

BusPuter Dokumentation

für HW Version 0.8

Brun von der Gönne

1. Februar 2018

Inhaltsverzeichnis

1 Vorwort	3
1.1 Intesnion	3
1.2 Der Name	3
2 Grundlegendes	3
2.1 Funktionsumfang	3
2.2 Komponenten	3
2.3 Aufbau des BusPuter	3
2.4 Die Handy App	4
2.5 SD Karte	4
2.6 Updates	4
3 Bedienung	4
3.1 Menüpunkte	4
3.2 Optionen	5
3.2.1 Dimmer	5
3.2.2 Clock and Temp.	5
3.2.3 Speed Offset	5
3.2.4 Schwellwerte/Warnungen	5
3.3 Alarmanlage	6
4 Konfiguration und Anschluss	6
4.1 'config.txt' auf der SD Karte	6
4.2 Power	6
4.3 Dimmer	6
4.4 Kühlwassertemperatur / Tankgeber	6
4.5 Drehzahmmesser	6
4.6 Speedpulse / Gala	7
4.7 Drucksensoren	7
4.8 Temperatursensoren	7
4.9 Blynk App	7
4.9.1 Einrichten	7
4.9.2 Karte	8
4.9.3 Notifications	9
5 Selber zusammenbauen	10
5.1 Komponenten	10
5.2 SIM808 Modul	10
5.3 Einkaufsliste	11
5.4 Mainboard	11
5.5 Display	13
5.6 Spannungsversorgung	14

6 Zusammenbau	16
6.1 Anschluss vom Display	16
6.2 Spannungsversorgung	17
6.3 SIM808	17
7 Source Code Komplizieren	17
7.1 Arduino IDE einrichten	17
7.2 Konfiguration in der 'config.h'	18
7.2.1 TinyGPS	18
7.2.2 Display	18
7.2.3 SD Logging	19
7.2.4 I2C Bus	19
7.3 Debugging	19
8 Erweiterungen	19
8.1 OneWire Temperatursensor	19
8.2 I2C Bus	19
8.3 sonstige Erweiterungen	19
9 Eigener Code	20
9.1 Definition	20
9.2 Funktion custom_init	20
9.3 Funktion custom_loop	20
9.4 Funktion custom_menu	20

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 verwenden und Einbau ist auf die entsprechenden Vorschriften zu achten.

Ein kommerzieller Vertrieb wird ausdrücklich untersagt.

1.1 Intesnion

Die Intension dieses Projekt war,

1.2 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.

2 Grundlegendes

2.1 Funktionsumfang

Aktuell sind folgende mögliche Funktionen integriert.

- Geschwindigkeitsmessung via GPS
- Geschwindigkeitsmessung via GALA
- Auswertung der Drehzahl
- Auswertung der Wassertemperatur vom Kombiinstrument
- Auswertung der Tankanzeige vom Kombiinstrument
- Anzeigen der aktuellen Position auf einem Smartphone mit der Hilfe der Blynk App
- Alarmierung beim verlassen des Geozaun
- stille Alarmanlage mit einer Meldung in der App

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

2.2 Komponenten

Neben den BusPuter an sich werden noch weitere Komponenten benötigt um den vollen Funktionsumfang nutzen zu können.

- eine aktive SIM Karte ohne PIN (die Eingabe eines PIN ist momentan nicht möglich)
- Kabel und Stecker für die Individuelle Anbringung und Verkabelung
- eine microSD Karte. 4GB sollten ausreichend sein.

2.3 Aufbau des BusPuter

Der BusPuter in der Version 0.8 hat 6 Eingänge die wie folgt aufgeteilt sind.

- 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. Zusätzlich sind diese Eingänge mit einen Poti zur Einstellung des Pegels versehen.
- 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 Schutzbeschaltung.

- 2 analoge Eingänge für Spannungen bis 10V. Diese sind aktuell für den Abgriff von der Tankanzeige sowie von der Wassertemperatur gedacht. Diese können auch für analoge Sensoren von VDO oder anderen verwendet werden. Vorausgesetzt ist allerdings die Implementierung in der Software des entsprechenden Sensors.

2.4 Die Handy App

Für die Bestimmung des Standortes und weiteren Funktionen kann der BusPuter mit der App “BLYNK“ kommunizieren. Diese App nutzt einen eigenen Cloud Dienst. Es ist allerdings auch möglich einen eigenen Server zu nutzen.

Die App kann bei Apple oder Google im Store bezogen werden. Kosten die eventuell anfallen können stehen nicht im Zusammenhang mit den BusPuter Projekt.

Blynk muss aktuell noch manuell eingerichtet werden. Dadurch kann aber jeder sich die Anordnung nach eigenen Belieben gestalten.

2.5 SD Karte

Auf der SD Karte wird die Konfiguration gespeichert und auch ein Log geschrieben.

Das Log wird als eine KML Datei geschrieben. Diese Datei enthält eine Aufzeichnung der GPS Daten und kann mit einem Betrachter (z.B. Google Earth) geöffnet werden.

Es werden auch die ermittelten Werte der einzelnen Ports/Sensoren mitgeschrieben. Diese werden als Kommentare in der KML Datei geschrieben und können mit einem Texteditor angeschaut werden. An einer weiteren Verarbeitungsmöglichkeit wird noch gearbeitet.

2.6 Updates

Die einfachste Möglichkeit ist die Aktualisierung über die SD Karte. Hierzu muss nur das Binary auf die SD Karte kopiert werden und nach “UPDATE.bin“ umbenannt werden.

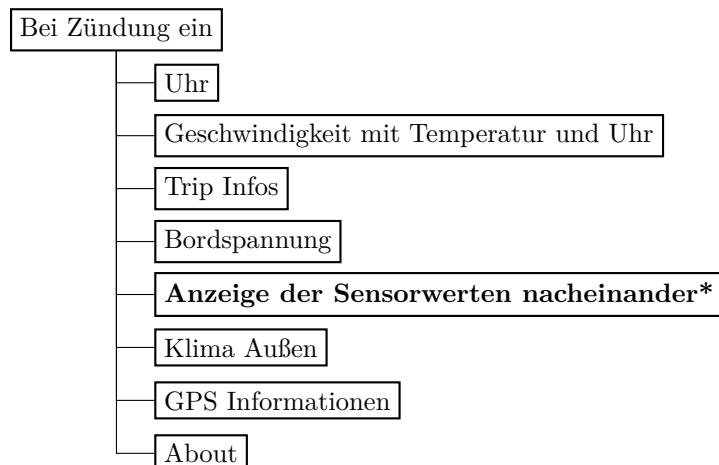
Die Aktualisierung über USB ist auch möglich. Dies geht dann am besten über die Arduino IDE

3 Bedienung

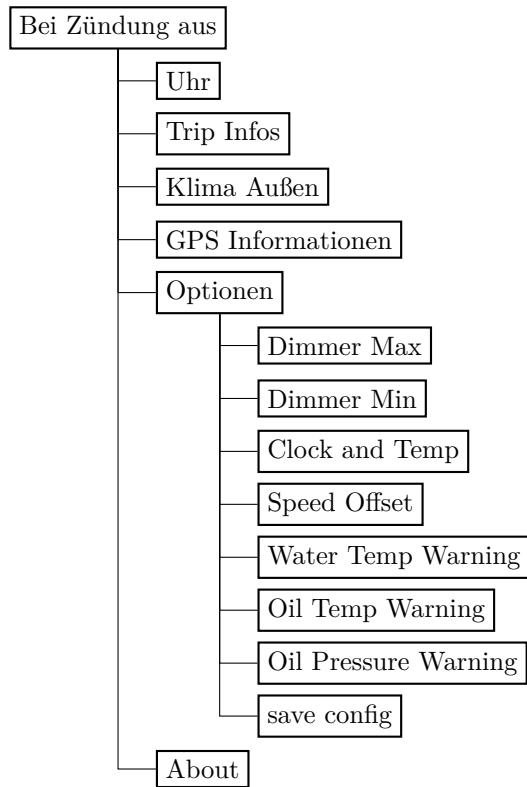
Alles über einen Taster... Die Menüführung ist so gestaltet das alles über einen einzelnen Taster erreichbar ist. Mit kurzen Tastendrücken springt man im Menü weiter und mit langen Tastendrücken kommt man in entsprechende Untermenüs.

3.1 Menüpunkte

Es gibt Menüpunkte die nur bei eingeschalteter Zündung angezeigt werden und welche die nur bei ausgeschalteter Zündung sichtbar sind.



***Anzeige der Sensorwerten nacheinander.** Hier werden die Werte der definierten Sensoren nacheinander angezeigt. Sensoren die keinen Eingang definiert haben werden nicht angezeigt.



3.2 Optionen

Am Ende des Hauptmenü gibt es den Punkt “Optionen“. Dieser ist nur bei ausgeschalteter Zündung sichtbar. Um in die Optionen zu gelangen muss die Taste lang gehalten werden. Um Optionen zu verändern muss auch die Taste lang gehalten werden. Ebenfalls werden mit einen langen Tastendruck die Werte übernommen.

Am Ende können die Einstellungen gespeichert werden. Sollte man die Änderungen nicht speichern sind diese zwar aktiv aber nach einem Reboot sind diese wieder verworfen.

3.2.1 Dimmer

Es kann die Helligkeit bei Maximum und Minimum vom Kombiinstrument für das Display eingestellt werden. Die gedimmte Helligkeit wird dann linear berechnet.

3.2.2 Clock and Temp.

Hier kann eingestellt werden ob im Standby unter der Uhr das Datum oder die Außentemperatur angezeigt werden soll.

¹ 1 = Uhr und Temperatur
0 = Uhr und Datum

3.2.3 Speed Offset

Hier kann festgelegt wie viel mehr tatsächlich angezeigt werden soll. Dieser Wert wird der ermittelten Geschwindigkeit hinzugerechnet. Dies dient zur Anpassung der Anzeige an den Tacho. Eine negative Beeinflussung ist nicht möglich.

3.2.4 Schwellwerte/Warnungen

Es können verschiedene Schwellwerte oder Warnungen aktiviert oder definiert werden. Wird ein Wert überschritten wird eine entsprechende Meldung auf dem Display angezeigt und falls ein akustisches Signal vorhanden ist auch eine akustische Ausgabe gemacht.

Hier können auch Schwellwerte definiert werden für Sensoren die nicht vorhanden sind. Diese Werte haben dann keine Auswirkungen

3.3 Alarmanlage

Bei der Alarmanlage handelt es sich um eine stille Alarmanlage. Der Anschluss einer Sirene ist nicht vorgesehen. Lediglich ein Signalisierung auf Handy ist verfügbar. Sollte eine Alarmanlage schon vorhanden sein so könnte diese angeschlossen werden um ein Alarm auch aufs Handy weiter zu leiten.

Über die Blynk App kann die Alarmanlage gesteuert werden. Sie kann manuell aktiviert werden und der Radius des Geozaun kann festgelegt werden.

Der Geozaun ist immer aktiv sobald die Zündung al

4 Konfiguration und Anschluss

4.1 'config.txt' auf der SD Karte

Der Großteil der Konfiguration kann auf einer SD Karte gespeichert werden und wird bei jedem Boot auch von der SD Karte gelesen.

Sollte noch keine 'config.txt' vorhanden sein so kann man die sich erstellen lassen. Dazu geht man unter Optionen einmal durch die Optionen durch und am Ende kann man die Konfiguration speichern. Dann wird die Konfiguration auf die SD geschrieben.

ACHTUNG!!! Die 'config.txt' wird bei jedem speichern erneut geschrieben. Kommentare oder anderer Inhalt wird dann komplett überschrieben.

4.2 Power

Pin 1 u. 2 sind für die Spannungsversorgung. Diese sollte unter 20V und über 6V liegen. ;-)

```
Pin 1 = +12V  
Pin 2 = GND
```

4.3 Dimmer

Der Dimmer des Kombiinstrumentes wird an Port 2 angeschlossen.

4.4 Kühlwassertemperatur / Tankgeber

Die Kühlwassertemperatur bzw. der Tankgeber kann an Port 3 oder Port 4 angeschlossen werden.

In der config.txt muss nur noch der entsprechende Port angegeben werden.

```
# Fuel Gauge Port  
fuel_port=3  
# Water Gauge Port  
water_temp_port=4
```

4.5 Drehzahmmesser

Das Drehzahlsignal kann auf Port 5 oder Port 6 angeschlossen werden.

Der Multiplikator wird als Index nach folgender Tabelle eingestellt.

Index	Wer
2	2 (WBX / 4 Zylinder Benziner)
3	2,5 (5 Zylinder Benziner)
4	keine Ahnung

In der config.txt müssen dann folgende Parameter entsprechend eingestellt werden.

```
# RPM Port  
rpm_port=5  
# RPM Multiplikator  
rpm_multipl=2
```

4.6 Speedpulse / Gala

Wenn man einen Gala Anschluss oder einen Speedpulse verbaut hat kann dieser auf Port 5 oder Port 6 angeschlossen werden.

```
1 # GALA Port  
speedpulse_port=6
```

4.7 Drucksensoren

Noch nicht fertig!

Drucksensoren können für Öldruck oder Benzindruck verwendet werden

4.8 Temperatursensoren

Noch nicht fertig!

Temperatursensoren können folgende Typen sein ... und für Wasser oder Öl verwendet werden.

4.9 Blynk App

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



Abbildung 1: Blynk App

4.9.1 Einrichten

Nach der Installation der App muss man sich als ersten Anmelden und ein neues Projekt anlegen.

Hier wählt man dann folgende Punkte

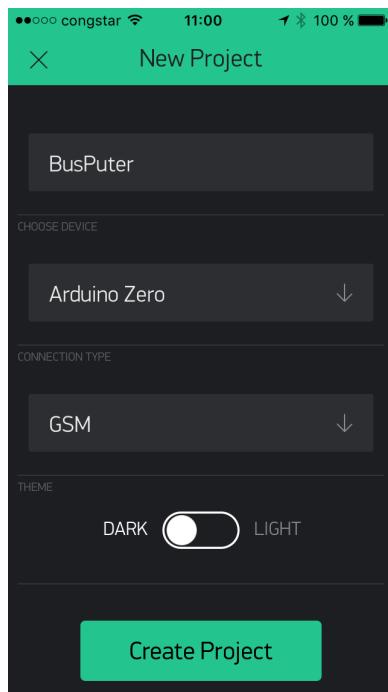


Abbildung 2: Blynk Einstellungen für neues Projekt

Man bekommt dann einen Auth Token. Dieser wird nun im BusPuter eingerichtet. Dieser wird in der config.h eingetragen.

```
blynk_key = ...
```

4.9.2 Karte

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

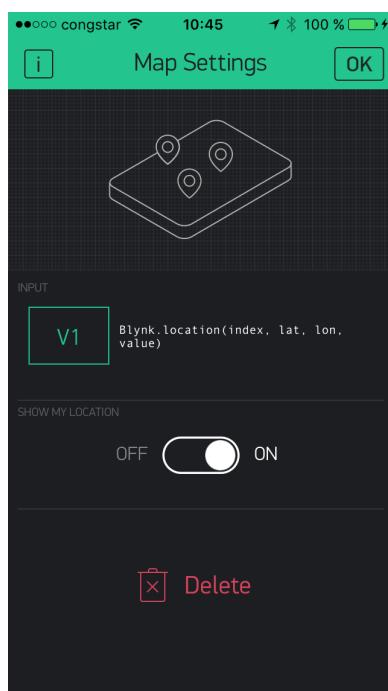


Abbildung 3: Blynk Karte konfigurieren

4.9.3 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.

5 Selber zusammenbauen

5.1 Komponenten

Folgende Komponenten werden zusätzlich 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)
- 5V Spannungswandler Wenn man ein Modul ohne LiPo Anschluss hat sollte die Spannungsversorgung nicht zu klein sein. Getestet wurde es mit den Modulen von Polulu. 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.

5.2 SIM808 Modul

Das SIM808 Modul 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.

Aktuell sind drei verschiedene Module getestet. Alle haben ihre Vor- und Nachteile.



Abbildung 4: Stecker am Display

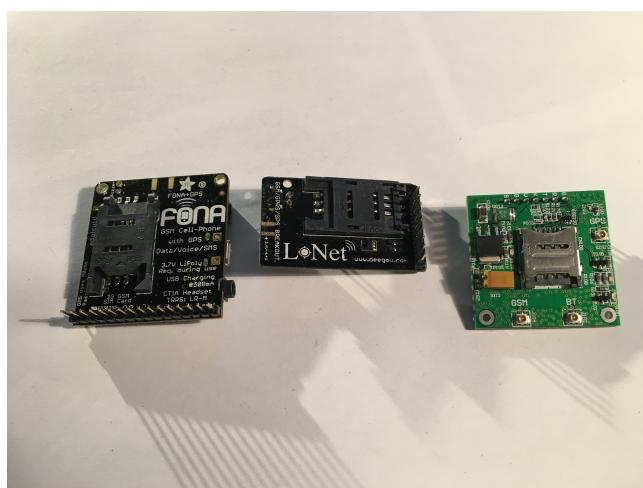


Abbildung 5: Stecker am Display

•	Adafruit FONA808	LoNet808	and-global BK-808 v2
Preis	hoch	mittel	günstig
Verfügbarkeit	bei diversen Händlern	nur import aus China	eBay
LiPo	erforderlich	erforderlich	nicht möglich
Backup über LiPo	ja	ja	nein
aktive GPS Antenne	nein (Umbau erforderlich)	ja	ja
SIM	normal	normal	micro SIM

5.3 Einkaufsliste

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

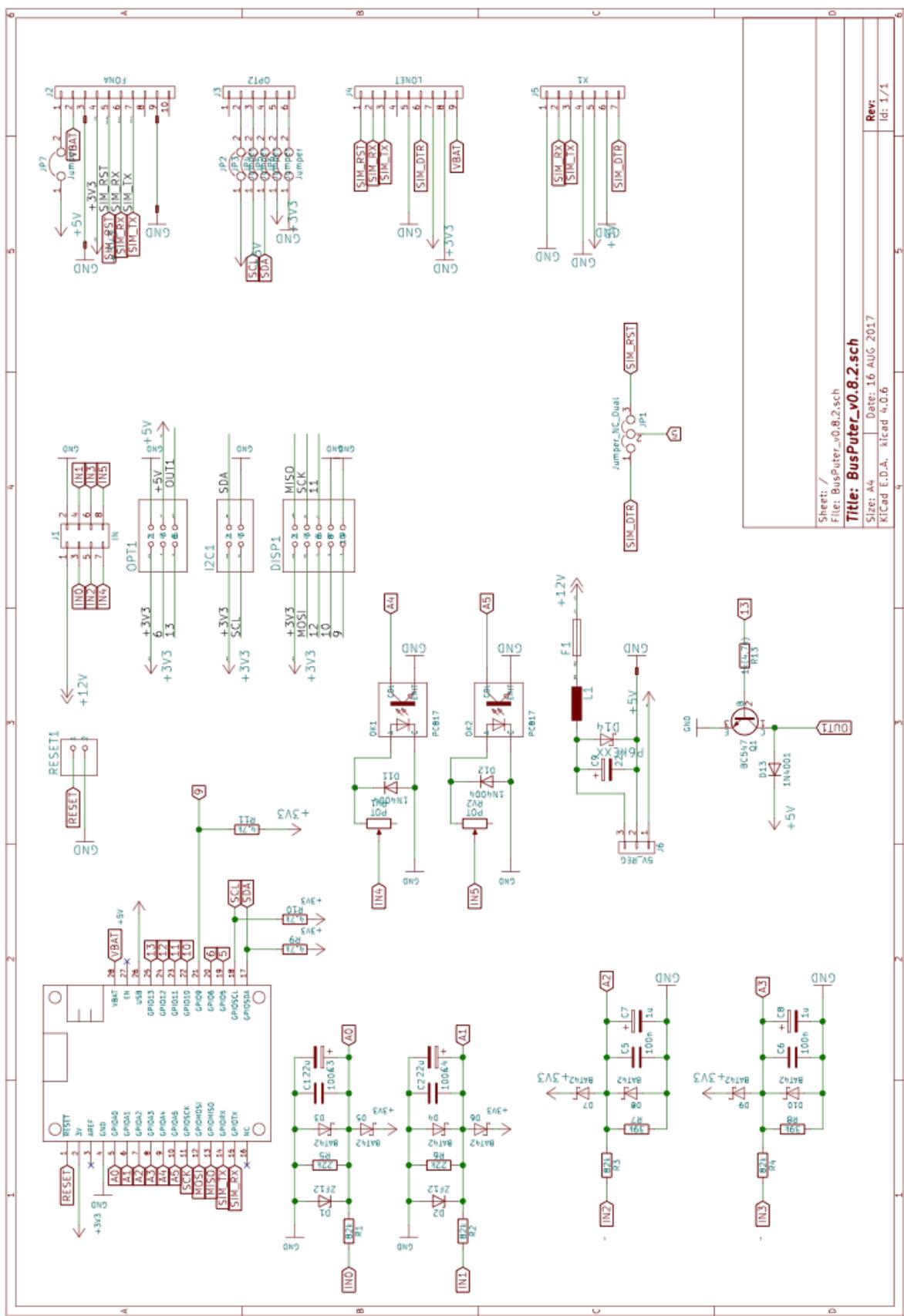
- Elektronische Teile für das Mainboard: <https://www.reichelt.de/my/1375409>
- Teile für das Display mit einem EA DOGS Display mit oranger hintergrundbeleuchtung: <https://www.reichelt.de/my/1386877> Es gibt auch eine Weiße Hintergrundbeleuchtung.
- Gehäuse <https://www.reichelt.de/Kunststoff-Kleingehaeuse/SD-20-GR-HALB/3/index.html?ACTION=3&LA=10030&ARTICLE=149277&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 vergleichbares.
- Den Feather bekommt man zum Beispiel bei 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 Polulu. 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 darauf geachtet werden, dass wir eine Eingangsspannung von 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>

5.4 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



5.5 Display

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

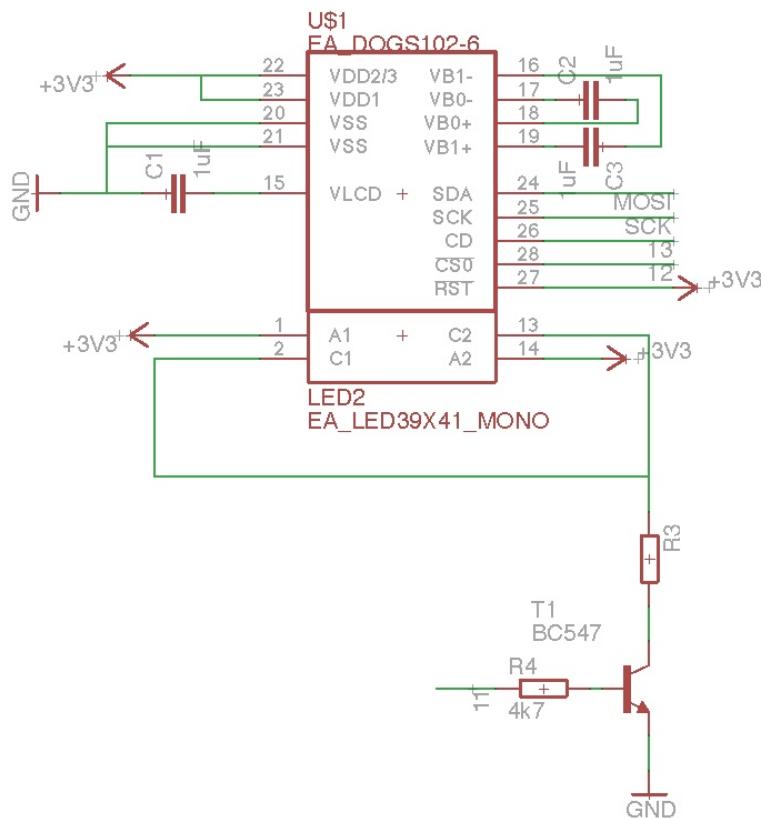


Abbildung 7: Schaltung des LCD

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

Abbildung 8: Auszug aus dem Datenblatt vom DOGS102

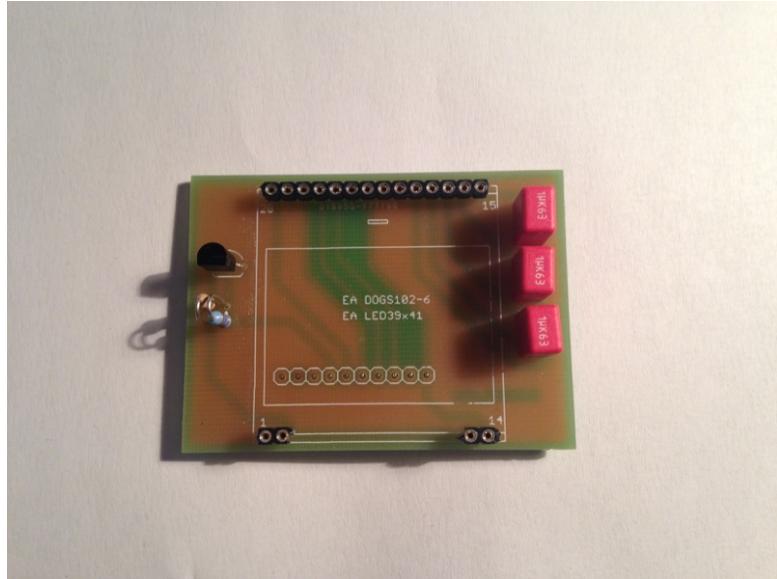


Abbildung 9: Oben

Der Stecker wird wie folgt eingelötet.

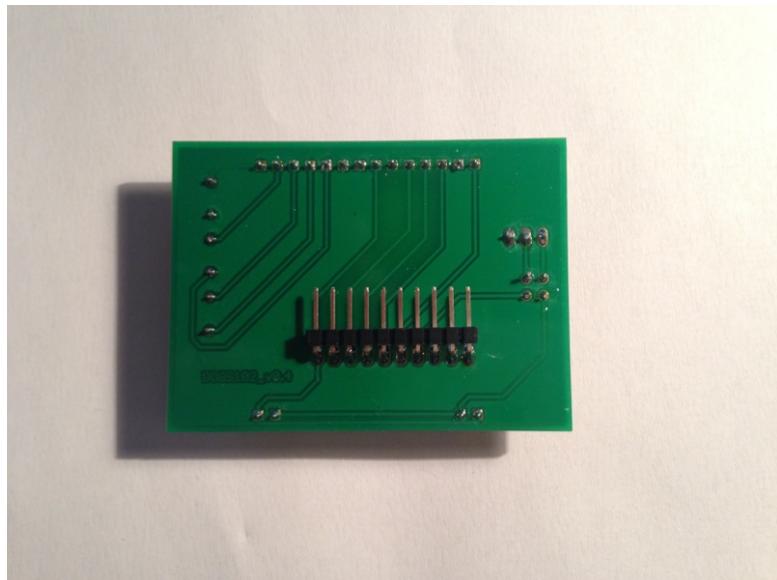


Abbildung 10: unten

5.6 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.

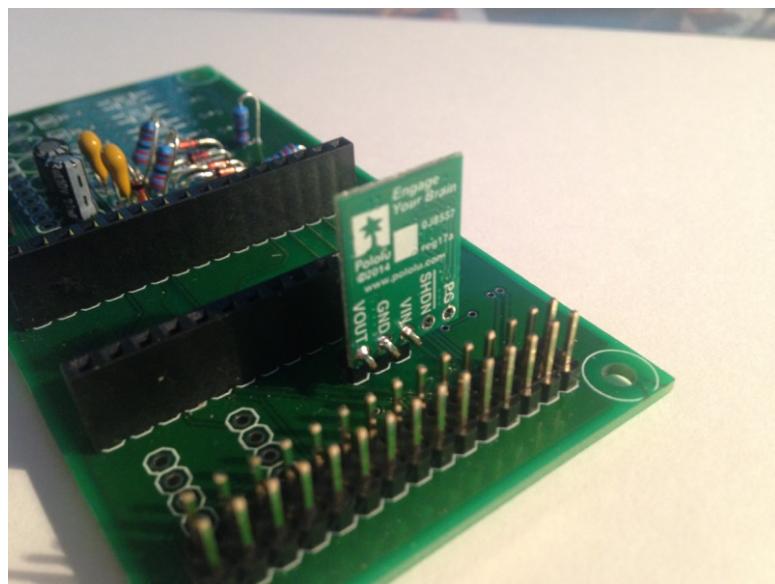


Abbildung 11: Spannungswandler von 12V auf 5V

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

6 Zusammenbau

6.1 Anschluss vom Display

Um das Display den Anforderungen entsprechend 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.

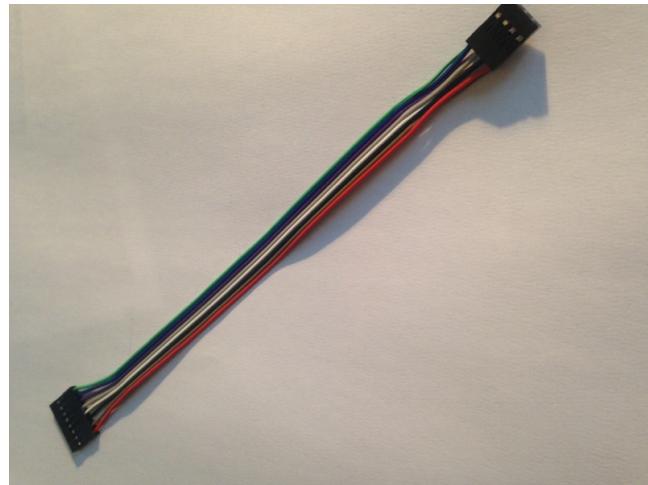


Abbildung 12: Display Kabel

Umbedingt 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.

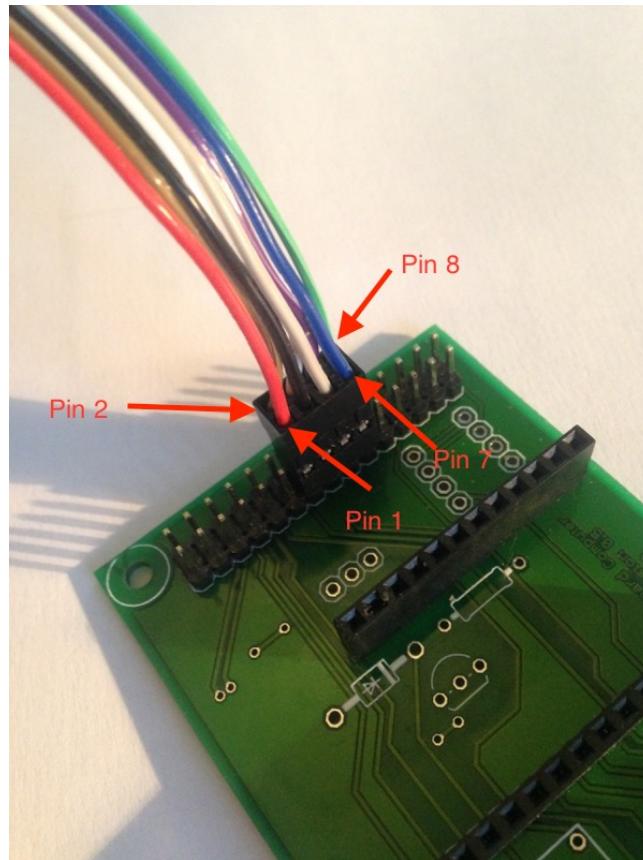


Abbildung 13: Stecker am BusPuter

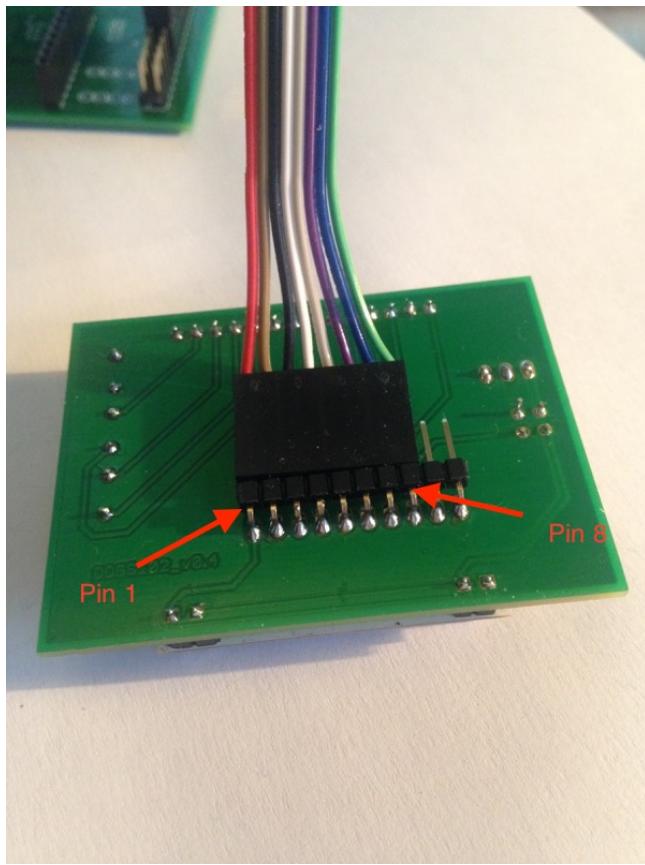


Abbildung 14: Stecker am Display

6.2 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.

6.3 SIM808

Das SIM808 wird einfach aufgesteckt.

ACHTUNG!!! Erst richtig drauf stecken und dann erst mit Spannung versorgen. Sonst gibt das Modul Rauchzeichen von sich. Das ist ein lustiger Effekt aber normalerweise sieht man ihn nur einmal.

7 Source Code Kompilieren

7.1 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.

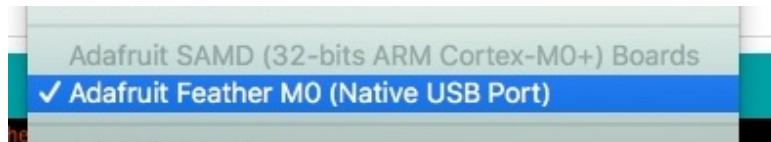


Abbildung 15: Arduino IDE

Anschließend sollte BusPuter compiliert werden können.

7.2 Konfiguration in der 'config.h'

In der "config.h" werden die Default Einstellungen gesetzt. Sobald eine Konfiguration auf einer SD-Karte vorhanden ist, dann wird diese verwendet.

Es gibt auch Parameter die nur hier angepasst werden können. Dazu zählt alles was während der Laufzeit nicht geändert werden kann. z.B. Display, SD-Karte...

7.2.1 TinyGPS

```
//TinyGSM configuration
#define TinyGSM
#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"

#define MYNUMBER "+4915141284285"
#define BLYNK_DEVICE_NAME "MyCar"
```

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

7.2.2 Display

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

// Port definition
#endif 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.

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.

7.2.3 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

7.2.4 I2C Bus

7.3 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 Erweiterungen

8.1 OneWire Temperatursensor

In der BusPuter Firmware ist als OneWire Temperatursensor der Dallas 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.

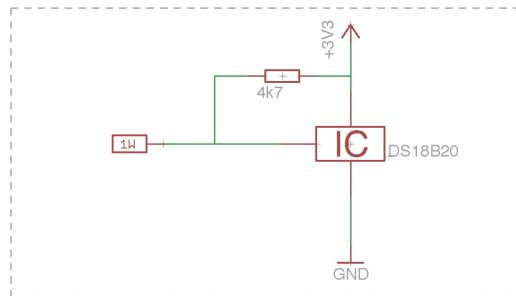


Abbildung 16: OneWire Temperatursensor DS18B20

8.2 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.

8.3 sonstige Erweiterungen

Für sonstige Erweiterungen einfach mal die Mailingliste anschreiben.

9 Eigener Code

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

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

9.1 Definition

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

9.2 Funktion custom_init

Diese Funktion wird bei der Initialisierung aufgerufen. Dies ist gleichzusetzen mit der Setup Funktion im Arduino.

9.3 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 verbraucht wird empfiehlt es sich immer nur nach einer bestimmten Zeit den Code auszuführen. Zum Beispiel:

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

  void custom_loop() {
6   // 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;
11 }
```

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 keinen Watchdog. Sollte der Programmcode nicht in einer entsprechenden Zeit abgearbeitet sein, dann ist das halt so und alles andere muss warten. Eine weitere Verarbeitung von anderen Programmteilen ist in der Zeit nicht möglich.

9.4 Funktion custom_menu

In dieser Funktion kann ein eigener Menüpunkt integriert werden.

Wichtig ist, dass am Ende folgende Zeilen kommen:

```
// this is needed to react 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.