

Object-Tracking with Wireless Sensor Networks

Diploma Thesis

Institute of Information Systems
Distributed Systems Group
Prof. Dr. F. Mattern

Martin Hinz
da@hinz.ch

Assistant: Kay Römer
Supervisor: Prof. Dr. Friedemann Mattern

7th November 2002 - 7th March 2003

Abstract

In this diploma thesis, a sensor network for tracking an instrumented car was developed and tested. Wireless sensor networks consist of many wirelessly connected nodes with limited computation ability that are able to sense the environment and communicate with other nodes or a base station. The randomly distributed nodes estimate their positions autonomously and send sensor data by means of an event-based protocol to the base station, which displays the track of the target, its current speed and direction. In order to fuse the events at the base station, time synchronization is used to establish a common physical time scale among the nodes.

Zusammenfassung¹

In dieser Diplomarbeit wurde ein Sensor Netz entwickelt, das ein prepariertes Auto verfolgen kann. Drahtlose Sensor Netze bestehen aus vielen drahtlos vernetzten Knoten mit limitierter Rechenleistung, welche Änderungen in der Umgebung wahrnehmen können und mit anderen Knoten oder einer Base Station kommunizieren. Die zufällig platzierten Knoten können ihre Position autonom bestimmen und senden die Sensordaten mittels eines ereignisbasierten Protokolls an die Base Station, welche den Weg, die momentane Geschwindigkeit und die Richtung des Autos anzeigt. Um die Ereignisse in der Base Station zu aggregieren, wird mittels Zeitsynchronisation eine einheitliche Zeit unter den Knoten approximiert.

¹A German summary is required by the department of computer science at ETH for a Diploma Thesis written in English.

Contents

1	Introduction	4
2	Foundations	5
2.1	Position Estimation	5
2.2	Time Synchronization	6
2.3	Object Detection	7
2.4	Sensor Networks	8
3	Tracking Experiment	11
4	Sensor Nodes	14
4.1	Introduction	14
4.2	Architecture overview	14
4.3	Node Location Estimation	15
4.3.1	Introduction	15
4.3.2	The Lighthouse Location System	15
4.3.3	Receiver Hardware and Software	17
4.4	Target Detection	19
4.4.1	Introduction	19
4.4.2	Receiver Overview	20
4.4.3	Hardware	20
4.4.4	Receiver Software	22
4.5	Time Synchronization	24
4.5.1	Introduction	24
4.5.2	Synchronization Algorithm	25
5	Communication protocol	27
6	Base Station	29
6.1	Introduction	29
6.2	Architecture Overview	29
6.3	BTNode Lookup	30
6.4	Data Storage	30

<i>CONTENTS</i>	3
6.5 Event Sorting	31
6.6 Data Fusion	32
6.6.1 Velocity and Direction	32
6.6.2 Track Generation	32
6.6.3 Tube of Probability of Stay	35
6.7 Graphical user interface (GUI)	36
6.8 Software Structure	39
7 Related Work	41
8 Conclusion and Outlook	42
A Communication Protocol	46
B Lighthouse Calibration	49

Chapter 1

Introduction

Wireless sensor networks consist of very small computers (nodes) equipped with sensing capabilities and low power wireless communication to exchange information. The applications of wireless sensor networks range from environmental and habitat monitoring, to home automation, medical applications, and the “smart battlefield”. A typical application is the tracking of objects. The nodes can be used to signal the detection of the target. These targets can be for example birds (to observe their behavior), oil slick (to monitor environmental pollution) or cars (to track vehicle movements on the ground).

As part of this thesis, an experimental sensor network for tracking a remote-controlled car was developed and tested. The object tracking experiment takes place indoor on an area not more than 10 square meters. The sensor nodes consist of a battery pack, two small sensor boards and a BTNode. The BTNode is an autonomous wireless communication and computing platform with a Bluetooth radio and an Atmel micro controller, see [14] for more details. To make the detection of the moving car easy for the nodes and the sensor cheap and low power consuming, the car is equipped with an infrared light source and the nodes with infrared receivers. The nodes detect the car at a distance of up to half a meter and send respective detection events to a base station.

Nodes have to estimate their physical location (the nodes are deployed in the environment without prior knowledge of their location). We used the Lighthouse Location System [1] for this purpose.

The reported events are collected and fused on a base station device with better communication and computing capabilities. To fuse the events and form higher level information such as the track of the car, current speed and direction, moreover, the time in a sensor network needs to be synchronized.

The remainder of this document is organized as follows: Chapter 2 gives an overview of the foundations used in this experiment. The documentation of the tracking experiment starts with an experiment overview (Chapter 3), followed by detailed descriptions of the involved devices (Chapter 4-6). After listing some related work in Chapter 7, the document will be concluded in Chapter 8.

Chapter 2

Foundations

2.1 Position Estimation

Many wireless sensor network applications like object tracking require that the locations of the sensor nodes can be determined by means of a so-called Location System. This typically requires to equip each node with a special positioning sensor to let them know their absolute position. However, most traditional systems are very costly in volume, money and power consumption.

In sensor networks not only a low power consumption (for a long battery life) is important. Some system may need fast updates of the node's position in case the nodes are moving, or the accuracy of measurement is more important.

Various approaches can be used for location estimation, which are sketched in the following paragraphs.

Global Positioning System (GPS)

GPS is a very common positioning system for gaining knowledge of the position outdoors. This system uses several satellites orbiting the earth with known positions. The GPS receiver listens to the signals emitted by each satellite and can compute its position very exactly by measuring time-of-flight of these signals. Currently this system is not only costly in power consumption, but also in volume and money. Another disadvantage is that GPS requires line-of-sight signal reception from the satellites and therefore does not work indoors or in the presence of dense vegetation or buildings. The position update, however, is very fast and the accuracy high compared to the large working-area (works everywhere on earth).

Radio Signal Strength

The increasing attenuation of radio signals with increasing distance is an important characteristic of radio propagation. In an ideal spherical propagation model,

the signal strength is correlated to the distance with the formula:

$$\text{signalStrength} \sim \frac{1}{\text{distance}^n} \quad \text{with } n \geq 2$$

Therefore, the measured signal strength of a received signal can be used to estimate the distance from the sender. If the distances to three or more senders with known locations are known, the position of the receiver can be obtained by multilateration.

The location estimation for mobile phone uses this technique. On the basis of the antenna positions and signal strengths, the mobile phone's location can be estimated. Some antennas even have several small antenna segments that divide space into sectors, allowing a more accurate estimation. The power consumption to receive the signal is rather low. Reflections on buildings can increase the inaccuracy. [20]

Radio Connectivity

A simplified version of the approach described in the previous section is the following. If a receiver can “hear” a sender, then the receiver's distance from the sender is smaller than the communication range. Various approaches can be used to derive a location estimate based on this observation. The algorithm in [9] locates a receiver to the centroid of the locations of the senders it can “hear”.

Convex position estimation [5] is a more elaborate technique to derive an area which bounds the possible locations of a receiver.

This idealized radio model approximation is not appropriate for indoor environments because of reflections and occlusion and because the computed position is not very accurate.[9]

Sound

A signal sent by radio travels much faster than an acoustic signal. This difference of velocity of propagation can be used to estimate the distance between two devices. The sender sends a signal at the same time by radio and by ultrasound. The receiver measures the difference of the arrival time and can estimate with the propagation speed the distance. Using the time difference of arrival as an estimate for time of flight of the ultrasound signal, the receiver can estimate its distance to the sender. [21, 22]

2.2 Time Synchronization

The base station collects all the detect events arriving from the nodes. To fuse these events, the real time of this data is important to derive the ordering of these events and to form higher level information or knowledge about the environment.

Network Time Protocol (NTP)

NTP (keeping the Internet clocks in phase) is a hierarchical system with one or more reference clocks at the top. The reference clocks are assumed to be accurate. A “child” device synchronizes its clock to the “father” device according to the hierarchy. The accuracy of a clock depends on how “close” a clock is to a reference clock (in the hierarchy), the network latency and the claimed accuracy of the clock.

Reference Broadcast Synchronization (RBS)

In [10] an algorithm is described that allows a set of receivers of a broadcast message to synchronize their clocks. Nodes periodically send broadcast messages to their neighbors. The receivers use the arrival time of the broadcast message as a point of reference for comparing their clocks. These messages can be sent any time and do not explicitly contain a timestamp. This design is called Reference-Broadcast Synchronization RBS.

Hop-by-Hop Synchronization

Another approach is described in [11]. Instead of synchronizing clocks, timestamps contained in messages are transformed to the clock of the receiving node. See Chapter 4.5 for details.

2.3 Object Detection

There are different ways to detect a moving object or objects. If the object is not under the control of the tracking party, it must be detected without special instrumentation of the object. In our task the moving object is under our control thus, we can instrument the object with special equipment for detection.

An important goal is power-efficient object detection, since power supply is limited. This can be achieved by low sampling rates and power saving signal processing algorithms. However, this can result in reduced accuracy. The power consumption can also be reduced by preprocessing the raw signal data at the nodes in order to reduce costly communication. The communication via Bluetooth (the wireless communication technology we used) is very power consuming. Keeping the number of messages low increases the battery life time.

The next step is not only to detect the target but also to classify it on the basis of some target characteristics.

Motion Detectors

Motion detectors detect movements of objects that have a higher temperature than the environment. Since motion detectors work in a passive way (the detector

does not emit any signals) the moving object will not be aware of the detector. These detectors are limited in their detection angle and range. The sensing of “cold” moving objects will not work, since these detectors work by detecting “body heat”. Moreover, they are sensible to the environmental temperature and solar radiation.

Magnetometers

Magnetic materials change the earth’s magnetic field slightly, and this change can be used for detection. Therefore, a magnetometer can be used to detect cars or other metal objects with a certain minimum mass. The output of the magnetometer has to be preprocessed in order to trim out the DC components of the earth’s magnetic field. A disadvantage is the high power consumption of such detectors [13].

Infrared Light

If the tracked object can be instrumented with IR light emitters, it can be easily detected with IR photo diodes. However, this approach is very sensitive to daylight and artificial light. Additionally, typical IR emitters are not omnidirectional. See Section 4.4.3 for details.

Acoustic Signals

Every moving object produces acoustic signals (e.g engine sound of moving vehicles). With a microphone this noise can be detected. Such a device is easy to manufacture and inexpensive. The produced electrical signal can be directly used and requires only little amplification.

Seismic Vibrations

Vibrations, produced by a moving object on the ground, can be sensed within some meters. A piezo-electric sensor converts the seismic signal into an electric signal. Seismic detectors are also used in high security environment to detect vibrations of the ground. They run with an extremely low current draw.

2.4 Sensor Networks

Wireless sensor networks consist of many sensor nodes. A sensor node is a very small computer that can communicate with other nodes and has sensor capabilities to sense certain parameters of the physical environment (temperature, acoustic signals, light, ...). An individual node is very limited in computing capacity and the sensed data of only one node does not give useful information

about the environment. However, by fusing data obtained from nodes throughout the network, valuable information can be obtained. To fuse the data, the point in time and location of a sensed phenomenon is important. The first requirement is achieved by synchronizing the clock of all devices, the second requirement is achieved by a location sensor attached to every node. Since the nodes are spread in the field and difficult of access, it is important that the power consumption is kept low to increase the lifetime of the nodes. This can be achieved by low power consuming position estimation and sensing techniques, by low power consuming communication techniques, and by reducing data communication, for example by preprocessing raw sensor data in the nodes.

In the following paragraphs we describe some experimental sensor nodes.

Smart Dust [18]

Smart Dust is a project at UC Berkley. A smart dust node is a autonomous sensing and communication node in a cubic millimeter. It has multiple sensors such as temperature, light, vibration, etc. Because of the small architecture, the development time and costs are very high. Nodes communicate with a base station by using a modulated laser beam (Figure 2.1).

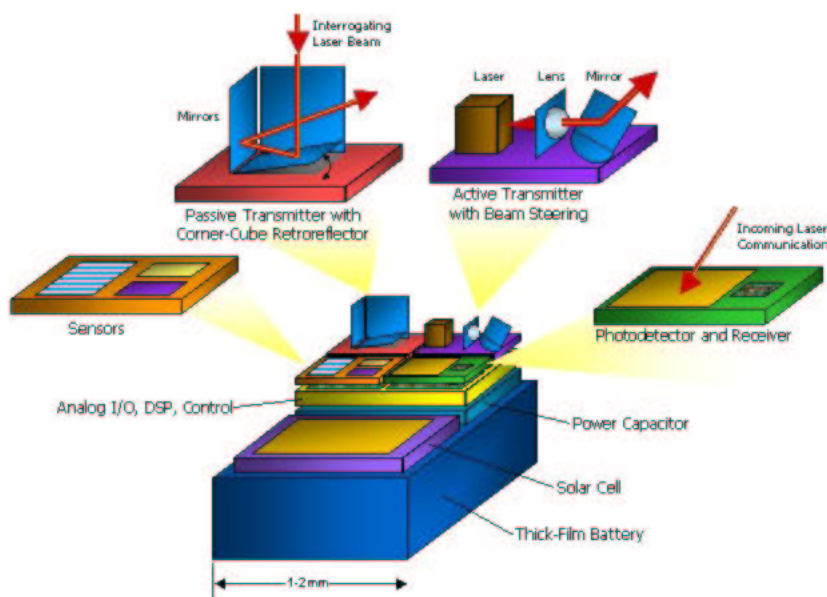


Figure 2.1: Smart Dust, property of [23].

Commercial-off-the-Shelf Dust (COTS Dust) [12]

Because the time to develop a complete Smart Dust system takes too long, an alternative node is needed to test basic behavior that can be produced in a shorter time. Such nodes are built using commercial-off-the-shelf components with all the basic functionalities, but the nodes are bigger in size (a cubic inch). An Atmel micro controller with sensors and a communication unit are the basic elements of COTS Dust. For communication a low power, low bandwidth RF transceiver is used. Different sensors can be connected to this node, for example temperature, light and pressure sensors (Figure 2.2).

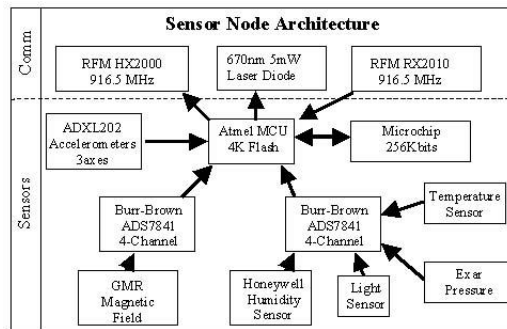


Figure 2.2: Architecture of COTS Dust, property of [12].

Bluetooth Node (BTNode) [14]

A BTNode is an autonomous wireless computing and communication platform with a Bluetooth radio and a micro controller (Atmel ATmega 128L). The node provides several ports to connect sensors or control elements and provides in-system programming. The battery lifetime is not very long, since Bluetooth communication consumes much power (Figure 2.3).

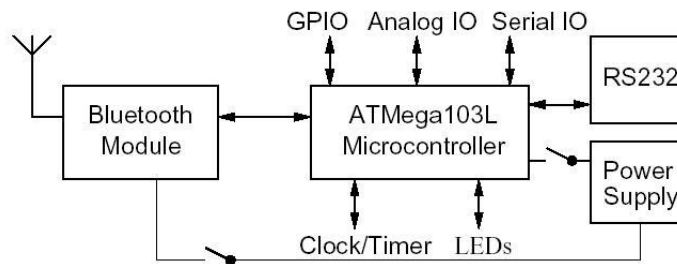


Figure 2.3: BTNode with Bluetooth radio and Atmel micro controller, property of [24].

Chapter 3

Tracking Experiment

The goal of the experiment conducted in this thesis, is to track a moving object and estimate its speed and direction with a sensor network. The sensor network consists of small sensor nodes randomly distributed on an area of about 10 square meters. Each node estimates its position autonomously. A base station device collects and fuses the data received from the nodes and displays the track of the target, its current speed and direction.

We used a laptop with Bluetooth as base station and seven sensor nodes (see Figure 3.1 for the experiment setting) with a microprocessor and Bluetooth chip, called BTNodes, see Section 2.4. Attached to each node are a battery pack, a sensor for estimating the node's location and a sensor to detect the moving object, shown in Figure 3.2. BTNodes and the base station communicate by means of an event-based protocol.

The tracked target is a specially instrumented remote-controlled car as shown in Figure 3.3. The BTNodes can estimate their position with the aid of a laser device that defines the origin of a 2-dimensional coordinate system.

First we describe in detail the sensor nodes in Chapter 4. In Chapter 5 the communication protocol is described, and Chapter 6 contains the details of the base station.

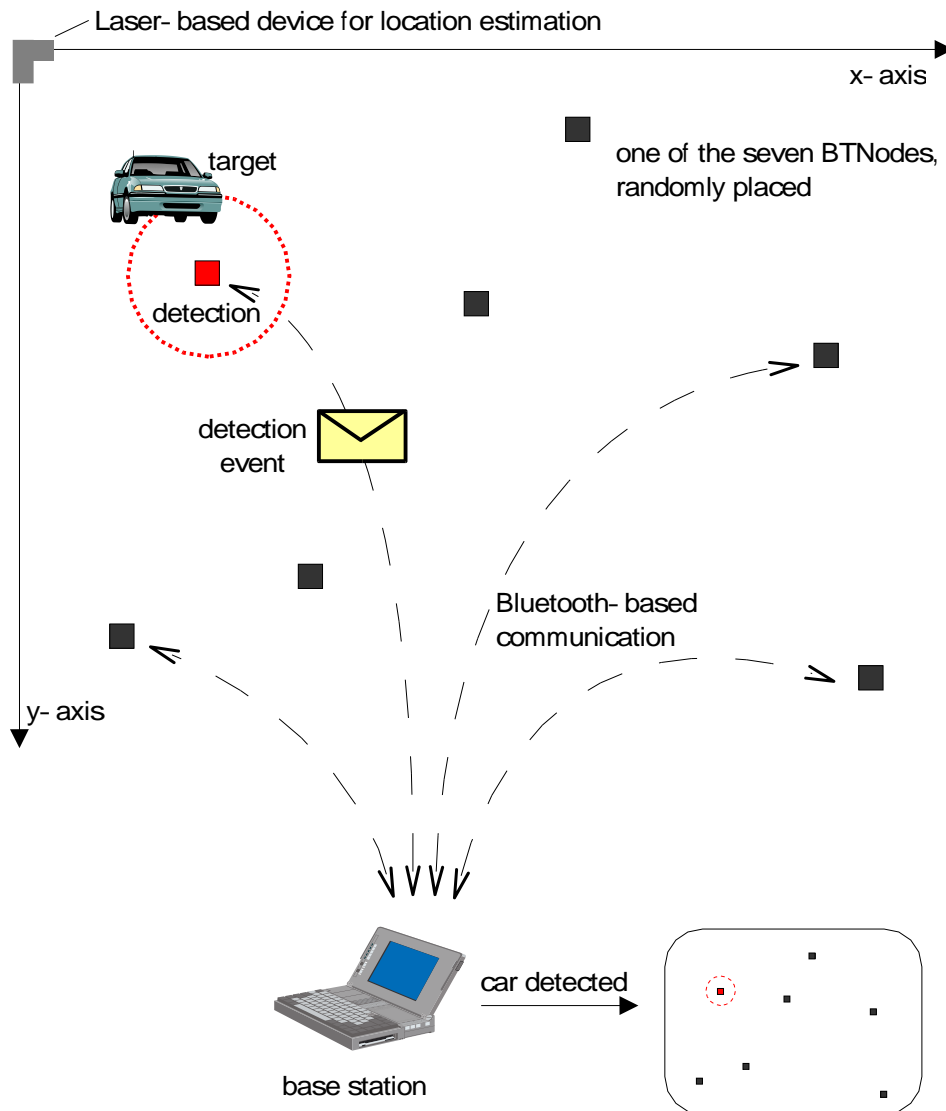


Figure 3.1: Experiment setting.

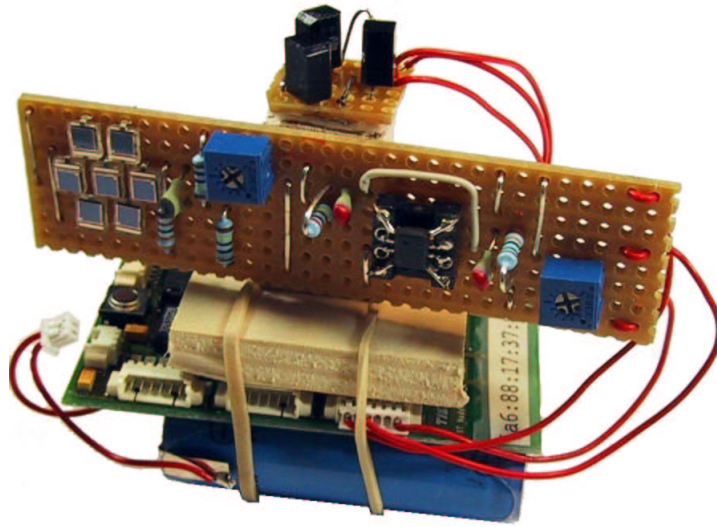


Figure 3.2: BTNode with sensors for location estimation and car detection.

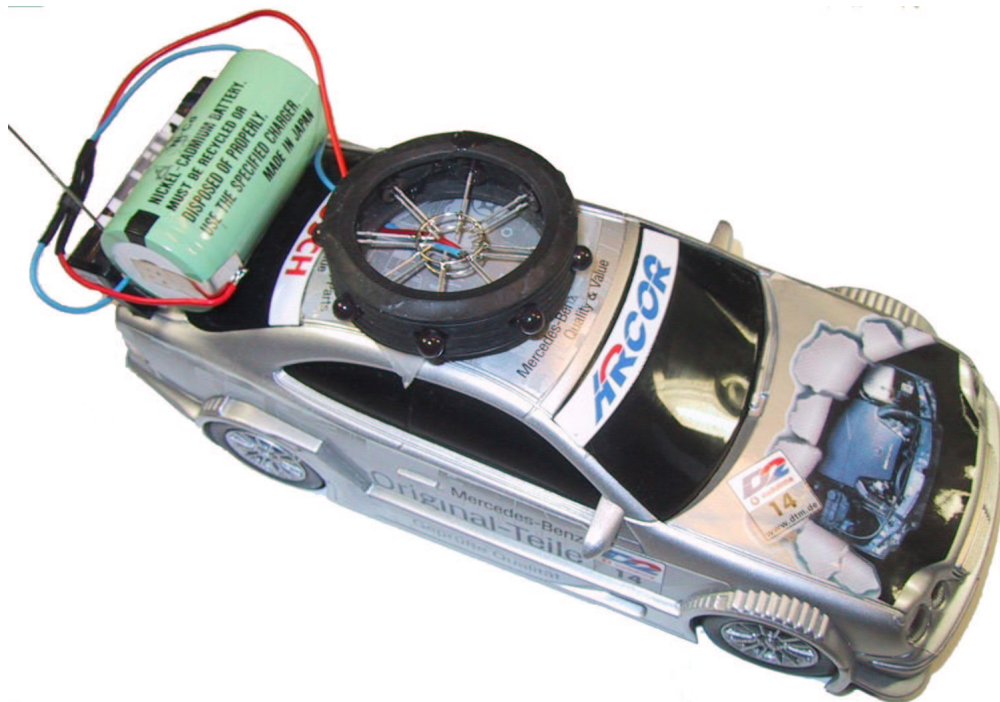


Figure 3.3: Remote-controlled car instrumented for detection by the BTNodes.

Chapter 4

Sensor Nodes

4.1 Introduction

The task of the randomly placed sensor nodes is to detect the car and report any detection to the base station. The raw data will be preprocessed on each node to keep the costly communication to the base station to a minimum. To track the car, the position of each detection and therefore the position of each node is required. In order to fuse events originating from different nodes into a speed estimate, a common physical time base has to be established among the nodes.

This chapter starts with a description of the location estimation procedure. The following sections describe the target detection and time synchronization.

4.2 Architecture overview

Figure 4.1 depicts the architecture of the sensor nodes. Each node contains a component for location estimation, which requires a high resolution clock. The event detection component is used to detect the moving car. The BTNode generates for every new position or new detection a message in the event processing, that will be exchanged, according to the transmission protocol. The data in the control messages arriving from the base station will be stored in the data storage, providing the components with control data.

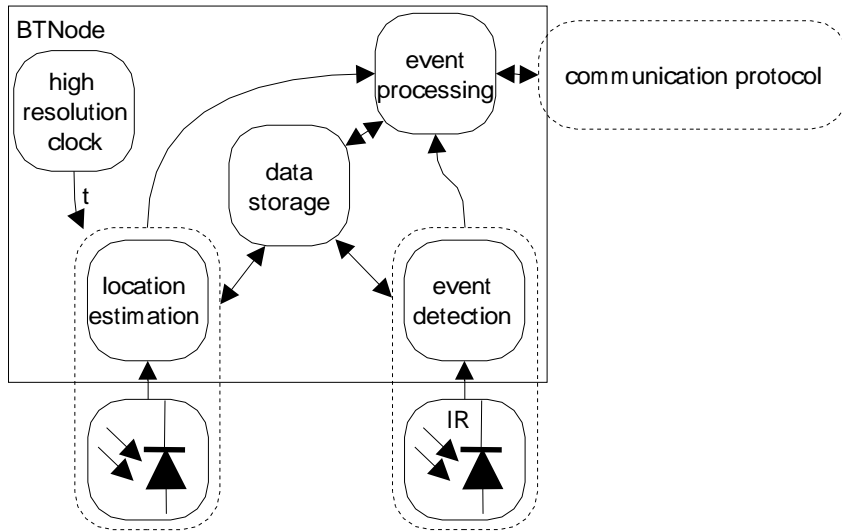


Figure 4.1: Overview of the BTNode.

4.3 Node Location Estimation

4.3.1 Introduction

To determine the position of the car, each BTNode needs to know its position. With the Lighthouse Location System, described in [1], each BTNode can estimate the position autonomously with a relatively small overhead.

4.3.2 The Lighthouse Location System

Consider an observer looking at a lighthouse as shown in the left hand side of Figure 4.2. The time t_{beam} , during which the observer sees the lighthouse “flash”, depends on the angle of beam spread α . Since α is independent of the observer’s distance from the lighthouse rotation axis, t_{beam} is independent of the distance as well.

Now consider a lighthouse with a parallel beam as shown in the right hand side of Figure 4.2. There the angle α (under which the observer sees the beam) depends on the distance from the lighthouse rotation axis. Hence, t_{beam} also depends on this distance. Using basic trigonometry, the observers distance from the lighthouse rotation axis can be easily inferred from t_{beam} . Using two such lighthouses, a simple 2D location system can be built as shown in Figure 4.3. The two lighthouses are assembled such that their rotation axes are mutually perpendicular. The distance d_1 and d_2 then equal the y and x coordinates of the observer in the 2-dimensional coordinate system defined by the lighthouse

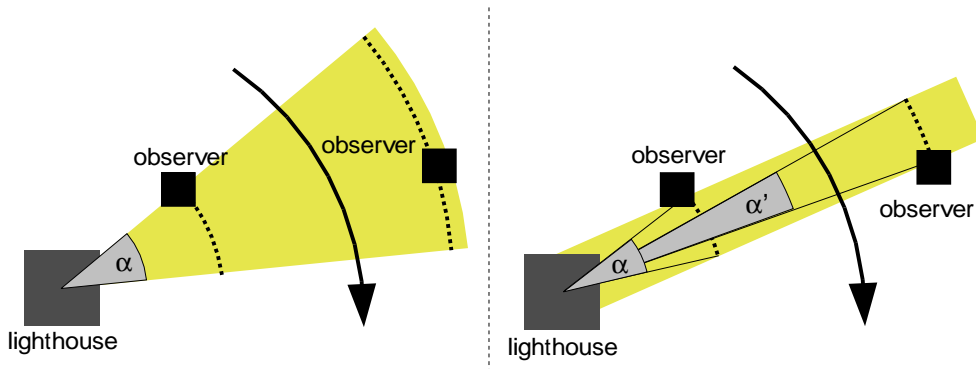


Figure 4.2: On the left a normal lighthouse, on the right a lighthouse with parallel beam.

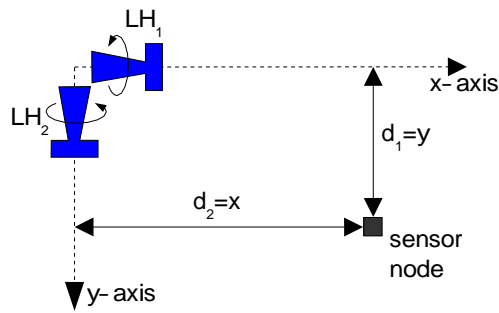


Figure 4.3: With two lighthouses, a simple 2D location system can be build.

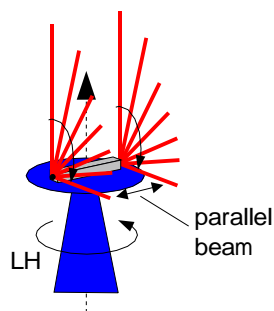


Figure 4.4: Two rotating laser beams simulate the outline of the parallel beam.

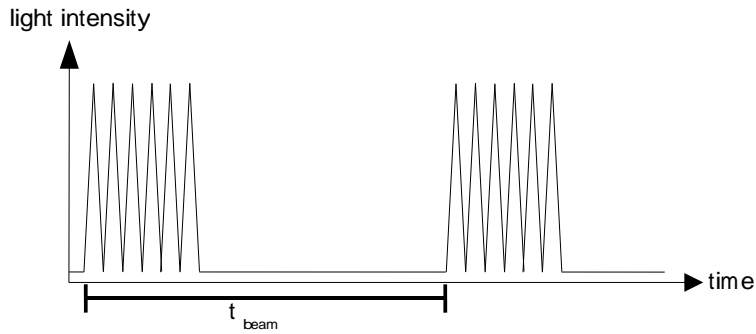


Figure 4.5: An observer sees the outline of the parallel beam as laser pulse sequences.

rotation axes.

As described in [1], a lighthouse with a parallel beam can be implemented by two “rotating laser beams” as depicted in Figure 4.4. The observer can identify a lighthouse by the rotation speed of its beams. An observer looking at such a lighthouse sees a sequence of very short flashes as the lighthouse platform carrying the rotating laser beams rotates by. t_{beam} can be obtained by measuring the amount of time between two such sequences of pulses as depicted in Figure 4.5.

4.3.3 Receiver Hardware and Software

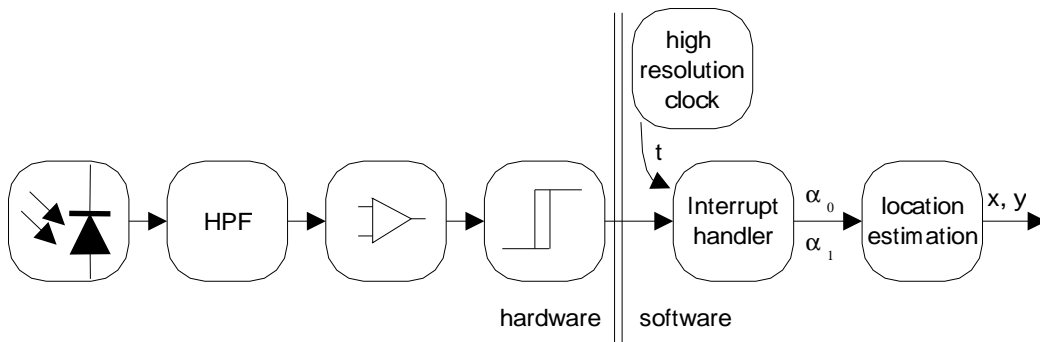


Figure 4.6: Schema of the receiver.

In order to measure t_{beam} as explained in Section 4.3.2, we use the following hardware, depicted in Figure 4.6.

A photo diode converts the incident light into a voltage proportional to the intensity of the light. The size of the photo diode matters for great distances to the lighthouse, because at least two laser flashes are needed to recognize the lighthouse [1]. The farther away, the fewer laser flashes are in a pulse sequence, see

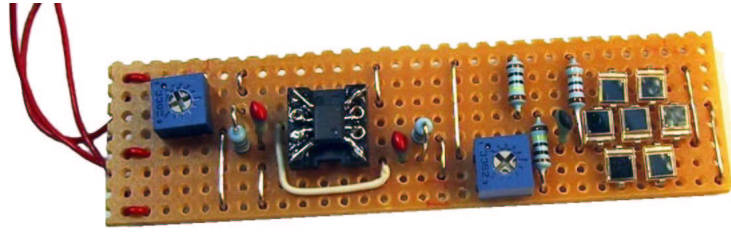


Figure 4.7: Receiver board with 7 photo diodes.

Figure 4.5. To increase the maximum distance to the lighthouse, several photo diodes are used on the receiver circuit board (see Figure 4.7). The high pass filter (HPF) removes lower frequency components resulting from artificial light (50Hz) and daylight. After amplifying the signal, a Schmitt Trigger transforms the voltage level into a digital signal. The output of the Schmitt Trigger is connected to an external interrupt port of the BTNode, such that each light flash triggers an interrupt.

We will now discuss the receiver software in more detail.

Interrupt Generation

The receiver will generate an interrupt every time the laser hits the photo diodes. An interrupt handler processes these interrupts by generating an interrupt event that holds the current time. The interrupt handler has to be shortrunning because many interrupts are generated in a burst (up to 130 interrupts in less than half a second).

In order to distinguish the two lighthouses, the rotation frequency f of the laser beams has to be measured as explained in Section 4.3.2. Since the rotation period $T = \frac{1}{f}$ of the two lighthouses differs by only 1ms, the clock resolution of 1ms of the BTNode System Software is not sufficient, since the laser rotation times can vary (Figure 4.8).

Therefore, we implemented a clock with higher resolution. For this purpose the ATmega128L micro controller provides several timer/counter registers. The value of such a timer register is increased by one upon each tick of the CPU clock in consideration of a prescale factor. An interrupt will be generated, whenever the counter register overflows[6].

With a CPU speed of about 8MHz and a prescaling of 256 we get:

$$\text{counterClockSpeed} = \frac{8MHz}{256} = 31250Hz$$

$$\text{counterResolution} = \frac{1}{\text{counterClock}} = \frac{1}{31250Hz} = 0.032ms$$

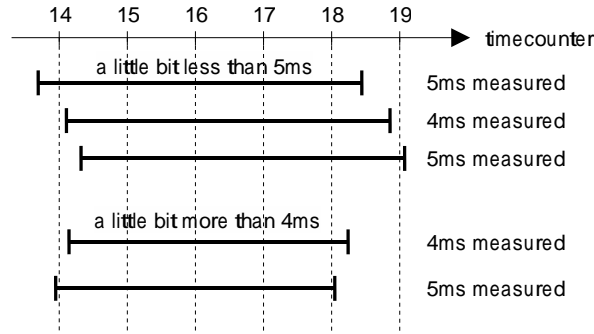


Figure 4.8: A time resolution of 1ms is not enough to distinguish between flashes of e.g. 4ms and 5ms, that vary slightly.

$$\text{overflowIntervall} = 2^{16} * \text{resolution} = 2^{16} * 0.032\text{ms} \approx 2.1\text{s}$$

An internal variable is increased by 2^{16} every time the overflow interrupt occurs. The current “clock value” can be derived by adding the current timer register value to this internal variable.

Handling the Interrupt Event

To compute the distance to the lighthouse, the angle α is of interest (Figure 4.2). The program measures the time that a lighthouse takes for one turn (t_{turn}) and the time that it “sees” the parallel beam (t_{beam} , see Figure 4.5). With these values, α can be calculated.

Location Estimation

Using the approach described in [1], the node position (x, y) is inferred from values α_1 and α_2 for the two lighthouses. To obtain more exact measurements, the lighthouse needs to be calibrated first, see Appendix B.

4.4 Target Detection

4.4.1 Introduction

The BTNode needs a sensor to detect the moving car. For this purpose, we equipped the car with IR light emitters. The sensor nodes use IR photo diodes to detect the car at a distance of up to 30cm. By measuring the incident light intensity, the sensor nodes can estimate the distance of the car.

4.4.2 Receiver Overview

Figure 4.9 depicts the receiver hardware and software. The infrared photo diode converts the current intensity of the incident infrared light into a proportional voltage that is transformed into a 10-bit number by a analog to digital converter (ADC) every 250ms. If the current value is greater than a threshold, then the difference to this threshold is sent to the event generator. If the value drops under this threshold, the value 0 is sent to the event generator. According to the reading rate value, the event generator emits a detect event containing the value difference. This reading rate value defines, how many events should be emitted. For example, a reading rate of 4 means every fourth time a sample is taken, a detect event will be emitted while the car is being detected.

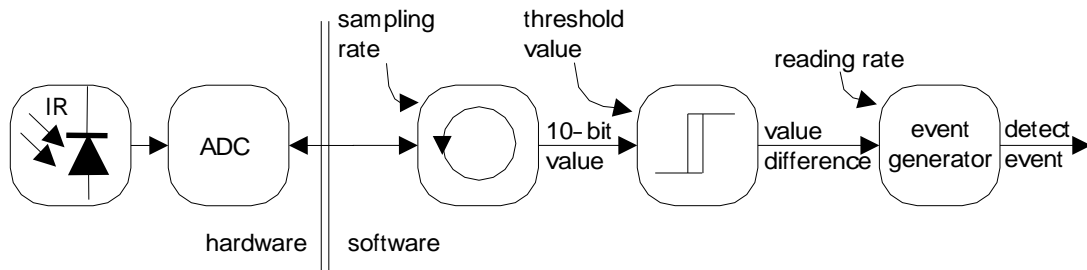


Figure 4.9: Overview of the receiver component.

4.4.3 Hardware

For reliable object detection, both the IR light emitters and the IR sensors should have omnidirectional characteristics. However, both IR LEDs and IR photo diodes typically have directional characteristics. Therefore we used multiple elements to emulate omnidirectional characteristics as depicted in figures 4.10 and Figure 4.11.

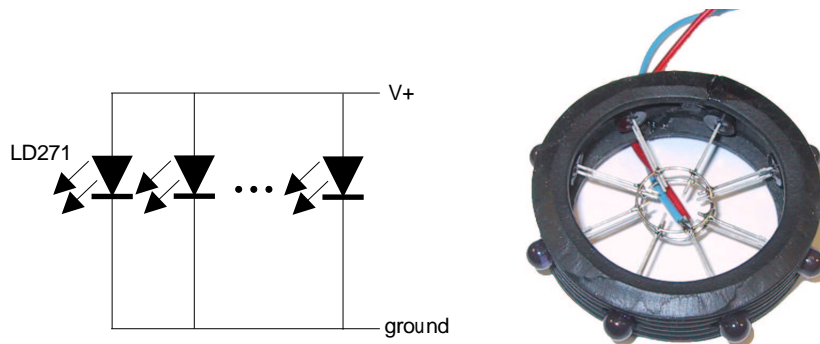


Figure 4.10: Circuit drawing and picture of the IR emitter mounted on the car.

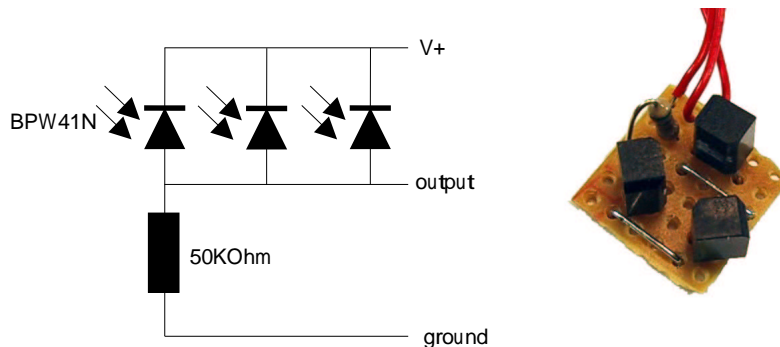


Figure 4.11: Circuit drawing and picture of the IR sensor attached to the BTNode.

The infrared emitter LD271 has a wavelength peak at 950nm and an emission angle of only 50° . To create a more or less omnidirectional light source, we mounted eight infrared diodes on top of the car, under an angle of 45° each (see Figure 4.10). A battery pack provides the diodes with power.

The IR receiver BPW41N is equipped with an IR Filter with a wavelength peak at 940nm and has a wide radiant sensitivity of 120° . Three receivers are mounted under 120° to cover the entire circle. A resistance divides the voltage between GROUND and V+ (Figure 4.11). The higher the resistances, the more sensitive the output voltage reacts to changes of the incident light intensity.

Although the used approach for object detection has clear disadvantages, we chose it for its simplicity and low cost. Moreover, a sampling rate of 4Hz is sufficient, which matches the limited processing power of the Atmel micro controller.

Unfortunately the light intensity produced by the eight infrared diodes depends on the angle (see Figure 4.12 on the left) and the same problem occurs with the three infrared receivers (see Figure 4.12 on the right). Experiment on a turning knob (the infrared source as well as the infrared receiver) showed that the intensity of the infrared signal varies enormously at the same distance.

Another disadvantage is the dependence on daylight. It can happen that all BTNodes generate a detection event if e.g. the sun suddenly breaks through the clouds and lights up the room. However, this could be fixed by modulating the emitted light and changing the receiver to detect this modulated signal.

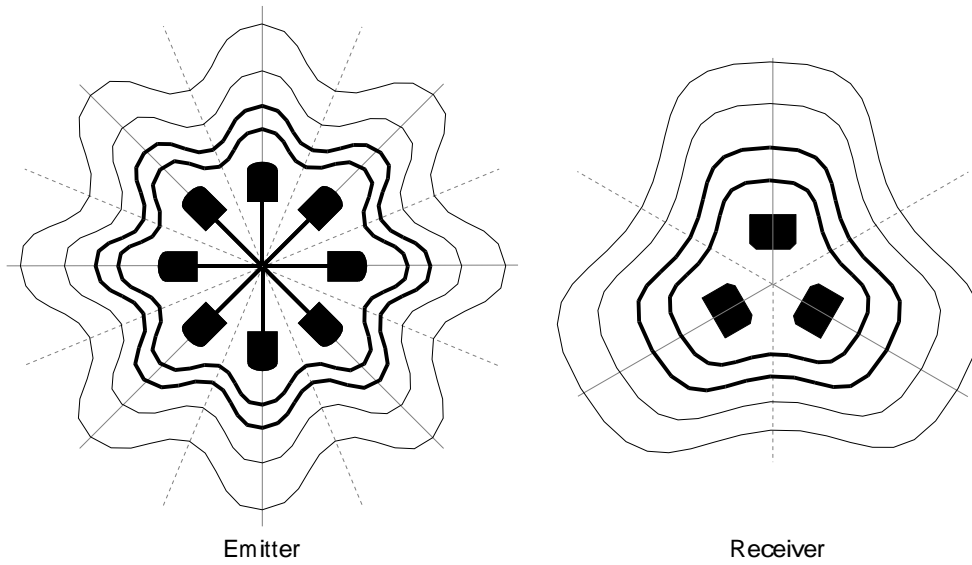


Figure 4.12: Light intensity of the emitter and sensitivity of the receiver depend on the angle.

4.4.4 Receiver Software

Variable Detection Threshold

The ambient light level (see Figure 4.13) is the value obtained from the ADC if the car is not present. This value is usually around 20. In order to adapt to changing daylight, this value can be dynamically changed.

The car is detected if the ADC output value raises above a certain threshold (ambient light level + gap), see Figure 4.13. The gap is needed to filter out small changes in ambient light intensity.

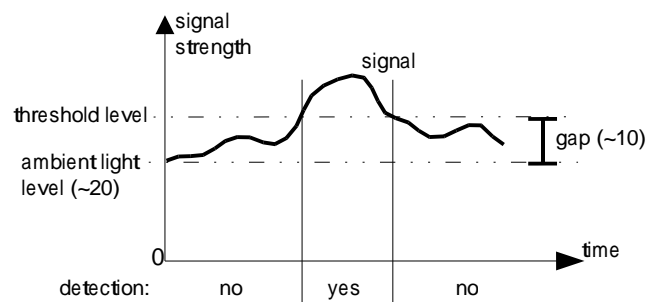


Figure 4.13: Detection depending on the signal value being higher or lower than the threshold level.

Reading the ADC

A timeout is used to trigger reading the ADC. Unfortunately the resolution of the timeout is only 250ms. Every time the timeout occurs, the current ADC value is read, compared to the threshold (ambient light level plus the gap value) and a new timeout will be set.

If the the car is detected (Figure 4.13), a detect event containing the difference between signal value and threshold will be sent to the base station. When the car is no longer detected, a detect event with value 0 will be sent.

Detection Event Frequency

In order to reduce communication overhead and energy consumption, the event generation frequency can be set to multiples of 250ms (i.e., 250ms, 500ms, 750ms, ...) by specifying a reading rate of 1, 2, 3, ... Assume for example, the car is within detection range of a node for one second. With a reading rate of 3, the node sends two detection events containing the difference between signal intensity and threshold, and one detection event containing the value 0 for the car leaving detection range. A reading rate value of 0 only generates an event if the signal crosses the threshold (target entering range of detection, target leaving range of detection).

Distance Estimation

To obtain the approximate distance between the car and the node, we assume that we have an ideal spherical propagation model of IR light emitted by the car, that correlates the signal strength to the distance as follows:

$$signalStrength = \frac{c}{distance^2} + s_0 \quad || \quad c = const, s_0 = const \quad (1)$$

The distance can not be less than $d_{min} = 5cm$ due to the volume of the BTNode. The signal strength s_{max} (depending on the battery charge) at this distance can be measured. The signal strength is close to 0 (we are looking at the signal value above threshold) at a distance of about $d_{max} = 30cm$. Using the two constraints, we obtain the following equation system:

$$0 = \frac{c}{d_{max}^2} + s_0 \quad || \quad s_{max} = \frac{c}{d_{min}^2} + s_0$$

Solving for c and s_0 , we obtain the following values:

$$c = \frac{s_{max} * d_{min}^2 * d_{max}^2}{d_{max}^2 - d_{min}^2} \quad || \quad s_0 = -\frac{s_{max} * d_{min}^2}{d_{max}^2 - d_{min}^2}$$

By solving formula (1) for distance and inserting c and s_0 , we obtain the following formula for distance in terms of IR signal strength:

$$distance = \frac{\sqrt{s_{max}} * d_{max} * d_{min}}{\sqrt{signalStrength(d_{max}^2 - d_{min}^2) + s_{max} * d_{min}^2}}$$

Corresponding graph shown in Figure 4.14 ($d_{min} = 5cm, d_{max} = 30cm, s_{max} = 500$).

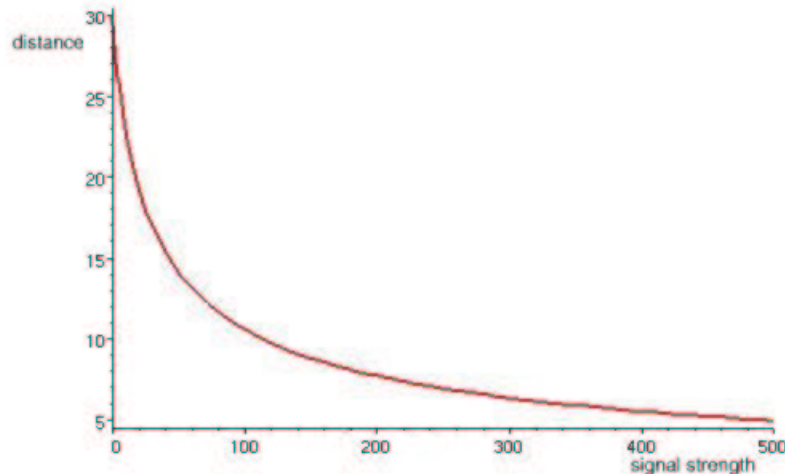


Figure 4.14: Correlation between signal strength and distance.

4.5 Time Synchronization

4.5.1 Introduction

The base station collects detection events from all sensor nodes in order to estimate speed, direction and the path of the tracked car. For this, the base station needs to know the points in time when the car was detected by the individual nodes. Therefore, detect events contain a time stamp. However, all time stamps have to share a common physical time scale. This is usually achieved by synchronizing the clocks of the sensor nodes. But, keeping the clocks in sync is very costly in terms of communication and energy overhead.

Therefore we use a different approach. We do not synchronize the clocks. Instead, timestamps are transformed from the senders clock to the receivers clock when sent from the sensor node to the base station as part of detection events.

4.5.2 Synchronization Algorithm

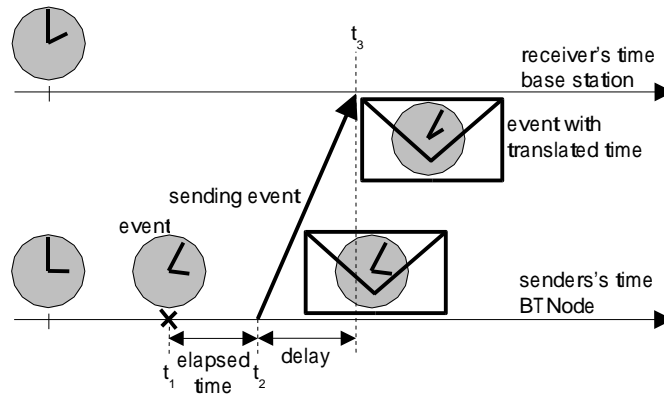


Figure 4.15: The timestamp in the message is transformed to the receiver's time.

Consider Figure 4.15. The sensor node detects an event at time t_1 and sends off the event to the base station at t_2 ($t_2 - t_1 = \text{elapsedTime}$). The base station receives the event at t_3 (with respect to its own clock). If we knew the delay for sending the message, then we could transform the timestamp t_1 to base station time by calculating $t_3 - \text{delay} - (t_2 - t_1)$. The delay can be estimated by measuring the round-trip-time in the sever and assuming $\text{delay} \sim \frac{\text{rtt}}{2}$, i.e:

$$\text{newTimestamp} = \text{arrivingTimestamp} - \frac{\text{rtt}}{2} - \text{elapsedTime}$$

The rtt can be computed by subtracting the idle time of the sender from the idle time of the receiver as depicted in Figure 4.16. Idle time means the elapsed time between two Bluetooth activities (receiving/sending or sending/receiving). To obtain the rtt, only messages generated by the nodes (position messages and detection messages) are taken into account, although only the detect events need to be synchronized. The position message (usually sent before any detection message) initializes the receivers idle time with the position acknowledgment. This approach assumes that the clock drift is negligible and that the delay can be estimated by $\frac{\text{rtt}}{2}$. The first assumption is valid, since both rtt and elapsed time are rather small. The latter assumption is true, since Bluetooth provides symmetrical communication channels.

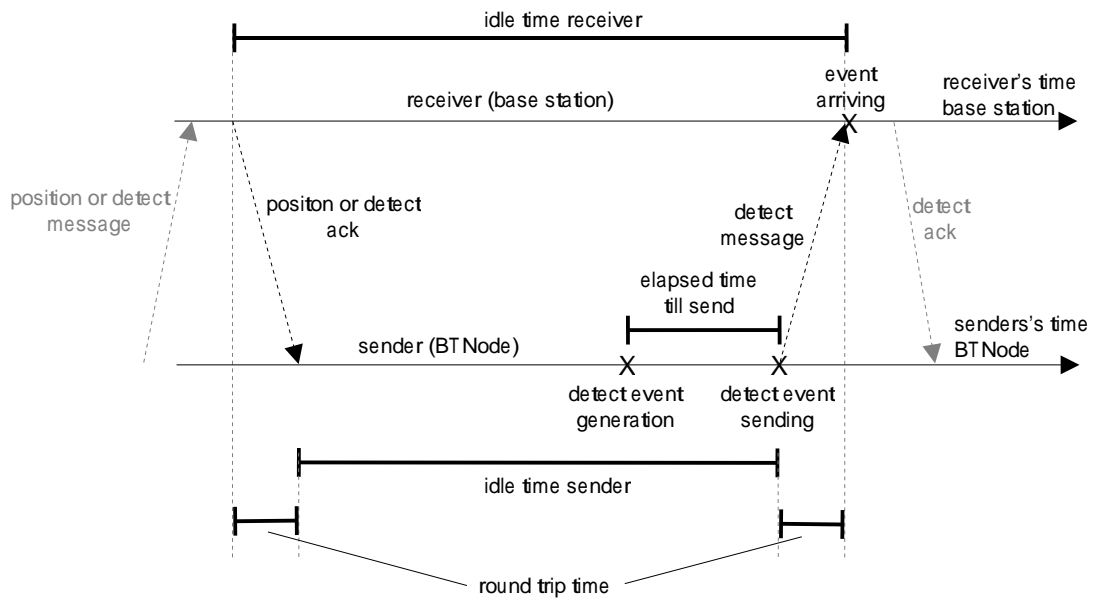


Figure 4.16: Time stamp transformation.

Chapter 5

Communication protocol

BTNodes and the base station communicate by means of an event-based protocol. The possible message types are:

message types	node \longleftrightarrow base station
POSITION	\longrightarrow
POSITION_ACK	\longleftarrow
DETECT	\longrightarrow
DETECT_ACK	\longleftarrow
THRESHOLD_SET	\longleftarrow
THRESHOLD_ACK	\longrightarrow
THRESHOLD_NACK	\longrightarrow
READING_RATE	\longleftarrow

The second row shows the direction of the message. Only an overview of each message is given in this chapter, for more details of the individual fields in each message, please refer to Appendix A.

POSITION (BTNode \rightarrow Base Station) Each time a new position has been computed, this message will be sent to the base station. It contains the x and y position and the angle α of each lighthouse. The angels are used for calibration (see Appendix B for more details).

POSITION_ACK (Base Station \rightarrow BTNode) The base station sends back an acknowledgment message after receiving the position of the node. An acknowledgment is only needed to initialize the time synchronization, see Chapter 4.5 for details.

DETECT (BTNode → Base Station) Each time a new detection event occurs, the node sends this message to the base station. It contains the difference between the current signal value (i.e., light intensity) and the threshold value. It also contains a time stamp representing the point of time of detection of the car.

DETECT_ACK (Base Station → BTNode) After receiving a detect message, the base station sends back an acknowledgment message to the node.

THRESHOLD_SET (Base Station → BTNode) This message contains the gap value to be set, see Figure 4.13. This message can be used to change the gap level and therefore the sensitivity to changes in ambient light intensity. And to change the ambient light level indirectly to adjust to a higher ambient light. Upon arrival of this message, the BTNode sets the ambient light level to the current signal value.

THRESHOLD_SET_ACK (BTNode → Base Station) After resetting the ambient light level and gap value the sum of these two values (i.e., the threshold) is sent back to the base station to give the base station device information about abnormally high ambient light intensity.

THRESHOLD_SET_NACK (BTNode → Base Station) The ADC transforms the IR receiver output voltage into a 10-bit value (0-1024). If the ambient light is too intense, the ADC can return a value of 1024 and car detection is not possible anymore. If the threshold value would be greater than 1000, this message is returned to the base station in return to a THRESHOLD_SET message.

READING_RATE (Base Station → BTNode) This message contains the reading rate value that defines the frequency of detect messages. Since there will not be any problems with setting this value, no acknowledgment is sent back.

Chapter 6

Base Station

6.1 Introduction

The base station collects detect events from the sensor nodes in order to compute and display velocity, direction, and the track of the tracked car. Moreover, the base station controls certain parameters of the sensor nodes like detection sensitivity and frequency. The base station consists of a laptop equipped with Bluetooth.

This chapter starts with an architecture overview of the base station. Afterward, every component introduced in the architecture is described in more detail. At the end of this chapter, the software implementation structure is sketched.

6.2 Architecture Overview

Depicted in Figure 6.1, the graphical user interface (GUI) component allows the user to establish connections to all BTNodes, which is handled by the “BTNode lookup” component. Information about BTNode connection and disconnection will be forwarded to the “data storage” component that holds all the information of a connected node. Furthermore, the GUI can be used to send control events to connected nodes, such as setting a new detection threshold or a new reading rate.

A new position of a node or a new threshold will be directly forwarded to the “data storage”, whereas the detection events need to be sorted first. Due to different transmission delays in the network, events need not arrive in order of their time stamps. The “sorting detect events” component reorders events according to their time stamps and forwards them to the “data storage”. The data in the “data storage” are displayed by the GUI. To give the user an idea about the track of the target, the “track generation” component stores the detect events with their corresponding positions and plots a track of the car.

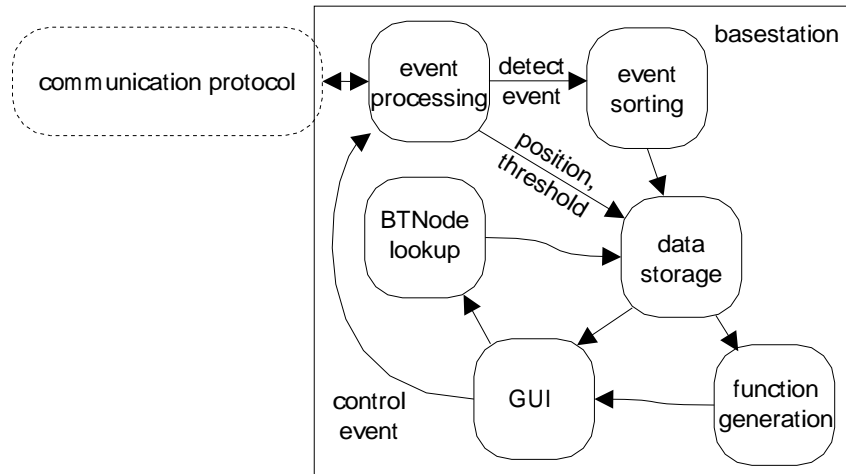


Figure 6.1: Overview of the base station architecture.

6.3 BTNode Lookup

At startup, the base station has to lookup active BTNodes which are participating in the sensor network. For this purpose, Bluetooth provides an Inquiry procedure. In order to distinguish sensor nodes for other Bluetooth devices, sensor nodes are named “BTNode DETECT 2.0.x” (x is standing for the current version loaded on the BTNode). First, the base station starts an Inquiry to find all Bluetooth devices in range, and then checks the remote name of every found device. Problems occurred if too many bluetooth devices are in the room, in case of a low node battery (no l2cap connection error), or a WLAN access point communicating on the same frequency.

6.4 Data Storage

For every connected sensor node the program stores it’s current position (x and y), number of detections (for statistical purposes), the IR receiver output (signal value) and timestamp of the last detection event, the threshold, and a timestamp with corresponding signal strength that will be used for computing the speed as explained in the next paragraph. To plot a track of the target, the position of the detection events will be stored.

Significant Timestamp and Signal

To compute the speed of the car between two nodes, we need to know the time that the car took from one node to the other. Imagine that a device sends three successive detect events with timestamps t_1 , t_2 , and t_3 and corresponding

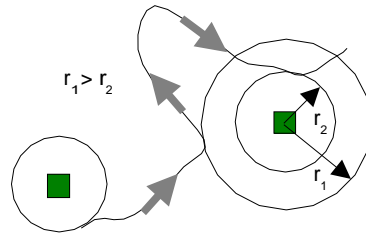


Figure 6.2: Target detected by same sensor node twice, closer on the second time, without being detected by other nodes.

signal values s_1 , s_2 , and s_3 . Which timestamp should we use for computing the detection time? In any case $t_1 \leq t_2 \leq t_3$ holds (due to event sorting), but there is no similar relationship among the signal strength values. Note that the closer the car approaches a node, the more accurately we can estimate the actual position of the car. Therefore, the point of closest approach (PCA) timestamp (timestamp of the event with $\max(s_i)$) is used to calculate the velocity of the object moving through the sensor field. This approach can lead to wrong results, if, for example, the car is detected by a sensor node, and finally returns closer to the same node without being detected by other nodes. In this case, the calculated speed would be too small (Figure 6.2).

6.5 Event Sorting

Due to different network message delays, the detect events do not necessarily arrive in order of their time stamps at the base station. Hence, a queue is used to reorder arriving events according to their time stamps before passing them to the “data storage” component.

The queue stores events sorted by time stamps in ascending order. Upon arrival, events are assigned a departure time (current time plus WAITTIME ms) before being inserted into the queue. Every CHECKTIME ms, the program goes through the queue and checks if an event’s departure time is due. If it finds such an event n , all events up to event n are removed from the front of the queue and forwarded to the data storage component. If the maximum network delay is smaller than WAITTIME, then this ensures correct ordering of events. We used WAITTIME=200ms. (Figure 6.3).

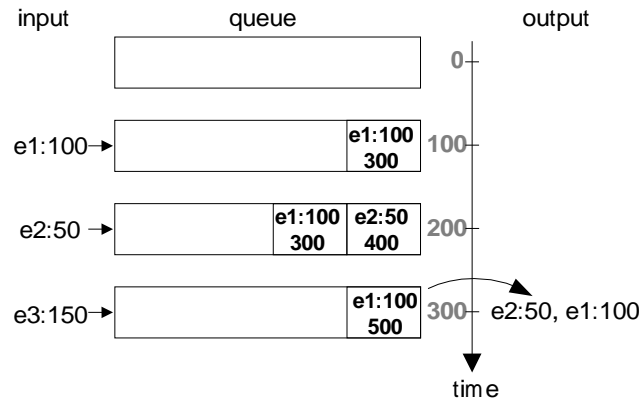


Figure 6.3: Sorting of the events with WAITTIME=200ms.

6.6 Data Fusion

6.6.1 Velocity and Direction

The current velocity and direction of the target are useful information for tracking. In a sensor network, only an approximation can be computed (lower bound of average speed). We simplify the computation of the velocity and direction by assuming, that the detected car has the same position as the detecting node. Since we only know the distance to the node, this assumption is only an approximation. Further it must be pointed out that the computed values represent the average speed and direction between the last two detecting nodes.

The direction is computed as the position vector from the second last detecting sensor node to the last detecting sensor node. As described in Section 6.4, the time stamps of the closest car approach are used to derive the time to compute the speed.

6.6.2 Track Generation

Knowing the position of each detection, a possible track of the target can be computed. It is unlikely that the target moves to one sensor node, makes a sudden turn, heading for the next sensor node in a straight line (Figure 6.4 on the left). To show the track more realistically, a Bezier curve over each line (edge) will be computed (Figure 6.4 on the right). To derive the Bezier curve, two additional bases (see Figure 6.5) are needed for every edge. They are computed by using the positions of the previous and next track points, which will be explained later in more detail.

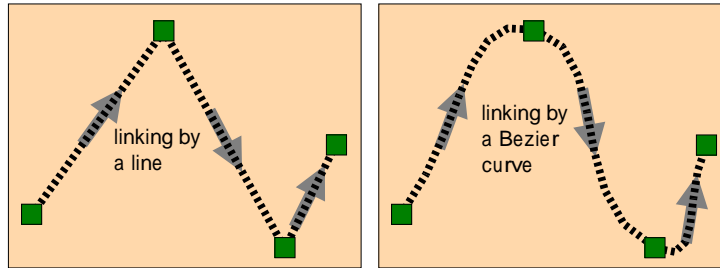


Figure 6.4: On the left: the trackpoints are linked by a straight line, on the right: a Bezier curve connects the trackpoints.

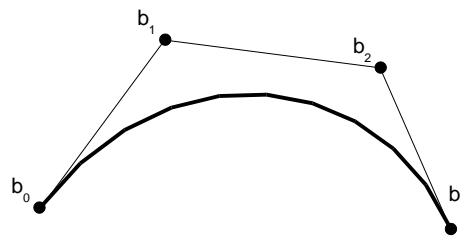


Figure 6.5: Cubic Bezier curve.

Bezier Curve

To construct a cubic curve, we need four control points (see Figure 6.5). The Bezier curve has the property to interpolate the endpoints b_0 and b_3 . The two points b_1 and b_2 , called bases, influence the derivations at the endpoints and the shape of the curve. The resulting curve can be described by the Bernstein polynomial¹:

$$x(t) = b_0(1-t)^3 + 3b_1t(1-t)^2 + 3b_2t^2(1-t) + b_3t^3$$

t runs from 0 to 1. $x(t)$ is the according point on the curve.

Because the curve needs to interpolate all trackpoints, the program constructs a piecewise Bezier curve over each edge with a C^1 -continuity. This continuity means that the first derivation of both Bezier curves adjacent to a sensor node, have the same value, in other words the last two points b_2^1 and b_3^1 of *curve*¹ and the first two points b_0^2 and b_1^2 of *curve*² form a line h^2 (note that $b_3^1 = b_0^2$), see Figure 6.6.

¹The deCasteljau Algorithm can be used to draw this curve as well.

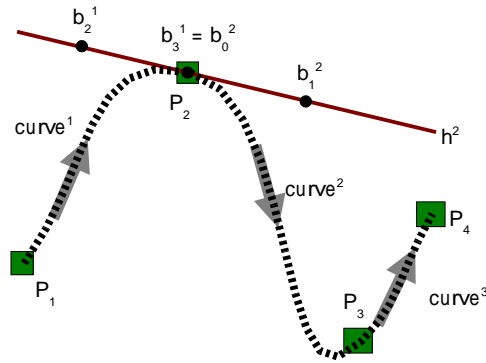


Figure 6.6: C^1 -continuity of the piecewise Bezier Curve.

But what is the gradient of line h and what are the positions of the bases b_2^1 and b_1^2 on this line? Take a look at Figure 6.7. The gradient Δ of line h^2 is the same as the gradient Δ of the line through the points P_1 to P_3 :

$$\Delta (h_n) = \Delta (P_{n-1}P_{n+1})$$

This formula can not be used at the begin and at the end of the curve, since there are not enough points. Therefore, at the begin P_0 is replaced by P_1 and at the end P_{n+1} by P_n .

Now the line is defined, but the positions of the bases b_2^1 and b_1^2 on h are still unknown. The program uses the distances $|P_1P_2|$ and $|P_2P_3|$ and maps the ratio to the distances $|b_2^1b_3^1|$ and $|b_0^2b_1^2|$ (see Figure 6.8):

$$\frac{|b_2^{n-1}b_3^{n-1}|}{|b_0^n b_1^n|} = \frac{|P_{n-1}P_n|}{|P_nP_{n+1}|}$$

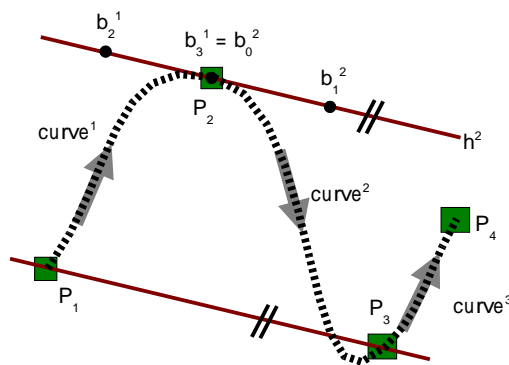


Figure 6.7: Line h^2 has the same gradient as the line through P_1 and P_3 .

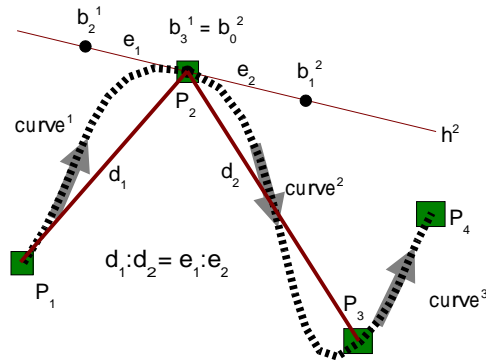


Figure 6.8: The ratio of the three trackpoints is mapped to the distance between the bases.

Again this formula cannot be used at the end points. Knowing the ratio, we still need to know the position of one of the bases. We use half of the distance between the trackpoints P_{n-1} and P_{n+1} for this:

$$|b_2^{n-1}b_1^n| := \frac{|P_{n-1}P_n| + |P_nP_{n+1}|}{k}$$

with $k=2$. The variable k changes the shape of the Bezier curve. The smaller k , the better the curve follows the line h^i . The bigger k , the more the curve resembles the straight lines between the trackpoints, depicted in Figure 6.9.

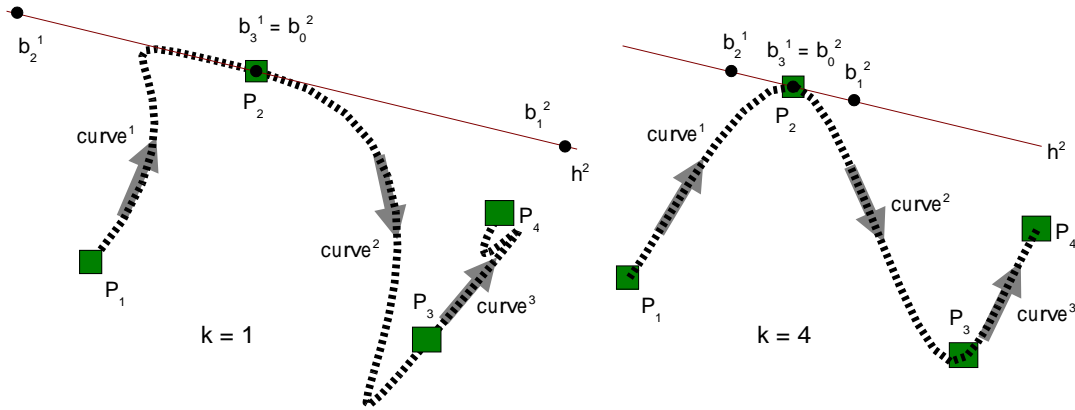


Figure 6.9: Small k on the left side, large k on the right side.

6.6.3 Tube of Probability of Stay

The track constructed in the previous section is only a vague approximation of the actual path the car moved along. However, the sensor nodes not only detected

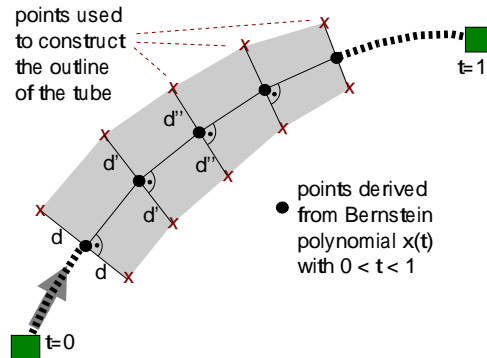


Figure 6.10: Tube construction.

the car, but know an estimate of the car's distance from the node. Using this information, we can construct a “tube”, such that the actual track of the car is contained in the tube with high probability. The width of the tube changes linearly from track point to track point according to the car's distance to the respective nodes. One could argue that the width should be the greater, the farther away from a node, because the farther away, the less the system knows about the moving object's actual position.

We will now describe, how such a tube can be constructed. Using the Bernstein polynomial, the track (i.e., the Bezier curve) is approximated by small line segments as depicted in Figure 6.10. The tube is constructed of similar small line segments parallel to the original line segments with distance d . Note, that d is linearly interpolated between the car's distance from the two endpoints of the track over each edge.

6.7 Graphical user interface (GUI)

The device control elements on the GUI are: lookup button, disconnect button², gap slide with set button to define the gap value and rate slide with set button to define the reading rate. A table shows various information about each connected node. One more slide and a clear button give the user the opportunity to set the length of the track displayed on the field on the right side and to clear all track data. (Figure 6.11)

On this tracking field every connected node is displayed as a green square. A node detecting the target turns red and a circle shows the approximate distance of the target to this node. To give the user more information about the target movements, a tube shows a likely track of the target.

²It is important to disconnect all devices, since quitting the program does not close the Bluetooth connections and all nodes are still part of a piconet that does not exist anymore.

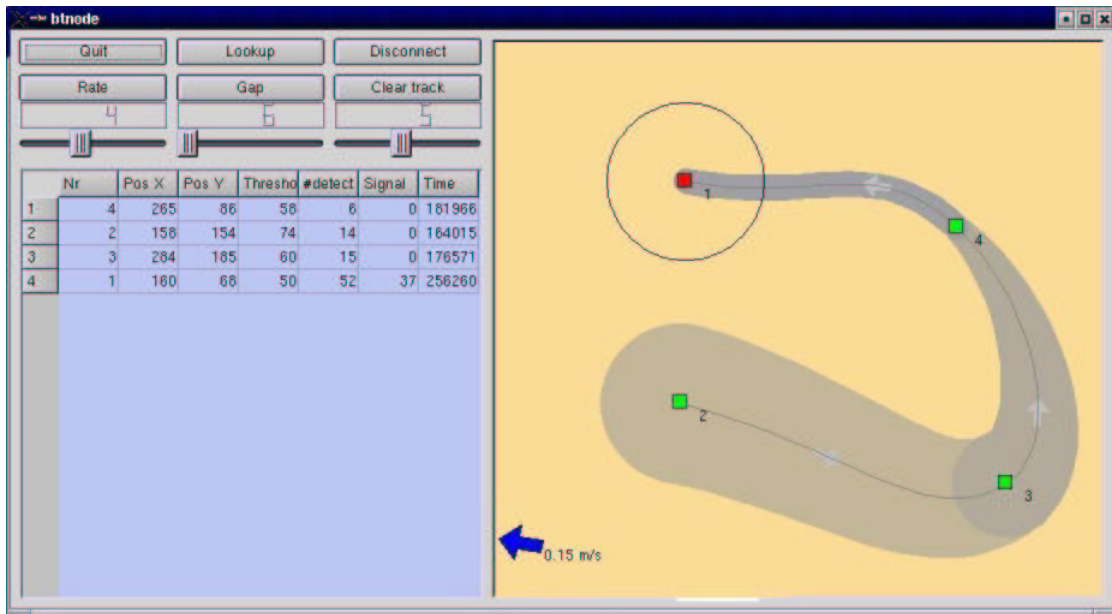


Figure 6.11: GUI

An arrow in the left bottom corner indicates the direction of the object (rather the average direction of the object between the last two sensor nodes) and the speed (average speed between the last two sensor nodes). For speed computing refer to Section Section 6.4.

To allow the user to differ between overlapping tube segments, the color fades the more to the background color, the older the tube segment is.

Some artifacts occur at the boundary points of the Bezier segments that do not disappear after painting the tube polygon, see Figure 6.12. A circle at each node hides these artifacts, see Figure 6.13.

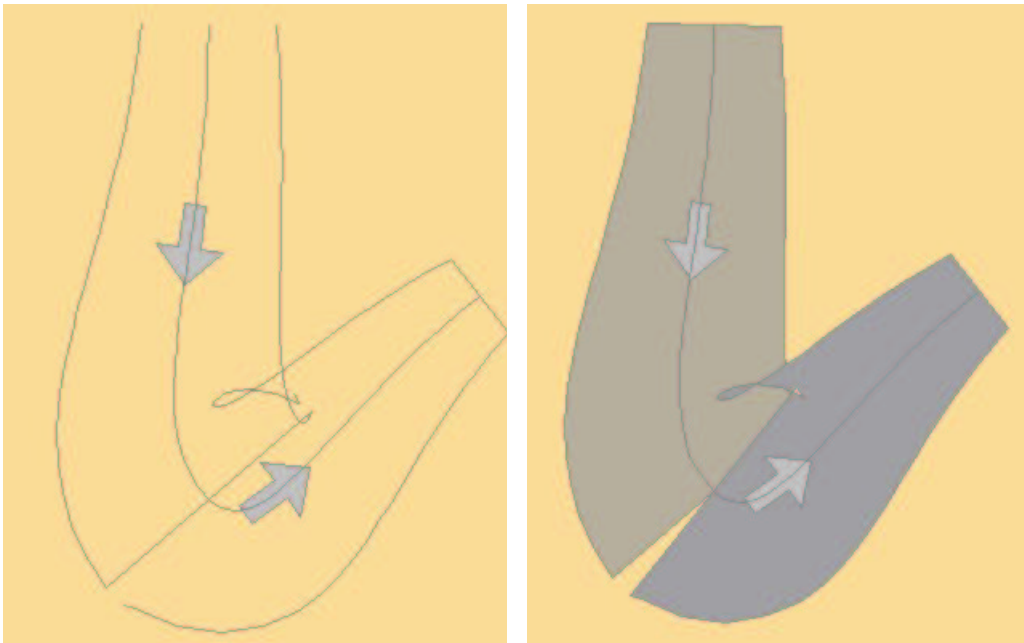


Figure 6.12: Artifact of the tube at the boundary points in the inner curve.

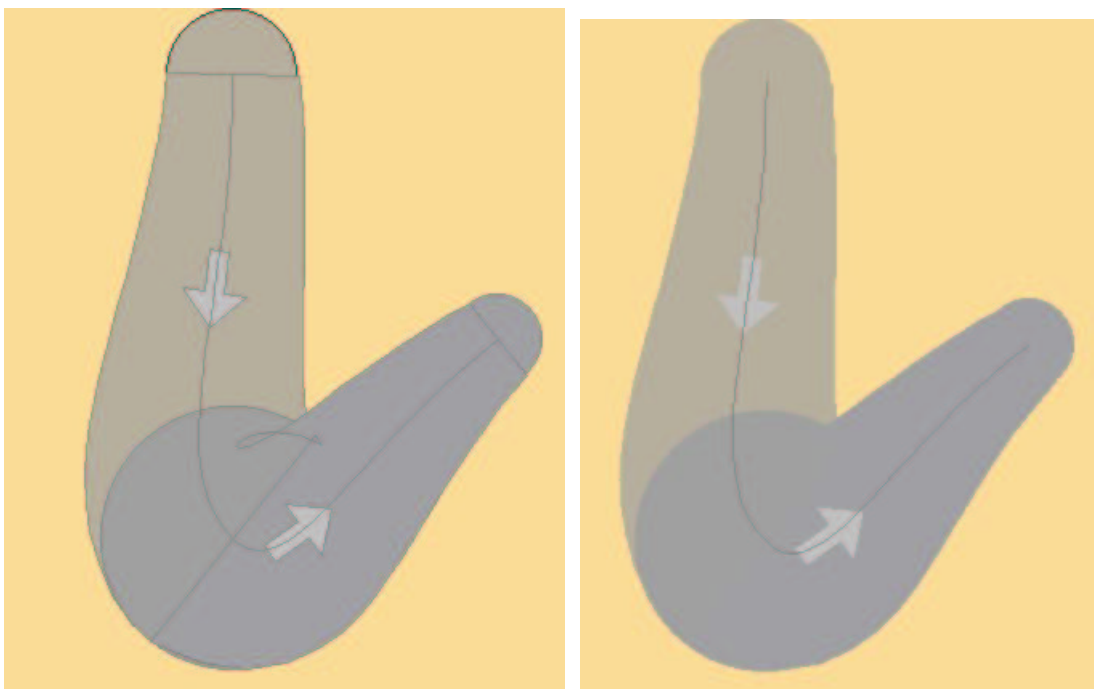


Figure 6.13: A circle hides possible artifacts.

6.8 Software Structure

Figure 6.14 shows an overview of the base station software structure. In order to decouple interaction with the BTNodes from interaction with the user, the base station software uses two threads. The Bluetooth thread cares for node lookup, connection management and data communication with the BTNodes. The main thread cares for everything else, including the user interface. Both threads interact via a well defined interface.

This interface consists of two groups of functions. The first group - output event handling - provides functions for controlling certain aspects of the BTNodes. The second group - input event handling - provides functions for dealing with events generated by the BTNodes.

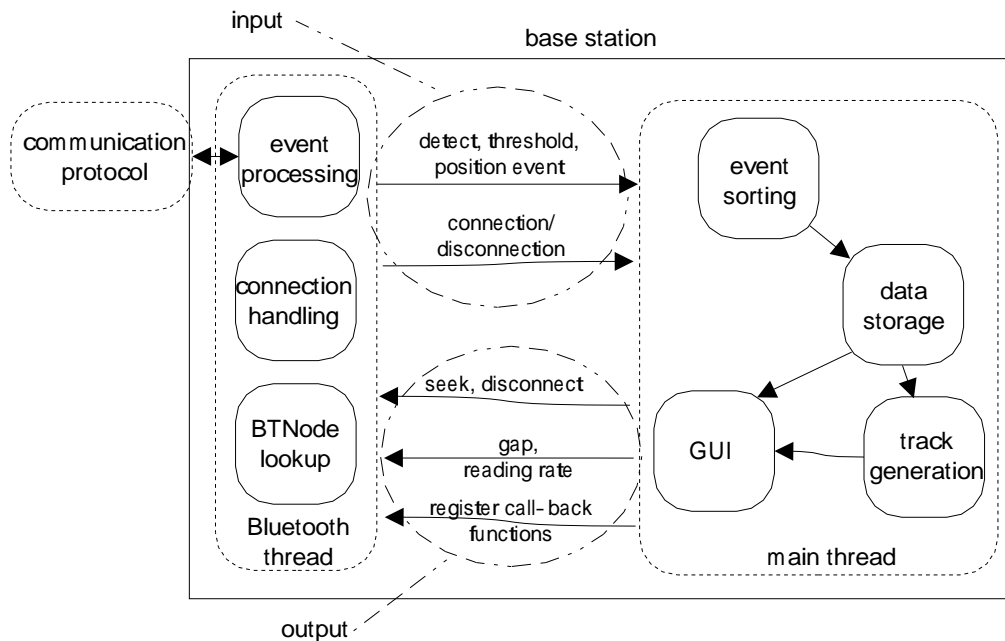


Figure 6.14: Decoupled interaction of the base station software.

Output Event Handling

lookup_node_event(): Initiates a Bluetooth inquiry in order to look up BTNodes participating in the sensor network and opens a connection to each discovered node.

threshold_node_event(int gap): Send a THRESHOLD_SET message (containing the new gap value) to all connected nodes.

rate_node_event(int rate): Send a READING_RATE message (containing the new rate value) to all connected nodes.

disconnect_node_event: Close all open connections to BTNodes

Input Event Handling

device_event(int number): This callback function forwards BTNode connection and disconnection events to the main thread.

register_device_method(void (*device_event_method) (int)): Used to register the callback function `device_event` with the Bluetooth thread.

data_event(int type, int index, int data1, int data2): This callback function forwards messages arriving from the BTNodes to the main thread. Possible message types are (see Chapter 5):

1. DETECT: `data1` holds the IR-light intensity, `data2` holds the timestamp.
2. POSITION: `data1` holds the x position, `data2` holds the y position of the node.
3. THRESHOLD_ACK: `data1` holds the threshold (see Figure 4.13 for details), `data2` holds 0.
4. THRESHOLD_NACK: `data1` and `data2` both hold 0.

register_data_method(void (*data_event_method) (int, int, int, int)): Used to register the callback function `data_event` with the Bluetooth thread.

Chapter 7

Related Work

In [13], a tracking experiment similar to ours is described. Real vehicles have been tracked with an UAV-delivered sensor network. An unmanned aerial vehicle (UAV) deploys a sensor network along a road. The goal is to detect and track vehicles passing through this network. The nodes on the ground establish a time-synchronized multi-hop communication network. The vehicle track information is transferred to the observer at the base camp via the UAV.

The advantage is, that the base station can be placed far from the tracking area while in our experiment the base station has to be near enough to keep connection to all nodes.

The magnetometers used to detect the vehicles, are able to detect passenger vehicles at more than 5 meters, bigger vehicles like busses and trucks at up to 10 meters. To place the nodes on the road and collect information from the nodes during the experiment, the UAV is equipped with GPS. The nodes are placed at predefined positions by throwing them off the UAV at the right moment. This is in contrast to our experiment, where nodes are placed randomly and find out there positions using a location system.

A tiered sensor network has been used for bird monitoring in [15] and [17]. The network consists of two types of nodes, Berkley motes [12] and PDA's. The PDA's act as cluster heads and collect and process data from the nodes in its cluster in order to locate and classify the bird. Based on spectrogram pattern matching, the target can be classified. The motes are densely deployed to increase the probability of nodes being close to the bird to be detected. Bird localization in [17] is based on beam forming using Time Difference Of Arrival (TDOA). To estimate the target location with TDOA, the nodes need fine-grained time synchronization [10]. The PDA's are equipped with GPS and provide the motes with location information.

[16] considers target tracking from a theoretical point of view and introduces information-driven dynamic sensor collaboration to reduce resource consumption, to increase detection quality, and to increase robustness against failures.

Chapter 8

Conclusion and Outlook

With this experiment we developed a sensor network that is able to track an instrumented car (infrared source attached to the car). The nodes estimate their positions by means of the Laser Localization System and detect the car with infrared photo diodes. The Bluetooth communication between the nodes and the base station uses an event-based protocol. With the time synchronization the base station is able to fuse the detection events. The base station provides the estimated track of the target, the velocity and the direction between the last two sensor nodes.

Our experiment runs only with a limited number of nodes due to the limited number of Bluetooth devices that can participate in a Piconet. To obtain more accurate tracks, more nodes are needed.

There are several ways to support a larger number of nodes. Nodes could be grouped into several Piconets with the base station switching between them. Instead of keeping permanent connections to the node, the base station could connect/disconnect before/after each message exchange. As a third option, Scatternets could be used. This has several potential implications such as increased network delay (nodes need to connect to the base station before sending data), the need for sending messages along multiple hops, and more inaccurate time synchronization along multiple hops.

The used time synchronization fits well for the experiment and allows the base station to reorder messages which arrive in the wrong order. This time synchronization would also work along multiple hops.

Due to interference with other Bluetooth devices, wireless access points and low BTNode batteries, an l2cap connection could not always be established. A less power consuming and more robust radio technology should be considered.

To obtain more accurate estimates of the distance of the tracked object from the detecting node, other detection strategies should be considered.

Once the calibration data have been measured and computed, the position estimation with the laser works very well (average error 6% of the node's distance from the base station). A high resolution clock makes the differentiation between

the two lighthouses possible.

If the nodes are densely deployed, the tracked car can be detected by multiple nodes at the same time. In this case a more accurate location estimation could be obtained by using multilateration.

Bibliography

- [1] Kay Römer: The Lighthouse Location System of Smart Dust, In MobiSys 2003, San Francisco, May 2003
- [2] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E.: "Wireless Sensor Networks: A Survey," Computer Networks (Elsevier) Journal, pp.393-422, Atlanta, USA, March 2002
- [3] Joe C. Chen, Kung Yao, and Ralph E. Hudson: Source localization and beamforming, In IEEE Signal Processing Magazine 19(2), March 2002
- [4] Li, Wong, Hu, Sayeed: Detection, Classification and Tracking of Targets in Distributed Sensor Networks, In IEEE Signal Processing Magazine, March 2002
- [5] Lance Doherty, Kristofer S. J. Pister, and Laurent El Ghaoui: Convex position estimation in wireless sensor networks. In Proceedings of IEEE Infocom 2001, Anchorage, Alaska, USA, April 2001
- [6] ATMEL ATmega128L Microcontroller Hardware Documentation, Atmel Corporation 2002
- [7] Silonex, The art of micro-engineering, www.silonex.com
- [8] Farnell Electronic Components United, www.farnell.com
- [9] Nirupama Bulusu and John Heidemann and Deborah Estrin: GPS-less Low Cost Outdoor Localization for Very Small Devices, In IEEE Personal Communications Magazine 7(5), pp.28-34, October 2000
- [10] J. Elson, L. Girod, and D. Estrin: Fine-Grained Network Time Synchronization using Reference Broadcasts, In OSDI 2002, Boston, USA, December 2002
- [11] Kay Römer: Time Synchronization in Ad Hoc Networks, In MobiHoc 2001, Long Beach, CA, USA, October 2001

- [12] Seth Edward-Austin Hollar: COTS Dust, University of California, Berkeley, USA, 2000,
http://www-bsac.eecs.berkeley.edu/~shollar/macro_motes/macromotes.html
- [13] UC Berkley and MLB Co: 29 Palms Fixed/Mobile Experiment, Tracking vehicles with a UAV-delivered sensor network, CA, USA,
<http://robotics.eecs.berkeley.edu/~pister/29Palms0103/>
- [14] Computer Engineering and Networks Laboratory (tik): BTnode rev2.2, Swiss Federal Institute of Technologie, Zurich, Switzerland
<http://www.inf.ethz.ch/vs/res/proj/smart-its/btnode.html>
- [15] Hanbiao Wang, Deborah Estrin and Lewis Girod: Preprocessing in a Tiered Sensor Network for Habitat Monitoring, Submitted for Publication, September 2002
- [16] Feng Zhao and Jaewon Shin and James Reich: Information-Driven Dynamic Sensor Collaboration of Tracking Applications, In IEEE Signal Processing Magazine, March 2002
- [17] Hanbiao Wang, Jeremy Elson, Lewis Girod, Deborah Estrin, Kung Yao: Target Classification and Localization in Habitat Monitoring, Computer Science Department, In ICASSP, Hong Kong, China, April 2003
- [18] Kris Pister UC Berkley and MLB Co: Smart Dust, CA, USA
<http://robotics.eecs.berkeley.edu/~pister/SmartDust/>
- [19] I.N. Broustein, K. A. Semendjajew, G. Russiol, H. Rühlig: Taschenbuch der Mathematik, Verlag H. Deutsch
- [20] Jan Beutel: Geolocation in a PicoRadio Environment, Master Thesis, 1999
- [21] L. Girod, D. Estrin: Robust Range Estimation using Acoustic and Multimodal Sensing, IROS 2001, Maui, Hawaii, Oct 2001
- [22] A. Sauvides, C.C. Itan, M.B. Srivastava: Dynamic Fine-grained Location in Ad-Hoc Networks of Sensors, MobiCom 2001, Rome, Italy, Oct 2001
- [23] Warneke: SmartDust
<http://www-bsac.eecs.berkeley.edu/~warneke/SmartDust/>
- [24] Jan Beutel, Oliver Kasten: A Minimal Bluetooth-Based Computing and Communication Platform, Technical report, ETH Zurich, May 2001

Appendix A

Communication Protocol

In the following we present a detailed description of the messages used in the communication protocol described in Chapter 5.

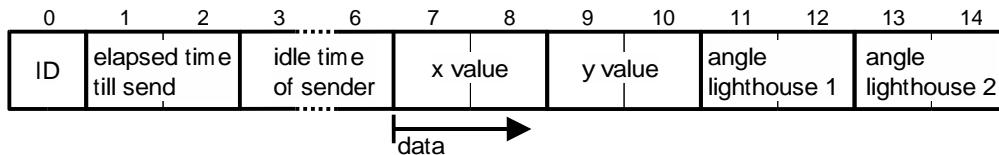
Bytes are numbered from left to right. Multi-byte quantities are stored in little endian.

Byte 0 of each message contains an ID identifying the message type. The following 6 bytes contain a timestamp as described in Chapter 4.5.

The remainder of the message depends on the message type.

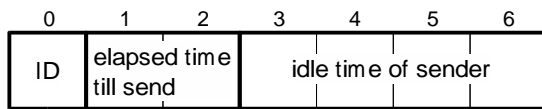
Position Message

Type: POSITION (ID=0) (node to base station)



- ID: 0
- elapsed time till send: time between occurrence of event and message being sent. For more detail, refer to Chapter 4.5.
- idle time of sender: time between two Bluetooth activities (send/receive or receive/send). For more detail, refer to Chapter 4.5.
- x value: node position x
- y value: node position y
- angle lighthouse 1: angle of $lighthouse_1$ (x axis)
- angle lighthouse 2: angle of $lighthouse_2$ (y axis)

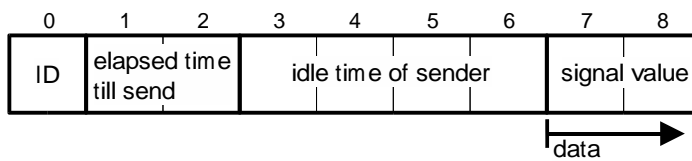
Type: POSITION_ACK (ID=1) (base station to node)



- ID: 1

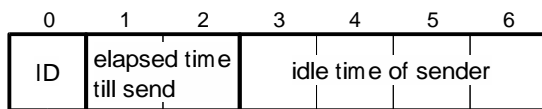
Detection Message

Type: DETECT (ID=20) (node to base station)



- ID: 20
- signal value: measured signal value

Type: DETECT_ACK (ID=21) (base station to node)



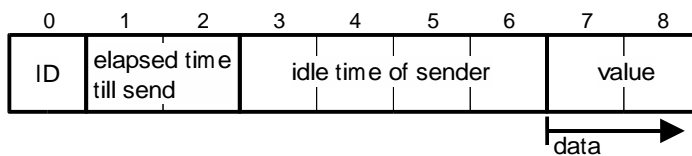
- ID: 21

Threshold Message

Type: THRESHOLD_SET (ID=10) (base station to node)

Type: THRESHOLD_ACK (ID=11) (node to base station)

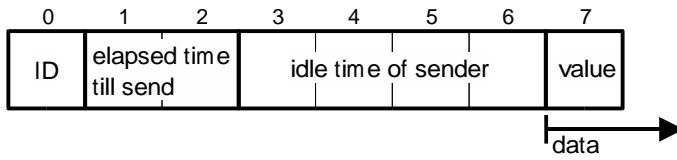
Type: THRESHOLD_NACK (ID=12) (node to base station)



- ID: 10 / value: gap value
- ID: 11 / value: threshold
- ID: 12 / value: threshold

Reading Rate Message

Type: **READING_RATE** (ID=30) (base station to node)



- ID: 30
- value: reading rate value

Appendix B

Lighthouse Calibration

As described in [1], the relationship between d (distance from lighthouse rotation axis), h (height over lighthouse center) and α can be described by the following formula for the 2D case:

$$2d\sin(\alpha) = C^b + \sqrt{d^2 + h^2}C^\beta + hC^\gamma$$

C^b , C^β and C^γ are calibration parameters defined by the lighthouse geometry. They can be obtained by measuring α_i , d_i and h_i for a set of reference points i and solving the resulting overdetermined system of equations.

Every measured point i adds the following equation to the system of equation:

$$2d_i\sin(\alpha_i) = C^b + \sqrt{d_i^2 + h_i^2}C^\beta + h_iC^\gamma$$

by rewriting this formula as

$$U_i = C^b + V_iC^\beta + W_iC^\gamma$$

$$\text{with } U_i = 2d_i\sin(\alpha_i), \quad V_i = \sqrt{d_i^2 + h_i^2}, \quad W_i = h_i$$

we obtain the following overdetermined equation system with n reference points ($n > 3$):

$$\begin{pmatrix} 1 & V_1 & W_1 \\ 1 & V_2 & W_2 \\ 1 & V_3 & W_3 \\ \vdots & \vdots & \vdots \\ 1 & V_n & W_n \end{pmatrix} \begin{pmatrix} C^b \\ C^\beta \\ C^\gamma \end{pmatrix} = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_n \end{pmatrix} \Rightarrow A\vec{x} = \vec{b}$$

In order to solve this “least squares” problem, we have to solve $A^T A \vec{x} = A^T \vec{b}$ [19].

With

$$A^T A = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ V_1 & V_2 & V_3 & \cdots & V_n \\ W_1 & W_2 & W_3 & \cdots & W_n \end{pmatrix} \begin{pmatrix} 1 & V_1 & W_1 \\ 1 & V_2 & W_2 \\ 1 & V_3 & W_3 \\ \vdots & \vdots & \vdots \\ 1 & V_n & W_n \end{pmatrix} =$$

$$\begin{pmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n V_i & \sum_{i=1}^n W_i \\ \sum_{i=1}^n V_i & \sum_{i=1}^n V_i^2 & \sum_{i=1}^n V_i W_i \\ \sum_{i=1}^n W_i & \sum_{i=1}^n W_i V_i & \sum_{i=1}^n W_i^2 \end{pmatrix}$$

and

$$A^T \vec{b} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ V_1 & V_2 & V_3 & \cdots & V_n \\ W_1 & W_2 & W_3 & \cdots & W_n \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n U_i \\ \sum_{i=1}^n V_i U_i \\ \sum_{i=1}^n W_i U_i \end{pmatrix}$$

this results in the following equation system:

$$\begin{pmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n V_i & \sum_{i=1}^n W_i \\ \sum_{i=1}^n V_i & \sum_{i=1}^n V_i^2 & \sum_{i=1}^n V_i W_i \\ \sum_{i=1}^n W_i & \sum_{i=1}^n W_i V_i & \sum_{i=1}^n W_i^2 \end{pmatrix} \begin{pmatrix} C^b \\ C^\beta \\ C^\gamma \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n U_i \\ \sum_{i=1}^n V_i U_i \\ \sum_{i=1}^n W_i U_i \end{pmatrix}$$

which can be solved by Gaussian elimination.

With the twelve reference points:

x	y	alpha_1	alpha_2
(100,	50)	0.24388	0.14494
(100,	150)	0.09649	0.15408
(200,	30)	0.37583	0.08796
(200,	200)	0.07171	0.09024
(100,	300)	0.05970	0.17597
(300,	130)	0.09449	0.06749
(200,	130)	0.09875	0.08852
(300,	300)	0.05096	0.07199
(400,	50)	0.22511	0.05759
(40,	200)	0.08244	0.38549
(400,	220)	0.05708	0.05148
(400,	120)	0.09615	0.05395

we obtain the following calibration values for the two lighthouses:

$$C_0^b = 23.6532cm, C_0^\beta = 0.0578964, C_0^\gamma = -0.0615746$$

$$C_1^b = 24.7447cm, C_1^\beta = 0.0578964, C_1^\gamma = -0.0185414$$

To estimate the error with these approximated calibration values, we consider the residue values: $\vec{r} = A\vec{x} - \vec{b}$. Listed bellow the residuen error of each reference point of *lighthouse*₁ and *lighthouse*₂.

x	y	LH1	LH2
(100,	50)	-0.0790	0.7000
(100,	150)	-0.6755	0.2568
(200,	30)	1.0662	-0.2789
(200,	200)	-0.7425	-0.6108
(100,	300)	0.4285	0.1043
(300,	130)	-0.3379	-1.1994
(200,	130)	-0.3559	-0.7918
(400,	50)	0.0805	-1.1421
(40,	200)	0.5549	1.0462
(400,	120)	0.2204	1.0914
(300,	300)	-0.6172	-2.2488
(400,	220)	0.4575	3.0732

For small d and h values the measurement error are rather small, but with increasing distance the error rapidly increases (last two reference points). We see in the last two rows that the error with increasing distance results from *lighthouse*₂ placed on the y-axis (in other words the x position in increasing distance is not very accurate if the y value is large as well). Computing the calibration values without these last two points showed no big difference in the resulting calibration parameters.

The higher error of *lighthouse*₂ is ascribed to mechanical problems (vibrations of the rotating laser beam).