ECE 492 - Final Report

NFC Sensor System

Using NFC to read attached sensors to a Bluetooth module and phone

Eric Anderson	Nigel Himmelreich	Mike Pappas

Using an NFC tag to read sensor data and send results over NFC to be read via Bluetooth

Group Number: 2

Abstract

The purpose of this paper is to outline our design and results for the ECE 492 Capstone Design project. For our project we have created an induction powered NFC module fitted with a temperature and light sensor, coupled with a reusable NFC to Bluetooth module using the Altera DE0 Nano board. This project was inspired by the idea of a disposable ‘smart-bandage’ that can monitor the status of a wound without needing to remove the bandage. This project is a proof of concept to show that it is possible to transmit sensor information over NFC in a relatively small form factor using induction power. We have also designed a companion Android application that will be able to read both NFC and Bluetooth data and display it to the user.

We determined that while a disposable NFC sensor tag was certainly possible, it was outside the scope of our time and abilities for this design course. As a result, we have successfully produced a non-disposable sensor module that can be powered using only the NFC induced current. Unfortunately due to ISO-14443B protocol issues with the PN532 NFC shield, we were unable to read more than serial number data using our reader module - however transmitting this data via Bluetooth was a success. Our Android application is able to successfully read both NFC and Bluetooth data and display it to the user in a convenient graph. Including the cost of the provided DE0 Nano, the total cost of this project was $293.90, along with taxes and shipping for the components.

Table of Contents

Functional Requirements

The goal of this project was to produce an inductively powered sensor module, using the NFC RF field, with the intention of proving the ‘smart bandage’ concept feasible. Along with this sensor module, a ‘reusable’ reader module was created to repeatedly obtain sensor data from the sensor module via NFC, then transmit this data over Bluetooth to a host device. For our application, we also created an Android companion application to assist with storing and displaying historical sensor data to the user.

The sensor module obtains the analog sensor data over I^2C and obtains a digital reading of the signal using a 12 bit analog to digital converter. The TI MSP430 MCU then packages the data into an NFC NDEF message loads it into the registers of the TI RF430 NFC module to be transmitted. The sensor module obtains power from the NFC RF field, but can also be powered from any USB power source.

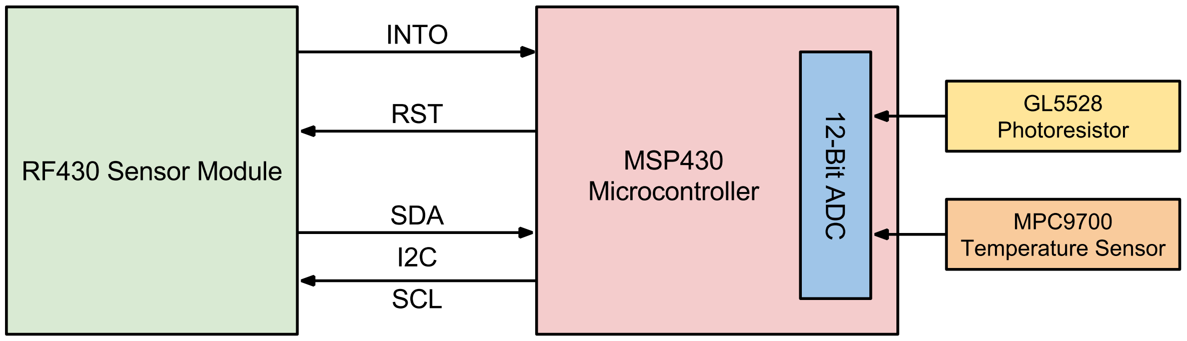
The reader module is intended to perform frequent NFC readings of the sensor module using the PN532 NFC module, as well as provide a continuous RF field to power the device. Once the reader has obtained the message via NFC, the data is transmitted over I^2C to the DE0-Nano FPGA, which decodes the NDEF message and obtains the readings. These readings can then be sent as strings over a serial connection to the Olimex MOD-BT Bluetooth module, which transmits the data to a host device. Upon startup, the reader module broadcasts as a pairable device, where any host capable of accepting serial Bluetooth data can connect.

The Android application is capable of using the provided Android NFC and Bluetooth API to obtain, store, and display sensor data to the user. After pairing with the Bluetooth module of the reader, the smartphone will obtain the serial Bluetooth data and store it for history and graphing. In addition, using an NFC capable smartphone, the application can read NFC sensor data directly from the sensor module and display this data on a convenient graph over different time ranges.

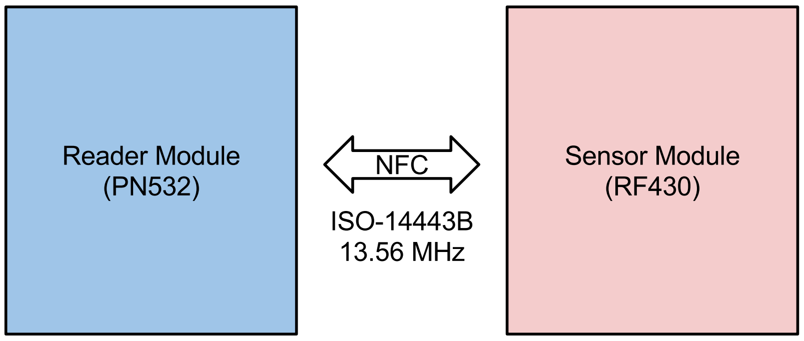
The functional requirements were met on the NFC sensor module side. The system is able to successfully obtain accurate data from the sensors and transfer them via NFC, and is able to do so using only RF power. This can be demonstrated using the Android application on an NFC enabled smartphone. Unfortunately the reader module is only able to obtain RF430 (sensor module) serial number and device ID due to unforeseen compatibility issues with the PN532 and the NFC ISO-14443B protocol. However, the reader module is able to successfully broadcast this device ID over Bluetooth demonstrating that the reader module’s NFC-Bluetooth interface is functional. Finally, the Android application successfully met all functional requirements, as it is able to display and store sensor data from both NFC and Bluetooth data sources.

Design and Operation

Sensor Module

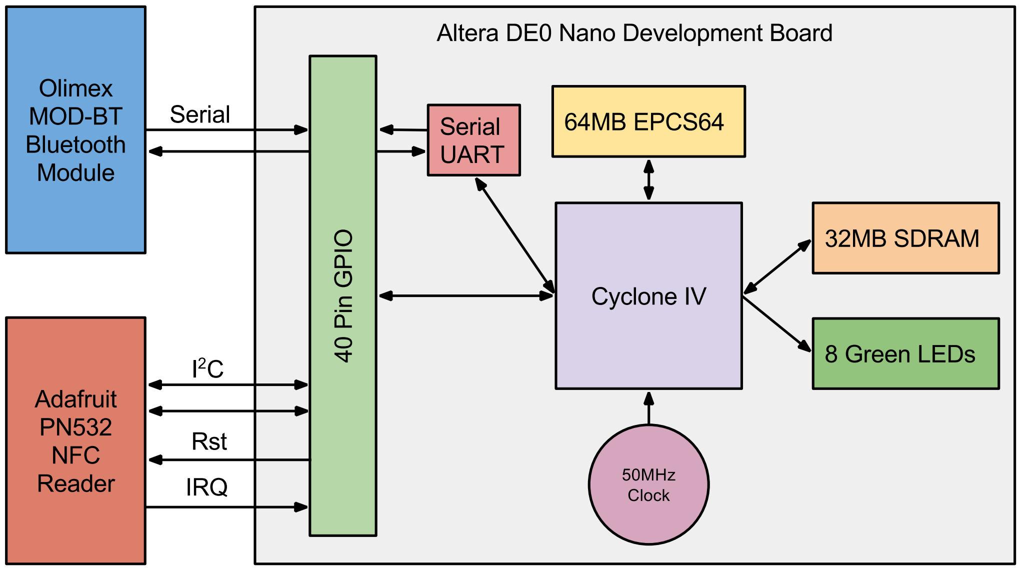


For the sensor module, two sensors (the GL5528 photoresistor and MPC9700 temperature sensor) are connected to the 12-bit analog to digital converter of the MSP430 microcontroller. As the values produced by these sensors change, the ADC measures these voltages and converts the signal to a digital value. These digital values are then passed over I2C connection to the RF430 Sensor Module.



At this point, the data is transferred over an NFC connection using the ISO-14443B protocol. This is a 13.56 MHz data connection that unfortunately utilizes a proprietary coding scheme and protocol initialization procedure. Since the RF430 is a passive NFC device, it requires a master NFC transceiver to initiate and transmit the protocol commands. For our design, the transceiver is represented by either the Adafruit PN532 NFC shield or an Android smartphone. The data is packaged in a NDEF (NFC data exchange format) message and stored within the RF430 module until a transceiver activates the device and requests the data message.

Reader Module



On the reader module, the Adafruit PN532 NFC reader is initialized as the master device for communicating with the sensor module. This is connected to the Altera DE0 Nano along with the Olimex MOD-BT Bluetooth module. To initialize the NFC reader, the DE0 Nano issues commands over the I^{2}C connection, pausing and waiting for responses from the PN532 reader. The commands are pulling the reset line high/low to reset the PN532 chip, configure the NFC reader to wait for tags and then trigger an IRQ, and then tell it to wait for a type B tag. When the NFC reader detects a tag within its field, it starts a connection with the RF430 module and issues an IRQ to the DE0 Nano, which on the MicroC/OS running on the DE0 Nano, it triggers an interrupt within the system. This interrupt releases a semaphore allowing the task to continue running after it was paused when it finished initializing the PN532 reader. The rest of task issues commands over I^{2}C connection to retrieve the target data collected by the PN532 reader. Included in this target data is the ID of the NFC tag. The ID is passed as string via a queue to the Bluetooth module.

The Olimex Bluetooth module provides the data from the NFC reader to the Android or other Bluetooth device. The DE0 Nano receives new ID data from the NFC task in the form of a string, which the Bluetooth task sends directly over the serial connection to the Bluetooth module. The Bluetooth module is initialized to accept any serial data as a server and retransmit it to any Bluetooth connected device that knows its name and pin number as serial data. The DE0 Nano continues to retransmit previously stored ID string until a new ID is read by the NFC reader.

Android Application

The Android app can be used with either the NFC tag on the sensor module or with the Bluetooth on the reader module. The first way the sensor readings can be taken is by NFC, which requires the phone to ‘tap’ against the RF430 module and it will issue a command to read the NDEF message. The second way to get sensor readings is via the Bluetooth connection, when the reader module is over top of the sensor module or when it recently read from the sensor module the Bluetooth connection receives data from the DE0 Nano, currently this is just the ID of the NFC tag. The app stores the values for the temperature (in Celsius) and the light value, a relative value with closer to 0 being brighter and near 1200 being near darkness. This data is stored in a XML file and is able to be viewed based on the date and time the reading was taken. The same data is used to generate a 1, 7, or 30 day graph for temperature and light values.

Bill of Materials

QTY	Part Name	Key Spec	Product URL	Datasheet	Cost Each ($)
Reader Module					
1	Altera DE0-Nano FPGA	FPGA, Cyclone IV, onboard SDRAM, EEPROM	DE0-Nano URL	DE0-Nano Datasheet	$76.13
1	Adafruit PN532 NFC/RFID Controller Shield	Philips PN532 NFC chip; SPI, I^2C , UART	PN532 URL	PN532 Datasheet	$51.73
1	Olimex MOD-BT Bluetooth Module	Serial UART	Olimex MOD-BT URL	Olimex MOD-BT Datasheet	$35.18
2	10kOhm Resistors	10kOhm	N/A	N/A	$0.20
1	Perf Board	N/A	N/A	N/A	$7.50
1	40-Pin Ribbon Cable	40 Pin Headers	N/A	N/A	$10.00
4	Nylon Standoffs and Screws	N/A	N/A	N/A	$0.50
1	40-Pin Header	N/A	N/A	N/A	$0.25
1	10-Pin Header	N/A	N/A	N/A	$0.10
2m	Wire	N/A	N/A	N/A	$0.75
Sensor Module					
1	TI MSP-EXP430FR5969 MCU Development Board	Ultra-Low Power; 16 MHz MCU	MSP-EXP430FR5969 URL	MSP-EXP430FR5969 Datasheet	$32.27
1	DLP Design RF430CL330 NFC Boosterpack	RF430CL330H with an antenna	DLP RF430CL330H URL	DLP RF430CL330H Datasheet	$12.88
1	Microchip Technologies MCP9700 IC Sensor Thermal	From -40 to +125°C with ±1°C accuracy	MPC9700 URL	MPC9700 Datasheet	$0.48
2	RF430CL330H	RF430CL330H NFC DYNAMIC TAG TARG	RF430CL330H URL	RF430CL330H Datasheet	$25.55
6	CDBU0130L	DIODE SCHOTTKY 30V 100MA 0603	CDBU0130L URL	CDBU0130L Datasheet	$0.58
1	Photodiode GL5528	Photocell Resistor	GL5528 URL	GL5528 Datasheet	$1.50
4	10kOhm Resistors	10kOhm	N/A	N/A	$0.20
1	1uF Capacitor	1uF	N/A	N/A	$0.20

1	0.1uF Capacitor	0.1uF	N/A	N/A	$0.15
1	Perf Board	N/A	N/A	N/A	$5.00
4	Nylon Standoffs and Screws	N/A	N/A	N/A	$0.50

Total Cost:	**$293.90**

Note: 3.3V Power supply is used, but not included in the total cost.

Reusable Design Units

Various sources of reusable design were available for our chosen hardware. Since the PN532 NFC Shield was used by a capstone group from a previous year, their code for the Altera DE2 was used to create some of the software for the DE0. For the sensor module, Texas Instruments provided full documentation as well as a fully functioning “Hello World” example featuring the RF430 and MSP430. Using this example code, we were able to modify it to fit our needs and build upon it to add additional features and functionality such as the 12 bit ADC and two analog sensors. Finally, the MOD-BT Bluetooth module provided by Olimex also provided example code for interfacing via the serial connection with the Bluetooth module including how to initialize the device and how to write values to the device.

In order to interface with our Bluetooth device, we needed to create a basic Android application. Google provided official documentation for operating the Bluetooth hardware onboard any Android device. In addition, they provided an example application that is capable of transmitting and receiving arbitrary data from another Bluetooth device.

Name	Source Location	Reusable Design	Source Size	Compiled Size
PN532 for the Altera DE2	https://www.ualberta.ca/~delliott/local/ece492/appnotes/2014w/G5_NFC_Reader_PN532/	-samConfiguration, inListPassiveTarget, readResponse	21.2KB	N/A
TIDA-00217	http://www.ti.com/tool/TIDA-00217#technicaldocuments	-RF430 Initialization, Read/Write Functions. -MSP430/RF430 example. -NFC Sensor Module Reference Design.	32.8KB	N/A
MOD-BT	https://www.olimex.com/Products/Modules/RF/MOD-BT/	-Bluetooth Project for Olimex 328 with Arduino IDE	21.0KB	N/A
Android Bluetooth	http://developer.Android.com/guide/topics/connectivity/Bluetooth.html http://developer.Android.com/samples/BluetoothLeGatt/index.html	-Bluetooth Permissions, Device Discovery, Connecting as a Client Example -Bluetooth Socket Input/Output Streams -Bluetooth LE Generic Attribute Profile (GATT) to transmit generic data between two devices	N/A	N/A

Datasheet

Interfaces:

Interface	**Purpose**
Bluetooth RF (physical layer)	Bluetooth Communication
Serial Peripheral Interface (SPI)	NFC to DE0, NFC to Sensor MCU
UART, I2C	DE0 to Bluetooth
ISM band of 13.56 MHz on ISO/IEC 18000-3	NFC Communication

IO Rates:

Component	**Maximum IO Rate**
RF430CL330H NFC Transponder	848 kbps (RF) [6]
MSP430FR5969 Ultra-Low Power MCU	10 Mbps (SPI) [4]
Olimex MOD-BT Bluetooth Module	1 Mbps (BT Ver. 1.2) [8]
Adafruit PN532 NFC/RFID Controller	424 kbps (RF) [1]

Maximum Clock Rates:

Component	**Maximum Clock Rate**
DE0 Development Board	50 MHz [7]
MSP430FR5969 Ultra-Low Power MCU	16 MHz [4]

Power Usage (Sensor Module):

Measurement	**Value (mW)**
Start Up Power	24.69
Idle Power	9.58
Active Power	17.5
Average Power	11.82

Note: These values were measured use Code Composer Studio 6 EnergyTrace technology.

Power Usage (Reader Module)

Measurement	**Value (mW)**
Active	231

Note: Current draw of 0.07A at 3.3V = 0.231W

Recommended Operating Conditions:

Component	**Operating Value**	**Min**	**Typical**	**Max**
DE0 Development Board (Cyclone IV) [7]	Voltage (V)	-0.5		3.6
	Current (Diode) (mA)			10
	Temperature (℃)	0		85

Adafruit PN532 NFC/RFID Controller [3]	Voltage (V)	2.7		5.4
	Current (mA)		25	30
	Temperature (℃)	-30		85
Olimex MOD-BT Bluetooth Module (Peripheral) [8]	Voltage (V)	1.8		3.3
	Current (mA)	N/A		N/A
	Temperature (℃)	-30		85
MSP430FR5969 MCU [4]	Voltage (V)	1.8		3.6
	Current (Diode) (mA)			±2
	Temperature (℃)	-40		85
RF430CL330H NFC Transponder [6]	Voltage (V)	2.0	3.3	3.6
	Current (Diode) (mA)			±2
	Temperature (℃)	-40		85

IO Table:

Unit	Signal Name	Wires In Bus	Signal
Temperature Sensor [2]			
	Temperature Output	1	To MCU
	Temperature Ground	1	To off board power supply or NFC antenna
	Temperature Voltage	1	To off board power supply or NFC antenna
Light Sensor [4]			
	Voltage	1	To off board power supply or NFC antenna
	Output	1	To MCU
NFC Controller - 1 [6]			
(Sensor side)	NFC Voltage	1	To off board power supply or NFC antenna
	NFC Ground	1	To off board power supply or NFC antenna
	NFC SCK	1	To MCU
	NFC CS	1	To MCU
	NFC Interrupt	1	To MCU
	NFC Reset	1	To MCU
MCU [4]			
(Sensor side)	Voltage	1	To off board power supply or NFC antenna
	Ground	1	To off board power supply or NFC antenna
	SCK	1	To NFC

	CS	1	To NFC
	Sensor Inputs	4	To temperature & light sensors
	Interrupt	1	To NFC interrupt
	Reset	1	To NFC reset
NFC Controller - 2 [3]			
(DE0 side)	NFC Voltage	1	To off board power supply
	NFC I2C SCA	1	FPGA to board
	NFC I2C SDL	1	FPGA to board
	NFC IRQ	1	FPGA to board
	NFC RST	1	FPGA to board
	NFC Ground	1	To off board power supply
Bluetooth Unit [8]			
	Bluetooth Transmit	1	FPGA to Board
	Bluetooth Receive	1	FPGA to Board
	Bluetooth Voltage	1	To off board power supply
	Bluetooth Ground	1	To off board power supply

Required power supply:

Bluetooth: 3.3V
NFC: 3.3V
MCU: 3.3V
Temperature Sensor: 3.3V
Light Sensor: 3.3V

Overall: 3.3V

Background Reading

TI - Dynamic Field-Powered NFC Reference Design for Data Logging, Access Control
http://www.ti.com/lit/pdf/tidu378

This article provided insight regarding powering the NFC and sensor chips inductively based off power from the antenna. It included schematics and suggested parts which is useful for our project in determining what parts to order and how to correctly attach them.

IEEE - An NFC transceiver using an inductively powered receiver for passive, active, RW and RFID modes (SoC Design Conference 2009)
http://ieeexplore.ieee.org.login.ezproxy.library.ualberta.ca/stamp/stamp.jsp?tp=&arnumber=5423924

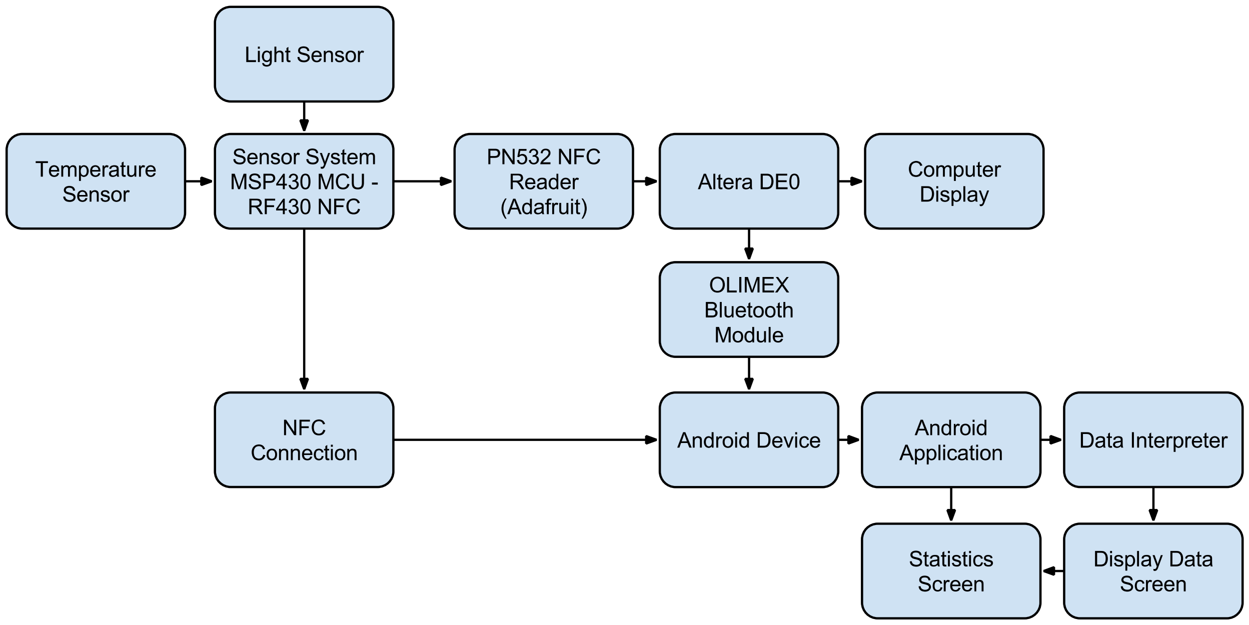
This article provided background to NFC specifically round the wireless connection and the powering of an NFC chip. This provided insight into the theoretical way to power an NFC chip via induction. This paper was not directly helpful in our design, but did allow us to determine an inductively powered NFC system is possible.

Software Design

The MSP430 microcontroller is designed to run code located on the built in flash memory. For the purpose of our design requirements to obtain and update sensor readings for each NFC interaction, the extra resources and processing power required for a RTOS would not have been suitable from an induction powered design standpoint. As a result, there is no operating system running on the MSP430. The code on the MSP430 is design to have quick initialization and does not rely on previously stored data, since it is required to perform immediate sensor readings when power is received, and is able to lose power mid-reading with no lasting effects.

The DE0 Nano runs the MicroC/OS operating system. There are 2 tasks in the system, 1 runs all communication with the NFC reader, the other runs the communication for the Bluetooth module. Both interact with a shared queue in which the NFC task pushes strings of data onto an appropriately sized queue of 10 values which is more than the NFC task can push into the queue in the time it takes the Bluetooth task to write the values out. The Bluetooth task can write values every second, where the NFC task can only read values approximately every 3 seconds from time of last read and until it has initialized itself and detected the tag again.The NFC task also uses an semaphore tied to an interrupt, so that when the NFC reader issues an IRQ, the interrupt in MicroC/OS releases the semaphore to allow the NFC task to continue. The NFC task blocks after sending commands and waiting for an ACK and then pauses again to wait for the PN532 to issue a response, this repeats for commands used for initialization and retrieving data.

The last of the software portion would be the Android application. The application has been built to be able to accept input from NFC or a Bluetooth connection. The NFC connection receives a NDEF message and the Bluetooth connection receives a string. The sensor values can be obtained from each of this, the NDEF message contains the message after a header and the Bluetooth connection sends the data as a string that can be split into both values. When the NFC activity is active on the Android application, when the phone is tapped to the sensor system the NDEF message is retrieved. When the Bluetooth activity is active, the phone waits for the Bluetooth connection to receive a string, which then closes the activity. In both cases the sensor values are extracted from either the NDEF message or string in the NFC and Bluetooth activity respectively. Once the data has been extracted it is displayed to the user. The data is also saved to memory with the date and time it was received so it can be reviewed later. Lastly a graph can be made from stored data for 1, 7, or 30 days along with a graph for each sensor value of light and temperature.



<u>Test Plan</u>

<u>Hardware</u>

To test the accuracy of the sensor output, we first measured the voltage output of the sensors using an oscilloscope. For the temperature sensor, we expected an output voltage of approximately 700mV corresponding to room temperature (baseline of 500mV for 0℃, +10mV per 1℃) [2]. By placing our fingers on the sensor, we looked to see the voltage rise and then drop back to ~700mV after releasing it. To test the light sensor, we again placed an oscilloscope on the corresponding analog input pin of the MSP430 and observed as the voltage changed based on the amount of light exposure. A voltage of 1200mV (the reference voltage) indicated little or no light was hitting the photoresistor, while lower values approaching 200mV indicated higher light values.

To test the analog to digital conversions of the MSP430, the debug editor of Code Composer Studio (an Eclipse-based IDE) was used to monitor the internal registers of the 12-bit ADC. The procedure outlined above was carried out once again to test the functionality of the sensors, this time looking to see that the values in the ADC registers corresponded to the oscilloscope temperature at that time.

To test the NFC functionality of the sensor module an Android smartphone was used to obtain the sensor values, comparing them to the values stored in the ADC registers. Since the our code was built upon the manufacturers example software to transmit "Hello World" over NFC, minimal testing was required for the RF430 to Android device functionality.

To test the induction power, an oscilloscope was placed on the Vcc and Gnd pins of the MSP430 and the device (along with the attached RF430) was brought into range of the PN532 RF field, which was broadcasting a constant field. Android devices were also tested, again by bringing the devices within range of the RF430 and observing the output voltage on the oscilloscope.

Software

To test the Android application after the NFC API was integrated into the application, the smartphone was simply brought into the range of the RF430 to perform a read, and the log of Android Studio was examined to determine the results. After ensuring that all protocol commands corresponding to the ISO-14443B protocol were entered correctly, a data dump from the NFC transaction was examined after each NFC transmission. By comparing the resulting data with the NDEF message inside the RF430 registers, testing of the NFC API could be performed.

Finally, the DE0 was configured to repeatedly broadcast strings over a serial Bluetooth connection. Using the Android application and pairing with the reader module, the string data could be observed on the output log on Android Studio. By extension, this allowed for any data to be sent over the serial Bluetooth connection.

Experimental Results

To power the sensor module with inductive power, we tested with both smartphone and the reader module. The reader module is able to provide 2.72V at a distance of 0 to 2.5 cm away from the PN532 reader, this enough to provide enough power to the sensor module to continue to function where the MSP430 and RF430 can run. After 2.5 cm, the voltage decreases until approximately 5 cm away, at which the point the voltage has dropped below 1.5V and the sensor module is no longer powered. For testing with the smartphone, at any distance from the sensor module, the phone could not power the module. The is due to the tested phones sending out a field in pulses of a microseconds and with voltage of approximately 2.6V.This is not long enough and barely enough power to start up the MSP430 and RF430 before the field is shut off by the phone.

To transfer data between the sensor module and the smartphone, we observed an average response time of 3 seconds between the smartphone being placed over the NFC tag and the application reporting the values. Some of this time is dedicated to overhead in the Android application for processing and displaying the sensor values. A quicker method of using the built-in NFC application for reading NDEF messages from a tag resulted in a read time of under 2 seconds. For the DE0 Nano, since it uses a real time operating system, measuring the response time can be difficult since the NFC reader has to wait for the system at certain points if the NFC task isn’t the active task. Nevertheless, we measured an average time of 3 seconds from the sensor module being placed overtop until the Android device has received the Bluetooth values. We also measured just over 1 second for the DE0 Nano to register via its LEDs that the sensor module was detected overtop the sensor and the ID was read.

Safety

Since this project is intended to emulate an NFC enabled body sensor, the primary safety concern with this design is bringing a current-carrying device in close contact with one’s skin. However, the current induced by an NFC antenna coil is typically less than or equal to 100µA, which is considered to be below the threshold of human sensation [5]. In addition, since this component operates without a battery as a passive NFC module, no stored charge is held within the circuit, aside from small charges within the ceramic 1uF and 0.1uF capacitors.

The reusable component of the design, which utilizes the DE0 Nano board, is USB powered and should be treated with similar precautions as most consumer electronics, and the supply current depends on the USB source port. The DE0 Nano board could be powered by batteries to ensure a completely wireless device.

The following table describes the maximum operating voltages for the NFC sensor module, and the accompanying reusable module:

Component	Maximum Operating Voltage
RF430 Dynamic NFC Transponder	4.1V [6]
MSP430FR5969 16 MHz MCU	4.1V [4]
DE-0 Development Board	5.7V [7]
Olimex MOD-BT Bluetooth Module	3.3V [8]
Adafruit PN532 NFC/RFID Controller	5.4V [3]

Given this device is operating an RF field in contact with skin, it is important to look at RF exposure in relation to the 13.57Mhz frequency range. Fortunately, the Specific Absorption Rate (SAR) produced by an NFC field is just 11.18 mW/kg, which is far below the recommended maximum 0.4W/kg for continuous exposure [14] [15].

<u>Environmental Impact</u>

With the exception of the GL5528 photocell, all hardware currently in use for our design is RoHS compliant including:

- Texas Instruments MSP-EXP430FR5969 Development Board
- DLP Design RF430CL330HTB NFC Module
- Microchip MPC9700 Thermistor
- Altera DE0
- OLIMEX MOD-BT Bluetooth Development Board

Unfortunately, the GL5528 photocell used to measure the light resistance contains cadmium sulphide, resulting in non-RoHS compliance [10]. To reduce the environmental impact of this device in the future, we could replace the photocell with an RoHS compliant variant, or remove the photocell altogether.

In addition, since the target PCB design is intended to be disposable with a bandage, it is important to note that the components of the PCB are not necessarily recyclable or decomposable - meaning they should be disposed properly as with other electronic devices.

Sustainability

The following table demonstrates the typical power usage for our MSP430 sensor module. Given that our device (assuming it is powered inductively) does not have power, nor store it, when an NFC field is not present, there is no sleep mode:

Measurement	Value (mW)
Start Up Power	24.69
Idle Power	9.58
Active Power	17.5
Average Power	11.82

Assuming the device is running 24/7 on a patient monitoring their temperature:

$$\frac{24\ hours}{1\ day} \cdot \frac{365\ days}{1\ year} \cdot Average\ Power\ (W) = 8760\ h/year \cdot 11.82mW = 103.54Wh/year$$

Given the cost of electricity of 8.7¢/kWh, the average cost per year to run this device 24/7/365 will be 0.9¢/year.

References

[1] Wikipedia. *Near field communication.* [Online] Available: http://en.wikipedia.org/wiki/Near_field_communication Accessed on: March 1, 2015.

[2] Microchip, AZ, USA. *Low-Power Linear Active Thermistor™ ICs* (2014) [Online]. Available: http://www.microchip.com/mymicrochip/filehandler.aspx?ddocname=en022859 Accessed on: March 1, 2015.

[3] Adafruit, NY, USA. *Adafruit PN532 NFC/RFID Controller Shield.* [Online]. Available: http://www.adafruit.com/product/789 Accessed on: March 1, 2015.

[4] Texas Instruments, TX, USA. *MSP430FP59xx Mixed-Signal Microcontrollers.* (2014) [Online]. Available: http://www.ti.com/lit/gpn/msp430fr5969 Accessed on: March 1, 2015.

[5] Elmwood Electric Inc. *Electrical Safety: The Fatal Current.* [Online]. Available: https://www.physics.ohio-state.edu/~p616/safety/fatal_current.html Accessed on: March 1, 2015.

[6] Texas Instruments, TX, USA. *RF430CL330H Dynamic NFC Interface Transponder.* (2014) [Online]. Available: http://www.ti.com/lit/ds/symlink/rf430cl330h.pdf Accessed on: March 1, 2015.

[7] Altera, CA, USA. *ALTERA Cyclone IV Development & Education Board (DE0-Nano).* [Online]. Available: ftp://ftp.altera.com/up/pub/Altera_Material/12.1/Boards/DE0-Nano/DE0_Nano_Datasheets.zip Accessed on: March 1, 2015.

[8] Olimex, Bulgaria. *MOD-BT development board.* (2009) [Online]. Available: https://www.olimex.com/Products/Modules/RF/MOD-BT/resources/MOD-BT.pdf Accessed on: March 1, 2015.

[9] Texas Instruments. *Dynamic Field-Powered NFC Reference Design for Data Logging, Access Control, and Security Applications.* [Online]. Available: http://www.ti.com/lit/ug/tidu378a/tidu378a.pdf Accessed on: March 1, 2015

[10] MicroController Pros LLC. *Mini Photocell, GL5528, Luminosity Sensor (LDR).* [Online]. Available: http://microcontrollershop.com/product_info.php?products_id=3469 Accessed on: March 1, 2015

[11] Altera, CA, USA. *RS232 UART Core for Altera's DE2/DE1 Boards* (2006). [Online]. Available: ftp://ftp.altera.com/up/pub/University_Program_IP_Cores/RS232.pdf Accessed on: March 1, 2015.

[12] K.Y. Lau, *Tutorial of Interfacing with RS232 UART*. University of Alberta, Edmonton, AB, CAN. [Online]. Available: https://www.ualberta.ca/~delliott/local/ece492/appnotes/2014w/G3_UART_Interface/G3_UART_Appnote_V2.pdf Accessed on: March 1, 2015.

[13] D. Fiske, M. Lam, D. Tiam. *Altera DE2 I2C Driver*. University of Alberta, Edmonton, AB, CAN. [Online]. Available: https://www.ualberta.ca/~delliott/local/ece492/appnotes/2014w/G5_I2C_Driver_Improved/G5%20-%20I2C%20Driver%20Appnote.pdf Accessed on: February 23, 2015.

[14] D. Fiske, M. Lam, D. Tiam. *PN532 NFC Reader/Writer Library*. University of Alberta, Edmonton, AB, CAN. [Online]. Available: https://www.ualberta.ca/~delliott/local/ece492/appnotes/2014w/G5_NFC_Reader_PN532/G5%20-%20PN532%20NFC%20Reader%20Appnote.pdf Accessed on: February 23, 2015.

[15] S. Cecil, G. Schmid, K. Lamedschwandner. *Numerical Assessment of Specific Absorption Rate in the Human Body Caused by NFC Devices.* [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5476463 Accessed on: March 2, 2015

[16] The Institute of Electrical and Electronics Engineers, Inc. *IEEE Standard for Safety Levels with Respect to Human Exposure to Radio Frequency Electromagnetic Fields, 3 kHz to 300GHz*. (2006). [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1626482 Accessed on: March 2, 2015.

[17] Stack Overflow, *How to receive serial data using Android Bluetooth.* [Online]. Available: http://stackoverflow.com/questions/13450406/how-to-receive-serial-data-using-Android-Bluetooth Accessed on: Feb. 23,2015

[18] Google INC. *Near Field Communication*. [Online]. Available: https://docs.google.com/a/ualberta.ca/document/d/1s5kqj3OUzZy6NgkHPDNF5X_1d5mg_0svK3-ShJzjvP0/edit#heading=h.tccz9tj6wokz. Accessed on: Feb. 1, 2015

[19] T. Witt / P. Thanigai. *MSP430FR59xx_adc12_01.c.* Texas Instruments. [Online]. Available: http://www.ti.com/lit/zip/slac536. Accessed on: Feb 13th, 2015

[20] Texas Instruments. *RF430 Examples*. [Online]. Available: http://www.ti.com/lit/zip/tidc487. Accessed on: Feb 13th, 2015

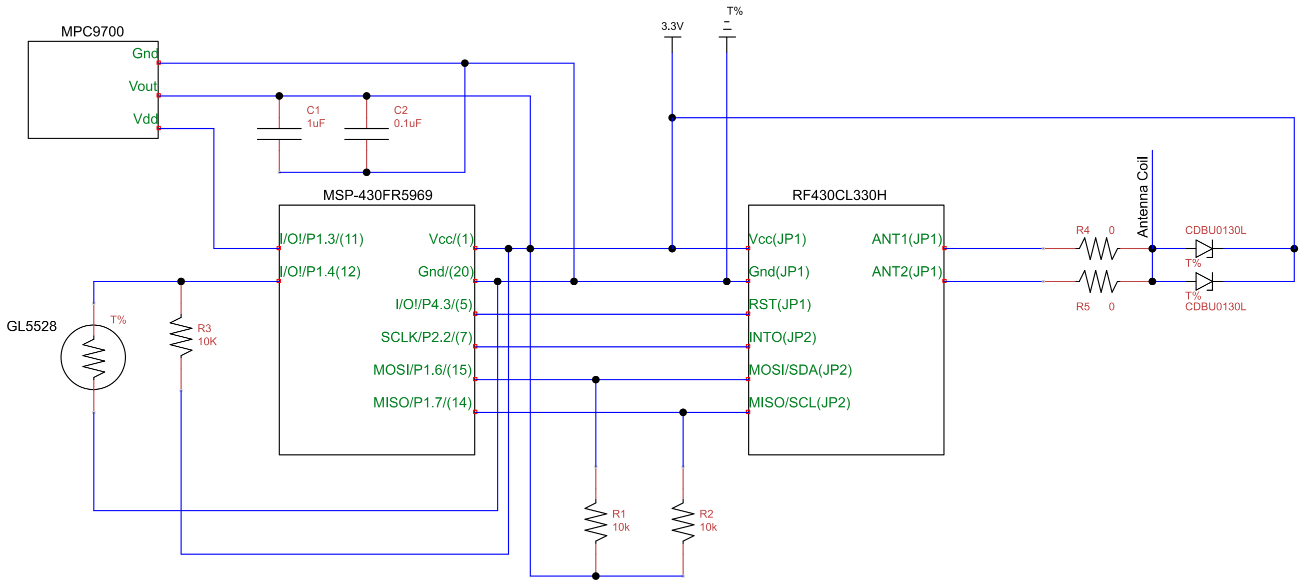
[21] NXP. *PN532 User Manual.* NXP, The Netherlands. [Online]. Available: http://www.nxp.com/documents/user_manual/141520.pdf Accessed on: March 9, 2015

Appendix

Quick Start Manual

Sensor Module

To assemble the sensor module for both induction and externally powered operation, first obtain all of the materials listed in the Bill of Materials section above under the "Sensor Module" heading. Connect the RF430CL330H target board, sensors, and diodes according to the following schematic:



Note that the CDBU0130L diodes must be soldered. Pins must also be soldered to the RF430CL330H board, since only through-holes are provided by the manufacturer. If an external power source is to be used, the power jumper on the "Power Select" section of the board must be changed from "Debugging" to "External". An external 3.3V source can then be connected to the "External Power" Vcc and Gnd pins.

Connect the MSP430 board to a Windows or Linux computer with a USB cable. Install Code Composer Studio 6 (an Eclipse based IDE) from the Texas Instruments website and open it. Create a new CCS Project with the target being a MSP430FR5969. Import the main.c, RF430.c and .h, and the Sensor.c and .h files into the project, replacing any existing main file. Build the project and load the code onto the MSP430 using the debug button. The software is now loaded into the board and it can be disconnected from the computer, if desired.

To perform a reading, simply bring a compatible ISO-14443B NFC device (such as an Android smartphone) in range of the RF430 and wait for a response. The MSP430 will push new sensor data to the RF430 after every read, so frequent readings will produce a smooth gradient of results. Since the RF430 is a passive tag however, large changes in sensor data between readings will need to be pushed to the RF430 before a reading is done, or else multiple readings will need to take place. To push new sensor data to the device, simply push button 4.5 at the bottom left of the board.

For induction power, readings can be done using a device that produces a sufficiently powerful and constant RF field, such as the PN532 or a dedicated NFC reader. Some smartphones may work, however many smartphones will only ‘pulse’ their field at set intervals rather than maintain a constant field, so results may vary.

Reader Module

To setup the reader module, you will need the parts listed in the bill of materials above. Once gather, the next step is to connect all the components, first connect the 40 pin ribbon cable from GPIO 00 on the DE0 Nano to headers on the prototype board. For the Bluetooth module, connect pin 1 to the 3.3V supply, pin 2 to ground, pin 3 to pin 2 on the ribbon cable, and pin 4 of the Bluetooth module to pin 4 on the ribbon cable. Next for the PN532 NFC reader, this first requires the soldering of the pins onto the board to complete assembly. Next connect the pin on the NFC reader labeled 5V to the 3.3V supply in order for the board to operate at 3.3V needed for the DE0 Nano, take one of the pins labeled GND and connect to the both the voltage supply ground and the DE0 Nano ground (pin 12) on the ribbon cable. Next on the NFC reader, connect the pin labelled Analog In 5 (the SCL) to pin 14 on the ribbon cable, Analog In 4 (the SDA) to pin 16 on the ribbon cable, Digital I/O 2 (the IRQ) to pin 16 on the ribbon cable, and then solder a pin to the RST hole and wire it to pin 18 of the ribbon cable. Now add a 10kOhm resistor between Analog In 5 and 3.3V and another resistor between Analog In 4 and 3.3V, these are our pullup resistors for the I^{2}C communication.

With the module wired up we can extract the existing project from the .qar file and program the DE0 Nano with the provided system from Quartus. This has already been compiled and is under output_files/DE0.sof. Next open the NIOS II Software Build tools for Eclipse. Create a new project under the “Nios II Application and BSP from Template” and use the niosII_system.sopinfo file and the “Hello-MicroC/OS-II” template. Upon creation, delete the existing .c file and replace with the files found in software section of the zip file. Rebuild the project, turn on the voltage source, and then program the software to the DE0 Nano. Optionally the software can be flashed to the board using the flash programmer. Note that the power supply must be on before the DE0 Nano is provided power.

Now with everything running, you can connect to the Bluetooth module based on the name and pin found in the G2Cap.c under the Bluetooth task using your Android smartphone with the application installed as per below. When the sensor module is brought over top of the reader it will transmit the ID of the tag.The lights on the DE0 Nano signify where the tasks are when not connected to a computer. The table below summarizes the usage of the LEDs which are split into 2 different sections, one for the NFC task and one for the Bluetooth task.

USB Plug Side									FPGA Side
NFC					Bluetooth				
Initializing PN532	ON	ON	ON	ON	ON	ON	ON	ON	Initializing Bluetooth

Waiting for target			ON	ON			ON	ON	Wait for new data
Found a target		ON		ON		ON		ON	New data arrived
					ON			ON	Repeating last data

Android Application

As this app is currently not on the market the phone settings need to be set to accept apps from other sources. Download the apk file on to the device and install it. From there, the app should open to the main screen. For the app to work correctly, please make sure the NFC and/or bluetooth is on depending on use. The apk of the current version may be download by going to https://onedrive.live.com/redir?resid=18268289d525bc02!70860&authkey=!ANtVwih7vAWY7zs&ithint=file%2capk.

Along the top is the menu bars. From left to right: “Get Reading” has the application set up to take an NFC reading directly from the sensor, “History” selection opens a view to select past history, while “Bluetooth” opens the app and gets it ready to receive an incoming string of data.

When the NFC section is open it just requires touching the device onto the sensor module. Data is received and parsed. Similarly with the bluetooth, when opened it waits for the incoming string. Once received it is parsed. Once the data is parsed in by NFC or bluetooth it is then displayed to the user.

The history can be used to view all old data by selecting the date from the list. The list can be scrolled through. By going into the graph displayed, it will displays all history, temperature by default. Various day increments can be selected from the menu, and the graph type can be displayed by selecting the buttons on the bottom to switch between temperature and light.

To be able to debug or make modifications to the program have a version of eclipse installed with the android sdk. Import the project into the eclipse and edit it like any other java project. Eclipse automatically debugs the app if the device being tested on is plugged into the computer, and you just run the project like any other java project.

Future work

To demonstrate a NFC sensor system concept we used temperature and light sensors. If this device were to be implemented as a smart bandage, more relevant sensors would need to be used to monitor the wound such as moisture/humidity, oxygen, and ammonia. As studies and tests are performed, algorithms and processes could be developed within an application to utilize these sensors to their fullest and predict the status of a wound. Since the light sensor was included in our design as simply a placeholder for other sensors, it would be removed from any future design featuring these new sensors.

This device could also be reduced to the size of a 1x1 inch PCB featuring only the MSP430, RF430 and surface mounted sensors, along with an antenna coil at the perimeter of the device. This design would be inductively powered only, and with some optimizations to the software could be read directly from a smartphone. This route would help lower the cost of the design, and improve practicality as the device size shrinks.

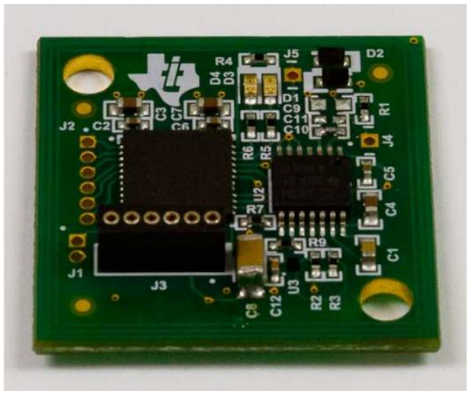


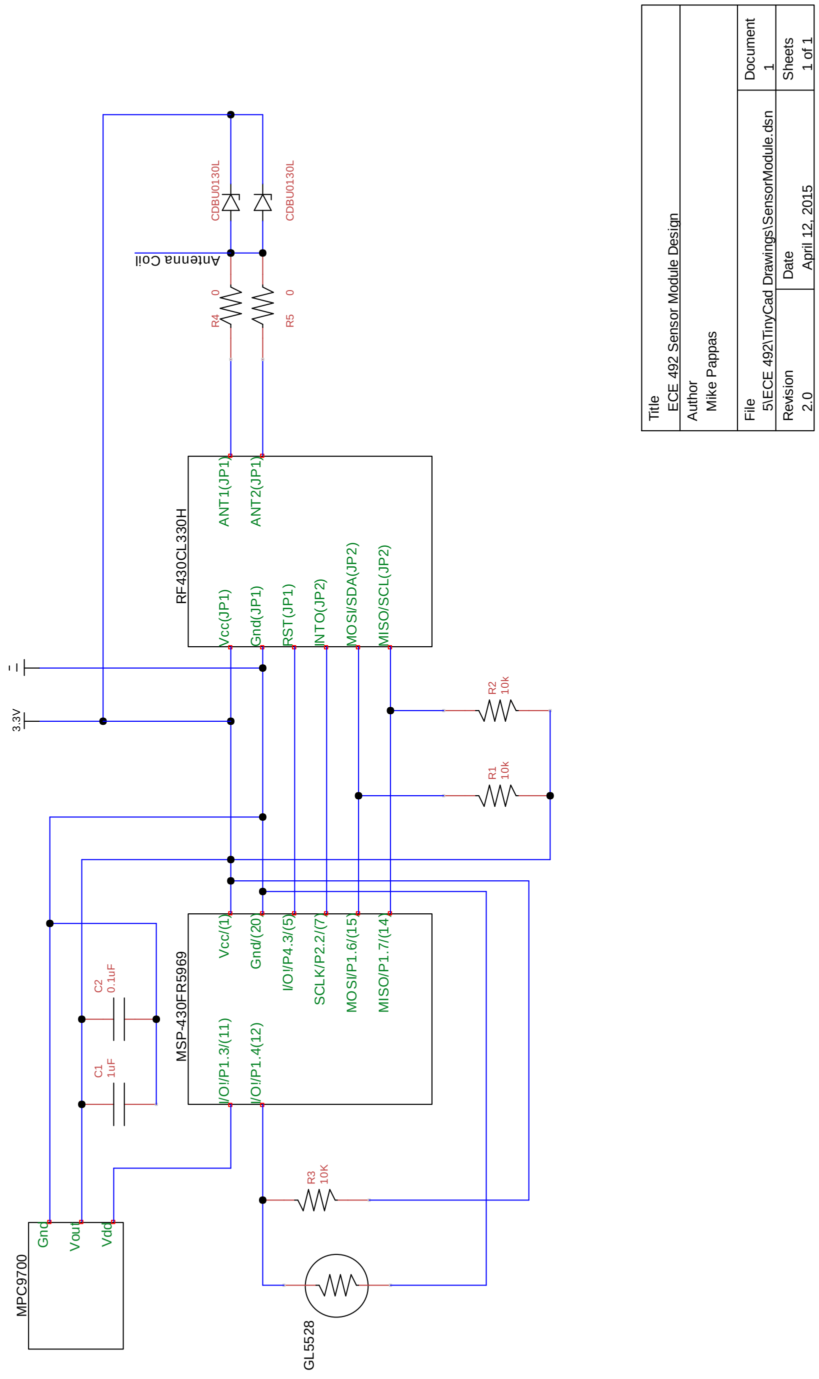
Figure 1 - Reference PCB design [3]

The reader module could likely be integrated onto a single PCB as well, featuring more compatible and reliable hardware than the PN532, and a smaller microcontroller unit than the DE0 Nano. This would allow for the device to be enclosed within an airtight and waterproof capsule, allowing it to be sterilized and cleaned for repeated use. This would be more practical in situations where the reader module is used by a hospital on more than one patient, since the device would need to be sterilized before additional uses.

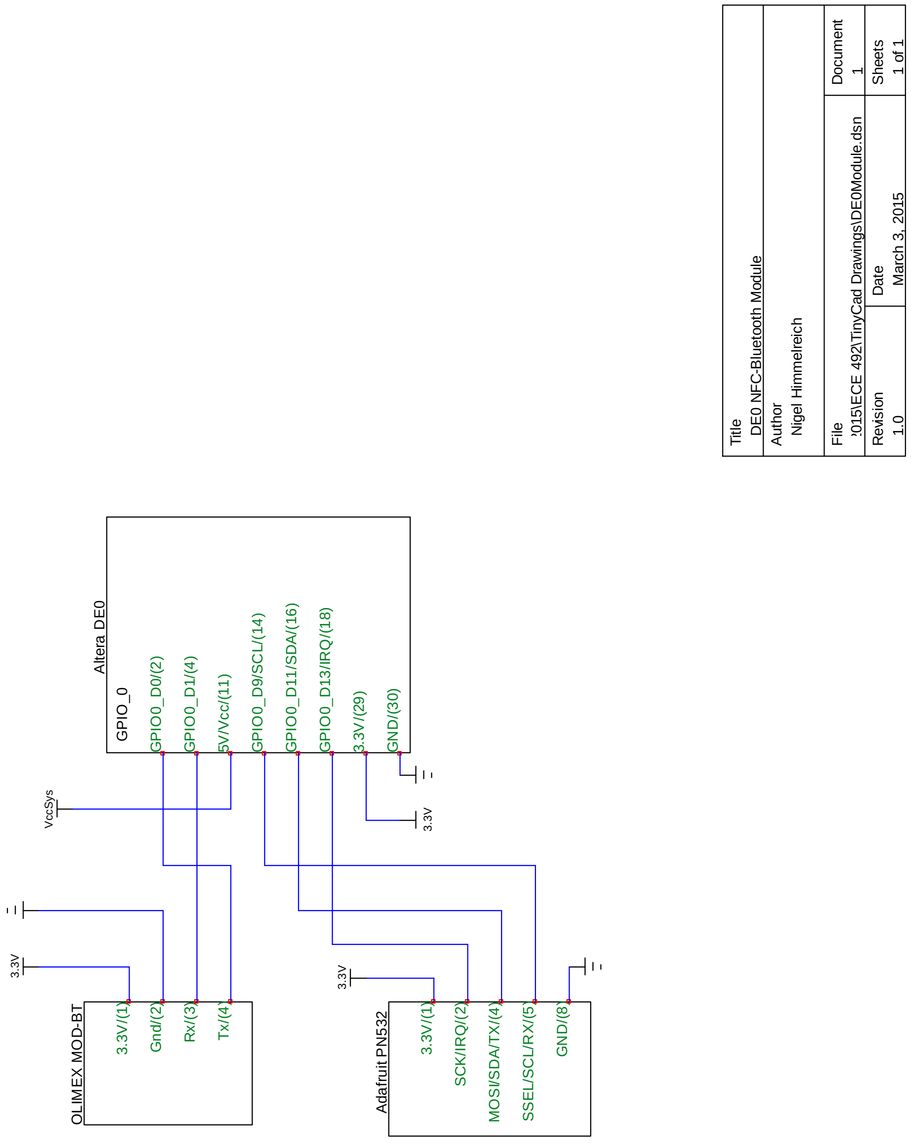
Finally, the entire design could be condensed and integrated into a single IC. Analog sensors could be swapped for digital counterparts which would eliminate the need for an ADC. This would provide both a boost in performance, and a reduction in cost. Since many of the features of the MSP430 are not required for this application, a dedicated IC would provide improved performance overall and allow for volume production. This would also greatly reduce the cost of the design, likely to just pennies or dimes per chip, making a disposable bandage concept more feasible.

Hardware Documentation

Sensor Module

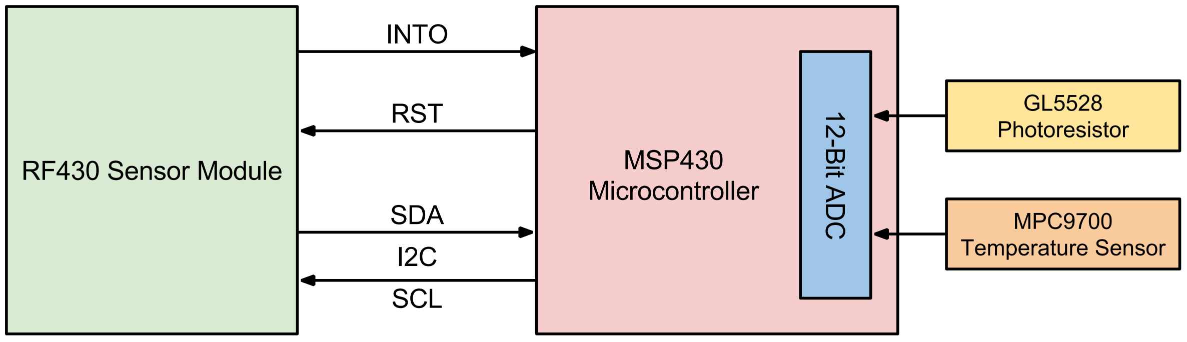


Reader Module

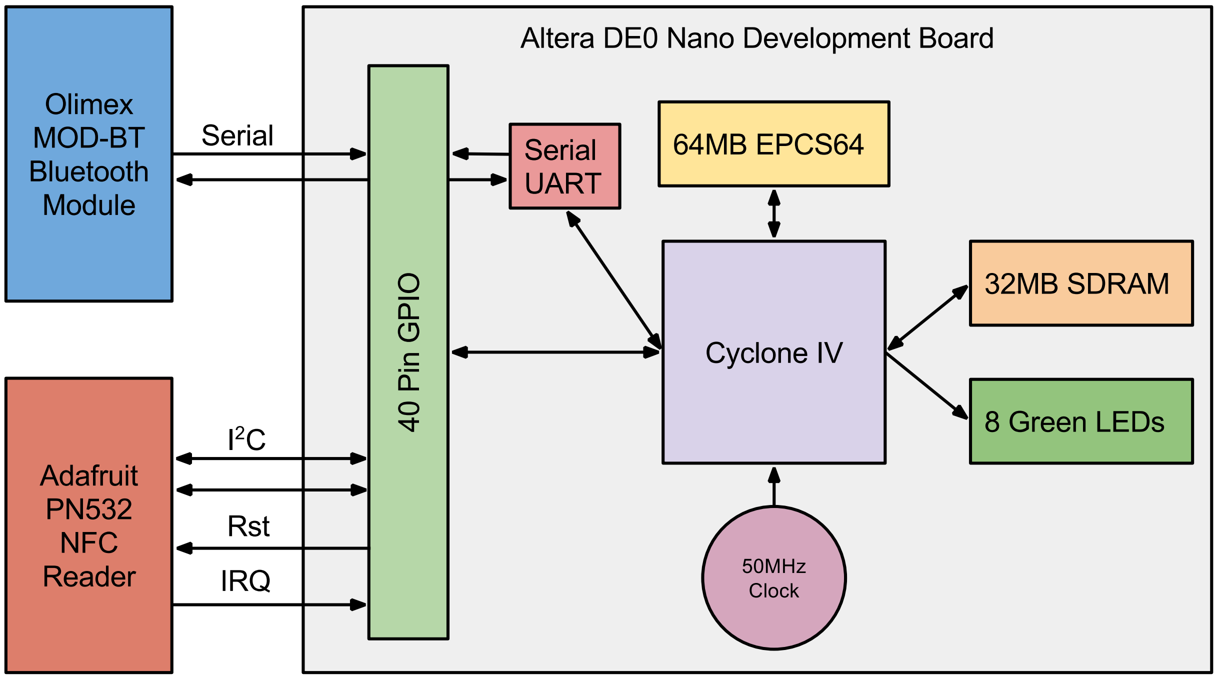


Block Diagrams

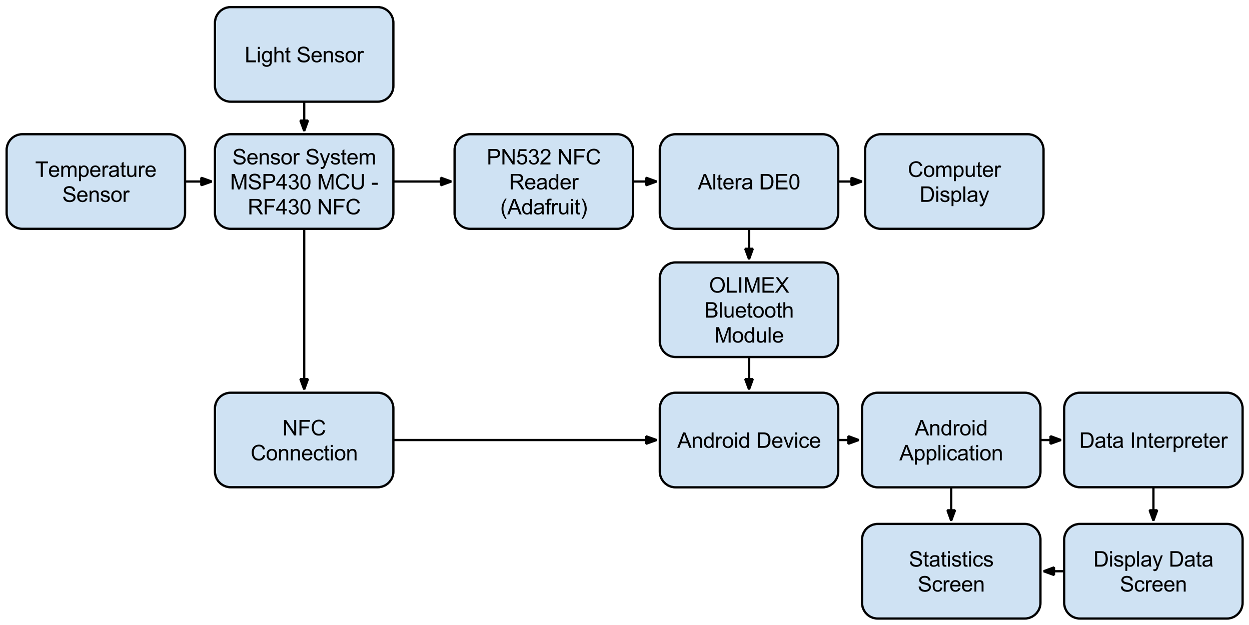
Sensor:



Reader:



Source Code



Source Code Index

File	Description	Status
Reader Module		
G2Capstone.vhdl	Top level file for the connections of FPGA	T
g2cap.c	Main MicroC/OS file with NFC and Bluetooth tasks	T
I2C.c	Provided I2C communication	T
I2C.h	Header file for provided I2C communication	T
btSerial.c	Library created for communicating with Bluetooth module	T
btSerial.h	Header file for above	T
pn532.c	Modified PN532 code for interacting with NFC reader	T
pn532.h	Header file for above	T
terasic_includes.h	Provided header file for I2C file and serial connection	T
Sensor Module		
main.c	Main task file with RF430 polling task. Initializes MSP430 and sensors.	T
RF430.c	RF430 read and write functions	T
RF430.h	RF430 constants and MSP430 pin headings for the RF430 connections	T
Sensors.c	ADC functions and sensor data conversions	T
Sensors.h	ADC constants and function declarations	T
Android Application		
Global.java	This holds any functions and variables that are commonly used around the applications	T
MainActivity.java	This is the activity that is run at launch, shows menu to navigate app	T

SerialBlue.java	This holds and deals with the bluetooth connection for the application, gest string and parses it	T
Graph.java	This is the activity for the graph, that creates the graphs and displays them	T
DataView.java	This displays the information for the selected date for measurements	T
History.java	This creates a list of dates to be selected to view data	T
NFC.java	This deals with the NFC connections, it opens the reading and gets the data to be parsed	T
Rading.java	Is the displayed class for getting the app ready to get the NFC connection	T
load.java	This loads the xml save file for history	T
save.java	This saves currently stored history from ram into xml file on memory	T
strings.xml	Holds data for strings used in other xml files that are constant	T
menu.bluetooth.xml	This the bluetooth menu selection	T
menu.data.xml	This the data view menu selection	T
menu.graph.xml	This the graph view menu selection	T
menu.history.xml	This the history selection menu selection	T
menu.main.xml	This the main screen menu selection	T
menu.simple.xml	For only when back to main menu selection is needed, used in the NFC and bluetooth strings	T
layout.activity_main.xml	Layout for the main screen	T
layout.bluetoothserial.xml	Layout for the bluetooth screen	T
layout.data.xml	Layout for displaying the data from selected history	T
layout.graph.xml	Layout for displaying the graph	T
layout.history.xml	Layout for creating the list for history to select	T
layout.nfc.xml	Layout for the nfc view	T