Lars Hanschke, Alexander Sowarka, and Christian Renner. Moving Task Scheduling to Co-Processors in Energy-Harvesting Systems. In Proc. of the 17th GI/ITG KuVS Fachgespräch Sensornetze, FGSN'18, Braunschweig, Germany, Sept. 2018.

Moving Task Scheduling to Co-Processors in Energy-Harvesting Systems

Lars Hanschke
Research Group smartPORT
Hamburg University of Technology
lars.hanschke@tuhh.de

Alexander Sowarka
Hamburg University of Technology

Christian Renner
Research Group smartPORT
Hamburg University of Technology
christian.renner@tuhh.de

***Abstract*—New applications for Wireless Sensor Networks (WSNs) pose new demands on energy-harvesting systems due to increased complexity and power consumption. However, microcontrollers with high computing power are over-dimensioned for energy-aware activity adaption. An assistant processor which schedules activity and monitors the energy state of the harvester can increase the energy efficiency of the whole platform. Therefore, we use a sleek communication interface for exchanging task information between assistant processor and sensor node. Even with communication overhead, a factor of 35 in energy can be saved, which allows the whole platform to increase energy efficiency.**

I. Motivation

The popularity of Wireless Sensor Networks (WSNs), based on still increasing computational power, new sensing capabilities [1] or sharing of WSNs [2], [3], offers a plethora of new application scenarios. Sensor nodes are equipped with multiple sensors, e.g. fine dust, humidity or ozone, and different radio interfaces, e.g. LoRa [4], WiFi [5] or IEEE 802.15.4. With an increasing number of different peripherals, the complexity of the underlying program structure grows steadily.

Supplying sensor nodes with ambient energy from renewable resources, e.g. solar energy, allows for reducing the environmental footprint of WSNs and also decreases maintenance costs. In many scenarios, energy harvesting allows perpetual operation, but if the energy budget is restricted, e.g. due to physical size limitation of node and energy storage, the consumption of the sensor node has to be adjusted carefully.

While application scenarios, such as body-area-networks show potential for intermittently-powered devices [6], surveillance-related systems require continuous operation. Common techniques, e.g. as presented in [7] and [8], ensure Energy-Neutral Operation (ENO) by adjusting the energy consumption of the sensor node minding both energy intake and current energy level. However, adjusting the activity of a sensor node, requires knowledge about future energy intake. For solar energy harvesting, various approaches exist, e.g. [9] and [10], which all share the necessity for up-to-date information on the current energy intake to learn characteristics. The periodic wake-ups to sample the generated power at the solar panel adds an energy overhead to the system but is mandatory for size-restricted energy-harvesting systems.

While applications become more complex and sensor nodes more capable but also more power-hungry, the energy overhead increases. Microcontrollers, which are designated for computing-intensive tasks, e.g. neural networks [11], operate very inefficiently for simple tasks, e.g. querying an ADC, due to overhead introduced by an operating system.

We argue that future sensor nodes powered by ambient energy need an assistant for energy purposes. This allows to easily exchange the microcontroller of the sensor node without adapting all energy-aware software. Additionally it allows for prolonged lifetime if power-hungry sensor nodes only concentrate on the designated task — rather than spending energy on adapting to harvesting intake. Approaches for adjusting the voltage of the node [12] or monitoring the energy-level [13] show the potential of assistant processors. However, a solution for energy-awareness including task scheduling is missing.

We show how an ultra-low-power microcontroller is used as a co-processor (COP) to assist the powerful but energy-expensive main processor (MP) in energy-aware behavior. Our approach encompasses the definition of a communication interface between both processors for task scheduling, computation of the energy-aware schedule for task execution and implementation of energy prediction schemes without the use of the MP.

II. Design

To reduce the overhead of using the MP for energy-awareness, we introduce our design concept for an assisting COP. We shift scheduling of tasks, as well as energy prediction and adaption of energy consumption to the COP.

The basic principle is as follows: during an initialization phase, the MP transmits information about its tasks to the COP, which builds a schedule satisfying time and energy constraints. Following, the MP queries the COP about which task to execute and how long it has to sleep before waking up again. This is repeated after execution of each task.

First, we introduce the main architecture and give an overview of the used peripherals. Second, we explain needed software components and third highlight central aspects of the communication interface between both processors.

A. Architecture & Overview

Similarly to our approach in [14], we consider an energy-harvesting platform using a solar cell, a supercapacitor and

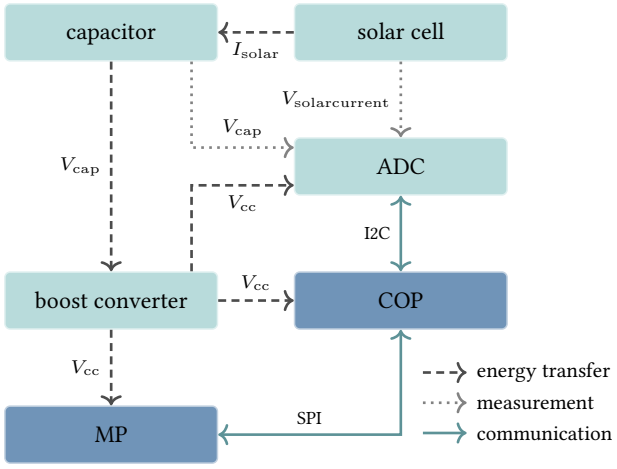

Fig. 1. Overview of the architecture; including communication between COP and MP as well as harvesting platform.

an external ADC. Figure 1 gives an overview of the platform. Incoming energy of the solar cell is directly fed into the capacitor. A small shunt resistor between solar cell and capacitor enables the ADC to measure the incoming energy of the solar panel. The energy level can be directly obtained from the voltage of the capacitor. However, our approach is not limited to this specific architecture as long as the COP has knowledge about the harvest intake and energy level of the sensor node.

The MP is configured by the developer to perform a distinct set of tasks. We assume a task to be an order of program commands, e.g. transmitting a packet, sampling a sensor or adjusting the air conditioning level. Since the application developer does not necessarily have knowledge about harvest prediction and adjustment of energy consumption, this responsibility is shifted to the COP.

To exchange information about tasks and scheduling, both processors are connected via a bus system of choice, e.g. I2C or SPI. Both follow the language of a communication interface, which we introduce in Section II-C.

B. Software Components

Our objective is to keep the software structure of the COP as flexible as possible; thus. we choose a layered software design, which is depicted in Fig. 2. Layers are only allowed to exchange information via defined interfaces.

In most microcontroller families, as the ARM Cortex M series [15], the manufacturer offers a Hardware Abstraction Layer (HAL) for interfacing with the microcontrollers and its peripherals. We rely on this library of STM, which allows for an easy exchange of COP hardware within the line of ARM Cortex M.

The COP implements four main components: `ADC`, `communication`, `scheduler` and `prediction`. Each component adheres to the layered design although not each component implements a functionality for each layer.

The `ADC` component handles interfacing with an external ADC. Methods for querying the voltage of the storage supercapacitor V_{cap} and the voltage at a shunt resistor $V_{\text{solarcurrent}}$ caused by the solar current, provide the information about the energy storage level and energy intake, respectively. The hardware layer ensures that the correct channel on the ADC is selected and the result is correctly converted to its physical representation, i.e. to millivolt and milliampere respectively.

The `prediction` component uses the energy readings to provide two important aspects: an estimation of the future energy intake and an energy budget ensuring depletion-free operation. Typically, prediction algorithms for energy harvesting, such as [9] and [10], give an estimate of the future energy intake within a fixed prediction horizon. This prediction horizon is additionally split into timeslots to decrease computational complexity. Our `prediction` component currently implements an Exponentially Weighted Moving Average (EWMA) filter with prediction horizon of 24 timeslots each spanning $T_{\text{slot}} = 1\,\text{h}$, as also used in [8]. Based on this estimate, algorithms adjust the activity of a sensor node by calculating the energy E, which can be taken from the capacitor without depletion. This energy $E = V_{\text{cc}} \cdot I_{\text{n}}^{*} \cdot T_{\text{slot}}$, with the assumption of a constant supply voltage V_{cc} is directly proportional to the average current I_{n}^{*}. In compliance with [8], I_{n}^{*} is called budget.

Based on the budget, the `scheduler` determines how many tasks the sensor node can fulfill within one timeslot. The scheduler models a task with a constant power $P = V_{\text{cc}} \cdot I_{\text{n}}$ which is consumed during the execution time t. In contrast to Dynamic Voltage Scaling (DVS) approaches like [12], this allows adjustment of the energy consumption only by altering the number of task execution. However, it is applicable without explicit changes in the supply voltage chain. Once the scheduler obtains the number of tasks to be executed, it evenly distributes the tasks within the timeslot to spread the current consumption. I_{n} and t per task have to be obtained from measurements by the developer and have to be transmitted during initialization phase of the communication between COP and MP, which we explain in the following.

C. Communication Interface

As mentioned in Section II-B, the communication between COP and MP is a very important aspect as it exchanges necessary information for task scheduling and energy budgeting. In general, two basic communication principles are possible: MP- or COP-initiated. While the latter allows the MP to shut off the Real-time Clock (RTC) circuit, it also requires wake-up capabilities on the communication bus, e.g. address matching. Since these vary between microprocessors, we opt for a MP-initiated communication interface.

1) Task Definition: To exchange information about the advised tasks, we implement a common task definition, which includes needed information for scheduling and energy consumption. Typically, the hardware configuration of a sensor node is known before deployment. Not necessarily all components have to be used simultaneously, but the capabilities of the node do not change over time.

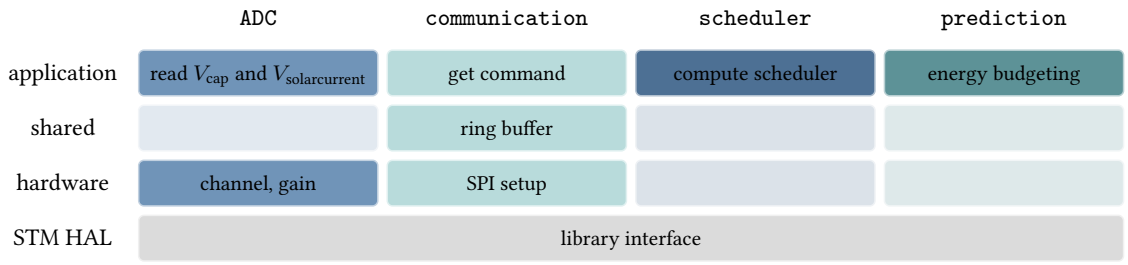

Fig. 2. Software layers of COP program; the hardware layer implements specialties of harvesting circuits, e.g. type of ADC or gain of the different channels; the shared layer contains information on the communication interface used by COP and MP; the application layer is independent of used hardware and easily exchangeable.

TABLE I
TASK PARAMETERS

name	description
state	μC-specific for power consumption, gives I_n
execTime	obtained at runtime or fixed
minInterval	time before execution is valuable again
maxInterval	maximum time after which execution is mandatory

This information is reflected in the *state* parameter for each task. An example for a task state is a turned-on radio interface and actively transmitting a packet. Right now, we assume that for each state, the developer has to measure the power consumption before node deployment. However, for future releases, we plan to measure this consumption online, e.g. by monitoring the decrease of capacitor voltage. Furthermore, the task definition contains time requirements: *execTime* reflects the time a sensor node needs to fulfill the advised task. It can be defined at compile time or measured at runtime by internal timers of the main processor. As Delay Tolerant Networking (DTN) [16] shows, tasks do not have to be executed as fast as possible — in most applications, a certain delay is tolerable. We reflect this matter by using a *minInterval* and *maxInterval* parameter for task definition. This allows the scheduler to effectively adjust the execution rate of task and thus adjustment of energy consumption. We summarize the parameters in Table I.

2) Initialization: Before the schedule is computed, the MP transmits the states and task definitions over the communication interface to the COP. At compile time, the COP does not know about the behavior of the MP. In this way, we ensure that a change of MP or the program structure of the MP does not affect the program of the COP. The initialization phase ends, when the MP may choose a prediction method, currently only EWMA, and forces the COP to compute the first schedule. If tasks change during runtime, e.g. due to remote commands [2], the MP is always capable of transmitting new tasks to its COP

3) Regular: At the end of each fulfilled task, the MP queries the COP about which task to execute next and how long to enter a sleep state before. The COP checks the list of next tasks, and answers with a previously announced unique task identifier and the sleep time before executing the task.

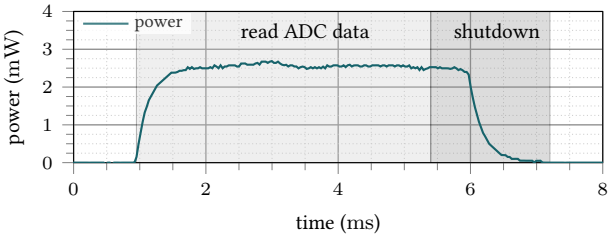

Fig. 3. Power consumption of the COP when querying the ADC of the platform; speed is limited by the conversion delay of the ADC.

The MP uses its internal timer to set the sleep time and, upon wake-up, executes the advised task.

III. RESULTS

The primary objective of the assistant COP is to reduce the energy cost for mandatory energy-aware measurements and calculations. This means that powering the COP and introducing the communication still has to consume less energy than powering the MP alone. Thus, we perform measurements of the different purposes of our platform and discuss the potential energy savings. We use the Espressif ESP32, with dual core Microcontroller Unit (MCU), up to 240 MHz clock frequency and integrated WiFi radio chip as the MP for our platform. The COP is a STM32L072RZ [15], which we run at 4.2 MHz. This allows to power down energy-costly clock circuits but still offers enough computing power. Both processors are supplied with $V_{cc} = 3.3\,\mathrm{V}$.

A. Measurements

We asses the power consumption of the COP by measuring the current at input side via an INA 139 measurement amplifier and record the time-varying current and supply voltage with the Keysight MSOX3014A oscilloscope.

1) Energy-Awareness: Due to the time overhead to boot the operating system, assessing the energy state of the platform is energy-intensive. We show the power consumption of querying the ADC in Fig. 3. Nearly constant 2.5 mW are consumed for 5 ms, while the execution is mainly limited by the communication with the ADC.

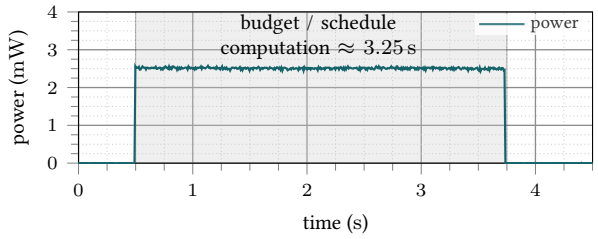

Fig. 4. Power consumption of the COP when calculating the energy budget for the next prediction horizon and calculate a schedule with four tasks.

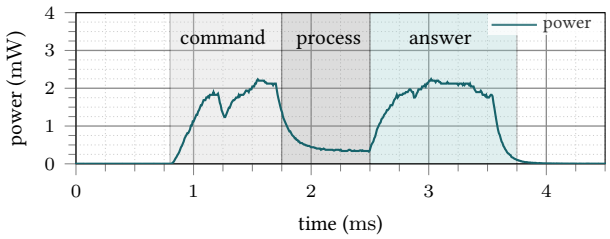

Fig. 5. Power consumption of the COP when communicating the advised sleep time and designated next task to the MP.

2) Scheduling: The energy level and harvest intake is used to perform the scheduling. This includes calculating the budget, but also distribute the advised tasks within the next timeslot. On the MP, this typically takes around 150 ms without booting the operating system. As shown in Fig. 4, the computation time increases by a factor of 22 to 3.25 s. However, the power demand of the ESP 32 while computing is roughly 100 mW — which is a factor of 40 more.

3) Communication: In contrast to scheduling and energy-awareness, both MP and COP have to be powered for communication. To decrease energy consumption, this period should be as short as possible. Right now, we implement a synchronized and thus blocking communication on MP side but we plan to add an asynchronous communication interface. During runtime, the energy overhead for communication is dominated by the regular exchange of wake-up information, i.e. sleep time and designated next task. Figure 5 depicts the power consumption of the COP while communicating with the MP.

B. Discussion

Surely, the usage of an assistant COP for energy-aware scheduling adds costs, due to the additional microcontroller, and complexity, due to the need for a communication interface. However, it decreases the energy consumption of power-hungry and complex sensor nodes and thus increases efficiency, i.e. performing more tasks with the same energy. Assuming minutely wake-ups to gather energy state and hourly re-computation of budget and schedule consumes roughly 7% of the energy stored in a 100 F capacitor with the ESP 32. In contrast, the combined approach with COP including regular communication with the MP only consumes 0.2% of the same capacitor — more than 35 times less.

IV. Conclusion

We showed the potential of an assistant processor for task scheduling for an energy-harvesting platform. Even with communication overhead, the energy savings due to monitoring of harvesting conditions, increase the overall platform energy efficiency. We plan to develop a harvesting platform for exchangeable MPs in future work to support spreading of energy-harvesting systems.

References

[1] C. Arora, N. Arora, A. Choudhary, and A. Sinha, "Intelligent Vehicular Monitoring System Integrated with Automated Remote Proctoring," in *Intelligent Communication and Computational Technologies*. Springer, 2018, pp. 325–332.

[2] J. Adkins, B. Campbell, B. Ghena, N. Jackson, P. Pannuto, and P. Dutta, "Energy Isolation Required for Multi-tenant Energy Harvesting Platforms," in *Proceedings of the Fifth ACM International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*, ser. ENSsys '17. ACM, 2017, pp. 27–30.

[3] T. La Porta, C. Petrioli, and D. Spenza, "Sensor-mission Assignment in Wireless Sensor Networks with Eergy Harvesting," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, ser. SECON '11. IEEE, 2011, pp. 413–421.

[4] M. Cattani, C. A. Boano, and K. Römer, "An Experimental Evaluation of the Reliability of LoRa Long-Range Low-Power Wireless Communication," *Journal of Sensor and Actuator Networks*, vol. 6, no. 2, 2017.

[5] L. Hanschke, J. Heitmann, and C. Renner, " Challenges of WiFi-Enabled and Solar-Powered Sensors for Smart Ports ," in *Proceedings of the 4th ACM International Workshop on Energy Neutral Sensing Systems*, ser. ENSsys '16. ACM, 2016.

[6] J. Hester and J. Sorber, "The Future of Sensing is Batteryless, Intermittent, and Awesome," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '17. ACM, 2017, pp. 21:1–21:6.

[7] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power Management in Energy Harvesting Sensor Networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 4, p. 32, 2007.

[8] C. Renner, S. Unterschütz, V. Turau, and K. Römer, "Perpetual Data Collection with Energy-Harvesting Sensor Networks," *Transactions on Sensor Networks (TOSN)*, vol. 11, no. 1, pp. 12:1–12:45, 2014.

[9] C. Renner, "Solar Harvest Prediction Supported by Cloud Cover Forecasts," in *Proceedings of the 1st ACM International Workshop on Energy Neutral Sensing Systems*, ser. ENSsys '13. ACM, 2013.

[10] A. Cammarano, C. Petrioli, and D. Spenza, "Pro-Energy: A Novel Energy Prediction Model for Solar and Wind Energy-harvesting Wireless Sensor Networks," in *IEEE 9th International Conference on Mobile Adhoc and Sensor Systems*, ser. MASS '12. IEEE, 2012, pp. 75–83.

[11] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher, "DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '17. ACM, pp. 4:1–4:14.

[12] U. Kulau, D. Bräckelmann, F. Büsching, S. Schildt, and L. Wolf, "REAPer – Adaptive Micro-Source Energy-Harvester for Wireless Sensor Nodes," in *Local Computer Networks Workshops (LCN Workshops), 2017 IEEE 42nd Conference on*. IEEE, 2017, pp. 1–8.

[13] C.-W. Yau, T. T.-O. Kwok, and Y.-K. Kwok, "Energy Gatekeeper Architecture for Enabling Rapid Development of Energy-Harvesting Internet of Things," in *Proceedings of the Fifth ACM International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*, ser. ENSsys'17. ACM, 2017, pp. 43–45.

[14] L. Hanschke, C. Renner, J. Brockmann, T. Hamann, J. Peschel, A. Schell, and A. Sowarka, " Light Insight — Emulation of Radiation Traces for Analysis and Evaluation of Solar-Harvesting Algorithms ," in *Proceedings of the 5th International Workshop on Energy Neutral Sensing Systems*, ser. ENSsys, Delft, Netherlands, November 2017.

[15] *Datasheet STM32L072x8*, STMicroelectronics, 9 2017, rev. 4.

[16] B. Gernert and L. Wolf, "PotatoScanner: Using a Field Sprayer As Mobile DTWSN Node," in *Proceedings of the 12th Workshop on Challenged Networks*, ser. CHANTS '17. ACM, 2017, pp. 47–49.