



Marko Haatanen

**SELF-ORGANIZING ROUTING PROTOCOL FOR BLUETOOTH LOW ENERGY
SENSOR NETWORKS**

SELF-ORGANIZING ROUTING PROTOCOL FOR BLUETOOTH LOW ENERGY SENSOR NETWORKS

Marko Haatanen

Spring 2012

Information Technology and Telecommunications

Oulu university of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Information Technology and Telecommunications

Author(s): Marko Haatanen

Title of thesis: Self-Organizing Routing Protocol for Bluetooth Low Energy Sensor Networks

Supervisor(s): Teemu Korpela

Term and year when the thesis was submitted: Spring 2012

Number of pages: 37

One of the main reasons for starting this project was to learn how to utilize Bluetooth Low Energy technology. It was started on the side of a European Union funded project that creates a power consumption metering system for households. Therefore the goal of the project was to create an individual plug-in power consumption meter that communicates with Bluetooth Low Energy technology built on an Arduino prototyping platform. This thesis focuses on the creation of the routing protocol for the system. The requirements for the routing protocol was that it should be light weight and flexible. It should also be self-organizing and able to recover from the removal of nodes.

Previous knowledge from several communication courses was necessary to create such a protocol. The specifics and possibilities of the Bluetooth Low Energy radio were gathered from several white papers. This information was used to produce the basics of the routing protocol, which was then developed further by studying the effects of the addition and modifications of the features. This was mainly done brainstorming ideas. Testing the protocol was not possible, since the hardware was not ready. However, several tests were designed to prove the proper functioning of the protocol.

In the end, a light weight and flexible model for the routing protocol was created that fulfilled the requirements set in the beginning of the project. The routing protocol works for the application at hand, and it is flexible enough to fill future needs as well.

Keywords: Bluetooth Low Energy, Arduino, routing, sensor network

ACKNOWLEDGEMENTS

This thesis work was carried out in the Information Technology department at Oulu University of Applied Sciences.

I wish to express my gratitude to the people who have supported me in this work. First, I want to thank Riitta Rontu, Senior lecturer, for supervising this thesis and for her valuable advice. In addition, I am grateful to Teemu Korpela for providing me with the opportunity to write this thesis, for all the valuable advice at all stages of the work and for being the instructor of this thesis. I would also like to thank the entire OCTES team for their general feedback. In addition I wish to thank Elina Bergroth for the opportunity to write this thesis in English. I want to give special thanks for my colleges Ilpo Huru, Pentti Lopakka and Ilpo Siira for familiarizing me with the work environment and their valuable input throughout the work of this thesis.

Finally, I am grateful to my dear parents Tarja and Harri, and friends Simo Kauppinen, Henri Piirto, Joel Särkkä, Hannes Jantunen and Timo Kiiskinen for their support and inspiration during my studies.

Oulu, March 10, 2012

Marko Haatanen

CONTENTS

ABSTRACT	3
ACKNOWLEDGEMENTS	4
CONTENTS	5
ABBREVIATIONS	7
1 INTRODUCTION	8
2 GOALS OF THIS THESIS	9
2.1 Goals of the project	9
2.2 Goals of the student	9
3 LOW ENERGY WIRELESS TECHNOLOGIES	10
4 BLUETOOTH LOW ENERGY	11
4.1 Core Specification 4.0	11
4.2 Comparing to regular Bluetooth	11
4.3 Topologies of BLE	12
4.4 Data transfer	13
4.4.1 Attribute Protocol and Generic Attribute Profile	13
4.4.2 Connection state	15
4.5 Encryption	15
4.6 Bluegiga BLE112	16
5 ARDUINO	17
5.1 Communication with the BLE112	17
5.2 Gateway Arduino	18
6 SENSOR DATA	19
7 ROUTING PROTOCOL	20
7.1 Network topology	20
7.2 Metric distance	21

7.3	Joining the network	21
7.4	Transmission of data	22
7.5	Data header	22
7.6	Sensor data header	23
7.7	Routing table	24
7.7.1	Periodical broadcasts	24
7.7.2	Updating of the routing table	24
7.7.3	Selection of primary direction	25
7.7.4	Address fields	25
7.7.5	Hop distance	25
7.7.6	Loss of connection	25
7.7.7	Receiving or sending	26
8	TESTING THE NETWORK	27
8.1	Testing the routing table	27
8.2	Measuring maximum range and metric distance	28
8.3	Setting up network	28
8.4	Removal of nodes	29
8.5	Testing the data transfer	29
8.6	Time measurements	30
9	DISCUSSION	31
10	CONCLUSION	33
11	REFERENCES	34
	APPENDIX 1: Definition of the task	
	APPENDIX 2: Code for echo test	

ABBREVIATIONS

AODV	Ad hoc On-Demand Distance Vector
API	Application programming interface
ATT	Attribute Protocol
BLE	Bluetooth Low energy
BTH	Bluetooth (regular)
CRC	Cyclic redundancy Check
DARPA	Defense Advanced Research Projects Agency
EU	European Union
GATT	Generic Attribute Profile
GW	Gateway
HEX	Hexadecimal (8-bit)
NPP	Northern Periphery Programme
OCTES	Opportunities for Community groups Through Energy Storage
PDA	Personal Digital Assistant
RFU	Reserved for Future Use
RIP	Routed Information Protocol
RPL	Routing Protocol for Low Power and Lossy Networks
RSSI	Received Signal Strength Indication
TI	Texas Instruments
UUID	Universally Unique Identifier

1 INTRODUCTION

The world we live in is a dangerous place. Knowledge of the surrounding environment can radically improve our safety. Integrating sensors in structures, machinery and everyday objects give us the possibility to sense small changes that would otherwise remain unnoted. With this possibility, we have a better chance to prevent catastrophic failures, and improve the emergency response, manufacturing productivity and conservation of natural resources. However, a barrier to the wide spread use of sensor networks has been the collection of data from these sensors. Long bundles of wires and optic fiber are prone to failure, and installation and maintenance costs are a significant. (1, p. 575)

With the advancements in low power technologies, wireless sensors are becoming a more attractive option. There are several technologies available to build such networks. The work in this thesis focuses mainly on the creation of a fault tolerant wireless transportation layer for sensor networks, using Bluetooth Low Energy (BLE) technology.

BLE was chosen, not for its abilities, but for it being a new technology that needed to be studied. When starting to read the specifications for the technology, it soon became clear that BLE was not going to be an easy task. It seemed it was not meant to support the network topology needed for the task. As a workaround, the routing had to be conducted on a higher layer; in this case an Arduino prototyping platform. See Appendix 1 for the definition of the task.

The sensor network was to consist of individual plug-in power consumption meters, and the data was to be collected with the Arduino platforms. The Arduino should also be responsible for holding the routing information of the network, because of the limitations of BLE.

This thesis was conducted as an individual side project from the European Union (EU) funded Opportunities for Community groups Through Energy Storage (OCTES) project. The overall objective of OCTES is to increase the viability of renewable energy solutions in the Northern Periphery Programme (NPP) region, by influencing consumer behaviour. (1)

As a note of the time notations in the thesis; the planning work was done before writing this thesis, some of the work with the Arduino is takes place while writing this thesis, and the final implementation of the routing protocol will be done in the future.

2 GOALS OF THIS THESIS

The Goals of this thesis is divided into two categories. First is the larger goals of the project and then the personal goals of the student.

2.1 Goals of the project

The general goal of the project was to create a single outlet power consumption meter that sends the collected sensor data wirelessly through a sensor network to a centralized database. The outcome of the project should also have the possibility to be easily transformed to the collection and transportation of any other kind of sensor data. BLE was chosen as the medium for the transportation of the data.

The main goal of the thesis was to create the routing protocol for the wireless sensor network. The protocol had to be able to expand the network easily and recover from the removal of individual nodes. The routing of data in the network was to be self-organizing. All data would be sent from the nodes to a single gateway (GW), which then sends it forward to a predefined server through an Ethernet connection.

The sensor nodes consisted of the sensor itself, the Arduino platform and the BLE radio. The BLE radio in use was the Bluegiga BLE112 chip, which did not have an application programming interface (API) for the Arduino platform when the project was started. The API is being made as a separate B.Sc. thesis.

2.2 Goals of the student

The BLE112 is such a new radio that during the beginning of the work there was little to no instructions on how to use it. However bits of information were received from the manufacturer on several requests. This caused some desperation in the beginning of the work, and it felt like the greatest goal was to overcome the inevitably catastrophic failure of the project, and to defeat the windmills standing in the way. Fortunately, it did not take long before it all started to look brighter.

One of the main personal goals of the thesis writer was to improve the skills in emerging a larger view, when creating a universally applicable protocol. This includes the consideration of all the current needs and foreseeing the possible needs of future applications. This skill is of vital importance when organizing large-scale projects.

3 LOW ENERGY WIRELESS TECHNOLOGIES

Information on the BLE technology was scarce. It seemed as if the books that could have been useful for the project were published only after the completion of this thesis. Therefore, the literature on BLE was mostly limited to technical specifications and advertisements. However, there was information available on other low power technologies and their routing protocols.

ZigBee is a competing low energy communication system, which was introduced in 2004. It has already gained a strong foothold in the market. It mainly differs from BLE in data rate and network topology. The routing in a ZigBee network is very different from a supposed BLE network, since it has the possibility to use a mesh topology on the link layer. Therefore, it would be unpractical to make a similar protocol for the ZigBee. (2)

The Defense Advanced Research Projects Agency (DARPA) funded Smart Dust project was an inspiring project, even though Smart Dust motes communicate optically with lasers. The basic idea is that a large number of millimeter sized sensor nodes are spread out over a large area and they all communicate to one gateway. (3) A spin-off company of the Smart Dust project is Dust Networks which has developed the idea further to use radio technology as a communications medium. At the current time, they do not have a BLE version of their product.

Using traditional routing protocols, such as the Routed Information Protocol (RIP) is not an appealing option. This is because they often make their routing decisions on the number of hops and require return receipts. They are also mostly designed for wired networks and do not take into account the needs for low power wireless networks. (5; 6)

One option for the routing protocol was to apply the Routing Protocol for Low Power and Lossy Networks (RPL) protocol. RPL is still an Internet-Draft, but it looks like a promising protocol. Its ability to utilize IPv6 addresses will certainly be welcomed by many applications. Simulations made to the protocol show very promising results. However, the inclusion of IPv6 and Point-To-Multi-Point control messages are not necessary for the sensor network created in this project. By not using RPL as the routing protocol, the size of the routing tables are radically decreased, and the packet overhead can be reduced due to the lack of two-way communication in the network. (7; 8)

4 BLUETOOTH LOW ENERGY

Chapter four includes selected essential information about the BLE technology. It also includes information on how the technologies are used in this project.

4.1 Core Specification 4.0

In June of 2010, Bluetooth reached version 4.0 of its Core specification. As a new amendment, Bluetooth Low Energy was introduced (4). This new technology brings Bluetooth into the market of low power and low cost devices. It is designed to be able to operate for years with a single coin cell battery. BLE will foremost compete with ZigBee, which already has a strong foothold in the market.

One key property that makes BLE save energy and work faster is the utilization of the frequency channels. BLE uses the same frequencies and channels as BTH; however, the number of advertising channels has been reduced to three. Having fewer channels to scan, on the wakeup, makes the procedure faster and more power efficient (5). Figure 1 is a map of the frequency channels and frequencies. The green channels are the advertising channels and the blue ones are data channels.

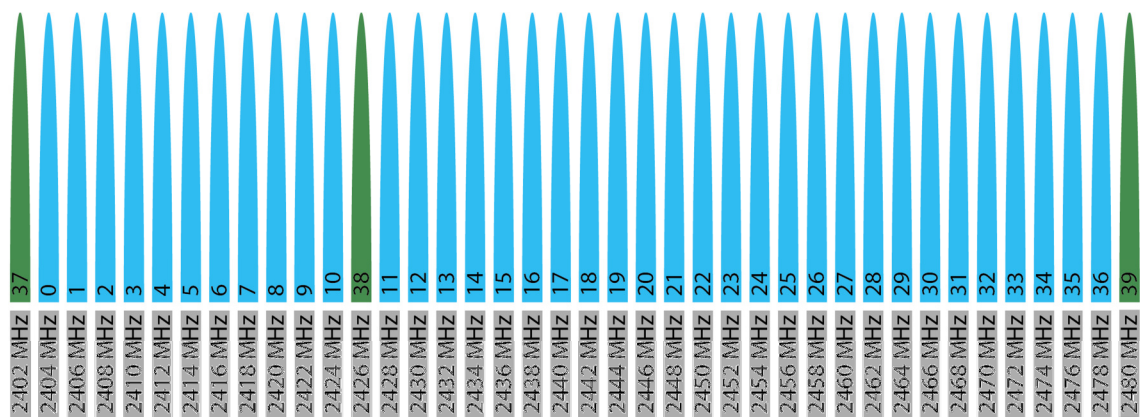


FIGURE 1: BLE link layer channels and frequencies (9, p. 2199)

4.2 Comparing to regular Bluetooth

The name Bluetooth Low Energy is slightly confusing, since it has very little to do with the regular Bluetooth (BTH). The differences between BTH and BLE are vast. The features they have in common are the frequency band and the basic structure of the communication protocol. However

they are not directly compatible. There are dual-mode chips that are capable of communicating with both BTH and BLE with the same radio and antenna, but they still use a different stack. The dual-mode chip will most likely be adopted by mobile phone and PDA manufacturers as a cost efficient way to include both technologies. (11, pp. 127-128)

The two have very different uses in applications. Where BTH is used in wireless hands-free applications and entertainment applications, BLE is designed for low power sensor data. Table 1 shows some key features of the technologies.

TABLE 1: Comparison of BTH and BLE

Technology	Classic Bluetooth	Bluetooth Low Energy
Frequency	2,4 GHz	2,4 GHz
Range	10 to 100 meters	10 to 100 meters
Data Rate	1-3 Mbps	1 Mbps
Security	64b / 128b	128b AES
Total time to send data	100 ms	<3 ms
Voice capable	Yes	No
Network topology	Scatternet	Star (single-mode)
Power consumption	1 as reference	0.01 to 0.5

4.3 Topologies of BLE

BLE is limited to point-to-point and simple one-level star topologies. In the case of a star topology, the center node is the master and the nodes around it are slaves, as shown in Figure 2. The slaves in BLE cannot communicate with each other directly, which basically makes this a point-to-point network where the master can maintain multiple connections, see FIGURE 3. (4) The slaves cannot have more than one connection at a time. Therefore, the mesh topology has to be implemented on a higher layer. In the case of this thesis, it is implemented on the Arduino platform.

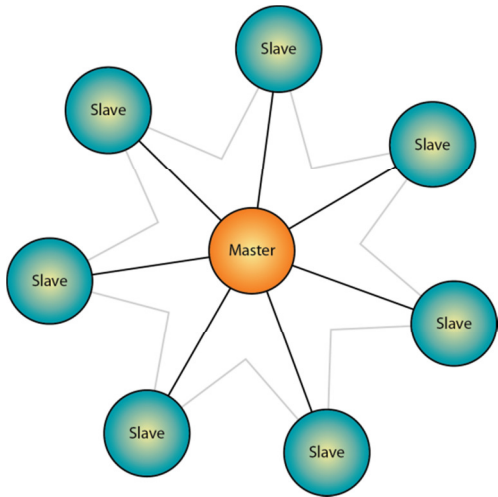


FIGURE 2: Star topology

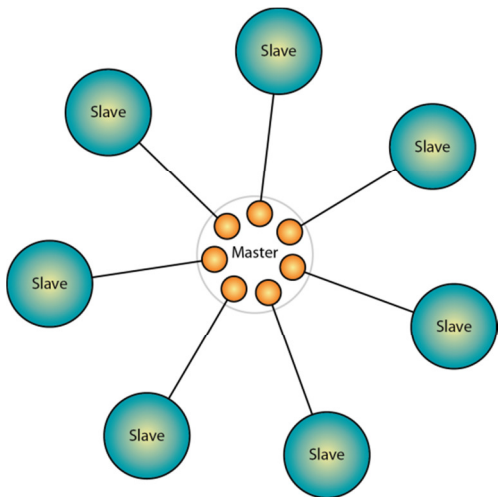


FIGURE 3: Master with multiple point to point connections

4.4 Data transfer

The only protocol used in BLE is the Attribute Protocol (ATT) (10, p. 46). It works on a Client-Server basis, where the client writes or reads data from the server. Therefore, BLE does not use the function of sending of data, per se. However, small amounts of data can be embedded in advertisement packets and be used as broadcasts to all neighboring nodes.

4.4.1 Attribute Protocol and Generic Attribute Profile

The ATT works on a Client-Server basis. The server stores data in attributes that the client can read, or alternatively write new data to the attribute. The attributes contain a Handle for

identification within the node, a Universally Unique Identifier (UUID) and the value of the attribute. The attributes also have permissions for reading and writing. Figure 4 visualizes an example of the use of the ATT. (5)



FIGURE 4: Attribute Protocol (5)

The Generic Attribute Profile (GATT) is similar to the ATT in its Client-Server architecture, but differs in the contents. GATT data is encapsulated in services and called characteristics, their types of contents are pre-defined by the Bluetooth organization. Figure 5 demonstrates the idea of the GATT. (5)

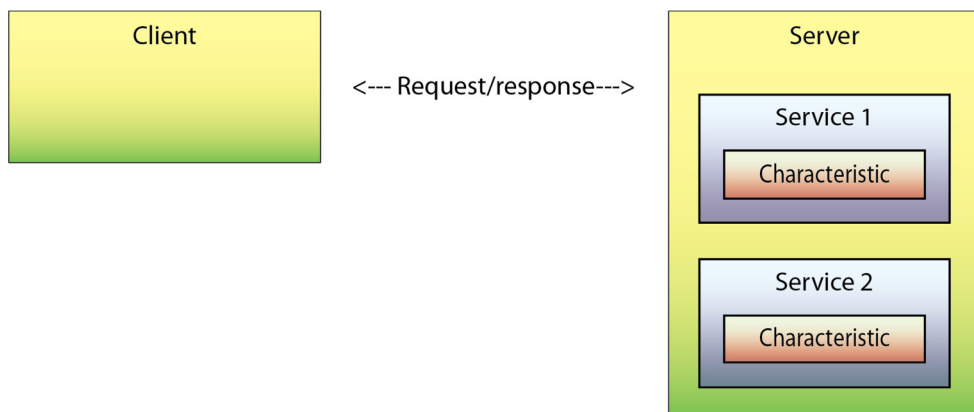


FIGURE 5: Generic Attribute Profile (5)

4.4.2 Connection state

When in idle mode, all the nodes in the network are passively scanning for transmissions. When a node has the need to transmit data towards the gateway, it goes into the advertising mode to let its neighbors know it has data to transfer. Then the neighbor receiving the advertisement initiates the connection and therefore it becomes the master, and the node with the data becomes the slave of the connection. After the slave accepts the connection, the encryption of the link is started. Now that the link is up and encrypted, the master reads the data from the slaves GATT. When the data is read and confirmed, the connection is closed and both nodes go back to the scanning mode. See Figure 6 for a visualization of the flow of the connection.

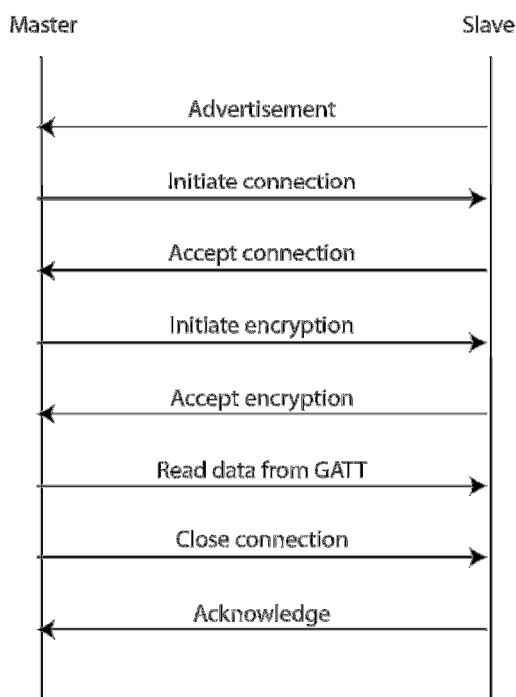


FIGURE 6: Flowchart of connection (5)

4.5 Encryption

BLE has the possibility to encrypt its link layer packet transmissions with a 128 bit AES encryption. The encryption is started after the connection is initiated and the devices have been bonded. Bonding of the devices is required for the encryption to be possible. (9, p. 2285)

4.6 Bluegiga BLE112

The BLE radio used in this thesis was a pre-production version of the Bluegiga BLE112 radio. Information on how to interact with the radio was released by Bluegiga in small amounts during the writing of this thesis.

The BLE112 is based on the Texas Instruments (TI) c2540 chip. It simplifies the interface with the c2540 to an on chip script language or a host application API. This makes it faster and simpler to build applications that implement the technology. However, the API was still in development during the writing of this thesis. This caused some setbacks especially to the writing of the API for Arduino.

The BLE112 was fitted on an adapter for the Arduino Mega 2560 by Pentti Lopakka. The board was also designed in-house. In the process of testing the system, the first prototype version of the board was destroyed. Version 1.1 corrected a pin location problem that was caused by the changing of the Arduino serial port to be used. Version 1.2 was functionally the same as version 1.1 but with some solved placements issues of some of the components. Figure 7 illustrates the adapter board.

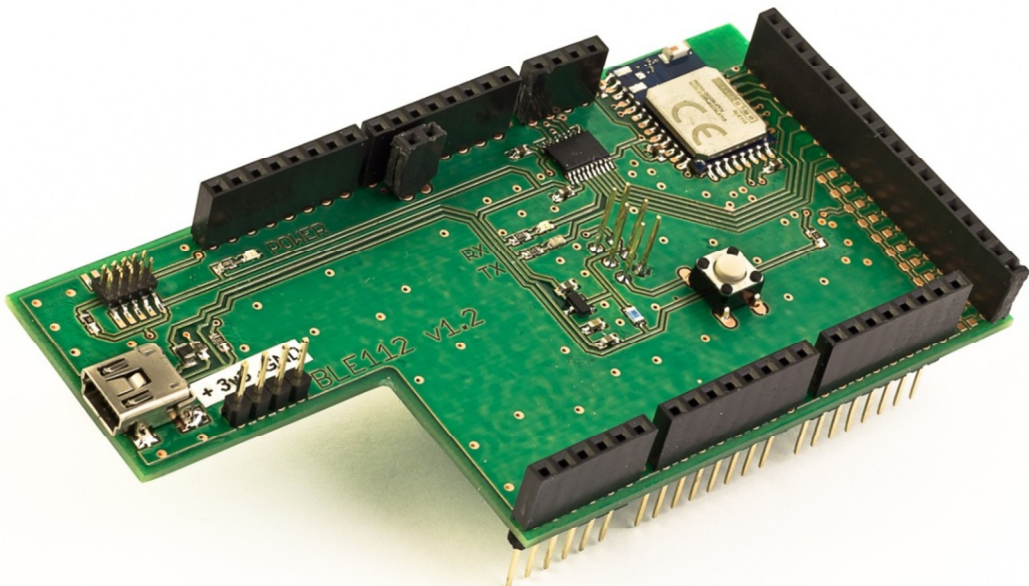


FIGURE 7: Bluegiga BLE112 on our adapter for Arduino Mega 2560

The BLE112 will, undoubtedly, be a simple and efficient way to implement BLE technology in the near future. However, for now it is still a work in the process.

5 ARDUINO

Due to the limitations of BLE, the Arduino was nominated as the heart of the system. It handles the collection of the sensor data, control of the BLE radio, keeps an up-to-date routing table of the network, relays the data of other nodes, and makes the decisions of where to send the collected data.

The Arduino is based on the ATmega2560 microcontroller. "It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button." (6) The Arduino Mega 2560 can be seen in Figure 8.

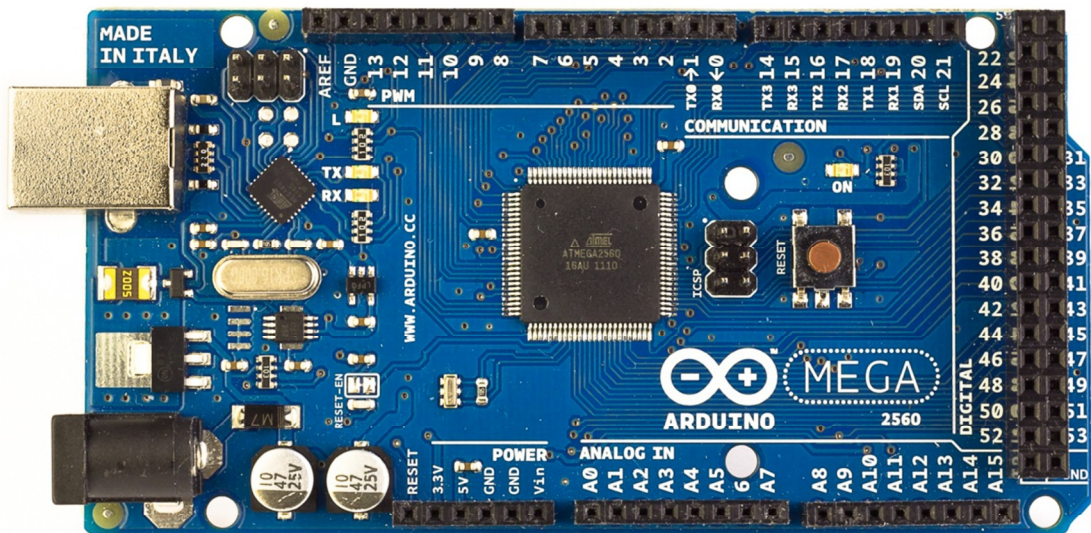


FIGURE 8: Arduino Mega 2560

5.1 Communication with the BLE112

The BLE112 API for Arduino is made as a separate B.Sc. thesis. The API is the translator between the Arduino and BLE (13, p. 111). Hence, it is required for the system to work. The BLE112 is controlled with 8-bit hexadecimal (HEX) commands. The Arduino seems to send data in the ASCII format natively. It even seems to converse the HEX input automatically to ASCII. This needs to be changed for the two to be able to communicate.

In the initial testing of the communication between the BLE112 and Arduino, it was found out that the use of the default Arduino serial port 0 caused the Arduino boot loader to fail. It seems that the serial port 0 can be used to control the boot sequence of the Arduino, and the BLE112 somehow caused it to go into a loop. Changing the connection to serial port 3 solved this problem.

For the test purpose, a firmware was created for the BLE112 that echoes everything it receives from the UART. This required the use of the built in BGScript scripting language. The scripting language enables the use of events and timers. For the echo script, an event for sending data whenever data was received was created. The data sent was a copy of the received data. A timer event was also created to send number 11 every second. This was to assure the connection was working and the BLE112 chip was still alive. See Appendix 2 for the code.

The echo signal was confirmed to be an exact copy of the original signal with the use of an oscilloscope. This proves that the HEX-ASCII conversion happens on the Arduino.

5.2 Gateway Arduino

The GW Arduino of the network is the only one different from the rest. It is equipped with an Ethernet shield. It collects the data from the sensor nodes and forwards it to the databases. The GW Arduino could function as a sensor as well, but the amount of traffic generated by the sensor network could be a problem for that.

Since the sensor network does not use IP-technology, the GW has to translate the sensor network addresses into IP-addresses. This is done using a static forwarding table. The table contains a list of possible databases in the IP-network and corresponding receiver names, used in the BLE network. When a packet has been received from the BLE side, it is opened and each sensor data packet is transmitted over the IP-network individually to the database corresponding to the receiver name in the header.

6 SENSOR DATA

The routing protocol does not take a stand on the sensor data. Each data type in the network is given a sensor type number, to identify the type of data it contains. The type number is eight bits long and network specific. Each network can therefore contain 255 different types of sensors.

The sensor data can be up to 16 bytes long. It can be divided into two 8-byte or four 4-byte data fields. This is specified in the data header.

For this thesis, the sensors to be used are power consumption meters. The Arduino handles the analogue to digital (A/D) transformation.

Figure 9 illustrates a current sensor that can be used to measure power consumption.



FIGURE 9: Current sensor

7 ROUTING PROTOCOL

The routing protocol described here was created specifically for this thesis and it was created more or less from scratch. This is because BLE cannot natively form mesh networks, and the network is built on the Arduino. The BLE112 acts as the means of transportation. Data is transmitted with two different methods; by broadcasting advertisements and by reading from the remote node ATT. While writing this, the data inclusion in broadcasting messages was not yet implemented on the BLE112, but according to Enrico Taddeo from Bluegiga Technologies Oy, it is planned for the next firmware update (personal communication, 12 December 2011).

The routing protocol is design having in mind that the sensors are mostly stationary, e.g. power consumption meters that are plugged in wall sockets. However, the routing protocol allows for moderate mobility since it is constantly aware of the surrounding nodes.

7.1 Network topology

A different kind of approach was necessary to overcome the limitations of the BLE network topologies. Instead of a traditional cloud network where all nodes are connected to each other, this network had to be passive. The nodes stay in the scanning mode until they need to transmit, or another node tries to send data to it. This way the nodes are basically not in a network at all for most of the time. The network is formed of several ad hoc point-to-point networks.

Since BLE has a very fast data transmission time with less than three ms from idle-to-idle time, it suits a multi-hop flat mesh network perfectly. (14, pp. 112-114) The network carries data only in one direction, from the sensor nodes towards the GW; this helps make the protocol light. Figure 10 demonstrates the data flow in the network.

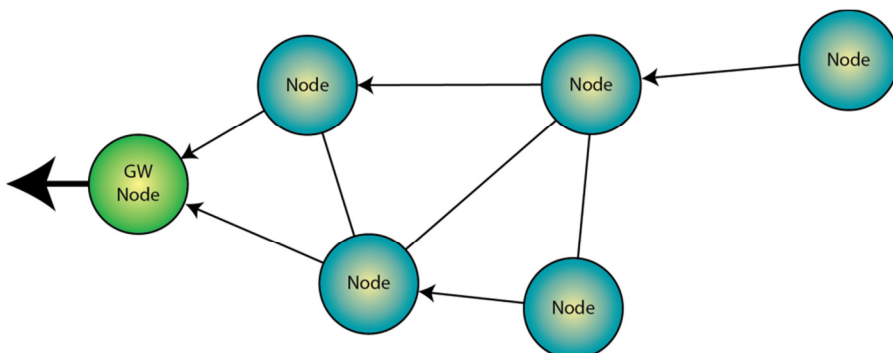


FIGURE 10: Flat mesh network

Obviously, BLE cannot make a network with this kind of topology, and therefore, the “intelligence” of the network has to be on a higher layer. In this case an Arduino platform is used. How the network is formed and which route the data takes is explained in detail in chapter 8. The goal is to have an intelligent self-organizing network.

7.2 Metric distance

The metric distance is the foundation of the routing in the network. This is used to find the way to the GW, and it is recalculated every time a broadcast is received.

The metric distance is calculated from the connection RSSI and the metric distance of the destination node. The further away from the gateway the node is, the greater the metric distance. The metric is a 16-bit number. Therefore, the possible values range from 0 to 65535. The GW has a metric distance of 1. Reading the RSSI from a connection gives a value from -127 to +20 (dBm). (9, p. 934) The formula for calculating the metric distance (x) is:

$$x = (RSSI - 20) * (-0,68) + \textit{metric distance of destination node}.$$

The RSSI is a positive number that is bigger when the connection strength is better. It is in the range of 0 to 100.

7.3 Joining the network

The network is built from the GW outward. When a new node wants to join the network, it broadcasts a signal indicating it does not have a route to the GW of the network it wants to connect to. The broadcast is a regular advertisement message with distance 65535 and next hop 0. It continues to broadcast the signal every 30 seconds. When another node receives a broadcast from a new node, it checks if the node is trying to get connected to the same network that it is a part of and then broadcasts its own metric distance to the GW. Now the new node has a direction to the GW. The joining procedure is illustrated in Figure 11.

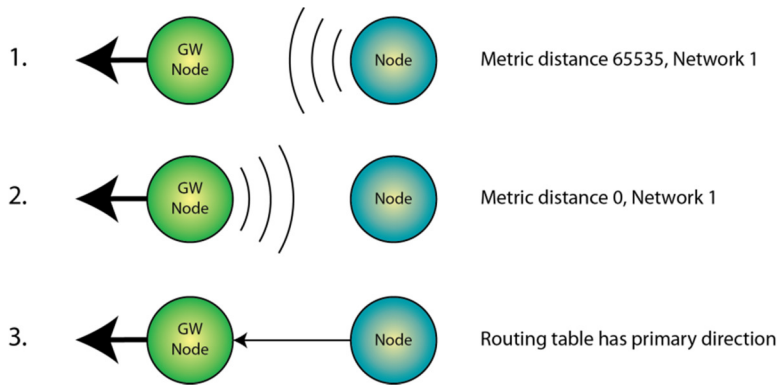


FIGURE 11: Procedure for joining the network

The GW does not answer any joining requests or broadcast itself before it has a connection to the database server. This is to prevent flooding of the data before it is ready to forward it.

7.4 Transmission of data

The data to be sent has a packet structure and it is stored in the ATT before sending. The ATT can contain up to 512 bytes of data. The BLE protocol handles the cyclic redundancy check (CRC) checks. The first bytes of the data packet contain header information explained in Table 2. These are explained in detail in section 7.5.

TABLE 2: Packet header contents (bits)

Type	Type	Type	Type	Type	Type	Type	Type
Init. Dist.	Init. Dist.	Init. Dist.	Init. Dist.	Version	Version	Version	Version
Hop count	Hop count	Hop count	Hop count	Hop count	RFU	RFU	RFU
Data amount	Data amount	Data amount	Data amount	RFU	RFU	RFU	RFU
RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU

Data transmission is made with a multi-hop ad hoc system. The collected data is transmitted to the next node in the direction to the GW. The next node is determined by its metric distance to the GW.

7.5 Data header

The data header contains an 8-bit *message type* field. This field gives the possibility to add features for the future. For now, it indicates that the data in the packet is sensor data. One future

application could be the ability to use a portable BLE device to program a nearby sensor node or to read its routing information.

The *initial distance* and *hop count* fields are a failsafe for lost packets. The initial distance is set at the origin of the data packet and it is not to be changed on its route to the GW, except when the data is integrated into another packet. In this case, it is considered that the original packet has reached the end of the line and the new packet has a new initial distance. The hop count is used to track the distance the data packet has traveled. It is initially 0 and incremented by one for every hop the packet makes. When a node receives a packet, it compares the hop count to the initial distance and determines if the packet has lost its way. If the hop count number is five more than the initial distance, it is discarded.

The *version* field initially has the value of 1. This gives the option of future modifications of the data packet structure with downward compatibility.

The *Data amount* field holds information of the amount of sensor data packets the packet contains. This needs to be used, since the packets can be combined, and therefore the packet size may vary.

RFU stands for Reserved for Future Use. These bits are important for the protocol to have the ability to adapt into the future use without major modifications to the basics of it.

7.6 Sensor data header

The sensor data header is the header for each individual sensor entry in the data packet. It contains information of the particular sensor data. It includes the name of the node, type of sensor, name of the receiver, sequential packet ID number, format of the data, and the length of the packet. See Table 3 for the header structure.

TABLE 3: Sensor data header (bits)

Name	Name	Name	Name	Name	Name	Name	Name
Sens.type	Sens.type	Sens.type	Sens.type	Sens.type	Sens.type	Sens.type	Sens.type
Receiver	Receiver	Receiver	Receiver	Receiver	Receiver	Receiver	Receiver
Packet ID	Packet ID	Packet ID	Packet ID	Packet ID	Packet ID	Packet ID	Packet ID
Format	Format	Length	Length	RFU	RFU	RFU	RFU

The *name*, *sensor type* and *receiver* names are all 8-bit long names to identify the origin and destination of the sensor data. It is used by the GW to determine where to send the sensor data and by the database to store the data in the correct location.

The *packet ID* is a sequential number that is used to determine the correct order of the packets in case they are received in a different order than they were sent. It also gives the possibility to see if any packets were lost in the transmission.

The Format determines if the data is 4x4 byte, 2x8 byte or 1x16 byte long. The Length field supplements the format by specifying how many of the fields are used of the possibilities the format allows.

7.7 Routing table

The routing table is the heart of the system. It is used to store information of the network and it is used to make decisions on where to send data. This section explains the function of the routing table.

7.7.1 Periodical broadcasts

All nodes broadcast their basic routing information on a 30-second interval. The information is embedded in advertisement messages. It includes the name of the node, metric distance of the node, hop distance, network name and the name of the node it primarily sends the collected data to. This is to prevent the forming of loops in case of disturbances in the connections between two nodes.

7.7.2 Updating of the routing table

The routing table is updated every time an advertisement broadcast is received and after broadcasting the node's own basic routing information. The model of the routing table can be seen in Table 4. The first line of the table contains the node's own information.

TABLE 4: Routing table

Own metric distance	Primary direction	Network name	Hop distance
---------------------	-------------------	--------------	--------------

Routing table:

is primary (1/0)	BLE MAC address	Sensor address	Metric distance	Hop Distance	Is present (0/1/2/3)	r/s
1	ff:ff:ff:ff:ff:ff	ff	50	1	0	s
0	ff:ff:ff:ff:ff:ff	ff	74	1	0	s
0	ff:ff:ff:ff:ff:ff	ff	142	2	0	r
0	ff:ff:ff:ff:ff:ff	ff	160	-	2	s

7.7.3 Selection of primary direction

The actual routing table part contains information on which route is primarily used as the transmission direction. This is chosen from the metric distance. The node with the lowest metric distance becomes the primary direction. In the case two connections have the same metric distances, the primary direction is chosen by random. This is to lower the load on a single node.

7.7.4 Address fields

The routing table has both the physical address of the BLE radio and the sensor address, which is the name of the node. Both addresses are needed, since addressing in the network is done with the shorter sensor address and addressing on a hop level from node to node is done with the physical address of the BLE radio.

7.7.5 Hop distance

The Hop distance is the number of hops to the gateway using that particular direction. This information is used to give the data packets their initial distance value. If the hop count of a data packet exceeds the initial distance with five, it is assumed to be lost and the forwarding stops. The data is lost without a notice to the sender. Hop distance is only applied on the entries marked as nodes to send the data to.

7.7.6 Loss of connection

If a node in the routing table has not been heard from in the 30-sec period, it is marked as not present. This is done by adding 1 to the "Is present" value. When the value is 0, the node is present and it can act as a destination for data transmission. If the value is more than 0, it is put

on hold. The value is reset to 0 if a transmission is received from the node. When the value reaches 3, it is expunged at the next periodic update. This is to keep the routing table clean.

In case the node later reappears and it is not aware of the connection break, it continues to transmit its own metric distance periodically. This broadcast is picked up by the neighboring nodes and they re-add it to their routing tables.

7.7.7 Receiving or sending

The last field in the routing table indicates if the node is the one that sends or receives data. This information is important to avoid loops. Data is not sent backwards in the system unless the routing table has no entries of nodes to send the data to. In this case, a loop warning message is sent backwards to notify of the faulty direction.

8 TESTING THE NETWORK

So far, testing has only been done for the communication between the Arduino and BLE112. Figure 12 is a photo of the assembled node with the latest BLE112 adapter board.

The testing of the routing protocol did not make it into this thesis, since the API was not done at this point. However, this section includes plans for the testing of the protocol. These tests should be run when they are possible.

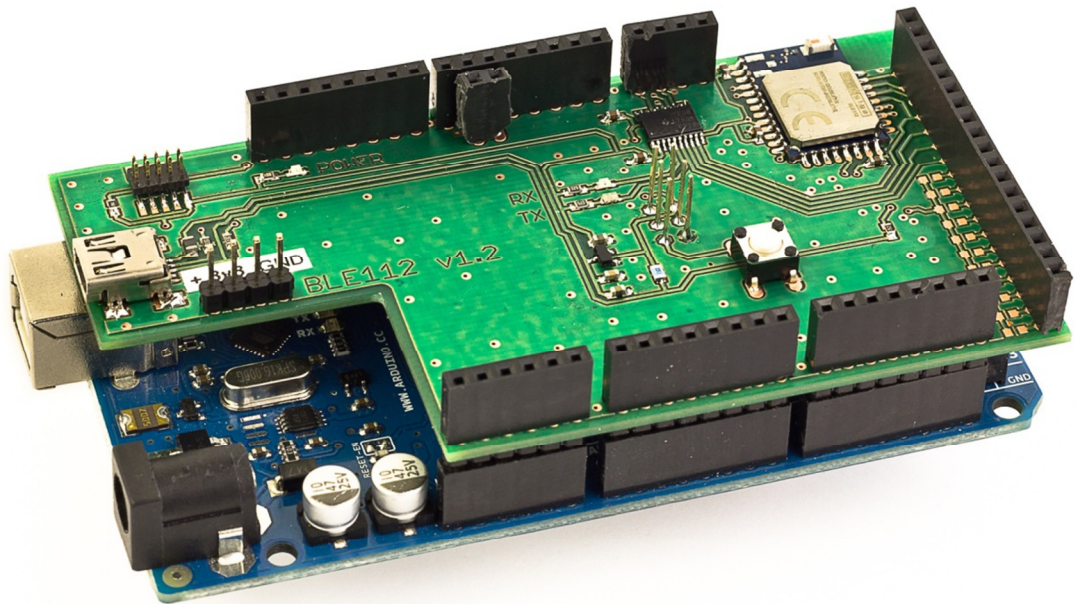


FIGURE 12: Assembled node without sensor

The actual code for the tests depends on the API. Therefore, these tests are described in human text instead of code.

8.1 Testing the routing table

This is a three-part-test that requires some extra code on the Arduino and the use of a connected computer. The aim is to make sure the routing table on the Arduino is formed correctly.

First, a single node is connected to the computer through the Arduino USB interface. The Arduino should be programmed to send the routing table to the USB interface. Since the node is alone, the routing table should only have information of the network name. This shows if the routing table is created correctly.

In the second part of the test, a GW-node is added to the network. Now the routing table of the first node should be populated with the routing information the GW node. This proves the correct basic population of the routing table.

In the third part of the test, the GW-node is turned off. After waiting for 30 seconds, the routing table of the first node should mark the GW-node as not present. The GW-node is expunged from the routing table after a two-minute-wait. This proves the correct functioning of the cleaning of the routing table in case of a lost connection.

At the end of these tests, it should be clear that the formation and population of the routing table is working properly. If for some reason one of these tests fails, the problem is sought out before continuing to the next tests.

8.2 Measuring maximum range and metric distance

For this test, the same equipment should be used as in the last test. The aim of the test is to find the maximum range between two nodes and to check the calculation of the metric distance. The node connected to the computer is stationary and the GW node should be mobile. Notes of the distance are taken between the nodes, as well as of possible obstacles and the metric distance.

The measurement is started with both nodes placed side by side. After both nodes are turned on, the routing table of the stationary node is read to make sure it is correctly populated. The GW-node is then moved away from the stationary node. This is done with a five meter increment and stopping for at least one minute for the routing table to update. It is important to make sure the node is pointing in the same direction on every stop and that there are no obstacles between the nodes throughout the test. The maximum distance is found when the node is marked as not present in the routing table. The point where the node can re-connect to the network is found by moving back one meter at a time.

8.3 Setting up network

For this test four nodes are needed, two of them are the same as in the former tests. Reading the routing table through the USB interface should be made possible on all nodes. The network is set up according to Figure 13. Node 3 should be out of range of the GW node, but in range of both node 1 and node 2.

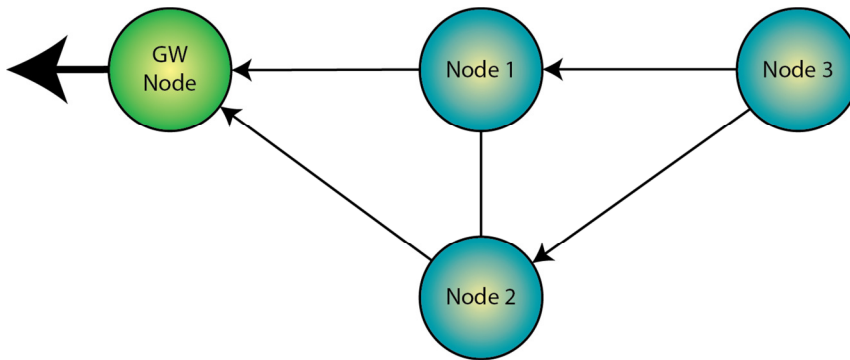


FIGURE 13: Test network 1

The routing tables are read from all nodes. Node 3 should choose Node 1 as the primary direction. When moving node 3 away from node 1 and closer to node 2, it should choose node 2 as its primary direction. By reading the metric distance of nodes 1 and 2, it is ensured that node 3 calculates the metric distances correctly and it chooses the correct node as its primary direction.

This test proves the correct forming of the routing network. If the test fails, the route selection algorithm or the calculation of the metric distance should be corrected.

8.4 Removal of nodes

The network is set up according to Figure 13. It should be made sure node 3 has node 1 as its primary direction, after which node 1 is turned off. Now node 3 should mark node 1 as not present and choose node 2 as its primary direction. The routing table is read several times during the process to see that the not present value increases up to 3, after which the node is removed from the routing table. The power is turned back on to node 1. Now the node will show up in the routing table of node 3, it should become the primary direction. Rejoining the network should take less than 30 seconds.

This test is to make sure the loss of connection feature works correctly. If there is a problem, the algorithm that calculates the “is present” value should be corrected.

8.5 Testing the data transfer

For this test the network is set up as a chain network, see Figure 14. This is done by placing the nodes in a chain formation and far enough to only be in the range of one other node in the direction of the GW node. It is made sure this is the case by reading the routing tables.

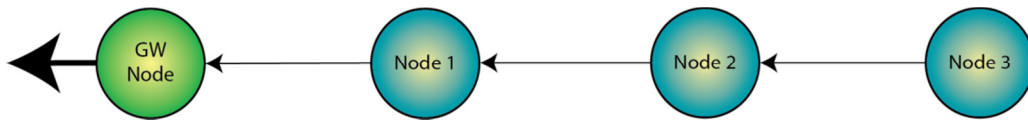


FIGURE 14: Test network 2

Node 3 needs to send test data towards the GW node. A full 16-byte data frame of ones (16 x 0xff) is sent. Header information is necessary, but the content is not important, since the data is not sent to an actual database. Monitoring node 2 will show if the data arrives and is forwarded to node 1. By also monitoring the GW node, it is made sure the data reaches its destination. When receiving the data at the GW node, the data integrity is checked.

This test should prove the concept of the routing protocol to be working as planned. If the data does not find its way to the GW node, the test is repeated and node 2 and 1 are monitored for the forwarding action.

8.6 Time measurements

This test is to determine the amount of time it takes for the data to travel from its destination to the GW node. The chain network illustrated in Figure 14 is used. A synchronized timing function is started on all Arduinos. This is for accurate timing of the packets.

A data frame containing the time of the sending is sent from node 3. This is ideally done automatically. The other nodes are set up to log the time when the data packet is received and when it is forwarded. The time it takes for the data to do one hop and how long the Arduinos processes it is calculated from the logs. The test is repeated with different distances between the nodes, taking notes of the metric distances. This information is used to compare metric distances to the time it takes to make a hop.

This test is to see how well the network performs. When more nodes are available, the test will be repeated with more nodes in the network and all of them will send data at the same time. This gives an estimate on how well the network performs when loaded.

9 DISCUSSION

Implementing an ultra-low energy BLE radio with the use of a power hungry Arduino is not very sensible. However, the concept of the system seems to be usable. The BLE112 chip contains an 8051 processor and some sensor inputs which could be used to build the entire routing system. But for now the utilization of the on-board processor is very limited and it is not capable of maintaining the routing tables and other routing information. Hopefully, at some point, there will be a firmware update that allows for greater use of the processor or maybe even a different processor that has some more advanced features on it. Then the system could actually be called a low energy sensor network.

The aim of the project was to create a fairly light routing protocol for the use of low energy sensor network. In this way the project was a success. There is still some excess overhead in the data packets that were left in for future use; this was left in for the possibility of development and universal compatibility. This overhead can easily be cut with the use of the message type and versions for a specific application. However, it was in the best interest for this project to create something that has more than one specific application.

Since the protocol does not have a feedback function, packet loss might become a concern in case of weak reception. Therefore, the protocol should not be used in critical systems, unless a return receipt function is implemented. This, however, increases the load on the network and requires a considerable overhaul on the routing table for allowing two-way communication. For now, the data flow in the network is only from the sensors towards the GW, and the confirmation on a successful data transfer can only be made on a per-hop basis.

This thesis deals with the scenario of only one GW. Nevertheless, there is no reason why the network could not have more than one GW. This could especially be useful in larger networks where hop counts, data congestion and delivery times would become uncontrolled. By adding a second gateway to the network, nodes close to it would automatically start to send data towards it due to the lower hop count and metric distance. When all the gateways are connected to the same IP network, the sensor data would automatically find its way to the database. The gateways would have to be configured in the same way; this could be maintained centrally through the IP network.

Another use of multiple gateways could be the implementation of data redundancy in the network. The idea would be to always send the sensor data to two different gateways. This would increase the reliability of the system, but it would require some changes to the routing table; mainly the addition of the route to the second GW.

One feature that could easily be implemented without massive changes to the system is the use of the message type field to read the routing table information and to reprogram a nearby sensor node. The nodes would only have to have the knowledge of these message types and the instructions on what to do when one arrives. The feature could be made with a specially designed BLE node or a smartphone. While writing this, at least the Apple iPhone 4S smartphone has integrated BLE capabilities (7). Since BLE is part of the BT4.0 standard, it is very likely that more manufacturers are going to implement this feature in their products in the future. An Android smartphone or tablet would be ideal, due to its simple application development and distribution.

10 CONCLUSION

The original time schedule for the thesis was tight to say the least. It was planned to be done in a few months. While writing this, it has been exactly three months since I started. It has been an intense experience which has improved my discipline towards writing scientific text.

Despite the difficult start of the project, it all started to congregate fairly fast. The basic concept of the routing protocol had already taken its form within the first week. In a few weeks most of the details were planned, with only some fine tuning made after that.

An interesting and educational part of the project was to communicate with the manufacturer of the BLE112 chip. The talent to explain problems that were found and to request features that were necessary, will definitely be useful in future projects. Business communication skills are vital in modern day society.

In the end, the protocol created did meet the objectives set in the definition of the task. It includes features to sense the surroundings for changes in the network topology and is capable of autonomously changing its routing accordingly. Future tests will show how effective the protocol is in action.

The work in this thesis was a good exercise to be in control of a moderate size project. Having to combine skills of different fields of engineering and making it all build up as one functioning protocol, was very instructive. This was definitely a good and comprehensive project for a bachelor's thesis. I am very pleased with the end results, both of the final product and the skills it taught me.

11 REFERENCES

1. **Wilson, Jon S.** *Sensor Technology Handbook*. Burlington, MA : Elsevier, 2005. ISBN 0-7506-7729-5.
2. **Oulu University of Applied Sciences.** Opportunities for Community groups Through Energy Storage (OCTES). Available at: <http://octes.oamk.fi/>. Date of data acquisition 27 November 2011.
3. **Kuorilehto, Mauri, et al., et al.** *Ultra-Low Energy Wireless Sensor Networks in Practice*. West Sussex : Wiley, 2007. ISBN 978-0-470-05786-5.
4. **Pister, Kris.** Smart Dust. *Berkeley University of California*. Available at: [Cited: 12 10, 2011.] <http://robotics.eecs.berkeley.edu/~pister/SmartDust/>. Date of data acquisition 10 December 2011.
5. **Black, Uyles.** *TCP/IP and Related Protocols*. New York, NY : McGraw-Hill, Inc., 1992. ISBN 0-07-005553-X.
6. **Xiong, C., Murata, T. and Tsai, J.** Modelling and Simulation of Routing Protocol for Mobile Ad Hoc Networks Using Coloured Petri Nets. *Conference in Research and Practice in Information Technology*. Available at: <http://crpit.com/confpapers/CRPITV12Xiong.pdf>. Date of data acquisition 13 December 2011.
7. **University of California.** OpenRPL. *Berkeley's OpenWSN Project*. Available at: <http://openwsn.berkeley.edu/wiki/OpenRpl>. Date of data acquisition 7 February 2012.
8. **Tripathi, J. and de Oliveira, J.** Performance Evaluation of Routing Protocol for Low Power and Lossy. *The Internet Engineering Task Force (IETF)*. Available at: <http://tools.ietf.org/html/draft-tripathi-roll-rpl-simulation-07>. Date of data acquisition 13 December 2011.
9. **Bluetooth Special Interest Group.** *Specification of the Bluetooth System, Core version 4.0*. 2010.
10. **Bluegiga.** BGT_Bluetooth_low_energy_technology.pfd. *Bluegiga*. Available at: <http://www.bluegiga.fi>. Date of data acquisition 10 December 2011.
11. **Obaidat, Mohammad S, Denko, Mieso and Woungang, Isaac.** *Passive Computing and Networking*. Hoboken, NJ : Wiley-Blackwell, 2011. ISBN 0-4707-4772-2.

12. **Arduino.** Arduino Mega 2560. *Arduino.* Available at: <http://arduino.cc/en/Main/ArduinoBoardMega2560>. Date of data acquisition 13 December 2012.
13. **Faludi, Robert.** *Building Wireless Sensor Networks.* Sebastopol, CA : O'Reilly, 2011. ISBN 978-0-596-80773-3.
14. **Gutiérrez, José A., Callaway, Edgar H. Jr. and Barrett, Raymond L. Jr.** *Low-Rate Wireless Personal Area Networks.* New York, NY : IEEE, 2007. ISBN 0-7381-4977-2.
15. **Apple Inc.** Apple - iPhone 4S - Technical Specifications. *Apple.* Available at: <http://www.apple.com/iphone/specs.html>. Date of data acquisition 7 February 2012.

LÄHTÖTIETOMUISTIO

Tekijä¹ Haatanen Marko, 040-5104047, t0hama02@students.oamk.fi

Tilaaaja² Oulun seudun ammattikorkeakoulu, Tietotekniikkaosasto

Tilaaajan yhdyshenkilö ja yhteystiedot³ Korpela Teemu, 050-5767895,
teemu.korpela@oamk.fi

Työn nimi⁴ Bluetooth Low Energy (BLE) reititysprotokolla

Työn kuvaus⁵ Tarkoituksena on luoda BLE-verkolle vikasietoinen reititysprotokolla.
Kirjallinen osuus englanniksi.

Työn tavoitteet⁶ Reititysprotokolla, mikä selviää yksittäisten laitteiden poistamisesta ja
lisäyksestä. Kerätty data lähetetään yhdyskäytävälle.

Tavoiteaikataulu⁷ Seuraava palaveri joulukuun alussa. Työ valmis helmikuun 2012
loppuun mennessä.

Päiväys ja allekirjoitukset⁸ 28.11.2011 

28.11.2011 

¹ Tekijän nimi, puhelinnumero ja sähköpostiosoite.

² Työn teettävän yrityksen virallinen nimi.

³ Sen henkilön nimi ja yhteystiedot, joka yrityksessä valvoo työn suoritusta.

⁴ Työn nimi voi olla tässä vaiheessa työnimi, jota myöhemmin tarkennetaan.

⁵ Työ kuvataan lyhyesti. Siinä esitetään muun muassa työn tausta, lähtötilanne ja työssä ratkaistavat ongelmat.

⁶ Esitetään lyhyesti ja selvästi työn tavoitteet.

⁷ Esitetään projektin tavoiteaikataulu. Silloin, kun työllä on välitavoitteita, myös ne merkitään aikatauluun.

Tavoiteaikataulun ja oppilaitoksen yleisaikataulun perusteella tekijä laatii oman aikataulunsa.

⁸ Lähtötietomuiستio päivätään ja sen allekirjoittavat tekijä ja tilaaajan yhdyshenkilö

project.xml (mandatory project file, out.hex is the firmware file)

```
-----Start of file-----
<?xml version="1.0" encoding="UTF-8" ?>
<project>
  <gatt in="gatt.xml" />
  <hardware in="hardware.xml" />
  <script in="echo.bgs" />
  <image out="out.hex" />
</project>
-----End of file-----
```

gatt.xml (mandatory file for GATT database, no database in this case)

```
-----Start of file-----
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>
</configuration>
-----End of file-----
```

hardware.xml (mandatory hardware settings file)

```
-----Start of file-----
<?xml version="1.0" encoding="UTF-8" ?>
<hardware>
  <sleeposc enable="true" ppm="30" />
  <usb enable="false" endpoint="none" />
  <txpower power="15" bias="5" />
  <usart channel="0" alternate="1" baud="115200" endpoint="none" />
  <wakeup_pin enable="true" port="0" pin="0" />
  <port index="0" tristatemask="0" pull="down" />
  <script enable="true" />
</hardware>
-----End of file-----
```

script.bgs (optional script file, mandatory when using BGScript)

```
-----Start of file-----
#Startup event
event system_boot(major ,minor ,patch ,build ,ll_version ,protocol_version ,hw )
  #start timer at 1second interval, handle 0, and repeating
  call hardware_set_soft_timer(32768,0,0)
end

#Timer event
event hardware_soft_timer(handle)
  #Send number 11 when timer event occurs
  call system_endpoint_tx(system_endpoint_uart0, 1, 11)
end

#Receive data event (endpoint is UART 0 in this test)
event system_endpoint_rx(endpoint ,data_len , data)
  #Echo received first cell of received data array back to endpoint
  call system_endpoint_tx(endpoint, 1, data(0))
end
-----End of file-----
```