Bachelor's thesis

Information Technology

Internet Technology

2014

Sileshi Ziena

EMBEDDED SYSTEM DEVELOPMENT WITH PSOC

– Orientation sensing and visualization with PSoC

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Internet Technology

2014 | 37 pages

Instructor: Patric Granholm

Sileshi Ziena Adal

EMBEDDED SYSTEM DEVELOPMENT WITH PSOC

This thesis describes a new embedded system development process. The thesis aims to demonstrate how embedded systems could be developed efficiently with the help of the new technological advances such as programmable systems on chips. A special focus is given on HW and SW programming of such systems. The project makes use of a new generation of a chip called Programmable System on Chip (PSoC) as its hardware platform. What distinguishes PSoC from a line of processors and system on chips is its programmable hardware. This feature allows embedded system designers to be able to customize part of the hardware programmatically in addition to writing a software application that runs on top of the system. This thesis introduces the development of an embedded system based on the PSoc chip and development environment provided by Cypress semiconductors. Finally, this thesis presents a position sensing application which demonstrates the development process of a modern day typical embedded system.

KEYWORDS:

Embedded Systems, PSoC, Sensors, PCB, Orientation Sensing.

CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF ABBREVIATIONS (OR) SYMBOLS

ADC	Analog to Digital Converter
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
CAD	Computer Aided Design
CPU	Central Processing Unit, is a hardware in any computer system that carries out
FPGA	Field-programmable Gate Array, is an integrated circuit that can be programmed in the field after manufacture
IC	Integrated Circuit
IDE	In Development Environment
I/O	Input Output
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
MCU	Microcontroller Unit
PC	Personal Computer
PCB	Printed Circuit Board is a board that mechanically supports electronic components and provides electrical tracks between them.
PGA	Programmable Gain Amplifiers
PSoC	Programmable System on Chip, is a group of integrated circuits including CPU, configurable integrated analog and digital peripherals
RF	Radio Frequency
SRAM	Static Random Access Memory
SoC	System on chip
USB	Universal Serial Bus
V	Volts

1 INTRODUCTION

In general, embedded systems are electronic systems that are designed to perform a relatively specific task as compared to general-purpose computers (PCs). This also means that such systems are associated with some resource constraints such as memory and display size. Therefore, special attention and methodologies are required in designing and developing such systems. In this thesis we shall see this development process with a special concentration on efficient development of such systems using state-of-the-art approaches (Pont, 2007;2006).

Currently embedded system development is a huge market where less time to market period is demanded from system developers. In order to address this, different International companies have come up with different system development methodologies. Among these, Cypress Semiconductors have introduced a programmable system on chip (PSoC). PSoC is a chip where different, usually required, digital and analog embedded system components are put together (integrated) in to a single chip. Besides integration of different system components, PSoC delivers different system level APIs that enable dynamic configuration of these system level components. Clearly, this approach results in a very compact system where less interface circuits are required. It has also the advantage of letting the developer to concentrate on better system feature designs rather than dealing with fine tuning different interface circuits enabling fast time to market.

This thesis introduces the development of an embedded system based on the PSoC chip and development environment provided by Cypress semiconductors. Also, alongside this, I used the simple printed circuit board (PCB) that could support the chip and be used as a development kit.

As a demonstration of embedded system development and programming we have proposed and implemented an orientation sensor and visualizer that could be fixed to a moving/vibrating object to detect and show the exact orientation of a given object relative to a two dimensional axis. Besides this, we believe that we have explored and presented the future paradigm of embedded system design and development.

2 BACKGROUND

Among others, this work relates to areas such as

- Embedded system design
- Programmable system on chips and
- Orientation sensing

In the following sub sections we will discuss these areas in detail.

2.1 Embedded System Design

If we look around, we will find ourselves to be surrounded by computing systems. Every year millions of computing systems are built destined for desktop computers (Personal Computers, workstations, mainframes and servers) but surprisingly, billions of computing systems are built every year embedded within larger electronic devices and still goes unnoticed. Any device running on electric power either already has a computing system or will soon have a computing system embedded in it.

Embedded systems are computer systems where simple resource constrained chip is used to perform a specific task in real or near real-time. Embedded systems differ from the conventional PCs due to their constrained system resources such as memory, processor speed, and battery life and display size.

Today, Embedded systems are found in cell phones, digital cameras, camcorders, portable video games, calculators, and personal digital assistants, microwave ovens, answering machines, home security systems, washing machines, lighting systems, fax machines, copiers, printers and scanners, cash registers, alarm systems, automated teller machines, transmission control, cruise control, fuel injection, anti-lock brakes, active suspension and many other

devices. Simply stated, all computing systems other than general-purpose computers are embedded systems.

In general, Embedded System Design is a way of working, organizing or performing one or many tasks according to a fixed set of rules, program or plan. In other words, an arrangement in which all units assemble and work together according to a program or plan. In this thesis the program plan is an Orientation sensing and visualization system with PSoC.

2.2 Programmable System on Chip

Programmable system on chip is a family of integrated circuits made by Cypress Semiconductors. The chip includes a CPU and mixed-signal arrays of configurable integrated analog and digital peripherals.

In fact, a Programmable System on Chip definitions varies depending on its use. One definition is an FPGA (field-programmable gate array), which is an integrated circuit that can be programmed in the field after manufacture, and contains so many logic gates that can be rearranged (programmed) in to an entire system, possibly a Programmable System on Chip (PSoC). An FPGA can be considered as a programmable SoC if it includes enough gates to allow the inclusion of a microprocessor and other necessary components needed in design and implementation of an embedded system as in Figure 1.

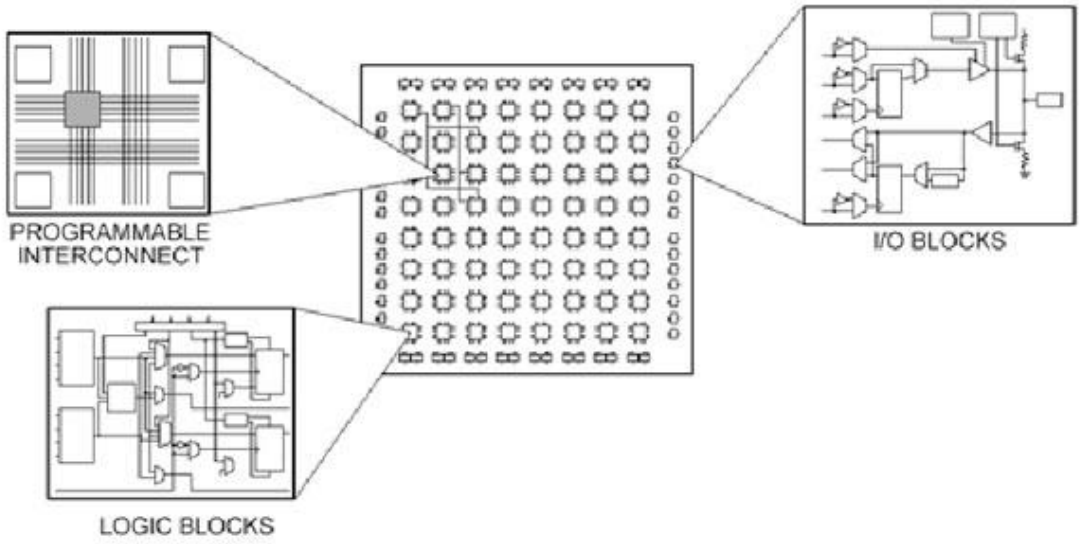

Figure 1. FPGA in programmable system on chip development (National Instruments, 2012).

Another type of programmable SoC is one in which a chip contains various fixed blocks, a microprocessor and peripherals that can be programmatically connected or disconnected. This differs from an FPGA-based programmable SoC in that the programmability is at a very high level of functionality and does not allow as much flexibility for low-level, user-defined functions to be designed on a chip. On the other hand, this family of PSoCs is easier to program.

Within the context of this thesis, a programmable SoC includes a microprocessor so that it is both hardware programmable and software programmable. There are two types of processors in modern PSoCs, hard processors and soft processors.

Hard Processor: A hard processor core is an FPGA with circuitry for a microprocessor embedded in the chip, surrounded by programmable logic. As shown in Figure 2, the processor in the lower right corner is fixed circuitry that takes the place of some programmable logic in an ordinary FPGA. The advantage of a hard processor is that the architecture is usually standard with support in terms of existing code, libraries, and tools such as compilers, debuggers, and operating systems from multiple vendors. Often each

programmable SoC vendor offers one hardware processor and its system requirements may limit the vendors and SoC families that can be used. Another advantage of a hard processor is that the processor circuitry is optimized for timing and power consumption and can be characterized by the vendor. (Zeidman, 2007).

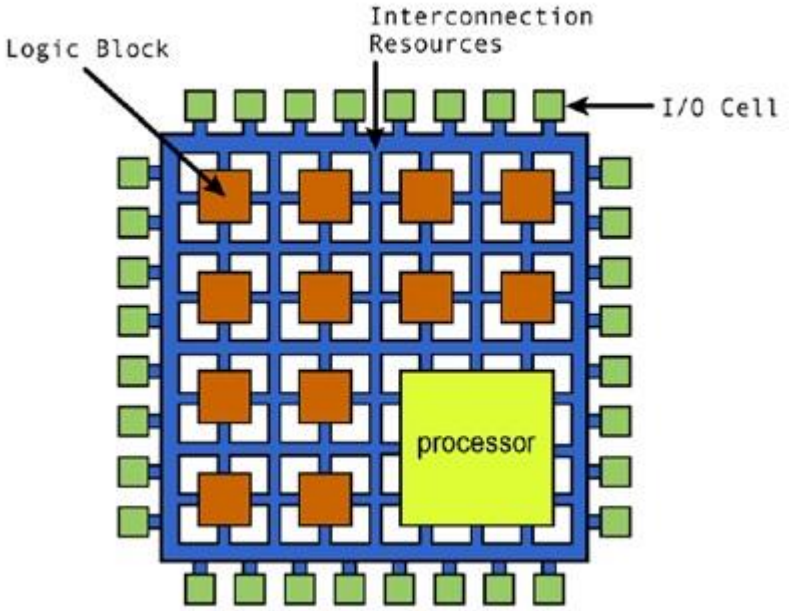

Figure 2. PSoC architecture with hard processor (Zeidman, 2007)

Figure 2 shows the general architecture of PSoC which consists of configurable logic blocks, configurable I/O blocks, and programmable interconnection with the hard processor.

Soft Processor: Any processor with sufficient logic resources can be a programmable SoC by including a soft processor. A soft processor advantage is that it can be configured optimally for a system and unnecessary functionality can often be removed, although significant changes may render the software compiler and other tools unusable (Zeidman, 2007).

To sum up, what makes this thesis different from previously developed similar projects is the use of Programmable System on Chip as our main base hardware platform. This means that most of the drivers and interface circuits needed between the analog sensors and the processor are found included in the chip and the system programmer/developer needs to route them to the ports and program them to take specific configuration tailored to the specifications of the system under development or even according to state of a running system. Figure 3 shows the general architecture of the PSoC with its three layers making up the entire chip.

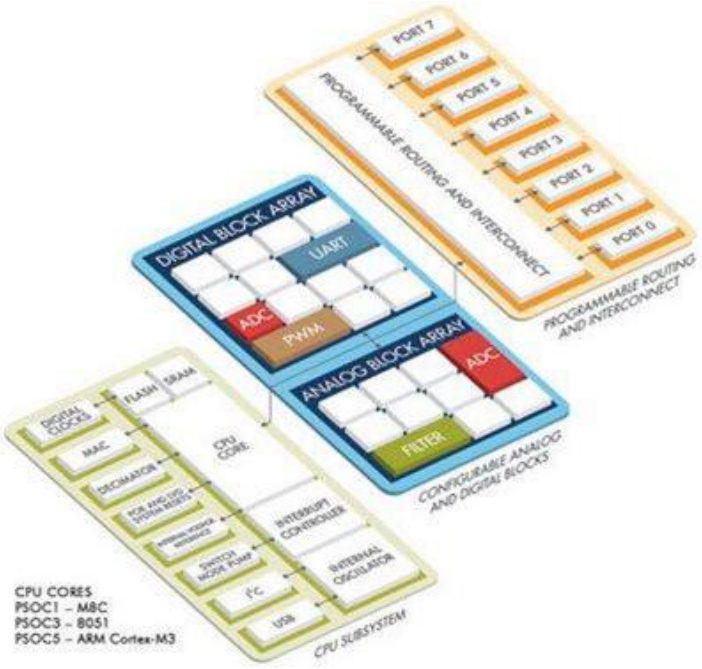

Figure 3. PSoC architecture and Its Layers (Cypress Semiconductors, 2013).

Figure 3 illustrates the layered architecture of PSoC, which is the basic concept behind its partially reconfigurable and programmable hardware. As can be noticed from the picture, the first layer contains its interconnection, the second layer contains analog and digital modules and the last layer contains CPU and its associated registers.

The interconnection layer of a PSoC replaces the conventional wires that exist between different components of an embedded system. Unlike the traditional interconnections, the PSoC interconnection can be re-arranged according to a particular embedded system platform specification, making it more fixable and reusable across an array of applications.

The digital block array layer found in PSoCs contains a set of predefined, useful, and configurable embedded components. These include ADCs, filters, and such usually necessary embedded system components.

The third layer of a PSoC contains the usually fixed components such as the micro controller unit along with its memory and other functional units.

Due to this layered architecture and re-configurability, the PSoC enables fixable system design and development.

2.3 Position Sensing

A three-axis accelerometer or three accelerometers arranged orthogonally can detect linear accelerations in three dimensions.

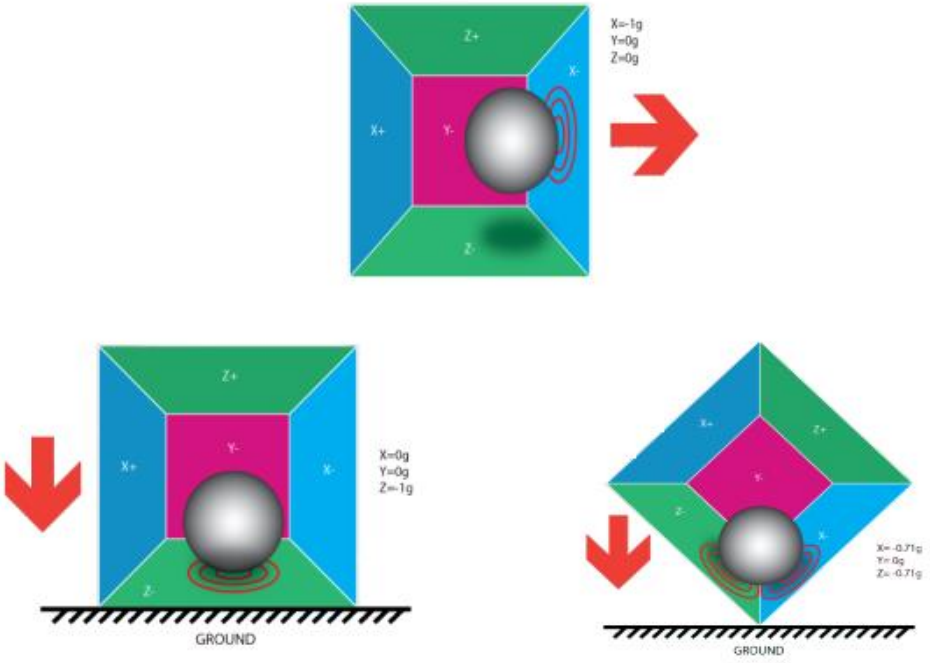

Figure 4. Basic concept of orientation sensing

In order to visualize this, one can imagine a ball inside a box with pressure sensitive walls. As we move the box in any given direction, the ball will press against the different walls, which tells us the direction of acceleration. If the accelerometer (sensor) is not moving, the ball will still push against one of the walls due to gravity. By comparing the readings (the pressing of the ball against the box walls) on the x, y and z-axis, we can work out the orientation of a stationary object. Figure 4 Illustrates how position (Orientation) sensing works.

Modern accelerometers, including the one used in this thesis are often the smallest Micro Electro-Mechanical *Systems (MEMS)*, consisting of little more than a beam anchored at one end with a proof mass (test mass used as a reference for measuring unknown quantity). Under the influence of external accelerations, the proof mass deflects from its neutral position. This deflection is measured in an analog or digital manner. Most commonly, the capacitance between a set of fixed beams and a set of beams attached to the proof mass is measured. This method is simple, reliable, and inexpensive.

3 MATERIALS AND METHODS

This section presents the materials and methods used in this thesis work.

3.1 Equipment Used

The different materials required in this thesis include:

- **PSoC designer.** This is a chip design and programming tool that one can use to customize a PSoC to meet one's own requirements. The PSoC designer has an integrated debug environment including in-circuit emulation and standard software debug features (see Figure 14 and Figure 16) (Cypress Semiconductors, 2013):
  - Extensive user module catalog
  - Integrated source code editor (C and Assembly)
  - Free C compiler
  - Application Editor GUI for device and user Module
  - Built-in Debugger
  - Built-in support for communication Interfaces
  - Integrated Circuit Emulation (ICE)

- **Microcontroller (CY8C27443-24PIX):** This is a Programmable System on Chip, manufactured by Cypress Semiconductors, which includes different analog and digital components such as amplifiers and DACs along side with microcontroller. Figure 5 depicts the PSoC and the PCB that supports it (Cypress Semiconductors, 2013).

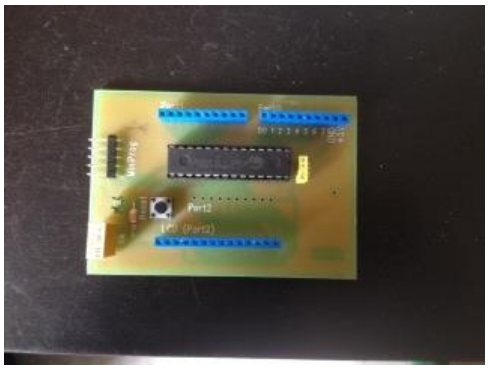

Figure 5. The PSoC Microcontroller

- **A Printed Circuit Board (PCB):** This is the board base for physically supporting and writing the surface-mounted and socketed components, in case of this thesis such as the screen, microcontroller, sensors and a USB-based programming tool (Menthor Graphics). Figure 6 depicts the PCB as viewed the bottom side. The top of the PCB can be seen in Figure 5.

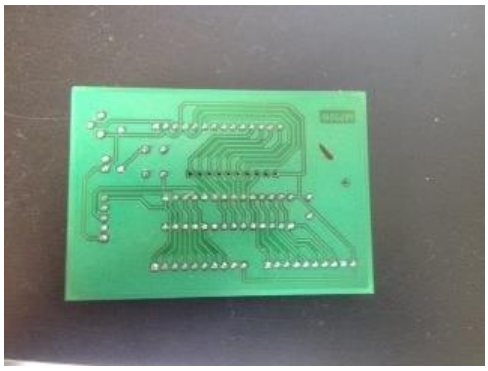

Figure 6. Bottom of the PCB

- **Power tip LCD module PC1602F B**: This is a type of display used in digital watches and many portable computers. In this thesis, the LCD screen is used to observe/display the digital or analog signals from the sensor. Figure 7 illustrates the LCD module used in this project (Powertip).

Figure 7. The LCD module

- **USB based programming tool**: This is used to transfer the Hex file produced by the compilers that is ready to run on the device (Cypress Semiconductors, 2013). Figure 8 illustrates the USB interface used to program the PSoC and transfer the final binaries.

Figure 8. USB based programming interface

- **Mentor Graphics tools**: These are a set of tools used to design a PCB that is able to support the chip (Menthor Graphics).

- **Position & accelerometer sensor**: This is used to measure or determine a physical position of the PSoC in order to sense and visualize its orientation relative to x and y-axis (vertical and horizontal position) (VTI Technologies, 2005). The sensor used in this work is the SCA610

series accelerometer. Figure 9 shows the pair of orthogonally placed accelerometers.

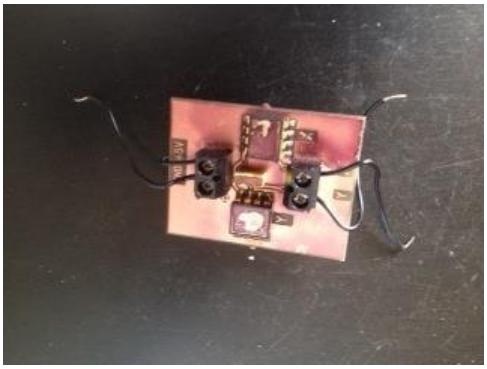

Figure 9. A pair of accelerometers used in the project.

3.2 Orientation Sensing and Visualization with PSoC

To illustrate the new way of embedded system development using PSoC, we will go through all phases of a sample application development, orientation sensing, and visualization.

Sensors are devices that convert one or more physical parameters into digital or analog signals which can be read by an observer or by an instrument. Such sensors, often referred to as transducers, convert physical parameters such as temperature, pressure, linear motion of objects (acceleration, velocity displacement, etc.), electrical and mechanical power, etc. into analog voltage/current or the digital equivalent.

Examples of sensors include

- **Position sensor** that is used to measure or determine a physical position.
- **Acceleration Sensor (accelerometer)** which measures the acceleration, applied to the device including the force of gravity.
- **Temperature sensor**: which are used on circuit boards, as part of thermal tests, in industrial controls, and in room controls such as in

calibration labs and data centers. There are many types of temperature sensors; most are passive devices: thermocouples, RTDs (resistance temperature detectors), and thermistors.

- **Gyroscope Sensor** which measures the rate or rotation in rad/s around a device's x, y and z axis. Usually its output is integrated over time to calculate a rotation describing the change of angles over the time step.
- **Geomagnetic field sensor** which is used to monitor changes in the earth's magnetic field.
- **Light sensors** which are mostly used on consumer products to determine the intensity of light; they are also used to control the brightness of a television or computer screen. Other uses include transmitting electrically charged signals and also detecting light and multiplying it.

Besides the sensors listed above there are other types of sensors which respond to an input quantity by generating a functionally related output in the form of an electrical or optical signal which can then be digitized, processed and visualized.

In this thesis, a pair of acceleration sensors (accelerometers) is used in order to sense and visualize the orientation of a circuit board or an object. As explained in Section 2.3, position sensors are useful for determining a device's physical orientation. The two accelerometers acquire the orientation of the circuit board and produce electric signals with a voltage proportional to the sensed orientation. These electrical signals are then digitized, processed, and displayed in a user friendly way. The next section presents the system overview in more detail.

3.2.1 System Overview

The main aim of this thesis is to explore the new paradigms of embedded system development and tools and demonstrate this with a sample system. In this endeavor we have chosen a position-sensing application using accelerometers. The general principle that we have followed is to sample two

analog accelerometers (inclinometers), filter this sampled inputs, calculate the respective inclination of the chip relative to a predefined normal position, and display this information with a user-friendly LCD interface.

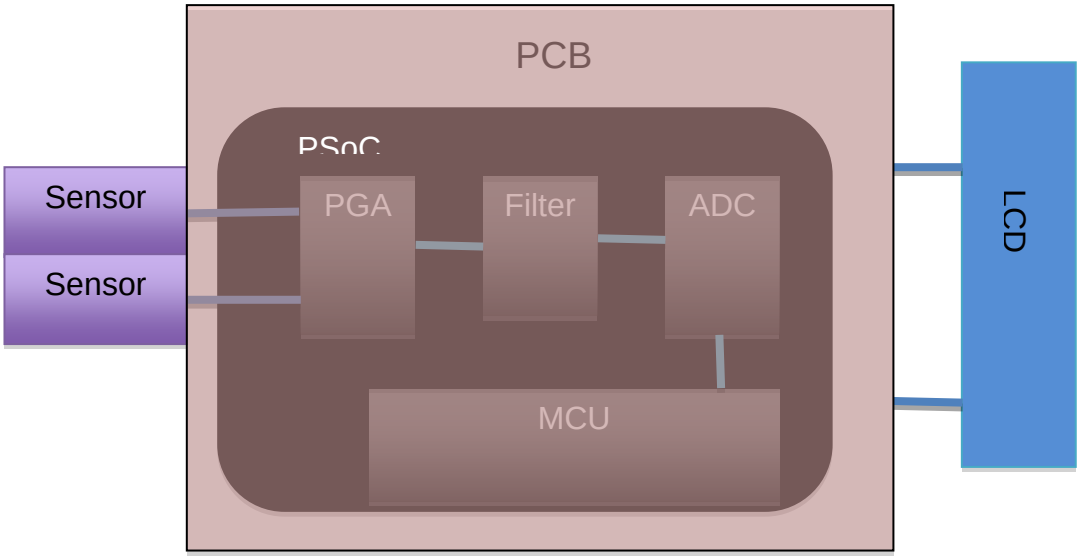

Figure 10. System overview

Figure 10 illustrates the general design of the proposed embedded system based on a programmable system on chip hardware platform. As can be noticed from the figure or architecture of the embedded system, the PSoC (middle black box) contains different components such as PGA, Filter, ADC, and MCU, which are partially routable and programmable using the Cypress designer tool (See Figure 14). This makes the entire embedded system development to be different from the traditional embedded system project development. The general use of components inside the PSoC and their explanation are as follows:

PGA: Programmable Gain Amplifiers is an electronic amplifier whose gain can be controlled by external analog/digital signals. A PGA is an integrated circuit (IC) that operates as a voltage amplifier and it has a differential input. That is, it has two inputs of opposite polarity. An operational amplifier has a single output

and a very high gain, which means that the output signal is much higher than the input signal. A programmable gain amplifier (typically operational amplifier) is often represented in a circuit diagram as in Figure 11.

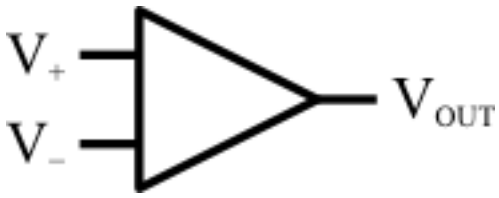

Figure 11. PGA Symbol

Figure 11 illustrates the symbolic form of a PGA. In this system we have used a pair of PGAs to amplify the signal coming form the accelerometers. Without such amplifications, the sensor outputs might become weak at times. These PGAs have been placed and routed on the PSoC as can be seen in Figure 14. Once the signals are amplified with the PGAs, they will then be connected to a filter for further processing.

Filter: A filter is an AC circuit that separates some frequencies from others within mixed-frequency signals. It is sometimes desirable to have circuits capable of selectively filtering one frequency or range of frequencies out of a mix of different frequencies in a circuit. A circuit that is designed to perform this frequency selection is called a Filter circuit, or just simply a filter.

In this thesis, another practical application of filter is presented where the amplified signal from the PGAs is smoothed out. This means that the sensed signal is cleaned from any form of noise that would affect the system precision. Once a clean signal is obtained, the system needs to change it to a digital form that can be manipulated by a micro controller unit (MCU). This conversion is done with an analog to digital convertor (ADC).

ADC: ADC stands for Analog to Digital Converter. Many electrical signals are analog in nature, which means a quantity varies directly from some other quantity. Let us assume voltage is the first quantity and pressure, temperature, force or acceleration can be the second quantity. For example in the Temperature Sensor the out voltage varies depending on the temperature, so if we can measure the voltage, then we can measure the temperature. However, most of the microcontrollers (or computers) are digital in nature. They can only differentiate or understand HIGH or LOW level signals. This means that we need to convert analog signals to digital so that they can be understood and processed by the microcontrollers. For instance, if the sensed signal voltage is more than 2.5 V, it will be read as 1 and if the sensed voltage is below 2.5 V then it will be read as 0. So to solve this problem, embedded systems have an ADC unit that will convert a voltage to a digital form so that it can be processed by microcontrollers. As can be seen in Figure 14, we have placed and routed a pair of ADCs for the purpose of adopting the signal from the sensors in to a digital form. Once the digital form of the signal is acquired, it can be presented to the MCU for further processing, in our case calculating which pixels to activate in the LCD display unit for visual presentation of the orientation information.

MCU can be considered as a small computer on a single integrated circuit (IC) containing a processor, memory and I/O peripherals. As the central unit of any embedded system, it is used to process inputs in to outputs. In this project, the MCU is used to calculate the pixel positions that need to be lit up in order to properly visualize the orientation of the object that the embedded system is attached to. This means that the MCU is responsible for computing the mathematical function that maps the sensed input signal to a pixel value.

3.2.2 PCB Design

In this thesis, we have used the custom PCB designed and developed at Metropolia University of Applied Sciences. The PCB is developed to support all

the components and interfaces of an embedded system. This PCB can also be used as a development kit for different projects. Figure 12 and Figure 13 show the top and bottom view of the PCB. The front side of the PCB contains the microcontroller at the center and the I/O ports at the corners. Note that the PCB does not contain a great deal of complicated interfacing circuitry like traditional embedded systems. These is because the PSoC contains several pre-fabricated blocks that can be placed, routed and used as different components needed in most of embedded systems such as ADCs and PGAs.

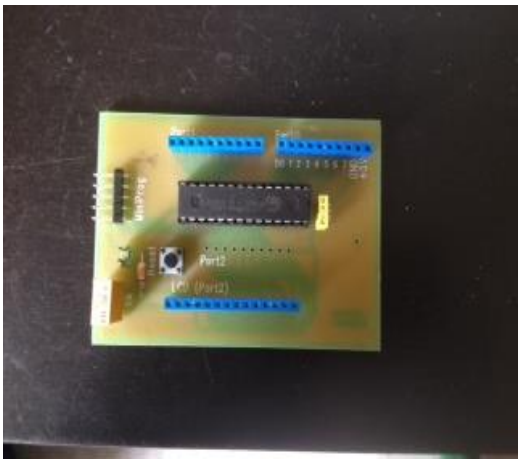

Figure 12. PCB top view

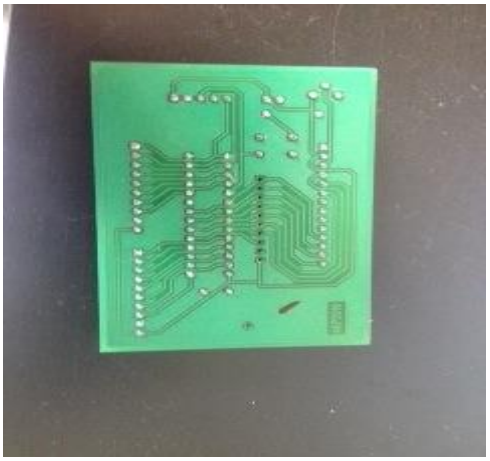

Figure 13. PCB bottom view

3.2.3 System Design

In the context of this thesis, system design means placing, configuring and routing of different analog and digital components that are required by the system. As the PSoC used in this project contains almost all of the necessary interfacing circuits inside the same chip, we had to learn and be able to place some required modules such as DACs, filters, timers, amplifiers and etc. on desired positions at the chip. In addition to this, each component is partially configured. Later on, we had to initialize and further configure these components programmatically. Figure 14 shows how some of the components are placed and routed on the PSoC designer's tool window.

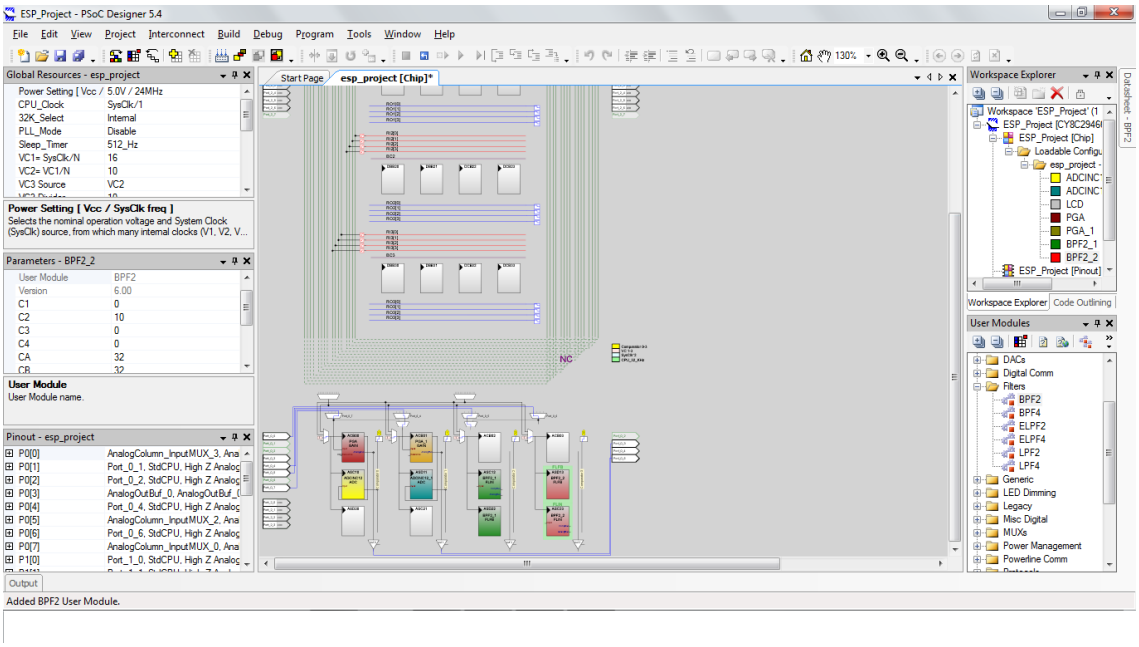

Figure 14. PSoC place and route using cypress PSoC designer.

As shown in Figure 14, a pair of ADCs, PGAs and filters are placed and routed. Once all the necessary components are placed and routed, they are partially configured as shown in Figure 15. These configurations can be changed or further fine-tuned in the application layer via API calls.

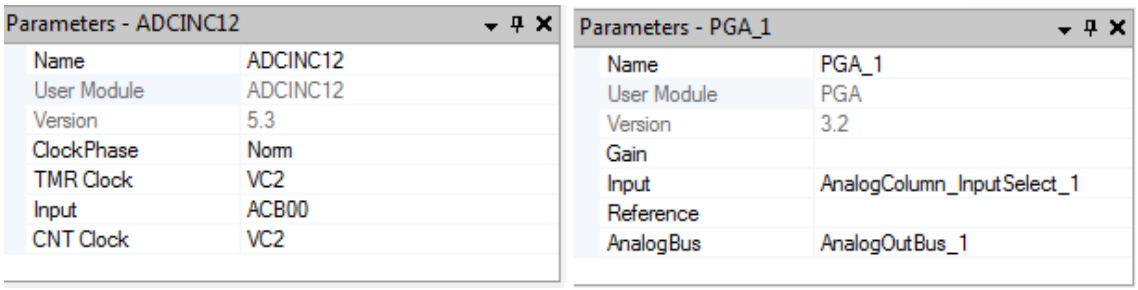

Figure 15. PSoC component configuration

3.2.4 System Programming

System programming is the phase where different components that were placed in the previous step are initialized and configured. In addition, the whole application logic is implemented at this stage. For our specific case, sensor inputs are taken and routed through the ports, amplifiers and filters before being sampled by ADCs and digitally manipulated to give rise to the orientation of the chip in terms of degrees which is displayed as inclined set of bar graphs on an LCD screen.

The following code shows the main application logic that is designed following a super loop approach of embedded programming. A while loop that runs forever is used to encapsulate the logic of our application.

```
// Function Prototypes
void initateAll(void);
void rd_calc_tilt(void);
void disp_tilt(void);
void disp_small_tilt(void);
void main(void){
        // Function starts LCD, PGA; ADC and TIMER and
        enable global interupts
        initateAll();
        // loop- forever
        while(1){
                rd_calc_tilt();
                disp_tilt();
                disp_small_tilt();
```

```
            }
}
```

Before the start of the while loop a function that is used to properly initialize all the required components is called. The following code listing shows parts of this initialization code. This initialization code is also responsible for starting all the components needed by the system.

```
// starts LCD,ADCand PGA, enables global inturapts
// Set the ADC to Sample continously
// Initiate LCD in bar graph mode
void initateAll(void){
          gDeg.fDegX = 0.0;
          gDeg.fDegY = 0.0;
          M8C_EnableGInt ; // Turn on interrupts
          M8C_EnableIntMask(INT_MSK0 , INT_MSK0_GPIO);
          //Enable GPIO-interrupts
          PGA_SetGain(PGA_G1_00);
          PGA_Start(PGA_MEDPOWER);
          PGA_1_SetGain(PGA_G1_00);
          PGA_1_Start(PGA_MEDPOWER);
          // Apply power to the SC Block
          ADCINC12_1_Start(ADCINC12_HIGHPOWER);
          // Have ADC run continuously
          ADCINC12_1_GetSamples(0);
          // Apply power to the SC Block
          ADCINC12_Start(ADCINC12_HIGHPOWER);
          // Have ADC run continuously
          ADCINC12_GetSamples(0);
          //turn on lcd
          LCD_Start();
}
```

Looking at our implementation further, we find the function implementations of the three methods that are called from inside the super loop. The following listing shows one of these function implementations. Thus, a particular function is responsible for continuously reading and calculating the tilt (orientation) on each axis.

```
void rd_calc_tilt(void){
          // Loop until value ready
          while(ADCINC12_fIsDataAvailable() == 0);
          // Clear ADC flag
          ADCINC12_ClearFlag();
          // Loop until value ready
          while(ADCINC12_1_fIsDataAvailable() == 0);
          ADCINC12_1_ClearFlag(); // Clear ADC flag
          //sample in x direction
          gData.iDataX=ADCINC12_iGetData();
          // sample in y
          gData.iDataY=ADCINC12_1_iGetData();
          //calc degree of tilt in X
          gDeg.fDegX =(float)(gData.iDataX*SCALE);
          //calc degree of tilt in Y
          gDeg.fDegY =(float)(gData.iDataY*SCALE);
}
```

Furthermore, the disp_tilt() and disp_small_tilt() functions as can be seen in the following listing, take the calculated tilt (orientation) and send a signal to pixels of the LCD screen which draws the appropriate tilt in a user friendly way.

```
void disp_tilt(void){


    // calculate and display tilt of one dimenssion Y!
    if(gDeg.fDegY>=0){
         LCD_InitVBG();
    if(gDeg.fDegY<=90 ){
         LCD_DrawVBG(1, 13, 2,
         (int)((gDeg.fDegY+SHIFTY)/SCALEY));
    }else if (gDeg.fDegY>90)
         //if degree is > 90,   incase, use maximum scale
         LCD_DrawVBG(1, 13, 2, 16);
    }

    if(gDeg.fDegY<=0){
         LCD_InitVBG();
    if(gDeg.fDegY>=-90){
         LCD_DrawVBG(1, 13, 2,
         (int)((gDeg.fDegY+SHIFTY)/SCALEY));
    }else if (gDeg.fDegY<-90)
```

```
            //if degree is < -90,    incase, use min. scale
            LCD_DrawVBG(1, 13, 2, 0);   }

}
```

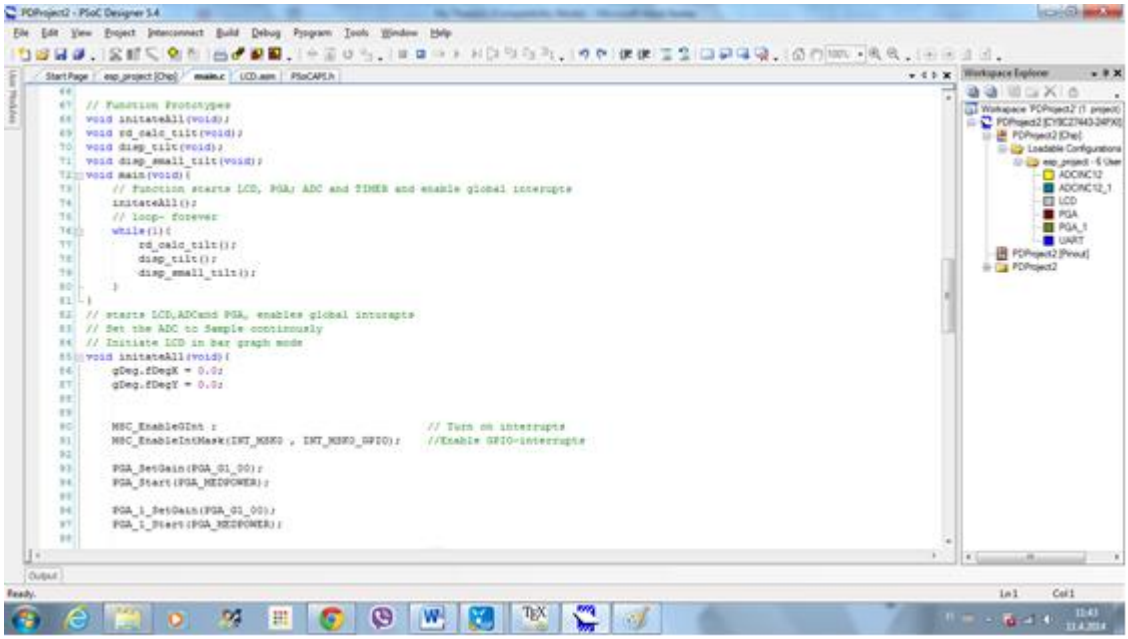

Figure 16: Programming environment

Finally, Figure 16 shows the programming environment (IDE) provided by the Cypress designer tool. One can switch between the design and route tool and the programming environment.

3.2.5 Results: Compilation and Deployment

Compilation and deployment of the system was the last part of the project. At this phase, a USB-based devices programming tool is used to upload net lists (hardware configuration files) and hex files (application executable). During this stage, some testing and modifications have also been made.

Once the placement, routing, configuring and programming of components of the PSoC are implemented, it is time to generate and build the components on the PSoC. This component generation process also generates an appropriate

set of APIs for the corresponding components so that they can be programmed from the application layer.

The next stage is to compile and build the software part that will run on top of the generated hardware. The APIs of each generated component can be used to configure, initialize, run, and stop each hardware component from inside our programs.

Finally, the application binary is uploaded via a USB connection and the system is ready to be used.

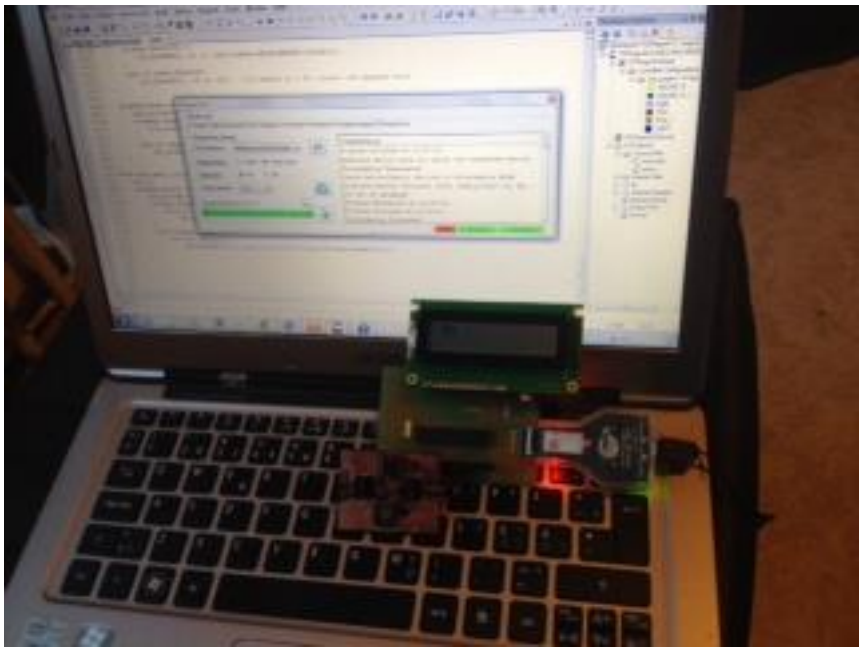

Figure 17. Overall system while running

Figure 17 illustrates the overall running system.

Our system should be capable of following the changes of the orientation of the object it is attached to regardless of which direction the change is made; hysteresis is the measure of this property. The output calculated at different orientations are measured and presented in Table 1.

Table 1. Performance test of the system

Orientation (Calculated)	ADC Output	X (V)	Y (V)
-90	-819	1.5	1.5
-60	-546	1.0	1.0
-45	-410	0.75	0.75
-30	-273	0.5	0.5
0	0	2.5	2.5
30	273	1.16	1.16
45	410	1.75	1.75
60	546	2.33	2.33
90	819	3.5	3.5

Figures 18-20 show the tests at different orientation. In all figures the first inclined bar shows the tilt (orientation) of the board along the x axis. The second vertical bar shows the tilt (orientation) in y axis.

Figure 18. Test 1

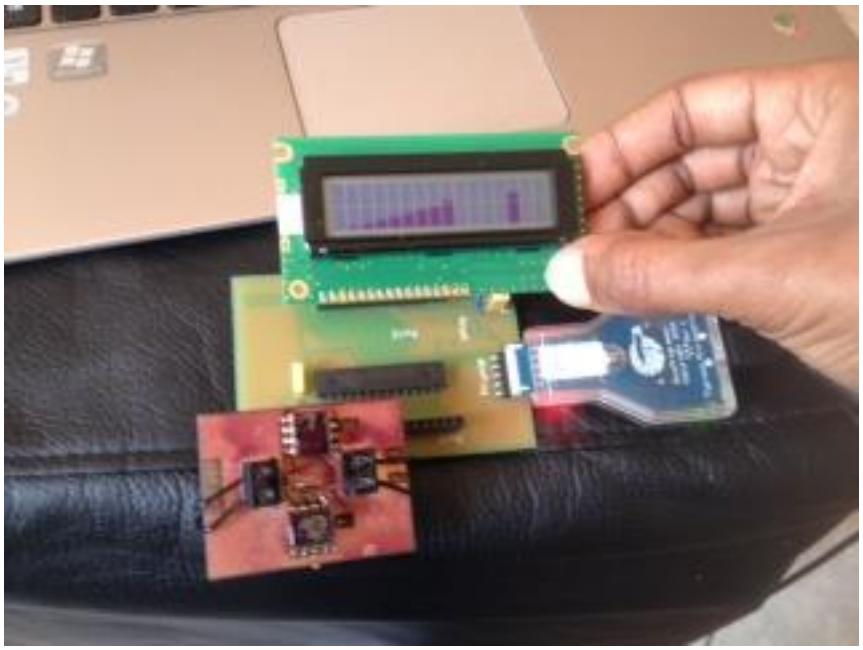

Figure 19. Test 2

Figure 20. Test 3

4 CONCLUSION

This thesis presented an overview of the advances made in the process of embedded system development. Due to the introduction of Programmable System on Chips, the development of embedded systems has changed significantly. Instead of building a system from a processor and a set of specifically designed input and output interfacing circuitry (i.e., ADC, Filter and etc.) along with their derivers, we can now use a Programmable System on Chip to easily design, customize and generate a piece of hardware with all the interfacing components and their drivers. This hardware (system on chip) can then be easily programmed. When one needs to change or redesign this embedded system, the PSoC can be reprogrammed with new components in to a new hardware making it more flexible and cost effective.

In this thesis work, firstly we designed the system components using PSoC Designer tool (Cypress Semiconductors, 2013). This includes the placing and routing of different components, such as the ADC filter, which are required by the system.

Secondly, we programmed the application logic, a position sensing application, which demonstrates the development process of a modern day typical embedded system.

REFERENCES

Balarin, F. P. (1997). *Hardware-software co-design of embedded systems: the POLIS approach.* Kluwer Academic.

Barr, M. (29. Nov 2007). *Netrino Technical Library*. Retrieved 20. April 2014 from http://www.netrino.com/Embedded-Systems/Glossary.

Cypress Semiconductors. (17. July 2013). *CY8C27443 PSoC Datasheet*. Retrieved 20. April 2014 From http://www.cypress.com/?docID=45148

Cypress Semiconductors. (17. July 2013). *Software Solutions from Cypress*. Retrieved 20. April 2014 from http://www.cypress.com/?rID=39531&source=header

Pont, J.M. (2006). *Programming Embedded system I*. Retrieved 20. April 2014 from http://www.le.ac.uk/eg/mjp9/pes1ohp_a4.pdf

Pont, J.M. (2007). *Programming Embedded System II*. Retrieved 20. April 2014 from http://www.le.ac.uk/eg/mjp9/pes2ohp_a4.pdf

Menthor Graphics. (1981). *PCB design tools and manuals*. Retrieved 20. April 2014 from http://www.mentor.com/products/pcb-system-design/

Montana State University. (11. Feb 2006). *Guide to Mentor Graphics PADS 2005* . Retrieved 20. April 2014 from http://www.coe.montana.edu/ee/lameres/courses/eele461_spring12/information/Guide_to_PADS.pdf

National Instruments. (03. May 2012). *FPGA Fundamentals*. Retrived 20. April 2014 from http://www.ni.com/white-paper/6983/en/

Powertip. *PC 1602-F LCD Module Datasheet*. Retrieved 20. April 2014 from http://www.powertipusa.com/pdf/pc1602f.pdf

VTI Technologies. (30. Dec 2005). *SCA610 Accelerometer Sensor Datasheet*. Retrieved 20. April 2014 from http://www.muratamems.fi/sites/default/files/uploads/sca610_accelerometer_rev_3.pdf

Zeidman, B. (27. July 2007). *Introduction to Programmable Systems on a Chip*. Retrived 20. April 2014 from http://www.eetimes.com/document.asp?doc_id=1274457

Appendix A: Source Code

```
.
//-------------------------------------------------------------------------
--------
// C main line
//-------------------------------------------------------------------------
--------

/*
*
*/

/*
*This lab is about "Oreantation sensor": througth the Use two-axis
acceleration
*sensor to measure lateral movement of table.A PGA is also used to
amplify
*the signal such that we start seeing very small movements of the
table.
*LCD module, in bar-graph mode is used to  show acceleration

*variation (=variance or absolute difference from the mean value) in
both x- and y-direction.
*
*Two acceleration sensors(X, Y) are used to measure tilt in both
Directions(SCA610 series tilt sensors, VTI.fi)
*Vdd = 5v
*Vout = 1.5 - 3.5(corresponding to -90 to 90 deg)
*Voffset = 2.5v (voltage out of sensor @ 0 degree )
*
*
*LCD in bar graphmode is used to display tilt intensities in X and Y
*2 PGAs
*2 ADCs to continously sample the analog sensor input in both
directions
*ADCs
*Vin = 0-5V (corresponding to -2048 to 2047 which is equal to 2^12
integer samples)
*Vref = 2.5v (coressponds to 0 integer sample value)
*Notice that we only use the range from (-819-819) since the voltage
out of the sensor is
*constrainted to 1.5-3.5v. thus aproprate scaling and shifting are
applied
*
*Clk input to ADCs is 150k=(Sysclk/16*10)
*
*/
#include <math.h>
#include <stdlib.h>                         // to enable integer
display instade of byte
#include <m8c.h>                                        // part specific
constants and macros
#include "PSoCAPI.h"                                    // PSoC API
definitions for all User Modules
```

```
#include "PSoCGPIOINT.h"                                    // library
header for user interupts
//#include "PWM8.h"                                                 //
include the PWM8 API header file
//#include "TIMER8.h"                                               //
include the Counter8 API header file

#define SCALE 90.0/819                                        //voltage is
sampled in to an int value b/n -819(1.5v) to 819(3.5v)

                                    //need to be scaled to 0-90 degree
#define SCALEX 15/90                                          //we use
bargraph mode on lcd to display tilt intensity in x, 15 horizontal
pixels are used

                                    //to display degree in 0-90 or -90-0
range thus a multiplying scale (15/90) is used
#define SCALEY 11.25                                          //we use
bargraph mode on lcd to display tilt intensity in y  16 vertical
pixels are used

                                    //to express 180 degree(-90 to 90)
therefore (180/16= 11.25) is used as a scaling factor
#define SHIFTY 90
          //used to shift -90 to 90 in to 0 to 180 scale to ease
drawing a bar graph 0 to 16
#define PI 3.14

typedef struct {
          float fDegX;
          //structure to store degree of tilt in X and Y
          float fDegY;

}Deg;

typedef struct {
          INT iDataX;
                    //stracture to  store sampled data from ADC for
the 2 directions
          INT iDataY;
}Data;

Deg  gDeg;
          //global datatype to store degree of tilt in X and Y
Data gData;
                    //global datatype to store data sampled by adc in
X and Y
char gTmp[40]="Esp!";
char gStr[] = "ON";

// Function Prototypes
void initateAll(void);
void rd_calc_tilt(void);
void disp_tilt(void);
void disp_small_tilt(void);
void main(void){
          // Function starts LCD, PGA; ADC and TIMER and enable
global interupts
          initateAll();
```

```
            // loop- forever
            while(1){
                        rd_calc_tilt();
                        disp_tilt();
                        disp_small_tilt();
            }
}
// starts LCD,ADCand PGA, enables global inturapts
// Set the ADC to Sample continously
// Initiate LCD in bar graph mode
void initateAll(void){
            gDeg.fDegX = 0.0;
            gDeg.fDegY = 0.0;



            M8C_EnableGInt ;                                                    //
Turn on interrupts
            M8C_EnableIntMask(INT_MSK0 , INT_MSK0_GPIO);
            //Enable GPIO-interrupts

            PGA_SetGain(PGA_G1_00);
            PGA_Start(PGA_MEDPOWER);

            PGA_1_SetGain(PGA_G1_00);
            PGA_1_Start(PGA_MEDPOWER);

            ADCINC12_1_Start(ADCINC12_HIGHPOWER);
            // Apply power to the SC Block
            ADCINC12_1_GetSamples(0);
                                    // Have ADC run continuously

            ADCINC12_Start(ADCINC12_HIGHPOWER);
            // Apply power to the SC Block
            ADCINC12_GetSamples(0);
                                    // Have ADC run continuously

            LCD_Start();
                                    //turn on lcd

            LCD_Position(0,1); // Place LCD cursor at row 0, col 5.
            LCD_PrString(gStr); // Print "PSoC LCD" on the LCD

}

//Loop until values(samples) are ready from the the two ADCs
responsible for sampling the tilt in X and Y direction
//When values found clear flags
//and store the values(samples) in a global structure, gData
//The ADCINC12 User Module implements a 12-bit incremental A/D that
generates a 12-bit, full-scale 2's
//complement output (+2047 to -2048 count range) corresponding to 0-5v
input and 2.5v ref  .
//thus 2 ADCs are used to sample voltage input from the sensors in the
X and Y direction.
//Note that sampled values could only range from -819 to 819
coressponding to 1.5-3.5v input margins that
//could be obtained from the accleration sensors(SCA610 by VTI.fi)
corresponding to -90 to 90 degree tilt
```

```
void rd_calc_tilt(void){

           while(ADCINC12_fIsDataAvailable() == 0);                //
Loop until value ready
                     ADCINC12_ClearFlag();
                                                 // Clear ADC flag
           while(ADCINC12_1_fIsDataAvailable() == 0);              //
Loop until value ready
                     ADCINC12_1_ClearFlag();
                               // Clear ADC flag
           gData.iDataX=ADCINC12_iGetData();
                     //sample in x direction
           gData.iDataY=ADCINC12_1_iGetData();              // sample
in y

           gDeg.fDegX =(float)(gData.iDataX*SCALE);
           //calc degree of tilt in X
           gDeg.fDegY =(float)(gData.iDataY*SCALE);
           //calc degree of tilt in Y

}

//To disply tilt in X we need to transform the sampe obtained from the
ADC to tilt range of 0-90 or -90-0 and
//plot it in either of left or right bargraph lots on the lcd.
//special note need to be taken here!., since the Vout of
//the sensor is 1.5-3.5 we will not be using the full range(i.e. -
2048-2047 ) but instade we are only be using
//a smaller range of -819-819 thus we need to scale 819 in to  a 90
degree range through
//a multiplying factor of 90/819( defind as SCALE) and diplay it in
either of the two x bar graph components
//according to its sign(left bargraph component(cells 5-8) for -ve
degrees and right bar graph components(cells 11-13)for +ve degrees )
//
//similarlly a central vertical bargraph component(at column 9 ) is
used to show the tilt intensity in the y direction
//approprate shfting and scaling is used to fit it in to a 2 cell 16
pixle graph.S

void disp_tilt(void){


// calculate and display tilt of one dimenssion Y!
                     if(gDeg.fDegY>=0){
                     LCD_InitVBG();
                     if(gDeg.fDegY<=90 ){
                               LCD_DrawVBG(1, 13, 2,
(int)((gDeg.fDegY+SHIFTY)/SCALEY));
                     }
                     else if (gDeg.fDegY>90)
                               LCD_DrawVBG(1, 13, 2, 16);   //if
degree is > 90, incase, use maximum scale
           }

           if(gDeg.fDegY<=0){
                     LCD_InitVBG();
                     if(gDeg.fDegY>=-90){
```

```
                                    LCD_DrawVBG(1, 13, 2,
(int)((gDeg.fDegY+SHIFTY)/SCALEY));
                        }
                        else if (gDeg.fDegY<-90)
                                    LCD_DrawVBG(1, 13, 2, 0);        //if
degree is < -90, incase, use min. scale
            }
}
void disp_small_tilt(void){
            int i;
            if(gDeg.fDegX>=-20 && gDeg.fDegX<=20){
                        LCD_InitVBG();
                        if(gDeg.fDegX>=0){
                                    for(i=2; i<=9; i++)
                                                LCD_DrawVBG(1,(i-
1),2,(int)((i-1)*5*tan(2*PI*gDeg.fDegX/360)));
                        }
                        else if(gDeg.fDegX<0){
                                    for(i=9; i>=2;i--)
                                                LCD_DrawVBG(1,i,2,(int)((-
i+10)*5*-1*tan(2*PI*gDeg.fDegX/360)));
                        }
            }
}
```