the network for alternative control of the robot.

A movie of the automation sequence is available online: https://vimeo.com/128020522

Database, sequence, and GUI support are provided in this IOC project under the `ip-2-13` subdirectory.

Schematic

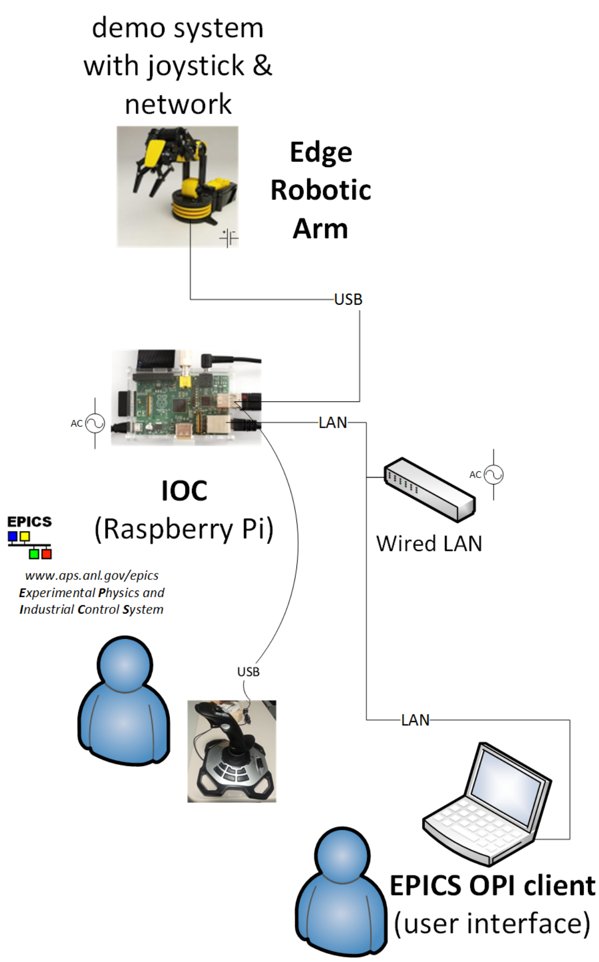

Fig. 1.17: schematic of automated sequence equipment

The sequence is run from a SNL program on the Raspberry Pi. The parameters for the sequence are entered from the EPICS OPI client ICSS BOY GUI on the laptop).

1.4.3 APS 11-BM: A real automation robot

The Advanced Photon Source beam line 11-BM sample change robot is an example of a real automation robot for X-ray science. This robot has more motorized axes, position encoders, and limit switches. The mechanical system has much lower backlash and higher lifting strength.

Also, the system has a bar code reader to identify samples before they are mounted on the instrument.

A movie of the APS 11-BM sample robot changing a sample is available online: https://www.youtube.com/watch?v=sowojskY7c4

1.4.4 2016 ANL Open House

On 2016-05-21, Argonne National Laboratory [1] hosted an open house. [2] The EPICS Edge Robot Arm was presented as a control system demonstration.

Connecting it all up

1. unplug the 5V transformer from 120 VAC power
2. plug the 5V transformer micro USB cable into the lower board (the touch screen board), that board will supply power to the RasPi
3. plug the RoboArm USB cable into the RasPi
4. plug the joystick/controller USB cable into the RasPi

Fig. 1.18: RasPi, joystick, and RoboArm connected

Power On

The power transformer is plugged in to the outlet strip to power up the RasPi.

The RoboArm USB interface has a power switch on the top of the battery box. Turn it on while the RasPi is starting.

Once the RasPi starts, the IOC should start within a minute (or two). The RoboArm LED (behind the grip) will turn on at the end of the IOC startup file, signalling the IOC has started.

If plugged in, the joystick should be able to operate. Test the LED to verify.

Start the GUI (manually)

On first logging in after startup, the GUI is supposed to start automatically, about a minute after the IOC starts. This is not working now.

Follow these steps to start the GUI from the touch screen.

[1] http://www.anl.gov
[2] http://www.anl.gov/events/argonne-open-house

Fig. 1.19: RasPi starting up after power is applied.

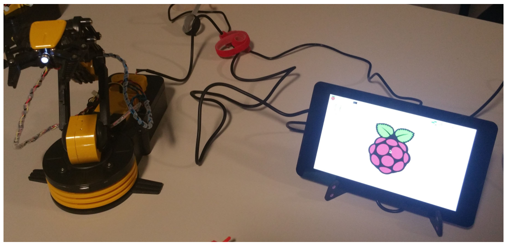

Fig. 1.20: The RoboArm LED is ON indicating the EPICS IOC has started.

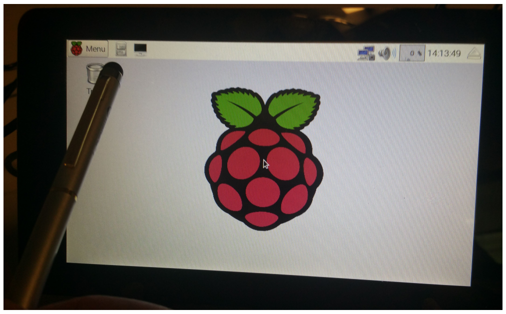

Fig. 1.21: Step 1. start the File Manager on the touch screen

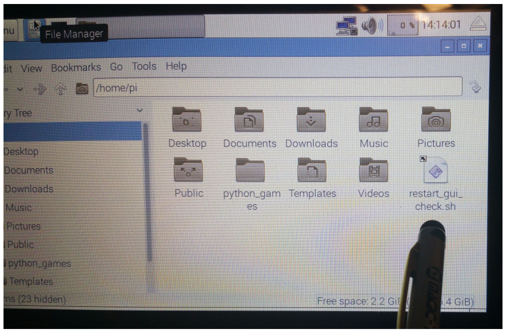

Fig. 1.22: Step 2. touch *restart_gui_check.sh*

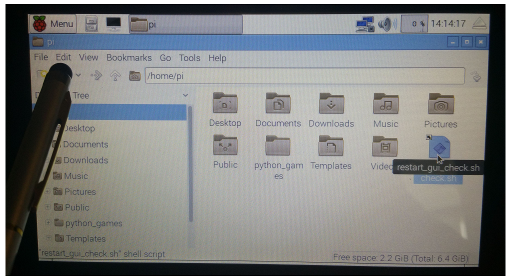

Fig. 1.23: Step 3. touch *Edit* menu

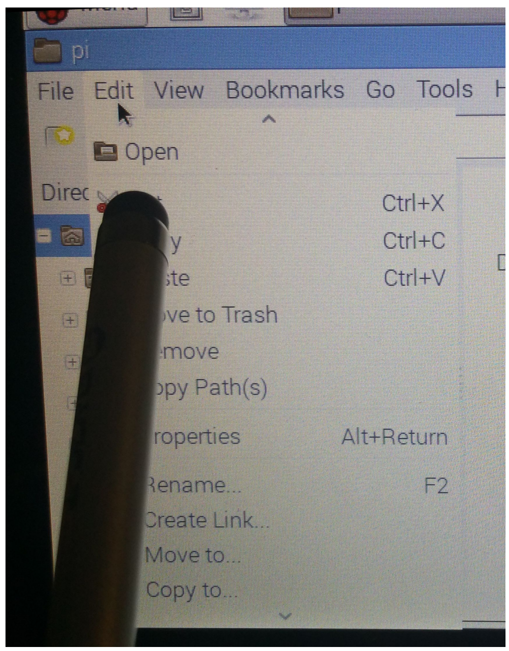

Fig. 1.24: Step 4. touch *Open* item

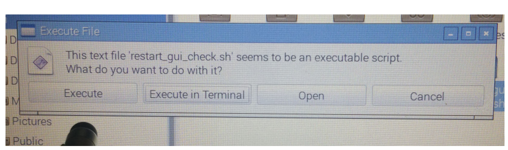

Fig. 1.25: Step 5. press *Execute* button

Python Touch Screen GUI

While the touch screen is multi-touch (the screen can indicate more than one touch at a time), the Python GUI is not prepared to handle more than one button press at a time.

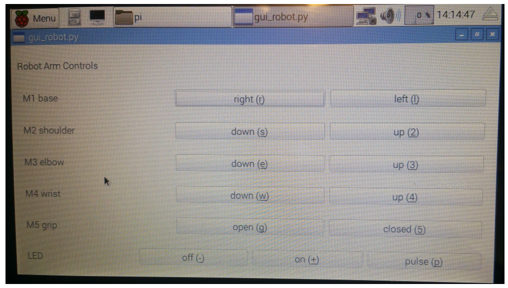

Fig. 1.26: Python Touch Screen GUI

Joystick Controls

The Robot Arm can be moved using a joystick or game controller plugged in to a USB port on the IOC. An EPICS *substitutions* file should be prepared for each new type of joystick, to map the buttons to the EPICS database actions.

The IOC is prepared for the joystick to be hot-swapped (unplugged, changed to a different one, ...

If you plug in a second recognized joystick, it will also work. If it is the same type as an existing plugged-in joystick, the second will be ignored unless you create additional EPICS support. But, who would ever do that?

Logitech Attack III joystick

This is a common joystick that was hanging around the house, waiting for something to do.

Fig. 1.27: image of Logitech Attack III joystick

Button Map The substitutions file describes the mapping of the buttons to EPICS actions:

```
file "$(USB)/usbApp/Db/AnalogAxis.template"
{
pattern
{P     R                PARAM               PORT          FLNK}
{xxx:  ATK:Vertical      VERTICAL_STATE      LOGITECH_ATK      xxx:ATK:shoulder}
{xxx:  ATK:Switch        SWITCH_STATE        LOGITECH_ATK      xxx:ATK:move_lock}
}

file "$(USB)/usbApp/Db/DigitalButton.template"
{
pattern
{P     R                PARAM               PORT          FLNK}
{xxx:  ATK:Trigger       TRIGGER_PRESSED     LOGITECH_ATK      xxx:ATK:led}
{xxx:  ATK:Button2       BUTTON2_PRESSED     LOGITECH_ATK      xxx:ATK:wrist}
{xxx:  ATK:Button3       BUTTON3_PRESSED     LOGITECH_ATK      xxx:ATK:wrist}
{xxx:  ATK:Button4       BUTTON4_PRESSED     LOGITECH_ATK      xxx:ATK:grip}
{xxx:  ATK:Button5       BUTTON5_PRESSED     LOGITECH_ATK      xxx:ATK:grip}
{xxx:  ATK:Button6       BUTTON6_PRESSED     LOGITECH_ATK      xxx:ATK:elbow}
{xxx:  ATK:Button7       BUTTON7_PRESSED     LOGITECH_ATK      xxx:ATK:elbow}
{xxx:  ATK:Button8       BUTTON8_PRESSED     LOGITECH_ATK      xxx:ATK:base}
{xxx:  ATK:Button9       BUTTON9_PRESSED     LOGITECH_ATK      xxx:ATK:base}
}

file "$(TOP)/iocBoot/$(IOC)/substitutions/AxisMove.template"
{
pattern
{P     R                LOCK                 AXIS              DEAD_LOW   DEAD_HIGH   OUT}
{xxx:  ATK:move_lock     0                    xxx:ATK:Switch    0          126         ""}
{xxx:  ATK:shoulder      xxx:ATK:move_lock    xxx:ATK:Vertical  100        150         xxx:A1:shoulder_
}

file "$(TOP)/iocBoot/$(IOC)/substitutions/ButtonMove.template"
{
pattern
{P     R                 LOCK               BUTTONA            BUTTONB           OUT}
{xxx:  ATK:led            0                  0                  xxx:ATK:Trigger    xxx:A1:led_onoff}
{xxx:  ATK:grip           xxx:ATK:move_lock  xxx:ATK:Button4     xxx:ATK:Button5    xxx:A1:grip_move}
{xxx:  ATK:elbow          xxx:ATK:move_lock  xxx:ATK:Button6     xxx:ATK:Button7    xxx:A1:elbow_move
{xxx:  ATK:base           xxx:ATK:move_lock  xxx:ATK:Button8     xxx:ATK:Button9    xxx:A1:base_move}
{xxx:  ATK:wrist          xxx:ATK:move_lock  xxx:ATK:Button2     xxx:ATK:Button3    xxx:A1:wrist_move
}
```

Logitech Dual Action Pro game controller

This is a common game controller that was hanging around the house, waiting for something to do.

Button Map The substitutions file describes the mapping of the buttons to EPICS actions:

```
file "$(USB)/usbApp/Db/AnalogAxis.template"
{
pattern
{P     R                PARAM               PORT          FLNK}
{xxx:  DUAL:Vertical      LSTICK_UD_STATE     LOGITECH_DUAL      xxx:DUAL:shoulder}
{xxx:  DUAL:Rotation      LSTICK_LR_STATE     LOGITECH_DUAL      xxx:DUAL:base}
}

```

Fig. 1.28: Logitech Dual Action Pro game controller

Fig. 1.29: front buttons of Logitech Dual Action Pro game controller

```
file "$(USB)/usbApp/Db/DigitalButton.template"
{
pattern
{P      R                PARAM                PORT           FLNK}
{xxx:   DUAL:Button1       BUTTON1_PRESSED     LOGITECH_DUAL      xxx:DUAL:led}
{xxx:   DUAL:Button2       BUTTON2_PRESSED     LOGITECH_DUAL      xxx:DUAL:move_lock}
{xxx:   DUAL:Button3       BUTTON3_PRESSED     LOGITECH_DUAL      xxx:DUAL:grip}
{xxx:   DUAL:Button4       BUTTON4_PRESSED     LOGITECH_DUAL      xxx:DUAL:grip}
{xxx:   DUAL:Button5       BUTTON5_PRESSED     LOGITECH_DUAL      xxx:DUAL:elbow}
{xxx:   DUAL:Button7       BUTTON7_PRESSED     LOGITECH_DUAL      xxx:DUAL:elbow}
{xxx:   DUAL:Button6       BUTTON6_PRESSED     LOGITECH_DUAL      xxx:DUAL:wrist}
{xxx:   DUAL:Button8       BUTTON8_PRESSED     LOGITECH_DUAL      xxx:DUAL:wrist}
}

file "$(TOP)/iocBoot/$(IOC)/substitutions/AxisMove.template"
{
pattern
{P      R                LOCK               AXIS              DEAD_LOW   DEAD_HIGH   OUT}
{xxx:   DUAL:move_lock     0                  xxx:DUAL:Button2   1           1           ""}
{xxx:   DUAL:base          xxx:DUAL:move_lock   xxx:DUAL:Rotation  100         150          xxx:A1:base_
{xxx:   DUAL:shoulder      xxx:DUAL:move_lock   xxx:DUAL:Vertical  100         150          xxx:A1:shoul
}

file "$(TOP)/iocBoot/$(IOC)/substitutions/ButtonMove.template"
{
pattern
{P      R                 LOCK                 BUTTONA              BUTTONB           OUT}
{xxx:   DUAL:led            0                    0                   xxx:DUAL:Button1    xxx:A1:led_onoff}
{xxx:   DUAL:elbow          xxx:DUAL:move_lock  xxx:DUAL:Button7     xxx:DUAL:Button5    xxx:A1:elbow_
{xxx:   DUAL:grip           0                  xxx:DUAL:Button3     xxx:DUAL:Button4    xxx:A1:grip_move
{xxx:   DUAL:wrist          xxx:DUAL:move_lock  xxx:DUAL:Button8     xxx:DUAL:Button6    xxx:A1:wrist_
}
```

Logitech Extreme 3D Pro joystick

This joystick has a twist action that makes it good for controlling the robot arm.

Fig. 1.30: Logitech Extreme 3D Pro joystick

Button Map The substitutions file describes the mapping of the buttons to EPICS actions:

```
file "$(USB)/usbApp/Db/AnalogAxis.template"
{
pattern
{P     R                PARAM               PORT          FLNK}
{xxx:  PRO:Vertical      VERTICAL_STATE      LOGITECH_3DPRO      xxx:PRO:shoulder}
{xxx:  PRO:Rotation      ROTATION_STATE      LOGITECH_3DPRO      xxx:PRO:base}
{xxx:  PRO:Switch        SWITCH_STATE        LOGITECH_3DPRO      xxx:PRO:move_lock}
{xxx:  PRO:Hat           HAT_STATE           LOGITECH_3DPRO      xxx:PRO:wrist}
}

file "$(USB)/usbApp/Db/DigitalButton.template"
{
pattern
{P     R                PARAM               PORT          FLNK}
{xxx:  PRO:Trigger       TRIGGER_PRESSED     LOGITECH_3DPRO      xxx:PRO:grip}
{xxx:  PRO:LButton       LBUTTON_PRESSED     LOGITECH_3DPRO      xxx:PRO:grip}
{xxx:  PRO:Button3       BUTTON3_PRESSED     LOGITECH_3DPRO      xxx:PRO:elbow}
{xxx:  PRO:Button5       BUTTON5_PRESSED     LOGITECH_3DPRO      xxx:PRO:elbow}
{xxx:  PRO:Button11      BUTTON11_PRESSED    LOGITECH_3DPRO      xxx:PRO:led}
}

file "$(TOP)/iocBoot/$(IOC)/substitutions/AxisMove.template"
{
pattern
{P     R                LOCK               AXIS              DEAD_LOW   DEAD_HIGH   OUT}
{xxx:  PRO:move_lock     0                  xxx:PRO:Switch    0          126          ""}
{xxx:  PRO:base          xxx:PRO:move_lock   xxx:PRO:Rotation  50         200          xxx:A1:base_move
{xxx:  PRO:shoulder      xxx:PRO:move_lock   xxx:PRO:Vertical  50         600          xxx:A1:shoulder_
}

file "$(TOP)/iocBoot/$(IOC)/substitutions/ButtonMove.template"
{
pattern
{P     R                 LOCK               BUTTONA            BUTTONB           OUT}
{xxx:  PRO:led            0                  0                  xxx:PRO:Button11   xxx:A1:led_onoff}
{xxx:  PRO:elbow          xxx:PRO:move_lock  xxx:PRO:Button3     xxx:PRO:Button5    xxx:A1:elbow_move
{xxx:  PRO:grip           xxx:PRO:move_lock  xxx:PRO:LButton     xxx:PRO:Trigger    xxx:A1:grip_move}
}

file "$(TOP)/iocBoot/$(IOC)/substitutions/DiscreteMove.template"
{
pattern
{P     R                 LOCK               AXIS            VALA    VALB     OUT}
{xxx:  PRO:wrist          xxx:PRO:move_lock   xxx:PRO:Hat      4       0        xxx:A1:wrist_move}
}
```

IOC Preparation

A Raspberry Pi 3 (RasPi3) was setup with linux raspbian jessie operating system on a 16 GB micro SD card (an 8GB card would be fine just as well). Once the system was setup and operations verified, the SD card was imaged for use on several systems. The units chosen for the demo were Raspberry Pi 2 (which can run the same software and architecture build).

For the open house demo system, a 7" touch screen was configured so that the computer will run without keyboard or mouse.

EPICS base release 3.14.12.5 [3] was installed (and built) in `/usr/local/epics/base`. The EPICS Edge Robot Arm project software [4] was installed using a `git clone` command in the */usr/local/epics*' directory. To facilitate configuration already in the project, a soft link was made as follows:

```
cd /usr/local/epics
ln -s epicsEdgeRoboArm/edgeRoboArmIOC  ./edgeRoboArmIOC
cd edgeRoboArmIOC/support
make release
make
```

Various packages were installed (using `sudo apt-get install <package>`) so that the build was successful. Among these:

- re2c
- libreadline
- libreadline-dev
- libusb-1.0.0
- libusb-1.0.0-dev

The IOC *must* be run by the root user to communicate through the USB port. The Python GUI can be run by the *pi* user (it communicates with the EPICS IOC using EPICS Channel Access protocol). Both of these are started by cron tasks. Each cron task checks every minute to see if its assigned process is not already running and that appropriate resources are available:

- IOC: the USB from the robot is plugged in and the robot arm is powered on
- GUI: the IOC is started

note: The GUI task is not starting from its cron task. The startup script must be run manually (see *Start the GUI (manually)*).

1.5 CHANGES

2015-05-16 source code moved to new GitHub account: https://github.com/bcda-aps/epicsEdgeRoboArm.git

2012-09-13 1.0 - original release

1.6 Credits

original EPICS IOC emplementation Jeff Gebhardt, APS BCDA group

joystick controls Keenan Lang, APS BCDA group

multitouch control idea Katherine Jemian

CSS BOY control screen BCDA summer students

python GUI Pete Jemian, APS BCDA group

[3] http://www.aps.anl.gov/epics/base/R3-14/12.php
[4] https://github.com/BCDA-APS/epicsEdgeRoboArm

1.7 Software License

```
Copyright (c) 2005 University of Chicago and the Regents of the University of
California. All rights reserved.

synApps is distributed subject to the following license conditions:
SOFTWARE LICENSE AGREEMENT
Software: synApps
Versions: Release 4-5 and higher.

   1. The "Software", below, refers to synApps (in either source code, or
      binary form and accompanying documentation). Each licensee is addressed
          as "you" or "Licensee."

   2. The copyright holders shown above and their third-party licensors hereby
      grant Licensee a royalty-free nonexclusive license, subject to the
          limitations stated herein and U.S. Government license rights.

   3. You may modify and make a copy or copies of the Software for use within
      your organization, if you meet the following conditions:
         1. Copies in source code must include the copyright notice and this
                    Software License Agreement.
         2. Copies in binary form must include the copyright notice and this
                    Software License Agreement in the documentation and/or other
                        materials provided with the copy.

   4. You may modify a copy or copies of the Software or any portion of it, thus forming a work based
         1. Copies in source code must include the copyright notice and this
                    Software License Agreement;
         2. Copies in binary form must include the copyright notice and this
                    Software License Agreement in the documentation and/or other
                        materials provided with the copy;
         3. Modified copies and works based on the Software must carry
                    prominent notices stating that you changed specified portions of
                        the Software.

   5. Portions of the Software resulted from work developed under a
      U.S. Government contract and are subject to the following license:
          the Government is granted for itself and others acting on its behalf a
          paid-up, nonexclusive, irrevocable worldwide license in this computer
          software to reproduce, prepare derivative works, and perform publicly and
          display publicly.

   6. WARRANTY DISCLAIMER. THE SOFTWARE IS SUPPLIED "AS IS" WITHOUT WARRANTY OF
      ANY KIND. THE COPYRIGHT HOLDERS, THEIR THIRD PARTY LICENSORS, THE UNITED
          STATES, THE UNITED STATES DEPARTMENT OF ENERGY, AND THEIR EMPLOYEES: (1)
          DISCLAIM ANY WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
          ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
          PURPOSE, TITLE OR NON-INFRINGEMENT, (2) DO NOT ASSUME ANY LEGAL LIABILITY
          OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR USEFULNESS OF THE
          SOFTWARE, (3) DO NOT REPRESENT THAT USE OF THE SOFTWARE WOULD NOT
          INFRINGE PRIVATELY OWNED RIGHTS, (4) DO NOT WARRANT THAT THE SOFTWARE WILL
          FUNCTION UNINTERRUPTED, THAT IT IS ERROR-FREE OR THAT ANY ERRORS WILL BE
          CORRECTED.

   7. LIMITATION OF LIABILITY. IN NO EVENT WILL THE COPYRIGHT HOLDERS, THEIR
      THIRD PARTY LICENSORS, THE UNITED STATES, THE UNITED STATES DEPARTMENT OF
          ENERGY, OR THEIR EMPLOYEES: BE LIABLE FOR ANY INDIRECT, INCIDENTAL,
```

CHAPTER 2

Indices and tables

- genindex
- search

This documentation was built May 17, 2016

---

release 1.0

published May 17, 2016

Index

C

E

J

L

T