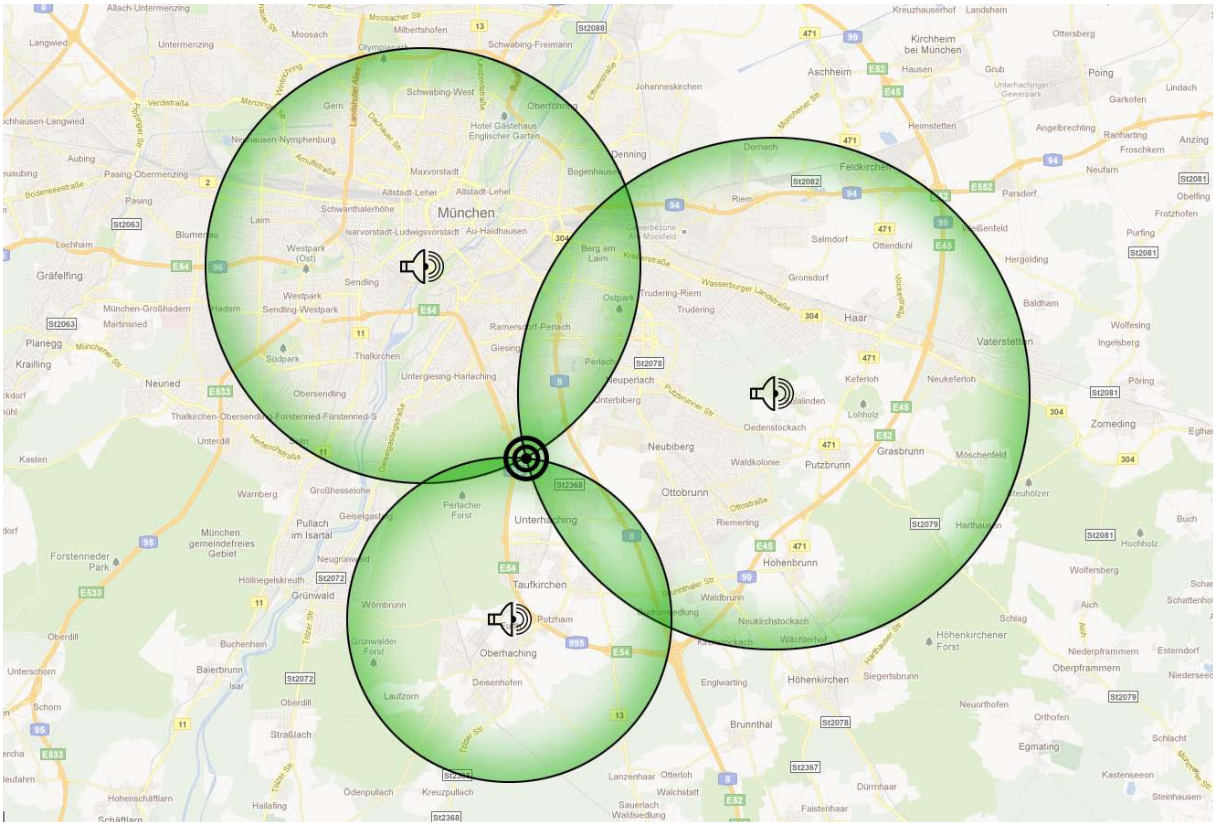
Universität der Bundeswehr München

Ingenieur Mathematik

<u>Projektarbeit</u>

Positionsbestimmung mit Ultraschall

2011



Autor: Friedemann Sebastian Winkler, Matr. Nr. 1080395

Betreuer: Prof. Dr. Gerdts, Institut für Mathematik und Rechneranwendung, LRT-1

Abgabe: 29.11.2011

Kurzfassung

Die genaue Bestimmung der Position eines Roboters ist ein fundamentales Problem in vielen Anwendungen. Auf der Suche nach einer Lösung wurden die verschiedensten Verfahren und Sensoren angewandt. Diese Arbeit konzentriert sich nach einem kurzen Überblick über die Verfahren auf die Positionsbestimmung in einer bekannten Umgebung mithilfe von Ultraschall auf der Grundlage des Global Positioning Systems. Verschiedene Möglichkeiten, diese Technologie zu nutzen, werden untersucht und auf Praxistauglichkeit getestet.

Grundproblem ist dabei die Synchronisation von Ultraschallsender und -empfänger. Auf die grundlegende Bestimmung von Messfehlern und Übertragungsgeschwindigkeiten bei der Übermittlung von Synchronisationssignalen folgen Implementierung und Test der Verfahren mit einem Phidget Single Board Computer. Zur Datenübertragung werden dabei insbesondere W-LAN, LAN und Bluetooth genutzt. Daneben kommt auch eine Variante ohne Synchronisierungssignal zum Einsatz, bei der die Synchronität durch gleich laufende Systemuhren gewährleistet wird.

Abstract

Exact knowledge of the position of a robot is a fundamental problem in many robotic applications. Many systems, sensors and techniques for mobile robot positioning have been introduced. After giving a short overview of the different approaches, this paper deals with positioning in known environment with ultrasonic sensors, based on the Global Positioning System. Several approaches to employ this technology are analyzed and their operational suitability is tested.

The basic problem is the synchronization of ultrasonic sender and receiver. In this paper measuring errors and signal delays of different methods are investigated and implementations on a Phidget single board computer are proposed and tested. In particular the transmission of synchronization signals is accomplished via LAN, W-LAN and Bluetooth. In addition an alternative solution without synchronization signal is tested that utilizes synchronized system clocks.

Inhaltsverzeichnis

1 Einleitung

Leonard und Durrant-White [01] haben das Grundproblem der Roboternavigation in drei Fragen zusammengefasst: „Wo bin ich?“, „Wohin bewege ich mich?“, „Wie kann ich dort hin gelangen?“.

Bis heute gibt es keine wirklich elegante Lösung für dieses Problem. Die verschiedenen Ansätze können in zwei Kategorien unterteilt werden: Relative und absolute Positionsbestimmung. In Ermangelung einer einzelnen, überall funktionierenden Methode werden in aktuellen autonomen Robotern meist zwei Ansätze kombiniert, jeweils einer aus jeder Kategorie.

1.1 Methoden der Positionsbestimmung

In diesem Abschnitt werden einige Sensoren und Herangehensweisen vorgestellt, mit denen eine Positionsbestimmung durchgeführt werden kann. Hierbei werden insbesondere die Odometrie und inertiale Navigation als Vertreter der relativen, sowie die Triangulation und Trilateration als Vertreter der absoluten Positionsbestimmung vorgestellt.

1.1.1 Odometrie

Odometrie ist die am häufigsten verwendete Methode zur Positionsbestimmung und Navigation von mobilen Robotern. Sie bietet Präzision auf kurzer Distanz, ist billig und erlaubt hohe Abtastraten. Grundidee der Odometrie ist die Aufzeichnung von Bewegungsinformationen. Man verwendet zum Beispiel Umdrehungszähler an den Rädern des Roboters und nutzt den linearen Zusammenhang zwischen der Anzahl an Umdrehungen und der zurückgelegten Strecke.

Aus der Grundidee folgt auch der Nachteil der Odometrie: Es wird keine absolute Position bestimmt, sondern eine relative, bezogen auf den Startpunkt. Durch die Integration der Messwerte über die Zeit entsteht unvermeidbar ein Fehler. Dieser wird immer größer, je weiter der Roboter fährt (Drift). Besonders problematisch sind Fehler in der angenommenen Orientierung, da sich durch den Winkelfehler der absolute Fehler in der Positionsmessung bei größerer Entfernung vom Ursprungsort immer weiter vergrößert. Außerdem entstehen Fehler in Situationen, bei denen die Räder durchdrehen oder blockieren können (das Messrad sollte daher kein angetriebenes Rad sein). Stöße, die den Roboter aus seiner Bahn schieben, können nicht erkannt werden. Odometrie alleine ist also nicht für eine zuverlässige Positionsbestimmung geeignet.

1.1.2 Inertiale Navigation

Inertiale Navigationssysteme nutzen Gyroskope und Beschleunigungssensoren, um die rotatorische und translatorische Beschleunigung zu messen. Die Messergebnisse werden zwei Mal integriert, um die Position zu bestimmen. Hierdurch entsteht der große Nachteil dieser Systeme: Durch die Integration wächst jeder kleine konstante Fehler unbeschränkt an, sogar während der Roboter still steht.

Tests an der University of Michigan [02] haben ergeben, dass das Verhältnis von verwertbarem Signal zum Rauschen sehr schlecht ist, vor allem bei kleinen Beschleunigungen. Der oben erwähnte Drift beträgt 1 bis 8 Zentimeter pro Sekunde, je nach Art und Stärke der Beschleunigungsänderung. Jede Unebenheit im Boden resultiert in einer gemessenen Komponente der Erdbeschleunigung. Auch Lagesensoren, die in aktuellen Beschleunigungsmessgeräten verbaut sind, können dies nicht ganz kompensieren. Daher ist auch die inertiale Navigation alleine über längere Zeiträume nicht für die Positionsbestimmung geeignet.

1.1.3 Triangulation und Trilateration

Bei Triangulation und Trilateration wird die absolute Position des Roboters bestimmt. Dazu dienen Funkfeuer, die an bekannten Orten aufgestellt sind.

Bei der Triangulation sind mindestens drei Sender im Raum verteilt, wie in Abbildung 1 zu sehen ist. Ein rotierender Sensor an Bord misst die Winkel λ_1, λ_2 und λ_3, bei denen es die Sender relativ zur Längsachse des Roboters „sieht“. Aus diesen drei Messungen können die beiden unbekannten Koordinaten x und y sowie die unbekannte Orientierung φ berechnet werden.

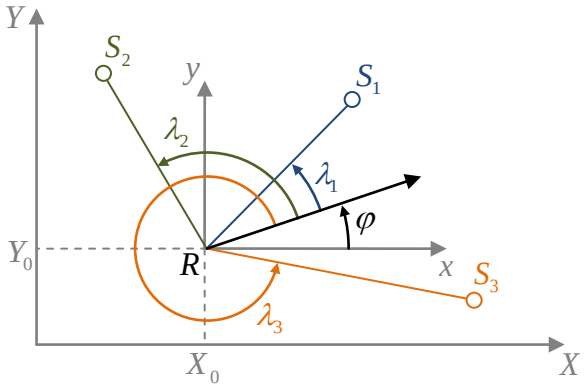


Abbildung 1: Positionsbestimmung durch Triangulation mit drei Sendern

Die Berechnung ist allerdings nicht trivial. Cohen und Koss analysieren in [03] verschiedene Algorithmen und kommen zu folgendem Ergebnis: Die „*Geometric Triangulation Method*“ funktioniert nur zuverlässig, wenn sich der Roboter innerhalb des von den drei Sendern aufgespannten Dreiecks befindet. Die „*Geometric Circle Intersection Method*“ produziert große Fehler, wenn die drei Sender auf demselben Kreis liegen. Die „*Newton-Raphson Method*“ versagt, wenn die Startposition zu schlecht geschätzt wird. Generell müssen für eine zuverlässige Berechnung mindestens zwei Sender einen Winkelunterschied von mehr als 90° haben und jedes Paar muss mindestens 45° auseinander liegen. Wegen der erheblichen Nachteile aller Verfahren werden meistens mehrere von ihnen intelligent kombiniert, um die individuellen Schwächen auszugleichen.

Bei der Trilateration werden im Gegensatz zur Triangulation nicht die Winkel zu den Sendern gemessen, sondern die jeweiligen Entfernungen. Dafür müssen die Sender ihre Signale zu bekannten Zeiten aussenden, damit die Übertragungszeit gemessen und daraus die Strecke berechnet werden kann. Dieser Ansatz, auf dem zum Beispiel das Global Positioning System (GPS) basiert, wird im folgenden Kapitel ausführlich behandelt.

1.2 Multilateration mit Ultraschall

Bei der Trilateration (oder im Fall von mehr als drei Sendern Multilateration) sind die Sender im Raum verteilt wie in Abbildung 2 dargestellt. Für diese Arbeit wurden Ultraschallsender verwendet, deren Signal sich mit Schallgeschwindigkeit ausbreitet.

1.2.1 Grundprinzip

Die Zeit vom Senden bis zum Empfangen des jeweiligen Signals wird gemessen. Die Entfernung zwischen Sender und Empfänger ergibt sich dann als Produkt von Zeit und Schallgeschwindigkeit.

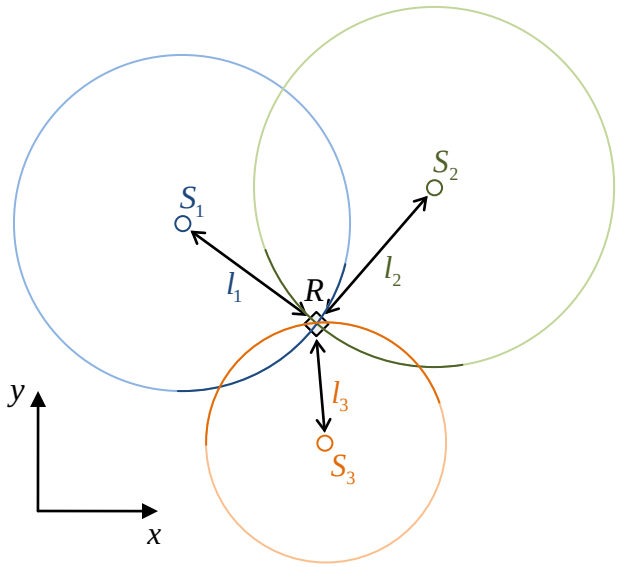


Abbildung 2: Positionsbestimmung mit Trilateration

Aus drei Entfernungen kann eindeutig die Position des Roboters bestimmt werden. Auch hier ist die Rechnung nicht trivial, es gibt aber robustere Algorithmen als bei der Triangulation. Ein Nachteil dieses Verfahrens ist, dass nur die Position, nicht aber die Orientierung des Roboters bestimmt werden kann.

1.2.2 Umsetzung

Als Roboter wurde für diese Arbeit ein Single Board Computer (SBC) des Hersteller Phidgets verwendet (siehe Abbildung 3). Auf dem SBC läuft Debian Linux, die Programme zur Positionsbestimmung wurden in C programmiert. Über USB ist ein 40 kHz Ultraschall-Entfernungssensor des Typs SRF02 der Firma Devantech (siehe Abbildung 4) angeschlossen. Dieses Bauteil ist eigentlich dazu gedacht, mit dem Sonarprinzip die Entfernung zur nächsten Wand zu bestimmen. Es wird ein Ultraschallsignal ausgesendet und die Zeit bis zum Eintreffen des Echos gemessen. Der Sensor lässt sich aber auch als reiner Empfänger verwenden. Über einen W-LAN Stick ist der SBC in das lokale Netzwerk eingebunden.

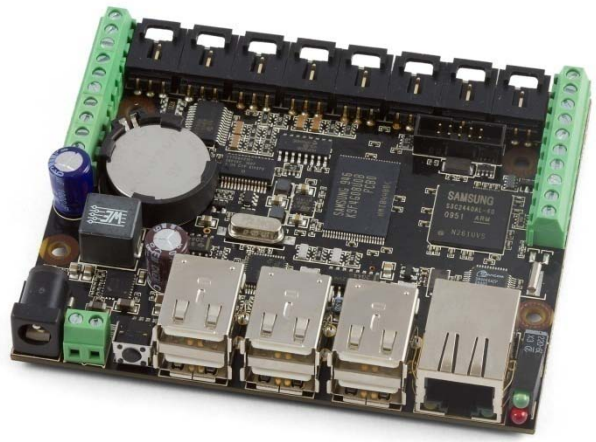


Abbildung 3: Phidgets Single Board Computer

Abbildung 4: SRF02 Ultraschallsensor

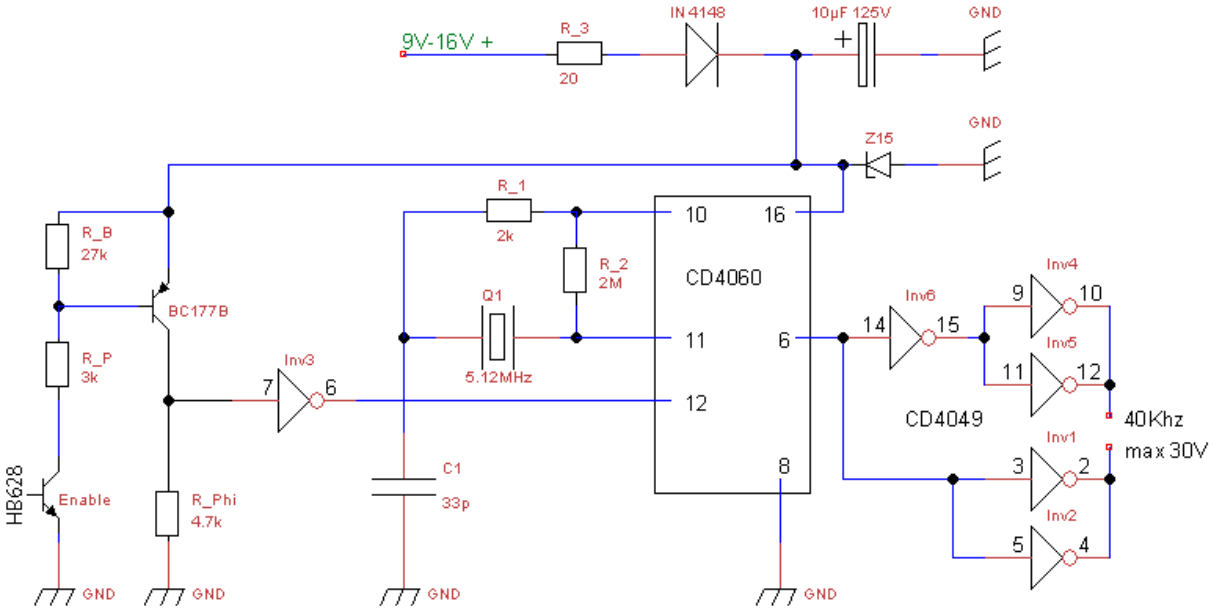


Abbildung 5: Schaltplan für einen Ultraschallsender

Als Ultraschallsender werden „400ST160“ Transmitter von Air Ultrasonic eingesetzt. Diese sind jeweils an eine quarz-stabilisierte Eigenbau-Schaltung angeschlossen (Schaltplan siehe Abbildung 5). Die vier Sende-Module sind alle gemeinsam an ein USB-Datenerfassungs- und Steuersystem vom Typ HB628 von H-Tronic angeschlossen (siehe Abbildung 6), das wiederum über USB mit einem Notebook verbunden ist. Das Notebook ist über W-LAN mit dem lokalen Netzwerk verbunden. Der gesamte Versuchsaufbau ist in Abbildung 7 dargestellt.

Abbildung 6: HB628 USB-Datenerfassungs- und Steuersystem

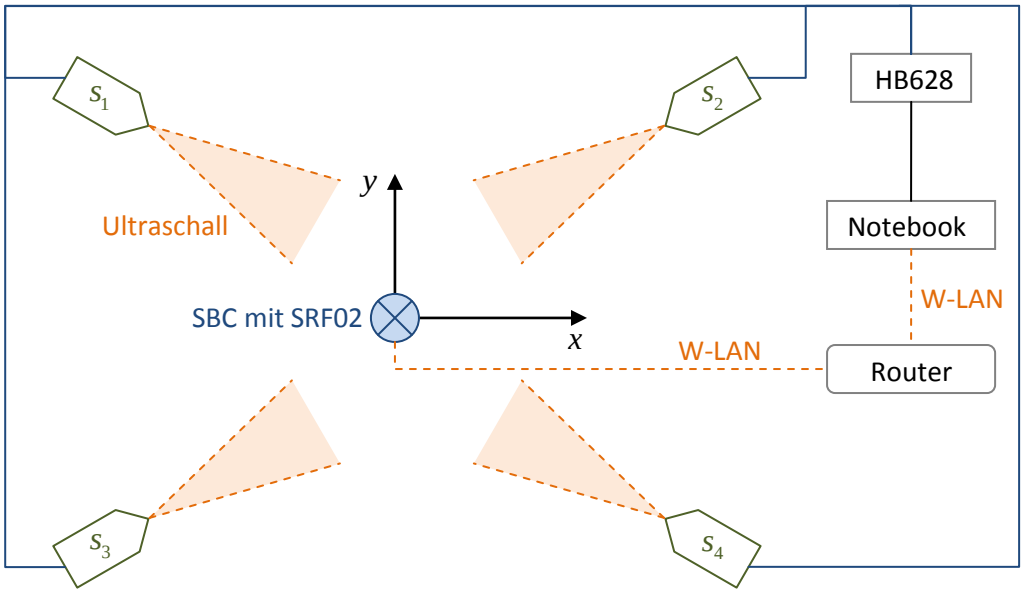


Abbildung 7: Versuchsaufbau zur Positionsbestimmung mit Ultraschall

Auf dem Notebook läuft das Programm zur Steuerung der Ultraschallsender, auf dem SBC das Programm zum Abfragen des Empfängers. Die Sender senden identische 40 kHz Signale, in die keine weiteren Informationen eingebaut werden können. Da der Empfänger den jeweiligen Sender nicht selbst identifizieren kann, darf nicht überall gleichzeitig gesendet werden. Dies wurde hier so gelöst, dass die Sender in kurzen Abständen nacheinander ausgelöst werden. Wenn alle Sender gesendet haben, beginnt nach einer kurzen Pause der nächste Zyklus. Es ergibt sich nun das Problem, dass dem SBC mitgeteilt werden muss, wann das jeweilige Signal gesendet wurde, damit die Zeit bis zum Eintreffen gemessen werden kann. Hierfür gibt es zwei Alternativen: Entweder wird zu vorher festgelegten Zeiten gesendet (etwa jede volle Sekunde) und die Systemuhren von Notebook und SBC werden synchronisiert. Oder es wird gleichzeitig mit dem Senden des Ultraschallsignals ein zusätzliches Synchronisierungssignal über Bluetooth oder LAN mit der Nachricht „Sender X wurde jetzt aktiviert“ übertragen. Diese Möglichkeiten werden im nächsten Kapitel diskutiert.

2 Verfahren zur Synchronisation

Die Programme zum Senden der Ultraschallsignale auf dem Notebook und zum Empfangen auf dem SBC müssen synchron laufen, um die Übertragungszeit bestimmen zu können. Die beiden Varianten „Synchronisationssignal“ und „synchrone Uhren“ haben jeweils Vor- und Nachteile, die in diesem Kapitel besprochen werden.

Das Übertragen eines Synchronisierungssignals ist immer mit einer Verzögerung verbunden. Diese resultiert aus der Ausführungszeit der entsprechenden Befehle im Programm sowie der gewählten Übertragungstechnologie. Es muss zwischen der konstanten Sendezeit und stochastischen Schwankungen unterschieden werden. Während sich die konstante Sendezeit wegkalibrieren lässt, führt das Rauschen zu unumgänglichen Messfehlern. Wir suchen also nach einer Technologie mit möglichst konstanter Übertragungszeit.

2.1 Bluetooth

Bluetooth ist eine proprietäre kabellose Technologie zur Datenübertragung über kurze Distanzen. Die Signale breiten sich mit Lichtgeschwindigkeit aus, wodurch die Sendezeit praktisch vernachlässigbar ist. Bluetooth-Geräte werden in C-Programmen mit Hilfe von Sockets angesprochen. Das verwendete Protokoll ist auf Sicherheit, Zuverlässigkeit und geringen Energieverbrauch optimiert, weniger hingegen auf die Echtzeit-Übertragung von Signalen.

Um die tatsächliche Übertragungszeit von Bluetooth-Signalen zu bestimmen, wurde für diese Arbeit ein Testprogramm implementiert. Das Programm schickt nacheinander 200 Signale an einen Empfänger. Dieser beantwortet die Signale sofort, das Sender-Programm misst die Zeit bis zum Eintreffen der Antwort. Die Hälfte dieser Zeit ist die Übertragungszeit vom Sender zum Empfänger, inklusive der Verarbeitung des Signals. Die Messwerte sind in Abbildung 8 dargestellt. Die mittlere Dauer beträgt 468 µs. In dieser Zeit legt das Ultraschallsignal eine Strecke von 16 cm zurück. Auffällig sind die Peaks, bei denen die Übertragungszeit über eine Millisekunde beträgt. Dies und die relativ großen Schwankungen deuten an, dass sich mit Bluetooth keine besonders gute Synchronisation durchführen lassen wird. Auch die Nicht-Echtzeitfähigkeit von Linux kann Ursache sein.

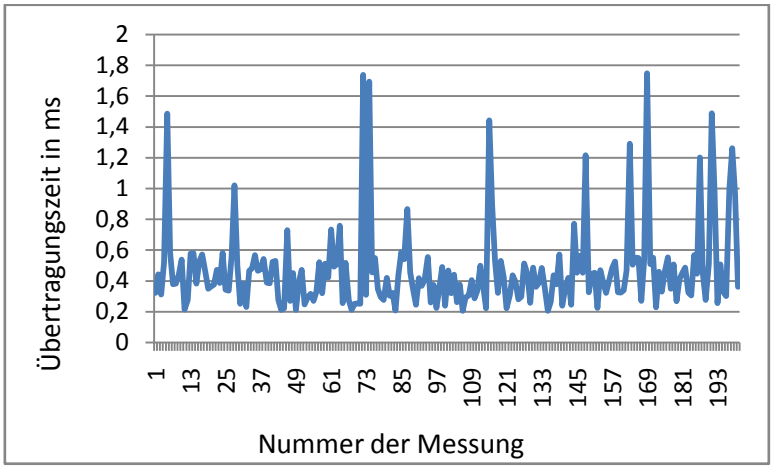


Abbildung 8: Übertragungszeit von Bluetooth-Signalen

2.2 Lokales Netzwerk

Der SBC hat keine eigenen Bedienelemente. Um das Empfängerprogramm auf ihm starten zu können, muss man sich daher vom Notebook aus über das lokale Netzwerk mit Secure Shell (SSH) beim SBC einloggen und ihn so fernsteuern. Da aus diesem Grund immer eine Netzwerkverbindung erforderlich ist, liegt es nahe, diese auch für Synchronisationssignale zu verwenden. Dafür gibt es verschiedene Protokolle, die in Frage kommen. In dieser Arbeit werden das Transmission Control Protocol (TCP) und das User Datagram Protocol (UDP) besprochen.

2.2.1 TCP über W-LAN

Das Transmission Control Protocol (TCP) ist eines der wichtigsten Bestandteile der Internetprotokollfamilie. Es bietet zuverlässige, geordnete Datenübertragung von einem Programm auf einem Computer zu einem anderen Programm auf einem anderen Computer. Bedeutende Anwendungen sind das World Wide Web (WWW), das File Transfer Protocol (FTP), Secure Shell (SSH), Peer-to-Peer Datentausch und E-Mail.

TCP übernimmt die Aufteilung der zu sendenden Daten in Pakete, die dann vom Internet Protocol (IP) gesendet werden. Bei der Übertragung durch stark belastete Netzwerke können IP Pakete doppelt oder in falscher Reihenfolge ankommen oder ganz verloren gehen. TCP bemerkt solche Probleme, fordert die erneute Übertragung des betreffenden Pakets an und sortiert die empfangenen Pakete neu, so dass sich der Benutzer darum keine Gedanken machen muss.

Um die tatsächliche Übertragungszeit zu bestimmen wurde ein Test analog zu Bluetooth (Kapitel 2.1) durchgeführt. Die Ergebnisse sind in Abbildung 9 zu sehen. Die mittlere Übertragungszeit beträgt 392 µs, also etwa 84 % der Sendezeit von Bluetooth. Es gibt außerdem weniger und kleinere Peaks. Die Varianz der Messwerte ist kleiner, die zu erwartende Messgenauigkeit somit höher. Allerdings ist zu bedenken, dass das Netzwerk im Gegensatz zu Bluetooth gleichzeitig auch anderen Anwendungen zur Verfügung steht. Wenn etwa gleichzeitig Daten aus dem Internet heruntergeladen werden, dann kann es zu sehr viel längeren Sendezeiten kommen als in diesem Test.

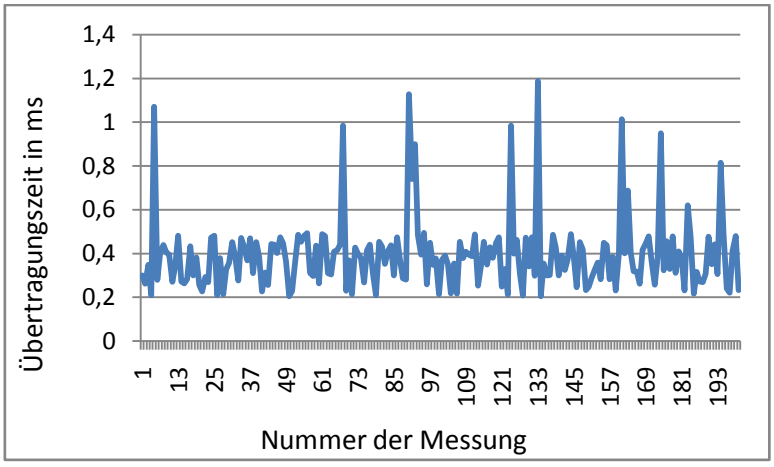


Abbildung 9: Übertragungszeit mit TCP über W-LAN

2.2.2 UDP über LAN

Das TCP ist dafür optimiert, Daten sicher und zuverlässig zu versenden. Diese Zuverlässigkeit wird mit einer längeren Sendedauer erkauft, es kann zu Wartezeiten im Sekundenbereich kommen. Für Echtzeitanwendungen wie Voice over IP (oder eben das Versenden von Startsignalen für die Positionsbestimmung mit Ultraschall) ist es daher nicht sinnvoll, dieses Protokoll zu verwenden. Für solche Anwendungen werden eher Protokolle wie das Real-time Transport Protocol (RTP) auf der Basis des User Datagram Protocols (UDP) empfohlen.

Für den folgenden Test wurde das TCP-Programm abgewandelt, so dass eine UDP-Verbindung zur Übertragung der Signale verwendet wird. Außerdem wurde die W-LAN Anbindung durch eine kabelgebundene Verbindung ersetzt, da die Latenzzeiten ansonsten zu groß waren. Um die Qualität der UDP-Verbindung zu testen, wurde die Übertragungszeit von Testsignalen im nicht belasteten und im belasteten Netzwerk analog zu den bisherigen Tests überprüft. Die Ergebnisse sind in Abbildung 10 und Abbildung 11 dargestellt. Die mittlere Übertragungszeit ist 251 µs bzw. 956 µs. Damit belegt UDP gleichzeitig den ersten und den letzten Platz. Es muss bei der späteren Verwendung auf jeden Fall darauf geachtet werden, dass das Netzwerk nicht durch andere Prozesse belastet wird.

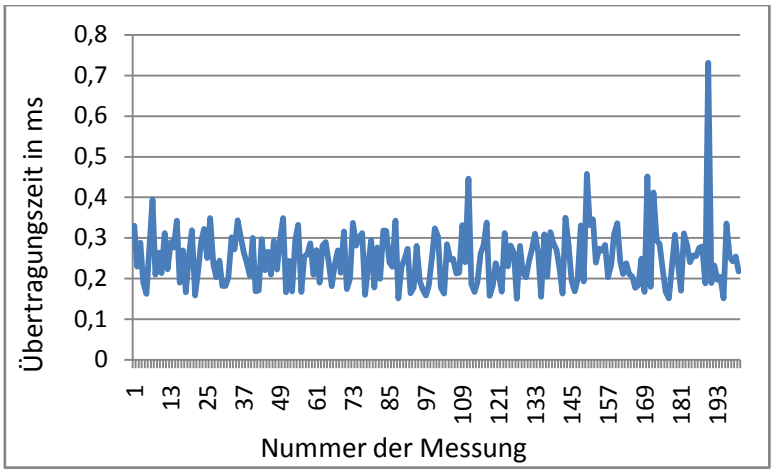


Abbildung 10: Übertragungszeit mit UDP im nicht belasteten LAN

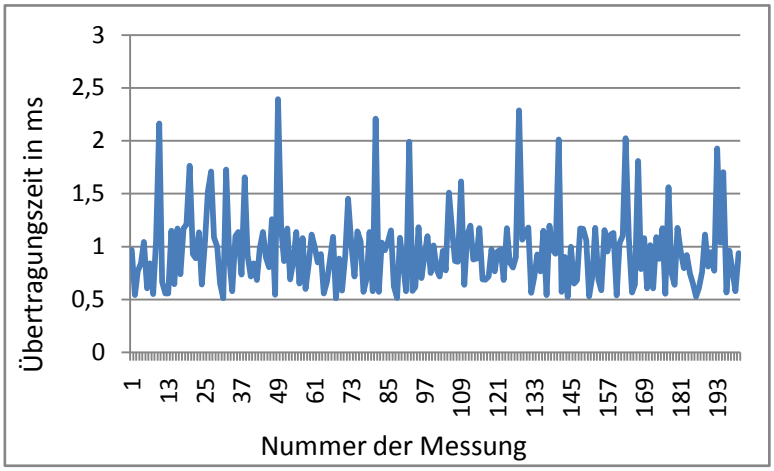


Abbildung 11: Übertragungszeit mit UDP im belasteten LAN

2.3 Synchronisation der Uhren

Anstatt gleichzeitig mit jedem Ultraschallsignal ein Synchronisationssignal zu senden, kann man auch feste absolute Zeiten vorschreiben, zu denen bestimmte Sender senden sollen. Dazu müssen die Uhren der beiden Computer synchron laufen.

Das Network Time Protocol (NTP) ist ein Protokoll und eine Software zur Synchronisierung von Uhren von Computern über paketbasierte Netzwerke mit variabler Verzögerungszeit. NTP verwendet den Algorithmus von Marzullo, um aus mehreren verrauschten Zeitquellen im Internet eine genaue Uhrzeit zu berechnen. Die variablen Verzögerungszeiten werden dabei durch einen Jitter-Buffer ausgeglichen. Es wird nicht nur die aktuelle Uhrzeit der Systemuhr umgestellt, sondern auch der Drift über längere Zeit gemessen und die Geschwindigkeit der Uhr angepasst. Die Datenübertragung erfolgt dabei über das im letzten Kapitel vorgestellte User Datagram Protocol. Über das öffentliche Internet wird so abhängig von der Qualität der Internetverbindung eine Genauigkeit von 5-100ms erzielt [04]. Im lokalen Netzwerk sind unter Laborbedingungen Genauigkeiten von etwas mehr als einer Millisekunde möglich.

Um die tatsächliche Genauigkeit unter den gegebenen Bedingungen zu testen, wurde für diese Arbeit ein Testprogramm entwickelt, das ausgehend von einem Startzeitpunkt alle 100 ms die aktuelle Zeit minus der Summe aus Startzeit und vergangener Zeit aufzeichnet. Wenn NTP ausgeschaltet ist, wird so die Genauigkeit der Zeitmessung deutlich. Mit eingeschaltetem NTP kann gesehen werden, wann und wie stark die Uhr verstellt wird. Die Synchronisation erfolgt dabei im lokalen Netzwerk mit dem Notebook als Server und dem SBC als Client. Die Ergebnisse sind in Abbildung 12 dargestellt. Offensichtlich schwankt die Zeit in einem Intervall von etwa 3 ms. Angesichts der Tatsache, dass ein Ultraschallsignal in 3 ms mehr als einen Meter zurücklegt, ist dies verhältnismäßig viel. Es ist also nicht damit zu rechnen, dass die Implementierung dieses Ansatzes ein besonders gutes Ergebnis liefern wird.

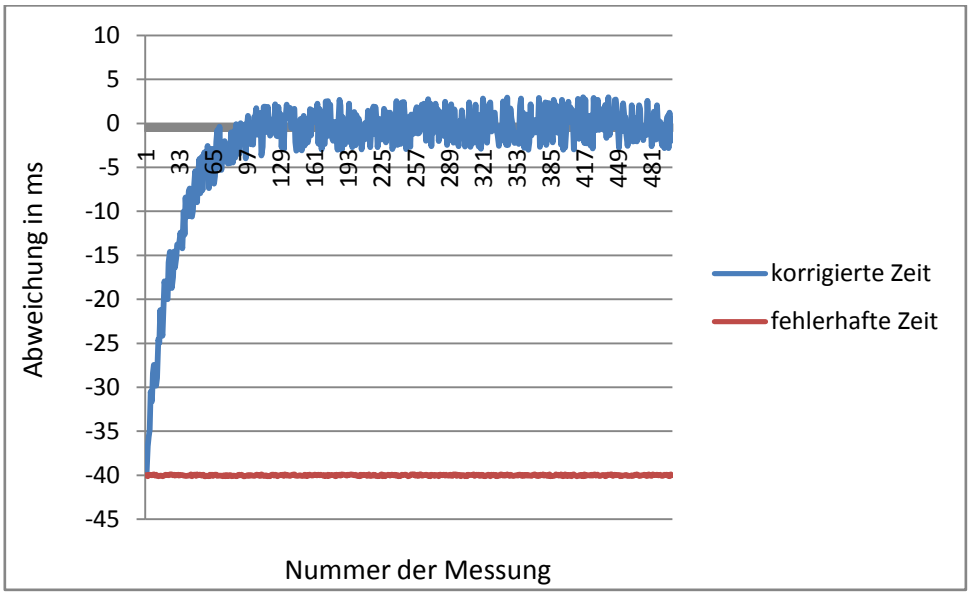


Abbildung 12: Verlauf der Zeitanpassung mit NTP

3 Implementierung und Tests

Um die bisher nur in der Theorie besprochenen Verfahren auf Praxistauglichkeit zu untersuchen, wurden alle Ansätze in C-Programmen implementiert. Dabei dienen jeweils ein Programm auf dem Notebook zum Auslösen der Sender und ein Programm auf dem SBC zum Empfangen der Ultraschallsignale, der Berechnung der jeweiligen Entfernungen und der Bestimmung der absoluten Position in der (x,y)-Ebene.

Die Berechnung wurde mit vier Sendern durchgeführt. Drei sind unbedingt erforderlich, der vierte erhöht die Genauigkeit. Ausgehend von einer geschätzten Startposition wird jedes Mal, wenn alle Entfernungen aktualisiert wurden, die neue Position bestimmt. Dies erfolgt mit der Gauß-Newton-Methode, angewandt auf das nichtlineare Problem der kleinsten Quadrate:

$$\min \frac{1}{2}\left\|F\left(x,y,\delta\right)\right\|_2^2$$

Dabei ist $\left(x,y\right)$ die aktuelle Position und δ der Synchronisationsfehler. F ist gegeben durch:

$$F\left(x,y,\delta\right) := \begin{pmatrix} \vdots \\ \sqrt{\left(x-x_i\right)^2+\left(y-y_i\right)^2+z_i^2}+\delta\cdot c \\ \vdots \end{pmatrix} - \begin{pmatrix} \vdots \\ d_i \\ \vdots \end{pmatrix}$$

Dabei ist $\left(x_i,y_i,z_i\right)$ die bekannte Position des Senders mit Nummer i (siehe Tabelle im Anhang), d_i die gemessene Entfernung zu diesem Sender und $c = 343\,m/s$ die Schallgeschwindigkeit in Luft bei Raumtemperatur. Die Jacobi-Matrix von F lautet:

$$F'\left(x,y,\delta\right) = \begin{pmatrix} \vdots & \vdots & \vdots \\ \dfrac{x-x_i}{\sqrt{\left(x-x_i\right)^2+\left(y-y_i\right)^2+z_i^2}} & \dfrac{y-y_i}{\sqrt{\left(x-x_i\right)^2+\left(y-y_i\right)^2+z_i^2}} & c \\ \vdots & \vdots & \vdots \end{pmatrix}$$

Ein Schritt des Gauß-Newton-Verfahrens ergibt für die aktualisierte Position:

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ \delta_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \\ \delta_k \end{pmatrix} - \left(F'\left(x_k,y_k,\delta_k\right)^T F'\left(x_k,y_k,\delta_k\right)\right)^{-1} F'\left(x_k,y_k,\delta_k\right)^T F\left(x_k,y_k,\delta_k\right)$$

Die Tests in den folgenden Abschnitten fanden in einem Raum von etwa 8 m mal 6 m statt. Ein Punkt in der Mitte wurde als Koordinatenursprung definiert, die Positionen der vier Sender von dort aus gemessen. Der Empfänger wurde stets direkt auf dem definierten Ursprung platziert, die „echte“ Position war also (0,0). Es folgen die Testergebnisse der verschiedenen Synchronisationsverfahren.

3.1 Bluetooth

Die Messergebnisse für die Variante mit Bluetooth-Synchronisationssignal sind in Abbildung 13 und Abbildung 14 dargestellt. Die mittlere X-Position beträgt 7,5 cm, der Median 3,2 cm und die Varianz 8,8 cm. Die mittlere Y-Position beträgt -30,2 cm, der Median -27,7 cm und die Varianz 11,2 cm. Wie erwartet gibt es Peaks und starkes Rauschen. Zur Verbesserung des Ergebnisses kann theoretisch für jeden Sendezyklus nur ein Synchronisationssignal gesendet werden. Dadurch hätten alle gemessenen Entfernungen des Zyklus die gleiche Abweichung, was zu einer höheren Genauigkeit der berechneten Position führen sollte. Allerdings müssten dafür im Empfänger-Programm mehrere Timer angelegt werden, die vom Synchronisationssignal die Zeiten bis zum Sendebeginn der anderen Sender messen. Im Praxistest ergab sich dadurch keine deutliche Verbesserung.

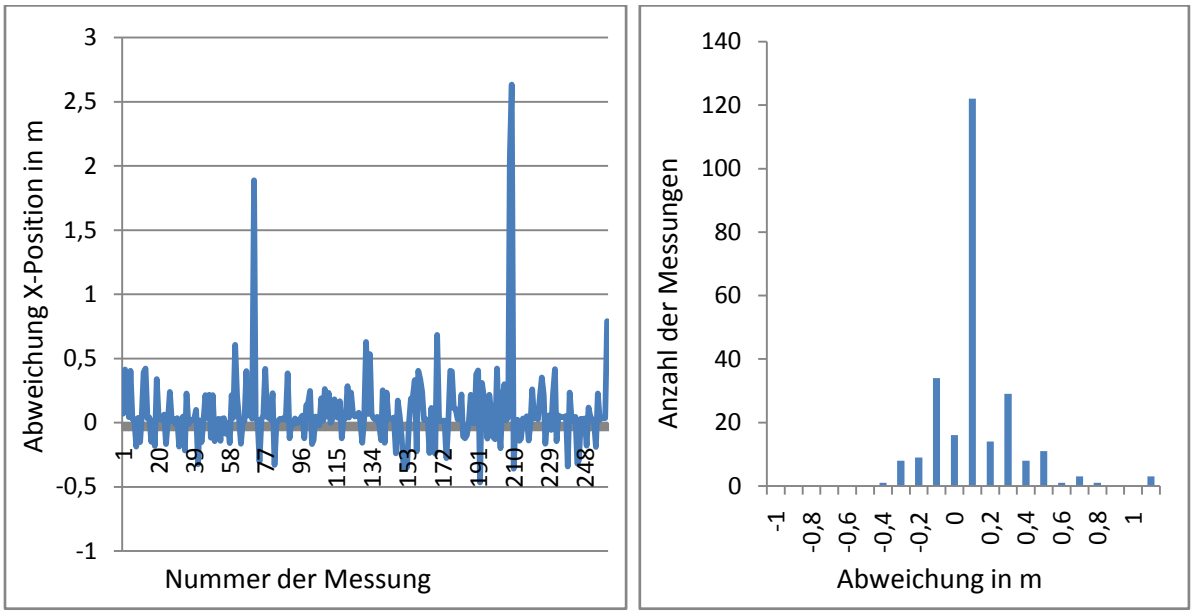


Abbildung 13: Mit Bluetooth bestimmte X-Position und Histogramm

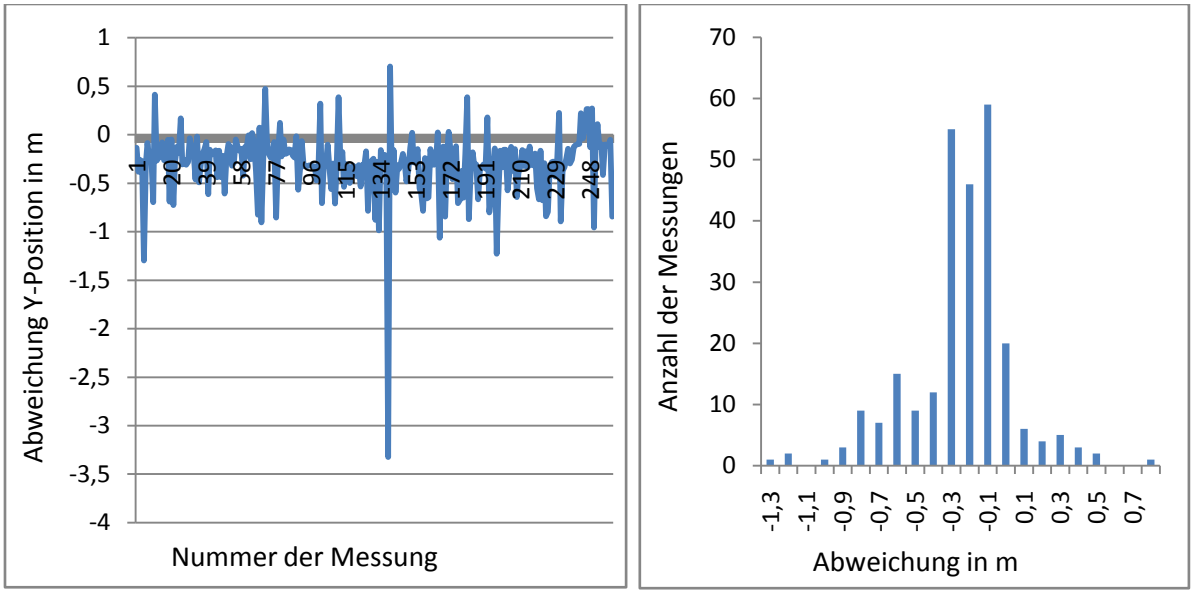


Abbildung 14: Mit Bluetooth bestimmte Y-Position und Histogramm

3.2 Lokales Netzwerk

Für die Variante mit Synchronisationssignalen über das lokale Netzwerk wurden sowohl Programme für TCP als auch für UDP entwickelt. Die Ergebnisse mit UDP sind sehr viel besser, daher werden hier nur diese wiedergegeben (siehe Abbildung 15 und Abbildung 16). Die mittlere X-Position beträgt 6,1 cm, der Median 6,2 cm und die Varianz nur 2,9 mm. Die mittlere Y-Position beträgt 2,4 cm, der Median 2,3 cm und die Varianz 7,1 mm. Wie an der sehr geringen Varianz zu sehen ist, gibt es fast gar kein Rauschen und nur eine konstante Abweichung. Diese liegt innerhalb der Messungenauigkeit, mit der die Position der Sender bestimmt wurde, und lässt sich durch eine Kalibrierung derselben beheben. Die Peaks können auch leicht ausgeglichen werden, indem Messwerte für ungültig erklärt werden, die weit von der letzten Position entfernt sind. Bei der hohen Messfrequenz gilt dies auch für einen bewegten Roboter. Insgesamt liegt die erreichbare Messgenauigkeit bei 1-3 cm. Zu bedenken ist allerdings, dass diese guten Resultate nur in einem unbelasteten Netzwerk realisierbar sind. Im belasteten Netzwerk ergaben sich im Test ähnliche Messwerte wie bei der Variante mit Bluetooth.

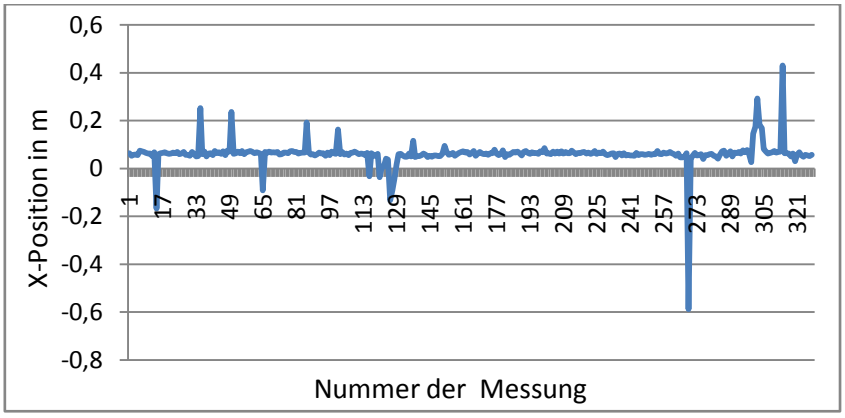


Abbildung 15: X-Position gemessen mit UDP

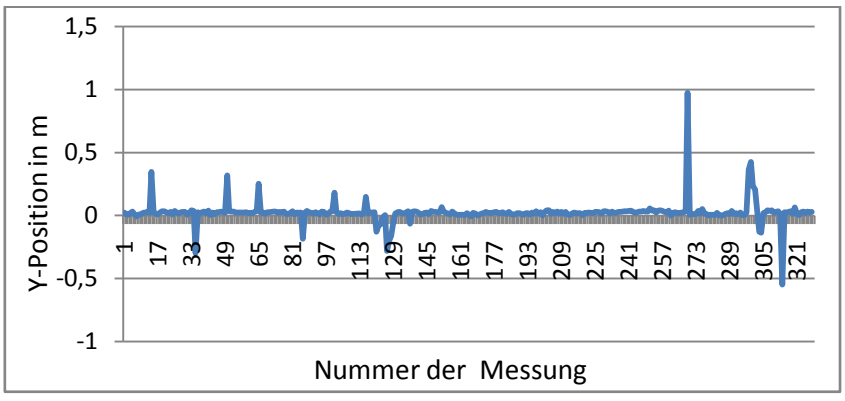


Abbildung 16: Y-Position gemessen mit UDP

Betrachtet man die Messergebnisse in der Ebene (siehe Abbildung 17), zeigt sich, dass meist nur einer der vier Sender ungenau ausgewertet wird. Die bestimmte Entfernung zu diesem Sender ist immer zu klein, da das Synchronisationssignal zu spät ankommt und die Ultraschall-Übertragungszeit zu kurz gemessen wird. Dadurch verschiebt sich die Gesamtposition in Richtung des ungenau gemessenen Senders.

Versuche mit bewegtem Empfänger haben ergeben, dass die Positionsbestimmung dieser Aufgabe noch nicht gewachsen ist. Während die Einzelentfernungen gemessen werden, verschiebt sich der Empfänger, und wenn die vierte Entfernung gemessen wurde, ist die erste schon sehr ungenau. Eine Positionsberechnung nach jedem Signal kann daran nichts ändern. Die Erhöhung der Sendefrequenz führt zu stark ansteigendem Rauschen, das eine genaue Messung unmöglich macht. Abhilfe könnte die Verwendung eines Beschleunigungssensors schaffen. Mit diesem könnten die gemessenen Entfernungen an die aktuelle Richtung und Geschwindigkeit angepasst werden.

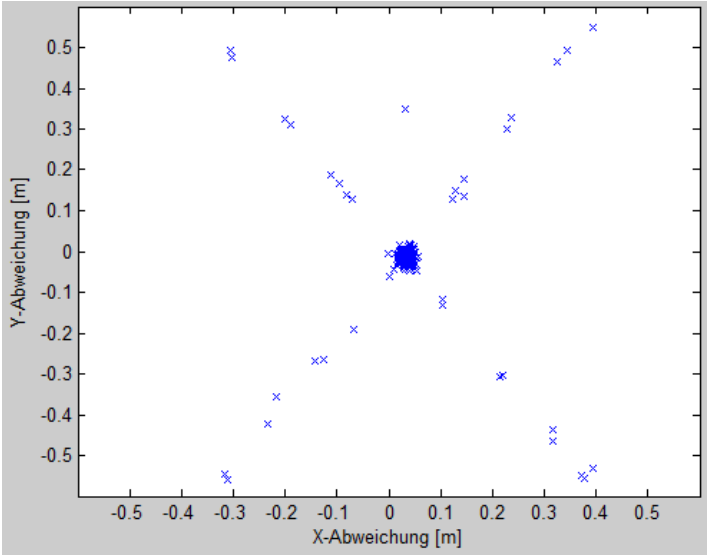


Abbildung 17: Messfehler in der Ebene, Auswertung von 600 Messungen mit UDP

3.3 Synchrone Systemuhren

Die Genauigkeit der Zeitsynchronisation über das Network Time Protocol ist nicht ausreichend, um verwertbare Messwerte zu realisieren. Im Test waren die bestimmten Entfernungen zu den Sendern so ungenau, dass das Gauß-Newton-Verfahren nicht konvergiert ist und die Werte für X- und Y-Position nach wenigen Sekunden gegen unendlich gingen. An diesem Ergebnis kann auch eine Verbesserung des Berechnungsverfahrens nichts ändern. Eine Globalisierungsstrategie, z.B. Schrittweitensuche mit Armijo-Liniensuche, findet zwar eine optimale Schrittweite, aber diese hilft nicht weiter, wenn aufgrund schlechter Messwerte die Richtung falsch ist. Es ist also eine bessere Technologie als das NTP nötig.

Denkbar wäre es, sowohl das Notebook als auch den SBC mit DCF77-Empfängern auszustatten. Der Zeitsignalsender DCF77 ist ein Langwellensender, der die meisten funkgesteuerten Uhren im westlichen Europa mit der genauen Atomzeit versorgt. Bei der üblichen Auswertelektronik von handelsüblichen DCF77 USB-Sticks muss man allerdings mit einer Ungenauigkeit von 5-150 ms rechnen, was deutlich schlechter ist als NTP. Bei größeren Antennen sind höhere Genauigkeiten realisierbar, es erhöht sich aber auch das Rauschen und die Signale können oft nicht mehr ausgewertet werden. Dieser Ansatz ist also wenig erfolgversprechend.

Eine weitere Alternative wäre die Verwendung eines gemeinsamen Signalgebers für Notebook und SBC. Hier würde sich die Pulse Per Second (PPS) API anbieten. Dies ist eine Erweiterung des Kernel Uhrmodells, das es erlaubt, die Systemuhr mit externen Zeitstempeln zu synchronisieren. Eine externe Quelle mit einer Frequenz von einem Puls pro Sekunde ist zum Beispiel ein GPS-Empfänger. Die generierten Zeitstempel haben eine hohe Frequenzstabilität und weniger als 500ns Abweichung zum Beginn der UTC-Sekunde. In Experimenten an der Universität Erlangen [05] wurden mit Hilfe von PPS über seriellen Anschluss Uhren auf weniger als 25 µs Abweichung gebracht. Dies entspricht bei Schallgeschwindigkeit etwa 8,5 mm, die Genauigkeit bei vier Sendern dürfte bei etwa 5 mm liegen.

4 Zusammenfassung

In dieser Arbeit wurden nach einem kurzen Überblick über die verschiedenen Methoden zur Positionsbestimmung in der Robotik mehrere Verfahren vorgestellt, die auf der Multilateration mit Ultraschallsendern basieren. Die Verfahren wurden zunächst theoretisch besprochen und ihre Erfolgsaussichten diskutiert. Anschließend wurden mögliche Implementierungen erörtert und Ergebnisse von Praxistests gegeben.

Die höchste Genauigkeit hat im Praxistest mit einer Abweichung von nur 1-3cm die Positionsbestimmung mit Synchronisationssignalen über das User Datagram Protocol im lokalen Netzwerk. Nachteile dieses Verfahrens sind die Tatsache, dass das Netzwerk ansonsten nicht belastet werden darf, und die Notwendigkeit einer kabelgebundenen Kommunikation, da die Latenz der W-LAN-Verbindung zu hoch ist. Die Methode mit Synchronisationssignalen über Bluetooth hat eine sehr viel geringere Genauigkeit, ist aber kabellos und nicht von der Belastung des Netzwerks abhängig. Bei der Synchronisation der Uhren von Notebook und Single Board Computer war die Genauigkeit des Network Time Protocol nicht ausreichend, um brauchbare Ergebnisse zu erzielen. Bei dieser Methode sind aber die Erfolgsaussichten am höchsten, mit einer besseren Technologie (etwa PPS) noch größere Genauigkeiten zu erreichen.

Literaturangaben

[01] J. Leonard, H. Durrant-Whyte: *Simultaneous Map Building and Localization for an Autonomous Mobile Robot*. Proc. of IEEE Int. Workshop on Intelligent Robots and Systems, pp 1442-1447, 1991

[02] J. Borenstein, H.R. Everett, L. Feng, and D. Wehe: *Mobile Robot Positioning – Sensors and Techniques*. Journal of Robotic Systems, Vol. 14 No. 4 pp. 231-249, 1997

[03] C. Cohen, F. Koss: *A Comprehensive Study of Three Object Triangulation*. Proc. of the 1993 SPIE Conference on Mobile Robots, Nov. 18-20, pp. 95-106, 1992

[04] www.ntp.org/ntpfaq

[05] http://www7.informatik.uni-erlangen.de/~ksjh/research/cluster/timesync/

Anhang: Anmerkungen zur Versuchsdurchführung

Die Versuche wurden alle in einem etwa 6x8 Meter großen Raum durchgeführt. Die Ultraschallsender waren dabei in den Ecken an der Decke befestigt und auf die Raummitte gerichtet.

Position der Sender im Raum (Angaben in Metern):

	s_1	s_2	s_3	s_4
x	$-4,17$	$3,89$	$3,91$	$-4,47$
y	$-2,6$	$-2,65$	$2,67$	$2,67$
z	$2,76$	$2,76$	$2,75$	$2,75$

Im Programm wird die z-Koordinate relativ zu der des Empfängers angegeben. Diese ist zurzeit konstant auf 37cm gesetzt, die Position wird nur in der Ebene bestimmt.

Durchführung einer Positionsbestimmung

Es folgt eine Auflistung aller notwendigen Schritte zur Positionsbestimmung mit Bluetooth und UDP. Die benötigten Programme sind auf CD beigelegt.

1. **Notebook mit Ubuntu Life-CD starten und einrichten**
   - Bei Bootproblemen die Optionen `noapic` und `nomodeset` auswählen
   - Netzwerkeinstellungen auf manuell umstellen und konfigurieren:
     Adresse: `137.193.70.13`
     Netzmaske: `255.255.255.128`
     Gateway: `137.193.70.126`
     DNS-Server: `137.193.6.6`
   - Benötigte Programmdaten ins Arbeitsverzeichnis kopieren:
     Ordner `blue` und `udp` von der CD in das Ubuntu-Verzeichnis kopieren
   - Bibliotheken installieren:
     Terminal: `sudo apt-get install libbluetooth-dev`
2. **SBC einrichten**
   - Benötigte Daten auf den SBC kopieren:
     Im Terminal in das Ubuntu-Verzeichnis des Notebooks wechseln. Unterordner `blue` und `udp` in das `root` Verzeichnis des SBC kopieren:
     ```
     scp –r blue root@137.193.70.12:~/blue
     scp –r udp root@137.193.70.12:~/udp
     ```
     Passwort: `ingmathe`
   - Über Secure Shell auf SBC einloggen:
     Neues Terminal: `ssh 137.193.70.12 -l root`
     Passwort: `ingmathe`
     Der aktuelle Pfad ist nun der `root` Ordner mit den Unterordnern `blue` und `udp`

- Bluetooth im SBC Terminal initialisieren:
  ```
  cd blue/init-bluetooth-sbc2
  bash ./runme.sh
  hcitool dev
  ```
  Bluetooth-Adresse notieren, normalerweise 00:0C:78:78:81:A0

3. **Test mit Bluetooth-Synchronisationssignalen durchführen**
   - Empfänger-Programm auf SBC kompilieren und starten:
     Im SBC Terminal in den Ordner `blue` wechseln, dann:
     ```
     make receiver
     nice –n -15 ./receiver
     ```
   - Sender-Programm auf Notebook kompilieren und starten:
     Im Notebook Terminal in den Ordner `blue` wechseln, dann:
     ```
     make transmitter
     sudo nice –n -15 ./transmitter 00:0C:78:78:81:A0
     ```
   - Positionsdaten werden automatisch über das SBC Terminal ausgegeben und auf dem SBC im Ordner `blue` in der Datei `output.txt` gespeichert
   - Nach Abschluss des Tests Programme beenden:
     Erst im SBC Terminal, dann im Notebook Terminal `Strg+C` drücken
   - Logdatei zur Auswertung auf das Notebook kopieren:
     Im Notebook Terminal: `scp root@137.193.70.12:~/blue/output.txt .`
4. **Test mit UDP-Synchronisationssignalen durchführen**
   - Empfänger-Programm auf SBC kompilieren und starten:
     Im SBC Terminal in den Ordner `udp` wechseln, dann:
     ```
     make receiver
     nice –n -15 ./receiver 2345
     ```
   - Sender-Programm auf Notebook kompilieren und starten:
     Im Notebook Terminal in den Ordner `udp` wechseln, dann:
     ```
     make transmitter
     sudo nice –n -15 ./transmitter 00:0C:78:78:81:A0
     ```
   - Positionsdaten werden automatisch über das SBC Terminal ausgegeben und auf dem SBC im Ordner `blue` in der Datei `output.txt` gespeichert
   - Nach Abschluss des Tests Programme beenden:
     Erst im SBC Terminal, dann im Notebook Terminal `Strg+C` drücken
   - Logdatei zur Auswertung auf das Notebook kopieren:
     NB Terminal: `scp root@137.193.70.12:~/blue/outputUDP.txt .`