Undervolting in WSNs - A Feasibility Analysis

Ulf Kulau, Felix Büsching and Lars Wolf
Institute of Operating Systems and Computer Networks
TU Braunschweig
Email: [kulau|buesching|wolf]@ibr.cs.tu-bs.de

***Abstract*—The energy consumption of electric circuits depends on the applied voltage level. This is used by dynamic voltage scaling approaches where the voltage is lowered up to a datasheet specified level. To reduce the energy consumption even further, it would be possible to power the electric circuits below the specified voltage levels. Considering Wireless Sensor Nodes, this 'undervolting' would save a substantial amount of energy and, hence, would lead to a significant longer lifetime of nodes and networks. Contrariwise, operating processors or nodes outside their specifications adds some extra incertitude to the system. In this paper, we analyze the effects of undervolting for a typical wireless sensor node in theory and practice. A prototype implementation is used to characterize the influence of lower-than-recommended voltage levels on the MCU. In addition, the impact of different temperatures is considered as well as the behaviour of an undervolted transceiver unit and, therefore, the effects on the wireless communication. While classical computer applications may contain too many hazards to outweigh the improved energy consumption when using undervolting, we show that it is particularly suitable for Wireless Sensor Networks (WSNs) with a huge potential of saving energy and the opportunity of novel power management approaches on every layer.**

I. Introduction

Methods to enhance the lifetime of sensor nodes and networks are very important. This will also remain so in the future since the improvement of battery capacity does not follow Moore's Law. One of these approaches is Dynamic Voltage Scaling (DVS). This technique adapts the voltage level of the node to the actual system load to save energy. The background of DVS is that nearly all components of a sensor node are realized in Complementary Metal-Oxide Semiconductor (CMOS) technology. One of the advantage of CMOS is that the major part of the power dissipation belongs to the dynamic power consumption P_{dyn}. During static operations ($f_{cpu} = 0Hz$) only a negligible leakage current occurs. Furthermore, the dynamic power consumption has a linear dependency on the clock rate (f_{cpu}) and a quadratic dependency on the voltage level (V). Equation 1 shows this relation, where C_L is a parasitic capacity which depends on the quality of the manufacturing process.

$$P_{dyn} \quad = \quad C_L \cdot f_{cpu} \cdot V^2 \tag{1}$$

Unfortunately, the switching delay of CMOS gates grows at lower voltage levels V. Hence, a certain clock rate (f_{cpu}) needs a specific minimum voltage level (V) to guarantee that a switching operation finishes within $\frac{1}{f_{cpu}}$. DVS uses these relationships and adapts both, the clock rate (f_{cpu}) and the corresponding voltage level ($V(f_{cpu})$) to the actual needed system load. Recently, the advantages of DVS in the application area of WSNs have been studied by several groups, e.g., [1], [2], [3]. Furthermore, in [4] a prototype implementation of a DVS capable wireless sensor node is presented. With regard to equation 1, all these approaches have in common that the energy efficiency of a sensor node is increased – in comparison to a fixed voltage source.

All these strategies avoid unsafe areas of operation by satisfying the specifications for adjusting the voltage level $V(f_{cpu})$ as stated in the respective datasheets. Nevertheless, during some experiments with DVS in wireless sensor nodes, we observed that microcontroller systems are able to work below their recommended voltage levels as well. So, the question came up: What happens if we ignore the specified operating area respectively how far can we go outside it?

An important aspect to be considered for this is that the voltage levels given in the specifications cover the widespread temperature range of microelectronic components. Yet, the threshold voltage (V_{th}) is temperature dependent. Looking at this temperature dependency of CMOS gates in equation 2, a microcontroller has to have the ability to run below $V(f_{cpu})$ under *normal* conditions (whereas V_{th0} is the threshold voltage at the temperature T_0, α is a temperature coefficient) [5].

$$V_{th}(T) = V_{th0} + \alpha \cdot (T - T_0) \tag{2}$$

It might be argued that we inherently accept a decreased reliability of the sensor nodes, since undervolting may have an impact on the correct execution of a program or the proper functioning of a component. However, if the malfunction risk is limited, we stay very well within the typical WSN assumptions, i.e., that not a single node is of importance but the whole WSN must function properly and, thus, algorithms [6], MAC protocols [7] and routing algorithms [8] for WSNs are designed to be fault tolerant. For such cases, a certain number of nodes can become temporary inoperative without the malfunction of the whole network. Thus, undervolting may just be another cause for known symptoms – with a huge potential to save energy. As we demonstrate in this paper, the potential for energy demand reduction is indeed significant and can be in the order of 30 %.

The paper is structured as follows: In the next section, general related work regarding undervolting is discussed. Afterwards, we present preliminary studies we performed to analyze the requirements for a prototype implementation. In section IV we discuss the impact of undervolting on an micro controller unit (MCU) and evaluate the gain of energy efficiency. Following, effects on the wireless communication are presented in section V Finally, we conclude the paper.

II. Related Work

While several DVS approaches for WSNs are presented in various publications [1], [2], [3], [4], as far as we know *active undervolting* has not been studied for WSN so far. In some articles regarding general computer architectures (e.g. [9]) undervolting is mentioned as a possibility to reduce the power consumption of a processing unit. In [10] undervolting is explicitly specified as a technique which reduces the power dissipation of a Power7 CPU. It is shown that the entire system's power consumption drops by 15.8% up to 26.4%, without affecting the computational performance. This causal way of dealing with a possibly higher error rate is also mentioned in [11]. Through 'probabilistic computing' it was possible to increase the energy efficiency of CMOS gates up to 300%. Yet, a more application-oriented utilization of undervolting can be found in the end-user community of Personal Computers (PCs). The over-clocking scene aims to increase the performance of commercial off-the-shelf (COTS) CPUs. For that, the clock rate is set manually above the level suitable according to the voltage level specifications. To detect errors, a rather vague stress test (e.g. calculation of prime numbers) is used to check the stability of the CPU. Nevertheless, for the reliable operation of classical computer applications the usage of undervolting may contain too many hazards to outweigh the improved energy consumption. However, for several WSN scenarios the situation is different since there suddenly appearing or failing nodes may be the rule rather than the exception [12], [8]. Moreover, WSNs may be exposed to varying environmental conditions [13] which influence the absolute minimum operating voltage. In summary, to the best of our knowledge, there is no related work which considers undervolting as a regular technique to increase the energy efficiency of small embedded devices like sensor nodes and ensures the operation by adaptive methods.

III. Prototyping and Preliminary Studies

In general, the architecture of a wireless sensor node can be divided into three major parts: Processing unit, transceiver unit, and sensing unit. Mainly all components are based on CMOS technology. Thus, with regard to equation 1, a downscaled voltage level has a positive effect on the energy efficiency of all parts. Nevertheless, in this section we focus on the processing unit only. The *processing unit* is the central component which executes the software and is responsible to control all other parts of the node. Due to the low power constraints in WSNs, usually a small ultra low power MCU is used for this purpose. In this paper, we use the common ATmega1284p 8-Bit micro controller [14] for the analysis of undervolting. This MCU is used in existing sensor node platforms such as [15], [16].

A. Hardware Prototype

Based on the prototype presented in [4] we prepared a simple testbed. Figure 1 shows a block diagram of the implementation, whereby we borrowed the basic design and the voltage scaling module. In addition, a second tiny microcontroller has been added, which is called the 'Secure Instance' (Secure Instance (SI)). The ATmega1284p is undervolted and, thus, might not be reliable. Hence, the tiny co-microcontroller can be used to validate checksums or help to recalibrate the clock rate.

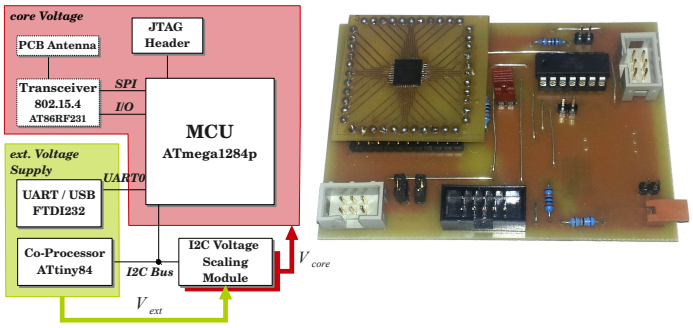


Figure 1. Block diagram and picture of the prototype implementation.

B. Recalibrating the Clock

It is necessary to observe the clock rate because the ATmega1284p generates the system clock with an internal RC-oscillator. The datasheet [14] shows a significant dependency between the oscillator speed and the voltage level even for recommended voltage levels. Thus, a stable main clock of the MCU cannot be guaranteed for this purpose. The first question is whether we will have to recalibrate the clock rate when we want to use undervolting. Therefore, we measured the deviation from a claimed clock rate (f_{cpu}) as a function of the deviation of the nominal voltage level with an oscilloscope.

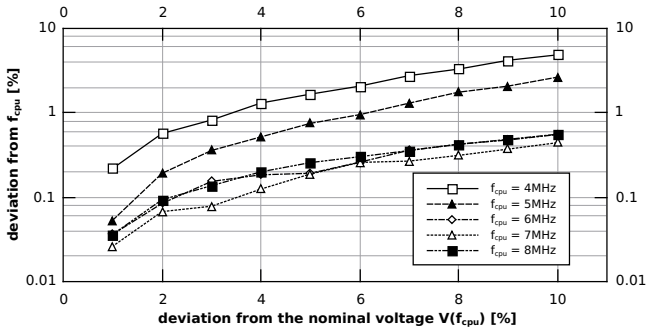


Figure 2. Clock rate: impact of undervolting and recalibration speed.

Figure 2 shows the impact on different clock rates; it can be seen that the deviation is significant. For example, we assume a clock rate of $f_{cpu} = 4MHz$ at the nominal voltage level $V(f_{cpu})$. With an undervolting of $V(f_{cpu}) - 10\%$ the oscillation of the internal RC oscillator decreases, so that the clock rate is reduced by about 4.86%.
A lower clock rate has minor effects on pure arithmetic logical unit (ALU) operations. Provided that no other errors occur through the undervolting, only the execution time rises. On the other hand, time critical processes like OS timer (e.g. within Contiki OS [17]) or asynchronous communication (e.g. Universal Asynchronous Receiver/Transmitter (UART)) are not reliable anymore. For example, the error rate of a UART communication grows. Under normal operation with $f_{cpu} = 4MHz$, $V = V(f_{cpu})$ and a baud rate of 9600 BAUD an error of $\varepsilon_{UART} < 0.2\%$ is negligible. The measured worst case of $f_{cpu} = 3.4655MHz$ at $V = V(4MHz) - 0.28V$ ends up in a communication error of $\varepsilon_{UART} < 16\%$ [14].
Thus, we have to recalibrate the clock rate during the usage

of undervolting. The idea is to check the clock rate of the undervolted MCU periodically with the SI. Based on the deviation within a period, the external instance can decide whether the clock rate has to be changed. This information is sent to the MCU which increments or decrements the internal RC-oscillator calibration register (OSCCAL). This linear recalibration approach is shown in figure 3 at a nominal clock rate $f_{cpu} = 4MHz$.

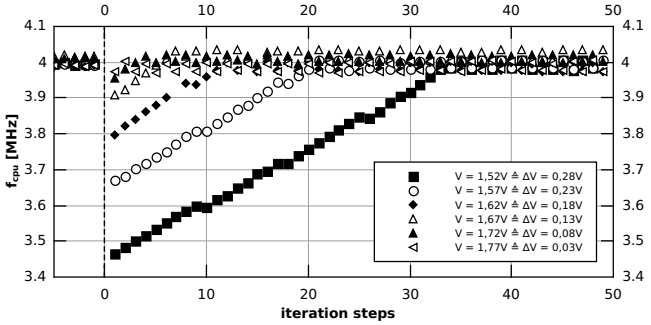


Figure 3. Clock rate: iterative recalibration of the clock rate.

We can see that it is generally possible to generate a stable clock rate even with undervolting. This result is important because it is a major requirement that undervolting should not influence the further usage of the MCU. Of course, the linear recalibration approach could be improved, so that the claimed clock rate could be reached in less iteration steps.

IV. Undervolting on MCUs - Characteristics

For the first tests we ignore a dedicated view on the actual temperature and define the room temperature $T_{lab} \approx 22°C$ as the normal operation condition. We ordered several ATmega1284p from three different distributors to minimize the risk of analyzing parts from the same product line. Based on the results of several tests, we show the behavior of an undervolted MCU in this section.

Various failures are possible when using undervolting on an MCU and they are not predictable without explicit knowledge about the internal structure of the integrated circuit (e.g. VHDL models). On a wireless sensor node the MCU controls all other parts so that a failure will effect the whole system. Accordingly, the question is how low the voltage level V can drop below the recommended $V(f_{cpu})$ without any recognizable effects on the practical usage of the MCU. Furthermore, for further generic solutions it is necessary to know, if every ATmega1284p has an individual probability distribution of failures or whether all are similar.

A full test of every component of the MCU is almost impossible. For example, the test of only one ALU operation (e.g. add) with $n = 2$ input variables of $m = 8Bit$ each would need a test vector of $(2^m)^n = 2^{16}$ cases. Thus, the observation of the ALU is done through a spot test. Related to [18] an adequate detection of calculation errors is done by a matrix multiplication. Nevertheless, it should be noticed, that an absolute failure detection is not necessary anyway. The fault tolerant behavior of a WSN should be able to handle sporadic failures even if we are not able to detect them here (cf. sections I, II). The peripherals tests are restricted to the Inter-Integrated Circuit (I^2C), Serial Peripheral Interface Bus (SPI), UART and General Purpose Input/Outputs (GPIOs). Theses checks are also only spot tests and should show that peripherals are generally able to run below the recommended voltage level. As described above, the clock rate was recalibrated continuously during the tests of the peripheral's functionality.

During all tests with undervolting up to a certain level no error of the peripherals occurs. We can observe that an operating point exists, where the ALU execution breaks down. The figures 4(a), 4(b) and 4(c) give the results of the functionality analysis for three different instances of the MCUs. From this we can derive some statements which help for the further considerations.

- In general it is possible to operate an MCU with undervolting. Up to a specific operating point, the probability that a failure occurs is very low.
- Malfunction starts to appear in a small, sharp region around this point. The characteristic curve for the output signal as a function of the input voltage correlates with the measured results [5].
- Although we use the same kind of micro controller (ATmega1284p), the absolute minimum voltage level of the three MCU instances differs. On MCU1 with $f_{cpu} = 4MHz$ the first errors occur at a deviation of $\delta_V = 12.22\%$ from $V(f_{cpu})$. At this point MCU2 and MCU3 run stable without any failures.
- The possible deviation from $V(f_{cpu})$, i.e., where no errors occur, grows with higher clock rates.

A. Power Consumption

The overall goal of this paper is to explore the feasibility and potential of undervolting to increase the average energy efficiency of a sensor node. Hence, the impact of undervolting on the power dissipation of the processing unit is a matter of particular interest. Since our approach operates outside the datasheet specifications, we have no information about the typical characteristics how the current consumption reacts on undervolting. Thus, we measured the current consumption of three MCUs at different clock rates f_{cpu} as a function of the deviation (in percent) from the recommended voltage level $V(f_{cpu})$.

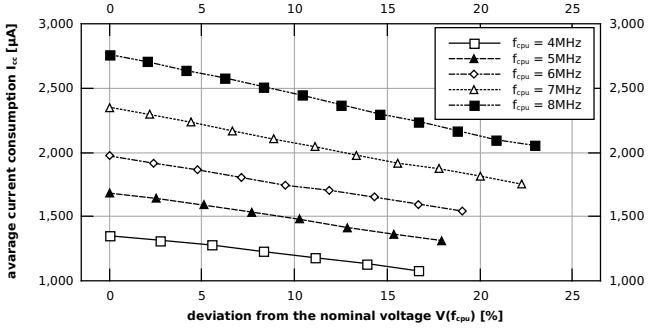
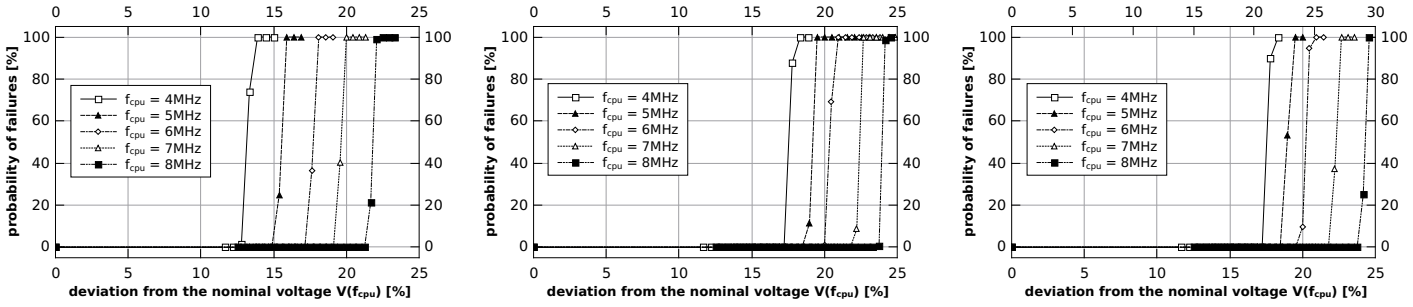


Figure 5. Current consumption vs voltage (mean value).

Figure 5 shows the average results of the measurements. As expected, it can be seen that the current consumption of an



(a) MCU 1 (b) MCU 2 (c) MCU 3

Figure 4. Functionality analysis for three different instances of the MCUs

.

MCU decreases with lower voltage levels. Independent of the clock rate f_{cpu}, the relation between the percental deviation δ_v and the percental saving of the current consumtion δ_i can be described as follows:

$$\delta_i \approx \frac{6}{5} \cdot \delta_v \tag{3}$$

A huge advantage of undervolting is that it has no other effects on the usage of an MCU. From this it follows, that we do not have any conflicts when using additional energy-aware concepts like dynamic power management (DPM), DVS, or low power listening (LPL). With regard to DVS, figure 6 shows the energy per clock cycle as a function of δ_v. This information

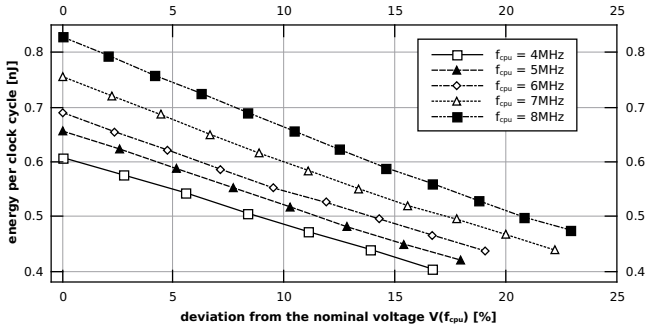


Figure 6. Energy per clock cycle vs voltage (mean value).

about the relation between energy, voltage, and clock rate is important because this influences the way how a DVS should be implemented. Basically there are two different strategies:

1) A task is executed at the highest clock rate to switch to the energy-aware sleep state as soon as possible.
2) The clock rate is adapted so that a task meets a deadline exactly.

In this case, i.e., for the measurement results we have, the second option would be the best choice. The reason is that the curves 'energy per clock cycle' of the different clock rates (cf. figure 6) do not show an intersection.
Table I summarizes the significant energy saving potentials of undervolting by stating maximum percental energy saving $max(\delta_e)$, average percental energy saving $mean(\delta_e)$, minimum energy per clock cycle E_{min}.

Table I. ENERGY SAVINGS WHEN USING UNDERVOLTING ON ATMEGA1284P COMPARED TO THE RECOMMENDED VOLTAGE LEVEL $V(f_{cpu})$.

$f_{cpu}[MHz]$	4	5	6	7	8
$E_{min}[pJ]$	404.0	420.2	437.0	466.8	475.0
$mean(\delta_e)[\%]$	16.60	18.30	19.06	20.02	22.27
$max(\delta_e)[\%]$	33.52	35.96	36.67	38.23	42.66

B. Influence of Temperature

The initial idea of undervolting for WSNs belongs to the presumption, that a MCU has to be tolerant against extreme temperatures, but this is not always needed. Hence, for the previous observations we assumed a 'normal' operating environment. The tests were done indoor with a moderate temperature of $T_{lab} \approx 22°C$. However, some application areas of WSNs have extreme temperature requirements, e.g., cold temperatures [19].

If a sensor node runs with undervolting and if the environmental temperature changes, the first question is in which direction we have to adapt the voltage level to avoid an increase of failures. To study the effects of temperature changes, we performed several experiments. For this, the MCUs under test are heated up and cooled down with a peltier element. As expected, we observed that a higher temperature T allows lower supply voltages. For example, the break-even point for MCU1 at $f_{cpu} = 4MHz$ moves from $\delta_V = 12.22\%$ ($T_{lab} \approx 22°C$) up to $\delta_V = 15\%$ ($T_{lab} \approx 50°C$).

Afterwards a climatic chamber was used to get more detailed information about the relation between the temperature and the break even point. The results in figure 7 validate the trend of the temperature dependency.

- A lower temperature increases the risk to get failures. The voltage level has to raise.
- A higher temperature involves the chance to increase the energy efficiency. The voltage level can be reduced to save more energy.

However, a deliberate temperature increase with the intention to save more energy through more extensive undervolting is not recommended. The aging of components accelerates with higher temperatures [20].

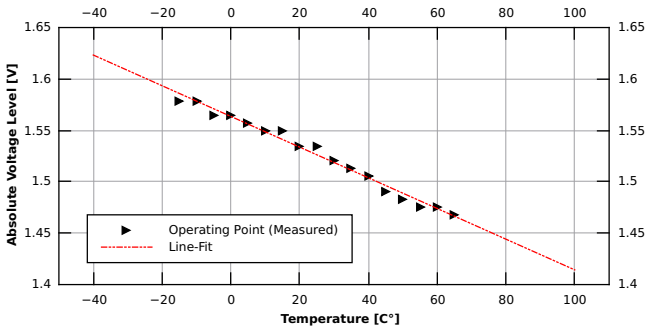


Figure 7. Minimum Voltage Level vs. temperature dependency.

C. Limitations and Interim Conclusions

Our approach is based on the ATmega1284p micro controller, but how about other processing units? In general, a wide spread temperature range of a component indicates potentials for undervolting (cf. equation 2). Otherwise, it is not possible to give a statement to every single micro controller whether it is able to be undervolted or not. We performed some short tests with a msp430 [21] of the TelosB [22] sensor node which showed that also this MCU can run below its recommended voltage level. Another question is, how other parts of a sensor node (sensors, actors) react on undervolting. With regard to the sensing quality an undervolting of sensors is not recommended. A sensor converts the physical world into electrical signals. These information is converted via an analog-digital converter which makes the data utilizable for the processing unit. To avoid that the analog section of the sensors could suffer from undervolting, our approach is bound to the core functionality of a sensor node: The processing and transceiver units. Therefore, sensing errors are not caused by a sensor itself.

A mainstay of our argumentation to use undervolting on wireless sensor nodes is that WSNs are fault tolerant per se. Moreover, as described above the sensing unit will deliver true values about the environmental state. Anyway, the undervolted MCU can effect calculation errors or bit flips. Within a redundant network topology those failures can be detected and fixed. To reduce the probability of failures a software redundancy (oversampling) can be used. In addition statistics about how trustful a sensor node runs with undervolting can help to decide which is the best strategy in such cases.

The overhead for implementing undervolting on a sensor node is manageable. The potential energy savings grant additional hardware components like the SI. Moreover, the computational requirements of the SI are less and its usage is periodic with low duty cycles.

V. Effects on Wireless Communication

Undervolting works for an MCU, but what if we include the transceiver unit into the same voltage path? The ability to communicate with other nodes is very important for every wireless sensor node. Thus, if undervolting would have a negative impact on this, the consequence could be a collapse of the WSN structure due to undervolting. However, temporary appearing and disappearing nodes can be handled through tolerant routing protocols like [23]. A real problem occurs when a node ends up isolated from the rest of the network. The isolated node would not be able to decide whether undervolting is responsible for its inability to communicate or just no messages are transceived and the transceiver unit is still fine. Two simple but inefficient approaches would avoid this issue:

- The transceiver unit is connected to a separated voltage path V_{rtx}. The voltage level of V_{rtx} is set to the requested value of the transceiver.
- The voltage level is temporary set to the recommended level of the transceiver unit.

A more efficient but daring solution would be:

- The transceiver unit is potentially undervolted too and the appearance of failures indicates the edge of functionality (cf. section IV).

With regard to the last alternative and if the sensor node uses, as it is common, an ARQ mechanism for reliable data transfer such as provided by the IEEE 802.15.4 standard for wireless communication, the detection of failures when transmitting data is already covered. Then a sensor node can take an increasing number of retransmissions as a sign that undervolting effects the transmission of data. Thus, the node can react itself and adapt the voltage level to higher values. Contrariwise, when a node receives no data it can not decide whether other nodes do not send any data or undervolting prohibits the reception. Hence, a WSN heartbeat could be sent periodically to all nodes. In scenarios without moving nodes, the absence of a heartbeat indicates an error. Based on information such as provided by the heartbeats or the ARQ mechanisms, a heuristic can be derived which decides if the voltage level has to be increased or if the actual failures are caused by other reasons.

To determine the behavior of the transceiver unit if we use undervolting, we measured the probability that a communication error occurs as a function of the voltage level. We used the AT86RF231 [24], a fully integrated transceiver unit which has a minimum recommended voltage level of $V_{rtx} = 1.8V$. For these measurements, the node uses Contiki OS [17] with the RIME communication stack. To detect failures during the communication, the undervolted sensor node stays in idle listening until it receives a request message from a second node. This message is answered subsequently. In practice both, the reception and the transmission of data break down at once. The result is shown in figure 8.

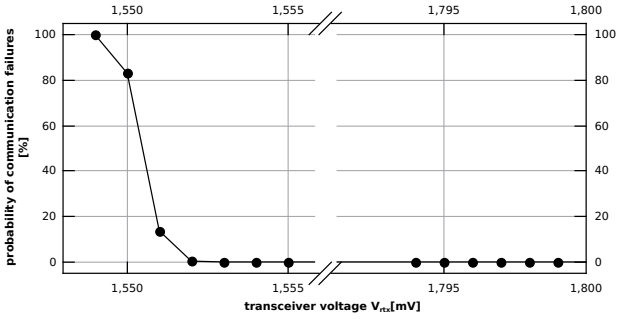


Figure 8. Undervolting induced communication errors.

It is well-known that the usage of the transceiver is one of the most power consuming process on a sensor node. Therefore, the influence of undervolting on the transceiver's current consumption is of particular interest. As the transceiver has a fixed transmit and reception power, the current consumption is not bound to the voltage level. The datasheet [24] underlines this trend of a weak relation between V_{rtx} and the current consumption. Nevertheless, a deviation from the recommended voltage of e.g. $\delta_v = 11\%$ leads to a saving of $\approx 7\%$.

On the other hand, undervolting can effect the link quality of the transceiver unit. According to [24] the transmission power decreases with lower voltage levels. Yet, during our measurements we do not observe an abnormal behavior of received signal strength indicator (RSSI) and link quality indicator (LQI) (distance $\approx 1m$) due to undervolting.

Overall we can conclude that, in general, it is possible to drive also the transceiver unit with undervolting. However, so far we have a limited set of experiments with that only.

VI. Conclusion

As far as we know, this paper presents the first study analysing and demonstrating the potential of undervolting for small devices like wireless sensor nodes. In this preliminary study we have shown that undervolting can save up to 42% of the energy per clock cycle compared to the minimum recommended voltage level of the used MCU. Hence, the lifetime of nodes and networks enhances significantly. With regard to the various application area of WSNs we considered the influence of different temperatures on the concept of undervolting as well. However, the presented results show that undervolting is a new chance to increase the energy efficiency of WSNs and optimize existing concepts on every layer. The already existing fault tolerance of WSNs offers the ideal basis for this power management technique. In conclusion, the lifetime of real-world WSNs and applications could be improved heavily. The individual applicability for undervolting depends on different factors, thus, no generic static solution exists, but dynamic and adaptable algorithms are possible. As future work, we plan to integrate a self-learning algorithm that is capable of predicting future system states by, e.g., taking the trend of environmental temperatures into account. Additionally, a large scale test is planned.

References

[1] W. Tuming, Y. Sijia, and W. Hailong, "A dynamic voltage scaling algorithm for wireless sensor networks," in *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, vol. 1, aug. 2010, pp. V1–554 –V1–557.

[2] T. Hamachiyo, Y. Yokota, and E. Okubo, "A cooperative power-saving technique using dvs and dms based on load prediction in sensor networks," in *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on*, july 2010, pp. 7 –12.

[3] L. B. Hoermann, P. M. Glatz, C. Steger, and R. Weiss, "Energy efficient supply of wsn nodes using component-aware dynamic voltage scaling," *Wireless Conference 2011 - Sustainable Wireless Technologies (European Wireless), 11th European*, pp. 1 –8, april 2011.

[4] U. Kulau, F. Büsching, and L. Wolf, "A node's life: Increasing wsn lifetime by dynamic voltage scaling," in *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*, 2013, pp. 241–248.

[5] R. J. Baker, *CMOS: Circuit Design, Layout, and Simulation*, 3rd ed. IEEE Press, 2010.

[6] J. Suomela, "Survey of local algorithms," *ACM Comput. Surv.*, vol. 45, no. 2, pp. 24:1–24:40, Mar. 2013. [Online]. Available: http://doi.acm.org/10.1145/2431211.2431223

[7] W. Lee, A. Datta, and R. Cardell-Oliver, "Fleximac: A flexible tdma-based mac protocol for fault-tolerant and energy-efficient wireless sensor networks," in *Networks, 2006. ICON '06. 14th IEEE International Conference on*, vol. 2, 2006, pp. 1–6.

[8] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[9] P. Bouvry, H. González-Vélez, and J. Kolodziej, *Intelligent Decision Systems in Large-Scale Distributed Environments*, ser. Studies in Computational Intelligence. Springer, 2011, vol. 362.

[10] M. Floyd, M. Allen-Ware, K. Rajamani, B. Brock, C. Lefurgy, A. Drake, L. Pesantez, T. Gloekler, J. Tierno, P. Bose, and A. Buyuktosunoglu, "Introducing the adaptive energy management features of the power7 chip," *Micro, IEEE*, vol. 31, no. 2, pp. 60–75, March-April.

[11] G. Anthes, "Inexact design: beyond fault-tolerance," *Commun. ACM*, vol. 56, no. 4, pp. 18–20, Apr. 2013. [Online]. Available: http://doi.acm.org/10.1145/2436256.2436262

[12] J. Ben-Othman, K. Bessaoud, A. Bui, and L. Pilard, "Self-stabilizing algorithm for energy saving in wireless sensor networks," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, 2011, pp. 68–73.

[13] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin, "Permasense: investigating permafrost with a wsn in the swiss alps," in *Proceedings of the 4th workshop on Embedded networked sensors*, ser. EmNets '07. New York, NY, USA: ACM, 2007, pp. 8–12. [Online]. Available: http://doi.acm.org/10.1145/1278972.1278974

[14] Atmel, "Atmega164a/164pa/324a/324pa/644a/ 644pa/1284/1284p," 2010. [Online]. Available: http://www.atmel.com/dyn/resources/prod_documents/doc8272.pdf

[15] F. Büsching, U. Kulau, and L. Wolf, "Architecture and Evaluation of INGA - An Inexpensive Node for General Applications," in *Sensors, 2012 IEEE*. Taipei, Taiwan: IEEE, oct. 2012, pp. 842–845.

[16] R. Mangharam, A. Rowe, and R. Rajkumar, "Firefly: a cross-layer platform for real-time embedded wireless networks," *Real-Time Systems*, vol. 37, pp. 183–231, 2007, 10.1007/s11241-007-9028-z. [Online]. Available: http://dx.doi.org/10.1007/s11241-007-9028-z

[17] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, nov. 2004, pp. 455 – 462.

[18] A. Rohani and H.-R. Zarandi, "An analysis of fault effects and propagations in avr microcontroller atmega103(l)," in *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, 2009, pp. 166–172.

[19] M. Doering, S. Rottmann, and L. Wolf, "Demonstration of a snowpack monitoring system based on inexpensive sensor nodes and solar-powered backhaul links," in *Proceedings of the 4th Extreme Conference of Communication (ExtremeCom 2012), Zurich, Switzerland*, 3 2012.

[20] F. Reynolds, "Thermally accelerated aging of semiconductor components," *Proceedings of the IEEE*, vol. 62, no. 2, pp. 212–222, Feb.1974.

[21] Texas Instruments, "Msp430f15x, msp430f16x, msp430f161x mixed signal microcontroller," 2002. [Online]. Available: http://www.ti.com/lit/ds/symlink/msp430f1611.pdf

[22] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*. Piscataway, NJ, USA: IEEE Press, 2005, p. 48.

[23] W.-B. Pöttner, O. Wellnitz, and L. Wolf, "Qos-aodv6e: An energy-balancing qos routing scheme for wsns," in *Proceedings of the Seventh International Conference on Networked Sensing Systems (INSS2010)*, Kassel, Germany, June 2010. [Online]. Available: http://www.ibr.cs.tu-bs.de/papers/poettner-inss2010-qosaodv6e.pdf

[24] Atmel, "At86rf231, low power 2.4 ghz transceiver for zigbee, ieee 802.15.4, 6lowpan, rf4ce, sp100, wirelesshart and ism applications," 2009. [Online]. Available: http://www.atmel.com/dyn/resources/prod_documents/doc8111.pdf