Delft University of Technology
Master of Science Thesis in Embedded Systems

# Testing Energy Harvesting Devices Realistic and Repeatable

**Boris Theunis Blokland**


TU Delft Delft University of Technology

# Testing Energy Harvesting Devices Realistic and Repeatable

Master of Science Thesis in Embedded Systems

Embedded and Networked Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Boris Theunis Blokland
b.t.blokland@student.tudelft.nl
borisblokland@gmail.com

15th November 2019

**Author**
Boris Theunis Blokland (b.t.blokland@student.tudelft.nl)
(borisblokland@gmail.com)

**Title**
Testing Energy Harvesting Devices Realistic and Repeatable

**MSc Presentation Date**
29th November 2019

**Graduation Committee**

| | |
|---|---|
| Dr.ir. Fernando Kuipers | Delft University of Technology |
| Dr. Przemysław Pawełczak | Delft University of Technology |
| Dr. Stefanie Roos | Delft University of Technology |

**Abstract**

The Internet-of-Things is a promising vision which enables trillions of sensor devices to be connected. A common bottleneck for such devices is the energy supply. Using batteries is expensive, it reduces the device life time and has a negative impact on the environment. Energy harvesting (EH) provides a more sustainable solution. Unfortunately, environmental energy may not always be available, causing devices to be intermittently-powered. Due to the random and volatile nature of environmental energy, it is hard to conduct repeatable tests for such devices.

Therefore, this thesis proposes a generic EH emulation testbed which can emulate a complete EH power supply chain in realistic and repeatable fashion. It builds on previous work [De Mil, EURASIP JWCN (2010)], increasing the accuracy by orders of magnitude allowing emulation comparable with actual execution. The emulation is evaluated for a RF harvesting and solar energy harvesting setup, operating under various input power and load conditions. Both setups show a error margin of less than 10% under most testing conditions. The key features of the design are: highly generic setup, software configurable storage capacitor, native energy aware debugging and simulation support. The emulation model is integrated in a state-of-the-art testbed called Shepherd [Geissdorfer, SenSys (2019)] to benefit form its features and make this work available for the scientific community.

# Preface

This thesis has been conducted in the Embedded and Networked Systems group at the Delft University of Technology. One year ago I started looking for a thesis topic until I arrived at the office of Przemysław Pawełczak. He introduced me to the topic of batteryless sensors, suggested several possible thesis topics and concluded that the scientific community lacked a testbed for experimenting with such batteryless devices. In my previous study of Electronics at the Rotterdam University of Applied Sciences I worked on several testbed setups. Therefore this topic interested me the most and thus began the final chapter of my Master.

I would like to thank various people who supported me during my work. In the first place Przemysław Pawełczak who enthusiastically supervised me throughout my work in the past year. I would also like to thank Kai Geissdoerfer from TU Dresden for his willingness to cooperate. Then I would like to thank Jasper de Winkel for many long and fruitful discussions which helped me in forming my thesis. Finally I thank my girlfriend Lisa and my parents for always supporting me in what I do!

Boris Theunis Blokland

Delft, The Netherlands
15th November 2019

# Contents

**Preface** **v**

**1 Introduction** **1**
1.1 Motivation . . . 1
1.2 Research Goal . . . 2
1.3 Contributions . . . 2
1.4 Thesis outline . . . 3

**2 Related Work** **5**
2.1 Background . . . 5
2.1.1 Energy Harvesting components . . . 5
2.1.2 Energy Harvesting Architecture . . . 6
2.2 Existing Wireless Sensor Networks . . . 6
2.2.1 Flocklab . . . 6
2.2.2 FIT IoT-Lab . . . 6
2.2.3 Indriya2 . . . 7
2.3 Existing Intermittently-Powered Platforms . . . 7
2.4 Existing Energy Harvesting Emulators . . . 7
2.4.1 Ekho . . . 9
2.4.2 Shepherd . . . 9
2.4.3 Environment Emulator . . . 9
2.5 Comparison Emulators . . . 10

**3 Design** **11**
3.1 Capacitor Voltage Model . . . 11
3.2 Output Capacitor Compensation . . . 12

**4 Implementation** **15**
4.1 Emulation Testbed Implementation . . . 15
4.1.1 Hardware . . . 15
4.1.2 Software . . . 16
4.2 Emulation Input Power . . . 17
4.3 Modelling Converter Efficiency . . . 18
4.3.1 RF Harvesting Efficiency Modelling . . . 18
4.3.2 Solar Energy Harvesting Efficiency Modelling . . . 18
4.4 Shepherd Integration . . . 20
4.4.1 Conversion To Non-Floating-Point . . . 20
4.4.2 Reducing Divisions . . . 21

4.4.3 Integer Square Root . . . . . 21

**5 Evaluation** **23**
5.1 Experimental Setup . . . . . 23
5.1.1 Evaluation Metric . . . . . 23
5.1.2 Hardware Setup . . . . . 23
5.1.3 Software Setup . . . . . 25
5.2 RF Harvesting . . . . . 26
5.3 Solar Energy Harvesting . . . . . 26
5.4 Repeatability . . . . . 27

**6 Future Work** **31**
6.1 Dynamic Capacitance . . . . . 31
6.2 Energy-Aware Debugging . . . . . 31
6.3 Simulation/Emulation Synergy . . . . . 31

**7 Conclusions** **33**

**A Initial Testbed Setup and Interface** **35**
A.1 Our Proposal . . . . . 35
A.2 Shepherd . . . . . 36
A.3 Comparison and Further Direction . . . . . 36

# Chapter 1

# Introduction

## 1.1 Motivation

The Internet-of-Things (IoT) is a promising vision which enables trillions of sensor devices to be connected [24]. A common bottleneck for such devices is the energy supply. Batteries are large, expensive, heavy and wear out after several years. This requires batteries to be recharged or replaced which can be very expensive, especially when sensor devices are placed in hard to reach places [52].

A sustainable solution is to replace batteries with energy harvesters. Herewith a device collects its energy from the environment. Possible energy sources are: solar, radio frequency (RF), thermal or kinetic energy. Unfortunately, environmental energy can be scarce and varying over time. The power provided by the energy harvester can be smaller than the required power for the sensor device to operate.

This lack of power causes erratic execution. A common use-case for sensor devices is to do a measurement and send this measurement to a base station wirelessly. If there is enough environmental energy available, the device will start up and do its measurement. Then it starts sending a packet. However, transmission fails because sending a radio packet is power hungry and the environmental energy is not sufficient. As a result the input power collapses and the device shuts down. When the input voltage is restored the device will start up and try again, but will never be able to send a full packet.

To mitigate this issue, energy is stored in a storage capacitor. At some point the capacitor has harvested enough energy for the device to perform a single task. The capacitor has reached the upper threshold voltage. Now the output is turned on and the capacitor voltage is applied to the target device. The device will run until the capacitor reaches its lower threshold voltage, after which the output is shut down and the capacitor charges until it reaches its upper threshold voltage again. This mechanism of turning on and off is shown in Figure 1.1.

The frequent power failing causes new challenges in developing software for such devices. It contrasts with the standard assumption that programs run continuously throughout execution. The programmer has to take care of this intermittent behavior by for instance storing data to non-volatile memory at certain intervals. It is difficult to predict how long a program can execute

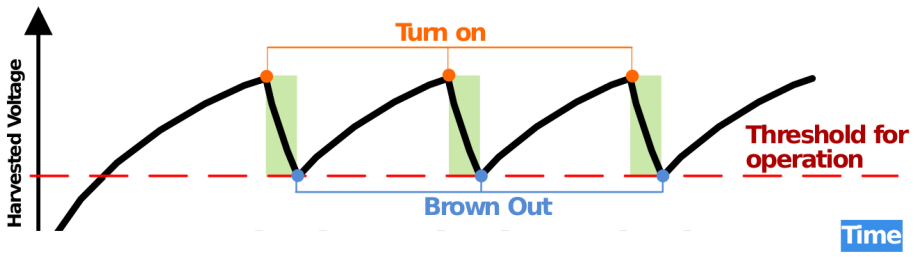



Figure 1.1: **Cycle of a energy harvesting capacitor voltage. First it charges until is has enough energy harvested, then it outputs its voltage to a target device discharging until it reaches brown-out voltage [7].**

before the next power failure. Therefore extensive testing is required to validate program execution.

However, it is hard to conduct repeatable tests due to the random nature of the energy source. While comparing two algorithms, it is impossible to conclude that one algorithm outperforms the other without knowing how much the difference in available energy contributed to the result.

One way to tackle this is to re-create energy harvesting (EH) conditions, do measurements over a long period of time and average the result. This is a tedious process and can be inaccurate, which leads us to our research goal.

## 1.2 Research Goal

Our research goal is to accelerate development for batteryless intermittently-powered devices, by building a testbed which can repeatably and realistically emulate energy harvesting conditions.

## 1.3 Contributions

The main contributions of this thesis are:

- A survey on existing intermittently-powered EH platforms.
- Improvements on an existing EH model to emulate the behavior of a complete EH power supply chain.
- A computer simulation which implements our model, written in both Matlab and C/C++.
- Hardware platform which allows us to not only simulate, but also emulate our model.
- Accuracy evaluation of our model by emulating two EH setups: RF harvester by the Powerharvester [32] and solar energy harvesting with a bq25570 [49].
- Integration of our model in state-of-the-art testbed Shepherd, to make this work available to the scientific community.

## 1.4 Thesis outline

The outline of the thesis is as follows: Chapter 2 provides a summary of related work. Chapter 3 describes the emulation model. Chapter 4 shows the implementation of the model in a custom develop hardware and the integration in a state-of-the-art testbed. Chapter 5 shows the evaluation of our model with real hardware. Finally Chapter 6 discusses future work and Chapter 7 presents the conclusion.

# Chapter 2

# Related Work

In this chapter we will look at related work to place our thesis in context. In Section 2.1 we present background information required to understand the related work. Section 2.2 describes wireless sensor network (WSN) testbeds. In Section 2.3 we will survey existing energy harvesting platforms to investigate what a EH WSN device node could look like. A discussion of various emulation solutions is stated in Section 2.4, concluded with a comparsion in Section 2.5.

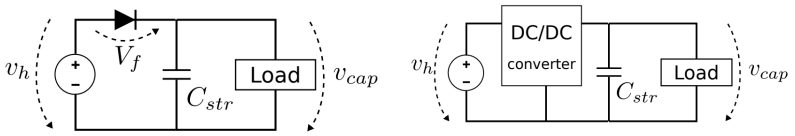
## 2.1 Background

For this thesis it is important to know the components of an energy harvesting architecture. We will refer to this in the discussion of related work and later on in the thesis.

### 2.1.1 Energy Harvesting components

A typical energy harvesting setup consists of several components:

- **Energy Source**: Environmental energy source, i.e. solar, RF or kinetic energy.
- **Harvester**: This device converts the environmental energy into electricity, i.e. a solar panel or RF-to-DC-converter.
- **Capacitor/Storage Device**: Different types of devices can be used to store the harvested energy, i.e. regular capacitor, super-capacitor or a rechargeable battery.
- **Converter (optional)**: Boosts the voltage of the harvester output to get in operation range of the target load.
- **Output Switch**: A switch controlling the output, based on the capacitor voltage. Without such control the capacitor would be instantly depleted when energy is available.




(a) Converter-less (b) Converter-based

Figure 2.1: **Two kind of energy harvesting architectures [21]. (a) directly outputs the capacitor voltage to the load. This output voltage might not reach the minimum voltage of the load. (b) overcomes this by adding a converter.**

### 2.1.2 Energy Harvesting Architecture

There are the two main energy harvesting architectures: converter-less and converter-based architectures. These are shown in Figure 2.1.

## 2.2 Existing Wireless Sensor Networks

For batteryless, intermittently-powered devices there are no publicly available testbeds. Work of [1] enlists properties and features that such a testbed should have, as well as presenting a minimal implementation. The authors call for a more coordinated action in this domain research.

On the other hand there are dozens of existing testbeds for battery-based WSN. There have been many surveys published in the past years that compare each of them in detail (see for instance [50], [22]). Our goal here is to revise these comparative studies expanding it with the recent developments in testbed deployments. As we show there are currently three active WSN testbeds.

### 2.2.1 Flocklab

Flocklab [26] is developed and run by the Computer Engineering and Networks Laboratory at the Swiss Federal Institute of Technology Zurich. It exists of 27 nodes, distributed over one level of the ETZ-building at ETH Zurich. It support various sensor nodes (MSP430 [42], CC2538 [43], STM32 [41]) and different wireless chips (868 MHz SX1211 [37] and 2.4 GHz CC2420 [44]). The key feature of Flocklab is that it can do time accurate pin tracing and actuation. Besides pin tracing it can emulate battery depletion by controlling the supply voltage.

### 2.2.2 FIT IoT-Lab

FIT IoT-LAB [2] is a large scale infrastructure for WSNs developed by a consortium of five French institutions of higher education and research with 1786 nodes, split over six different sites in France. It supports various sensor nodes (MSP430 [17], ARM Cortex M3 [19] and Cortex-A8 [18]) and different wireless chips (802.15.4 PHY @ 800 MHz or 2.4 GHz [28, 44]). The key feature of the IoT-LAB is that it supports mobile robots.

| Company | Description | Energy Source |
|---|---|---|
| EnOcean [15] | Various batteryless solutions for i.e. Building Automation and Smart Home. | Solar, thermal, kinetic |
| Powercast [31] | Provides wireless power solutions, RFID tags, RF power transmitter, RF power harvester. | RF |
| Williot [53] | Makes a batteryless bluetooth beacon device based on RF harvesting. | RF |
| Everactive [16] | Provides batteryless monitoring solutions to mainly the industry. Related to the university of Virginia and Michigan. | Solar, thermal |
| BelluTech [6] | BelluTechs patented batteryless wireless miniature sensors continuously track and record exposure to environmental and operating conditions. | RF |
| Matrix Industries [27] | Makes a batteryless thermoelectric powered smartwatch. | Solar, thermal |
| Nowi Energy [29] | Company provides a Energy Harvesting Power Management chip. | Solar, RF, thermal |

Table 2.1: **Commercial Energy Harvesting Related Companies.**

### 2.2.3 Indriya2

Indriya2 [4] is a three-dimensional WSN deployed across three floors of the School of Computing, at the National University of Singapore. Currently, Indriya2 can support a mixture of TelosB [11], SensorTag CC2650 [45] and SensorTag CC1350 [46]. Indriya2 is an upgrade of indriya [14], new features are: supporting heterogeneous sensor devices, multiple users to schedule jobs over non-overlapping set of heterogeneous nodes at the same time and a real-time publish/subscribe architecture to send/receive data to/from the testbed nodes.

## 2.3 Existing Intermittently-Powered Platforms

The field of intermittently-powered energy harvesting devices is promising, but still immature and only a few companies provide such devices commercially. Table 2.1 shows several companies which are commercially active in the field of energy harvesting devices. Table 2.2 shows some popular and recent developed intermittently-powered platforms in scientific research.

## 2.4 Existing Energy Harvesting Emulators

In this section three existing energy harvesting emulators are discussed: Ekho, Shepherd and Environment Emulator.

| Platform | Description | MCU | Radio | Energy Harvester | Energy Source | Capacitor Size | Year | Citations |
|---|---|---|---|---|---|---|---|---|
| WISP [36] | Family of sensors that are powered and read by UHF RFID readers | MSP430 | Backscattering | Transducer and rectifiers | RF | 10-50 µF | 2008 | 705 |
| E-WEHP [51] | A Batteryless Embedded Sensor-Platform Wireless Powered From Ambient Digital-TV Signals | PIC24F | NA | Transducer and rectifiers | RF | 100 µF | 2013 | 136 |
| AD PZT Energy Harvester [25] | Self-Powered Wireless Sensor Node Enabled by an Aerosol-Deposited PZT Flexible Energy Harvester | MSP430 | CS2500 | Flexible piezoelectric energy harvester | Kinetic | 1000 µF | 2016 | 86 |
| Umich Moo [54] | Improvement on design of WISP | MSP430 | Backscattering | Transducer and rectifiers | RF | NA | 2011 | 76 |
| Monjolo [13] | Energy-Harvesting AC Power metering which draws zero power under zero load conditions | MSP430 | CC2420 | CR2550, LTC3588 | Electric | 500 µF | 2013 | 48 |
| SPWTS [39] | A novel self-powered wireless temperature sensor based on thermoelectric generators | nRF24L | nRF24L | TEC12706 | Thermal | NA | 2014 | 41 |
| Flicker [24] | Configurable development board for batteryless IoT | MSP430 | Various | Various | Solar, RF, Kinetic | 47 µF | 2017 | 25 |
| Step Counter [35] | Intermittently-Powered Energy Harvesting Step Counter for Fitness Tracking | MSP430 | NA | Ferroelectret insole | Kinetic | 4.7-16 µF | 2017 | 10 |
| Capybara [9] | Co-designed hardware/software power system with dynamically reconfigurable energy storage capacity | MSP430, CC2650 | CC2650 | Solar panels | Solar, energy source emulation | 770 µF-67.5 mF | 2018 | 32 |
| Pible [20] | BLE batteryless platform | CC2650 | CC2650 | Solar panels | Solar | 220 mF-1 F | 2018 | 4 |
| TPC Bike[38] | A Transiently-Powered Wireless Cycle Computer | MSP430 | nRF24L01 | Coil | Magnetic | 50-235 µF | 2017 | 1 |

Table 2.2: **Research based intermittently-powered platforms. Citations are taken from `scholar.google.com` on 28 October 2019.**

### 2.4.1 Ekho

Ekho emulates energy harvesters based on their I-V (current-voltage) curve [23]. This curve describes current drawn from the harvester with respect to the load voltage. By rapidly sweeping the load from a very small, to a very large resistance, this curve can be recorded.

As environmental energy changes over time, the I-V curve changes as well. For example: a solar panel produces less power when a cloud moves before the sun. These changes are captured by creating a trace of I-V curves.

Unfortunately, the software of Ekho has its shortcomings. It uses no hardware timers, which is one of the least requirements to do accurate time based measurements. Moreover, no set of pre-recorded traces is made available to work with. Recording I-V curves generate a lot of data points, restricting the length of a recording to 2–3 minutes on a 4 GB SD card. The Ekho setup would still require a fixed capacitor and has no converter. Therefore it could only support converter-less architectures.

### 2.4.2 Shepherd

Shepherd[1] [21] is portable testbed for batteryless IoT. It can both record and emulate EH conditions. The testbed is based on TI's bq25504 boost converter [47]. During recording, an energy harvester is attached to the converter (i.e. a solar panel), and the input current and voltage are measured. During emulation, a variable current source outputs the pre-recorded input current at the input of the converter. By exploiting the Maximum Power Point Tracking (MPPT) of the bq25504, the input voltage can be fixed to a reference voltage. This reference is set to the pre-recorded input voltage. Now, both input current and voltage are the same as during recording, therefore successfully emulating a pre-recorded trace.

The downside of using this method, is that it is focuses on a single converter. This excludes emulating converter-less applications, or applications that require other converters. Also the bq25504 is at end-of-life, forcing the testbed to use the substitute bq25505 [48]. This new converter does not allow the user to disable MPPT. Therefore every 16 seconds, the converter will shut down for 256 ms, causing inaccuracy to the emulation. Another downside of this long 16 second MPPT period is that it does not respond to quick changes in input energy, which might not be viable for intermittent applications. The testbed uses a fixed size capacitor. If one wants to perform a test on different capacitor values, it is required to replace these capacitors by hand every time.

### 2.4.3 Environment Emulator

The work of [12] presents an environment emulator (EE). This emulates a capacitor based on its current-voltage relation

$$V(t) = \frac{1}{C} \int_{t_0}^{t} I(\tau) d\tau + V(t_0),$$

[1] Please note that at the start of this thesis, Shepherd was not yet published. When we were confronted with Shepherd, we decided to re-focus the thesis. Our initial setup is discussed in Appendix A.

| | Low latency | Converter-less | Converter-based | Realistic input | Dynamic capacitor |
|---|---|---|---|---|---|
| **Ekho** | ✗ | ✓ | ✗ | ✓ | ✗ |
| **Shepherd** | ✓ | ✗ | ✓ | ✓ | ✗ |
| **EE** | ✗ | ✓ | ✗ | ✗ | ✓ |
| ***This work*** | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 2.3: **Comparison with existing energy harvesting emulators. *Low latency* indicates that the emulation responds fast enough to a changing load; *Converter-less* and *converter-based* refer to the supported architectures; *Realistic input* tells whether the input can reflect real harvesting conditions; *Dynamic capacitance* is whether the system storage capacitance can be changed without replacing a fixed capacitor by hand.**

where $V(t)$ is the capacitor voltage, $C$ is the capacitance and $I(\tau)$ is the difference in current flowing in the capacitor. The emulator uses a fixed input current to emulate an energy harvester. By measuring the output current, the difference in current flowing in the capacitor can be determined and thus the capacitor voltage can be calculated. A DAC is used to output this virtual capacitor voltage to a target device.

Here the environmental energy is emulated by a fixed input current. This is a very simplistic model and does not take the changing of environment into account. Also this is a converter-less setup, in which converter-based applications cannot run. The capacitor voltage is updated every 12.5 ms. Especially for emulating smaller capacitors, this is too slow, since the voltage can change rapidly while transmitting radio packets.

However, this work will form the basis of the solution proposed in this thesis.

## 2.5 Comparison Emulators

Table 2.3 shows the existing energy harvesting emulators compared against the the work presented in this thesis. The comparison is based on important characteristics of emulation which determine the accuracy and flexibility. It is shown that related work leaves room for improvement and supports the relevance of this thesis' work.

# Chapter 3

# Design

In this chapter we describe the model of our energy harvesting emulation. In our model we emulate the behavior of the aforementioned storage capacitor and converter. The basics of the model are based on the work of [12]. The model is extended by introducing a converter and adding other improvements.

## 3.1 Capacitor Voltage Model

We start from the current-voltage relation of a capacitor

$$V_{\text{cap}}(t) = \frac{1}{C}\int_{t_0}^{t} I(\tau)d\tau + V_{\text{cap}}(t_0), \tag{3.1}$$

where $V_{\text{cap}}(t)$ is the capacitor voltage, $C$ is the capacitance, $I(\tau)$ is the sum of in- and outgoing current over a period of $d\tau$ and $V_{\text{cap}}(t_0)$ is the initial capacitor voltage at $t = 0$.

By taking the derivative of (3.1), we get

$$\Delta V_{\text{cap}} = \frac{\Delta I \Delta t}{C}. \tag{3.2}$$

We now define $V_{\text{cap}}(n)$ as a discrete function, implementing the integral of (3.1) as

$$\begin{cases} V_{\text{cap}}(0) = V_{\text{lt}}, \\ V_{\text{cap}}(n) = V_{\text{cap}}(n-1) + \Delta V_{\text{cap}}, \end{cases} \tag{3.3}$$

where $n$ is an integer indicating the iteration of the function and $V_{\text{lt}}$ the lower threshold capacitor voltage at which the output turns off.

We define $\Delta I$ as

$$\Delta I = I_{\text{cin}} - I_{\text{cout}} - I_{\text{leakage}}, \tag{3.4}$$

where $I_{\text{cin}}$ is input current charging the capacitor, $I_{\text{cout}}$ is current flowing out of the capacitor based on the load and $I_{\text{leakage}}$ is the static leakage current of the capacitor. $I_{\text{cin}}$ is derived as

$$I_{\text{cin}} = I_{\text{in}} \frac{V_{\text{in}}}{V_{\text{cap}}} \eta_{\text{in}}(I_{\text{in}}, V_{\text{in}}), \tag{3.5}$$

where $V_{\text{cap}}$ is the voltage on the capacitor and $\eta_{\text{in}}(I_{\text{in}}, V_{\text{in}})$ is the input efficiency of the converter as function of input current $I_{\text{in}}$ and input voltage $V_{\text{in}}$. Note that a converter can be composed of two stages. The first stage converts the input voltage to the capacitor voltage. The second stage converts the capacitor voltage to the desired output voltage. $\eta_{\text{in}}$ and $\eta_{\text{out}}$ define the efficiency of the first and second stage. Some converters only have the first stage and directly output the capacitor voltage to the load.

$I_{\text{cout}}$ is defined as

$$I_{\text{cout}} = I_{\text{out}} \frac{V_{\text{out}}}{V_{\text{cap}} \eta_{\text{out}}(I_{\text{out}}, V_{\text{out}})}, \tag{3.6}$$

where $I_{\text{out}}$ is the measured output current flowing into the load, $V_{\text{out}}$ is the measured output voltage applied to the load and $\eta_{\text{out}}(I_{\text{out}}, V_{\text{out}})$ is the output efficiency as function of output current $I_{\text{out}}$ and output voltage $V_{\text{out}}$. $V_{\text{out}}$ gets determined by

$$V_{\text{out}} = V_{on} b(n, V_{\text{cap}}), \tag{3.7}$$

where $V_{on}$ is the voltage when the output is on, $b(n, V_{cap})$ is a boolean determining the output state as function of the capacitor voltage defined as

$$\begin{cases} b(0)(V_{\text{cap}}) = \text{false}, \\ b(n)(V_{\text{cap}}) = \begin{cases} \text{true}, & \text{if not}(b(n-1)) \text{ and } (V_{\text{cap}} > V_{\text{ut}}), \\ \text{false}, & \text{if } b(n-1) \text{ and } (V_{\text{cap}}, < V_{\text{lt}}), \\ b_{n-1,} & \text{otherwise}, \end{cases} \end{cases} \tag{3.8}$$

where $V_{\text{ut}}$ is the upper threshold capacitor voltage and $V_{\text{lt}}$ the lower threshold capacitor voltage at which, respectively, the output turns on and off.

## 3.2 Output Capacitor Compensation

Converters can have a small output capacitor. When the output turns on, the bigger storage capacitor instantly charges the output capacitor. This causes the storage capacitor voltage to drop as shown in Figure 3.1. To increase the accuracy of our emulation we model this voltage drop by calculating $V_{\text{new}}$.

As the output turns on, energy will transfer between the capacitors, defined as

$$E_{\text{new}} = E_{\text{old}} - E_{\text{output}}, \tag{3.9}$$

where $E_{\text{new}}$ and $E_{\text{old}}$ is the energy level in the storage capacitor before and after the output turns on respectively; $E_{\text{output}}$ is the energy stored in the output capacitor. We are interested in the capacitor voltage. The relation between capacitor voltage and energy is defined as

$$E = \frac{CV^2}{2}, \tag{3.10}$$

where $E$ is the energy in the capacitor, $C$ is the capacitance and $V$ the capacitor voltage. We combine (3.10) and (3.9)

$$\frac{C_{\text{storage}} V_{\text{new}}^2}{2} = \frac{C_{\text{storage}} V_{\text{old}}^2}{2} - \frac{C_{\text{output}} V_{\text{new}}^2}{2}. \tag{3.11}$$

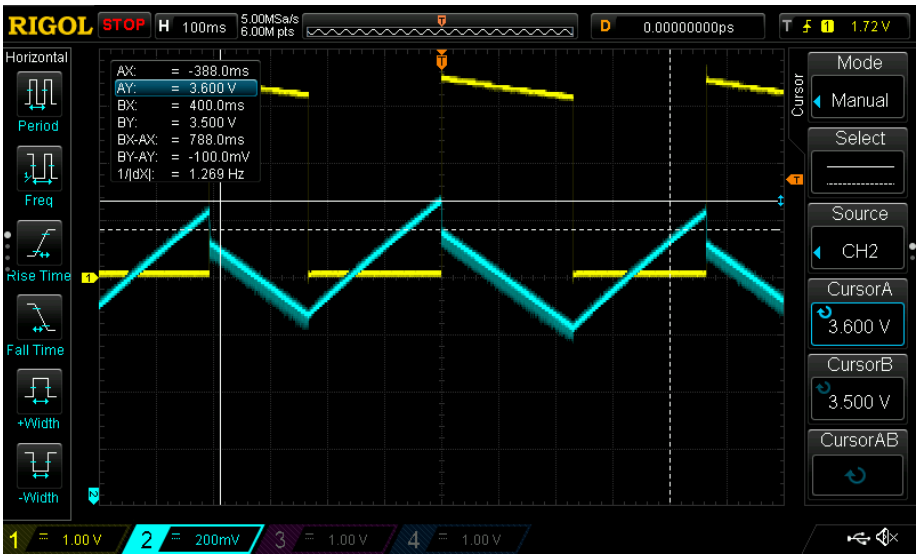



Figure 3.1: **Scope image of storage capacitor voltage (in blue) and output voltage (in yellow) from a solar-powered bq25570 converter with a** $94\,\mu\mathrm{F}$ **storage capacitor,** $22\,\mu\mathrm{F}$ **output capacitor and** $1\,\mathrm{k}\Omega$ **load. While the output is off, the capacitor voltage charges until it reaches its upper threshold voltage. When the output voltage turns on, the capacitor voltage drops** $0.1\,\mathrm{V}$**.**

Rewriting (3.11) we get

$$V_{\mathrm{new}} = \sqrt{\frac{C_{\mathrm{storage}}}{C_{\mathrm{storage}} + C_{\mathrm{output}}}} V_{\mathrm{old}}. \tag{3.12}$$

# Chapter 4

# Implementation

In this chapter we define the implementation of the model presented in Chapter 3. In Section 4.1 we describe our own implementation of the model on a custom developed hardware. Section 4.2 describes how input values are chosen for the emulation. In Section 4.3 is explained how the efficiency of the converter is implemented. Finally, Section 4.4 describes the integration of our model into the existing Shepherd platform.

## 4.1 Emulation Testbed Implementation

In this section we describe our custom implementation of the energy harvesting emulation testbed. We first take a look at the used hardware, and then we describe how we implemented the model in software. The model is implemented for emulating two different actual EH setups: RF harvesting using a P2110 Powerharvester [32] and solar energy harvesting using a bq25570 [49].

### 4.1.1 Hardware

The emulator runs on a Teensy 3.5 [30] 120 MHz ARM Cortex-M4 micro controller with Floating Point Unit. It incorporates two ADCs and a DAC for, respectively, current sense, voltage sense and voltage output.

The accuracy of the emulator greatly depends on the sampling period. The current consumption of a target device can change rapidly. Sending a radio packet, creating a peak in power demand, can be as short as hundreds of microseconds. These transients need to be captured by our emulator. Therefore the sampling speed is set to 100 kHz.

We have chosen for a Teensy since it is easy available, can handle measurements at this sampling speed, provides a SD card slot which can be used to store recording traces and an USB port for connection with a PC to output measurements. On top of that Teensy comes with software libraries which ease development. A custom PCB is developed to provide a power supply and an analog front-end for the Teensy, as shown in Figure 4.1.

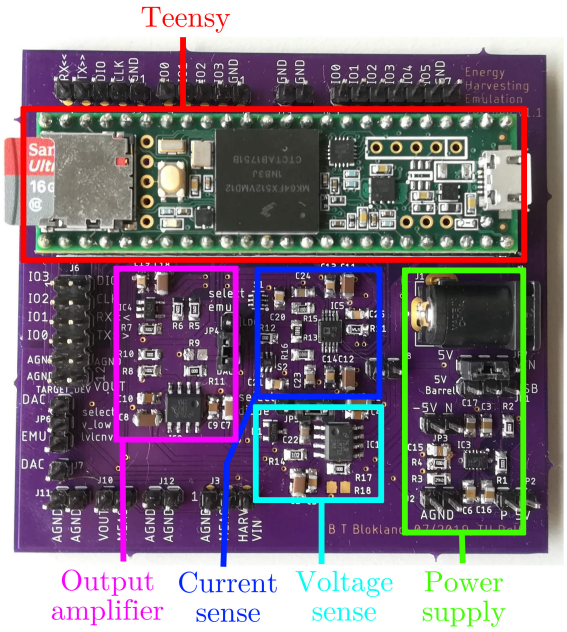



Figure 4.1: **A picture of the custom developed hardware on which the energy harvesting emulator is run. Board dimensions are** $7 \times 8$ cm

.

### 4.1.2 Software

Directly implementing the model as described in Chapter 3 is impossible, since the voltage and current values are not in real units, but represented in 10-bit integer values

$$V' = V\frac{2^{10}-1}{3.3}, \tag{4.1}$$

$$I' = I\frac{2^{10}-1}{0.033}. \tag{4.2}$$

An easy solution would be to convert these logic values to floating point, execute the algorithm, and then convert them back to logic values. This however adds instructions to the algorithm. And we need our software as fast as possible, because of the high sampling speed.

The emulation algorithm runs at 100 kHz, leaving theoretically 10 µs of computation time for a single iteration. Because reading from a SD card also takes time, this is reduced to ~6 µs. The emulator software flow is shown in Figure 4.2.

If the model is converted from real to logic values, run-time conversions are not needed and thus the algorithm runs faster. Combining (3.2), (4.1) and (4.2), it follows that:

$$\Delta V'_{\text{cap}}\frac{3.3}{2^{10}-1} = \frac{\Delta I'\frac{0.033}{2^{10}-1}\Delta t}{C}. \tag{4.3}$$

We can rewrite (4.3):

$$\Delta V'_{\text{cap}} = \frac{\Delta I'\Delta t}{100C}. \tag{4.4}$$

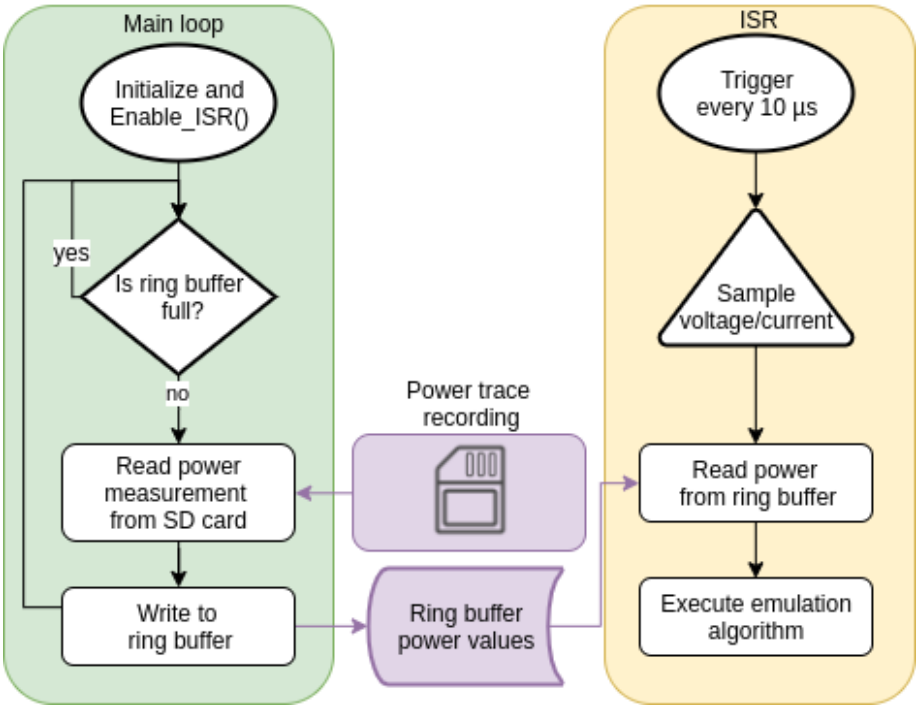



Figure 4.2: **Software flow of the emulator. The main loop continuously fills a ring buffer with power data from a SD card. An Interrupt Service Routine (ISR) runs every 10 µs to sample output voltage and current, and read new power data to update the capacitor voltage. The purple lines show the flow of data.**

We convert (3.4) into using logic values:

$$\Delta I' = I'_{\text{cin}} - I'_{\text{cout}} - I'_{\text{leakage}}. \tag{4.5}$$

Combining (3.5), (4.1) and (4.2), it follows:

$$I'_{\text{cin}} = I'_{\text{in}} \frac{V'_{\text{in}}}{V'_{\text{cap}}} \eta_{\text{in}}. \tag{4.6}$$

Finally we can combine (3.6), (4.1) and (4.2) in a similar way and it follows that:

$$I'_{\text{cout}} = I'_{\text{out}} \frac{V'_{\text{out}}}{V'_{\text{cap}} \eta_{\text{out}}}. \tag{4.7}$$

## 4.2 Emulation Input Power

The emulation model requires an input voltage $V_{\text{in}}$ and input current $I_{\text{in}}$. These input values are not only dependent on the EH source, but are also dependent on the input impedance of the converter. EH Converters like the bq25570 [49] set their input impedance to match the maximum power point of the EH source.

At this the point the input impedance is optimal such that power drawn from the EH source is maximized.

$V_{\text{in}}$ and $I_{\text{in}}$ could in theory be modelled, however, this would be complex because the interaction between the EH source and converter should be modeled as well. And be re-done for every energy source and converter. Another approach is to record these values from an actual converter connected to a EH source. This method has two advantages over the modelling approach:

- **Generic**. Any combination of EH source and converter can be recorded in this way.
- **High accuracy**. Both behavior of EH source and converter are captured, as well as how the converter input impedance changes the EH source.

Therefore our implementation of this model uses these recorded values. Recording is supported by the same hardware as described in Section 4.1.1.

## 4.3 Modelling Converter Efficiency

The efficiency of a converter is dependent on the input voltage, input current, output voltage and output current. The input values change with the available amount of environmental energy. The output values change when for example a sensor node starting to send a radio packet. To cover this behavior in emulation, the converter efficiency should change as well. For the RF energy harvesting setup and solar energy harvesting setup, two different methods are used.

### 4.3.1 RF Harvesting Efficiency Modelling

The P2110 RF Powerharvester [32] contains both the energy harvester and converter in an integrated chip. Therefor it is not possible to tap into the converter input current/voltage signal. However the Powerharvester does provide a Received Signal Strength Indicator (RSSI), output $V_{\text{RSSI}}$, representing the actual RSSI. This signal is a voltage ranging between 10 mV and ~1 V. The mapping between $V_{\text{RSSI}}$ and the actual RSSI is shown in Figure 4.3. Note that to get the actual input power flowing into the capacitor, we should take the efficiency of the harvester into account. The efficiency is shown in Figure 4.4. Both graphs are re-plotted in Matlab, by visually extracting the data points. Using a cubic interpolation function, two functions are obtained from the graph: $f(V_{\text{RSSI}})$ and $\eta(P)$. $f(V_{\text{RSSI}})$ is the input power as function of the $V_{\text{RSSI}}$ signal, and $\eta(P)$ is the harvester efficiency as function of input power $P$. Using both functions, the actual input power $P_{\text{in}}$ can be extracted from the $V_{\text{RSSI}}$ signal

$$P_{\text{in}} = f(V_{\text{RSSI}})\eta(f(V_{\text{RSSI}})). \tag{4.8}$$

### 4.3.2 Solar Energy Harvesting Efficiency Modelling

The input and output efficiency of the bq25570 are shown in Figure 4.5 and Figure 4.6. These graphs are captured in lookup tables. These lookup tables are 2D-arrays which hold the convert efficiency and can be indexed using the input/output current. The graphs are defined on a logarithmic scale so the

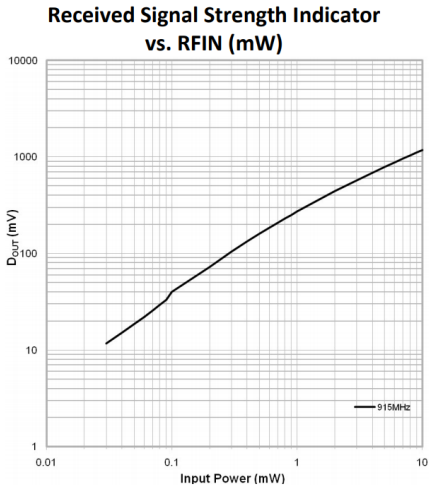



Figure 4.3: **Mapping of $V_{\mathbf{RSSI}}$ to actual input power [32].**

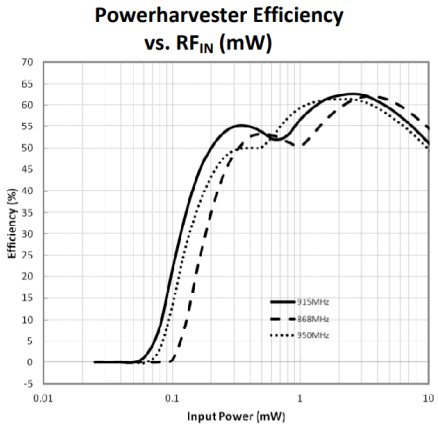



Figure 4.4: **Efficiency of Powerharvester based on input power [32].**

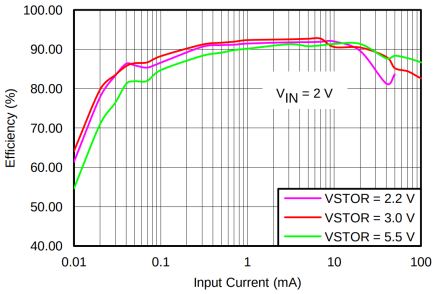



Figure 4.5: **Input efficiency of bq25570 for $V_{\mathbf{in}} = 2$ [49].**

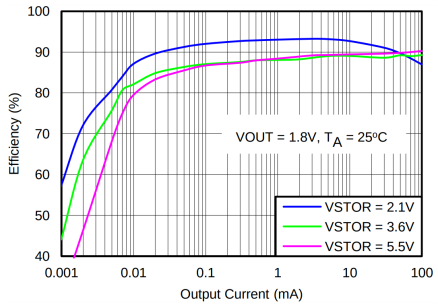



Figure 4.6: **Output efficiency of bq25570 for $V_{\mathbf{out}} = 1.8$ [49].**

current cannot be used to linearly lookup the efficiency. This is solved by splitting up the lookup tables, in to four separate lookup tables which have a linear x axis. Before doing the lookup operation, it is determined in which range the current is: 0–0.1 mA, 0.1–1 mA, 1–10 mA or 10–100 mA. Based on this, the appropriate lookup table is selected. Next the table index is calculated as

$$i_{\text{table}} = \frac{I_{\text{lookup}}}{I_{\text{max}}} n_{\text{entries}}, \tag{4.9}$$

where $i_{\text{table}}$ is the table index, $I_{\text{lookup}}$ is the input current, $I_{\text{max}}$ is the maximum current for that table and $n_{\text{entries}}$ is the amount of table entries.

Note that the efficiencies are not just dependent on current, but also on voltage. These are not taken into account in this implementation because they hardly contribute to an improvement in accuracy given the test parameters used during evaluation.

## 4.4 Shepherd Integration

In this section we describe how we integrated our emulation model in the Shepherd platform. Biggest challenge was to port our implementation written for a ARM Cortex-M4 micro controller with floating point unit and hardware division support, to a Programmable Real-Time Unit (PRU) core of the BeagleBone which does not have these features. Note that the emulation runs at a sampling rate of 100 kHz, leaving 10 µs of execution time for the entire event loop. In this event loop the PRU also has to sample its ADCs and communicate or exchange buffers with the main BeagleBone processor. This leaves in the worst case only ~2–3 µs for the emulation algorithm to execute.

### 4.4.1 Conversion To Non-Floating-Point

When working with floating points, the programmer usually does not have to worry about the size of variables. However, when working with integers, different issues come up, i.e. fractions cannot be stored, the result of a computation could be too large for the target variable and the remainder of every division is discarded. Especially the last problem is an issue, since our algorithm calculates the difference in capacitor voltage over a period of 10 µs, which can be in the order of nV.

#### Dynamically Updating Voltage Range

The Shepherd ADC defines the logic representation of voltage as

$$V_{\text{logic}} = V_{\text{real}} \frac{2^{18} - 1}{2 \cdot 4.092}. \tag{4.10}$$

From (4.10) it follows that the smallest value that can be represented in logic is 31.25 µV. As we said, this is not accurate enough. Therefore we increase the granularity of our logic representation by using the maximum amount of bits. Since we work with signed 32-bit numbers, we have 31 bits at our disposal. Our new logic voltage representation becomes:

$$V_{\text{logic}} = V_{\text{real}} \frac{2^{31} - 1}{2 \cdot 4.092}. \tag{4.11}$$

With this new representation our smallest value becomes ~3.815 nV which is accurate enough for the applications we tested during evaluation.

Now that we have tackled our remainder discard problem, the other problem concerning floating point to integer conversion pops up. If we multiply our new voltage value, it will quickly overflow. If we divide something with this new voltage, the result will quickly become zero.

Therefore we come up with a hybrid solution. When we want to store our new capacitor voltage, we use the accurate 31-bit representation. When we want to do an arithmetic operation, we bit shift the voltage back to the original 18-bit representation.

#### Representation of Fractions

Since our efficiency $\eta$ is defined as a real number ranging from 0.0 to 1.0 it cannot be directly represented as an integer. We mitigate this issue by first

multiplying our efficiency by a factor of $2^{13}$. We use this new number as input for the multiplication where the efficiency is required. After the multiplication, we divide the result by the same $2^{13}$ to get the correct value. We can do this division efficiently by right bit shifting the number with 13 bits. We have chosen this 13 bits as a trade-off between getting maximum accuracy, but still being able to multiply with a 18-bit voltage value without overflowing.

### 4.4.2 Reducing Divisions

As said before, in the worst case there is only ~2–3 µs execution time for the algorithm to run. Although the processor runs at 200 MHz, it does not have hardware division support. We have experimentally discovered that divisions can take up to ~2 µs. This leaves room for only one division in our algorithm. However, in the emulator code we identified four divisions in our algorithm and we decided to optimize the code as follows:

- In (3.2) and (4.9) the division is replaced by a multiplication with the multiplicative inverse of the divisor. This inverse would become a fraction and is handled in the same way as discussed in Section 4.4.1.
- In (3.6) the divisor, $V_{\text{cap}}$, is not a constant. Therefore we cannot precompute the multiplicative inverse. We solve this by replacing $V_{\text{cap}}$ with the constant $V_{\text{cavg}}$, the average cap voltage. This is calculated by taking the mean of the upper and lower threshold voltage.

### 4.4.3 Integer Square Root

In Section 4.1.2 we calculate the capacitor voltage drop at the moment the output turns on. This calculation uses a square root. There is no native support for integer square root. Therefore we use an implementation of a digit-by-digit square root algorithm. Here we still face the problem that the result of the square root will be smaller then 1. We solve this by first multiplying the input with $1024 \cdot 1024$. This ensures the result is larger than one, and the remaining fraction can be discarded while maintaining accuracy. That number is used to multiply with the capacitor voltage. To undo the multiplication in the previous stage, the output is divided by 1024. Because this is a factor of 2, the division can be replaced by a bit shift. In this way no unwanted divisions are used.

# Chapter 5

# Evaluation

In this chapter we evaluate the model described in Section 4.1. The evaluation is done by comparing emulation with the actual operation of two different EH setups. The test setup is shown in Section 5.1. The emulation of RF harvesting and solar energy harvesting is evaluated in Section 5.2 and Section 5.3. Finally Section 5.4 tells something about the repeatability of the emulation.

## 5.1 Experimental Setup

We evaluate our model by first recording the input and output of an actual energy harvester, given a certain input, storage capacitor and load. The recording is used as input for the emulation. Then we estimate our model parameters, run a computer simulation based on the measured input values and compare the results with the recorded output. Based on these results we optimize our model parameters. After the optimal model parameters for the computer simulation are found, the actual emulation is run on our emulator hardware described in Section 4.1.1. Final optimizations are done to cover the differences in computer simulation and hardware emulation.

### 5.1.1 Evaluation Metric

The emulation is evaluated by comparing the average output on/off times of the emulation and actual EH setup taken over a period of 10 seconds for RF emulation, and 40 seconds for solar emulation. This evaluation metric is chosen because it defines the behavior of an EH setup compared to a continuously powered setup. To use this as a fair metric, the load should be equal for both the emulation and EH setup.

Another metric we considered is the average amount of energy delivered over a period of time. However, this does not include off times, which is important information for intermittently-powered systems. Also, due to hardware constraints, it is not possible to record both input and output power at the same time with the custom developed hardware used during evaluation.

### 5.1.2 Hardware Setup

There are two similar hardware setups during recording.

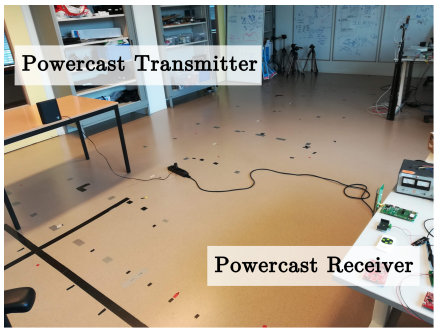



Figure 5.1: **Hardware setup during RF harvesting.**

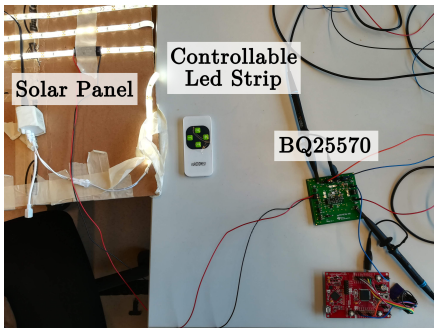



Figure 5.2: **Hardware setup during solar energy harvesting.**

### RF Harvesting Setup

Figure 5.1 shows the setup during the recording of RF harvesting. The TX91501 Powercaster Transmitter [10] is used as energy source. We use the P2110 Powerharvester Evaluation Board [32] as energy harvester and converter. Our emulator hardware is re-used to record the input power and output on/off times. The emulator outputs these recordings over a serial USB connection to a laptop. A schematic setup is shown in Figure 5.3.

The Powerharvester Evaluation Board comes with two storage capacitor sizes: 1000 µF and 10 mF. We envision our emulation to be used in intermittently-powered applications. The more storage capacity a sensor node has, the less it will run intermittently-powered. Therefore the 1000 µF capacitor is chosen. We have also experimented with smaller capacitor sizes but these caused the harvester to become unstable.

At first, a constant load is chosen to evaluate with, because a constant load can be easily simulated. Computer simulations are used to find the initial model parameters. We have chosen our load 377 Ω. This is small enough such that the system is never continuously powered, and large enough to mimic the power consumption of a sensor node while sending a radio packet. Finally we verify our model with a MSP430FR5969 development board as realistic target load.

### Solar Energy Harvesting Setup

Figure 5.2 shows the setup during the recording of solar energy harvesting. A dimmable led strip is used as environmental energy, a SLMD121H04L solar cell as energy harvester and a bq25570 as converter. Three capacitor values are used: 94 µF, 470 µF, 1000 µF. These values are chosen to test a range of on/off times, where 94 µF is the minimum required for the bq25570 to operate correctly. For each capacitor a constant load is selected: 3333 Ω, 1000 Ω and 480 Ω. These values are selected such that the capacitor is not instantly depleted and the output is never continuously on. As dynamic load a MSP430FR5969 development board is used, with and without a low-power radio. A schematic setup is shown in Figure 5.4.

The bq25570 efficiency is well specified in the datasheet for various input voltage and currents. The efficiency specifications of the P2110 Powerharvester Evaluation Board is however very limited. Therefore we have chosen to evaluate

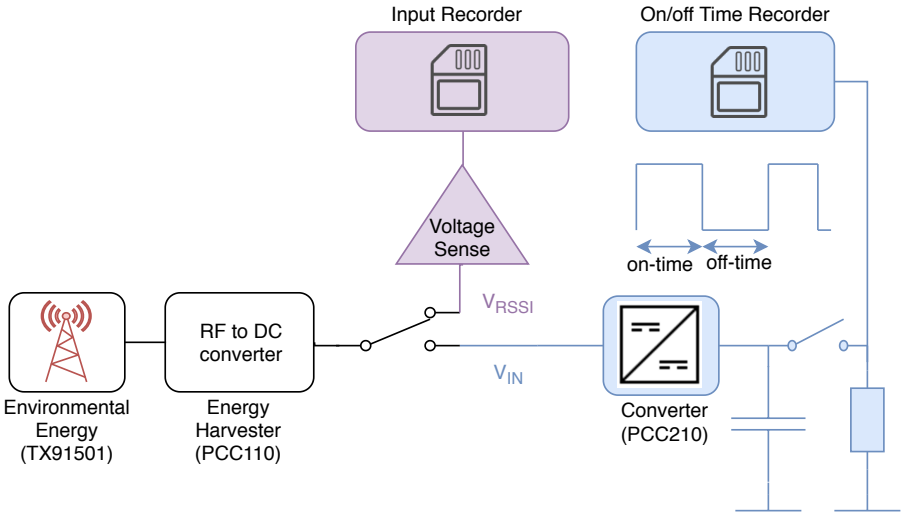



Figure 5.3: **Schematic setup during recording of RF harvesting. Recording happens in two stages. In the first stage the input power is recorded by measuring the $V_{RSSI}$, a voltage representing the actual Received Signal Strength Indicator (RSSI). In the second stage the output on/off times are recorded by connecting the EH output to the input $V_{in}$ of the PCC210 boost converter [33]. Note that these two stages cannot happen simultaneously. The Environmental energy is changed at every recording by varying the distance between the RF transmitter and RF receiver.**

the bq25570 setup more extensively compared to the RF harvesting setup.

### 5.1.3 Software Setup

During recording, the emulator hardware measures input current/voltage and output state at a sampling rate of 100 kHz. These measurements are send to a laptop for running simulations and evaluation. Outputting the values as text would take too much time for the Teensy to process. Therefore a small binary protocol is written to efficiently output the data as shown in Figure 5.5. To optimize sending, these 8 byte messages are buffered into a 64 byte package, to match the maximum packet size of full-speed USB. On the laptop runs a python script, which stores the recordings in a csv-file.

This csv-file is later read by a Matlab script which converts these readings into a binary file containing both input current and voltage. A Matlab and C/C++ simulation uses these readings as input. The simulation is used to find optimal model parameters. When optimal parameters are found, the binary input recordings file is stored on a SD card. The emulator hardware uses the SD card to get input power values.

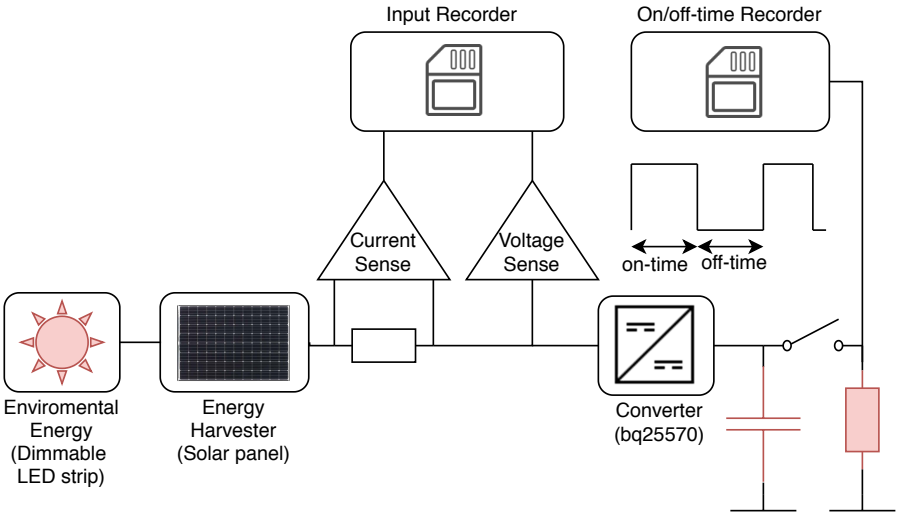



Figure 5.4: **Schematic setup during recording of solar energy harvesting. Input current/voltage and output on/off time is recorded. The light intensity, capacitor and load are varied over different recordings.**

| 8 bits | 8 bits | 8 bits | 8 bits | 16 bits | 16 bits |
|---|---|---|---|---|---|
| start byte | counter | reserved | output state | current | voltage |

Figure 5.5: **Representation of measured data to maximize data transfer on USB port (see Section 5.1.2). Start byte identifies the message. Counter is incremented every message and checked at the receiver to see if any packets are lost.**

## 5.2 RF Harvesting

In this section we evaluate the emulation of a P2110 Powerharvester. We evaluate the performance at five different input power levels by changing the distance between the RF transmitter and the Powerharvester. As described in Figure 5.3, we first measure the RSSI and then we do another measurement of the output on/off times.

Average output on/off times over a period of 10 seconds are used as a metric to evaluate emulation of the Powerharvester. The results of the evaluation are shown in Figure 5.6. The results show that for all test runs, the maximum error is less than 10%. Test runs with a constant load show a negligible error deviation. Test runs with a dynamic load show a maximum of 1.5% on time error deviation.

## 5.3 Solar Energy Harvesting

In this section we evaluate solar energy harvesting emulation with the bq25570, a nano power boost charger and buck converter for energy harvester powered

applications. It has two stages: the first stage boost the solar output voltage to a capacitor voltage ranging from 3.2 V to 3.5 V; the second stage is a buck-converter stepping the capacitor voltage down to a constant output of 2.23 V.

Note that the threshold voltages are only checked every ~50 ms. Therefore the output on/off times will always be a multiple of this number. We have implemented this behavior in our model. The converter evaluates the maximum power point every 16 seconds for a duration of ~256 ms. During this period the open circuit voltage of the solar panel is measured, and no energy is harvested. These mechanism can effect the accuracy of our evaluation. To counter this, we have increased the measurement period from 10 s to 40 s.

There are three main parameters in the system, which determine the converter output: input power, storage capacitor and the system load. Assume one wants to test $x$ data points for each of those parameters, one would end up with a total of $3^x$ tests to execute. This would quickly explode the amount of tests to run. Each test would require a change in hardware setup, 40 seconds to record and 40 seconds to emulate. Therefore we have come up with a limited set of test runs which is practical to test, but still explores a wide range of converter execution. Table 5.2 shows the set of test parameters and Figure 5.7 shows the result of the evaluation.

From the results it follows that with a constant load, the maximum error is less than 10% and maximum deviation is less than 1.5%. With a dynamic load we see similar results, except for test run 14, which shows an outlier. This can be explained as an quantization error. The bq25570 checks the capacitor threshold voltages every ~50 ms. Now the actual EH setup for test run 14 has an on time of ~100 ms. Assume the emulation measures the output current smaller than it actually is. This causes the virtual capacitor voltage at 100 ms to be still above the threshold voltage. Now the on time is increased to 150 ms. Because the capacitor is discharged for a longer period of time, the off time of the capacitor increases as well. Now we see a off time error of more than 30%. While as the bq25570 were to check the capacitor voltage continuously, the error would be smaller than 10%.

## 5.4 Repeatability

Key feature of our emulation is repeatability. This is a main advantage over using an actual energy harvester. The repeatability can be achieved by using the same input values for various emulations. This is hardly possible for an actual energy harvester, since environmental tends to change over time.

Given an equal input power trace and target load, the only variation between emulations is caused by noise on the ADC voltage and current sense. The on/off time error consistency is evaluated for all constant load measurements presented in Figure 5.6 and Figure 5.7. We have chosen to only evaluate for constant load situations, because dynamic loads introduce more deviation in on/off time which are not caused by the emulation, but by deviations in the load itself. After running seven consecutive emulations under the same input power and load conditions, we have shown that the deviation in output on/off time error is less than 1.5% for all constant load measurements.

| Test run | Mean input power (µW) | Storage capacitor (µF) | Load |
|---|---|---|---|
| 1 | 3279.4 | 1000 | 377 Ω |
| 2 | 3019.5 | 1000 | 377 Ω |
| 3 | 2144.7 | 1000 | 377 Ω |
| 4 | 1299.6 | 1000 | 377 Ω |
| 5 | 768.0 | 1000 | 377 Ω |
| 6 | 567.2 | 1000 | 377 Ω |
| 7 | 3453.6 | 1000 | MSP430 |
| 8 | 2670.8 | 1000 | MSP430 |
| 9 | 2369.7 | 1000 | MSP430 |
| 10 | 1669.4 | 1000 | MSP430 |
| 11 | 1224.4 | 1000 | MSP430 |
| 12 | 878.5 | 1000 | MSP430 |

Table 5.1: **Set of input parameters used for evaluating the RF energy harvesting. The MSP430 load runs an infinite loop while toggling a LED.**

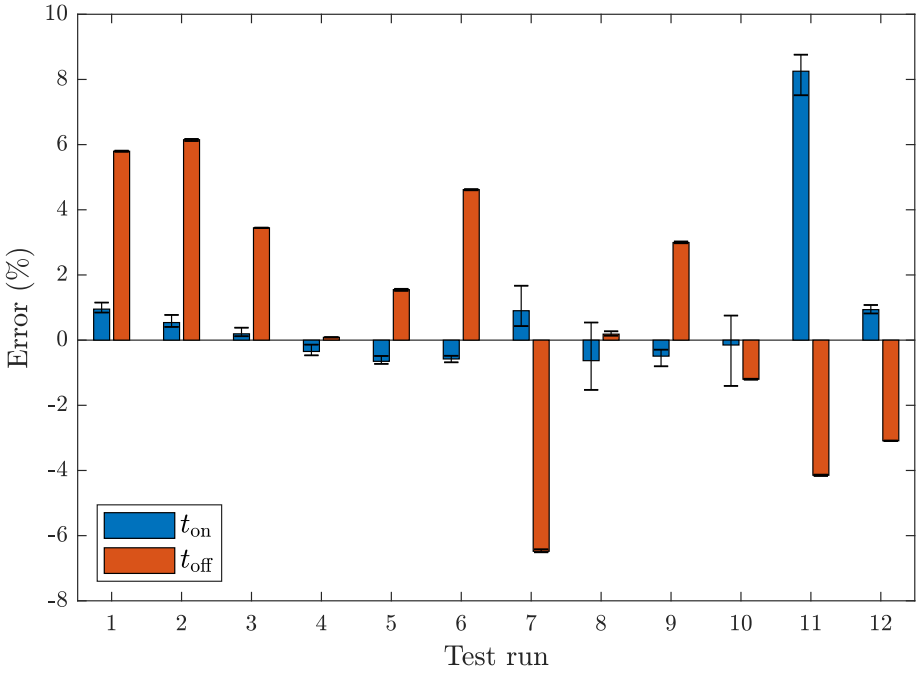



Figure 5.6: **RF energy harvesting emulation output on time error $t_{on}$ and output off time error $t_{off}$ measured over a period of 10 seconds. The emulations are repeated seven times to determine deviation. The error bars show the maximum and minimum deviation. Test run parameters are defined in Table 5.1.**

| Test run | Mean input power (µW) | Storage capacitor (µF) | Load |
|---|---|---|---|
| 1 | 867.5 | 1000 | 480 Ω |
| 2 | 1537.1 | 1000 | 480 Ω |
| 3 | 2254.5 | 1000 | 480 Ω |
| 4 | 889.5 | 470 | 1000 Ω |
| 5 | 1596.7 | 470 | 1000 Ω |
| 6 | 2279.1 | 470 | 1000 Ω |
| 7 | 416.8 | 94 | 3333 Ω |
| 8 | 887.2 | 94 | 3333 Ω |
| 9 | 1568.7 | 94 | 3333 Ω |
| 10 | 838.0 | 470 | MSP430 |
| 11 | 1497.6 | 470 | MSP430 |
| 12 | 2150.2 | 470 | MSP430 |
| 13 | 852.8 | 94 | MSP430 + Low-power Radio |
| 14 | 1462.8 | 94 | MSP430 + Low-power Radio |
| 15 | 2096.2 | 94 | MSP430 + Low-power Radio |
| 16 | 701.5 | 470 | MSP430 + Low-power Radio |
| 17 | 1411.0 | 470 | MSP430 + Low-power Radio |
| 18 | 2044.0 | 470 | MSP430 + Low-power Radio |

Table 5.2: **Set of input parameters used for evaluating the solar energy harvesting. The single MSP430 load runs an infinite loop while toggling a LED. The MSP430 with low-power radio sends a packet and then listens for incoming packets until power outage.**

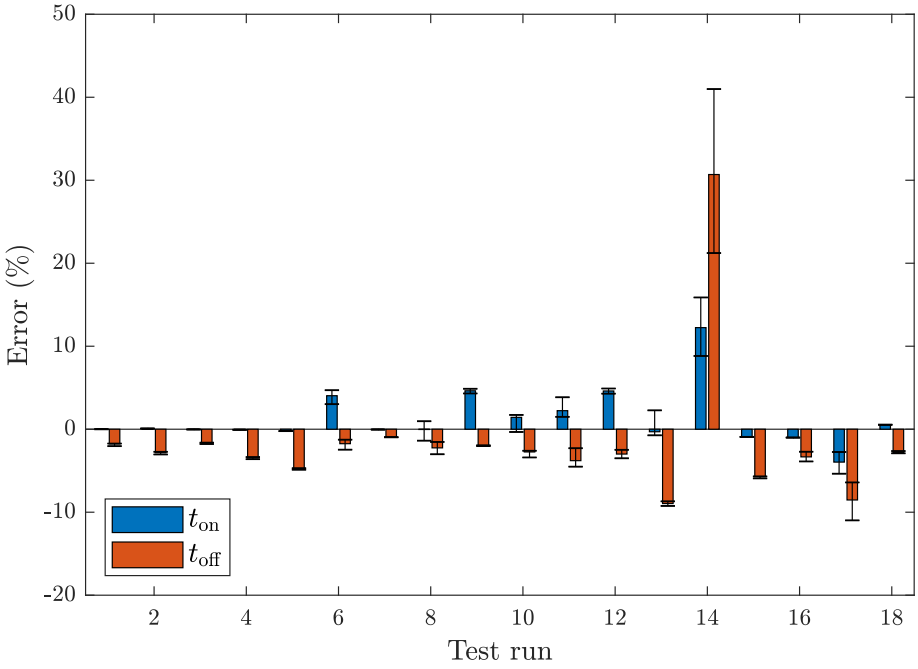



Figure 5.7: **Solar energy harvesting emulation output on time error $t_{on}$ and output off time error $t_{off}$ measured over a period of 40 seconds. The emulations are repeated seven times to determine the deviation. The error bars show the maximum and minimum deviation. Test run parameters are defined in Table 5.2.**

# Chapter 6

# Future Work

In this chapter we discuss possible future work divided over three topics: dynamic capacitance, energy-aware debugging and simulation/emulation synergy.

## 6.1 Dynamic Capacitance

The energy harvesting emulation model used in this thesis comes with one main advantage compared to the state of the art: it has a virtual capacitor. By means of software configuration, any capacitor value and threshold voltage can be emulated. Future work could exploit this feature to:

- find an optimal capacitor value for a wireless sensor network. The question one might ask: what is the impact of using a bigger or smaller capacitor on network stability or throughput?
- find an optimal capacitor value to perform some intermittent task. One way to do it, is to do a test sweep over various capacitor sizes and see what results in an optimal output.

## 6.2 Energy-Aware Debugging

Debugging is an essential process in developing embedded software. However debugging in a intermittently-powered, energy-constrained context implicates new problems. Work of [8] provides a solution to this problem with analog circuity to maintain a constant capacitor voltage while executing debugging tasks. This analog circuity introduces inaccuracy and complexity. Because in our work we virtualize the capacitor, we can maintain capacitor voltage purely in software. Therefore we can apply energy aware debugging with less complexity and with more accuracy.

## 6.3 Simulation/Emulation Synergy

Assume one wants to test a network with 1000 nodes. This is impractical with real hardware devices, but simulation could be an outcome. The model we presented works both in emulation and simulation. Energy consumption

of a WSN node is extensively modeled, for example in [55, 40, 3]. Combining these existing energy consumption models with our energy harvesting emulation model enables simulation of a large (WSN) network. Depending on the accuracy of the energy consumption model, one can have a simulation which can be as accurate as the emulation because they use the same code base.

# Chapter 7

# Conclusions

In this thesis we have designed, implemented and evaluated a generic energy harvesting emulation model used for intermittently-powered batteryless IoT devices. The model is based on the work of [12]. While [12] fails to accurately emulate a real world situation, our model implementation can repeatably emulate the entire input power chain of an energy harvesting device, including the energy harvester, converter(s) and storage capacitor. We evaluated our emulation with two different setups: a Powercast RF harvester [32] and an industry standard bq25570 energy harvesting boost-converter [49] with solar power input.

We have shown for RF harvesting emulation that the mean output on/off time error is less than 8%, measured with a 1000 µF storage capacitor over various input power signal strengths using both a constant and dynamic load. For the bq25570 solar energy harvesting emulation we have measured the emulation accuracy more extensive using different storage capacitors with both constant loads and dynamic loads. This emulation has shown a similar error margin of less than 10%. However, there are some outliers caused by quantization of the bq25570. Under constant load conditions, the on/off time error deviation is less than 1.5% for both RF and solar. Thereby we conclude that the emulation is realistic and repeatable.

We envision our emulation to be used in the context of a testbed for batteryless systems powered by harvested energy. Therefore we have integrated our emulator in Shepherd [21], a state-of-the-art testbed for batteryless IoT devices.

# Appendix A

# Initial Testbed Setup and Interface

In this appendix we envision our initial setup of the testbed and compare it with Shepherd. At the start of this thesis, Shepherd was not yet published. Six months after the start of this thesis, we were confronted with the Shepherd paper. At that point it was decided to re-focus the thesis towards the emulation part and integrate our work into the Shepherd platform.

## A.1 Our Proposal

The initial vision on our testbed was to have a central server which users can access remotely via a web interface and upload their firmware. This central server would connect to testbed nodes, schedule tests to run and present the results to the users via the same web interface. This is a similar setup as the WSN testbeds discussed in Section 2.2.

Since the setup would be similar, we contacted all actively maintained WSN testbeds if it was possible to share their source code so we could re-use it. Only Flocklab was willing to do this. However, since Flocklab exists for over 10 years, their source code is out-dated, hard to change and not future proof. Therefore a new web interface is developed to provide a proof-of-concept for the testbed setup, shown in Figure A.1.

Now the interface is defined, the testbed nodes are left to be defined. Main features are the ability to emulate energy harvesting conditions, program a target device, record the power consumption and GPIO state changes, and communicate with the central server.

Networking is one of the most interesting topics when it comes down to intermittently-powered energy harvesting devices. Therefore we envision our testbed to support up to four target devices. Our proposed testbed node architecture is shown in Figure A.2. Judging by the survey on intermittently-powered platforms in Table 2.2, we see that the MSP430 is the most used MCU. Therefore we support the programming of MSP430 target devices.

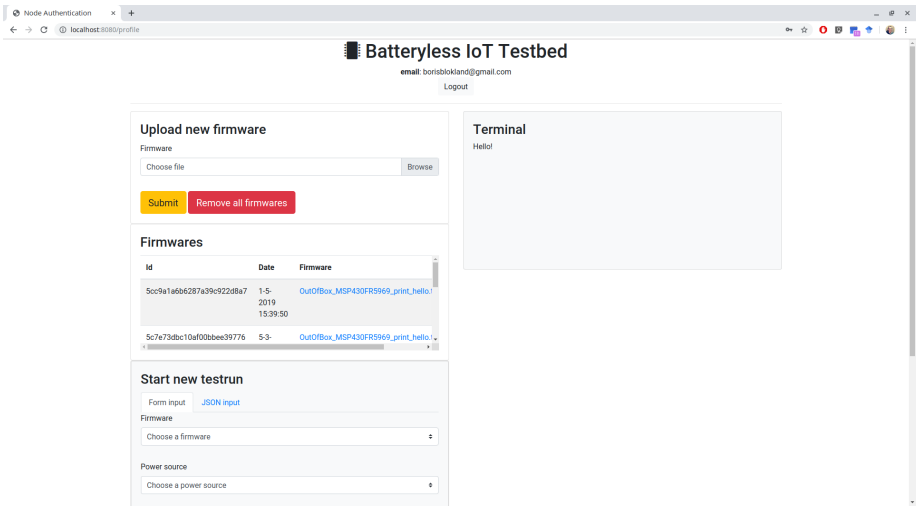



Figure A.1: **Screenshot of the web interface in proof-of-concept testbed. Users can upload firmware, run a test and download test results (logic analyzer trace and serial output log). While test is running, terminal outputs state of the test.**

## A.2 Shepherd

Besides the emulation Shepherd offers interesting testbed features. The testbed consist of a BeagleBone [5] with a hardware extension cape on top of it. The BeagleBone is a small embedded computer comparable to the more popular Raspberry Pi [34]. Due to its small size the testbed is portable. A bi-directional level converter is provided such that the target device, which could operate at another voltage level, can output debug messages over UART or output information by toggling GPIO pins. These debug messages and GPIO state changes are recorded with their time information. Shepherd also allows time-synchronized emulation over various nodes, such that multiple devices can be tested at the same time, using time-synchronized input recordings.

The user interface of the testbed is a simple command-line utility, which can be executed via ssh. There is no central server which handles all the tests. Instead, to run a test on multiple nodes, another command-line utility is available. This provides similar functionality, but takes a list of Shepherd nodes as input argument.

## A.3 Comparison and Further Direction

Looking at our proposed setup and the Shepherd's implementation, we see that Shepherd lacks the central server and user interface. On the other hand, we see that the testbed node architecture is practically the same as our proposed setup. Therefore we envision Shepherd could work as a testbed node in our proposed WSN testbed.

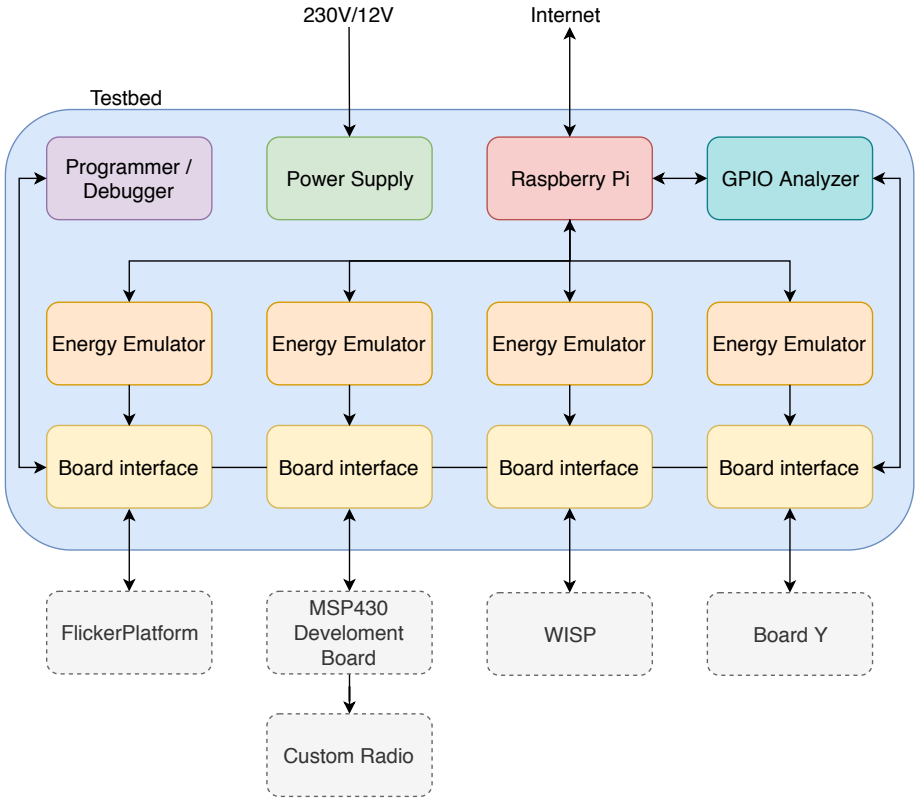



Figure A.2: **Proposed testbed node architecture. Up to four energy emulator channels are supported, controlled by a Raspberry Pi, also managing internet connection, device programming and GPIO analysis.**

# Bibliography


[1] Henko Aantjes, Amjad Yousef Majid, and Przemysław Pawełczak. A testbed for transiently powered computers. In *Proc. HLPC (ASPLOS workshop)*, Atlanta, USA, April 2–6, 2016. ACM.

[2] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frederic Saint-Marcel, Guillaume Schreiner, and Julien Vandaele. FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed. In *Proc. WF-IoT*, Milan, Italy, December 14–16, 2015. IEEE.

[3] Vivek Agarwal, Raymond A. DeCarlo, and Lefteri H. Tsoukalas. Modeling energy consumption and lifetime of a wireless sensor node operating on a contention-based MAC protocol. *IEEE Sensors Journal*, 17(16):5153–5168, 2017.

[4] Paramasiven Appavoo, Ebram Kamal William, Mun Choon Chan, and Mobashir Mohammad. Indriya2: A Heterogeneous Wireless Sensor Network (WSN) Testbed. In *Proc. TRIDENTCOM*, Shanghai, China, December 1–3, 2018. Springer.

[5] BeagleBoard.org Foundation. BeagleBone. `https://beagleboard.org`, 2019. Last accessed: Sep. 27, 2019.

[6] BelluTech. Always-on sensing with battery life up to 10x longer, or batteryless. `https://bellutech.com/`, September 2017. Last accessed: Oct. 16, 2019.

[7] Naveed Anwar Bhatti. *System support for transiently-powered embedded sensing systems.* Phd thesis, Politecnico di Milano, Milano, Italy, 2018.

[8] Alexei Colin, Graham Harvey, Alanson P Sample, and Brandon Lucia. An Energy-Aware Debugger for Intermittently Powered Systems. *IEEE Micro*, 37(3):116–125, 2017.

[9] Alexei Colin, Emily Ruppel, and Brandon Lucia. A Reconfigurable Energy Storage Architecture for Energy-harvesting Devices. In *Proc. ASPLOS*, Williamsburg, USA, March 24–28, 2018. ACM.

[10] Powercast Corp. TX91501 915 MHz Powercaster Transmitter. `https://www.powercastco.com/wp-content/uploads/2016/11/User-Manual-TX-915-01-Rev-A-4.pdf`, August 2016. Last accessed: Oct. 17, 2019.

[11] Crossbow. Telosb Mote Platform. https://www.willow.co.uk/TelosB_Datasheet.pdf, 2012. Last accessed: Oct. 17, 2019.

[12] Pieter De Mil, Bart Jooris, Lieven Tytgat, Ruben Catteeuw, Ingrid Moerman, Piet Demeester, and Ad Kamerman. Design and implementation of a generic energy-harvesting framework applied to the evaluation of a large-scale electronic shelf-labeling wireless sensor network. *EURASIP Journal On Wireless Communications and Networking*, 2010(1):34–36, 2010.

[13] Samuel DeBruin, Bradford Campbell, and Prabal Dutta. Monjolo: An energy-harvesting energy meter architecture. In *Proc. SenSys*, Rome, Italy, November 11–14, 2013. ACM.

[14] Manjunath Doddavenkatappa, Mun Choon Chan, and Akkihebbal L Ananda. Indriya: A low-cost, 3D wireless sensor network testbed. In *Proc. TRIDENTCOM*, Shanghai, China, April 17–19, 2011. Springer.

[15] EnOcean. EnOcean Products. https://www.enocean.com/en/products/, November 2009. Last accessed: Oct. 16, 2019.

[16] Everactive. Everactive bateryless wireless sensors generate a new class of data for the analytics of the future Industrial IoT. https://everactive.com/, June 2019. Last accessed: Oct. 16, 2019.

[17] FIT IoT-LAB. WSN430 Open Node. https://www.iot-lab.info/hardware/wsn430/, April 2011. Last accessed: Oct. 17, 2019.

[18] FIT IoT-LAB. A8 Open Node. https://www.iot-lab.info/hardware/a8/, May 2014. Last accessed: Oct. 17, 2019.

[19] FIT IoT-LAB. M3 Open Node. https://www.iot-lab.info/hardware/m3/, May 2014. Last accessed: Oct. 17, 2019.

[20] Francesco Fraternali, Bharathan Balaji, Yuvraj Agarwal, Luca Benini, and Rajesh Gupta. Pible: battery-free mote for perpetual indoor BLE applications. In *Proc. BuildSys*, Shenzhen, China, November 7–8, 2019. ACM.

[21] Kai Geissdoerfer, Mikołaj Chwalisz, and Marco Zimmerling. Shepherd: A Portable Testbed for the Batteryless IoT. In *Proc. SenSys*, Ney York, USA, November 10–13, 2019. ACM.

[22] Alexander Gluhak, Srdjan Krco, Michele Nati, Dennis Pfisterer, Nathalie Mitton, and Tahiry Razafindralambo. A Survey On Facilities For Experimental Internet Of Things Research. *IEEE Communications Magazine*, 49(11):58–67, 2011.

[23] Josiah Hester, Lanny Sitanayah, Timothy Scott, and Jacob Sorber. Realistic and repeatable emulation of energy harvesting environments. *ACM Transactions on Sensor Networks*, 13(2):16:1–16:33, April 2017.

[24] Josiah Hester and Jacob Sorber. Flicker: Rapid Prototyping for the Batteryless Internet-of-Things. In *Proc. SenSys*, Delft, Netherlands, November 5–8, 2017. ACM.

[25] Geon-Tae Hwang, Venkateswarlu Annapureddy, Jae Hyun Han, Daniel J. Joe, Changyeon Baek, Dae Yong Park, Dong Hyun Kim, Jung Hwan Park, Chang Kyu Jeong, and Kwi-Il Park. Self-powered wireless sensor node enabled by an aerosol-deposited PZT flexible energy harvester. *Advanced Energy Materials*, 6(13):16–23, 2016.

[26] Roman Lim, Federico Ferrari, Marco Zimmerling, Christoph Walser, Philipp Sommer, and Jan Beutel. FlockLab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Proc. IPSN*, Philadelphia, USA, April 8–11, 2013. ACM.

[27] Matrix Industries. Unprecedented efficiency in self-charging thermoelectric devices. `https://www.matrixindustries.com/`, October 2018. Last accessed: Oct. 16, 2019.

[28] Microchip. AT86RF231 High Performance RF-CMOS 2.4 GHz Radio Transceiver. `https://www.microchip.com/wwwproducts/en/at86rf231?tab=tools`, 2012. Last accessed: Oct. 17, 2019.

[29] Nowi Energy. NOWI: Enabling the Internet of Things. `https://www.nowi-energy.com/`, July 2019. Last accessed: Oct. 16, 2019.

[30] Paul J. Stoffregen. Teensy USB Development Board. `https://www.pjrc.com/store/teensy35.html`, 2014. Last accessed: Sep. 04, 2019.

[31] Powercast Corp. Powercast hardware. `http://www.powercastco.com`, 2014. Last accessed: Mar. 30, 2018.

[32] Powercast Corp. P2110B 915 MHz RF Powerharvester Receiver. `https://www.powercastco.com/wp-content/uploads/2016/12/P2110B-Datasheet-Rev-3.pdf`, December 2016. Last accessed: Sep. 04, 2019.

[33] Powercast Corp. PCC110/PCC210 Powerharvester Chipset. `https://www.powercastco.com/wp-content/uploads/2018/06/PCC110-PCC210-Overview-V1.6-ONE-PAGE.pdf`, May 2018. Last accessed: Oct. 21, 2019.

[34] Raspberry Pi Foundation. Raspberry Pi 4. `https://www.raspberrypi.org/`, 2019. Last accessed: Oct. 17, 2019.

[35] Alberto Rodriguez, Domenico Balsamo, Zhenhua Luo, Steve P. Beeby, Geoff V. Merrett, and Alex S. Weddell. Intermittently-powered energy harvesting step counter for fitness tracking. In *Proc. SAS*, Glassboro, USA, March 13–15, 2017. IEEE.

[36] Alanson P. Sample, Daniel J. Yeager, Pauline S. Powledge, Alexander V. Mamishev, and Joshua R. Smith. Design Of An RFID-Based Battery-Free Programmable Sensing Platform. *IEEE Transactions On Instrumentation And Measurement*, 57(11):26–28, 2008.

[37] Semtech. SX1211 ultra Low Power (3mA RX) RF Transceiver 862-960 MHz. `https://www.semtech.com/products/wireless-rf/fsk-transceivers/sx1211`, January 2018. Last accessed: Oct. 16, 2019.

[38] Uvis Senkans, Domenico Balsamo, Theodoros D. Verykios, and Geoff V. Merrett. Applications of Energy-Driven Computing: A Transiently-Powered Wireless Cycle Computer. In *Proc. ENSSys*, Delft, Netherlands, November 5, 2017. ACM.

[39] Yongming Shi, Yao Wang, Yuan Deng, Hongli Gao, Zhen Lin, Wei Zhu, and Huihong Ye. A novel self-powered wireless temperature sensor based on thermoelectric generators. *Energy Conversion and Management*, 80(1):110–116, 2014.

[40] Boris Snajder, Vana Jelii, Zoran kalafati, and Vedran Bilas. Wireless sensor node modelling for energy efficiency analysis in data-intensive periodic monitoring. *Ad Hoc Networks*, 49(3):29–41, 2016.

[41] STMicroelectronics. STM32L433CC ultra-low-power with FPU ARM Cortex-M4 MCU 80 MHz with 256 Kbytes Flash, LCD, USB. `https://www.stmicroelectronics.com.cn/en/microcontrollers-microprocessors/stm32l433cc.html`, October 2016. Last accessed: Oct. 16, 2019.

[42] Texas Instruments. MSP430 ultra-low-power sensing and measurement MCUs. `http://www.ti.com/microcontrollers/msp430-ultra-low-power-mcus/overview.html`, March 2006. Last accessed: Oct. 16, 2019.

[43] Texas Instruments. CC2538 Powerful System-On-Chip for 2.4-GHz IEEE 802.15.4-2006 and ZigBee Applications. `http://www.ti.com/product/CC2538`, December 2012. Last accessed: Oct. 15, 2019.

[44] Texas Instruments. CC2420 Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee Ready RF Transceiver. `http://www.ti.com/product/CC2420/support`, May 2014. Last accessed: Oct. 16, 2019.

[45] Texas Instruments. CC2650STK SimpleLink Bluetooth low energy/Multi-standard SensorTag. `http://www.ti.com/tool/CC2650STK`, January 2016. Last accessed: Oct. 17, 2019.

[46] Texas Instruments. CC1350STK Simplelink CC1350 SensorTag Bluetooth and Sub-1GHz Long Range Wireless Development Kit. `http://www.ti.com/tool/CC1350STK`, January 2017. Last accessed: Oct. 17, 2019.

[47] Texas Instruments. bq25504 ultra low-power boost converter with battery management for energy harvester applications. `http://www.ti.com/lit/ds/symlink/bq25504.pdf`, March 2019. Last accessed: Oct. 15, 2019.

[48] Texas Instruments. bq25505 ultra low-power boost charger with battery management and autonomous power multiplexer for primary battery in energy harvester applications. `http://www.ti.com/lit/ds/symlink/bq25505.pdf`, March 2019. Last accessed: Oct. 15, 2019.

[49] Texas Instruments. bq25570 nano power boost charger and buck converter for energy harvester powered applications. `http://www.ti.com/lit/ds/symlink/bq25570.pdf`, March 2019. Last accessed: Oct. 11, 2019.

[50] Anne-Sophie Tonneau, Nathalie Mitton, and Julien Vandaele. How to choose an experimentation platform for wireless sensor networks? A survey on static and mobile wireless sensor network experimentation facilities. *Ad Hoc Networks*, 30(1):115–127, 2015.

[51] Rushi J. Vyas, Benjamin B. Cook, Yoshihiro Kawahara, and Manos M. Tentzeris. E-WEHP: A batteryless embedded sensor-platform wirelessly powered from ambient digital-TV signals. *IEEE Transactions On Microwave Theory And Techniques*, 61(6):2491–2505, 2013.

[52] Zhong Lin Wang. Self-powered nanosensors and nanosystems. *Advanced Materials*, 24(2):280–285, 2012.

[53] Williot. Battery-Free Bluetooth Technology - Connecting People with Products. `https://www.wiliot.com/about`, January 2018. Last accessed: Oct. 16, 2019.

[54] Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. Moo: A batteryless computational RFID and sensing platform. *University of Massachusetts Computer Science Technical Report UM-CS-2011-020*, 2011. `https://spqr.eecs.umich.edu/moo/`.

[55] Hai-Ying Zhou, Dan-Yan Luo, Yan Gao, and De-Cheng Zuo. Modeling of node energy consumption for wireless sensor networks. *Wireless Sensor Network*, 3(01):18, 2011.