SSC20-WKVI-05

Wireless Bus Interconnects for Small Satellite Systems

Adam Fifth, Russel Trafford, Kenneth Wagner, Timothy Roche, Jake Matteo, Sangho Shin
Rowan University
201 Mullica Hill Road, Glassboro, NJ 08028; 609-892-1522
fiftha7@rowan.edu

ABSTRACT

For small satellite engineering systems, successfully managing the hundreds of in-system interconnections caused by a wired interface is a major element in the success of a SmallSat mission. Testing, integration, and mission operation of SmallSat subsystems frequently requires system interfaces to be reconfigured for extended mission capability and system reliability. We propose a Bluetooth Low Energy (BLE) based wireless interface technology to enable post-deployment reconfiguration of in-system interfaces. This wireless interface will improve system reliability while enabling ad hoc system level changes and reducing the probability of subsystem failure. While a wireless interconnect brings many benefits, its implementation raises potential technical challenges, including additional power consumption, data latency, interference with ground communications, susceptibility, and emissions. This work presents the concepts of wireless interface technologies, proof-of-concept experimental results of a BLE-based wireless interface system, and analysis of effective solutions for the aforementioned technical challenges. By limiting the RF power of the wireless interfaces, the susceptibility, emissions, and power consumption were be made minimal. Latency and interference were also be minimized through software optimization and error correction techniques. Proof-of-concept prototype lab experiments demonstrate the feasibility and adaptability of the proposed technology with increased ability to reconfigure assets compared to traditional wire-based interconnects.

INTRODUCTION

Motivation

Small satellites are generally referred to as satellites with a mass less than 1100 lb. These satellites were once considered to be limited in their application and scope because of their small size. Now, thanks to companies like SpaceX and Google, small satellites are being mass produced for commercial and scientific applications. With the proliferation of small satellites, the need for smaller subsystem circuit boards along with the ability to upgrade quickly, easily, and cheaply is needed.

Satellites are made up of many interconnected subsystems, and while this isn't a problem when satellites have the space to handle, as satellites shrink in size, wires and headers become a large percentage of satellite volume and increasingly more difficult to manage. Despite this, Wired connection interfaces are still used due to their simplicity of implementation, ability to facilitate high speed connections, and negligible power requirements.

In the CubeSat standard, an entire satellite and scientific mission can be constructed in units as small as 10 cm x 10 cm x 10 cm.[1] Oftentimes, these CubeSats utilize the PC/104 header to connect the vertically stacked subsystems.[2] Between each subsystem, this header consumes 8.5 mm in vertical height.[3] While some subsystems require this vertical clearance, others only require enough distance to avoid touching. If this dynamic spacing capability was leveraged, additional subsystems could be deployed in the same confined space, enabling more complex missions.

The main functions that a wired interconnect provide can be divided into the three categories: power distribution, inter-subsystem communication, and sensor data transfer. While power distribution must remain wired, subsystem communication and data transfer could be converted to a wireless solution.

A small footprint wireless interface would not only reclaim lost volume, but would also streamline prelaunch assembly, mitigating the risk of system failure. Currently, system debugging often requires complete disassembly of wired connections between subsystems. This poses a risk of pin fatigue or damage on each system teardown. Converting to a wireless approach will remove this risk, as only the minimal power connections and mechanical fasten-

ers would need to be removed, reducing the risk of electrical connection failure.

In addition to easing assembly, wireless interconnects also allow for ad hoc system-level changes post-deployment. In current wired systems, all possible interconnects between subsystems must be accounted for prior to system deployment. A wireless solution would enable dynamic satellite configuration based on updated requirements, allowing for resources to be repurposed in novel ways. For instance, if a temperature sensor malfunctioned during deployment, the wireless interface could easily allow for rerouting of data from another subsystem's sensor to the damaged subsystem. This functionality could further be expanded to repurpose retired small satellites for entirely new missions through Over-the-Air Updates (OTA).

Objective

The decision was made to utilize the Bluetooth Low Energy (BLE) as the protocol for the wireless interconnect bus during the proof of concept phase of this project.[3] BLE has the advantage of a low power consumption and a hierarchical nature of storing data. This phase of the project involves experimental analysis of the challenges posed by such a system.[3] We investigate the feasibility of a BLE subsystem interconnect by detailing how a wireless bus could take advantage of the BLE stack and interface with a subsystem. We then identify key technical challenges and potential solutions. Finally, we develop a proof-of-concept wireless interface and collect experimental data to measure key system metrics, drawing conclusions on the practicality of such an interface.

APPROACH

Subsystem Design & Architecture

For a BLE interconnect to be a viable alternative to a traditional wired system, it must have a means to interface with existing subsystem designs including COTS hardware. This means that each subsystem must have either a microcontroller or microprocessor to handle its overall communication management. We propose that each subsystem will have the ability to communicate via UART to a second low-power microcontroller handling BLE communications. Each subsystem will issue commands to the BLE controller to send and receive data over the BLE wireless network. A diagram of the proposed subsystem is depicted in Figure 1. With this method, each subsystem's Main Control Unit (MCU) would conserve its own computational resources for mission related tasks, including operating Transducers (TD's) and performing system health checks, while the BLE controller will handle the wireless communication. This abstracts the wireless interface from the subsystem MCU.

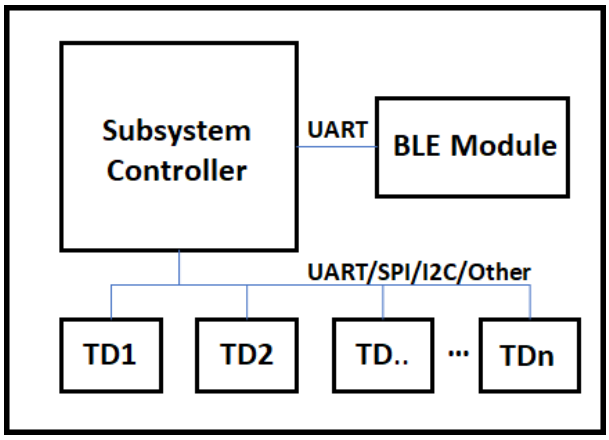

Figure 1: Proposed Subsystem BLE Interface

In the proposed BLE interconnect system, all subsystem communication will be transmitted in an organized data structure called the Generic Attribute (GATT) protocol in the BLE stack.[4] The GATT protocol can be edited on the fly and enables the possibility of post-deployment reconfiguration. All information sent over BLE must be broken down into individual characteristics and grouped together by a data type called a service. Finally, these services are further grouped together into profiles.[4] We suggest that the BLE interconnect should have two separate profiles per subsystem: one for the subsystem health and one for transducer information. The proposed data structure can be seen in Figure 2.

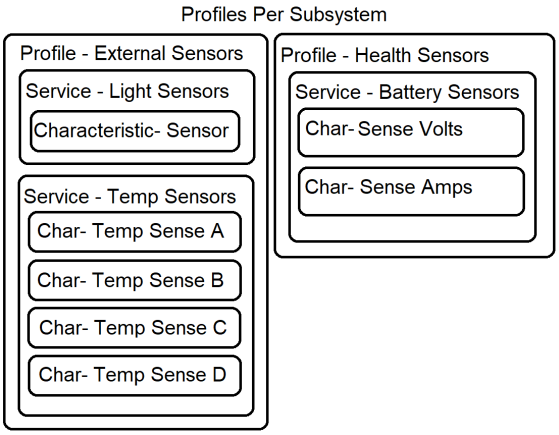

Figure 2: Proposed GATT Data Structure

In terms of the wireless interface's network architecture, the various subsystems will be connected in a star topology to a central node, such as the Command and Data handling (CD&H) module. The BLE interface will be configured as a one-to-many network.[4] This would allow CD&H to be connected to each subsystem simultaneously. This is considered a semi-distributed architecture.[3]

Methodology

To transform the current wired interface between subsystems into a wireless one, some significant technical challenges need to be resolved. Among these challenges are considerations of power consumption, latency, throughput, packet error rate (PER), signal leakage, near-field effects, and interference. When evaluating each of these areas we will consider the impact on the total mission effectiveness and reliability as well as acceptable performance changes due to the transition from wired to wireless interconnections.

Power Usage

With a wireless interface incorporated into a satellite architecture, electrical power budgets must be adjusted for the increased power draw. BLE became one of the top choices for this interface system due to the low power requirements. During normal operation, the BLE radio cycles between transmitting data and conserving power via idling.[5] The total set time for each one of these cycles is called the connection interval.[5] When there is no activity, the radio spends the time idling, conserving power.

Average current consumption is seen to decrease exponentially as the connection interval increases.[6] This means for subsystems that do not need constant communication with CD&H, power consumption could be made minimal through increasing this connection interval. However, this would negatively impact the throughput of the connections.[5]

Power consumed by the BLE controller was measured both while actively transmitting and when the modules were unpaired. In addition, the power consumed by the entire prototype subsystem, consisting of both an BLE controller and a central MCU was also measured in the same configurations. For all tests, the transmit power was set to $-14\,\mathrm{dBm}$ and the connection interval was set at its minimum value of $7.5\,\mathrm{ms}$.

Latency

There are two forms of latency that need to be considered, round trip and system latency. Round-trip latency is the amount of time it take for one data packet to transfer from one device to another and the second device transfer an acknowledgement back.[5] BLE round trip latency is a function of the BLE connection interval as well as the bit error rate.[6] Reducing PER is the primary way to improve round-trip latency. System latency is the amount of time it takes for a packet to leave the subsystem MCU, transfer through the BLE interface, and arrive at another subsystem MCU. System latency is reduced through software optimizations and minimization of command overhead transmitted along with the data.

Measuring both the BLE latency and system latency gives a full perspective on the total latency for incorporating BLE into a satellite. For BLE latency, the transmitting BLE controller sets one of its output pins high when data is transferred. On the receiving end, the second BLE controller sets one of its own output pins to a digital one upon receiving the packet. The time difference between the signals is calculated using an oscilloscope. The experimental configuration can be seen in Figure 3.

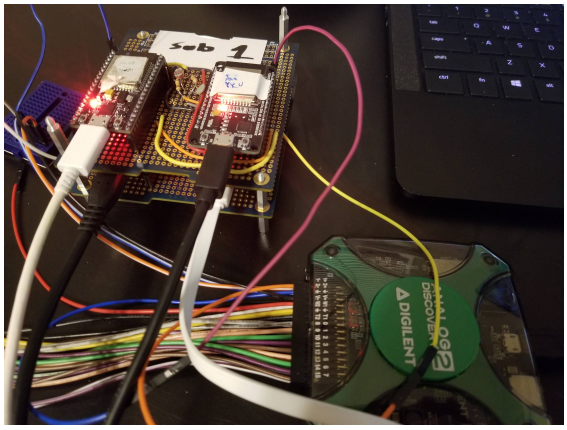

Figure 3: Latency Experimental Setup

Subsystem latency is measured using a similar method. The transmitting central MCU sets its pin to a digital 1 prior to sending the transmit command over UART and the receiving MCU sets its pin high upon receiving the packet. For both latency metrics, the average and maximum latency, a large number of runs is used to verify the data. Approximately 600 runs for each latency measurement is used to verify the results. Since these subsystems need to operate within a satellite in various configurations, testing is

also performed in three settings: subsystems side by side, stacked, and enclosed in a mock 1U CubeSat frame.

Throughput

The throughput of a BLE connection is dependant on a variety of factors. The physical layer of Bluetooth (PHY) is the lowest layer of the Bluetooth stack. As part of the protocol, this layer can transmit data at a maximum speed of 1 Mbps,[4] which is the overall limiting factor in throughput and accounts for both user data and packet overhead. The maximum speed that has been achieved for BLE user data throughput is 237 Kbps.[6] Only a certain number of packets can be sent per data packet group, called a connection event in BLE. Therefore, throughput would be maximized if the length of the connection interval is minimized.[5] In addition, interconnect throughput can also be maximized by increasing the baud rate of the UART connection between the central MCU and the BLE controller.

Throughput of the wireless interface is divided into two metrics: Packet Throughput and Payload Throughput. Packet throughput is the absolute throughput of the BLE interface, including all of the packet overhead. However, payload throughput considers solely the rate of transfer of subsystem application data.[6] For both metrics, the systems are placed 10 cm away from each other and the power level is set to -14 dBm. Through the use of the Teledyne Lecroy Frontline BPA® low energy Bluetooth® protocol analyzer, both the packet and the payload throughput over time is recorded over the transmission of 100,000 packets.

Signal Leakage

When replacing a wired interconnect with a wireless one, electromagnetic leakage is produced. Being a potential security risk, the leakage must be mitigated. This can be achieved by reducing transmission power below the receiving threshold of another satellite as well as using various encryption techniques. While the minimum BLE transmit power is -14 dBm, there can still be methods to intercept that signal as well as interrupt interconnect transmissions. This means that encryption needs to by utilized along with power minimization. The BLE standard supports Elliptic-curve Diffie-Hellman (EC&DH) cryptography to minimize vulnerabilities to passive eavesdropping or Man-in-the-Middle (MITM) attacks in addition to AES encryption.[7] The effects of these encryption technologies on other parameters such as latency and throughput will need to further examined in future research.

Packet Error Rate

Data integrity of the proposed BLE interface is analyzed by evaluating the Packet Error Rate (PER) for the system. PER is the measure of the total number of incorrect transmissions divided by the total number of correct transmissions. The PER can give us valuable insight into what occurs during a BLE interface data exchange on the link-layer(LL) level of the Bluetooth stack.[4] With this kind of insight, greater attention can be paid to the specific type of packets sent, for example data or advertising packets. Using PER as a means of analysis, aspects of the protocol that are affected by metrics such as distance and transmission power level on a per-packet basis can be more closely defined.

Figure 4: PER Experimental Setup

In order to test the PER, two ESP32's are used to simulate two SmallSat subsystems. Utilizing a Teledyne Lecroy Frontline BPA® low energy Bluetooth® protocol analyzer, a series of 1000 transmissions of a test payload is transmitted and the PER is recorded. This test is performed at distance of 5 cm to 30 cm in increments of 5 cm as shown in Figure 4. These distances are typical distances that subsystems could be separated by in a 1U-3U CubeSat form factor.[1] For each distance, transmission power levels are swept from -14dBm to 7dBm. In addition, the retransmission rates for each test were recorded.

Near-Field Effects

Near and Far Field regions occur when electromagnetic radiation scatter off an object, such as an antenna, as seen in Figure 5. Antennas normally oper-

ate in the Far Field when sending and receiving signals, where the field acts "normal". The near field is broken into 2 regions, the Reactive and Fresnel regions.

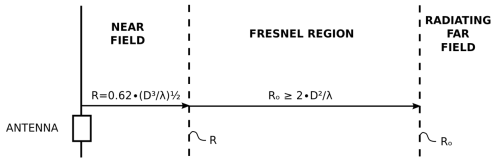

Figure 5: Antenna regions

The BLE modules have the potential for being in extremely close proximity to the near-field boundaries that can cause distortion and transmission errors. Calculating the range of the reactive field, Equation 1, and the region between the radiating near field and far field, Equation 2, is given by,

$$R \leq 0.62 sqrt \frac{D^2}{\lambda} \tag{1}$$

$$R \geq 2\frac{D^3}{\lambda} \tag{2}$$

where R is the radial distance from the antenna, D is the longest linear length of the antenna and λ for the wavelength. These values will be utilized for maintaining a minimum separation distance of the antennas. This will save potential failures from putting subsystems too close in proximity.

RESULTS AND ANALYSIS

System Power Usage

Prior to testing, software optimizations were made to minimize power consumption due to coding irregularities. This gives the ability to accurately calculate the additional power requirements needed for implementation of a wireless system. As shown in Figure 1, when the BLE controller was unpaired and off, it consumed below 50 mW, but when the device was on and connected to another subsystem, the power consumed increased to 645 mW. When the entire subsystem of two ESP32 modules is included in this measurement, the consumed power became 461 mW when unpaired and 805 mW when paired. These power figures may be further improved by placing the microcontroller into a high-level sleep mode whilst still maintaining the BLE connection.

Table 1: BLE Interface Power Usage

System Status	BLE Device	Entire Subsystem
Not Paired	<50 mW	461 mW
Paired	645 mW	805 mW

Latency

In the collected latency results for BLE to BLE communication, seen in Table 2, the average latency ranged from 7.1 ms to 4.7 ms. It was found the lowest latency was observed in the enclosed mock-cubesat frame, most likely due to the metal foil dampening of external sources of 2.4 GHz interference. This latency is higher than the BLE latency of 3 ms observed in other research,[6] most likely due to the software overhead.

Table 2: BLE to BLE Round Trip Latency

	Side by Side	Stacked	Enclosed
Average	7.1 ms	5.2 ms	4.7 ms
Maximum	20.7 ms	19.1 ms	11.4 ms

The subsystem central MCU to central MCU latency results can be seen in Table 3. The one-way latency ranged from 197.6 ms to 202.8 ms depending on the testing environment. This added latency is primarily due to the time taken for UART communications to send the transmit command and also due to the data rate of 9600 baud of the UART bus. These results could further be improved by optimizing the communication between central MCU and the BLE device, likely through increasing the baud rate.

Table 3: Central MCU to MCU Latency

	Side by Side	Stacked	Enclosed
Average	197.6 ms	202.8 ms	198.9 ms
Maximum	267.2 ms	295.6 ms	273.2 ms

Throughput

The prototype BLE interface is measured to have an average packet throughput, including BLE overhead, of 47.212 Kbps, while having an average payload throughput of 21.284 Kbps, as seen in Figures 6 and 7 respectively.

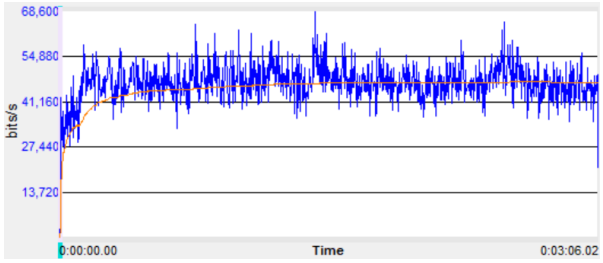

Figure 6: Measured Packet Throughput

As preliminary results, these could further be improved by streamlining the BLE controller software to optimize for speed of communication. This could include optimization of the ESP32 BLE library for the application of wireless interconnects.

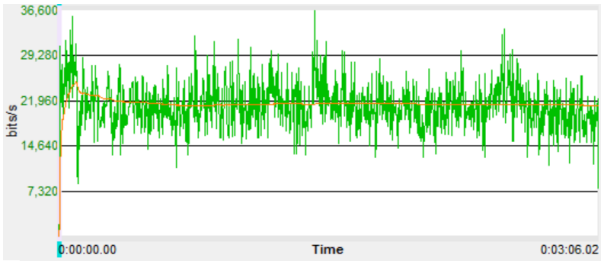

Figure 7: Measured Payload Throughput

Near-Field Effects

The reactive part of the Near Field for the ESP32, specifically, is within 3.03 mm while the Fresnel region extends to 3.32 mm, which were obtained using Equations 1 and 2, respectively. This means that as long as the antennas are separated by at least 3.32 mm, where the far field starts, near field interference should be mitigated.

Packet Error Rate

It was found that the BLE interface had a PER not exceeding 1.93% seen in Figure 8 for any transmit power tested. This low PER was likely due to the Cycle Redundancy Check (CRC) afforded by the Bluetooth 4.2 Standard.[4] By appending redundant bits to each transmitted packet, the receiver side was able to check whether the packet was correct while adding minimal power consumption, effectively affording a lower transmitter power level for a given distance.[8] For packets that could not be corrected through CRC, packet retransmission had to take place. In testing, the rate of packet retransmission did not exceed 0.2% for any transmit power tested or distance.

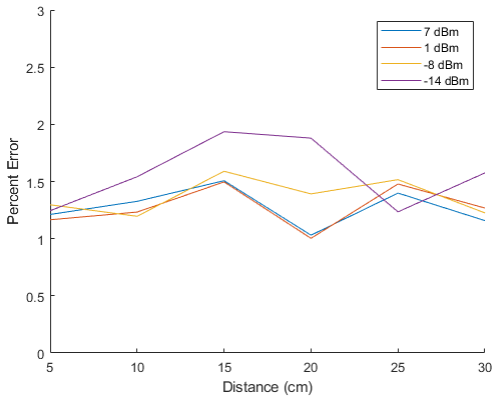

Figure 8: Packet Error Rate

DISCUSSION AND FUTURE WORK

After completing feasibility testing, we found that many of the technical challenges outlined in previous sections were minimal and could be remedied through software optimization and minimizing erroneous environmental factors. In regards to BLE latency, our results approximated that of other research[6] with an average latency of 4.7 ms when enclosed in a 1U mock CubeSat frame, which is an acceptable latency level for a majority of small satellite missions. PER was also found to be minimal with a measured error rate of under 2% for all transmit powers tested. However, most of these errors were corrected through the appended CRC bits. Less than 0.2% of transmissions required retransmission.

Although there is an increase in power consumption compared to a wired system, the increased reliability and reconfigurability afforded by a wireless interface is of greater importance than the minimal increase in power consumption. In addition, the power used by the BLE module could further be reduced through the use of a larger BLE connection interval[5] or by entering the BLE controller into a high level sleep mode that maintains BLE connectivity.[9]

In terms of throughput, the BLE interface achieved an average payload data rate of 21.284 Kbps. This lower data rate is due to BLE's focus on low power consumption. This puts the BLE interface at a disadvantage in some applications, such as transmitting large photos, which benefit from a higher throughput speed. Future research should be conducted to determine if a combination between BLE and Bluetooth Classic could be made. Subsystems requiring higher throughput could support both BLE and

Bluetooth. However, subsystems not requiring these high data speed could just support the power efficient BLE standard, leading to a compromise between the two technologies.

Due to the star topology of BLE networks, the ability for a client to multicast to numerous devices simultaneously is not supported by the Bluetooth 4.2 core specification.[4] Improvements in reliability can be made if this standard was adopted by future iterations. Future research should explore the possibility of utilizing a mesh network, supported in Bluetooth 5.[10] A mesh network would both enable simultaneous subsystem communication and enable intercommunication, potentially reducing latency.

One last area that needs further study is the ability for integrating BLE on individual transducers not tied to specific subsystem boards. Payloads often have sensors that aren't located on the main payload control board. This will need to be done to accomplish successful elimination of all wired data lines.

CONCLUSION

Through detailed analysis of the potential challenges, we found that a BLE communication bus within a small satellite is a viable alternative to a wired one. The BLE interface provides dynamic satellite configuration, ease of assembly, and flexibility in post deployment system changes without significant increase to the power budget. To integrate the wireless interface into existing subsystems, a UART communication bus was proposed between the subsystem central MCU and the BLE controller, abstracting wireless communication from design implementation.

After implementing improvements to potential challenges, a prototype BLE interface was developed. Experimental results were then collected for key system parameters. It was measured that BLE to BLE latency was as low as 4.7 ms, and the packet error rate did not exceed 1.93% and retransmission rates did not pass 0.2%. In addition, the power consumed by the BLE controller was measured to be 645 mW, but this could be reduced by placing the controller in a high level sleep mode or by increasing the connection interval. Average payload throughput was measured to be 21.284 Kbps, limited by BLE's low power operation. While such a BLE interface requires further optimization for deployment in a SmallSat, early testing indicates that a BLE interconnect is a viable alternative to a wired interconnect.

Acknowledgments

This work was sponsored in part by the NJSGC grant (No.5860)

References

[1] The CubeSat Program, Cal Poly SLO. *CubeSat Design Specification*, February 2014. Rev.13.

[2] J. Bouwmeester, M Langer, and E.K.A. Gill. Survey on the implementation and reliability of cubesat electrical bus interfaces. *CEAS Space Journal*, 2016.

[3] R. Trafford, N. Gorab, T. Smith, A. Fifth, J. Schmalzel, R. Krchnavek, and S. Shin. Wireless bus interconnects for flexible and reliable cubesat signal integrations. In 33^{rd} *Annual AIAA/USU Conference on Small Satellites*, 2019.

[4] Bluetooth. *Bluetooth 4.2 Specification*, December 2014.

[5] Jacopo Tosi, Fabrizio Taffoni, Marco Santacatterina, Roberto Sannino, and Domenico Formica. Performance evaluation of bluetooth low energy: A systematic review. *Sensors*, 17(12), 2017.

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

[7] K. Scarfone and J. Padgette. Guide to bluetooth-security. NIST Special Publication, May 2017.

[8] Evgeny Tsimbalo, Xenofon Fafoutis, and Robert J Piechocki. Crc error correction in iot applications. *IEEE Transactions on Industrial Informatics*, 13(1):361–369, 2017.

[9] ESP32 Sleep Mode— Espressif Systems. (Accessed on 04/26/2020).

[10] Bluetooth. *Bluetooth 5.0 Specification*, December 2016.