UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

Arduino basierte Sensorknoten für Fließgewässermonitoring

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science (B.Sc.)
im Studiengang Informatik

vorgelegt von
Sergei Diez

Erstgutachter:	Prof. Dr. H. Frey (Institut für Informatik)
Zweitgutachter:	Dr. F. Neumann (Institut für Informatik)

Koblenz, im Januar 2017

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

Zusammenfassung

In dieser Arbeit wird untersucht, ob man einen Hardwareprototyp für Adhoc Netze auf Basis von Arduino erstellen kann, der für die Gewässerüberwachung geeignet ist. Ziel der Prototypentwicklung ist einen Sensorknoten mit modularem Aufbau zu entwickeln, der die Möglichkeit bietet Komponenten leicht auszutauschen. Zusätzlich sind bei diesem Einsatzgebiet einige Anforderungen an den Sensorknoten gestellt, die erfüllt werden müssen. Diese Anforderungen leiten sich von dem Tmote Sky Sensorknoten ab, somit soll der hier neu erstellte Sensorknoten eine Alternative zu diesem darstellen und alle seine Funktionen erfüllen. Dazu werden in dieser Arbeit verschiede erhältliche Arduino Mikrokontroller Versionen auf ihre Tauglichkeit zu einem Sensorknoten überprüft. In der weiteren Arbeit wird der Aufbau der Prototypen dokumentiert. Hierbei werden die verwendete Hardware und ihre Kosten veranschaulicht. Der Folgende erstelle Prototyp ermöglicht es, durch leicht austauschbare Funkmodule, Daten über die drei Funkfrequenzen von $433\,\mathrm{MHz}$, $866\,\mathrm{MHz}$ und $2{,}40\,\mathrm{GHz}$ zu verschicken. Zum Abschluss der Arbeit wird der Prototyp einem Experiment unterzogen, die seine Tauglichkeit zur Gewässerüberwachung auf die Probe stellen. Dazu wurden Messungen auf Boden und auf dem Wasser durchgeführt und ausgewertet. Am Ende konnte der Prototyp fast alle gestellten Anforderungen erfüllen, nur die Kosten waren etwas zu hoch.

Abstract

This Work analyzes if a hardware prototype on an Arduino basis for an Adhoc Network can be created. The objective of the prototype development is, the creation of a sensor node with a modular design, where components can be easily changed. Furthermore the application area has requirements, which the node must fulfill. These requirements are derived from the Tmote Sky sensor node, therefore the new created sensor node must be a possible alternative for it and fulfill the same functions. For that purpose this study reviews some available Arduino microprocessors on their suitability for a sensor node. Later in the work the composition of the sensor node is documented. For this, the hardware and their costs are illustrated. The created hardware prototype allows, through easily changed radio modules, the covering of $433\,\mathrm{MHz}$, $866\,\mathrm{MHz}$ and $2{,}40\,\mathrm{GHz}$ radio frequencies. At the end of the work, the sensor node prototype is used in an experiment to check for the suitability for water monitoring. For this, an experiment was performed on land and on water and the results evaluated. In the end the prototype fulfilled most of the requirements, but the cost was a little too high.

1 Einleitung

Um den gesunden Zustand dieser unserer Fließgewässer zu gewährleisten, ist eine kontinuierliche Beobachtung deren Zustände nötig. Allgemein kann das Gewässermonitoring als geplanter Prozess der Probenahme, Messung und Auswertung verschiedener Wassercharakteristika definiert werden. Hierbei spielt die Wassertemperatur eine wichtige Rolle und die Erhebung von Wassertemperaturen ist wichtig um zum Beispiel Grenzwerte wie Temperaturen von mehr als 25 °C, die kritisch für Fische sind, zu erfassen und darauf zu reagieren. Eine Möglichkeit die Wassertemperatur zu erheben, sind punktuelle Messungen durchzuführen, dies geschieht meistens durch Messstationen und Feldmessungen [1].

Die Messungen der Wasserqualität, die unmittelbar vor Ort durchgeführt werden, heißen In-Situ-Messungen und werden mit zumeist präzisen Temperaturfühlern durchgeführt. Die fest installierten Messstationen sammeln dabei Daten über einen längeren Zeitraum an einer festen Stelle, welches sie zur Überwachung einer größeren Fläche ungeeignet macht. Die Erhebung durch Feldmessungen ist dabei für eine größere Fläche sehr aufwendig. Um diese Probleme anzugehen, wird ein anderes System zum Messen benötigt. Eine Möglichkeit dieses Problem anzugehen wurde im Rahmen einer Masterthesis aus dem Bereich der Gewässerökologie [2] erkundet. Diese schlägt vor, dass durch den Einsatz von Sensornetzen Ortsmessungen in Fließgewässer einfacher durchgeführt werden können.

Sensornetze oder drahtlose Ad-hoc-Netze sind Funknetze, bei dem sich zwei oder mehr Geräte zu einem vermaschten Netzwerk verbinden, was auch als Meshnetzwerk bezeichnet wird. Hierbei werden Daten von Sensorknoten zu Sensorknoten über Netzwerkprotokolle weitergeleitet. Die Sensorknoten haben zumeist die Aufgabe Daten zu erheben und über drahtlose Übertragung an einen Empfänger zu übermitteln. Der Empfänger ist oft mit einem Computer verbunden, wo die erhobenen Daten weiter ausgewertet werden können. Ein Sensorknoten besteht im Grunde aus den gleichen Komponenten wie ein Computer. Er hat einen Prozessor und einen Datenspeicher, dazu kommen eine Stromquelle, Sensoren und ein Modul zur Funkkommunikation. Zusätzlich ist die Hardware zumeist auf Energieeffizienz optimiert.

In dem vorgesehenen Konzept in [2] sollen Sensorknoten in einem Gewässer Daten sammeln und die ermittelten Daten über drahtlose Kommunikation an einem Empfänger übermitteln. Diese Empfänger werden an fest bestimmten Positionen am Ufer aufgestellt und speichern die übermittelten Daten der Sensorknoten zur weiteren Verarbeitung. Ziel dieses Konzepts ist es ein Gewässer großflächig und effizient abzudecken.

Die Masterarbeit in [3] griff dieses Konzept auf und zeigte, dass das Konzept drahtloser Sensornetzwerke für die Erhebung von Temperaturen in Fließgewässern grundsätzlich geeignet ist, jedoch der Kommuni-

kationsradius der verwendeten Sensorknoten (Tmote Sky) auf dem Wasser zu klein für die Kommunikation in einem freischwimmenden Sensornetzwerk ist. Aufgrund der Limitierung durch die Tmote Sky Hardware lässt sich dieses Problem nicht einfach lösen. In einigen wissenschaftlichen Arbeiten wurden Sensorknoten auf der Basis von „*Arduino*“ verwendet [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. Arduino ist ein modular aufgebauter Mikroprozessor und somit lassen sich Komponenten leicht austauschen, womit man auf verschiedene Anforderungen Szenarien einfach reagieren kann.

In dieser Arbeit wird eine kostengünstigere Alternative für einen Tmote Sky Sensorknoten zur Gewässermonitoring gesucht. Dazu wurde ein drahtloser Sensorknoten Prototyp auf Arduino Basis erstellt. Dabei kann der Sensorknoten seine Position bestimmen, das Funkmodul ist einfach auszutauschen, Messinstrumente (Thermometer) sind einfach anzuschließen und der Knoten ist mit der Grundausstattung preislich vergleichbar, mit dem Tmote Sky Sensorknoten. Zusätzlich wurde dieser neue Sensorknoten im Feld auf die Eignung zur Gewässermonitoring getestet. Es wurden dafür zwei Bodenmessungen und eine Wassermessung durchgeführt. Dabei waren drei verschiedene Funkfrequenzen eingesetzt worden, die Frequenzen von $433\,\mathrm{MHz}$, $866\,\mathrm{MHz}$ und $2{,}40\,\mathrm{GHz}$. Während des Versuchs wurden zuerst kleine Datensätze und anschließend große Datenpakete übertragen. Diese Daten sind gefüllt mit den aufgenommen Messdaten der angeschlossenen Messinstrumenten. Die gemessenen Daten werden dabei im Sensorknoten selbst und an dem am Empfänger hängenden PC protokolliert. Am Ende der Arbeit konnte der Prototyp fast alle an ihm gestellten Anforderungen erfüllen, nur die Kosten waren mit 170 € etwas zu hoch.

Die Arbeit gliedert sich wie folgt: In Kapitel 2 werden verschiedene verwandte Arbeiten mit einem Sensorknoten auf Arduino Basis betrachtet. Kapitel 3 beschäftigt sich mit der Hardware des Sensorknotens. Es gibt eine Übersicht über Arduino selbst, Arduino Shields und weiterer benötigter Hardware. Über dem Aufbau des Empfängers, des Sensorknoten Prototypen und dessen Anforderung wird im Kapitel 4 informiert. In Kapitel 5 wird der endgültige Sensorprototyp im Feld getestet. Es werden Messungen mit verschiedenen Funkfrequenzen auf Land und Wasser durchgeführt und die erworbenen Ergebnisse interpretiert.

2 Verwandte Arbeiten

Der Arduino Mikroprozessor Board wird in der Forschung zu vielen verschiedenen Zwecken eingesetzt. Einige Veröffentlichungen nutzten den Arduino Mikroprozessor als Basis für ein Wireless Sensor Network. Oft wird dieser als Sensorknoten benutzt, der über Funk mit anderen Knoten kommuniziert. Diese eingesetzten Sensorknoten werden im folgenden Abschnitt genauer betrachtet.

Piyare und Lee [4] beschäftigen sich mit der Integration von Cloud zu Wireless Sensor Networks, für Sensor Daten Erfassung und das Teilen sowie Nutzen von REST basierenden Web Services. Dabei dient ein Arduino UNO, mit weiterer Hardware, als Basisstation für den dazugehörigen Sensorknoten, der Daten erfasst und weiter leitet [4].

Urdiain, Romero, Doggen, Dams und Houtven [5] nutzen ein Seeeduino, ein Arduino kompatibles Board, als Sensorknoten zur Überwachung von Parkplätzen in einem WSN. Diese erkennen periodisch, ob ein Auto einen Parkplatz nutzt, und senden diese Informationen über Funk an einen Empfänger [5].

Valente, Sanz, Barrientos, Cerro, Ribeiro und Rossi [6] nutzten ein System aus WSN und einem Flugzeugroboter, um in Echtzeit Frost Überwachung in einem Weingut zu betreiben. Dabei sammeln Sensorknoten Daten über Boden, Wetter und Status der Ernte. Die genutzten Sensoren „TelosB“ stammen hierbei vom Hersteller Crossbow. Der Flugzeugroboter ist zuständig für die Überbrückung des WSN zu der Basis Station. Dabei ist er mit einem Arduino basierenden Sensorknoten ausgestattet um die Kommunikation mit dem WSN zu ermöglichen [6].

Die Wissenschaftler in der Universität Sains Malaysia beschäftigen sich mit der Entwicklung eines WSN zur Überwachung der globalen Erderwärmung. In der Arbeit [7] basiert das System auf der Wireless ZigBee Technologie und nutzt ein Arduino Uno als Mikroprozessor um Umweltparameter zumessen. Dabei werden diese Umweltdaten, wie Carbon Dioxide, Oxygen und Temperatur, temporär im Arduino gespeichert und über Zigbee an eine Basisstation übertragen [7].

Mitilineos, Kyriazanos, Segou, Goufas und Thomopoulos [8] untersuchten WSN basierte Lokalisierung, durch Implementierung, Verwendung und Evaluation von verschiedenen Lokalisierungssystemen. Darunter befand sich auch das WSN basierende WAX System, ein Hausintern entwickeltes System mit einem Sensorknoten auf Basis von dem Arduino Mikroprozessor und einem XBee (ZigBee fähig) Radio Modul [8].

Boonsawat, Ekchamanonta, Bumrungkhet und Kittipiyakul [9] präsentierten ein WSN Prototyp System für die Temperatur Überwachung in einem Gebäude. Das Netzwerk besteht aus einem Koordinator, der über Funk Daten von jedem WSN Temperaturüberwachungsknoten sammelt. Alle WSN Sensorknoten bestehen dabei aus einem Arduino Mikrokontroller und ei-

nem XBee Kommunikation Modul, basierend auf dem IEEE 802.15.4/Zigbee Standard [9].

Tatsiopoulos und Ktena [10] entwickelten und implementierten einen Temperatur-, Luftfeuchtigkeits- und Lichtmesser, durch Verwendung von offenen Technologie Standards und kommerziellen Komponenten, um Umweltdaten sowohl im Inneren als auch draußen zu überwachen. Am Herzen dieses entwickelten Prototyps liegt ein Arduino Mikroprozessor, an dem die verschiedenen Sensoren liegen und ein XBee Modul zur Kommunikation [10].

Yawut und Sathapath Kilaso [11] beschäftigten sich mit drahtlosen Sensornetzwerken für Wetter- und Katastrophenalarm. Ihr entwickeltes System ist in zwei Haupteile unterteilt, in einen Sensorknoten und einen Koordinator. Der Sensorknoten besteht dabei aus einem Arduino Mikrokontroller, Licht, Druck, Temperatur und Luftfeuchtigkeit Sensoren und einem XBee Modul zur Kommunikation [11].

Im Feld der menschlichen Gesundheit ist das Sammeln von Daten in Echtzeit vital. Mehrere wissenschaftliche Arbeiten beschäftigen sich hier mit dem Einsatz von WSN in der Medizin [12, 13, 14]. Dabei werden die drahtlosen Netzwerke oft verwendet, um Patienten zu überwachen. Zum Einsatz kommen auch hier Sensorknoten auf Arduino Basis vor.

Zulkifli, Harun und Azahar [12] beschäftigten sich hauptsächlich mit der Überwachung der Zustände von Medizinischen Patienten durch WSN. Hier werden Daten von multiplen Pulsoximetern in einem Ardiuno Nano gespeichert und verarbeitet. Danach werden diese Daten über XBee an eine Zentrale Einheit versendet, wo diese Informationen von multiplen Pulsoximetern simultan angezeigt werden [12].

Die Arbeit [13] beschäftigt sich mit Thema „*Ubiquitous healthcare*“, welches sich von „*Ubiquitous Computing*“ ableiten lässt. Beim Ubiquitous Computing sind einzelne Nutzer von einer Menge Informationsgeräten umgeben, mit denen Sie in ihrem Alltagsleben interagieren und die Sie dabei unterstützen. Ubiquitous healthcare ist dabei ein neu entstehendes Umfeld in der Gesundheitspflege. Dazu wurde eine Gesundheitsüberwachungs Anwendung für ubiquitäre WSN entwickelt und implementiert. Die Datensammlung geschieht hier über Pulssensoren, welche Arduino Boards nutzen um die Daten über Funkmodule zusenden [13].

Kioumars und Tang [14] arbeiteten an einem System, welches die Herzrate und Körpertemperatur von Menschen überwachen kann. Die Daten wurden durch Sensoren von einer Gruppe Freiwilligen gesammelt, diese Sensoren sind von dem Forschungsteam zum Testen des Systems entwickelt worden. Das Herz des Systems besteht aus dem Arduino Mikrokontroller, Hardware und Software, einem Temperatur Sensor, einem Herzfrequenzsensor und einem XBee Radio Modul [14].

3 Hardware

Drahtlose Ad-hoc oder Sensornetze sind Netzwerke, die häufig als vermaschte Netze oder Meshnetzwerke bezeichnet werden. Dabei verbinden sich zwei oder mehr Endgeräte ohne feste Infrastruktur zu einem Funknetz, wobei Datenpakete von Knoten zu Knoten über Netzwerkprotokolle weitergeleitet werden. Diese Sensorknoten bestehen im Grunde aus einem Prozessor, Datenspeicher, Stromquelle, Sensoren und einem Funkmodul zur Kommunikation. Somit besteht eine große Auswahl an vorhandener Hardware zum Bau eines Sensorknotens. Zur Entwicklung eines drahtlosen Sensorknotens auf Arduino Basis, müssen die möglichen Angebote der Hardwarekomponenten analysiert werden. Im Folgenden werden die verschiedenen Arduino Versionen vorgestellt.

Anforderungsprofil Die Masterarbeit „Fließgewässermonitoring mit drahtlosen Sensornetzen“ zeigte, dass das Konzept drahtloser Sensornetzwerke für die Erhebung von Temperaturen in Fließgewässern grundsätzlich geeignet ist, jedoch ist der Kommunikationsradius der Tmote Sky Sensorknoten auf dem Wasser zu klein für das senden in einem freischwimmenden Sensornetzwerk. Aufgrund der Limitierung durch die Tmote Sky Hardware lässt sich dieses Problem nicht einfach lösen [3]. Der Sensorknoten Prototyp dieser Arbeit soll deswegen auf der Basis von Arduino aufgebaut werden. Arduino ist modular aufgebaut und somit lassen sich Komponenten leicht austauschen, womit man auf verschiedene Anforderungen Szenarien einfacher reagieren kann. Dabei soll der Sensorknoten seine Position bestimmen können, das Funkmodul einfach austauschbar sein, Messinstrumente (Thermometer) einfach anschließbar und der Arduino Sensorknoten sollte Preislich günstiger sein, als der Tmote Sky Sensorknoten.

3.1 Arduino Versionen

Arduino ist eine Open Source Plattform, basierend auf Flexibilität, sowie einfach zu nutzender Hard- und Software. Das Arduino Mikrokontroller Board wurde als ein flexibles Board konzipiert, das heißt, dass es modular aufgebaut wurde, wodurch zusätzliche Komponenten leicht an das Board angeschlossen werden können. Diese Flexibilität ermöglicht es dem Arduino Board, leicht auf verschiedene Anforderungen zu reagieren. Alle Arduino Boards haben eines gemeinsam, sie werden durch das „*Arduino IDE* (integrierte Entwicklungsumgebung)“ programmiert. Darüber hinaus können eine Menge Unterschiede unter den Boards bestehen. Unterschiede reichen von der Anzahl an Ein- und Ausgängen bis Geschwindigkeit, Betriebsspannung und Form, dies sind nur einige der möglichen Variablen.

Veröffentlicht wurde Arduino im Jahre 2005 als ein Werkzeug für Studenten an der Interaction Design Institute Ivrea (IDII), die wenig oder gar keine

Erfahrung mit Elektronik oder Programmieren hatten. Arduino ist hierbei eine Open Source Plattform, dadurch sind alle Schaltpläne und Quellcodes freierhältlich unter öffentlichen Lizenzen, wodurch verschiedene Hersteller ihre eigenen Versionen von Arduino herstellen können. Entwickelt wurde der erste Arduino Prototyp von Massimo Banzi, ein Professor für interaktives Design in Lugano sowie Kopenhagen und David Cuartielles, ein Professor für interaktive Technologien in Malmö, an dem Interaction Design Institut Ivrea im nördlichen Italien. Der Name „Arduino“ wurde von einer Bar in Ivrea übernommen, diese wiederum bezog ihren Namen vom König Arduin, der im Jahre 1002 gekrönt wurde. Als Grund für die Entwicklung galt ein Budget Mangel. Wie viele andere Professoren arbeitete Banzi mit einem BASIC Stamp Mikrokontroller, aber diese hatte laut Banzi zwei Probleme: Es hatte nicht genug Prozessorstärke für Projekte seiner Studenten und es war etwas zu teuer mit Kosten um die 100 US Dollar. Somit kam die Idee einen eigenen kostengünstigeren Mikrokontroller zu erstellen. Ein Arduino besteht aus einem Input/Output Board mit einem Mikrokontroller und digitalen sowie analogen Ein- und Ausgängen. Beim Arduino werden verschiedene Eingaben eingelesen und in Ausgaben umgewandelt. Einige mögliche Eingaben sind Licht, das auf einem Sensor trifft oder ein Knopf, der gedrückt wird. Ausgaben wäre zum Beispiel das Aktivieren eines Motors oder das Einschalten einer LED-Lampe. Die Entwicklungsumgebung basiert auf *Processing* und das Programmieren selbst erfolgt mit *C* bzw. *C++* [15, 16].

Im Folgenden werden einige Arduino Plattformen aufgezählt und ihre Eigenschaften verglichen. Durch die große Anzahl an verschiedenen Arduino Versionen beschränkt sich diese Arbeit zumeist auf Versionen von dem Original Hersteller.

3.1.1 ATmega328 Boards

Der Mikrokontroller ATmega328 ist ein 8-Bit-Mikrokontroller, der in einigen Arduino Plattformen eingesetzt wird und für viele elektronische Projekte eingesetzt werden kann. Durch den Mikrokontroller allein sind fast alle Arduino Boards in diesen Abschnitt, im Sinne der Input/Output Anzahl und Speicher, identisch. Die Unterschiede der Arduino Boards in diesem Abschnitt stammen zumeist in Dingen wie Programmschnittstellen, Form und Betriebsspannung. Der Mikrokontroller ATmega328 hat eine maximale Taktfrequenz von $20\,\mathrm{MHz}$, $32\,\mathrm{kB}$ programmier Platz, $2\,\mathrm{kB}$ SRAM, $1\,\mathrm{kB}$ EEPROM, 1 UART Anschluss, maximal 6 PWM Anschlüsse, maximal 8 analoge Eingänge, maximal 23 digitale Ein- und Ausgänge [17]. Es gibt sehr viele Arduino Modelle, die in diese Kategorie gehören, jedoch würde die vollständige Aufzählung den Rahmen dieser Arbeit sprengen, somit wurde der Abschnitt nur auf die wichtigsten und dem Projekt relevanten Arduinos eingeschränkt.

Arduino Uno Der *Arduino Uno* ist der „Standard" der Arduino Boards. Es hat zwei verschiedene Versionen, es gibt das Board in einer *SMD-Version* (Surface-Mounted Device) und einer *DIP-Version* (Dual Inline Package), welche sich an der Bauart des verwendeten Mikrokontrollers unterscheidet. Bei der DIP-Version gibt es einen DIP-Sockel, wo der Kontroller aufgesteckt werden kann, bei der SMD-Version ist der Kontroller kleiner und auf dem Board festgelötet. Dabei hat die DIP-Version den Vorteil, dass im Falle eines Defekts, der Kontroller sich leicht und kostengünstig ersetzten lässt. Erweiterungsplatinen, die mit den Arduino Boards kompatibel sind und die den Boards mehr Funktionalität bieten, werden „Shields" genannt. Das *Uno* Board ist mit mehr Shields kompatibel als alle anderen Arduino Versionen [18]. Ein USB-Interface ist mit der seriellen Schnittstelle des Prozessors verbunden. Über das USB-Interface erfolgt die Verbindung mit dem PC. Als Verbindungskabel zum Rechner wird ein USB A-B-Kabel benötigt. Hierüber werden auch die Programme geladen. Das Arduino Uno wird dabei mit der Arduino Software (IDE) programmiert, außerdem ist der Uno mit den meisten Programm Bibliotheken kompatibel, da er die neueste Version der am längsten weiterentwickelten Arduino-Reihe ist. In den meisten Büchern und Tutorials wird dieses Arduino Model verwendet. Der Arduino Uno hat auch größte Anzahl an verfügbaren Programm Bibliotheken. Siehe Abbildung 1a.

Arduino Pro *Arduino Pro* ist eine reduzierte Version von dem Arduino Uno. Es besitzt immer noch den ATMega328 Mikrokontroller, aber bei ihm sind alle fest gelöteten Verbindungen und der USB-zu-Serial Stecker entfernt worden. Wie der Uno zuvor kommt der Arduino Pro auch in 2 Variationen: $5\,\mathrm{V}/16\,\mathrm{MHz}$ und $3{,}30\,\mathrm{V}/8\,\mathrm{MHz}$. Das $6\,\mathrm{V}/16\,\mathrm{MHz}$ Board läuft mit der gleichen Spannung und Taktfrequenz wie der Uno. Die $3{,}30\,\mathrm{V}/8\,\mathrm{MHz}$ Variante läuft mit einer kleineren Betriebsspannung, dies erleichtert die Stromversorgung, sodass diese einfacher durch Batterien getätigt werden kann. Es heißt jedoch auch, dass die Taktfrequenz herunter gesenkt werden musste und nur mit $8\,\mathrm{MHz}$ arbeitet [19]. Da das Arduino Pro für semipermanente Installation in Objekten gedacht ist, fehlen die vorgelöteten Verbindungsstecker. Semipermanente Installation bedeutet im Generellen, das alle benutzten Verbindungen zum Arduino festgelötet werden müssen, im Gegensatz zu Pins, wo das Ein- und Ausstecken einfach geschieht. Das Board ist mit den meisten Shields kompatibel, aber die Verbindungen müssen selber an das Board angebracht werden, um die Shields aufstecken zu können. Durch die fehlenden Verbindungsstecker braucht man mehr als ein USB Kabel, um im Pro Board programmieren zu können. Man braucht ein externes Board, um vom USB des Computers zum Serial des Arduinos, zu übersetzen. Das Arduino Pro wird dabei mit der Arduino Software (IDE) programmiert [19]. Siehe Abbildung 1b.

Arduino Pro Mini Das *Arduino Pro Mini* ist eine stark miniaturisierte Version des Arduino Pro. Wie das Pro besitzt es einen ATMega328, jedoch sind durch seine viel kleinere Form alle Pinlöcher neu arrangiert, welches eine Shield Kompatibilität verhindert. Zudem sind alle Pins entfernt und auch der USB-zu-Serial Stecker fehlt. Wie die Pro Version kommt das Arduino Pro Mini auch in 2 Variationen: $5\,\mathrm{V}/16\,\mathrm{MHz}$ und $3{,}30\,\mathrm{V}/8\,\mathrm{MHz}$. Das $5\,\mathrm{V}/16\,\mathrm{MHz}$ Board läuft mit der gleichen Spannung und Taktfrequenz wie der Uno. Das $3{,}30\,\mathrm{V}/8\,\mathrm{MHz}$ Board läuft mit einer kleineren Betriebsspannung, dies erleichtert die Stromversorgung, sodass diese einfacher durch Batterien getätigt werden kann. Jedoch heißt es auch, dass die Taktfrequenz herunter gesenkt werden musste und nur mit $8\,\mathrm{MHz}$ arbeitet. Da auch der Arduino Pro Mini, wie der Arduino Pro, für semipermanente Installation in Objekten gedacht ist, fehlen die vorgelöteten Verbindungsstecker. Somit wird beim Pro Mini auch ein externes Board gebraucht um vom USB des Computers zum Serial des Arduinos zu übersetzten, damit der Pro Mini auch programmiert werden kann. Das Arduino Pro Mini wird dabei mit der Arduino Software (IDE) programmiert [20]. Siehe Abbildung 1c.

Arduino Nano Das *Arduino Nano* basiert auf dem Arduino Uno, wurde jedoch speziell für das Arbeiten mit dem *Breadboard* (Stecksystem) entwickelt. Es kann nämlich einfach auf ein Breadboard aufgesteckt werden. Dadurch ist es um einiges kleiner als der Uno und die Pins wurden auf Stiftleisten herausgeführt, sodass das Board mit allen gängigen Stecksystemen kompatibel ist. Durch das kleinere Profil fehlt dem Nano jedoch eine DC-Strombuchse, so wie die Kompatibilität zu Shields. Anders als beim Uno verbindet sich der Nano mit einem Mini-USB Kabel. Die USB Verbindung ermöglicht die Stromversorgung und dient zum Programmieren des Nanos. Das Arduino Nano wird mit der Arduino Software (IDE) programmiert [21]. Siehe Abbildung 1d.

Seeeduino v4.2 Da Arduino ein Open Sourceprojekt ist, gibt es viele verschiede Versionen von verschiedenen Herstellern. *Seeeduino* basiert auf dem Design des Arduino Uno mit einigen Modifikationen. Die wichtigste Zusatzmöglichkeit ist die Umschaltung zwischen $5\,\mathrm{V}$ und $3{,}30\,\mathrm{V}$, womit der Einsatz von $3{,}30\,\mathrm{V}$ Shields ermöglicht wird. Somit kann durch die geringere Spannung die Lebensdauer von möglichen Batterien gespart werden. Zu den Erweiterungen zählt u.a. eine Zusatzreihe Anschlusspins, welche im Steckbrett-kompatiblen Raster angeordnet sind, sodass auch gewöhnliche Breadboards als Unterlage verwendet werden können. Dazu kommen 3 Grove Schnittstellen, zwei I2C und ein UART, wodurch das Board sich einfach mit anderen Grove Schnittstellen verbinden kann. Da Seeeduino in der Form dem Arduino Uno gleicht und die Betriebsspannung zwischen $5\,\mathrm{V}$ und $3{,}30\,\mathrm{V}$ wechseln kann, ist das Board mit mehr Shields kompatibel als

alle anderen Arduino Versionen [22].

Ein USB-Interface ist mit der seriellen Schnittstelle des Prozessors verbunden. Über das USB-Interface erfolgt die Verbindung mit dem PC. Als Verbindungskabel zum Rechner wird ein Mikro-USB Kabel benötigt. Hierüber werden auch die Programme geladen. Wie bei dem Arduino Uno wird dabei mit der Arduino Software (IDE) programmiert, außerdem ist der Seeeduino mit den meisten Programm Bibliotheken kompatibel, da er auf der Architektur des Uno's basiert und somit die neueste Version der am längsten weiterentwickelten Arduino-Reihe ist [22]. Siehe Abbildung 1e.

Eine Übersicht der Eigenschaften dieser Arduinos gibt die Tabelle 1, diese Angaben basieren auf den Datenblättern der Hersteller [18, 19, 20, 21, 22].

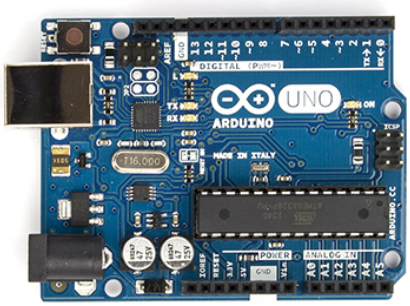
3.1.2 ATmega32U4 Boards

Der 8-Bit-Mikrokontroller ATmega32U4 ist weitgehend gleich mit dem ATmega328. Wie auch beim ATmega328 können die Arduino Boards mit diesem Chip in vielen Projekten eingesetzt werden. Es hat eine maximale Taktfrequenz von $16\,\mathrm{MHz}$, $32\,\mathrm{kB}$ programmier Platz, $2{,}50\,\mathrm{kB}$ SRAM, $1\,\mathrm{kB}$ EEPROM, 1 UART Anschluss, maximal 8 PWM Anschlüsse, maximal 12 analoge Eingänge, maximal 23 digitale Ein- und Ausgänge [27]. Der größte Unterschied zum ATmega328 ist, dass der Chip eine eingebaute USB Schnittstelle hat, was einen zweiten Prozessor überflüssig macht und dadurch die Kosten des ATmega32U4 sinken. Zusätzlich sind höhere Durchsatzraten bei der Kommunikation möglich. Außerdem ermöglicht es den ATmega32U4 Boards von einem verbunden Computer, als Maus und Tastatur erkannt zu werden, wo durch die Arduino Boards Eingaben am Computer vornehmen können. Wie auch beim ATmega328 gibt es sehr viele Arduino Modelle, die in diese Kategorie gehören und die vollständige Aufzählung würde den Rahmen dieser Arbeit sprengen, somit wurde der Abschnitt nur auf die wichtigsten und dem Projekt relevanten Arduinos eingeschränkt.

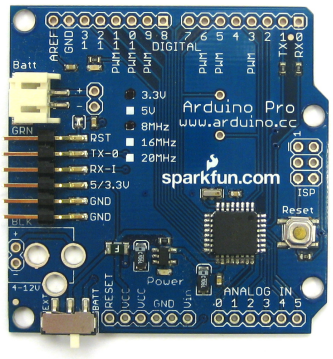
Arduino Leonardo Wie der Ardunio Uno bei den ATmega 328 Boards ist der *Leonardo* der Standard der ATmega32U4 Boards. Es teilt die Form und die Position der Ein- und Ausgänge des Arduino Uno, wodurch es auch mit den meisten Shields kompatibel bleibt. Der Ardunio Leonardo hat wenige Unterschiede zum Ardunio Uno. Es fehlt ein Mikrochip und somit fehlt ein USB-zu-Serial-konvertierender IC. Des Weiteren ist die USB-Verbindung anders, der Leonardo benutzt zum Verbinden mit einem Computer ein Mikro-B USB Kabel. Das Arduino Leonardo wird, wie auch die anderen Arduinos, mit der Arduino Software (IDE) programmiert. Die meisten Programmbibliotheken funktioniert auch mit dem Arduino Leonardo. Lediglich exotischere Programmbibliotheken funktionieren damit u. a. nicht [28]. Siehe Abbildung 2a.

Tabelle 1: ATmega328 Boards mit Eigenschaften

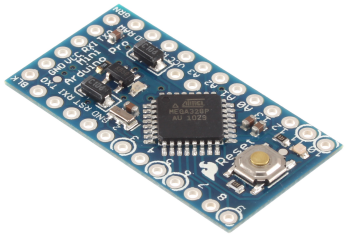
V MHz	Uno	Pro	Pro Mini	Nano	Seeeduino v4.2
Taktfrequenz	16 MHz	16 MHz / 8 MHz	16 MHz / 8 MHz	16 MHz	16 MHz
SRAM	2 kB	2 kB	2 kB	2 kB	2 kB
EEPROM	1 kB	1 kB	1 kB	1 kB	1 kB
Flash Memory	32 kB	32 kB	32 kB	32 kB	32 kB
Digitale I/O Pins	14	14	14	14	14
PMW Pins	6	6	6	6	6
Analoge Pins	6	6	6	8	6
Betriebs-spannung	5 V	5 V / 3,30 V	5 V / 3,30 V	5 V	3,30 V / 5 V
Versorgungs-spannung (empfohlen)	$7 - 12$ V	$5 - 12$ V / $3,3 - 12$ V	$5 - 12$ V / $3,3 - 12$ V	$7 - 12$ V	$7 - 12$ V
Versorgungs-spannung (Grenze)	$6 - 20$ V	—	—	$6 - 20$ V	—
Power Jack	X	X	—	—	X
ICSP	X	X	—	X	X
Reset Knopf	X	X	X	X	X
Verbindungs-schnittstelle	USB	Externes Board nötig	Externes Board nötig	Mini USB	Mikro USB
Shield Kompatibel	X	X	—	—	X
Zusätze / Merk-male	—	Batterie Power Jack, Alle Pins müssen selbst gelötete werden	Alle Pins müssen selbst gelötete werden	Breadboard kompatibel	Zusätzliche Raster Löcher für Breadboard Kompatibili-tät, Schalter um Betriebs-spannung zu ändern, 3 extra Grove Anschlüsse (zwei I2C und 1 UART)
Dimension / Größe	$53,4 \times 68,6$ mm	$52,0 \times 53,3$ mm	18×33 mm	18×45 mm	$53,4 \times 68,6$ mm
Preis	ca. 20 €	ca. 15 €	ca. 10 €	ca. 23 €	ca. 20 €



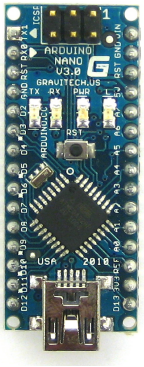
(a) Arduino Uno (Bildquelle: [23])



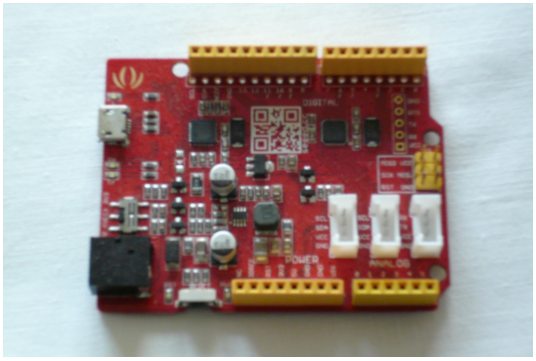
(b) Arduino Pro (Bildquelle: [24])



(c) Arduino Pro Mini (Bildquelle: [25])



(d) Arduino Nano (Bildquelle: [26])



(e) Seeeduino

Abbildung 1: Arduino Versionen (1)

Arduino Micro Wie auch der Arduino Pro Mini ist der *Arduino Micro* eine stark miniaturisierte Version des Arduino Leonardo, ähnelt aber dem Nano, in dem es auch Breadboard kompatibel ist. Als auch der Leonardo hat der Micro einen ATmega32U4 Mikrokontroller, jedoch mussten durch seine miniaturisierte Form alle Pins in eine neue Position gebracht werden, was eine Shield Kompatibilität verhindert. Im Unterschied zum Pro Mini, besitzt der Micro einen Mikro-B-USB Stecker, wodurch es leicht mit einem USB-Kabel mit einem Computer verbunden werden kann und keine zusätzliche Hardware benötigt wird. Das Arduino Micro wird, wie auch die anderen Arduinos, mit der Arduino Software (IDE) programmiert. Die meisten Programmbibliotheken funktioniert auch mit dem Arduino Micro. Lediglich exotischere Programmbibliotheken funktionieren damit u. a. nicht [29]. Siehe Abbildung 2b.

Arduino Pro Micro Der *Arduino Pro Micro* ist wie der Arduino Micro eine stark miniaturisierte Version des Arduino Leonardo. Wie der Leonardo hat der Micro einen ATmega32U4 Mikrokontroller, und wie bei Arduino Micro wurden wegen der kleineren Form alle Pins neu arrangiert, welches eine Shield Kompatibilität verhindert. Auch besitzt der Pro Micro, wie der Arduino Micro, einen B-USB Stecker, wodurch es leicht mit einem Kabel mit einem Computer verbunden werden kann und es keine zusätzliche Hardware benötigt. Durch seine noch kleinere Form als beim Arduino Micro, hat er jedoch weniger digitale Ein-und Ausgänge. Außerdem fehlen jegliche Pins und müssen daher selbst dran gelötet werden. Wie die Arduino Pro Version kommt das Arudino Pro Micro auch in 2 Variationen: $5\,\mathrm{V}/16\,\mathrm{MHz}$ und $3{,}30\,\mathrm{V}/8\,\mathrm{MHz}$. Das $5\,\mathrm{V}/16\,\mathrm{MHz}$ Board läuft mit der gleichen Spannung und Taktfrequenz wie der Uno. Das $3{,}30\,\mathrm{V}/8\,\mathrm{MHz}$ Board läuft mit einer kleineren Betriebsspannung, dies erleichtert die Stromversorgung, sodass diese einfacher durch Batterien getätigt werden kann. Jedoch heißt es auch, dass die Taktfrequenz herunter gesenkt werden musste und nur mit $8\,\mathrm{MHz}$ arbeitet. Das Arduino Pro Micro wird, wie auch die anderen Arduinos, mit der Arduino Software (IDE) programmiert. Zur Verbindung mit einem Computer wird lediglich ein Mikro-B USB Kabel benötigt. Die meisten Programmbibliotheken funktionieren auch mit dem Arduino Pro Micro. Lediglich exotischere Programmbibliotheken funktionieren damit u. a. nicht [30, 31].

Eine Übersicht der Eigenschaften dieser Arduinos gibt Tabelle 2, diese Angaben basieren auf den Datenblättern der Hersteller [28, 29, 30, 31].

3.1.3 Weitere Boards

Durch die große Anzahl an verschiedenen Arduino Modellen, gibt es viele Arduinos die nicht einem Mikrokontroller Board zugeordnet werden

Tabelle 2: ATmega32U4 Boards mit Eigenschaften

	Leonardo	Micro	Pro Micro
Taktfrequenz	16 MHz	16 MHz	16 MHz / 8 MHz
SRAM	2,50 kB	2,50 kB	2,50 kB
EEPROM	1 kB	1 kB	1 kB
Flash Memory	32 kB	32 kB	32 kB
Digitale I/O Pins	20	20	18
PMW Pins	7	7	5
Analoge Pins	12	12	9
Betriebs-spannung	5 V	5 V	5 V / 3,30 V
Versorgungs-spannung (empfohlen)	$7 - 12$ V	$7 - 12$ V	$5 - 12$ V / $3,3 - 12$ V
Versorgungs-spannung (Grenze)	$6 - 20$ V	$6 - 20$ V	—
Power Jack	X	—	—
ICSP	X	X	—
Reset Knopf	X	X	—
Verbindungs-schnittstelle	Mikro USB	Mikro USB	Mikro USB
Shield Kompatibel	X	—	—
Zusätze / Merkmale	Einige exotische Programm bibliothe-ken funktionie-ren nicht	Breadboard kompatibel	Alle Pins müssen selbst gelötete werden
Dimension / Größe	$53,4 \times 68,6$ mm	18×48 mm	18×33 mm
Preis	ca. 20 €	ca. 20 €	ca. 20 €

können. In diesem Abschnitt werden zwei prominente Arduino Versionen vorgestellt.

Arduino Mega *Arduino Mega* hat alle Eigenschaften, welche Arduino Uno hat. Zusätzlich besitzt es viele Verbesserungen, wie mehr digitale und analoge Ein- und Ausgänge, sowie mehr Arbeitsspeicher. Dabei basiert es auf einem ATmega2560 Mikrokontroller, welches dieses Upgrade ermöglicht. Durch die neuen Zusätze ist der Mega ein paar Zentimeter länger als der Uno, wodurch es mit den meisten Shields kompatibel bleibt, die für den Arduino Uno, *Arduino Duemilanove* und *Arduino Diecimila* entworfen wurden. Ein USB-Interface ist mit der seriellen Schnittstelle des Prozessors verbunden. Über das USB-Interface erfolgt die Verbindung mit dem PC. Als Verbindungskabel zum Rechner wird ein USB A-B-Kabel benötigt. Hierüber werden auch die Programme geladen. Das Arduino Mega wird dabei mit der Arduino Software (IDE) programmiert [32], siehe Abbildung 2c.

Arduino Due Das *Arduino Due* ist das erste Arduino Board, welches auf einen 32-Bit-Mikrokontroller basiert. Wie auch das Mega Board besitzt es mehr digitale und analoge Ein- und Ausgänge, mehr Speicher und viele weitere Verbesserung gegenüber dem Arduino Uno. Anders als der Arduino Mega arbeitet es mit einer viel höheren Taktfrequenz. Es ist primär für kompliziertere Projekte gedacht, die von seiner stärkeren Leistung profitieren. Außerdem ist er um einiges größer und teurer als der Uno oder Leonardo. Im Unterschied zu anderen Arduino Boards, arbeitet der Arduino Due mit $3{,}30\,\mathrm{V}$, dass bedeutet, es ist oft nicht kompatibel mit Shields, da diese oft mit $5\,\mathrm{V}$Spannung arbeiten. Ein USB-Interface ist mit der seriellen Schnittstelle des Prozessors verbunden. Über das USB-Interface erfolgt die Verbindung mit dem PC. Als Verbindungskabel zum Rechner wird ein USB-Kabel benötigt. Hierüber werden auch die Programme geladen. Das Arduino Due wird dabei mit der Arduino Software (IDE) programmiert [33], siehe Abbildung 2d.

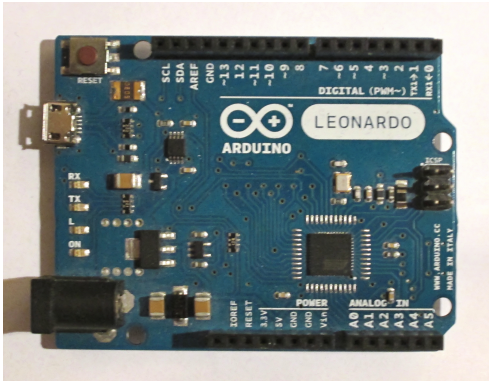
Eine Übersicht der Eigenschaften dieser Arduinos gibt die Tabelle 3, diese Angaben basieren auf den Datenblättern der Hersteller [32, 33].

3.2 Arduino Shields

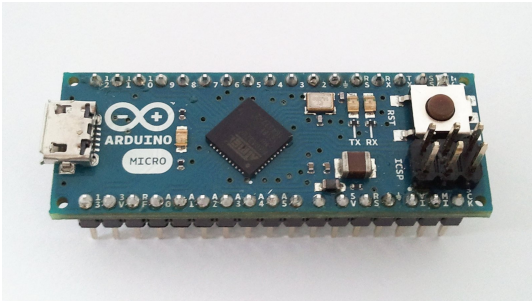
Shields nennt man Erweiterungsplatinen, die zum Aufstecken auf einen Arduino gedacht sind. Der Zweck dieser modularen Platinen ist es, in dem Arduino mehr Funktionalität zu installieren. Die Funktionen reichen von Funkkommunikation und Vernetzung, zur Anzeige (LCD) und zum Lesen von GPS-Daten. Sollte man eine bestimmte Funktionalität haben wollen, besteht eine gute Chance, dass es ein Shield gibt, dass dies erfüllen kann. Es gibt Hunderte verschiede Arduino Shields und deren Zahl steigt

Tabelle 3: Weitere Boards mit Eigenschaften

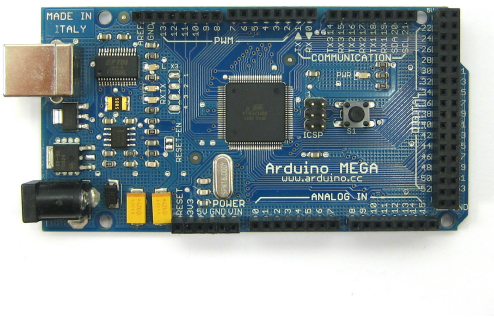
	Mega	Due
Taktfrequenz	16 MHz	84 MHz
SRAM	8 kB	96 kB
EEPROM	4 kB	—
Flash Memory	256 kB	512 kB
Digitale I/O Pins	54	54
PMW Pins	15	12
Analoge Pins	16	12
Betriebsspannung	5 V	3,30 V
Versorgungspannung (empfohlen)	$7 - 12$ V	$7 - 12$ V
Versorgungspannung (Grenze)	$6 - 20$ V	$6 - 16$ V
Power Jack	X	X
ICSP	X	X
Reset Knopf	X	X
Verbindungschnittstelle	USB	USB
Shield Kompatibel	X	X
Zusätze / Merkmale	4 UART Anschlüsse	4 UART, 1 SPI, 2 DAC, 2 TWI, 1 JTAG Anschluss, 1 Lösch Knopf, nur 3,3 V Shield Kompatibilität
Dimension / Größe	$53,3 \times 101,5$ mm	$53,3 \times 101,5$ mm
Preis	ca. 40 €	ca. 40 €



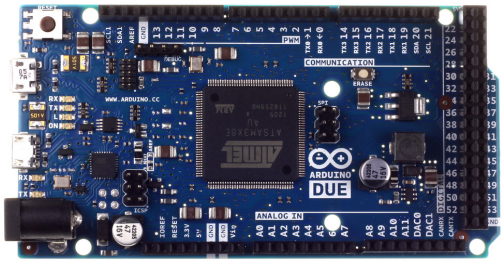
(a) Arduino Leonardo (Bildquelle: [34])



(b) Arduino Micro (Bildquelle: [35])



(c) Arduino Mega (Bildquelle: [36])



(d) Arduino Due (Bildquelle: [37])

Abbildung 2: Arduino Versionen (2)

stetig. Dabei haben die Shields oft eine Programmierbibliothek mit sich assoziiert. Diese Bibliotheken ermöglichen es einem einfach die Hardware Funktionen des Shields zu implementieren. Die Shields werden einfach auf den Arduino aufgesteckt. Der Arduino besitzt dafür Reihen von Plastik Löchern, Headers genannt, wo auf der Unterseite des Shields vorhandene Metallspitzen, genannt Pins, drauf passen. Viele Shields bieten auch noch die Möglichkeit weitere Shields aufzunehmen, diese werden einfach wie beim Arduino drauf gesteckt. Manche Shields nutzen jeden Pin des Arduino, andere wiederum nur einige. Wenn man Shields auf aufeinanderstapelt, muss man darauf achten, dass diese nicht die gleichen Pins benutzen. Durch die große Anzahl an erhältlichen Arduino Shields, beschränken wir uns in dieser Arbeit auf die nötigsten Shields für einen Sensorknoten mit unserem Anforderungsprofil.

Arduino Wireless SD Shield Das Wireless SD Shield gibt dem Arduino Board die Fähigkeit kabellose Kommunikation durch ein Wireless Modul durchzuführen. Es basiert auf dem XBee Modul von dem Hersteller Digi, es lassen sich aber auch andere Module mit gleichen Anschlüssen verbinden. Die Module werden einfach auf den Shield aufgesteckt. Die Reichweite dieser Module beträgt in Gebäuden bis zu 30 Meter oder draußen im Freien bis zu 100 Meter. Genau Reichweiten und benutze Funkfrequenzen sind von dem eingesetzten Modul abhängig. Die Platine hat auch einen kleinen Schalter, mit dem man auswählen kann, ob das Modul die Serial- / USB-Verbindung ersetzt oder es im Command Modus betrieben wird, womit sich verschieden Sende- und Meshnetzwerk Optionen konfigurieren lassen. Außerdem besitzt der Shield einen integrierten Mikro-SD-Karten-Slot, welches sich einfach über die SD-Programmierbibliothek ansprechen lässt. Der Preis des Wireless SD Shield liegt bei ca. 20 € [38].

SparkFun XBee Shield Wie auch der Wireless SD Shield, bietet der *SparkFun XBee Shield* dem Arduino Board die Fähigkeit kabellos Kommunikation mit einem Wireless Modul auszuführen. Auch hier wird ein XBee Wireless Modul Sockel verwendet, wodurch auch andere Module mit den gleichen Anschlüssen sich verbinden lassen. Der Shield bezieht über den 5 V Pin des Arduinos seine Stromversorgung, diese wird über einen Regulator am Shield auf 3,30 V reguliert und dem Kommunikationsmodul eingespeist. Der Shield hat einen kleinen Schalter, der die Serielle Verbindung des Shields XBee Sockels zum Arduino ändert. Man kann hierbei zwischen den digitalen Pins 0 und 1, zu den digitalen Pins 2 und 3 wechseln. Wie auch die anderen Shields wird der SparkFun Xbee Shield einfach auf dem Arduino aufgesteckt. Der Preis des Shields liegt bei ca. 15 € [39].

SparkFun RFM22 Shield Das RFM22 ist ein kostengünstiges ISM FSK Transceiver Modul, welches Kommunikation auf $433\,\mathrm{MHz}$ ISM und anpassungsfähige Sendestärke bis zu $+20\,\mathrm{dBm}$ liefert. Der *RFM22 Shield* verbindet das RFM22 Modul mit den zuständigen Verbindungen des Arduinos. Die RF22 Programm Bibliothek erlaubt es dem Arduino Nachrichten mit dem Standard „3-wire SPI“ zu senden und zu empfangen. Die Bibliothek beinhaltet Programmcode um verschiede Verbindungsstrukturen aufzubauen, von simplen nicht-adressierten Punkt-zu-Punkt Kommunikation bis zu einem voll adressierten Netzwerk mit Klient und Routern. Der Shield bietet zudem eine Loch Verbindung, wo eine Drahtantenne angebracht werden kann. Der Preis dieses Shields liegt bei $30\,€$ [40].

SparkFun Cellular Shield - MG2639 Der *SparkFun Cellular Shield* ist eine gute Alternative für ein Arduino Projekt, wenn eine Verbindung gebraucht wird, aber es kein WiFi oder Ethernet Verbindung gibt. Das ZTE MG2639 Modul, dass auf dem Shield aufgebaut wurde, unterstützt SMS, TCP, UDP und kann sogar benutzt werden um Telefonanrufe zu machen oder zu empfangen. Dies bedeutet, dass man Textnachrichten schreiben und senden kann, oder das Arduino mit dem Internet verbinden kann. Außerdem besitzt es einen Integrierten GPS Empfänger, wodurch man seine Position bestimmen kann. Der Schaltkreis des Shields lässt die Stromversorgung des Moduls zu auswählbaren $3{,}30\,\mathrm{V}$ oder $5\,\mathrm{V}$ umstellen. Abhängig vom Zustand, in dem es sich befindet, kann das MG2639 Modul relativ stromhungrig sein. Mit einem maximalen Stromverbrauch nimmt der Shield ca. $350\,\mathrm{mA}$ auf. Normalerweise bezieht er nicht so viel, aber benötigt ca. $260\,\mathrm{mA}$ bei einem Telefonanruf oder 80 mA bei einer Netzwerk Übertragung. Die Telefon und GPS Funktionen des MG2639 Moduls benötigen externe Antennen, die mit dem Modul verbunden sind. Dafür gibt es zwei U.FL Verbindungen an den Seiten des Chips. Der Preis dieses Shields liegt bei ca. $70\,€$ [41].

SparkFun GPS Shield Der *GPS Shield* erleichtert das Einfügen von GPS Funktionen zu einem Arduino. Der Shield ist kompatibel mit mehreren verschiedenen GPS Modulen. Es hat einen EM-406 GPS Receiver und es gibt Standflächen für EM-408 and EB-85A Verbindungen (Pins). Dazu gibt es noch Platz für ein UP501 GPS Modul. Ein An/Aus Schalter ist im Shield integriert, damit lässt sich die Stromzufuhr zu den Modulen kontrollieren. Außerdem hat das Board noch einen Reset Knopf. Der Shield hat auch eine „Standfläche“ für eine 12-Millimeter-Knopfbatterie, die als Back-up dienen kann. Der Shield ist in Standard Arduino-Shield Form und ist mit allen normalen Arduino Boards kompatibel. Hierbei liegt der Preis bei ca. $15\,€$ [42].

SparkFun GPS Logger Shield Der *SparkFun GPS Logger Shield* stattet den Arduino mit Zugriff auf ein GPS-Modul, SD Speicher Karte Sockel und alles weitere Nötige aus, um das Arduino in ein Gerät zur Positionsbestimmung zu entwickeln. Der Shield basiert auf einem GP3906-TLP GPS Modul, ein 66 Kannelieger GPS Empfänger mit einer MediaTek MT3339 Architektur und bis zu 10 Hz Update Rate. Das GPS-Modul streamed konstante Updates über ein TTL Level-Serial Port, welche man auf die SD-Karte speichert und/oder anderweitig benutzt. Alles auf dem Shield ist stark konfigurierbar. Ein Schalter erlaubt die Auswahl des GPS Moduls UART Schnittstelle zwischen Hardware oder Software Ports, die SD-Karte operiert über einen Hardware SPI Port, welches kompatibel mit den meisten Arduinos sein sollte und es gibt noch extra etwas freien Platz für weitere Komponenten, die für ein Projekt gebraucht werden könnten. Der GPS Logger Shield Hauptstromzufuhr erfolgt durch den $5\,\mathrm{V}$ Pin des Arduinos. Diese Spannung ist auf $3{,}30\,\mathrm{V}$ runter reguliert, welche dann das GPS-Modul und die SD-Karte versorgt. Diese zwei Komponenten verbrauchen im Durchschnitt $30\,\mathrm{mA}$, aber es gibt Spitzen bis zu $100\,\mathrm{mA}$. Weiterhin gibt es Platz für eine $12\,\mathrm{mm}$ Knopf Batterie, die als Backup Stromquelle dienen kann. Preislich liegt der SparkFun GPS Logger Shield bei ca. $45\,€$ [43].

SparkFun Weather Shield Der *Weather Shield* ist ein einfach zu nutzender Arduino Shield, welche dem Arduino die Möglichkeit gibt barometrischen Druck, relative Luftfeuchtigkeit, Helligkeit und Temperatur zu messen. Der Shield hat auch noch weitere Verbindungen auf dem Board, um optionale Sensoren für Windgeschwindigkeit, Regenmessung und GPS zur Positionsbestimmung und akkurater Zeitmessung. Der Shield benutzt den HTU21D für Luftfeuchtigkeit, den MPL3115A2 für barometrischen Druck und den ALS-PT19 für die Lichtsensoren. Es stehen Programmbibliotheken für HTU21D und MPL3115A2 bereit. Das Weather Shield arbeitet mit einer Spannung von $3{,}30\,\mathrm{V}$ bis zu $16\,\mathrm{V}$ und hat einen eingebauten Spannungsregulator. Der Shield wurde für den Arduino Uno entwickelt und funktioniert nicht mit einigen anderen Arduino Boards ohne Modifikationen. Der Preis des Shields liegt bei ca. $40\,€$ [44].

3.3 Vergleich

Den Anspruch von digitalen Eingängen erfüllen alle hier aufgezählten Arduino Versionen. Da die hier durchgeführten Sensor Funktionen keine große Rechenkapazität oder eine große Anzahl an digitalen Ein-/Ausgängen benötigen, fallen die teureren Arduino Versionen Mega und Due aus dem Vergleich. Die Arduino Versionen Pro, Pro Mini und Pro Micro sind zur Prototyp Entwicklung eher ungeeignet. Da alle Pins dort fest verlötet werden müssen verlieren sie die Flexibilität der anderen Arduino Versionen und somit lassen sich zum Beispiel die Funkmodule nicht einfach auswechseln.

Deswegen eignen sich diese Arduino Versionen eher zu Projekten, wo die Hardware festgelötet wird und nicht schnell gewechselt werden muss.

Durch die Breadboard Kompatibilität ermöglichen die Arduino Versionen Nano und Micro schnell verschiedene Schaltungen auf dem Breadboard zu erstellen. Jedoch hat die kleinere Form ihren Preis, es fehlt die Form und die Einsteckpins für Shields, dadurch sind sie nicht fähig Shields aufzunehmen und verlieren die Arduino Modularität. Der Arduino Leonardo steht dem Uno in Nichts nach, von der Taktfrequenz bis hin zu dem Preis; nur wenige exotische Programm Bibliotheken funktionieren auf dieser Version nicht. Leider wurde der Arduino Leonardo aus dem offiziellen Sortiment des originalen Herstellers genommen und es wird auch nicht weiter an dem Leonardo entwickelt.

Als die längste weiterentwickelte Arduino Reihe, erfüllt der Arduino Uno alle Anforderungen unseres Sensorknotens. Er ist mit den meisten Programm Bibliotheken kompatibel, die meisten Shield funktionieren mit ihm und preislich liegt er nicht weit von den günstigeren Arduino Versionen zurück. Der Arduino Uno stellt eine ausgezeichnete Wahl für den zu erstellenden Sensorknoten dar. Aber der Seeeduino bietet einige Vorteile gegenüber dem Uno. Von der Anzahl der digitalen Ein- und Ausgängen bis hin zum Preis sind die beiden Versionen identisch. Jedoch hat der Seeeduino einen zusätzlichen Schalter um die Betriebsspannung zwischen $3{,}30\,\mathrm{V}$ und $5\,\mathrm{V}$ zu wechseln und drei weitere Grove Anschlüsse für vereinfachte Verbindungen. Seeeduino hat noch zusätzliche Raster Löcher, wodurch leichte Modifikation mit Hilfe von Stiftleisten, es Breadboard kompatibel gemacht werden kann. Seeeduino stellt sich unter dem Anforderungsprofil, in diesem Vergleich, als beste Wahl dar.

4 Aufbau Prototypen

Die Eignung eines Sensorknotens auf Arduino Basis zu den Anforderungen des Gewässermonitorings werden mit dem Bau von Prototypen in dieser Arbeit untersucht. In diesem Kapitel wird der Aufbau der Prototypen sowie die eingesetzte Software beschrieben.

4.1 Anforderungen

Die Anforderungen des Sensorknotens ergeben sich aus dem Zweck des Fließgewässermonitoring und der schon durchgeführten Masterarbeit über „Fließgewässermonitoring mit drahtlosen Sensornetzen“ [3]. Der Sensorknoten Prototyp dieser Arbeit soll in der Lage sein, seine Position und derzeitige Zeit mittels eines GPS-Moduls zu ermitteln. Weiterhin sollte es in der Lage sein, Temperaturmessungen durchzuführen, sowie diese erhobenen Daten zwischenzuspeichern. Diese gespeicherten Messdaten sollen auch an einem Computer zusammengeführt werden, wo sie weiter ausgewertet werden können. Weiterhin sollte die Kommunikation verschiedene Funkfrequenzen abdecken können, dazu gehören $433\,\mathrm{MHz}$, $866\,\mathrm{MHz}$ und $2{,}40\,\mathrm{GHz}$. Zu diesem Zwecke sollen die Komponenten leicht austauschbar sein, ohne tiefe Eingriffe in die Hardware des Prototyps durchzuführen. Die Stromversorgung soll durch eine Batterie geschehen und das Betreiben des Prototyps über einige Zeit sichern. Es wird angestrebt, dass die Kosten dieses Prototypen unter den Prototypen von [3] liegen.

4.2 Erster Prototyp

In Abbildung 3 sieht man den ersten Prototypen und alle seine Komponenten, zusätzlich wird das Bild mit farblichen Ringen unterteilt. Dabei sieht man im Ring A den Wireless SD Shield mit einem Xbee Funkmodul, sowie den $433\,\mathrm{MHz}$ Funkmodulen und dem Thermistor. Im Ring B liegen der GPS Shield mit GPS-Modul und die GPS-Modul-Antenne. Im Ring C liegt das Arduino Uno und im Ring D ein Batteriehalter mit An- / Aus Knopf und einer $9\,\mathrm{V}$ Batterie, die zur Stromversorgung dient. Die gesamt Kosten des ersten Prototyps belaufen sich auf insgesamt auf $130\,€$, siehe Tabelle 6. Für die Kosten der Grundkomponenten, siehe Tabelle 5 und für die Kosten der 3 Funkmodule, siehe Tabelle 4.

4.2.1 Basisplattform

Der Sensorknoten Prototyp dieser Arbeit wird auf der Basis von Arduino Uno und seinen verschiedenen Ableger aufgebaut. Es bietet genug Ein- und Ausgabe Pins für die verschiedenen Module des Prototyps. Ein USB-Interface ist mit der seriellen Schnittstelle des Prozessors verbunden. Über

Einzelteile	Kosten
Arduino Uno	20 €
Thermistor	4 €
GPS-Bee Kit	30 €
Wireless SD Shield	20 €
SD-Karte	4 €
GPS Shield	2 €
Summe	80 €

Tabelle 4: Kosten der Grundkomponenten des ersten Sensorknotens

Funkmodul	Kosten
XBee Series 1	25 €
RFbee	20 €
XY-MK-5V / XY-FST	5 €
Summe	50 €

Tabelle 5: Kosten der Funkmodule

	Kosten
Grundkomponente	80 €
Funkmodule	50 €
Summe	130 €

Tabelle 6: Kosten des ersten Sensorknotens

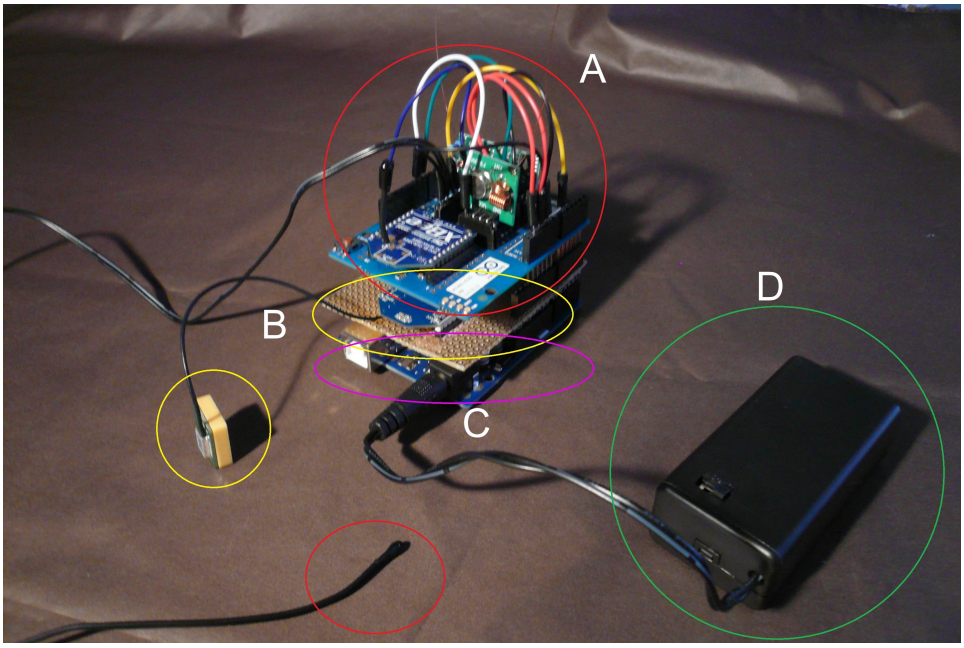


Abbildung 3: Erster Prototyp: Wireless SD Shield, XBee und 433 MHz Modul (A), Thermistor (B) GPS Shield, GPS Modul und GPS Antenne (C), Arduino Uno (D) Batteriehalter mit 9 V Batterie

das USB-Interface erfolgt die Verbindung mit dem PC. Zum Programmieren des Mikrokontrollers dient die kostenlose Arduino Software (IDE). Preislich liegt Arduino Uno um die 20 €. In Tabelle 7 werden die genutzten Pins des Arduinos anschaulich vorgestellt.

4.2.2 Temperatursensor

Zur Temperaturmessung wurde der „10K Precision Epoxy Thermistor - 3950 NTC“ vom Hersteller „Adafruit“ gewählt. Diese werden oft in Klimaanlagen, Wasserrohren und anderen nassen Stellen verwendet. Es hat eine Temperaturmessreichweite von -55 °C bis zu 125 °C, eine Länge von 45 cm und ein Durchmesser von 3,50 mm. Weiterhin ist die Wasserdichte des Thermistors durch eine Beschichtung von PVC (Polyvinylchlorid) gesichert, die Temperaturen von -55 °C bis 105 °C standhält [45]. Zum korrekten Auslesen der Temperatur wird ein kleiner Schaltkreis um den Thermistor benötigt. Dieser besteht aus dem 3,30 V-Pin und AREF-Pin des Arduinos, einem 10k Ohm Resistor, dem Thermistor und den GND-Pin des Arduinos. Zur Auslesung der Temperaturdaten selbst wird lediglich ein analoger Eingang des Arduino Uno's benötigt. Die Kosten des Thermistors belaufen sich auf ca. 4 €. In Abbildung 5b ist dieser Schaltkreis dargestellt.

Pin	Verwendungszweck
Digital 0	Verbindung mit Computer, Funkmodul
Digital 1	Verbindung mit Computer, Funkmodul
Digital 2	Verbindung mit GPS-Modul
Digital 3	Verbindung mit GPS-Modul
Digital 4	Verbindung mit SD-Karte
Digital 5	LED (Grün)
Digital 6	LED (Rot)
Digital 7	
Digital 8	433 MHz Sender Modul
Digital 9	433 MHz Empfänger Modul
Digital 10	433 MHz Sender Modul Kontroller
Digital 11	Verbindung mit SD-Karte (SPI)
Digital 12	Verbindung mit SD-Karte (SPI)
Digital 13	Verbindung mit SD-Karte (SPI)
Analog 0	Thermistor Ausleser

Tabelle 7: Verwendungszweck der Pins vom Arduino Uno

4.2.3 GPS

Als GPS-Modul verwendet der Prototyp das von „Seeed Studio“ vertriebene „GPS Bee Kit“, welche eine Platine mit einem GPS-Chipsatz und einer daran angeschlossenen Antenne ist. Dieses GPS-Modul hat die Form und Pin Verteilung des XBee Formats, wodurch es mit den meisten XBee Kompatiblen Shields verträglich ist. Die Datenkommunikation geschieht über eine UART-Schnittstelle mit 3 V Signalpegel und formatiert wird der GPS-Datenstrom mit dem NMEA-0183 Protokoll. Der Betriebsspannungsbereich reicht von 2,70 V bis 3,60 V. Verbunden wird der GPS Bee Kit über die Pins 2 und 3 des Arduinos und kommuniziert über den UART-Bus mit einer Datenrate von 9600 Bit/s. Die Kosten des GPS Bee Kit liegen bei ca. 30 € [46].

4.2.4 Wireless SD Shield

Die Kompatibilität des Wireless SD Shield mit verschiedenen Funkmodulen macht ihn zur guten Wahl, um leicht verschiedene Funkfrequenzen abzudecken. Diese werden einfach auf den dazu vorhandenen Sockel aufgesteckt und es werden keine weiteren Schritte benötigt. Der integrierte Mikro-SD-Karten-Slot ermöglicht auch das einfache Speichern der Auswertungsdaten der Sensoren auf einer Mikro-SD-Karte. Durch den auf dem Shield befindlichen Schalter lässt sich leicht von der Kommunikation mit dem Funkmodul zur Kommunikation mit dem Computer über USB Kabel wechseln. Die Kosten des Shields liegen bei ca. 20 €.

In Abbildung 4a, ist ein Schaltbild des Wireless SD Shields mit den

verwendeten Komponenten zusehen, weiterhin sieht man in der Abbildung 4b, wie die hinzugefügten Pins auf der Rückseite verlötet wurden, um eine optimale Platzverwendung zu gewährleisten. Zur Übersicht wurden die beiden Schaltkreise der $433\,\mathrm{MHz}$ Funkmodule und des Thermistors in getrennten Abbildungen dargestellt. In Abbildung 5a sieht man wie die $433\,\mathrm{MHz}$ Empfänger und Sender auf die Pins gesteckt wurden, sowie wie die Verbindung zum Arduino hergestellt wird. In Abbildung 5b sieht man den Thermistor Schaltkreis auf dem Shield.

SD Karte Als SD Karte wurde die $8\,\mathrm{GB}$ große microSDHC Karte vom Hersteller „Kingston“ gewählt. Diese passt in den SD-Slot des Wireless SD Shields und nutzt die $3{,}30\,\mathrm{V}$ als Betriebsspannung. Dabei reichen $8\,\mathrm{GB}$ Speicher, um alle in den Experimenten nötigen Speicherbedarf abzudecken. Der Preis der Speicherkarte liegt bei ca. $5\,€$ [47].

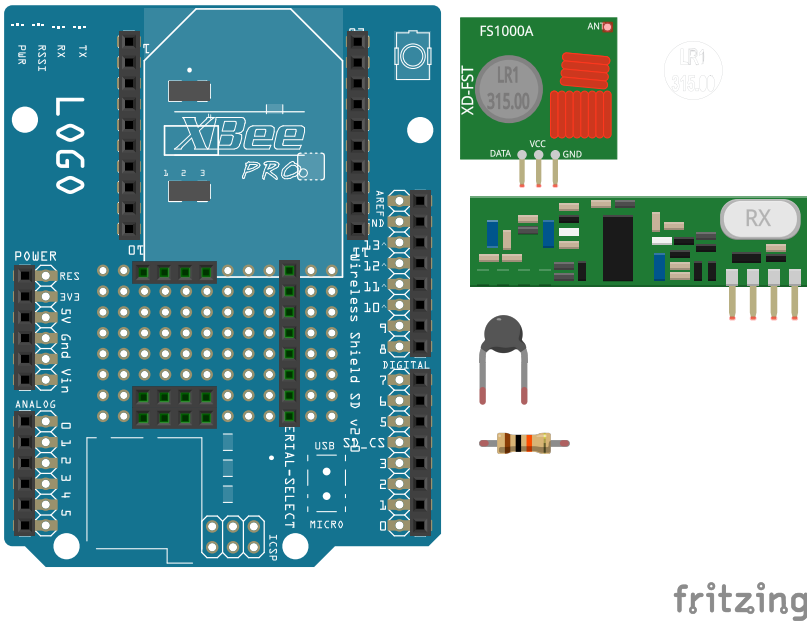
4.2.5 GPS Shield

Zum Halten des vorhandenen GPS-Moduls wurde ein selbst entwickelter Shield genutzt. Es basiert auf einer Lochrasterplatine, die mit $2{,}54\,\mathrm{mm}$ Buchsenleisten versehen wurde, sodass der Shield einfach auf den Arduino aufgesteckt werden kann. Weiterhin wurden $2{,}50\,\mathrm{mm}$ Buchsenleisten verwendet um einen XBee Sockel zu fertigen, wo dann das GPS-Modul einfach aufgesteckt werden kann. Der Shield verwendet dabei die Pins 2 und 3 des Arduino Uno, um mit dem GPS zu kommunizieren. Weiterhin hat der Shield 2 LEDs, eine Grüne und eine Rote, die die Pins 5 und 6 nutzen. Die Kosten des selbst gebauten Shields liegen bei ca. $2\,€$.

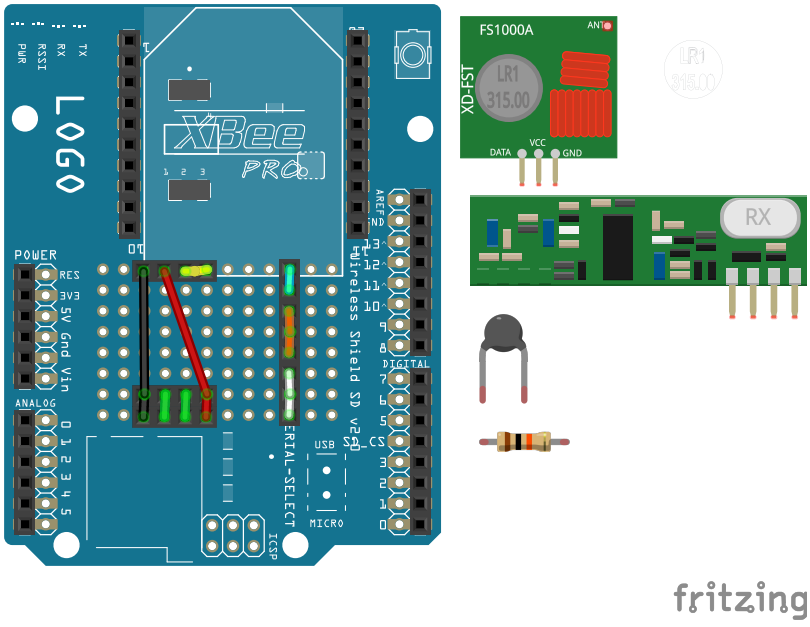
In Abbildung 6a und Abbildung 6b sieht man den selbst entwickelten GPS Shield von oben und unten. Dazu wurde auch ein Schaltbild entwickelt. In Abbildung 7a und Abbildung 7b sieht man, wie der eigene GPS Shield verkabelt und gelötet wurde.

4.2.6 Funk Module

Xbee XBee 1mW Wire Antenna - Series 1 (802.15.4) ist ein von Digi International entwickeltes Funkmodul, welches in der Funkfrequenz von $2{,}40\,\mathrm{GHz}$ arbeitet. Siehe Abbildung 8a. Mit den Werkseinstellungen synchronisieren sich die XBee Funkmodule automatisch mit anderen kompatiblen XBee Modulen in Reichweite und man kann, ohne weitere Arbeit, Daten funken. Standardmäßig läuft das XBee Funkmodul mit einer Baudrate von 9600, die Datenkommunikation geschieht mit einem $3{,}30\,\mathrm{V}$ Signalpegel, die Betriebsspannung liegt bei $2{,}80\,\mathrm{V}$ bis $3{,}40\,\mathrm{V}$ und die Sendereichweite wird mit $90\,\mathrm{m}$ im Freien und mit $30\,\mathrm{m}$ in Gebäuden angegeben. Möchte man Einstellungen des Moduls ändern, lässt es sich einfach mit einem Programm von Digi International erreichen. Änderung der Baudrate, Schlafmodus,

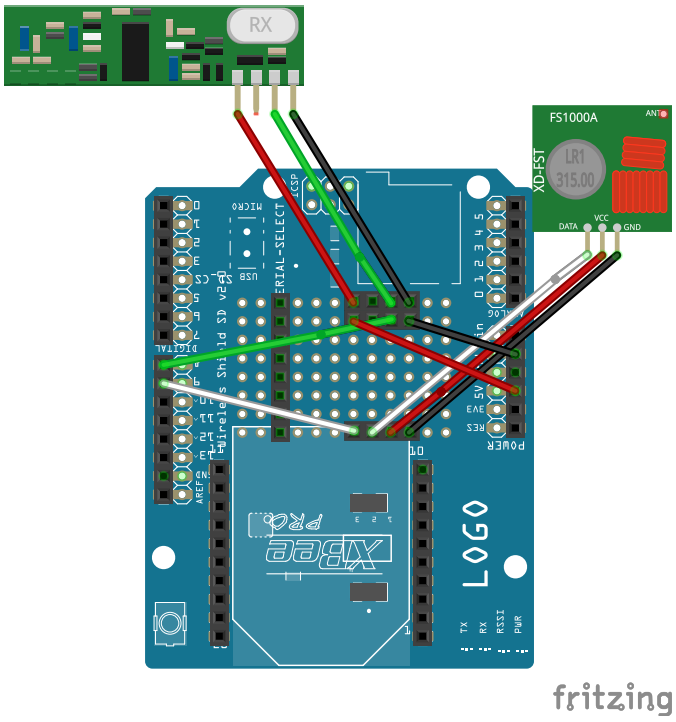


(a) Pins und verwendete Komponente

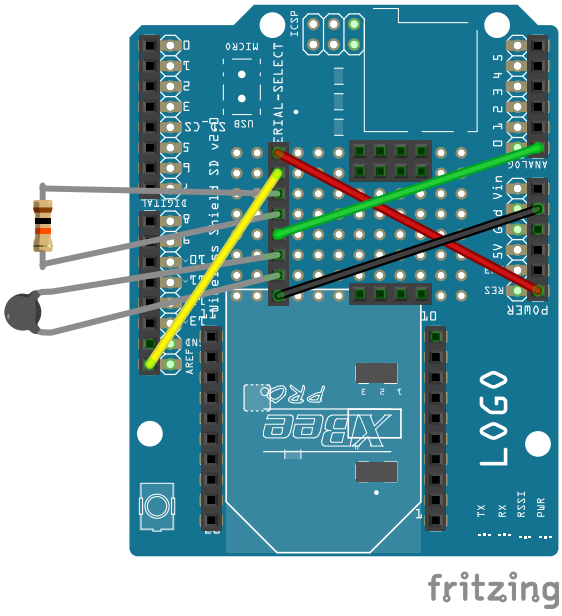


(b) Schaltkreis der verlöteten Pins

Abbildung 4: Wireless SD Shield Schaltbild

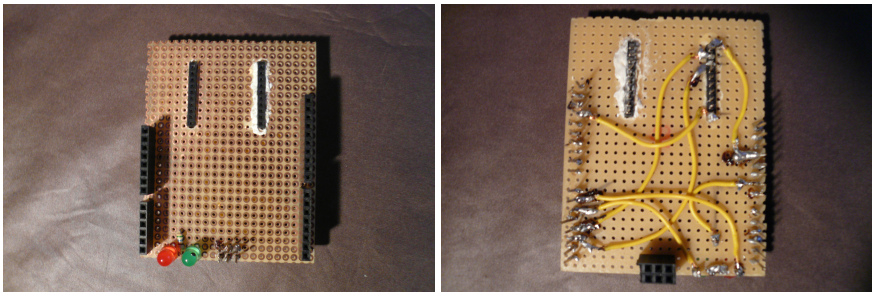


(a) Schaltkreis vom 433 MHz Funkmodul



(b) Schaltkreis des Thermistors

Abbildung 5: Wireless SD Shield Schaltbild mit Komponenten Schaltkreisen



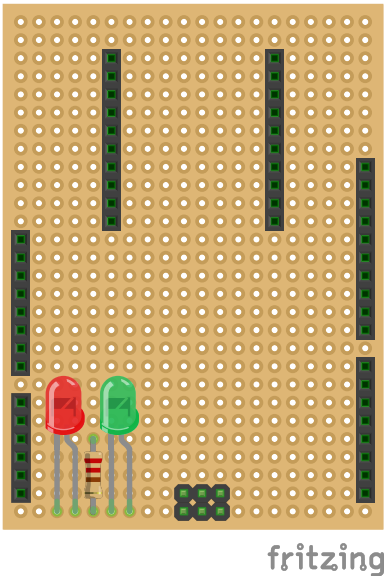
(a) Ansicht: Oben **(b)** Ansicht: Unten

Abbildung 6: GPS Shield

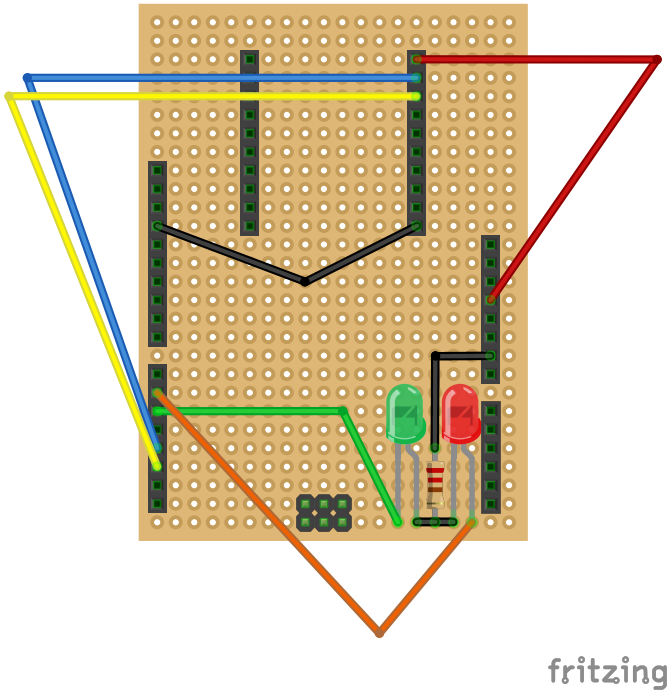
Betriebsspannung und viele weitere Einstellungen sind möglich. Der Preis des Funkmoduls liegt bei ca. 25 € [48].

RFbee Das vom Seeed Studio entwickelte RFbee Funkmodul arbeitet auf der Funkfrequenz von 868 MHz MHz und kann auch auf 915 MHz umgestellt werden. Wie auch beim XBee Modul, können sich die RFbee Module schon mit Werkseinstellungen von allein miteinander verbinden. Siehe Abbildung 8b. Standardmäßig läuft das RFbee Funkmodul mit einer Baudrate von 9600, die Datenkommunikation geschieht mit einem 3,30 V Signalpegel, der Betriebsspannungsbereich liegt bei 3 V bis 3,60 Vund die Sendereichweite wird mit 120 m im freien und mit 50 m in Gebäuden angegeben. Das Modul lässt sich auch über Konsolenbefehle weiter konfigurieren. Man kann Veränderungen an der Adressierung durchführen, Frequenz umschalten oder das Modul in den Schlafmodus versetzen. Weiterhin ist das RFbee Modul auch mit dem XBee Sockel des Wirelesss SD Shields kompatibel. Der Preis dieses Funkmoduls liegt bei ca. 20 € [49].

XY-MK-5V / XY-FST XY-MK-5V / XY-FST ist ein Sender und Empfänger Modul Set, das über die 433 MHz Funkfrequenz kommuniziert. Das Sender Modul hat eine mögliche Betriebsspannung von 3,50 V bis 12 V, die Sendefrequenz liegt bei 433 MHz und es wird eine mögliche Sendereichweite von 20 m bis 200 m angegeben. Dabei hat es eine Größe von 19×19 mm. Es braucht zur Kommunikation mit dem Arduino Uno einen digitalen Pin. Siehe Abbildung 8c. Das Empfänger Modul arbeitet mit einer Betriebsspannung von 5 V und die Empfangsfrequenz liegt bei 433 MHz. Die Größe des Moduls beträgt 30×14 mm. Zur Kommunikation des Arduin Uno's wird, wie beim Sender, ein digitaler Pin benötigt. Da beide Module ohne Antenne geliefert wurden, wurde ein 17 mm langes Kupferkabel als Antenne an beide Module angelötet. Der Preis pro Set liegt bei ca. 5 € [50].

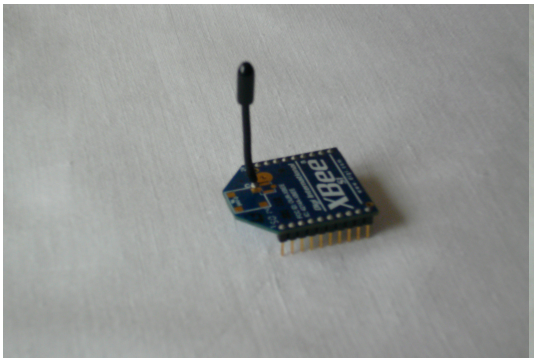


(a) Ansicht: Oben

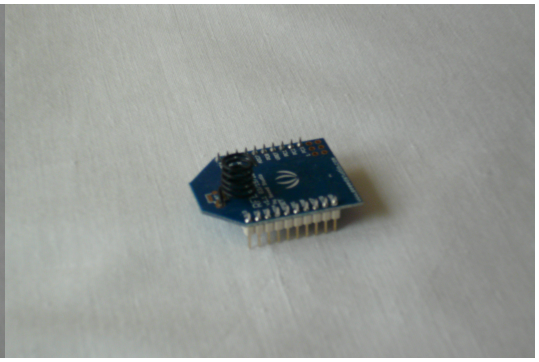


(b) Ansicht: Unten

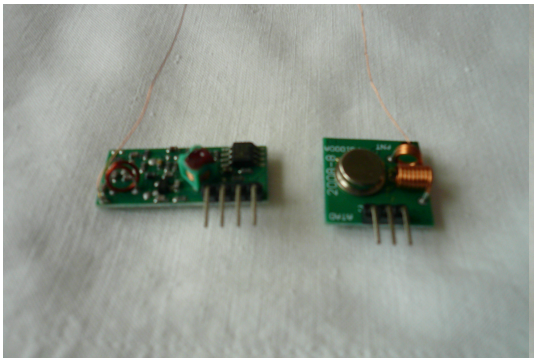
Abbildung 7: GPS Shield Schaltbild



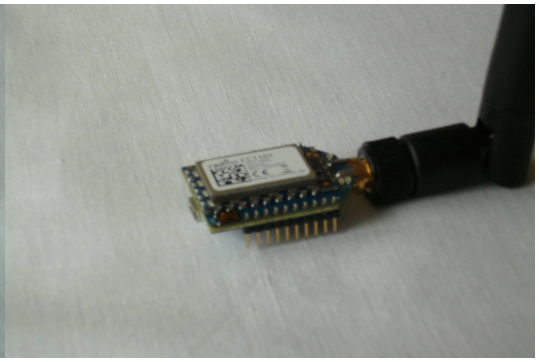
(a) XBee



(b) RFbee



(c) XY-MK-5V / XY-FST



(d) Radino CC1101

Abbildung 8: Funkmodule

4.2.7 Software

Zum Programmieren des Sensorknotens wird die Open Source Arduino Software (IDE) verwendet. Diese läuft auf Windows, Mac OS X und Linux, geschrieben wird in Java und es basiert auf Processing und anderer Open Source Software. Die Main Loop Schleife des Sensorknotens kann man in 5 Schritte unterteilen:

1. gpsToTinyGPS(1000)
2. temp = thermistor()
3. gpsToText
4. writetoSD()
5. sendMessage(Text)

Im ersten Schritt wird jede Sekunde die Daten aus dem GPS-Modul ausgelesen und zu einem Tiny+ GPS Objekt formatiert. Tiny+ ist dabei ein Open Source GPS Datenbibliothek. Im zweiten Schritt wird die Temperatur über den Thermistor bestimmt und einer Variablen zugewiesen. Beim dritten Schritt werden die GPS Daten sowie die Temperatur zu einem String hinzugefügt. Im vierten Schritt wird dieser String auf der SD-Karte gespeichert und im letzten Schritt wird der String über das Funkmodul gesendet.

Temperatur Messung Code Zur Temperaturmessung mit dem vorhandenen Thermistor, welcher ein Heißleiter (NTC) ist, kann die *Steinhart-Hart-Gleichung* verwendet werden. Diese beschreibt Veränderungen eines elektrischen Widerstandes bei einem Thermistor zufolge einer Temperaturänderung [51].

$$\frac{1}{T} = A + B\ln(R) + C(\ln(R))^3 \tag{1}$$

T ist die Temperatur in Kelvin. R der Widerstand bei T in Ohm. A, B und C sind Steinhart-Hart Koeffizienten, die vom Typ und Model des Thermistors und von dem geschätzten Temperaturbereich abhängig sind. Diese Formel ist jedoch ziemlich Komplex und benötigt Kenntnis von mehreren Variablen des Thermistors, die nicht vorhanden sind. Deswegen wird hier die simplere *B-Parameter* Formel benutzt, die sich von der Steinhart-Hart-Gleichung ableiten lässt [51].

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}\ln(\frac{R}{R_0}) \tag{2}$$

Hierbei muss man nur T_0 wissen, welche die Raumtemperatur mit $25\,^{\circ}\mathrm{C}$ ist. B ist der Temperaturkoeffizient des Thermistors, welcher bei

dem genutzten Thermistor einen Wert von 3950 hat. R_0 ist der Widerstand bei Raumtemperatur, bei dem genutzten Thermistor ist es ein Wert von 10K Ohm. Nun kann man R, der gemessene Widerstand des Thermistors, mit dem analogen Pin von Arduino auslesen. Dadurch erhält man T, Temperatur in Kelvin, die einfach in Celsius umgerechnet wird [51].

4.2.8 Probleme

Im Laufe der Arbeit entstanden einige Schwierigkeiten beim Aufbau des Prototyps. Hier werden kurz die größten Probleme und ihre Auswirkungen erläutert.

Speicherplatz SRAM Der Arduino Uno hat einen SRAM mit der Größe von $2\,\mathrm{kB}$. Die ersten Prototyp Versuche zeigten, dass man schnell an die Grenzen des SRAM stößt, den schon mit einer $75\,\%$ Belegung des SRAMs wird der Betrieb des Arduino instabil, da der Arduino selbst $25\,\%$ des SRAMs zum Laufen benötigt. Hierbei belegten das Arduino IDE und der Code zur Messung, GPS Lokalisierung und Kommunikation, insgesamt $20\,\%$ des SRAMs. Dazu kommt die SD-Karte-Programmbibliothek mit $30\,\%$ Belegung, die GPS-Programmbibliothek mit $8\,\%$ Belegung und $18\,\%$ Belegung der Programmbibliothek der $433\,\mathrm{MHz}$ Sender- und Empfangsmodule. Wodurch die SRAM-Grenze zur Arduino überschritten und es instabil wurde. Das heißt, dass das Programm erfolgreich in den Arduino hochgeladen wird, aber es nicht läuft oder fehlerhaft ausgeführt wird. Anpassungen an der $433\,\mathrm{MHz}$ Programmbibliothek, ermöglichte das Senken des benötigten SRAM knapp unter der Grenze. Wodurch jedoch die gesamt Anzahl der zur übertragenden Bytes des $433\,\mathrm{MHz}$ Senders und Empfängers verkleinert wurden. Dadurch ist es nicht mehr möglich Temperatur und GPS Lokalisierung mit nur einem Sendedurchlauf zu übertragen. Diese müssen nun in Teilen gesendet werden, welche leichter verloren gehen können oder unvollständig ankommen.

433 MHz Modul Set Erste Versuche mit dem XY-MK-5V / XY-FST Sender und Empfänger Modul Set zeigten, dass ohne Antenne die Übertragungsreichweite mehrere Zentimeter nicht übersteigt und man eine Antenne benötigt. Nach anbringen einer $17\,\mathrm{mm}$ langen Kupferdrahtantenne erhöhte sich die Reichweite auf ungefähr $20\,\mathrm{m}$. Durch die Speicherprobleme des Arduinos und die eher große SRAM Anforderung des $433\,\mathrm{MHz}$ Modulen, ist eine weitere Überlegung das Sender und Empfänger Paar mit einem anderen Funkmodul in der nächste Prototyp Version zu ersetzten.

Funkmodul	Kosten
XBee Series 1	25 €
RFbee	20 €
XBee-PRO Shield (Radino CC1101)	45 €
Summe	90 €

Tabelle 8: Kosten der Funkmodule (2. Version)

	Kosten
Grundkomponente	80 €
Funkmodule	90 €
Summe	170 €

Tabelle 9: Kosten des zweiten Sensorknotens

4.3 Zweiter Prototyp

Mit den Speicherproblemen des ersten Prototyps ergab sich die Notwendigkeit einen weiteren Prototyp zu erstellen, der mit neuer Hardware dieses Problem umgeht. In Abbildung 9 sieht man den zweiten Prototyp und alle seine Komponenten. Die größte visuelle Veränderung dieser Version ist die Entfernung der Pins, die für das anstecken des XY-MK-5V / XY-FST Sender und Empfänger Modul Sets gedacht waren. Der Grund des Entfernens war die neue Hardware, ein XBee-PRO Shield mit einem Radino CC1101 $433\,\mathrm{MHz}$ Funkmodul. Der durch seine Kompatibilität mit dem XBee Sockel des Shields eine gute Wahl als Ersatz für das Funkmodulset darstellt, jedoch behinderten die festgelöteten Pins des Sets das Einstecken des XBee-PRO Shields, da dieser etwas größer als ein normales XBee Funkmodul ist. Somit wurde ein neuer Shield mit dem Thermistor Schaltkreis genommen, siehe Abbildung 10 für ein Schaltbild des Shields. Die gesamt Kosten des zweiten Prototyps belaufen sich auf insgesamt $170\,€$, siehe Tabelle 9. Die Kosten für die Grundkomponenten und für die 3 Funkmodule, sieht man in Tabelle 4 und Tabelle 8.

Radino CC1101 433 MHz Als Ersatz des $433\,\mathrm{MHz}$ XY-MK-5V / XY-FST Sender und Empfänger Modul Sets dient der XBee-PRO Shield mit dem Radino CC1101 $433\,\mathrm{MHz}$ Funkmodul, der mit einer $15\,\mathrm{cm}$ langen Antenne ausgestattet ist, siehe Abbildung 8d. Zudem kann man den XBee-PRO Shield selbst programmieren, womit man keine Arduino Programmbibliothek braucht und die $18\,\%$ RAM Belastung des Modulsets entfällt. Programmiert wird der XBee-PRO Shield mit einer älteren Arduino IDE Version,

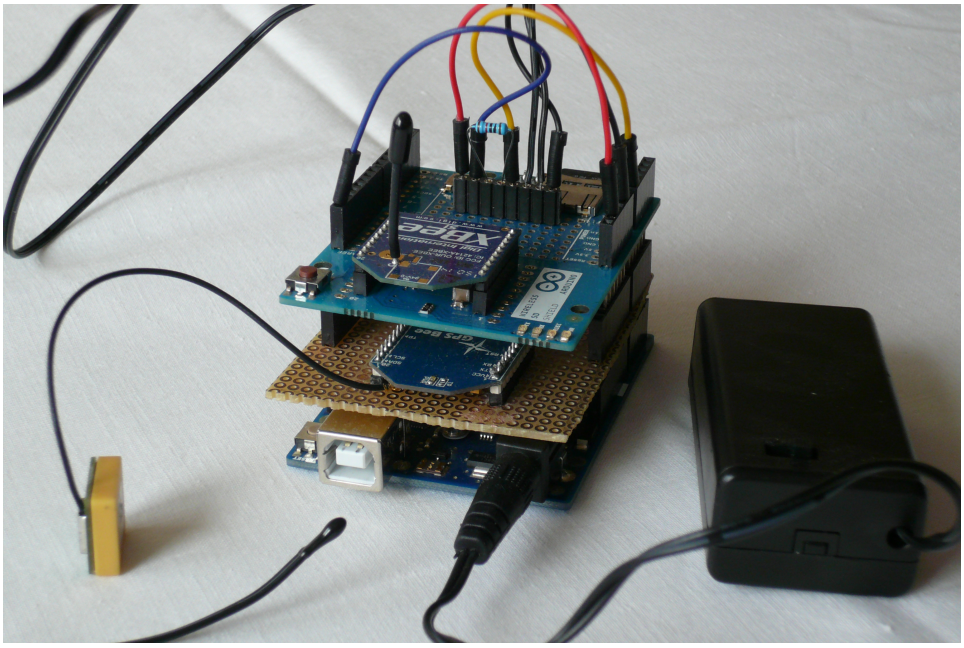


Abbildung 9: Zweiter Prototyp

der Version 1.6, den dazugehörigen Treiber findet man auf der Hersteller Seite [52]. Bevor man den Treiber mit dem Arduino IDE uploaden kann, muss der XBee-PRO Shield in den Bootloader Modus versetzt werden, dies geschieht durch zweimaliges Drücken des Resetknopfs. Der Vorteil des Radino CC1101 liegt in der Kompatibilität mit dem XBee Sockel des Wireless SD Shields, wodurch man die verschiedenen Funkmodule einfach austauschen kann. Weiterhin wird im Arduino weniger RAM verbraucht und der Programm Code konnte vereinfacht werden, da alle Funkmodule über den XBee-Sockel mit dem Arduino kommunizieren. Der Preis des XBee-PRO Shield mit dem Radino CC1101 liegt bei ca. 45 € [52].

4.4 Empfänger

In Abbildung 11 sieht man den Empfänger und alle seine Komponenten. Die Basisplattform ist ein Arduino Uno und als Shield wurde hierbei der SparkFun XBee Shield ausgewählt, da er mit dem gleichen XBee Sockel wie der Wireless SD Shield ausgestattet ist. Zusätzlich kann er über seinen Schalter die XBee Sockel Verbindung auf die digitale Pins 2 und 3 umstellen, wodurch eine gleichzeitige Verbindung über die digitalen Sockel 0 und 1 zum Computer ermöglicht wird. Dadurch können die gleichen Funkmodule, wie beim Sensorknoten, angewendet werden und zusätzlich die empfangen Daten direkt, über ein USB-Kabel, an den Computer weiter geleitet werden, wo diese weiterverarbeitet werden können.

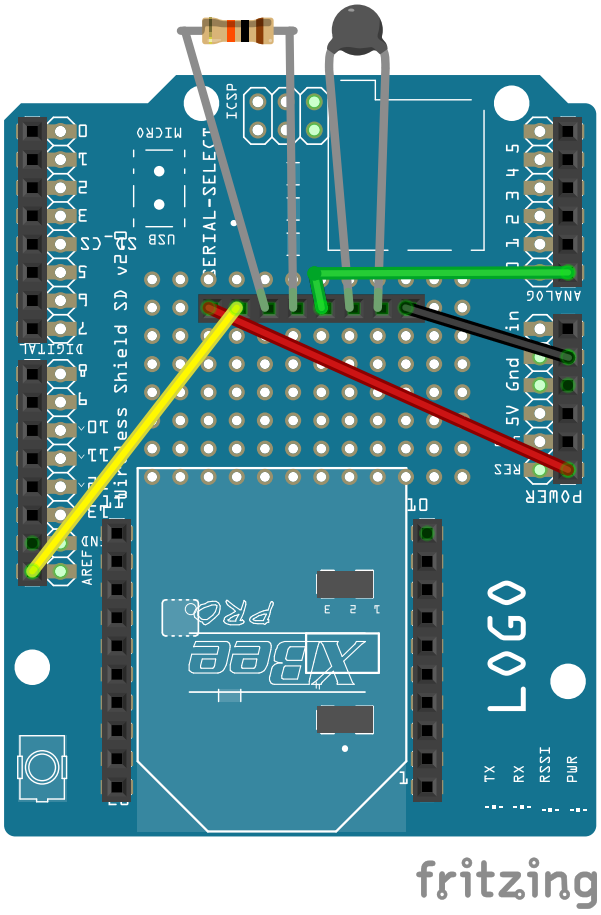


Abbildung 10: Wireless SD Shield Schaltbild (2. Version), mit Thermistors Schaltkreis

Pin	Verwendungszweck
Digital 0	Verbindung mit Computer, Funkmodul
Digital 1	Verbindung mit Computer, Funkmodul
Digital 2	Verbindung mit GPS-Modul
Digital 3	Verbindung mit GPS-Modul
Digital 4	Verbindung mit SD-Karte
Digital 5	LED (Grün)
Digital 6	LED (Rot)
Digital 7	
Digital 8	
Digital 9	
Digital 10	
Digital 11	Verbindung mit SD-Karte (SPI)
Digital 12	Verbindung mit SD-Karte (SPI)
Digital 13	Verbindung mit SD-Karte (SPI)
Analog 0	Thermistor Ausleser

Tabelle 10: Verwendungszweck der Pins vom zweitem Prototypen

Pin	Verwendungszweck
Digital 0	Verbindung mit Computer
Digital 1	Verbindung mit Computer
Digital 2	Verbindung mit Funkmodul
Digital 3	Verbindung mit Funkmodul
Digital 4	
...	...
...	Keine weiteren Pins benutzt

Tabelle 11: Verwendungszweck der Pins vom Empfänger

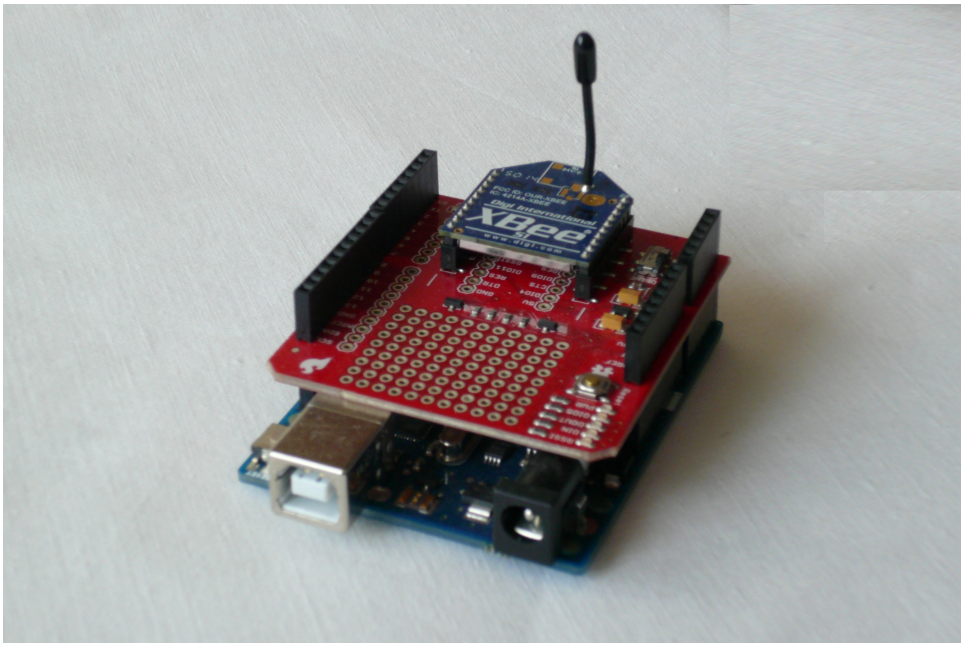


Abbildung 11: Empfänger mit Xbee Funkmodul

5 Messungen / Experiment

Zum Überprüfen des Konzepts von leicht austauschbaren Funkmodulen und Kommunikation auf der Wasseroberfläche wurden Messungen am Land und auf dem Wasser durchgeführt. Dabei wurde der zweite Sensorprototyp und der Empfänger in festen Abständen fixiert und der Sensorknoten sendete nun jede Sekunde einen Datensatz an den Empfänger. Der Empfänger leitet den Datensatz an einen, über USB angeschlossenen Computer, wo diese gespeichert werden. Wenn alle 120 Datensätze übertragen wurden, sendet der Sensorknoten noch 30 Datenpakete an den Empfänger, die auch am Computer gespeichert werden. Nachdem dies geschehen ist, wird das Funkmodul ausgetauscht und der Vorgang wiederholt, bis alle drei Funkfrequenzen abgedeckt worden sind. Im Laufe des Versuchs wurden nur die Funkmodule ausgetauscht, es erfolgte keine Veränderung im Code des Arduino.

5.1 Messdaten

Für die durchgeführten Messungen wurde der Sensorknoten mit einem eigenen entwickelten Code über die Arduino IDE programmiert. Dabei sendet er jede Sekunde einen Datensatz mit Sender ID, Paketsequenznummer, GPS Koordinaten, GPS Datum, GPS Zeit und die gemessene Temperatur. Zudem wird jeder versendete Datensatz auf der SD-Karte des Wireless SD Shields gespeichert. Nachdem 120 Datensätze versendet wurden, schickt der Sensorknoten Datenpakete, welche die größtmögliche Datenmenge zum versenden haben. Diese Datenpakete bestehen aus vier Datensätzen, die zuvor versendet wurden und sich auf der SD-Karte befinden. Insgesamt werden 30 Datenpakete versendet. Dieser Sendevorgang wurde mit allen 3 Funkmodulen durchgeführt, damit die $2{,}40\,\mathrm{GHz}$, $866\,\mathrm{MHz}$ und $433\,\mathrm{MHz}$ Funkfrequenzen abgedeckt werden. Der Empfänger, ausgestattet mit den gleichen Funkmodulen wie der Sensorknoten, empfängt die Datensätze, sowie die Datenpakete und leitet diese umgehend an einen über USB verbunden Computer weiter, wo diese in einer Textdatei zur weiteren Verarbeitung gespeichert werden.

Zur weiteren Analyse der empfangenen Daten, wird die Paketsequenznummer verwendet, dadurch lässt sich die Auslieferungsrate der Datensätze und Datenpakete zwischen dem Sensorknoten und dem Empfänger feststellen. Durch die Auslieferungsrate lässt sich erkennen, wie viele der gesendeten Daten des Sensorknotens auch beim Empfänger ankommen und somit lässt sich die Korrektheit der benutzten Funkfrequenzen beurteilen. Weiterhin werden die gespeicherten Daten auf Vollständigkeit überprüft, d. h. jeder Datensatz und jedes Datenpaket wird vom Inhalt überprüft, dass alle Werte korrekt angekommen sind.

Durch Limitierungen am Arduino können die gängigen Kenngrößen

Abstand	Funkfrequenz
2,5 m	2,4 GHz, 866 MHz und 433 MHz
5 m	2,4 GHz, 866 MHz und 433 MHz
10 m	2,4 GHz, 866 MHz und 433 MHz
15 m	2,4 GHz, 866 MHz und 433 MHz
20 m	2,4 GHz, 866 MHz und 433 MHz

Tabelle 12: Boden- /Wassermessung: Abstände und Funkfrequenzen

Received Signal Strength Indication (RSSI) und *Link Quality Indicator* (LQI), wie beim Tmote Sky Sensorknoten von [3], nicht verwendet werden, da durch die Anforderungen der Arbeit die benutzten Funkmodule leicht austauschbar sein sollten. Um diese Kennwerte beim Arduino auslesen zu können, müsste für jedes Funkmodul ein eigener Code geschrieben werden, wodurch diese nicht mehr einfach ausgetauscht werden könnten.

5.2 Aufbau der ersten Messung

Die Messungen erfolgten mit einem Sensorknoten und dem Empfänger, die in festen Abständen von $2{,}50\,\mathrm{m}$, $5\,\mathrm{m}$, $10\,\mathrm{m}$, $15\,\mathrm{m}$ und $20\,\mathrm{m}$ aufgestellt wurden. Dies erfolgte an zwei Orten, eine Messung an Land und eine Messung am Wasser in der Nähe eines Ufers. Um alle drei benötigten Funkfrequenzen abzudecken, wurden insgesamt 30 Messungen durchgeführt.

15 Messungen auf dem Boden, im Freien mit fünf verschiedenen Abständen und drei Funkfrequenzen, siehe Tabelle 12. Sowie 15 Messungen auf dem Wasser mit fünf verschiedenen Abständen und drei Funkfrequenzen, siehe Tabelle 12.

Die Messungen auf dem Land fanden im freien und auf einer ebenen Fläche statt. Dabei wurden der Sensorknoten und der Empfänger, mit dem gleichen Funkmodul ausgestattet, auf dem Boden in den festgelegten Abstand hingelegt und die Datensätze, sowie Datenpakte gesendet. Dies wurde nun bei den nächsten Abständen wiederholt. Während des Versuchs bestand Sichtlinie zwischen den Knoten und es befanden sich keine weiteren Hindernisse in der Umgebung. Die Messungen auf dem Wasser wurden am Ufer eines Sees durchgeführt. Dafür wurden der Sensorknoten und der Empfänger mit Schnur vor dem Abtreiben in den festen Abständen fixiert und die Daten gesendet. Wie auch bei der Land Durchführung bestand Sichtlinie zwischen den Knoten und es befanden sich keine weiteren Hindernisse in der Umgebung.

5.3 Ablauf der ersten Messung

Die Messungen auf dem Wasser wurden dabei auf dem Steinsee in Neuwied vorgenommen, da dieser leicht zugänglich ist und ein seichtes langes

Abbildung 12: Bodenmessung bei 5 m Abstand

Ufer bietet, wo die 20 Meter Abstand leicht erreicht werden konnten. Die Landmessungen fanden auf der Landstraße / Wanderweg in der Nähe des Steinsees statt. Beide Standorte sind von Wohngebieten weit entfernt und sollten somit wenige Interferenzen, die die Funkfrequenzen stören könnten, bieten. Das Wetter zur Zeit des Versuchs war kühl, trocken und windstill.

Bei der Landmessung wurden die Abstände von $2{,}50\,\mathrm{m}$, $5\,\mathrm{m}$, $10\,\mathrm{m}$, $15\,\mathrm{m}$ und $20\,\mathrm{m}$ abgemessen und markiert. Anschließen wurde der Empfänger und der Sensorknoten auf dem Boden, in Sichtlinie und ohne Hindernisse, platziert und die Messung wurde durchgeführt. Der Boden bei dem Versuch bestand dabei aus Pflastersteinen siehe in Abbildung 12, mit Sensorknoten und Markierung Pfahl bei $5\,\mathrm{m}$ Abstand. Bei jedem Abstand wurden 120 Datensätze und 30 Datenpaketen mit einer Funkfrequenz Versand. Dies wurde mit allen 3 Funkfrequenzen durchgeführt. Insgesamt dauerte der Landversuch mit allen 3 Funkfrequenzen ca. 1 Stunde, hinzu kommt noch die Zeit für Abmessungen und Aufbau.

Wie auch beim Land wurden für die Wassermessungen diesmal die Uferabstände markiert und anschließend wurde der Sensorknoten und der Empfänger in herkömmlichen Tupperware-Dosen an den Markierungen ins Wasser gelassen, siehe Abbildung 13 für die Wassermessung bei $5\,\mathrm{m}$ Abstand. Auch hier wurde jeder Abstand mit 120 Datensätzen sowie 30 Datenpaketen mit einer Funkfrequenz abgedeckt. Dies wurde mit allen 3 Funkfrequenzen durchgeführt. Die Messungen belaufen sich wie beim Landversuch um die 1 Stunde, hinzukommt Abmessungen und Aufbau.

Abbildung 13: Wassermessung bei 5 m Abstand

5.4 Ergebnisse der ersten Messung

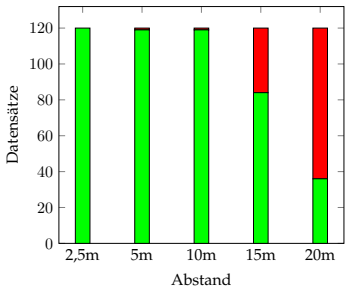
Die Auswertung der gewonnen Daten wurde für die Bodenmessungen und die Wassermessungen getrennt vorgenommen. Dabei werden die gespeicherten Daten des Computers genutzt. Hierbei werden die Datensätze sowie die Datenpakete auf erfolgreichen Empfang und Vollständigkeit überprüft, dies geschieht für alle Abstände und jede Funkfrequenz. Im Folgenden werden die gewonnen Werte und Ergebnisse der Messung vorgestellt.

Beginnend mit der Bodenmessung bei der $2{,}40\,\mathrm{GHz}$ Funkfrequenz. Bei den Datensätzen gingen in den kleineren Distanzen alle Datensätze ein, bis auf ein oder zwei Aussetzer die verloren gingen. Erst bei den größeren Distanzen von $15\,\mathrm{m}$ und $20\,\mathrm{m}$ gingen mehrere Datensätze verloren. Bei $15\,\mathrm{m}$ waren es $30\,\%$ der verschickten Datensätze und bei $20\,\mathrm{m}$ waren bis zu $70\,\%$. Hier kann man deutlich feststellen, dass bei größerer Distanz weniger Datensätze erfolgreich ankommen sind. Siehe Abbildung 14 a) für eine grafische Darstellung. Bei den Datenpaketen ist das Ergebnis ähnlich wie bei den Datensätzen. Bei den kleineren Abständen kamen alle Datenpakte erfolgreich an. Selbst bei der größeren Distanz von $15\,\mathrm{m}$ gingen nur einige Pakete unvollständig ein. Erst ab der größten Distanz von $20\,\mathrm{m}$ gingen $60\,\%$ der gesendeten Datenpakte nicht ein und einige waren unvollständig. Siehe Abbildung 15 a) für eine grafische Übersicht. Auffallend bei den Datenpakten ist, dass nur hier unvollständige Daten eingegangen waren. Datensätze kamen entweder erfolgreich ein oder gingen verloren, es gab

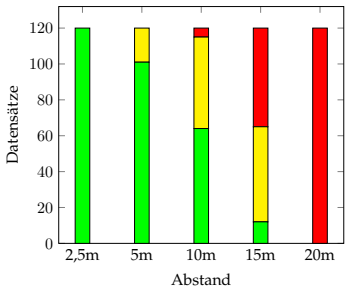
dort keine Unvollständige Datensätze.

In der Funkfrequenz von $866\,\mathrm{MHz}$ sind alle 120 Datensätze bei $2{,}50\,\mathrm{m}$ Abstand erfolgreich angekommen. Jedoch beginnen schon in der mittleren Distanz, Datensätze unvollständig einzugehen. Wo bei $5\,\mathrm{m}$ Abstand $16\,\%$ und bei $10\,\mathrm{m}$ $42\,\%$ unvollständig waren. Bei den größten Distanzen gehen die ersten Datensätze verloren und in größerer Zahl als bei der $2{,}40\,\mathrm{GHz}$ Frequenz. Bei der $15\,\mathrm{m}$ Distanz waren $44\,\%$ verloren gegangen und zusätzliche gingen noch $45\,\%$ unvollständig ein. In der größten Distanz gab es einen totalen Empfangsausfall und kein einziger Datensatz ging beim Empfänger ein. In Abbildung 14 b) wird dies in einem Balkendiagramm dargestellt. Bei den versendeten Datenpaketen bildet sich ein ähnliches Bild. In den mittleren Distanzen gehen um die $50\,\%$ der Datenpakte unvollständig bei dem Empfänger ein. Bei den größeren Abständen gehen zusätzlich viele Datenpakte verloren. Bei $15\,\mathrm{m}$ sind es bis zu $73\,\%$ Pakete und bei $20\,\mathrm{m}$ Abstand ging kein einziges Paket beim Empfänger ein. Die grafische Abbildung 15 b) zeigt dies an. In der Funkfrequenz $866\,\mathrm{MHz}$ fällt auf, dass nur in dem kleinsten Abstand alle Daten erfolgreich ankommen sind. Zusätzlich kann man gut sehen wie bei zunehmender Distanz erst Daten unvollständig ankommen, bis überhaupt keine Daten empfangen werden.

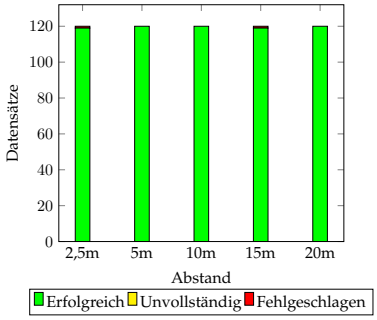
Bei der $433\,\mathrm{MHz}$ Funkfrequenz gingen bei den Datensätzen grundsätzlich alle Datensätze bei allen Abständen Erfolgreich ein. Es gab nur hin und wieder einen Aussetzer und ein Datensatz ging verloren. Siehe Abbildung 14 c) für eine grafische Darstellung des Datenverlaufs. Wie auch schon bei den Datensätzen gingen zumeist alle Datenpakte Erfolgreich ein, jedoch gab es auch hier einen Aussetzer und ein Datenpaket ging unvollständig ein. Sieh Abbildung 15 c) für eine Übersicht. Auffallend bei dem $433\,\mathrm{MHz}$ Bereich ist, dass grundsätzlich alle Daten erfolgreich ankommen, jedoch kann es zu Ausfällen kommen, wodurch ein wenig Daten verloren gehen. Im Vergleich mit den anderen Frequenzen zeigt sich $433\,\mathrm{MHz}$ mit der besten Auslieferungsrate bei der Bodenmessung.



(a) Messungen mit 2,4 GHz

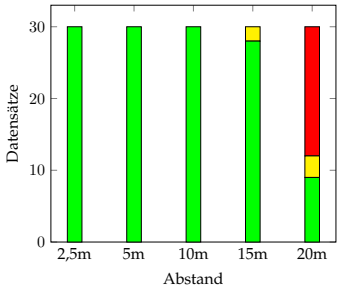


(b) Messungen mit 866 MHz

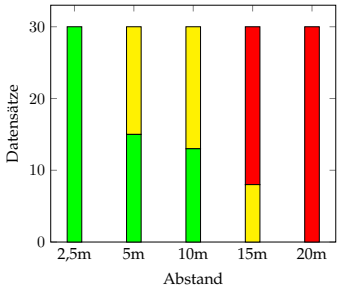


(c) Messungen mit 433 MHz

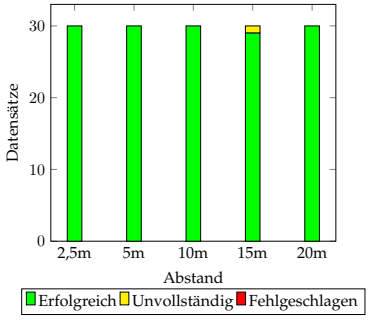
Abbildung 14: Bodenmessungen mit Datensätzen



(a) Messungen mit 2,4 GHz



(b) Messungen mit 866 MHz



(c) Messungen mit 433 MHz

Abbildung 15: Bodenmessungen mit Datenpaketen

In der Wassermessung bei $2{,}40\,\mathrm{GHz}$ Funkfrequenz gingen bis zu den größeren Abständen alle Datensätze erfolgreich ein. Erst bei $15\,\mathrm{m}$ und $20\,\mathrm{m}$ gingen $8\,\%$ der Datensätze nicht beim Empfänger ein. Siehe Abbildung 16 a) für eine grafische Übersicht. Im Vergleich mit der Bodenmessung gingen hier viel mehr Datensätze erfolgreich ein. Auch bei den Datenpaketen sieht das Bild ähnlich aus. Erst bei $20\,\mathrm{m}$ Abstand gingen ein paar Pakete unvollständig ein und einige gingen verloren, siehe Abbildung 17 a). Auch hier gingen im Vergleich zur Bodenmessung mehr Pakete erfolgreich beim Empfänger ein.

Bei $866\,\mathrm{MHz}$ beginnen schon bei kürzester Distanz die ersten Datensätze unvollständig einzugehen, auch gehen dort einige verloren. Wo bei $2{,}50\,\mathrm{m}$ und $10\,\mathrm{m}$ nur ein paar Datensätze unvollständig sind, sind bei $5\,\mathrm{m}$ Abstand ganze $39\,\%$ unvollständig angekommen. Bei den zwei größten Distanzen verhält es sich ähnlich, wo ca. $50\,\%$ Datensätze verloren gingen und ca. $40\,\%$ unvollständig ankamen. Siehe Abbildung 16 b) für eine grafische Darstellung. Hier fällt im Vergleich zur Bodenmessung auf, dass bei der größten Distanz mehr Datensätze unvollständig eingingen und weniger verloren waren. Bei den Datenpaketen verhält es sich wie bei den Datensätzen. Schon von Anfang an gehen einige Pakete unvollständig ein und bei der Distanz von $5\,\mathrm{m}$ gab es ein Tief von $90\,\%$ unvollständig angekommenen Paketen. Bei den größten Distanzen ähnelt der Verlauf der Pakete dem der Datensätze, wo bei beiden ca. $75\,\%$ der Pakete verloren gingen und der Rest unvollständig ankam. Siehe grafische Übersicht in der Abbildung 17 b). Auch hier fällt im Vergleich zur Bodenmessung auf, dass bei der größten Distanz mehr Datenpakete unvollständig eingingen und weniger verloren waren.

In der Funkfrequenz von $433\,\mathrm{MHz}$ verhält es sich wie bei der Bodenmessung. Wo grundsätzlich alle Datenpakte erfolgreich bei allen Abständen ankamen. Jedoch gab hin und wieder einen Aussetzer und ein Datensatz ging verloren oder war unvollständig. Siehe Abbildung 16 c) für eine grafische Darstellung des Datenverlaufs. Auch bei den Datenpaketen verhält es sich gleich, wo wie bei der Bodenmessung fast alle Datenpakete Erfolgreich beim Empfänger ankamen, siehe Abbildung 17 c). In dieser Funkfrequenz ähneln sich die Boden- und Wassermessung.

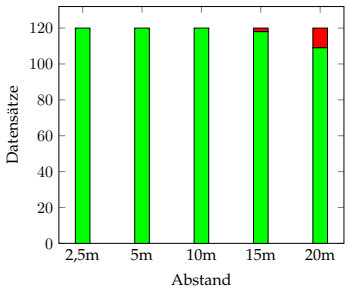
Diese Messreihe zeigte deutlich, das in diesem Experiment die $433\,\mathrm{MHz}$ Funkfrequenz die besten Ergebnisse liefert. Wo, bei der Landmessung, als auch bei der Wassermessung grundsätzlich alle versendeten Daten erfolgreich ankamen. Die $2{,}40\,\mathrm{GHz}$ Funkfrequenz gab das zweitbeste Ergebnis, wo erst bei einer Distanz von $15\,\mathrm{m}$ Daten anfingen verloren zugehen. Auffallend war hier, dass die Wassermessung eine viel bessere Datenübertragung bot, als bei der Bodenmessung. Wo, bei der Bodenmessung, beim Abstand von $20\,\mathrm{m}$ bis zu $60\,\%$ Daten verloren gingen, sind bei der Wassermessung nur $8\,\%$ nicht angekommen. Das schlechteste Ergebnis des Experiments bot die $866\,\mathrm{MHz}$ Funkfrequenz. Bei dieser kamen nur in dem kleinsten Abstand alle Daten erfolgreich an. Schon in der mittleren Distanz waren bis zu $42\,\%$ der Datensätze unvollständig eingegangen und bei den größten Abständen

sind keine Daten erfolgreich angekommen. Bei der Wassermessung fällt im Vergleich zur Bodenmessung auf, dass bei der größten Distanz Daten geschafft haben unvollständig anzukommen.

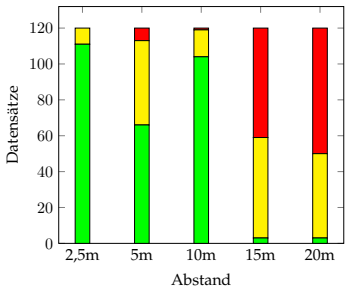
5.5 Aufbau der zweiten Messung

Nach der Analyse der ersten Messung wurde ein Spectrum Analyzer zu diesem Experiment erlangt. Damit kann man verschiedene Funkfrequenzbereiche überwachen, deswegen wurde eine zweite Messung mit gleichzeitiger Verwendung des Spectrum Analyzer durchgeführt. Die zweite Messung erfolgte wie schon bei der ersten Messung mit dem Sensorknoten und dem Empfänger, welche auch wieder in den festen Abständen von $2{,}50\,\mathrm{m}$, $5\,\mathrm{m}$, $10\,\mathrm{m}$, $15\,\mathrm{m}$ und $20\,\mathrm{m}$ aufgestellt wurden. Jedoch wurde diese Messung nur am Boden durchgeführt. Insgesamt wurden 15 Messungen auf dem Boden, im Freien mit fünf verschiedenen Abständen und drei Funkfrequenzen getätigt, siehe Tabelle 12.

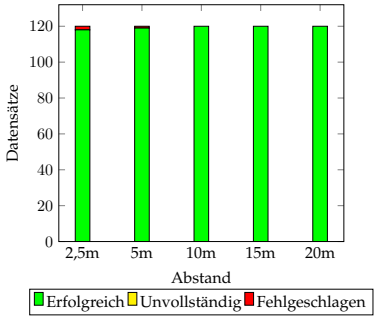
Diese Messung fand im Freien und auf einem Wanderweg im Felde statt. Es bestand freie Sichtlinie zwischen dem Sensorknoten und dem Empfänger. Zusätzlich zum Empfänger wurde ein Spektrum Analyse Gerät eingesetzt, um die zur Übertragung genutzte Funkfrequenz zu überwachen. Dabei wird der „HighEnd EMC Spectrum Analyzer SPECTRAN HF-60105“ von dem Hersteller „Aaronia AG“ hergestellt. Der Spectran hat eine mögliche Frequenzreichweite von $1\,\mathrm{MHz}$ bis zu $9{,}40\,\mathrm{GHz}$, abhängig von der genutzten Antenne. Bei dieser Messung stand eine „OmniLOG 30800“ Antenne zur Verfügung, die einen Bereich von $300\,\mathrm{MHz}$ bis zu $8\,\mathrm{GHz}$ abdeckt, wodurch die genutzten Messungsfrequenzen abgedeckt sind. Der Spectran wurde während des Versuchs über ein USB-Kabel mit dem gleichen Computer wie beim Empfänger verbunden, wo der gemessene Funkfrequenz Verlauf gespeichert wurde.



(a) Messungen mit 2,4 GHz

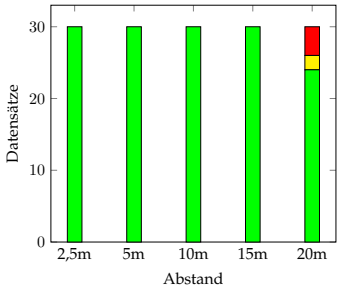


(b) Messungen mit 866 MHz

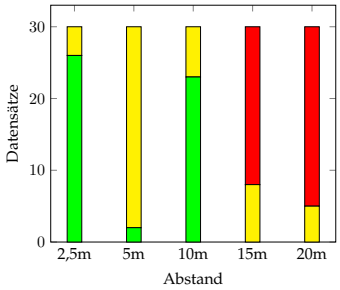


(c) Messungen mit 433 MHz

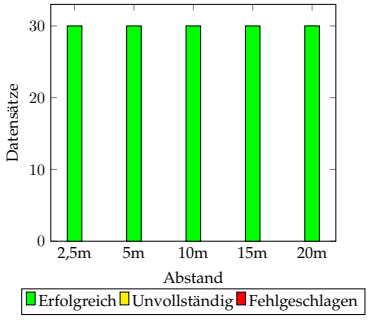
Abbildung 16: Wassermessungen mit Datensätzen



(a) Messungen mit 2,4 GHz



(b) Messungen mit 866 MHz



(c) Messungen mit 433 MHz

Abbildung 17: Wassermessungen mit Datenpaketen

Abbildung 18: Zweite Bodenmessung bei 5 m Abstand

5.6 Ablauf der zweiten Messung

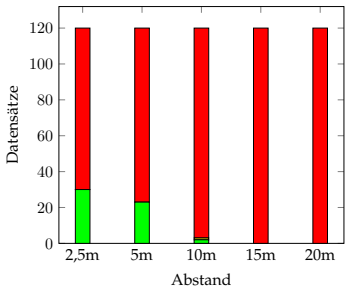
Diese Landmessung fand auf einem Feldweg in der Nähe von Heimbach-Weis statt, die Entfernung zum Ort war ca. $3\,\mathrm{km}$, um Interferenzen bei dem Versuch zu umgehen. Das Wetter zur Zeit des Versuchs war etwas kühler als beim ersten Versuch, es war Windstill und es bestand etwas mehr Luftfeuchtigkeit. Auch hier wurden die nötigen Abstände von $2{,}50\,\mathrm{m}$, $5\,\mathrm{m}$, $10\,\mathrm{m}$, $15\,\mathrm{m}$ und $20\,\mathrm{m}$ gemessen und markiert. Anschließend wurde der Empfänger zusammen mit dem Spectran in Sichtlinie zum Sensorknoten platziert. Dabei war der Boden mit Kieselsteinen versehen, siehe Abbildung 18. Genau wie bei der ersten Messung wurden 120 Datensätze und 30 Datenpakete mit einer Funkfrequenz versandt. Was auch mit allen 3 Frequenzen durchgeführt wurde. Zusätzlich wurden Aufnahmen mit dem Spectran vorgenommen. Eine Aufnahme von ca. 5 Minuten, mit der genutzten Funkfrequenz vor dem Versuch und eine Funkaufnahme für jeden Abstand während des Versuchs. Die erste Aufnahme dient dabei als Vorlage für die dortige Funkfrequenz ohne jegliche Interferenz durch den Sensorknoten. Die Funkaufnahmen während des Versuchs belaufen sich dabei auf ca. 4 Minuten pro Aufnahme. Insgesamt dauerte der Versuch ca. 1 Stunde, hinzu kommt noch die Zeit zum Aufbau des Versuchs.

5.7 Ergebnisse der zweiten Messung

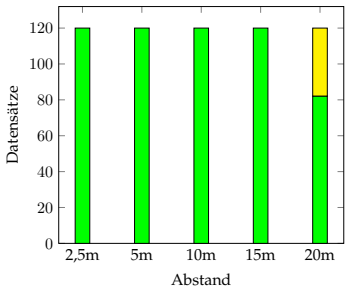
Wie schon bei der ersten Messung wurden für die Auswertung die gespeicherten Daten des Computers genutzt. Hierbei wurden die Sensormessdaten und die Spectran Daten getrennt betrachtet. Wie zuvor wurden die Datensätze und die Datenpakete auf erfolgreichen Empfang und Vollständigkeit überprüft. Weiterhin wurde über die Spectran Daten der Funkfrequenzraum der genutzten Frequenz in CSV-Dateien ausgelesen und verarbeitet. Im Folgenden werden die gewonnenen Werte und Ergebnisse der Messung vorgestellt.

Beginnend mit der Messung bei der $2{,}40\,\mathrm{GHz}$ Funkfrequenz. Bei den gesendeten Datensätzen von der kleinsten bis zur größten Distanz gingen viele Datensätze verloren. Schon bei $2{,}50\,\mathrm{m}$ und $5\,\mathrm{m}$ waren ca. $75\,\%$ verloren gegangen. Ab $10\,\mathrm{m}$ Abstand gingen kaum noch Datensätze bei dem Empfänger ein, siehe Abbildung 19 a). Auffallend hier ist, dass eine große Menge von Daten verlorenen gegangen war, in allen Abständen, im Vergleich zur ersten Messung. Bei den Datenpaketen verhält es sich ähnlich, wo ab $15\,\mathrm{m}$ alle Datenpakte und bei $5\,\mathrm{m}$ ca. $90\,\%$ aller Pakete verloren gegangen waren. Eine Ausnahme bei dieser Bodenmessung ist, dass bei der kürzesten Distanz von $2{,}50\,\mathrm{m}$ jedes einzelne Datenpaket erfolgreich angekommen war. Abbildung 20 a) zeigt dies grafisch. Diese Messergebnisse stehen stark im Kontrast zu den Ergebnissen der ersten Messung, wo erst bei den größten Abständen Daten verloren gingen.

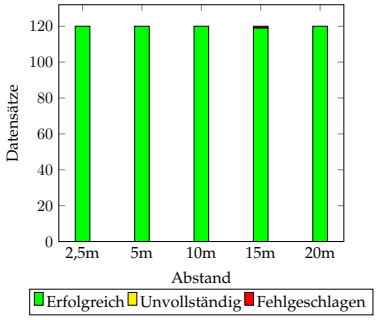
Im Frequenzbereich von $866\,\mathrm{MHz}$ ist das Ergebnis dafür besser, als im $2{,}40\,\mathrm{GHz}$ Bereich. Hier waren alle 120 Datensätze, bis zur größten Distanz, erfolgreich eingegangen. Erst bei $20\,\mathrm{m}$ waren ca. $32\,\%$ unvollständig angekommen. Siehe Abbildung 19 b). Hier fällt auf, dass keine Datensätze verloren gingen und nur sehr wenige unvollständig ankamen, im Gegensatz zu der ersten Messung. Bei den Datenpaketen verhält es sich ähnlich wie bei den Datensätzen. Auch hier gingen keine Datenpakete verloren, nur einige gingen unvollständig ein. Wie zuvor war der meiste Verlust bei der größten Distanz mit ca. $53\,\%$ unvollständige Pakete, jedoch hatte der Abstand von $5\,\mathrm{m}$ zusätzlich eine Unvollständigkeit von $15\,\%$. Siehe Abbildung 20 b). Auch dieses Ergebnis der Messung steht im starken Kontrast zur ersten Messung an Land, wo eine Menge Daten verloren gegangen war oder unvollständig ankam.



(a) Messungen mit 2,4 GHz

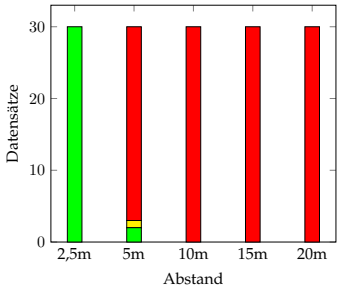


(b) Messungen mit 866 MHz

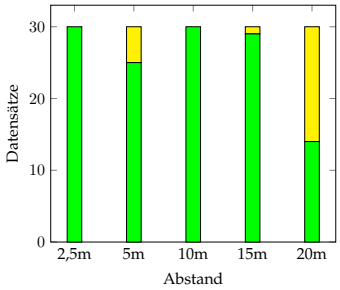


(c) Messungen mit 433 MHz

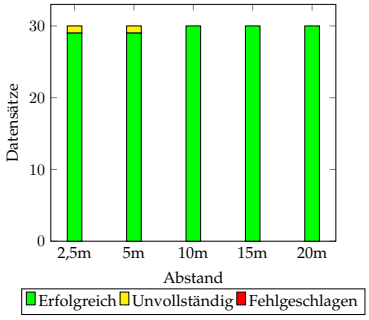
Abbildung 19: Zweite Bodenmessungen mit Datensätzen



(a) Messungen mit 2,4 GHz



(b) Messungen mit 866 MHz



(c) Messungen mit 433 MHz

Abbildung 20: Zweite Bodenmessungen mit Datenpaketen

Bei der Frequenz von 433 MHz verhält es sich wie bei der ersten Messung. Grundsätzlich kommen bei allen Abständen alle verschickten Datensätze an, jedoch gibt es hin und wieder einen Aussetzer, wodurch ein Datensatz verloren oder unvollständig angekommen war. Siehe Abbildung 19 c). Auch bei den Datenpaketen ist es gleich. Alle Datenpakete kommen erfolgreich an, außer ein Paket, das verloren oder unvollständig ankommt. Siehe Abbildung 20 c) auf Seite 50. Im Vergleich zu den ersten beiden Frequenzen sind diese Messergebnisse der 433 MHz Funkfrequenz identisch mit der ersten Boden- und Wassermessung.

Es besteht die Vermutung, dass die neue Umgebung der zweiten Messung, Einfluss auf das Verhalten des 2,40 GHz und 866 MHz Funkkanals ausgeübt hat. Eine mögliche Erklärung ist, dass der Schotter des Bodens Auswirkung auf die Kanäle hatte und die 866 MHz Frequenz dadurch eine bessere Leistung, als die 2,40 GHz Frequenz, erbringen kann.

Die gespeicherten Spectran Frequenzaufnahmen wurden mit der dazu gehörigen Software von der Aaronia AG bearbeitet und als CSV-Dateien gespeichert. Daraus wurden die durchschnittlichen und maximalen Werte der Funkfrequenz ermittelt und als Grafiken dargestellt. Siehe Anhang A für eine Anleitung zur erzeugen der CSV-Dateien.

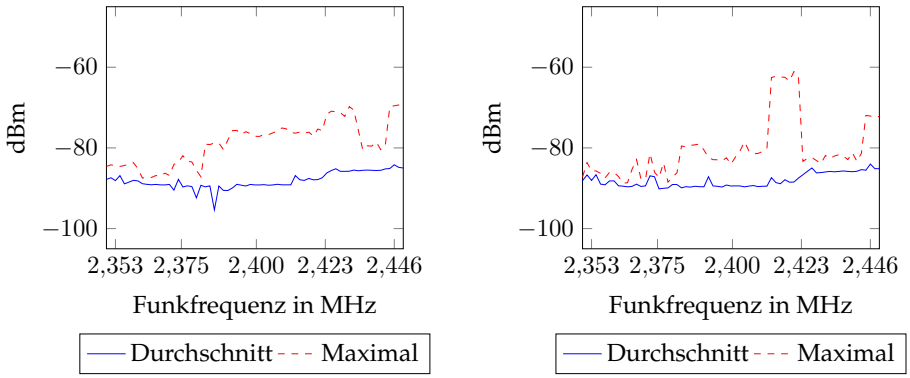
Die Funkfrequenz von 2,40 GHz, vor und während der Messung, sieht man in Abbildung 21. Hier kann man erkennen, dass der durchschnittliche Funkverkehr der Frequenz, vor und während des Versuchs, bei -85 dBm liegt. Außer im 2385 MHz Bereich, wo eine Tiefenspitze von bis zu -95 dBm gibt. In dem Werten vor dem Versuch liegt der niedrigste maximale Wert bei -85 dBm. Ab der 2380 MHz Linie bis zu 2435 MHz steigt der maximale Wert auf ca. -75 dBm. Wo es kurz auf -80 dBm sinkt und bei dem 2445 MHz Bereich wieder auf ca. -75 dBm aufsteigt. Siehe Abbildung 21 a). In den Werten vor dem Versuch sieht man bei der maximalen Linie eine deutliche Spitze bei dem Bereich von 2415 MHz bis 2425 MHz, wo ein Wert von -65 dBm erreicht wird. Danach fällt der Wert auf -85 dBm und steigt bei 2445 MHz auf -75 dBm. Der niedrigste erreichte Wert liegt bei -85 dBm im Bereich von 2350 MHz bis 2380 MHz, siehe Abbildung 21 b). Hier ähnelt sich der maximale Verlauf vor und nach dem Versuch, einzig der große Peak beim Versuch im Bereich von 2415 MHz bis 2425 MHz zeigt, wo die Datensätze und Datenpakete versendet wurden.

Den Verlauf der Funkfrequenz von 866 MHz, vor und während des Versuchs, kann man in Abbildung 22 betrachten. Hier sieht man, dass bei beiden den durchschnittlichen sowie die maximalen Werte bei -75 dBm liegen. Beide haben einen maximalen Ausschlag von -55 dBm im Bereich von 816 MHz bis 825 MHz und einen durchschnittliche Spitze von -65 dBm im gleichen Bereich. Die einzigen Unterschiede zwischen vor und nach dem Versuch sind, dass es beim Versuch noch eine Spitze im Bereich von 912 MHz bis 916 MHz mit -55 dBm gibt und das vor dem Versuch eine Tiefenspitze beim maximalen und durchschnittlichen Verlauf im Bereich von 844 MHz

bis 848 MHz mit -87 dBm erreicht wird. Hier wird nicht bei 100 % Sicherheit erkennbar, wo die Daten des Sensorknotens versendet worden waren. Eine mögliche Erklärung wäre, dass es im Bereich von 844 MHz bis 848 MHz geschah, wo in Abbildung 22 b) keine Tiefenspitze von -87 dBm gab.

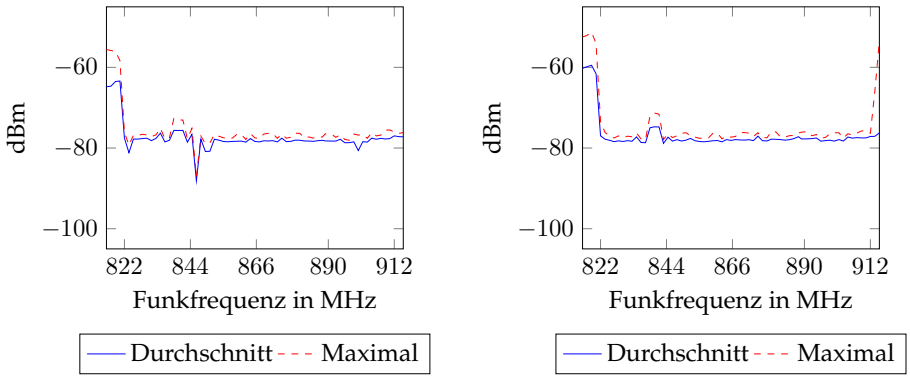
Den Funkkanal von 433 MHz vor und während der Messung kann man in Abbildung 23 betrachten. Hier überlappen sich die durchschnittlichen und maximalen Werte vor dem Versuch bei einem durchschnittlichen Wert von -77 dBm. Selbst die 3 kleinen Maximalen Wertspitzen, im Bereich 470 MHz bis 480 MHz mit einem Wert von -70 dBm, lagen nur etwas höher als die die Durchschnittlichen Spitzen. Das wiederholte sich auch so während des Versuchs, wo beide Linien einen durchschnittlichen Wert von -77 dBm erreichten. Der größte Unterschied liegt im Bereich von 432 MHz bis 435 MHz, wo eine Wertspitze von -50 dBm erreicht wird. Hier kann man deutlich sehen, dass die Daten des Sensorknoten im 433 MHz Bereich versandt worden waren.

Die zweite Messreihe zeigte wieder mal, das die 433 MHz Funkfrequenz das beste Ergebnis liefert, wo alle Daten grundsätzlich beim Empfänger angekommen sind. Ein möglicher Grund für diese ausgezeichnete Leistung ist die im Vergleich große Antenne des Funkmoduls. Die 866 MHz Funkfrequenz gab das zweitbeste Ergebnis. Hier gingen bis zur größten Distanz alle Daten erfolgreich ein. Nur wenige Daten waren unvollständig empfangen worden. Dies stellt einen sehr großen Unterschied zur ersten Messung dar, wo viele Daten unvollständig oder gar nicht angekommen waren. Das schlechteste Ergebnis lieferten die 2,40 GHz Funkfrequenz. Hier gingen in allen Abständen eine große Menge Daten verloren. Dies steht im starken Kontrast zur ersten Messung, wo 2,40 GHz Funkfrequenz ein besseres Ergebnis lieferte. Eine mögliche Erklärung für das Verhalten im 866 MHz und 2,40 GHz Funkfrequenzbereich währe, dass der neue Versuchsort der zweiten Messung, sich auf das Experiment ausgewirkt hat. Hier währen weitere Messungen an verschiedenen Orten nötig, um sich auf eine definitive Ursache festlegen zu können. Eine weitere Auffälligkeit der Messreihe war das Ergebnis bei dem Spectran, wo bei der 866 MHz Funkfrequenz nicht erkennbar war, wann die Daten des Prototyps verschickt wurden. Auch hier kann man sich auf keine definitive Ursache festlegen. Mögliche Gründe währen, dass die Antenne nicht sensibel genug im 866 MHz Bereich ist oder es Interferenzen gab, die aber nicht in den Daten abgebildet wurde.



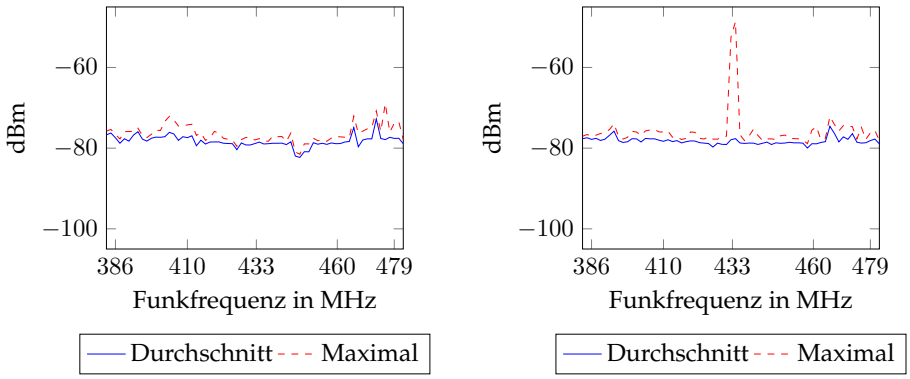
(a) Frequenz vor dem Versuch **(b)** Frequenz während des Versuchs

Abbildung 21: Zweite Bodenmessungen im Funkrequenzenbereich 2400 MHz



(a) Frequenz vor dem Versuch **(b)** Frequenz während des Versuchs

Abbildung 22: Zweite Bodenmessungen im Funkrequenzenbereich 866 MHz



(a) Frequenz vor dem Versuch **(b)** Frequenz während des Versuchs

Abbildung 23: Zweite Bodenmessungen im Funkrequenzenbereich 433 MHz

6 Fazit und Ausblick

In dieser Arbeit wird untersucht, ob ein Sensorknoten auf Arduino Basis erstellt werden kann, der eine vernünftige Alternative zum Sensorknoten Prototyp auf Basis vom Tmote Sky darstellt, dabei sollte er alle Funktionen des Prototypen aus [3] erfüllen und zudem möglichst Kostengünstiger sein. Der entwickelte Sensorknoten Prototyp ist in der Lage durch Messinstrumente die Umgebungstemperatur zu messen und er ist fähig durch GPS seine Position zu bestimmen. Außerdem ist es ihm möglich, erhobene Daten auf einer SD-Karte zwischenzuspeichern, bevor er diese über Funk an einen Empfänger weiter sendet. Dabei stehen ihm drei verschiedene Funkmodule zur Verfügung, die die Funkfrequenz von $2{,}40\,\mathrm{GHz}$, $866\,\mathrm{MHz}$ und $433\,\mathrm{MHz}$ abdecken. Zusätzliche können diese Funkmodule einfach ausgetauscht werden, ohne Änderung an der Software des Sensorknotens durchzuführen. Diese Modularität ist ein herausragender Vorteil zu der Verwendung dieses Sensorknotens. Die gemessenen Daten werden dabei im Sensorknoten selbst und an dem am Empfänger hängenden PC protokolliert. Die Kosten des ersten Sensorknotenprototyps belaufen sich dabei auf 130 €. Der zweite verwendete Sensorknotenprototyp kostet insgesamt 170 €, wobei 80 € für die Basisplattform anfallen und 90 € für alle 3 Funkmodule.

Die ausgewerteten Daten der Messungen mit dem Sensorknotenprototyp zeigen ein nicht eindeutiges Bild der verwendeten Funkfrequenzen. Hier wechselten sich die Übertragungsfähigkeiten der $2{,}40\,\mathrm{GHz}$ und $866\,\mathrm{MHz}$ Funkfrequenzen, abhängig von Messstandort. Bei $2{,}40\,\mathrm{GHz}$ war sie gut und bei 866 MHz eher schlecht, nach einem Ortswechsel änderte sich dies enorm. Einzig die Funkfrequenz von $433\,\mathrm{MHz}$ bot eine konstant gute Übertragungsfähigkeit bei allen durchgeführten Messungen. Mögliche Gründe hier wären die verschiedenen Antennengrößen der Funkmodule, die bei $433\,\mathrm{MHz}$ mit ca. $15\,\mathrm{cm}$ deutlich über der anderen liegt. Hier sind weitere Messungen nötig, um alle Faktoren der Übertragungsfähigkeit des Sensorknotens zu erforschen und verbessern zu können.

Die gestellten Anforderungen dieser Arbeit konnten mit dem Arduino Sensorknoten Prototyp grundsätzlich erfüllt werden. Jedoch stellten sich bei der Entwicklung und der Implementierung des Prototyps einige Kritikpunkte zur Verbesserung dar. Ein Kritikpunkt ist die Größe des Speichers des Arduino Uno. Hierauf stieß man bei der Erstellung des ersten Prototyps, da schon mit einer $75\,\%$ Belegung des SRAMS der Betrieb des Arduino Uno instabil wird. Dieses Problem konnte zwar durch Änderungen an Programmbibliotheken vermindert werden, konnte aber erst durch das Wechseln auf ein anderes, teureres Funkmodul beseitigt werden. Eine weitere Möglichkeit dieses Problem zu umgehen und die Modularität des Arduino zu erhalten, wäre es eine stärkere Version des Arduino zu nehmen, die mehr Speicher und andere Verbesserung bietet, wie z. B. der Arduino Mega. Der Nachteil dieses Vorgehens wäre lediglich die erhöhten Kosten

des Arduino; die ausgewählte Hardware bliebe kompatibel. Jedoch müsste der Code an die neue Hardware angepasst werden. Eine radikalere Option währe es einen vollkommen anderen Mikroprozessor zu nehmen, wie den Raspberry Pi, welcher aber nicht die Modularität des Arduino bieten kann.

Ein weiterer Kritikpunkt ist, dass die gängigen Kenngrößen RSSI und LQI nicht ausgelesen werden können. Dieses Problem entsteht durch die drei verwendeten Funkmodule, da jedes Modul einen eigenen Code bräuchte, um diese Werte auslesen zu können. Durch den limitierten Speicherplatz war dies in dem erstellten Sensorprototyp nicht möglich. Auch hier wäre bessere Hardware eine Möglichkeit das Problem zu lösen.

Den letzten Kritikpunkt stellen die Kosten des Sensorknotens dar. Die Kosten des Sensorknotens aus der Masterarbeit [3] beläuft sich auf 118 €, in dieser Arbeit wurde angestrebt, unter diesem Wert zu liegen. Mit den Kosten für die Grundplattform mit 80 € liegt man noch gut darunter. Fügt man noch ein Funkmodul von $2{,}40\,\mathrm{GHz}$ mit ein, kann man mit derselben Funkfrequenz funken. Dabei liegt man dann bei einem Wert von 105 €, welcher vergleichbar mit den angestrebten Sensorknoten ist. Jedoch kostet die volle angestrebte Flexibilität, mit verschiedenen Funkfrequenzen, einiges mehr und der Endpreis des vollen Prototyps liegt bei 170 €. Ein Aufpreis von $44\,\%$ gegenüber dem in [3]. Eine Möglichkeit dieses Problem anzugehen, wäre es, nicht teure XBee Funkmodule zu verwenden, sondern auf billigere Funkmodule zu wechseln, die jedoch extra Code benötigen. Dies bedeutet wiederum, dass man eine stärkere Arduino Version benötigt.

Mit der Erfahrung, die im Laufe dieser Arbeit gemacht wurde, kann man die Aussage treffen, dass der Arduino Uno ungeeignet erscheint, um einen komplexeren Sensorknoten Prototyp zu erstellen, da man kaum Freiheiten hat, um am Code des Programms zu arbeiten, bis man schnell auf den Speichermangel des Arduino Uno stößt. Jedoch erfüllte der hier erstellte Sensorknoten Prototyp, trotz Schwierigkeiten, die geforderten Anforderungen und besitzt noch Potenzial, das weiter ausgeschöpft werden kann. Des Weiteren gibt es noch mehr Arduino Versionen die eine stärkere Hardware bieten und dadurch diese Probleme umgehen. Dabei gilt es Vor- und Nachteile abzuwägen. Die starke Modularität des Arduinos, zum Beispiel die XBee kompatiblen Funkmodule und die dadurch entstehenden höheren Kosten. Andererseits nimmt man weniger Modularität in Kauf, in dem man kostengünstigere Module nimmt, aber dafür mehr Programmspeicher braucht. Weiterhin kann man die Kosten weiter senken, in den man z.B. Shield-Module selber erstellt, welches nur etwas Handwerksgeschick erfordert. Die Arduino Modularität bietet noch viel Potenzial für einen Sensorknoten, das noch weiter ausgereizt werden kann. Von selbst erstellten Modulen zu neu veröffentlichter Hardware bestehen noch eine Menge Möglichkeiten das Konzept des Sensorknotens auf Arduino Basis weiter zu untersuchen.

Literatur

[1] Nicolas Steeb. Fliessgewässer-Monitoring im Kanton Thurgau, Koordinierte Anwendung des Modul-Stufen-Konzepts für ein flächendeckendes Messnetz. Master's thesis, Eidgenössische Technische Hochschule Zürich (ETHZ), 2014.

[2] Amir Yousefi, Ansgar Taflinski, Florentin Neumann, and Hannes Frey. Sensorknoten Netzwerk für In Situ-Temperaturerhebung von Wasseroberflächen in Fließgewässern. Technical report, Universität Koblenz-Landau, 2014.

[3] Ansgar Taflinski. Fließgewässermonitoring mit drahtlosen Sensornetzen. Master's thesis, Universität Koblenz-Landau, 2015.

[4] Piyare Rajeev and Seong Ro Lee. Towards Internet of Things (IOTS): Integration of Wireless Sensor Network to Cloud Services for Data Collection and Sharing. In *International Journal of Computer Networks and Communications Vol.5, No.5*, pages 59–72, 2013.

[5] Luis Ostiz Urdiain, Carlos Pita Romero, Jeroen Doggen, Tim Dams, and Patrick Van Houtven. Wireless Sensor Network Protocol for Smart Parking Application Experimental Study on the Arduino Platform. In *2nd Int. Conf. Ambient Computing, Application., Services and Technologies.-AMBIENT'12*, pages 45–48, 2012.

[6] João Valente, David Sanz, Antonio Barrientos, Jaime del Cerro, Ángela Ribeiro, and Claudio Rossi. An Air-Ground Wireless Sensor Network for Crop Monitoring. In *Sensors 2011*, pages 6088–6108, 2011.

[7] A. Abdullah, O. Sidek, N.A. Amran, U.N. Za'bah, F. Nikmat, Hadi Jafar, and Munajat Abdul Hadi. Development of Wireless Sensor Network for Monitoring Global Warming. In *Advanced Computer Science and Information Systems ICACSIS*, 2012.

[8] S. A. Mitilineos, D. M. Kyriazanos, O. E. Segou, J. N. Goufas, and S. C. A. Thomopoulos. Indoor Localization with Wireless Sensor Networks. In *Progress In Electromagnetics Research Vol. 109*, pages 441–474, 2010.

[9] Vongsagon Boonsawat, Jurarat Ekchamanonta, Kulwadee Bumrungkhet, and Somsak Kittipiyakul. XBee Wireless Sensor Networks for Temperature Monitoring. In *2nd ECTI-Conference on Application Research and Development (ECTI-CARD 2010)*, 2010.

[10] Christos Tatsiopoulos and Aphrodite Ktena. A Smart ZIGBEE Based Wireless Sensor Meter System. In *Systems, Signals and Image Processing, IWSSIP*, 2009.

[11] Cholatip Yawut and Sathapath Kilaso. A Wireless Sensor Network for Weather and Disaster Alarm Systems. In *International Conference on Information and Electronics Engineering IPCSIT Vol.6*, 2011.

[12] N.S.A. Zulkifli, F.K.Che Harun, and N.S. Azahar. Centralized Heart Rate Monitoring Telemetry System Using ZigBee Wireless Sensor Network. In *Proceedings of the IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI 2012)*, 2012.

[13] Hermawan Kemis, Ndibanje Bruce, Wang Ping, Tony Antonio, Lee Byung Gook, and Hoon Jae Lee. Healthcare Monitoring Application in Ubiquitous Sensor Network: Design and Implementation based on Pulse Sensor with Arduino. In *Information Science and Service Science and Data Mining (ISSDM)*, 2012.

[14] Amir Hoshang Kioumars and Dr. Liqiong Tang. Wireless Network for Health Monitoring: Heart Rate and Temperature Sensor. In *Fifth International Conference on Sensing Technology*, 2011.

[15] David Kushner. The Making of Arduino. `http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino`, 2011. Eingesehen am 09.01.2017.

[16] Arduino LLC. Credits. `https://www.arduino.cc/en/Main/Credits`. Eingesehen am 09.01.2017.

[17] Atmel Corporation. Atmega328. `http://www.atmel.com/devices/atmega328.aspx?tab=parameters`. Eingesehen am 09.01.2017.

[18] Arduino LLC. Arduino/Genuino UNO. `https://www.arduino.cc/en/Main/ArduinoBoardUno`. Eingesehen am 09.01.2017.

[19] Arduino LLC. Arduino PRO. `https://www.arduino.cc/en/Main/ArduinoBoardPro`. Eingesehen am 09.01.2017.

[20] Arduino LLC. Arduino Pro Mini. `https://www.arduino.cc/en/Main/ArduinoBoardProMini`. Eingesehen am 09.01.2017.

[21] Arduino LLC. Arduino Nano. `https://www.arduino.cc/en/Main/ArduinoBoardNano`. Eingesehen am 09.01.2017.

[22] Seeed Studio. Seeeduino v4.2. `http://wiki.seeed.cc/Seeeduino_v4.2/`. Eingesehen am 09.01.2017.

[23] Wikimedia Commons. Bild Arduino Uno. `https://upload.wikimedia.org/wikipedia/commons/a/a8/ArduinoUno_R3_Front_450px.jpg`. Eingesehen am 09.01.2017.

[24] Wikimedia Commons. Bild Arduino Pro. https://upload.wikimedia.org/wikipedia/commons/a/af/Arduino_Pro.jpg. Eingesehen am 09.01.2017.

[25] Wikimedia Commons. Bild Arduino Pro Mini. https://upload.wikimedia.org/wikipedia/commons/0/01/Arduino_Pro_Mini_%282%29.jpg. Eingesehen am 09.01.2017.

[26] Wikimedia Commons. Bild Arduino Nano. https://upload.wikimedia.org/wikipedia/commons/8/8d/Arduino_Nano.jpg. Eingesehen am 09.01.2017.

[27] Atmel Corporation. Atmega32u4. http://www.atmel.com/devices/atmega32u4.aspx?tab=parameters. Eingesehen am 09.01.2017.

[28] Arduino LLC. Arduino Leonardo. https://www.arduino.cc/en/Main/ArduinoBoardLeonardo. Eingesehen am 09.01.2017.

[29] Arduino LLC. Arduino MICRO and Genuino MICRO. https://www.arduino.cc/en/Main/ArduinoBoardMicro. Eingesehen am 09.01.2017.

[30] SparkFun. Pro Micro - 5V/16MHz. https://www.sparkfun.com/products/12640. Eingesehen am 09.01.2017.

[31] SparkFun. Pro Micro - 3.3V/8MHz. https://www.sparkfun.com/products/12587. Eingesehen am 09.01.2017.

[32] Arduino LLC. Arduino Mega. https://www.arduino.cc/en/Main/ArduinoBoardMega. Eingesehen am 09.01.2017.

[33] Arduino LLC. Arduino Due. https://www.arduino.cc/en/Main/ArduinoBoardDue. Eingesehen am 09.01.2017.

[34] Wikimedia Commons. Bild Arduino Leonardo. https://upload.wikimedia.org/wikipedia/commons/3/3d/Arduino_Leonardo_board_%28cropped%29.JPG. Eingesehen am 09.01.2017.

[35] Wikimedia Commons. Bild Arduino Micro. https://upload.wikimedia.org/wikipedia/commons/e/e6/Arduino_Micro.jpg. Eingesehen am 09.01.2017.

[36] Wikimedia Commons. Bild Arduino Mega. https://upload.wikimedia.org/wikipedia/commons/0/01/Arduino_Mega.jpg. Eingesehen am 09.01.2017.

[37] Wikimedia Commons. Bild Arduino Due. https://upload.wikimedia.org/wikipedia/commons/3/36/ArduinoDue_Front.jpg. Eingesehen am 09.01.2017.

[38] Arduino LLC. Arduino Wireless SD Shield. https://www.arduino.cc/en/Main/ArduinoWirelessShield. Eingesehen am 09.01.2017.

[39] SparkFun. SparkFun XBee Shield. https://www.sparkfun.com/products/12847. Eingesehen am 09.01.2017.

[40] SparkFun. SparkFun RFM22 Shield - 434MHz. https://www.sparkfun.com/products/retired/11018. Eingesehen am 09.01.2017.

[41] SparkFun. SparkFun Cellular Shield - MG2639. https://www.sparkfun.com/products/13120. Eingesehen am 09.01.2017.

[42] SparkFun. SparkFun GPS Shield.

[43] SparkFun. SparkFun GPS Logger Shield. https://www.sparkfun.com/products/13750. Eingesehen am 09.01.2017.

[44] SparkFun. SparkFun Weather Shield. https://www.sparkfun.com/products/12081. Eingesehen am 09.01.2017.

[45] LLC Adafruit Industries. 10K Precision Epoxy Thermistor. https://www.adafruit.com/products/372. Eingesehen am 09.01.2017.

[46] Seeed Studio. GPS Bee kit. http://wiki.seeedstudio.com/wiki/GPS_Bee_kit. Eingesehen am 09.01.2017.

[47] Kingston. microSDHC Klasse. https://www.kingston.com/datasheets/SDC4_de.pdf. Eingesehen am 09.01.2017.

[48] SparkFun. XBee / XBee-PRO RF Modules. https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf. Eingesehen am 09.01.2017.

[49] Seeed Studio. RFbee V1.1 - Wireless Arduino compatible node. http://wiki.seeed.cc/RFbee_V1.1-Wireless_Arduino_compatible_node/. Eingesehen am 09.01.2017.

[50] Inc. Amazon.com. COM-FOUR® XY-MK-5V / XY-FST 433 MHz Funk - Sende und Empfänger Modul Set für Raspberry und Arduino Wireless Transmitter Receiver (4 Stück). https://www.amazon.de/COM-FOUR%C2%AE-XY-MK-5V-XY-FST-433-Funk/dp/B00W74GC40. Eingesehen am 09.01.2017.

[51] Stanley R. Steinhart, John S. Hart. Calibration curves for thermistors. In *Deep Sea Research and Oceanographic Abstracts Volume 15, Issue 4*. Pergamon Press, 1968.

[52] In-Circuit. radino XBee-PRO Shield, adapter board for radino/32 with XBee footprint. `http://shop.in-circuit.de/product_info.php?products_id=182`. Eingesehen am 09.01.2017.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BASIC Beginners All-purpose Symbolic Instruction Code

Breadboard Lochrasterplatinne

CSV Comma Separated Values

DAC Digital-To-Analog Converter

dBm Dezibel Milliwatt

DC direct current

DIP Dual Inline Package

EEPROM Electrically Erasable Programmable Read Only Memory

FSK Frequency Shift Keying

GHz GigaHertz

GPS Global Positioning System

I/O Input/Outpu

I2C Inter-Integrated Circuit

ICSP In-Circuit Serial Programming

ID IDentification

IDE Integrated Development Environment

IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrial, Scientific and Medical
JTAG	Joint Test Action Group
kB	Kilobyte
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LQI	Link Quality Indicator
MHz	MegaHertz
NTC	Negative Temperature Coefficient
PWM	Pulse-Width Modulation
RAM	Random Access Memory
REST	REpresentational State Transfer Architektur
RSSI	Received Signal Strength Indication
SD	Secure Digital
SDHC	Secure Digital High Capacity
SMD	Surface-Mounted Device
SMS	Short Message Service
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TCP	Transmission Control Protocol
TTL	Transistor-Transistor-Logik
TWI	Two Wire Interface
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Data Protocol
USB	Universal Serial Bus
WSN	Wireless Sensor Network

A Spectran CSV-Datei Erzeugung

Hier folgt eine Einleitung um mit dem „MCS Spectrum Analyzer“ Programm des Spectrans CSV-Dateien zuerzeugen. Siehe Anhang A.

- 1. Öffne Programm „MCS Spectrum Analyzer“
- 2. Messungen -> Öffne Messdatei
- 3. Wähle Aufgezeichnete Messdatei aus, Format .mdr (MCS Dateiaufzeichnung), öffne diese
- 4. Bei der Frage „Pause zwischen Sweeps (in ms)“ auf „cancel“ drücken
- 5. Messungen -> “Aufzeichnung starten“
- 6. Wähle als Dateityp „CSV TAbelle (*csv)“ aus und gib einen Namen ein -> Speichern
- 7. Drücke Button „Wiedergabe pausieren“, nun sollte die Aufzeichnung ablaufen und in die CSV-Datei geschrieben werden
- 8. Messungen -> „Aufzeichnung stoppen“
- 9. Gespeicherte CSV-Datei kann nun bearbeitet werden.