geolink

OpenTracker 2

User Manual

Release **1.3.3**

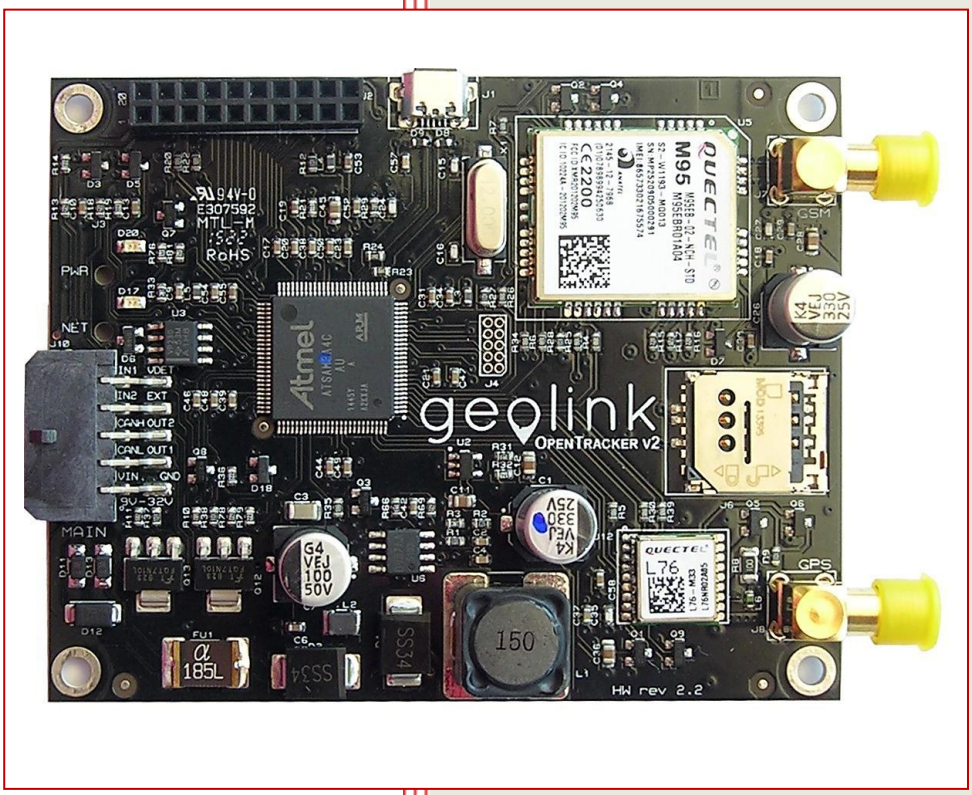


Table of Contents

Introduction

The OpenTracker v2 is the first 100% Arduino compatible, fully open source, commercial grade GPS/GLONASS vehicle tracker development board that comes with a free web interface for tracking.

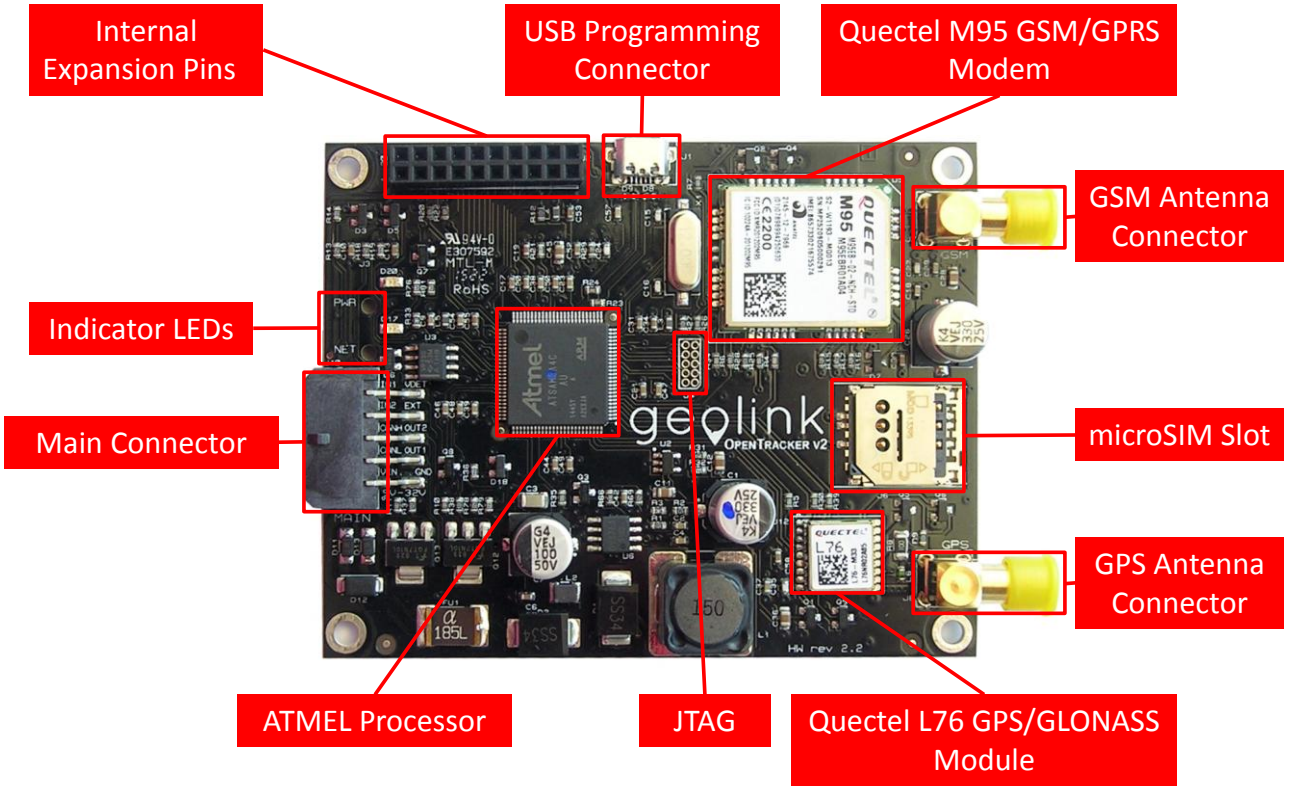
The OpenTracker v2 hardware includes the same powerful 32-bit ARM Controller as the Arduino DUE, a GSM/GPRS modem for wireless connectivity, a GPS/GLONASS module with Assisted-GPS, CAN-BUS, plenty of I/O, and a wide operating temperature range of -35°C to +80°C.

The hardware is pre-programmed and able to send tracking data right out of the box; you will only need the SIM card and to register the device on our free-to-use tracking software.

After the configuration is complete, you will see your device on the map. The amount of trackers per user is unlimited.

Hardware Introduction

Board Overview



Processor

The ATMEL SAM3A4C ARM Cortex-M3 CPU is a 84MHz 32-bit ARM Core microcontroller with 256KB embedded dual-bank Flash (2 x 128 kbytes) and 64 kbytes embedded SRAM.

Processor datasheets are available at http://www.atmel.com/devices/SAM3A4C.aspx

Modem

Quectel Quadband M95 GSM/GPRS Modem

- GPRS data uplink transfer: max 85,6kbps
- Quad-band:
  - GSM850
  - GSM900
  - DCS1800
  - PCS1900

GPS Module

Quectel L76 GPS/GLONASS Module

- Multi-MGSS engine for combined GPS, GLONASS and QZSS
- Easy™, self-generated orbit prediction for instant positioning fix
- Assisted-GPS to enable fast Time-To-First-Fix (TTFF)
- Ultra low tracking power consumption – 18mA
- Always Locate™, an intelligent algorithm for power saving
- 99 acquisition / 33 tracking channels, up to 210 PRN channels
- Supports DGPS, SBAS(WAAS/EGNOS/MSAS/GAGAN)
- Anti-Jamming, Multi-tone Active Interference Canceller

Take note of the IMEI code on the modem (you can also scan the QR code) because you will need it to register your device on the Geolink website.

Quick Start Guide

Install Arduino IDE

The Arduino IDE is an integrated development environment used for OpenTracker.
If you plan to modify the program code or write your own there is no way around it.

Please note:
The tracker comes pre-programmed, if you only want to get started tracking just skip this step.

OpenTracker requires **Arduino IDE 1.6.4** or later. The download should be just below 100MB while downloading the installer and the installation process you may move on the next steps.

Please download latest Version here:
http://arduino.cc/en/Main/Software

Insert SIM Card

OpenTracker has got one Micro SIM slot. Larger SIM cards can be cut into micro SIM in any Store that sells SIM Cards or Mobile phones.

1. Unlock slot

2. Open cover

3. Insert SIM

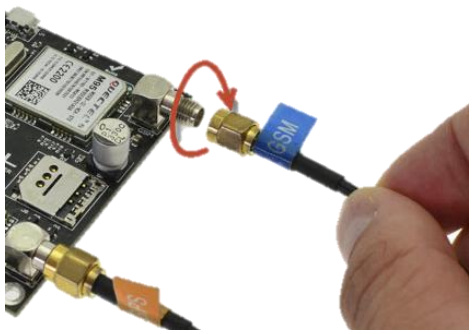
4. Close slot and lock it

For your convenience the SIM PIN should be deactivated.
If this is not an option, the SIM PIN has to be entered into the “tracker.h” and flashed into the Tracker.

Connect Antennas

Take the antenna cable ends, plug them one after another into the antenna connector on the OpenTracker. Turn the antenna cable connector **clockwise**. Ensure to connect the antennas tightly.

Make sure to plug the GPS/GLONASS antenna (here shown orange) into the connector labeled GPS on the board and the GSM antenna (here shown in blue) into the connector labeled GSM.

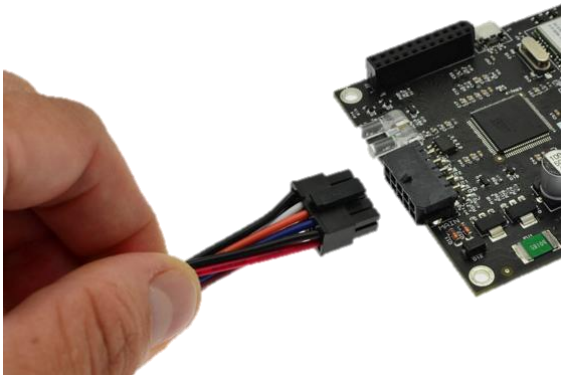


Connect USB Cable

Use a micro USB Type B cable and plug it into the board as shown in the picture below.
This is an optional step. Only necessary if new software has to be installed or developed.

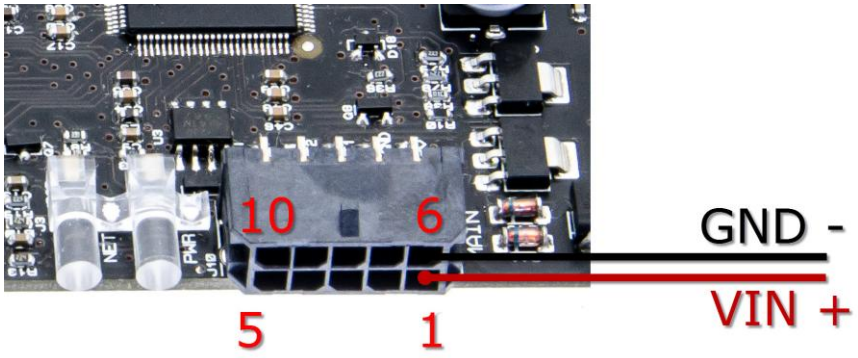
Connect MOLEX Cable

Ensure the Latch on the connector is faced up as shown in the picture below.
Push the connector until it clicks.



Connect Power source

OpenTracker supports 12V/24V nominal power supplies (the full range is 9 – 32 Volts DC).
Please connect Positive (+) terminal to Pin 1 and Negative (–) terminal to Pin 6, as in the picture below.



Start up the tracker

Please ensure to insert the correct data, as wrong information can cause higher costs and even be a source of connection problems.

Setup APN

Send SMS message to the inserted SIM card number to configure APN for your GSM provider (WAP APN are not supported):

```
#pass,apn=APN_NAME
```

Example:

```
#pass,apn=internet
```

The module will reply with the following response:

```
APN saved
```

Send 2 SMS messages to the inserted SIM card number to configure APN username and password (optional for some GSM providers – default username and password are empty):

```
#pass,gprsuser=APN USERNAME
#pass,gprspass=APN_PASSWORD
```

Example:

```
#pass,gprsuser=guest
#pass,gprspass=guest
```

The module will reply with the following response:

```
APN username saved
APN password saved
```

Change SMS password

Optionally, change SMS password by sending following SMS command:

```
#pass,smspass=NEW_PASSWORD
```

Example:

```
#pass,smspass=mynewpass
```

After changing the SMS password, the SMS commands should begin with your new password, like:

```
#mynewpass,COMMAND=ARGUMENTS
```

Please note, depending on the status of the device, it may take a while until the SMS response is sent.
To speed up the process you may turn off ignition or temporarily disconnect VDET.

SMS Commands

Summary of the SMS commands for the initial configuration. The unit will accept only one command at a time and send a reply on successful command execution (this may take a while).

Command	Syntax	Example	Reply
Set APN	#PASS,apn=NEWAPN	#pass,apn=internet	APN saved
Set APN Username	#PASS,gprsuser=NEWUSERN	#pass,gprsuser=guest	APN username saved
Set APN Password	#PASS,gprspass=NEWPASS	#pass,gprspass=guest	APN password saved
Set SMS Password	#PASS,smspass=NEWSMSPASS	#pass,smspass=newpass	SMS password saved
Set Sending Interval (sec)	#PASS,int=INTERVAL_SECONDS	#pass,int=60	Interval saved
Set SIM Pin	#PASS,pin=NEW_PIN	#pass,pin=1234	PIN saved

Please allow some time for the device to reboot after the last SMS reply. Do not immediately turn it off.

Register

To use the Geolink web interface (www.geolink.io) you need to register your OpenTracker device. Please follow this guide: https://geolink.io/guide.php

Make sure your device is configured, powered and ignition is on, so that it connects to the tracking server.

External I/O

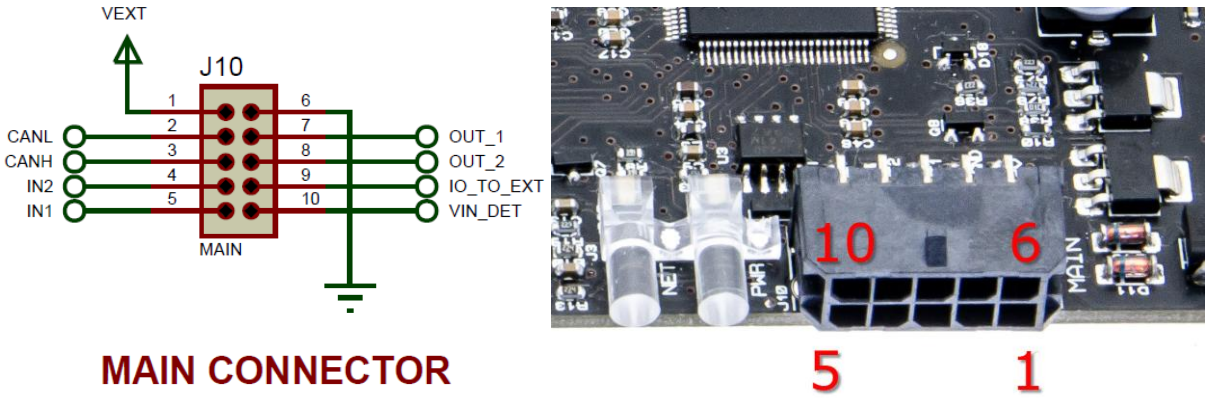
Main Connector Type

We use the Molex 0430451006 Connector on our board.

Micro-Fit 3.0™ Family

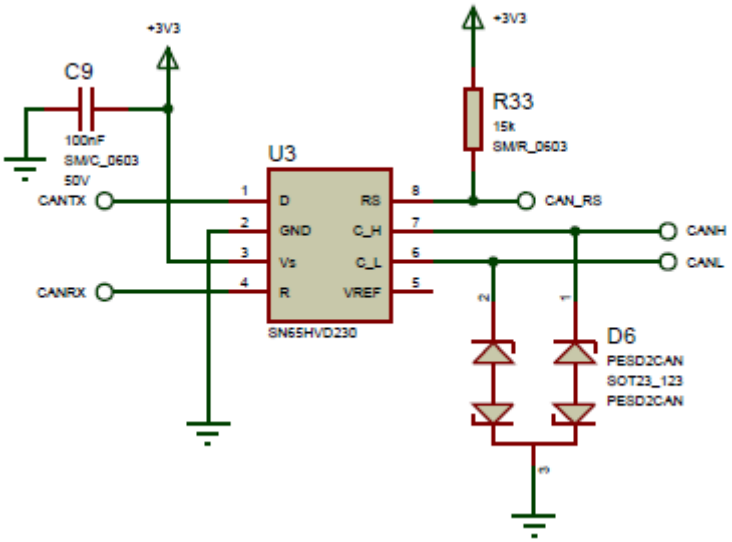
- Fully isolated contacts
- Full polarization
- Positive locks
- 3.00mm (.118") Pitch
- Up to 5.0A per circuit
- 600V AC rating
- UL 94V-0, CSA, TUV approved

Main Connector Pin assignment



CAN BUS Interface

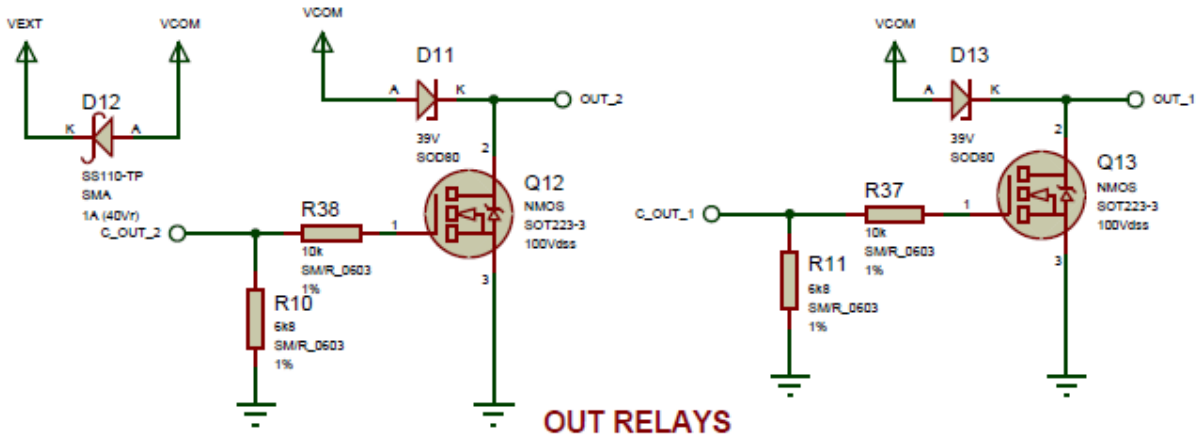
OpenTracker features a standard CAN Bus interface on the main connector. The CAN bus needs a termination resistor (typical 120 Ohm) which is not onboard and will have to be set by the user.
We are using the Ti SN65HVD230 CAN Transceiver. For more info and electrical specifications refer to the Component datasheets at: http://www.ti.com/product/sn65hvd230



Using CAN

We provide example code in our github repository at **https://github.com/geolink/opentracker**.

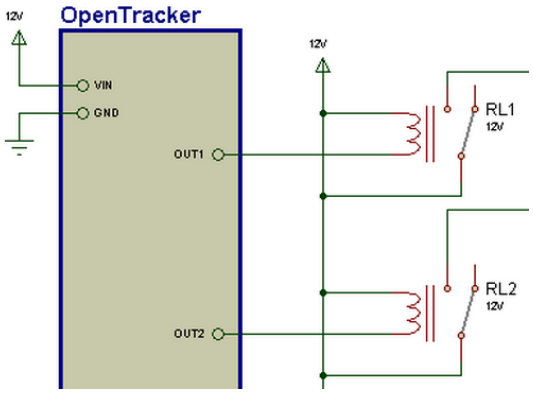
Open Collectors (relay connection)



Operating conditions		
	MIN	MAX
Switching voltage	0VDC	32VDC
Switching current	-	500mA

In this example were using relays with a rated switching voltage of 12VDC. Also the 12V is used to be switched, meaning were will turn on/off 12V rated devices like Lights/Motors/Horns etc.

To do that you will need to connect the first coil contact to 12V and the second to OpenTracker as shown in the picture below.



Once the command is given to switch the relay the PIN OUT1/OUT2 will be switched to ground internally so current can flow and the relays can switch.

Hands on recommendation for Relay usage

- Only use car relays and certified cables inside vehicles!

- A relay socket will ensure proper connection!

- Don´t do it yourself if you are unsure what you are doing. Ask a car mechanic to do this connections for you instead.

Please note:

It is not our responsibility to ensure you connect the relay the appropriate way. Always ensure you read the specifications of the relay you intend to use and the devices you want to operate.

NEVER, work on circuitries under voltage.
ALWAYS, remain within the manufacturers specifications.

Using the Relay outputs (OUT1 / OUT2)

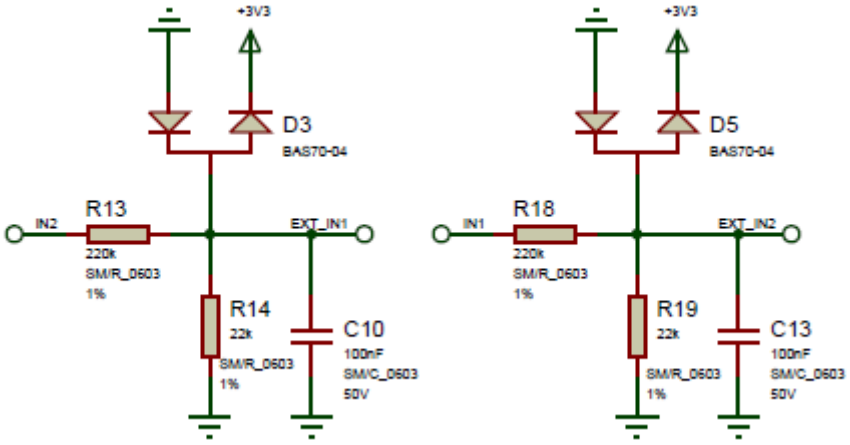
This example shows how to initialize the outputs and switch them on and off.

```
1.    void setup() {
2.        // Relay output
3.        pinMode(PIN_C_OUT_1, OUTPUT); // Initialize pin as output
4.        digitalWrite(PIN_C_OUT_1, LOW); // Set PIN LOW
5.        pinMode(PIN_C_OUT_2, OUTPUT); // Initialize pin as output
6.        digitalWrite(PIN_C_OUT_2, LOW); // Set PIN LOW
7.    }
8.
9.    void loop() {
10.     digitalWrite(PIN_C_OUT_1, HIGH);   // switch the relay on
11.     digitalWrite(PIN_C_OUT_2, HIGH);   //  switch the relay on
12.
13.     delay(3000);               // wait
14.
15.     digitalWrite(PIN_C_OUT_1, LOW);   // switch the relay off
16.     digitalWrite(PIN_C_OUT_2, LOW);   // switch the relay off
17.
18.     delay(3000);               // wait
19.   }
```

Inputs

The two inputs on the Main Connector (IN1 and IN2) are analog inputs and are able to measure voltages up to 32 volts.

Operating conditions		
Analog input Resolution (Vpin = Vinput * 22 / (220+22))	8 bit	
	MIN	MAX
V input	0VDC	32VDC



Using the Inputs (IN1 / IN2)

This example shows how to read the analog inputs on the Main Connector (External IO) and print the output to the debug port.

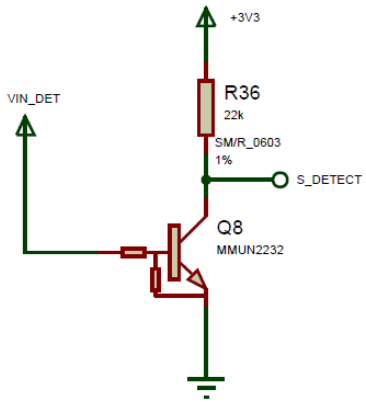
```
1.      #define DEBUG 1          //enable debug msg, sent to serial port
2.      #define debug_port SerialUSB
3.
4.      #ifdef DEBUG
5.        #define debug_print(x)  debug_port.print(x)
6.      #else
7.        #define debug_print(x)
8.      #endif
9.
10.   // Variables will change:
11.   int outputValue;
12.   int sensorValue;
13.
14.
15.   void setup() {
16.     // put your setup code here, to run once:
17.
18.   }
19.
20.   void loop() {
21.
22.   // Read IN1 Value
23.         // read the analog in value:
24.         sensorValue = analogRead(AIN_EXT_IN1);
25.         // map it to the range of the analog out:
26.         outputValue = sensorValue * (242.0f / 22.0f * ANALOG_VREF / 1024.0f);
27.
```

```
28.         // print the results to the serial monitor:
29.         debug_print(F("IN1 = " ));
30.         debug_print(outputValue);
31.         debug_print(F("V ("));
32.         debug_print(sensorValue);
33.         debug_print(F(")"));
34.         debug_port.println(" ");
35.
36.   // Read IN2 Value
37.         // read the analog in value:
38.         sensorValue = analogRead(AIN_EXT_IN2);
39.         // map it to the range of the analog out:
40.         outputValue = sensorValue * (242.0f / 22.0f * ANALOG_VREF / 1024.0f);
41.
42.         // print the results to the serial monitor:
43.         debug_print(F("IN2 = " ));
44.         debug_print(outputValue);
45.         debug_print(F("V ("));
46.         debug_print(sensorValue);
47.         debug_print(F(")"));
48.         debug_port.println(" ");
49.
50.         delay(1000);
51.   }
```

Voltage detection VDET (ignition detection)

The voltage detection is designed to give feedback about the power status. If the Pin VDET is connected to the Ignition line of a car the Tracker is able to detect a logical 1 (Ignition off) or a logical 0 (ignition on) on S_DETECT. This is useful to put the tracker asleep or wake it up. VDET is a Digital input.

Operating conditions	VDET Logic 1 (ignition off)		VDET Logic 0 (ignition on)	
	MIN	MAX	MIN	MAX
V input	0VDC	< 0,7VDC	> 2VDC	32VDC



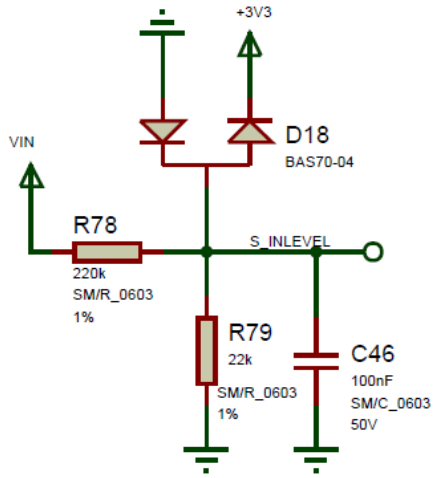
Using Ignition detection (VDET)

The ignition detection is a simple Input to detect if the vehicle is started. By adding code the user can define what to do when ignition is turned on / off. In this example the Tracker will print a string to the debug port.

```
1.      #define DEBUG 1          //enable debug msg, sent to serial port
2.      #define debug_port SerialUSB
3.
4.      #ifdef DEBUG
5.        #define debug_print(x)  debug_port.print(x)
6.      #else
7.        #define debug_print(x)
8.      #endif
9.
10.
11.   void setup() {
12.
13.   // Ignition detection
14.       pinMode(PIN_S_DETECT, INPUT); // Initialize pin as input
15.
16.   }
17.
18.   void loop() {
19.
20.   // Check If Ignition is on
21.     if (digitalRead(PIN_S_DETECT) == LOW)
22.       debug_print(F("Ignition detected!"));
23.   }
```

Battery monitoring (AIN_S_INLEVEL)

OpenTracker hast got the possibility to monitor the input voltage. This is done via the power input (VIN) and no additional cables have to be connected to the tracker.



Using Battery Monitoring

The following example shows how to measure the Supply voltage and print it to the debug port.

```
#define DEBUG 1          //enable debug msg, sent to serial port
#define debug_port SerialUSB

#ifdef DEBUG
  #define debug_print(x)  debug_port.print(x)
#else
  #define debug_print(x)
#endif

// Variables will change:
int outputValue;
int sensorValue;


void setup() {
  // put your setup code here, to run once:

}

void loop() {

// Read VIN Value
      // read the analog in value:
      sensorValue = analogRead(AIN_S_INLEVEL);
      // map it to the range of the analog out:
      outputValue = sensorValue * (242.0f / 22.0f * ANALOG_VREF / 1024.0f);

      // print the results to the serial monitor:
      debug_print(F("VIN = " ));
      debug_print(outputValue);
      debug_print(F("V ("));
      debug_print(sensorValue);
      debug_print(F(")"));
      debug_port.println(" ");

      delay(1000);
}
```

Customizable EXT PIN

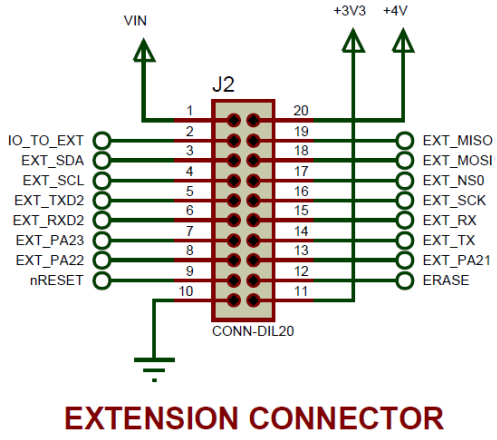
Using EXT PIN

The EXT Pin on the main connector has no function. It is routed to the internal I/O connector.
This may be used for anything the user desires. For example with a 1-wire communication or additional analog lines. Simply connect the IO_TO_EXT Pin on the Internal I/O pin header to your custom setup.

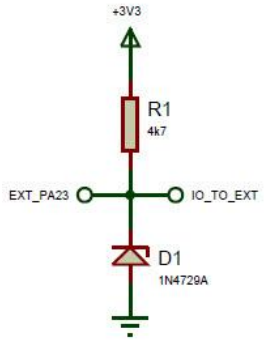
> **Please note:**
> When connecting any Internal I/O Pin to the IO_TO_EXT that those have 3.3V levels and are directly connected to the MCU. We recommend to protect the pin against overvoltage.

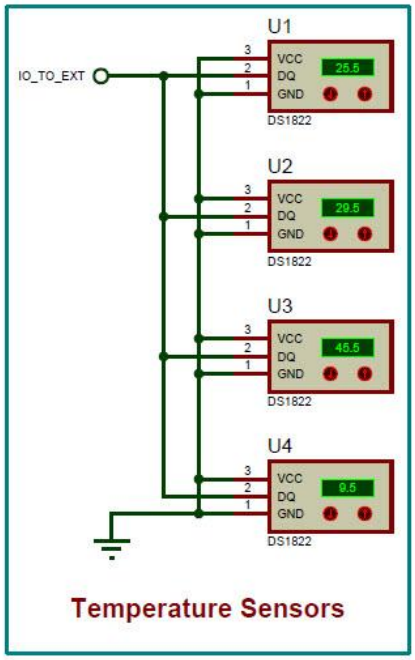
Example usage - One Wire

In this example four DS1820 Temperature sensors are driven in 1-wire parasite power mode. Including basic I/O protection with a 3.6V Zener diode (D1).



EXTENSION CONNECTOR

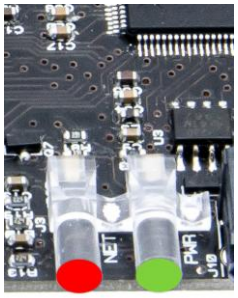


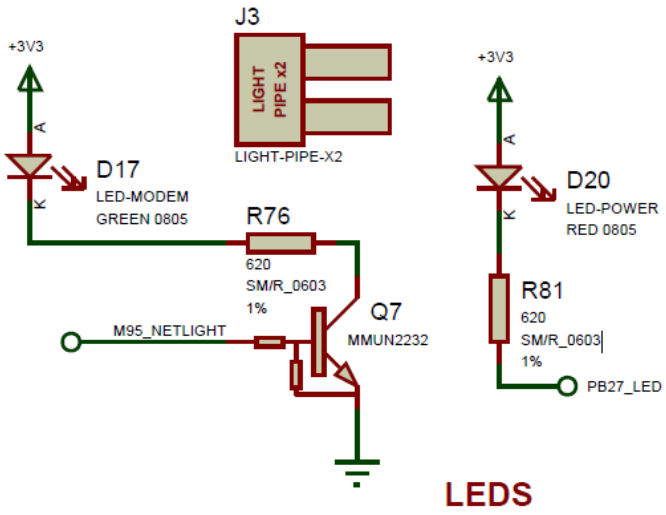


Outside Opentracker

LEDs

OpenTracker has two LEDs for status indication. The Green LED named NET indicates the network status of the M95 modem and cannot be used for other purpose as connected directly to the GSM Modem.
The Red LED can be programmed to indicate various information.





Using the LEDs

PWR LED (red)

The power LED can be programmed.
Please see below the Arduino Example:

```
#include <avr/dtostrf.h>

void setup() {

  //setup led pin
  pinMode(PIN_POWER_LED, OUTPUT); // Set LED as Output
  digitalWrite(PIN_POWER_LED, LOW); // Set LED initially off
}

void loop() {

  // Switch the Power LED
  digitalWrite(PIN_POWER_LED, HIGH);
  delay(800);
  digitalWrite(PIN_POWER_LED, LOW);
  delay(800);

}
```

NET LED (green)

This LED is just used by the M95 modem for network feedback and directly connected to the M95 Pin NETLIGHT.

State	Module function
off	The module is not running
64ms on / 800ms off	The module is not synchronized with network (no network)
64ms on / 2000ms off	The module is synchronized with network (has network)
64ms on / 600ms off	GPRS data transfer is ongoing

Internal I/O

The Internal I/O 20-pin Expansion connector is a standard 2.54mm Pin header which is not populated. It features the following pinout:

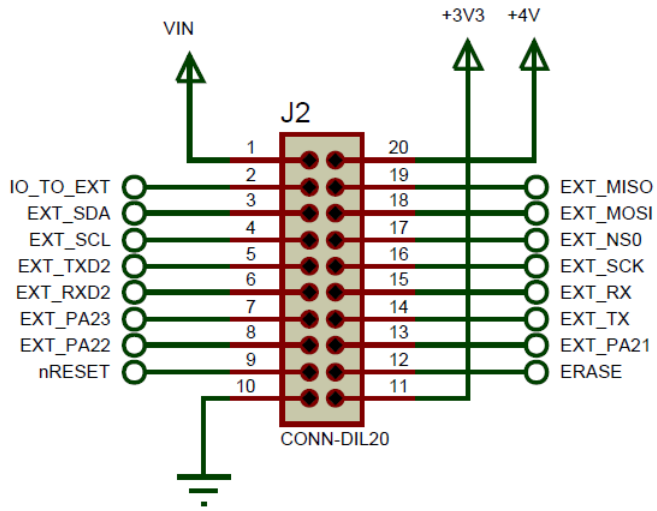
- 1 x SPI
- 2 x UART
- 1 x I2C
- Analog
- PWM
- GPIO
- (nReset)
- (Erase)
- 3.3V
- 4V
- VIN
- GND

PLEASE NOTE: **IMPORTANT NOTE!**

The I/O voltage levels are not 5V tolerant!
Only use 3.3V levels!

The internal I/O are left for custom Expansions done by the user to add functionalities used in special applications.

Schematic

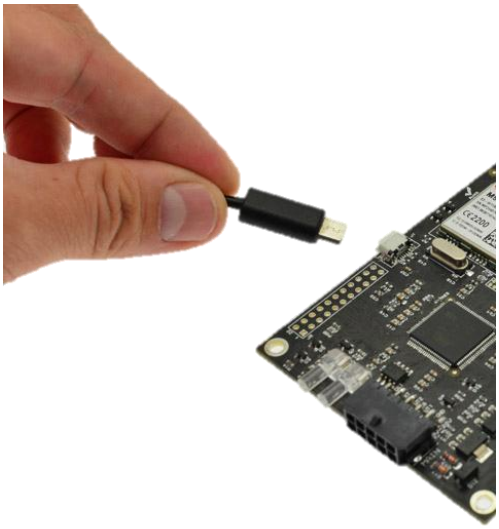


EXTENSION CONNECTOR

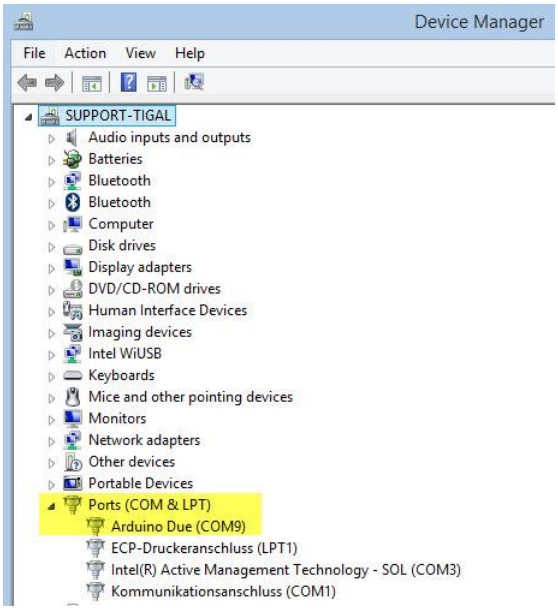
USB Interface

The USB connector is used to program and debug the OpenTracker board.
To be able to communicate with the tracker via USB the tracker needs to be connected to a power supply as well.

1. Install Arduino IDE 1.5.7 or later on the workstation used.
2. Use a micro USB Type B cable and plug it into the board.



3. If the drivers have been installed correctly can be checked with the device manager (Windows only) When board is powered up and the USB cable is connected OpenTracker will be recognized as Arduino DUE in section Ports (COM & LPT) as shown below. Only the COM port number will vary.



Please note:

If you encounter troubles with the USB driver we recommend to read the following pages:
http://arduino.cc/en/Guide/ArduinoDue#toc8

JTAG Interface

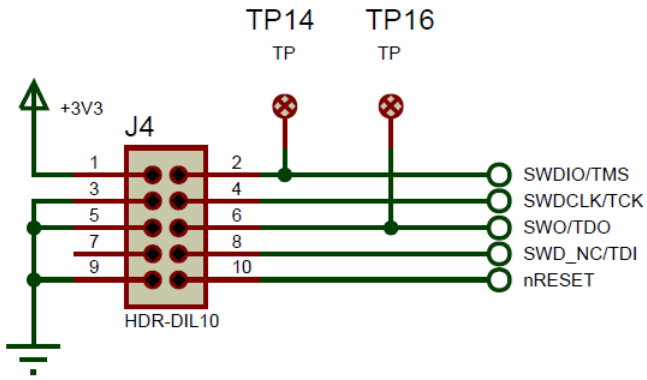
OpenTracker has got a JTAG interface which may be used for programming and debugging. The JTAG connector is a not populated 10-pin .050 inch pitch male header.

The recommended Debugger is the Atmel SAM-ICE for Atmel SAMA5, SAM3, SAM4, SAM7 and SAM9 ARM® core-based microcontrollers in connection with ARM-JTAG-20-10 connector Adapter from Olimex.

More info at:

Atmel SAM-ICE http://www.atmel.com/tools/atmelsam-ice.aspx?tab=overview
ARM-JTAG-20-10 https://www.olimex.com/Products/ARM/JTAG/ARM-JTAG-20-10/

Schematic



Software

Arduino IDE

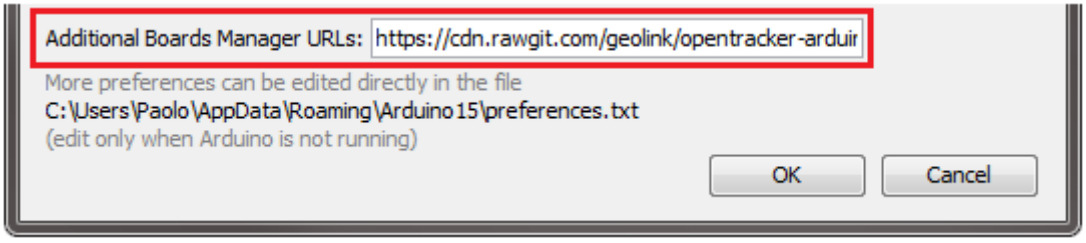
If you just want to use OpenTracker to Track your device you do not need to do these steps. It is all there and configured for this use. But if you want to take advantage of all IO and additional features follow the instructions below and you are ready to develop your own software with Arduino IDE.

Adding OpenTracker 2 as Board to Arduino IDE

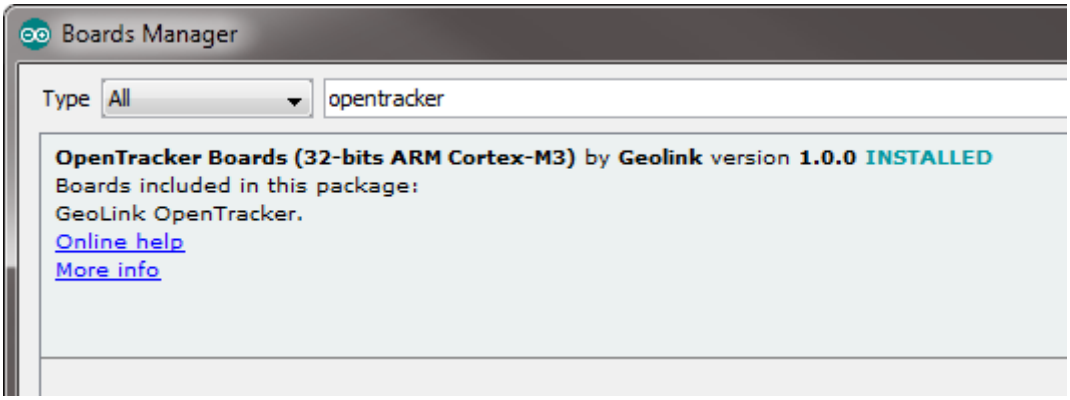
Make sure you have a recent Arduino IDE (version 1.6.4 or later).

Open the Preferences dialog from the File menu and add the following address to the “Additional Boards Manager URLs” text box:

https://cdn.rawgit.com/geolink/opentracker-arduino-board/master/package_opentracker_index.json



Open the Boards Manager window from the Tools menu, write “opentracker” in the text box at the top, choose “OpenTracker Boards” from the list below and click the Install button.



Using Arduino IDE with OpenTracker

Once the previous steps in this chapter have been performed successfully the Arduino IDE is ready to use.

We additionally provide examples and development updates on GitHub.
Please review the repositories regularly at https://github.com/geolink/opentracker.

Troubleshooting

Reset Soft bricked board – Flash locked

During development process it may happen the board will be bricked and compiler output will look like:

```
Erase flash

Write 51852 bytes to flash

Flash page is locked

[                              ] 0% (0/203 pages)

put DueFlashStorage.cpp to arduino-1.5.6-r2/libraries/DueFlashStorage
```

Please follow the below instructions to solve this.

How to Reset a Soft bricked board

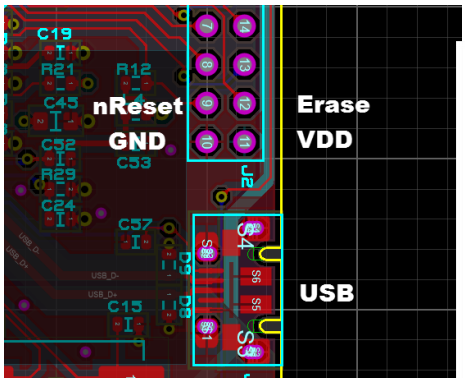
You can try to force a mass erase on the MCU Flash. This is accomplished by:

- power ON
- short NRESET to GND and hold
- short ERASE to VDD and hold
- release NRESET
- wait a few seconds

You can also:

- power OFF
- short ERASE to VDD and hold
- power ON
- wait a few seconds

We placed NRESET and ERASE on the expansion connector and near VDD and GND, so that it's easy to just use a jumper or a clip to perform the above operations:



If the erase is successful, you can reset the board again or power cycle it, and connect the USB cable again.

You will have to change serial port setting on the Arduino IDE, before programming, because the COM port will change to the bootloader COM.

Operating Conditions

Parameter	MIN	Typical	MAX
Operating temperature	- 35°C		+ 80°C
Storage temperature	- 45°C		+ 90°C
Power supply	+ 9 VDC	12 VDC / 24 VDC	+ 32 VDC
Supply current (I GSM/GPS on @ 12V / 24V)		400 mA / 200 mA	
Supply current (I GSM/GPS off @ 12V / 24V)		100 mA / 50 mA	
Supply current (I sleep @ 12V / 24V)		5 mA	
Internal Expansion I/O voltage level	0 VDC	-	3.3 VDC
External I/O VDET (I switch)	0 VDC	-	32 VDC
External I/O IN1 / IN2	0 VDC	-	32 VDC
External I/O OUT1 / OUT2 (Vin switch)	0 VDC		32 VDC
External I/O OUT1 / OUT2 (I switch)			500mA

How to get support

Please feel free to contact us with any questions, queries or suggestions.
If your question is about technical support or troubleshooting for one of our products, we kindly ask you to check first our documentation for a possible solution.

If you cannot find the solution you are looking for, then please write to support@geolink.io providing all possible details.