Dissertation

Sensor Node for Medical Application

Mohd Asri bin Ahmad

5592

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

June 2008

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

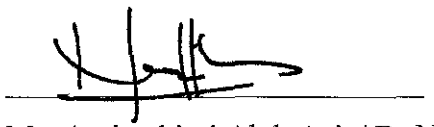
SENSOR NODE FOR MEDICAL APPLICATION

by

Mohd Asri bin Ahmad

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:



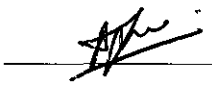
Ms. Azrina binti Abd. Aziz/ Dr Nor Hisham Hamid
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2008

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



MOHD ASRI BIN AHMAD

ABSTRACT

Sensor network has been used in many applications namely military, medical and environment. It is used to monitor and collect data from physical environment. It is used to monitor and collect data from physical environment. The sensor network is basically a collection of sensor nodes usually interconnected wirelessly to perform a specific task. The aim of this project is to design and develop an automated system for medical application. The system will be using sensor nodes for collecting temperature data of the patient diagnosed with dengue fever. This system designed is to replace the current method used in hospitals to save clinicians' time and to have proper database system for future access.

The sensor node system will connect patients to the PC through wireless connection mode. The temperature reading will be stored on the computer and graphs will be plotted automatically. It is envisaged that the system built will help clinicians to perform their job faster.

ACKNOWLEDGEMENTS

First of all, the author is grateful to Allah S.W.T. for the completion of this Final Year Project. Deepest gratitude goes to project supervisors Ms. Azrina Abd. Aziz and Dr Nor Hisham Hamid for their guidance and patience in ensuring that the author accomplished the project objectives, also for their supports and commitments. Special thanks to FYP technologist, Ms. Siti Hawa Mohd Tahir for her assistance in helping the author completed the tasks related.

The author also would like to express a thousand appreciations to other Electrical & Electronics lecturers and colleagues who have involved directly or indirectly with this project for all the help and guidance throughout the year.

The gratitude extended to author's family who has given undivided support to the author throughout the completion of the project. The author will always be grateful for their sacrifices and sincerity.

Finally, million thanks to all parties who had also contributed directly or indirectly towards the success of this project.

Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADC	Analog Digital Converter
AP	Access Point
ED	End Device
FIFO	Flow In Flow Out
IrDA	Infrared Data Association
MCU	Microcontroller Unit
RF	Radio Frequency
SARS	Severe Acute Respiratory Syndrome
TB	Tuberculosis
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
WSN	Wireless Sensor Network

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Body's core temperature is a measure of body ability to generate and get rid of heat [1]. Temperature is one of the illness indicators of our body. Many diseases such as dengue fever, SARS, TB and Malaria can be detected by changes in the body temperature. Patients diagnosed with these diseases are normally placed in intensive care. Their body temperature will be monitored continuously for safety measured if something bad happens in sudden to them.

There are many places to measure temperature. The most common places are the mouth, armpit, ear, forehead and rectum (shows in Figure 1.1). It usually been measured by using a gadget called mercury thermometer. The normal body's core temperature of a healthy, resting adult human being is stated to be at 98.6 degrees Fahrenheit or 37.0 degrees Celsius [2]. When our body is too hot, the readings will be higher than normal core body temperature. When the body is cold, the reading will be vice versa.

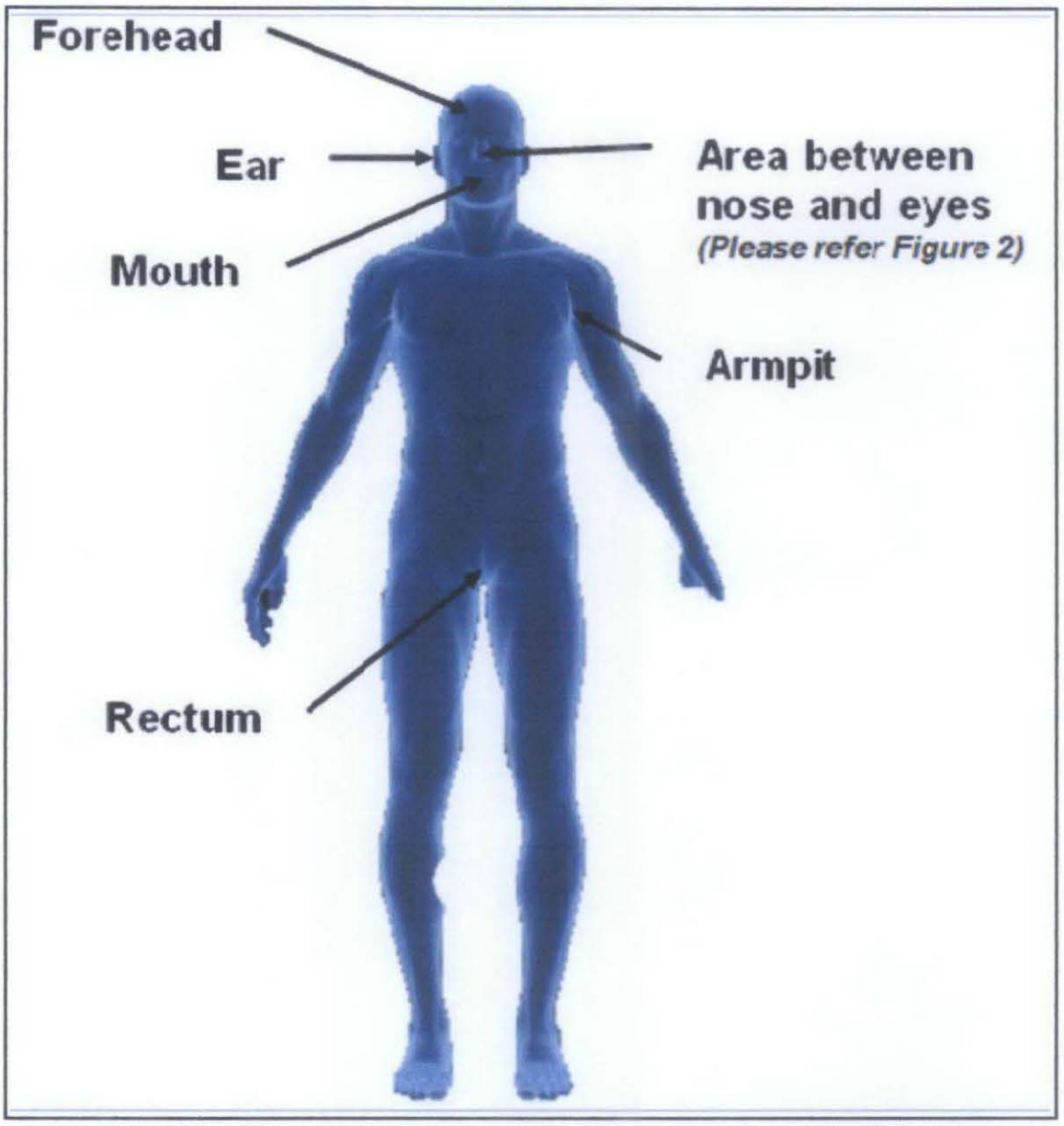


Figure 1.1: The most common places for temperature measurement.

The main objective of this project is to develop an automated system to collect temperature data from patients. Currently clinicians are measuring temperature using thermometer at a periodic time. The temperature data gathered are then used to plot graph for analysis purpose. As an alternative, a system composed of wireless sensor network is proposed to collect temperature data which will later be used during analysis process.

The sensor network in this project will consist of two sensor nodes (patients) communicating with the main host (PC). The sensor nodes used to collect temperature of patients will be placed at eye's cord.

1.1 Problem Statement

Monitoring temperature continuously is very important for certain disease like dengue fever, SARS and Malaria [4]. By doing this, we can detect and manage the infection early and prevent any side complications. Yale researcher Dr. M. Marc Abreu has identified an area of the brain he calls the brain temperature tunnel, which allow physicians to easily measure the body's core temperature by monitoring a patch of skin [5]. This patch of skin claimed by the researcher is located at eyes cord. From the area, we can measure the temperature continuously and more accurately compared to other areas such as forehead, armpit and mouth.

The existing method of collecting temperature data is done manually at hospitals. Usually the clinician will measure the temperature at patient's armpit, mouth or forehead. This data measurement will be made periodically. After gathering all the data, a graph will be plotted manually. Then the graph's pattern will be analysed by doctor for further medical examination.

Current problems of doing these normal methods are time consuming. Clinicians need to repeat the same steps to gather the data before it can be analysed by the doctor.

As an alternative, this project will use wireless sensor node to monitor the temperature continuously. By using the sensor node we can speed up the diagnosing time and save time. This sensor node will be placed at eye's cord for high accuracy data measurement.

This system will also allow many patients to be monitored simultaneously. It is because our application can be integrated with network which can be connected by multiple numbers of sensor nodes for each wireless sensor network. Another advantage of this system is we can have proper and accurate database system. The database system can be easily accessed by the doctor because it will be digitally indexed.

1.2 Objectives

The objective of this project is to build a sensor network that can be used to measure and collect patient's temperature diagnosed with dengue disease. It will help the clinicians in gathering their patient's temperature data and stored it in digital form for further use. With the sensor network clinician can save more time and acquire more accurate data for diagnosis. By having digital data of their patient's we can develop a proper and accurate database system. With this database system we can have proper medical history of our patient which may be important for future works.

1.3 Scope of Study

This project takes into consideration the study of dengue fever that happens in Malaysia. Wireless sensor node will be used to measure the temperature of the patient who diagnosed with dengue fever. With the data taken by the sensor node, it could have a better way of analysing the dengue disease.

At the end of this project, the following will be developed.

- A better way to measure body's core temperature who diagnosed with the disease.
- We also can develop wireless sensor network for monitoring many patients simultaneously.
- Moreover this project will provide help to clinician to enhance their normal methods which have a lot of problems.
- Develop a proper database system for their patients

CHAPTER 2

LITERATURE REVIEW

2.0 Wireless Sensor Network

Wireless sensor communications and digital electronics have enabled developments of multifunction sensor nodes that can be communicated at distance. This technology is called as wireless sensor network (WSN). The concept of wireless sensor network is very simple. It consists of multiple numbers of sensor node, processor, transceiver and power supply as their basic components. With these components we can develop a lot of sensor node application which can be used in many fields namely military, health and environment. A wireless sensor network is a wireless network of distributed devices that using sensors to measure certain parameters such as temperature, pressure, vibration and others.

Figure 2.1 shows architecture of wireless sensor networks. It consists of a large number of sensor nodes over the area of interest. Each node is equipped with one or several sensors (suited to the application), a short range (10-100m) radio and a microcontroller. The nodes form an ad-hoc network capable of sending the sensed data to one or more base stations that further forward the data using a long haul link to the monitoring center [6].

2.1 Sensor Node

The sensor node consists of four basic parts which are processing unit, sensing unit, transceiver and power unit [6]. From the Figure 2.1 it is their basic architecture of sensor node unit. Figure 2.2 show the typical architecture of the sensor node.

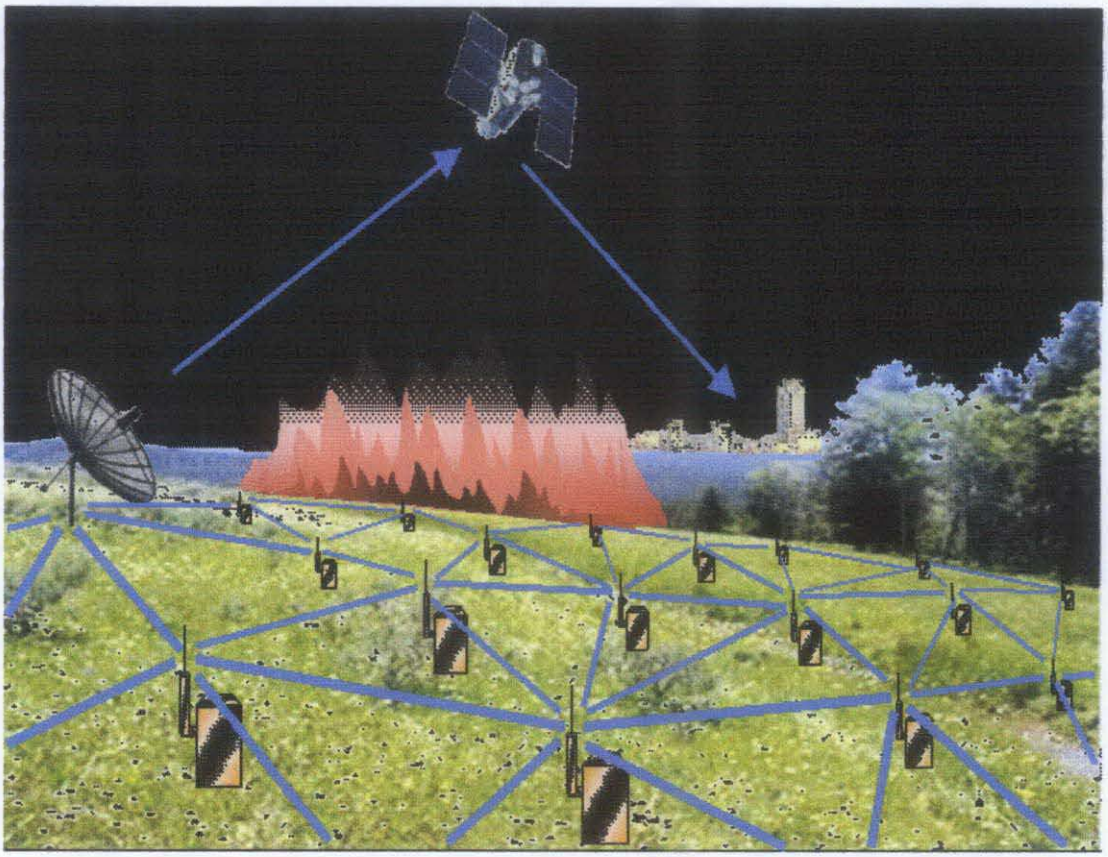


Figure 2.1: Sample architecture of wireless sensor networks [6]

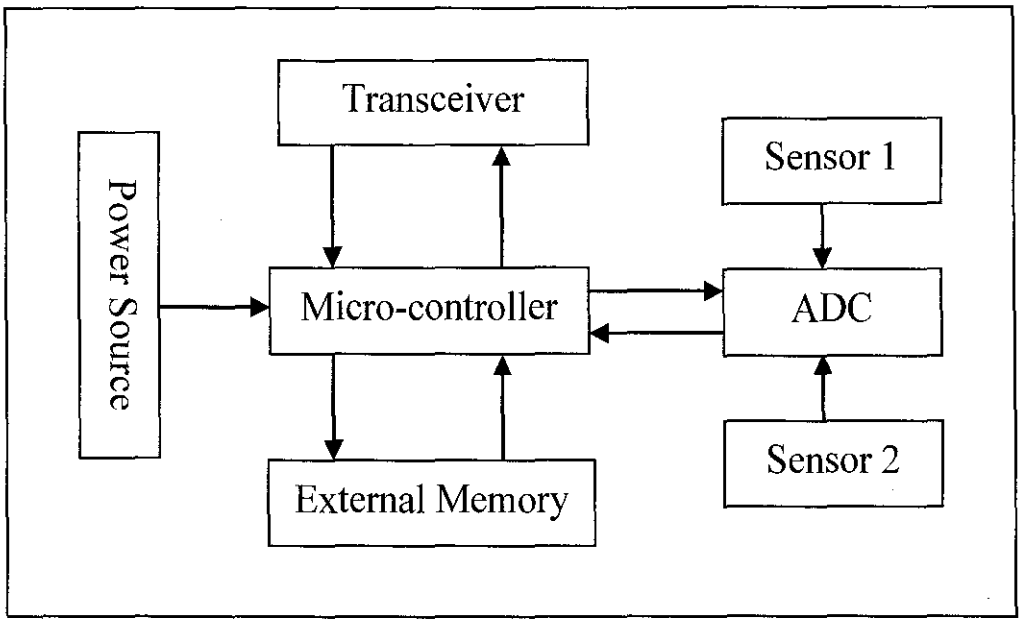


Figure 2.2: The sensor node architecture [7]

2.1.1 Processing unit

The main function of processing unit is to perform tasks, process data and control the functionality of other components in the sensor node [7]. An example of processing unit is microcontrollers which are most suitable choice for its flexibility to connect to other devices, programmable, power consumption is less, as these devices can go to sleep state and part of controller can be active.

2.1.2 Transceiver

The various choices of wireless transmission media are Radio frequency, Optical communication (Laser) and Infrared. Laser requires less energy, but needs line of sight for communication and also sensitive to atmospheric conditions. Infrared like laser, needs no antenna but is limited in its broadcasting capacity. Radio Frequency (RF) based communication is the most relevant that fits to most of the WSN applications. WSN's use the communication frequencies between about 433MHz and 2.4GHz [7]. The functionality of both transmitter and receiver are combined into a single device know as transceivers.

2.1.3 Power source

Power consumption in the sensor node is for the sensing, communication and data processing. More energy is required for data communication in sensor node. Energy expenditure is less for sensing and data processing. The energy cost of transmitting 1Kb a distance of 100 m is approximately the same as that for the executing 3 million instructions by 100 million instructions per second/W processor [7]. Power is stored either in Batteries or Capacitors. Batteries are the main source of power supply for sensor nodes.

2.1.4 Sensor Unit

Sensors are hardware devices that produce measurable response to a change in a physical condition like temperature and pressure. The continual analog signal sensed by the sensors is digitized by Analog-to-Digital converter and sent to controllers for further processing. Characteristics and requirements of sensor node should be small size, consume extremely low energy, operate in high volumetric densities, be autonomous and operate unattended, and be adaptive to the environment.

2.2 Dengue Fever [10]

Dengue fever is an infectious disease carried by mosquitoes and caused by any of four related dengue viruses. This disease used to be called "break-bone" fever because it sometimes causes severe joint and muscle pain that feels like bones are breaking, hence the name [8]. Dengue fever occurs in tropical and sub-tropical regions and usually increases in the hot and humid months. Dengue fever is not a new disease. In recent years, dengue fever has become a major international public health concern.

2.2.1 Overview of dengue fever

Dengue fever can be caused by any one of four types of dengue virus: DEN-1, DEN-2, DEN-3, and DEN-4 [8]. A person can be infected by at least two if not all four types at different times during his/her lifetime, but only once by the same type.

A human can get dengue virus infections from the bite of an infected *Aedes* mosquito. Mosquitoes become infected when they bite infected humans, and later transmit infection to other people they bite. Two main species of mosquito, *Aedes aegypti* (Figure 2.3) and *Aedes albopictus*, have been responsible for all cases of dengue transmitted in this country. Dengue is not contagious from person to person.

Figure 2.3: Female *Aedes aegypti* mosquito [10]

2.3.2 Symptoms of dengue fever

Symptoms of typical uncomplicated (classic) dengue usually start with fever within 4 to 7 days after been bitten by an infected mosquito and include

- High fever, up to 105°F
- Severe headache
- Retro-orbital (behind the eye) pain
- Severe joint and muscle pain
- Nausea and vomiting
- Rash

The rash may appear over most of our body 3 to 4 days after the fever begins, and then subsides after 1 to 2 days. We may get a second rash a few days later. Symptoms of dengue hemorrhagic fever include all of the symptoms of classic dengue such as marked damage to blood and lymph vessels and bleeding from the nose, gums, or under the skin, causing purplish bruises. This form of dengue disease can cause death.

Symptoms of dengue shock syndrome--the most severe form of dengue disease--include all of the symptoms of classic dengue and dengue hemorrhagic fever, such as

- Fluids leaking outside of blood vessels
- Massive bleeding
- Shock (very low blood pressure)

This form of the disease usually occurs in children (sometimes adults) experiencing their second dengue infection. It is sometimes fatal, especially in children and young adults [8].

Basically, dengue commences with high fever and other signs as listed above for 2 to 4 days. Then, the temperature drops rapidly and intense sweating takes place. After about a day with normal temperature and a feeling of well-being, the

temperature rises abruptly again. Rashes (small red bumps) show up on the arms, legs and the entire body simultaneously along with fever [9]. However, rashes rarely occur on the face. The palms of the hands and soles of the feet may be swollen and bright red. Although the patient may feel exhausted for several weeks, most cases of dengue take approximately one week to recover. Once a person recovers from dengue, he or she will have antibodies in their bloodstream which will prevent them from having a relapse for about a year.

Figure 2.4 shows a number of total cases of dengue in Malaysia from year 2000 to 2005. The highest cases occurred in 2002 till 2004. It shows that a lot of cases of dengue occurred in Malaysia.

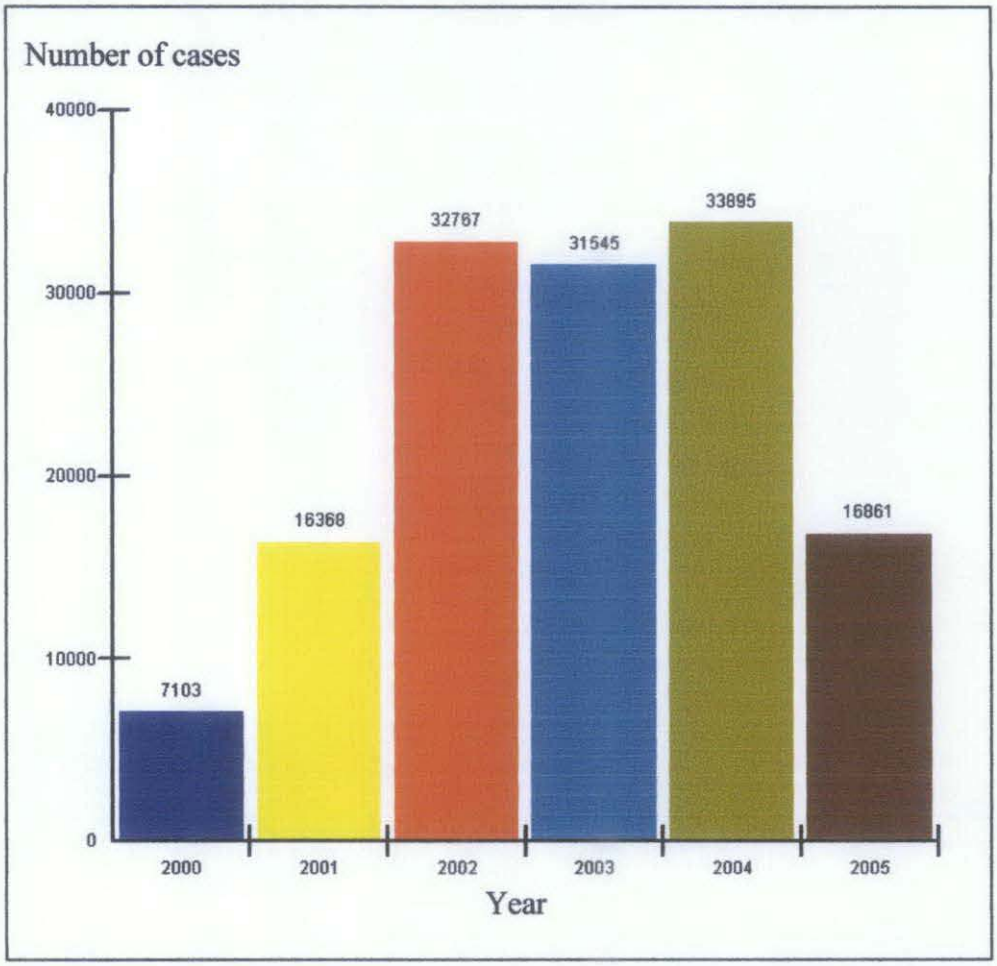


Figure 2.4: Number of total cases of dengue in Malaysia from year 2000 to 2005

2.3 Brain Temperature Tunnel

At the past, the most accurate temperature measurement of human is collected at rectum. It is the most accurate type of body temperature measurement [2]. But it is definitely, by far, not the most comfortable method to measure the body temperature of an individual.

However, the current study conducted by Yale researcher Dr. M. Marc Abreu has identified an area of the brain he calls the brain temperature tunnel, which will allow physicians to easily measure the body's core temperature by monitoring a patch of skin [5]. This area is the most accurate type of body temperature measurement. The discovery has the potential to prevent death from heat stroke and hypothermia, and detect infectious diseases such as dengue fever and SARS.

He found that a small area of skin near the eyes and the nose is the point of entry for the brain temperature tunnel as shown in Figure 2.5. His research shows that this area is connected to a thermal storage center in the brain, and the area has the thinnest skin and the highest amount of light energy. The brain temperature tunnel has enabled the creation of systems that enhance performance while maximizing safety in hot or cold temperatures and preventing dehydration.

Figure 2.5: Eye patch that measures the body's core temperature [5]

2.4 The sensor node system

In this project eZ430-RF2500 from Texas Instrument will be used as the sensor node. The eZ430-RF2500 is the world's smallest low-power wireless development tool. Figure 2.6 shows the eZ430-RF2500 development tool. The tool includes a USB emulator to program and debug the application in-system and 2.4-GHz wireless target board featuring the highly integrated MSP430F2274 ultra-low-power MCU. The MSP430F22x4 combines 16-MIPS performance with a 200-ksps 10-bit ADC and 2 op-amps and is paired with the CC2500 multi-channel RF transceiver designed for low-power wireless applications. Figure 2.7 shows eZ430-RF2500 flow connections of the USB debugging interface and the target boards.

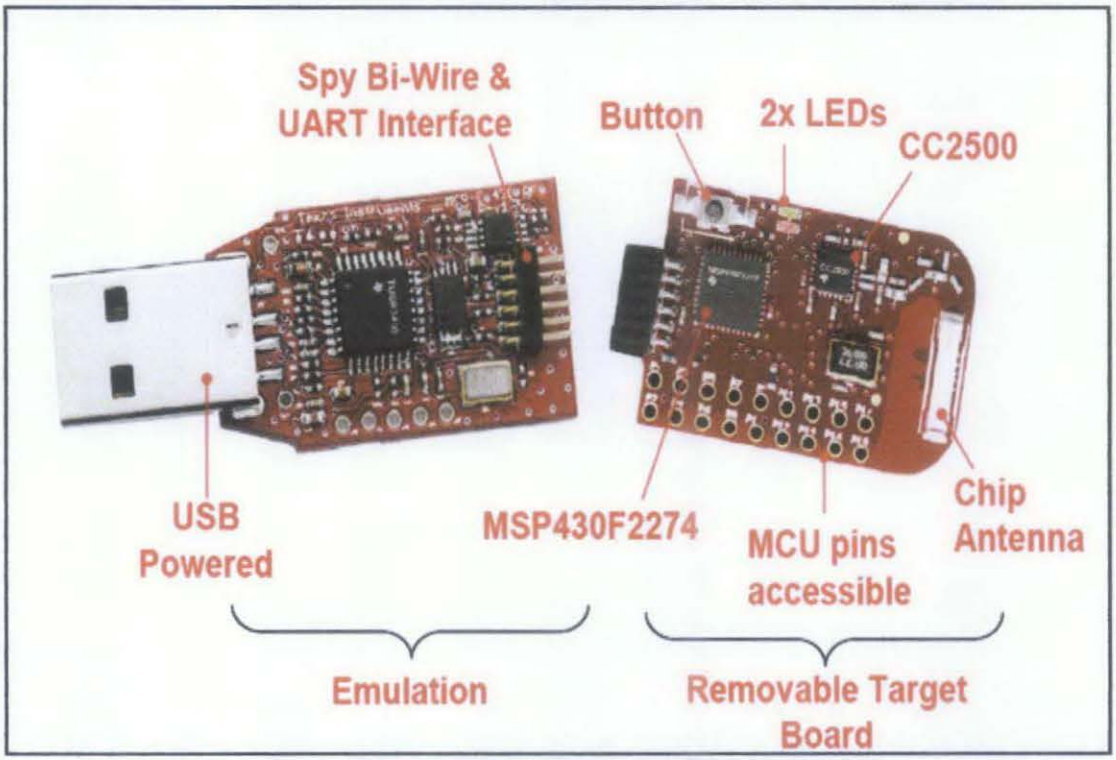


Figure 2.6: eZ430-RF2500 development tool

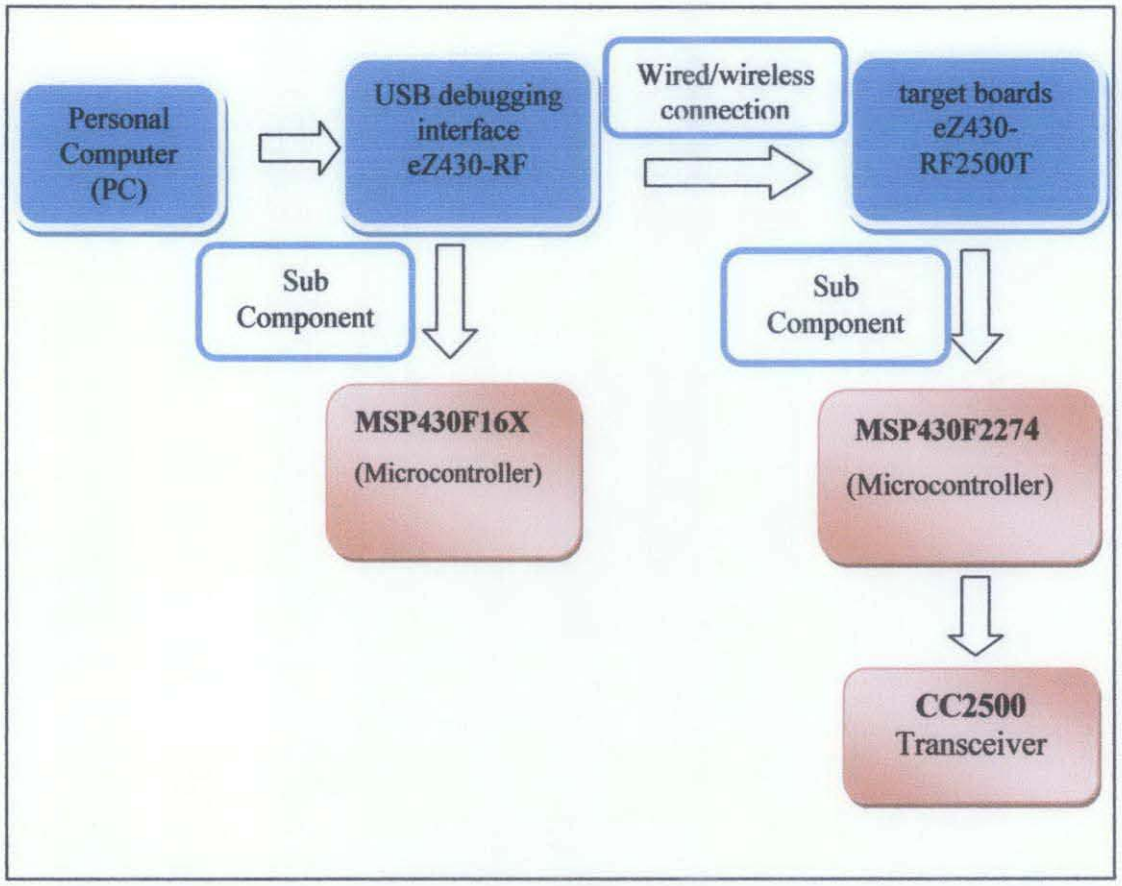


Figure 2.7: eZ430-RF2500 Flow Connections

2.4.1 Microcontroller (MSP430F2274)

The following is the specification of microcontroller (MSP430F2274):

- 200ksps 10-bit SAR ADC
- 2 built-in Op-Amps
- Watchdog timer, 16-bit Timer_A3 and B3
- USCI module supporting UART/LIN, (2) SPI, I2C, or IrDA
- Five low power modes drawing as little as 700nA in standby

2.4.2 Microcontroller's Architecture

A 16-bit RISC CPU, peripherals and flexible clock system are combined by using a von-Neumann common memory address bus (MAB) and memory data bus (MDB). Partnering a modern CPU with modular memory-mapped analog and digital peripherals, the MSP430 offers solutions for today's and tomorrow's mixed-signal applications.

2.4.2 Transceiver (CC2500)

The CC2500 is a low-cost 2.4 GHz transceiver designed for very low-power wireless applications. The circuit is intended for the 2400-2483.5 MHz ISM (Industrial, Scientific and Medical) and SRD (Short Range Device) frequency band. The RF transceiver is integrated with a highly configurable baseband modem. The modem supports various modulation formats and has a configurable data rate up to 500 kBaud. CC2500 provides extensive hardware support for packet handling, data buffering, burst transmissions, clear channel assessment, link quality indication, and wake-on-radio. The main operating parameters and the 64-byte transmit/receive FIFOs of CC2500 can be controlled via an SPI interface. In a typical system, the CC2500 will be used together with a microcontroller and a few additional passive components.

CHAPTER 3

METHODOLOGY

3.1 Project Methodology

This project is implemented into several stages to complete as shown in Figure 3.1

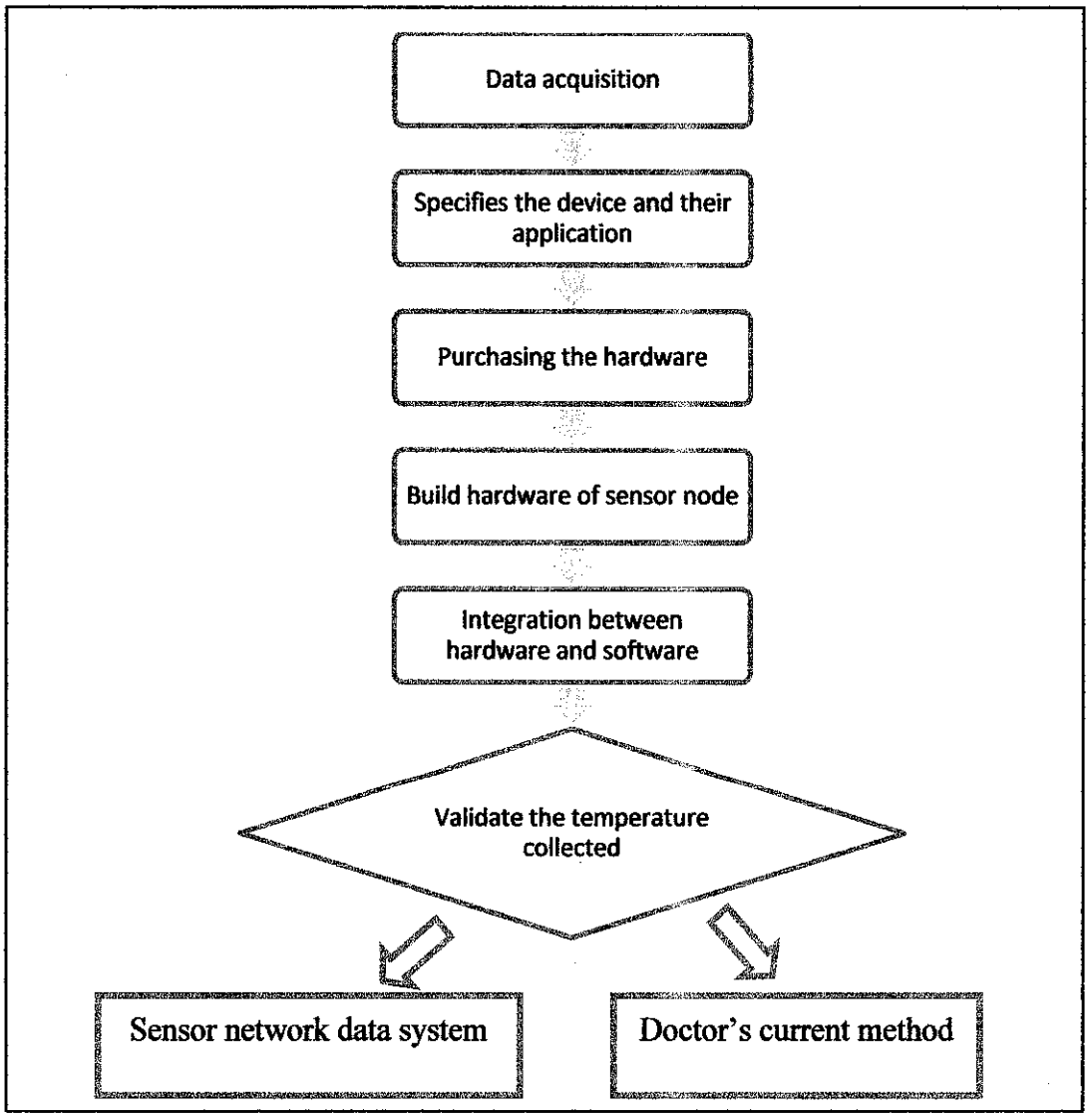


Figure 3.1: Project methodology

Figure 3.1 shows the stages involved in project methodology. The first stage involves data gathering process. This phase is the most crucial phase and provides the author all the useful and important knowledge for design progress. Books, Internet and Journals are the most practical sources for data gathering. After several researches have been done, the devices and their application for the system are specified. It was then followed by purchasing the hardware and the software for the system. Then the system is implemented by build the hardware and integrating it with the software applications. The final stage is validating process in which data from sensor network data system is compared to data from and doctor's current method. Several test plans will be implemented on a set of data to validate the accuracy of the system designed.

3.2 Project Architecture

Figure 3.2 indicates the proposed system architecture for this project. A sensor node (target board, Z430-RF2500) will be placed at each patient 1 and patient 2. USB debugging interface of Z430-RF2500 is being placed at host PC. The host PC will collect the temperature data and plot it into a graph. The graph plotted will be used during analysis process by the clinician for further diagnosis.

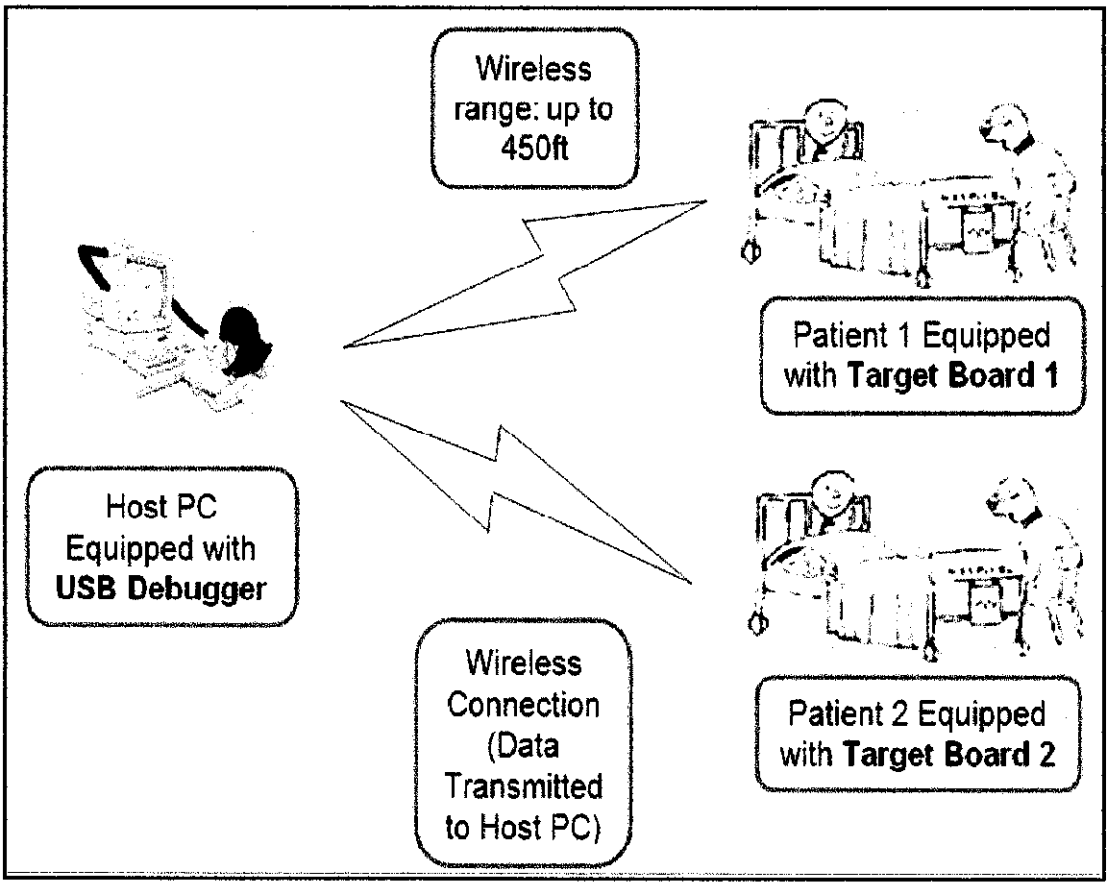


Figure 3.2: The proposed system architecture

3.2.1 Transmit and receive part

The sensor node from each patient will be connected wirelessly to the host PC. Temperature measurement will be taken by the sensor node and then transmitted wirelessly to the host PC (USB Interface). The receiver at USB debugging interface will receive the measurement data from the sensor node. The wireless range is up to 450ft for 10kbps. The transmission media that was being used by the system is radio frequency (RF).

3.2.2 Data analysis

The temperature data received at host PC will be stored and analysed in the database system, MySQL. Temperature data stored will be used to plot the graph and to be analysed for further use.

3.3 Tools for the sensor node (eZ430-Rf2500)

Tools used are IAR Embedded Workbench (shown in Figure 3.3) and Code Composer Essential. IAR Embedded Workbench for MSP430 is an Integrated Development Environment (IDE) for building and debugging embedded applications for MSP430 microcontrollers. The IDE includes a 4K limited C-Compiler/Unlimited Assembler/FET Debugger/ Simulator. The FET Debugger is a fully integrated debugger for source and disassembly level debugging with support for complex code and data breakpoints.

The Code Composer Essentials is used for integrated programming and debugging environment for the MSP430 ultra-low power microcontrollers.

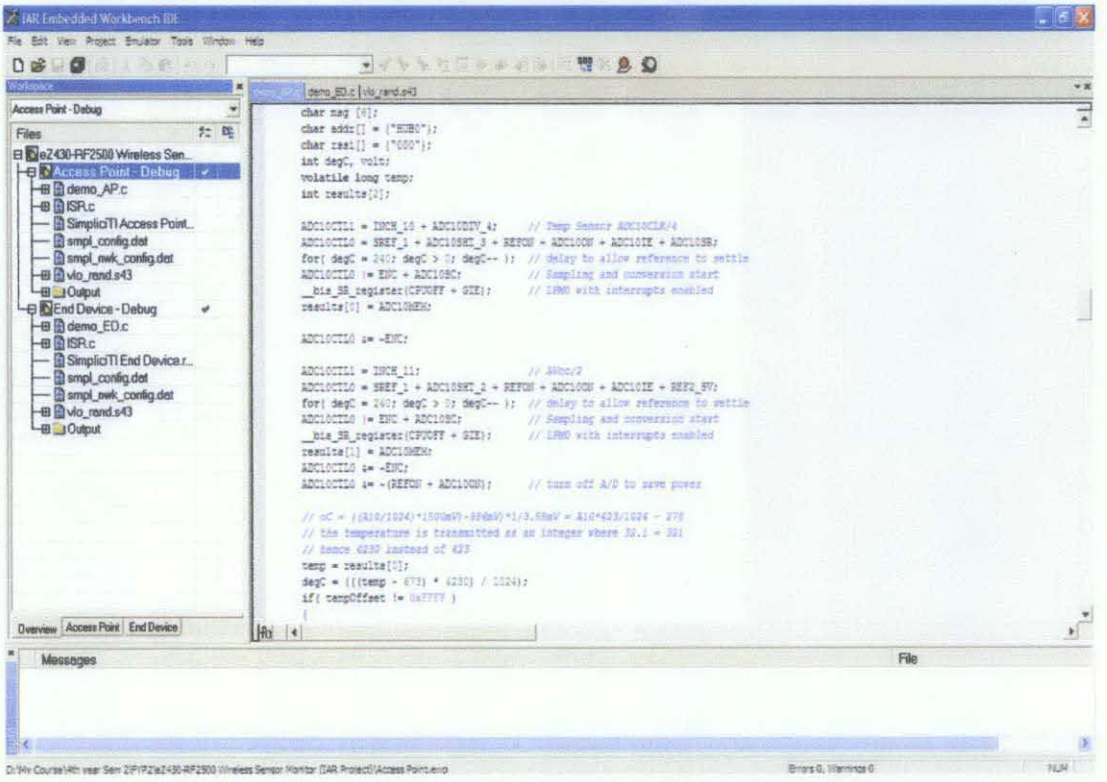


Figure 3.3: IAR Embedded Workbench Kickstart Workspace

CHAPTER 4

RESULTS AND DISCUSSION

In developing the system, the author has divided the works into four main parts which are:

i) Debugging and running the firmware
ii) Download the firmware into the boards
iii) Set up the system
iv) Gather and analyse the result
v) Validate result

Figure 4.1 shows the simple connection of the USB debugger and target board to the PC. Both of these devices are connected to each other through wireless connection. The medium for the connection is Radio Frequency (RF).

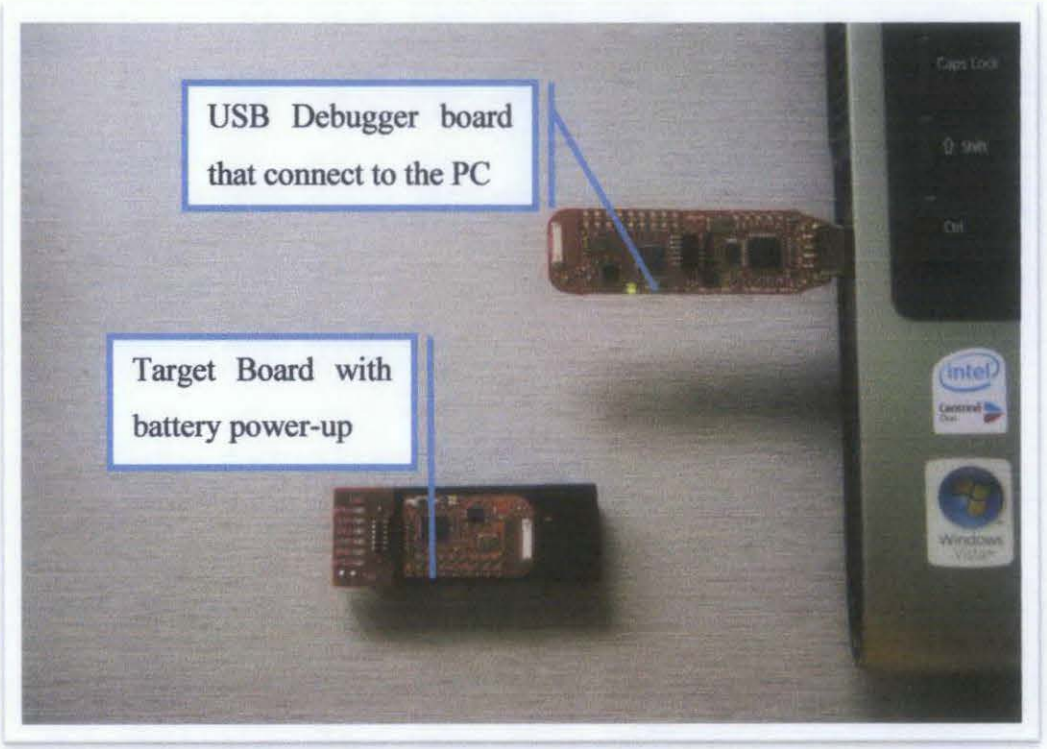


Figure 4.1: The USB Debugger and Target Board

Figure 4.2 shows that the flowchart for the Access Point's source code (refer to **Appendix A**). Firstly it will initializes both the communication between the MSP430 and the CC2500 radio and the LEDs/switches on the board that are to be used in the application. After hardware initialization, APs and EDs in the wireless sensor network create a random 4-byte address, write that address into flash memory for reuse on system reset, and then write over their default build-time device address. Since a SimpliciTI protocol AP identifies new devices on the network by their device addresses, storing this randomly-generated address in flash and checking this predefined location at device initialization ensures that an ED that has lost power or gets reset is always recognized as the same device (is given the same link ID) by the AP and that if the AP goes down itself, any ED that used the AP address to identify their respective SimpliciTI network sees the same AP on network reset. The random address is created using the results from the TI get Random Integer from VLO function inside the vlo_rand.s43 library file provided with the project. This library uses the rising edges of the very low frequency oscillator clock found in MSP430x2xx devices to trigger samples of a system clock that are then interpreted into a 4-byte device address. By changing the frequency of the system clock between triggers, the randomization of resulting device addresses is increased, and the user can be confident that two devices do not create the same network address.

The MCU_Init() function performs further MSP430-specific initializations that are necessary for the application. These include:

- The DCO and MCLK are set to run at 8 MHz
- Timer A is set to trigger interrupts at 1-second intervals.
- The universal serial communication interface (USCI) UART is initialized to communicate with the PC COM port to 9600 Baud and RX/TX, and interrupts are enabled.

Once the hardware initialization is complete, the TI splash screen is transmitted to the COM port on the PC and the program calls the SMPL_Init(sCB) network initialization function. The sCB parameter is a function pointer to a callback function that is executed within the interrupt service routine (ISR) upon packet reception by the AP

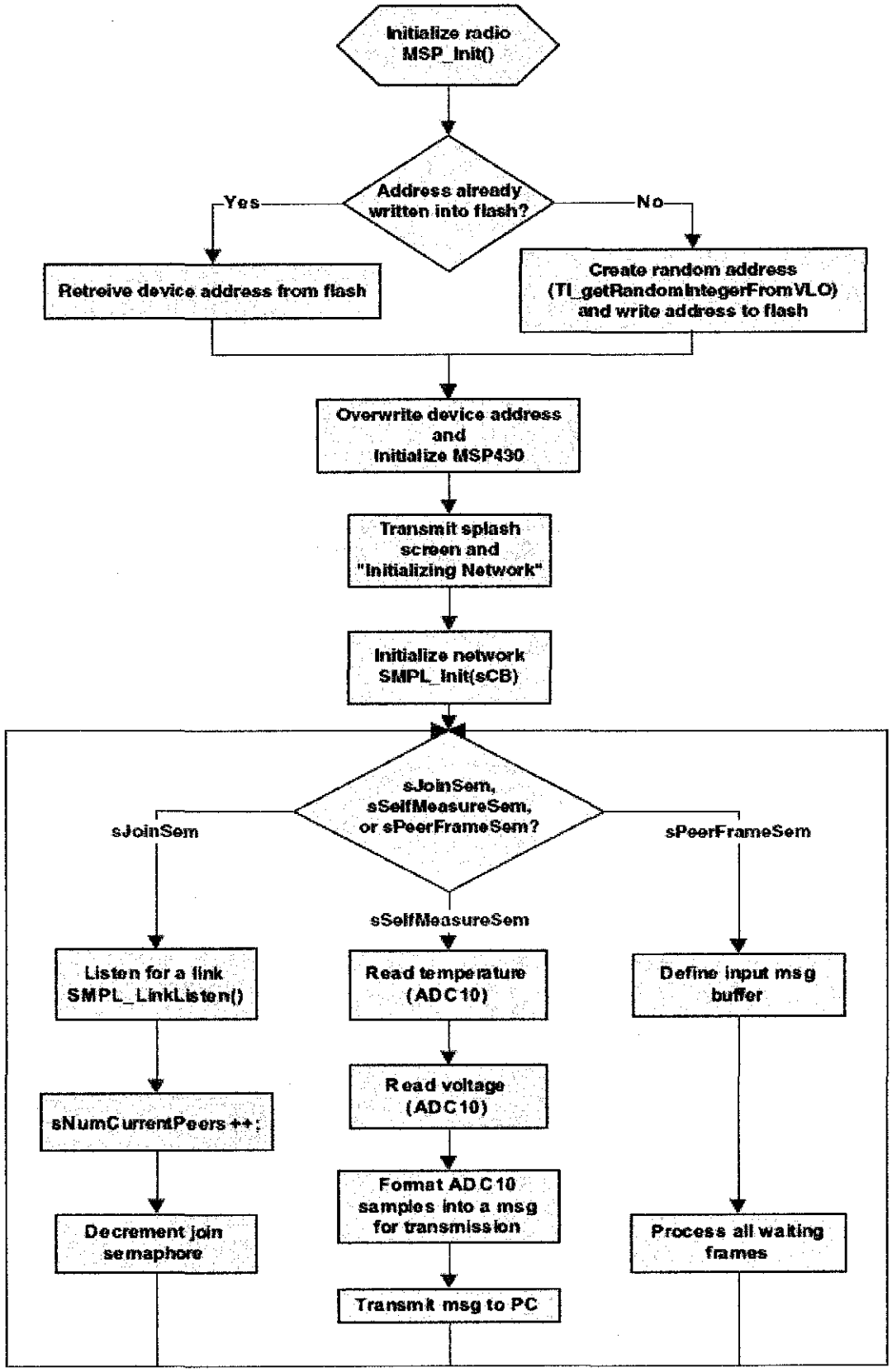


Figure 4.2: Flowchart for Access Point (AP)

Figure 4.3 shows that the flowchart for End Device's (ED) source codes (refer to **Appendix B**). EDs on the network exist purely to instantiate the network's application or intended function. In this instance, EDs initialize onto the network, then wake up once a second to sample and communicate their ambient temperature/battery voltage. A notable difference exists, however, in their method of initialization. The parameter an ED passes to its SMPL Init() function is a void pointer to the nonexistent callback function it would use to receive messages from peers.

This application has no need for a callback function that indicates the receipt of messages from other nodes on the network, because an ED's responsibilities are only to transmit its collected data. If an ED were to be capable of receiving messages, it could do so in two ways:

1) In the case that an ED sleeps and wakes up to receive messages from the AP, it would call SMPL_Receive() on wake-up to sample the AP output buffer for any stored messages.
2) In the case that an ED is always on and always listening for incoming messages, it would implement a callback function similar to the sCB function in the Access Point firmware.

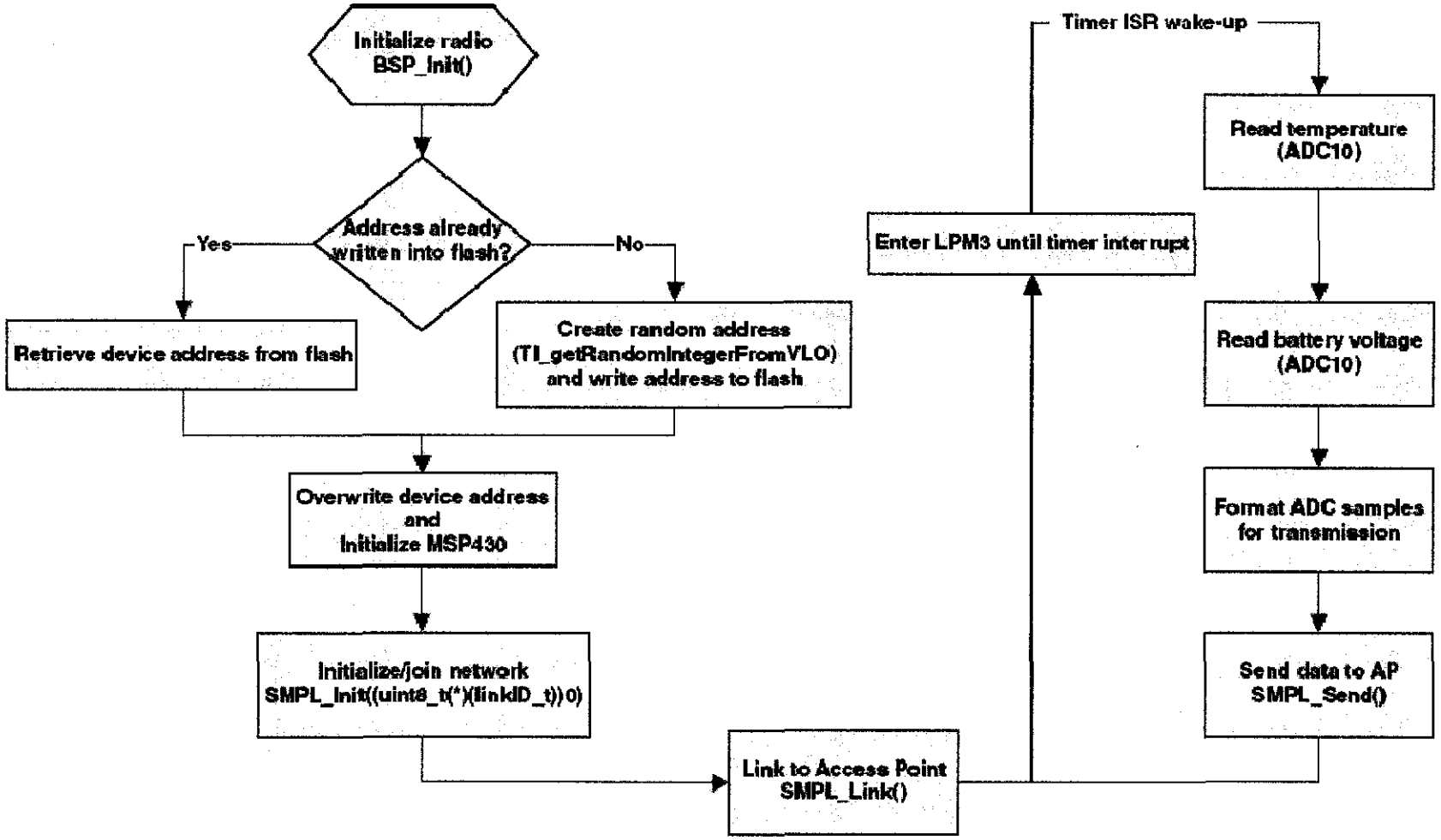


Figure 4.3: Flowchart for End Device (ED)

Figure 4.4 shows the some of the components of the prototype. The heat sink paste is used as a heat transfer medium between the sensor node and the copper sheet. In order to have minimal space in the case, the coin battery (CR2025) also used in the prototype. Figure 4.5 shows the placement of the target board inside the case.

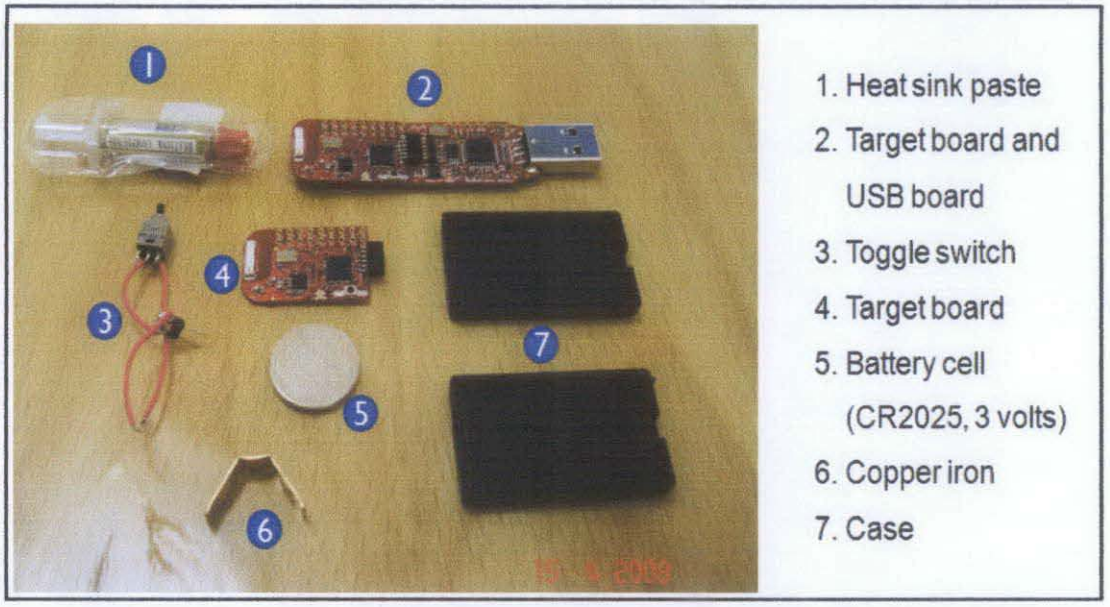


Figure 4.4: Components of the prototype.

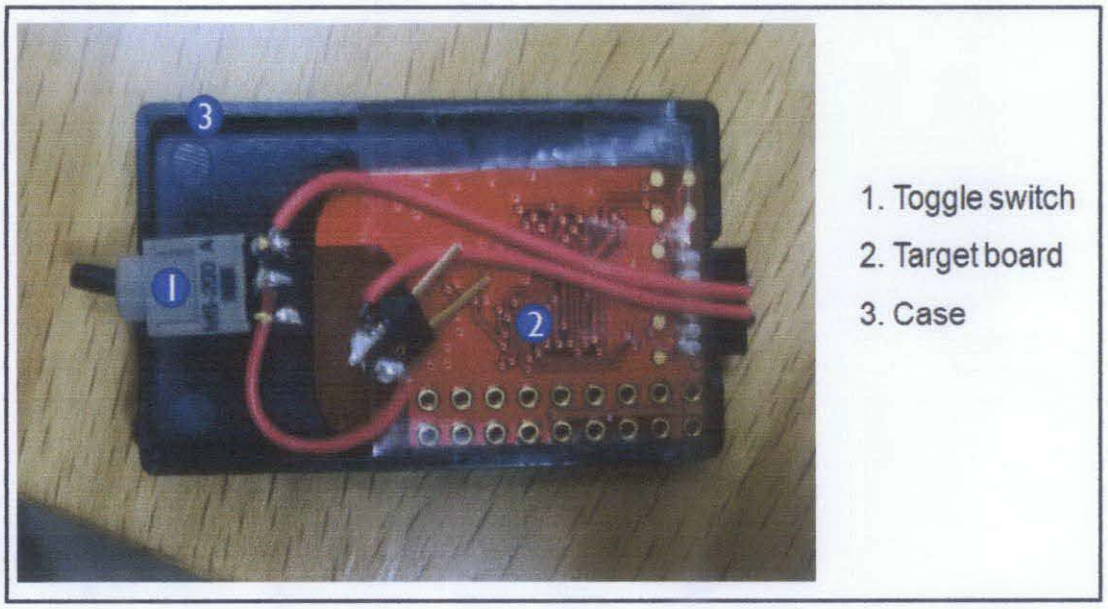


Figure 4.5: The target board been placed inside the case

Figure 4.6 shows the iron sheet be place on the prototype. As the best heat conductor, iron sheet is used in the prototype. It will be place at the area between nose and eyes as shown in Figure 2.5.

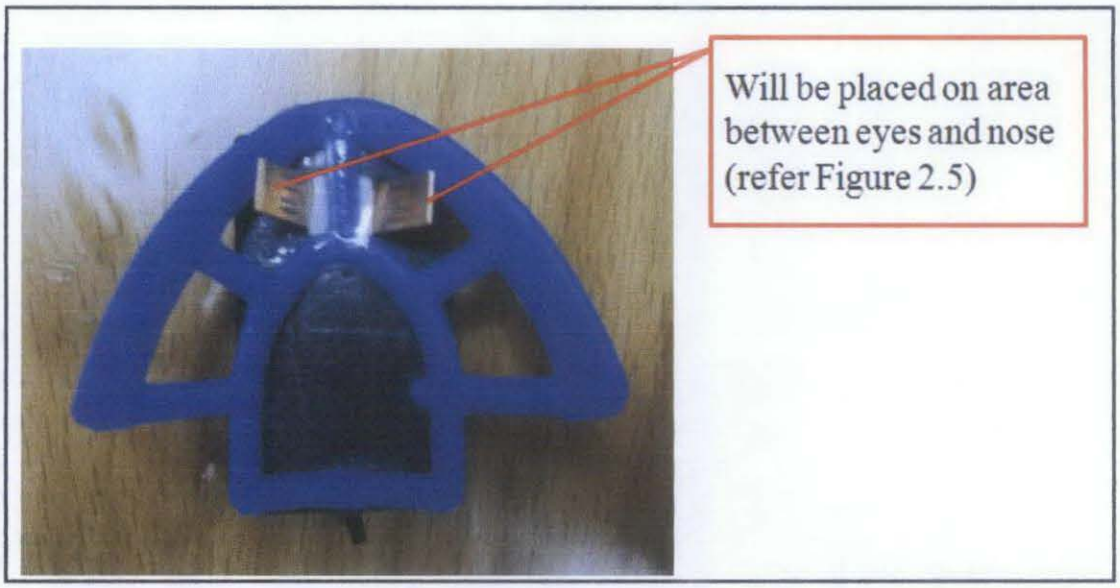


Figure 4.6: The prototype

Figure 4.7 shows the prototype of the sensor node used on the patient. The sensor node (target board) will collect temperature data on the patient and then transmit it to the host pc (USB debugging interface)

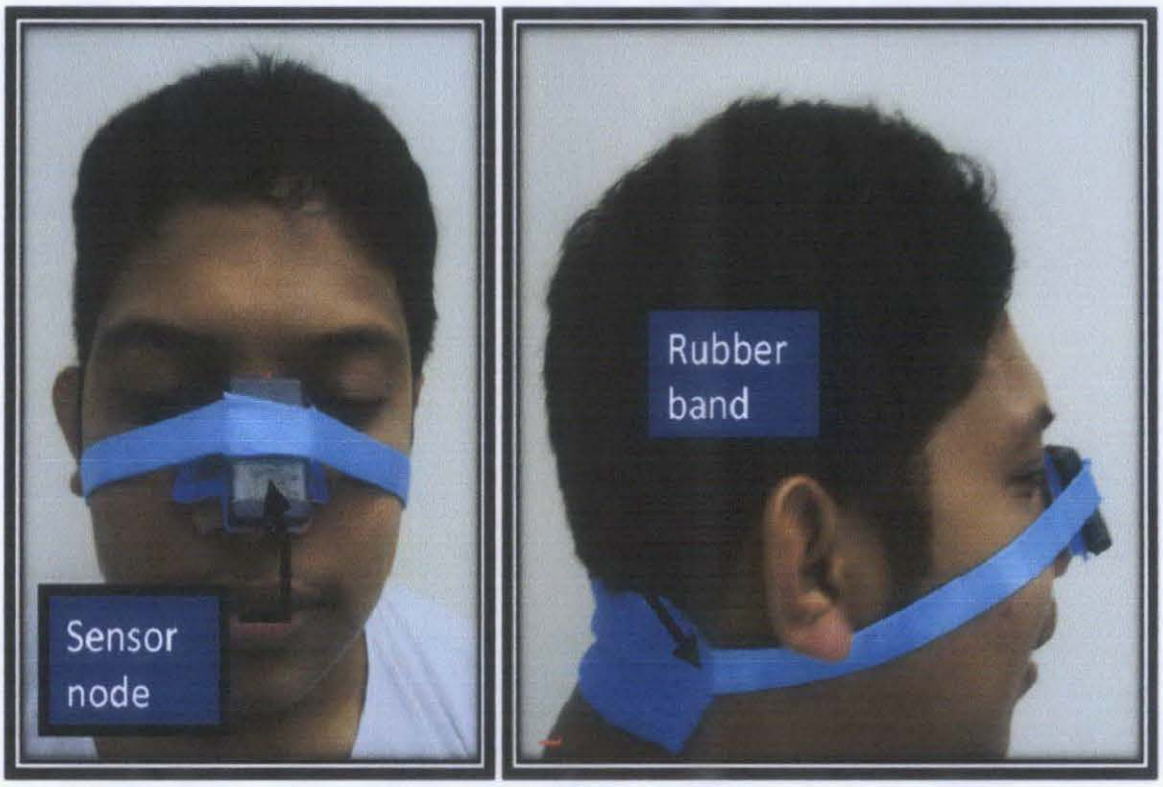


Figure 4.7: The prototype of the sensor node used on the patient.

Figure 4.8 shows the PC visualizer, the center node is the Access Point and the attached bubbles are the End Devices. The PC application displays the temperature of both the End Devices and Access Point. Additionally, the PC application is capable of simulating distance from its access point when the End Devices are moved.

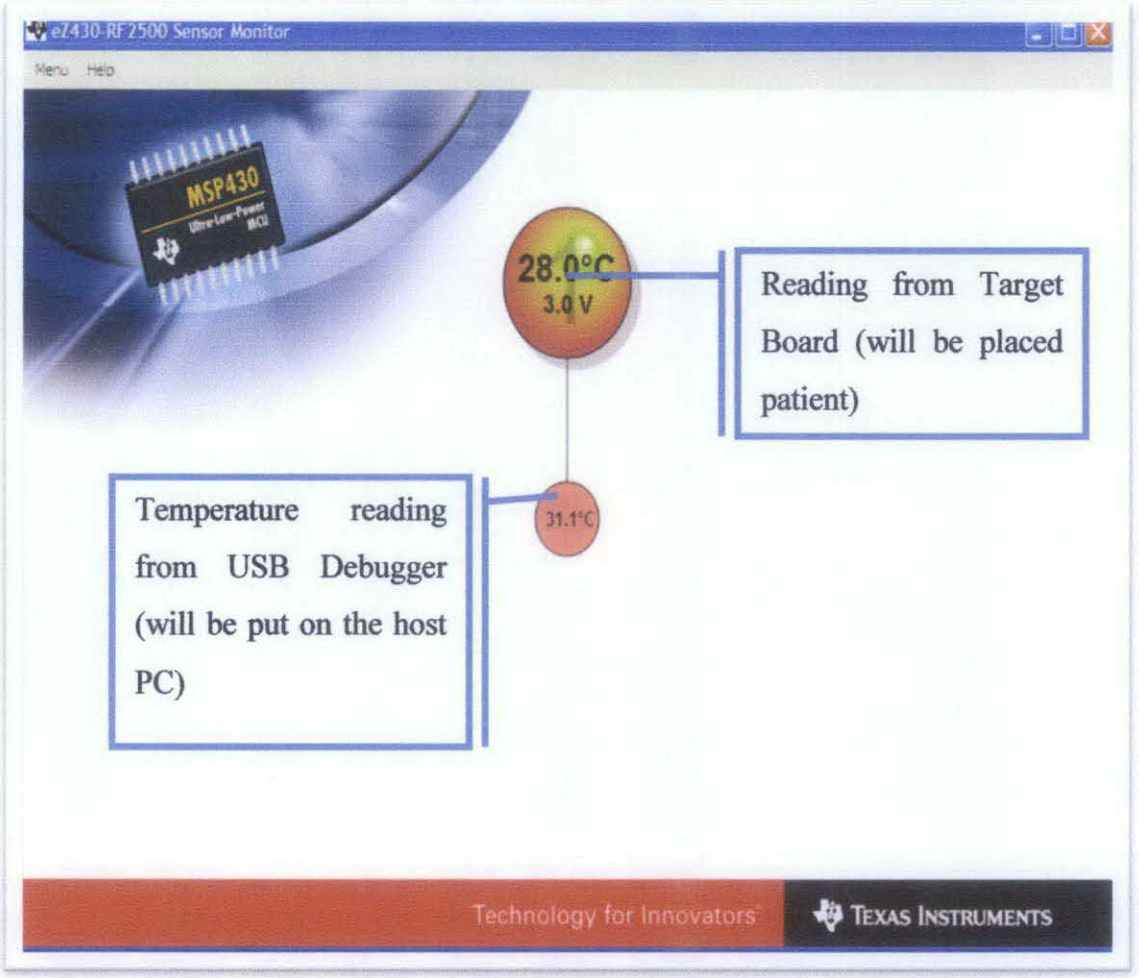


Figure 4.8: The PC Visualizer

Figure 4.9 it shows the temperature readings from the board. From here we can analyse the temperature readings and maintain the system.

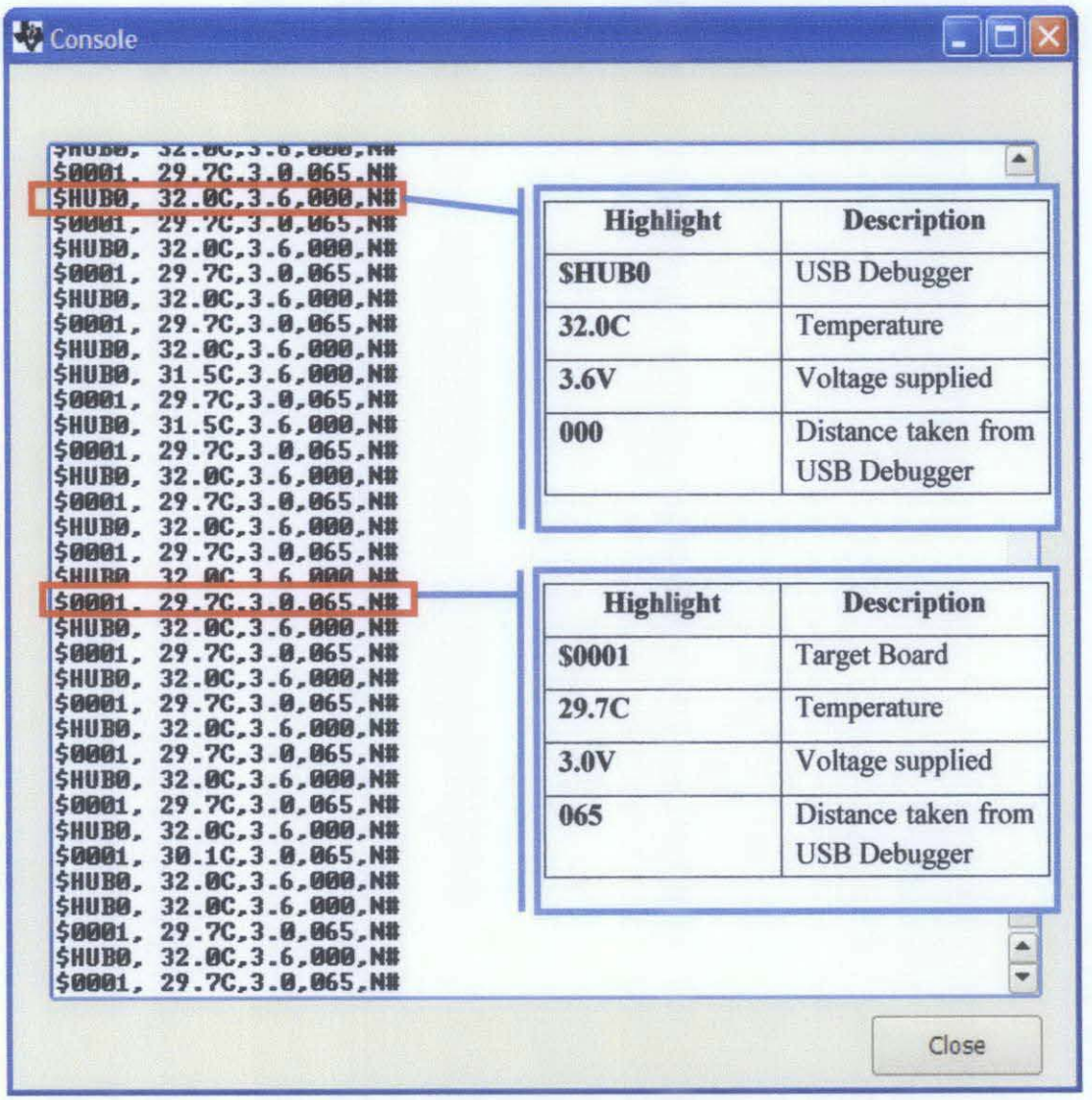


Figure 4.9: Temperature readings result from the console.

REFERENCES

[1] Campbell, Neil A. Biology, 3rd ed, California Benjamin Cummings, 1987:790.

[2] "Temperature, Body", World Book Encyclopaedia Chicago, Field Enterprise, 1996.

[3] Kaydos-Daniels SC et al. Body temperature monitoring and SARS fever hotline, Taiwan. Emerg Infect Dis [serial online] 2004 Feb .

[4] Maria G. Essig, Body Temperature, Blue Shield of California, 2007.

[5] Dr. M. Marc Abeau, "Discovery reveals new way to monitor body temperature", Yale Bulletin & Calendar, Volume 31, Number 33.

[6] I.F. Akyildiz et al, Wireless sensor networks: a survey, Georgia Institute of Technology, 2001.

[7] Jason Lester Hill, System Architecture for Wireless Sensor Networks, University of California, Berkeley, 2003.

[8] National Institutes of Allergy and Infectious Disease (NIAID), 27 October 2007, http://www3.niaid.nih.gov/healthscience/healthtopics/dengue/.

[9] Malaysia Medical Associates- Dengue and Dengue Haermorrhagic Fever. http://medicine.com.my/wp/?p=292.

[10] World Health Organization (WHO), 27 October 2007, http://www.who.int/globalatlas/DataQuery/default.asp.

[11] Texas Instrument, eZ430-RF2500 Development Tool, User's Guide, 2007.

[12] Crossbow Wireless Sensor Network, 27 October 2007, http://www.xbow.com/Products/productdetails.aspx?sid=160.

[13] Cirronet Zigbee, 27 October 2007, http://www.cirronet.com/zigbee.htm.

[14] PIC12F675 Tutorial, 21 March 2008, http://www.best-microcontroller-projects.com

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In this project, the wireless sensor network system will be used to help clinician to collect data. It will provide a better way to obtain more accurate body's core temperature of patient diagnosed with dengue. The wireless sensor network system can be used for monitoring many patients simultaneously.

With the sensor network, clinician can save more time and acquire more accurate data for diagnosis. By having digital data of their patient's we can develop a database system. With this database system we can have proper medical history of our patient which may important in the future works.

5.2 Recommendation

Increase number of sensor nodes to monitor large number of patients. Research on the wireless sensor network system which related to another disease could be conducted such as for Malaria and SARS. The accuracy and reliability of the system can be improved in the future by using more efficient database system.

APPENDIX A

Source code for Access Point (AP)

```
#include "bsp.h"
#include "bsp_leds.h"
#include "bsp_buttons.h"
#include "nwk_types.h"
#include "nwk_api.h"
#include "nwk_frame.h"
#include "nwk.h"
#include "TI_CC_spi.h"

#include  "msp430x22x4.h"
#include "vlo_rand.h"

#define MESSAGE_LENGTH 3
void toggleLED(uint8_t);
void TXString( char* string, int length );
void MCU_Init(void);
void transmitData(int addr, signed char rssi,  char msg[MESSAGE_LENGTH] );
void transmitDataString(char addr[4],char rssi[3], char msg[MESSAGE_LENGTH]);
void createRandomAddress();

//data for terminal output
__no_init volatile int tempOffset @ 0x10F4; // Temerature offset set at production

// reserve space for the maximum possible peer Link IDs
static linkID_t sLID[NUM_CONNECTIONS];
static uint8_t  sNumCurrentPeers;

// callback handler
static uint8_t sCB(linkID_t);

// work loop semaphores
static uint8_t sPeerFrameSem;
static uint8_t sJoinSem;
static uint8_t sSelfMeasureSem;

// mode data verbose = default, deg F = default
char verboseMode = 1;
char degCMode = 0;

void main (void)
{
```

```
bspIState_t intState;

WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
{  // delay loop to ensure proper startup before SimpliciTI increases DCO
 volatile int i;
 for(i = 0; i < 0xFFFF; i++){}
}
P1DIR = 0xFF;
P1OUT = 0x00;
P2DIR = 0x27;
P2OUT = 0x00;
P3DIR = 0xC0;
P3OUT = 0x00;
P4DIR = 0xFF;
P4OUT = 0x00;

BSP_Init();
createRandomAddress();
MCU_Init();
//Transmit splash screen and network init notification
TXString( (char*)splash, sizeof splash);
TXString( "\r\nInitializing Network....", 26 );

SMPL_Init(sCB);

// network initialized
TXString( "Done\r\n", 6);

// main work loop
while (1)
{
 // Wait for the Join semaphore to be set by the receipt of a Join frame from a
 // device that supports and End Device.

 if (sJoinSem && (sNumCurrentPeers < NUM_CONNECTIONS))
 {
  // listen for a new connection
  SMPL_LinkListen(&sLID[sNumCurrentPeers]);
  sNumCurrentPeers++;
  BSP_ENTER_CRITICAL_SECTION(intState);
  if (sJoinSem)
  {
   sJoinSem--;
  }
  BSP_EXIT_CRITICAL_SECTION(intState);
 }

 // if it is time to measure our own temperature...
 if(sSelfMeasureSem)
 {
  char msg [6];
  char addr[] = {"HUB0"};
  char rssi[] = {"000"};
  int degC, volt;
  volatile long temp;
  int results[2];
```

```
ADC10CTL1 = INCH_10 + ADC10DIV_4;    // Temp Sensor ADC10CLK/4
ADC10CTL0 = SREF_1 + ADC10SHT_3 + REFON + ADC10ON + ADC10IE + ADC10SR;
for( degC = 240; degC > 0; degC-- ); // delay to allow reference to settle
ADC10CTL0 |= ENC + ADC10SC;          // Sampling and conversion start
__bis_SR_register(CPUOFF + GIE);     // LPM0 with interrupts enabled
results[0] = ADC10MEM;

ADC10CTL0 &= ~ENC;

ADC10CTL1 = INCH_11;                 // AVcc/2
ADC10CTL0 = SREF_1 + ADC10SHT_2 + REFON + ADC10ON + ADC10IE + REF2_5V;
for( degC = 240; degC > 0; degC-- ); // delay to allow reference to settle
ADC10CTL0 |= ENC + ADC10SC;          // Sampling and conversion start
__bis_SR_register(CPUOFF + GIE);     // LPM0 with interrupts enabled
results[1] = ADC10MEM;
ADC10CTL0 &= ~ENC;
ADC10CTL0 &= ~(REFON + ADC10ON);     // turn off A/D to save power

// oC = ((A10/1024)*1500mV)-986mV)*1/3.55mV = A10*423/1024 - 278
// the temperature is transmitted as an integer where 32.1 = 321
// hence 4230 instead of 423
temp = results[0];
degC = (((temp - 673) * 4230) / 1024);
if( tempOffset != 0xFFFF )
{
  degC += tempOffset;
}

temp = results[1];
volt = (temp*25)/512;

msg[0] = degC&0xFF;
msg[1] = (degC>>8)&0xFF;
msg[2] = volt;
transmitDataString(addr, rssi, msg );

toggleLED(1);
sSelfMeasureSem = 0;
}

// Have we received a frame on one of the ED connections?
// No critical section -- it doesn't really matter much if we miss a poll
if (sPeerFrameSem)
{
 uint8_t    msg[MAX_APP_PAYLOAD], len, i;

 // process all frames waiting
 for (i=0; i<sNumCurrentPeers; ++i)
 {
   if (SMPL_Receive(sLID[i], msg, &len) == SMPL_SUCCESS)
   {
     ioctlRadioSiginfo_t sigInfo;
     sigInfo.port = sLID[i];
     SMPL_Ioctl(IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_SIGINFO, (void *)&sigInfo);
     transmitData( i, (signed char)sigInfo.rssi, (char*)msg );
```

```
        toggleLED(2);
        //processMessage(msg, len);
        BSP_ENTER_CRITICAL_SECTION(intState);
        sPeerFrameSem--;
        BSP_EXIT_CRITICAL_SECTION(intState);
      }
    }
   }

 }

}


void createRandomAddress()
{
 volatile unsigned int rand;
 addr_t lAddr;

 do
 {
  rand = TI_getRandomIntegerFromVLO();    // first byte can not be 0x00 of 0xFF
 }
 while( (rand & 0xFF00)==0xFF00 || (rand & 0xFF00)==0x0000 );

  lAddr.addr[0]=(rand>>8) & 0xFF;
  lAddr.addr[1]=rand & 0xFF;
  rand = TI_getRandomIntegerFromVLO();
  lAddr.addr[2]=(rand>>8) & 0xFF;
  lAddr.addr[3]=rand & 0xFF;
  SMPL_Ioctl(IOCTL_OBJ_ADDR, IOCTL_ACT_SET, &lAddr);
}

/*------------------------------------------------------------------------------
*
------------------------------------------------------------------------------*/
void transmitData(int addr, signed char rssi,  char msg[MESSAGE_LENGTH] )
{
 char addrString[4];
 char rssiString[3];
 volatile signed int rssi_int;
 if( rssi < 0 )
 {
  _NOP();
 }
 addrString[0] = '0';
 addrString[1] = '0';
 addrString[2] = '0'+(((addr+1)/10)%10);
 addrString[3] = '0'+((addr+1)%10);
 rssi_int = (signed int) rssi;
 rssi_int = rssi_int+128;
 rssi_int = (rssi_int*100)/256;
 rssiString[0] = '0'+(rssi_int%10);
 rssiString[1] = '0'+((rssi_int/10)%10);
 rssiString[2] = '0'+((rssi_int/100)%10);
 if( rssi < 0 || rssiString[0] == '/' || rssiString[1] == '/' || rssiString[2] == '/')
```

```
 {
  _NOP();
 }
 rssi++;
 rssi_int++;
 transmitDataString( addrString, rssiString, msg );
}
/*---------------------------------------------------------------------------
*
---------------------------------------------------------------------------*/
void transmitDataString(char addr[4],char rssi[3], char msg[MESSAGE_LENGTH] )
{
 char temp_string[] = {" XX.XC"};
 int temp = msg[0] + (msg[1]<<8);

 if( !degCMode )
 {
  temp = (((float)temp)*1.8)+320;
  temp_string[5] = 'F';
 }
 if( temp < 0 )
 {
  temp_string[0] = '-';
  temp = temp * -1;
 }
 temp_string[4] = '0'+(temp%10);
 temp_string[2] = '0'+((temp/10)%10);
 temp_string[1] = '0'+((temp/100)%10);

 if( verboseMode )
 {
  char output_verbose[] = {"\r\nNode:XXXX,Temp:-XX.XC,Battery:X.XV,Strength:XXX%,RE:no
          "};

  output_verbose[46] = rssi[2];
  output_verbose[47] = rssi[1];
  output_verbose[48] = rssi[0];

  output_verbose[17] = temp_string[0];
  output_verbose[18] = temp_string[1];
  output_verbose[19] = temp_string[2];
  output_verbose[20] = temp_string[3];
  output_verbose[21] = temp_string[4];
  output_verbose[22] = temp_string[5];

  output_verbose[32] = '0'+(msg[2]/10)%10;
  output_verbose[34] = '0'+(msg[2]%10);
  output_verbose[7] = addr[0];
  output_verbose[8] = addr[1];
  output_verbose[9] = addr[2];
  output_verbose[10] = addr[3];
  TXString(output_verbose, sizeof output_verbose );

 }
 else
 {
```

```
  char output_short[] = {"\r\n$ADDR,-XX.XC,V.C,RSI,N#"};

  output_short[19] = rssi[2];
  output_short[20] = rssi[1];
  output_short[21] = rssi[0];


  output_short[8] = temp_string[0];
  output_short[9] = temp_string[1];
  output_short[10] = temp_string[2];
  output_short[11] = temp_string[3];
  output_short[12] = temp_string[4];
  output_short[13] = temp_string[5];

  output_short[15] = '0'+(msg[2]/10)%10;
  output_short[17] = '0'+(msg[2]%10);
  output_short[3] = addr[0];
  output_short[4] = addr[1];
  output_short[5] = addr[2];
  output_short[6] = addr[3];
  TXString(output_short, sizeof output_short );

 }
}
/*------------------------------------------------------------------------------
*
------------------------------------------------------------------------------*/
void TXString( char* string, int length )
{
 int pointer;
 for( pointer = 0; pointer < length; pointer++)
 {
  volatile int i;
  UCA0TXBUF = string[pointer];
  while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
 }
}
/*------------------------------------------------------------------------------
*
------------------------------------------------------------------------------*/
void MCU_Init()
{
 BCSCTL1 = CALBC1_8MHZ;                  // Set DCO
 DCOCTL = CALDCO_8MHZ;

 BCSCTL3 |= LFXT1S_2;                    // LFXT1 = VLO
 TACCTL0 = CCIE;                         // TACCR0 interrupt enabled
 TACCR0 = 12000;                         // ~1 second
 TACTL = TASSEL_1 + MC_1;                // ACLK, upmode

 P3SEL |= 0x30;                          // P3.4,5 = USCI_A0 TXD/RXD
 UCA0CTL1 = UCSSEL_2;                    // SMCLK
 UCA0BR0 = 0x41;                         // 9600 from 8Mhz
 UCA0BR1 = 0x3;
 UCA0MCTL = UCBRS_2;
 UCA0CTL1 &= ~UCSWRST;                   // **Initialize USCI state machine**
```

```
  IE2 |= UCA0RXIE;                  // Enable USCI_A0 RX interrupt
  __enable_interrupt();

}
/*-------------------------------------------------------------------------
*
-------------------------------------------------------------------------*/
void toggleLED(uint8_t which)
{
  if (1 == which)
  {
    BSP_TOGGLE_LED1();
  }
  else if (4 == which)
  {
    BSP_TOGGLE_LED3();
  }
  else
  {
    BSP_TOGGLE_LED2();
  }
  return;
}
/*-------------------------------------------------------------------------
* Runs in ISR context. Reading the frame should be done in the
* application thread not in the ISR thread.
-------------------------------------------------------------------------*/
static uint8_t sCB(linkID_t port)
{
  if (port >= PORT_BASE_NUMBER)
  {
    sPeerFrameSem++;
  }
  else if (SMPL_PORT_JOIN == port)
  {
    sJoinSem++;
  }

  // leave frame to be read by application.
  return 0;
}

/*-------------------------------------------------------------------------
* ADC10 interrupt service routine
-------------------------------------------------------------------------*/
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF);        // Clear CPUOFF bit from 0(SR)
}
/*-------------------------------------------------------------------------
* Timer A0 interrupt service routine
-------------------------------------------------------------------------*/
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
```

```
  sSelfMeasureSem = 1;
}

/*-------------------------------------------------------------------------
* USCIA interrupt service routine
-------------------------------------------------------------------------*/
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
  char rx = UCA0RXBUF;
  if ( rx == 'V' || rx == 'v' )
  {
    verboseMode = 1;
  }
  else if ( rx == 'M' || rx == 'm' )
  {
    verboseMode = 0;
  }
  else if ( rx == 'F' || rx == 'f' )
  {
    degCMode = 0;
  }
  else if ( rx == 'C' || rx == 'c' )
  {
    degCMode = 1;
  }

}
```

APPENDIX B

Source code for End Device (Target Board)

```
#include "bsp.h"
#include "nwk_types.h"
#include "nwk_api.h"
#include "TI_CC_spi.h"
#include "bsp_leds.h"
#include "bsp_buttons.h"
#include "vlo_rand.h"

#include "ccrf_regDefs.h"

void linkTo(void);
void MCU_Init(void);

void toggleLED(uint8_t);

__no_init volatile int tempOffset @ 0x10F4; // Temerature offset set at production

void createRandomAddress();

void main (void)
{
 WDTCTL = WDTPW + WDTHOLD;              // Stop WDT
 {  // delay loop to ensure proper startup before SimpliciTI increases DCO
  volatile int i;
  for(i = 0; i < 0xFFFF; i++){}
 }
 // SimpliciTI will change port pin settings as well
 P1DIR = 0xFF;
 P1OUT = 0x00;
 P2DIR = 0x27;
 P2OUT = 0x00;
 P3DIR = 0xC0;
 P3OUT = 0x00;
 P4DIR = 0xFF;
 P4OUT = 0x00;

 BSP_Init();

 createRandomAddress();              // set Random device address each startup
```

```
  BCSCTL1 = CALBC1_8MHZ;                // Set DCO after random function
  DCOCTL = CALDCO_8MHZ;
  BCSCTL3 |= LFXT1S_2;                  // LFXT1 = VLO
  TACCTL0 = CCIE;                       // TACCR0 interrupt enabled
  TACCR0 = 12000;                       // ~ 1 sec
  TACTL = TASSEL_1 + MC_1;              // ACLK, upmode

  // keep trying to join until successful. toggle LEDS to indicate that
  // joining has not occurred. LED3 is red but labeled LED 4 on the EXP
  // board silkscreen. LED1 is green.
  while (SMPL_NO_JOIN == SMPL_Init((uint8_t (*)(linkID_t))0))
  {
    toggleLED(1);
    toggleLED(2);
    __bis_SR_register(LPM3_bits + GIE);     // LPM3 with interrupts enabled
  }
  // unconditional link to AP which is listening due to successful join.
  linkTo();

}

void createRandomAddress()
{
  volatile unsigned int rand;
  addr_t lAddr;

  do
  {
    rand = TI_getRandomIntegerFromVLO();    // first byte can not be 0x00 of 0xFF
  }
  while( (rand & 0xFF00)==0xFF00 || (rand & 0xFF00)==0x0000 );

    lAddr.addr[0]=(rand>>8) & 0xFF;
    lAddr.addr[1]=rand & 0xFF;
    rand = TI_getRandomIntegerFromVLO();
    lAddr.addr[2]=(rand>>8) & 0xFF;
    lAddr.addr[3]=rand & 0xFF;
    SMPL_Ioctl(IOCTL_OBJ_ADDR, IOCTL_ACT_SET, &lAddr);
}

void linkTo()
{
  linkID_t linkID1;
  uint8_t  msg[3];

  // keep trying to link...
  while (SMPL_NO_LINK == SMPL_Link(&linkID1))
  {
    __bis_SR_register(LPM3_bits + GIE);     // LPM3 with interrupts enabled
    toggleLED(1);
  }

  // Turn off all LEDs
  if (BSP_LED1_IS_ON())
  {
```

```
 toggleLED(1);
}
if (BSP_LED2_IS_ON())
{
 toggleLED(2);
}
while (1)
{
 volatile long temp;
 int degC, volt;
 int results[2];
 SMPL_Ioctl( IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_SLEEP, "" );
 __bis_SR_register(LPM3_bits+GIE);      // LPM3 with interrupts enabled
 SMPL_Ioctl( IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_AWAKE, "" );

 toggleLED(2);
 ADC10CTL1 = INCH_10 + ADC10DIV_4;      // Temp Sensor ADC10CLK/4
 ADC10CTL0 = SREF_1 + ADC10SHT_3 + REFON + ADC10ON + ADC10IE + ADC10SR;
 for( degC = 240; degC > 0; degC-- );   // delay to allow reference to settle
 ADC10CTL0 |= ENC + ADC10SC;            // Sampling and conversion start
 __bis_SR_register(CPUOFF + GIE);       // LPM0 with interrupts enabled
 results[0] = ADC10MEM;

 ADC10CTL0 &= ~ENC;

 ADC10CTL1 = INCH_11;                   // AVcc/2
 ADC10CTL0 = SREF_1 + ADC10SHT_2 + REFON + ADC10ON + ADC10IE + REF2_5V;
 for( degC = 240; degC > 0; degC-- );   // delay to allow reference to settle
 ADC10CTL0 |= ENC + ADC10SC;            // Sampling and conversion start
 __bis_SR_register(CPUOFF + GIE);       // LPM0 with interrupts enabled
 results[1] = ADC10MEM;
 ADC10CTL0 &= ~ENC;
 ADC10CTL0 &= ~(REFON + ADC10ON);       // turn off A/D to save power

 // oC = ((A10/1024)*1500mV)-986mV)*1/3.55mV = A10*423/1024 - 278
 // the temperature is transmitted as an integer where 32.1 = 321
 // hence 4230 instead of 423
 temp = results[0];
 degC = ((temp - 673) * 4230) / 1024;
 if( tempOffset != 0xFFFF )
 {
  degC += tempOffset;
 }
 /*message format,  UB = upper Byte, LB = lower Byte
 -------------------------------
 |degC LB | degC UB |  volt LB |
 -------------------------------
   0       1       2
 */

 temp = results[1];
 volt = (temp*25)/512;
 msg[0] = degC&0xFF;
 msg[1] = (degC>>8)&0xFF;
 msg[2] = volt;
```

```
    SMPL_Send(linkID1, msg, sizeof(msg));
    toggleLED(2);
  }
}

void toggleLED(uint8_t which)
{
  if (1 == which)
  {
    BSP_TOGGLE_LED1();
  }
  else if (2 == which)
  {
    BSP_TOGGLE_LED2();
  }
  return;
}

/*-------------------------------------------------------------------------
* ADC10 interrupt service routine
-------------------------------------------------------------------------*/
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF);        // Clear CPUOFF bit from 0(SR)
}
/*-------------------------------------------------------------------------
* Timer A0 interrupt service routine
-------------------------------------------------------------------------*/
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
  __bic_SR_register_on_exit(LPM3_bits);       // Clear LPM3 bit from 0(SR)
}
```