Vision system - Hardware*

Rizwin Shooja
rsa12004@student.mdh.se

Aseem Rastogi
ari12003@student.mdh.se

ABSTRACT
This report elucidates the hardware used in the vision system of the AUV Naiad and the firmware associated with it. The vision system facilitates detecting objects of various shapes and colours using its high resolution image sensors. The operating system used in the vision system hardware is Linux and the system interacts with other devices in the robot through a CAN bus. The challenges faced by the team in achieving the requirements, like coding in Ada and using state of the art products in embedded systems, are also mentioned in the report.

1. INTRODUCTION
The primary objective of the vision system is to detect target objects of various shapes and colours and communicate to the other components within the robot to take necessary actions. The system comprises of two GIMME-2 stereovision boards [1] equipped with a hard processor [2] called Zynq 7020 from Xilinx and two 10 Megapixel OV10810 [7] image sensors. The Zynq processor consists of an ARM Cortex A9 dual-core processor and a FPGA in the same silicon chip. The FPGA architecture augments the image processing capabilities of the system as it can do arithmetic operations in parallel, unlike the CPUs which do it sequentially. The board has 8 GB of RAM for the Processing System (ARM processor) and 4 GB of RAM for Programmable Logic (FPGA). There are two fast Ethernet ports and one gigabit Ethernet port for high speed communication with other devices. There is a slot for external memory storage where the image processing binary files are stored. The operating system installed on the GIMME-2 board is Linux (version 3.6.0-xilinx).

*This report was written during the fall of 2013 in an advanced level project course at Mälardalen University, Sweden.

2. IMPLEMENTATION
At first, the GIMME-2 board receives a CAN message from the mission controller depending on the mission. The message IDs for the GIMME-2 board with cameras facing forward and GIMME-2 board with cameras facing downward are different as both are intended for different purposes depending on the mission. When a mission is set, the cameras search for the desired target. Based on the image processed data, corresponding CAN messages are sent back to the Mission Control System when the mission objectives are completed. The images that are captured by the cameras on-board are transferred to the memory using a high speed Low Voltage Differential Signalling (LVDS) bus. The desired action is then determined and transmitted over to the CAN bus.

2.1 Operating system
The operating system used on the GIMME-2 board is Linux customized by Xilinx for Zynq processors which is based on the vanilla kernels from versions 3.0 and above. The user can interact with the Linux on the board either through serial interface or through SSH client. The team developed a boot image considering that the boot medium is the flash memory of the board. Booting from SD card is not working as there is a PCB flaw for pin mapping on the GIMME-2 board. The driver for Platform USB Cable II has to be installed in order to get the JTAG interface working for flashing the device.

Creating a bootable Linux for Zynq is quite challenging as Linux boots up only when a complex combination of criteria is met. The files necessary for booting Linux have to be built from scratch as the files available on the Xilinx website are mostly for the Zynq evaluation board. The procedure and tools used for building the necessary files are described in detail in Xilinx wiki [5].

The First Stage Boot Loader (FSBL) is the first file loaded on power up which does the early system initialization. FSBL is created in Xilinx SDK using the system hardware description of the platform. Then the Universal bootloader starts which loads the Linux kernel image and the wrapped ramdisk image.

2.2 CAN driver
Xilinx has released the drivers for many peripherals in the mainline kernel [6]. However, the driver for the CAN module is not available in the mainline kernel and hence it has to be built from source. A generic CAN driver hosted on Linux

known as can4linux [3] is supported by the Zynq Linux. The driver code is customized so as to get it working for Zynq and compiled for the target platform. On power up, the boot image is copied from the QSPI flash memory into the RAM and hence whatever changes made on the kernel will not be saved back into the flash memory. Hence the RAM disk image has to be edited by copying the driver module at an appropriate location and the start-up script is modified accordingly so that the driver gets loaded when the Linux boots up. This RAM disk image is used to create the boot file and hence it loads along with other drivers.

The system command *devmem* is used in order to access the registers of the Zynq processor through Linux. The clock for the CAN module is configured using the devmem command and added to the startup script so that the clock is set to the desired value when the board boots up. Other parameters such as baud rate and mask are set in the program code. Another challenge was to write the code for testing in Ada when the driver code is written in C language. Few functions have to be imported from C into Ada in order to meet the requirement.

The cross compiler for Ada from Adacore is not available free of cost. So the team went for an alternative which is compiling the sources statically on the Pandaboard [**?**] or BeagleBone Black and copying the binaries on to the GIMME-2 for execution. Since the other two boards also have ARM Cortex A-series processor and run Ubuntu as the operating system, the binaries generated will be supported in the same architecture.

3. RESULT

When the GIMME-2 board is powered with 12V, the device boots up and the booting sequence can be observed through the serial terminal. The team used *gtkterm* in a Linux host for interacting with the device.

The image processing files stored on the memory card are tested successfully on the GIMME-2 board. Core affinity to the image processing tasks are assigned and the functionalities are executed in parallel by exploiting the dual-core processor.

The CAN module is also tested by using the Kvaser CAN interface cable connected to a PC. The baud rate is set and messages are sent and received successfully over the CAN bus. Message filtering is also tested successfully by screening a set of message IDs.

4. CONCLUSION

The vision system is tested successfully in terms of booting an operating system from flash memory and executing the image processing files in it. The system can act accordingly based on the mission by receiving CAN messages and reply back when the mission is completed successfully.

4.1 Future work

One of the Ethernet ports is not mounted which can be fixed later on. Then the GIMME-2 board can act as a hub in receiving all the messages from the Mission Control System and transmit to the ground station about the current status of the AUV.

When the PCB flaw of the SD card is fixed, proper operating systems like Ubuntu can be installed on the GIMME-2 board [9] which then doesn't have to rely on the pandaboard for compiling its source files. Even OpenCV can be installed on the GIMME-2 as Xilinx has already released the support for Ubuntu on the Zynq platform.

5. APPENDIX

5.1 Getting started with GIMME-2 board

This section describes the step by step procedure to get started with the GIMME-2 board. The operating system used on the host machine is Ubuntu 12.04. The prerequisites for initializing the GIMME-2 board are:

1. Micro-USB Male (Type B) to USB Male (Type A) connector or an Ethernet cable.
2. Power supply. The board should be supplied with voltages in the range +12V to +24V.
3. If the user is planning to use the USB cable, then a serial terminal, for example *gtkterm*, should be installed on the host PC to interact with the Linux on the GIMME-2 board.

Assuming that Zynq Linux is already installed on the GIMME-2 board, the user can interact with it in either of the two ways described below. Ensure that the jumper configuration is 0100 before powering on the board.

5.1.1 Using serial interface

Connect the micro-USB to USB connector to the port labelled *USB0* on the board and connect the other end to the host PC. Set the voltage of the power supply to +12V and connect it to the GIMME-2 board. Now open the serial terminal and the booting steps can be seen there. Once the booting is over, the user gets a `zynq>` prompt on the screen.

5.1.2 Using Ethernet

Connect one end of the Ethernet cable to the middle port among the three on the GIMME-2 board and the other end to the host machine. Now open the bash terminal and execute `ssh root@192.168.1.10`. Type in *root* when password is prompted. The user gets a `zynq>` prompt on the screen.

5.2 Creation of Zynq Boot Image

The procedure for creating a boot image is quite complex and time consuming. For the sake of the new users, all the files required to create a boot image are available on the Subversion repository for project Naiad [8]. Additional to the prerequisites mentioned in section 5.1, the user needs to do the following:

1. Install the full package of Xilinx ISE Design Suite[10] including the Software Development Kit (SDK) on the host PC.
2. Install the driver for Xilinx Platform USB Cable II [4]. If Step 1 is done in Windows OS, then no need to install the driver as it comes with the package.

Using SDK, new boot image can be created by giving offsets to the binary files as mentioned in the user guide [8]. If amendments have to be made to the Linux kernel like installing a new Linux driver, then the RAM disk image should be modified and a new boot image should be created with the new RAM image. All the steps for modifying RAM image are also detailed in the user guide [**?**].

6. REFERENCES

[1] af inventions. Gimme2 user guide, June 2013.

[2] B. H. Fletcher. Fpga embedded processors - revealing true system performance. In *Embedded Systems Conference*. Memec, 2005.

[3] can4linux driver. http://www.can-wiki.info/can4linux/man/index.html.

[4] Driver installation instructions for platform usb cable ii. http://rmdir.de/ michael/xilinx/.

[5] Getting started guide for xilinx zynq. http://www.wiki.xilinx.com/Getting+Started.

[6] Linux drivers for xilinx devices. http://www.wiki.xilinx.com/Linux+Drivers.

[7] Ov1810 product desciption. http://www.ovt.com/products/sensor.php?id=101.

[8] Setup guide and boot files for gimme-2 board. https://v-lagerlunda.ita.mdh.se:8443/svn/cdt508/projectht13/hardware/GIMME-2.

[9] Ubuntu on zynq platforms. http://www.wiki.xilinx.com/Ubuntu+on+Zynq.

[10] Xilinx ise design suite - full installer for linux. http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html.