TinyBlue: A Bluetooth LE Module for TinyOS

Thomas Bauer
Stanford University
Department of Electrical Engineering
Stanford, USA
tbauer01@stanford.edu

Dave Deriso
Stanford University
Institute for Computational and Mathematical Engineering
Stanford, USA
dderiso@stanford.edu

***Abstract*— Recent advances in Bluetooth low energy (BLE) technology have enabled long term wireless connectivity with extremely efficient power management. This technology has sparked a fast growing trend in small connected sensor-based consumer electronics based on BLE. Most of these devices use the same basic architecture: a sensor, a microcontroller, and a BLE transceiver. Despite the widespread use of this architecture, little research has been released on optimizing the software system to complement the power management of the BLE stack. The present work demonstrates that the integration of Bluetooth low energy and TinyOS, a power efficient operating system for distributed sensing, leads to a very power efficient distributed communication platform.**

Keywords—TinyOS; Bluetooth; Low Energy; Rivendale; BLE

I. INTRODUCTION

Bluetooth Low Energy (BLE), which was introduced as part of the Bluetooth 4.0 specification, is an exciting wireless technology that gives mobile application developers unprecedented access to external hardware and provides hardware engineers with easy and reliable access to their devices from every major mobile operating system. Recent advances in low energy bluetooth (BLE) technology have enabled long term wireless connectivity with extremely efficient power management. This technology has sparked a fast growing trend in small connected sensor-based consumer electronics based on BLE. Most of these devices use the same basic architecture: a sensor, a microcontroller, and a BLE transceiver. Despite the widespread use of this architecture, little research has been released on optimizing the software system to complement the power management of the BLE stack. The present work demonstrates that the integration of Bluetooth low energy and TinyOS, a power efficient operating system for distributed sensing, leads to a very power efficient distributed communication platform.

II. A BRIEF HISTORY OF LOW ENERGY RADIO

A. Origins

Norman Abramson, a professor at the University of Hawaii, developed the world's first wireless computer communication network, ALOHAnet in 1971. The system was based on low-cost Ham radios, and was deployed on seven computers spread across four islands to communicate with the central computer on the Oahu Island without using phone lines. Since then, wireless networking has become smaller, more efficient, and ubiqutous in modern electornics.

B. Wi-Fi

In 1991, AT&T invented WaveLAN, the precursor to Wi-Fi, which was intended for use in cashier systems. In 1997, the IEEE developed the 802.11-1997 standards for wireless networks and later branded it as "Wi-Fi," since it sounded better than "IEEE 802.11b Direct Sequence." The original specification operated in the 2.4GHz spectrum and provided for 1-2 Mbit/sec, but was later upgraded to 11 Mbit/sec in 1999. The latest standard, 802.11af, was approved in 2014 , occupies 5 channels, and achieves a data rate of 426.7 Mbit/s for 6 and 7 MHz channels and 568.9 Mbit/s for 8 MHz channels. The 802.11ax wifi standard is in the process of proposal by Huawei Corporation, which at the time of writing has demonstrated 10.53 Gbit/sec on the 5 GHz band. Although Wi-Fi is a very efficient wireless technology, it is optimized for large data transfer using high-speed throughput and is not really suitable for coin cell operation. A Wi-Fi device consumes approximately 116 mA at 1.8 V (116 mA x 1.8 V = 0.210 W) when transmitting a 40 Mbps User Datagram Protocol (UDP) payload, achieveing a power per bit of 0.210/40,000,000 = 0.00525 μW/bit. Unfortunately, current consumption does not reduce when throughput is reduced in a Wi-Fi chipset. Most new power-saving specifications are still being written, or are not in mass production yet, thereby lengthening the time-to-market of such advancements.

C. Zigbee

ZigBee® was established in 2003 by a group of 16 companies as a low-power wireless specification based on the IEEE Standard 802.15.4-2003. It was designed for mesh networking deployable sensors, such as smart meters, home automation, and remote control units. A Zigbee device consumes 0.035706 W when transferring 24 bytes of data (192 bits). Assuming 192 bits/sec are transferred, the power per bit = 0.035706/192 = 185.9 μW/bit. In an open environment Zigbee radios have about 100m range, however ZigBee radios do not hop frequencies and are susceptible to interference making them a tough choice for mesh networks.

D. Near-Field Communication

Near-field communication (NFC) was established in 2004 by the NFC Forum, founded by Nokia, Sony, and Philips, which specified the ISO 13157 standard as an extension of radio-frequency identification (RFID) that enables with two-way interactions and limits communication range for security purposes. NFC is very different from other low-power wireless technologies. It works up to a range of approximately 5 cm and consumes relatively more power. Passive NFC tags can be completely unpowered, only becoming active when an NFC field is present, which eliminates NFC from many of the use cases discussed here. NFC operates at 13.56 MHz on ISO/IEC 18000-3 air interface and at rates ranging from 106 kbit/s to 424 kbit/s. NFC consumes approximately 15mA at 1.8 V (15 mA x 1.8 V = 0.027 W) when reading, which achieves a power per bit = .027/424 = 0.064 μW/bit.

E. ANT

ANT is a low-power proprietary wireless technology which operates in the 2.4 GHz spectrum. It was established in 2004 by the sensor company Dynastream, and for some reason there is no explanation for what ANT stands for. Typically, the ANT transceiver device is treated as a black box and shouldn't require much design effort to implement into a network. Its primary goal is to allow sports and fitness sensors to communicate with a display unit, for example a watch or cycle computer. It also typically operates from a coin cell. ANT+ has taken the ANT protocol and made the devices interoperable in a managed network, thereby guaranteeing that all ANT+ branded devices work seamlessly. ANT devices may operate for years on a coin cell. ANT devices are not subject to the extensive conformance and interoperability testing applied to other standardized technologies. ANT+ is introducing a new certification process in 2011 which will be chargeable and a prerequisite for using ANT+ branding. An ANT device is configured to transmit 256 bits/second and consumes 61 μA (3 V x 61 μA = 0.183 mW), thus achieving a power per bit = 0.183 mW / 256 bits = 0.71 μW/bit.

TABLE II
WIRELESS COMMUNICATION STANDARDS

	Range (typical)	Data Rate (max)	Power Cons.*	Cost per chip	Frequency
Zigbee	10-75m	20kbps/ 40kbps/ 250kbps	30mW	$2	868MHz/ 915MHz/ 2.4GHz
Bluetooth	10-100m	1-3Mbps	2.5-100mW	$3	2.4GHz
IrDA	1m	16Mbps	-	$2	Infrared
MICS	2m	500kbps	25μW	-	402-405 MHz
802.11g	200m	54Mbps	1W	$9	2.4GHz

*Power consumption refers to maximum consumed power when the chip is on and it is sending/receiving.

Bluetooth® low energy started in 2006 as a project in the Nokia Research Centre with the name Wibree, and was renamed Bluetooth Ultra-Low-Power and then Bluetooth low energy. The aim of this technology is to enable power sensitive devices to be permanently connected to the Internet while only being powered by a coin cell battery. Bluetooth Low Energy is governed by the Bluetooth v4.0 specification. A device that operates Bluetooth v4.0 may not necessarily implement other versions of Bluetooth; in such cases it is known as a single-mode device. Most new Bluetooth chipsets from leading Bluetooth silicon manufacturers are dual devices that support Bluetooth and the new LE functionality. Connectable advertising packets (adverts) are broadcast every 500 ms. Each packet has 20 bytes of useful payload and consumes 49 μA at 3 V. For this particular setup, adverts are spread across all three channels, with the positive side effect of increasing robustness over a single-channel technology. The typical power consumtion is 49 μA x 3 V = 0.147 mW and bytes per second = 20 x (1 second/500 ms) x 3 channels = 120 bytes/second, achieving a power per bit = 0.147 mW/(120 bytes/second x 8) = 0.153 μW/bit. It should be noted that this configuration uses connectable packets. Therefore, the advertising device is also scanning after each advert. This consumes significant power, but is still lower than its nearest competitor. By increasing the payload to 31 bytes per packet and configuring for broadcast only, power per bit efficiency would be improved further. This would occur due to the increase in protocol efficiency from 20 payload bytes to 31 for the same overhead.

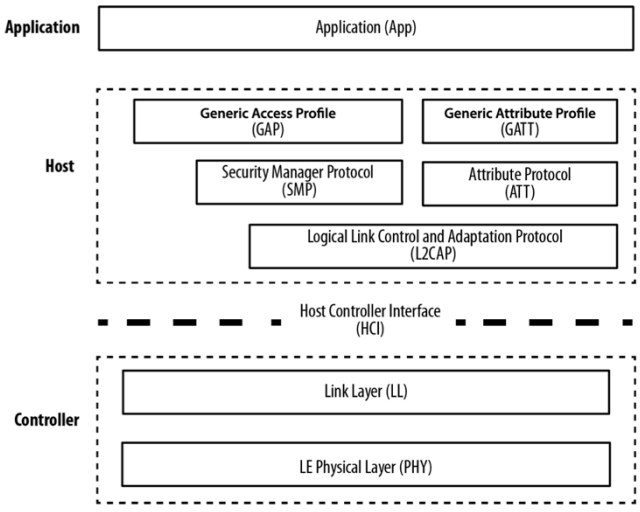
III. Bluetooth Low Energy

A. Bluetooth Low Energy vs Classic Bluetooth

Bluetooth Low Energy (BLE, also marketed as Bluetooth Smart) is the centerpiece of the Bluetooth 4.0 Core Specification. While BLE has many similarities with its predecessor, it departs from classic Bluetooth in many ways. The on-air protocol, the upper protocol layers, and the applications are different and incompatible between the two technologies. While Bluetooth classic focused on a strict set of use cases like audio and data transfer, BLE was designed for simplicity and extensibility, so that a developer could interface with any accessory without having to know a great deal about the underlying technology. In addition, BLE was designed to have the lowest possible power consumption, optimized for low cost, low bandwidth, low power, and low complexity and designed to easily exchange data. BLE also has an inherent resource requirement assymetry where masters, such as smartphones and tablets, require more resources, while slaves, such as beacons, use far less resources enabling them to run on inexpensive microcontrollers and radios with smaller power supplies.

B. Protocol Stack

The BLE protocol stack consists of three main components: (1) Application manages the actual use case (logic, user interface, and data handling), (2) Host containing the upper layers of the Bluetooth protocol stack, and (3) Controller containing the lower layers of the Bluetooth protocol stack including the physical radio. These can be integrated into the same chip or broken into separate ICs and connected via a standard Host Controller Interface (HCI), which allows interoperability between hosts and controllers produced by different companies.



Let's dive into the details of how the stack functions from the bottom up.

The physical (PHY) layer contains the physical communications circuitry, which modulates and demodulates analog signals into digital symbols at 1Mbit/sec (which fixes the upper bound on the output rate) using Gaussian Frequency Shift Keying (GFSK, which uses Gaussian filter to smooth positive/ negative frequency deviations, which represent a binary 1 or 0). The digitial signal is over the 2.4 GHz ISM (Industrial, Scientific, and Medical) band, which is divided into 40 channels. To avoid interference from other radios, BLE uses *frequency hopping spread spectrum* where the data channel is shifted by some arbitrary constant whenever it encounters resistance. Since there are 40 channels in the band and 3 are used for the advertiser signal, the shift is calculated as *new channel* = (*current channel* + *hop*) mod 37.

The Link Layer (LL) is a combination of hardware and software that interfaces with PHY and manages the timing, connection parameters, encryption, advertising, scanning, starting, and stopping connections as well as the "link state," which characterizes the role of the device. The roles include: Advertiser: blindly sends advertising packets before active connection without knowledge of the presence of a scanner, Scanner: blindly scans for advertising packets before active connection without knowledge of an advertiser, Master: initiates a connection and manages it later, and Slave: accepts a connection request and follows the master's timing. The LL also checks packets received against a 24-bit CRC, and requested a re-send whenever an error occurs; the LL on the master will resend the packet until it finally passes the test on the receiver.

The Host Controller Interface (HCI) is a standard serial protocol for host-controller communications with a pre-defined set of commands and events, a data packet format, flow control rules, and other procedures that vary by transport, such as UART, USB, SDIO, etc.

The Logical Link Control and Adaptation Protocol (L2CAP) provides a protocol multiplexer that takes multiple protocols from the upper layers and encapsulates them into the standard BLE packet format (and vice versa). fragmentation and recombination, a process by which it takes large packets from the upper layers and breaks them up into chunks that fit into the 27-byte maximum payload size of the BLE packets on the transmit side. On the reception path, it receives multiple packets that have been fragmented and recombines them into a single large packet that will then be sent upstream to the appropriate entity in the upper layers of the host. To draw a simple comparison, L2CAP is similar to TCP, in that it allows a wide range of protocols to seamlessly coexist through a single physical link, each with a different packet size and requirements.

The Attribute Protocol (ATT) is a simple client/server stateless protocol based on at - tributes presented by a device. In BLE, each device is a client, a server, or both, irrespective of whether its a master or slave. A client requests data from a server, and a server sends data to clients. The protocol is strict when it comes to its sequencing: if a request is still pending (no response for it has been yet received) no further requests can be sent until the response is received and processed. This applies to both directions independently in the case where two peers are acting both as a client and server. Each server contains data organized in the form of attributes, each of which is assigned a 16-bit attribute handle, a universally unique identifier (UUID), a set of permissions, and finally, of course, a value. The attribute handle is simply an identifier used to access an attribute value. The UUID specifies the type and nature of the data contained in the value.

The Security Manager (SM) is both a protocol and a series of security algorithms designed to provide the Bluetooth protocol stack with the ability to generate and exchange security keys, which then allow the peers to communicate securely over an encrypted link, to trust the identity of the remote device, and finally, to hide the public Bluetooth Address if required to avoid malicious peers tracking a particular device.

The Generic Attribute Profile (GATT) builds on the Attribute Protocol (ATT) and adds a hierarchy and data abstraction model on top of it. In a way, it can be considered the backbone of BLE data transfer because it defines how data is organized and exchanged between applications.

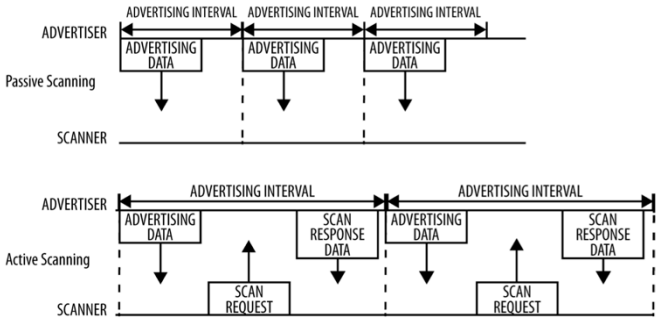
The Generic Access Profile (GAP) dictates how devices interact with each other at a lower level, outside of the actual protocol stack. GAP can be considered to define the BLE topmost control layer, given that it specifies how devices perform control procedures such as device discovery, connection, security establishment, and others to ensure interoperability and to allow data exchange to take place between devices from different vendors.

C. *Packets*

Data packets are the workhorse of the protocol and are used to transport user data bi-directionally between the master and slave. These packets have a usable data payload of 27 bytes, but additional procotols further up the stack typically limit the actual amount of user data to 20 bytes per packet, although that logically depends on the protocol being used. For example: the L2CAP packet header takes up four bytes, which means that the effective user payload length is 27 - 4 = 23.

It's worth noting here that the BLE has two kinds of packets, advertising and data, but only one format, which simplifies the implementation. carry a Advertising packets carry a payload up to 31 bytes of data in addition to basic header information and are sent at a fixed rate defined by the advertising interval, which ranges from 20ms to 10.24s, shorter intervals increase the probability of those packets being received by a scanner with the cost of higher power consumption. The scanner has similar parameters, interval and scan window, that define how often and long a scanner will listen for potential advertising packets, which also has impacts power consumption since it varies the amount of time the radio must be turned on.

Advertising packet types can be classified by three properties: connectability, whether a scanner can connect on receipt; scannability, whether a request scan packet can be sent; and directability, whether a packet is sent to a particular scanner. The scanner also has two distinct modes: passive, where the advertiser is unaware if a packet is received, and active, where the scanner requests a "scan response" packet upon receipt.



D. *Handshake*

To establish a connection, a master first starts scanning to look for advertisers that are currently accepting connection requests. The advertising packets can be filtered by Bluetooth Address or based in the advertising data itself. When a suitable advertising slave is detected, the master sends a connection request packet to the slave and, provided the slave responds, establishes a connection. The connection request packet includes the frequency hop increment, which determines the hopping sequence that both the master and the slave will follow during the lifetime of the connection.

E. *Nordic nRF8001 Hardware*

Nordic Semiconductors has been involved in low-power wireless solutions for years and, as a board member on the Bluetooth SIG, has helped define and shape the core BLE standard since its inception. Widely known in the wireless market for its popular, general-purpose radio-frequency (RF) silicon solutions, it was

one of the first companies to get affordable BLE peripheral-mode silicon to market (the nRF8001)

Nordic provides numerous examples (after registering your kit, as discussed in the note following this paragraph) based on this development kit, making it an easy choice if you just want to get started with something you know will work. Most of the demo code uses Keil's uVision (with the IAR toolchain as a second option), which is freely available from ARM for noncommercial use on projects smaller than 32 KB (which should actually cover a broad range of projects, because the SoftDevice is not included in the 32 KB limit, only the application code).

IV. TinyBlue Module

A. TinyOS

TinyOS is an operating system designed for low-power wireless embedded systems. Fundamentally, it is a work scheduler and a collection of drivers for microcontrollers and other ICs commonly used in wireless embedded platforms.

Add BLE to a tinyos mote via the active message interface and measure the power consumption compared to the original radio.

Integrate the Nordic BLE stack into TinyOS to allow motes to talk over bluetooth, this could not be a substitute for the original radio as was envisioned because Bluetooth is a connection based protocol. Tiny os was built on broadcasting openly or to specific nodes but a connection did not have to be established ahead of time in either case. So rather than attempt to connect the BLE stack into the active message interface and compare power consumption the project focused adding BLE functionality to tiny os as an expanded feature and not a substitute for the original radio.

B. "Micable" Platform

A new platform was constructed, micable, which was an adaptation of the micaz platform to incorporate the new blue tooth chip. At the time the micaz was selected because it was believed that this was the only mote platform that made SPI lines externally available via the MIB…… expansion board. It was later discovered that the only SPI line on the expansion board was SCLK, so the SPI was bit banged using the GeneralIO interface connected at the platform level the necessary IO lines.

C. Bluetooth Application Control Interface

The Nordic stack consisted of 2 main parts(high and low level platform independent software drivers) and 2 parts that were used for data encoding and queuing. Each part of the nRF8001 stack provided by Nordic became a tiny OS interface and a top level driver configuration was used to wire them together and ultimately to the lowest platform dependent layer.

-lib_aci – This was the top level provided by the nRF Bluetooth stack. It was used to determine what commands would be sent to the hal_aci_tl interface.

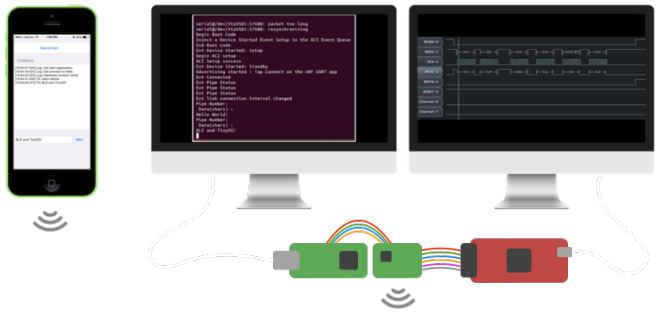
-hal_aci_tl- This module was the lowest platform independent level of the driver. It was told what command to send by lib_aci and determined how the commands would be sent. It used pins set at the platform level to bit bang spi commands out that were placed in the tx queue and add incoming message to the rx queue.

-acilib – this module encoded standard messages defined by the application command interface.

-aci_queue- This module was used to perform operations of the incoming and outgoing queues to that application command interface.

D. TinyOS Demo App

A UART test application was adapted from the Arduino library to run in Tiny OS which allowed the mote to successfully send messages back and forth between an Iphone and the micaz mote. The mote is the client and the iphone is the server in this configuration. Once the server has initiated a connection it can send information over BLE that the mote will print to the terminal via a serial port.



V. Conclusion

TABLE I. TABLE TYPE STYLES

Table Head	Table Column Head		
	Table column subhead	*Subhead*	*Subhead*
copy	More table copy[a]		

[a] Sample of a Table footnote. (*Table footnote*)

Fig. 1. Example of a figure caption. (*figure caption*)

References

[1] Abramson, Norman. "THE ALOHA SYSTEM: another alternative for computer communications." In *Proceedings of the November 17-19, 1970, fall joint computer conference*, pp. 281-285. ACM, 1970.

[2]