Development And Analysis Of A Low Power Sensor Network For A Parking Garage Application

Husein Vilic
Abstract

Development And Analysis Of A Low Power Sensor Network For A Parking Garage Application

Husein Vilic

This study covers the development and analysis of a Low Power Sensor Network, Parking Garage Application. An analysis of the Arduino platform is presented with a structural overview as well as the design choices made during the development of the parking garage application. This includes the analysis of power usage, sensor accuracy and sensor network communication. The entire system consists of the sensor nodes, a server and an Android application. The users are, through the application, informed whether any vacant parking spaces are available.
Innehåll

1 Introduction 1

2 Problem Statement 2

3 Background And Related Work 2
   3.1 Related Work ................................................. 3
      3.1.1 Smartparking ........................................... 3
      3.1.2 Vacant Parking Space Detection in Static Images .......... 3
      3.1.3 Parking Guidance System Utilizing Wireless Sensor Network and Ultrasonic Sensor .................. 4

4 Hardware 4
   4.1 Choice of Microcontroller And Board .......................... 4
   4.2 Choice Of Sensor And communication Module .................. 5
   4.3 Arduino Boards .............................................. 5
      4.3.1 Arduino Uno .............................................. 6
      4.3.2 Arduino Nano And Arduino Pro Mini ....................... 7
      4.3.3 Atmega328P-PU ........................................... 7
      4.3.4 Summary ................................................ 7
   4.4 Ultrasonic Sensor HC-SR04 And NRF24L01+ Transceiver Module .... 8
   4.5 Raspberry pi 3 ................................................ 9

5 Power Consumption 9
   5.1 Test Phase I - Arduino Boards ............................... 9
      5.1.1 Software ................................................ 10
      5.1.2 Test Structure .......................................... 11
         5.1.2.1 Arduino Uno ........................................ 12
         5.1.2.2 Arduino Nano ...................................... 12
         5.1.2.3 Arduino Pro Mini .................................. 13
         5.1.2.4 Atmega328P-PU .................................... 13
      5.1.3 Results ................................................ 14
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.3.1 Arduino Uno</td>
<td>15</td>
</tr>
<tr>
<td>5.1.3.2 Arduino Nano</td>
<td>16</td>
</tr>
<tr>
<td>5.1.3.3 Arduino Pro Mini</td>
<td>18</td>
</tr>
<tr>
<td>5.1.3.4 Atmega328P-PU</td>
<td>19</td>
</tr>
<tr>
<td>5.1.4 Discussion</td>
<td>21</td>
</tr>
<tr>
<td>5.1.4.1 &quot;Extended Standby Mode&quot; And &quot;Power Save Mode&quot;</td>
<td>22</td>
</tr>
<tr>
<td>5.2 Test Phase II - System</td>
<td>22</td>
</tr>
<tr>
<td>5.2.1 Software</td>
<td>22</td>
</tr>
<tr>
<td>5.2.1.1 Sensor Node</td>
<td>22</td>
</tr>
<tr>
<td>5.2.1.2 Base Node</td>
<td>24</td>
</tr>
<tr>
<td>5.2.2 Results</td>
<td>24</td>
</tr>
<tr>
<td>5.2.2.1 Complete Phase - Atmega328P-PU Circuit</td>
<td>25</td>
</tr>
<tr>
<td>5.2.2.2 Complete Phase - Arduino Nano</td>
<td>26</td>
</tr>
<tr>
<td>5.2.2.3 Complete Phase - Comparison</td>
<td>27</td>
</tr>
<tr>
<td>6 Sensor Positioning</td>
<td>28</td>
</tr>
<tr>
<td>6.1 Results</td>
<td>29</td>
</tr>
<tr>
<td>7 Packet Loss</td>
<td>31</td>
</tr>
<tr>
<td>7.1 Test And Result</td>
<td>32</td>
</tr>
<tr>
<td>7.1.1 Atmega328P-PU Circuit And Arduino Nano</td>
<td>32</td>
</tr>
<tr>
<td>8 Application</td>
<td>33</td>
</tr>
<tr>
<td>9 Discussion</td>
<td>35</td>
</tr>
<tr>
<td>9.1 Cost And Power</td>
<td>35</td>
</tr>
<tr>
<td>9.2 Packet Loss</td>
<td>35</td>
</tr>
<tr>
<td>9.3 Issues</td>
<td>35</td>
</tr>
<tr>
<td>9.4 User Experience</td>
<td>35</td>
</tr>
<tr>
<td>9.5 Alternative Implementation</td>
<td>36</td>
</tr>
<tr>
<td>9.6 Helpful Tools</td>
<td>36</td>
</tr>
<tr>
<td>9.6.1 Setting Up</td>
<td>37</td>
</tr>
</tbody>
</table>
1 Introduction

Embedded systems are all around us in almost every part of our daily life. They exist in our household and in the public. This includes washing machines, refrigerators, microwaves, TVs, cars, airplanes, vending machines, smart homes, security systems, industries and many more. Given that embedded systems exists in many of our day to day products, embedded systems continuously is expanding because of the rapid digitalization of our society.

This thesis is part of that digitalization. It is the documentation and creation of an embedded system that will monitor parking spaces and provide relevant information for the users. The system regularly monitors vacant parking spaces in real time and provide information to the users through a mobile application.

The system can be broken down in three parts. The embedded part (the sensor), the back end (the server) and the front end (Android application). Each of these parts carry an important role for the entire system. The sensors are gathering data that is then sent to the base node. The base node is receiving all the data and also handling the server part. This information is then processed at the server and is then added to a database. All the information stored at the database is accessible by the front end. In this case, an Android application.

In Figure 1 one can view the entire structure of the system. Sensor data, collected at the parking lot whether the parking is vacant or not, is sent wirelessly to the base node and then processed and added to a database. The phone application can then retrieve and present the data in a user friendly manner.

![An overview of the entire system structure.](image)

The sensor nodes periodically activates a ultrasound sensor to measure the distance to the closest object within a range of 5 to 150 centimeters (2-59 inches). If the measured distance allows the
conclusion that a car is parked in the parking lot or that it is empty, this information is sent wirelessly to the base node where it is stored in the database.

2 Problem Statement

The purpose of this paper is to analyze and create an embedded application that will be used at the company Semcon in Västerås. Conditions that the work is set out to fulfill is threefold; economical, energy consumption and functional. The project is supposed be affordable to make. The budget is set to be a maximum of 6 US dollars per sensor node (excluding the base node). This decision was made in order to encourage the possibility of other enthusiasts to continue development or build similar systems. It was also a decision in order to restrict the choice of available boards. Other companies that build similar systems (read more in “Related work” section) are a lot more expensive. That is another reason why this economical restraint was set.

The majority of the system is powered by batteries. It therefore becomes important to make the system low-power driven without compromising the economical aspect. The operator of the system is not supposed to change batteries every week in order for the system to function. This will also lower the long term cost (change of batteries) of the system. In addition to the economical and consumption restrictions, the system needs to be able to perform its core functionality. This means developing a system that is able to fulfill its main task in a desirable/functional manner.

In conclusion there is three problem statements this report covers. It includes a system that is power efficient, affordable to make and performs in the extent of its purpose.

3 Background And Related Work

There are many different types of embedded systems. Some systems are safety critical systems, meaning that a potential error might lead to loss of life or more preferably like John C. Knight wrote in his article “[…] a system is safety-critical when we depend on it for our well being.”[1]. For instance self driving cars[2], aircrafts, medical equipment, American Galileo[3] and many more. Then there are systems that are less critical. These systems are washing machines, microwave, refrigerator and other similar systems that would not necessarily put a human life at danger and that we would depend on for our well being. All these systems often have one or several controllers that are designated for a specific task. These controllers handle different kind of sensor-data and then perform an action depending on this data. The system covered on this report is created in the same way, there is a microcontroller handling a specific task, checking for vacant parking, and sending the sensor data to the base-node (the server).

Additionally, this system aims to contribute to the deployment of an urban Internet of Things (IoT). In the article ”Internet of Things for Smart Cities” a smart city was defined with the following: “[…] the final aim is to make a better use of the public resources, increasing the quality of the services offered to the citizens, while reducing the operational costs of the public
administrations.” [4]. This is pursued with a "Urban IoT" development, which bring a number of benefits. One of these being parking, "An urban IoT, indeed, may bring a number of benefits in the management and optimization of traditional public services, such as transport and parking, lighting, surveillance and maintenance of public areas, preservation of cultural heritage, garbage collection, salubrity of hospitals, and school.”. The economical aspect comes into play as the idea to create a affordable system for further development still holds, it will also benefit society, as it provides a cheaper way to help create smart cities. The environment is also taken in consideration. Streamlining the parking process will reduce the amount of time spent searching for a parking space. This will help lower additional cars in traffic and have an positive impact on emissions.

3.1 Related Work

It already exists companies dedicated to find available parking space. Some parking houses provide information both outside and inside the parking house showing the amount of available parking spaces. Other have information provided on a application where the parking spaces are scattered around the city. These systems have one thing in common, they use sensors in order to determine availability. There are several to choose from, infrared, magnetic, ultrasonic are some examples.

In this paper a parking management system was developed using a ultrasonic sensor that monitors vacant parking spaces in real time. Below one can read related systems tackling the same problem in different ways.

3.1.1 Smartparking

Smartparking is a company that was established in Perth, Scotland 1993 [5]. They specialize in parking technology and have multiple well known clients. They offer "on road” (outside parking) and parking house solutions. In both cases they use a "In-Ground Vehicle Detection Sensors” which features both infrared and magnetic technology. The sensor data is then handled with their "SmartRep” software. This allows them to have real time accessibility to the sensor data. In addition to parking management they also have parking house payment solutions that integrates with their parking system.

Comparing the system developed in this paper and Smartparking, this project only uses a ultrasonic sensor. The data handling in this paper uses a server where all the data is uploaded and an application where the data can be viewed by end-users, which is similar to what this project aims to do. Important to note is that this project is economically more beneficial.

3.1.2 Vacant Parking Space Detection in Static Images

A student from the University of California, San Diego created a system where a camera was used with histogram classification and vehicle feature detection [6]. The system produced very
good results, but with the drawback of not handling occluded parking spaces. If the parking space is behind cars or other obstacles it will not work. It could be solved by adding multiple cameras, but this will add additional cost.

Depending on the project this approach might be more appropriate than a sensor-based implementation. If for instance the entire parking lot is well illuminated and not occluded, this approach would only require one camera. This would mean a lower cost than for instance using the "Smartparking" approach. What is important to note is that the project demonstrates that there are multiple ways of solving the parking space problem.

3.1.3 Parking Guidance System Utilizing Wireless Sensor Network and Ultrasonic Sensor

A paper done by students from University of Malaya made a system that uses a WSN (wireless sensor networks) with a ultrasonic sensor in order to determine the availability of parking [7]. In addition to checking parking with the ultrasonic sensor, a shortest-path algorithm (A-star shortest path algorithm) is used in order to guide the driver to the parking space streamlining the parking process. To manage the parking cost, RFID technologies were used.

Both the University of Malaya and this paper have many similarities. They both use a ultrasonic sensor and they use a wireless sensor network to handle communication between the sensors. They differ in two areas. The University of Malaya paper uses a A-star algorithm after it has retrieved data from the sensors in order to calculate the shortest path to the parking space. It also uses RFID technology to determine the fee that the driver must pay by having timestamps of when the RFID tag has been registered and re-registered.

4 Hardware

As mentioned earlier microcontrollers and microcontroller boards are often used and serve an important purpose in todays embedded applications. Applications range from clocks, smart watches to autonomous driving cars provided from Tesla Inc [8] to critical systems in aircrafts. Choosing one microcontroller that accommodates a specific embedded application is as important as the application itself.

There are many manufactures of microcontrollers and microcontroller boards. They offer a panoply of choices covering the consumer demand. A few providers are; Cypress, Innovasic, Microchip, Renesas Electronics, Atmel Corporation, Zilog and many more.

4.1 Choice of Microcontroller And Board

The requirements for this thesis is to research and develop a system optimized in three different areas. As mentioned in section "Problem Statement", these are economical, energy consumption and functional. The hardware of choice is dictated by the criteria/requirements mentioned.
For this project the microprocessors from Atmel Corporation [9] were chosen. These are often associated with Arduino and their single-board microcontrollers, as almost all these boards use microprocessors from Atmel Corporation. They are inexpensive and accessible for the average consumer. The entire testing/development was done on the Arduino boards and also on a standalone microcontroller.

The following microcontroller and boards were used; Arduino Uno, Arduino Nano, Arduino Pro Mini and the microcontroller Atmega328P-PU. All of the boards that were used were not original Arduino boards but rather copies bought from eBay. It is important to mention that Arduino is an open source computer hardware and software company, meaning that the hardware reference designs are available to the public where third party companies are able to provide almost identical boards with a more competitive price. The results might differ slightly, but as a whole it should be identical with the original manufactured boards.

4.2 Choice Of Sensor And communication Module

As seen in section "Related Work" there are several approaches in creating a parking system. One underlining goal as mentioned in section "Problem Statement" is to build a system that is within the 6 US dollars limit, but it also should be power efficient. The NRF24L01+ transceiver module fulfills this.

The sensor module had to have additional requirements. The environmental factor was contributing to decide what sensor to use. The sensor needed to function indoors and outdoors. It was required to handle low light settings when indoor. The Ultrasonic sensor HC-SR04 does exactly this, as it uses ultrasonic waves to sense the vehicle.

4.3 Arduino Boards

Different Arduino boards have their own benefits and limitations. Analyzing from an application perspective, one Arduino board might be more beneficial than the other. Things to consider are; power consumption, connectivity (the amount of connected devices the board can handle), external devices (is the board able to power them), Wi-Fi and bluetooth requirements (does the system need any wireless connectivity), ethernet and more. Given the nature of the application covered in this report, the chosen boards and microcontroller were analyzed with two factors in mind, power consumption and reliable communication.

The following subsections presents a more detailed description of the different Arduino boards and microcontroller. The first three sections will cover the Arduino boards where as the last will cover the microcontroller. All of the Arduino boards use the AVR line Atmel MCU, therefore it will all be covered in the "Atmega328P-PU" section below.
4.3.1 Arduino Uno

The Arduino Uno is the largest board among the three and it has to offer the most in terms of accessibility and flexibility, but with that comes a larger power cost. This is covered more in detail under section "Power Consumption". The following is an overview of the important components;

**Microcontroller**
Covered in "Atmega328P-PU" section below.

**USB connector**
The USB connector has a twofold purpose. It is used to transfer code onto the MCU and also to power the entire Arduino board. The host computer will provide 5V to the board.

**Power Port**
An additional way to power the Arduino Uno is to use the power port. This port handles 6-20V (7-12V is recommended).

**USB Interface Chip**
This is the bridge between Computer’s USB Port and the serial port of the processor.

**Digital I/O Pins**
There are 14 Digital I/O pins and six of these pins are PWM (Pulse-Width Modulation), they simulate an analog output.

**Analog Input Pins**
There are six Analog input pins, these convert the signal from analog into a digital value.

**Power Pins**
The five pins consist of a Vin, 5V, 3V3, GND and IOREF. The Vin-pin is where the input voltage is applied from an external power source. The 5V and 3V3-pin provides a regulated 5V and 3V output. The IOREF-pin provides a voltage reference for a shield (boards that can be plugged on top of the Arduino PCB) and lastly the GND-pin which is the ground.

**Voltage Regulator**
Regulates the incoming voltage to 5V and 3V. These can then be accessed through the power pins (5V and 3V3 pins).

**Master Reset Button**
Button allowing a reset of the program on the Arduino Uno. This is done with a logical pulse to reset pin of microcontroller.

**ICSP Header Pins**
The ICSP pins (In-Circuit Serial Programming) are used to program the AVR microcontrollers (directly without the need of a bootloader).
RX/TX LED
The leds help visually see when there is ongoing communication.

4.3.2 Arduino Nano And Arduino Pro Mini

The Arduino Nano is a significantly smaller board than the Arduino Uno. Almost all of the on-board peripherals are the same as with the Arduino Uno. The exceptions are the power port, which it does not have, two extra analog pins (total of eight), no IOREF and a smaller version of the USB connector. On Arduino Nano the type of connector used is the Mini-b USB.

The smallest of three boards is the Arduino Pro Mini. This board has more hardware restrictions. It does not have a USB connector, USB Interface Chip, it has six analog pins (same as the Arduino Uno) and there are only three power pins (Raw uses regulator, VCC and GND). The remaining is in alignment with the other boards.

4.3.3 Atmega328P-PU

Lastly there is the microcontroller itself. In all of the above mentioned boards the Atmega328P-PU is used in all of them. The Atmega328P-PU does not have added parts as the above boards, that excludes a master reset button, USB interface chip, power port and other components. It is the bare bone brain of all boards mentioned. The microcontroller has 32kB of flash memory (2kB SRAM and 1kB EEPROM) where a 8-bit AVR CPU is used. It is able to operate on a maximum frequency of 20MHz and utilize 23 I/O pins.

4.3.4 Summary

The different Arduino boards and microcontroller fulfill the requirements from an economical, energy consumption and functional perspective. The bigger board offer more options to developers. With these increased options comes increased power consumption. The smaller size do present constraints, but in theory should have a lower power consumption.

In order to summarize in a convenient way Figure 2 below is provided. The chart summarizes the different boards and their components as an overview in order to understand where they differ.
4 Hardware

<table>
<thead>
<tr>
<th>Components</th>
<th>Arduino Uno</th>
<th>Arduino Nano</th>
<th>Arduino Pro Mini</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>USB Connector</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Power Port</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>On board USB Interface Chip</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Power Pins</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Voltage Regulator</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Master Reset Button</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>ICSP Header Pins</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>RX/TX LED</td>
<td>YES</td>
<td>YES</td>
<td>NO (only two leds)</td>
</tr>
</tbody>
</table>

Figur 2: Table summarizing components on each Arduino board

4.4 Ultrasonic Sensor HC-SR04 And NRF24L01+ Transceiver Module

The system will use two external devices in order to check any empty parking spaces and in order to transmit information. The former will use a Ultrasonic Sensor (HC-SR04). The sensor operates by releasing a sound wave (40k Hz) and retrieving after it hits an objects. Below are some specifications of the HC-SR04 sensor.

4-Pins, Operating Voltage And Distance

It uses four pins to operate. They consist of Vcc, Trig, Echo and GND. The Vcc is where the input voltage is applied, Trig sends out a sound wave and Echo captures it on the way back. The sensor operates on 5 volts. The distance achieved is 2 cm to 400 cm.

In order to transmit information a NRF24L01+ transceiver module is used. The module transmits through the radio bands. More specifically it uses the ISM band (industrial, scientific and medical radio bands)[10]. It operates with the embedded baseband protocol engine Enhanced ShockBurst[11], providing support for more advanced autonomous protocol operations but also less complicated ones. The module uses GFSK modulation (Gaussian frequency-shift keying) and has multiple configurable settings, such as, frequency channel, output power and air data rate. Below are some specifications of the NRF24L01+ transceiver module.

8-Pins And Operating Voltage

These pins consist of CSN, SCK, MISO, MOSI, IRQ, CE, Vcc, GND. The CSN is the SPI(Serial Peripheral Interface) chip select. SCK is the clock for the SPI and MOSI/MISO are SPI slave data input and slave data output respectively. The IRQ is a interrupt pin and CE is a chip enable
that activates RX or TX mode. Lastly there is Vcc for the input voltage and ground. It operates from 1.6 volts to 3.6 volts and achieves 40 meters in the open and about half in closed areas.

4.5 Raspberry pi 3

The server-side of the system was implemented on the Raspberry Pi 3 (Rp3). The Rp3 is an single-board computer that initially was intended for education purposes only, but extended this target market and as of 2018 have sold 19 million units.

There is available information of the board and the amount of units sold on their website[12]. Important to note is that the Rp3 has an extended 40-pin GPIO header and Ethernet capabilities. The Rp3 will run the server of the system and it will be powered through a wall socket because of the availability it needs to have on the user end.

5 Power Consumption

In order to choose the most suited Arduino board for this specific application one must evaluate the power consumption of the entire system, in a so called complete phase. A complete phase is defined as the entire application, when it is functioning and fulfilling the project requirements.

Before tests were done on the complete phase, tests on different individual parts of the system were made. These are divided in two parts. The first one, "Test Phase I", covers tests that are made on the individual boards and on the microcontroller without any external hardware (Ultrasonic Sensor HC-SR04 and the NRF24L01+ transceiver module).

The second test phase, "Test Phase II", covers tests with the additional hardware attached. This will cover both the Ultrasonic Sensor HC-SR04 and the NRF24L01+ transceiver module with the board/microcontroller.

5.1 Test Phase I - Arduino Boards

Before any calculations were made on the second test phase (the entire system) an initial test phase was done to evaluate the power consumption of each board and microcontroller without any additional hardware attached. The purpose of this was twofold. Firstly, to understand how much the different boards and MCU consume when idling and secondly to try and minimize the power being consumed in that idling state.

Atmel Microcontrollers allow the programmer to control different power saving modes. All the tests were performed on empty code. The tested modes were the following:

- No Power-Saving Mode
- Standby Mode
5 Power Consumption

- Extended Standby Mode
- Power Save Mode
- Power Down Mode
- Power Down Mode + ADC Off

All of these modes shut down unused modules in the MCU and that allows it to save power. There are four areas that are selectively switched off in this process: active clock domains, oscillators, wake-up sources and the software BOD (Brown-out Detector). In Figure 3 one can see what is turned on while the MCU is in respective mode. The empty boxes imply that it is turned off. What Figure 3 does not show is the "Power Down Mode + ADC Off" and "No Power-Saving Mode". The "Power Down Mode + ADC Off" is the most strict state, but the "Power Down Mode" and the ADC were shut off separately. These modes and the ADC are all changed by manipulating the registers in the Atmega328P-PU. What is important to note is that whether the active clock domains, oscillators or other component is turned off, it saves power. This does not necessarily mean switching of every part of the system to be beneficial. Some systems require timers or other parts to be active in order to function. One would also want to exit sleep mode after some time. This can either be done with an external interrupt or with an internal timer.

A important component that is always available, regardless of power saving mode, is the "Watch Dog Timer". This is the timer that is used when waking the system up, out of the sleep mode.

![Figure 3: Different power saving modes with an overview of active clock domains and wake-up sources. (Taken from Atmega328P-PU datasheet[13])](image)

5.1.1 Software

The code used to test the power saving is straight forward. The main idea is to let the microcontroller run for ten seconds in order for it to stabilize the current flow. After the microcontroller has a steady current draw it will enter a specific power saving mode (mentioned in previous section). Measurements are made before and after entering any mode. In Figure 4 one can view the code used for the specific "Power Down Mode + ADC Off" test.

The Setup is left empty and all testing code is in the loop section. Because a stable current is preferred for the measurements, on row 8 one can find the "delay(10000);" statement, this delay will last for ten seconds. After 10 seconds when a stable current draw is achieved the ADC
is turned off. In order to manipulate the microcontroller bitwise operators are necessary if no libraries are used. In these tests no additional libraries were used.

Turning off the ADC one must make a bit shift to the 7:th place in the register (ADCSRA), then inverting it to a zero and applying an AND operation. This can be seen on row 10. Similarly when entering different power saving modes a register is manipulated. On row 11 this done for the "Power Down Mode" by setting the second bit and also the first bit on row 12. Lastly the assembly code on row 13 is used in order to execute the sleep instruction.

Important to note is that the code used to enter other sleep modes is similar to the one in Figure 4. The difference would be in how the bitwise operations are done on the SMCR register. Depending on the power saving mode, different bitwise operations need to performed.

```c
void setup() {

}

void loop() {
  delay(100000);
  ADCSRA &= ~(1 << 7); // ADC is turned off
  SMCR |= (1 << 2); // Power Down Mode
  SMCR |= 1; // Enable sleep
  __asm__ __volatile__("sleep"); // assembly code to go to sleep
}
```

Figur 4: Code testing the "Power Down Mode + ADC Off" mode

### 5.1.2 Test Structure

The purpose of the tests are to analyze the power consumption with the different power saving modes in combination with a range of operating voltages. Given that the final system will have a power supply where, with time the voltage will drop, it is interesting to understand how different voltage levels affects the idling (power saving) state. In addition, this is done to achieve an overview of the different hardware and their difference in power consumption. The finished application will spend most of the time in a power saving state. It is therefore again, important to review the amount of power consumption in the power saving mode.

In order to test this, various voltage levels are used. For the Arduino Uno, Arduino Nano(with and without power LED) and Arduino Pro Mini board (5 volt 16Mhz version), voltages used are 5, 6, 7, 8, 9, 10, 11 and 12. Lastly the microcontroller Atmega328P-PU is tested on 1.2, 1.5, 2, 3, 4 and 5 volts.

Five simple programs are written to test (see Figure 4 for “Power Down Mode + ADC Off”
example) the different modes and how much power they save. Before the board/microcontroller enters a power saving mode, one initial measurement is made and then another right after entering. This is repeated ten times on each voltage input in order to calculate an average. All the boards use an Atmega328P-PU microcontroller.

5.1.2.1 Arduino Uno

The test were setup using the following equipment;

- An Arduino Uno board with Atmega328P-PU
- USB cable
- Power supply

Setting up the Arduino Uno was straight forward. The only connection needed was the USB connection. With the USB cable the code was uploaded to the board. Upon completing the code upload the USB cable was removed and voltage was provided from an external power supply. This connected on the Vin-pin and the ground to the GND-pin on the Arduino Uno.

5.1.2.2 Arduino Nano

During the testing process a decision was made to remove the constant powered ”power on” LED on one of the Arduino Nano boards. This modified Arduino Nano was also tested with exactly same code and voltages as the non-modified.

![Arduino Nano](image)

Figur 5: Arduino Nano unmodified (including power on LED indicator)

The test were setup using the following equipment;

- An Arduino Nano board with Atmega328P-PU
5 Power Consumption

- Mini-B USB cable
- Power supply

Similarly to the Arduino Uno the only necessary connection was the Mini-B USB connection to the laptop. When code was uploaded to the board the USB was removed and a external power supply provided power to the Vin-pin and GND-pin.

5.1.2.3 Arduino Pro Mini

This board required soldering because it came without pre-mounted headers. The test were setup using the following equipment:

- An Arduino Pro Mini board with Atmega328P-PU
- FTDI serial TTL to USB 6-pin cable (FT232BL)
- Power supply

![Arduino Pro Mini](image)

Figur 6: Arduino Pro Mini

5.1.2.4 Atmega328P-PU

The test were setup using the following equipment:

- A 16 MHz crystal
  The Atmega328P-PU only has a 8 MHz internal system clock. This 16 MHz crystal is needed if one wants to run the microcontroller anything higher than 8 MHz.
- A 10k resistor
  The resistor is not necessary for our testing purposes, but is good practice to have because of the weaker internal pull up resistor in the Atmega328P-PU reset pin. The purpose of this is to remove any potential sporadic resets in a noisy environment.
- Two 22 picofarad (ceramic) capacitors
- An Arduino Uno board with USB

The only usage of the Arduino board was to program the microcontroller.

In Figure 7 one can take a closer look on the schematics used in order to program the microcontroller. There are five important connections allowing this process. Power and ground are supplied through the Arduino Uno board using the 5v-pin and GND-pin (red and black wires). The board is in turn powered through the USB-port.

In order send data to the Atmega328P-PU we will use the 0-pin and 1-pin. This will be the RX and TX-pins, the pins that are used for serial communication. On the microcontroller one can find these on the 2-pin and 3-pin.

Because the Atmega328P-PU only has an internal 8MHz clock and because the test is to be performed on a 16MHz clock, a 16 MHz crystal was added with two 22 picofarad capacitors. This external crystal is then connected to the 9-pin (PB6) as input to the inverting Oscillator amplifier and the 10-pin (PB7) as output from the inverting Oscillator amplifier. This will allow the microcontroller to operate at 16MHz. The reset button was also connected to the microcontroller in the case of any upload problems that might occur. When the microcontroller is programmed the Arduino Uno is removed and power from a voltage supply is used instead.

### 5.1.3 Results

All of the results measured under section ”Test Phase I” are covered under this section. Important to note is that these results only represent the power consumption on a board and microcontroller with no external sensors or receivers attached. In Figure 8 one can view a base case from all the different boards and the microcontroller.
5.1.3.1 Arduino Uno

Figure 9 demonstrates the power consumption of the Arduino Uno at different voltages during the various power saving modes. The voltages seen in the figure are 5 to 12 with 1 voltage increment.

Looking at Figure 9 one can conclude, as expected, that the mode with the lowest power consumption is the "Power Down Mode + ADC" mode (the purple bar) because most of the board components are turned off. There is a noticeable drop in power consumption when 5 volts is used in comparison to the other voltages. All voltages from 6 to 12 have a similar power consumption drop when a power saving mode is entered, where as 5 volts is significantly lower. The 5 volts test also has a much lower consumption in the "Before entering power saving" mode (Light blue bar, which indicates the power consumption when running a delay statement. See more under section "Software" (2.1.1) for code details.) This is probably part to the on-board regulator being bypassed, saving additional power consumption.
5 Power Consumption

Figur 9: Diagram showing results of testing done on different power saving modes on the Arduino Uno at different voltages. It is in logarithmic scale.

The dark blue bar ("Extended Standby Mode") and the yellow bar ("Power Save Mode") both have very similar pattern throughout the voltages. Looking at Figure 3 there is only one difference, the main clock source. With "Extended Standby Mode" this is enabled but with "Power Save Mode" it is not. This difference did not seem to affect the consumption except when tested for 6 voltage, where a small difference occurred and the "Power Save Mode" consumed less (by a small margin).

In conclusion the Arduino Uno has an idle power consumption of around 50mA and a optimal power saving mode ("Power Down Mode + ADC" mode) of around 30mA, with the exception of the 5 voltage test. Arranging the different power saving modes from most efficient to least efficient (on all voltage levels) it would be "Power Down Mode + ADC Off", "Power Down Mode", "Standby Mode", "Power Save Mode" and "Extended Standby Mode".

5.1.3.2 Arduino Nano

Figures 10 and 11 demonstrates the power consumption of the Arduino Nano at different voltages during the various power saving modes. The voltages seen in the figures are 5 to 12 with 1 voltage increment. Both of the figures are in logarithmic scale. As mentioned in section "Arduino Nano" (2.1.3) the test was performed on two Arduino Nano boards. One of the boards had the constantly on power LED removed and the other was unmodified.

Figure 10 are the results from the unmodified board (with power LED). Similarly to the Arduino Uno the most power efficient mode is the purple bar ("Power Down Mode + ADC"). There is also a noticeable drop in power consumption in all the power saving modes when 5 volts is used.
5 Power Consumption

in comparison to the rest of the voltages.

The results from the modified version (without power LED) can be seen in Figure 11. These results show a 63% (from 6 to 12 voltage range) decrease in power consumption when comparing the "Power Down Mode + ADC Off" mode and a 15% (from 6 to 12 voltage range) improvement when comparing the "Before any power saving" mode (light blue bar) on both boards.

In conclusion, with the unmodified version, the Arduino Nano has an idle power consumption of 20.5mA and an optimal power saving mode ("Power Down Mode + ADC" mode) of around 4.9mA, with the exception of the 5 volt test. The modified version (without power LED) has an idle power consumption of 17.4mA and a optimal power saving mode of around 1.8mA, with the exception of the 5 volt test. This behavior, at 5 volt, is similar to the Arduino Uno board. The regulator is also bypassed here, in addition to this there is smaller drop with the Nano than with the Uno.

![Figure 10: Diagram showing results of testing done on different power saving modes on the Arduino Nano (with power LED) at different voltages.](image-url)
5 Power Consumption

Figure 11: Diagram showing results of testing done on different power saving modes on the Arduino Nano (without power LED) at different voltages.

The order from most to least efficient power saving mode (on all voltage levels) is similar to that achieved on the Arduino Uno when analyzing the unmodified board results. This was "Power Down Mode + ADC Off", "Power Down Mode", "Standby Mode", "Power Save Mode" and "Extended Standby Mode".

When the modified board is analyzed, more specifically the 5 voltage, one can find that the "Power Down Mode" has a lower power consumption than the "Power Down Mode + ADC" mode. The result is further discussed in "Discussion" (section 2.1.7).

5.1.3.3 Arduino Pro Mini

In Figure 12 one can view results from the Arduino Pro Mini board with the same voltage range used on the Arduino Uno and Arduino Nano. Examining the results one will come to the conclusion that the same applies on the Arduino Pro Mini board as the previous boards. The lowest power consumption is provided by the "Power Down Mode + ADC Off" mode.
Analyzing the 12 voltage test one can see a substantial increase in power consumption compared to the rest of the voltages. Running at the voltage range of 6 to 11 seem to have approximately the same amount of power consumption. The 5 volt test has a noticeable smaller consumption.

In conclusion the Arduino Pro Mini has an idle power consumption of 21.4mA and an optimal power saving mode ("Power Down Mode + ADC Off") of around 3.2mA, with the exception of the 5 voltage test and 12 voltage test. The arrangement from most to least efficient power saving mode (on all voltage levels) is "Power Down Mode + ADC Off", "Power Down Mode", "Standby Mode", "Power Save Mode" and "Extended Standby Mode".

5.1.3.4 Atmega328P-PU

Figure 13 demonstrates the power consumption of the Atmega328P-PU at different voltages during the various power saving modes. The voltages seen in the figure are 5, 4 and 3 and the diagram is in logarithmic scale.

By viewing the diagram one can determine the most optimal power saving mode, which is the purple bar (Power Down Mode + ADC Off). Depending on what voltage is provided to the board the microcontroller will vary the power consumption, but for every voltage the "Power Down Mode + ADC Off" mode will provide the lowest power consumption. As one can see from Figure 9 the power consumption can reach approximately as low as a one hundredth of an ampere.

The light blue bar ("Before any power saving") is the initial power consumption of the microcon-
controller before entering any power saving mode. The purple and light blue bars have numerical values attached to them.

Figur 13: Diagram showing results of testing done on different power saving modes on the Atmega328P-PU at voltages 3, 4 and 5.

In addition to this one can also see that the ADC contributes to a decent amount of power consumption. Comparing "Power Down Mode + ADC Off" and "Power Down Mode" there is ten times more consumption when the ADC is on. Fortunately for the finished application there will not be any need of an ADC. This will be turned of and additional power will be saved.

In Figure 14 one could see the remaining voltages that were tested, these include 1.2, 1.5, 2 volts. The behavior of the Atmega328P-PU at these voltages was unpredictable. Looking at the numbers there is clearly no response from the different power saving modes.
5 Power Consumption

Figur 14: Diagram showing results of testing done on different power saving modes on the Atmega328P-PU at voltages 2, 1.5 and 1.2.

5.1.4 Discussion

Analyzing the result one can clearly come to the conclusion that the Atmega328P-PU is the most power efficient in comparison to the Arduino boards. When the Atmega328P-PU is operating on 3 to 5 voltages allowing it to make use of the power saving modes, it can result in approximately 100 times lower power consumption in comparison to the Arduino Pro Mini and Arduino Nano.

The Arduino Nano board was tested with and without a power LED. This LED was constantly on when the board was powered. Viewing the result one can see that there is a 63% drop (from 4.9mA to 1.8mA) in power consumption when comparing the two boards in the "Power Down Mode + ADC Off" mode. When the unmodified Arduino Nano is compared with the Arduino Pro Mini, the Pro Mini has lower power consumption. This does not hold true when comparing the Pro Mini to the modified Nano board. Important to note is that the Arduino Pro Mini also has a power LED constantly on when the board is running. No tests were performed on a Pro Mini without the power LED.

The Arduino Uno is least efficient. Comparing it to the other boards the most efficient power saving mode is still more than the other boards in their "Before any power saving" state.

Given this result the ideal choice of hardware would be the atmega328P-PU alone without any board. The microcontroller alone will provide a significantly lower power consumption allowing the application to run on batteries for longer periods of time.
5.1.4.1 "Extended Standby Mode" And "Power Save Mode"

The overall result across the different boards and microcontroller showcases identical patterns except in one case. Looking more specifically at "Extended Standby Mode" and "Power Save Mode" one can see some irregularities. In theory the "Extended Standby Mode" should consume equal to or more than the "Power Save Mode". In Figure 3 one can see that the main clock source is not enabled when the microcontroller is in "Power Save Mode". One less active oscillator should in theory have less power consumption as it does not need to power a extra clock. This project will use the power saving mode that has the least power consumption and will not be using "Extended Standby Mode" or "Power Save Mode".

5.2 Test Phase II - System

This phase reviews more extensively the power consumption of the entire system. The Atmega328P-PU and the Arduino Nano (without power LED) were used in this second phase. The complete system was placed in the environment mentioned in section "Packet Loss" seen in Figure 23. Both hardware were running the exact same code, with the exception of a 2N7000G MOSFET on the Atmega328P-PU circuit board which acted as a switch and therefore had additional code. The Atmega328P-PU circuit and Arduino Nano were sending information to the same base node. The distance from both sensor nodes and the base station was the same in order to monitor the packet loss in a fair (section "Packet Loss"). To power the sensors, two set of batteries were used, one for each sensor. The battery used for the Arduino Nano was a 2400mAh, 9 volts battery and for Atmega328P-PU circuit a 2400mAh, 9 volts battery as well.

The program was set to send a packet every 8th second. During the other 7 seconds the system was in a power saving mode, more precisely the "Power Down Mode + ADC Off" mode. To understand the code flow one can view more in depth explanation under the "Software" section below.

5.2.1 Software

It is important to understand the code when discussing the power consumption. The code used in "Test Phase II" is different from the one used in "Test Phase I". Instead of only having power saving code, the program needs to handle the ultrasonic sensor and also the NRF24L01+ transceiver module. In addition to this libraries were used for a more readable code. Below are two sections covering the software on the "Sensor Node" and the "Base Node".

5.2.1.1 Sensor Node

The structure of the code can be seen in Figure 15 and Figure 16. The flow diagram below (Figure 15) demonstrates the setup-part of the code. In this section the initialization of sensors and watchdog is performed. This includes setting certain pins as input/output, starting the radio and choosing a power level.
When the setup has been initialized the loop function will be executed. In Figure 16 the flow diagram can be viewed. This is where calculations are performed and messages are sent. When the system is not in power saving mode the calculations are performed with the help of the Ultrasonic sensor. The Ultrasonic sensor will send out a sound wave and calculate the distance. The formula to calculate this will be:

\[
Distance = \frac{Speed\ of\ sound \times Time\ taken}{2} \quad \text{where,}
\]

\[
Speed\ of\ sound = 343\ meters\ per\ second \quad \text{and,}
\]

\[
343\ meters\ per\ second \Rightarrow 29\ microseconds\ per\ centimeter
\]

The 343 meters per second (2) is approximately 29 microseconds per centimeter. The “Time taken” needs therefore be divided with 29. This will proved the distance in centimeters. Lastly there is a division by 2 removing the additional time it took for the sound wave to travel. This yields the final equation (4):

\[
Distance_{cm} = \frac{Time\ taken}{29} \quad \text{cm}
\]
When the measurements have been made the distance will either be within or outside of the condition (between 5 - 150 centimeters) seen in Figure 16. In each case, a message will be transmitted containing an ID-number of the node sending it and a boolean informing if the parking space is empty or occupied. After the transmission the node will enter a power saving mode. It will stay in this mode until the watchdog timer expires. It will then wake up from its power saving mode and perform a ultrasonic measurement again.

5.2.1.2 Base Node

The base node is powered all the time. It is set to listen for incoming packages and update a file containing data of the parking spaces. The data received from the nodes is a struct with two values. The first value being the ID-number of the sending node and the second value is a boolean indicating whether a parking is vacant. The base node will search the data file and update the data value with the correct ID-number. The base node is connected to the Raspberry pi which has Internet connection. Users will through the Android application retrieve the data stored on the Raspberry pi and display it on the phone.

5.2.2 Results

The systems were set up and regularly checked in order to analyze the power consumption. Below one can find overall performance with each system. In addition to having the entire system
analyzed, the system is broken down in parts where each module and their power consumption also is analyzed individually.

The power consumption of the complete phase can be divided into two parts. The first part is when the entire circuit is idling in a power saving mode. This is where the system will spend most of the time. The second part covers power consumption when the system is out of the power saving mode performing measurements and sending data. It is important to reduce the amount of consumption during the power saving mode because most of the time the system will be in an idling state.

The complete phase was tested on the Arduino Nano board (without power LED) and on the Atmega328P-PU. The purpose is to analyze the average consumption in the idle state, the measurement/sending state and in total.

5.2.2.1 Complete Phase - Atmega328P-PU Circuit

The Atmega328P-PU had the lowest power consumption. During the power saving periods, the circuit was using 2.7mA. When the system was using the ultrasonic sensor and sending the data, it used on average 20.4mA. The time it took for the system to calculate distance and send data, was on average 1 second. This was done every 30 seconds. In Figure 17 one can view the different power consumption in idle mode and measuring/sending mode.

![Figure 17: Demonstrating the different power consumption of both system. The blue bar is the idling mode (Power Saving mode) and the red is the measuring/sending mode.](image)

The battery used for the system was alkaline and it operated in an environment of 21 Celsius on average. During testing, the system voltage was tested every day. The test was supposed to last the entire battery life cycle. On the eleventh day of testing the base node stopped receiving packets from the Atmega328P-PU system. Further investigation concluded that the Atmega328P-
PU system continuously drew 20mA from the battery. Further testing was canceled and a debugging process ensued (more under the “Discussion” section). The voltage measurements during the ten days are shown in Figure 18.

![Voltage measurements during a span of 10 days.](image)

**5.2.2.2 Complete Phase - Arduino Nano**

The second system that was tested was the Arduino Nano without the power LED. Seen in Figure 17 one can see that both systems are quite the same when measuring and sending data. Where they differ is in the power saving mode. The Arduino Nano runs at 4,7mA, consuming an additional 2mA. This system was tested during same conditions using same method of testing. The duration of this test was 14 days. Throughout testing the system operated as expected without any problems. In Figure 19 one can view the results of the Arduino Nano system.

![Voltage measurements during a span of 14 days.](image)
5.2.2.3 Complete Phase - Comparison

Additionally a diagram was made to compare the two systems. In Figure 20 one can view both of the system voltage results. They cover 10 days and demonstrate how the different power consumptions affect the voltage over time.

![Figure 20: Voltage levels of both systems over time.](image)

The Atmega328P-PU system has a lower power consumption during the power saving mode. Because both systems are in a power saving mode the majority of time, it will be the determining factor to the big difference in voltage values. One could very roughly calculate theoretically how much of battery life the two system would have given an average consumption. To calculate this one needs the average consumption of the system.

**Atmega328P-PU System**

Every 30 seconds a measurement was made and a packet was sent. This process lasted 1 second. The system therefore spending 3.3% of the time consuming 20.4mA. During the remaining 96.7% the system is consuming 2.7mA. The total average consumption is thus given with the following:

\[
\text{Total average consumption} = (0.033 \times 20.4mA) + (0.967 \times 2.7mA) \quad \text{and,} \quad (5)
\]

\[
\text{Battery life} = \frac{\text{Battery Capacity}}{\text{Total average consumption}} = \frac{2400mA\text{h}}{3.2841mA} \approx 30\text{ days} \quad (6)
\]

This means the Atmega328P-PU system would approximately last 30 days given ideal conditions on battery and environment.
6 Sensor Positioning

Arduino Nano

Using the same calculations as for the Atmega328P-PU system, the Arduino Nano would last 19 days in perfect condition. The significant difference, as mentioned earlier, is the "power saving mode" consumption. With the Atmega328P-PU one can reach much lower consumption in these saving modes. A contributing factor being additional on-board components on the Arduino Nano.

6 Sensor Positioning

Given that, the ultrasonic sensor calculates distance by sending out a sound wave and then retrieving it, some results have interesting outcomes when the surface is not smooth. For the sensor to work properly, having a correct bounce-back, it would ideally have a totally flat surface. It would often calculate values in the span of 2000-3000 when directed towards a shirt that is rumpled.

In order to further analyze the ultrasonic sensor in a relevant manner for this project, a sensor node was used and placed along a parking rail. Whenever relevant values would be retrieved from the sensor, a diode would light up and when unwanted values would be retrieved the diode would blink multiple times. This sensor node was moved across the rail testing multiple parked cars. Some of the cars were parked in reverse. The node was kept in a central position in relation to the parked car. Additionally some other tests were done with the sensor node placed underneath the vehicle and above the vehicle (face down). These were the most relevant positions to test.

In Figure 21a one can view the parking lot with the railing. The sensor node was placed on the railing in a small package. This packaging helped indicate when the diode was lighting up and when it was blinking, making the inside glow red.

In total the entire test was performed on 50 cars. The test, where placing the node on the railing and in a central position was done on all of the 50 cars. The test where the sensor node was placed underneath the vehicle was done on 25 cars. Lastly, the above vehicle (face down) test was done on 25 vehicles. The different tests were made in order to examine different possible placements of the sensors. When the sensor node was placed underneath the car, the placement was in the center of the vehicle.
In Figure 21b one can view a sample of the different kind of bumpers that the sensor was tested on. The common factor with most of the models was the front grill. Almost every car has a grill in the front for the intake of air. This was worth showcasing as the sensor was frequently hitting these grills directly when placed on the rail. On each test the sensor node was left for 30 seconds, calculating measurements every second.

### 6.1 Results

For the rail test, where the sensor node was placed on the railing facing the front of the cars, the result showed a 96% (48 vehicles) successful calculation on the first ultrasound transmission. On the second ultrasound transmission and rest of the transmissions, there was a 100% (50 vehicles) successful rate. The cars were all randomly selected with different models, meaning there were a wide range of size, shapes, grills and height, partly demonstrated in Figure 21b. This initial miss was probably because of the shape of the grill.
The test performed straight down from above provided the best result. Every ultrasound wave sent returned accurate measurements. This meant a 100% successful transmissions on all of the 25 vehicles. Lastly there were test performed with the sensor placed underneath the car. The results showed a 96% (24 vehicles) rate of successful transmissions. One vehicle had wrong calculations on every transmission until it was physically moved again. This was probably due to the uneven surfaces underneath. The second placement of the node was close to the first. With this new placement, the transmission was successful throughout the transmissions. These results can be viewed in Figure 22.
7 Packet Loss

One important aspect with this project is the functionality. The system should be able to perform in a sufficient way, meaning that there should not be any unwanted system behavior. One crucial part of this project is that the users should retrieve the correct information. The unwanted scenario is to have a user get information that a parking is occupied but it actually being available. There are two reasons behind this unwanted behavior. One reason being how often the system will be polling. If there are changes between the polling periods these unwanted scenarios can occur. Another reason is the packet loss in the system. If packets are lost during run time, information will be lost increasing the chances of unwanted behavior.

A balance of the polling period and the packet loss in the system should be analyzed. Given that the range contributes to packet loss an overall increase of polling in the system will compensate for the packet loss. For instance, if a system looses every fifth packet sent, one could increase the amount of packets sent during a set period of time, compensating for this packet loss. This change will increase the power consumption. It is therefore important to find a balance with polling. The ideal scenario would be to reduce the polling to a minimum and still keep a low packet loss percentage.

![Figur 23: Pictures of the parking garage where the system was placed.](image)

In Figure 23 one can view the location of the sensors and server. In the top picture (A) the entire garage width is captured. It is approximately the range shown in the top picture that the
sensor was communicating with the base node (10-15 meters). The lower left picture (B) was taken from the sensor node position. Across from this position, on the other side of the garage, is where the base node was located. The lower right picture (C) was taken from the base node position, facing the sensor node.

7.1 Test And Result

During the power consumption test, where two sensor nodes were left running for 10 (Atmega328P-PU) and 14 (Arduino Nano) days, there was also packet tracking to figure out how many packets were retrieved at the base node. In Figure 23 one can view where the sensor nodes were placed in correlation to the base node. During this test there were 28800 packets sent from the Atmega328P-PU and 40320 packets sent from the Arduino Nano. Same sensor and radio module was used.

7.1.1 Atmega328P-PU Circuit And Arduino Nano

Given that the Atmega328P-PU circuit stopped working after the tenth day, data related to the lower battery life packet performance was not documented. Analyzing the available data, one could see some packet loss. On average the package loss was 4% on a day of run-time. In total that would correspond to an hour of packet loss every day where a packet is sent every 30 seconds.

![Figure 24: Demonstrating the different sensor results](image)

The Arduino Nano ran additional 4 days and had a better packet loss rate of appropriately 2%. It got an increase of packet loss during the later days. The increase was not significant, but battery
life seemed to have a small contributing factor. From day 12 to 14, the packet loss increased to 5% a day. Even if the nodes were placed the same distance away from the base node, the different results were probably because of the different vehicle sizes that were parked. This difference certainly had some affect on the result. The Arduino Nano might have also been placed in a more unfavorable position, where other signals might disturbed the communication. During the later stages of the Arduino Nano sensor testing, there is a noticeable increase in packet loss, as seen in Figure 24, this is probable due to the decreased power provided by the power source.

8 Application

Figure 25a shows the start screen of the application developed for the end user. By tapping on "SEMCON VÄSTERÅS" one is directed to the list of parkings (Figure 25b). Important to note is the missing "Parking 2" which is occupied and does not appear in the list. When the desired parking is chosen, tapping it opens a map with the selected parking highlighted (Figure 25c). There is possibilities to view other parking garages by tapping on "KONTOR" in Figure 25a. This directs the user to a list of available parking garages, as seen in Figure 25e. A small help pop-up is available with instructions of how to use the application. It is located in the start screen as "HJÄLP" and pops up on top of the start screen, seen in Figure 25d.
Figur 25: Entire Android application
9 Discussion

This section covers different topics. Below one can find the discussion regarding the three different requirements set out for this project. There is also a section about the issues throughout the project, alternative implementation of different parts of the system, the user experience and a helpful tool that is worth a mention.

9.1 Cost And Power

There were three goals set up before developing and testing the entire system. It needed to cost less than 6 US dollars, have a low power consumption and have reasonable functionality (low packet loss rate). To conclude what was made and tested, a system was created that fulfills these three requirements. The system performs what is expected from it and delivers it with these three criteria in mind. In terms of cost it is just below 6 US dollars, making it an economical alternative to other similar products on the market. The power consumption of the sensor is in the lower, battery driven levels, fulfilling the second criteria of this thesis. The Atmega328P-PU circuit should be able to provide at least three weeks of usage.

9.2 Packet Loss

Both systems had tolerable packet loss rate, as the packet loss was spread out throughout the day rather than continuously. This meant that if a packet was lost, after 30 seconds another packet was sent and retrieved. The reason behind the packet loss could be different cars driving by blocking the signal or the type of car parked where the sensor is placed. Larger vehicles could block the communication resulting in packet loss. Overall the system functions as it should despite the packet loss, because of the 30 seconds interval between the packets.

9.3 Issues

The Atmega328P-PU circuit stopped working after the tenth day. When a further analysis of the circuit was made, a conclusion was made that the external crystal oscillator most probably was not working. The microcontroller did not respond to any code that was uploaded. The reason behind this is, as mentioned earlier, not totally clear and would require further debugging.

Given that this project had hands on circuit building, a lot of times there were wires that did not work, or components that were needed at the time, but not available. When there was a wire that was not properly inserted or was loose, plenty of time was spent locating the issue and fixing it.

9.4 User Experience

The user experience is an area that always can be improved. The system works and provides the desired information to the users. In terms of the front-end of the system (application) there still
are improvements to be made. For instance, have a picture that is updated rather than having a list of parkings. This would provide a more intuitive user experience. Further improvement can also be made in terms of system functionality. Providing the option to reserve a parking spot, before driving to the parking garage or implementing a “shortest path to the parking space” in a larger parking garage.

9.5 Alternative Implementation

One possibility is ”Low Powered Bluetooth” communication. This would allow end-users to connect with a mobile application to view data. This approach has drawback such as range and user friendliness. Because it exchanging data using short-wavelength the range is limited (from 2.4 to 2.485 GHz). This in turn forces the user to be in range of the base-node, restricting the users from the data if they are out of range. The benefit of ”Low Powered Bluetooth” is that there is no need for a server or Internet connection. There would only exist a Bluetooth connection.

Similarly to Bluetooth in terms of range one can use a local web server. The server would allow users to connect to it and fetch data. A requirement for this would be that both, the server and end-users, would have to be connected to the same network. This means that users need to be in range of the network in order to connect through Wi-Fi. One benefit with this method is that availability depends on the Wi-Fi range. If there is good Wi-Fi coverage in a larger building users will be able to retrieve data without being nearby the base-node. Using Bluetooth users would need to be physically close to the base node.

A public web server would provide the most flexibility. This would allow retrieval of data independently of physical location. All requirements needed is that the users have Internet access. The drawback with this method would be security. In order to implement a public server, port forwarding is required. All ports that are opened become a potential security risk. If a malicious user would get a hold of the Internet IP of the server they would be able to manipulate data, any connected switches or other components. This can be protected with password or other authentication methods. This project does not aim to implement or analyze security aspects of the system, but rather point out potential risks.

9.6 Helpful Tools

Blynk is a application allowing remote usage and monitoring of Arduino without the need of a Ethernet shield or any other additional add ons. This allows for instance monitoring on the Blynk mobile application. In the beginning of this project the base node used to collect all the data from the other sensors was a Arduino Uno Rev3 directly connected to a laptop. In order to place the sensor nodes and test them without the need of carrying around a laptop to read the data received on the base node, an mobile application was used. This allowed monitoring directly on the mobile.

The information sent from the Arduino to the laptop is forwarded to the Blynk cloud, which is synced with an account. As soon there is a connection established everything sent to the
computer is forwarded to the cloud where it is then displayed to that specific account. These accounts can create projects and each project has a "Auth Token" that is sent via email when a project has been created.

9.6.1 Setting Up

This was set up on a MacBook Pro (13-inch, 2016) with macOS Sierra version 10.12.6. With the provided Blynk libraries, tools, token and website tutorial a connection can be established. In addition to this "socat" was needed before any connection could be made. Socat is a command line based utility that establishes two bidirectional byte streams and transfers data between them. Socat was installed using "Homebrew".

10 Conclusion And Future Work

In summary, this thesis set out to test, document and develop a system with a set of requirements in mind. The system was developed and tested. Based on these tests, the system achieves the three requirements (economical, power consumption, functionality).

This thesis is a good foundation for further development. There is room to expand on multiple fronts. A booking system could be implemented, where users can book parkings right before driving to the parking garage. There are possibilities of implementing and using RFID technology to charge for the parked time, or to protect the parking from unauthorized people using the parking space. If for example, there is no RFID tag used, a notification can be sent to a responsible organization and a ticket could be issued. Another possibility is to implement a "fastest route algorithm" as seen in section "Related Work". This could help in larger parking garages in order to streamline the user experience.

There are many ways this project can be expanded upon to help contribute to the Urban IoT. Hopefully this thesis is some of that groundwork done and other ideas can build upon this.
Referenser


