On-board Energy Consumption Assessment for Symbolic Execution Models on Embedded Devices

Antonio Bordonaro[¶], Salvatore Gaglio[*], Giuseppe Lo Re[†], Gloria Martorella[‡], Daniele Peri[§]
Department of Engineering, Viale delle Scienze, Ed. 6, Palermo, Italy
Email: [¶]antonio.bordonaro@unipa.it, [*]salvatore.gaglio@unipa.it, [†]giuseppe.lore@unipa.it,
[‡]gloria.martorella@unipa.it, [§]daniele.peri@unipa.it

***Abstract*—Internet of Things (IoT) applications operate in several domains while requiring seamless integration among heterogeneous objects. Regardless of the specific platform and context, IoT applications demand high energy efficiency. Adopting resource-constrained embedded devices for IoT applications means ensuring low power consumption, low maintenance costs and possibly longer battery life. Meeting these requirements is particularly arduous as programmers are not able to monitor the energy consumption of their own software during development or when applications are finally deployed. In this paper, we discuss on-board real-time energy evaluation of both hardware and software during the development phases and prospect the inclusion of energy-aware capabilities into symbolic execution models for resource-constrained devices, which have not been widely explored before. To provide baseline estimations, tests were carried out on different hardware and software configurations.**

I. Introduction

One of the main critical issues of Internet of Things (IoT) [1]–[3] applications concerns powering deployed devices [4], [5]. IoT applications are often required to run on low power, especially if energy is provided by batteries [6], [7]. Often nodes are deployed and supposed to perform their work for a long time without requiring human intervention so to reduce maintenance costs. Therefore, it is important to measure, analyze and improve the energy efficiency of these devices in order to fulfill the QoS requirements of the application [8], [9]. This is typically achieved through and initial analysis aimed at identifying all the factors affecting energy consumption and, successively, directly addressing, where possible, these factors by implementing appropriate strategies. While analysis and evaluation of energy consumption through appropriate benchmark suites is a common choice [10], [11], comparing experimental benchmark setups makes it possible to evaluate differences and determine the consistence of results with real environments [12]. Because of this, the whole process of developing energy-constrained applications becomes difficult and costly. There is a need for tools that expose the software effect on the energy consumption of a system. Such energy transparency will allow programmers, toolchains, and runtime systems to make energy-aware decisions in order to meet the strict energy constraints of the IoT [4]. For example, many approaches operate at the system level (rather than on the single node) proposing algorithms to reduce power consumption in a Wireless Sensors Area Network (WSAN) [13] or defining new communication protocols that are more efficient and can be used in application scenarios that require distributed and interconnected nodes [14]. The evaluation of complex systems requires the configuration of numerous hardware and software parameters (processor, operating system, drivers, peripherals, etc.), considering a large design space. The resulting complexity restricts the use of board-oriented approaches [15]. Compared to these, transistors or gate-level simulators are typically slower and require more memory but facilitate the analysis of various architectural and software options, enabling to identify performance energy or bottlenecks of the whole system [15]. Authors of [16] propose obtaining energy values directly from the analysis of the source-code without requiring simulation or even compilation. A further higher-level approach is proposed in [17], in which the source-code is converted in a Colored Petri net model, which is used to estimate the energy cost of a given application. The authors of [18] propose techniques to evaluate the energy consumption of software written for a particular platform, including the estimation of consumption in the development process. In another work [19] a shunt resistor is embedded into a USB cable to measure current and bus voltage of the target using an external ADC. All of these approaches are expensive and their size prevents their integration with into the devices of a testbed. Other techniques use symbolic execution to operate at the code layer, providing the programmer with tools that allow to determine the energy requirements of the application code [20] to support the energy aware application development in the initial phases of coding. There are also resource-efficient software development environments that support high-level languages such as C or Python [21]. In [8] the authors present a simple, economical, and accurate measurement platform consisting of a INA219 measurement module and a baseboard executing the measuring code to evaluate the impact of some design parameters on the energy consumption of the target system.

High-level programming and symbolic reasoning [22] has been demonstrated as viable even on resource-constrained devices [23] and also able to model specifications of hardware subsystems that turn into executable on-board verification code [24].

With the aim to extend the symbolic programming environment DC4CD [23] with energy-aware capabilities, in this paper we describe the evaluation of energy consumption of different hardware and software configurations, and different embedded programming approaches. Specifically, we eval-

uated the impact of the symbolic environment running on a resource-constrained platform on the energy consumption of the whole system. The paper is organized as follows. Section II presents the approach. The experimental evaluations are described in Section III. Finally, conclusions and future work are reported in Section IV.

II. Approach

Extending the symbolic programming environment for resource-constrained devices with energy-aware capabilities requires adding on-board energy metering to the target (Fig. 1). To estimate the impact that a symbolic programming environment has on the energy consumption of a resource-constrained embedded devices, we designed the experimental setup to allow for evaluations of the impact of both compiled and interpreted programs on energy consumption of the node.

Compiled programs for embedded devices are usually cross-compiled and the resulting binary image flashed into the storage of the microcontroller (MCU) or of the system on a chip (SOC). An interpreted program consist instead of symbols that are executed by an interpreter running on the target MCU or SOC. In this case, the time-consuming cross-compilation and flashing of the binary image procedures are performed only once while the application code can be sent to the node and updated whenever it is needed.

We also considered two measurement schemes:

- **On-board (ONB)**: the current sensor is powered by the target board, which runs the data acquisition software as well as its own application code (Fig. 1)
- **Off-board (OFB)**: the support board powers the sensor and executes the software for the samples acquisition (Fig. 2);

The sampling rate and the acquisition duration can be configured.

As a target for our experiments we used the Nucleo-STM32F446RE [25]. This board is equipped with the STM32F446RE MCU, which is based on the ARM Cortex-M4 core. The board includes an MCU programmer (ST-Link) that also powers the board from the host USB interface. As a power sensor we used the INA219B, a low-cost current and bus voltage monitoring chip, which supports I2C communication.

MCUs and SOCs contains many peripherals that can be disabled by software to reduce energy consumption when not needed in an application. For our target we defined configurations incrementally: the i+1 configuration is obtained from the i configuration by enabling additional peripherals. Each configuration is identified by the symbol **Hi** (for example, H3 refers to configuration 3). We defined 8 configurations, as reported in Table II. Configuration H0 had the sole ST-Link programmer powered, while CPU-NP had the programmer and the CPU powered and all the peripherals disabled.

The experimental evaluation also considers the software configuration, i.e. the program that the microcontroller runs during the measurement. Each software configuration is identified by the label **Six** (Software Configuration i, language x), in which i is the index associated with the configuration and

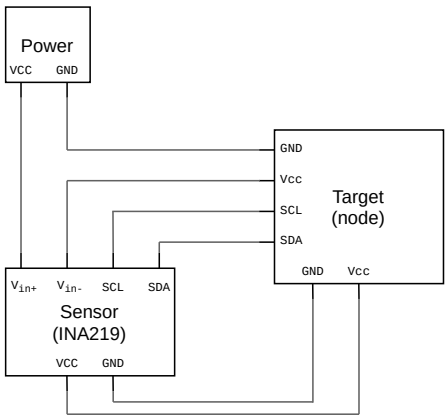

Fig. 1. Setup for on-board measurements

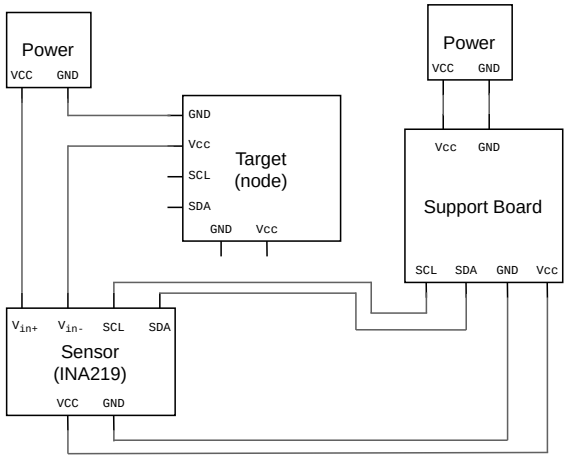

Fig. 2. Setup for off-board measurements

x can be either C or F, to denote the programming language used (C for the C language, F for the Forth language).

We used the following software configurations:

- **S0 - While-true (WHTR)**: consists in a *while(true);* statement and provides a baseline for software evaluation;
- **S1 - Idle-Blink (IBLK)**: blinks the on-board LED at regular intervals of 3 seconds using a busy loop;

Name	ST-Link	CPU-NP	Tim	GPIO	I2C	USART	SPI
H0	✓	✓					
H1	✓						
H2	✓		1,2				
H3	✓		1-5				
H4	✓		1-5	A-H			
H5	✓		1-5	A-H	1-3		
H6	✓		1-5	A-H	1-3	1-6	
H7	✓		1-5	A-H	1-3	1-6	1-3

TABLE I
HARDWARE CONFIGURATIONS OF NUCLEO STM32-F446RE

- **S2 - Timer-Blink (TBLK)**: like the previous benchmark, blinks the on-board LED at 3-second intervals. In this case, however, a hardware timer is used to generate an interrupt every 3 seconds. The interrupt causes the execution of the ISR associated with the timer, which executes the blinking instructions.

III. Experimental Results

We carried out several experiments with combinations of hardware and software configurations in both the on-board and off-board schemes. For each test 1000 readings were collected from the current sensor with period T=700ms. Then the mean (μ) and standard deviation (σ) of the measurements were calculated. In order not to clutter the charts only the first 150 samples were reported.

We defined structured names for the experiments. From the name of an experiment, it is thus possible to retrieve the specific (i) acquisition scheme, (ii), software configuration, (iii) programming language, and (iv) hardware configuration. The **ONB** and **OFB** prefixes respectively denote the on-board from off-board acquisition schemes.

Software configurations are identified by the symbols **S0C**-**S2C** (for C implementations), **S0F**-**S2F**, for Forth implementations. Hardware configurations are identified by the labels **H0**-**H6**.

We carried out the following experiments:

- **H0**
- **OFB-S0C-Hi**
- **OFB-S0F-Hi**
- **OFB-S1x-H1**
- **OFB-S2x-H1**
- **ONB-S1x-H1**
- **ONB-S2x-H1**

The first measurement concerns the energy evaluation of the ST-Link programmer. Estimating correctly the consumption of the programmer is essential, at this stage, because its consumption is not directly related to that of the microcontroller and its peripherals. Considering, initially, the hardware aspects, it can be seen that the measurements performed on hardware configurations characterized by a greater number of enabled peripherals show much higher consumption. For example, Fig. 3 shows the consumption for the same software configuration (S0C) but with different hardware configurations (H1 and H7). The consumptions related to H7 configuration are significantly higher. The *sawtooth* modulation recurs in all the samplings, sign that is a characteristic of the power circuitry and the sensor. Recognizing this pattern could improve the energy-aware abilities of the target. Table II collects the current consumption data of these measurements. Expectedly, excluding the baseline (H0 in Tab II), the H7 configuration has a 14% higher consumption than the H1 configuration. The experimental evaluation also shows how much the software configuration and the type of language used affect energy consumption. The comparison of OFB-S0C-H7 and OFB-S0F-H7 in Fig. 3 shows that the consumption related to the OFB-S0F-H7 measurement (Forth implementation) is higher than

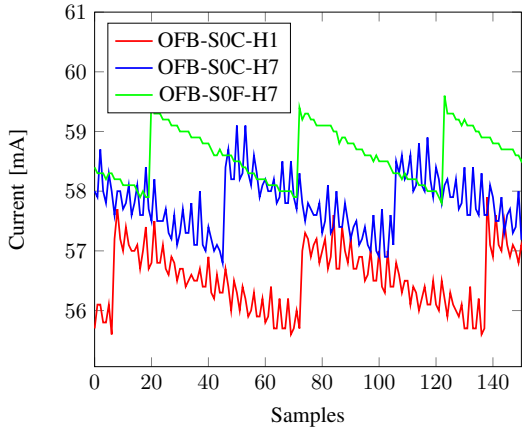

Fig. 3. Comparison between OFB-S0C-H1, OFB-S0C-H7 and OFB-S0F-H7.

the corresponding C implementation. Table II, which contains the energy consumption data of the OFB-S0F-H7 and OFB-S0C-H7 measurements, shows that, excluding the baseline, the interpreter affects energy consumption for the 7.43%.

Figure 4 shows that the on-board acquisition scheme, in addition to being characterized by much higher average consumption, has a standard deviation of the samples higher than in the off-board measurement. Also, the sawtooth pattern of the other measurements is undetectable. In fact, Fig. 3 shows that the OFB-S0F-H7 measurement, although with higher average consumption than the C implementation, shows standard deviation values comparable to all the other off-board measurements.

	Average Current [mA]	**Std. Dev. [mA]**
H0	46.910	0.1909
OFB-S0C-H1	56.451	0.4762
OFB-S0C-H7	57.788	0.5284
OFB-S0F-H7	58.596	0.4381

TABLE II
Comparison of different hardware configuration in C and Forth

IV. Conclusions

In this paper, we described the evaluation of energy consumption in varying hardware and software configurations, and different embedded programming approaches. In addition, with the aim of including energy assessment in the software development process, specifically in the symbolic programming environment DC4CD, we also introduced an on-board measurement scheme. The experimental results show how energy consumption is affected by hardware and software configurations, and by the embedded programming approach. Specifically, the interpreter, although enhancing the functionality of the node, showed a small impact on energy consumption. Future work will integrate the power-aware capability into

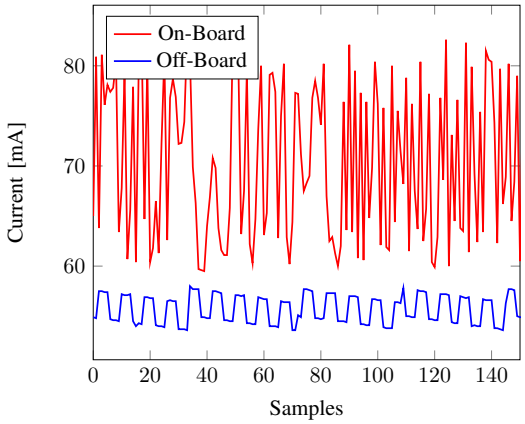

Fig. 4. Comparison of ONB-S1F-H1 and OFB-S1F-H1

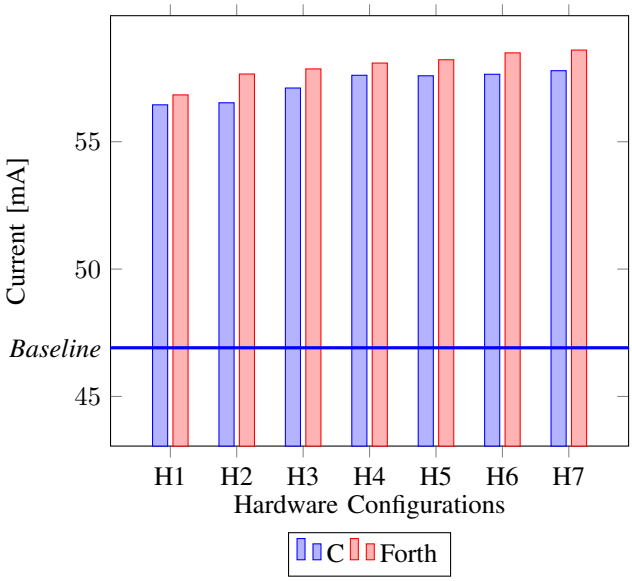

Fig. 5. Energy Consumptions of different Hardware Configurations

DC4CD extend also its hardware targets. To improve the energy evaluations the benchmarking suite will be extended and models of the consumption trends will be included to better characterize the parameters that affect energy consumption.

References

[1] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[2] D. Niyato, X. Lu, P. Wang, D. I. Kim, and Z. Han, "Economics of Internet of Things (IoT): An information market approach," *arXiv preprint arXiv:1510.06837*, 2015.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] K. Georgiou, S. Xavier-de-Souza, and K. Eder, "The IoT Energy Challenge: A Software Perspective," *IEEE Embedded Systems Letters*, vol. 10, no. 3, pp. 53–56, 2018.

[5] A. De Paola, P. Ferraro, G. Lo Re, M. Morana, and M. Ortolani, "A fog-based hybrid intelligent system for energy saving in smart buildings," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 7, pp. 2793–2807, 2020.

[6] N. Miura, Y. Koizumi, Y. Take, H. Matsutani, T. Kuroda, H. Amano, R. Sakamoto, M. Namiki, K. Usami, M. Kondo *et al.*, "A scalable 3D heterogeneous multicore with an inductive ThruChip interface," *IEEE Micro*, vol. 33, no. 6, pp. 6–15, 2013.

[7] D. Peri, "Body area networks and healthcare," *Advances in Intelligent Systems and Computing*, vol. 260, pp. 301–310, 2014.

[8] B. Dezfouli, I. Amirtharaj, and C.-C. C. Li, "EMPIOT: An energy measurement platform for wireless IoT devices," *Journal of Network and Computer Applications*, vol. 121, pp. 135 – 148, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804518302479

[9] P. Cottone, S. Gaglio, G. Lo Re, and M. Ortolani, "User activity recognition for energy saving in smart homes," *Pervasive and Mobile Computing*, vol. 16, no. PA, pp. 156–170, 2015.

[10] R. Bertran, A. Buyuktosunoglu, M. S. Gupta, M. Gonzalez, and P. Bose, "Systematic energy characterization of CMP/SMT processor systems via automated micro-benchmarks," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2012, pp. 199–211.

[11] I. Tsekoura, G. Rebel, P. Glösekötter, and M. Berekovic, "An evaluation of energy efficient microcontrollers," in *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*. IEEE, 2014, pp. 1–5.

[12] C. Haas, J. Wilke, and V. Stöhr, "Realistic simulation of energy consumption in wireless sensor networks," in *European Conference on Wireless Sensor Networks*. Springer, 2012, pp. 82–97.

[13] M. Hodzic and I. Muhic, "Binary algorithm for energy saving in low power wireless sensor networks," in *2016 International Symposium on Industrial Electronics (INDEL)*. IEEE, 2016, pp. 1–6.

[14] T. Huynh, Y. Tan, and K. Tseng, "Energy-aware wireless sensor network with ambient intelligence for smart LED lighting system control," in *IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2011, pp. 2923–2928.

[15] F. Rosa, L. Ost, T. Raupp, F. Moraes, and R. Reis, "Fast Energy Evaluation of Embedded Applications for Many-core Systems," in *2014 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Sep. 2014, pp. 1–6.

[16] J. Castillo, H. Posadas, E. Villar, and M. Martínez, "Energy consumption estimation technique in embedded processors with stable power consumption based on source-code operator energy figures," in *XXII conference on design of circuits and integrated systems*, 2007.

[17] G. Callou, P. Maciel, E. Tavares, E. Andrade, B. Nogueira, C. Araujo, and P. Cunha, "Energy consumption and execution time estimation of embedded system applications," *Microprocessors and Microsystems*, vol. 35, no. 4, pp. 426–440, 2011.

[18] K. Georgiou, S. Kerrison, Z. Chamski, and K. Eder, "Energy Transparency for Deeply Embedded Programs," *ACM Trans. Archit. Code Optim.*, vol. 14, no. 1, pp. 8:1–8:26, Mar. 2017. [Online]. Available: http://doi.acm.org/10.1145/3046679

[19] F. Kaup, P. Gottschling, and D. Hausheer, "PowerPi: Measuring and modeling the power consumption of the Raspberry Pi," in *39th Annual IEEE Conference on Local Computer Networks*, Sep. 2014, pp. 236–243.

[20] T. Hönig, C. Eibel, R. Kapitza, and W. Schröder-Preikschat, "SEEP: exploiting symbolic execution for energy-aware programming," *ACM SIGOPS Operating Systems Review*, vol. 45, no. 3, pp. 58–62, 2012.

[21] A. Mehta, R. Baddour, F. Svensson, H. Gustafsson, and E. Elmroth, "Calvin constrained—A framework for IoT applications in heterogeneous environments," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1063–1073.

[22] S. Gaglio, G. Lo Re, G. Martorella, and D. Peri, "High-level Programming and Symbolic Reasoning on IoT Resource Constrained Devices," *EAI Endorsed Transactions on Cognitive Communications*, vol. 15, no. 2, 5 2015.

[23] ——, "DC4CD: A Platform for Distributed Computing on Constrained Devices," *ACM Transactions on Embedded Computing Systems*, vol. 17, no. 1, Dec. 2017. [Online]. Available: https://doi.org/10.1145/3105923

[24] ——, "WSN Design and Verification Using On-Board Executable Specifications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 710–718, 2019.

[25] STMicroelectronics, "Nucleo64 Board Datasheet," https://www.st.com/resource/en/user_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf.