Optimizing the BLE Transmission of Sensor Data
An Exploratory Study

Remco van Dijk
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
r.vandijk-4@student.utwente.nl

ABSTRACT

Diabetic foot is a condition caused by long-term diabetes side effects, such as feeling disorders (neuropathy) and blood flow disorders (angiopathy). This condition can cause painful ulcers, which severely affect the patient's quality of life and makes up an estimated 40% of diabetes care costs worldwide. To help prevent this condition, ExPressure-1 is being developed to allow for early detection of foot ulcers, as well as to quantitatively assess therapeutic measures, such as specialized shoes and insoles. ExPressure-1 forms a grid of pressure sensors inside of a textile sock, which covers the whole foot and can thus form an accurate image of the pressure profile around the foot. This research is an exploratory study into finding ways to optimize the existing data transfer protocol over Bluetooth Low Energy (BLE), so that the relatively large amount of data generated by the sensor array can be transmitted within the required time. During this research it was demonstrated that certain BLE parameters, such as the connection interval, can be adjusted to optimize performance. Lastly, a rudimentary data compression scheme is shown to improve the stability of the connection.

Keywords

Pressure Sensor, Sock, Data Compression, Bluetooth Low Energy (BLE), Throughput, Diabetes, Diabetic Foot

1. INTRODUCTION

Diabetes is a chronic disease that is becoming more prevalent across the globe, with about 422 million people suffering from the disease worldwide [1]. Apart from the immediate symptoms, long term diabetes patients often develop secondary symptoms such as neuropathy and angiopathy. This combination often leads to diabetic foot problems, since wounds form more easily because of the patient's impaired feeling, combined with a suboptimal blood flow that causes these wounds to heal more slowly. All in all, diabetic foot problems not only pose a significant threat to a patient's quality of life, but it is estimated that 40% of all diabetes care costs are related to the treatment of foot problems.

Following these findings, it makes sense to implement some way of preventing these secondary disorders from occurring. To this end, the ExPressure-1 project was started, it aims to develop a sensor sock that measures the pressure load of a shoe around the foot. Using this technology, it would be possible to optimize and quantitatively evaluate the design of preventive measures, such as therapeutic insoles and orthopaedic shoes. Moreover, it would allow high-risk pressure areas on a person with diabetic foot disorder to be detected early, which could prevent the formation of an ulcer.

The goal of this research is to implement and optimize a part of ExPressure-1, namely the data transfer protocol between sock and server. The sock is layered with a matrix of 70 pressure sensors, of which measurements should be taken at a rate of 100 Hz using an Arduino Nano 33 BLE. The gathered data should then be transmitted through BLE [2] to a server that represents the data visually.

The scope of this research is limited to developing the data pre-processing and transfer protocol, since there are already implementations in place for the gathering of sensor data, as well as a server-side application for testing purposes.

2. PROBLEM STATEMENT

The main objective behind this research is to have a stable transmission of data that guarantees integrity and is fast enough to accommodate the relatively large amount of data that is generated.

Each of the sock's 70 sensors generates a 10-bit value, 100 times per second. Knowing this, it can be calculated that this equates to a rate of 70 kbit/s which has to be achieved by this protocol. However, it has been shown that the real data transmission rate in BLE may fluctuate below 70 kbit/s, depending on a multitude of outside factors and some of the BLE parameters [3, 4].

2.1 Research Questions

Following this problem statement, a research question can be formed:

How could data be transferred over BLE in a stable way, while ensuring integrity and achieving a minimum average speed of 70 kbit/s?

This can be split into 3 partial research questions:

1. **Integrity**: What is needed to ensure the correctness of the data received by the server?
2. **Stability**: Which techniques can be employed to reduce the fluctuation of the BLE transmission rate?
3. **Throughput**: Which measures can be taken to ensure that all sensor data is transmitted to the server in the allotted time?

35th Twente Student Conference on IT July 2nd, 2021, Enschede, The Netherlands.

3. BACKGROUND

This section will provide some of the necessary background information about the relevant layers within the BLE protocol stack shown in Figure 1, as well as the definition of relevant BLE settings.

3.1 BLE Protocol Stack

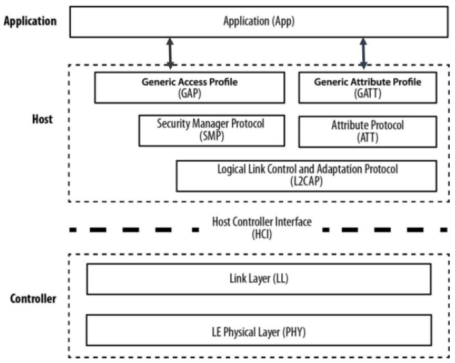

Figure 1. BLE Protocol Stack

The BLE protocol stack[5] essentially consists of two main layers, the controller and the host layer. The controller consists of the physical layer and the link layer, which is not the focus of this research, however it implements some underlying functionalities such as collision detection and avoidance.

Above the controller, there is the host layer, which accesses the controller through the Host Controller Interface (HCI). The host layer is the main point of interest with this research, since it contains some protocols that allow the application layer to tailor the connection parameters to its needs, as well as providing several other important functionalities.

One such protocol is the Logical Link Control and Adaptation Protocol (L2CAP), which allows the application to set the connection interval (CI) and handles packet fragmentation, among other functionalities.

The other protocol that this research focuses on is the Attribute Protocol (ATT), which - among other functionalities - handles exchanging the Maximum Transmission Unit (MTU) parameter. Furthermore, it advertises to connected devices which services and characteristics are available and the methods by which these can be accessed.

3.2 BLE Connection Parameters

There are three different connection parameters that are relevant to this research, namely the CI, MTU and the transmission method.

Firstly, the CI is a time value and indicates how much time passes between the start of each connection event. Lower values allow for more frequent connection events, but will generally result in increased power consumption.[6]

Secondly, the MTU is an integer value that indicates the maximum bytes of application data that may be sent in a single packet. If the number of bytes exceeds the MTU, the packet will be fragmented by the L2CAP layer.

Finally, there are two relevant methods of transmission in this context: notifications and indications. Indications guarantee the reliable delivery of data through acknowledgement packets, whereas notifications do not. Generally, this leads to a larger data throughput in notifications, with the downside of some lost data.

4. RELATED WORK

In this section, previous work related to the aforementioned research questions is discussed. This includes previous work by the faculty, as well as a literary overview. In order to obtain the literature, some general databases - such as Scopus and Google Scholar - were used, as well as more specific ones, such as ACM digital library and IEEE Xplore. Important keywords used while searching these databases include, but are not limited to: 'compression', 'BLE', 'throughput' and 'wireless sensor networks'.

First of all, some related experiments have already been conducted by the Pervasive Systems (PS) group. These experiments have shown that in practice, the maximum achievable throughput at this point is about one third of what needs to be achieved. Moreover, it was demonstrated that the connection is highly unstable, with peaks in connection speed going as high as twice the required speed, but a nominal rate that is far too low.

Secondly, the PS group has already constructed a hardware prototype that consists of the sensor array, along with an Arduino Nano that is able to capture the sensor data. They have also produced a server script for conducting tests, from which the former measurements were taken.

Apart from this, there has been quite some previous work on BLE in general [7, 2] that describes the functioning of the BLE protocol stack. However, more importantly, a lot of research has been done on the optimization of BLE parameters as well, which will be very useful in this research. [8, 9, 3, 10, 4]

Furthermore, it has been shown that the CRC that is used in all BLE packets can not only be used to detect errors, but also to correct them in 60% of the cases.[11] This could be a promising method to improve research question 1.

Lastly, a lot of research has already been conducted in the field of lossless data compression in general. [12, 13] With this research however, it would mainly be beneficial to look at a specific kind of data compression technique, namely time series compression. [14, 15] The reason for this is that time series compression is built on the assumption that the data is correlated through time, which is the case in this application.

5. METHODOLOGY

5.1 Experiment Setup

In order to determine the best way of answering the research questions, a number of controlled experiments have been designed. These experiments will mainly be focused on determining the best parameters for the BLE connection, as well as testing a rudimentary data compression algorithm.

The testing environment will utilize Nordic Semiconductor's nRF Connect mobile application to receive and log incoming measurements from an Arduino Nano 33 BLE. Experiments will be recorded for 5 minutes at a time to obtain reliable results.

Lastly, it should be noted that some of the experiments were recorded on slightly different timescales. This is due to a limitation in nRF Connect application. However, this should not hinder the observations, since it was possible to record 5 minutes at minimum of constant transmission for every experiment. This is more than enough to give a valid result.

5.1.1 Metrics

Some different metrics should be taken into account in order to evaluate the findings and experimental results. It should be noted that all metrics are expressed as a frequency, namely the amount of full measurements that can be sent during one second, where a full measurement is defined as the list containing one reading from every sensor. More specifically, the mean of this frequency is used to evaluate throughput (RQ3) and its standard deviation is used to evaluate stability (RQ2).

5.1.2 Control Group

To have a series of valid experiments, there needs to be a control group - or control experiment - to compare them to. This control experiment will utilize the existing solution, which has the following set of important connection parameters: the CI is set to 45.0ms and the MTU is set to 23 bytes. These are the standard parameters, as enforced by the Android implementation of the BLE protocol stack.

Furthermore, it is important to note that certain physical parameters are maintained, such as minimal interference from other Bluetooth devices and 2.4GHz Wi-Fi signals, as well as a distance of 1 meter between devices.

Lastly, BLE notifications are used to send individual measurements, one at a time.

5.2 Experiment Descriptions & Hypotheses

This section will describe the design and reasoning behind the different experiments that were conducted. The general objective of these experiments is to explore different possibilities for optimizing the BLE connection stability and throughput.

5.2.1 BLE Connection Interval

With this experiment, the goal is to observe the effects of setting the CI to its lowest possible setting, which in this case is 15.0ms instead of the 7.5ms mentioned in the BLE specification[7]. However, this does not detract from the observations that could be made, since it is still significantly less than the standard used in the control experiment.

It is expected that the lower CI will yield a significantly improved throughput with no significant effect towards the stability of the connection. This is expected because a lower CI means that more connection events can happen per second.

However, it could be said, according to F. J. Dian, A. Yousefi and S. Lim [3], that a higher CI will actually result in the highest throughput, since more packets could be sent continuously during longer connection events. However, this is not the case in practice, because the Android implementation of BLE limits the number of application layer packets that can be sent during a single connection event.

5.2.2 BLE Maximum Transmission Unit

During this experiment, the effects of increasing the MTU are observed. During the control experiment, this is set to the BLE standard[7] of 23 bytes. This relatively small packet size is maintained in order to limit the peak current - and thus the power consumption - during transmission. However, since we are currently only interested in finding ways to increase the throughput and stability, increasing power consumption is not a concern.

As for the hypothesis, it is expected that increasing the MTU will yield a improvement in terms of throughput. The reason for this is the fact that a full measurement could be sent within a single packet, without needing fragmentation, thus removing much of the protocol overhead associated with fragmentation.

On the other hand, having a large MTU might have a negative effect on the stability of the connection. Having a larger MTU will make it so that more data is lost for every packet dropped. Furthermore, it is expected that the packet loss rate will actually be greater, since the probability of having one or more bit errors in a packet increases with the amount of bits per packet.

5.2.3 Data Compression

With this experiment, the effects of a rudimentary data compression algorithm are tested. This algorithm compresses the sensor data by representing it more optimally. With the current solution, sensor readings are stored in short integers, which are 16-bit values. However, each sensor only has a 10-bit accuracy, meaning that each sensor reading in the current solution has 6 bits of redundant padding. It can then be calculated that removing this padding - and adding it back on the receiving end - will yield a 37.5% reduction in the amount of application data.

When using this improvement, it is expected that an improvement will be visible, since there is noticeably less data that needs to be transmitted. However, this improvement will most likely be significantly less than 37.5% with respect to throughput, because the overhead is still the same. Lastly, it is not expected that this method will have any noticeable impact on the stability of the connection.

5.2.4 Reliable Data Transfer

The last experiment will cover the effects on throughput of implementing research question 1: data integrity. To achieve this, we need only one measure on top of the existing solution: application layer packet acknowledgements. That is because the existing solution already implements bit-error detection using a Cyclic Redundancy Check (CRC).[16] To achieve this, BLE indications are used instead of notifications.[2]

For this experiment, it is expected that the throughput will decrease significantly, since much of the bandwidth will be taken up by acknowledgement packets. Moreover, retransmissions will most likely need to happen frequently, taking up even more bandwidth. Lastly, it is not expected that this will have any noticeable impact on connection stability.

6. RESULTS & DISCUSSION

In this section, the results of the experiments will be presented. The results in practice will be related to the hypotheses and theory. Lastly, some preliminary conclusions will be drawn from the observations.

6.1 Control Experiment

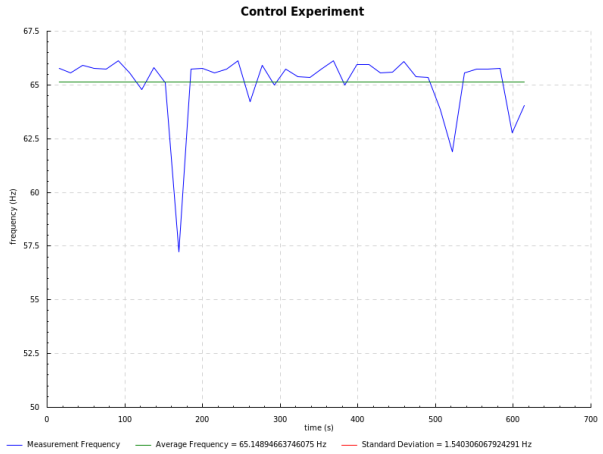

Figure 2. Control test

In Figure 2, we can see the results of the control experiment. These results will act as a reference for the other experiments. We see an average frequency of 65.1 Hz, which is still far from the goal of 100 Hz. Furthermore, there standard deviation is at 1.54 Hz, which is the result of the fluctuations visible in the graph.

6.2 BLE Connection Interval

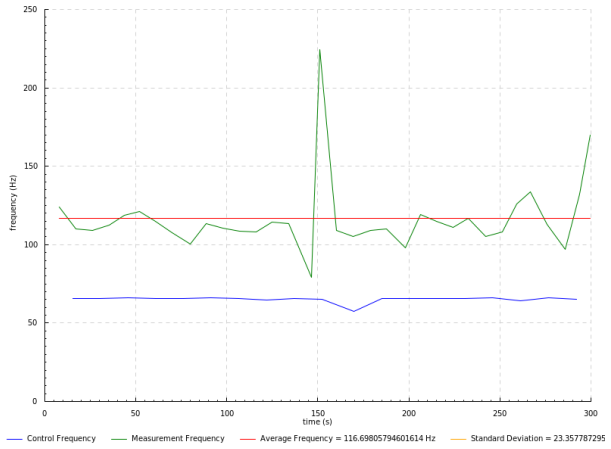

Figure 3. Connection Interval set to 15.0ms

Figure 3 shows the results from setting the CI to 15 ms. As expected, we see a significant increase in the average frequency, which has risen to 117 Hz, reaching the goal of 100 Hz. This can be explained by the fact that the connection events in the control experiment were not saturated with transmissions. Whereas with a shorter CI, more of the available time can now be utilized.

However, a less desirable result from this experiment is that the connection stability has decreased, which is evident from the standard deviation being much higher at 23.4 Hz. Unfortunately, there is no clear explanation for this phenomenon. However, this should not have a great impact in practice, since the average throughput is still far above the goal. Therefore, any dips in throughput will be amply compensated for by future peaks.

6.3 BLE Maximum Transmission Unit

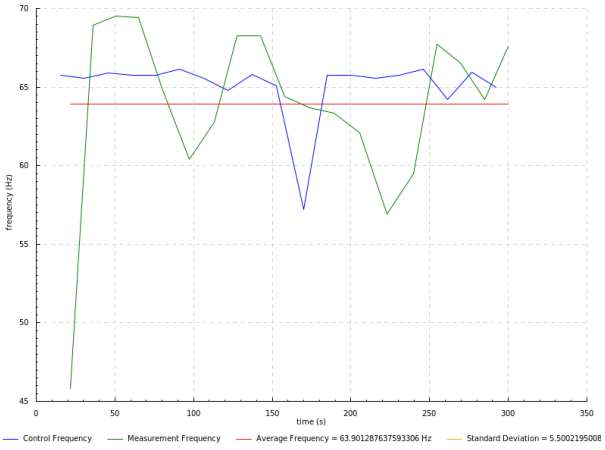

Figure 4. MTU set to 247 Bytes

Figure 4 shows the effects of setting the BLE MTU to 247 bytes, instead of the standard 23 bytes. With this experiment, a rather insignificant decrease can be observed in the average throughput: where the control experiment achieved 65.1 Hz, this experiment achieved 63.9 Hz. This is an unexpected result, since there is definitively less data to send, namely some of the overhead is removed. This could be explained by the Android API limiting the number of application level packets, as described before in section 5.2.1. Since the fragmentation process is not relevant at the application layer, we see no real benefit, since the available time is not fully used for transmission.

Furthermore, the figure shows quite a significant decrease in stability, moving from a 1.54 Hz standard deviation in the control experiment to 5.50 Hz with a larger MTU. However, this was expected, as explained in section 5.2.2.

6.4 Data Compression

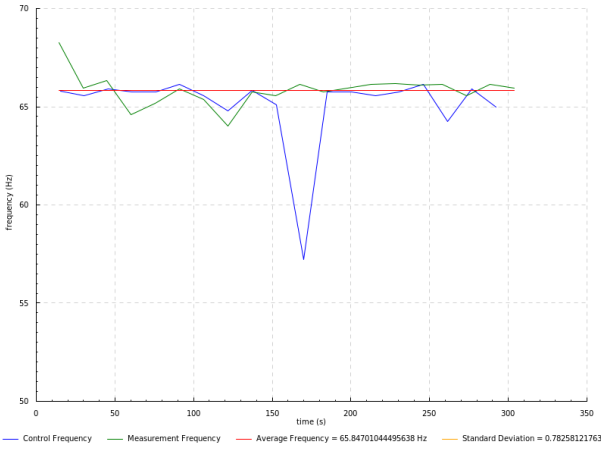

Figure 5. With Data Compression

With this experiment, the effects of the data compression scheme - as described in section 5.2.3 - are observed. Again, we see unexpected results in Figure 5: the average throughput is practically the same as the control experiment. Despite the significantly reduced size of data that

needs to be transmitted, there are no significant improvements. This situation can be explained with the same reasoning as the MTU experiment: because the CI is not fully utilized in neither the control nor this experiment, having less data per application packet does not make a difference.

As for the stability, there are some observable improvements as the standard deviation decreased to 0.783 Hz. This could possibly be explained by the same reasoning as to why the stability decreased with a larger MTU. Namely, since less data is sent, it follows that there is a lower probability of bit-errors.

6.5 Reliable Data Transfer

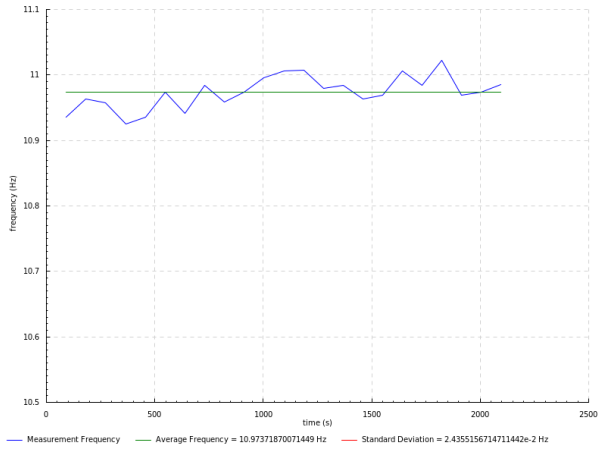

Figure 6. Using Indications

Lastly, an experiment was conducted to observe the effects - especially on throughput - of utilizing an application level packet acknowledgement scheme. The results of this experiment are visible in Figure 6. As expected, reliable data transmission comes at a rather high cost in terms of throughput, as the average frequency drops from 65.1 Hz down to 11.0 Hz. As explained in section 5.2.4, this is due to the substantial amount of small acknowledgements packets needing to be sent, along with any retransmissions.

Furthermore, a slight decrease in stability is observed, as is evident from the increased standard deviation at 2.44 Hz. This is somewhat unexpected, but it could be explained by the fact that when packet loss occurs, the subsequent retransmission(s) create even more delays.

7. CONCLUSIONS

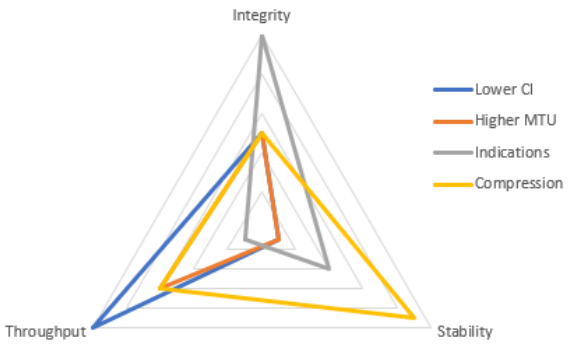

Figure 7. Strengths and Weaknesses

In this section, conclusions are drawn from the experimental results. The strengths and weaknesses of each experimental method are visible in Figure 7.

7.1 Integrity

In order to solve the first research question, the first solution that comes to mind would be to use indications as the method of transmission, since this enforces application layer packet acknowledgements. However, this solution is impractical, since it was observed that the throughput decreases by an order of magnitude. Therefore, we propose to use notifications instead.

Furthermore, all BLE packets - including notifications - provide a CRC, which could be used to detect and discard erroneous packets or even correct them in around 60% of corrupted packets.[11]

7.2 Stability

As for the second research question, it has been observed that the only positive change came from using data compression as described in section 5.2.3. Additionally, implementing this does not appear to have any disadvantages, therefore we propose to use this solution in any case.

On the other hand, it should be noted that all other experiments decreased the connection stability slightly, except for lowering the CI, which brought rather significant instability.

7.3 Throughput

Regarding the third research question, the largest increase in throughput by far was seen with a lower CI. However, other experiments that were expected to provide a similar improvement had not performed nearly as well. This leads to the conclusion that the CI is the major bottleneck in terms of throughput.

The underperforming experiments could all be explained due to the fact that the number of packets sent for each connection event was being limited. However, this effect can be alleviated by using a lower CI.

This theory leads to the hypothesis that measures such as data compression and setting a higher MTU will be seen when combined with a lower CI. However, investigating this is outside the scope of this current research and will be left as possible future work.

8. FUTURE WORK

In this section, some possible future work is discussed, which builds on the current findings.

One such possibility arises from the conclusion on throughput: it can be argued using both findings and theory that combining different measures with a lower CI could yield promising results. Therefore, we propose to continue experimentation using a lower CI in the control group, as this seemed to be the major bottleneck in the system.

Furthermore, during the experimentation phase, some significant - yet unexplained - hardware issues arose. This eventually led to the decision to use an Android phone as a receiver. However, a significant difference was observed when using different receiver hardware. This difference cannot be explained within the context of this research, but it might prove useful to conduct further research into different hardware solutions.

9. REFERENCES

[1] World Health Organization. Diabetes. https://www.who.int/health-topics/diabetestab=tab$_1$.*Accessed* : 25 − 4 − 2021.

[2] Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.

[3] F. John Dian, Amirhossein Yousefi, and Sungjoon Lim. A practical study on bluetooth low energy (ble) throughput. In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 768–771, 2018.

[4] Eunjeong Park, Myung-Sup Lee, and Saewoong Bahk. Data rate and transmission power adaptation for bluetooth low energy (poster). In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '19, page 550–551, New York, NY, USA, 2019. Association for Computing Machinery.

[5] Luca Leonardi, Gaetano Patti, and Lucia Lo Bello. Multi-hop real-time communications over bluetooth low energy industrial wireless mesh networks. *IEEE Access*, PP:1–1, 05 2018.

[6] J. Li, M. Bhuiyan, X. Huang, B. McDonald, Todd Farrell, and E. A. Clancy. Reducing electric power consumption when transmitting ecg/emg/eeg using a bluetooth low energy microcontroller†. In *2018 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pages 1–3, 2018.

[7] R. Heydon. *Bluetooth Low Energy: The Developer's Handbook*. Pearson Education, 2012.

[8] PrithviRaj Narendra, Simon Duquennoy, and Thiemo Voigt. Ble and ieee 802.15.4 in the iot: Evaluation and interoperability considerations. In Benny Mandler, Johann Marquez-Barja, Miguel Elias Mitre Campista, Dagmar Cagáňová, Hakima Chaouchi, Sherali Zeadally, Mohamad Badra, Stefano Giordano, Maria Fazio, Andrey Somov, and Radu-Laurentiu Vieriu, editors, *Internet of Things. IoT Infrastructures*, pages 427–438, Cham, 2016. Springer International Publishing.

[9] Andreina Liendo, Dominique Morche, Roberto Guizzetti, and Franck Rousseau. Ble parameter optimization for iot applications. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7, 2018.

[10] Carles Gomez, Ilker Demirkol, and Josep Paradells. Modeling the maximum throughput of bluetooth low energy in an error-prone link. *IEEE Communications Letters*, 15(11):1187–1189, 2011.

[11] Evgeny Tsimbalo, Xenofon Fafoutis, and Robert Piechocki. Fix it, don't bin it! - crc error correction in bluetooth low energy. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 286–290, 2015.

[12] Tossaporn Srisooksai, Kamol Keamarungsi, Poonlap Lamsrichan, and Kiyomichi Araki. Practical data compression in wireless sensor networks: A survey. *Journal of Network and Computer Applications*, 35(1):37–59, 2012. Collaborative Computing and Applications.

[13] Myungho Yeo, Dongook Seong, Yongjun Cho, and Jaesoo Yoo. Huffman coding algorithm for compression of sensor data in wireless sensor networks. In *Proceedings of the 2009 International Conference on Hybrid Information Technology*, ICHIT '09, page 296–301, New York, NY, USA, 2009. Association for Computing Machinery.

[14] Aleksey A. Belov and Aleksander Y. Proskuryakov. Time series compression in telecommunication systems for environmental monitoring of polluting emissions. In *2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, pages 391–395, 2018.

[15] Lucie Klus, Roman Klus, Elena Simona Lohan, Carlos Granell, Jukka Talvitie, Mikko Valkama, and Jari Nurmi. Direct lightweight temporal compression for wearable sensor data. *IEEE Sensors Letters*, 5(2):1–4, 2021.

[16] W. W. Peterson and D. T. Brown. Cyclic codes for error detection. *Proceedings of the IRE*, 49(1):228–235, 1961.