epicsEdgeRoboArm Documentation

Release 1.0

Jeff Gebhardt, Pete Jemian

Sep 27, 2017

Contents

EPICS support for the OWI Edge Robotic Arm over USB

Note: This project is under construction and is not ready

release 1.0

published Sep 27, 2017

maintainer Pete R. Jemian

email jemian@anl.gov

docs http://epicsEdgeRoboArm.readthedocs.org

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

CHAPTER 1

Contents

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

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

- 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:

- 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 and posted online by a third party, enabling communication from a Linux computer.

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.

2012 ANL Energy Showcase - First Demo of EPICS IOC

In preparation for the 2012 Argonne National Laboratory Energy Showcase (an open house for the community[3]), the BCDA group [#] created linux-based EPICS controls[4] for the robot arm to simulate how robots install samples into 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.

[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

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

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.

Demo System with joystick

Recently, USB joystick control was added to the IOC[5] which enables truly headless (no GUI needed) operations. In the photo here, a wooden marble tree instrument[6] is used to provide an interesting target for the robot arm actions.

photo

Fig. 1.3: photograph of demo system, with marble tree instrument

---

EPICS IOC

The IOC must be run as root. EPICS base 3.1412.1 (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.

[5] BCDA: http://www.aps.anl.gov/bcda

[6] First IOC was created by Jeff Gebhardt, APS BCDA group

[1] synApps: http://www.aps.anl.gov/bcda/synApps/

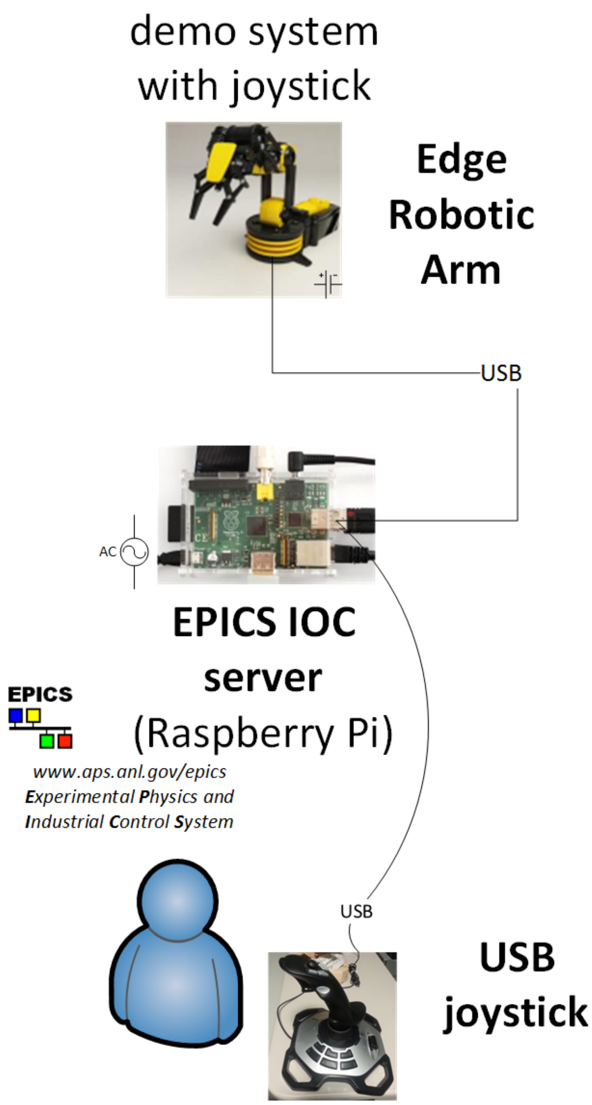

Fig. 1.4: Basic connection schematic for headless IOC and operation using a joystick

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")
```

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*

```
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")
}
```

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`.

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

```
#!/bin/bash

# restart_ioc_check.sh

# 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

```

```

export EPICS_HOST_ARCH=linux-arm

# 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 [ "${ioc_off}" != "" ]; then
    #echo "IOC is not running"
    #/usr/local/epics/base/bin/linux-arm/caRepeater &
    /usr/local/epics/base/bin/${EPICS_HOST_ARCH}/caRepeater &

    cd /usr/local/epics/synAppsRobo/support/xxx-5-6/iocBoot/iocLinux/
    ./in-screen.sh
  fi
fi
```

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.

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.

Joystick support

Keenan Lang, APS BCDA group, had developed an HMI 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.

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.

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:

CSS BOY client

Python - PyEpics and PyQt4 client

Joystick - IOC support (not really a client)

Examples

Get the ball rolling: The Marble Tree

Programmable Sequence

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

[1] 2012 ANL Energy Showcase: https://www.flickr.com/photos/argonne/7996170862/in/album-72157631558448229

[2] BCDA: http://www.aps.anl.gov/bcda

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.

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, positione 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

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

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

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."
```

CHAPTER 2

Indices and tables

- genindex
- search

This documentation was built Sep 27, 2017

Index