Evaluation of a lightweight debugging platform for mobile sensor networks

Bachelorarbeit im Studiengang Angewandte Informatik - Systems Engineering am Institut für Informatik und Wirtschaftsinformatik der Universität Duisburg-Essen

Michael Krane

2233018

Essen, September 18, 2015

Supervisor: Hugues Smeets
1. Examiner: Prof. Dr. Pedro José Marrón
2. Examiner: Prof. Dr. Gregor Schiele

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Ich habe alle Stellen, die ich aus den Quellen wörtlich oder inhaltlich entnommen habe, als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Essen, am September 18, 2015

Abstract

In this thesis we evaluate SHAMPU, a testbeds for WSNs developt at the University Duisburg-Essen. SHMAPU is a platform, which allows the debugging and reporgramming of the attached sensor node. Out focus for this evaluation is the wireless communication between the SHAMPU nodes, for which the ANT protocol is used. We design and run, several different experiments, which try to determine the capabilities of the used ANT-chip.

List of Figures

List of Tables

Table of Source codes

Contents

1 Introduction

In the current years the amount of embedded devices has drastically increased. This is mostly due to the emergence of the Internet of Things (IoT), as well as the existence of several user friendly kits, like the RaspberryPi or the Arduino. However it is quiet complex to monitor and debug such devices, once they deployed. Even more difficult is it to monitor a whole WSN. In order to address this problem, three are different possiblities, e.g. the use of WSN testbed, which are attached to the sensor node and thus allow the node to be monitored.

Another problem of WSN, is the reconfiguration of the nodes once they are deployed, e.g. if the task the nodes fulfills changes. Most of the times, the nodes need to be collected and programmed one by one.

In order to address both those problem SHAMPU was developed. SHAMPU is a WSN testbed framework, which is attached to an exisiting node and is thus capable to not only monitoring the mote, but also to reprogram it. All SHAMPU devices are part of their own network and conenected to a base station, which acts as a master and controls all the SHAMPU nodes. The communication between the SHAMPU nodes is handle by and ANT chip.

In this thesis we evaluate that part of the SHAMPU framework and determine the capabilities and limitations of the ANT chip. Especially important are the amount of data which can be transfered inside the ANT network and the communication range of the network itself. To accuralty assess the ANT network, we design and run several experiments, which cover a wide range of use-cases.

1.1 Related work

There are already several different WSN testbeds available. Each of these testbeds were designed to fulfill a different role.

Some testbeds like FlockLab [1] provide a wide array of or Minerva [2] .... However these testbeds rely on a fixed infrastructure, the node to be tested has to be directly connected to the testbed. This makes it difficult to test and debug already deployed WSN, since it might not be possible to attach the testbed to the nodes.

Other solutions use a WSN for the testbed itself. For examples Sensei-UU [3] use testbeds which are directly attached to the tested node and communicate the debugging info via wirless 802.11 networks. This allows the WSN to be tested and debugged while it is deployed. The powerdraw of 802.11 networks, compared to other technologies, like Bluetooth is huge. This makes it hard to run the nodes of a battery and without an external power source.

BTNodes [4] or Smart-Its [5] use Bluetooth to address the power issue. But Bluetooth introduces different problems, such as size limitations for the network and a difficult set-up and use of more complex network topologies. The newest version of Bluetooth tries to address this with the introduction of ScatterNets, but the setup and maintenance of the network still remains challenging.

2 Technical Background

2.1 SHAMPU

SHAMPU (Single chip Host for Autonomous Mote Programming over USB) [6] is a WSN testbed, which allows the remote debugging and reprogramming of sensor nodes. The main advantages over other testbeds (see section 1.1) are the portability, low cost, small size and low energy consumption. SHAMPU is used as an extension to an already existing sensor node. The only requirement is that the node provides an USB-Interface, which is connected to SHAMPU. This single connection the the attached node allows SHAMPU itself to be completely OS independent.

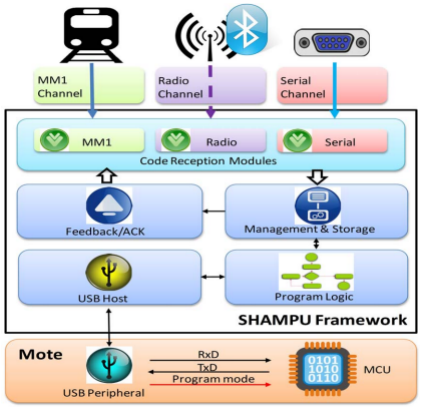

Figure 1: Overview of the SHAMPU Framework

The SHAMPU framework (see Figure 1) itself is split into multiple modules. For this thesis the most important part is the Code Reception Module, which allows the use of different protocols to connect to and communicate with the SHAMPU device. One possibility for the wireless communications with the SHAMPU device is the ANT protocol, on which we focus in this evaluation.

2.2 ANTAP1MxIB RF

In order to use the ANT protocol each SHAMPU-mote is equipped with an ANTAP1MxIB RF Transceiver Module. The module was chosen because of it's small form factor (20mm x 20mm) and very low power draw. The ANTAP1M can handle up to 4 different ANT channels with a combined message rate between 0.5Hz and 200 Hz.

To more easily test the capabilities of the ANT chip, we use a SHAMPU base-station for all the experiments. This base-station (see Figure XYXY) contains the ANTAP1MxIB and a serial interface, which allows the ANT to asynchronously communicate over RS-232 with a PC.

2.3 ANT

ANT [7] is a wireless protocol which operates in the 2.4 GHz ISM Band. Originally developed in 2003 by Dynastream Innovations Inc. for the use in wireless sensors. The ANT protocol is designed for the use in low power WSNs and puts a focus on scalability and ease of use.

One of the advantages ANT has over other protocols, like Bluetooth or ZigBee is the high level of abstraction the ANT Protocol provides.

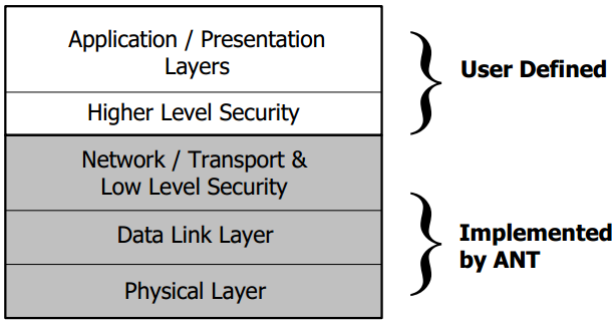

Figure 2: OSI-Layer vs. ANT Protocol

This is achieved by the incorporation the first 4 OSI-layer (see Figure 2) into the ANT protocol and thus allowing even low-cost MCU to setup and maintain complex wireless networks, since all the details of the communication is handle by the ANT-chip.

2.3.1 ANT Topology

In order for the ANT protocol to work each mote needs to be part of a network. As shown in figure 3 the ANT protocol can be used to create simple or much more complex networks. Each mote inside a network is called an ANT node and in order for two nodes to communicate with each other they need to be connected via a channel.

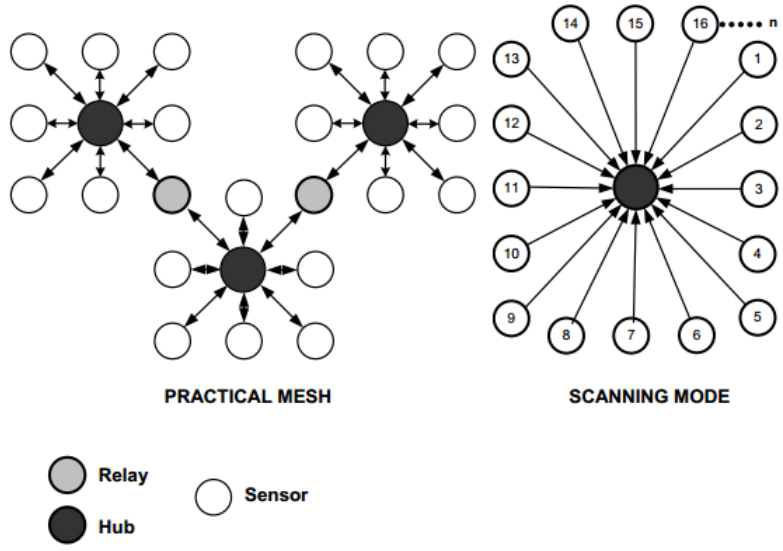

Figure 3: Example ANT Topologies

2.3.2 ANT Channels

Most of the available channel types are bidirectional, but the ANT protocol still differentiates between master and slaves nodes. While a master nodes mostly sends data and a slave nodes mostly receives data, the slave still has the ability to response to the incoming data.

The ANT protocol knows 125 different channels, each 1 MHz wide. Each Channel can support a data rate of 1 Mbps and up to 65533 nodes. To avoid interference between the channels isochronous self adjusting TDMA technology is used, this allows the ANT nodes to change

the transmit timings and even the frequency that is being used for the current channel. This is completely handle by the ANT protocol and thus the user has no control over the process.

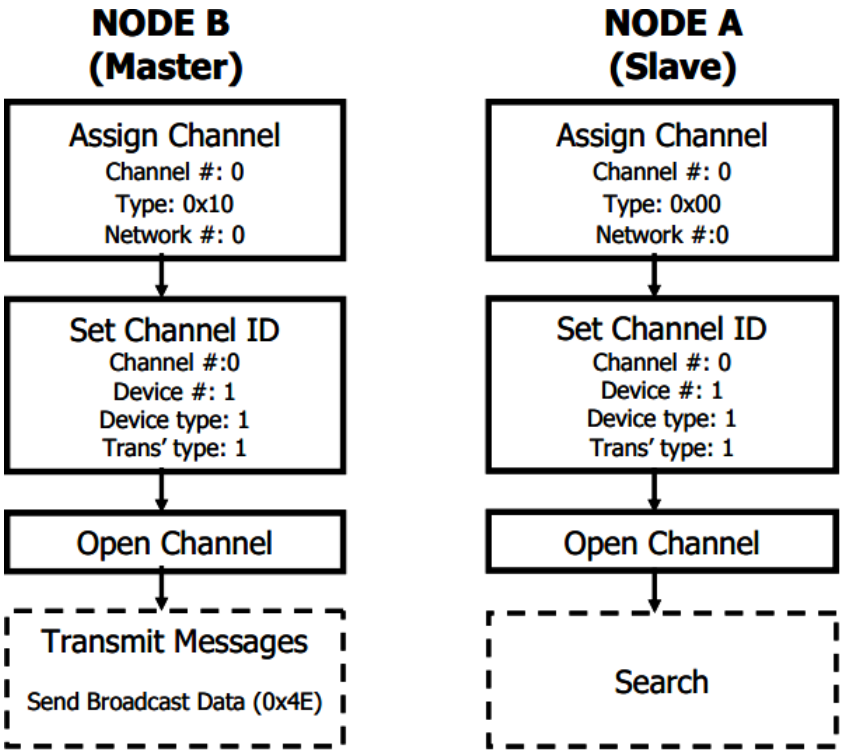

Figure 4: ANT Channel Establishment

Figure 4 shows the procedure to open up a channel between two ANT nodes. In the first step the Channel configuration is set, most important here is the channel type: 0x00 means that the node is a slave and 0x10 means that the node acts as a master. The next step is to set the Channel ID, here the slave nodes needs to set the values of the master device to correctly connect. It is however possible to set wildcard values, which will allow the slave to any device sending on the same frequency. Before the final step, it is optionally possible to set the frequency of the channel and the message period. These steps are however not mandatory, since ANT defaults to fixed values for these: 2466 Hz transmission frequency and a message period of 8192. The last step is the opening of the channel itself, here care should be taken that the master opens the channel before the slave devices opens it.

The ANT protocol distinguishes between 2 different Channel types:

Independent Channels

Independent Channels are used if there is only one node, which is transmitting data. There is no limit of the amount of slave devices, which receive the messages being send out. Furthermore the message being send out is broadcast to all nodes, it is not possible to only address a specific node.

Shared Channels

Shared Channels are used if there are more than one node which sends data. This type of channel is made possible by the use of a Shared Channel Address, however this reduces the amount of data that can be transmitted at a time. All ANT nodes still receive every message, but only forward messages which have a matching address. The

Channel master can decided to either use one or two bytes as the address, which allows for either 255 or 65535 slave devices in the same channel.

2.3.3 ANT Communication

The ANT protocol supports 3 different data types: broadcast, acknowledge and burst. The data type is not part of the channel configuration and thus channels are able to use any combination of data types. The only exceptions are unidirectional channels, which can only ever send broadcast data. The different types of data types differ in how the data is handled and transmitted.

- Broadcast data
  Broadcast data is the most basic datatype and the default. To start a broadcast transmission, the command needs to be issued only once, since the last send packet is periodically resend as a broadcast. Even if the last packet was part of a burst or acknowledge transmission. Figure (xyxcy) shows, how each broadcast transmission is aligned to the channel message period. Since there is no answer from the receiving node, it is not possible to determine if the packet was transmitted correctly or not.
- Acknowledge data
  Acknowledge data can be used to make sure a node received a transmitted packet. After the reception of a acknowledge packet, the node will send a message back to the sender. Acknowledge data can only be used with bidirectional channels, however both the master and the slave can use it. Figure (Xyxy) shows that, just like broadcast acknowledge data is always aligned to a time slot, but the answer from the receiver is send immediately back to the sender.
- Burst data
  Burst data provides a method to quickly transmit large amounts of data. This is achieved by ignoring the normal channel time slots and sending the packets right after each other. This allows for a transmission rate of up to 20 kbps, much higher than other data transmissions types. Similar to ackownledge data at the end of the transmission the sender is informed if the transfer failed or succeeded. The drawback of the methods is, that burst data is prioritized over all other transmissions and can interrupted other transmissions over the same channel.

2.3.4 ANT messages

In the ANT protocol each message has the in Figure 5 specified basic format. Each message

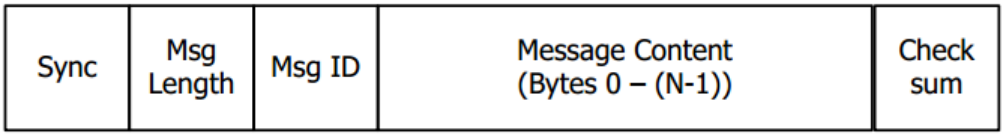

Figure 5: Ant message structure

starts with a special Sync-Byte and ends with a checksum, which is calculated by xoring all previous bytes. The Msg Length byte is the number ob Message Content bytes. The Msg ID byte specifies which kind of data is contained in the message. The ANT protocol also provides

an extended message format, which allows to attached further information to each message. The maximum length of the message content is 8 for each of the three data types.

3 Evaluation of SHAMPU

In order to assess the capabilities of SHAMPU we plan and run experiments designed to evaluate the SHAMPU framework according to the following use-cases:

- **Scheduled Data-transmission**
  In order for SHAMPU to work as a debugging and logging platform, the base-station needs to periodically receive data from all the nodes in the network. ANT provides two different data types which can be used to transport data in this case: Broadcast data and Acknowledge data. The Experiments designed for this use case try to determine the maximum throughput for each of the two transfer modes.
- **Unscheduled Data-transmission**
  There are several possible cases, where its not feasible to use a scheduled data-transmission. Instead the burst mode must be used, which allows to transmit data at a much higher rate. This use-case covers several different scenarios, e.g.:
  - Reprogramming of a node: SHAMPU is able to reprogram the attached node. For this, it needs to receive a new firmware, which can be a several hundred kB big.
  - SHAMPU RAM-Dumps: SHAMPU has 128kB of RAM, which can be used to collect data during an experiment. At the end the complete memory needs to be transmitted back to the base-station.
- **Communication Range**
  Since SHAMPU is architecture independent it can be used in different situations. Thus it is important to know, how far away from the base station the nodes can be placed. In addition, since SHAMPU tries to be energy efficient, it might be viable to reduce the range for smaller setups to save even more energy.

For this we designed different experiments, which test one or more of the described categories. The following section describes the experiments according to the following template:

Description
A description of the experiment and the the category being evaluated.

Use-Case
The use-case the experiment tries to test.

Network Topology and Configuration
A diagram of the network topology in which the experiment is run and the configuration of the complete network. Default values are not listed.

Testing methodology
A description on how the experiment is performed.

Result
The results of the experiment and any additional collected data.

3.1 Common Experiment parameters

If not otherwise noted in the experiment description each experiment were run in the Mobility lab of the Networked embedded Systems group at the University Duisburg-Essen (SA 327) and run with the following settings:

Device Number	33
Device Type	1
Transmission Type	1
ID_CHAN1	0
FREQ_CHAN1	2466 Hz
ID_CHAN2	1
FREQ_CHAN2	2477 Hz
Message Period	8192
min_Channel_Period	0x00A4
max_Channel_Period	0xFFFF

Table 1: ANT default configuration

Also there is an important difference between the message period used in the pseudo code and the frequency used in the figures, which display the results. The period describes the size of the gap between two messages, then frequency describes how many messages are send in 1 second. The channel period p can be used to calculate the frequency of the time slots $f_t = \frac{32678}{p}$.

3.2 Experiment 1: Broadcast Data Transfer between two nodes

Description

Broadcasting is one way to periodically transmit data between 2 or more ANT nodes. Since all broadcast packets are synchronized to a fixed time-slot, the data throughput can be increased by decreasing the channel period. The experiment itself is split into two parts. In the first part we try to determining the highest possible data throughput. Also we try to determine if the channel period has an effect on the time it takes for a slave node to find and join an existing channel. The second part is a test of the highest found data throughput, here we try to evaluate if there are any variations of the data throughput over a much longer interval.

Use-Case

Scheduled Data-transmission

Network Topology and Configuration

Diagram for the experiment: 2 nodes

```
channelPeriod = max_Channel_Period
while (channelPeriod >= min_Channel_Period) {
  ANT_SetChannelPeriod(ID_CHAN1, channelPeriod)
  ANT_OpenChannel(ID_CHAN1, ANT_Bidirectional_Master)
  ANT_SendBroadcastData(ID_CHAN1, [0x01, 0x02, 0x03, 0x04])
  wait_for_user_input()
  ANT_CloseChannel(ID_CHAN1)
  channelPeriod = decreaseChannelPeriod()
}
```

Source code 1: Broadcast data single channel (Master)

```
channelPeriod = max_Channel_Period
while (channelPeriod >= min_Channel_Period) {
  for (i in 0..9) {
    ANT_SetChannelPeriod(ID_CHAN1, channelPeriod)
    ANT_OpenChannel(ID_CHAN1, ANT_Bidirectional_Slave)
    count = 0
    for (100 seconds)
      if (receivedPacket() == ANT_BROADCAST_DATA)
        count++;
    print (count * 8 / 100) + " Bytes per second"
    wait_for_user_input()
    ANT_CloseChannel(ID_CHAN1)
  }
  channelPeriod = decreaseChannelPeriod()
}
```

Source code 2: Broadcast data single channel (Slave)

Network Topology and Configuration

The 2 nodes are placed right next to each other.

Testing methodology

Node A acts as the master and node B as the slave. For both nodes the channel period is set to the highest value and the channel is opened. Node B records how long it took to join the channel and how many bytes it received over a 100s interval. The measurement is repeated 10 times and the average values are saved. Then the channel period is decreased and the process repeated. For the second part, the channel period is set to the value, which achieved the highest speed. The experiment is then left running for a period of 10 hours, from 9 AM to 7 PM in one continues run.

Result

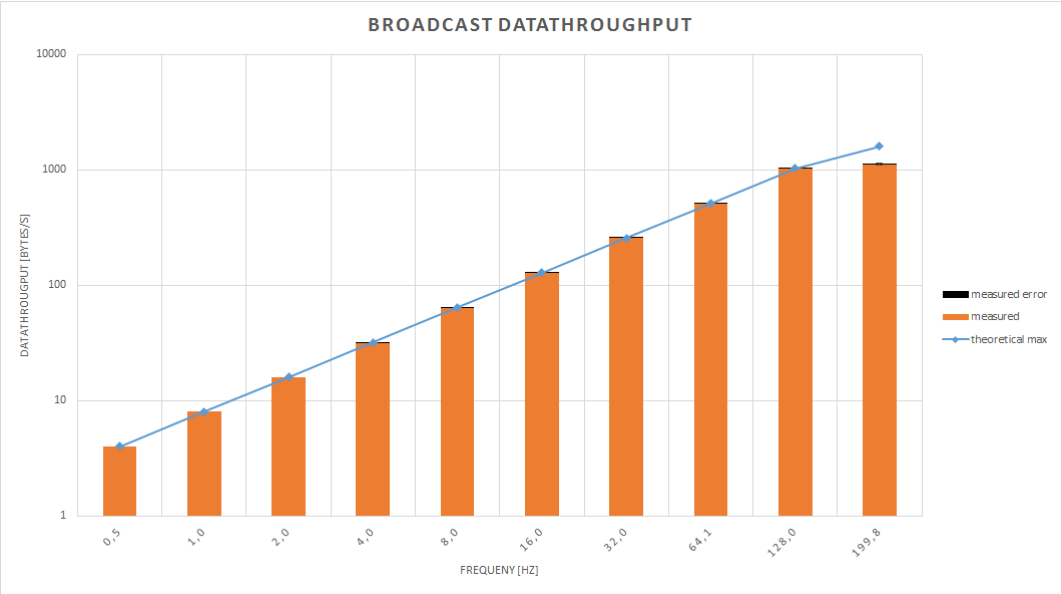

Figure 6: Broadcast datarate

Figure 6 shows the achieved transmission speeds for the different frequencies. Up a frequency of 128 Hz the measured data rate, matches the expected theoretical maximum rate. For the highest supported frequency (199.8 Hz), the measured data rate is much lower the maximum rate. Even accounting for transmission errors the difference can't be fully explained.

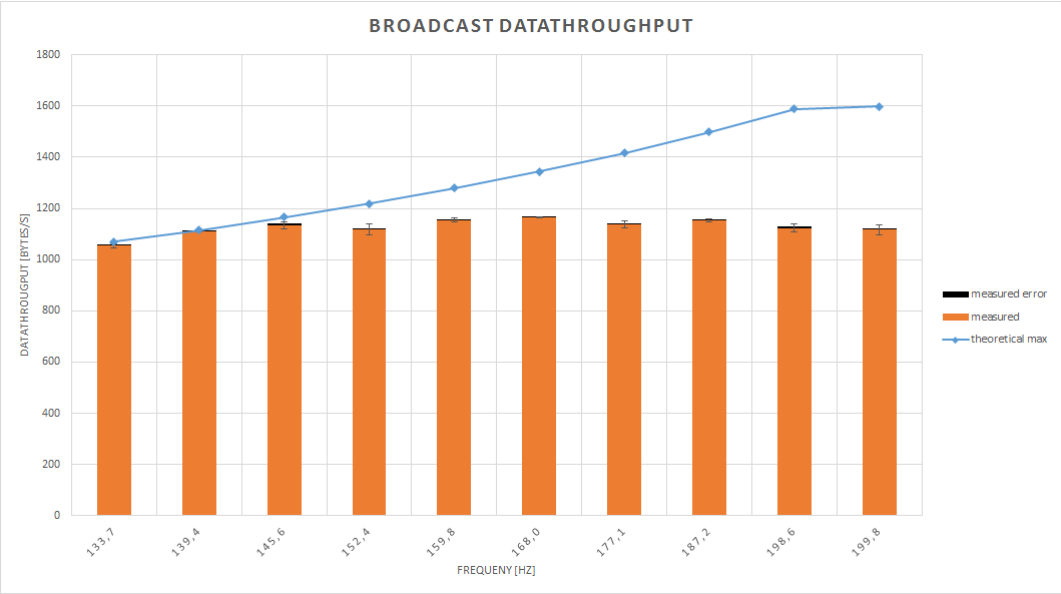

Figure 7: Broadcast datarate

As seen in figure 7 the measured data rates for frequencies above 140 Hz all fall short of the expected values. However the average of these frequencies is consistent at around 1100 Bps. The experiment was repeated multiple times, at different times and places and thus an environmental factor can be discarded. That means that the reason for the upper limit has to be with the hardware itself. See section 4.2 for a discussion about possible reasons for this upper limit.

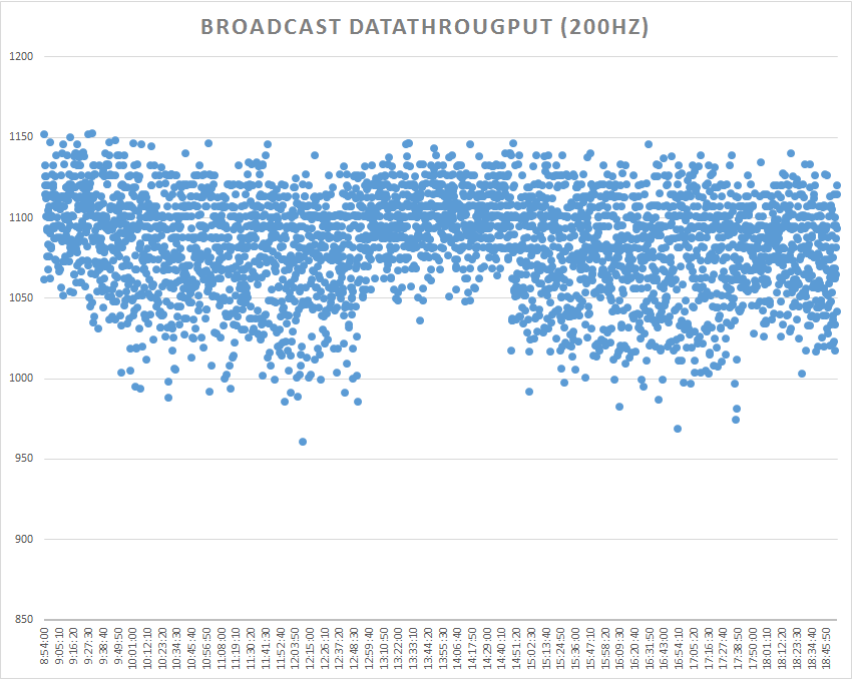

Figure 8: Broadcast datarate

Figure 8 shows the transmission speed for the highest supported frequency 199.8 Hz over time. The results show, that the data throughput stays pretty consistent with a std. deviation of only 30 Bps, or 2.7%. Interesting to note is the time period from 1:00PM to 2:50 PM, here the deviation from the average is much smaller. The exact reason for this in unknown, since during this the environment of the test setup didn't change at all.

3.3 Experiment 2: Broadcast Data Transfer between multiple nodes

Description

In Experiment 1 we determined the channel period, which allows for the maximum throughput. In this experiment we try to determine, how this value is affected by the amount of channels in the network. SHAMPU needs two channels, to work correctly: One channel, which sends data from the base station to the nodes in order to control them, and another channel with which the nodes can send information and debugging informations back.

Use-Case

Scheduled Data-transmission

Network Topology and Configuration

Diagram for the experiment: 2 nodes

```
channelPeriod = max_Channel_Period
while (channelPeriod >= min_Channel_Period) {
      ANT_SetChannelPeriod(ID_CHAN1, channelPeriod)
      openChannel(ID_CHAN1, ANT_Bidirectional_Master)
      ANT_SendBroadcastData(ID_CHAN1, [0x01, 0x02, 0x03, 0x04])
          ANT_SetChannelPeriod(ID_CHAN2, channelPeriod)
          openChannel(ID_CHAN2, ANT_Bidirectional_Slave)
  count = 0
  for (10 seconds)
    if (receivedPacket() == ANT_BROADCAST_DATA)
      count++
  print (count * 8 / 10) + " Bytes per second"
  wait_for_user_input()
  ANT_CloseChannel(ID_CHAN1)
  ANT_CloseChannel(ID_CHAN2)
  channelPeriod = decreaseChannelPeriod()
}
```

Source code 3: Broadcast data transfer two channels (Master)

```
channelPeriod = max_Channel_Period
while (channelPeriod >= min_Channel_Period) {
  ANT_SetChannelPeriod(ID_CHAN1, channelPeriod)
  openChannel(ID_CHAN1, ANT_Bidirectional_Slave)
  ANT_SetChannelPeriod(ID_CHAN2, channelPeriod)
  openChannel(ID_CHAN2, ANT_Bidirectional_Master)
  ANT_SendBroadcastData(ID_CHAN2, [0x01, 0x02, 0x03, 0x04])
  count = 0
  for (10 seconds)
    if (receivedPacket() == ANT_BROADCAST_DATA)
      count++
    print (count * 8 / 10) + " Bytes per second"
  wait_for_user_input()
  ANT_CloseChannel(ID_CHAN1)
  ANT_CloseChannel(ID_CHAN2)
  channelPeriod = decreaseChannelPeriod()
}
```

Source code 4: Broadcast data transfer two channels (Slave)

Network Topology and Configuration

The 2 nodes are placed right next to each other.

Testing methodology

In this experiment each nodes acts as a master for a different channel. Node A is the master for Channel 0 and Node B is the master for Channel 1. The measurements itself are identically to experiment 1, except that the data is recorded on both Nodes. The channel period is decreased until the data throughput drops, or the connection can no longer be established.

Result

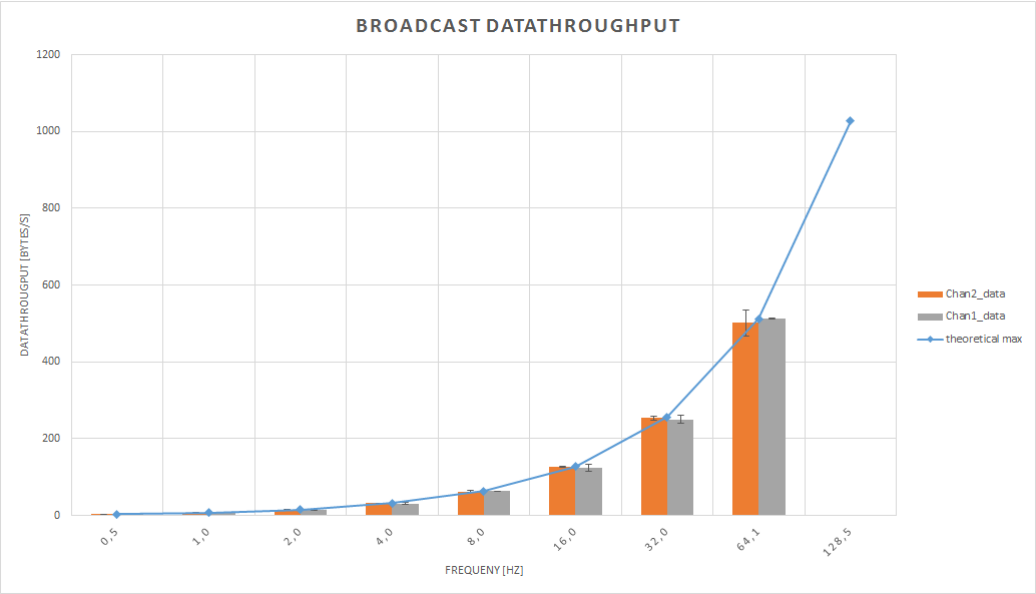

Figure 9: Broadcast data through - 2 channels

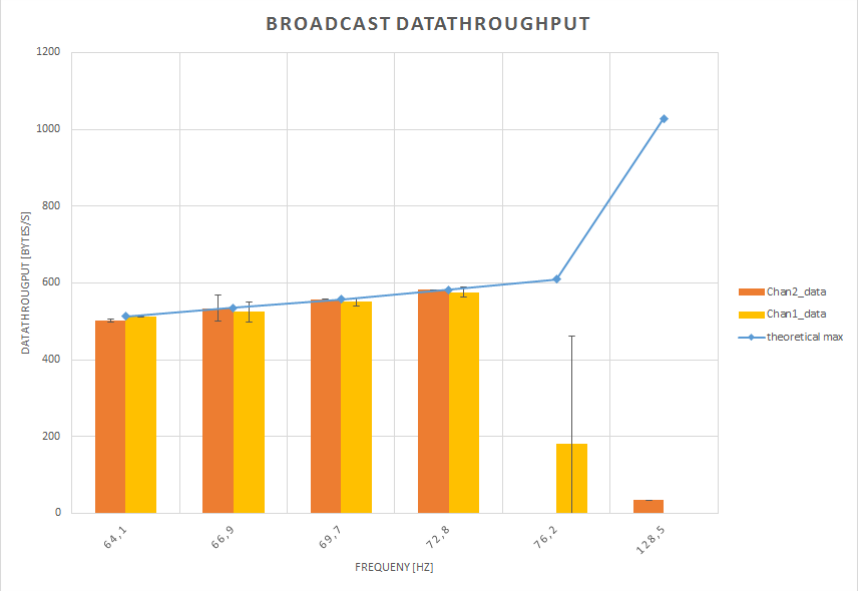

Figure 10: Broadcast data through - 2 channels

Figures 9 shows the results of the measurements, the left bar displays the throughput for channel 0, the right bar displays the throughput for channel 1. The blue line show the theoretical maximum of the data throughput, for the given frequency. Up to a frequency of 64 Hz the data throughput increases with the frequency for both channels. However there is a complete drop at 128 Hz. Figure 10 shows that somewhere between 72 Hz and 76 Hz the data throughput begins to cut off. The combined data rate for two channels in thus around 1160 Bps, which is in line with the result from experiment 1. From this data we can assume, that the whole available datarate of 1160 Bps, is simply split among all existing channels.

3.4 Experiment 3: Acknowledge Transfer delay

Description

In this experiment we try to determine the time it takes for a node to received and acknowledge a transmitted packet. The value is important, to be able to determine the reaction time of SHAMPU to commands send by the base-station, for example to set up a separate channel for a burst transmission.

Use-Case

Scheduled Data-transmission

Network Topology and Configuration

Diagram for the experiment: 2 nodes

```
channelPeriod = max_Channel_Period
while (channelPeriod >= min_Channel_Period) {
  ANT_SetChannelPeriod(ID_CHAN1, channelPeriod);
  ANT_OpenChannel(ID_CHAN1, ANT_Bidirectional_Master);
  duration = 0.0
  for (i in 0..10) {
    ANT_SendAcknowledgedData(ID_CHAN1, [0x01, 0x02, 0x03, 0x04])
    start = getTime()
    wait_for_ack()
    print (getTime() - start) + " s"
  }
  ANT_CloseChannel(ID_CHAN1)
  channelPeriod = decreaseChannelPeriod()
}
```

Source code 5: Acknowledge data delay (Master)

```
channelPeriod = max_Channel_Period
while (channelPeriod >= min_Channel_Period) {
  ANT_SetChannelPeriod(ID_CHAN1, channelPeriod);
  ANT_OpenChannel(ID_CHAN1, ANT_Bidirectional_Slave);
  wait_for_user_input();
  ANT_CloseChannel(ID_CHAN1);
  channelPeriod = channelPeriod >> 1;
}
```

Source code 6: Acknowledge data delay (Slave)

Testing methodology

Node A acts as the master and node B as the slave. For both nodes the channel period is set to the highest value and the channel is opened. Node A then sends a total of 10 acknowledge messages, and measures how long it takes until it receives the acknowledge signal. The channel period is then decreased and the experiment repeated.

Result

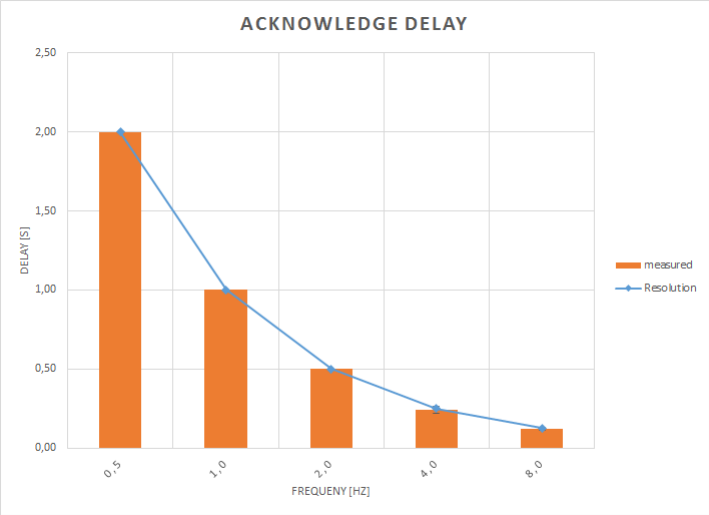

Figure 11: Acknowledge delay

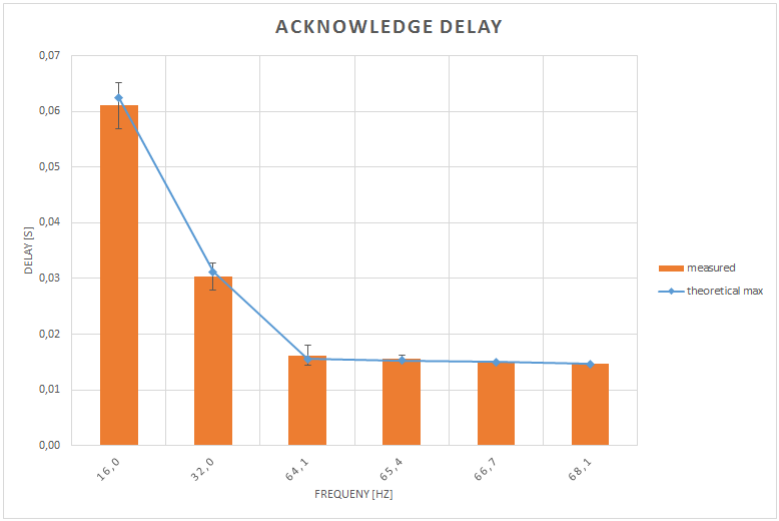

Figure 12: Acknowledge delay

Figure 11 and Figure 12 show the delays for the tested frequencies. The blue line shows the size between two timeslots, and can act as a resolution for each measurement. For the lower frequencies the results are skewed, since each acknowledge packet is aligned to a time-slot and thus the biggest part of the delay is waiting for the next time-slot. For frequencies greater than 64 Hz the data is much more useful, since the resolution is lower than the measured values. The delay for those frequencies is around 18 ms, which no notable difference between 128 Hz and 200 Hz. Since the procedure if the same for the higher channel periods, it can be assumed that the delay is the same as for the lower channel periods.

3.5 Experiment 4: Acknowledge Data Transfer between two nodes

Description

This experiment is almost the same as experiment 1. The main difference is that we use acknowledge data instead of broadcast. It is also important to note that the master records how many successful packets are transmitted. The main goal of the experiment is, to see if the data throughput is decreased compared to broadcast data, especially for smaller channel periods.

Use-Case

Scheduled Data-transmission

Network Topology and Configuration

Diagram for the experiment: 2 nodes

```
channelPeriod = max_Channel_Period
while (channelPeriod >= min_Channel_Period) {
  ANT_SetChannelPeriod(ID_CHAN1, channelPeriod)
  ANT_OpenChannel(ID_CHAN1, ANT_Bidirectional_Master)
  count = 0
  for (10 seconds) {
    ANT_SendAcknowledgedData(ID_CHAN1, [0x01, 0x02, 0x03, 0x04])
    wait_for_ack()
    count++
  }
  print (count * 8 / 10) + " Bytes per second"
  ANT_CloseChannel(ID_CHAN1)
  channelPeriod = decreaseChannelPeriod()
}
```

Source code 7: Acknowledge data transfer (Master)

```
channelPeriod = max_Channel_Period
while (channelPeriod >= min_Channel_Period) {
  ANT_SetChannelPeriod(ID_CHAN1, channelPeriod)
  ANT_OpenChannel(ID_CHAN1, ANT_Bidirectional_Slave)
  wait_for_user_input()
  ANT_CloseChannel(ID_CHAN1)
  channelPeriod = channelPeriod >> 1
}
```

Source code 8: Acknowledge data transfer (Slave)

Testing methodology

The testing methodology is the same as experiment 1a, except that the master sends acknowledge messages and waits for the slave to confirm the successful transmission before sending the next packet.

Result

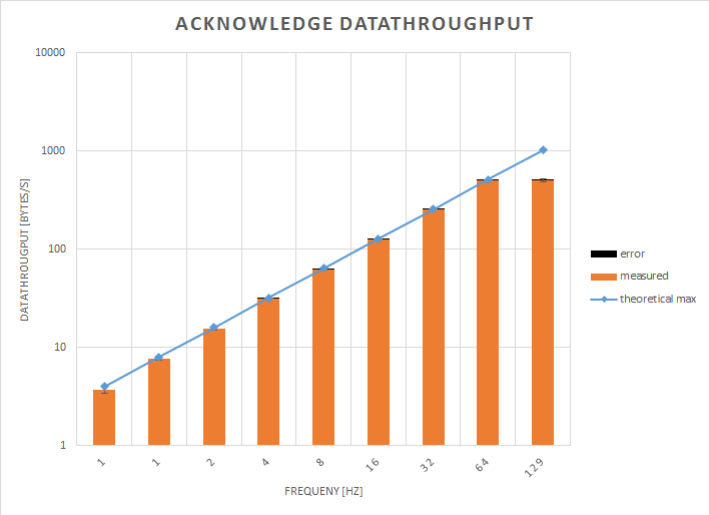

Figure 13: Acknowledge datarate

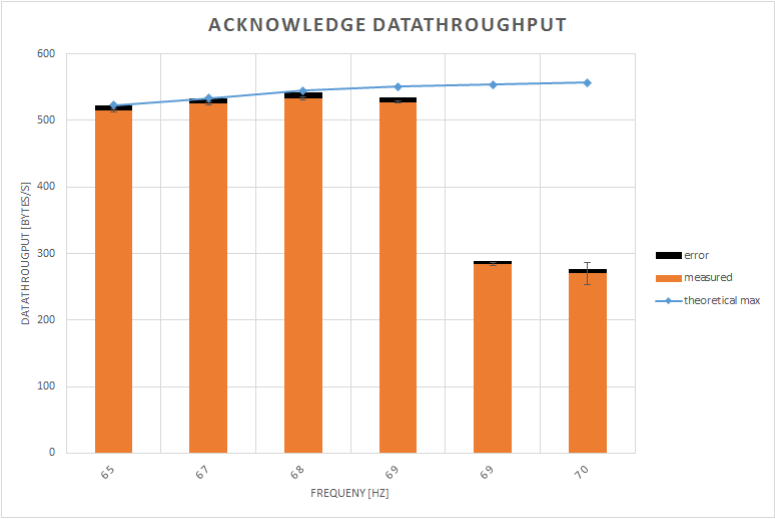

Figure 14: Acknowledge datarate

Figure 13 shows the measured transmission speeds for the different tested frequencies. For the lower frequencies, the values align very well the with maximum data rate. Notable however is the sharp drop off at 129 Hz. Figure 14 shows that the drop off in the data throughput rate happens around 69 Hz. This result can be explained with the results of experiment 3. The delay of an acknowledge message is roughly 18 ms. The problem is that ANT retransmits the last sent packet as a broadcast packet, if the frequency for this retransmission is too high it can interfere with the acknowledge message that the slave sends back to the master. For example at 200 Hz the message is broadcast every 5 ms resulting in too much data for the channel to handle. The transmission is thus interrupted. Since there is no increase of the data throughput between 64 Hz and 128 Hz, but a huge drop off at 69 Hz it can be assumed that somewhere around 69 Hz the

channel is at the maximum data throughput of around 1100 Bps. Again see section 4.2 for a discussion about possible reasons for this upper limit.

3.6 Experiment 5: Burst Data Transfer between two nodes

Description

Burst data transmissions make it possible to drastically increase the throughput rate. This allows a SHAMPU base station to quickly transmit a new firmware to a node or a node to dump its RAM back to the base-station. According to the specification rates of up to 20 kbps can be achieved. To fully utilize this speed a baud rate of 50000 is needed. Since we are using 19200 baud we expect the maximum speed to be less than 20 kbps. Furthermore we also try to determine, if the size of the burst transfer has an impact on the speed, since longer burst can disrupt communications on other channels.

Use-Case

Unscheduled Data-transmission

Network Topology and Configuration

Diagram for the experiment: PC with ANTUSB-m stick and 1 Node

```
size = START_SIZE
ANT_OpenChannel(ID_CHAN1, ANT_Bidirectional_Slave);
while (size >= END_SIZE) {
  for (i in 0..10) {
    wait_for_first_burst_packet()
    start = getTime()
    wait_for_last_burst_packet()
    print (getTime() - start) / (size - 1) " Bytes per second"
  }
  size = 2 * size
}
```

Source code 9: Burst data transfer (Slave)

Testing methodology

Since the burst transmission mode, of our ANT-library is not working correctly (see section 4.3) we use an ANTUSB-m Stick, which acts as a master. Node B is a normal base station and receives the burst transfers. On the master side, the program ANTWareII [XYXYXY] is used to create the channel and then send burst with different sizes. Each size is send 10 times and the values recorded and averaged. The size is then doubled and the experiment repeated. Note should be take that it is not possible to incorporate the first burst packet into the calculation, since the start of the burst can only be determined, after the ANT chip already received the first packet.

Result

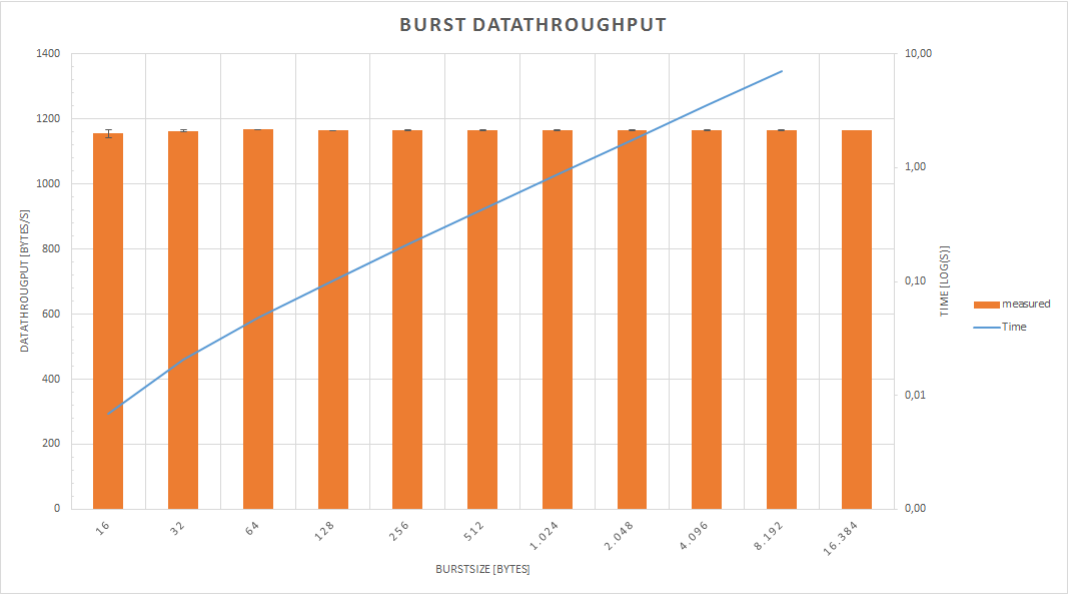

Figure 15: Burst datarate

Figure 15 shows the achieved data rates for the different packet sizes and also the length of the transfer. As seen the values all hover around the same value of 1165 Bps. This is about the same value which can be achieved with the other two types of data. Again see section 4.2 for a discussion about possible reasons for this upper limit.

3.7 Experiment 6: Maximal communication Range

Description

In this experiment we try to determine the correlation between the maximum range and the power setting of the ANT radio. One of SHAMPU's advantages is the low power consumption, so it might be possible to further reduce the power consumption by decreasing the power level of the ANT radio, especially in smaller environments where there is no need for such a long range. According to the datasheet the maximum range for communication is 30m, however the ANT documentation doesn't specify for which power setting this range can be achieved.

Use-Case

Communication Range

Network Topology and Configuration

Diagram for the experiment: 2 nodes x meter apart. with distances from 0.5 m to 30m (floor in SM 3xx is 27m long) A node is master, B node is slave. Node B is set up on a movable platform.

Testing methodology

At the beginning of the experiment Node A and B are placed right next to each other. Node A acts as a master and keeps broadcasting the same message. Node B is the slave and tries to connect to the channel. If the connection is successful, the distance between the two nodes is increased by 0.4 meters. This process is repeated until Node B no longer can connect to the channel and the connection times out, this happends after 30s of searching. At this point the distance is no longer increased, but decreased until Node B ss able to successfully connect to the channel. The whole experiment is then repeated for each available power setting.

```
for (pSetting in Available_PowerSettings) {
  ANT_SetTransmitPower(pSetting)
  openChannel(ID_CHAN1, ANT_Bidirectional_Master)
  ANT_SendBroadcastData(ID_CHAN1, [0x01, 0x02, 0x03, 0x04])
  wait_for_user_input();
  closeChannel(ID_CHAN1);
}
```

Source code 10: max communication range (Master)

```
distance = 0
stopInc = false
loop {
openChannel(ID_CHAN1, ANT_Bidirectional_Slave)
wait_until(received_Packet  == ANT_BROADCAST_DATA ||
           received_Packet == ANT_MESSAGE_EVENT_RX_SEARCH_TIMEOUT)
if (stopInc) {
  if (wasTimeout) distance -= .5
    else print "Connection found : " + distance
  } else {
    if (wasTimeout) {
      stopInc = true; distance -= .5
      print "Connections lost : " + distance
    } else distance += .5
  }
     closeChannel(ID_CHAN1);
}
```

Source code 11: max communication range (Slave)

Result

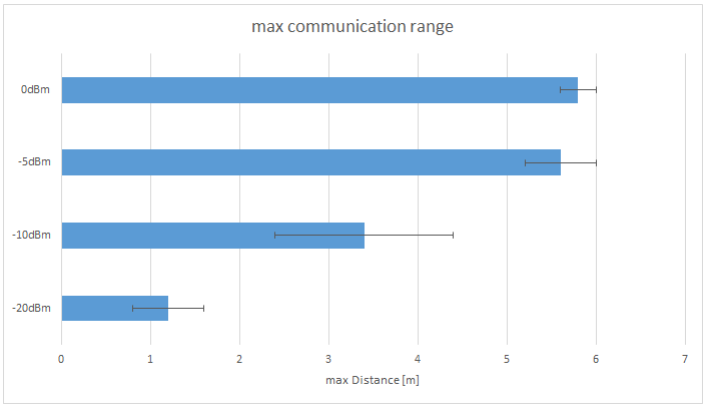

Figure 16: max comunication range

Figure 16 shows the transmission range for each power setting. The results of this experiment seem disappointing, since we didn't get close to the maximum range of 30m. However the ANT documentation states that the maximum range of 30m can only be achieved in "optimal conditions". It is possible for many different factors to interfere with the ANT signal, such as multiple 802.11 networks in the vicinity, or even the plastic case of the base station.

4 Conclusion

TODO

4.1 Summary

4.2 max DataThroughput

All our experiments seem to confirm that there is a hard limit for the maximum data throughput, which can be archived with our current setup. The ANT AP1MxIB supports a maximum frequency of 200 Hz, so we would expect to see a maximum transmission rate of 1600 Bps, however we miss this limit by around 500 Bps or 30%. For a burst transmission there should be upto 20 kbps of throughput, again we only achieved around 1100 Bps, a loss of 1400 Bps or 56%.
To eliminate environmental influences, the experiments were repeated in different locations and at different times of the day and night. Since the throughput didn't change noticeable, it can be concluded that there aren't any easily eliminated environmental factors, which affect the maximum rate. That means that the reason for the lower throughput has to do with the hardware or software.

If we exclude environmental factors that leave there possible reasons:

- **RS-232 Connection**
  The base station is connected over a serial connection to a PC. For the speed we chose 19200 baud, which should be more then enough for a maximum message rate of 200 Hz. For the burst mode however, this is one limiting factor. With 19200 baud, the theoretical maximum is at 1920 Bps. This explains some, but not all of the measured difference.
- **ANT API**
  The software we use is not officially support by ANT and it is thus possible that there are some bugs, which have an impact of the performance. Expect for the missing burst modus (see 4.3), there were no unusual measured values during the experiments. If there is a bug, it might be hard to find, without rewriting all the experiments and running them with the offical ANT library [XYXYX]
- **ANT AP1MxIB**
  The ANT chip we use is fairly old... blablabla messages queues blablabla might not even be possible to achieve the promissed speeds hard to test, since blackbox.

4.3 Future Work

Due to time and the limited hardware availability we were not able to fully explore and evaluate the possibilities of ANT in the SHAMPU network. Especially these three areas should be revisted:

- **Power consumptions**
  Each experiment tries to find the maximum of either the data throughput or the communication range. We did not measure how the power consumption changes, if, for example the message period is reduced. Since SHAMPU tries to be very low-powered this is an important measurement, for the decision if SHAMPU can be used for a specific application or not. The datasheet of the ANT AP1MxIB module lists already several values [XYXYY]: Here the maximum current draw seems to be around 5 mA for a continuous burst transmission, and around 40 µA for a normal broadcast operation.
- **Burst mode**
  We use a custom library, which provides an API to interface with the ANT-Chip. For this thesis an atempt was made, to add the missing burst transfer mode. However due to time constrains we were unable to get the mode correctly working. Burst packets need to have a precise timing and due to the blackbox nature of ANT is it hard to debug.
- **Shared channels**
  Due to missing hardware, we were only able to test the communication between two nodes. Thus shared channels could not be tested, however we except the available data throughput to be in proportion with the results from our experiments. ANT used up to 2 bytes from the payload to specify the address of the receiver, which means that we expect to loses about 12.5% to 25% of throughput if shared channels are in use.

Bibliography

[1] R. Lim, F. Ferrari, and M. Zimmerling, "FlockLab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," *Proceedings of the 12th ...*, pp. 153–165, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2461402

[2] P. Sommer, "Minerva : Distributed Tracing and Debugging in Wireless Sensor Networks."

[3] O. Rensfelt, F. Hermans, C. Ferm, P. Gunningberg, and L. Larzon, "Sensei-UU: A Nomadic Sensor Network Testbed Supporting Mobile Nodes," 2009. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Sensei-UU+:+A+Nomadic+Sensor+Network+Testbed+Supporting+Mobile+Nodes#0

[4] T. Moser and L. Karrer, "The EventCollector Concept," *Topology*.

[5] O. Kasten and M. Langheinrich, "First Experiences with Bluetooth in the Smart-Its Distributed Sensor Network The Smart-Its Prototype," *System*, no. 00, 2000.

[6] H. Smeets, C.-Y. Shih, T. Meurer, and P. J. Marrón, "Demonstration Abstract: A Lightweight, Portable Device with Integrated USB-host Support for Reprogramming Wireless Sensor Nodes," in *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, ser. IPSN '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 333–334. [Online]. Available: http://dl.acm.org/citation.cfm?id=2602339.2602401

[7] Dynastream Innovations Inc., "ANT Message Protocol and Usage," pp. 1–134, 2013.