Simultaneously Connecting Devices through Bluetooth Smart

Andreas Rüst, Mirco Gysin, Andreas Müller
Zurich University of Applied Science (ZHAW)
Institute of Embedded Systems (InES)
Winterthur, Switzerland
andreas.ruest@zhaw.ch

Martin Würms
MSR Electronics GmbH

Seuzach, Switzerland
m.wuerms@msr.ch

***Abstract* — Modern data loggers record and store large amounts of data from many different types of sensors. This enables their use in various applications to capture different measurands. However wireless access to the recorded data opens countless possibilities for novel and innovative applications, for example in the health monitoring of humans or livestock. In many of these applications a central device connects simultaneously to several data loggers. This paper discusses the experiences made during the design, implementation and test of an energy efficient wireless interface to a data logger. The implemented system serves as an application example for Bluetooth Smart. It allows a user to monitor ongoing measurements of several data loggers through a smartphone. Additionally the recorded data of several loggers can be simultaneously uploaded to a central gateway. The available Bluetooth Smart chips and software stacks place limitations on usable bandwidth as well as on the number of simultaneous connections. Although most modern smartphones include hardware for Bluetooth Smart, implementations differ widely across platforms, thus creating additional effort for the app programmer. This paper presents measurement results and introduces the implemented approaches to optimize these parameters.**

Keywords* — *Bluetooth Smart, Bluetooth Low Energy, data loggers, wireless, sensors

I. Introduction

Data loggers such as the MSR145WD from MSR Electronics GmbH [1] are electronic devices that record and store data measured from different types of sensors. A microcontroller is used to control the sensor measurements and to log the collected data together with a time stamp in a flash memory. Typical measurement quantities include humidity, temperature, pressure, acceleration (e.g. vibration, shock events), voltage, light and others. The rate of data collection can be as low as 2 Bytes/s when measuring temperature or as high as 400 Bytes/s in case a 3-axes accelerometer is used. As the measurements are typically carried out simultaneously on several sensors over several hours or days, the recorded data can easily encompass several megabytes.

Traditionally the recorded data is read out using a USB connection. However in many applications it is not possible to connect the data logger through a wire while the measurements are ongoing. Examples are monitoring applications for health, rotating machinery or shipping containers. In such cases a wireless access to the data logger enables a direct access to the recorded data while the measurements are still ongoing.

The wireless access through a smartphone can be used in an online mode to observe recent measurement results and to verify that the device is capturing the parameters correctly. E.g. that the sensor is in the desired place and that the measurement results are within the expected range. This can avoid having to repeat a measurement. Without wireless access a failed measurement is often only detected after many hours or days when the data logger is removed from the application and connected through USB to an analysis tool.

In addition to an online mode the wireless connection enables configuration of the data logger and reading of the complete flash memory with all the recorded measurements. This can either be done through a smartphone or alternatively through a central gateway (i.e. a personal or embedded computer) with a USB Bluetooth Smart dongle. Although the wireless connection offers a lower data rate compared to the USB, a wireless read-out of the flash memory is very attractive in many applications. It allows the upload to a database to take place in parallel to an ongoing data measurement.

II. Project

A. *Choice of Technology*

The presented project was launched at the beginning of 2012 with the goal of adding a wireless interface to the existing data logger device of MSR Electronics GmbH. As the logger is battery-powered a low power transceiver was a central requirement. It was important that the chosen technology would allow direct data access both from a smartphone app and from software running on a central gateway. For both cases it is mandatory that simultaneous connections to several data logger devices are possible. With its growing deployment in smartphones, Bluetooth Smart or Bluetooth Low Energy (BLE) was a natural choice which in retrospect has turned out to be a good solution. Fig. 1 shows the possible connection options for a wireless data logger.

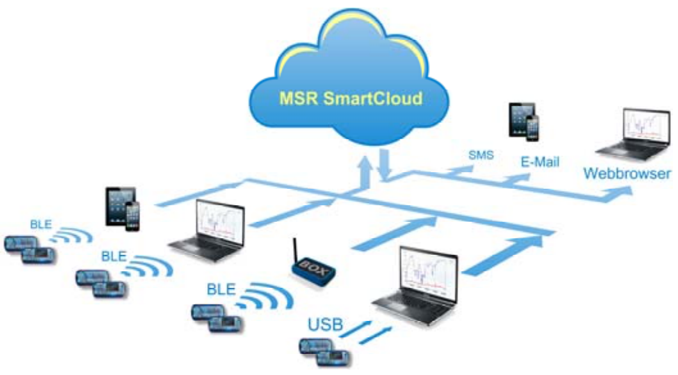

Fig. 1. Connection options for a data logger: (1) BLE to smartphone, (2) BLE to PC, (3) BLE to embedded computer (BOX) and (4) USB to PC.

B. *Implementation*

On the data logger side a Bluetooth Smart Module has been connected to the field proven MSR logging device through a Universal Asynchronous Receiver Transmitter (UART). The Bluetooth Smart Module has an integrated microcontroller and transceiver. The microcontroller runs the stack software supplied by the semiconductor vendor and an application that has been developed specifically for the present application. Fig. 2 shows a photograph of the data logger with Bluetooth Smart.

The first setup on the central side uses a commercially available Bluetooth Smart dongle connected through USB to a gateway. The application software has been written in Python and can run either on a personal computer (e.g. a laptop) or an embedded Linux computer. It contains several test modes to verify the link through long-term tests.

The second setup on the central side is a smartphone with a logger app that has been developed within the project. When the project started the Apple smartphones using iOS were the only devices that provided a stable and well documented development environment for Bluetooth Smart. Therefore this has been the main smartphone development platform throughout the project.

The system as described in [1] is in operation on selected customer sites.

Fig. 2. MSR data logger with Bluetooth Smart

III. APPLICATION EXAMPLE

An important target market for MSR data loggers is the monitoring of physical activity. The device can be used as a personal health monitor to record the individual levels of activity. The monitored person can wear the data logger attached to a leg for a week or more before the battery needs recharging. The recorded data is evaluated and interpreted by dedicated PC software. The generated charts allow an individual healthcare management or planning of complementary fitness activities.

Similarly the data logger is used to monitor individual livestock. Interpretation of the collected data allows monitoring of the breathing, walking, standing and lying behavior of an animal. For dairy cows the time of rumination can be measured.

In both use cases a wireless read-out of the recorded data starts as soon as the data logger is in the communication range of a central gateway or base station. Such a base station is usually located in a central location like an office or a cafeteria for humans or a barn in the case of animals. Several base stations can cooperate as shown in Fig. 3 to ensure coverage for larger facilities. The base stations can be embedded PCs with cable based Ethernet connections. The set-up allows read-out without interrupting the ongoing measurements.

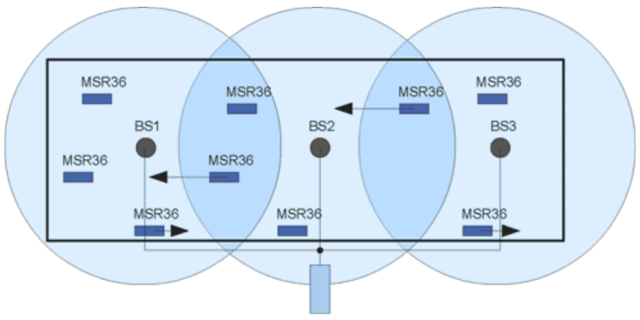

Fig. 3. Three Base Stations (BS) reading several data loggers

IV. EXPERIENCES (LESSONS LEARNED)

This section describes selected issues and experiences made during the course of the project.

A. *Data Rate Limitations*

One of the key issues during the implementation of the solution has been the limited data rate. Based on the Bluetooth Low Energy standard [2] one could expect that the payload data rate from a slave to a master could be as high as 230 kbit/s (see calculations below). However when determining the maximum achievable data rate in a specific implementation, several additional limitations have to be considered. Depending on the implementation, hardware resource constraints and software performance restrictions can lower the actually achieved data rate by more than a factor of 10 compared to the expected rate. Moreover the situation is exacerbated if several connections with the same master are taking place simultaneously.

Although we are well aware that Bluetooth Low Energy has been optimized for low power and therefore targets low data rate applications, there is a need for moderate data rates in applications like data loggers. For the presented application a twofold approach has been applied: On one hand we tried to choose variable parameters as well as possible to optimize the data rate. On the other hand the use cases have been adapted to fit the limitations, e.g. for some use cases the online mode has been used to substitute for the slow read-out of the complete data memory.

The calculation of the maximum data rate or throughput from the slave to the master is based on the definition of the connection event in [2]. See Fig. 4. Variable parameters include the length of the connection interval as well as the number of packets within a connection event and the size of the packets.

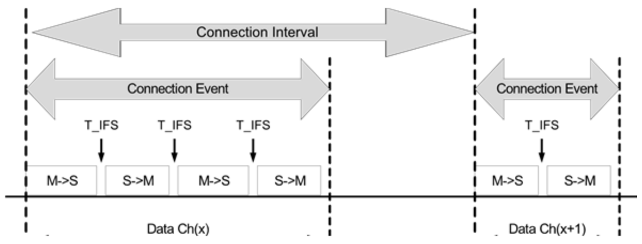

Fig. 4. Connection events and connection interval in [2].

For our calculation we assume that the master sends empty Protocol Data Units (PDU) and that the slave sends its payload data as an Attribute Protocol (ATT) notification, see Fig. 5 for the packet format. For each packet an overhead of 10 octets applies. This consists of preamble (1 octet), access address (4 octets), PDU header (2 octets) and CRC (3 octets). The length for an empty PDU (containing only the PDU header) is equal to the overhead, i.e. 10 octets. An ATT notification adds an additional overhead of 7 octets, which consists of a (Logical Link Control and Adaptation Protocol (L2CAP) header (4 octets) and the ATT opcode/attribute handle (3 octets). Therefore an exchange from master to slave and back from slave to master will use an overhead of 27 octets or 216 bits. The nominal bit rate of Bluetooth Low Energy is 1 Mbit/s, i.e. the duration of one bit cell is 1µs. For each exchange we have to add twice the time for the Inter Frame Space (IFS) which is 2 * 150 µs = 300 µs.

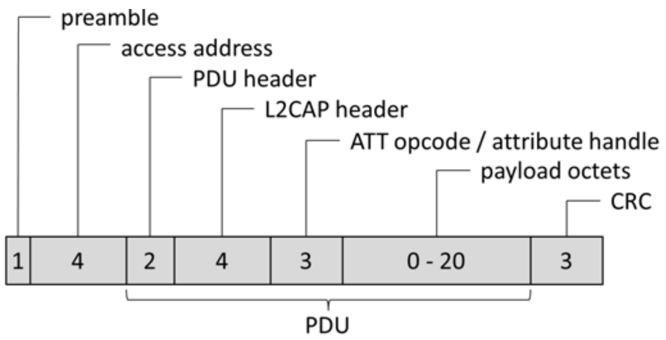

Fig. 5. BLE packet format for ATT notification showing the number of octets

To calculate the upper bound of the data rate we assume a connection event that fills the complete connection interval. This will result in back-to-back connection events.

If we now define the number of payload octets within the notification as *p*, we have *p**8 bits of payload per exchange. The time required for one exchange can be calculated as

$$T = T_{IFS} + T_{overhead} + T_{payload} = 300 + 216 + p*8 \ [\mu s] \qquad (1)$$

Based on (1) we can express the maximum achievable data rate as

$$d_{max}(p) = (p*8) / (516 + p*8) \ [kbit/s] \qquad (2)$$

According to the specification of Bluetooth Low Energy the payload field in a data PDU can be no longer than 27 octets. Considering the overheads mentioned above for L2CAP and ATT this leaves room for at most p = 20 octets of data payload. Therefore we can calculate the maximum data rate with (2) as $d_{max}(20) = 236$ kbit/s.

In many real world implementations additional limitations apply. Particularly these can be:

- Slave restrictions on the number of notifications per connection event. The restriction can be as low as 3 notifications due to limitations on buffer space.
- Minimum length for a connection interval. In the Bluetooth Low Energy standard the minimum time for a connection interval is defined as 7.5 ms. However some masters (especially smartphones) impose limits of 30 ms and above.
- Early termination of a connection event. E.g. due to two consecutive CRC errors in the same connection event or if either the master or the slave does not receive a packet from the other side.

As a result if we assume 3 notifications per connection event, a connection interval of 30 ms and a payload of 20 octets we will get a data rate of only 60 payload octets per 30 ms which corresponds to 16 kbit/s.

B. *Number of Simultaneous Connections*

One of the differentiating parameters on the central side is the number of simultaneous connections that are available. On the Apple iOS smartphones up to 8 simultaneous connections have been successfully tested.

Using the USB dongle on the gateway the number of simultaneous connections is currently limited to 3. The limitation seems to be based on resource constraints inside the microcontroller. However different BLE software stack implementations using identical hardware seem to have different upper limits on the number of simultaneous connections. As a result in some use cases more than one gateway is used to service all loggers. Of course, as a positive side effect, an additional gateway also increases the aggregated bandwidth.

C. *Establishing Connections*

Another important aspect during the course of the project has been how data loggers are associated (or paired) with central devices. There may be more data loggers within the range of a central device than available simultaneous connections. Therefore the central device has to carefully select the connections to data loggers. It has to ensure that all the required data loggers are given a fair chance to connect. Selecting the connections based on a list of recently connected data loggers prevents starvation of individual data loggers, i.e. all loggers are able to transmit their data.

The data loggers communicate important parameters during the advertisement process. These parameters include a network identification, a device address, a “complete local name”, and a unique series number of the data logger. The network identification can be configured by the user and allows operating several distinct networks in the same vicinity. The other parameters support the described process to select the next connection.

V. MEASUREMENT RESULTS

This section presents selected measurements of achieved data rates in cases where a single master device maintains simultaneous Bluetooth Smart connections with several slave devices. The measurements have been done for different master devices. On the slave side a publicly available CC2540 keyfob from Texas Instruments [3] has been used. The keyfob runs a dedicated test application that uses the TI Bluetooth Low Energy stack (version 1.3.2). The test application sends the requested number of packets with a payload of 20 and 16 octets respectively. Each packet contains one notification with a two octet sequence number and the other octets set to zero.

A. *Test Sequence*

The tests are carried out with 5000 packets using the following sequence:

- The master (i.e. the smartphone) scans for available slaves
- Connections to all the slaves are established and services and characteristics are discovered by the master
- Exchange of test parameters. E.g. the master sets the desired number of packets and the slaves report the selected connection interval for display in the app.
- The master sends a command to start the test and the slaves start to transmit their test packets.
- The master logs the start time
- As soon as all packets have arrived, the master logs the end time and performs the calculation to display the achieved data rate. Packets that require retransmission are reflected in the result as they increase the measured time to transmit all the packets.

B. *iOS (iPad 3/iPhone 4S)*

TABLE I. shows the achieved data rates per slave for simultaneous connections between an iPad 3 (version with Wi-Fi + cellular) running iOS 7.0.4 and several keyfobs. For these measurements notifications containing 20 payload octets have been used. The Wi-Fi and cellular functions have been turned off during the test. As far as possible no other applications were running on the iPad.

The limiting factors for the data rate are (1) a connection interval of 30 ms imposed by the iPad/iOS and (2) the maximum of 3 notifications per connection interval given by the keyfob. As a result 60 payload octets arrive every 30 ms, which yields a resulting data rate of 16 kbit/s. These settings could be clearly verified using a sniffer.

The data rate of 16 kbit/s can be sustained up to 5 slaves. With 6 slaves the data rate starts to drop and with 8 slaves we see connections with remarkably lower data rates.

TABLE I. DATA RATES ON IOS 7.0.4 WITH N SLAVES USING 20 PAYLOAD OCTETS PER NOTIFICATION

N	**Achieved Data Rate [kbit/s]**								
	1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***Avg.***
1	15.9	-	-	-	-	-	-	-	15.9
2	15.9	16.0	-	-	-	-	-	-	16.0
3	15.9	16.0	16.0	-	-	-	-	-	15.9
4	15.9	16.0	16.0	16.0	-	-	-	-	16.0
5	15.9	15.9	15.9	15.9	15.9	-	-	-	15.9
6	14.0	14.1	14.0	14.0	14.0	11.3	-	-	13.6
7	13.4	13.4	13.4	10.6	13.4	11.6	8.8	-	12.1
8	12.9	5.3	12.9	12.8	12.9	11.9	4.1	12.0	10.6

The measurements presented in TABLE II. used the same set-up, but this time notifications with only 16 octets have been used. Interestingly this results in slightly higher data rates. By reducing the payload per notification we can increase the achieved data rates. The shorter packet size allows the keyfob on the slave side to send more than three notifications per connection event. The number of notifications varies between connection events, but can be as high as six. The effect is caused by an improved usage of the FIFO buffers in the slave.

TABLE II. DATA RATES ON IOS 7.0.4 WITH N SLAVES USING 16 PAYLOAD OCTETS PER NOTIFICATION

N	**Achieved Data Rate [kbit/s]**								
	1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***Avg.***
1	17.0	-	-	-	-	-	-	-	17.0
2	17.0	17.0	-	-	-	-	-	-	17.0
3	17.0	17.0	16.9	-	-	-	-	-	17.0
4	16.9	17.0	17.0	17.0	-	-	-	-	17.0
5	16.9	17.0	16.9	17.0	17.0	-	-	-	17.0
6	15.3	15.2	15.2	15.2	15.3	15.2	-	-	15.2
7	17.0	17.0	12.6	17.0	17.0	16.9	3.5	-	14.4
8	4.3	8.5	17.0	17.0	17.0	17.0	7.3	10.8	12.3

C. *Android (Nexus 7)*

The app has been ported to a Nexus 7, generation 2013 running Android 4.4 KitKat. The application establishes the connections to several slaves and successfully exchanges the test parameters. However the Android callback only returns notifications from a single slave, i.e. we have not yet been able to successfully operate simultaneous connections on this platform.

On the other hand it is possible to choose a much lower connection interval compared to the iPad. This allows achieving substantially higher data rates. The achieved data rates for a single slave are shown in TABLE III.

Assuming a connection interval of 7.5 ms and three 20 octet notifications per connection event results in 60 payload octets each 7.5 ms. This corresponds to a data rate of 64.0 kbit/s. The measured data rate is slightly below at 57.2 kbit/s. Analysis with the sniffer shows that this is due to early terminated connection events.

TABLE III. DATA RATES ON ANDROID 4.4 KITKAT

Connection Interval [ms]	**Achieved Data Rate [kbit/s]**	
	20 Payload Octets	***16 Payload Octets***
7.5	57.2	46.0
28.75	15.8	16.6

VI. CONCLUSIONS

This paper describes the application of Bluetooth Smart in a wireless data logger device and discusses selected experiences made during the design of the system. In particular it presents measurement results for achieved data rates when simultaneously transmitting data from several slaves to a smartphone. It demonstrates that hardware dependent choices of parameters like the number of payload octets can have an effect on system performance. With Apple iOS sustainable data transfers over up to eight simultaneous connections have been demonstrated. With Android we were able to achieve a high data rate but only on a single connection. Other smartphone platforms like Windows 8 and Blackberry will be studied in future work.

ACKNOWLEDGMENT

A special thank you goes to Marcel da Silva who laid the foundation for this project with his great engineering work.

The presented project has been supported by the Swiss Commission for Technology and Innovation (CTI).

REFERENCES

[1] MSR145WD Wireless Data Logger with BLE, Display & MSR SmartCloud, Datasheet, MSR Electronics GmbH, 2013

[2] Specification of the Bluetooth System Version 4.1, Bluetooth SIG, December 2013

[3] Bluetooth Low Energy; CC2540/41 Mini Development Kit; User's Guide. Document Number: SWRU270C; Document Version: 1.2; Development Kit Part Number: CC2540DK-MINI; Texas Instruments, January 2013