Design and Evaluation of Single-Board Computer Based Power Monitoring for IoT and Edge Systems

Loic Guegan*, Salma Tofaily*, Issam Raïs*
Department of Computer Science, UiT The Arctic University of Norway, Tromso, Norway*
Corresponding authors: {firstname}.{lastname}@uit.no

***Abstract*—Internet of Things (IoT) and edge platforms are very complex systems. They are heterogeneous in terms of hardware and software. In these systems, being able to document the energy consumed by the nodes is important. To mitigate the impact of such systems on the energy consumption and improve their energy efficiency, research experiments involving power monitoring tools are required.**

However, in the context of IoT and edge platforms, having access to a power monitoring system to monitor specific nodes can be challenging. Similarly, building a power monitoring tool can be time consuming and complex. Single-Board Computers (SBC) based monitoring systems are flexible since they are accessible and provide full integration with existing computer systems, such as test-beds. However, building and evaluating this type of monitoring tool is challenging.

In this paper, we propose a SBC-based power monitoring system that aims at being simple to reproduce and plug-able into existing test-beds. In addition, a framework to evaluate the performance of such systems is detailed. The results show that, with this framework, accurate evaluation of the sample read performance of I^2C-based power meters can be performed. Details about the advantages and weaknesses of our setup are highlighted.

***Index Terms*—power monitoring, performance evaluation, single-board computer, Linux**

I. Introduction

In recent years, the proportion of Internet of Things (IoT) and edge nodes is continuously increasing. By 2025, it is estimated that there will be around 40 billion connected IoT devices [1]. In such context, studying the energy consumption of these types of nodes is crucial. It allows to document their energy consumption, and evaluate the efficiency of energy saving strategies.

There are several approaches to perform such studies. The first one, consists in using simulation. Several simulators from the literature are able to model the power consumed by computer systems [2], [3]. This approach is highly flexible as no physical hardware are involved in the experiments. However, if finer-grained power measurements are required, solely relying on simulation for critical applications could be a problem, as theoretical predictions can mismatch practical measurements.

The second approach, consists in using already established test-beds that provides power monitoring tools allocated to specific computing hardware [4], [5]. This approach is convenient as the user does not need to design, setup and calibrate computing nor monitoring tools. However, it can be challenging to find a test-bed that provides hardware that fits to the user requirements. In specific cases, human intervention must be requested for setting up such hardware [6].

Another approach, is to build a power monitoring system that is able to measure the energy consumed by specific hardware provided by the user. In the context of IoT and edge platforms, where computer systems can be heterogeneous, this approach allows the user to monitor any types of hardware. However, building such monitoring system can be challenging [7] and expensive [8], [9]. It is crucial to evaluate the performance of this power monitoring system to ensure that the collected measurements are accurate.

In this paper, we design and evaluate a Single-Board Computer (SBC) based power monitoring system that is accessible in terms of budget and components, and easy to reproduce. This system combines existing computer hardware to create a portable and flexible power monitoring node, that can be integrated into an IoT and Edge test-bed. The evaluation proposed in this work focuses on the capability of the monitoring node to read samples that are generated at high frequencies by the power meter hardware. The contributions of this paper are:

- An evaluation of an SBC-based power monitoring node that uses I^2C based power meters
- A Linux driver for the INA260 I^2C interface
- A framework to evaluate the performance of SBC-based power monitoring node
- A study of the impact of Pub/Sub-based samples offloading to achieve continuous monitoring for test-beds applications

This paper is organized as follow. Section II presents the state of the art related to this work. The Section III details the proposed hardware and software stack for power monitoring. The Section IV details the experimental protocol used to evaluate our experimental setup. Then, Section V presents the results. Finally, Section VI concludes the work.

II. Related Work

Prototypes measuring power consumption for IoT edge nodes exist [10], [7], [4], [11]. The efficiency of the power monitoring setup depends on the whole system [11] including (i) the monitoring sensor (ii) the observer node hardware and (iii) the power monitoring software stack [11], [7], [10].

Several power monitoring setups have performance bottlenecks, with regards to the maximum sampling rate [4], [11]. Evaluating reading performance at an early stage allows to build solutions supporting sampling rate specifications, requested by IoT applications [10].

A. Precise I^2C compliant power monitoring sensors

There are several methods for direct current sensing such as Current Sense Amplifiers (CSAs) [12], [13]. The CSA method provides better accuracy over other direct current sensing methods. Texas Instruments (TI) is a recognized manufacturer for CSA, with set of devices widely used for energy consumption measurements [7], [4], [14].

The TI INA219 is a heavily used digital-output CSA [7], with a current measurement error of 0.50 %. This error increases when measuring power [15], [16]. However, studies on IoT edge nodes cannot be performed with around 0.50 % measurement error. It is the case in [6], where measured power variability for the idle state on a single node is 0.42 %. In the IoT edge context, high precision and accuracy is important and often crucial [6].

The INA226 has a maximum of 0.10 % shunt voltage gain error, and a maximum of $10\,\mu V$ shunt offset voltage [17]. An external shunt resistor is needed [17]. FIT IoT-LAB [4], a large scale open experimental IoT testbed, uses it to monitor power consumption. The overall accuracy in the setup depends on errors from the CSA and the shunt resistor [16]. The overall accuracy of the power monitoring setup is not reported.

In [14], an INA260 is used to build a power monitoring setup for IoT edge nodes. It is a precise monitoring chip with an I^2C interface and an integrated shunt resistor [18]. At normal temperature, INA260 has a measured maximum system gain error of 0.15 %, and a maximum current offset of $5\,mA$. This chip simplifies power monitoring setup, while providing high precision, high sampling rate, and a wide range of measurements.

A breakout board can host the INA260 and provide a physical interface between the controller node and the measured load (e.g node under test). [19], [20]. In [21], a Printed Circuit Board (PCB) is designed. The VCP Monitor Click [20] is a board based on the INA260, requiring no soldering. But a compatible bus interface at the controller node is needed.

B. Reading performance from observer node

In the FIT IoT-LAB, each observer node reads power from one INA226. The minimum valid period between two consecutive measured samples is $664\,\mu s$, higher than the minimum supported period between two samples by INA226, $140\,\mu s$. The reason for not supporting the minimum sampling rate is not explained [4].

In [22], BeagleBone Black is used to collect measurements from a power monitoring sensor, via the SPI interface. Events that needs to be measured require high sampling frequency, that cannot be theoretically supported by the BeagleBone Black SPI interface. Therefore, SPI protocol is implemented on the Programmable Real-Time Unit (PRU). The evaluation shows that the required sampling frequency can be achieved. The work does not investigate the use of multiple sensors on the serial interface.

In [11], a power monitoring unit is developed to measure power of IoT nodes. The setup includes an Arduino Uno and an INA226, configured for the shortest supported conversion time, $140\,\mu s$. The time between two consecutive sample averages is $4.35\,ms$. The maximum speed of the I^2C interface on the observer node has been identified as a bottleneck for reading performance. Thus, identifying the reading performance capabilities is important to achieve the required sampling rate.

In [7], a power monitoring setup, EMPIOT, is developed using the INA219. Two different observer nodes and I^2C drivers are explored. Two methods to write power monitoring data to file are evaluated: batched and continuous. They are implemented using two buffering mechanisms. Sampling rate is evaluated against I^2C bus speed. It is shown that different I^2C drivers can have different overheads, affecting the sampling rate. The most efficient buffering mechanism for two nodes (Raspberry Pi 3, Raspberry Pi Zero) differs. They conclude that the power monitoring performance depends on hardware architecture, software stack and the number of available cores. The performance when offloading monitoring data to another node, are not investigated, nor using one observer node with multiple sensors.

There is no driver for INA260 in the Linux kernel. In [23], an open source framework is developed, to monitor the power of two GPUs. The setup includes an array of four INA260 on an Adafruit, and a Raspberry Pi. A C library is developed for the Raspberry Pi to read the power measurements via I^2C. This library is based on SMBus™ and measurements are stored on a SD card. The shared setup allows a sampling rate up to 5000 samples per second, for each of the two GPU. The impact of the Raspberry Pi reading performance when varying the number of INA260 is not investigated.

In [24], a power measurement system based on the INA260 is presented. The system consists of modules, each with four INA260, connected to an I^2C bus that

can be shared across up to four modules. The analysis of allowed sampling rate is based on theoretical computations that consider only INA260 specifications. However, a power monitoring system contains other components and factors that can affect reading performance (e.g., I^2C speed of the observer node, software used to collect measurements). In [7], results show that the actual time needed to get a sample from INA219 is longer than the data-sheet reported values. Experiments are needed to evaluate the actual supported sampling rates in a power measurement system, with multiple senors.

C. Summary

Power monitoring systems use observer nodes and connected power sensors. Evaluations of power monitoring setups show bottlenecks in several works.

By sharing the communication buses, more than one power sensor can be attached to an observer node. Several existing works estimate reading performance of multiple sensors by computation, based on system architecture and data-sheet values. However, performance evaluations using data-sheets values can be inaccurate.

To the best of our knowledge, there is no driver for the Linux kernel supporting the INA260. In theory, the BeagleBone Black I^2C speed is fast enough to support the INA260 maximum sampling rate when reading power samples from a single power sensor [18]. But, no related work measures experimentally the reading performance of an observer node from several power sensors.

III. Experimental Setup

This section details the SBC based power monitoring hardware used in the work. The software approach used to interact with this hardware is also presented.

A. Hardware description

This work proposes to use a SBC-based power monitoring node. It consists of a BeagleBone Black board that act as the observer node. It is equipped with up to four INA260 chips.

The INA260 is a power monitoring sensor with a precision integrated shunt. It allows to monitor current, voltage and power of a given load from $0\,\mathrm{V}$ to $36\,\mathrm{V}$ and up to $15\,\mathrm{A}$. The INA260 provides I^2C and SMBus™ compatible interfaces. It is compatible with Fast mode ($1\,\mathrm{kHz}$ to $400\,\mathrm{kHz}$) and High-Speed mode ($1\,\mathrm{kHz}$ to $2.94\,\mathrm{MHz}$). This work focuses on the I^2C interface as it provides greater communication performance compared to SMBus™. The INA260 is entirely controlled via the I^2C interface. Its eight internal registers are exposed, and read/write operations can be performed to configure the chip and collect samples (current, voltage or power).

To configure and collect samples from the INA260, our setup uses a BeagleBone Black SBC. This board is equipped with an I^2C bus compatible with Standard mode (up to $100\,\mathrm{kHz}$) and Fast mode (up to $400\,\mathrm{kHz}$). For easing the connection between the INA260 chips and the BeagleBone Black, a mikroBUS Cape extension is used. This extension provides a standardized connection to link Click board™ based hardware with the BeagleBone Black. Our setup used four VCP Monitor Click that are equipped with INA260. This entire setup is depicted on Figure 1. It shows the BeagleBone Black (left), the mikroBUS Cape extension (middle) and one VCP Monitor Click (right). Note that, only one VCP Monitor Click is shown. Connections between the different components are represented with red labels. The four VCP Monitor Click boards can be connected to slots 1, 2, 3 and 4 of the mikroBUS Cape extension. This extension is attached to the BeagleBone Black. All the mentioned BeagleBone Black pins (GND, 3V3, SCL and SDA) are shared across all mikroBUS Cape slots, when four INA260 are connected to the same I^2C bus and powered with $3.3\,\mathrm{V}$ DC. To prevent collisions, the physical addresses of the INA260 on the I^2C bus must be adjusted. This is achieved, by rewiring the zero-ohms resistors A1 and A0 (c.f Figure 1), according to the INA260 and VCP Monitor Click datasheets.

To fully utilize this hardware setup in an efficient manner, a software stack must be carefully designed. The next section details this software stack.

B. Exposing INA260 registers to the user space

Reading INA260 samples from the BeagleBone Black via I^2C bus must be done efficiently, as it happens with a delay. This delay, noted d, must be minimized since it impacts the precision of the monitoring setup. If d is too high, samples generated by the INA260 are going to be missing (not read by the BeagleBone Black). To minimize d, the software stack must be designed carefully, and must aim at reducing any potential overhead.

The current Linux kernel release (v6.5.2) does not provide support for the INA260. Only specific Texas Instruments power monitoring chips, without integrated shunts, are supported by the kernel via the *ina2xx* driver. To mitigate the delay d that could be introduced by I^2C libraries, and to maximize the control over the entire monitoring pipeline from the INA260 up to the end user, a Linux driver is designed specifically for the INA260 [25]. This driver is compatible with the BeagleBone Black kernel v4.19.94-ti-r42.

This driver exposes all the registers and fields of the INA260 through the sysfs Virtual File System. Read and write operations on the INA260 registers are performed through the exposed files from the *registers* directory of the given INA260. Similarly, read and write operations can be performed on specific register fields. The driver provides access to the last samples (current, voltage and power) generated by the INA260. To collect these samples, the user can perform continuous read from the current (in ampere), voltage (in volts) or power (in watts) files, located at the root of the given INA260.

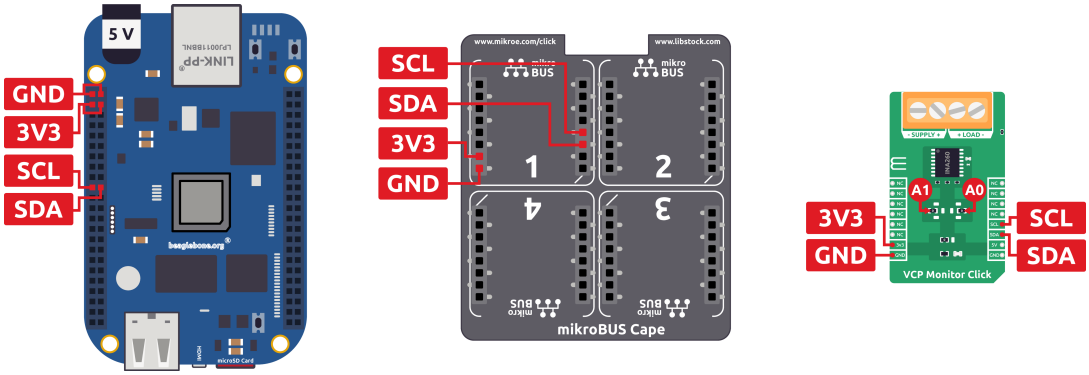


Fig. 1: Power monitoring setup with 1 BeagleBone Black (left), 1 mikroBUS Cape (middle), 1 VCP Monitor Click (right). All mentioned pins (GND, 3V3, SCL and SDA) are shared across all mikroBUS Cape slots (1, 2, 3 and 4).

With this approach, INA260 chips can be configured, and measurements can be collected using regular system calls. No additional libraries are required, and the overhead (delay d) is reduced and under control. A second version of the driver is also provided in [25]. This other version uses the Linux Hardware Monitoring subsystem to provide standardize access to sensors devices. However, this paper uses the previously presented driver, providing better support for INA260 (e.g: fields accesses etc.). For more recent Linux kernel releases, an up to date version of these two drivers are provided in [25].

IV. Experimental Protocol

A. Sample read performance evaluation

To quantify the sample read performance achievable with the experimental setup, a first set of experiments are designed. A program, written in C, continuously read power samples from INA260 using the previously presented driver. It stops when a certain amount of samples are collected. In this work, when a sample is collected, it is always associated with a timestamp. Two storing approaches are investigated.

in-memory: keeps the samples in the Random Access Memory (RAM) for the whole experiment duration. It ensures low overhead during the experiment since no additional processing are required. At the end of the experiment, samples are all stored in a file on the BeagleBone Black SD card. This approach has one disadvantage: a mechanism must ensure that enough RAM is available for the whole experiment duration.

in-file: stores each sample read in a file. It is more robust in case of unforeseen events such as power shortage. However, it generates additional I/O that can impact the monitoring performance and increase the delay d during the experiment. Since samples are immediately stored after reading, the RAM is not the limiting factor. Instead, the amount of non-volatile storage on the SD card, becomes the bottleneck. To perform continuous sample read, other mechanisms based must be studied.

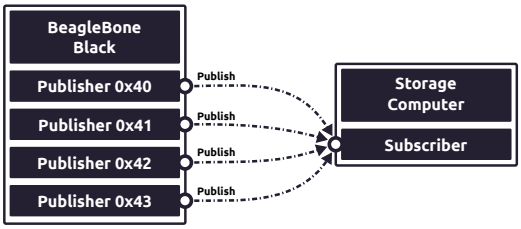


Fig. 2: ZeroMQ based Pub/Sub experiments using one BeagleBone Black and one external node, for storage.

B. Pub/Sub-based samples offloading evaluation

Continuously monitoring the power consumption of devices is important. However, reading power samples continuously at high frequency generates a lot of data. It can quickly saturate the local storage. One solution to this problem is to offload the samples (during the experiment) on another node with higher storage capabilities. This offloading process takes additional computational and I/O resources.

To evaluate the feasibility of this approach with our setup, we designed a Pub/Sub based application written in C, using ZeroMQ, detailed Figure 2. It shows four publisher processes, located on the BeagleBone Black. These processes collect continuously power samples from the INA260. They send samples periodically to a subscriber process, located on a storage computer. These steps are repeated until the end of the experiment.

C. Experimental parameters

To run experiments, the four INA260 must be configured. The most important parameters of the INA260 are located in the configuration register (pointer address 0xFF). This register provides the following fields:

- **AVG:** Averaging mode (number of measurements used to generate a sample)

- **VBUSCT:** Bus voltage conversion time (time spent generating a bus voltage sample)
- **ISHCT:** Shunt current conversion time (time spent generating a current sample)

These parameters impact the INA260 samples accuracy. They also influence the frequency at which samples are generated by the INA260. The INA260 datasheet does not explicitly define how these parameters contribute to its sampling frequency. Thanks to the theoretical values provided by the datasheet, we deduced that INA260 sampling frequency f is computed as:

$$f = \frac{1}{N_{\text{AVG}} \times (T_{\text{VBUSCT}} + T_{\text{ISHCT}})} \quad (1)$$

Where N_{AVG} corresponds to the number of collected measurements that are averaged, T_{VBUSCT} the bus voltage conversion time and T_{ISHCT} the shunt current conversion time. The value of these three parameters depends on the experimental context. For accurate measurements, the value of these three parameters must be maximized. This work uses the default INA260 configuration with $N_{\text{AVG}} = 1, T_{\text{VBUSCT}} = 1.1\,\text{ms}, T_{\text{ISHCT}} = 1.1\,\text{ms}$. Thus, we define $f_{\text{default}} = 454.54\,\text{Hz}$. The configuration used in our experiments for four INA260 are detailed in Table I. According to the INA260 datasheet and Equation 1, the highest achievable sampling frequency with the INA260 is $f_{\text{max}} = 3571.429\,\text{Hz}$ (with $N_{\text{AVG}} = 1, T_{\text{VBUSCT}} = T_{\text{ISHCT}} = 140\,\mu\text{s}$).

For maximum throughput between the INA260 chips and the BeagleBone Black, their respective I^2C clock frequencies must be adjusted to the highest common supported frequency. As detailed in Section III, it corresponds to the I^2C Fast mode frequency of $400\,\text{kHz}$. This parameter is specified in Table I. By default, the BeagleBone Black is programmed to use the Standard mode. To allow the Linux kernel I^2C subsystem to use the Fast mode frequency, the BeagleBone Black Device Tree specification must be updated and recompiled.

Regarding the monitored hardware, four different loads are used (see Table I). Aside from the resistor, we choose to monitor hardware that fit in IoT and edge applications and withing the INA260 load limitations.

The parameters used for sample read performance evaluation are detailed in Table I. These experiments evaluate the reading performance using up to four INA260 concurrently, and store the samples either in-memory (RAM) or in a file. The experiment runs until each INA260 collects $500\,000$ samples.

Parameters used for the Pub/Sub experiment are detailed in Table I. Up to four INA260 are used concurrently for a duration of $300\,\text{s}$. Every minute, each publisher offloads its collected samples to the subscriber. One sample queue is used per publisher, thus, no measurements can be collected during the offloading process.

D. Metrics

To measure the performance of our setup, three metrics are introduced. The *sample read frequency*, f_r in Hz, is the number of samples read that happens each second. This metric is computed from the file containing all collected samples, generated by each experiment. The *average delay* $\overline{d}$ in seconds measures the average delay over one second between consecutive reads. It permits to measure the stability of samples readings, highlighting potential missed values.

The *percentage of samples that are off delay* for a given experiment, according to a threshold is noted $p_{\text{od}}(x)$. Taking more than x seconds categorises a sample as *off delay*. This metric is computed as follows: Given the samples collected from an experiment, lets define the delay of a sample i as $d_i = t_i - t_{i-1}$ with t_i the time at which the sample i is read. Lets define the set of delays $D = \{d_i\}$ and $D' = \{d_i | d_i > x\}$. Then, $p_{\text{od}}(x) = \frac{\mathbf{card}(D')}{\mathbf{card}(D)} \times 100$.

Using an INA260 with a sampling frequency f, the maximum tolerable delay between sample reads is $x = \frac{1}{f}$. The lower $p_{\text{od}}(\frac{1}{f})$ is, the lower the amount of missing samples is going to be.

V. Evaluation Results

This section analyses the results of our experimental protocol. All these experiments are reproducible, and available online [26].

A. Sample read frequency f_r

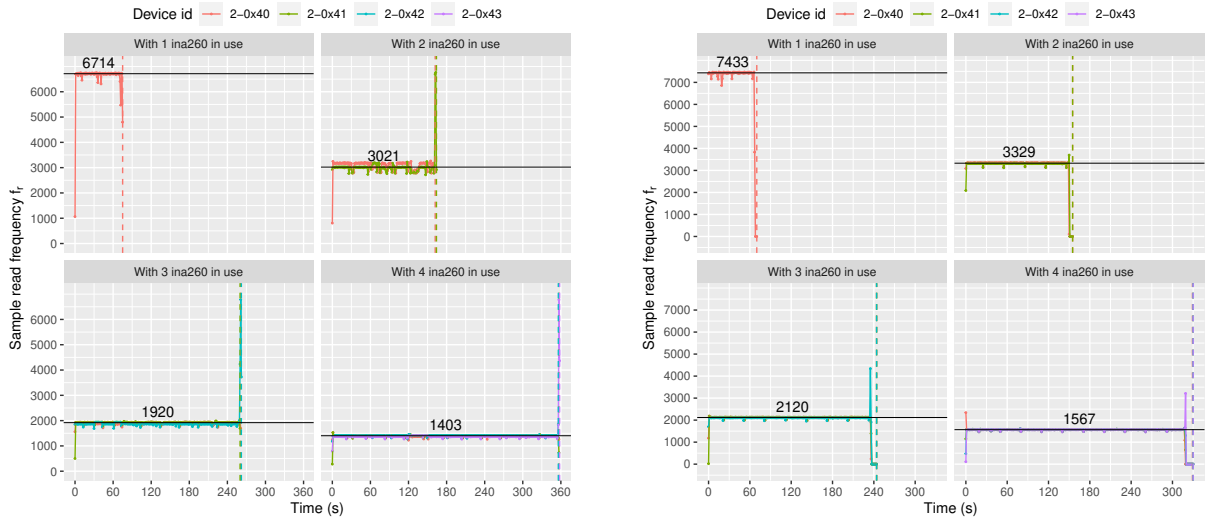
Figure 3 presents the results of the sample read performance evaluation. Figure 3a and Figure 3b correspond to *in-file* and *in-memory* experiments, respectively.

Figure 3a depicts the evolution of the sample read frequency f_r through time when 1 to 4 INA260 are used concurrently. Each color corresponds to f_r for a given INA260. Dash lines indicates the end of the experiment for a given INA260. The horizontal black line encompasses the median sample read frequency f_r for all INA260 of a given experiment.

in-memory experiments results show that, with one INA260, our setup is able to reach a median f_r of $6714\,\text{Hz}$. For the whole duration of the experiment with one INA260, we have $f_r > f_{\text{max}} > f_{\text{default}}$. Aside from the startup, and end phase, small drops in f_r can be observed during the experiment. However, these drops never exceed $600\,\text{Hz}$. Consequently, with one INA260, we are not loosing any power measurements. However, when increasing the number of concurrent INA260, the median f_r is significantly reduced. However, this decrease is not linear. It can be due to a combination of mechanisms involved in software and hardware to handles concurrent accesses to the I^2C bus. In addition, the sample read frequency f_r is also unstable. Still, in all cases $f_r > f_{\text{default}}$. However, with two or more

TABLE I: Experimental parameters.

Hardware/Experiment	Parameter	Value
INA260	Configuration register	0x6127 *(default value)*
	Sampling frequency	454.54 Hz ($f_{default}$)
	I^2C physical addresses	{0x40, 0x41, 0x42, 0x43}
BeagleBone Black	I^2C Clock frequency	400 kHz, Fast mode
Monitored load	Address 0x40	1 kΩ resistor, 5.0 V DC
	Address 0x41	Raspberry Pi 3 model B (2016), 5.1 V DC
	Address 0x42	Raspberry Pi 3 model B+ (2017), 5.1 V DC
	Address 0x43	Raspberry Pi 4 model B (2018), 5.1 V DC
Sample read experiments	Concurrent INA260 used	1, 2, 3 and 4
	Sample storage modes	*in-memory* or *in-file*
	Collected samples per run	500 000
Pub/Sub experiments	#INA260 used concurrently	1, 2, 3 and 4
	Experiment duration	300 s
	Samples publishing interval	60 s
	Sample queues per publisher	1



(a) *in-file* experiments

(b) *in-memory* experiments

Fig. 3: Sample read performance evaluation. Time series of the sample read frequency f_r.

INA260, $f_r < f_{\max}$. In this case, our setup will lead to a significant amount of missed samples, when the INA260 sampling frequency uses $f_{\max}$.

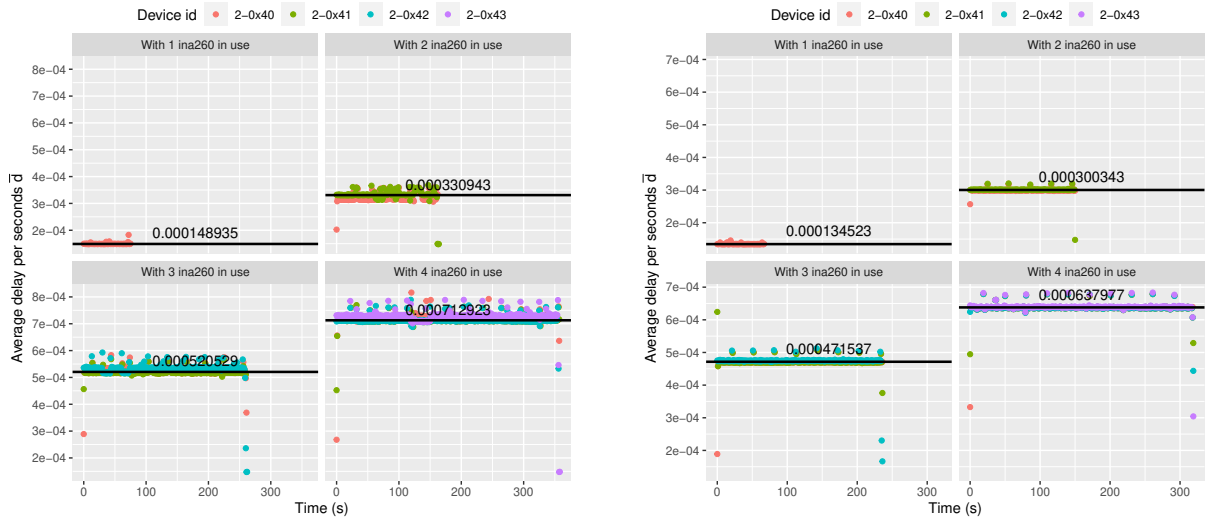
Figure 3b shows the results for the *in-memory* experiments. When using with one INA260, the median f_r increases to 7433 Hz (+719). In addition, compared to the *in-file* results, f_r is more stable through time. Since no additional I/O and computations are required to keep samples in memory, the sample read process has stable performance. Still, with two or more INA260 in use, $f_r < f_{\max}$. Storing the samples on the SD card makes $f_r = 0$, at the end of the experiment. The more INA260 are used, the longer this writing period lasts.

To summarize, both approaches (*in-file* or *in-memory*) have their pros and cons. Keeping samples in memory provides more stable and slightly higher f_r, but a writing period is required to save the measurements. In addition, enough RAM is needed on the BeagleBone Black for the whole experiment duration. Writing samples to files provides better reliability, but generates more I/O that lead to unstable and slightly lower f_r.

B. Sample read delay

Figure 4 presents the average delay $\overline{d}$ for the sample read performance evaluation of the *in-file* and *in-memory* experiments.

Figure 4a presents, for the *in-file* scenario, the average delay $\overline{d}$ between consecutive sample reads for each second. These results are shown when 1 to 4 INA260 are used concurrently. Each color corresponds to $\overline{d}$ for a given INA260. The dash lines indicates the end of the experiment for a given INA260. The horizontal black line encompasses the median $\overline{d}$ for all INA260 of a



(a) *in-file* experiments

(b) *in-memory* experiments

Fig. 4: Sample read performance evaluation. Time series of average delay $\overline{d}$.

given experiment. Overall, the average delay between sample read $\overline{d}$ is stable using one INA260. Increasing the number of concurrent INA260 in use, introduces more variability in the average delay $\overline{d}$. But overall, the average delay $\overline{d}$ reflects the associated f_r. As an example, with one INA260, $f_r \approx \frac{1}{\overline{d}} \approx 6714.338\,\text{Hz}$.

Table II shows the percentage of missing samples ($p_{\text{od}}(x)$ metric) for each experiments and for both INA260 sampling frequency setup f_{default} and f_{max}. For the *in-file* experiments, using INA260 configured with f_{default}, the percentage of off delay samples is lower than 0.08%. Up to four INA260 can be used concurrently. However, if the INA260 are configured to use f_{max} sampling frequency, only one INA260 can be used, due to more than 95% samples being off delay.

Figure 4b shows the average delay $\overline{d}$ metric for the *in-memory* experiments. Keeping samples in memory provides more stability on the average delay $\overline{d}$, since less I/O and computations are required. Results from Table II highlight this stability. Using INA260 configured with the f_{default}, p_{od} is lower than 0.03% as $\overline{d}$ is getting closer to its real value. Using INA260 configured with f_{max} increases p_{od}, compared to *in-file*. A slightly higher amount of short delays, when samples are written to file, are measured. The majority of these delays are located in the first $100\,\text{s}$ and seems related the management of I/O operations by the operating system. Overall, *in-memory* provides lower, and more stable delay $\overline{d}$.

To summarize, this power monitoring setup allows for high sample read frequency (more than $6000\,\text{Hz}$ at best). However, the p_{od} metric reveals that, even with high f_r values, and an average delay $\overline{d}$ that is stable and reflect f_r, a significant amount of samples can be off delay. This leads to a high amount of missing samples. Hence,

TABLE II: $p_{\text{od}}(x)$, for each experiment

	in-file		*in-memory*	
#INA260	$p_{\text{od}}(\frac{1}{f_{\text{default}}})$	$p_{\text{od}}(\frac{1}{f_{\text{max}}})$	$p_{\text{od}}(\frac{1}{f_{\text{default}}})$	$p_{\text{od}}(\frac{1}{f_{\text{max}}})$
1	0.01%	1.31%	0.00%	0.07%
2	0.01%	95.62%	0.00%	99.04%
3	0.04%	99.06%	0.02%	99.60%
4	0.07%	99.39%	0.03%	99.85%

up to four INA260 configured with a sampling frequency of f_{default} can be used with our setup. However, only a single one configured with f_{max} can be used at a time.

C. Pub/Sub

Figure 5 details f_r for the performance evaluation of the Pub/Sub-based offloading approach, generated using the power samples and associated timestamp, collected on the subscriber node. When using one INA260, the median f_r is comparable to *in-memory* (Figure 4b). With more than one INA260, the median f_r is even lower compared to *in-file* (Figure 4a). This is due to the additional resources used by the publisher processes. A significant drop in f_r is visible every $60\,\text{s}$, corresponding to the sample publishing interval (Table I).

To mitigate the impact of publishing on f_r, two approaches can be used. Increasing the samples publishing interval or using more than one sample queue per publisher. However, additional experiments need to be made to evaluate potential drawbacks. The impact of the Pub/Sub publisher processes on $\overline{d}$ and p_{od} must also be investigated. These results show that our setup is able to offload samples during experiments, while maintaining a high sample read frequency median f_r on the non-publishing phases.

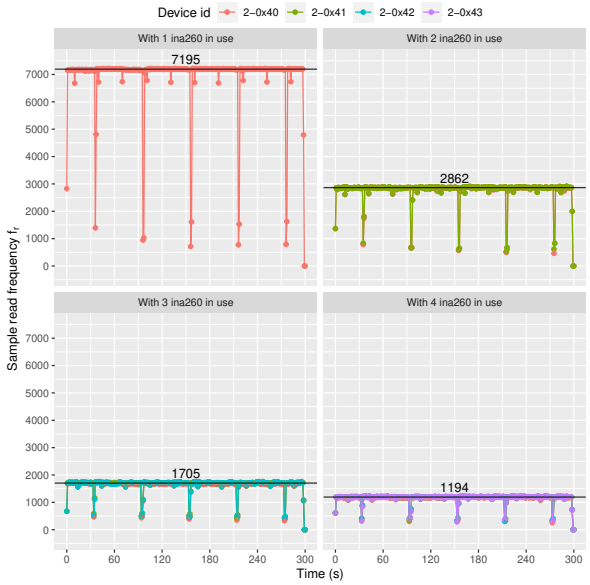


Fig. 5: Pub/Sub-based offloading performance evaluation. Time series of the sample read frequency f_r.

VI. Conclusion

A design and evaluation of an SBC-based power monitoring system that is accessible and entirely reproducible has been proposed. This setup uses a BeagleBone Black combined with a mikroBUS Cape and four VCP Monitor Click that communicates with the I^2C interface. This choices for hardware setup are precisely detailed and the proposed drivers are provided.

An evaluation framework to study the performance of the sample read frequency of a power monitoring device from a computer system is proposed. The evaluation results of our setup shows that a single INA260 can be used with its maximum sampling frequency. However, if more than one INA260 is used on the same I^2C bus, it leads to significant drop in the sample read frequency. The sampling frequency of the INA260 must be adjusted according to the amount of INA260 used concurrently.

This work provides early results on the impact of samples offloading during the measurement period. The results show a significant drop in the sample read frequency during the publishing phases. Strategies that could mitigate this phenomenon are proposed. We plan to perform in depth studies of the suggested strategies to leverage maximum sample read performance. An evaluation of the integration of such monitoring device into an IoT and edge computing test-bed is also considered.

References

[1] AT&T, "At&t brings cellular connectivity through iot innovations," AT&T, 2021. [Online]. Available: https://about.att.com/story/2021/blues_wireless.html

[2] L. Guegan, I. Raïs, and O. Anshus, "Validation of esds using epidemic-based data dissemination algorithms," in *DCOSS-IoT*, 2023.

[3] I. Raïs, L. Guegan, and O. Anshus, "Impact of loosely coupled data dissemination policies for resource challenged environments," in *CCGrid*, 2022.

[4] C. Adjih *et al.*, "Fit iot-lab: A large scale open experimental iot testbed," in *WF-IoT*. IEEE, 2015.

[5] D. Balouek *et al.*, "Adding virtualization capabilities to the Grid'5000 testbed," in *Cloud Computing and Services Science*, ser. Communications in Computer and Information Science, I. I. Ivanov, M. van Sinderen, F. Leymann, and T. Shan, Eds. Springer International Publishing, 2013, vol. 367.

[6] S. Tofaily, I. Raïs, and O. Anshus, "Quantifying the variability of power and energy consumption for iot edge nodes," in *DCOSS-IoT*, 2023.

[7] 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, 2018.

[8] P. Tian, X. Ma, C. A. Boano, Y. Liu, F. Yang, X. Tian, D. Li, and J. Wei, "Chirpbox: An infrastructure-less lora testbed." in *EWSN*, 2021.

[9] K. D. digital multimeter. [Online]. Available: https://no.farnell.com/keithley/dmm7510/dmm-bench-7-5-digit-10a-1kv/dp/2770044

[10] R. Lim *et al.*, "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *IPSN*. New York, USA: Association for Computing Machinery, 2013.

[11] M. Hoss, F. Westmeier, and J.-P. Akelbein, "Low cost high resolution ampere meter for automated power tests for constrained devices." in *CERC*, 2019, pp. 81–88.

[12] T. Instruments, "Simplifying current sensing." [Online]. Available: https://www.ti.com/lit/eb/slyy154a/slyy154a.pdf

[13] ——, "Current sense amplifiers." [Online]. Available: https://www.ti.com/lit/sg/slyb194e/slyb194e.pdf

[14] F. Dar *et al.*, "Upscaling fog computing in oceans for underwater pervasive data science using low-cost micro-clouds," vol. 4, no. 2, mar 2023.

[15] T. Instruments, "Ina219 zero-drift, bidirectional current/power monitor with i2c interface." [Online]. Available: https://www.ti.com/lit/ds/symlink/ina219.pdf

[16] ——, "Low temperature drift integrated shunt vs active temperature compensation of shunt." [Online]. Available: https://www.ti.com/lit/an/slya032/slya032.pdf

[17] ——, "Ina226 high-side or low-side measurement, bi-directional current and power monitor with i2c compatible interface." [Online]. Available: https://www.ti.com/lit/ds/symlink/ina226.pdf

[18] ——, "Ina260 precision digital current and power monitor with low-drift, precision integrated shunt." [Online]. Available: https://www.ti.com/lit/ds/symlink/ina260.pdf

[19] Adafruit, "Adafruit ina260 high or low side voltage, current, power sensor." [Online]. Available: https://www.adafruit.com/product/4226

[20] MIKROE, "Vcp monitor click." [Online]. Available: https://www.mikroe.com/vcp-monitor-click

[21] F. Megías Teruel, "Design of an ammeter for the test of printed circuits," B.S. thesis, Universitat Politècnica de Catalunya, 2019.

[22] G. Kazdaridis *et al.*, "Everun: Enabling power consumption monitoring in underwater networking platforms," in *Proceedings of the 11th Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*, 2017.

[23] M. Karimi, Y. Wang, and H. Kim, "An open-source power monitoring framework for real-time energy-aware gpu scheduling research," in *Open Demo Session of IEEE Real-Time Systems Symposium (RTSS@ Work)*, 2022.

[24] S. Köhler *et al.*, "Pinpoint the joules: Unifying runtime-support for energy measurements on heterogeneous systems," in *ROSS*, 2020.

[25] L. Guégan, "INA260 Linux driver source code." [Online]. Available: https://gitlab.com/manzerbredes/ina260-sysfs-driver

[26] ——, "BeagleBone Black experiments source code." [Online]. Available: https://gitlab.com/manzerbredes/ina260-beaglebone-performance