1. Aufgabenstellung

Unsere Aufgabenstellung ergab sich aus einem Vorhaben der Firma ISIS IC. ISIS IC installiert komplette Systeme mit aktiver RFID-Technik, um beispielsweise Sensormesswerte per Funk zu überwachen. Diese RFID-Hardware liefert als „Abfallprodukt“ den sogenannten RSSI-Wert, einen Messwert der Feldstärke, mit der ein Tag von einem Empfänger empfangen wird. Die Idee von ISIS IC war nun, dass man diesen Wert theoretisch zur Positionsbestimmung nutzen können müsste, da er abhängig vom Aufenthaltsort des Tags ist. Aus dieser Idee heraus ergab sich unsere Aufgabenstellung. Wir bekamen die Aufgabe, einen Prototyp einer Software für eine Positionsbestimmung mittels RSSI-Werten zu entwickeln und mit diesem auch erste Versuche mit der RFID-Hardware durchzuführen, um die grundsätzliche Realisierbarkeit der Idee zu prüfen. Für die Entwicklung dieses Prototypen hatte ISIS IC folgende Voraussetzungen:

- Der Datenaustausch soll über eine bereits existente MySQL-Datenbank ablaufen; zur Hardware-Anbindung wird bereits existente Software genutzt
- Die Software soll sich in eine Webapplikation integrieren lassen
- Die Software soll eine 2D-Positionsbestimmung ermöglichen und dabei mehrere 2D-Ebenen unterstützen
- Die Position eines Objekts soll grafisch auf einer im System hinterlegbaren Karte in Form eines ebenfalls frei wählbaren Icons dargestellt werden
- Es sollte sich auf Wunsch auch der Bewegungspfad eines Objekts verfolgen lassen
- Die Software soll sich an verschiedene Räumlichkeiten anpassen lassen

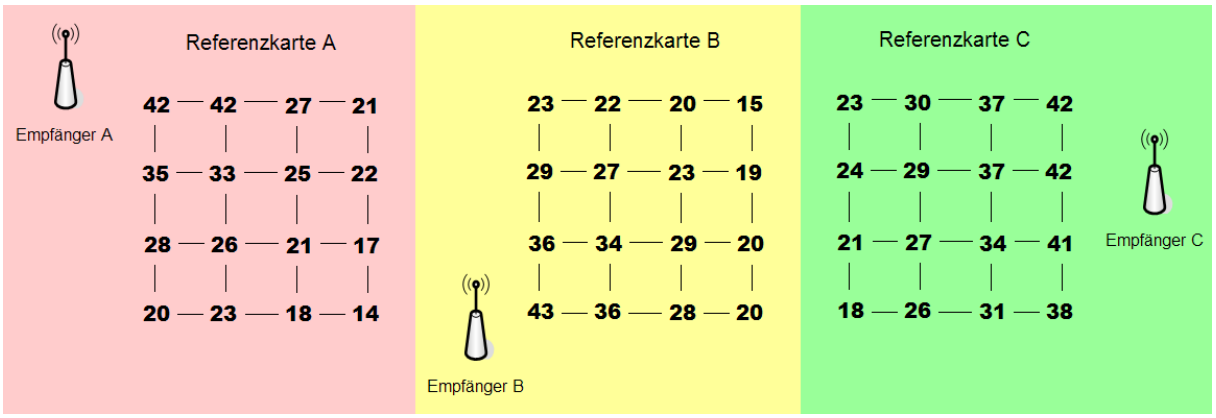
Aus diesen Voraussetzungen und unseren eigenen Erwartungen an das Projekt ergaben sich schließlich folgende Randbedingungen für unsere Arbeit:

- Die Software wird in Form zweier Java-Applets realisiert, welches direkt auf die Datenbank zugreift: eines für die eigentliche Positionsbestimmung, eines zur Administration und Kalibrierung
- Zur Positionsbestimmung werden wir ein Gitter von Referenzpunkten heranziehen, welche einmalig ausgemessen werden müssen (Kalibriervorgang)
- Die Positionsbestimmung wird auf einer 2D-Ebene durchgeführt, von denen mehrere im System erstellbar sein werden. Jeder Ebene werden individuell mehrere RFID-Empfänger zugeordnet
- Die Position eines Tags wird grafisch auf einer im System hinterlegten Karte (Bilddatei) in Form eines für jeden Tag frei wählbaren Icons angezeigt
- Der Bewegungspfad eines Objekts lässt sich zusätzlich einblenden
- Die Details der Hardware-Anbindung sind für unsere Arbeit nicht relevant, da dafür bereits Software existiert

3. Logik

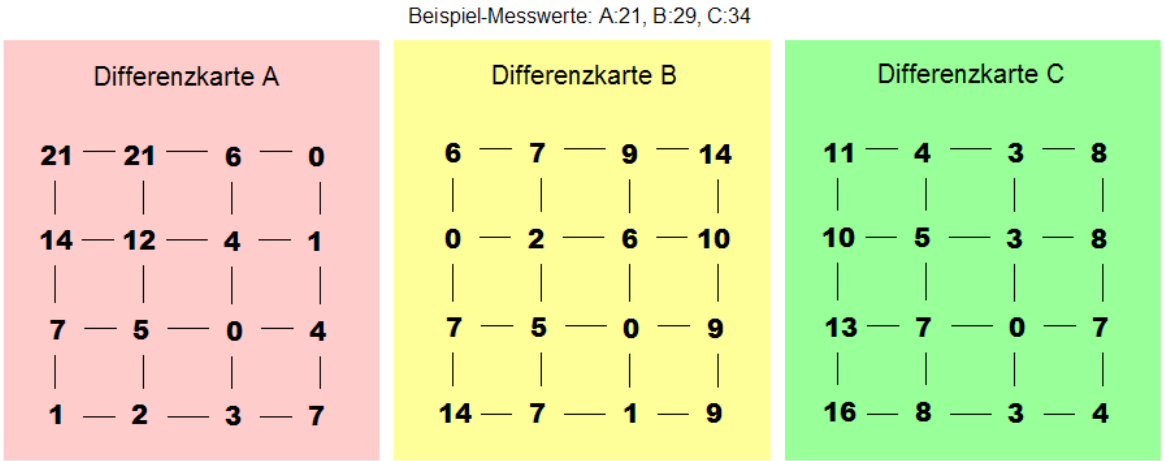
3.1 Die Positionsbestimmung in der Theorie

Wir nutzen für die eigentliche Positionsbestimmung ein Gitter aus Referenzpunkten. Für jeden Referenzpunkt ist ein Satz RSSI-Referenzwerte gespeichert, in welchem jeder dort empfangene Funkempfänger mit einem Wert vertreten ist. Man kann diese eine große Referenzpunktekarte also auch wie mehrere Karten mit je einem Referenzwert auffassen: eine Karte pro Funkempfänger.



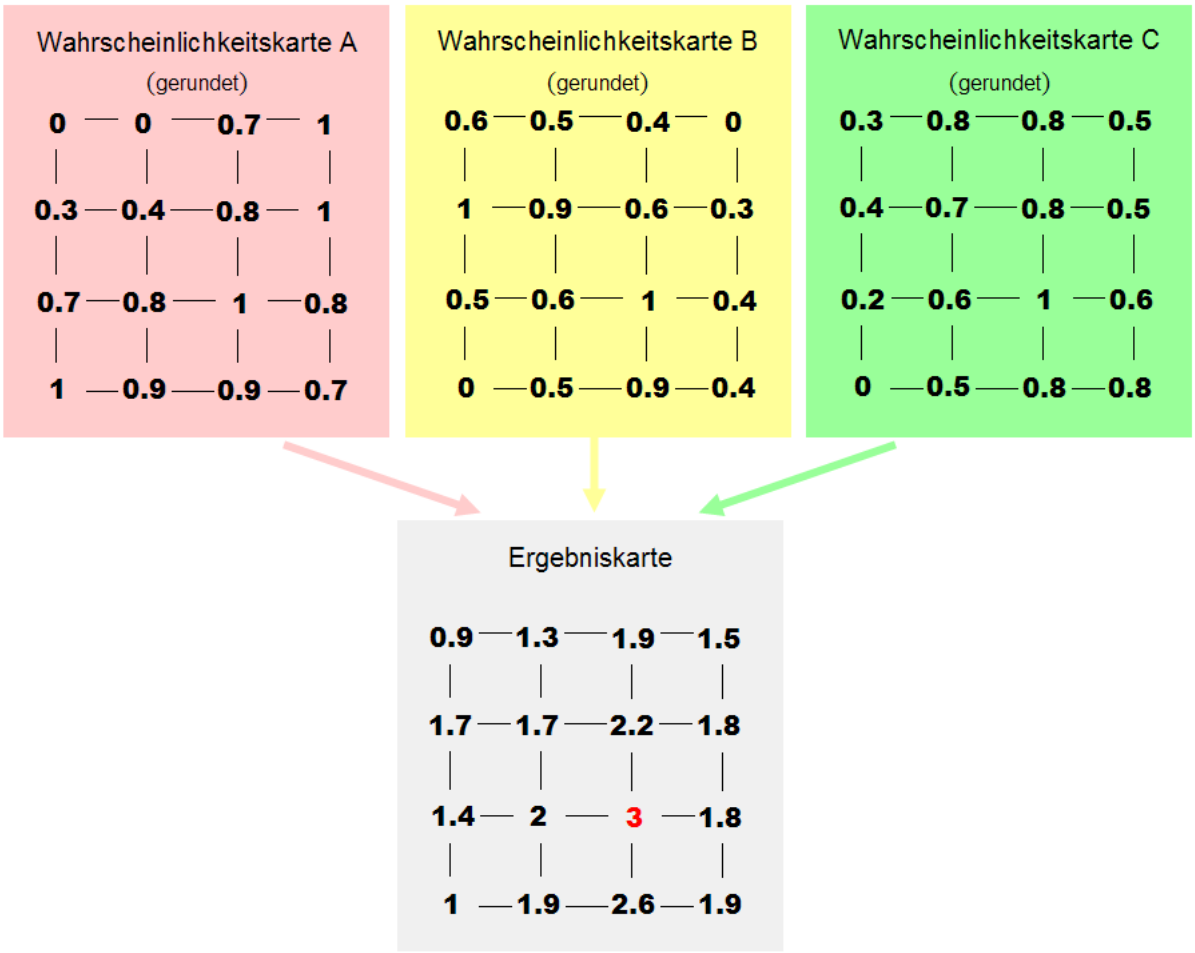
Wird nun ein aktueller Messwertesatz (bestehend aus Messwerten aller Funkempfänger zur selben Zeit) herangezogen, um mit Hilfe der gespeicherten Referenzwerte die wahrscheinlichste Position des sendenden Tags zu bestimmen, so läuft folgender Algorithmus ab:

1. Zunächst werden – sollten mehr als drei Empfänger im Messwertesatz enthalten sein – die drei stärksten Messwerte bzw. Empfänger herausgefiltert.
2. Für diese drei Messwerte wird anschließend je eine Differenzkarte erstellt. Die Differenzkarte hat dieselbe Breite und Höhe wie die Referenzkarten, ihre Werte ergeben sich dabei aus der Differenz des jeweiligen Werts der Referenzkarte zum aktuell gemessenen Wert. Die Differenzwerte sind dabei immer positiv, von negativen Ergebnissen wird der Betrag genommen.



3. Aus diesen Differenzkarten wird anschließend durch Normalisieren (d.h. die Werte der Karte werden mit einem Faktor multipliziert, so dass der größte vorkommende

Wert exakt 1.0 entspricht) und Invertieren (neuer Wert = 1 - alter Wert) je eine Wahrscheinlichkeitskarte erstellt, deren Werte für jeden Punkt die Wahrscheinlichkeit angeben, dass sich das Tag dort befunden hat.



4. Die drei Wahrscheinlichkeitskarten werden addiert, in der Ergebniskarte repräsentiert der Punkt mit dem höchsten Wahrscheinlichkeitswert die wahrscheinlichste Position des Tags.

3.2 Die Positionsbestimmung im Java-Applet

Die Umsetzung dieses theoretischen Algorithmus in Java ist hauptsächlich mit Hilfe einer Klasse (der sogenannten WeightMap) gelöst. Instanzen dieser Klasse repräsentieren eine Differenzkarte/Wahrscheinlichkeitskarte/Ergebniskarte und besitzen Methoden, die den oben beschriebenen Algorithmus implementieren. Die Referenzdaten sind im Grid-Objekt gespeichert, welches jede Ebene besitzt, und werden zur initialen Erstellung der Differenzkarten herangezogen.

3.3 Die Auflösung

Der genutzte Algorithmus beschränkt die Auflösung der Positionsbestimmung auf die Auflösung des Referenzgitters. Ein offensichtlicher Weg, die Auflösung zu erhöhen, wäre somit die Erhöhung der Anzahl der Referenzpunkte. Dies führt allerdings zu stark ansteigendem Aufwand bei der Ermittlung der Referenzwerte.

Zur Erhöhung der Auflösung ohne zusätzlichen Aufwand bei der Ermittlung der Referenzwerte wurde eine Interpolationsroutine zur Erhöhung der Referenzgitterauflösung implementiert. Der Interpolationsfaktor ist (derzeit allerdings nur „hard-coded“ im Java-Quellcode) frei anpassbar.
Bei unseren Tests hat sich allerdings herausgestellt, dass eine Erhöhung des Interpolationsfaktors schon sehr früh keinen Mehrnutzen mehr bringt (> Faktor 4), sondern lediglich die zur Positionsbestimmung nötige Rechenleistung in die Höhe treibt. Grund dafür ist die mangelnde Genauigkeit der von der Hardware gelieferten Werte (näheres hierzu in 5.2.2).

5. Probleme und Erkenntnisse

5.1 Probleme bei der Entwicklung der Software

5.1.1 Die Umstellung auf Hibernate

Nachdem die Datenbankschicht unserer Anwendung bereits unter Nutzung von JDBC implementiert und lauffähig war, entschieden wir uns dazu, eine Version der Schicht zu entwickeln, die Hibernate als Persistenzframework zum Zugriff auf die Datenbank nutzt. Diese Entscheidung wurde hauptsächlich getroffen, weil wir uns vom Einsatz von Hibernate in einer praktischen Anwendung einen hohen Lerneffekt versprachen. Die neue Datenbankschicht sollte nach außen dieselben Schnittstellen aufweisen wie die alte, um auf lange Sicht die alte Schicht möglichst einfach gegen die neue austauschen zu können. Dieser Teil des Vorhabens erwies sich als unproblematisch, anders sah es aber bei der Einbindung von Hibernate in die neue Datenbankschicht aus. Dadurch, dass wir alle keinerlei Erfahrung im Umgang mit einem komplexen Persistenzframework wie Hibernate hatten, stellte sich die Einarbeitung in das Framework als sehr zeitraubend heraus. Große Teile der Zeit, die die Umstellung auf Hibernate insgesamt kostete, flossen in zahllose Änderungen an bestehenden Klassen, um diese „kompatibel“ zu Hibernate zu machen.
Unsere Erkenntnis aus dieser Umstellung ist daher, dass wir einen Großteil dieser Änderungen hätten vermeiden können, wenn wir direkt von Anfang an die Nutzung von Hibernate als festen Projektbestandteil eingeplant und unsere Klassen und Objekte von vorneherein entsprechend konstruiert hätten. Durch die investierte Zeit in die Umstellung ist aber eines der Hauptziele dieses Vorhabens, nämlich der Lerneffekt, definitiv erreicht worden.

5.2 Probleme mit der Hardware

5.2.1 Unterschiede zwischen der Sensite- und der Coronis-Hardware

Zu Projektbeginn stellte uns die Firma ISIS IC die bei ihnen bereits bewährte Hardware von der Firma Sensite Solutions vor. Dabei handelt es sich um Funkempfänger, die von aktiven Tags empfangene Datenpakete untereinander weiterleiten können, bis diese Daten schließlich über eine serielle Schnittstelle und eine bereits bestehende Software in die Datenbank gelangen. Bei dieser Hardware sendeten die Tags in vorbestimmten Intervallen, und bei jedem Sendevorgang entstanden an jedem Empfänger automatisch die von uns benötigten RSSI-Werte.
Zu unseren Tests erhielten wir allerdings eine neue Hardware von einem anderen Hersteller, der Firma Coronis. Wir erfuhren allerdings erst spät, welch großer Unterschied zwischen den auf den ersten Blick ähnlichen Hardwaretypen liegt. Bei der Coronis-Hardware ist jede im Funknetzwerk aktive Komponente („Tags“ in diesem Sinne gibt es bei dieser Hardware gar nicht) gleichzeitig Sender und Empfänger. Die Funkkomponenten kommunizieren mit einem bidirektionalen Protokoll untereinander. Die sich für uns ergebenden Probleme aus diesen Unterschieden waren im Einzelnen:

- Es existierte keinerlei Software für die Anbindung der Coronis-Hardware an die Datenbank
- Die von uns benötigten RSSI-Werte entstanden nicht mehr als „Abfallprodukt“, sondern mussten gezielt erzeugt werden

- Die Konfiguration der Coronis-Hardware stellte uns zu Beginn mangels Kenntnissen über die Funktionsweise der Hardware vor Probleme

Die Lösung dieser Probleme erfolgte in mehreren Schritten. Zunächst vereinbarten wir in einem klärenden Gespräch mit der Firma ISIS IC einen Termin, um gemeinsam Erfahrung mit der Konfiguration der Coronis-Hardware zu sammeln. Bei dieser Gelegenheit haben wir auch gemeinsam erste Tests bezüglich der Konstanz der empfangenen RSSI-Werte im HdM-Gebäude gemacht – sowohl mit der Sensite- als auch der Coronis-Hardware. Dabei stellte sich heraus, dass die Sensite-Hardware an bestimmten Orten unerklärliche, aber sich drastisch auswirkende und nicht vorhersagbare Sprünge im gemessenen RSSI-Wert zeigt. Dieses Verhalten konnten wir bei der Coronis-Hardware nicht feststellen, was uns zu dem Schluss brachte, dass die Sensite-Hardware nicht für unsere Zwecke brauchbar ist.
An diesem Tag vereinbarten wir auch das weitere Vorgehen, um das zweite und dritte Problem anzugehen: die fehlende Software sowie die Problematik, RSSI-Werte zu erzeugen. Wir einigten uns auf ein zweigleisiges Vorgehen: auf der einen Seite entwickelten wir eine Windows-Software in C++, die direkt mit der Hardware kommuniziert und ein Notebook mit WLAN-Anschluss und einem Coronis-Sender/Empfänger zu einer Art „provisorischem Tag“ macht. Diese Software (genannt „RSSIExplorer“) sendet dazu periodisch einen Broadcast an alle in Reichweite befindlichen Empfänger, der diese dazu veranlasst, mit dem Feldstärkewert zu antworten, mit dem der Sender am Notebook bei ihnen zu empfangen ist. Die Antworten wertet der RSSIExplorer aus, zeigt die RSSI-Werte an und schreibt auf Wunsch entsprechende Datensätze mit den Werten in die Datenbank.
Auf der anderen Seite startete ISIS IC die kurzfristige Entwicklung eines „Tags“ auf Basis eines Mikrocontrollers und eines Coronis-Sende/Empfangsmoduls. Dieses Tag kann so konfiguriert werden, dass es periodisch eine Liste gespeicherter Empfänger nach den RSSI-Werten fragt und die Ergebnisse an einen Empfänger schickt, der diese Werte an einen Rechner weiterreicht. Nachdem wir dieses neue Tag bekommen hatten, passten wir den RSSIExplorer dahingehend an, als dass dieser nun auch auf die Nachrichten dieses speziellen Tags hört, daraus die RSSI-Werte extrahiert und diese wie gehabt in der Datenbank ablegt. Damit waren beide Probleme so weit gelöst, dass wir erste ernsthafte Tests mit der Coronis-Hardware und unserer Software durchführen konnten.

5.2.2 Die mangelnde Genauigkeit der Hardware

Leider stellte sich in unseren Tests heraus, dass die für uns momentan verfügbare Hardware den Ansprüchen zur Positionsbestimmung nicht wirklich genügt. Die RSSI-Werte schwanken teilweise bereits ohne Veränderung der Position, und Abschattung der Funksignale durch menschliche Körper beispielsweise ruft sehr deutliche Schwankungen hervor. Insgesamt lässt die Robustheit der Messwerte gegenüber Umgebungseinflüssen stark zu wünschen übrig, so dass sich nur eine relativ ungenaue Position (je nach Umgebung bis auf max. 10m genau) bestimmen lässt, deren Richtigkeit auch massiv von einer sauberen und korrekten Kalibrierung abhängt.
Mögliche Lösungsansätze für dieses Problem reichen von softwareseitigen Optimierungen wie einer Mittelung mehrerer Messwerte für jeden Punkt bei der Kalibrierung statt der Speicherung eines einzelnen Messwertes bis zur Nutzung vollständig anderer Hardware. Inwieweit dieses Problem aber wirklich lösbar ist, ist zweifelhaft, da Messungen der Feldstärke aus physikalischen Gründen immer eine relativ hohe Anfälligkeit gegen Umgebungseinflüsse haben. Deutlich stabilere Werte dürfte erst eine Laufzeitmessung liefern, für die aber spezialisierte und teure Hardware vonnöten ist.