BusPuter Dokumentation

für HW Version 0.5

Brun von der Gönne

28. September 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.

Ein kommerzieller Vertrieb wird ausdrücklich untersagt.

1.1 Der Name

Auch wenn man durch den Namen denken mag, dass dieses Projekt nur für den VW Bus bestimmt ist, irrt sich. Ja, dieses Projekt findet im Bulli seine erste Verwendung aber der Name stammt von der Intention das er vielseitig bleiben soll und aus diesem Grund ein I2C Bus besitzt sowie optional eine 1-Wire Bus.

1.2 Funktionsumfang

Aktuell sind folgende Funktionen integriert.

- Geschwindigkeitsmessung via GPS
- Geschwindigkeitsmessung via GALA
- Auswertung der Drehzahl
- Auswertung der Wassertemperatur vom Kombiinstrumen
- Auswertung der Tankanzeige vom Kombiinstrument
- Anzeigen der aktuellen Position auf dem Handy mit der Hilfe von Blynk

Der Funktionsumfang ist allerdings noch viel größer. Es wird ja auch ständig dran weiterentwickelt.

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

- TinyGSM (aktuell nur aus meinen Repository)
- Blynk
- U8g2
- OneWire
- SdFat

Die passende TinyGSM Libary kann sich unter https://github.com/brvdg/TinyGSM/archive/master.zip runter geladen werden. Auch diese muss manuell installiert werden.
Danach muss die Arduino IDE neu gestartet werden.

Auch muss in der Arduino IDE das entsprechende Board noch gewählt werden.

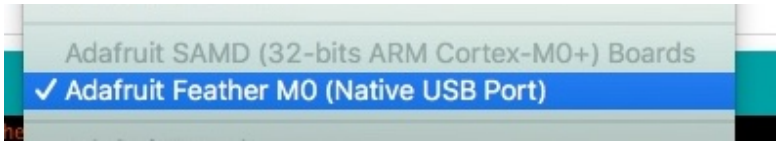

Abbildung 1: Arduino IDE

Anschließend sollte BusPuter compiliert werden können.

2.2 Komponenten

Folgende Komponenten werden auch noch benötigt:

- Adafruit Feather M0 Adalogger (Falls keine SD Karte benötigt wird, kann man auch die Variante ohne SD nehmen)
- LoNet808 / Adafruit FONA808 / oder anderes SIM808 basierenden Modul (z.B. von and-global. Diese gibt es bei eBay aus China)
- eine aktive SIM Karte ohne PIN (die Eingabe eines PIN ist nicht möglich)
- 5V Spannungswandler Wenn man ein Modul ohne LiPo Anschluss hat sollte die Spannungsversorgung nicht zu klein sein. Getestet habe ich es mit den Modulen von Polulo. Aber eigentlich sollte es egal sein was man nimmt. Hauptsache die Pinbelegung und die Spannung / Strom stimmt.
- Kleinteile wie Jumperwires und die dazu passenden Gehäuse. Diese gibt es überall und sind einfach zu bearbeiten.

2.3 Einkaufsliste

Hier meine Einkaufsliste. Sie ist von nur nur ein Vorschlag. Natürlich kann man jeglichen beliebigen Händler für die Komponenten nehmen.

- Elektronische Teile für das Mainboard: `https://www.reichelt.de/my/1375409`
- Teile für das Display mit einen EA DOGS Display mit oranger hintergrundbeleuchtung: `https://www.reichelt.de/my/1386877` Es gibt auch eine Weiße Hintergrundbeleuchtung.
- Gehäuse `https://www.reichelt.de/Kunststoff-Kleingehaeuse/SD-20-GR-HALB/3/index.html?ACTION=3&LA=10030&ARTICLE=149277&GROUPID=7715&artnr=SD+20+GR+HALB` und `https://www.reichelt.de/Kunststoff-K SD-10-GR-HALB/3/index.html?ACTION=3&LA=10030&ARTICLE=149274&GROUPID=7715&artnr=SD+10+GR+HALB` oder ein anderes.
- Den Feather habe ich von EXP-Tech `http://www.exp-tech.de/adafruit-feather-m0-adalogger` Mann kann aber auch einen anderen nehmen. Es muss nur ein Adafruit Feather in der M0 Variante sein. Also mit 32Bit Prozessor. Andere Boards passen nicht, da hier die Pinbelegung anders ist
- ein SIM808 basierendes Modul z.B.: `http://www.exp-tech.de/adafruit-fona-808-shield-mini-cellular-gsm-gp` Mein persöhnliche Favorit ist das LoNet808. Der ist leider schwer zu beschaffen. Ich selber habe auch Module von and-global. Diese gibt es bei eBay aus China. Da aber selber schauen.
- Falls bei den Modul keine Antenne dabei sind braucht man noch passende Antennen
- Jumper Wires. Was man da nimmt kommt drauf an wo man ihn hin bauen will. Dementsprechend lang müssen die Kabel sein. Ich habe z.B. diese hier: `http://www.exp-tech.de/wires-with-pre-crimped-terminals-20-` sie sind zwar nicht die billigsten aber dafür schön lang. Die gibt es auch noch länger `http://www.exp-tech.de/wires-with-pre-crimped-terminals-10-piece-rainbow-assortment-f-f-60-150-cm`
- 5V Spannungswandler Wenn man ein Modul ohne LiPo Anschluss hat sollte die Spannungsversorgung nicht zu klein sein. Getestet habe ich es mit den Modulen von Polulo. Aber eigentlich sollte es egal sein was man nimmt. Hauptsache die Pinbelegung und die Spannung / Strom stimmt. z.B: `http://www.exp-tech.de/pololu-5v-1a-step-down-voltage-regulator-d24v10f5`. Wenn man ein anderes Modul nimmt muss auch datauf geachtet werden, dass wir eine Eingangspannugn vün über 12V haben die dann auf 5V runter geregelt werden müssen.

- zur besseren Handhabung empfehlen sich auch Gehäuse für die Jumper Wires. z.B. 2x12 fürs Mainboard `http://www.exp-tech.de/0-1-2-54mm-crimp-connector-housing-2x12-pin-5-pack`, fürs Display `http://www.exp-tech.de/0-1-2-54mm-crimp-connector-housing-1x8-pin-10-pack`

3 Aufbau des BusPuter

Der BusPuter in der aktuellen Version hat 6 Eingänge. 2 Digitale die über einen Optokoppler getrennt sind, 2 Analoge für Spannungen bis 15,6V

- 2 digitale Eingänge die mit einen Optokoppler ausgestattet sind. Diese können für eine GALA oder Speedpulse Signal und für Signale vom Drehzahlmesser (aktuell nur WBX getestet) verwendet werden.
- 2 analoge Eingänge für Spannungen von bis zu 15,6V. Ein Port ist für Klemme 15 fest vorgesehen. Damit kann festgestellt werden ob gewisse Funktionen gebraucht werden oder abgeschaltet werden könne. Der andere ist für die Spannung von der KI Beleuchtung vorgesehen. Diese Eingänge haben eine größere Schutzschaltung.
- 2 analoge Eingänge für Spannungen bis 10V. Diese sind aktuell für den Abgriff von der Tankanzeige sowie von der Wassertemperatur gedacht.

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.
???BILDER AKTUALISIEREN

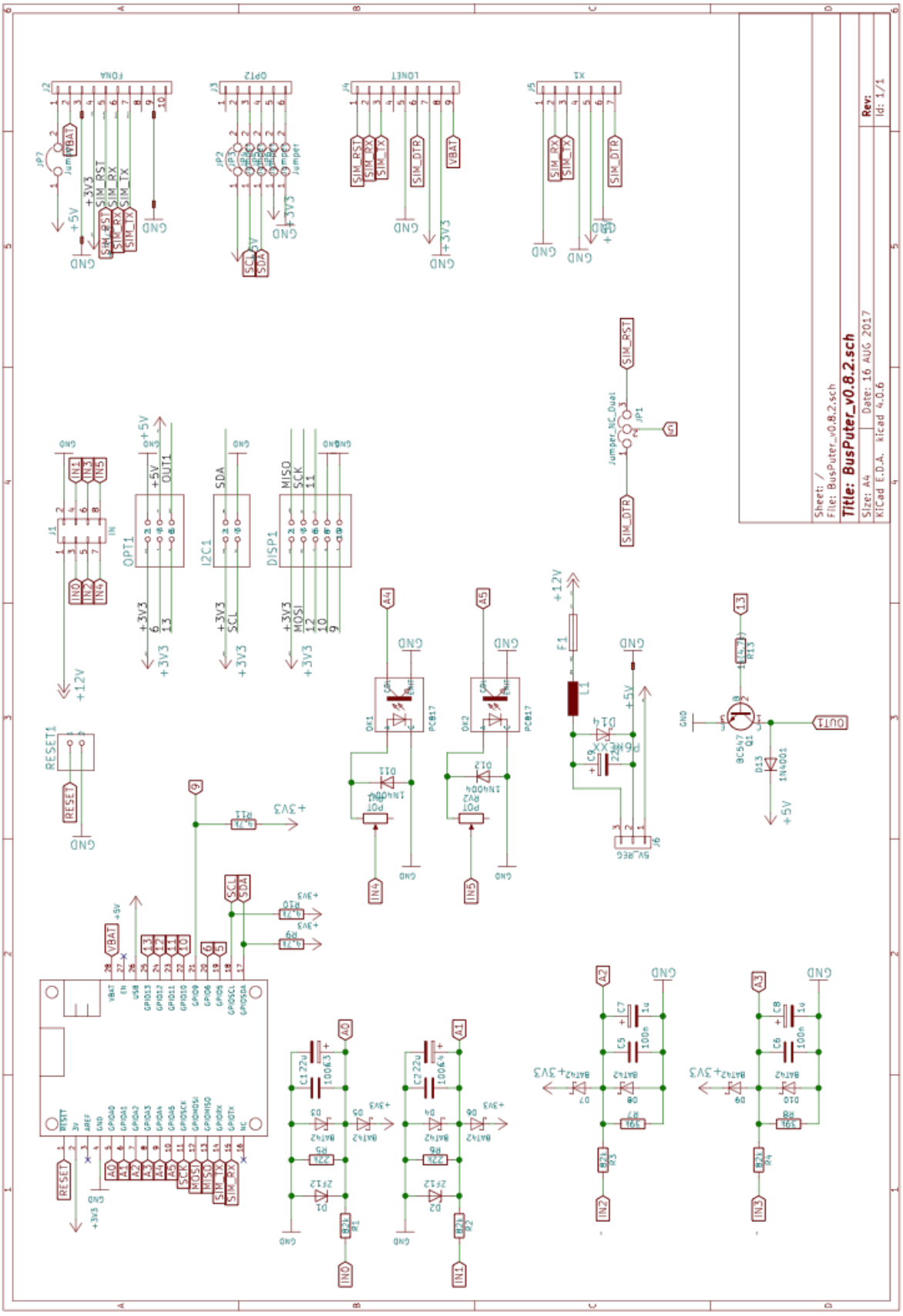

Abbildung 2: Schaltplan des BusPuter Mainboard

3.2 Display

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

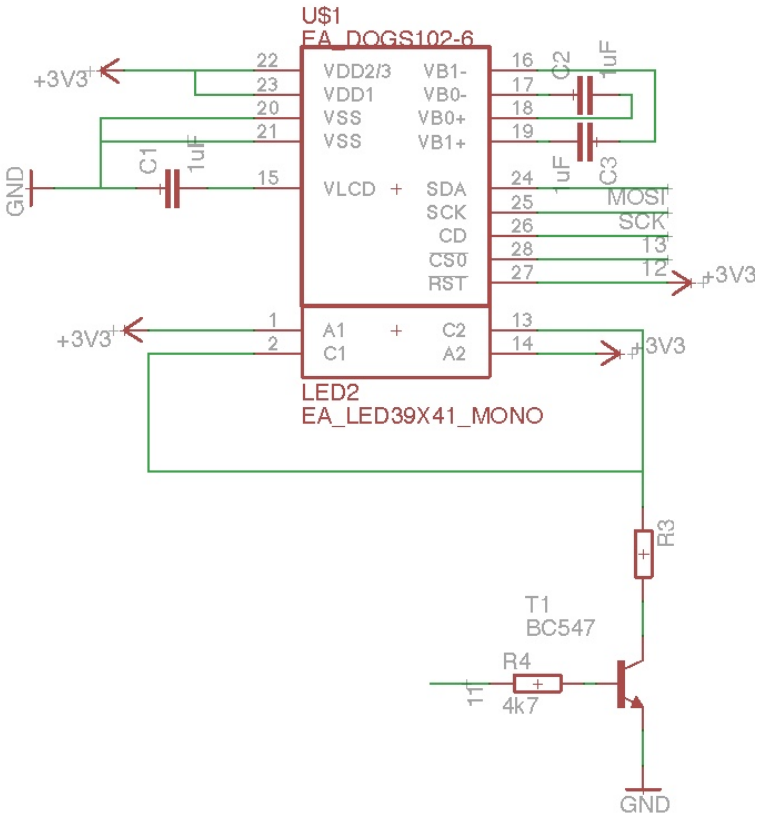

Abbildung 3: Schaltung des LCD

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

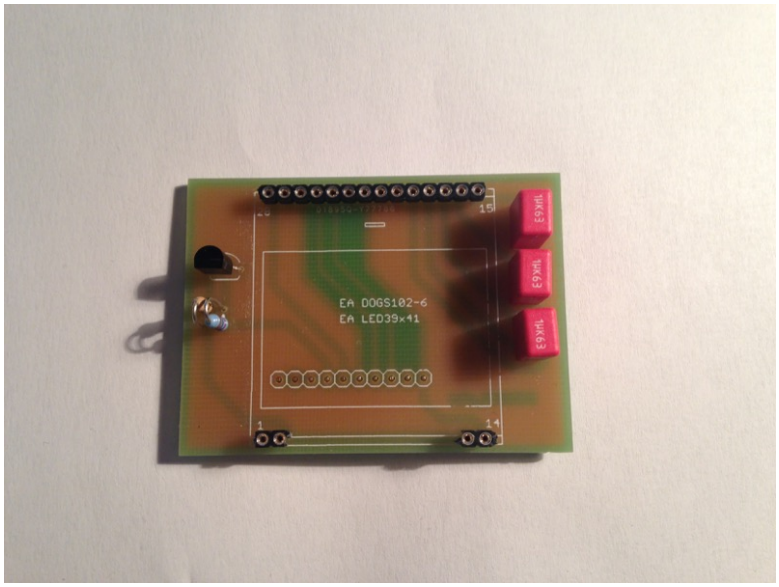

Abbildung 5: Oben

Der Stecker wird wie folgt eingelötet.

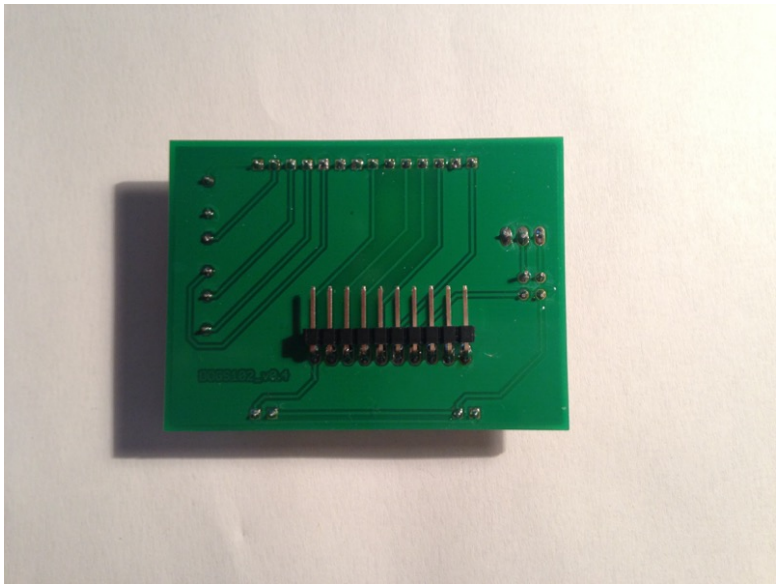

Abbildung 6: unten

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.

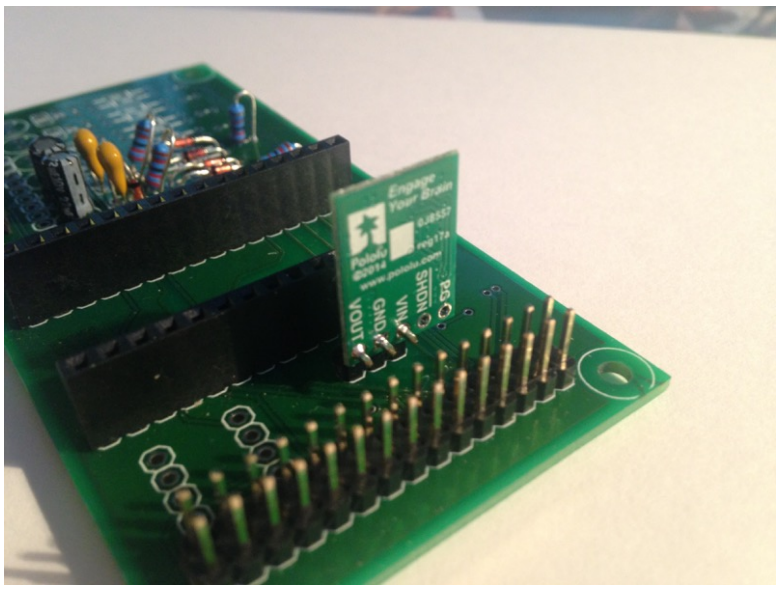

Abbildung 7: Spannungwandler von 12V auf 5V

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.

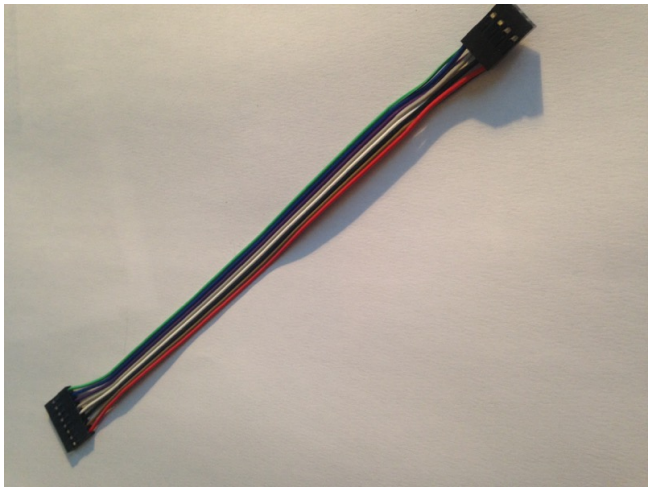

Abbildung 8: Display Kabel

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.

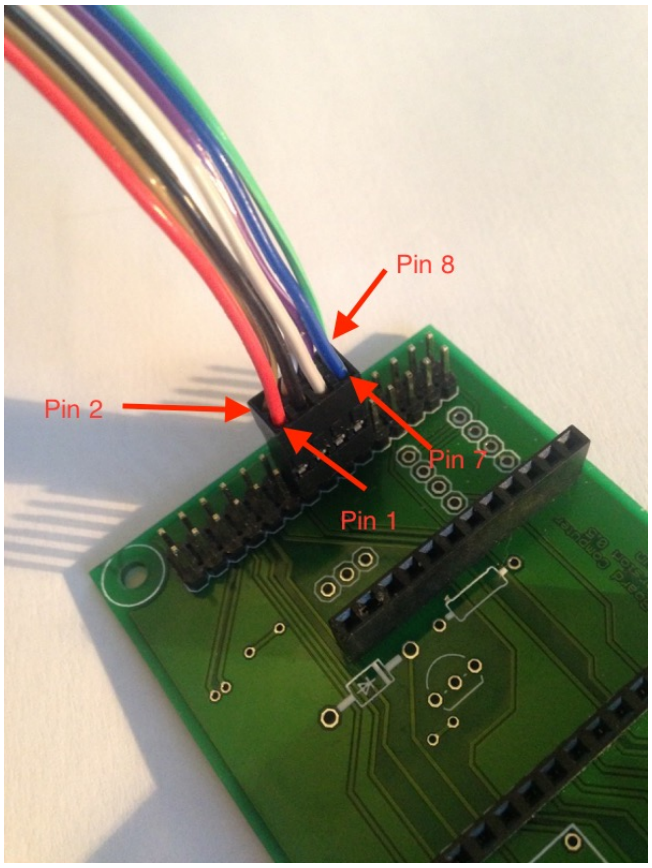

Abbildung 9: Stecker am BusPuter

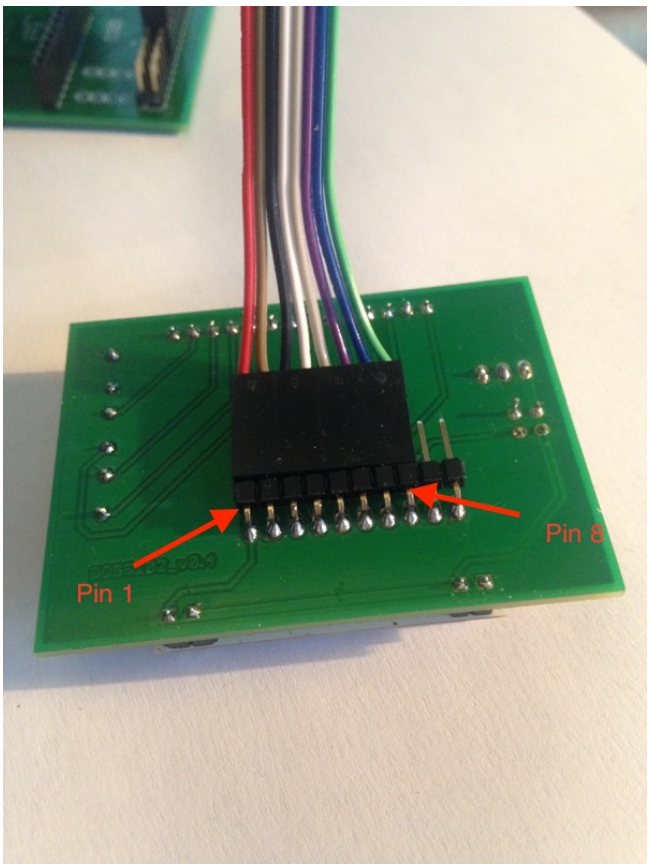

Abbildung 10: Stecker am Display

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 SIM808???

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 3,3V und GND sowie einen Digitalen Pin. Hier könnte z.B. der Pin 25 (Arduino Pin 6) genutzt werden.

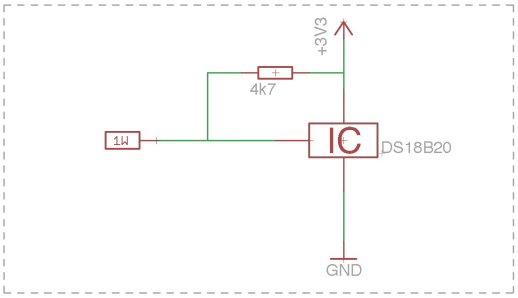

Abbildung 11: OneWire Temperatursensor DS18B20

4.6 sonstige Erweiterungen

5 Konfiguration

In der unveränderten Firmware könnte der BusPuter schon direkt verwendet werden. Zusätzlich können auch noch weitere Parameter angepasst werden.

Damit auch Tracking funktionalitäten verwendet werden können muss auf jeden Fall ein paar Einstellungen gemach werden. Für eine einwandfreie Funktion der Komponenten muss etwas am Sourcecode konfiguriert werden. Dies wird in der “config.h“ gemacht.

5.1 SIM808

```
//SIM808 configuration
#define SIM808
#define SIM_APN "internet.t-mobile"
#define SIM_USER "t-mobile"
#define SIM_PASS "tm"
#define SMS_Keyword "YOUR KEYWORD"
#define BLYNK_KEY "YOUR OWN BLYNK KEY"
```

Mit „#define SIM808“ wird die SIM808 Funktionalität erst aktiviert. Sollte das Module nicht aktiviert werden, dann ist auch keine GPS Bestimmung mehr möglich.

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

ACHTUNG!!! Eine Konfiguration einer PIN ist aktuell noch nicht implementiert (Stand 13.09.17).

5.2 Display

```
// Display Configuration
#define U8G2_DISPLAY
#define DOGS102
//#define OLED
//#define NOKIA

// Port defination
#ifdef U8G2_DISPLAY
#define U8G2_DISPLAY_BG_LED 10
#define U8G2_DISPLAY_CS 11
#define U8G2_DISPLAY_DC 12
#define U8G2_DISPLAY_RST 10

// defination of LCD dimmer
#define DIMMER 2
#define DIMMER_MAX_mV 13800
#define DIMMER_MIN_mV 4000
#define DIMMER_MAX 150
#define DIMMER_MIN 20

#endif // U8G2_DISPLAY
```

Hier wird die Konfiguration für das Display gesetzt. Es ist auch möglich den Busputer ohne Display zu betreiben. Dazu muss die “U8G2_DISPLAY“ auskommentiert werden.

Mit den Werten unter “LCD_LED_MAX“ und “LCD_LED_MIN“ wird der Helligkeitsbereich angegeben. Wird ein OLED verwendet, dann kann kein Dimmer verwendet werden.

5.3 SD Logging

```
// SD Card
#define SDCARD
#define SD_CS 4
```

Aktivieren oder deaktivieren der SD Karte. Falls kein Mitschreiben erwünscht ist oder keine SD Karte vorhanden ist. Ist diese Funktion aktiviert muss eine SD Karte verwendet werden. Ansonsten kann der BusPuter nicht richtig starten

5.3.1 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.

Sie ist per default auf Port 4 konfiguriert. Auch hier kann bei bedarf in der config.h die Anzeige angepasst werden.

5.3.2 Tankanzeige

Die Tankanzeige ist kein Präzisionsinstrument. Sie hat vom Werk aus schon eine relativ hohe Toleranz. Per default auf den Port 3 konfiguriert. Falls die Anzeige falsch geht kann in der config.h diese angepasst werden.

5.3.3 RPM Signal

Das RPM Signal kann vom Kombiinstrument verwendet werden. Das Signal wird auf Port 5 angeschlossen.

```
// RPM gauge
#define RPM_PORT 5
#define RPM_MULTIPL 2
```

Hier muss der Faktor entsprechend eingetragen werden. Dies entspricht die Anzahl der Impulse pro Umdrehung.

5.3.4 GALA/Speedpulse Signal

Dieses Signal kann auf Port 6 angeschlossen werden. Zusätzlich kann noch eingestellt werden was für die Geschwindigkeitsanzeiger hergenommen wird.

```
/*
   Speed source
   1 = Speedpulse
   2 = GPS
*/
#define SPEEDSOURCE 2
```

5.4 I2C Bus

Der Busputer verfügt über einen I2C Bus über den weitere Komponenten angeschlossen werden. 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.

5.4.1 SI7021 Temperatur- und Luftfeuchtigskeitssensor

Momentan ist ein SI7021 Temperatur- und Luftfeuchtigskeitssensor implementiert. Dieser wird automatisch erkannt. Ist kein OneWire Temperatursensor vrbaut, dann wird dieser für die Außentemperatur hergenommen. Ansonsten wird er für die Innentemperatur genommen. Dies kann auch in der config.h überteuert werden.

6 Blynk App

Damit man die Postition seines Fahrzeuges bestimmen kann benötigt man die Blynk App. Diese gibt es für IPhone und Android.

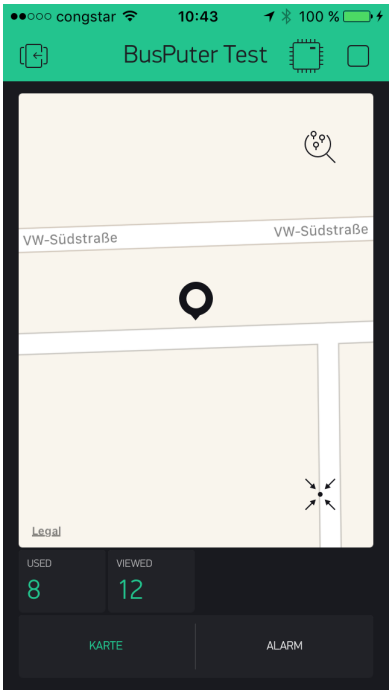

Abbildung 12: Blynk App

6.1 Einrichten

Nach der Installation der App muss man sich als ersten Anmelden und ein neues Projekt anlegen.
Hier wählt man dann folgende Punkte

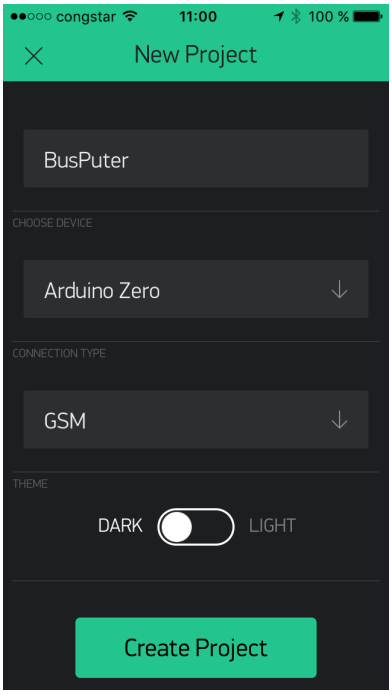

Abbildung 13: Blynk Einstellungen für neues Projekt

Man bekommt dann einen Auth Token. Dieser wird nun im BusPuter eingerichtet. Dieser wird in der config.h eingetragen und anschließend auf die Hardware gebrannt.

```
/*
#define BLYNK_KEY "Your Auth Token"
```

6.2 Karte

Damit man die Position auf einer Karte sehen kann, muss das Widget “Map“ hinzugefügt werden. Diese muss dann wie folgt eingestellt werden.

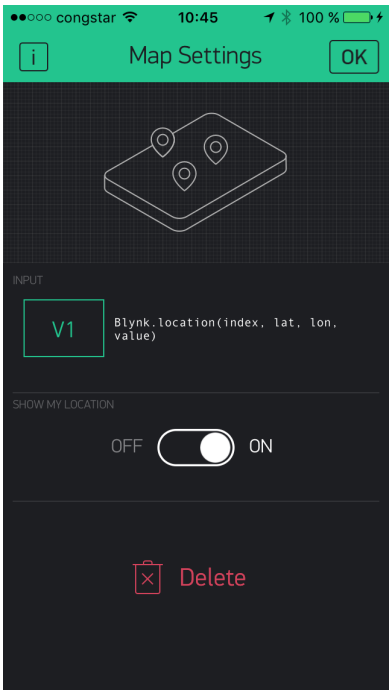

Abbildung 14: Blynk Karte konfigurieren

6.3 Notifications

Der BusPuter kann über die Blynk App auch Push Nachrichten schicken. Diese werden dann sofort angezeigt auch wenn die App gerade nicht offen ist.

Dazu muss das Widget “Notification“ hinzugefügt werden.

7 Erweiterte Informationen

7.1 Debugging

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

Es gibt verschiedene Debuglevels. Die Ausgabe erfolgt nur über die Serielle Konsole. Im normalen Betrieb sollte aber jede Ausgabe abgeschaltet werden.

8 Eigener Code

!!! DIESER BEREICH MUSS NOCH ÜBERARBEITET WERDEN !!!

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

8.1 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.

8.2 Funktion custom_init

Diese Funktion wird bei der Initialisierung aufgerufen.

8.3 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.

8.4 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.