Automatic Energy Model Generation with MSP430 EnergyTrace

Birte Friesel
birte.friesel@uos.de
Universität Osnabrück
Osnabrück, Germany

Lennart Kaiser
lekaiser@uos.de
Universität Osnabrück
Osnabrück, Germany

Olaf Spinczyk
olaf@uos.de
Universität Osnabrück
Osnabrück, Germany

ABSTRACT

Professional energy measurement systems are expensive, especially when it comes to systems usable for automated measurements. In exchange, they provide a well-known measurement range, detailed accuracy guarantees, and a computer interface. DIY solutions from researchers have improved the situation considerably, with hardware cost in the $100 range, but are typically not commercially available. We are interested in even more affordable solutions, and examine the capabilities of the "EnergyTrace" technology embedded on the TI MSP430FR5994 LaunchPad, which is commercially available for less than $20. Out of the box, we observe a maximum error of 210 µA in the 100 µA to 10 mA range, but no support for automatic model generation. With single-point calibration and a custom synchronization and drift compensation algorithm, we are able to further reduce the error to 53 µA and perform entirely automated measurements and energy model generation on 3.3V MCUs and peripherals with a maximum timestamp error of 0.95 ms.

CCS CONCEPTS

• **Hardware** → **Power estimation and optimization**; • **Software and its engineering** → *Embedded software*; • **Computing methodologies** → *Modeling methodologies.*

KEYWORDS

energy models, energytrace, automation, energy measurements

ACM Reference Format:
Birte Friesel, Lennart Kaiser, and Olaf Spinczyk. 2021. Automatic Energy Model Generation with MSP430 EnergyTrace. In *Benchmarking Cyber-Physical Systems and Internet of Things (CPS-IoTBench '21), May 18, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3458473.3458822

1 INTRODUCTION

Energy models are a useful utility for designers of low-power embedded systems, as they allow reasoning about the expected battery lifetime or the required minimum energy-harvesting capabilities for various usage scenarios even before the hardware has been manufactured. They often take the form of functions or automata. Functions provide the total power consumption of a system component for a set of parameters such as radio transmit power or CPU sleep state [11]; automata describe the different states and configurations a hardware device may be in, the corresponding power consumption, and transitions between states [15]. Hybrid models with configuration-dependent power functions for different states and transitions are also possible [6].

However, lacking a repository of energy models for ultra-low-power MCUs and peripherals, and faced with an often insufficient level of detail in manufacturer datasheets, being able to generate energy models is equally important as knowing how to use them [7]. A high level of automation is desirable to increase reproducibility and minimize cost and human error sources.

Model generation works by having a benchmark program exercise (a sufficient subset of) all hardware configurations and gathering energy traces while doing so. The energy readings are mapped to the different configurations and distilled into functions and/or automata. A hardware configuration is made up of hardware state (e.g. receive, transmit, idle, or sleep) and configuration parameters (e.g. transmit power, packet size, or screen brightness).

Mapping energy readings to configurations (i.e., identifying benchmark events such as function calls and corresponding hardware state changes in the traces) can be done manually by looking for changes in the hardware's energy behaviour. However, if two consecutive hardware configurations have nearly identical energy consumption, or the effects of a configuration change do not manifest immediately, the resulting model may be inaccurate. In this case, and also when performing automatic model generation, the measurements must be augmented with synchronization signals which can be used to determine configuration changes. Typically, this is done by logging the state of an IO pin with each energy sample, and having the benchmark toggle it to indicate changes.

In our experience with automating energy model generation for embedded system components, measurement hardware which is suitable for automatic model generation isn't easy to come by. It is often expensive, either in terms of money (hundreds to thousands of dollars), or time (weeks spent building and evaluating a measurement system tailored towards a specific task).

With hands-on student labs and work-from-home setups in mind, we are interested in less expensive alternatives, which do not require a custom design or substantial soldering skills, but still provide usable results. This led us to take a look at the EnergyTrace technology embedded in TI's MSP430 LaunchPad evaluation board series, which are commercially available for less than $20. They augment a 16-bit ultra-low-power MSP430 MCU with programming, debugging, and energy measurement capabilities – without any additional hardware requirements. However, they do not support the kind of synchronization signals typically used for automatic model generation.

As we are not aware of publicly available accuracy figures, we obtained them using MSP430FR5994 LaunchPads and present them

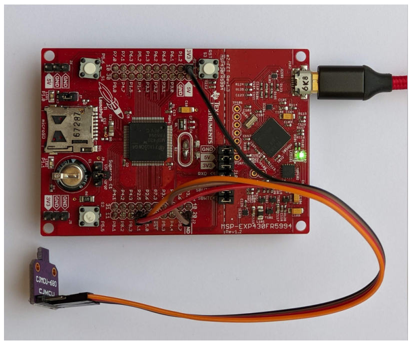


Figure 1: An MSP430FR5994 LaunchPad (top) performing energy measurements of a BME680 air quality sensor (bottom left). No further components are required.

in section 3. Additionally, we contribute a synchronization and drift compensation algorithm usable with both EnergyTrace and similar measurement devices without built-in synchronization support. We show that, utilizing single-point calibration and our synchronization technique, automatic energy model generation is possible with maximum errors of 53 µA and 0.95 ms, at a cost of less than $20.

The next section gives an overview of EnergyTrace's measurement concept and software ecosystem. We evaluate its energy measurement accuracy in section 3, and follow up with our synchronization method in section 4. After looking at related work in section 5, we conclude in section 6.

2 ENERGYTRACE

EnergyTrace is made up of two parts: The control and measurement circuitry and firmware on the LaunchPad, and a client library.

2.1 Hardware

An MSP430FR5994 LaunchPad consists of three MSP430 MCUs: the MSP430FR5994 itself (target MCU), an MSP430F5528 (host MCU) connected to the microUSB port for programming, debugging, and USB-to-UART conversion, and an MSP430G2452 (EnergyTrace MCU) connected to a DCDC converter [13]. Fig. 1 shows a LaunchPad performing measurements on a BME680 air quality sensor. The EnergyTrace MCU and DCDC converter are located in the bottom right PCB corner, the MSP430FR5994 target MCU is in the middle.

The DCDC converter is responsible for converting 5V, provided via USB, to 3.3V for the MSP430FR5994 target and its peripherals. Essentially, it is an inductive charge pump, as shown in Fig. 2: an inductor (L) and capacitor (C) store energy and provide it to the consumer (+3V3). A controller (MSP430) operates a transistor switch (Q) between the inductor and the 5V supply voltage to provide a recharge pulse to the inductor whenever the output voltage drops below a threshold. Pulse width and threshold are configured so that the output voltage is regulated in a tight range around 3.3 V.

Each recharge pulse transfers a constant amount of energy, E_{pulse}, which the firmware calibrates with three on-board resistors (2.2, 3.3, and 6.8 kΩ, corresponding to 0.485, 1.0, and 1.5 mA)

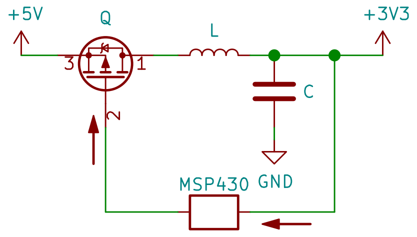


Figure 2: Simplified schematic of the EnergyTrace DCDC converter [13]. The MSP430 EnergyTrace MCU forms a feedback loop by monitoring the output voltage (+3V3) and regulating the inductor L (using transistor Q) in response to it.

before each measurement sequence. The amount of energy transferred in a time interval is $\#_{\text{pulses}} \cdot E_{\text{pulse}}$ [5, 13]. A similar method has been used in 2008 by iCount [4].

Counting charge pulses has a major advantage over more prevalent shunt-based circuits, which measure current by monitoring the voltage drop over a shunt resistor. Shunt-based methods can only estimate the energy consumption by interpolation of current measurements and may miss energy spikes which are shorter than the measurement interval. To minimize the risk of this happening, a high sample rate is required. Charge counting, on the other hand, measures energy and not current. By design, it cannot miss spikes, regardless of its sample rate.

The EnergyTrace-monitored 3.3V rail is not just connected to the target MCU, but also available on 3V3 pins. Thus, any 3.3V-compatible device can be connected to the EnergyTrace circuit for measurements. The LaunchPad can also act as USB-to-UART bridge by disconnecting the UART jumpers between host and target MCU (visible in the middle of Fig. 1) and attaching an external MCU instead. Disconnecting the MSP430FR5994 MCU entirely is not possible, however – it must be present on the debug port for EnergyTrace to work, so measurements will be offset by its sleep mode current draw of about 2.4 µA (8 µW).

2.2 PC Software and USB Interface

EnergyTrace is meant to be used with TI's Code Composer Studio or IAR Embedded Workbench IDEs, which is useful for developers, but not so much for automated measurements. Instead, we rely on TI's closed-source msp430 client library to provide an API to EnergyTrace features, the open-source EnergyTrace CLI `energytrace-util`[1] for API access, and our own `msp430-etv`[2] wrapper with additional post-processing and analysis features.

The API allows starting/stopping EnergyTrace measurements and providing a callback function, which is called periodically to handle new EnergyTrace readings. Measurements are passed as a list of events; each event i consists of a timestamp T_i (32 bit, 1 µs resolution), current I_i (32 bit, 1 nA), voltage U_i (16 bit, 1 mV), and cumulative energy E_i (32 bit, 100 nJ).

Cumulative energy is the amount of energy transferred since the start of the measurement. By default, the target MCU is reset

[1]https://ess.cs.uos.de/git/software/energytrace-util
[2]https://ess.cs.uos.de/git/software/msp430-etv

when starting a measurement, so it is identical to the energy used for program execution.

As each measurement describes total time and energy usage up to the corresponding timestamp, we calculate energy consumption and duration for each pair of measurements. For measurement i, we consider the interval $[T_{i-1}, T_i]$ with

- duration $\Delta T_i = T_i - T_{i-1}$,
- energy consumption $\Delta E_i = E_i - E_{i-1}$, and
- reported mean current I_i.

The current I_i is not measured by the EnergyTrace MCU, so it must be calculated either on the LaunchPad or by the msp430 library.

3 BASELINE EVALUATION

We examine the credibility of values provided by the API in general, and the accuracy of energy readings in detail. All measurements were performed at room temperature on eight identical MSP430FR5994 Rev 1.2 LaunchPads bought between 2016 and 2020, using firmware version 31200000 and libmsp430.so version 31200004. They were connected to a notebook computer, which was being used for work close to the LaunchPads most of the time.

3.1 API Data

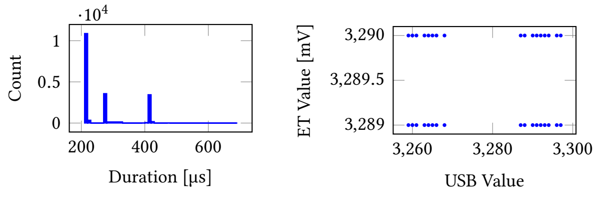
Preliminary tests indicated that the EnergyTrace firmware and/or client library may perform post-processing (e.g. low-pass filtering) of measurement properties before providing them via its API. We assess this by comparing them with raw USB traffic.

3.1.1 Benchmark. Our benchmark program toggles one of the LaunchPad's on-board LEDs and writes a line to UART once per second. It keeps the MCU in a low-power sleep state inbetween. We used the Wireshark utility to monitor USB traffic between host and LaunchPad while performing EnergyTrace measurements and compare data reported by the API with raw USB data and the benchmark's expected behaviour.

3.1.2 Observations. EnergyTrace data is transmitted in bulks of five samples each at a rate of about 750 Hz (giving a mean sample rate of 3.75 kHz). Each sample contains a timestamp, voltage, and cumulative energy reading. UART data is buffered and flushed once 64 bytes have been accumulated or a timeout of at least 60 ms has expired. Both EnergyTrace and UART transmissions use a low-priority USB block transfer mode without real-time guarantees, and are also sent over different USB endpoints. UART messages are therefore not a suitable synchronization method.

EnergyTrace samples are not equidistant. Instead, the first sample in a bulk transfer block typically has a difference of 400 µs to its predecessor, whereas the remaining four mostly map to two clusters around 210 and 270 µs, and outliers go up to 690 µs. See Fig. 3(a) for a histogram of observed ΔT_i values. Timestamps also appear to be transmitted with varying resolution. More than 99 % of durations (ΔT_i) have 650 ns per least significant bit, with additional clusters around 1 µs, 1.5 µs, and 2.5 µs.

It also appears that the EnergyTrace library performs nontrivial post-processing of received samples before passing them to API consumers. Notably, the relation between a single voltage sample in USB traffic and the corresponding data point returned by the API,



(a) Histogram of sample durations reported by EnergyTrace API; bin size is 10 µs.

(b) Relation of values observed in USB traffic to voltages reported by the EnergyTrace API.

Figure 3: Voltage and Duration values reported by the API.

and the relations between time and energy deltas in USB traffic and API data, are neither left- nor right-unique.

The cumulative energy values in USB traffic are not monotonic: About one in 50,000 samples is lower than its predecessor, indicating a (highly unlikely) *negative* energy flow. The corresponding API values are monotonic and consistent with expectations, however. The mean resolution of energy data is 365 nJ per least significant bit. In periods with very low power consumption, individual samples may report $\Delta E_i = 0$ due to the DCDC recharge frequency falling below the sample rate. Here, an energy difference of 1 (in USB traffic) corresponds to an API value of either 200 or 300 nJ.

For voltage readings, USB samples are likely transmitted as millivolt values. We observe values from 3259 to 3297 in USB data, but API samples only contain 3289 and 3290 mV in our benchmark (see Fig 3(b)). This may be caused by a low-pass filter implemented in the library. The mean deviation of API voltage from USB voltage (assuming millivolt samples) is 11 mV.

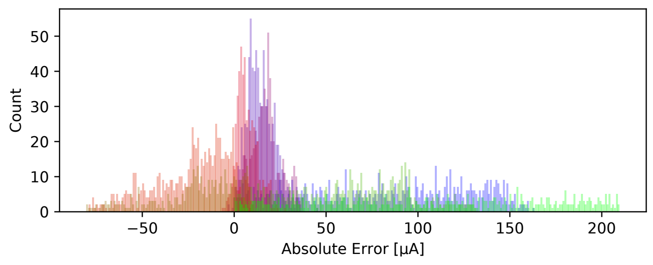
As USB samples do not contain current readings, we conclude that they are calculated on the host computer running the EnergyTrace library. We find that they are filtered and downsampled to 10 to 750 Hz depending on power consumption. Therefore, we discard current data provided by the API from now on, and calculate current and power as follows: $I_i = \frac{\Delta E_i}{U_i \cdot \Delta T_i}$ and $P_i = \frac{\Delta E_i}{\Delta T_i}$.

3.2 Energy Measurement Accuracy

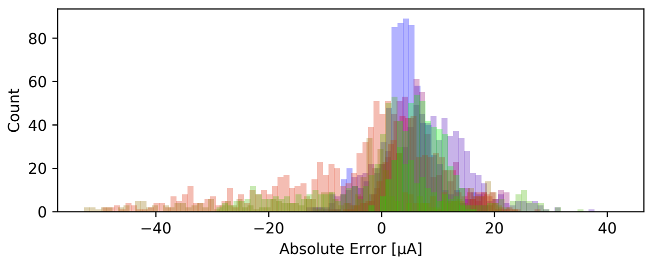
We now examine the accuracy of energy readings reported by the EnergyTrace API. As preliminary tests showed that the measurement range of our boards is limited to about 25 mA, we performed benchmarks in the 0 to 10 mA range, and calculated the EnergyTrace current as noted above.

3.2.1 Benchmark. To minimize error sources, we did not use on-board peripherals or benchmark programs here. Instead, we put the MCU into a low-power sleep mode (LPM2, without wake-ups) and connected a programmable current sink in parallel, using the LaunchPad's GND and 3V3 pins. We note that data is offset by the MCU's sleep mode current draw of about 2.4 µA (8 µW).

The current sink is a Keysight N6785A Source / Measurement Unit (SMU) in a Keysight N6705B DC Power Analyzer. We configured the built-in arbitrary waveform generator to control the sink current using a trapezoid function. After a ten-second hold at 0 mA, it ramps up to 10 mA (33 mW) over a 90-second interval, holds there for 20 seconds, and ramps down to 0 mA, again over



(a) On-board calibration



(b) On-board and external calibration

Figure 4: Histogram of EnergyTrace measurement error. Colors indicate different LaunchPad boards; bin size is 1 µA.

a 90-second interval. This cycle repeats three times, leading to a total benchmark duration of 14 minutes per device. As the SMU can only source and sink a discrete set of voltages and currents, ramp-up and ramp-down consist of 99 constant-current steps. Each step takes 909 ms; the step-width is 101 µA (333 µW). Voltage and sink current are logged to a USB drive.

To avoid interference from a ground loop between computer and power analyzer, we connected the LaunchPad to a computer running on battery power, without wired Ethernet. We inserted a low-dropout Schottky diode between the LaunchPad's 3V3 pin and the SMU to ensure that it does not damage the LaunchPad by back-powering it when the sink current is 0 mA. We did not take measures to minimize electro-magnetic interference (EMI), e.g. by placing the setup inside a metal cage, as we expect it to be used without such measures in practice as well.

For an additional verification measurement, we replaced the LaunchPad with an N6784A current source (built into the same power analyzer as the N6785A sink) and logged both sink and source currents. As the SMUs have a current measurement accuracy of 0.025 % + 10 µA, the difference between source and sink current should be limited to 25 µA at a sink current of 10 mA, and 20 µA in the µA range.

We use the PELT changepoint detection algorithm provided by the Python3 `ruptures` module to automatically split measurements into constant-current steps for analysis [9, 14].

3.2.2 Observations. With just the on-board calibration automatically performed by EnergyTrace, we observe a maximum absolute error of 210 µA. Relative error is up to 6.9 % at 100 µA and nearly constant at up to 2.9 % between 500 µA and 10 mA. Some boards manage an error below 1 % in this range. Refer to Fig. 4(a) for individual error distributions.

When using the 10 mA measurement for single-point calibration, absolute error decreases to 53 µA, and the maximum relative error is 5.2 % below 500 µA and 1.2 % above. See Fig. 4(b) for details.

In the reference measurement, with the power analyzer used both as source and sink, we observe an offset of up to 43 µA in the range below 800 µA. This is twice the expected maximum error of 20 µA and may be caused by a ground loop inside the chassis, and/or EMI. Beyond 800 µA, maximum deviations are 33 µA and 2.3 %, respectively. As this is only slightly higher than expected, we consider the EnergyTrace accuracy results presented in the preceding paragraphs to be sound.

4 SYNCHRONIZATION

Automatic model generation relies on a synchronization mechanism to map benchmark events (e.g. function calls) to measurement timestamps. Preferably, a measurement system provides a digital input which is sampled with the same frequency and time base as the energy readings. Thus, connecting an IO pin to the digital input and toggling it on each benchmark event is sufficient. Combined with a log of benchmark events, which is periodically dumped via UART or another suitable channel and also automatically analyzed, this allows for mapping IO events (indicating *when* something happened) to log entries (*what* happened) without user input, even when dealing with nondeterministic hardware behaviour (e.g. interrupts depending on radio transmission success/failure).

In our case, events are the start and end of transitions of an automaton describing a single hardware device. Transitions are typically caused by driver functions or signalled using interrupts, so start and end of a transition coincide with start and end of the corresponding function. Once the automaton and its interaction with driver functions and hardware configuration (e.g. radio bit rate) have been specified, model generation with our framework is entirely automatic. For details, we refer to [6].

EnergyTrace does not provide a digital synchronization input. We determine the start/end time of the benchmark using *in-band* synchronization, and use one of the MSP430's built-in counters to obtain relative timestamps for each state and transition inbetween.

For in-band synchronization, our benchmarks use the LaunchPad's on-board LEDs to generate synchronization pulses with a well-defined duration and power consumption. There is one pulse at the start and one at the end of the benchmark.

The counter is connected to the 16 MHz CPU clock without dividers to achieve maximum accuracy. It starts when the benchmark start pulse ends, and its value is read and reset for each benchmark event. When the benchmark end pulse begins, it is stopped and also read. Between these pulses, it runs continuously. Therefore, the time (in seconds) between the two synchronization pulses is equal to the sum of obtained counter values divided by 16 MHz.

By logging the counter values for each event, including them in the UART dump, and automatically detecting the benchmark start/end synchronization pulses in the energy traces, we determine the corresponding EnergyTrace timestamp for each event and perform automatic model generation for all hardware states and transitions.

As EnergyTrace measures whole-system energy, UART dumps affect energy measurements and timing. However, they occur at deterministic points, and only when the system is in an idle state. We automatically identify and ignore them during model generation.

Cycle counter and EnergyTrace timer are fed by different (uncalibrated) clock sources at slightly different speeds. They may exhibit a different amount of clock drift, and respond to environmental effects (e.g. temperature fluctuations) in a different manner. We devised a drift compensation approach and use it to minimize the effect of drift on the synchronized event timestamps, as follows.

4.1 Drift Compensation

We detect changepoints in the power measurements within a 20 ms-window around each *uncompensated* event timestamp (as determined by the cycle counter) using PELT [9]. Assuming that the actual event timestamp coincides with a change in the device's energy consumption, each *changepoint* is a candidate for it.

Let $T_1, \ldots, T_n$ be the uncompensated timestamps of events 1 to n, and C_i the set of changepoints for each event $i \in \{2, \ldots, n-1\}$. Timestamps T_1 and T_n are synchronization pulses, which must have a drift of zero, so we define $C_1 = \{T_1\}$ and $C_n = \{T_n\}$.

We select the *compensated* event timestamps $\hat{T}_i \in C_i$ so that

$$\sum_{i=1}^{n-1} \left|(\hat{T}_i - T_i) - (\hat{T}_{i+1} - T_{i+1})\right|$$

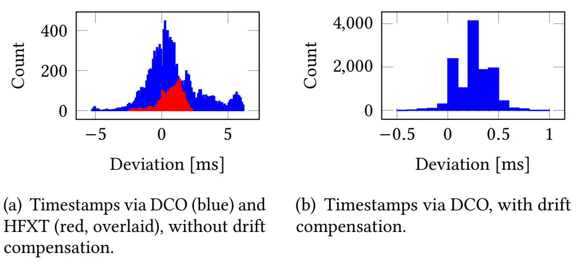
is minimal. This exploits that the evolution of clock drift is small for consecutive events (i.e., the difference between uncompensated and compensated timestamp should be nearly the same). It is based on the method used when manually analyzing benchmarks without synchronization signals, but much faster and more accurate.

To handle events which are too short or insignificant to be found by changepoint detection, we allow the timestamps $\hat{T}_i$ of individual events to be determined by the mean drift of their neighbours instead of a changepoint candidate from C_i. We add a 270 µs-penalty to the cost function (see above) for each event using such a calculated, non-changepoint timestamp to ensure that this only happens when no viable changepoints have been found. The penalty is equal to the mean measurement interval.

We now evaluate the accuracy of event timestamps obtained both with and without drift compensation.

4.2 Benchmark

We wrote a custom driver whose only job is to flash one of the LaunchPad's LEDs for a specific duration. It provides several driver functions, corresponding to flash durations between 100 µs and 10 ms. Its automaton consists of a single IDLE state (in which the MCU sleeps) and one transition for each function (i.e., for each flash duration). Energy traces obtained from automated benchmarks performed with this driver can be partitioned into two sets: If mean power is above 2 mW, either the LED is on and a transition function is being executed, or a UART dump is in progress. Otherwise, the LED is off and the automaton is in its IDLE state. Note that we chose the power threshold so that we can reliably detect 100 µs flashes, even though each sample covers an interval of up to 690 µs.



(a) Timestamps via DCO (blue) and HFXT (red, overlaid), without drift compensation.

(b) Timestamps via DCO, with drift compensation.

Figure 5: Error distribution histogram of reported event timestamps for 256-second benchmarks.

Again, UART activity is deterministic, so we automatically identify and remove it. This leaves us with precise timestamps for each transition independent of clock drift. We use these as ground truth.

Our measurement framework is deliberately unaware of this power partitioning scheme and only working with on-board timer data and benchmark start/end synchronization points. By comparing the transition timestamps calculated by our framework with the ground truth, we determine the timestamp error caused by clock drift during benchmark execution. Note that we may over-estimate the error by up to 690 µs due to the limited resolution of EnergyTrace measurements, as we have to assume that state/transition boundaries and EnergyTrace interval boundaries overlap.

We used three benchmark configurations with different idle times between transitions, corresponding to a total benchmark duration (that is, the time between start and end synchronization pulse) of 31, 82, and 256 seconds, respectively. Each of them performed 2500 transitions. We ran each benchmark sequentially on three different LaunchPads and performed a total of five benchmark runs per LaunchPad. We used the MSP430's built-in Digitally Controlled Oscillator (DCO) as 16 MHz clock source. It is the most accurate high-frequency clock available on the board, and also the default.

We performed additional measurements with a 16 MHz crystal as clock source. For this, we soldered the crystal and two 22 pF capacitors onto the HFXT pads of one of the LaunchPads.

4.3 Observations

With DCO and no drift compensation, we estimate a maximum drift of 83 ppm. However, our benchmarks (with a maximum distance of 128 s to the nearest synchronization point) do not exhibit the corresponding worst-case error of 10.62 ms; we observe up to 6.2 ms. With the crystal, drift decreases to 73 ppm (2.49 ms). Fig. 5(a) shows the error distribution across all observed event timestamps.

Drift compensation reduces the maximum error to 0.95 ms, regardless of clock source and benchmark duration (see Fig. 5(b)). Although transitions shorter than 1 ms are not reliably found by changepoint detection, our algorithm handles them well.

4.4 Alternatives

Some LaunchPads, including the MSP430FR5994 variant, support EnergyTrace++. Here, each sample also contains the current state of CPU and peripherals. As our benchmarks only wake up the CPU to perform a transition, its state is a reliable synchronization marker for the benchmark's progress. This eliminates clock drift issues, as energy and CPU state records use the same clock source.

However, EnergyTrace++ decreases the sample rate to 1.1 kHz – so it cannot provide higher timing accuracy than EnergyTrace with drift compensation – and increases MCU power consumption due to the debug interface requests needed to determine its state [5]. We were unable to obtain satisfactory results with it.

We also tested synchronization with a cheap external Logic Analyzer instead of the built-in cycle timer. Both uncompensated drift (105 ppm / 6.07 ms) and performance after drift compensation (0.94 ms) were similar to DCO measurements.

5 RELATED WORK

Most publications related to automatic model generation with off-the-shelf hardware we are aware of focus on smartphones and laptops, which have built-in measurement capabilities thanks to their battery management units. For example, DevScope models hardware states using a controller with 104 µA sensing resolution, 1 % measurement error, and an update rate of just 0.28 Hz [8].

An approach entirely without synchronization signals has been proposed by Cherifi et al. [3]. It uses changepoint detection only, and works even if no hardware timers are available. However, it is neither well-suited for events which do not cause an observable change in the energy consumption, nor for hardware whose energy behaviour changes independent of benchmark events.

The iCount system is also nearly off-the-shelf: By soldering a single connection to an already-present DCDC converter's feedback pin, it can measure energy with a maximum error of 20 % [4]. It uses nearly the same method as EnergyTrace, but is far less accurate.

Apart from that, research focuses on building and testing custom measurement devices. We give examples below.

EnergyBucket uses capacitors as charge pumps. Measurement error is less than 2 % in the 1 µA to 50 mA range with a consumption-dependent measurement interval ranging from 1152 to 0.005 Hz, at an estimated cost of $60 plus assembly and calibration time [1].

MIMOSA employs capacitors to mirror the device's energy consumption instead, achieving an error of less than 10 µA at a constant sample rate of 100 kHz [2].

FlockLab is a distributed testbed. Measurements are accurate within 10 % at 100 µA and less than 0.5 % beyond 500 µA, using a shunt resistor and a sample rate of either 28 kHz or 56 kHz [10].

RocketLogger provides a mobile measurement solution with shunt resistor and dynamic range switching. It is accurate within 0.09 % at up to 500 mA and a maximum sample rate of 64 kHz [12]. Hardware cost is about $50.

6 CONCLUSION

We have shown how to perform entirely automated measurements using an MSP430 LaunchPad's EnergyTrace API and a custom synchronization and drift compensation algorithm, applicable to any measurement device without built-in synchronization support.

Out of the box – without any external hardware – we observe a maximum error of 210 µA in the 100 µA to 10 mA range. Timestamps are affected by a clock drift of 83 ppm; we observe a maximum error of 6.2 ms during 256-second benchmarks.

With single-point calibration and our drift compensation algorithm, maximum error decreases to 53 µA and 0.95 ms.

The biggest drawback of EnergyTrace is its combination of low sample rate and lack of digital synchronization input. Our algorithm limits the resulting inaccuracy, but energy samples occurring immediately before or after a benchmark event may still be mis-attributed to the preceding or following state/transition.

While it is certainly not a suitable replacement for measurement devices and testbeds designed specifically for high accuracy and automation – and not meant to be one, either – we are positively surprised by the capabilities of such an affordable device.

We find that EnergyTrace is best suited for components with either negligible energy demand in transitions, or sufficiently long (multi-millisecond) transitions to make the mis-attribution error nearly irrelevant. As long as users are aware of its limitations, it is a handy tool not just for educational use, but also for comparative measurements. Users willing to trade 157 µA of additional error for convenience may even skip the calibration step.

REFERENCES

[1] Jacob Andersen and Morten Tranberg Hansen. 2009. Energy bucket: A tool for power profiling and debugging of sensor nodes. In *Third International Conference on Sensor Technologies and Applications (SENSORCOMM'09)*. IEEE, 132–138.

[2] Markus Buschhoff, Christian Günter, and Olaf Spinczyk. 2013. MIMOSA, a Highly Sensitive and Accurate Power Measurement Technique for Low-Power Systems. In *Real-World Wireless Sensor Networks*. Springer Berlin Heidelberg.

[3] N. Cherifi, T. Vantroys, A. Boe, C. Herault, and G. Grimaud. 2017. Automatic Inference of Energy Models for Peripheral Components in Embedded Systems. In *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*. 120–127.

[4] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler. 2008. Energy Metering for Free: Augmenting Switching Regulators for Real-Time Monitoring. In *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*. 283–294. https://doi.org/10.1109/IPSN.2008.58

[5] Brittany Finch and William Goh. 2014. *MSP430 Advanced Power Optimizations: ULP Advisor Software and EnergyTrace Technology*. Technical Report SLAA603. Texas Instruments. https://ti.com/lit/an/slaa603/slaa603.pdf

[6] Birte Friesel, Markus Buschhoff, and Olaf Spinczyk. 2018. Parameter-Aware Energy Models for Embedded-System Peripherals. In *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*. 1–4. https://doi.org/10.1109/SIES.2018.8442096

[7] D.C. Harrison. 2016. Busting myths of energy models for wireless sensor networks. *Electronics Letters* 52 (August 2016), 1412–1414(2). Issue 16.

[8] Wonwoo Jung, Chulkoo Kang, Chanmin Yoon, Donwon Kim, and Hojung Cha. 2012. DevScope: A Nonintrusive and Online Power Analysis Tool for Smartphone Hardware Components. In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis* (Tampere, Finland) *(CODES+ISSS '12)*. ACM, New York, NY, USA, 353–362. https://doi.org/10.1145/2380445.2380502

[9] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. 2012. Optimal detection of changepoints with a linear computational cost. *J. Amer. Statist. Assoc.* 107, 500 (2012), 1590–1598.

[10] Roman Lim, Federico Ferrari, Marco Zimmerling, Christoph Walser, Philipp Sommer, and Jan Beutel. 2013. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 153–165.

[11] Rahul Murmuria, Jeffrey Medsger, Angelos Stavrou, and Jeffrey M Voas. 2012. Mobile application and device power usage measurements. In *Sixth International Conference on Software Security and Reliability (SERE)*. IEEE, 147–156.

[12] Lukas Sigrist, Andres Gomez, Roman Lim, Stefan Lippuner, Matthias Leubin, and Lothar Thiele. 2017. Measurement and Validation of Energy Harvesting IoT Devices. In *Proceedings of the 2017 Design, Automation and Test in Europe Conference and Exhibition (DATE 2017)*. Lausanne, Switzerland.

[13] Texas Instruments 2016. *MSP430FR5994 Launchpad Development Kit (MSP-EXP430FR5994)*. Texas Instruments. https://ti.com/lit/ug/slau678b/slau678b.pdf

[14] Charles Truong, Laurent Oudre, and Nicolas Vayatis. 2020. Selective review of offline change point detection methods. *Signal Processing* 167 (2020), 107299.

[15] Hai-Ying Zhou, Dan-Yan Luo, Yan Gao, and De-Cheng Zuo. 2011. Modeling of node energy consumption for wireless sensor networks. *Wireless Sensor Networks* 3, 1 (2011), 18.