National University of Science and Technology

Think in Other Terms

DEPARTMENT OF ELECTRONIC ENGINEERING

Master of Philosophy Project Report (2006 – 2008)

STUDENT:	**Blessing Mahomva**
TITLE OF PROJECT:	**Multisensor Educational Robot as a System on Chip Research Platform, (General Purpose System-on-Chip Hardware / Software Co-design Environment).**
LABORATORIES:	**Interdisciplinary Institute of Integrated Systems (I3S), Microlab. Bern University of Applied Sciences Berner Fachhochschule Switzerland.**
SUPERVISORS:	**Prof. M. Collier Prof. M. Jacomet**

Bern University Of Applied Sciences, Switzerland

ABSTRACT

The technology in electronic engineering is changing at a fast rate. Designers want to design faster and more energy efficient systems that will cope with the industrial and consumer market demands. Another challenge is to design smaller but very powerful products that are portable, e.g. mobile phones, multimedia gadgets e.t.c.

The design of Educational Robot modules to be used as a System-on-Chip research platform for researchers and students is undertaken as an MPhil project. The project is designed in such a way as to give the researcher an open platform on which he/she is flexible to implement more than one solution to any given application so that valid conclusions on a more efficient method are drawn. This Educational Robot platform is codenamed the GECKO3 by the Microlab at the Bern University of Applied Sciences, Switzerland where this MPhil project was undertaken. The modules designed are the GECKO3Basic, GECKO3Power and the GECKO3Sensor boards that are all discussed in detail in this report.

If adopted, this research platform will help to teach this subject in a very interesting way in electronic engineering courses at college and university level.

ACKNOWLEDGEMENTS

Many people contributed to the success of this MPhil project. First I would like to appreciate the support I got from my girlfriend “Bonnie” who so understood while I spent 14 months away from her in Switzerland, undertaking this MPhil project. I also thank my mother for sacrificing so much in her life for me to get all the education I have acquired.
I would like to thank my supervisors, Mike Collier who facilitated this scholarship and Marcel Jacomet whose guidance, support and close supervision during this project was priceless.
I also appreciate all members of New International Church in Switzerland for making me feel at home while I was in a foreign land, they showed we how big the Church of God is by their love and support. Last and not least, are the members of the Microlab team at the Bern University of Applied Sciences who were so accommodating and helpful.

The satisfaction that a man will ever have in this life is fulfilling his God given purpose, for man is created for God’s glory!

CONTENTS

1. INTRODUCTION

For education and research projects, a new state-of-the-art general purpose hardware/software co-design platform (GECKO3) is under development by the Microlab team.

This platform is a highly integrated environment, miniaturised to the size of a credit card. Two objectives are followed in this platform; 1st, the possibility to attach numerous sensors and actuators serves educational purposes, 2nd, the very high number of configurable hardware gates is basis for future research projects in the area of hardware algorithms.

The System-on-chip (SoC) subject has been in the industry for some time now with a number of electronic products now based on this technology. Some universities have made it part of their curriculum while it's still a new subject to a number of universities in developing countries.

In this MPhil project, work was focused on the development of the GECKO3 power board (GECKO3Power) that houses some dc-dc converters, battery charger circuits, dc supply and battery switch-over circuits, H-bridges for driving motors, ultrasonic rangers and infrared detectors, and the GECKO3 sensor board (GECKO3Sensor) that houses a MEM acceleration sensor, electronic compass module, Bluetooth radio and a CMOS camera. The developed platform will be used for both educational and research purposes at the Bern University of Applied Sciences in Switzerland and the National University of science and technology in Zimbabwe.

2. SYSTEM-ON-CHIP (SoC)

It is important to recognize that integrating more and more functionality on a chip has always existed as a trend by virtue of Moore's Law, which predicts that the numbers of transistors on a chip will double every 18-24 months [1]. Integrated circuits designers are continuously taking up the challenge to keep pace with Moore's Law. This then means that more and more room is being created for the SoC design engineer to fully exploit the resources that are being offered by FPGA designers.

Today many industrial and commercial products contain SoC solutions.

System-on-Chip can be best explained by comparing it with System-in-Package (SiP) and System-on-Board (SoB) as shown in figure 1 below.

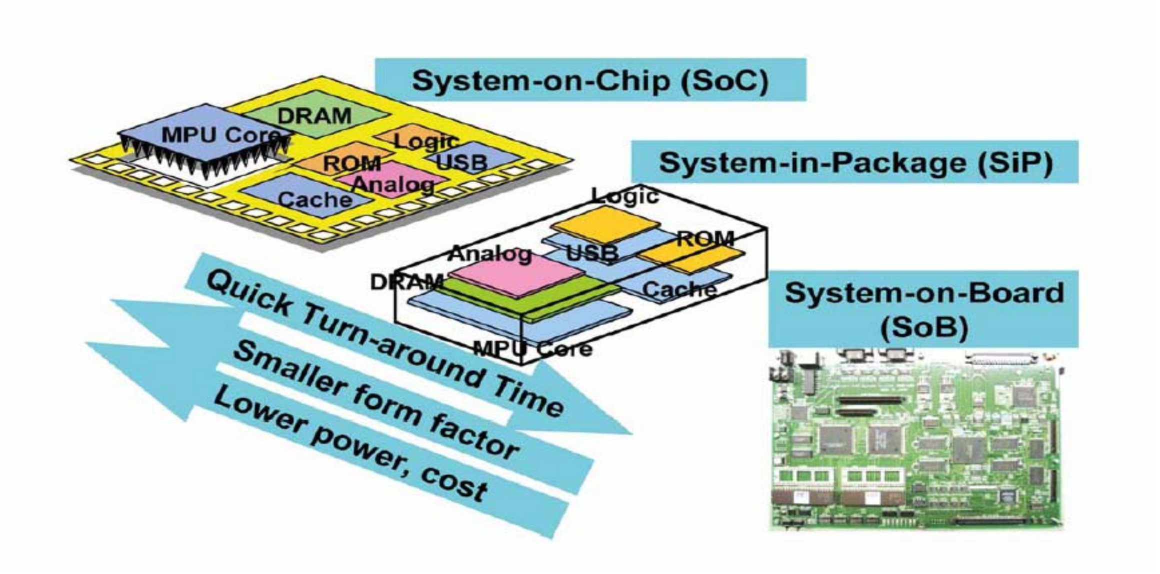


Figure 1, SoC versus SiP versus SoB.

In SiP, separate chips are packaged together into a system with a very small form factor using a common two-dimensional (2-D) or three-dimensional (3-D) substrate while in SoB separate chips are placed on a board and linked to each other by strips of copper.

In SoC, Intellectual property (IP) blocks are integrated on a single chip.
IP blocks are the highest level building blocks of a SoC. A library of reusable IP blocks with various timing, area, power configurations is the key to SoC success as it allows mix-and-match of different IP blocks so that the SoC integrator can apply the tradeoffs that best suit the needs of the target application.
IP cores can be categorized as either *soft*, *firm* and *hard* [1].

- Soft IP blocks are specified using RTL or higher level descriptions. There are more suitable for digital cores, since a hardware description language (HDL) is process-independent and can be synthesized to the gate level. This form has the advantage of flexibility, portability, and reusability, with the drawback of not having guaranteed timing or power characteristics.
- Hard IP blocks have fixed layouts and are highly optimized for a given application in a specific process. They have the advantage of having predictable performance and disadvantages of cost and limited portability that may greatly limit the areas of application.
- Firm IP blocks are provided as parameterized circuit descriptions so that designers can optimize cores for their specific design needs. Firm IP offers a compromise between soft and hard, being more flexible and portable than hard IP, yet more predictable than soft IP.

The relationships and tradeoffs of the three IP blocks are depicted in figure 2 below.

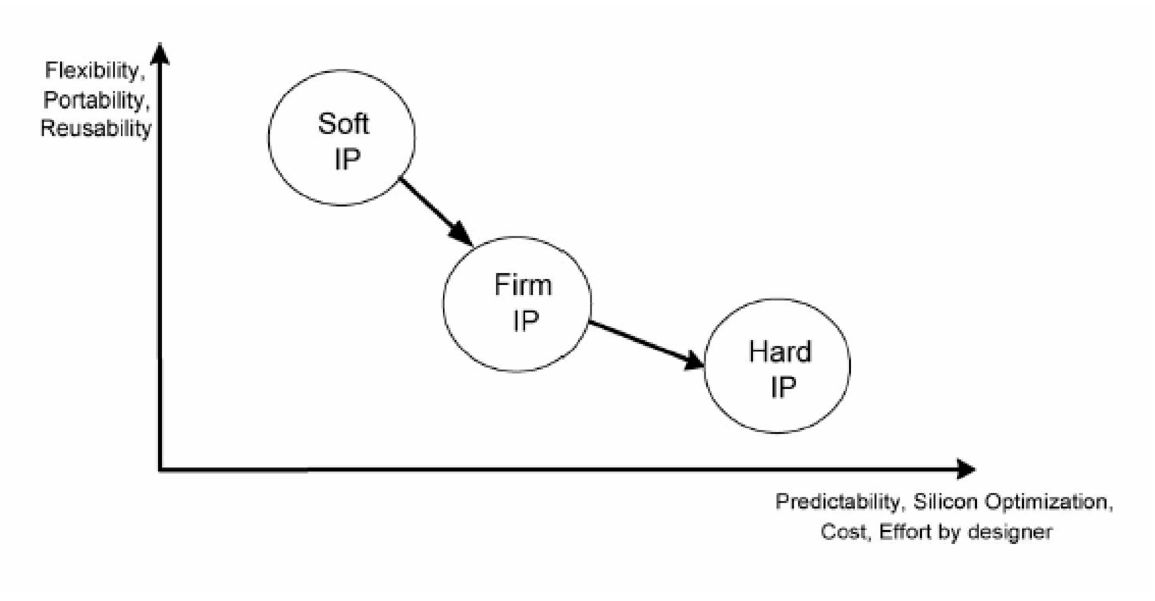


Figure 2, Different types of IP blocks.

These IP blocks, e.g. CPU, DSP, Memory, DMA, and I/O etc…, are interconnected and communicate with each other by the use of standard SoC busses [1,2,4].

There are four main bus architectures, AMBA, AVALON, CORECONNECT and WISHBONE.

AMBA

- Advanced High-performance Bus (AHB)
- Advanced system bus (ASB)
- Advanced peripheral bus (APB)

AVALON

CORECONNECT

- Processor local bus (PLB)
- On-chip peripheral bus (OPB)
- Device control register bus (DCR)

WISHBONE

Figure 3 below shows an example of a platform implementing the AMBA bus architecture.

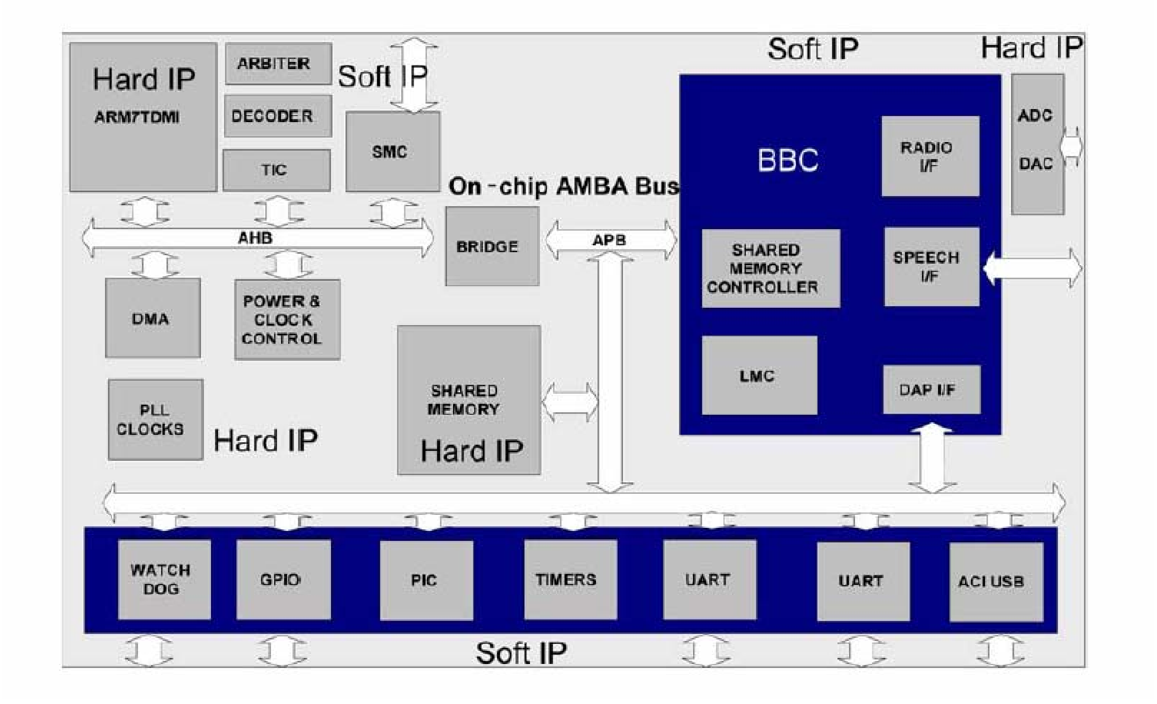


Figure 3, Example showing IP cores connected by the AMBA bus.

3. GECKO3 SYSTEM

3.1 GECKO3 System

The GECKO3 system is a general purpose hardware/software co-design environment for real-time information processing for system-on-chip (SoC) solutions. The GECKO3 system supports a new design methodology for SoC, which necessitates co-design of software, fast hardware and dedicated real-time signal processing hardware. Within the GECKO3 system, the GECKO3Main board represents an experimental platform, which offers the necessary computing power for speed intensive real-time algorithms as well as the necessary flexibility for control intensive software tasks. The GECKO3Main board can be used for research and industrial projects. The complete system including additional boards with sensors and actuators, GECKO3Sensor and GECKO3Power discussed in this report is used for both educational and research purposes.

The purpose of the GECKO3Power board is to supply power to the whole platform as well as house some basic sensors and motor driver circuits. The GECKO3Sensor contains a number of sensors, this board together with the GECKO3Main board gives students and researchers the challenge of developing their own hardware and software algorithms to manipulate the raw outputs form these sensors or use the IP (Intellectual Property) cores from Xilinx or other third party IP core developers. The choices implemented will be governed by speed, power consumption and precision requirements of the target application. The integrated optional CMOS camera and OLED display will offer the task of developing more computational intensive image processing hardware and software algorithms.

Besides these two boards discussed in this project, the other boards and the mechanical system are being designed and developed by the Microlab team as separate projects.

3.2 Project specifications

The objective of this MPhil is to design and develop the GECKO3Power and the GECKO3Sensor board. These two boards have the following specifications.

GECKO3Power specifications:

- 2 x 3.3V (3A) DC-DC step-down converters.
- 1 x 5V (80mA) DC-DC step-up converters.
- 1 x DC and USB Lithium Ion battery charger circuit.
- 1 x Battery-DC switch-over circuit.
- 2 x H-Bridge DC motor control circuits.
- 2 x Infrared sensor and dictator pairs utilizing two Sigma-Delta ADC circuits for the conversion of the analog signals to digital values to be interpreted by the FPGA.
- 1 x Ultrasonic sensor circuits using the I2C communication protocol for communicating with the FPGA.

GECKO3Sensor specifications:

- 1 x 2-dimensional electronic compass module interfaced to the FPGA by an SPI ADC using the Xilinx SPI IP Core.

- 1 x 2-dimensional acceleration sensor module interfaced to the FPGA by a Xilinx Delta-Sigma ADC IP Core.
- 1 x Bluetooth radio module interfaced to the FPGA by a Xilinx UART IP Core
- 1 x Optional Add-On CMOS 16/8bit digital output camera whose implementation is open, meaning any researcher or student can design their Hardware and/or Software to use this module.
- 1 x Optional Add-On OLED display module that researchers and students can use to display their data and/or images from the camera module.

Prototype experiments will be done using the low cost Xilinx Spartan-3E Starter Board that houses a 500 000-gate Xilinx Spartan-3E XC3S500E FPGA as the SoC platform [10]. From these experiments, tutorials that can be used for teaching this subject using an educational robot as a system-on-chip research platform can be designed. The Xilinx EDK (Embedded Development Kit) will be used to develop the SoC Hardware/Software Co-design solutions [8].

On the complete robot, a main board containing a very large 2 million gate FPGA, housing for example, soft-core 32 bit RISC processors, high speed custom designed hard ware, large dynamic and flash memory banks, and USB 2.0 and LAN interfaces will be used as the SoC platform. This board is being developed as a different project by the Microlab team.

In this MPhil project, prototypes of the GECKO3Power and GECKO3Sensor are designed, developed and tested. The SoC solutions for some of the parts will be demonstrated. It should be noted that the methods used are not ultimate; since it’s a research platform; the system is designed to be open so as to allow flexibility in the development of solutions either in software, hardware or both.

3.3 Validity of the project

This subject has been made part of the electronic engineering core syllabus in many universities around the world due to its emerging industrial importance [3,9].
It is important for universities in developing countries to seriously consider this subject since it has a vast number of advantages; some of these advantages have been highlighted below.

- Integrating hardware such as memory blocks, embedded processors, Digital Signal Processors e.t.c. on a single chip means an overall reduction in size of the final project. This helps the industrial electronics engineer to provide products that take very little panel space and consumer electronic engineers will give their customers gadgets that are small and comfortable to handle.
- The cost of overall products is reduced due to the use of cheap or open reusable IP cores.
- Power consumption has also been greatly reduced in the FPGAs that form the SoC platforms, an advantage to battery operated products, e.g. cellphones.
- The Mean-Time-Before-Failure of the products is considerably reduced due to the reduction of the overall number of hardware components.

4. GECKO3Power

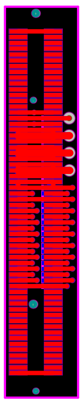
This board is the one that supplies power to the whole GECKO3 platform. Great care was taken in designing this board because the power consumption of the entire platform had to be taken into consideration to ensure that every expansion board gets sufficient power. The board can source its power either from a rechargeable 3.7V Lithium Ion battery or from a 5V dc adapter.

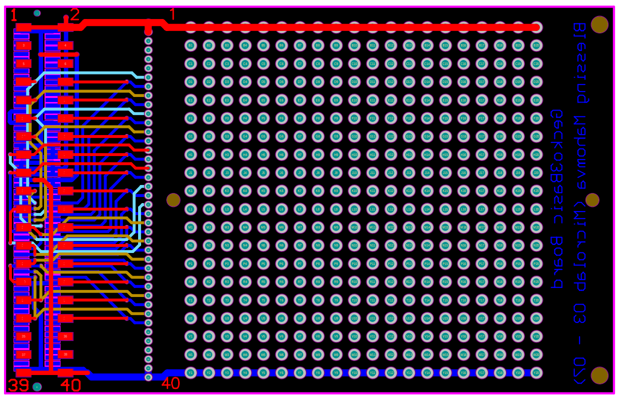
Besides the power supply circuits, this board also contains two H-bridge motor drivers for driving two small dc motors on the mechanical housing. Also included are the ultrasonic rangers for measuring distance, detecting objects and map generation with clever programming. Some infrared detectors are also included at the bottom of this board to be used as line or surface detectors.

To facilitate simple experiments and tutorials on the GECKO3Power board, a simple board called the GECKO3Basic board was also developed as a sub unit of the power board.

4.1 GECKO3Basic

This board along with the GECKO3Power was designed to allow some basic experiments that include the control of motors, ultrasonic sensors and infrared line detectors to be done. As shown below, the GECKO3Basic board is mainly a prototyping area that facilitates a link between the sensors and motor control circuits on the GECKO3Power with the control logic circuits such as CPLDs and DSP emulation equipment for control engineering experiments.


A



B

Figure 4, PCB files images of the GECKO3Basic boards

Shown above are the two boards that constitute the GECKO3Basic. Board **A** is a connector that converts the 60Pin connector on the GECKO3Power to a 120Pin connector that connects to board **B** or the GECKO3Main board. A bread board can also be mounted onto this board using the two holes shown along the horizontal axis.

The following image shows the two boards stacked together while tests on the GECKO3Power prototype were being conducted.

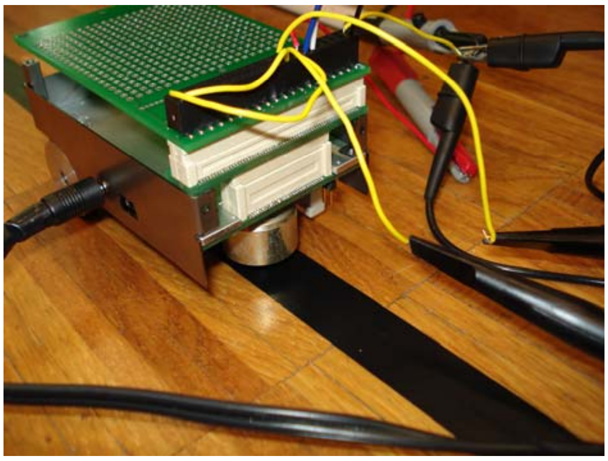


Figure 5, GECKO3Power and GECKO3Basic boards stacked together.

4.2 GECKO3Power

This board is the one that provides the power of the complete platform hence it was designed in a way that allows it to sustain all the power requirements that the main and expansion boards will require together with the motors drive system.
The power sources are a 3.7V (3900mA/h) Li-Ion battery or a 5V (4A) wall adapter that is also used to charge the battery, the battery can also be charged though USB.

This board is divided into three sections, the power, motor driver and sensor section. These different sections are discussed in detail in the following passages.

4.2.1 Power Section

The power section consists of the dc wall adapter and battery switch-over circuit, the battery charger circuit and dc-dc converters. The dc wall adapter and battery switch-over circuit output is connected to the backbone connector that all expansion boards are able to connect to; this allows different boards with different power supplies to do their own voltage conversions form this supply.

DC wall adapter and battery switch-over circuit.

This circuit facilitates the automatic switch-over between the wall adapter and the battery. This circuit was built using the LTC4412 Low Loss PowerPath Controller from Linear Technology and the IRF7702 Ultra Low On-Resistance P-Channel Power MOSFET from International Rectifier.
The LTC4412 controls an external P-channel MOSFET to create a near ideal diode function

for power switchover or load sharing. When conducting, the voltage drop across the MOSFET is typically 20mV. For applications with a wall adapter or other auxiliary power source, the load is automatically disconnected from the battery when the auxiliary source is connected. Figure 6 below shows the circuit diagram of the switchover circuit. DCin is the input from the wall adapter; VBat is the battery input while Vout is the output to all the converters.

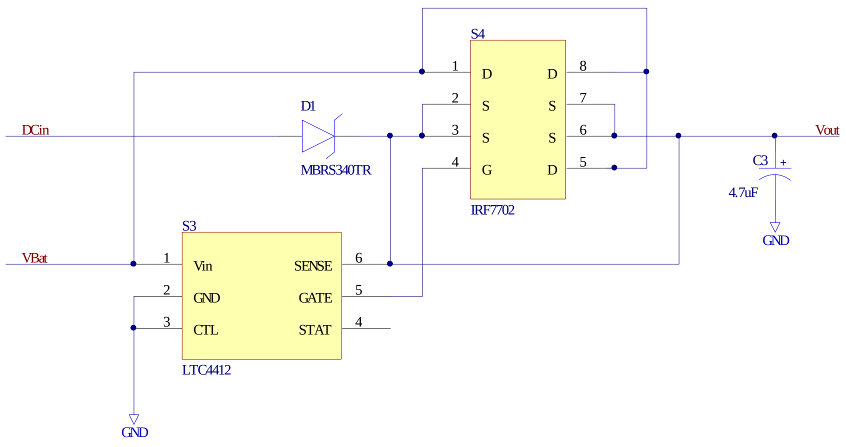


Figure 6, DC wall adapter and Battery switch-over circuit.

Battery Charger Circuit

This circuit is built using the LTC4075 Dual Input USB/Wall Adapter Standalone Li-Ion Battery Charger IC.

The LTC4075 are standalone linear chargers that are capable of charging a single-cell Li-Ion battery from both wall adapter and USB inputs. The charger can detect power at the inputs and automatically select the appropriate power source for charging. Figure 7 below shows the battery charger circuit.

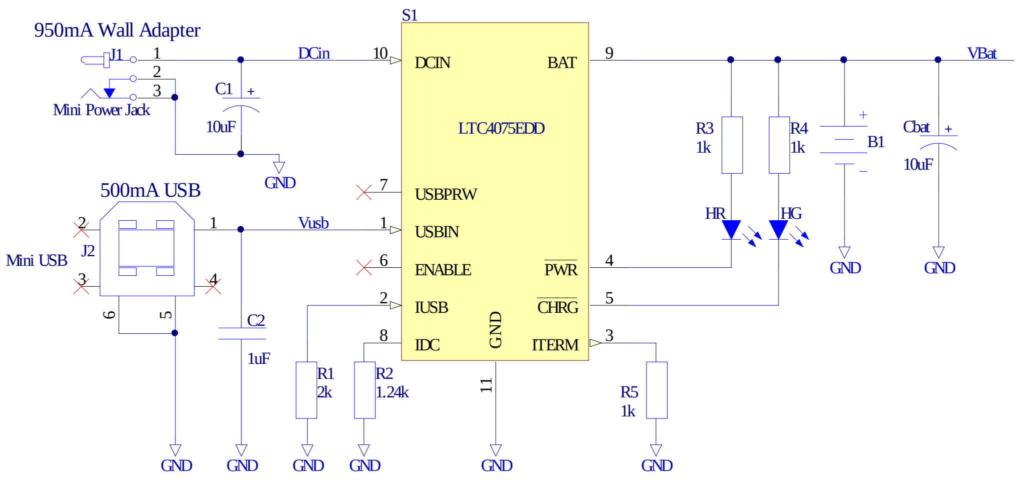


Figure 7, Dual input USB/Wall adapter standalone Li-Ion battery charger.

DC-DC Converters

This section consists of two dc-dc voltage converters built using LTC3412A, 3A Step-Down Regulator ICs from Linear Technology. It operates from an input voltage range of 2.25V to 5.5V and provides a regulated output voltage from 0.8V to 5V while delivering up to 3A of output current.

One of these converters provides power to the GECKO3Power board circuits that include the sensor circuits and motor driver circuits while the other one provides 3.3V power to all the digital circuits on the expansion boards.

A choice of these converters compared to linear voltages regulators was influenced by their ability to deliver high currents with very little heat generation on a small footprint.

Figure 8 below shows the circuit diagram of these dc-dc voltage converters.

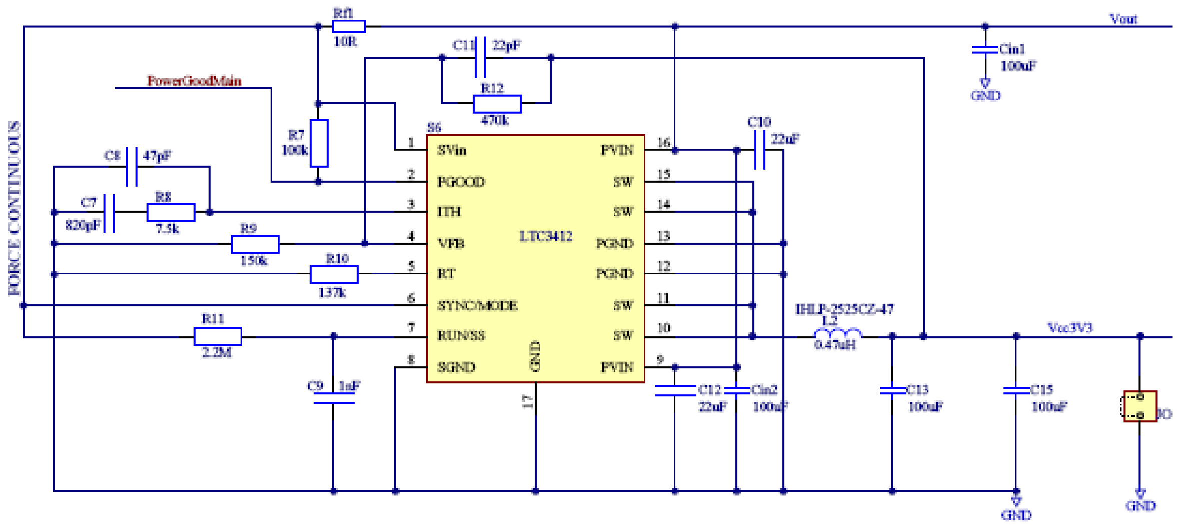


Figure 8

Loading tests on this prototype converter were carried out and the following results were obtained, figure 9 below.

As seen from the graph below, this circuit is capable of supplying 3A of current at 3.3V with a voltage drop of just 0.1V.

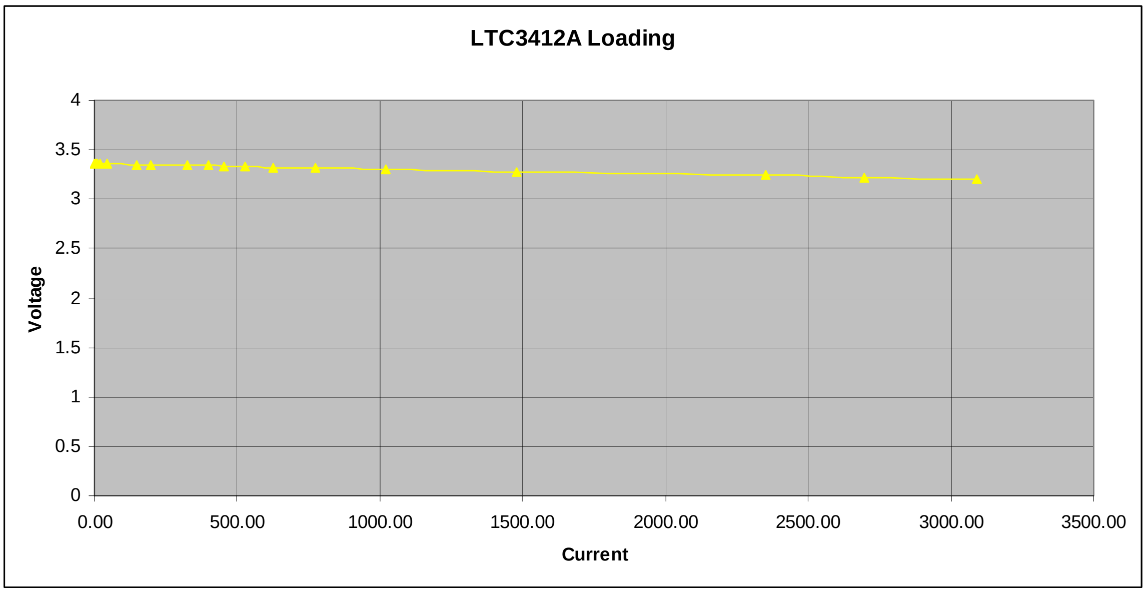


Figure 9, LTC3412A loading tests.

4.2.2 Motor Driver Section

The robot is powered by two Maxon (252802), 3.6V, 0.75W motors with two channel 16 bit pulse encoders. These motors are driven by two H-bridge motor driver circuits built using Vishay Si9986 buffered H-bridge driver ICs. Figure 10 below shows the circuit diagram of these circuits.

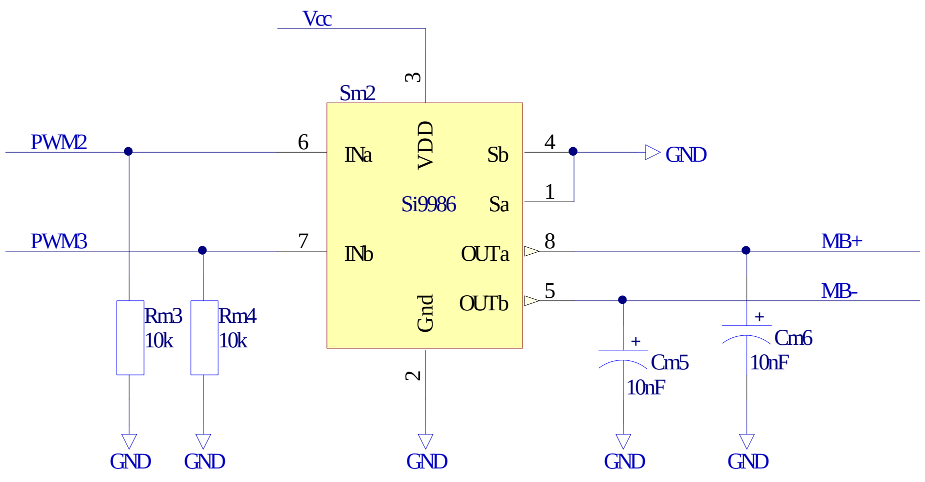


Figure 10, H-bridge motor drivers.

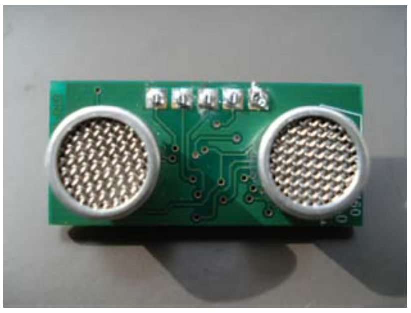
4.2.3 Sensor Section

Two sensors were also incorporated onto this board, the ultrasonic range finders and the infrared line detectors. These sensor systems were designed built and tested; their circuits and test results are discussed in detail in the following passages.

Ultrasonic rangers

This sensor module can be used to detect objects and their distance from the robot and with some complex algorithms it can also be used to do sonar mapping of the environment. The ultrasonic ranger circuits were designed using the SRF10 ultrasonic modules. The modules are powered by a 5V supply and communicate with the FPGA through i2c communication protocol. Because of its 5V power supply, a dc-dc step-up converter was built together with 5V-3.3V voltage translators to enable its interface to the 3.3V digital system.

Figure 11 below shows the images of this ultrasonic module.



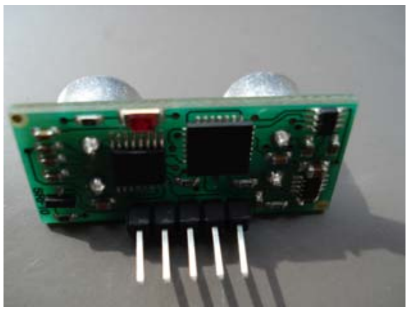


Figure 11, SRF10 ultrasonic sensor

The dc-dc step-up converter was built using an LM2703 dc-dc step-up converter while the voltage translators were built using BNS20 MOSFETs. The circuit diagrams of these circuits are shown below.

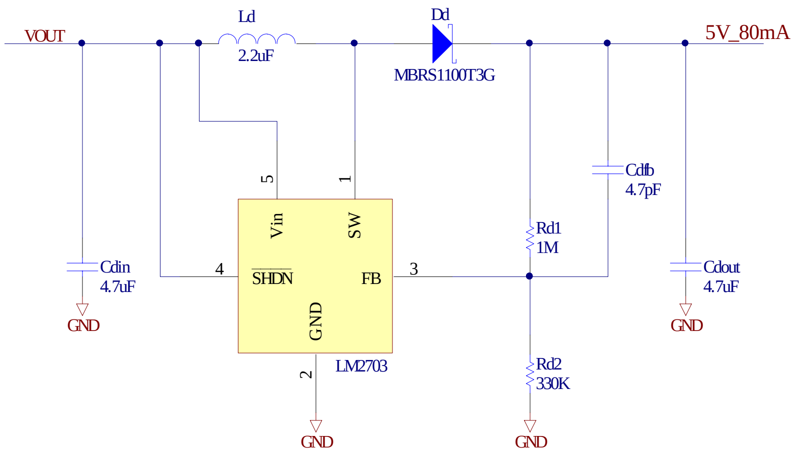


Figure 12, LM2703 step-up inverter circuit.

The ultrasonic module communicates with the FPGA using the i2c protocol. The main i2c bus to the FPGA is 3.3V hence the 5V signal voltages from the module must be translated to 3.3V. Figure 13 below shows the circuit diagram for this voltage translator circuit.

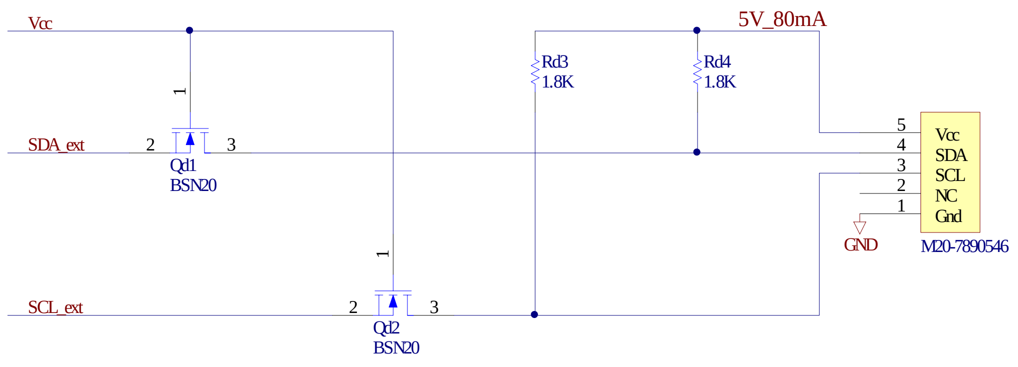


Figure 13, 5V – 3.3V voltage translator circuit.

The tests were done using an IAR Systems development kit with a Philips LPC2106 microcontroller based on the Arm7 core. The ultrasonic ranger was placed at different heights above the floor to come up with the lowest height above ground with minimal reflection from the floor, a height of 15mm above the ground proved to be optimum.

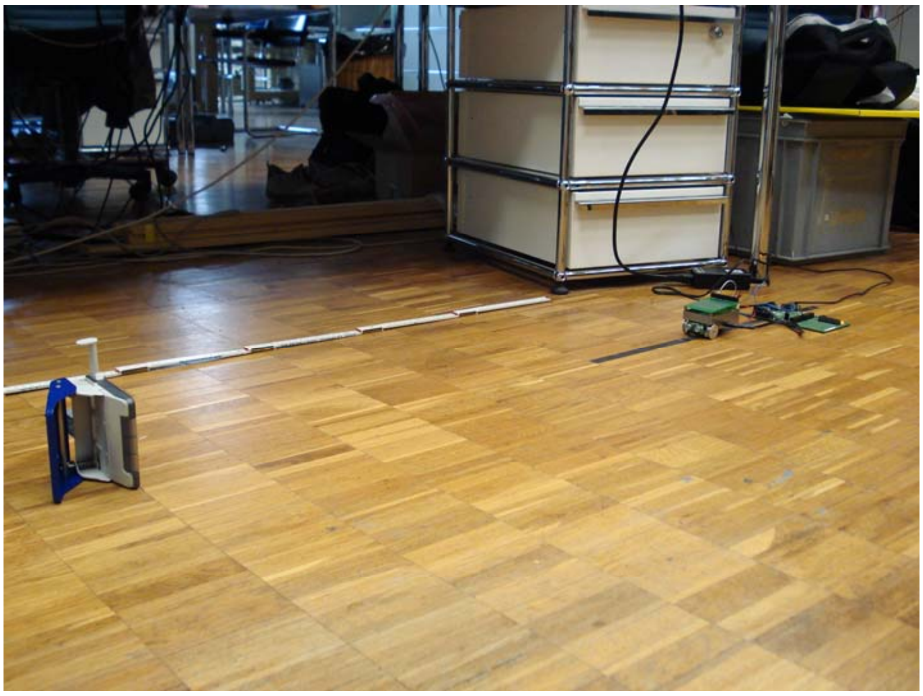


Figure 14, Ultrasonic distance tests

The results of the distance measurements were read on 8 LEDs on the IAR development kit. Some sample reading is shown below, figure 15. The reading is showing a binary value of 01011110 which is equal to 94cm.

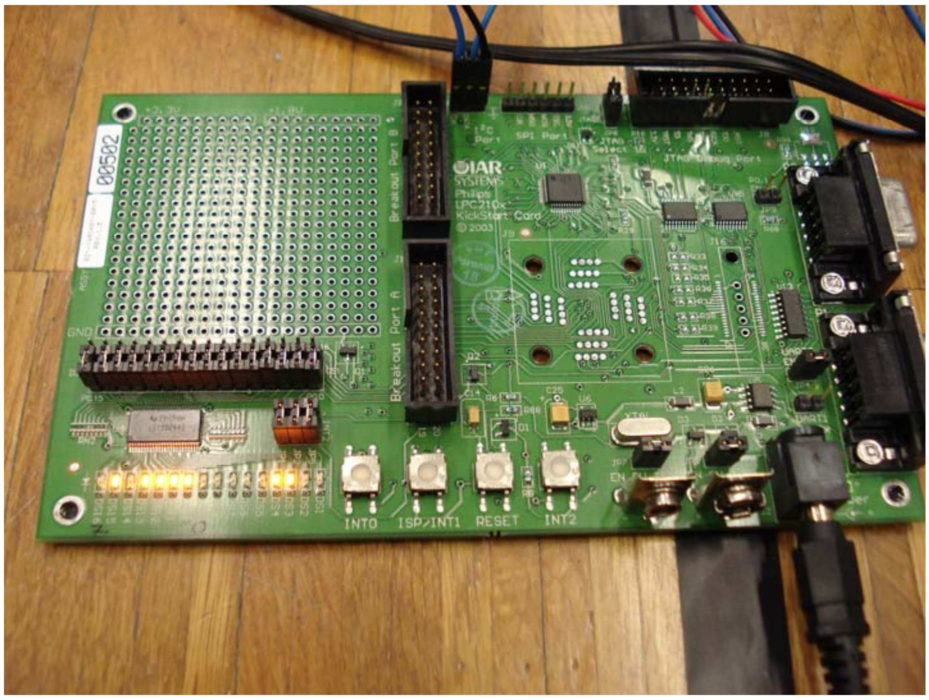


Figure 15

Results for two different conditions are shown in figure 16 and 17 below. The gain of the ultrasonic sensor was varied between 140 and 700 and the height was 15mm above the floor which is the height the sensor will be positioned on the robot.

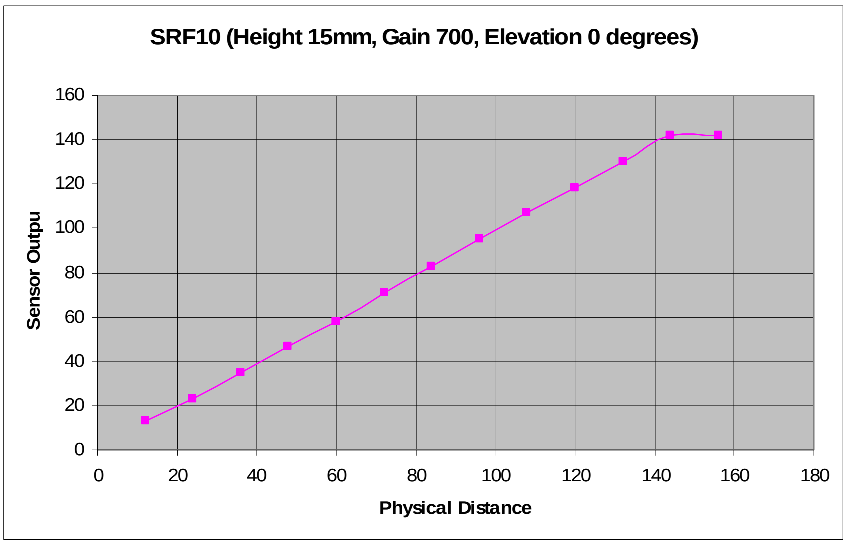


Figure16, Results at Gain = 700

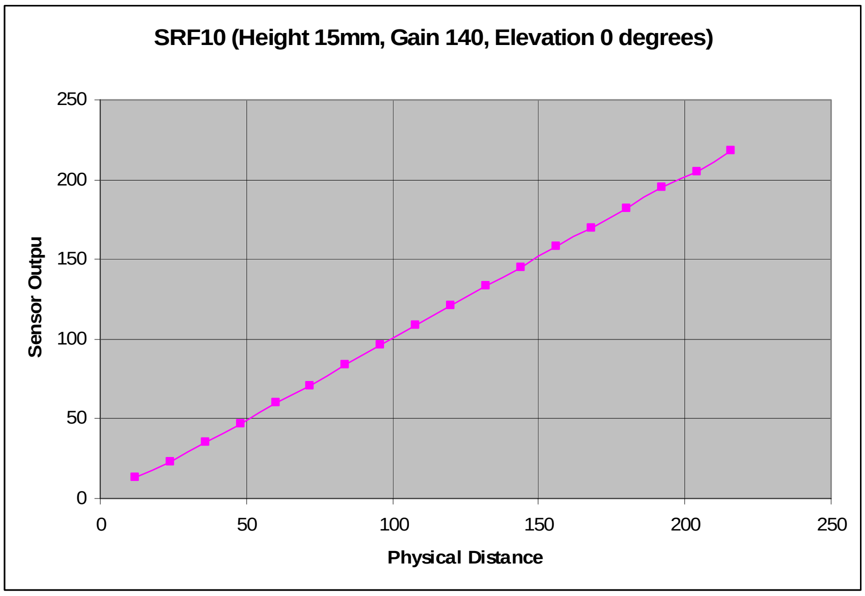


Figure 17, Results at Gain = 140

As shown from the graphs above, at a height of 15mm and the ultrasonic ranger with the maximum gain of 700, the sensors can detect the distance of objects accurately up to a maximum distance of about 140cm. After that the readings become constant due to the detection of the reflected waves by the floor. This is because the 400ST/R100 ultrasonic sensors used in this module have a wide beam angle of about 72 degrees at -6dB as shown in figure 18 below.

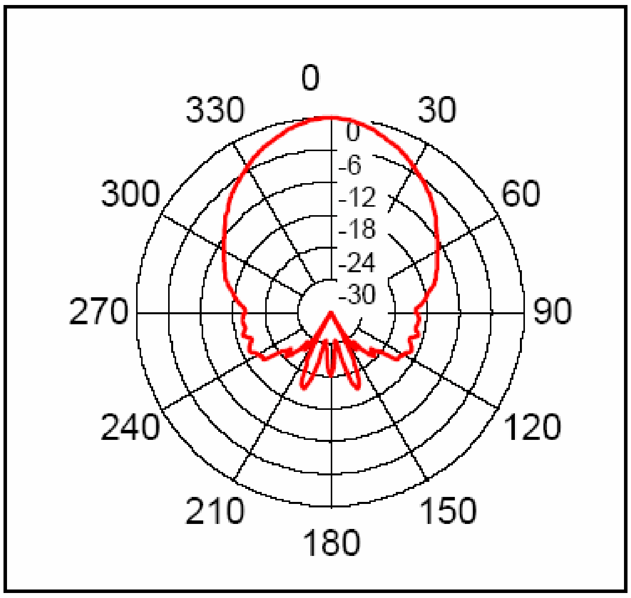


Figure 18, 400ST/R100 beam pattern

To get a better range of detection of at least 200cm (2m), the gain was reduced to 140. As shown above the results were satisfactory. The reduction of the gain caused the sensors to detect objects in the center of beam more effectively than those further from the centre. This came as an advantage since false alarms of objects outside the robot's path will be ignored.

Line Detector Sensors

The TCND5000 Infrared transmitter and receiver sensors are located on the power board at the bottom of the robot as shown in the diagram below.

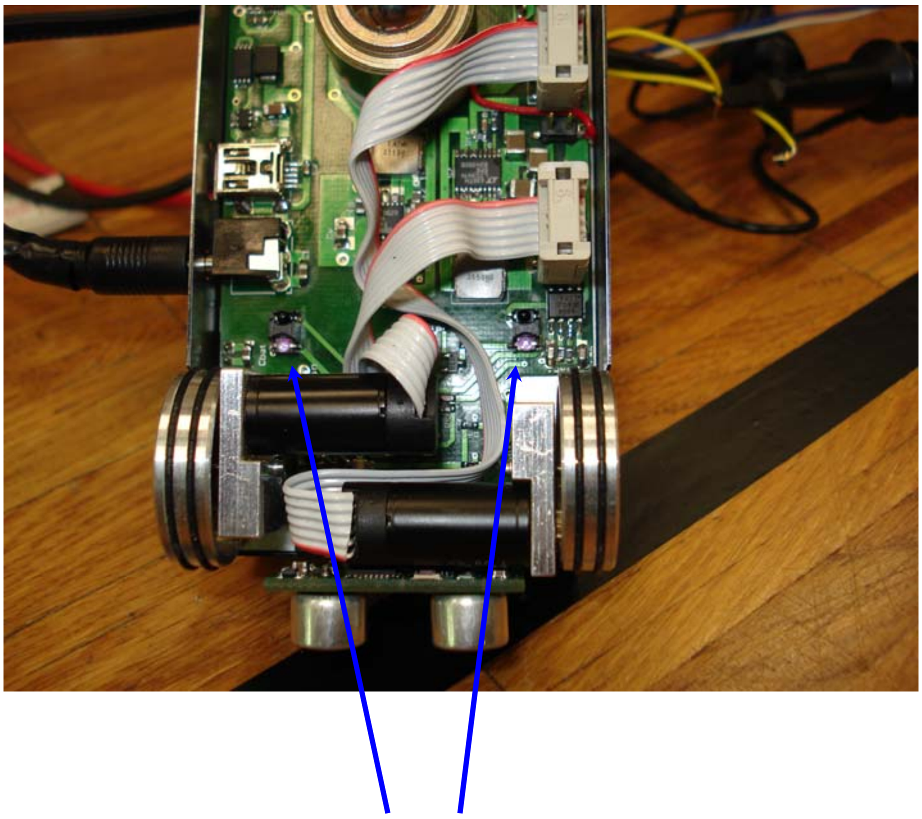


Figure 19 TCND5000 Sensors

The transmitter is an infrared diode and the detector is a photodiode. Since the detector is a photodiode, it functions like a current source.

As shown in the circuit diagram below, a lossy Sigma-Delta analog to digital converter was used. The lossy Sigma-Delta analog to digital converter is implemented using a 74HC74 flip-flop and a resistor and capacitor as external components only.

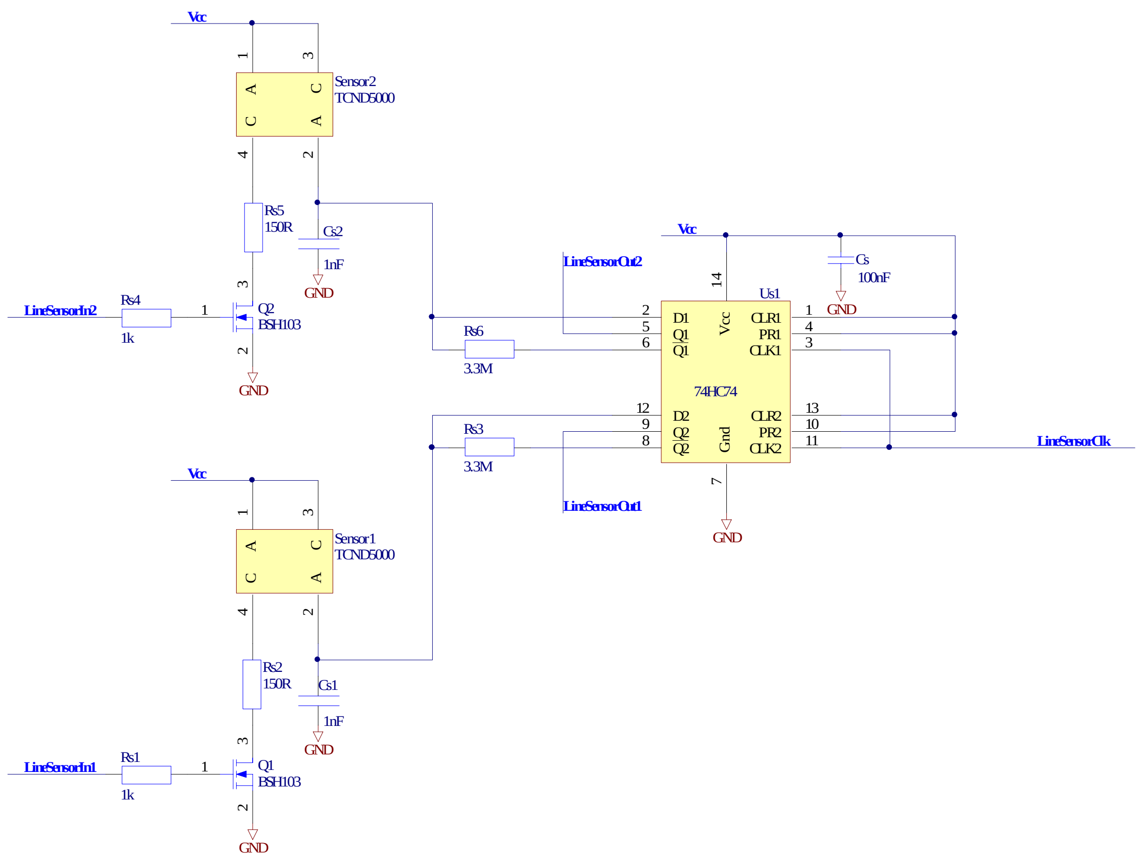
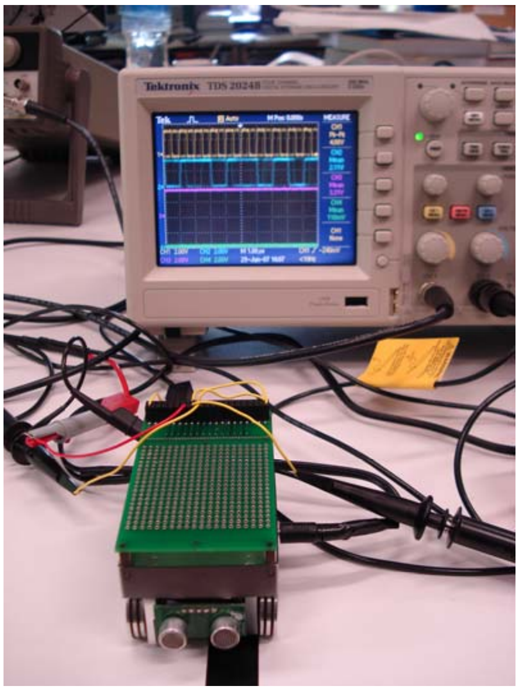


Figure 20, Infrared line detector circuit.

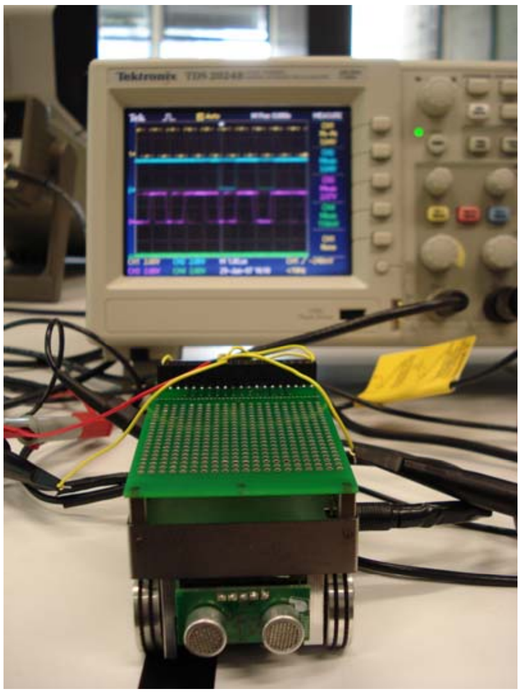
The output of the Sigma-Delta converter is a string of digital pulses that are counted over a given number of clock pulses. If the output of the sensor is high (more current), the output has more 1s (ones) than 0s (zeros) and vice-versa. By experimentation one can draw a relationship between the value of the output and the number of pulses, but for this application only a threshold needs to be set which when passed the output can be considered a high or low depending on the colour of the line the robot should follow and of surfaces the robot will be comparing the line with.

Shown below are images showing the outputs on the oscilloscope of the two sensors while placed on different parts of the line being tracked.

The yellow channel is the input clock at 1Mhz, the blue channel is the right sensor output and the pink channel is the output of the left sensor.



Left sensor on white surface, hence Output is high. Pink channel has more 1s.



Right sensor on white surface, hence Output is high. Blue channel has more 1s.

Figure 21, tests on the TCND5000 infrared line detectors.

4.2.4 GECKO3Power PCB design and fabrication

This four layer PCB was designed using Altium Designer software. Since the board is a densely populated credit card sized PCB, four layers were chosen to ensure the proper grounding of the dc-dc converters and also minimize the transmission of noise to the sensor analog circuits.

Figure 22 below shows the PCB being designed and analysed using Altium Designer software.

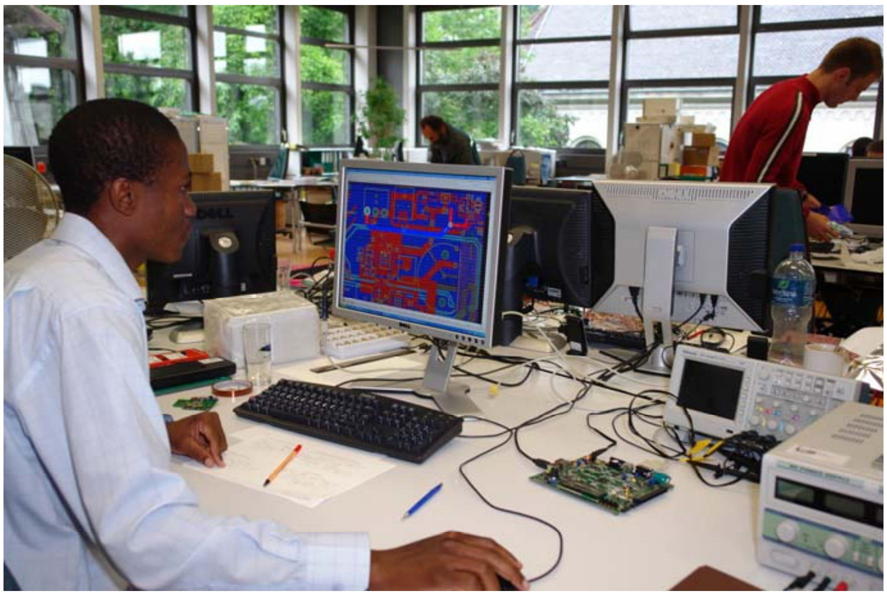


Figure 22

Figure 23 below shows the completed prototype PCB file with all the different layers clearly visible. The red and light blue layers constitute the signal layers and the brown and dark blue layers are the power and ground tracks respectively.

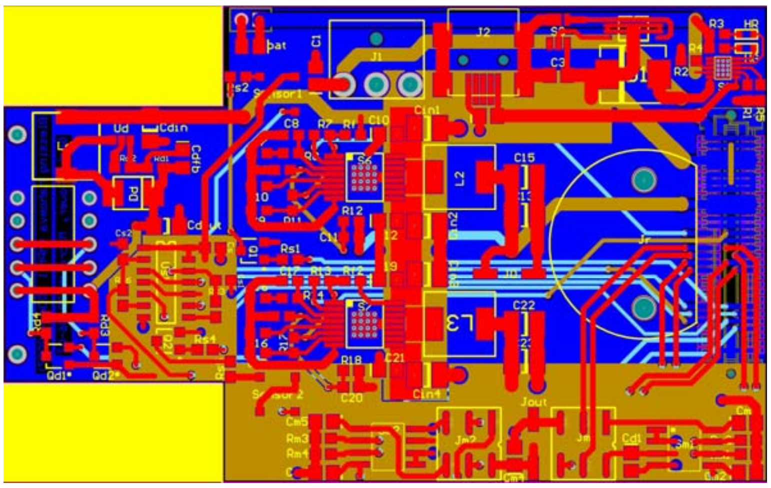


Figure 23

The following image, figure 24 shows the final prototype of this PCB under production.

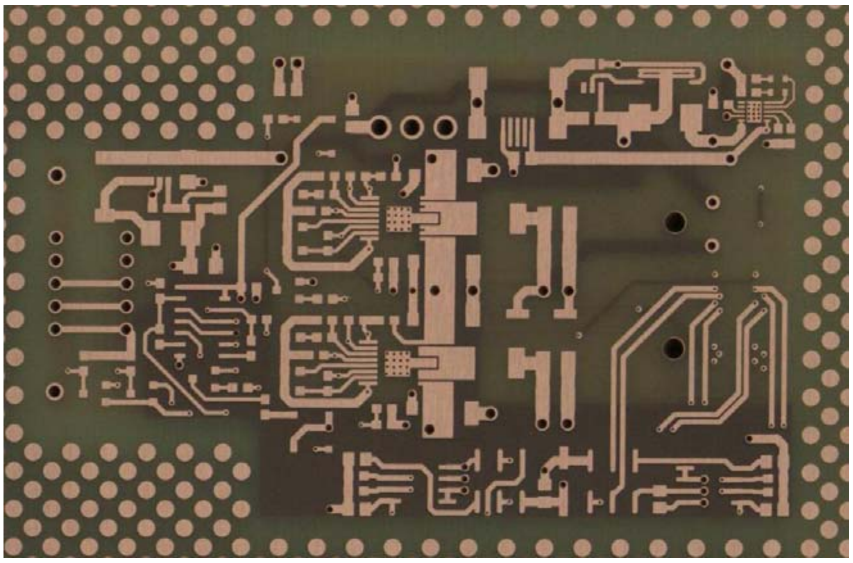


Figure 24

5. GECKO3Sensor

The GECKO3Sensor board, rich with high performance sensors will create a very good plant for researchers and students to explore the system-on-chip subject in depth. A number of sensors with raw outputs were included to allow the researchers to use IP cores of their choice to process the data or design their own cores. Both software and/or hardware solutions for these sensors can be implemented. The sensors included are a 2 dimensional MEM acceleration sensor, a 2 dimensional electronic compass and a Bluetooth radio. A CMOS camera and an OLED display were included as optional modules that can be plugged onto this board; these two modules give the researchers more computational intensive tasks.

Great care was taken in designing this board due to the mixed signals on this board. The power planes for the digital and analog section were separated to minimize digital noise entering the analog section of the board. The placing of the Bluetooth radio on the PCB was done in such a way to minimize the penetration of RF signals into the board.

5.1 TPS61092 5V power supply

The power section of this board was one of the critical sections since the digital and analog power planes had to be carefully separated and decoupled.

Figure 25 below is the circuit diagram for the 5V power supply. The filtering of the digital Vcc5V and the analog AVcc5V is done by Rp1 and the capacitors shown.

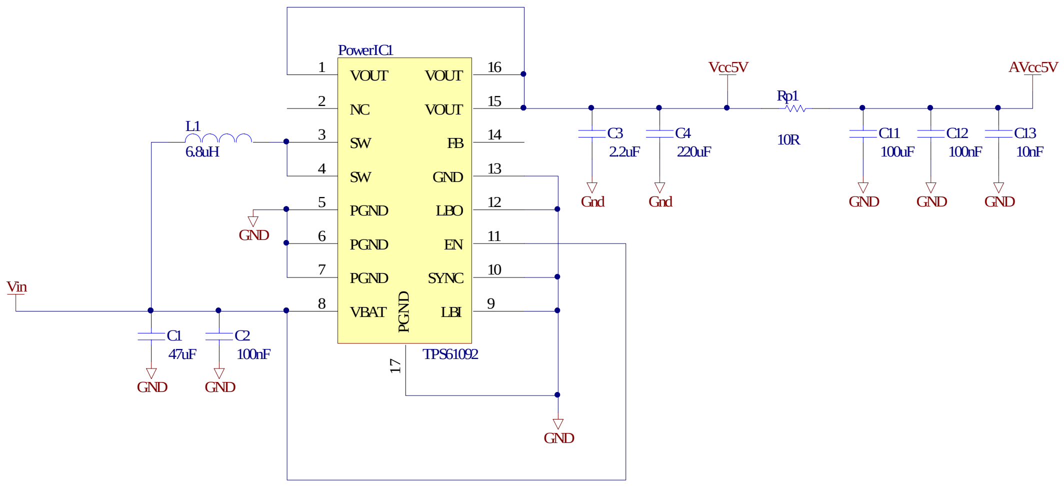


Figure 25

The step-up switching power supply circuit shown above is based on the Texas Instruments TPS61092. Vin is the power supply line from the power board. Vcc5V is the 5V digital supply for the camera and AVcc5V is the 5V analog supply for the electronic compass.

5.1.1 TPS61092 loading tests and results

Loading tests for this 5V power supply was done to determine if the power supply is capable of delivering the required power while its output voltage is still within the required range. The camera draws a maximum of 40mA while the flipping circuit for the camera draws a maximum of 1A. As shown in the loading results in figure 26 below, this circuit is capable of supplying up to 1.2A at an acceptable voltage of 4.8V with an input of 3.6V.

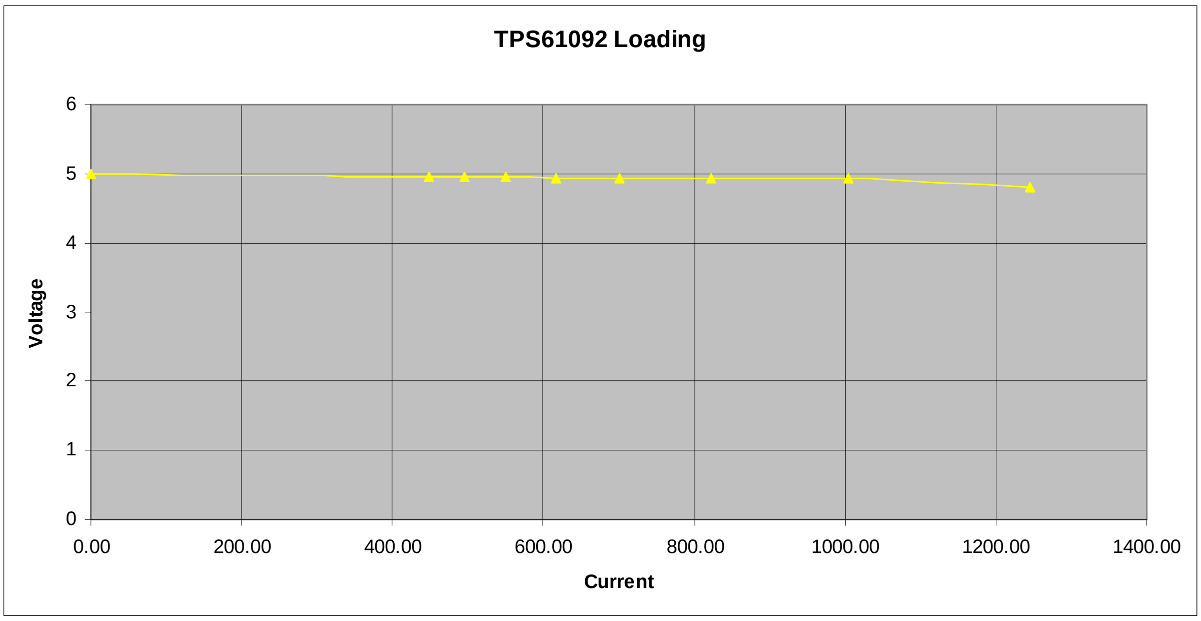


Figure 26

5.2 TPS79333 3.3V power supply

Figure 27 below shows the circuit diagram for the analog 3.3V power supply.

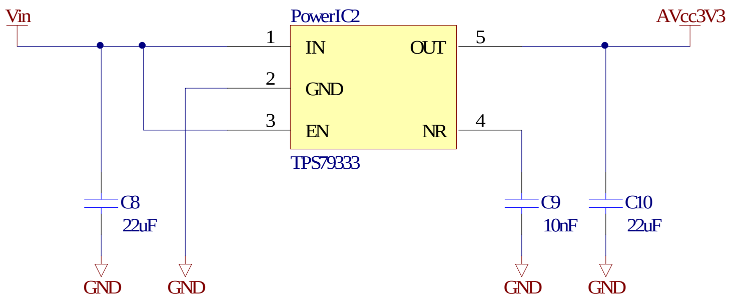


Figure 27

This linear regulator for the 3.3V analog circuits and the Bluetooth radio was designed using the Texas Instruments TPS79333 linear regulator. This is a Low Drop Out and low noise regulator ideal for radio and analog applications.

5.2.1 TPS79333 loading tests and results

Loading tests for the 3.3V shown above were done and the following results obtained, figure 28. The total current requirements of the 3.3V analog circuits is below 250mA, the results shown above indicate that this circuit can deliver up to 300mA at an acceptable output of 3.29V with an input of 3.6V.

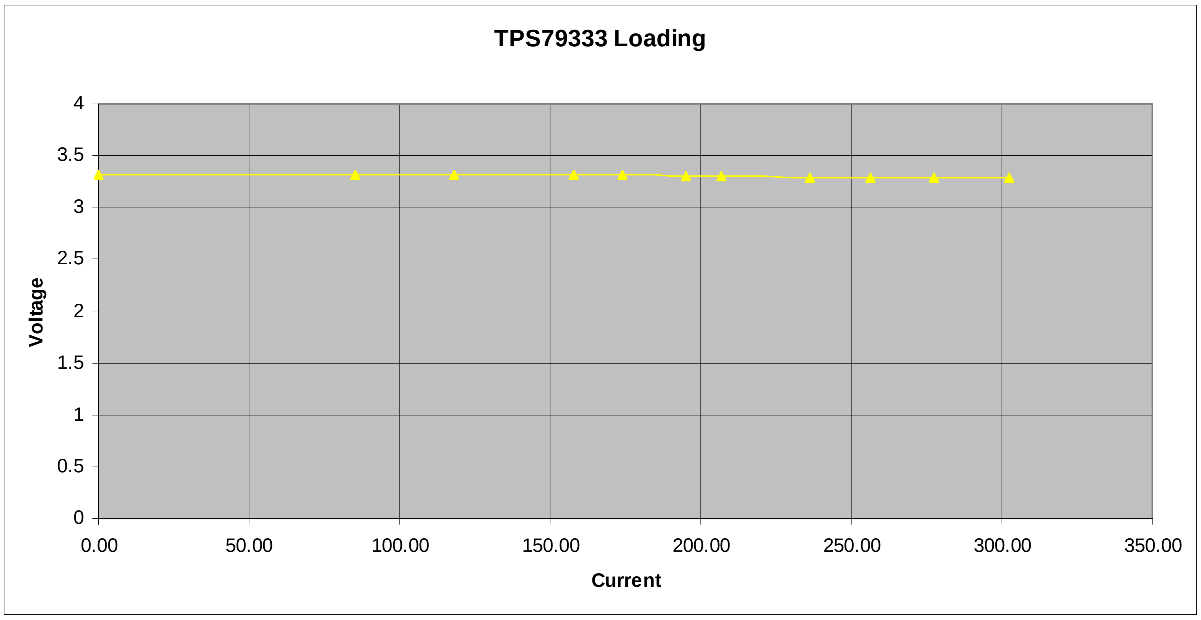


Figure 28

5.3 MEM Acceleration Sensor and Delta-Sigma IP-Core

The Micro Electro Mechanical (MEM) Acceleration sensor was included as part of the sensor board, this sensor can measure both dynamic acceleration (e.g. Vibration) and static acceleration (e.g. gravity). Different applications like the measuring of the acceleration of the robot, vertical inclination by measuring the component of the acceleration of free fall and collision detection of the robot with objects can be implemented.

The module is built using a dual-axis sensor, ADXL203E from analog devices. The output of the sensor is an analog voltage that is proportional to the acceleration. This output voltage can be converted to a digital signal using an analog to digital converter for easy interfacing with the FPGA.

In order to minimize components and use the available IP cores, the Delta-Sigma Analog to Digital converter (ADC) IP from Xilinx is implemented. This soft IP core is designed to interface with the PLB (Processor Local Bus). The general circuit diagram of the Delta-Sigma ADC with its external components is shown in figure 29 below.

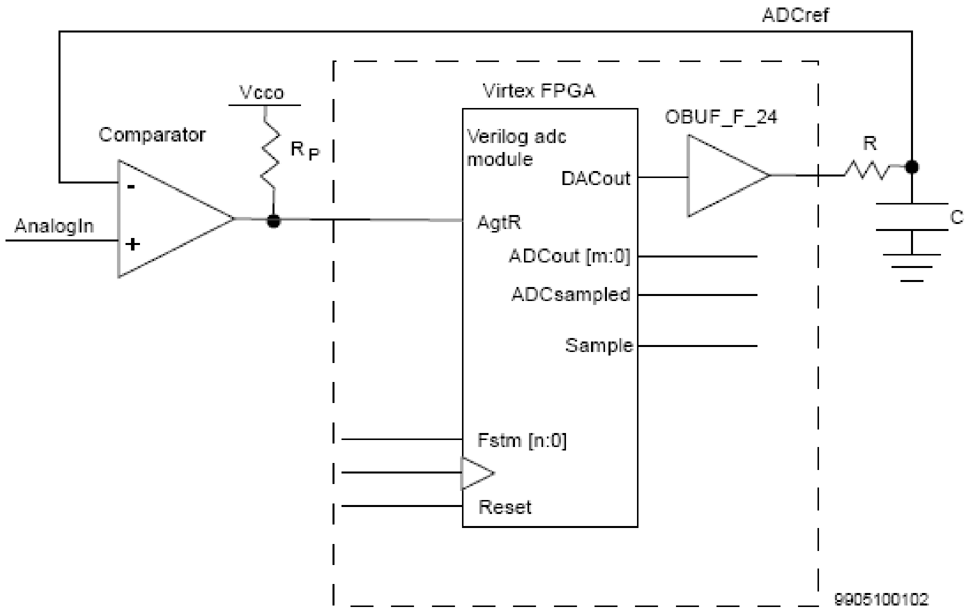


Figure 29

This Delta-Sigma ADC uses an internal Delta-Sigma digital to analog converter (DAC), the DAC is used to generate a reference voltage for the negative input to the external comparator. The analog signal feeds the positive input of the comparator. The voltage range of the DAC output is 0V to Vcco, so that is also the range of analog voltages that may be converted. The analog level is determined by performing a serial binary voltage search, starting at the middle of the voltage range. For each complete sample, only the upper bit of the DAC input is initially set, which drives the reference voltage to midrange. Depending on the output of the comparator, the upper bit is then cleared or it remains set, and the next most significant bit of the DAC input is set. This process continues for each bit of the DAC input.

Because of the serial nature of both the DAC and the analog sampling process, this ADC is useful only on signals that are changing at a fairly low rate. Since our sensor can be used effectively up to 60Hz with a resolution of 1mg, this IP core is perfect for our application. The circuit diagram implemented on this board is shown in figure 30 below.

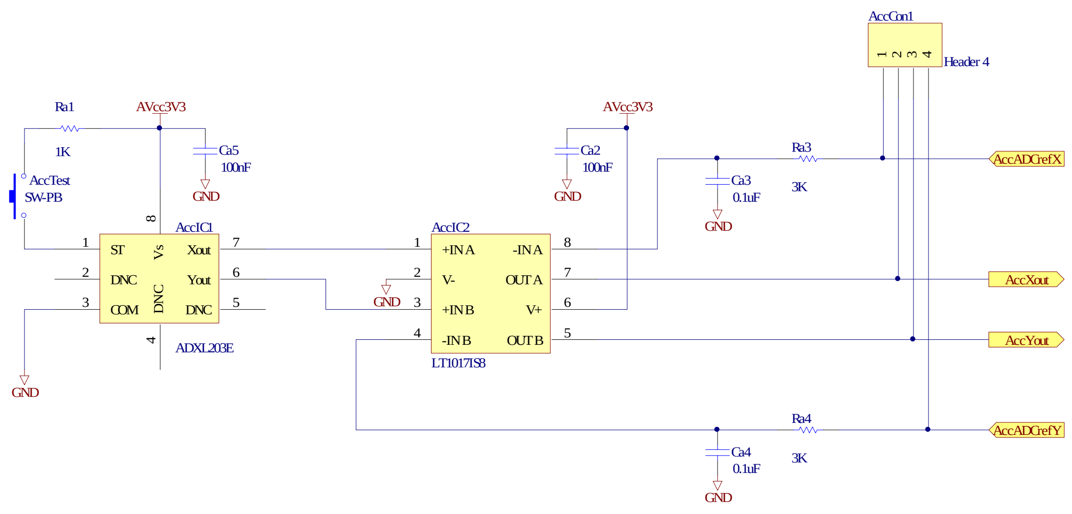


Figure 30

5.3.1 Delta-Sigma ADC tests and results

To demonstrate and verify the functionality of this IP core with a more complex signal, a 50Hz sinusoidal analog signal was input on the positive input of the comparator.

The idea was to see if the filtered signal from the internal DAC followed the analog input because this value is the one that is read internally.

The result is shown in figure 31 below. The orange signal is the original signal while the green signal is the pulse string output of the DAC before the filter and the blue signal is the reconstructed signal after passing through the filter.

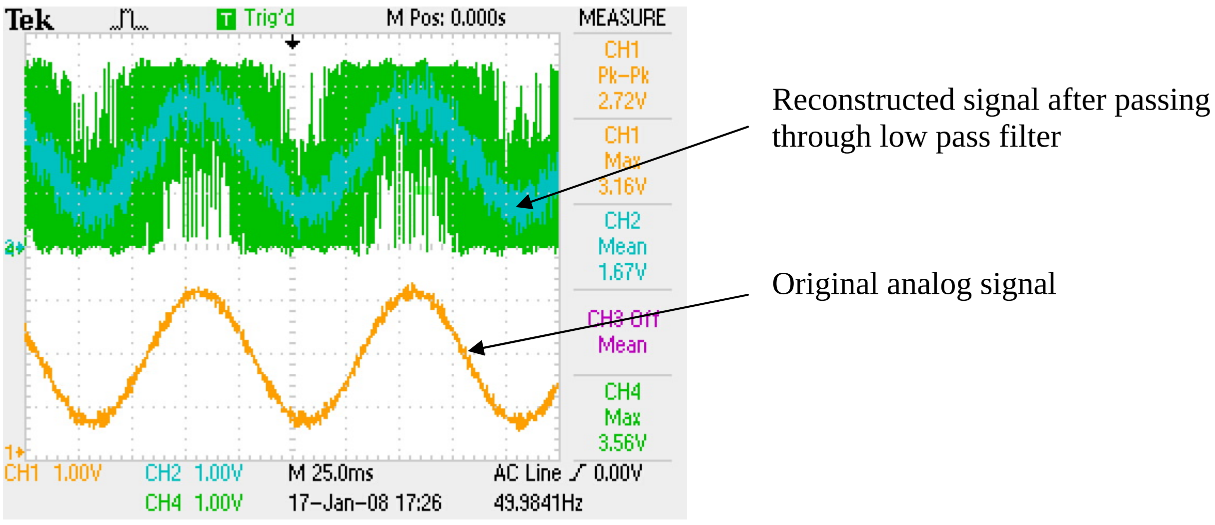


Figure 31

5.4 Compass Module and SPI IP-Core

A compass module that could be used by the robot for navigation purposes and other applications was also included.

This sensor module was built using an extremely sensitive two-dimensional magnetic field sensor, KMZ52 electronic compass chip made by Philips semiconductors. This sensor can sense and measure the earth's weak magnetic field.

The circuit diagram for this sensor module is shown in figure 32 below.

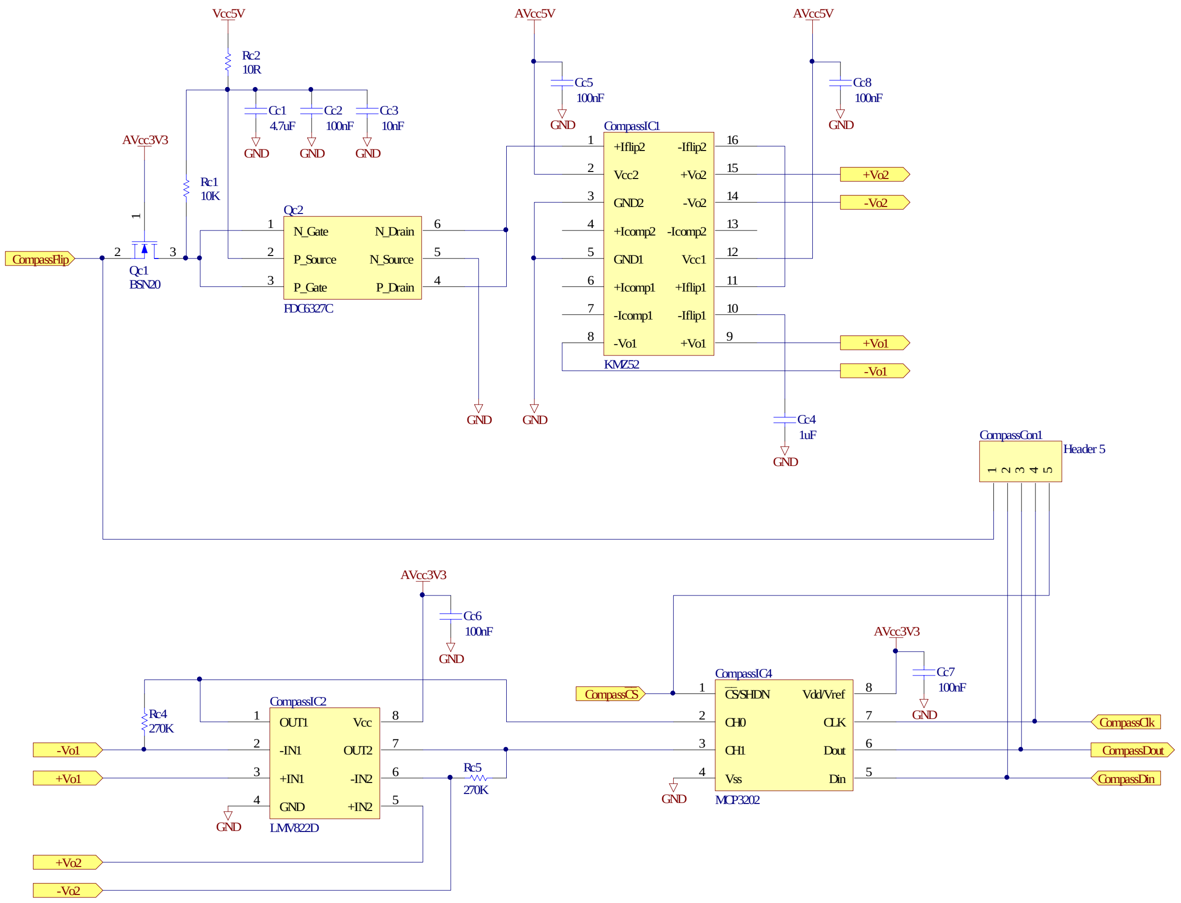


Figure 32

As seen from the diagram above, there are three different voltages supplying this module, Vcc5V for the flipping circuit, AVcc5V for the analog compass module chip and AVcc3V3 for the amplifier and the analog to digital converter.

The output is a digital signal in SPI serial format from an MCP3202 ADC from Microchip. Hence the interface to the FPGA must be able to interpret the SPI communication protocol. The Xilinx OPB Serial Peripheral Interface (SPI) soft IP core used to interface with the ADC connects to the OPB and provides the controller interface to any SPI device such as EEPROMs. The OPB SPI module is a full-duplex synchronous channel that supports four-wire interface (receive, transmit, clock and slave select) between one master and one slave. If more than one slave or masters are to be implemented, to eliminate conflicts, software is required to arbitrate bus control.

The OPB SPI modules are shown in the top level block diagram in figure 33.

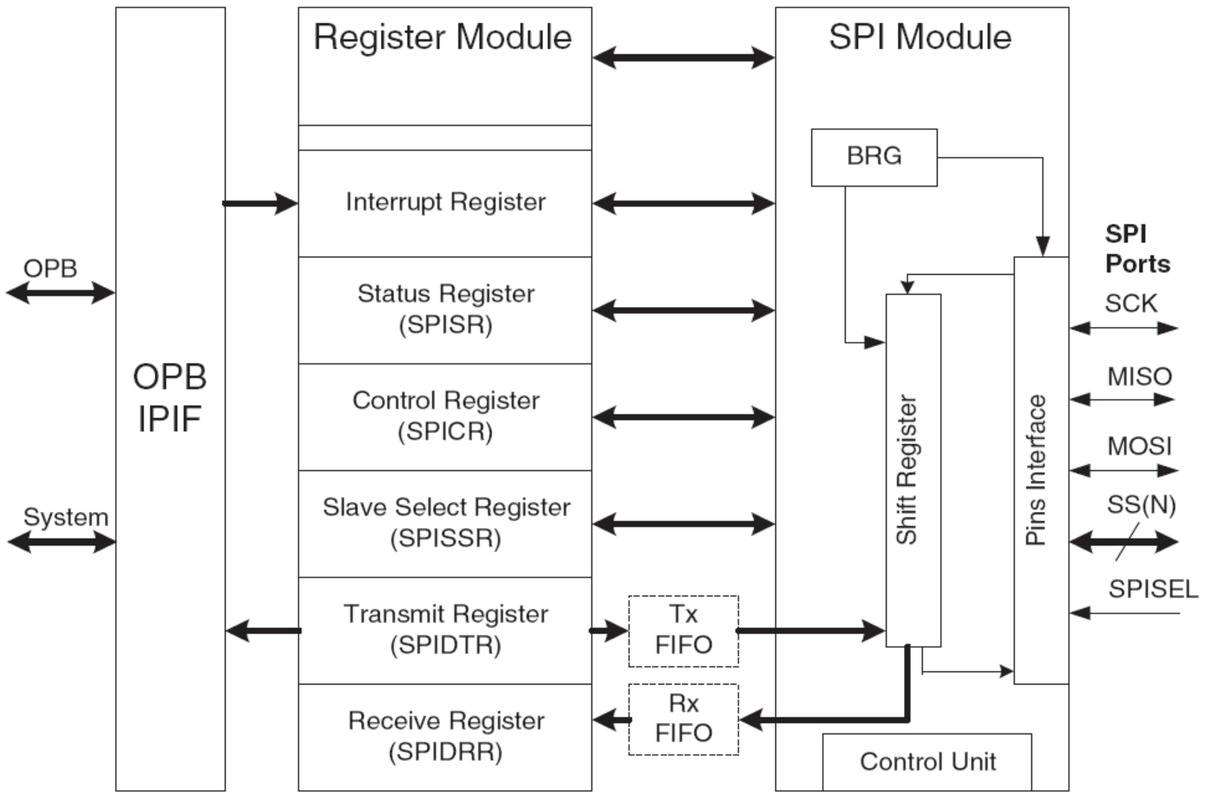


Figure 33

5.4.1 Compass module and OPB SPI soft IP core tests and results

The board was rotated through 360° to obtain its behavior in the earth's magnetic field. This experiment was done away from other magnetic fields so that a near to true earth's field is represented by the results.

The result of the two outputs X and Y are shown in figure 34 below.

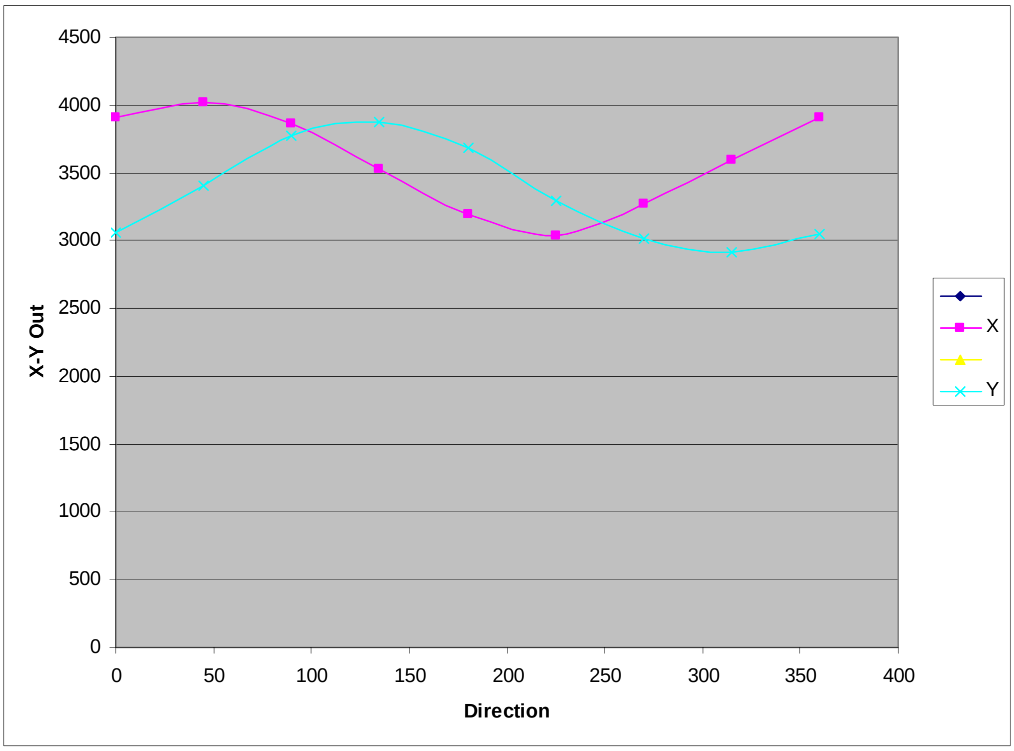


Figure 34

These results are consistent with the theoretical results obtained from the Philips application notes for this compass chip, AN00022. The angles need not to be the same due to the different reference points.

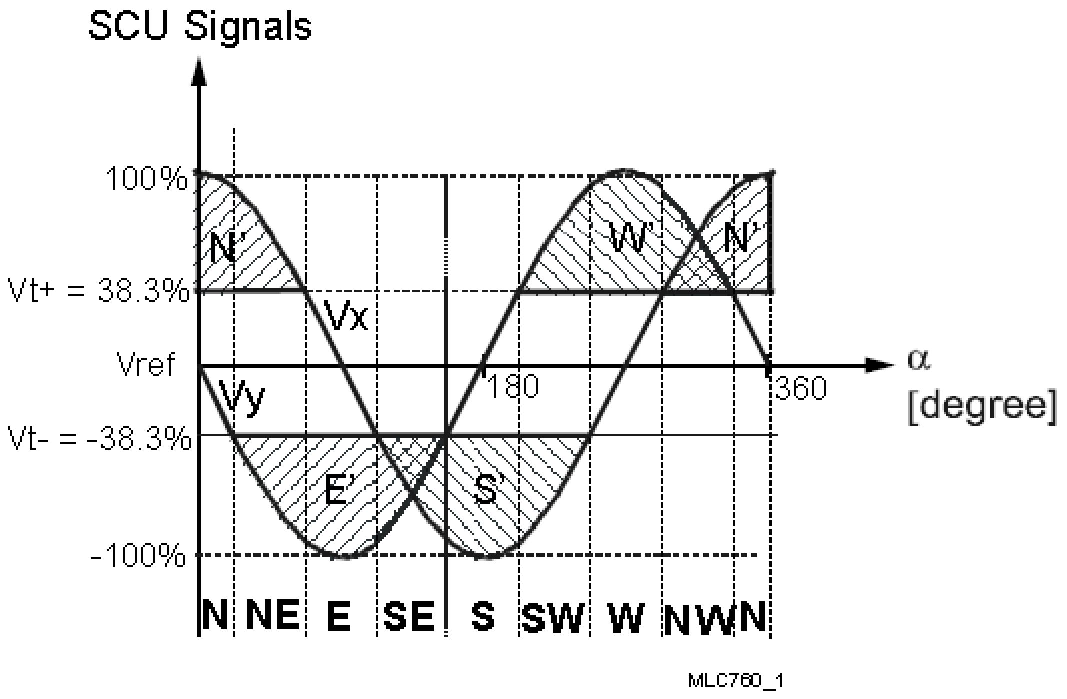


Figure 35

In practice, the earth field at the compass may be superimposed by other magnetic fields or distorted by nearby ferrous materials. An efficient compensation of such effects is required in order to achieve reliable azimuth readings.

As for any sensor system, only errors caused by deterministic interference sources can be compensated. In this case deterministic means that the interference source is at a fixed position relative to the compass and that its magnitude is constant versus time. The sensor might also need to be compensated from tilt. All these effects that need to be compensated give a computational intensive challenge to the researcher designing the hardware and/or software for this module. These open challenges make this module an effective research tool.

5.5 Bluetooth® radio

The Bluetooth radio used is the WT11-A made by Bluegiga. WT11 is class 1 Bluetooth 2.0+EDR (Enhanced Data Rates) module. WT11 is an integrated and sophisticated Bluetooth module, containing all the necessary elements from Bluetooth radio to antenna and a fully implemented protocol stack. Therefore WT11 provides an ideal solution for developers who want to integrate Bluetooth wireless technology into their design with limited knowledge of Bluetooth and RF technologies.

By default WT11 module is equipped with an easy-to-use iWRAP firmware. iWRAP enables users to access Bluetooth functionality with simple ASCII commands delivered to the module over serial interface.

The circuit diagram for this radio and supporting components is shown in figure 36 below.

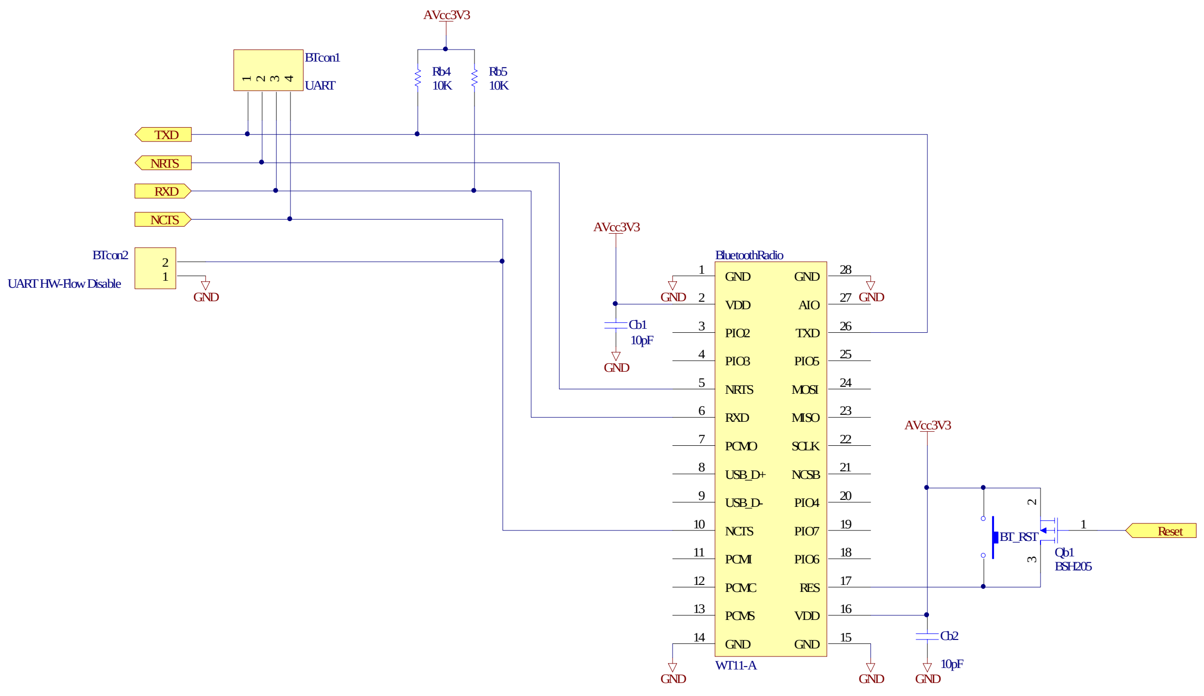


Figure 36

5.5.1 iWRAP

iWRAP is an embedded firmware running entirely on the RISC processor of WRAP THOR modules. It implements the full Bluetooth protocol stack as illustrated in figure 37 below, and no host processor is required to run it.

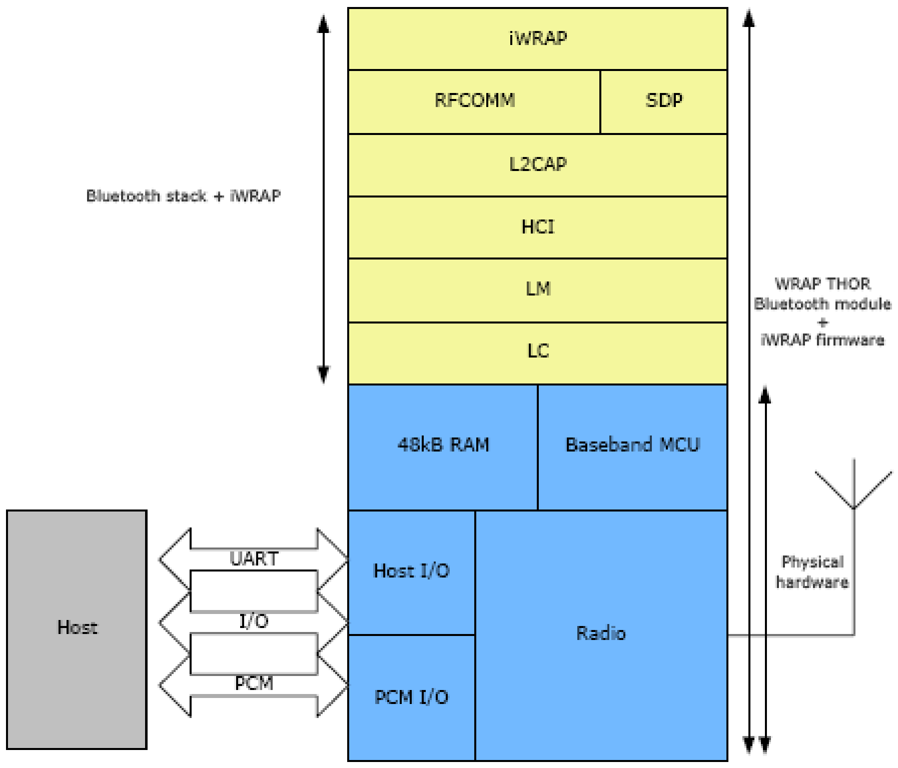


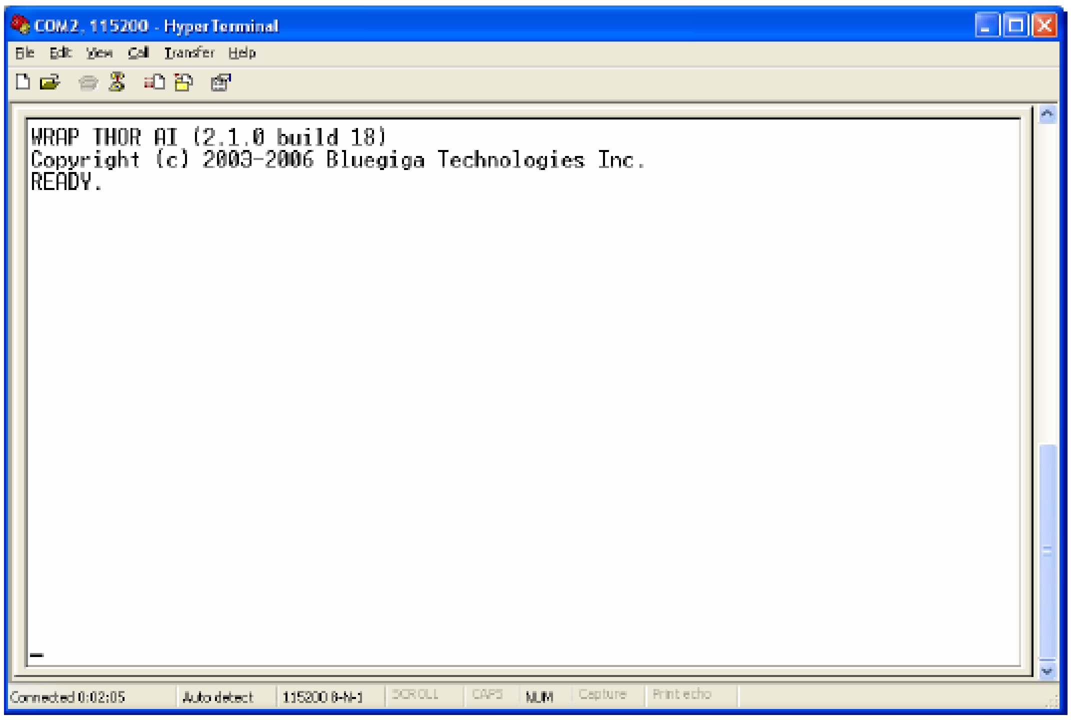
Figure 37

All software layers, including application software, run on the internal RISC processor in a protected user execution environment known as the Virtual Machine (VM).

The host processor interfaces to iWRAP firmware through one or more of the physical interfaces, which are also shown in figure 37. The most common interfacing is done through the UART interface by using the ASCII commands that iWRAP firmware supports. With these ASCII commands, the host can access Bluetooth functionality without paying attention to the complexity, which lies in the Bluetooth protocol stack.

The user can write application code, which runs on the host processor and controls iWRAP firmware with ASCII commands.

If the module is connected to a PC serial port, one can use terminal software such as HyperTerminal to communicate and configure the module. Shown in figure 38 below is the boot prompt of the module.



```
WRAP THOR AI (2.1.0 build 18)
Copyright (c) 2003-2006 Bluegiga Technologies Inc.
READY.
```

Figure 38

When showing ready, the module is in command mode, this is when it is ready to receive ASCII commands. The ASCII commands are listed in the iWRAP user guide. This can also be done from a host processor.

5.5.2 Tests and results

This radio is interfaced to the FPGA using a Xilinx UART Lite soft IP core. This is a full duplex interface; ASCII commands are sent and received using only two wires. To communicate, only the baud rate and parity need to be configured.

To test the Bluetooth radio communication, the compass test was repeated this time transmitting the readings to the computer via Bluetooth radio using SPP (Serial Port Profile) which is a virtual wireless UART serial interface instead of a hard wired serial link. The results obtained where the same as those got without the radio.

The behavior of the other sensors remained the same with and without the radio showing that no RF interference was penetrating the PCB.

5.5.3 Further Applications

This Bluetooth radio can be used by students and researchers for communication, and also for trial navigation applications by measuring RSSI (Receiver Signal Strength Indication) which gives a rough estimation of the distance between the receiver and transmitter. With some complex geometry and mathematics and at least three transmitters in a given area the position of the robot can be estimated in that particular area. With a number of robots, interesting experiments of robot in a group can be done.

5.6 CMOS Camera and OLED Display

The CMOS Camera module (C3188A) and the OLED Display are two optional modules that were added onto the GECKO3Sensor board to enable students and researchers to do some images processing experiments. This will ensure that some more computational intensive experiments are carried out in software and/or hardware.

5.6.1 C3188A CMOS Camera Module

The C3188A is a 1/3" colour camera module with digital output. It uses OmniVision's CMOS image sensor OV7620. The digital video port supplies a continuous 8/16 bit-wide image data stream. All camera functions, such as exposure, gamma, gain, white balance, colour matrix, windowing, are programmable through 12C interface. Figure 39 below shows the circuit diagram of the camera module and its power supply interface.

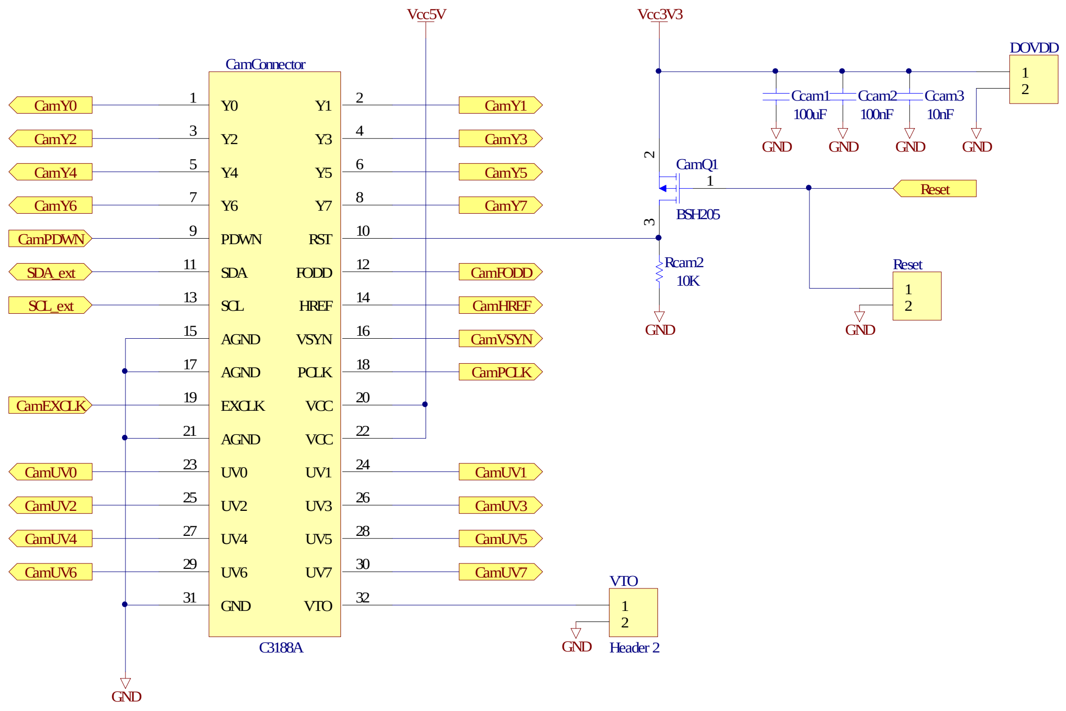

Vcc5V
Vcc3V3
CamConnector
C3188A
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
PDWN RST SDA FODD SCL HREF AGND VSYN AGND PCLK EXCLK VCC AGND VCC
UV0 UV1 UV2 UV3 UV4 UV5 UV6 UV7 GND VTO
CamY0 CamY1 CamY2 CamY3 CamY4 CamY5 CamY6 CamY7
CamPDWN SDA_ext SCL_ext CamEXCLK CamFODD CamHREF CamVSYN CamPCLK
CamUV0 CamUV1 CamUV2 CamUV3 CamUV4 CamUV5 CamUV6 CamUV7
DOVDD
Ccam1 100uF
Ccam2 100nF
Ccam3 10nF
CamQ1
BSH205
Reset
Rcam2 10K
VTO
Header 2
GND

Figure 39

This camera module is supplied by a 3.3V dc supply for its digital I/O and a 5V dc supply that powers the whole camera. This camera also has an analog monochrome output that can be used if required.

5.6.2 P11701 OLED Display

The P11701 is a 128x3x160 full colour passive OLED (Organic LED) module with a controller for many compact portable applications. The function block diagram for the display module is shown in figure 40 below.

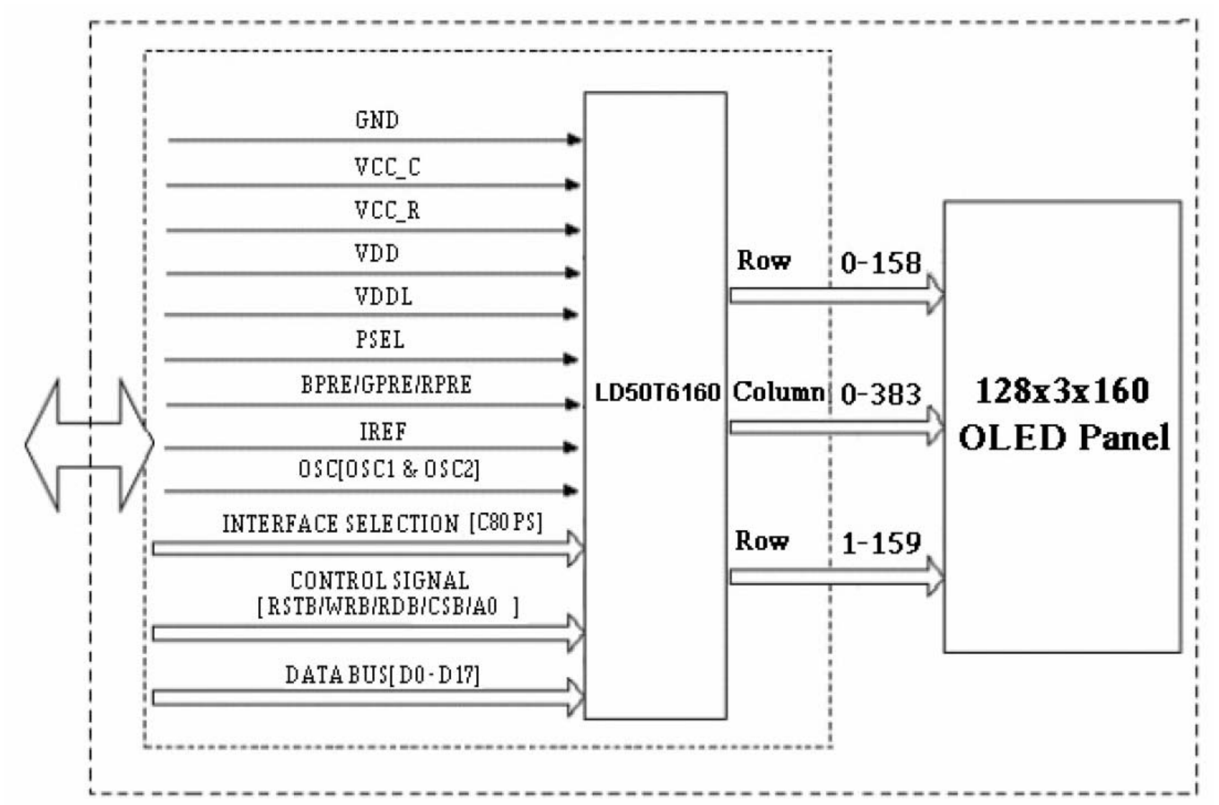


Figure 40

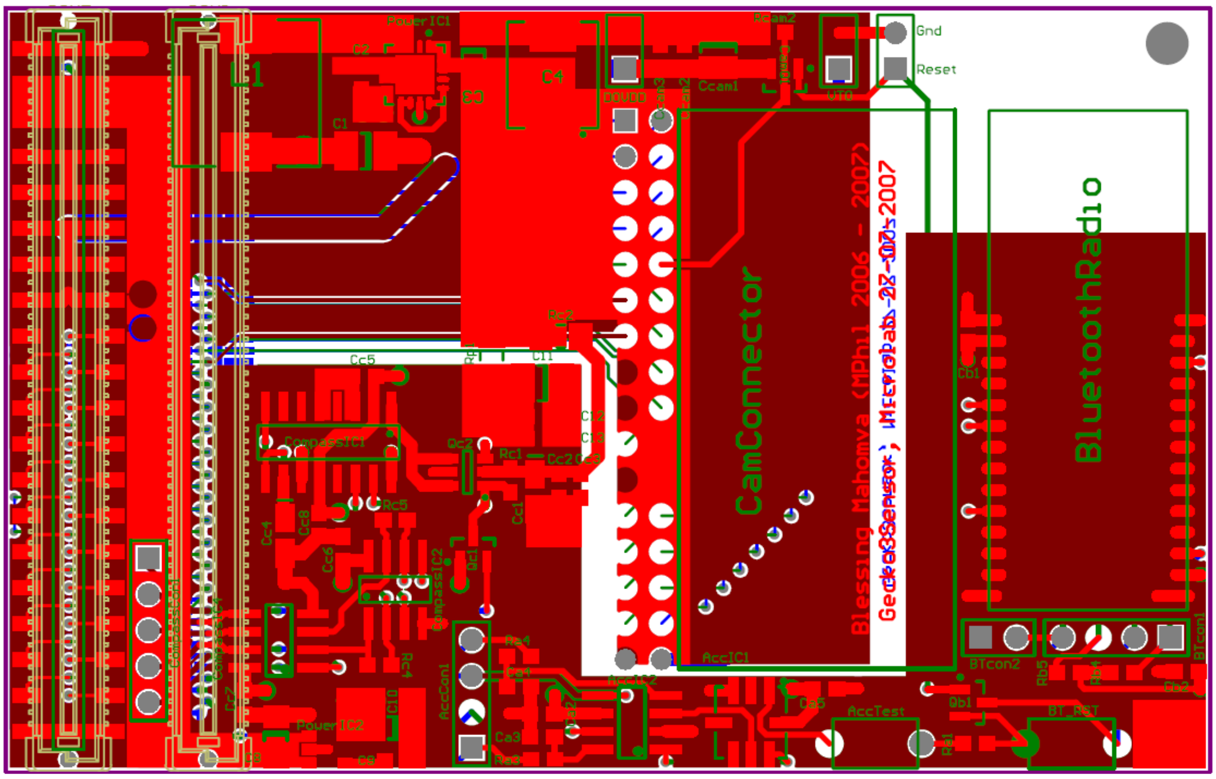
This optional OLED display was experimented on and tested by a NUST student who was attached at the Bern University of Applied Sciences as his attachment project and the results were impressive. The circuit that was designed and tested was adopted and modified to fit as a plug in board to the sensor board.

5.7 PCB Design

Due to a mixed signal nature of this board and a radio module on the GECKO3Sensor board, a lot of care was taken in the design of the PCB board.

The board was designed in such a way to minimise the interference that the switching of digital signals can cause on the analog signals. The PCB board around the Bluetooth radio module was also designed to avoid the penetration of RF signals into the board.

The Altium Designer software was used to design this four layer board. Figure 41 below shows the developed board as seen in the Altium Designer tool. As seen from the figure below, the ground planes, shown in maroon, were separated for the analog and digital sections of the board. The power plans were also separated to reduce the effects of digital signals on the analog signals. The antenna of the Bluetooth radio was placed on the portion of the PCB that is not routed; this was done to ensure that no tracks under the antenna will pick up any RF signals.



CamConnector
BluetoothRadio
Blessing Mahomva (MPhil 2006 - 2007)
Gecko3Sensor, Microlab 22-07-2007
Gnd
Reset

Figure 41

Figure 42 below shows the PCB board that has been developed from the Altium Designer image shown in figure 41 above. Here the board has been completely fabricated before cutting it to the right size. There are a number of vias seen around the Bluetooth radio area; these also help to reduce the penetration of RF signals into the PCB board.

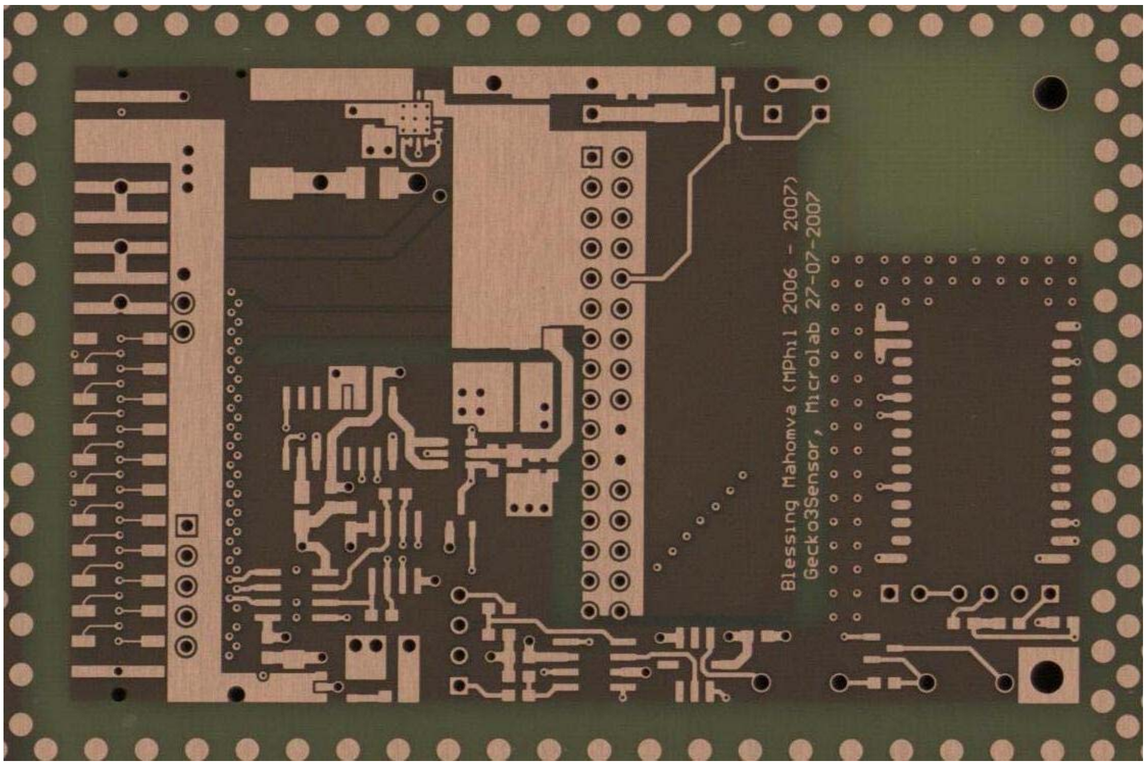

Blessing Mahomva (MPhil 2006 - 2007)
Gecko3Sensor, Microlab 27-07-2007

Figure 42

6. Finished Prototypes

The three prototypes, GECKO3Basic, GECKO3Power and the GECKO3Sensor that were designed and developed in this MPhil project were completed successfully. The GECKO3Basic and the GECKO3Power boards are already being used in lessons at the Bern University of Applied Sciences.

The following images show these completed boards.

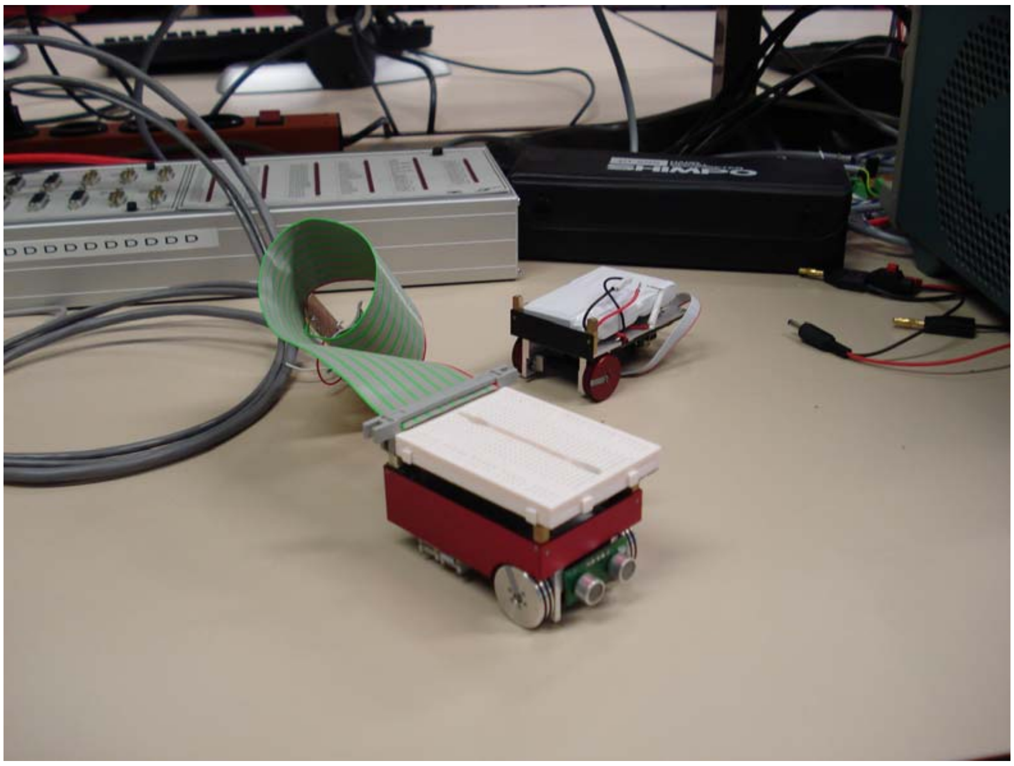


Figure 43, GECKO3Basic and GECKO3Power

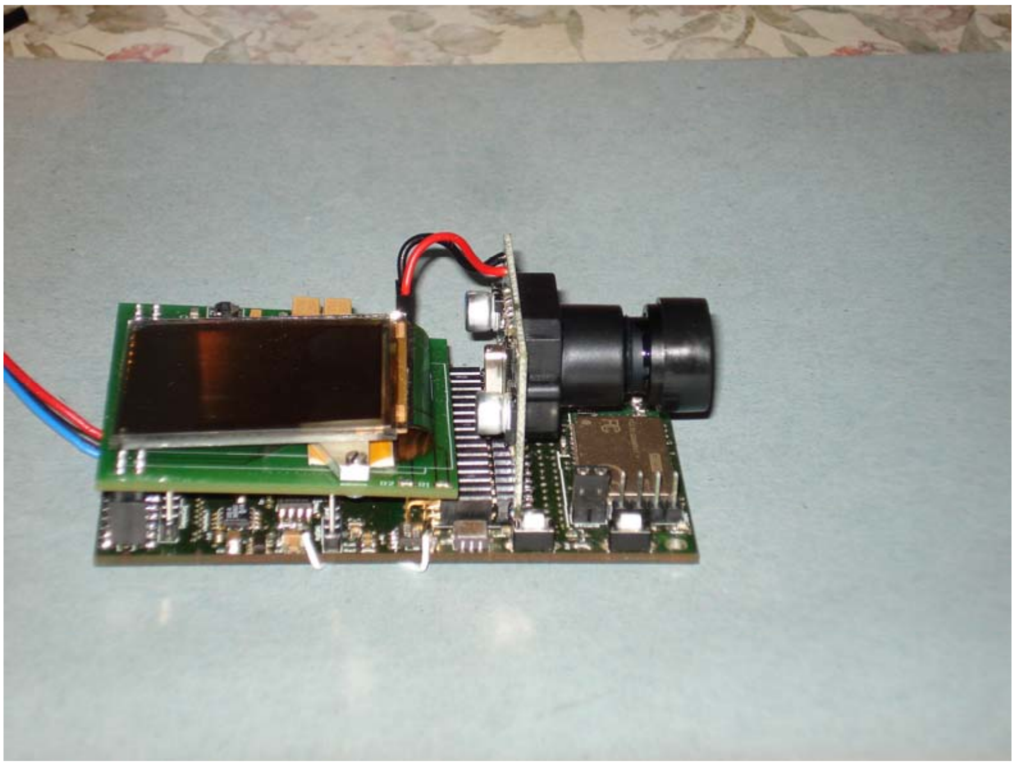


Figure 44, GECKO3Sensor

7. Conclusion

The objectives of this project which were to design and develop the GECKO3Basic, GECKO3Power and GECKO3Sensor board which made up important parts of the SoC research platform for research and educational purposes were met.
Test and experiments were carried out on all these three boards and the obtained results were satisfactory as shown in the previous chapters. By the end of 2007 the GECKO3Basic and GECKO3Power boards were already being used by the Bern University of Applied Sciences for educational purposes in the DSP course.

Now for this subject that has proved to be of great importance in the Electronic Engineering curriculum can be introduced at the National University of Science and Technology, NUST to help students design and develop sophisticated products with the use of very little hardware. This does not only help them design complex projects but it also reduces the cost significantly both in terms of time and money since a lot of discrete components can be replaced by soft IP cores which the students and researchers either design themselves or use pre-designed ones.

Xilinx has a university program that they launched to support universities in the SoC subject. Through this program some development tools are sold to universities at reduced prices. This program can be taken advantage of by universities in the developing countries.

BIBLIOGRAPHY

1. Resve Saleh, Shahriar Mirabbasi, Guy Lemieux and Cristian Grecu, “System-on-Chip: Reuse and Integration”, Proceedings of the IEEE, Vol. 94, No. 6, June 2006.

2. Dario L. Sancho-Pradel and Roger M. Goodall, “System-on-Chip (SoC) Design for Embedded Real-Time Control Applications”, Department of Electronic and Electrical Engineering, Loughborough University, LE11 3TU, UK, 2003.

3. William D. Mensch, Jr. and Dennis A. Silage, “System-on-Chip design methodology in engineering education”, The Western Design Center, Inc and System Chip Design Center, Electrical and Computer Engineering, Temple University, Philadelphia, USA.

4. Shebli Anver, Olivier Gachelin, Pierre Kestener, Herve Le Provost and Irakli Mandjavidze, “FPGA-based System-on-Chip Designs for Real-Time Applications in Particle Physics”, 14th IEEE Real Time Conference, Stockholm, Sweden, June 6-10, 2005.

5. Marcel Jacomet, Jörg Breittenstein, Markus Hager and William Chigutsa, “The GECKO System”, GECKO v2.0, Microlab, January 29, 2003.

6. Analog Devices, ADXL103/203 data sheets, Analog Devices, Accessed 19 Jan 2007, http://www.analog.com/UploadedFiles/Data_Sheets/279349530ADXL103_203_0.pdf

7. Philips Semiconductors, KMZ51/52 data sheets and application notes, Philips Semiconductors, Accessed 18 Jan 2007,
http://www.nxp.com/acrobat_download/datasheets/KMZ52_1.pdf
http://www.nxp.com/acrobat_download/applicationnotes/AN00022_COMPASS.pdf

8. Xilinx Inc, System Generator tools, Xilinx, Accessed 20 Dec 2006,
http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm

9. Dan Gale, “System-On-Chip Research Infrastructure for Canadian Universities”, Canadian Microelectronics Corporation, Queen’s University, Kingston, Ontario, Canada, K7L 3N6.

10. Spartan-3E development board, Accessed 9 April 2007,
http://www.xilinx.com/bvdocs/userguides/ug230.pdf

APPENDIX A

List of Development Tools used

1. DXP, Altium Designer for PCB development.
2. Xilinx EDK, Platform Studio used for designing and developing SoC solutions.

Examples and tutorials for using the Xilinx EDK can be downloaded from the Xilinx website.

APPENDIX B

Compass SPI Hardware code Example

```
-------------------------------------------------------------------------------
-- opb_spi_0_wrapper.vhd
-------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

library opb_spi_v1_00_e;
use opb_spi_v1_00_e.All;

entity opb_spi_0_wrapper is
  port (
    SCK_I : in std_logic;
    SCK_O : out std_logic;
    SCK_T : out std_logic;
    MISO_I : in std_logic;
    MISO_O : out std_logic;
    MISO_T : out std_logic;
    MOSI_I : in std_logic;
    MOSI_O : out std_logic;
    MOSI_T : out std_logic;

    SPISEL : in std_logic;

    SS_I : in std_logic_vector(0 to 0);

    SS_O : out std_logic_vector(0 to 0);

    SS_T : out std_logic;

    OPB_Rst : in std_logic;

    IP2INTC_Irpt : out std_logic;

    Freeze : in std_logic;

    OPB_Clk : in std_logic;

    OPB_select : in std_logic;

    OPB_RNW : in std_logic;

    OPB_seqAddr : in std_logic;

    OPB_BE : in std_logic_vector(0 to 3);

    OPB_ABus : in std_logic_vector(0 to 31);

    OPB_DBus : in std_logic_vector(0 to 31);

    SPI_DBus : out std_logic_vector(0 to 31);

    SPI_xferAck : out std_logic;

    SPI_errAck : out std_logic;

    SPI_toutSup : out std_logic;

    SPI_retry : out std_logic

  );

  attribute x_core_info : STRING;

  attribute x_core_info of opb_spi_0_wrapper: entity is "opb_spi_v1_00_e";


end opb_spi_0_wrapper;
```

```
architecture STRUCTURE of opb_spi_0_wrapper is

 component opb_spi is
  generic (
    C_FIFO_EXIST : INTEGER;
    C_OPB_SCK_RATIO : INTEGER;
    C_SPI_SLAVE_ONLY : INTEGER;
    C_NUM_OFFCHIP_SS_BITS : INTEGER;
    C_NUM_SS_BITS : INTEGER;
    C_NUM_BITS_SR : INTEGER;
    C_DEV_BLK_ID : INTEGER;
    C_DEV_MIR_ENABLE : INTEGER;
    C_FAMILY : STRING;
    C_BASEADDR : std_logic_vector;
    C_HIGHADDR : std_logic_vector;
    C_INTERRUPT_PRESENT : INTEGER;
    C_OPB_AWIDTH : INTEGER;
    C_OPB_DWIDTH : INTEGER;
    C_IP_REG_BAR_OFFSET : std_logic_vector;
    C_DEPTH : INTEGER;
    C_NUM_BITS_REG : INTEGER;
    C_OCCUPANCY_NUM_BITS : INTEGER
  );
  port (
    SCK_I : in std_logic;
    SCK_O : out std_logic;
```

```
SCK_T : out std_logic;

MISO_I : in std_logic;

MISO_O : out std_logic;

MISO_T : out std_logic;

MOSI_I : in std_logic;

MOSI_O : out std_logic;

MOSI_T : out std_logic;

SPISEL : in std_logic;

SS_I : in std_logic_vector(0 to (C_NUM_SS_BITS-1));

SS_O : out std_logic_vector(0 to (C_NUM_SS_BITS-1));

SS_T : out std_logic;

OPB_Rst : in std_logic;

IP2INTC_Irpt : out std_logic;

Freeze : in std_logic;

OPB_Clk : in std_logic;

OPB_select : in std_logic;

OPB_RNW : in std_logic;

OPB_seqAddr : in std_logic;

OPB_BE : in std_logic_vector(0 to ((C_OPB_DWIDTH/8)-1));

OPB_ABus : in std_logic_vector(0 to (C_OPB_AWIDTH-1));

OPB_DBus : in std_logic_vector(0 to (C_OPB_DWIDTH-1));

SPI_DBus : out std_logic_vector(0 to (C_OPB_DWIDTH-1));

SPI_xferAck : out std_logic;

SPI_errAck : out std_logic;

SPI_toutSup : out std_logic;

SPI_retry : out std_logic
```

```
    );
  end component;

begin

  opb_spi_0 : opb_spi
    generic map (
      C_FIFO_EXIST => 0,
      C_OPB_SCK_RATIO => 128,
      C_SPI_SLAVE_ONLY => 0,
      C_NUM_OFFCHIP_SS_BITS => 0,
      C_NUM_SS_BITS => 1,
      C_NUM_BITS_SR => 8,
      C_DEV_BLK_ID => 4,
      C_DEV_MIR_ENABLE => 0,
      C_FAMILY => "spartan3e",
      C_BASEADDR => X"40a00000",
      C_HIGHADDR => X"40a0ffff",
      C_INTERRUPT_PRESENT => 1,
      C_OPB_AWIDTH => 32,
      C_OPB_DWIDTH => 32,
      C_IP_REG_BAR_OFFSET => X"00000060",
      C_DEPTH => 16,
      C_NUM_BITS_REG => 8,
      C_OCCUPANCY_NUM_BITS => 4
    )
```

```
port map (
  SCK_I => SCK_I,
  SCK_O => SCK_O,
  SCK_T => SCK_T,
  MISO_I => MISO_I,
  MISO_O => MISO_O,
  MISO_T => MISO_T,
  MOSI_I => MOSI_I,
  MOSI_O => MOSI_O,
  MOSI_T => MOSI_T,
  SPISEL => SPISEL,
  SS_I => SS_I,
  SS_O => SS_O,
  SS_T => SS_T,
  OPB_Rst => OPB_Rst,
  IP2INTC_Irpt => IP2INTC_Irpt,
  Freeze => Freeze,
  OPB_Clk => OPB_Clk,
  OPB_select => OPB_select,
  OPB_RNW => OPB_RNW,
  OPB_seqAddr => OPB_seqAddr,
  OPB_BE => OPB_BE,
  OPB_ABus => OPB_ABus,
  OPB_DBus => OPB_DBus,
  SPI_DBus => SPI_DBus,
  SPI_xferAck => SPI_xferAck,
      SPI_errAck => SPI_errAck,

      SPI_toutSup => SPI_toutSup,

      SPI_retry => SPI_retry

    );


end architecture STRUCTURE;
```

Compass UART Hardware code Example

```
-------------------------------------------------------------------------------
-- rs232_dce_wrapper.vhd
-------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

library opb_uartlite_v1_00_b;
use opb_uartlite_v1_00_b.All;

entity rs232_dce_wrapper is
  port (
    OPB_Clk : in std_logic;
    OPB_Rst : in std_logic;
    Interrupt : out std_logic;
    OPB_ABus : in std_logic_vector(0 to 31);
    OPB_BE : in std_logic_vector(0 to 3);
    OPB_RNW : in std_logic;
    OPB_select : in std_logic;
    OPB_seqAddr : in std_logic;
    OPB_DBus : in std_logic_vector(0 to 31);
    UART_DBus : out std_logic_vector(0 to 31);
    UART_errAck : out std_logic;
    UART_retry : out std_logic;
    UART_toutSup : out std_logic;
    UART_xferAck : out std_logic;
    RX : in std_logic;
    TX : out std_logic
  );
  attribute x_core_info : STRING;
  attribute x_core_info of rs232_dce_wrapper: entity is "opb_uartlite_v1_00_b";

end rs232_dce_wrapper;

architecture STRUCTURE of rs232_dce_wrapper is

  component opb_uartlite is
    generic (
      C_BASEADDR : std_logic_vector;
      C_HIGHADDR : std_logic_vector;
      C_OPB_DWIDTH : integer;
      C_OPB_AWIDTH : integer;
      C_DATA_BITS : integer;
      C_CLK_FREQ : integer;
      C_BAUDRATE : integer;
      C_USE_PARITY : integer;
      C_ODD_PARITY : integer
    );
    port (
      OPB_Clk : in std_logic;
      OPB_Rst : in std_logic;
      Interrupt : out std_logic;
      OPB_ABus : in std_logic_vector(0 to C_OPB_AWIDTH-1);
      OPB_BE : in std_logic_vector(0 to C_OPB_DWIDTH/8-1);
      OPB_RNW : in std_logic;
      OPB_select : in std_logic;
      OPB_seqAddr : in std_logic;
      OPB_DBus : in std_logic_vector(0 to C_OPB_DWIDTH-1);
      UART_DBus : out std_logic_vector(0 to C_OPB_DWIDTH-1);
      UART_errAck : out std_logic;
      UART_retry : out std_logic;
      UART_toutSup : out std_logic;
      UART_xferAck : out std_logic;
      RX : in std_logic;
      TX : out std_logic
    );
  end component;

begin

  rs232_dce : opb_uartlite
    generic map (
      C_BASEADDR => X"40600000",
      C_HIGHADDR => X"4060ffff",
      C_OPB_DWIDTH => 32,
      C_OPB_AWIDTH => 32,
      C_DATA_BITS => 8,
      C_CLK_FREQ => 50000000,
      C_BAUDRATE => 115200,
      C_USE_PARITY => 0,
      C_ODD_PARITY => 0
    )
    port map (
      OPB_Clk => OPB_Clk,
      OPB_Rst => OPB_Rst,
      Interrupt => Interrupt,
      OPB_ABus => OPB_ABus,
      OPB_BE => OPB_BE,
      OPB_RNW => OPB_RNW,
      OPB_select => OPB_select,
      OPB_seqAddr => OPB_seqAddr,
      OPB_DBus => OPB_DBus,
      UART_DBus => UART_DBus,
      UART_errAck => UART_errAck,
      UART_retry => UART_retry,
      UART_toutSup => UART_toutSup,
      UART_xferAck => UART_xferAck,
      RX => RX,
      TX => TX
    );
```

```
end architecture STRUCTURE;
```

Compass Software code Example

```
/******************************************************************


/* This is the list of files that must be included to access the peripherals:
 * xtmrctr.h - to access the timer
 * xgpiol.h - to access the general purpose I/O
 * xintc_l.h - access interrupt controller.
 * xparameters.h  - General purpose definitions.  Must always be included
 *        when any drivers/print routines are accessed.  This defines
 *        addresses of all peripherals, declares the interrupt service
 *        routines, declare STDIN/STDOUT devices etc.
 */

#include <xtmrctr.h>
#include <xintc_l.h>
#include <xgpio.h>
#include <xparameters.h>
#include "stdio.h"
#include "xio.h"
#include <xspi.h>
#include <xspi_l.h>
#include <mb_interface.h>

#define Rx_Empty  1
#define Rx_Full   2
```

```
#define Tx_Empty  4

#define LEDChan   1

/* Global variables: count is the count displayed using the
 * LEDs, and timer_count is the interrupt frequency.
 */
int one_second_flag = 0;
int i, j = 3, x = 0, y = 0, z = 0;
unsigned int timer_count = 1;
unsigned int xy = 0, Xbyte_count = 0, Ybyte_count = 0, spi_status, Xcount = 0, Ycount = 0;
Xuint16 Xsend_buf[2], Xreceive_buf[5], Xmin = 4096, Xmax = 0, Xout[10], Xmean = 0,
Xmin_max;
Xuint16 Ysend_buf[2], Yreceive_buf[5], Ymin = 4096, Ymax = 0, Yout[10], Ymean = 0,
Ymin_max;
XGpio gpio, leds;
XSpi compass_spi;
XTmrCtr timer1;

/*
 * Interrupt service routine for the timer.  It has been declared as an ISR in
 * the mss file using the attribute INT_HANDLER.  libgen automatically
 * registers it as the routine to be called when an interrupt occurs.  The exception
 * handler ensures that all registers are correctly saved, and that the return from
 * the interrupt occurs correctly.  The ISR can be written as a normal C routine.
 * The peripheral can be accessed using XPAR_<peripheral name in the mhs
file>_BASEADDR
 * as the base address.
 */

 //*****************************************************************
void timer_int_handler(void * baseaddr_p)
   {
   unsigned int csr;
         /* Read timer 0 CSR to see if it raised the interrupt */
         csr = XTmrCtr_mGetControlStatusReg(XPAR_DELAY_BASEADDR, 0);
         /* If the interrupt occured, then increment a counter */
         if (csr & XTC_CSR_INT_OCCURED_MASK)
         {
                  XGpio_DiscreteClear(&gpio,1,0x0001);
                  XGpio_DiscreteClear(&gpio,1,0x0001);
                  XGpio_DiscreteClear(&gpio,1,0x0001);
                  XGpio_DiscreteClear(&gpio,1,0x0001);
                  XGpio_DiscreteClear(&gpio,1,0x0001);
                  XGpio_DiscreteSet(&gpio,1,0x0001);
         }

         /* Clear the timer interrupt */
         XTmrCtr_mSetControlStatusReg(XPAR_DELAY_BASEADDR, 0, csr);

   }
 //*****************************************************************
void spi_int_handler(void * baseaddr_p)
   {


   }
//*****************************************************************
int main() {


  /* Registering the timer interrupt handler*/
  XIntc_RegisterHandler(XPAR_OPB_INTC_0_BASEADDR,
XPAR_OPB_INTC_0_DELAY_INTERRUPT_INTR,
                    (XInterruptHandler)timer_int_handler, (void
*)XPAR_DELAY_BASEADDR);


  /* Registering the spi interrupt handler*/
  XIntc_RegisterHandler(XPAR_OPB_INTC_0_BASEADDR,
XPAR_OPB_INTC_0_OPB_SPI_0_IP2INTC_IRPT_INTR,
                    (XInterruptHandler)spi_int_handler, (void
*)XPAR_OPB_SPI_0_BASEADDR);


  /* Initialize and set the direction of the GPIO connected to LEDs */
  XGpio_Initialize(&leds, XPAR_LEDS_8BIT_DEVICE_ID);
  XGpio_SetDataDirection(&leds,LEDChan, 0);


  /* Initialize and set the direction of the GPIO */
  XGpio_Initialize(&gpio, XPAR_GPIO_DEVICE_ID);
  XGpio_SetDataDirection(&gpio, 1, 0);


  /* Start the interrupt controller */
  XIntc_mMasterEnable(XPAR_OPB_INTC_0_BASEADDR);
  XIntc_mEnableIntr(XPAR_OPB_INTC_0_BASEADDR,
XPAR_DELAY_INTERRUPT_MASK | XPAR_OPB_SPI_0_IP2INTC_IRPT_MASK);


  /* Set the number of cycles the timer counts before interrupting */
  XTmrCtr_mSetLoadReg(XPAR_DELAY_BASEADDR, 0, 48000);


  /* Reset the timers, and clear interrupts */
  XTmrCtr_mSetControlStatusReg(XPAR_DELAY_BASEADDR, 0,
XTC_CSR_INT_OCCURED_MASK | XTC_CSR_LOAD_MASK );


  /* Enable timer interrupts in the interrupt controller */
  XIntc_mEnableIntr(XPAR_DELAY_BASEADDR,
XPAR_DELAY_INTERRUPT_MASK);


  /* Start the timers */
  XTmrCtr_mSetControlStatusReg(XPAR_DELAY_BASEADDR, 0,
XTC_CSR_ENABLE_TMR_MASK | XTC_CSR_ENABLE_INT_MASK |
                                        XTC_CSR_AUTO_RELOAD_MASK |
XTC_CSR_DOWN_COUNT_MASK);


  /* SPI Initialisation */
  XSpi_Initialize (&compass_spi, XPAR_OPB_SPI_0_DEVICE_ID);       //Initialise
  XIntc_mEnableIntr(XPAR_OPB_SPI_0_BASEADDR, 0x00000016);
          //Enable SPI intrerrupts
  //XSpi_Start(&compass_spi);
                    //Start Interrupts
  XSpi_mSetControlReg (XPAR_OPB_SPI_0_BASEADDR, 0x00000086);
  XSpi_mSetSlaveSelectReg (XPAR_OPB_SPI_0_BASEADDR,0x00000001);        //Disable
slave
  XSpi_mDisable (XPAR_OPB_SPI_0_BASEADDR);                                //Inhibit
master transfer

  /* Enable MB interrupts */
  microblaze_enable_interrupts();
  //***************************************************************

   while(1)
    {
          if(xy == 0)
          {

                    if(Xcount == 0)
                              Xsend_buf[0] = 0x00000001;
                    else if(Xcount == 1)
                              Xsend_buf[0] = 0x000000A0;

                    XSpi_mSetSlaveSelectReg (XPAR_OPB_SPI_0_BASEADDR,0x00000000);
             //Enable slave
```

```
                XSpi_mSendByte(XPAR_OPB_SPI_0_BASEADDR, Xsend_buf[0]);
            //Load Data to be sent
                for(i = 0; i < 250; i++);
                XSpi_mEnable (XPAR_OPB_SPI_0_BASEADDR);
                        //Enable master transfer
        for(i = 0; i < 15000; i++);
        Xcount = 1;

                Xbyte_count = Xbyte_count + 1;

        if(Xbyte_count == 1)
          Xreceive_buf[0] = XSpi_mRecvByte(XPAR_OPB_SPI_0_BASEADDR);

        else if(Xbyte_count == 2)
                Xreceive_buf[1] = XSpi_mRecvByte(XPAR_OPB_SPI_0_BASEADDR);

        else if(Xbyte_count == 3)
          {
           Xreceive_buf[2] = XSpi_mRecvByte(XPAR_OPB_SPI_0_BASEADDR);
                      Xbyte_count = 0;
                      Xcount = 0;
                      XSpi_mSetSlaveSelectReg
    (XPAR_OPB_SPI_0_BASEADDR,0x00000001);   //Disable slave

           Xmin_max = Xreceive_buf[1]<<8 | Xreceive_buf[2];
        if(Xmin_max > Xmax)
          Xmax = Xmin_max;
        if(Xmin_max < Xmin)
          Xmin = Xmin_max;


          Xmean = (Xmin + Xmax)/2;
                  Xout[z] = Xmean;
                  xy = 1;
                }
        }
//-------------------------------------------------------------------------------------------
                if(xy == 1)
                {


                for(i = 0; i < 100000; i++);
     if(Ycount == 0)
                        Ysend_buf[0] = 0x00000001;
                else if(Ycount == 1)
                        Ysend_buf[0] = 0x000000E0;


                XSpi_mSetSlaveSelectReg (XPAR_OPB_SPI_0_BASEADDR,0x00000000);
          //Enable slave
                XSpi_mSendByte(XPAR_OPB_SPI_0_BASEADDR, Ysend_buf[0]);
          //Load Data to be sent
                for(i = 0; i < 250; i++);
                XSpi_mEnable (XPAR_OPB_SPI_0_BASEADDR);
                                //Enable master transfer
        for(i = 0; i < 15000; i++);
        Ycount = 1;

                Ybyte_count = Ybyte_count + 1;

        if(Ybyte_count == 1)
          Yreceive_buf[0] = XSpi_mRecvByte(XPAR_OPB_SPI_0_BASEADDR);

        else if(Ybyte_count == 2)
                Yreceive_buf[1] = XSpi_mRecvByte(XPAR_OPB_SPI_0_BASEADDR);

        else if(Ybyte_count == 3)
          {
           Yreceive_buf[2] = XSpi_mRecvByte(XPAR_OPB_SPI_0_BASEADDR);
                        Ybyte_count = 0;
                        Ycount = 0;
                        XSpi_mSetSlaveSelectReg
    (XPAR_OPB_SPI_0_BASEADDR,0x00000001);   //Disable slave

           Ymin_max = Yreceive_buf[1]<<8 | Yreceive_buf[2];

           if(Ymin_max > Ymax)
                Ymax = Ymin_max;
           if(Ymin_max < Ymin)
```

```
                    Ymin = Ymin_max;


                Ymean = (Ymin + Ymax)/2;
                                Yout[z] = Ymean;
                                z = z + 1;
                                xy = 0;
                          }


                    }
//---------------------------------------------------------------------------------------------
        if(z == 10)
        {
            z = 0;
            x = Xout[0];
            y = Yout[0];
            while(z < 9)
            {
            z = z + 1;
            x = x + Xout[z];
            y = y + Yout[z];
            }

            Xmean = (x / 10);
            Ymean = (y / 10);
            /*Display Result*/
            XGpio_DiscreteWrite(&leds, LEDChan, Xmean);
            //xil_printf("Compass Xmin is %d\r\n",Xmin);

            //xil_printf("Compass Xmax is %d\r\n",Xmax);

            xil_printf("Compass Xmean = %d Ymean = %d\r\n",Xmean, Ymean);

            z = 0;

            Xmax = 0;

            Xmin = 4096;

            Ymax = 0;

            Ymin = 4096;

        }

            for(i = 0; i < 100000; i++);


              }

}
```