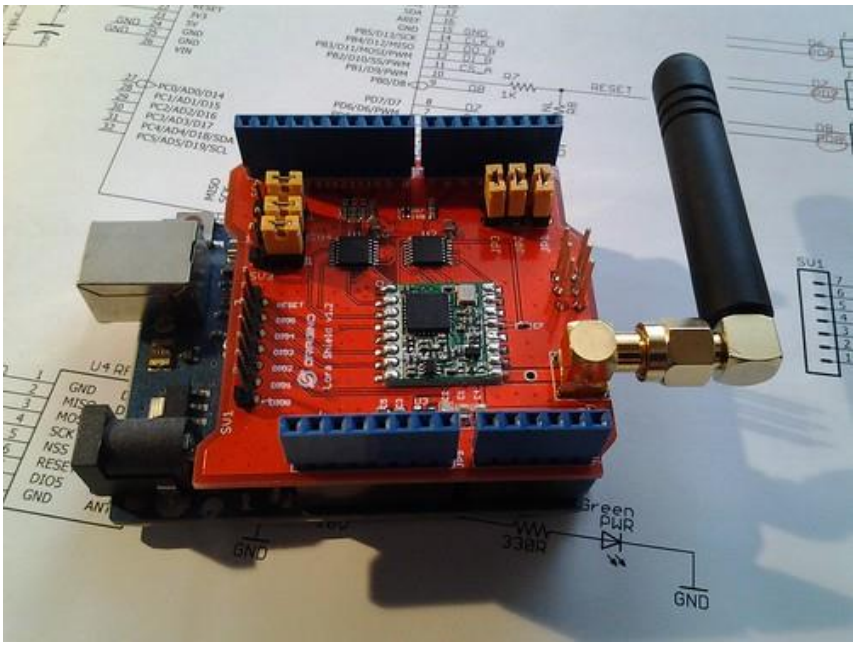
Projektbericht im Fach Technologie

Kabelloser Sensordatentransfer via LoRaWan

Berufliches Gymnasium
Schwerpunkt Elektrotechnik
Darmstadt 2019/2020

Kaan Biloglu



Inhaltsverzeichnis

Einführung ins Projekt

In diesem Projektbericht soll die Herstellung von 2 Endprodukten beschrieben werden, die miteinander Kommunizieren und sich Sensordaten schicken können. Die Datenübertragung soll über das Funksystem LoRa laufen.

Das Funksystem LoRa von Semtech entwickelt, ermöglich eine Datenübertragung auf enorme Reichweiten und verbraucht sehr wenig Energie. Eine Handelsübliche AA Batterie soll laut Hersteller bis zu 10 Jahre für die Funktechnik ausreichen.

Ziel dieses Laborversuchs war es also mithilfe der LoRa Funktechnik, einen Sender und einen Empfänger zu erstellen, wobei der Sender Daten mit gewöhnlichen Sensoren ermittelt und diese am Empfänger abzulesen sind.

Material

Zur Herstellung dieser Sender und Empfänger entschied ich mich für zwei Arduinos und erweiterte diese mit 2 Draguino Lora Shields und den mitgelieferten Antennen. Bei den zwei Arduinos handelte es sich um Arduino Unos und deren mitgelieferten Kabel und Breadboards. Des Weiteren hatte ich mehrere Sensoren zur Verfügung um die Temperatur und die Luftfeuchtigkeit zu messen. Ich entschied mich für den DHT22, da er einen großen Temperaturbereich abdeckt und die Temperatur dabei sehr genau ablesen kann.

Konzept

Die Idee war folgende. Auf die Arduinos werden die Draguino LoRa Shields gesteckt somit habe ich schon 2 Komponenten die mit der richtigen Programmierung schon miteinander kommunizieren können. Dann wird an das erste Board ein LCD Display angeschossen, auf dem später die Werte abzulesen sein werden. An das zweite Board kommen die Sensoren, die die Messwerte ermitteln.

Praktischer Teil

Die Aufgabe ließ sich in zwei Teilaufgaben gliedern. Einmal das Aufbauen des Projekts und die Planung und zum anderen das Programmieren des Projekts. Als Elektrotechniker bekommt man auf der HEMS einen Einblick in die Grundlagen des Programmierens in der E-Phase.

Zunächst war der Plan die zwei Stationen miteinander zu verbinden, sodass sie miteinander kommunizieren können. Die beiden Shields waren also auf den beiden Arduinos und jedes Shield hatte eine Antenne

Programmierung der Shields

Zur Programmierung benutzte ich einen vorgefertigten Code von AEQ-Web (mehr dazu in den Quellen) und änderte einige Details zur vereinfachung.

Die Codes sind natürlich so ausgelegt, dass es einen Sender und einen Empfänger gibt. Zur Vereinfachung habe ich den Sender „Messstation“ und den Empfänger „Basis“ genannt.

Code der Messstation

```
#include <SPI.h>
#include <RH_RF95.h>
RH_RF95 rf95; //Hier werden alle meine Bibliotheken eingebunden
//Messstation soll die Temperatur und Luftfeuchtigkeit messen und die Werte an die Basis senden.
void setup()
{
 Serial.begin(9600); //Damit das Programm weiß wann es starten soll
 delay(4000);
 if (!rf95.init()){
  Serial.println("init failed");
 }
}
void loop()
{
 uint8_t data[] =("Daten der Messung"); //Hier werden die Daten gesendet alles in Anführungsstrichen wird gesendet
 rf95.send(data, sizeof(data));
 rf95.waitPacketSent();
```

```
 uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
 uint8_t len = sizeof(buf);
 if (rf95.waitAvailableTimeout(3000)) //auf Bestätigung warten
 {
  if (rf95.recv(buf, &len))  //Wenn die Basis bestätigt hat
  {
    Serial.println("Warte auf Bestätigung... ");   //Wartet auf eine Bestätigung von der Basis
    Serial.print("Bestätigung der Basis: ");
    Serial.println((char*)buf);
    Serial.print("Signalstärke: ");
    Serial.print(rf95.lastRssi(), DEC);
    Serial.println(" dBm");
    Serial.println("_____________________________________________________________");
  }
  else
  {
    Serial.println("Keine Daten erhalten! ERROR");
  }
 }
 else
 {
  Serial.println("Keine Antwort von der Basis!");
 }
 delay(400);
}
```

Dieser Code sorgt also dafür, dass die Messstation eine beliebige Nachricht an die Basis sendet. Hier sende ich ein „Daten der Messung“ (grün markiert im Code). Die Messstation schickt diese Nachricht dann an die Basis und wartet dann auf eine Bestätigung, ob die Basis diese Nachricht erhalten hat (im Code gelb) und wenn die Nachricht angekommen ist wir auch noch abgefragt mit welcher Signalstärke diese gesendet wurde (im Code türkis). Wenn die Basis keine Daten von der Messstation erhalten hat wird (im Code lila) das Ausgegeben und im Falle, dass die Basis nicht antworten, weil sie nicht angeschlossen ist oder Defekt ist wir auch die (rot im Code) ausgegeben.

Code der Basis

```
#include <SPI.h>
#include <RH_RF95.h>
RH_RF95 rf95;
//Die Basis soll Messwerte der Messstation erhalten und diese dem Nutzer dann ausgeben.
void setup()
{
 Serial.begin(9600);
 dht.begin();
 delay(4000);
 if (!rf95.init()){
  Serial.println("init failed");
 }
}
void loop()
{
 if (rf95.available())
```

```
{
  uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
  uint8_t len = sizeof(buf);
  if (rf95.recv(buf, &len))
  {
    // Erhaltene Daten ausgeben
    Serial.println("Daten von der Messstation: "); // Erhaltene Daten ausgeben
    Serial.println((char*)buf);
    Serial.print("Signalstärke: ");
    Serial.print(rf95.lastRssi(), DEC);
    Serial.println(" dBm");

    // Bestätigung senden
    uint8_t data[] = "Daten Erfolgreich erhalten!";
    rf95.send(data, sizeof(data));
    rf95.waitPacketSent();
    Serial.println("Sende Bestätigung");
    Serial.println("_____");
  }
  else
  {
    Serial.println("Keine Daten erhalten! ERROR");
  }
 }
}
```

Dieser Code ist das Gegenstück zum vorherigen Code. Zunächst werden auch die die Bibliotheken eingebunden. Er erfasst dann die Nachricht von der Messstation und gibt diese auf dem Seriellen Monitor aus und gibt zusätzlich an mit welcher Signalstärke die Nachricht gesendet wurde (gelb im Code). Dann sendet die Basis eine Bestätigung an die Messstation (türkis im Code). Sollte er keine Daten von der Messstation erhalten gibt er dem Benutzer eine Fehlermeldung aus (rot im Code).

Ergebnis

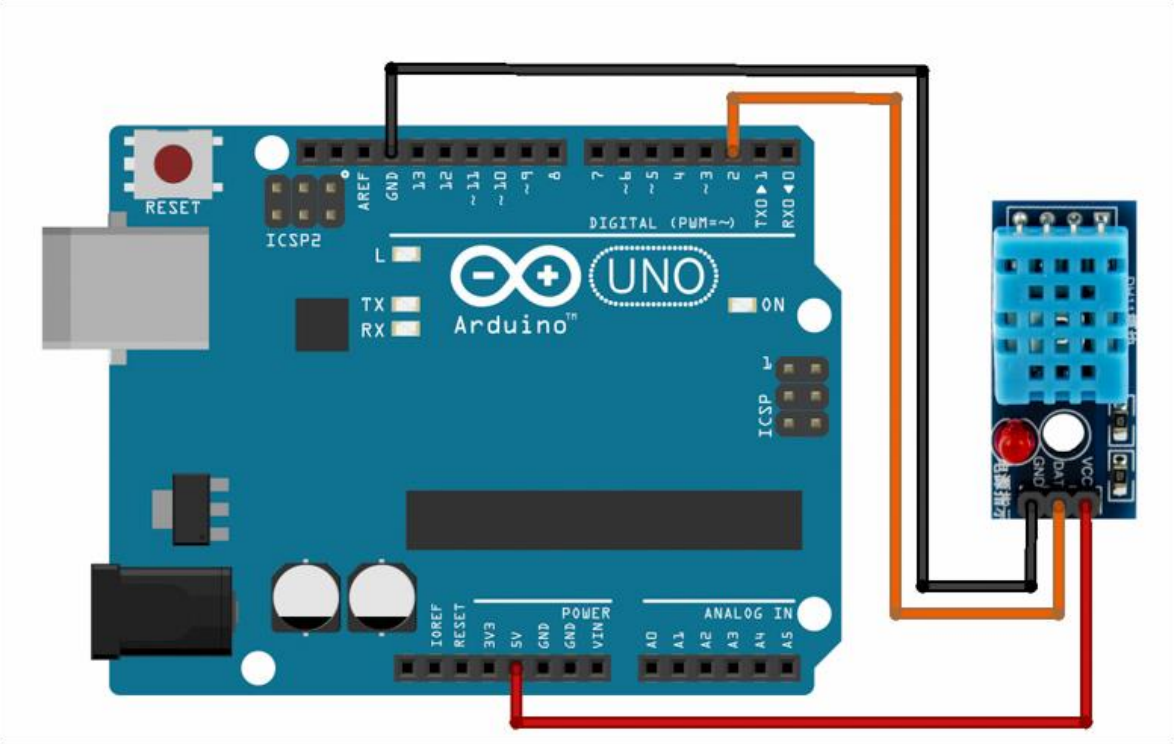
Nach der Programmierung konnte ich die zu sendende Nachricht (Im Code der Messstation grün) ändern und so verschiedene Texte Via der LoRa Funktechnik auf meinen zwei Arduinos senden und empfangen. Dies geschah im Labor Raum 314 mit einer Signalstärke von ca. -110 dBm.

Installation der Sensoren

Nun möchte ich die Shields mit Sensoren ausstatten und mein Projekt so erweitern, dass ich kleine Wetterstationen habe die ich an verschiedenen Orten in der Hems platzieren kann. Hierfür muss ich nur die Messstation modifizieren, die Basis bleibt erhalten.

Schaltungen

Die Messstation besteht aus dem Shield, einem Breadboard, dem DHT22 und Verbindungskabeln. Der DHT22 ist ähnlich wie der DHT11 in der Anwendung. Es ändert sich nur etwas beim Programmieren, weshalb wir den DHT22 genauso verschalten können wie den DHT11.
Den DHT22 schließt man auf ein Breadboard an und verbindet den Plus Pol des Sensors über das Breadboard mit dem 5V Ausgang des Arduinos. Den Minus Pol verbindet man mit Ground auf dem Arduino ist das der GND und die Datenübertragung kann man in einen beliebigen Eingang stecken.



Programmierung des Sensors

```
#include "DHT.h"
DHT dht(A2,DHT22);

void setup()
{
 Serial.begin(9600); //Damit das Programm weiß wann es starten soll
 dht.begin(); //Hier starten meine Sensoren

void loop()
{

 delay(2000);
 float t=dht.readTemperature(); //Hier wird festgelegt unter welcher Variable diese gespeichert wird
 float l=dht.readHumidity();//Hier wird festgelegt unter welcher Variable die Luftfeuchtigkeit gespeichert wird
 if (isnan(t) || isnan(l))
 {
  Serial.println("Es konnten keine Messwerte ermittelt werden! ERROR");
 }
 else
 {
  Serial.println("Gemessene Daten: "); //Hier werden die Daten schon ausgegeben, dies dient nur zur überprüfung an der
Basis und spielt später keine Rolle mehr
  Serial.print("Temperatur: ");
  Serial.print(t);
  Serial.println("°C");
```

```
  Serial.print("Luftfeuchtigkeit: ");
  Serial.print(l);
  Serial.println("%");
  Serial.println(" ");
  Serial.println("Weiterleiten...");
  Serial.println("Senden...");
  Serial.println(" ");
}
```

Das ist der Code nur für die Sensoren. Hier wird die Temperatur gemessen und ausgegeben und auch die Luftfeuchtigkeit. Dieser Abschnitt kommt also dann zum Restlichen Code der Messstation dazu.

Problem

In meinen Versuchen diesen Code zum restlichen hinzuzufügen geriet ich auf folgendes Problem.

Ich kann Texte verschicken die in Anführungsstrichen stehen jedoch kann ich keine Variablen verschicken. Meine Temperatur ist als Variable t deklariert. Das Bedeutet wenn ich t ausgebe gebe ich die derzeitige Temperatur aus. T ändert sich alle 2 Sekunden, da ich einen Delay von 2000ms habe. Ich muss also eine Variable versenden und nichtmehr nur einen Text der sich nicht ändern kann.

Problemlösung

Um dieses Problem zu lösen versuchte ich verschiedene Personen mit qualifizierteren Kenntnissen über das Programmieren aufzusuchen.

Lösung 1

Mein erster Anlaufpunkt war das Internet. Ich suchte im Internet vergeblich nach Menschen mit ähnlichen Problemen und schrieb im Anschluss selbst einen Post indem ich nach Ideen fragte, um mein Problem zu lösen. Die Arduino Community hatte aber auch keine Ahnung wie man dieses Problem lösen könnte, da sie selber zwar Ahnung vom Programmieren haben, jedoch nicht die Hardware, um es selbst durchzuführen.

Lösung 2

Zweite Anlaufstelle war also meine Familie. Mein Onkel beschäftigte sich mit der PHP Programmierung beruflich und konnte mir mit seinem Wissen auch nicht weiterhelfen da ich eine Spezielle Version von C++ benutzte, da diese Version die von den Arduinos Unterstütze war. Meine neue Erkenntnis war also, dass das Arduino C und C++ ist nicht gleiche sind.

Lösung 3

Meine dritte Möglichkeit war es altmodisch in die Bücherei zu gehen und dort nachzuschlagen. Ich bin aber zu TU und habe mir dort Informatik Studenten gesucht. Mir bot ein Florian Hirsch seine Hilfe an. Er ist Student im 3. Semester in Informatik und Programmiert auch in seiner Freizeit gerne.

Sein Vorschlag war es meine Variable in einen String umzuwandeln, da er vermutet hat, dass man eine float oder einen normalen Integer nicht über das Arduino C versenden kann. Also muss ich meine Variable über einen neu deklarierten String in ein neues Array packen, um es so zu versenden.

Nach mehreren Versuchen und verschiedenen Herangehensweisen, hat es in der TU und auch zuhause nicht funktioniert,

Lösung 4

Als letztes bin ich schließlich zu meinem Technologie Lehrer Herr Bersch gegangen, welcher mich an unsere Programmierlehrerin des BG brachte.

Frau Oppel ist 15 Jahre in der Software AG tätig gewesen und hat dort viel Erfahrung sammeln können. Jedoch muss man dazu sagen, dass man dort keine Arduinos programmiert und sie auch aus diesem Grund nicht das Arduino C kennt. Sie gab mir ebenfalls den Tipp das Ganze in einen String umzuwandeln, was ich aber schon versucht hatte. Auch beim wiederholten versuch mit Frau Oppel hat es nicht funktioniert.

Lösung 5

Beim Experimentieren mit Frau Opel war ich in ihrer Daten Verarbeiter Klasse, welche ich auch um Hilfe fragte. Ein Vorschlag der vermutlich im Normal C++ funktioniert hätte war es, eine Switch zu erstellen. Die Switch wählt je nach Temperatur aus welchen Case wir wählen und im Case wird meine Data also das was in den „Anführungsstrichen“ steht ausgewählt.

Der Vorschlag fiel jedoch auch weg, da man in einer Switch keine neuen Variablen Deklarieren oder schon Vorhandene Variablen Redeklarieren kann.

Offene Probleme

Das große Problem ist es also die Variable so zu deklarieren, dass man sie als Data versenden kann. Für einen erneuten Versuch empfehle ich das Arduino Board wegzulassen und eventuell auf einen Raspberry Pie zu wechseln. Ich vermute, dass dieser andere Programmiermöglichkeiten bietet, die wahrscheinlich die Probleme des Arduinos nichtig machen. Weiterführend kann man das Buch „Beginning LoRa Radio Networks with Arduino“ als Grundlage zur Bearbeitung des Projekts benutzen. In diesem wird genau erklärt wie man vorgehen muss und es werden auch Vergleiche zum Raspberry Pie aufgezeigt.

Quellen

(Alle Quellen wurden zuletzt am 11.01.2020 aufgerufen und waren frei zugänglich)
https://www.semtech.com/lora/what-is-lora

https://www.uni-koblenz-landau.de/de/landau/fb7/umweltwissenschaften/uchemie/wiss-schr/lab-bericht

https://www.dragino.com/products/module/item/102-lora-shield.html

https://www.lora-wan.de/#eigenes-lorawan-netzwerk

https://www.golem.de/specials/lora/

http://www.airspayce.com/mikem/arduino/RadioHead/

http://www.anarduino.com/miniwireless/

https://github.com/adafruit/DHT-sensor-library

https://www.bastelgarage.ch/index.php?route=extension/d_blog_module/post&post_id=1

https://www.aeq-web.com/arduino-lora-wireless-shield-for-iot-direct-mode/

https://www.youtube.com/watch?v=9cs25G3G7CY

https://www.youtube.com/watch?v=xVJ-L0_A5xE&t=35s

https://www.youtube.com/watch?v=IStuUv9eAmE

http://arduino-praxis.ch/2016/05/16/testbericht-dragino-lora-shield/

Literatur: „Beginning LoRa Radio Networks with Arduino“ – Apress ISBN 978-1-4842-4356-

Anhang

Vollständiger Code der Messstation

```
#include <SPI.h>
#include <RH_RF95.h>
#include "DHT.h"
DHT dht(A2,DHT22); //Hier werden alle meine Bibliotheken eingebunden
RH_RF95 rf95;
//Messstation soll die Temperatur und Luftfeuchtigkeit messen und die Werte an die Basis senden.

void setup()
{
 Serial.begin(9600); //Damit das Programm weiß wann es starten soll
 dht.begin(); //Hier starten meine Sensoren
 delay(4000);
 if (!rf95.init()){
  Serial.println("init failed");
 }
}
void loop()
{
 delay(2000);
 float t=dht.readTemperature(); //Hier wird festgelegt unter welcher Variable diese gespeichert wird
 float l=dht.readHumidity();//Hier wird festgelegt unter welcher Variable die Luftfeuchtigkeit gespeichert wird
 if (isnan(t) || isnan(l))
 {
  Serial.println("Es konnten keine Messwerte ermittelt werden! ERROR");
 }
 else
 {
  Serial.println("Gemessene Daten: "); //Hier werden die Daten schon ausgegeben, dies dient nur zur überprüfung an der
Basis und spielt später keine Rolle mehr
  Serial.print("Temperatur: ");
  Serial.print(t);
  Serial.println("°C");
  Serial.print("Luftfeuchtigkeit: ");
```

```
   Serial.print(l);
   Serial.println("%");
   Serial.println(" ");
   Serial.println("Weiterleiten...");
   Serial.println("Senden...");
   Serial.println(" ");
  }
  Serial.println("An die Basis gesendet");
  Serial.println(" ");
  uint8_t data[] =("Daten der Messung");/Hier werden die Daten gesendet
  rf95.send(data, sizeof(data));
  rf95.waitPacketSent();
  uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
  uint8_t len = sizeof(buf);

  if (rf95.waitAvailableTimeout(3000))//auf Bestätigung warten
  {
   if (rf95.recv(buf, &len)) //Wenn die Basis bestätigt hat
   {
    Serial.println("Warte auf Bestätigung... "); //Hier bekommt die Messstation eine Rückmeldung ob sie etwas and die Basis
gesendet hat. Auch die dient nur zur Überprüfung
    Serial.print("Bestätigung der Basis: ");
    Serial.println((char*)buf);
    Serial.print("Signalstärke: ");
    Serial.print(rf95.lastRssi(), DEC);
    Serial.println(" dBm");
    Serial.println("_____________________________________________________________");
   }
   else
   {
    Serial.println("Keine Daten erhalten! ERROR");
   }
  }
  else
  {
   Serial.println("Keine Antwort von der Basis!");
  }
  delay(400);
```