Diplomarbeit

Garden Monitoring

Schule	TEKO Schweizerische Fachschule Bern
Fachrichtung	Informatik Systemtechnik
Klasse	B-TIS-17-T-a
Diplomand	Timo Fankhauser
Ort, Datum	Lanzenhäusern, 02.11.2020

Informationen zu dieser Arbeit

Version	Datum	Autor	Änderung
0.1	18.09.2020	Timo Fankhauser	Erstellung des Dokuments
0.2	20.09.2020	Timo Fankhauser	Vorstudie erstellen
0.3	24.09.2020	Timo Fankhauser	Grobkonzept erstellen
0.4	26.09.2020	Timo Fankhauser	Detailkonzept erstellen
0.5	15.10.2020	Timo Fankhauser	Realisierung erstellen
0.6	28.10.2020	Timo Fankhauser	Ergänzungen
1.0	01.11.2020	Timo Fankhauser	Fertigstellen des Dokuments

Tabelle 1 Informationen zur Dokumentation

Beteiligte	Timo Fankhauser
Benutzer	Familie Fankhauser
Prüfung	Timo Fankhauser
Diplomlehrer	Jörg Schenker
Experte	Patrick Graber

Auftraggeber	Familie Fankhauser
Projektname	Garden Monitoring
Autor	Timo Fankhauser
Version	1.0

Management Summary

Im Rahmen dieser Diplomarbeit soll ein Überwachungssystem entstehen, welches dem Auftraggeber die Gartenarbeit erleichtern soll. Die Gartenarbeit ist anstrengend und nimmt viel Zeit in Anspruch, ausserdem stellt sich immer wieder die Frage was die optimalen Bedingungen für den Pflanzenwachstum sind. Jedes Gemüse hat andere Ansprüche, wie z.B. den Standort, die benachbarte Pflanze sowie die korrekte Bewässerung.

Die Zielsetzung der Arbeit Garden Monitoring ist es, den Aufbau und Inbetriebnahme des Überwachungssystems aufzuzeigen. Die Konfiguration des Systems lassen sich bedürfnisorientiert auf den Endbenutzer einstellen. Das Resultat dieser Diplomarbeit ist ein Grundlagendokument, welches in einem privaten Gemüsegarten eingesetzt werden kann.

Während des Studiums habe ich verschiedene technologiebasierte Lösungsmöglichkeiten kennengelernt, die ich nun bei diesem Projekt in die Praxis umsetzen kann. Durch meine Vorkenntnisse habe ich mich entschieden die Aufgabenstellungen mit Hilfe eines Mikrocontrollers und diversen Sensoren (Temperatursensor, Feuchtigkeitssensor, Bodenfeuchtigkeitssensor, Regensensor) umzusetzen. Mit der LoRa Netzwerktechnik werden die Daten in die Wohnung des Kunden übermittelt und auf einem Touchscreen Display angezeigt.

Inhaltsverzeichnis

Abbildverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

µC	Micro Controller
HMI	Human-Machine Interface
LED	Light-Emitting Diode
LoRa	Long Range
UART	Universal asynchronous receiver-transmitter
RSSI	Received Signal Strength Indicator

Glossar

HMI

Ein HMI ist ein Gerät oder eine Software, welches resp. welche dem Benutzer die Kommunikation mit Maschinen oder Produktionsanlagen ermöglicht.

LED

Die Abkürzung LED steht für Licht emittierende Diode und bezeichnet ein elektronisches Halbleiter-Bauteil, welches leuchtet sobald Strom hindurchfließt.

UART Schnittstelle

Der UART ist die gängige serielle Schnittstelle an PCs und Mikrocontrollern. Es können Dateien in Wörter übertragen werden von 5 – 9 Bit. Am gängigsten sind 8 – 9 Bit, das sind auch von Mikrocontrollern unterstützenden Modi.

RSSI

Beim RSSI handelt es sich um den Empfangsfeldstärke-Anzeiger einer Mobilstation. Der RSSI-Wert ist ein wichtiger Indikator für den Funkkanal und die empfangene Feldstärke. Der RSSI-Indikator ist ein Parameter der die gesamte empfangene Signalstärke für die Funkkanalbreite beinhaltet. Er ist abhängig von der Sendeleistung (EIRP) der Funkstation und der Entfernung zwischen Funkstation und Mobilstation.

Diplomand

Personalien

Name	Fankhauser
Vorname	Timo
Adresse	Steinenbrünnen 18
	3148 Lanzenhäusern
Mobile	+41 79 458 67 72
E-Mail	timo.fankhauser@gmail.com
Heimatort	Trub BE
Zivilstand	verheiratet
Kinder	Tochter Emma, 4 Monate

Abbildung 0-1 Foto Diplomand

Beruflicher Lebenslauf

Elektromonteur EFZ

Im Sommer 2005 begann ich die Ausbildung als Elektromonteur EFZ, die ich 2009 erfolgreich abschloss. Nach dem Militärdienst nahm ich meine Arbeit bei meinem Ausbildungsbetrieb Muff & Schmutz AG mit einer Festanstellung wieder auf. Ich war an diversen Projekten beteiligt. Zudem war ich der bauleitende Monteur einer grossen Überbauung.

Techniker Kundendienst Alarmsysteme

Von August 2013 bis Dezember 2016 arbeitete ich bei der Securiton AG als Techniker im Kundendienst der Geschäftsstelle Bern. Seit Januar 2017 bin ich nun als Fachtechniker bei der Securiton AG angestellt. Zu meinen täglichen Aufgaben gehört u.a. das Ausführen von Instandhaltungs- und Instandsetzungsarbeiten an den Brand-, Einbruch- und Überfallmelde-anlagen sowie Anpassungs- und Erweiterungsarbeiten. Zudem erledige ich Einsätze im Störungsdienst während den Geschäftszeiten und führe Inspektions- und Wartungsarbeiten durch. Weiter leiste ich regelmässig telefonischen Support sowie Pikettdienst innerhalb der Pikettorganisation.

1. Initialisierung

1.1. Ausgangslage

Im Rahmen der Diplomarbeit soll ein Überwachungssystem für einen privaten Gemüsegarten entstehen. Es wird ein abgesetzter Sender im Garten installiert, welcher alle Sensorwerte (Temperatur, Luftfeuchtigkeit, Bodenfeuchtigkeit; Regen) detektiert und an den Empfänger in der Wohnung übermittelt. Der Sender soll wetterbeständig gebaut werden und optional mit einer eigenen Stromquelle versorgt werden. Der Empfänger wird mit einem Display versehen, auf welchem die verschiedenen Werte angezeigt werden. Bei zu niedriger Bodenfeuchtigkeit wird ein optischer Alarm angezeigt. Zusätzlich besteht die Option ein Bewässerungssystem zu integrieren und den Verbindungsstatus anhand einer LED anzuzeigen.

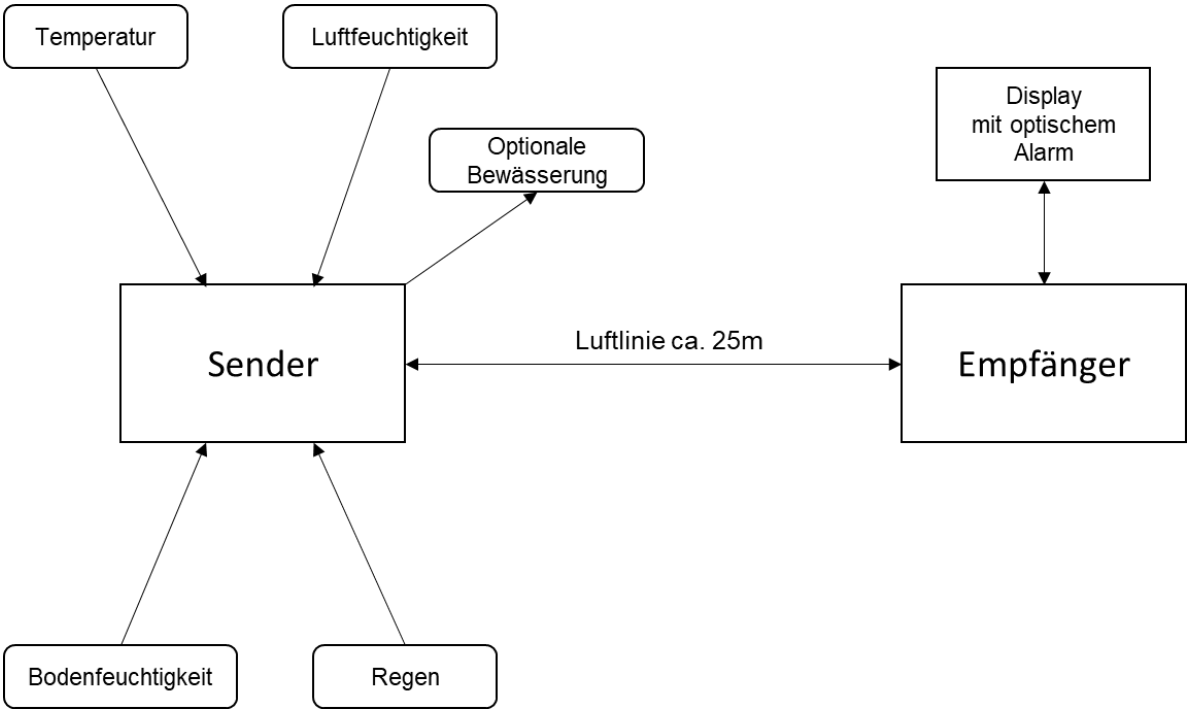


Abbildung 1-1 Blockdiagramm Garden Monitoring

1.2. Projektorganisation

Das Projekt wird von Timo Fankhauser erstellt, ausgeführt und getestet.
Die Gesamtverantwortung liegt beim Projektleiter und Jörg Schenker wird als Diplomlehrer das Projekt begleiten. Weiter wird Patrick Graber an der Präsentation als Experte teilnehmen.

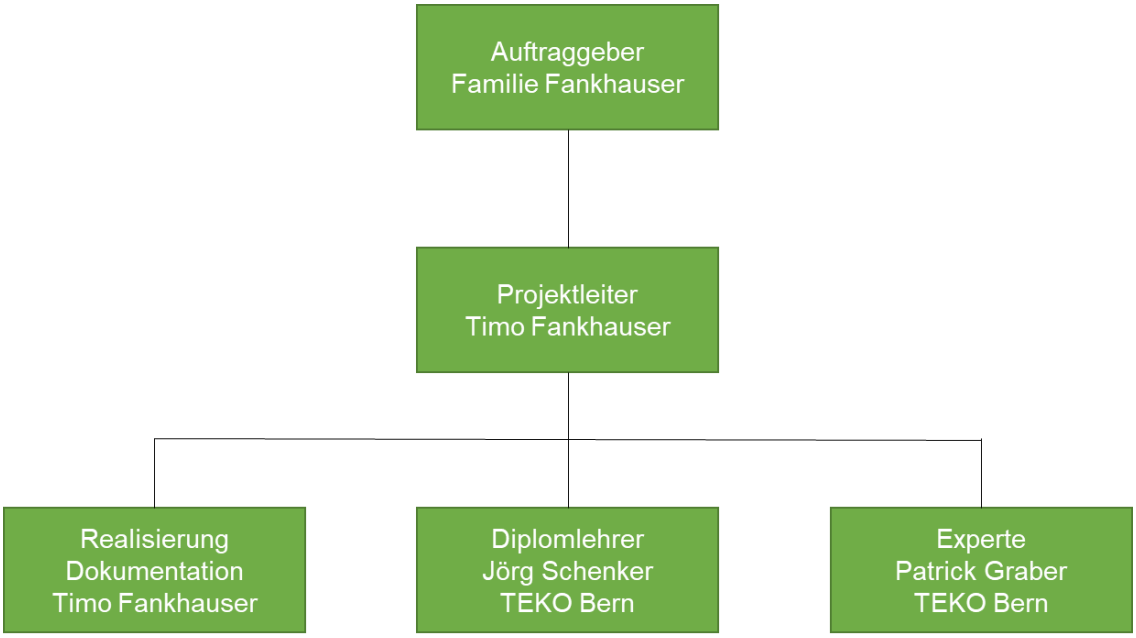


Abbildung 1-2 Projektorganisation

1.3. Projektstruktur

In dieser Diplomarbeit wird das Basis-Phasenkonzept angewendet. Dieses Konzept wurde im Kurs Projektmanagement an der TEKO unterrichtet und soll eine optimale Grundlage für die Diplomarbeit liefern. Die letzten zwei Phasen Einführung und Erhaltung sind kein Teil dieser Diplomarbeit.

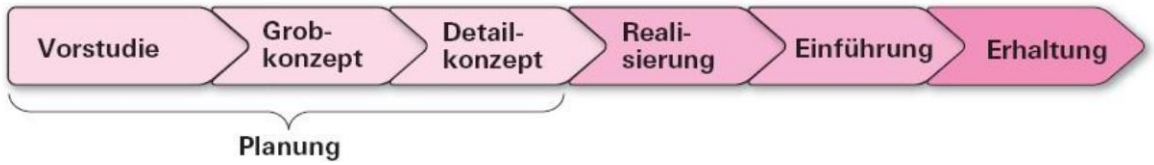


Abbildung 1-3 Basis-Phasenkonzept

Themen der Phasen

Vorstudie

- Projektfindung
- Entscheid
- LoRa
- µC

Grobkonzept

- Auswahl Hardware µC / LoRa
- Entscheid
- Auswahl Hardware HMI
- Verbindungstest

Detailkonzept

- Pheripherie
- Umsetzung und Funktionen

Realisierung

- Funktionen
- HMI
- Funktionen mit HMI
- Optischer Alarm

- Zusätzliche Optionen
- Endprodukt
- Test mit Endprodukt

1.4. Projektziele

Die Ziele der Diplomarbeit werden in drei Teile aufgeteilt. Der erste Teil beinhaltet die persönlichen Ziele des Diplomanden, der zweite Teil die Ziele des Auftraggebers und im dritten Teil werden die zusätzlichen Ziele beschrieben.

1.4.1 Persönliche Ziele

- Know-how µC vertiefen
- Know-how LoRa vertiefen
- Know-how Programmierung vertiefen
- Zufriedenstellung Auftraggeber

1.4.2 Muss Ziele Auftraggeber

- Gewünschte Funktionen umsetzen
- Das Endprodukt ist wettertauglich
- Stabile und Störungsfreie Kommunikation zwischen Sender und Empfänger
- Optische Alarmierung bei zu geringer Bodenfeuchtigkeit
- Kosten unter 400.- halten

1.4.3 Kann Ziele

- Sender wird mit eigener Stromquelle versorgt
- Relaisschaltung für Bewässerung EIN/AUS
- Statusanzeige Verbindungsüberwachung durch LED

1.5. Projektabgrenzung

Die Diplomarbeit beinhaltet einen Sender- und einen Empfängerprototypen, die mit einfachen Mitteln montiert werden. Der Sender wird mit einer Kabeldose und einem Holzpfahl installiert. In einem alten, kleinen Koffer werden der Empfänger und das HMI Display verdrahtet und installiert. Ausserdem wird die optionale Bewässerung mit einem Relaismodul realisiert, welches nur die Schaltung vornimmt. Die Bewässerung selbst ist kein Bestandteil der Diplomarbeit.

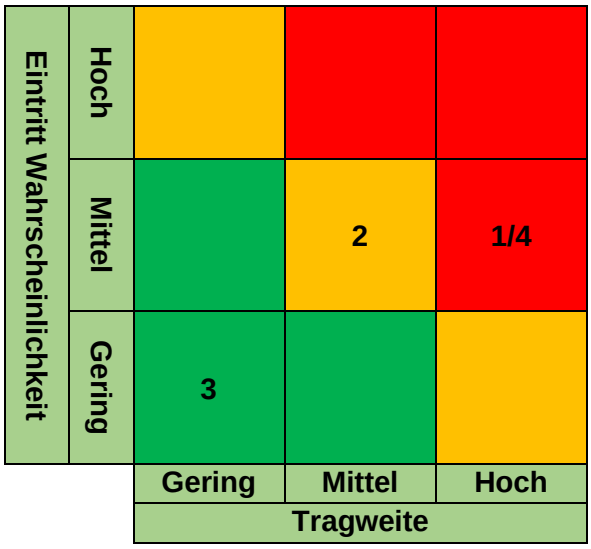
1.6. Projektrisiken

Es bestehen keine grossen Risiken bei diesem Projekt, da der Teamleiter nicht auf andere Personen angewiesen ist.

1.6.1 Risiken

Nr.	Risiko	Definition
1	Erfahrung	Fehlende Erfahrung mit µC, LoRa Funktechnologie und HMI
2	Zeitaufwand	Der Bau der Prototypen nimmt viel Zeit in Anspruch
3	Kosten	Die Kosten von CHF 400.00 werden überschritten
4	Programmierung	Fehelende Erfahrung mit der Programmiersprache C

Tabelle 2 Risiken



1.6.2 Massnahmen

Nr.	Risiko	Definition
1	Erfahrung	Vor dem Start der Diplomarbeit Grundkenntnisse erarbeiten. Bei Schwierigkeiten sich mit dem Diplomlehrer austauschen
2	Zeitaufwand	3 Wochen Ferien beim Arbeitgeber beantragen
3	Kosten	Genaue Evaluation des benötigten Materials
4	Programmierung	Vor dem Start der Diplomarbeit Grundkenntnisse erarbeiten. Bei Schwierigkeiten sich mit dem Diplomlehrer austauschen

Tabelle 3 Massnahmen

1.7. Terminplan SOLL Meilensteine

Für die Diplomarbeit werden wichtige Projekttermine und Meilensteine definiert. Für das Projektmanagement sind die Meilensteine zentral, sie definieren Etappen die anhand von Zwischenzielen überprüft werden können.

18.09.2020	Start Diplomarbeit
18.09.2020	Vorstudie Hardware Software
19.09.2020	Material bestellen
06.10.2020	Aufbau des Prototyps
07.10.2020	Inbetriebnahme Prototyp
07.10.2020	Prototyp testen
08.10.2020	Dossier erstellen
12.10.2020	Aufbau des Systems
13.10.2020	Platzierung und Inbetriebnahme Garden Monitoring
16.10.2020	Garden Monitoring testen
23.10.2020	Dossier beenden
27.10.2020	Onlinepublikation erstellen
02.11.2020	Abgabe Diplomarbeit
05.11.2020	Präsentation erstellen
14.11.2020	Präsentation durchführen

Tabelle 4 Meilensteine

1.8. Terminplan IST

Datum	Thema	Aufwand in Stunden
Initialisierung		
18.09.2020	Start Diplomarbeit	4
18.09.2020	1. Sitzung mit Diplomlehrer	2
20.09.2020	Projektstruktur definieren	5
22.09.2020	Ziele/Risiken/Aufgabenabgrenzung	6
24.09.2020	Projekttermine definieren	4
Vorstudie		
26.09.2020	Projektfindung	2
26.09.2020	Entscheid	1
27.09.2020	LoRa und µC	8
Grobkonzept		
28.09.2020	Auswahl µC LoRa	5
29.09.2020	Auswahl Hardware HMI	4
30.09.2020	Material bestellen	2
03.10.2020	Verbindungstest	6
Detailkonzept		
04.10.2020	Peripherie	10
05.10.2020	Umsetzung und Funktionen	8
Realisierung		
06.10.2020	PIN Belegung Dragino/Sender	2
06.10.2020	Aufbau Sender Holzbrett	25
08.10.2020	Aufbau Empfänger	20
10.10.2020	HMI Nextion Display	10
12.10.2020	Vorbereitung 2.Sitzung	1
12.10.2020	2. Sitzung mit Diplomlehrer	1
16.10.2020	Funktionen mit HMI	5
18.10.2020	Optischer Alarm	15
21.10.2020	Statusüberwachung LED	10
22.10.2020	Energiequelle Sender	5
24.10.2020	Endprodukt Sender	15

25.10.2020	Endprodukt Empfänger	5
26.10.2020	Test mit Endprodukt	10
27.10.2020	Management Summary erstellen	3
28.10.2020	Abschluss Dossier erstellen	15
30.10.2020	Online-Publikation erstellen	5
01.11.2020	Abgabe Diplomarbeit	2
Präsentation		
07.11.2020	Gliederung Präsentation	3
08.11.2020	Präsentation erstellen	6
14.11.2020	Durchführung Präsentation	2
	Total Aufwand in Stunden:	**227**

Tabelle 5 Terminplan IST

2. Vorstudie

2.1 Projektfindung

Da der Arbeitgeber des Diplomanden in der Sicherheitsbranche tätig ist, war es von Anfang an schwierig ein passendes Thema im Bereich der Firma Securiton zu finden. Die Produkte der Firma müssen viele Tests und Sicherheitszertifikate durchlaufen, damit sie in den Vertrieb gelangen können. Die Zeit von sechs Wochen wäre dafür zu kurz gewesen, ausserdem hätte es Schwierigkeiten damit gegeben Alles zu dokumentieren, da z.B. Anschlussschemas nicht an die Öffentlichkeit gelangen dürfen.

Somit hat sich der Diplomand dafür entschieden ein Thema zu wählen, welches nichts mit der Firma Securiton zu tun hat. Zuerst hat der Diplomand an eine kleine Sicherheitsüberwachung im Ferienhaus seiner Grossmutter gedacht. Die Distanz vom Wohnort des Diplomanden zum Ferienhaus ist jedoch zu gross, aus diesem Grund entstand die Idee für das Garden Monitoring, welches schlussendlich auch umgesetzt wurde.

2.2 Entscheid

Als auch die Familie des Diplomanden mit dem Themenvorschlag einverstanden war, schrieb er die Themeneingabe und reichte diese bei der TEKO ein. Im Kurs Embedded System wurden erste Grundkenntnisse mit dem Arduino erarbeitet und kleine Systeme gebaut. Ausserdem wurde den Diplomanden im Kurs Netzwerktechnik 2 einiges über das Thema LoRa beigebracht, somit wusste der Diplomand wie er seine Diplomarbeit angehen kann.

2.3 LoRa

Quelle:

https://www.elektronik-kompendium.de/sites/kom/2203171.htm

2.3.1 Allgemein

LoRa ist ein offener Funkstandard für ein Low Power Wide Area Network (LPWAN), welcher nur kleine Datenmengen übermitteln, jedoch besitzen sie eine hohe Reichweite. In allen LoRa-Sendern und -Empfängern kommen ausschließlich die Chips von Semtech zum Einsatz, weshalb der Betrieb eines solchen Netzwerks von einem einzigen Hersteller abhängig ist.
LoRaWAN ist die Bezeichnung für ein Funknetzwerk auf Basis von LoRa. LoRa und nutzt Frequenzbänder aus den lizenzfreien ISM-Bändern. Damit kann ein LoRaWAN eine Alternative oder Ergänzung zum klassischen Mobilfunknetz mit zentralem Netzbetreiber sein. Als Abgrenzung zum klassischen Mobilfunk wird ein LoRaWAN auch als 0G-Netz bezeichnet.
Da LoRa ein offener Funkstandard ist, kann jeder ein LoRaWAN als IoT- oder M2M-Netzwerk mit bidirektionaler Kommunikation aufbauen oder eine Community-basierte Lösung nutzen.
Hinweis: Zu beachten ist, dass LoRaWAN in den USA ungleich zu LoRaWAN EU ist. Das hat Einfluss auf die Übertragungsrate und damit auch auf den Energieverbrauch.

Eigenschaften	
Verbindung	Uplink-orientiert, bidirektional, Quittungsbetrieb möglich
Modulation	Chirp-Spread-Spectrum und FSK
Netzarchitektur	Die Endgeräte kommunizieren mit Gateways, welche die Datenpakete an einen Server übermitteln. Der Server verfügt über Schnittstellen für die Anbindung an IoT-Plattformen und -Applikationen.
Frequenzbereiche	In Europa 868 MHz (863–870 MHz, unterteilt in mehrere Subbänder) und in den USA 915 MHz. Die Kanalnutzungsdauer ist in vielen Ländern regulatorisch begrenzt (Arbeitszyklus).

Reichweite	Je nach Topografie bis 2 km in Stadtgebieten und bis 15 km in ländlichen Gebieten. Dabei wird eine gute Durchdringung von Gebäuden erreicht.
Energieverbrauch	Zwischen 10 mA und 100 nA im Ruhemodus. Je nach Anwendungsfall beträgt die Batte-rielebensdauer 2 bis 15 Jahre.
Funkkanalband-breite	125 kHz
Empfindlichkeit	-137 dBm
Sendeleistung	+20 dBm bzw. maximal 25 mW
Datenpakete	EU: max. 51 Byte / USA: max. 11 Byte Nutzdaten pro Paket
Übertragungsrate	zwischen 250 Bit/s und 50 kBit/s

Tabelle 6 LoRa Eigenschaften

2.3.2 Übertragungstechnik

Um eine hohe Effizienz bei Datentransfer und Energieverbrauch zu erreichen, nutzt LoRaWAN eine Frequenzspreizung. Dadurch können Interferenzen weitestgehend vermieden und schmalbandige Störungen umgangen werden.
Das Übertragungsverfahren heißt „Chirp Spread Spectrum“. Hierbei erfolgt die Signalübertragung als eine Art Zirpen. Dabei wird der Zirp-Impuls über einen großen Frequenzbereich gespreizt. Wahlweise kann die Bandbreite für eine hohe Datenrate oder eine robuste Übertragung genutzt werden. Der Spreizfaktor (Spread Factor) und die Bandbreite bestimmen, wie hoch die Datenrate sein kann und wie hoch die Empfangswahrscheinlichkeit ist.
Signale, die mit verschiedenen Spreizungsfaktoren moduliert und über denselben Frequenzkanal übertragen werden, stören sich nicht gegenseitig. Die Orthogonalität der Spreizfaktoren ermöglicht gleichzeitiges Senden mehrerer Endgeräte im selben Kanal.
Die LoRa-Signale sind sehr robust gegen In-Band- und Out-of-Band-Interferenzen. Ihre Unempfindlichkeit gegen Mehrwegeempfang oder Fading sorgt in städtischen Gebieten für eine hohe Reichweite.

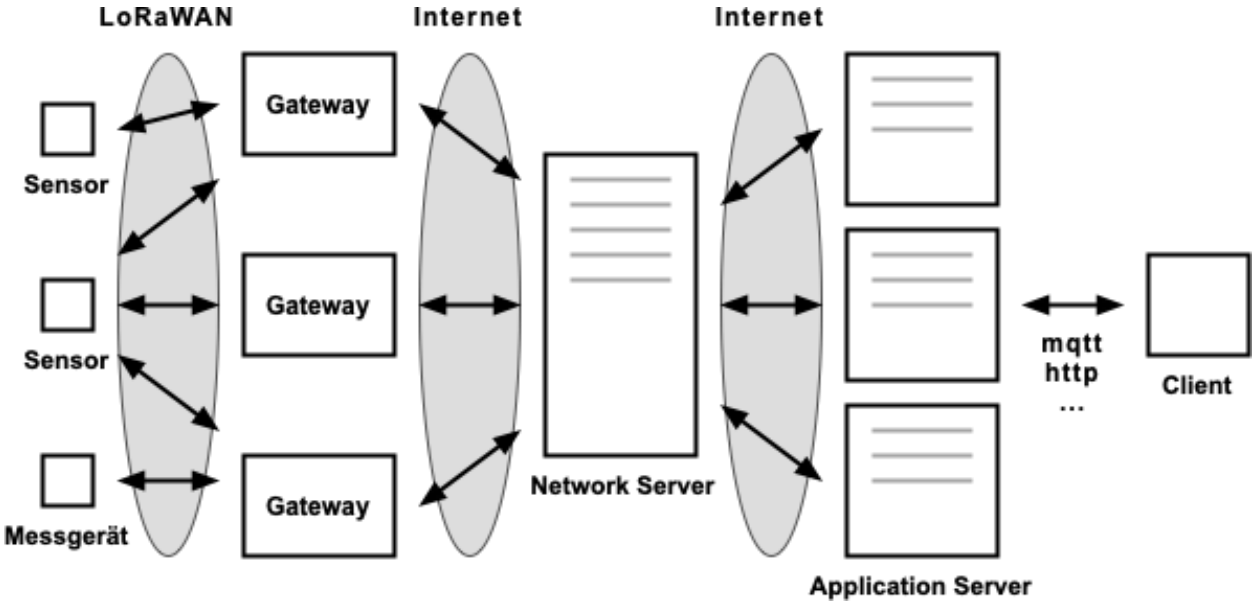


Abbildung 2-1 Netzstruktur LoRa

Die LoRaWAN-Netzarchitektur besteht aus vielen Endgeräten in Form von Sensoren und Aktoren, mehreren Gateways und einem zentralen Network Server. Die Endgeräte kommunizieren mit den Gateways. Und die Gateways sind mit dem Network Server verbunden. Der Network Server kommuniziert über verschiedene Protokolle (z.B. REST, MQTT, usw.) mit einer Anwendung, die zum Beispiel als Applikation in der Cloud betrieben wird.
In einem LoRaWAN sind Gateways die Empfänger für die Funksignale bei 868 MHz. Hier empfangen die LoRa-Chips die Zirp-Signale. Auf der anderen Seite sind die Gateways mit dem Internet verbunden.
Die Gateways eines LoRaWAN bilden im Optimalfall ein engmaschiges Netz und können über die ganze Welt verteilt sein.
Eine Nachricht kann von einem oder mehreren Gateways empfangen werden. Die Gateways leiten diese ohne weitere Eingriffe an die Network Server weiter.
In einem LoRaWAN sind die Network Server dafür zuständig, den Absender zu identifizieren und das Paket an einen Application Server weiterzuleiten.
Der Network Server kümmert sich unter anderem darum, dass eine Nachricht nur einmal beim Application Server ankommt, egal wie viele Gateways diese empfangen haben.

2.3.3 Private oder Community Network

Grundsätzlich kann jeder ein eigenes LoRaWAN betreiben. Da LoRa im zuteilungsfreien Frequenzbereich arbeitet sind keine Lizenzkosten für Frequenzen notwendig.
Wer nur in einem beschränkten Bereich ein LoRaWAN aufbauen muss, für den kann der Betrieb eigener Gateways und Server sinnvoll sein.
Ist man jedoch auf ein weitflächiges Funknetz angewiesen, kann man sich auch am Community-basierten Netzwerk „The Things Network (TTN)“ beteiligen. Bei diesem Netz zum Mitmachen betreibt man nur ein eigenes Gateway, welches über das Internet mit den TTN-Servern spricht. Bezüglich der Sicherheit muss man nur dem Network Server vertrauen, dass er die empfangenen Datenpakete zustellt und dem Application Server, der die Inhalte entschlüsseln kann.

2.3.4 Reichweite

LoRa hat eine hohe Sensibilität von -137 dBm, was die Verfügbarkeit des Netzes erhöht. Die Signale durchdringen Gebäudemauern problemlos und erreichen so zum Beispiel auch Kellerräume oder andere, sogenannte Deep-Indoor-Standorte.
Die Reichweite zwischen Sender und Empfänger beträgt je nach Umgebung und Bebauung ca. 3 km (Stadt), ca. 6 km (Vororte) und bis zu 13 km (ländliche Gebiete).
Wie groß der Abstand zwischen LoRa-Sender und Empfänger sein darf, ist vom Spreading Factor, der Bandbreite, der gewählten Sendeleistung des LoRaChips und der verwendeten Antenne abhängig.

2.3.5 Übertragungsrate

Zur Maximierung der Batterielebensdauer und der Steuerung der Gesamtnetzkapazität (begrenzt durch regulatorische Vorgaben), steuert LoRa die Datenrate und den HF-Ausgang mittels adaptiver Datenrate (ADR) einzeln für jedes Endgerät.
Die Kommunikation zwischen Endgerät und Gateway erfolgt auf verschiedenen Frequenzkanälen mit unterschiedlichen Datenraten. Die Datenraten reichen von 0,3 bis 50 kBit/s. Die physikalische Paketgröße beträgt 64 Byte brutto. Für den Header werden 13 Byte benötigt. Es bleiben also 51 Byte für die Nutzdaten übrig.

Hinweis: In den USA ist die maximale Kanalbelegungszeit auf 400 ms begrenzt. Das führt dazu, dass nur maximal 11 Byte Nutzdaten pro Paket übertragen werden können.
Der SF12 (Spreizfaktor) bei 125 kHz (Bandbreite) kommt nur auf 250 Bit/s (Datenrate). Dafür nimmt der Empfänger den Zirp-Impulse mit sehr hoher Wahrscheinlichkeit wahr, weil es ihm vergleichsweise leicht fällt, die Signale vom Rauschen zu unterscheiden.
Die schnellste spezifizierte Kombination ist SF7 bei 250 kHz Bandbreite. Damit kommt man auf 11.000 Bit/s.

2.3.6 Energieverbrauch

Das LoRa-Modulationsverfahren ermöglicht eine optimale Sendeleistung bei möglichst niedriger Leistungsaufnahme durch den Transmitter. Der geringe Energieverbrauch ermöglicht eine Lebensdauer der Batterie von bis zu 15 Jahre.
Das vereinfacht die Handhabung und ist kostengünstig, weil keine separate Stromversorgung gebraucht wird.

2.3.7 Endgeräteklassen

LoRa unterscheidet verschiedene Endgeräteklassen, wobei nur die Klasse A für Anwendungen im Internet der Dinge interessant sind. Das Endgerät befindet sich in einem batterieschonenden Zustand und sendet nur kurz bei einer Zustandsänderung. Zum Endgerät kann dann nur in dieser Zeit etwas gesendet werden.
Zusammen mit dem Funkmodul sind diese Endgeräte wegen ihrer Einfachheit sehr günstig und eignen sich auch zur Deckung eines hohen Bedarfs.
Sollen Endgeräte auch außerhalb dieser Frist adressiert werden können, muss faktisch die Geräteklasse B oder C gewählt werden, was den Stromverbrauch stark erhöht und je nach Netz gar nicht unterstützt wird.

2.3.8 Anwendungen

LoRa ist primär für statische Sensor-Anwendungen gemacht. Typische Anwendung sind Statusinformationen erfassen, abfragen und austauschen. Mit Sensoren, die sich an beliebigen Orten befinden, können Informationen ermittelt oder gewonnen werden, die auf einfache Weise in eine Applikation integriert werden können.

- Smart Home
- Smart City
- Smart Factory
- Smart Farming
- Smart Transport
- The Things Network (TTN)

„The Things Network“ (TTN) ist eine Community-basierte Initiative zur Errichtung eines globalen IoT-Funknetzwerks auf Basis von LoRa. Die Idee ist, dass man sich auf der ganzen Welt in Reichweite zu einem TTN-Gateway befindet.
Die Bereitstellung, Errichtung und Betreuung der Gateways liegt dabei in den Händen Freiwilliger bzw. Teilnehmer des Netzwerks. Wer ein eigenes LoRaWAN-Projekt plant und ein TTN-Gateway betreibt, versorgt immer auch seine Umgebung. Das macht LoRaWAN interessant für Bastler und kommerzielle Nutzer gleichermaßen. Die Organisation betreibt dafür Network Server und Application Server.

2.4 µC

Quelle:

https://www.gruenderszene.de/lexikon/begriffe/microcontroller#:~:text=Begriff.&text=Bei%20einem%20Mikrocontroller%20handelt%20es,Ein%2DChip%2DSystem%20gibt.

2.4.1 Allgemein

Bei dem englischen Wort Microcontroller handelt es sich um einen Ein-Chip-Computersystem. Generell werden Microcontroller auch als Halbleiterchips bezeichnet, die auch einen Prozessor sowie eine Peripheriefunktion enthalten.
Bei einem Mikrocontroller handelt es sich um einen bestimmten Mikrorechner, welcher neben dem Prozessor noch weitere wichtige Komponenten und Elemente von einem Kreisumfang (Peripherie) auf einem Chip zusammengefügt, sodass es ein Ein-Chip-System gibt.
Die Anwendungsfälle für Microcontroller werden meistens integriert und ausgesucht und sollen im besten Fall die Aufgaben mit wenigen erforderlichen externen Bausteinen ausfüllen und können so auch in größeren Stückzahlen hergestellt werden. Microcontroller sind so in der Lage die verschiedensten Steuerungs- und Kommunikationsaufgaben zu lösen.

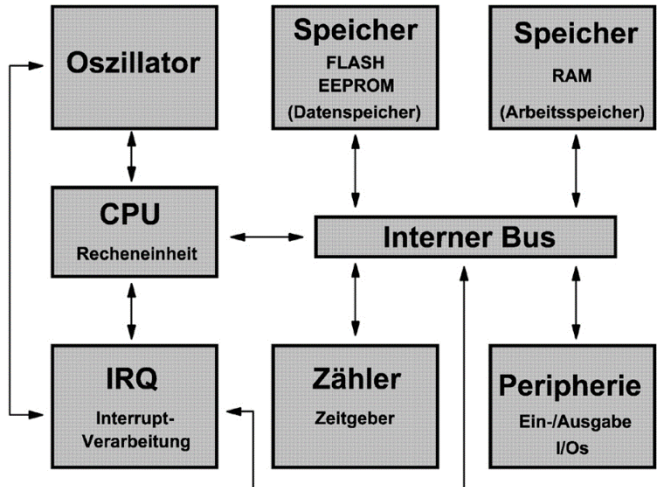


Abbildung 2-2 Aufbau µC

2.4.2 Einsatzbereiche von Microcontroller

Der Microcontroller ist in den meisten Fällen in Systemen eingebettet und befindet sich in alltäglichen technischen Gebrauchsprodukten, wie beispielsweise in Waschmaschinen, Geld - oder Telefonchipkarten, in der Unterhaltungselektronik, wie Radios, Fernsehgeräten oder Fernbedienungen.
Mikrocontroller werden von der Leistung und Ausstattung auf die jeweilige Anwendung passend eingegliedert. Daher haben sie gegenüber „normalen“ Computern deutliche Vorteile bei den Kosten und der Leistungsaufnahme. Kleine Mikrocontroller sind in höheren Stückzahlen für deutlich unter 1 € verfügbar.
Auch wird den Microcontrollern im Internet der Dinge eine größere Rolle zuteil gemacht.

2.4.3 Die Programmierung von Mikrocontrollern

Die Programmierung von Mikrocontrollern findet häufig in den Programmiersprachen Assembler oder C statt. Außerdem werden auch die Programmiersprachen BASIC, Pascal, Forth, Ada oder C++ genutzt.

3. Grobkonzept

3.1 Auswahl Hardware µC / LoRa

Im Embedded System Unterricht der Schule TEKO wurde bereits mit einem Arduino Uno. Deshalb steht dem Diplomanden dieser Bauteil bereits zur Verfügung. Die LoRa Technologie wurde nur in der Theorie unterrichtet. Es gibt viele verschiedene µC mit LoRa Funktionen, zwei Varianten davon habe ich genauer angeschaut.

3.1.1 LSN50 V2 wasserdichter LoRa Sensor Node 868MHz Dragino

- STM32L072CZT6 MCU
- SX1276/78 LoRa Wireless Modem
- Frequenz: 868MHz
- ISP Bootloader vorinstalliert
- I2C, LPUSART1, USB
- 18x Digitale I/Os
- 2x 12bit ADC, 1x 12bit DAC
- MCU wake up über UART oder Interrupt
- LoRa™ Modem
- Preamble Detektion
- Baudrate konfigurierbar
- LoRaWAN 1.0.2 Spezifikation
- Software basiert auf den STM32Cube HAL Treibern
- AT Kommandos zum Setzen von Parametern
- 4000mAh Batterie mit langer Lebensdauer

Abbildung 3-1 LSN50 V2 LoRa Sensor Node

Der LSN50 V2 von Dragino ist ein geeigneter LoRa-Node. Er ist in einem Wasserdichten Gehäuse verbaut und besitzt einen 4000mAh Akku.

3.1.2 Dragino LoRa Shield - 868MHz v1.4

- Kompatibel mit 3.3V oder 5V Arduino Board
- Kompatibel mit Arduino Leonardo, Uno, Mega, DUE
- Frequenzband Band: 868MHz
- Sehr geringer Stromverbrauch
- Externer Antennen Anschluss
- Technische Details:
- 168dBmaximum Link Budget
- +20dBm - 100mW konstanter RF Ausgang
- +14dBm Hochleistungsverstärker
- Programmierbare Datenrate von bis zu 300kbps
- Hohe Empfindlichkeit: bis zu -148dBm
- Frontend: IIP3= -12.5dBm
- Immunität gegen Blocking
- Niedriger RX Strom von 10.3mA, 200nA zum Halten der Register
- Integrierter Synthesizer mit einer Auflösung von 61Hz
- FSK, GFSK, MSK, GMSK, LoRa TM und OOK Modulation
- Eingebauter Bit Synchronisierer zur Taktwiederherstellung
- Preamble Detection
- 127dB Dynamischer RSSI Bereich

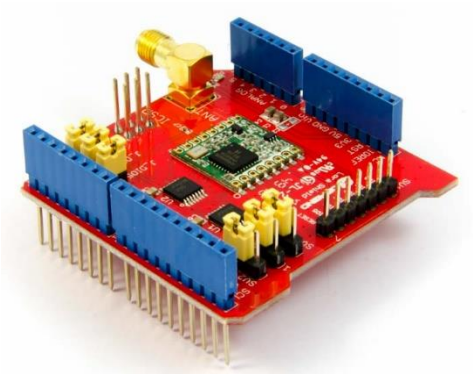


Abbildung 3-2 Dragino LoRa Shield

Das LoRa Shield basiert auf einem HopeRF RFM95W/RFM98W und eignet sich perfekt für professionelle drahtlose Sensornetzwerkanwendungen wie Bewässerungssysteme, Messdatenerfassung, Smartphone Erkennung, Gebäudeautomation usw.

3.1.3 Arduino MEGA

Mikrocontroller:	ATmega2560
Betriebsspannung:	5V
Eingangsspannung:	7-12V
Eingangsspannung:	6-20V
Digitale E / A-Pins:	54
Analoge Eingangsstifte:	16
Gleichstrom pro E / A-Pin:	20 mA
Gleichstrom für 3,3 V Pin:	50 mA
Flash-Speicher:	256 KB
SRAM:	8 KB
EEPROM:	4 KB
Taktfrequenz:	16 MHz
LED_BUILTIN:	13

Abbildung 3-3 Arduino MEGA

3.1.4 Arduino UNO

Mikrocontroller:	ATmega328P
Betriebsspannung:	5V
Eingangsspannung:	7-12V
Eingangsspannung:	6-20V
Digitale E / A-Pins:	14
PWM Digital I / O-Pins:	6
Analoge Eingangsstifte:	6
Gleichstrom pro E / A-Pin:	20 mA
Gleichstrom für 3,3 V Pin:	50 mA
Flash-Speicher:	32 KB
SRAM:	2 KB (ATmega328P)
EEPROM:	1 KB (ATmega328P)
Taktfrequenz:	16 MHz
LED_BUILTIN:	13

Abbildung 3-4 Arduino UNO

3.2 Entscheid

Durch die Erfahrung mit dem Arduino entscheidet sich der Diplomand für die Variante Arduino mit dem LoRa-Shield von Dragino. Die Selbstversorgung des LSN50 V2 ist ein grosser Vorteil, jedoch ist die Einarbeitung in die Software sehr zeitaufwändig. Ausserdem ist die serielle Anbindung des HMI Displays ein weiterer Vorteil des Arduino Mikrocontroller.

Für den Aussensender im Garten entscheidet sich der Diplomand für ein Arduino Mega mit einem LoRa Schield. Der Grund dafür ist die grössere Speicherkapazität und die zusätzlichen vorhandenen I/O. Falls die Zusatzoption mit weiteren Bodenfeuchtigkeitssensoren realisiert werden soll, bringt das Arduino Mega die richtigen Voraussetzungen mit. Für den Empfänger in der Wohnung wird das bereits vorhandene Arduino Uno eingesetzt und ebenfalls mit den LoRa Shield von Dragino bestückt.

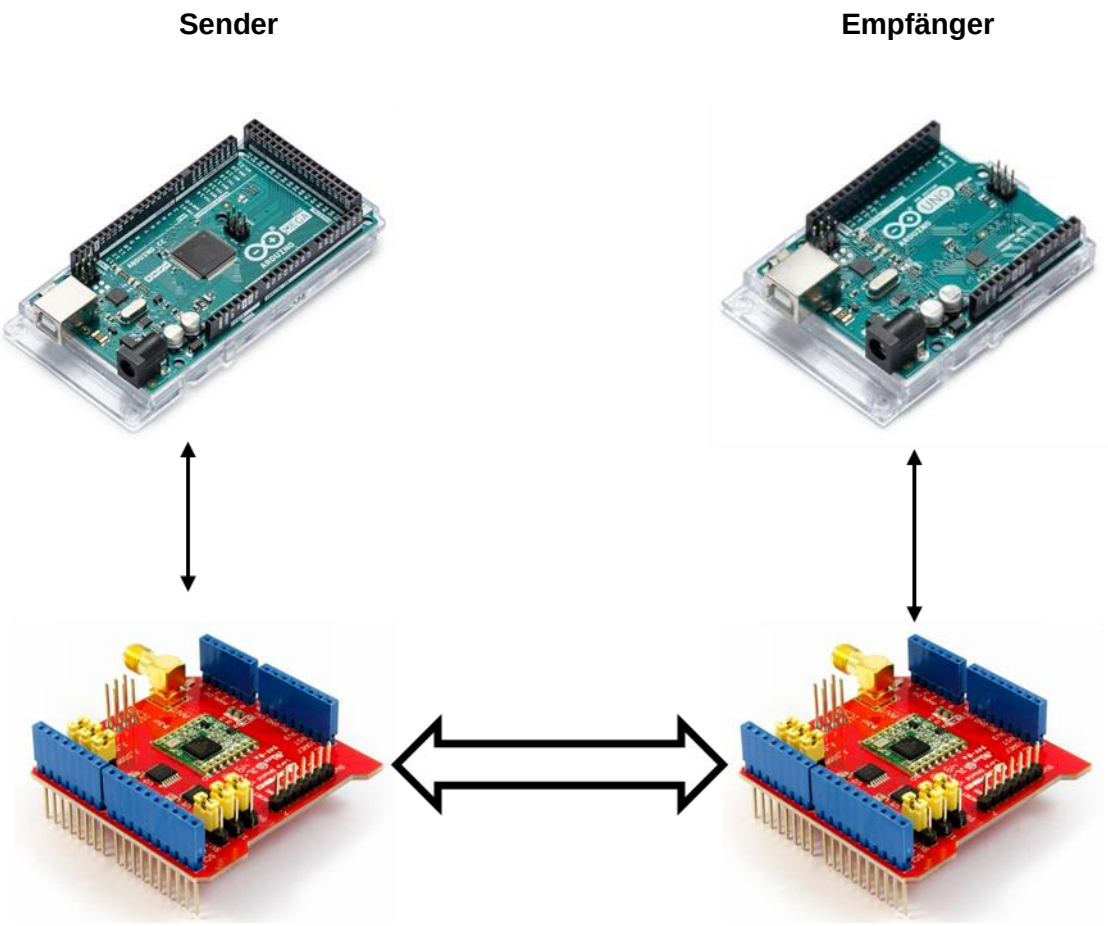


Abbildung 3-5 Übersicht Entscheid

3.3 Auswahl Hardware HMI

Um die Sensorwerte anzuzeigen und optional die Bewässerung zu starten verlangt der Auftraggeber ein Display mit Touch-Funktion. Die Auswahl fiel auf das ITead Studio 3.5 Zoll HMI LCD Nextion-Display.

Technische Daten

- Eingebaute RTC (Realtime Clock)
- GPIO-Support
- Bilddiagonale 3.5"
- Auflösung 480 x 320 Bildpunkte
- 65k Farben
- TFT-Anzeige mit integriertem Touch (Resistive)
- TTL 4-Pin Interface
- 32M Flash Speicher
- EEPROM: 1024 Bytes
- RAM: 8192 Bytes
- microSD-Kartenleser für Firmware-Upgrades
- Sichtbarer Bereich: 73.44mm x 48.96mm
- Grösse: 100.5 x 54.94mm
- Helligkeit Einstellbar von 0 - 180 nit, in 1%-Schritten
- Stromverbrauch 5V 145mA (Speisung mit 5V 500mA empfohlen)
- Kompatibel mit Arduino, Raspberry Pi und vielen mehr

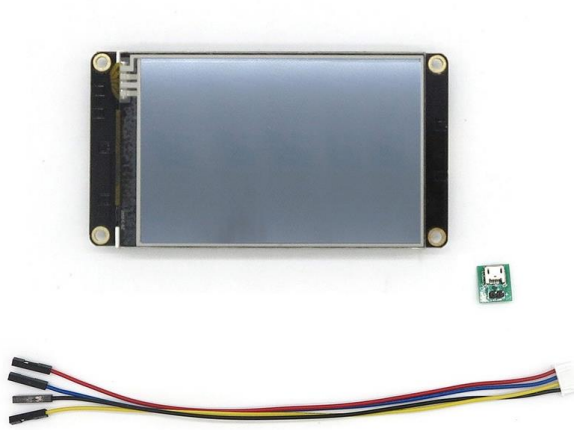


Abbildung 3-6 ITead Studio 3.5 Zoll HMI Nextion Display

Mit dem Nextion Editor kann schnell und einfach eine Bedieneroberfläche programmiert werden. Anschliessend werden mit einer Micro SD Karte die Werte auf das Display gespeichert. Nach dem Neustart und dem entfernen der SD Karte auf dem Display sind die benutzerdefinierten Einstellungen auf dem HMI Gerät abgespeichert. Die Kommunikation mit der Arduino erfolgt über die serielle UART Schnittstelle. Gemäss Hersteller sollte die Stromversorgung nicht über Arduino Stromversorgung angeschlossen werden, sondern über ein externes 5V Netzgerät.

3.4 Verbindungstest

Zuerst muss eruiert werden, wie die Punkt zu Punkt Verbindung zwischen den beiden LoRa Knoten funktioniert. Beim ersten Verbindungstest wird mit einer sehr kurzen Distanz getestet, beim zweiten Test wird mit der originalen Distanz gemessen.

3.4.1 Beispielsketch

Auf folgender Website hat der Diplomand den Beispielsketch für den Verbindungstest gefunden.
https://www.aeq-web.com/arduino-lora-wireless-shield-for-iot-direct-mode/

Mit diesem Beispielsketch wird im seriellen Monitor die Empfangsfeldstärke RSSI angezeigt.

Abbildung 3-7 Beispielsketch Verbindungstest Slave

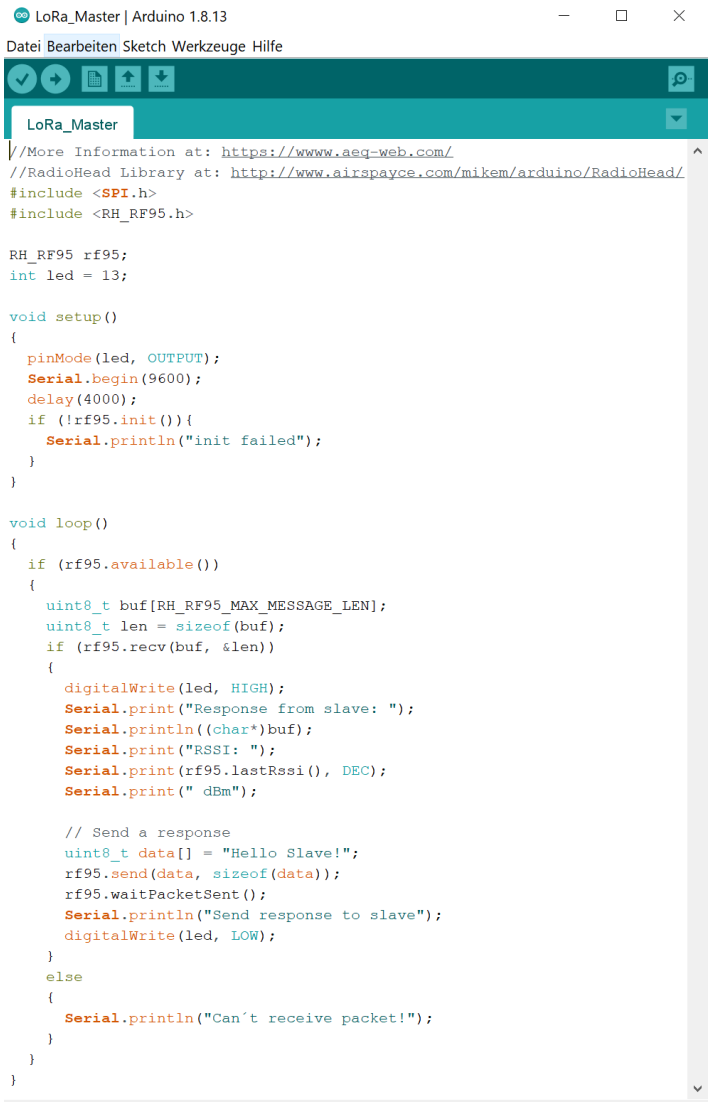


Abbildung 3-8 Beispielsketch Verbindungstest Master

Wichtig: Damit der Sketch richtig funktioniert muss die Datei RH-RF95.cpp im Verzeichnis der RadioHead-Bibliothek modifiziert werden. In der Datei unter «setFrequency» muss die Frequenz auf 868,0 MHz umgeschrieben werden.

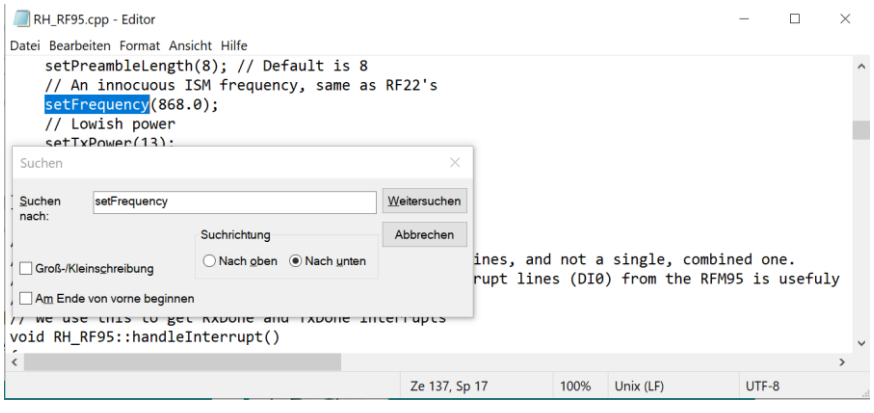


Abbildung 3-9 Frequenz anpassen

3.4.2 Erster Verbindungstest

Der erste Verbindungstest mit ca. 0.3m Abstand zeigt eine Verbindung von -8 dBm.

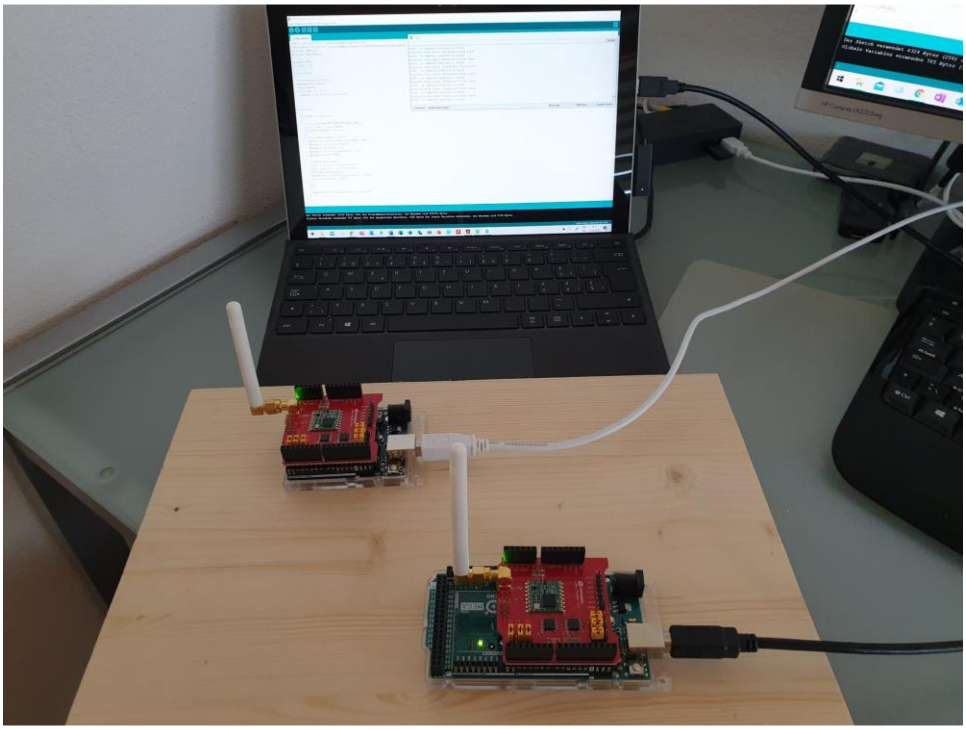


Abbildung 3-10 Erster Verbindungstest

Abbildung 3-11 Serieller Monitor erster Verbindungstest

3.4.3 Zweiter Verbindungstest

Beim zweiten Verbindungstest mit ca. 25m Distanz wird eine stabile Verbindung von 93-95 dBm erreicht. Da der LoRa-Shield ein RSSI von -137dBm toleriert, sollte dies für das Garden Monitoring kein Problem darstellen.

Abbildung 3-13 Zweiter Verbindungstest Sender

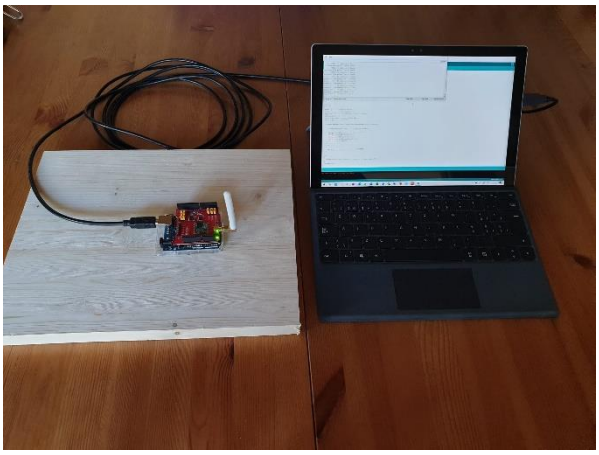


Abbildung 3-12 Zweiter Verbindungstest Empfänger

Abbildung 3-14 Distanz zweiter Verbindungstest

Abbildung 3-15 Serieller Monitor zweiter Verbindungstest

4. Detailkonzept

4.1. Peripherie

Folgende Peripheriegeräte wurden für die Diplomarbeit Garden Monitoring verwendet:

4.1.1 Adafruit Temperatur/Feuchtigkeitssensor AM2315

Der AM 2315 ist ein I2C-Schnittsellen Temperatur- und Feuchtigkeitssensor der in einem geschlossen Gehäuse ausgeliefert wird. Dadurch ist er ideal für den Einsatz bei Wind und Wetter.

Abbildung 4-1 Adafruit Temperatur/Feuchtigkeitssensor AM2315

4.1.2 Velleman Bodenfeuchtigkeitssensor

Der analoge Bodenfeuchtigkeitssensor wird mit 5V betrieben. Sobald die Sensorteile mit Wasser bedeckt sind, ändert sich der Analogwert am SIG-Anschluss. Je kleiner dieser Wert ist desto trockener ist der Sensor.

Abbildung 4-2 Velleman Bodenfeuchtigkeitssensor

4.1.3 Regensensor Modul

Der analoge Bodenfeuchtigkeitssensor wird mit 5V betrieben. Sobald sich Regen oder Schnee auf dem Sensor befinden, ändert sich am analogen Ausgang der Widerstand. Je höher der gemessene Wiederstand ausfällt desto trockener ist der Sensor. Ausserdem kann über den digitalen Ausgang anhand eines Potentiometers der Schwellwert festgelegt werden wann der Ausgang schaltet.

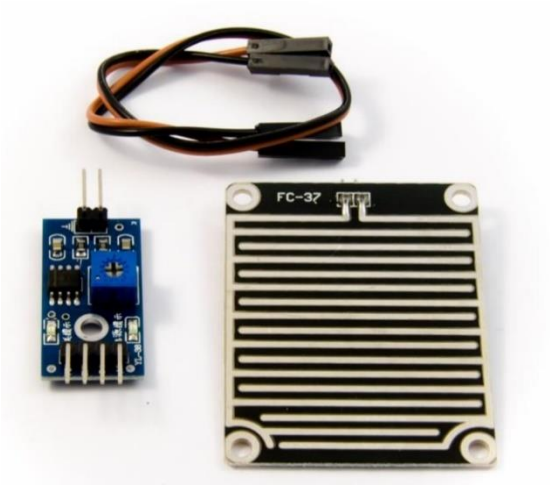


Abbildung 4-3 Regensensor Modul

4.1.4 Optionales MPPT Solar Power Management Modul

Das Solarstrommanagementmodul ist für 6V – 24V Solarmodule ausgelegt. Es kann einen 3,7V Li-Akku aufladen und hat einen geregelten 5V/1A USB Ausgang.

Abbildung 4-4 MPPT Solar Power Management Modul

4.1.5 Optionaler 14500 3.7V Li-Ion Akku 750mAh

Dieser Akku eignet sich für MPPT Solar Power Management Modul und ist mit seinen 750mAh ideal für das Garden Monitoring.

Abbildung 4-5 14500 3.7V Li-Ion Akku 750mAh

4.1.6 Optionale Solarzelle 12V 250mA 3W

Diese Monokristalline Solarzelle eignet sich bestens für Arduino Projekte.

Abbildung 4-6 Solarzelle 12V 250mA 3W

4.1.7 Kosten

Material	Lieferant	Anzahl	Stückpreis (CHF)	Preis (CHF)
Arduino UNO	Bastelgarage.ch	1	27.90	27.90
Arduino MEGA	Bastelgarage.ch	1	36.10	36.10
Dragino LoRa	Bastelgarage.ch	2	22.90	45.80
Nextion Touchdisplay	Digitec.ch	1	72.90	72.90
Regensensor	Bastelgarage.ch	1	3.90	3.90
Bodenfeuchtigkeits-sensor	Digitec.ch	1	10.90	10.90
Adafruit Tempera-tur/Feuchtigkeitssensor	Digitec.ch	1	41.10	41.10
Solarzelle 12V 250mA 3W	Bastelgarage.ch	1	19.90	19.90
MPPT Solar Power Ma-nagement Modul	Bastelgarage.ch	1	14.90	14.90
14500 3.7V Li-Ion Akku 750mAh ICR14500	Bastelgarage.ch	1	7.50	7.50
Breadboard klein	Bastelgarage.ch	4	4.90	19.60
Dupont Kabel	Bastelgarage.ch	1	2.90	2.90
Micro SD	Digitec.ch	1	22	22
Elektrobox	Bauhaus	1	17.90	17.90
Zaunpfahl	Bauhaus	1	9.50	9.50
			Total	**352.80**

Tabelle 7 Kosten

4.2. Umsetzung und Funktionen

Das Projekt Garten Monitoring wird mit einem Sender im Garten und einem Empfänger in der Wohnung umgesetzt. Der Sender wird in der Mitte des Gemüsegartens platziert und wird über eine Entfernung von 25m die Sensorwerte an den Empfänger in der Wohnung übermitteln.

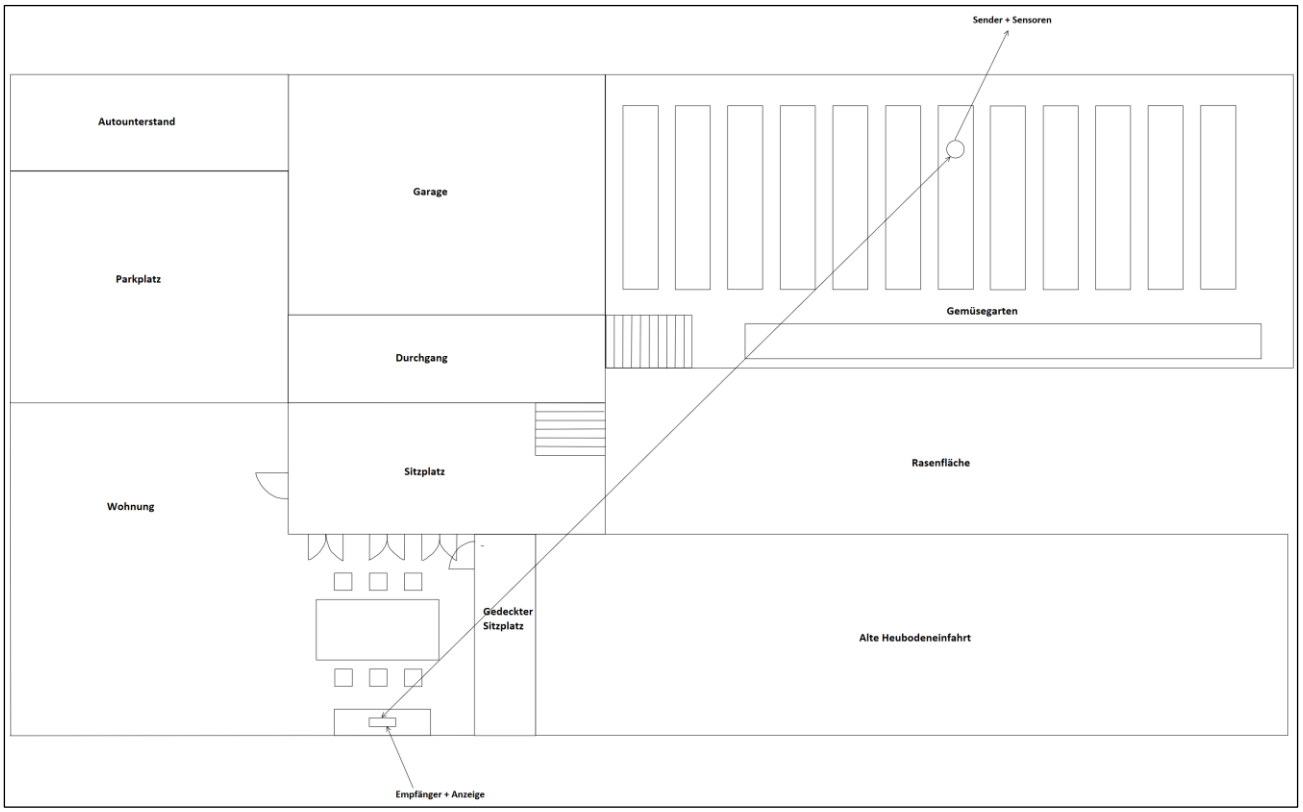


Abbildung 4-7 Situationsplan

4.2.1 Aufbau und Funktionen Sender

Der Sender hat mehrere Sensoren aufgeschaltet, welche ihre Messwerte an das Arduino MEGA übergeben. Der µC übermittelt anschliessend die Sensordaten mit Hilfe des Dragino-LoRa-Shield an den Empfänger.
Der Temperatur- und Feuchtigkeitssensor AM2315 wird über den I2C-Bus betrieben, er verfügt über einen eigenen Controller, welcher beiden Sensoren eine I2C-Schnittstelle bietet. Die Adresse des Geräts kann nicht verändert werden, somit wird nur ein AM2315 pro I2C-Bus unterstützt.
Der analoge Bodenfeuchtigkeitssensor wird über einen analogen Eingang ausgewertet. Analog wie auch digital wird der Regensensor an den µC angeschlossen. Er liefert einen Messwert, welcher am analogen Eingang ausgewertet wird. Der digitale Eingang kann für seine Zustandsänderung verwendet werden.
Die Speisung des Senders wird mit einer 9V Batterie umgesetzt, als Option besteht eine Photovoltaik betriebene Lösung.
Als zweite Option wird ein 2-Kanlal Relais in das System eingebettet, das die Bewässerung ein- und ausschaltet.

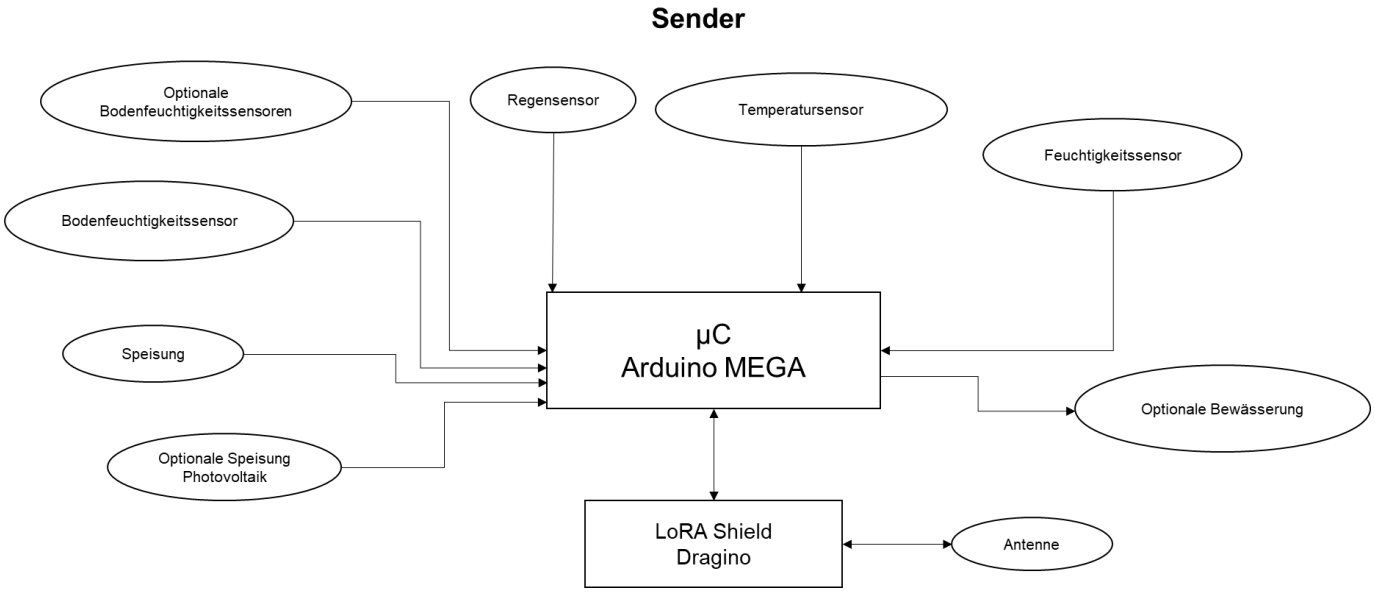


Abbildung 4-8 Blockdiagramm Sender

Der Aufbau des Senders muss wetterbeständig sein. Die Komponenten werden auf einem Zaunpfahl mit einer Nassabzweigdose montiert.

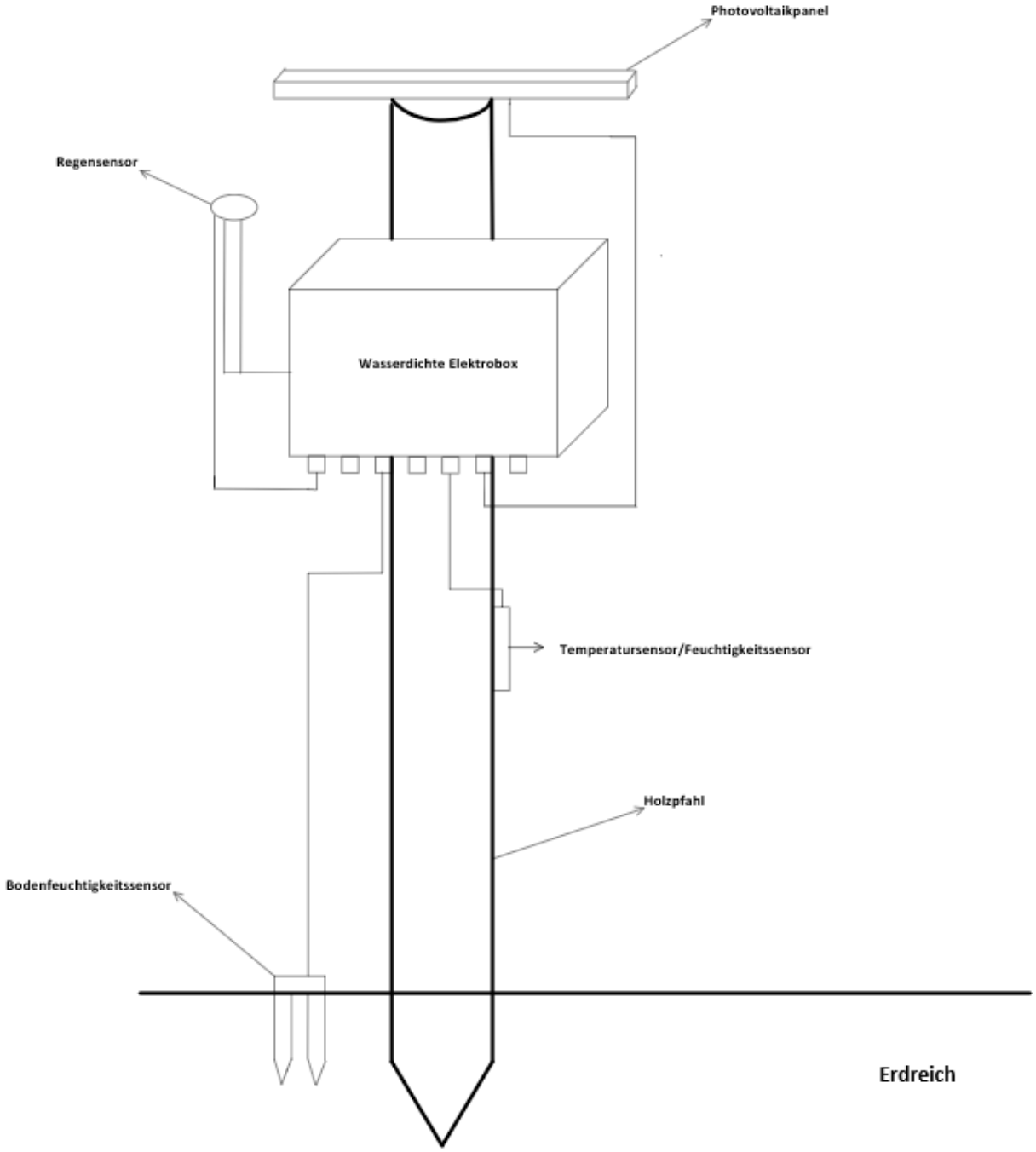


Abbildung 4-9 Planung Aufbau Sender

4.2.2 Aufbau und Funktionen Empfänger

Wie der Sender wird auch der Empfänger mit einem Dragino-LoRa-Schield bestückt. Die empfangenen Sensordaten werden über die erste serielle Schnittstelle des Arduino UNO an das HMI-Display von Nextion übertragen. Alle Messdaten der Sensoren sollen auf dem Display angezeigt werden. Bei einem zu hohen Messwert des Bodenfeuchtigkeitssensors soll es eine optische Alarmierung auf dem Display generieren. Die optionale Funktion auf dem Empfänger ist eine Statusanzeige, die durch eine LED an den µC angeschlossen wird. Die LED soll beim empfangen der Sensordaten kurz aktiviert werden.

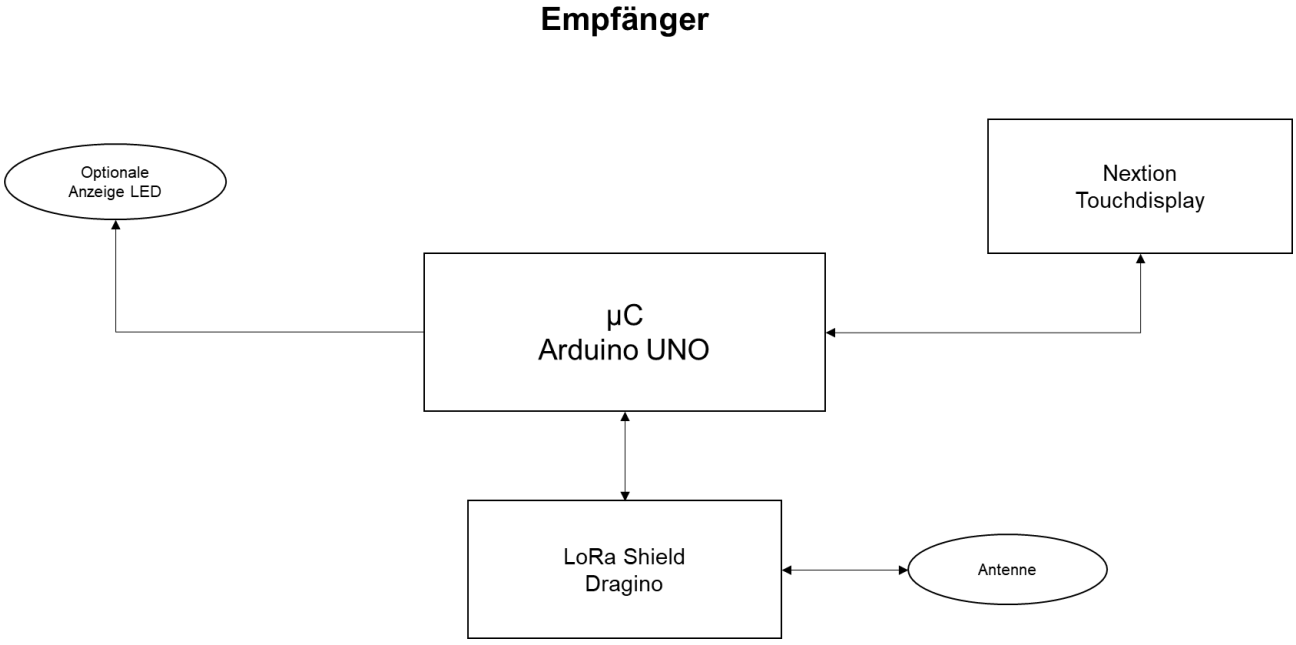


Abbildung 4-10 Blockdiagramm Empfänger

5. Realisierung

Für den Prototyp werden die Hardware Komponenten des Senders auf ein Holzbrett geschraubt. Der Empfänger wurde bereits in die Halterung eingebaut. Auf dem Sender-Holzbrett befindet sich das Arduino MEGA mit dem Dragino-Schield, der Temperatur- und Feuchtigkeitssensor, der Bodenfeuchtigkeitssensor sowie der Regensensor. Die Empfänger-Halterung ist mit dem Arduino UNO, dem Dargino-Schield und dem Nextion Touch-Display ausgestattet. Zuerst werden die einzelnen Funktionen umgesetzt und getestet. Letztens werden alle Funktionen zusammengefügt.

Hardware Sender

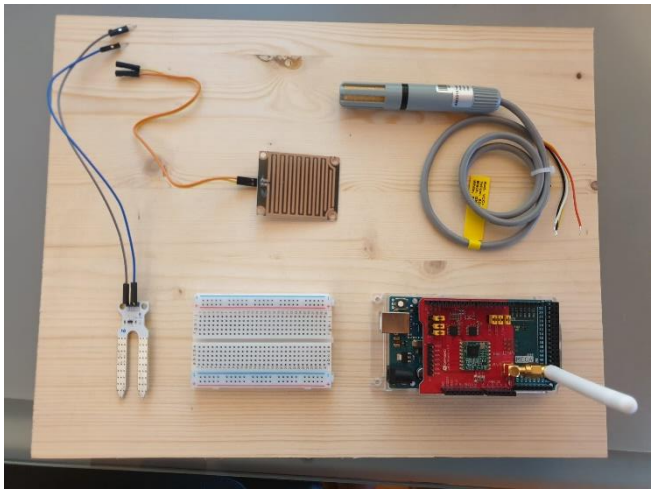


Abbildung 5-1 Hardware Sender

Hardware Empfänger

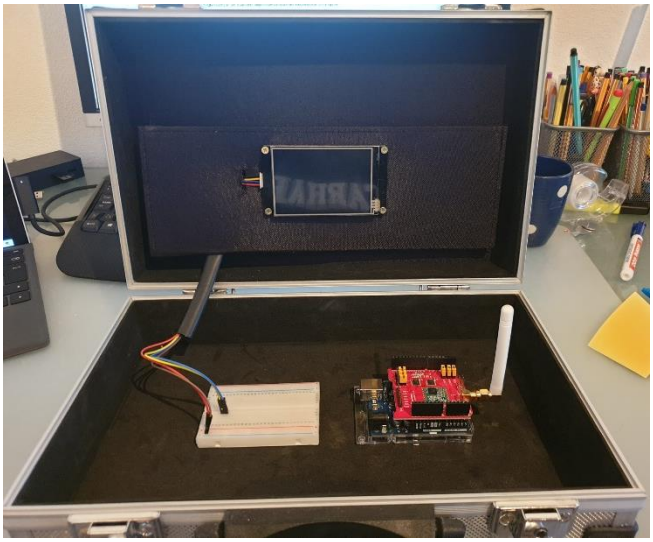


Abbildung 5-2 Hardware Empfänger

5.1. Funktionen

Im ersten Schritt werden alle Sensoren auf den Empfänger implementiert und mit Hilfe des seriellen Monitors getestet.

5.1.1 PIN Belegung Dargino Shield

Folgende PIN's sind bereits durch das Dragino-Shield belegt und können nicht mehr verwendet werden:

- Chip RESET = D9
- Chip DIO0 = D2
- Chip DIO5 = D8
- Chip DIO2 = D7
- Chip DIO1 = D6
- R9 - R11
- SV2 – SV4

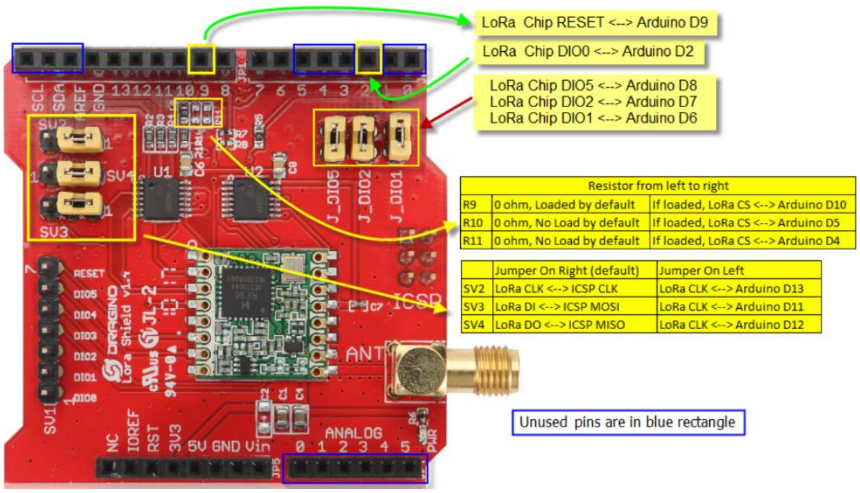


Abbildung 5-3 PIN Belegung Dagino-Shield

5.1.2 PIN Belegung Sender

Die Sensordaten auf dem Sender werden an folgenden Pins ausgewertet:

PIN	Sensor	Eingang
SCL	Temperatur-Feuchtigkeitssensor	I2C
SDA	Temperatur-Feuchtigkeitssensor	I2C
D5	Regensensor	Digital 5
A0	Bodenfeuchtigkeitssensor	Analog 0
A1	Regensensor	Analog 1

Tabelle 8 PIN Belegung Sender

5.1.3 Aufbau Sender Holzbrett

Der I2C Temperatur- und Feuchtigkeitssensor, der analoge Bodenfeuchtigkeitssensor, der analog/digitale Regensensor und das Dragino-LoRa-Shield werden im zuerst auf dem Arduino MEGA aufgeschaltet.

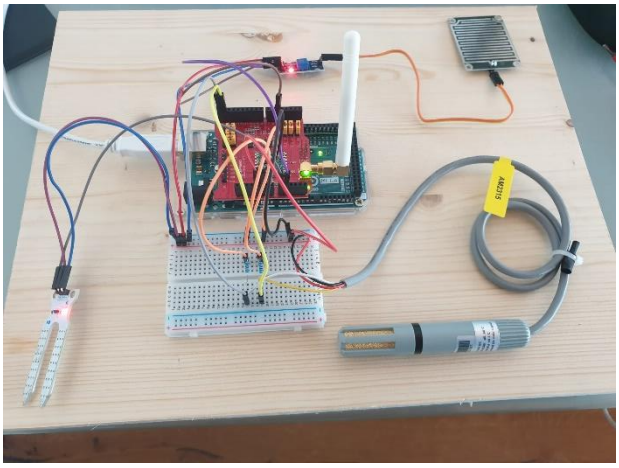


Abbildung 5-4 Sensorwerte auf Sender

Alle Komponenten wurden auf dem Holzbrett anhand des Schemas verkabelt und in Betrieb genommen. Das Dragino-LoRa-Shield wird auf das Arduino aufgesteckt.

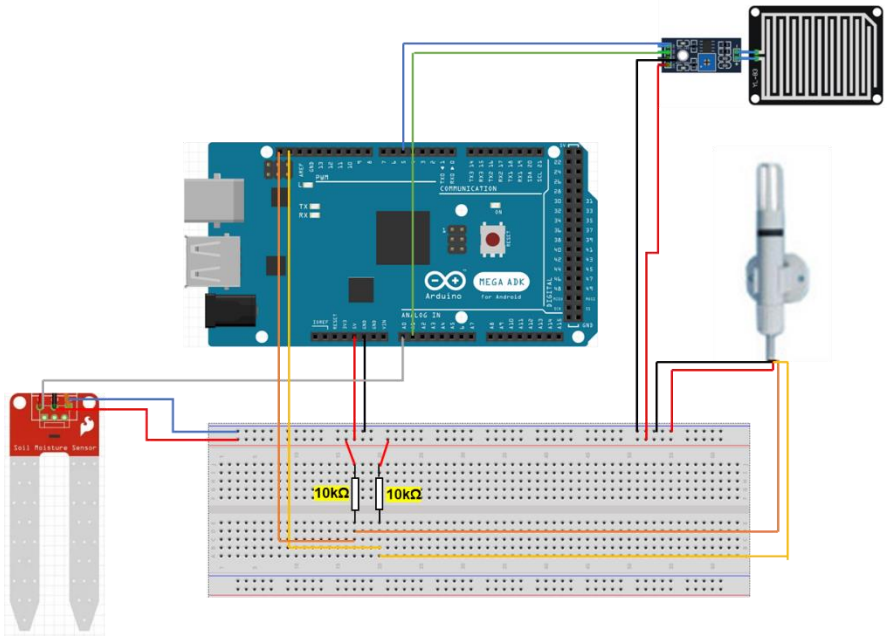


Abbildung 5-5 Anschlussschema Sensoren Sender

5.1.4 Sensoren am seriellen Monitor

Der Sketch zu den Sensoren soll alle Werte auf dem seriellen Monitor wiedergeben. Die Daten vom Regensensor und Bodenfeuchtigkeitssensor werden in den Messwerten des analogen Eingangs angezeigt.

Der erste Sketch ist wie folgt aufgebaut:

- Die Bibliothek Wire.h ermöglicht dem Arduino mit Geräten zu kommunizieren, welche das I²C-Protokoll verwenden.
  Adafruit_AM2315.h ist die Bibliothek für den AM2315 Feuchte - und Temperatursensor.

- Im void setup werden die Pins A0 und A1 als Inputs definiert.
  Die serielle Kommunikation wird gestartet sowie der AM2315 initialisiert.

- Im loop wird der Temperatur- und Luftfeuchtigkeitswert in einer float Variable gespeichert. Float ist ein Datentyp für Fließkommazahlen, also eine Zahl mit Dezimalpunkt. Fließkommazahlen wie beispielsweise 21.10 haben eine höhere Auflösung als ganze Zahlen. Dieser Datentyp wird häufig zur Annäherung an analoge und kontinuierliche Werte verwendet. Sie werden als 32 Bit (4 Byte) Informationen gespeichert.
  Die Messwerte des Bodensensors und des Regensensors werden in einem Byte Data Typ gespeichert. Da Byte sich um einen Typ ohne Vorzeichen handelt, kann er keine negative Werte darstellen. Das ist für das Garden Monitoring nicht relevant, da die Station nur in den warmen Monaten eingesetzt wird.

  Im nächsten Schritt wird mit am2315.readTemperature() die Temperatur definiert. Der AM2315 braucht eine Verzögerung von 2000ms um seine Werte zu aktualisieren, danach kann mit am2315.readHumidity() die Luftfeuchtigkeit definiert werden.

  Am Schluss werden die Werte am seriellen Monitor ausgeben und nach einer Verzögerung von 2000ms startet der loop wieder.

temperatur_boden_regen

```
1  #include <Wire.h>
2  #include <Adafruit_AM2315.h>
3
4  Adafruit_AM2315 am2315;
5
6  void setup()
7  {
8   pinMode(A0, INPUT);
9   pinMode(A1, INPUT);
10
11  Serial.begin(9600);
12  am2315.begin();
13 }
14 void loop()
15 {
16  float Temperatur;
17  float Luftfeuchtigkeit;
18  byte Bodensensor = analogRead(A0);
19  byte Regensensor = analogRead(A1);
20
21
22  Temperatur = am2315.readTemperature();
23  delay(2000);
24  Luftfeuchtigkeit = am2315.readHumidity();
25
26  Serial.print("Temperatur C: ");
27  Serial.println(Temperatur);
28
29  Serial.print("Luftfeuchtigkeit %: ");
30  Serial.println(Luftfeuchtigkeit);
31
32  Serial.print("Bodenfeuchtigkeit:");
33  Serial.println(Bodensensor);
34
35  Serial.print("Regensensor");
36  Serial.println(Regensensor);
37
38  delay(2000);
39 }
```

- Bibliotheken einbinden
- Input Pins definieren
- serielle Kommunikation starten
- am2315. initialisieren
- Lesen Sensorwerte
- Temperatur anzeigen am seriellen Monitor
- Luftfeuchtigkeit anzeigen am seriellen Monitor
- Bodensensor anzeigen am seriellen Monitor
- Regensensor anzeigen am seriellen Monitor
- Verzögerung von 2000ms

Abbildung 5-6 Sketch Sensoren Sender 1

Im ersten Bild des seriellen Monitors sind alle Sensorwerde ersichtlich. Die Fühler des Boden- und Regensensors sind trocken.

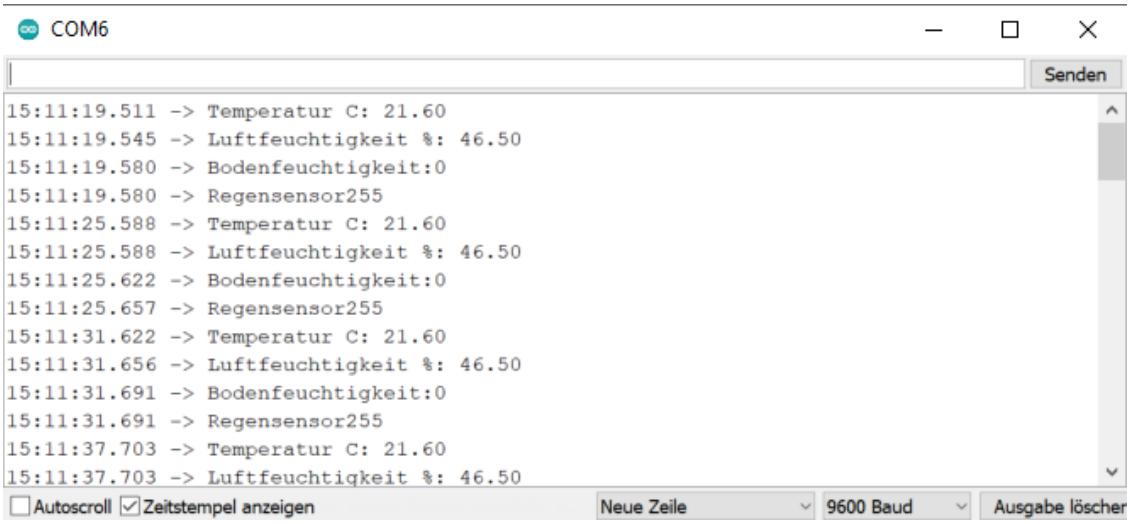


Abbildung 5-8 Serieller Monitor trockene Sensoren

Beim nachfolgenden Bild des seriellen Monitors werden die Fühler des Boden- und Regensensors befeuchtet. Die Messwerte der analogen Eingänge verändern sich.

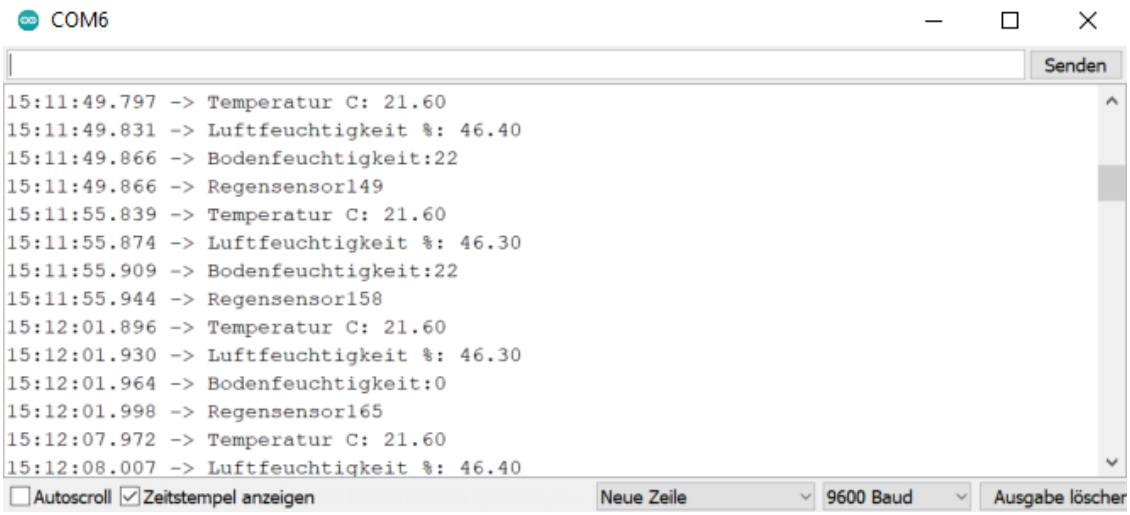


Abbildung 5-9 Serieller Monitor feuchte Sensoren

5.1.5 Aufbau Empfänger

Das HMI Display von Nextion sowie das Dragino-LoRa-Shield werden auf das Empfänger Arduino UNO aufgeschaltet. Ausserdem sind alle Komponenten bereits in der definitiven Halterung montiert.

Abbildung 5-10 Aufbau Empfänger

Anhand des Schemas wird das HMI Nextion Display an den Mikrocontroller angeschlossen. Das Dragino-LoRa-Shield wird auf das Arduino aufgeschaltet.

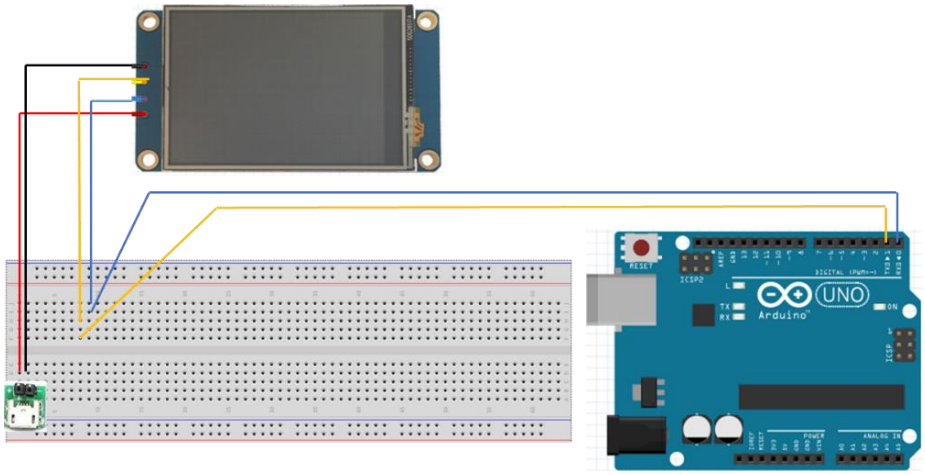


Abbildung 5-11 Schema Empfänger

5.1.6 Sensorwerte Senden/Empfangen

Der Sketch zum Senden und Empfangen, über die Punkt-zu-Punkt Verbindung mit dem Dragino-LoRa-Shield, soll alle Sensorwerte auf dem seriellen Monitor wiedergeben. Die Daten vom Regensensor und Bodenfeuchtigkeitssensor werden in den Messwerten des analogen Eingang angezeigt.

Der Sketch des Empfängers ist wie folgt aufgebaut:

- Die Bibliothek SPI.h, die für eine schnelle Kommunikation zwischen zwei Mikrocontrollern eigesetzt wird, wird für den Empfänger gebraucht.
  Mit der Bibliothek RH_RF95.h wird das Dragino-LoRa-Shield eingebunden.

- Im void loop startet die serielle Kommunikation und der rf95 wird initialisiert.

- Im void loop werden die Daten, wie im Beispielsketch von der RadioHead Bibliothek, standartmässig mit dem Datentyp uint8_t versendet. Es ist eine universeller Datentyp, welcher auch in anderen Programmiersprachen verwendet wird. In der Programmierung des Arduino ist ein unit8_t als byte zu verstehen. Leider wird der ursprüngliche float-Datentyp beim Empfänger in ein byte umgewandelt, was zur Folge hat, dass am seriellen Monitor nur eine ganze Zahl ausgegeben wird.

  Am Anfang wird buf definiert und die maximale Länge eingestellt, anschliessend werden die buf gelesen.

  Am Schluss werden die Werte am seriellen Monitor ausgeben und nach einer Verzögerung von 2000ms startet der loop wieder.

Empfaenger_mit_seriellen_Monitor §

```
1  #include <SPI.h>
2  #include <RH_RF95.h>
3
4  RH_RF95 rf95;
5
6  void setup()
7  {
8    Serial.begin(9600);
9    rf95.init();
10 }
11
12 void loop()
13 {
14 if (rf95.available())
15 {
16 uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
17 uint8_t len = sizeof(buf);
18 if (rf95.recv(buf, &len))
19 {
20  byte Temperatur = buf[0];
21  byte Luftfeuchtigkeit = buf[1];
22  byte bodensensor = buf[2];
23  byte regensensor = buf[3];
24
25  Serial.print("Temperatur °C:    ");
26  Serial.println(Temperatur);
27
28  Serial.print("Luftfeuchtigkeit %:    ");
29  Serial.println(Luftfeuchtigkeit);
30
31  Serial.print("Bodensensor:    ");
32  Serial.println(bodensensor);
33
34
35  Serial.print("Regensensor:    ");
36  Serial.println(regensensor);
37
38  }
39 delay(2000);
40 }
41 }
```

- Bibliotheken einbinden
- serielle Kommunikation starten
- Rf95 initialisieren
- buf definieren
- Länge von buf definieren
- Werte aus buf lesen
- Temperatur anzeigen am seriellen Monitor
- Luftfeuchtigkeit anzeigen am seriellen Monitor
- Bodensensor anzeigen am seriellen Monitor
- Regensensor anzeigen am seriellen Monitor
- Verzögerung von 2000ms

Abbildung 5-12 Sketch Sensorwerte empfangen

Die Programmierung des Senders ist eine Erweiterung des Sketch «Sensoren am seriellen Monitor».

Der Sketch des Senders ist wie folgt aufgebaut:

- Die Bibliotheken SPI.h, Adafruit_AM2315.h und RH_RF95.h werden eingebunden.

- Im void setup werden die Inputs definiert und der am2315 sowie der rf95 initialisiert.

- Im void loop werden wie im ersten Sender Sketch die Sensordaten in float und byte Datentypen gespeichert.

  Die ausgelesenen Werte werden nun in das uint_8-array message gespeichert und anschliessend gesendet.

  Zum Abschluss des loop ist wieder eine Verzögerung von 2000ms eingebaut.

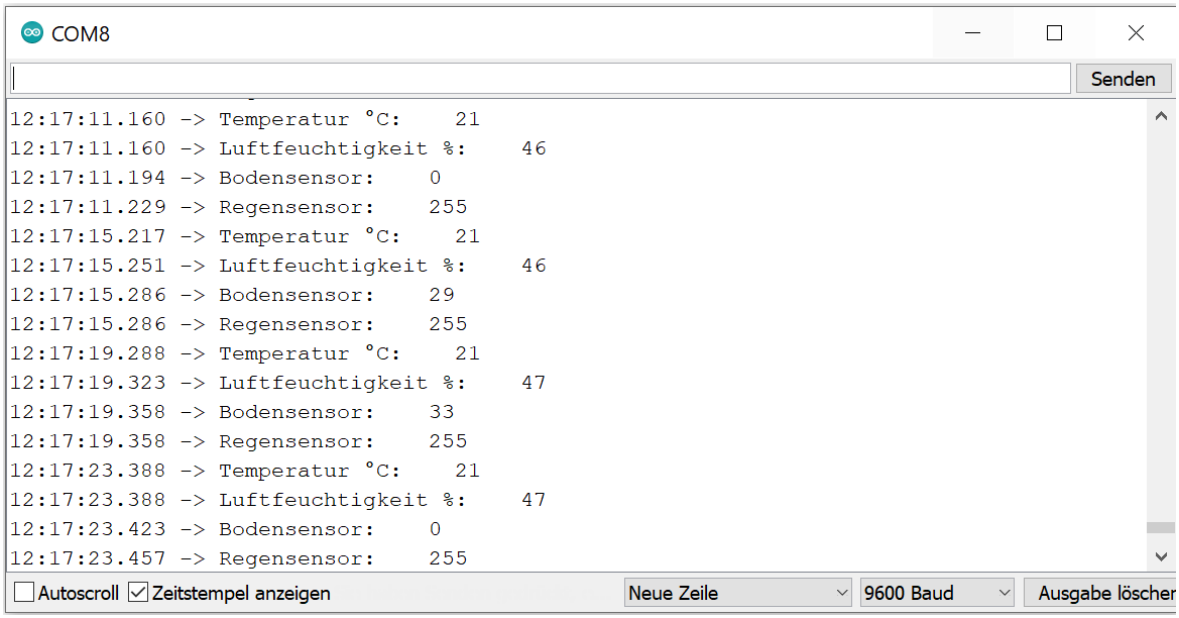


Abbildung 5-13 Serieller Monitor Sensorwerte empfangen

Sender_serieller_Monitor §

```
1  #include <SPI.h>
2  #include <Adafruit_AM2315.h>
3  #include <RH_RF95.h>;
4
5
6  RH_RF95 rf95;
7  Adafruit_AM2315 am2315;
8
9  void setup()
10 {
11 pinMode(A0, INPUT);
12 pinMode(A1, INPUT);
13
14 rf95.init();
15 am2315.begin();
16 }
17 void loop()
18
19 {
20 delay(2000);
21 float Temperatur;
22 float Luftfeuchtigkeit;
23 byte Bodensensor = analogRead(A0);
24 byte Regensensor = analogRead(A1);
25
26
27 Temperatur = am2315.readTemperature();
28 delay(2000);
29 Luftfeuchtigkeit = am2315.readHumidity();
30
31 uint8_t message[4];
32
33 message[0] = Temperatur;
34 message[1] = Luftfeuchtigkeit;
35 message[2] = Bodensensor;
36 message[3] = Regensensor;
37
38
39 rf95.send(message, sizeof(message));
40 rf95.waitPacketSent();
41 }
```

- Bibliotheken einbinden
- Input Pins definieren
- Am2315. initialisieren
- Rf95 initialisieren
- Verzögerung 2000ms
- Lesen der Sensorwerte
- message befüllen
- Message senden

Abbildung 5-14 Sketch Sensorwerte senden

5.2. HMI

Das Nextion Touchdisplay wird mit dem Nextion-Editor programmiert. Der Editor kann kostenlos von der Homepage von Nextion heruntergeladen werden.
Die Programmierung von Textfeldern, Nummernfeldern, verschiedenen Seiten etc. wird mit dem Editor erstellt und mit einer microSD Karte auf das Display geladen. Nach einem Neustart des Displays kann die microSd Karte entfernt werden, da die Programmierung auf dem Display gespeichert ist.
Das Touch-Display wird über die serielle Schnittstelle RX (Empfangen) und TX(Senden) mit dem Arduino verbunden. Die wechselnden Sensordaten werden somit über diese Schnittstelle geschickt. Wichtig ist, dass die RX/TX Schnittstellen zwischen dem µC und dem Display müssen gekreuzt werden, da beide Teilnehmer die Daten auf dem RX-Port empfangen und Daten über den TX-Port senden. Die Baudrate ist standartmässig auf 9600 eingestellt was für das Projekt Garden Monitoring ausreichend ist.

Der Befehl des Nextion Displays besteht immer aus einem Klartextanweisung.
Ein Beispiel ist t1.txt=“Temperatur“ und dazu die Stopp-Sequenz “0xff 0xff 0xff“. Die Stopp-Sequenz ist notwendig und das Nextion Display erkennt, wann der Befehl zu Ende ist. Diese Sequenz entspricht dezimal : 255 255 255 und ist aus diesem Grund so definiert, weil die 255 nicht mehr im Definitionsbereich des einfachen Zahlensatzes liegt. Somit erkennt das Display die Stopp-Sequenz.

5.2.1 Nextion-Editor

Der Nextion Editor ist wie folgt aufgebaut:

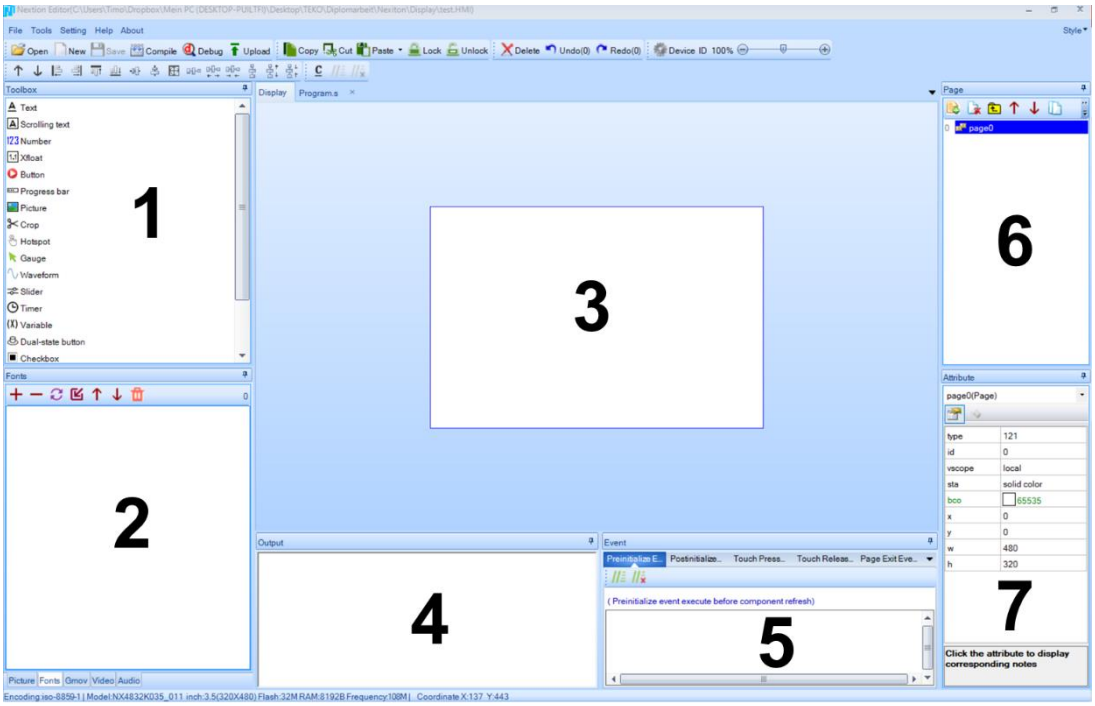


Abbildung 5-15 Nextion-Editor

Nr:	Funktion
1	Mit der Toolbox können alle verfügbare Komponenten ausgewählt werden.
2	Hier findet man die Ressourcen wie Bilder oder Zeichensätze.
3	Im Bearbeitungsbereich sieht man das Display.
4	Im Compiler findet man allfällige Fehler.
5	Hier wird der Code für die Eventbehandlung angegeben.
6	Hier sieht man alle Seiten
7	Hier sieht man die Eigenschaften der ausgewählten Komponenten.

Tabelle 9 Nextion-Editor

5.2.2 Ressourcen

Im Ressourcen Speicher werden alle Zeichensätze (Fonts) und Bilder gespeichert. Für das Garden Monitoring wurden folgende Fonts definiert:

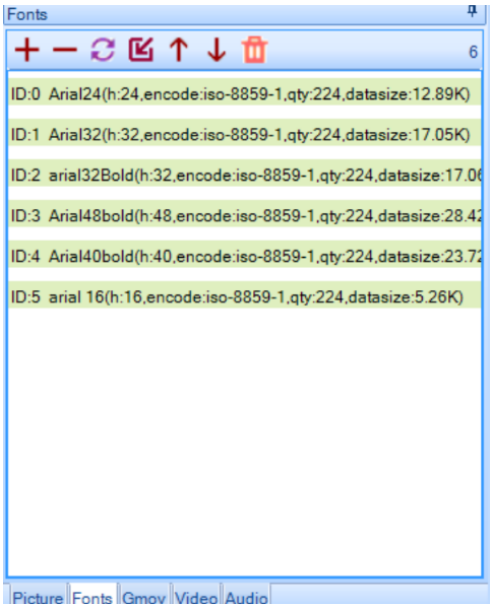


Abbildung 5-16 Ressourcen Fonts

5.2.3 HMI Seiten

Für das Überwachungssystem werden zwei Seiten auf dem Display benötigt. Der Start-bildschirm ist eine kleine Titelseite, welche nur ein Button programmiert hat. Der Button hat die Funktion auf die zweite Seite mit den Sensoren und dem optischen Alarm zu wechseln. Die Buttons auf den Seiten werden mit dem Touch Release Event umgesetzt, das bedeutet sobald der Button losgelassen wird werden die Seiten gewechselt.

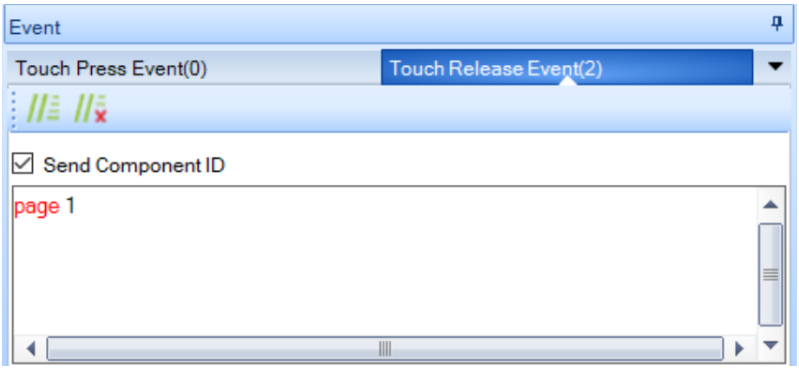


Abbildung 5-17 Touch Release Event

Auf dem Display des Garden Monitorings werden die folgenden gezeigten Seiten verwendet.

Abbildung 5-18 Seite 1 Display

Objektname	Typ	Eigenschaften
t0	Textfeld	txt="Garden Monitoring"
t1	Textfeld	txt="Diplomarbeit TEKO Timo Fankhauser"
b0	Button	Touch Release Event txt="Sensoren".

Tabelle 10 Komponenten Seite 1

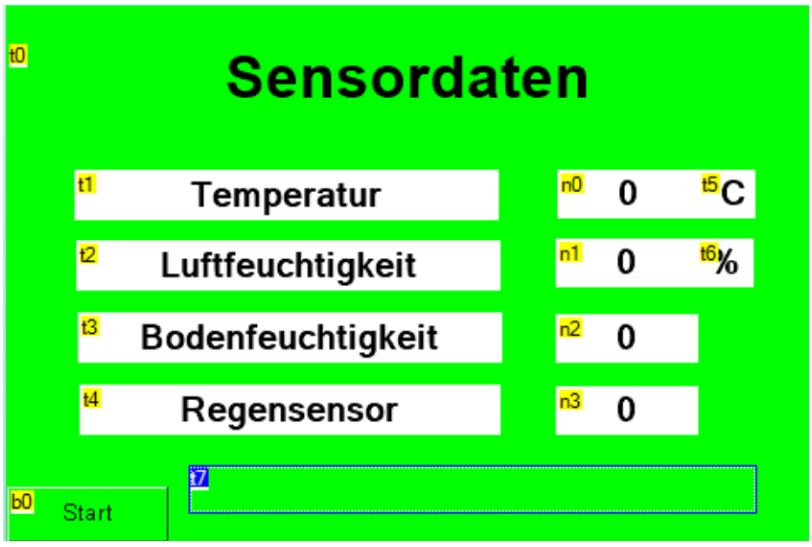


Abbildung 5-19 Seite 2 Display

Objektname	Typ	Eigenschaften
t0	Textfeld	txt=“Sensordaten“
t1	Textfeld	txt=“Temperatur“
t2	Textfeld	txt=“Luftfeuchtigkeit“
t3	Textfeld	txt=“Bodenfeuchtigkeit“
t4	Textfeld	txt=“Regensensor“
t5	Textfeld	txt=“°C“
t6	Textfeld	txt=“%“
t7	Textfeld	txt=““ (wird vom Arduino verändert)
n0	Nummernfeld	val=“Temperatur“ (wird vom Arduino verändert)
n1	Nummernfeld	val=“Luftfeuchtigkeit“ (wird vom Arduino verändert)
n2	Nummernfeld	val=“Bodensensor“ (wird vom Arduino verändert)
n3	Nummernfeld	val=“Regensensor“ (wird vom Arduino verändert)
b0	Button	Touch Release Event txt=“Start“

Tabelle 11 Komponenten Seite 2

5.2.4 Debugg

Um die Programmierung des Displays zu testen gibt es eine hilfreiche Funktion auf dem Nextion-Editor, den Debugger. Auf dem Debugg Menu kann man einen Simulationsmodus starten, auf welchem man manuelle Werte eingeben kann.

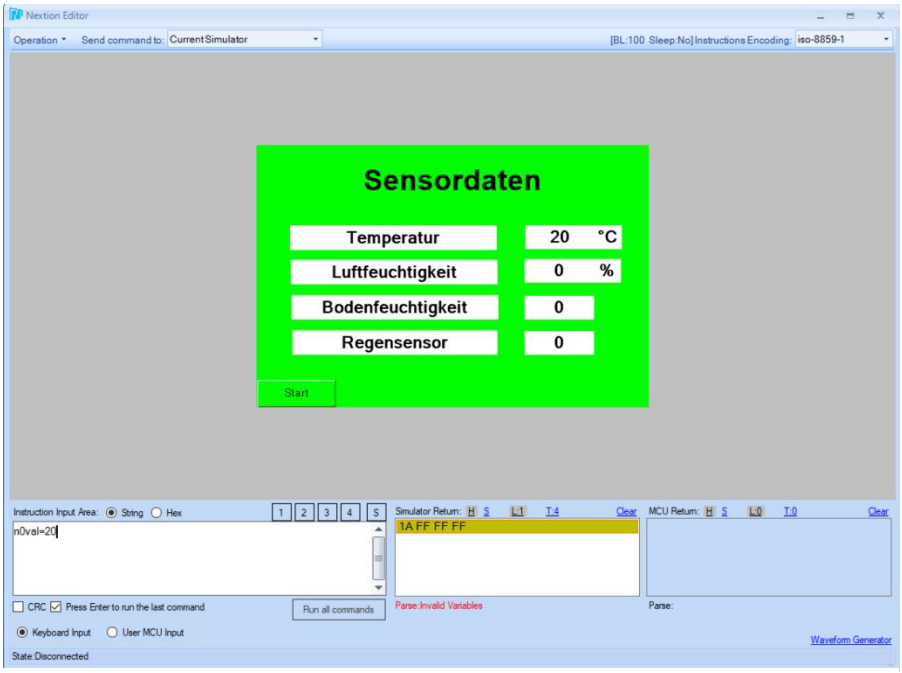


Abbildung 5-20 Debugg Nextion-Editor

5.2.5 Test HMI

Im nächsten Schritt wird die Programmierung des Displays mit der Funktion der seriellen Schnittstelle des Arduino UNO getestet

Folgender kurzer Sketch wurde dafür erstellt:

Test_HMI §

```
1  byte Temperatur = 20;
2  byte Luftfeuchtigkeit = 41;
3  byte Regensensor = 80;
4  byte Bodenfeuchtigkeit = 24;
5  byte endsequenz[3] = {255, 255, 255};
6
7  void setup()
8  {
9  Serial.begin(9600);
10
11 }
12
13 void loop()
14 {
15
16   Serial.print("n0.val=");
17   Serial.print(Temperatur);
18   Serial.write(endsequenz, 3);
19
20   Serial.print("n1.val=");
21   Serial.print(Luftfeuchtigkeit);
22   Serial.write(endsequenz, 3);
23
24   Serial.print("n2.val=");
25   Serial.print(Bodenfeuchtigkeit);
26   Serial.write(endsequenz, 3);
27
28   Serial.print("n3.val=");
29   Serial.print(Regensensor);
30   Serial.write(endsequenz, 3);
31
32 }
```

- Werte setzen
- Endsequenz setzen
- Serielle Kommunikation starten
- Temperatur schreiben an serieller Schnittstelle
- Luftfeuchtigkeit schreiben an serieller Schnittstelle
- Bodensensor schreiben an serieller Schnittstelle
- Regensensor schreiben an serieller Schnittstelle

Abbildung 5-21 Sketch Test HMI

Das byte array endsequenz wurde so definiert, damit nicht immer wieder die Sequenz 0xff 0xff 0xff gesendet werden muss.

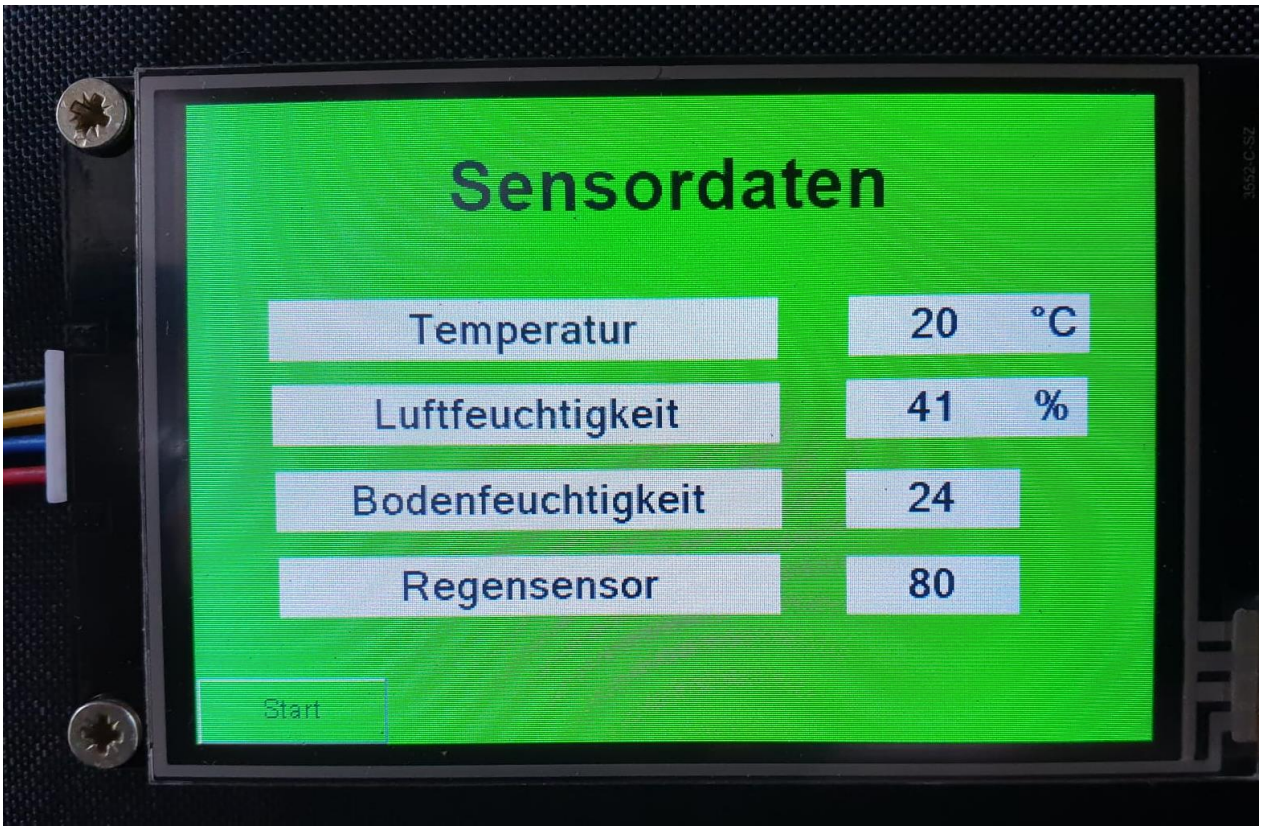


Abbildung 5-22 Dispaly Sketch HMI

5.3. Funktionen mit HMI

Mit der anschliessenden Programmierung des Empfängers werden alle Funktionen bis zu diesem Stand getestet. Die Sensorwerte werden vom Sender auf den Empfänger übermittelt und anschliessend über die serielle Schnittstelle auf dem Display angezeigt. Der Sketch wurde aus den Programmierungen «Sensorwerte Senden/Empfangen» und «Test HMI» zusammengefügt.

Empfaenger_mit__HMI §

```
1  #include <SPI.h>
2  #include <RH_RF95.h>
3
4  RH_RF95 rf95;
5
6  byte endsequenz[3] = {255, 255, 255};
7
8  void setup()
9  {
10   Serial.begin(9600);
11   rf95.init();
12 }
13
14 void loop()
15 {
16   if (rf95.available())
17   {
18     uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
19     uint8_t len = sizeof(buf);
20    if (rf95.recv(buf, &len))
21    {
22    byte Temperatur = buf[0];
23    byte Luftfeuchtigkeit = buf[1];
24    byte bodensensor = buf[2];
25    byte regensensoranalog = buf[3];
26
27
28    Serial.print("n0.val=");
29    Serial.print(Temperatur);
30    Serial.write(endsequenz, 3);
31
32    Serial.print("n1.val=");
33    Serial.print(Luftfeuchtigkeit);
34    Serial.write(endsequenz, 3);
35
36    Serial.print("n2.val=");
37    Serial.print(bodensensor);
38    Serial.write(endsequenz, 3);
39
40    Serial.print("n3.val=");
41    Serial.print(regensensoranalog);
42    Serial.write(endsequenz, 3);
43    }
44    delay(2000);
45 }
```

- Bibliotheken einbinden
- byte endsequenz setzen
- serielle Kommunikation starten
- Rf95 initialisieren
- buf definieren
- Länge von buf definieren
- Werte aus buf lesen
- Temperatur an HMI Nummernfeld n.0 senden
- Luftfeuchtigkeit an HMI Nummernfeld n.1 senden
- Bodensensor an HMI Nummernfeld n.2 senden
- Regensensor an HMI Nummernfeld n.3 senden
- Verzögerung von 2000ms

Abbildung 5-23 Sketch Funktionen mit HMI

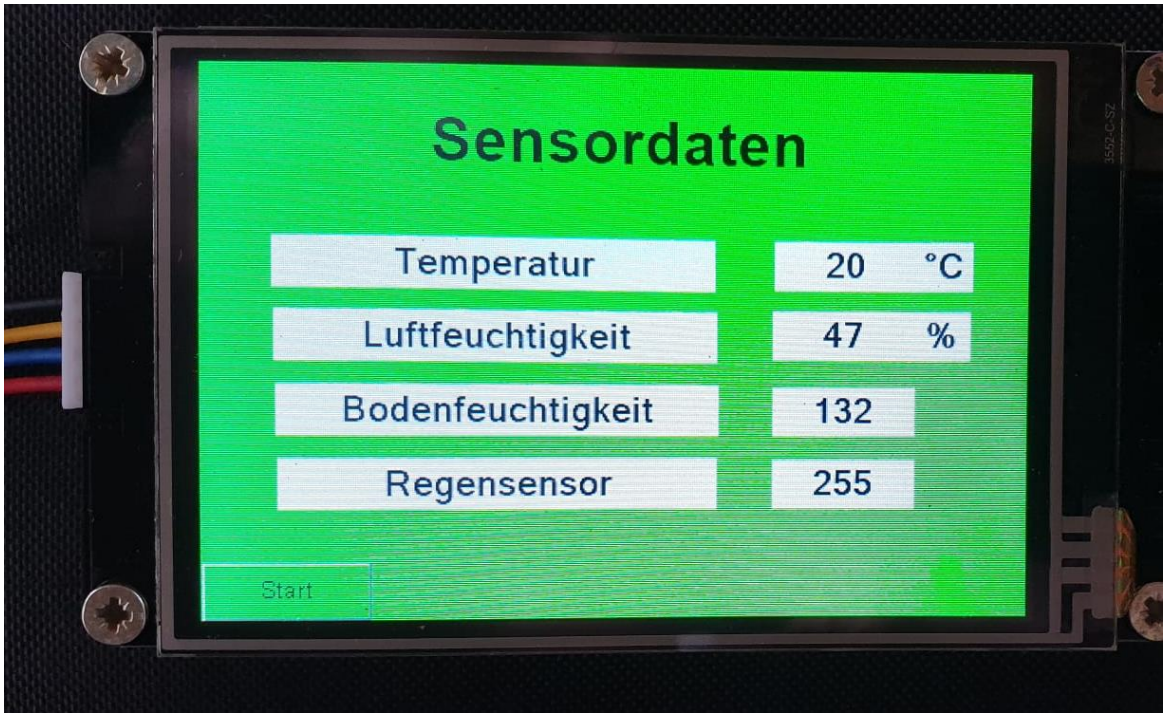


Abbildung 5-24 Display Funktionen mit HMI

5.4. Optischer Alarm

Die folgende Programmierung ist eine Erweiterung des letzten Codes «Funktionen mit HMI». Es werden nur die geänderten Programmzeilen gezeigt.
Neu wird ein int für den minimalen Messwert des Bodensensors definiert und die boolische Funktion bodefgt_gut eingebettet. Mit der if und else Anweisung wird der Messwert ausgewertet und an die serielle Schnittstelle gesendet. Ist der Messwert höher als 119 erscheint auf dem Display kein Alarm. Sobald der Messwert unter 120 liegt, wird auf dem Display «Alam Bodenfeuchtigkeit!» angezeigt.

Empfaenger_mit__HMI_Alarm §

```
1  #include <SPI.h>
2  #include <RH_RF95.h>
3
4  RH_RF95 rf95;
5
6  int bodenfgt_min = 120;
7  bool bodenfgt_gut;
8  byte endsequenz[3] = {255, 255, 255};
9
```

- Bibliotheken einbinden
- variablen definieren
- byte endsequenz setzen

```
49       if (bodensensor >= bodenfgt_min)
50         {
51         Serial.print("t7.txt=\"\"");
52         Serial.write(endsequenz, 3);
53         bodenfgt_gut = true;
54         }
55        else
56         {
57          Serial.print("t7.txt=\"Alarm Bodenfeuchtigkeit!\"");
58          Serial.write(endsequenz, 3);
59          bodenfgt_gut = false;
60          }
```

- Bodensensorwerte vergleichen
- Text an HMI Textfeld t7 senden
- bool bodenfgt_gut definieren

Abbildung 5-25 Sketch optischer Alarm

Abbildung 5-26 Display optischer Alarm

5.5. Zusätzliche Optionen

Es können zwei Kann Ziele des Projekts Garden Monitoring umgesetzt werden. Die Statusüberwachung wird mit Hilfe eines LED angezeigt, zusätzlich wurde eine eigene Energiequelle für den Sender umgesetzt.

5.5.1 Statusüberwachung LED

Für die Statusüberwachung wird wiederum der vorgängige Sketch erweitert. Es werden nur die geänderten Programmzeilen gezeigt.
Immer wenn der Sender eine Packet mit den Sensorwerten sendet, blinkt für eine kurze Zeit eine grüne LED beim Empfänger. Somit hat man die Gewissheit, dass der Sender noch funktioniert.
Es wird ein int für die LED sowie ein int für das Packet definiert. Jedes mal wenn ein Packet beim Empfänger ankommt wird das int Packet =1 geschaltet und der Ausgang 3 mit dem angeschlossenen LED auf HIGH gestellt, somit leuchtet die LED auf. Am Schluss des Sketchs schickt der Empfänger ein Antwortpacket an den Sender und der Ausgang 3 wird auf LOW gestellt, somit wird die LED ausgeschaltet.

Sketch_Verbindugsueberwachung §

```
1  #include <SPI.h>
2  #include <RH_RF95.h>
3
4  RH_RF95 rf95;
5
6  int led_betrieb = 3;
7  int packet = 0;
8  int bodenfgt_min = 120;
9  bool bodenfgt_gut;
10 byte endsequenz[3] = {255, 255, 255};
11
19 void loop()
20 {
21   if (rf95.available())
22   {
23     uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
24     uint8_t len = sizeof(buf);
25    if (rf95.recv(buf, &len))
26    {
27     packet = 1;
28     digitalWrite(led_betrieb, HIGH);
29     byte Temperatur = buf[0];
30     byte Luftfeuchtigkeit = buf[1];
31     byte bodensensor = buf[2];
32     byte regensensoranalog = buf[3];

65          uint8_t data[] = "Antwortpacket";
66          rf95.send(data, sizeof(data));
67          rf95.waitPacketSent();
68          digitalWrite(led_betrieb, LOW);
```

- Bibliotheken einbinden
- Variablen definieren
- Variablen Pins zuweisen
- byte endsequenz setzen
- buf definieren
- Länge von buf definieren
- Packet =1
- led_betrieb einschalten
- Werte aus buf lesen
- Antwort senden
- led_betrieb aus

Abbildung 5-27 Sketch Verbindungsüberwachung

5.5.2 Energiequelle Sender

Die Versorgung des Senders wird mit einem Solar Power Management Modul, einem 14500 3.7V Li-Ion Akku 750mAh und der Solarzelle 12V 250mA 3W sichergestellt. Der Aufbau der Energiequelle sieht wie folgt aus:

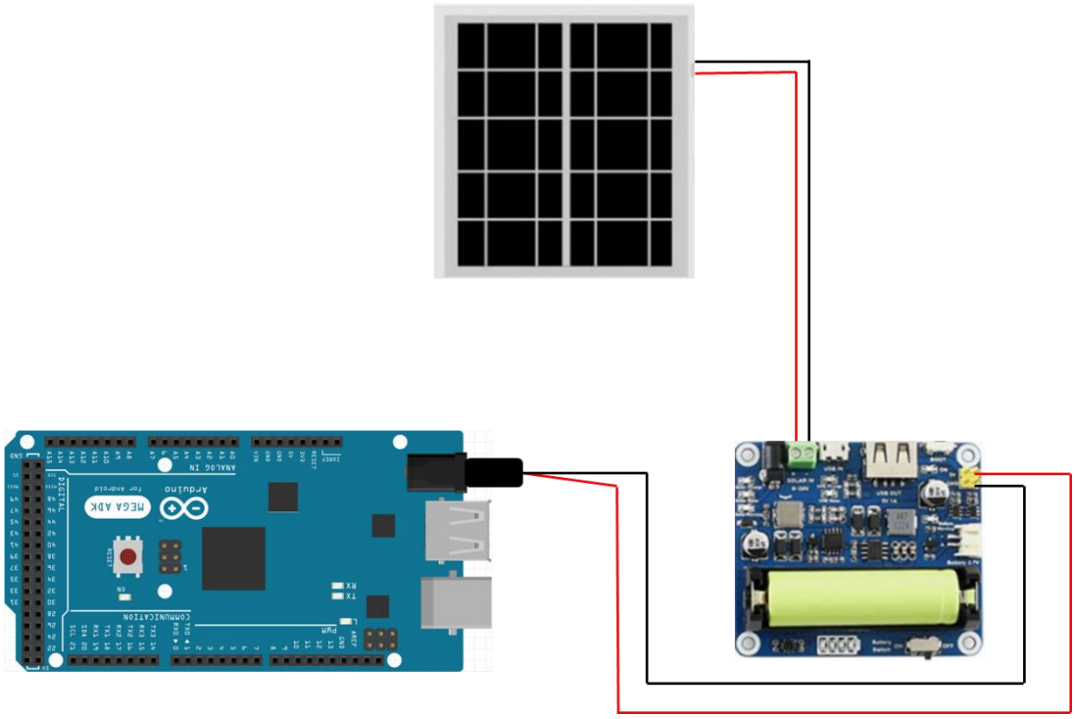


Abbildung 5-28 Schema Energieversorgung

Zusätzlich kann mit der LowPower.h Bibliothek beim Sender Strom gespart werden. Der Sender wird für 8s in einen Sleep Modus gesetzt. Dass heisst, dass alle Sensoren stromlos sind und das Arduino MEGA für diese Zeit im Energiesparmodus ist.

Sender_mit_HMI_Low_Power

```
1  #include <SPI.h>
2  #include <Adafruit_AM2315.h>
3  #include <RH_RF95.h>;
4  #include "LowPower.h"

43 LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
44 }
```

- Bibliothek LowPower.h einbinden
- Sleep Modus 8s

Abbildung 5-29 Sketch Low Power

5.6. Endprodukt

Der Sender des Garden Monitorings wurde in eine wetterfeste Aussenstation und der Empfänger ist in der vorgesehen Halterung montiert.

5.6.1 Sender

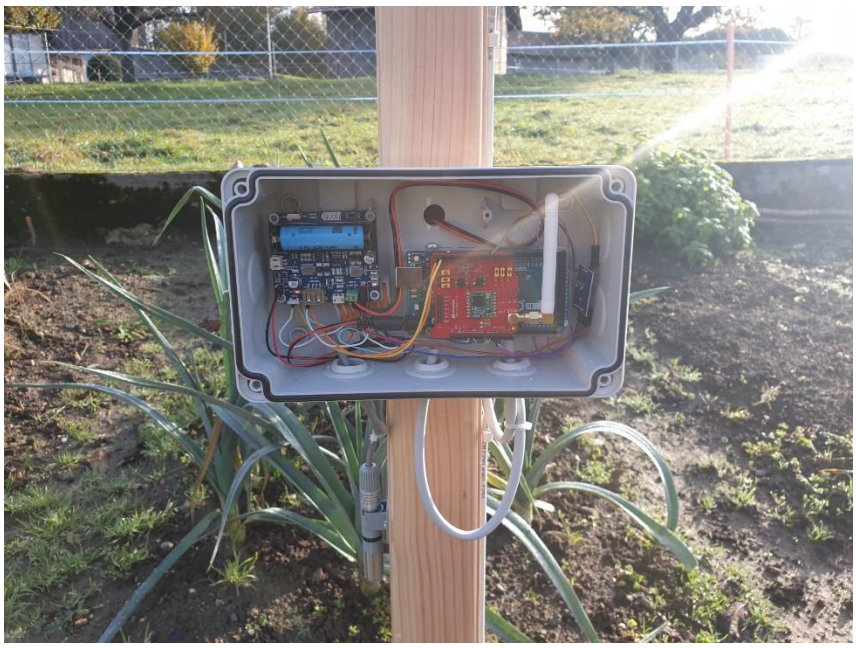


Abbildung 5-30 Endprodukt Sender mit Dose offen

Abbildung 5-31 Endprodukt Sender

5.6.2 Flussdiagramm Sender

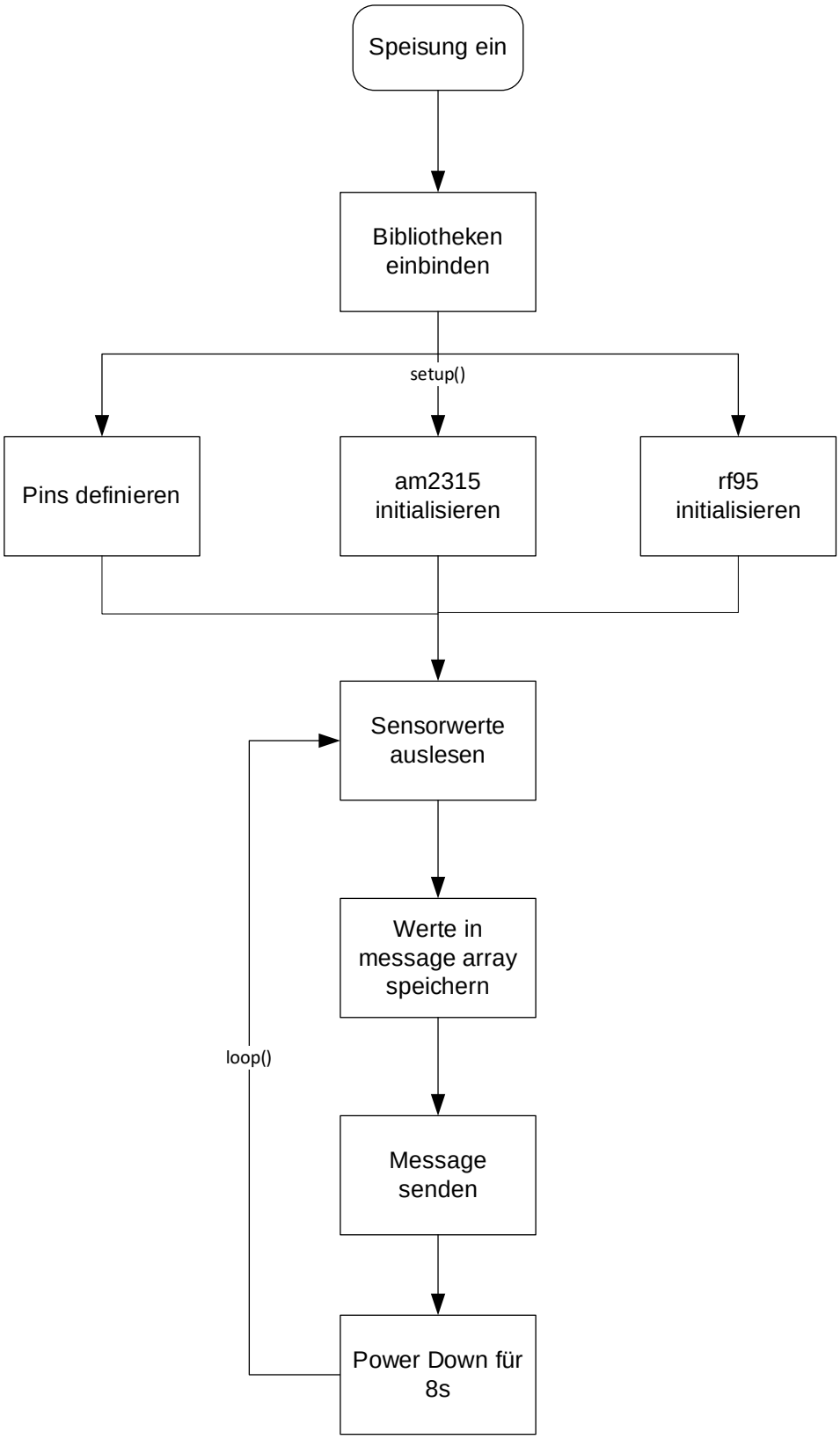


Abbildung 5-32 Flussdiagramm Software Sender

5.6.3 Empfänger

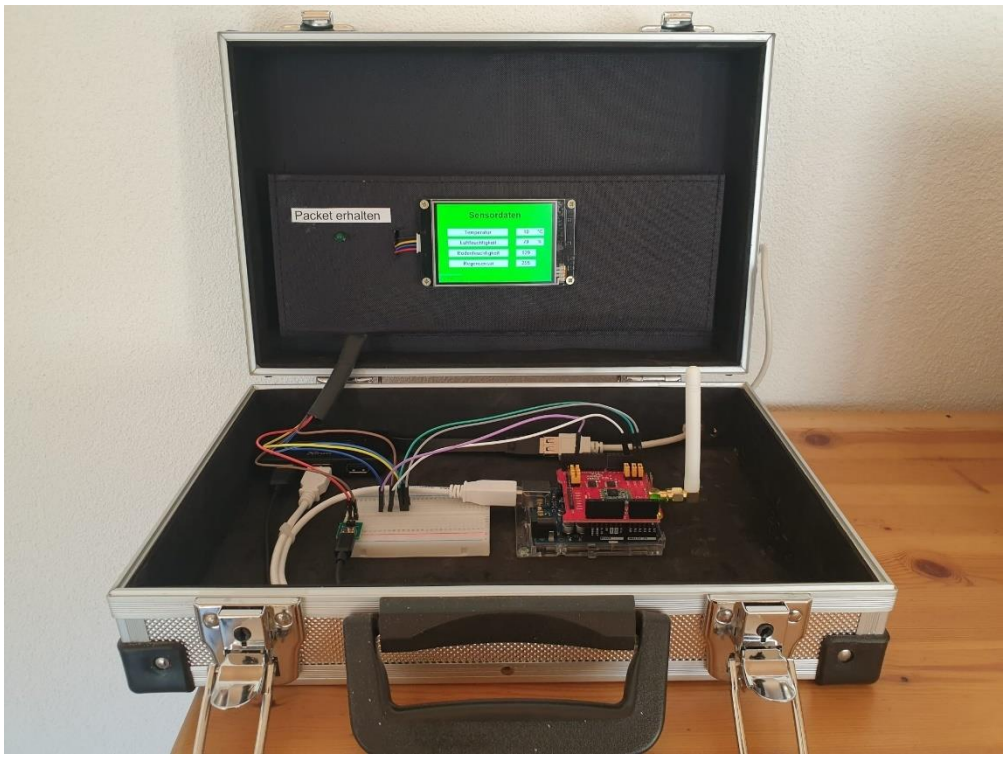


Abbildung 5-33 Endprodukt Empfänger

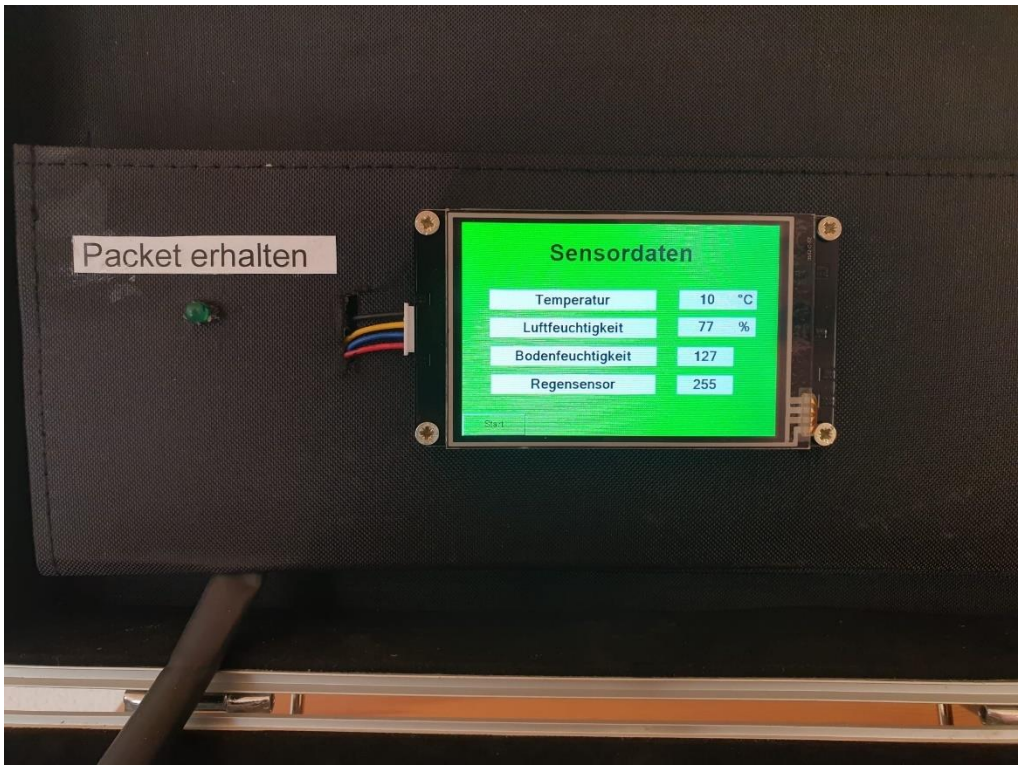


Abbildung 5-34 Endprodukt Sender Display und LED Verbindungsüberwachung

5.6.4 Flussdiagramm Empfänger

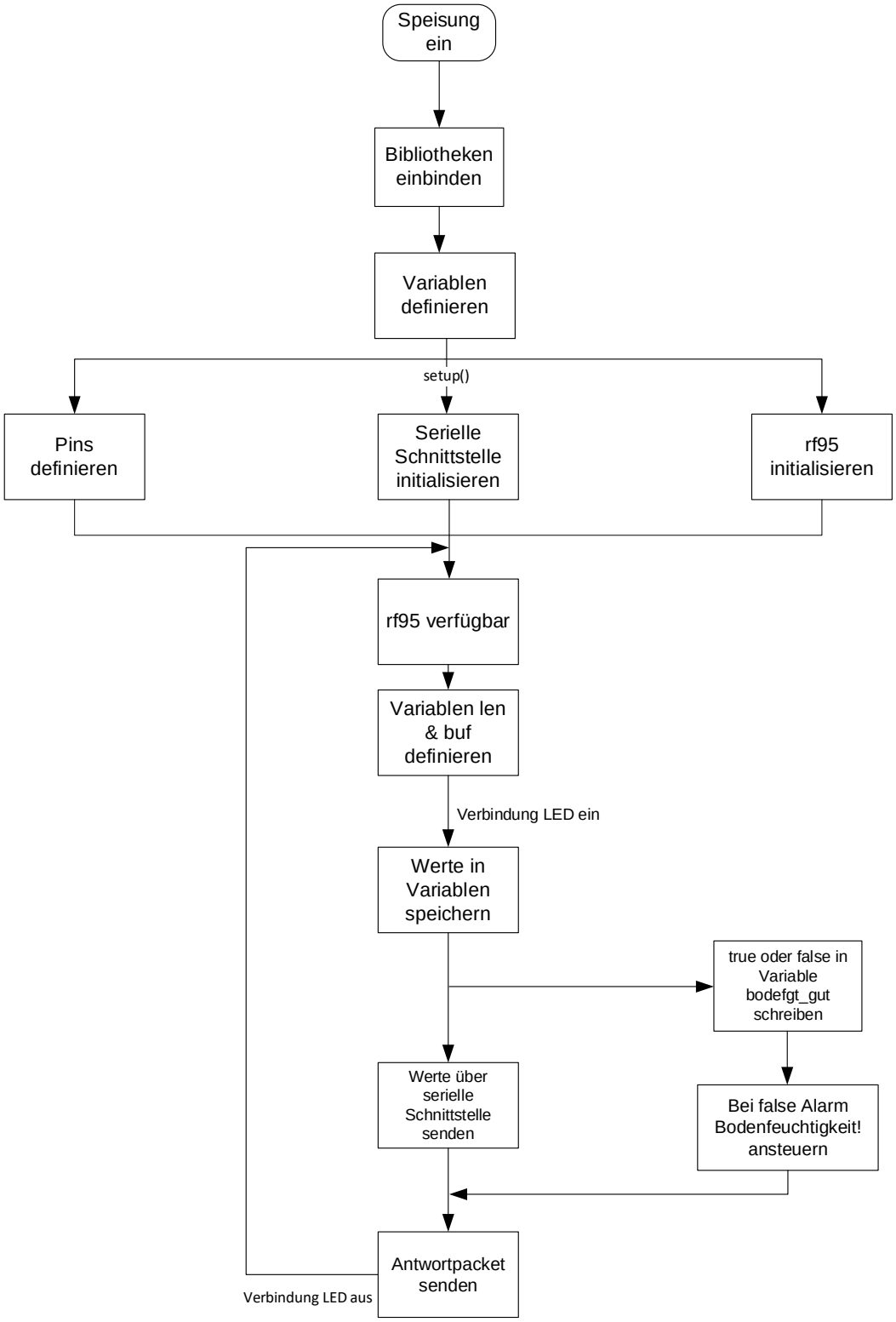


Abbildung 5-35 Flussdiagramm Software Empfänger

5.7. Test mit Endprodukt

Ob das Garden Monitoring einsetzbar ist, wird mit folgenden Tests überprüft und protokolliert.

5.7.1 Fehlerklassen

Die festgestellten Fehler werden in Klassen von 0-4 eingestuft.

Nummer	Fehlerklassen	Beschreibung
0	Fehlerfrei	Beim getesteten Modul/System wurden keine Fehler entdeckt. Es arbeitet zu 100% fehlerfrei.
1	Unwesentlicher Mangel	Es wurden Fehler entdeckt, die den Betrieb des Modul/System nicht beeinträchtigen, wie z.B. Schreibfehler.
2	Leichter Mangel	Es wurden Fehler entdeckt, welche noch nicht als kritisch angesehen werden müssen. Diese Fehler können aber nicht bestehen und müssen in absehbarer Zeit behoben werden.
3	Schwerer Mangel	Es wurden Fehler entdeckt, welche den Betrieb des Modul/System nicht mehr sicherstellen. Es sind Fehler die nicht sofort gelöst werden können.
4	Kritischer Mangel	Es wurden Fehler entdeckt, welche das gesamte Projekt beeinflussen. Solche Fehler sind als kritisch anzusehen und haben zum Teil so schwerwiegende Folgen, dass unter Umständen das ganze Projekt in Gefahr ist.

Tabelle 12 Fehlerklassen

5.7.2 Testfallbeschreibung

Jeder Testfall wir anhand folgender Tabelle durchgeführt.

ID/Bezeichnung	**T-XX**	**Titel**
Beschreibung	Beschreibung des Testfalls	
Testvoraussetzung	Text	
Durchführung	Beschreibung	
Durchläufe geplant	XX	
Erwartetes Ergebnis	Resultat	
Testmethode	Black-Box-Test	
Fehlerklasse	Fehlerklasse	
Festgestellter Fehler	Beschreibung	

Tabelle 13 Testfallbeschreibung

5.7.3 Testplan

Nummer	**Aktivität**	**Verantwortlich**	**Termin**
1	Vorbereitung	Timo Fankhauser	24.10.2020
2	Testing	Timo Fankhauser	26.10.2020
3	Abschluss	Timo Fankhauser	28.10.2020

Tabelle 14 Testplan

5.7.4 Testprotokoll

ID/Bezeichnung	T-01	Verbindung
Beschreibung	Die Verbindung über die originale Distanz zwischen dem Sender und dem Empfänger muss stabil sein. Es soll keine unerklärlichen Verbindungsunterbrüche geben.	
Testvoraussetzung	Sender und Empfänger müssen über die LoRa Punkt-zu-Punkt verbunden sein.	
Durchführung	Über einen Zeitraum von 2 Stunden werden alle 10s Sensorwerte vom Empfänger an den Sender übermittelt	
Durchläufe geplant	2	
Erwartetes Ergebnis	Da die Verbindung gemessen wurde sollten keine Verbindungsprobleme entstehen.	
Testmethode	Black-Box-Test	
Fehlerklasse	0	
Festgestellter Fehler	keine	

Tabelle 15 Testfall-01

ID/Bezeichnung	T-02	Optischer Alarm Bodenfeuchtigkeit
Beschreibung	Ist der Messwert beim Bodenfeuchtigkeitssensor kleiner als 120 muss auf dem HMI Display ein optischer Alarm angezeigt werden .	
Testvoraussetzung	Sender und Empfänger müssen über die LoRa Punkt-zu-Punkt verbunden sein.	
Durchführung	Der Bodenfeuchtigkeitssensor wird in das trockene Erdreich gesetzt, so muss der optische Alarm auf dem HMI Display sichtbar sein. Anschliessend wird das Erdreich befeuchtet und der Alarm muss sich quittieren.	
Durchläufe geplant	4	
Erwartetes Ergebnis	Die optische Alarmierung entspricht der Anforderung.	
Testmethode	Black-Box-Test	
Fehlerklasse	0	
Festgestellter Fehler	keine	

Tabelle 16 Testfall-02

ID/Bezeichnung	T-03	LED-Betrieb
Beschreibung	Sobald der Empfänger ein Sensorwertpacket empfängt, wird kurz die LED led_betrieb angesteuert.	
Testvoraussetzung	Sender und Empfänger müssen über die LoRa Punkt-zu-Punkt verbunden sein.	
Durchführung	Dem Sender wird die Speisung genommen, somit darf die led_betrieb beim Empfänger nicht mehr angesteuert werden.	
Durchläufe geplant	5	
Erwartetes Ergebnis	Die LED-Betrieb entspricht der Anforderung	
Testmethode	Black-Box-Test	
Fehlerklasse	0	
Festgestellter Fehler	keine	

Tabelle 17 Testfall-03

ID/Bezeichnung	T-04	Energieversorgung Sender
Beschreibung	Der Sender wird mit einer Photovoltaik-Solarzelle und einem 3.7V Li-Ion Akku 750mAh betrieben. Am Tag wird der Sender durch die Solarzelle versorgt und der Akku aufgeladen. In der Nacht wird der Sender vom Akku gespeist.	
Testvoraussetzung	Die Energieversorgung muss am Sender verdrahtet und eingeschaltet sein.	
Durchführung	Über 24 Stunden wird der Sender mit der Energieversorgung betrieben.	
Durchläufe geplant	2	
Erwartetes Ergebnis	Der Sender wird 24 Stunden versorgt.	
Testmethode	Black-Box-Test	
Fehlerklasse	2	
Festgestellter Fehler	Durch das schlechte Wetter konnte die Solarzelle den Akku während des Tages nicht ganz aufladen, was zur Folge hatte, dass der Sender am frühen Morgen nicht mehr elektrisch versorgt werden konnte.	

Tabelle 18 Testfall-04

5.7.5 Testfazit

Die Tests sind gut verlaufen. Der Test mit der Energieversorgung, welcher nicht zufriedenstellend war, ist nur ein leichter Mangel. Die Fehlerbehebung kann mit einer leistungsstärkeren Solarzelle behoben werden. Ausserdem wird das Garden Monitoring nur von Anfang April bis ca. Ende September eingesetzt. Durch die längeren Tage wird somit auch der Akku optimaler geladen.

6. Reflektion

6.1. Beteiligte

Die Zusammenarbeit mit allen Beteiligten verlief einwandfrei. Die Aufgabenstellung wurde von der TEKO klar formuliert und mein Diplomlehrer Jörg Schenker konnte meine Fragen während der Diplomarbeit klären. Der Auftragsgeber, meine Familie hatte grosses Verständnis dafür, dass ich in den letzten Wochen viel Zeit in meine Diplomarbeit steckte und deshalb etwas weniger Zeit für meine Familie hatte.

6.2. Zielerreichung

Die Ziele, welche ich mir selbst gesetzt habe, konnten alle erreicht werden. Das Know-how in den Themen µC und LoRa sowie der Programmierung konnte ich mit dieser Arbeit deutlich vertiefen. Mein Auftraggeber ist sehr beeindruckt, was in so kurzer Zeit für ein Endprodukt entstanden ist.

Auch die Ziele des Auftraggebers konnten alle umgesetzt werden. Die gewünschten Funktionen wurden alle implementiert und der Sender konnte wettertauglich umgesetzt werden. Auch die maximalen Kosten von 400.- Franken wurden nicht überschritten.

Des Weiteren konnten noch zwei zusätzliche Ziele erreicht werden. Eine kleine einfache Verbindungsüberwachung wurde realisiert, weiter wurde der Sender mit einer eigenen Energieversorgung ausgestattet.

6.3. Termineinhaltung

Der Ablauf meiner Diplomarbeit war strukturiert und ich konnte mich an die meisten von mir vorgegebenen Termine halten. Der Start der Arbeit lief nach Plan, jedoch bei der Umsetzung mit der Programmierung verlor ich viel Zeit. In den drei Wochen Ferien während des zweiten Teils der Diplomarbeitszeit, konnte ich mich voll und ganz diesem Projekt widmen und somit holte ich die verlorene Zeit schnell wieder auf.

6.4. Lessons Learned

Ich konnte während der Diplomarbeit viele Dinge anwenden, welche ich gelernt habe. Das Programmieren der beiden µC mit allen Komponenten war für mich jedoch sehr anspruchsvoll, weshalb mir zu Beginn die Codes einige Probleme bereiteten. Ich verlor dabei viel Zeit, konnte aber mit viel Enthusiasmus und Interesse zwei funktionierende Codes schreiben. Diese Codes sind sicher nicht perfekt und könnten wahrscheinlich einfacher umgesetzt werden, jedoch erfüllen sie ihre Funktion.
Zudem besass ich noch nicht viel Erfahrung mit dem Schreiben und Umsetzen einer solchen umfangreichen Arbeit. Ich habe gelernt, dass es wichtig ist, den Terminplan so gut wie möglich einzuhalten um zeitgerecht abschliessen zu können.

6.5. Verdankungen

Jörg Schenker
An unseren zwei Besprechungsterminen konnte ich alle meine Fragen zur Diplomarbeit stellen. Ausserdem war Jörg Schenker telefonisch immer erreichbar.

Firma Securiton AG
Durch die von der Firma Securiton AG gewährten drei Wochen Ferien, konnte ich mich voll und ganz auf die Diplomarbeit fokussieren konnte.

Hugo Fankhauser
Mein Vater halft mir beim Aufbau des Senders, da er als gelernter Schreiner die besten Voraussetzungen dafür mitgebracht hat.

Cristina Fankhauser
Meine Frau Cristina las die Diplomarbeit durch und half mir beim finalisieren der Arbeit.

Therese Binggeli
Meine Vermieterin Therese Binggeli genehmigte mir den Aufbau des Garten Monitorings im Gemüsegarten.

6.6. Schlusswort

Die Diplomarbeit war für mich ein sehr spannendes Projekt mit vielen Hochs und Tiefs. Ich interessiere mich sehr für diese Thematik und konnte mich dadurch sehr gut motivieren alles so zu realisieren wie ich es mir vorgestellt habe. Da ich gelernter Elektromonteur bin, war es für mich sehr anspruchsvoll. Jedoch machte es mir grosse Freude dieses Projekt in die Tat umzusetzen. Ich bin sehr stolz auf das Endprodukt, welches ich ganz bestimmt im nächsten Frühjahr beim Start der Gartensaison einsetzen werde. Ich freue mich auf die Präsentation meines Projekts Garden Monitoring.

7. Eigenständigkeitserklärung

Ich bestätige, dass ich die vorliegende Diplomarbeit selbstständig verfasst und alle benutzten Quellen gekennzeichnet habe. Diese Arbeit wurde weder in gleicher noch in ähnlicher Form bereits einer Prüfungskommission vorgelegt.

Name / Vorname:

Fankhauser Timo

Ort / Datum / Unterschrift:

Lanzenhäusern, 01.10.2020

8. Quellen

8.1. Quellen Hardware Material

https://www.bastelgarage.ch/arduino-uno-rev-3-smd-board-atmega328?search=arduino%20uno
(Arduino UNO)

https://www.bastelgarage.ch/arduino/boards/arduino-mega-2560-kompatibles-board
(Arduino MEGA)

https://www.bastelgarage.ch/dragino-lora-shield-868mhz-v1-4-arduino?search=dragi
(Dragino-LoRa-Shield)

https://www.digitec.ch/de/s1/product/itead-studio-nextion-enhanced-nx4832k035-35-hmi-lcd-touch-display-uhr-elektronikmodul-5998760
HMI Nextion Display)

https://www.bastelgarage.ch/regensensor-modul-mit-analog-digital-ausgang?search=regensens
(Regensensor)

https://www.digitec.ch/de/s1/product/velleman-boden-feuchtigkeitssensorwasserpegelsensor-entwicklungsboard-kit-12256953
(Bodenfeuchtigkeitssensor)

https://www.digitec.ch/de/s1/product/adafruit-temperaturfeuchtigkeitssensor-am2315-sensor-elektronikmodul-6079339
(Temperatur-Luftfeuchtigkeitssensor)

https://www.bastelgarage.ch/solarzelle-12v-250ma-3w?search=solar
(Solarzelle)

https://www.bastelgarage.ch/mppt-solar-power-management-modul-fur-6v-24v-solar-panel?search=mppt
(Power Management Modul)

https://www.bastelgarage.ch/14500-3-7v-li-ion-akku-750mah-icr14500?search=3.7
(Akku)

8.2. Quellen Software/Bibliotheken

https://www.aeq-web.com/arduino-lora-wireless-shield-for-iot-direct-mode/
(Sketch für RSSI-Messung)

https://github.com/PaulStoffregen/SPI
(Bibliothek SPI.h)

https://github.com/adafruit/Adafruit_AM2315
(Bibliothek Adafruit_AM2315.h)

https://github.com/hallard/RadioHead
(Bibliothek Radio Head)

https://github.com/rocketscream/Low-Power
(Bibliothek LowPower.h)

8.3. Quellen Information

https://www.elektronik-kompendium.de/sites/kom/2203171.htm
(LoRa)

https://www.gruenderszene.de/lexikon/begriffe/microcontroller#:~:text=Begriff.&text=Bei%20einem%20Mikrocontroller%20handelt%20es,Ein%2DChip%2DSystem%20gibt.
(µC)

https://www.youtube.com/watch?v=26CJEGbRa9g
(Temperatur-Bodenfeuchtigkeitssensor)

https://www.instructables.com/Arduino-Rain-Sensor-Sketch/
(Regensensor)

http://fraunerd.de/feuchtigkeit-messen-mit-arduino/
(Bodenfeuchtigkeitssensor)

https://www.boecker-systemelektronik.de/Seite-/-Kategorie-1/NextionTutorials/Der-Nextion-Editor
(Nextion-Editor)

https://cdn.sparkfun.com/assets/learn_tutorials/8/0/4/RFM95_96_97_98W.pdf
(Rf95)

https://www.arduino.cc/en/Reference/ArduinoLowPower
(Low Power)

- Buch Arduino Elektronik, Programmierung, Basteln ISBN: 978-3-8362-3648-5
- Buch Projektmanagement für Führungsfachleute ISBN 978-3-7155-7672-5

9.Programmcode

9.1. Sender

```
#include <SPI.h>
#include <Adafruit_AM2315.h>
#include <RH_RF95.h>;
#include "LowPower.h"


RH_RF95 rf95;
Adafruit_AM2315 am2315;

void setup()
{
pinMode(A0, INPUT);
pinMode(A1, INPUT);

rf95.init();
am2315.begin();
}
void loop()

{
delay(2000);
float Temperatur;
float Luftfeuchtigkeit;
byte Bodensensor = analogRead(A0);
byte Regensensoranalog = analogRead(A1);

Temperatur = am2315.readTemperature();
delay(2000);
Luftfeuchtigkeit = am2315.readHumidity();
```

```
uint8_t message[6];

message[0] = Temperatur;
message[1] = Luftfeuchtigkeit;
message[2] = Bodensensor;
message[3] = Regensensoranalog;

rf95.send(message, sizeof(message));
rf95.waitPacketSent();

LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
}
```

9.2. Empfänger

```
#include <SPI.h>
#include <RH_RF95.h>

RH_RF95 rf95;

int led_betrieb = 3;
int packet = 0;
int bodenfgt_min = 120;
bool bodenfgt_gut;
byte endsequenz[3] = {255, 255, 255};



void setup()
{
  pinMode(led_betrieb, OUTPUT);
  Serial.begin(9600);
  rf95.init();
}

void loop()
{
  if (rf95.available())
  {
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
   if (rf95.recv(buf, &len))
   {
    packet = 1;
    digitalWrite(led_betrieb, HIGH);
    byte Temperatur = buf[0];
    byte Luftfeuchtigkeit = buf[1];
    byte bodensensor = buf[2];
    byte regensensoranalog = buf[3];
```

```
    Serial.print("n0.val=");
    Serial.print(Temperatur);
    Serial.write(endsequenz, 3);

    Serial.print("n1.val=");
    Serial.print(Luftfeuchtigkeit);
    Serial.write(endsequenz, 3);

    Serial.print("n3.val=");
    Serial.print(regensensoranalog);
    Serial.write(endsequenz, 3);

    Serial.print("n2.val=");
    Serial.print(bodensensor);
    Serial.write(endsequenz, 3);

      if (bodensensor >= bodenfgt_min)
        {
        Serial.print("t7.txt=\"\"");
        Serial.write(endsequenz, 3);
        bodenfgt_gut = true;
        }
       else
        {
         Serial.print("t7.txt=\"Alarm Bodenfeuchtigkeit\"");
         Serial.write(endsequenz, 3);
         bodenfgt_gut = false;
        }
        uint8_t data[] = "Antwortpacket";
        rf95.send(data, sizeof(data));
        rf95.waitPacketSent();
        digitalWrite(led_betrieb, LOW);
    }
 }
}
```