PART B: Project Proposal

SUMMARY OF THE PROJECT

Heading	Description
Project Title	Certified IoT Application
Project Acronym	CIoTA
Project Keywords	Formal programming language, certification, Internet of Things, proof system, reliability, interactive systems, explainability
Project Summary	Applications of the Internet of Things (IoT) immerse civilian and industrial infrastructures into an interconnected web of hybrid devices which, from many other benefits, improves coordination and observation of complex distributed systems by sensing and actuating the environment. As such devices are given more effectful tasks, guaranteeing their safety becomes critical both individually (the effect of a program error on its immediate environment) and collectively (the effect of a local failure on synchronized networks). Much research has studied embedded software safety: static analysis detects logical errors before running programs, Theorem provers synthesize programs that are correct by design. However, little efforts considers the combined effects of running a program in its environment, that is, the implicit protocols that govern the actions of an individual IoT device and the collective reactions in incurs on interconnected devices, digital or analog: the decisions it takes affect the immediate surrounding (energy consumption, actuation) or the whole network (traffic rerouting). These are not observable from within the logic of the program, yet should be reflected. The **CIoTA – Certified IoT Application –** project advocates logical reasoning to comprehend the co-evolution of IoT “apps” and their environment. CIoTA proposes to use automated theorem proving to model abstractions of such "apps" and the protocols that relate them with their environment, to reason about runtime resource consumptions, such as energy, time, and more generally on the effects a program has on its networked environment through sensing and actuation. CIoTA will be applied on effects of critical C programs, widely used for the development of IoT applications, refactored using a theorem prover. CIoTA therefore leverages mathematical tools and formal verification to enable programming engineers to increase safety, reliability, and predictability of sensitive IoT applications with mathematical proofs.

PROJECT INSERTION IN BRITTANY’S S3

For a complete list of Brittany S3 objectives, Strategic Innovation Areas and subdomains please refer to the Guide of Applicants. Indicate a and/or b, and c.

Justify the choice of Strategic Objective(s), Strategic Innovation Areas (SIA) and subdomains in section 2.1.

Heading	Description

a	Strategic Innovation Areas (SIA)	The secure, responsible digital economy
	Subdomains	3.1 Cybersecurity 3.4 Images and content / Networks and iot 3.6 Sober and responsible digital technologies 3.7 Data and intelligence
b	Insertion in the transversal axis on transitions	Yes Digital and industrial transition Energy and ecological transitons
c	Strategic Objective(s)	**Secure and responsible digital economy:** SO7: Boosting innovation in digital technologies and applications (electronics, photonics, space, Images and content, networks and connected objects, mobility) SO9: Building leadership in the European cybersecurity and digital security industry **Digital and industrial transition:** SO17: Incorporating dimensions of ethics, respect for individual liberties and ecological responsibility into digitalisation projects **Energy and ecological transitions:** S019: Bringing out innovations with a positive impact and/or "low tech" and deploying the circular economy.

DECLARATION ON HONOUR CONCERNING SUPERVISION

☒ I confirm I have contacted a Breton researcher and s/he agreed to supervise my project if this application is successful

Name of the contacted researcher(s)	Jean-Pierre Talpin
Laboratory and host institution	TEA, Inria Rennes

Please note that a letter of recommendation from the future supervisor or lab is not necessary and will not be examined during the evaluation.

PROJECT PROPOSAL (10 pages maximum excluding references)

1. Scientific project (*approx. 4 pages*)

1.1. General context

Methodology decomposes software engineering into a cycle of two main steps: prototyping and validation. The prototyping phase consists in designing a system that exhibits desired behaviors, a set of functionalities (network of

smoke detectors in a forest, humidity controllers in a datacenter, etc.). The certification phase consists in showing that the system has no undesired behavior, therefore that the prototype *safely* implements its functionality. Certification is critical and costs time and resources in proportion to the complexity of the program to verify. Certification is particularly challenging for low-cost interconnected devices in the Internet of Things (IoT), sensing inputs from and actuating the physical world. Current engineering practices for validating such devices are inadequate, and require the use of state-of-the-art research to encompass the complexity of such systems.

As Edsger W. Dijkstra famously said, "Program testing can be used to show the presence of bugs, but never to show their absence!". The current practices in IoT software engineering are mostly using test suites to reveal the presence of bugs in programs [1-3]. The most primitive form of testing takes the program that has been written and checks that, for some inputs, the outputs are matching the specification. The program is changed until all the tests pass. A more involved form of testing builds a mathematical model of the program that has been written, and tests are performed on that model. Yet, in both cases, some events of the program are not considered and left out from the certification procedure, hence letting some possibilities of failure during execution. The alternative to testing, instead, is to formally *prove* that the program meets its specification. To extend the quote of Dijkstra: "Program proving can be used to show the absence of bugs". In case of critical system, formally proving the safety of a program is the only reasonable certification procedure that can prevent harmful consequences. The IoT, within the class of reactive system [10-12], is one of such interconnected and effectful digital system that may cause harm if the effects on the environment are not properly understood.

The Internet of Things (IoT) are interconnected devices that coordinate to achieve complex measurements, and actuations [4]. To give an idea of the size of such network, it is consired that there are more "Things" (such as embedded devices, sensors, and actuators) connected on the "Internet" than human living on the planet. The recent emergence of low cost and yet powerful embedded device (ESP32 with RISC-V architecture [5]) is yet another sign that the complexity of IoT network is increasing. Those devices measure physical quantities (weather stations, agricultural plants, energy consumption [6]), and also compute decisions that trigger physical actions and change the state of the world (Industry 4.0, smart cities, smart grid [7-9]). For example, consider a collection of distributed programs that detect humidity in the air, and sense temperature. Suppose that few of those devices are positioned on a strategic place to provide an early notification that a fire starts in a forest. Those devices are autonomously powered, and therefore the use of energy is critical. The program that runs on such devices, besides its functionality (send a notification when a fire is detected), requires energy to process information. The effect of running a program on an IoT device is therefore split into two parts: the logical sequence of instructions of the program, and the effect of executing an instruction on its physical resources (e.g., consumption of energy from its battery, or moving an object in space).
The first kind of effect is currently the primary focus of most certifications procedure for the IoT. Powerful verification mechanisms can detect whether a program has logical inconsistencies [13-15]. Most of current program analysis checks that the input-output behavior of a program corresponds to its logical specification.
The second kind of effect could however preempt the logic of the program, and is therefore more critical. For instance, running out of energy or performing an irreversible actuation could cause damage to the computing device itself and to its surrounding [16,17]. Such effects are dominant in the case of reactive systems, that interact with an environment, and in the specific case of the IoT. In this case, the analysis of the logical specification of a program is no longer sufficient, as the effects of running such program within an environment must be taken into account.

Recent research has developped mathematical model to **capture runtime effects** (such as energy, time) of reactive programs. A powerful proof system, such as Coq, represents a program abstractly as a function, and provably certify its runtime properties. Currently, Inria is leading in developping verified compiler for the C programming language, and many works have used Coq to show that compilation of complex programs are correct [18-25]. However, the type of programs in the IoT are not currently easily handled by Coq, mostly due to the reactive nature (dynamic input/output at runtime), the effect of such programs on their environment and on the physics (actuation and sensing), and the interconnection of reactive programs into a network (IoT). Theoretical and practical works explore how to formalise sequential programs in the Coq proof systems [26]. Still, no direct transformations are available between widely used

language for the IoT (such as C) and the specification language Gallina used in Coq, which would increase the adoption of prove techniques and make programs safer.
There is a challenge to make the research technics practically available in proof assistant to programmers, without requiring more formal background and mathematical knowledge. The popularization of such approach in the domain of the IoT is therefore challenging and its success will increase both the safety and the resilience of reactive applications.

The aim of the CIoTA project is to **bridge the gap** between current unsafe practices for programming IoT, and powerful research methods to **prove** programs execution **safe**. To achieve this goal, we propose to extend formal methods used in proof system so that engineer can certify the safety of IoT applications, including proving safety of runtime effects on their environment.
Various domain specific languages exist for the IoT, and programmers are proficient in some of them. Instead of creating a new domain specific language, the CIoTA project is agnostic to the programming language and maps a program (e.g., program in C) to a common and expressive mathematical model (its semantic model). The mathematical model is then compatible with a proof system where a programmer can prove that the runtime effects of its program are safe. If such safety proof is accepted by the proof assistant, then a **certified** binary is generated in a targeted architecture (ARM, RISC-V, x86) [21-24] that, by construction, complies to the safety proofs. The CIoTA projects therefore decouples the logic of the programs from the effects on the architecture and the environment. Different safety conditions may hold for the same program in different environment under different architectures.

Benefits of the CIoTA approach: (1) reason formally and statically about runtime **effects** of a program on its resources and environment, and **provably safeguard** the execution of an IoT application; (2) automate and prove that the compilation, on a targeted architecture, **preserves** the properties of the program; (3) leverage research and formal methods to make current programming practices for the IoT **safer**.

1.2. Originality and Excellence

The IoT community has a wide range of programming languages each of them with different specifities [27]. Table 1 is a comparative listing of research programming languages (left of CIoTAT) and industrial programming languages (right of CIoTAT). The CIoTA's originality stems from bridging the gap between powerful technics in the research area on programming languages for reactive systems, and industrial practices on widely adopted programing languages. As shown in Table 1, the CIoTA is a programming language agnostic interface that leverages formal method analysis for reactive systems. The strength of CIoTA is that it does not define a new programming language, but translates constructs of one language (e.g., C) to another formal language (e.g., Gallina) in a transparent way.

Table 1: Benchmark of programming languages for the IoT

	Haskell [29]	Simulink [34]	Coq [18]	LUSTRE [28]	Isabelle	F*	Agda	**CIoTA**	C [32]	Java [30]	Python [31]	Rust [33]
Industrial adoption		✓						✓	✓	✓	✓	
Certification			✓		✓			✓				
Verified programming			✓		✓	✓	✓	✓				
Formal semantics	✓	✓	✓	✓	✓	✓	✓	✓				✓
IoT support		✓		✓				✓	✓	✓	✓	✓
Extraction			✓		✓			✓				

We describe in the next Sections the concepts and approaches used in CIoTA to advance the state of the art. The CIoTA

naturally splits into certification at the software level for the logic of the IoT application, and certification at the hardware for the effect at runtime of the running binary.

1.2.1 Software: formal model for runtime effects of IoT applications

Each of the industrial programming language in Table 1 has an open source framework to program IoT devices. The HomeAssistant framework [31], in Python, provides primitives to program home automations. The RIOT [32] operating system, written in C, provides system libraries for applications with low level controls. Those frameworks fail, however, to bring safey warranty as the certification procedure consists, in most of the cases, either of few tests, or no test at all. The Simulink framework [34] provides additional primitives for simulation and verification of a network of interacting devices. However, the level of certification is limited by the underlying fixed assumptions of the tool. Finally, Tock [33], written in Rust, and RIOT-fp, an extension of the RIOT operating system, are both extending some native IoT frameworks with additional safety primitives, so that, in the former case, the programmer can analyse memory allocation, and in the latter case, the programmer can use pre-verified programming primitives.

The compiler of Lustre's Scade [51] variant is certified, but Scade does not have verified programming capabilities to prove programs correct wrt. a mathematical specification, like Coq, Isabelle [36], Agda [35], liquid haskell [33] of F* [34]. On the other hand, LH, Agda and F* do not have a certified compiler, hence our choice of Coq and dX, over the alternative HOL4+CakeML [36], of equivalent features yet lesser support in the environment of our projects within Inria.

Our approach is in the line with RIOT-fp, but different as we will embed in the Coq proof assistant a fragment of the programmer's language. The Coq proof assistant has a powerful underlying calculus that enables reasoning and proving properties of program's behavior. The language used in Coq is Gallina, that syntactically and semantically differs from industrial languages which leads to a lower acceptance and usability of the tool in industrial purposes. The **first novelty** of the CIoTA project will be to identify, in Gallina, structures of reactive programs implemented in the IoT community. Similarly to what has already been done for sequential state based programs [35], we establish an equivalence between mathematical programs in Gallina and operational programs in the target language (such as C or another language), which has the consequence to decrease the entry gap for programmer engineers as there is no need to learn the Gallina language to start using our framework.

The CIoTA project **extends the current Coq proof system to support reactive systems analysis**. Recent research has shown that reactive systems are better described by functions with contexts and effects [16-17]. However, Coq does not yet provide sufficient tactics to easily prove equivalences and safety of programs. Formally, reactive programs are composition of reactive functions with effects. We base our theory on mathematical objects that are called monads [16,31,32] and comonads [17], which adequatly extend the notion of function to capture effects of the architecture and of the environment on the computation. Using a proof assistant to reason about a widely used programming language (such as C) has been done successfully [21-24, 26], but cannot yet support reactive programming, due to both theoretical and practical challenges. We extend dx [26] so that engineering programmers have access to this framework to prove safety properties of reactive programs.

1.2.2 Hardware: certification of IoT applications

The certification of IoT applications is a trade of between adding static constraints to the programming language (as Rust does for C programs by preventing some unsafe memory manipulation), or validating the execution dynamically with test and verification on the compiled program. The advantage of the former approach is that the program is oblivious to the target architecture, while the advantage of the second approach is that the certification can be much precise as the program has already been compile to execute on a fixed architecture.
CIoTA abstracts the target architecture, while providing static means to check that the runtime effecte of a program are safe. We will build on top of the Compcert toolchain that certified the correctness of compilation of a C program to a parametric architecture (RISC-V, ARM, x86). More precisely, the CIoTA approach provides **novel mathematical abstraction of runtime effects of a program on the architecture** (energy and time consumption) and its environment (actuation and sensing), such that a C program can be proven safe within such parametric context. Once the C program

is safe, the Compcert toolchain generates a certified C binary to run on the targetted architecture. The challenge is to reason about program equivalence not only with respect to the input/ouput response (extentional equality) but also with respect to their effect at runtim (intentional equality). For that end, the CIoTA defines an abstract interface to model a processor architecture and an environment as a measure of the effects of a program instruction on its environment.
As an application, we will formalize in Coq the implementation in C of a node in the MQ Telemetric Transport protocol [37], one of the most used protocol for message exchange between IoT nodes. We will prove runtime quality of service properties (energy or time consumption), and extract an executable on a targetted architecture (ESP32 or SMT32) that preserves the proven properties.

1.3 Research Methodology

The CIoTA projects defines a formal framework to prove reliability and safety of runtime effects of IoT applications. Figure 1 shows the different components and their interdependences, that we explain hereafter in three steps: the abstraction step (I. II. and III.), the certification step (IV.), and the extraction step (V.).
The proposed research naturally decomposes into both theoretical and practical components. Our research methodology is to interleave both developments, by using practical tools to implement our theoretical results, prove their soundness, and demonstrate their applicability. As a result, the theory will be restricted to one that can be implemented and constructively verified, and the use of theorem prover and process extraction will confirm and demonstrate the validity and applicability of the theory.

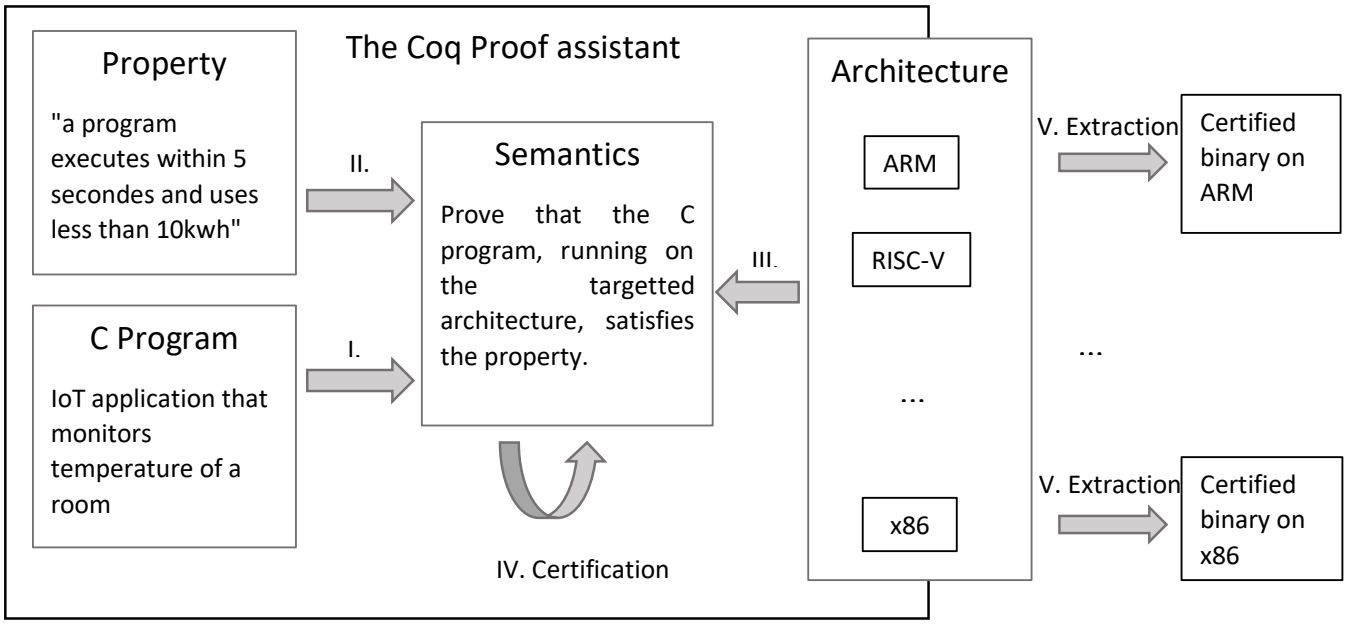


Figure 1: Architecture of the CIoTA project.

1.3.1 Abstraction

The CIoTA approach abstracts a widely used programming language for IoT (such as C) into a shared formal semantic model. Step I represents common structures widely used for programming IoT applications into mathematical objects. Currently, the same approach is done for some sequential programs [26] and programmers can specify, using the same syntax, their program in a proof assistant. CIoTA extends the formal semantics by using effectful notion of computation [16,17], such that the resulting description formally captures the distributed and real-time nature of IoT systems, and provide means to reason about the behavior of a single node within a network of interacting nodes and an environment.

Step II requires new characterisation of programs by not only referring to their extensional equality, i.e., two programs are the same if they have the same input and output relations, but also to model their intentional equality [16,38], i.e., two programs are the same if the effects of their execution on their environment are the same. By changing the kind of equality to include runtime considerations (energy usage, time of execution), we can certify runtime properties before deployment and search for more efficient specifications that, e.g., save energy or computation time. Our work will extend current theoretical results about intentional semantics of functional programming languages, by considering other computational model that include time and energy consumption of programs. We will implement our model in the Coq theorem prover, and produce a library for automatically verifying proofs that a program has some runtime characteristics [17,39,40].
The research will develop practical tools such as a static semantics for reactive process, and will develop a (typing) relation between the formalized semantic and desirable runtime properties. The theory will limit to the use of computable constructions in order to render possible the practical demonstration of the theoretical results.

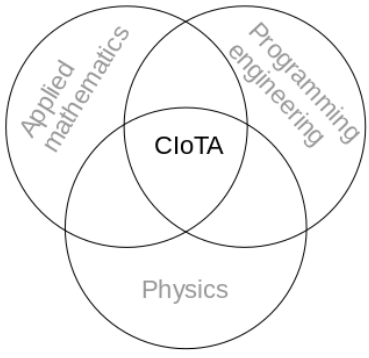
Step III is a necessary step to model effects of a program on a targetted architecture. A program is a high level description of an assembly program using instructions of the processor architecture. As a result, a program translates to different assembly codes when compiled on different processors. The runtime effects of a program (time and energy consumption) are dependent on the processor that executes the program, which may vary under different applications. Instead of focusing on a specific architecture (RISC-V, ARM, x86, …), we abstract an architecture as a metric applied on an instruction. For a fixed architecture, the metric gives a concrete value for the effect of a program, and the use of the Coq proof assistant makes reasoning about resource consumptions possible.

1.3.2 Certification
The proof that a program satisfies runtime properties provide a certificate that, under the architecture and envrionmental specification, the runtime behavior of the program is safe. This step will extend current theoretical work on compositional verification (session types [45], process calculus [46]) to include runtime properties such as energy consumption.

Step IV transforms a set of interacting program specification such that it eventually satisfies the desired property. For instance, a set of IoT devices may be deemed unsafe due to the latency of one of its parts [41]. The substitution of a faster program that has the same input/output behaviour may allow to prove the resulting IoT network safe. The goal of this step is to define general mechanisms that are not restricted to some specific processor architecture.

1.3.2 Extraction
The generation of an executable that preserves the properties proved on the specification is a critical and essential part of the project.
Step V generates a set of executable programs that can be deployed, for instance, on the IoT network. This step will extend existing tools to support code generation for reactive processes [18, 26].

1.4 Inclusion of international, interdisciplinary and/or intersectoral aspects



This project combines three disciplines: programming engineering, applied mathematics for computer science, and physics.

Research wise, the teams with which I will collaborate in France are: the TEA team at Inria center of the university of Rennes with Jean-Pierre Talpin, and the 2XS team of CNRS Crystall in Lille (David Nowak). Both team specialises in one of the topics. The first in programming engineering and low level compilation correctness toolchains, and the second in mathematics applied to computer science with advanced research in reactive systems. My research background focuses on the latter branch, on the interaction between digital and physical systems [10, 41-44].

Industrial partners in Brittany are also involved in the success of the project, as Mitsubishi is a leading manufacturer of reactive systems. Jean-Pierre Talpin is a coordinator of the MERCE Mitsubishi collaboration with Inria center of the university of Rennes, and bi-monthly meetings with the research and development team at Mitsubishi will influence the practicality and usability of our approach into an industrial context.
Currently, a major focus is given to equip industries with IoT devices and automate production chains (such as Industry 4.0 [7]). Our project will provide additional safety to such automation systems, and provide safeguard measures on the energy consumptions, and critical time of executions. Overall, the project extends current computational model with measures that are not purely functional (input/output relation), but also consequential of the running program. Several other measures can therefore be considered, depending on the discipline and the sector of application.

The project will also reinforce a close collaboration with a group at CWI in Amsterdam, where the applicant did his PhD thesis. The group of Computer Security, with Farhad Arbab and Hans-Dieter Hiep, studies both theoretical and practical approaches to certify safety of reactive systems. The collaboration consists of two visits per year, for discussing results, and of monthly online meeting for presentation of research.

2. Impact of the project *(approx. 2 pages)*

2.1 Impact

The CIoTA project bridges the gap between powerful research techniques for proof based programming, and engineering programming for the IoT. Due to their highly reactive nature, and unpredictability of their surrounding environment, the engineering of IoT systems require state of the art of research in monitoring runtime effects and proving safety of program execution. We list a series of foreseen impacts of the CIoTA project.

Scientific impact: The type system that we will develop enables reasoning about runtime properties. While this is already studied for some effectful functions [16,17], there is no uniform framework to discuss reactive, energy aware processes as effectful functions. We will contribute to the state of the art by extending the set of effectful functions that can be compiled and reliably executed.
The fellowship will publish the result to CAV2024, and POPL2025, two leading conferences in the field. The first set of results to CAV2024 will consists of the specification in a proof assistant of effectful reactive programs, and a library for leveraging C code to our framework. The second set of results consists of extraction methods to certify runtime properties of IoT applications, which requires abastraction of the environment and architecture on which the application will be running. The research is in line with current theoretical questions to reconsider current model of computation to integrate effects of physics on programs [16].

Environmental impact: The theory that results from the project will bring more understanding on the effects of a program on its environment. This is in line with the general sustainable approach to technology [48,49], and our project may result in appropriate carbon footprint measure for softwares, with an accurate energy label for IoT programs. Ecological research on computation would therefore benefit from advanced techniques in computer science to capture, compositionally, the energy spent at every step of a program computation.

Societal impact: Our societies are nowadays deeply dependent on technology. From the food industry, to the healthcare systems, reliable communications and reliable automations are critical to sustain the development and stength of our society. The CIoTA contributes to reinforce the digital security of our system by researching state of the art open problems and proposing practical tools to improve the safety and reliability of future digital systems.

Economic impact: reduce the energy cost of large scale programs (linky counter) if there is a reliable mechanism to compare runtime properties of program (via intentional equality). Our framework will enable new methods to compare runtime effects of programs, and to compare their efficiency on their effects on the environment.

Regional impact: The region of Bretagne has a very diverse landscape of possible parteners for research collaboration. The TEA team already actively collaborates with Mitsubishi, MERCE. The ecological and energy sectors also benefits from the technological advances, such as the energy grids and Linki counter. Such counters are built using STM32 components, and our approach to certify safety of runtime effects may be of interest for possible future collaborations with STM electronics and Enedis. Moreover, the Direction General de l'Armement has its main digital branch at Bruz, in Brittany. As witnessed by past projects [50], the security of the IoT system is of crucial importance for the defense of critical infrastructures. Our project, in its latter stage, can find applications to certify the security of IoT protocols.

The CIoTA project addresses the strategic objective SO7 by leveraging state of the art research results to programming engineering. The tools developped in the CIoTA project opens the construction of new IoT application with the certification of reliability and safety that is today not possible. Industrial applications in the domain of networks and connected object will benefit from the research. The strategic objective SO9 is deeply related, as the Inria team at Rennes is working closely with MERCE (Mitsubishi Electric R&D Centre Europe), based in Renens in Brittany. MERCE will therefore act as an industrial advisor in the CIoTA project, which the results will contribute towards building leadership in the European cybersecurity. The primary focus of the CIoTA project is to model runtime effects in IoT application. This objective contributes to SO17 by providing mathematical model to give tools to engineers to define and control their ecological responsibility in digitalisation project. As a result, the CIoTA project also contributes to SO19 by leveraging formal tools to develop IoT application with positive impact in terms of time, energy consumption, and effects on their environment.

2.2 Career Development

The ability for programs to alter their environment is a new feature that requires rethinking of the practices of programming to increase safety of such systems, and theoretical approaches to faithfully define the notion of what such program computes.
The project that I submit is in line with a larger research agenda, which is to develop theoretical and practical methods to program reliable reactive systems. The immediate theoretical and practical goals are to unify the effectful nature of reactive systems with the fundamental models of *computations as a function*. By developing a toolchain for certified IoT programs, this project helps to set the first steps toward a more general framework for certified cyber-physical applications. As described earlier, the project is both at the front of research activities (call for such results in the highest conference in the field), and at the front of industrial preoccupations (as witnessed by the collaboration with MERCE Mitsubishi).
On a personal aspect, the CIoTA project will be the first step towards a long term objective to pass the CRCN concours of 2024 and continue the research as a permanent team member of the Inria center of the university of Rennes.

The **Bienvenue post-doc fellowship** is a unique opportunity to perform advanced and novel research in the **TEA team** of the Inria center of the university of Rennes**,** leader in the research of reactive systems and formal proofs. My background focuses on formal model for the design of reactive systems, and more specifically on algebraic semantics for specification of interaction among digital and physical components. As a consequence, my research experience gives me a strong background on the CIoTA research topic, which allows me to immediately start experimenting with ideas on the first and future steps. My expertise on semantics approaches to model reactive systems gives me necessary skills and knowledge to apply to certification of reactive systems, and obtain significant scientific achievement.
I will benefit from the theoretical and practical knowledge of Jean-Pierre Talpin in the TEA team, and David Nowak in the Crystal team, when working on the Coq theorem provers, and its extensions.

I did my studies at IMT Atlantiques, and I am eager to start partnership with the school to explain to students how research can be chosen as a meaningful career direction. The region is also very dynamic and offers multiple opportunities to grow research organically with industrial (e.g., smt-semielectronics, orange, DGA) or scholar partners (e.g., labfab network, Irisa).

Finally, I am still in close contact with the research group at CWI Amsterdam, and I will continue to strengthen the link between CWI and Inria, as CWI focuses more on theoretical algebraic models, and the TEA team leverages formal methods to . Inria has an associate team campaign, which allows you to set up a collaboration for 2 years and receive small funding for some travels [47].

2.3 Transfer of knowledge and training

Host team: The TEA team at Inria center of the univeristy of Rennes will host the CIoTA project. The team has participated in several European projects, has strong international collaborations (Chinese Academy of Science), and strong industrial collaborations (Mitsubishi). The team has expertise in leveraging formal methods to engineering programming, with recent research publications in top conferences [22, 23] on end-to-end verification of operating system services in the theorem prover Coq.

In particular, Jean-Pierre Talpin (TEA) chairs the collaborative framework program between Inria and Mitsubishi European Research Centre in Rennes (MERCE). MERCE will actively participate in CIoTA, first by the organization of monthly meetings, as this project relates in both the application domain (IoT), programming base (C) of the partner, and a forthcoming CIFRE Ph.D. grant.

New skills: I want to learn to use the Coq theorem prover, and extend its current capabilities to reason about reactive programs. Inria is pioneer in this domain, and currently leading the development of the tool. The TEA team of the Inria center and 2XS team at CNRS Crystal are at the forefront of the research [18] to use and extend the Coq proof assistant for end-to-end verification of exokernel services and reactive systems [20,26]. They intend to submit an Inria Challenge project on this combined research effort. Our projects are an opportunity to add a productive and fruitful branch to the Coq proof assistant.

Transfert of knowledge: During my PhD, I studied formal models and practical applications to design correct by construction communication protocols. The advanced algebraic concepts (co-inductive data types, formal models for cyber-physical coordination) developped during my PhD will be valuable not only for the CIoTA project, but also for other projects oriented towards proving safety of interactive reactive system in the TEA team. The CIoTA project will allow me to apply powerful theoretical results to practical settings within the Coq proof assistant.
The institute has partnership with the IMT Atlantiques engineering school, from which I graduated. I therefore expect to collaborate on research topics with engineering students, and teach advanced and applied research topics. The knowledge developed in the research will be transferred to both scholar and industrials partners.
I contributed to several first authored conferences and journal papers, and have developed a taste for research that pushes me to undertake and discuss challenging ideas.

2.4 Communication, Dissemination and exploitation of results
Dissemination: the theoretical results will lead to two high quality papers, that will be peer reviewed and presented at top conferences on programming and verification, e.g., POPL 2024, OOPSLA 2024 and CAV 2025. Simultaneously, the project leads to artefact that will be made open source, and provided as a Coq library for certification of IoT applications. The community interested in such topic is both academic (topics of interest for POPL and CAV conferences) and industrial (topic of interest for MERCE Mitsubishi). The library will be presented to the research development community and will be motivated by the certification of runtime effects in the MQTT protocol.

Communication: The project also focuses on leveraging formal methods to a wider audience. The library will be accompanied by a tutorial and a detailed procedure to start using our tool. The project will lead to a presentation to the general audience of the Inria center of the university of Rennes. Finally, the researcher involved in the proposal will stay in contact with Dutch research institutes, as common interests have been created between the researchers at CWI and the project applicant. In that sense, close collaboration with the CWI for dissemination of research results will be ensured by annual visits and joint research works.

Being part of the bienvenue scholarship is joining a group of 25 postdocs, and a cohort of 50 alumni. Yearly events will be an opportunity to talk about my research to other fellow in a different way.

3. Implementation of the project *(approx. 2 pages)*

3.1 Work programme, resources and risks

3.1.2 Work packages and risks

The work consists of three main work packages, that will last over the duration of the project:

WP1: Formal models for abstraction of runtime effects of the IoT. Bridge the gap between an imperative style programming for IoT applications (such as C) and a powerful type system for safety (in Coq):

- Task 1.1: Definition, within the Coq proof assistant, of a fragment of the C programming language for IoT applications. This step takes extends existing approach [26] to reactive programs, so a part will be dedicated to understanding the existing tool and theoretical results around dx.
- Task 1.2: The formal mathematical model of reactive program will be defined in Coq, within the same period as the practical embedding of C programs in Coq. Then, the framework will provide a formal semantics for C programs as functions with effects in the Coq proof assistant.
- Task 1.3: The effects of reactive programs defined in Tasks 1.1/1.2 will be parametric to the architecture and the environment in which the program will run. The abstraction provides formal primitive to define abstractions for different architectures, and physical environments that grounds the effects of a programs.

Deliverables: library for Coq with a support of an extended syntax of the C programming language to model IoT systems. Presentation of the theoretical results and demonstration of the tool at POPL2024, largest conference in the field.

WP2: Certification of runtime behavior. Certification is issued after that the program is proven safe with respect to some properties on its effect. This work package builds on the formal model established in WP1, defines a logic to specify properties of effects at runtime, and extends proof methods in the Coq proof assistant to more easily deal with reactive systems with possibly unbounded behavior.

- Task 2.1: Develop a logic for specifying safe and unsafe effects at runtime. This logic will be used in the Coq proof system, together with the formal model of WP1 to prove that a program is safe.
- Task 2.2: Extend the kind of equivalence on programs in Coq. Two programs are equivalent if, besides having the input/output response (extensional equality), their effects are the same (intentional equality). We extend a library to reason about intentional equality for programs with effects.
- Task 2.3: Proof tactics in Coq depends on the structure of the data (or function) that is studied. WP1 defines new structure for reactive systems, we will extend the proof tactics in Coq to facilitate proving results on reactive programs.

Deliverables: library in Coq for proving runtime safety of effectful program in terms of time and energy consumption given a specific architecture and environment. Presentation of theoretical results and demonstration of the tool at CAV2025, largest conference in the field.

WP3: Extraction of executables. The Coq toolchain [18] currently does not easily support code extraction for reactive systems, as such systems are interacting with their environment, and not necessarily terminating. The extraction of executable will therefore need to extend the Coq toolchain to generate valid and executable specification for interactive processes.

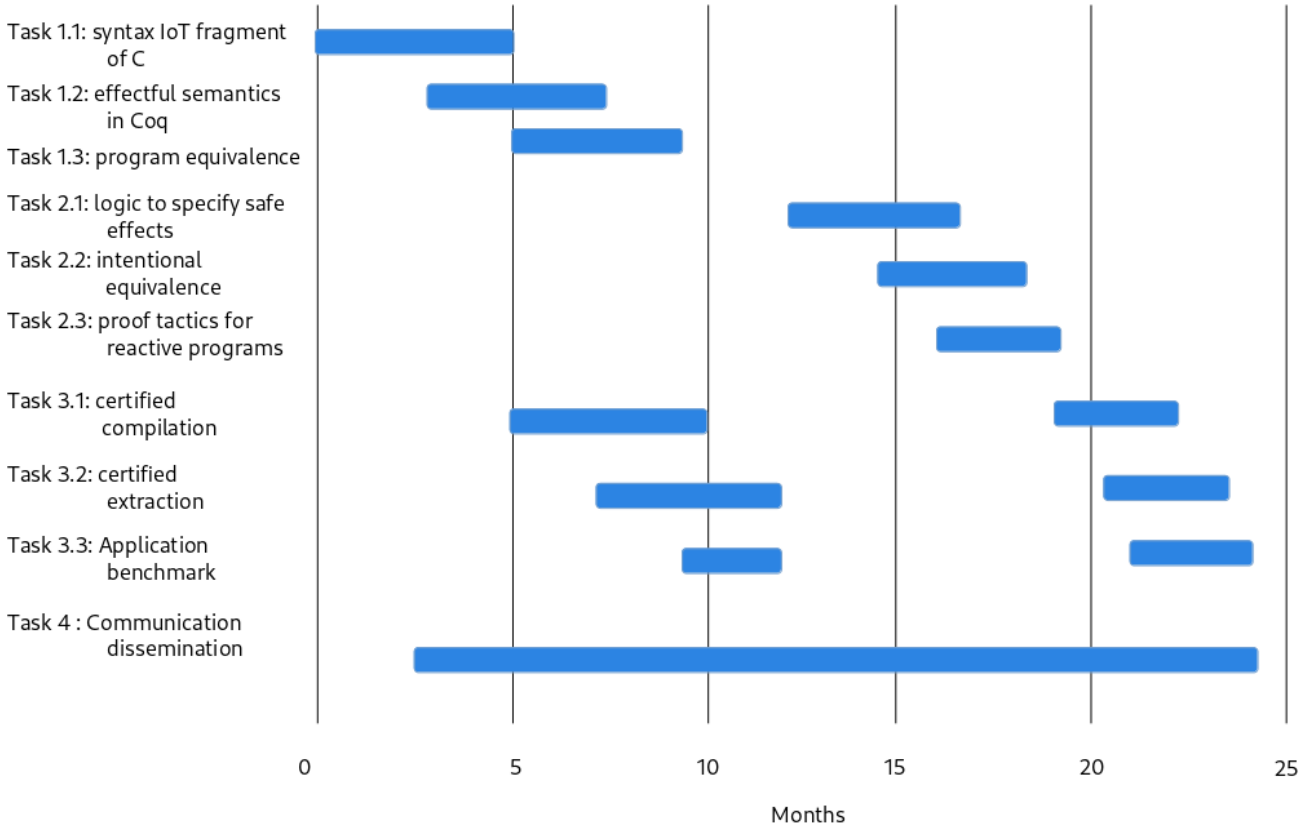
- Task 3.1: Once Task 1.1 and 1.2 are completed, we will test our approach by generating executables on a toy microcontroller (STM32) for a temperature sensor reading and temperature contol. The extraction makes use of a verified compiler (Compcert), that will have to be intergrated within our framework.
- Task 3.2: Once Task 1.3 is completed, we will test our approach by generating a distributed implementation on a set of interacting IoT devices (SMT32 with communication). The protocol of interaction between the devices will be formalised in Coq, and safety measures proven on the runtime effects. Experiments will be run to quantify the accuracy of the generated code.

- Task 3.3: Once Task 2.2. is completed, a program can be proven more efficient in Coq, and therefore its effects at runtime less costly. We will benchmark such results by monitoring effects of proven optimized application vs non-optimized application.

Deliverables: a documentation and an open access to use the tool that extracts, from a C reactive program and a proof of safety properties in Coq, a binary on a targetted architecture. The results will be submitted to POPL2025.

WP4: Dissemination and communication.

- Task 4: Project management as well as communication and dissemination will cover the full project duration. Communication of the results will be done at the seminar of Inria (sci-rennes), and dissemination of results will make use of seminar within the TEA and Crystal Inria team, seminar (acg) at CWI, and bimensual meeting with MERCE Mitsubishi.



3.1.2 Resources:

Resources are human resources and digital facilities. The contract in the host team at Inria will provide necessary resources for the success of the project (such as an office and personal computer). The CIoTA project

3.2 Choice of the Host Institution and the Host Laboratory

The TEA team (Time Event Architecture) of the INRIA center at the univerisity of Rennes hosts the CIoTA project. The group has a strong background in using proof assistants for reasoning about real world program safety [Shenghao?], providing the best environment for this project's development. At the TEA group, the project will be supervised by professor Jean-Pierre Talpin. Additionally, regular meeting will be scheduled with David Nowak of the Cristal team at Lille for discussing formal models for reactive systems.

References

References should be listed here and do not count towards the page limitation.

[1] Moez Krichen and Stavros Tripakis. “Conformance testing for real-time systems”. In: Formal Methods Syst. Des. 34.3 (2009), pp. 238–304. doi: 10.1007/s10703-009-0065-1. url: https://doi.org/10.1007/s10703-009-0065-1.

[2] Alireza Souri and Monire Norouzi. “A State-of-the-Art Survey on Formal Verification of the Internet of Things Applications”. In: J. Serv. Sci. Res. 11.1 (2019), pp. 47–67. doi: 10.1007/s12927-019-0003-8. url: https://doi.org/10.1007/s12927-019-0003-8.

[3] Souri A, Rahmani, AM, & Jafari Navimipour N (2018b), “Formal verification approaches in the web service composition: A comprehensive analysis of the current challenges for future research”. International Journal of Communication Systems 31(17): e3808

[4a] Norbert Wiener. Cybernetics: or Control and Communication in the Animal and the Machine. 2nd ed. Cambridge, MA: MIT Press, 1948.

[4b] Yen-Kuang Chen. “Challenges and opportunities of internet of things”. In:17th Asia and South Pacific Design Automation Conference. 2012, pp. 383–388. doi: 10.1109/ASPDAC.2012.6164978.

[5] Espressif. ESP32-C3. url: https://www.espressif.com/en/products/socs/esp32-c3 (visited on 2023).

[6] Aditya Gaur et al. “Smart City Architecture and its Applications Based on IoT”. In: Procedia Computer Science 52 (2015). The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015), pp. 1089–1094. issn: 1877-0509. doi: https://doi.org/10.1016/j.procs.2015.05.122. url: https://www.sciencedirect.com/science/article/pii/S1877050915009229.

[7] Harald Schöning. “Industry 4.0”. In: it Inf. Technol. 60.3 (2018), pp. 121–123. doi: 10.1515/itit-2018-0015.

[8] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. “A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications”. In: IEEE Access 7 (2019), pp. 167653–167671. doi:10.1109/ACCESS.2019.2953499.

[9] Luigi Atzori, Antonio Iera, and Giacomo Morabito. “The Internet of Things: A survey”. In: Comput. Networks 54.15 (2010), pp. 2787–2805. doi: 10.1016/j.comnet.2010.05.010.

[10] Benjamin Lion, Farhad Arbab, and Carolyn L. Talcott. “A semantic model for interacting cyber-physical systems”. In: J. Log. Algebraic Methods Program. 129 (2022), p. 100807. doi: 10.1016/j.jlamp.2022.100807. url: https://doi.org/10.1016/j.jlamp.2022.100807.

[11] Edward A. Lee. “Cyber Physical Systems: Design Challenges”. In: 11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2008), 5-7 May 2008, Orlando, Florida, USA. IEEE Computer Society, 2008, pp. 363–369. doi: 10.1109/ISORC.2008.25.

[12] Christos G. Cassandras and Stéphane Lafortune. Introduction to Discrete Event Systems, Second Edition. Springer, 2008. isbn: 978-0-387-33332-8. doi: 10.1007/978-0-387-68612-7.

[13] Edmund M. Clarke, Orna Grumberg, Daniel Kroening, Doron A. Peled, Helmut Veith: “Model checking”, 2nd Edition. MIT Press 2018, ISBN 978-0-262-03883-6

[14] Fokkink, W. (2013). “Introduction to process algebra”. springer science & Business Media.

[15] Watterson, C., & Heffernan, D. (2007). Runtime verification and monitoring of embedded systems. *IET software*, *1*(5), 172-179.

[16] Moggi, E. (1991). Notions of computation and monads. *Information and computation*, *93*(1), 55-92.

[17] Uustalu, T., & Vene, V. (2008). Comonadic notions of computation. *Electronic Notes in Theoretical Computer Science*, *203*(5), 263-284.

[18] Leroy, X., Blazy, S., Kästner, D., Schommer, B., Pister, M., & Ferdinand, C. (2016, January). CompCert-a formally verified optimizing compiler. In *ERTS 2016: Embedded Real Time Software and Systems, 8th European Congress*.

[19] Daniel Ricketts. "Verification of Sampled-Data Systems using Coq". PhD thesis. University of California, San Diego, USA, 2017. url: http://www.escholarship.org/uc/item/5n1899s2.

[20] "A Formal Correctness Proof for an EDF Scheduler Implementation". Florian Vanhems, Vlad Rusu, David Nowak, Gilles Grimaud. 28th Real-Time and Embedded Technology and Applications Symposium. IEEE, 2022.

[21] "HAVM: verified programming of a hybrid eBPF virtual machine with JIT acceleration". Shenghao Yuan, Frédéric Besson, Jean-Pierre Talpin, Koen Zandberg, Emmanuel Baccelli. Submitted to the International Conference on Computer-Aided Verification, 2023.

[22] "Femto-Containers Runtime: Ultra-Lightweight Virtualization andf Fault Isolation For Small Software Functions on Low-Power IoT Microcontrollers". Koen Zandberg, Emmanuel Baccelli, Shenghao Yuan, Frédéric Besson, Jean-Pierre Talpin. 23rd ACM/IFIP International Middleware Conference. Usenix, 2022. Artifact available on Github, Femto-Containers.

[23] "End-to-end Mechanized Proof of an eBPF Virtual Machine for Micro-controllers". Shenghao Yuan, Frédéric Besson, Jean-Pierre Talpin, Samuel Hym, Koen Zandberg, Emmanuel Baccelli. International Conference on Computer-Aided Verification, 2022. Artifact available on Gitlab, CertFC.

[24] "Verified Functional Programming of an IoT operating system's bootloader". Shenghao Yuan, Jean-Pierre Talpin. International Conference on Formal Methods and Models for System Design. ACM, 2021.

[25] "Verified Functional Programming of an Abstract Interpreter". Lucas Franceschino, David Pichardie, Jean-Pierre Talpin. Static Analysis Symposium. ACM, 2021. With artifact "AbIntFStar" on Gitlab.

[26] Samuel Hym, dx, https://gitlab.univ-lille.fr/samuel.hym/dx

[27] Corno, Fulvio and De Russis, Luigi and S{\'a}enz, Juan Pablo, "How is open source software development different in popular IoT projects?" in IEEE Access, 2020

[28] Paul Caspi, Daniel Pilaud, Nicolas Halbwachs, John Plaice: "Lustre: A Declarative Language for Programming Synchronous Systems." POPL 1987: 178-188

[29] Haskell https://wiki.haskell.org/Functional_Reactive_Programming

[30] Java Thingsboard https://github.com/thingsboard/thingsboard

[31] Python HomeAssistant https://github.com/home-assistant/core

[32] C RIOT https://github.com/RIOT-OS/RIOT

[33] Rust https://github.com/tock/tock

[30] Simulink https://fr.mathworks.com/solutions/internet-of-things.html

[31]Tim Lukas Diezel and Sergey Goncharov. “Towards Constructive Hybrid Semantics”. In: 5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020). Ed. by Zena M. Ariola. Vol. 167. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 24:1–
24:19. isbn: 978-3-95977-155-9. doi: 10 . 4230 / LIPIcs . FSCD . 2020 . 24. url: https://drops.dagstuhl.de/opus/volltexte/2020/12346.

[32]Sergey Goncharov, Renato Neves, and José Proença. “Implementing Hybrid Semantics: From Functional to Imperative”. In: CoRR abs/2009.1432 (2020). arXiv: 2009.14322. url: https://arxiv.org/abs/2009.14322.

[33] Liquid Haskell https://ucsd-progsys.github.io/liquidhaskell/

[34] F* https://www.fstar-lang.org/

[35] Agda https://wiki.portal.chalmers.se/agda/pmwiki.php

[36] CakeML https://cakeml.org/

[37] MQ Telemetry Transport (MQTT): https://mqtt.org/

[38] Brookes, S., & Geva, S. (1991). “Computational comonads and intensional semantics”. Carnegie-Mellon University. Department of Computer Science.

[39] Zhao, T., Berger, A., & Li, Y. (2020, November). Asynchronous monad for reactive IoT programming. In *Proceedings of the 7th ACM SIGPLAN International Workshop on Reactive and Event-Based Languages and Systems* (pp. 25-37).

[40] Power, J., & Watanabe, H. (2002). Combining a monad and a comonad. Theoretical Computer Science, 280(1-2), 137-162.

[41] Benjamin Lion, Farhad Arbab, Carolyn L. Talcott: A formal framework for distributed cyber-physical systems. J. Log. Algebraic Methods Program. 128: 100795 (2022)

[42] Benjamin Lion, Farhad Arbab, Carolyn L. Talcott: Runtime Composition Of Systems of Interacting Cyber-Physical Components. In proceeding WADT (2022)

[43] Benjamin Lion, Farhad Arbab, Carolyn L. Talcott: A Rewriting Framework for Cyber-Physical Systems. In proceeding Isola (2022)

[44] Benjamin Lion, Cyber-physical agent framework in Maude, Zenodo, 10.5281/zenodo.6592275 (2022)

[45] Hüttel, H., Lanese, I., Vasconcelos, V. T., Caires, L., Carbone, M., Deniélou, P. M., ... & Zavattaro, G. (2016). “Foundations of session types and behavioural contracts”. *ACM Computing Surveys (CSUR)*, *49*(1), 1-36.

[46] Lanese, I., Bedogni, L., & Di Felice, M. (2013, March). Internet of things: a process calculus approach. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (pp. 1339-1346).

[47] https://intranet.inria.fr/Vie-scientifique/International/Relations-internationales/Programmes-bilateraux.

[48] Pereira, F.; Correia, R.; Pinho, P.; Lopes, S.I.; Carvalho, N.B. Challenges in Resource-Constrained IoT Devices: Energy and Communication as Critical Success Factors for Future IoT Deployment. *Sensors* **2020**, *20*, 6420. https://doi.org/10.3390/s20226420

[49] Daler Rakhmatov, Sarma Vrudhula, and Deborah A. Wallach. 2002. Battery lifetime prediction for energy-aware computing. In Proceedings of the 2002 international symposium on Low power electronics and design (ISLPED '02). Association for Computing Machinery, New York, NY, USA, 154–159. https://doi.org/10.1145/566408.566449

[50] https://iot.bzh

[51] Bourke, T., Brun, L., Dagand, P. É., Leroy, X., Pouzet, M., & Rieg, L. (2017, June). A formally verified compiler for Lustre. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation* (pp. 586-601).