Getting started with the LeanXCam

Adrien Briod – ver. 0.3 – 09.03.2012

The LeanXCam is a fairly user-friendly way to approach embedded linux and image processing. The code and hardware are open-source, and the wiki at https://github.com/scs/leanXcam/wiki/ offers quite a lot of information. Still, the first steps can seem a bit obscure if one isn’t too familiar with Linux or C compiling, etc. This document aims at getting you started from (almost) zero. If something isn’t clear, don’t hesitate to contact the author.

First steps

First, follow the guide “First steps” available here: https://github.com/scs/leanXcam/wiki/Getting-started. This will allow you to communicate with your LeanXcam, see the first images from the camera, and install the SDK.

A few notes and reminders related to this guide:

- You have to configure Windows so that it uses a fixed IP, in order to communicate with the camera through ethernet.
- Type the address 192.168.1.10 in your browser in order to access the web-view application that displays what the camera sees on a webpage.
- I suggest to use Putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/) to launch an SSH connection with the camera.
  Note: On Windows 7, if you want to use Telnet follow this advice to enable it: http://visibleprocrastinations.wordpress.com/2009/06/15/enable-telnet-in-windows-7/
- The login information for an SSH to 192.168.1.10 is:
  o user: root
  o Password: Oscar
- Make sure that you understand that the SDK is actually a complete operating system installation (Ubuntu) that you run on your computer thanks to a Virtual Machine (thanks to a virtualization program called "Oracle VM VirtualBox"). It allows you to compile your programs and test them on the virtual machine, OR send them to the Linux system installed on the camera and run the program on the camera.
  You could use your current system (windows, apple, ..) to build and transfer the programs to the LeanXcam, but we recommend to use the Virtual Machine as it has already been configured for this task, has the required toolchain and other things installed.
- You can transfer files from Windows to the virtual machine thanks to the “share” folder (careful to use this exact name), that you can find in /media/share on the virtual machine.

Now that the first steps are done, launch the virtual machine. A few warnings will appear, but there's mainly one shortcut to remember:
1. CTRL + F to put the virtual machine in full screen or go back to windowed mode (careful, only the CTRL key on the right side of the keyboard works)

Eclipse is installed on the Virtual Machine (shortcut on top left, with violet circle). If you are familiar with Eclipse, you can use it, but it’s not necessary. You can very well use the text editor gedit as well.

Go visit your home folder at /home/oscar (click “Places”-“Home folder” on top left). You’ll see that a few folders exist already. For example:

2. The web-view application is the one that runs on the camera by default, and of which you see the output when you type http://192.168.1.10 in your browser (if you haven't re-programmed the camera).
3. Oscar is the collection of all the libraries used to run the LeanXCam's basic function. See documentation here: https://github.com/scs/leanXcam/wiki/Oscar-Software-Framework-Manual-Description

Compiling your first project with OpenCV

However, the Oscar library is not up to date, and we need some additional libraries in order to do some nice image processing. We'll be using OpenCV and the LeanXCam OSC (Oscar code collection), and in order to not start coding our project from zero, we’ll also use the 'web-view-cpp' application as a starting point.

Follow these steps in order to put these elements into place and run the web-view-cpp application:

1. Download the three up-to-date libraries from https://github.com/scs/leanXcam/wiki/Downloads -> Frameworks (oscar, osc-cc and opencv), as well as the web-view-cpp app.
2. Save them in '/home/oscar' and unzip them (tar -xzf filename.tgz).
3. Rename the new folders to 'opencv-2.x.x', 'osc-cc', 'oscar' (rename the existing oscar folder into oscar-old) and 'web-view-cpp'.
4. Go in the web-view-cpp folder and run “./configure”. Here, say that you want to use Opencv, osc (with all options), and configure the path (default ones are wrong). Here’s the full configuration (correct values are [in brackets]):

   Is the Oscar Framework already in the folder ./oscar? (y/n) [n]:
   Enter the path to the Oscar Framework. [.../oscar]:
   Enter the IP Address of the target device. [192.168.1.10]:
   Do you want to enable debugging symbols? (y/n) [y]:
   Do you want to enable IO simulation on the target? (y/n) [n]:
   Use g++ compiler instead of gcc? Say no if this is a C only project. (y/n) [y]:
   Are you using OpenCV? (y/n) [y]:
   Enter the path to the OpenCV Library. [.../opencv-2.x.x]:
   Are you using Oscar Code Collection? (y/n) [y]:
   Enter the path to the Oscar Code Collection. [.../osc-cc]:
   Do you want to use the OSC-CC module leanncv? (y/n) [y]:
   Do you want to use the OSC-CC module leanXoverlay? (y/n) [y]:
   Do you want to use the OSC-CC module vchecker? (y/n) [y]:
   Select the board you are using (Mesa-SR4k/leanXradio/leanXcam/indXcam). [leanXcam]:
5. Then, run ‘make opencv’, then ‘make oscar’, then ‘make osc-cc’. This is only necessary once, and it takes quite some time (especially to build opencv).

6. Finally, to build the web-view-cpp application and transfer it to the camera, run ‘make deploy’. You’ll need to have the camera connected with the Ethernet cable, and to give the root password (‘oscar’).

7. You then just need to launch it by running ‘make run’. You can see the results by connecting to 192.168.1.10.

8. In order to have this application run by default when you start the leanxcam, you have to set the environment variable ‘runscript’. First, ssh the camera with ssh 192.168.1.10. Then, type in this command:

   `fw_setenv runscript /mnt/app/web_view_cpp.app/run.sh`

   Note: More about the environment variables and μlinux in general can be found here: https://github.com/scs/leanXcam/wiki/Users-Guide-Using-uClinux

Now we have all necessary libraries installed. The Web viewer application shows us how to grab images from the camera and provides us with a web interface displaying the image and that is easy to configure at our needs.

**Web viewer app**

The important files of the Web Viewer app are the following:

- **Main_class.cpp**: This is where the main loop is run, and the functions of all three other modules are called.
- **Camera.cpp**: This is where the camera configuration is done
- **Ipc.cpp**: This is where the requests from the web interface are treated
- **Image_processing.cpp**: This is where the image processing is done
- **www/index.html**: Template for the web interface

Note: In the original app, there are a few things that are not suited to a robotic application, and it is strongly recommended to download the modified web-view-cpp application from the wiki, in order to correct the following issue:

The Web-view application works in a mode where all actions are triggered by commands sent through the web interface. For example, the image acquisition is triggered by a javascript script on the web interface that requires for a frame every second or so. This is not ideal when the camera is supposed to run without anyone connected through the web interface. Therefore, the modified application constantly fetches images from the camera in a loop, independently from the web interface, and transmits them if the web application asks for them.

The app on the wiki also has the example code useful for the following chapters
Tutorial: Customizing the web interface

1. Study a bit the file www/index.html and compare some element names with what you can find in the ipc.cpp file.

2. Try to add an option (checkbox) to the web interface. Name it ‘activate_processing’. It will be used to activate or deactivate the image processing on the board.

3. Add code to ipc.cpp so that you can store the value of this new option in the structure ‘m_web_settings’.

4. Try transferring this option to the ‘image_processing.cpp’ module, and replace the line `if(false)` by a test on your value.

5. Now, try to modify the code so that the frame rate (computed in the main_class.cpp loop) is transmitted to the web interface

Image processing

1. Study the code in image_processing.cpp. Do you understand each function? If not, use the documentation.

2. What is the use of the first bit of code that extracts the Hue value?

3. Try improve the code in order to detect the biggest red blob. Look at the OpenCV functions: cvDilate(), cvErode(), cvFindContours().

4. Show where the blob is on the image by superposing a rectangle to the original image with cvBoundingRect() and cvRectangle();

UART communication

An example of UART communication is showed in the Image_processing.cpp (The string “Hello!” is constantly sent through UART).

The default baudrate is 9600, and the UART port of the LeanXcam can be found if you unscrew the case of the camera. The Hardware manual (see below, chapter "Documentation) shows where the header can be found on the board. The pin the furthest from the image sensor is the ground pin. The only thing that prevents you to connect the TX and RX pins to an Arduino is that the voltage is between -6 and +6V, whereas UART is generally between 0V to 5V on most microcontroller boards. You can use a simple voltage converter that you can solder on the board made by Alessandro.

1. Try reading the string sent by the LeanXcam through UART with your PC, a serial to USB converter and Alessandro’s converter.

2. Try sending through UART the values of the center of the blob found in the previous chapter as a binary sequence of two floats.

Documentation

This tutorial shows you a few application examples. However, to reach the full potential of the LeanXcam, it will of course be required to read the documentation. Here are a few useful pages that you’ll want to visit to learn more:
• LeanXCam Documentation Wiki:
  https://github.com/scs/leanXcam/wiki/Documentation
• Oscar framework manual:
• Oscar framework documentation:
  In the virtual machine, go to /home/oscar and execute ‘make doc’. The doc can then be found in
  /home/oscar/documentation/index.html
• LeanXCam hardware manual:
• OpenCV documentation:
  http://opencv.willowgarage.com/documentation/
  (warning: the current version of OpenCV is 2.3, but the version installed on the LeanXcam is 2.0,
  also not all OpenCV functions are available)
• LeanCV documentation:
  https://github.com/scs/leanXcam/wiki/LeanCV

In general, you’ll want to search for unknown function names that you find in your code to know where
they’re declared. You can use the following function:

    find . | xargs grep 'string_to_look_for' -s

**Debugging**

On the camera (first ssh to 192.168.1.10 and go to the web_view_cpp.app folder):

    root:/mnt/app/web_view_cpp.app> gdbserver 192.168.1.10:7777 ./app

On the Virtual machine (PC):

    oscar@oscar:~/web-view-cpp$ ddd app_target.gdb -debugger bfin-uclinux-gdb

This launches the debugger program (very old looking program, you’ll have to deal with it if you want to
use a debugger). On the command line of the debugger, run:

    (gdb) target remote 192.168.1.10:7777

    And then click ‘Cont’ to run the program. (yeah, don’t click ‘Run’ that won’t work!).