Milestone 3

Ian Brown, Jordan Gaeta, Gleb Alexeev

March 31, 2016

1 Introduction

In this milestone, we had the following task:

- Develop firmware to upload counter data from trail counter

We wanted to develop this firmware while understanding how it would be used in the field. That is:

1. A user comes within range of the trail counter.
2. The user connects to the counter via Bluetooth.
3. Counter data is transmitted to the phone and saved.
4. The counter disconnects and returns to its normal state after transmission.

This process assumes that the counter is occasionally powering on and advertising its Bluetooth channel to end users. If no connection is found, it returns back to its normal state. In order to accomplish all of these tasks, we needed to complete the following:

- Develop a method to check if the board held a connection with another device
- Develop a method to advertise the device's Bluetooth channel on some interval
- Allowing a packet larger than 256 bytes to be sent across Bluetooth
- Develop a method to disconnect once the data has been transmitted

2 An Aside About IRQ Pins

In order to check if the board was connected to another device, we initially sought to check the IRQ pin on the Bluetooth device. In the documentation for the device, it is explained that the IRQ pin is raised when data is available on the chip of the device. In our testing, we noticed quickly that this was not

the case. When our iPhone was connected to the device and sent a message across, the IRQ pin stayed low during the transaction. In fact, none of the pins fluctuated when the message was sent. Only when we sent the device an AT command to retrieve the message did we notice the IRQ pin set high. However, in previous transactions with the device, we noticed that the IRQ pin was set high whenever the CS pin was set low. This meant we were receiving false-positives about the nature of the IRQ pin, and needed to approach detecting connections differently.

3 Detecting a Connection

After abandoning the IRQ approach, we were fortunate to learn that there is an AT command that checks if the Bluetooth device currently holds a connection. Thanks to our previous milestones and the General Access Profile Bluetooth commands, we were able to transmit 'AT+BLEGAPGETCONN' to the device. The return values are as follows:

- 0: The board is currently not connected to any device
- 1: The board is currently connected to a device

This return value was the first element of a return payload sent back by the device. Because of our previous understanding of SDEP, we were able to extract this value and use it as a flag.

4 Advertising on an Interval

In order to simulate the trail counter's intermittent run state, we wanted to develop an advertising interval for the Bluetooth device. This meant the device would be visible to other devices for a short period of time before disappearing. Similar to the process of detecting a connection, there is an AT command that starts and stops advertising on the Bluetooth device. Thanks to our previous milestones, we were able to transmit 'AT+GAPSTARTADV' and 'AT+GAPSTOPADV' to the device in order to start and stop advertising, respectively. Creating a thread that slept between the transmission of these two commands would trivially simulate an advertising interval.

5 Extremely Large Payloads (>256 bytes)

According to SDEP, the maximum packet size that can be transmitted is 256 bytes. This is because the buffer sent across Bluetooth only contain bytes, and there must be a byte dedicated to the size of the message payload. Payloads that are bigger than 256 bytes must sent separately. In addition, only 20 bytes may be sent at a time, with subsequent sends including a 4 byte header. This means for each 256 byte packet, 52 bytes must be header information, leaving only

204 bytes for the message payload. Finally, our milestone requires the board to send messages to a connected device. This means using the AT command 'AT+BLEUARTTX=', a 13 character command. Overall, this leaves only 191 bytes for a raw message. In order to handle this caveat, we wrote a wrapper that takes the following:

- A pointer to a data buffer
- The size of a data buffer

This wrapper then creates the command to send parts of the message 191 bytes at a time.

6 Disconnecting the Device

Following the transmission of data, we want to quickly disconnect the board from a connected device in order to conserve as much power as possible. Fortunately, there is an AT command that disconnects the Bluetooth device from any connected device. Thanks to our previous milestones, we were able to transmit 'AT+GAPDISCONNECT' to the device and disconnect from our iPhone. We include this transmission following our large payload transmission.

7 Problem Areas

During the completion of this milestone, we ran into problems during implementation that did not directly effect our work, but will effect the trail counter's use in the field. These problems are listed as follows:

- The thread that handles the advertising interval, checking for a connection, transmitting large payloads, and disconnecting requires a larger amount of memory allocated in order to work. The standard 128 bytes crashes the entire system.
- Packets larger than 4096 bytes are not being sent properly. The system crashes near the beginning of transmission.
- Transmissions could be faster. This could be fixed with adjustments to sleep intervals in our write methods.

We will continue to address these issues.