VeloTrack
Hardware & Software Manual

Joren Heit
jorenheit@gmail.com

1 Installation

1.1 Software Repository

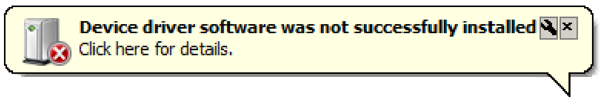
All software packages are hosted in the following GitHub repository:
`https://github.com/jorenheit/velotrack`
Software mentioned in this document can be downloaded here, or alternatively one can clone into this repository using Git (details on using Git are outside the scope of this document).

1.2 Windows

The application itself can be installed simply by extracting the ZIP-archive to an arbitrary directory and running the `velotrack.exe` executable. Drivers need to be installed in order for the interface to communicate to the Arduino hardware inside the module. These drivers are included in the Arduino software, which can be downloaded from `www.arduino.cc`. This will require installing the Arduino IDE, which might or might not be desired (see also Section 3). Therefore, standalone drivers have been included in the ZIP-archive, which can be installed according to the instructions below.

Installing Standalone Drivers

1. When the module is first plugged in, Windows should inform about a new device, for which (most probably) no drivers have been found (Figure 1).



Device driver software was not successfully installed
Click here for details.

Figure 1: Unless the Arduino drivers have already been installed or present on the system, Windows will not be able to sucessfully install the driver software for the VeloTrack module.

2. To install these drivers, open the **Device Manager** (e.g. by running `devmgmt.msc` from the Start- or Run-menu, or navigating to it through Computer-Properties). An unknown device should be listed, as seen in Figure 2. Right-click this device and choose “Update Driver Software...”.

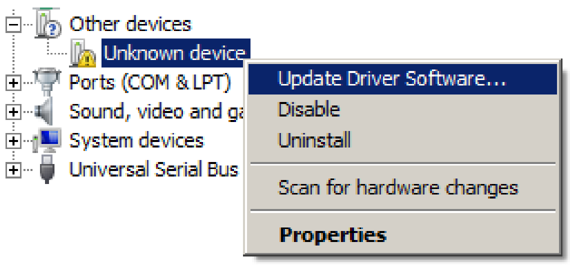
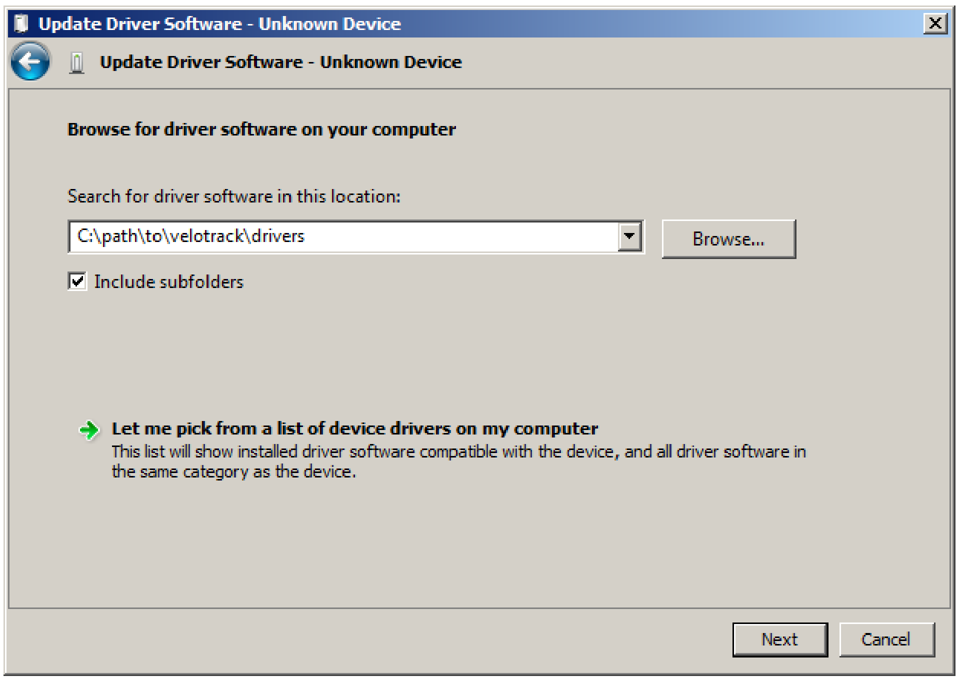


Figure 2: Right-click the unknown device listed in the Device Manager, and choose "Update Driver Software".

3. In the window that pops up, rather than choosing the default option which will search automatically for drivers, choose "**Browse my computer for driver software**". This will allow you to provide the drivers manually.
4. Enter the path to the *drivers* subdirectory, which is present in the folder where the VeloTrack ZIP-archive has been extracted (Figure 3), and click *Next*.



Update Driver Software - Unknown Device

Update Driver Software - Unknown Device

Browse for driver software on your computer

Search for driver software in this location:

C:\path\to\velotrack\drivers

Browse...

[✓] Include subfolders

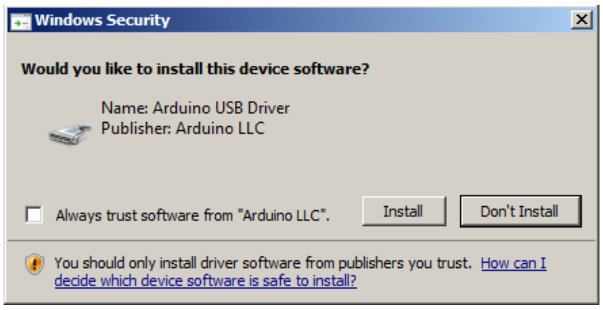
Let me pick from a list of device drivers on my computer

This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Next Cancel

Figure 3: Enter the path to the *drivers* folder, which is a subdirectory of the velotrack folder.

5. The wizard will look for drivers in the specified folder and start installing them. A window will pop-up to verify that these drivers should be installed (Figure 4). Click *Install* to confirm.


Windows Security

Would you like to install this device software?

Name: Arduino USB Driver
Publisher: Arduino LLC

Always trust software from "Arduino LLC".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Figure 4: Confirm the driver installation.

6. After the installation has finished successfully, the Arduino is listed in the Device Manager, including the (virtual) COM-port it is connected to (e.g. `COM3`, Figure 5). It is advised to keep a (mental) note of this port, as it will be used later in the VeloTrack software.

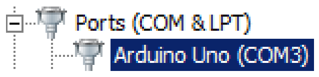


Figure 5: After installation, the Arduino is listed in the Device Manager.

1.3 Linux

There is both a 32-bit and 64-bit version of VeloTrack available for GNU/Linux systems as a precompiled binary. These can be installed simply by extracting the corresponding tarball and running the `velotrack.sh` shell-script. Running the executable (`velotrack`) directly will not work, unless the proper version of Qt5 is installed on the system. The wrapper-script will set up the shared-library path such that the dynamic libraries included in the `lib/` subdirectory will be loaded.

It's not necessary to install any drivers for the Arduino.

1.4 Compiling from Source

To compile VeloTrack on any system supported by the Qt framework, the project file `velotrack.pro` may be used. Loading this file in the Qt Creator IDE will allow you to compile the project, provided that the `QSerialPort` has been installed on the target system (e.g. on Debian/Ubuntu systems, this can be done by installing the `libqt5serialport5-dev` package, available from the repositories). Alternatively, from the commandline (assuming a Unix-like shell), run:

```
qmake -config release # this will produce a Makefile
make # compile and link the binary
./velotrack # run the program
```

2 Using the `VeloTrack` Interface

2.1 Overview

After opening the executable (`velotrack.exe` on Windows, `velotrack.sh` on Linux), you will be greeted by a splash screen (click to hide), and the main-window (Figure 6). The interface is quite minimal and contains the following elements:

1. *Port-selection list*: select the port to which the module is connected. This port-number corresponds to the one that was listed in the Device Manager (Figure 5).
2. *Connect/Disconnect*: A connection between the PC and the VeloTrack module at the selected port will be attempted when this button is clicked. When a connection is active, this button serves to terminate this connection.
3. *Scan*: Clicking this button will refresh the ports listed in the dropdown menu. When the USB cable is plugged in or unplugged, this will not be automatically detected. The scan-button can then be used to update the list.
4. *Start*: The big square button on the left, containing the green triangular symbol pointing to the right, is used to start a new measurement.
5. *Download*: The big square button on the right, containing the blue triangular sumbol pointing downward, is used to download measured data from the module to the computer. After downloading, you will be prompted to save the downloaded data to a file, securing its contents. Saving the content to disk can be delayed to, or redone at any later time by using the File - Save (Ctrl+S) action.
6. *auto-download*: When this box is checked, the data will be downloaded automatically when a measurement has been finished.
7. *Plot-area*: A large area of the window is dedicated to a graph-box, that shows a preview of the data after it has been downloaded to the computer. The data shown in this area always corresponds to the data currently present on the computer, which can be exported to disk using the File - Save (Ctrl+S) action.
8. *Status*: In a small region below the plot-area, the current status of the program is shown. Some errors will also be presented in this region.

2.2 Performing a Measurement

When the device has been connected using the *Connect* button, a succesful connection is indicated in the status-area ("Connected"). Also, while connecting, the red indication-LED will blink once. A measurement can be started now by clicking the *Start*-button; the yellow LED on the module lights up. The module has been configured to wait until

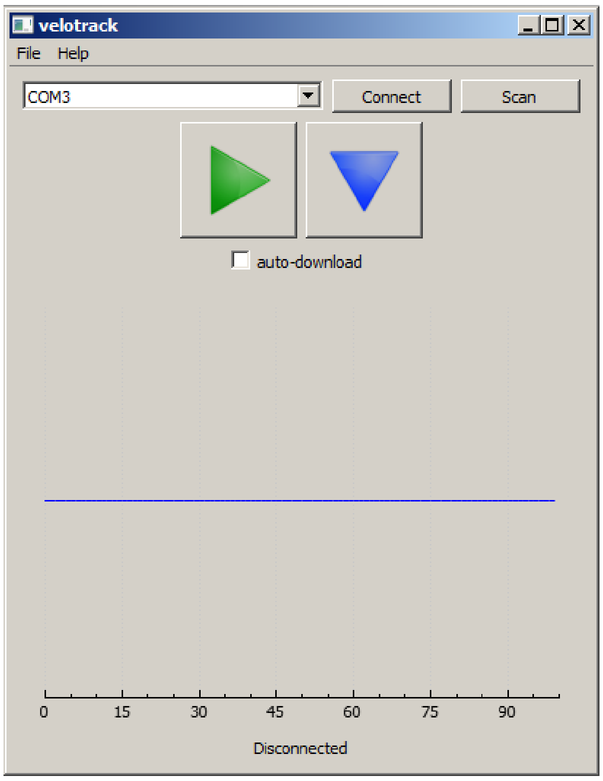


Figure 6: The VeloTrack main-window.

a certain displacement-threshold has been exceeded (this threshold can be reconfigured by reprogramming the module, as described in Section 3). As soon as the threshold is exceeded, the green light will turn on to indicate that the device is now measuring. The duration of the measurement is fixed and the light will turn off as soon as the measurement is finished.

When the measurement is finished, the data can be downloaded from the device. With the *auto-download* option checked, this will be done automatically; otherwise, hit the *Download*-button. After downloading, a dialog is prompted allowing you to save the data to a plaintext (`.txt`) file, and the results will be displayed in the graph-area. To save this data at a later time, click *Save* in the *File*-menu (or press `Ctrl+S`).

2.3 Warning

- Starting a measurement will erase the data already present in the buffer *of the module*. Make sure to download the data before starting a new measurement.
- Downloading the data will overwrite the data already present in the buffer *on the computer*. Make sure to save the data to a file before commencing a new download.

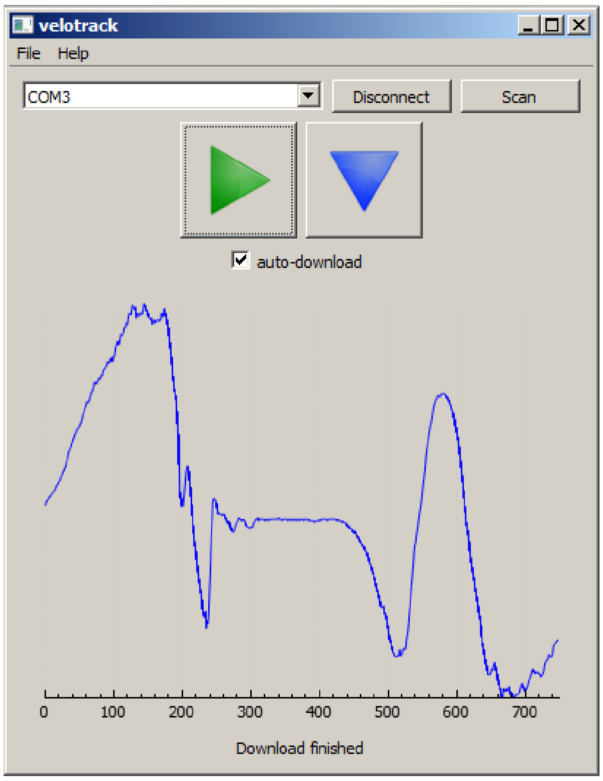


Figure 7: The VeloTrack main-window after a successful measurement and download.

3 Modifying the Firmware

Internally, the module contains an Arduino Mega 2560 development board. This can be reprogrammed through the USB-port, most easily using the official Arduino IDE (`www.arduino.cc`). However, after installing, a few additional steps have to be taken to enable C++11 support in the Arduino IDE (Integrated Development Environment), which is necessary to compile the firmware:

3.1 C++11 Compatibility

The compiler behind the Arduino IDE should be passed a flag that tells it to enable C++11 code. Only relatively new compilers support these features, so first make sure that version `>=1.5.7` of the Arduino IDE is installed. If so, locate the `platform.txt` file, which can probably be found at

```
{path to Arduino}/hardware/arduino/avr/platform.txt
```

In this file, edit the `compiler.cpp.flags` line by adding `--std=c++11` to it. For example (without the line-break):

```
compiler.cpp.flags=--std=c++11 -c -g -Os -w
                   -fno-exceptions -ffunction-sections -fdata-sections
```

After restarting the IDE, it should now be able to compile C++11 programs.

3.2 Changing Measurement Parameters

The firmware contains a set of 3 parameters that are most likely to be changed:

1. `RECORD_TIME` in milliseconds
2. `RECORD_INTERVAL` in microseconds
3. `RECORD_DELAY` in steps (1 revolution consists of 2000 steps)

These parameters are defined near the top of the `driver.ino` source-file, which can be found in the `src/driver` folder (not to be confused with the `velotrack/drivers` folder, which contains the Arduino drivers for Windows described in Section 1.2).

`RECORD_TIME` defines the duration of a measurement in milliseconds. During this time, a measurement will be recorded every `RECORD_INTERVAL` microseconds. Therefore, the total number of measurements recorded is equal to:

$$N_{rec} = \frac{1000 \times \texttt{RECORD\_TIME}}{\texttt{RECORD\_INTERVAL}}$$

The value of N_{rec} is important due to the memory limitations of the Mega 2560 chip, which has only 8kB of SRAM available to store measurements. At 2 bytes per record, a total in the order of 4000 measurements can be stored on the device.

$$N_{bytes} = \frac{N_{rec}}{2}$$

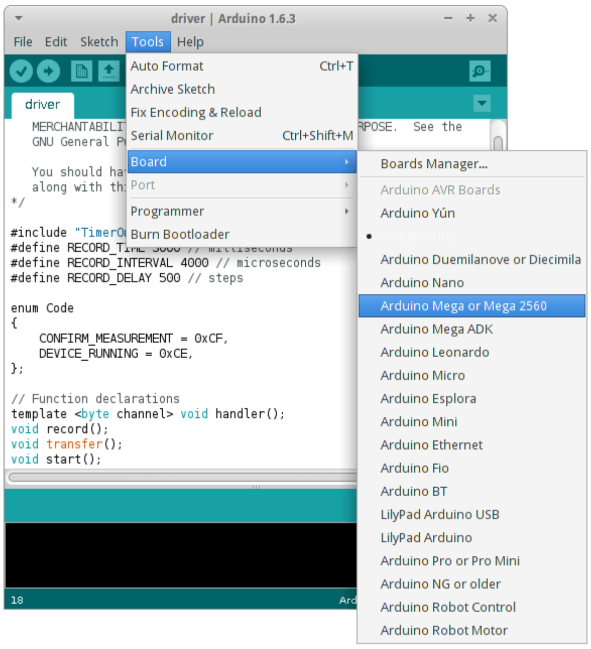
The Arduino environment will warn when the program requires more memory than available.

`RECORD_DELAY` specifies the number of steps that has to be exceeded before the measurement is actually started. The datasheet of the HEDS-5700 rotary encoder [1] specifies a total of 500 cycles per revolution of the disk. Every cycle consists of 4 detectable *steps* (changes of state), adding up to a total of 2000 steps per revolution.

After changing these values accordingly, make sure the Arduino IDE is configured for the Arduino Mega 2560 board (Figure 8), and upload the modified firmware using the Upload-button (Figure 9).

State	S1	S2	S3	S4
Channel A	1	1	0	0
Channel B	0	1	1	0

Table 1: The different states of the rotary encoder, as defined in [1] and Figure 10.



```
   MERCHANTABILIT                               RPOSE.  See the
   GNU General P

   You should ha
   along with th
*/

#include "TimerO
#define RECORD_TI
#define RECORD_INTERVAL 4000 // microseconds
#define RECORD_DELAY 500 // steps

enum Code
{
    CONFIRM_MEASUREMENT = 0xCF,
    DEVICE_RUNNING = 0xCE,
};

// Function declarations
template <byte channel> void handler();
void record();
void transfer();
void start();
```

Figure 8: Before compiling and uploading the modified firmware, make sure the IDE knows for which type of Arduino it is intended by selecting “Arduino Mega or Mega 2560” in the Tools - Board menu.

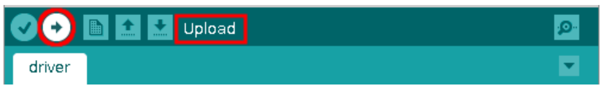


Figure 9: Use the *Upload* button to compile and upload the new firmware to the module. The IDE will report about the memory usage in the black console area at the bottom of the window. Any messages warning you abou stability-issues due to high memory percentages may be ignored.

4 Signal and Connector

The rotary encoder emits two signals, on channels A and B respectively. Figure 10 (taken directly from [1]) shows these signals with respect to eachother. The phase-difference between channel A and B allows for the detection of 4 states, denoted S1-S4 in Figure 10 (Table 1). The direction of rotation can be deduced from the transition between states, as illustrated in Table 2. The transition-table is then used to determine positive and

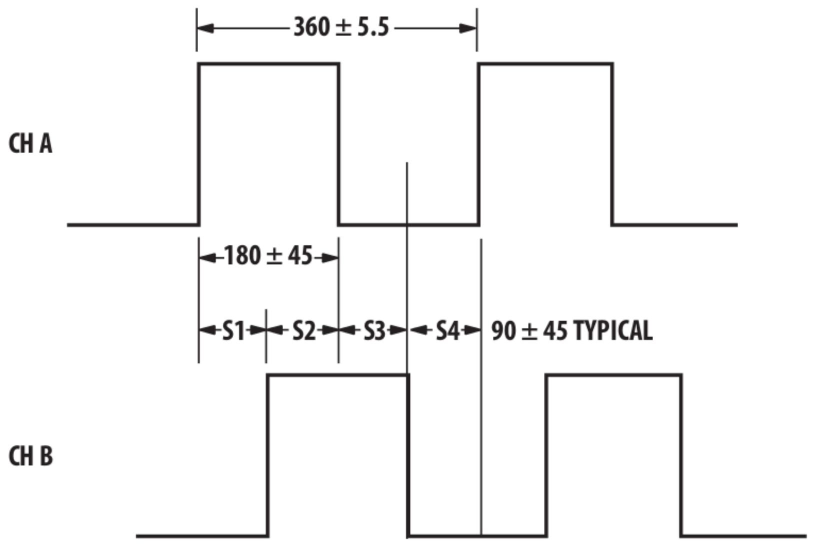


Figure 10: Signals from the rotary encoder

	S1	S2	S3	S4
S1	0	+1	±2	−1
S2	−1	0	+1	±2
S3	±2	−1	0	+1
S4	+1	±2	−1	0

Table 2: Transition-table to map state-transitions to displacement.

negative displacements. The firmware in the module (`driver.ino`) ignores ambiguous transitions (e.g. S1 to S3), i.e. in the table that is implemented in the firmware, these entries are set to 0.

The pin-layout of both the connection to the rotary encoder and the VeloTrack module is shown in Figure 11.

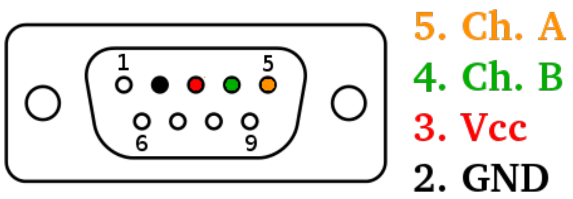


Figure 11: Pin-layout of the connector

5 Contact

For questions and support, please feel free to contact me at `jorenheit@gmail.com`.

References

[1] HEDS 5700 Datasheet, `http://www.avagotech.com/docs/AV02-3575EN`