# ENERGY EFFICIENCY FOR IOT SOFTWARE IN THE LARGE PROGRESS REPORT

FRANCISCO SANT'ANNA, UERJ

## 1. ORIGINAL PROPOSAL

Considering the projected scale of the IoT for the next decade and the role of low-power standby towards energy efficiency [1], this research project has the following goals:

(1) Address energy efficiency through rigorous use of standby.
(2) Target constrained embedded architectures that form the IoT.
(3) Provide standby mechanisms at the programming language level that scale to all applications.
(4) Support transparent/non-intrusive standby mechanisms that reduce barriers of adoption.

This proposal lies at the bottom of the software development layers—transparent programming language mechanisms—meaning that *all* applications would take advantage of low-power standby modes automatically, without extra programming efforts. We expect that existing energy-unaware applications will benefit from savings in the order of 50%. We will require a budget of R$70k: R$60k to support three students, R$5k for equipment, and R$5k for publication and travel costs.

The bulk of this proposal is grounded on the design of CÉU [3], a new reactive programming language targeting resource-constrained embedded systems, which I have been working for the past 8 years. The language is grounded on the synchronous concurrency model, which trades power for reliability and has a simpler model of time that suits most requirements of IoT applications. In this model, all reactions to the external world are guaranteed to be computed in bounded time, ensuring that applications always reach an idle state amenable to standby mode.

## 2. TIMELINE OF MAIN EVENTS

**Jul/17:** Early explorations with the core idea of the project.

**Sep/17:** Submission of the proposal to Serrapilheira.

**Dec/17:** Grant award notification from Serrapilheira.

**Jan/18:** Submission of two student scholarships to *Google Summer of Code (GSoC'18)*[1].

**Fev/18:** First fully-working prototype of CÉU with support for interrupts and a power manager.

[1] *GSoC'18*: `https://summerofcode.withgoogle.com/`

**Fev/18:** Submission to *LCTES'18* [2] of a work-in-progress paper entitled *"Transparent Standby for Low-Power, Resource-Constrained Embedded Systems: A Programming Language-Based Approach"* [4] and a full paper entitled *"A Memory-Bounded, Deterministic and Terminating Semantics for the Synchronous Programming Language* CÉU*"* [5].

**Mar/18:** Admission into the graduate program of Electrical Engineering at UERJ.

Invitation to be program chair of *REBLS'18* [3].

**Mar-May/18:** Engineering efforts on the prototype: refactoring, optimizations, more drivers, multiple platforms (AVR & SAMD).

**Apr/18:** Grant agreement signature with Serrapilheira.

Grant award notification from Google for the two *GSoC'18* students[4]: Anny (from UERJ) will work on a integrated programming environment for CÉU. Naveen (from India) will work on new device drivers.

**Apr/18:** First full-application experiments with significant energy savings ranging from 20-99%.

**Jun/18:** Admission of Anna (from PUC-Rio) as an intern to work on new IoT applications.

**Jun/18:** Trip to present the two papers on *LCTES'18*.

## 3. PROGRESS REPORT

Since the grant notification at the end of last year, our main goal was to reach, as fast as possible, a working prototype of our core research idea. I worked towards a new version of our programming language CÉU that would provide automatic standby for all applications: a programmer writes an IoT application unaware of standby or energy constraints and the language puts the device to sleep whenever possible.

In parallel, I worked with colleagues from PUC-Rio on a formal semantics of CÉU and a proof that all valid programs will react to an external event in bounded time, with bounded memory, and deterministic behavior. This proof is an important result because it shows that applications always reach an idle state amenable to standby. The formalization of CÉU already appeared on my PhD thesis [2], but the proofs are new results from this year.

By the end of February was the submission deadline for *LCTES'18*, the main annual venue that links programming languages and embedded systems. Fortunately, we had enough time to submit a full paper with the proofs [5] and a work-in-progress paper describing our approach for transparent standby at the language level [4], both of which were accepted for publication. Even though we

[2] *19th Annual ACM SIGPLAN / SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*: `https://conf.researchr.org/track/LCTES-2018/LCTES-2018-papers`.

[3] *5th ACM SIGPLAN Workshop on Reactive and Event-based Languages & Systems*: `https://2018.splashcon.org/track/rebls-2018-papers`.

[4] *LabLua@GSoC'18*: `https://summerofcode.withgoogle.com/organizations/5641514328260608/`

| | Arduino | Céu | |
|---|---|---|---|
| | | M1 | M2 |
| Empty | 3.7 | 0.002 | |
| Blink | 6.0 | 3.1 | |
| Sensor | 11.4 | 7.7 | |
| Radio | 19.5 | 15.8 | 3.0 |
| Protocol | 19.6 | 15.9 | |

```
// "Blink" in C/Arduino
void loop (void) {
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
}
```

```
// "Blink" in Ceu
loop do
    emit PIN(13, high);
    await 1s;
    emit PIN(13, low);
    await 1s;
end
```

FIGURE 1. Energy consumption for 5 scenarios. “Blink” in C/Arduino and CÉU.

did not have a comprehensive evaluation of energy savings, the conceptual framework and working prototype raised enough interest in the community for discussion on the conference.

At this moment, we have more concrete evaluation scenarios that show significant reduction in energy consumption due to standby in the microcontroller and other peripherals. Figure 1 shows the energy consumption of five applications originally in C/Arduino which we rewrote in CÉU. The first case illustrates how a naive “Empty” application in C/Arduino remains 100% of the time awake while the same application in CÉU sleeps in the most efficient standby mode. The “Blink” example shows that CÉU preserves the sequential, easy-to-read style of C/Arduino, while consuming much less energy. This is due to the assumptions CÉU, based on its tractable semantics, can extract from the program to be sure that it can sleep on the calls to `await`. The “Radio” example is another interesting case because the radio driver in CÉU identifies that the radio peripheral can enter in sleep mode to consume much less energy (toggling between active and passive radio modes).

Considering the second phase to come and the interactions with Serrapilheira since the beginning of the first phase and, it became more and more clear that the grant evaluation criteria will be mostly based on long-term relevance of the research and that it will encourage more fundamental aspects. In this direction, we chose to invest considerable efforts on the foundations of the language, highlighting its distinguished semantic aspects that enable transparent standby. The paper with the termination proof is the first result in this front, and we want to continue in this direction by employing automatic verification tools, such as TODO. More concretely, we want to automate the manual proofs we currently have and also answer program specification questions such as ”Can we prove that this application will turn off the radio 10 minutes after receiving a certain message?” Another education At the beginning of this year I was invited to be the program of *REBLS’18* which is The REBLS fundamentals of reactive languages foundational models of reactive languages

Specification correctness

Dafny is a verification-aware programming language - program verifier

LiquidHaskell logical predicates that let you enforce critical properties at compile time - Avoid Infinite Loops - Enforce Correctness Properties - Prove Laws by Writing Code

REBLS is about the future of our area

How IoT development will affect the new generation and how our undergraduate courses are ready to .

- new results - education: IoT and Cyber physical strong on graduate, not - lack of accessible languages and concurrency programming environments - specially energy efficiency

- incluir o standby de componentes na avaliacao - mostrar o que nao d pra fazer com TinyOS - como seria o caso de par/or em que o finalize remove o periferico da lista de possiveis awakes? - e o caso de componentes que so inteiramente desligados - caso de um par/or que mata o - radio listening - o radio inteiramente - como desligar dentro de finalize coisas que so desligadas via emit/await?

questions such as, can I prove that this particular program will

My initial budget plan was to spend around R$5k per month with research staff (80% of the budget). In particular, we wanted a professional embedded system engineer early in the project full time on the heavy-lifting work on drivers and platforms to serve as basis for for developing IoT applications. However, since the grant became available only on April, I could not hire such professional. Nonetheless, I could dedicate most of my time until May on the engineering aspects of the project, since the core language mechanisms were developed between December and February. At the end, the outcome was actually favorable since I overestimated the importance of an experienced domain-specific engineer at an early stage. The saved money was redirected to the international trip which was not initially planned.

Another difficulty that we had to deal was the lack of students to work in the area of embedded systems, even when offering above-average remuneration. Furthermore, as a new professor in my University I was not associated with any graduate program until mid March, when I joined the Electrical Engineering program in the field of *Networks & Distributed Systems*. However, we currently only offer Master degrees and most of the students have part-time commitment with the program. The scholarships from Google for *GSoC'18* raise more interest on students and we could hire an undergraduate student that is working

part time

(Doctorate degrees

the University currently has no Doctorate program with

On April we were notified about two scholarships

My current plan is

Another

porting SAMD From the very beginning,

the concept

describing the

all programs

Cu is a synchronous programming language for embedded soft real-time systems. It focuses on control-flow safety features, such as safe shared-memory concurrency and safe abortion of lines of execution, while enforcing memory bounded, deterministic, and terminating reactions to the environment. In this work, we present a small-step structural operational semantics for Cu

and a proof that reactions have the properties enumerated above: that for a given arbitrary timeline of input events, multiple executions of the same program always react in bounded time and arrive at the same final finite memory state.

By the end of February

- Theoretical - Core - Engineering - Experiments

## REFERENCES

[1] OECD/IEA. More data less energy—Making network standby more efficient in billions of connected devices. Technical report, International Energy Agency, 2014.

[2] F. Sant'Anna. *Safe System-level Concurrency on Resource-Constrained Nodes with Céu*. PhD thesis, PUC–Rio, 2013.

[3] F. Sant'anna, R. Ierusalimschy, N. Rodriguez, S. Rossetto, and A. Branco. The design and implementation of the synchronous language cÉu. *ACM Trans. Embed. Comput. Syst.*, 16(4):98:1–98:26, July 2017.

[4] F. Sant'Anna, A. Sztajnberg, A. L. de Moura, and N. Rodrigues. Transparent standby for low-power, resource-constrained embedded systems: A programming language-based approach (short wip paper). In *Proceedings of the 19th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, LCTES 2018, pages 94–98, New York, NY, USA, 2018. ACM.

[5] R. C. M. Santos, G. F. Lima, F. Sant'Anna, R. Ierusalimschy, and E. H. Haeusler. A memory-bounded, deterministic and terminating semantics for the synchronous programming language céu. In *Proceedings of the 19th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, LCTES 2018, pages 1–18, New York, NY, USA, 2018. ACM.