Intel Do-It-Yourself Challenge
Pololu USB Controller

Nicolas Vailliet
www.Intel-Software-Academic-Program.com
paul.guermonprez@intel.com
Intel Software
2014-02-01

Pololu Servo controller

Controlling servos and more

Pololu card allows you to control up to 24 servo motors. Interfaced with USB or Serial port, this card adapts targeted position, speed and acceleration according to your commands. Each servo is connected to a channel and can be controlled separately. An external source of power is required.

intel Software

We'll use …

OS
Ubuntu 12.04 LTS (other Oses are supported)

An Intel Galileo board and network connectivity
We assume you know how to communicate with the embedded OS and performing GPIO commands.

A Pololu Micro Maestro 6-channel USB servo controller
We also need a **Mini-A Mini-B USB cable**.

A full Clanton Linux image
This will provide development tools and libusb-1.0 to use USB port.

The cross-compile toolchain for Intel Quark on your computer
This is a good practice. Even if our sample codes do not link to many libraries, Intel Galileo is not a compilation platform. Compile your C/C++ program with the cross-compile toolchain on your computer instead of using g++ on your board. See our other courses to do so.

intel Software

Installation

Installation

- Boot your board and wait for the USB LED to be on.
- Connect the USB cable on **host** USB port.
- Connect the Pololu controller at the other extremity.

intel Software

Installation

Connect to your board with SSH.
Execute

cat /sys/kernel/debug/usb/devices

You should see product name : Pololu Micro Maestro
Keep in mind or write down the ProdID. Here it's 0x89 or 137.

```
T:  Bus=01 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#=  3 Spd=12   MxCh= 0
D:  Ver= 2.00 Cls=ef(misc ) Sub=02 Prot=01 MxPS= 8 #Cfgs=  1
P:  Vendor=1ffb ProdID=0089 Rev= 1.01
S:  Manufacturer=Pololu Corporation
S:  Product=Pololu Micro Maestro 6-Servo Controller
S:  SerialNumber=00061515
C:* #Ifs= 5 Cfg#= 1 Atr=c0 MxPwr=100mA
A:  FirstIf#= 0 IfCount= 2 Cls=02(comm.) Sub=02 Prot=01
A:  FirstIf#= 2 IfCount= 2 Cls=02(comm.) Sub=02 Prot=01
I:* If#= 0 Alt= 0 #EPs= 1 Cls=02(comm.) Sub=02 Prot=01 Driver=cdc_acm
E:  Ad=81(I) Atr=03(Int.) MxPS=  10 Ivl=1ms
I:* If#= 1 Alt= 0 #EPs= 2 Cls=0a(data ) Sub=00 Prot=00 Driver=cdc_acm
E:  Ad=02(O) Atr=02(Bulk) MxPS=   8 Ivl=0ms
E:  Ad=82(I) Atr=02(Bulk) MxPS=   8 Ivl=0ms
I:* If#= 2 Alt= 0 #EPs= 1 Cls=02(comm.) Sub=02 Prot=01 Driver=cdc_acm
E:  Ad=83(I) Atr=03(Int.) MxPS=  10 Ivl=1ms
I:* If#= 3 Alt= 0 #EPs= 2 Cls=0a(data ) Sub=00 Prot=00 Driver=cdc_acm
E:  Ad=04(O) Atr=02(Bulk) MxPS=   8 Ivl=0ms
E:  Ad=84(I) Atr=02(Bulk) MxPS=   8 Ivl=0ms
I:* If#= 4 Alt= 0 #EPs= 0 Cls=ff(vend.) Sub=04 Prot=01 Driver=(none)
root@clanton:~#
```

intel Software

USB communication with libusb

libusb

A powerful library

libusb is a very convenient library to enable USB support in your program. It is available in C++ and many common used languages. http://www.libusb.org/

Let's take a look at a sample code!

The following code is generic and show you how to use the library. Don't hesitate to take a look at the library's online documentation.

Libusb code sample

```
#include <libusb-1.0/libusb.h>                  // we include library's headers
using namespace std;

int main(int argc, char *argv[]){

        libusb_device **devs;                   // will contain USB devices
        libusb_context *ctx = NULL;             // will contain the USB context
        libusb_device_descriptor descr;         // will a device's descriptor

        // a pointer to the USB device you want to connect to
        libusb_device *mydevice;

        int r;                                  // to get error code to debug
        ssize_t cnt;                            // number of USB devices detected

        r = libusb_init(&ctx);                  // initialize USB communication
        libusb_set_debug(ctx, 3);               // set debug level to info
        cnt = libusb_get_device_list(ctx, &devs);   // get device list
```

libusb code sample

```
int found = 0;        //a boolean to know if we found the device we want
ssize_t i;
for(i = 0; i < cnt; i++) {

        //for each device, we read the descriptor file
        r = libusb_get_device_descriptor(devs[i],&descr);

        if(descr.idProduct  == XXX)   //if this is the product number we want
        {
                mydevice = (devs[i]);                   //keep this device
                found = 1;
                break;
        }
        libusb_unref_device(devs[i]);
        //free allocated memory for getting access to device list and descriptors
}
```

libusb code sample

```
if(found)
{
        //table of handler (to handle USB endpoints, USB channels
        libusb_device_handle **handle;

        //Opening our device
        r = libusb_open(mydevice,handle);

        //Just in case, we want to be sure that Linux kernel is not trying to handle this USB device
        r = libusb_detach_kernel_driver(*handle,0);
        if(r < 0 && r != LIBUSB_ERROR_NOT_FOUND) {
                cout<<"Error: detach kernel driver failed"<<endl;
                return 1;
        }

        //We tell the system we are going to manage this USB interface
        r = libusb_claim_interface(*handle,0);

        //Add your stuff here
        //For example, performing a control transfer
        r = libusb_control_transfer(            ...             );
```

libusb code sample

```
        //When you're done, release the interface
        r = libusb_release_interface(*handle,0);
        //and close the USB channel
        libusb_close(*handle);
}

//to free allocated memory, free device list and USB context
libusb_free_device_list(devs, 1);
libusb_exit(ctx);

//End the program
return 0;
}
```

Pololu USB controller

Pololu USB controller

Next slides gather three different programs.

These programs ask the Pololu controller to perform a single action. It opens the USB channel, performs a control transfer thru it, and closes it.

The three programs concern:

- Setting up the target position of a servo motor
- Setting up the speed of a servo motor
- Setting up the acceleration of a servo motor

Here is the code and we explain how to run it right after.

Pololu USB controller

Set_target.cpp

```
#include <iostream>
#include <string>
#include <libusb-1.0/libusb.h>
#include <unistd.h>
#include <stdlib.h>

using namespace std;

int main(int argc, char *argv[]){

		if(argc < 3)
		cout << "Usage: binary servo_number target_value" << endl;

		int servo = atoi(argv[1]);		//First parameter to give is the servo channel number
		int value = atoi(argv[2]);		//Second one is the value

		libusb_device **devs;
		libusb_context *ctx = NULL;
		libusb_device_descriptor descr;
		libusb_device *pololu;

		int r;
		int found = 0;
		ssize_t cnt;

		 r = libusb_init(&ctx);
		libusb_set_debug(ctx, 3);
		cnt = libusb_get_device_list(ctx, &devs);
```

Pololu USB controller

```
ssize_t i;
for(i = 0; i < cnt; i++) {
        r = libusb_get_device_descriptor(devs[i],&descr);

        if(descr.idProduct  == 137)          //here is the prodID you kept in mind
                                             //use decimal or hex value, it's up to you
        {
                pololu = (devs[i]);
                found = 1;
                break;
        }
        libusb_unref_device(devs[i]);
}

if(found)
{
        libusb_device_handle **handle;
        r = libusb_open(pololu,handle);
        r = libusb_detach_kernel_driver(*handle,0);
        if(r < 0 && r != LIBUSB_ERROR_NOT_FOUND) {
                cout<<"Error: detach kernel driver failed"<<endl;
                return 1;
        }
        r = libusb_claim_interface(*handle,0);
```

Pololu USB controller

```
//set position
        r = libusb_control_transfer(   *handle,
                                       0x40,          //request type
                                       0x85,          //request
                                       value,         //value
                                       servo,         //servo number
                                       NULL,
                                       0,
                                       5000       );

//0x85 is the request corresponding to a set target position action
r = libusb_release_interface(*handle,0);
libusb_close(*handle);
}

libusb_free_device_list(devs, 1);
libusb_exit(ctx);
return 0;
}
```

Pololu USB controller

Set_accel and Set_speed

For the two other programs, copy your set_target.cpp file and to set_accel.cpp and set_speed.cpp. Replace the control transfer function call by these ones.

Set_accel

```
r = libusb_control_transfer(      *handle, 0x40,
                                           0x84,
                                           value,
                                           servo | 0x80,
                                           NULL,0,5000);
```

Set_speed

```
r = libusb_control_transfer(      *handle, 0x40,
                                           0x84,
                                           value,
                                           servo,
                                           NULL,0,5000);
```

Compiling and running

Embedded GCC command to compile with libusb

```
g++ set_target.cpp `pkg-config  --libs libusb-1.0` -o set_target
g++ set_speed.cpp `pkg-config  --libs libusb-1.0` -o set_speed
g++ set_accel.cpp `pkg-config  --libs libusb-1.0` -o set_accel
```

With the cross compile toolchain

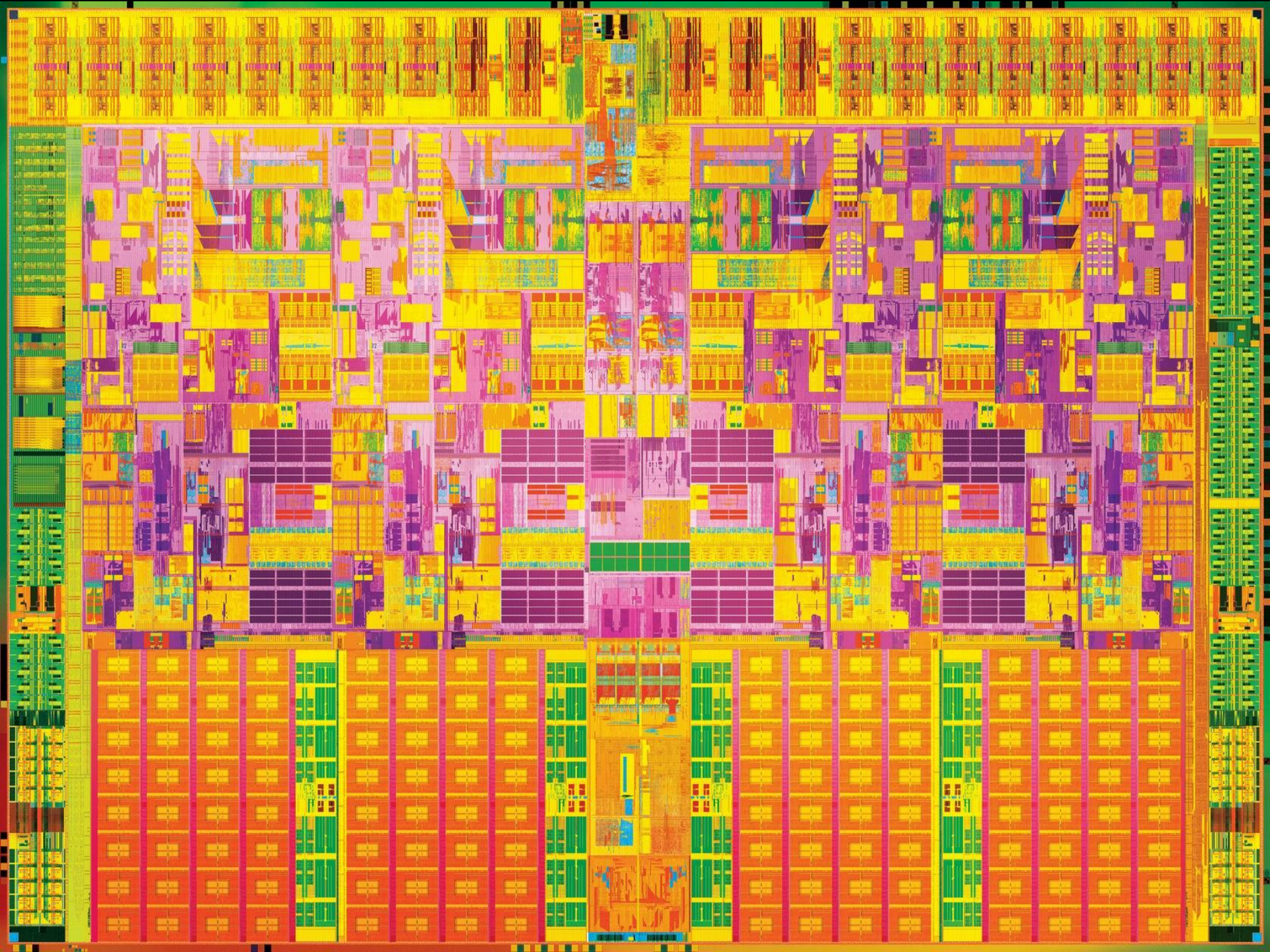
```
${CXX} set_target.cpp `pkg-config  --libs libusb-1.0` -o set_target
```

And then, send the binary file to your board.

Running these programs

```
# To set target position to 1500 microseconds (multiply by 4)
# on channel (servo) number 3.
./set_target   3   6000
# Setting speed value to mininum on servo number 1.
./set_speed   1   995
```

intel Software



License Creative Commons – By 3.0

You are free:

- **to Share** — to copy, distribute and transmit the work
- **to Remix** — to adapt the work
- to make commercial use of the work

Under the following conditions:

- **Attribution** — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

With the understanding that:

- **Waiver** — Any of the above conditions can be waived if you get permission from the copyright holder.
- **Public Domain** — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
- **Other Rights** — In no way are any of the following rights affected by the license:
  - Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
  - The author's moral rights;
  - Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.
- **Notice** — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

http://creativecommons.org/licenses/by/3.0/