INVESTIGATION OF BLUETOOTH COMMUNICATIONS FOR LOW-POWER EMBEDDED SENSOR NETWORKS IN AGRICULTURE

A. D. Balmos, A. W. Layton, A. Ault, J. V. Krogmeier, D. R. Buckmaster

ABSTRACT. *Internet of things (IoT) sensor networks that monitor agricultural equipment can provide useful data to optimize a farm's productivity. For such a network to become widely adopted, the sensors should have battery lives at least as long as a full farming season. While the standard low-power wireless sensor communication platforms, e.g., Zigbee and ANT, may potentially satisfy this requirement, they lack common hardware support in consumer devices. By using more ubiquitous technology, such as cellphones and tablets, the costs and complexity and be significantly reduced. Nearly all mobile devices currently on the market have Bluetooth capability, making it a viable wireless protocol choice. In addition, the recent release and adoption of the Bluetooth 4.0 Low Energy (BLE) standard has solidified it as a strong candidate for use in very low power, long battery life networking. This work investigates how various BLE configurations affect network size, network throughput, and sensor battery lifetimes. We present a strategy to select network parameters that achieve at least a certain sensor update interval and connection latency while simultaneously minimizing the sensor's energy requirements and data latency and conforming to network size and throughput constraints. We present a simple BLE energy model created from current consumption measurements of the commercially available TI CC2540 BLE module. The combination of this BLE energy model and a simple battery capacity model allows the estimation and validation of sufficient battery life. Sensor lifetimes on the order of multiple years can easily be achieved for large update interval and connection latency sensors, which would be typical in mobile agricultural equipment.*

Keywords. *Bluetooth, Bluetooth Low Energy (LE/BLE), Farm data automation, Internet of Things (IoT), Low power, Sensor, Sensor network.*

Farming has recently seen, and will continue to see, a rapid growth in the use of technology. Owners are now better able to plan, manage, and work their farms with precision (Sassenrath et al., 2008; Heege, 2013). One obvious example of innovation is auto-steer tractors that use high-quality GPS sensors to guide steering. These systems not only ensure that an entire field is worked without wasteful overlap (Li et al., 2009) but also that it is worked with high efficiency and accurate placement of materials. However, like many other state-of-the-art agriculture technologies, auto-steer is typically limited to the sensors, monitors, and software installed, or approved, by the manufacturer. Farming decisions could be made significantly more effective if simple, low-cost, and flexible wireless Internet of Things (IoT) style sensors were available to easily collect the data that farmers need for their farming practices (Ivanov et al., 2015; Taylor et al., 2014).

A significant body of research has been produced in the area of agricultural wireless sensor networks. For example, Zhang (2004) commented on the practicality of Bluetooth 3.0 by considering requirements such as coverage area and the impact of the agricultural environment. ZigBee was similarly studied by Hebel (2006). Over the course of a year or a season, the environment in which a wireless sensor network must operate changes because of land use and vegetation growth. The link quality over various terrains and crops has been experimentally measured (Tate et al., 2008). Path loss and other quality issues have been studied for indoor applications as well. For example, Darr and Zhao (2008) developed models for the wireless environment in a poultry facility.

There has also been interest in the development of new sensors that leverage wireless sensor networks. For example, O'Shaughnessy et al. (2013) outfitted center-pivot irrigation systems with a sensor network that could precisely control the water applied to a sorghum crop, ultimately increasing yield and reducing water usage. Another group used a network of passive capillary wick samplers to estimate and monitor real-time drainage fluxes in support of soil pollutant transport modeling (Jabro et al., 2012). As a final example, Koç et al. (2013) developed a Bluetooth 3.0 based controller and Android application that remotely monitors and controls

Submitted for review in January 2015 as manuscript number ITSC 11173; approved for publication by the Information, Technology, Sensors, & Control Systems Community of ASABE in July 2016. Presented at the 2013 ASABE Annual Meeting as Paper No. 131620559.

The authors are **Andrew D. Balmos, ASABE Member,** Graduate Student, **Alexander W. Layton,** Graduate Student, **Aaron Ault, ASABE Member,** Research Scientist, and **James V. Krogmeier, ASABE Member,** Professor, School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana; **Dennis R. Buckmaster, ASABE Member,** Professor, Department of Agricultural and Biological Engineering, Purdue University, West Lafayette, Indiana. **Corresponding author:** James Krogmeier, 465 Northwestern Ave., West Lafayette, IN 47907-2035; phone: 765-494-3530; e-mail: jvk@purdue.edu.

Vol. 59(5): 1021-1029 ISSN 2151-0032 DOI 10.13031/trans.59.11173

a pull-type field sprayer.

The current literature tends to be limited to sensor network applications with predominately fixed infrastructures. In this work, we demonstrate operator-centric IoT sensor networks, similar to the above sprayer example, by considering network performance in the following real-life uses:

1. A wireless identification tag attached to every implement on a farm, whose sole purpose is to identify itself to any nearby operator. This information is useful in autogenic (Welte et al., 2013) and other systems. For example, automatic logs of the day's activities can be generated from nothing more than the identification tag data and GPS tracks of the equipment.
2. A sensor that records whether an implement is working, e.g., by monitoring pressure within a hydraulic line, machine vibration, or shaft rotation. Data from this sensor could also be used to improve the accuracy of the above activity log.
3. Weight sensors on carts that report real-time weight measurements directly, and therefore yield measurements indirectly, during silage and similar harvests. A similar Bluetooth-based sensor has been proposed by Lee et al. (2002).

The above list of uses is not exhaustive, but many IoT devices fit within this framework. For example, the identification tag represents a sensor with slowly changing data, the implement status sensor has data with a slow to moderate change rate but requires low latency, and finally the weight sensor has a high change rate.

Until recently, most traditional wireless communications protocols, such as ZigBee and ANT, have had certain shortcomings. These include, but are not limited to, energy usage, consumer-level availability, and ease of use and deployment. Many of these issues are mitigated by the new Bluetooth Low Energy (BLE) section of the Bluetooth 4.2 specification (Bluetooth SIG, 2014). BLE has been shown to have lower power requirements in IoT applications than ZigBee or ANT (Dementyev et al., 2013; Siekkinen et al., 2012), and the popularity of previous versions of Bluetooth in cell phones and tablets has enabled rapid adoption of BLE at the consumer level. ZigBee and ANT provide more advanced networking architectures, such as star, tree, and mesh topologies, as compared to BLE's lone point-to-point network with one master and many slaves, i.e., a piconet. Even so, the simplicity of BLE has helped lower the barrier for development and reduce deployment complexity. Additionally, mobile device operating systems, such as Android and iOS, offer users and developers flexible, extendable, and familiar software environments that are already portable and battery operated. Therefore, the inexpensive and ubiquitous mobile device, often already owned by individuals, can be easily made into an IoT network clusterhead. However, some unanswered questions remain about BLE's ability to meet the technical constraints of sensors:

- Can a BLE network support enough sensors?
- Does a BLE network have sufficient throughput?
- Is a BLE sensor low-power enough to last at least an entire growing season?

This article investigates those questions and presents evidence suggesting that BLE is a viable technology for use in agricultural IoT sensor networks. The work presents a simple description of the physical and link layers and the associated configuration parameters. Using that knowledge, we present a scheme of selecting the parameters such that battery life and latency are jointly optimized while still meeting or exceeding requirements for the network's maximum number of sensors and throughput. Finally, to show that sensors configured with this scheme can achieve sufficient battery life, a basic usage model is formulated using the sensor's average current draw in its various modes of operation and a simple battery capacity model. To develop the usage model, we experimentally measured the current needs of the Texas Instruments CC2549 BLE module (Texas Instruments, 2013). With these results, it is clear that sensors equipped with BLE can satisfy typical agricultural requirements.

BLUETOOTH LOW ENERGY

BLE, most recently defined by the Bluetooth 4.2 specifications (Bluetooth SIG, 2014), simplifies portions of the Bluetooth 3.0 protocol (Bluetooth SIG, 2009), in order to achieve energy savings. The physical layer is similar to the Bluetooth 3.0 design with relaxed requirements, such as timing accuracy, channel bandwidth, and modulation filtering. In-depth details of BLE are outside the scope of this work (e.g., see Gomez et al., 2012, and Bluetooth SIG, 2014, for further information). Here we focus only on the components of BLE that are important for energy consumption, particularly on the sensor side.

BLE has five major operating modes: Scanning, Advertising, Connected, Initiating, and Standby. The Advertising mode broadcasts data or connection requests, in the form of advertisements, to other sensors. The Scanning mode receives advertisements, and the Initiating mode forms the connection. Once a connection is established, the Connected mode transmits packets to the connected sensor. The Standby mode is simply a lower power setting reserved for use when the device is not participating in a BLE network.

Figure 1a illustrates the various paths taken to activate a mode. If a device activates Connected mode via the Initiating mode, it becomes a master. On the other hand, if a device activates the Connected mode via the Advertising mode, it becomes a slave. At any given time, a slave may only be connected with one master. However, a master may have many simultaneous slave connections, e.g., a piconet. Therefore, a sensor network clusterhead must operate as a master, and a sensor must operate as a slave. All communications are between master and slave. Slaves do not talk with other slaves, nor masters with other masters. A connected sensor may transmit advertisements by simultaneously being in the Connected and Advertising modes as long as the two modes do not need to transmit at the same time.

In the work considered here, we are primarily interested in the sensor side. Therefore, the relevant mode activations are the Advertising and Connected modes, as the Scanning and Initiating modes are for masters only. In an agricultural

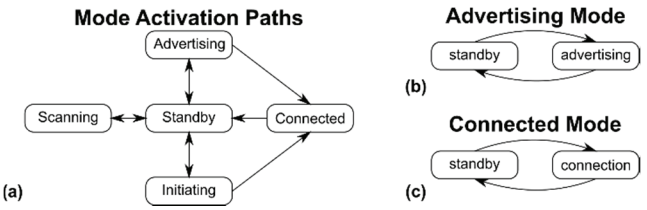

Figure 1. (a) Diagram of the allowable paths a BLE module can take to activate a mode. A sensor would activate Connected mode through Advertising and a clusterhead through Initiating. (b) State diagram of a BLE module exclusively in the Advertising mode. (c) State diagram of a BLE module exclusively in the Connected Mode.

sensor network, sensors will exclusively be in the Advertising mode or the Connected mode. We make this assumption because a sensor design intended to minimize energy usage should not waste energy advertising itself to others while it is actively connected to a master.

ADVERTISING MODE

As shown in figure 1b, a BLE device exclusively in the Advertising mode has only two possible states: standby and advertising. The standby state has no required BLE tasks, and therefore most of the hardware will enter a low-power mode. However, the advertising state requires the hardware to periodically transmit advertisement packets. Transmitting is typically a high-power operation that may result in peak currents that could affect the selection of battery chemistry.

As shown in figure 2, the Advertising mode is characterized by only one parameter, *advInterval*, which is the designed length of time between advertising states. However, to help prevent different BLE devices with equal *advInterval* values from repeatedly interfering with each other, each device automatically adds a random delay, *advDelay*, to *advInterval* at the beginning of every advertising state. The value of *advDelay* is between 0 and 10 ms and is chosen independently each time it is used.

All devices may have their own value for *advInterval*; however, it must remain between 0.02 to 10.24 s and can only be incremented in steps of 0.625 ms. As *advInterval* increases, the device transmits less often, resulting in lower energy consumption.

CONNECTED MODE

As shown in figure 1c, a BLE device exclusively in the Connected mode has only two possible states: standby and connection. The standby state has no required BLE tasks, and therefore most of the hardware will enter a low-power mode. However, the connection state requires the hardware to periodically transmit at least one packet, even if there is

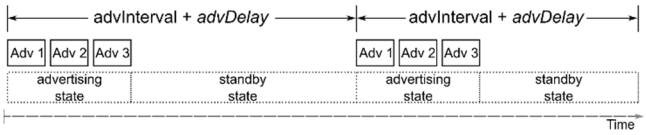

Figure 2. Example of the advertising packets transmitted during Advertising mode. Transmissions are represented by solid-line rectangles; Adv 1, Adv 2, and Adv 3 are transmitted on a separate radio frequency reserved for advertisements only, and *advDelay* is a random delay added by the BLE physical layer to avoid continuing collisions with other advertising devices. Also noted are the states of the device over time, represented as dotted rectangles.

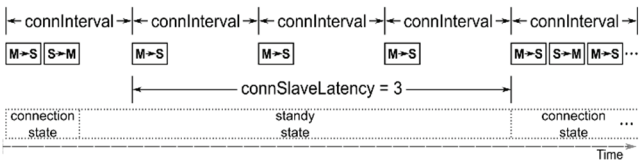

Figure 3. Example of the interaction between a connected master and slave. In this particular connection, *connSlaveLatency* is set to 3, and the slave elects to miss the full length of time. The first *connInterval* shown has only the initial round-trip event. However, in the last *connInterval* shown, the master has initiated several round-trip events. Solid-line rectangles labeled M→S are packets sent from the master to the slave, and the response packets from the slave to the master are labeled S→M. Also noted are the states of the slave device over time, represented as dotted rectangles.

no data to send. Transmitting is typically a high-power operation that may result in peak currents that could affect the selection of battery chemistry.

As shown in figure 3, the Connected mode is characterized by the *connInterval*, *connSlaveLatency*, and *connSupervisionTimeout* parameters. All three parameters are shared by both the master and the slave and are agreed upon as part of the connection process. The *connInterval* parameter is the designed length of time between connection states. During a connection state, the master addresses the slave by transmitting a packet, and the slave responds by transmitting a packet back, completing a so-called round-trip event. If the master, slave, or both have no data queued, then they may send an empty payload packet. A slave using the slave latency feature to save energy may skip the connection state for up to *connSlaveLatency* consecutive lengths of *connInterval* time. Figure 4 is a flowchart showing the decisions made when considering whether to enter the connection state or not.

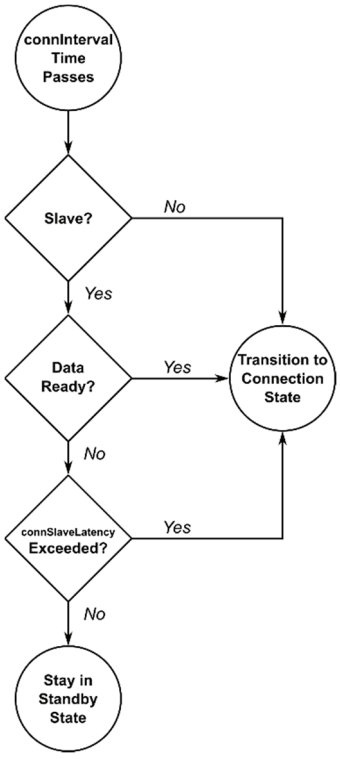

Figure 4. Logic used by a BLE device when deciding whether or not to enter the connection state.

The *connInterval* parameter can be set to a value between 0.0075 and 4 s in steps of 1.25 ms, and *connSlaveLatency* can be any integer between 0 and 499 connection states. As *connInterval* increases, the device transmits less often, resulting in smaller energy consumption. When a slave increases *connSlaveLatency*, it can choose to transmit less and thereby reduce the energy consumption further.

The example in figure 3 indicates that there may be more than one round-trip event per connection state. The master determines the number of round-trips that occur. If either the master or the slave has more data to send, the master can elect to continue the conversation. However, communication must not be continued if any of the following conditions occur:

- The master has commitments to other slaves.
- There is not enough room in the current *connInterval* to complete a full round-trip.
- Neither side has more data to send.
- Too many errors have occurred.

Connection mode can be terminated upon the request of either the master or the slave, or because the physical radio link is broken for longer than *connSupervisionTimeout* seconds. Therefore, equation 1 must be satisfied:

$$(connSlaveLatency + 1) \times connInterval \leq connSupervisionTimeout \quad (1)$$

In agricultural scenarios, it is unlikely that sensors would need to quickly disconnect from one clusterhead and connect to another; therefore, it is assumed that *connSupervisionTimeout* should always be set to its maximum value of 32 s. As an example, if the tractor driver changes, all of the tractor's sensors need to establish a connection with the new operator's mobile device (clusterhead). However, the sensors would not know to do this until the connection with the old, now unavailable, clusterhead is determined lost, *connSupervisionTimeout* seconds later. Even with the maximum timeout, the new connections would typically form before the operator had the opportunity to begin any work. In the event that this is not true, a sensor may buffer data during the outage, ensuring no data loss. The *connSupervisionTimeout* parameter may range from 0.1 to 32 s in steps of 10 ms. The *connSupervisionTimeout* value has no direct effect on energy consumption. However, as shown in equation 1, if the parameter is set too small, it may impose a limit on the maximum value of *connSlaveLatency* and *connInterval* that results in a configuration with higher energy usage.

HIGHER LAYERS

Higher layers of BLE include features to help facilitate communication and data transfer; in general, these protocols are not important when selecting the physical layer parameters. However, to calculate network throughput, it is important to know how many data octets are in a packet. We assume that the sensor will use the Generic Attribute Profile (GATT) framework, which defines a set of procedures that a device can use to read, write, and discover stored data points, called characteristics. A GATT notification allows a slave to automatically send up to 20 octets, or 160 bits, of data in the next available connection interval after it changes in the device's internal GATT table. The notification is sent using a BLE L2CAP B-frame, which allows for up to 20 octets of data plus 7 octets of headers, resulting in a maximum frame size of 27 octets (Bluetooth SIG, 2014). Therefore, the master does not need to waste any time or throughput sending specific read requests. For the remainder of this work, during Connection mode, we assume that all packets sent by the slaves are GATT notifications and all packets sent by the master are empty, which represents the most common scenario of sensors generating data to be used at the clusterhead.

PARAMETER SUMMARY

A BLE network is completely configured by the four parameters *advInterval*, *connInterval*, *connSlaveLatency*, and *connSupervisionTimeout*. For the Advertising mode, *advInterval* is the designed length of time between advertising states. For the Connected mode, *connInterval* is the designed length of time between connection states, *connSlaveLatency* is the number of connection states a device is allowed to skip, and *connSupervisionTimeout* is the length of time a device will wait before leaving the Connected mode if it is has not received any packets. Table 1 reviews the allowable range, step size, and the impact on the master and slave energy and data latency for each of these parameters.

NETWORK DESIGN PROBLEM

The Advertising mode parameters do not influence the Connected mode, and vice versa; therefore, we can view them as two independent problems that can be solved individually. The Advertising mode design is simple; it only requires the selection of *advInterval*. When a nearby clusterhead is initiating, *advInterval* sets the worst-case connection latency; *advInterval* does not impact the network's maximum size, data latency, or throughput.

The Connected mode design is slightly more complex, requiring the selection of *connInterval*, *connSlaveLatency*, and *connSupervisionTimeout*. From an earlier assumption, *connSupervisionTimeout* is set to 32 s. The values of *con-*

Table 1. Relationships between energy usage and data latency of the master and slave with respect to the individual BLE parameters.

Mode	Parameter	Range	Step	Advertiser/Slave Energy	Scanner/Master Energy	Latency
Advertising	*advInterval*	20 ms to 10.24 s	0.625 ms	Inversely proportional	Directly Proportional	Directly proportional
Connected	*connInterval*	7.5 ms to 4 s	1.25 ms	Inversely proportional	Inversely proportional	Directly proportional
	connSlaveLatency	0 to 499	1	Inversely proportional	Not affected	Slave → master: no affect; Master → slave: directly proportional
	connSupervisionTimeout	100 ms to 32 s	10 ms	N/A	N/A	N/A

nInterval and *connSlaveLatency* have many possible options. The following sections develop a strategy for selecting *connInterval* and *connSlaveLatency* such that battery life is maximized and the data latency is simultaneously minimized while still meeting or exceeding requirements for the network's maximum number of sensors and throughput.

Network Size Constraints

A network configuration scheme needs to respect a necessary minimum number of sensors, or network size. The BLE specifications technically allow a network to have a nearly unlimited number of sensors; however, there is a bottleneck. For a master to ensure that it will not miss any connection states, it may not schedule a connection state to begin during the first round-trip of another. Therefore, as shown by equation 2, the maximum number of slaves supported in a network (N_s) is the number of complete round-trips that fit within the smallest *connInterval* of the entire network. The length of a round-trip for the notification-only networks considered in this article is 676 µs (Bluetooth SIG, 2014):

$$N_s = \left\lfloor \frac{connInterval}{676\,\mu s} \right\rfloor \quad (2)$$

where $\lfloor \bullet \rfloor$ is the floor operator, i.e., round down.

BLE piconets are point-to-point networks, and farm equipment and infrastructure tend to be spread out over a large geographical area. Therefore, the number of supported slaves is, in general, large compared to the number of sensors in range of any one clusterhead. However, the network becomes limited if there is a sensor with a small update interval. An example of this would be a low update interval sensor operating with large *connSlaveLatency* to reduce energy consumption that also uses the smallest *connInterval* (7.5 ms) to improve its data latency. In this case, the network is limited to 11 total sensors (7.5 ms / 0.676 ms).

If network size is of concern, the network designer should use equation 2 to determine a lower bound on *connInterval* based on the projected maximum network size.

Network Throughput Constraint

Similar to the network size constraint, the network designer must understand how throughput is restricted in order to design a configuration strategy. The maximum achievable throughput is a completely full notification (20 octets or 160 bits), multiplied by the number of round-trips per second. The restriction on the number of packets per second is based on the requirement that a master not miss the beginning of any connection states. Therefore, assuming the master has scheduled all of the slaves such that there is not any wasted time, the number of packets that can fit within the smallest *connInterval* of the entire network is also the largest block of consecutive packets. For example, either the network has the maximum number of slaves, each completing only one round-trip, or there are fewer slaves, some transmitting multiple notifications per connection state. Therefore, the total network throughput (T) follows equation 3:

$$T = \frac{160\ \text{bits}}{connInterval} \left\lfloor \frac{connInterval}{676\,\mu s} \right\rfloor \quad (3)$$

As shown in figure 5, the network throughput staggers up and down. The stagger becomes less in amplitude and approaches an asymptote of 236.68 kbits s^{-1} as *connInterval* is increased. This stagger is a result of there not being enough time to complete a round-trip communication at the end of some *connInterval*s, and therefore the time is wasted. As *connInterval* becomes larger, the wasted time is a smaller percentage of the total transmitting time, so throughput is less affected.

Network throughput may not be the most important consideration for an agricultural BLE sensor network because even the minimum theoretical throughput, 219.4 kbits s^{-1}, is significantly larger than the needs of most sensors. As an example, again consider the weight sensor used for mapping silage yield. If the tractor were using a low-cost GPS, with roughly 3 m of error, and traveling at the speed of 5 mph (2.2 m s^{-1}), the sensor would need an update interval of roughly 1.3 s to produce a yield map with the same level of spatial error. Assuming the weight measurement requires five octets of data for each set of axles, and that the silage cart has three axles, then an overall measurement is 15 octets in length. Therefore, the sensor's throughput needs are 15 octets of data per 1.3 s, or 0.09 kbits s^{-1}. If the GPS was of the best quality, with roughly 3 cm of error, then the sensor would require an update interval of 13.6 ms and therefore a throughput of roughly 8.8 kbits s^{-1}. It is unrealistic to achieve a silage yield map with a resolution of 1 cm or even a weight sensor that could provide accurate measurements that quickly, so it is likely the network designer would choose a larger update interval. However, this is a good example of the throughput requirements for a quickly updating sensor.

If the network throughput is of concern, the network designer should use equation 3 to determine a lower bound on the value of *connInterval* based on the required network throughput.

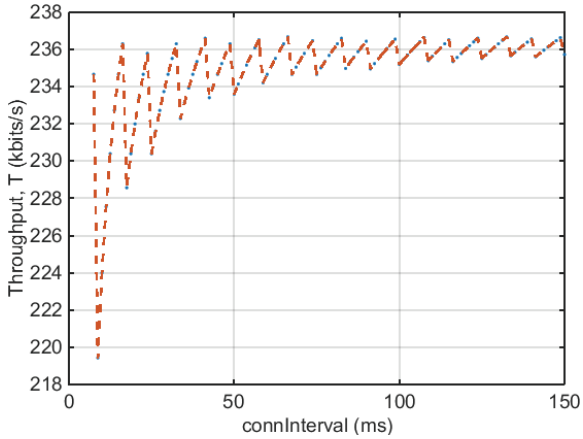

Figure 5. Theoretical maximum network throughput (notifications only), shared among the slaves, as a function of the smallest *connInterval* in the entire network. The trend continues, eventually reaching a maximum of 236.68 kbits s^{-1}, as *connInterval* is increased to 4 s.

PARAMETER SELECTION

This section presents a simple method that a network designer could use to select the parameters that meet the required update interval, connection latency, network size, and network throughput.

Advertising Mode

There is only one parameter to select in the Advertising mode, *advInterval*. As mentioned previously, this parameter is entirely based on the sensor's required connection latency. To maximize battery life, *advInterval* should equal the maximum connection latency that the sensor can tolerate, thereby minimizing the required number of advertisement transmissions and minimizing the mode's power. For example, an identification tag on a tractor can easily use the largest connection latency (10.24 s) because, in general, it takes a much longer time for an operator to begin work then for the identification, or connection, to be made. However, weight sensors on a silage cart may need a small connection latency because the cart typically pulls alongside a harvester that is already in operation.

Connected Mode

An obvious strategy for picking *connInterval* and *connSlaveLatency* is to maximize *connInterval* and then maximize *connSlaveLatency*, ensuring that the desired update interval is achieved, i.e., *connInterval* × (*connSlaveLatency* + 1) = update interval. However, this method provides the worst-case data latency. If the data are ready early or if errors caused the initial round-trip to fail, then the slave may have to wait the full *connInterval* again to transmit. Because of this, we alternatively opt to minimize *connInterval* and maximize *connSlaveLatency* while still meeting the update interval requirement.

The alternative method achieves the maximum battery life because the sensor is still only required to transmit once per update interval. However, because *connInterval* is minimized, the potential data latency is reduced. Equation 4 provides the value of *connInterval* that maximizes battery life and minimizes data latency and still achieves the network requirements. However, *connInterval* may be set to a larger value if required for network size and/or network throughput constraints. Equation 5 calculates the necessary *connSlaveLatency* to meet the update internal requirement and still minimize energy consumption, given a particular *connInterval*. As written, the three-input minimize function of equation 5 ensures that equation 1 is satisfied and that the allowable range of *connSlaveLatency* (maximum of 499) is respected. If the update interval is not exactly achievable, then equation 5 will slightly undershoot the interval. If preferred, equation 5′ can be used instead to produce a configuration that meets or slightly overshoots the update interval. Note that these equations only differ in the use of the ceiling versus the floor operator in the first argument of the minimum function:

$$connInterval = \begin{cases} 7.5\text{ ms} & 0 < \text{Update Interval} < 3.7425 \\ \left\lceil \dfrac{\text{Update Interval}}{500 \times 1.25\text{ ms}} \right\rceil \times 1.25\text{ ms} & 3.7425\text{ s} \le \text{Update Interval} < 32\text{ s} \\ 65\text{ ms} & \text{Update Interval} \ge 32\text{ s} \end{cases} \quad (4)$$

$$connSlaveLatency = \min\left(\left\lfloor \frac{\text{Update Interval}}{connInterval} \right\rfloor, \left\lfloor \frac{32\text{ s}}{connInterval} \right\rfloor, 500 \right) - 1 \quad (5)$$

$$connSlaveLatency = \min\left(\left\lceil \frac{\text{Update Interval}}{connInterval} \right\rceil, \left\lfloor \frac{32\text{ s}}{connInterval} \right\rfloor, 500 \right) - 1 \quad (5')$$

where $\lceil \bullet \rceil$ is the ceiling operator, i.e., round up.

If the update interval exceeds *connSupervisionTimeout* (32 s), then the method is no longer able to minimize the slave's power in the same sense. In this case, the sensor cannot be silent for the full update interval because a supervision timeout would occur. Therefore, the method selects network parameters such that a slave connection event occurs once per *connSupervisionTimeout* as well as minimizes *connInterval*.

As an example of the method's usage, consider the weight sensor for silage yield mapping, discussed earlier. It required an update interval of 1.3 s and 13.6 ms for the low-cost and high-quality GPS situations, respectively. Therefore, using equation 4, *connInterval* is set to 7.5 ms in both cases. Equation 5 produces a *connSlaveLatency* of 173 for the low-cost GPS, resulting in sensor update every 1.2975 s, and 1 for the high-quality GPS, resulting in a sensor update every 7.5 ms.

As an additional example, consider the implement-engaged sensor that reports if a tractor's implement is working the ground. Assume that it is updating at a speed suitable for pairing with a low-cost GPS and is in a network that has a size restriction of at least 55 sensors. In this case, *connInterval* must be at least 37.5 ms to allow for that many sensors. However, equation 4 suggests that the optimal value, considering only energy usage optimization, is 7.5 ms. Because the network size restriction requires 37.5 ms, this larger value must be chosen instead of 7.5 ms, sacrificing some power usage in order to support a large network size. As a result, *connSlaveLatency* is set to 35, resulting in a sensor update every 1.3125 s.

BLE BATTERY MODEL

In order to estimate the battery life of a BLE sensor, an energy usage model is required. Therefore, we measured the average current draw of a Texas Instruments CC2540 BLE module on the CC2540DK-MINI development board (Texas Instruments, 2013) while it transmitted advertisements and maximum payload GATT notifications. However, the development board has various sensors and components unrelated to Bluetooth that affect the current measurements. As suggested by the AN092 application notes (Kamath and Lindh, 2012), we modified the board by removing those components, leaving only what is necessary for the BLE module to function. We also modified the on-board 3.0 V coin cell battery holder so that an external Agilent E3631A power supply (Agilent, 2013) could be used as the source.

To measure the current draw versus time, we installed a 0.47 Ω shunt resistor between the external power supply and the development board's modified 3.0 V coin cell battery holder. An Agilent MSO-X 4024A oscilloscope (Agilent,

2014) captured the voltage waveform dropped on the sense resistor after being amplified by a Texas Instruments INA210 bi-directional zero-drift series current shunt monitor (Texas Instruments, 2014). The oscilloscope was configured to trigger on a voltage spike caused by an inrush current when the BLE module transitioned states. As a result, all the captured waveforms start directly at the beginning of each transmission event.

Figures 6 and 7 are several examples of the collected waveforms recorded during the advertising and connection states, respectively. Clearly, there is some small variation in the length of these events, although the size and shapes of the various curves are quite predictable. An advertising state requires an average current draw of 8.6 mA and takes 4.0 ms to complete. A connection state requires an average current draw of 7.8 mA and takes 3.0 ms to complete. The average current consumption during the standby state is 1.1 μA. We rounded the average measurements to two significant digits to reflect our estimation of the measurement error.

The BLE standard does not specify a power control scheme, so we assumed there is not one in use. All transmissions in this work were done at a constant transmit power of 0 dBm. As a result, the distance between the master and slave does not affect the energy usage. It has been shown that 0 dBm transmit power provides a BLE communication range on the order of 100 m (Ekström et al., 2012). BLE's physical layer bit rate is 1 Mbits s^{-1} (Bluetooth SIG, 2014). We assumed that the maximum amount of data transmitted in a connection state is the 20 octets of payload data in a GATT notification. As a result, the time required to transmit the sensor's data is at most 5% of the entire connection state, and therefore payload length does not appreciably affect the average current draw. To be on the safe side, figure 7 was captured using 20 octet data payloads; therefore, at worst, the average current will be slightly overestimated for smaller payloads.

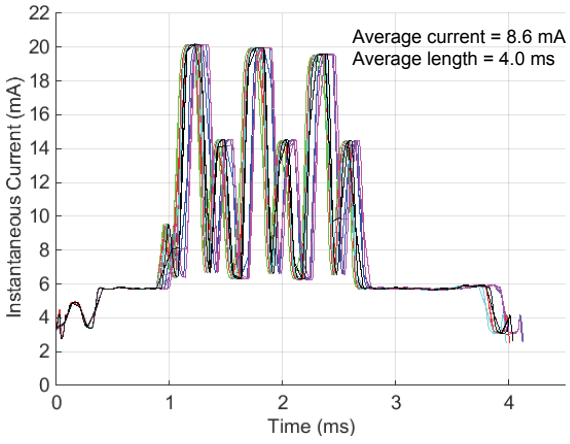

Figure 6. Selected current profiles from a CC2540 during the advertising state. The profiles were captured by triggering on a current rush caused by the transition from standby state. The CC2540 transmit power was set to 0 dBm.

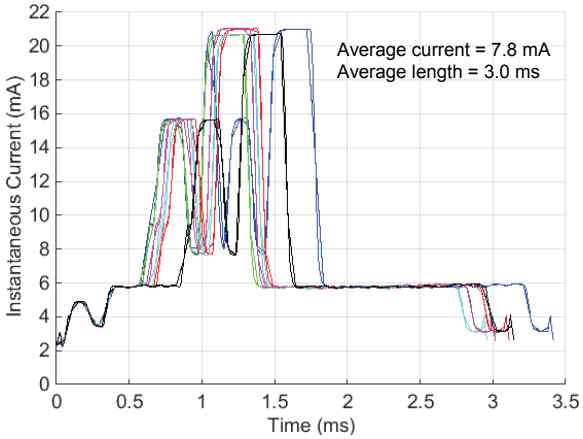

Figure 7. Selected current profiles from a CC2540 during the connection state. The profiles were captured by triggering on a current rush caused by the transition from standby state. The CC2540 transmit power was set to 0 dBm.

To calculate the average current, there needs to be an assumption of the sensor's usage. Here, we define a parameter, yearly usage, which is the number of hours per year that the sensor is in connected mode, i.e., entering the connection state every update interval seconds. The rest of the time, the sensor is in advertising mode, i.e., entering the advertising state every connection latency seconds. The total time per year that the device is in the connected state is therefore T_c = yearly usage × 66 × 20 s, and the time spent in the advertising state is $T_a = 365 \times 24 \times 60 \times 60 - T_c$ s. When in the advertising state, the total time spent transmitting advertisements equals:

$$T_a^T = 4.0 \times 10^{-3} \left(\frac{T_a}{\text{Connection Latency}} \right) \text{ s}$$

and the total time in standby is $T_a^S = T_a - T_a^T$ s.

Similarly, for the connected state:

$$T_c^T = 3.0 \times 10^{-3} \left(\frac{T_c}{\text{Update Interval}} \right) \text{ s}$$

and $T_c^S = T_c - T_c^T$ s. Therefore, the average yearly current (mA) is a function of update interval, connection latency, and yearly usage, through T_a^T, T_c^T, and T_c, and it is given by equation 6:

$$\hat{I} = \frac{8.6\,\text{mA} \times T_a^T + 7.8\,\text{mA} \times T_c^T + 0.0011\ \text{mA} \times \left(T_c^S + T_a^S \right)}{T_a + T_c} \qquad (6)$$

A simple battery capacity model, shown in equation 7, is used to estimate the time until a battery would die under the average yearly current usage:

$$T_{death} = \frac{\text{Capacity}}{\hat{I}} \qquad (7)$$

Figure 8 shows example battery life estimate curves as a function of update interval, connection latency, and yearly usage, assuming the sensor was configured as suggested in equations 4 and 5. Clearly, the connection latency is very important and can have dramatic effects on the sensor's lifetime. This is due to a combination of advertising being costlier in terms of energy than notifications and the sensor spending the majority of its life in the Advertising mode. This leads to a curious result: large update interval sensors

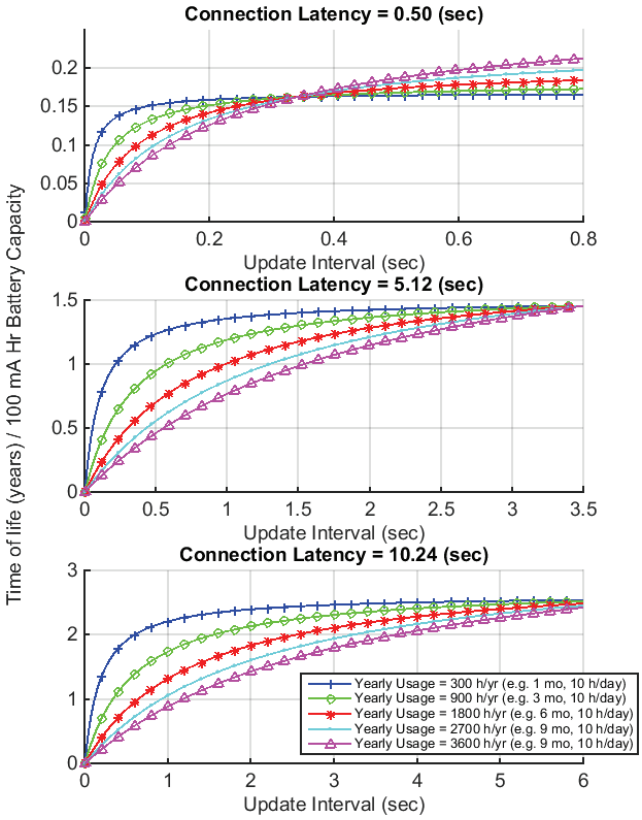

Figure 8. Expected years of life per 100 mA-h of battery capacity for various connection latency, update interval, and yearly usage values assuming the sensor has the configuration given by equations 4 and 5.

have longer battery lives when in use more often because, for large update interval sensors, the connected state is the lower-power state. In fact, the lifetime curves cross each other at the update interval where the energy spent in the Advertising mode equals the energy in the Connected mode. To the left of the crossing point, the Connected mode uses the most energy; to the right, the Advertising mode is more expensive. For very large update intervals, not shown here, the battery life model can predict unbelievably large lifespans. In this case, second-order battery effects, such as self-discharge, non-constant current discharge, and battery inefficiency, would be the limiting factor for battery lifetime.

Considering only the energy used by the BLE radio itself, the expected battery lives per 100 mA-h, given reasonable values of update interval, connection latency, and yearly usage, are longer than a typical farming season. Only for very low connection latency and update interval sensors, which are rare in agricultural sensor networks, would sensors require a battery change midseason. For example, an identification sensor tag that has been configured with the largest connection latency (10.24 s) and an update interval of 15 s can be expected to last anywhere from 2.6 to 3.1 years on a single 100 mA-h coin cell battery (e.g., Energizer, 2014) for usages between 300 and 3600 h year^{-1}.

As a slightly more intense data example, an implement-engaged detection sensor is expected to last approximately 2.3 years when configured for a low-cost GPS (update interval of 1.3 s), connection latency set to the maximum (10.24 s), and a yearly usage of 300 h year^{-1}. However, a weight sensor for silage yield mapping, configured the same way but with a more appropriate connection latency of 2.56 s, is expected to last 0.8 year, or about 10 months.

Finally, as a high data rate sensor example, consider the weight sensor for silage yield mapping configured for a high-quality GPS. That is, a sensor with an update interval of 13.6 ms, connection latency of 2.56 s, and yearly usage of 300 h year^{-1} is expected to last 0.14 year, or about two months.

Conclusion

The viability of several basic yet representative examples of sensors that would be used in agricultural wireless sensor networks has been demonstrated. In particular, this article presented:

- Evidence that BLE-equipped sensors in an agricultural sensor network can easily achieve sufficient support for the network size and throughput, and most importantly provide battery lives that are at least as long as a normal farming season.
- How BLE parameters affect the possible network size and throughput.
- A strategy to configure the network to maximize the battery life and reduce data latency.
- A battery life model that can estimate the life of a sensor in a certain configuration to verify that the sensor will operate over a sufficient lifetime.

Acknowledgements

The work reported in this article was partially supported by a USDA NIFA grant titled “Improving Agricultural Management with Autogenic Mobile Technology.”

References

Agilent. (2014). InfiniiVision 4000 X-Series. Santa Clara, CA: Agilent Technologies, Inc. Retrieved from http://cp.literature.agilent.com/litweb/pdf/5991-1103EN.pdf

Agilent. (2013). E363xA Series programmable DC power supplies datasheet. Santa Clara, CA: Agilent Technologies, Inc. Retrieved from http://cp.literature.agilent.com/litweb/pdf/5968-9726EN.pdf

Bluetooth SIG. (2009). Specification of the Bluetooth system, Ver. 3.0. Kirkland, WA: Bluetooth Special Interest Group. Retrieved from https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=174214

Bluetooth SIG. (2014). Specification of the Bluetooth system, Ver. 4.2. Kirkland, WA: Bluetooth Special Interest Group. Retrieved from https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439

Darr, M. J., & Zhao, L. (2008). A model for predicting signal transmission performance of wireless sensors in poultry layer facilities. *Trans. ASABE, 51*(5), 1817-1827. http://dx.doi.org/10.13031/2013.25297

Dementyev, A., Hodges, S., Taylor, S., & Smith, J. (2013). Power consumption analysis of Bluetooth Low Energy, ZigBee, and ANT sensor nodes in a cyclic sleep scenario. In *Proc. Intl. Wireless Symp. (IWS)* (pp. 14-18). Piscataway, NJ: IEEE.

http://dx.doi.org/10.1109/IEEE-IWS.2013.6616827
Ekström, M. C., Bergblomma, M., Lindén, M., Björkman, M., & Ekström, M. (2012). A Bluetooth radio energy consumption model for low-duty-cycle applications. *IEEE Trans. Instrum. Meas., 61*(3), 609-617. http://dx.doi.org/10.1109/TIM.2011.2172997
Energizer. (2014). Energizer CR2025 datasheet. St. Louis, MO: Energizer Holdings. Retrieved from http://data.energizer.com/PDFs/cr2025.pdf
Gomez, C., Oller, J., & Paradells, J. (2012). Overview and evaluation of Bluetooth Low Energy: An emerging low-power wireless technology. *Sensors*, 12(9), 11734-11753. http://dx.doi.org/10.3390/s120911734
Hebel, M. A. (2006). Meeting wide-area agricultural data acquisition and control challenges through Zigbee wireless network technology. In *Proc. 4th World Congress Conf. Computers in Agriculture and Natural Resources* (pp. 234-239). St. Joseph, MI: ASABE. http://dx.doi.org/10.13031/2013.21879
Heege, H. J. (2013). *Precision in crop farming: Site specific concepts and sensing methods: Applications and results*. Dordrecht, The Netherlands: Springer Science + Business. http://dx.doi.org/10.1007/978-94-007-6760-7
Ivanov, S., Bhargava, K., & Donnelly, W. (2015). Precision farming: Sensor analytics. *Intel. Syst., 30*(4), 76-80. http://dx.doi.org/10.1109/MIS.2015.67
Jabro, J. D., Iversen, W. M., & Evans, R. G. (2012). Performance evaluation and accuracy of passive capillary samplers (PCAPs) for estimating real-time drainage water fluxes. *Appl. Eng. Agric., 28*(4), 537-542. http://dx.doi.org/10.13031/2013.42083
Kamath, S., & Lindh, J. (2012). Measuring Bluetooth Low Energy power consumption. Application Note AN092. Dallas, TX: Texas Instruments, Inc.
Koç, C., Guneytepe, A., Perktas, B., & Aykanat, I. (2013). Development of a mobile app for remote monitoring and control of a pull type field sprayer. ASABE Paper No. 131619153. St. Joseph, MI: ASABE. http://dx.doi.org/10.13031/aim.20131619153
Lee, W. S., Burks, T. F., & Schueller, J. K. (2002). Silage yield monitoring system. ASAE Paper No. 021165. St. Joseph, MI: ASAE. http://dx.doi.org/10.13031/2013.10382
Li, M., Imou, K., Wakabayashi, K., & Yokoyama, S. (2009). Review of research on agricultural vehicle autonomous guidance. *Intl. J. Agric. Biol. Eng., 2*(3), 1-16. http://dx.doi.org/10.3965/j.issn.1934-6344.2009.03.001-016
O'Shaughnessy, S. A., Evett, S. R., Colaizzi, P. D., & Howell, T. A. (2013). Wireless sensor network effectively controls center pivot irrigation of sorghum. *Appl. Eng. Agric., 29*(6), 853-864. http://dx.doi.org/10.13031/aea.29.9921
Sassenrath, G. F., Heilman, P., Luschei, E., Bennett, G. L., Fitzgerald, G., Klesius, P., ... Zimba, P. V. (2008). Technology, complexity and change in agricultural production systems. *Renew. Agric. Food Syst., 23*(4), 285-295. http://dx.doi.org/10.1017/S174217050700213X
Siekkinen, M., Hiienkari, M., Nurminen, J. K., & Nieminen, J. (2012). How low energy is Bluetooth Low Energy? Comparative measurements with Zigbee/802.15. 4. In *Proc. Wireless Commun. Network Conf. Workshops (WCNCW)* (pp. 232-237). Piscataway, NJ: IEEE. http://dx.doi.org/10.1109/WCNCW.2012.6215496
Tate, R. F., Hebel, M. A., & Watson, D. G. (2008). WSN link budget analysis for precision agriculture. ASABE Paper No. 084955. St. Joseph, MI: ASABE. http://dx.doi.org/10.13031/2013.24935
Taylor, K., Griffith, C., Lefort, L., Gaire, R., Compton, M., Wark, T., ... Trotter, M. (2014). Farming the web of things. *Intel. Syst., 28*(6), 12-19. http://dx.doi.org/10.1109/MIS.2013.102
Texas Instruments. (2013). 2.4 GHz Bluetooth low energy system-on-chip. Dallas, TX: Texas Instruments, Inc. Retrieved from http://www.ti.com/lit/ds/swrs084f/swrs084f.pdf
Texas Instruments. (2014). INA21x-Q1 automotive-grade, voltage output, low- or high-side measurement, bidirectional, zero-drift series, current-shunt monitors. Dallas, TX: Texas Instruments, Inc. Retrieved from http://www.ti.com/lit/ds/symlink/ina214-q1.pdf
Welte, J., Ault, A., Bowman, C., Layton, A., Noel, S., Krogmeier, J., & Buckmaster, D. (2013). Autogenic mobile computing technologies in agriculture: Applications and sensor networking for smart phones and tablets. In *Proc. EFITA-WCCA-CIGR Conf. Sustainable Agriculture through ICT Innovation*, Paper No. C0340. Retrieved from http://www.cigr.org/Proceedings/uploads/2013/0392.pdf
Zhang, Z. (2004). Investigation of wireless sensor networks for precision agriculture. ASABE Paper No. 041154. St. Joseph, MI: ASAE. http://dx.doi.org/10.13031/2013.16175