Getting started with SDK and Gimme2

cbg13002

September 2017

1 Getting started

1. Install the Xilinx Software Development Kit (SDK), it can be installed from the same image as Vivado as a stand alone program. If you dont have Vivado already installed you can find the SDK download at `https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html`. However, only the SDK will be needed for programming the Gimme2 board, Vivado is used for the hardware connections which is already implemented. Choose the version 14.4 for the download to avoid compatibility issues. Open the installer by running the file called "xsetup" in the downloaded "Xilinx_Vivado_SDK.." folder and choose SDK for installation. Remember to run the file with sudo to be able to install the program in the default directory /opt/Xilinx/SDK.

2. Manage a license, the webPACK is a good suggestion to avoid a limitation of time (30 days for Trial and for Evaluation). After filling in all necessary information a license file (Xilinx.lic) will be sent to your registered e-mail address.

3. Go to Vivado License Manager (VLM) and click "Load License" and then on the button "Copy License...". Download the .lic file and choose it with the file explorer that popped up after pressing "Copy License...". Now go to your Home directory and press Ctrl + H to show hidden folders and files and make sure that the Xilinx.lic exists in the folder ".Xilinx".

4. In the same directory (your Home directory), open .bashrc with gedit, nano, vim or another text editor with sudo, for example: *sudo vim .bashrc*, and add the following to the end of the file:

   source /opt/Xilinx/SDK/2014.4/.settings64-Software_Development_Kit.sh
   and
   export XILINXD_LICENSE_FILE=/home/⟨user name⟩/.Xilinx/Xilinx.lic

5. Start up the SDK by opening a terminal and navigate to /opt/Xilinx/SDK/2014.4/bin and run *./xsdk*. Import the project "tmp" that is located in gimme2/sdk/sdk_2014.4/tmp. The tmp project contains all necessary files for compiling the code for the Gimme2. To be able to build it two settings have to be made. Right click in the Project Explorer (on the left side in the SDK) on 'tmp' and choose 'properties'. Choose the tab C/C++Build and then the 'Directories' tab, click the icon with a green plus symbol and include the path /gimme2/sdk/sdk_2014.2/src.
   Right click in the Project Explorer (on the left side in the SDK) on 'tmp' and choose the 'Paths and Symbols' under 'C/C++ General'. Go to 'Source Location' and press 'Link Folder', name it for instance src2 and check the box 'Link to folder in the filesystem' and enter the path to /gimme2/sdk/sdk_2014.2/src. Now you should be able to build the project.

2 General information

The main file for programming is called view_dump_thread.c and tmp.elf is the outputted compiled file when the project is built. The .elf file should be created in your workspace but it has also appeared to be outputted in the tmp folder from the imported destination.
To connect to Gimme2 you can either connect your computer via ethernet cable directly to the board or via the Naiad router (which is connected via ethernet to Gimme2). If you want to visualize the cameras you should choose the computer to board method, the reason will be explained later. Either way, type *sudo ssh 192.168.1.10* and then type in your password and then the password 'root' for Gimme2. If you chose the computer to board method you probably need to turn off your wifi connection. If the connection cant be established, try to add a static IP address by adding a few lines to the file /etc/network/interfaces. Do *sudo vim interfaces* and add if not already present:

iface ⟨connection name ⟩inet static
    address 192.168.1.xx
    netmask 255.255.255.0

and then comment out the line 'iface ⟨connection name ⟩inet dhcp'. xx indicates an arbitrary number that is not already used in the network. Type *ifconfig* to check your connections.
Copy the .elf file from a new terminal (that is not connected to Gimme2) to the microSD card in the gimme2 by typing *sudo scp path/to/tmp.elf root@192.168.1.10:/media/card*. Then execute it by navigating to /media/card and run *./tmp.elf* to run the program on the gimme2 board. Now the programming is running and you can execute commands like toggling camera, applying harris corner or harris edge detection. This will be described more in section 3. To visualize the cameras you need to get a video stream from the cameras. If you are in Windows you can simply run the program CVSharp located in gimme2/CVSharp/CVSharp/bin/Debug/CVSharp.exe. To make it work in Ubuntu (16.04) a version of Wine will fix the compatibility issues. Download Wine, this can be done graphically from Ubuntu Software if preferred, and run *winefile* in a new terminal. A file explorer will be prompted, navigate to gimme2/CVSharp/CVSharp/bin/Debug/CVSharp.exe and then double click the CVSharp.exe file. Now a small window will appear showing the camera stream. This stream is only for visualization and is run on your computer. The stream consists of UDP packets sent from the Gimme2 and to get the visualization the ethernet cable should be used. It is possible to establish a connection via the Naiad network with CVSharp as well but the UDP stream will in this case flood the network, making the connection extremely slow for all users connected to the Naiad network. The CVSharp window will have problems to stay updated since most of the UDP packets wont reach to the computer. The UDP stream can however easily be turned off when using the wifi by pressing the 'p' button. More about this in the next section.

The images are decoded in the FPGA and the data is stored in a memory that is shared between the FPGA and the CPU. The CPU reads the start address of the image data from the shared memory, creates a file on the microSD card and then transfers the data.
There exists a manual for setting up the Gimme2 called Gimme2Manual.pdf. This manual is rich on information but it should also be noted that the setup described in the manual is done and more beyond the manual is implemented.
The OS is Ubuntu 16.04 and it is a Xilinx compressed version installed in the flash memory. The flash memory has a very limited size which makes it more complicated to install applications like ROS on the board since they have to be built on the microSD card.

3 Controlling the Gimme2

When the program is running there are options to use a number of commands. Use "sudo ssh 192.168.1.10", root:root, on host computer. Then if the Naiad is used, navigate to /media/card/lab_test and run ./copy_so to add a library file to the lib folder. Then run tmp.elf from the same folder as copy_so. If the "leaked" Gimme2 is used, navigate to /media/card and simply rum tmp.elf. The following keyboard shortcuts can be used to control the stereo camera:
q - quit
w - take a photo (writes an image pair)
o - take continuous photos (starts/stops writing images)
t - toggle camera (left/right for display)
a,s,d,f - increase exposure time (1000,100,10,1 time units)
z,x,c,v - decrease exposure time (1000,100,10,1 time units)
e - toggle automatic exposure (default on)
g - increase gain
b - decrease gain
h - toggle Harris
r - toggle Stereo (presidence over h)
j - increase Harris corner threshold
m - decrease Harris corner threshold
k - increase Harris edge threshold
, - decrease Harris edge threshold
p - toggle image live stream (default off. Running CVSharp as explained in section 2 is required for visualization)

4 Point cloud Generation

Run sudo ssh -X 192.168.1.117 to connect to Odroid (-X is for redirecting the graphic from odroid to host computer)
Navigate to catkin_ws
Run ./RunPointCloud⟨X⟩,where ⟨X⟩can be "Bmp", "BmpData", "Jpg" or "JpgData"
Choose the corresponding function (for example SaveBmpImage()) in Xilinx SDK, compile and transfer tmp.elf to the sd card
Open a new terminal on host computer (your computer)
Run:
export ROS_MASTER_URI=http://192.168.1.117:11311 (the IP might be dynamic, in that case change 117 to 217)
export ROS_IP=⟨your IP address⟩
rviz rviz

If you get a warning that says:
"ROS_HOSTNAME / ROS_IP is set to only allow local connections, so a requested connection to '192.168.1.117' is being rejected." then replace export ROS_IP= ⟨your IP address⟩with 'export ROS_HOSTNAME =⟨your hostname⟩'.
Write 'hostname' in a terminal to get your hostname.
If you havent added "source ./devel/setup.bash" to bashrc you have to type 'source ./devel/setup.bash' in the terminal as well.

Now you should see a point cloud in rviz. if you have no topic to listen on, click 'Add', then 'By topic' and then find the topic that contains a Point cloud. If it still does not show the point cloud, try change

'Fixed Frame' under 'Global Options' from 'map' to 'naiad' or 'base_link'

There are two lines in each script on the Odroid (/root/catkin_ws) that copies all recieved files to /root/catkin_ws/src/SWARMs/tcp_socket_communication/calib_imgs. This can be a good help when debugging or taking calibration images.