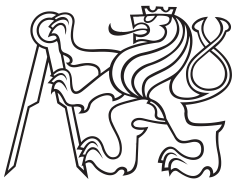
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of control engineering

Wireless Internet of Things device for vibration monitoring

Aleksandr Simakov

Supervisor: doc. Ing. Radislav Šmíd, Ph.D.
Field of study: Cybernetics and Robotics
Subfield: Systems and control
May 2017

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Simakov Aleksandr**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Bezdrátový IoT modul pro monitorování vibrací**

Pokyny pro vypracování:

Navrhněte a realizujte zařízení pro snímání vibrací pomocí piezoelektrického akcelerometru, digitalizaci signálu a výpočetní zpracování založené na mikrokontroléru řady STm32 s malou spotřebou. Měřicí modul s bateriovým napájením bude komunikovat s nadřazeným pomocí rozhraní Bluetooth LE, nadřazený modul napájený ze sítě bude připojen do WAN pomocí WiFi rozhraní. Kromě přímého přístupu by mělo být možné data ukládat do IoT cloudu ThingSpeak.

Seznam odborné literatury:

[1] firemní literatura STmicroelectronics
[2] R. B. Randall: Vibration-based Condition Monitoring: Industrial, Aerospace and Automotive Applications, Wiley 2011
[3] J. G. Ganssle: The Art of Designing Embedded Systems, Newness 2008

Vedoucí: doc. Ing. Radislav Šmíd, Ph.D.

Platnost zadání: do konce letního semestru 2017/2018

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S. -4-

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
Děkan

V Praze, dne 30. 1. 2017

Acknowledgements

First of all, I would like to thank my supervisor doc. Ing. Radislav Šmíd, Ph.D. for his amazing support, all the advice and patient guidance throughout the entire project. I have been extremely lucky to have such an intelligent, polite and friendly supervisor, who came out not only as a talented teacher but as an outstanding manager as well. It was a great honor to have a chance to work with him. Another person who I would also like to thank for introducing to me the world of microprocessors is doc. Ing. Jan Fischer, CSc.

This thesis would not exist without nice people and friends around me. A big thank you to: Miroslav Kubíček, Danil Merzlov, Aleksandr Koxin for making the study at CTU more fun. I would also like to thank Ondřej Hruška for sharing his experience about ESP8266 Wi-Fi module.

Special thanks to my wife who has always been supporting me despite anything.

Declaration

I declare that I wrote the presented thesis on my own and that I cited all the used information sources in compliance with the Methodical instructions about the ethical principles for writing an academic thesis.

Prague, 25. May 2017

Abstract

The presented thesis is dedicated to a creation of an experimental IoT platform with low energy consumption, which monitors vibrations and sends processed data to the web server ThingSpeak.com via Wi-Fi. The aim of this thesis is to design and realize the platform, which will be able to monitor vibrations of machines that involve rotary mechanisms such as motors, pumps, compressors, fans, belt conveyors, and gearboxes.

This device has been realized with STM32 Nucleo-F411RE board and X-NUCLEO-IDB05A1 Bluetooth Low Energy expansion boards from ST-Microelectronics. The Wi-Fi module that sends data to the web server is ESP8266-WROOM-02 from Espressif systems, one of the cheapest Wi-Fi module on the market. The device has been realized with the option of being able to access the raw measurements directly via serial port.

Keywords: IoT, vibration monitoring, STMicroelectronics, Nucleo-F411RE, Bluetooth, ESP8266, embedded systems, Edge Node computing, ThingSpeak.com, dweet.io

Supervisor: doc. Ing. Radislav Šmíd, Ph.D.

Abstrakt

Tato diplomová práce je věnována vytváření experimentální IoT platformy s nízkou spotřebou, které monitoruje vibrace a odesílá zpracovaná data na webový server ThingSpeak.com prostřednictvím WiFi rozhraní. Cílem této diplomové práci je navrhnout a realizovat platformu, která by mohla monitorovat mechanické vibrace strojů, obsahující rotující se mechanizmy, jako například motory, čerpadla, kompresory, ventilátory, dopravní pasy a převodovky.

Toto zřízení bylo realizováno pomocí STM32 Nucleo-F411RE a X-NUCLEO-IDB05A1 Bluetooth Low Energy desek od firmy STMicroelectronics. Wi-Fi modul, který odesílá data na webový server, byl zvolen ESP8266-WROOM-02 od Espressif systems, jeden z nejlevnějších Wi-Fi modulů na trhu. Zařízení bylo realizováno s možností přístupu k nezpracovaným naměřeným datům přímo, prostřednictvím sériového portu.

Klíčová slova: IoT, monitorování vibrací, STMicroelectronics, NucleoF411RE, Bluetooth, ESP8266, vestavěné systémy, Edge Node vypočítání, ThingSpeak.com, dweet.io

Překlad názvu: Bezdrátový IoT modul pro monitorování vibrací

Contents

Figures

Tables

Chapter 1

Introduction

Nowadays more and more production lines are pretty close to fully automated, and human's intervention into a process is less frequent. This requires automatic fault detection algorithms to be involved as well as the devices being capable of keeping monitoring the states of a process for a long time in an autonomous mode, especially when the places where the devices are mounted are not so easily accessed and it is desired to mount them there once and for years. This requires in its turn an efficient energy management of the device as well as self-diagnostic capabilities. The opportunities with IoT are incredibly high nowadays. With growing availability of the internet all over the world, increasing speed of communication and decreasing the ping values the monitoring of different values has become almost real time. The number of connected devices worldwide is steadily growing and according to [6] will reach 50 billion devices by 2020.

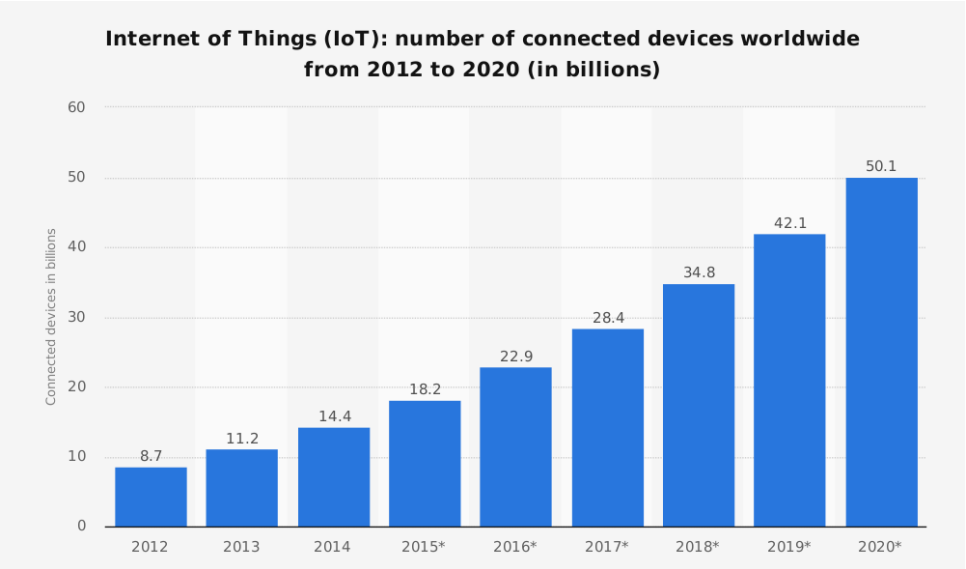


Figure 1.1: Number of connected IoT devices worldwide from 2012 to 2020 (in billions). The image is taken from [6].

1.1 Edge computing

IoT endpoints (i.e., devices with sensors) frequently do not have communication capabilities to transmit all their sensor data in a secure, reliable, and cost-efficient manner to the core. Here are a few of the most common obstacles :

- sensors may only support low energy protocols to conserve battery power
- mobile devices leveraging cellular communication lack coverage in certain locations
- mobile communication links are often bandwidth constrained or expensive
- wide area connections can introduce too much latency for real-time decision making

Furthermore, certain local systems, must autonomously make decisions in real-time in case a critical failure concerning safety and cannot wait for instructions sent from the cloud. Edge processing can address these challenges. An edge processing unit is a physical device, typically referred to as an IoT gateway, also called a fog node.

1.2 State of the art

1.2.1 Health monitoring at the edge

"The Industrial Internet of Things (IIoT) revolution is in full development as industrial providers continue integrating smart connected solutions into their existing operations. A high level challenge is that many IIoT solutions are tasked with capturing and moving massive amounts of data from a myriad of sensors to the cloud. Often, this creates an inadvertent crippling effect from a bottleneck deluge of raw data. With a dumb sensor approach, the typical "send everything to the cloud" model can be drastically inefficient. This often presents cloud platforms with the near impossible task of extracting a few morsels of valuable data from an overwhelming amount of noise."[5], - claims Ian Beavers from Analog Devices.

The intent of the solution is to identify and predict machine performance issues in advance of failure. At the edge sensor node, a multi-axis high dynamic range accelerometer can monitor vibrations at various locations on industrial machines. The raw data can be filtered and judged based on frequency domain interpretation within a microcontroller (MCU). A fast Fourier transform (FFT) compared against known performance limits can be processed for testing against pass, fail, and warning alerts downstream (Figure 1.2). Though machine maintenance can be performed on a regular basis, it is often not done with intelligence about a machine's health condition.

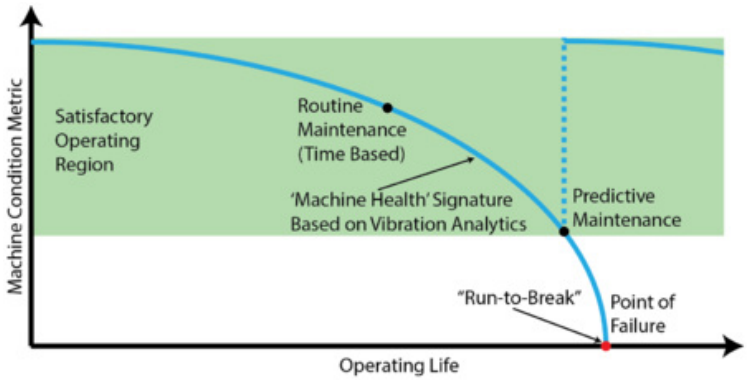


Figure 1.2: Machine operating life evaluation. The image is taken from[5]

By analyzing the vibration performance of a specific machine, Edge Node alerts can provide points of a potential failure.

If mechanical signature frequencies of interest are known precisely, the sample rate of the analog-to-digital converter (ADC) and FFT size within the MCU can be planned such that the maximum amount of energy falls within the width of a single histogram bin. This will prevent signal power from leaking across multiple bins and diluting the precision of the amplitude measurement[5].

The figure 1.3 is an example of an FFT where specific predetermined zones are interpreted within the Edge Node MCU for more than one observed mechanical component. Bin energy that peaks within the required green zone represents satisfactory operation, while the yellow and red zones indicate warning and critical alarms, respectively. Instead of transmitting the full sensor bandwidth, a lower data rate alarm or trigger breadcrumb can alert the system of an excursion event within the zones of interest [5].

Figure 1.3: FFT bin energy can be used to trigger alarms. The image was taken from [5]

1.2.2 DIY projects

There are a lot of similar DIY devices on the internet especially on this, so called, ”DIY heaven” portal `www.instructables.com`. Most of them are based on Arduino, though. With the operational frequency of 16MHz and few kilobytes of SRAM in Arduino, those projects are only useful as study material and are not very much applicable in industrial scale.

The available on the internet DIY projects have several key disadvantages to be used in industry, mainly due to the fact they they have an educational purpose. These devices are usually too big in size and do not have energy consumption optimization making a human to be locked to a certain solution.

There is plenty of projects with using ESP8266 as a Wi-Fi module on the internet as well, some of which turned out to be useful at the beginning of this thesis.

1.2.3 Industrial vibration monitoring

Apart from the IoT vibration monitoring devices, a typical industrial vibration monitoring system would look like as it is offered by the company ”ifm”© and are shown on the figure 1.4.

The electronic vibration monitor monitors online the overall vibration condition of machines and equipment according to DIN ISO 10816. The sensor measures the rms vibration velocity on a non-rotating component surface. When an adjustable limit value is exceeded the unit sets to alarm via a switching contact. In addition the characteristic value is provided as current signal (4...20 mA) for connection to the process control system. The figures and the description are taken from [9].

(a) : MEMS accelerometer from ifm.

(b) : Typical vibration sensor setup.

Figure 1.4: Principal scheme of the device.

1.3 Aims of this thesis

The aim of this thesis is to create an experimental platform under the following conditions:

- The platform must be based on STM32 microcontroller
- Allowing to monitor vibrations with an analog accelerometer, digitize and process the data
- The measuring module must be powered by batteries and must communicate with the connected to WAN module via Bluetooth Low Energy.
- The connected to WAN module must be powered from a regular 230V plug and must be able to send the important processed part of the data to the IoT cloud service ThingSpeak.com.
- The measured data must also be available directly from the measuring module.

A small amount of data transmitted wirelessly will be advantageous compared to a wired connection, especially if there is a huge motor with multiple points to monitor, for instance. The mounting place of an accelerometer might also be hardly accessible making wireless data transmission more comfortable to use. There are multiple options for wireless data transmission, Wi-Fi and Bluetooth are among them. In this thesis, it is proposed to implement the Edge Node with Bluetooth interface which will send processed data to the IoT Wi-Fi Gateway via BLE [1]. This solution allows a potential enlarging of the net of the Edge Nodes using, for example, MQTT communication protocol. On the one hand, using BLE on the IoT Edge Node side instead of Wi-Fi shortens the range of the possible devices location, on the other hand, this allows to significantly reduce power consumption making it viable to run the Edge Node on batteries for a long time. The IoT Wi-Fi Gateway will collect data from the Edge Node and use Wi-Fi interface to upload data to the web services.

[1] Bluetooth Low Energy

Chapter 2

Concept and structure of the device

In this chapter the main nodes of the device and their roles are going to be described. The overview of the main software and hardware components is going to be presented. In the end the main IoT cloud platforms are going to be mentioned.

2.1 Principal scheme of the device

The suggested structure of the device consisting of the IoT Edge Node and the IoT Wi-Fi Gateway is presented on the figure 2.1.

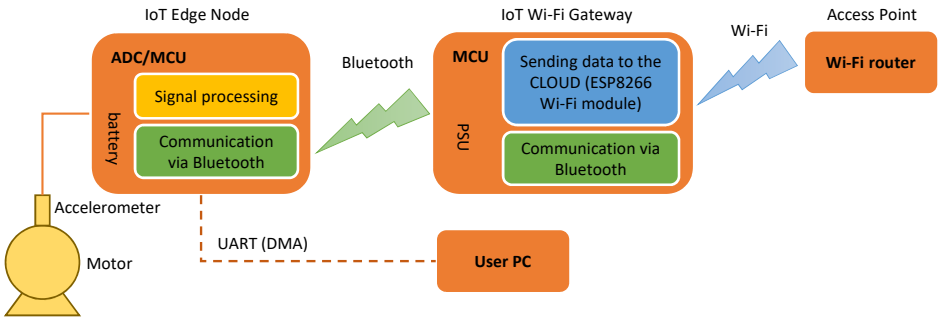


Figure 2.1: Principal scheme of the device.

The "Signal processing" block includes obtaining data from an analogue accelerometer via AD converter with a certain sampling frequency and execute signal processing algorithms (FFT, RMS, Crest Factor). The green "Communication via Bluetooth" block on the Edge Node side is responsible for sending processed data by packets to the IoT Wi-Fi Gateway at a certain time. This block also triggers sampling when it received the command to do so from the Gateway.

The green "Communication via Bluetooth" block on the IoT Wi-Fi Gateway side is responsible for receiving all the packets from the IoT Edge Node and transform them into one buffer, and sending it to the ESP8266 Wi-Fi module afterwards. This block also sends command to the IoT Edge Node to start

sampling as soon as it gets confirmation from the ESP8266 Wi-Fi module that it is ready to send data to one of the web servers.

The blue "ESP8266 Wi-Fi module" block is responsible to connect to Access Point, to receive data and to generate different HTTP/1.1 requests for different web servers. This block also initializes the chain of events leading to the start of sampling on the Edge Node side by notifying the Gateway MCU that it is ready to receive the next data.

The device will consist of 2 boxes and can be potentially extended to a larger number. The IoT Wi-Fi Gateway box consists of the STM32-Nucleo-F411RE board, X-NUCLEO-IDB05A1 BLE expansion board both from STMicroelectronics and developed PCB with ESP8266-WROOM-02 Wi-Fi module on it. This box will receive data from other SERVER (IoT Edge Nodes) boxes and will send important information about the state of the monitored equipment to one of the web services, either `www.dweet.io` or `ThingSpeak.com`.

The IoT Edge Node box will be designed to work in an autonomous mode and will be supplied with energy from four AA batteries. It will consist of the STM32-Nucleo-F411RE board, X-NUCLEO-IDB05A1 BLE expansion board and a developed PCB with DIN connector to connect a 3-dimensional accelerometer. The detailed principal scheme of the device is presented on the figure 2.2.

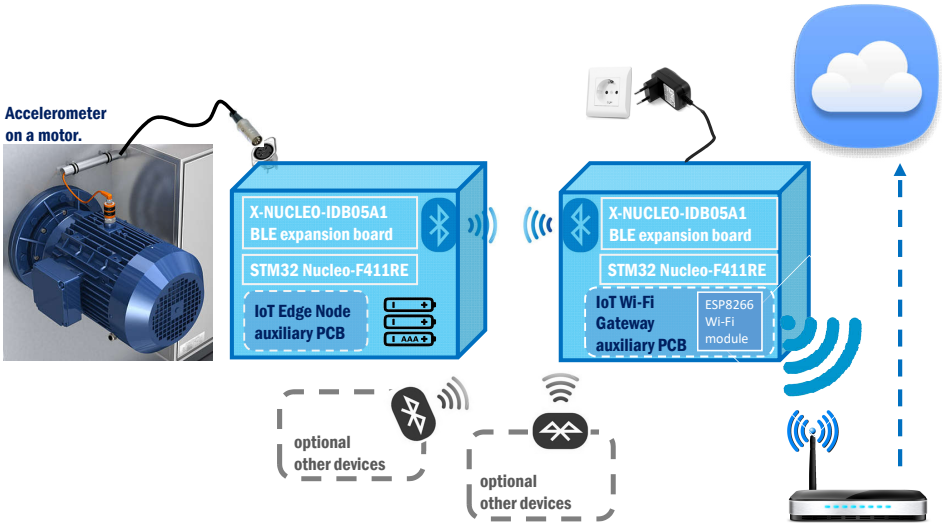


Figure 2.2: Detailed principal scheme of the device.

2.2 IoT cloud platforms

Developing for the Internet of Things is a complex endeavor, and nobody wants to do it from scratch. IoT data platforms offer a jumping-off point by combining many of the tools needed to manage a deployment from device management to data prediction and insights into one service. [8]

2.2.1 Proprietary services

IoT Cloud Platform solutions backed by large publicly traded companies:

- Microsoft Azure IoT Suite
- IBM Watson IoT
- Xively
- Google Cloud IoT
- Autodesk Fusion Connect
- AWS IOT
- ThingWorx
- Zebra Zatar Cloud

2.2.2 Open Source IoT platforms

Among the data management services with open source licenses are:

- macchina.io
- SiteWhere
- ThingSpeak
- Dweet.io

ThingSpeak.com offers 7 license options, the main are:

- Free version. Allows to upload data every 15s, multiple channels with a possibility to send up to 3 million message per year.
- Student paid version (55 €). Allows to upload data every 1s, multiple channels with a possibility to send up to 33 million messages per year.

Dweet.io offers three types of services:

- Developer free version. Allows to upload data every 5s for one device without data storage available. Only last 5 cached ”dweets” are kept and that is for 5 hours. Allows sending alerts via e-mail in case of crossing a pre-set threshold.

- Professional paid version (actually `dweetpro.io`). Supports up to 100 devices, allows to update data every 1s per device with unlimited 30 days data storage and other features.
- Enterprise paid version (actually `dweetpro.io`). Custom version for more than 100 devices.

Chapter 3

Communication software

The goal pursued in this chapter is to describe the communication aspect of the device. There are three main sides of the communication inside the experimental platform. The first one is the IoT Edge Node. It can send data to the IoT Wi-Fi Gateway via BLE (Regular mode) or to a user PC (Raw data mode). The second side is the CLIENT, which receives data from the SERVER via BLE and sends them to the ESP8266 Wi-Fi module via UART. The ESP8266 Wi-Fi module is the third side of the communication chain, it is responsible for sending data to web IoT services. The software for the Nucleo board in the IoT Wi-Fi Gateway and in the IoT Edge Node have been developed in SW4STM32 - System Workbench for STM32: free IDE, by adapting the sample codes from STMicroelectronics. [14]

3.1 Wi-Fi communication

This section is dedicated to Wi-Fi module firmware development whos task is to transfer the processed data from some edge node sensor to the web IoT services. At first the overview of the ESP8266 Wi-Fi module is going to be presented.

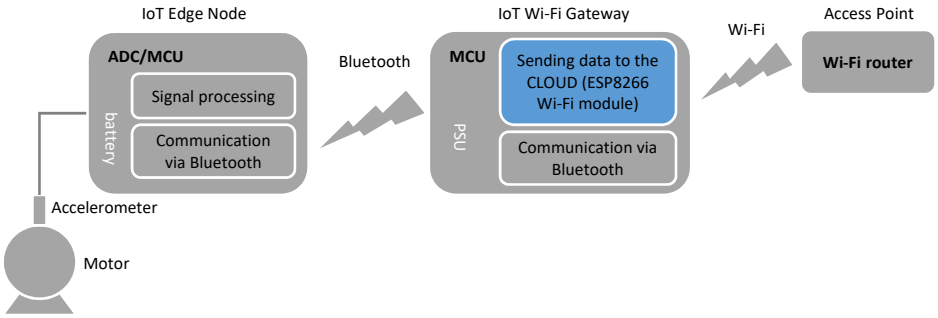


Figure 3.1: ESP8266 Wi-Fi module as a part of the device.

3.1.1 ESP8266 Wi-Fi module overview

The ESP8266 Wi-Fi module is a product of the company Espressiff, which a few years ago exploded the interest to feature devices with internet connection due to its affordable prices. The documentation available on the internet is sometimes controversial and not very clear but it is getting better every month. The community who uses this device is growing very fast, and thus the quality of the documentation is getting better with each release. There are few options with the firmware choice. Thanks to free available ESP8266 SDK (Software Development Kit) there are few ready firmwares for easier development, for example, NodeMCU (`https://github.com/nodemcu/nodemcu-firmware`) or ESP Easy (`https://www.letscontrolit.com`). NodeMCU is using scripting language Lua to build a binary file and then flashes it to ESP8266. ESP Easy allows flashing its firmware with the opportunity to change settings of connected sensors on web interface hosted by ESP8266 through a router. But the most common option is flashing with the Arduino.

ESP8266 Wi-Fi module is present in different variations, they differ by the amount of memory available, antenna type and amount of GPIO pins. Some of them are on the figure 3.2.

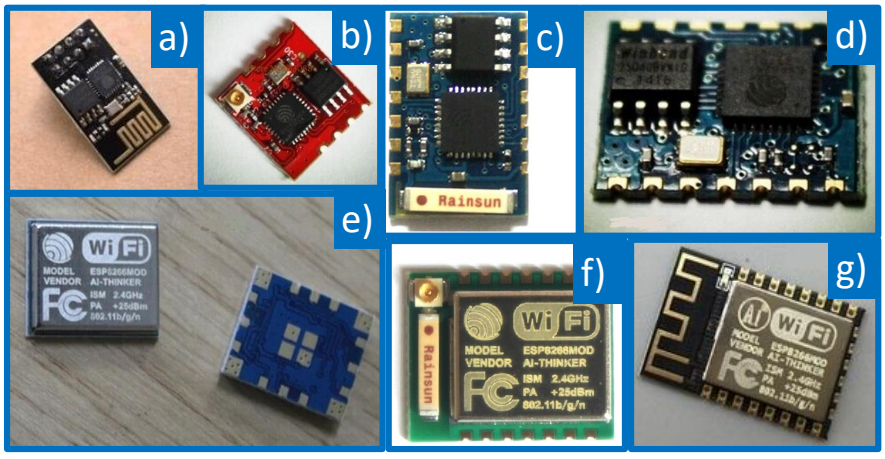


Figure 3.2: Different types of ESP8266 modules: a) ESP8266-01; b) ESP8266-02; c) ESP8266-03; d) ESP8266-04; e) ESP8266-06; f) ESP8266-07 g) ESP8266-12F.

3.1.2 ESP8266 firmware development

The ESP8266 module has a default pre-installed firmware which is used in most of the projects on the internet. That firmware is based on AT commands and is only good for small projects and the projects with very few features.

For our purposes, it is better to be flexible with possibilities of the module and built a custom firmware to fulfill the task. Espressif offers two types of the Software Development Kits [22]

```
POST /dweet/for/esp8266_simakale HTTP/1.1
Host: dweet.io
Connection: close
Content-Type: application/json
Content-Length: LENGTH_OF_JSON_STRING


{
"Variable1": "value1",
"Variable2": "value2",
...
}
```

- Non-OS SDK. It is not based on an operating system and uses timers and callbacks as the main way to perform various functions - nested events, functions triggered by certain conditions.

- RTOS SDK. It is based on FreeRTOS, a multi-tasking OS. One can use standard interfaces to realize resource management, recycling operations, execution delay and other features.

There are also alternatives to Espressif's official SDK. To fulfill the task of this thesis it was decided to use the open source Non-OS *ESP-open-SDK* [19], which is based on the GCC toolchain. The repository with this project provides the integration scripts to build a complete standalone SDK (with toolchain) for software development. It also provides sufficient information regarding the installation and usage of that source.

ESP8266 WiFi module uses UART0 to communicate with the Nucleo board inside the IoT Wi-Fi Gateway box. When the data come from the Nucleo, the UART interrupt handler calls the functions that will transform the data into different strings, based on which web server the data are going to be uploaded.

Dweet.io

The sample code, publicly available on [20], was adapted and used as a base for this thesis. That sample is capable of connecting to Access Point and sending a JSON string to dweet.io web service in the form, shown on top of the page.

One of the jobs that ESP8266 Wi-Fi module does once it receives the message from the CLIENT is it picks up certain values from that buffer and transforms them into a JSON string, which will later be appended to the POST request.

ThingSpeak.com

The firmware for uploading data on ThingSpeak.com was built based on the sample code for `dweet.io`, although it needed more modification due to the difference in the requirements of those servers. The sample request, which was used to send data to `ThingSpeak.com` web server is shown below.

```
GET https://api.thingspeak.com/update?
api_key=MY_API_KEY&field1=value1&field2=value2...

HTTP/1.1

Host: https://api.thingspeak.com
Connection: close
```

The API key is available in the main menu once a user gets registered on the portal. One of the jobs that ESP8266 Wi-Fi module does once it receives the message from the Nucleo board inside the IoT Wi-Fi Gateway box is that it picks up certain values from that buffer, inserts them instead $value1...value2...$ fields generating a big string which later on gets appended to the GET request and sent to the web server.

3.1.3 ESP8266 firmware flashing

The example for *dweet.io* comes with a Makefile, which must be modified with the paths to the toolchain and the flashing tool (esptool.py). The flashing tool in its turn provides possibilities of speeds to flash a firmware via UART. It supports up to 921600 baud rate, but it happens that the firmware is flashed with errors. The compromise between speed and successful flashing is 115200 baud rate, which was used to flash the firmware.

To flash a firmware, the $IO0$ Pin of ESP8266 must be pulled down during start up. For convenience, there is a $SW2$ switch implemented on the printed circuit board (figure 5.8). To boot the module from the flash memory, the $IO0$ Pin must be pulled up or floating.

3.2 Bluetooth Low Energy

In this section the communication between the IoT Edge Node and the IoT Wi-Fi Gateway is going to be discussed. The key points of BLE communication and the BLE GAP roles are going to be presented.

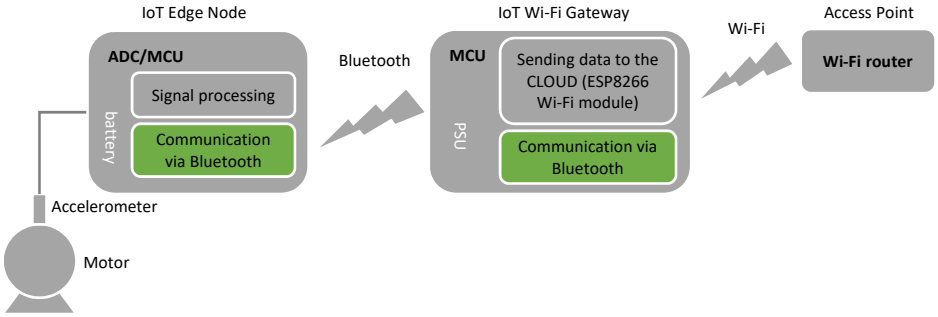


Figure 3.3: Bluetooth Low Energy modules as parts of the device.

Bluetooth low energy is the power-version of Bluetooth that was built for the Internet of Things (IoT). The power-efficiency of Bluetooth$^{\text{ő}}$ with low energy functionality makes it perfect for devices that run for long periods on power sources, such as coin cell batteries or energy-harvesting devices. Native support for Bluetooth technology on every major operating system enables development for a broad range of connected devices, from home appliances and security systems to fitness monitors and proximity sensors [10]

3.2.1 Bluetooth Low Energy GAP Roles

Bluetooth Low Energy devices can operate in one or more Generic Access Profile (GAP) roles at the same time (provided the Link Layer supports this):

- Broadcaster
- Observer
- Peripheral
- Central

The role imposes restrictions and enforces behavior, so it is generally fixed in the design stage of the device [13]. Two role pairs (”Broadcaster/Observer” and ”Peripheral/Central”) are defined, allowing devices to communicate with each other. The pair ”Peripheral/Central” is suitable for this thesis, since it assures a protected communication between the two modules. Moreover, one of possibly desired modes as a future work proposal could be to send processed data with the periodicity up to couple hours. During that time there is no need to advertise the data and it can save power. If there were more modules involved in the net, there were no need in keep data private and the periodicity of exchanging data is quite small, then one could think about creating the ”Broadcaster/Observer” pair.

Broadcaster/Observer pair implements unidirectional, connection-less communications.

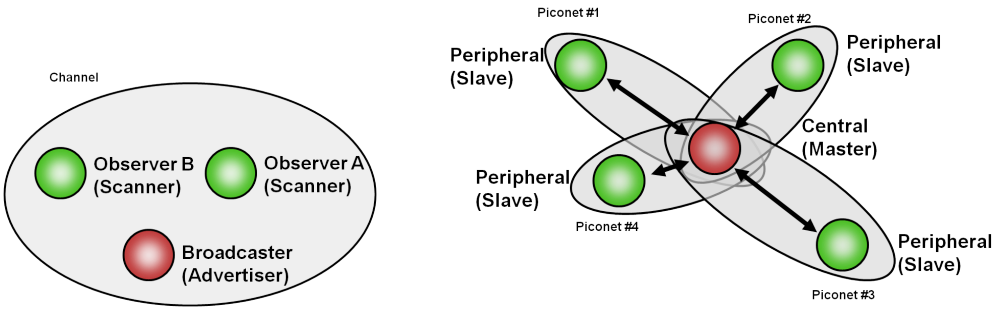
- *broadcaster (advertiser)* periodically sends advertising packets with data.

- *observer (scanner)* scans for broadcasters and listens for advertising data.

Peripheral/Central pair implements bidirectional, connection-oriented communications.

- *peripheral* device plays Slave role in the Link Layer and advertises by using connectable advertising packets. It is optimized to consume the least amount of processing power and memory.
- *central* device plays Master role in the Link Layer. It is capable of establishing and managing a connection and may be connected to various devices simultaneously.

The visualization of the above mentioned pairs is shown on the figure 3.4.



(a) : Broadcaster/Observer **(b) :** Peripheral/Central

Figure 3.4: Network Topology of the Broadcaster/Observer and Peripheral/-Central pairs. The images were taken from [13]

3.2.2 Sending float type values

Float type values are built of 4 bytes. To be able to send floats they have to be split into 4 pieces 1 byte each. This can be implemented by using unions, a special data type that allows to store different data types in the same memory location. The example of splitting a float type variable into 4 bytes is demonstrated below.

```
1 union {
2         float   f;
3         uint8_t ch[4];
4 } floatNchars;
```

3.3 IoT Edge Node's communication modes

The IoT Edge Node (the "SERVER" if using BLE GAP role terminology) can work in two modes:

a) **Regular mode**. Sending the processed data to the IoT Wi-Fi Gateway, which will send it to the cloud.

b) **Raw data mode**. Sending raw data (directly from ADC converter) to the laptop using UART and DMA.

3.3.1 Raw data mode

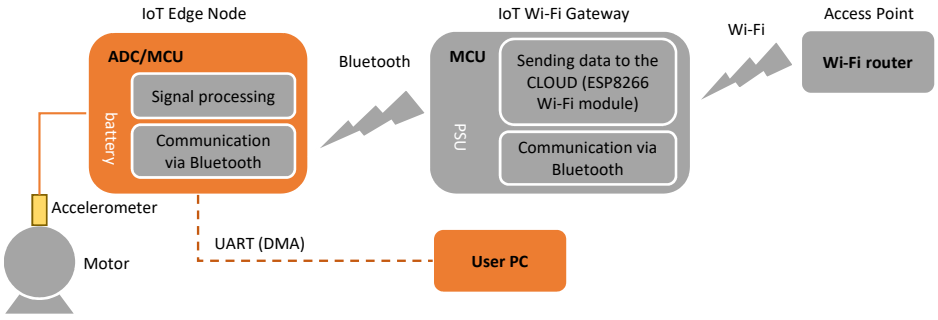


Figure 3.5: Active pieces of the device working in raw data mode.

Direct readings of the accelerometer in a time domain might also be of an interest. In this thesis it is going to be implemented in the following way.

At the start up the User Button is checked whether it is pushed or not. If yes, the Edge Node goes into raw data mode and starts sampling until it fills in the buffer. As soon as it is full it sets the variables $ready2send = 1$ and $fft\_buf\_full = 1$. A simple counter is running in the SysTick interrupt handler called every $1ms$. The counter is reset every time it reaches or for some reason goes over 5000 counts. In the main loop the program waits until the counter reaches 5000 ticks ($5s$) and checks the $ready2send$ variable. If it's TRUE, it sends data via UART DMA to serial port. The scheme of this process is shown on the figure 3.6

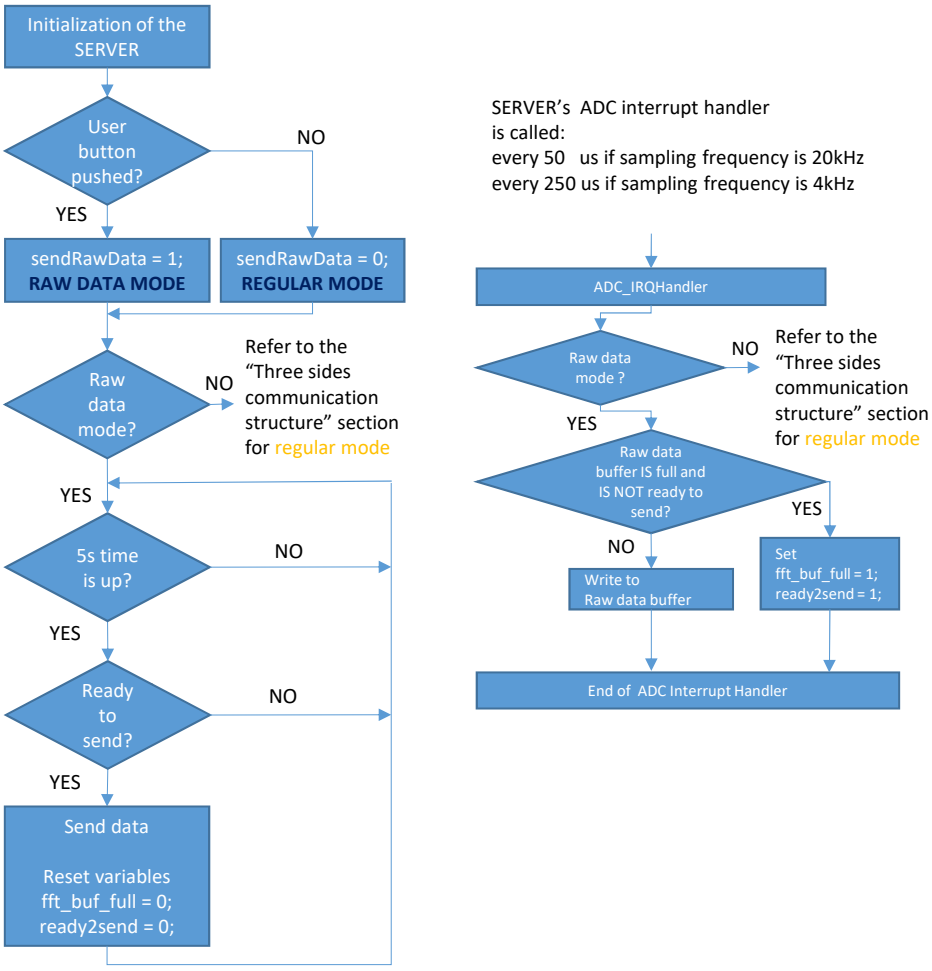


Figure 3.6: Main program and ADC interrupt handler logical scheme in raw data mode.

The STM32-Nucleo-F411RE board has 12-bit AD converter (the maximum number it can get is 4095), thus the values should be written into at least $uint16\_t$ type variables. In order to be able to send this values via UART each value must be split into two pieces: first 8 bits piece is obtained by right shifting an $uint16\_t$ value, the second is 8 remaining bits. The method of doing so is shown below:

```
1 RawData_buf[fft_buf_idx]     = adc_x >> 8;
2 RawData_buf[fft_buf_idx + 1] = adc_x;
3 RawData_buf[fft_buf_idx + 2] = adc_x >> 8;
4 RawData_buf[fft_buf_idx + 3] = adc_y;
5 RawData_buf[fft_buf_idx + 4] = adc_z >> 8;
6 RawData_buf[fft_buf_idx + 5] = adc_z;
7 fft_buf_idx += 6 ;
```

where adc_x, adc_x, adc_z are the re-scaled to $0V...3V$ readings from the

AD converter, the fft_buf_idx is incremented each time the ADC interrupt handler is called, the $RawData\_buf$ is actually the buffer which will be send via UART DMA later on. One should keep in mind the way this buffer was built once trying to read the data on the PC side with the help of MATLAB for example.

3.3.2 Regular mode. Three sides communication structure. SERVER – CLIENT – ESP8266 Wi-Fi module

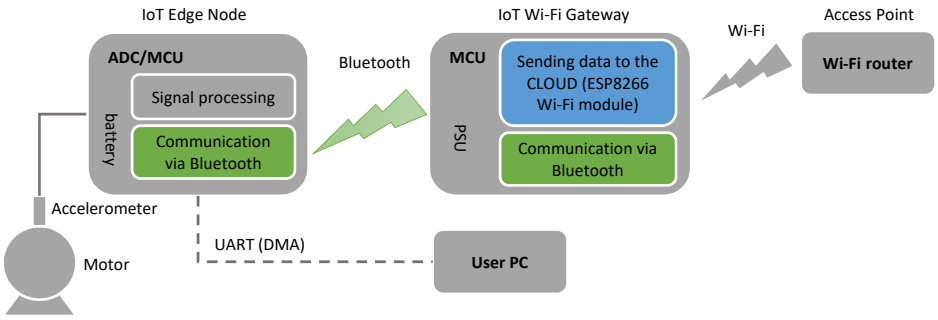


Figure 3.7: 3 side communication overview.

The Nucleo-F411RE and ESP8266 Wi-Fi module communicate via UART at 115200 baudrate. For the amount of data necessary to transfer, this standard of serial communication is enough. The two Nucleo-F411RE boards communicate via BLE.

A sample program "SampleAppThT" developed by STMicroelectronic has been adapted as a base for the software development for this thesis. This program is available as a part of "Bluetooth Low Energy software expansion for STM32Cube" [14]. The copyright notices about where the sample program is coming from are kept in the headers of all modified files.

The chain of the communicating devices consists of the IoT Edge Node (SERVER) and IoT WiFi Gateway (CLIENT + ESP8266 WiFi module). The CLIENT needs to have some rules of communication with the WiFi module connected to it as well as with the Edge Node. The simplified model of the communication rules is shown on the figure 3.8.

The IoT WiFi Gateway and the IoT Edge Node can start their work independently. On every start up the CLIENT will hold ESP8266 WiFi module in a reset state during the complete initialization and let it turn on after the initialization is complete. The ESP8266 WiFi module in its turn will have 55s to return "Ready" message to the CLIENT. The timer is implemented as a simple counter in a Systick interrupt handler. There can be multiple reasons for not receiving this message in time, but the most common one is the disconnect from the Access Point or not being able to connect to it due to some problems on the Access Point side. If the message doesn't come in time, the interrupt calls ErrorHandler function, where the

timer, responsible for sending pulses to the watchdog's WDI Pin (refer to the 5.2.3 subsection for more details on this Pin), gets deactivated leading to the entire system restart. This makes it possible to use one hardware watchdog to watch both devices, ESP8266 and the CLIENT.

The CLIENT's initialization finishes much earlier than the "Ready" message comes, and it can connect to the SERVER in the meantime. After both conditions are met, the "Ready" message is received, and the CLIENT is connected to the SERVER via BLE, the CLIENT triggers the sampling on the SERVER's side by sending "CLIENTREADY" message to it. After filling the buffer the SERVER starts signal processing algorithms, it calculates RMS, Crest factor and FFT. Then it sends a starting message "StartMSG" to the CLIENT notifying that there is going to be 96 bytes coming in packets. The CLIENT on its side starts $500ms$ timer. This time is allotted for all the data to be transferred from the SERVER and normally it takes $10 - 110ms$ depending on the clock rate on which the SERVER runs. The CLIENT controls the number of packets it has received and looks at the time when the last packet has come. If one of the packets gets lost, the expected amount of packets will be met only after the next sampling on the SERVER side. The next sampling will be triggered by the chain of events, starting from ESP8266 "Ready" message to the CLIENT (which is sent every $5s$ if we broadcast to dweet.io or $15s$ if we broadcast to ThingSpeak.com) leading to sending "CLIENTREADY" message to SERVER from the CLIENT. Therefore the time to receive 96 bytes will be way over $500ms$ and everything that was received before will be ignored, and the CLIENT will wait for another new 96 bytes.

After sending the starting message to the CLIENT, the SERVER starts searching five highest amplitudes and the corresponding frequencies (the $MAXFREQ\_NUM$ parameter in the program for the SERVER) of the spectrum obtained from FFT and sends to the CLIENT twelve pairs of values. The first pair is RMS and $CrestFactor$ values; the second is the first highest amplitude from the spectrum and the corresponding frequency and so on. Each value is of a float type; thus, each packet is 8 bytes, giving in total 96 bytes to be transferred to the CLIENT. As soon as the CLIENT received all the data it is then written to one 96 bytes buffer and is sent to the ESP8266 WiFi module as soon as it's ready.

That data triggers the UART interrupt on the ESP8266 side where the received data is used in different ways, depending on what server (dweet.io or ThingSpeak.com) we are going to broadcast the data.

After sending the data to the web server, the TCP connection is getting closed, which calls $tcp\_disconnect()$ callback function, where 5 or 15 seconds timer is started again. When the time is up, the new "Ready" message is sent to the CLIENT triggering the chain of events, and this process goes over and over endlessly. The block of this chain of events described above is marked with bold blue dash line on the 3.8 figure.

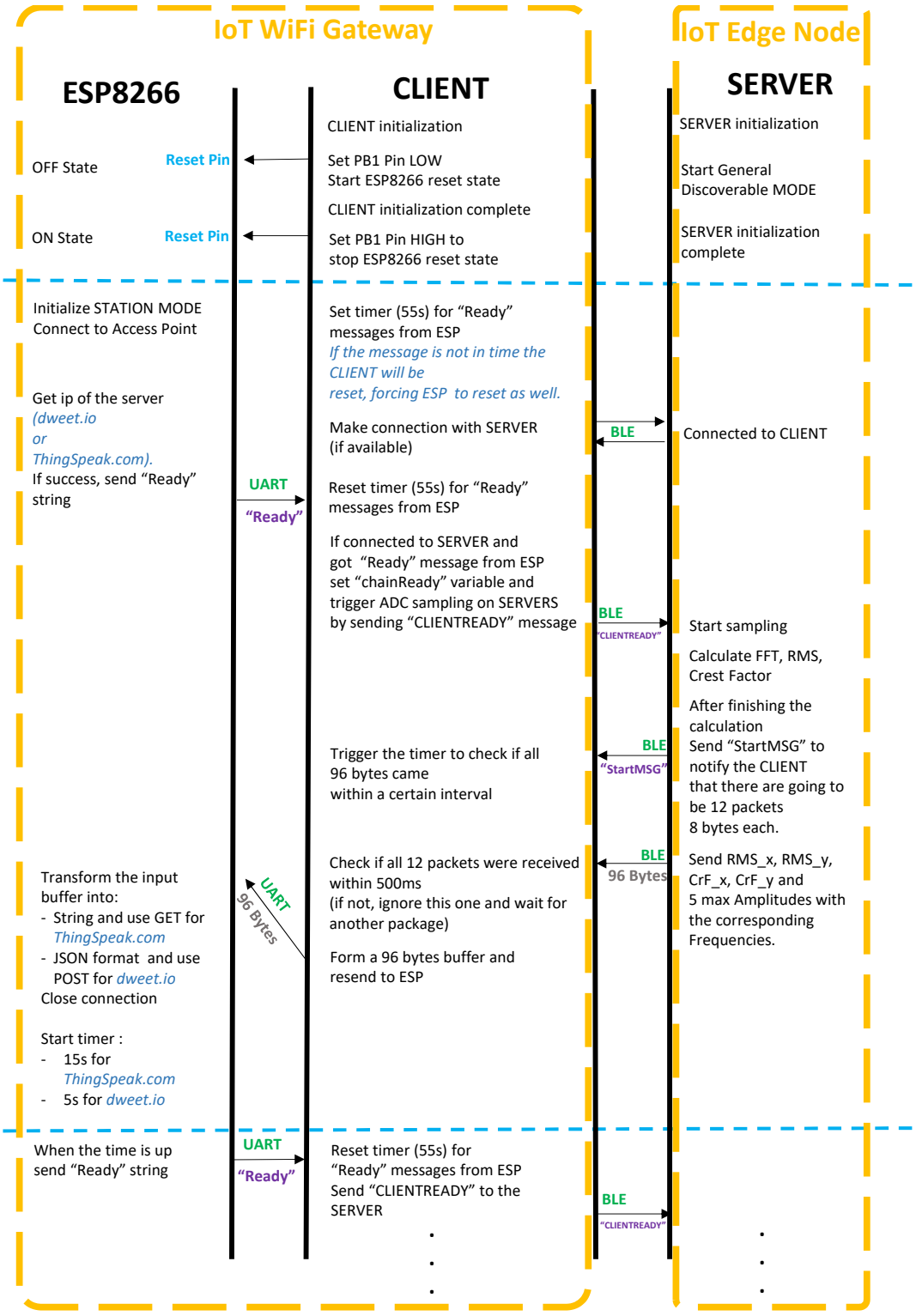


Figure 3.8: Three side communication setup.

Chapter 4

Vibration signal processing

In this chapter the signal processing block of the IoT Edge Node will be described. The overview of the accelerometers that were used during the thesis development will be presented. The main issues of an accelerometer mounting will also be discussed. The signal processing algorithms are chosen to be DFT, RMS and Crest Factor. Since the amount of data is going to be limited by only important information, only few first amplitudes from the spectrum are going to be send to the IoT Wi-Fi Gateway. The algorithm responsible for searcing that valueble data will also be presented in this section.

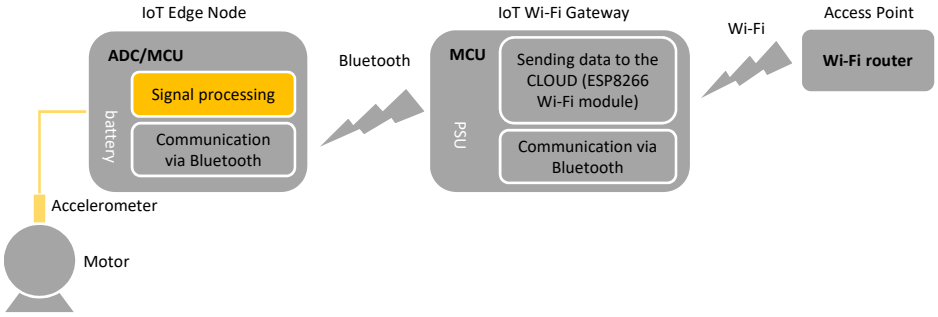


Figure 4.1: Signal processing block with accelerometer as a part of the device.

4.1 Accelerometers

Accelerometers can be classified by the physical principals, based on which they are designed. The most common accelerometers for the diagnostical purposes are

- **piezoelectric**
- **piezoresistive**
- **capacitive**

Parameter	Conditions	Min.	Typ.	Max.	Unit
Measurement range		±3	±3.6		g
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	$V_S = 3V$	270	300	330	mV/g
Bandwidth X_{OUT}, Y_{OUT}			1600		Hz
Bandwidth Z_{OUT}			550		Hz
Sensor Resonant Frequency			5.5		kHz

Table 4.1: Electrical characteristics of the analog accelerometer ADXL335.

Piezoelectric accelerometers use piezoceramics (namely $PbZrO_3$ or $PbTiO_3$) or single crystals (for example, SiO_2) and are advantageous at higher frequencies. Besides that, they have relatively low weight and high temperature range.[1] **Piezoresistive** accelerometers are preferred in case of estimating large amplitudes of the vibrations. **Capacitive** accelerometers are mainly MEMS[2] devices. They are advantageous in the low frequency range and they are very sensitive and precise.

At the laboratory test it was decided to use analogue MEMS accelerometer from Analog Devices© ADXL335. The key characteristics are shown in the table 4.1, more detailed description can be found in the official data sheet [17]. The accelerometer, wired with DIN connector is shown on the figure 4.2.

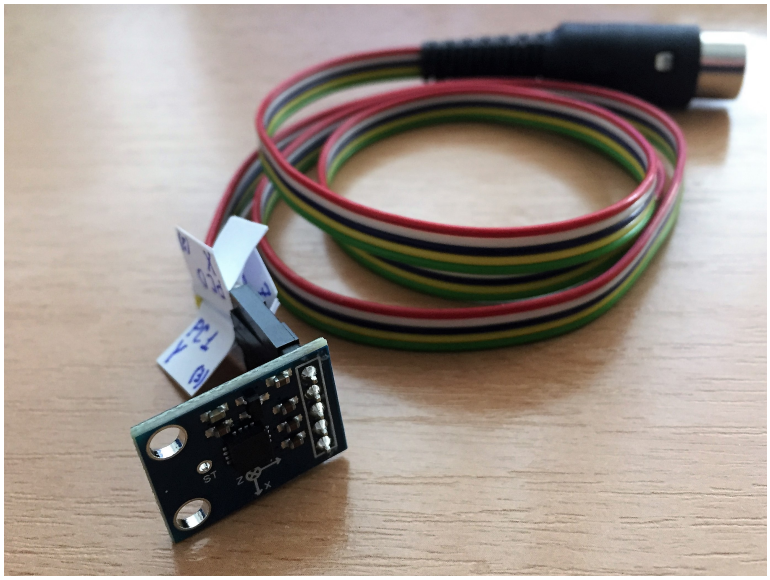


Figure 4.2: Analogue MEMS 3-axis accelerometer ADXL335.

This small printed circuit board with the accelerometer was ordered from ebay and has the recommended by the Analogue Devices electrical scheme,

[1] A much more detailed physical description of the accelerometers can be found in Czech technical literature "Technická diagnostika" by Marcel Kreidl and Radislav Šmíd [1], page 179.

[2] Micro Electro-Mechanical Systems.

which is shown on the figure 4.4. There is a Low Dropout CMOS Voltage Regulator LM6206N3 on the PCB, which reduces 5V power supply coming from the IoT Edge Node auxiliary PCB down to the required 3V.

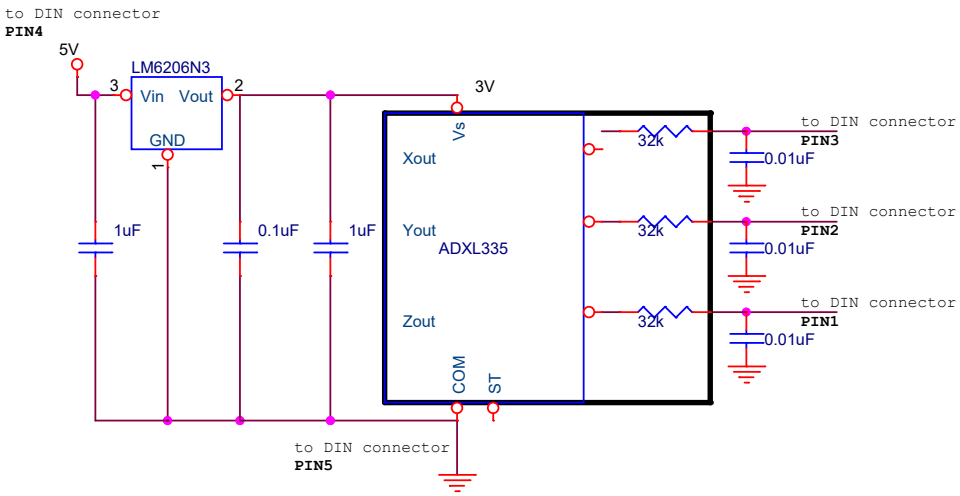


Figure 4.3: Electrical scheme of the printed circuit board for the ADXL335 accelerometer.

Another accelerometer, which was tested is piezoelectric ACH-01 from *measurement specialties*. It was used as a breadboard prototype but can easily be connected to one of three channels via DIN5 connector (refer to the 5.6 figure) after changing the gain of the output signal. More details on this accelerometer can be found in [18].

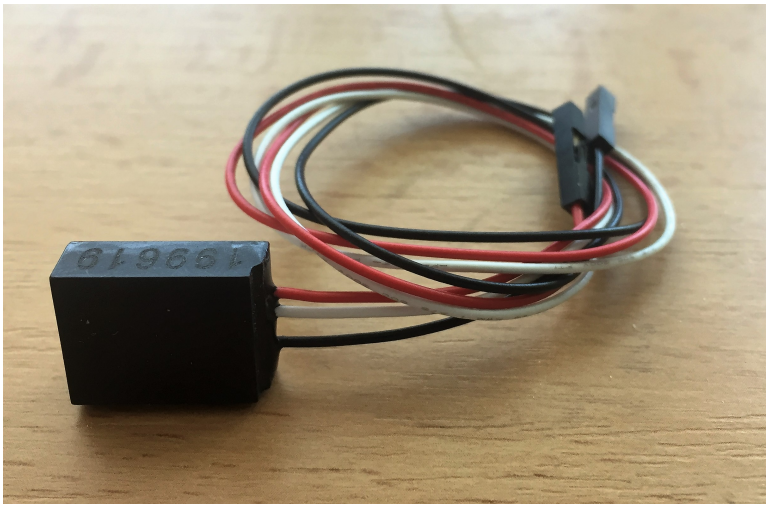


Figure 4.4: ACH-01 piezoelectric ultra low power accelerometer.

4.2 Accelerometer mounting tips

This whole condition monitoring process is useless if an accelerometer is mounted incorrectly no matter how sophisticated and advanced techniques have been used on signal processing stage. It goes without saying that whoever is going to use this device should be well aware of what one is going to measure and estimate. Wrong accelerometer placement can lead to overlooking of important issues.

If we assume that this device is going to be used on the motors, it is usually the bearing what is being monitored. When measuring vibration, the accelerometer must always be attached as close as possible to the bearing. Moreover, it must be mounted as close as possible to the center line of the bearing to avoid picking up distorted signals. The above mentioned idea is apparent from the figure 4.5

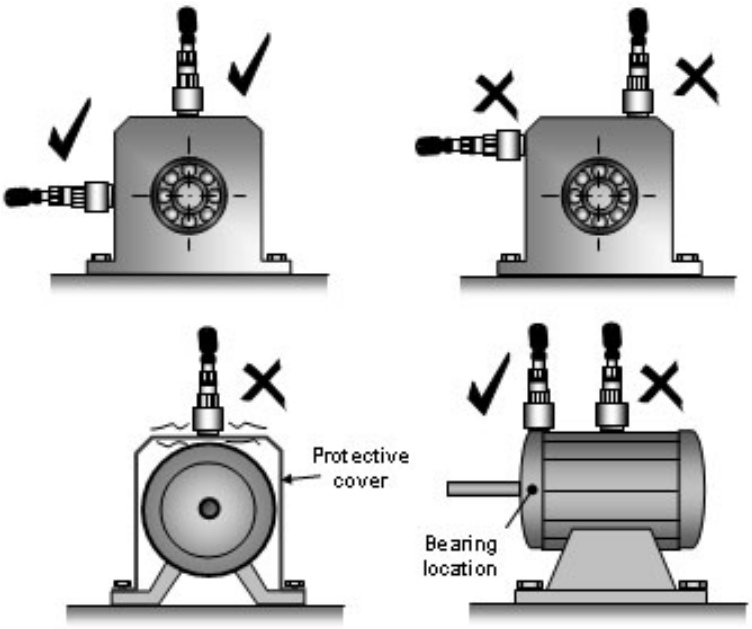
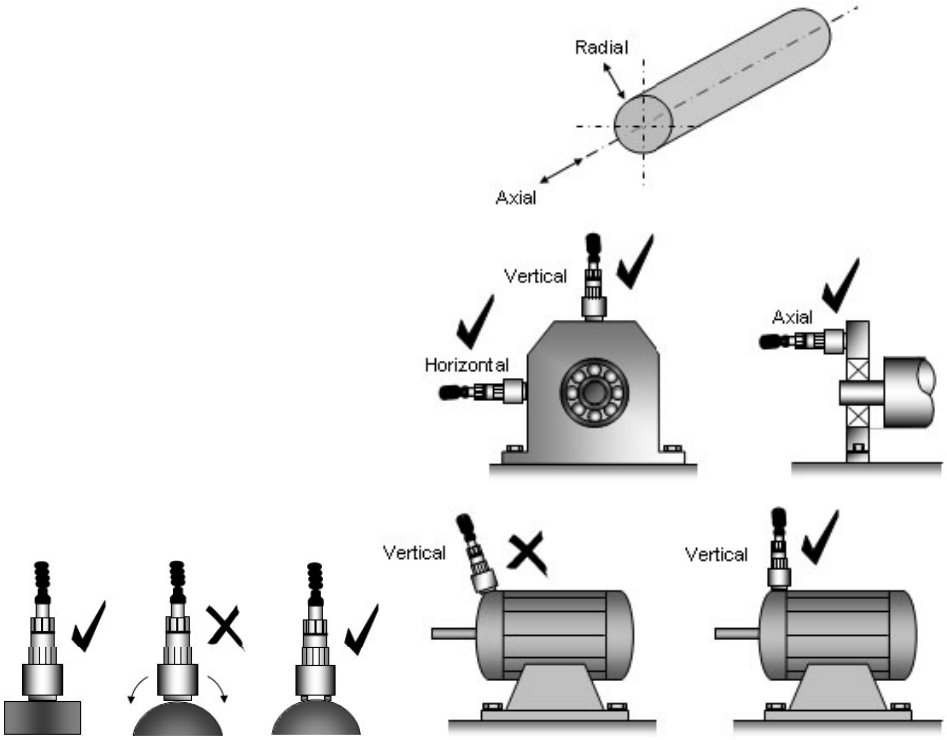


Figure 4.5: Proper versus wrong accelerometer mounting. The image was taken from [4].

For the accelerometer to measure true vibration behavior, it needs to undergo exactly the same vibratory movement as the vibrating unit. An accelerometer must, therefore, be attached firmly to the vibrating unit so that it does not move independently of the unit. The contact area must be free of all kinds of debris, rust, flaking paint etc. Different situations require the accelerometer to be oriented differently. For example, to detect parallel misalignment the accelerometer is usually mounted in the radial direction of the bearings, but to detect angular misalignment the accelerometer needs to be mounted in the axial direction. [4].



(a) : Firm and bad mounting **(b) :** Correct and wrong orientation

Figure 4.6: Other tips on mounting an accelerometer. The images were taken from [4].

Needless to say that if one is going to follow a certain machine parameter throughout a relatively long period, for the data to be representative to make judgments, the accelerometer must be placed at the same place.

4.3 Discrete Fourier Transform

One of the diagnostic tools, used at vibration monitoring, is Fast Fourier Transform. It can provide valuable information about what happens with the signal in frequency domain. The information hidden there is quite difficult if not impossible to see from the time domain.

The algorithm of Fast Fourier transformation was borrowed from [23]. It takes two buffers of complex numbers (the first is with actual data and the second is used as a temporary buffer to store values during sorting stage) and the number of elements (must be a power of two). The result of this function is an array of complex numbers which is later on used to calculate one-side spectrum. For this purpose the function $bins2spec()$ was used. It calculates the amplitudes of the complex numbers for the first half of the array and doubles it; normalizes them with respect to the number of elements and write the values into the predefined array.

4.3.1 Searching for n maximum amplitudes of the spectrum

The algorithm of searching n maximum amplitudes of the spectrum is implemented in a $getMaxFreqAndAmpl()$ function in the main program for the IoT Edge Node (SERVER). It take a one sided spectrum float type array and write n maximum amplitudes into the predefined $maxAmpl[n]$ array and the corresponding frequencies into the predefined $maxFreq[n]$ array. At first the algorithm searches the highest amplitude and writes down both values, then it zeros k neighboring points from both sides and starts the search again until it finds all n elements. To zero k neighboring points around the found maximum is crucial, since the main lobe of the spectrum may be quite wide and the next search of the maximum will stumble into the neighboring point. The number of maximum amplitudes to search is defined as $MAXFREQ\_NUM$ parameter and the number of neighboring with maximum amplitude points to eliminate is defined as $NEIGHBOURS\_NUM$ parameter in the $main.h$ file in the IoT Edge Node's (SERVER's) program.

The method of searching these n maximum amplitudes is shown on the figure 4.7. When the algorithm finds maximum amplitude it remembers it's index. Then, the algorithm checks whether the elements we are going to set to zero are within the range of the array and, if no, sets all the elements that are within the range.

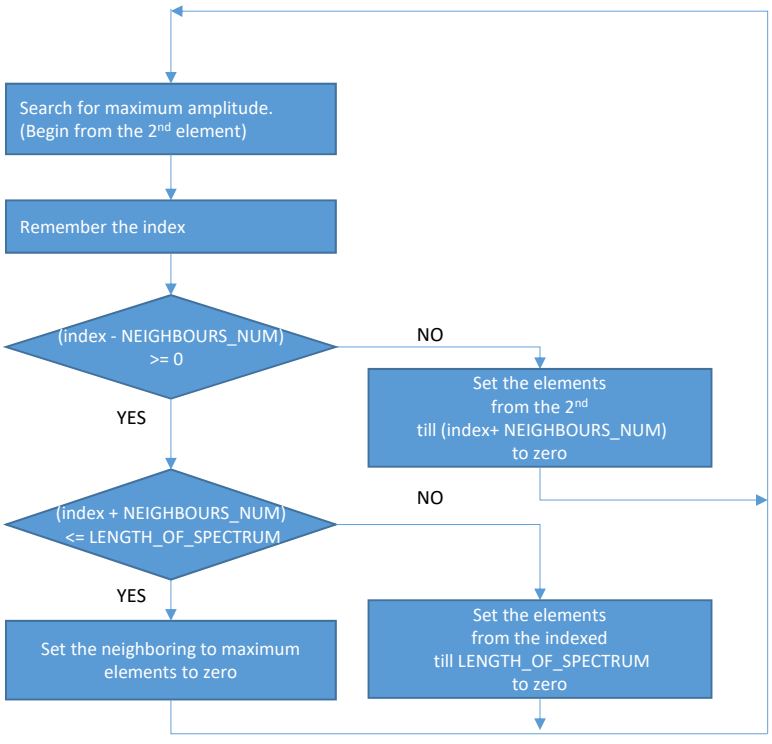


Figure 4.7: The method of searching a particular number of maximum amplitudes from a spectrum.

The results of the algorithm can be seen on the figure 4.8. The grey square dot shows the maximum found and the red stars are the three neighbors to be eliminated from both sides.

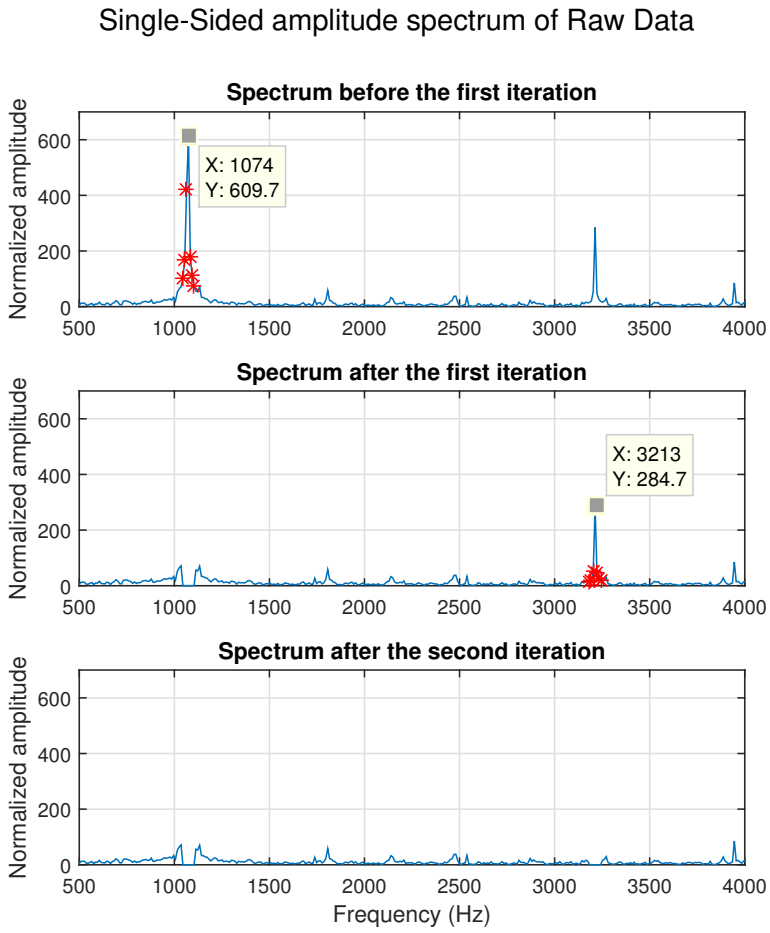


Figure 4.8: Spectrum evaluation after two iterations of the algorithm to search for maximum amplitudes.

4.4 Root Mean Square and Crest factor

Root mean square value (RMS) refers to the most common mathematical method of defining the effective component of an alternating wave [3] [24]. In our case it gives us a feeling of how severe the vibrations are, or at least it can provide us information about the change rate at long term monitoring, which will allow to predict when will a failure occur.

Since the values of the ADC converter get re-scaled according to its resolution and the accelerometer vibrates around its idle state we have to disregard the dc offset and treat the signal as if it had zero mean value, in other words fluctuates around zero. Thus, the RMS values for each axis have

been calculated according to the following formula:

$$RMS = \sqrt{\frac{1}{n}\sum_{i=0}^{n} k_i^2 - \left(\frac{1}{n}\sum_{i=0}^{n} k_i\right)^2}, \tag{4.1}$$

where k_i is the value of a signal at time i and n is the number of all points of a sampled signal.

Crest factor is a measure of a waveform, showing the ratio of peak values to the effective value. In other words, crest factor indicates how extreme the peaks are in a waveform [3] [25]. Since the RMS value has been modified and has been treated as a signal with zero DC offset, the maximum value must also be decreased by the mean value. Thus, the Crest Factor values for each axis are calculated according to the following formula:

$$Crest\ Factor = \frac{max(abs(k)) - \frac{1}{n}\sum_{i=0}^{n} k_i}{RMS} \tag{4.2}$$

where the operator $max()$ means maximum value, and $abs()$ means absolute value.

4.5 Other signal processing algorithm for the vibration monitoring

The classical signal processing algorithm have been discussed and implemented in the sections above. A deeper investigation into the vibration nature can be taken if using cepstrum analysis for example. The cepstrum can be seen as information about rate of change in the different spectrum bands. [26]. The power cepstrum of a signal is defined as the squared magnitude of the inverse Fourier transform of the logarithm of the squared magnitude of the Fourier transform of a signal. [27] .

Chapter 5

Modeling and PCB design

In this section the development of the auxiliary PCBs for the IoT Edge Node and the IoT Wi-Fi Gateway)the blocks marked light blue on the figure 5.1) is going to be described.

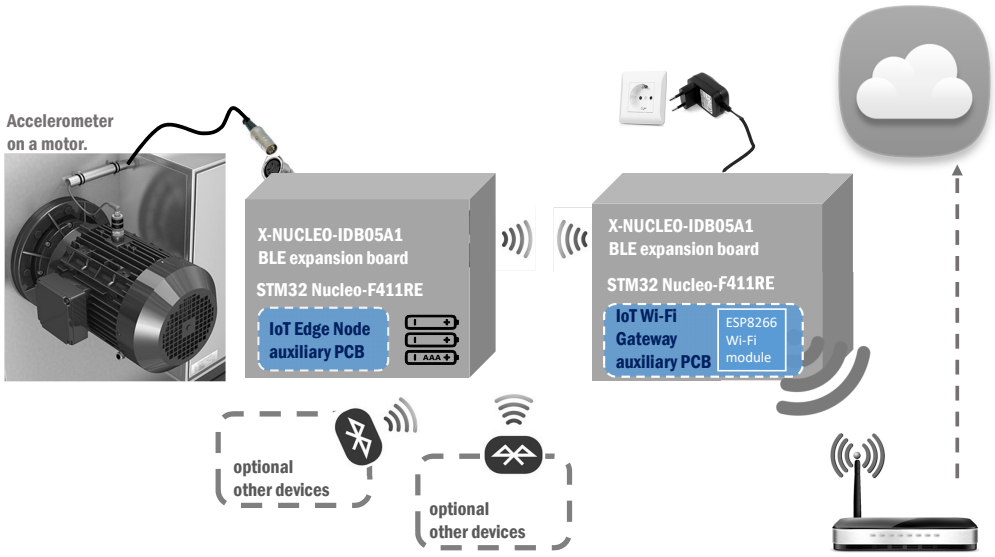
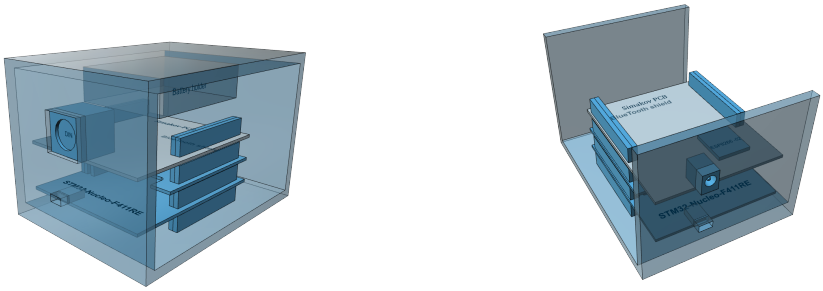


Figure 5.1: The developed components of the device.

5.1 3D models of the device and the breadboard prototype

For better visualization of the device 3D models of the IoT Edge Node and the IoT Wi-Fi Gateway were developed and are presented on the following figure 5.2.

Before creating a printed circuit board the breadboard prototype was



(a) : IoT Edge Node **(b) :** IoT Wi-Fi Gateway

Figure 5.2: 3D models of the IoT Edge Node and the IoT Wi-Fi Gateway boxes.

developed. ESP8266-01 WiFi module was used instead of esp8266-WROOM-02 at that time and a piezoelectric accelerometer ACH-01. The breadboard prototype is shown on the figure 5.3. This prototype is working without the IoT Wi-Fi Gateway (CLIENT) stage and sends processed data from the IoT Edge Node (SERVER) directly to the cloud. This was done so in order to learn how to work with ESP8266 and communicate with it via UART. The communication with the CLIENT via Bluetooth was tested simply with two Nucleo boards. The rest of the test work was conducted after the the PCBs were produced.

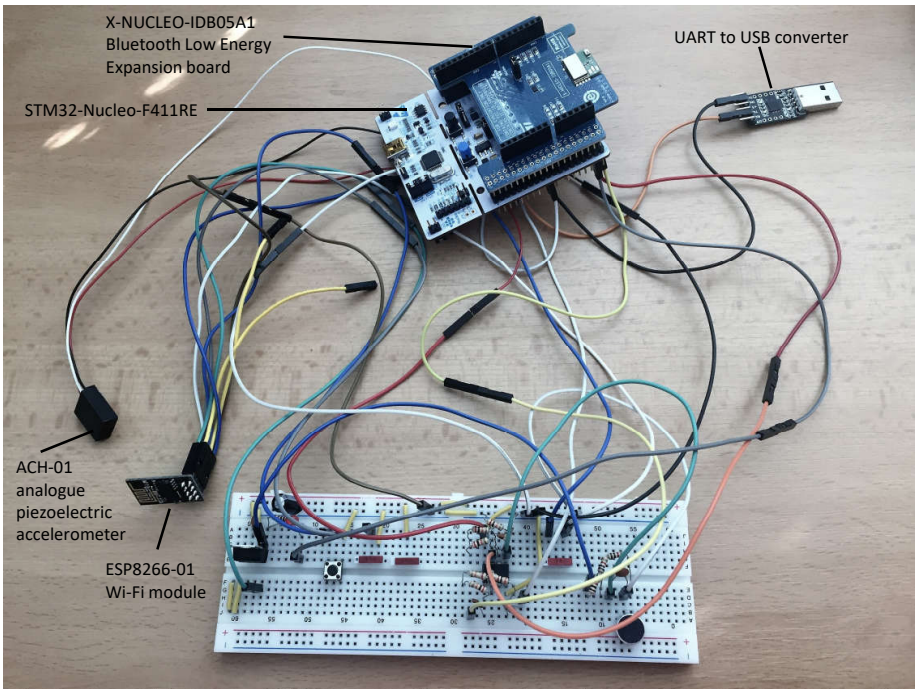


Figure 5.3: Breadboard prototype.

	Parameter	Test conditions	Min.	Typ.	Max.	Unit
V_O	Output voltage	$I_O = 500\ mA$	4.9	5	5.1	V
V_O	Output voltage	$I_O = 5\ mA$ to $1.5\ A$, $V_I = 6.5$ to $15\ V$	4.8	5	5.2	V
ΔV_O	Load regulation	$I_O = 5\ mA$ to $1.5\ A$		8	25	mV
V_I	Maximum input voltage	$I_O = 5\ mA$			17	V
V_d	Dropout voltage	$I_O = 0.5\ A$ $I_O = 1.5\ A$		200 500	400 900	mV mV
$V_{I(max)}$	Absolute MAX input voltage				30	V

Table 5.1: Electrical characteristics of the low drop voltage regulator L4940.

5.2 Electrical scheme

5.2.1 Power supply

It was decided to create two double-layers auxiliary PCBs for the IoT Edge Node and the IoT Wi-Fi Gateway. The IoT Edge Node board is powered with a set of 4x1.5V AA batteries giving 6V in total. There are two options how to power NucleoF411RE board either 5V or 3.3V. The second option entails the loss of debugging feature. Since our design is for testing purposes and we want to leave some flexibility and be able to debug, we power up both the IoT Edge Node and the IoT Wi-Fi Gateway boards with 5V. To reduce the voltage from 6V down to stable 5V the L4940 LDO stabilizator from STMicroelectronis with capacitors are used. This stabilizator has a very low voltage drop, less than 0.4V at 500mA. The electrical characteristics of this stabilizator and electrical scheme of 5V power supply for the IoT Edge Node board are shown on the table 5.1 and the figure 5.4 respectively. Full specification is in the data sheet [15].

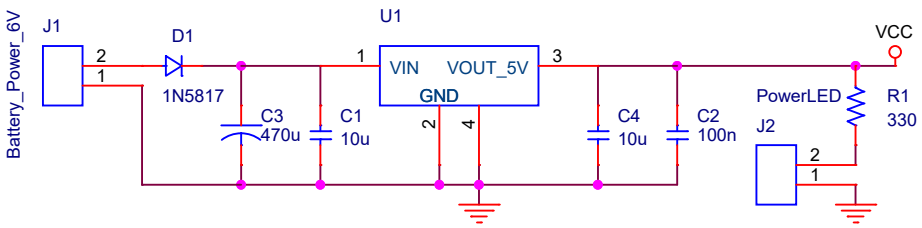


Figure 5.4: NucleoF411RE 5V power supply scheme for the IoT Edge Node.

The IoT Wi-Fi Gateway box is designed to work with regular 230V 50Hz plug. The AC/DC adapter brings up to 6V (constrained by TC1262 stabilizator) to the DC power jack on PCB. On the IoT Wi-Fi Gateway auxiliary PCB the power supply for the NucleoF411RE and ESP8266-WROOM-02 Wi-Fi module is done separately, since ESP8266 can not be powered with

	Parameter	Test conditions	Min.	Typ.	Max.	Unit
V_O	Output voltage	see in [16]	$3.3 - 2.5\%$	$3.3 \pm 0.5\%$	$3.3 + 2.5\%$	V
$\Delta V_{OUT}/V_{OUT}$	Load regulation	see in [16]		0.002	0.01	%/mA
V_I	Input operating voltage	see in [16]	2.7		6.0	V
$V_{IN} - V_{OUT}$	Dropout voltage	$I_L = 100\ \mu A$		20	30	mV
		$I_L = 100\ mA$		60	130	mV
		$I_L = 300\ mA$		200	390	mV
		$I_L = 500\ mA$		350	650	mV
$V_{IN(max)}$	Absolute MAX input voltage				**6.5**	V

Table 5.2: Electrical characteristics of the low drop voltage regulator TC1262.

a higher voltage than 3.3V and, on the other hand, we want to keep the debug feature on the NucleF411RE side. To ensure stable 3.3V, the TC1262 stabilizator was chosen. Its electrical characteristics and the electrical scheme of the power supply of the IoT Wi-Fi Gateway box are shown on the table 5.2 and the figure 5.5 respectively.

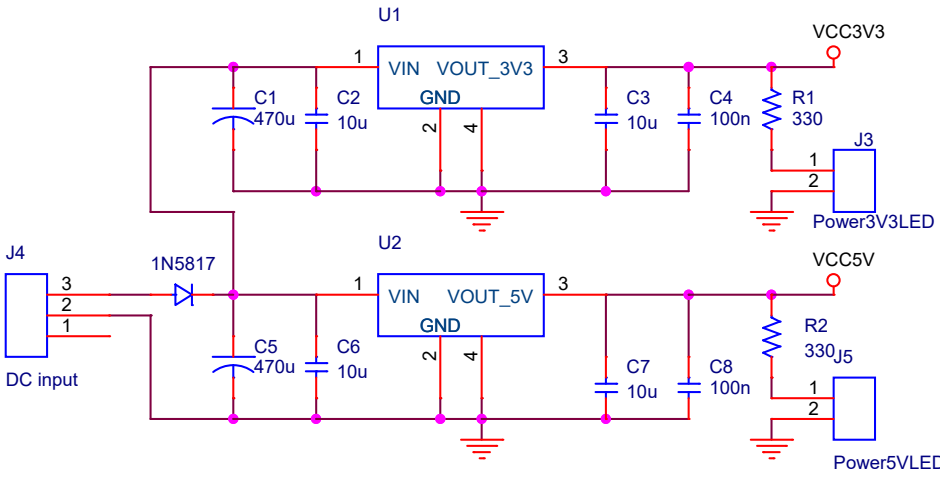


Figure 5.5: 5V power supply scheme for the NucleoF411RE and 3.3V for ESP8266-WROOM-02 Wi-Fi module.

5.2.2 Amplification of the signal from the accelerometer

The IoT Edge Node auxiliary board contains a standard DIN 5 pins connector, where the 3-axial accelerometer can be connected, and 2x2 operational amplifiers (one is not used and kept as a spare one) to amplify the signals from the accelerometer. The operational amplifiers are configured in their non-inverting format with gain = 2. But this gain can be easily changed by changing the smd resistor ($R4$, $R5$, $R8$, $R9$, $R12$, $R13$) for a signal from an accelerometer needs other gains, depending on the application requirements. The electrical scheme of the connection is on the figure 5.6.

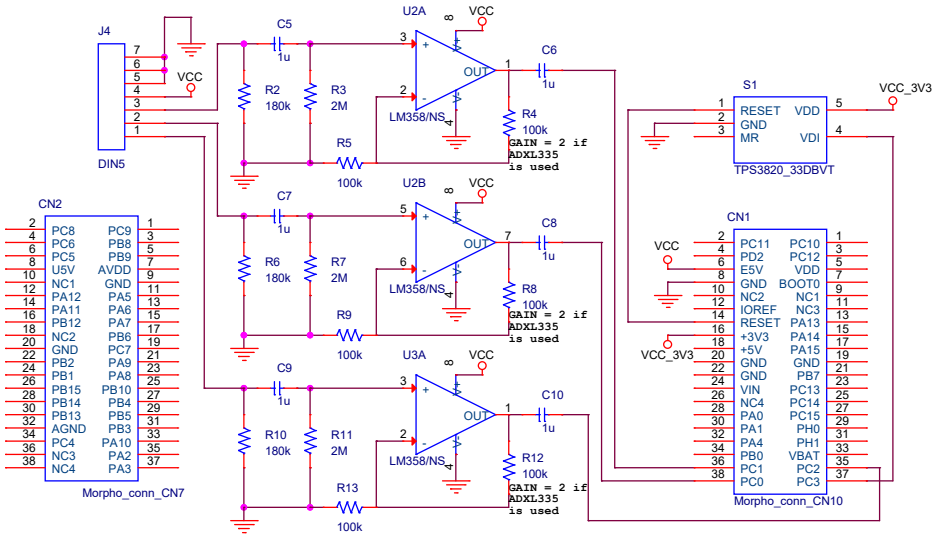


Figure 5.6: The electrical scheme of the 3-channel accelerometer connection and amplification.

5.2.3 Processor supervisory circuit (watchdog)

The IoT Edge Node auxiliary PCB is equipped with a hardware processor supervisory circuit from Texas Instruments TPS3820-33DBVT ($S1$ on the figure 5.6). It "watches" the microprocessor, so that when it gets stuck, the supervisory circuit resets it. As soon as WDI pin gets LOW, the internal timer of the watchdog starts counting 200ms ($t_{t(out)}$). This timer is reset by a falling edge on the WDI pin (to detect a falling edge, the HIGH pulse must have $100ns$ pulse width minimum). If the time is up, the output pin RESET gets LOW and resets the microprocessor. For this specific watchdog, the RESET pin remains LOW for 25ms (t_d). The timing diagram of this watchdog is shown on figure 5.7

5.2.4 ESP8266-WROOM-02 Wi-Fi module connection

The IoT Wi-Fi Gateway auxiliary PCB contains ESP8266-WROOM-02 SMD Wi-Fi module, switches to be able to flash firmware into it and the watchdog. The electrical scheme of the module and the watchdog connection is on the figure 5.8. The connectors $J1$ is meant to be paired with $J6$, and $J2$ with $J7$ by flexible flat cable. It was decided to leave these connections via cable to keep flexibility and be able to connect other GPIO pins from Nucleo to GPIO pins on ESP8266-WROOM-02.

The resulting schemes of the printed circuit boards are on the figure 5.9.

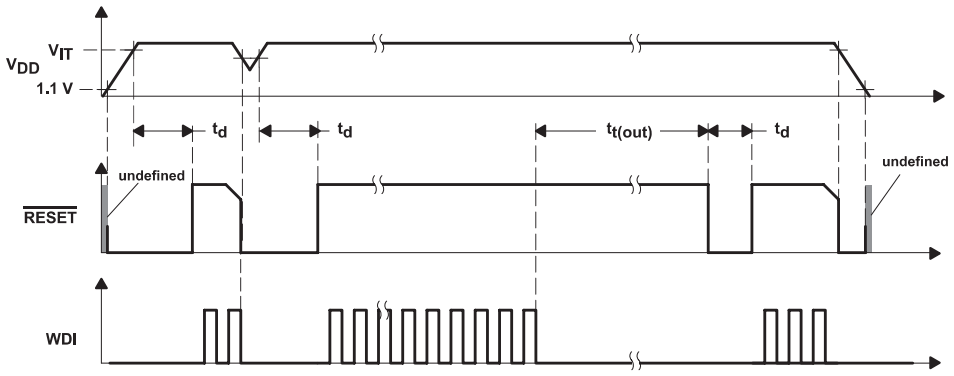


Figure 5.7: Timing diagram of the TPS3820 watchdog.

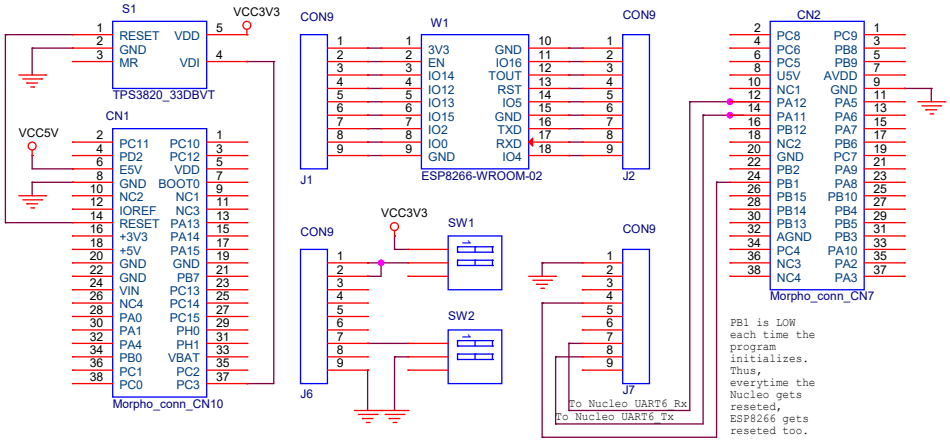
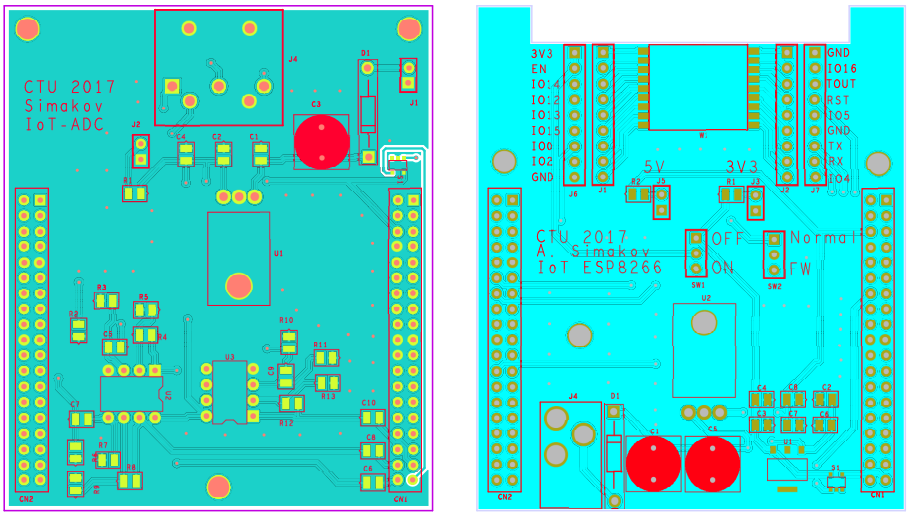


Figure 5.8: Timing diagram of the TPS3820 watchdog.



(a) : IoT Edge Node auxiliary PCB **(b) :** IoT Wi-Fi Gateway auxiliary PCB

Figure 5.9: Top view of the PCBs.

Chapter 6

Results of the device testing

6.1 Test equipment

The verifying tests have been conducted in the laboratory with the signal generator (figure 6.1) and the ”shaker” (source of vibrations). In the laboratory test the MEMS ADXL335 accelerometer was used. It was attached to the shaker to read the vibrations as it is shown on the figure 6.2.

Figure 6.1: Signal generator.

6.2 Raw data mode tests

To read the data from the IoT Edge Node in a RAW DATA MODE refer to *b*), mentioned in 2.1, a simple MATLab script can be used. The example of the script is shown on the next page.

In the first test the signal generator produced 237Hz sine wave type vibration signal on the shaker. In the second, the 237Hz square wave type vibration signal has been used. The results of the readings from the accelerometer are shown on the figure 6.3. The spectrum of the X-Axis is shown on the figure 6.4. We can clearly see that the frequency set on the signal generator

```
1
2  % Create port instance to receive data from
3  s = serial('COM3');
4  s.Terminator='';
5
6  % Define the amount of data to be received
7  % (depends on STM32 RAM)
8  samples = 2048*6;
9  samplingRate = 25e-5; % 250 us
10
11 % Set the baudrate and input buffer size
12 set(s,'BaudRate', 115200,'InputBufferSize', 2048*6);
13
14 % Open the port and write data into predefined array
15 fopen(s);
16 fprintf('Sampling will be for %.1f s \n',...
17     samples*samplingRate/6)
18 out = zeros(samples,1);
19 out(1:end,1) = fread(s,samples,'uint8');
20 fprintf('Transmission complete \n');
21
22 % Input data sorting according to Buffer built algorithm
23 u = 0;
24 for i=1:6:length(out)
25     u = u + 1;
26     rawData_x(u) = hex2dec(strcat(...
27         char(dec2hex(out(i  ))), char(dec2hex(out(i+1)))));
28     rawData_y(u) = hex2dec(strcat(...
29         char(dec2hex(out(i+2))), char(dec2hex(out(i+3)))));
30     rawData_z(u) = hex2dec(strcat(...
31         char(dec2hex(out(i+4))), char(dec2hex(out(i+5)))));
32 end
```

Figure 6.2: Shaker.

is correctly detected. The spectrum also allowed to make judgements about the type of the wave. We can clearly see the odd harmonics present on the spectrum for the square wave type signal. In the third test the signal generator produced a square wave with five times lower amplitude at 100Hz. It is still possible to determine that the wave is not pure sine wave but has odd harmonics. The plot is shown on the figure 6.5

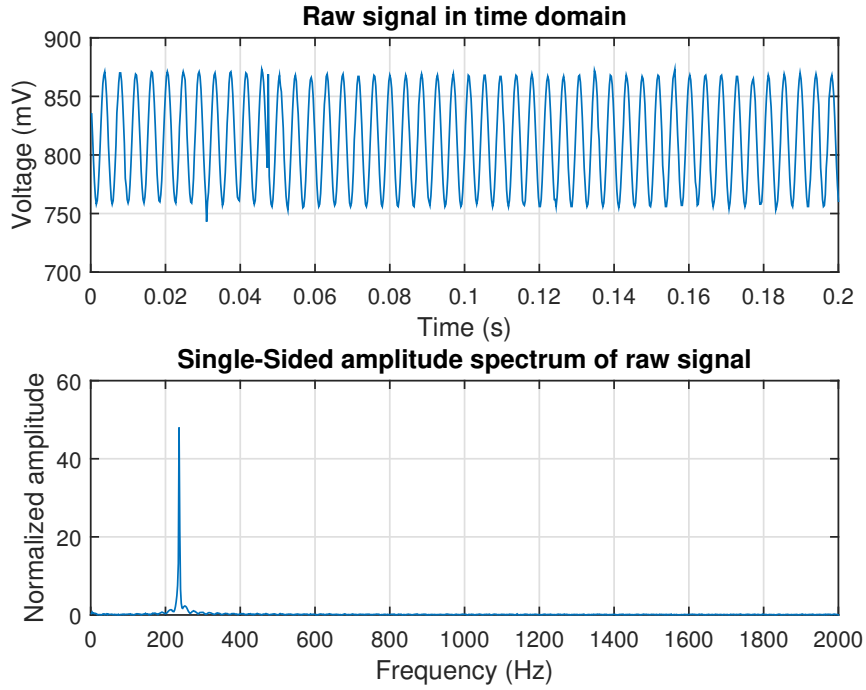


Figure 6.3: Plot of the 237Hz sine wave vibration signal on X-Axis channel.

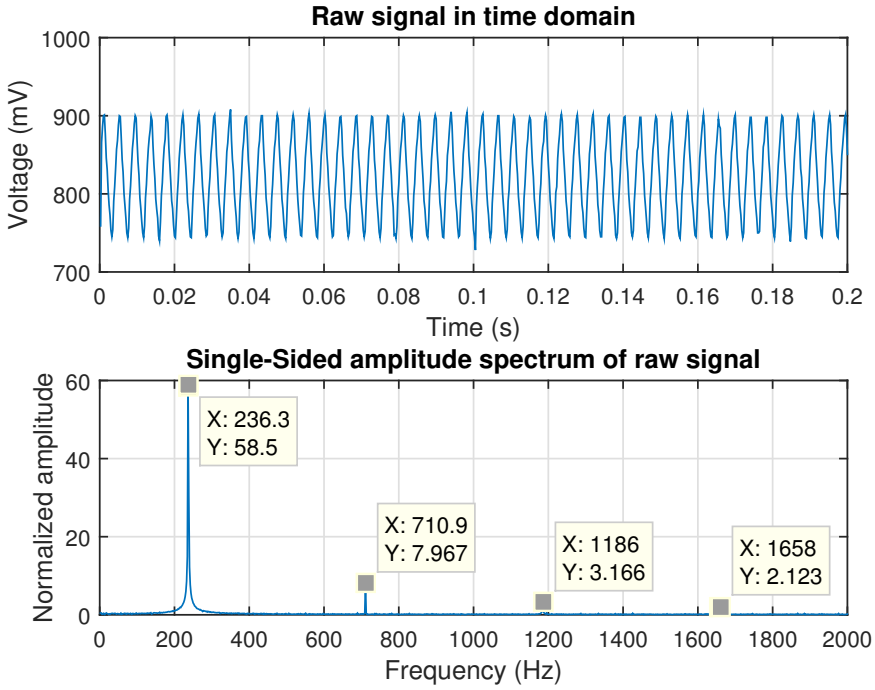


Figure 6.4: Spectrum of the 237Hz square wave vibration signal on X-Axis channel.

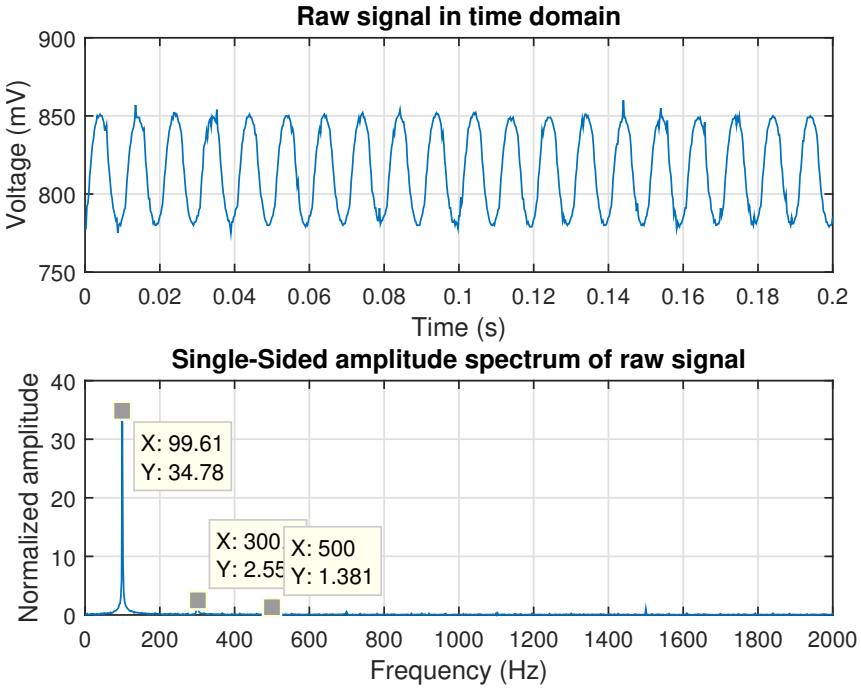


Figure 6.5: Spectrum of the 100Hz square wave vibration signal with 5 times lower vibration intensity on X-Axis channel.

6.3 Parameters of the MCUs.

The Nucleo board in the vibration Monitoring IoT Node ((Edge Node)) has predefined parameters for quick switching between 100MHz, 60MHz and 40MHz clock rate in the *cube_hal.h* file. FFT_BUF defines the size of the FFT buffer $SAMPLING\_FREQ$ defines the sampling frequency of the AD converter. $NEIGHBOURS\_NUM$ defines the number of neighbors to eliminate when searching for max amplitude of the spectrum. $MAXFREQ\_NUM$ defines the number of max amplitude points to write down and send to IoT Wi-Fi IoT Gateway.

Parameter	default value	Range
Clock rate	40MHz	(60MHz, 100MHz)
FFT_BUF	2048	powers of 2 (2048 is max for 3 axis)
$SAMPLING\_FREQ$	4kHz	up to 20kHz has been teested
$NEIGHBOURS\_NUM$	3	
$MAXFREQ\_NUM$	5	

Table 6.1: The parameters of the vibration Monitoring IoT Node (Edge Node).

Frequency, Hz	Shape	Signal Peak-to-Peak amplitude, mV	Number of points
350	Sine	100	10
350	Square	100	8
570	Sine	100	6
570	Square	100	7
717	Sine	100	7

Table 6.2: Operational modes of the vibration source for the tests with ThingSpeak.com web server.

6.4 ThingSpeak.com regular mode tests

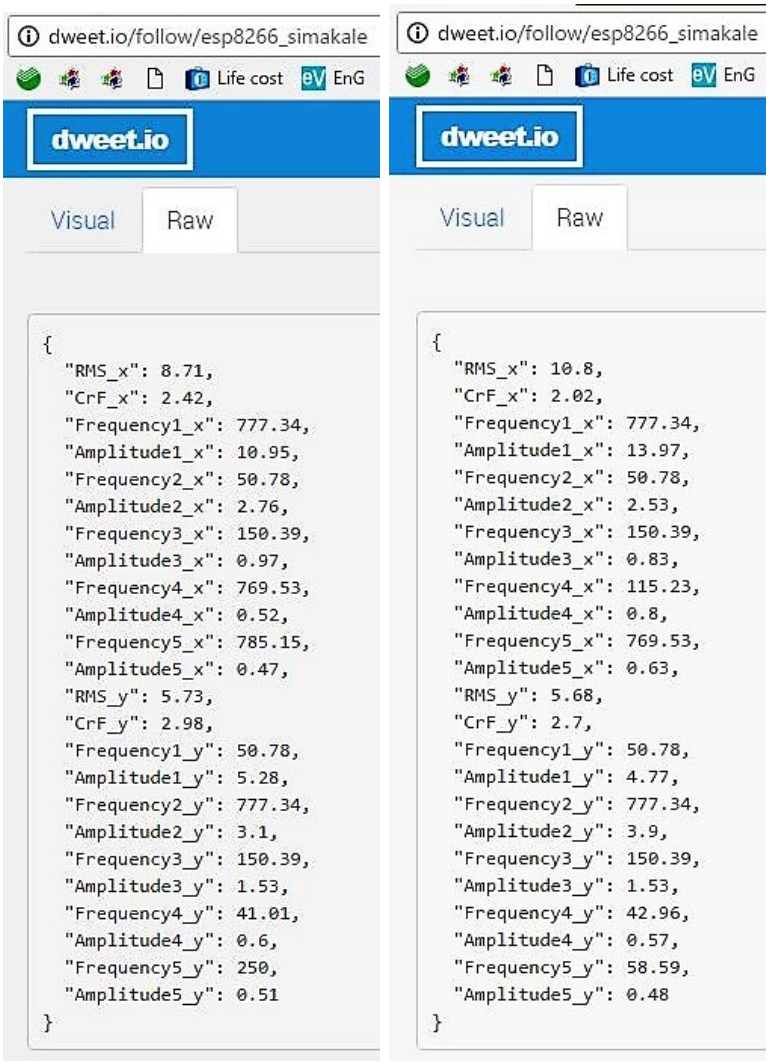
To test the regular mode with ThingSpeak.com web server the shaker produces the vibrations according to the 6.2 table. The results of data publicly available on `https://thingspeak.com/channels/271852` are shown on the figure 6.6. The device perfectly detects the frequency of the vibrations. It also detects the change in the energy of the vibrations, when the signal shape changes from sine into square. This is evident from the amplitudes and RMS values, since the RMS value of a square wave is higher than of a sine wave.

On the plots from ThingSpeak.com it can also be noticed that the transition from 570Hz to 717Hz has a wider time gap than between the other points. This happened due to receiving not all packets from the IoT Edge Node, thus the message had been ignored and only the next, fully received one, was used to send to the web server. This web server supports maximum 8 variable per channel, thus only 3 highest amplitudes with their frequencies have been sent there along with RMS and Crest Factor values. For better visualization, the figure 6.6 contains only 4 out of 8 variables.

Figure 6.6: View of the data posted on the ThingSpeak.com web server.

6.5 Dweet.io regular mode tests

To conduct the tests with the Dweet.io web server the corresponding firmware has been flashed to the ESP8266 Wi-Fi module. The equipment to produce vibrations remained the same. For the results shown on the figure 6.7 the signal of 777Hz was used with 70mV Peak-to-Peak signal amplitude. The results on the left are for the sine wave, on the right - for the square wave. If compare with ThingSpeak.com, here we can upload much more variables, as a result all 96 bytes received from the IoT Wi-Fi Gateway have been uploaded to dweet.io.



(a) : 777Hz sine wave **(b) :** 777Hz square wave

Figure 6.7: Dweet.io JSON string POST results.

Chapter 7

Conclusions

In this thesis, the IoT vibration monitoring device has been built. It consists of two boxes: the IoT Edge Node, which measures the vibrations and processes the data and the IoT Wi-Fi gateway, which sends the data, received from the IoT Edge Node, to the web services. (the boxes are shown on the figures 7.1, 7.2, 7.4 and 7.5). The vibration monitoring IoT Node has an input connector to connect an accelerometer. The performance of the piezoelectric accelerometer ACH-01 has been tested during the prototyping stage, the performance of the analog MEMS accelerometer ADXL335 has been tested during the laboratory experiments. The vibration Monitoring IoT Node (Edge Node) is designed in the way so that it digitizes the analog signal from the accelerometer and calculates FFT, RMS and Crest Factor values for each axis. The vibration Monitoring IoT Node is powered by 4 AA batteries and communicates with the IoT Wi-Fi IoT Gateway via Bluetooth. The IoT Wi-Fi IoT Gateway is powered from a regular 230V plug via an AC/DC adapter and is capable of connecting to WAN. Two firmwares for two web servers (ThingSpeak.com and Dweet.io) have been developed. The IoT Wi-Fi IoT Gateway can send data to ThingSpeak.com web server as well as to Dweet.io. The proposed system enables different kinds of research and development of MCM[1] specific signal processing algorithms.

7.0.1 Future work proposals

The future work can be conducted in the sphere of decreasing energy consumption to increase the lifetime of the batteries attached to the IoT Edge Node. This can be done by sending it into sleep mode and wake it up as soon as there is a necessity. Thus, the IoT Edge Node can be awake for $\backsim 2s$ and then sleep, for example, for 1 hour.

The enhancements of the upcoming Bluetooth® 5 focus on increasing functionality for the Internet of Things (IoT)[11]. With up to 4x the range, 2x the speed and 8x the broadcasting message capacity this will become even more useful. It is available to manufacturers since early 2017 and may be soon there will appear new expansion boards with Bluetooth 5 on it, which will allow developing devices with bigger range and speeds, keeping the power

[1] Machine Condition Monitoring

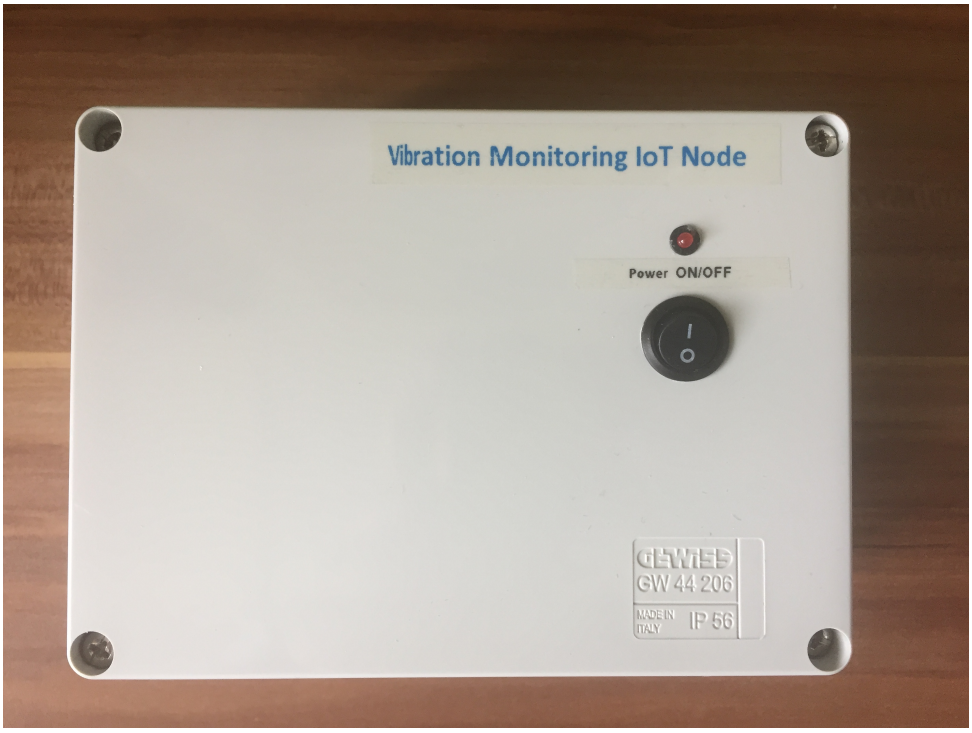


Figure 7.1: IoT Edge Node front view.

Figure 7.2: IoT Edge Node top view.

consumption at the level of previous generation.

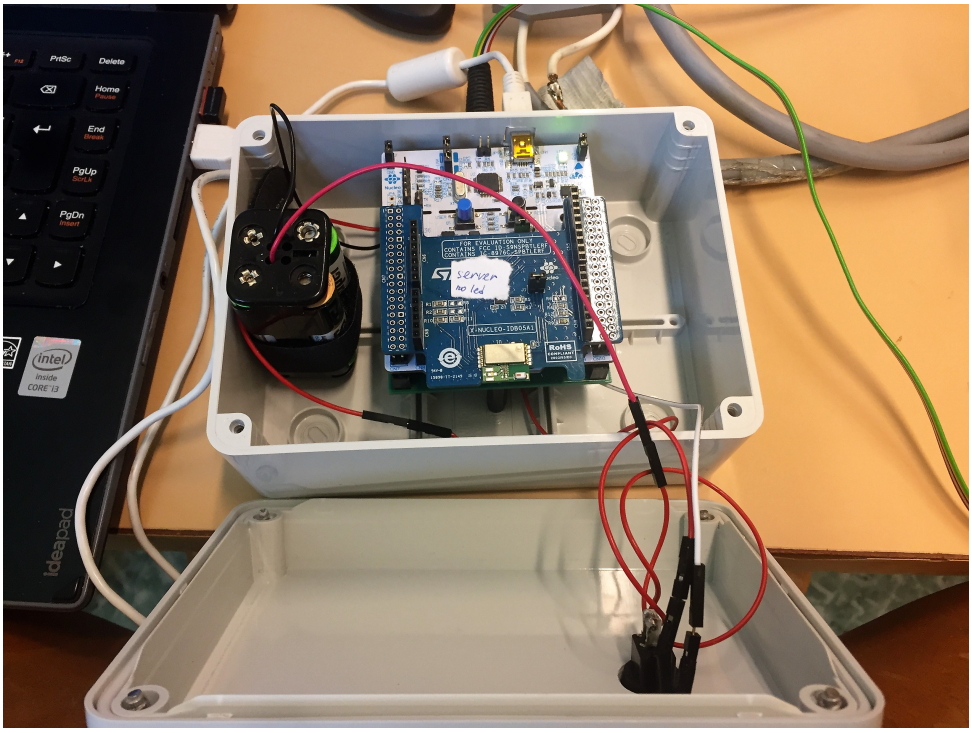


Figure 7.3: IoT Edge Node inside view.

Figure 7.4: IoT Gateway front view.

Figure 7.5: IoT Gateway top view.

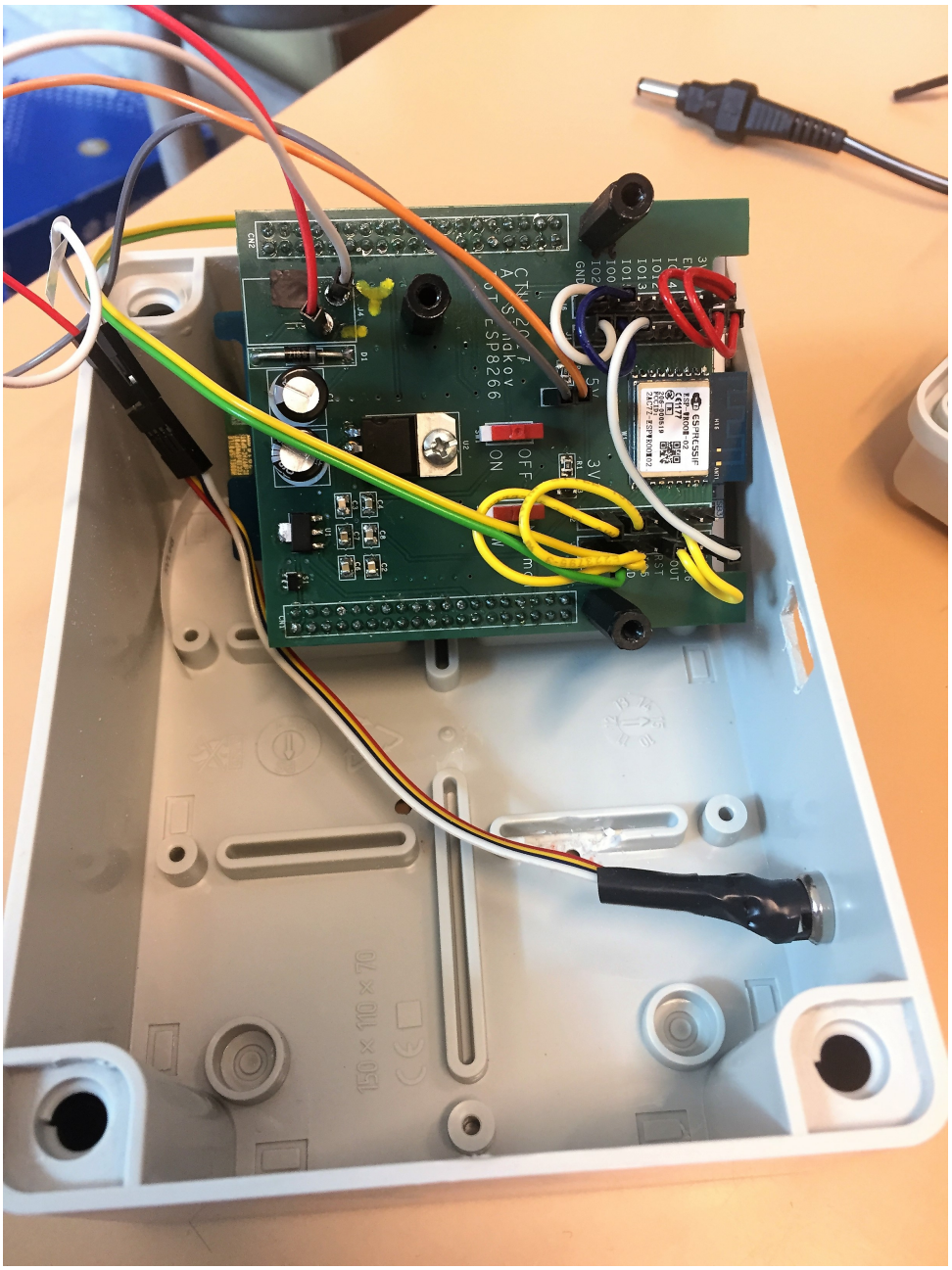


Figure 7.6: IoT Gateway inside view (view from the bottom).

Appendix A

Abbreviations

Abbreviation	Meaning
IoT	Internet of Things
IIoT	Industrial Internet of Things
BLE	Bluetooth low energy
GPIO	General purpose input output
MEMS	Micro Electro-Mechanical Systems
FFT	Fast Fourier Transform
GAP	Generic Access Profile
SDK	Software Development Kit
MCU	MicroController Unit
ADC	Analog to Digital Converter
PSU	Power Supply Unit
DMA	Direct Memory Access
DIY	Do It Yourself
JSON	JavaScript Object Notation
MQTT	Message Queue Telemetry Transport
WAN	Wide Area Network

Appendix B

Bibliography

[1] Marcel Kreidl and Radislav Šmíd *Technická diagnostika* BEN - technická literatura, Praha 2006.

[2] Robert Bond Randall. *Vibration-based condition monitoring : industrial, aerospace and automotive applications* 2011. ISBN 978-0-470-74785-8

[3] Rolf Iserman *Fault-Diagnosis Systems. An Introduction from Fault Detection to Fault Tolerance.* Springer Berlin Heidelberg New York. ISBN-10 3-540-24112-4

[4] `http://reliabilityweb.com/articles/entry/how_is_vibration_measured` which was created based on *Beginner's Guide to Machine Vibration*, copyright © Commtest 1999, 2006

[5] `http://embedded-computing.com/articles/innovation-in-sensor-analytics-at-the-edge/#`

[6] `www.statista.com`

[7] `https://assets.cdn.sap.com/sapcom/docs/2017/03/068181cb-ae7c-0010-82c7-eda71af511fa.pdf`

[8] `https://www.postscapes.com/internet-of-things-platforms/`

[9] `http://ifm-datalink.com/products/gb/ds/VKV021.htm`.

[10] `https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/low-energy`.

[11] `https://www.bluetooth.com/specifications/adopted-specifications`.

[12] Overview of the BLE Profiles application for X-CUBE-BLE1, expansion for STM32Cube. AN4642 Application note. DocID027341 Rev 3.

[13] Microchip developer HELP. `http://microchipdeveloper.com/wireless:ble-gap-roles`

[14] http://www.st.com/content/st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-expansion-software/x-cube-ble1.html

[15] L4940 1.5 A very low drop voltage regulator IC. DocID2141 Rev 14.

[16] TC1262 500 mA Fixed Output CMOS LDO. DS21373C.

[17] ADXL335: Small, Low Power, 3-Axis ±3 g Accelerometer Data Sheet (Rev. B)

[18] *ACH*_01 Technical specification.

[19] https://github.com/pfalcon/esp-open-sdk

[20] https://github.com/esp8266/source-code-examples/tree/master/dweet

[21] Espressif. ESP8266 Non-OS SDK API Reference.

[22] Espressif. ESP8266 SDK Getting Started Guide.

[23] http://www.math.wustl.edu/~victor/mfmm/fourier/fft.c

[24] https://en.wikipedia.org/wiki/Root_mean_square

[25] https://en.wikipedia.org/wiki/Crest_factor

[26] https://en.wikipedia.org/wiki/Cepstrum

[27] Norton, Michael Peter; Karczub, Denis. *Fundamentals of Noise and Vibration Analysis for Engineers.* Cambridge University Press. November 17, 2003. ISBN 0-521-49913-5