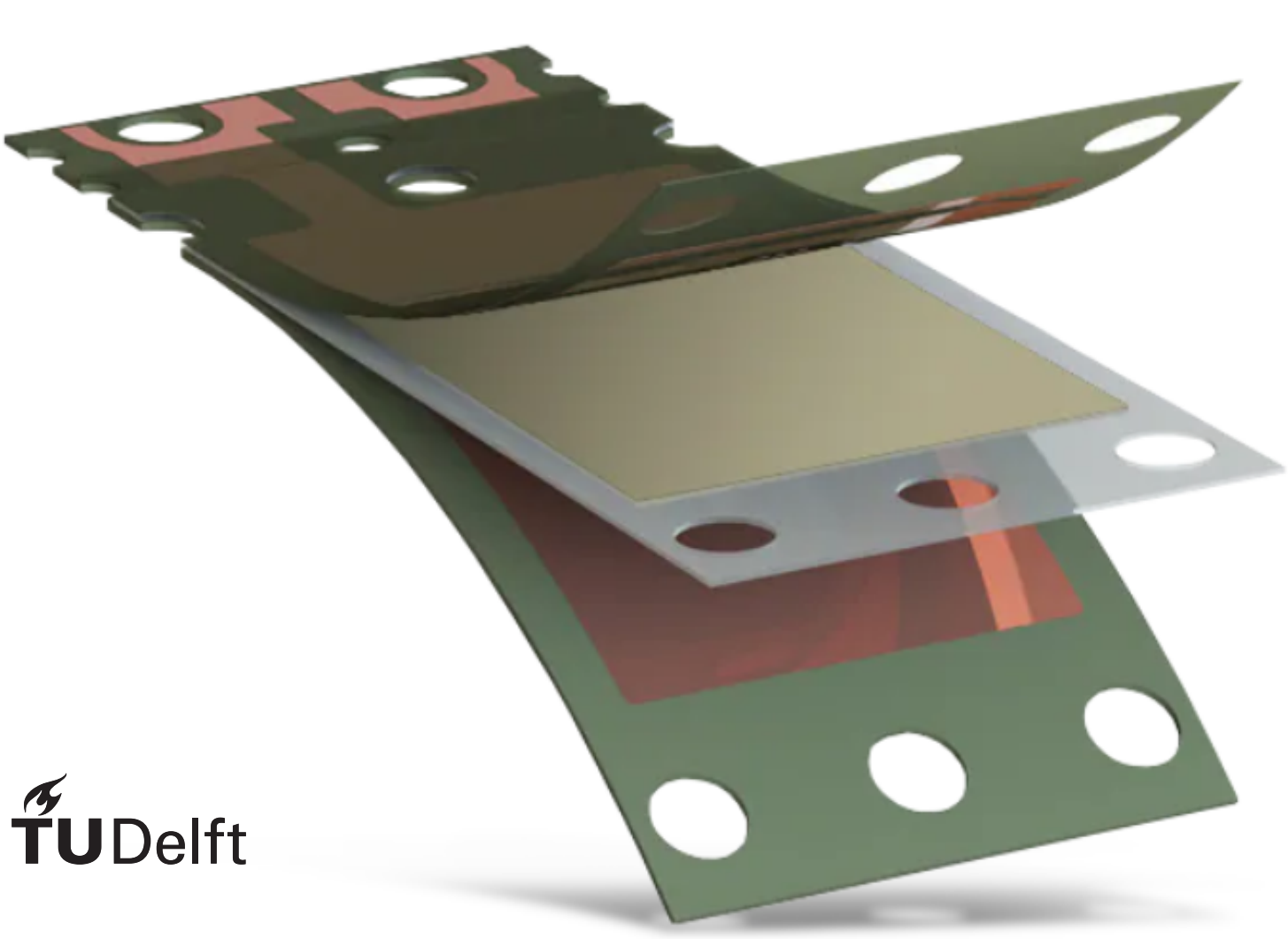
Piezoelectric Energy Harvesting System

for Low-Power IoT Sensors

Alessio Solter & Remon van Bemmel

BSc Thesis



TU Delft

Piezoelectric Energy Harvesting System

for Low-Power IoT Sensors

by

Alessio Solter & Remon van Bemmel

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Friday June 27th, 2025 at 15:30.

Student number:	Remon van Bemmel	5610370
	Alessio Solter	5413567
Thesis committee:	Dr. Sijun Du	TU Delft, supervisor
	Prof.dr.ir. A.H.M. Smets	TU Delft, jury member
	Dr. Mottaqiallah Taouil	TU Delft, jury member
Project duration:	April 21, 2025 – June 27, 2025	

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

This thesis presents the design and evaluation of a low-power wireless sensor system intended to operate in conjunction with a vibrational piezoelectric energy harvester and a power conditioning circuit inside a plane wing. The system was tested in a controlled laboratory environment under various configurations, including different transmission modes, power levels, and sampling frequencies, to assess its performance, energy efficiency, and stability.

Key findings indicate that sampling frequency significantly influences measurement stability. Continuous transmission mode, where data is sent without delay, while offering high temporal resolution, introduced greater variability in temperature readings, likely due to environmental noise and power supply fluctuations. In contrast, burst mode, which only sends 3 measurements in quick succession, achieves a better balance between resolution and stability, suggesting that incorporating intentional delays can mitigate noise sensitivity. Thermal recovery tests revealed that sampling density affects response time, with delayed mode showing slower returns to baseline, possibly due to self-heating or insufficient data resolution.

An attempt to implement a low-power sleep mode revealed that actual current consumption did not decrease as expected, indicating an incomplete power-down mode and highlighting the need for further investigation into power management. The power consumption study confirmed that oscillator frequency and transmission power significantly affect current draw, although an anomaly observed at 4 MHz in delayed mode suggests potential efficiency zones in the transceiver's operation.

Integration with the energy harvester demonstrated the system's ability to function under energy-constrained conditions. Testing with different storage capacitors showed that a 680 μF capacitor provided the best balance between data transmission quality and energy availability. These results highlight trade-offs between energy storage capacity and responsiveness, and emphasize the importance of hardware optimization in energy-autonomous sensor systems.

Preface

This thesis has been written as part of the Bachelor Graduation Project, in order to obtain the degree of Bachelor of Science for the study Electrical Engineering at the TU Delft. Our work focused on the design of a low-power wireless sensor network system to be integrated with a piezoelectric energy harvester and a power conditioning circuit, suitable for deployment inside aircraft.

We would like to express our sincere gratitude to our supervisor, Dr. Sijun Du, as well as our daily supervisors, Wenyu Peng and Zijun Qui, for their continuous support and guidance during this 10-week project. Their assistance proved to be important during our research and kept us focused on the critical elements.

We would also like to express our thanks to our fellow group members Laurens-Jan Cammeraat, Wester Landman, Ivo Sokal and Vincent Verkoren for their contributions and collaborations, which made the successful development of a functioning prototype possible.

Alessio Solter & Remon van Bemmel
Delft, June 2025

Contents

1 Introduction

1.1. Internet of Things

The advancement of technology has led to an increase of the usage of so-called 'smart' devices. Smart devices are part of a concept known as Internet of Things (IoT), which involves equipping electronic devices with sensors, software and network connectivity [1]. This allows these devices to communicate data wirelessly, giving them more utility and simplifying the installation process. When removing the need for data cables, a greater flexibility is offered when designing larger systems. On top of that, routing and installing of these data cables is a costly process, especially when working in remote or hard-to-reach locations.

According to [2], IoT devices can be divided into five different types. These are:

1. Portable devices (smartwatch, wireless headphones)
2. Set-and-forget devices (home security, smart thermostat)
3. Semi-permanent devices (structural health monitoring in bridges or tunnels)
4. Battery-less (RFID, smart tags)
5. Powered appliances (smart fridge, smart lighting)

While Internet of Things provides a lot of opportunities, it also introduces some new challenges. Despite the removal of data cables due to IoT connections, devices still need a supply of power. Power could be supplied by an electrical outlet, as is the case for Type 5 implementations, such as smart lighting systems, where only the control of the lighting is done wirelessly. While this works for some applications, in most cases it is not preferable, as it removes the advantages that IoT provides by adding cables to an otherwise wireless device. For this reason, most IoT devices make use of a compact battery in order to store energy. Batteries do however have some disadvantages, such as limited lifetime, safety hazards and high maintenance costs when replacement is required [3].

A second challenge that comes with IoT is the need for stable, low-power data transmission. This is important for all types of IoT devices, but especially for type 2 and 3, as they have to function for multiple years without having to replace the battery. Popular options when it comes to wireless communication are WiFi, Bluetooth or cellular networks, but these consume too much power in normal operation conditions. For low-power operation, specialized protocols are available [4], such as Bluetooth Low Energy (BLE), Zigbee or Z-Wave for short range communication. For longer ranges, Low-Power Wide Area Network (LPWAN) protocols are used.

1.2. Piezoelectric Energy Harvesting

A novel solution which could be used to power these IoT devices, is making use of self-sufficient systems such as piezoelectric energy harvesters (PEHs). These systems collect mechanical energy from vibrations and convert it to electrical energy due to the piezoelectric effect; when mechanical stress is applied to certain solid materials, such as specific crystals, electric charge will build up within these solids [5, 6]. The energy density for mechanical vibration PEHs is relatively high (0.3 $\mu W/mm^3$), which makes it perfect for IoT sensor cases that usually have a power range from several μW to 1 mW [7].

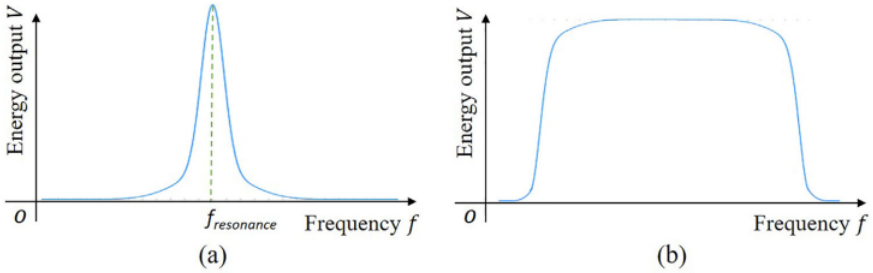


Figure 1.1: Energy output depending on frequency for a piezoelectric energy harvester, (a) Resonance based PEH, (b) Ideal PEH [7].

Mechanical vibration PEHs have an resonance based operation. The material properties and dimensions of the harvester determine a dominant resonance frequency. Energy harvesting is most efficient when the frequency of the energy source closely matches the harvester's resonance frequency, meaning that even a slight deviation can drastically reduce energy output, as illustrated in Fig. 1.1(a) [7, 8]. These types of harvesters are also referred to as mono-stable structures since they have a singular stable point, but practical applications usually call for a harvester with a wider frequency bandwidth, as can be seen in Fig. 1.1(b). Real world vibrations are variable and can change in frequency and amplitude over time, thus having an energy harvester with a wide frequency band leads to a higher power generation [9] and improved reliability and efficiency [10].

By harvesting energy from the environment, PEHs give rise to battery-free or self-powered systems, reducing the amount of maintenance required, while extending device lifetime. These systems have gained a lot of attention in the field of low energy electronics as a sustainable solution, particularly for wearable devices [11], wireless sensors [12], monitoring the structural health of bridges [13] and even in the medical field to power implantable devices [14, 15]. On the other hand, their energy output is relatively small, limiting the current possibilities [7]. Ongoing advancements in the material science and the efficiency continue to broaden the range of practical uses for this sustainable solution.

1.3. Problem Definition

Piezoelectric energy harvesting presents a promising solution for powering IoT devices in environments where mechanical vibrations are abundant. In order to implement such a system, two key criteria must be met: the IoT device is not allowed to consume more power than the PEH can produce and the IoT system must be designed with ultra-low power consumption in mind.

When looking at these criteria, low-power IoT sensors provide a great application, since they usually function within the power limitations set by the PEH and are critical for virtually all IoT systems. Ensuring that such a system functions reliably with the limited available power, requires both the hardware and communication method to be optimized for low power consumption.

When designing a PEH-powered IoT sensor, the project can be divided into three main components. These are the piezoelectric transducer, which converts mechanical vibrations into electric power; the power conditioning circuit, which processes the transduced energy into a stable DC power output;

and finally the IoT sensor integration, managing the data acquisition and wireless communication. This thesis focuses on the third module: Developing an energy efficient, reliable and functional IoT sensor system, being able to operate within the power limits set by the piezoelectric energy harvester.

In this thesis, such a system is designed and evaluated for employment within aircraft wings. Here, the system is used to monitor peripherals in hard-to-reach locations, with the goal of simplifying installation and maintenance, reducing weight by removing large sections of wiring and making the design process easier for airplane manufacturers.

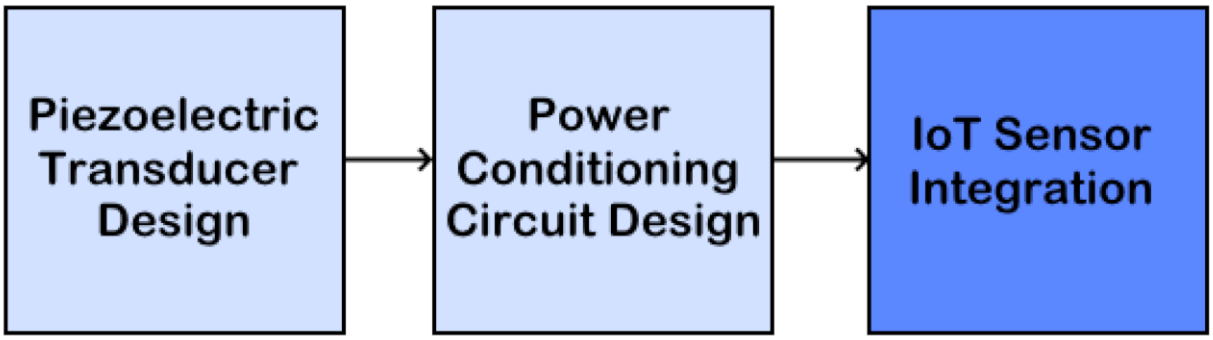


Figure 1.2: Subgroup division for the project

1.4. State of the Art

Research regarding PEH's has almost increased exponentially during the last decades, as can be seen in Fig. 1.3, giving improved insight into circuit design, materials, different energy sources and various applications [16]. The application that has seen increased importance in a wide spectrum of fields is Wireless sensor networks (WSN). These networks are comprised of four general parts: a power supply unit, a set of sensors, a processing unit (usually a microcontroller) and a wireless communication unit [17]. In the case when the power supply is an energy harvester, the system evolves into a self-sustaining Energy harvesting wireless sensor network (EHWSN) which has increased lifetime [18].

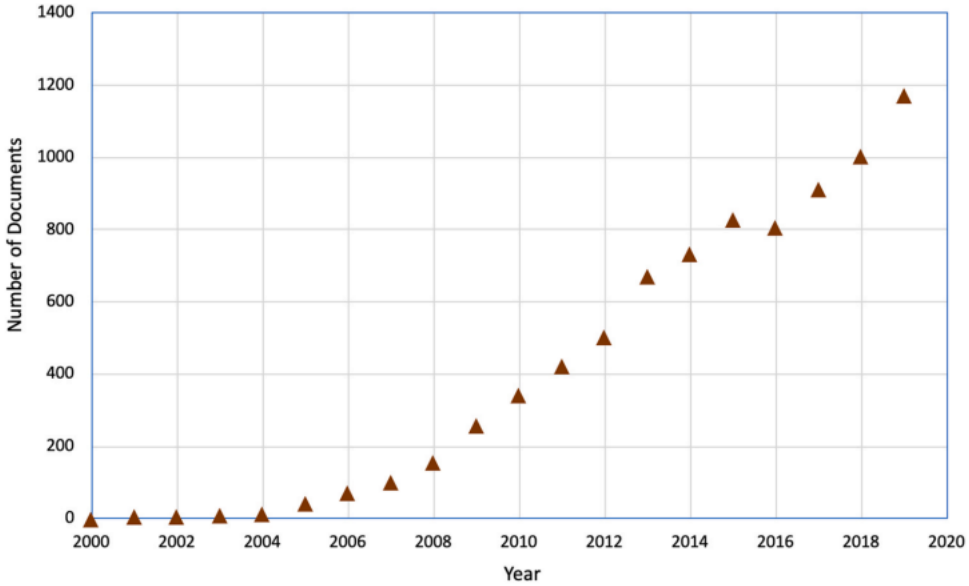


Figure 1.3: Amount of papers published regarding "piezoelectric energy harvesting" [16]

In order to make the system actually autonomous, ultra-low power techniques are required aimed to make the power consumption as low as possible. One way to achieve this is by duty cycling the system. This is a technique to switch the WSN on and off periodically to reduce power consumption. The duty cycle then is the ratio of the time that the module is active divided by the total time; the sum of the time it is active and asleep [19]. By implementing an instance of ”aggressive duty cycling”, the module can operate in an ultra low power mode such that it is active for 1% of the time, and is idle/asleep for the remaining part. On the other hand, there are downsides to this implementation. The amount of active time must be long enough for the WSN to operate correctly, when the duty cycle is too low, it may either block the sent signal or send an incomplete signal. In addition, the idle state for most WSN systems still consumes a lot of energy. In this case it will be better to put the module into a (deep) sleep mode or turn it off entirely while it is inactive. Booting up the system will then take some startup time, which also uses a significant amount of energy, but it is not as much as staying in idle mode.

However, energy consumption is mainly dominated by the wireless communication and is dependent on the number of bits to be transmitted. Computations require significantly less energy in contrast to transmissions, where for example a 1-bit transmission uses 500-1000 times more energy compared to a single 32-bit computation [20] Compressing the data to a smaller bit size can thus reduce the energy consumption of a WSN system. This can be done using ”compressed sensing” [21], which uses a sub-Nyquist sampling rate to guarantee a high accuracy for the recovered signal. Whereas energy is saved with this implementation, the data transmission latency increases due to a lower sampling rate. This is especially visible in applications that require real time monitoring, but does not show significant downsides in applications without, like in the case of a piezoelectric harvester that needs time to charge in order to output a steady voltage for a short period. Another solution would be to compress the data at the sensor nodes and decode them at the ”base station”, as described in [22]. Despite increasing the amount of computational energy, it will remain insignificant with respect to the transmission energy.

[23] proposes a solution that uses Dynamic voltage-frequency scaling (DVFS) to minimize the average power consumption in a WSN system by adjusting the voltage and/or frequency depending on the workload. The system will not consume any unnecessary energy due to running on maximum voltage and frequency. Unfortunately, this solution will not work for this paper, since the main goal is to harvest as much energy as possible from the PEH and use it on a specific frequency to power the sensor system.

Besides low power operations, having a low power sensor is also of the essence. [24] shows an overview of the most known and used low-power and low-cost sensors available, as well as their current state-of-the-art performance.

1.5. Structure of thesis

The outline of this thesis is as follows:
Chapter 2 describes the functional, system and trade-off requirements for the sensor and wireless communication. Chapter 3 then explores the available low-power sensor types and communication methods. Chapter 4 focuses on the design and implementation of the system, based on the information and requirements provided in the previous chapters. Chapter 5 presents the results of the measurements, which are subsequently discussed in Chapter 6. Finally, Chapter 7 provides the conclusion and offers recommendations for future work on this topic.

2

Programme of Requirements

The goal of this thesis is designing and evaluating an Internet of Things sensor system for integration with a piezoelectric energy harvester (PEH). The system will be designed for employment within aircraft wings, taking sensor data in hard-to-reach areas. The PoR contains an overview of the functional, system and trade-off requirements for the project, covering the functionality, environmental constraints and the electrical characteristics the system should adhere to. In chapter 4, Design and Implementation, these requirements are taken into account in order to make the correct design choices.

2.1. Functional Requirements

The functional requirements of the system describe what the system must be able to do.

- The system must be compatible with the piezoelectric energy harvester.
- The system must be able to communicate wirelessly with a central computer.
- The system must be able to take periodic measurements from its surroundings.
- The system should be compatible with different types of sensors.
- The average power consumption of the system must not be higher than the power supplied by the piezoelectric energy harvester.
- The system should be robust and reliable, being able to withstand the environmental vibrations.

2.2. System Requirements

These are the requirements about the system specifications, such as performance, reliability and efficiency.

- The communication module should be able to transmit data over a distance of at least 10 meters.
- The system must be able to transmit sensor data at least once every 60 seconds.
- The system must operate within the aircraft temperature range of -40°C to +85°C.
- The system must be able to function on a input voltage of 3.3V.

2.3. Trade-off requirements

These are the requirements that the system preferably complies with as much as possible.

- The weight, cost and power consumption of the system should be kept to a minimum, while still adhering to the other requirements.
- The system should take measurements as often as is possible while also keeping the sensor data accurate.

3

Wireless Sensing

In the previous chapters, the premise of using piezoelectricity for wireless sensing was established. Since piezoelectric energy harvesting has a very low energy output, the main challenge for the sensor system is making it incredibly low-power. In this chapter several low-power sensor types and communication methods are explored. In the subsequent chapter, these options are then compared in order to find the best fit for the purposes of this thesis.

3.1. Sensors

There are many different kinds of sensors, each with their specific use cases, ranging from motion detection to environmental monitoring and biomedical applications. In order to determine which sensor types are most suitable for IoT sensor integration, especially when powered by a piezoelectric energy harvester, it is essential to first establish a comprehensive overview of the available sensor technologies and their respective power consumption.

In a paper on sensor technologies [24], Gazis et al. offer an extensive overview of eleven different sensor types, covering categories such as temperature, humidity, pressure, acceleration and optical sensors. In addition, they attribute each of these sensor types a power consumption label, indicating whether the sensor typically has a low, moderate or high power consumption.

Considering the constraints that arise when using piezoelectric energy, only sensor applications with extremely low average power consumption are viable. This either implies that the sensor consumes very little power during active operation, or the sensors active periods are extremely short and infrequent, separated by an extended dormant interval, during which the PEH can accumulate energy. With these restrictions, many modern sensor types are quickly excluded from practical use in such systems.

An example of an incompatible sensor class would be the optical speed and distance sensors. They typically rely on active light sources, such as lasers or infrared LEDs, and operate at high sampling rates, ensuring precise and continuous measurements. This results in a demand for sustained power input, making them unsuitable for applications that rely on low-power energy sources like PEHs.

However, this limitation does not apply for all sensor categories. There are many IoT applications where continuous data transmissions are not necessary and periodic sampling is sufficient. Environmental monitoring would be a key example. Sensors in weather stations typically collect data on temperature, humidity and barometric pressure. According to the Bureau of Meteorology [25], these stations have a broadcast frequency ranging from once every 10 minutes to once an hour. In such cases, if sensors are configured to stay in a low-power sleep mode in the inactive period, the average energy consumption can be kept extremely low.

Furthermore, advancements in low-power sensor systems, including duty cycling, power gating [26] and localized computing, have led the way to even more efficient energy management. Modern sensors often come with built-in microcontrollers that handle local signal processing, reducing the need for frequent wireless communication, thus decreasing the power requirements.

In addition to environmental sensing, other promising low-power sensor types for piezoelectric-powered IoT systems include:

- Low-frequency accelerometers
- PIR (Passive Infra-Red) motion sensors [27]
- Vibration sensors (piezo or MEMS)
- Light intensity sensors

These sensors function effectively with low duty cycles and low energy consumption, allowing for compatibility with the irregular, limited output of piezoelectric energy harvesters. Successful integration can be achieved with careful sensor selection and optimized system architecture, prioritizing efficient data acquisition and minimal transmission time.

3.2. Wireless communication

While selecting appropriate low-power sensors is important for a PEH-powered IoT system, wireless communication often accounts for the majority of the energy consumed by such a system. For this reason, choosing the right communication protocol is equally as valuable to ensure that the complete system remains within the energy constraints provided by the harvester.

A variety of wireless communication methods are available for IoT design, each having their own up- and downsides with respect to power consumption, data rate and range. For PEH powered sensors, the focus must be on ultra-low power and data rate transmission, allowing the system to transmit small bursts of data when enough energy has been harvested. The most relevant communication technologies for this purpose include:

BLE (Bluetooth Low Energy)
Bluetooth Low Energy (BLE) is a version of the standard Bluetooth protocol optimized for low power consumption, designed specifically for shorter range and low data-rate communication. BLE operates in the 2.4 GHz frequency band and supports connection oriented and advertising-based transmission, making it a good option for low-latency sensor transmission [28].

BLE is widely used in wearable devices, fitness trackers, smart home systems and mobile biomedical applications, due to its low energy requirements and ease of implementation with smartphones. It typically has a sleep current around 1-10 μA, with active transmitter/receiver current ranging from 5-30 mA, depending on the chip and output power [29]. With aggressive duty cycling, the average current consumption can drop below 100 μA, making it viable for integration with PEHs.

LoRaWAN (Long-Range Wide Area Network)
LoRaWAN is a network protocol that operates on the LoRa physical layer. It is specifically designed for long-range, low-power and low data-rate communication. LoRa operates in the 433-, 868- or 915-MHz frequency bands [30], while using Chirp Spread Spectrum modulation (CSS), allowing for long-range communication of up to 15 km in rural areas.

LoRaWAN is commonly used in smart cities, agriculture, healthcare and other remote sensing applications [31]. Devices spend most of their time in sleep mode, only waking up briefly to transmit data, often at intervals ranging from a couple of minutes to several hours. Transmit currents are typically 20-40 mA, but with sleep currents going all the way down to 1 μA, the average current can lie around 10 μA [32]. With long sleep durations and infrequent transmissions, LoRaWAN becomes a suitable option for PEH-powered applications.

Zigbee
Zigbee is a low-power mesh networking protocol operating in the 2.4 GHz band, developed for short-range communication in home automation, industrial monitoring and automotive monitoring [33]. It supports robust mesh networking, allowing data to be exchanged between nodes in order to extend the range and reliability.

In standard application, Zigbee consumes 10-20 mA, with only a few micro amps in sleep mode. The mesh capabilities can increase communication overhead, which might not be ideal for energy harvesting nodes that transmit infrequently. However, Zigbee is a good practical option in situations where mesh coverage is preferred and nodes can remain mostly dormant.

Wi-Fi HaLow [34]
Wi-Fi HaLow is a low-power, long-range Wi-Fi standard that operates in sub-1 GHz bands. It is designed for IoT applications, offering better range and lower energy requirements than the standard Wi-Fi. Wi-Fi HaLow can support tens of megabits per second with a range of about a kilometer, capable of supporting thousands of devices.

Typical use cases include industrial IoT, smart agriculture and connected infrastructure. Power consumption is higher than BLE or Zigbee, with transmission currents well above 30 mA, but its extended range and IP-based networking capabilities make it suitable for systems where slightly more energy is available. With limited data transmission and careful timing it may be compatible in a PEH-powered system.

Proprietary 2.4 GHz protocols
Many low-power transceivers, such as the nRF24L01 [35], use proprietary 2.4 GHz protocols based on Gaussian Frequency Shift Keying (GFSK) modulation. These protocols are not part of any standardized communication protocol such as BLE or Zigbee, but instead allow full control over the packet structure, transmission timing and addressing. This makes them very interesting options when working with highly constrained systems like the ones powered by PEH.

The nRF24L01, for example, can support data rates of 250 kbps to 2 Mbps and consumes 7-13 mA during transmission, with a sleep current below 1 μA. Its Enhanced Shockburst protocol provides automatic acknowledgments and retransmissions, helping to ensure reliable communication with minimal additional code. These proprietary protocols are widely used in DIY project, wireless peripherals and commercial systems such as toys or industrial sensors.

4

Design and Implementation

When attempting to design an Internet of Things sensor system, it is important to identify the different modules such a system is composed of. As determined in 1.4, four main modules can be distinguished. These being the power supply, microcontroller, sensor and wireless communication module.

Power Requirements
The IoT sensor is powered by the piezoelectric energy harvesting system. How this harvester and its corresponding regulation circuit functions and how it has been designed, tuned and tested, can be found in the theses of the other two subgroups. While this thesis doesn't cover the intricacies of the harvester, the design process of the IoT sensor does require some knowledge about the PEH. In order to make the correct design choices, it is important to know a couple of the characteristics of the PEH. First of all, the harvester supplies a voltage of approximately 3.3V and is able to deliver 400 μW sustained to a load of 27 kΩ at a frequency of 60 Hz. This leads to the first restriction for the system: The average power consumption of the circuit cannot be higher than the power supplied by the PEH. Additionally, the harvesting circuit can deliver a current of 5 mA for approximately 300 ms to a load of 500 Ω. This is with a capacitor of 680 μF.

With these power requirements set, it is time to proceed with the design of the IoT sensor system. As indicated before, this system consists of a microcontroller, sensor and wireless communication module. Determining which component should be used is a tricky endeavor, but with the requirements set for the system, an educated decision can be made on the different microcontrollers, sensors and communication modules available.

Sensor Consideration
In section 3.1, an extensive overview of available sensor types is provided, especially covering those that are promising for integration in a PEH-powered system. Since the sensor is implemented in the airplane environment, preferably those located in the harder-to-reach areas, a sensor fitting this application should be chosen. Of the many sensors available in aircraft, these are some examples [36]:

- Environmental sensing (air pressure, humidity, quality and temperature)
- Structural Health Monitoring (stress detection, fatigue, vibration sensing)
- Navigation and Control sensors(GPS, IMUs, speed and altitude)
- Engine and Propulsion (temperature, pressure, vibrations)

These sensors are of great importance for safe and comfortable operation of the aircraft, but not all of them are viable options for integration in a PEH-powered system. Sensors required for navigation and control often have high power draw or are part of quick control loops, requiring high sampling speeds, making them unfit for PEH-powering. While structural health monitoring (SHM) would be a

viable option in terms of power consumption, testing the sensor system could be difficult within the time span of this thesis. With SHM, a large test setup is required where strain and fatigue of a structural component such as a beam can be tested. Because this thesis wants to prove the feasibility of using PEH for wireless sensing, SHM is not a viable option within the time available.

Sensor types that are viable for use with PEH are the environmental sensors, especially temperature, pressure and humidity. These sensors usually have low average power consumptions, can be tested and verified pretty easily, and require lower sampling speeds. With sensors, there are two categories: analog and digital sensors. Analog sensors output a voltage correlated with the peripheral being measured. With an analog-to-digital converter (ADC), this voltage can then be transformed into a digital signal. Digital sensors have an internal ADC, directly outputting a digital signal. To see which option would be better for the system, both are tested. For the analog sensor, the STLM20W87F temperature sensor has been chosen. The datasheet for this sensor claims that the sensor has a ultra-low quiescent current of 4-8 μA, which is a great option for PEH-powering. The digital sensor is the 3-in-1 BME280 sensor. This sensor is able to measure temperature, humidity and barometric pressure, while still having very low active power of 3.6 μA.

Wireless Communication Consideration
Section 3.2 covered a multitude of available wireless communication protocols, fitting for low-power IoT sensor design. Within the context of an aircraft, the sensor data does not have to travel further then 50 meters, typically only 20. Because of the low range requirement, communication methods such as LoRaWAN and Wi-Fi HaLow would not be a good fit, as they are made for ranges far greater than that. They would consume unnecessary amounts of power that could otherwise be used for increasing the data rate. Similarly, Zigbee has mesh networking capabilities that are not necessary in the context of this thesis. Because of the mesh networking, Zigbee typically consumes slightly more power than BLE or proprietary 2.4GHz protocols do.

In the end, the nRF24L01 module has been selected for its:

- Low power consumption, especially with its available standby and power-down modes.
- Adequate range for many aircraft monitoring applications (10-100 meters)
- 2.4 GHz operation, allowing global unlicensed use
- Simplicity of implementation

The nRF24L01 operates using a proprietary 2.4 GHz protocol designed by Nordic Semiconductor, that trades the complexity and interconnection abilities of standardized protocols for greater energy efficiency and control over the communication. This fits the goal of developing a PEH-powered wireless sensor. While other protocols such as BLE or Zigbee may be considered in future iterations where additional power management or integration requirements exist, or where mesh networking could provide potential benefits, the nRF24L01 provides optimal balance between ease of implementation, power efficiency and sufficient communication performance for this thesis project.

Microcontroller Consideration
When choosing the microcontroller there are several requirements. First, it needs to be low-power, just like the other components. It also has to have SPI serial communication capabilities, since the nRF24L01 uses this method to communicate with its microcontroller. In order to read out the analog sensor, the microcontroller is required to have a pin that can function as an ADC. The BME280 can communicate via both I2C and SPI, so either of these communication methods would be viable. Standard accessible microcontroller development boards such as the Arduino Uno are great for a system such as this, except for the fact that it consumes a lot of power due to the many features it carries. Several stand-alone microcontroller were considered, including the STM32 series and 8-bit AVR microcontrollers like the ATmega328P. While many of these offer great performance and features, they often come with higher active currents or more complex development environments, increasing the overall power consumption and design complexity.

Ultimately, the ATtiny85 [1]was selected, due to its ultra-low power consumption, being only 100 nA in power-down mode and 300 μA in active mode. It has sufficient processing capabilities for basic sensor handling and SPI-based communication. It contains an internal 8 MHz oscillator and is well documented online [37].

While more powerful alternatives are available, they often exceed the power limitations of the PEH-powered system. The ATtiny85 has a great balance between performance and power efficiency, providing just enough functionality for periodic sampling and wireless transmission via the SPI connected nRF24L01 module.

[1]This is ultimately not the chosen microcontroller for the final design 4

Sensor testing

The sensors will be initially tested without the microcontroller and thus connected directly to the Arduino UNO. Then it will be transmitted using the transceiver to the receiver side, where the sensor is read out from the serial monitor, this way the wireless communication between both transceivers can also be assessed. The setup for this test can be found below in Figure 4.1.

Digital

The digital sensor, BME280, is connected to the Arduino Uno and the measurements are read out inside the serial monitor, showing the temperature, pressure and humidity. This connection is established using the specific I2C pins on the Arduino Uno, namely SDA and SCL (pins A4 and A5 respectively).

Analog

For the rest of this thesis, the STLM20W87F analog sensor will be used instead of the BME280 digital sensor. During the design process was found that is was not possible to use the digital sensor with the current micro controllers. In order to setup the I2C communication, it was required to use the ”Wire” library in Arduino IDE. Yet this library was too big for the ATtiny85 as well as the Atmega8. There does exist a smaller version of this library, tinyWire. Unfortunatly, both the tinyWire and BME280 libraries proved to contain too much complications, which caused to system to be dysfunctional. Thus leading to the implementation of the analog sensor, which is more straightforward to program and takes up less Flash memory.

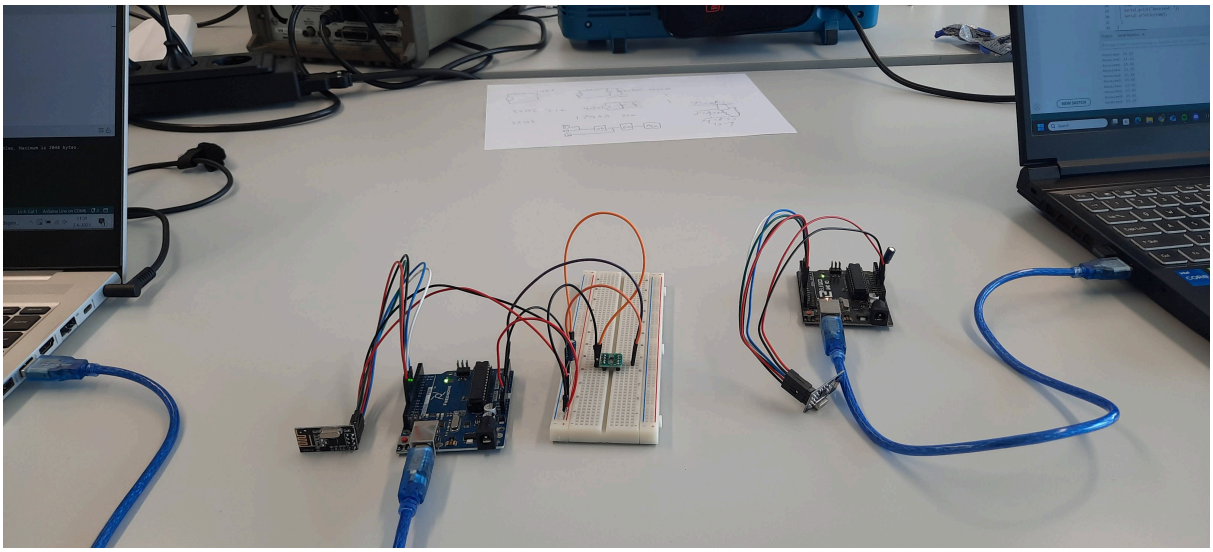


Figure 4.1: Setup for the first communication test using Arduino Uno and analog sensor

The conversion from the output of the analog sensor (STLM20W87F [38]) is done as follows: The sensor outputs a value between 0 and 1024, this value is then converted to an output voltage using Equation 4.1. This conversion depends on the voltage across the system, in this case 5.0 Volts, and uses a small calibration factor to account for inconsistencies between the V_{out} and the voltage that arrives at the microcontroller. How this value is determined is explained in the next chapter 5. Afterwards, the output voltage is used in Equation 4.2 to obtain the temperature. Since the temperature of this sensor is inversely proportional to the output voltage, an increase in temperature goes hand in hand with a decrease in output voltage [39]. This value is finally printed in the serial monitor in the Arduino IDE application, an example can be found in Appendix B, Transmission data [9].

$$V_{out}\ [\text{V}] = 1.04 * (\frac{3.3 * \text{Output value}}{1024}) + 0.02 \tag{4.1}$$

$$\text{Temperature } [°\text{C}] = \frac{1000 * V_{out} - 1866.3}{-11.69} \tag{4.2}$$

Transmission modes

ATtiny85 problems

After making sure the sensor is functional, the microcontroller is connected to the system and initialized as the ”brain” of the system. Before this, the ATtiny85 has been tested to function with some simple code, making an LED blink, but when attempting to configure the ATtiny85 to communicate with the nRF24L01 transceiver, the microcontroller did not initiate contact with the transceiver. After this problem occurred, the question arose whether it was the microcontroller or the transceiver that was malfunctioning. With an Arduino Uno acting as microcontroller, it could be determined that the transceiver functions as intended, so the problem lies with the ATtiny85. Because of the limited time available, it was decided to make a switch to a different microcontroller, the ATmega8-16PU [39], which did initiate contact with the transceiver. All tests from this point on are done with this microcontroller

System evaluation

With the full system connected, it can now be evaluated under three distinct transmission modes: a continuous mode, a delayed mode with a 1-second interval between measurements and a burst mode that sends 3 measurements consecutively before waiting for 1 second. First a baseline for the ambient temperature is created at the start of the measurement. Afterwards the response time of the analog sensor is evaluated by bringing an external heat source in the proximity of the sensor, leading to a change of temperature. This source will stay close for some time to allow the sensor to reach a new thermal equilibrium. Finally, the recovery time for the sensor to return back to the baseline is analyzed by removing the heat source. An overview of the system can be seen below in Figure 4.2 and has been worked out in more detailed schematics. These can be found in Appendix C, system schematics [10].

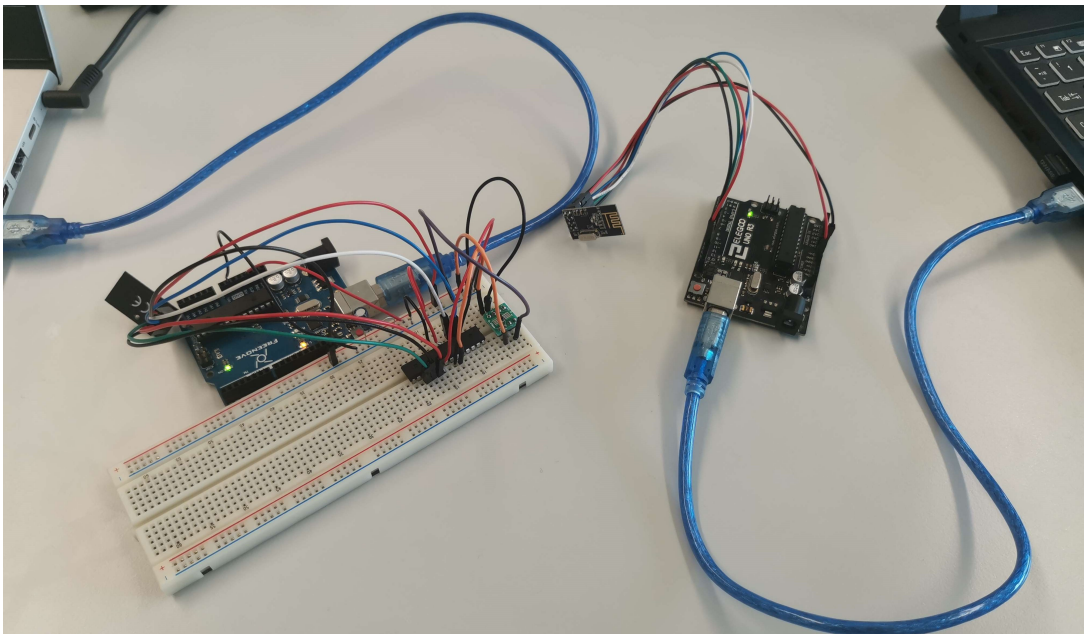


Figure 4.2: Setup for the second communication test with an on-board microcontroller reading the value from the analog sensor

Sleep mode

Next, the feasibility of implementing the power-down mode on the microcontroller is evaluated. In this power-down mode, the microcontroller should consume significantly less power, allowing the system to stay on after a measurement is performed, instead of depleting the complete capacitor. This will be done using the same setup as for the transmission modes, but with a modified version of the code designed to integrate sleep mode, which can be found in Appendix B, listing [8.6]. The main goal is to allow the system to be asleep for approximately 30 seconds, waking it up when a measurement is required. This approach will drastically reduce the power consumption compared to the previous case where it is continuously active.

Power consumption

Before measuring the current running through the system in active mode, the theoretical power consumption is calculated at a voltage level of 3.3V. This is done for varying amounts of time between measurements. The necessary values to calculate the power consumption are directly taken from each respective data sheet and shown below in Table 4.1.

Component	Voltage range (V)	Active current (μA)	Sleep current (μA)	Active time (ms)
BME280 [40]	1.71 - 3.6	3.6	0.1	13
STLM20W87F [38]	2.4 - 5.5	8.0	0.02	0.5
ATtiny85 [37]	2.7 - 5.5	300	0.1	13.5
ATmega8-16pu [39]	2.7 - 5.5	3600	0.5	13.5
Nrf24L01 [35]	1.9 - 3.6	120 - 11300 [2]	0.9	0.470

Table 4.1: Components used during the testing process

With the use of the following Equations 4.3 4.4 4.5 4.6, the average current consumption and later the total power consumption can be obtained. These have been calculated for both use with the ATtiny85 and the ATmega8-16PU. While the ATtiny85 did not work in our tests, it is still interesting to see what the power consumption would have been, as people online have had the system working [41]. The results of these calculations can be found in Table 4.2.

$$Q_{component} = I_{active} * t_{active} + I_{PowerDown} * t_{PowerDown} \tag{4.3}$$

$$Q_{total} = \sum_{i=1}^{n} Q_{component,i} \tag{4.4}$$

$$I_{avg} = \frac{Q_{total}}{t_{total}} \tag{4.5}$$

$$P_{avg} = I_{avg} * V \tag{4.6}$$

Mode	Analog		Digital	
Microcontroller	ATtiny85	ATmega8	ATtiny85	ATmega8
P_{avg} (μW, t_{total} = 1s)	16.93~34.25	165.23~182.56	17.33~34.65	165.63~182.95
P_{avg} (μW, t_{total} = 10s)	4.72~6.46	20.74~22.48	4.99~6.73	21.02~22.75
P_{avg} (μW, t_{total} = 60s)	3.59~3.88	7.36~7.65	3.86~4.15	7.63~7.92
P_{avg} (μW, t_{total} = 600s)	3.39~3.42	4.95~4.98	3.65~3.68	5.22~5.25

Table 4.2: Average power consumption for a singular measurement during varying periods

The system is tested in order to validate these calculations by connecting it to a function generator. This way it is possible to measure the current running through the system and thus measure the

[2]Depending on the output power of the transmission, range is shown between min and max value [35]

power consumption of the system. An overview of the connection between the system and the function generator can be seen in Figures 4.3 and 4.4.

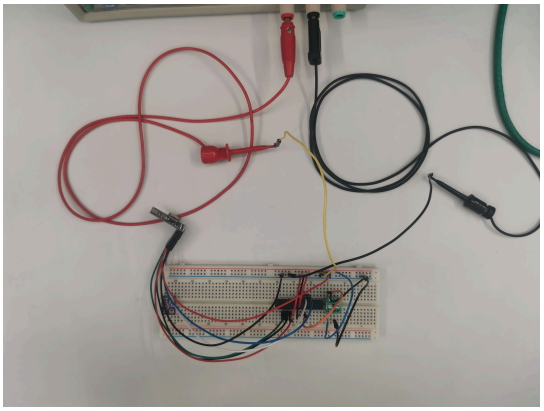


Figure 4.3: Connection with function generator, seen from the top

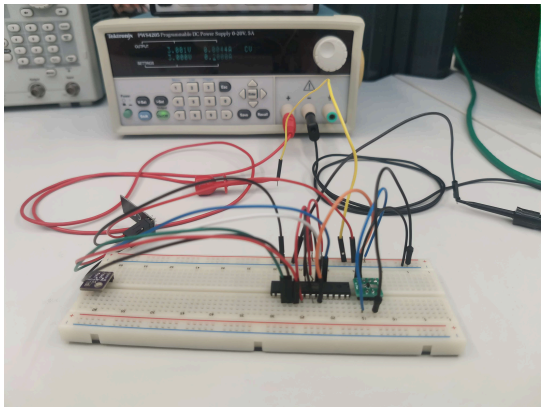


Figure 4.4: Connection with function generator, seen from the side

Total system measurements

Finally, the wireless sensor system is connected to the piezoelectric energy harvester and the power conditioning circuit. This configuration is tested with a variety of storage capacitors to determine the most optimal one in terms of measurements per minute, as well as the fraction of useful measurements per second. The size of the capacitors determines the amount of time require to charge as well as the time it is able to supply a stable voltage. A large capacitor takes longer to fully charge, yet is able to provide a stable output for longer periods of time, while a small capacitor can repeat measurements more often for shorter periods of time. An overview of the system setup can be found in Figure 4.5.

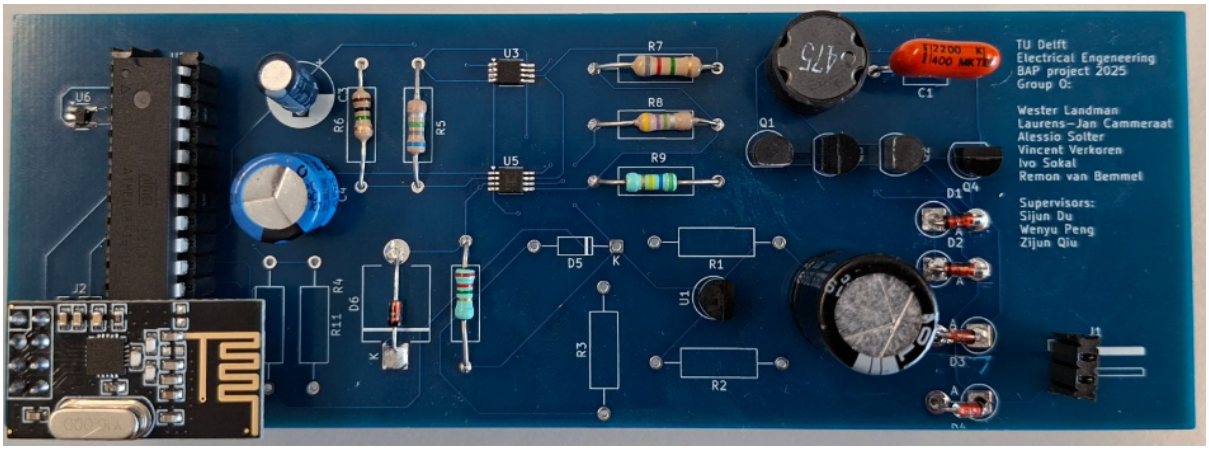


Figure 4.5: Overview of the entire system integrated on a PCB

The system is designed with a supply voltage of 3.3V in mind. However, the results of the temperature measurements were a little off when using this value as the supply voltage in Formula 4.1. It appears that the PEH actually supplies a voltage which is slightly higher than this 3.3V. Because of that, the 3.3V in this formula is changed to 3.4V for measurements done with the complete system.

5

Results

This chapter explains the tests done to verify the designs and implementation methods described in the last chapter 4. The function of the system is tested using various transmission modes to determine the best functioning operations, as well as tested when connected as output to the piezoelectric transducer and power conditioning circuit.

Sensor testing

First, the analog sensor was tested using the setup shown in Figure 4.1. The output voltage of the sensor was measured using an oscilloscope, with a probe connected to the output of the sensor, and compared with the V_{out} values shown in the serial monitor of the Arduino IDE at the receiver side. These measurements can be found in the table below, Table 5.1. As can be seen clearly from comparing the values, there exists a 0.02V difference between the measurements. This means that there is are some losses during the transmission of the values through the transceivers, which led to the decision to add a small compensation factor to the equation for the output voltage, namely the 0.02V difference which is placed at the end of Equation 4.1.

PC Terminal (V)	Oscilloscope (V)
1.40	1.42
1.51	1.53
1.52	1.54
1.54	1.56
1.56	1.58
1.57	1.59
1.61	1.63

Table 5.1: Measured output voltage of the analog sensor from the serial monitor compared with measurements with an oscilloscope

Next, the digital sensor was tested using almost the same setup as for the analog sensor, but using I2C for the serial communication with the Arduino Uno. The measurements for this setup can be found in Figures 5.1, 5.2 and 5.3. Unfortunately, the digital sensor can only be tested using a connection to the Arduino Uno, since integration with the microcontroller failed due to the limited Flash storage.

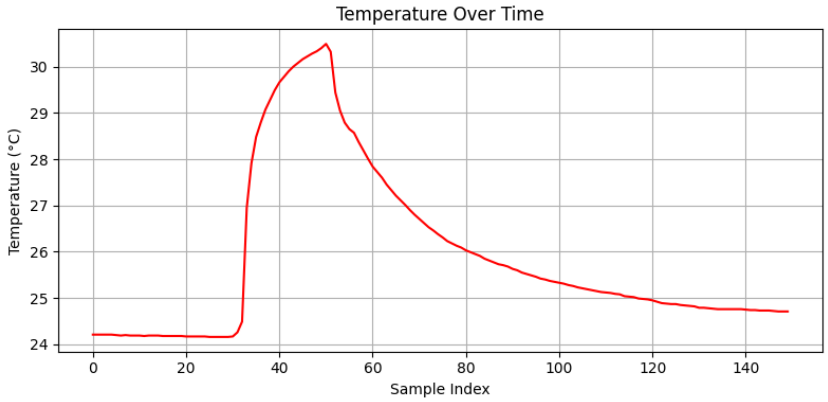


Figure 5.1: Temperature measurement using BME280

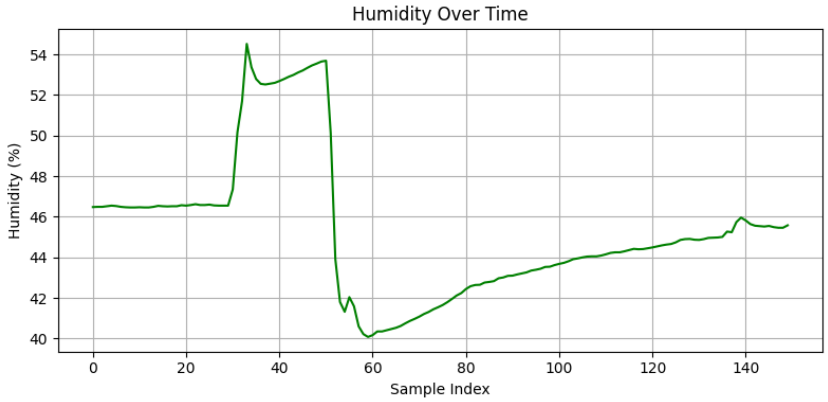


Figure 5.2: Humidity measurement using BME280

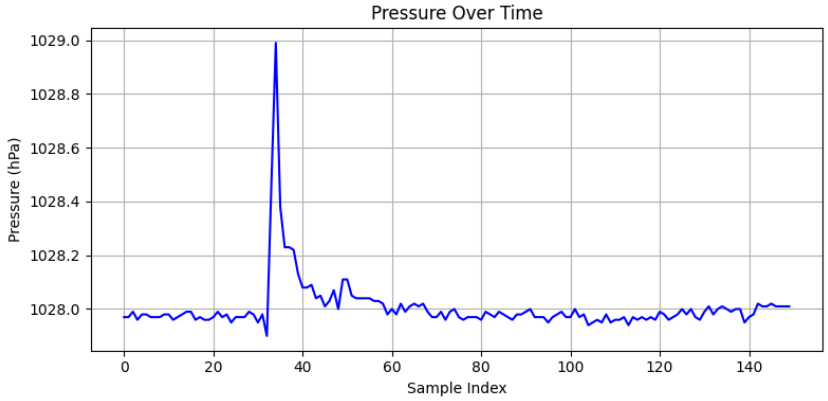


Figure 5.3: Pressure measurement using BME280

Transmission modes

Afterwards the data transmission of the analog temperature data was tested using various transmission modes, as explained in the last chapter (4). Three different transmission modes have been used during these measurements, namely:

- Continuous transmission mode
- Delayed transmission mode (1 second delay)
- Burst transmission mode (3 consecutive measurements, then 1 second delay)

The results of the measurements were copied from the serial monitor and converted into graphs, using the code shown in Appendix 8.8 (listing 8.8), as can be seen below in Figures 5.4, 5.5 and 5.6 respectively. Each measurement started with an initial period of rest without external influences to create a baseline. Afterwards a small heat source was brought close to the temperature sensor for a limited amount of time to determine its instantaneous response to a change in temperature, as well as the time it would take to recover back to the baseline. The graphs show that the sensor can react instantaneously to a change in temperature and thereafter slowly reach the thermal equilibrium of the external heat source. However, the different transmission modes exhibit varying levels of measurement stability. The continuous transmission mode shows the greatest variations due to its frequency at which the measurements occur; a small change in voltage supplied to the module or a small change in the ambient circumstances can result in deviations, which are clearly visible in the graphs. Whereas the continuous mode shows values lying in the range of 29 to 30 °C, the other transmission modes produce more stable measurements, consistently around 28 °C, due to their internal delay of 1 second after each measurement.

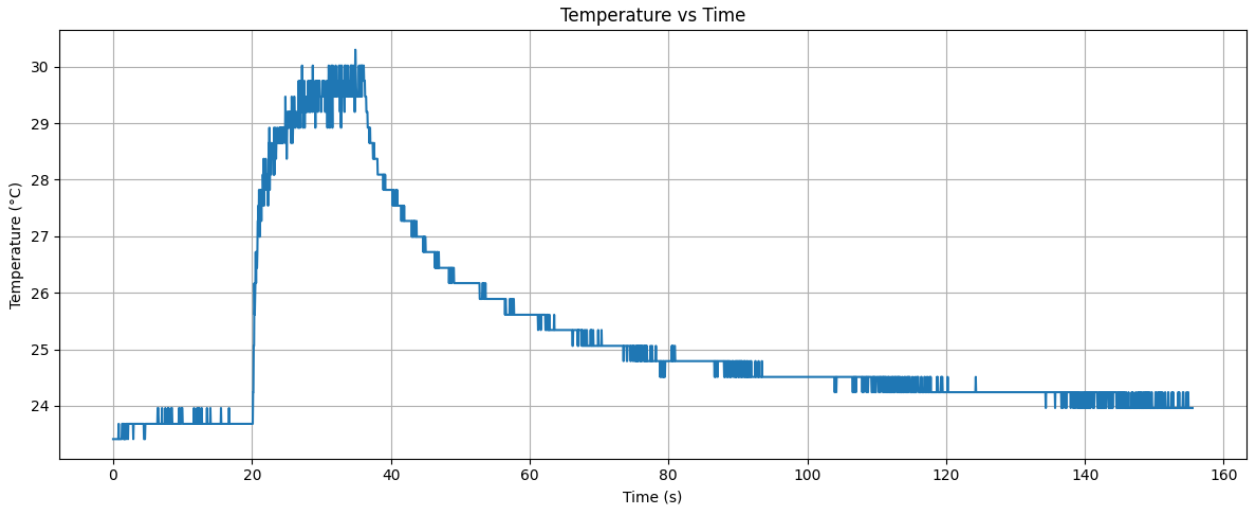


Figure 5.4: Measurement of an increase of temperature using continuous transmission mode

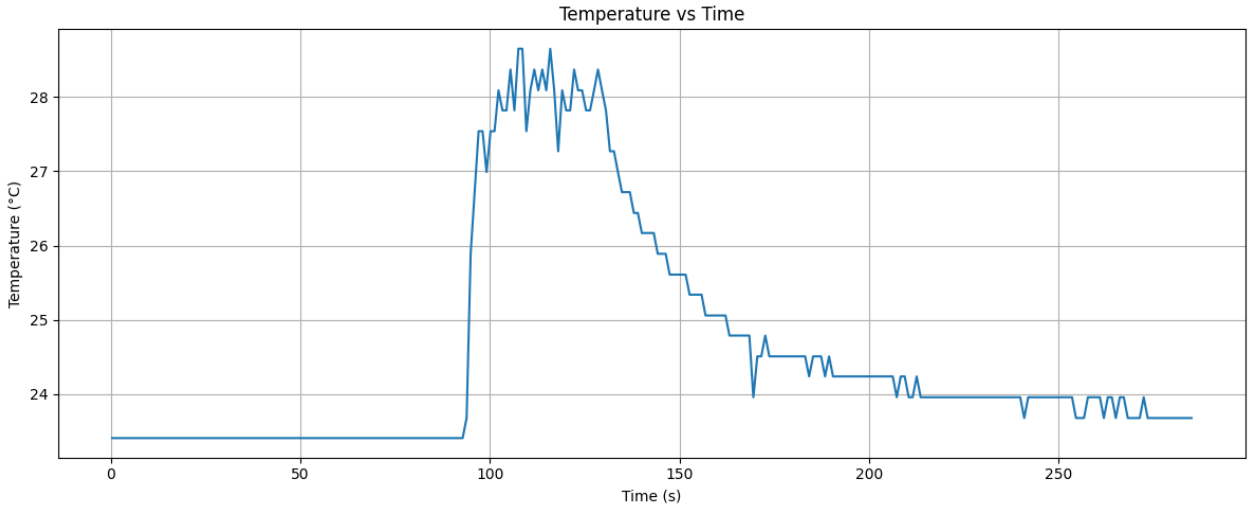


Figure 5.5: Measurement of an increase of temperature using delayed transmission mode

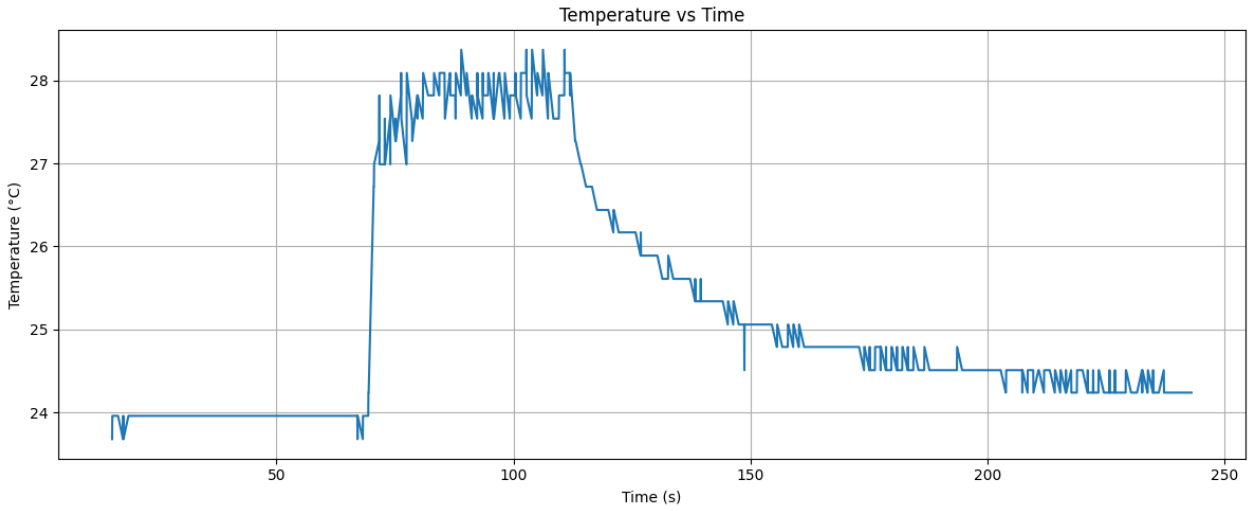


Figure 5.6: Measurement of an increase of temperature using burst transmission mode

As for the time required to converge back to the baseline, this also varies depending on the type of transmission mode. From Figure 5.4 can be seen that it takes around 97 seconds before the temperature has returned to the baseline when the continuous transmission mode is turned on, starting at 36 seconds and having returned around 133 seconds. This range is almost the same for when burst transmission is applied, yet a little smaller, taking about 95 seconds. In contrast, when the delayed version is used, the system takes significantly longer to return to the baseline, requiring approximately 125 seconds. Possible explanations for this difference could be due to having a lower amount of data points available compared to continuous and burst mode or a potential self-heating effect of the analog sensor in delayed mode, resulting in prolonged or sustained heating. Although both burst and delayed modes operate with a 1-second interval between transmissions, burst mode captures three measurements per cycle, resulting in a threefold increase in data point density. This leads to significantly higher temporal resolution compared to the delayed mode, which records only one measurement per interval.

Sleep mode

When attempting to test the functionality of the sleep mode of the ATmega8, the current consumption stayed well above the mA threshold, indicating that the power-down mode was not enabled. The code shown in Appendix 8, listing 8.6, did cause the system to do a measurement correctly after every 30 seconds, but in the downtime, the current hovered around the 2 mA mark, which is too high.

Power consumption

Following that, the power consumption of the system was measured when connected to a function generator in the lab. First the ATmega8 microcontroller was tested while being disconnected from the analog sensor and transceiver. Thereafter the analog sensor was added and finally the transceiver was reconnected to form the entire system again. The entire system was tested by varying the level of transmission power as well as varying the transmission mode of the data. The drawn current was measured and logged for two cases, namely for an internal oscillator frequency of 4 MHz and 8 MHz, and are depicted in Tables 5.2 and 5.3. These are the most commonly used operating frequencies for the internal oscillator. As can be seen from the tables below, the current of the module varies with both the transmission power and the transmission mode. An increase in the transmission power leads to an increase of active current since the system requires more power for its transmission. Similarly, changing the transmission mode to a continuous form, by means of removing the delay, leads to a higher amount of current due to the system being active continuously. In addition, increasing the frequency of the internal oscillator leads to an increase of current. An increase of frequency causes the clock period to become shorter, allowing for more measurements to be taken within the same amount of time compared to a case when the internal frequency is 4 MHz.

Components (4 MHz)	Transmission power	Transmission mode	Current (mA)
ATmega8	X	X	4.5
ATmega8 + Sensor	X	X	4.6
Complete system	LOW	Burst 3x	5.0
Complete system	MIN	Delayed	4.9
Complete system	LOW	Delayed	5.0
Complete system	MAX	Delayed	4.7
Complete system	MIN	Continuous	12.0
Complete system	LOW	Continuous	13.0
Complete system	MAX	Continuous	17.8

Table 5.2: Current consumption of the module using different transmission powers and modes for 4 MHz

Components (8 MHz)	Transmission power	Transmission mode	Current (mA)
ATmega8	X	X	5.8
ATmega8 + Sensor	X	X	6.0
Complete system	LOW	Burst 3x	6.7
Complete system	MIN	Delayed	6.6
Complete system	LOW	Delayed	6.7
Complete system	MAX	Delayed	6.9
Complete system	MIN	Continuous	12.8
Complete system	LOW	Continuous	14.2
Complete system	MAX	Continuous	18.6

Table 5.3: Current consumption of the module using different transmission powers and modes for 8 MHz

Total system measurements

Finally, the wireless sensor system was connected to the energy harvester and the power conditioning circuit to test the functionality of the whole system. The system was tested for four different size storage capacitors to determine the optimal configuration in terms of the amount of (useful) measurements sent per minute. The results of this test can be seen in Figure 5.7 and the individual measurements per capacitor can be found in Appendix 9. From the graph can be seen that the 680 μF capacitor results in the highest amount of measurements, total as well as useful, per minute. The amount of measurements per minute for the larger capacitors is lower since they require more time to be fully charged, while they do sustain longer burst of correct measurements. On the other hand, the smaller capacitors charge up faster, but give smaller bursts of data.
When the storage capacitor become too small, the value for the supply voltage in Equation 4.1 had to be lowered to 3.3V to give accurate measurements. This is due to the fact that the capacitor can store less energy and charges quicker. As can be seen in the individual measurements per capacitor, at a certain point the measurements start to get distorted, at that point the capacitor is close to being empty, causing the analog sensor to receive less voltage.

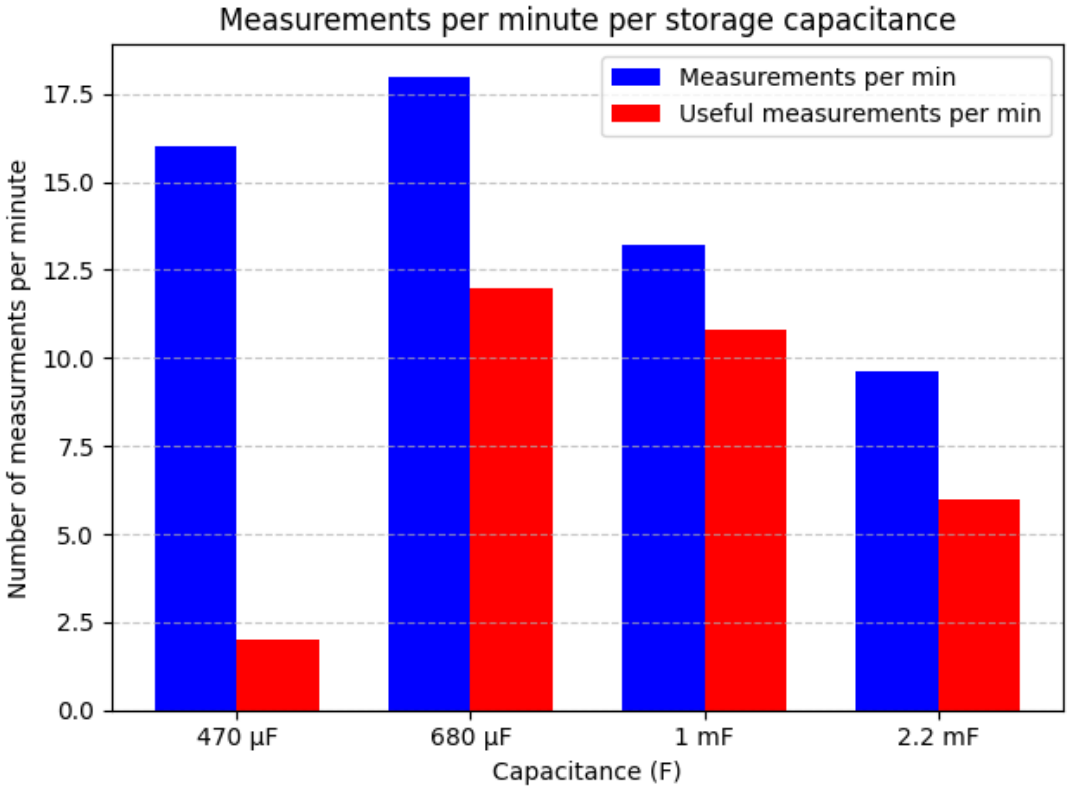


Figure 5.7: Influence of the storage capacitor size on the amount of measurements done per minute

6

Discussion

The experimental results obtained in this thesis provide valuable insights into the behavior and performance of a wireless sensor system operating under various configurations. The system's response to changes in temperature, transmission mode, power settings, and energy availability reveals both strengths and limitations of the current implementation.

A key finding from the transmission mode evaluation is the significant influence of sampling frequency on measurement stability. Continuous transmission mode, despite offering high temporal resolution, introduced greater variability in temperature readings. This instability is likely caused by environmental noise and fluctuations in the supply voltage, which are more pronounced due to the high measurement rate. In contrast, the delayed and burst modes exhibited more consistent performance, with burst mode showing the most favorable balance between resolution and stability. These results suggest that incorporating intentional delays can reduce noise sensitivity and improve wireless sensor systems.

The analysis regarding recovery time after thermal variations further emphasized the role of sampling density. While all modes showed an relatively fast response to external heat sources, the delayed mode had a noticeably slower return to baseline. One possible explanation is the self-heating of the sensor due to longer idle periods between samples. Another contributing factor could be the limited number of data points, making it more difficult to capture the temperature decay curve accurately.

The attempt to implement a low-power sleep mode revealed an unexpected limitation in the system. Although the microcontroller successfully paused for 30 seconds between measurements, the measured current did not decrease as expected. This suggests that either the microcontroller did not fully enter a low-power state or that other components, such as the sensor and transceiver, remained active. Given the importance of energy efficiency in remote or energy-harvesting applications, this issue warrants further investigation, possibly including hardware-level debugging or the use of external power management systems to ensure a low-power operation.

The power consumption study provided a comprehensive view of how transmission mode, transceiver power level, and oscillator frequency affect the system's energy usage. As anticipated, higher oscillator frequencies and continuous transmission increased current draw. However, an anomaly was observed when measuring at 4 MHz with MAX power in the delayed mode, where the current consumption was unexpectedly lower than in the low- and mid-power cases. This could be attributed to either measurement error or the transceiver operating within a more efficient radio frequency output range, where the ratio between power and current is improved.

Integration with the energy harvester and power conditioning circuit demonstrated the system's ability to function autonomously under energy-constrained conditions. Testing with various storage capacitor sizes showed that a 680 μF capacitor yielded the best performance in terms of quantity and quality of data transmission. Smaller capacitors delivered shorter bursts of data and were more susceptible to

voltage drops, which led to distorted sensor readings, while they are on the other hand faster at charging. This highlights the trade-off between energy storage capacity and responsiveness as well as the importance of capacitor sizing for energy harvester systems.

Overall, the system demonstrated good responsiveness, provided reasonable measurement data and demonstrated the potential for energy-autonomous operation. However, limitations related to power management and environmental stability still remain, offering opportunities for refinement in future designs.

7

Conclusion

In this thesis, a low-power wireless sensing system was designed, implemented, and evaluated with a focus on measurement accuracy, transmission stability, and power efficiency. The analog sensor output was initially validated against oscilloscope readings, revealing a consistent 0.02 V deviation when transmitted via the wireless transceivers. This difference was corrected by introducing a compensation factor in the output voltage calculation.

Three transmission strategies: continuous, delayed, and burst mode were investigated to evaluate their impact on system performance. Burst mode emerged as the most balanced approach, offering greater measurement stability compared to continuous mode, while maintaining higher temporal resolution than delayed mode. Recovery times after thermal adjustments varied between modes, with delayed mode showing slower convergence, likely due to reduced data density or minor self-heating.

Although a sleep mode was implemented in code, power consumption did not decrease as expected. This suggests either improper implementation or hardware limitations that prevent effective use of low-power states. Further investigation is necessary to determine the exact cause.

Power consumption analysis revealed the expected interactions between current draw and parameters such as oscillator frequency, transmission power, and transmission mode. Higher frequencies and continuous transmission increased power usage. An anomaly was observed at 4 MHz under maximum transmission power in delayed mode, where current draw was unusually lower. This may be due to measurement errors or the transceivers possessing an optimal efficiency at higher output power.

Finally, the system was integrated with an energy harvester, in combination with a power conditioning circuit, and tested using four different storage capacitor values. The 680 μF capacitor yielded the highest number of both total and useful measurements per minute, representing the best trade-off between charge time and sustained operation. Smaller capacitors charged more quickly but produced shorter data bursts. It was also observed that at very low storage capacities, the supply voltage dropped enough to distort measurements, forcing a temporary adjustment of the reference voltage in calculations.

Overall, this work demonstrates the viability of a low-power, wireless sensor system powered by vibrational energy harvesting. The results provide practical insights into optimizing transmission strategies and hardware configurations for maximizing performance and energy efficiency in constrained environments.

Future work

While this thesis successfully demonstrated the design and implementation of a low-power wireless sensing systems, several aspects still require more improvement, work and research.

First of all, the implementation of a properly functioning sleep mode to reduce the power consumption of the system. Currently the system draws, even on the lowest current consumption mode, around 16.5 μW at all times. Implementing a sleep mode that could work hand in hand with the already low duty cycle could result in an even lower power consumption.

In addition, depending on the energy availability of the system, a form of adaptive switching technology could be implemented. This will only be possible if the power consumption of the system is lower, but it would be very worthwhile to research further. Having an adaptive system that can decide how much measurements are feasible depending on the amount of energy available, will transform it into an autonomous system. This ensures that the system will require minimal to no human intervention when placed on the hard-to-reach areas, like with our case inside an airplane wing.

Having a network of wireless sensors could also improve the functionality of the system and mostly improve the independence of the system. Whenever a node fails or the connection between 2 nodes is broken, it will be possible for the network to reroute the signal throughout other nodes to allow the signal to reach its destination. The system will in this case be ”self-healing”, since it restores the broken connection by means of rerouting it.

Moreover, a digital sensor could be used instead of an analog sensor. Analog sensors require to be powered continuously and are more susceptible to noise and signal degradation compared to digital sensors. We were unable to implement the digital sensor in our configuration, because the required ”Wire” library necessary to setup the I2C communication was too big in bytes for the storage of the ATtiny85 as well as the ATmega8. There also exists a tiny Wire library, unfortunately we were unable to make that version, in combination with the library for the BME280, functioning. It is thus required to have a microcontroller with approximately the same power consumption, or of course less, and more storage space to ensure the Wire library can be used.

Finally, this implementation requires rigorous amounts of field testing under the actual environmental circumstances of the plane wing in mid flight. Currently the system has been tested at room temperature and only given a small increase of temperature, but the temperature inside the wing could chill down to -45°C. Also, it is unclear if the system can withstand the pressure from being kilometers up in the air. These two circumstances could cause the electronics to get damaged or even fail. Furthermore, the signals from the transceivers could possible interfere with the RF communication of the avionics. Overall, the system should be tested with extreme caution to ensure that it works, while upholding the (safety) regulations linked to aviation.

Bibliography

[1] IBM. *What is the Internet of Things (IoT)?* May 12, 2023. URL: https://www.ibm.com/think/topics/internet-of-things (visited on 05/02/2025).

[2] Hrishikesh Jayakumar et al. *Powering the Internet of Things*. 2014. URL: http://dx.doi.org/10.1145/2627369.2631644.

[3] Carmela Mangone et al. “Design and performance of flexible polymeric piezoelectric energy harvesters for battery-less tyre sensors”. In: *Smart Materials and Structures* 31.9 (July 18, 2022), p. 095034. DOI: 10.1088/1361-665x/ac8211. URL: https://doi.org/10.1088/1361-665x/ac8211.

[4] Shadi Al-Sarawi et al. *Internet of Things (IoT) Communication Protocols: Review*. 2017.

[5] Zhengbao Yang a, Alper Erturk, and Jean Zu. *On the efficiency of piezoelectric energy harvesters*, pp. 26–37. URL: http://dx.doi.org/10.1016/j.eml.2017.05.002.

[6] Ivan A. Parinov and Alexander V. Cherpakov. *Overview: State-of-the-Art in the Energy Harvesting Based on Piezoelectric Devices for Last Decade*. 2022, pp. 765–. URL: https://doi.org/10.3390/sym14040765.

[7] Haitong Liang et al. *A review on vibration-based piezoelectric energy harvesting from the aspect of compliant mechanisms*. 2021, p. 112743. URL: https://doi.org/10.1016/j.sna.2021.112743.

[8] Eli S Leland and Paul K Wright. “Resonance tuning of piezoelectric vibration energy scavenging generators using compressive axial preload”. In: *Smart Materials and Structures* 15.5 (Sept. 8, 2006), pp. 1413–1420. DOI: 10.1088/0964-1726/15/5/030. URL: https://doi.org/10.1088/0964-1726/15/5/030.

[9] Zhang Qichang, Yang Yang, and Wang Wei. “Theoretical Study on Widening Bandwidth of Piezoelectric Vibration Energy Harvester with Nonlinear Characteristics”. In: *Micromachines* 12.11 (Oct. 23, 2021), p. 1301. DOI: 10.3390/mi12111301. URL: https://doi.org/10.3390/mi12111301.

[10] S Zhao et al. “Co-optimization of a piezoelectric energy harvesting system for broadband operation”. In: *Journal of Physics Conference Series* 1407.1 (Nov. 1, 2019), p. 012010. DOI: 10.1088/1742-6596/1407/1/012010. URL: https://arxiv.org/abs/1910.00557.

[11] Young-Gyun Kim et al. “Improved performance of stretchable piezoelectric energy harvester based on stress rearrangement”. In: *Scientific Reports* 12.1 (Nov. 9, 2022). DOI: 10.1038/s41598-022-23005-2. URL: https://doi.org/10.1038/s41598-022-23005-2.

[12] Action Nechibvute, Albert Chawanda, and Pearson Luhanga. “Piezoelectric Energy Harvesting Devices: an alternative energy source for wireless sensors”. In: *Smart Materials Research* 2012 (May 21, 2012), pp. 1–13. DOI: 10.1155/2012/853481. URL: https://doi.org/10.1155/2012/853481.

[13] Alessandro Pracucci et al. “Integration of Piezoelectric Energy Harvesting Systems into Building Envelopes for Structural Health Monitoring with Fiber Optic Sensing Technology”. In: *Energies* 17.7 (Apr. 8, 2024), p. 1789. DOI: 10.3390/en17071789. URL: https://doi.org/10.3390/en17071789.

[14] Brendan L. Turner et al. *Ultrasound-Powered Implants: A Critical Review of Piezoelectric material selection and applications*. 2021, pp. 2100986–. URL: https://doi.org/10.1002/adhm.202100986.

[15] Qiongfeng Shi, Tao Wang, and Chengkuo Lee. “MEMS based broadband Piezoelectric Ultrasonic Energy Harvester (PUEH) for enabling Self-Powered Implantable biomedical devices”. In: *Scientific Reports* 6.1 (Apr. 26, 2016). DOI: 10.1038/srep24946. URL: https://doi.org/10.1038/srep24946.

[16] Nurettin Sezer, Muammer Koç, and The Authors. “A comprehensive review on the state-of-the-art of piezoelectric energy harvesting”. In: *Nano Energy* 80 (2021), p. 105567. URL: https://doi.org/10.1016/j.nanoen.2020.105567.

[17] Marco Grossi. *Energy Harvesting Strategies for Wireless Sensor Networks and Mobile Devices: A Review*. 2021, pp. 661–. DOI: 10.3390/electronics10060661. URL: https://doi.org/10.3390/electronics10060661.

[18] Felix Mazunga and Action Nechibvute. “Ultra-low power techniques in energy harvesting wireless sensor networks: Recent advances and issues”. In: *Scientific African* 11 (Feb. 10, 2021), e00720. DOI: 10.1016/j.sciaf.2021.e00720. URL: https://doi.org/10.1016/j.sciaf.2021.e00720.

[19] Junaid Ahmed Khan, Hassaan Khaliq Qureshi, and Adnan Iqbal. *Energy management in Wireless Sensor Networks: A survey*. July 11, 2014. URL: http://dx.doi.org/10.1016/j.compeleceng.2014.06.009.

[20] Barclay Shaw et al. *Energy-Aware wireless microsensor networks*. Mar. 2002.

[21] Mohammad Abdur Razzaque and Simon Dobson. “Energy-Efficient sensing in wireless sensor networks using compressed sensing”. In: *Sensors* (Feb. 12, 2014), pp. 2822–2859. DOI: 10.3390/s140202822. URL: www.mdpi.com/journal/sensors.

[22] Bolaji Omodunbi et al. *A Review of Energy Conservation in Wireless Sensor Networks*. No.5. 2013, pp. 17–18.

[23] Runtong Zhang et al. “A Study on an Energy Conservation and Interconnection Scheme between WSN and Internet Based on the 6LoWPAN”. In: *Mobile Information Systems* 2015 (Jan. 1, 2015), pp. 1–11. DOI: 10.1155/2015/592613. URL: https://doi.org/10.1155/2015/592613.

[24] Alexandros Gazis et al. “Comprehensive review: sensor technologies, instrumentation, and signal processing in low-power IoT”. In: *Academia Engineering* 2.1 (Feb. 28, 2025). DOI: 10.20935/acadeng7541. URL: https://doi.org/10.20935/acadeng7541.

[25] Bureau of Meteorology. *About weather station data*. URL: http://www.bom.gov.au/climate/data/stations/about-weather-station-data.shtml#:~:text=At%20the%20majority%20of%20locations,report%20on%20an%20hourly%20basis..

[26] Hailin Jiang, M. Marek-Sadowska, and S.R. Nassif. *Benefits and costs of power-gating technique*. Oct. 2005. URL: https://ieeexplore.ieee.org/abstract/document/1524207?casa_token=W1-BIjk1pcAAAAAA:ozzw_66FXtcmaTpfFzhlJiNymfXQgaaeFV2UrpkTSz73j0CejosrC5NE-mVXomZ-7YGoMiadTGo.

[27] Sujay Narayana et al. *PIR sensors*. Apr. 2015. DOI: 10.1145/2737095.2742561. URL: https://doi.org/10.1145/2737095.2742561.

[28] Elke Mackensen, Matthias Lai, and Thomas M. Wendt. *Bluetooth Low Energy (BLE) based wireless sensors*. Oct. 2012. URL: https://ieeexplore.ieee.org/document/6411303.

[29] Mohammad Afaneh. *Bluetooth Low Energy Power Consumption; How to Achieve Maximum Battery Life*. May 2023. URL: https://novelbits.io/ble-power-consumption-optimization/.

[30] Jetmir Haxhibeqiri et al. “A Survey of LoRaWAN for IoT: From Technology to Application”. In: *Sensors* 18.11 (Nov. 16, 2018), p. 3995. DOI: 10.3390/s18113995. URL: https://www.mdpi.com/1424-8220/18/11/3995.

[31] TEKTELIC Communications Inc. *LoRaWAN - Most Common Applications and Use Cases | IoT For All*. Dec. 2024. URL: https://www.iotforall.com/lorawan-most-common-applications-and-use-cases.

[32] Lluís Casals et al. “Modeling the Energy Performance of LoRaWAN”. In: *Sensors* 17.2364 (Oct. 2017), pp. 2–30. DOI: 10.3390/s17102364. URL: www.mdpi.com/journal/sensors.

[33] Ramya Muthu, M. Shanmugaraj, and R. Prabakaran. *STUDY ON ZIGBEE TECHNOLOGY*. Tech. rep. July 2011. DOI: 10.1109/ICECTECH.2011.5942102. URL: https://ieeexplore-ieee-org.tudelft.idm.oclc.org/abstract/document/5942102.

[34] Il-Gu Lee et al. “WiFi HaLow for Long-Range and Low-Power Internet of Things: System on Chip Development and Performance Evaluation”. In: *IEEE Communications Magazine* 59.7 (July 2021), pp. 101–107. DOI: 10.1109/mcom.001.2000815. URL: https://doi.org/10.1109/mcom.001.2000815.

[35] Nordic Semiconductor ASA. *NRF24L01 Single Chip 2.4GHz Transceiver Product Specification*. July 2007. URL: https://cdn.sparkfun.com/datasheets/Wireless/Nordic/nRF24L01_Product_Specification_v2_0.pdf.

[36] Ankit Singh and AZoSensors. *Aircraft Sensor Systems: A Technical Overview of Safety, Navigation, and Performance Monitoring*. Mar. 2025. URL: https://www.azosensors.com/article.aspx?ArticleID=1614.

[37] Atmel. *ATtiny25/45/85 [DATASHEET]*. Aug. 2013. URL: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2586-avr-8-bit-microcontroller-attiny25-attiny45-attiny85_datasheet.pdf.

[38] STMicroelectronics. *Ultra-low current 2.4 V precision analog temperature sensor*. 12495. June 2010, pp. 1–19. URL: www.st.com.

[39] *ATmega8(L)*. Feb. 2013. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf.

[40] Bosch Sensortec. *BME280 Data sheet*. Jan. 2024. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf.

[41] Instructables. *NRF24L01+ with ATTiny85 3 pins*. Sept. 2017. URL: https://www.instructables.com/NRF24L01-With-ATtiny85-3-Pins/.

8

Appendix A

Used code

Listing 8.1: Communication with analog sensor, Arduino Uno, TX side

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

int test;

RF24 radio(9, 10); // CE, CSN

const byte address[6] = "00001";

void setup() {
  radio.begin();
  pinMode(A5, INPUT); // A5 is one of the analog ports on the Arduino Uno
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_LOW);  // Use LOW for stability with weak power
  radio.setDataRate(RF24_250KBPS); // Improve range and reliability
  radio.stopListening();  // Transmitter mode
}

void loop(){
  test = analogRead(A5);
  radio.write(&test, sizeof(test));
  delay(100);
}
```

Listing 8.2: Test of digital sensor with Arduino uno, modified version of ”bmetest”, which can be found inside the downloaded bme280 library inside Arduino IDE

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

#define BME_SCK 13
#define BME_MISO 12
#define BME_MOSI 11
#define BME_CS 10

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C
//Adafruit_BME280 bme(BME_CS); // hardware SPI
//Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // software SPI

unsigned long delayTime;

void setup() {
    Serial.begin(9600);
    while(!Serial);    // time to get serial running
    Serial.println(F("BME280 test"));

    unsigned status;

    // default settings
    status = bme.begin(0x76);
    // You can also pass in a Wire library object like &Wire2
    // status = bme.begin(0x76, &Wire2)
    if (!status) {
        Serial.println("Could not find a valid BME280 sensor, check wiring, address,
        Serial.print("SensorID was: 0x"); Serial.println(bme.sensorID(),16);
        Serial.print("
ID of 0xFF probably means a bad address, a BMP 180 or BMP 085\n");
        Serial.print("   ID of 0x56-0x58 represents a BMP 280,\n");
        Serial.print("        ID of 0x60 represents a BME 280.\n");
        Serial.print("        ID of 0x61 represents a BME 680.\n");
        while (1) delay(10);
    }

    Serial.println("-- Default Test --");
    delayTime = 1000;

    Serial.println();
}


void loop() {
    printValues();
    delay(delayTime);
}
```

```
void printValues() {
    Serial.print("Temperature = ");
    Serial.print(bme.readTemperature());
    Serial.println(" °C");

    Serial.print("Pressure = ");

    Serial.print(bme.readPressure() / 100.0F);
    Serial.println(" hPa");

    Serial.print("Humidity = ");
    Serial.print(bme.readHumidity());
    Serial.println(" %");

    Serial.println();
}
```

Listing 8.3: Communication with analog sensor, Arduino Uno, RX side

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9,10);  // CE, CSN

const byte address[6] = "00001";

int test;
float temp;
float vout;

void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_LOW);
  radio.setDataRate(RF24_250KBPS);
  radio.startListening();  // Receiver mode
  Serial.println("Ready to receive...");
}

void loop() {
  if (radio.available()) {
    int test;
    radio.read(&test, sizeof(test));
// 5V used since this is a test on Arduino
    vout = 1.04*(5*float(test))/1024; // Convert output value to Vout
    temp = (1000*vout - 1866.3) / -11.69; // Convert Vout to temperature (°C)
    Serial.print("Received: ");
    Serial.println(temp);
  }
}
```

Listing 8.4: Communication with analog sensor, controlled by ATmega8, TX side

```
#include <nRF24L01.h>
#include <RF24.h>

int test;
int i=0;

// CE on Pin 3, CSN on Pin 4 (ATmega8 physical pins: 5 and 6)
RF24 radio(3, 4);

const byte address[6] = "00001";

void setup() {
  pinMode(A0, INPUT);
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);  // Use LOW for stability with weak power
  radio.setDataRate(RF24_250KBPS); // Improve range and reliability
  radio.setChannel(50);
  radio.stopListening();  // Transmitter mode
}

void loop() {
  //Comment out "for" when using continuous or delayed
  //for (i=0;i<3;i++) {
  test = analogRead(A0);     // Read analog value from sensor at pin A0
  radio.write(&test, sizeof(test));
  //}

  delay(1000);           // Comment out when using continous
}
```

Listing 8.5: Communication with analog sensor, controlled by ATmega8, RX side

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9,10);  // CE, CSN

const byte address[6] = "00001";

int test;
float temp;
float vout;

void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_LOW);
  radio.setDataRate(RF24_250KBPS);
  radio.setChannel(50);
  radio.startListening();  // Receiver mode
  Serial.println("Ready to receive from ATmega8...");
}

void loop() {
  if (radio.available()) {
    int test;
    radio.read(&test, sizeof(test));
    vout = 1.04*(3.4*float(test))/1024;      // Analog value to vout
    temp = (1000*vout - 1866.3) / -11.69;    // Vout to temperature (°C)
    Serial.print("Received: ");
    Serial.println(temp);
  }
}
```

Listing 8.6: Test with sleep mode, TX side

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <EEPROM.h>
#include <avr/sleep.h>
#include <avr/wdt.h>
#include <avr/interrupt.h>

#define RESET_COUNT_ADDR 0  // EEPROM address to store reset counter

RF24 radio(3, 4);        // CE, CSN
const byte address[6] = "00001";
int raw = 0;

uint8_t readResetCount() {
  return EEPROM.read(RESET_COUNT_ADDR);
}

void writeResetCount(uint8_t count) {
  EEPROM.write(RESET_COUNT_ADDR, count);
}

void setupWatchdogReset() {
  cli(); // Disable interrupts


  MCUSR &= ~(1 << WDRF);         // Clear watchdog reset flag

  WDTCR = (1 << WDCE) | (1 << WDE);
  WDTCR = (1 << WDE) | (1 << WDP2) | (1 << WDP1); // 1s timeout

  sei(); // Enable interrupts
}

void enterSleep() {
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable();
  sleep_cpu();       // Sleep until watchdog resets the MCU
  sleep_disable();
}

void setup() {
  analogReference(DEFAULT);

  uint8_t resetCount = 0;

  if (MCUSR & (1 << WDRF)) {
    resetCount = readResetCount();
    resetCount++;
    if (resetCount > 30) resetCount = 30;
  } else {
    resetCount = 0; // Normal power-on or external reset
  }
```

```
  MCUSR = 0;     // Clear reset flag

  writeResetCount(resetCount);

  pinMode(A0, INPUT);
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_LOW);
  radio.setDataRate(RF24_250KBPS);
  radio.setChannel(50);
  radio.stopListening();

  setupWatchdogReset();

  if (resetCount < 30) {
    enterSleep(); // Go back to sleep
  }
  else {
    int raw = analogRead(A0);
    radio.write(&raw, sizeof(raw));

    writeResetCount(0);      // Reset counter after transmission
  }
}

void loop() {
  // Not used. All logic runs in setup due to reset-based flow.
}
```

Listing 8.7: Test with sleep mode, RX side

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9,10);  // CE, CSN

const byte address[6] = "00001";

int test;
float temp;
float vout;

bool startMillis = 1;
int baseMillis;

void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_LOW);
  radio.setDataRate(RF24_250KBPS);
  radio.setChannel(50);
  radio.startListening();  // Receiver mode
  Serial.println("Ready to receive from ATmega8...");
}

void loop() {
  if (radio.available()) {
    radio.read(&test, sizeof(test));
    if (startMillis == 1){          \\A counter to calculate the time
      baseMillis = millis();        \\ in between measurements
      startMillis = 0;
    }
    vout = 1.04*(3.3*float(test))/1024;
    temp = (1000*vout - 1866.3) / -11.69;
    Serial.print("Received: ");
    Serial.print(float(millis()-baseMillis)/1000);
    Serial.print("s, ");
    Serial.println(temp);
    }
}
```

Listing 8.8: Python code for extracting analog data from Arduino IDE

```
import matplotlib.pyplot as plt
import re

# Data extraction from file
with open("Contineous Temp 4Mhz, low.txt") as file:
  for data in file:
    # Split samples
    samples = data.split('---')

    # Initialize lists
    timestamps = []
    temperatures = []

    # Extract data from sample with Regex
    for sample in samples:
        match = re.search(r'([\d.]+)s, Received: ([\d.]+)', sample)
        if match:
            time = float(match.group(1))
            temp = float(match.group(2))
            timestamps.append(time)
            temperatures.append(temp)

# Plotting
plt.figure(figsize=(12, 5))
plt.plot(timestamps, temperatures)
plt.title('Temperature vs Time')
plt.xlabel('Time (s)')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Listing 8.9: Python code for extracting digital data from Arduino IDE

```
import re
import matplotlib.pyplot as plt

# Your raw data string
with open("Digital sensor test.txt") as file:
  for data in file:
    # Regex pattern to extract all groups of Temperature, Pressure, Humidity
    pattern = r"Temperature\s*=\s*([\d.]+)\s*°CPressure\s*=\s*([\d.]+)\s*hPaHumidity\s*=\s

    # Find all matches
    matches = re.findall(pattern, data)

    # Convert matches to separate lists
    temperatures = [float(temp) for temp, _, _ in matches]
    pressures = [float(pres) for _, pres, _ in matches]
    humidities = [float(hum) for _, _, hum in matches]

plt.figure(figsize=(8, 4))
plt.plot(temperatures, color='red')
plt.title("Temperature Over Time")
plt.xlabel("Sample Index")
plt.ylabel("Temperature (°C)")
plt.grid(True)
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 4))
plt.plot(pressures, color='blue')
plt.title("Pressure Over Time")
plt.xlabel("Sample Index")
plt.ylabel("Pressure (hPa)")
plt.grid(True)
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 4))
plt.plot(humidities, color='green')
plt.title("Humidity Over Time")
plt.xlabel("Sample Index")
plt.ylabel("Humidity (%)")
plt.grid(True)
plt.tight_layout()
plt.show()
```

9

Appendix B

Transmission data

```
09:56:19.927 -> Received: 26.37, 1.56
09:56:19.958 -> Received: 25.49, 1.57
09:56:19.958 -> Received: 25.20, 1.57
09:56:19.991 -> Received: 25.49, 1.57
09:56:20.024 -> Received: 25.20, 1.57
09:56:20.024 -> Received: 25.20, 1.57
09:56:20.057 -> Received: 25.20, 1.57
09:56:20.090 -> Received: 24.90, 1.58
09:56:20.123 -> Received: 24.61, 1.58
09:56:20.123 -> Received: 24.03, 1.59
09:56:20.155 -> Received: 23.44, 1.59
09:56:20.188 -> Received: 22.56, 1.60
09:56:20.221 -> Received: 22.56, 1.60
09:56:20.221 -> Received: 19.06, 1.64
09:56:20.254 -> Received: 12.92, 1.72
09:56:20.286 -> Received: 8.53, 1.77
09:56:20.319 -> Received: 1.81, 1.85
09:56:20.319 -> Received: -5.21, 1.93
09:56:20.352 -> Received: -13.98, 2.03
09:56:20.385 -> Received: -22.17, 2.13
09:56:20.385 -> Received: -29.77, 2.21
```

Figure 9.1: Measurements per minute test for 2.2 mF

```
09:43:49.332 -> Ready to receive from ATmega8...
09:45:38.414 -> Received: 34.41, 1.46
09:45:38.461 -> Received: 34.41, 1.46
09:45:38.474 -> Received: 34.41, 1.46
09:45:38.474 -> Received: 34.41, 1.46
09:45:38.508 -> Received: 33.86, 1.47
09:45:38.541 -> Received: 34.13, 1.47
09:45:38.573 -> Received: 33.86, 1.47
09:45:38.573 -> Received: 33.32, 1.48
09:45:38.606 -> Received: 31.69, 1.50
09:45:38.639 -> Received: 22.73, 1.60
09:45:38.674 -> Received: 7.26, 1.78
```

Figure 9.2: Measurements per minute test for 1 mF

```
09:59:21.118 -> Ready to receive from ATmega8...
10:00:24.150 -> Received: 12.04, 1.73
10:00:24.150 -> Received: -1.12, 1.88
10:00:24.187 -> Received: 1.81, 1.85
10:00:24.250 -> Received: -1.41, 1.88
10:00:24.276 -> Received: -2.87, 1.90
10:00:24.276 -> Received: -6.67, 1.94
10:00:24.276 -> Received: -11.94, 2.01
10:00:24.309 -> Received: -14.57, 2.04
10:00:54.287 -> Received: 21.69, 1.61
10:00:54.319 -> Received: 19.93, 1.63
10:00:54.319 -> Received: 19.64, 1.64
10:00:54.352 -> Received: 19.93, 1.63
10:00:54.385 -> Received: 18.47, 1.65
10:00:54.417 -> Received: 18.18, 1.65
10:00:54.417 -> Received: 16.13, 1.68
10:00:54.450 -> Received: 13.50, 1.71
10:00:54.484 -> Received: -3.75, 1.91
10:01:25.632 -> Received: 19.93, 1.63
10:01:25.665 -> Received: 14.96, 1.69
10:01:25.665 -> Received: 13.50, 1.71
10:01:25.697 -> Received: 10.87, 1.74
10:01:25.730 -> Received: 10.58, 1.74
10:01:25.762 -> Received: 7.36, 1.78
10:01:25.762 -> Received: 5.90, 1.80
```

Figure 9.3: Measurements per minute test for 680 μF

```
10:03:53.452 -> Ready to receive from ATmega8...
10:04:44.973 -> Received: 20.52, 1.63
10:04:45.009 -> Received: 16.42, 1.67
10:04:45.038 -> Received: 15.55, 1.68
10:04:45.038 -> Received: 12.62, 1.72
10:04:45.071 -> Received: 11.45, 1.73
10:04:45.105 -> Received: 9.70, 1.75
10:04:45.105 -> Received: 7.36, 1.78
10:04:45.138 -> Received: 3.27, 1.83
10:05:14.802 -> Received: 19.06, 1.64
10:05:14.835 -> Received: 12.33, 1.72
10:05:14.835 -> Received: 10.28, 1.75
10:05:14.868 -> Received: 6.48, 1.79
10:05:14.868 -> Received: 5.02, 1.81
10:05:14.902 -> Received: 0.34, 1.86
10:05:14.934 -> Received: 0.34, 1.86
10:05:14.970 -> Received: -6.97, 1.95
10:05:43.754 -> Received: -23.05, 2.14
10:05:43.785 -> Received: -4.04, 1.91
10:05:43.817 -> Received: -12.23, 2.01
10:05:43.850 -> Received: -11.94, 2.01
10:05:43.883 -> Received: -18.95, 2.09
10:05:43.883 -> Received: -24.51, 2.15
10:05:43.916 -> Received: -30.36, 2.22
```

Figure 9.4: Measurements per minute test for 470 μF

10

Appendix C

System schematics

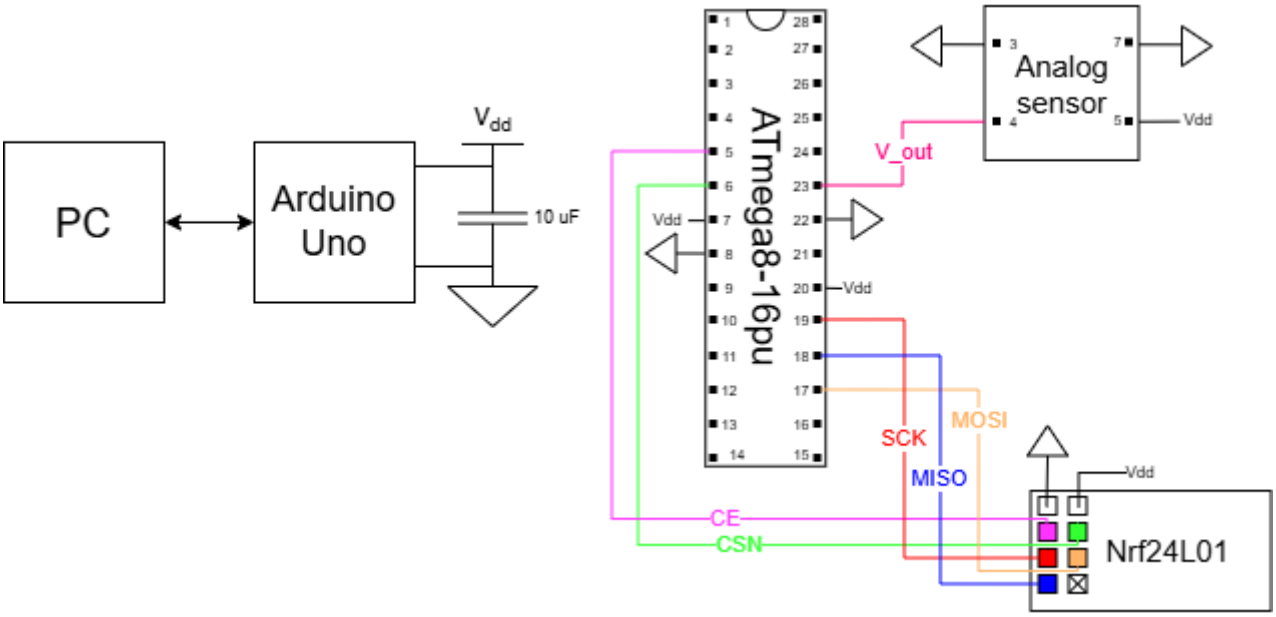


Figure 10.1: Schematic of Transmitter part made in Draw.io , from the setup in figure 4.2

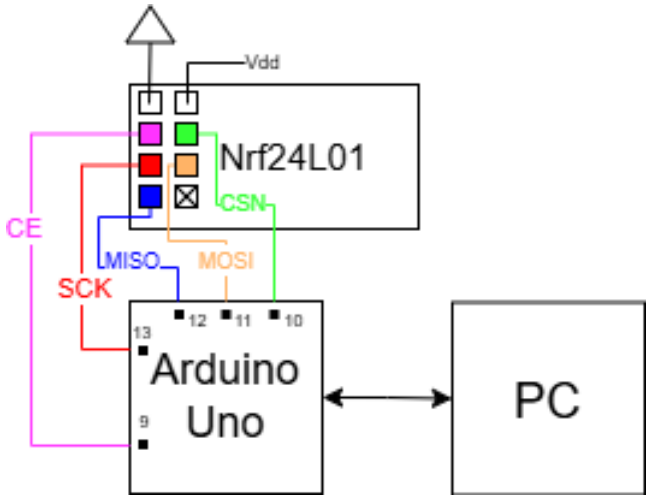


Figure 10.2: Schematic of Receiver part made in Draw.io , from the setup in figure 4.2