REDUCING CHIP COUNT WITH A PROGRAMMABLE SYSTEM ON CHIP IN PERSONNEL DETECTION USING SIGNAL SCAVENGING

A Thesis

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

MENGXUAN MA

Dr. Harry W. Tyrer, Thesis Supervisor

December 2015

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

REDUCING CHIP COUNT WITH A PROGRAMMABLE SYSTEM ON CHIP IN PERSONNEL DETECTION USING SIGNAL SCAVENGING

presented by **Mengxuan Ma**,

a candidate for the degree of **Master of Science**,

and hereby certify that, in their opinion, it is worthy of acceptance.

Professor Harry W. Tyrer

Professor Marjorie Skubic

Professor Mihail Popescu

ACKNOWLEDGEMENTS

I would like to express my appreciation to the persons who help me with this project. I would like to express my sincere gratitude to my supervisor Dr. Tyrer for that he gives me such a fantastic topic, so that I can have a chance to implement some applications on this advanced system on chip. I am extremely grateful for his assistance and suggestions throughout my project and thesis. I would also like to thank my committee, Dr. Skubic and Dr. Popescu.

Also, I would like to thank Hancheng Wu, who inspired me, supported me and helped me out with some issues in this project. I have to thank my entire lab colleague, who helped me conduct some experiments. I would like to give special thanks to Chinchao Suriyakul who gave me lots of valuable suggestions and Zhenduo Zhai who assisted me with many of the experiments.

Finally, I would like to convey my heartfelt gratitude to my family. Without their support, it is no possible for me to study here and chase my dream.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Our laboratory has implemented a personnel detection system, Smart Carpet, which can detect the walking of older adults. The existing system consists of traditional individual components and chips, leading to a large system size, which is inconvenient.

To reduce the number of chips and the size of the system, we have implemented a fully functional hardware system for the Smart Carpet using Programmable System on Chip (PSoC). Using PSoC's functional units, we meet the requirement of smart carpet functionality. Using software, the program can be loaded to PSoC chip to control each component on the chip using in the system. The system implemented by a PSoC has been successfully tested with the PSoC development board. Based on the functional units we need, we have designed the necessary peripherals circuitry for the chip, which I have placed onto a Printed Circuit Board (PCB). The I^2C communication with PSoC has been applied to achieve multiple boards working cooperatively. The new design can handle 80 sensors. It enjoys smaller size and scalable with carpet size.

The reduction in the total number of IC chips is from 17 to 1 and components from 98 to 56 and the reduction of the system size from about 26.3 square inches to 8.7 square inches.

CHAPTER 1 INTRODUCTION

Electronic devices have been changing the quality of people’s daily lives for decades. Recently, many efforts have been paid to how to utilize emerging technologies to bring elderly adults safe and high quality lives. Falls are the leading cause of injury for the elderly adults. To revolutionize fall prevention, our laboratory has implemented an in-home personnel fall detection sensor system, Smart Carpet. The system works as following: sensors are installed under the carpet and connected to operational amplifiers. Amplified sensor signals are fed to and selected by multiplexers. A differential amplifier amplifies the selected signal and an ADC converts the amplified signal to digital data. The microprocessor then performs an ASCII conversion on the digital data, and feeds the converted data to the serial port of a PC for further processing and data display. In the existing system, all of the functional units are implemented by traditional individual chips, leading to big system size. As a result, with individual components, it is not easy to maintain such a system, and the bulk size makes the implementation not scalable.

In this work, we use the Programmable System on Chip (PSoC) to reduce the system size because it incorporate the hardware functionality and make the system scalable. On the hardware side, we have implemented a fully functional hardware system for Smart Carpet using PSoCs provided by Cypress Semiconductor Company. It includes a routing scheme to map PSoC's functional units to the previous hardware, and we tested them on the PSoC development board. Based on the functional units needed, we have designed the power supply system, the necessary peripherals for the PSoC chip, and have drawn up the Printed Circuit Board (PCB) to integrate these components. On the software side, we have written the code to drive all the functional units and implemented the I2C communication between multiple PSoC boards. The latter makes multiple boards work cooperatively; the whole system can then scale up as the Carpet size increases. This new design has been tested successfully with 64 sensors. It reduces the total number of IC chips from 17 to 1 and components from 98 to 56, and cuts the system size from about 26.3 square inches to 8.7 square inches, compared to the traditional system.

CHAPTER 2 BACKGROUND

2.1 Fall Detection in Eldercare Technology

Falls are the leading cause of injury in persons over the age of 65 years and can be the primary etiology of accidently deaths among seniors. The mortality rate for falls increases dramatically with age accounting for 70 percent of accidental deaths in persons 75 years of age and older [1]. Older adults living alone are at great risk of delayed assistance following a fall [2]. An in-home sensor system which can continuously detect falls of elder adults could revolutionize fall prevention and care [3].

Many methods have been investigated for detecting falls for elder adults; these include the following two general categories: wearable and positional aware methods.

Wearable devices allow an individual to manually push a button in the event of fall, and wearable devices that automatically detect a fall using sensors are applied. For example, some researchers use accelerometers and tilt sensors to automatically detect a fall event [4-6]. However, wearable devices must be continuously worn, require batteries to be recharged, and may simply be forgotten by the user. In the case of devices requiring action on the part of the wearer, loss of consciousness following a fall would prevent use.

Positional awareness overcomes some of the aforementioned drawbacks. A number of researchers use some environmentally mounted sensors for fall detection, such as floor vibration sensors [4, 7], passive infrared sensors [8], acoustic sensors [7, 9], and video-based sensors, including traditional cameras [10] and depth imaging sensors [11].

2.2 Smart Carpet System

We proposed a low cost fall detection system, Smart Carpet, using floor sensor array as a mean to detect personnel. The system uses signal scavenging technique to detect presence of the person [12-15].

Signal scavenging uses a sensor made from a conductive material to pick up stray 60 Hz noise which may come from electrical power lines, nearby electrical equipment and other stray electromagnetic signals. Although this strong 60 Hz signal seems to be useless energy, it has been shown to be a valuable signal source to detect the presence of humans and also of human falls [16, 17].

The system shown in Figure 2-1 consists of sensors installed under the carpet to detect a person activities, electronic boards to read the data from the sensors, then a computer to process the data from the sensors and send notifications when fall occurs.

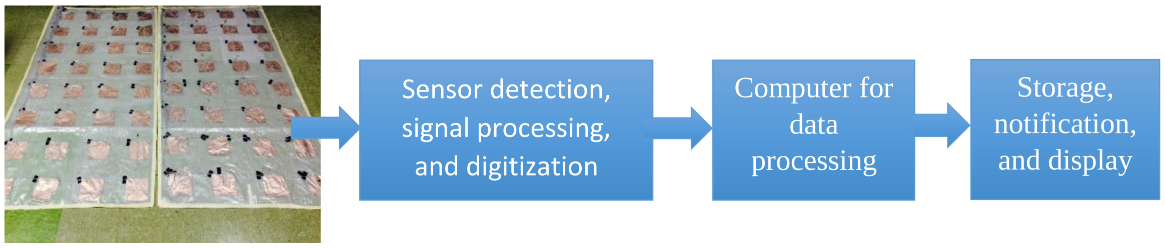

Sensor detection, signal processing, and digitization
Computer for data processing
Storage, notification, and display

Figure 2-1 The function block of Smart Carpet system

One installs the sensors for the signal scavenging technique under carpets; it consists of 2 main layers as shown in Figure 2-1. The top layer is the sensor layer, and the lower layer is the ground layer. Both sensors and grounding are separated and covered by a plastic sheet. We have used electrical conductors including copper or aluminum. An experiment [18] shows that both copper and aluminum have an output impedance of 10 MΩ.

In the previous system, the sensor pads under the carpet are connected to operational amplifiers and then selected by multiplexers controlled by the microprocessor. A differential amplifier amplifies the selected signal and an ADC converts the amplified signal to digital data. The microprocessor then performs an ASCII conversion on the digital data, and feeds the converted data to the serial port of a PC for further processing and data display. All of the functional units above are implemented by traditional individual chips, leading to big system size. As a result, it is not easy to maintain such system, and the bulk size make the implementation not scalable.

2.3 The PSoC Advantage

We used the PSoC® 5LP chip [19] from Cypress Semiconductor Inc. It is a programmable embedded system-on-chip, integrating configurable analog and digital peripherals, memory, and a microcontroller on a single chip. It has 32-bit ARM Cortex-M3 core plus DMA controller and digital filter processor, at up to 80 MHz. Its programmable digital and analog peripherals enable custom functions. It provides flexible routing of any analog or digital peripheral function to any pin. These feature makes it possible to reduce the hardware components using PSoC chip.

The software and hardware side of PSoC has been investigated to map PSoC's functional units to meet the requirement of smart carpet hardware.

Cypress Semiconductor Inc., which designs and sells PSoC 5lp chips, also provides a free integrated design environment named PSoC Creator for PSoC 5lp. PSoC Creator enables concurrent hardware and firmware editing, compiling and debugging of PSoC 5LP systems with no code size limitations. PSoC peripherals are designed using schematic capture and simple graphical user interface (GUI) with over 120 pre-verified, production-ready PSoC Components™ [20].

PSoC® MiniProg3 is an all-in-one programmer for PSoC 1, PSoC 3, PSoC 4, and PSoC 5LP architectures, a debug tool for PSoC 3, PSoC 4, and PSoC 5LP architectures, and an USB-I2C Bridge for debugging I2C serial connections and communicating with PSoC devices [21]. We use PSoC® MiniProg3 to program the PSoC 5lp Chip and load the system program to the PSoC chip on our design boards.

To convert the analog signal to digital signals, the ADC_DelSig module [22] in PSoC Chips has been used, which provides a low-power, low-noise front end for precision measurement applications. An internal low-pass filter is used to filter out the noise outside the desired input signal bandwidth. This makes delta-sigma converters good for both high-speed medium resolution (8 to 16 bits) applications, and low-speed high-resolution (16 to 20 bits) applications. The sample rate can be adjusted between 10 and 384000 samples per second, depending on mode and resolution.

To support multi-board communication, we use I^2C communication module in PSoC chips for sending and receiving data. The I^2C component allows networking multiple devices on a single board or small system. The system can be designed with a single master and multiple slaves, multiple masters, or a combination of masters and slaves. As shown in Figure 2-2, the I^2C bus requires external pull-up resistors. The pull-up resistors (RP) are primarily determined by the supply voltage, bus speed, and bus capacitance. Pull-up resistors typical value is from 2 kΩ to 10 kΩ [23].

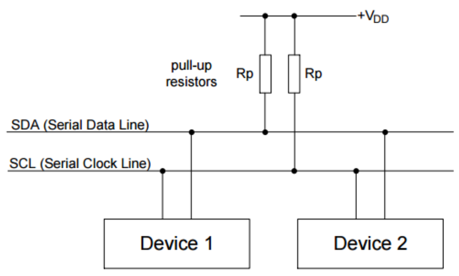


Figure 2-2 the principle of connection of devices to I2C bus. Pull-up resistors Rp typical value is from 2 kΩ to 10 kΩ.

The CY8CKIT-050 PSoC® 5LP Development Kit enables one to evaluate, develop and prototype high precision analog, low-power and low-voltage applications powered by Cypress’s CY8C58LP high precision analog device family. We initially tested our designed system on The CY8CKIT-050 PSoC® 5LP Development Kit. After that, we designed and manufactured our PSoC 5lp based Printed Circuit Board. We used this manufacture chips because of this resources provided to support the design. Other manufactures are just now coming up with similar systems, but they were not available when we started this project.

We used CadSoft EAGLE PCB design software [30] to design our 4-layer PSoC PCB boards. In this software, one can easily design the schematic, create the board from the schematic, place the components, and route the signal and produce CAM files. We chose Advanced Circuit, the Printed Circuit Board manufacturing and assembly company to manufacture our PCB board.

CHAPTER 3   METHODS

Our goal is to integrate the devices that constitute the hardware of the Smart Carpet, a sensor system for detecting personnel, in which a PSoC 5LP chip replaces many chips in the predecessor design. The main job of the hardware part is to input the sensor signals through I/O ports. One sensor signal is selected to be amplified, sampled and processed at a time. The microprocessor produces the multiplexer control signals and provides the ASCII conversion to feed the communication port of a PC for further processing and data display. So we take a signal from one of many sensors and convert to useful data. In this chapter, we first discuss how we implemented each main hardware module on the purchased PSoC 5LP development kit, and how we connected them into the whole system. Then, we describe our first Printable Circuit Board using only three IC chips, we show how we designed and constructed our second boards to add multi-board communication function to improve performance.

Cypress Semiconductor Inc., which designs and sells PSoC 5lp chips, also provides a free integrated design environment named PSoC Creator for PSoC 5lp. One can create a project, draw the circuit by adding components from the library, configure the properties of components, and write the programs to control each component. After creating the project, PSoC Creator compiles the project to a .hex file to program and configure the by PSoC. PSoC Creator is the necessary tool to create the application project, towrite the control programs and to generate the executable file for PSoC chip.

We summarize below the procedures for implementation of a PSoC 5lp based application on PSoC Creator, we implement each applications following these five procedures.

1. Draw the schematic
2. Configure the modules and set up GPIO pins.
3. Write program to control modules.
4. Compile the project and check the interconnection of PSoC 5LP chip.
5. Load the program to the PSoC 5lp based boards [24].

3.1 Preliminary Data

3.1.1 Signal Input and Output

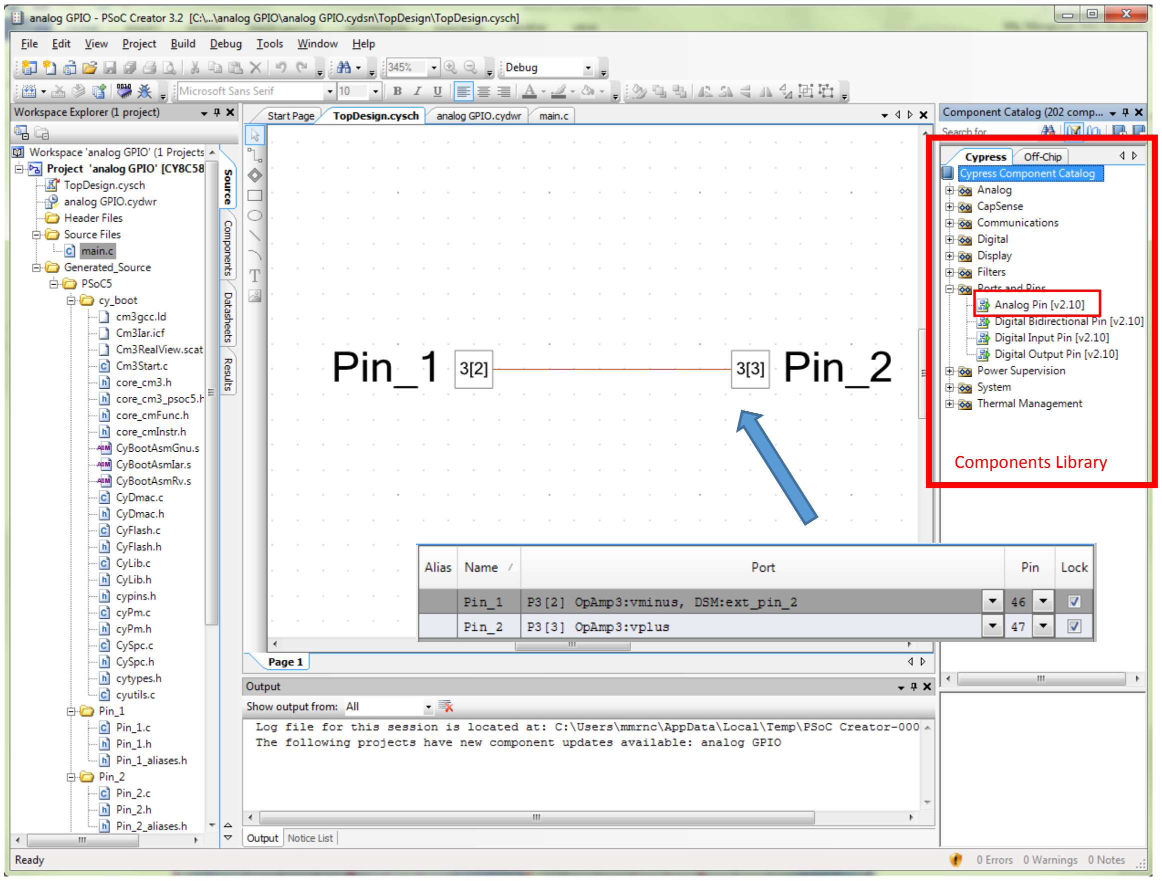
General purpose input/outputs (GPIOs) provide the entry for sensor signals to the Programmable System on Chip (PSoC), which can be considered to initiate the sensor detection system. One important feature is that GPIOs are programmable, and Cypress provides a set of APIs to dynamically control GPIOs. PSoC Creator provides resources to facilitate this.

We create a project in PSoC Creator, build the GPIO test circuit where an input GPIO pin is connected to an output GPIO pin directly. This demonstrates that a signal can go through the PSoC 5lp chip, and confirms the programmable feature. The test compares the signals on both input pin and output pins.

We created the GPIO test circuitry schematic in PSoC Creator Top Design window. We add two analog pins from ‘ports and pins’ category of components library (shown on the right of Figure 3-1) and connected these two pins together.

Then we set the hardware GPIO ports to the pins we added as follows. Importantly, GPIO pins are required to assign to the hardware GPIO pins on PSoC chip. There are about 6 sets of GPIO groups are available in the PSoC Creator in .cydwr window, which are P0[0]-P0[7], P2[0]-P2[7], P3[0]-P3[7], P4[0]-P4[7], P5[0]-P5[7], P6[0]-P6[7] and P1[2],P1[4],P1[5],P1[6],P[7]. Here input GPIO Pin_1 is assigned to P3[2] and the output GPIO pin Pin_2 is assigned to P3[3]. No extra configuration is needed for these two pins. Also, since only two pins needs to be physically connected, no control program is needed. In short assigning the GPIO groups to the pins sets the GPIO ports to the hardware.

Next, we compiled this project. When compilation is finished, a .hex file can be found in the project folder. To check the interconnection of the PSoC 5lp chip, we can open the analog function in .cydwr window as shown in Figure 3-2 below. It is obvious that only two GPIO pins are selected and connected together. After successfully compiling the program, we can load the program. Hex file into the chip. The PSoC 5lp development kit provides the programming interface. One programming method is to connect the usb port of kit to computer. After the usb connection is set up, the program can be downloaded by clicking the program bottom on Creator. The PSoC 5lp kit will execute the program as long as it is powered.



Alias	Name	Port	Pin	Lock
	Pin_1	P3[2] OpAmp3:vminus, DSM:ext_pin_2	46	✓
	Pin_2	P3[3] OpAmp3:vplus	47	✓

```
Log file for this session is located at: C:\Users\mmrnc\AppData\Local\Temp\PSoC Creator-000
The following projects have new component updates available: analog GPIO
```

Figure 3-1 Circuit design and pin configuration of GPIO Creator project. Two analog pins(Pin_1 and Pin_2) from components library are connected to each other. Pin_1 is configured to GPIO port P3[2] and Pin_2 is configured to GPIO port P3[3].

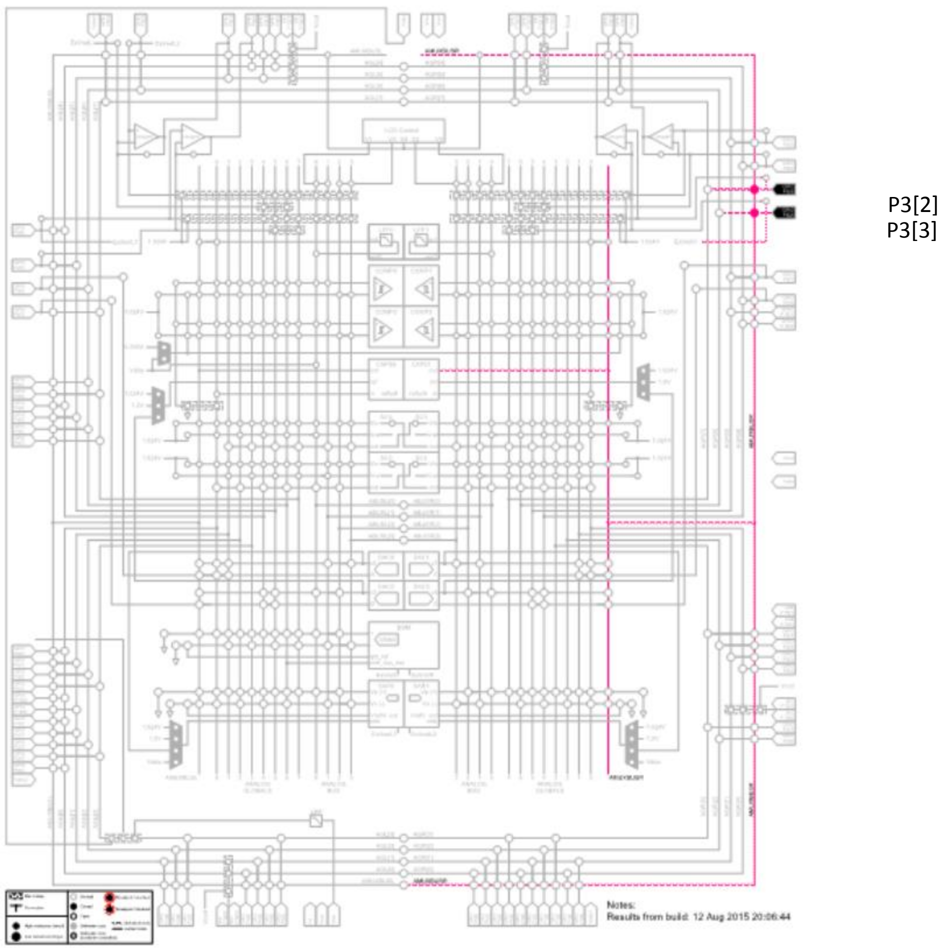


Figure 3-2 The interconnection of PSoC 5LP chip after GPIO project has been compiled. The highlighted route shows GPIO port P3[2] and P3[3] are connected to each other, which matches the design in schematic.

To test the GPIO project, we can use a sensor signal as input to the PSoC Pin_1 and observe the signals on both input pin and output pin by oscilloscopes shown in Figure 3-3.

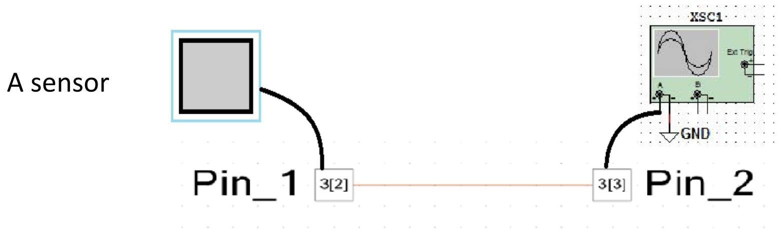
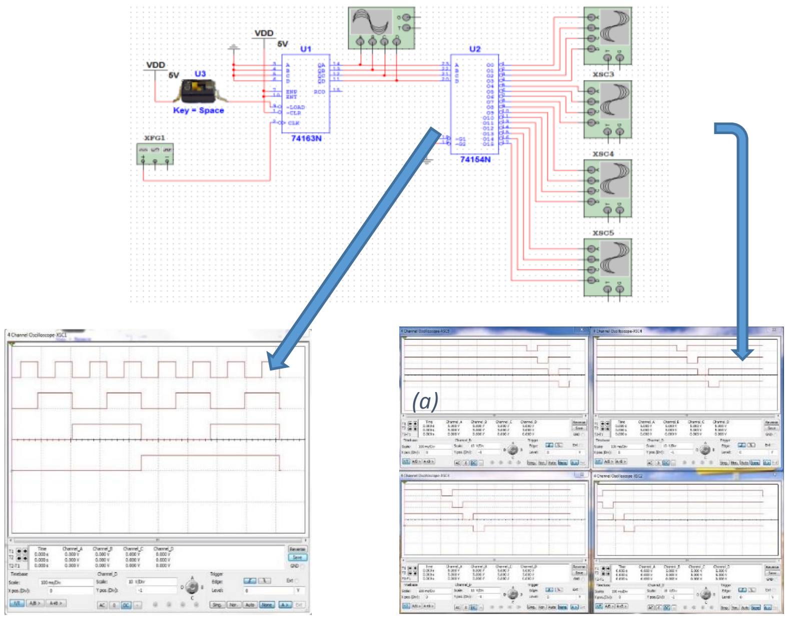


Figure 3-3 To test the GPIO performance, we connected a sensor to the input pin Pin_1 and connected an oscilloscope to observe the output.

3.1.2 Sensor Addressing Demonstration

We've studied a single GPIO pin's programmable feature, now to handle multiple sensors in the scavenging system, we need to test the PSoC 5LP chip performance for acquiring large numbers of signals from multiple GPIOs.

We designed the circuit shown in Figure 3-4(a) to provide 16 output signals. A 74163 chip counts a clock signal and stores it as a 4-bit number, which will then be demultiplexed to a 16-bit signal by a 74154 chip. One output will be selected by a low voltage signal at a time. So only 1 bit is 0 and the rest are 1. We simulated the circuit in Multisim, circuit simulation software. The simulation waveform of the outputs of 74163 and 74154 are shown in Figure 3-4. Figure 3-4(b) shows the output of the 74163 counter, and Figure 3-4 (c) shows the output of the 74154 demultiplexer.



(b) (c)

Figure 3-4 shows the circuit and Multisim simulation of generating addressing signals for PSoC chip. A clock signal is counted into 4 bit number by the chip 74163 shown from the output of the chip and from the outputs of chip 74154, one output will be selected by a low voltage signal at a time.

This signal can be regarded as a sensor address to select GPIO pins on the PSoC 5LP chip. To show the performance of GPIO pins, we created a project on PSoC Creator. In schematic shown in Figure 3-5, we placed 16 digital input pins for connecting to the outputs from 74154. These 16 input pins are connected to 16 digital output pins whose outputs can be programmed to appear on an LCD. The compiler configures the GPIO ports randomly on the PSoC, unless specified, as we have done here.

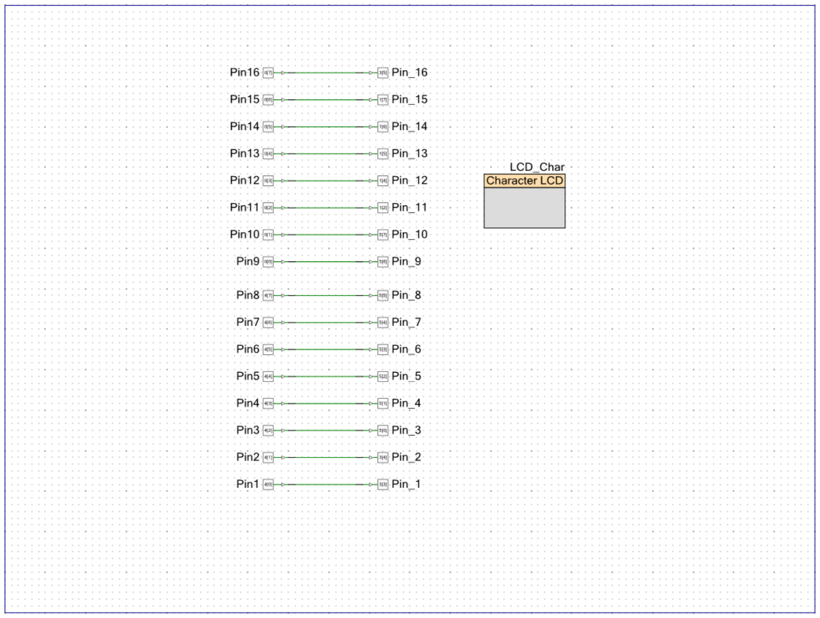


Figure 3-5 Circuit design and pin configuration of multiple GPIO pins Creator project. 16 digital inputs are connected to 16 digital output pins. All these pins are configured to GPIO pins randomly. 16 input GPIO pins collect the signals from the chip 74154. Data received from 16 output pins are sent to LCD for display.

We wrote a program to control the microprocessor in the PSoC as well as output the data to the LCD. During the data collection and display process, the microprocessor reads and stores input signals and send to LCD. The pseudo code is shown in Figure 3-6.

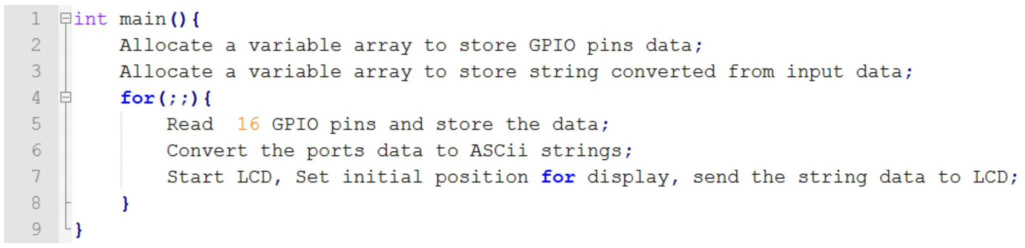

```
int main(){
    Allocate a variable array to store GPIO pins data;
    Allocate a variable array to store string converted from input data;
    for(;;){
        Read  16 GPIO pins and store the data;
        Convert the ports data to ASCii strings;
        Start LCD, Set initial position for display, send the string data to LCD;
    }
}
```

Figure 3-6 pseudo code of multiple GPIO pins project.

3.2 Four Sensor Test System

3.2.1 Test of Analog GPIO Pins

To test the main modules on PSoC 5lp chip, we produced a 4 sensors array. GPIOs are the entries to the PSoC system. So we connect an output of one sensor to an input GPIO. In order to observe the behavior of GPIO pins, we connect the input GPIO to an output GPIO and connect an output GPIO to an oscilloscope. The block diagram of the four sensor carpet GPIO system is shows in Figure 3-7.

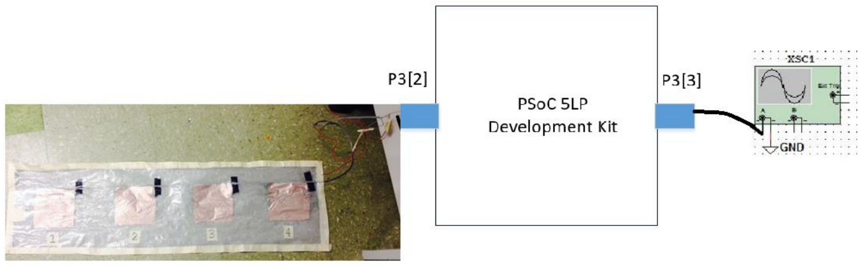


Figure 3-7 Four sensor carpet system. One sensor from 4-sensor array is connected to GPIO input pin P3[2] and an oscilloscope is connected to GPIO output pin P3[3].

We loaded the program designed and discussed in 3.1.1, and programed GPIO pin P3[2]to be input pin and P3[3] to be the output pin.

3.2.2 Implementation of an Analog Multiplexer Module

The PSoC 5LP chip contains one ADC module and 4 Programmable Gain Amplifiers (PGAs). To process all the signals from sensors, we require signal multiplexing: only one GPIO's output signal can be selected to send to the next module. We choose the GPIO pins, and we use the internal analog multiplexer on the resources. A multiplexer selects one of several input signals and forwards it to the output. The GPIO pins are connected to analog resources, or to each other, through a series of analog routing buses joined by switches and AMUXs. The two primary analog routing buses are the analog global (AG) bus and the analog mux (AMUX) bus [25]. The analog routing of the PSoC's upper left quadrant is shown in Figure 3-8.

The PSoC does not have a recognizable multiplexer device. It implements an analog multiplexer (AMUX) rapidly switching several input pins to analog routing buses. The PSoC creator provides a routine called AMUX_FastSelect (channels) to provide inputs to perform the multiplexing function.

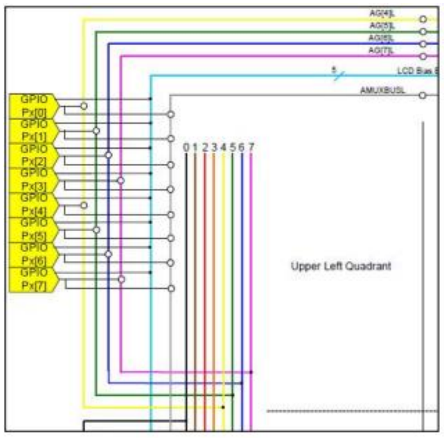
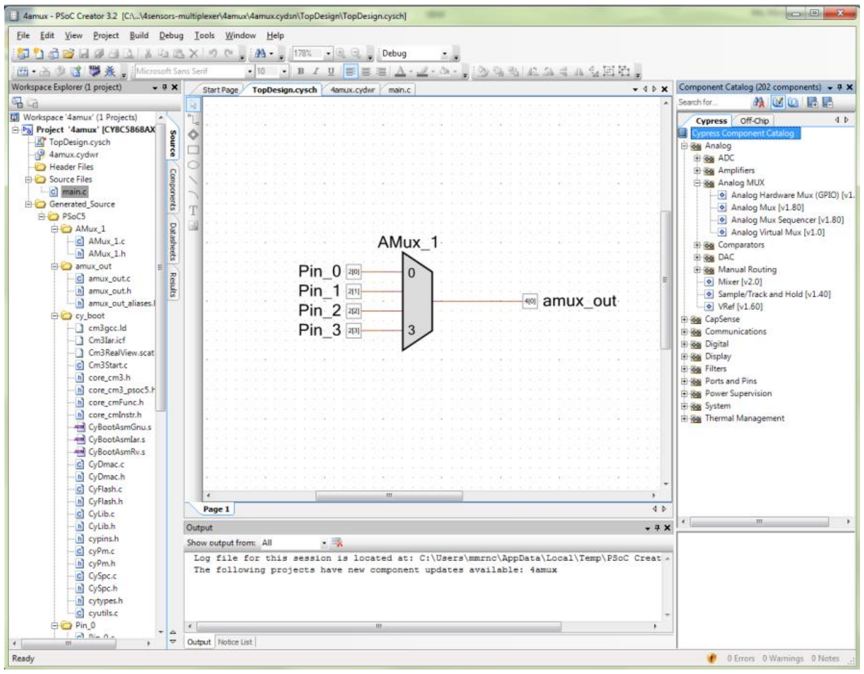


Figure 3-8 Upper-left analog routing quadrant. Each GPIO is connected to analog modules through analog routing buses joined by switch and multiplexers [25].

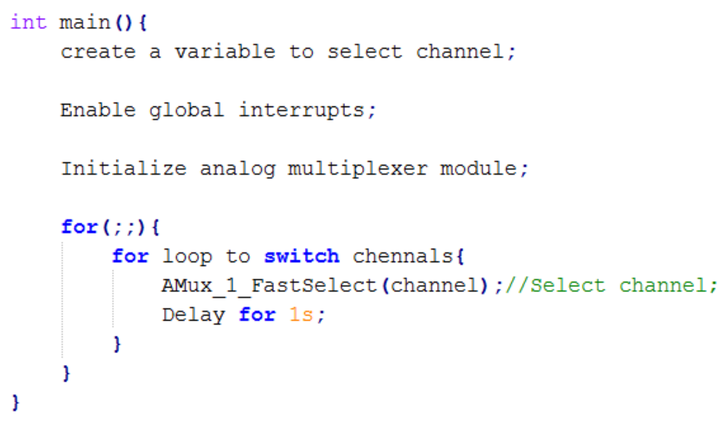
We created a project in PSoC Creator to investigate the feature of the Amux on PSoC chip. First of all, we drew the schematic by connecting the GPIO pins to the input and output pins on the Amux. The components come from the library listed on the right in the Figure 3-9.


4amux - PSoC Creator 3.2 [C:\...\4sensors-multiplexer\4amux\4amux.cydsn\TopDesign\TopDesign.cysch]
File Edit View Project Build Debug Tools Window Help
Workspace Explorer (1 project)
Start Page
TopDesign.cysch
4amux.cydwr
main.c
Component Catalog (202 components)
Cypress
Off-Chip
Cypress Component Catalog
Analog
ADC
Amplifiers
Analog MUX
Analog Hardware Mux (GPIO)
Analog Mux [v1.80]
Analog Mux Sequencer [v1.80]
Analog Virtual Mux [v1.0]
Comparators
DAC
Manual Routing
Mixer [v2.0]
Sample/Track and Hold [v1.40]
VRef [v1.60]
CapSense
Communications
Digital
Display
Filters
Ports and Pins
Power Supervision
System
Thermal Management
AMux_1
Pin_0
Pin_1
Pin_2
Pin_3
0
3
amux_out
Page 1
Output
Show output from: All
Output
Notice List
Ready
0 Errors 0 Warnings 0 Notes

```
Log file for this session is located at: C:\Users\mmrnc\AppData\Local\Temp\PSoC Creat
The following projects have new component updates available: 4amux
```

Figure 3-9 Circuit design and pin configuration of analog multiplexer Creator project. 4 GPIO input pins P2[0]-P2[1] are connected to a AMUX configured with 4 channels. A GPIO output pin P4[0] is connected to the AMUX output.

Since the analog multiplexer is controlled by the programming internal microprocessor, we first initialize the analog multiplexer, then call the function AMux_FastSelect(channel number) to select the channels. In order to have a period of time to collecting the signals from one channel, we set a delay as 1s here. The minimum delay is 5 milliseconds. The pseudo code is shown in Figure 3-10.



```
int main(){
    create a variable to select channel;

    Enable global interrupts;

    Initialize analog multiplexer module;

    for(;;){
        for loop to switch chennals{
            AMux_1_FastSelect(channel);//Select channel;
            Delay for 1s;
        }
    }
}
```

Figure 3-10 pseudo code of analog multiplexer control program

After the project compiles, we can see the interconnection shown in Figure 3-11 where four input GPIO routes are connected to the AMUX bus, in turn the AMUX bus is connected to the output GPIO. During execution, the microprocessor runs and controls the AMUX bus to choose the programmed input.

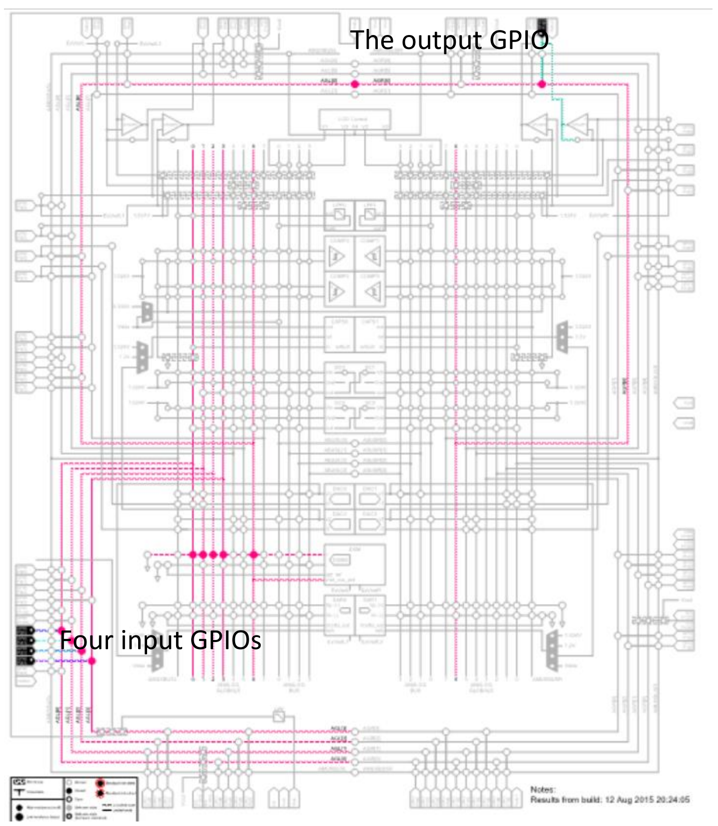


Figure 3-11 The interconnection of PSoC 5LP chip when AMUX project is compiled.

We load the program to the development kit, and run the program. We observe the result of the AMUX output by connecting an oscilloscope to the output and four sensors to each input shown in Figure 3-12.

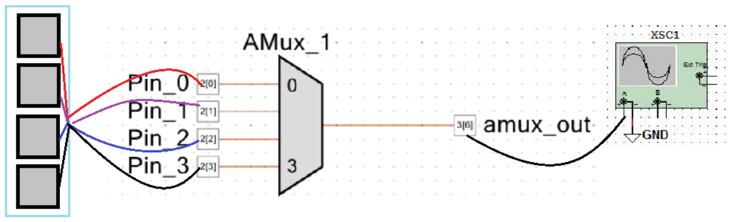


Figure 3-12 Processor collects one GPIO's signal for 1s in a time. We are supposed to observe the high-voltage signal appearing once every 4 seconds when activating one sensor.

3.2.3 Implementation of an Asymmetrical Structure Differential Amplifier

Our sensors produce a signal by interacting with the ubiquitous ambient 60 cycle electromagnetic noise. Essentially the activated sensor (touched sensor) produces a 60 cycle signal of about 300 mV peak to peak while the non-activated sensor produces about 30 mV peak to peak [15].

The differential amplifier is used to increase the level of signal coming from the sensors, and it also reduces the 60 cycle noise signal by subtracting the sensor signal with a noise reference signal. The differential amplifier should give a good differentiation of the signal between activated and non-activated sensors.

We designed the two opamp differential amplifier using two programmable gain amplifiers (PGAs) as shown in Figure 3-13 below.

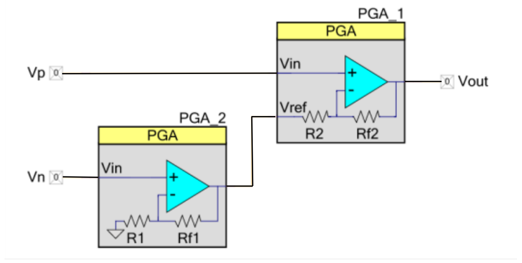


Figure 3-13 The structure of two opamp differential amplifier. Vp is the differential input and Vn is the reference input.

Its behavior is governed by the equation [26],

$$Vout = Vp * \left(1 + \frac{Rf2}{R2}\right) - Vn\left(1 + \frac{Rf1}{R1}\right) \qquad \text{-Equation 1}$$

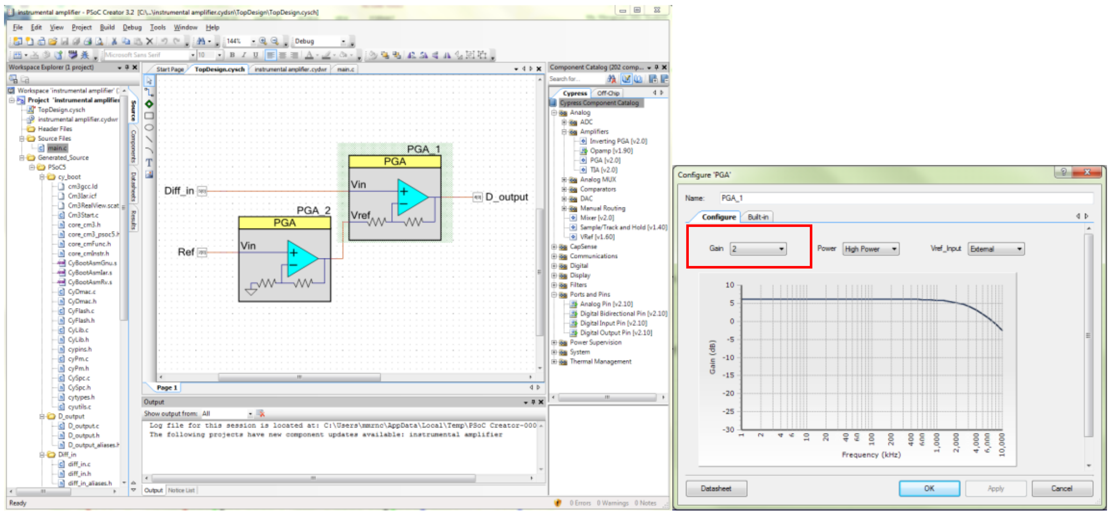
When the input to feedback resistor ratio of the first opamp is the inverse of the second, it behaves similar to a differential amplifier. If

$$\frac{Rf2}{R2} = \frac{Rf1}{R1} \qquad \text{-Equation 2}$$

In this case, Equation 1 becomes

$$Vout = (Vp - Vn) * \left(1 + \frac{Rf2}{R2}\right) \qquad \text{-Equation 3}$$

We created the differential amplifier project in PSoC Creator, built the 2-PGA differential amplifier circuit using the components provided by PSoC Creator library shown in Figure 3-14(a). The amplitude of active sensor signal can be about 1 volt observed from the output of analog multiplexer. The maximum voltage value the PSoC 5lp can provide is 3.3 volts. As a result, we configure the gain (Rf2/R2 or Rfi/Ri) of each PGA to be 2, so that the gain of differential amplifier from equation 3 is 3. The gain of 2 PGAs can be configured in configure 'PGA' window shown in Figure 3-14(b).



```
Show output from: All
Log file for this session is located at: C:\Users\mmrnc\AppData\Local\Temp\PSoC Creator-000
The following projects have new component updates available: instrumental amplifier
```

Name: PGA_1
Gain: 2
Power: High Power
Vref_Input: External

(a) (b)

Figure 3-14(a) shows the schematic in PSoC creator. These two PGAs are provided by Cypress library selected on the right side in the figure. (b) shows the configuration window of PGA. Gain is set to 2 for these two PGAs.

PGAs are required to be initialized by the program. In the program, we firstly enabled global interrupts, and then initialized PGAs. The ‘forever-for-loop’ makes the chip continuously execute. The pseudo code is listed in Figure 3-15.

```
int main(){

    Enable global interrupts;
    Initialize two PGA modules;
    for(;;){

    }
}
```

Figure 3-15 pseudo code of Programmable gain amplifier control program

After the project was compiled, we observe the interconnection of the PSoC 5lp. The wires marked as red and blue are the input signal routes. The black wire connects the two PGAs in the middle of the Figure 3-16. The green wire is the output of differential amplifier.

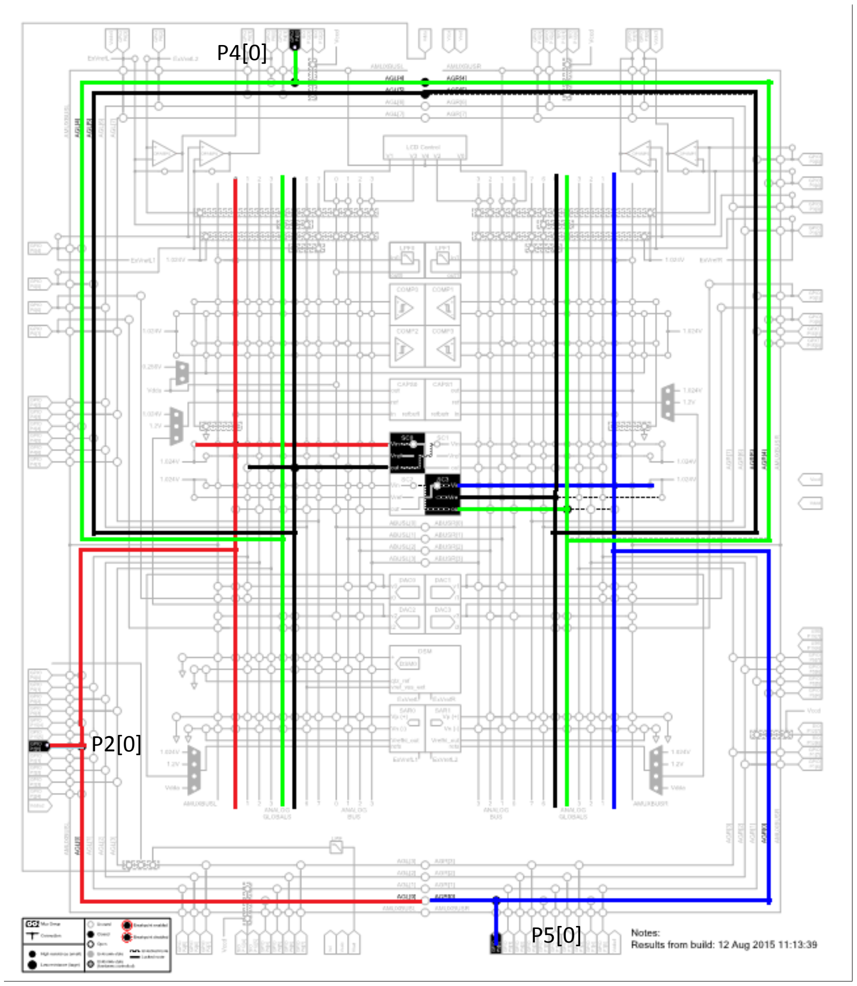


Figure 3-16 The interconnection of the PSoC 5LP chip. Two PGAs are placed in the center of the figure and connected to each other. GPIO P2[0] is the reference input connected to one PGA and GPIO P5[0] is the differential input connected to the other PGA. GPIO P4[0] is the output of this differential amplifier.

To test the performance of this two-PGA differential amplifier, a sensor signal from a four sensor array can be used as input signal, and another (not part of the array) sensor can be used as the reference. To match the output impedance from sensor with the input impedance with PGA, 10Mohm resistors are connected to each input GPIO of two PGAs and to ground. The output can be observed by connecting the output pin to an oscilloscope as shown in Figure 3-17.

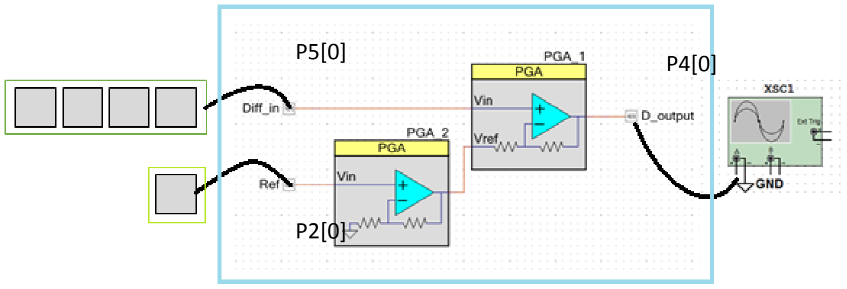


Figure 3-17 Test system of 2-PGA differential amplifier. One sensor output from the 4-sensor array can be used as the differential input. A sensor can be reference. To observe the output of the differential amplifier, an oscilloscope is connected to the output of the amplifier.

3.2.4 Implementation of an ADC and UARTUSB Module

The Smart Carpet system is able to scavenger the sensor signals, process those signals by microprocessor and transfer them to the software part, for example a PC. However, microprocessor is capable of detecting binary signals which are digital signals. As a result, after increasing the amplitude of the signals to the desirable level, we need to change those analog signals to digital signal by applying an Analog to Digital Converter (ADC). PSoC 5LP chip integrates Delta Sigma Analog and Digital Converter (ADC_DelSig) in the analog subsystem which can be used for analog signals to digital signal converting in our design.

As to send the processed data out, a communication module is needed. Universal Serial Bus (USB) is an industry standard which defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between electronic devices [5]. The software board Raspberry pi used on processing data in Smart Carpet system also supports USB communication as well as PSoC 5lp chip. The PSoC Creator Component Catalog contains a Schematic Macro implementation of a communications device class (CDC) interface (also known as USBUART). To start a USBUART-based project, drag the USBUART Schematic Macro labeled 'USBUART (CDC Interface)' from the Component Catalog onto your design. The schematic is created by placing and connecting ADC and USBUART following the differential amplifier as shown in Figure 3-18.

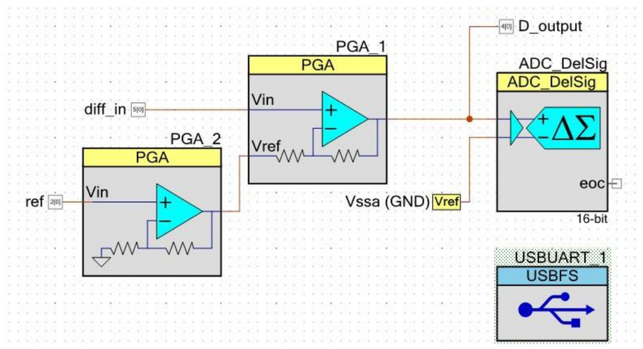
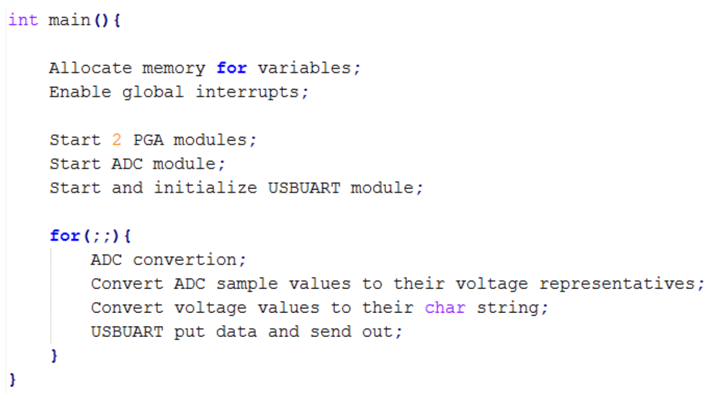


Figure 3-18 Circuit design and pin configuration of ADC and USBUART Creator project. ADC_DelSig module converts the analog signals from differential amplifier to dignal signals. USBUART module sends the digital result to PC. The differential input, reference input and output are configured to be GPIO ports P5[0],P2[0] and P4[0].

In the program as shown in Figure 3-19, we firstly allocated memory for variables and enable global interrupts. Then we initialized the modules. In the forever loop, ADC starts conversion. Functions are called to convert sample values to voltage representatives and then convert to ASCII strings. Finally, ASCII data is put to USBUART sending function to be sent out.



```
int main(){

    Allocate memory for variables;
    Enable global interrupts;

    Start 2 PGA modules;
    Start ADC module;
    Start and initialize USBUART module;

    for(;;){
        ADC convertion;
        Convert ADC sample values to their voltage representatives;
        Convert voltage values to their char string;
        USBUART put data and send out;
    }
}
```

Figure 3-19 pseudo code of ADC and USB control program

The interconnection of the chip is shown in the Figure 3-20 below. GPIO pins P2[0] and P5[0] are connected to the inputs of two PGAs highlighted by red and greed lines. GPIO port P4[0] is the output of the differential amplifier which is also the input of the ADC highlighted in blue. The USB communication ports are highlighted in purple.

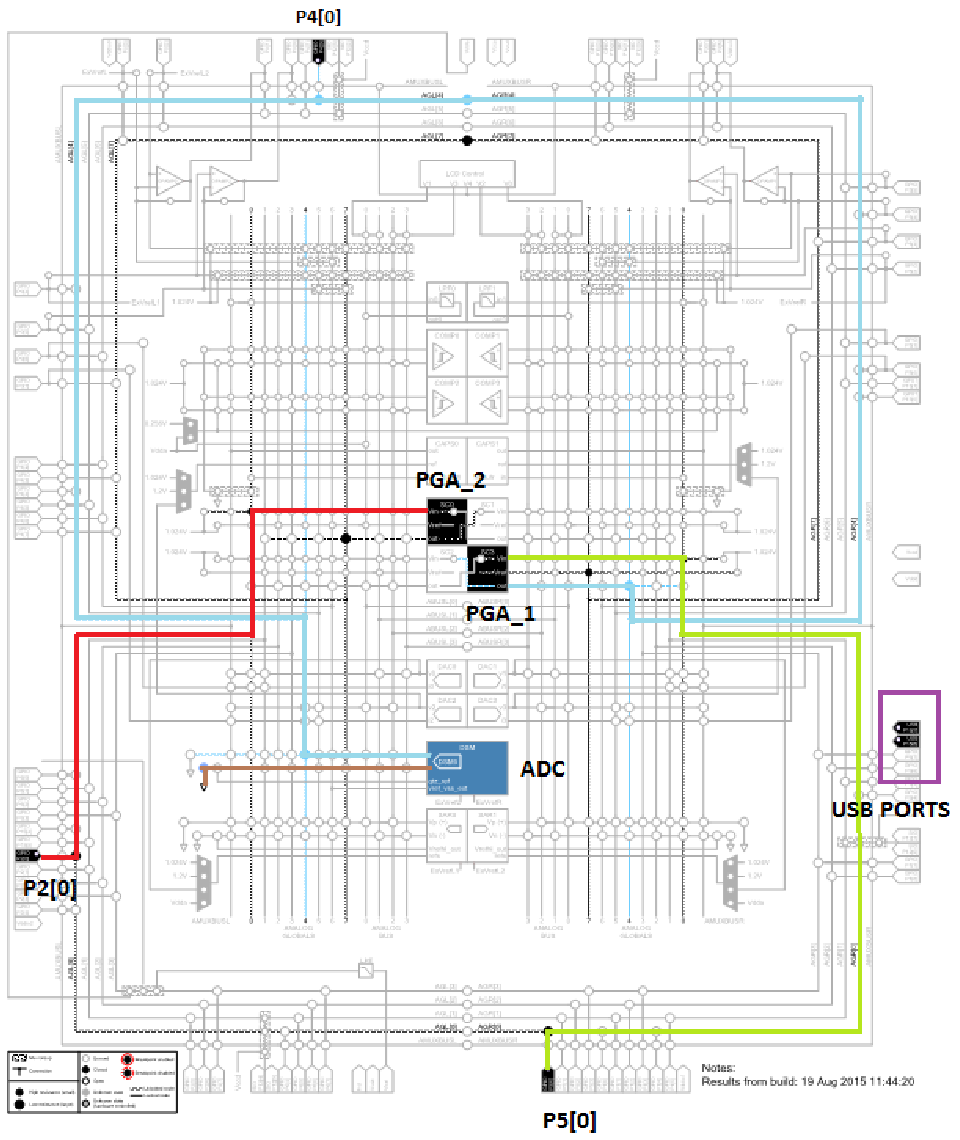


Figure 3-20 The interconnection of PSoC 5LP chip with ADC and USBUART modules added to the project. The output of the differential amplifier P4[0] is connected to the input of ADC highlighted in blue. USB ports are pointed in the purple rectangular.

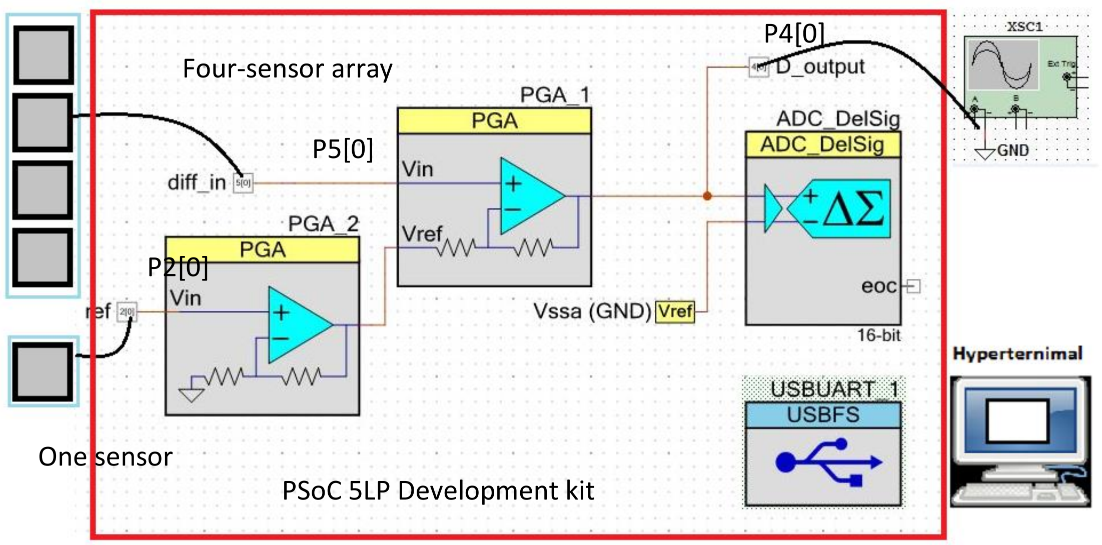


Figure 3-21 The test system of ADC and USBUART modules. Input signal is from one output of 4-sensor array. A sensor signal is used as reference. To test the results converted by ADC and transferred by USB. We compare the amplifier output signal displayed in an oscilloscope to the data displayed in HyperTernimal.

To test the system, one output of the four-sensor array can be connected to the differential input of the amplifier. A single sensor used as reference is connected to the reference input. An oscilloscope is connected to the output of the differential amplifier to observe the behavior of this amplifier. The development kit is connected to a PC via USB. The digital value can be received and displayed on software HyperTerminal. The test system connection is presented in Figure 3-21.

3.3 Single Board System

3.3.1 Implementation of a 32-Sensor-Signal Scavenging System

From the output of Delta Sigma ADC, we set the voltage values of sensors. The information we are interested in extracted from sensor data is the sensor status (active or non-active) and the location of the sensors. One bit number is sufficient to store one sensor status. If sensor is active, the status number can be set to 1. Otherwise we set it to 0. A voltage threshold is chosen to separate activation from non-activation. The positioned data can be put into a 32 sensors bit map. For example say, the second sensor is active, the status array is 0000 0000 0000 0000 0000 0000 0000 0010. We can combine four binary numbers into their hexadecimal representation, so the status array is in hexadecimal as 00000002H. We convert the array to its ASCII number representation, so the data can be ready to send out.

This process is implemented and controlled by the microprocessor. We created a project of 32-sensor scavenging system and the schematic is presented in Figure 3-22. Note that the number of AMux input channels has been changed to 32.

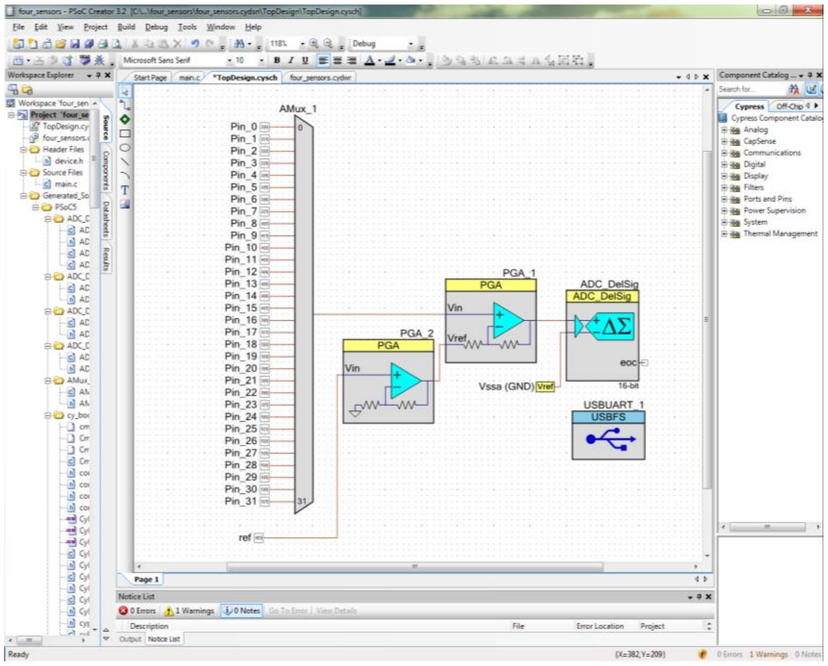
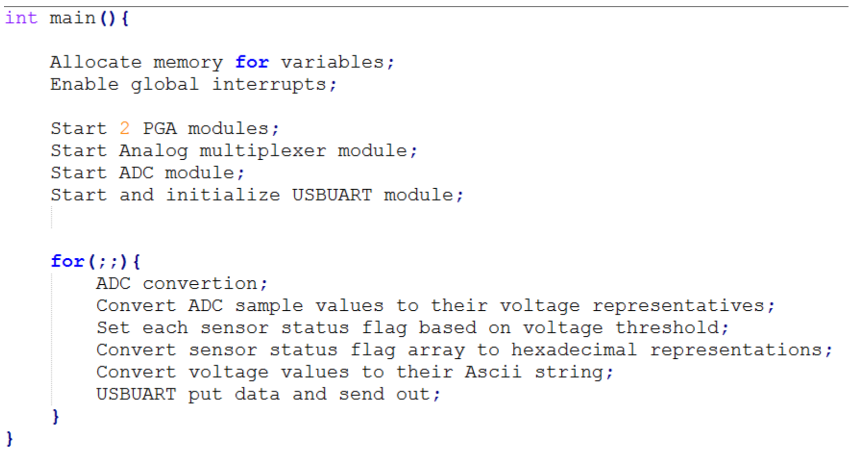


Figure 3-22 The circuit design and pin configuration of 32 sensor detection system. An analog multiplexer is connected to a differential amplifier. The amplified signal is fed input ADC_DelSig. The printable digital data is sent out by USBUART module.

For programming, we update the status array by comparing the digital voltage values of sensors with the voltage threshold, next convert the array to hexadecimal, then convert it to its ASCII representation, finally send to USBUART.



```
int main(){

    Allocate memory for variables;
    Enable global interrupts;

    Start 2 PGA modules;
    Start Analog multiplexer module;
    Start ADC module;
    Start and initialize USBUART module;


    for(;;){
        ADC convertion;
        Convert ADC sample values to their voltage representatives;
        Set each sensor status flag based on voltage threshold;
        Convert sensor status flag array to hexadecimal representations;
        Convert voltage values to their Ascii string;
        USBUART put data and send out;
    }
}
```

Figure 3-23 pseudo code of 32 sensor detection system control program

After compiling the project, we observed the interconnection in the PSoC5lp, 32 GPIO pins are highlighted.

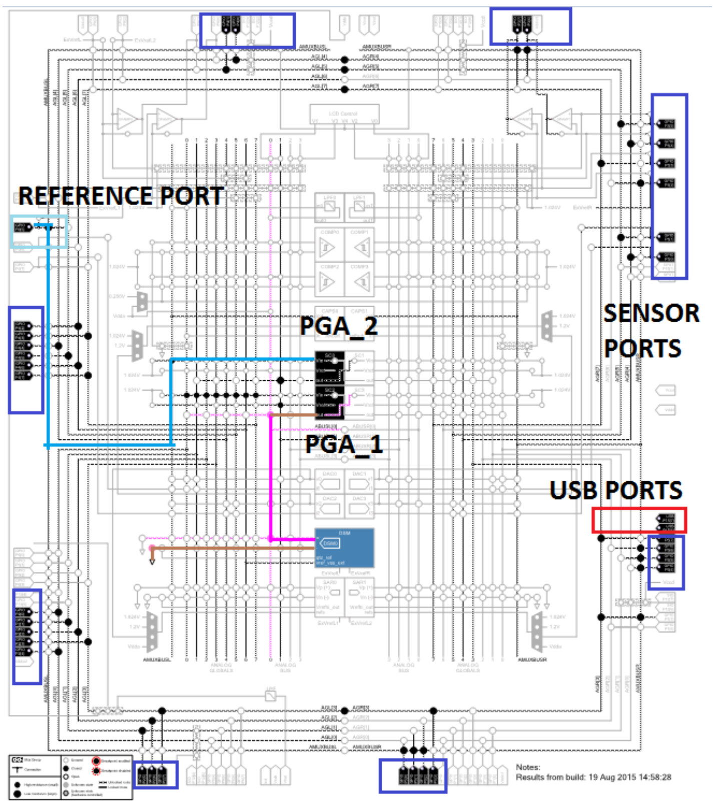


Figure 3-24 The interconnection of PSoC 5LP chip of 32 sensor detection system. There are 32 GPIO pins are highlighted for 32 output from sensors

To test this project, we connected 32 inputs to sensor array with 32 sensors on it. We used a single independent sensor as reference. HyperTerminal installed in a computer received and displayed the data sent by PSoC development kit.

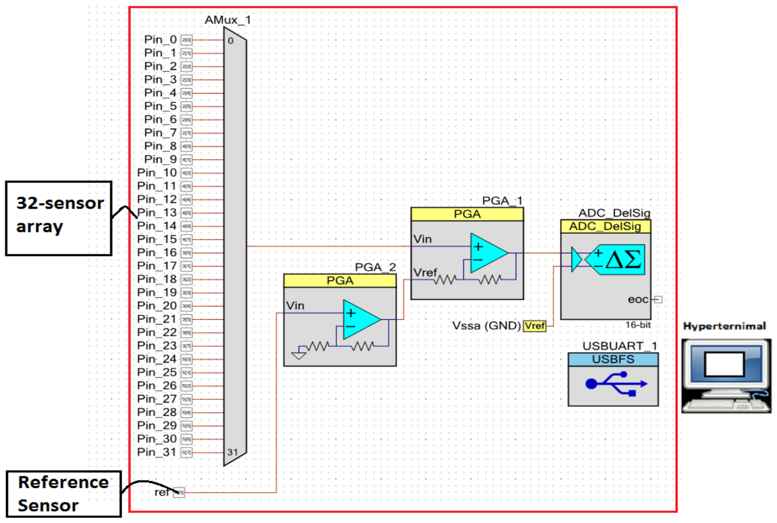


Figure 3-25 32 sensor testing system. We connect the 32-sensor array to the analog multiplexer 32 input pins. A signal sensor is used as a reference. The digital data string is sent out through USB to a PC and displayed on HyperTerminal.

3.3.2 The First PCB Boards

We've implemented the 32 sensor signal scavenging system on PSoC 5LP development kit by applying an analog multiplexer, differential amplifier, DelSig ADC and USBUART. The system can detect the sensor status, convert the sensor signals to a series of printable strings and send them to a PC. However, the PSoC 5LP development kit is for testing. It contains some modules that are not related to our system. As a result, we made our Printed Circuit Boards for designed scavenging system.

The main component on the board is a PSoC 5LP chip. To make it work, we need to provide a suitable power system for the chip. The chip is programmable, so that we need to design a program loading circuit. The board is required to communicate with a PC, so that we need to design a USB based communication circuit. In addition, some components in the chip are connected to other components. For instance, we connect a 10Mohm resistance to each input of a differential amplifier to match the impedance of the sensors and the amplifiers.

3.3.2.1 Analog, digital and grounding power consideration

It is desirable to provide separate analog and digital power supplies for sensitive analog systems. Separate power and ground pins are designed for the analog and digital blocks in PSoC 5LP. In analog power domain, V_{DDA} supplies for all analog peripherals and analog core regulator. Importantly, V_{DDA} must be the highest voltage present on the device, which means all the other supply pins must be equal to or less than V_{DDA}. The output of the analog core regulator is V_{CCA}. The return path for analog is V_{SSA} which is the ground for all analog peripherals. In digital power domain, V_{DDD} supplies for all digital peripherals and digital core regulator. V_{CCD} is the output of the digital core regulator. The return path is V_{SSD}, ground for all digital logic and I/O pins. There is another domain for I/O pins in which the power is provided by V_{DDIO}. The last domain is for boost which is not used in our designed system. A summary of the power supply connections is given in Table 3-1 [27].

Table 3-1 Psoc 5LP Power Domain[28]

Power Domain	Associated Power and Return Pins
Analog	VDDA,VCCA,VSSA
Digital	VDDD,VCCD,VSSD
I/O pins	VDDIO0,VDDIO1,VDDIO2,VDDIO3,VSSD

The analog system power voltage, applied to V_{DDA} relative to V_{SSA}, can be up to 5.5 V, and must be greater than or equal to all other applied power voltages. The minimum voltage that can be applied to any V_{DDX} power pin is 1.8 V [28]. Zhenduo Zhai, an undergraduate student in our lab, helped us to design the power supply circuit with analog and digital power separated. The analog power voltage is 3.3 V which is equal to digital power voltage. The power supply circuit is presented in Figure 3-26 below.

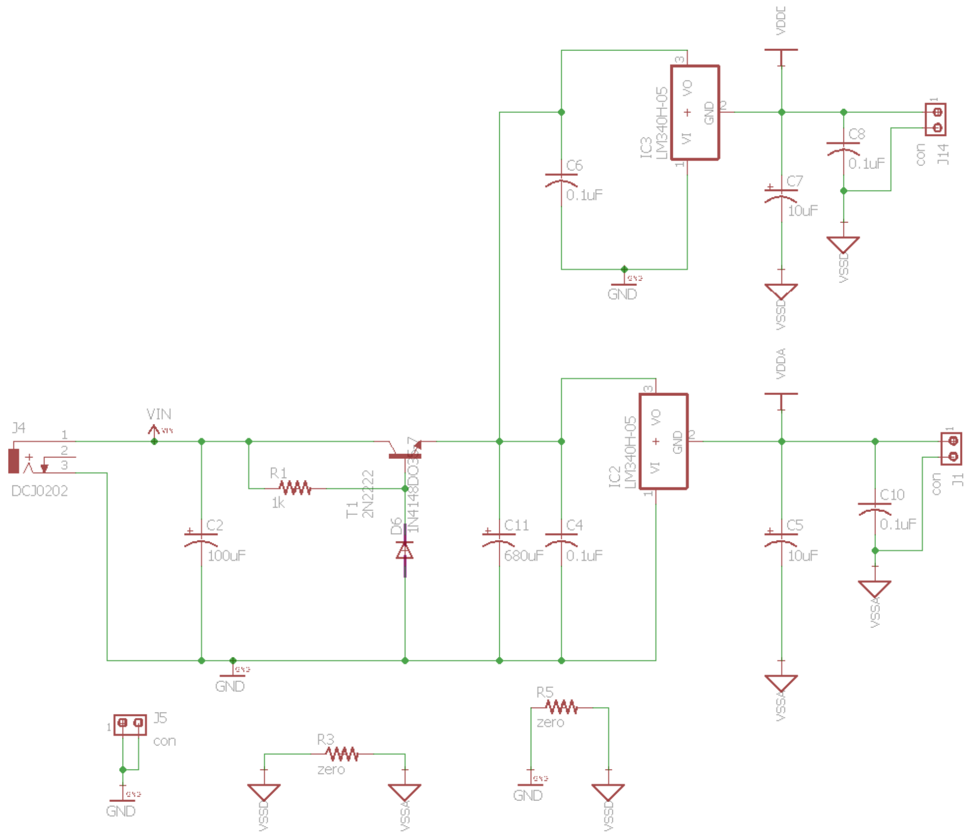
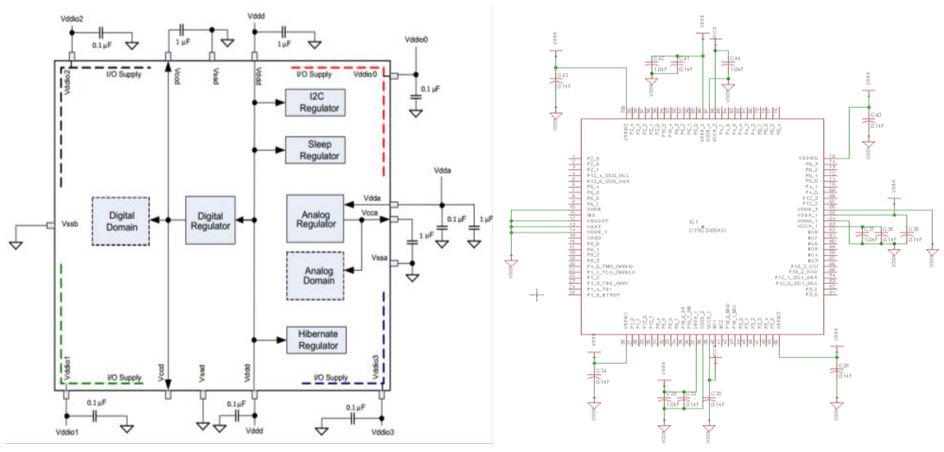


Figure 3-26 Power supply circuit for PSoC based PCB board. Transistor T1 together with the capacitor C11 converts the fluctuant voltage into DC voltage. The regulators IC1 and IC2 change the DC voltage to 3.3 volts which is the output of this power supply circuit.

The output signals V_{DDD} related to V_{SSD} and V_{DDA} related to V_{SSA} can provide the power to both PSoC chip and other off-chip components. To reduce power supply noise throughout the PSoC chip, each V_{DDX} pin should be connected to a 0.1 µF ceramic decoupling capacitor, as Figure 3-27(a) shows [28].



(a) (b)

Figure 3-27(a) shows PSoC 5LP power system provided by data sheet. VDDX pins are better to connect to ceramic decoupling capacitor.(b) is the PSoC 5LP power system designed for Smart Carpet PSoC based PCB board.

Based on the power supply requirement, I designed a power system for the PSoC 5LP chip, shown in Figure 3-27(b) above. Since the boost is not being used, the VBAT, VSSB, and VBOOST pins must be connected to ground, and the IND pin needs to unconnected [28].

3.3.2.2 Support circuits

a. Designed program loading circuit for PSoC chips

PSoC 5LP chips support the serial wire debug (SWD) interfaces for device flash programming and debug. The interface exists on a set of pins in I/O port 1. We choose to use a Cypress CY8CKIT-002 MiniProg3, a programming device provided by Cypress Semiconductor to program the PSoC 5LP chip. MiniProg3 can program the PSoC 5LP chip using the serial wire debug (SWD) interfaces. Figure 3-28 shows the connections between this header for MiniProg 3 and the device SWD pins [28].

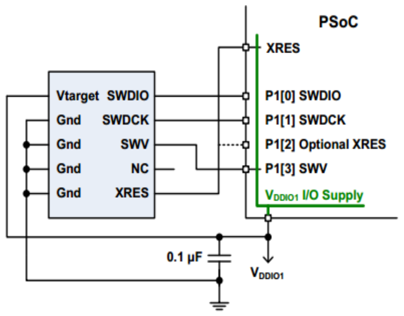


Figure 3-28 SWD Connections to PSoC 5LP. A 10-pin, dual row, 50-mil pitch connector shown on the left is connected to P1[0]-P1[3] and XRES pins on PSoC 5LP[28].

Based on this principle, I designed the programming interface circuit as shown in Figure 3-29 below. Port SWDIO, SWDCK, SWO, TDI and /XRES are connected to P1[0],P1[1],P1[3],P1[4] and XRES ports of the PSoC5lp chip.

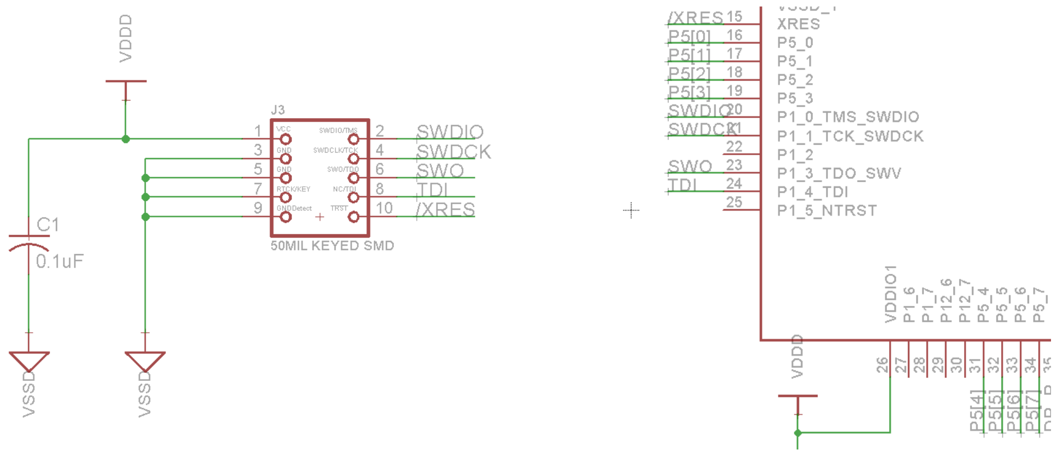


Figure 3-29 programming interface circuit of PSoC 5LP based boards. 50-mil pitch connector is used for Mini-prog3

b. Designed USB connection circuit

USB is a serial bus, using four shielded wires for the USB 2.0 variant: two for power (VBUS and GND), and two for differential data signals (labelled as D+ and D− in pinouts) [27, 29] as shown in Table 3-2.

Table 3-2 USB 1.x/2.0 mini/micro pinout

Pin	Name	Description
1	V_{BUS}	+5 V
2	D−	Data−
3	D+	Data+
4	ID	not connected
5	GND	Signal ground

Based on this principle, I designed the circuit in Figure 3-30 below for programming. D- (DM) and D+ (DP) pins are connected to the 22ohm resistors and then the resistors are connected to P15[7] and P15[8] on PSoC 5lp chip.

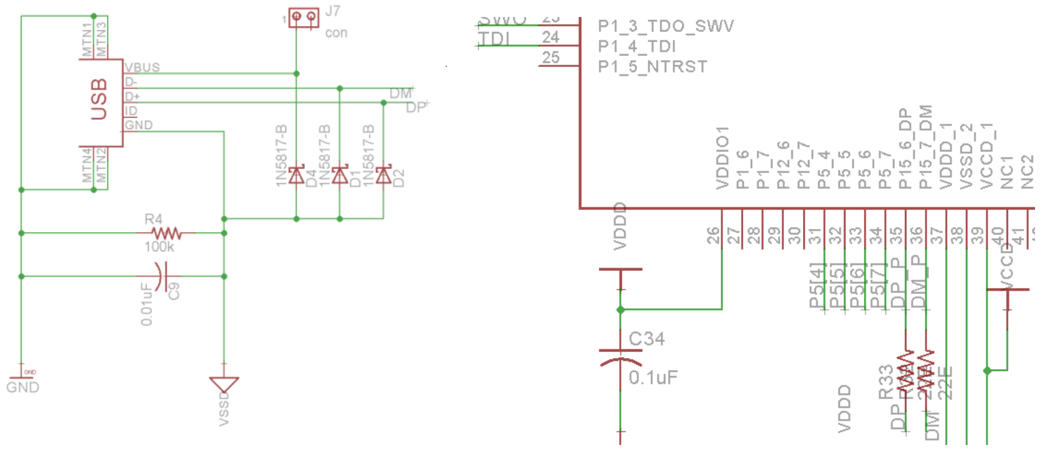


Figure 3-30 The connection between USB port to PSoC 5LP chip.

c. Differential amplifier

To match the impedance between the sensor output signals to the PGAs inputs, we need to connect 10Mohm resistors to the inputs of differential amplifier, which are differential input and reference input. In the designed system, GPIO P0[5] or P0_5 is assigned to be the differential input pin and GPIO P0[3] or P0_3 is assigned to be the reference input pin. While P0_6 is the output of analog multiplexer. The peripheral circuit of differential amplifier is present in the Figure 3-31 below.

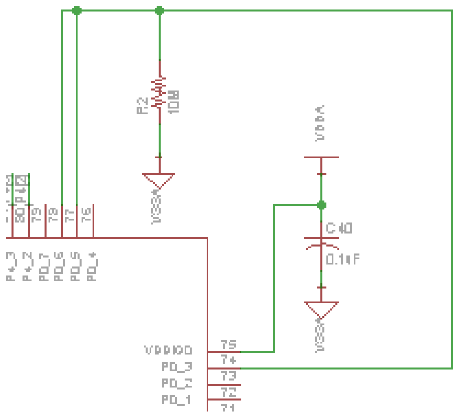


Figure 3-31 peripheral circuit of differential amplifier. GPIO P0[5] or P0_5 is assigned to be the differential input pin and GPIO P0[3] or P0_3 is assigned to be the reference input pin. While P0_6 is the output of analog multiplexer

3.3.2.3 Manufacture

We used a free software PCB Eagle [30] to design our PSoC 5lp based PCB boards. For PSoC 5lp system, it is best to use a multi-layer PCB with separate layers dedicated to the V_{SS} and V_{DD} supplies. This gives good decoupling and shielding effects. It is recommended to provide separate fills on these layers for V_{SSA}, V_{SSD}, V_{DDA}, and V_{DDD} [31]. As a result, we designed 4-layer PCB layout for my boards including separately power layer and ground layer. The first layer contains most of routes. The second layer is the ground layer, where it includes the ground plane, the V_{SSD} plane and the V_{SSA} plane. The third layer is for power including V_{DDA} and V_{DDD} planes. The bottom layer holds some I/O routes.

In the first layer which is also called top layer shown in Figure 3-32, most of the wires are routed including the wires of power supply circuit, USB connection and programming and debugging interface. The standard decoupler for external power is a 100 μF capacitor. Supplementary 0.1 μF capacitors are placed as close as possible to the VSS and VDD pins of the PSoC 5LP chip, to reduce high frequency power supply ripple.

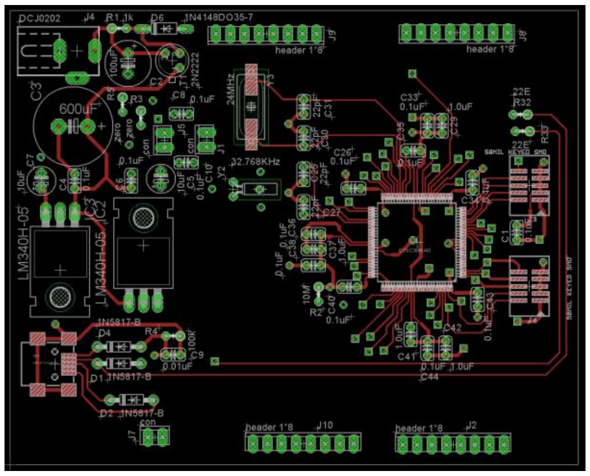


Figure 3-32 The top layer layout of the first PSoC 5lp based PCB board.

The second layer shown in Figure 3-33 is ground layer; all the ground wires are routed in this layer. There are three ground planes in this layer, which are GND plane, V_{SSA} plane and V_{SSD} plane. All the pads needed to connect to the ground can directly connect to specific plane to avoid ground loops. R3 and R5 are resisters with 0 ohm. These two resisters are designed to connect to each other and gather all the ground returns.

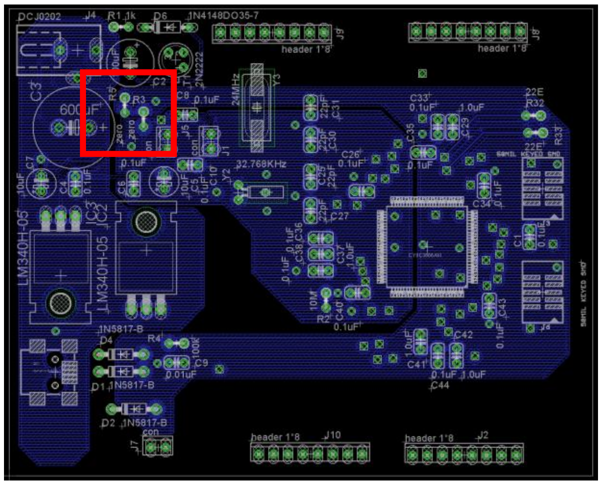


Figure 3-33 The second layer layout of the first PSoC 5lp based PCB board.

The third layer shown in Figure 3-34 is the power layer which contains V_{DDA} and V_{DDD} planes. All the power pads can directly connect to specific plane. The left half of PSoC chip gathers analog power pins and the right half gathers the digital pins.

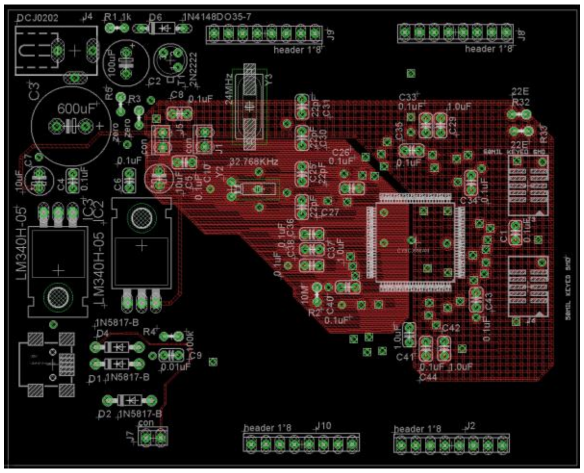


Figure 3-34 The third layer layout of the first PSoC 5lp based PCB board.

The fourth layer as shown in Figure 3-35 also called the bottom layer contains all the routes which cannot be routed in the first layer. Mostly they are the wires to connect the sensor input connectors and the GPIO pins.

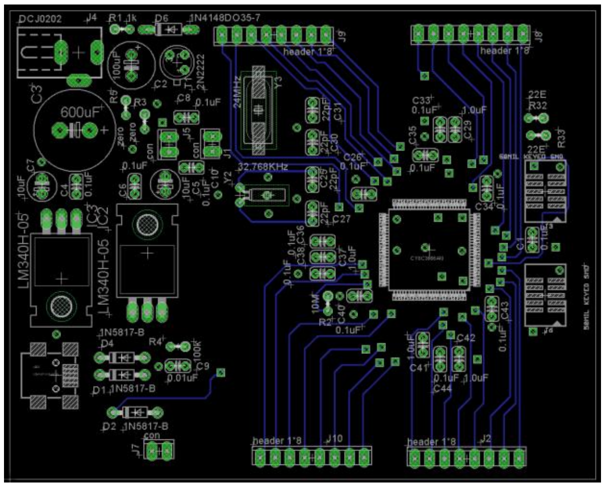


Figure 3-35 The bottom layer layout of the first PSoC 5lp based PCB board.

3.3.2.4 Components Assembly

Most of the components on the first PCB boards are through-hole devices, which are easy to be assembled by hand. However, we chose PSoC 5LP chips with 100-pin TQFP package, which mean it is a 100-pin Surface Mount Device (SMD). The pitch, the distance from two pins, is 0.5mm. To solder such tiny SMD, we need special materials compared with soldering through-hole devices. Critical for the process is unsoldering wick and flux pen. For solder, a 0.040" diameter works well for this work. A solder iron with a screwdriver tip is essential [32].

The first step of the soldering process is to put the chip on the board and move it until all of its pins are aligned with all the pads. Once the chip is lined up, carefully tack down one pin in each of two corners to secure in on the board. Sometimes, alignment check and adjustment is necessary. Then once you are sure the chip is in the right place, put some flux on the pins. The most important step is solder. Make sure the tip of your iron has some solder but not much. Let the iron tip touch the some pins on the chip for about 1 second and repeat for all the pins. Avoid making solder bridges; If you do, add more flux and use unsoldering wick to separate it. The first board is presented in Figure 3-36 below.

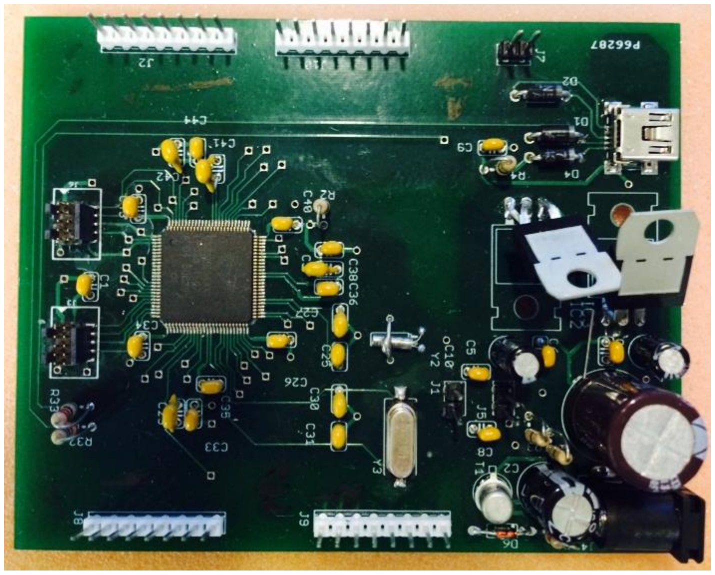


Figure 3-36 The first PSoC5LP based Smart Carpet board. The chip assembled on the left side of the board is PSoC5LP chip.

3.4 Two Boards System – the Second PCB Boards

3.4.1 A three-Programmable Gain Amplifier Differential Amplifier

In order to improve the performance of differential amplifier, we applied a 3-Programmable Gain Amplifier (PGA) one with symmetrical structure. The classical three opamp differential amplifier is a popular topology for its high input impedance and CMRR as shown in Figure 3-36. It has two stages: the differential amplifier and difference amplifier. The differential amplifier stage provides high input impedance and common mode rejection. The difference amplifier stage nulls the common mode signal and provides the differential to single ended conversion [26].

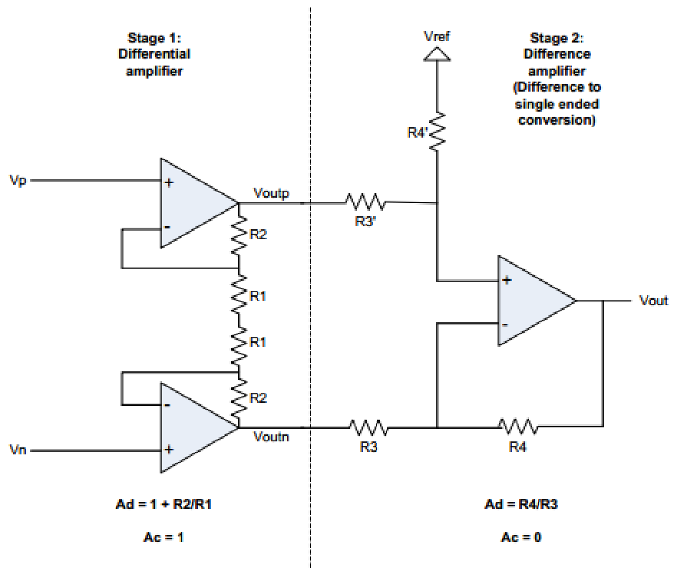

Stage 1: Differential amplifier
Vref
Stage 2: Difference amplifier (Difference to single ended conversion)
R4'
Vp
Voutp
R3'
R2
R1
R1
R2
Vout
Vn
Voutn
R3
R4
Ad = 1 + R2/R1
Ac = 1
Ad = R4/R3
Ac = 0

Figure 3-37 Classical Three Opamp Instrumentation Amplifier

The differential output voltages of the differential amplifier are given by

$$\mathrm{Voutp} = \mathrm{Vn} + (\mathrm{Vp} - \mathrm{Vn}) * (1 + \frac{R2}{2R1}) \qquad \text{-Equation 4}$$

$$\mathrm{Voutn} = \mathrm{Vp} + (\mathrm{Vn} - \mathrm{Vp}) * (1 + \frac{R2}{2R1}) \qquad \text{-Equation 5}$$

$$\mathrm{Voutp} - \mathrm{Voutn} = (\mathrm{Vp} - \mathrm{Vn}) * (1 + \frac{R2}{R1}) \qquad \text{-Equation 6}$$

By expressing Voutp and Voutn in terms of differential (Vd) and common mode signals (Vc), we can get,

$$\mathrm{Voutp} = \mathrm{Vc} + \frac{\mathrm{Vd}}{2} * (1 + \frac{R2}{R1}) \qquad \text{-Equation 7}$$

$$\mathrm{Voutn} = \mathrm{Vc} - \frac{\mathrm{Vd}}{2} * (1 + \frac{R2}{R1}) \qquad \text{-Equation 8}$$

$$\text{Where} \quad \mathrm{Vc} = \frac{\mathrm{Vp+Vn}}{2} \text{ and } \mathrm{Vd} = \mathrm{Vp} - \mathrm{Vn} \qquad \text{-Equation 9}$$

The second differential amplifier is governed by

$$\text{Voutp} = \frac{R4}{R3} * (\text{Voutp} - \text{Voutn}) \qquad \text{-Equation 10}$$

We implement the classical three opamp differential amplifier in PSoC creator as shown in Figure 3-38 below with 3 programmable gain amplifiers(PGAs) and two external resistors. The gains of PGA_1 and PGA_2 are both 1 and the gain of the PGA_3 is 2. To observe both analog output and digital values from this differential amplifier, we also place an ADC and an USBUART followed. USBUART is able to send the digital value to a PC through USB, so that we display the received data on HyperTerminal installed in the PC.

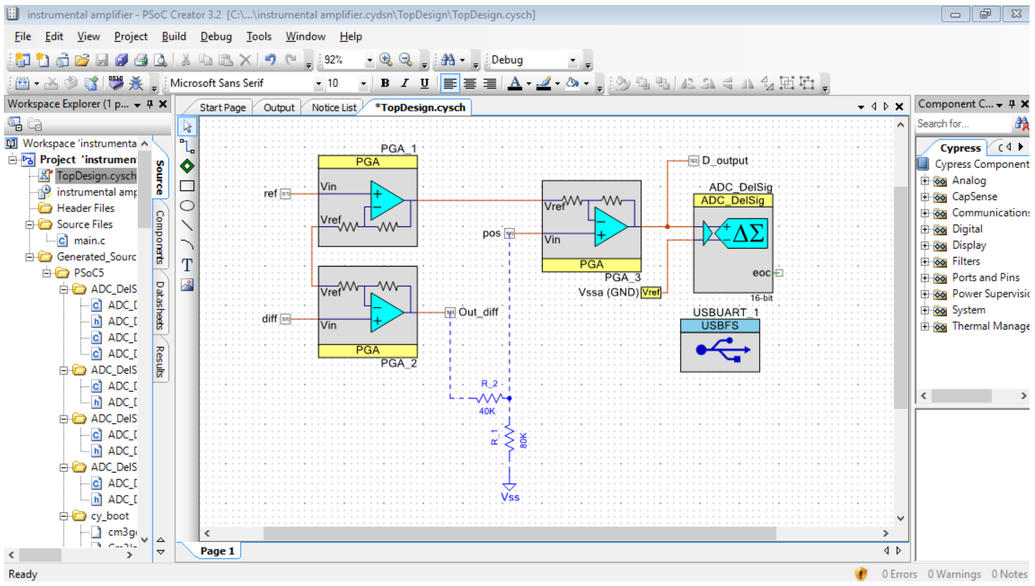
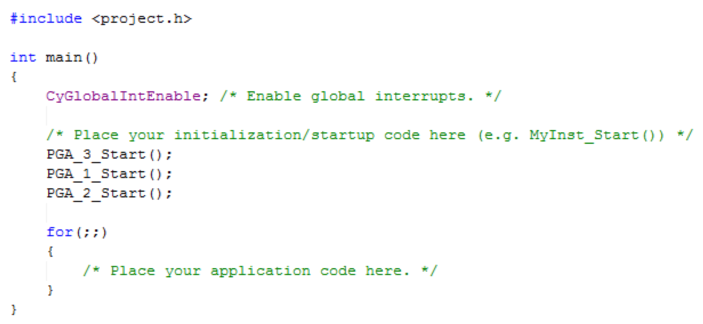


Figure 3-38 shows the schematic of 3-PGA differential amplifier in PSoC creator. These three PGAs are provided by Cypress library selected on the right side in the figure. The two blue resistors are off-chip components which need to be connected to the boards.

PGAs are required to be initialized by the program. In the program, we firstly enabled global interrupts, and then initialized PGAs. The ‘forever-for-loop’ makes the chip continuously execute. The pseudo code is presented in Figure 3-39.



```
#include <project.h>

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    PGA_3_Start();
    PGA_1_Start();
    PGA_2_Start();

    for(;;)
    {
        /* Place your application code here. */
    }
}
```

Figure 3-39 The pseudo code of the differential amplifier

After the project was compiled, we observe the interconnection of the PSoC 5lp. The differential input and reference input are routed to GPIO port P3[6] and P3[6]. The differential amplifier's output is fed into ADC. We can also find USB ports in the Figure 3-40.

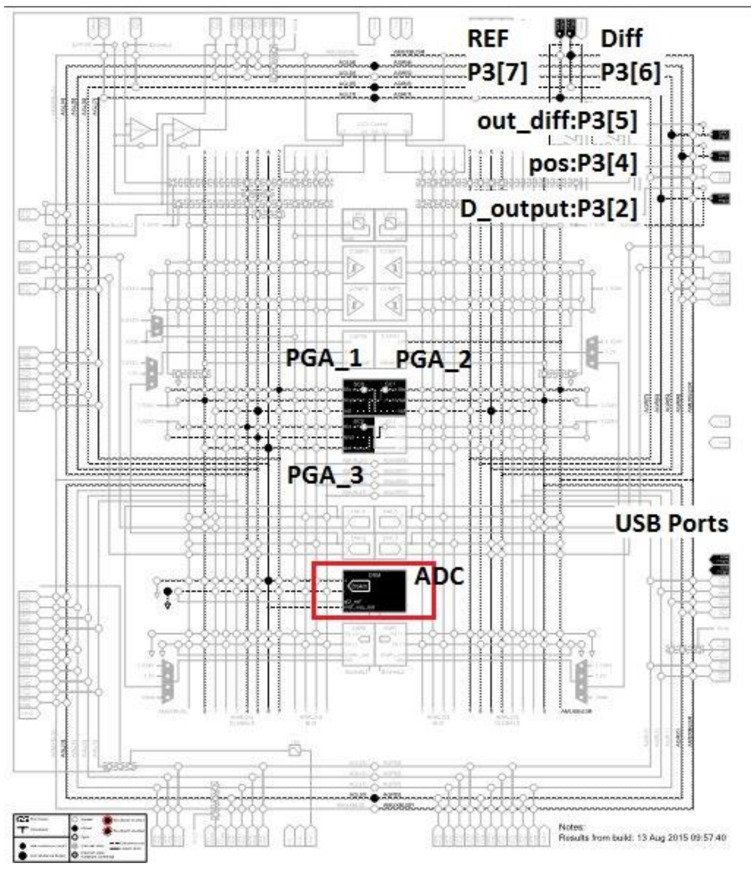


Figure 3-40 The interconnection of PSoC chip after 3-PGA differential amplifier project compiled.

To test the performance of this three-PGA differential amplifier, a sensor signal from a four sensor array is used as input signal, and another individual sensor is used as the reference. The differential amplifier needs two external resistors with values of 40k and 80k. To match the output impedance from sensor with the input impedance with PGA, 10Mohm resistors are connected to each input GPIO of two PGAs and to ground. The analog output can be observed by connecting the output pin to an oscilloscope and the digital data from amplifier can be displayed on HyperTerminal on a PC by connecting the PSoC board to the PC through USB as shown in Figure 3-41.

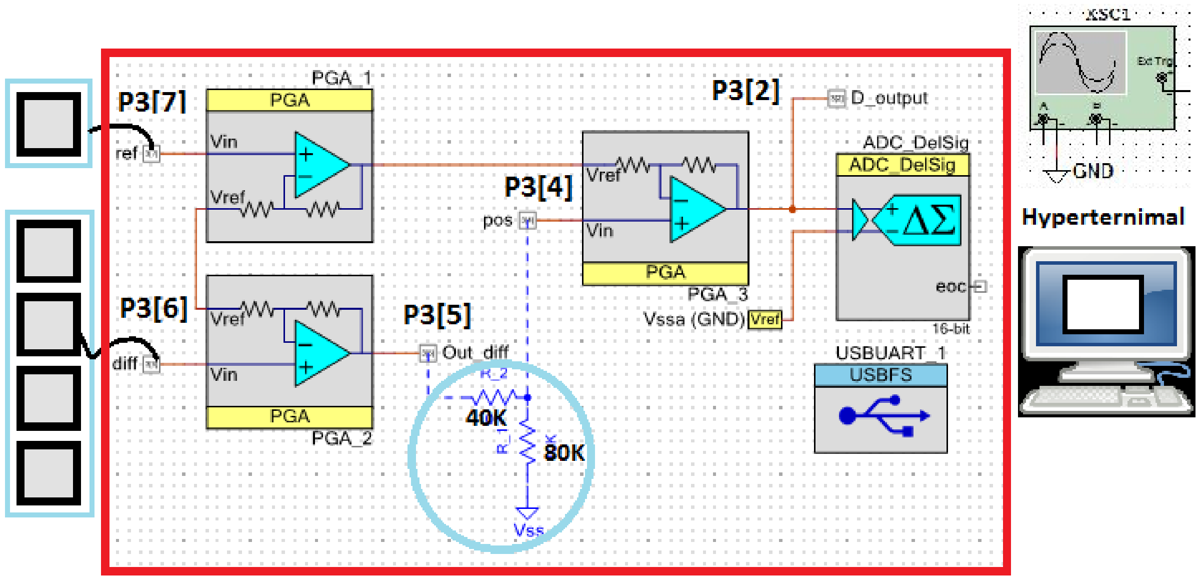


Figure 3-41 The test circuit of differential amplifier. A sensor signal from a four sensor array can be used as input signal and another (not part of the array) sensor can be used as the reference.

3.4.2 Boards Power Consideration and Support Circuits

In the second board, we used the same power supply circuits as the one in the first board. The power supply circuit can provide the separate analog power and digital power. They both have voltage values of 3.3 volts as presented in Figure 3.26. We also used some ceramic decoupling capacitors to connect the power supply to the power pins on PSoC 5lp chip. The connection of power ports is shown in Figure 3-27 as the same as the first board. Also, the second board contain the same USB communication surface as the first boards. For the second board, on the one hand, we improved the performance of differential amplifier, so we designed a new support circuit on the board for the new differential amplifier. On the other hand, we implement multi-board communication module I^2C, so we designed the support circuit for it as well.

3.4.2.1 Differential Amplifier

To match the impedance between the sensor signals to the PGAs inputs, we need to connect 10Mohm resistors to the inputs of differential amplifier, which are differential input and reference input. In the second boards, GPIO P3[6] or P3_6 is assigned to be the differential input pin and GPIO P3[7] or P3_7 is assigned to be the reference input pin. Connector J7 is used to connect reference sensor. Also, based on the schematic in Figure 3-37, to make the gain of the third PGA to be 2, we need to connect two external resistors with value 40kohm and 80kohm in P3[4] and P3[5]. The peripheral circuit of differential amplifier is present in the Figure 3-42 below.

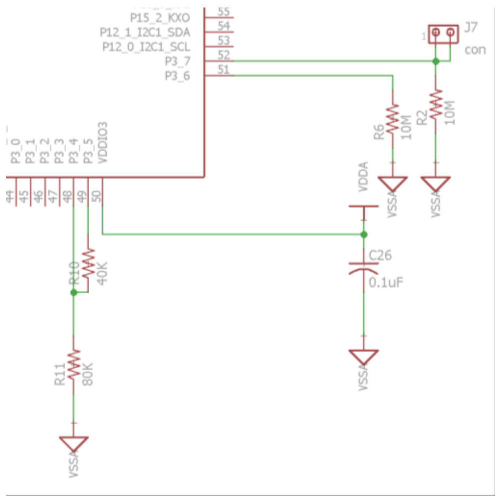


Figure 3-42 The peripheral circuit of differential amplifier.R2 and R6 are connected to match the impedance between sensors and PGAs inputs. R10 and R11 make the gain of third PGA in differential amplifier to be 2.

3.4.2.2 I^{2}C

One board can collecting up to 40 sensor signals which is not enough for Smart Capet system. To support multi-board communication, we use I2C communication module in PSoC chips for sending and receiving data. The I^{2}C bus requires external pull-up resistors. In our second board, GPIO pin P12[4] and P12[5] are assigned to be I^{2}C clock bus and data bus. Pull-up resistors value is 2 kΩ.

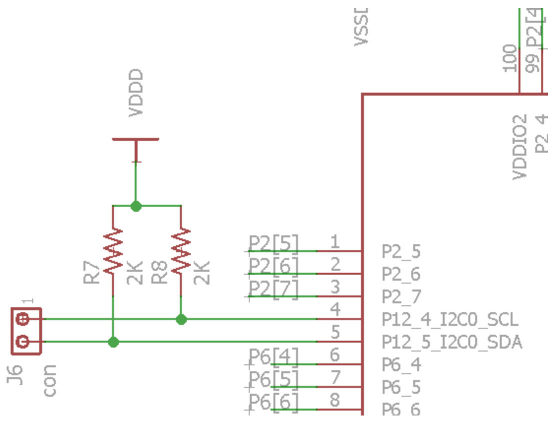


Figure 3-43 The peripheral circuit of I2C communication interface. Pull-up resistors Rp typical value is set to be 2 kΩ.

3.4.3 Manufacture

Using the same principles suggested by Cypress Semiconductor Inc., I again designed a board layout with four layers. The improvement is to assign all the analog routes to the first layer and all the digital routes to the bottom layer. The second layer is the ground layer and the third layer is the power layer. Since digital signals have voltage levels of 3.3 volts, while analog signals are relatively smaller, separating the digital and analog signals can reduce the influence from digital signals to analog signals and improve the performance of the boards.

There should be a single point for gathering all ground returns. To achieve this, wires connect the analog and digital grounds to a single ground return.

It is necessary to use some capacitors for decoupling. Supplementary 0.1 μF capacitors have been placed as close as possible to the V_{SS} and V_{DD} pins of the device, to reduce high frequency power supply ripple.

The first layer as shown in Figure 3-45 is designed as analog layer which contains most of the analog routes. Similar to the first board, Supplementary 0.1 μF capacitors are placed as close as possible to the V_{SS} and V_{DD} pins of the PSoC 5LP chip, to reduce high frequency power supply ripple. However, we rotated the PSoC chip to connect more GPIO ports to sensor connecters. In the second board, there are 5 pairs of sensor input connectors which can be connected to 40 sensors.

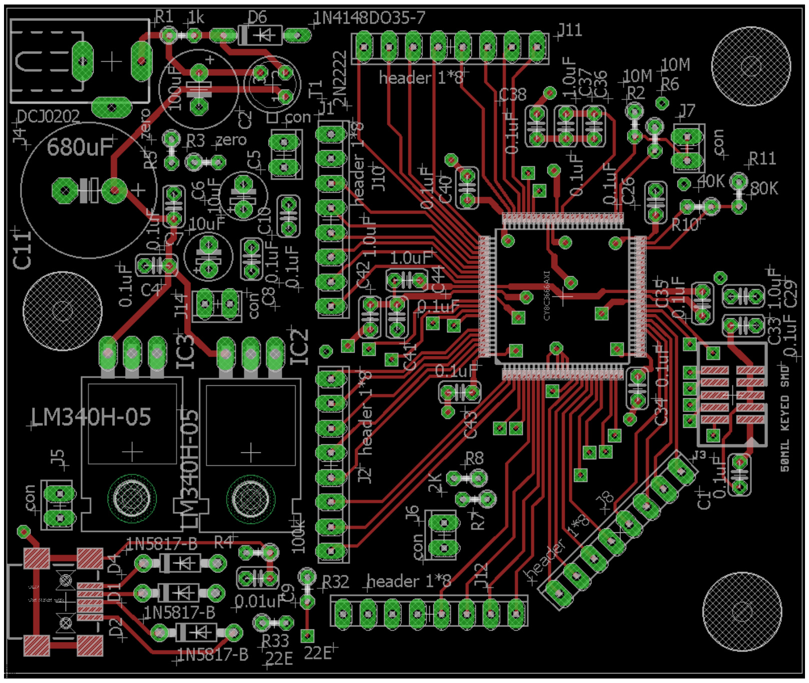


Figure 3-44 The top layer layout of the second PSoC 5lp based PCB board.

The second layer as shown in Figure 3-46 is ground layer. We designed separate analog and digital ground planes; both connected to R3 and R4 (in the white box) with the first layer image superimposed have value of 0 ohms and connect to ground plane. So that R3 and R4 is considered as a single point gathering all the ground planes. It is placed between the power supply source and the PSoC device itself. Note that the V_{SSD} plane was hatched intentionally to distinguish between the V_{SSD} and V_{SSA} power planes.

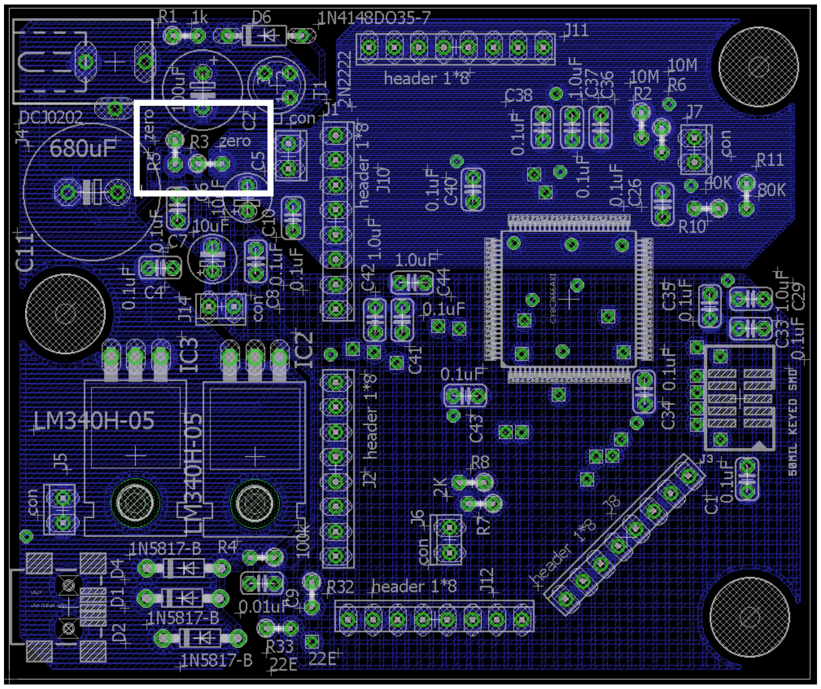


Figure 3-45 The second layer layout of the second PSoC 5lp based PCB board.

The third layer as shown in Figure 3-47 is the power layer. Compared with the first boards, we changed the power planes to be solid. They can be seen with the first layer superimposed.

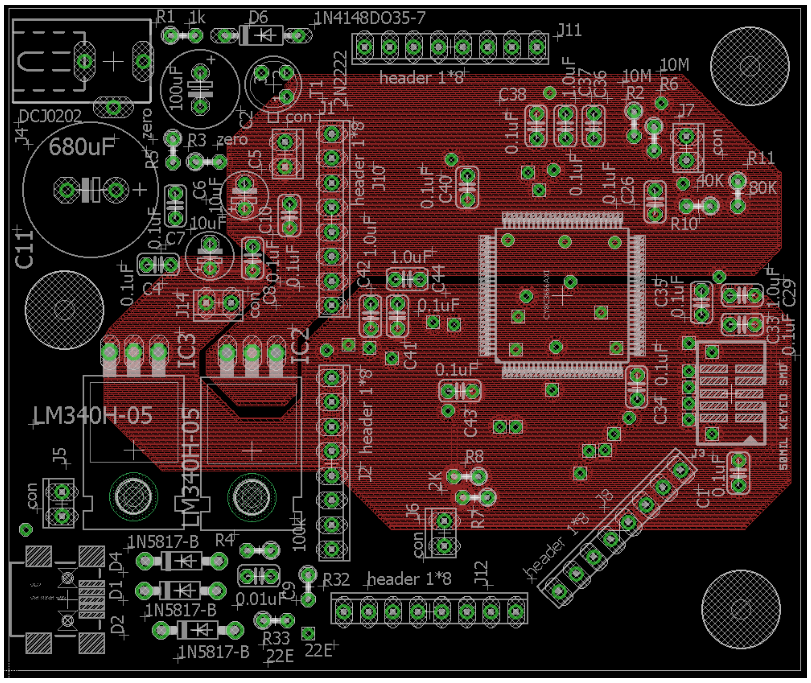


Figure 3-46 The second layer layout of the second PSoC 5lp based PCB board.

The fourth (or bottom) layer, as shown in Figure 3-48 is the digital layer housing the digital routes.

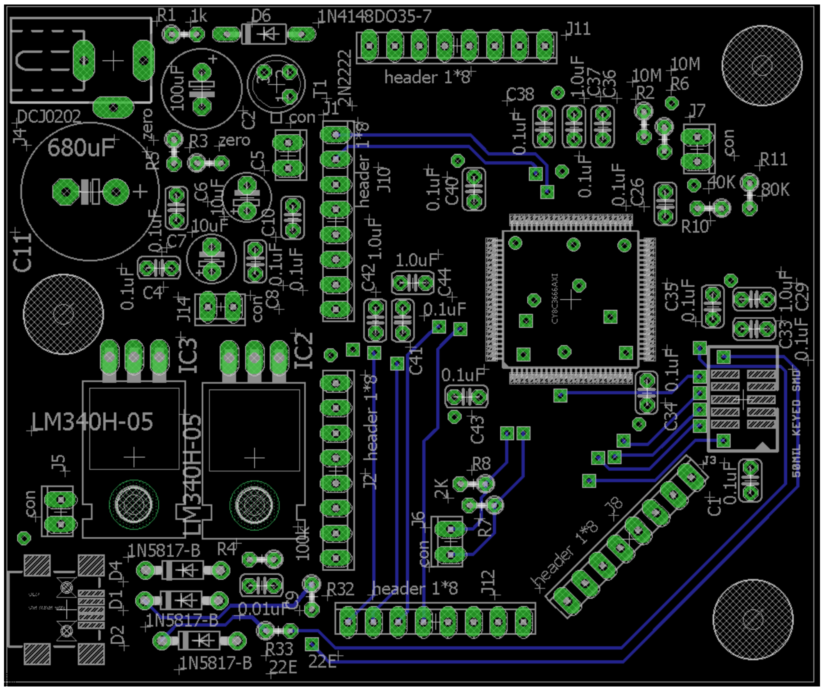


Figure 3-47 The bottom layer layout of the second PSoC 5lp based PCB board.

3.4.4 Implementation of a Two Board Sensors Signal Scavenging System

While we increased the input GPIO ports to be 40 on the second version boards, 40 sensor connections are still not enough for smart carpet sensor system. If there are 64 sensors needed to be scanned, two boards will be employed. However, all the sensor data has to be sent by one board using a USB connection; requiring two boards to communicate with each other. We implemented send and receive communication between boards using I2C communication protocol. The I^2C bus is an industry-standard, two-wire hardware interface. This requires assigning one board to be the master and the others to be slaves. PSoC 5LP chip integrates I^2C component which supports I^2C slave, master, and multi-master configurations.

We created two projects on PSoC Creator for both master board and slave board. In the schematic of master board as shown in Figure 3-49, an analog multiplexer with 32 channels first collects and selects sensor signal for process. A differential amplifier then amplifies the selected signal and subtracts the noise signal. An ADC converts analog signal to digital signals which are finally processed to sensor status arrays and stored in internal memory. The master board also receives the 32 sensor status arrays from the slave board. After the master board gets the 64 sensor status arrays, the microprocessor on master board converts them to their ASCII representation and USBUART sends them out to a PC or a raspberry-pi.

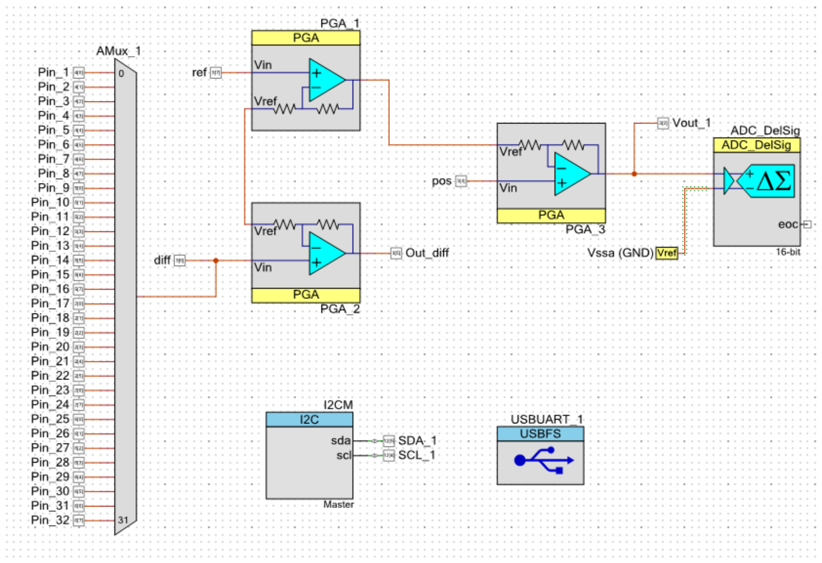
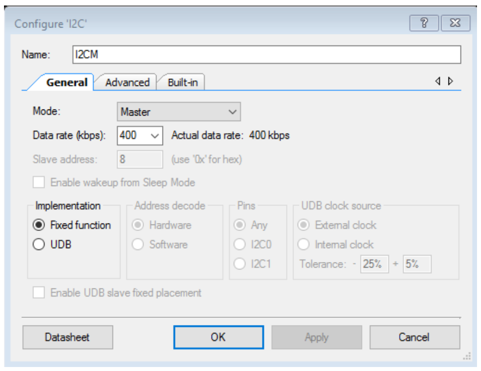


Figure 3-48 the schematic of master board. Master board can collect signals from 32 sensors and it also has USB communication port so that the processed data can be sent to a PC. I2CM is used to receive data from slave board.

The configuration of I^{2}C master module named as I^{2}CM is shown in Figure 3-50 below. The mode is master and the data rate is 400 kbps.



Configure 'I2C'

Name: I2CM

General | Advanced | Built-in

Mode: Master

Data rate (kbps): 400 Actual data rate: 400 kbps

Slave address: 8 (use '0x' for hex)

☐ Enable wakeup from Sleep Mode

Implementation: [✓ Fixed function] UDB

Address decode: [✓ Hardware] Software

Pins: [✓ Any] I2C0 I2C1

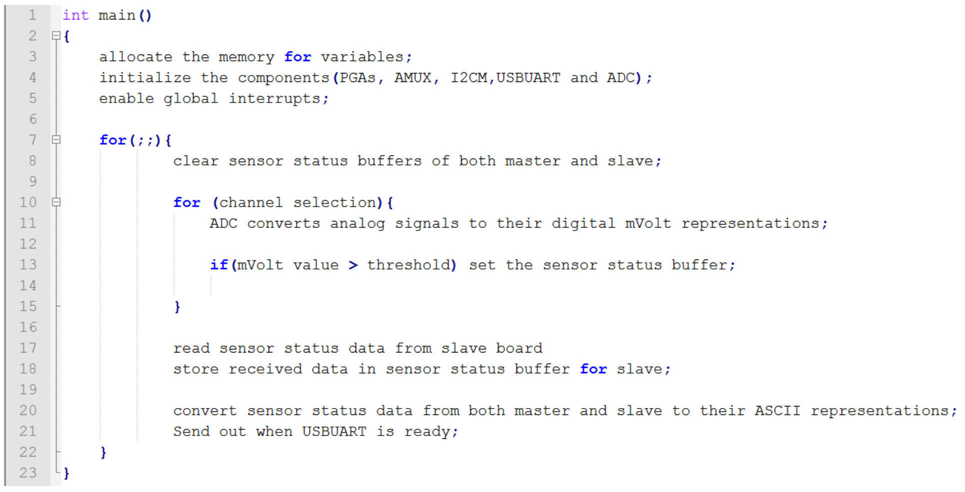
UDB clock source: [✓ External clock] Internal clock

Tolerance: - 25% + 5%

☐ Enable UDB slave fixed placement

Figure 3-49 The configuration of I2C master module. The data rate is 400 kbps.

To make the modules on master board work, microprocessor is required to control each module via program. We present the pseudo code in Figure 3-51 below. We first allocate memory for variables, initialize the components and enable global interrupts. Next one channel signal is chosen by AMUX and converted to digital value. Then we update the sensor status buffer by comparing the sensor digital value with a threshold value. After sensor status buffer is updated, master board will wait until it receives the sensor status array from slave. Finally, the processor on master board converts sensor status arrays both from master and slave to printable data and the USBUART module sends the data out.



```
int main()
{
    allocate the memory for variables;
    initialize the components(PGAs, AMUX, I2CM,USBUART and ADC);
    enable global interrupts;

    for(;;){
            clear sensor status buffers of both master and slave;

            for (channel selection){
                ADC converts analog signals to their digital mVolt representations;

                if(mVolt value > threshold) set the sensor status buffer;

            }

            read sensor status data from slave board
            store received data in sensor status buffer for slave;

            convert sensor status data from both master and slave to their ASCII representations;
            Send out when USBUART is ready;
    }
}
```

Figure 3-50 the pseudo code master board.

In the slave board project, slave board is also designed to collect 32 sensor signals, convert to digital sensor status array. The difference between the slave board and the master board is that the slave board will send the binary digital sensor status array to the master board without any further processing. The schematic of the slave board is shown in Figure 3-52.

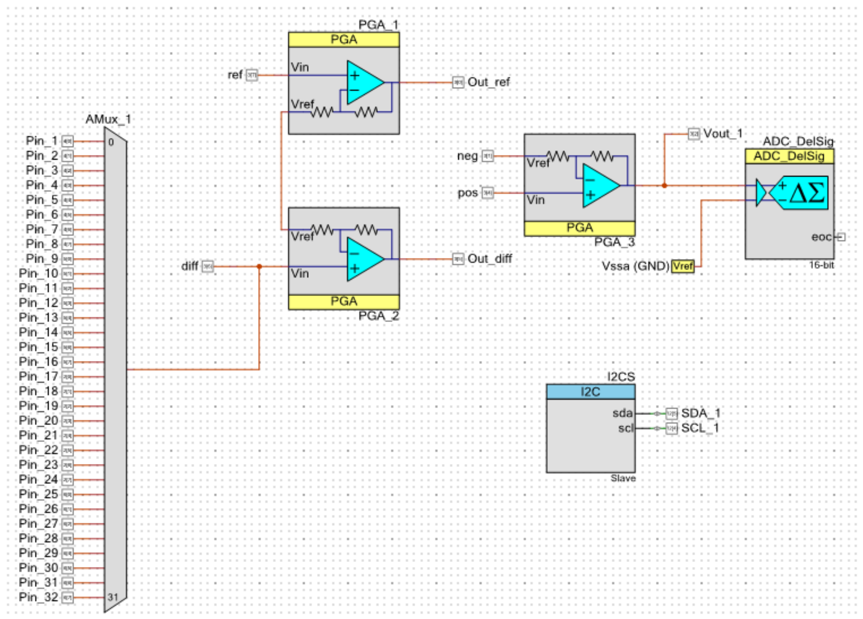
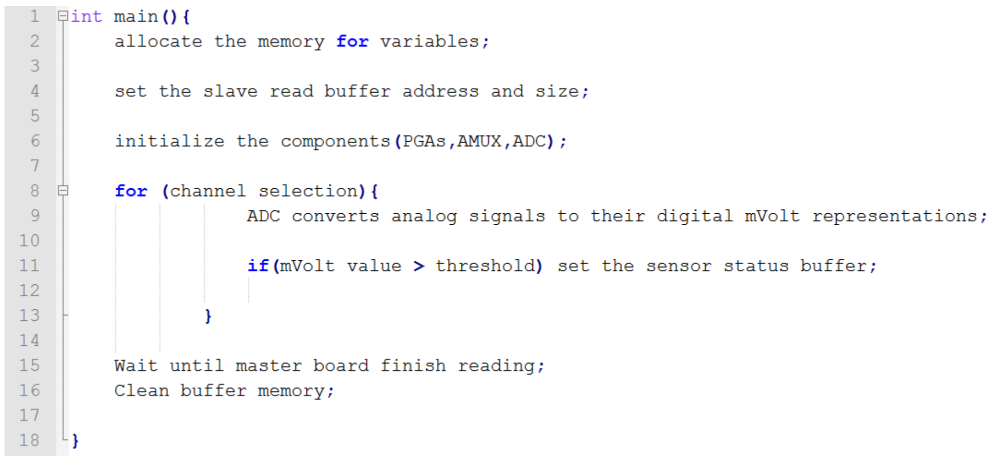


Figure 3-51 the schematic of the slave board. The slave board can collect signals from 32 sensors. I2CS is used to send data to the master board.

To make the modules on slave board work, microprocessor is required to control each module via program. After updating the sensor status of 32 sensors, the data will be directly sent out to master board. We present the pseudo code in Figure 3-53 below.



```
int main(){
    allocate the memory for variables;

    set the slave read buffer address and size;

    initialize the components(PGAs,AMUX,ADC);

    for (channel selection){
                ADC converts analog signals to their digital mVolt representations;

                if(mVolt value > threshold) set the sensor status buffer;

            }

    Wait until master board finish reading;
    Clean buffer memory;

}
```

Figure 3-52 the pseudo code master board.

3.4.5 Walking Experiment on Two Board System

We built up a two-board testing system by connecting 64 sensors (two segments) to the input ports of two boards. Each board is connected to 32 sensors (one segment) and connected to each other through I^{2}C buses. We set one board to be master board and the other to be slave board. The slave board is responsible to collect 32 sensors signals, process them to a sensor status binary array and send it to the master board. The master board is responsible to collect signals from the rest of 32 sensors and receive the data sent from slave board, convert all sensor status data to printable sensor frames and finally send the sensor status frames out to a Raspberry-Pi. Raspberry-Pi will send the sensor status frames to the cloud as long as it receives the data from PSoC two board system, so that we can access the data from the cloud anytime. The two board system is presented in figure 3-54.

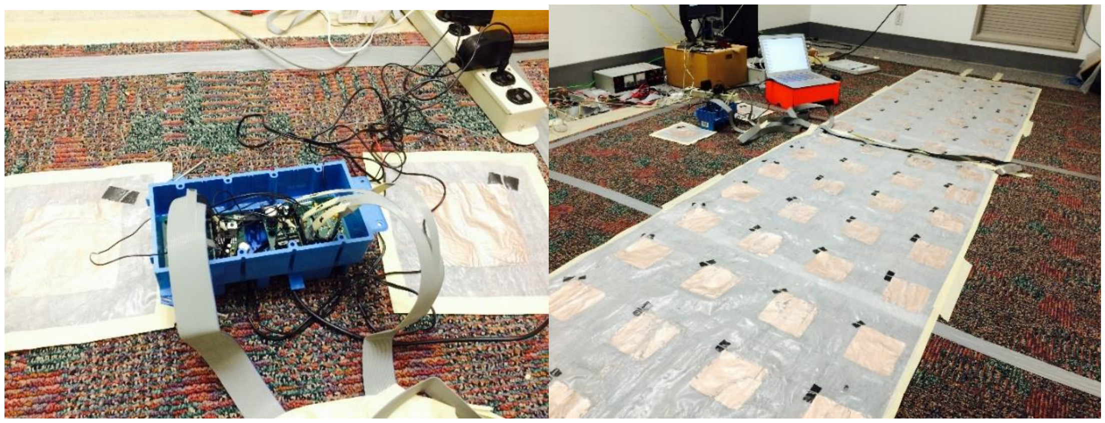


Figure 3-53 shows the 64-sensor signals scavenging system. The two individual sensors are used as reference sensor for each board differential amplifier. The two boards are powered by a 5-volt power adapter. Each board is connected to one sensor segment shown on the right.

To test the performance of the two-board system, we conduct a walking experiment to check if the activating sensor data stored in cloud matches the person's walking movement. Two volunteers (a male and a female) are asked to walk over the carpet shown in the Figure 3-54. They walked the full length of the segment(half round), two full segment lengths(one round) and finally four full segment lengths (two rounds) for several times.

One method we used to compare the cloud data and the walking movement is average velocity. During one walking experiment, we determined walking start time and stop time by a stopwatch. In the cloud, we record the frame start time and stop time. The velocity of a person walking can be calculate by Equation 11

$$Average\ velocity\ = \frac{The\ length\ of\ the\ carpet}{walking\ stop\ time - walking\ start\ time} \qquad \text{-Equation 11}$$

The velocity from data in cloud can be calculated by Equation 12

$$Average\ velocity\ from\ cloud\ data\ = \frac{The\ length\ of\ the\ carpet}{frame\ stop\ time - frame\ start\ time} \qquad \text{-Equation 12}$$

The second method we check the two boards performance is the average number of active sensors per step. So for each experiment, we counted the number of active sensors from cloud data, or we counted how many steps the volunteer walked. The average number of active sensors per step can be calculated by the Equation 13 below

$$average\ active\ sensors\ per\ step = \frac{\text{Numbers of active sensors}}{\text{number of steps}} \qquad \text{-Equation 13}$$

CHAPTER 4 RESULTS

4.1 Preliminary Studies for I/O and Sensor Addressing

Programmable System on Chip (PSoC) uses General Purpose I/O (GPIO) pins which are the entry for every input signal. Each GPIO pin can be programmed to send in the signals to its following modules. We initially conducted two experiments, first to confirm the programmable features of an input to an output, second to choose the input GPIO to read the input data.

4.1.1 Signal Input and Output

In the first GPIO experiment, we configured two analog GPIO pins on PSoC development kit [19] to connect an input pin to an output pin. In the test block diagram, Figure 4-1, input Pin1 has been programmed to GPIO port P3[2] connected to output Pin2 which is programmed to GPIO port P3[3]. A sensor is used as the input signal. The input and output signals, (Pin 1 and Pin 2) are both connected to oscilloscopes.

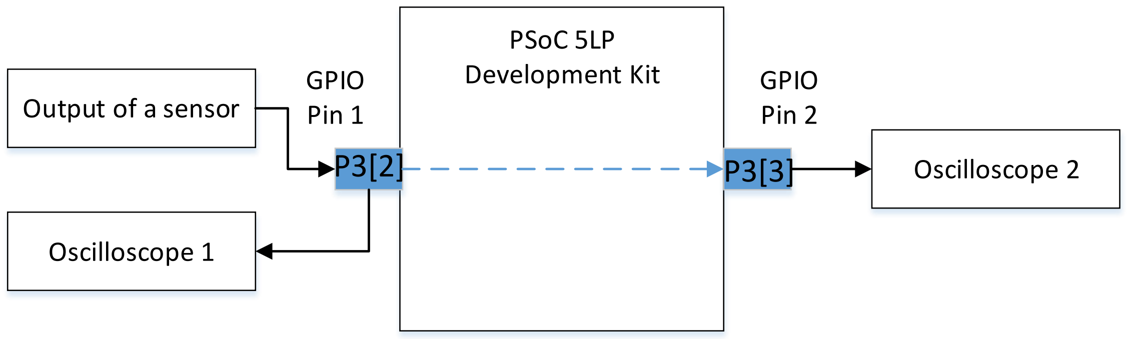


Figure 4-1 Testing the programming of GPIO pin P3[3] as input and pin P3[3] as output in the PSoC chip: Pin P3[2] acted as input for the sensor, and both pins were connected to oscilloscopes.

In Figure 4-2, the amplitude of output signal observed from Pin 2 arises from the input signal observed from Pin 1. The connection is linear and there is no difference in period. In addition, the sensor input signal produces the same output signal as shown in Figure 4-3. For our use, the GPIO pins on PSoC are adequate.

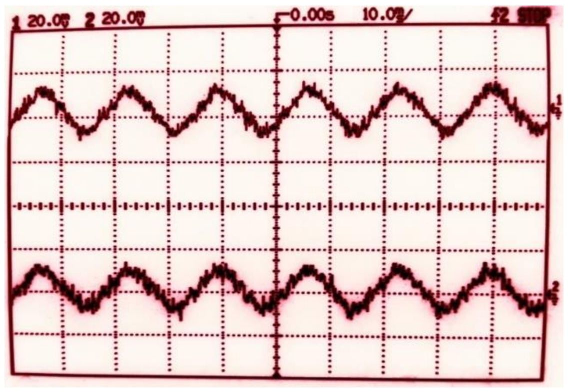


Figure 4-2 Analog GPIO Pin 1, the input signal of the sensor, and Pin 2, the signal out. They are slightly different and show the non-active sensor.

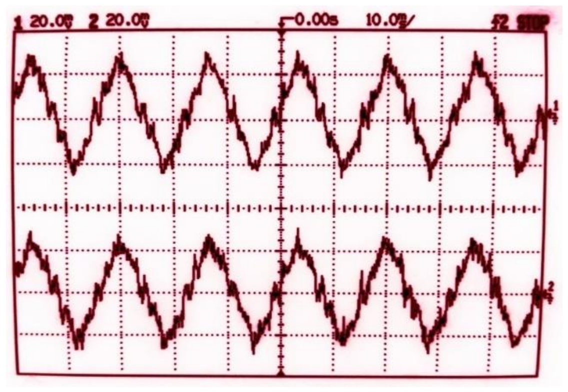


Figure 4-3 Analog GPIO Pin 1, the input signal of the sensor, and Pin 2, the signal out. They are the same and show the active sensor.

4.1.2 Sensor Addressing Demonstration

Smart Carpet system is used to scavenge signals from some number of sensors. We needed to use the PSoC system to provide data acquisition on a large number of sensor signals so this is required for multiple GPIOs, for which we designed the following experiment. We produced addresses to access a GPIO pin, which was presented to have a connection to sensors.

The block diagram of the experiment is shown in Figure 4-4. A 74163 chip is used to count a clock signal stored as a 4-bit number, which will then be demultiplexed to a 16-bit signal by a 74154 chip with one bit highlighted. For example, the 4 bit number output C_H produces an output of 74154 to 1110111111111111. As a result, the signal out of chip 74154 can be regarded as a sensor addressing signal to select one sensor at a time. We then fed the addressing signals to 16 GPIO pins of PSoC board.

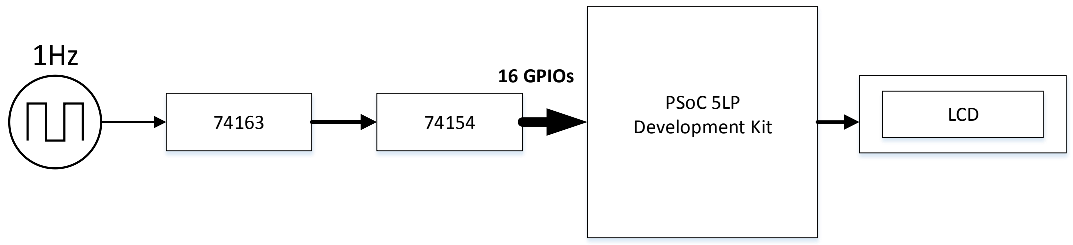


Figure 4-4 Block diagram of addressing experiment. A 74163 chip counts a clock signal stored as a 4-bit number, which will then be demultiplexed to a 16-bit signal by a 74154 chip with one bit highlighted. Next, the 16-bit signals are read by PSoC5lP chip through GPIOs and finally displayed by LCD.

The GPIO pin status can be shown on the LCD in the PSoC development board in Figure 4-5. Note that 1110111111111111 is displayed on LCD, which means the fourth most-significant bit has been selected. This is a clear demonstration that we can access PSoC GPIO pins, we can collect signals from multiple pins, and then we can control the input pins.

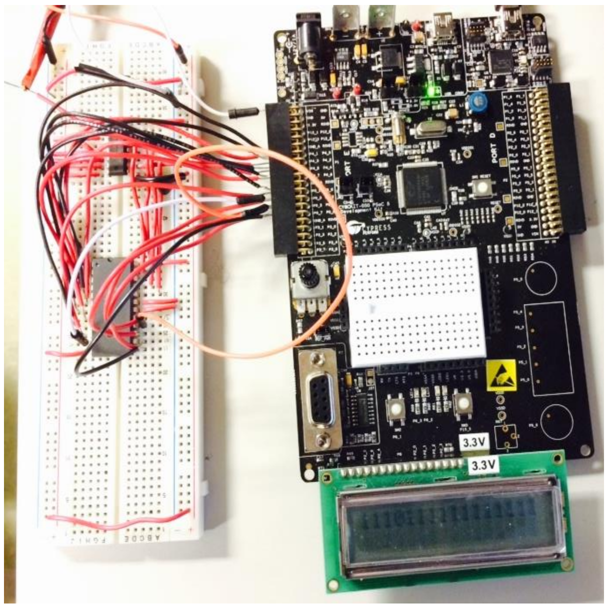


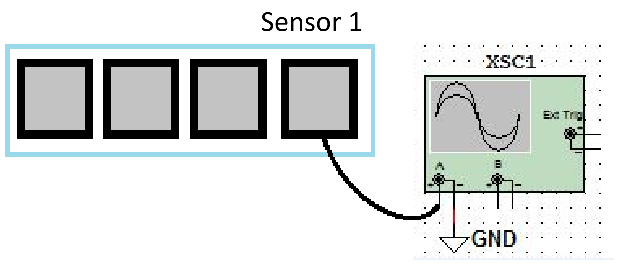
Figure 4-5 GPIOs are connected to 74154's outputs which will provide addressing signal. LCD displays the selected GPIO currently. 1110 1111 1111 1111 is shown in LCD currently.

4.2 Four Sensor Test System

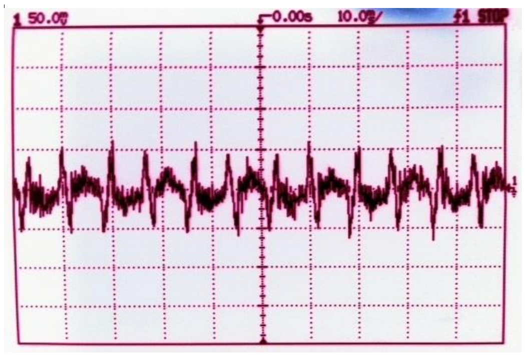
We tested the main modules of PSoC system using a four-sensor array. The sensors have the same design as those used in the Smart Carpet system.

4.2.1 Testing the Analog GPIO Pins

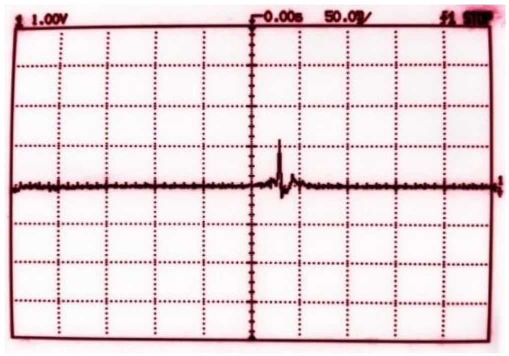
The four-sensor array produces one signal from each of the 4 sensors. These will be invoked as the input signal for testing the design of PSoC5LP system. This experiment shows the output of this four-sensor array whether active or not entering and exiting the PSoC5LP chip.



(a)



(b)



(c)

Figure 4-6 (a) shows sensor1 is connected to the Oscilloscope. When sensor 1 is not active, the output signal from the sensor is a period signal about 100 millivolts amplitude and 17 millisecond period shown in (b). When the sensor 1 is triggered by feet and active, the output signal is shown in picture right. A pulse with about 1volt amplitude appears shown in (c).

To build up the test circuit, we connected an oscilloscope to one sensor, say sensor 1 in Figure 4-6(a). We set the display mode of oscilloscope to be 50 millivolts per grid vertically and 10 milliseconds per grid horizontally. If sensor 1 is not active, we can see a waveform with about 100 millivolts amplitude and 17 millisecond period shown in Figure 4-6 (b). For an active sensor, we measured the sensor signal triggered by feet. There is an obvious pulse appearing with amplitude of 1000 millivolts shown in Figure 4-6 (c). Note that the vertical gain is at 1 volt per grid and horizontal gain is 5 milliseconds per grid. Hence the much larger signal appears muted. The experiment repeated for the other 3 sensors with different GPIO pins. Presenting in the important demonstration of the programmability of the PSoC5LP pins, we were now ready to have a 4 sensor PSoC based detector under program control.

4.2.2 Analog Multiplexer Module Selects Multiple Sensor Signals

The resource in PSoC5LP chip is limited, for example only four operational amplifiers and an ADC integrated in PSoC5LP chip. This requires a multiplexer to select one sensor, and we may require many. The PSoC5LP has 2 multiplexers which have the abilities to access 64 sensors.

To test the functionality of analog multiplexer integrated in PSoC5LP chip, we used the four-sensor array, and set the channel of analog multiplexer to four. The input pins of analog multiplexer on the chip are assigned to 2[0]-2[3], which are also connected to four sensor outputs shown in figure 4-7. The output pin of multiplexer 3[6] is connected to an oscilloscope.

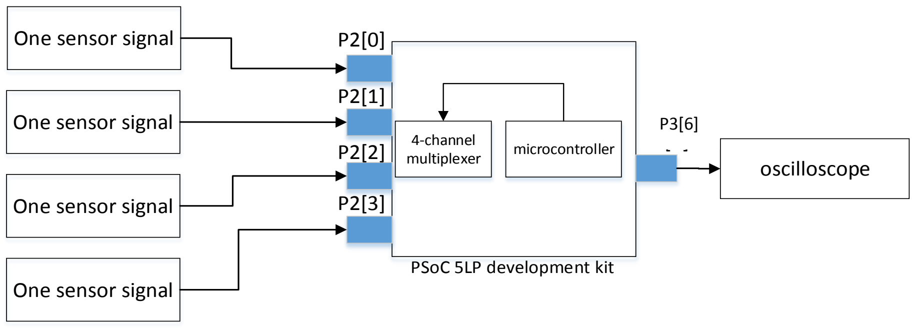
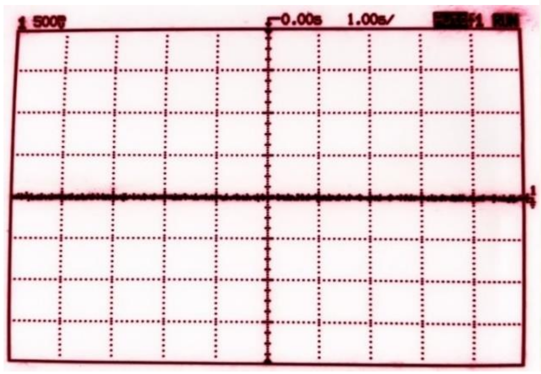
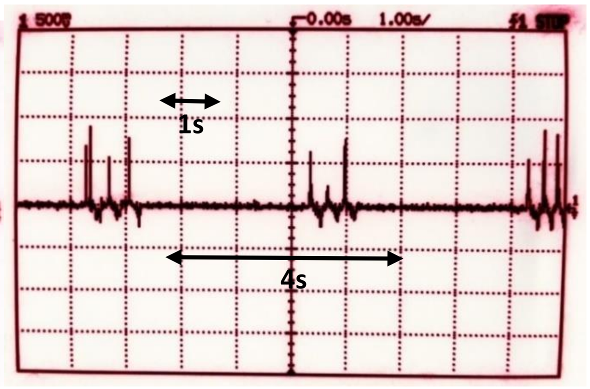


Figure 4-7 Block diagram of multiplexer experiment. Each sensor on four-sensor array is directly connected to each of the GPIOs 2[0]-2[3] on PSoC 5LP chip. Microcontroller inside the chip controls the multiplexer on choosing the channels. The signal of selected channel is sent out from P3[6] and displayed on an oscilloscope.

We programmed the selection time period of each channel in the multiplexer to 1 second. In the experiment, we activated only one sensor on the four-sensor array. The oscilloscope trace in Figure 4-8 (a) shows the high amplitude of the active sensor signal occurring every 4 seconds. In the figure 4-8 (b) the non-activated sensors are clearly in the noise.

(a) (b)

Figure 4-8 Microprocessor collects one GPIO's signal, also one channel of multiplexer, for 1s at a time. When only one sensor is active, the high-voltage signal comes once every 4 seconds shown in (b).

4.2.3 Simulation of Two-Programmable Gain Amplifier Differential Amplifiers

In the system, the module following by the analog multiplexer is a differential amplifier. A differential amplifier helps distinguish the sensor signal from the noisy background by subtracting the input signal with a reference signal.

We first designed a differential amplifier using 2 programmable gain amplifiers (PGAs) with a gain of 3. After the differential amplifier program is loaded to the PSoC5LP development board, we fed one sensor output from the 4-sensor array to the differential input and fed a single sensor output to the reference input. The output of differential amplifier can be observed by an oscilloscope. The block diagram of this experiment is shown in the Figure 4-9. We also connected a 10 M Ohm resister to each input of differential amplifier (P5[0] and P2[0]) and the resister then connects to the ground. Because in our previous work, we showed the impedance of using a 10 M Ohm input resistance for preserving the sensor signal fidelity. Low impedance collapses the voltage since the current rearrangement caused by adding an object to the sensor in values.

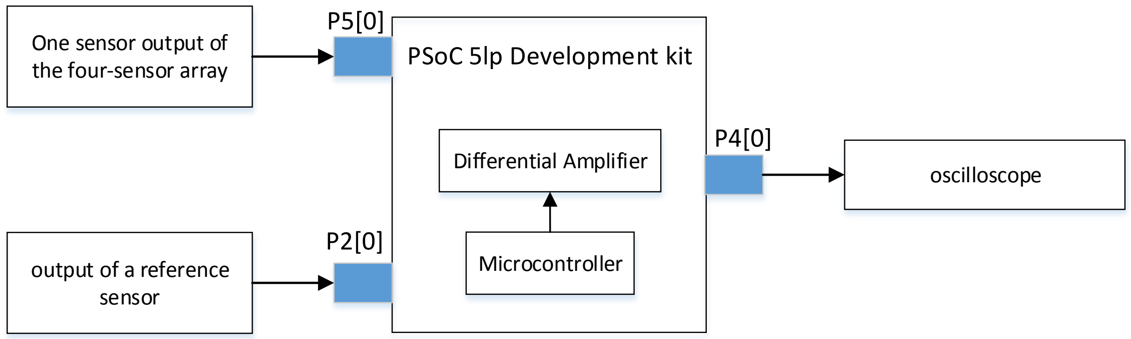


Figure 4-9 Block diagram of 2-PGA differential amplifier experiment. GPIOs P5[0], P4[0] and P4[0] are programmed to connected to the differential amplifiers' differential input, reference input and output. We connected one sensor from 4-sensor array to one GPIO input P5[0] and a single sensor to a GPIO input P2[0]. An oscilloscope is connected to GPIO P4[0] to display the output of this differential amplifier.

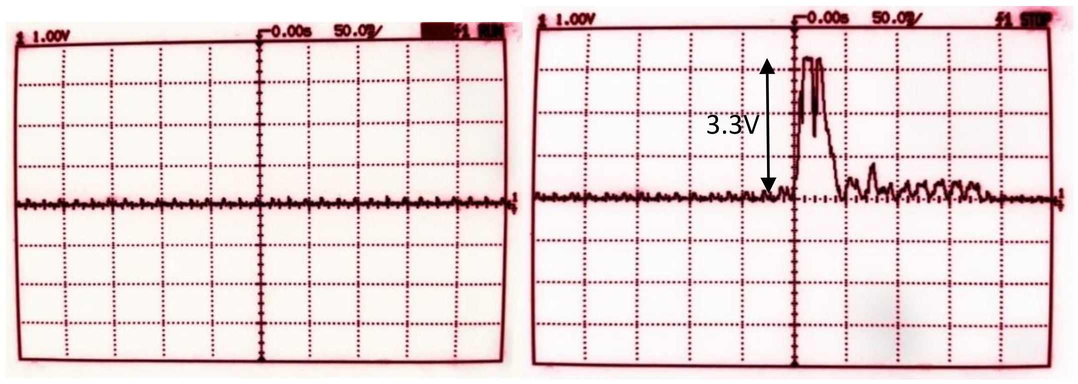


Figure 4-10 experiment results of a 2-PGA differential amplifier with non- symmetrical structure. The picture left shows the output of a non-active sensor. The picture right shows the output of one active sensor.

Figure 4-10 shows the output signal of differential amplifier captured from the oscilloscope. The left one shows the output of differential amplifier when the selected sensor is not active. When the selected sensor is triggered by feet, there is a pulse with about 3.3 volts amplitude shown on the oscilloscope.

4.2.4 Implementation of an ADC and UARTUSB Modules

In the system, after the sensor signal noise has been reduced and the signal amplified, we then converted it to a digital signal by an ADC module, pre-processed it by a microprocessor and transferred it by a USBUART module. To convert analog to digital, we sampled signal, which provides convenience on processing, storing and transferring data. We set the ADC resolution to be 16 bit. The microprocessor converts the binary signal to printable characters, then the USBUART sends the data to a computer using USB port. That data is also displayed on HyperTerminal on computer. The block diagram of this experiment is shown in Figure 4-11. As discussed in method, the binary signal is actually converted to an intermediate data, then converted to ASCII.

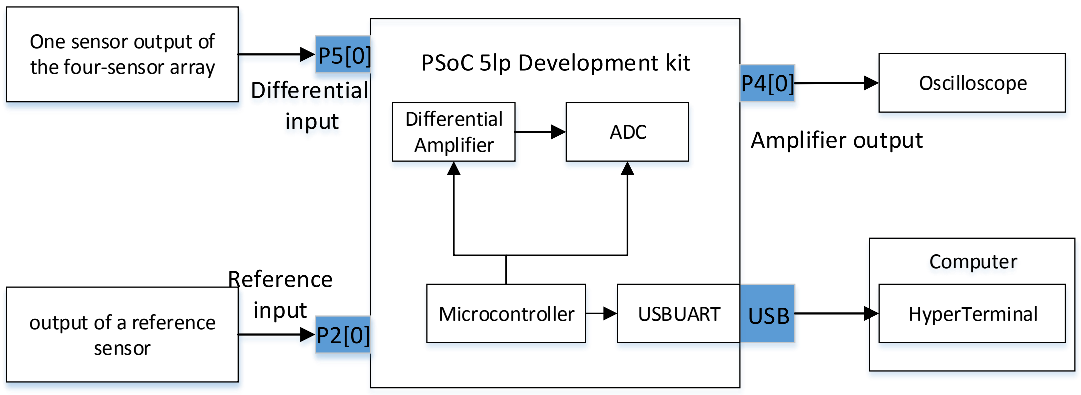
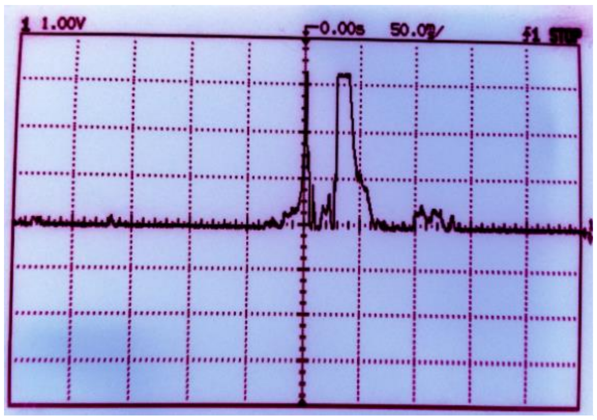
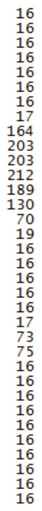


Figure 4-11 Block diagram of ADC and UARTUSB modules experiment. An ADC is programmed to convert output signal from differential amplifier to digital voltage numbers which is sent to a computer via USB controlled by USBUART module in PSoC5lp chip. Microcontroller controls the differential amplifier module, ADC module and USBUART module during this process.

In order to compare the signal sent into ADC with the data displayed on HyperTerminal, we also connected an oscilloscope to output pin of differential amplifier. The results are shown in Figure 4-12. The waveform of an active sensor signal after being amplified is displayed in the picture left. The amplitude of that signal is about 3.3 volts, which also reflects the number shown in HyperTerminal shown in the picture right. The picture in the middle presents the voltage values in millivolts when sensor is non-active.




```
16
16
16
16
16
16
16
17
164
203
203
212
189
130
70
19
16
16
16
16
16
17
73
75
16
16
16
16
16
16
16
16
16
16
```

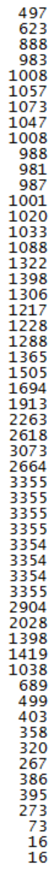

```
497
623
888
983
1008
1057
1073
1047
1008
988
981
987
1001
1020
1033
1088
1322
1398
1306
1217
1228
1288
1365
1505
1694
1913
2263
2618
3073
2664
3355
3355
3355
3355
3354
3354
3354
3355
2904
2028
1398
1419
1038
689
499
403
358
320
267
386
395
273
73
16
16
```

Figure 4-12 The waveform of differential amplifier and the voltage values in millivolts shown in HyperTerminal. The amplitude of the pulse triggered by an active sensor after being amplified can be as high as about 3.3 volts shown in the picture right. This is also matches the voltage values from output of ADC shown in the picture right. The picture in the middle shows the voltage value of a non-active sensor.

4.3 Single Board System

4.3.1 32-Sensor-Signal Scavenging System

We've shown the operation of main modules for the Smart Carpet PSoC based hardware system. The results demonstrate that we can use PSoC 5LP based system to scavenge the sensor signals for the Smart Carpet. We now show module integration to create the system in Figure 4-13, which is a 32 sensor personal detection system.

The input signals are from a sensor array of 32 sensors integrated into an array for use in the Smart Carpet system [15]. 32 GPIOs on the PSoC5LP development kit collect the signal from every sensor, and each sensor is selected by a 32-channel programmed multiplexer in the PSoC. The selection period for each channel is 5 milliseconds (also programmed), which means the signal scavenging time for one sensor at a time is 5 milliseconds. The selected sensor signal is amplified by two-PGA implemented differential amplifier with gain 3. P2[0] is assigned to reference signal input, provided by a single sensor. We also connected 10 mega ohm resistances to the inputs of the amplifier to match the input impedance of the sensors. An ADC converts the amplifier analog signals to digital signals. Incidentally the programming of the PSoC allows us to replace many chip with the PSoC.

We are interested in the status of sensors and the location. Sensor status is either active or non-active, where activation is represented by 1 bit binary number. The microcontroller in the PSoC converts digital signal to voltage which we compares with 800 millivolts, a set threshold. Clearly a voltage value of the sensor greater than threshold is set to 1 (active) by the microcontroller. Otherwise status number is 0 (non-active). For the location of sensors, the microcontroller arranges the status numbers of 32 sensors to an 8 byte hexadecimal representation. These are converted to ASCII numbers and the USBUART module sends out the ASCII numbers through USB to the PC. The ASCII numbers representing the sensor status and location are finally displayed in software HyperTerminal in a computer.

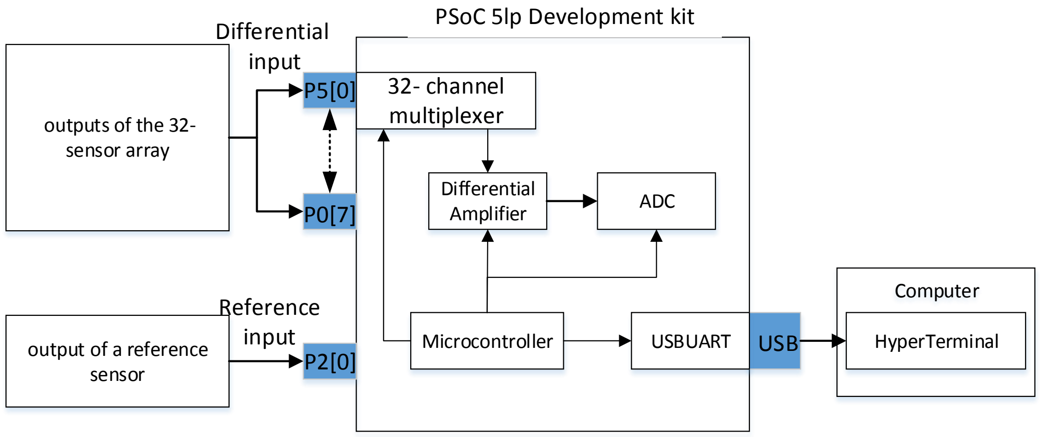
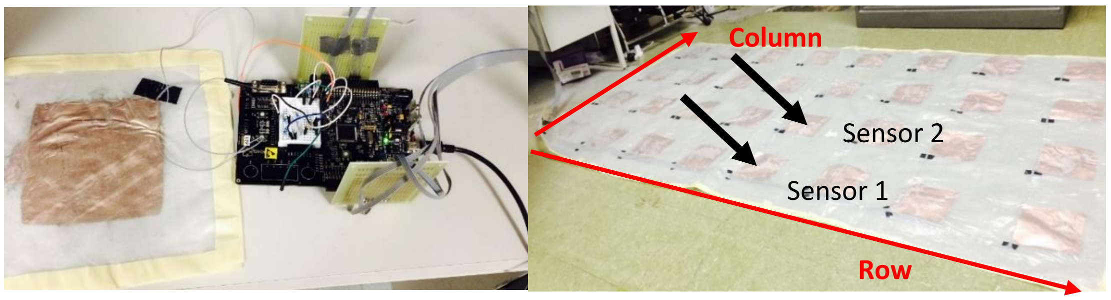
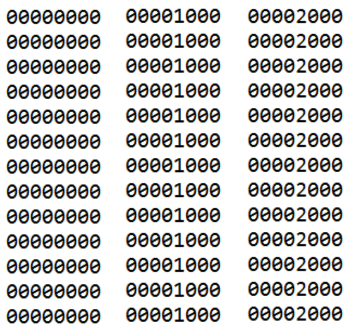


Figure 4-13 The block diagram of 32-sensor signal scavenging system.32 sensors are connected to 32 GPIOs on PSoC5LP development kit. These 32-channnel signals are selected by a multiplexer, amplified by a differential amplifier and converted to digital signal by an ADC. Finally, microcontroller converts the ADC output to printable strings, which are sent out to computer through USB.



(a)



```
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
00000000  00001000  00002000
```

(b)

Figure 4-14(a) left shows the reference sensor connects to the purchased development kit. The white section is proto board showing the circuitry developed to support the functionality. (a) right shows the segment of 32-sensor signals for the scavenging system. (c) shows three columns. On the left shows no activated sensor. The results of tapping one sensor at a time are shown in the middle (sensor 1) and one the right (sensor 2).

This 32-sensor system implemented on the development kit is presented in Figure 4-14 (a). The results received by HyperTerminal are presented in Figure 4-14 (b). For no activated sensors, the status is 0 shown in Figure 4-14 (b) left. Activating sensor 1 in the sensor segment, which is in the 5th row and the 1st column in Figure 4-14 (a) right, sets the sensor's status flag to 1. The hexadecimal representation is 00001000 shown in middle of the Figure 4-14 (b). While activating sensor 2 produce the results shown in the Figure 4-14 (b) right.

To be sure, we have simplified the representation which shows the repetitive activation of the sensor in its column. The activated sensor is indeed 1, non-activated sensors are zero, hence, the middle column of data has 7 zeroes and 1 one. The 1 corresponds to the 5th row in the 1st segment column. IN a scan of all sensors in the segment, The status flag array turns to be 0000 0000 0000 0000 0001 0000 0000 0000 for activating sensor (5,1). Note that in Hexadecimal the number is 0000 1000.

4.3.2 Board Construction and Test of 32-Sensors System: First Board

After having the whole system tested by development kit, we designed our first PCB board. We need to include power supply, USB communication, programming, peripheral circuits for the PSoC chip, and support for the differential amplifier.

The program used on our first board is the same as the one on the development kit. The block diagram of this experiment is shown in Figure 4-15.

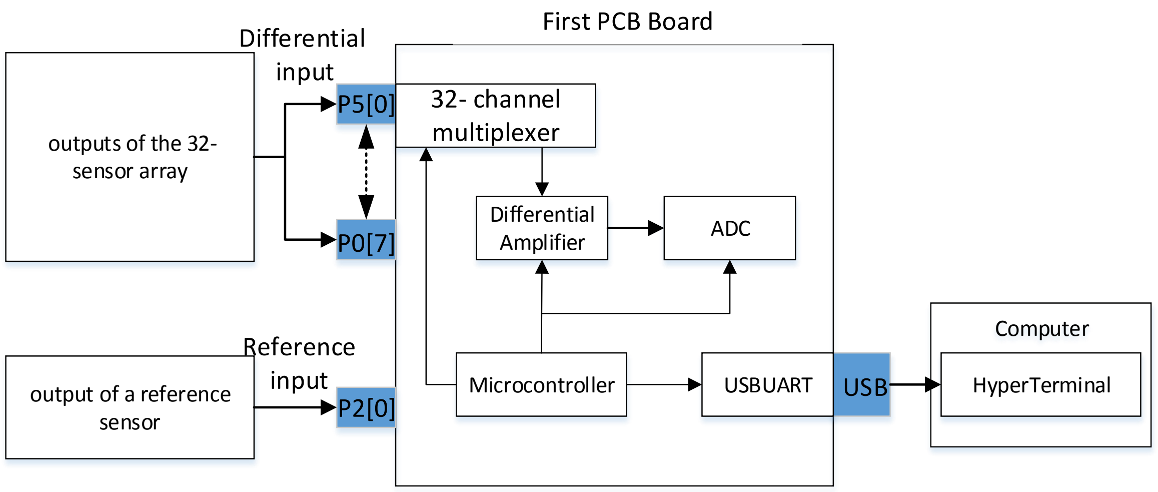
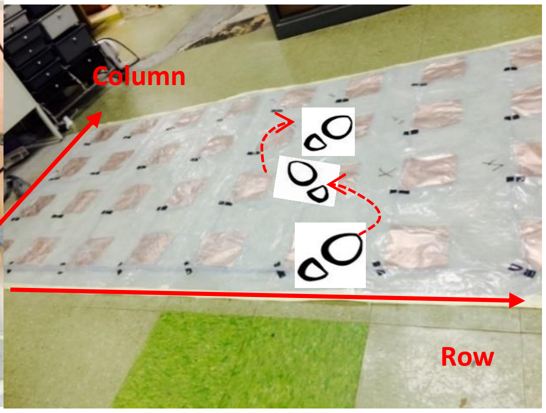


Figure 4-15 Block diagram of 32-sensor system experiment on the first PSoC5LP based PCB board. We loaded the same program used in development kit to our first PCB board. 32 sensor signals are sent into the board through 32 GPIOs. The sensor activity results are displayed on HyperTerminal in a Computer.

This 32-sensor system implemented on our first PSoC5LP based board is presented in Figure 4-16 (a) and (b). The results received by HyperTerminal are shown in Figure 4-16 (c). Again if no activation, the sensor status is set to 0 shown in the Figure 4-16 (c) left. We then activated the sensor by letting someone walk the path shown in Figure 4-16 (b), the result is demonstrated in the Figure 4-16 (c) right. The first step activates the sensor on the 6th row and the 1st column. The ASCII combination string is $0000\ 0100_{16}$. When he stepped on the second sensor in the 6th row and the 2nd column, the ASCII combination string $0000\ 0020_{16}$. String $0000\ 0040_{16}$ appeared for the activating sensor in the 6th row and the 3rd column.



(a) (b)

```
00000000  00000000
00000000  00000000
00000000  00000000
00000000  00000100
00000000  00000100
00000000  00000100
00000000  00000100
00000000  00000100
00000000  00000000
00000000  00000300
00000000  00000000
00000000  00000200
00000000  00000200
00000000  00000200
00000000  00000200
00000000  00000200
00000000  00000000
00000000  00000400
00000000  00000400
00000000  00000400
00000000  00000400
00000000  00000400
00000000  00000400
00000000  00000000
```

(c)

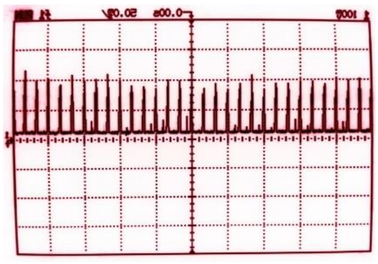
Figure 4-16 We designed constructed and tested a Printed Circuit Board (PCB) to receive data from 32 sensors using a PSoC5LP system on a chip shown in (a) and (b). (c) shows results of this PCB board system. The result of no activated sensor is shown left. The results of walking on one sensor at a time are shown on the right, and are expressed in Hexadecimal.

4.4 Board Construction and Test of Two-Board Signal Scavenging System: Second Board

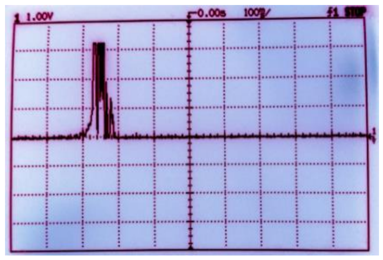
We can improve the first PCB board system for two reasons. The first reason we did not use the function that provides for multiple boards communication. The second reason is to improve the performance of the 2-PGA differential amplifier. Thus, we made the second PCB board with multi-board communication module integrated and improved the differential amplifier performance by designing a three PGA differential amplifier.

4.4.1 Three-PGA Differential Amplifier with Symmetrical Structure

Figure 4-17 below shows noise and signal of the two-PGA differential amplifier. We use with the setup shown in Figure 4-11. When the sensors are not active, the waveform from the output of the differential amplifier is shown in the Figure 4-17 (a). This is noise with a minimum value is 16 millivolts, but can reach 200 millivolts. Similarly connecting the input to the reference and stepping on a sensor, produces a large signal shown in the Figure 4-17 (b). Even if the differential input is connected to the reference input pin, the result is not 0, so this differential amplifier can't provide differential isolation.



16
17
164
203
203
212
189
130
70
19
16
16
16
16
16
17
73
75
16



(a) (b)

Figure 4-17 some features of asymmetrical structure differential amplifier

We designed a new differential amplifier which has a symmetrical structure with 3 programmable gain amplifiers. To test the performance of this new differential amplifier, we conducted an experiment on the PSoC5LP development kit. The difference input is fed by one sensor output of a 4-sensor array and reference input is fed by a single sensor. The amplifier output can be observed by an oscilloscope as well as HyperTerminal on a computer. The block diagram of this experiment is illustrated in Figure 4-18.

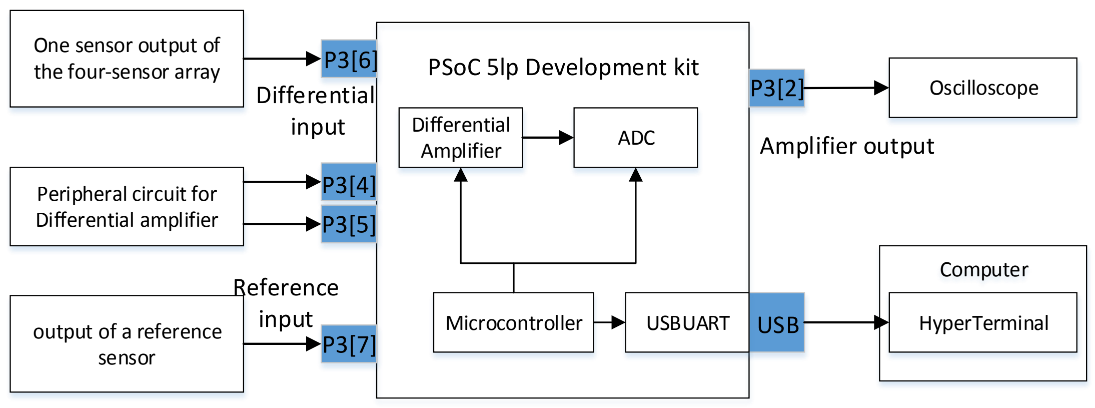
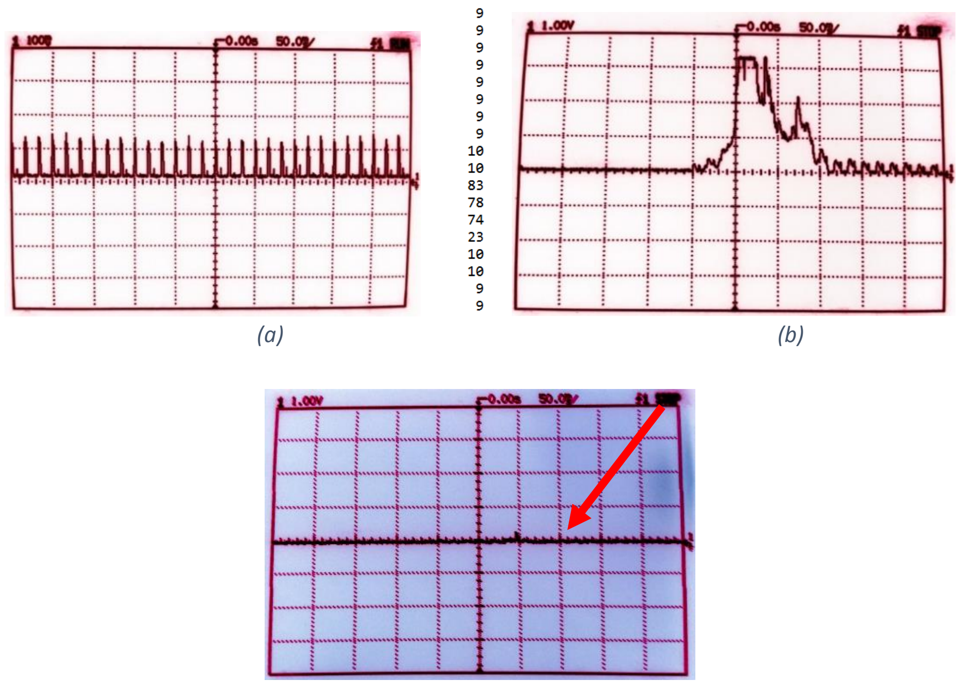


Figure 4-18 Block diagram of a 3-PGA differential amplifier experiment. To build up the circuit of this new designed differential amplifier, some off-PSoC chip components are required. So we assigned another 2 GPIO pins to connect the PSoC kit with peripheral circuit of differential amplifier. One sensor output from four sensor array is used as differential input. A single sensor is used as reference input. To compare the output from amplifier with the data sent to computer, we also connected an oscilloscope to amplifier output.

Figure 4-19 shows some outputs of 3-PGA differential amplifier. When the sensors are not active, the waveform from the output of this differential amplifier is shown in the Figure 4-19 (a). The minimum value is 9 millivolts, but the high is about 80 millivolts much lower than the 2-PGA case. The noise decreases by half compared with the 2- PGA differential amplifier. The output signal from an active sensor is illustrated in Figure 4-19 (b). The maximum voltage of this waveform is about 3.3 volts. Figure 4-19 (c) shows the result of differential input pin connected to the reference input pin. Whether the sensor is active or not, there is no obvious pulse appearing shown in Figure 4-19 (c), but the arrow shows the minor result of a step on a connect sensor. Thus the newly designed amplifier has better performance than the 2-PGA differential amplifier.



(c)

Figure 4-19 experimented results of a three-PGA differential amplifier with symmetrical structure. (a) shows the output the non-active sensor. The maximum voltage is about 80 millivolts and the minimum voltage is about 9 millivolts. (b) shows the output of one active sensor. The maximum voltage is about 3.3 volts. (c) shows the output of an active sensor when differential amplifier's different input is connected to reference input.

4.4.2 Board Construction and Test of 64-Sensors Signal Scavenging System

To improve system performance, we used more GPIO inputs and the I2C module integrated on the chip. We designed and manufactured another two boards. Each board has 40 GPIO input pins available for sensors.

Since one piece of sensor carpet array has 32 sensors, we assigned one board to collect sensor signals from one piece of sensor array. With the help of I^2C module, these two boards can work simultaneously. One is used as master and the other is as a slave. Master board will collect all the data representing 64 sensor status and send to a computer.

The block diagram of this 64-sensor system is shown in Figure 4-20 below. Different from the first board, the peripheral circuits for I2C and 3-PGA differential amplifier are implemented on the boards.

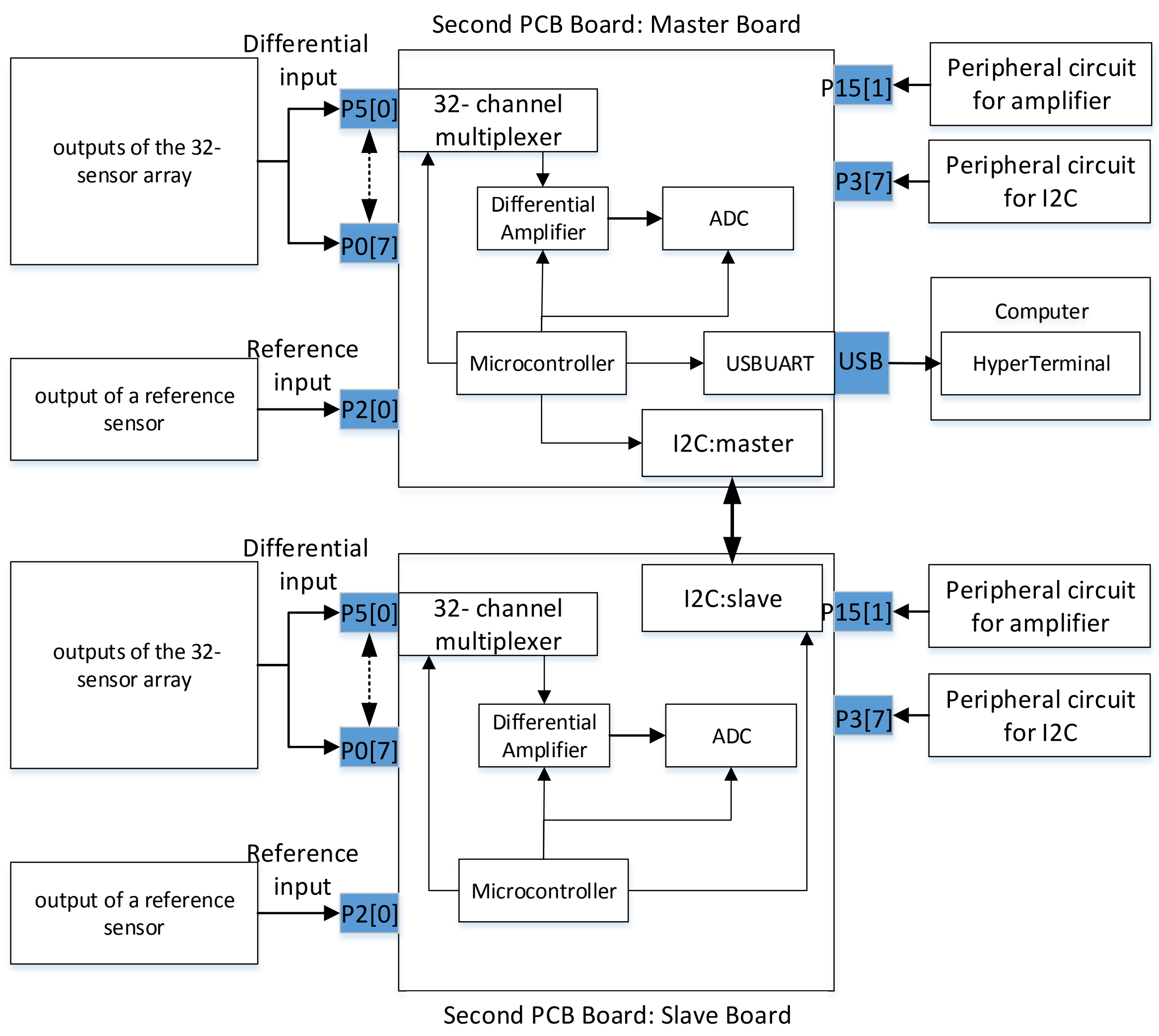
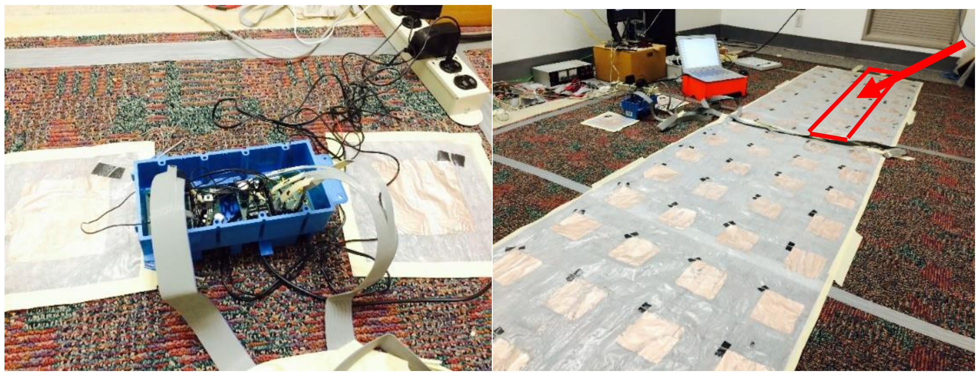
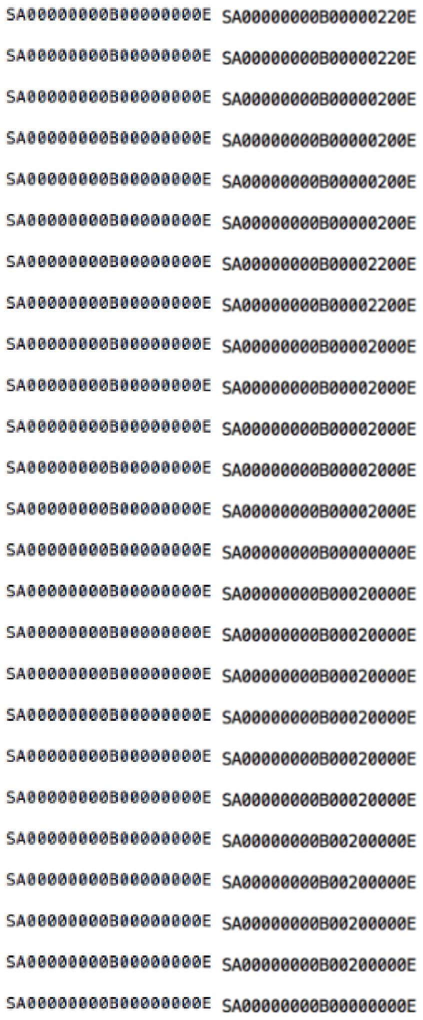


Figure 4-20 Block diagram of the second PCB boards system. Each board scavenges 32 sensors through 32 GPIOs. The sensor signals are selected, amplified, converted to binary signals. Master board collects the binary sensor data through I2C bus controlled by I2C module and converts 64 sensor data to printable data and sends to software part through USB.



(a)



```
SA00000000B00000000E SA00000000B00000220E
SA00000000B00000000E SA00000000B00000220E
SA00000000B00000000E SA00000000B00000200E
SA00000000B00000000E SA00000000B00000200E
SA00000000B00000000E SA00000000B00000200E
SA00000000B00000000E SA00000000B00000200E
SA00000000B00000000E SA00000000B00002200E
SA00000000B00000000E SA00000000B00002200E
SA00000000B00000000E SA00000000B00002000E
SA00000000B00000000E SA00000000B00002000E
SA00000000B00000000E SA00000000B00002000E
SA00000000B00000000E SA00000000B00002000E
SA00000000B00000000E SA00000000B00002000E
SA00000000B00000000E SA00000000B00000000E
SA00000000B00000000E SA00000000B00020000E
SA00000000B00000000E SA00000000B00020000E
SA00000000B00000000E SA00000000B00020000E
SA00000000B00000000E SA00000000B00020000E
SA00000000B00000000E SA00000000B00020000E
SA00000000B00000000E SA00000000B00020000E
SA00000000B00000000E SA00000000B00200000E
SA00000000B00000000E SA00000000B00200000E
SA00000000B00000000E SA00000000B00200000E
SA00000000B00000000E SA00000000B00200000E
SA00000000B00000000E SA00000000B00000000E
```

(b)

Figure 4-21 Picture (a) and (b) show the 64-sensor signals scavenging system and its result. The result of nobody walking is shown in picture (b) left. The result of walking on one column sensors is shown in the picture (b) right.

The performance is presented in Figure 4-21 above. We installed two PCB boards in a thick plastic electrical box. The power wire and USB wire can access the pins on the boards through the holes on the box. 64 GPIOs on two boards are connected to two pieces of sensor array. The ASCII strings can be sent to either a computer or a raspberry pi. The system connection is presented in Figure 4-21 (a). No activation produces the data shown in the Figure 4-21 (b) left. We tested the system by walking from the very right of the carpet in the first column to the left and results are shown in the Figure 4-21 (b) on the right. Each frame begins with S, followed by segment A then B and finished with E. Normally, these data are sparse since the rate of data collection is on the order of 10s of milliseconds. A human single step may take 10s of millisecond.

4.4.3 Reduction of System Size and Cost

With the help of PSoC5LP chip, many hardware components have been integrated into one chip; we gain benefits on the number of chip, size of system and the system cost.

Since the main digital and analog modules implemented by traditional IC chips in previous Smart Carpet system has been replaced by one PSoC chip, the number of chips has been efficiently reduced. Table 4-1 shown below lists the IC chips and components for both previous system implemented by traditional chips and the PSoC based system processing 32 sensor inputs signals.

From the table 4-1, we can see the two advantages of applying PSoC to smart carpet system with respect of chips and components numbers. The chips for processing sensors data have been reduced. The PSoC chip, CY8C5868AXI-LP035Y, provides the functionalities of the microprocessor, opamp, multiplexer, logic level converter and ADC, which are the main components for processing the sensor signals, so that the number of chips is reduced from 17 to 1. And second, the total number of components has been reduced from 98 to 55. Straightforwardly a straight component consolidation to a single chip is valuable. Less obvious is the reduction in resistors. The impendence of sensor signals (10 M ohms) must match the inputs of multiplexers. The previous system placed parallel resisters between each sensor output and the multiplexer input, needing 32 resisters. If we increase the number of sensors, we need the same number of resisters, which can be really large. However, experiments showed that impedance matching can occur between the sensor output and the input to the programmable gain amplifier. So we can gain from reducing tens of resistors.

Table 4-1 The list of chips and components of system implemented using traditional IC chips and PSoC based system.

Components on Last version board	Quantity	**Components on PSoC based board**	**Quantity**
*PIC18F4455 microprocessor	1	*CY8C5868AXI-LP035	1
*TL084 op-amp	9	LD1117V33 regulator	2
*CD4051 multiplexer	4	1N5817-B diode	3
*MAX232 logic level converter	1	1N4148DO35-7 diode	1
*MCP3001 ADC	1	2N2222 transistor	1
*NMH0509 DC-DC converter	1	Resisters	12
7805 regulator	1	Capacitors	24
7905 regulator	1	Headers	9
Resistors	47	50MIL KEYED SMD	1
Capacitors	10	DCJ0202 jack	1
DIP sockets	17		
20MHz oscillator	1		
DIP switch	1		
RS-232 female connector	1		
LED	1		
Headers	4		
Total chips	17	Total chips	1
Component total	98	Component total	55

*Note that we define a component with more than 3 pins as a chip.

With reduced number of chips, the size of the Printed Circuit Board (PCB) can be reduced as well. Figure 4-22 presents the Smart Carpet PCB board implemented by traditional chips (left) and the PSoC based board (right).

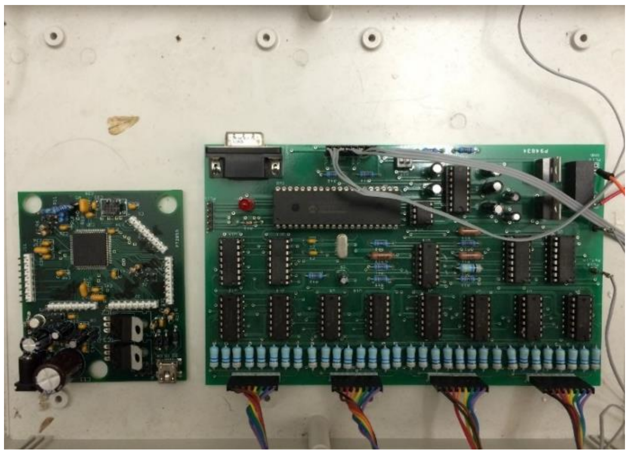


Figure 4-22 Board sizes of PSoC based system (left) and previous system using traditional IC chips (right). The size of the previous system is 6.69 by 3.93 inch, while the one implemented with PSoC only sizes 3.22 by 2.69 inch.

The dimension of the previous board is 6.69 by 3.93 inch, while the one implemented with PSoC only sizes 3.22 by 2.69 inch. The area of the previous board is 26.3 square inches and PSoC based board is 8.7 square inches. We can further reduce the size of the system by replacing the through hole chip package with surface mount device package; and by separating the power supply module from master/slave boards. Additionally we only need one power supply module for all master/slave boards. So future expansion can further reduce board area by requiring less power supply (currently we have one power supply module per board).

We can also put PCB boards into a commercial electrical box that is designed for wall installation; we can power it by directly connecting the board power to residential power system. Figure 4-24 shows the previous system and the PSoC based system with electrical box is shown in Figure 4-23. We put two PSoC based boards in the blue box.

Comparing the box for PSoC boards shown in Figure4-23 with the one used on previous board system shown in Figure4-24, we found the size reduces from about 4.3*11*13.4inches to 3.9*3.9*7.8 inches.

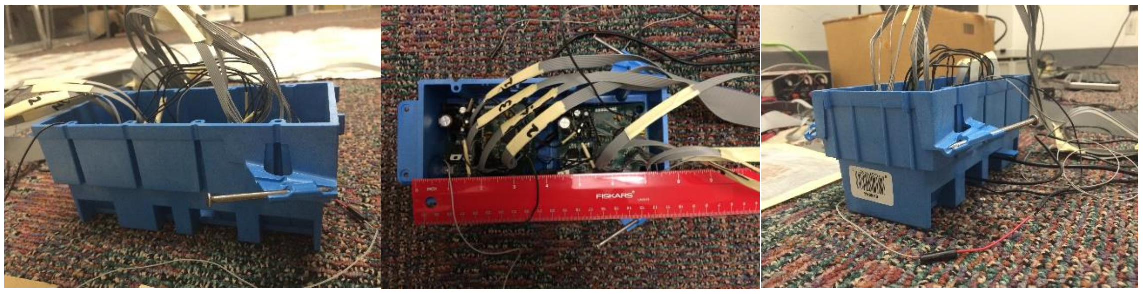


Figure 4-23 the Commercial electrical box for PSoC boards. Two processing boards can be put in a commercial electrical box.

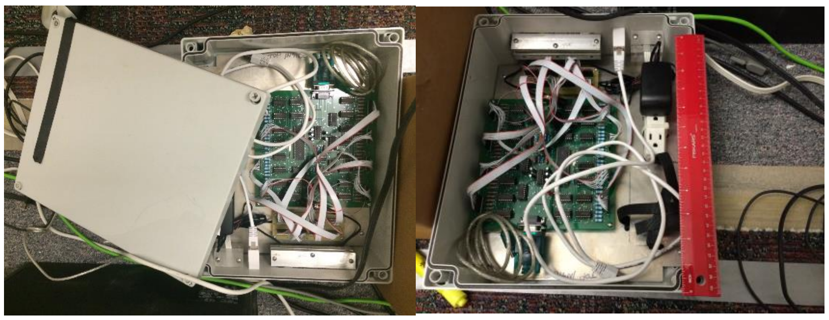


Figure 4-24 Commercial electrical box for previous smart carpet board.

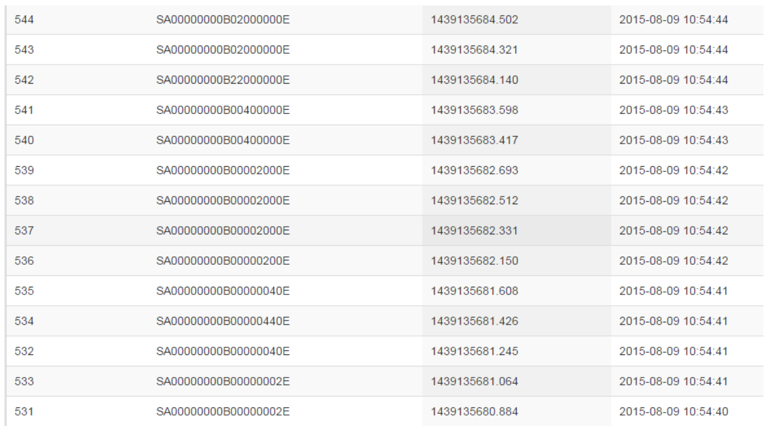
The cost has also been reduced for making one board using traditional chips or a PSoC based board. The cost of boards (see table 4-2) using traditional chips is $34.18 and with PSoC is $23.25. The traditional board processes 32 sensor signals and the PSoC board is designed for 40 sensor signals. Costs, of course, are subject to economic factor including reduction for value. And the PSoC can be programmed for varying number of inputs vs outputs. The total number of I/O pins is 62, some may be needed for other functionality, so at most we can design for 48 inputs, 0 outputs and other necessary uses.

Table 4-2 Cost comparison of making a PSoC based board and a traditional IC chip based board to process 32 sensors

Components on Last version board	quantity	**Components on PSoC based board**	**Cost**
1 PIC18F4455 microprocessor	$4.20	1 CY8C5868AXI-LP035	$13.70
9 TL084 op-amps	$7.00	2 LD1117V33 regulators	$1.00
4 CD4051 multiplexers	$2.00	3 1N5817-B diodes	$1.00
1 MAX232 logic level converter	$1.00	1 1N4148DO35-7 diode	$0.05
1 MCP3001 ADC	$1.74	1 2N2222 transistor	$0.05
1 NMH0509 DC-DC converter	$7.00	12 resisters	$1.00
1 7805 regulator	$1.34	24 capacitors	$1.20
1 7905 regulator	$0.30	9 headers	$1.00
47 Resistors	$2.4	50MIL KEYED SMD	$3.25
10 capacitors	$0.5	DCJ0202 jack	$1.00
17 DIP sockets	$2		
1 20MHz oscillator	$0.05		
1 DIP switch	$0.05		
1 RS-232 female connector	$4.00		
1 LED	$0.1		
4 headers	$0.5		
The total cost	$34.18	The total cost	$23.25

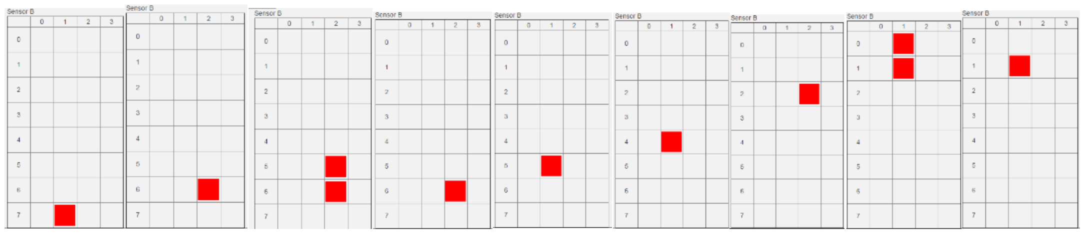
4.5 Walking Experiment

In the previous experiment, we observed the movement of a person walking and activating the sensors. To further demonstrate the accuracy of the walking detection, we conducted this experiment in which we walked over the sensors and recorded the sensor activity. We sent the data collected from the PSoC 5LP boards to raspberry-pi then sent to the cloud. So that the sensor status data stored is shown in Figure 4-25(a). The first column is an identity of the frame. The second column contains the sensor status frame sent from the PSoC boards. The third and the fourth column record the time when the client receives the data frame. A map of a person walking movement is illustrated in Figure4-25 (b). Starting from the left, the lowest row sensor (number 7,1) shows activation as the person proceeds up the sensor segment. The lower row number sensors are activated.



544	SA00000000B02000000E	1439135684.502	2015-08-09 10:54:44
543	SA00000000B02000000E	1439135684.321	2015-08-09 10:54:44
542	SA00000000B22000000E	1439135684.140	2015-08-09 10:54:44
541	SA00000000B00400000E	1439135683.598	2015-08-09 10:54:43
540	SA00000000B00400000E	1439135683.417	2015-08-09 10:54:43
539	SA00000000B00002000E	1439135682.693	2015-08-09 10:54:42
538	SA00000000B00002000E	1439135682.512	2015-08-09 10:54:42
537	SA00000000B00002000E	1439135682.331	2015-08-09 10:54:42
536	SA00000000B00000200E	1439135682.150	2015-08-09 10:54:42
535	SA00000000B00000040E	1439135681.608	2015-08-09 10:54:41
534	SA00000000B00000440E	1439135681.426	2015-08-09 10:54:41
532	SA00000000B00000040E	1439135681.245	2015-08-09 10:54:41
533	SA00000000B00000002E	1439135681.064	2015-08-09 10:54:41
531	SA00000000B00000002E	1439135680.884	2015-08-09 10:54:40

(a)



(b)

Figure 4-25(a) shows data arrays displayed on Cloud it is send out from usb module on PSoC system received by Raspberry-Pi which finally send the data to the Cloud. (b) is the map of sensor carpet illustrating movement.

Male and female volunteers are asked to walk over the carpet as shown in Figure 4-21 (a). They walked the full length of the segment (half round), two full segment lengths (one round) and finally traverse the two segments twice (two rounds). They started from the empty space beside sensor segment A, and then walked over segment A and B. For full-round walking and two-round walking experiments, they can also turn around at the empty spaces beside segment A and B shown in Figure 4-26.

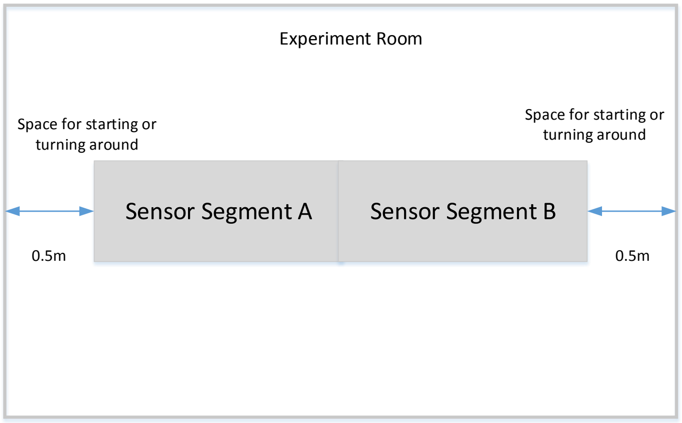
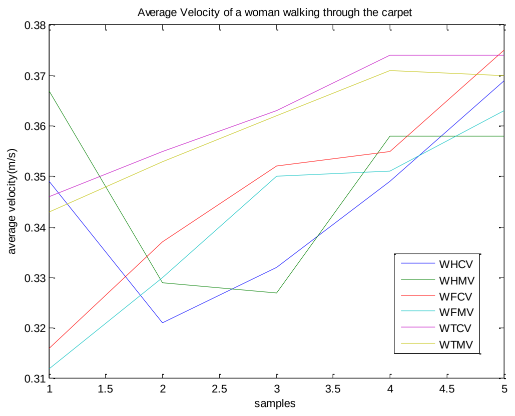

Experiment Room
Space for starting or turning around
Space for starting or turning around
Sensor Segment A
Sensor Segment B
0.5m
0.5m

Figure 4-26 Experiment room setting and two sensor segments layout. Two sensor segments are put in the center of the experiment room and align to each other. The distances between the walls to the left side of segment A and the right side of segment B are both about 0.5 meter.

We determined their walking start time and stop time by a stopwatch and the number of steps for each sample, and we subtracted the first frame start and the last frame stop times. The average velocity of the female volunteer is shown in Figure 4-27(a) and the male's is shown in (b). The figure 4-27 legend sample tags term like WHCV indicate how the data was obtained: 'W' means woman and 'M' means man of the first letter in the sample tags, 'H','F' and 'T' represents half round walking, full round walking and two rounds walking, 'C' means the walking time is measured by frame start and stop time and 'M' means the walking time is measured by stopwatch. Finally V is velocity. We can see from the Figure 4-27 (a) and (b), for the same type of experiment, the average velocity from frame data follows the data measured by stopwatch, and has little difference.



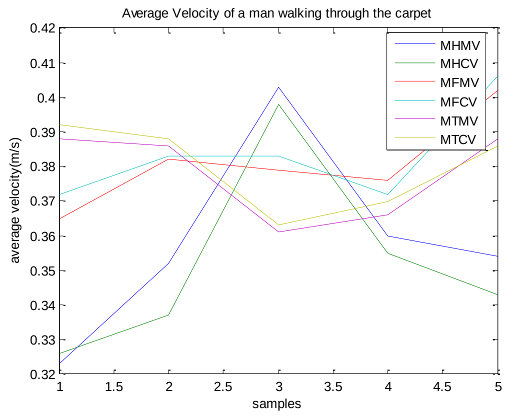


Figure 4-27 for the same type of experiment, the average velocity from frame data follows the data measured by stopwatch, and has little difference.

To obtain the average activating sensors per step, we also determined how many steps the volunteer walked for each experiment and counted the total number of activating sensors from the frame data, as shown in Table 4-3. The first and second column in the table shows the identity of each experiment and the gender of the volunteer. The number of steps of each experiment is listed in column 3. The total number of activating sensors is listed in column 4 and the average activating sensors per step is presented in column 5. The first ten sets of data are obtained from half round walking experiments. The second ten sets of data are obtained from full round walking experiments. The third ten sets of data are obtained from two round walking experiments. We can see from the table, the average actives sensor per step varies from 1.3 to 2.4 and the result of male is slightly greater than female, which is reasonable.

Table 4-3 Average Active Sensors per Step

ID	gender	Number of steps	Number of active sensors	average active sensor/step
1	Female	11.000	19.000	1.727
2		11.000	18.000	1.636
3		11.000	19.000	1.727
4		11.000	17.000	1.545
5		11.000	15.000	1.364
6	Male	10.000	24.000	2.400
7		10.000	18.000	1.800
8		10.000	22.000	2.200
9		10.000	14.000	1.400
10		10.000	19.000	1.900
11	Female	21.000	29.000	1.381
12		21.000	29.000	1.381
13		21.000	26.000	1.238
14		21.000	28.000	1.333
15		21.000	28.000	1.333
16	Male	19.000	32.000	1.684
17		19.000	30.000	1.579
18		19.000	29.000	1.526
19		19.000	33.000	1.737
20		19.000	29.000	1.526
21	Female	42.000	61.000	1.452
22		39.000	65.000	1.667
23		40.000	64.000	1.600
24		40.000	75.000	1.875
25		40.000	72.000	1.800
26	Male	36.000	70.000	1.944
27		35.000	74.000	2.114
28		35.000	74.000	2.114
29		36.000	75.000	2.083
30		37.000	77.000	2.081

CHAPTER 5   DISCUSSION

Our laboratory has designed and implemented an unobtrusive sensor monitoring system to detect personnel. One benefit is that it detects falls, as has been discussed also here [12-15]. The system consists of sensors placed on the floor, electronic boards to read the data from the sensors, then a computer to process the data from the sensors and send notifications as requested. The electronic board consists of the microcontroller, the amplifier, the multiplexer and other components to handle signals from many sensors before sending the data to the computer [15]. A microprocessor controls the circuit and sends the output to a computer.

Specifically, our laboratory has constructed systems to connect to multiple sensors using multiplexers to select data. The sensor signal then goes to a differential amplifier, which increases the sensor signal and reduces the 60 Hz noise signal. The A/D converter then samples the output of the differential amplifier and converts its output to a digital number representation. For serial communication, we have the RS-232 communication for sending data to a computer, also the I^2C communication for sending and receiving data between boards [15].

We designed a substantially smaller system using programmable system on chip (PSoC). It integrates analog subsystem and digital subsystem. Analog subsystem includes analog multiplexers and programmable gain amplifiers. Digital subsystem includes analog to digital converter (ADC), USBUART module and I^2C module. This required both software and hardware to implement the PSoC system beginning with a routing scheme to route PSoC's functional units to meet the requirement of smart carpet hardware. We first tested the programmability of PSoC by programing one or multiple GPIO pins to detect sensor signals. Next we implemented a four-sensor system on PSoC development kit. In this system, four sensors were connected to four GPIOs selected by a multiplexer. Selected sensor signal was amplified by a differential amplifier and converted to digital data by an ADC. Data from all four sensors data was sent to a computer by USBUART. Then, we implemented 32-sensor system with the PSoC development kit board. Finally, we designed PSoC based PCB boards which can work cooperatively using I^2C.

A main benefit from applying PSoC chip to the Smart Carpet system is to reduce number of chips. Thus, we can reduce the board size and that of the whole system. Cost comparisons for the board using traditional chips with a PSoC based board shows they have been reduced.

A. The number of chips from the traditional design is 17 but for the PSoC design with the same functionality has only 1 chip.

B. The traditional board size is 26.3 square inches and the PSoC design is 8.7 square inches.

C. See Figure 4-23 and 4-24 where we have place a ruler for size comparison. The size of the electrical box reduces from about 4.3*11*13.4inches to 3.9*3.9*7.8 inches.

Finally, to demonstrate the PSoC operability compared to the traditional system, we performed a series of walking experiments from which we derived velocity and average active sensor per step. This demonstrated that the PSoC design provided data as well as the traditional design. We are able to monitor personnel walking on the carpet. This raises our confidence in having an electronic board that can be incorporated easily and at low cost.

CHAPTER 6 CONCLUSION

We have developed a personnel fall detection system to monitor walking activities on the floor. The sensor pads under the carpet provide the signal sequences. We use a signal chip, a Programmable System on Chip (PSoC) to connect to the sensors. By programming we can route each sensor to a differential amplifier that amplifies the selected signal and then to an ADC that converts the amplified signal to digital data. The microprocessor in the PSoC then performs an ASCII conversion on the digital data, and feeds the converted data to the serial port of a PC using USBUART for further processing and data display by an external computer. All of the functional units above are implemented with the single PSoC chip, the PSoC 5LP from Cypress Semiconductor Inc. This reduces the individual chips, leading to reduction numbers of components and smaller system size.

To use the PSoC, we mapped its functional units to meet the requirement of smart carpet hardware. This has been successfully tested with the PSoC development board. We use a USB port to send data to an external computer. Based on the functional units we need, we designed the power supply system and the necessary peripherals for the PSoC chip, from which we created the Printed Circuit Board (PCB). Using the on board I^2C communication in the PSoC multiple boards can work cooperatively. This makes the whole system scalable as the Carpet size increases.

This has reduced the total number of IC chips from 17 to 1 and components from 98 to 56 and reduced the board size from about 26.3 square inches to 8.7 square inches.

Currently, one power supply is integrated in each 40-sensor hardware board. However, one power supply is enough for the wholes system. As a result, we can design a board in the future with one power supply to support multiple PSoC chips placed in parallel. Each PSoC chips can handle at most 48 sensors. This can make the Smart Carpet hardware board more scalable and smaller than current design.

REFERENCE

[1] G. F. FULLER. (2000, Apr 1,2000). *Falls in the Elderly*. Available: http://www.aafp.org/afp/2000/0401/p2159.html

[2] E. E. Stone and M. Skubic, "Fall detection in homes of older adults using the microsoft kinect," *Biomedical and Health Informatics, IEEE Journal of,* vol. 19, pp. 290-301, 2015.

[3] M. Rantz, M. Skubic, C. Abbott, C. Galambos, M. Popescu, J. Keller*, et al.*, "Automated in-home fall risk assessment and detection sensor system for elders," *The Gerontologist,* vol. 55, pp. S78-S87, 2015.

[4] M. Alwan, P. J. Rajendran, S. Kell, D. Mack, S. Dalal, M. Wolfe*, et al.*, "A smart and passive floor-vibration based fall detector for elderly," 2006, pp. 1003-1007.

[5] T. Degen, H. Jaeckel, M. Rufer, and S. Wyss, "SPEEDY: A Fall Detector in a Wrist Watch," 2003, pp. 184-189.

[6] K. Doughty, R. Lewis, and A. McIntosh, "The design of a practical and reliable fall detector for community and institutional telecare," *Journal of Telemedicine and Telecare,* vol. 6, pp. 150-154, 2000.

[7] Y. Zigel, D. Litvak, and I. Gannot, "A method for automatic fall detection of elderly people using floor vibrations and sound—Proof of concept on human mimicking doll falls," *Biomedical Engineering, IEEE Transactions on,* vol. 56, pp. 2858-2867, 2009.

[8] A. Sixsmith, N. Johnson, and R. Whatmore, "Pyroelectric IR sensor arrays for fall detection in the older population," 2005, pp. 153-160.

[9] Y. Li, Z. Zeng, M. Popescu, and K. C. Ho, "Acoustic fall detection using a circular microphone array," 2010, pp. 2242-2245.

[10] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Robust video surveillance for fall detection based on human shape deformation," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 21, pp. 611-622, 2011.

[11] M. Kepski, B. Kwolek, and I. Austvoll, "Fuzzy inference-based reliable fall detection using Kinect and accelerometer," 2012, pp. 266-273.

[12] R. V. Neelgund, "Floor sensor development using signal scavenging for personnel detection system," 2010.

[13] U. G. Shriniwar, "Data control for signal scavenging for a personnel detection system," University of Missouri--Columbia, 2010.

[14] K. R. Gadre, "Fall detection system using low cost computing and online communication," University of Missouri--Columbia, 2012.

[15] C. Suriyakul, "A new circuit for personnel detection using signal scavenging," Masters thesis for fulfillment of MS Degree, University of Missouri--Columbia, 2015.

[16] H. W. Tyrer, R. Neelgund, A. Mohammed, and U. Shriniwar, "Signal scavenging for passive monitoring in eldercare technology," 2009, pp. 6167-6170.

[17] H. W. Tyrer, R. Neelgund, U. Shriniwar, and K. K. Devarakonda, "Faux-floor development system for personnel detection using signal scavenging sensors," 2010, pp. 394-397.

[18] M. P., "Input Impedance Measurments for Copper and Aluminum Sensors," University of Missouri, Senior report in the department of Electrical and Computer engineering 2014

[19] C. Semiconductor. (2015). *PSoC® 5LP*. Available: http://www.cypress.com/products/psoc-5lp

[20] C. Semiconductor. (2015). *PSoC® Creator™ Integrated Design Environment (IDE)*. Available: http://www.cypress.com/products/psoc-creator-integrated-design-environment-ide

[21] C. Semiconductor. MiniProg3 User Guide [Online]. Available: http://www.cypress.com/file/44091/download

[22] C. Semiconductor, "Delta Sigma Analog to Digital Converter (ADC_DelSig)," 2015.

[23] C. Semiconductor, "I2C Master/Multi-Master/Slave," 2015.

[24] C. Semiconductor. (2015). *PSoC® Creator™ Quick Start Guide*. Available: http://www.cypress.com/file/195271/download

[25] G. Reynolds. (2015). *Using PSoC®3 and PSoC 5LP GPIO Pins*. Available: http://www.cypress.com/file/45381/download

[26] P. Sekar. Instrumentation Amplifier Using PSoC®3 [Online]. Available: http://www.cypress.com/file/53376/download

[27] S. L. Garfinkel. (1999). *USB deserves more support*. Available: http://simson.net/clips/1999/99.Globe.05-20.USB_deserves_more_support+.shtml

[28] M. Ainsworth. PSoC®3 and PSoC 5LP Hardware Design Considerations [Online]. Available: http://www.cypress.com/file/44581/download

[29] *USB*. Available: https://en.wikipedia.org/wiki/USB

[30] (2015). *CadSoft EAGLE PCB Design Software*. Available: http://www.cadsoftusa.com/eagle-pcb-design-software/?CMP=KNC-GUS-FUS-GEN-SUP-CAD-Eagle-PCB

[31] M. Hastings. PSoC® 3, PSoC 4, and PSoC 5LP Mixed-Signal Circuit Board Layout Considerations [Online]. Available: http://www.cypress.com/file/136286/download

[32] "How to Solder QFP, TSSOP, SOIC, and Other Surface Mount Parts."

[33] Song Tian, Da Li, and Ying Yao. "Multi-source data oriented flexible real-time information fusion platform on FPGA." Electronics, Communications and Control (ICECC), 2011 International Conference on. IEEE, 2011.

APPENDIX A - MANUAL OF USING THE PSOC BASED SMART CARPET BOARD

1. Power the board

The board can be powered using an AC/DC adaptor. Connect the power adapter to J3 shown in Figure A-1. Use adaptors with the following specifications:

Connector: 2.1/5.5mm Plug

Input: AC100 ~ 240V, 50/60Hz, 2A max

Output: DC5V, 2000mA

Dimensions: 2.75"(L) x 1.125"(W) x 1.875" (H)

2. Connect to ground

Connect one of ground pins on board to the earth ground using a ground wire shown in Figure A-1.

3. Connect to computer

To send scavenging results, the ASCII strings, to the computer, connect the USB cable to both board USB connector and the computer.

4. Connect to the sensors

The current PSoC board provides 40 ports to connect to sensor outputs which are J2, J8, J10, J11, and J12 shown in Figure A-1. The 8 ports from left to right on J11 connect to GPIO P0[0] -P0[7]. The 8 ports from left to right on J12 connect to GPIO P6[0] – P6[7]. The 8 ports from left to right on J8 connect to GPIO P5[0] –P5[7]. The 8 ports from top to bottom on J2 connect to GPIO P2[0] -P2[7]. The 8 ports from up to bottom on J10 connect to GPIO P4[0] –P4[7]. All the ports are able to use collect sensor signal, while they are required to be configured using a program.

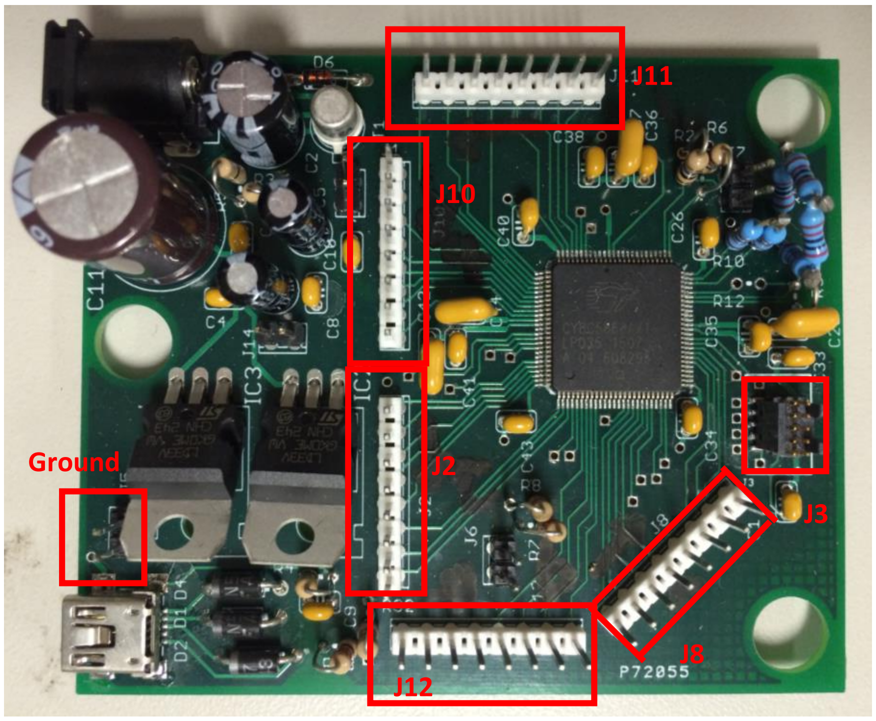


Figure A-1 PSoC based Smart Carpet board connection. Connector J2, J8, J10, J11 and J12 are used to connect to sensors. J3 is the programming connector.

5. Programming the board

The board can be programmed using a MiniProg 3 which provides SWD programming through the 10-pin connector J3 shown in Figure A-1. To program the PSoC 5lp chip on board, first download and install free software PSoC Programmer provided by Cypress Semiconductor Inc. to a computer. Next, connect Minprog 3 10-pin connector to connector J3 on board. Then connect the MiniProg3 to your computer's USB port using the USB cable. At last, run PSoC Programmer, add the path of the source code to be downloaded to the board and click program button. PSoC Creator is the software which you can write your program, configure the modules and GPIO ports, and generate the source codes.

APPENDIX B – SOURCE CODE FOR THE MASTER BOARD

```
/* ========================================
 * Smart Carpet PSoC based master board
 * University of Missouri
 * ECE department
 * Mengxuan Ma
 * 11/5/2015
* ========================================
*/
#include <project.h>
#include "stdio.h"

/* The I2C Slave address by default in a PSoC device is 8 */
#define I2C_SLAVE_ADDRESS    8
/* Set the write buffer length to be 8 bits or 1 bytes */
#define MRD_BUFFER_SIZE      4
#define SENSOR_NUM           32

int main(){
    int16 adc_result;
    int16 adc_result_mv;

   /*store ascii represent of sensors and send to PC*/
    uint8 UART_ascBuffer_master[8]={0x00};
    uint8 UART_ascBuffer_slave[8]={0x00};

    /*store sensors' binary numbers*/
    uint32 master_buffer = 0x00000000;
```

```
uint32 slave_buffer = 0x00000000;

uint8 channel;
uint8 i;

/*I2C parameters*/
uint8 read_status;

/*used to change binary value to ascii value*/
uint32 mask = 0x0000000F;
uint32 temp = 0x00000000;

uint8 newline[2]={"\n\r"};  // for creating a new line and carriage return
uint8 startsign[2]={"SA"};
uint8 segb[1]={"B"};
uint8 endsign[1]={"E"};
uint8 warning[13] = {"\n\ri2c error\n\r"};

/////////////components initialization////////////////////
/* Place your initialization/startup code here (e.g. MyInst_Start()) */
/*hardware parts:PGA and AMux start*/
PGA_1_Start();
PGA_2_Start();
PGA_3_Start();
AMux_1_Start();

/*start I2C*/
I2CM_Start();

/* Enable Global Interrupts */
CyGlobalIntEnable; /* Enable global interrupts. */
```

```
/* Start USBFS Operation with 3V operation */

USBUART_1_Start(0u, USBUART_1_3V_OPERATION);


/*start adc*/

ADC_DelSig_Start();

          ADC_DelSig_StartConvert();


/* Wait for Device to enumerate */

while(!USBUART_1_GetConfiguration());


/* Enumeration is done, enable OUT endpoint for receive data from Host */

USBUART_1_CDC_Init();


for(;;) {

  master_buffer = 0; slave_buffer = 0;

  for(channel=0;channel<SENSOR_NUM;channel++){


    AMux_1_FastSelect(channel); CyDelay(5u);

    /* Place your application code here. */

    /*Start ADC conversions */

                    ADC_DelSig_StartConvert();


                    /* Wait for next ADC result to be available and read data*/

                    ADC_DelSig_IsEndConversion(ADC_DelSig_WAIT_FOR_RESULT);

                    adc_result =ADC_DelSig_GetResult16();

    adc_result_mv = ADC_DelSig_CountsTo_mVolts(adc_result) ;


    /* Stop ADC conversion */

                    ADC_DelSig_StopConvert() ;


              /* Convert ADC data to ascii format for sending it to UART hyperterminal.

                    Include stdio.h if you want to use this function. */
```

```
        if(adc_result_mv>500){
            master_buffer |= 1 << channel;
        }

        //sprintf((char *)UART_txBuffer,"%1d %5d",channel,adc_result_mv);
        adc_result_mv=0;
      } //end amux loop

    /////////////////Read Slave/////////////////////
        I2CM_MasterClearStatus();

        do{
            read_status = I2CM_MasterReadBuf(I2C_SLAVE_ADDRESS, (uint8 *)&slave_buffer,
MRD_BUFFER_SIZE, I2CM_MODE_COMPLETE_XFER);
        }while(read_status!=I2CM_MSTR_NO_ERROR);

        while(  I2CM_MasterStatus()&I2CM_MSTAT_XFER_INP );
        // Master finishes reading, so reset slave buffer pointer to the beginning of the buffer
        // I2CM_MasterClearReadBuf();
        read_status = I2CM_MasterClearStatus();

        if(read_status & I2CM_MSTAT_ERR_XFER){
            while(USBUART_1_CDCIsReady() == 0u){};
            USBUART_1_PutData(warning, 13); }
        ////////////////////////////////////////

        for(i=0;i<(SENSOR_NUM/4);++i){
            temp = master_buffer & mask;
            if (temp < 10){
                                    UART_ascBuffer_master[i] = (uint8)temp + 0x30;
                                }else {
                                        UART_ascBuffer_master[i] = (uint8)temp + 0x37;}
        master_buffer = master_buffer >> 4;
    }//End of conversion loop


    for(i=0;i<(SENSOR_NUM/4);++i){
        temp = slave_buffer & mask;
        if (temp < 10){
                        UART_ascBuffer_slave[i] = (uint8)temp + 0x30;
                    }else {
                            UART_ascBuffer_slave[i] = (uint8)temp + 0x37;}
        slave_buffer = slave_buffer >> 4;
    }//End of conversion loop


    while(USBUART_1_CDCIsReady() == 0u);    /* Wait till component is ready to send more data to
the PC */
    USBUART_1_PutData(newline, 2);


    while(USBUART_1_CDCIsReady() == 0u);    /* Wait till component is ready to send more data to
the PC */
    USBUART_1_PutData(startsign, 2);


    while(USBUART_1_CDCIsReady() == 0u);
    USBUART_1_PutData(UART_ascBuffer_master, 8);


    while(USBUART_1_CDCIsReady() == 0u);    /* Wait till component is ready to send more data to
the PC */
    USBUART_1_PutData(segb, 1);


    while(USBUART_1_CDCIsReady() == 0u);
    USBUART_1_PutData(UART_ascBuffer_slave, 8);


    while(USBUART_1_CDCIsReady() == 0u);    /* Wait till component is ready to send more data to
the PC */
    USBUART_1_PutData(endsign, 1);
```

```
        while(USBUART_1_CDCIsReady() == 0u);    /* Wait till component is ready to send more data to
the PC */

        USBUART_1_PutData(newline, 2); }

}

/* [] END OF FILE */
```

APPENDIX C – SOURCE CODE FOR THE SLAVE BOARD

```
/* ========================================
 * Smart Carpet PSoC based slave board
 * University of Missouri
 * ECE department
 * Mengxuan Ma
 * 11/5/2015
 * ========================================
*/
#include <project.h>
#include "stdio.h"

/* Set the write buffer length to be 16 bites or 2 bytes */
#define RD_BUFFER_SIZE     4
#define SENSOR_NUM         32

int main(){
    int16 adc_result;
    int16 adc_result_mv;

    /*store sensors' binary numbers*/
    uint32 buffer = 0x00000000;

    uint8 channel;

    /*set I2C slave status*/
    I2CS_SlaveSetAddress(8);

    I2CS_SlaveInitReadBuf((uint8 *)&buffer, RD_BUFFER_SIZE);
```

```
/////////////components initialization////////////////////
/* Place your initialization/startup code here (e.g. MyInst_Start()) */
/*hardware parts:PGA and AMux start*/
PGA_1_Start();
PGA_2_Start();
PGA_3_Start();
AMux_1_Start();

/*start I2C*/
I2CS_Start();

/* Enable Global Interrupts */
CyGlobalIntEnable; /* Enable global interrupts. */

/*start adc*/
ADC_DelSig_Start();
        ADC_DelSig_StartConvert();



for(;;)
{
  for(channel=0;channel<SENSOR_NUM;channel++){

    AMux_1_FastSelect(channel); CyDelay(5u);
    /* Place your application code here. */
    /*Start ADC conversions */
                ADC_DelSig_StartConvert();

                /* Wait for next ADC result to be available and read data*/
                ADC_DelSig_IsEndConversion(ADC_DelSig_WAIT_FOR_RESULT);
```

```
                        adc_result =ADC_DelSig_GetResult16();
        adc_result_mv = ADC_DelSig_CountsTo_mVolts(adc_result) ;



        /* Stop ADC conversion */
                        ADC_DelSig_StopConvert() ;


                /* Convert ADC data to ascii format for sending it to UART hyperterminal.
                        Include stdio.h if you want to use this function. */


        if(adc_result_mv>500){
            buffer |= 1u << channel;
        }else{
            buffer &= ~(1u << channel);            }
        adc_result_mv=0;
      } //end amux loop


        //while( I2CS_SlaveGetReadBufSize()< RD_BUFFER_SIZE );
        //buffer=0;
        while (  I2CS_SlaveStatus() & I2CS_SSTAT_RD_BUSY  );
        //{while( I2CS_SlaveGetReadBufSize()< RD_BUFFER_SIZE );}
        //I2CS_SlaveClearReadStatus();
        I2CS_SlaveClearReadBuf();  }
}

/* [] END OF FILE */
```