



TKN

Telecommunication
Networks Group

Technical University of Berlin
Telecommunication Networks Group

Measuring the node energy consumption in a USB based WSN testbed

Andreas Köpke
koepke@tkn.tu-berlin.de

Berlin, November 30, 2007

TKN Technical Report TKN-07-008

TKN Technical Reports Series Editor:
Prof. Dr.-Ing. Adam Wolisz

Copyright at Technical University of Berlin. All Rights reserved.

Abstract

Energy consumption is one of the critical concerns in sensor networks. In particular, there is substantial interest to investigate the distribution of the energy consumption among different nodes depending on the protocol stack applied for specific traffic characteristics. Unfortunately, experimental investigations to this point are rather difficult, as means for cheap, easy to deploy, while precise estimation of actual energy consumption are not really available.

As a solution we developed an affordable and precise circuit that measures the energy consumption in situ. It delivers the result to the sensor node, enabling nodes to take their remaining energy into account. Furthermore, it is to some extent independent from the node type and the testbed type.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 2 |
| 2 | Requirements and design decisions | 2 |
| 3 | Circuit | 3 |
| 3.1 | Signal conditioning | 4 |
| 3.2 | Signal conversion | 5 |
| 3.3 | Control interface | 5 |
| 4 | Performance Evaluation | 6 |
| 4.1 | Constant current draw | 6 |
| 4.2 | Bursty consumption | 7 |
| 4.3 | Node software | 10 |
| 5 | Related work | 10 |
| 6 | Conclusion | 11 |
| 7 | Acknowledgements | 11 |
| A | Schematic | 12 |
| A.1 | Drawing | 12 |
| A.2 | Explanations | 13 |
| B | Top layout | 13 |
| C | Bottom layer | 14 |
| D | Bill of Materials | 14 |
| | References | 14 |

1 Introduction

The selection of protocols for Wireless Sensor Networks (WSNs) is frequently driven by the consideration of the network lifetime which, in turn, depends directly on the energy consumption of individual nodes. The latter is not easy to estimate in advance: the actual traffic patterns as well as the time variable quality of connectivity among individual nodes can strongly influence the results. It is difficult to model all the influences in a simulation, or an analytic evaluation, hence there is a strong interest in experiments.

Running such experiments without a supporting infrastructure (a testbed) is a tedious work, especially when more than 20 nodes are involved. Thus, researchers all over the world started to build testbeds. The first generation testbeds have, however, lacked support to measure energy consumption. Instead, it usually was computed from the states of different node components e.g. the radio modem. In fact, as most of the testbeds provide the power supply from the testbed infrastructure rather than from batteries, energy consumption estimates were out of scope of practical verification. But even if the nodes have been battery driven the estimation of remaining energy within individual batteries is very imprecise at least.

The goal of the research described in this paper was to develop a low cost, precise tool for on-line monitoring of the energy consumption of individual nodes within an existing testbed. In addition, the result of the energy monitoring should be available at any time to the node itself, making it possible to investigate different energy dependent protocols.

The existing testbeds differ in the way the control infrastructure and the power provisioning is solved. Most of the testbeds depend on power and control supply via a wired infrastructure, being either Ethernet or USB. The TWIST testbed [2] developed at TU Berlin demonstrates how USB based power control can be used efficiently for precise topology configuration. Therefore the following considerations are focused on USB based testbeds.

2 Requirements and design decisions

The USB based testbed defines the starting point for the design: the measurement circuit should not interfere with the functionality of the testbed to run and control experiments. Moreover, it should not require any modifications either on the side of the testbed or on the side of the sensor nodes. The measurement circuit is therefore designed to mimic a short USB extension cable, plugged between the node and its connection to the testbed.

We believe that results of the energy consumption measurements have to be available within the node under study during the experiment, in order to be used as inputs for operation optimization in numerous protocols and settings. We envision our circuit to deliver result to the node, using its pin header. This pin header is present at each sensor node, for connecting sensors and other additional hardware.

Once the measurement circuit is attached to the testbed and the node, it should—ideally—not influence the operation of the node, and deliver a result that is precise and easy to process. More formally, it must meet the following requirements.

Form of result The measurement result should be delivered to the node during the experiment, and the first result should be available at most one hour after the measurement was started. This time is a somewhat arbitrary choice, but it determines the minimal duration of an experiment in a sensor network.

Precision The deviation of the measured energy consumption should not differ by more than 5% from the real energy consumption. To limit the measurement range and simplify the design we do not require the circuit to measure the energy consumption of a sleeping node. This constant basic consumption is typical for each sensor node platform and can be measured independently.

Perturbation The measurement influences the node in two ways: The sensing part of the measurement circuit may lead to small variations in the supply voltage that is applied to the node. In addition, the node uses some computational resources to process the result.

The microprocessors and radio modems accept small variations in the supply voltage, because they can run from batteries where the supply voltage drops over time. The voltage drop caused by the measurement circuit should be at most 0.1 V, which is acceptable for all chips on the node.

The additional load on the microprocessor should be as small as possible; it should use less than 1% of the processing power. The measurement circuit may wake up a sleeping node to process the result, but this additional energy consumption should not exceed 1% of the energy consumption in the sleep state.

3 Circuit

The basic idea how to measure the energy consumption of a node is as follows: A voltage source (V) supplies the node with energy. The shunt (R) converts the flowing current into a voltage that is used as an input for the circuit. The node controls the operation of the circuit and receives the result from it.

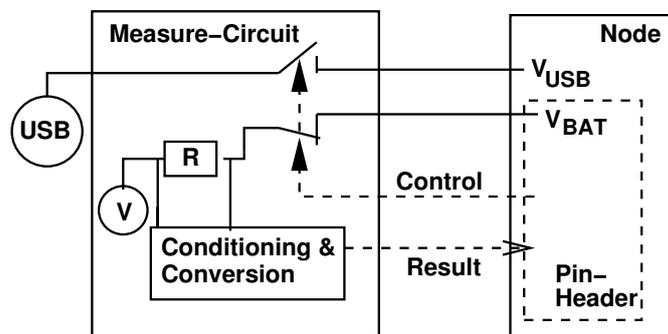


Figure 1: Idea of the circuit

The idea of the circuit is shown in Figure 1. To enhance readability, we dropped the return path for the current, and just show the supply lines. Instead of connecting the node directly to the testbed, we connect it to the measurement circuit, which is

in turn connected to the testbed. This way, a shunt could be inserted into the power supply line of the USB connection. But there is a problem with this approach: For an energy consumption measurement the nodes should run under conditions that reflect their intended deployment. This implies that they can not rely on their USB interface for energy provision or data exchange. In addition, the USB circuitry on the node (the voltage regulator that converts the 5 V USB supply voltage to 3 V, the serial-to-USB converter chip, etc.) consume a considerable amount of energy: it is in the same order of magnitude as the maximum consumption of the microprocessor of the node. This makes a precise measurement difficult. Instead of relying on the USB connection to power the node, we decided to make it more realistic: in measurement mode, the node is powered via its battery connector. This connector is attached to a voltage regulator on the measurement circuit, which is in turn powered by the testbed. As a result, the measurement circuit behaves like a battery for the node. This allows us to make a precise measurement: the current flows from the circuit via the shunt to the node. The USB circuitry on the node is not powered; it can not influence the measurement results.

The node controls the operation of the circuit, and the circuit signals the result to the node. For this, the measurement circuit is connected to the node's pin header. How the node controls the measurement circuit is described in more detail in section 3.3, in the next section we describe how the circuit arrives at a result that it can signal to the node.

3.1 Signal conditioning

The difference between the sleep current and the active current of sensor node platforms implies a large input range of the measurement circuit. For the eyesIFX platform, the energy consumption varies between $12 \mu\text{A}$ during sleep and around 30 mA when all components (processor, radio, flash memory...) are on. The tmote sky consumes between $8 \mu\text{A}$ during sleep and slightly more than 50 mA when all its components are on. This implies a dynamic range of the circuit of more than four decades. We simulated several solutions of the sensing and signal conditioning part (resistor and amplifier, diode and amplifier, logarithmic amplifier, current feedback amplifier, split approaches using two amplifiers...), but they did not perform satisfactory. They demanded a high number of precise components and a careful noise shielding. The latter seemed questionable, given the noisy environment like radio modems on the nodes and fluorescent lights in the offices.

But do we actually need to have such a large input range of four decades? The nodes microprocessor needs about 2 mA when active (cmp. Figure 3), to measure it an input range from 1 mA and 50 mA would be sufficient. So we dropped the requirement to measure the sleep energy consumption: It is a constant consumption that can be measured with a good multimeter. If necessary, it can be added to the total energy consumption when the experiment is over. This enabled a much simpler approach to measure the power consumption of the node: an input range of two decades is sufficient to detect all changes from sleep to active mode, and the sensing part is reduced to a shunt of 1Ω .

3.2 Signal conversion

The current drawn by a node is proportional to its power consumption, but we want to measure its energy consumption, and to communicate the result to the node. This is the task of the conversion stage: it integrates the voltage drop over the shunt, and triggers an interrupt on the node once a certain amount of energy was consumed. This voltage-to-frequency conversion is very handy for sensor nodes: the node can do whatever it needs to do and is only disturbed once a in a while to process the result. In our circuit we use the LTC4150 [4] for the signal conditioning and conversion. This IC amplifies and integrates the voltage drop over the $1\ \Omega$ shunt. Once the integrator reaches a certain threshold voltage, an interrupt is triggered, and the integration is reset. This IC makes our solution affordable and compact. On the downside it can not measure voltages below $100\ \mu\text{V}$. Since the node consumes more than $100\ \mu\text{A}$ in any active mode, this disadvantage is not important.

3.3 Control interface

The LTC4150 provides the signal conditioning and conversion part of our circuit, but the node still needs an interface to control the operation of the circuit. The design of the control interface is crucial, because it must ensure that the testbed can reprogram the node even if the program on the node is malfunctioning, e.g. by switching to measure mode by accident.

Choosing the mode As a basic protection, we require the control part of the circuit to switch to USB mode by default and to prevent an accidental switch into measure mode. When the components of the control part are properly wired, the circuit will be in USB mode by default. It will return to this mode whenever the USB power supply switched off and on, regardless what software is running on the node.

In principle, a single pin of the microprocessor is enough to switch into measure mode and back. But the pins of the microprocessor are versatile and one can not rely on a certain state. Also, the node software will change the pin status during boot up, and an unaware programmer might switch to measure mode by accident. To prevent such accidents, the microprocessor should send a command to switch the modes.

We evaluated several possible switches, but in the end we decided to use a simple binary counter. The node toggles a pin repeatedly which is connected to the input of a binary counter. The output pins of the counter change accordingly: they represent the number of state changes on the input pin as binary high and low states. Once the node has counted up to a certain number, the circuit is switched into measure mode. This greatly reduces the probability of an accidental switch, and the default state of the counter ensures that the circuit is in USB mode after power up.

Brown out safety When the circuit and the node are in measure mode, the voltage regulator on the node that converts the $5\ \text{V}$ from USB to the $3\ \text{V}$ necessary to power the node is—just like the whole USB circuitry—switched off. When

the node switches from measure mode to USB mode, the node's voltage regulator needs some time to stabilize. During this time, the node may experience a brown out: the supply via the nodes battery connector is cut while the power provided by the USB circuitry on the node is not yet stable. To prevent this, we use the counter again to ensure a make-before-break switch. By counting up, the USB circuitry on the node is enabled. Once the power supply from it is stable, the node continues to count. This will cut the power supply via the battery connector and the node is now in USB mode without noticing the switch over in its supply voltage.

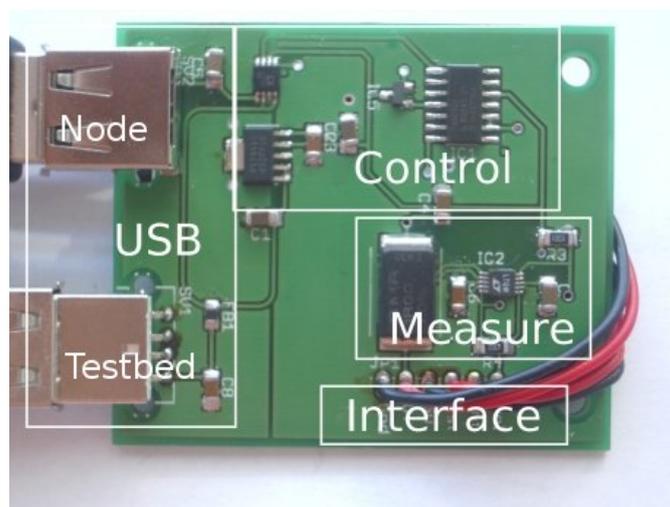


Figure 2: Final Circuit

Final circuit Figure 2 shows the final circuit with its elements. The USB part acts like a short USB cable and provides the measurement circuit with energy. The control part enables the node to switch from USB into measure mode and back. It uses two pins on the interface: one is the count input and the other one is used to reset the counter (and the measurement circuit) into its default state. During the measurement, two pins of the measurement circuit supply the node with energy, these pins are connected to the battery connector of the node. The result of the measurement is exported on two pins of the measurement circuit—the circuit can deliver the result not only to one, but to two nodes. This is useful when the energy consumption of a closed source node program shall be measured, because the processing of the result can be done by a second node, without modifying the program on the node under examination.

4 Performance Evaluation

4.1 Constant current draw

Sensor nodes consume energy in bursts: after performing some action, they go to sleep. But the circuit can not measure currents below a certain threshold, and in

this part of the performance evaluation we identify this threshold as well as the precision of the measurement circuit.

The circuit can deliver the result to two nodes, and this feature is used for the performance evaluation. One node is connected to one of the two result pins. It measures the time that passes between two interrupts and delivers it via USB to the computer. To compute the energy consumption, we take the average time between two interrupts of at least 10 interrupts. The time between two interrupts is very stable, that is the standard deviation is so small that even a single measurement already provides an accurate result. Using this time, the energy consumption in [mAs] can be easily computed. The corresponding average current in [mA] allows an easier comparison of the results.

The consumer under examination is connected to the power supply pins of the circuit. For the constant current draw evaluation, we use a resistor. The performance evaluation was done at 3 V, and the result is shown in Table 1. The high quality digital multimeter Fluke 182 is used as a reference.

| <i>Resistor</i> [Ω] | <i>Circuit</i> [mA] | <i>Fluke 182</i> [mA] | Rel. Err. [%] |
|---------------------------------|------------------------|--------------------------|------------------|
| 100 | 29.53 | 29.49 | 0.14 |
| 1k | 3.016 | 2.999 | 0.6 |
| 10k | 0.298 | 0.300 | 1 |
| 24.3k | 0.115 | 0.121 | 5 |
| 47.5k | 0.048 | 0.065 | 26 |

Table 1: Constant current draw

In the interesting region down to 0.3 mA the deviation between the value measured by the multimeter and the value measured by the coulomb counter is very small and well in the acceptable range.

4.2 Bursty consumption

The evaluation of the circuit for bursty power consumption is quite tricky—we have to use an oscilloscope for the reference measurement. An oscilloscope can track the energy consumption over time, but it has some limitations. Our digital oscilloscope has an 8 bit resolution and 10 MHz (or 20 MS/s) bandwidth. While the bandwidth is sufficient, the resolution introduces a quantization error.

The evaluation requires the definition of a short program that the circuit should be able to capture. The smallest possible program wakes up on a timer, reschedules it and puts the eyesIFX node back to sleep. Its power profile is shown in Figure 3.

This provides one measurement point but allows only a limited conclusion on the behavior of the circuit under dynamic conditions. A second program is shown in Figure 4. In this program, the four LEDs of the node are switched on, the timer is rescheduled and then the LEDs are switched off.

In the third case, the node runs a MAC protocol: the node switches on the radio, listens to the channel for a while and turns it off again. The power profile is shown in Figure 5. The highest peaks in the energy consumption occur when the microprocessor and the radio are both on. When the radio is on and a byte is

detected, the microprocessor is woken up and processes the byte, causing a peak in the energy consumption. For this performance evaluation there is no sender, and all “bytes” are just noise. This introduces a small random component into the evaluation.

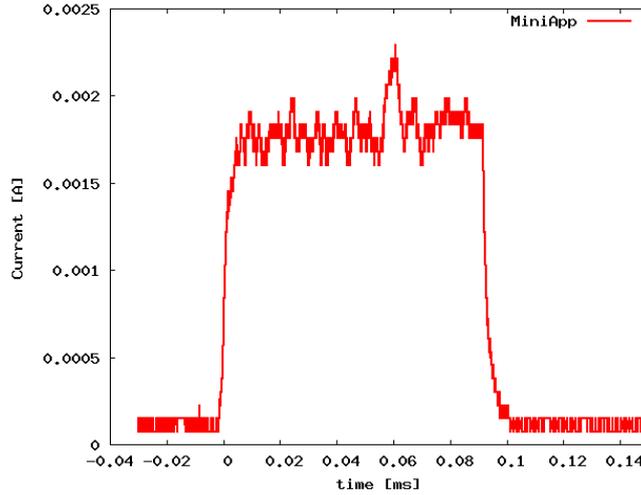


Figure 3: Periodic program

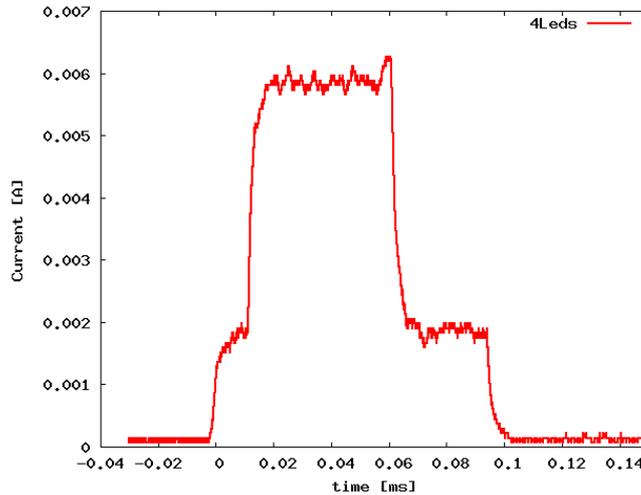


Figure 4: Periodic program with LEDs

The evaluation is done using these three programs. For each program we varied the “duty cycle”: each program has a timer that restarts it. The time between two restarts is varied; it is shown in the “cycle” column in the table.

Using the measurements of the oscilloscope, the energy consumption of one burst (E_a) can be computed by integrating the area under the curve. The total energy consumption of the node can be computed by counting the number (N) of these bursts in time t and adding the sleep energy consumption ($E_s = t \cdot 0.012 \text{ mA}$). The total consumed energy E can be computed as $E = NE_a + E_s$. The en-

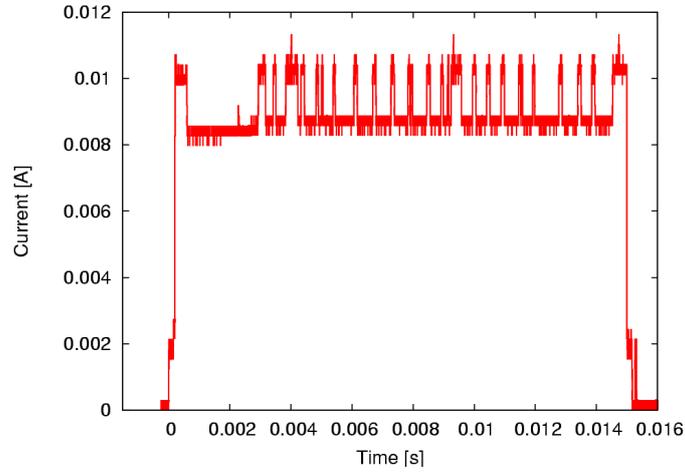


Figure 5: MAC protocol energy consumption

ergy consumption of the periodic program without LEDs is $0.157 \pm 0.014 \mu\text{As}$ per run, with LEDs it is $0.366 \pm 0.011 \mu\text{As}$ per run and for the MAC protocol it is $132.4 \pm 4.7 \mu\text{As}$ per wake up. The quantization error is different for each program, because each of them is measured using a different input range of the oscilloscope.

| <i>Cycle</i> [ms] | <i>Circuit</i> [mA] | <i>Osc.</i> [mA] | Rel. Err. [%] |
|----------------------|------------------------|---------------------|------------------|
| 0.122 | 1.33 | 1.29 ± 0.12 | 3 |
| 0.144 | 0.674 | 0.651 ± 0.06 | 3.4 |
| 0.288 | 0.338 | 0.331 ± 0.003 | 1.9 |
| 0.977 | 0.17 | 0.172 ± 0.002 | 1 |
| 1.953 | 0.084 | 0.092 ± 0.001 | 9 |
| 3.901 | 0.034 | 0.052 ± 0.001 | 52 |

Table 2: Periodic program without LEDs

| <i>Cycle</i> [ms] | <i>Circuit</i> [mA] | <i>Osc.</i> [mA] | Rel. Err. [%] |
|----------------------|------------------------|---------------------|------------------|
| 0.144 | 1.56 | 1.51 ± 0.09 | 3.6 |
| 0.288 | 0.782 | 0.76 ± 0.05 | 3.2 |
| 0.977 | 0.394 | 0.39 ± 0.02 | 2.7 |
| 1.953 | 0.199 | 0.199 ± 0.01 | 2 |
| 3.901 | 0.104 | 0.106 ± 0.006 | 0.1 |
| 7.81 | 0.046 | 0.059 ± 0.003 | 28 |

Table 3: Periodic program with LEDs

Usually, the consumption measured with the circuit is within or very close to the bounds measured with the oscilloscope. For the periodic programs the results are imprecise once the consumption drops below 0.1 mAs. An explanation may be the leakage current of the integrator. One impulse generated by the internal inte-

| <i>Cycle</i> [ms] | <i>Circuit</i> [mA] | <i>Osc.</i> [mA] | Rel. Err. [%] |
|----------------------|------------------------|---------------------|------------------|
| 50 | 2.8 | 2.61 ± 0.09 | 5 |
| 100 | 1.41 | 1.31 ± 0.05 | 5.2 |
| 250 | 0.54 | 0.52 ± 0.02 | 0.3 |
| 500 | 0.272 | 0.261 ± 0.01 | 1.8 |
| 1000 | 0.148 | 0.131 ± 0.005 | 2.4 |
| 2000 | 0.074 | 0.065 ± 0.002 | 5.6 |

Table 4: MAC protocol

grator represents $30 \mu\text{As}$ —hence 180 runs are needed before the internal counter of the LTC4150 is incremented. If the runs are spaced too far in time, the charge of the integrator has a chance to leak away. The observation with the MAC protocol seems to support this hypothesis. Here, only 1 of 4.4 integrations is not yet stored in the internal counter and has a chance to leak away. In this case, the measured energy consumption stays close to the required precision.

4.3 Node software

The software that must be installed on the node to control a measurement is very simple. The node must be able to toggle a pin repeatedly to switch the circuit into measure mode, and it must handle the signaled interrupt. It could use it to increment a counter, and this counter would represent the energy consumed by the node during an experiment with a fixed duration. The interrupt can be handled in less than 20 instructions or at most $11 \mu\text{s}$ on an eyesIFX node: it uses a negligible amount of processing resources. The interrupts appear at most once every 0.6 s, the circuits impact on the processing resources of the node is about 20 ppm. The maximum time between two interrupts is indeed less than one hour: The circuit fully meets our specification on the processing overhead requirement.

5 Related work

We started to evaluate possible solutions to measure the energy consumption, because no solutions were available. By now, this situation has improved and here we give a brief overview of these solutions.

An interesting solution is the software based approach of [1]. Whenever a component is turned on or off, it informs the energy estimator, which keeps track of the consumed energy. This approach needs some support from the operating system, but it can be implemented with reasonable overhead. They used a 4kHz clock to track how long a component is switched on. This time resolution is a critical parameter: on the one hand, must be high enough such that all interesting changes can be monitored, but on the other hand it should be low enough to limit the impact on the operation of the sensor node. This software based approach should be verified with our hardware based approach.

SPOT [3] on the other hand is an interesting hardware based approach. It is designed to measure the energy consumption very precisely, and achieves a very

high resolution down to $1\ \mu\text{A}$ for constant currents and seems to capture the bursty energy consumption, but the paper does not attempt to quantify how precise the results are. The high resolution of the circuit comes at a price in complexity and costs: It consists of two boards, the measurement board and the adaptor board that makes SPOT somewhat independent of the node platform. We assess that SPOT costs around 35 €, while our solution costs less than 15 €.

6 Conclusion

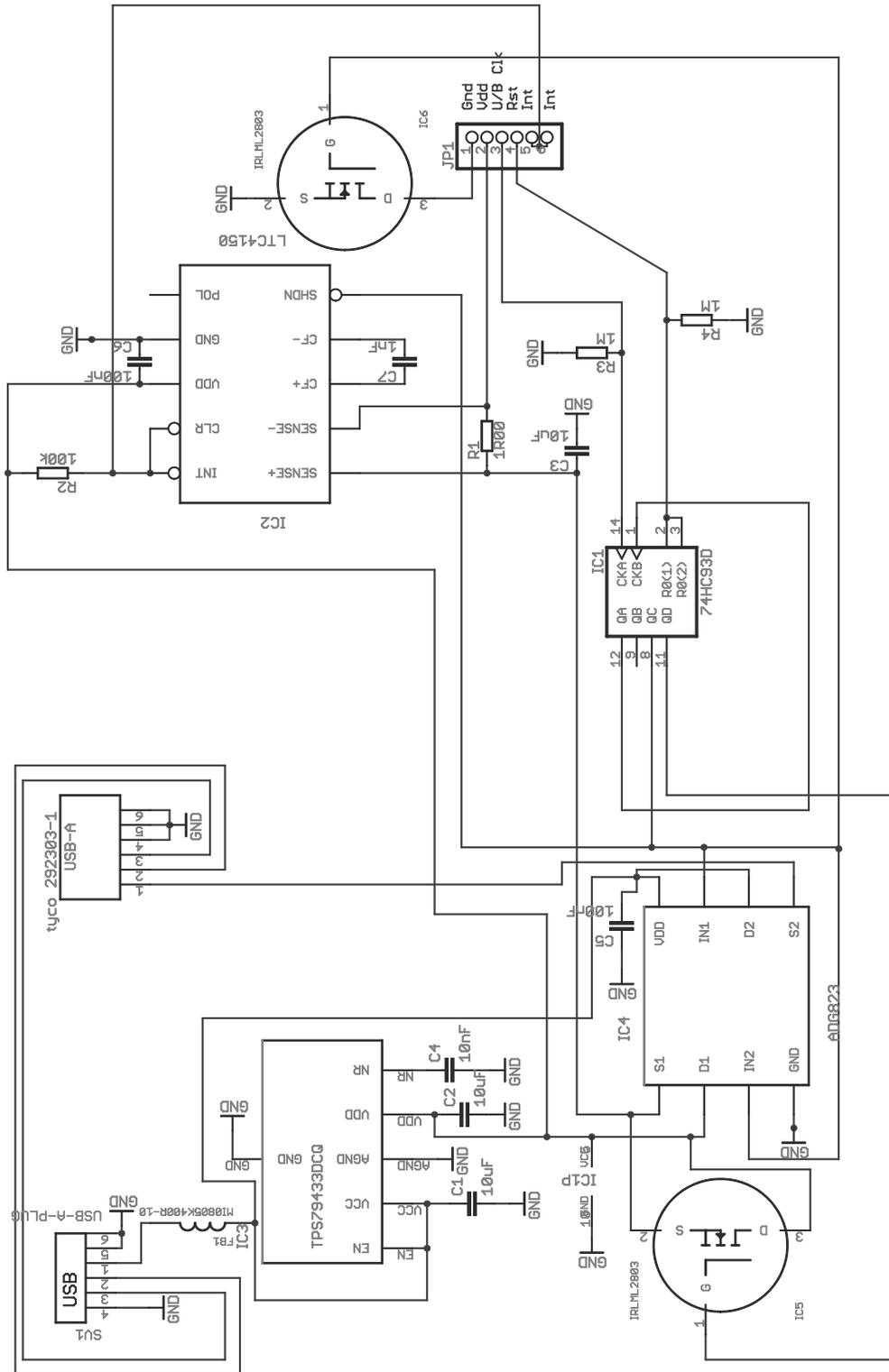
We described a precise and affordable measurement circuit that can be attached to sensor nodes that have a USB interface, a battery connector and three free pins on their pin header. We showed that it delivers precise results for a wide range of sensor node programs, and are currently equipping our testbed with it.

7 Acknowledgements

We would like to thank our colleagues in the Condell project Prof. Ruedi Seiler, Jan Sablatnig, Sven Grottke and Jiehua Chen for their help in preparing this paper.

A Schematic

A.1 Drawing



A.2 Explanations

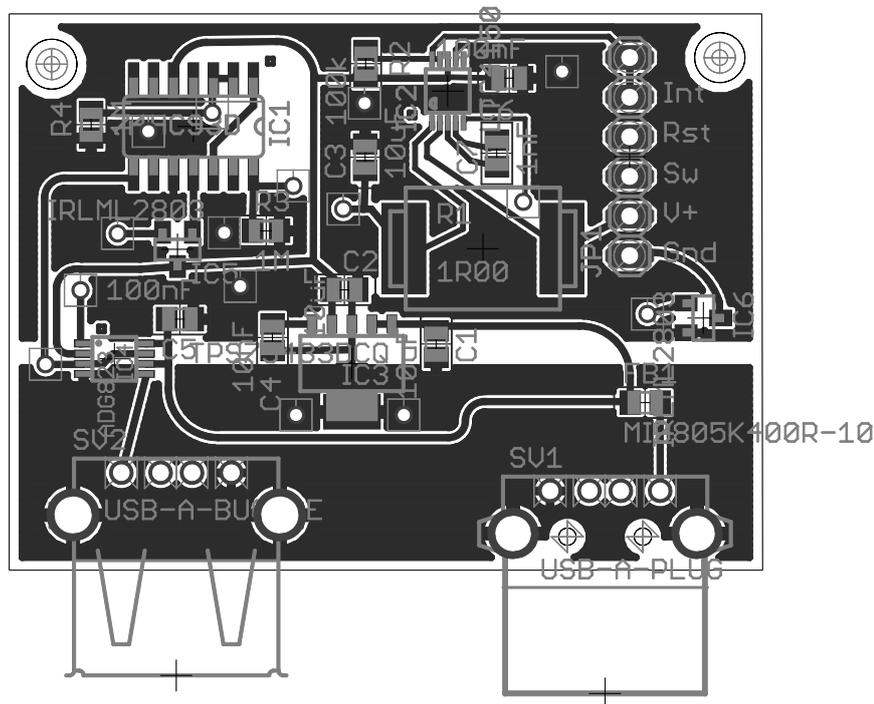
Many parts of the schematic just wire the main chips (the AD823 an Analog Devices audio switch, the TPS79433 a Texas Instruments voltage regulator – we choose the 3.3V option for our testbed, the LTC4150 an Linear Technologies coulomb counter and the binary counter 74HC93D here from NXP Technologies) together and hence need little explanation, some parts should be explained in more detail.

There are two NMOS gates in the schematic. IC5 allows current to flow to the battery connector (Vdd and Gnd on the pin header) even if the switch is turned off. This allows the brown out safe switch from measure mode to USB mode.

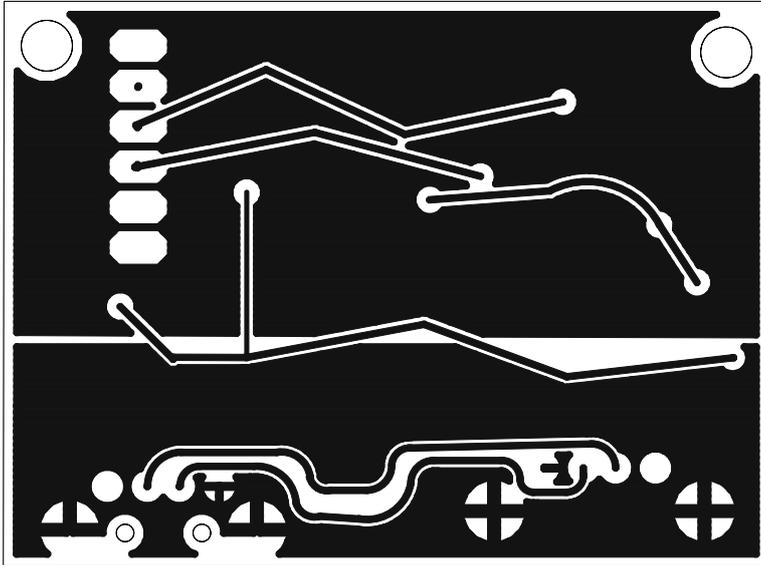
IC6 allows current to flow via the pin header Gnd *only* if the circuit is in measure mode. In any other case, the Gnd on the pin header is not connected. This prevents that current is pushed back from the node into the circuit.

R3 and R4 tie the input gates of the counter to Gnd. This is necessary: if the node does not switch the pins that control the operation of the circuit into output mode, this would result in two CMOS inputs being connected. They would pick up any noise, resulting in a lot of switch operations. The resistors prevent that by tying them to Gnd.

B Top layout



C Bottom layer



D Bill of Materials

| <i>Part</i> | <i>Value</i> | <i>Package</i> | <i>Description</i> |
|-------------|----------------|----------------|---|
| C1 | 10uF | C0805 | Capacitor |
| C2 | 10uF | C0805 | Capacitor |
| C3 | 10uF | C0805 | Capacitor |
| C4 | 10nF | C0805 | Capacitor |
| C5 | 100nF | C0805 | Capacitor |
| C6 | 100nF | C0805 | Capacitor |
| C7 | 1nF | C0805 | Capacitor |
| FB1 | MI0805K400R-10 | FB0805 | Ferrite Bead |
| IC1 | 74HC93D | SO14 | Decade, divide by twelve and binary COUNTER |
| IC2 | LTC4150 | MSOP10 | Linear Technology coulomb counter |
| IC3 | TPS79433DCQ | SOT223-6 | low dropout, low noise voltage regulator |
| IC4 | ADG823 | MSOP8 | low resistance, dual switch |
| IC5 | IRLML2803 | SOT23 | HEXFET gate |
| IC6 | IRLML2803 | SOT23 | HEXFET gate |
| JP1 | | 1X06 | PIN HEADER |
| R1 | 1R00 | SMR | SMD SHUNT RESISTOR |
| R2 | 100k | R0805 | Resistor |
| R3 | 1M | R0805 | Resistor |
| R4 | 1M | R0805 | Resistor |
| SV1 | USB-A-PLUG | | USB-A-PLUG |
| SV2 | USB-A-BUCHSE | | USB-BUCHSE |

References

- [1] Adam Dunkels, Fredrik Osterlind, Nicolas Tsiftes, and Zhitao He. Software-based On-line Energy Estimation for Sensor Nodes. In *Fourth Workshop on Embedded Networked Sensors*, Cork, June 2007.
- [2] Vlado Handziski, Andreas Köpke, Andreas Willig, and Adam Wolisz. TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *REALMAN '06: Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 63–70, New York, 2006. ACM Press.
- [3] Xiaofan Jiang, Prabal Dutta, David Culler, and Ion Stoica. Micro Power Meter for Energy Monitoring of Wireless Sensor Networks at Scale. In *Proc. of International Conference on Information Processing in Sensor Networks*, April 2007.
- [4] Linear Technologies. *LTC4150 Coulomb Counter / Battery gas gauge*. www.linear.com.