II-Management and Performance Analysis of Sensor Networks

UA_6012FTIIOT

2.4 GHz vs 868MHz

Author:
Mats De Meyer
Quentin Van Ravels

Student Number:
20129544
20130716

May 22, 2018

Contents

1 Introduction & Goals **1**

2 Background Theory **1**
2.1 2.4 GHz vs 868 MHz 1
2.2 Power Consumption 2

3 Methods & Procedure **2**
3.1 Power Consumption 2
3.1.1 Energest 3
3.1.2 KeySight N6705B DC Power Analyzer 3
3.2 Latency 4
3.2.1 Implicit Network Time Synchronisation: Timesynch 4
3.2.2 Altered Timestamp Printing 4
3.2.3 Python Script 5
3.3 Package Delivery Ratio 5

4 Results **5**
4.1 Power Consumption 5
4.2 Latency 6
4.3 Package Delivery Ratio 8

5 Conclusion **8**

A Appendice: Datasheet info **9**

List of Figures

1 Latency Confidence Interval of 95% for 2.4 GHz 7
2 Latency Confidence Interval of 95% for 868 MHz 7

List of Tables

1 The calculated energy used of the energest runs. 6
2 KeySight DC Power Analyzer runs. 6
3 Zoul power modes current usage. 9
4 Radio power consumption comparison. 9

1 Introduction & Goals

This project will test the performance of a point to point unicast link, featuring two Zolertia Re-Motes. The Zoul mote have two radio modules, being the cc1200 at 868 MHz and the cc2538 at 2.4GHz. When keeping every parameter identical when switching radios, we can get a good comparison between the two frequencies. The goal of this project is to compare power consumption, packet delivery ratio and latency between the two frequencies. Our code and results can be found on github.

2 Background Theory

This section gives some background information about the used frequencies and the power consumption of IoT devices.

2.1 2.4 GHz vs 868 MHz

With the rise of the Internet of Things, there is a rapid increase in connected and wireless devices. This creates huge flood in the wireless spectrum. With a lot of current technology such as Wi-Fi, Bluetooth, 802.15.4 and even microwave ovens all using the 2.4 GHz frequency band, there is a high chance of interference. On the contrary, the sub GHz ISM band is much less populated. In Europe, the 868 MHz frequency is used whereas in America they use 915 MHz. These radio bands are free for unlicensed used, but duty cycle and transmission power regulations have to be taken into account. A lot of recent IoT wireless technologies use these sub GHz bands and sometimes go even lower, to 433 MHz.

Besides the interference benefits, sub GHz also benefit from the longer wavelength associated with the lower frequencies. The longer the wavelength, the better the signal propagates through obstacles. When looking at IoT applications, devices are often places in awkward locations yet still have to send their data. The main downside to using these frequency band are the duty cycle regulations, often allowing only a 1% radio active time. This combined with limited data rate only make these frequency band useful for small packets, or short bursts of data.

2.2 Power Consumption

When working with low power wireless devices, the radios are often the most power hungry components. While the CPU uses a considerable amount of energy in active mode, most modules offer several sleep modes. These modes are meant to conserve energy when the CPU is in idle mode. Current draw often drops from tens of mA to below a single mA in these sleep modes. When running a simple program, the CPU can sleep for most of the time. The radios do have wake up for transmitting and for listening to messages. With current draw of these radios also often being in the tens of mA, the time on air has to be minimized to maximize battery life.

3 Methods & Procedure

When comparing two different frequency, we have to be able to choose which radio is used when compiling the code. This is done in the project-conf.h file, by commenting and uncommenting line 2 in Listing 1. When commented, the 2.4 GHz cc2538 radio is used. When uncommented, the 868 MHz cc1200 radio is used.

The Radio Duty-Cycle (RDC) protocol used is nullrdc, the MAC protocol is CSMA, and the network layer is rime. This remains constant during all of the measurements. The rime addresses are 2d:c8 for the sender, 2d:f9 for the receiver.

```
1 #define NETSTACK_CONF_RDC                nullrdc_driver
2 #define CC1200_CONF_SUBGHZ_50KBPS_MODE   1
3 #define NETSTACK_CONF_RDC_CHANNEL_CHECK_RATE   16
4 #define NULLRDC_802154_AUTOACK           1
```

Listing 1: project-conf.h file, showing the used protols.

3.1 Power Consumption

To compare the power consumption between the two frequencies, a program in c is written. This program is based on the send-unicast.c file, but altered to send 300 messages, one every second. To avoid startup abnormalities when certain modules are starting up, a 10 second wait time is implemented before sending the messages. On the receiving node, the receive-unicast.c file is used, with some minor changes to the output print when a message is received. The payload of the packet is a string of 5 characters, thus 5 bytes.

3.1.1 Energest

Contiki has a module called Energest, providing a software-based energy estimation. It basically tracks the time various hardware components are used and in which mode they are being used. When combining these durations with the power consumption values from the datasheets, one can calculate an estimated energy consumption. This is tested on the transmitting mote, printing the CPU, LPM, IRQ, TRANSMIT and LISTEN value in the console after each sent message. These console logs are converted to a usable csv file using a python script.

Our energest values are cumulative meaning we get the total time of the componentens at the end. These values are returned in clock ticks. To convert these values to seconds, the amount of ticks has to be divided by the amount of ticks per second, being 32768 (=`RTIMER_SECOND`). Knowing the energest values in seconds and the currents from the datasheets (found in Table 3 and Table 4), we can calculate the charge by multiplying the energest times by the current (`Q = I * t`). When multiplying the charge with the voltage we get the energy in Joules (`E = Q * V`). The voltage used is 3.7V. This energy can then be converted to mWh, to compare with the values from the DC Power Analyzer.

3.1.2 KeySight N6705B DC Power Analyzer

To perfrom real measurements, the KeySight N6705 DC Power Analyzer is used. This devices power the motes (at 3.7V), but also measures the voltage, current and power the module is using. The machine itself has an interface and the ability to take screenshots and export to USB, but also has a much more user friendly PC interface. With a USB connection, the machine can be controlled from a computer with a more user friendly user interface. A datalogging fucntion is present, making us able to run the same program as before. The duration of a single run is set to 5 minutes and 30 seconds, to make sure nothing is missed. These data logs are stored in a .dlg file, but can easily be exported to a .csv file. This exported data is timestamped voltage, current and power data. The period between samples is set to 1ms in the program, but shows as 0.00100352s in the logs.

However, when trying to compare this data with our cumulative energest values we stumble on some issues. The energest is only able to provide a cumulative power consumption of the components over the full time of measuring. To also get a cumulative total of the DC power analyzer data, the logs are opened in the software again. With the use of markers, we place one right before the first message and after the last message. The measurements windows shows us the used charge/energy

between said markers in mWh. These values for each of the 10 runs can now be compared with the calculated energest values.

3.2 Latency

To test latency, the same approach is taken. Five runs on each frequency of 300 messages, one every second are done. Different c files are used for this, removing the Energest library and extra's. The main logic does remain the same.

This time, the packet number is sent as the payload. This ensures us taking the latency of a single packet, not comparing data of two different packets. Furthermore, the package delivery can easily be checked with this data.

3.2.1 Implicit Network Time Synchronisation: Timesynch

To synchronise the clocks of the Zoul modules, the TimeSynch module from Contiki is used. It implicitly synchronises the clocks of all nodes in a network. It is implicit in that no explicit time synchronisation messages are sent. Instead, every radio message is timestamped. Each device has an authority level and when receiving a message from a higher authority, the devices adjusts its clock toward the clock of the sending node. This is done for every incoming packet.

Unfortunately, when trying to use this module on the Zoul motes we run into some issues. When compiling, errors are thrown even when configuring the Makefile and project-conf.h file correctly. Upon further investigation, we find that this module as of this moment only works on cc2420-based platforms such as the Z1 module.

3.2.2 Altered Timestamp Printing

As an easy alternative for the Timesynch module, the timestamp that is printed with every console message is altered. This is done by altering the timestamp file, found in the tools directory. The timestamp will now be shown as `"%Y%m%d,%H:%M;%S"` in localtime. The new timestamp line is shown in Listing 2.

```
perl -e 'use Time::HiRes qw(time);use POSIX qw(strftime);$|=1; while
    (<>){my $t = time;my $date = strftime "%Y%m%d,%H:%M:%S", localtime
    $t;$date .= sprintf ".%03d", ($t-int($t))*1000;print $date . ",$_
    ";}'
```

Listing 2: Altered timestamp file, to measure latency.

3.2.3 Python Script

A python script was written to analyse the delay in packet delivery for 1500 messages per radio interface. This is done by a simple subtraction of the timestamps. Afterwards the resulting difference was plotted using confidence intervals.

3.3 Package Delivery Ratio

Measuring the package delivery ratio in our case is very straightforward. Because the payload of the message in the latency project is the packet counter, we can easily check if every packet is delivered by comparing the sent and received console logs.

4 Results

In this section, the obtained results using the methods explained above are discussed.

4.1 Power Consumption

As we can see, there is a notable difference between the two measurement types. This is most likely due to some unforeseen elements in the energest calculations. We are not 100% sure which sleep mode the motes are in, but even when calculating with the least efficient sleep mode, there is a significant difference between energest and power analyzer data. The energest results can be seen in Table 1, the power analyzer data in Table 2. Another discrepancy is the fact that the data of the power analyzer shows 868 MHz using significantly more energy when compared to 2.4 GHz. This is in contrast to our energest values, where 2.4 GHz uses a few tents of mWh's more.

Energy used at 2.4 GHz	Energy used at 868 MHz
6.41250803 mWh	6.244905725 mWh
6.414452658 mWh	6.244631805 mWh
6.412775607 mWh	6.243686299 mWh
6.412627243 mWh	6.244270295 mWh
6.412697721 mWh	6.244418672 mWh

Table 1: The calculated energy used of the energest runs.

Energy used at 2.4 GHz	Energy used at 868 MHz
8.86901 mWh	9.785586 mWh
8.868033 mWh	9.785022 mWh
8.871524 mWh	9.784965 mWh
8.875026 mWh	9.784966 mWh
8.877704 mWh	9.78594 mWh

Table 2: KeySight DC Power Analyzer runs.

4.2 Latency

As can be seen in the histogram plot 1 and 2 for respectively the 2.4 GHz and 868 MHz radio, the latencies are very centred. This is partially due to the low precision up to a hundredth of a second. Apart from that we can see that the expected latency is barely deviates from the mean, with the value about 4 to 5 hundredths of a second for 2.4 GHz, 13 hundredths of a second for 868 MHz and very few outliers.

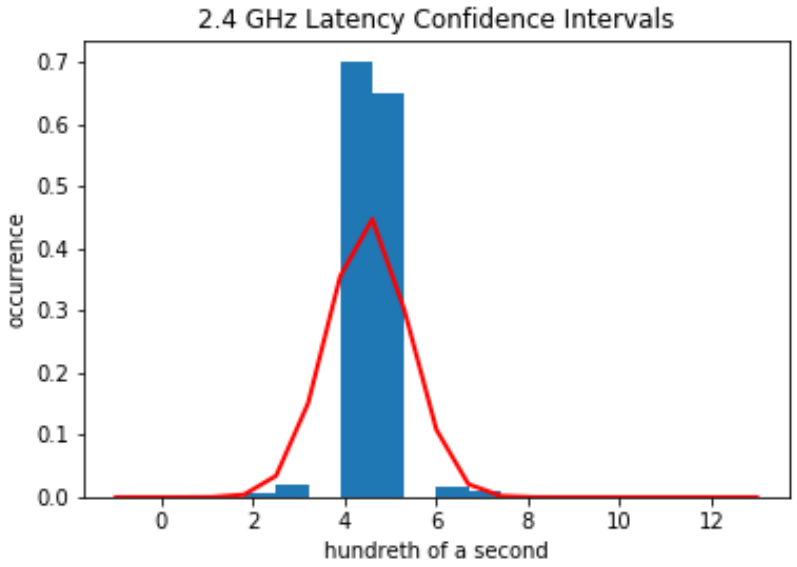

Figure 1: Latency Confidence Interval of 95% for 2.4 GHz

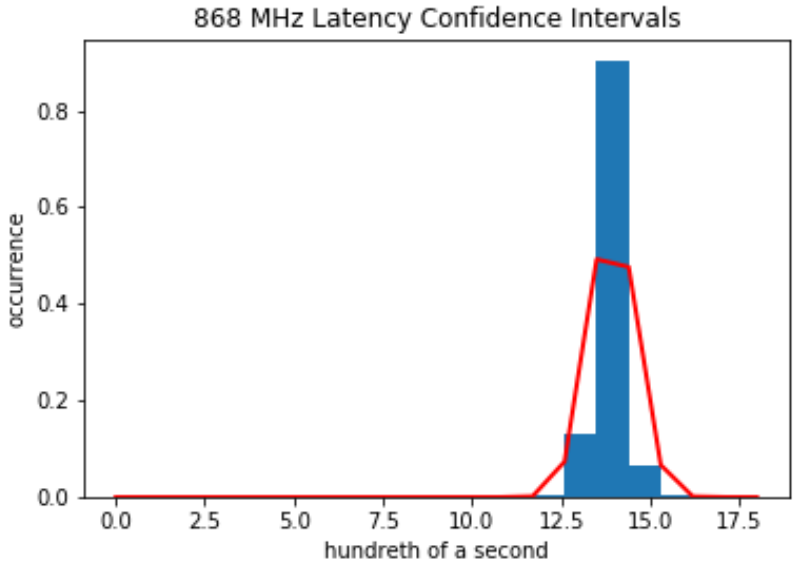

Figure 2: Latency Confidence Interval of 95% for 868 MHz

4.3 Package Delivery Ratio

Because we used the packet counter as the payload of the messages, we can easily check if there are lost packets. During our experiments performed with both radios, we got a PDR of 100% (1500/1500) for each frequency.

This is mainly due to packets only being sent every second, and `NULLRDC_802154_AUTOACK` being set to 1. When upping the frequency and disabling acknowledgements, we could see some packet loss in our data.

5 Conclusion

In this project, we attempted to keep as much parameters identical when switching between frequencies. By only changing radio, this keeps the measurements consistent. When comparing latency between 868 MHz and 2.4 GHz, we can see that the latency does in fact significantly increase when using 868 MHz. This is expected due to the cc1200 (868MHz) using a lower datarate and lower frequency, thus having a long on air time.

The the same data is used to test the package delivery ratio. On both frequencies we measure a PDR of 100%. This is due to us only sending a packet each second, and using acknowledgements.

For the energy consumption, we need a clearer picture to unravel which of both radios saves the most power. One thing is clear however, the energy used remains within the same scope for both frequencies. Thus from a perspective of power, both could be considered interchangeable.

Both frequency bands are feasible. At 868 MHz you sacrifice latency and datarate for less chance of interference. At 2.4 GHz it's the exact opposite, allowing lower latency and a higher throughput with the risk of interfence with current popular wireless technologies such as Wi-Fi and Bluetooth.

Appendix A Appendice: Datasheet info

The difference in current drawn between the CPU power modesof the Zoul module is shown in Table 3. A comparison of the power consumption of both the 2.4GHz radio (cc2538) and the 868MHz radio (cc1200) is shown in Table 4.

CPU Power Mode	**Current Drawn**
Active-Mode	20 mA
Power Mode 1	0.6 mA
Power Mode 2	1.3 μA
Power Mode 3	0.4μA

Table 3: Zoul power modes current usage.

Radio Used	**Transmit Current**	**Receive Current**
cc1200 (868 MHz)	35-46 mA (@10-14dBm)	19 mA
cc2538 (2.4 GHz)	24 mA	20 mA

Table 4: Radio power consumption comparison.