Energy-Aware Model-Driven Development of a Wearable Healthcare Device

José Antonio Esparza Isasa[⋆], Peter Gorm Larsen[⋆] and Finn Overgaard Hansen[†]

Department of Engineering[⋆], Aarhus School of Engineering[†].
Aarhus University. Finlandsgade 22, Aarhus N 8200 Denmark
{jaei,pgl}@eng.au.dk[⋆], foh@mail.tdcadsl.dk[†]

Abstract. The healthcare domain is experiencing an expansion of wearable embedded devices. These devices are typically battery powered and expected to deliver a safe and reliable service to the patient regardless of its power reserves. Being energy efficient brings an additional level of complexity to the development of these solutions. In this paper we propose the application of a well-founded model-driven energy-aware approach to tackle the energy consumption in such solutions addressing all their critical subsystems: control software, communication and mechanical components. The approach enables exploration of the design space, reduces prototyping costs and helps in build confidence in the proposed solution. We demonstrate this approach in a case study focused on the development of an intelligent compression stocking to treat leg-venous insufficiency. We also discuss how this approach has benefited the development of the actual device.

Keywords: Energy consumption, energy-aware design, wearable devices, pervasive healthcare, cyber-physical systems

1 Introduction

Wearable embedded systems are a concrete kind of Cyber-Physical Systems (CPSs) that are experiencing a widespread application in the healthcare domain. Given their nature these devices are mainly battery powered and present a challenge from the *sustainability* point of view. Additionally the medical domain adds *safety* and *security* constraints during their operation. These properties are know as the S3 properties of CPSs [2], they are common to many other systems but specially relevant in medical CPSs. Being energy efficient as well as satisfying the functional requirements is a complex challenge when designing these kinds of devices. The application of abstract modelling at the system level to cope with this complexity has been proposed [8]. These models enable the analysis of system properties and they can be progressively transformed into concrete system realizations. This approach is know as model-driven development.

We propose a new way to conduct energy-aware model-driven development of complex embedded solutions, enabling the analysis of energy consumption from different perspectives: communication, computation and electro-mechanical [13]. In this paper we present how we have applied this approach to the design of a medical-grade intelligent compression stocking in order to study energy consumption and steer the development of the device. The application of this approach has resulted in the redesign of the mechanical configuration and control software leading to energy savings of up to 33%. Additionally we have explored several system architectures involving software and hardware at the modelling level, being able to choose the optimal one for the current implementation.

The remainder of this paper is structured as follows: Section 2 gives an overview of the modelling approach applied to the design of a concrete healthcare device, which is presented in section 3. Section 4 ellaborates on the results achieved so far. Sections 5 and 6 present related and future work respectively. Finally, section 7 concludes this paper.

2 Energy-aware model-driven design

CPSs are complex systems composed of electro-mechanical, software, electronic and communication components. We propose the application of a heterogeneous modelling approach, that incorporates the notion of consumed energy, to take all these elements into consideration in a single design effort [13]. This section introduces the modelling technologies used and outlines this approach.

2.1 Modelling technologies used

Given the heterogeneous composition of a CPS we have used two different modelling paradigms that can be used cooperatively to develop systems. These paradigms are used to represent the system and to some extent the interaction with the environment through Discrete Event and Continuous Time models. The tools supporting them are:

Overture[1]: is a modelling environment that supports the creation and simulation of VDM Real-Time models [22, 16]. This formal notation is suitable for the representation of Discrete Event (DE) control logic. It supports the deployment of different model parts on different simulated CPUs that are connected by buses. This makes it possible to simulate some of the distributed properties of a CPS.

20-Sim[2]: is a modelling environment that supports the creation of Continuous Time (CT) models based on differential equations. Additionally it incorporates higher level abstractions such as bond graphs and libraries of ready made components for different areas (pneumatics, electronics and hydraulics among others).

DESTECS/Crescendo[3]: is a framework that connects the Overture real time interpreter with the simulation core of 20-Sim [3, 6]. This provides a common notion of time for both simulators and makes it possible to run them in parallel with interactions (co-simulation) [7].

[1] Overture project official website: `www.overturetool.org`
[2] 20-sim official website `www.20sim.com`
[3] This framework is currently known as Crescendo but it was called DESTECS in previous published work (see `www.crescendotool.org`)

These tools are general purpose modelling tools within their respective domains, that can be used to study different system functionalities not necessarily involving energy consumption. For additional details on the tools and how they can be applied refer to [6] and continuations of this work in the INTO-CPS[4] project [4, 5, 18, 19].

2.2 Model-based approach to energy consumption

Our design approach [13] takes into consideration energy consumption in systems from three different angles: mechanical, software and communication.

Mechanical modelling. Mechanical components incorporated in a CPS are typically the most energy consuming ones. These mechanical and electro-mechanical components are controlled from software and typically feature different kinds of sensors and actuators. Ways to decrease their energy consumption include the design of new mechanical architectures or the optimization of control algorithms. This implies that one needs to consider both electro-mechanics and control logic together in order to find an energy efficient system configuration. An example of an electro-mechanical subsystem in a wearable healthcare device could be the components responsible for a blood pressure measurement in a pervasive blood pressure measurement monitor (i.e., inflatable cuff, pump, manometer and valves) .

Electro-mechanical devices are best modelled using CT abstractions such as differential equations and this makes the application of 20-Sim suitable in this case. On the other hand the control logic behind these components is best represented using DE abstractions such as VDM-RT. We use Crescendo in order to enable collaborative simulation making it possible to simulate complex control logic represented in VDM-RT with accurate physical models created under 20-sim. We proposed a particular way of applying this technology so energy consumption can be taken into account and one can perform trade-off analysis among different candidate solutions [10].

Software modelling. Energy consumption caused by software execution can be quantified at different levels of abstraction ranging from the microarchitectural level all the way up the operating systems services. We have analyzed it from a development point of view, in which a system designer can decide for how long and when the software execution can be halted and the processing core put in a low-power sleeping mode. Such a mode is supported by most of today's microcontrollers and the way it is used can have a significant impact on energy consumption. We propose the application of the VDM-RT modelling language to study how different sleeping policies can be used in the system. VDM-RT is appropriate in this case because it can represent control logic using a DE paradigm. Additionally it provides a virtual execution environment through the CPU abstraction, which we have extended so it is able to be put to sleep and activated through specific operations [11]. Regarding the case of the pervasive blood pressure measurement, an example of software in a healthcare device could be the logic that determines when a measurement has to be conducted and detects abnormal patient conditions.

[4] This is an acronym for "Integrated Tool Chain for Model-based Design of Cyber-Physical Systems" and information can be found at `www.into-cps.au.dk`.

Communication modelling. Energy consumption on the communication side as in the software case can be studied at different levels of abstraction. We focus our study on characterizing the usage of the network interface (for instance 802.15.4 or Bluetooth Low Energy (BLE) radio) in terms of mode of operation and duration. This analysis can be conducted to some extent with the VDM-RT modelling language thanks to its ability to model distributed aspects [17]. VDM-RT incorporates the abstraction BUS, that is used to connect different CPU execution environments. We proposed specific ways in which this can be used to represent small scale network topologies and how the notion of energy can be incorporated to it [13].

An example of a communication subsystem following the previous example of pervasive blood pressure measurements could be a Blueetooth Low Energy radio that allows this device to be part of a Body Area Network for patient monitoring.

Even though we have used the formal notation VDM-RT to represent software related aspects, we have not conducted formal verification over the models. The validation of the models has been conducted purely through simulation. The rationale behind this approach is to be able to explore the design space in a cost effective way. Since VDM-RT is used to represent the system it can be possible sometime in the future to conduct formal verification.

3 Design of an intelligent medical grade compression stocking

In order to analyze the validity of our approach we have applied these techniques to a concrete, real case study. This case study is based on the European Ambient Assisted Living project e-Stocking, in which we are creating an intelligent compression stocking to treat leg-venous insufficiency[5]. This system is required to deliver a compression that ranges from 40 mmHg at the ankle level to 20 mmHg below the knee. This wearable healthcare device is composed of mechanical, software and communication subsystems and since it is a medical device it must conform to a high level of quality. The device can be seen in Figure 1. This compression principle is based on three different inflatable bladders. The system consists of a set of pumps and a valve, an embedded control unit, a radio communication interface and a battery. A complete description of this system can be found in [15].

3.1 Design challenges

The main design challenges to be addressed during the development of this device are:

- The operational time should be between 12 to 14 hours, enabling the patient a complete day of use without requiring a battery recharge.
- The pressure delivered to the limb shall be constant and within the prescribed range. This implies that the stocking shall feature a regulation mechanism that makes sure that the delivered compression is inside the specified interval.

[5] e-Stockings project official website: `http://www.e-stockings.eu/`.

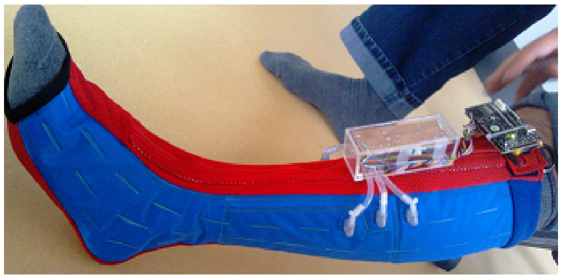

Fig. 1: The e-Stocking prototype.

– The stocking controller should be able to communicate with a smartphone acting as an internet gateway and/or a configuration tool.

We aim to tackle these challenges to some extent through the application of our modelling approach by: gaining a better understanding of the problem and being able to provide solution candidates based on the Design Space Exploration (DSE) conducted through modelling.

3.2 Component power consumption

The stocking system is composed of components of very different nature with different power consumption figures across different orders of magnitude. In order to give a better overview of the subsystem power consumptions we present the power requirements[6] of the key components of the system in Table 1.

Table 1: Overview of the typical components power consumption:

Component	**Current draw**	**Voltage**	**Power consumption**
Pump	110 mA	3 V	330 mW
Valve	120 mA	3 V	360 mW
Manometers	1.4 mA	3.3 V	4.62 mW
CPU Active / Sleeping / Hibernating	20 mA / 10 uA / 10 nA	3.3 V	66 mW / 33 uW / 33 nW
Radio Tx/Rx	35/40 mA	3.3 V	115.5/132 mW

Inspection of these figures reveals that electromechanical components such as pump and valves are the most power demanding. Since these are heavily used to administer the compression, initial efforts to minimize energy consumption should be focused in the mechanical area.

[6] These figures should be considered as average and approximate within their respective order of magnitude

3.3 Mechanical co-modelling

The mechanical side (also know as a *plant* in the control engineering domain) of the compression stocking has been represented in a 20-sim CT model. The plant control logic has been modelled in VDM-RT. The combination of the two models has been co-simulated using Crescendo. An overview of the mechanical side is given in Figure 2 and its components are described below.

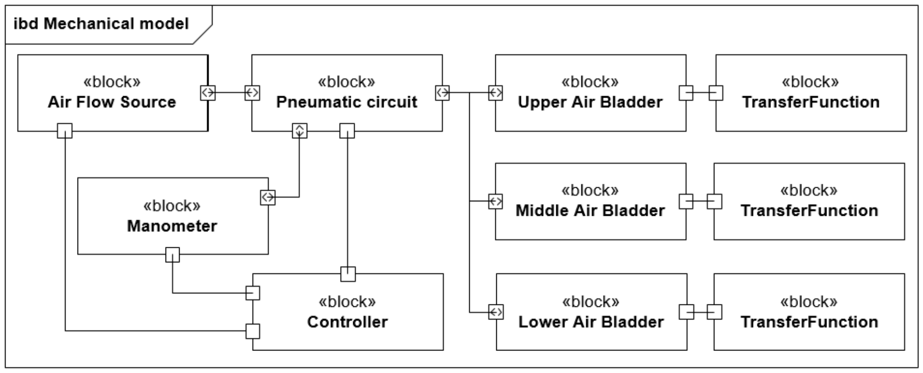

Fig. 2: SysML Internal Block Diagram of the mechanical model.

Air Flow Source: represents the pump that generates the air flow to be directed to the air-bladders to build pressure. It is controlled by the software controller and the effective airflow can be modulated.

Pneumatic Circuit: represents a concrete arrangement of valves and tubing connections. The valves are controllable from the software and this makes it possible to direct the air-flow to a single air bladder. The valves are also responsible for the venting of the air bladders, however this cannot be modulated.

Manometer: is an air pressure sensor that can monitor the level of compression in the different air bladders. Due to the way it is connected through the pneumatic circuit a single manometer can be used to monitor any of the three chambers.

Air Bladders: represent the chambers integrated in the stocking in which pressure builds-up. Their air intake can be locked without having to energized the valves.

Transfer Functions: are different mathematical expressions that determine how the pressure in the air chambers map to the pressure over the skin, representing the effective compression. They can be replaced depending on the compression principle under evaluation and the rest of the model still can be reused.

Controller: contains the necessary interface definitions to communicate the mechanical model with the software control logic modelled in VDM-RT.

Figure 3 presents the top level representation of the 20-sim plant model. The complete model connects two more bladder subsystems (shown with the dashed box) to the distribution valve, but they have been removed in this case for clarity. The pneumatic circuit presented above is decomposed in a *Distribution valve* and one *Pass valve* per bladder subsystem. The *Distribution valve* is responsible for directing the air flow to the air bladder that has to be inflated. The *Pass valve* is responsible for locking the air bladder once inflation has been completed. Hence, in order to inflate an air bladder two valves have to be energized. The block *LegSegment* introduced the transfer function that maps the pressure built in the *AirBladders* with the pressure exerted over the leg. This model incorporates the notion of power consumption in the most power demanding components: the *Distribution valve*, the *Pass valve* and the *Pump*. When the models are simulated the power consumption figures are integrated over time, resulting in the energy consumption for each individual component. The models are instrumented so both power and energy consumption are monitored variables in the simulation but without having an impact on the simulation performance.

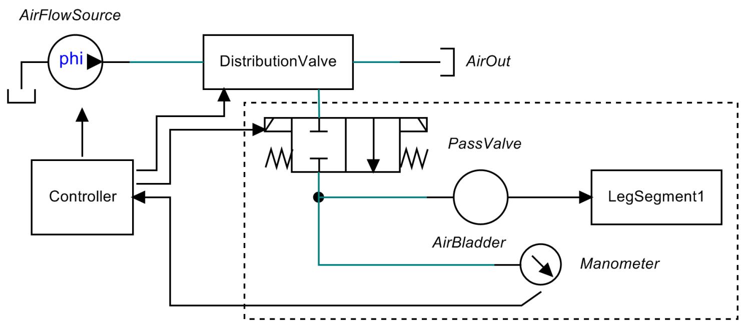

Fig. 3: 20-sim CT model of the compression stocking.

The software controller modelled in VDM-RT contains the necessary interfaces to control the simulated sensors (manometer) and actuators (valve and pump). Additionally it features different regulation algorithms and configurations. These regulation algorithms aim at maintaining the pressure in the air bladders constant and constitute the core logic of the DE models. The most relevant ones are:

Simple regulation: that is based on conditional logical statements and do not modulate the pump air flow. This implies that the pump will be engaged fully in case additional compression is needed and the air bladders will be vented in case overpressure occurs.

PID regulation: Proportional Integral Derivative [24], that modulates the pump airflow depending on the deviation from the target pressure. Due to this behaviour it is said to be proportional in the inflation. Due to the mechanical construction of the system it is not possible to modulate the venting process and therefore it cannot be made proportional in the deflation. Therefore, a full PID controller cannot be incorporated in this system.

Listing 1 shows part of the model for a proportional regulation of the pressure in one of the air bladders. In this case the pump is driven proportionally to the error (`err`): the difference between current and target pressure (setPoint). Prior to engage the pump at specific rate, the controller sets the air distribution valve and opens the pass valve for the target air bladder (AB1).

```
err := setPoint - manometerAB1.getPressure();
if (err > threshold) then (
  pTerm = err * pGain;
  controller.airDistribution.airToAB1();
  controller.passValveAB1.open();
  controller.pump.setPumpRate(pTerm); );
```

Listing 1: Proportional regulation applied to air bladder 1.

The execution of these models has enabled us to explore different regulation configurations to determine their effectiveness as well as to compare them regarding their energy performance. We evaluated two different scenarios: compression from an idle state and regulation after an under-pressure event occurred. Such an event might happen due to small leaks that have an impact on the pressure level during treatment. The simulation of the models allowed us to draw the following conclusions:

– The energy consumption during inflation and regulation is proportional to the time this process has taken and this is due to the particular configuration of the pneumatic circuit.
– This time can be reduced by inflating the air bladders as fast as possible, by configuring the PID controller with a high proportional constant,
– This implies that the valves that have to be triggered during inflation will be energized as briefly as possible, hence decreasing the energy consumption.

Applying this principle to the design of the software controller has lead to a decrement of ≈33% in energy consumption.

Additionally we have been able to explore ways to stabilize the controller taking into consideration different PID configurations. Given the fact that only the inflation can be controlled in a proportional manner and not the deflation, this has turned to be a challenging task. We decided to apply an error window that determines whether or not the regulation must be executed (hysteresis). This window should be taken into account in order to determine the periodicity of the regulation logic. A higher error window would result on a higher period for the real time thread that executes the controller. This could have an impact on the way the software makes use of the computational resources and therefore in the energy consumption. This is discussed further in the section below.

3.4 Software modelling

The control regulation logic introduced in the previous section is deployed as a software component, executed by the CPU integrated on the e-Stocking microcontroller. This CPU features a number of low power operational states to choose between when developing applications. Depending on the low power state used by the developer (Sleeping or Hibernating[7]) different kinds of wake-up mechanisms are available. In this work we have considered the following and most common ones:

Wake-up on sleep timer expiration: The CPU remains in a low power state until an internal timer overflows, generating an internal (within the chip) interrupt that wakes up the CPU. Applying this wake-up mechanism typically implies using the more energy demanding low power states such as the Sleep mode.

Wake-up on external event: The CPU remains in a low power state until an external event generates an interrupt that activates it. Applying this wake-up mechanism allow the use of Sleep or Hibernation modes.

These mechanisms facilitate the implementation of two different software regulation strategies for the e-Stocking case study:

Periodic regulation: in which the regulation logic is executed as a periodic thread. The period that determines how often the logic has to be executed can be determined by the study of the control requirements carried out during the mechanical modelling, presented in the section above. While the regulation is not executing, the processor can be put to sleep. This approach makes use of a *Wake-up on sleep timer expiration* strategy.

Event-triggered regulation: in which the regulation logic is executed once a pressure loss has been detected by smart sensors. The CPU can be put to sleep for an undefined period of time and remain in that state until it is notified by any of the sensors. This approach makes use of a *Wake-up on external event* strategy.

The energy consumption will depend directly on the strategy adopted and how it uses the low power features based on sleeping modes. These regulation strategies are implemented through two different system architectures, shown in the UML deployment diagram presented in Figure 4. These architectures are:

Architecture A: implements a periodic regulation strategy in which the CPU executes regulation logic and moves to sleeping state. After the sleep timer has expired it wakes up the CPU and executes the regulation process again. The system uses passive pressure sensors that have to be actively polled by the CPU.

Architecture B: implements an aperiodic regulation strategy in which the CPU hibernates until an abnormal pressure level is detected by the pressure sensors. This architecture makes use of Smart Pressure sensors, capable of monitoring the pressure levels independently of the main CPU. Once a pressure deviation has been detected by these pressure sensors they generate an external interrupt that wake up the main CPU, that will finally execute the regulation logic. The power consumption of these sensors is higher than the passive sensors but still negligible and orders of magnitude below the power consumption of the CPU executing the control logic.

Common to both architectures are the `CPU` core and the `PWMDriver` block, that is responsible for generating the airflow to inflate the chambers.

[7] Modern CPUs incorporate several low power modes, the most common being: a *Sleep* mode with a current draw within the order of microamps and a *Hibernation* mode, with a current draw of nanoamps and less reactive than the first one. Mode names vary depending on the manufacturer.

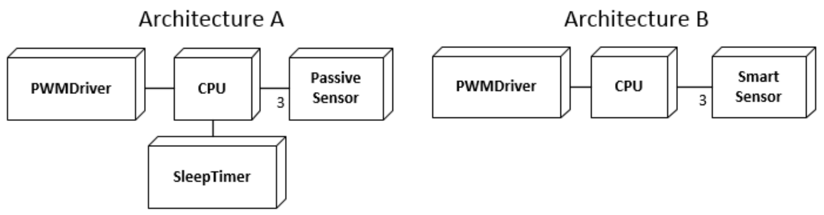

Fig. 4: Deployment diagram with architectures for the two different regulation strategies.

Both architectures have been modelled in VDM-RT as shown in the deployment diagram presented in Figure 5. In this diagram we use nodes to represent the VDM-RT CPUs (stereotyped as <<CPU>>). In these CPUs we deploy different parts of the model to run independently and they are connected through VDM-RT BUSes. This model can be configured in two different ways to represent either Architecture A or B, by using the model of the `SleepTimer` or the `WakeUpInterrupt` respectively. The `Controller` class deployed in the node `mcu` models the logic that implements the regulation functionality presented in the previous Listing 1 as well as the logic that determines whether the CPU is sleeping or not.

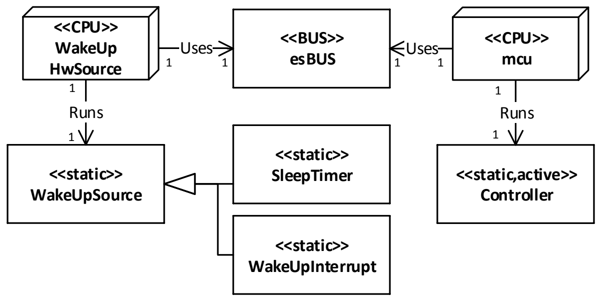

Fig. 5: Class diagram showing the models general structure.

We simulated these two architectures under a common scenario in which we considered the regulation over a period of time of 200 ms. In this scenario we simulated a pressure loss taking place at 150 ms. We simulated both system models and represented their power consumption over time, producing the activity graphs shown in Figure 6. Based on these CPU activity graphs, taking into consideration CPU manufacturer specifications and basic CPU current draw measurements we were able to predict concrete average power and energy consumption figures.

The simulation of these models show that the software used in Architecture A results in a higher energy consumption, since it causes periodic unnecessary system wake-ups to check pressure levels even though the regulation is not needed. This problem is solved in Architecture B where the CPU will not be activated until a regulation is needed.

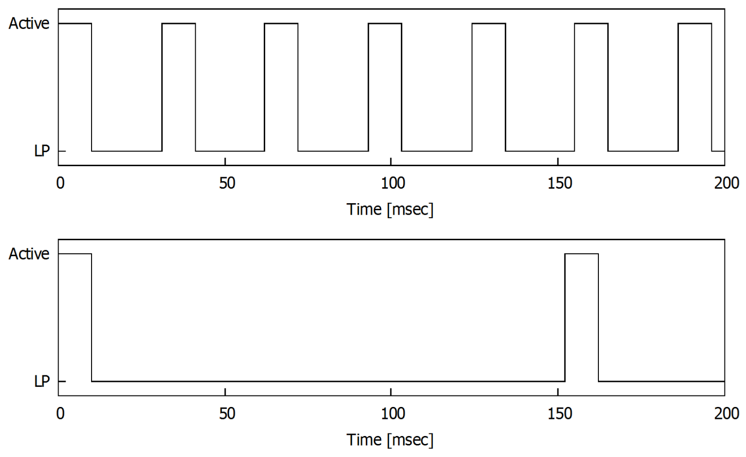

Fig. 6: Activity over time predicted by the models in Architectures A (upper graph) and B (lower graph) in the scenario under study.

However this comes at the cost of having to integrate more complex smart sensors[8], able to work independently from the CPU.

In order to validate our predictions we realized both architectures, implemented both regulation strategies and measured their power consumption. Finally, after numerically integrating the power consumption measured over time we determined the total energy consumption. The energy consumption predicted differed from the actual energy consumption by less than 5%. The measurements conducted over the two concrete system realizations are presented in Figure 7. Due to the specifics of the processor used in the prototype (ARM Cortex M3) it was possible to use a hibernation mode in the implementation of architecture B. Such a mode has lower current draw than the sleep mode (four orders of magnitude below). This is reflected in the measurements presented in Figure 7.

This initial application of modelling to energy consumption on the software side shows that our approach is sufficiently accurate to validate at the abstract modelling level computation issues from the power and energy consumption point of view.

Additionally it is worthwhile remarking that the same kind of analysis can be conducted with any software functionality running on the systems CPU that might have an impact on energy consumption. Obvious candidates for further analysis would be communication software and security protocols.

3.5 Communication modelling

The e-Stocking case present several communication scenarios [9]. We have focused on the most relevant from the energy consumption point of view:

[8] The power consumption of these sensors is not taken into consideration because it is negligible if compared with the power consumption of the CPU.

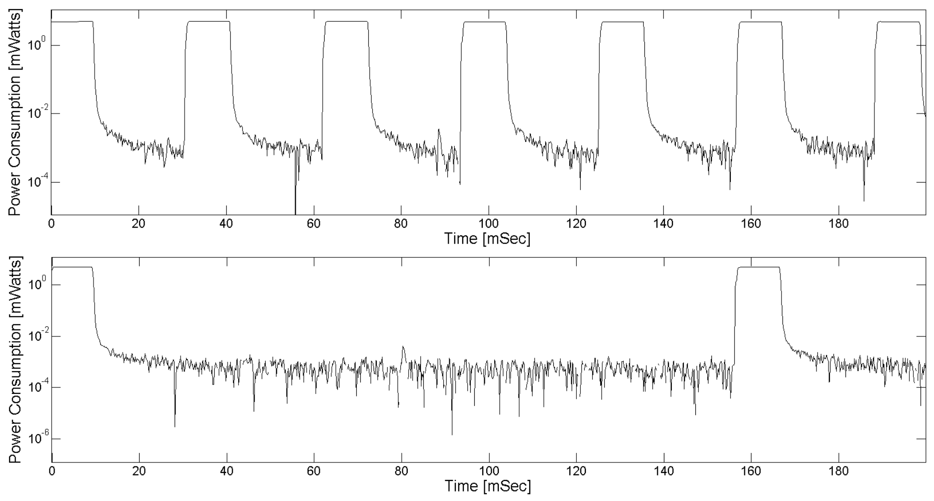

Fig. 7: Actual power consumption of the control logic execution over time in Architectures A (upper graph) and B (lower graph). Note the logarithmic vertical scale

Health monitoring: In this scenario the stocking transmits current and historical data regarding treatment adherence and condition evolution.

Calibration and configuration: In this scenario the stocking settings that determine how the treatment is conducted (mainly pressure levels) are set through an external device.

In order to model these scenarios, we proposed the application of the Distributed Real-Time features of VDM-RT. The models developed using VDM-RT follow the structure presented in [13]. This structure supports the modelling of small-scale networks such as Body Area Networks with several nodes interacting. In this case this structure can be simplified since there are only two nodes communicating. This structure is applied to this case as shown in the UML deployment diagram presented in Figure 8. The resulting model uses VDM-RT CPUs as execution environments in which the communication logic is run. The nodes represent the e-Stocking and an external client that connects to it in order to perform Health monitoring or Calibration and configuration as described above. The connection between the two nodes is represented through a VDM-RT BUS representing a 802.154 link. Each node is able to transmit data to the other by pushing it through the bus to the target receive buffer (`eSBuffer` and `clientBuffer`). Each node is able to receive data by reading its local receive buffers.

The modelling approach we have followed to study communication is composed of the following steps:

1. **Abstract protocol modelling (Model 1):** consists of high level modelling of the communication process, capturing the interactions between the devices. These models represent the information exchanged by tokens. No notion of energy is present at this stage.

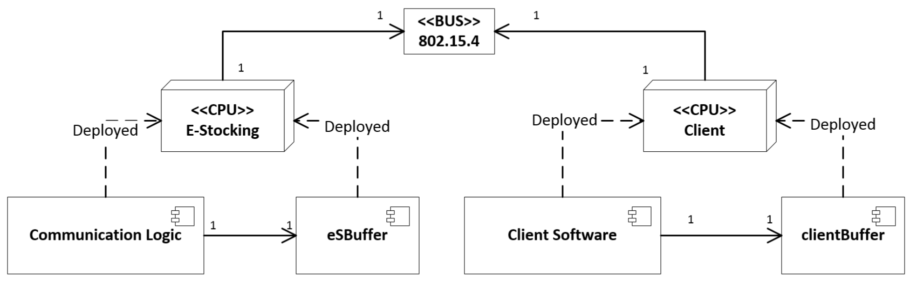

Fig. 8: UML deployment diagram showing the structure of the VDM-RT communication models.

2. **Definition of application level messages (Model 2):** consist on the creation of the application level Protocol Data Units (PDU) that will be exchanged between the devices. These messages can be made part of the model and replace the tokens.
3. **Profiling of the communication interface power consumption:** this is the characterization of the network interface used to communciate the devices. This profiling can be made experimentally or by consulting the product datasheets. As a result of this step it should be possible to have a rather precise estimation of the transmission and reception power consumption per byte.
4. **Profiling energy consumption of each message exchanged in the communication:** this is the construction of a look-up table in which it is described how much energy it takes to send and receive each message that compose the protocol.
5. **Model execution and production of energy consumption estimations (Model 3):** results on a high-fidelity energy consumption estimation based on the previous models capturing the interactions and the characterization of the energy consumption of each package exchange.
6. **Protocol rework:** in case it is desired one can rework the interactions or the protocol in order to lower the power consumption of the communication.

The initial abstract model captures the interactions between the client and the e-stocking in terms of who is initiating the communication and what information is exchanged. The communication logic executed in each party is modelled in separate procedural threads handling the communication buffers. Initially these models do not make use of the notion of time present in VDM-RT since communication-related times have not been determined yet. Taking as an example the configuration scenario and focusing on the communication in the e-Stocking node, we can see the output of such a model in the first column of Table 2.

After this initial modeling, and following step two of our communication modelling approach, we defined the application level messages that correspond to the tokens used previously. This gives a better idea regarding the amount of information that has to be exchanged between the client and the stocking. The results of this modelling step are shown as an addition to model execution log shown before in Table 2, column 2.

Once these modelling stages were completed we proceed to study the communication interface power consumption during transmission and reception. We determine

Table 2: e-Stocking communication model execution results for the configuration scenario:

Model 1 **Interactions**	**Model 2** **Information**	**Model 3** **Time, Energy Consumption**
<– – RX <ConnectionRequest>	s0e	0.5 mSec +22 μJoules
– – >TX <ACK >	s1e	1 mSec +64 μJoules
<– – RX <SetPressures >	s53203040e	1.5 mSec +22 μJoules
– – >TX <ACK >	s1e	2 mSec +64 μJoules
<– – RX <Disconnection>	s2e	2.5 mSec +22 μJoules
Total communication time 2.5 mSec, Total energy consumption: 194 μJoules		

the power consumption of the interface in transmission and reception experimentally and based on these measurements determined the energy consumed per transmitted and received message. Due to the extra headers added by layers underneath the application, the high baudrate of the communication link, and similar application message lenghts, there is very small difference among the messages considered in terms of transmissions and receptions times and, therefore, in power consumption. Figure 9 shows the power consumption in the communication interface when transmitting and receiving a single byte.

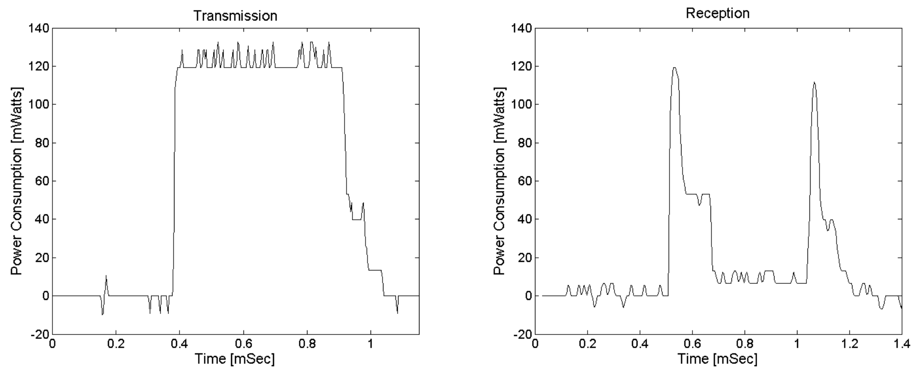

Fig. 9: Power consumption for a single byte transmission and reception in the communication interface.

After gaining insight into the physical implications in the communication interface of different kinds of messages, it is possible to establish real transmission and reception times as well as real power consumption figures. This makes it possible to incorporate the measurements to the communication models and create high fidelity estimations in terms of communication times and communication energy consumption. The final models are able to produce extended logs in which time and energy are taken into account. Table 2 shows the extended model execution results in column 3.

Besides analyzing the configuration scenario, we have also studied the health monitoring scenario. In this case we have proceed in a similar manner, following the steps defined previously, with the only difference that we decided to rework the communication logic, sending longer messages containing more information to minimize the energy spent on transmitting headers.

4 Results and Discussion

The creation of the initial mechanical model already had a positive impact on the development process without taking energy consumption into consideration. This preliminary model helped us in getting an understanding of the pneumatics domain before delving into particular implementation aspects. Regarding the particular case study, through the modelling of the mechanical subsystem it became clear that establishing a transfer function that determined how pressure inside the air bladders mapped to the skin was critical to achieve an effective regulation. This led to the incorporation of a manometer into the system to enable measurements of air pressure in the system. Both suggestions have been transferred to the real implementation of the system being carried out in parallel. Additionally, through the application of modelling we were able to evaluate an alternative compression mechanism and discard it since it was not a feasible solution. The same conclusion was reached by a partner working in parallel but through prototyping.

Regarding energy consumption in computation, other approaches to energy savings apply dynamic frequency and voltages scaling. Such an approach is not especially helpful in this case because the operations carried out during the regulation are not especially computationally intensive. Additionally there is a bottleneck time-wise in the interaction with the physical environment (such as the sensor's response time). Finally, there are very small variations between different operating frequencies in the CPU used and therefore one cannot save as much energy as in the current approach through the application of sleeping operational modes.

The estimates that we were able to achieve through the application of modelling in this case study were coarse grained estimates, fit for the purpose of facilitating taking design decisions and trade-off analysis. For more precise figures (especially in the mechanical case) it is necessary to perform concrete measurements on the concrete realization.

The application of models has allowed us to conduct DSE without having to conduct extensive prototyping of several solutions and without having to run long batches of experiments manually. This brings the advantage of reduced costs in system development due to lesser expenditures in development time and prototyping material.

This approach to system design would be ideally carried out by at least two engineers: one with a thorough knowledge of mechanics and physical modelling and a second expert in embedded systems, including both hardware and software issues. Experts in embedded systems typically have certain knowledge of system communications and therefore can also cover energy consumption on that side. In our case this methodology has been applied by a single engineer specialized in the embedded area but with a basic knowledge of pneumatics. Occasional support on mechanical modelling was provided by experts on the field. The modelling conducted by this single engineer resulted in several design suggestions and input to the development of the compression stocking being carried out in parallel. From an industrial perspective developers who wishes to use a similar approach would need to learn the different modelling notations. Given that you understand the underlying theory that is not hard but as with any other technology “A fool with a tool is still a fool”, so there is no doubt that there is a modest investment in how to model things sufficiently accurate so the models get competent.

5 Related work

The energy consumption in CPS in general and in wearable devices is a well recognized problem [23, 2]. Typical approaches to wearable healthcare systems are based on the development of prototypes. Extensive examples of systems developed following this approach can be found in the literature (e.g. [25, 20]). However, these approaches are purely focused on system functionality rather than energy efficiency and therefore they address it at the end of the development process and as a final factor to optimize. Opposed to this prototyping-based methodology one can only find limited related work in which a model-driven approach is applied to the development wearable healthcare systems. Anliker et. al. propose a concrete method to design distributed wearable systems, making special emphasis on the analysis of computation and communication logic [1]. This approach considers energy consumption among other design factors and they formalize them through the application of different metrics to perform DSE. Previous modelling work exists on wearable health monitoring devices through Stochastic Petri Nets [21]. This work is very focused on the software functionality and does not take energy consumption into consideration.

Finally, the development of wearable healthcare devices can benefit from some of the model-based approaches to CPS design, such as the one shown in DESTECS [3] and presented before the one proposed by Jensen et. al. [14]. Even though these do not make special emphasis on the energy consumption of the system, the provided tools that can be used following the approach we proposed in [13] so energy consumption can be taken into account during the development process and described from different perspectives.

6 Future work

The analysis conducted in this case study has enabled the production of energy consumption estimations by conducting simple initial measurements combined with modelling. At this point we are considering how this model-based engineering approach can benefit from partial system prototyping. This could potentially be facilitated by our previous work in Hardware in the Loop and the combination of models and partial system realizations in a single co-execution [12]. Additionally we are applying the same energy-aware model-driven engineering approach discussed in here to a second case study in the medical domain: a platform for The Pacemaker Formal Methods Challenge[9], paying especial attention to the energy consumption analysis.

[9] More details can be found in `http://sqrl.mcmaster.ca/pacemaker.htm`

7 Conclusion

We have presented the application of an energy-aware model-based approach to the development of a wearable medical device: a compression stocking to treat leg-venous insufficiency. This approach has taken into consideration three critical subsystems that compose this solution: mechanical, computation and communication subsystems. The modelling techniques presented here combined with partial prototyping have helped us during the analysis, design and implementation of this compression stocking. Thanks to the application of modelling we have been able to evaluate different mechanical compression principles, redesign mechanical subsystem to reduce energy consumption, evaluate different regulation algorithms and get a grasp on how the software can be configured to reduce energy consumption. Additionally we have evaluated the energy consumption in different communication scenarios.

Hopefully this work will inspire other medical device developers and convince them to apply a model-based approach instead of a prototyping-driven one, allowing them to gain confidence on the solution designed and to reduce development costs.

Acknowledgements

This research is partially funded by the EU Ambient Assisted Living Joint Programme, eStockings Project under grant agreement no. AAL-2011-4-020.

References

1. Urs Anliker, Jan Beutel, and Matthias Dyer et al. A Systematic Approach to the Design of Distributed Wearable Systems. *IEEE Transactions on Computers*, 53(8):1017 – 1033, August 2004.
2. Banerjee, A. and Venkatasubramanian, K.K. and Mukherjee, T. and Gupta, S. K S. Ensuring Safety, Security, and Sustainability of Mission-Critical Cyber-Physical Systems. *Proceedings of the IEEE*, 100(1):283–299, 2012.
3. J. F. Broenink, P. G. Larsen, M. Verhoef, C. Kleijn, D. Jovanovic, K. Pierce, and F. Wouters. Design Support and Tooling for Dependable Embedded Control Software. In *Proceedings of Serene 2010 International Workshop on Software Engineering for Resilient Systems*, pages 77–82. ACM, April 2010.
4. John Fitzgerald, Carl Gamble, Peter Gorm Larsen, Kenneth Pierce, and Jim Woodcock. Cyber-Physical Systems design: Formal Foundations, Methods and Integrated Tool Chains. In *FormaliSE: FME Workshop on Formal Methods in Software Engineering*, Florence, Italy, May 2015. ICSE 2015.
5. John Fitzgerald, Carl Gamble, Richard Payne, Peter Gorm Larsen, Stylianos Basagiannis, and Alie El-Din Mady. Collaborative Model-based Systems Engineering for Cyber-Physical Systems – a Case Study in Building Automation. In *INCOSE 2016*, Edinburgh, Scotland, July 2016.
6. John Fitzgerald, Peter Gorm Larsen, and Marcel Verhoef, editors. *Collaborative Design for Embedded Systems – Co-modelling and Co-simulation*. Springer, 2014.
7. Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: State of the art. Technical report, feb 2017.

8. Sandeep K.S. Gupta, Tridib Mukherjee, Georgios Varsamopoulos, and Ayan Banerjee. Research Directions in Energy-Sustainable CyberPhysical Systems. *Sustainable Computing: Informatics and Systems*, 1(1):57 – 74, 2011.
9. Finn Overgaard Hansen, Troels Fedder Jensen, and José Antonio Esparza. Distributed ICT Architecture for Developing, Configuring and Monitoring Mobile Embedded Healthcare Systems. In *International Conference on Health Informatics (HEALTHINF 2014)*, March 2014.
10. José Antonio Esparza Isasa, Finn Overgaard Hansen, and Peter Gorm Larsen. Embedded Systems Energy Consumption Analysis Through Co-modelling and Simulation. In *Proceedings of the International Conference on Modeling and Simulation, ICMS 2013*. World Academy of Science, Engineering and Technology, June 2013.
11. José Antonio Esparza Isasa, Peter W.V. Jørgensen, and Claus Ballegaard. Modelling Energy Consumption in Embedded Systems with VDM-RT. In *Proceedings of the 4th International ABZ conference.*, July 2014.
12. José Antonio Esparza Isasa, Peter W.V. Jørgensen, and Peter Gorm Larsen. Hardware In the Loop for VDM-Real Time Modelling of Embedded Systems. In *MODELSWARD 2014, Second International Conference on Model-Driven Engineering and Software Development*, January 2014.
13. José Antonio Esparza Isasa, Peter Gorm Larsen, and Finn Overgaard Hansen. A Holistic Approach to Energy-Aware Design of Cyber-Physical Systems. *International Journal of Embedded Systems*, 9(3):283–295, June 2017.
14. J.C. Jensen, D.H. Chang, and E.A. Lee. A Model-Based Design Methodology for Cyber-Physical Systems. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 1666–1671, 2011.
15. Troels Fedder Jensen, Finn Overgaard Hansen, and José Antonio Esparza et al. ICT-Enabled Medical Compression Stocking for Treatment of Leg-Venous Insufficiency. In *International Conference on Biomedical Electronics and Devices (BIODEVICES 2014)*, March 2014.
16. Peter Gorm Larsen, Nick Battle, Miguel Ferreira, John Fitzgerald, Kenneth Lausdahl, and Marcel Verhoef. The Overture Initiative – Integrating Tools for VDM. *SIGSOFT Softw. Eng. Notes*, 35(1):1–6, January 2010.
17. Peter Gorm Larsen, John Fitzgerald, and Sune Wolff. Methods for the Development of Distributed Real-Time Embedded Systems using VDM. *Intl. Journal of Software and Informatics*, 3(2-3), October 2009.
18. Peter Gorm Larsen, John Fitzgerald, Jim Woodcock, Peter Fritzson, Jörg Brauer, Christian Kleijn, Thierry Lecomte, Markus Pfeil, Ole Green, Stylianos Basagiannis, and Andrey Sadovykh. Integrated Tool Chain for Model-based Design of Cyber-Physical Systems: The INTO-CPS Project. In *CPS Data Workshop*, Vienna, Austria, April 2016.
19. Peter Gorm Larsen, John Fitzgerald, Jim Woodcock, and Thierry Lecomte. *Trustworthy Cyber-Physical Systems Engineering*, chapter Chapter 8: Collaborative Modelling and Simulation for Cyber-Physical Systems. Chapman and Hall/CRC, September 2016. ISBN 9781498742450.
20. M. I. Mokhlespour and O. Zobeiri and R. Narimani and et. al. Design and Prototyping of Wearable Measuring System for Trunk Movement Using Textile Sensors. In *Proceedings of the 20th Iranian Conference on Electrical Engineering, (ICEE2012)*, pages 1571 – 1575, May 2012.
21. Alexandros Pantelopoulos and Nikolaos Bourbakis. SPN-Model based Simulation of a Wearable Health Monitoring System. In *Proceedings of the 31st Annual International Conference of the IEEE EMBS*, pages 320–323. IEEE, September 2009.
22. Marcel Verhoef, Peter Gorm Larsen, and Jozef Hooman. Modeling and Validating Distributed Embedded Real-Time Systems with VDM++. In Jayadev Misra, Tobias Nipkow,

and Emil Sekerinski, editors, *FM 2006: Formal Methods*, Lecture Notes in Computer Science 4085, pages 147–162. Springer-Verlag, 2006.

23. Timo Vuorela. *Technologies for Wearable and Portable Physiological Measurement Devices*. PhD thesis, Tampere University of Technology, 2011.
24. Tim Wescott. PID Without a PhD. *Embedded Systems Design*, (86 - 108), October 2000.
25. Kiing-Ing Wong. Rapid Prototyping of a Low-Power, Wireless, Reflectance Photoplethysmography System. In *Proceedings of the 2010 International Conference on Body Sensor Networks*, pages 47 – 51, 2010.