UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Trading Computation Time for Energy

Kann man Rechen-Leistung gegen Energie eintauschen?

Bachelorarbeit

verfasst am
Institut für IT-Sicherheit

im Rahmen des Studiengangs
IT-Sicherheit
der Universität zu Lübeck

vorgelegt von
Daniel Widmer

ausgegeben und betreut von
Prof. Dr. Esfandiar Mohammadi

Lübeck, den 19. Juni 2021

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

_____

Daniel Widmer

Zusammenfassung

Das ständige Wachstum von Rechenleistung erfordert eine adäquate Expansion von Energiequellen. Da der Fortschritt Dieser jedoch nicht mit den entsprechenden Forderungen mithalten kann, sind neue Ansätze an dieser Stelle unabdinglich. Weiterentwicklungen im Bereich der energiesparenden Technologien, besonders im Internet der Dinge, gewinnen an Effizienz, während auch die Leistung designierter Hardware für Blockchain Anwendungen wächst. Eine Verbindung beider Technologien könnte eine angemessene Lösung erzeugen, die das Problem der batteriebetriebenen Geräte adressiert. Diese Bachelor-Arbeit evaluiert eine mögliche Kreuzung jener beiden Welten in Hinsicht auf Praktikabilität im Feld Internet of Things. Es wurden drei verschiedene Experimente auf Prototyp Hardware mit niedrigem Energie-Verbrauch umgesetzt, um den Strom-Verbrauch hinsichtlich der verschiedenen Szenarien zu untersuchen. Die quantitative Analyse über alle implementierten Anwendungsfälle erlaubt eine Klassifizierung von angemessenen und unmöglichen Anwendungen. Dadurch konnte gezeigt werden, dass dieser Ansatz in der Tat vielversprechend ist, dennoch weitere Verbesserung in Punkten der Hardware Umsetzung benötigt, um sein wahres Potential auszuschöpfen.

Abstract

Computational strength grows, so does its requirement for power. For the growth of energy sources that cannot keep up with its demand, new solutions to satisfy this need are urgently desired. Evolving advancements of power-saving technologies in the Internet of Things gain in efficiency, just as does the power of designated hardware for Blockchain applications. A connection of both technologies may provide a suitable solution addressing the energy demands of battery-powered devices. This Bachelor's thesis evaluates a possible intersection of both worlds for suitability in the field of the Internet of Things. Three experiments involving low-energy prototyping hardware were made to analyze the power consumption with respect to the different scenarios. A quantitative analysis of all implemented use cases allows for the classification of suitable and unsuitable applications. Thereby it is shown that this approach is promising indeed, yet requires further enhancement in hardware implementation to exhaust its true potential.

Acknowledgements

I would like to express my gratitude to my thesis supervisor Prof. Esfandiar Mohammadi of the Institute for IT Security at Universität zu Lübeck. Not only he and the ITS staff provided aid during the experiments. Prof. Mohammadi also was available for all kinds of help, even at most inconvenient times. — Thank you!
Nevertheless, I could not have written one single sentence nor a line of code in this Bachelor's thesis without the continuous support of my family and friends.
Thanks also to the team at Grammarly Inc and the Arduino community. Finally, I want to express my gratitude to one and all, who have helped me directly or indirectly.

Contents

1

Introduction

The buzzword “Scalability” is not only an extremely valuable property of modern cloud technologies, databases, and distributed systems. It can also be seen as one of the most important themes in the development of computers and computational power, in general. Latter rises since the beginning of Moore’s law(Moore et al., 1965), yet has not ceased even after the end of his predictions. Still, other urgently necessary factors aside from computational power require attention, such as energy and its efficient usage. Especially in the field of mobile computers, e.g. wireless sensor nodes, one may profit greatly from further advancements in energy-saving technologies.
For the whole topic concerning energy-saving measures neither is a new matter to society nor technology, miscellaneous efforts have taken those techniques to a new level in terms of efficiency. Especially in the field “Internet of Things”, where low-energy properties are indispensable, the evolution of energy-efficient technologies has been evolving in the course of the past years. Still, the dependence on external power sources in most scenarios may be considered a challenge for engineers, when designing new applications. Thus, satisfying the requirement of power over longer periods, than already covered, may prove to be a very profitable goal for upcoming technologies.

1.1 Problem Statement

Summary. Rising computational power increases energy consumption. New approaches that reduce the high power demands are urgently desired.

In general, modern technologies enable suitable energy sources for various applications. Thinking in great scales, nowadays miscellaneous power plants are capable of providing energy for countless facilities, utilities, and households. However, when considering power sources that are required to fit into devices of small form factors, energy supplies are limited, too. Especially in the field of IoT, i.e. where the form factor is kept very small, one faces the following challenges when considering batteries:

- *Capacity* scales with the battery its size.
- *Durability* of a single charge scales with the transmission rate.
- *Recharging* a battery usually denies the node its service.

Summarizing the three issues, one may want to conclude batteries to be a *major bottle-neck* in wireless and mobile applications. Therefore, any approach to reducing power consumption will necessarily lead to longer continuous services while preserving the battery its small form factor. This Bachelor's thesis aim is to investigate a new, yet unconventional approach to tackle these challenges and to evaluate it for practicability.

1.2 Contributions of this Thesis

Summary. Breaking the symmetry of power demands in communication may be the key to reduced energy consumption for battery-powered nodes. A new approach is required to increase energy efficiency alongside the rising power demands due to growing computational strength.

In this Bachelor's thesis a new approach is evaluated: Is it beneficial to trade computation power for battery lifetime, using the following scheme?
Consider a communication between a *sender* and a *receiver* entity. One may want to break the symmetry of computational efforts between the individuals to re-adjust their loads. This can be achieved by assigning a new task to each entity in this communication:

1. Have the *sender* hash its plaintext message, trim it down to a square root of the original length and finally send this handle to the *receiver*.
2. Let the *receiver* perform a brute force search after it receives the handle from the *sender*. Thereby the original message is re-generated.

Through exhaustive exploration of the search space by a *receiver* entity, the possibility of producing a mismatching plaintext is very low.
To estimate the proposed memory scheme its usefulness, multiple use cases will be evaluated alongside different implementations. The "LoRa" technology is chosen to be the medium concerning transmissions as the protocol itself is extremely power-efficient. Experiments are to represent the new approach its efficiency in finding a solution for the energy challenge.

1.3 Contribution Details

The *Hashing Memory Scheme* describes the idea introduced above in-depth. In the following, consider m a message represented in binary of length n. h will describe the message its hash digest and m' the recovered plain text.

Definition 1.1 (The Hashing Memory Scheme). Let there be a communication between a *sender* and a *receiver*. Assign new tasks in order to break the symmetry of computational effort between the entities:

- **Sender Task**: Store a value $m \in \{0,1\}^n$.
  1. Compute $H(m) = h \in \{0,1\}^{\sqrt{n}}$ where H is a known hash function.
  2. Store h which only needs square root memory.

– **Receiver Task**: Load value m given a handle h.

1. Run a brute force search for a given input m' in $O(2^{|h|})$.
2. Output the first preimages $m' \in \{0,1\}^{|h|^2}$ such that $H(m') = h$, with small probability that $m \neq m'$.

1.4 Related Work

Summary. For this approach is a rather unconventional attempt to tackle the energy challenge, there exist multiple complementary techniques but rather less work that is comparable to the introduced scheme.

Concerning complementary approaches, there exist techniques that already provide solutions to the general problem, which turned out to be suitable building blocks for the experiments, e.g. the LoRaWAN protocol, prototyping chips as well as efficient designated hardware and hashing algorithms. However, alternatives that may be applied before or even after the proposed scheme were not considered, despite showing quite promising results. Examples for the latter may be compression or battery-less approaches in the Internet of Things.
The first related work is the still ongoing development of low-power communication technologies in IoT. Multiple potential candidates will be evaluated (c.f. Subsection 2.1) before committing to one technology in the experiments. As already announced, LoRa and LoRaWAN were chosen due to their promising properties. Technical details will be given in Chapter 2.
Utilizing this technology, the project described in (Howerton and Schenck, 2020) is very likely to be considered a piece of related work for its implementation of LoRa(WAN) and sensors:

Example (Air Quality Measurement (Howerton and Schenck, 2020)). After the COVID-19 outbreak not only the citizens of Charlottesville changed their behavior due to the regulations. Howerton and Schenck from the University of Virginia distributed sensor kits to compare air quality before and after the outbreak. The sensors were programmed to measure particulate matter, CO_2 as well as temperature and humidity. "LoRa" was chosen to be the medium for communication between nodes and gateways. Their experiment turned out to be more expensive and less enduring than expected.
As their conclusion considers dedicated chips for the use case to be a suitable option regarding mass production, an opportunity for enhanced energy-saving mechanisms opens. New approaches as such may help the project to increase the lifetime of the nodes as well as decreasing costs (i.e. for larger batteries).

There exists a further fundamental, one very basic building block realizing the first: the development of prototyping hardware. The most famous choices may be the Arduino microcontroller family or Raspberrypi single-board computers regarding early development steps. However, there exist countless further manufacturers and architectures regarding prototyping chips. Also, for miscellaneous IoT protocols, there happen to be derivates of boards that implement certain extensions through modified hardware or

add-ons. This related topic allows for experiments around the proposed idea, thereby enabling an iterative, yet prototype-oriented approach regarding development. For not only do basic library functionalities deviate from each other but also implementations concerning APIs of the protocols do, there exist many well-documented approaches which are freely available online. A very helpful aid during prototyping was provided in various articles written by Rui and Sara Santos, such as (RandomNerdTutorials.com, 2021).
Classic sensor nodes, concerning most IoT solutions, have in common that they depend on batteries. Nevertheless, there exists a very relevant, yet complementary work presented in (Peng et al., 2018) that does, in fact, not depend on batteries:

Example (Battery-Less LoRa (Peng et al., 2018)). In contrast to the previous examples (also including the experiments in this Bachelor's thesis), a new approach concerning energy-efficient communication using "LoRa" was developed that does not require a battery to be attached to the node. Through the collection of ambient backscatter (i.e. ambient "LoRa" transmissions), a so-called "PLoRa-tag" collects the energy required for transmission. Experiments have shown that sending 284 Bytes demands the tag to charge for 17 minutes.
As the proposed memory scheme creates the opportunity to shrink a transmission of its payload, the "PLoRa" experiment may be enhanced even further. This could either cause the payload to gain in length or the charging time to decrease.

Regarding the "shortening"-part of the proposed memory scheme, there also exist complementary approaches. In information theory, there are various kinds of algorithms capable of compressing data. Well-known examples may be Run Length Encoding, Huffman Encoding, the Shannon-Fano-Algorithm, Arithmetic Encoding, and the Lempel-Ziv-Welch-Algorithm. In this Bachelor's thesis, compression as such is not evaluated any further despite sharing the same aim of reducing energy demands by increasing computational efforts. This topic, including efficient coding schemes for specific applications, may be potential future work when considered an extension to the proposed memory scheme.
The ultimate related theme is designated hardware. There exist not only many distinct applications requiring special solutions as such, but also different motivations. Regarding this Bachelor's thesis attempt, two motives are of interest concerning the different entities (c.f. Subsection 1.2). The first incentive is the most common, i.e. *speed* in the context of computational strength. As proposed, a "Strong Receiver" entity is required to perform computationally complex tasks in a high-speed manner. Hardware realizing this kind of function is sophisticated super-computers, spoken generally. Nowadays, high-end machines as such can be found not only in scientific facilities but are also available for purchase (e.g. online) due to the ongoing rise of computational power. The second incentive is increasing in popularity as well: *Low-Power* hardware. For the "Weak Sender" entity is not passive at all, computational tasks require handling on battery-powered devices, as well. Due to the limited availability of energy resources concerning machines of a small form factor, this property is a challenge in the field of chip design. The corresponding application found in this Bachelor's thesis is the cryptographic chip realizing hashing algorithms in a performant (i.e. fast) manner while retaining the low-energy profile.

1.5 Structure of this Thesis

Summary. First, the preliminaries are pointed out, necessary definitions will be given and decisions are made. Secondary the implementations and experiments are described, their results will be evaluated. Finally, a conclusion is drawn and an outlook will be given.

After reasoning about the urgency of new approaches dealing with the increasing power demands, this thesis its approach is described. In Chapter 2, “Preliminaries”, information regarding fundamental building blocks will be provided. Also, that chapter will assist in backing the decisions taken, regarding the experimental prototypes. With respect to the choice of experiments, Chapter 3 introduces the tools (i.e. program code) that were used for measurement explained in Chapter 4, “Experiments”. This chapter will not only provide an overview of the approach its performance regarding the three distinct applications but also attempts to interpret the outcomes by outlining multiple learnings. To provide a closure concerning the topics introduced, experiments evaluated and learnings found, a conclusion will give an overview as well as an outlook on the field and the proposed idea its potential impact.

2

Preliminaries

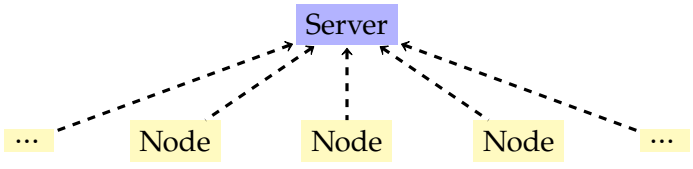
Before introducing the actual experiments, a few more explanations and definitions are required. First the whole topic "Internet of Things" is introduced as it is the thesis its field of application. This section includes an overview of multiple options of technologies considering wireless transmissions. Afterward, further insights on the thesis its contribution are provided. As a new scheme of communication is introduced, the participating entities are described with their responsibilities concerning their relationship. Finally, a multitude of *classic* use cases considering sensor nodes in IoT will be evaluated whether the proposed memory scheme is expected to be beneficial nor not.

2.1 The Internet of Things

Summary. Three core features that are important for the proposed memory scheme will be focussed on: The "Star Topology", "Duty Cycle" and miscellaneous technologies in IoT.

Since 1990, when John Romkey first connected his toaster to a network, in fact, the first "Thing", the whole area of the "Internet of Things" has grown by leaps and bounds. According to the dictionary, the Internet of Things is a "network of everyday devices, appliances, and other objects equipped with computer chips and sensors that can collect and transmit data through the internet."(dictionary.com, n.d.). The "Things" mentioned, describe a large set of devices, e.g. a sensor node. As a single end-point (i.e. a sensor) is less common to be the only non-server node in the network, larger structures require a kind of organization. That is the network its *topology*. A very common solution for structuring a network as such, is the "Star Topology":

Definition 2.1 (Star Topology). The star topology (also known as star network) consists of a central node, that is the server (also: network server, gateway, central node), and multiple surrounding nodes (here: sensor nodes).

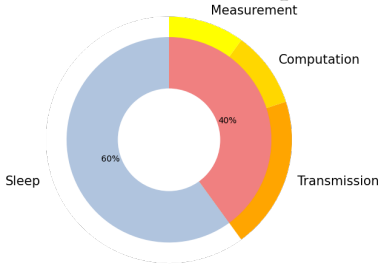


Large sensor networks in the field of IoT profit greatly from the robust structure provided by the star topology. In case a single sensor node fails or its link to the server breaks, no other entity of the network is affected at all.
The following fundamental definition describes a (sensor-) node its ability to circulate through a so-called "Duty Cycle":

Definition 2.2 (Duty Cycle). IoT nodes can switch between two modes: When in *active* mode, an end-device can perform actions such as taking measurements or handling transmissions. In contrast, when in *passive* (alternatively "sleep") mode, the node is basically turned off, either for a specified period or until an external interrupt occurs. As the energy consumption of both modes is fundamentally different, architects in the field of IoT strive towards finding the optimal ratio between them during a period. The completion of one passive, as well as one active time slice, is called the duty cycle.

The experimental prototypes described in the succeeding chapter adapt the following exemplary cycle:

Example 2.3 (Duty Cycle). This duty cycle consists of a node staying three minutes in sleep and two minutes in active mode. Here the active slice also contains sub-slices that are measurement, computation and transmission.



1. Active, measure using a sensor.
2. Active, process the measurements.
3. Active, transmit the computed values.
4. Passive, sleep until wake-up.

IoT Technologies

Summary. There exist countless protocol technologies in the field of IoT. Suitable candidates for this thesis are required to support long-range transmissions but do not depend on over-average data rates.

By choosing suitable technology for the proposed memory scheme its application in wireless sensor nodes, one may want to consider two basic criteria.

Criterion (Long Range). *When using a* wireless *sensor node, a user may want to maximize the distance separating sender and receiver. Otherwise, the need for wireless transmissions* (*e.g. concerning wireless sensors*) *can be neglected and a wired connection may be of better use.*

Criterion (Low Rate). *As data rate* (*here: the number of messages per temporal unit*) *scales with the length of messages, one does not require messages of large sizes, for the required computational power of the receiver is limited* (*c.f. Figure 2.2*).

For practical considerations there were a few additional but non-trivial criteria, such as availability of prototyping hardware, license model of the protocols, implementation costs and basic energy efficiency, too.
Figure 2.1 provides an overview over current state-of-the-art IoT protocols and their properties concerning the introduced criteria.

Technology	max. Range	max. Data Rate	Basis
NFC	10 cm	420 kbps	ISO/IEC 18000-3
Z wave	30 m	50 Kbps	Z-wave Alliance
Zigbee	100 m	250 kbps	Zigbee 3.0
WiFi	100 m	1 Gbps	IEEE 802.11-ac
Bluetooth	150 m	1 Mbps	Bluetooth 4.2 core specification
LoRaWAN	15 km	50 Kbps	LoRaWAN
SigFox	50 km	1 kbps	Sigfox
Cellular	200 km	170 kbps	UMTS/HSPA

Figure 2.1: A selection of communication technologies used in IoT. (Sonee, n.d.)

Summary. Most protocols either violate the first criterion or over-kill the second. However, a suitable technology may be LoRa and LoRaWAN.

As for criterion one, the first five protocols come with insufficient range. The last two technologies turned out to be unsuitable, considering the attempt of creating a prototype. Nevertheless, examples such as Bluetooth or WiFi are no suitable prototypes either, as their power consumption is incomparably high. Thus LoRaWAN is the choice for the experiments in this thesis.
LoRaWAN, which is currently developed by the "LoRa Alliance", was first introduced by SemTech. The latest specification was released in October 2020 as version 1.0.4.(Alliance, 2020). Using the "Chirp Spread Spectrum", LoRa transmitters are capable of sending packets, i.e. releasing so-called "Chirps" into the air such that a recipient can receive this message. CSS supports the following frequency bands in different regions(Vangelista, Zanella, and Zorzi, 2015):

$$\overbrace{169MHz,\ 433MHz,\ 433MHz}^{\text{USA}} \text{ and } \overbrace{868MHz}^{\text{Europe}}$$

In addition to the proposed features of LoRa and LoRaWAN, including the long communication range and data rate, also security characteristics are important to consider. (Yang et al., 2018) explains how the LoRaWAN standard covers the common assets of the "CIA-triad": Confidentiality, Integrity, and Authenticity. Despite the existence of miscellaneous attacks on those security primitives, such as "Replay through Counter Overflow" or even "Self-Replay through Selective Jamming", applying the proposed memory scheme does in fact *not* add any security issues to the setup.
Justified by the appealing features mentioned above, the experimental setup will apply LoRa(WAN) utilizing the $868MHz$-frequency in a node-to-node scenario.

2.2 "Weak Sender — Strong Receiver" Principle

Summary. After defining broad requirements for technology in a network, all entities require specifications in detail. The proposed memory scheme its fundamental principle will be defined as well as its parts.

As announced in Section 1.3, the Bachelor's thesis approach to decrease energy requirements is to apply the introduced memory scheme. To enable this new concept, the following principle is required as a basis for the whole concept:

Definition 2.4 ("Weak Sender – Strong Receiver" Principle). As the server end-point (c.f. Definition 2.1) requires a connection to all sensor nodes in the star network, one can consider it the "Strong Receiver". An endpoint such as a sensor node, however, is not required to manage responsibilities, unlike the server node. Therefore sensor nodes are considered "Weak Senders" in this principle.

Strong Receivers

Even before the boom of blockchains, designated hardware for cryptographic purposes was a serious matter. Aside from the rare capability of reversing cryptographic one-way functions using brute-force search for plain text messages up to a fixed length, cracking hashes is the basis for the "Mining" concept of cryptocurrencies. As the value of those currencies grows, the "Mining Reward", that is the compensation of *breaking* a hashed block, increases, too. Due to the rising difficulty of that certain task, the request for hardware that can keep up with the required efforts increases, too. Thereby more and more powerful chips are being created, leading to the development of mighty ASIC-miners. An ASIC-miner its purpose is to generate cryptocurrencies by joining a so-called pool, that is a connection between multiple "miners", working in a kind of team to break the most recent hash-block.
A piece of designated hardware as such is most-suitable as a "Strong Receiver". The greater its computational strength, the longer the messages transmitted by a node may grow. A piece of mining-hardware is characterized not only by its energy footprint but foremost by its hash-rate:

Definition 2.5 (Hash-Rate). The hash rate is defined by number of hashes that can be generated in a specified time frame. The most commonly used unit is H/s, that is the number of computable hashes per second. Further descriptions and examples use the following unit conversion:

$$\underset{\text{Tera-H/s}}{1\frac{TH}{s}} = \underset{\text{Giga-H/s}}{10^3\frac{GH}{s}} = \underset{\text{Mega-H/s}}{10^6\frac{MH}{s}} = \underset{\text{Kilo-H/s}}{10^9\frac{kH}{s}} = \underset{\text{H/s}}{10^{12}\frac{H}{s}} \tag{2.1}$$

Not only pieces of designated hardware can compute hash digests for plain text messages, but also graphic processing units and some central processing units are capable of reaching decent hash rates. Still, there is a major gap in comparison between ASIC-miners and GPUs. Figure 2.2 displays current developments of ASIC-miners.
Considering machines of great strength also means great power consumption for the strong receiver, as shown on the x axis of Figure 2.2 as well. To set a reasonable upper bound regarding the required computational power consider the following:

Fact 2.6. A byte, consisting of eight bits, is the smallest data type a transmitter can send in a package, independent from protocol and technology.

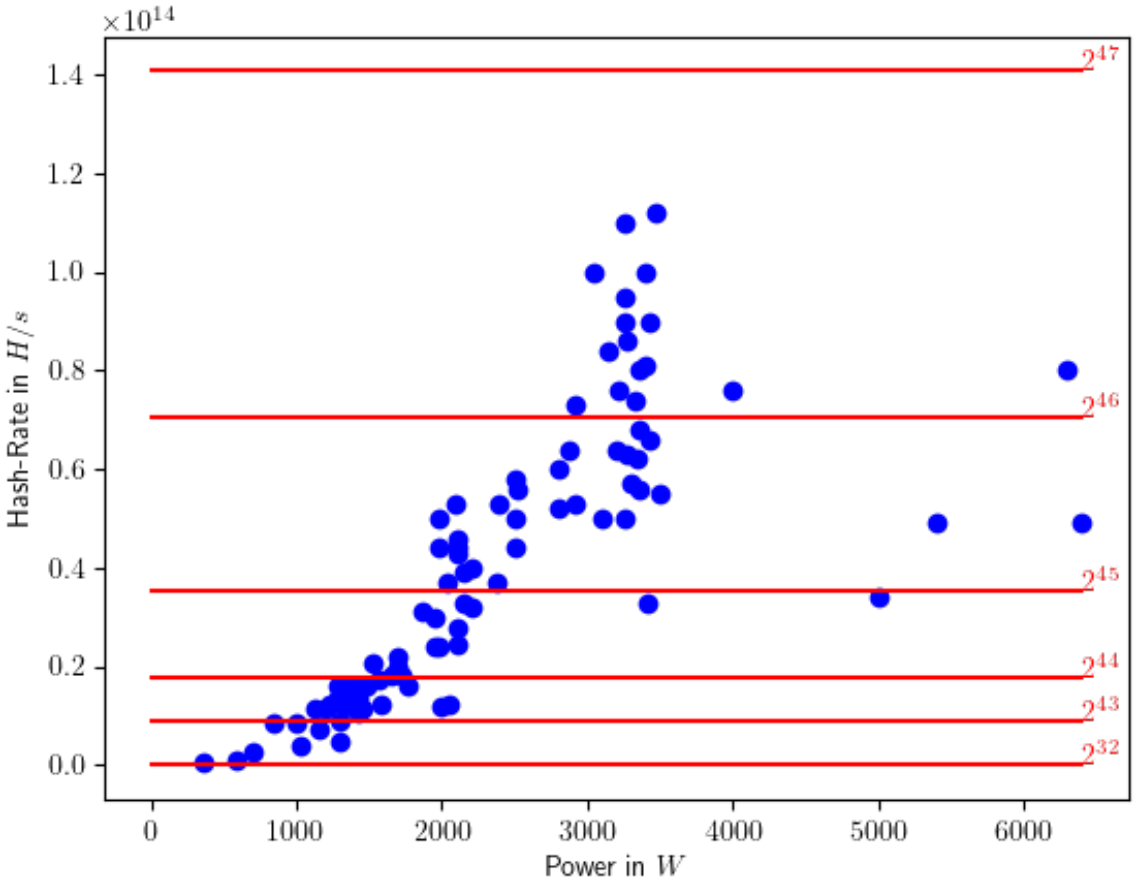


Figure 2.2: Comparison of ASIC-miners using data from (Value, 2021)

Respecting Fact 2.6 the smallest packet has a length of eight bits. That means, when applying the proposed memory scheme, one could transport 64 bits of information in a single transmission. Unfortunately, as displayed in Figure 2.2, not a single miner from the ranking can fulfill the task of covering a search space of 2^{64} bits in one second. Therefore, neglecting Fact 2.6, transmissions will waste bits of information that could be used if a much stronger device existed.
The second effect of Fact 2.6 is the lower bound for a strong receiver:

Theorem 2.7 (Lower Bound for Strong Receivers). *The lower bound for computational strength of a strong receiver is* $2^{32}\frac{H}{s}$.

Proof. Apply Equation 2.1

$$2^{32}\frac{H}{s} \approx 4.3 \times 10^{9}\frac{H}{s} = 4.3\frac{MH}{s}$$

□

The proposed strength shown in Figure 2.2 matches the requirement. Thus designated hardware such as an "ASIC Miner" are suitable strong receivers.

Weak Senders

In contrast to the proposed mining hardware that can break given hashes, the devices considered "Weak Senders" can be mere prototyping boards. Despite the valid scenarios where those chips are used in practice, boards such as the "ESP32" chip also are a solution for experiments where requirements may change during the process.
The "ESP32" microcontroller is manufactured by "Espressif Systems" since 2016 and is the successor of the "ESP8266" chip. Powered by two microprocessors with low power requirements, clock frequencies up to 240MHz can be established(Systems, 2016). The "ESP32" supports various variants of WiFi and Bluetooth. There exist solutions and derivates that support even more protocols, such as the "WiFi LoRa 32 (V2)" by Heltec Automation. Furthermore, there is also a great selection of additional hardware, e.g. camera modules, displays, etc.

Remark 2.8. The main reason choosing an "ESP32" board as a prototype is the embedded cryptographic co-processor that supports the SHA-2 hashing algorithm.

Mode	Consumption	Description
Active	up to 260mA	Communication is enabled
Modem-Sleep	up to 20mA	CPU is enabled
Light-Sleep	0.8mA	
Deep-Sleep	0.15mA	CPU is disabled
Hiberation	2.5μA	RTC is enabled

Figure 2.3: ESP32 power modes and their current consumption(Systems, 2016)

The power consumption of the ESP32, the basic version of this chip (i.e. without LoRa capabilities, display) in five different steps is described in Figure 2.3.
Due to the low energy footprint of this microcontroller as well as the fast and low-power co-processor, "ESP32" boards are used in the experiments.

2.3 Choice of Experiments

Use Cases in IoT

Summary. In the "Internet of Things" not only many solutions exist but also countless use cases for the new technologies. Due to multiple limitations, only a small subset of applications are potential candidates that may profit from the proposed memory scheme, such as sensors.

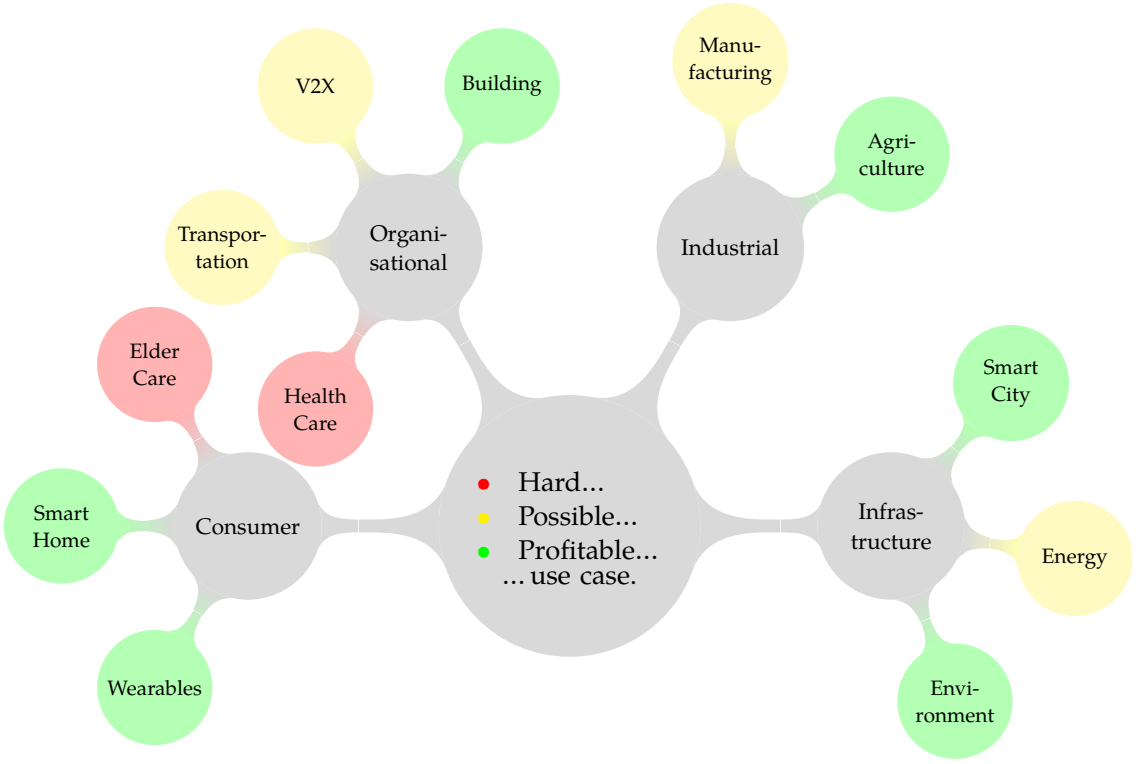
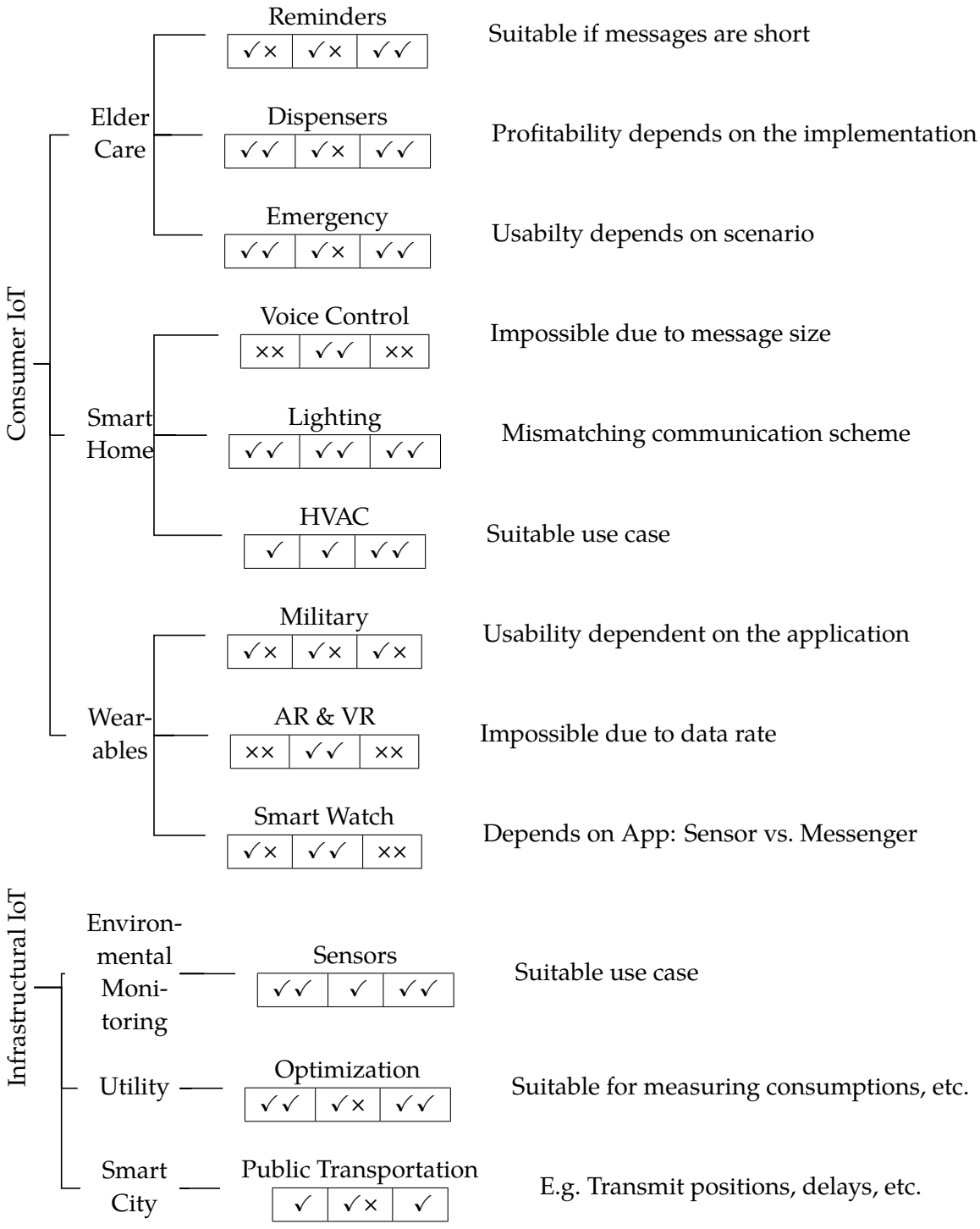
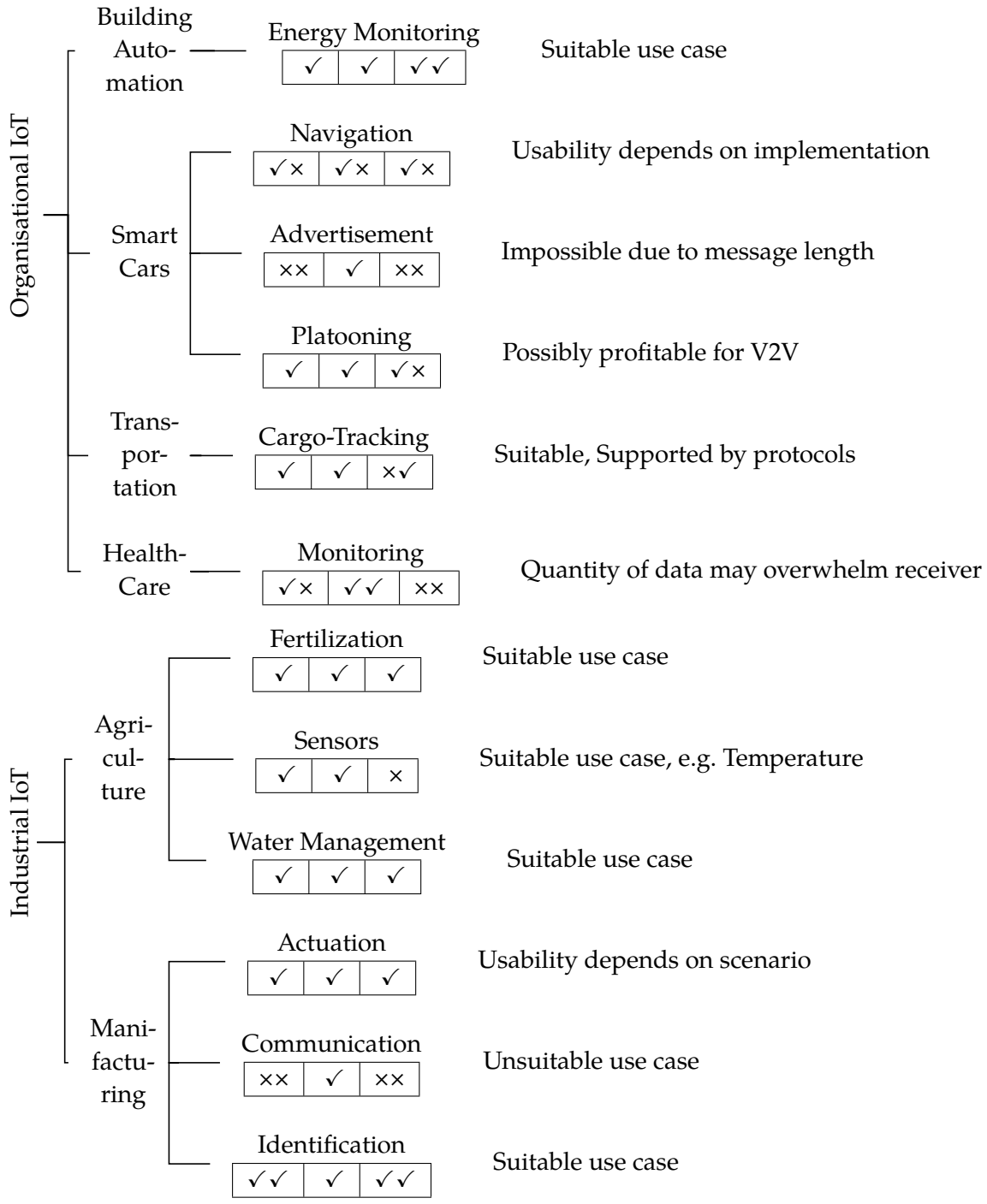


Figure 2.4: Selection of Use Cases in the Internet of Things.

The many different applications in IoT may be grouped into Consumer-, Organisational-, Industrial- and Infrastructural IoT and more categories. Each field has very characteristic applications, which again have different scenarios where IoT may be applied. A selection of use cases will be evaluated according to three criteria, which are Message-Length, Communication-Range, and required Bandwidth. An overview is provided in Figure 2.4. The following trees describe the single non-leaf nodes in the diagram, including their use case categories with examples and an evaluation on each. Every application example is annotated due to three criteria (Message Length, Range Requirements, Bandwidth Requirements) as follows:

	✓✓	✓	✓×	××
Requirement is	Over-Met	Satisfied	Conditionally Met	Unsatisfied


Consumer IoT
Elder Care
Reminders
✓× ✓× ✓✓
Suitable if messages are short
Dispensers
✓✓ ✓× ✓✓
Profitability depends on the implementation
Emergency
✓✓ ✓× ✓✓
Usabilty depends on scenario
Smart Home
Voice Control
×× ✓✓ ××
Impossible due to message size
Lighting
✓✓ ✓✓ ✓✓
Mismatching communication scheme
HVAC
✓ ✓ ✓✓
Suitable use case
Wear-ables
Military
✓× ✓× ✓×
Usability dependent on the application
AR & VR
×× ✓✓ ××
Impossible due to data rate
Smart Watch
✓× ✓✓ ××
Depends on App: Sensor vs. Messenger
Infrastructural IoT
Environ-mental Moni-toring
Sensors
✓✓ ✓ ✓✓
Suitable use case
Utility
Optimization
✓✓ ✓× ✓✓
Suitable for measuring consumptions, etc.
Smart City
Public Transportation
✓ ✓× ✓
E.g. Transmit positions, delays, etc.



Conclusion

The most suitable use cases having most of the criteria satisfied at least are sensor nodes. Therefore the experiments will respect this as a guideline and measurement nodes will be evaluated.

2.4 Physical Considerations

Summary. Before introducing the experiments, a basic block of great importance for this thesis requires an introduction. That is the splitting of total energy into multiple parts according to a duty cycle, as proposed in (Bouguera et al., 2018).

As proposed in Definition 2.2, a node can be in different states. Therefore current as well as time are different in these states. In (Bouguera et al., 2018) there is a differentiation of portions concerning the total energy E_{total} introduced that will be used as base for all computations concerning energy of sensors in this thesis.

Definition 2.9 (Total Energy of a Sensor Node(Bouguera et al., 2018)). Let E_{total} be the total energy that a sensor node uses in a single "run" (that is one iteration of a specified duty cycle). E_{total} consists of two basic parts, that are E_{active} and E_{passive}.

$$E_{\text{total}} = E_{\text{passive}} + E_{\text{active}} \tag{2.2}$$

Through application of the physical fundamental on E_{passive} in Equation 2.2 one obtains:

$$E_{\text{total}} = P_{\text{passive}} T_{\text{passive}} + E_{\text{active}} \tag{2.3}$$

Instead of applying it again, (Bouguera et al., 2018) suggests to divide E_{active} in multiple parts. That is the key point:

$$E_{\text{active}} = E_{\text{WU}} + E_{\text{M}} + E_{\text{Proc}} + E_{\text{WUT}} + E_{\text{Tr}} + E_{\text{R}} \tag{2.4}$$

Therefore E_{active} consists of device wake-up energy E_{WU}, energy during measurement E_{M}, power per time concerning the measurements their processing E_{Proc}, E_{WUT} that is the energy consumed during transceiver wake-up, E_{Tr} and E_{R} that are energies during transmission and reception respectively.

Thus the different phases that require measurement are: E_{WU}, E_{M}, E_{Proc}, E_{WUT} and E_{Tr}. E_{R} will be neglected as behavior concerning the reception of a sensor node is not implemented in the experiments.

3

Implementation and Approach

Recalling Section 1.2 and Definition 1.1, the challenge now is to develop an actual software implementation for the introduced transformations and transportation. Both are pictured in Figure 3.1 with respect to the single steps. Before explaining the experimental setup, the prototypes require a first introduction, this primarily concerns the "Sender Task" (c.f. Figure 3.1 left). Preparations for the environment will be explained, followed by instructions on how to set up the boards. Currently three different basic sketches for sensors can be deployed:

- Hall Sensor: Measures intensities of magnetic fields (here: "hall")
- MQ2 Sensor: Measures intensities of gas (here: "gas" or "MQ2")
- Touch Sensor: Measures intensities of physical touch (here: "touch")

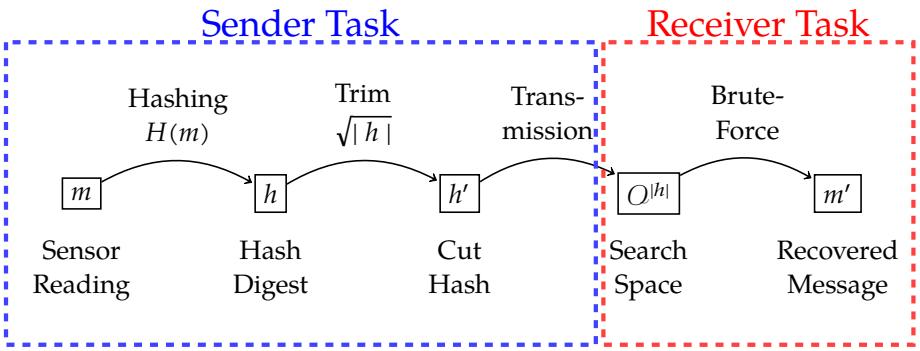


Figure 3.1: Data Transformation and Transportation in the Hashing Scheme

3.1 Preparations

To establish a well functioning interface between the via USB connected board and the code itself one would want to choose the Arduino IDE that is available for download from the official site: `www.arduino.cc`. Depending on the platform, very specific actions have to be taking in order of setting up the IDE properly (e.g. proper setting of access rights

to device). The correct library corresponding to the used microcontroller is available at `https://github.com/HelTecAutomation/Heltec_ESP32`. After setting up IDE and correct libraries all software preparations are done.

Remark. Every time a new sketch is flashed to the connected microcontroller a reboot into the so-called flashing mode is required. This is handled automatically via the `Heltec_ESP32` library.

All steps are also documented in the corresponding `README` file in (Widmer, 2021).

	Hall Sensor Node	**Gas Sensor Node**	**Touch Sensor Node**
`sleep` in seconds	Default is 1	Default is 1	Default is 1
`scalar`	Default is 1	Default is 1	Default is 1
`gas_pin`	Unused	Default is 37	Unused
`touch_pin`	Unused	Unused	Default is 12

Figure 3.2: Parameters for the Makefile to pre-seed the generated code

3.2 Code Generation and Deployment

To ease the deployment of a sensor node that utilizes the hashing memory scheme, a Makefile is provided in (Widmer, 2021) that generates a proper source code file depending on parameters for a chosen use case.
There exist three different relevant targets that enable the code generation: hall_sketch, gas_sketch and touch_sketch. When invoking the make command on the desired target one can optionally set the parameters, specified in Figure 3.2.

Example 3.1 (Touch Sensor Source Code Generation). To create the appropriate source code file for the touch sensor, that implements a sleep duration of three seconds, ignores scaling completely and uses the twelfth on-board GPIO pin, one may want to invoke the following shell code:

```
make touch_sketch \
        sleep=3 \
        scalar=1 \
        touch_pin=12
```

Any kind of successful building will result in a newly created file that can be found in ./sketch/sketch.ino.

Remark. The current version of the Arduino IDE (June 2021) requires each sketch to be placed into a separate directory. Thus the Makefile makes use of an extra folder, as recommended.

After generating the code one may want to manually open the ./sketch/sketch.ino file in the IDE to flash it to the board.

3.3 Hardware Extension

Except for the implementation of the gas sensor node, there is no further hardware urgently required. However, the touch sensor node may be extended by attaching a wire (e.g. a jump wire with a solid tip) to enhance the actual measurements.
When deploying the gas sensor node one additionally requires three jump wires and the MQ2 sensor module. This extension may be applied according to Figure 3.3.

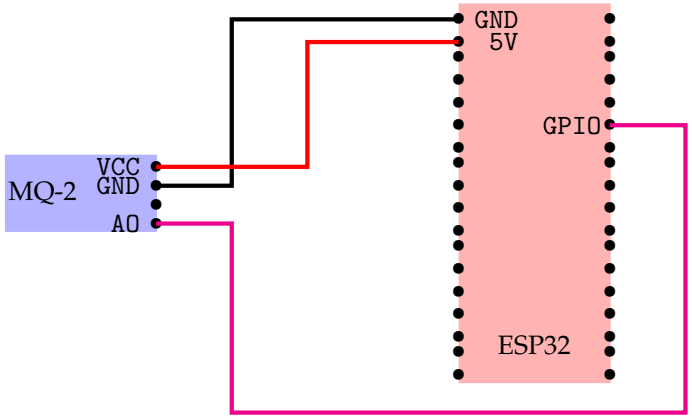


Figure 3.3: Suggested additional Wiring for the MQ-2 sensor

3.4 Actual Program

The code that may be generated using the tools provided in (Widmer, 2021) establishes the base for the measurement process described in Chapter 4. For detailed information on the single steps taken, please refer to the documented source code. Once the properly flashed microcontroller is booted up, it will enter the first state, described in Figure 3.4. Using the corresponding sensor that has been set up, the board will take the measurements, apply the hashing procedure to it using the onboard cryptographic processor. Afterward, the generated data will be transmitted using the LoRa protocol alongside the original data (i.e. that has neither been hashed nor cut). The protocol allows for a sender to just release the signal until a receiver eventually captures it (c.f. Subsection 2.1). This property of LoRa enables faster prototyping and measurement, for the sensor node can (e.g. during experiments) set to be agnostic towards a potential receiver. During any relevant process, the board will measure the times that those steps take to perform. This data can be captured and will prove useful for the experiments later on.

Remark. The sketches provided, contain the previously mentioned redundancy of sending via both schemes by intention. An implementation that is ready for any kind of production scenario would not only require removing this redundancy but also to test and refactor the implementation by industrial standards.

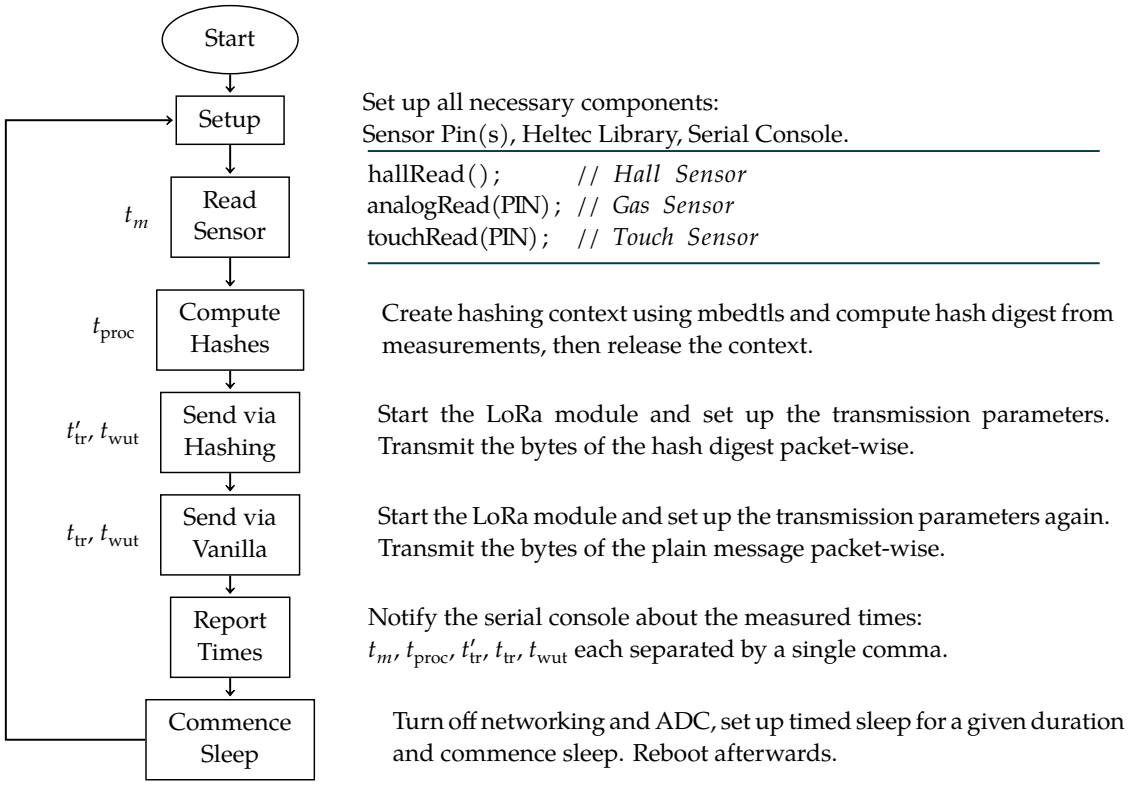


Figure 3.4: Flow Chart of the Sketch

3.5 Trouble Shooting

During setup and experiments, multiple issues were faced regarding the individual sensors. This section provides comments on multiple situations that one may consider helpful when facing similar hurdles during implementation.
First and foremost, as already explained in Fact 2.6 the current implementation is limited by neglecting any additional coding schemes such as compression. Thereby the results degrade, compared to theoretical expectations. This hurdle may provide a suitable starting point for future work, as it potentially allows for far better efficiency when using appropriate encoding.
There may be further issues regarding the hardware setup. For manufacturers provide suitable *sketches* that allow for sanity checks on a new microcontroller, one may be enabled to exclude an error during production easily. Those codes are available online, provided by the corresponding manufacturer, e.g. linked to public repositories.
Nevertheless, due to critical disturbances during the flashing procedure, individual boards or components may break (or even brick). It is advised to avoid scenarios such as that. "Bricking" was observed during the experiments only on one out of three boards that were of the same kind and manufacturer. However, in most cases, the issue can be resolved through performing multiple resets, reboots, and bootloader re-flashes.
There were no known issues that happened during experiments with the Hall sensor node. Concerning the touch application, one may face the issue of not receiving any valid

signal. Despite the working prototype of this Bachelor's thesis, utilizing GPIO pin 12, one may want to refer to the pin-out-diagrams and data sheets provided by the manufacturer on `heltec.org/project/wifi-lora-32`.
Due to the additional electronic component used in the experiment capturing gas intensities, i.e. the MQ2 sensor, one may face additional issues here. During the experiments and their preparation, it turned out to be very useful, checking the sensor with a minimal working sketch for correct functioning using a common lighter. The physical knob attached to the sensor module can be used for fine-tuning, e.g. to set an appropriate threshold. However, during intended usage, one may want to avoid high intensities (c.f. the lighter example) as this exposure *can* actually bias the sensor readings until power-off.
Due to the inexhaustible number of software issues that may occur, no complete guide can be provided. Nevertheless, there are countless resources available online, that may assist in resolving hurdles, extending the code, and customization. There exist numerous tutorials and communities (concerning both, hard- and software development) having experienced members, providing help regarding problems during development.

4

Experiments

Summary. Three experiments with different sensors were made. Their performances concerning energy consumption are evaluated for the classic as well as for the proposed memory scheme. Under special conditions, large portions of power can be saved.

After evaluating a large number of different applications of IoT in Subsection 2.3, the conclusion states that sensors are the devices that are expected to perform best when applying the hashing memory scheme. As already mentioned in Chapter 3, three sensors are chosen: a Hall sensor, the MQ2 sensor that measures the air quality, and a touch sensor.
At first, the experimental approach will be described, how the energy savings are measured and computed. Afterward, the three experiments will be described in detail, including their actual performance. To conclude this chapter a comparison between the applications will be given.

4.1 Approach

Summary. To compute the differences in energy consumption of the experiments, both, time and current measurements, are required. In combining them later on, one can see the difference the proposed memory scheme can make. Measurements will consider different phases of a duty cycle using distinct parameters.

As proposed, two different kinds of measurement are required for the computation of the savings: the current consumed by the prototype and the time different steps in the duty cycle take. From physical basics, it is known that energy (in Watt-hours) is the product of time (in seconds) and current (in Ampere). Therefore both are urgently required to measure.
While the current measurement is performed by taking single measurements with an appropriate measurement device, the capturing of durations happens in a *quantitative* approach. Concerning time, one may find this approach appropriate, as the number of data collected for each step is ten thousand. Thus the *standard error* can be approximated as follows:

$$\frac{\sigma}{\sqrt{n}} = \frac{\sigma}{\sqrt{10,000}} = \frac{\sigma}{100}$$

Therefore the probability of the measurement to be erroneous is about 0.01, which one may consider comparably small or even rather negligible.

Time Measurement

Summary. Time measurement will be handled as follows: The programmed microcontroller outputs durations of single processes in a duty cycle while a script on an external computer records this data.

To measure the five different durations of the specified duty cycle (that are T_{WU}, T_{M}, T_{Proc}, T_{WUT} and T_{Tr}), the microcontroller is programmed through the tutorial provided in (Widmer, 2021) (c.f. Chapter 3).
After installing the firmware as well as the additional hardware to the board, a script on an external machine can capture the data to analyze it later on. All required software can be set up appropriately by following the provided instructions.
During the time measurement process, the microcontroller performs the loop described in Figure 3.4 – that is the server-side program. The single steps in the flow chart perform the functions expected from a sensor node with one exception: Both memory schemes are executed sequentially. That means that first the data is transmitted utilizing the hashing memory scheme and sent using the vanilla memory scheme. During the analysis of the captured data, the corresponding values will be handled separately.
The script running in parallel (e.g. on an external machine) is described in Figure 4.1 –

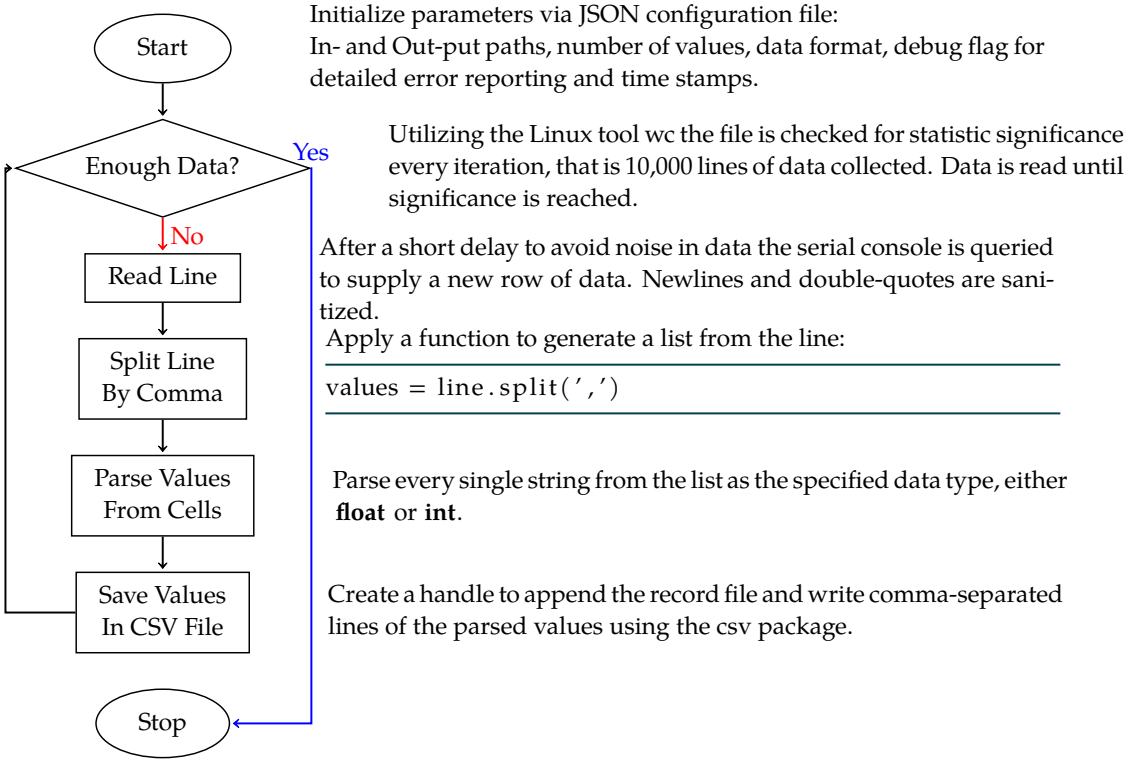


Figure 4.1: General Time Measurement: Client Side

that is the client-side program. Data is not only captured by executing the appropriate

script but also converted into the correct number formats and saved to comma-separated-value files (i.e. CSV format).
To achieve convincing results, the measurement process is executed until ten thousand data rows are captured. Concrete results will be provided in the following sections.

Current Measurement

Summary. Current measurement requires a second firmware that executes the different steps of the duty cycle sequentially for a longer time duration. During this time, a multimeter device is attached to the board and actual currents are measured.

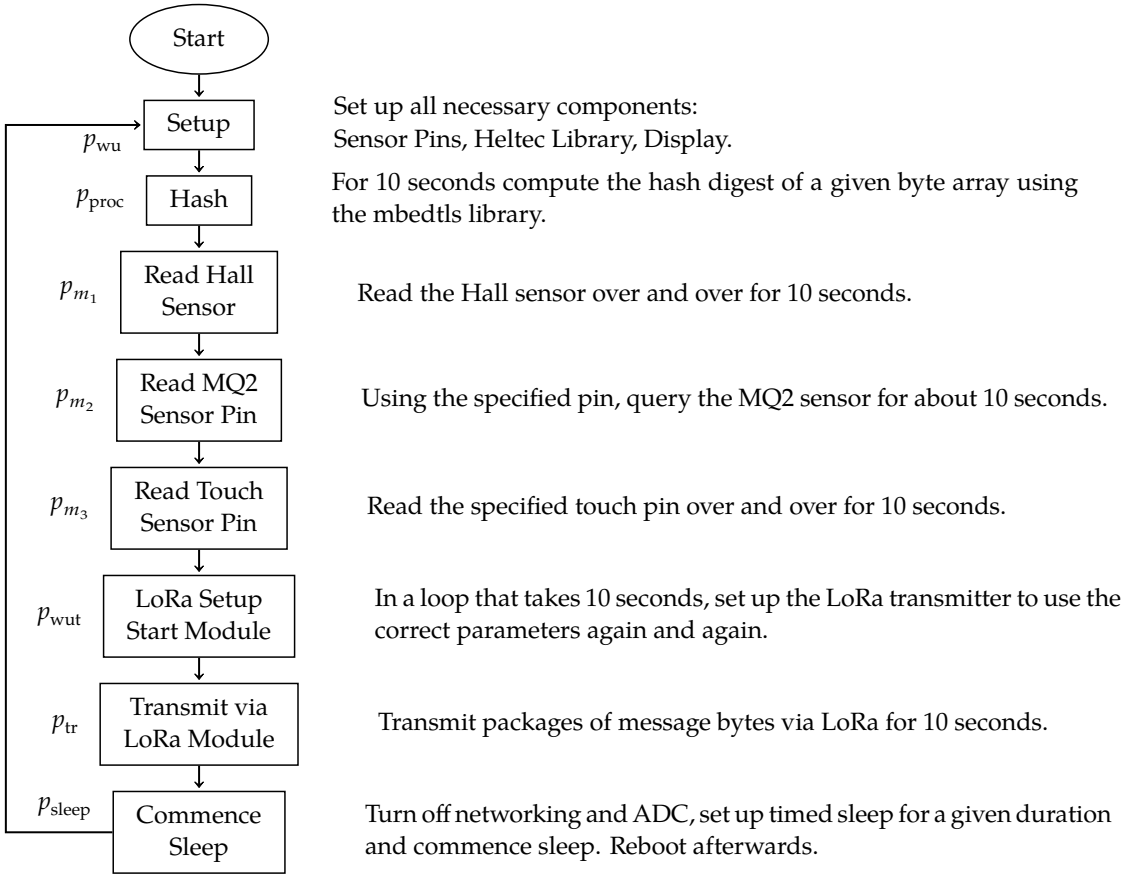


Figure 4.2: Current Measurement using the ESP32 Micro Controller

After describing the quantitative process of capturing data concerning timing behaviour of the microcontroller also the currents drawn require measurement. Another approach is used here: A special firmware (that can be found in (Widmer, 2021)) is programmed to the ESP32 board and a multimeter is used. The flow chart given in Figure 4.2 shows the behaviour of that special firmware. Single steps are executed over longer durations. Using the instructions, one can reproduce the whole set up for this measurement approach. A basic schematic for the system is given in Figure 4.3.

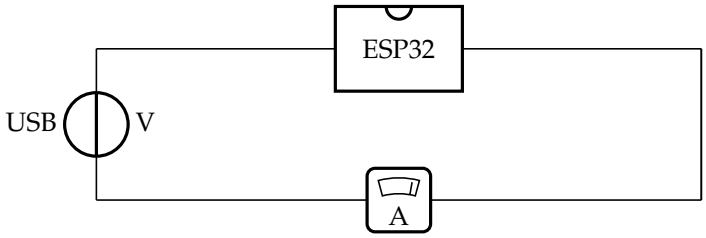


Figure 4.3: Schematic for Current Measurement

4.2 Hall Sensor Node

Summary. Deploying the Hall-effect sensor does not require additional hardware but can save up to 50.69 percent of the energy the vanilla scheme uses.

The first experiment implementing the proposed memory scheme considers a very basic measurement: A Hall-effect sensor. The Hall sensor measures the magnitude of a magnetic field utilizing the Hall-effect. This use case is chosen as a first example for the sensor itself is embedded into the ESP32 chip. Thus neither additional wires nor further hardware is required to deploy the first sensor node.

Measurements

Times in μs			
T_{WU}		261077	
T_{M}	259	2057	20112
T_{Proc}	225	632	4700
T_{WUT}		43	
T_{Tr}	413988	672211	3562564
$T_{\text{Tr}}{}'$	521764	2136411	8378964

Currents in mA	
P_{Sleep}	56.4
P_{WU}	56.4
P_{M}	63.6
P_{Proc}	76.7
P_{WUT}	70.4
P_{Tr}	188.7

Figure 4.4: Measured values during the Hall Sensor Node Experiment

Measurements were taken according to the introduced method from Section 4.1: A generated firmware is flashed to the microcontroller, afterward the "listener script" captures measured timing behavior. Also, current measurements are taken by the described method. Average measurement results are described in Figure 4.4. The three columns in the table corresponding to the times that were measured. In the first column, no *scaling* was applied but in the second there were ten loads of measurement taken at once, in the third column hundred.

Evaluation

Through application of the computation scheme introduced in Definition 2.9 one can calculate concrete values over the average values given in Figure 4.4 for both memory

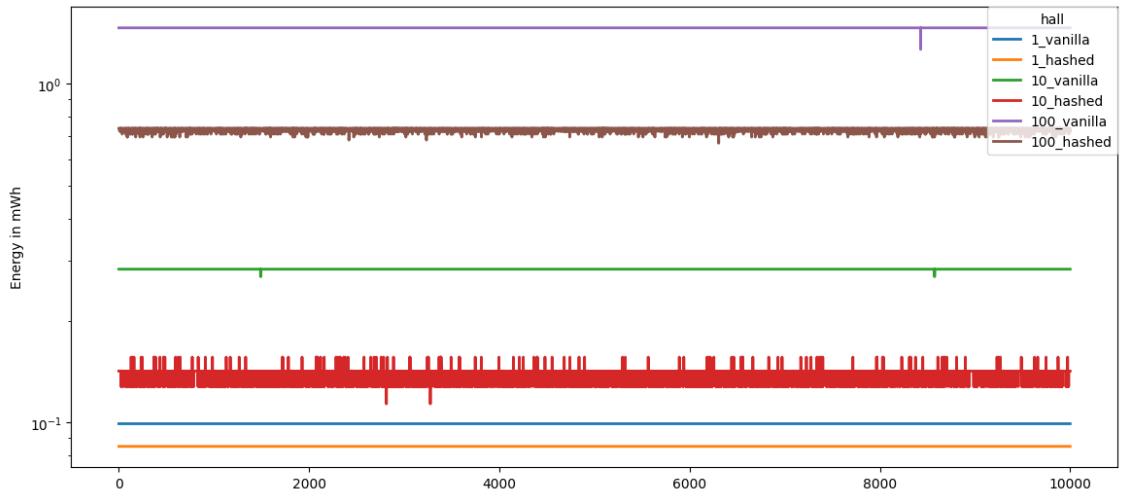


Figure 4.5: Plot of Captured Data aggregated with Measured Power and Voltage.

schemes.

$$\begin{aligned} E_{\text{total}} &= E_{\text{WU}} + E_{\text{M}} + E_{\text{Proc}} + E_{\text{WUT}} + E_{\text{Tr}} \\ &= P_{\text{WU}}T_{\text{WU}} + P_{\text{M}}T_{\text{M}} + P_{\text{Proc}}T_{\text{Proc}} + P_{\text{WUT}}T_{\text{WUT}} + P_{\text{Tr}}T_{\text{Tr}} \end{aligned} \tag{4.1}$$

- Scaling 1: $E_{\text{Hashing}} = 0.085mWh$ and $E_{\text{Vanilla}} = 0.099mWh$
- Scaling 10: $E_{\text{Hashing}} = 0.14mWh$ and $E_{\text{Vanilla}} = 0.284mWh$
- Scaling 100: $E_{\text{Hashing}} = 0.734mWh$ and $E_{\text{Vanilla}} = 1.464mWh$

In Figure 4.6 the total savings are displayed. *Inner* segments correspond to the *vanilla* memory scheme, *outer* segments to the *hashing* scheme. The charts represent the consumption during transmission as red segment and miscellaneous consumptions in blue (i.e. wakeup energies, hashing, measurement).

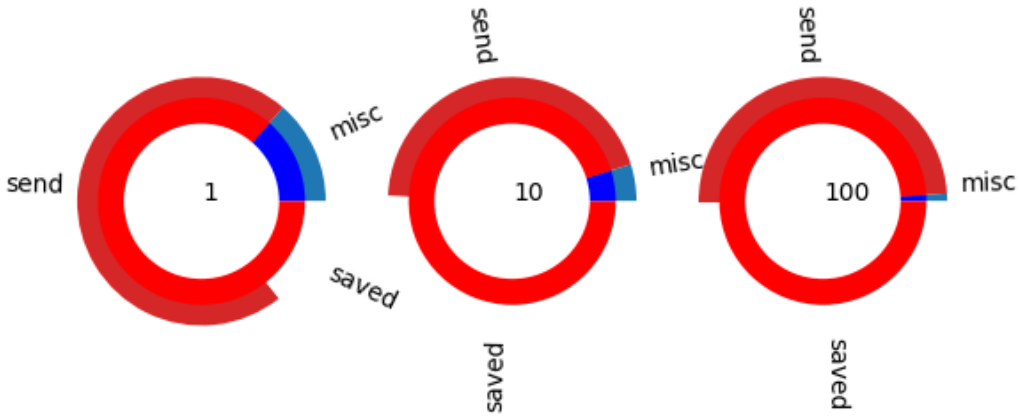


Figure 4.6: Comparison of Ratios between different Implementations of Scaling.

Summary. Depending on the scaling the actual savings are 14.27%, 50.69%, 49.83%.

Discussion

Summary. Further improvements are possible. The efficiency of the proposed memory scheme is reduced due to the prototyping stage of the hardware.

Despite the impressive savings pictured in Figure 4.6 the results provided in Figure 4.4 still are worse than specified in the microcontroller its datasheet (c.f. Figure 2.3). Multiple sources of error may have caused these deviations. The primary issue may be the fact that the defined power consumptions cannot be achieved using a prototype board, i.e. additional features that cannot be turned off completely draw enormous amounts of power. This is the main reason for not achieving the desired consumption during deep-sleep mode (c.f. Figure 4.4). Therefore far better results can be achieved using non-prototype hardware, i.e. designated microcontrollers for the hashing memory scheme. A secondary but rather minor reason may be noise during power measurement. Although a multimeter was used which is conform with industrial quality standards, the connections may have resulted in rather worse results than expected.

4.3 Air Quality Sensor Node

Summary. Deploying the Air-Quality node does indeed require additional hardware, i.e. the external MQ2 sensor board. Due to supplementary power demands, the overall consumption rises, still 50.09 per cent of energy can be saved.

The second experiment implementing the hashing memory scheme is about deploying a more advanced sensor to measure air quality. In order to enable those measurements, an additional piece of hardware is required. That is the MQ2 sensor. The required setup concerning the connections is given in Figure 3.3.

Measurements

Times in μs			
T_{WU}		261077	
T_{M}	85	438	3920
T_{Proc}	231	627	4699
T_{WUT}		48	
T_{Tr}	413988	747869	4105402
T_{Tr}'	505689	1512422	8378326

Currents in mA	
P_{Sleep}	142.2
P_{WU}	144.9
P_{M}	147.8
P_{Proc}	162.2
P_{WUT}	155.8
P_{Tr}	261.1

Figure 4.7: Measured Values during the Air Quality Sensor Node Experiment

Again, the measuring processes followed the described procedure in Section 4.1. The actual measurements provided in Figure 4.7 show the expected ratio between the single steps. Still the single power measurements deviate even more from Figure 2.3.

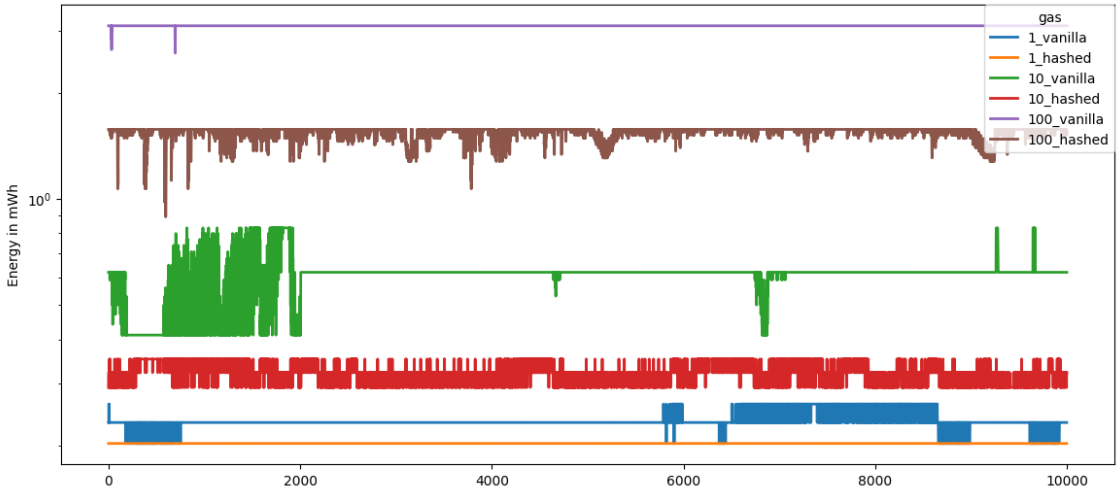


Figure 4.8: Plot of Captured Data aggregated with measured Power and Voltage.

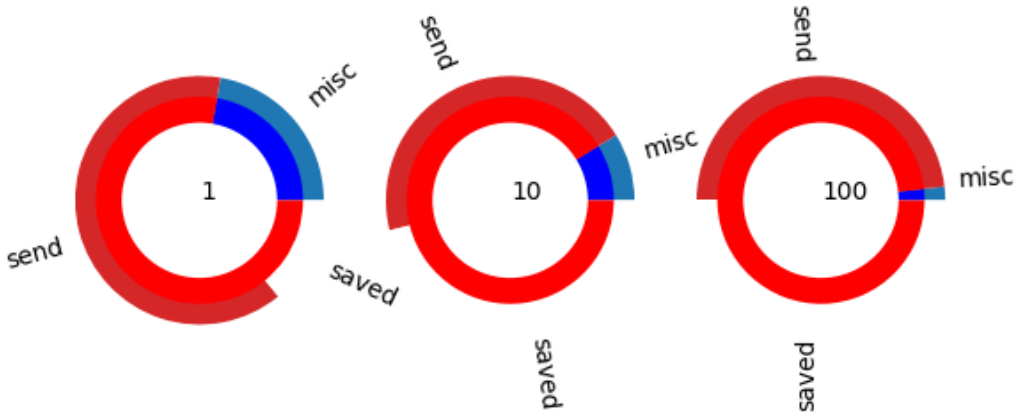


Figure 4.9: Comparison of Ratios between different Implementations of Scaling.

Evaluation

After the second application of Definition 2.9, concrete values in power consumption were computed:

- Scaling 1: $E_{\text{Hashing}} = 0.203mWh$ and $E_{\text{Vanilla}} = 0.236mWh$
- Scaling 10: $E_{\text{Hashing}} = 0.324mWh$ and $E_{\text{Vanilla}} = 0.601mWh$
- Scaling 100: $E_{\text{Hashing}} = 1.543mWh$ and $E_{\text{Vanilla}} = 3.092mWh$

Figure 4.9 points out the concrete savings following the same scheme as in Section 4.2 where the white part of the outer rings show the saved amount of energy.

Summary. Dependent on the scaling factor, the actual savings are 14.07%, 46.10%, and 50.09%.

Discussion

Summary. Again, the performance is expected to increase greatly under different circumstances considering designated hardware.

As already announced in the comments on the measured power consumption values, the deviation from Definition 2.9 is greater than in any other experiment. This happens due to the great power demands of the MQ-2 sensor itself. Also, different voltages were applied to compute the concrete energy values as the sensor itself draws up $5V$ instead of $3.3V$. Especially the consumption during idle mode diverges, as the MQ-2 chip cannot be "turned off" and thus continues to draw current during sleep mode. Still, as both memory schemes require the same amounts of energy for the single phases, the savings are rather high, c.f. Section 4.2.
When designing a piece of designated hardware for this very special use case, one may profit greatly from implementing options to turn off the sensor completely during the phases in which it is not required, as well as several other features.

4.4 Touch Sensor Node

Summary. The deployment of a touch sensor node is independent of additional hardware, again. Therefore no additional power to supply supplementary boards is required. Thus, concrete savings can rise to 65.93 percent.

The ultimate experiment that implements the proposed memory scheme is to set up a sensor node that measures the intensity of physical touch. As the microcontroller is equipped with this sensor already, the board itself can be powered with $3.3V$ (c.f. Section 4.2), and thus no additional hardware is required.
To measure different intensities, one may want to specify one of the appropriate GPIO pins. To ease the process, an additional jumper wire (and optically, a conductive pad) can be applied to that certain pin. The capacitive touch sensor usually is used as an external interrupt to wake up the board from various sleep modes. When touching the sensor it will output the intensity of the physical contact after activation, that is an integer greater or equal than 0. Without any contact with the sensor, its value will remain zero. Thus

one can distinguish not only intensities but also durations of touching interactions. The optional pad may provide better results for larger areas in very special applications.

Measurements

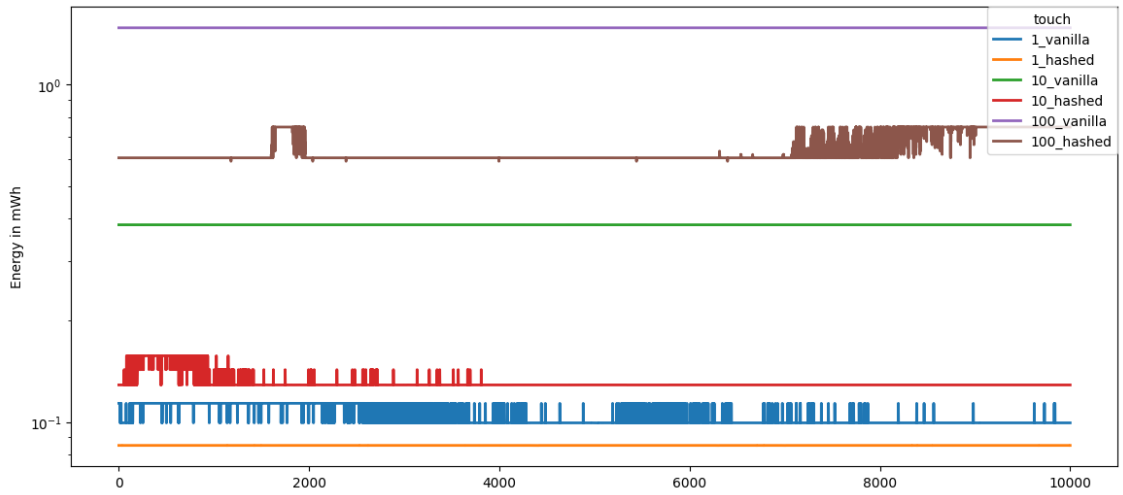


Figure 4.10: Plot of captured Data aggregated with measured Power and Voltage.

Times in μs			
T_{WU}		261077	
T_{M}	2067	19986	199291
T_{Proc}	225	632	4700
T_{WUT}		43	
T_{Tr}	413988	672211	3562564
T_{Tr}'	521764	2136411	8378964

Currents in mA	
P_{Sleep}	56.6
P_{WU}	56.4
P_{M}	56.2
P_{Proc}	76.7
P_{WUT}	70.4
P_{Tr}	188.7

Figure 4.11: Measured Values during the Touch Sensor Node Experiment

Figure 4.10 displays the aggregated results of all measurements done for this experiment over the specified ten thousand iterations. No additional pad nor wire was used during the measurements. Averaged results can be found in Figure 4.11.

Evaluation

The ultimate application of Definition 2.9 results in the following three values of power consumption

- Scaling 1: $E_{\text{Hashing}} = 0.085mWh$ and $E_{\text{Vanilla}} = 0.104mWh$
- Scaling 10: $E_{\text{Hashing}} = 0.131mWh$ and $E_{\text{Vanilla}} = 0.384mWh$
- Scaling 100: $E_{\text{Hashing}} = 0.640mWh$ and $E_{\text{Vanilla}} = 1.473mWh$

The actual savings are pictured in Figure 4.12 where the white outer segment shows the ratio of unused power.

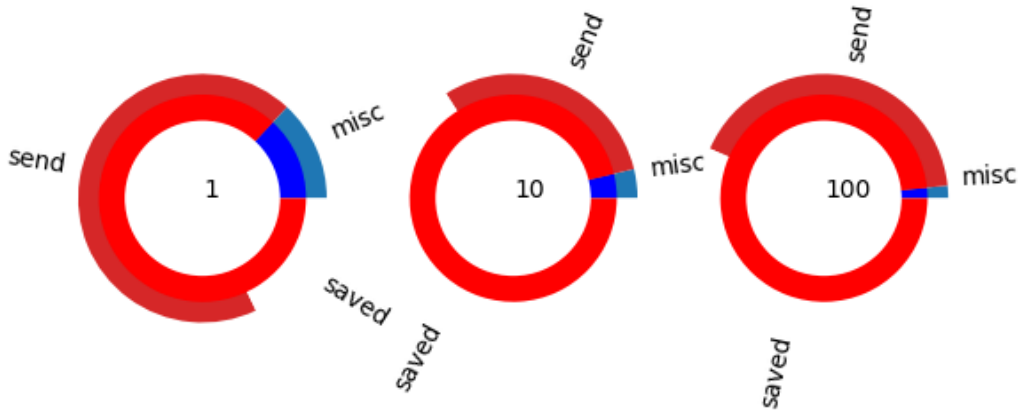


Figure 4.12: Comparison of Ratios between different Implementations of Scaling.

Summary. Actual savings dependent on the scaling chosen are 17.93%, 65.93%, and 56.53%.

Discussion

Summary. According to the measurements, high savings can be achieved. The enhancement is unconditional despite the immense effect of scaling. Still, there is a great potential un-used, due to unsuitable hardware.

One may already recognize the touch sensor node to be the most profitable use case among all experiments. Still there exists the potential to increase efficiency even further by following the same ideas as before. The identical issues about deviations in power consumption, compared to the values found on the datasheet, apply to this experiment.

4.5 Interpretation

Comparing the three experiments, one can easily identify a few similarities that provide arguments towards the usability of the proposed memory scheme.

Learning (Unconditional Enhancement). Without considering scaling in the first place, one can see the proposed memory scheme capable of saving energy for the three introduced use cases. No scaling ever decreases the energy efficiency of the sensor nodes as shown in the single experiments.

Learning (Scaling Matters). Despite the different outcomes between scaling factors ten and hundred in the single experiments, the results of scaling are always better than without scaling.

Learning (Un-Exhausted Potential). Concluding the discussion sub-sections of the three experiments, another common fact can be derived: The proposed memory scheme is limited by the microcontroller used, for now. By using designated hardware developed alongside the recommendations one may profit even further from the hashing scheme.

The Unconditional Enhancement

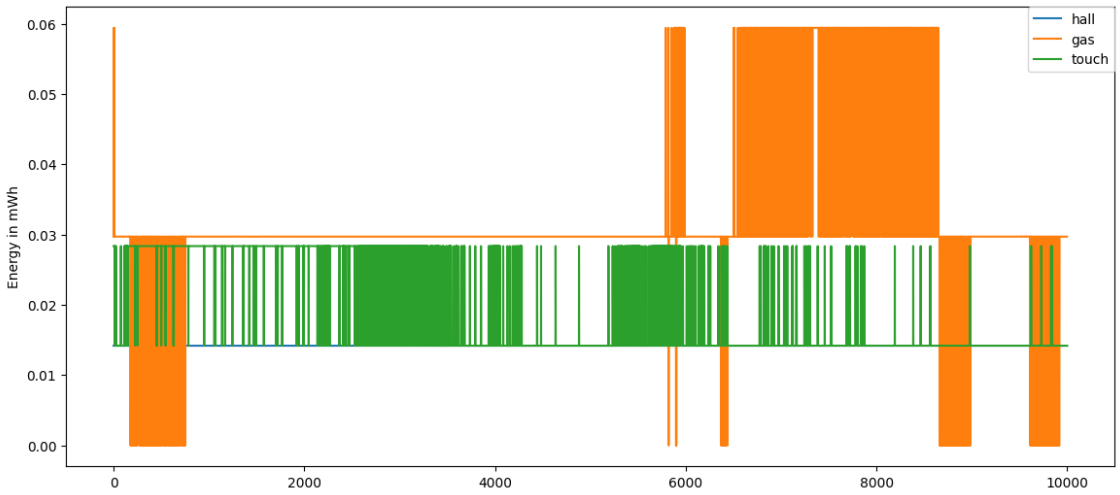


Figure 4.13: The Differences between Hashing and Vanilla Memory Scheme.

After facing the actual measurements in Figure 4.5, Figure 4.8 and Figure 4.5 one could already see that the line of the hashing approach is always dominating the line of the vanilla memory scheme. Figure 4.13 clearly shows the statement in the first Learning 4.5 as true: Even without scaling, the hashing memory scheme always enhances the energy efficiency.

The Importance of Scaling

Despite the definitive improvements, scaling can still boost the memory scheme its efficiency up to fifty percent and further. When observing the average savings, one can see that only approximately 15% can be saved while the amount of spared energy due to scaling may grow up to 54% (i.e. on average).
Therefore the choice of a proper duty cycle can be vital to consider concerning the hashed memory scheme.

The Un-Exhausted Potential

When considering messages of length 32 and their reduction to size 6, one may be disappointed to see the fifty percent decrease only. Accepting Fact 2.6 causes a harsh decrease in comparison to the theoretical efficiency. One may profit from applying a very special coding scheme to the hash digests before transmission. A scheme as such may be capable of merging multiple messages in order of saving complete transmissions.
Another downside of the implementations used for these experiments is the fact that the hibernation power-mode (c.f. Figure 2.3) could not be enabled (i.e. replaced by deep-sleep). Despite the savings of up to fifty percent of energy, implementing the actual hibernation mode (i.e. instead of deep-sleep) is expected to increases efficiency even further.

Nevertheless, the one major reason for reduced efficiency is the hardware itself, as already explained. When designing dedicated microprocessors, one may want to keep in mind some of the following ideas:

- The co-processor enabling fast cryptographic functions is a critical part and must not be removed.
- Many features provided by the Heltec chip are secondarily unnecessary but primarily a great drain for power, such as the screen, multiple sensors (i.e. depending on the use case), etc. Those should be removed.
- There may exist further enhancements that still require further research but are not covered in this thesis, such as pipe-lining approaches to chain measurement and hashing or alternative hashing functions.

Summary. Concluding the experiments, one may find the results convincing for the hashing memory scheme to be profitable on the one hand. On the other hand, the proposed enhancements may result in even greater efficiency.

5

Conclusion

Summary. Several conclusions can be drawn from this Bachelor's thesis. Primarily, the main question may be considered answered. Secondarily, no decent amount of energy was saved in any experiment but ultimately there is still an enormous potential for improvements that may cause great success in the future.

After analyzing the experiments as well as their results in detail, a summary will refresh all the important learnings such that the succeeding section may provide an outlook on the whole field.

5.1 Summary

Summary. The thesis its first aim was to introduce the new hashing memory scheme. Its second objective was to identify its actual potential, as it is, and whether it was of any use.

Recalling the hashing memory scheme introduced in Section 2.2, breaking the symmetry in communication (considering weak sender and strong receiver) was described as beneficial for the overall energy consumption of sensor nodes. By increasing the burden of the strong receiver, the weak sender its responsibilities decreased harshly. Three experiments implementing this new scheme not only showed promising timing behavior. Their measured and computed power demands were also found to have decreased for half the requirements of the common (i.e. the vanilla scheme) method. Thus the experiments providing evidence for quantitative analysis were a success and a hint towards the proposed idea its actual usability.
Due to limitations of physical nature and the given implementations concerning hardware, first estimations regarding efficiency clearly overshot the results. Multiple reasons provided in Subsection 4.5 were identified in the end, while other limitations were known from the beginning, such as Fact 2.6.
Due to the hard- and firmware implementations, concerning the microcontroller, there was no way to achieve actual square-root-sized messages. Considering even minimal preconditions for tests, there was always potential wasted, which advanced coding schemes may improve.
Hardware used in the experiments provided the ideal conditions for prototyping. Still, measuring hardware in this state of development is a setup for failure, when expecting

reasonable results (i.e. better outcomes).
The design of an appropriate duty cycle is vital to the hashed memory scheme its efficiency. As shown in the conclusion concerning the experiments in Section 4.5, primarily the hashing memory scheme its enhancement is unconditional but secondarily the design (i.e. the careful min-maxing) of an appropriate duty cycle can improve it to maximum efficiency.

5.2 Outlook

Summary. Considering a very special theoretical implementation, one may notice a gap between the performances of the two memory schemes. When considering a sufficiently large power source (i.e. an array of batteries), the hashing memory scheme can power a node longer than *three times* the duration a vanilla-node would, in this concrete example.

Despite the early development phase concerning implementations of the proposed memory scheme, first notional examples may help to visualize the effect in a concrete scenario.
Considering an array consisting of ten standard batteries, where each has a capacity of $7.4Wh$, one can see concrete differences between the two memory schemes and their performance. Looking at the most profitable experiment (i.e. touch sensor for scaling 10) one may want to imagine a concrete implementation. As this is a theoretical example, neglect the fact that measured currents drawn in passive mode are extremely high and fall back to the specification (c.f. Figure 2.3). Thus, the following values will be used to specify a duty cycle.

in ms					
T_{WU}	T_{M}	T_{Proc}	T_{WUT}	T_{Tr}	T'_{Tr}
261.077	19.986	0.632	0.04	672.2	2136
in mA					
P_{Sleep}	P_{WU}	P_{M}	P_{Proc}	P_{WUT}	P_{Tr}
0.0025	56.4	56.2	76.7	70.4	188.7

As one can deduce from summing over the table its left side, an active part of the duty cycle takes approximately one second when applying the proposed memory scheme but almost two and a half seconds, considering the vanilla scheme. Thus, let

$$T_{\text{Sleep}} = 59046.065ms \qquad \text{and} \qquad T'_{\text{Sleep}} = 57582.897ms$$

Depending on the use case its specification, the duty cycle needs certain fine-tuning. For the sake of this example, assume one would want to count customers by querying a touch sensor ten times each cycle. Furthermore, consider the following two frequencies:

1. In every run, measurements will be taken ten times but transmitted once.
2. A whole iteration takes a single minute.

When considering that concrete battery, that can supply the required voltage of $3.3V$, one can compute concrete values of energy in mWh:

$$\begin{aligned} E_{\text{Hashing}} &= 3.3V \times \left(T_{\text{WU}}P_{\text{WU}} + T_{\text{M}}P_{\text{M}} + T_{\text{Proc}}P_{\text{Proc}} + T_{\text{Tr}}P_{\text{Tr}} + T_{\text{Sleep}}P_{\text{Sleep}}\right) \times 60 \\ &\approx 7.86mWh \\ E_{\text{Vanilla}} &= 3.3V \times \left(T_{\text{WU}}P_{\text{WU}} + T_{\text{M}}P_{\text{M}} + T'_{\text{Tr}}P_{\text{Tr}} + T'_{\text{Sleep}}P_{\text{Sleep}}\right) \times 60 \\ &\approx 23.05mWh \end{aligned}$$

When comparing both computed energies to the battery array its capacity, one charge suffices to power the sensor applying the *vanilla* scheme for approximately **three hours** while a sensor implementing the *hashing* memory scheme can last up to almost **nine and a half hours**.
To, once more, point out the importance of the Bachelor's thesis its aim, a final example will be given to visualize the potential change a proper implementation of this attempt could make for a concrete application introduced in (Gallagher, 2018).

Example 5.1 (Rhino IoT (Gallagher, 2018)). Due to the trouble caused by poachers in the "Mkomazi National Park", Tanzania, a solution to save the endangered black rhinos was urgently desired. "LoRaWAN" makes use of the MAC-parameter "TDOA" (Time-Difference-of-Arrival). Researchers cooperating with Semtech Corporation made great use of this parameter by installing a tracking system for those threatened animals. One of the system its core features is the fact that no external location sensor, such as GPS, is required. By implanting their designed sensor nodes into the black rhinos their horns, patterns of poachers were analyzed over long ranges.
Despite the low power properties of "LoRa" and the design of battery and sensor to last longer than a decade, there may still exist the potential for longer durations of operation. A new approach in energy-efficient technology may expand the time a sensor node can survive. In concrete, this extension by a reasonable factor could result in a product that may last for half a rhino its life expectancy(Wikipedia contributors, 2021) or more.

Closure

Through introducing the hashing memory scheme, a new idea is established: Greater computational power increases energy demands **but it also brings the potential to decrease them even further**. Despite the comparably small savings, achieved in the Bachelor's thesis its experiments, the progress of hardware design (i.e. the increase of computational power) will raise not only this approach its effect but also its importance.

Bibliography

Alliance, L. (2020). LoRaWAN 1.0.4. specification. lora-alliance.org *1*.

Bouguera, T., Diouris, J.-F., Chaillout, J.-J., Jaouadi, R., and Andrieux, G. (2018). Energy consumption model for sensor nodes based on LoRa and LoRaWAN. Sensors *18*, 2104.

dictionary.com (n.d.). *internet of things*. https://www.dictionary.com/browse/internet-of-things. [Online; accessed 04-December-2020].

Gallagher, S. (2018). *We know you hate the Internet of Things, but it's saving megafauna from poachers*. https://arstechnica.com/information-technology/2018/06/rhino-iot/. [Online; accessed 2020-10-19].

Howerton, J.M. and Schenck, B.L. (2020). The Deployment of a LoRaWAN-Based IoT Air Quality Sensor Network for Public Good. In 2020 Systems and Information Engineering Design Symposium (SIEDS), pp. 1–6. DOI: 10.1109/SIEDS49339.2020.9106676.

Moore, G.E. et al. (1965). *Cramming more components onto integrated circuits*.

Peng, Y., Shangguan, L., Hu, Y., Qian, Y., Lin, X., Chen, X., Fang, D., and Jamieson, K. (2018). PLoRa: A passive long-range data network from ambient LoRa transmissions. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, pp. 147–160.

RandomNerdTutorials.com (2021). *ESP32 with LoRa using Arduino IDE – Getting Started*. https://randomnerdtutorials.com/esp32-lora-rfm95-transceiver-arduino-ide. [Online; accessed 2021-06-10].

Sonee, S. (n.d.). *Top IoT Communication Protocols Updated 2021 [ZigBee, NFC, And More]*. https://hashstudioz.com/blog/top-iot-communication-protocols-2020. [Online; accessed 27-January-2021].

Systems, E. (2016). ESP32 Datasheet.

Value, A.M. (2021). *Miners profitability*. https://www.asicminervalue.com/. [Online; accessed 2021-01-29].

Vangelista, L., Zanella, A., and Zorzi, M. (2015). Long-range IoT technologies: The dawn of LoRa™. In Future access enablers of ubiquitous and intelligent infrastructures, Springer, pp. 51–58.

Widmer, D. (2021). *bachelor-daniel_widmer-communication_vs_brute_force*. https://git.its.uni-luebeck.de/theses/bachelor-daniel_widmer-communication_vs_cmputation.

Wikipedia contributors (2021). *Black rhinoceros — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Black_rhinoceros&oldid=1026366158. [Online; accessed 10-June-2021].

Yang, X., Karampatzakis, E., Doerr, C., and Kuipers, F. (2018). Security vulnerabilities in LoRaWAN. In 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI), IEEE, pp. 129–140.