GreenInvents USB 2.0 current sensor

This manual contains all information necessary to perform accurate, scientifically usable current measurements of any USB-powered device using the GreenInvents USB 2.0 current sensor. This document is applicable to:

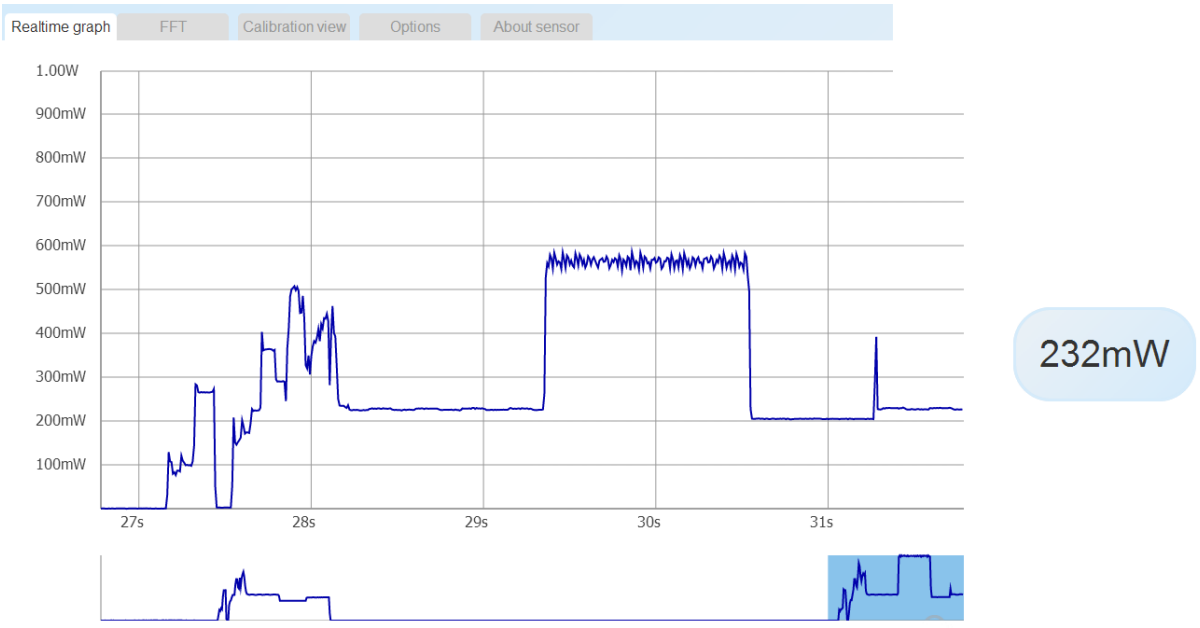
Model number	Name	Version/Revision
GI-USB1	GreenInvents USB 2.0 current sensor version 1	Revision 3 Revision 4 Revision 5
	GI-Proxy-v2	Build 115
	HTML5 application	Canvas Build 100 (DB version 2)

1. Product overview

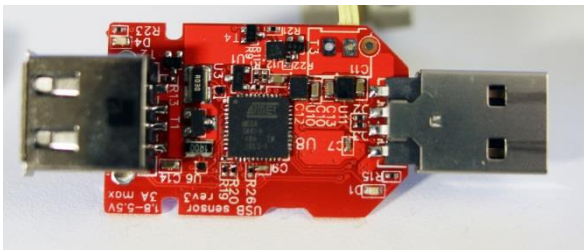
GI-USB1 is a USB-stick-sized, highly accurate USB power consumption measurement device. GI-USB1 is applicable to power measurement of both general USB devices as well as mobile devices. This enables measurement of the power consumption of mobile applications and the power impact of changes in settings in great detail. Furthermore, it may be used by developers and users of USB devices to ascertain and optimize the power consumption of their USB devices and to test features like USB selective suspend. Consumers may use the device to measure the impact of USB devices on the battery life of their laptops.

Typical application example

Besides the USB-stick, the device comes with an easy-to-use browser-based interface software which allows at-a-glance power consumption information as well as in-depth realtime waveforms and analysis.



2. GreenInvents USB 2.0 Scientific Current sensor GI-USB1



GI-USB1 is GreenInvents B.V.'s first USB-specific current sensing solution. It enables sensing power consumption of any Universal Serial Bus (USB) device with unprecedented accuracy and temporal resolution.

- Current sense range up to 2.5A
- Selectable 82mA max current range, for sensing low-power standby modes
- Current resolution down to 2.5µA in high-resolution mode
- 12-bit ADC
- Selectable decimation+averaging as follows: /1, /4, /16, /64, /256
- Higher levels of decimation available with truncation
- Up to 125kS/s sample rate
- No aliasing artefacts – ADC always runs at 125kS/s internally
- 20kHz analog bandwidth
- Basic accuracy ±0.75% ±4mA (long term)
- Better accuracy attainable via Calibration Mode (CM)
- Calibration values stored in EEPROM
- Separate battery/Power supply connection to offload PC USB port for high-current measurements
- Supports Battery Charging 1.1 (BC1.1) specification for analyzing devices charging over a Charging Downstream Port (CDP) with up to 1.5A
- Power supply connection supports 2.7-5.5V input voltage (board still requires 4.5-5.5V from upstream USB port to function)
- 0-85˚C temperature range
- Low-power design leaves maximum power output available to Device Under Test (DUT)
- Data transfer over easy-to-use virtual serial port (USB-CDC)
- Firmware updates via Driverless Mass Storage Bootloader (DMSB) – just drag and drop firmware files from our website into the FIRMWARE drive and the device will auto-update
- Optional calibration (±0.1% 1-month, ±0.25% 1-year) with certificate
- Includes standalone browser-based HTML5 interface with full functionality
- Can be used remotely over Ethernet with the HTML5 application

3. Order codes and compatibility

Order code	Description
GI-USB1	Base sensor unit. Comes with HTML5 application, quick start guide, digital manual and digital specification sheet
GI-USB1-Cal	Calibration service. Comes with a full characterization of the unit over the entire operating temperature and frequency range. Calibration results are included in compatible forms for the application and as a printed table of values.
GI-USB1-Lead	Lead and cable set for GI-USB1. Comes with a high-current USB-A to micro-usb cable, USB-A to USB-B cable, Banana/alligator clip to battery input connector cables

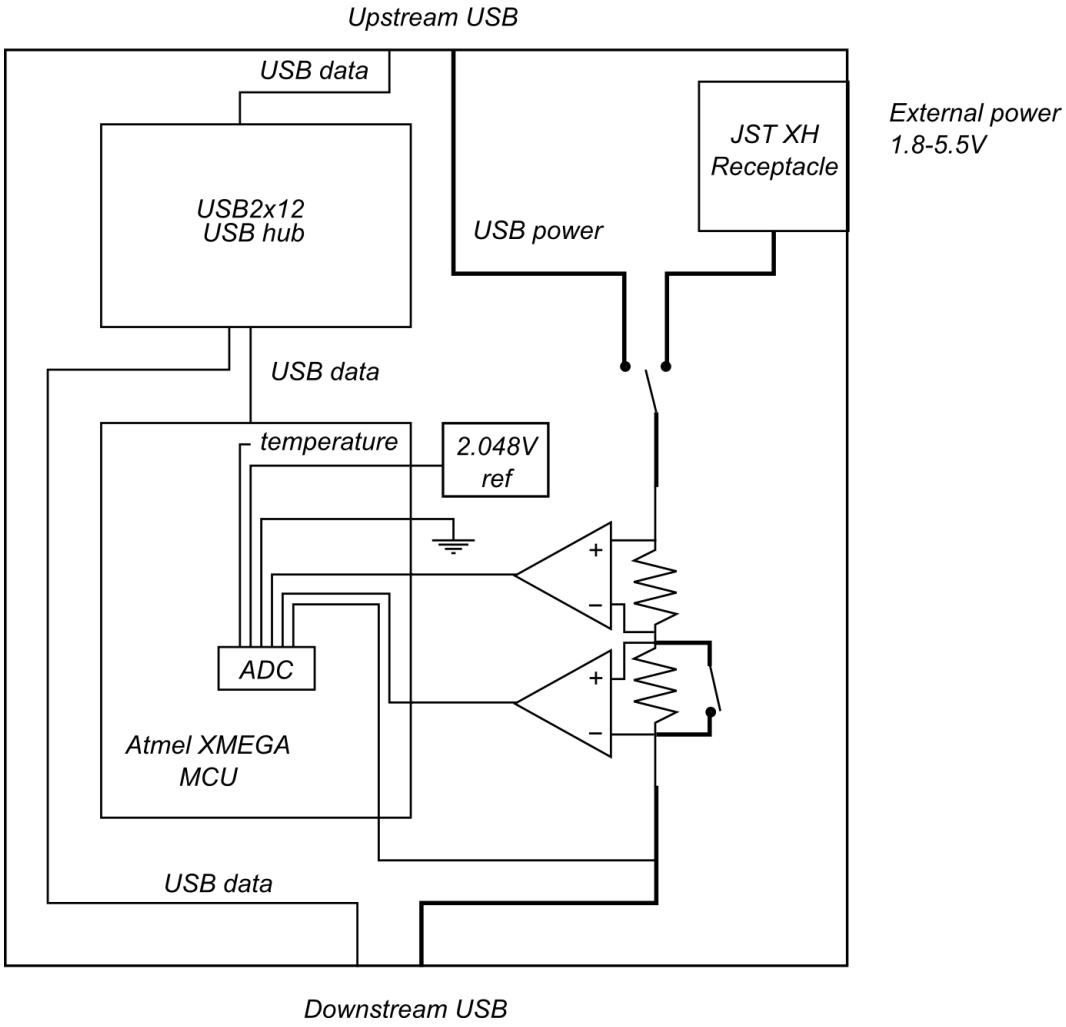
3.1 Compatibility notes

Hub controllers	Intel ICH8
	Texas Instruments TUSB3410*

* GI-USB1 is limited to USB 1.1 and full-speed (12Mbps) and USB 2.0 hi-speed (480Mbps) downstream rates

Operating systems (HTML5 application)	Microsoft Windows 7
	Microsoft Windows 8/8.1
	Microsoft Windows 10

4. System overview



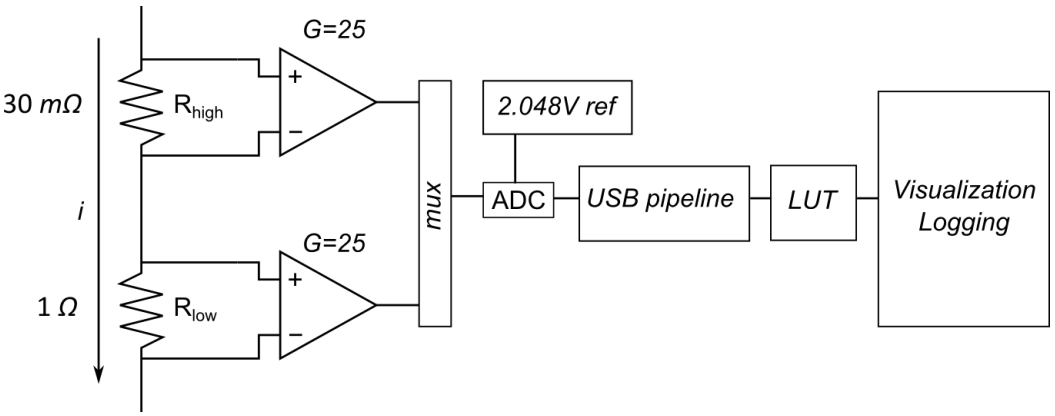
GI-USB1 is designed to easily measure power draw of any USB-powered or battery-powered device under test (DUT) . GI-USB1 presents itself to the host computer as a USB hub, allowing for both the DUT and on-board microcontroller to enumerate themselves as USB devices. USB power flows through two current shunt resistors with associated amplifiers, which generate a signal proportional to the current flowing into the DUT. In order to reduce voltage drop for high-current applications, the high-value current shunt may be shorted in software. Output voltage, as well as temperature and ground offset, are measured as well. This information is then transported over USB and may be visualized in the supplied HTML5 or Matlab applications, or in a custom application written on the basis of the supplied protocol information.

5. Basic theory of digital measurements

In this chapter, the basic theory and principles behind digital measurements of analog signals as applicable to GI-USB1 will be discussed. This is important for the interpretation and post-processing of measurements, as an incomplete understanding of the measurement chain may lead to incorrect assumptions and interpretations.

5.1 The measurement chain

Refer to Figure X for a complete schematic representation of the functional blocks in the measurement chain.



The measurement chain starts at the main current path towards the DUT. The current is passed through a low- and high-value measurement resistor shunt resistor in series. For high-current applications, the high-value (1 Ω) resistor may be shorted using a very low resistance p-channel MOSFET in order to avoid high supply voltage drops in the measurement chain. The current going through the measurement resistor develops a voltage proportional to the current and resistor value (basic Ω's law). This voltage is amplified using an INA126A4 high-side current shunt amplifier. The resulting voltage is fed into the XMEGA ADC, where it is converted to a digital code, piped directly to the computer application over USB and optionally Websockets. Finally, the data is filtered through a lookup table to correct for (pre-determined or user-measured) measurement offsets.

In an ideal world, this measurement chain would be perfect and nothing needs to be done – not even the lookup table part – to get a perfectly accurate and precise power measurement. Unfortunately there are a lot of small, though in critical applications measurable errors that need to be accounted for.

The next chapter details all the significant sources of measurement errors that can be expected when operating this instrument. The HTML5 sensor interface application is designed to correct for all these errors to such an extent that operation within the specified accuracy is guaranteed. The information in the following chapters is not necessary reading material for normal operation. However, if the user intends to design a proprietary signal processing application, it is highly advised to take all factors in the next chapters into careful consideration.

5.2 Measurement errors

Measurement errors roughly fall into three categories:

1. Physical measurement errors
2. Time-domain errors
3. Digital domain errors

5.2.1 Physical measurement errors

5.2.1.1 Shunt resistor value offset

The value of the shunt resistors are not exactly 30mΩ and 1 Ω; they may have an offset of up to 1% in either direction (i.e 29.7-30.3 mΩ and 0.99 to 1.01 Ω). The 1 Ω resistor is a thick film type, which is almost without exception cut from a by design within-specification thick film resistive wafer with a controlled surface impedance. This means that if one would make a histogram of a large population of these resistors, most resistors would be exactly 1.00 Ω, a small proportion would be 0.2% off, even less resistors would be 0.4% off… etc… and almost no resistors would be 0.8% off. So, if we actually try this with a few resistors on the production tape we end up with [Figure 1: Resistor offset error distribution]:

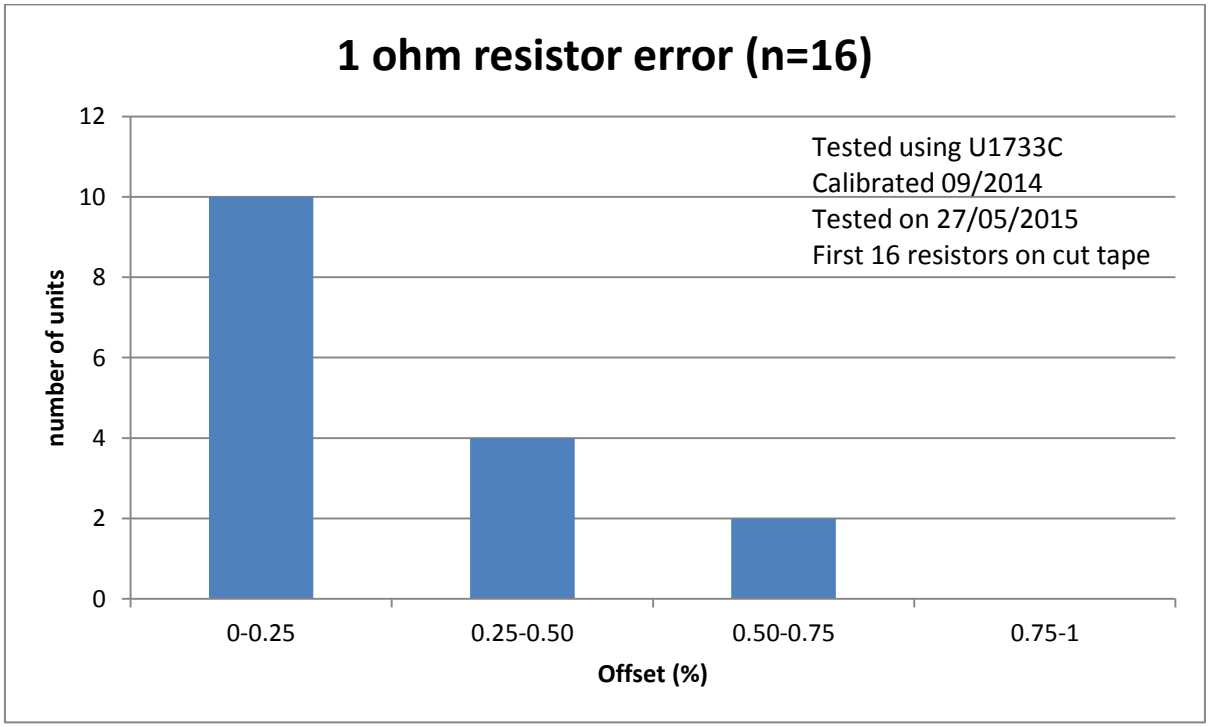


Figure 1: Resistor offset error distribution

Even with only a sample size of 16, we already see a typical bell curve (Gaussian distribution) that we expect.

The 30mΩ resistor is made using metal film, which generally does not work in the same way. Because of manufacturing details that are beyond the scope of this text, as well as the low tolerance for error with such low absolute resistances, these resistors are either laser trimmed or binned, whichever is cheaper

for the manufacturer. If a similar test would be done with a suitably large population of these resistors, one would find almost equal amounts of resistors in every error bin.

The relevance of these distributions will become clear in chapter 5.2.3

5.2.1.2 Shunt resistor temperature and current dependence

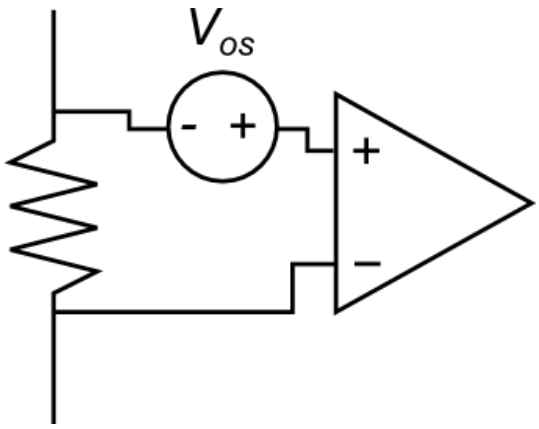
The materials used for resistors are chosen to be as stable as possible over the applicable temperature range, but no material is perfectly stable. Small variations of resistance over temperature will occur. The temperature dependence of both thick film and metal film resistors is non-linear.

There are two distinct sources of temperature variation in a shunt resistor application: ambient temperature and power dissipation in the resistors. Ambient temperature variations heat up or cool down the entire board and will cause isotropic temperature variations in the resistors. Heating due to current flowing through the resistor will cause hot spots and consequently has a larger relative impact on resistance stability. Typically, thick film resistors will heat up most in the exact middle spot of the resistor and almost not near the edges whereas metal film resistors will have a more gradual temperature profile over the package (as a consequence of the roughly 2 orders of magnitude better thermal conductivity of the resistive material). Therefore, GI-USB1 has very conservative thermal dissipation in the sense resistors, especially in the 1 Ω resistor (6mW max.), and thermally enhanced resistor packages have been chosen.

The temperature coefficient (change of resistance with temperature) of both sense resistors is 100ppm/K.

5.2.1.3 Amplifier input offset voltage

By far the biggest contributor to measurement error in the current measurement chain is the amplifier input-referred offset voltage. This is a combination of various physical phenomena that manifests itself as a small additional voltage in series with the actual voltage present between the non-inverting and inverting inputs of the amplifier.



This offset voltage usually has some fixed base value at 25C, with small amounts of temperature and common mode voltage dependence. Note that even though these values are input-referred, it is a composite value that combines both actual input offsets, output offsets and feedback offsets. This also

means that the offset voltage varies quite nonlinearly over temperature and especially common mode voltage.

The input offset voltage also limits the minimum measurable current. INA216A1 has a typical input offset voltage of 30 µV at 25C, which means it may effectively sense 0V when 30 µA actually flows through the 1 Ω current shunt, thus erroneously reporting no current flow.

5.2.1.4 ADC errors

The ADC used is the internal 12-bit SAR ADC in the XMEGA microcontroller. Successive Approximation Register ADCs (SAR ADCs) work by quickly sampling and holding the input voltage, then doing a binary search until the corresponding digital representation of the value has been found. The major advantage of SAR ADCs is that a well-designed implementation always has no missing codes, that is, if the input is varied monotonely up or down, the binary representation of the result will also monotonically go up and down. A disadvantage of SAR ADCs is its inherent reliance on a DAC and analog comparator, which are limited in their bandwidth and precision and subsequently limit the practical bandwidth and bit depth of this particular low-power implementation to about 12 bits at 1MSPS or 16 bits at 1kSps.

ADC errors are roughly divided into:

1. DNL
2. INL
3. Gain error
4. Offset error
5. Quantization error
6. Clipping
7. Noise

Please refer to [Figure 2: INL, DNL, Offset, Gain error visualized] for a visual represenation of DNL, INL, Gain error and offset error.

DNL, or differential nonlinearity, is the worst-case difference in the step size of a code; i.e. on GI-USB1, ideally a 1 LSB increase in the code corresponds to a voltage difference of 0.5mV. In order guarantee to not have missing codes, DNL must be less than or equal to 0.5LSB.

INL the worst-case difference between the ideal and actual code corresponding to an analog value.

Offset an gain error. In the explanation on DNL and INL given above, I used the ideal transfer function from analog to digital as a reference point for DNL and INL, but in reality this is often not the case. Instead, a best fit linearized transfer function is used which may not coincide with the ideal transfer function. It may be slightly offset but still parallel – this is called the offset error. It may not be parallel; this is called a gain error. Offset errors can be measured either as a voltage or code offset; care must be taken to recognize that these may have a negative sign. In the image, it is shown that a positive voltage offset corresponds to a negative code offset.

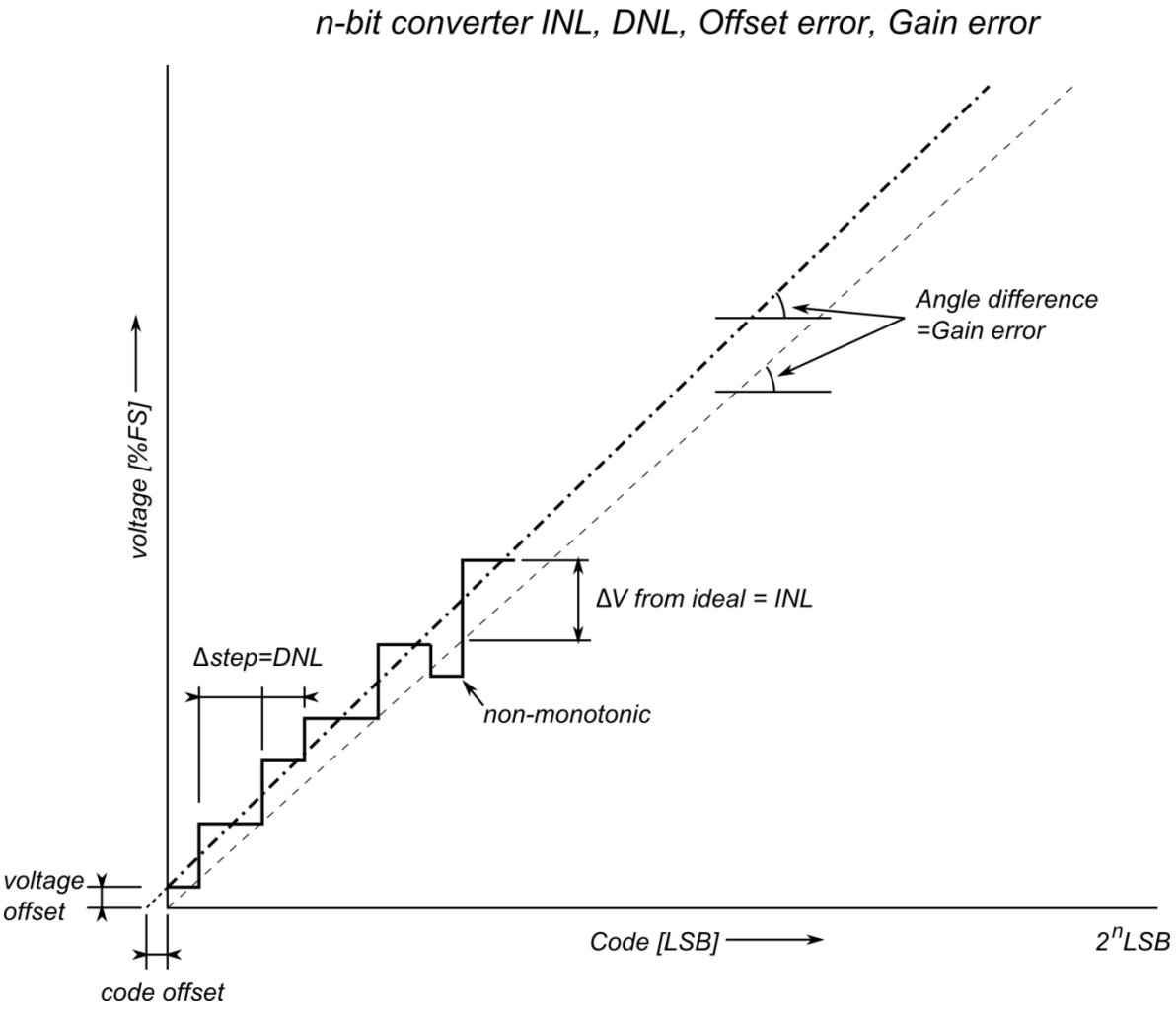


Figure 2: INL, DNL, Offset, Gain error visualized

Quantization. Digital integer representations of an analog signal necessarily have a stepwise nature, there is no in-between values in digital representation. This means that quantization - the act of digitizing an analog value - causes a little bit of error in representing those in-between values. In an ideal ADC, the quantization error is always ½ LSB at the most. Sometimes, once in a while slightly larger or smaller steps are implemented by the ADC that cause larger errors.

Clipping. When the input voltage exceeds or undershoots the range that can be represented by the ADC implementation, generally the digital representation gets clipped and contains no information about the actual analog representation anymore.

Noise is both a source of error and an important tool in ADCs. Too much noise may reduce the effective resolution, but no noise at all limits the ability for ADCs to use resolution-enhancing techniques like oversampling. If a signal with an analog value in between two quantization levels exists on the input, an ADC with absolutely no noise will always represent this analog value by the closest quantization level. If a little bit of noise (ideally about 1LSB) is added to the analog value, the representation will switch

between two adjacent digital representations and a number of subsequent data numbers may be averaged to find the actual analog value, thus increasing the effective resolution.

GI-USB1 inherently relies on oversampling to attain higher resolution results in a dedicated high-resolution mode. Mathematically, one can increase the resolution of an ADC by *n* bits through averaging of 2^{2n} successive samples. GI-USB1 can oversample up to 2^8=256, i.e. up to an effective resolution of 16 bits. This obviously reduces the effective sample rate by a factor of 256, reducing the native samplerate of 125kS/s to 488S/s.

Typical ADC INL, DNL, maximum effects of gain and offset error for the XMEGA ADC are 3.5LSB. The noise has an RMS value of 2.2LSB and a peak-to-peak value of ± 6LSB

5.2.1.5 Reference errors

The ADC attains its measurement range from the external voltage reference. Its 12-bit measurement range corresponds to input voltages between 0 and 2.048V, i.e. 1LSB = 0.5mV. The reference is an LM4040 with 0.5% basic accuracy and 100ppm/K temperature coefficient.

5.2.2 Time-domain errors

5.2.2.1 Attenuation at higher frequencies

When measuring high frequency inputs, it is important to observe the effects that the analog input circuitry has on the measurement. This will avoid (large) errors when interpreting high frequency components in the signal.

Let us consider the analog input circuitry on GI-USB1:

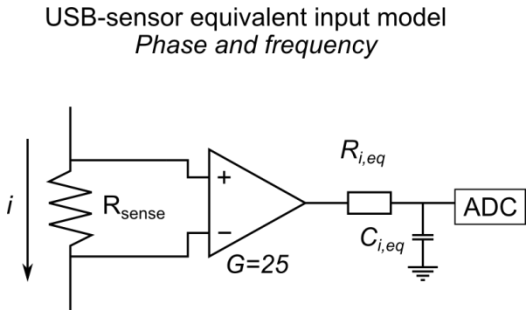


Figure 3: USB sensor equivalent input circuitry

The current sense resistor (R_{sense}) transforms the output current into a small voltage which is amplified 25 times through a current sense amplifier with a -3dB bandwidth of 20kHz. This is then fed into the ADC circuitry, which has an equivalent input resistance ($R_{i,eq}$) of 4.0 kΩ (typ) and equivalent input capacitance ($C_{i,eq}$) of 4.4pF. The output impedance of the amplifier is 42 Ωs (typ). This means that for fast-changing signals, signals at high frequencies will experience attenuation and phase shift on top of the essentially constant (for the frequencies that this design allows) propagation delay of the analog measurement chain (see chapter X). Specifically, the signal will experience -3dB (50% attenuation) at:

$$f = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 4.0\times10^{3} \cdot 4.4\times10^{-12}} = 9.0 \times 10^{6} Hz = 9.0 MHz \qquad \text{(Equation 1)}$$

Because the specified bandwidth for the amplifier is 20kHz (-3dB) and thus much lower than the input bandwidth of the ADC, this attenuating effect can be neglected as the amplifier will have a much larger effect on attenuation.

The next issue is the frequency at which attenuation is measurable. The amplifier is specified as having a -3dB point of 20kHz (i.e. an RC constant of 7.96×10^{-6}), but it will already start attenuating the signal at much lower frequencies. The expression for the gain of an RC filter as a function of frequency is:

$$G(\omega) = \frac{1}{\sqrt{1+(\omega RC)^2}} \qquad \text{(Equation 2)}$$

Where $\omega = 2\pi f$. The ADC in GI-USB2 is 12-bit, which means that a gain of $1 - \frac{1}{2^{12}}$ is theoretically measurable (i.e. 1LSB error). Substituting this into Equation 2 yields:

$$1 - \frac{1}{2^{12}} = \frac{1}{\sqrt{1+(\omega \cdot RC)^2}} \qquad \text{(Equation 3)}$$

Solving for ω yields $\omega = 2776.52$, so $f = 442\ Hz$. Also note that this is based on the assumption of a sinusoidal input signal. Any signal under 442 Hz will be attenuated less than the theoretical minimum error of the measurement chain. Any signal above that value will have a correct average value, but will be distorted by attenuation. Note that for non-sinusoidal signals, this information can be used to calculate the minimum rise time that will not yield measurement errors.

Aside from attenuation at higher frequencies, the signal will also have a phase error dependent on frequency. If phase accuracy is important to your measurements, make sure to apply appropriate back-transformations on your measurement data to compensate for phase errors.

5.2.2.2 Simultaneity and sampling jitter

SAR ADCs work by sampling and holding the analog input voltage in a very short amount of time, and then converting this input voltage into a digital representation over the course of at least an order of magnitude more time. After that, again about an order of magnitude more time passes before the result is transferred to the end user application.

The variation in time between conversions by the ADC is called jitter. Jitter in a microcontroller is almost exclusively caused by jitter in the clock generation subcircuitry. For XMEGA, typical jitter as configured in GI-USB2 is about 2µs. Another possible cause of sampling jitter is firmware implementation, specifically the firmware that triggers ADC conversions. In GI-USB1, extensive care is taken to make ADC sampling completely jitter-free.

Because the USB 2.0 implementation in GI-USB1 only allows for 1-ms frames, data will arrive at least about 1200µs after they were first sampled. Usually this is not a problem as display and logging software introduces much larger delays, but if an application is desired where GI-USB1 is used as a triggering device, this information may be valuable to assessing the performance.

GI-USB1 samples various different signals, including ground pin voltage, power source input voltage, temperature and reference voltage. Because of the nature of the XMEGA ADC, not all signals are

captured exactly simultaneously. The maximum time between two samples that appear in the same timecoded packet is 5µs. This means GI-USB1 can introduce quite significant instantaneous power errors (when multiplying the voltage and current signals) if the power source has a significant AC component at high frequency.

5.2.2.3 Aliasing

The analog measurement chain in GI-USB1 is fixed, which means that the analog input bandwidth is always about 20kHz (-3dB). GI-USB1 thus always measures the input signal with the maximum selectable sample rate (125kS/s) in order to avoid aliasing errors. If lower sampling rates are selected, multiple subsequent measurements are averaged as discussed in subchapter 5.2.1.4.

5.2.3 Digital domain errors

5.2.3.1 Assumptions about averaging

GI-USB1 and the HTML5 application implement binary decimation. In this practice, 4^n samples are averaged together to produce a resulting code that has n bits extra precision. In order for this to yield accurate results, it is required that the original signal is both slightly noisy (so as to produce enough difference in subsequent samples that any quantization error is averaged out) and sufficiently smooth. In other words, mathematically decimation is equivalent to a DC averaging filter on a multimeter. As GI-USB1 is designed as a purely DC instrument and cannot measure AC current, this generally yields perfectly accurate results. However, in limited use cases a DC averaging instrument will not yield accurate results. Specifically:

1. If the current is zero-crossing (GI-USB can measure very small negative currents)
2. If either the voltage or current is clipping
3. If the AC component of the current is particularly large at frequencies near the sampling frequency.

The on-board capacitance on GI-USB1 is designed to be large enough that point 3 should never occur, unless an overvoltage event occurs. However, the other two situations can still arise in specific applications, even though these are not intended operating conditions for GI-USB1. In critical applications, the user should identify and correct these issues for accurate measurements.

5.2.3.2 LUT calculation errors

The HTML5 application uses look-up tables (LUTs) to correlate raw device data (ADC output data) to calibration values. Each individual current and voltage data point is fed into a function that takes into account temperature, source voltage and calibration values to produce an accurate result. The lookup tables use linear interpolation to guess calibration offset at data points that have not been explicitly calibrated against. Because the device offset as a function of voltage, temperature and initial offset is often not a linear function, this can induce nonlinearity errors. Care must be taken to produce a sufficient number of calibration samples to correct for this. The default calibration routine is designed in such a way to guarantee operation within the specified accuracy.

5.2.3.3 Lag

The measurement, communication and processing chain necessarily induces time lag. This is due to:

- ADC sampling time (1µs)
- Firmware buffering (0.4ms)
- USB buffering (1-10ms)
- Client-side buffering (1-10ms)
- Client-side processing and visualization (16-33ms)

Because of the fact that modern high-level operating systems have indeterminate timing, as well as the fact that USB and OS-level drivers do not have mandatory timing or buffering constraints, it is not possible to guarantee constraints on the amount in which measurements lag real time. This is why the HTML5 application is designed as an event-driven system, acting on data as it comes in instead of working on an absolute timescale. Important to note is that this also means that it is not possible to guarantee signal time-correctness. The only guarantee provided is that each sample received spans $1/125000^{th}$ of a second and that all samples are collected in definite sequence, but exactly when it was collected is indeterminate.

5.2.4 Other errors

5.2.4.1 Correcting for voltage drop

GI-USB1 measures the voltage of the power source at the output connector solder pads. However, especially at high output currents, the voltage drop in the connector and cabling may be significant. It is highly recommended to measure the resistance of your apparatus to correct for this if you wish to exclude cable losses from your power measurements. This is only significant for currents much higher than a few hundred mA.

5.2.4.2 INA216A1 shunt amplifier supply current

GI-USB1's on-board circuitry is entirely powered via the USB plug that also carries the data signals from and to the computer. The only exception to this is the supply current for the two on-board INA216A1 current shunt amplifiers. This current is drawn from the selected power source, which may be either USB or the external input on the JST XH auxiliary power connector. The combined current draw of these two amplifiers is at most 60µA. If very small batteries are used as an auxiliary power source, or if battery lifetime measurements are made, this has to be taken into account.

This current is not measured by GI-USB1. Only the 30mΩ shunt resistor conducts part of this current, which is too little to be measurable by the measurement chain.

6. Installing the GI-USB1 hardware

6.1 Windows XP and lower

On Windows XP and lower, GI-USB1 should automatically install. To check whether GI-USB1 has been succesfully installed, open Device Manager (Press the Windows-key + R, then type 'devmgmt.msc' (without quotes)). In Device Manager, there should be no unknown USB devices and one device under 'Ports (COM & LPT)'.

6.2 Windows Vista and 7

Windows Vista and 7 need the supplied 'Lufa VirtualSerial.inf' driver installation file. When GI-USB1 is plugged in, the operating system will notify you that installation of the new device was not successful. Go to Device Manager (Press the Windows-key + R, then type 'devmgmt.msc' (without quotes)) , right-click the new device (called 'USB-sensor', possibly truncated to 'USB-sens') and choose 'Update Driver Software'. Then choose 'Browse my computer for driver software', and in the next screen choose the supplied 'Lufa VirtualSerial.inf'. When a dialog window pops up that asks for permission to install unsigned driver software, click continue. GI-USB1 has now been successfully installed.

6.3 Windows 8, 8.1 and 10

Windows 8 and newer does not allow for unsigned driver installation anymore. This means that in order to install this hardware, unsigned driver installs must be enabled first. This is done by the following procedure:

- Press Windows key + C to bring up the Charms bar
- Click 'Settings'
- Click 'Change PC Settings'
- Click 'Update & Recovery' on the left side of the screen
- Click the 'Recovery'option on the left side of the screen
- Click 'Restart now'
- Once the computer has rebooted, click 'Troubleshoot'
- Then click 'Advanced options'
- Then choose 'Startup settings' and restart again
- After this second reboot, choose option 7: 'Disable driver signature enforcement'

After this procedure, the driver can be installed as in the procedure outlined for Windows Vista and Windows 7

6.4 Linux and OS X

Neither Linux nor OS X need any drivers to install; they will automatically recognize GI-USB1 as a USB CDC-ACM device.

7. Using the HTML5 application

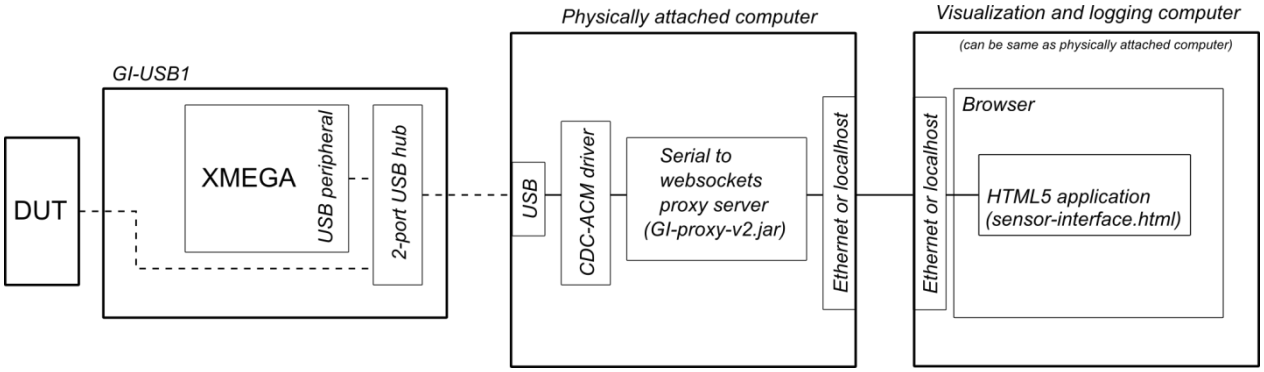


Figure 4: Software flow for GI-USB1

Please refer to [Figure 4: Software flow for GI-USB1]. GI-USB1 communicates with the Physically Attached Computer (PAC) via USB. It does so through a USB 2.0 hub, which enables both the sensor and the DUT to communicate with the PAC. The PAC runs a small cross-platform compatible Java application that receives data from the USB device (via an OS-specific CDC-ACM driver) and sends this over ethernet using the HTTP Websockets protocol. The Visualization and Logging Computer (VLC) receives this data and processes this through the HTML5 application. Note that the PAC and VLC can be the same physical computer but can also be computers on the same local network or on the other side of the world communicating over any internet connection. Because of the local nature of USB and the likely requirement to directly communicate with the DUT, GI-USB1 is primarily designed with local, relatively low-latency connections in mind.

7.1 Using the websocket proxy

GI-USB1 uses a websocket proxy application to convert the virtual serial port signals into a websocket stream which can be read by browser applications. This application is necessary to run the web application. To be able to use GI-USB1, start 'GI-proxy-v2' on the computer to which GI-USB1 is attached. A window will open with some debugging, connection and connection speed information. No further action is required, you can safely minimize this program and let it run in the background.

The data read from the device does not necessarily have to be visualized or logged on the same computer. In some applications, for instance accelerated life tests or emissions tests, it is more convenient to have the DUT in a physically different location. The websockets proxy acts as a websocket server, which can be accessed from anywhere within the same network. If the correct ports are forwarded, it can even be accessed from anywhere in the world over the internet.

For port forwarding instructions on various internet equipment, please refer to the excellent website http://portforward.com/. By default, GI-proxy-v2 communicates over ports 5331 and 8080. If these ports are not usable, these can be modified by editing the file config.cfg. Please refer to the troubleshooting section for more information.

7.2 Using the application

The websocket proxy is now running in the background (see [7.1 Using the websocket proxy]). In order to start measuring, just open the 'sensor-interface.html' file. Make sure that the following supplied files are in the same folder as sensor-interface.html:

1. Board.js
2. Fft.js
3. Serial.js
4. Support.js
5. Sensor-interface.css

The interface will automatically search for the device, connect to it and start displaying time series data.

By default, the application refreshes the time series data 15 times per second (15 frames per second, or FPS). This may be changed at the bottom of the screen using the FPS drop-down list. Note that higher frame rates can be very taxing on the computer. Low-end computers can usually smoothly display up to 15fps, high-end computers can go up to 60fps and phones or mobile devices should generally stay below 5fps.

7.3 Using the FFT functionality

The HTML5 application contains fast fourier transform, or FFT functionality. This displays the frequency domain variations in the data. This is useful for analyzing applications with strong harmonic power signal content. In order to switch to FFT view, simply click on the FFT tab at the top of the screen in the application.

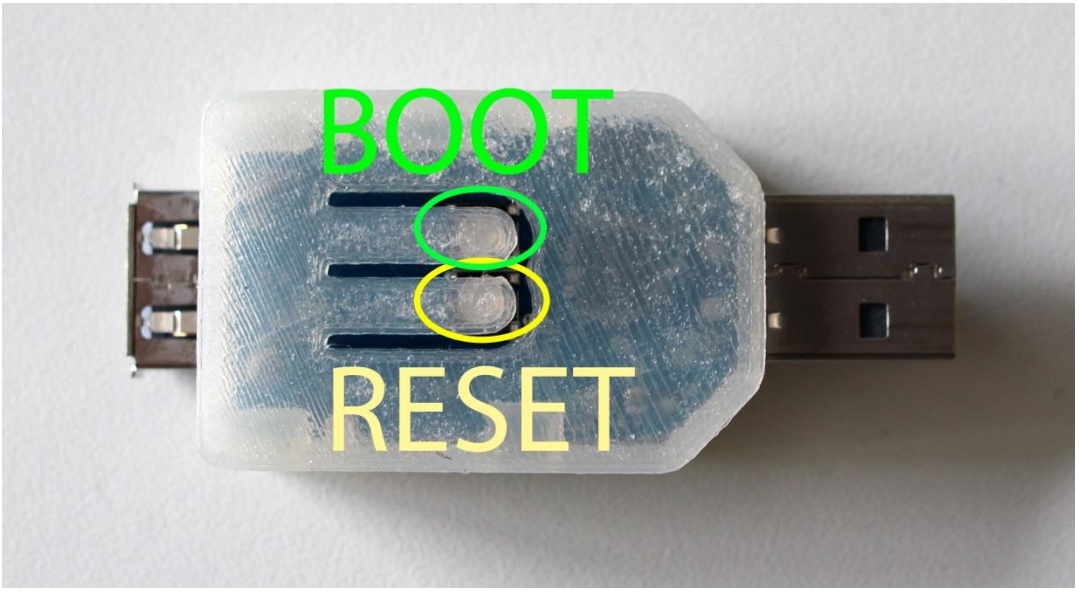
The FFT has an averaging function which averages together multiple FFT samples. Use the 'FFT averages' drop-down box at the bottom of the screen to increase or decrease the number of averages taken to draw an FFT. Note that the update frequency on screen is also dependent on the framerate; at 10fps 100 FFT averages will take 10 seconds, whereas at 60fps it will take less than 2 seconds to average 100 FFTs.

7.4 Calibration mode

Currently, calibration mode is not available for use. This feature will be added in the next major release.

8. Upgrading firmware

GI-USB1 firmware updates may appear on the GreenInvents support website. In order to update the firmware, download the 'FLASH.BIN' file from the firmware update page, making sure it is designated for GI-USB1. Then, reboot the USB sensor device into its firmware update mode by pressing the BOOT button (see image) while plugged into a USB port. The device will appear as a USB disk with two files on it: 'EEPROM.BIN' and 'FLASH.BIN'. Simply copy over the new FLASH.BIN to the USB disk and confirm to overwrite the old firmware. The programming will take approximately 1 second. After five seconds, pull out the device and plug it into the USB port again. It will now run the updated firmware.



9. Writing custom software for GI-USB1

This section will be updated after the next major release

10. Glossary of terms

DUT – Device under test

ADC – Analog to Digital converter

EEPROM – Electrically Erasable Programmable Read Only Memory, a section of non-volatile (will survive a power cycle) memory commonly used for storing calibration values and other device specific variables

CDC – Communications class device, a class of USB devices that is used for communications interfaces like modems, serial ports and networking interfaces

ACM – Abstract Control Model, a subclass of CDC that emulates serial ports over USB

CM – Calibration mode, an operational mode of GI-USB1 that allows the operator to, using an automated process, calibrate the analog-to-digital transfer function with respect to voltage, current, temperature and frequency.

BC1.1 – Battery Charging 1.1 specification, a physical layer adaptation of USB that allows upstream USB devices to signal towards downstream devices how much current can be drawn from the USB power lines without digital communications.

CDP – Charging downstream port, part of the BC1.1 specification that specifies the physical layer configuration to allow up to 1.5A current draw.

USB – Universal Serial Bus, a physical and protocol specification for digital communications between computer systems and peripheral devices.

DMSB – Driverless Mass Storage Bootloader, a piece of firmware on GI-USB1 that allows firmware upgrades without using external programmers. Upon activating the DMSB, GI-USB1 re-enumerates the device as a USB flash drive, allowing for a firmware upgrade by dragging and dropping a file into the USB flash drive.

XMEGA – a family of microcontrollers from Atmel, Inc.

LUT – Look-Up Table. A table of values that corresponds one or more input values to an output value.

PAC - Physically Attached Computer. The computer or computing device that is directly attached to the sensor via USB.

VLC – Visualization and Logging Computer. The endpoint computer receiving, processing and visualizing the data coming in from the sensor.

FFT – Fast fourier transform. A mathematical algorithm that transforms time-series data into frequency domain data.

11. Troubleshooting

11.1 Running the PAC and VLC on different computers over a network connection

In order to use the sensor interface over a network connection, two things need to happen:

- The HTML5 application needs to know which IP address or domain to connect to and which port
- The network equipment needs to open and forward port 8080 and 5331 to the PAC

By default, the HTML5 application connects to localhost. This parameter can be changed in the options tab under 'management URL'. If you are connecting over the internet, make sure to use the external IP address of the PAC. This can be found using various webservices like http://www.whatismyipaddress.com. For internal (local area) networks, use your local network IP, which can be found by running the command 'ipconfig' in a command prompt (Windows) or running the command 'sudo ip addr show' in a terminal under OS X or Linux.

Port forwarding for network equipment is beyond the scope of this manual. Make sure you are aware of all the network routers and modems between the PAC and VLC and individually forward the ports to each next node.

11.2 Changing ports on the websocket proxy

By default, the websocket proxy uses port 8080 for management and 5331 for measurement data transfer. If the PAC and VLC are the same physical computer, communication goes via OS sockets and requires no networking capabilities. However, if the application is run via the internet, on some types of network equipment the default ports 8080 and 5331 are either not configurable for port forwarding or already in use by other applications.

In order to change these addresses, create a new text file or use the supplied template 'config.cfg'. Open this in a text editor. The configuration file should read:

```
serial_proxy_port = 5331

config_server_port = 8080
```

After modifying this configuration file, make sure it is saved in the same folder as the .jar file and restart the proxy application.

12. Errata and usage hints

Warning: do not short battery input terminals. This may damage the device because of reverse current flow through the p-channel MOSFETs used to switch between USB and external power.

Warning: do not connect an external power source with a voltage lower than approx. 4.2V when the DUT is being powered by USB power. Reverse current flow through the p-channel MOSFET may overcharge or damage your external power source.

Notice: Take care not to press the BOOT and/or RESET buttons on the back of the device during normal operation. This may lock up the client application and cause data loss.

12.1 Bug reporting and feature requests

Bugs can be reported and are tracked via an installation of the Mantis bug tracking system on http://enijssen.com/bugs. This bug tracker is not public; if you wish to contribute to the project you can request a login via emile@greeninvents.com.

It is also possible to report bugs and feature req uests via e-mail. Please e-mail me at emile@greeninvents.com with 'bug report' in the subject line.

13. Index