Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

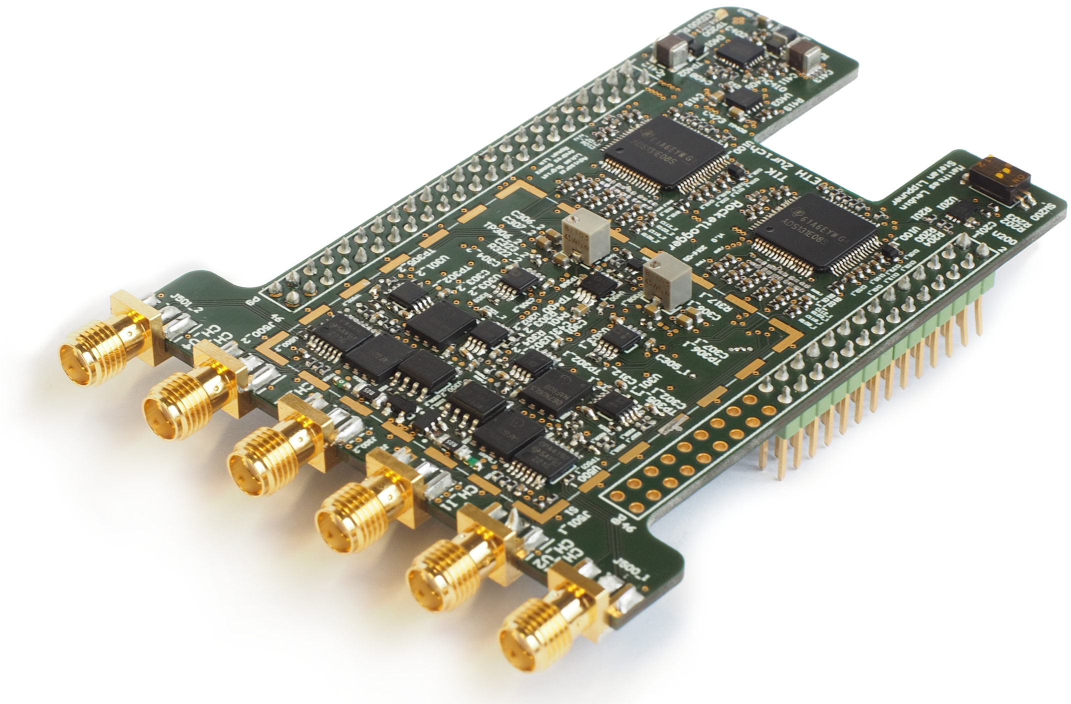
Institut für
Technische Informatik und
Kommunikationsnetze

Department of Information Technology and
Electrical Engineering

Spring Semester 2016

RocketLogger: Mobile Data-Logger for Ultra-Low Current and Power Measurements

Semester Thesis, SA-2016-40



Matthias Leubin, Stefan Lippuner
mleubin@student.ethz.ch, lstefan@student.ethz.ch

June 2016

Supervisors: Lukas Sigrist, lukas.sigrist@tik.ee.ethz.ch
Andres Gomez, gomeza@tik.ee.ethz.ch
Roman Lim, lim@tik.ee.ethz.ch

Professor: Prof. Dr. L. Thiele, thiele@tik.ee.ethz.ch

Abstract

With the appearance of wearable devices and the Internet of Things (IoT), energy harvesting nodes are becoming more and more important. These small standalone sensors and actuators, which harvest very limited amounts of energy from their environment, are driving researchers into the development of ultra-low power devices. To evaluate new designs, very accurate and fast measurement systems are required, which are able to capture the rapidly changing currents. On the other hand, the necessity of real-world experiments creates a demand for compact and portable equipment.

In this thesis we present the RocketLogger, a measurement device, which combines both properties, mobility and accuracy. It is able to measure powers in the range of 100 pW up to 2.5 W within the same measurement, while not being bigger than a human hand. The custom analog front-end allows the logging of power on several channels with sampling rates of up to 64 kSPS. In addition, it guarantees high precision over a dynamic range of 165 dB and has a seamless auto-ranging capability. The software handles the acquisition and logging of up to 13 Mbit/s of sensor data. In addition, it enables remote control and monitoring of the ongoing measurements and the handling of the stored samples through a simple and easy-to-use web interface.

Acknowledgements

We would like to thank our advisors, Lukas Sigrist, Andres Gomez and Roman Lim, for their support during our semester project. Furthermore, we would like to thank the TIK for allowing us to realise this project.

Matthias Leubin, Stefan Lippuner, June 2016

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

The advances in ultra-low power microcontroller and sensor development have opened a new application field. The reduced power consumption enables so called energy harvesting nodes, which can be deployed in environments, where only little energy is available and no batteries can be used. A typical application is the IoT, where a large number of small, embedded devices is connected to remotely sense and control objects. An example of a harvesting node is shown in Figure 1.1.

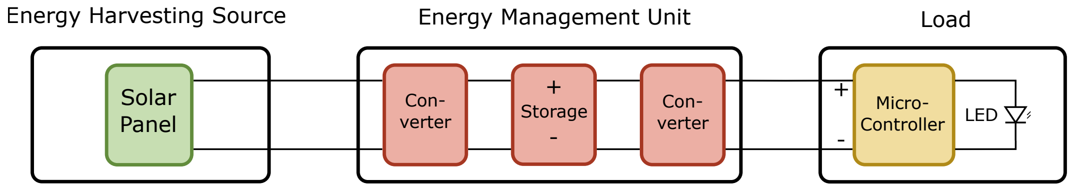


Figure 1.1.: Harvesting node

A typical architecture of such a system consists of three main parts: The source harvests the energy needed to power the system from its environment. In this example a solar panel is used. The energy is fed to the energy management unit (EMU), where it is buffered. The EMU supplies the load, which may, for example, be a sensor or an actuator. In Figure 1.1 it is simulated with a microcontroller and a flashing LED. In addition, the EMU also manages the power consumption of the load, to adapt it to the currently available harvesting power. An implementation of an EMU can be found in [1].

The big advantage of this setup is that does not rely on a large battery, which would need to be replaced or recharged periodically. Instead it uses a much smaller energy storage device and an additional power source. Therefore, it is suited for mobile applications and particularly for wearables or small standalone sensor devices.

The energy, harvested by these sources, can be very small and highly variable. The provided power changes dependent on the environment. In addition, the load may also have similar properties. The common trend in reducing power consumption even more results in ever lower sleep power and very short, but comparably high active power.

To allow the development of efficient energy management strategies, it is important to precisely know the characteristics of the harvesting source as well as the load.

1.2. The Task

The task of this thesis is to build a mobile data logger for precise characterization of such energy harvesting nodes. The device is capable of measuring both source and load at the same time to allow for synchronized analysis. It is able to measure the ultra-low harvesting and sleep current down to tens of nanoamperes. At the same time, it is capable of measuring the comparably high active power of more than 100 mA, which may occur, when for example wireless radio or flash memory components are used.

The task includes the design of an analog measurement circuit. It is implemented as an extension board, also called cape, for the BeagleBone, an embedded Linux platform. It is able to measure currents down to the micro- and nanoampere range and voltages down to the millivolt range.

The software is implemented on top of a Linux operating system (OS) running on the BeagleBone. It is responsible for the basic data management and storage. Furthermore, it allows the user to remotely control and monitor the logging.

Finally, the device is evaluated under lab as well as real-world conditions. In addition, it is compared to existing solutions and high-precision lab equipment.

1.3. Contributions

In this thesis we make the following contributions:

- We combine two different current measurement circuits to achieve a larger dynamic range. It includes a seamless automatic range switching, implemented in the analog front-end itself.
- We design the RocketLogger printed circuit board (PCB) as an extension board for the BeagleBone. The board includes several voltage and current channels, power supplies, as well as a digital interface.
- The software, implemented on the BeagleBone, is able to control the sampling on the cape and is optimized for high-rate data acquisition and storing.
- We design a web interface, which enables remote control of the logging. It also displays a data preview of the ongoing measurements and thus allows the user to check the setup.

1.4. Related Work

In this section four solutions, which are related to the previously explained task, are presented.

ACME

ACME is an open source solution for all-in-one multi-channel power and temperature measurement [2]. It was developed by BayLibre and comes as a cape for the BeagleBone. Since it is not able to measure the ultra-low currents and does not provide the required dynamic range, it is not suited for our application.

Ekho

Ekho is an emulator capable of recording energy harvesting environments and accurately recreating those conditions in the lab [3]. This makes it possible to perform realistic and repeatable experiments on newly developed harvesting devices. However, it has a limited accuracy, with a mean error of less than 77.4 µA. In addition, it is only able to measure one channel at a time and thus is not suited either.

FlockLab

FlockLab is a testbed for profiling wireless embedded systems developed at ETH Zurich [4]. It provides various services, namely general purpose input/output (GPIO) tracing and actuation, serial I/O, and power profiling. The current measurement, however, is limited to 160 mA and has a noise floor of about 5 µA. Therefore, it does not fulfil the requirements.

SmartEye

SmartEye is an observer platform for IoT testbeds [5], with a focus on energy efficiency to enable autonomous operation. Power profiling is only possible for devices connected via USB and no accuracy is specified. Therefore, it is not suited for energy harvesting environments, where multiple devices are connected to each other.

1.5. Outline

This report will first give a short introduction into the basics of current measurement circuit design in Chapter 2. Chapters 3 and 4 will then describe the design of the hardware and software of our own power logger in detail. Finally, Chapters 5 and 6 will show the achieved results and draw a conclusion.

2. Theory

In this chapter we will first introduce some of the models used to describe noise in electrical circuits. Based on this we will present some basic current measurement topologies and analyse their respective advantages and disadvantages.

2.1. Noise in Electronic Circuits

The following section is based on two books: [6, 7]. Refer to [6, ch. 11] for noise in electronic circuits and [7, ch. 1] for a more thorough, mathematical analysis of noise and random processes.

Noise is one of the limits on measuring physical quantities with electronic circuits. It is caused by small current and voltage fluctuations within the circuit itself. External signals which are picked up by a circuit are referred to as interference and also have to be addressed when designing high accuracy devices. While it is theoretically possible to amplify even very small signals to a high level, the amplification will also add noise to the signal. For a very small signal this noise contribution might be much larger than the actual signal. Different types of noise are distinguished and the ones relevant to this project will be discussed in this chapter.

As noise is a random phenomenon, it can only be modelled using stochastic methods. Generally, we treat it as a stationary, ergodic random process. If we have such a random process X, we can define its average:

$$\mu = E[X(t)] \tag{2.1}$$

and its autocorrelation:

$$R_X(\tau) = E[X(t)X(t-\tau)] \tag{2.2}$$

Where $E[\cdot]$ is the expectation value over time.

Often it is more convenient to investigate dynamic systems in the frequency domain, so we define the power spectral density (PSD) of X, $S_X(f)$, as the Fourier transform of its autocorrelation $R_X(\tau)$:

$$S_X(f) = F\left\{R_X(\tau)\right\}(f) = \int\limits_{-\infty}^{\infty} R_X(\tau)\exp(-2\pi i f\tau)d\tau \tag{2.3}$$

If we apply an ideal filter with bandwidth f_b to this random process, the average output power is:

$$P_X = \int\limits_{-f_b}^{f_b} S_X(f)df \tag{2.4}$$

2.1.1. Thermal Noise

Thermal noise is a type of noise, which is present in resistors. It is generated by the random, thermal motion of the electrons in the resistors and does not depend on the presence or absence of a current flow. The PSD of thermal noise is constant and directly proportional to the absolute temperature of the resistor.

$$S_X(f) = \frac{1}{2}\cdot 4kT \tag{2.5}$$

Where k is the Boltzmann constant and T is the absolute temperature.

It can either be modelled as a series voltage source, or as a shunt current source, as shown in Figure 2.1. Note that both of these sources are non-directional and that the noise voltage and current scale with the square root of the bandwidth.

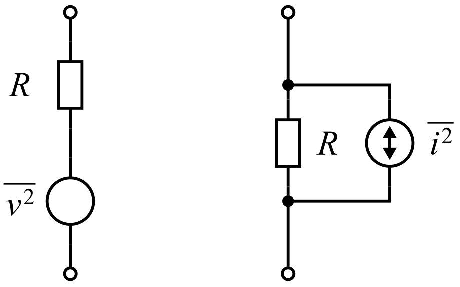


Figure 2.1.: Thermal noise model. Figure after [6].

$$\overline{v^2} = E[v(t)^2] = \int_{-f_b}^{f_b} S_X(f)R \cdot df = 4kTRf_b \tag{2.6}$$

$$\overline{i^2} = E[i(t)^2] = \frac{4kTf_b}{R} \tag{2.7}$$

Where f_b is the bandwidth of interest, R is the resistance of the resistor, and $\overline{x} = E[x]$ denotes the time-average.

2.1.2. Shot Noise

Shot noise is a type of noise, associated with direct-current in p-n junctions. As electrons and holes are only able to cross the p-n junction in discrete charges, the current is not continuous, but instead composed of many random current pulses. This can be modelled as an additional noise current with constant PSD:

$$\overline{i^2} = \overline{(i(t) - I)^2} = 2qI_Df_b \tag{2.8}$$

Where q is the elemental charge, I_D the direct-current through the diode, and f_b the bandwidth.

2.1.3. Flicker Noise (1/f Noise)

Flicker noise is a different type of noise, found in active devices and associated with a direct-current flow. While its sources vary, they are usually related to contaminations and crystal defects. It differs from shot and thermal noise, in that its PSD is not constant, but increases at low frequencies. It introduces a noise current with a spectral density of: [6]

$$\overline{i^2} = K_1 \frac{I^a}{f^b} f_b \tag{2.9}$$

Where I is the direct-current, f_b is the bandwidth, and K_1, a, and b are device-specific constants. As the origins of flicker noise lie in manufacturing defects, these constants vary by orders of magnitude and must be measured for each device if the noise performance is not specified. Due to its frequency dependence, flicker noise is most important at low frequencies, while thermal and shot noise usually dominate at higher frequencies.

2.1.4. Noise in Circuits

When introduced in a circuit, noise is propagated and amplified like a regular signal. Therefore, it is possible to model it using equivalent noise sources at different points in a circuit. If we have two independent, normal distributed noise voltage sources $v1, v2$ in series, we can replace them by a single noise source v_s as follows:

$$\begin{aligned}
v_s(t) &= v_1(t) + v_2(t) && (2.10)\\
E[(v_s(t))^2] &= E[(v_1(t) + v_2(t))^2] && (2.11)\\
\Leftrightarrow E[(v_s(t))^2] &= E[(v_1(t))^2] + 2E[v_1(t)]E[v_2(t)] + E[(v_2(t))^2] && (2.12)\\
\Leftrightarrow E[(v_s(t))^2] &= E[(v_1(t))^2] + E[(v_2(t))^2] && (2.13)\\
\Leftrightarrow \qquad \overline{v_s(t)^2} &= \overline{v_1(t)^2} + \overline{v_2(t)^2} && (2.14)
\end{aligned}$$

In the general case we can collapse two independent, zero-mean noise sources into a single source by adding their PSDs.

$$S_{X+Y}(f) = S_X(f) + S_Y(f) \tag{2.15}$$

For many circuits, including measurement devices, it is convenient to reference the noise to the input of the device. This allows the noise to be compared to the signal level. An example of this is shown in Figure 2.2, where the combination of flicker noise, and thermal or shot noise is illustrated nicely.

2.2. Current Measurement Circuits

Now that we have covered the basics of noise in electronic circuits, we can look at actual circuits which are used to measure currents. We assume that we want to convert it to a digital value using a voltage input analog-to-digital converter (ADC). Therefore, we need to design a circuit, which converts the input current into a voltage for the ADC. We will discuss four different topologies in this section.

2.2.1. Shunt Current Meter

One basic circuit to measure current is the shunt current meter. A shunt resistor R_{sh} is inserted in the path of the current I_{in} and the voltage across the resistor $V_{sh} = R_{sh}I_{in}$ is measured. The circuit under measurement will see a voltage difference $V_{in} = R_{sh}I_{in}$ compared to the case without the shunt current meter. This voltage drop, also called *burden voltage*, is generally undesired and kept relatively small. This necessitates that

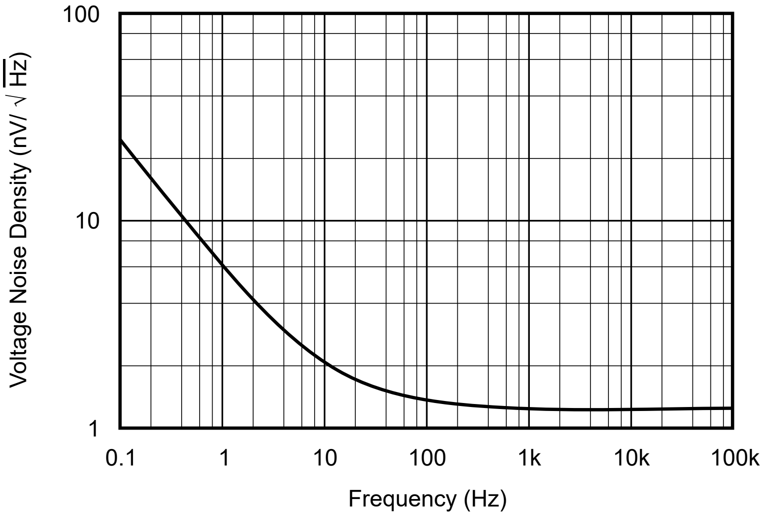


Figure 2.2.: OPA2211 operational amplifier input voltage noise density. At frequencies below 100 Hz the flicker noise dominates, while thermal and shot noise limit the performance at higher frequencies. Figure taken from [8]

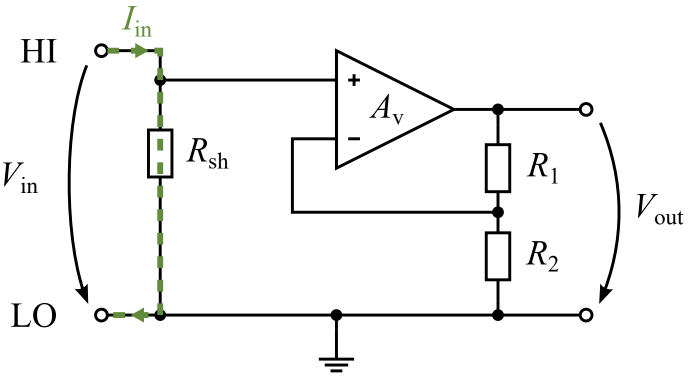


Figure 2.3.: Shunt current meter. The input current flow is highlighted in green. After [9]

we choose a small R_{sh}. As the measured voltage V_{sh} is identical to the burden voltage, it will also be small. In order to digitise this using an ADC it usually has to be amplified first, for instance using an operational amplifier in a non-inverting case, as shown in Figure 2.3. In this case the output voltage will be:

$$V_{out} = \frac{R_1 + R_2}{R_2} I_{in} R_{sh} \tag{2.16}$$

In general, the LO input terminal does not necessary have to be connected to ground. An instrumentation amplifier configuration can be used to remove a common-mode voltage offset, but both terminals still need to be within the range of the amplifier inputs.

The main drawback of this approach is, that the voltage we have to measure is identical to the burden voltage of the meter. The input noise of our amplifier will be constant and put a lower limit on the current resolution. This results in a trade-off between burden voltage at high currents and the measurement resolution, which is more critical at low currents. We will come back to this trade-off in Section 3.1.1. The advantage on the other hand is, that the effect on the measured circuit is very predictable, essentially corresponding to the insertion of a resistor in the path of the circuit [9].

2.2.2. Feedback Current Meter

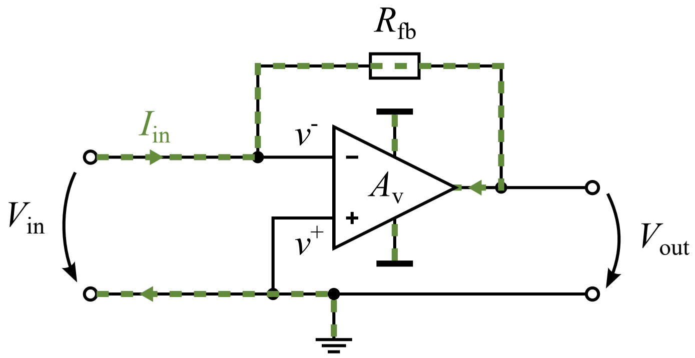


Figure 2.4.: Feedback current meter. The input current flow is highlighted in green. After [9]

The feedback current meter is an improvement on the shunt current meter, in that it reduces the burden voltage and decouples it from the output voltage. As shown in Figure 2.4, the current still flows into a resistor R_{fb}, but then it flows into the output of the operational amplifier and back to the input terminal through the power supply, closing the current loop. The purpose of the operational amplifier is, to keep the voltage difference between its two input terminals as small as possible. Using the gain equation of the operational amplifier (Equation 2.17) and Kirchhoff's Voltage Law, we can derive the input to output relation:

$$V_{out} = A_v(s)\left(v^+ - v^-\right), \quad A_v(s) \gg 1 \tag{2.17}$$

$$\Rightarrow V_{out} = A_v(s) \cdot -V_{in} = V_{in} - R_{fb}I_{in} \tag{2.18}$$

$$\Leftrightarrow \quad V_{in} = \frac{R_{fb}I_{in}}{1 + A_v(s)} \approx 0\,\mathrm{V} \tag{2.19}$$

$$\wedge \; V_{out} = \frac{-A_v(s)}{1 + A_v(s)} \cdot R_{fb}I_{in} \approx -R_{fb}I_{in} \tag{2.20}$$

Where $A_V(s = j\omega)$ is the gain of the amplifier as a function of the angular frequency $\omega = 2\pi f$. Ideal operational amplifier: $A_v(s) = \infty$.

Note that, assuming a large amplifier gain $A_v(s)$, the feedback resistor has essentially no influence on the burden voltage. We can therefore choose a large value for R_{fb} and get a large output voltage as well. This allows us to measure very small currents and have a very small burden voltage at the same time. One of the drawbacks is that the burden voltage depends on the gain of the amplifier and will therefore grow, as the gain decreases for higher frequencies[1]. Also the fact that the entire current has to flow through the amplifier, limits the current range to the maximum output current of the operational amplifier.

2.2.3. Logarithmic Feedback Current Meter

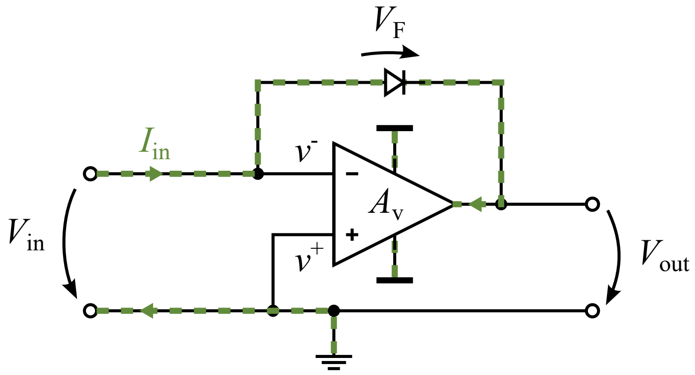


Figure 2.5.: Logarithmic feedback current meter

In order to increase the dynamic range of the feedback current meter, it is also possible to replace the feedback resistor R_{fb} with a feedback transistor or diode, as shown in Figure 2.5. As the diode current depends exponentially on its forward voltage V_F,

[1] General-purpose operational amplifiers have a low frequency pole in their transfer function, in order to be stable with any negative feedback.

the output voltage will be a logarithmic function of the measured current. Using this technique, it is possible to measure multiple decades worth of current with a single ADC:

$$I_{in} = I_S \left(e^{\frac{V_F}{v_T}} - 1 \right) \Leftrightarrow V_{out} \approx -V_F = -v_T \cdot \ln \left(\frac{I_{in}}{I_S} + 1 \right) \qquad (2.21)$$

Where $v_t = kT$ is the thermal voltage, k is the Boltzmann constant, T is the absolute temperature, and I_S is the saturation current of the diode.

Note that the output voltage depends linearly on $v_T = kT$. In order to get a current reading, the measurement needs to be adjusted for the junction temperature of the diode or transistor. This, and the relatively large manufacturing variations of the active devices, makes it hard to measure current accurately. These compensations are easier to handle in an integrated circuit, and commercial products are available. They are still limited to the relatively low output current of an operational amplifier and lose some accuracy compared to a linear feedback amplifier in return for their increase dynamic range[2] [9].

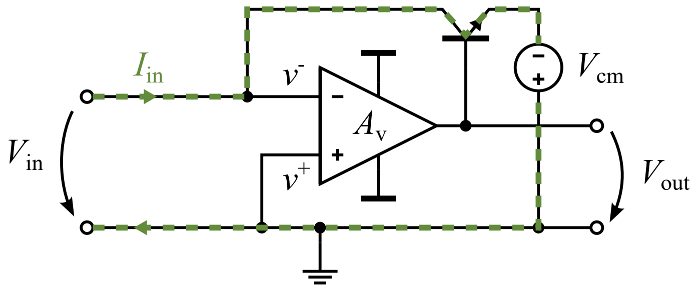


Figure 2.6.: Modified BJT feedback current meter. The offset voltage V_{cm} is required to compensate for the voltage drop across the BJT.

One possible change to the basic version is to use a bipolar junction transistor (BJT) instead of a diode, as shown in Figure 2.6. As the transistor provides current amplification, it reduces the output current demand on the operational amplifier. Using this circuit, it is possible to extend the range of the feedback current meter significantly, but the accuracy and temperature issues remain. The power dissipation in the BJT is also significant and further worsens the temperature problem.

[2] The LOG114 achieves ±20 % average total accuracy over 6 decades, or ±6 % average total accuracy over 5 decades [10, p. 8].

3. Hardware Design

As we have covered the theoretical background, we can now discuss the design process behind the hardware of the RocketLogger. As mentioned in the introduction, the goal of the project was to design a device, which is capable of measuring the power consumption of energy harvesting devices. It was decided that the BeagleBone would be used for data processing and that an add-on PCB, also called cape, would be designed for the analog front-end.

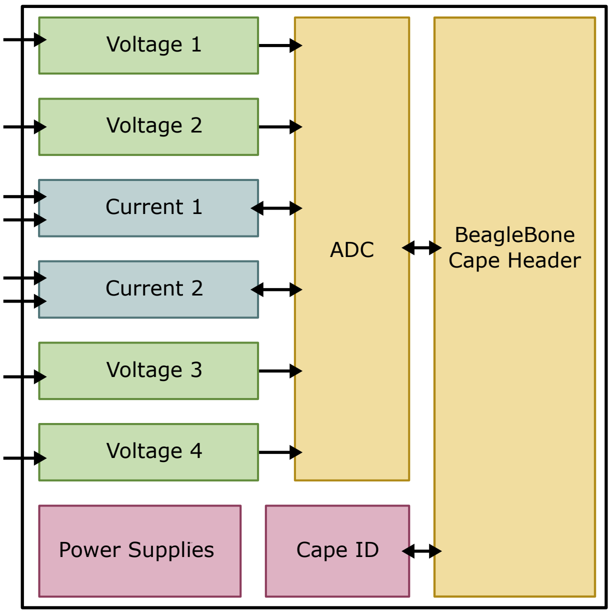


Figure 3.1.: RocketLogger Cape Concept

Figure 3.1 shows a block diagram of the hardware concept. On this PCB four voltage channels and two current channels would be connected to one or more ADCs. The ADCs in turn would be connected to the BeagleBone via a digital interface and the expansion headers. The current channels were identified as the main challenge, as they require an extremely large dynamic range, and the initial efforts were focused on finding a suitable solution for these.

We will first discuss the current channel architecture, then the important component choices and noise estimation, and finally the integration on the cape.

Current Range	500 mA
Accuracy	0.1 % to 1 %
Resolution	$\leq$ 100 nA
Burden Voltage	$\leq$ 100 mV
Sampling Rate	1 kSPS to 10 kSPS

Table 3.1.: Current channel goals

3.1. Current Channel

Energy harvesting devices usually operate by executing tasks in bursts. During these tasks the current draw can be relatively large, but they only last for a very short time. In between the bursts, the circuit enters a sleep mode, where the supply current is extremely low. In order to characterise the power consumption, the measurement device must be able to measure current at both extremes. An additional difficulty lies in the fast switching between the two states. While the actual values will vary wildly depending on the specific device, we would like to be able to measure as many of them as possible.

Table 3.1 shows the goals that were set at the beginning of the project. Note that, while the accuracy requirement is not too strict, the range of the current measurements is extremely large. The minimum required dynamic range is 134 dB or 22.3 bit with very sharp current steps. In order to achieve this dynamic range at the specified sampling rates, several different architectures were considered and will be presented in this section.

3.1.1. Single-Range Shunt Current Meter

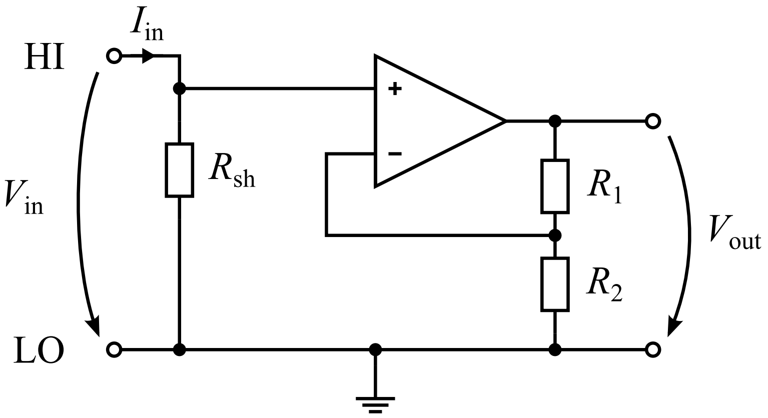


Figure 3.2.: Shunt current meter. After [9]

The simplest approach is to just use a shunt current meter, as described in Section 2.2.1. We can calculate the maximum allowable shunt resistance as follows:

$$V_{in,\max} \geq I_{in,\max} \cdot R_{sh} \Leftrightarrow R_{sh} \leq \frac{V_{in,\max}}{I_{in,\max}} \tag{3.1}$$

With our specific goals, this yields $R_{sh} \leq 200\,\mathrm{m\Omega}$. Therefore, a current of 100 nA will produce a burden voltage of $V_{sh} \leq 20\,\mathrm{nV}$ across the shunt resistor. At 1 kSPS, the highest resolution commercially available 32-bit ADCs, have an input noise root mean square (RMS) voltage of around 139 nV, which is not enough to achieve our desired resolution directly [11]. When instead choosing to amplify the signal before passing it to the ADC, the input voltage noise of the amplifier becomes problematic. Even "ultra-low" noise devices have an RMS input voltage noise of about 20 nV over a bandwidth of 500 Hz [12], making it implausible to measure the burden voltage this accurately. A more sophisticated approach is required.

3.1.2. Single-Range Feedback Current Meter

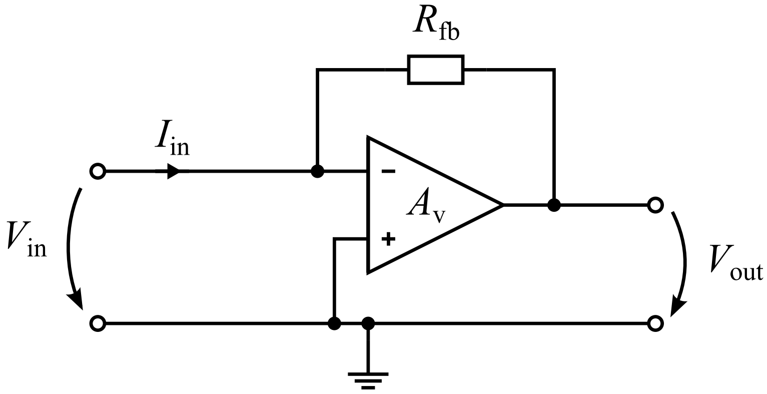


Figure 3.3.: Feedback current meter. After [9]

Using a feedback current meter, we can overcome the problem of the low voltage measurement by choosing a larger feedback resistor R_f. The main limitation in this case is the output current of the operational amplifier and the dynamic range of the ADC. ADCs with (RMS) signal-to-noise ratio (SNR) $\geq$ 134 dB exist, if just barely [13]. Ideally we would prefer a larger dynamic range, which might be achieved using multiple channels and amplifiers. The output current of the amplifier is a larger problem though, as most precision amplifiers have a maximum output current of about 10 mA. High current parts usually have a high input offset voltage and require large input bias currents. In addition to this, the high power dissipated in the circuit would significantly reduce battery life and disturb the other precision electronics on the cape.

Using a logarithmic feedback current meter alleviates the constraints on the ADC, but does not help with the power dissipated in the circuit. The reduced accuracy is also problematic, especially since it is not easily possible to correct for the thermal voltage on a PCB.

3.1.3. Dual Range Current Meter

As none of the previously mentioned approaches fully satisfy our requirements, we can try to combine two approaches and exploit their respective advantages. We have several channels responsible for a part of the current range, and switch between them when necessary. Shunt current meters are attractive for high current ranges, while the feedback current meters work well for low currents.

One of the key decisions is, how to switch between the ranges. The options we considered are:

- Manual switching: The user operates a switch, which physically connects the desired channel.
- Program controlled switching: The software samples the currently active channel and decides if it should switch it, which is then done using active devices, such as relays or transistors.
- Analog range switching: The analog front-end automatically generates the control signals for the range switching devices.

One of the characteristics of energy harvesting devices is the extreme difference between the ultra-low sleep currents and the active currents. If we want to measure the current accurately, our device needs to be able to cope with the fast switching between the two states. Manual range switching is not a real option, due to its high latency. Switching the range in software requires that we receive a sample from the ADC and then react based upon this information. This implies that we lose some of our samples in the process, if they are outside the original range. If possible we would like to avoid this, and therefore chose to implement the range switching in the analog front-end.

For the actual implementation of the dual range we chose to combine a shunt current meter with a linear feedback current meter. The shunt current meter is always active and is used for measurements above 2 mA and for the range switching. The range switching circuitry is a window comparator, which checks if the input current $|I_{in}|$ is below 2 mA. If that is the case, the feedback current meter is enabled, which allows a more accurate measurement of the input current. At the same time the shunt current meter output is still valid, but is only used for the range switching. The actual switching operation is performed by two MosFETs, which connect the low side of the shunt resistor either directly to ground, or to the virtual ground at the input of the feedback current meter. We also considered using relays to switch between the two ranges, but decided against using them since they take several milliseconds to switch.

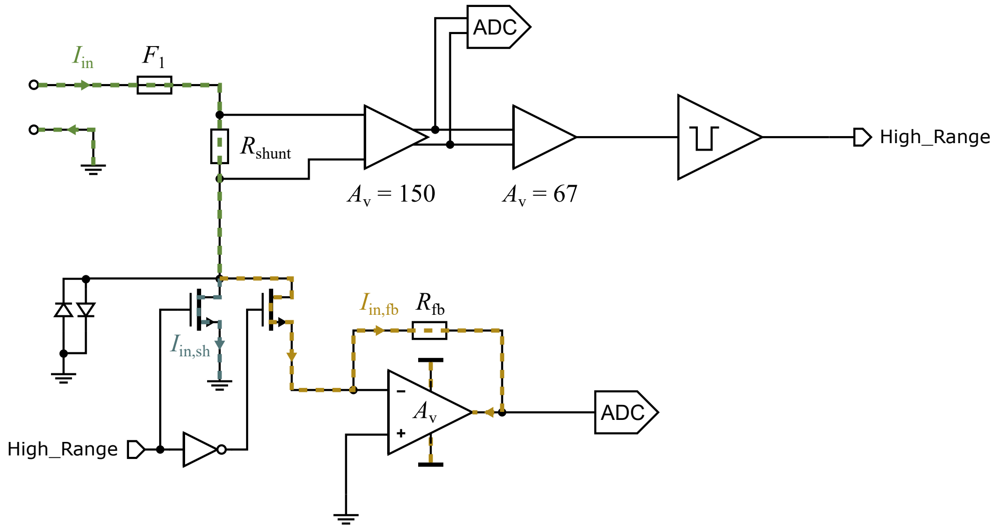


Figure 3.4.: Dual range current meter

Unfortunately, even MosFETs cannot be switched on and off instantly and therefore we have to consider what happens during the range switching. On the output side the effects are not too significant. At a sampling rate f_s of 10 kSPS each sample takes 100 µs and it is easily possible to switch the range more quickly than that. In the worst case this would invalidate two samples. In order to avoid aliasing, we will need to provide a low-pass filter with cut-off $f_c \leq \frac{f_s}{2}$ at the output of the current meter. This filter will have a 10 % to 90 % rise time on the order of $t_{\text{rise}} \approx \frac{0.35}{f_c} \geq 0.699 \cdot \frac{1}{f_s} = 69.9\,\mu\text{s}$ [6]. Therefore, we can conclude that we will not lose any additional samples as long as we do not exceed a range switching time of $(1 - 0.699)\frac{1}{f_s} = 30.1\,\mu\text{s}$. This is easily achievable using MosFETs. During the transition period we can also use the output of the shunt current meter, as it is still valid.

Range switching also has a negative impact on the input side. If our feedback current meter cannot handle the input current the burden voltage will be significantly larger than during the regular operation. In order to limit this effect, and to protect the circuit, we have added two back-to-back Schottky diodes in parallel to the MosFETs. These carry the current during the transition from one range to the other and limit the burden voltage to about 200 mV during this time. If even this effect on the measured circuit is not tolerable, it is also possible to force the current meter into the high current range in software.

In order to ensure the stability of the range-switching process, the bandwidth was limited at both the MosFET driver and the window comparator. Furthermore, a hysteresis of 200 µA was added to avoid oscillation due to noise. To protect the channel in case of continued overload a fuse was added at the terminal. During our semester project we built a prototype of the current channel and were able to verify its operating principle and performance.

3.1.4. Two-Channel Current Measurement

In energy harvesting scenarios, source and load are often separated by an EMU. For characterization, we have to measure the currents into both of them at the same time. In order to measure multiple currents simultaneously, usually completely isolated current meters are used. Isolating the current channels would however have required the addition of several duplicate power supplies and isolation devices. Since PCB area is scarce on our portable logger, we wanted to avoid this if possible. If we place some restrictions on the system under measurement, we can avoid the isolation altogether. The restrictions are as follows:

1. The entire system must have a shared potential, or it must be possible to create a shared potential.
2. It is only possible to measure currents into this shared potential.

Note that the first requirement is usually satisfied, since a lot of circuits share a *GND* connection. The second part is more restrictive, in that it only allows simultaneous low-side current measurements. With some consideration it is generally also possible to measure the relevant currents using this arrangement. Figure 3.5 shows a typical energy harvesting setup, where both constraints are satisfied. While the current I_C into circuit C is measured with a negative sign, one can easily correct for this in software. If isolated measurements are necessary, it is also possible to use multiple battery-powered RocketLoggers, or to combine them with other measurement instruments.

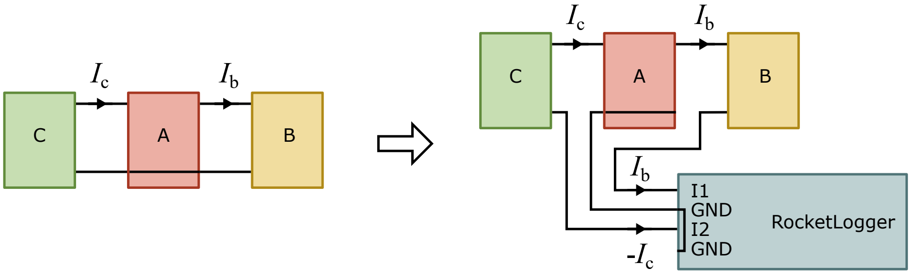


Figure 3.5.: Two channel current measurement

3.2. Voltage Channel

In addition to the current measurements, we would also like to capture several voltages at the same time. The main challenge in designing the voltage channel was this simul-

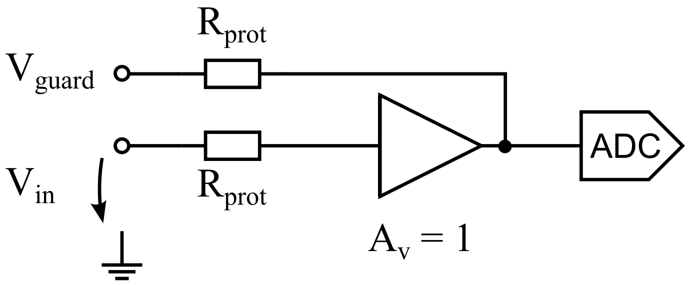


Figure 3.6.: Voltage channel diagram

taneous measurement of voltage and current. Any current into the voltage channel will disturb the current measurement. In order to achieve a very low input current, the voltage input is buffered by a rail-to-rail voltage follower. The output signal of this amplifier can now be used in the rest of the front-end, without loading the actual input. For a high impedance measurement device, leakage currents through the PCB or cables also begin to matter. In order to combat this effect, the output of the voltage buffer is used like a shield around the input conductor. Since there is almost no voltage difference between these two conductors, the leakage current will be very small as well. This technique, known as *guarding*, also improves the dynamic performance of the voltage measurement [9].

In order to protect the channel from excessive voltages on the input, and shorting on the guard conductor, a current limiting resistor R_{prot} was added. Note that this resistor does not correspond to the input impedance of the voltage channel, which is significantly larger.

3.3. Component Selection

Now that we have a basic topology for our measurements channels, we can formulate the requirements for each component. Based on this we can select the most appropriate product and then design the final circuits and evaluate their performance.

3.3.1. ADC

Let us first consider the ADC. In order to differentiate between two small signals, it requires a sufficiently large dynamic range. There are two equivalent ways to specify this, either as a SNR or as effective number of bits (ENOB), which usually refer to the RMS noise floor at 0 V input. The conversion between the two forms can be done as follows:

$$\mathrm{SNR} = 20\frac{\ln 2}{\ln 10} \cdot \mathrm{ENOB} = 6.02 \cdot \mathrm{ENOB} \tag{3.2}$$

Whenever we switch to a higher current range, the noise will increase compared to that of the lower range. We specify a 0.1 % relative noise floor for the readings at these switching points, with some margin for the hysteresis. This gives us a very high dynamic range requirement for the shunt current range of 114 dB. We alleviate this, by splitting it in two parts, the original high range, and a new medium range. For this, we can use a second amplification stage before the ADC, whose added noise will be negligible[1]. The SNR requirements are listed in Table 3.2.

	Noise Floor	**Maximum**	**Dynamic Range**
Voltage Range	100 µV	5 V	94 dB
Current Shunt (Full)	1 µA	500 mA	114 dB
Current Shunt (High)	11.2 µA	500 mA	87 dB
Current Shunt (Medium)	1 µA	11.2 mA	87 dB
Current Feedback (Low)	20 nA	2 mA	100 dB

Table 3.2.: ADC dynamic range requirements

Further important specifications include the errors introduced by temperature drift and non-linearity. It is easily possible to correct for constant offsets in software, but device temperature dependent errors are harder to deal with. Refer to [14] for a more in-depth discussion of ADC performance characteristics.

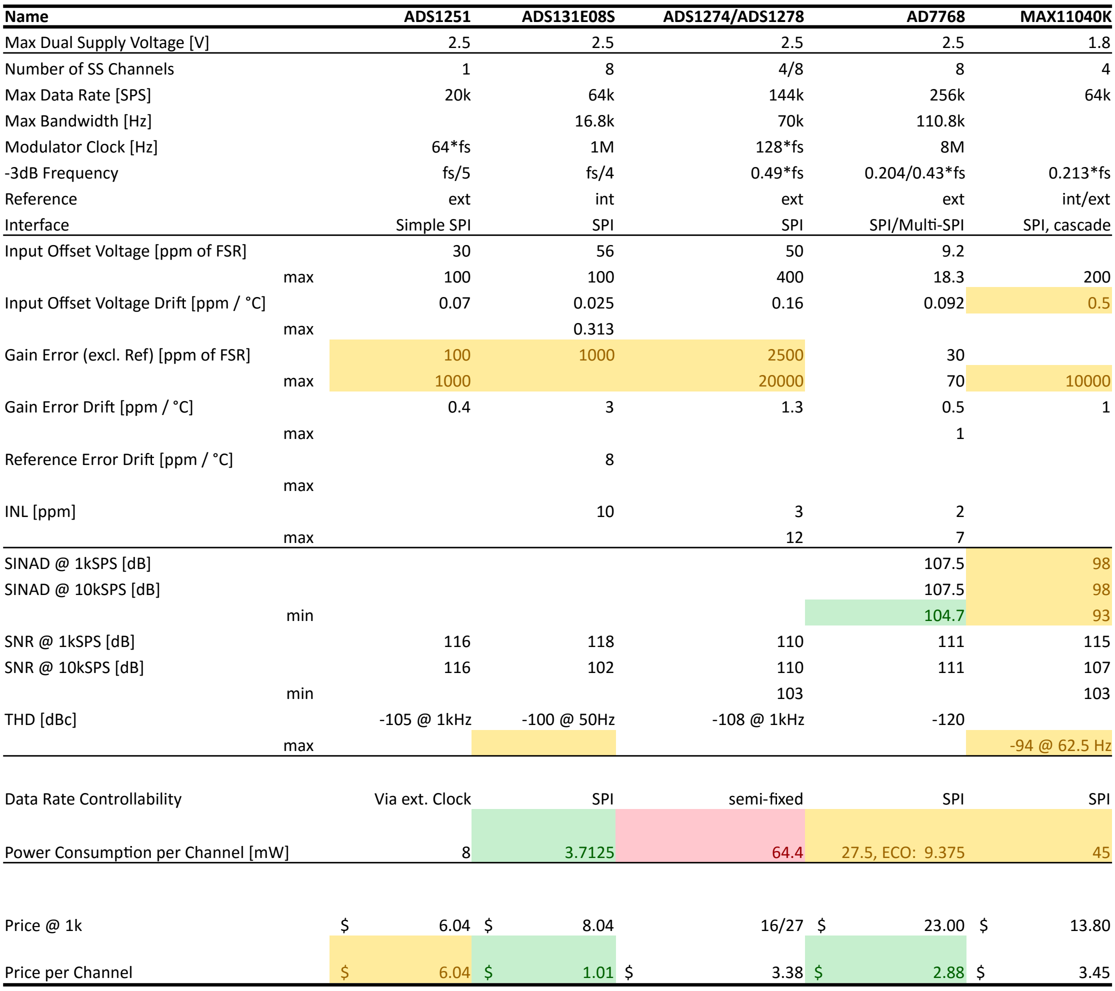
Table 3.3 shows some of the ADCs we evaluated. Both the *AD7768* and the *ADS131E08S* would have been a good fit for our design. In the end we decided to go with the *ADS131E08S* due to its low power consumption, good availability, low price and integrated voltage reference. As it has integrated programmable gain amplifiers (PGAs), no additional hardware was required for the medium current range. With a total of 10 voltages measured simultaneously we used two ADCs on our RocketLogger cape.

The ADC RMS input noise voltages for the internal 4 V reference with a 5 V power supply and unity gain are as follows [15]:

$$\overline{(v_{\mathrm{n,adc,in}}(f_b = 16.8\,\mathrm{kHz}))^2} = (1.47\,\mathrm{mV})^2 \tag{3.3}$$

$$\overline{(v_{\mathrm{n,adc,in}}(f_b = 262\,\mathrm{Hz}))^2} = (5.78\,\mathrm{\mu V})^2 \tag{3.4}$$

[1] In an amplification cascade the noise of the first amplification stage will usually dominate, if it has high gain [7].



Name		ADS1251	ADS131E08S	ADS1274/ADS1278	AD7768	MAX11040K
Max Dual Supply Voltage [V]		2.5	2.5	2.5	2.5	1.8
Number of SS Channels		1	8	4/8	8	4
Max Data Rate [SPS]		20k	64k	144k	256k	64k
Max Bandwidth [Hz]			16.8k	70k	110.8k	
Modulator Clock [Hz]		64*fs	1M	128*fs	8M	
-3dB Frequency		fs/5	fs/4	0.49*fs	0.204/0.43*fs	0.213*fs
Reference		ext	int	ext	ext	int/ext
Interface		Simple SPI	SPI	SPI	SPI/Multi-SPI	SPI, cascade
Input Offset Voltage [ppm of FSR]		30	56	50	9.2	
	max	100	100	400	18.3	200
Input Offset Voltage Drift [ppm / °C]		0.07	0.025	0.16	0.092	0.5
	max		0.313			
Gain Error (excl. Ref) [ppm of FSR]		100	1000	2500	30	
	max	1000		20000	70	10000
Gain Error Drift [ppm / °C]		0.4	3	1.3	0.5	1
	max				1	
Reference Error Drift [ppm / °C]			8			
	max					
INL [ppm]			10	3	2	
	max			12	7	
SINAD @ 1kSPS [dB]					107.5	98
SINAD @ 10kSPS [dB]					107.5	98
	min				104.7	93
SNR @ 1kSPS [dB]		116	118	110	111	115
SNR @ 10kSPS [dB]		116	102	110	111	107
	min			103		103
THD [dBc]		-105 @ 1kHz	-100 @ 50Hz	-108 @ 1kHz	-120	
	max					-94 @ 62.5 Hz
Data Rate Controllability		Via ext. Clock	SPI	semi-fixed	SPI	SPI
Power Consumption per Channel [mW]		8	3.7125	64.4	27.5, ECO: 9.375	45
Price @ 1k		$ 6.04	$ 8.04	16/27	$ 23.00	$ 13.80
Price per Channel		$ 6.04	$ 1.01	$ 3.38	$ 2.88	$ 3.45

Table 3.3.: ADC evaluation (selection). Values are taken from the respective datasheets.

3.3.2. Shunt Current Meter Amplifier

In order to measure the voltage across the shunt resistor we use an instrumentation amplifier. Low input current, ultra-low noise, and low offset voltage drift are the most important considerations for this device. Table 3.4 shows the amplifiers we looked at in detail. We decided to go with the *AD8421BRZ*, since it was the lowest noise part, which was able to meet our other requirements [16].

3.3.3. Feedback Current Meter Operational Amplifier

The critical features for the operational amplifier in the feedback current meter are the input offset voltage drift, input offset voltage, and the input bias current. Low noise

Name		INA333	INA129	AD8421	MAX4197	CS3002
Max Dual Supply Voltage [V]		2.75	18	18		3.35
Offset Voltage [uV]		10	10			
	max	25	50	25/60		10
Offset Voltage Drift [uV/ °C]			0.2			0.01
	max	0.1	0.5	0.2/0.4		0.05
Bias Current [nA]		0.07	2	0.1	6	0.1
	max	0.2	5	0.5	20	1
Input Noise Density @1kHz [nV/rt(Hz)]		50	8	3	8.7	6
	max			3.2		
Gain @10kHz [dB]		30	60	>60		90
	min					
Price @1k [$]		2.03	3.94	5.21/2.94		2.54

Table 3.4.: Shunt current meter amplifier evaluation. The AD8421 is available in two different quality grades. Values are taken from the respective datasheets.

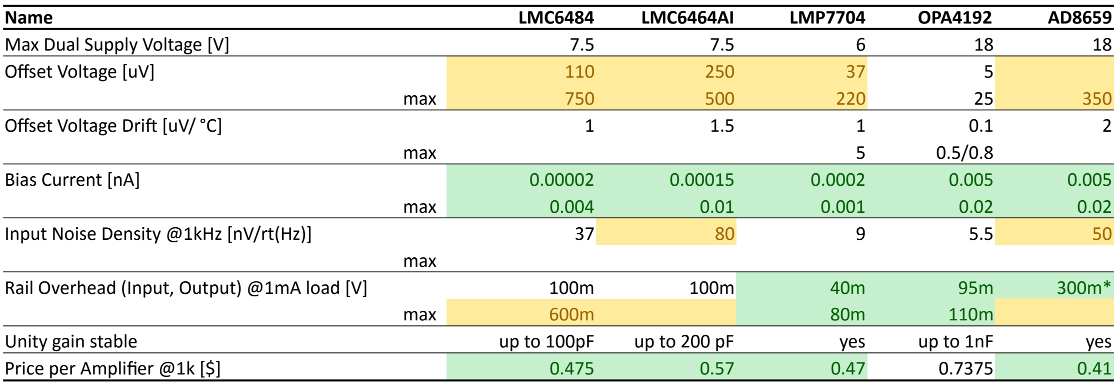
and a high gain over the entire frequency range are also desired. Since the feedback is variable, the unity gain stability is also required. Table 3.5 shows some of the evaluated operational amplifiers. We selected the *ADA4522* due to its excellent input offset performance [17].

Name		MAX4475	MAX44250	OPA2211	OPA209	OPA2227	OPA827	OPA192	AD8676	ADA4522
Max Dual Supply Voltage [V]		2.75	10	18	18	18	18	18	18	27.5
Offset Voltage [uV]		70	3	60	35	5	75	5	12	1
	max	350	6	185	150	100	150	25	50/60/100	5
Offset Voltage Drift [uV/ °C]		0.3	0.005	0.35	1	0.1	0.1	0.1	0.2	0.004
	max	6	0.026		3		2	0.5/0.8	0.6	0.022
Bias Current [nA]		0.001	0.2	50	1	2.5	3	0.005		0.05
	max	0.15	1.4	350	4.5	10	10	0.02	2	0.15
Input Noise Density @1kHz [nV/rt(Hz)]		4.5	5.9	1.1	2.2	3	4	5.5	2.8	5.8
	max									
Gain @10kHz [dB]		53	70	80	65	55	70	60	60	50
	min									
Unity gain stable		yes	up to 400 pF	yes	yes	yes	yes	up to 1nF	yes	yes
Price @1k [$]		1.2	1.34	7.45	1.71	2.39	6.95	1.26	2.65	2.11

Table 3.5.: Feedback current meter amplifier operational evaluation (selection)

3.3.4. Voltage Input Buffer Amplifier

A rail-to-rail amplifier is required for the voltage channel input buffer. A low rail overhead allows us to measure voltages very close to the supply of the amplifier and a low input current increases our input impedance. Multiple good options exist, and we picked the *OPA4192* due to its low offset voltage, low noise and low rail overhead [18].



Name		LMC6484	LMC6464AI	LMP7704	OPA4192	AD8659
Max Dual Supply Voltage [V]		7.5	7.5	6	18	18
Offset Voltage [uV]		110	250	37	5	
	max	750	500	220	25	350
Offset Voltage Drift [uV/ °C]		1	1.5	1	0.1	2
	max			5	0.5/0.8	
Bias Current [nA]		0.00002	0.00015	0.0002	0.005	0.005
	max	0.004	0.01	0.001	0.02	0.02
Input Noise Density @1kHz [nV/rt(Hz)]		37	80	9	5.5	50
	max					
Rail Overhead (Input, Output) @1mA load [V]		100m	100m	40m	95m	300m*
	max	600m		80m	110m	
Unity gain stable		up to 100pF	up to 200 pF	yes	up to 1nF	yes
Price per Amplifier @1k [$]		0.475	0.57	0.47	0.7375	0.41

Table 3.6.: Voltage input buffer amplifier evaluation

3.4. Noise Calculations

Once we selected all the components, we were able to design the entire circuit, including some auxiliary hardware. Based on those circuits we can calculate the expected noise performance of our channels and check if we meet our original goals.

3.4.1. High Range Current Channel

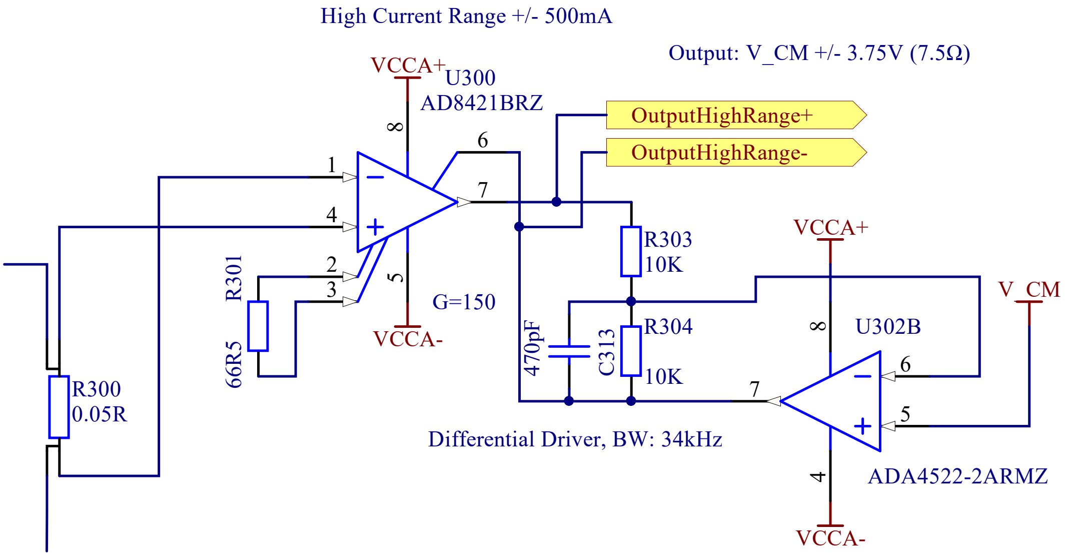


Figure 3.7.: High range current front-end

Let us first perform a noise analysis for the high range current front-end, shown in Figure 3.7. The operational amplifier *U302B* is used to set the common-mode voltage $V_{cm} = 2.35\,\text{V}$. We start by calculating the noise contributions of the individual sources, referenced to the output.

The source resistance R_S will cause a noise current $\overline{(i_{n,S})^2} = \frac{4kTf_b}{R_S}$, that is propagated to the output:

$$\overline{(v_{\text{n,hi,out,src}}(f_b))^2} = 4kTR_S f_b \cdot \left(\frac{R_{sh}}{R_{inp} + R_S}\right)^2 A_{\text{hi}} \tag{3.5}$$

Where f_b is the bandwidth of interest, $R_{inp} = R_{sh} + R_{\text{fuse}} + R_{\text{switch}}$ is the input resistance of the current meter, and $A_{\text{hi}} = 150$ is the voltage amplification. $R_{sh} = R_{300} = 50\,\text{m}\Omega$ is the shunt resistance, R_{fuse} the resistance of the fuse, and R_{switch} is the MosFET switch resistance. Note that the source resistance is not controllable at design-time, but depends on the measured system.

The output voltage noise contribution of the instrumentation amplifier voltage noise is as follows [16]:

$$\overline{(v_{\mathrm{n,out,ina,v}}(f_b))^2} = \overline{(v_{\mathrm{n,ina,out}}(f_b))^2} + \overline{(v_{\mathrm{n,ina,in}}(f_b))^2} \cdot (A_{\mathrm{hi}})^2 + 4kTR_G f_b \cdot (A_{\mathrm{hi}})^2 \tag{3.6}$$

Where $R_G = R_{301} = 66.5\,\Omega$ is the gain resistor of the instrumentation amplifier, $\overline{(v_{\mathrm{n,ina,out}}(f_b))^2}$ is the output voltage noise of the amplifier, and $\overline{(v_{\mathrm{n,ina,in}}(f_b))^2}$ is the input voltage noise of the amplifier.

The instrumentation amplifier also has an input current noise $\overline{(i_{\mathrm{n,ina,in}}(f_b))^2}$, which propagates to the output as follows:

$$\overline{(v_{\mathrm{n,out,ina,i}}(f_b))^2} = \overline{(i_{\mathrm{n,ina,in}}(f_b))^2} \cdot (R_{sh}||(R_{inp} + R_S))^2 \cdot (A_{\mathrm{hi}})^2 \approx \overline{(i_{\mathrm{n,ina,in}}(f_b))^2} \cdot (A_{\mathrm{hi}} R_{sh})^2 \tag{3.7}$$

If we assume that the instrumentation amplifier has perfect control over its differential output voltage, we can neglect the effects of the common mode voltage op-amp *U302B*. This gives us the following relation for the noise voltage at the output of the high current range:

$$\overline{(v_{\mathrm{n,hi,out}}(f_b))^2} = \overline{(v_{\mathrm{n,hi,out,src}}(f_b))^2} + \overline{(v_{\mathrm{n,out,ina,v}}(f_b))^2} + \overline{(v_{\mathrm{n,out,ina,i}}(f_b))^2} + \overline{(v_{\mathrm{n,adc,in}}(f_b))^2} \tag{3.8}$$

If we transfer this noise voltage to the input current, we get:

$$\overline{(i_{\mathrm{n,hi,in}}(f_b))^2} = \overline{(v_{\mathrm{n,hi,out}}(f_b))^2} \left(\frac{1}{A_{\mathrm{hi}} R_{sh}} \right)^2 \tag{3.9}$$

The medium current range utilises the PGA with a gain of 12x to reduce the impact of the ADC noise. The *ADC* bandwidth for 1 kSPS and 64 kSPS is 262 Hz and 16.8 kHz respectively. Inserting the values, we get the following numerical values:

$$\overline{(i_{\text{n,hi,in}}(f_b = 16.8\,\text{kHz}))^2} = (196\,\mu\text{A})^2 + \frac{4kTR_Sf_b}{(R_{inp}+R_S)^2} \tag{3.10}$$

$$\overline{(i_{\text{n,hi,in}}(f_b = 262\,\text{Hz}))^2} = (1.55\,\mu\text{A})^2 + \frac{4kTR_Sf_b}{(R_{inp}+R_S)^2} \tag{3.11}$$

$$\overline{(i_{\text{n,mid,in}}(f_b = 16.8\,\text{kHz}))^2} = (16.6\,\mu\text{A})^2 + \frac{4kTR_Sf_b}{(R_{inp}+R_S)^2} \tag{3.12}$$

$$\overline{(i_{\text{n,mid,in}}(f_b = 262\,\text{Hz}))^2} = (1.36\,\mu\text{A})^2 + \frac{4kTR_Sf_b}{(R_{inp}+R_S)^2} \tag{3.13}$$

3.4.2. Low Range Current Channel

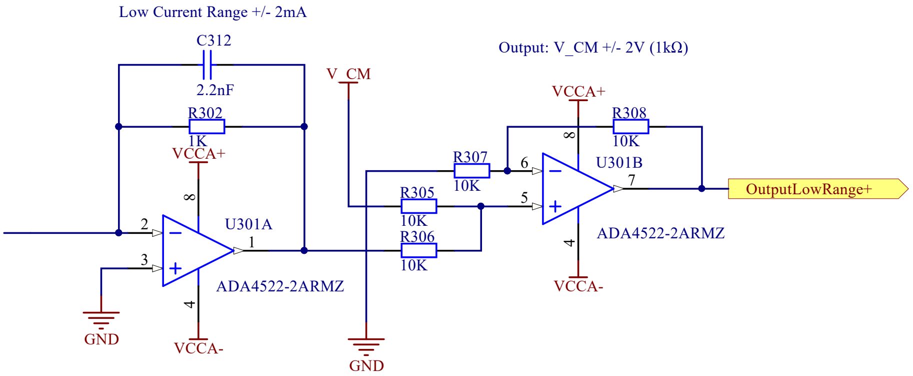


Figure 3.8.: Low range current front-end

Figure 3.8 shows the schematic for the low current range front-end. The output of the feedback meter needs to be shifted to the common mode voltage V_{cm} before it can be digitised by the ADC.

The transfer function of the input noise is:

$$\overline{(v_{\text{n,lo,out,src}}(f_b))^2} = 4kTR_Sf_b \cdot \left(\frac{R_{fb}}{R_{inp}+R_S}\right)^2 \tag{3.14}$$

One of the advantages of the the feedback current meter circuit, used in the low current range, is the fact that the input voltage noise of the primary op-amp (*U301A*) is not amplified, but simply transferred to the output. However, as the second op-amp implements a gain of two, its noise is amplified. Note that for this operational amplifier, only the total input voltage noise $\overline{(v_{\mathrm{n,i,op,in}}(f_b))^2}$ is specified.

$$\overline{(v_{\mathrm{n,i,out,opA,v}}(f_b))^2} = \overline{(v_{\mathrm{n,i,op,in}}(f_b))^2} \tag{3.15}$$

$$\overline{(v_{\mathrm{n,i,out,opB,v}}(f_b))^2} = \overline{(v_{\mathrm{n,i,op,in}}(f_b))^2} \cdot 2^2 \tag{3.16}$$

The noise contributions from the various gain resistors are as follows:

$$\begin{aligned}\overline{(v_{\mathrm{n,out,r,gain}}(f_b))^2} = 4kTR_{fb}f_b + \left(\frac{4kTR_{305}f_b}{4} + \frac{4kTR_{306}f_b}{4} + 4kTR_{307}f_b\right) \cdot 2^2 \\ + 4kTR_{308}f_b\end{aligned} \tag{3.17}$$

Where $R_{fb} = R_{302}$ is the feedback resistor.

Neglecting the noise contribution of the high-range shunt resistor and the switching MosFET, the overall output noise becomes:

$$\begin{aligned}\overline{(v_{\mathrm{n,lo,out}}(f_b))^2} = \overline{(v_{\mathrm{n,lo,out,src}}(f_b))^2} + \overline{(v_{\mathrm{n,out,opA,v}}(f_b))^2} + \overline{(v_{\mathrm{n,out,opB,v}}(f_b))^2} \\ + \overline{(v_{\mathrm{n,out,r,gain}}(f_b))^2} + \overline{(v_{\mathrm{n,adc,in}}(f_b))^2}\end{aligned} \tag{3.18}$$

If we transfer this noise voltage to the input current, we get:

$$\overline{(i_{\mathrm{n,lo,in}}(f_b))^2} = \overline{(v_{\mathrm{n,lo,out}}(f_b))^2} \cdot \frac{1}{R_{fb}^2} \tag{3.19}$$

Inserting the values for 64ksps and 1ksps into Equation 3.19, the figures are:

$$\overline{(i_{\mathrm{n,lo,in}}(f_b = 16.8\,\mathrm{kHz}))^2} = (1.47\,\mu\mathrm{A})^2 + \frac{4kTR_Sf_b}{(R_{inp} + R_S)^2} \tag{3.20}$$

$$\overline{(i_{\mathrm{n,lo,in}}(f_b = 262\,\mathrm{Hz}))^2} = (5.82\,\mathrm{nA})^2 + \frac{4kTR_Sf_b}{(R_{inp} + R_S)^2} \tag{3.21}$$

Note that the noise on this range is dominated by the input noise of the ADC, whereas the analog circuitry on its own has a noise floor of a mere 636 pA RMS input noise. The limiting factors are the nonlinearities in the amplifiers and ADC, as well as the drift of the circuit.

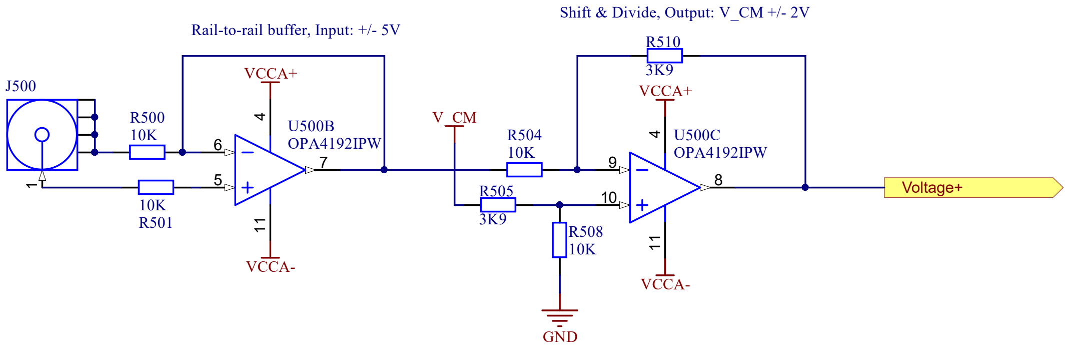


Figure 3.9.: Voltage channel schematic

3.4.3. Voltage Channel

Figure 3.9 shows the schematic for our voltage channels. In addition to the circuitry discussed in Section 3.2 the voltage is scaled with a gain of $g = \frac{R_{510}}{R_{504}} = \frac{v_{\text{out}}}{v_{\text{in}}} < 1$ to match the input voltage range of the ADC. We will also need to consider this additional circuitry in our noise analysis. The contribution of the op-amps and resistors are as follows:

$$\overline{(v_{\text{n,v,out,op,v}}(f_b))^2} = \overline{(v_{\text{n,v,op,in}}(f_b)} \cdot g)^2 + \overline{(v_{\text{n,v,op,in}}(f_b))^2} \tag{3.22}$$

$$\begin{aligned}(v_{\text{n,v,out,r}}(f_b))^2 &= 4kTf_b g^2 R_{501} + 4kTf_b(1+g)^2(R_{505}||R_{508}) \\ &+ 4kTf_b\frac{(R_{510})^2}{R_{504}} + 4kTf_bR_{510}\end{aligned} \tag{3.23}$$

From this we can determine the equivalent input noise and calculate numerical values for 64 kSPS and 1 kSPS.

$$\begin{aligned}\overline{(v_{\text{n,in}}(f_b))^2} = &\left(\overline{(v_{\text{n,v,out,op,v}}(f_b))^2} + \overline{(v_{\text{n,v,out,r}}(f_b))^2} + \overline{(v_{\text{n,adc,in}}(f_b))^2}\right) \cdot \frac{1}{g^2} \\ &+ 4kTf_b\frac{R_S \cdot (R_{inp,v})^2}{(R_{inp,v} + R_S)^2}\end{aligned} \tag{3.24}$$

$$\overline{(v_{\text{n,in}}(f_b = 16.8\,\text{kHz}))^2} \approx (3.77\,\text{mV})^2 + 4kTf_bR_S \tag{3.25}$$

$$\overline{(v_{\text{n,in}}(f_b = 262\,\text{Hz}))^2} \approx (15.1\,\mu\text{V})^2 + 4kTf_bR_S \tag{3.26}$$

Where R_S is the source resistance and $R_{inp,v} \approx 1\,\text{T}\Omega$ is the input resistance of our voltage channel.

3.5. Cape

Some additional hardware is required to operate the channels and the ADC. A total of four positive and negative low-noise voltage rails is generated from the BeagleBone system voltage. The BeagleBone cape standard also requires the addition of an identification memory and some switches, in order to identify the cape and its power and pin demand. A PCB containing all these circuits was designed and manufactured.

4. Software Design

4.1. Overview

An overview over the entire system can be found in Figure 4.1. One can see the cape, which was thoroughly analysed in Chapter 3, as well as the BeagleBone. The latter is used for the control of the sampling process and the data managing. As stated in 1.2, a main requirement for the logger is the mobility. Therefore, the BeagleBone cannot have a mains power connection, but needs to be powered by a battery. As, in addition, the sampling process should by controllable by a remote user interface (UI), the BeagleBone needs to have a network connection as well.

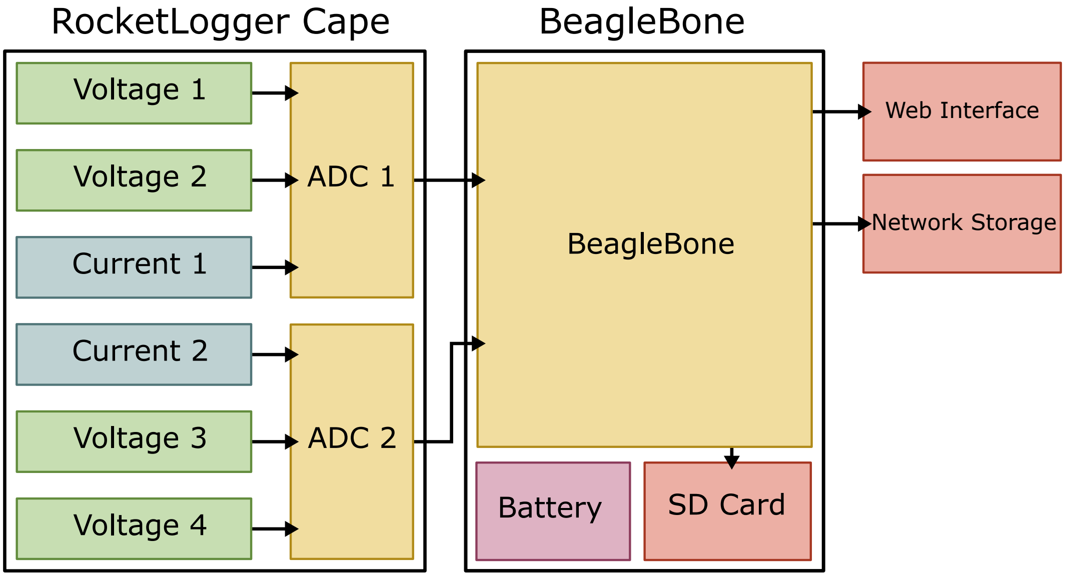


Figure 4.1.: System overview

The software is implemented on a Debian distribution for the BeagleBone. This allows a simple implementation, particularly of the remote UI.

Before explaining the main software stack, the interface to the cape and its requirements will be analysed.

4.1.1. Interface

The interface between the BeagleBone and the cape consists of three main connections, and is shown in Figure 4.2. The serial peripheral interface (SPI), which is used for the data acquisition, will be explained in Section 4.1.1. The ADCs of the cape should operate synchronously to ensure that all channels are sampled at the same time. Therefore, an option is to provide them with an external clock signal. To have the ability of forcing the current channels to the high range (see Section 3.1.3), digital range control signals are needed. The detailed implementation of these interfaces will be described in Section 4.2.1

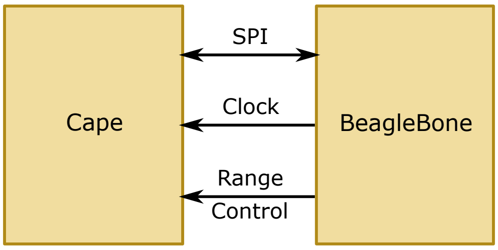


Figure 4.2.: Cape interface

SPI

A more detailed view on the SPI between the BeagleBone and one ADC can be found in Figure 4.3. One can see that in addition to the four normal SPI wires at the bottom, there are two separate control lines. The *start* wire is used to initiate the sampling of the ADC, which then notifies the BeagleBone with the *data_ready* wire, whenever it has converted a new sample. The latter then can fetch the data via the SPI, where it acts as the master, since the ADCs only support slave mode. This procedure guarantees that sampling and data acquisition are synchronous and the BeagleBone always receives valid data.

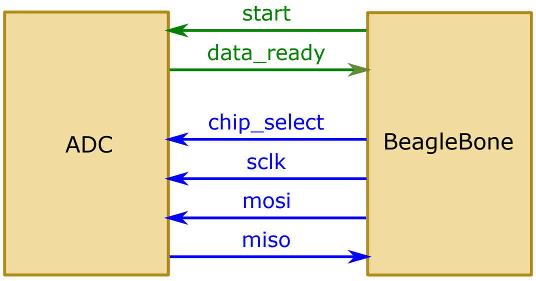


Figure 4.3.: SPI interface

4.1.2. Basic Tests

In order to test the connection to the ADC, some basic tests were done using the multichannel serial port interface (McSPI) of the BeagleBone's ARM processor. This hardware module can easily be configured via the Linux *spidev* driver and controlled via the *ioctl* function, which manipulates the underlying device parameters of the generated *sysfs* special file. The ADC control signals were connected to GPIOs of the BeagleBone, which can be accessed in a similar way as the McSPI module, using the *gpiolib* driver and the *fcntl* function. The test setup is shown in Figure 4.4.

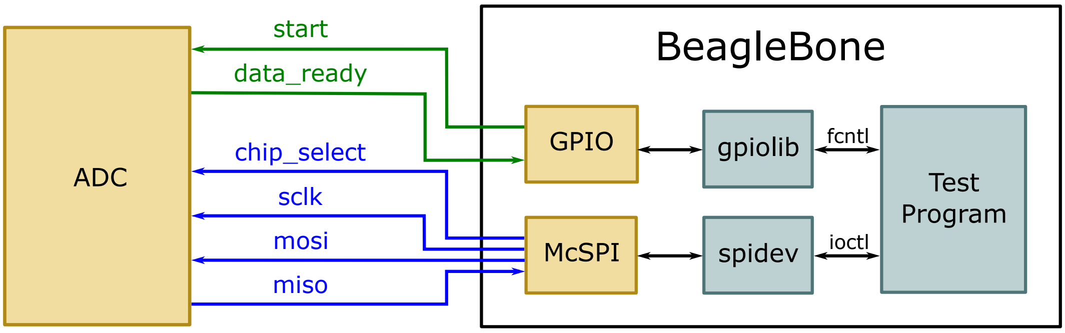


Figure 4.4.: Basic SPI test

This setup is rather simple, but has some severe drawbacks:

- Since the ADCs do not have a buffer, we need to collect the sample before the next conversion has finished. Else the sample will be lost. As mentioned above, the software however needs to wait for the *data_ready* signal, before it can get the sample. Therefore, the signal needs to traverse all the layers up to the user space test program, which then has to notify the McSPI module. Since a non-real-time OS is running on the BeagleBone, the delays and the response times are unpredictable, and can take up to some hundreds of milliseconds. During practical experiments it could be seen that it is not possible to sample with rates larger than 1 kSPS, without losing samples. This however is far below the capabilities of the ADCs.
- The entire sampling control is done on the CPU and thus it is busy doing this low-level control. Therefore, it does not have resources for the tasks of the higher layers, which mainly comprise the processing and storing of the data.
- In high rate operation one is not able to control all timing parameters as precisely as required by the ADCs.

An approach to solve these problems would be to write a custom Linux driver, which does the control in the kernel space directly. This would certainly lower the response times, they will however not be exactly predictable. The maximal sampling rate of the

ADCs is $f_{max} = 64\,\text{kSPS}$, which results in a sampling period of only $T_S = \frac{1}{f_{max}} \approx 15.6\,\mu\text{s}$. As this cannot be guaranteed by the non-real-time OS, another solution has to be found. The one that was implemented in this thesis will be explained in the following section.

4.2. Software Layers

The software is organized in several layers. The entire stack can be seen in Figure 4.5 and will be explained in a bottom-up approach.

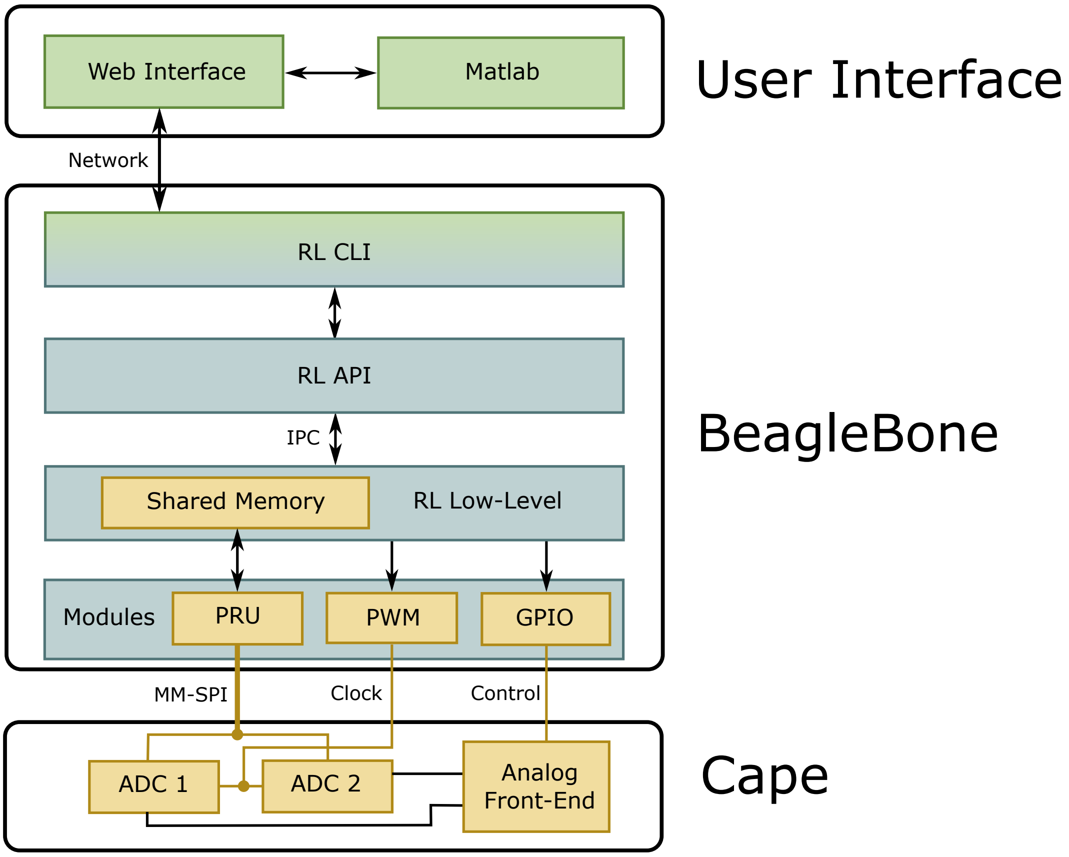


Figure 4.5.: Software overview

4.2.1. Modules

The lowest layer consists of the hardware modules which handle the interface to the cape. These need to be enabled using device tree overlays (DTOs) and are configured and controlled by the low-level layer. The three modules will be explained separately in the following.

GPIO

The GPIO module is needed to control the range forcing signals of the cape. As these are fixed for the whole measurement, they are not time-critical and we do not have to bother about the delays mentioned in Section 4.1.2. Therefore, the Linux *gpiolib* driver can be used in the same way, as it was done for the control signals in the basic tests.

PWM

For the generation of the external ADC clock, a pulse-width modulation (PWM) module of the BeagleBone's processor can be used. It is set up to generate a square signal with the desired ADC clock frequency $f_{adc} = 2.048\,\text{MHz}$. The configuration is very similar to the one of the GPIO module, using the Linux *sysfs* interface.

SPI

The data interface is the most difficult part to implement. As mentioned before, it is very time-critical, and we cannot use software to react on the *data_ready* signal. Unfortunately, the McSPI does not have the ability to react on GPIOs directly. Therefore this module cannot be used.

Another idea was to use the multichannel audio serial port (McASP) module of the BeagleBone. This is a general-purpose audio port, which has the ability to respond to synchronization signals and thus could fit our requirements. Unfortunately, it does not support SPI but only specific audio protocols, and cannot be used either.

The solution that was chosen in this thesis, is to use the programmable real-time unit (PRU).

PRU The PRU is a 200 MHz co-processor with a 32 bit RISC architecture. It needs to be programmed with its own assembly instruction set, has fixed execution times for most instructions and direct access to some GPIOs. It therefore perfectly fits our real-time requirements.

The entire data interface, including acquisition and management, is implemented on the PRU. One cannot use the SPI out of both the user space and the PRU, since the BeagleBone's pin muxes cannot be changed at run time. This implies, that the basic configuration of the ADCs, which needs to be set after every reboot, has to be done in the PRU as well.

Implementing the entire SPI protocol on the PRU seems to be quite involved, but comes with several convincing advantages:

- Due to the fixed execution times and the direct GPIO access, on has tight control over all timings. This makes it possible to have extremely low response times.
- The PRU has access to the main memory, and can communicate with higher level programs via interrupts. Thus the sampled data can be written to the RAM without any CPU interaction, and the CPU is free for other tasks.
- Programming the SPI protocol from scratch allows to have a highly specialized and more efficient setup. The timings can be optimized for the requirements of the ADCs. In addition, it allows a simple solution to the problem discussed in the next section.

ADC Arrangement

Since two ADCs are needed to sample all channels, a good arrangement has to be found to read the values of both. With some basic calculations on the required data rate one can estimate if a serial read-out is possible. The total data rate on the SPI is

$$R_{SPI} = 2 \cdot (N_{channels} N_{bits} + N_{status}) \cdot f_{dr} \approx 13\,\text{Mbit/s}, \tag{4.1}$$

where $N_{channels} = 5$ is the number of channels per ADC, $N_{bits} = 16$ is the precision of the samples at highest data rate $f_{dr} = 64\,\text{kSPS}$, and $N_{status} = 24$ is the number of status bits that are sent during each SPI transaction.

As the ADCs allow a maximum SPI clock frequency of 20 MHz, serial read-out should be possible. However, Equation 4.1 does not consider the time to switch the *chip_select* lines, and the additional time used in the PRU for management and to store the values. Since this time was not exactly known yet at the point where the PCB needed to be ordered, it was decided to look for another solution.

A possibility would be to put the two ADCs in a daisy-chain configuration and read them out serially. This would ease the implementation, since we would not have to switch between the two. Unfortunately, this configuration has some additional timing requirements and therefore this approach was dropped.

Another possibility would be, to just use two different SPIs for the two ADCs, and read them out in parallel. The problem with this solution is the synchronization between the two interfaces. It might be difficult to finally get all values into correct order. Since a much simpler solution exists, this one was abandoned as well.

In the approach we chose in this thesis, the fact that the entire SPI is implemented in assembler anyway is exploited. We could use a slightly modified version of the interface, which we called a multi-MISO SPI (MM-SPI). Instead of only one shared *miso* wire, we connect the data lines of both ADCs to the PRU separately, whereas the other SPI wires (except the *chip_select*) remain shared. Thus it was possible to read them out in parallel, without losing synchronicity.

4.2.2. Low-Level

After the implementation of the cape interface, we now can move up to the low-level layer. This layer is a C-program that is responsible for the configuration and the control of the three hardware modules. Setting up the GPIO and the PWM module is straightforward and can be done as explained in Section 4.1.2.

For the PRU there is the PRUSSDRV application programming interface (API) provided by Texas Instruments. This is a user space library with basic control functions that use the *uio_pruss* driver to access the PRU. It makes it possible to load code and data into the internal memory of the PRU and to start the execution. In addition to this rather simple interface, a data passing method, that allows the PRU to continuously write out the acquired data, had to be designed.

Data Passing

Since the PRU has access to the main memory, the data passing was implemented using a switched-buffer method on shared memory. The setup can be seen in Figure 4.6.

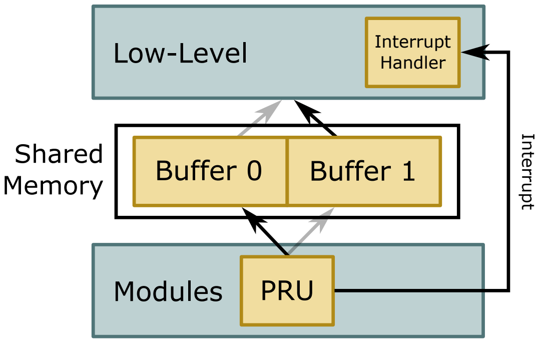


Figure 4.6.: Switched buffer data passing

The memory, that is assigned to the kernel module of the PRU, is split into two buffers, which the PRU can write alternately. Using the Linux *mmap* function, these memory buffers can be mapped into the user space of the low-level C-program, where the data can be read and passed on. To ensure synchronization between the two layers, the PRU emits an interrupt every time a buffer is full, and continues writing to the other one. The C-program catches this interrupt and reads the full buffer. In addition, the buffers get numbered by the PRU, which allows the upper layer to perform an overrun check. When the reading is too slow, the low-level detects a discontinuity in the buffer numbering and can notify the user that samples were lost.

Data Storing

After a buffer is read from the shared memory, the samples run through a basic processing step, where the offsets and scales are corrected. More details on this calibration can be found in Section 4.4.

Afterwards, the buffer gets written to a file on the SD card. In a first approach, it was decided to store the values in a comma-separated values (CSV) format, since it is human readable and eases debugging. However, tests at high data rates showed that the conversion to text is too slow and buffers are lost. Therefore, an option to store the values in binary format was included, which allows the logging of all channels with full sampling rate.

In addition to the sampled values, the ADCs also transmit, if the low range current channels are valid or not. The PRU stores this information as well.

To connect the low-level sampling procedure to the UI, another layer, which will be discussed in the following section, had to be added.

4.2.3. API

The purpose of the API software layer is, to provide the user with a C-library, with some basic functions to control the sampling process. This interface can for example be used for UI implementations. It has some basic modes and functions, which will be briefly explained in the following. More details can be found in Appendix A.

Since the logging is expected to run standalone and over a longer period, the sampling process should run continuously and in background. Therefore, an interface had to be implemented, which allows the API to communicate with the background low-level process. An overview over the different functions and the appropriate inter process communication (IPC) methods can be found in Figure 4.7 and will be explained in the following.

Sample The *sample* function is the main function to start the sampling. It starts the low-level process in background, stores its process ID, the current mode, and all configuration to a shared memory. It can be used with different modes:

- Sample-limited: This mode acquires a limited number of samples and stops automatically.

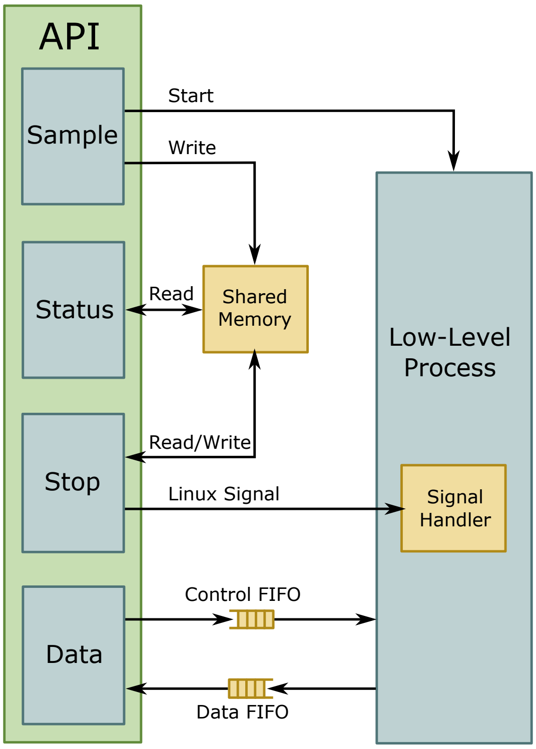


Figure 4.7.: Inter process communication

- Continuous: This mode starts the continuous sampling in background. The sampling can be stopped with the *stop* function.
- Meter: This mode does not store the samples to the file, but prints the current values directly on the console.

Status The *status* function is used to check the current state of the RocketLogger. It therefore reads and returns all configuration from the shared memory.

Data The *data* function is used to get the instantaneous values, when the sampling is running continuously. It can, for example, be used to implement a UI which should also provide some kind of visualisation of the current measurement. As can be seen in Figure 4.7, it is implemented using two first-in first-out (FIFO) buffers. These FIFOs are Linux special files and can be generated using the *mkfifo* function. Using the *control* FIFO, the *data* function notifies the background process, that it is waiting for data. Afterwards it blocks while trying to read the *data* FIFO. When storing the next buffer, the low-level program stores the current values in the *data* FIFO, if requested, and the *data* function can continue. This implementation has some advantages:

- The blocking reading of the *data* function implies some kind of synchronization between the two layers. A UI can simply trigger a data request and will always get the correct data only once. This avoids the overhead of an asynchronous implementation, where the UI has to do oversampling, to be sure to get all the data.
- The FIFO special file is not a real file in the file system, but the data is passed internally by the kernel. Therefore, this approach does not use the SD card.
- Due to the two FIFOs, the low-level program only writes out the values, if they are requested. This ensures that one always gets the current data. This is an advantage for remote UIs, since they can be closed and reopened at run-time and can have unpredictable delays.

Stop To stop the continuous sampling, one can use the *stop* function. Using the process ID, stored in the shared memory, it sends a Linux signal to the background process. The signal handler triggers the shutdown of the sampling process in the PRU, the closing of all modules, and the storing of buffered data to the file. Additionally, it also updates the shared memory with the new state.

On top of this software stack, a UI, which will be explained in the following, is implemented.

4.3. User Interface

The uppermost part of the RocketLogger software is the UI. As stated in Section 1.2, the sampling should be remotely controllable. However, it was left open to decide, whether the software of the existing ACME device (see Section 1.4), or a custom solution should be used. The main advantage of the ACME solution is its integration into an open source signal analysis software suite, called *sigrok*. It includes a command line interface (CLI) and a graphical oscilloscope.

To integrate a measurement device into *sigrok*, one would have to write the appropriate drivers. These were analysed in this thesis and an integration should be possible. Unfortunately, this software suite did not match our requirements. For example, it is not possible to run continuous measurements with its graphical UI. Therefore, it was decided to implement a custom UI, which consists of the following three parts.

4.3.1. CLI

As a basic and simple interface, a CLI was implemented. It gives the user access to the different API modes and functions explained in Section 4.2.3. In addition, it also allows to configure the following options:

- File format and name
- Channel selection
- Sampling rate

This CLI can be used as a UI on its own and serves as a basis for the graphical interface.

4.3.2. Web Interface

The graphical UI was decided to be realized as a web application, because of its simplicity and ease of use. It is implemented using HTML and JavaScript on the client-side, whereas a small PHP script is used for calling the RocketLogger CLI functions on the BeagleBone. It supports the continuous sampling mode, the different configurations stated above, and the direct download of the data file.

In addition, it also provides a basic visualisation of the ongoing measurements. In two different, one windows for voltages and one for currents, the values of the last ten seconds of the active channels are plotted. This allows the user to get an impression of the sampled data and to check, whether the measurement setup is correct. It was implemented using the *Flot* JavaScript plotting library[1].

4.3.3. Matlab

Matlab was used for data analysis and post-processing. The implemented functions can be used to automatically read in and plot values from CSV and binary files. The final merging of the three current channels is also done in Matlab. An explanation of the different functions can be found in Appendix A.

4.4. Calibration

To correct the offset and gain errors of the measurement front-end, a calibration has to be done for every device. All current and voltage channels were designed to be linear. Therefore, it was decided to implement a first-order calibration, where only an offset and the linear scaling are adapted. As an implementation in the C-code turned out to be cumbersome, Matlab was used again.

To calibrate the device, one has to reset the calibration via the CLI, or the web interface, and log a predefined sweep for every channel. The data file can then be read into Matlab, which automatically recognizes the steps and averages them. Afterwards it calculates

[1] http://www.flotcharts.org/

the offset and scale of the best linear fit and stores them into a file, which finally can be moved onto the BeagleBone, where it is read in at the start of each measurement.

The evaluation of the calibration can be found in Chapter 5.1.1.

5. Evaluation

The final task during our semester project was the evaluation of the completed RocketLogger. First we will present the DC and AC characteristics of the device, measured against lab instruments. The accuracy of those devices exceeds the desired accuracy for the RocketLogger and we can neglect their errors for most measurements. The second part consists of some example measurements, captured using the RocketLogger. Note that the erratum discussed in Section A.9.1 significantly deteriorates the medium- and long-term stability of the device. Therefore, the accuracy measurements were made shortly after a calibration.

5.1. DC Accuracy

Figures 5.1 and 5.2 show the DC accuracy and noise of the RocketLogger for the voltage and current channels respectively. As a reference the Keithley *2450* source measure unit (SMU) or the Keysight *34470A* multimeter was used. We were unable to conduct all the measurements with the Keithley *2450* SMU due to its significant 50 Hz noise. Instead, we used a lab power supply with a shunt resistor as a source, when necessary. The RMS noise corresponds to 6.5 nA and 7.85 µV respectively. Both channels exceed our original design goals, especially in terms of noise floor.

5.1.1. Calibration

The performance of the calibration can be found in Figure 5.3. It displays the remaining integral nonlinearity of a voltage channel, which is mainly caused by the ADC. It was measured using the Keithley *2450* SMU and a sweep over the entire range. The nonlinearity could be compensated in software by using a higher order calibration.

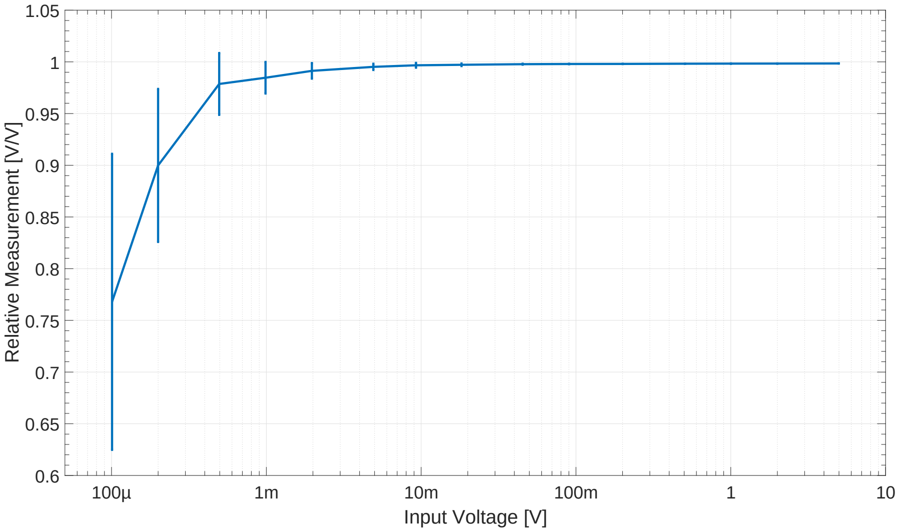


Figure 5.1.: Voltage channel DC accuracy and noise at 1 kSPS. The bars represent the ±1σ interval. A perfect measurement would correspond to a flat line at 1.

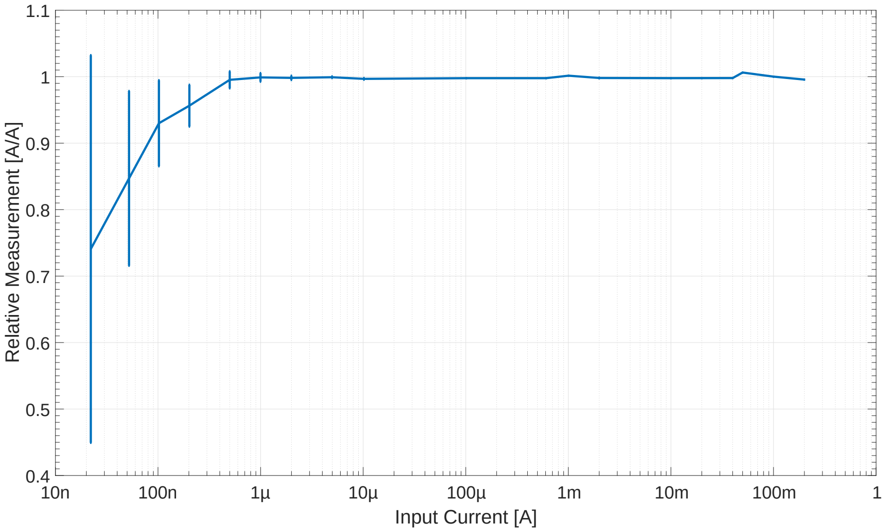


Figure 5.2.: Current channel DC accuracy and noise at 1 kSPS. The bars represent the ±1σ interval.

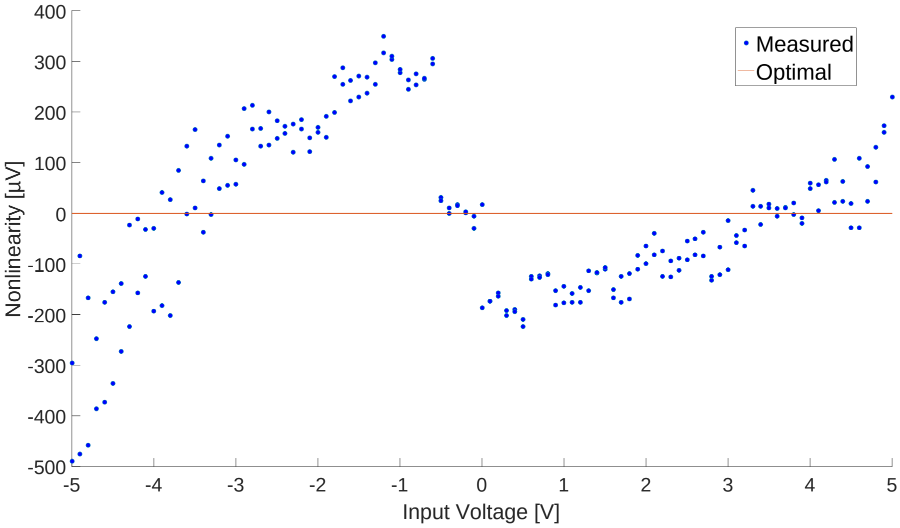


Figure 5.3.: Integral nonlinearity

5.2. AC Performance

In order to measure the AC performance of the logger, we applied a frequency sweep using the Agilent *33600A* waveform generator. The envelope of the logged signal then corresponds to the frequency response of the RocketLogger. At 64 kSPS the voltage channel reaches −3dB attenuation at 10 kHz, while the current channel reaches its −3dB point at 8 kHz. For the lower sampling rates, the cut-off frequency corresponds to that of the digital filter in the ADC.

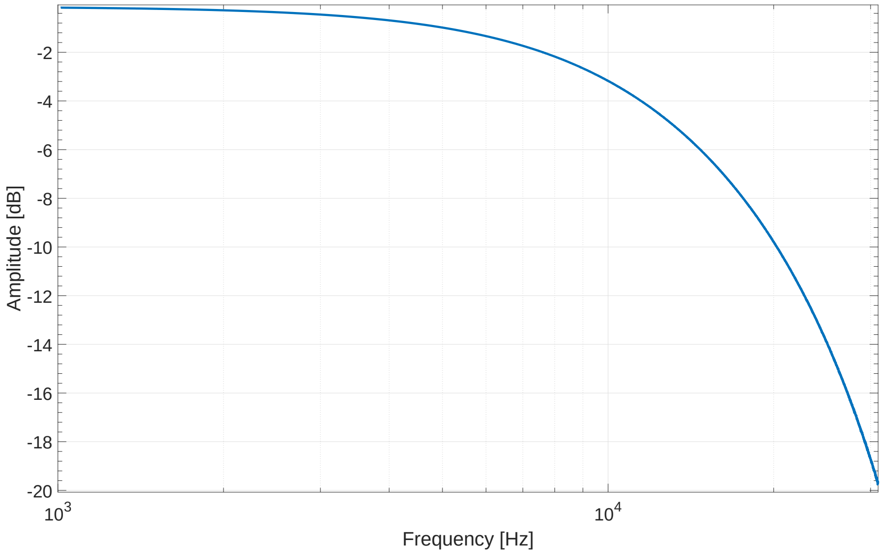


Figure 5.4.: Voltage channel transfer function at 64 kSPS.

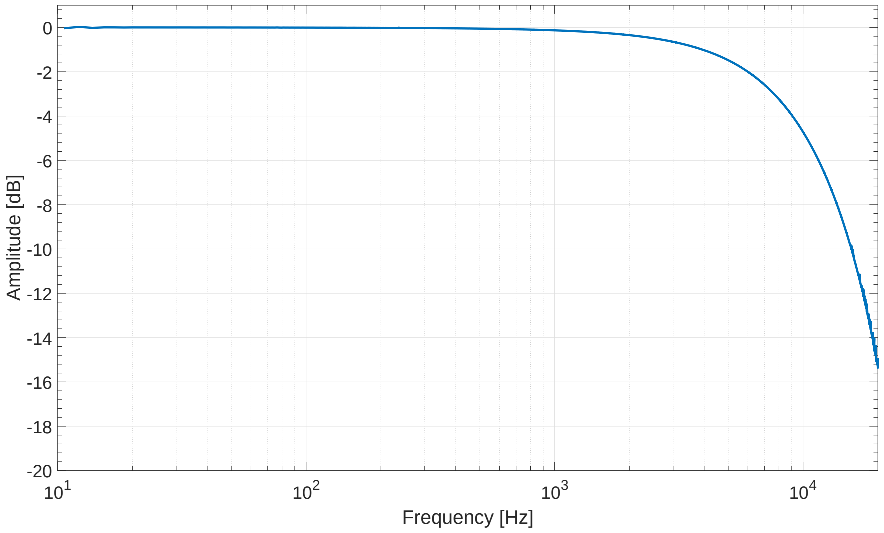


Figure 5.5.: Medium range current channel transfer function at 64 kSPS.

5.3. Burden Voltage and Range Switching

The DC burden voltage over the input current range was measured using the Keithley *2450* SMU. As can be seen in Figure 5.6, it almost perfectly corresponds to a resistance of 94 mΩ.

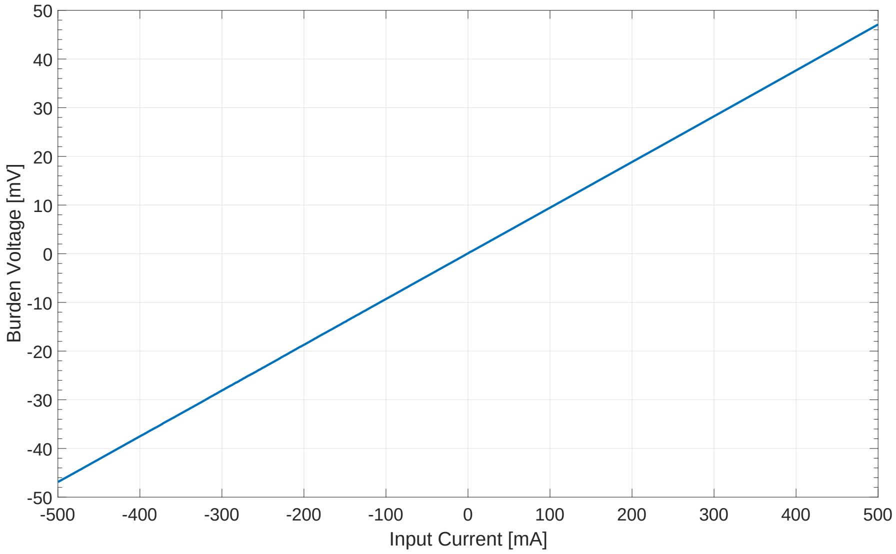


Figure 5.6.: Burden voltage over the input current range.

The trace of the burden voltage during the range switching, shown in Figure 5.7, allows us to evaluate the effect on the measured circuit. All traces show a transition from zero to a certain current in the high range. Once the input current is connected, the burden voltage rises, since the feedback current meter cannot keep up with the current change. The diodes limit the burden voltage in this phase to a maximum 200 mV. For low currents $\leq$ 5 mA the amplifier eventually catches up. However, after about 1.2 µs the range switch starts and deactivates the feedback current meter. Following an initial negative voltage pulse[1], the voltage converges to the resistive behaviour after about 1.5 µs.

[1] The root cause of this behaviour was not found, but probably lies with some capacitive or inductive behaviour on the RocketLogger. The negative voltage step coincides with the start of the discharge of the low range MosFET. The high range MosFET gate charging starts some 10s of nanoseconds later.

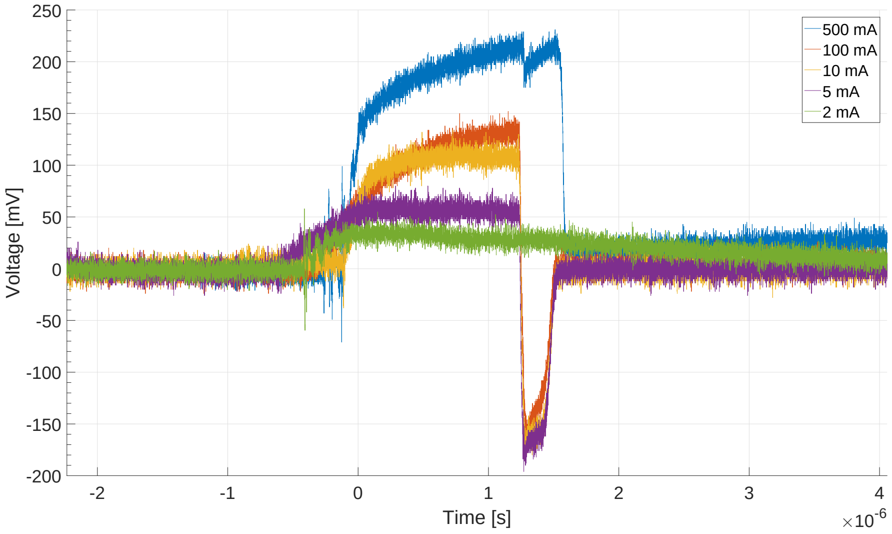


Figure 5.7.: Burden voltage during range switching. Note that the negative voltage spike for 2 mA is delayed, due to the lower amplitude.

5.4. BeagleBone Black Supply Measurement

Figure 5.8 shows the power consumption of the BeagleBone during sampling. This measurement was taken with the active RocketLogger attached to the BeagleBone itself. Note that the supply current of the RocketLogger is not included, even though its supply voltage is also measured. For this measurement the two boards were connected to separate power supplies, although they usually share a single one.

The BeagleBone handles a buffer every second and then writes it to the SD card. The corresponding current burst at every full second is nicely visible. At the same time the supply voltage drops, due to the resistance of the USB cable to the power supply. The measured power varies from 1.07 W to 1.18 W, with an overall average of 1.12 W.

5.5. Harvesting Blinky

As a real-world evaluation, we measured the harvesting node example explained in Section 1.1. Due to the flashing LED simulating the load, we called it the harvesting blinky. Figure 5.9 shows the storage voltage and the load current. Every time the LED[2] is turned on, one can see a current of about 3 mA, and a decreasing storage voltage.

[2] A LED would not draw enough current, to force the channel to switch out of the low range. Therefore, it has been replaced by a simple resistor during these measurements.

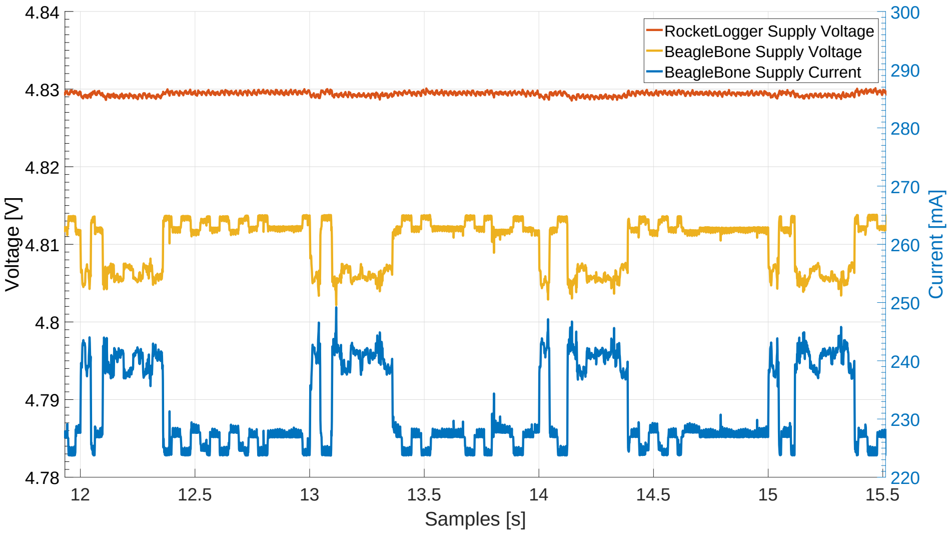


Figure 5.8.: BeagleBone Black supply measurement.

In between these bursts, the load only consumes about 400 nA of sleep current. The figure shows the large dynamic range as well as the fast range-switching capability of the RocketLogger. The small current spikes of about 1 µA are caused by the sleep timers of the microcontroller.

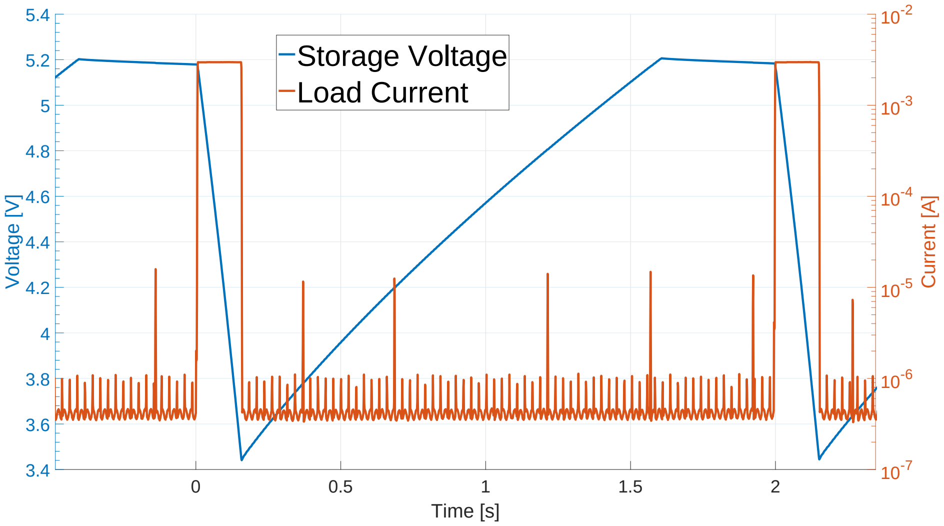


Figure 5.9.: Harvesting blinky

If we want to investigate the larger current spikes that occur during the sleep phase of the microcontroller, we can do a full measurement. Figure 5.10 shows a plot, where all channels of the RocketLogger are used at the same time. In addition to the previous setup, we also measure the solar current, the LED voltage, the load supply voltage, and the solar voltage. It can be seen that every time there is a step in the supply voltage of the load, there is a spike of about 10 µA in the supply current. These are caused by the voltage regulation on the EMU.

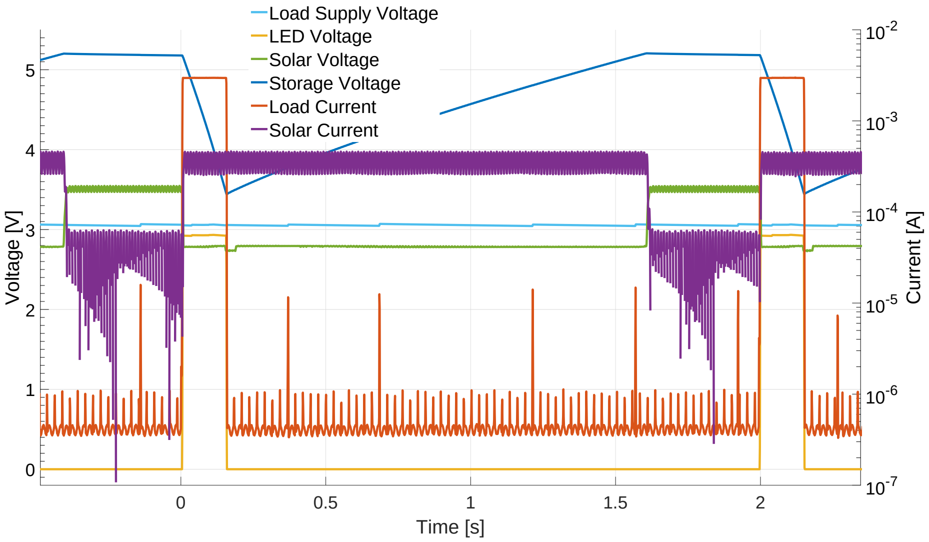


Figure 5.10.: Harvesting blinky: full measurement

5.6. Performance

	Minimum	Typical	Maximum
Sampling Rate [a]	1 kSPS		64 kSPS
Analog Bandwidth [a]	262 Hz		10 kHz
Number of Samples			2^{31}
Voltage Channel Input Range	−5.5 V		5.5 V
Voltage Channel RMS Noise Floor [b]		7.97 µV	
Voltage Channel RMS Noise Floor [c]		1.38 mV	
Short-Term Voltage Channel DC Accuracy [e]		0.05 % + 200 µV	
Voltage Channel Input Impedance		1 TΩ	
Current Channel Input Current	−500 mA		500 mA
Current Channel HI RMS Noise Floor [b]		1.27 µA	
Current Channel HI RMS Noise Floor [c]		65.2 µA	
Short-Term Current Channel HI DC Accuracy [e]		0.5 % + 200 µA	
Current Channel MID Input Current	−40 mA		40 mA
Current Channel MID RMS Noise Floor [b]		1.24 µA	
Current Channel MID RMS Noise Floor [c]		9.22 µA	
Short-Term Current Channel MID DC Accuracy [e]		0.01 % + 3 µA	
Current Channel LOW Input Current	−2 mA		2 mA
Current Channel LOW RMS Noise Floor [b]		5.42 nA	
Current Channel LOW RMS Noise Floor [c]		537.8 nA	
Short-Term Current Channel LOW DC Accuracy [e]		0.01 % + 20 nA	
DC Input Resistance		94 mΩ	
Range Switching Time		1.5 µs	
Transient Burden Voltage [d]		200 mV	
Power Consumption		1.8 W	
Output Data Rate			10.8 GB/h

[a] Using the binary file format
[b] With a sampling rate of 1 kSPS
[c] With a sampling rate of 64 kSPS
[d] Input current step: 0 mA to 500 mA
[e] Consider the erratum discussed in Section A.9.1

Table 5.2.: RocketLogger performance.

6. Conclusion

We have examined the requirements to log the power consumption of energy harvesting devices. Using this information, we have evaluated the suitability of multiple current measurement topologies. We have designed, built, and tested the RocketLogger, a portable measurement instrument for logging the power consumption of mobile, energy harvesting devices.

The RocketLogger can measure voltages up to ±5.5 V and currents up to ±500 mA. It supports sampling rates ranging from 1 kSPS to 64 kSPS, and has a maximum analog bandwidth of 10 kHz. At 1 kSPS the voltage and current channels have an extremely high dynamic range of 123 dB and 165 dB respectively. This is made possible by the fast range switching, which was implemented in the analog front-end.

The software, running on the BeagleBone, is capable of processing and storing the data in real-time. To ensure that no samples are lost, a custom interface was implemented on the PRU. The logging can be started and stopped using a web interface, which also displays the most recent measurements. The data is stored on the SD card of the BeagleBone and can be downloaded for analysis in Matlab.

Once some of the issues discovered during testing are resolved, the RocketLogger should prove to be a valuable instrument for analysing the power consumption of upcoming energy harvesting devices.

A. RocketLogger User's Guide

A.1. Feature Description

The RocketLogger is a portable voltage and current logger, designed to characterise energy harvesting devices. It has a total of four voltage channels and two auto-ranging current channels.

The RocketLogger can measure voltages up to ±5.5 V and currents up to ±500 mA. It supports sampling rates ranging from 1 kSPS to 64 kSPS, and has a maximum analog bandwidth of 10 kHz. At 1 kSPS the voltage and current channels have an extremely high dynamic range of 123 dB and 165 dB respectively. This is made possible by the fast range switching, which was implemented in the analog front-end.

The software, running on the BeagleBone, is capable of processing and storing the data in real-time. To ensure that no samples are lost, a custom interface was implemented on the PRU. The logging can be started and stopped using a web interface, which also displays the most recent measurements. The data is stored on the SD card of the BeagleBone and can be downloaded for analysis in Matlab.

A.2. Typical Application

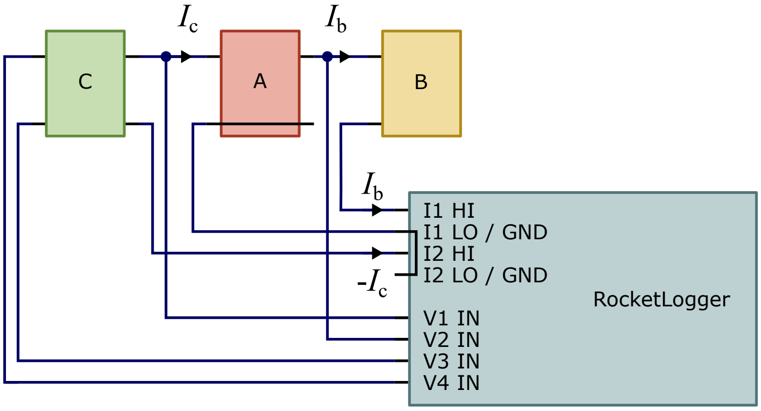


Figure A.1.: Typical application

A.3. Absolute Maximum Ratings

	Minimum	Maximum
Supply Voltage	−0.3 V	5.5 V
Current Channel Input Current		±1 A
Voltage Channel Input Voltage		±25 V
Voltage Channel Input Current		±10 mA

A.4. Recommended Operating Conditions

	Minimum	Maximum
Supply Voltage	4.75 V	5.25 V
Current Channel Input Current		±500 mA
Voltage Channel Input Voltage		±6 V

A.5. Performance

Refer to Section 5.6.

A.6. Important Considerations

- The *LO* (outer conductor) terminal of both current channels are shorted together and to the internal ground of the RocketLogger. The resistance between both terminals is $< 3\,\mathrm{m}\Omega$ and not protected against overload. If the BeagleBone is connected to mains earth, these connectors will be as well.
- The *HI* (inner conductor) terminals of the current channels are virtually connected to the grounds. These inputs are fused with a *046702.5NR* 2.5 A fuse.
- The voltage inputs (inner conductor) have an extremely high input impedance, but only as long as the inputs stay within ±6 V of the ground.
- The voltage guards (outer conductor) are connected to the output of the buffer amplifier. Leave these floating.
- Only use probes where both connectors are wrapped in isolating material

A.7. Measurement Recommendations

- Only connect one current *LO* port to the same potential to avoid ground loops.
- Use low resistance cables for high current measurements.
- Use a battery or an isolated power supply to reduce the effect of 50 Hz noise.
- Use a sampling rate of at most 16 kSPS for the best ADC performance.
- Digital interfaces and other measurement equipment can draw significant currents. Disconnect them if necessary.
- For input voltages outside the allowed voltage range, the voltage channels have an (differential) input resistance of 10 kΩ.

A.8. Input Structure

Figures A.2 and A.3 show the input structure for the current and voltage channels of the RocketLogger.

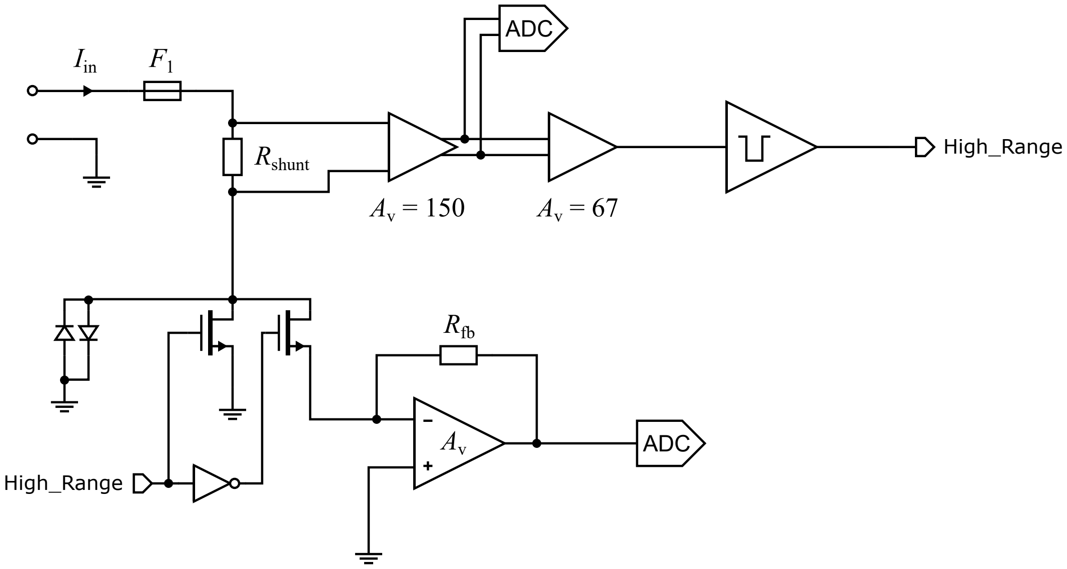


Figure A.2.: RocketLogger current channel diagram

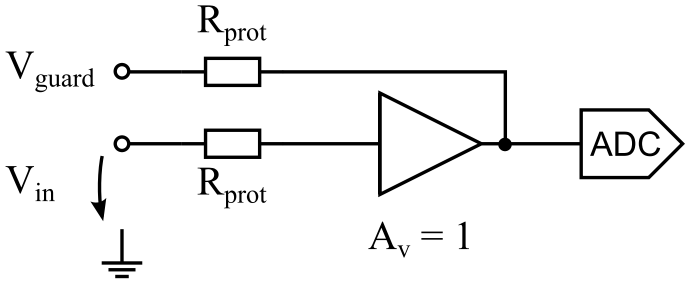


Figure A.3.: RocketLogger voltage channel diagram

A.9. Errata

A.9.1. High Drift of Readings

The drift of the measurements over time and temperature is on the order of 2 %, while the expected drift should be orders of magnitude lower. Figure A.4 shows the drift over one night.

Probable cause: The reference voltage drift of the ADC is much higher than specified. Measurements indicate a temperature coefficient of about 170 ppm, while the datasheet specifies 8 ppm as a typical value. The reference voltage is also outside the specified minimum and maximum range.

Workaround: None, compensate for temperature in software.

Fix: Use a better voltage reference.

A.9.2. High Current Non-Linearity

Measurements of positive currents, higher than 450 mA are not accurate. The error is higher for values very close to 500 mA and does not occur for negative currents. The measurement error is plotted against input current in Figure A.5.

Probable cause: The AD8421 is not entirely linear over its specified output voltage range. For high, positive currents we are approaching the higher limit on the output voltage and the in-amp can no longer drive the load sufficiently.

Workaround: Use a negative current measurement, or compensate in software.

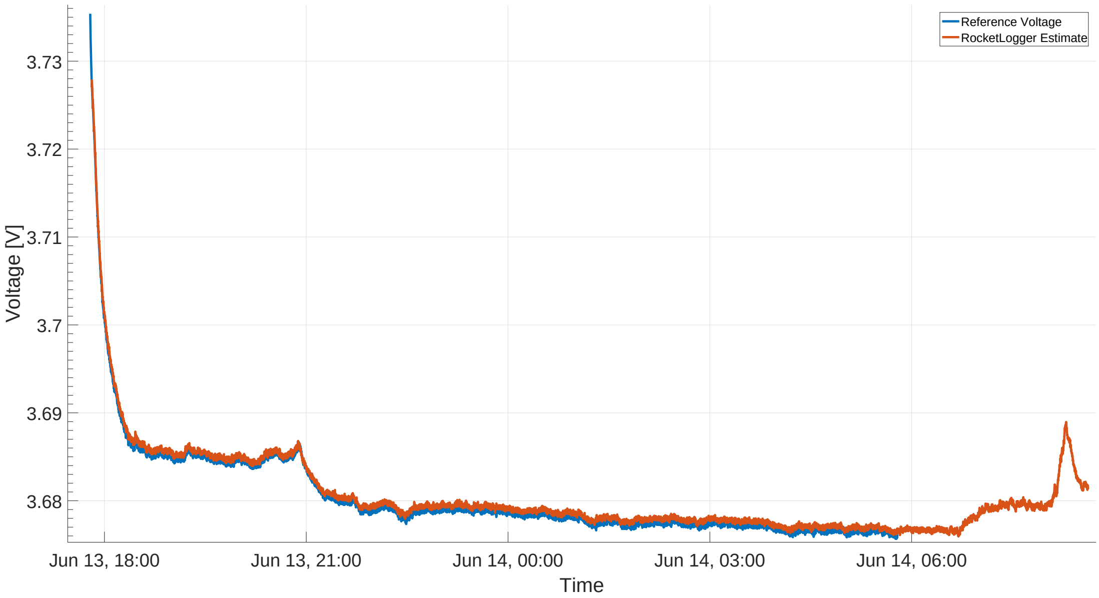


Figure A.4.: Voltage reference temperature drift. The step at 21:00 corresponds to a temperature difference of about 10 °C and a reference voltage difference of about 1700 ppm. The scaled inverse of the RocketLogger measurement of a constant voltage is shown as an indication of the impact. The error of the measurement is about 15 mV or 1.5 %.

A.9.3. 5V Digital Regulation

Relatively large current glitches occur on *all* current ranges. They are correlated with a drop in the digital 5 V power supply.

Cause: The 5 V system supply from the BeagleBone is unregulated and large voltage drops occur for a long time. The voltage drop is coupled into the current measurement and logged. The supply for the digital circuitry on the RocketLogger should be regulated and free of these voltage spikes.

Workaround: Use a separate 5 V power supply for the RocketLogger.

Fix: Use an additional voltage regulator to power the digital 5 V circuitry.

A.9.4. TCR2EE Footprint

The footprint for the TCR2EE 1.2 V voltage regulator is mirrored.

Workaround: Short pins 4 (VOUT) and 5 (NC) on the PCB.

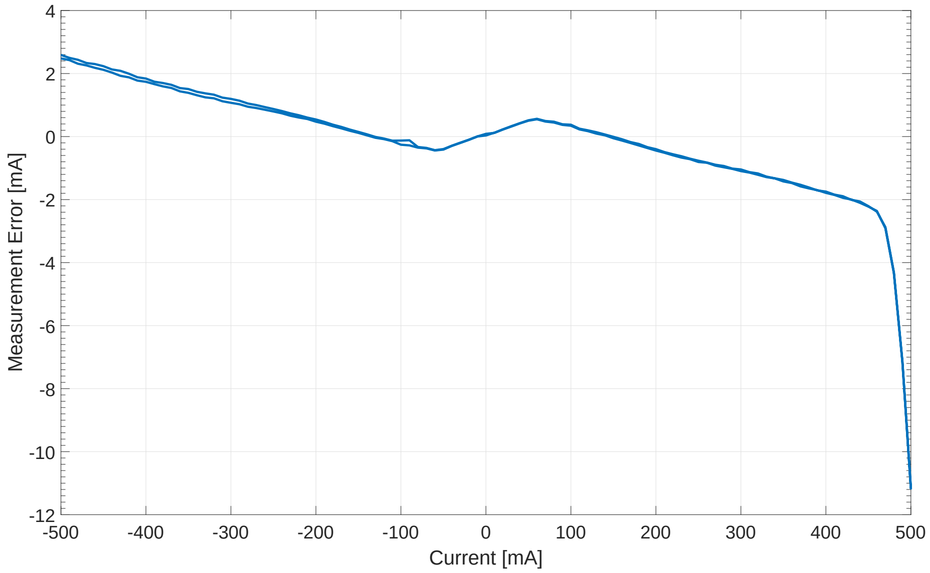


Figure A.5.: Error on the high current range

A.10. API Usage

This section describes the usage of the RocketLogger API. It is a C-library with the following functions:

Sample

rl_sample ret = rl_sample(conf);

Description:	Starts the logging with a fixed amount of samples.		
Argument:	conf	rl_conf struct	Configuration
Return:	ret	integer	Returns 1, if successful.

rl_continuous ret = rl_continuous(conf);

Description:	Samples in background until stopped.		
Argument:	conf	rl_conf struct	Configuration
Return:	ret	integer	Returns 1, if successful.

rl_meter ret = rl_meter(conf);

Description:	Starts the RocketLogger Meter. The Meter prints the current values to the terminal.		
Argument:	conf	rl_conf struct	Configuration
Return:	ret	integer	Returns 1, if successful.

Stop

rl_stop ret = rl_stop();

Description:	Stops the continuous sampling.		
Return:	ret	integer	Returns 1, if successful.

Status

rl_get_status state = rl_get_status(print);

Description:	Returns 1, if the RocketLogger is running, and 0 if not. In addition, it also prints the configuration to the terminal, if print is set to 1.		
Argument:	print	integer	Set print to 1, if configuration should be printed on the terminal.
Return:	state	integer	Returns 1, if running, else 0.

rl_get_status_web state = rl_get_status_web();

Description:	Prints the status to the terminal. For easier processing in the web client, it only prints the values, not the description.		
Return:	state	integer	Returns 1, if running, else 0.

Data

rl_get_data ret = rl_get_data();

Description:	Prints 100 values of the last buffer to the terminal in the JSON format. This can be easily processed in a web client.		
Return:	ret	integer	Returns 1, if successful.

Calibrate

rl_reset_calibration rl_reset_calibration();

Description:	Resets the calibration file. This is used when the device is recalibrated.

Configuration

The rl_conf configuration struct has the following members:

Name	**Type**	**Description**
state	integer	State of RocketLogger
rate	integer	Sampling rate in kSPS (1,2,4,8,16,32,64)
update_rate	integer	File update rate in Hz (1,2,5,10)
number_samples	integer	Number of samples to take (0 for continuous sampling)
channels	integer	Channel selection (e.g. I1H|I1M|I1L)
force_high_channels	integer	Force high channel selection (e.g. I1|I2)
enable_web_server	integer	1: webserver plotting enabled
binary_file	integer	1: binary output file, 0: csv output file
store	integer	1: file storing, 0: no file storing
file	char array	Filename
channels_string	char array	Unused (only for debugging)
force_high_channels_string	char array	Unused (only for debugging)

Every channel has a specified number:

I1H	1	I2H	32
I1M	2	I2M	64
I1L	4	I2L	128
V1	8	V3	256
V2	16	V4	512

The current channels have an additional number, used for high range forcing:

I1	1
I2	2

A.11. Matlab

This section describes the usage of the implemented Matlab functions.

Reading

rl_read_bin [values, header, time] = rl_read_bin(filename, old_file)

Description:	Imports a binary file.	
Arguments:	filename	Path to data file.
	old_file	Set old_file to 1, when reading in old data files.
Returns:	values	The sampled values.
	header	The header struct, extracted from the file. It is used for plotting.
	time	Timestamps with one entry per buffer.

rl_read_csv [values, header] = rl_read_csv(filename)

Description:	Imports a CSV file.	
Argument:	filename	Path to data file.
Returns:	values	The sampled values.
	header	The header struct, extracted from the file. It is used for plotting.

rl_merge_currents [merged_values, range] = rl_merge_currents(values, header)

Description:	Merges the three current ranges of both channels, if logged.	
Arguments:	values	The values to merge.
	header	The header struct of the input values.
Returns:	merged_values	The merged values.
	range	A vector with the active range of every sample.

Plotting

rl_plot [values, header] = rl_plot(filename, binary, old_file)

Description:	Imports and plots all sampled channels from a file.	
Arguments:	filename	Path to data file.
	binary	Set binary to 1, if a binary file is used.
	old_file	Set old_file to 1, when reading in old data files.
Returns:	values	The sampled values.
	header	The header struct, extracted from the file.

rl_plot_values rl_plot_values(values, header)

Description:	Plots all sampled channels from values that are already imported.	
Arguments:	values	The values to plot.
	header	The header struct, extracted from the file.

rl_plot_merged rl_plot_merged(merged_values, header)

Description:	Plots all channels with merged current ranges.	
Arguments:	merged_values	The values to plot.
	header	The header struct, extracted from the file.

Calibration

rl_cal_read [offsets, scales] = rl_cal_read(filename)

Description:	Reads and returns calibration values from a specified file.	
Argument:	filename	The calibration file.
Returns:	offsets	The calibration offsets, read from the file.
	scales	The calibration scales, read from the file.

rl_cal_write rl_cal_write(offsets, scales)

Description:	Writes the offsets and scales to the file 'calibration.dat'.	
Arguments:	offsets	The calibration offsets to write.
	scales	The calibration scales to write.

rl_calibrate [scales, offsets] = rl_calibrate(v, i1l, i1m, i1h, i2l, i2m, i2h, plotPareto)

Description:	Analyses the measured sweeps, calculates the calibration parameters and writes them to 'calibration.dat' file.	
Arguments:	v	Voltage sweep values (all four voltage channels at the same time).
	i1l	Current channel one low range sweep values.
	i1m	Current channel one medium range sweep values.
	i1h	Current channel one high range sweep values.
	i2l	Current channel two low range sweep values.
	i2m	Current channel two medium range sweep values.
	i2h	Current channel two high range sweep values.
	plotPareto	Set plotPareto to 1, if pareto-optimal calibration error figures should be plotted.
Returns:	offsets	The calculated calibration offsets.
	scales	The calculated calibration scales.

B. Original Problem

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Semester Thesis at the
Department of Information Technology and
Electrical Engineering

for

Matthias Leubin
and
Stefan Lippuner

Mobile Data-Logger for Ultra-Low Current and Power Measurements

Advisors: Lukas Sigrist
Andres Gomez
Roman Lim

Professor: Prof. Dr. Lothar Thiele

Handout Date: 11.03.2016
Due Date: 17.06.2016

1 Project Description

1.1 Introduction

A common trend in today's low-power microcontroller development and sensor node design is to reduce their active and especially their sleep power consumption ever more with every iteration. This allows aggressive power management but also opens new application fields where only very little energy is available, for example energy harvesting driven sensor nodes.

This field of ultra-low power systems also demands for new measurement equipment with lower current and power measurement ranges to characterize the power and energy consumption of these ultra-low power devices. The integration of many, previously separated components into a single System-on-Chip (SoC) also reduces the size of these devices and makes them small enough for a broad field of mobile applications, e.g. all kind of wearable applications. To further reduce the size by removing the comparably large but also costly energy storage device, many of these applications start focusing on energy harvesting sources. Using energy harvesting, the systems are directly powered by energy available form the environment, e.g. [4, 5], instead of relying on a battery that needs to be recharged periodically. The power generated by these harvesting sources is not only very small, but it can also be highly variable, depending on the harvesting environment. Therefore, it is very important to have real-world traces for development, simulation and testing of novel energy management strategies for this new field of applications.

For accurate and easy to use characterization of both the sensor nodes and the harvesting sources, it is important to have a mobile data logger with support for ultra-low standby and harvesting power measurements. At the same time, the logger should also provide the capability of a high measurement range to support characterization of power-hungry components such as wireless radios.

2 Project Goals

The overall goal of this semester thesis is to build a mobile data-logger for ultra-low current and power measurement for the characterization of energy harvesting sources and wireless sensor nodes. The students will design an analog circuit that allows measuring currents in the range from hundreds of milliamperes down to the micro- or nanoampere range. Combined with standard voltage measurements with millivolt accuracy, this allows precise characterization of energy harvesting sources as well as the energy consumption of wireless sensor nodes.

The analog front-end will be designed as an extension board, a so-called cape, for the BeagleBone Black embedded Linux platform [2]. Beside a measurement circuit for logging ultra-low currents, this cape will also require an additional control circuitry for seamless range switching, because the desired dynamic range cannot be covered with one single measurement range. The students will not only simulate the performance of the circuit during development, but also design a PCB to test and evaluate the performance of their design with real hardware implementation under

lab and real-world conditions.
Furthermore, the students will implement the software side of the data acquisition and management on top of a Linux operating system. This includes basic sensor data acquisition, management and display functionality. The system's performance will not only be simulated with SPICE and evaluated using sample power traces, but also compared to existing shunt resistor based solutions, like ACME [1].

3 Tasks

The project will be split up into several subtasks, as described below:

3.1 Familiarization with Ultra-Low Current Measurement Circuits and Components

In a first step the student will familiarize himself with circuit designs for low current measurements like the feed-back ammeter circuit [3] and commercially available analog components for this circuit.

3.2 Simulation of Low Current Measurement Circuits

Then the student will simulate the performance of the circuit using analog chips with different characteristics in order to decide on the combination of components that provide the best performance.

3.3 Design of an Automatic Range Switching Extension

The desired high dynamic range demands for automatic range switching. This can either be implemented in software for more flexible control, but for best performance this is implemented as part of the analog front-end using discrete components. The student's task will be to design and integrate a switching circuit as part of the analog front-end and verify its functionality based using SPICE simulation. This will allow later comparison of the discrete to the software based range switching.

3.4 PCB Design of a Low-Current Measurement Cape

The developed and simulated analog front-end will then be extended with voltage measurement capability. The student will design a PCB for a BeagleBone Black low power measurement cape, that can later be used for evaluation of the real world performance of the circuit.

3.5 Familiarization with Existing Power Measurement Board

In a first step the student will familiarize himself with an existing shunt resistor based measurement extension for the BeagleBone Black called ACME [1]. Even if not targeted to measure currents down to the micro or nano ampere range it shall be setup for this current range for later comparison measurements. Furthermore, the goal of this task is to study the software that is used for acquiring and displaying data with that measurement solution.

3.6 Implementation of Basic Logging Functionality

The students task then is to implement the low level code to configure and read out an analog to digital converter (ADC) using an ADC demo board. And to store them locally in the filesystem.

3.7 Visualization and Management of Acquired Measurement Traces

Once the data acquisition software is completed, the students task is to display acquired measurement to provide the user a first visualization to verify if the data acquisition was successful. This will either be implemented as a web-server that displays a simple page or using the software which is also part of the earlier studied ACME measurement environment. In addition to the data visualization a simple interface for remote control of the data acquisition (start/pause/resume/stop) and trace management (download/delete/name/rename) will be implemented.

3.8 Putting Together the Building Blocks

Once the PCB design is in production, the hardware and software shall be put together by adapting the software for the used ADC demo board to the final ADC if necessary. The software should then be ready for carrying out the experiments for the performance evaluation of the designed measurement device.

3.9 Evaluation of the Data Logger Performance

At the end of this project the performance of the developed measurement front-end will be evaluated in a controlled lab environment, as well as under real-world harvesting conditions. The performance will be compared to state-of-the-art embedded analog measurement systems like ACME, and on the other hand compared to high precision, stationary lab measurement equipment to evaluate the limitation of the proposed design.

3.10 Thesis Report and Final Presentation

Finally, a thesis report is written that covers all aspects of the project. The results are also presented in a final presentation during the group meeting of the Computer Engineering Lab.

4 Project Organization

4.1 Weekly Meeting

There will be a weekly meeting to discuss the project's progress based on a schedule defined at the beginning of the project. A short report concerning each week's progress (accomplished goals, difficulties, questions, next steps) should be provided at latest the day before the meeting.

4.2 Thesis Report

Two hard copies of the report need to be turned in at the end of the thesis. The copy remains property of the Computer Engineering and Networks Laboratory. A copy of the developed software needs to be handed in on CD or DVD along with the thesis report.

4.3 Initial and Final Thesis Presentation

In the first month of the project, the topic of the thesis will be presented in a short presentation during the group meeting of the Computer Engineering Lab. The duration of the talk is limited to five minutes. At the end of the project, the outcome of the thesis will be presented in one 20 minutes talk. It will take place during the group meeting of the Computer Engineering Lab.

4.4 Work Environment

The work will be carried out in the framework of the SNF project "Transient Computing Systems". This means that the results of this work can be used by the involved project partners if the project goals are met.

References

[1] Baylibre, Inc. Acme - baylibre. `http://baylibre.com/acme/`. [online].

[2] BeagleBoard.org Foundation. Beagleboard.org - black. `http://beagleboard.org/black`. [online].

[3] Keithley Instruments, Inc. Low level measurements handbook. 2013.

[4] Y. Lee, S. Bang, I. Lee, Y. Kim, G. Kim, M. H. Ghaed, P. Pannuto, P. Dutta, D. Sylvester, and D. Blaauw. A modular 1mm3 die-stacked sensing platform with low power I2C inter-die communication and multi-modal energy harvesting. *IEEE J. Solid-State Circuits*, 48(1):229–243, 2013.

[5] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith. WISPCam : A Battery-Free RFID Camera. In *IEEE RFID*, 2015.

Zurich, March, 2016

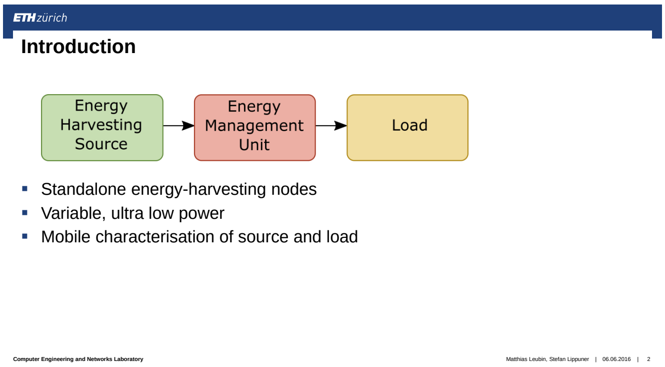
C. Presentation Slides

ETH zürich

RocketLogger: Mobile Data-Logger for Ultra-Low Power Measurements

Semester Thesis, Spring 2016
Matthias Leubin, Stefan Lippuner
Advisors: Lukas Sigrist, Andres Gomez, Roman Lim
Prof. Lothar Thiele

Computer Engineering and Networks Laboratory — Matthias Leubin, Stefan Lippuner | 06.06.2016 | 1

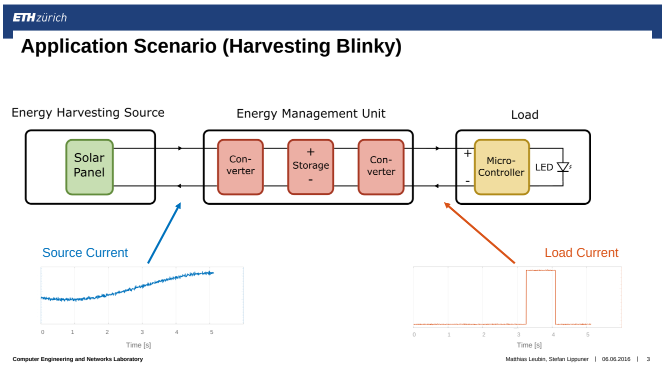


ETH zürich

Introduction

- Standalone energy-harvesting nodes
- Variable, ultra low power
- Mobile characterisation of source and load

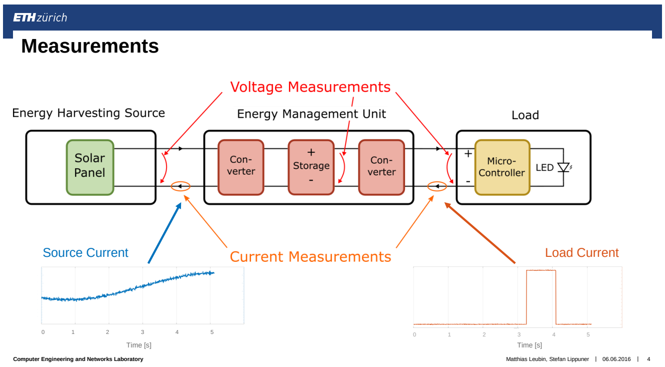
Computer Engineering and Networks Laboratory — Matthias Leubin, Stefan Lippuner | 06.06.2016 | 2



ETH zürich

Application Scenario (Harvesting Blinky)

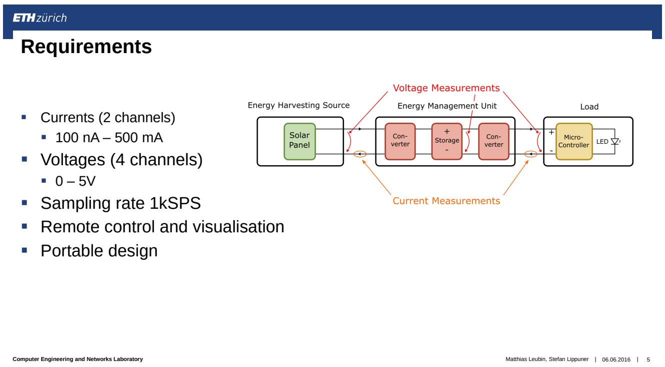
Computer Engineering and Networks Laboratory — Matthias Leubin, Stefan Lippuner | 06.06.2016 | 3



ETH zürich

Measurements

Computer Engineering and Networks Laboratory — Matthias Leubin, Stefan Lippuner | 06.06.2016 | 4

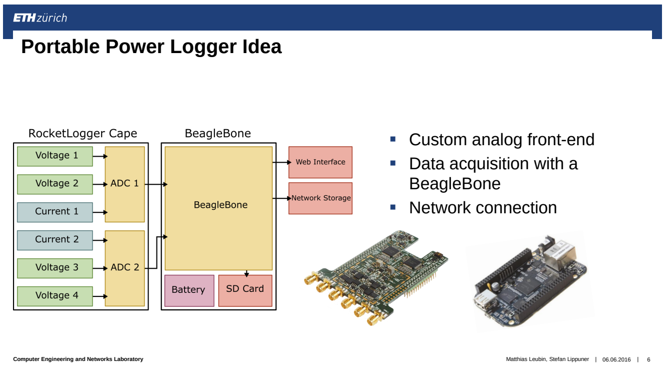


ETH zürich

Requirements

- Currents (2 channels)
  - 100 nA – 500 mA
- Voltages (4 channels)
  - 0 – 5V
- Sampling rate 1kSPS
- Remote control and visualisation
- Portable design

Computer Engineering and Networks Laboratory — Matthias Leubin, Stefan Lippuner | 06.06.2016 | 5

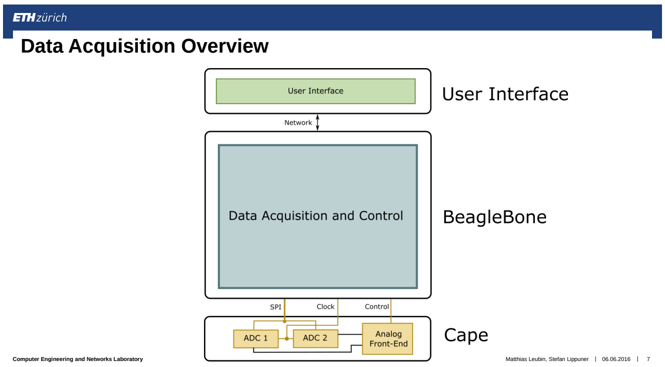


ETH zürich

Portable Power Logger Idea

- Custom analog front-end
- Data acquisition with a BeagleBone
- Network connection

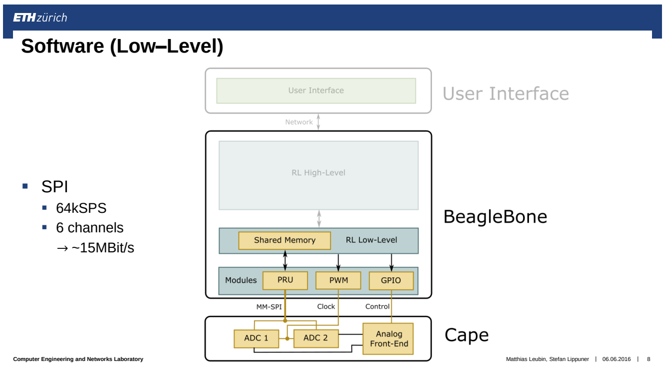
Computer Engineering and Networks Laboratory — Matthias Leubin, Stefan Lippuner | 06.06.2016 | 6



ETH zürich

Data Acquisition Overview

Computer Engineering and Networks Laboratory — Matthias Leubin, Stefan Lippuner | 06.06.2016 | 7

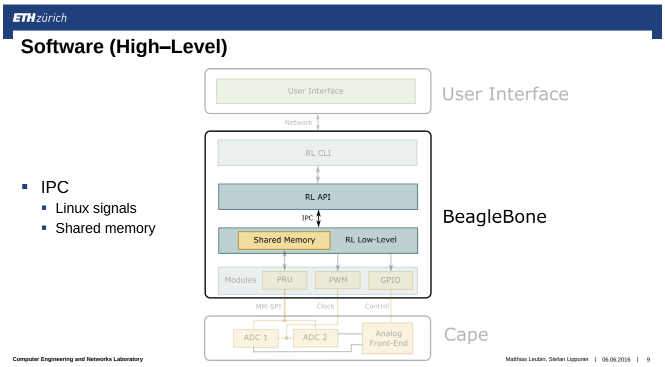


ETH zürich

Software (Low–Level)

- SPI
  - 64kSPS
  - 6 channels
    → ~15MBit/s

Computer Engineering and Networks Laboratory — Matthias Leubin, Stefan Lippuner | 06.06.2016 | 8

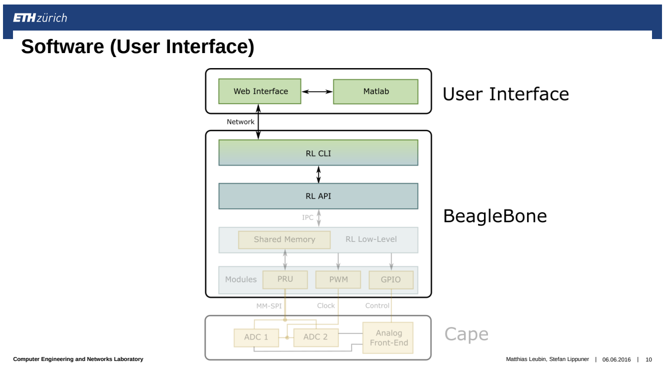


ETH zürich

Software (High–Level)

- IPC
  - Linux signals
  - Shared memory

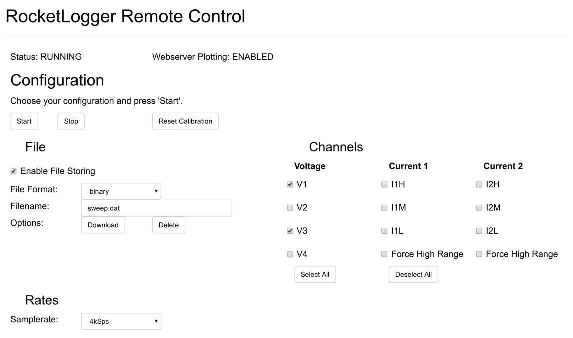
Computer Engineering and Networks Laboratory — Matthias Leubin, Stefan Lippuner | 06.06.2016 | 9



ETH zürich

Software (User Interface)

Computer Engineering and Networks Laboratory — Matthias Leubin, Stefan Lippuner | 06.06.2016 | 10



RocketLogger Remote Control

Status: RUNNING   Webserver Plotting: ENABLED

Configuration

Choose your configuration and press 'Start'.

File

☑ Enable File Storing

File Format: binary

Filename: sweep.dat

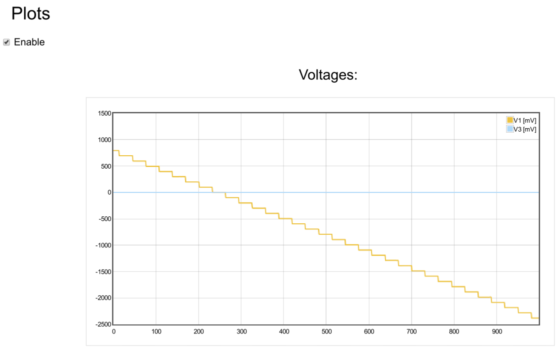
Options: Download Delete

Channels

Voltage	Current 1	Current 2
☑ V1	□ I1H	□ I2H
□ V2	□ I1M	□ I2M
☑ V3	□ I1L	□ I2L
□ V4	□ Force High Range	□ Force High Range

Rates

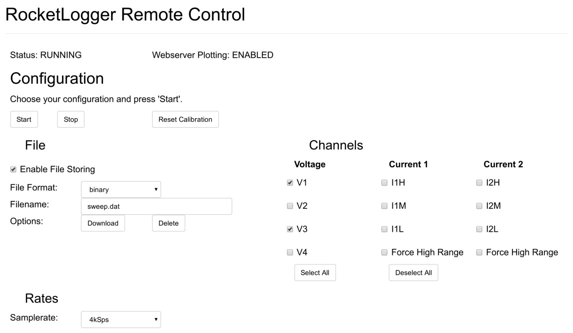
Samplerate: 4kSps



Plots

☑ Enable

Voltages:



RocketLogger Remote Control

Status: RUNNING   Webserver Plotting: ENABLED

Configuration

Choose your configuration and press 'Start'.

File

☑ Enable File Storing

File Format: binary

Filename: sweep.dat

Options: Download Delete

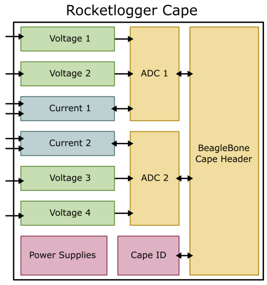
Channels

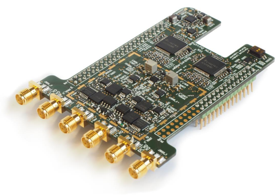
Voltage	Current 1	Current 2
☑ V1	□ I1H	□ I2H
□ V2	□ I1M	□ I2M
☑ V3	□ I1L	□ I2L
□ V4	□ Force High Range	□ Force High Range

Rates

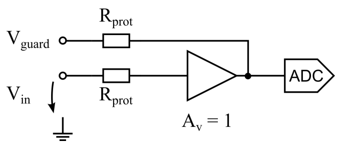
Samplerate: 4kSps

Cape Overview



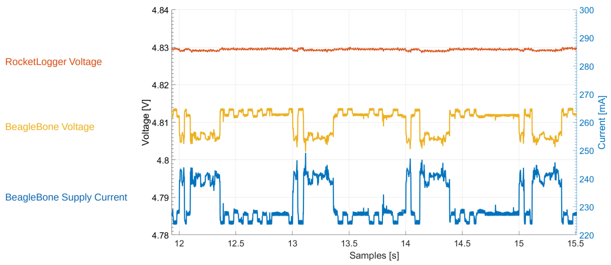


Voltage Channel

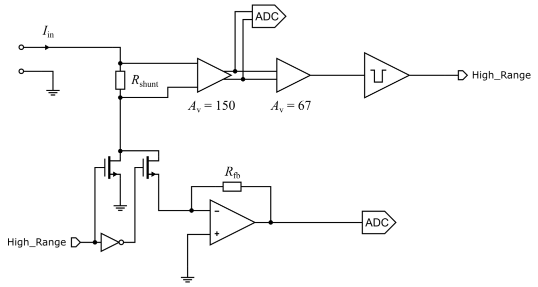


- Low input current
  - Input buffer op-amp
  - Guard around input
- Protection resistors
- Resultion
  - 10s of µVs

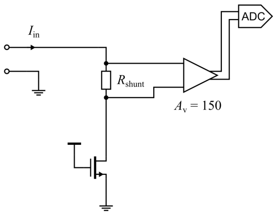
Voltage Channel – Measurement



Current Channel

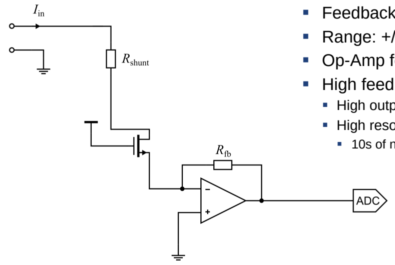


Current Channel – High Range



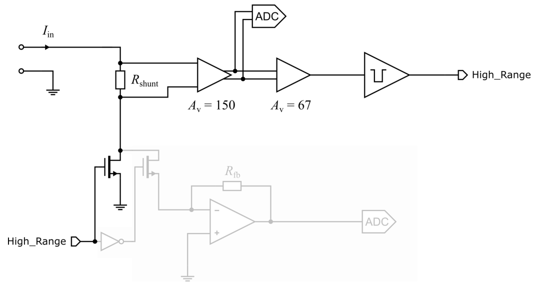
- Shunt current meter
- Range: +/- 500mA
- Low shunt resistor
  - Low burden voltage
- Small measured voltage
  - Too noisy for low currents

Current Channel – Low Range

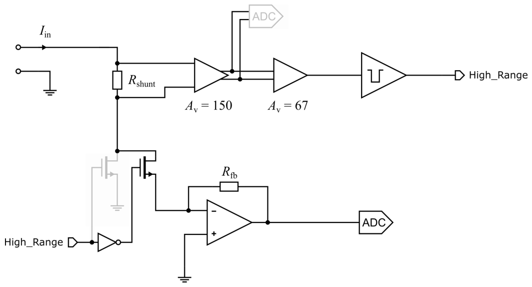


- Feedback current meter
- Range: +/- 2mA
- Op-Amp forces $V^+ - V^- = 0$
- High feedback resistor
  - High output voltage
  - High resolution
    - 10s of nA

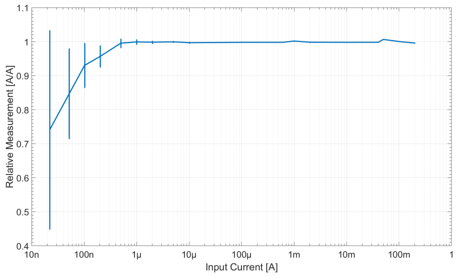
Current Channel – High Range Mode



Current Channel – Low Range Mode

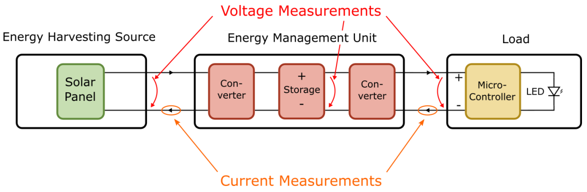


Current DC Measurements

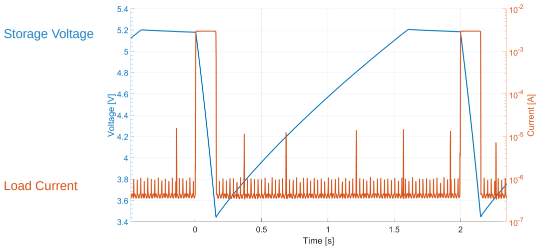


Sample Rate: 1kSPS
RMS Noise: 6.5nA

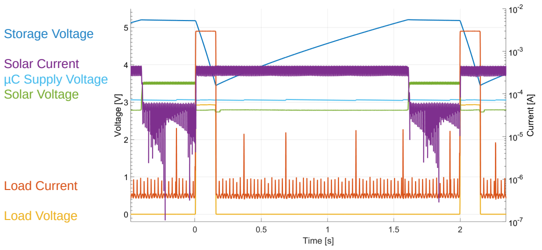
Harvesting Blinky Setup



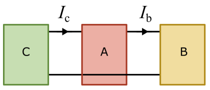
Harvesting Blinky Example



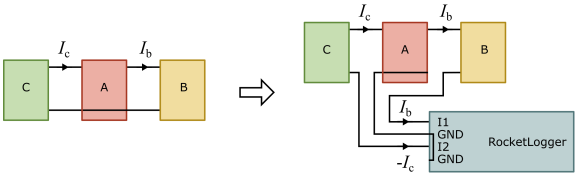
Harvesting Blinky Example



Non-Isolated Dual Current Measurement



Non-Isolated Dual Current Measurement

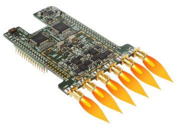


Performance

Metric	RocketLogger
Sampling Frequency	1kSPS to 64kSPS
Analog Bandwidth	0.26kHz to 10kHz
Voltage Range	-6V to 6V
Voltage Noise (RMS)	7.85µV
Input Impedance	~1TΩ
High Current Range	-500mA to 500mA
High Range Current Noise (RMS)	1.23 µA
Low Current Range	-2mA to 2mA
Low Range Current Noise (RMS)	6nA
Power Consumption	~1.8W
Output Data Rate	Up to 10.8 GB/h

Conclusion

- Portable power logger
- High accuracy
- Fast range switching
- Easy-to-use web interface

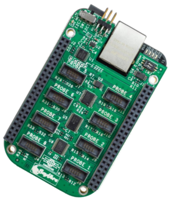


Questions

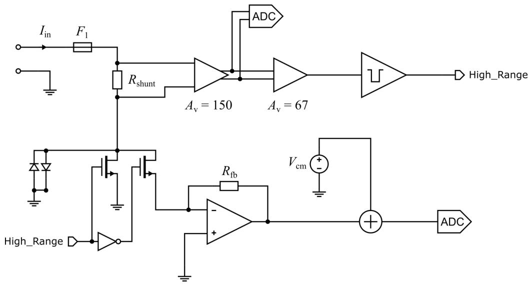
Additional Slides

ACME Performance

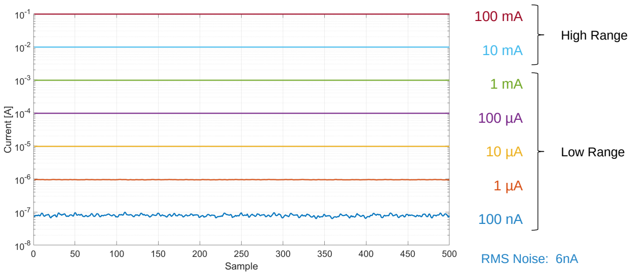
Metric	ACME
Sampling Frequency	< 500SPS
Voltage Range	0 to 32V
Current Range	0 to 10A (in 3 ranges)
Range switching	fixed
Current Noise (RMS)	5 µA



Current Channel

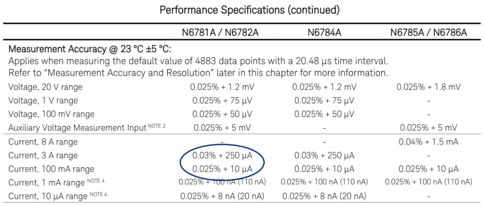


Current DC Measurements



RMS Noise: 6nA

SMU Specs



Performance Specifications (continued)

	N6781A / N6782A	N6784A	N6785A / N6786A
Measurement Accuracy @ 23 °C ±5 °C: Applies when measuring the default value of 4883 data points with a 20.48 µs time interval. Refer to "Measurement Accuracy and Resolution" later in this chapter for more information.			
Voltage, 20 V range	0.025% + 1.2 mV	0.025% + 1.2 mV	0.025% + 1.8 mV
Voltage, 1 V range	0.025% + 75 µV	0.025% + 75 µV	-
Voltage, 100 mV range	0.025% + 50 µV	0.025% + 50 µV	-
Auxiliary Voltage Measurement Input NOTE 2	0.025% + 5 mV	-	0.025% + 5 mV
Current, 8 A range	-	-	0.04% + 1.5 mA
Current, 3 A range	0.03% + 250 µA	0.03% + 250 µA	-
Current, 100 mA range	0.025% + 10 µA	0.025% + 10 µA	0.025% + 10 µA
Current, 1 mA range NOTE 4	0.025% + 100 nA (110 nA)	0.025% + 100 nA (110 nA)	0.025% + 100 nA (110 nA)
Current, 10 µA range NOTE 4	0.025% + 8 nA (20 nA)	0.025% + 8 nA (20 nA)	-

List of Acronyms

ADC	analog-to-digital converter.
API	application programming interface.
BJT	bipolar junction transistor.
CLI	command line interface.
CSV	comma-separated values.
DTO	device tree overlay.
EMU	energy management unit.
ENOB	effective number of bits.
FIFO	first-in first-out.
GPIO	general purpose input/output.
IoT	Internet of Things.
IPC	inter process communication.
McASP	multichannel audio serial port.
McSPI	multichannel serial port interface.
MM-SPI	multi-MISO SPI.
OS	operating system.
PCB	printed circuit board.
PGA	programmable gain amplifier.
PRU	programmable real-time unit.
PSD	power spectral density.
PWM	pulse-width modulation.

RMS	root mean square.
SMU	source measure unit.
SNR	signal-to-noise ratio.
SPI	serial peripheral interface.
TIK	Institut für Technische Informatik und Kommunikationsnetze.
UI	user interface.

Bibliography

[1] A. Gomez, L. Sigrist, M. Magno, L. Benini, and L. Thiele, "Dynamic energy burst scaling for transiently powered systems," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. IEEE, 2016, pp. 349–354.

[2] *ACME - BayLibre*, BayLibre, rev. B, [Accessed 2016-06-15]. [Online]. Available: http://baylibre.com/acme/

[3] J. Hester, T. Scott, and J. Sorber, "Ekho: Realistic and repeatable experimentation for tiny energy-harvesting sensors," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 2014, pp. 330–331.

[4] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*. IEEE, 2013, pp. 153–165.

[5] R. Pozza, A. Gluhak, and M. Nati, "Smarteye: an energy-efficient observer platform for internet of things testbeds," in *Proceedings of the seventh ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*. ACM, 2012, pp. 59–66.

[6] P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits*, 5th ed. Wiley, 2009.

[7] S. Haykin, *Communication Systems*, 4th ed. Wiley, 2001.

[8] *OPA2x11 Datasheet*, Texas Instruments, November 2015, rev. H, [Accessed 2016-06-10]. [Online]. Available: http://www.ti.com/lit/gpn/opa2211a

[9] Keithley, *Low Level Measurement Handbook*, 7th ed. Keithley, 2013.

[10] *LOG114 Datasheet*, Texas Instruments, March 2007, rev. A, [Accessed 2016-06-10]. [Online]. Available: http://www.ti.com/lit/ds/symlink/log114.pdf

[11] *ADS126x Datasheet*, Texas Instruments, July 2015, rev. B, [Accessed 2016-06-10]. [Online]. Available: http://www.ti.com/lit/gpn/ads1262

[12] *ADA4898 Datasheet*, Analog Devices, 2015, rev. E, [Accessed 2016-06-10]. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/ADA4898-1_4898-2.pdf

[13] *AD7177-2 Datasheet*, Analog Devices, 2016, rev. B, [Accessed 2016-06-17]. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/AD7177-2.pdf

[14] B. Baker, *A Glossary of Analog-to-Digital Specifications and Performance Characteristics*, Texas Instruments, October 2011, rev. B, [Accessed 2016-06-15]. [Online]. Available: http://www.ti.com/lit/an/sbaa147b/sbaa147b.pdf

[15] *ADS131E08S Datasheet*, Texas Instruments, January 2016, rev. A, [Accessed 2016-06-15]. [Online]. Available: http://www.ti.com/lit/ds/symlink/ads131e08s.pdf

[16] *AD8421 Datasheet*, Analog Devices, 2012, rev. 0, [Accessed 2016-06-10]. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/AD8421.pdf

[17] *ADA4522 Datasheet*, Analog Devices, April 2016, rev. D, [Accessed 2016-06-15]. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/ADA4522-1_4522-2_4522-4.pdf

[18] *OPAx192 Datasheet*, Texas Instruments, November 2015, rev. E, [Accessed 2016-06-15]. [Online]. Available: http://www.ti.com/lit/ds/symlink/opa192.pdf