Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

General-Purpose Wireless Distance Sensor

Semester Thesis

Roland Meier
meierrol@ee.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Pascal Bissig, Dr. Samuel Welten
Prof. Dr. Roger Wattenhofer

February 25, 2014

Abstract

Thanks to the availability of different wireless communication techniques, external devices can extend today's smartphones. Such an external device can be any kind of a sensor, for example.

This thesis describes a general-purpose wireless distance sensor, which connects to a smartphone or any other device equipped with Bluetooth technology. Further, an Android application which handles the measurements from multiple sensors and visualizes them is developed.

Several use-cases for this sensor are described. One example is to use a couple of sensors as a parking assistant in a home vehicle hall. The sensor devices as well as the smartphone application are developed mainly with this use-case in mind.

The thesis describes the trade-off between energy consumption and latency and shows how a practical solution can be reached.

The sensor devices are evaluated with respect to energy consumption, interference between multiple sensors, disturbances as well as the connection establishment time.

Contents

Chapter 1

Introduction

Today's smartphones are equipped with many sensors. As for example, sensors for temperature, gravity, magnetic field and light [1]. However, in addition to using the built-in sensors, it is possible to extend a smartphone with external sensors, which can be connected via Bluetooth for example.

The use-cases described in the following section motivate the development of general-purpose wireless distance sensors and a corresponding smartphone application.

1.1 Use-Cases

1.1.1 Car Parking Assistant

Modern cars are more and more characterized by their electronic assistants. Regarding the parking procedure, there are systems to help parking or even to park the car autonomously. All these systems have at least one thing in common: they are built into new cars and they are not designed to be integrated in an older car (except some third party systems which are difficult to integrate). The opposite is true for the distance sensor developed in this thesis: it can easily be mounted to every car (for example by using a magnet) and does not need any wired connections or power supply as it has an integrated battery. The data are sent to the driver's smartphone and displayed in a flexible and extensible application. Consider Figure 1.1 for an example how three distance sensors could be positioned on a car.

1.1.2 Parking Deck Management

Some parking deck operators have already implemented a system where a small signal light over each parking area indicates whether the place is occupied (described for example in [2]). This signal helps arriving drivers to find a free place

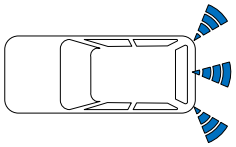

Figure 1.1: Distance sensors mounted on a car

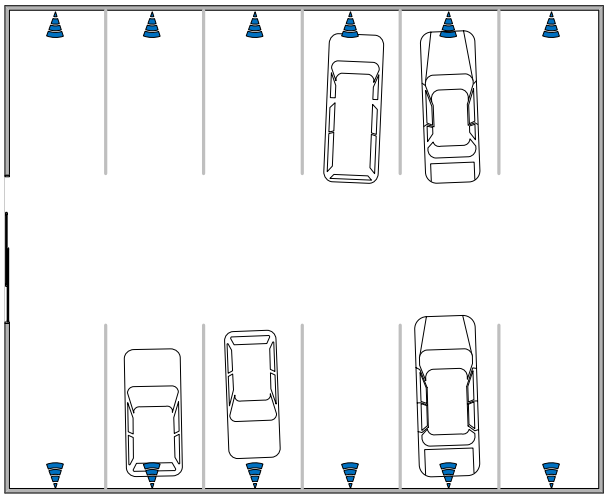

Figure 1.2: Distance sensors mounted in a parking deck

by looking for a green light on the ceiling. Using the distance sensor developed in this thesis would allow the following extension. A distance sensor is mounted on the wall behind each parking area (as depicted in Figure 1.2). As soon as a car drives into this parking area, the sensor connects to the driver's smartphone and informs the driver about the current interspace to the wall.

1.1.3 Home Vehicle Hall

Similar to the scenario in the parking deck, sensors can also assist in a private vehicle hall. Since the application is designed to handle data from multiple sensors, it is also possible to mount for example a sensor on each wall of the vehicle hall (Figure 1.3) in order to provide more information to the driver.

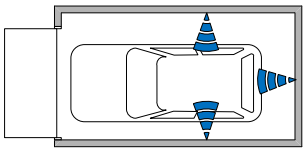

Figure 1.3: Distance sensors mounted in a home vehicle hall

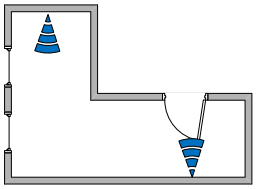

Figure 1.4: Distance sensors mounted in an appartment

Figure 1.5: Distance sensor mounted in a tank

1.1.4 Surveillance

Distance measurements can be used for various surveillance tasks, too. A change in the measured distance can for example mean that

- Somebody is inside a room
- A door was opened
- An object was moved (for example, the car in the vehicle hall or an object in an exhibition)

Figure 1.4 shows an example of how two distance sensors can be used to monitor a room.

1.1.5 Liquid Level Measurement

Measuring the liquid level can be achieved by placing a distance sensor on the upper end of a tank, for example (see Figure 1.5). This could be used in buildings heated by fuel oil. The sensor could then send an alert to a smartphone as soon as the oil level is below a predefined value.

1.1.6 Letterbox

In a similar way as measuring liquid level, one could also measure the height inside a letterbox. A reduction of the measured height does then indicate that something was delivered into the letterbox. This way, the inhabitant does not need to go to the letterbox but can check from inside the apartment whether he or she received a packet.

1.2 Related Work

Numerous ultrasonic-based parking assistant systems already exist. They can be classified in systems that are mounted on the car (such as [3] (displays data on an external display) and [4] (displays data on built-in radio and navigation-display), both powered by the car battery) and systems that are mounted in the garage (such as [5] and [6], which are both powered by battery and visualizing the measurements through a signal light mounted on the wall of the vehicle hall. Both can handle only one sensor.). There are also already described projects using an Arduino to realize a garage parking assistant [7]. However, to the author's best knowledge, there is no publicly described project of a parking assistant that uses a smartphone to visualize distance measurements.

The same holds for the other described use-cases: there are existing solutions, but they are not (only) consisting of sensors and a smartphone.

The sensor devices and the application described in this report are designed such that they can be used in different situations such as the previously described use-cases and others.

1.3 Goals and Contributions

The goal of this semester thesis is to build general-purpose wireless distance sensors. Furthermore, an Android application to display data from multiple sensors in an attractive way is developed.

The distance sensors are designed to be used in use-cases such as the ones described in Section 1.1. This requires that they are

- small (such that they can be mounted on a car, for example)
- battery-powered (the need for an external power supply would dramatically reduce the flexibility)
- easy to handle (knowing how to pair a Bluetooth device with a smartphone should be enough to use the sensors).

The fact that the sensor devices are battery-powered further requires special attention on the energy consumption in order to run as long as possible without needing to recharge the battery. Regardless of the hardware components used, the mode of operation will also influence the energy consumption of a sensor device. For example, instead of taking distance measurements continuously, the sensor device can take measurements only once in a certain time interval. But while reducing the energy consumption, this would increase the latency of the sensor devices. Therefore, a reasonable trade-off must be found.

This report describes the implementation of the sensor devices using commercial off-the-shelf hardware components as well as the implementation of the smartphone application. Furthermore, the implemented components are evaluated with respect to performance aspects (such as their susceptibility to interferences with other ultrasonic sensors), energy consumption and cost. The report is finalized by the conclusion which describes the applicability of the implemented hard- and software to the use-cases described in Section 1.1.

Chapter 2

Implementation

2.1 Sensor Device

The sensor device's task is to perform distance measurements and send the data via Bluetooth to a smartphone. This must be done in an energy-efficient way since a battery powers the device. The main components of each sensor device are

- Ultrasonic distance sensor. A commercial off-the-shelf (COTS) sensor is used which performs the measurements and provides the results through different interfaces.
- Bluetooth module. The used module acts as a Bluetooth slave device and provides a serial interface to the microcontroller.
- Microcontroller. For the sake of flexibility, an Arduino board is used.

The following subsections will describe the different components and their interplay in more detail.

2.1.1 Ultrasonic Distance Sensor

To measure the distance between the sensor and the next object, an ultrasonic sensor uses sound waves at frequencies above 20'000 Hz, which is beyond the range of human hearing. The sensor measures the distance by emitting a sound wave and then listening until an echo of the sound wave arrives. The round trip time – the time between sending the sound wave and receiving the echo – can then be used to calculate the physical distance between the sensor and the targeted object [8].

Compared to optical sensors, ultrasonic sensors have the advantages that they can also be used for translucent objects such as windows (see use-case in Section

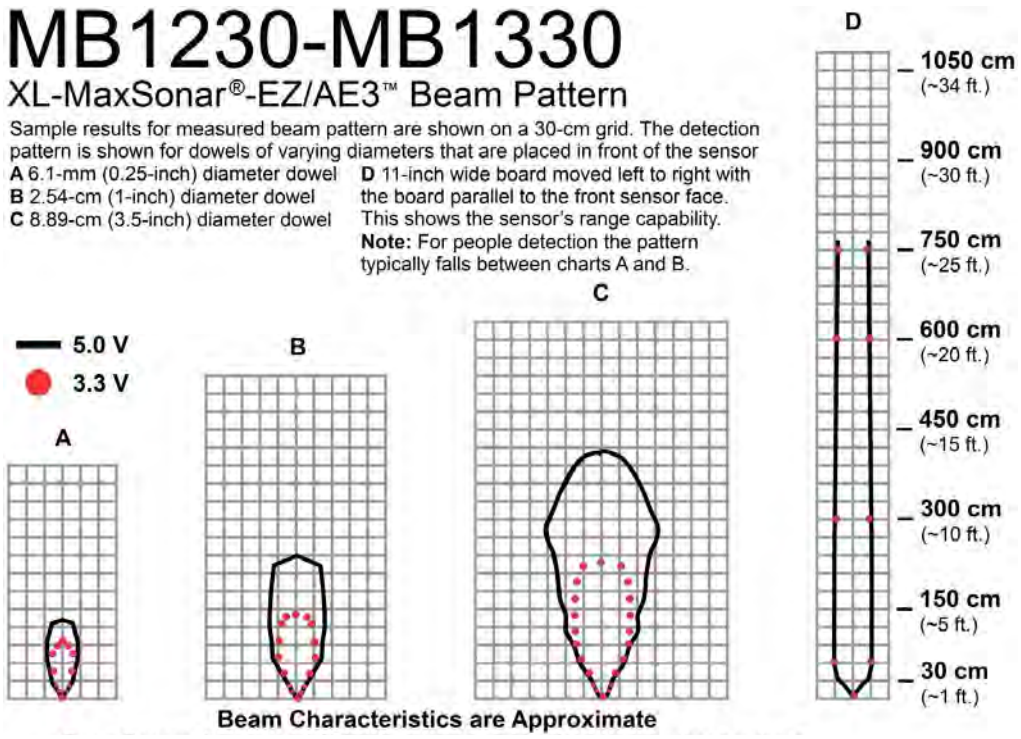

Figure 2.1: XL-Maxsonar EZ3 beam pattern [9]

1.1.4) and liquids (see use-case in Section 1.1.5) and highly reflective or metallic surfaces (such as a car, see use-cases in Section 1.1.3 and 1.1.2). However, unlike optical sensors, ultrasonic sensors are susceptible to temperature fluctuations or wind [8].

The ultrasonic sensor used in this project is a "XL-Maxsonar EZ3" [9] (see Figure 2.2). It can measure distances from 20 cm up to 765 cm with a resolution of 1 cm and provides access to the measurement data through different interfaces (RS-232, analog, pulse width). The sensor operates at 42 kHz and its supply voltage can be anywhere between 3.3 V and 5 V. The sensor's beam pattern for targets in different sizes is depicted in Figure 2.1.

2.1.2 Bluetooth Connection

Regarding wireless communication techniques, most modern smartphones are equipped with cellular, Wi-Fi, Bluetooth and probably NFC. The choice to use Bluetooth for the communication between the smartphone and the sensors is based on the following facts:

- Cellular connections are not feasible because they need a SIM and are too

complex to implement.

- Bluetooth is much less energy-consuming than Wi-Fi [11]
- The range of an NFC connection is only about 10 cm [12], which is far too small for this application.

Bluetooth allows to easily transmit data from multiple devices (in this project: multiple sensors transmit data to one smartphone) over short distances. Before two Bluetooth-enabled devices can communicate, they need to be paired [13].

The used Bluetooth module for this project is a "BT2S Bluetooth to Serial Slave" by Virtualbotix. It is a simple module which allows to transmit data via Bluetooth as if one would transmit it via a serial interface. It acts as a slave device, which means that the sensor device will not be able to initiate a connection but the smartphone must do it [14].

2.1.3 Arduino Microcontroller

To process the measurements from the ultrasonic sensor and send them to the Bluetooth module as well as to turn the sensor and the Bluetooth module on and off, a microcontroller is needed. Because of its flexible and easy-to-use design, the Arduino platform is used.

Arduino is a physical computing platform which is released under open-source licence and bases on a simple microcontroller board and a development environment [15].

Numerous variants of Arduino boards exist, which differ in terms of size, CPU speed, interfaces, memory, power consumption and others [16]. For the purpose of a wireless distance sensor, the power consumption (because the sensor is battery-powered) and the size (the sensor device should be small) are important while other attributes like CPU speed and memory size are less relevant.

Based on the above mentioned criteria, the "Arduino Pro Mini" built by Sparkfun (see Figure 2.3) is a suitable board for this project. It is equipped with an ATmega328 CPU running at 8 MHz, can be powered at any voltage between 3.3 V and 12 V and its size is only 18×33 mm [17]. Further, the number of analog (8) and digital (14) I/O pins [17] will be more than sufficient to connect and control the ultrasonic sensor and the Bluetooth module.

2.1.4 Sensor Device Prototype

The previously described components are put together to build a prototype of the wireless distance sensor. Besides these components, only two transistors and

Figure 2.2: XL-Maxsonar EZ3 ultrasonic distance sensor [10]

Figure 2.3: Arduino Pro Mini [17]

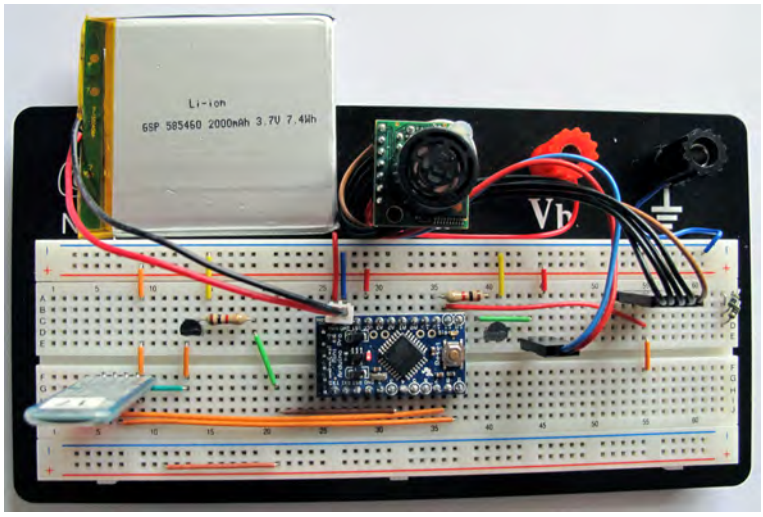

Figure 2.4: First prototype on a breadboard

two resistors are needed to turn the ultrasonic sensor and the Bluetooth module on and off. Being able to turn these components on and off is important to save energy. Figure 2.7 depicts the connection scheme. The first prototype was built on a breadboard (Figure 2.4), while the second prototype was created more compact and integrated into a small box (Figure 2.5 and 2.6).

2.2 Arduino Programming

The purpose of the Arduino microcontroller is to receive the measurement data from the ultrasonic sensor, forward them to the Bluetooth module as well as to turn on and off the sensor and the Bluetooth module depending on whether they are used. The software running on the Arduino can be described in a simple flow chart, which is shown in Figure 2.8. Described in words, the Arduino does the following:

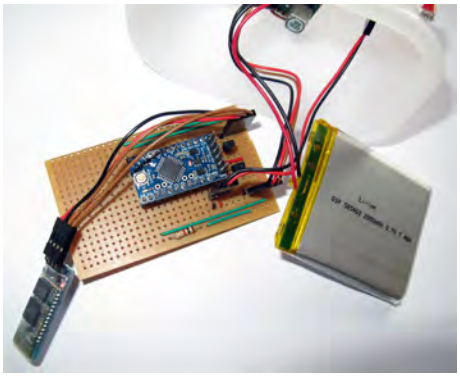

Figure 2.5: Second prototype electronics

Figure 2.6: Second prototype in box

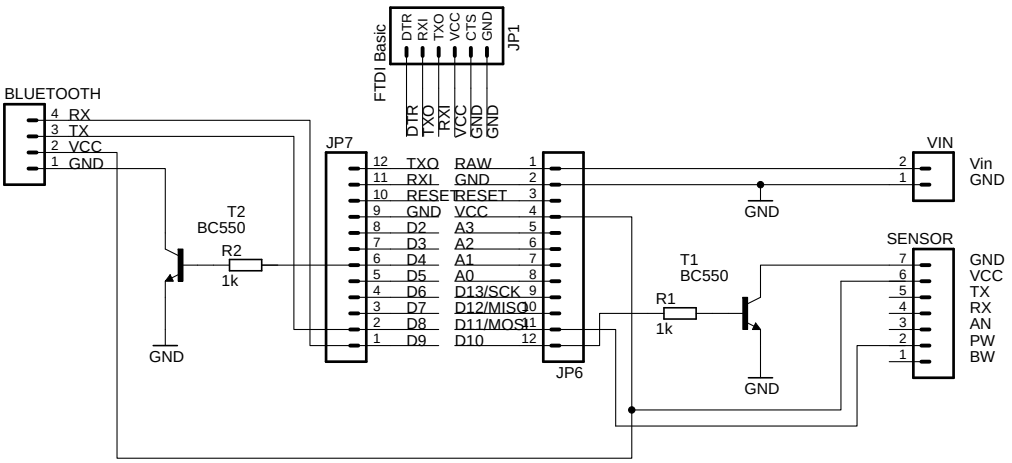

Figure 2.7: Connection scheme of a sensor device

1. After initializing, the distance sensor is powered on and performs a measurement.
2. If the measured distance differs by more than 50 cm from the measured distance in the previous cycle, turn on the Bluetooth module (if it is not already running).
3. If the difference is less than 50 cm, decrease a counter. When this counter reaches zero, the Bluetooth module is turned off. The resulting behaviour is that the sensor transmits C (the initial value of the counter) measurements after it recognized the last big difference (> 50 cm). For example, the big difference would occur when a car drives into the garage, followed by many small differences when the car manoeuvres to the right position.
4. Turn off the sensor.
5. Send the measured distance to the Bluetooth module.
6. Sleep for T seconds, then go to step 2.

Implementing the described behaviour is straightforward. However, two aspects are discussed in more detail: the encoding of the distance measurements and the sleep mode.

2.2.1 Encoding Data for Bluetooth Transmission

Bluetooth transmission is done by using the "SoftwareSerial" library, which provides an easy way to read and write data from/to the serial interface where the Bluetooth module is connected.

To implement the desired behaviour of the distance sensor, only the write operation is needed. The measurement data are encoded as strings of the following format: `{[key],[value]}`. For the distance measurements, the key `d` is chosen. A possible message is therefore `{d,123}` which would report a measured distance of 123 cm. This encoding scheme allows to define additional message types with different keys, even though this is not necessary for the basic implementation of the distance sensor.

2.2.2 Saving Energy Through Sleeping

Due to the fact that the sensor device is battery-powered, it must consume as little energy as possible. In each of the use-cases described in Section 1.1, the sensor device will not need to do anything most of the time. Only after something relevant happens (for example, a car gets near the sensor), the device needs to become active and transmit measurements. Therefore, for most of the time it is

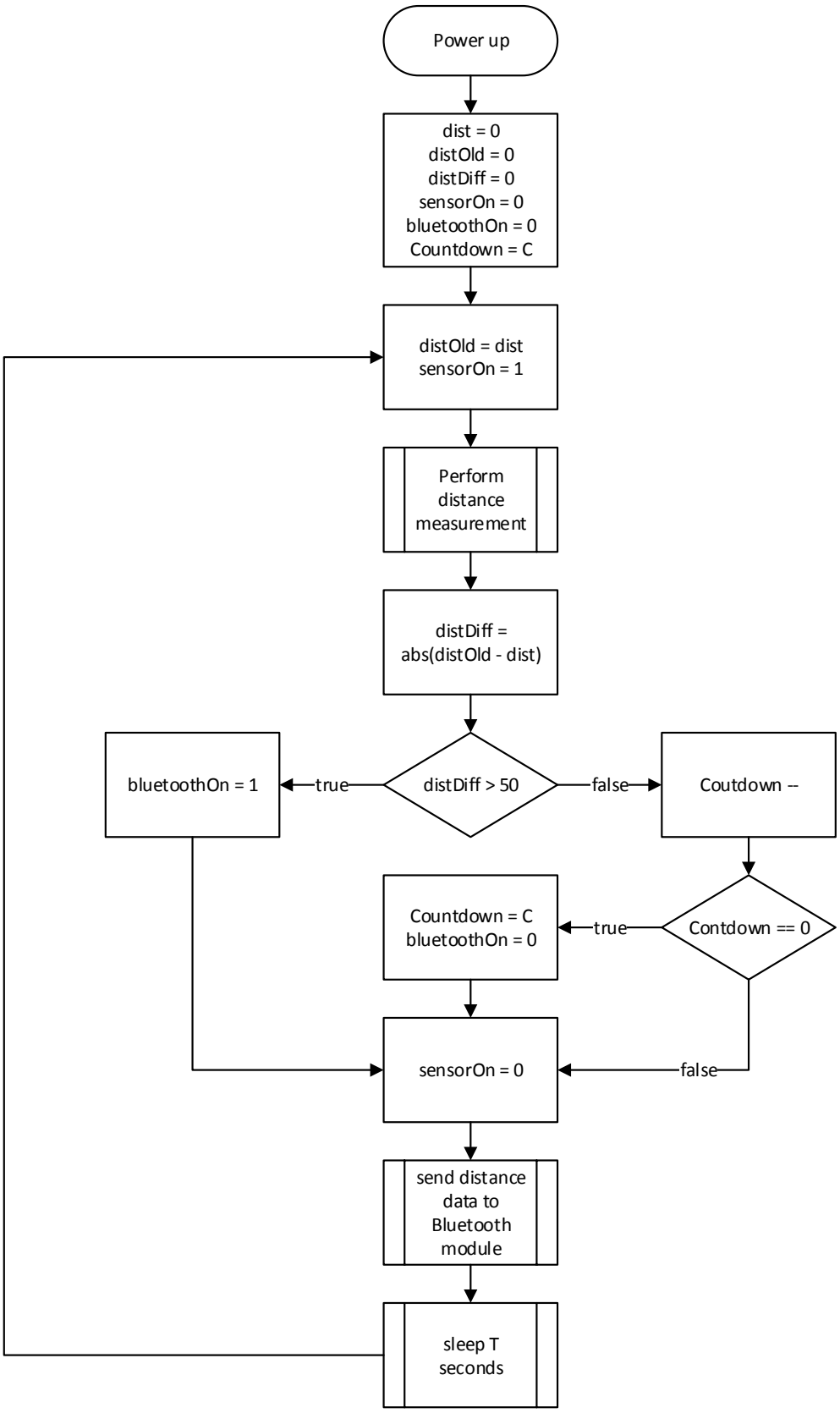

Figure 2.8: Flow chart of the Arduino's behaviour

sufficient if the sensor performs periodic measurements (in intervals of a couple of seconds for example) and only becomes active if it detects a relevant change. Becoming active in this context means that the Bluetooth module is turned on and the transmission of the measurements starts.

The ATmega328 processor, which is built in the used Arduino board, has six sleep modes [18]. Since there is no other source which could wake the Arduino, a watchdog timer is used. The sleep mode, which saves the most energy but still provides the possibility to wake up by a watchdog timer is the "Power-down" mode. That is why this mode is used in the sensor device.

2.3 Smartphone Application

In summary, the smartphone application must provide the following functionality:

- Connect via Bluetooth to multiple distance sensors
- Visualize the received data in a customizable way

Because of its popularity and open design, Android is selected as target operating system for the application. In general, any device, which is capable of communicating through Bluetooth Serial Port Profile (SPP), can receive the sensor data.

2.3.1 User Interface

The user interface is split into three tabs:

- Sensors-tab: displays the state (connected or not) of paired sensors and allows to manually connect to these sensors (see Figure 2.9).
- Data-tab: shows a graphical representation of the received sensor data (see Figure 2.10).
- Settings-tab: allows to change the position and orientation of the sensor gauges for the graphical representation (see Figure 2.11).

2.3.2 Bluetooth Connections to Sensors

Before a distance sensor can be used in the application, it needs to be paired with the smartphone. After this is done, the sensor is listed in the sensors-tab

Figure 2.9: Sensors-tab

Figure 2.10: Data-tab

Figure 2.11: Settings-tab

and is considered when the application searches for available sensors. Connecting to available sensors is done either manually, when the user clicks on the button (Figure 2.9) or automatically, when the application is started or the user returns to the application (`onResume`-event in Android). Since the application should only connect to distance sensors but not to other paired Bluetooth devices, there is a need for a mechanism to identify distance sensors. A simple way to do this is to add a predefined prefix to the name of each distance sensor. Doing so, the prototypes of the distance sensors are named in the following way: `DistSens_[last two hexadecimal characters of the MAC address]`. This led to the names `DistSens_17` and `DistSens_79`.

The described approach allows a very user-friendly way to add or remove sensor devices. To add a new sensor, the user simply needs to pair it with the smartphone. Unpairing leads to removing the sensor.

2.3.3 Handling Received Sensor Data

The application receives data from the sensors by a serial Bluetooth connection and handles them globally. The class that handles received sensor data maintains a list of the most recent distance measurement from each connected sensor and provides notifications through listeners when a new measurement is received.

2.3.4 Visualize Sensor Measurements

The received distance measurements from the sensor devices are visualized in a graphical way by showing a gauge for each active sensor in the data-tab (see Figure 2.10). The position and orientation of the gauges can be defined in the settings-tab. This provides the flexibility needed in order to use the application in different contexts and with a different number of sensors. For example, the sensor gauges can be positioned in the same arrangement as they are mounted in the garage.

Each gauge indicates the distance in eleven discrete steps (0% to 100% in steps of 10%). Although the distance sensor can measure distances up to more than seven meters (Section 2.1.1), for many use-cases it is not reasonable to cover the whole sensor range with a gauge. In the car parking scenario, for example, a driver is not really interested if the distance to a wall is five or seven meters. This is why the application defines a visualization range, which is then linearly mapped to a gauge level. Looking again at the car parking scenario, a visualization range of 20 cm to 2 m may be reasonable. This means, that distance measurements greater than two meters are considered as infinitely wide away (100% gauge) and distances below 20 cm are considered as infinitely close (0% gauge). For different use-cases, the mapping can be adapted.

Chapter 3

Evaluation

In this section, the performance of the sensor device, the smartphone application and their interplay are evaluated. More precisely, the following questions are answered:

- Do multiple sensors cause problems because of interference?
- How long does it take to establish a Bluetooth connection between sensor device and smartphone?
- How long can the sensor device be powered without recharging the battery?
- What are the cost of a sensor device?

3.1 Performance in Presence of Multiple Sensors

Since all the ultrasonic sensors operate on the same frequency (42 kHz [9]), simultaneous operation would affect the measurement. However, since the sensors do not perform measurements permanently, but only once every couple of seconds, the chance that two sensors perform a measurement at the same time is reduced. In the current implementation of the sensor device, a measurement is performed roughly every two seconds and takes 0.3 seconds (see Figure 3.1). Additionally, to be vulnerable to such disturbances, the sensors must be arranged such that one sensor receives the emitted ultrasound waves by the other. For example, this is the case if the two sensors are arranged in parallel and close to each other or if they are mounted on two opposed walls of a garage.

3.2 Establishing a Connection

Experiments show that the time for establishing a Bluetooth connection (time between Bluetooth module is switched on and the connection with a listening

and paired smartphone is established) is always around eight seconds. This is true for different scenarios like various distances between sensor and smartphone and various environments (in a building, in free-space).

3.3 Energy Consumption and Battery Lifetime

If powered, the sensor device is always in one of three operation modes:

- StandBy: It periodically performs measurements to detect a relevant event like a vehicle coming close. The Bluetooth module is turned off.
- BT advertising: The Bluetooth module is turned on and advertises itself such that a smartphone could establish a connection.
- Connected: The sensor device is connected to a smartphone and the measured distances are transmitted.

The measured power characteristics of these operation modes are plotted in Figure 3.1. Especially in the "StandBy" and the "Connected" mode, clear spikes show the intervals when the ultrasonic sensor is active.

Figure 3.2 shows a plot which relates the sleeping duration between two measurements to the average power in "StandBy"-mode. Figure 3.3 shows a plot, which relates the latency (sleeping duration + measurement duration) to the battery lifetime with the currently used battery (2000 Wh at 3.7 V). The current implementation of the sensor device operates with a sleeping duration of about two seconds.

3.4 Cost of a Sensor Device

Table 3.1 shows the calculation of the cost for the hardware needed to build one of the prototype sensors.

It is noticeable that the ultrasonic range sensor causes more than half of the total cost.

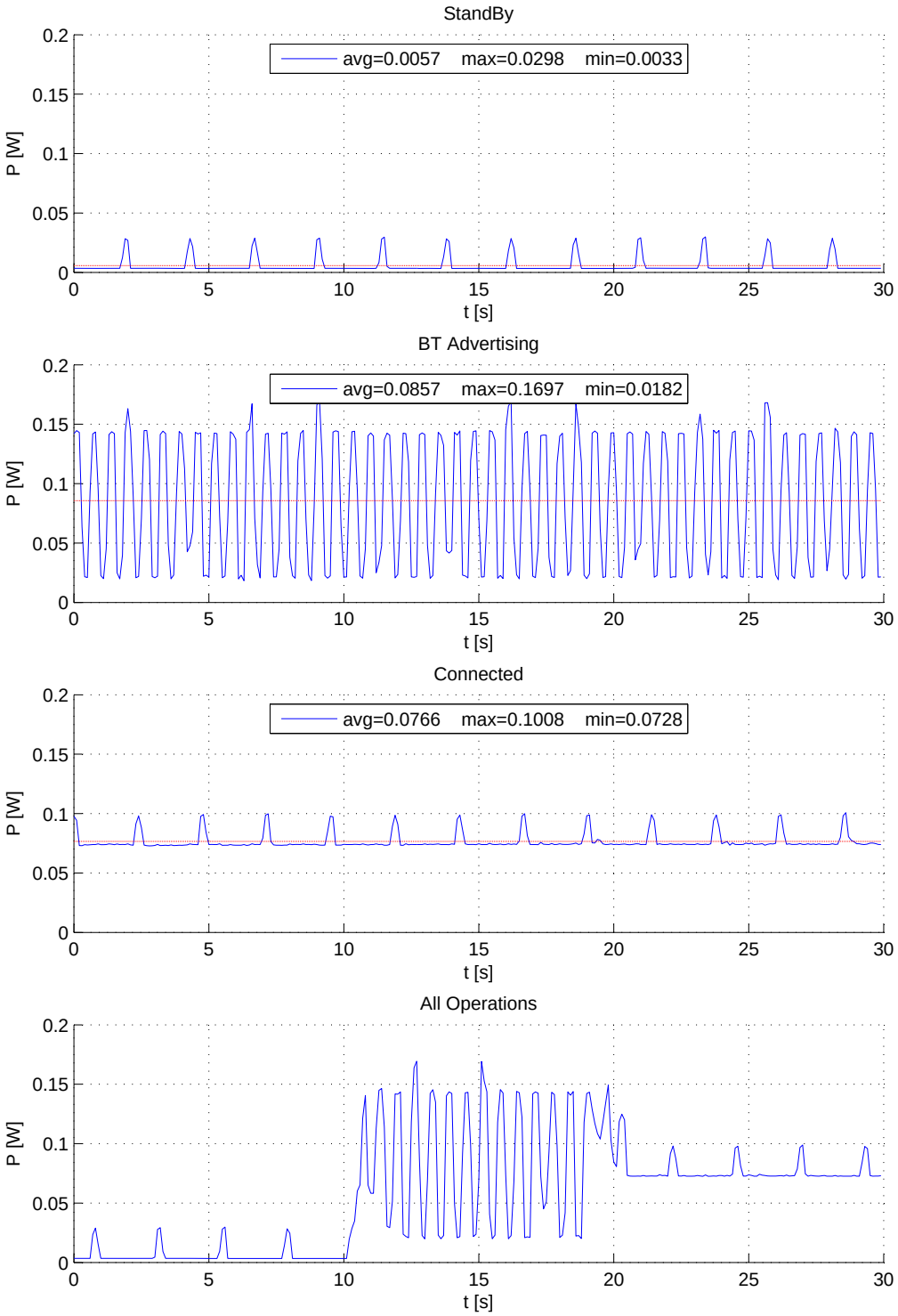

Figure 3.1: Power in different modes: "StandBy" (only periodic measurements), "BT Advertising" (Bluetooth module active and advertising) and "Connected" (connection between sensor device and smartphone established). Red line: average power.

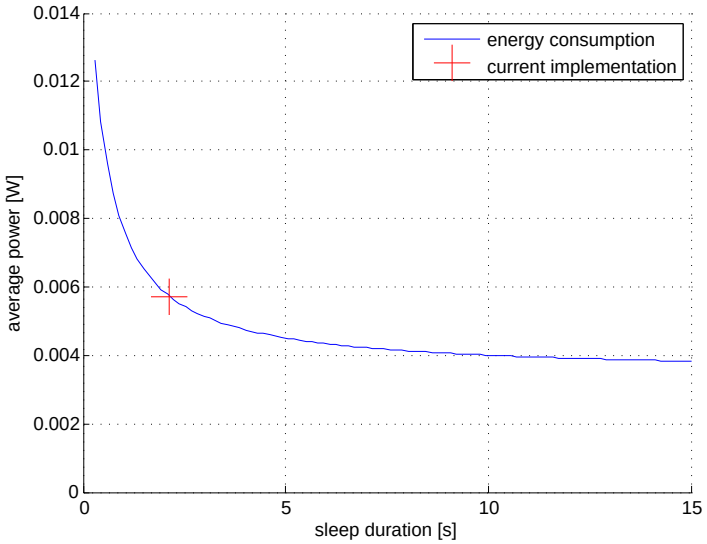

Figure 3.2: Relation between sleep duration and average power in "StandBy"-mode

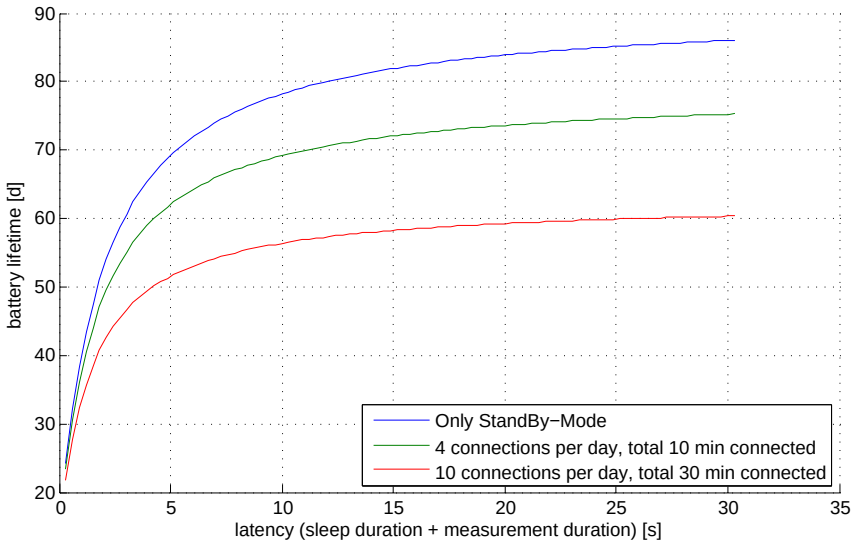

Figure 3.3: Relation between latency (sleeping duration + measurement duration) and the battery lifetime for different scenarios. The topmost curve represents a scenario where the sensor is in "StandBy"-mode the whole day, while the middle curve is calculated for a scenario with four connections per day and ten minutes total connected time and the lowermost curve is calculated for a scenario with ten connections per day and 30 minutes total connected time.

Table 3.1: Price calculation for one sensor device. Prices: Bluetooth module [21], others [20]

Amount	**Component**	**Price per item**
1	Arduino Pro Mini	$9.95
1	Ultrasonic Range Finder	$49.95
1	Bluetooth Module	$14.95
2	BJT Transistor	$0.95
2	1kΩ Resistor	$0.25
1	Breadboard	$3.95
1	Battery	$16.95
Total		**$98.15**

Chapter 4

Discussion

4.1 Performance in Presence of Multiple Sensors

Assuming a uniformly distributed power-up time of the sensor devices and that a distance measurement takes 0.3 seconds and taking into account that the sleep duration is two seconds, the probability that two sensors disturb each other is upper bounded by the following calculation:

$$P[\text{disturbance}] \leq \frac{t_{\text{window}}}{t_{\text{measurement}} + t_{\text{sleep}}} = \frac{2 * t_{\text{measurement}}}{t_{\text{measurement}} + t_{\text{sleep}}} = \frac{2 * 0.3s}{0.3s + 2s} = 0.26$$

This is only an upper bound because disturbance can only occur when a sensor is listening for an echo. At the very beginning of a measurement, when a sensor sends the ultrasonic signal, it does not listen for an echo. This is why t_{window} is slightly reduced in reality. Furthermore, an ultrasonic wave propagates 15 m (twice the highest measurable distance of the used sensor) in only about 45 ms. Assuming that the sensor needs to send only one ultrasonic wave for each measurement, the collision domain would be dramatically reduced.

Although the calculated probability seems to be high, it is not too much a problem in practice since such disturbances only occur if the sensors are arranged in a way they can influence each other.

To avoid disturbances by multiple sensors completely, there would be need to coordinate the measurements. This could be done by the smartphone: instead of receiving measurements when they are provided, the smartphone could explicitly ask for a measurement one sensor after another.

4.2 Establishing a Connection

For the car-related scenarios described in Section 1.1, eight seconds to establish a connection is a sufficiently short time. The liquid-level measurement use-case is

even less time-critical. Therefore, eight seconds are also feasible in this scenario.

If used for surveillance purposes, a delay of eight seconds might be too high. However, this is not the only problem which arises in this use-case. For example, a burglar could cut the sensor device's power supply or jam the Bluetooth signal [22]. Because of these risks, there is a need for additional measures which might require a permanent Bluetooth connection. This would then make the connection establishment time irrelevant.

4.3 Energy Consumption and Battery Lifetime

Since a battery powers the sensor devices, their energy consumption should be as low as possible. Given the built prototypes, the energy consumption can be fine-tuned by varying the sleep duration between two measurements. Enlarging the sleep duration results in a lower energy consumption but also in a higher response time. Figure 3.2 shows the dependency between sleep duration and average power while Figure 3.3 relates the latency to the battery lifetime. For the current implementation, a sleep duration of two seconds is chosen as a reasonable trade-off between energy consumption and response time.

The battery runtime is calculated for the scenario where the distance sensor is mounted in a home vehicle hall with the following assumptions. The sensor is active four times a day: when the driver goes to work in the morning, comes home for lunch, goes to work after lunch, and comes home in the evening. Every time, the sensor is active for 150 seconds. The rest of the day, the sensor is in "StandBy"-mode with a sleep duration of two seconds. In this scenario, the used battery with a capacity of 2000 mAh at 3.7 V can power the sensor device for about 50 days as can be seen in Figure 3.3. Therefore, the user needs to charge the battery only once every 50 days, which is very user-friendly. Especially in use-cases where the size of a sensor device does not matter, bigger or multiple batteries can be used to further enlarge the runtime.

4.4 Cost of a Sensor Device

A price of almost $100 is too much for a single distance sensor to be used in the described scenarios. However, the cost can be dramatically reduced by using cheaper components and building the sensors at larger scale. As an example, the ultrasonic sensor that causes more than half of the total price, could be replaced by a cheaper one (ultrasonic sensors are available for only a couple of dollars [23]). While the developed prototypes provide some flexibility for experiments (Arduino, ultrasonic sensor with various outputs), this flexibility would not be needed in a final product. This makes it possible to use smaller and even more energy-efficient components.

Chapter 5

Conclusion and Outlook

The goal of this thesis was to develop general-purpose wireless distance sensors and a corresponding smartphone application. As described in the report, this goal was accomplished. However, several aspects could be optimized or extended:

Energy consumption With respect to their suitability for daily use, the energy consumption turns out to be one – if not the most – crucial aspects of the sensor devices. To further lower the energy consumption and thus to extend the battery lifetime, the following approaches can be considered:

- Use an application-specific microprocessor instead of the general-purpose Arduino platform. Using a microprocessor which provides exactly the functionality needed for the sensor device might significantly lower the energy consumption in "StandBy"-mode.
- Use an infrared sensor for approximate measurements. Depending on the application context, an infrared sensor could be added to the sensor device to perform approximate measurements in the "StandBy"-mode. As soon as the infrared measurements indicate a new object, the ultrasonic sensor is enabled to perform more accurate measurements. Since an infrared diode consumes less power than the ultrasonic sensor, this extension would save energy in "StandBy"-mode. However, infrared-related issues like the limited range and problems with detecting dark objects must be considered.
- Use Bluetooth Low Energy. Using Bluetooth Low Energy would reduce the energy consumption during the time a sensor device is connected to a smartphone.
- Set different sleep durations for the "StandBy" and the "Connected" state. For example in the car-related use-cases, it might be suitable to take measurements in large intervals if nothing happens and to take measurements in shorter intervals as soon as a smartphone is connected and a car comes close to the sensor.

Maintenance and configuration functionality The current implementation of the sensor devices does only transmit measurement data and can neither receive commands from the smartphone nor does it provide any configuration possibility except by changing the Arduino-code. For a better usability in practice, some additional informations and functions are imaginable:

- Report the battery status to the smartphone
- Allow configuration of the sleep duration
- Allow configuration of threshold which activates the sensor
- Allow the smartphone to switch the Bluetooth module off

There are even more necessities if the sensor device is extended to work in more use-cases (see also the conclusions on the use-cases in Section 5.1).

Coordinated measurements to avoid interferences In order to avoid problems because of interferences between multiple ultrasonic sensors, the sensor devices could be changed in a way that measurements are only executed when the smartphone asks for it. If the smartphone requested a measurement from each connected sensor in sequential order, two connected ultrasonic sensors would never be active simultaneously.

5.1 Use-Cases

Looking at the use-cases described in Section 1.1, it turns out that the developed sensors and the application can be used for the car parking assistant (Section 1.1.1) and the home vehicle hall (Section 1.1.3) without any modification.

In order to use the system in a parking deck, the problem might arise that there are many active sensors reachable but the application should only connect to the nearest one. One trivial but not very user-friendly solution to this problem is to ask the user which parking area he is going to use. Knowing the number of the parking area, the smartphone can then establish a connection to the respective sensor (to do so, a sensor could advertise itself with the parking area number in its name).

For the security-related use-cases, several modifications to the sensor device as well as to the application would be necessary. For example, a sensor device which wants to raise an alert should be able to initiate a connection to the smartphone and therefore must act as a Bluetooth master device, which is not possible with the used Bluetooth module. For further examples, the devices must be protected against jamming and power outages.

To measure a liquid level or the height inside a letterbox, there is a need for a mechanism to make sure that the Bluetooth-module is in advertising mode at the time the smartphone wants to establish a connection. In both cases, the sensor device does not know whether the smartphone currently wants to establish a connection. To solve this problem, the Bluetooth slave module could be replaced by a Bluetooth master module, which is then capable of establishing a connection to a smartphone. If the measured height is below or above a certain threshold, the sensor device could then try to establish a connection to the smartphone every five minutes, for example.

Bibliography

[1] "SensorManager — AndroidDevelopers", Google. Last access: 13.02.2014 [online]. URL: http://developer.android.com/reference/android/hardware/SensorManager.html

[2] "London Westfield SuperMail", Bagus Prakoso. Last access: 13.02.2014 [online]. URL: http://www.gangsenggol.com/2008/11/12/london-westfield-supermall-just-make-sure-you-dont-get-separated

[3] "Einparksystem mit 6 Sensoren", car-media.ch. Last access: 13.02.2014 [online]. URL: http://www.car-media.ch/shop/pi/EINPARKSYSTEM-mit-6.html

[4] "OEM VW Touran OPS Parking Sensor System Front + Rear", CarsEquipment. Last access: 13.02.2014 [online]. URL: http://www.cars-equipment.com/www/en/shop/park-pilot-sets-kits-2/oem-vw-touran-ops-parking-sensor

[5] "LANDI: Garagen Parksensor", Landi. Last access: 13.02.2014 [online]. URL: http://www.landi.ch/laden/deu/garagen-parksensor_1171183.shtml

[6] "Ultraschall-Garagen-Einparkhilfe GEH100", ELV journal. Last access: 13.02.2014 [online]. URL: http://www.elv.ch/Ultraschall-Garagen-Einparkhilfe-GEH100/x.aspx/cid_726/detail_32136

[7] "Ultrasonic Garage Parking Assistant with Arduino and an ATtiny85", rclymer. Last access: 13.02.2014 [online]. URL: http://www.instructables.com/id/Ultrasonic-Garage-Parking-Assistant-with-Arduino/?lang=de

[8] "Q&A: Ultrasonics Basics", Banner Engineering. Last access: 13.02.2014 [online]. URL: http://www.bannerengineering.com/training/faq.php?faqID=34

[9] "XL-MaxSonar- EZ Series", MaxBotix Inc. Last access: 13.02.2014 [online]. URL: http://maxbotix.com/documents/MB1230-MB1330_Datasheet.pdf

[10] "Ultrasonic Range Finder - XL-Maxsonar EZ3", SparkFun Electronics. Last access: 13.02.2014 [online]. URL: https://www.sparkfun.com/products/9494

[11] "Bluetooth vs. Wi-Fi Power Consumption", Elise Vogler, Demand Media. Last access: 13.02.2014 [online]. URL: http://science.opposingviews.com/bluetooth-vs-wifi-power-consumption-17630.html

[12] "NFC (Near Field Communication)", GSMArena.com. Last access: 13.02.2014 [online]. URL: http://www.gsmarena.com/glossary.php3?term=nfc

[13] "Bluetooth Basics", Bluetooth SIG, Inc. Last access: 13.02.2014 [online]. URL: http://www.bluetooth.com/Pages/Basics.aspx

[14] "Virtuabotix BT2S Bluetooth to Serial Slave for Arduino", Virtuabotix LLC. Last access: 13.02.2014 [online]. URL: https://www.virtuabotix.com/product/bt2s-bluetooth-serial-slave-arduino-versalino-microcontrollers

[15] "Arduino - Introduction", Arduino. Last access: 13.02.2014 [online]. URL: http://www.arduino.cc/en/Guide/Introduction

[16] "Arduino - Compare", Arduino. Last access: 13.02.2014 [online]. URL: http://arduino.cc/en/Products.Compare

[17] "Arduino Pro Mini 328 - 3.3V/8MHz - DEV-11114", SparkFun Electronics. Last access: 13.02.2014 [online]. URL: https://www.sparkfun.com/products/11114

[18] "ATmega48PA/88PA/168PA/328P", Atmel. Last access: 13.02.2014 [online]. URL: http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf

[19] "Low-Power Arduino Using the Watchdog Timer", Fiz-ix. Last access: 13.02.2014 [online]. URL: http://www.fiz-ix.com/2012/11/low-power-arduino-using-the-watchdog-timer

[20] "SparkFun Electronics", SparkFun. Last access: 13.02.2014 [online]. URL: https://www.sparkfun.com

[21] "BT2S Bluetooth to Serial Slave", Amazon. Last access: 13.02.2014 [online]. URL: http://www.amazon.com/gp/product/B006RBK9ZW

[22] "Bluetooth Jamming", Steven Köppel, ETH Zurich 2013. Last access: 13.02.2014 [online]. URL: ftp://ftp.tik.ee.ethz.ch/pub/students/2012-HS/BA-2012-16.pdf

[23] "Ultrasonic Sensors", Futurlec. Last access: 13.02.2014 [online]. URL: http://www.futurlec.com/Ultrasonic_Sensors.shtml