Jonathan Procter

CDC Computerized Cognitive Testing Platform

Bluetooth Communication for Embedded Systems

Introduction

Tiny computers exist all around us inside the devices we use every day. Whether it's the 8-bit processor that programs the spin speed of your toothbrush, or the microcontroller that allows you to chose items in a vending machine, these embedded systems require careful design. Often you will want to communicate wirelessly to a system you have created to control it and transfer data. Bluetooth communication is a popular method to accomplish this, as it does not require an existing network like Wifi does. This technical review briefly summarizes some of the options available on the market for Bluetooth communication in embedded systems, describes the functionality of Bluetooth technology, and explains how one might implement it on a given system.

Commercial Options of Embedded Bluetooth Devices

Embedded devices are typically designed with low cost and low power in mind, therefore the specific protocol of Bluetooth Low Energy, or BLE, is required. BLE is similar to regular Bluetooth but has a lower 1Mbps bandwidth unlike Bluetooth's 3Mbps, meaning you cannot stream audio on it, but you can still transfer smaller data like that from multiple sensors [1]. All the device options reviewed in this section will be BLE devices with low cost and low power draw.

A leading device for Bluetooth communication is the Adafruit Bluefruit LE UART Friend. For $17.50, the Bluefruit offers a small 21mm x 32mm x 5mm chip with serial pin connection [2]. The chip takes an average current of 1.44 mA, and peak current of 13.5 mA [2]. A special bonus to the Bluefruit is a free Android/iOS phone application with premade buttons, sliders, and controls for use with your device that can make it simple to set up functionality without need for much code. Similar to the Bluefruit, the $15.75 Cypress CY5676-ND is a serial connected chip sizing in at 45mm x 27mm x 2 mm [3]. Cypress' solution uses an average current 1.89 mA, and peak current of 15.6 mA [3]. These two options are both very small chips that can be wired straight into any device's circuit, allowing them to be flexible to the design of the embedded system.

A less common method of enabling Bluetooth on a device is through a dongle such as the Bluegiga BLED112. The BLED112 is a $10.45 USB dongle with dimensions of 17mm x 12mm x 6.5mm [4]. Unlike other options in this section, the BLED112 cannot be wired directly into a device's circuit and instead a USB port must be hooked up via a breakout board for the dongle, increasing cost and size. The

current draw of the BLED112 varies greatly depending on the breakout board used for the USB port. Another option for embedded systems is to use a microcontroller or internal system that already has Bluetooth functionality on it, such as the Raspberry Pi Zero W. At $10.00, the Pi Zero W's massive 66mm x 30.5mm x 5.0mm size stores more than just a Bluetooth chip, including a single-core CPU, 512MB RAM, mini HDMI and USB ports, microUSB power adapter, and a Wifi card [5]. It also draws 80mA of current on average, much higher than any other option in this section [5]. The biggest disadvantage of using the Pi Zero W is that you cannot add it to an existing system for Bluetooth functionality. Instead, you must use the Pi Zero W as your main computing unit. However, if the Pi Zero W has everything you need for your device already, then this option is simply a cheap bonus.

Bluetooth Technology

Bluetooth communication is a protocol that allows data to be transferred in the frequency range of 2.4GHz. Networks are created in this frequency range called piconets that have a master and any number of slaves that the master device can send and request data from [6]. The devices summarized in the previous section are considered the slave devices in these piconets, and the master devices would be a laptop or a phone or any device used to control the embedded system. On the slave devices, there is a unique 48-bit Bluetooth address to identify it on the network [6]. A master device typically will see a list of all Bluetooth slave devices within range. Upon selecting one, a request is sent to the slave device for its Bluetooth address and then the master and slave pair is formed [6]. The slave devices reviewed in the previous section will then use Serial Port Profile interface in order to receive and send data with the master device [6]. This profile allows for easy translation of data into the serial connection the Bluetooth device is wired in on the embedded system's circuit.

Bluetooth Low Energy has come into play in recent years to provide embedded systems with smaller, lower cost, and lower power options. The main method of creating BLE versus regular Bluetooth is having more sleep states in the cycles of the chips. By keeping the active state cycle current draw the same, but lowering the current draw from sleep states, BLE can vastly decrease power consumption while keeping connectivity consistent, at the sacrifice of lower bandwidth [7].

Bluetooth Implementation on Embedded Systems

After choosing a Bluetooth device to use in your system, the remaining steps are to combine it with your existing hardware and configure any necessary software to enable it. In the case of the two chips reviewed in the earlier section, you must connect the pins from the chip to your microcontroller or microprocessor of your choice, either on a breadboard or through a soldered board. Once the serial

connection is made, you must also power the device either from an external power source, or from the power coming out of your microcontroller or system.

The software stack to use depends on what microcontroller or microprocessor you choose for your system. A microcontroller like an mbed or Arduino would likely have Serial libraries in C/C++ for you to use to receive the strings of data from the Bluetooth device. Once you are receiving the data strings properly, you can then code the appropriate tasks on your device depending on what data is being received. How often you check the Bluetooth device for new data, or the baud rate, is something else to consider depending on the speed of the processor in use. Most Serial libraries allow you to set the baud rate of your connection. A higher baud rate will provide a higher response time from master to slave, at the cost of potentially hogging hardware resources like the I/O and memory controllers in your system. After the Bluetooth device is wired up and the code is written, you can then connect the master device to your embedded system and begin testing.

[1] Argenox, “A Guide to Selecting a Bluetooth Chipset,” 2019. [Online]. Available: https://www.argenox.com/library/bluetooth-low-energy/a-guide-to-selecting-a-bluetooth-chipset/. [Accessed: March 3, 2019].

[2] Adafruit, Adafruit Bluefruit LE UART Friend - Bluetooth Low Energy (BLE) Product Information. 2019. [Online]. Available: https://www.adafruit.com/product/2479. [Accessed: March 3, 2019].

[3] Cypress Semiconductors, “CY5676 PRoC BLE 256 KB Module,” Aug. 31, 2015. [Online]. Available: https://www.cypress.com/file/177106/download. [Accessed: March 3, 2019].

[4] Silicon Labs, “Bluegiga BLED112 Bluetooth Low Energy Dongle,” N.D. [Online]. Available: https://www.silabs.com/products/wireless/bluetooth/bluetooth-low-energy-modules/bled112-bluetooth-smart-dongle. [Accessed: March 3, 2019].

[5] Raspberry Pi Foundation, Raspberry Pi Zero W Product Page. N.D. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-zero-w/. [Accessed: March 3, 2019].

[6] Sparkfun, “Bluetooth Basics,” 2017. [Online]. Available: https://learn.sparkfun.com/tutorials/bluetooth-basics/all. [Accessed: March 3, 2019].

[7] Electronic Design, “Designing Next Gen Bluetooth Low Energy System for the Internet of Things,” Sep. 2, 2015. [Online]. Available: https://media.digikey.com/pdf/Application%20Notes/Cypress%20Application%20Notes/Next_Gen_Bluetooth_ExecSumm.pdf. [Accessed: March 3, 2019].