Seizure Tracker

By: Zachary Reardon

Advisor: Andrew Danowitz

Senior Project Final Report

California Polytechnic State University, San Luis Obispo

Computer Engineering

June 7, 2015

Table of Contents

7. Abstract

The goal of this project is to make a Module that enables anyone watching a person with disabilities to record any seizure that this person has under their care. Then when the Doctor/parent receives their patient/kid they are able to download all the seizures that their patient/kid had while they were away. The Doctor/parent then has the exact time, the duration of the seizure, and the type of seizure for all the seizures that have occurred while they were away from their patient/kid. From this data it is possible that the Doctor may see some sort of patterns that enables him to make better medical decisions for the patient/kid.

8. Introduction

This project came about due to my sister, Amy, who has daily seizures. The Doctors do not know what causes her seizures and her disability has not been clinically diagnosed. My parents want a way for Amy's Aide to document the seizures that she has while she is in the Aide's care. They are hoping to gain some type of indication of why her seizures are occurring from the data that the seizure tracker records at home and at school. The solution to this problem is broken into two parts. The first part of the solution is to make a Module that records seizures. A simple solution would be to make an Android application that records Amy's seizures, but she has no use for a phone. More importantly, my sister cannot protect her valuables, so using a Module that is only able to record seizures reduce the risk of the Module being lost. The second part is having some way to get the seizure information off of the Module. My parents have Android phones, and Android phones are Bluetooth capable. So, I built an Android application that gets the seizure information off of the Module via Bluetooth.

9. Design

This section covers what the different parts of the system look like as a whole, and what design constraints exist. This design section will specifically be covering the design considerations, and then exploring the design of the Module, then the Android application and lastly the networking between the Module and Android application.

9.1. Designing for the User

In order to create a product that will be successful, it is important to define who the user of each part of this project will be, and what is important from their point of view. From this consideration design constraints will become apparent.

9.1.1. Module's user

The person most likely to use this Module is the person who is watching the handicap person. Typically this is an Aide or Nurse. An Aide or Nurse typically watches multiple patients at the same time. The Module needs to be quick to activate so that the user (Aide or Nurse) is able to perform the medical procedures associated with the patient's seizure. From this comes the idea that the user cannot be fumbling around with the Module, it must be intuitive enough to pick up and use. The Module must also be able to run off of battery power for at least a week. This is required so that the Module's batteries do not need to be changed constantly, which would add more things that the Aide or Nurse has to do daily.

Once the seizure is done the user still has multiple other patients to take care of. So, the amount of screens they have to go through to finish recording a seizure must be minimal to enable the user to return to the other tasks they must do.

9.2. Android's user

The typical user of the Android application will be the person who cares about the data, a Doctor or Relative of the handicap person. We will be focusing on the Doctor because what he wants from the Module is exactly the same as the Relative, an answer. The Doctor's care will more likely be more personable than that of the Nurses, because he sees one patient at a time. So, this part of the application can be more complicated.

The Android application and Module must be reliable. It is unacceptable for the Doctor to be sitting for an extended amount of time trying to get the data off of the Module. Even more importantly is that the Doctor gets correct information off of the Module, because he will be using this information to make medical choice on what the patient should or should not do.

9.3. Design Constraints

After defining who the users will be it is the design constraints are easy to define.

- A device to record data on seizures.
- Android application to get the data off of the device.
- The device must be under 100 dollars.

- The entire interaction with any part of the UI for this project should take less than thirty seconds.
- The data collected by the device, transmitted, and received by the Android application must be accurate.
- Must be able to run off of battery power for a week.

9.4. Module Design

The following section defines what a User Interface looks like, and the components that the Module is made up of and why those components were selected.

9.4.1. UI Design

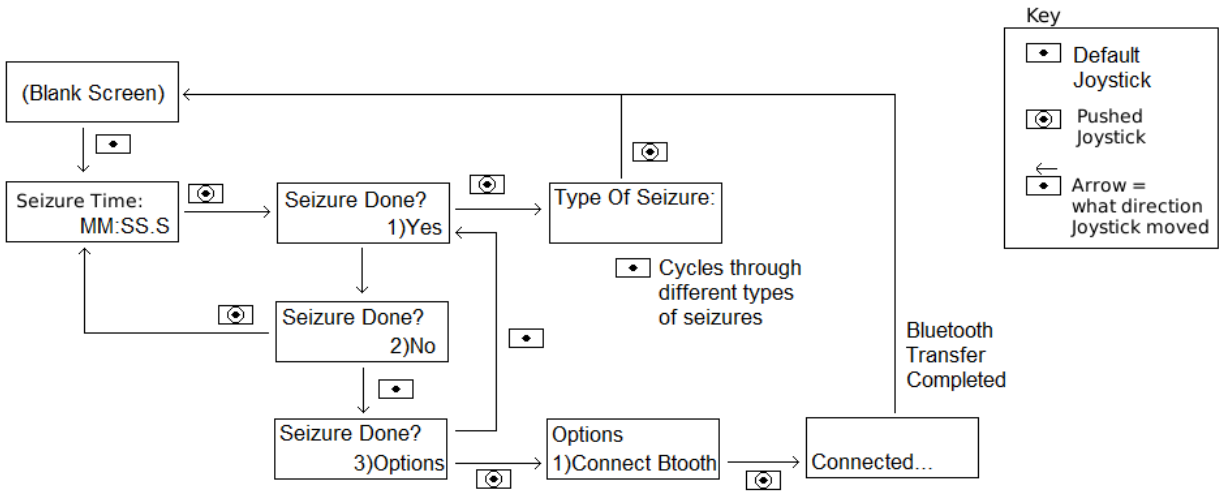

Figure 1: Module UI

The User Interface for the Module is quite simple as can be seen in Figure 1. It is designed so that the user is able to pick it up and use it instantly. The user's main focus should be the person having a seizure and not the Module.

Also, because the Module will mainly be used by Nurses/Aides we want to keep time it takes to use it to a minimum. That means that the number of screens to go through to record a seizure should be minimal. If the interaction depth is not kept to a minimum then the Aide may not want to use the Module and will avoid using it.

The user uses a Joystick to interact with the User Interface. A typical use of the Module starts off with the patient having a seizure. The user then grabs the Module and moves the Joystick in any direction to start the timer. The user then pushes down on the button, when the patient stops having the seizure, to bring up the confirmation screen. This screen ensures that

the user meant to stop the timer. If they did not mean to stop the timer and select “2)No”, the Module goes back to the timer screen with the timer updated for however long they were on the confirmation screen. If they meant to stop the timer and select “1)Yes”, the timer saves the amount of time it had when it went to the confirmation screen. The user then is able to cycle through the types of seizures by pressing the Joystick in any direction. Once the type of seizure is on the screen the user presses down to confirm the patient had that type of seizure. The Module then blanks out the screen and waits for the next time someone moves the Joystick.

When the user wants to get the seizure data off of the Module the user first moves the Joystick to start the time. The user then presses down on the button, causing the Module to display the confirmation screen. The user cycles through the confirmation screen by pressing the Joystick in any direction, until he gets to “3)Options”. The user then presses down the button to enable the Bluetooth chip to begin broadcasting, and after the Android application connects it automatically transfers the data.

For each seizure, the system records: duration of the seizure, time of day, and the type of the seizure. These three statistics are the basic statistics needed to analyze why a person is having seizures. More fields can be added at a later time, but because this is a prototype it is acceptable to stick to the basics. Also, this version of data collection enables the Doctor to get a complete picture and then begin to narrow down to the root cause of the patient’s seizures. For example, if the data comes out that the patient has most of their seizures midday, then the Doctor can decide to collect data asking questions about what the patient was doing midday. This customization is planned for later versions.

9.4.2. Module Architecture

Figure 2 shows how the Module is hooked up to its components that were chosen using decision matrixes. Decision matrixes are used to help make a decision on what component to use. A row contains one category (ex. Cost) and its associated weight and the components (ex. Arduino Uno) rated for this particular category. A ranking of ten means that this component is perfect for that particular category, while a 0 means it is the worst case scenario. The ranking is then multiplied by the weight and the column is added to produce a total. The component with the largest total is the component that should be selected.

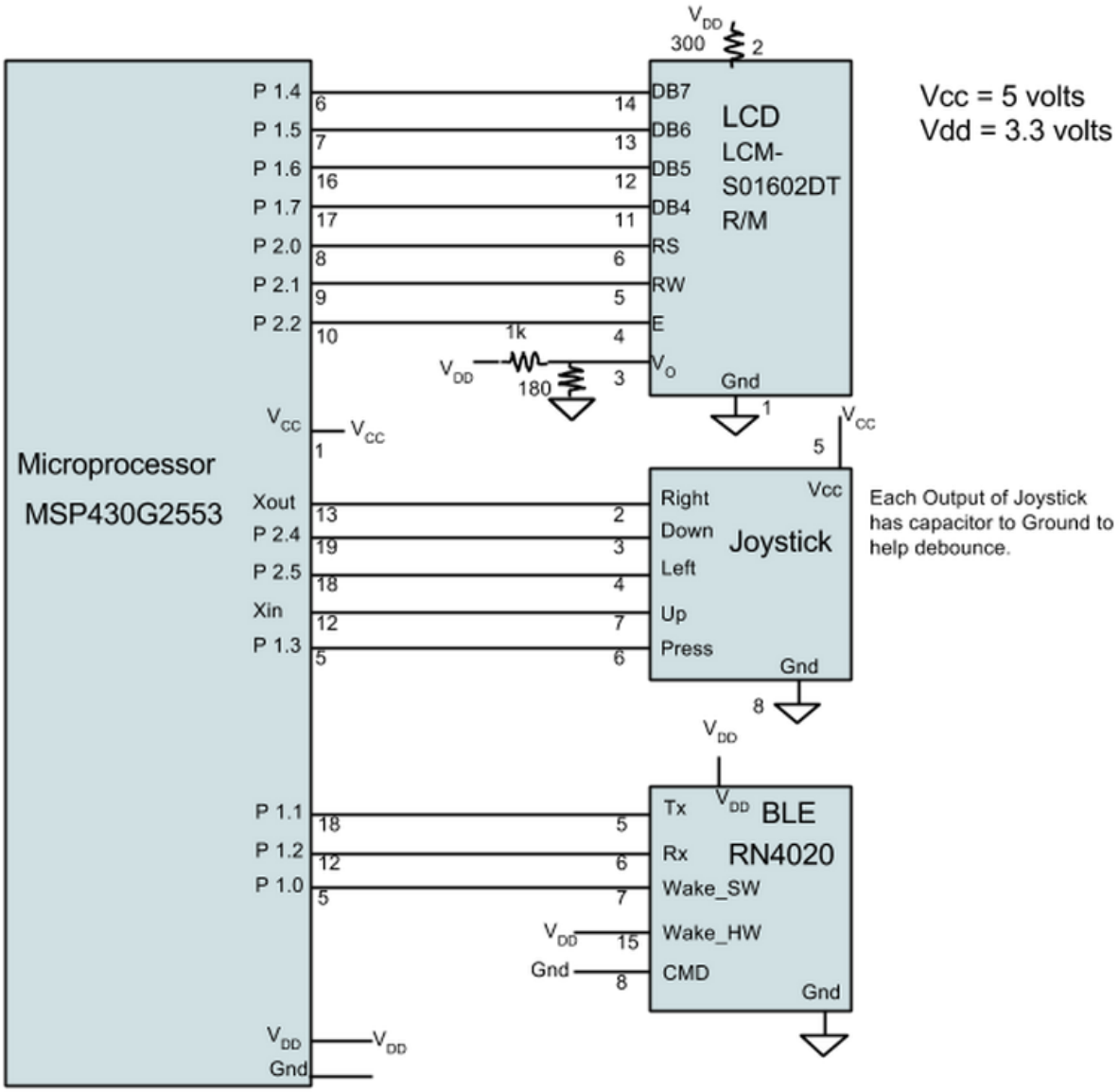

Figure 2: Module Architecture

9.4.2.1. Bluetooth Low Energy Component

A Bluetooth Low Energy component is needed in order to connect the Module to the Android phone.

I have never used Bluetooth before so I needed to use a Module that was reliable and easy to use. The RN4020 appeared to be the best option because it has the appearance of substantial documentation on how to make it work. It requires 3 pins. It uses UART to get the commands to make it do something. It also comes on a development board which enables

serial commands to be sent to it when the component is connected to a computer. This functionality enables the testing of the Module without going through the Android application.

9.4.2.2. Microprocessor

A microprocessor is needed in order to control the other components that make recording a seizure possible.

		MSP430G2553	Arduino Uno	Raspberry Pi
Reliability	.3	10	10	10
Experience w/ Component	.2	10	0	2
Cost	.15	10	10	1
Amount of pins	.1	7	6	10
Timer	.1	9	9	10
Power Consumption	.1	6	4	2
UART Enabled	.05	10	10	10
Total:		**9.2**	**6.9**	**6.25**

Table 1: Microprocessor Decision Matrix

Based off of Table 1 the MSP430G2553 is the best microprocessor to use. It has a max clock speed of 16 MHz, 512 bytes of RAM, and 16 KB of flash/ROM. It has two timers, two UARTs, and 16 GPIO pins. The main reason that this microprocessor was selected was because I have already had experience with it. With this experience comes the ability to program the microprocessor so that it is more reliable. It is also the cheapest option that comes with everything that is needed to complete the project. It is able to communicate to the RN4020 component through UART, keep track of the time, and has enough pins to interface with the other two components.

9.4.2.3. Display

Two options for displays were considered. The first being a simple LCD panel, and the second being a miniature touch screen. The LCD panel is simple, easy to interface with, and requires minimal wires when compared to a touchscreen. A touchscreen is not the optimal solution for our user. If our user has on nylon surgical gloves, mittens, or anything over their finger they cannot use a capacitive touch screen, meaning only resistive touchscreens can be used. A reliable resistive touch screen greatly increases the cost of the Module. Also, a touch screen requires a lot of computing power (floating points), needlessly raising the cost of the microprocessor needed to handle the touch screen.

The LCD panel selected for this project was the LCM-S01602DTR/M. This LCD is able to display 2 rows of 16 characters. This particular model was selected for its low cost. It is a simple LCD with four data wires and 3 wires that are toggled off and on to write characters to specific spots of memory, which cause the display to display that character on a specific spot on the screen.

9.4.2.4. Interaction Component

The interface the user uses to interact with the Module is by a Joystick. The Joystick can be moved up, down, right, left, pressed down or any combination of these motions. Originally, it was decided that rubber buttons, like those on a TV remote, were to be used. But, the rubber buttons that had the functionality and visual appeal needed were not sold in quantities that are acceptable for a prototype. While searching for the rubber buttons a Joystick made by Parallax was found. It is very easy to interface to because it converts the movement of the stick in any direction to a combination of DC outputs. It is visually appealing, and easy for the user to use.

9.5. Android Application

In order to get the data off of the Module a Bluetooth enabled device is needed to connect to the Module. The design constraints require this device to be an Android phone.

9.5.1. UI Design

Figure 3 shows the UI for the Android application.

The “Connect to Bluetooth” button, seen in Figure 3 below, enables the user to download the seizure info off of the Module. The user clicks on the button and then the Android application displays all Bluetooth devices in its vicinity. The user can then select the Module. Once selected, the application automatically connects to the Bluetooth device and begins to transferring the seizure data from the Module to the application. The data is then stored on the phone in CSV (Comma Separated Values) format in the Downloads folder. This format was selected because it is supported by most spreadsheet applications.

The CSV file has three columns: date, type of seizure, and duration of the seizure. The date column data format is week day, month, day, time, time zone, and year (for example Mon Jun 01 03:03:57 PDT 2015). The dates are based off of the time zone the phone is in when the data is downloaded off of the Module. The next column is a string of the type of seizure. The last column is the duration of the seizure in milliseconds.

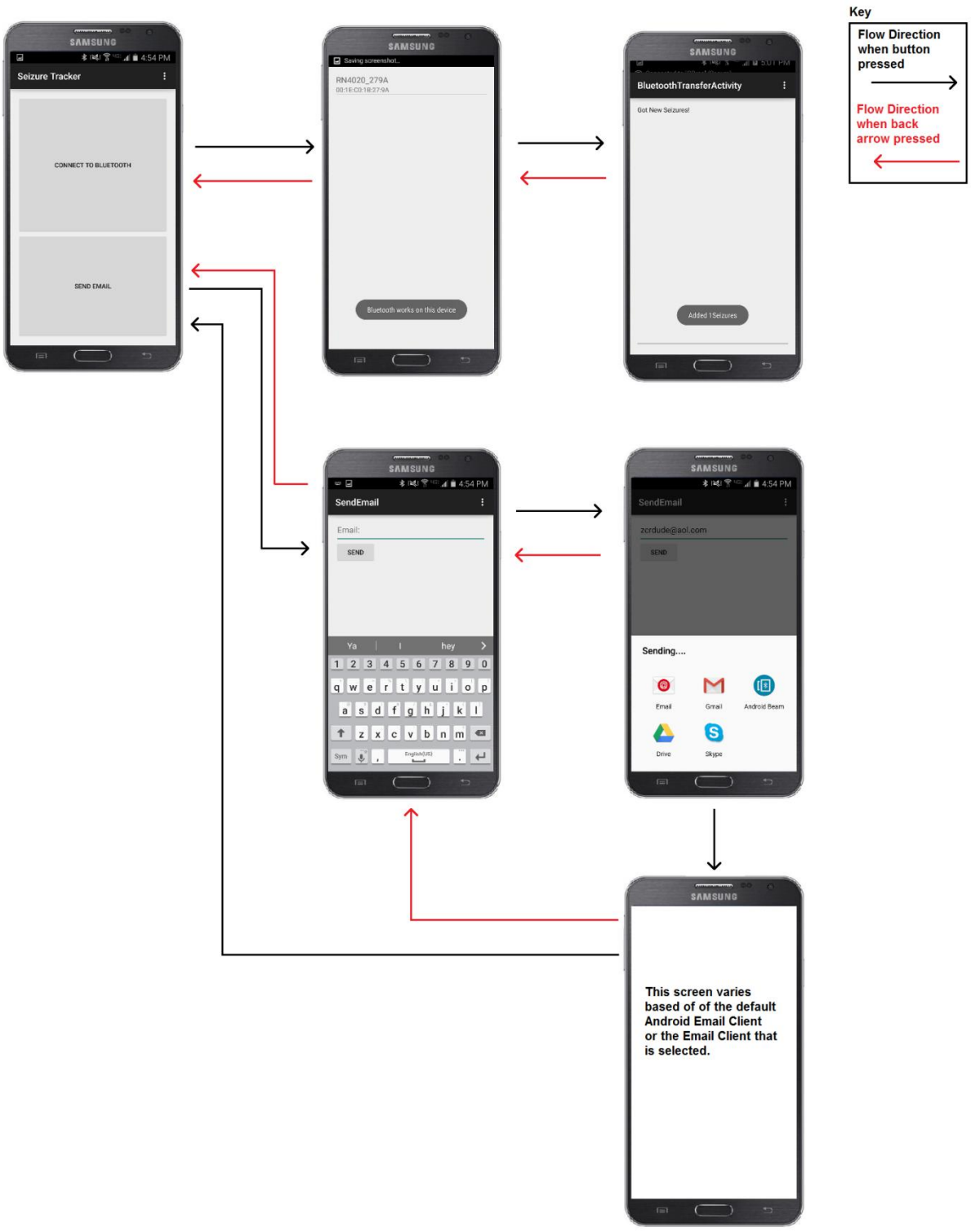

Figure 3: Android UI

The “Send Email” button enables the user to open their email client with the attachment with the CSV values in it. Unfortunately, I could not get the attachment to stick because I ran out of time and each email client appears to have a different way of attaching a file to it.

9.5.2. Code Implementation

The Android application requires API level 18 or Android 4.3, in order to access the libraries that use BLE.

Android provides a BLE library with GATT functionality. It is easier to program the Android application than that of the Module, but still required significant amount of time to go through the online library of functions for GATT and testing the functionality out through trial and error. All the read and write commands were asynchronous and there was no way to set them to be synchronous, which is what is required to complete the data transfer. Asynchronous means that the when trying to read a value, via a function call, the function returns without the value, but the application is still trying to read the value off of the Bluetooth component. Once the read is successful, the application pauses where it is in the program and handles the value that was read. For data transfer, the value is needed to be returned before the Android application can do anything else. To solve this problem the application spin waits while waiting for a value in an int to change, signifying that the value requested to be read has been read. The application cannot read all the fields and then go from there because there is a common bug where the Android application appears to forget to read all of the fields requested.

9.6. Networking Design

The section will begin by going over the microprocessor to BLE connection and end with how the seizure data is transferred from the Module to the Android phone.

9.6.1. Microprocessor to BLE

The microprocessor and RN4020 communicate over UART at 115200 BAUD; the UART uses 8 data bits, 1 stop bit, and has no parity bit.

The RN4020 uses Generic Attribute Profile (GATT). GATT is built on Bluetooth Low Energy (BLE) fundamentals. GATT enables the RN4020 to communicate to other devices. In GATT every value that needs to be sent across the Bluetooth connection must have its own characteristic. Each characteristic requires its own service in order to exist in GATT.

GATT sets up services each one having their own characteristics. Each characteristic has its own value, and each characteristic has its own permissions that are used to determine if a connected device can read, write, notify, or indicate to that characteristic. Notify and indicate send the new value to the connected device each time that characteristic is changed. Indicate differs from notify by requiring the connected device to authenticate in a particular way to the

notification. For this project notify is used because there is no need for a special authentication feature. Each private characteristic (covered later) has an associated 128 bit Universally Unique Identifier (UUID) that is a unique characteristic different from all other characteristics of any service on the Bluetooth component.

The RN4020 has specific commands that change the characteristics' values, set up service-characteristic relations, and a lot of other things. Yet it does not have the ability that most GATT devices have, it has no command to specify what type of data is in a particular characteristic. It permanently sets all characteristic values to be ASCII-Hex. This means that, all values have to be converted into Hex and then converted into its ASCII equivalent. So, in order to set a characteristic to the number 10, the first step is to convert that number to its hex equivalent, 0xA (1010), and then convert that number to ASCII which would make it become 0x41 (01000001) and then send it to the RN4020.

This process would be easy on most fully fleshed out Operating Systems, but with a microcontroller, code must be written that converts a value to its hex equivalent and converts that hex to its ASCII equivalent.

9.6.2. Microprocessor to Android Application

The communication between the microprocessor and the Android application that goes through the RN4020 is displayed in Figure 4 below. GATT has predefined services, called public services, which are listed online. None of the public services listed fit with the values that were to be transferred, so the need to set up a new service and characteristics occurred. A service that is not a defined public service is called a private service, and its characteristics are called private characteristics. Through serial connection to the RN4020, 1 private service with 4 private characteristics were set up. The service has its own 128 bit UUID and each characteristic has its own 128 bit UUID. The Time's characteristic takes up two characteristic slots because notifications were enabled on the characteristic. With notifications on for the Time characteristic the RN4020 tells the device that is connected to it when the Time characteristic is updated.

The initial setup for the data transfer is straight forward. The microprocessor tells the Bluetooth Module to begin broadcasting for a connection. The Android application then connects to the RN4020 which tells the microprocessor that a device has connected to it.

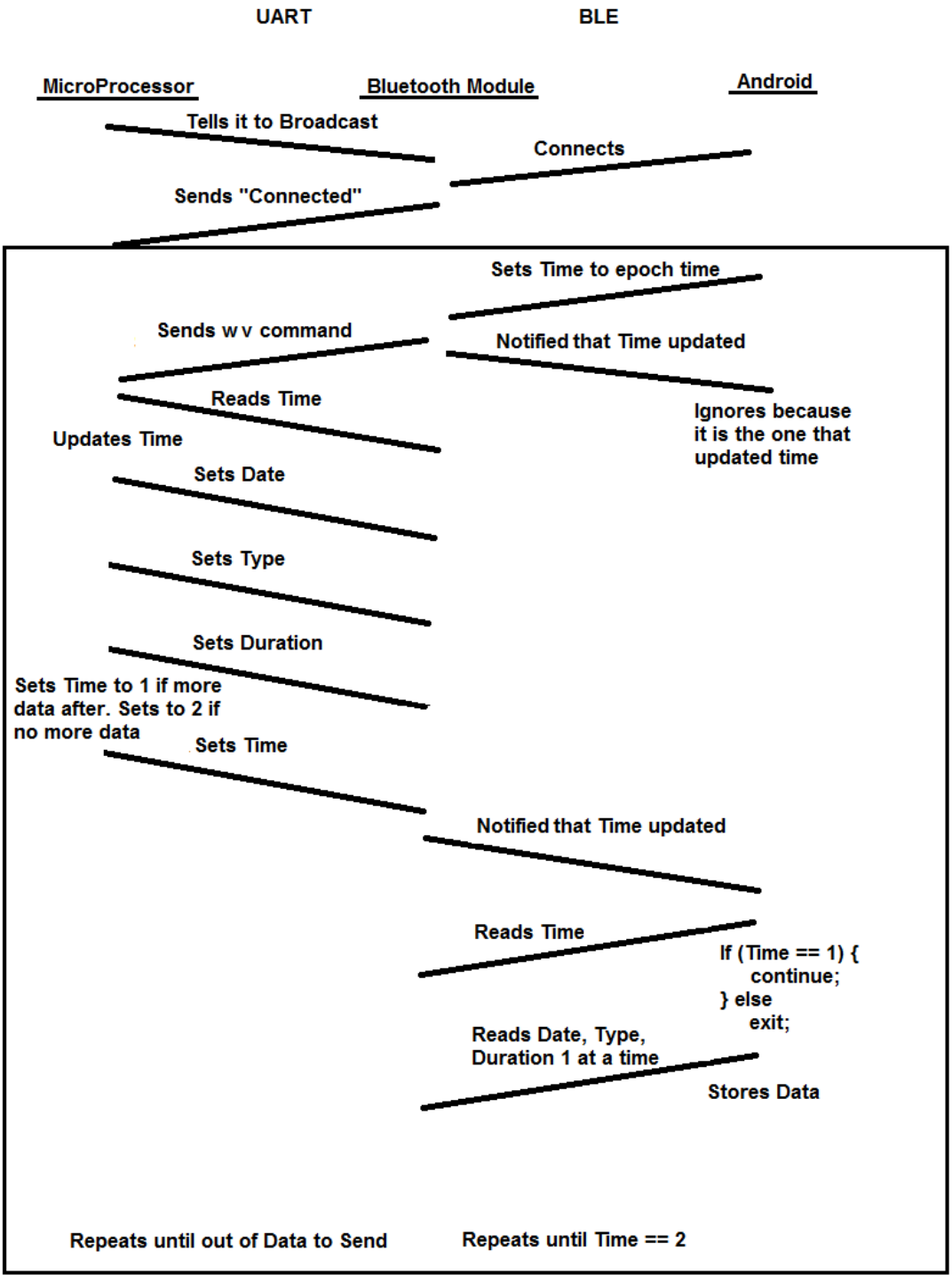

Figure 4: Network Flow

Then begins the data transfer stage. The Android application sets the Time characteristic to the true time in milliseconds since epoch. The time is placed in the characteristic so that the microprocessor can update its time field to remove any skewing of time due to its cheaper

clock, and it also lets the microprocessor know that the Android application is ready to receive data. The microprocessor then updates the Date, Type, and Duration characteristic for the first seizure it has stored. It then updates the Time characteristic to 1 if the data in the three characteristics are valid, or 2 if there was no more data to write, and the data is incorrect. The Android application is notified of the change to the Time characteristic. If the Android application reads that the Time value as 2 it stops. If the value is 1, it reads the values, and then sets the time field to the true time in milliseconds since epoch and repeats this process.

10. Bill of Material

Component Name	Component Type	Manufacture	Price
MSP430G2553 Launchpad	Microprocessor	TI	$9.99
LCM-S01602DTR/M	LCD	LUMEX	$8.38
RN4020 w/ PICTAIL Plus	Bluetooth Module	Microchip	$49.99
5-Position Switch	Joystick	Parallax	$9.99
Various Resistors and Capacitors	Resistors/Capacitors	N/A	$1.00
		Total:	**$79.35**

Table 2: BOM for Prototype

The cost of the prototype displayed in Table 2 is more than expected. The goal for the price of the Module was to be around 50 dollars. By removing the cost of the development boards used, a more agreeable cost for a final product, as seen in Table 3 below, is achieved. Also, with the production of more Modules the price per component is greatly reduced.

Component Name	Component Type	Manufacture	Price
MSP430G2553	Microprocessor	TI	$2.80
LCM-S01602DTR/M	LCD	LUMEX	$8.38
RN4020	Bluetooth Module	Microchip	$8.84
5-Position Switch	Joystick	Parallax	$9.99
Various Resistors and Capacitors	Resistors/Capacitors	N/A	$1.00
PCB board			~$15.00
3-D printed Case			~$5.00
3.3 Voltage Regulator			$1.95
		Total:	**$52.96**

Table 3: Estimate for Manufactured Module

11. Conclusion

This project was very challenging. The project currently has a lot of bugs, and does not provide a reliable connection from the Android application to the BLE component. I have never worked with Bluetooth before, so I had to learn about Bluetooth, GATT, Bluetooth programming on Android, and the RN4020 UART commands. Additionally the RN4020 tended to dump its memory, causing it to forgetting its name, and how to connect to the components. This also causes it to forget its private characteristics, requiring me to connect it to my computer and reprogram it. Each dump resulted in a loss of time; time used to debug the debug, and fix the issue.

In order to improve the reliability of the Bluetooth, the RN4020 needs to be replaced. I will try to find a component that is more robust and is able to set characteristics using any encoding.

In the future the plan is to replace the RN4020, and fix the major bugs, such as the inability to send emails in the Android application. Also, it would be nice to enable the Module to go back screens. For example, if the user accidently clicks on the Bluetooth they can go back to the confirmation screen.

Besides that I enjoyed this project because I got to do thing that I have never done before and learned a lot about Bluetooth Low Energy and GATT.

12. References

MSP430 Family Users Guide
http://www.ti.com/lit/ug/slau144j/slau144j.pdf

PmodCLP™ Parallel LCD Display Module Reference Manual
http://www.digilentinc.com/Data/Products/PMOD-CLP/PmodCLP_rm_RevA.pdf

16COM / 40SEG DRIVER & CONTROLLER FOR DOT MATRIX LCD
http://www.mrc.uidaho.edu/mrc/people/jff/340/341/ECE%20341%20Reference%20Materials/Digilent%20Cerebot%2032MX7ck/Digilent%20Pmod%20Boards/SamsungKS0066U.pdf

RN4020 Command Sheet
http://ww1.microchip.com/downloads/en/DeviceDoc/70005191B.pdf

RN4020 Datasheet
http://ww1.microchip.com/downloads/en/DeviceDoc/50002279A.pdf

Android API
https://developer.android.com/reference/android/package-summary.html