Leiden University

Computer science

Constructing an open-source toolchain
and investigating sensor properties
for radio tomography

Name:	Tim van der Meij
Student number:	1115731
Date:	June 22, 2014
1st supervisor:	Walter Kosters (LIACS)
2nd supervisor:	Joost Batenburg (CWI & MI)

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

In this bachelor thesis, we present an open-source toolchain for radio tomographic imaging (RTI) using radio signals with approximately the same frequencies as WiFi signals, as well as practical guidelines for setting up an effective sensor network. RTI is a technique for wirelessly detecting or tracking objects or persons in a sensor network using radio signals. It is an inverse problem whereby measurements need to be converted to information about objects or persons in the network. After discussing how to set up the wireless sensor network, we will go into detail on how the toolchain has been constructed and which experiments have been performed to determine the network setup guidelines. Measurements obtained from the sensor network allow us to make a reconstruction image of the network using imaging algorithms. One of the results of our research is that the shape of the network appears to have a large impact on the effectiveness of the sensor network.

Keywords: radio tomographic imaging, sensor network, toolchain, wireless, WiFi

Contents

1 Introduction **4**
- 1.1 Motivation and applications . . . 4
- 1.2 Research questions . . . 5
- 1.3 Research group . . . 5

2 Related work **5**
- 2.1 The multi-Spin protocol . . . 5

3 Setting up the network **6**
- 3.1 ZigBee protocol . . . 6
- 3.2 CC2530 Development Kit . . . 6
- 3.3 Multipath propagation and interference . . . 7

4 Developing the open-source toolchain **8**
- 4.1 Spin packets . . . 8
- 4.2 Compilation using SDCC . . . 9
- 4.3 Flashing the nodes . . . 10
- 4.4 Tools . . . 11

5 Experiments **11**
- 5.1 Antenna direction . . . 13
- 5.2 Symmetric properties . . . 14
  - 5.2.1 Fixed node positions . . . 15
  - 5.2.2 Swapped node positions . . . 16
- 5.3 Corner nodes versus non-corner nodes . . . 16
- 5.4 Walking versus running . . . 17
- 5.5 Circle-shaped network versus square network . . . 19

6 Conclusions and future work **20**

Glossary **21**

References **21**

Appendix **23**

1 Introduction

Radio tomographic imaging (RTI) is an emerging technique that is used to detect or track objects or persons in a sensor network using radio signals. It is an inverse problem whereby we use gathered measurements to derive information about the person(s) or object(s) inside the sensor network. Once we have gathered the measurements, an imaging algorithm can solve this problem mathematically and output a reconstruction image. For this bachelor thesis, we have set up a sensor network with nodes that emit radio signals at approximately the same frequencies as WiFi signals (2.4 GHz). A USB dongle will collect the measurements for digital processing. For the reconstruction it is important to calibrate the system by creating a baseline measurement, which will serve as a correction for new measurements to only detect new objects or persons in the network.

1.1 Motivation and applications

There are several reasons why one would want to use RTI with radio signals instead of other detection methods like cameras or infrared solutions. For instance, cameras fail to perform a proper detection in the dark. Infrared cameras cannot see through large obstructions such as walls or large objects, like trucks in a garage. Radio frequency (RF) signals have the advantage that they can travel through many of these obstructions and they work during day and night time. On top of that, detection/tracking with radio signals is harmless and it is practically impossible to identify a person [1].

Due to the advantages of RF signals and the fact that the nodes are relatively cheap, RTI has many potential applications. The first applications that come to mind are security applications. One could place RF nodes around a warehouse and get a reconstruction of the objects and persons inside. An alarm can be triggered if movement is detected during night time. A second application can be found in health care. Nodes can be placed around the apartment of an elderly person and one could mark particular regions of interest in the house. If someone is at one place in the house for a long time, that might indicate that the person fell and cannot stand up on his own.

A third application is gaining insight in behaviour of customers in a store. Recently, some stores in the Netherlands admitted to using WiFi tracking in their stores to get an idea of the behaviour of their customers [2]. WiFi tracking gave the stores some interesting insights, for example at which product stands customers were looking for a long time. That might indicate that the customers are interested in the product, but think that the price is too high. Specific discounts can then be offered to win the customers over. Another interesting question for a store is "When do I need to have more staff in the store?". However, the revelation caused concerns in the Netherlands because the measurements are not anonymous and the stores did not announce that they were tracking customers. MAC addresses and other phone data were collected with the measurements, which is not allowed if the customers have not given explicit permission to do so.

RTI could be of help here. Not only can the stores still get the insights they want, the measurements are also anonymous. It is practically impossible to identify the customers because no personal data is collected to make the reconstruction images (which contain only blobs representing persons because RTI cannot do much better right now). The nodes can be made small enough to not take too much space. An amplifier can be used on the nodes to increase the range of the signal.

1.2 Research questions

This thesis will answer the following research questions. How can we use the hardware and existing software to perform measurements for RTI purposes? How can we extend the existing software? How can we set up an effective sensor network given environmental properties and multipath propagation? Can we develop an open-source toolchain for RTI?

1.3 Research group

In order to research this interesting field, an RTI research group has been formed consisting of members from Leiden University as well as members from CWI Amsterdam. The research group consists of the following members (in alphabetical order):

- Joost Batenburg (CWI Amsterdam)
- Folkert Bleichrodt (CWI Amsterdam)
- Walter Kosters (Leiden University)
- Tim van der Meij (Leiden University)
- Alyssa Milburn (Leiden University)

The group is assisted by Tristan van Leeuwen and Willem Jan Palenstijn, both from CWI Amsterdam. This bachelor thesis is the result of four months of research and serves as the final work for the computer science bachelor program at Leiden University.

2 Related work

Most of the previous work has been done by researchers from the University of Utah [1]. They have graciously provided their own RF protocol called multi-Spin.

2.1 The multi-Spin protocol

The *multi-Spin protocol* is an RF communication protocol that specifies the order of transmission of the nodes in a sensor network and makes sure that the nodes are synchronized. Using the multi-Spin protocol, measurements on different frequency channels

can be performed (using channel hopping) [3].

The protocol divides time into rounds, cycles and slots. A round includes α TDMA cycles, with α being the number of frequency channels to measure on. Each TDMA cycle contains $\beta + 3$ slots, where β equals the number of nodes in the sensor network. The remaining three slots can be used to issue commands from the listener node to the RF sensor nodes. In every slot, only one RF sensor is broadcasting a packet, whereas the other nodes are only receiving packets [4].

The multi-Spin protocol works with sweeps. One *sweep* means that each node has broadcasted once and therefore that all links in the network have been used once. The order of transmission is defined by the ID of the nodes. This node ID is set when compiling the RF sensor software for each specific node. Texas Instruments' CC2530 hardware, which we have used among others for this project, can handle three sweeps per second.

3 Setting up the network

To build the sensor network, we have decided to use the CC2530 Development Kit from Texas Instruments [5]. The main reason for making this choice is that the CC2530 SoC is supported by the multi-Spin protocol. The CC2530 SoC operates on low voltage and communicates using the ZigBee protocol on frequencies around 2.4 GHz.

3.1 ZigBee protocol

The *ZigBee protocol* is a wireless standard for low-power wireless networks. One of the key features of the ZigBee protocol is the support for *mesh networks*, i.e., networks where all nodes are connected to each other. This is exactly what we need for the sensor network, which also consists of RF nodes that need to form a mesh network. The ZigBee protocol can also handle other topologies such as point-to-point connections [6]. ZigBee and WiFi can coexist without interference if we use many channels and a high RF output power.

3.2 CC2530 Development Kit

The CC2530 Development Kit consists of several components to set up a sensor network. Its main components are described below [7]:

- Two SmartRF05 evaluation boards (see Figure 1). These boards are used to flash the software onto the nodes and for debugging purposes. They both have an LCD and control component such as buttons and a joystick. Both can be connected to a PC via USB.
- Two CC2530 evaluation modules with antennas (see Figure 2). These smaller boards contain the CC2530 chips on which we can flash the multi-Spin RF sensor software. Each evaluation mode forms a node in the network.

- A CC2531 USB dongle (see Figure 3). This USB dongle also has a CC2530 chip and will be flashed with the multi-Spin listener node software. It will receive all Spin packets and make sure that we can process them on the computer.

Note that in order to make large sensor networks, we have bought additional evaluation modules, antennas and CC2530 battery boards. The battery boards are essentially circuit boards powered by batteries on which we can attach evaluation modules, thereby creating solid and independently powered nodes for the sensor network.

Figure 1: SmartRF05 evaluation board [7].

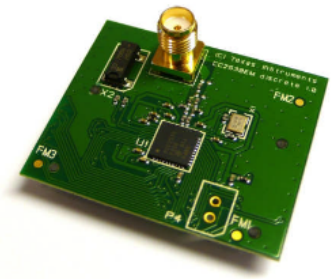


Figure 2: CC2530 evaluation module (without antenna) [7].

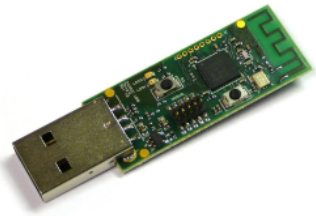


Figure 3: CC2531 USB dongle [7].

3.3 Multipath propagation and interference

The environment where the sensor network is placed is of great importance for the accuracy of the measurements. Especially indoor environments tend to add more noise to the measurements than outdoor environments. This is due to two factors: multipath propagation and interference. *Multipath propagation* is the effect where a sent radio signal arrives at the receiver using two or more paths. Reflection and refraction are two typical causes of multipath propagation. If this happens, *interference* can cause the received signal to either be weaker (destructive interference or fading, see Figure 5) or stronger (constructive interference, see Figure 4), depending on the amplitude and phase difference of the radio signals [8]. Both are considered to be noise: the original signal and therefore the measurement is distorted. Measurements in outdoor environments typically contain less distortion because there is less multipath propagation. Multipath propagation in indoor environments is caused primarily by walls, desks and heavy equipment, whereas in outdoor environments there are not many objects that cause signals to be reflected or refracted.

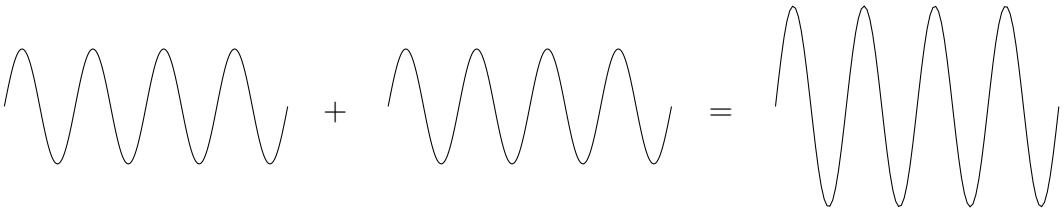


Figure 4: Constructive interference.

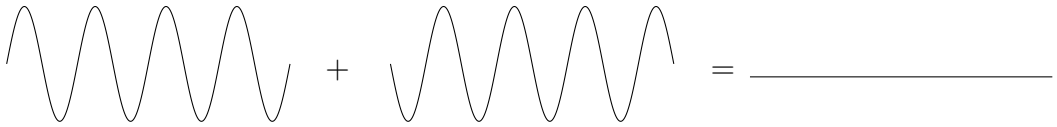


Figure 5: Destructive interference.

4 Developing the open-source toolchain

We will first look into how the packets that multi-Spin uses are defined, after which we will outline how the toolchain has been constructed. The entire open-source toolchain including usage instructions can be found on https://www.github.com/timvandermeij/radio-tomography.

4.1 Spin packets

The signals broadcasted by the RF nodes contain a Spin packet as payload. The *Spin packet* structure `spinPacket_t` is defined in the multi-Spin software in `xpand2531/spin_multichannel.h`. We have extended multi-Spin with correlation value measurements. The adapted Spin packet structure that we use is defined as follows (in C):

```
typedef struct {
    unsigned int packet_counter;
    char TX_id;
    signed char RSS[MAX_NUM_NODES];
    char CORR[MAX_NUM_NODES];
    signed char TX_channel;
} spinPacket_t;
```

The *correlation value* can be seen as the bit error rate, where a value of 110 indicates a frame of maximum quality and a value of 50 indicates a minimum quality frame. We have added the correlation value of each link to the Spin packet (in the `CORR` array) as an additional heuristic for usage in the reconstruction phase. The RF signal contains the correlation value in the least significant seven bits of the frame-check sequence field (right after the CRC OK field) [9]. Because we read one byte at a time from the received RF signal, we can extract the correlation value using the bit mask `0x7F` on the last byte. With this addition, we can define the complete packet format of a Spin packet broadcasted by the RF nodes [10]. A Spin packet contains:

- the packet counter which is increased by one for each broadcasted packet;
- the ID of the node transmitting the packet;
- an array with β RSS measurements;
- an array with β correlation value measurements;
- the frequency channel on which the packet is broadcasted.

4.2 Compilation using SDCC

The toolchain that we have built is primarily made for usage on Linux. However, the toolchain can also be adapted for usage on Windows or other operating systems. This is because the toolchain's dependencies (such as SDCC and Python) are available for all operating systems and most of them are open-source. On top of that, we have adapted the multi-Spin and the Texas Instruments USB firmware library code to make them both compatible with the open-source Small Device C Compiler (SDCC).

Before we started working on this, the code could only be compiled on Windows with IAR Embedded Workbench, a commercial development environment. After our modification to the code of both multi-Spin and Texas Instruments' USB firmware code, the entire codebase can be compiled with SDCC on practically every operating system available. These efforts make sure that usage of the toolchain is no longer limited to one operating system and a commercial compiler, which also makes the toolchain more accessible for researchers or students without a large budget.

Adapting the original code for usage with SDCC has been rather challenging. The first problem that we faced was modifying the USB firmware for usage with SDCC. Texas Instruments had packed their USB descriptors in an S51 file, which is a file containing assembly code that is processed by the commercial compiler in IAR Embedded Workbench during compilation. However, because it is not clear how IAR does this because their code is closed-source, we had to find a way to make the contents of the S51 file available to SDCC. We have done this by defining the USB descriptors in the USB firmware library configuration file (in C) such that they will be compiled by SDCC. By using that approach,

we did not have to worry about converting and linking the assembly code ourselves.

The second major issue we encountered was that SDCC is handling interrupts in another way than IAR. It turns out that SDCC demands that interrupt handlers are always defined in `main.c`, which is not the case for IAR. For instance, the USB interrupt for the listener node had to be defined in `main.c` as follows:

```
void usbirqHandler(void);
void usb_irq_handler(void) __interrupt 6 {
    usbirqHandler();
}
```

We then handle the interrupt in `main.c` and run the USB interrupt handler code defined in the USB firmware source code.

After successfully transforming the codebase to compile with SDCC, we have provided build automation. The multi-Spin code consists of two parts: software for the listener node (USB dongle) and software for the RF nodes. Both must be compiled individually using the steps outlined in the usage instructions for the toolchain. We have provided build automation (a Makefile) for both parts of the software. The result of running `make` is (among others) an *Intel HEX file* that contains the compiled code.

4.3 Flashing the nodes

The Intel HEX files obtained in the previous section have to be flashed onto the CC2530 chips. In order to do this, we use "cc-tool", a simple and open flashing tool for programming 8051-based SoCs [11]. The tool uses the debug port of the CC2530 chip in order to program the microcontroller. A pulse is put on the reset line of the debug port to enter debug mode and to be able to transfer data from the debug port to flash memory.

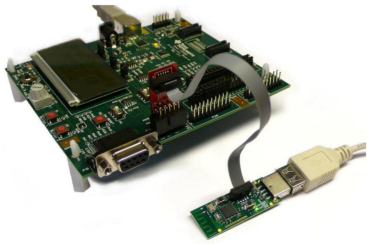


Figure 6: USB dongle connected to SmartRF board [7].

To flash an RF node or the listener node, one must connect the SmartRF board to the computer via USB. After that, the RF node or the listener node must be connected to the SmartRF board with the matching debug connector. Figure 6 shows how to do this for the listener node (the USB dongle). When the hardware is set up successfully, one can simply run `cc-tool -ew listener-node.hex -v` to flash the listener node chip or `cc-tool -ew rf-node.hex -v` to flash an RF node chip.

4.4 Tools

Furthermore, the toolchain contains several tools that we have written for working with the sensor networks and for analyzing their behaviour. We list these tools (all written in Python) below:

- *Sniffer*: a simple application that captures the received Spin packets from the serial connection and displays them in a readable format.
- *Realtime 2D plotter*: an application that listens for Spin packets and updates a 2D line plot almost immediately (only for usage with two RF nodes).
- *Measurement framework*: a framework for performing experiments with the sensor network. It fetches packets from the serial connection, filters them according to rules and exports the results as text files or LaTeX plots (using the "pgfplots" package).

The measurement framework is programmed in an object-oriented way to provide easy to use and readable code for researchers. The following Python code fragment shows the code needed to perform a simple experiment. Note that we assume that the channels are given to the application via command line arguments.

```
# Get packets for ten seconds
sniffer = Sniffer()
sniffer.start(10)
sniffer.stop()
packets = sniffer.getPackets()

# Only keep packets from node 1 to node 2
filter = Filter(packets)
packets = filter.where('fromNode', 1)
packets = filter.where('toNode', 2)

# Write the results to a text file and create two LaTeX plots
export = Export(packets)
export.txt('measurements.txt')
export.tex('plot_rss.tex', 'rss', channels)
export.tex('plot_corr.tex', 'corr', channels)
```

5 Experiments

In order for a reconstruction to be accurate, it is important to know how to set up the sensor network in the most effective way and to understand the properties of sensor networks. The aim of our experiments is to find out which properties have an impact

on the signal strength of the individual links (and therefore on the quality of the entire network) and which do not. With this information we can construct better sensor networks to get more accurate reconstructions.

The experiments have been conducted inside the same room at CWI Amsterdam, which has a length of 480 cm and a width of 440 cm, is entirely empty except for a small table for the operator and has been left unchanged during the experiments. The room has been empty to get consistent results and to make sure that the properties of this specific room are left out of the equation as much as possible. Our goal is to research properties of the individual nodes and the entire network that apply for practically every room or environment. Figure 7a shows a picture from inside the experiment room. During the experiments we made sure that there was no movement because that could cause variations in signal strengths. We also made sure to not plug the USB dongle directly into the laptop, but instead use a USB connector cable (as we discovered during earlier tests that this might improve the measurements).

All experiments have been done on the four frequency channels 11, 16, 21 and 26 (taken from a range between 11 and 26) to make sure that a good portion of the available channels is tested. Measurements have been collected for exactly 10 seconds using ZB500 nodes. We did not use the Texas Instruments nodes for these final experiments because the ZB500 nodes (which also have a CC2530 chip) are cheaper and lack an amplifier, which is beneficial because amplifiers can increase multipath effects.

As can be seen in Figure 7b, we have constructed our own stands to keep all RF nodes at the same height and position. Because the goal of our research is also to get the best reconstruction for the lowest price possible, we used materials that we could get for a low price. Each complete stand costs 15 euros and we have built 16 of them. We use a parasol base, a one meter long PVC pipe and several connector parts for each stand.

(a) Inside the experiment room.

(b) One of the sixteen stands.

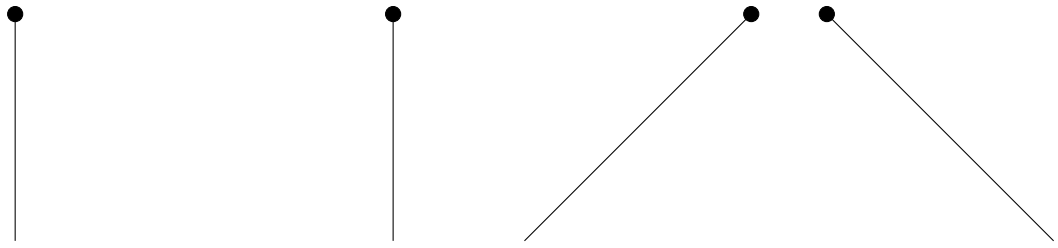
Figure 7: The experiment room and one of the stands.

5.1 Antenna direction

This first experiment is done to provide insight into the best antenna direction. Each node has an antenna that can be pointed in three directions with different angles: horizontal, diagonal or vertical (see Figure 8). In order to find out which direction gives the highest and most stable RSS values, we have tried all of them. The results can be found in Figure 9. We did the experiments with two nodes that had a distance of two meters in between them. The plots show RSS values of the link from node 1 to node 2. The horizontal axis shows the measurement number and the vertical axis shows the RSS value (ranging approximately between 0 and -90 dBm) of each channel for that measurement.

The results of this experiment are somewhat surprising and can partly be explained by knowing that the used antennas are dipole antennas. Those antennas emit signals, roughly from the center of the antenna, in a way that resembles a torus [12]. By positioning the antennas vertically, the toroidal surfaces largely overlap and therefore most signals arrive at the destination node, causing constant and normal RSS measurements (see Figure 9a). If we position the the antennas diagonally, the toroidal surfaces both point in a different direction. Any signal that does arrive at the destination node is most likely to have arrived there through multipath effects, causing the RSS values to drop drastically as depicted in Figure 9b.

While the previous results were exptected, positioning the antennas horizontally gave rather surprising results. In that case the toroidal surfaces still do not overlap, which does cause the RSS values to be lower than with the antennas in a vertical direction as can be seen in Figure 9c, but the RSS values are still rather high in comparison to the 45 degree case. The exact reason for this remains an open question, but this experiment does indicate that positioning the antennas vertically gives the highest and most stable RSS values.

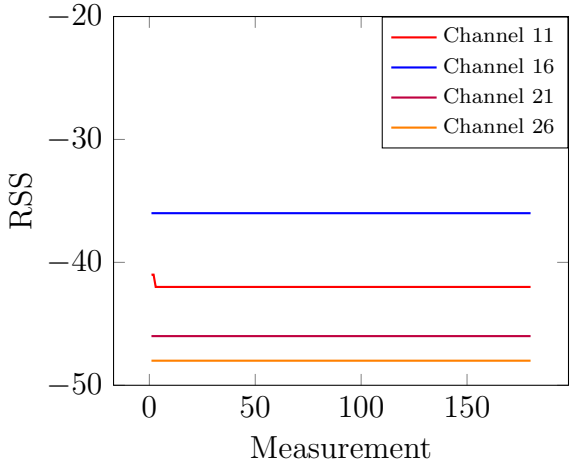


(a) Vertical (90 degrees).

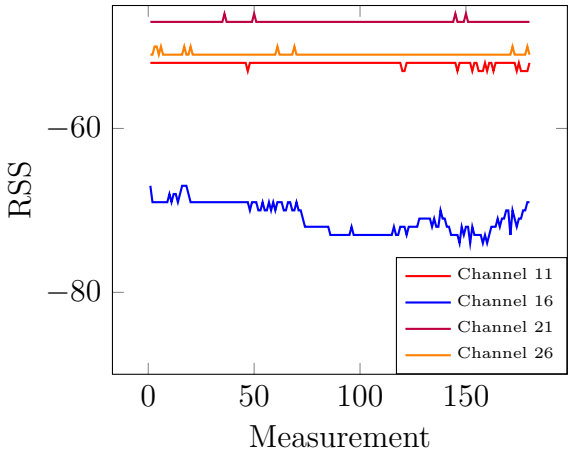
(b) Diagonal (45 degrees).

(c) Horizontal (0 degrees).

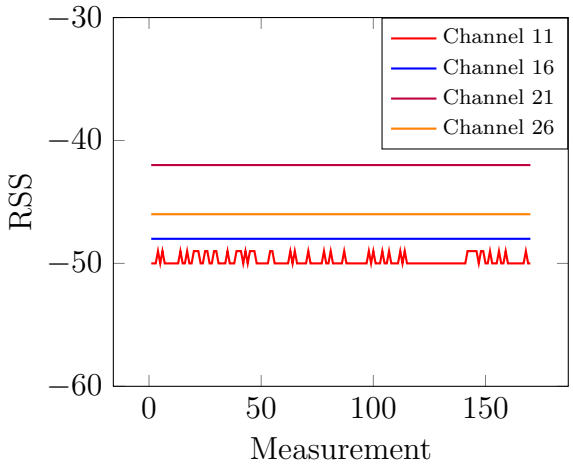
Figure 8: The three used antenna directions for two nodes facing each other.



(a) Vertical (90 degrees).



(b) Diagonal (45 degrees).



(c) Horizontal (0 degrees).

Figure 9: RSS plots of three different antenna directions. Note that the vertical axes differ per plot.

5.2 Symmetric properties

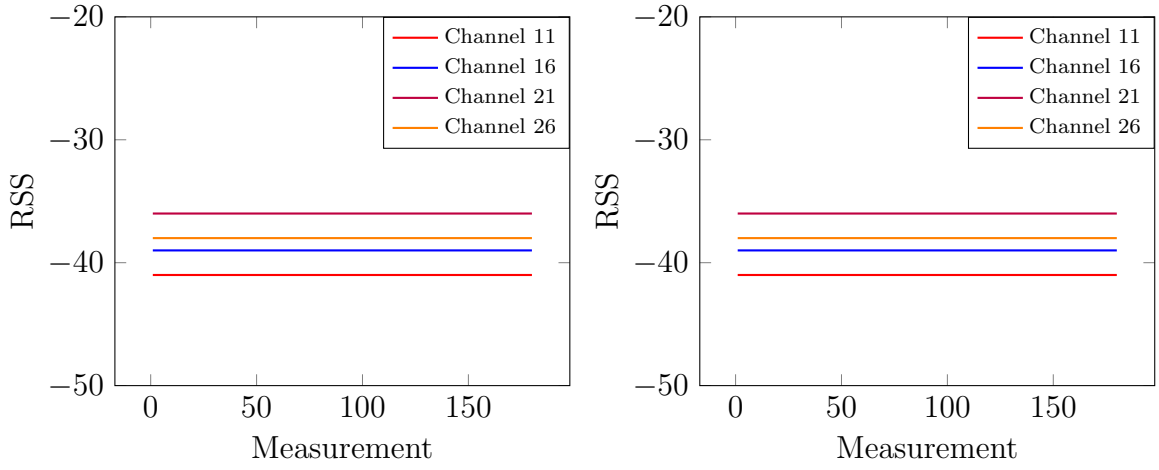
One might expect each link in the network to have symmetric properties, i.e., the RSS value of the link between nodes x and y should be equal to that of the link between nodes y and x if the nodes are equal (the same hardware and the same battery power). To verify this hypothesis, we have made a setup involving two nodes facing each other. In order to get more conclusive results, we have split this experiment into two parts that are outlined below. For these experiments, we have used equal hardware with new (full) batteries.

5.2.1 Fixed node positions

For the first part of this experiment, the node positions are fixed. We consider the link between node 1 and node 2 and the reverse link, which is the link between node 2 and node 1. The difference is purely the direction in which the packets are sent; the setup itself is fixed. Figure 10 shows the results that we obtained. The measurement number (horizontal axis) is plotted against the RSS value for each channel of a measurement (vertical axis).

Comparing Figure 10a and Figure 10b directly confirms our hypothesis. The two nodes do show symmetric properties because both plots are exactly the same. Doing this experiment multiple times will yield the same results. However, small alterations can occur because we are testing in an inside environment and completely avoiding multipath effects is practically impossible, even though we were able to greatly minimize this by making the room empty and by not moving during the experiments.

When we processed the results, we noticed that even though the RSS values of both links are the same, those of the link from node 2 to node 1 were all exactly one channel behind. The cause of this lies in the design of the RF node code in multi-Spin. Unless we are the last node to transmit, we cannot avoid having collected some RSS values before we do a channel hop and so when the node transmits, it will transmit the RSS values from the previous channel. We have adjusted for this to make sure that Figure 10b displays the actual results and not the skewed results.



(a) RSS values of link 1 to 2.

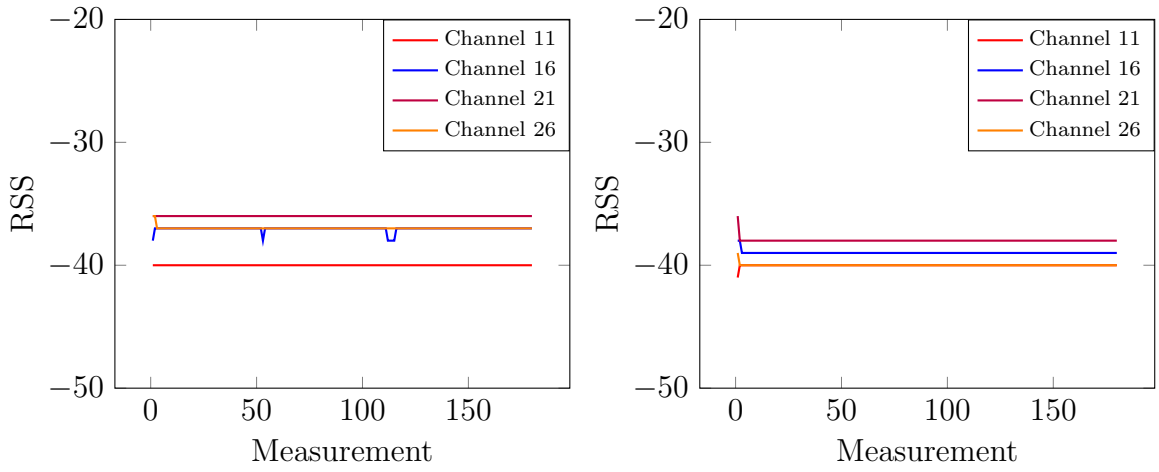
(b) RSS values of link 2 to 1.

Figure 10: RSS plots when the nodes have a fixed position.

5.2.2 Swapped node positions

For the second part of this experiment, the physical nodes have been swapped. This time we only consider the link from node 1 to node 2. Since the nodes are physically swapped, we will also measure the reverse link. Therefore, for this experiment the measured link is fixed and the physical node positions are changed, which is opposite to the previous experiment but should (largely) yield the same results. Figure 11 shows the obtained results.

If we compare Figure 11a and Figure 11b, we see however that the nodes do not show symmetric behaviour anymore because the same channels do not have the same RSS values. This can be explained by the fact that, even though we carefully switched the node positions to make sure we would not change their positions in the stands, slight alterations in the node positions can have a large effect on multipath effects. Because we cannot place the nodes in exactly the same position, we will always have to deal with different multipath effects, which is what we see in the plots. This shows that RSS values can change really easily, making them hard to work with.



(a) RSS values: node 1 left, 2 right.

(b) RSS values: node 1 right, 2 left.

Figure 11: RSS plots when the node positions have been swapped.

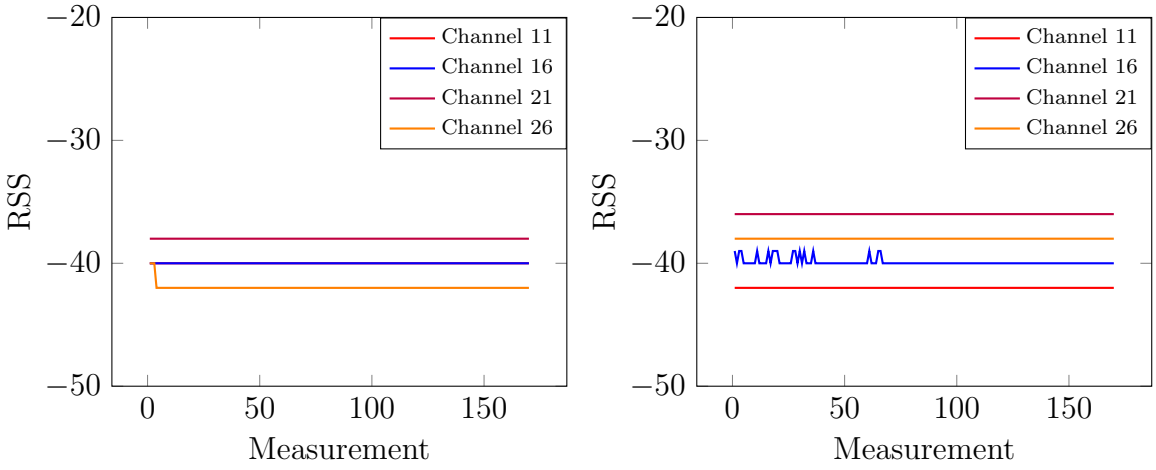
5.3 Corner nodes versus non-corner nodes

Would it be beneficial to avoid placing nodes in the corners of the room? We expected that placing nodes in the corners of a room might increase multipath effects. To test this hypothesis, both setups have been made: one with the two nodes in the two left corners of the room (see Figure 12a) and one with the same two nodes centered against the wall (see Figure 12b). The results of this experiment can be seen in Figure 13. RSS values from node 1 to node 2 are displayed in the plots.

(a) Corner nodes experiment setup. (b) Non-corner nodes experiment setup

Figure 12: Overview of the two used setups for this experiment.

Comparing Figure 13a (where channel 11 and channel 16 have the exact same RSS values) with Figure 13b shows that avoiding corner nodes seems to perform a bit better on practically all frequency channels. However, the results are quite similar, so if avoiding the corners is not a possiblity then there will be no major loss in signal strength. The signal strength will for instance still be good enough for a proper reconstruction. The small variations in RSS in Figure 13b can be explained by a very small movement at the start of the experiment, even though we attempted to minimize this as much as possible. Fortunately, this does not have any impact on the overall results presented here.



(a) Corner nodes. (b) Non-corner nodes (center of wall).

Figure 13: RSS plots when we placed the nodes in the corners or not.

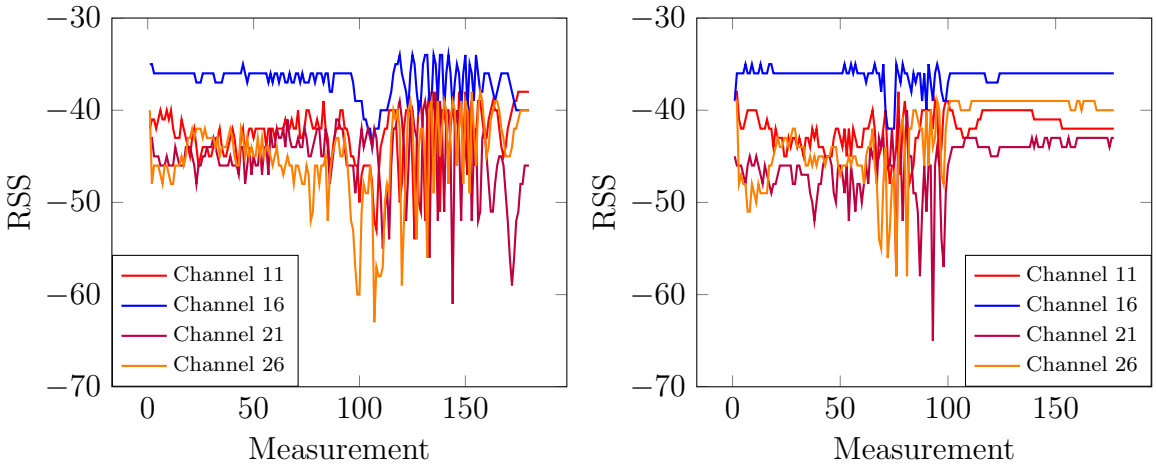
5.4 Walking versus running

Another interesting question is how big the sample size should be when monitoring the network. In order to investigate the network in real time, we would also need a realtime reconstruction. However, reconstruction can be a time-consuming task because of the fact that RTI is an inverse problem. The better the reconstruction image, the longer it takes to produce the image. This is an important trade-off when using RTI for security purposes.

If we choose a higher quality image, that inevitably means that the reconstruction will take some more time. During the time that the reconstruction requires, however, new measurements will keep on coming in, so it will be necessary to define a sample size, i.e., a fixed number of measurements for which we will trigger the reconstruction. Other measurements will be rejected as we cannot possibly process each single one of them. This experiment aims to show that we cannot set the sample size too high because persons running through the sensor network might not be detected that way.

We have set up two nodes facing each other (exactly two meters apart). For the first measurement, the test person crossed the link between the nodes with a normal speed, whereas for the second experiment the test person crossed the link between the nodes at a regular running speed. Both measurements can be found in Figure 14. The RSS values (vertical axis) for each measurement (horizontal axis) are plotted.

There are two observations for these plots. The first observation is that walking and running causes a lot of noise on all channels. This is because as the test person is moving, the multipath effects change all the time, causing a mix of high and low RSS values. The reconstruction phase will have to compensate for this. The second observation is that running through the link caused a shorter and cleaner spike, whereafter the network quickly recovers. Walking was detected between measurements 100 and 160 approximately, whereas running was only detected between measurements 65 and 100 approximately. This means that the spike is $\frac{160-100}{100-65} = \frac{60}{35} \approx 1.7$ times smaller if the person is running through the network. If our sample size does not take running or other fast movements into account, we might not detect this in the reconstruction. Therefore we would have to settle for a slightly lower quality image if we want to detect more movement.



(a) Walking.

(b) Running.

Figure 14: RSS plots showing the difference between walking or running.

5.5 Circle-shaped network versus square network

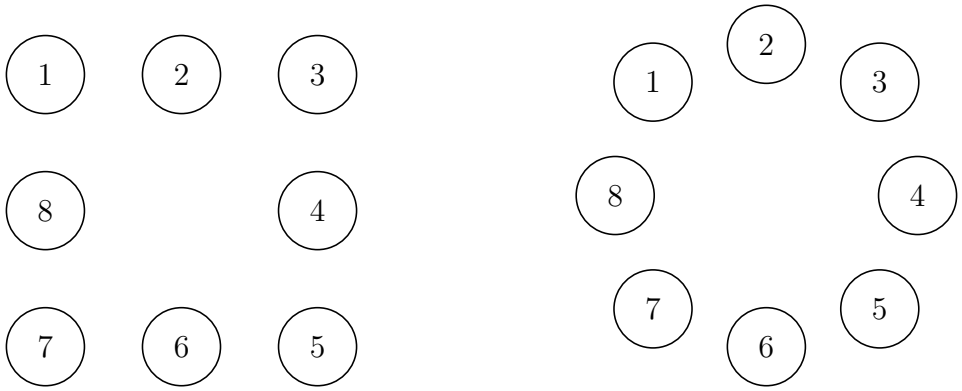
The positioning of nodes in the sensor network is important for a stable and effective sensor network. Essentially, only circular and square shapes are considered to be useful. It is however not clear which one of the two is better and how they relate to each other. This experiment aims to figure out which shape yields the best and also most stable RSS values. In order to research this, we made made two networks with eight nodes, one having a circular shape and one having a square shape (see Figure 15). After collecting the measurements (we only used channel 21 as that one appeared to be the best and most stable in the previous experiments), we processed them by taking the average RSS value $\bar{x}$ of each link and calculating the standard deviation σ of each link, the latter of which is defined as

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

with n being the total number of measurements for a particular link and x_i being the measurement at position i in the row of measurements for that link. The standard deviation gives a measurement for the amount of variation from the average $\bar{x}$, which is an indication whether or not the RSS measurements are stable. More stable measurements lead to better reconstructions, so this is desired behaviour of a sensor network.

The results of this experiment can be found in the appendix of this bachelor thesis. As can be seen there, a circular shape tends to yield better and more stable measurements than a square shape (see Figure 15a for the used numbering). The RSS values of the circular network are higher and the standard deviations are lower, meaning that the network is more stable and constant than the square network. A possible explanation for this is that in the square network, the links between node 1 and 5 or node 3 and 7 are larger in comparison with for instance the link between node 2 and 4. This is not the case in the circular network: those distances are the same in that case. In the appendix we can see that the circular network has less peaks in comparison to the square network, especially for the links mentioned before. This indicates that keeping the same distance for those links is beneficial for the RSS stability.

A second explanation could be that in the square network the link between node 5 and node 7 has node 6 right inbetween, which can be an obstruction. If we use a circular shape, this is not the case anymore, causing higher RSS values.



(a) Square shape. (b) Circular shape.

Figure 15: The two used network shapes with node numbering.

6 Conclusions and future work

In this bachelor thesis we have presented the results of four months of research in the field of radio tomography. We have created an open-source toolchain to help with radio tomographic imaging and radio tomography research. The toolchain is based on the multi-Spin protocol as developed by researchers from the University of Utah. In summary, our changes include converting multi-Spin and the Texas Instruments USB library for usage with the open-source SDCC compiler, implementing correlation value support and adding tool support (build automation, sniffer tools and a measurement framework) for working with the CC2530 chips.

Aside from the open-source toolchain, we have also done experiments to research sensor properties and to be able to compose a set of guidelines for setting up an effective sensor network. Our experiments have been done in an indoor environment on purpose. Even though an outdoor environment would avoid preventing multipath effects considerably, an indoor environment is a more realistic use case and therefore interesting to research. The guidelines for setting up an effective sensor network can be summarized as follows:

- Upright antennas appear to give the best performance.
- Each link in the network has symmetric properties, but small changes in the position of a node can cause variations in signal strengths.
- Avoiding corner nodes is not necessarily better than using the corners.
- The sample size needs to be chosen carefully for realtime reconstruction. A small sample size causes better reconstructions, but is also computationally more expensive. A larger sample size is computationally cheaper, but one might miss quick movements like running in the reconstruction.

- Using a circular shape for the sensor network appears to yield much better results than using a square shape.

Future work might include but is not limited to researching the effect of different types of materials on the signal strength, finding out how the signal strength decreases if length or height differences between nodes are increased and implementing dynamic configuration of the nodes in the sensor network. Being able to dynamically adjust properties of the sensor network (such as used frequency channels or transmission power) would not only save a lot of time for flashing the nodes but would also provide insight into dynamic behaviour of the network. Researching height differences is relevant for a 3D reconstruction.

In conclusion, radio tomography is an interesting and emerging area of research with lots of open problems to research. Our work aims to facilitate research in this area by providing an open-source toolchain and network setup guidelines for setting up effective sensor networks.

Glossary

MAC address — Media access control address
RF — Radio frequency
RSS — Received signal strength
RTI — Radio tomographic imaging
SoC — System on chip
TDMA — Time division multiple access

References

[1] J. Wilson and N. Patwari. "Radio tomographic imaging with wireless networks". In: *IEEE Transactions on Mobile Computing* 9.5 (2010), pp. 621–632.
DOI: 10.1109/TMC.2009.174.

[2] J. Schellevis. "Wifi-tracking: Winkels volgen je voetsporen (Dutch)". Jan. 2014.
URL: https://www.tweakers.net/reviews/3385/1.

[3] O. Kaltiokallio, M. Bocca, and N. Patwari. "Enhancing the accuracy of radio tomographic imaging using channel diversity". In: *2012 IEEE 9th International Conference on Mobile Adhoc and Sensor Systems (MASS)*. 2012, pp. 254–262.
DOI: 10.1109/MASS.2012.6502524.

[4] M. Bocca, O. Kaltiokallio, and N. Patwari. "Radio tomographic imaging for ambient assisted living". In: *Evaluating AAL Systems Through Competitive Benchmarking*. Vol. 362. Communications in Computer and Information Science. Springer, 2013, pp. 108–130.
DOI: 10.1007/978-3-642-37419-7_9.

[5] Texas Instruments. "CC2530 development kit". Apr. 2010.
URL: http://www.ti.com/tool/cc2530dk.

[6] ZigBee Alliance. "ZigBee 2012 specification: Features at-a-glance". Dec. 2012.
URL: https://docs.zigbee.org/zigbee-docs/dcn/07-5299.pdf.

[7] Texas Instruments. "CC2530 development kit user's guide". Apr. 2010, p. 6.
URL: http://www.ti.com/lit/ug/swru208b/swru208b.pdf.

[8] J. Belcher. "MIT course notes: Interference and diffraction". Dec. 2010.
URL: http://ocw.mit.edu/courses/physics/8-02sc-physics-ii-electricity-and-magnetism-fall-2010/nature-of-light/interference/MIT8_02SC_notes31to32.pdf.

[9] Texas Instruments. "CC253x system-on-Chip solution for 2.4-GHz IEEE 802.15.4 and ZigBee applications (specification)". Jan. 2014, pp. 230–234.
URL: http://www.ti.com/lit/ug/swru191e/swru191e.pdf.

[10] M. Bocca. "multi-Spin v2.0 cheatsheet". Jan. 2014, p. 2.
URL: https://sites.google.com/site/boccamaurizio/multi-Spin_v2.0.zip.

[11] cc-tool. "Support for Texas Instruments CC debugger in Linux". Feb. 2013.
URL: http://sourceforge.net/projects/cctool.

[12] P. Jonsson and J. Sidén. "Investigation of antennas for RFID tags on paperbased products". Research report R-01-11. Mid Sweden University, 2001.
URL: http://miun.diva-portal.org/smash/get/diva2:31664/FULLTEXT01.pdf.

Appendix

Link	Circular RSS	Circular standard deviation	Square RSS	Square standard deviation	Best
$1 \rightarrow 2$	-38	1.41	-38	0	Square
$1 \rightarrow 3$	-44	9.06	-43	1	Square
$1 \rightarrow 4$	-44	2	-42	1	Square
$1 \rightarrow 5$	-44	2.45	-49	1.73	Circular
$1 \rightarrow 6$	-46	1.41	-48	1	-
$1 \rightarrow 7$	-40	0	-42	1.41	Circular
$1 \rightarrow 8$	-38	1.41	-40	0	-
$2 \rightarrow 1$	-38	1.73	-39	1	-
$2 \rightarrow 3$	-31	1	-37	2.65	Circular
$2 \rightarrow 4$	-35	0	-35	8.66	Circular
$2 \rightarrow 5$	-36	8.66	-43	12.17	Circular
$2 \rightarrow 6$	-44	2.24	-43	9.27	Circular
$2 \rightarrow 7$	-36	1	-36	8.94	Circular
$2 \rightarrow 8$	-37	1.41	-37	1.73	-
$3 \rightarrow 1$	-44	0	-42	1.41	-
$3 \rightarrow 2$	-30	0	-36	2.65	Circular
$3 \rightarrow 4$	-31	1	-34	1	Circular
$3 \rightarrow 5$	-37	1	-38	9.22	Circular
$3 \rightarrow 6$	-39	8.89	-40	9.17	-
$3 \rightarrow 7$	-38	1	-46	10	Circular
$3 \rightarrow 8$	-40	2.45	-47	8.19	Circular
$4 \rightarrow 1$	-44	2	-42	0	Square
$4 \rightarrow 2$	-35	1	-35	0	Square
$4 \rightarrow 3$	-31	0	-34	12.12	Circular
$4 \rightarrow 5$	-31	1.73	-34	2.83	Circular
$4 \rightarrow 6$	-38	2	-37	2	Square
$4 \rightarrow 7$	-37	1	-40	10.44	Circular
$4 \rightarrow 8$	-42	0	-42	0	-

Table 1: Comparison of circular and square network results (part 1 of 2). $x \rightarrow y$ represents the link from node x to node y.

Link	Circular RSS	Circular standard deviation	Square RSS	Square standard deviation	Best
$5 \rightarrow 1$	-44	2.24	-49	1.73	Circular
$5 \rightarrow 2$	-36	1	-42	8.25	Circular
$5 \rightarrow 3$	-37	1.41	-39	3	Circular
$5 \rightarrow 4$	-30	1	-33	8.89	Circular
$5 \rightarrow 6$	-34	0	-35	0	Circular
$5 \rightarrow 7$	-36	1.41	-39	9.06	Circular
$5 \rightarrow 8$	-41	3	-42	4.9	Circular
$6 \rightarrow 1$	-46	1.41	-48	1	Circular
$6 \rightarrow 2$	-43	1.73	-43	1.73	-
$6 \rightarrow 3$	-39	1	-41	2	Circular
$6 \rightarrow 4$	-38	2	-37	2	Square
$6 \rightarrow 5$	-34	0	-35	0	Circular
$6 \rightarrow 7$	-34	1.41	-36	1	Circular
$6 \rightarrow 8$	-38	12.53	-39	1	Square
$7 \rightarrow 1$	-42	0	-44	1	Circular
$7 \rightarrow 2$	-36	0	-37	2	Circular
$7 \rightarrow 3$	-39	1	-48	9.9	Circular
$7 \rightarrow 4$	-38	1	-41	5.92	Circular
$7 \rightarrow 5$	-38	1.41	-41	2.45	Circular
$7 \rightarrow 6$	-35	1	-37	1	Circular
$7 \rightarrow 8$	-34	0	-36	3	Circular
$8 \rightarrow 1$	-37	1.41	-40	0	Circular
$8 \rightarrow 2$	-36	8.83	-35	8.66	-
$8 \rightarrow 3$	-39	2.24	-46	12.29	Circular
$8 \rightarrow 4$	-42	1.41	-41	0	Square
$8 \rightarrow 5$	-41	3	-41	10.05	Circular
$8 \rightarrow 6$	-39	1.73	-39	1.73	-
$8 \rightarrow 7$	-33	8.54	-34	2.45	Square

Table 2: Comparison of circular and square network results (part 2 of 2). $x \rightarrow y$ represents the link from node x to node y.