BusPuter Dokumentation

für HW Version 0.5

Brun von der Gönne

27. Januar 2017

Inhaltsverzeichnis

1 Vorwort

Der BusPuter ist ein OpenSource Projekt welches ständig weiterentwickelt wird. Daher kann es zu Abweichungen zwischen der Dokumentation und den Sourcecode geben.

Es wird jedigliche Verantwortung für Schäden die durch den Nachbau entstehen abgelehnt. Beim Nachbau und der Veränderung ist auf die entsprechenden Vorschriften zu achten.

2 Vorbereitung

2.1 Arduino IDE einrichten

Damit die BusPuter Sourcen richtig kompiliert werden können, müssen noch folgende Librarys und Komponenten in der Arduino IDE hinzugefügt werden.

Das Hinzufügen der Feather M0 Kompatibilität ist von Adafruit relativ gut erklärt. https://learn.adafruit.com/adafruit-feather-m0-adalogger/setup

Anschließend können dann folgende Bibliotheken unter *Sketch -> Bibliothek einbinden -> Bibliotheken verwalten* runter geladen werden

- Adafruit FONA Library
- Adafruit SleepyDog Library
- U8g2

Die SdFat Bibliothek ist aktuell leider noch nicht enthalten. Diese muss man sich noch aus den Internet runterladen und manuell installieren. Download unter https://github.com/greiman/SdFat/archive/master.zip. Entpacken und Anschließend das Verzeichnis *SdFat* in das Library von Arduino kopieren.
Danach muss die Arduino IDE neu gestartet werden.
Anschließend sollte BusPuter compiliert werden können.

2.2 Komponenten

Folgende Komponenten werden auch noch benötigt:

- Adafruit Feather M0 Adalogger
- LoNet808
- eine aktive SIM Karte ohne PIN
- Spannungswandler
- Kleinteile für die analogen Eingänge

3 Aufbau des BusPuter

Je nach Version der Hauptplatine sind zwei bis sechs analoge Eingänge vorgesehen. Die ersten beiden Eingänge sind fest vorgesehen. Alle weiteren Eingänge sind optional. Bei einen Mainboard mit sechs analogen Eingängen reicht es auch erstmals aus, wenn nur die ersten beiden Eingänge bestückt sind.

ACHTUNG!!! Auf den Mainboard mit der Version 0.5 ist der Enable Pin vom Arduino auf Masse gezogen. Dadurch startet der Arduino nicht. Beim Zusammenbauen diesen Pin einfach aus den Stecker lösen. So das er nicht verbunden werden kann. Abbildung 1.

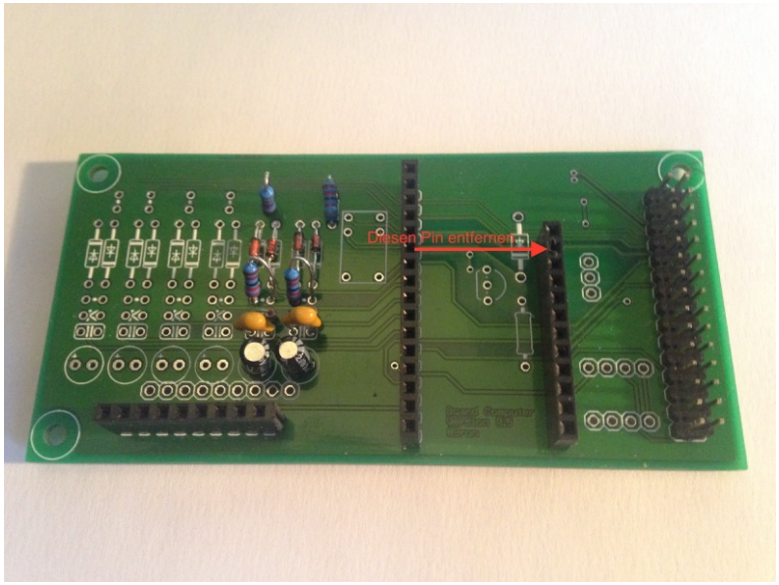

Abbildung 1: Display Kabel

3.1 Mainboard

Das Ding muss halt auch zusammen gelötet werden.

Die Stiftleisten und Buchsenleisten einlöten.
Bei den Analogen Ports (Spannungsteiler) muss auf die Polarität der Dioden und des Elko geachtet werden.

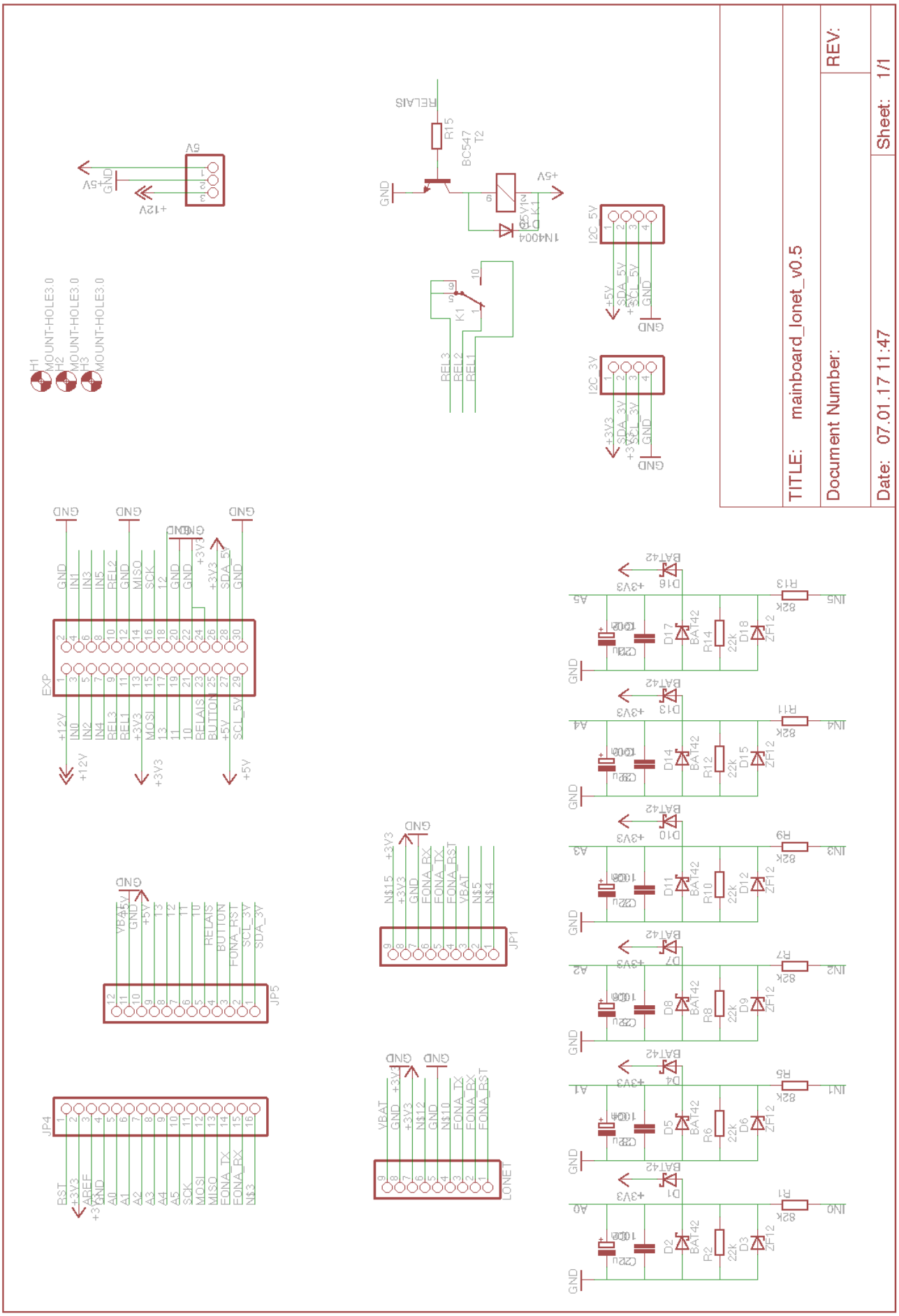

Abbildung 2: Schaltplan des BusPuter Mainboard

3.2 Display

Beim Display muss auf die Werte der Widerstände geachtet werden. Der Widerstand R3 muss anhand der verwendeten Hintergrundbeleuchtung verwendet werden.

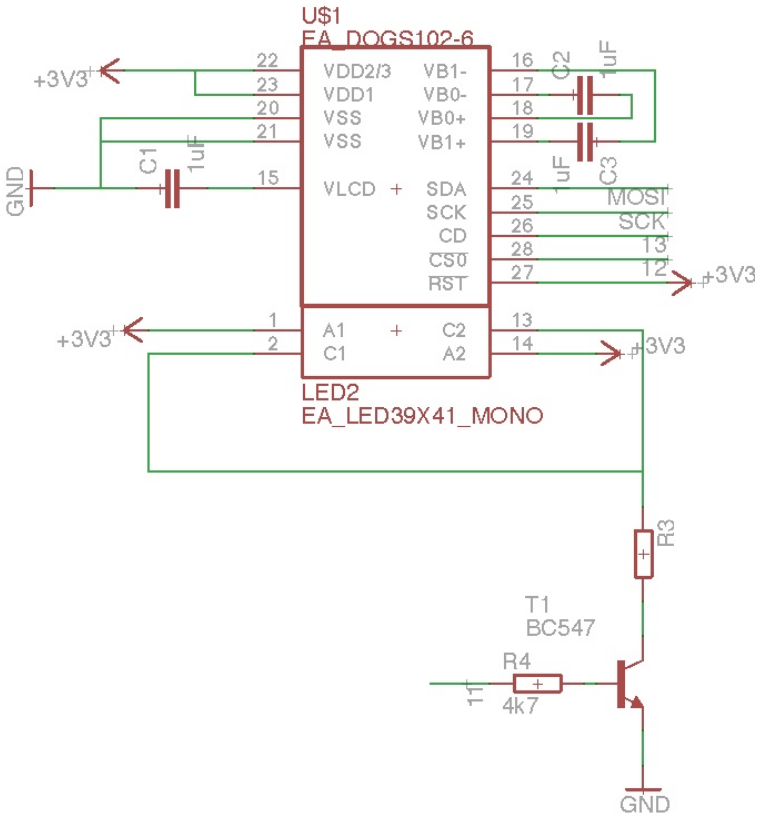

Abbildung 3: Schaltung des LCD

amber EA LED39x41-A	Forward voltage	Current max.	Limiting resistor @ 3,3 V	Limiting resistor @ 5 V
Connected in parallel	2,2 V	80 mA	15 ohm	36 ohm
Connected in series	4,4 V	40 mA	CAT4238	15 ohm

white EA LED39x41-W	Forward voltage	Current max.	Limiting resistor @ 3,3 V	Limiting resistor @ 5 V
Connected in parallel	3,3 V	60 mA	0 ohm	28 ohm
Connected in series	6,6 V	30 mA	use CAT4238	

Abbildung 4: Auszug aus dem Datenblatt vom DOGS102

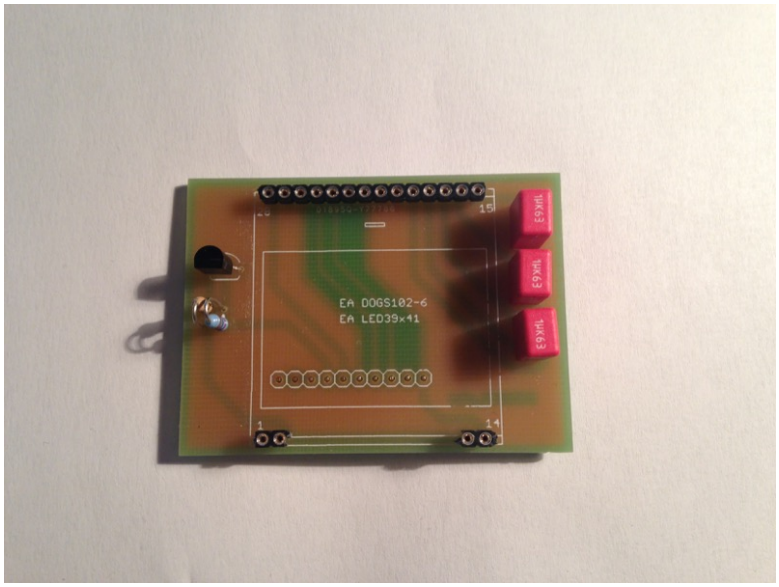

Abbildung 5: Oben

Der Stecker wird wie folgt eingelötet.

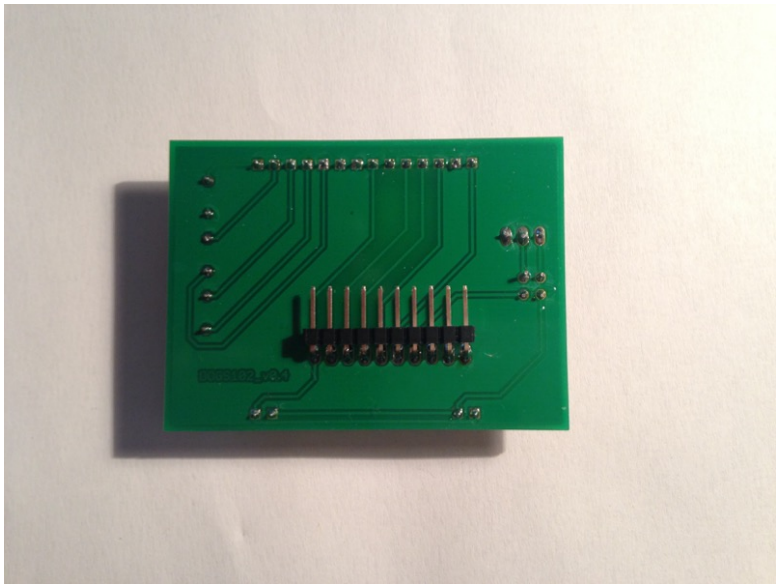

Abbildung 6: unten

3.3 LoNet808

Der Stecker vom LoNet werden so eingelötet, dass der SIM Kartenhalter nach oben zeigt.

3.4 Spannungsversorgung

Für die Spannungsversorgung kann ein beliebiges Modul verwendet werden, welches 12-20V in 5V umwandelt. Bei der Auswahl ist allerdings drauf zu achten, dass die Pinbelegung “Vin - GND - Vout“ ist.

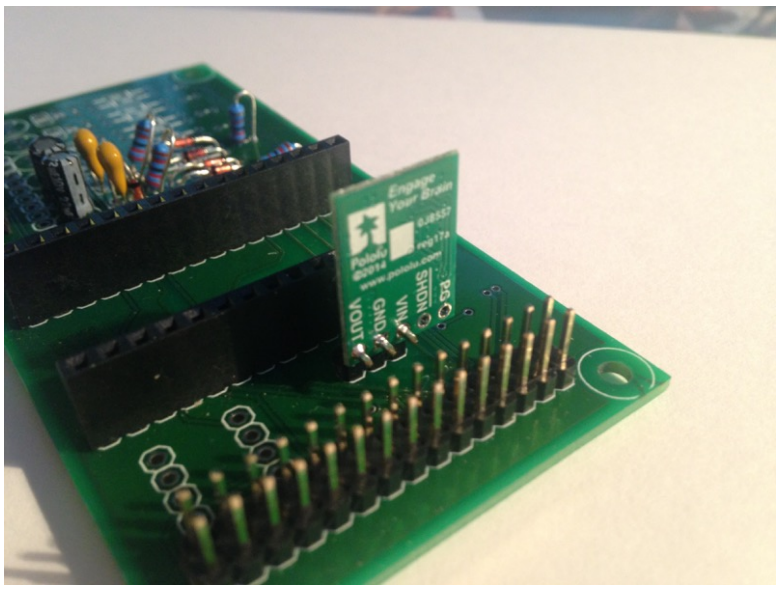

Abbildung 7: Spannungwandler von 12V auf 5V

Pin 1 vom großen Stecker ist 12V und Pin 2 ist Masse. Je nach Ausführung

4 Zusammenbau

4.1 Anschluss vom Display

Um das Display vernünftig anzuschließen empfiehlt es sich ein passendes Kabel zu fertigen. Auf der BusPuter Seite ein 2x4 poliger Stecker und auf der Display Seite ein 1x8 poligen Stecker.

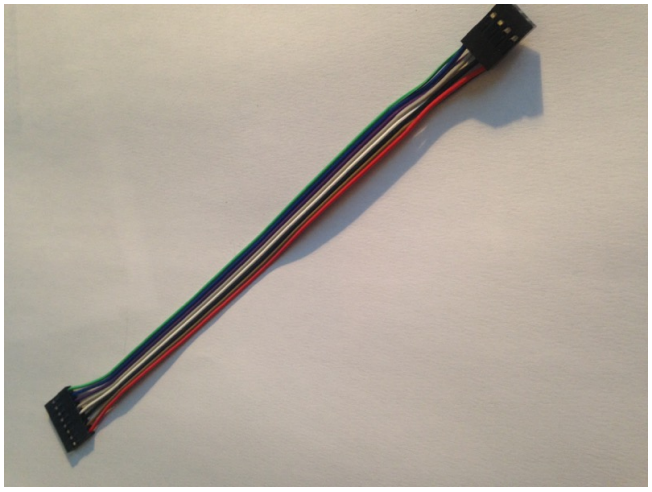

Abbildung 8: Display Kabel

Bitte auf die Polung von Display und BusPuter achten. Da ansonsten eins oder beides zerstört werden kann. Auf den BusPuter ist der Pin 13-20 für das LCD vorgesehen.

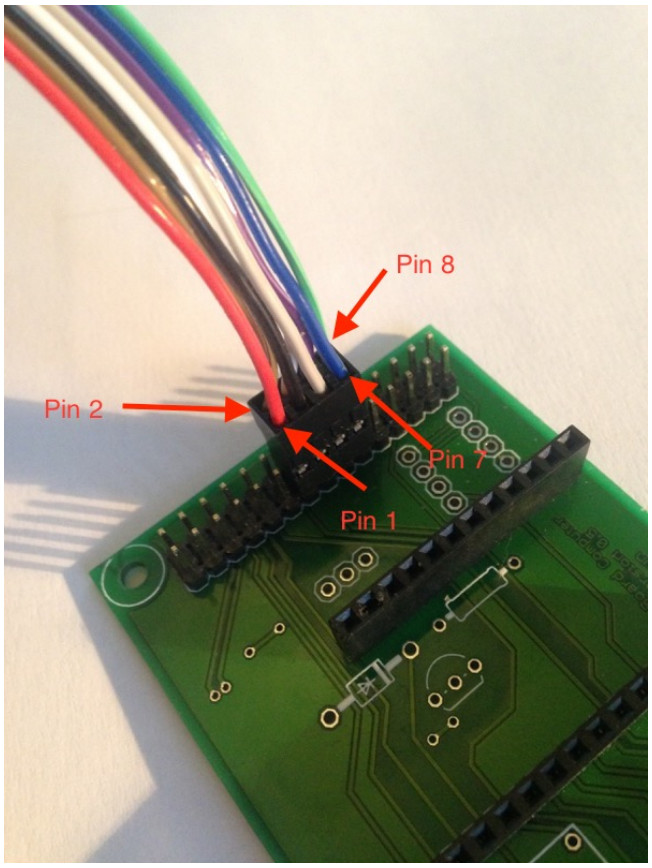

Abbildung 9: Stecker am BusPuter

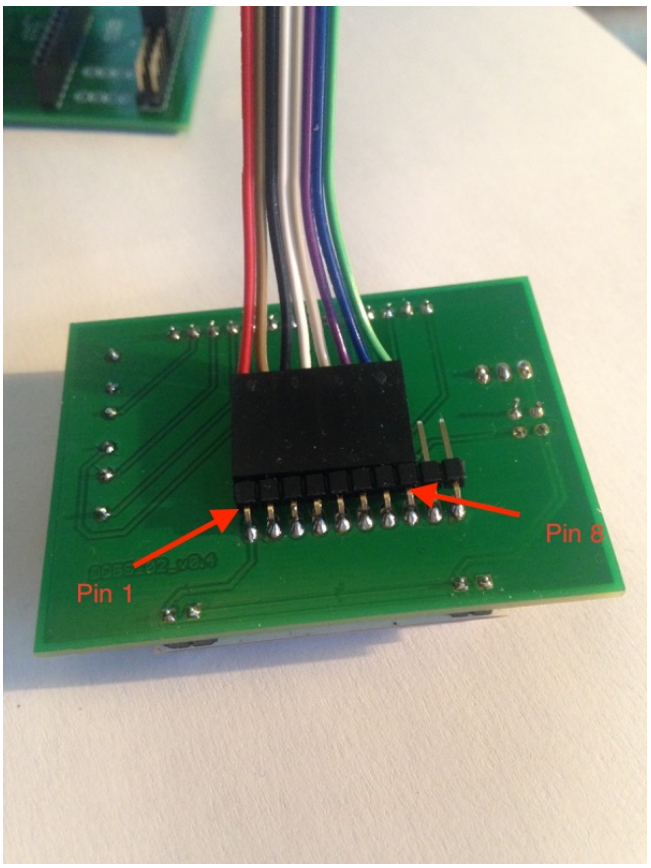

Abbildung 10: Stecker am Display

4.2 Spannungsversorgung

Die Spannungsversorgung kann über mehrere Wege erfolgen.
Die empfohlenen Methode ist über die 12V Bordspannung. Alternativ kann der BusPuter auch über den USB Anschluss des Feather Board erfolgen. Für den Notfall und als Backup besteht auch die Versorgung über einen LiPo Akku zur Verfügung.

4.3 Taster

Der Taster kann an Pin 23 (Arduino Pin 9) und 24 (GND) angeschlossen werden. In der Configuration (config.h) wird der Pin unter "#define BUTTON_PIN_1 9" definiert. Der Taster kann auch an einen anderen freien Port angeschlossenwerden, da der Pin 9 unter anderem auch für das Relais genutzt werden kann. Dies ist in der Basisfirmware allerdings noch nicht implementiert. (Stand Version 170115a)

4.4 LoNet808

Der Anschluss des LoNet808 wird empfohlen, da momentan in der Software noch keine alternative Uhr implementiert ist (Stand Version 170115a). Das LoNet Module ist für die Bestimmung der GPS Koordinaten, der Geschwindigkeit und der Uhrzeit verantwortlich. Nebenbei kann darüber eine SMS Kommunikation realisiert werden und das Tracking übers Internet ist auch damit möglich. Genauere Informationen kommen dann in späteren Abschnitten.

In der BusPuter Software ist das GSM/GPS Module per Default aktiviert. Es wird die Adafruit Fona Library verwendet. Um das GPS/GSM Modul zu deaktivieren, muss in der „config.h" der Parameter „#define FONA" auskommentiert werden.

4.5 OneWire Temperatursensor

In der BusPuter Firmware ist als OneWire Temperatursensor der Dalles DS18B20 implementiert. Dieser benötigt eine Spannungsversorgung von 5V und GND sowie einen Digitalen Pin. Hier könnte z.B. der Pin 25 (Arduino Pin 6) genutzt werden.

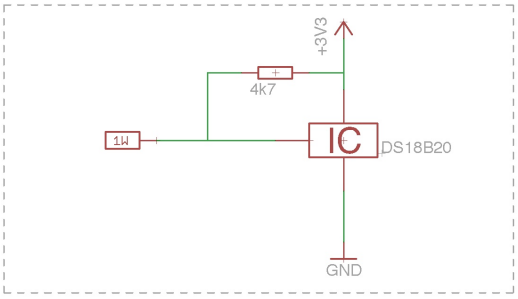

Abbildung 11: OneWire Temperatursensor DS18B20

4.6 sonstige Erweiterungen

5 Konfiguration

Für eine einwandfreie Funktion der Komponenten muss etwas am Sourcecode konfiguriert werden. Dies wird in der “config.h“ gemacht.

5.1 LoNet808

```
// Fona Libary
#define FONA

#define SMS_Keyword "PWD"
#define MyNumber "+49171234567"
#define GPRS_APN "internet.t-mobile"
#define GPRS_user "t-mobile"
#define GPRS_pw "tm"
#define TRACKING_URL "my.tracking.url/gps.php?visor=false&latitude=%s&longitude=%s&altitud
```

Mit „#define FONA“ wird die FONA/LoNet Funktionalität erst aktiviert. Sollte das Module FONA nicht aktiviert werden, dann ist auch keine GPS Bestimmung mehr möglich.

“MyNumber“ definiert die nicht die Telefonnummer der SIM Karte im GSM Module sondern die Nummer die bei Alarmierungen informiert werden soll.

Die Zugangsdaten für den Internetzugriff kann man beim Provider erfragen oder einfach mal kurz Google benutzen. Dieses Beispiel ist auch für Congstar verwendbar.

Die “TRACKING_URL“ zeigt auf die Seite wo die Trackinginformationen hinterlegt werden.

ACHTUNG!!! Eine Konfiguration einer PIN ist aktuell noch nicht implementiert (Stand 170117a).

5.2 Display

```
// Display Configuration
#define LCD
#define LCD_LED_MAX 255
#define LCD_LED_MIN 0
```

Aktivieren und deaktivieren des LCD. Des weiteren kann man die maximale und minimale Helligkeit des Displays konfigurieren. Das Dimmen des Display passiert über die normale KI Beleuchtung. Mit den Werten unter “LCD_LED_MAX“ und “LCD_LED_MIN“ wird der Helligkeitsbereich angegeben.

5.3 SD Logging

```
// SD Card
#define SDCARD
```

Aktivieren oder deaktivieren der SD Karte. Falls kein Mitschreiben erwünscht ist oder keine SD Karte vorhanden ist.

5.4 Debugging

```
// enable Debugging
#define INFO
//#define DEBUG
//#define TRACE
```

Es gibt verschiedene Debuglevels. Die Ausgabe erfolgt nur über die Serielle Konsole.

6 Erweiterungen

6.1 Analoge Ports

Der Busputer verfügt über 6 analoge Ports. Wobei die ersten beiden Ports schon vergeben sind. (Start/Bordspannung und KI Beleuchtung). An den anderen freien Ports können zur Spannungsmessung von weiteren Komponenten verwendet werden. Über einen Spannungsteiler können über die analogen Port Spannungen mit bis zu 15.6V gemessen werden.

6.1.1 Kühlwassertemperaturanzeige vom Kombiinstrument

Die Kühlwassertemperaturanzeige vom Kombiinstrumen kann zwischen Anzeige und Sensor abgegriffen werden.

ACHTUNG!!! Diese Messmethode ist nicht wirklich genau und kann daher nur eingeschränkt genutzt werden. Da mir aktuell weder ein Datenblatt vom Instrument noch vom Sensor vorliegen, können hier hohe Toleranzen auftauchen.

Implementierung Der Messpunkt ist zwischen Instrument und Sensor. Einfach eine Verbindung auf einen freien Port des Busputers legen.

Konfiguration Konfiguriert wird wie folgt:

```
// VW Tempsensor
#define VW_WATER_TEMP A2
```

Einfach nur den Port definieren der verwendet werden soll.

6.1.2 Tankanzeige

Noch nicht programmiert.

Implementierung

Konfiguration

6.2 I2C Bus

Der Busputer verfügt über einen I2C Bus über den weitere Komponenten angeschlossen werden. Es kann selber entschieden werden ob der I2C Bus auf 5V oder auf 3,3V basiert. Der Arduino hat einen 3,3V basierten I2C Bus. Über einen Pegelwandler kann der Bus auch auf 5V angehoben werden. Es ist allerdings drauf zu achten, dass alle weiteren Komponenten auch die entsprechenden Spannung vertragen.

6.3 SI7021 Temperatur- und Luftfeuchtigskeitssensor

7 Eigener Code

In der neusten Version des BusPuter Code gibt es eine einfache Möglichkeit eigenen Code zu implementieren.

7.1 Definition

Alle konfiguration dieses Moduls wird im Reiter “custom’"gemacht. Damit dieser Bereich überhaupt mit compiliert wird, muss die Option “#define CUSTOM“ gesetzt werden.
Die weiteren Bestandteile sind drei Funktionen die benötigt werden.

7.2 Funktion custom_init

Diese Funktion wird bei der Initialisierung aufgerufen.

7.3 Funktion custom_loop

Diese Funktion wird im normalen Loop aufgerufen. Und zwar immer wenn alle anderen Programmteile abgearbeitet sind.
Damit nun nicht sinnlos CPU Zeit verbraten wird empfiehlt es sich immer nur nach einer bestimmten Zeit den Code auszuführen. Zum Beispiel:

```
#define CUSTOM_TIMER 200 // 200ms
unsigned long custom_timer = 0;

void custom_loop() {

  // to save cpu power run this function only every (above defined) time
  if ( custom_timer < millis() ) {
    // put your code here

    custom_timer = millis() + CUSTOM_TIMER;
  }
}
```

In diesen Beispiel wird der Code nur alle 200ms ausgeführt. Die Abfrage einer bestimmten Temperatur oder eines anderen Wertes muss man nicht mehrere tausend mal pro Sekunde machen. Bei Temperaturen reicht eine Abfrage vielleicht auch alle 2s oder noch länger.

ACHTUNG!!! Der BusPuter hat einen Watchdog. Sollte der Programmcode nicht in einer entsprechenden Zeit abgearbeitet sein, kann der Watchdog den BusPuter resetten.

7.4 Funktion custom_menu

In dieser Funktion kann ein eigener Menüpunkt integriert werden.

Wichtig ist, dass am Ende folgende Zeilen kommen:

```
// this is needed to reackt on the button action
switch (button_1) {
  case 1: MainMenuPos++; break;
  case 2: MainMenuPos--; break;
}
button_1 = 0;
```

Dadurch wird es möglich im Menü einen Punkt weiter zu springen. Unter “case 2“ (langer Tastendruck) könnte noch verändert werden.