Monitoring System for a Laundry System with eZ430-RF2500 Development Kit

E90 Senior Design Project

David Kwon
Department of Engineering
Advised by Professor Molter

May 08, 2009

Abstract

Monitoring system for a laundry room can be used effectively by students from wasting their valuable resources such as time, energy, and focus. The system will make the current status of washers and dryers available online so that students will be prevented from making multiple meaningless trips to a laundry room. This report outlines all the details of circuit designs that are used in this project as well as eZ430-RF2500 development kit, which is a crucial tool that the project uses to transmit information wirelessly.

Table of Contents

1. Introduction

Life is tough for college students. Academic work makes life harder. Practical affairs, though usually not very mentally challenging, are also trying because of the time they consume. One of these affairs is laundry. Doing laundry is not hard, but finding available washers and dryers in a laundry room becomes a daunting task during busy nights and weekends. Many students sometimes find themselves stuck in the laundry room with unavailable washers and dryers and have to make multiple trips to the room to find out if they become available. Some students, including the author of this paper, have to bring their laundry basket back to the dormitory, which makes the overall experience of doing laundry less enjoyable. Of course, not all students are frequently subjected to laundry room inconveniences, but most students have experienced this problem during their academic years.

Figure 1. A picture of the LED indicator panels on washer and dryer. Top) Washer Display Panel, Bottom) Dryer Panel

In order to avoid such an event, a monitoring system becomes necessary. Such a system will help students avoid making multiple meaningless trips to the laundry room, thus saving them invaluable time and effort. However, how can the system be built without physically tampering washers and dryers? The main purpose of this project is to implement a system that will not require a user to tamper with the devices. An external wireless device will be built by monitoring changes that occur externally while the washers and dryers are in operation. For example, the washers and dryers that are installed in Parrish have LED lights as seen in Figure 1. These lights indicate whether the machines are on or off. The

lights also indicate in which washing cycle the washer is currently operating or in which drying mode the dryer is currently running. By monitoring these LED lights, one can build a laundry monitoring system without physically tampering the washers and dryers. In order to monitor the LED light changes, an OPT101, a monolithic photodiode and single-supply transimpedance amplifier, was used, and eZ430-RF2500 development kit was used as a mode of wireless data transmission.

This report will include an explanation of how the laundry room monitoring system was designed. The report will briefly go over several assumptions made about the system and write about the electronic components and circuitry designs that were used in this project. Then, conclusion and suggested future work will be presented followed by results.

2. A Monitoring System Design

2.1. Designing a System Based on Several Assumptions

In this project, an assumption is made that all washers and dryers that are installed in Swarthmore dormitories are the same. This means that there is an explicit underlying assumption that all washers and dryers on Swarthmore campus have the exact LED light panels. Another assumption is the LED lights installed on the washers and dryers are functional. It is possible that some LED lights are out of service. Since the project depends on the LED lights, non-functioning LED lights can be problematic. For the purpose of this project, a consideration of such lights will be disregarded when designing the monitoring system.

2.2. Project Design

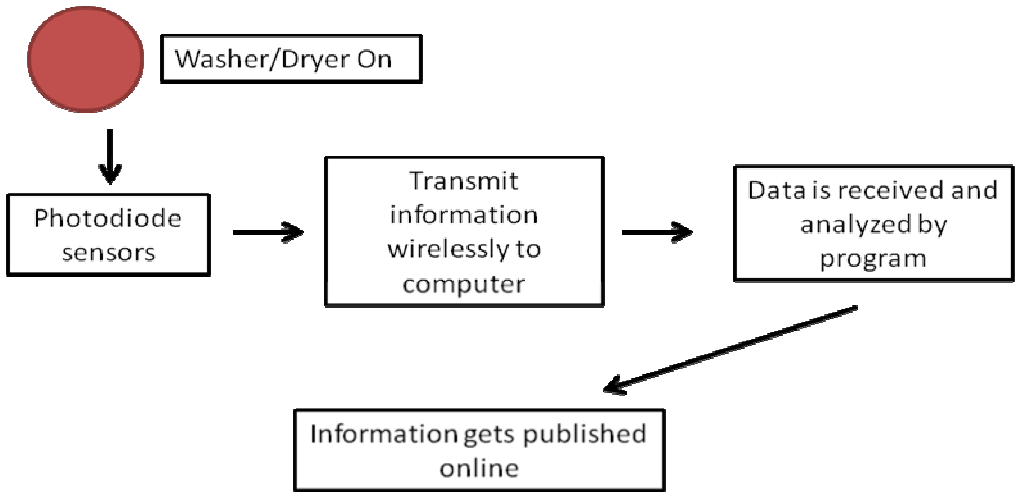

Figure 2. A basic scheme of the monitoring system

A basic scheme of the monitoring system is seen in Figure 2. The trigger mechanism in this system is the LED light. When the washers and dryers are off, the LED lights are off as well. Once the machines start to operate, so do the LED lights. In order to detect this light change, a photodiode is used in this project. It will be installed near the LED lights to monitor them. Since each machine has three LED lights, three photodiode sensors will be needed. Then, the outcome of this event will need to be transmitted by a wireless sensor. This wireless sensor will receive and transmit data to an appropriate device, likely a computer, which will use the data to update information and possibly publish it online.

2.3. eZ430-RF 2500 Development Kit, Simpliciti, and Temperature Sensor Demo Program

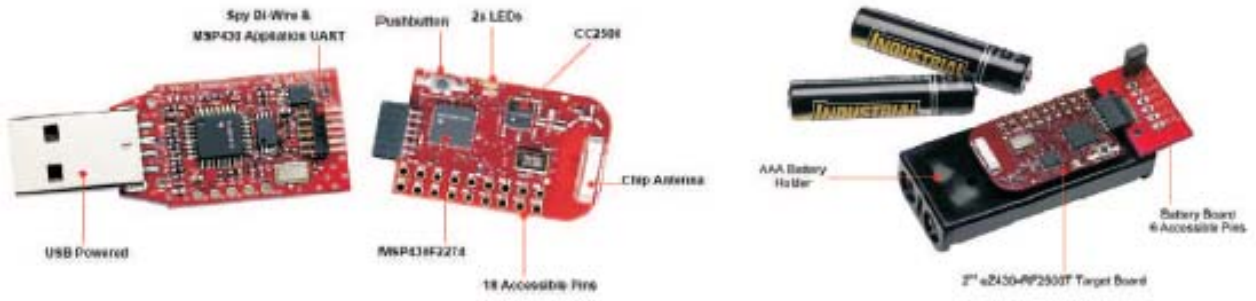

Figure 3. eZ430-RF2500 development tool kit (Left: access point, Right: end device)

eZ430-RF2500 is a USB-based wireless development kit unit developed by Texas Instruments. The kit includes two target boards. One target board is called an access point, the gateway that is connected to computer via USB connection and receives an incoming data from an end device. The device, another target board, is the board that collects information from a sensor and wirelessly transmits the information to the access point.

Each target board uses a MSP430F2274 microcontroller. The microcontroller belongs to the MSP 430 family, which is manufactured by Texas Instruments. MSP 430 has been designed specifically for wireless data transmission. Therefore, it uses ultra-low-power. Also, the target board uses a CC2250, a transceiver manufactured by Texas Instruments, and has a built-in temperature sensor. The transceiver operates at 2.4 GHz. A unique feature about eZ430-RF2500 is its USB debugging interface. This debugging interface allows users to debug each target board more conveniently. Each board has 18 accessible pins for development purposes. Although each pin can technically be altered for various development purposes, some of them have specific functions. For example, Pin 1 and Pin 2, which are used in the monitor system, are designed as ground and supply voltage. Pin 3, which is another Pin used in this project, is considered as a general I/O, or it can be used as an analog input just as it is used in this project. In this project, the output that will be resulted by LED lights will be connected to Pin 3. It means the output coming from the photodiode sensors will be connected to Pin 3 as an input.

eZ430-RF 2500 is capable of using various wireless network protocols. One of them is XBee, which has been a popular choice of wireless network protocols for many Swarthmore students in the past with their E90 projects. However, instead of using XBee, Simpliciti, a protocol developed by Texas Instruments, was used. Simpliciti is simple and is capable of various wireless designs. One can use the protocol to make end devices behave as if they were in an ad-hoc network.

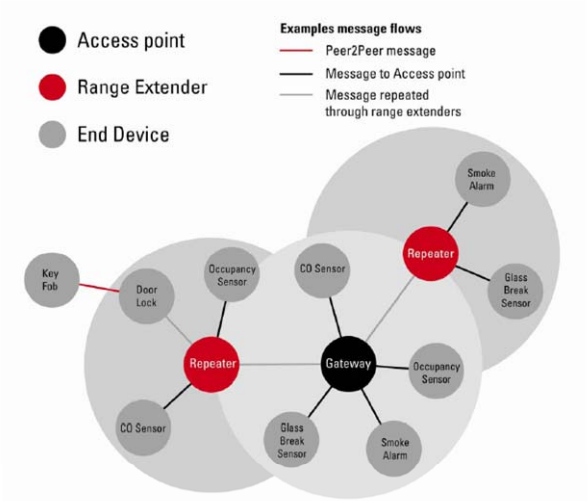

Figure 4. Simpliciti Network

In this project, data transmission is strictly allowed between end devices and access points as seen in Figure 4. This means an end device will only allow transmitting a datum to an access point, not to another end device nearby. Also, the reason why Simpliciti was chosen to be the network protocol was the convenience. eZ430-RF2500 kit includes a demonstration program called eZ430-RF2500 Temperature Sensor Program.

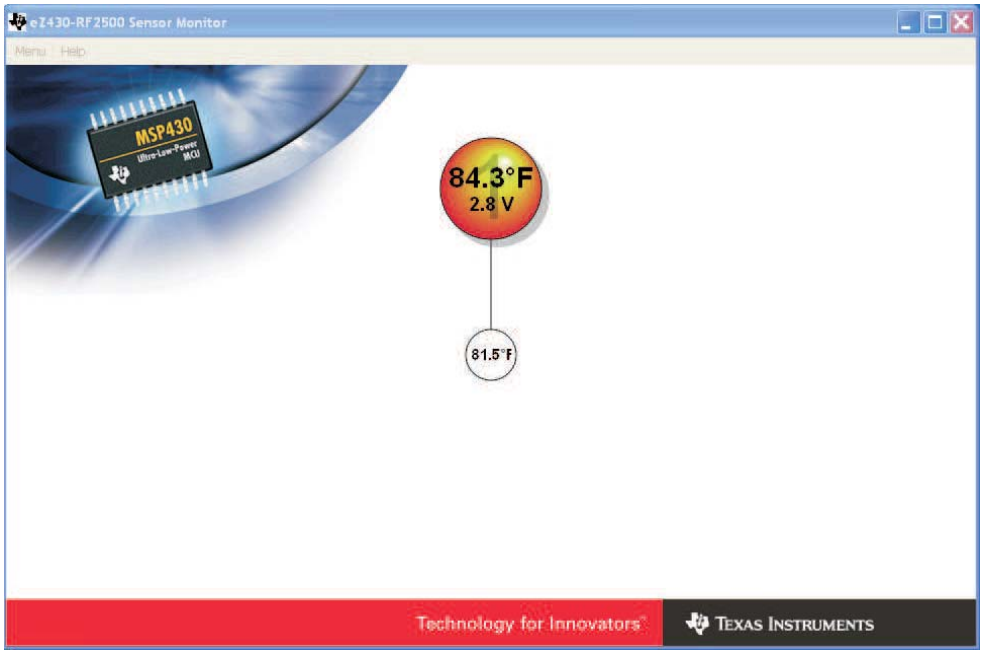

Figure 5. An image of eZ430-RF2500 temperature sensor program

As seen in Figure 5, the development kit includes the executable program that displays the temperature that is collected by the temperature sensor installed on the target board. With the program, the develop kit provides helpful sample programs of the temperature sensor network for both access point and end device in C, and the programs use Simpliciti. It was deemed too adventurous to explore the option of using XBee as a wireless network protocol; Simpliciti was chosen to be used as the network protocol. Time was limited in this project, and switching the network protocol from Simpliciti to XBee seemed unnecessary, given that Simpliciti was already confirmed to be functional with the demo temperature program.

2.4. OPT 101 – Monolithic Photodiode and Single-Supply Transimpedance Amplifier

Figure 6. OPT 101 (image from TME Electronics)

OPT 101, which is also known as the monolithic photodiodes and single-supply transimpedance amplifier, is used a photo sensor. It will detect whether an LED light is on or off in this project. A single-supply voltage between 2.7 and 36 V powers OPT 101. The maximum supply voltage with eZ430-RF2500 is 3 V, which makes OPT 101 a suitable photo sensor in this project.

As a photodiode sensor, OPT 101 is very sensitive to light. With room light provided by fluorescent light bulbs, OPT 101 outputs as high as 70% of its maximum

output. This leaves a very small space for the observation in changes of voltage values. Therefore, the output voltage with ambient light should be lowered. In this project, this is achieved by building a small cube made out of a black sheet of paper. The cube will cover the entire surface of OPT101 and shield ambient light.

OPT 101 has a built-in amplification. Therefore, no amplifier needs to be built in order to amplify the generally-considered weak "results" from diodes. In addition, the built-in amplifier is actually a transimpedance amplifier, which converts current to voltage. Therefore, the generated output will be voltage, meaning the voltage will be produced when a light is detected by OPT 101. The output voltage from OPT 101 then will be connected to Pin 3 of the end device for data transmission.

2.5. Sensor Circuit Design for Washer and Dryer

In this project, op-amps, operational amplifiers, were used to average output voltage that is produced by OPT 101 and to provide voltage amplification. Voltage amplification was used to monitor different washing cycles such as washing and spinning. TLV 2772, a Texas Instruments product, was used in this project. TLV 2772 is called the Dual 2.7-V High Slew Rate Rail-To-Rail Operational Amplifier. Like OPT101, it also uses a single-supply between 2.7 and 5.5 V. Its supply voltage is also a match with the eZ430-RF 2500 supply.

As briefly noted in the previous section, an end device of eZ430-RF2500 is supplied by 2 AAA batteries. This means its supply is a unipolar supply. So, the feature Rail-To-Rail that TLV 2772 has becomes crucial in this project. Rail-To-Rail op-amps like TLV 2772 can output voltages near both supply nails. In addition, TLV 2772 is extremely easy to work with. With the supply voltage being single-supply, non-inverting configuration was used in this project.

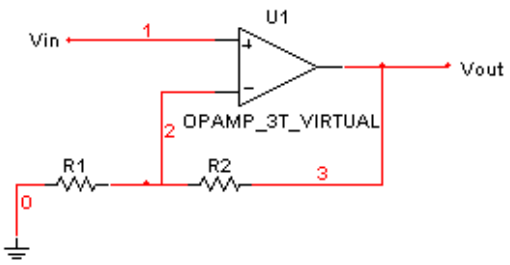

Figure 7. Non-inverting amplifier

A simple non-inverting configuration of op-amp is seen in Figure 7. In non-inverting configuration, gain is defined as $\frac{V_{out}}{V_{in}} = 1 + \frac{R_2}{R_1}$ (Equation 1). In order to average output voltage by using this configuration, a passive voltage averager needs to be built first. This can easily be done by connecting each voltage of interest with resistors in parallel. The output voltage from this passive network is the average of all the voltages, which are summed at the common node. Then, the average output can be fed into the positive terminal of op-amp as V_{in} as seen in Figure 8. Now, if R_1 has much more resistance than R_2, then the resistance ratio from Eq. 1 becomes approximately zero. Then, gain will approximately be equal to 1. This translates into the following: $\frac{V_{out}}{V_{in}} = 1$. Since V_{in} is $\frac{V_1 + V_2 + V_3}{3}$, V_{out} must be $\frac{V_1 + V_2 + V_3}{3}$ (Equation 2) as well. With this set-up, voltages that are collected by three OPT 101s installed on each LED lights can now be averaged out. This means whenever one of the LED lights goes on, then the average value of output voltage will also increase as well.

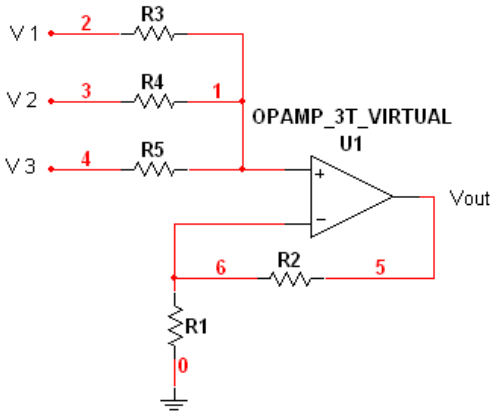

Figure 8. Non-inverting amplifier as a voltage averager

For the dryer, only one LED light will be activated once the machine starts operating. A circuit design is shown in Figure 9.

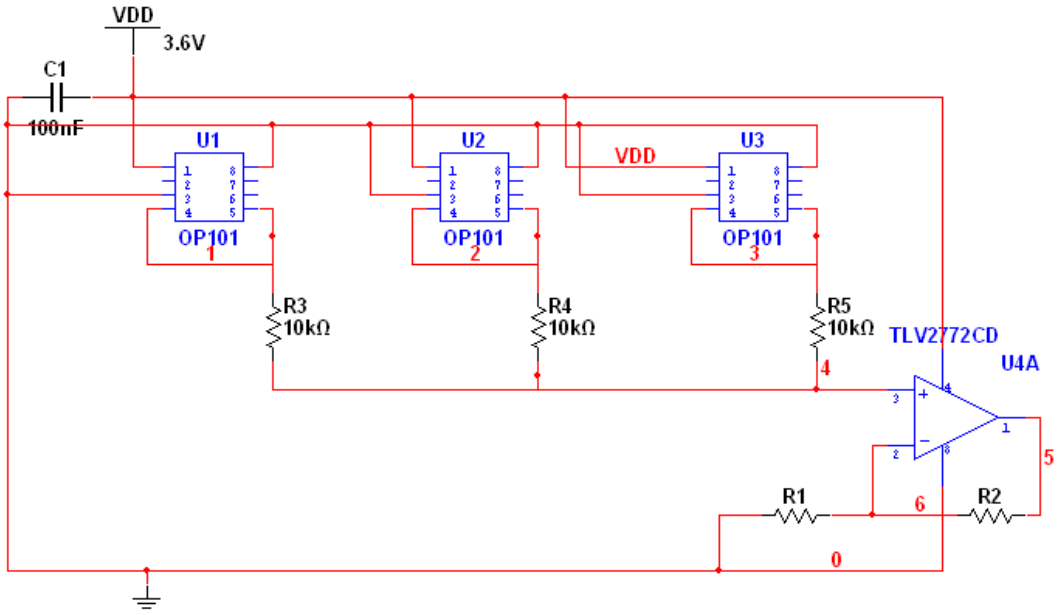

Figure 9. A non-inverting configuration of circuit design for the dryer, where R1 is much greater than R2

For the washer, the circuit design requires a little more work. Out of three LED lights, two LED lights can be on at the same time. The "add softener" light may come on with the Washing cycle LED light. However, this "add softener" light is ignored in this experiment for the following reason. It is likely that not many students will take advantage of this feature. On the other hand, indicating different washing cycles seems to be a nice feature for students because it indicates whether the washing operation is soon to be terminated. For example, "spinning cycle" generally indicates the last phase of washing cycle. This information will allow students to prepare for their trip to a laundry room. However, there is a limitation in this project, which will be discussed in the next section.

The limitation is only one output voltage can be transmitted. How exactly does this affect the laundry system? Here is an example. Let's say OPT 101 outputs

50 mV with no LED on and outputs 100 mV with the LED light on. Here is what happens.

Table 1. A table of how the average output changes with one LED light on (the LED 2 "add softener" light is dependent on LED 1. This table only looks at the case where only one LED light is on. Since a case when only LED 2 is on never exists, the average output voltage for this case does not exist)

	All LED lights off	LED 1 (washing cycle) on	LED 2 (add softener*) on	LED 3 (spinning cycle) on
Average output voltage	50mV	66.6mV		66.6mV

With only one output voltage data set transmittable, this configuration will not differentiate the cycles between washing and spinning. An expected outcome from having either LED 1 or 2 on is the same. In order to produce a different outcome value, an amplifier is built for one of the LED lights. In this case, LED 3, which is installed for spinning cycle, is chosen for the amplifier. The gain of this amplifier was chosen as 7.8, which was large enough to differentiate the average output voltage when LED 1 is on. Here is the result.

Table 2. A table of how the average output changes with one LED light on with a non-inverting amplifier for LED 3 with a gain of 7.8 (the LED 2 "add softener" light is dependent on LED 1. This table only looks at the case where only one LED light is on. Since a case when only LED 2 is on never exists, the average output voltage for this case does not exist)

	All LED lights off	LED 1 (washing cycle) on	LED 2 (add softener*) on	LED 3 (spinning cycle) on
Average output voltage	50mV	66.6mV		293.3mV

As seen in Table 2, the expected average output voltage value when LED 3 is on is higher than the expected average voltage value when LED 1 is on. A circuit scheme is seen in Figure 10.

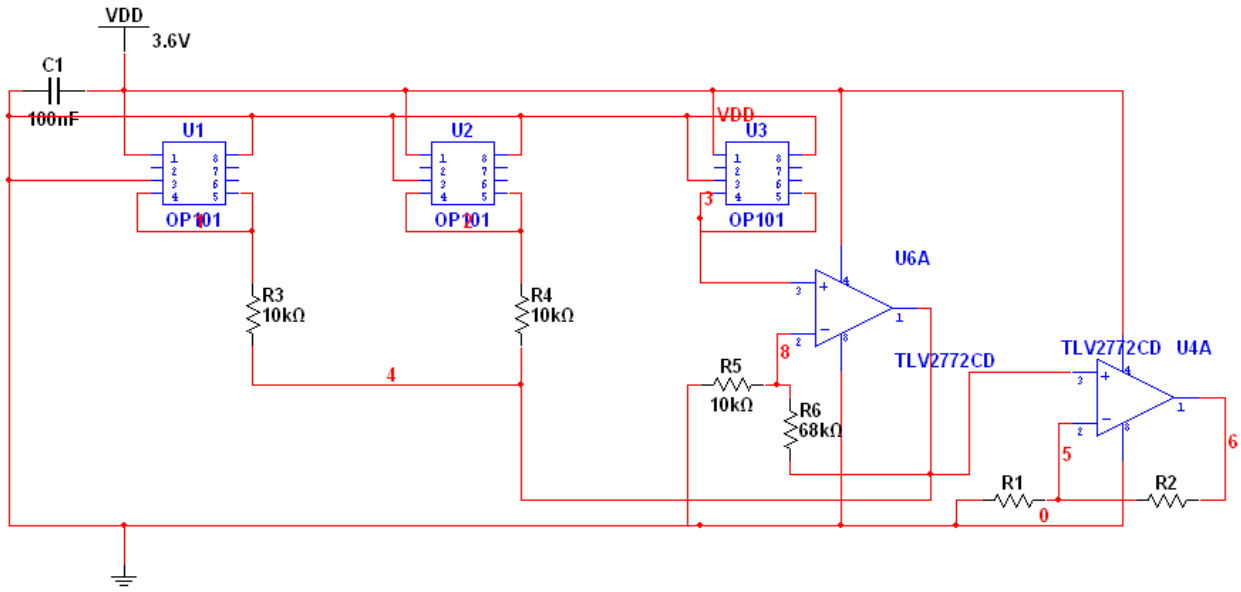

Figure 10. A non-inverting configuration of the circuit design for the washer, where R1 is much greater than R2

2.6. Project Difficulty and Limitation

Although C programming codes for the access point and end device were used, there were slight program modifications that were required for this project. For programming modification, IAR Embedded Workbench, which was a software included in this development kit, was used. IAR Embedded Workbench was also used to compile and build a program on each target board. Some of the modifications were made with the demo programs. First, the demo programs output temperatures in two different modes: verbose and minimal modes. The output data from verbose mode can be seen in Figure 11 and 12.

```
Node:HUB0,Temp: 91.2F,Battery:3.5V,Strength:000%,RE:no
```

Figure 11. A typical data transmission in verbose mode displaying node, temperature in Fahrenheit, battery voltage, strength, and RE indicator (to show whether or not a received packet has gone through a Range Extender, which can be ignored in this project).

Minimal mode is simple, but a useful/powerful way to place some of the data acquired by eZ430-RF2500. By using software like Java, C, or even MATLAB, one

can put the information into arrays, and conveniently choose an array of interest. In this project, the modification was made with the battery information.

```
$HUB0, 89.6F,3.5,000,N#
```

Figure 12. A typical data transmission in minimal mode displaying node, temperature in Fahrenheit, battery voltage, strength, and RE indicator.

In this project, verbose mode is completely discarded. The programming codes involved with the mode are erased. Here is the reason: MATLAB will be used to collect information. When importing data, the data that include non-integers will be difficult to compare and analyze numerically in MATLAB. Therefore, only minimal mode is allowed so that integer values can be collected into arrays. As seen in Figures 11 and 12, a comma will be used as a delimiter.

Besides the difficulty with dealing with non-integer values in MATLAB, another problem arises when one uses the temperature demo program. As seen in Figures 11 and 12, there are a limited number of data that can be transmitted by using the demo program. The number of transferrable data fields did not change for the following reason. The structures of the temperature sensor program were delicate. Therefore, it was feared that the integrity of the program codes would be broken when the data algorithms and structures were altered. This meant the only way to transmit the average output voltage from washers and dryers was to replace the battery voltage information with the average output voltage. There was another limitation that was associated with the limited availability of transmittable data. How does the program correctly identify if an incoming message was sent out by the washer or the dryer?

The node, which can be seen in Figures 11 and 12 only as HUB0, is assigned by Simpliciti based on the order of transmission. One may think an end device "$001" installed on a washer will always carry the node "$001" so that "$001" will always be associated with that washer. However, this isn't true. If the data transmission of "$001" was followed by the transmission of "$002," then the node of "$001" will actually be "$002." Therefore, an identifier was needed to indicate

whether the end device was installed on the dryer or the washer. There was only one data field remaining to be modified: temperature. The default program read the value from the temperature sensor installed on the eZ430-RF2500 target board. A slight modification in the programming was made so that the value could be used to identify between the washer and the dryer. For example, a value of 0 was applied to the temperature data field for the washer, and it produced a value of 32 while a value of 30 was applied to the temperature data field for the dryer, and it resulted in 37.

2.7. Programming Design

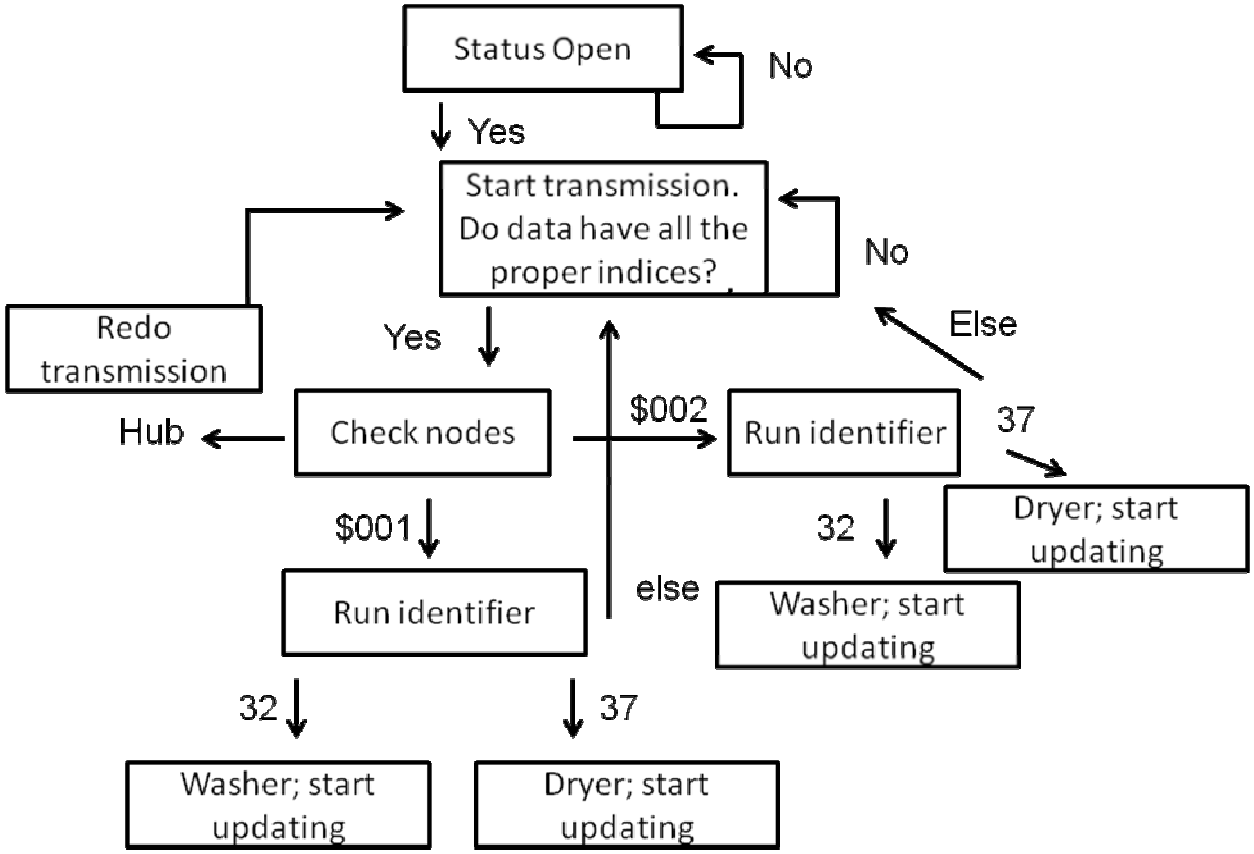

Figure 13. A program flowchart in MATLAB

The programming design is seen in Figure 13, which shows the flowchart of how the programming in MATLAB works. First, MATLAB finds if the used COM port is open. Since the access point is connected via USB, the communication that is made between the access point and end device can be collected by reading off the appropriate COM port the Simpliciti protocol uses. If the COM port is not open or available, then the program tries again to make sure the port is not available. If the port is open or becomes available (open), then the next flow can proceed. Now, the data transmission begins. As it turned out, reading COM port data from MATLAB was a little harder than expected. For some reason, MATLAB would sometimes omit or add extra erroneous information, which changes the sizes of arrays. Therefore, the program checks if the collected set of arrays has a proper number of indices. If it does, then the next procedure can proceed. There will be three ways to identify the washer or the dryer. If an identifier indicates that the incoming data was sent by node "#01" and had a value of "32" in the temperature data field, then it is the washer. If the node had a value of "37" in the temperature data field, then it is the dryer. Sometimes, MATLAB will have an erroneous value that is neither 32 nor 37. If this error ever happens, then the flow needs to be re-directed to the beginning of the flowchart. Once the flow reaches its end, then the chart repeats.

3. Apparatus Set-Up

The exploration of this project was done in a laboratory, not in a real laundry room. The basic problem was all the circuit components were installed on a breadboard. This made the circuits physically inaccessible. In order to simulate the LED light panel settings on the dryer and washer, an alternate LED light panel was designed to simulate a real laundry room setting. Pictures of the apparatus set-up can be seen in Figures 14, 15, and 16. In Figure 14, a small black paper cube that mentioned in an earlier section of the report can be seen. The cube houses both OPT 101 and LED light. In Figure 15, two switches for the washer and dryer can be seen along with eZ430-RF2500. Figure 16 provides a clear picture of the set-up as a whole.

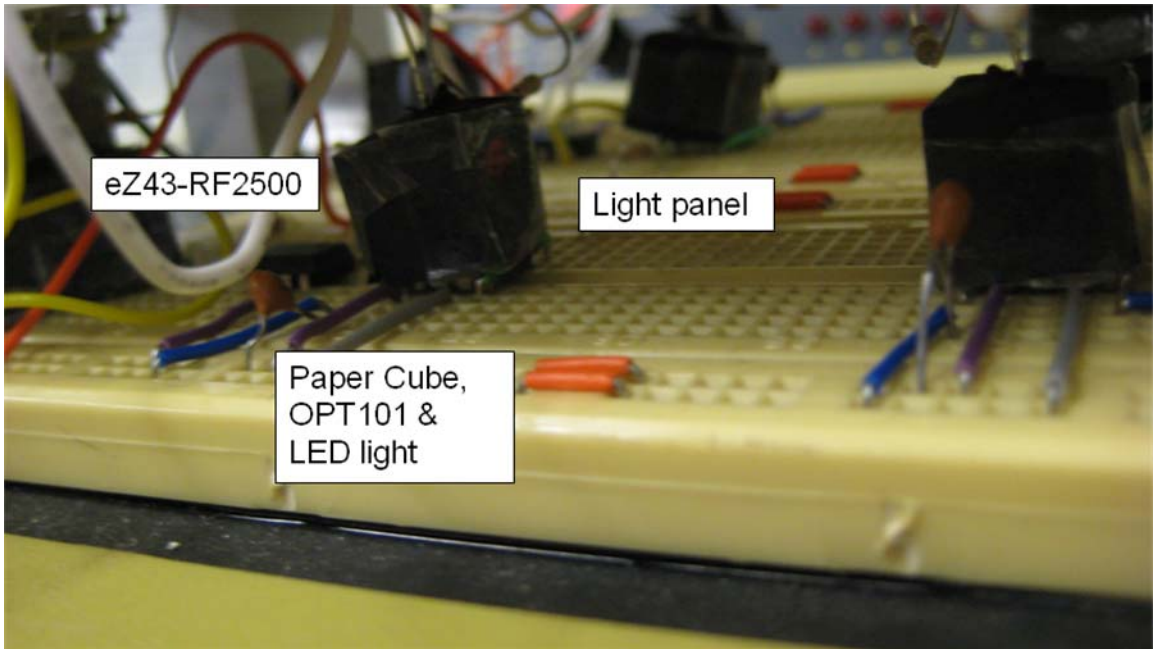

Figure 14. Set-up from close-up

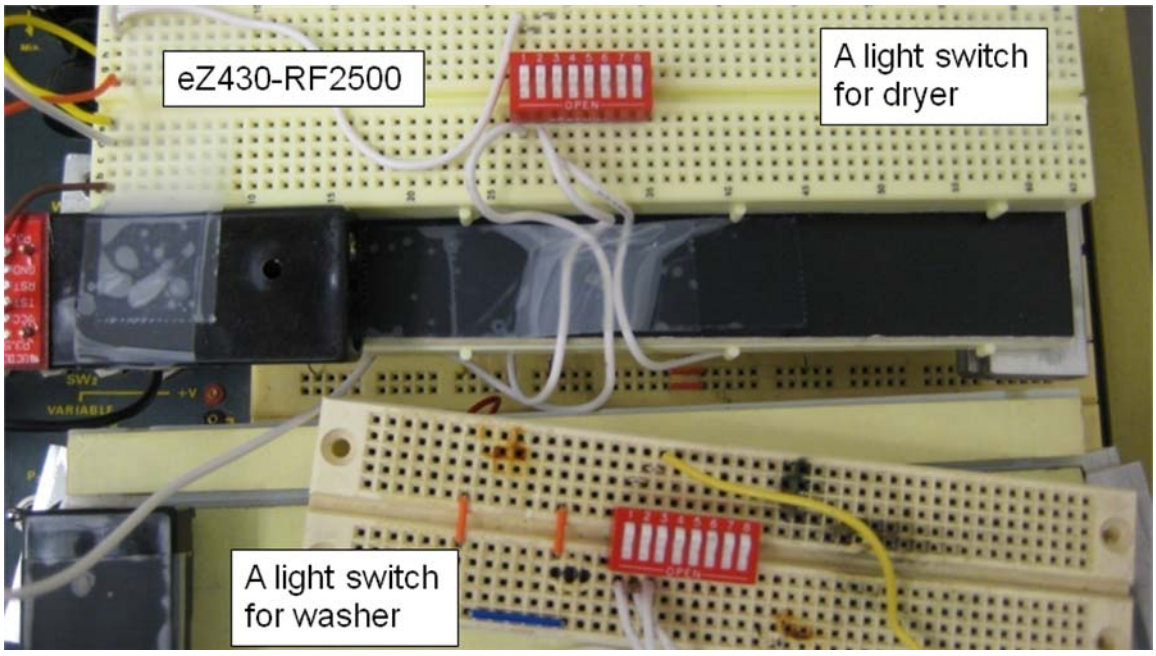

Figure 15. Set up from birdview

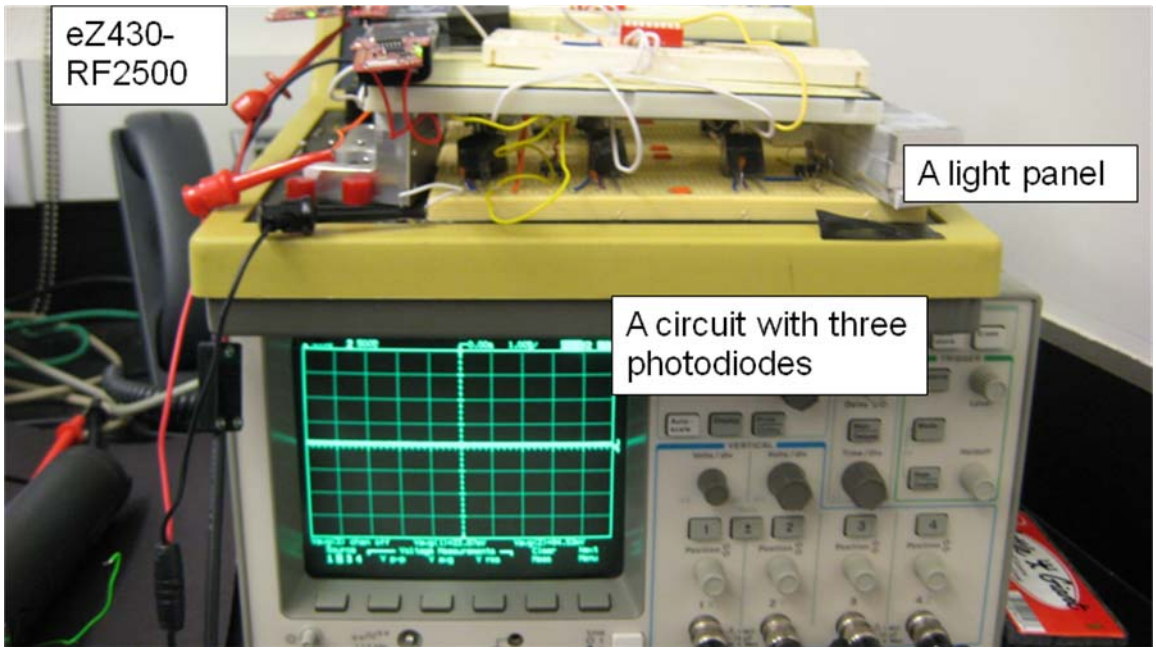

Figure 16. Set-up from sideview

4. Results

4.1. Numerical Results

The following set of values was recorded when LED light was on/off simulating real washers and dryers.

Table 3. A table of results collected by the sensors installed on washer and dryer

Average output voltage\Light	All LED off	LED 1 on; LED 2 and 3 off	LED 2 on; LED 1 and 3 off	LED 3 on; LED 1 and 2 off
Dryer	29.4 mV	73.5 mV	79.1 mV	60.4 mV
Washer	51.1 mV	83.3 mV		142 mV

For the dryer, the average output voltage behaved as expected: the voltage value was a lot higher with one of the LED lights turning on. However, the average output voltages slightly varied from case to case. This is the case because the sensitivities of OPT 101 vary. Also, the sizes of punctured holes on top of each paper cube were different. Even a slightest amount of ambient light that travels through the small

holes can make a subtle difference. However, the difference caused by this problem did not have a significant effect on the overall results.

For washer, the results were as expected. With LED 3 on, the average output voltage was higher than the one produced with LED 1 on. However, the expected gain from LED 3, which was 7.8, was not observed. The actual gain was somewhat lower. However, this problem did not also have a significant effect on the overall results.

4.2. Results via HyperTerminal

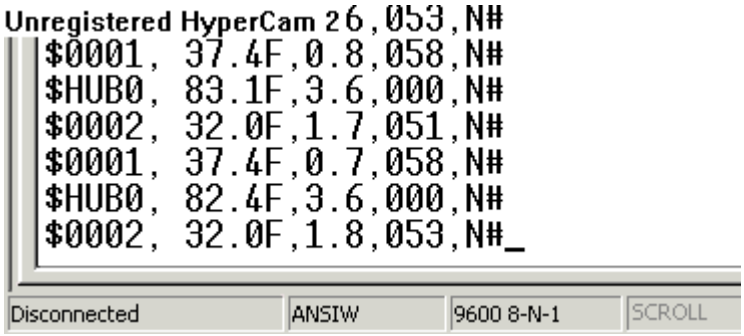

Figure 17. An image of the data transmission that can be seen via HyperTerminal

A record of data transmission can be seen via using HyperTerminal. Instead of using MATLAB to read the COM port, HyperTerminal was used in this lab to monitor the average output voltages from OPT 101 and temperature identifier. Figure 17 shows the set of data collected from end devices "$001" and "$002" and an access point "$HUB0." In this case, "$001" is installed on the dryer since its temperature shows a value of 37. "$002" is installed on the washer, given that the temperature value is 32.

4.3. Results via MATLAB

As seen in the Figure, the MATLAB program displays the message as well as the set of data values.

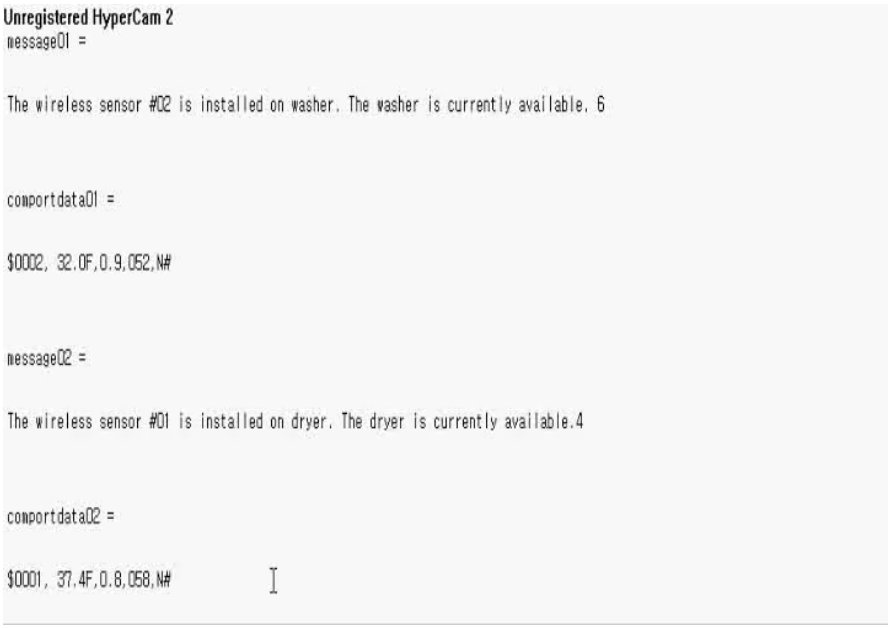

Figure 18. An image of how the program displays in MATLAB

Each end device will have its individual message identifying which device it is installed on and indicating the current status of the machine. Inside the MATLAB programming code, the messages will individually be saved to text files (*.txt) so that they can be viewed on web browser.

4.4. Results via Webpage

In order to display the results online, the MATLAB code needs to run on a server. This means the current directory of MATLAB needs to be changed to a server, which requires permission to access the server. Then, a simple webpage can be written such as the one seen in Figure 19. The webpage divides into two frames: top and bottom. The top frame displays basic information with instructions. The bottom frame displays the status.

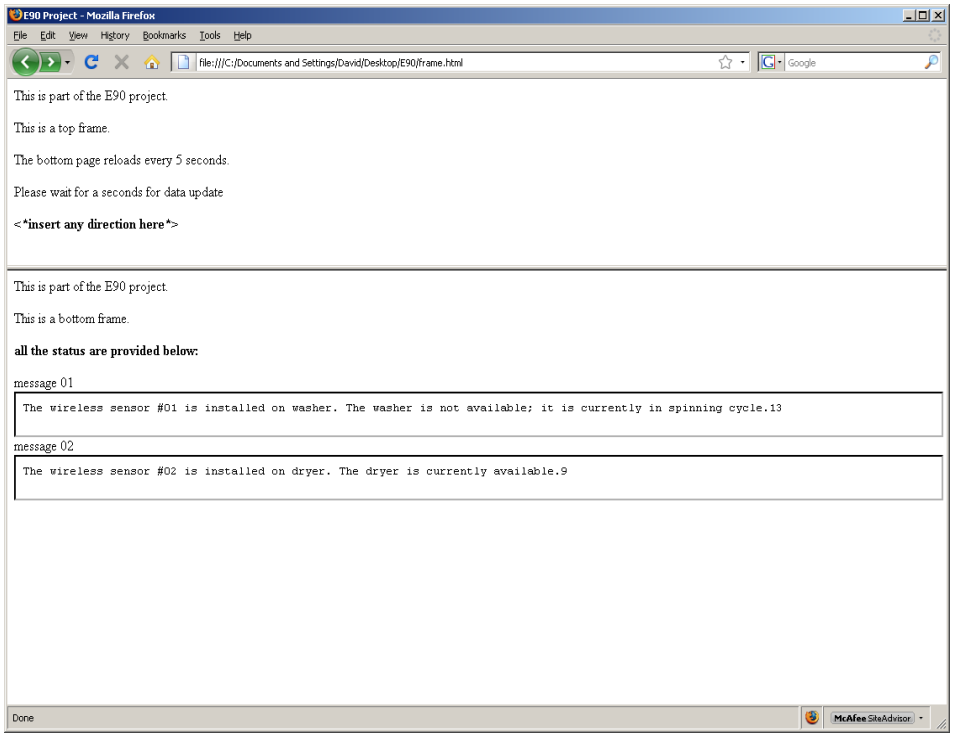

Figure 19. An image of how the webpage displays the status of washer and dryer

5. Conclusion / Suggested Future Work

All the knowledge learned from taking electric circuit courses and computer science were integrated into this E90 project. From creating a circuit design to making proper decisions to execute, this senior project as a whole showed how I grew and matured at Swarthmore while studying engineering. In the end, the monitoring system for a laundry room was successfully developed with basic, and yet functional, features.

For future work, an actual implementation is sought. To do so, designing a case that will house the circuits with eZ430-RF2500 is desirable. Also, producing an executable program is desired as well. Since the current program depends on MATLAB, it will not be able to run on the machines without MATLAB. In addition, the current website can be improved by becoming more user-friendly and user-interactive.

6. Acknowledgements

I would personally like to thank Professor Molter for giving me an opportunity to work with her. This project was an exciting journey for me, and I hope Professor Molter feels the same. I would like to thank Professor Molter for her patience and trust in me with this project. I would also like to thank Professor Cheever for introducing me to the eZ430-RF2500 development tool. This wireless development tool was an integral part of my project, and without Professor Cheever's help, this laundry monitoring system would have never been developed. Also, I would like to thank Professor Carr Everbach for providing me the idea of how to initialize my MATLAB program. Lastly, I would like to thank Mr. Ed Jaoudi for ordering all the circuitry parts that were used in this project.

7. Appendices

Appendix A. Access Point Code in C

```
//*****************************************************************************
//   eZ430-RF2500 Temperature Sensor Access Point
//
//   Description: This is the Access Point software for the eZ430-2500RF
//                Temperature Sensing demo
//
//
//   L. Westlund
//   Version    1.02
//   Texas Instruments, Inc
//   November 2007
//   Built with IAR Embedded Workbench Version: 4.09A
//*****************************************************************************
//Change Log:
//*****************************************************************************
//Version:  1.02
//Comments: Changed Port toggling to abstract method
//          Removed ToggleLED
//          Fixed comment typos/errors
//          Changed startup string to 1.02
//Version:  1.01
//Comments: Added support for SimpliciTI 1.0.3
//          Changed RSSI read method
//          Added 3 digit temperature output for 100+F
//          Changed startup string to 1.01
//Version:  1.00
//Comments: Initial Release Version
//*****************************************************************************

//* Modified by David Kwon for E90 Project
//* All the modifications are noted in the program


#include "bsp.h"
#include "mrfi.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 "msp430x22x4.h"
#include "vlo_rand.h"

#define MESSAGE_LENGTH 3
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
const char splash[] = {"\r\n----------\r\n     ****\r\n     ****          eZ430-
RF2500\r\n          ******o****        Temperature Sensor Network\r\n********_///_****      Copyright
2007\r\n  ******/_//_/*****    Texas Instruments Incorporated\r\n    **  ***(__/*****      All rights
reserved.\r\n      *********    Version 1.02\r\n       *****\r\n        ***\r\n----------
----------\r\n"};

__no_init volatile int tempOffset @ 0x10F4; // Temperature offset set at production
__no_init volatile char Flash_Addr[4] @ 0x10F0; // Flash address set randomly

// 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)
{
  addr_t lAddr;
  bspIState_t intState;

  WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
  {
  // delay loop to ensure proper startup before SimpliciTI increases DCO
  // This is typically tailored to the power supply used, and in this case
  // is overkill for safety due to wide distribution.
    volatile int i;
    for(i = 0; i < 0xFFFF; i++){}
  }
  if( CALBC1_8MHZ == 0xFF )                 // Do not run if cal values are erased
  {
    volatile int i;
    P1DIR |= 0x03;
    BSP_TURN_ON_LED1();
    BSP_TURN_OFF_LED2();
    while(1)
    {
      for(i = 0; i < 0x5FFF; i++){}
      BSP_TOGGLE_LED2();
      BSP_TOGGLE_LED1();
    }
  }

  BSP_Init();

  if( Flash_Addr[0] == 0xFF &&
      Flash_Addr[1] == 0xFF &&
      Flash_Addr[2] == 0xFF &&
      Flash_Addr[3] == 0xFF )
```

```
{
  createRandomAddress();                // set Random device address at initial startup
}
lAddr.addr[0]=Flash_Addr[0];
lAddr.addr[1]=Flash_Addr[1];
lAddr.addr[2]=Flash_Addr[2];
lAddr.addr[3]=Flash_Addr[3];
SMPL_Ioctl(IOCTL_OBJ_ADDR, IOCTL_ACT_SET, &lAddr);

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/5
    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 );
    BSP_TOGGLE_LED1();
    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.lid = sLID[i];
        SMPL_Ioctl(IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_SIGINFO, (void *)&sigInfo);
        transmitData( i, (signed char)sigInfo.sigInfo[0], (char*)msg );
        BSP_TOGGLE_LED2();
        BSP_ENTER_CRITICAL_SECTION(intState);
        sPeerFrameSem--;
        BSP_EXIT_CRITICAL_SECTION(intState);
      }
    }
  }
 }
}

/*------------------------------------------------------------------------------
*
------------------------------------------------------------------------------*/
void createRandomAddress()
{
  unsigned int rand, rand2;
```

```
  do
  {
    rand = TI_getRandomIntegerFromVLO();    // first byte can not be 0x00 of 0xFF
  }
  while( (rand & 0xFF00)==0xFF00 || (rand & 0xFF00)==0x0000 );
  rand2 = TI_getRandomIntegerFromVLO();

  BCSCTL1 = CALBC1_1MHZ;                    // Set DCO to 1MHz
  DCOCTL = CALDCO_1MHZ;
  FCTL2 = FWKEY + FSSEL0 + FN1;             // MCLK/3 for Flash Timing Generator
  FCTL3 = FWKEY + LOCKA;                    // Clear LOCK & LOCKA bits
  FCTL1 = FWKEY + WRT;                      // Set WRT bit for write operation

  Flash_Addr[0]=(rand>>8) & 0xFF;
  Flash_Addr[1]=rand & 0xFF;
  Flash_Addr[2]=(rand2>>8) & 0xFF;
  Flash_Addr[3]=rand2 & 0xFF;

  FCTL1 = FWKEY;                            // Clear WRT bit
  FCTL3 = FWKEY + LOCKA + LOCK;             // Set LOCK & LOCKA bit
}

/*-------------------------------------------------------------------------
*
-------------------------------------------------------------------------*/
void transmitData(int addr, signed char rssi,  char msg[MESSAGE_LENGTH] )
{
  char addrString[4];
  char rssiString[3];
  volatile signed int rssi_int;

  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);

  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;
  }
  else if( ((temp/1000)%10) != 0 )
  {
    temp_string[0] = '0'+((temp/1000)%10);
  }
  temp_string[4] = '0'+(temp%10);
  temp_string[2] = '0'+((temp/10)%10);
  temp_string[1] = '0'+((temp/100)%10);

  // David's change: Verbose Mode disabled. Reason: When reading output from MATLAB,
  // The textstring needs to output "numbers" only.
  // When in verbose mode, it displays numbers with characters, which is not good for MATLAB.
  //
  // Therefore, verbose mode disabled for my purpose.

    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();
}
/*---------------------------------------------------------------------------
* 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 lid)
{
  if (lid)
  {
    sPeerFrameSem++;
  }
  else
  {
    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 = 0;
  }
}
```

Appendix B. End Device Code in C

```
//*****************************************************************************
//   eZ430-RF2500 Temperature Sensor End Device
//
//   Description: This is the Access Point software for the eZ430-2500RF
//                Temperature Sensing demo
//
//
//   L. Westlund
//   Version    1.02
//   Texas Instruments, Inc
//   November 2007
//   Built with IAR Embedded Workbench Version: 4.09A
//*****************************************************************************
//Change Log:
//*****************************************************************************
//Version:  1.02
//Comments: Changed Port toggling to abstract method
//          Removed ToggleLED
//          Fixed comment typos/errors
//          Changed startup string to 1.02
//Version:  1.01
//Comments: Added support for SimpliciTI 1.0.3
//          Changed RSSI read method
//          Added 3 digit temperature output for 100+F
//          Changed startup string to 1.01
//Version:  1.00
//Comments: Initial Release Version
//*****************************************************************************

// Some of the contents were taken from TI's Temperature Demo Program
// for eZ430-RF2500.
// Edited by David Kwon for E90 Project


#include "bsp.h"
#include "mrfi.h"
#include "nwk_types.h"
#include "nwk_api.h"
#include "bsp_leds.h"
#include "bsp_buttons.h"
#include "vlo_rand.h"

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

__no_init volatile int tempOffset @ 0x10F4; // Temperature offset set at production
__no_init volatile char Flash_Addr[4] @ 0x10F0; // Flash address set randomly

void createRandomAddress();

void main (void)
{
  addr_t lAddr;
  WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
  {
  // delay loop to ensure proper startup before SimpliciTI increases DCO
  // This is typically tailored to the power supply used, and in this case
```

```
// is overkill for safety due to wide distribution.
  volatile int i;
  for(i = 0; i < 0xFFFF; i++){}
}
if( CALBC1_8MHZ == 0xFF )                 // Do not run if cal values are erased
{
  volatile int i;
  P1DIR |= 0x03;
  BSP_TURN_ON_LED1();
  BSP_TURN_OFF_LED2();
  while(1)
  {
    for(i = 0; i < 0x5FFF; i++){}
    BSP_TOGGLE_LED2();
    BSP_TOGGLE_LED1();
  }
}

// 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();

if( Flash_Addr[0] == 0xFF &&
    Flash_Addr[1] == 0xFF &&
    Flash_Addr[2] == 0xFF &&
    Flash_Addr[3] == 0xFF )
{
  createRandomAddress();                  // set Random device address at initial startup
}
lAddr.addr[0]=Flash_Addr[0];
lAddr.addr[1]=Flash_Addr[1];
lAddr.addr[2]=Flash_Addr[2];
lAddr.addr[3]=Flash_Addr[3];
SMPL_Ioctl(IOCTL_OBJ_ADDR, IOCTL_ACT_SET, &lAddr);
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))
{
  BSP_TOGGLE_LED1();
  BSP_TOGGLE_LED2();;
  __bis_SR_register(LPM3_bits + GIE);     // LPM3 with interrupts enabled
}
```

```
  // unconditional link to AP which is listening due to successful join.
  linkTo();
}

void createRandomAddress()
{
  unsigned int rand, rand2;
  do
  {
    rand = TI_getRandomIntegerFromVLO();    // first byte can not be 0x00 of 0xFF
  }
  while( (rand & 0xFF00)==0xFF00 || (rand & 0xFF00)==0x0000 );
  rand2 = TI_getRandomIntegerFromVLO();

  BCSCTL1 = CALBC1_1MHZ;                    // Set DCO to 1MHz
  DCOCTL = CALDCO_1MHZ;
  FCTL2 = FWKEY + FSSEL0 + FN1;             // MCLK/3 for Flash Timing Generator
  FCTL3 = FWKEY + LOCKA;                    // Clear LOCK & LOCKA bits
  FCTL1 = FWKEY + WRT;                      // Set WRT bit for write operation

  Flash_Addr[0]=(rand>>8) & 0xFF;
  Flash_Addr[1]=rand & 0xFF;
  Flash_Addr[2]=(rand2>>8) & 0xFF;
  Flash_Addr[3]=rand2 & 0xFF;

  FCTL1 = FWKEY;                            // Clear WRT bit
  FCTL3 = FWKEY + LOCKA + LOCK;             // Set LOCK & LOCKA bit
}
void linkTo()
{
  linkID_t linkID1;
  uint8_t  msg[3];

  // keep trying to link...
  while (SMPL_SUCCESS != SMPL_Link(&linkID1))
  {
    __bis_SR_register(LPM3_bits + GIE);     // LPM3 with interrupts enabled
    BSP_TOGGLE_LED1();
    BSP_TOGGLE_LED2();
  }

  // Turn off all LEDs
  if (BSP_LED1_IS_ON())
  {
    BSP_TOGGLE_LED1();
  }
  if (BSP_LED2_IS_ON())
  {
    BSP_TOGGLE_LED2();
  }
  while (1)
  {
    // This part of the program is greatly edited
    // in order to display the output voltage

    //volatile long temp;
    int degC;
    volatile long sample;
    int voltage;
```

```
  //signed int output; //output of the voltage on PIN 3
  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, "" );

  P2DIR |= 0x01;   // OPEN PORT 2 and PIN 3 (PIN 3 = P2.1)
  ADC10AE0 = 0x01; // Enabling Port for the ADC10
  ADC10CTL0 = ADC10SHT_2 + ADC10ON + ADC10IE + ADC10SR; //Intiaiting A to D conversion
  ADC10CTL1 = INCH_0;   //Enabling Channel A0
  for( degC = 240; degC > 0; degC-- ); // delay to allow reference to settle
  ADC10CTL0 |= ENC + ADC10SC; //sampling and conversion initiate
  __bis_SR_register(CPUOFF + GIE); //LPM0 with interrupts enable
  sample = ADC10MEM; // store binary results from ADC10MEM
  ADC10CTL0 &= ~ENC;  //
  ADC10CTL0 &= ~(REFON + ADC10ON);        // turn off A/D to save power

  voltage = (sample*25)/51;  //measuring average output voltage
  // the denominator value can be changed to amplify the voltage value
  msg[0]=30; //this also can be assigned to 0 if the machine is washer.
  //Dryer is assigned with the value of 30.
  msg[1]=0; //does not use in this program (from temperatuer demo program)
  msg[2]=voltage; //store average output voltage


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



  if (SMPL_SUCCESS == SMPL_Send(linkID1, msg, sizeof(msg)))
  {
    BSP_TOGGLE_LED2();
  }
  else
  {
    BSP_TOGGLE_LED2();
    BSP_TOGGLE_LED1();
  }
 }
}

/*-------------------------------------------------------------------------
* 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)
}
```

Appendix C. MATLAB Programming

```
s = serial('COM5','BAUD',9600);          % Create serial object (PORT Dependent)

pause(0.0);

fopen(s);
%fprintf(s, 'Device Name:Temperature:Voltage:Byte:Other');

status = get(s, 'Status'); %obtain the status information of the serial port

pause(0.0);
counter = 0;
data = fscanf(s);

pause(0.0);

message01 = '1'; %create an initial message
message02 = '2'; %create an initial message


datarcvd01 = 0; %if data is received, the the value changes to 1
datarcvd02 = 0; %if data is received, the the value changes to 1

while(status == 'open') % check if the port is open (available)
   data = fscanf(s); % scan the data

   %properly name the data fields
   [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
   device = char(device);
   type = char(type);
   voltage = voltage;

   %check if all the indices have the required sizes
   %if not, then the repeat
   if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
      error = 1;
      while(error == 1)
         data = fscanf(s);
         [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
         device = char(device);
         type = char(type);
         voltage = voltage;
         if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
            error = 1;
         else
            error = 0;
         end
      end

   else
      ;
   end
   pause(0.0);

   while(device == '$HUB0') %  if your device is access point (HUB0), the re-read it
      data = fscanf(s);  %gets out of the loop
```

```
    [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
    device = char(device);
    type = char(type);
    voltage = voltage;
    if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
       error = 1;
       while(error == 1)
          data = fscanf(s);
          [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
          device = char(device);
          type = char(type);
          voltage = voltage;
          if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
             error = 1;
          else
             error = 0;
          end
       end

    else
       ;
    end
    pause(0.0);
 end;

 %once the previous while loops, we know that the transmitted data comes
 %from end; device

 while((datarcvd01 == 0) | | (datarcvd02 == 0))

    if(device == '$0001')
       voltage01 = voltage;
       type = type(1:2);

       %display the information
       if(type == '32')
          if(voltage01 < 1.3)
             message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is
currently available.'];
             datarcvd01 = 1;
          elseif(voltage01 > 1.3 & voltage01 < 2.2)
             message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is not
available; it is currently in washing cycle.'];
             datarcvd01 = 1;
          else
             message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is not
available; it is currently in spinning cycle.'];
             datarcvd01 = 1;
          end;
       else
          if(voltage01 < 1.0)
             message02 = ['The wireless sensor #' device(4:5) ' is installed on dryer. The dryer is
currently available.'];
             datarcvd01 = 1;
          else
             message02 = ['The wireless sensor #' device(4:5) ' is installed on dryer. The dryer is
currently not available; it is currently in operation.'];
             datarcvd01 = 1;
          end;
```

```
        end;
        data = fscanf(s);  %gets out of the loop
        [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
        device = char(device);
        type = char(type);
        voltage = voltage;
        if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
           error = 1;
           while(error == 1)
              data = fscanf(s);
              [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
              device = char(device);
              type = char(type);
              voltage = voltage;
              if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
                 error = 1;
              else
                 error = 0;
              end
           end
        else
           ;
        end
        pause(0.0);
        conter = 1;

     elseif(device == '$0002')

        voltage02 = voltage;
        type = type(1:2);
        if(type == '32')
           if(voltage02 < 1.0)
              message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is
currently available. '];
              datarcvd02 = 1;
           elseif(voltage02 > 1.3 & voltage02 < 2.2)
              message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is not
available; it is currently in washing cycle.'];
              datarcvd02 = 1;
           else
              message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is not
available; it is currently in spinning cycle.'];
              datarcvd02 = 1;
           end;
        else
           if(voltage02 < 1.0)
              message02 = ['The wireless sensor #' device(4:5) ' is installed on dryer. The dryer is
currently available.'];
              datarcvd02 = 1;
           else
              message02 = ['The wireless sensor #' device(4:5) ' is installed on dryer. The dryer is
currently not available; it is currently in operation.'];
              datarcvd02 = 1;
           end;
        end;
        data = fscanf(s);  %gets out of the loop
        [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
        device = char(device);
        type = char(type);
```

```
    voltage = voltage;
    if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
        error = 1;
        while(error == 1)
            data = fscanf(s);
            [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
            device = char(device);
            type = char(type);
            voltage = voltage;
            if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
                error = 1;
            else
                error = 0;
            end
        end

    else
        ;
    end
    pause(0.0);
    counter = 1;



else
    data = fscanf(s);
    [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
    device = char(device);
    type = char(type);
    voltage = voltage;
    if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
        error = 1;
        while(error == 1)
            data = fscanf(s);
            [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
            device = char(device);
            type = char(type);
            voltage = voltage;
            if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
                error = 1;
            else
                error = 0;
            end
        end

    else
        ;
    end
    pause(0.0);

    while(device == '$HUB0') %  if your device is access point (HUB0), the re-read it
        data = fscanf(s);  %gets out of the loop
        [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
        device = char(device);
        type = char(type);
        voltage = voltage;
        if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
            error = 1;
            while(error == 1)
```

```
                data = fscanf(s);
                [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
                device = char(device);
                type = char(type);
                voltage = voltage;
                if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
                   error = 1;
                else
                   error = 0;
                end
             end

          else
             ;
          end
          pause(0.0);
       end;


       if(device == '$0001')
          voltage01 = voltage;
          type = type(1:2);
          if(type == '32')
             if(voltage01 < 1.0)
                message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is
currently available.'];
                datarcvd01 = 1;
             elseif(voltage01 > 1.3 & voltage01 < 2.2)
                message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is
not available; it is currently in washing cycle.'];
                datarcvd01 = 1;
             else
                message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is
not available; it is currently in spinning cycle.'];
                datarcvd01 = 1;
             end;
          else
             if(voltage01 < 1.0)
                message02 = ['The wireless sensor #' device(4:5) ' is installed on dryer. The dryer is
currently available.'];
                datarcvd01 = 1;
             else
                message02 = ['The wireless sensor #' device(4:5) ' is installed on dryer. The dryer is
currently not available; it is currently in operation.'];
                datarcvd01 = 1;
             end;
          end;
          data = fscanf(s);  %gets out of the loop
          [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
          device = char(device);
          type = char(type);
          voltage = voltage;
          if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
             error = 1;
             while(error == 1)
                data = fscanf(s);
                [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
                device = char(device);
                type = char(type);
```

```
                voltage = voltage;
                if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
                    error = 1;
                else
                    error = 0;
                end
            end

        else
            ;
        end
        pause(0.0);
        counter = 1;

    elseif(device == '$0002')
        voltage02 = voltage;
        type = type(1:2);
        if(type == '32')
            if(voltage02 < 1.0)
                message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is currently available.'];
                datarcvd02 = 1;
            elseif(voltage02 > 1.3 & voltage02 < 2.2)
                message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is not available; it is currently in washing cycle.'];
                datarcvd02 = 1;
            else
                message01 = ['The wireless sensor #' device(4:5) ' is installed on washer. The washer is not available; it is currently in spinning cycle.'];
                datarcvd02 = 1;
            end;
        else
            if(voltage02 < 1.0)
                message02 = ['The wireless sensor #' device(4:5) ' is installed on dryer. The dryer is currently available.'];
                datarcvd02 = 1;
            else
                message02 = ['The wireless sensor #' device(4:5) ' is installed on dryer. The dryer is currently not available; it is currently in operation.'];
                datarcvd02 = 1;
            end;
        end;
        data = fscanf(s);  %gets out of the loop
        [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
        device = char(device);
        type = char(type);
        voltage = voltage;
        if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
            error = 1;
            while(error == 1)
                data = fscanf(s);
                [device,type,voltage,strength,other] = strread(data,'%s%s%3.2f%d%s','delimiter',',');
                device = char(device);
                type = char(type);
                voltage = voltage;
                if(size(data) ~= [1 23] | size(device) ~= [1 5] | size(type) ~= [1 2])
                    error = 1;
                else
                    error = 0;
```

```
                end
            end

        else
            ;
        end
        pause(0.0);


    end;

end;

end;

clc
message01 %display the message in commond window in MATLAB
message02 %display the message in commond window in MATLAB
pause(0.2);
fid = fopen('message01.txt', 'w'); %output the message to a text file
fprintf(fid, message01);
fid = fopen('message02.txt', 'w'); %output the message to a text file
fprintf(fid, message02);
datarcvd01 = 0; %by the end of this loop, the data is received.
% Therefore, change the value to 0 to initialize the loop.
datarcvd02 = 0; %by the end of this loop, the data is received.
% Therefore, change the value to 0 to initialize the loop.



end;
```

Appendix D. HTML code source : Main Page

```
<HTML>
<HEAD>
<TITLE>E90 Project</TITLE>
</HEAD>
<FRAMESET rows="30%, 70%">
  <FRAME src="top.html">
  <FRAME src="bottom.html">
</FRAMESET>
</HTML>
```

Appendix E. HTML code source : Top Frame

```
<html>
<head>
<title>Instruction Page</title>
</head>
<body>
<p>This is part of the E90 project.
<p>This is a top frame.

<p>The bottom page reloads every 2 seconds.
<p>Please wait for a seconds for data update
<b><p><*insert any direction here*></b>

</body>
</html>
```

Appendix F. HTML code source : Bottom Frame

```
<html>
<head>
<title>Instruction Page</title>
</head>
<body>
<p>This is part of the E90 project.
<p>This is a bottom frame.
<b><p>all the status are provided below:</b>
<p>
<p>message 01
<iframe src="message01.txt"
width="100%" height="10%"
align="left"></iframe>
<p>
<p>message 02
<iframe src="message02.txt"
width="100%" height="10%"
align="left">
</iframe>
<meta http-equiv="refresh" content="2" >
</body>
</html>
```