SNOW BAT: A high precise WSN based location system

Marcel Baunach, Reiner Kolla, Clemens Mühlberger

Department of Computer Science V
University of Wuerzburg, Bavaria, Germany
{baunach, kolla, muehlberger}@informatik.uni-wuerzburg.de

Abstract As various applications require concrete knowledge about the position of objects, different methods for localization exist. Unfortunately, most off the shelf methods lack precision or are not applicable indoors. In this paper we present SNOW BAT, a full featured decentralized ultrasonic system for three dimensional localization and tracking of mobile objects. SNOW BAT is a sensor net solution supporting self-configuration for easy deployment, robustness for fault tolerance and simultaneous monitoring of several objects for increased scalability. Applied techniques and results will be discussed in detail allowing a brief comparison to similar systems available.

1 Introduction

Many applications depend on precise spatial knowledge of various objects. For example a mobile robot in a factory can only work properly if it knows its own position and the position of the objects to be handled. Another scenario is explained in [1]. Here a distributed sensor network detects the source of gunfire by analyzing sound propagation. Several localization systems optimized for different requirements are available. Some of them are named in [2]. For example GPS is designed for coarse outdoor navigation. Thus it does not meet various sensor network constraints because it is somewhat slow, expensive and relies on a satellite infrastructure. In fact, our robot would not come along with such a large scale system for the reasons given.

We aim on a highly precise location system based on a wireless sensor network (WSN) with special focus on easy deployment, scalability, fault tolerance, energy efficiency and high localization frequency. That's why we use small and efficient hard- and software based on the versatile sensor node SNOW^5[3] which offers the required computational power and expandability. We also pay attention on other key factors like measurement speed, reliability, node size and prizing, some denoted as critical for WSNs in [4]. Real-time operation was no problem due to the node's operating system SMARTOS [5] and its wireless MAC protocol SMARTNET.

To address the realization of the goals mentioned above in detail, the paper is structured as follows: Basic concepts and characteristics of localization systems are shown in section 2. The SNOW BAT system and its applied algorithms are described in section 3 followed by the explanation of the corresponding hard- and software in section 4. A short comparison to similar ultrasonic localization systems follows in section 5. The paper is concluded with a short survey and an outlook to future work.

2 Localization concepts and algorithms

This section classifies several aspects of spatial localization methods. Initially, we'll take a look at basic characteristics of various concepts in 2.1. Approaches for distance measurement follow in 2.2, whereas section 2.3 introduces different algorithms for computing a concrete position from measured distances.

2.1 Characteristics

First, depending on the underlying application, objects can be localized either *relative* to each other or *absolute* to a given reference point or coordinate system respectively. Next, localization can occur *periodically* or just *sporadically* when required. Furthermore, the *initiator* of the localization process can either be the object to be located itself or the surrounding environment. We also distinguish between *active*, *passive* and *interactive* localization. In the first case a device determines its position autonomously within a passive environment whereas in the second case the position of a passive object is determined solely by the environment. Interactive localization combines both methods and thus requires adequately equipped object and environment. The final points to consider are whether the implementation of the algorithm supports only *2D* or even *3D* localization and if it computes fast enough to *track* mobile nodes up to a certain velocity or just static ones.

As mentioned in [6], localization systems can also be determined by the coupling of system's anchor nodes. In a *tightly coupled* system the anchors are wired to a central unit whereas all nodes of the *loosely coupled* system use wireless communication. Also according to [6], in a *centralized* structure a centric device controls measurement and calculates the locations, unlike in *decentralized* systems. Where the first requires devices with sufficient computational power, the latter causes considerable network traffic resulting in increased energy consumption of the overall system. In *hierarchical* structures special units are privileged, that means more to control but also more to compute. Additionally, an *iterative* system needs a few runs for localization. However, all concepts have to deal with fault tolerance to compensate measurement errors and faulty nodes.

2.2 Distance measurement

As mentioned in [2], different measurement approaches exist to acquire concrete distances to objects. Some, like *time of arrival* (ToA), *time difference of arrival* (TDoA) and *angle of arrival* (AoA) make use of signal traveling times or reception angles respectively. Others, like *received signal strength indication* (RSSI) refer to a signal property that depends among other things on the traveled distance.

2.3 Positioning algorithms

For geometric position calculation of an object o some methods are outlined in [7]: *Hyperbolic trilateration* is a basic technique most commonly used in ToA systems. It

computes the position of o by intersection of at least three circles representing the distances from well-known reference positions. *Triangulation* is applied in AoA systems and thus uses the angles between o and some reference points to compute o's position. However, sensing of angles is sometimes hard to solve. Unfortunately, both methods are rather useless for real world applications as measurement is always inaccurate due to loose sensor data with fixed resolution, rounding errors, etc. For example three or more circles whose radii were measured will most likely not intersect in one common point.

Multilateration is dedicated to TDoA systems and resolves these problems by using an (overestimated) system of distances to three or more reference points. Various maximum likelihood algorithms can be applied to minimize the error between the measured and the estimated distances.

3 The SNOW BAT system

This section describes the intended operation modes for the SNOW BAT system beginning with its basic concepts and the classification in section 3.1 according to the characteristics given (2.1). Techniques for distance measurement (3.2) and position calculation (3.3) will be illuminated afterwards. Methods for deployment close this section.

3.1 Basic concepts and classification of the SNOW BAT system

The SNOW BAT system is intended for high speed and precise 3D localization of sensor nodes within a concrete environment. Since the SNOW BAT hardware is very compact, it can easily be mounted on almost any object. By using a combination of mobile and static nodes, tracking the position of a mobile robot in a production hall is as imaginable as guiding a model helicopter during takeoff from or landing on an especially prepared platform outdoors. Solely the setup, mounting and orientation of the nodes must be adapted to the specific application. The basic idea is to equip mobile objects with sensor nodes carrying ultrasonic (US) transmitters and the environment with static sensor nodes carrying ultrasonic receivers. Both types of nodes use radio transceivers for time synchronization, data transmission and initiation of the localization process. To track objects in two or even three dimensions, SNOW BAT specifies positions as absolute coordinates within a virtual coordinate system embedded in the observed environment and offers a resolution of $1\ \mathrm{mm}$. The point of origin can be chosen freely during deployment, which is greatly simplified due to self calibration of the static anchor nodes $S_1 \dots S_n$ (see section 3.4). It can even be readjusted during operation.

It is important to notice, that all algorithms and computations described within this paper are executed on the SNOW^5 sensor node featuring TI's micro controller MSP430 at $8\ \mathrm{MHz}$. No additional hardware like PDAs or even full grown computers are required, except for observation and debugging. Furthermore, no wiring between nodes is needed as even time synchronization is accomplished via wireless communication.

The localization process as shown in figure 1 designates SNOW BAT as a decentralized system of three stages: time synchronization, distance measurement and position computation.

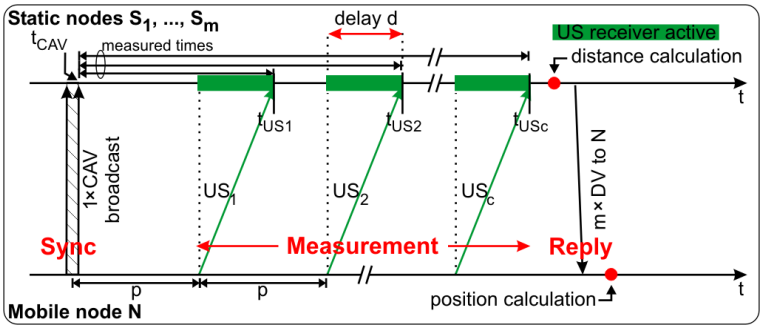

Figure 1. The SNOW BAT localization process using multichirp distance measurement

Localization can be initiated individually by a mobile client node N or by the environment for a single or for several nodes at once. This can be done sporadically as well as periodically. However, exact localization always relies on measuring the distances between a client and at least m anchors with $m \geq 1 + dim$ where dim denotes the dimension of the observed space. Overestimating this linear system yields increased fault tolerance with each additionally calculated distance.

The TDoA between a synchronizing radio packet and an ultrasonic signal is used for distance computation. Both signals will be emitted by the client. The radio signal can be sent as single- or multicast for addressing one ore more anchors at once. An US signal consists of one or more periodic chirps (unichirp / multichirp) and will be detected by all listening anchors within US range. The TDoA and the corresponding distance is computed by each anchor after receiving both signals and sent back to the client along with the anchor's absolute position. Finally the client uses multilateration with a special maximum likelihood algorithm outlined in section 3.3 to compute its position. Thus, SNOW BAT is an interactive system according to section 2.1.

3.2 Time synchronization and distance measurement

As SNOW BAT is a TDoA system, it determines the delta in the time of flight of two differently fast signals s_1 and s_2 traveling through space from a point A to a point B. This time difference will be called delay d. In the case of SNOW BAT s_1 is a radio signal propagating with the speed of light $v_{\text{radio}} = 299.792.458\ \text{m/s}$ (in vacuum) and s_2 is an ultrasonic wave whose speed actually depends on the medium it traverses. For now, we will assume air at $20\ ^\circ\text{C}$ and $101.3\ \text{kPa}$ as medium and thus $v_{\text{US}} = 343\ \text{m/s}$. Since the range of our US transmitter is at most $s_{max} = 10\ \text{m}$, we neglect the time of $33.4\ \text{ns}$ the radio packet requires to travel this distance, because is is too fast to be measured by our hardware and the evolving loss in accuracy is only about $0.01\ \text{mm}$.

Whenever a node N wants to localize itself, it sends a single hop radio message called *chirp allocation vector* (CAV) to all static nodes it wants to use for distance measurement. Since SMARTOS' [5] MAC protocol SMARTNET supports grouping of nodes by sub addresses, this can either be a broadcast to all nodes in range, to a certain group of nodes or even to a specific one. In order to limit the number of reached nodes

to a reasonable value, the radio TX power is adjusted during operation. In a perfect scenario exactly the nodes receiving a CAV will also receive the US signal later on. Anyway, the m static nodes that received the CAV are called $S_1 \dots S_m$, where $m \leq n$.

The CAV fulfills two functions: First, it is required for wireless time synchronization between sender and receivers which is indispensible within TDoA systems. Second, it contains 28 bytes of information describing the characteristics of the US signal emitted by client N and to be expected at a static node S_j $(1 \leq j \leq m)$:

- *chirpCount* c specifies the number of chirps that will be emitted by N
- *period* p specifies the time in μs between the end of CAV transmission and the beginning of the first chirp as well as the period of subsequent chirps if $c > 1$
- *timestamp* t specifies the time in μs when the CAV was sent by N. This information will only be used as identifier for matching the CAV to the results of the measurement.

SNOW BAT uses ChipCon's CC1100 [8] as radio transceiver for it is already available on the SNOW^5 node, provides sufficient data rates up to $500\ \mathrm{kbit/s}$ and various modulation methods. Besides, its advanced integration into SMARTOS allows the SNOW BAT system to perform very efficient time synchronization by using the extremely low latency interrupt handling methods offered by this real time operating system [5]. The overall transmission time of one CAV plus synchronization, preamble and CRC checksum takes approximately $d_{\mathrm{CAV}} = 1.3\ \mathrm{ms}$ at $250\ \mathrm{kbit/s}$. The time shift between sender and receiver can be determined with a precision of $2\ \mu$s which corresponds to a distance of $0.69\ \mathrm{mm}$ for an US chirp to travel. A wired infrastructure for synchronization as proposed by some other similar systems (see section 5) would improve this inaccuracy just insignificantly as it is mainly caused by the inevitable interrupt latency of the MCU.

As soon as the successful transmission of the CAV is indicated by the radio transceiver, N waits for exactly one period p before emitting the first out of c US chirps. One chirp consists of an user definable number of US pulses. We found out, that the optimal chirp is 12 pulses long as this is enough to guarantee the proper settling of the US transmitters and receivers within a maximum US range of $10\ \mathrm{m}$ and short enough to allow a rapid multi chirp measurement due to fast fading echoes. Nevertheless, the length of a chirp is adjustable at runtime to dynamically increase or decrease its range.

To easily understand the detection of the TDoA between the CAV and the chirp, we refer to the figures 2 and 3 which show detailed scope snapshots of the relevant signals. Channel 1 shows the US transmitter of the mobile node N. Channel 3 displays the output of the US receiver of the static node S_j whose base voltage can be adjusted to control the receiver's sensitivity. Additionally, the radio transmitter induces a slight interference in this signal's test line which helps to easily observe the CAV. With a high level, channel 4 indicates that S_j is ready to detect an incoming US chirp.

Once an anchor S_j receives a valid CAV, it synchronizes to N by computing its own local time ($\pm 1\ \mu$s) when the CAV was completely sent by N. From this point in time it waits for one period p to elapse and then activates its US receiver (signal 4 switches to high level). Virtually at the same moment, the sender sends its first chirp.

The reference output voltage of the US receiver V_{Ref} is software configurable to adjust the sensitivity of the US detector. As soon as it passes a fixed threshold of $V_{\mathrm{thr}} = 1.65\ \mathrm{V}$, the MCU's capture compare unit throws an interrupt to immediately record the

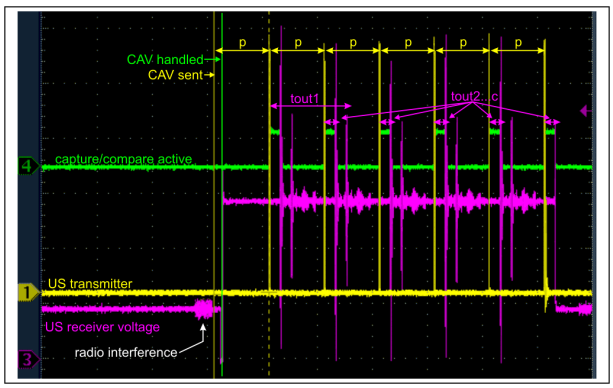

Figure 2. SNOW BAT multichirp distance measurement signal

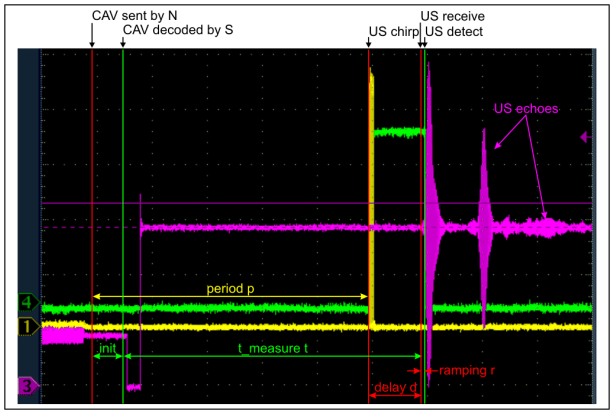

Figure 3. SNOW BAT multichirp distance measurement signal zoomed to the first chirp

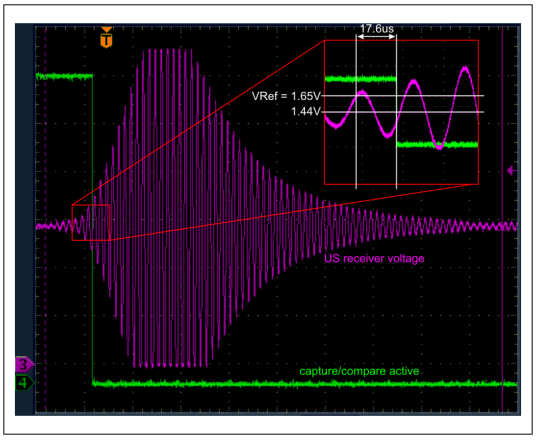

Figure 4. Ultrasonic chirp detection (40 kHz, 12 periods)

current time in μs. This happens with a constant latency of 7.5 μs and will immediately be compensated by SMARTOS. Afterwards the receiver will be deactivated for the following reasons: first it saves energy (see figure 6), second this avoids the misinterpretation of detected reflections like echoes or multipath waves as chirps and finally, other mobile nodes could interleave their localization without mutual disturbance. If another chirp is to be expected, the receiver will be reactivated after another period p and thus at the same instant the sender is supposed to send the chirp.

Unfortunately, we have to deal with a physical phenomena generally introduced by ultrasonic transceivers [9]: the signal generated by the MSP430's capture compare unit to stimulate the US transmitter is a clean 40 kHz square wave. Nevertheless, the US transmitter does not transform the signal lossless into an acoustic wave but needs a short settling time called *ramping*. This problem gets even worse at the receiver which is much more sluggish when synchronizing to the incoming wave. Figure 4 shows the received chirp signal at a frequency of 40 kHz. The slow but clearly observable rising of the amplitude also depends on the traveled distance and can sporadically result in chirp detection shifted by one period in either direction – even on immobile nodes. This involves a possible measurement error of ± 8.6 mm. We found out, that using $V_{\text{Ref}} = 1.44$ V reduces the likelihood of this problem to occur significantly. Additionally, the first few amplitudes will never be detected. Choosing V_{Ref} and V_{thr} too close results in false detection of noise as chirp. Fortunately, the time t_{ramping} between the arrival and the detection of the signal is nearly constant within two periods and thus it is easy to compensate (see formula (1)).

To yet obtain high precision SNOW BAT offers multichirp measurement as this attenuates the inaccuracies described so far. It reduces fluctuations of subsequent distance measurements caused by the ramping phenomena and the unavoidable interrupt latency of the MCU. The idea is to send several chirps and use the average TDoA for distance calculation. Different test series showed that a chirpCount $c = 3$ produces a good trade off between accuracy and the required time T for the measurement (see formula (2)).

To already achieve maximum fault tolerance during the US detection, the implementation of the delay calculation copes with adverse circumstances. It is immune against missing some of the c chirps and still provides a useful result: Let t_{CAV} be the time when the CAV was received and $t_{\text{US}_1} \dots t_{\text{US}_c}$ the times when the corresponding US chirp was detected. Additionally, v_i for $i = 1 \dots c$ is 1 if the i-th chirp was detected, 0 otherwise. Finally $V = \sum_{i=0}^{c} v_i$ is the total number of detected chirps. If at least one chirp was detected, the delay can be computed as follows (see also figure 1):

$$d = \frac{\sum_{i=1}^{c} (t_{\text{US}_i} - t_{\text{CAV}}) \cdot v_i - p \cdot \sum_{i=1}^{c} i \cdot v_i}{V} - t_{\text{ramping}} \tag{1}$$

The sums in formula (1) will be computed iteratively within a for loop from $i = 1$ to c waiting for the next chirp or a timeout. During the first pass, the timeout is defined as $t_{out_1} = p + \frac{s_{max}}{v_{\text{US}}}$ allowing to measure the maximum distance $s_{max} = 10$ m. If this first timeout exceeds, S_j will abort the measurement. Otherwise, all further passes use a timeout of $t_{\text{US}_1} - t_{\text{CAV}} + 145$ μs beginning with transmission of the next chirp. This grants a small tolerance of 49 mm to the following measurements and is required for mobile nodes. If at least one chirp was detected, S_j will compute the delay d_j after the

for loop according to formula (1). Otherwise, S_j will cancel its computation and waits for the next CAV.

The final step for S_j is to convert the measured delay into a distance s_j and to return it back to N. The SMARTNET MAC protocol provides fast and safe delivery of this information. The calculation is done by the m static nodes simultaneously within time d_{distCalc}. They can also be equipped with sensors to consider information about the surrounding medium. The distance along with some other information will be returned by sending a *distance vector* DV (see below). This packet requires $d_{\text{DV}} = 1.3\ \text{ms}$ to transmit. In case that N requested the distance to more than one node at once by sending the CAV as broad- or groupcast, the DV will only be transmitted by those nodes that could perform a measurement. This behavior reduces network traffic and relieves N from processing packets containing useless information. However, if N requested the distance to a specific single static node or returning of a DV was forced within the CAV, a response DV will be send in any case potentially just indicating an out-of-range situation. The maximum amount of time T required to perform a distance measurement can now be computed with $1 \leq m' \leq m$ as total number of answering anchors and $s' = \max\{s_i | 1 \leq i \leq m'\},\ 0 \leq s' \leq s_{max}$ as follows:

$$T = d_{\text{CAV}} + c \cdot p + \frac{s'}{v_{\text{US}}} + d_{\text{distCalc}} + m' \cdot d_{\text{DV}} \tag{2}$$

To help N adapting to the environmental situation for obtaining even better results during later localizations, the DV of anchor S_j contains some additional information:

- *nodeID* of S_j
- *chirpCount* V_j specifies the number of chirps detected by S_j
- *delay* d_j specifies the measured time of flight of the US signal in µs
- *distance* s_j specifies the distance derived from the delay d_j
- $pos(S_j)$ specifies the absolute position of S_j in mm within the virtual coordinate system
- *timestamp* t as copy of the corresponding CAV's timestamp for easy matching

By analyzing the total number and the content of the received DVs, N can not only perform its localization as described in section 3.3 but also adjust its radio and ultrasonic range. Receiving many out-of-range DVs indicates that the radio traveled much further than the US signal. As this obviously caused some anchors to wait unnecessarily for the chirps from N, they were not available for the localization of other clients. In this case N reduces its radio TX power. On the other hand, N increases its radio TX power to reach more anchors if too little information returned in order to perform a precise, fault tolerant localization. Increasing the chirp length by some periods is another option to reach more nodes. Calculation of an optimal trade-off between radio power, chirp length and even chirpCount during runtime will be subject to further research.

Finally we can say, that precision and speed of the distance measurement is very high despite of several physical problems, off-the-shelf US transducers and wireless synchronization. This was achievable due to the multichirp technique, the applied hardware and the outstanding real time capabilities of SMARTOS. It takes approximately $80\ \text{ms}$ to compute all distances with $s' = 4\ \text{m}$ from a client to four anchors as required for adequate 3D localization. This even allows simultaneous tracking of some nodes with a good spatial and temporal resolution.

3.3 Multilateration with progressive maximum likelihood algorithm

This section roughly outlines a method to compute the client's position $pos(N)$ from m' measured distances to static nodes as described in 3.2. We are currently researching a new fault tolerant and distributed algorithm which solves this problem for 3D space by progressively improving the result upon reception of new DVs. Thus, the algorithm starts immediately upon reception of the first DV and does not have to wait until a fixed number of distances was received. This allows the user to freely define a trade-off between precision and computation time. Additionally, incorrectly determined distances will be detected and ignored completely.

As the general algorithm is rather complex and will be addressed in future work, we'll refer to a concrete SNOW BAT scenario which is illustrated in figure 5. The goal is to track a node N moving on the floor of a hall and sending US chirps upwards. Anchors listening downwards are installed on the ceiling at known coordinates. Assuming floor and ceiling to be parallel reduces the localization problem from $dim = 3$ to $dim = 2$.

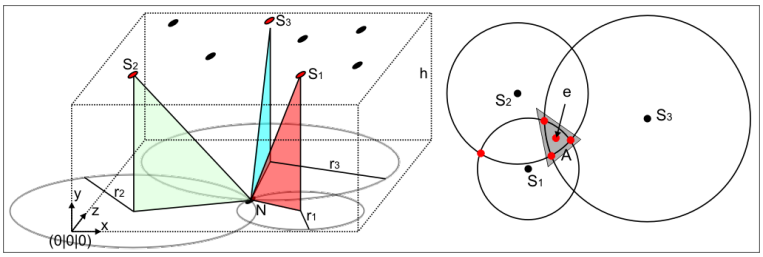

Figure 5. Position computation in two dimensions using maximum likelihood

The SNOW BAT system already supports some load balancing depending on the observed space in order to speed up the position computation. In this scenario a static node S_j knows the height h of the hall as its y-coordinate. Thus, it does not only calculate the direct distance s_j to N but also its projected distance on the floor and adds it to the DV. This corresponds to the radius r_j of a virtual circle projected to the ground with S_j as center.

We'll just outline the algorithm here referring to figure 5. As soon as N received at least two DVs, it starts to compute the intersection points of the corresponding virtual circles. In theory, the one point where all m' circles intersect is $pos(N)$. Unfortunately, there is no such common point in practice as the distance measurement will never be that precise (see 3.2). This is why SNOW BAT uses a maximum likelihood algorithm for position calculation. The algorithm iteratively intersects circles described by the incoming DVs until a position can be confirmed due to an area A containing a significant majority of densely placed intersection points or a timeout is reached. The position of N is assumed to be the geometric center of A, e.g. figure 5 shows point $e = pos(N)$.

3.4 Deployment

Deployment of the static nodes is exceptionally easy to accomplish. First, SNOW BAT does not rely on any preinstalled infrastructure. Furthermore, it does not require the anchors to be positioned along a specific pattern as some other similar systems mentioned below do. Yet it must be assured, that a mobile node can always reach $dim + 1$ static nodes within the dim-dimensional space to be observed. Another requirement is, that each static node knows its exact coordinates within the virtual coordinate system. This information can either be programmed manually into every single node during its installation or automatically via self-configuration after deploying the anchors. Whereas the first method requires a measuring tape and a lot of time, the latter only requires a mobile reference node:

The mobile reference node R performs at least $dim + 1$ distance measurements from exactly defined points within the virtual coordinate system to the static nodes in range. During this initialization stage, SNOW BAT works just the other way around described in section 3.3 as for now, the anchors localize themselves. As soon as a static node measured at least n distances to R at different positions, it can compute its own position using the algorithm outlined in 3.3. The CAV and DV packets contain special information to indicate the configuration stage: the CAV sent by R will contain its current position. A static node will only reply to a CAV by sending a DV when it localized itself successfully and thus indicates its readiness. As soon as all or a sufficient, user definable number of static nodes know their position, the configuration stage is completed and SNOW BAT is ready for action.

We found out, that manual localization of the static nodes results in a more precise operation of the SNOW BAT system than self-configuration does. This is not very astonishing as localization inaccuracy has double influence within this concept. Until now it nearly doubles the variation of the localization results. However, self-configuration greatly simplifies deployment and it is proven to work within a real world application.

4 SNOW BAT hardware and software

This chapter describes the SNOW BAT hardware and software. As the design of the ultrasonic extension board MICADUS for the SNOW^5 sensor node is rather straight forward, we'll just take a short look at some basic concepts and its interface (4.1). Main focus is given to the software implementation (4.2) using the powerful real time and multitasking capabilities of SMARTOS, an operating system for small devices like sensor nodes.

4.1 The ultrasonic hardware extension MICADUS

The SNOW BAT system uses the MICADUS extension board stacked on mobile and static SNOW^5 sensor node as basic hardware (see figures 7 and 8). Apart from a full duplex ultrasonic transceiver, the MICADUS extension features a microphone amplifier, a circuit for calibrated interfacing of up to six analog sensors, a customizable reference voltage generator and a 3 V button cell holder which can also be used for powering the

mode	power consumption [mA]
idle mode	7.6
active mode	11.3
US RX	23.7
US TX (3 V)	23.1
US TX (9 V)	1.5

Figure 6. SNOW^5 power consumption with installed MICADUS extension

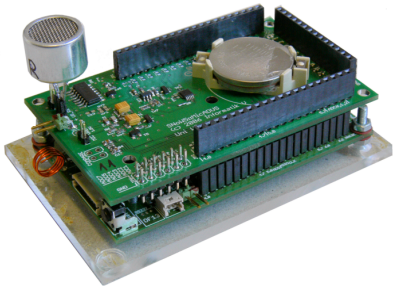

Figure 7. The SNOW^5 sensor node with stacked MICADUS extension board and US receiver

main board. Furthermore it contributes to the stackable design of SNOW^5 by passing all available signals to headers. Thus, it does not constrain the functionality of other concurrently equipped extensions and simplifies signal observation. To guarantee energy efficiency, all modules can be enabled independently (see figure 8). For each module this can either be done via software, ensuring maximum flexibility or via hardware, saving one GP-I/O pin per module at the MCU. Figure 6 outlines the power consumption of SNOW^5 with installed MICADUS extension.

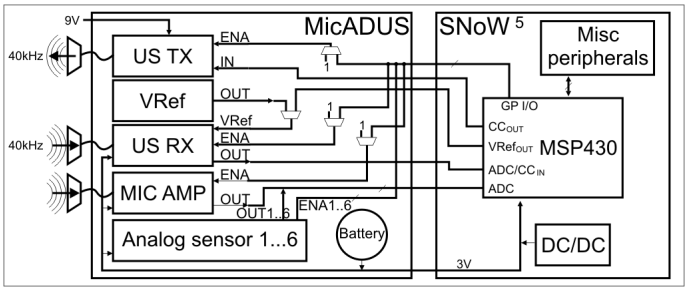

Figure 8. MICADUS schematic

The US transmitter circuit requires a 9 V power supply and drives an off-the-shelf transmitter up to 100 kHz. The desired frequency (40 kHz in our case) will be generated by the MCU's capture compare unit and is very accurate. The amplification from the MCU's 3 V to the required 9 V is done by the circuit. Note, that a single 9 V battery block is sufficient for powering the whole node when US transmission is desired.

The US receiver circuit also uses an off-the-shelf receiver and is powered with 3 V. The incoming signal will be amplified and directly passed to the MCU. It is customizable weather the signal drives an ADC or a capture compare input. Currently, we use a capture compare port as it provides insignificantly less accurate results than the ADC does. The button cell is sufficient for powering the whole node when only US reception is desired.

4.2 SNOW BAT software design using the SMARTOS real time system

The implementation of the basic software structure is rather easy due to the outstanding multitasking and event concept introduced by SMARTOS [5]. For example, active waiting loops can be completely avoided. It is even possible to run additional tasks concurrently on the node without reducing the precision of the localization. For a description of the basic SMARTOS internals refer to [5, 10]. The SNOW BAT software module was conceived as a combination of a driver controlling the US hardware and a library providing a simple interface to the functionality of the localization system. The SNOW BAT hardware was declared as SMARTOS resource and thus can only be used by one task at a time. This is required, since simultaneous usage of the US hardware by two or more tasks would corrupt the results.

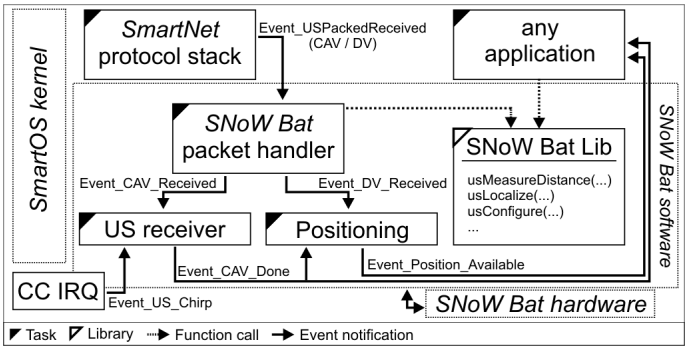

Figure 9. The SNOW BAT software design and SMARTOS integration

The library provides several functions for configuration, distance measurement and localization. A simple task which just invokes the localization process periodically and waits at most $200\ \mathrm{ms}$ for a result is as simple as this:

```
SMARTOS_TASKENTRY(TASK_US) {
  coordinate coord;
  while (1) {                    // run forever
    sleep(1000000);              // wait for 1s
    usLocalize(3, &coord);       // localize with chirpCount=3
    if (waitEventFor(&EVENT_POSITION_AVAILABLE, 200000)) { /* ok */ }
    else { /* handle error */ }
  }
}
```

Reception and evaluation of US signals is entirely encapsulated within the module and thus requires no additional code provided by the user. Internally, this is organized in three tasks which communicate via events. This makes the system highly flexible as the algorithms for distance measurement and localization can easily and individually be substituted by new ones. Even selecting one out of many implemented techniques at runtime is no problem.

The packet handler is implemented as a task infinitely waiting for incoming radio packets. Therefore it initially registers an unique SMARTNET port ID which is compara-

ble to a TCP port within the TCP/IP protocol. As soon as SMARTNET receives a packet with this ID, it passes its content along with the timestamp of reception to the SNOW BAT packet handler. The time of reception specifies the end of the radio reception in µs and equals the end of the packet transmission at the sender. A constant time lag for packet handling will automatically compensated by SMARTNET. If the packet is a CAV, this timestamp is of utmost importance for the distance measurement as described in section 3.2. Furthermore, the packet handler will either invoke a `CAV_Received` or a `DV_Received` event depending on the packet's content.

The algorithm analyzing incoming US chirps is also implemented as infinitely running task waiting for a `CAV_Received` event. As soon as this event occurs, it activates the US receiver as well as the MCU's capture compare port and records the timestamp of every single detected US chirp within an interrupt service routine. Then it calculates the corresponding distance as described in section 3.2. The value will be returned to the node that sent the CAV. Upon request, an additional `CAV_Done` event can be invoked to notify a task of a running application.

The third task contains the positioning algorithm and waits for `DV_Received` events. Just during deployment stage, it also accepts `CAV_Done` events for self-configuration as described in section 3.4. In any case, it feeds the received DVs to the localization algorithm in use and invokes a `Position_Available` event as soon as a concrete position was calculated. In most cases, this event will notify the application's task that initiated the distance measurement.

To show the implementation of the distance measurement and the simple integration into SMARTOS, the following listing shows some interesting parts of the US receiver task:

```
SMARTOS_TASKENTRY(TASK_US_RECEIVER) {
  while (1) {
    waitEvent(&EVENT_CAV_RECEIVED);
    timeout = usData.RXPacket->period + 29155;    // 10[m] / 343[m/s] in [us]
    nextExpectedChirp = usData.usCAVReceptionTime;

    activateUSRx();
    for (cci = 1 ; cci <= usData.RXPacket->chirpCount ; cci++) {
      nextExpectedChirp += usData.RXPacket->period;
      sleepUntil(&nextExpectedChirp);
      /* activate capture compare port and hardware IRQ */
      chirpDetected = waitEventFor(&EVENT_US_CHIRP, timeout);
      /* deactivate capture compare port and hardware IRQ */
      if (chirpDetected) {
        delaySum += (usData.usChirpReceptionTime - usData.usCAVReceptionTime);
        if (cci == 1) timeout = delaySum + 145;
        pDetected = pDetected + cci;   // more periods detected
        cDetected++;                   // one more chirp detected
      } else {
        if (cci == 1) goto next;
        continue;
      }
    }
    deactivateUSRx();
    /* compute distance, assemble and transmit distance vector DV */
next:
    setEvent(&EVENT_CAV_DONE);
  }
}
```

The compiled binary of our testbed including SMARTOS, SMARTNET, SNOW BAT and a serial console driver for debugging contains a total number of 5 tasks and requires only 18 kB ROM and 2 kB RAM. The remaining 30 kB ROM and 8 kB RAM are available for additional tasks. Just like the hardware, the application is also designed to allow exactly the same binary to be flashed on static and mobile nodes. The required operation mode is jumper selectable during deployment.

5 Comparison

As mentioned in sections 1 and 2.2, various localization systems using several distance measurement techniques already exist. This section compares the localization architectures Cricket [11, 12], Active Bat [13] and AHLoS [14] to SNOW BAT, all using ultrasonic and radio as ranging techniques for measurement. Table 1 gives a short survey of the named localization systems.

As described in section 3, the active clients of SNOW BAT initiate the localization process by sending a radio signal followed by some US chirps. This minimizes the interval between two localization processes. If a client initiates measurement, it has to wait only as long as it has received enough DVs or a timeout (cf. 3.3), otherwise a client needs to wait as long as enough anchors within its US range have finally sent their US signals (e.g. in Cricket). In SNOW BAT, all anchor nodes receiving both, radio and US signal, transmit their calculated distances to the initiator who estimates its position out of these distances by multilateration (see 3.3). This decentralized and loose coupled system offers not only an overall energy saving but also sporadic localization process.

However, the tightly coupled system Active Bat allows a high precision and an accurate time synchronization by wiring all passive anchors to a central unit. But this suffers from a limitation of possible anchors and a more complicated deployment. Additionally, failure of the central unit renders the whole system inoperable.

On the other hand, the decentralized Cricket uses active anchors, periodically sending informations about their position. They divide a specific area into a grid, thus a client can calculate its location from the received signals. Drawbacks are the inaccurate localization by sectioning and the special deployment of the anchors along a fixed pattern. To avoid interference of the anchor's signals, each anchor has to wait a forced delay which makes tracking of mobile nodes impossible.

The nodes in AHLoS are either localized or unlocalized. Initially, all anchors are labeled as localized. Thus, an unlocalized client which is connected to at least three anchors can calculate its position. Otherwise, if it is connected to less than three located nodes, it must calculate its position iteratively. If a client hereupon knows its position, it becomes also localized and can act as anchor to-be. Indeed this system has a high energy consumption and lots of communication overhead by reason of iteration loops.

6 Conclusion and future work

In this paper we have shown a high precise WSN localization system using ultrasonic and radio signal as ranging techniques for measurement. After introducing some concepts and characteristics for localization (section 2), we explained the SNOW BAT sys-

tem in detail in section 3, including the technique for time synchronization and distance measurement, the used multilateration algorithm and the ease of deployment by means of self-configuration. Section 4 not only deals with the used hardware, especially the

Localization system	**Cricket**	**Active Bat**	**AHLoS**	**SNOW BAT**
structure	decentralized	centralized	iterative	decentralized
coupling	loose	tight	loose	loose
accuracy in [mm]	< 100	< 90	> 20	< 15
coordinates	relative	absolute	absolute	absolute
deployment	medium	difficult	easy	easy
anchor has US	TX	RX	RX, TX	RX (TX)
client has US	RX	TX	RX, TX	TX (RX)
scalability	medium	bad	good	good
TX node	Beacon	Bat unit	Medusa	SNOW^5
RX node	Listener	Receiver unit	Medusa	SNOW^5
nodes' MCU	PIC	Xilinx FPGA	AVR8535	MSP430
user privacy	yes	no	no	yes
other coexistent tasks supported	no	no	no	yes

Table 1. Short comparison of localization systems

MICADUS extension board for SNOW^5, but also with the implemented software based on the real-time operating system SMARTOS. A short comparison to similar localization architectures closes this paper.

To reach a an even higher accuracy than at present, the distance calculation has to reflect the speed of US signals depending on the medium and its current properties, e.g. temperature and humidity as well as pressure have a strong effect on the velocity of ultrasonic waves in air. We will also analyze a higher US frequency, since we expect strongly improved precision during chirp detection at the doubled US frequency of 80 kHz (cf. section 3.2).

On algorithmic side we are currently researching the implementation and comparison of various maximum likelihood algorithms. Computation of fitness values for anchor nodes derived from their DVs over time could probably increase the accuracy and stability of the calculated position.

We will also research the usage of the A/D converter of SNOW^5 instead of its capture compare unit as mentioned in section 4.1. Thus, not only the precision of measured distances would increase but also a profile of the environment could possibly be established. On the other hand, this could slow down the measurement frequency since the MCU has to perform complex digital signal processing.

Our long-term goal is the steering of a mobile unit equipped with the SNOW BAT system by supplying a concrete route via an user interface. Therefore, the versatility of SNOW^5 is of great advantage as it already provides several features like D/A converters for motor control. Finally, this would turn the wireless sensor network into an even more interesting wireless sensor/actor network.

References

[1] GYULA, SIMON, MIKLÓS MARÓTI, ÁKOS LÉDECZI, GYÖRGY BALOGH, BRANISLAV KUSY, ANDRÁS NÁDAS, GÁBOR PAP, JÁNOS SALLAI and KEN FRAMPTON: *Sensor network-based countersniper system*. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12, New York, NY, USA, 2004. ACM Press.

[2] TSENG, YU-CHEE, CHI-FU HUANG and SHENG-PO KUO: *Positioning and location tracking in wireless sensor networks*. In ILYAS, MOHAMMAD and IMAD MAHGOUB [15], chapter 21, pages 1–13.

[3] KOLLA, REINER, MARCEL BAUNACH and CLEMENS MÜHLBERGER: *SNoW^5: A versatile ultra low power modular node for wireless ad hoc sensor networking*. In MARRÓN, PEDRO JOSÉ (editor): *5. GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, pages 55–59, Stuttgart, July 2006. Institut für Parallele und Verteilte Systeme.

[4] BULUSU, NIRUPAMA: *Introduction to Wireless Sensor Networks*. In BULUSU, NIRUPAMA and SANJAY JHA [16], chapter 1, pages 1–19.

[5] KOLLA, REINER, MARCEL BAUNACH and CLEMENS MÜHLBERGER: *SNoW^5: a modular platform for sophisticated real-time wireless sensor networking*. Technical Report 399, Institut für Informatik, Universität Würzburg, January 2007.

[6] BULUSU, NIRUPAMA: *Localization*. In BULUSU, NIRUPAMA and SANJAY JHA [16], chapter 4, pages 45–57.

[7] BEUTEL, JAN: *Location Management in Wireless Sensor Networks*. In ILYAS, MOHAMMAD and IMAD MAHGOUB [15], chapter 20, pages 1–23.

[8] CHIPCON AS, Oslo (Norway): *CC1100 Data Sheet*, 2005.

[9] MÁGORI, VALENTIN: *Ultraschallsensoren zur Abstandsmessung und Präsenzdetektion*. In TRÄNKLER, HANS-ROLF and ERNST OBERMEIER (editors): *Sensortechnik*, chapter 10.2, pages 511–553. Springer, 1998.

[10] KOLLA, REINER: *YaOS – Yet another Operating System*, 2005. presentation.

[11] PRIYANTHA, NISSANKA BODHI, ANIT CHAKRABORTY, and HARI BALAKRISHNAN: *The Cricket Location-Support system*. In *Proceedings of the 6th Annual ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, August 2000.

[12] PRIYANTHA, NISSANKA BODHI: *The Cricket Indoor Location System*. PhD Thesis, Massachusetts Institute of Technology, June 2005.

[13] WARD, ANDY, ALAN JONES, and ANDY HOPPER: *A New Location Technique for the Active Office*. IEEE Personal Comm., 4(5):42–47, October 1997.

[14] SAVVIDES, ANDREAS, CHIH-CHIEH HAN, and MANI B. STRIVASTAVA: *Dynamic fine-grained localization in ad-hoc networks of sensors*. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 166–179, New York, NY, USA, 2001. ACM Press.

[15] ILYAS, MOHAMMAD and IMAD MAHGOUB (editors): *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC Press, 2005.

[16] BULUSU, NIRUPAMA and SANJAY JHA (editors): *Wireless Sensor Networks - A Systems Perspective*. Artech House, 2005.