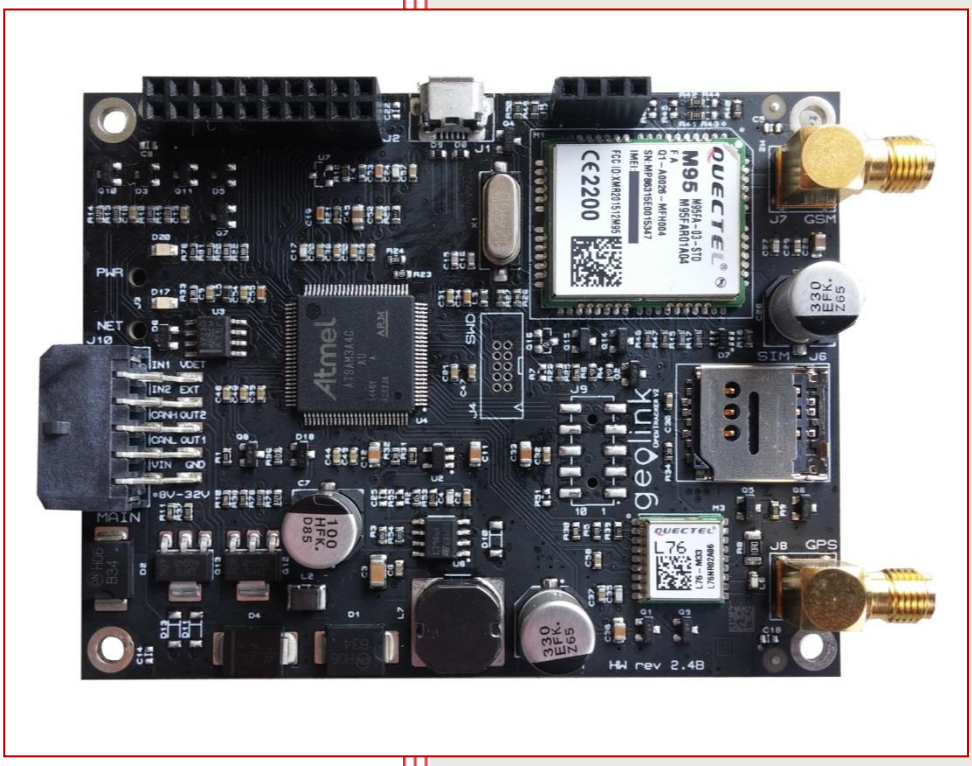
geolink

# OpenTracker 2.4

## User Manual

Release **1.4.0**



# 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.

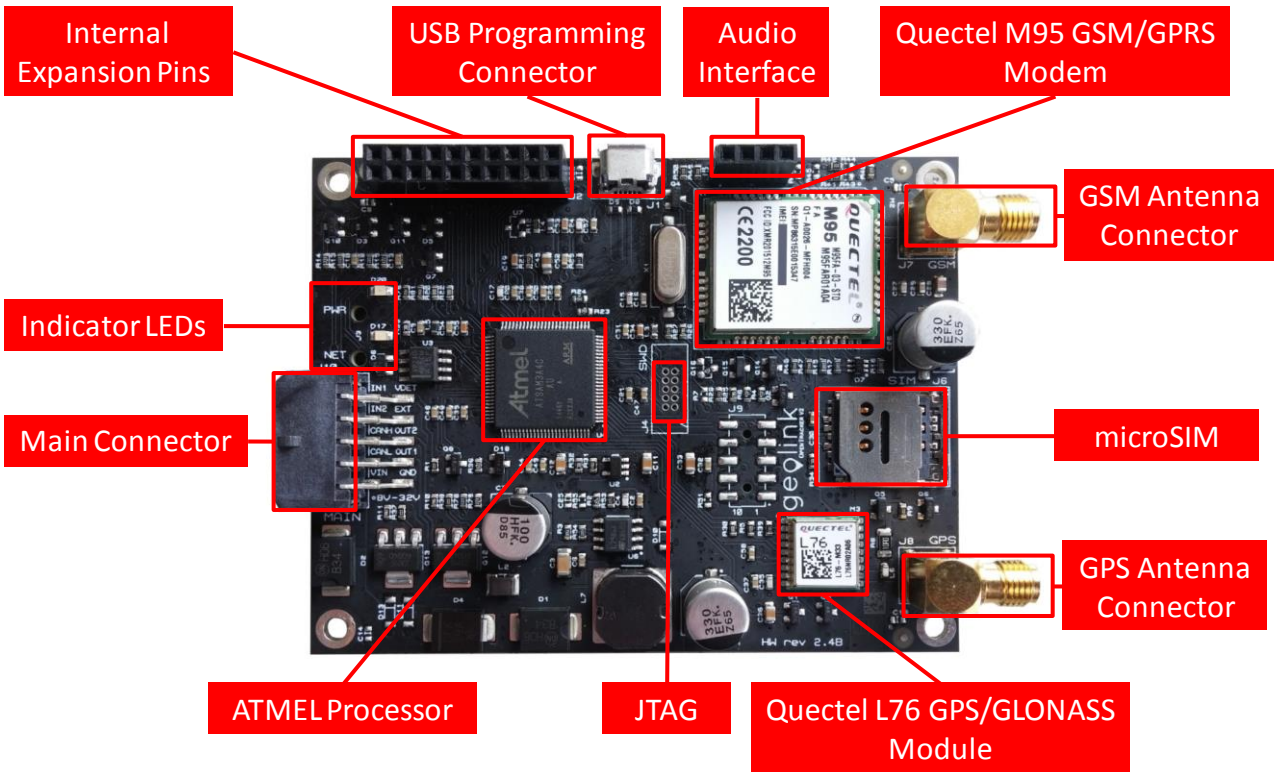
The OpenTracker v2 hardware includes the same powerful 32-bit ARM controller as the Arduino DUE, a GSM/GPRS modem for wireless connectivity, a GPS/GLONASS module with Assisted-GPS, CAN-BUS, plenty of I/O, and a wide operating temperature range of -35°C to +80°C.

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 on-line tracking software.

After the configuration is complete, you will see your device on the map.

# Hardware Description

## Board Overview




OpenTracker is an open-source development platform for vehicle tracking applications, compatible with Arduino software and development environment.

The OpenTracker hardware includes a powerful 32-bit controller (as found on the Arduino DUE), a quad-band GSM/GPRS modem with micro-SIM card socket, GPS/GLONASS module, CAN interface, 2 analog inputs, 2 relay driving outputs, an internal expansion interface (including SPI, I2C, UART, GPIO, ADC and modem audio) and a micro-USB interface to develop additional features.

Design files, firmware and Arduino board support packages can be found on GitHub:
https://github.com/geolink/opentracker-hardware
https://github.com/geolink/opentracker
https://github.com/geolink/opentracker-arduino-board

## Processor

The Atmel SAM3A4C microcontroller is a 84MHz 32-bit ARM Cortex-M3 core with 256KB embedded Flash (2 x 128 KB) and 64 KB embedded SRAM.

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

## Modem

**Quectel M95 - Quadband GSM/GPRS Modem**


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

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.

## GPS Module

**Quectel L76 - GPS/GLONASS Receiver**


- 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

# 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 1.6.7** 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 the 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

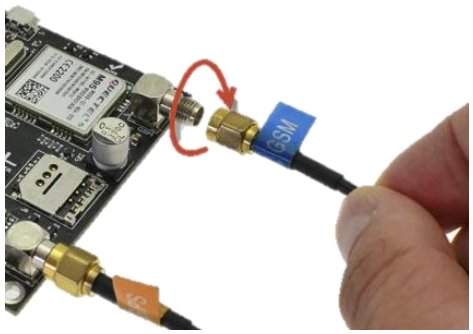
4. Close slot and lock it

> 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.

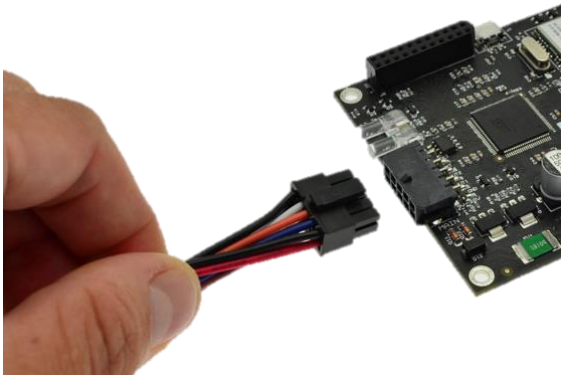


## Connect USB Cable

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

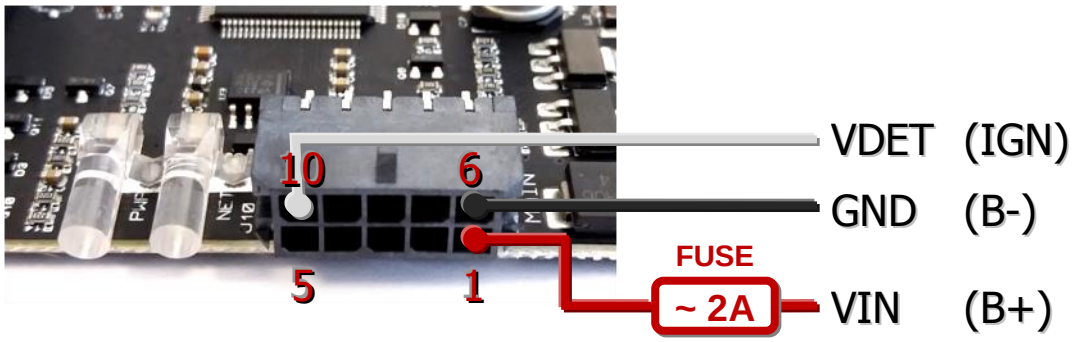
Ensure the Latch on the connector is faced up as shown in the picture below.
Push the connector until it clicks.



## Connect Power source

OpenTracker supports 12V/24V nominal power supplies (the full range is 8 – 32 Volts DC).

Please connect first the battery negative (B–) terminal to pin 6 (black wire), then the positive (B+) terminal to pin 1 (red wire) through a 2A fuse, as in the picture below. Then connect vehicle ignition (IGN) to pin 10 (white wire), or connect pin 10 to pin 1 if ignition output is not available




## Start up the tracker

Please ensure to insert the correct data, as the wrong information may cause higher costs on your SIM bills or 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 current status of the device, it may take a while until the SMS response is sent.

## 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.

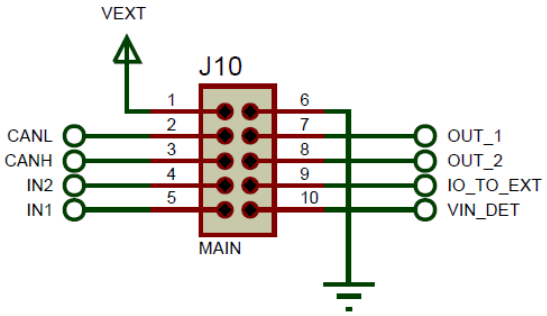
# External I/O

## Main Connector

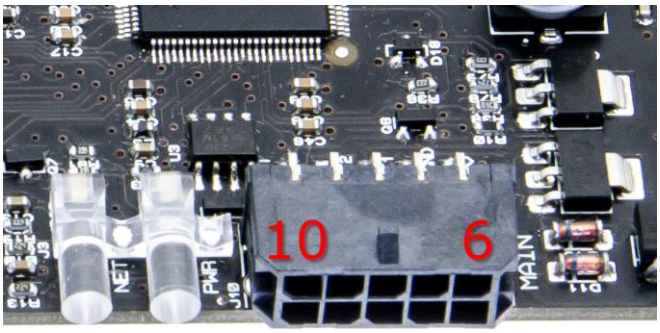
The main connector for external power and I/O is compatible with Molex p/n 0430451006 (Micro-Fit 3.0™)

- Fully isolated contacts
- Full polarization
- Positive locks
- 3.0mm (.118") Pitch
- Current rating: 5.0A (per circuit)
- Voltage rating: 250V AC,DC
- UL 94V-0

### Pin assignment



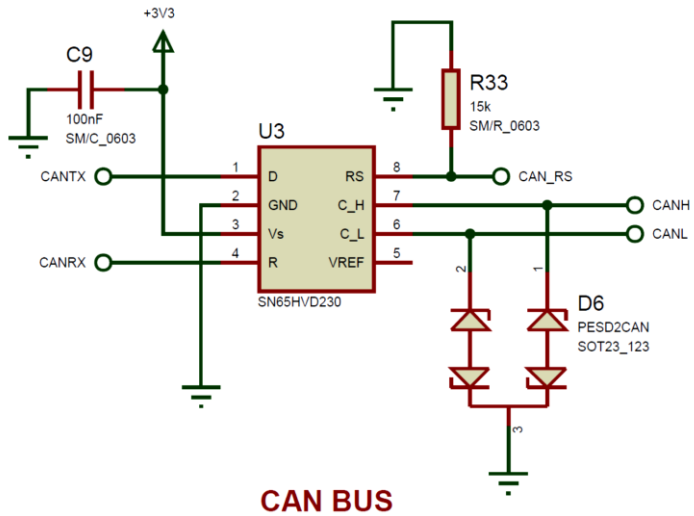

MAIN CONNECTOR




## 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 on-board (but usually present on the vehicle bus).

The transceiver used is SN65HVD230. For more info and electrical specifications refer to the component datasheet at: http://www.ti.com/product/sn65hvd230

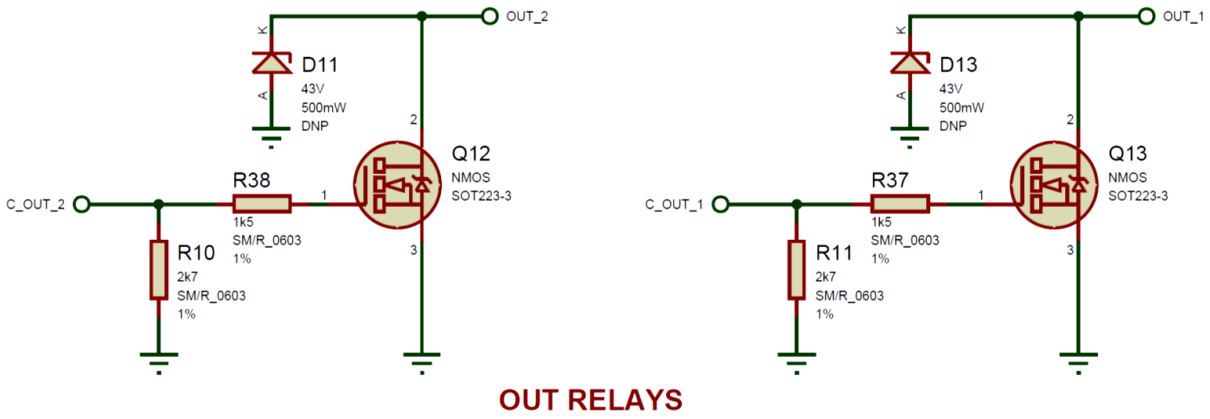



CAN BUS

### Using CAN

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

## Digital Outputs




**OUT RELAYS**

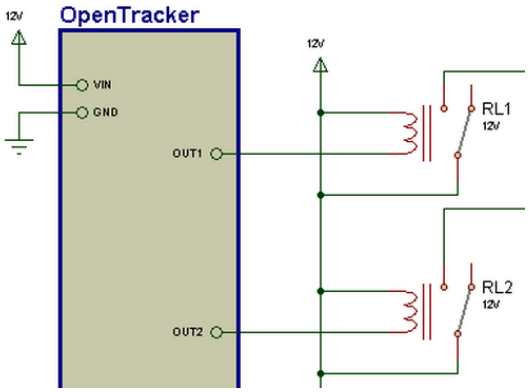
| Parameter | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|
| Relay Output Current | | 0.2 | 0.5 | A |
| Relay Output Voltage1 | 0 | - | 36 | V |

**Notes:**

1. Relay driving outputs OUT1 and OUT2 are open-drain outputs suitable for relay coils, which should be powered externally from the same voltage as the whole device. They are protected against voltage transients and short-circuits.
   Hardware versions earlier than “rev2.4” do not have short-circuit protection, if that is required insert an external fuse in series with each output.

### Example Relay connections

In this example we are using relays with a rated coil voltage of 12V DC. The same voltage is used for switching contacts to power 12V rated devices like Lights/Motors/Horns etc.




When the software activates a digital output, the pin OUT1/OUT2 is connected to ground internally, allowing current flow inside the coil 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.

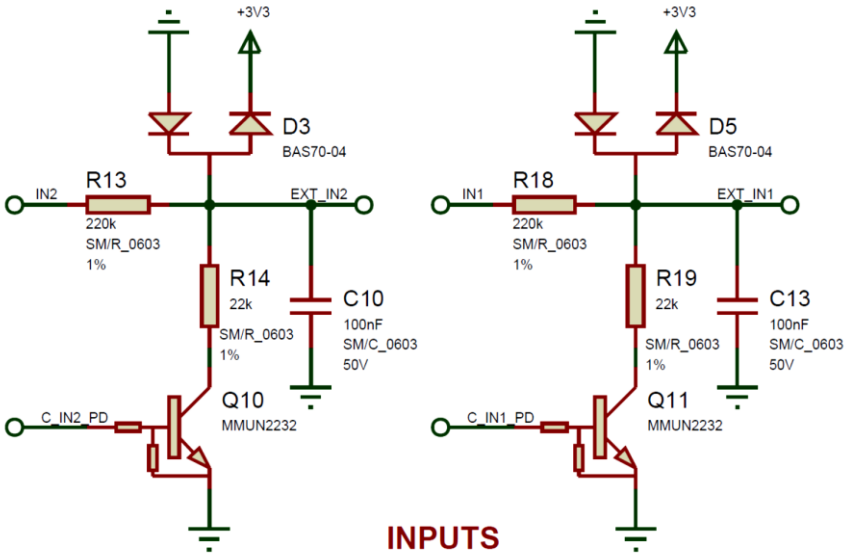
NEVER, work on circuitries under voltage.
ALWAYS, remain within the manufacturers specifications.

## 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.   }
```

## Analog Inputs




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

| Parameter | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|
| Analog Input Voltage2 (Normal Range) | 0 | - | 36 | V |
| Analog Input Voltage2 (Reduced Range) | 0 | - | 3.3 | V |
| Analog Input Accuracy | | ±0.15 | | V |
| Analog Input Impedance | 200 | 230 | 260 | kΩ |
| Analog Input Bandwidth | 7 | - | 79 | Hz |

**Notes:**

2. Analog inputs IN1 and IN2 can be configured by the software to accept a reduced input range to improve precision of analog readings in a low voltage range.

## Using the Inputs (IN1 / IN2)

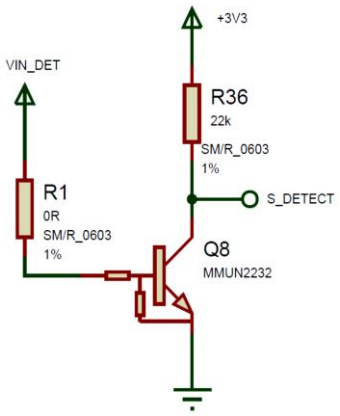
This example shows how to read the analog inputs on the Main Connector (External IO) and print the output to the debug port.

```
#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 IN1 Value
      // read the analog in value:
      sensorValue = analogRead(AIN_EXT_IN1);
      // 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("IN1 = " ));
      debug_print(outputValue);
      debug_print(F("V ("));
      debug_print(sensorValue);
      debug_print(F(")"));
      debug_port.println(" ");

// Read IN2 Value
      // read the analog in value:
      sensorValue = analogRead(AIN_EXT_IN2);
      // 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("IN2 = " ));
      debug_print(outputValue);
      debug_print(F("V ("));
      debug_print(sensorValue);
      debug_print(F(")"));
      debug_port.println(" ");

      delay(1000);
}
```

## Voltage detection VDET (ignition detection)

The voltage detection is designed to give feedback about the ignition 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 | 0.6VDC | 1.5VDC | 32VDC |




IGNITION

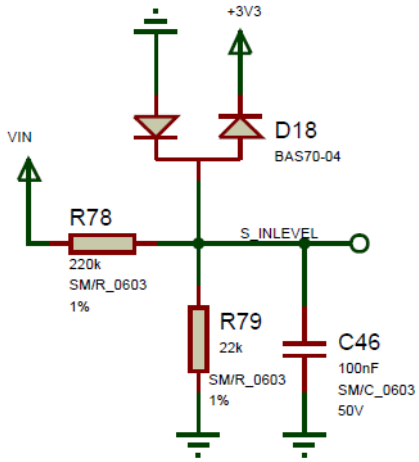
### Using Ignition detection (VDET)

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 can monitor the power supply 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

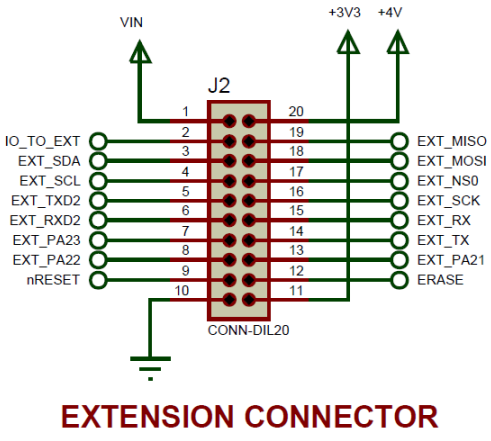
## Using EXT PIN

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.

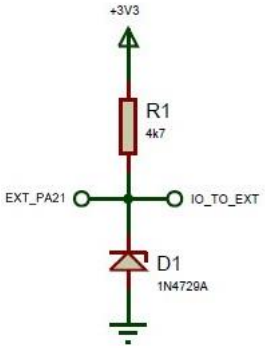
> **Please note:**
> When connecting any Internal I/O Pin to the IO_TO_EXT that those have 3.3V levels and are directly connected to the MCU. We recommend to protect the pin against overvoltage.

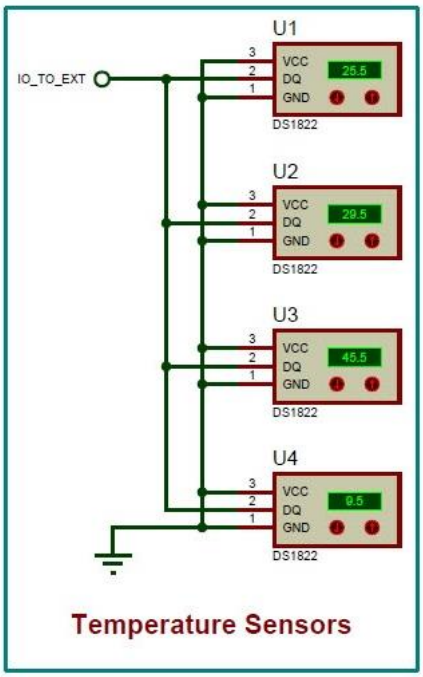
## Example usage - One Wire

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

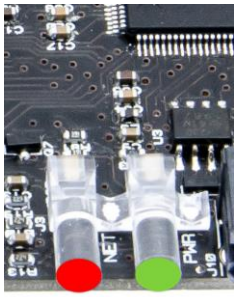


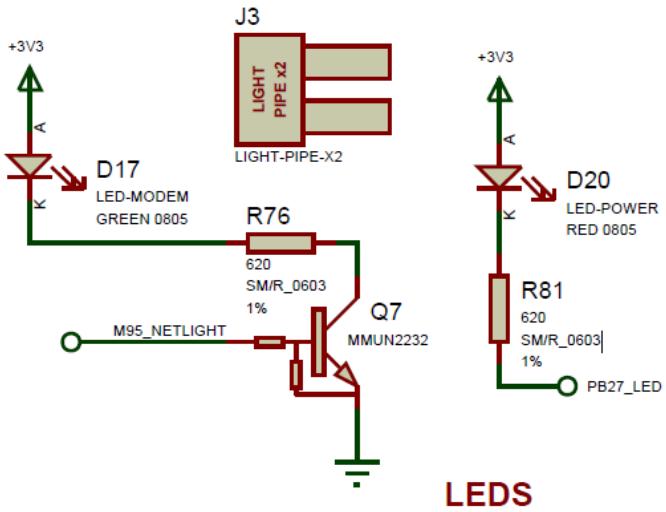




Outside Opentracker

# 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.




LEDS

## 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 Expansion connector is a 20-pin standard 2.54mm pitch female header.
It features the following interfaces and signals:

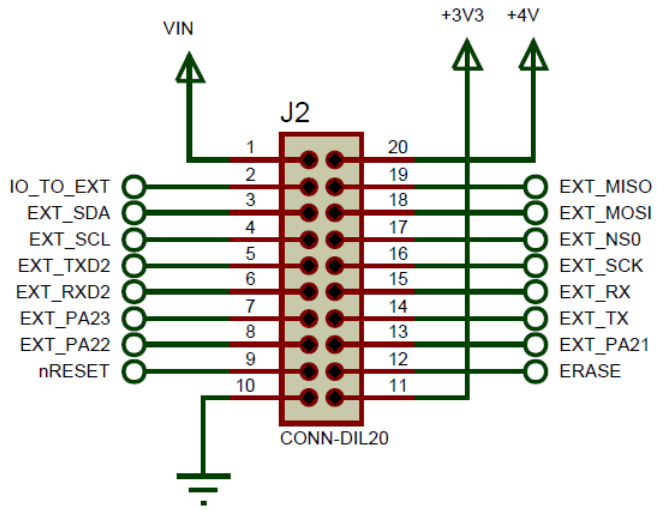
- 1 x SPI
- 2 x UART/USART
- 1 x I2C
- 2 x Analog Input (ADC)
- 2 x PWM (1 shared with UART TX)
- GPIO (shared with any other peripherals)
- RESET (open-drain input/output from main controller)
- ERASE (full chip erase request)
- 3.3V
- 4V (exact voltage may vary around 4.3V)
- VIN
- GND

**PLEASE NOTE:** **IMPORTANT NOTE!**

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

The internal I/O is reserved for custom Expansion boards done by the user to add functionalities for special applications.

## Pin assignment




EXTENSION CONNECTOR

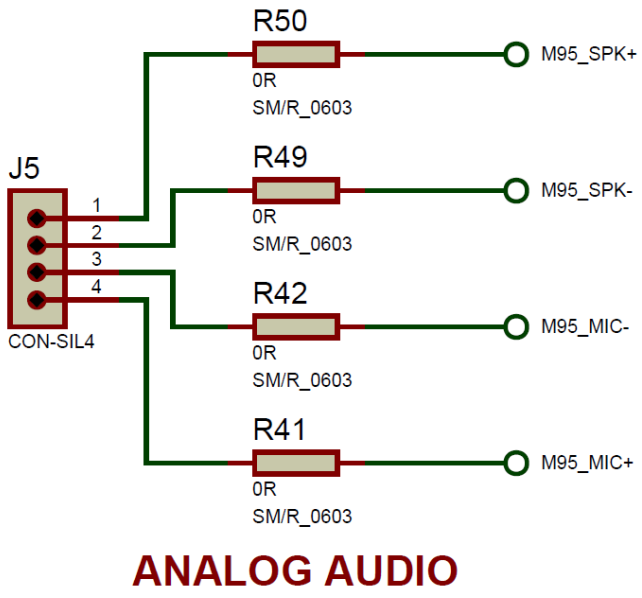
# Audio Interface

Since board revision 2.4 the OpenTracker has internal connections for the modem audio interface, enabling voice call features.

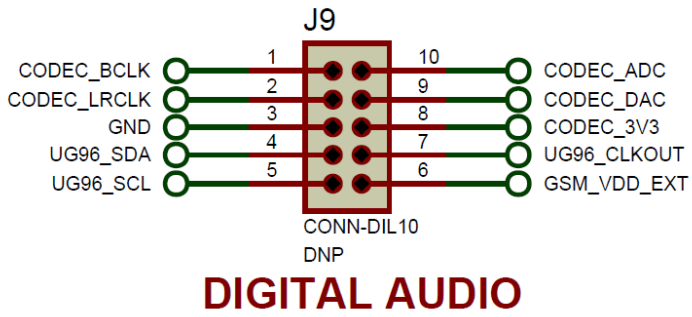
The 2G modem has analog audio support, with direct connection to an electrect condenser microphone for input and to an 8 Ohm speaker for output (max 500mW) in a “handsfree” configuration.

Board variant with 3G modem has an optional digital audio interface for connection to an external PCM audio codec.

## Connector schematic (2G modem only)



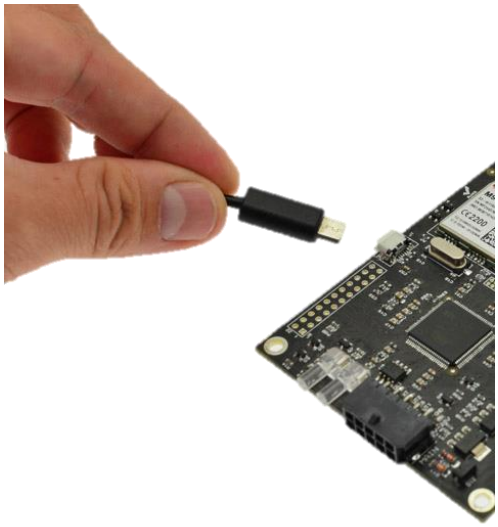

## Optional connector (3G modem only)



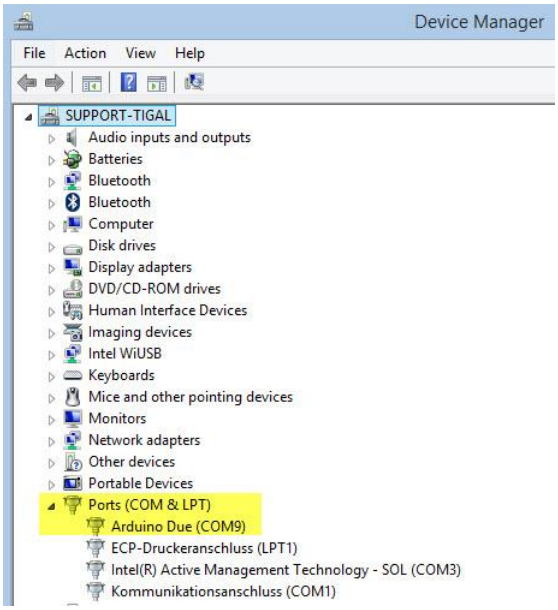
# USB Interface

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 1.6.7 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

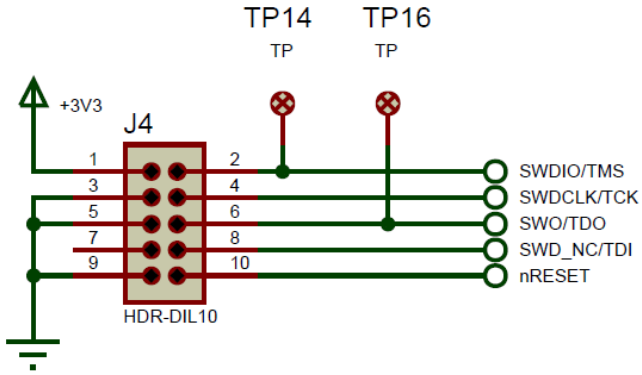
OpenTracker has got a JTAG interface which may be used for programming and debugging. The JTAG connector is a not populated 10-pin .050 inch pitch male header.

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

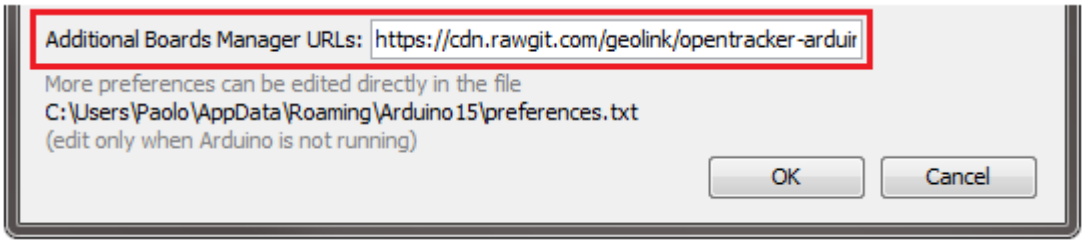
If you just want to use OpenTracker to Track your device you do not need to do these steps. It is all there and configured for this use. But if you want to take advantage of all IO and additional features follow the instructions below and you are ready to develop your own software with Arduino IDE.

### Adding OpenTracker 2 as Board to Arduino IDE

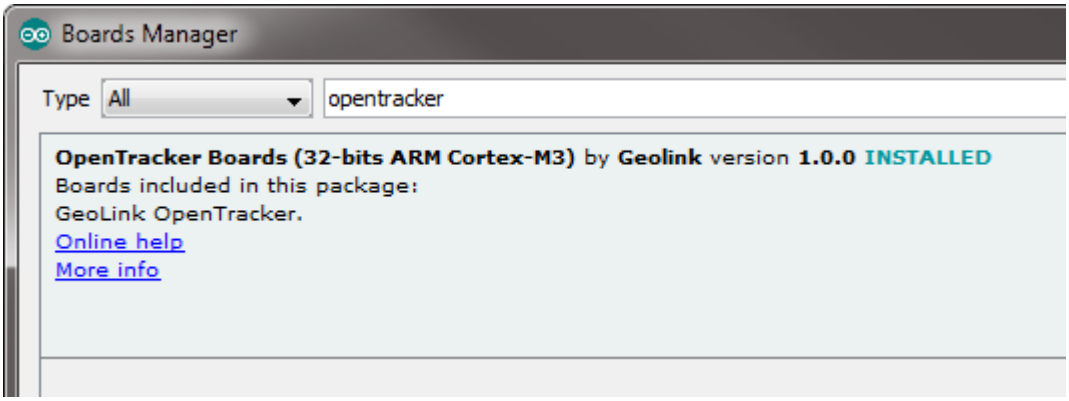
Make sure you have a recent Arduino IDE (version 1.6.7 or later).

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

https://raw.githubusercontent.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)
```

Please follow the below instructions to solve this.

### How to Reset a Soft bricked board

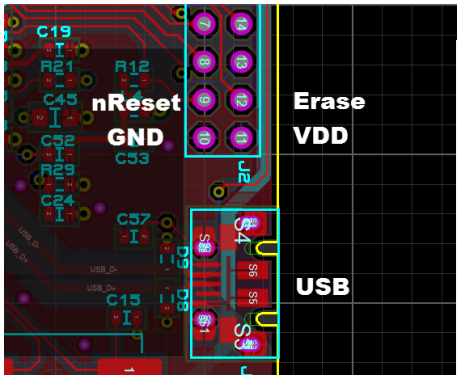
You can try to force a mass erase on the MCU Flash. This is accomplished by:

- 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.

# Electrical Characteristics

## Absolute Maximum Ratings

Stresses exceeding the maximum ratings may permanently damage the device or affect reliability. The device may not function or be operable outside the recommended operating conditions (see below).

| Parameter | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|
| Power Supply Voltage (VIN) | 8 | - | 32 | V |
| Power Supply Current | 0 | - | 1.8 | A |
| Analog Input Voltage | 0 | | 36 | V |
| Relay Output Current | 0 | | 1 | A |
| Operating Temperature | -40 | - | +85 | °C |

## Recommended Operating Conditions

| Parameter | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|
| Power Supply Voltage (VIN) | 9 | 12/24 | 30 | V |
| Power Supply Current | - | 0.25/0.12 | 1.65 | A |
| Internal Expansion I/O Voltage (VIO) | 0 | - | 3.3 | V |
| Ambient Temperature[26] | -35 | 25 | +80 | °C |

**Notes:**

1. Operating the modem outside the recommended temperature range may lead to RF performance issues.

# 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.