Expertise
and insight
for the future

Taneli Liljasto

Power Management in ARM Cortex M0+ with Real Time Operating System

Metropolia University of Applied Sciences

Bachelor of Engineering

Information and Communication Technology

Bachelor's Thesis

30 November 2019

Author Title Number of Pages Date	Taneli Liljasto Power Management in ARM Cortex M0+ with Real Time Operating System 46 pages 30 November 2019
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Smart Systems
Instructors	Keijo Länsikunnas, Senior Lecturer

The main goal of the thesis was to research an existing battery powered device and modify the existing software to include battery and power management related features based on the research. The main purpose of the battery related features was to be able to determine and indicate the battery charge level. The purpose for power management related features was to diagnose the power consumption of the device and investigate if it could be impacted with modifying the software.

The thesis explains the basic characteristics of lithium-ion batteries as a power source and explains how to determine the charge level of a battery as to the state of charge and how to implement this feature afterwards. Also, the voltage levels of the system were investigated and based on the discoveries a safety feature was designed and implemented. The purpose of the safety feature was to safeguard using the device on low voltages. Power saving was researched and implemented on an existing real time operating system (RTOS) using its features and the low power capabilities of the component.

The goals were reached, the device had capabilities to read the battery voltage which could be used to roughly determine the state of charge. Optimizing the existing software had an impact on power usage, with research the power usage was decreased further by taking advantage of the features of the RTOS and the low power features of the component.

Keywords	Microcontroller, RTOS, Battery charge

Tekijä Otsikko Sivumäärä Aika	Taneli Liljasto Virranhallinta ARM Cortex M0+ -mikrokontrollerilla ja reaaliaikaisella käyttöjärjestelmällä. 46 sivua 30.11.2019
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Älykkäät järjestelmät
Ohjaajat	Lehtori Keijo Länsikunnas

Insinöörityön tarkoituksena oli tutkia olemassa olevaa, akulla toimivaa laitetta ja ohjelmoida tutkimuksen pohjalta olemassa olevaan ohjelmistoon akkuvaraukseen ja virranhallintaan liittyviä ominaisuuksia. Akkuvaraukseen liittyvien ominaisuuksien päätarkoitus oli luoda varaustilanteesta kertovat indikaattorit, joiden avulla käyttäjä osaa arvioida latauksen tarpeellisuuden. Virranhallintaan liittyvissä ominaisuuksissa tarkoitus oli kartoittaa laitteen virrankulutus ja tutkia, mikäli ohjelmistoa muokkaamalla voisi olla vaikutusta virrankulutukseen.

Insinöörityössä perehdyttiin yleisesti litium-ioni-akun toimintaperiaatteeseen ja akun ominaisuuksiin virtalähteenä. Lisäksi perehdyttiin erilaisiin tapoihin mitata ja määrittää akkuvaraus sekä miten sellainen olisi mahdollista toteuttaa laitteelle jälkikäteen. Samalla kartoitettiin laitteen toiminnan kannalta oleelliset jännitetasot sekä suunniteltiin kartoituksen perusteella turvamekanismeja, joiden rooli oli rajoittaa laitteen käyttöä liian alhaisilla jännitteillä. Virransäästö toteutettiin olemassa olevan reaaliaikaisen käyttöjärjestelmän päälle selvittämällä sen ja komponenttien mahdollisuudet vähäiseen virrankulutukseen. Lopulta ohjelmistoon tehtiin muutokset, jossa hyödynnettiin reaaliaikaisen käyttöjärjestelmän ja komponenttien sisäänrakennettuja ominaisuuksia osana virransäästöä.

Insinöörityön tavoitteessa onnistuttiin; laitteen suunnittelussa oli alun perin otettu huomioon akkuvarauksen lukuun liittyviä toiminallisuuksia, joita hyödyntämällä akkuvarauksesta oli mahdollista tehdä karkea arvio. Samalla havaittiin, että laitetta oli mahdollista käyttää liian alhaisilla jännitteillä, jolloin laitteen toiminnassa saattoi ilmetä toimintavarmuuteen liittyviä ongelmia. Toimintavarmuuden varmistamiseksi luotiin ominaisuus, joka tarkkaili akkuvarausta sekä esti laitteen käytön liian alhaisella jännitteellä. Virransäästössä olemassa olevan ohjelmiston optimoiminen toi merkittäviä säästöjä virrankulutukseen. Lisäksi selvitystyön pohjalta oli mahdollista toteuttaa lisäominaisuuksia, joiden pohjalta laitteen komponenttien virtaominaisuuksia pystyttiin kontrolloimaan ohjelmallisesti ja näin saavuttamaan vielä suurempia säästöjä virrankulutuksessa.

Avainsanat	Mikrokontrolleri, RTOS, Akkuvaraus

Contents

List of Abbreviations

1 Introduction 1

2 State-of-Charge 2

2.1 Basics of Battery 2
2.2 Battery Characteristics 3
2.3 History of Li-ion Battery 5
2.4 Characteristics of Li-ion Battery 6
2.5 Battery Management System (BMS) 8
2.6 Charging Li-ion Battery 10
2.7 Overview of Voltage Characteristics of System 12
2.8 Reading Voltage Levels (A/D Conversion) 13

3 Implementing State of Charge 17

3.1 Implementing ADC 17
3.2 Interpreting ADC Readings 19
3.3 Determining State-of-Charge 21
3.4 Implementing Power Manager with Safety Features 22

4 Power Saving 25

4.1 Microcontroller 27
4.2 IMU Sensor 30
4.3 Radio Module 31
4.4 Display 32
4.5 Real Time Operating System (RTOS) 33
4.6 Advanced Low Power Mode 38
4.7 Measuring and Analyzing Low Power Modes 40

5 Discussion and Conclusions 43

References 45

List of Abbreviations

A/D	Analog to Digital
ADC	Analog to Digital Converter
A	Ampere, measurement of current
CC	Constant-current, Li-Ion battery charging mode until certain voltage is reached
CV	Constant-voltage, Li-Ion battery charging mode to finish the charging
DSP	Digital signal processing, signal manipulation and processing
Interrupt	Signal which is used to trigger events in microproseccors.
Kernel	Operating system's base which makes use of all of the processor's capabilities
LA	Lead-acid type battery, invented by Planté
Li-Ion	Lithium ion type battery
Li-NMC	Lithium ion type battery with nickel-manganese-cobalt based chemistry
NiFe	Nickel-iron based battery invented by Jungner
RTOS	Real time Operating system
SAR	Successive-approximation-register based ADC
SoC	System-of-Charge, power level of the battery in percentage
SoC	System on a Chip, microcontroller solution packed on a chip, includes processor core and peripherals.

SOR	State-of-charge indicates system’s charge level
Power	Usually expressed in Watts and product from voltage and current.
V	Volt, measurement of energy level
W	Watt, power rate from product of volt (V) and current (A).
Wh	Watt-hour, power(W) consumption per hour

1 Introduction

The purpose of the thesis was to research power management in embedded systems and implement one on an existing device. The theoretical part of the project is separated into two parts, in the first part the main topic is getting to know and indicate the power level (state-of-charge) of the device and the second part discusses power consumption and how to save it.

The first part covers what a power level is and how to determine one, the theory part is mainly focused on topics related to the already existing design and implementation of the device but for greater knowledge and understanding of power levels, the theory takes in consideration some alternative designs and architectures as well makes small comparisons on their key features.

The body of the device consists of electronics circuitry which has many components, each having their own function and purpose, the heart of the device is a microcontroller (CPU) which main function is to make use of all other components connected to it. Main components in addition to the CPU are an LED display to communicate with the user, a button to control the device, a motion sensor to sense real-world movements, a radio to communicate with other electronic devices and a Li-ion based battery to allow portable usage of the device.

When the first part relies heavily on the existing circuitry designed and implemented by another party, the second part is solely programmatical where the device can be programmed to follow pre-determined actions. The goal of the second part is to research the capabilities of an Arm Cortex M0+ based microcontroller and a real time operating system on power saving and make use of these features by implementing them as part of the whole power management created in the first part of the project.

2 State-of-Charge

Ever since invention of storing energy there has been some solutions to indicate a power level. Determining a power level is not very straightforward since there are many variables that has impact on how the power level may be obtained and large variation in precision [1,11]. In general a state of charge indicates power level represented as a percentage value and this chapter covers theoretical part on how to determine a state of charge.

2.1 Basics of Battery

In the 1800's Alessandro Volta made a discovery where an electrochemical reaction between two metal plates generated continuous electrical force. This led to the invention of a battery. The first electrochemical battery, which is also known as Voltaic pile, was built from copper and zinc plates which were stacked on top of each other and separated by a cloth which was soaked in salty water [1,11].

A modern battery has evolved a lot from the first battery but the working principle has remained the same as visualized in Figure 1, in plain a battery consists of two metal plates which creates poles, positive side (cathode) and a negative side (anode), between cathode and anode there is an electrolyte which works as an promoting catalyst, making the electricity flow between poles when the battery is charging or discharging [12]. Compared to the first battery the copper plates worked as a cathode, zinc as an anode and the soaked cloth was the electrolyte [2,3].

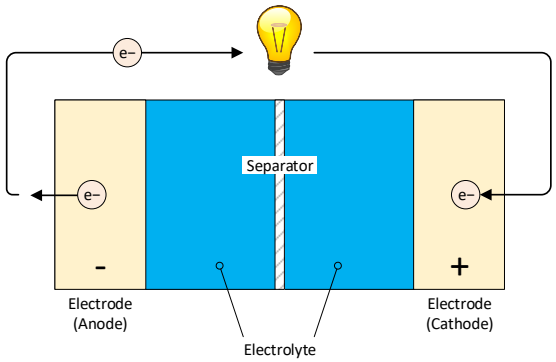

Figure 1. Working principle of basic battery under discharge [2,3].

The function of a separator (shown in Figure 1) is to provide a barrier between the anode and the cathode to avoid direct contact between anode and cathode which would cause a short circuit. The separator not only separates, it also has a feature which enables exchange of lithium ions from one side to another, when discharging lithium ions transfer from anode to cathode and during recharge vice versa. A separator does not isolate sides completely and some small current may pass which is known as self-discharge, eventually long storage will deplete the charge of the battery [14].

The properties of a battery are determined by the electrochemistry of materials used, ever since the first battery scientists have made many advancements with mixing different materials in search of best battery, a battery that could store large amount of electricity (capacity), have high electrical pressure (voltage) and high flow of electric charge (current) alongside of compact size.

Some achievements worth mentioning in battery evolution include:

- 1800, first battery invented by Alessandro Volta [1,11].
- 1859, first re-chargeable battery invented by Gaston Planté, the battery was based on lead-acid (LA) chemistry [1,13].
- 1899, invention of the nickel-cadmium (NiCd) and nickel-iron (NiFe) based batteries by Waldemar Jungner [1,13].
- 1990, first commercial introduction of the nickel-metal hydride (NiMH) battery from Sanyo Electric Co. [1,13].
- 1991, first commercial introduction of the lithium-ion (Li-ion) battery from Sony Corporation [1,13].

2.2 Battery Characteristics

A battery has many characteristics that manufacturers have measured and documented under technical documentation, also known as datasheets. With batteries there are many aspects that needs to be taken in consideration when choosing which battery type is the right one for the application. The primary goal for the designer is to have most optimal solution for the application, but in many occasions some compromises may have to be

made. Figure 2 tries to visualize the design goal vs. actual solution with taking in consideration the large variety of different aspects involving the decision making about a battery resulting into compromises [13].

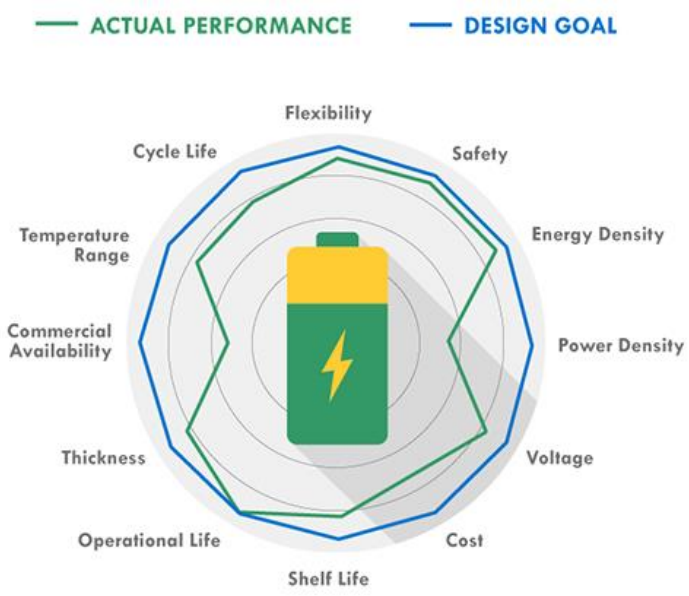

Figure 2. Battery design compromises, goal vs. reality [13].

Every time a battery is recharged a physical stress is created causing the material to age which cannot be reversed, modern consumer batteries are built as maintenance free solutions where cells are closed, it means that the battery manufacturer has ensured the cell to contain enough materials to achieve cycle life mentioned in the technical documentation, in other words the cycle life indicates how many times a cell can be discharged and recharged before aging causes the cell's maximum capacity to decrease to level of 80% from the maximum capacity [1,111].

When a battery is discharged and charged the process creates heat which expedites aging and thus the pressure inside the cell results to increase, if the cell is charged too rapidly it may create enough heat and pressure to cause the cell to expand and cause short circuit which may result into explosion or fire. Pressure changes has been taken in consideration by manufacturers with the design of the cell's structure, li-ion batteries have built-in safety valve to release too high pressures to minimize risks [11].

Charge and discharge characteristics are represented as C-rate where the C stands for standard unit of electric charge (Coulomb) and equals to current represented as Amperes per an hour (Ah) also known as capacity. For example, a battery with 2.4Ah (2400mAh) capacity is represented as 1C. C-rate is a way to represent larger charge and discharge rates with multiplier and divider as with previous example a 2C would be two times the capacity (4.8Ah) and C/2 would be maximum capacity divided by two (1.2Ah) [1,24]. Cell specific C-rates can be found from the datasheet which explains performance on different discharge and charging loads.

2.3 History of Li-ion Battery

First studies about lithium as a base of a battery date from 1913 by Gilbert N. Lewis, although the research about lithium really started to take steps in the 1960s and 1970s [2,4]. The first non-rechargeable Li-ion batteries came into market in 1970 [1,12]. The creation of a rechargeable li-ion battery took great steps in 1973 when M. Stanley Whittingham and Fred Gamble made discoveries where they detected promising actions with lithium on quantum mechanical level that finally led to first working rechargeable battery which was demonstrated in 1976 [2,13]. The battery was not stable enough and failed due to safety problems [1,12].

Research on li-ion batteries continued and Whittingham's discoveries gave ideas on how to proceed with researching li-ion batteries. The next great progress came in late 1970s when John B. Goodenough and his co-workers at Oxford university made new discoveries on quantum mechanical level to find more suitable components with better electrochemistry. With these findings Goodenough was able to increase cell's voltage from 2.5V to almost 4-5V, it was a significant discovery and had a great impact on battery development [2,8].

In 1985 a group led by Akira Yoshino made a breakthrough which led to cells that had higher charging voltage, higher energy density, more recharging times and were safer than previous versions. In 1991 Sony corporation introduced first rechargeable battery, the li-ion battery has evolved ever since becoming a power source used by millions of people every day and the industry is estimated to even grow rapidly in the future because of the automobile industry's contribution on moving from fossil fuels to electricity [3].

The li-ion battery is considered to have great impact on the world, from revolutionizing mobile industry to transitioning from fossil fuels to electrically powered transportation. John B. Goodenough, M. Stanley Whittingham and Akira Yoshino were awarded with Nobel Prize in Chemistry 2019. [2,1]

2.4 Characteristics of Li-ion Battery

Compared to other rechargeable batteries the li-ion based batteries have succeeded in becoming very popular, one of the reasons for this is higher output voltage which means higher power solutions can be created with less cells compared to other popular battery types. The reported output voltage for a li-ion battery is typically 3.6V which is known as nominal voltage, however the measured output voltage varies from 3.0V to 4.2V, the output voltage varies on how much capacity is left in the battery [1,27].

The cell is equipped with a protection circuit of which the function is to protect the cell from extreme currents and voltages. The cell has a built-in safety valve as shown in Figure 3 which can release extra gasses generated due mechanical stress of charge or discharge. After the gasses are released the valve closes and the cell may remain operational. However, in case of rapid extreme pressure, for example caused by a short circuit the valve mechanism will self-destruct resulting in the cell being completely disabled in order to avoid fire [18].

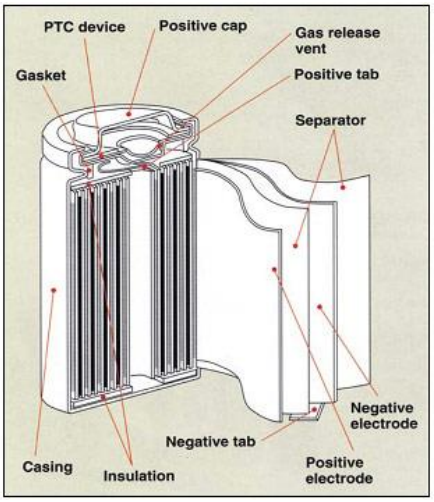

Figure 3. Cross section of an 18650 type Li-NMC cylindrical battery [11].

A typical Li-ion cell comes in a cylindrical shape and the electrochemistry of the cathode is based on Lithium-Nickel-Manganese-Cobalt-Oxide (LiNiMnCoO2), also known as Li-NMC type battery. With good characteristics the Li-NMC has been gaining lots of popularity, for years it has been widely used in laptops and within the past few years it has been gaining more and more popularity by an electric car manufacturer Tesla as they have been using Li-NMC cells as the base of their car batteries resulting the Li-NMC to be the most commonly used li-ion battery in 2018 [3].

The cathode of the Li-NMC battery is typically 1-1-1 where one part is nickel, one part manganese and one part cobalt. It is possible to successfully create a Li-NMC cell with the combination of 5-3-2 but because of the high market price of cobalt the manufacturers try to use as little cobalt as possible and are moving away from cobalt based cathodes towards nickel based systems. The only drawback compared to cobalt based systems is a slightly lower voltage. [4]

The electrochemistry of an NMC type battery allows the cell to have high capacity and continuous high power output. It excels in every aspect compared to a popular competitor Ni-MH, it has a lower self-discharge rate, higher life cycle, higher nominal voltage and is maintenance free. A comparison of the main characteristics can be viewed in Table 1 [16]. Cadmium is a very toxic metal which has resulted to the Ni-Cd batteries to be banned in consumer electronics within the European Union region, there are still some use cases where Ni-Cd is allowed but these are mainly special cases apart from consumers [17]. Ni-Cd is included to the comparison in Table 1 mainly because it used to be a very popular battery type and used in a wide variety of devices.

Table 1. Overview of the main characteristics of the most popular rechargeable battery systems [17].

Battery system	**Nickel-cadmium (NiCd)**	**Nickel-metalhydride (NiMH)**	**Lithium-ion (Li-ion)**
Typical operating voltage (V)	1.2	1.2	3.6
Discharge Cutoff Voltage (V)	1	1	2.5-3.0
Specific energy density (W*h/kg)	45-80	60-120	100-190

Self-discharge rate(%/month)	20	30	<10
Cycle life	300-700	300-600	500-1000
Maintenance requirement	30-60 days (discharge)	60-90 days (discharge)	Not required

The battery in the present project is an 18650 type (known also as 168A) rechargeable li-NMC battery with cylindrical shape. The type naming of the battery comes from the dimensions of the cell where the diameter is 18 mm and the length 65 mm. The battery has nominal voltage of 3.6V and capacity 2400mAh, the cut-off voltage is 3.0V which also determines the minimal possible voltage level for the device. The charging voltage is 4.2V and maximum charging current 0.7C (1680mAh).

2.5 Battery Management System (BMS)

In general a battery management system signifies a system which manages and controls the charging and discharging processes of a battery. The system consists of components which each have their own purpose and depending on the overall design the precision may vary greatly. Most basic battery controls are built from charge control and capacity monitoring whereas more sophisticated systems have more monitoring and control over the overall process and can monitor the health of the battery, which can be expressed as State-of-Health (SoH) [1,3].

The battery management system is the foundation to determine and represent the battery level, which is expressed as state-of-charge (SoC). Since the very first batteries there have been numerous solutions to determine and express the state represented as a percentage value from the maximum possible charge related to the capacity. [1,25]. In 1938 Heyer introduced a battery capacity indicator as a gauge meter [1,17].

There are multiple methods to determine the state of charge, one way is called the current integration method which is also known as coulomb counting or book-keeping system. A coulomb counting is a method where the battery energy is measured and integrated by time, it measures how much charge is flowing during discharge processes and tries to estimate the state of charge. This method has problems with accuracy since the

measurements starts to drift after a while and it needs to be re-calibrated by time to time in order to maintain accuracy. In order to use coulomb counting it needs to be utilized with appropriate hardware which is usually in external component and can be accessed through a data interface [1,33].

The second way is using a voltage method where the battery voltage is used to determine the state of charge. This method is also called a look-up table method where the voltage is compared to table that has determined values for the sate of charge. Since the output voltage of the cell varies on different temperatures the temperature needs also to be measured and populated in the look-up table, the accuracy of this method depends on how many values the table has [1,28]. For this solution a temperature sensor and an analog-digital converter is needed, Figure 4 visualizes how temperature affects the output voltage.

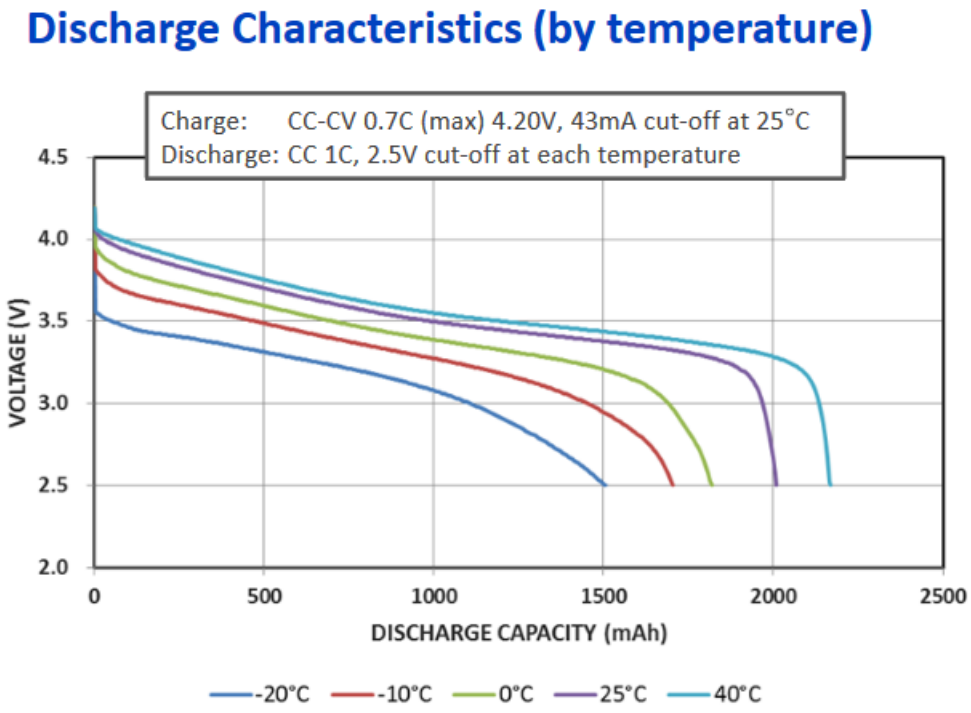

Figure 4. Discharge of an 18650 type li-NMC cell.

The third way is to use an adaptive way which use of coulomb counting and voltage method as well as keeps record of how many times the battery has charged and discharged in order to estimate battery life with co-operation of filters. The adaptive way is considered to be best one in cases where the battery usage and user behavior are not

predictable. There are proprietary components available which are designed to be a whole solution and communicate through data buses and report the state-of-charge and state-of-health directly to the microcontroller which needs to report the user about the situation.

2.6 Charging Li-ion Battery

Li-ion batteries have safety features that are mandatory for safe usage, in addition to gas vent the battery has a safety circuit to protect the cell from harmful operations which could have catastrophic outcomes. One of the purposes for the protective circuit is to observe the cell's voltage level, when the voltage level decreases below designed level, which manufacturer has determined as minimum level, the circuit does not allow any more energy output from the cell, this feature is called as cut-off voltage. Additionally the circuitry observes and limit charging characteristics that are too high as high voltages and currents may lead to fire hazard [21].

There is ready made charging controller components available for a lithium-ion, the simplest are for a certain cell type whereas some have more features which can be configured. Figure 5 shows a circuitry where R1 resistor with different value configures features and modes, in this case the charging controller is configured to act automatically when a battery is connected thus is working as an independent system. However, there are systems available that are programmatically configurable and even systems that have all of the functionality, including state-of-charge exist.

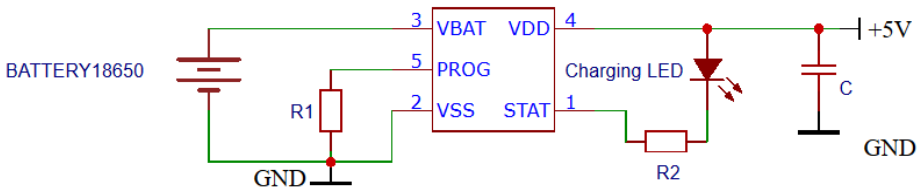

Figure 5. The charge management controller circuit.

Charging a Li-ion battery is rather straightforward, charging is usually made in three steps, in first step the controller checks if the voltage of the cell is at least at cut-off voltage, if it is not then the charger uses only 10% of the charging current until the voltage of the cell is at the minimum level to switch into full charging current. In the second step

the battery is charged with constant current (CC) until the voltage level of the battery is increases to 4.2V. In the third step, when the voltage level is achieved, the battery is charged with constant voltage, at the same time the current is slowly decreased until it is only ~5% of the charging current. Figure 6 visualizes how charging is executed.

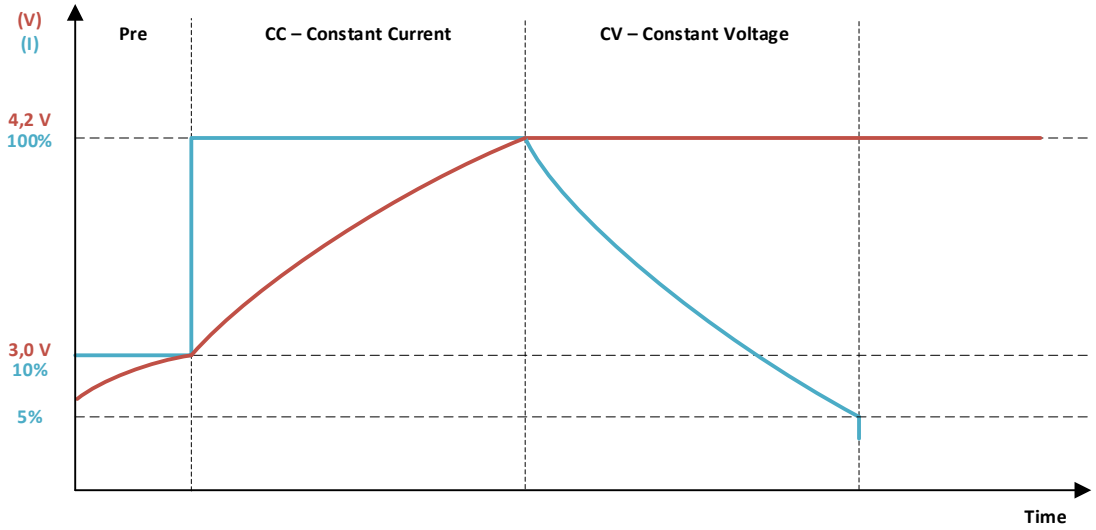

Figure 6. Complete charge cycle.

The device has an appropriate li-ion charge management controller which takes care of the charging and observes the battery level. The controller is capable to automatically detect when the battery is present and when the charging cable is plugged or unplugged, the circuit has an LED which is activated when the charging is in progress.

In general when a charger is connected the controller checks the battery level and charges it if needed, the charging starts with pre-mode (Figure 7) if the voltage level is too low, after that the charger changes to the full charging mode where the battery is charged with constant current until the voltage of the cell has reached 4.2 volt, then the charger changes mode to constant voltage and feeds the cell with voltage until the charging current is only at the level of 5%. Then the charging controller changes to upkeep mode where it observes the battery level and recharges if necessary.

2.7 Overview of Voltage Characteristics of System

The system is powered with a battery of which the output voltage varies between 3.0V and ~4.17V. Large variations are not ideal for components and make the voltage reading not possible as described in the next chapter. The device has a voltage regulator component that takes the output voltage of the battery and regulates it to constant 3.3V with ±5% error rate. In order to create a power manager the overall voltage characteristics need to be known to determine thresholds.

However, the battery output is not directly used to power the overall system, the circuit is designed to use 33 V constant voltage and needs to be stable. To provide stable voltage a power regulator is used, the regulator component is capable to convert input voltages from 2,1 V to 16 V, the regulator is used to provide constant output of 3,3 V. Regulated voltage not only makes the overall circuit design more straightforward it is also mandatory to get reliable voltage level measurements and thus is fundament part of the power measurement. The minimum power level of the system is already observed with the ADC readings and set to 3,3 V. In order to create a comprehensive power manager the voltage levels of all components need to be known in order to understand and determine limits. All necessary voltage limits are listed in Table 2.

Table 2. Peripherals with minimum and maximum voltage, populated from highest to lowest.

Component	**Minimum voltage (V)**	**Maximum voltage (V)**
Regulated power circuit	3.3	16
Battery	3	4.2
Microcontroller	2.4	3.6
IMU sensor	2.4	3.45
Radio	2.1	3.6
LED display	1.8	2.3

The component with highest minimum voltage determines the absolute minimum voltage level of the system, according to Table 2 the highest minimum voltage is associated with the regulated voltage of the system which is used to power the components and it is also

used to determine the state of charge. Observing Table 2 shows that the battery would yield before any component would. All of the components have their purpose and play an important role.

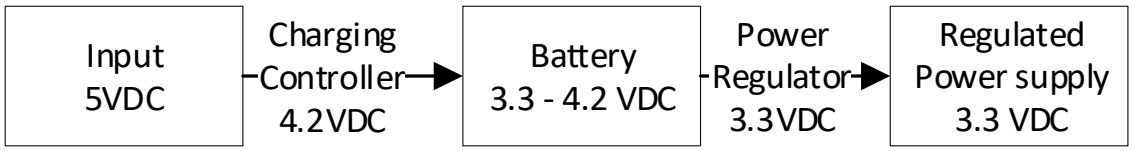

Figure 7. Voltage level conversion and distribution.

The complete power distribution of the Device is explained in Figure 7. The charger feeds the device with 5VDC which is then controlled by charging controller and regulated down to 4.2 volts to charge the battery. The battery then feeds the overall system of which the output is regulated down to 3.3V. The voltage characteristics are now known as the minimum voltage is 3.3VDC which is required by components and the state-of-charge and the maximum voltage is 4.2V which is determined by the battery. Components themselves are capable of operating on lower voltages which may be beneficial in other devices.

2.8 Reading Voltage Levels (A/D Conversion)

As the state-of-charge is determined from the voltage and as the voltage is an analogue component and the processor is digital component there needs to be some way to get analog signal into digital world. A common solution for this is to use analog-digital converter (ADC), the overall procedure of reading and handling the analog signal with ADC is called digital signal processing (DSP) [5,1].

There are many different architectures and designs for reading analog signals ADC, each having their own strengths and weaknesses in different applications. The microcontroller has a built-in analog digital converter which is based on successive-approximation-register (SAR) architecture, its advantages are that it has low power consumption and high accuracy [7,7].

When making a quick comparison between different ADC architectures the SAR, the type ADC can be considered as a good choice for measuring the voltage level, one of the advantages is that it is in many cases built-in to the microcontroller and does not need complex design for the circuitry as long as the measured input voltage does not exceed the reference voltage and the tolerance of the chip. If the measured voltage is above the reference voltage, then the SAR ADC's conversion algorithm fails in comparison and ADC readings are invalid.

One of the competitive designs is Pipelined ADC, when compared to the Pipelined architecture, the pipelined ADC has parallelism which has higher throughput but uses more power and has more latency [7,6]. When compared to the Flash ADC architecture, the Flash ADCs are faster and have higher sampling rates but are more complex to design and implement thus are not commercially viable solutions [7,6]. When compared to the Sigma-Delta architecture, Sigma delta converter ADCs are primarily suitable in digital audio applications, they consume more silicon area which brings some design challenges. Sigma-Delta also has higher resolution which comes in cost of speed since producing a final sample needs at least 16 times sampling [7,6]. Figure 8 illustrates working principle of SAR ADC.

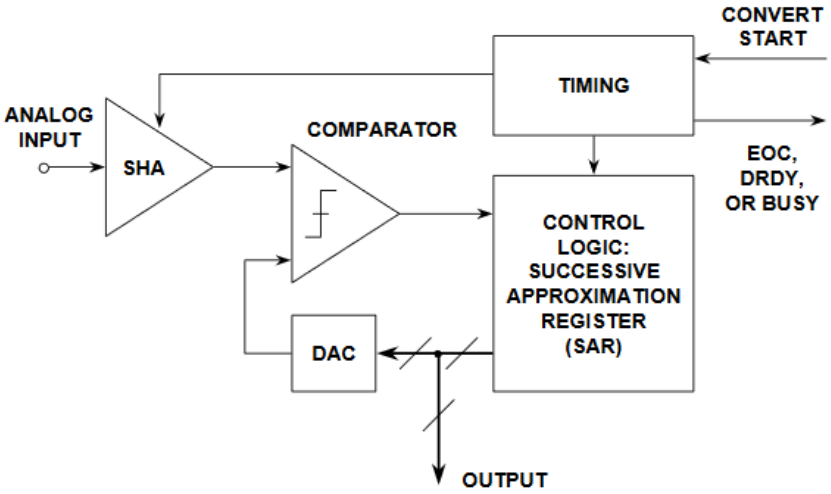

Figure 8. Basic Successive Approximation ADC [6,185].

The working principle of the basic successive approximation ADC is quite straightforward, when the conversion is started (Figure 8) the ADC goes into busy state and starts

conversion, first it gets a sample of the regulated reference voltage and reads the analog input voltage with the sample-and-hold (SHA) circuit, in the beginning the successive approximation register is set to zero and the most significant bit (MSB) is set to 1. The actual conversion is made internally by converting digital signal to analog (DAC) and run this through comparator. The comparator runs the value of the reference voltage against the signal coming from the SHA, the comparison is made by cycling through bits, if the signal from DAC is higher than analog voltage, then the current bit is reset (set to 0) and next bit is set (as 1), this cycle continues until all the bits have cycled through creating an analog signal represented as value in binary format [6,186].

The built-in ADC has a resolution of 12bits which means that valid conversion indicates the value from 0 to 4096. The sampling rate in the present project is as high as 2 million samples per second which means that the ADC can take samples from the analog voltage at that rate.

The ADC output can be calculated using the following formula (Formula 1):

$$\frac{(2^{12}-1) * AnalogInput(V)}{Reference(V) - MinimumVoltage(V)} = ADCOutput$$

(1)

Formula 1 is provided by the manufacturer and is valid when the Input(V) does not exceed the Reference(V). The reason for the validity is that the design of the SAR type ADC is not able to output measurements when the reference voltage is lower thus it is not possible in practice to approximate the value [6,185].

The manufacturer of the microcontroller has instructed that the input voltage of the ADC should not exceed power voltage which in this case is regulated 3.3V. The manufacturer warns that higher voltages will lead to unreliable measurements and shorter life expectancy. As the battery output is well outside the manufacturer's instructions the voltage needs to be scaled down, this has been solved with a voltage divider visualized in Figure 9.

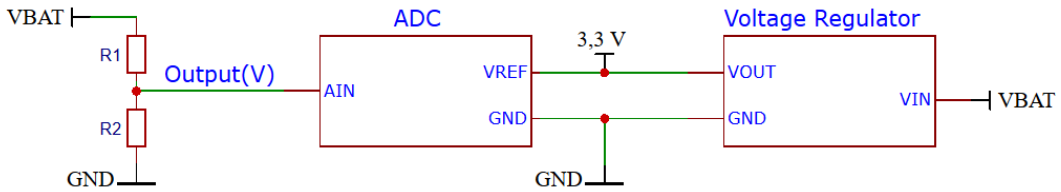

Figure 9. Simple voltage divider

The voltage divider is an analogous circuitry which needs to be designed and implemented when the device is designed thus cannot be implemented afterwards. The voltage divider of the device is designed to decrease the measured voltage from the battery by 66.67% which solves the problem and makes voltage levels more acceptable for the ADC. Formula 2 is used to solve the voltage problem.

$$Output(V) = VBAT * \frac{R2}{(R1+R2)} \quad (2)$$

Formula 2 is based on a voltage division rule where using Ohm's law reduces the magnitude of the voltage.

Since all voltage characteristics are known Formulas 1 and 2 can be used to calculate all the expected values and the values can be used as guidelines. Formula 2 is used to check if the voltage divider works as expected since those values are used to pre-calculate the expected ADC measurements. Voltage divider values are checked with a multimeter when actual measurements are done. All pre-calculated values are represented in Table 3.

Table 3. Expected voltage divider values and pre-calculated ADC readings.

Supply voltage (V)	Voltage divider output (V)	ADC converted value
4.2	1.40	1737
4.1	1.37	1696
4.0	1.33	1655
3.9	1.30	1613
3.6	1.20	1489

3.3	1.10	1365
3.0	1.00	1241

Table 3 populates the expected values which were calculated with Formulas 1 and 2, these values are going to be used as guidelines to check if there are any variations on different levels. For testing purposes six values between minimum 3.3V and maximum 4.2 are reviewed. For testing purposes also a lower value than the reference voltage is tried and the results are reviewed.

3 Implementing State of Charge

In this chapter the knowledge researched from theory in previous chapter is used to implement a state of charge indication and some safety features regarding the voltage levels are created to protect the device. Expected values calculated from theory in previous chapter is used as a guidance while developing necessary features.

3.1 Implementing ADC

The ADC of the microcontroller has many options as to how it can be configured, one way is to generate interrupts on different events, for example the ADC may generate an interrupt when it detects a crossing on high or low threshold. The ADC is capable of operating in a low power mode which may be useful in some cases where an interrupt is needed. In this project the ADC does not need interrupts since the ADC is going to be manually triggered by software every two seconds and performs a single ADC conversion.

In order to use the ADC it needs to be initialized and configured according to the manufacturer's instructions which clearly state that when an ADC conversion is performed it needs certain digital signal processing in sequence to avoid spurious conversions. The manufacturer of the microcontroller has provided a chip library as a driver with helper functions to speed up software development. Listing 1 shows how the ADC channel is initialized and configured to use software trigger related features and how the actual conversion sequence is executed.

```
static void initADC() {
      /* Setup ADC for 12-bit mode and normal power */
      Chip_ADC_Init(LPC_ADC, 0);

      /* Setup for maximum ADC clock rate using sycnchronous clocking */
      Chip_ADC_SetClockRate(LPC_ADC, ADC_MAX_CLOCK_RATE);

      /* Setup a sequencer to do the following:
      Perform ADC conversion of ADC channels 0 only */
      Chip_ADC_SetupSequencer(LPC_ADC, ADC_SEQA_IDX,
                  (ADC_SEQ_CTRL_CHANSEL(1) | ADC_SEQ_CTRL_MODE_EOS));

      /* Set voltage trim (Reference voltage is between 2.7 V and 3.6 V */
      Chip_ADC_SetTrim(LPC_ADC, ADC_TRIM_VRANGE_HIGHV);

      /* Need to do a calibration after initialization and trim */
      Chip_ADC_StartCalibration(LPC_ADC);
      while (!(Chip_ADC_IsCalibrationDone(LPC_ADC))) {}

      /* Clear all pending interrupts */
      Chip_ADC_ClearFlags(LPC_ADC, Chip_ADC_GetFlags(LPC_ADC));

      Chip_ADC_EnableSequencer(LPC_ADC, ADC_SEQA_IDX);
      /* ADC input 1 is on PIN mapped to FUNC1 */
      Chip_IOCON_PinMuxSet(LPC_IOCON, PORT, PINNUM, (IOCON_FUNC1 | IO-
      CON_MODE_INACT | IOCON_ADMODE_EN));
}
```

Listing 1. ADC configuration and initialization function.

When the ADC conversion is triggered after every two seconds the sample is read twice and as the manufacturer has instructed that the first sample should be discarded. The driver has confirmation functionalities to communicate with the ADC to determine if the data is valid, after confirmation the value is read from the registry of the ADC and converted to integer. During implementing the ADC all ADC values are output to the debugging terminal to examine if the pre-calculated values were correct and to investigate if the driver works as intended.

For implementing the ADC functionality a development environment was built which consists of a Windows PC, high precision digital multimeter (Hewlett Packard 34401A) and a laboratory power supply (Twintex TP-1305) which output was measured with True RMS capable digital multimeter (CRM DT-9918T). The debug terminal was connected from the UART port of the device to the PC which had Putty installed on it. All measurements were logged into an Excel spreadsheet and turned into graphs for visual inspection. Figure 10 shows the testing environment.

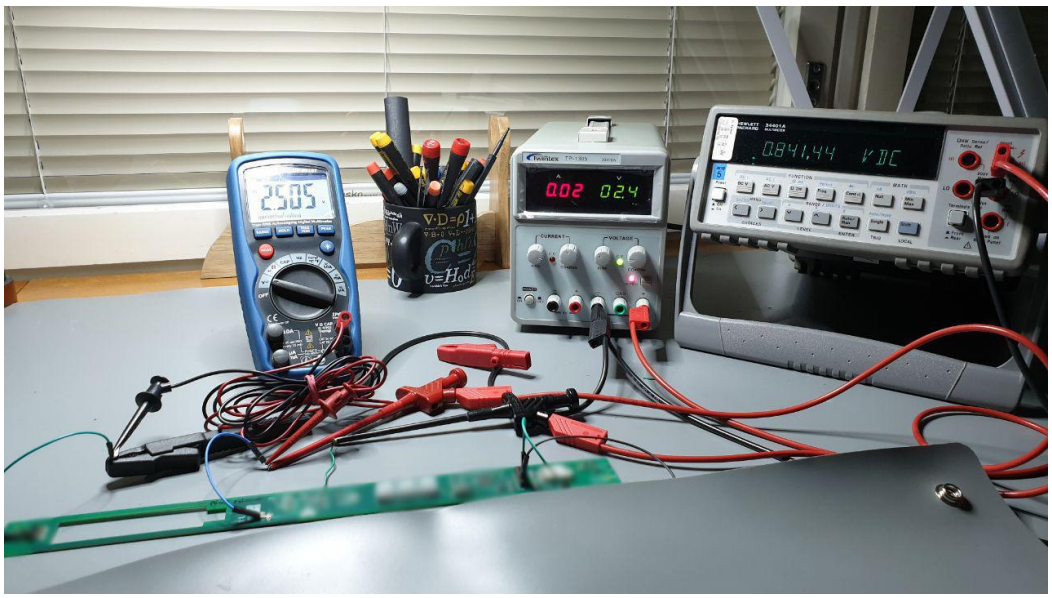

Figure 10. Test setup for ADC reading

The testing environment shown in Figure 10 was built to be ESD protected and had all necessary tools for inspecting voltage levels. The device was supplied with laboratory power which voltage level was inspected with a multimeter, the voltage after voltage divider was observed with the high precision multimeter.

3.2 Interpreting ADC Readings

The device was powered with regulated and measured voltage to investigate how the voltage divider performed in real life, pre-calculated values were used in comparison. The voltage divider was working as expected with a little variation which can probably be explained by tolerances in resistors, the calculated voltage drop is approximately 66,67 percent whereas in real world the actual voltage drop is between 64.6-66,5 percent. The results are shown in Figure 11

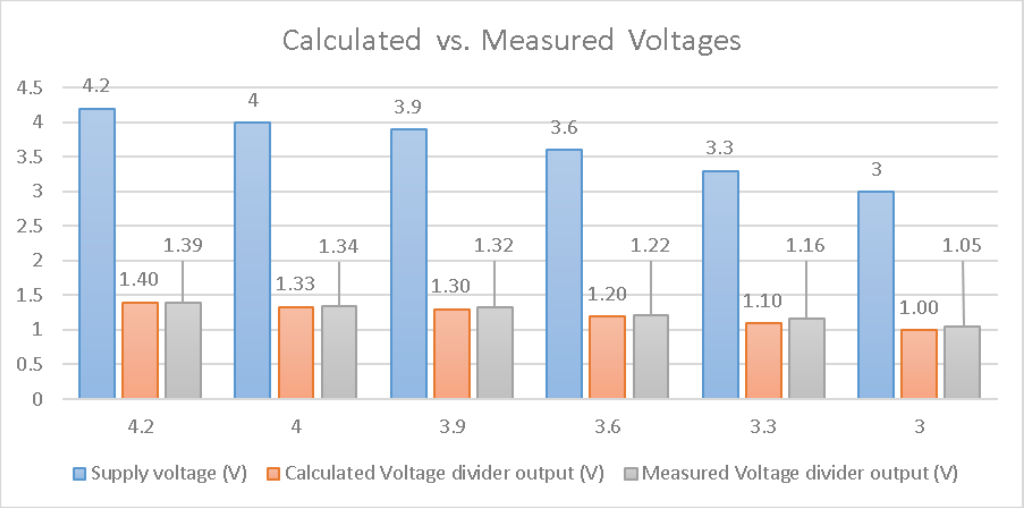

Figure 11. Comparison between calculated and measured voltages on voltage divider output

After the voltage divider was measured and confirmed the measurements did not match with the pre-calculated values since they were based on zero tolerance in the output of the voltage divider, when all pre-calculations were corrected with the actual measurements the numbers started to match with only slight variations. ADC measurements are shown in Figure 12.

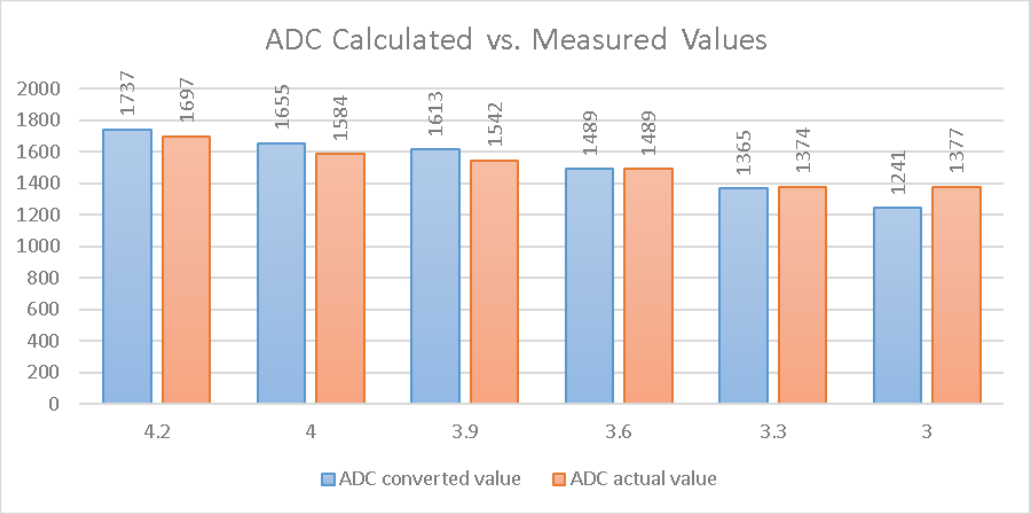

Figure 12. Comparison between calculated and measured ADC values

Figure 12 shows what happens when the supplied voltage is under the reference voltage and the ADC is sampled, the reading becomes unreliable since the reference voltage is not the same anymore thus the result is not comparable with other results.

3.3 Determining State-of-Charge

The device does not have proprietary components for determining the state of charge, as the device is intended to be used indoors where the temperature is stable and the characteristics of the battery are known, the state of charge of the battery can be determined from the voltage reading. With the voltage reading approach the precision is quite similar to that of the look-up table but without temperature compensation which makes the state of charge to be more alike a ballpark estimation than precise indication. The state of charge can be obtained with Formula 3 where the SoC can be calculated from the ADC values.

$$SoC(\%) = 100 * \frac{MeasuredValue - MinLevel}{StepsBetweenMinAndMax} \quad (3)$$

Formula 3 determines the state of charge from range between the maximum and minimum value.

Values used in Formula 3 are measured and determined with the ADC and can be found in Table 4. The state of charge can now be determined which was the goal for the first part of the study but since the voltage of the battery can drop below the level which is needed for the voltage regulator to work without problem, the state of charge feature needs to have some safety features since at too low a voltage the device starts to have troubles and the device becomes unreliable which is not user friendly.

The values in Table 4 were read from the ADC in normal room temperature. The table acts as a safety guidance when designing the safety features as well as some kind of a lookup table which is used to determine the overall state of charge that can be of guidance for the user to indicate when a recharge might be necessary.

Table 4. Thresholds related to safety and user notifications.

Item	**ADC value (Battery%)**	**Value in Formula 3**
Minimum value	1377 (0%)	MinLevel
Safety threshold for power down “Low power function”	1385 (2%)	-

Full capacity	1736 (100%)	-
Charging “Charging started”	>1736 (>100%)	-
Level high	1629 - 1736 (70 - 100%)	-
Level medium	1521 - 1628 (40 - 69%)	-
Level low	1413 - 1520 (10 - 39%)	-
Level critical	1385 – 1412 (2 – 9%)	-
ADC measurement	1377-1736 (0-100%)	MeasuredValue
Distance between min. and max.	359	StepsBetweenMinAndMax

Values in Table 4 are used to define the safety thresholds, the battery has four different determinations for levels which function as guidance for the user. Levels *high* and *medium* are levels where the user does not need to think about recharging, in level *low* the user needs to start considering recharging and *critical* is the one where the user needs to seriously consider recharging since when the battery level is at *minimum value* the safety features of the device will turn the device off.

3.4 Implementing Power Manager with Safety Features

The main purpose of the power manager is to take care of reading the ADC and determine the state of charge with the help of thresholds determined in Table 4. One of the most important functions is to protect the device from low voltages, in case of low voltage the manager will inform the user about low voltage and powers the device off. When the device is powered again the manager will immediately check the state of charge and if the voltage level is under the low level threshold then the user is informed about low power and the device is powered off. The flow of the power level check is visualized in Figure 13.

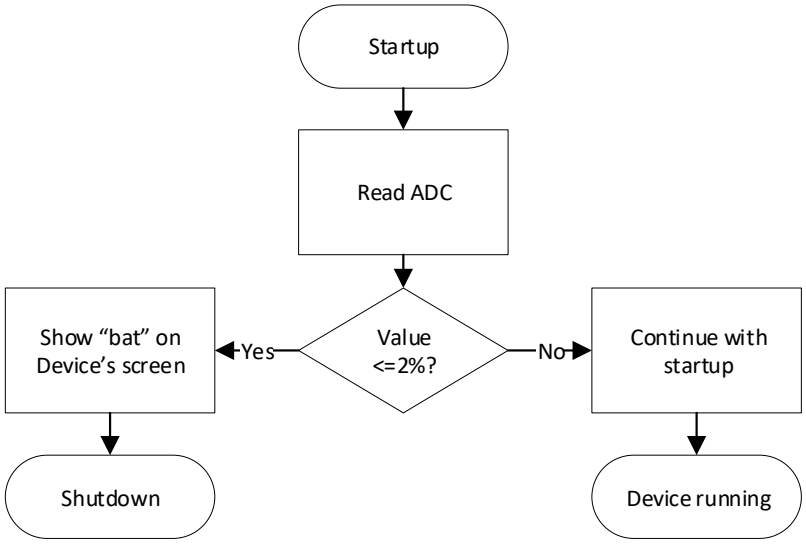

Figure 13. Flow of the power level check during the startup.

When the device is booted and the state of charge is above the minimal threshold the power manager allows the device to start. Figure 14 visualizes how the power manager functions after a successful start. The power manager reads the ADC after every two seconds and calculates the state of charge, if the state of charge is less or equal to 2 percent the power manager powers the device off, otherwise the device keeps running normally.

The device is capable of reporting its state of charge to a connected mobile device. Reporting is done after every five minutes or on a request, however the state of charge is not reported if the device is under heavy load. The power manager is responsible for informing the user about charging and state of charge. When the device is connected with a charger the ADC reading is noticeably higher than what the battery can output, thus it can be assumed the device is charging. When the device detects charging the power manager changes the mode of the device to charging and notifies the user momentarily with a message “chg”.

When the charger is unplugged the ADC reading drops to a level that is typical for a battery. The charging manager changes the mode of the device to not charging and shows the user momentarily the state of charge. When the device is powered off the power manager is also off which means the user is not notified about charging. In order

to detect charging, for example in cases where the user is uncertain if the device is charging, the device needs to be powered on.

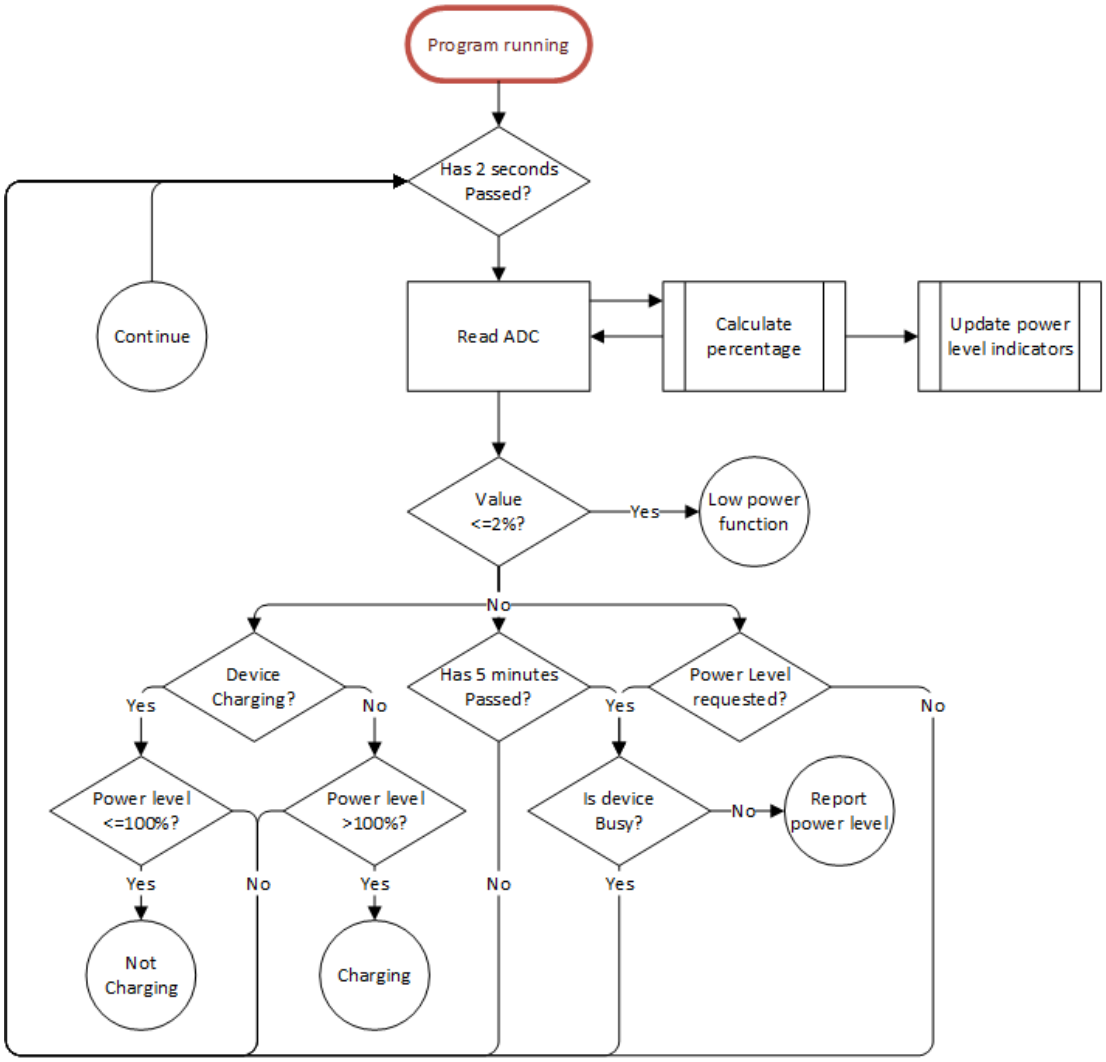

Figure 14. Flow of the power manager

The power manager in general has an important role on how the device operates, on low voltage the device becomes unreliable and the state of charge does not work. The overall user experience and reliability plays an important role on how users are going to adopt the usage of the device. The power manager has a great impact on how the device operates in certain cases, for example in a situation where the device is connected to a mobile app and the mobile app has a software update for the device, in that case the power manager checks the state of charge and if it is low it informs the user to plug the charger to continue.

4 Power Saving

Power saving has many advantages over the device's lifespan and user experience. In general a whole device consists of multiple parts where each part has their own purpose and power usage. The whole power usage can be divided into multiple sectors as demonstrated in Figure 15.

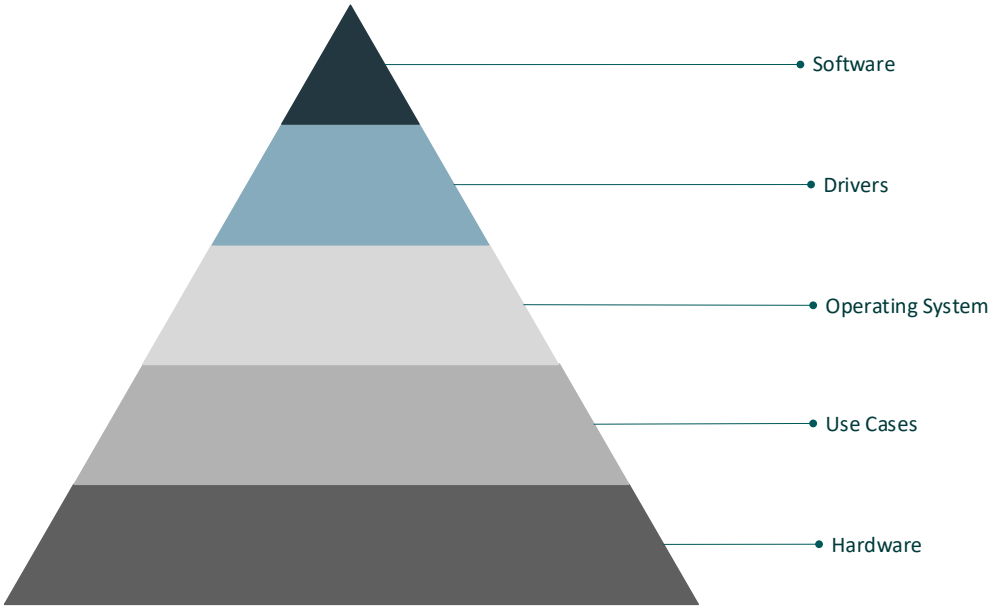

Figure 15. Overview of a typical structure of device's working principle [19].

In Figure 15 the hardware is the base of a device, the critical part of the power consumption is determined by selected parts. A device generally consists of multiple components where the microcontroller acts as the brains of the device. Many components have low-power features and these features can be configured and utilized programmatically accessing them with the microcontroller. However, some components cannot be accessed with data buses and are configured with the design of an electrical circuit and thus acts as an independent system, for an example in this project the battery charging control is an independent system which was configured when the device was designed [19].

On the second level there is use cases which is not a physical component but it is as important as it consists of real world scenarios where the basic usage and power consumption of the device is thought and estimated, in practice the usage of the device is predicted and taken in consideration from a software perspective. For example when a

device is in use it needs to perform accordingly to the determined requirements and a possible implementation of power features needs to be carefully reviewed and designed so the device would not suffer from underachievement [19].

On the third level there is a real time operating system (RTOS) which is the closest software to hardware and widely used with microcontrollers as it has a small size and has a kernel which is capable of taking in advantage and control the features of the microcontroller. An RTOS is small sized and equipped with powerful features and plays well with the low power features of the microprocessor. Using an RTOS simplifies the overall software development.

On the fourth level are the drivers which are usually device specific and used to making use of the capabilities of the device, in practice a driver consists of helper functions that make it easier for a software engineer to configure and utilize devices. For example a device can be configured to generate a signal when certain conditions are met (interrupts) or some functionalities can be disabled in order to reduce power consumption. Drivers may include precautions and instructions for cases where the device has hardware limitations or the device manufacturer has determined specific directions with interacting with the device [19].

On the top level the software engineer binds everything together by creating software that makes the device to operate. From this level the software with co-operation of the operating system can control the whole device, for example the software may have a power manager which reduces power usage by putting the hardware of the device to sleep when the device is not used for a short period of time and performs shutdown when the device is non-operational for a long period of time [19].

4.1 Microcontroller

In a modern microcontroller the power consumption is generally already minimal by design and has many configurable low power features. Although the power consumption of he system is greatly influenced by hardware selections and should be taken into consideration when selecting hardware, a software engineer has the possibility to take advantage of low power features and reduce power consumption even more by designing and implementing power saving features programmatically.

Depending on the microcontroller architecture there is usually many different options for low-power features but mainly they can be roughly determined and divided into two classes, i.e. suspend and hibernate. In both classes there are advantages and disadvantages, primarily in the areas of power and responsiveness [19].

In the suspend mode the microcontroller is powered down except for the memory which is not powered down to maintain the running software in the random-access memory (RAM) for a quick resume. In the suspend mode the power consumption is noticeably lower and the wake-up time is considerably faster compared to hibernate [20]. This mode is suitable in solutions where the microcontroller of the device can work at lower clock speed and still be responsive enough for the required purpose.

In the hibernate mode the microcontroller is completely powered down and the software is written in flash or other non-volatile memory (NVM) that is capable to hold the data while the memory is turned off. The power consumption in hibernate mode is significantly lower but resuming from hibernate takes a longer time since the software needs to be copied back to the RAM, but is still considered faster than cold boot [20]. This mode is more suitable in solutions where the device can sleep longer times and longer startup time is not an issue, for example in solutions where a sensor is first read, then data is passed forward after which the device can resume back to the hibernate mode.

In the present project the microcontroller is based on a 32bit Cortex M0+ processor which is built on the foundation of an Arm v6-M architecture, the architecture is designed and patented by Arm Holdings. Arm Holdings does not manufacture microprocessors, their goal is to develop and improve microcontroller architecture by their own and in co-oper-

ation with licensees. In generally the Arm Holdings is focused on developing the processor core (CPU) of the architecture where licensees build other functionalities such as data buses, device controllers, programmable input-output ports and communication buses etc. around the core which is called System on chip (SoC).

By design the M0+ is a low powered processor and has built-in low power functionalities which can, depending on usage, reduce the power consumption even more. The low power features are built-in the Cortex M0+ core and can be interacted with SYSCON block and SCR register, these features are gathered under a power-management-unit (PMU). The SCR register is a software interface for configuring the microprocessor before entering to a low power mode, in practice it plays a critical part when the microcontroller enters and exits from the low power mode. The SYSCON block stands for system configuration and is always present, it is used for configuring the functionalities of the microcontroller such as peripherals, interrupts and clocks when utilizing core features such as entering and leaving a low power mode.

In order to utilize low power features the microcontroller needs to be told what to do when entering the low power mode, when or how it is going to exit from it and what actions need to be taken when exiting a low power mode. Basically when the system is put into the low power mode it can be configured in multiple ways, one thing which is important to take care of and handle is the clocking of the system so all the timing sensitive functions will not break which would cause the system to become unstable and which would cause numerous reliability problems.

One of the Cortex M0+ low power features by design is that all unnecessary peripherals and features are powered down by default which makes controlling low power features on a software level much more straightforward since the software engineer has to generally take care of the features which have been enabled manually. The microcontroller has a total of five different low power modes which all have their own advantages and purposes which are suitable for different use cases. General characteristics for these modes with the manufacturer's reported typical power consumption are listed in Table 5.

Table 5. Low power modes from highest to lowest power consumption with typical power consumption.

Level	Power modes	Features	Typical consumption (mA)
0	Active mode	The basic system features that are enabled by default.	7.8
1	Sleep mode	Most basic, affects only Cortex M0+ core. Everything else will work as configured.	3.3
2a	Deep-sleep Mode	Clock is not received by peripherals and the internal flash memory is in stand-by mode.	0.275
2b	Power Down Mode	Clock is not received by peripherals and the internal memory (Flash) is powered down.	0.005
3	Deep power-down mode	Entire system is shutdown except some exceptions. Wake-up of will restart the microcontroller.	0.0012

Reviewing Table 5 shows that for the project purposes the low power level 3 is not suitable since recovering from the deep power-down mode means in practice that the microcontroller will reset on wake-up and perform restart which with all peripheral and system related configurations will take inconveniently long time and reduce overall user experience. For practical purposes the table reveals that levels 1 and 2a seem to be more promising for the needs and will still have significantly lower consumption compared to active mode.

For the overall performance the wake-up time may have a great impact on how fast the device recovers from a low power mode and becomes responsive, in practice when the microcontroller wakes up it needs to re-enable and re-configure all the needed peripherals and features that are necessary for normal usage of the device, these features are populated in Table 6 along with a power saving class categorization.

Table 6. Typical wake-up times and power saving class.

Level	Power modes	Wake-up time (µs)	Class
0	Active mode	-	-
1	Sleep mode	2.6	Suspend
2a	Deep-sleep Mode	4.4	Suspend
2b	Power Down Mode	86.8	Hibernate
3	Deep power-down mode	276	Hibernate

Inspecting the wake-up times of the microprocessor from low power modes in Table 6 shows that the wake-up times are increased considerably when resuming from hibernate states, in practice these numbers alone do not play a significant role since resuming from the deepest low-power state takes 0.000276 seconds which for human is really fast, however in the end the wake-up time will sum up with other times and play its own part in the big picture.

4.2 IMU Sensor

An inertial measurement unit (IMU) is a digital sensor which is used to measure and interpret movements from real-world. The sensor is a crucial component of the device since its purpose is to measure the orientation of the device and keep the user informed about it. The sensor was decided on and included in the design before the project started thus the sensor needed to be studied and utilized as efficiently as possible.

The sensor has a built-in digital signal processing unit which is capable of processing and filtering the sensor readings into noiseless and stable output. In fact, using a specially designed proprietary processor has its advantages over the microcontroller since it takes off a lot of unsuitable load which could cause the system to have timing delays which would be noticeable to the user as decreased responsiveness.

The sensor has many capabilities and features which can be configured through an Inter-Integrated Circuit (I^2C) data bus. By default the sensor is powered down and in order to utilize the sensor it needs to waken up and configured accordingly to the need. The basic configuration includes instructions about the characteristics of the connection speed between the sensor and the microcontroller, what is the sensitivity of the sensor, how the data will be outputted and reported to the microcontroller.

In the project the sensor was configured to communicate on the maximum available speed which is at 400 kHz rate. The data is put into the first in, first out (FIFO) queue and formatted as pre-calculated orientation data. When the queue is full the sensor will generate an interrupt to inform the microcontroller about new data being available, which is then read in bursts from the microcontroller and handled onwards. Table 7 introduces the energy consumption of the sensor.

Table 7. The sensor's power consumption.

Power mode	Power consumption (mAh)	Difference compared to active mode (how much smaller in %)
Active mode	3.9	-
Low power at 1.25 Hz rate	0.01	99.74
Low power at 5 Hz rate	0.02	99.49
Low power at 20 Hz rate	0.07	98.21
Low power at 40 Hz rate	0.14	96.41

The sensor has a low power feature where all operations are shut down except for acceleration sensor. Limitations of the capability of the low power mode to only sense acceleration with greatly decreased intervals makes it mostly suitable in cases where the device is in the sleep mode and needs to be waked if the device starts to move again. When the sensor is configured to enter a low power mode it is configured to sleep most of the time and will wake up periodically to check If the sensor has moved, in case of movement the sensor will generate an interrupt to inform the microcontroller. Table 7 shows that the low power mode has a great impact on power usage, the rate indicates how many times per second the core of the sensor will wake up to check if any movement has occurred during the low power mode.

4.3 Radio Module

The device has a radio module to communicate wirelessly with other devices, the chip of the radio module is based on Bluetooth version 4.0. Compared to other components on the device, the module has its own microcontroller which executes its own proprietary software that was created by previous project members, and since it is implemented with completely different coding language thus deeper inspection and modifying makes it to fall out of the scope of this project.

The module communicates through universal asynchronous receiver-transmitter (UART) protocol, when the device is powered on, the microcontroller activates the software of the radio module through this protocol. In general the software creates a relay to forward data coming from the microcontroller through UART to wirelessly connected device and vice versa, the radio module is capable of handling connections independently and does not need actions from the microcontroller. Radio's power consumption is listed in Table 8.

Table 8. Power consumption of the radio.

Power mode	Typical power consumption (mAh)
Active mode TX	16
Active mode RX	8.7
Ultra low power mode	0.0026

When the relay software is activated, the module starts to broadcast advertisements to nearby devices indicating that it is ready to be connectable, however if no connection is established within four minutes the software of the module stops broadcasting and puts the module to the low power mode, Table 8 lists the power consumption characteristics and from it can be concluded that when the device is used without any radio connections the power consumption decreases drastically.

4.4 Display

The device has three 7-segment LED display modules to represent sensor readings in a numerical format, the display segments are controlled individually and for ease of use a driver is used for representing characters or numbers on displays. In Figure 16 it can be observed that one display module contains seven segments plus a decimal point, making it total of eight indicators per module.

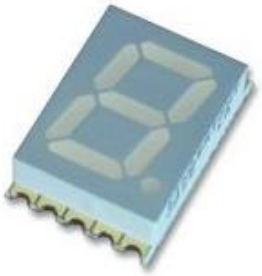

Figure 16. Image of an LED display module.

Each segment on the module uses around 24mA which makes one module to use total of 192mA when all segments are active. If all three modules were in active state, the total usage would be 576mA which is significant compared to any other component on the device. Table 9 shows typical power consumption when a number is represented on the module.

Table 9. Power consumption of a display module.

Number represented	Typical power consumption (mAh)
“8.” With decimal point	192
“9”	144
“8”	168
“7”	72
“6”	144
“5”	120
“4”	96
“3”	120
“2”	120
“1”	48
“0”	144

In normal use the decimal point is not used, since at least one module is always present the minimal power usage is when number “1” is represented and maximum when number “8” is represented. Table 9 populates power consumption on one display module when representing different numbers, the difference between minimum “1” and maximum “8” consumption is 120mA. Since the sensor values shown on the screen depend on the user behavior and cannot be determined precisely, a rough mean value is going to be used as a ballpark estimation of the power usage of the display, the mean value for all numbers from 0-9 is around 118mA, even with the mean value the display is the most power consuming component.

4.5 Real Time Operating System (RTOS)

RTOS is a real time operating system mainly developed and used on embedded systems for a microcontroller, the name real time means that the operating system can respond to its environment in the shortest possible time [9,515]. Real time operating systems are widely used and can be found in different type of solutions, from safety related products such as car anti-lock braking system (ABS) to space operated devices such as satellites.

The RTOS is generally a kernel, a kernel makes use of host system’s architecture and takes over the control of the device. The RTOS has many features and simplifies software development for a microcontroller which minimizes the complexity of the software architecture.

A typical microcontroller has a single processor core which allows the microcontroller to execute one thing at a time, real time operating system has a feature called tasks that are threads running software. Tasks makes it possible to perform more than one activity as “multi-tasking" with a single core processor, this is possible with a feature called scheduling. Scheduler is the heart of the RTOS and keeps everything in order, there are multiple algorithms that can be used to accomplish scheduling, the most common ones are cooperative, round-robin and preemptive scheduling, not all of them are suitable to function alone and often they are used in co-operation.

Cooperative scheduling (visualized in Figure 17) is generally a simple scheduling algorithm which is not ideal for a real time operating system, tasks are not prioritized by their importance and the scheduler has no control over execution time, which may cause issues in situations where a task consuming too much processing time causes problems by leaving less processing time for other tasks [9,519]. Simplified cooperation scheduling can be represented as a state machine and thus work more efficiently within tasks than controlling them.

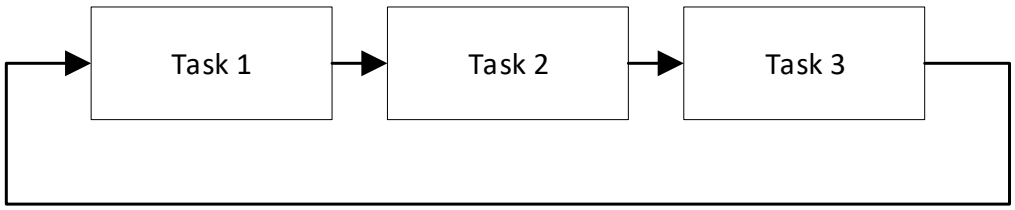

Figure 17. Simple cooperative scheduling demonstrated as a state machine [9,519].

Round-robin scheduling (visualized in Figure 18) is an algorithm where the CPU time is shared equally between each task. Tracking is done by a counter which tracks the execution time of every task and when the counter is finished the task execution is ended and put into the end of the cycle and a next task is executed and the counter starts from the beginning. This algorithm is very familiar as cooperative algorithm with the difference that the task may not complete. Because of the counter, one great drawback of round-robin is that tasks should be designed to have the same execution time, this is not useful and makes designing the software architecture more complicated. This scheduling method by alone is not useful in real time operating systems [9,519].

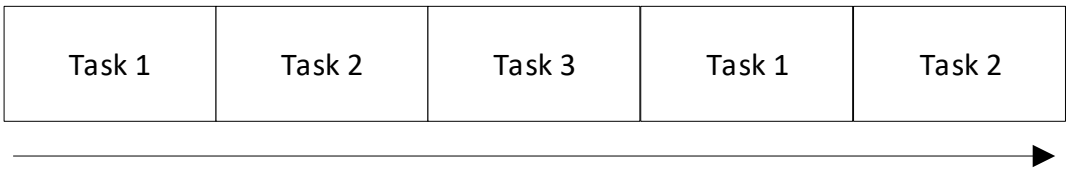

Figure 18. Round-robin scheduling algorithm demonstrated [9,519].

The preemptive scheduling algorithm is based on priority, meaning all tasks have a number that indicates a priority, a task with highest priority will get most processing time, if there is another task with the same priority level, a round-robin algorithm is used. Tasks that are not in the suspended state are handled by the scheduler, tasks may have different states, such as *ready to run*, *running* and *blocked*. Task in *ready to run* state is waiting for processing time from the scheduler, *running* state is a state where the task is currently executed and *blocked* state is a state where a task was running but a task with higher priority was awaken and needed the resource so the scheduler put the currently running lower priority task into *blocked* state. Only a task with running state can be put into *blocked* state.

When the scheduler starts to change a state of a task it performs a procedure called context switching, in a context switch the resources of the current task (context) are saved into memory and the state is changed, then the resources of the next task are loaded from the memory and the state changed to *running.* Figure 19 visualizes how the prioritizing and context switch takes in place.

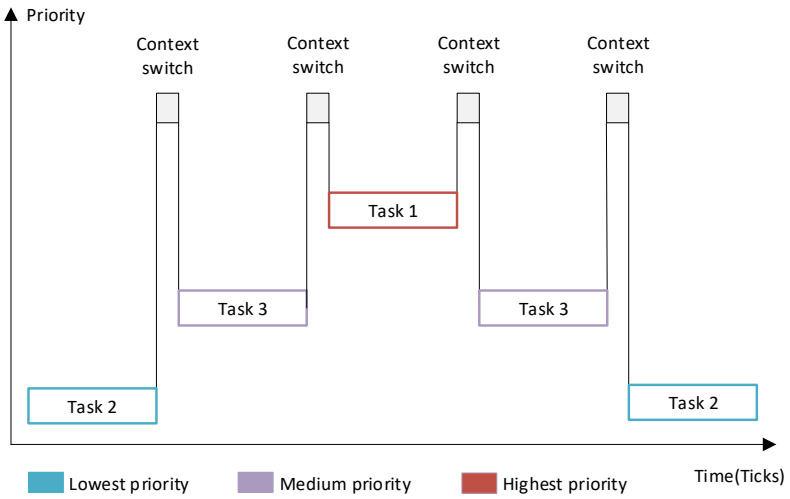

Figure 19. Priority base preemptive algorithm.

Underneath an RTOS scheduler are ticks, ticks are used for timing the operations of the scheduler thus has a crucial part in how everything works. Along with time priority number a task has a wait time defined which means the maximum time a task can wait until it needs to run again, this time is measured as tick. Ticks are generated by the systick feature which, depending on the implementation of the microcontroller, generates tick interrupts with the core clock. Constantly generated ticks at fast pace keep the microcontroller awake thus using power constantly, an RTOS has a feature called “tickles idle”. Tickless idle means that an RTOS kernel puts the microcontroller to sleep for some determined time and wakes up again when an interrupt occurs. In practice every time there is nothing to execute and the system is idling, an RTOS kernel makes use of this time and puts the microcontroller to sleep until next operation needs to be executed [10].

An RTOS has multiple ways to communicate between tasks, one of these ways are queues, in practice software can divide operations between tasks where the first task reads some data, passes it to another task which processes the data and forwards the data to the third task which is for example driving a display and represents the acquired and processed data.

Figure 20 visualizes similar behavior where the sensor is read and the data is processed and represented using queues. When a queue is configured at a receiving task it needs to provide information about how long it needs to wait at most until it is ran, in other words, the receiving queue is ran every time there is data coming from the queue but if there is no data the queue will be ran at specified intervals, unless it is configured to wait indefinitely. When the delay is configured to be maximum the task is never run when there is nothing received. Using maximum delay uses fewer resources thus the microprocessor may sleep a longer time between executions.

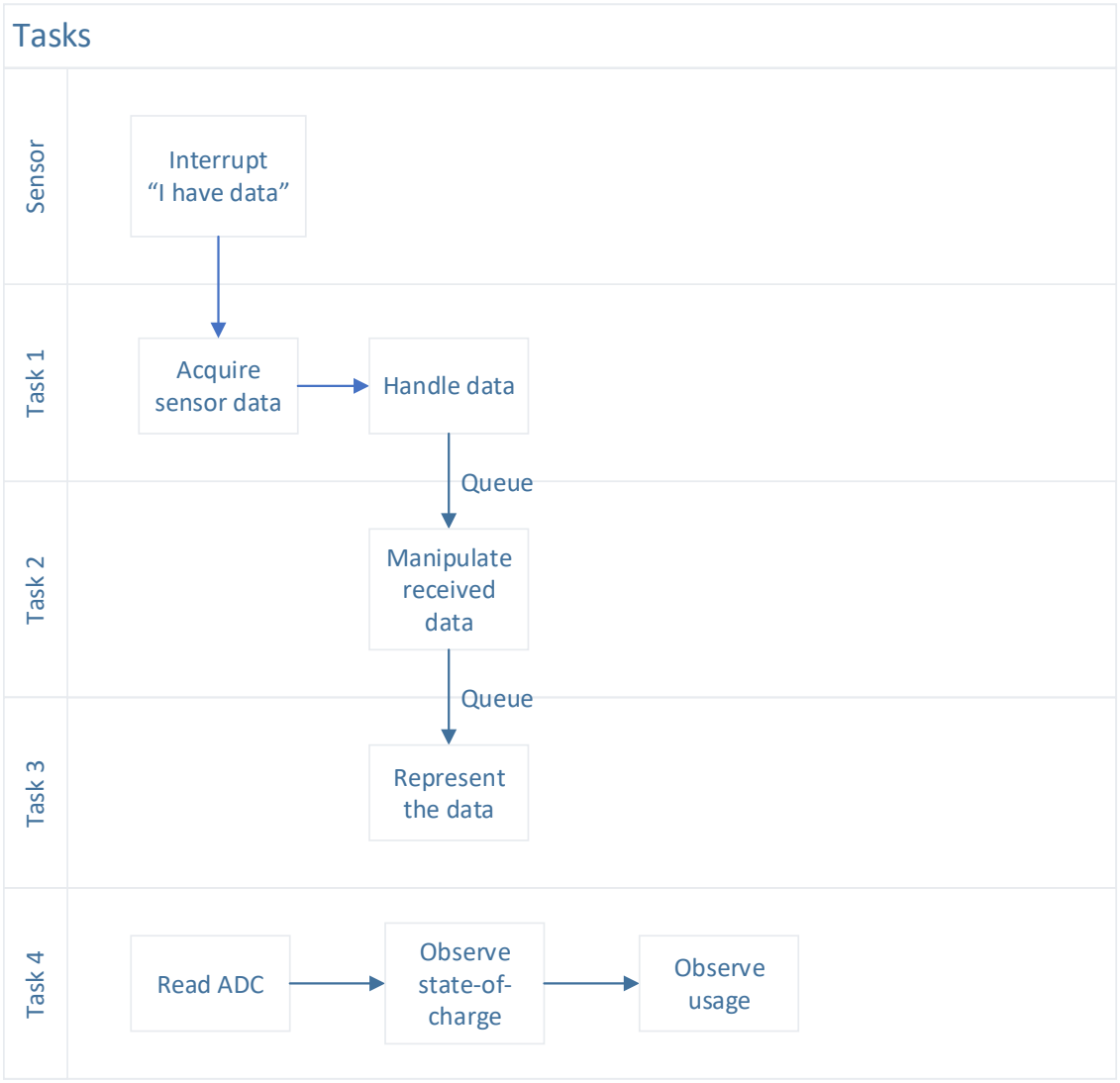

Figure 20. Example of how queues and tasks can be used.

In Figure 20 the sensor informs the microcontroller about new data with an interrupt, an RTOS then wakes up task 1 which then acquires the data, performs some signal handling to it. After the data is handled it is passed to a queue and task 1 then goes back to sleep while task 2 wakes up to receive the data from the queue, performs some manipulation to the data, passes it to queue and goes to sleep. Task 3 wakes up, receives the data and outputs it to the display and goes back to sleep. The example above makes use of the tickles idle feature and in, whereas tasks 1,2 and 3 are run only on demand, Task 4 is ran after every two seconds to perform power manager duties and act based on pre-determined behavior which was specified in Chapters 2 and 3.

4.6 Advanced Low Power Mode

An RTOS's tickles idle saves power during usage but the microprocessor and peripherals are capable of performing even more power saving with separately configurable low power modes. The advanced low power mode needs more actions from the software engineer and more thoughtful design on how everything works in practice, the purpose of a low power mode is to save power when the device is not in use thus the low power mode should not have a negative impact on how the device is operating.

Everything needs to be in order when low power features are utilized, before the device may enter to a low power mode all conditions need to be met, the scheduler of the RTOS needs to be aware when a low power mode is activated. The manufacturer of the microcontroller has a low power driver that configures the microcontroller for a low power mode, when the microcontroller is put into low power mode it needs some instructions how to operate, such as what features are going to be powered off while the microcontroller is sleeping and how it is going to exit from low power mode and what needs to be done immediately after resuming to active mode.

Figure 21 shows the basic flow what the device is doing when entering and resuming from the low power mode. When the device is not in use for ten seconds, meaning it is not moving the power manager gives permission to enter to a sleep mode. When the device starts to enter to a low power mode the processor is configured to sleep until an interrupt occurs, interrupts in this project are possible to be generate from the sensor of with a button. When the processor is sleeping all other functions except for memory are not in use, including the display which is powered off. When the device wakes up everything necessary for the normal operation of the device are configured back in use. The microcontroller is basically kept in the sleeping mode with a wait for an instruction (WFI) which is a basic property for Cortex M0+ microcontroller.

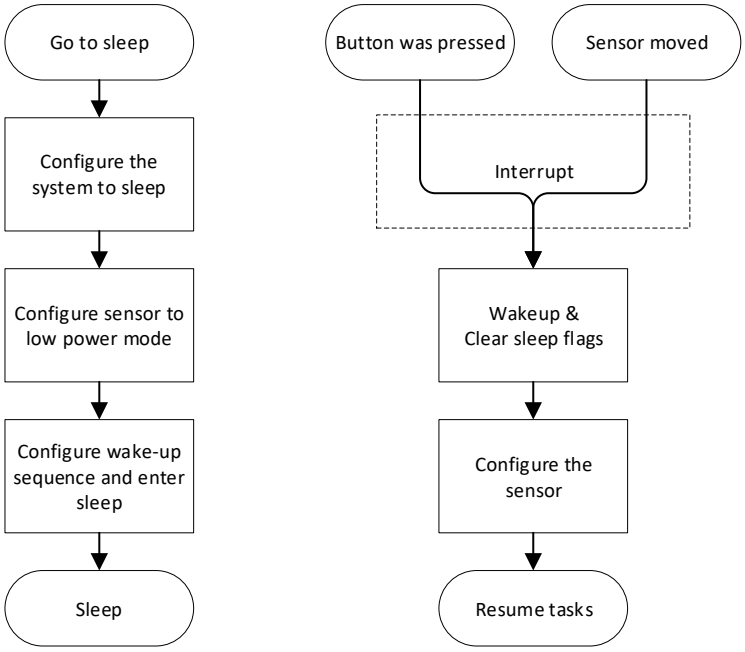

Figure 21. Flow of the low power mode

The sensor is constantly observing the environment and generating interrupts periodically which will unnecessarily wake the microcontroller from a sleep mode thus the sensor needs to be put into the sleep mode. The sensor has a low power mode where it wakes up periodically to check for any movements, when it detects movement the sensor immediately wakes the microcontroller with an interrupt.

When the sensor or button generates an interrupt the processor core becomes aware of it and starts to execute instructions which were configured before entering a low power mode. In this project the microcontroller configures all data buses and clocks to operational including the display. The sensor is configured back to its normal behavior and everything is resumed back to state in which they were before entering the low power mode.

The low power mode will save a lot of power when the device is not operational, in practice the device is usually operational for whole time it is powered on and possible occasions for low power mode are more rare but not impossible. As an extra feature the device has a power off timer where after 15 minutes of inactivity it will turn the device off in order to save maximum amount of power.

4.7 Measuring and Analyzing Low Power Modes

Measurements for the low power were carried away in same development environment as the voltage measurements introduced in Section 3. The measurement equipment consists of the same instruments but the connection was altered to measure current instead of voltage which can be seen in Figure 22.

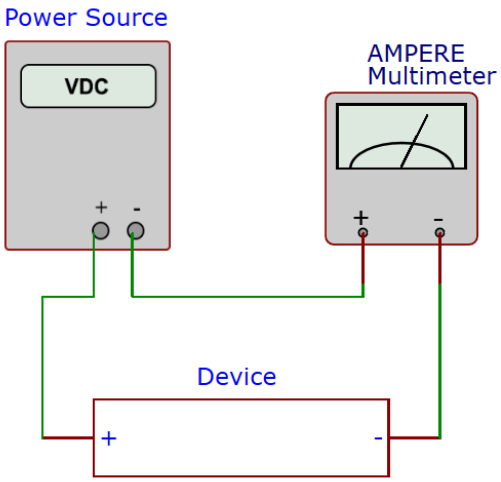

Figure 22. Power consumption measurement

For development and measurement purposes the device was programmed to enter to a low power mode with the button of the device. When the button is pressed again an interrupt occurs and the device exits from the low power mode. As the power consumption of the display varies on the user's behavior and the difference between the lowest and highest consumption was very high the display was left out from the power measurements to achieve as precise results as possible. Since the radio module is running its own with proprietary software which was not modified for this project, and thus cannot be controlled it was turned on and not interpreted in any way to create as realistic power measurements as possible. Figure 23 visualizes changes in consumption where the display is left out from theoretical measurements.

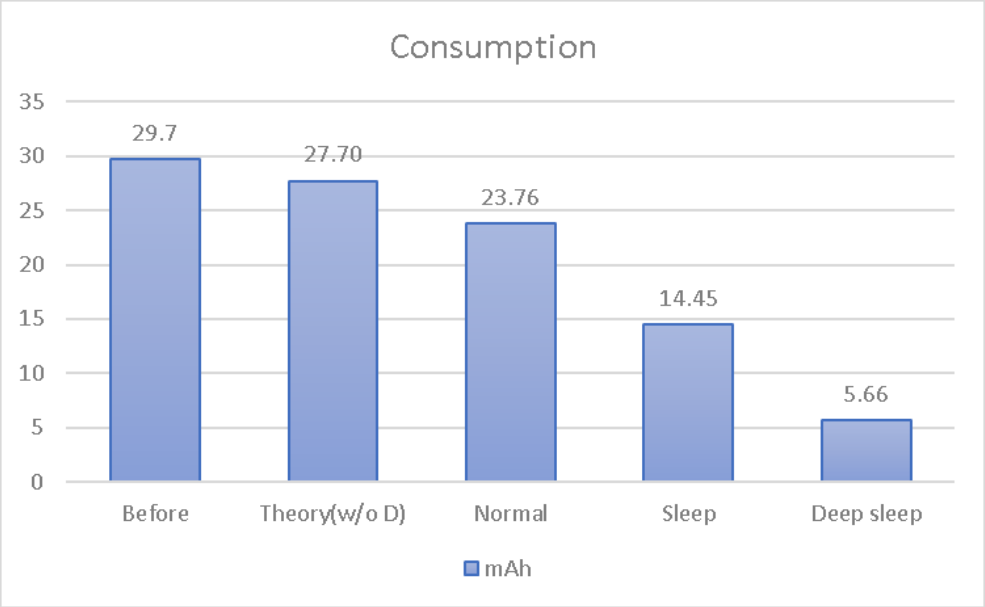

Figure 23. Power consumption in different modes.

Figure 23 shows the difference between the different modes, the before value was obtained with old software that existed before the project and did have some room for development in the scope of efficiency and on how the software could make more use of the features of the RTOS, re-designing the software to support more RTOS brought major improvements to the consumption with a 5.94mA save which is 20% decrease in overall consumption.

There is a difference between theoretical power consumption without the display and the normal mode where everything was in active state, this can be explained with the implemented tickles low power features of the RTOS which utilize sleeps as efficiently as possible.

The difference between normal, sleep and deep sleep varies greatly, the sleep mode uses 9.31mA less compared to normal mode which is 39,18% smaller. In other words, even using a light sleep mode where all peripherals are active has a great impact on power consumption. Deep sleep on other hand is a completely different story, compared to sleep mode it consumes 8.79mA less and is 60,83% less which is a significant decrease. When comparing the deep sleep with normal mode the energy saving is on a completely different level, in deep sleep the device consumes 18.1mA less making it

being 76.18% smaller compared to normal mode. Savings in consumption impacts greatly battery life as shown in Figure 24.

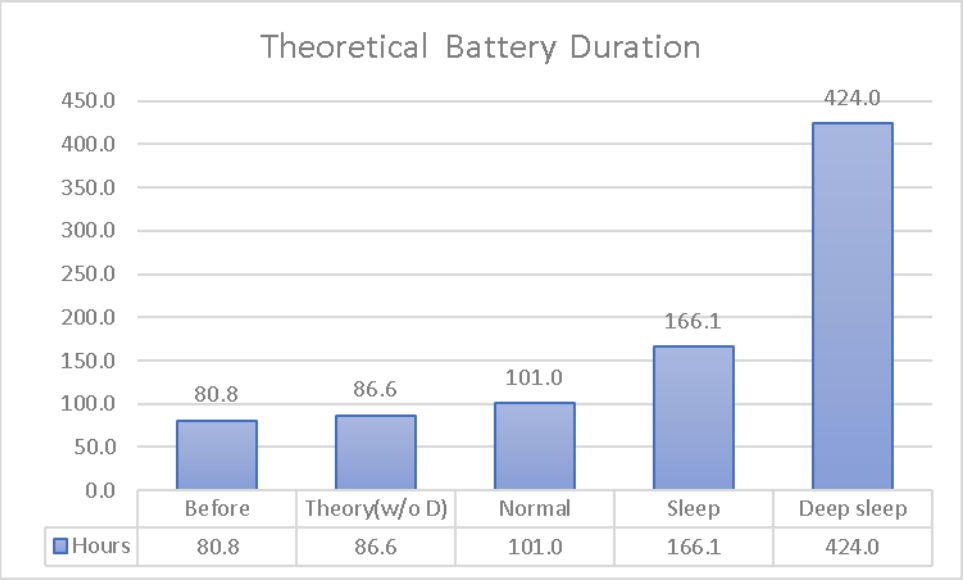

Figure 24. Theoretical battery duration on 2400 mAh battery.

Figure 24 illustrates how long the battery would theoretically last in that mode. When comparing the 20% impact between old software, before the project and optimized software the reality is that with more optimized software and making use of the features of the RTOS the theoretical battery duration increases 20.2 hours which is a huge improvement when considering that improvements were made on the software level.

When comparing the differences in low power modes it comes more obvious how large the impact is on battery duration. The comparison between normal and sleep modes shows that the continuous sleep mode would increase battery life by 65.1 hours. On the other hand, the effect of deep sleep on battery life is in its own scale, compared to sleep mode the deep sleep gives 257.9 hours more battery life and compared to normal mode the deep sleep has 323 hours more battery life.

Even though sleep and deep sleep modes have greater battery duration those modes are not active the whole time, but for the software developer measurements will give some sense which modes should be emphasized even more.

5 Discussion and Conclusions

The battery part of the project was successfully implemented and it improved the overall product, with knowing the battery level the user does not have to guess the battery state anymore and do unnecessary charging when there is still power left. Safety features will give the device more value as the features increased the overall reliability.

In the second part the software optimizations decreased the power usage greatly and as a result the software works more reliably with an RTOS. However, the low power features need more development and investigation since the low power mode of the sensor had sometimes some issues where it did not enter the low power mode or it had issues with detecting movement and the microcontroller did not get any interrupts to wake up, as a precaution the button of the device was determined as one wakeup source.

With overall development there was many issues where the software development IDE refused to co-operate and the computer needed to have restarts which had a great impact on how much time the software development took. There were some issues with the manufacturer's libraries, especially some configuration values and register pointers were incorrect but fixable with use of technical documentation, solving these problems took several development hours.

The development on the project is continued and the knowledge and results from this project will be used to greatly improve the device, some ideas for future development are investigated as to increasing the reliability of the movement detection of the sensor in the low power mode. When the sensor issues are solved the overall power saving needs more optimizations. The radio controller software is going to be studied and implemented as a part of the whole power manager which gives more control over the device.

Li-ion batteries are widely used within consumer electronics and became the most popular battery type used in the automobile industry, li-ion batteries have excellent electrochemical characteristics which make them optimal for different devices. Battery powered devices consist of multiple components of which each one has their role in the overall system. Right from the beginning it is important to take power usage and the lifetime of the device into consideration when selecting parts since components determine all of the

overall characteristics of the device. Nowadays microcontrollers and electrical components have low power features and even the manufacturing processes make it possible to create even less power hungry components.

It is possible to have a great impact on how the device consumes power, many components have built-in low power features that can be utilized programmatically. The microcontroller which is the brain of the device may use real time operating systems (RTOS) to achieve an even more robust system with its powerful features where the software development is more straightforward. One of the greatest low power features that an RTOS has, is tickles modes where when the device is not doing anything it can be put into sleep until the operating system needs to act again. This has a great impact on battery life.

References

1 Valer Pop, Henk Jan Bergveld, Dimitry Danilov, Paul P.L. Regtien, Peter H.L. Notten. Philips Research Volume 9 - Battery Management Systems – Accurate State-of-Charge Indication for Battery-Powered Applications: Springer; 2008.

2 Olof Ramström. 2019. Scientific Background on the Nobel Prize in Chemistry 2019.https://www.nobelprize.org/prizes/chemistry/2019/advanced-information/. [Accessed 1.11.2019].

3 Research and Markets. 2019. Web-document. URL:https://www.globenewswire.com/news-release/2019/10/17/1931007/0/en/Lithium-ion-Batteries-World-Market-Analysis-Forecast-Study-2019-2024-Key-Players-are-Panasonic-Tesla-Samsung-LG-Chem-and-Contemporary-Amperex-Technology.html [Accessed 1.11.2019].

4 Types of Lithium-ion. 2019. Web-document. URL: https://batteryuniversity.com/learn/article/types_of_lithium_ion [Accessed 25.10.2019].

5 Amir Zjajo, José Pineda de Gyvez. Low-Power High-Resolution Analog to Digital Converters: Springer; 2011.

6 Data Conversion Handbook: Analog Devices Inc.; 2004.

7 Understanding SAR ADCs: Their Architecture and Comparison with Other ADCs. 2001. Web-document. URL:https://pdfserv.maximintegrated.com/en/an/AN1080.pdf [Accessed 4.11.2019].

8 Mastering the FreeRTOS Real Time Kernel – A Hands-On Tutorial Guide. URL: https://www.freertos.org/wp-content/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf [Accessed 6.11.2019].

9 Dogan Ibrahim. Advanced PIC Microcontroller projects in C: Newnew; 2008.

10 FreeRTOS Tickless idle. Web-document. URL:https://www.freertos.org/low-power-tickless-rtos.html [Accessed 1.11.2019].

11 Battery structure. 2019. Web-document. URL: https://batteryuniversity.com/learn/article/a_look_at_cell_formats_and_how_to_build_a_good_battery [Accessed 25.10.2019].

12 How does Electrolyte work? Web-document. URL: https://batteryuniversity.com/learn/article/bu_307_electrolyte [Accessed 10.10.2019].

13 Omar, Noshin & Daowd, Mohamed & Van den Bossche, Peter & Hegazy, Omar & Smekens, Jelle & Coosemans, Thierry & Van Mierlo, Joeri. Rechargeable Energy Storage Systems for Plug-in Hybrid Electric Vehicles. Article; URL: https://www.researchgate.net/publication/231169804_Rechargeable_Energy_Storage_Systems_for_Plug-in_Hybrid_Electric_Vehicles-Assessment_of_Electrical_Characteristics [Accessed 10.10.2019].

14 Vidyu Challa, How To Select The Right Battery For Your Application?, Web-document. URL: https://www.dfrsolutions.com/blog/how-to-select-the-right-battery-for-your-application-part-1-battery-metric-considerations [Accessed 1.11.2019].

15 What is the Function of the Separator?, Web-document. URL: https://batteryuniversity.com/learn/article/bu_306_battery_separators [Accessed 1.11.2019].

16 Battery Cell Comparison, Web-document. URL: https://www.epectec.com/batteries/cell-comparison.html [Accessed 1.11.2019].

17 MEPs ban cadmium from power tool batteries and mercury from button cells, Press release. URL: https://www.europarl.europa.eu/news/en/press-room/20131004IPR21519/meps-ban-cadmium-from-power-tool-batteries-and-mercury-from-button-cells [Accessed 1.11.2019].

18 Safety valve of lithium ion battery. Patent. URL: https://patents.google.com/patent/CN203134886U/en [Accessed 1.11.2019].

19 Power Saving Modes for Embedded Systems, Web-document. URL: https://www.embedded-computing.com/why-you-didn-t-get-the-rtos-business-and-other-embedded-software-stories/power-saving-modes-for-embedded-systems [Accessed 1.11.2019].

20 Power management in embedded software, web-document. URL: https://www.embedded.com/power-management-in-embedded-software/ [Accessed 1.11.2019].

21 Charging Lithium-ion, web-document. Url: https://batteryuniversity.com/learn/article/charging_lithium_ion_batteries [Accessed 1.11.2019].