



TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Hygro#3

Carina Primas, Dominik Schmidt, Patrick Stillrich, Alexander Resnik

17. September 2020

Inhaltsverzeichnis

1	Einleitung	3
2	Technische Rahmenbedingungen	3
3	Hardware	3
3.1	Satelit	3
3.2	Basis	3
4	Software	4
4.1	Into	4
4.2	char bare protocol (cbp)	4
4.3	Satelit	5
4.3.1	Versorgungsspannung	5
4.3.2	low power	5
4.3.3	eeh210	6
4.3.4	cbp Realisierung	6
4.4	Basis	6
5	Ergebnis	6
6	Ausblick	6
7	Anhang	6
7.1	Akkuspannung	6
7.2	lowpower sleep	6
7.3	cbp Satelit	7
7.4	basis	7

1 Einleitung

Mit diesem Produkt soll die Luftfeuchtigkeit innerhalb und außerhalb eines Gewächshauses kontinuierlich gemessen werden, um dadurch zu gewährleisten, dass die Pflanzen optimale Feuchtigkeitsbedingungen haben. Es soll beispielsweise eine Pumpe bei zu niedriger Luftfeuchtigkeit bedienen können und eine Belüftung bei zu hoher Luftfeuchtigkeit aktivieren. Der Funktionsbereich liegt zwischen -10°C und 45°C .

2 Technische Rahmenbedingungen

Das Produkt wird aus 3 Einheiten bestehen. Eine davon wird eine Basisstation sein, die anderen 2 werden Satelliten sein.

Jede Einheit wird über einen Sender und einen Empfänger verfügen. Mit diesen wird eine 2- Richtungskommunikation realisiert. An jedem Satelliten wird ein Feuchtigkeitssensor verbaut. Diese Feuchtigkeitssensoren werden mit einem Mikrocontroller verbunden. Ziel ist es zuerst eine Grundfunktion herzustellen, im Nachhinein soll es eine Möglichkeit geben weitere Sensoren hinzuzufügen, Beispielsweise einen Lichtsensor, einen Luftdrucksensor, Bodenfeuchte oder einen CO_2 - Sensor. Die Satelliten werden mit Hilfe eines Akkus versorgt, dabei wird die Spannung überwacht. Außerdem werden wir Schutzfunktionen für Überlast und Kurzschlüsse hinzugefügt. Die Basis wird über Relais verfügen. Hiermit kann dann eine Abluft / Zuluft geschaltet werden. Mittels 2 Tastern kann man die minimale und maximale Luftfeuchtigkeit anpassen. Dies passiert in einer vorgegebenen Schrittweite. Des Weiteren soll es möglich sein die aktuellen Daten an einem Display auszulesen.

3 Hardware

3.1 Satellit

blindtext

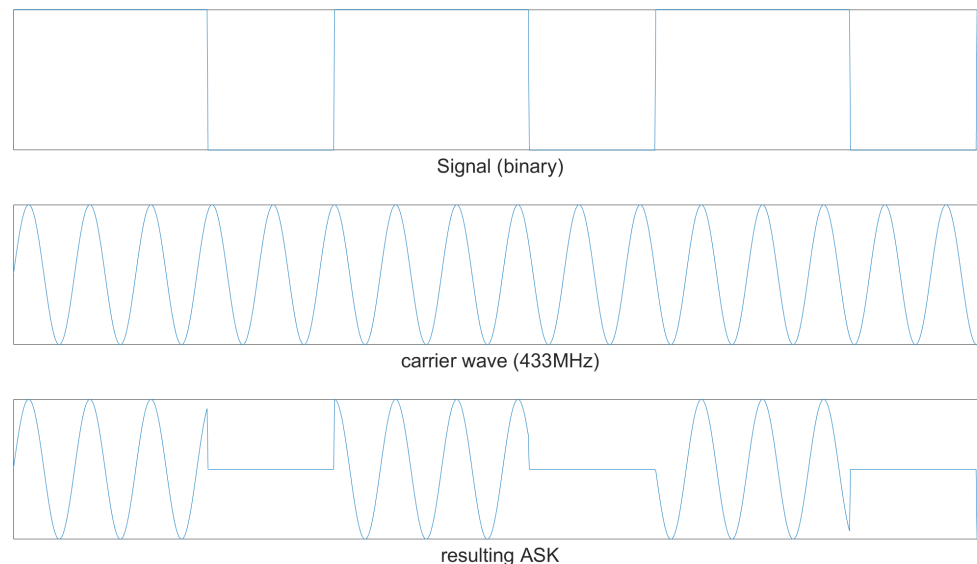
3.2 Basis

blindtext

4 Software

4.1 Intro

Bei der Funkverbindung wurde auf das Radiohead-Paket[1] zurückgegriffen. Dieses Paket unterstützt viele gängige Sender/Empfänger-Kombinationen. Des weiteren ist anzumerken das hier die "einfache" ASK¹ Modulationsart verwendet wird. Dies wird durch das sogenannte On-Off Keying realisiert.



Nachrichten des Radiohead-Pakets bestehen in unserer Anwendung aus unterschiedlichen Teilen. Daraus ergibt sich eine stabile Funkverbindung.

36 Bits	12 Bits	8 Bits	16 Bits	n Bits	16 Bits
Training Preamble for timing	Start Symbol 0xb38	Nachrichtlänge	Frame Check Sequence	Nachrichten payload	FCS 0x0F

Alle Daten welche nach dem Start Symbol versendet werden codiert übertragen. Somit ist ein Byte 2x6 bit lang.

4.2 char bare protocol (cbp)

Im zuge dieser Arbeit wurde ein eigenes Protokoll entwickelt, welches auf eine Paket basierte Übertragung, mit einem Master und n Slaves, aufbaut. In diesem Fall wird das

¹Amplitude Shift Keying/Amplitudenumtastung

Radiohead-ASK. Das Protokoll wurde cbp² genannt. Um in diesem Protokoll Daten zu Übertragen werden die Daten zu einem Char-array / einem String zusammengefügt. Verschiedene Daten werden durch einen Buchstaben gekennzeichnet. Anschließend werden die zu übertragenden Daten angehängt. Das format der Daten ist float oder int. Datensätze werden durch ein Komma getrennt. Ein Beispiel für einen solchen Datensatz könnte sein:

v3.8,t26.5,h33.2

Wie die Verschiedenen Zeichen interpretiert werden wird Ebenfalls festgelegt:

Char	Einheit	Funktion
v	V	Batteriespannung
t	°C	Sensor Temperatur
h	%	Relative Luftfeuchtigkeit
a	—	Anzahl der verfügbaren Aktoren

4.3 Satellit

4.3.1 Versorgungsspannung

Um die Versorgungsspannung zu messen wird die interne 1.1V interne Spannungsreferenz des ATmega verwendet. Der verwendete ADC verwendet als maximal Spannung die Versorgungsspannung. Diese Entspricht also dem Maximum von 1024 (10 Bit). Durch Messung der Internen Referenzspannung können dann Rückschlüsse auf die Versorgungsspannung getroffen werden. z.B. ergibt die Messung der Referenz einen Wert von 350, so kann durch die Formel $\frac{1.1V}{ADCResult} * 1024$ berechnet werden. Siehe 7.1 Zeile 8.

4.3.2 low power

Die Anforderungen an die Laufzeit wurden durch ein “deep-sleep” realisiert. Hierfür wurde auf das git von Rocketscream zurückgegriffen[2]. Hier kann eine maximale sleep Dauer von 8s eingestellt werden. Für größere Zeiten muss ein Externer Interrupt festgelegt werden. Um eine weitere externe Schaltung zu vermeiden wird hier die Clock abgeschaltet nicht abgeschaltet. Diese wird verwendet um einen Timer zu steuern, welcher nach 8s einen Interrupt erzeugt. Mit diesem wacht der ATmega auf. Dies wird wiederholt um größere Zeiten zu erreichen. Siehe 7.2

²Carmens-beef-protocol

4.3.3 eeh210

4.3.4 cbp Realisierung

Die cbp Realisierung basiert darauf, dass sämtliche Daten sich durch, z.B. `dtostrf()`, zu einem String formatieren lassen. Diese Teilstrings werden anschließend durch Kommas getrennt aneinandergereiht. Hierfür wird hauptsächlich die funktion `strcat()` verwendet. Siehe 7.3

4.4 Basis

blindtext

5 Ergebnis

6 Ausblick

7 Anhang

7.1 Akkuspannung

```
1 float fReadVcc() {
2     ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
3     delay(5); //delay for 5 milliseconds
4     ADCSRA |= _BV(ADSC); // Start ADC conversion
5     while (bit_is_set(ADCSRA, ADSC)); //wait until conversion is complete
6     int result = ADCL; //get first half of result
7     result |= ADCH << 8; //get rest of the result
8     float batVolt = (iREF / result) * 1024; //calculate battery voltage
9     return batVolt;
```

7.2 lowpower sleep

```
1     digitalWrite(ENABLE_RXTX, LOW);
2     for(;low_power_sleep<20;low_power_sleep++){
3         LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
```

```
4     }  
5     low_power_sleep = 0;
```

7.3 cbp Satelit

```
1     char data[24] = "";  
2     char temp[5] = "v";  
3     strcpy(data, temp);  
4     dtostrf(fReadVcc(), 4, 2, temp);  
5     strcat(data, temp);  
6  
7     strcat(data, ",h");  
8     dtostrf(rh, 3, 1, temp);  
9     strcat(data, temp);  
10  
11    strcat(data, ",t");  
12    dtostrf(t, 3, 1, temp);  
13    strcat(data, temp);
```

7.4 basis

```
1 dummy
```

Literatur

- [1] AirSpayce Pty Ltd, *RadioHead Packet Radio library for embedded microprocessors*. [Online]. Available: <http://www.airspayce.com/mikem/arduino/RadioHead/>
- [2] Rocket Scream Electronics, *Low-Power*. [Online]. Available: <https://github.com/rocketscream/Low-Power>