Internet of Things Patterns for Devices

Lukas Reinfurt[1,2], Uwe Breitenbücher[1], Michael Falkenthal[1],
Frank Leymann[1], Andreas Riegg[2]

[1]Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{firstname.lastname}@iaas.uni-stuttgart.de

[2]Daimler AG, Stuttgart, Germany
{firstname.lastname}@daimler.com

BIBTEX:

```
@inproceedings{Reinfurt.2017,
  author    = {Reinfurt, Lukas and Breitenb{\"u}cher, Uwe and Falkenthal,
Michael and Leymann, Frank and Riegg, Andreas},
  title     = {Internet of Things Patterns for Devices},
  booktitle = {Ninth international Conferences on Pervasive Patterns and
Applications (PATTERNS) 2017},
  year      = {2017},
  pages     = {117--126},
  publisher = {Xpert Publishing Services (XPS)}
}
```

Internet of Things Patterns for Devices

Lukas Reinfurt[1], Uwe Breitenbücher,
Michael Falkenthal, Frank Leymann
Institute of Architecture of Application Systems
University of Stuttgart
Stuttgart, Germany
email:{firstname.lastname}@iaas.uni-stuttgart.de

Andreas Riegg
[1]Daimler AG
Stuttgart, Germany
email:{firstname.lastname}@daimler.com

Abstract—**Devices are an important part of the Internet of Things. They collect data from their environment with sensors and, based on this data, also act on their environment by using actuators. Many use cases require them to support characteristics such as being cheap, light, small, mobile, energy efficient, or autonomously powered. This creates constraints for available energy sources and leads to different kinds of operating modes. Based on existing terminology and additional examples, we describe these energy constraints and the operation modes in the form of Patterns. These Patterns are interconnected with other Patterns to form an Internet of Things Pattern Language that enables practitioners to find and navigate through proven solutions for their problems at hand.**

Keywords—Internet of Things; Patterns, Devices; Constraints.

I. INTRODUCTION

The development of the Internet of Things (IoT) is gaining momentum. Companies and research institutes create new technologies, standards, platforms, applications, and devices in rapid succession. As a result, it becomes increasingly hard to keep track of these developments.

We started creating IoT Patterns to help individuals working in this area [1][2]. By methodically collecting common problems and their solutions and abstracting them into Patterns, we are building up an IoT Pattern Language. These Patterns help others understand the core issues and solutions in the IoT space and provide them with the means to apply these solutions to problems in their own projects.

Devices are an important part of the IoT, as they are the point where sensors and actuators bridge the gap between the real world and its digital representation. To fulfill the vision of the IoT, a world where nearly everything works together to react and automatically adjusts to its environment, devices have to be ubiquitous. They come in all shapes and sizes and will be located not only in controlled indoor environments but also outside and in harsh conditions. For example, some of them are required to be mobile and are located off the power grid.

Such requirements lead to constraints in cost, size, weight, or available power and hence influence the choice of power source. Different power sources also require different operation modes. For example, Bormann et al. describe different energy sources and operation modes in their terminology for constrained-node networks [3].

Based on this terminology and additional sources describing the application of IoT devices in real world scenarios, we created six Patterns for IoT devices with different energy sources and operation modes. Devices can be ALWAYS-ON DEVICES, PERIOD ENERGY-LIMITED DEVICES, LIFETIME ENERGY-LIMITED DEVICES, or ENERGY-HARVESTING DEVICES, depending on the energy source they use. The energy source also influences a device's operation mode, thus it can be an ALWAYS-ON DEVICE or a NORMALLY-SLEEPING DEVICE.

The rest of this paper is structured as follows: Section II provides a short overview of previous work related to this paper. Section III briefly summarizes our understanding of Patterns and our previously published IoT Patterns. Section IV introduces six new IoT Patterns for devices and shows how they are connected among themselves and to the already presented ones. Section V describes three of the six new Patterns, namely PERIOD ENERGY-LIMITED DEVICE, ENERGY-HARVESTING DEVICE, and NORMALLY-SLEEPING DEVICE, in detail. Finally, Section VI provides a summary and outlook.

II. RELATED WORK

The Pattern concept was first introduced by Alexander et al. in the architecture domain [4]. Since then, the concept has been applied in other domains. Examples from IT include the Messaging Patterns by Hohpe et al. [5] or the Cloud Computing Patterns by Fehling et al. [6]. There has also been work on the Pattern writing process itself [7][8][9][10].

We presented our first five IoT Patterns, DEVICE GATEWAY, DEVICE SHADOW, RULES ENGINE, DEVICE WAKEUP TRIGGER, and REMOTE LOCK AND WIPE [1]. We later added three more Patterns, namely DELTA UPDATE, REMOTE DEVICE MANAGEMENT, and VISIBLE LIGHT COMMUNICATION [2]. These Patterns are not concerned with IoT devices themselves but do already mention the terminology by Bormann et. al [3]. They present a terminology for constrained nodes, constrained networks, and constrained-node networks. They describe some aspects of why and how different energy sources and operation modes occur, but not in the form of Patterns. The Pattern format used in this paper adds more to this description in form of the forces, the result section, and the benefits and drawbacks, as well as the interconnection with other Patterns.

Eloranta et. al published a Pattern Language for designing distributed control systems [11]. These Patterns focus on larger machinery and are not concerned with small constrained devices and the implications of these constraints.

Other Patterns in the IoT space exist which are not concerned with the devices themselves. Qanbari et. al present four Patterns for edge application provisioning, deployment, orchestration, and monitoring, which use existing technologies like Docker or Git that are not suited for constrained devices [12].

III. IoT Patterns Overview

The Patterns presented in this paper and our previous work follow the ideas of Alexander [4] and others [7][8][9][10]. As described in more detail in [1][2], we identified these Patterns by collecting material from product pages, manuals, documentation, standards, whitepapers, and research papers. Once reoccurring descriptions became evident, we grouped them and extracted the core principles into the more abstract Pattern format. The format is also described in more detail in [1][2] but, in short, is made up of the following elements: The **Name**, **Icon**, and **Aliases** help to identify the Pattern. The short **Problem** and **Solution** sections contain the core issue and steps to resolve it. The **Context** and **Forces** describe where the problem occurs and why it is hard to solve, while the **Result** section gives more details on the solution. Other relevant Patterns are listed as **Related Patterns**. Existing products which implement the Pattern and were used as sources are summarized under **Known Uses**. Table 1 provides an overview of our earlier Patterns, including a short summary of the problems they are solving and a brief description of how they solve it.

TABLE 1. OVERVIEW OF OUR PREVIOUS IOT PATTERNS

Device Gateway	**P.:** You want to connect many different devices to an already existing network, but some of them might not support the networks communication technology or protocol. **S.:** Connect devices to an intermediary Device Gateway that translates the communication technology supported by the device to communication technology of the network and vice-versa.
Device Shadow	**P.:** Some devices are only intermittently online to save energy or because of network outages. Other components want to interact with them but do not know when they will be reachable. **S.:** Store a persistent virtual representation of each device on some backend server. Include the latest received state from the device, as well as commands not yet sent to the device. Do all communication from and to the device through this virtual version. Synchronize the virtual representation with the actual device state when the device is online.
Rules Engine	**P.:** Throughout its operation, a system receives a wide range of messages from devices and other components. You want to react in different ways to these messages. **S.:** Pass all messages received from devices through a Rules Engine. Allow users to define rules that evaluate the content of incoming messages or metadata about the message against a set of comparators. Also allow external data sources to be included in these comparisons. Let users associate a set of actions with these rules. Apply each rule on each message and trigger the associated actions if a rule matches.
Device Wakeup Trigger	**P.:** Some devices might go into a sleep mode to conserve energy and only wake up from time to time to reconnect to the network. During sleep, they are not reachable on their regular communication channels. In some instances, other components might have to contact sleeping devices immediately. **S.:** Implement a mechanism that allows the server to send a trigger message to the device via a low energy communication channel. Have the device listening for these triggering messages and immediately establish communication with the server when it receives such a message.
Remote Lock and Wipe	**P.:** Some devices might be lost or stolen. You want to prevent attackers from misusing the functionality of the device, or from gaining access to the data on the device or to the network through the device. **S.:** Make the device a managed device that can receive and execute management operations from the backend server. Allow authorized users to use the backend server to trigger functionality on the device that can delete files, folders, applications or memory areas, revoke or remove permissions, keys, and certificates, or enable additional security feature. Execute triggered functions as soon as the device receives them and provide an acknowledgment to the backend.
Delta Update	**P.:** You want to reduce the size of messages containing sensor data without losing any information. **S.:** Store the last message send. Calculate the delta from the current data to this message. Also, calculate a hash of the current data. Send only the delta and the hash to the receiver. Let the receiver merge the delta with its current state and check, if it matches the received hash.
Remote Device Management	**P.:** You want to manage a large number of devices remotely. **S.:** Set up a management server on the backend. Add management clients to the device which you want to manage. Send management commands from the server to the client and let the client execute these commands locally on the device.
Visible Light Communication	**P.:** You need to use wireless communication in a crowded area, but you cannot use the crowded radio spectrum. **S.:** Use visible light for short distance wireless communication. Modulate messages into the light by turning the light on and off. Do it fast to not impede normal light usage and to be invisible to the human eye.

IV. Internet of Things Patterns for Devices

In this paper, we add six new IoT Patterns for devices, three of which are presented in detail in Section V. This section presents an overview of all of them in Table 2 and 3, including some additional explanations of the Patterns not further described in Section V.

Related Patterns are organized into two groups. The first group, *Energy Supply Types*, is summarized in Table 2 and describes Patterns based on different forms of energy sources a device might use. Which one of these is applicable depends on the use case and its environment. If for example, a device is required for a wearable use case, then a Mains-

Powered Device is not an option. But the environment of the use case might also not provide sufficient ambient energy for an Energy-Harvesting Device.

The second group, *Operation Modes,* is summarized in Table 3 and lists different Patterns based on a device's mode of operation. These often depend on the amount of energy available to the device. For example, if a device is an Energy-Harvesting Device, it will in many cases not have enough energy to be an Always-On Device and has to be a Normally-Sleeping Device.

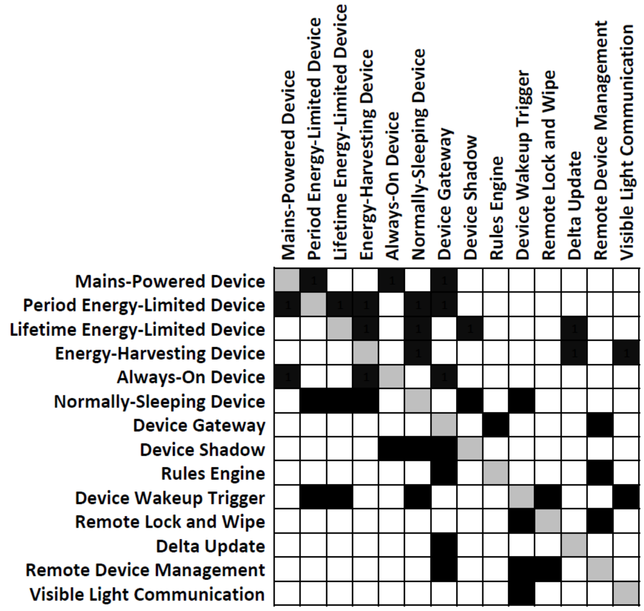
TABLE 2. OVERVIEW OF THE NEW IOT PATTERNS CONCERNED WITH DEVICE ENERGY SOURCES

Energy Supply Types	
Mains-Powered Device	**P.:** You need to power a stationary device, which requires a lot of energy. **S.:** Connect the device to mains power. **(**These types of devices do not have a direct limitation on energy. They are useful if batteries or energy harvesting do not provide enough power or are too maintenance intensive for the intended use case. They trade in more power for dependency on the grid and loss of mobility. They often are always-on but using other energy saving operation modes can lower energy cost.)
Period Energy-Limited Device (Section 0)	**P.:** You need to power a device, which requires a fair amount of power. The device is mobile or located in a remote place. Moreover, mains power is not available. **S.:** Use a replaceable or rechargeable source of energy to power the device. Implement a notification mechanism that informs you when the power source is nearly empty. Replace or recharge the power source when needed.
Lifetime Energy-Limited Device	**P.:** You need to power a device, which requires a small amount of power. The device is mobile or located in a remote place. You want to minimize maintenance. **S.:** Build an energy source into the device, which will last for the entire expected lifetime of the device. **(**Integrating a non-renewable energy source into a device can make sense if renewal is made difficult or impossible by the device's placement and if mains power is not available. The device should consume little energy and should have a known maximum lifetime. A normally-sleeping operation mode should be used to further maximize lifetime. Once the energy source is depleted, the device is useless, but until then it is low in maintenance and costs, simple and cheap to build, and highly independent.)
Energy-Harvesting Device (Section V.B)	**P.:** You need to power a device with very little power needs. The device is mobile or located in a remote place. Its environment is stable and predictable. **S.:** Integrate an energy harvesting component, such as a solar cell, into the device. Use it to turn the energy available in the device's surroundings into power for the device. Use components and technologies optimized for low-power usage to make the most of the harvested energy.

TABLE 3. OVERVIEW OF THE NEW IOT PATTERNS CONCERNED WITH DEVICE OPERATION MODES

Operation Modes	
Always-On Device	**P.:** You have a device with an unlimited energy supply and need to have it available and responsive at all times. **S.:** Leave the device turned on and connected at all times. **(**Leaving a device always on allows it to constantly take measurements and communicate with others, which may be required for some use cases. This requires more energy than other energy saving operation modes. Thus, being mains-powered or completely powered by energy harvesting is useful, or otherwise, maintenance will be high.)
Normally-Sleeping Device (Section V.C)	**P.:** You have a device with a limited energy supply. You want to minimize the power used by the device. **S.:** Program the device to disable its main components when they are not needed. Leave a small circuit powered which reactivates the components after a predefined amount of time has passed or when an event occurs.

These new Patterns do not exist in a vacuum. They are connected among themselves and to the Patterns which we previously presented [1][2]. Fig. 1 shows an overview of all the connections between the IoT Patterns. A black box in a row means that the Pattern represented by this row relates to the Pattern represented by the column in which the box is placed (the gray boxes show, where a Pattern is compared with itself). For example, in row four, a black box in column six shows that the Energy-Harvesting Device Pattern mentions the Normally-Sleeping Device Pattern. The nature of the connection is not further elaborated in this figure but could be interesting for future research.



	Mains-Powered Device	Period Energy-Limited Device	Lifetime Energy-Limited Device	Energy-Harvesting Device	Always-On Device	Normally-Sleeping Device	Device Gateway	Device Shadow	Rules Engine	Device Wakeup Trigger	Remote Lock and Wipe	Delta Update	Remote Device Management	Visible Light Communication
Mains-Powered Device	▨	■			■		■							
Period Energy-Limited Device	■	▨	■	■		■	■							
Lifetime Energy-Limited Device			▨	■		■		■				■		
Energy-Harvesting Device				▨		■						■		■
Always-On Device	■			■	▨		■							
Normally-Sleeping Device		■	■	■		▨		■		■				
Device Gateway							▨		■				■	
Device Shadow					■	■	■	▨						
Rules Engine							■		▨				■	
Device Wakeup Trigger		■	■			■				▨	■			■
Remote Lock and Wipe										■	▨		■	
Delta Update							■					▨		
Remote Device Management							■			■	■		▨	
Visible Light Communication										■				▨

Figure 1. Connections between IoT Patterns.

As the applicability of the IoT device Patterns presented in this paper is heavily influenced by the particular use case, it seems reasonable to choose them as entry points into the IoT Pattern Language when designing an IoT system. Their selection then greatly influences the design of the remaining system by suggesting or forcing certain additional Patterns. For example, if a use case requires a Period Energy-Limited Device, then also being a Normally-Sleeping Device will greatly enhance its energy efficiency. Adding a Device Shadow will make the overall system more robust and using a Device Wakeup Trigger will allow you to communicate with a Normally-Sleeping Device in an instant if necessary.

In turn, if new devices should be added to an existing IoT system, the design decisions elaborated in the architecture of the existing system will dictate which kinds of devices can be added without modifications, or what modifications have to be made to support a specific kind of device.

V. Detailed IoT Patterns for Devices

In this section, we describe three IoT Device Patterns in more detail. Out of the *Energy Supply Types* category we describe the Period Energy Limited Device and the Energy-Harvesting Device Pattern. From the *Operation Mode* category, we describe the Normally-Sleeping Device Pattern.

A. Period Energy-Limited Device

Aliases: Rechargeable

Context: You have a device, which needs a fair amount of energy to work but does not necessarily require mains power, such as a device that takes regular sensor readings, communicates, and powers actuators. Besides, your use case dictates a specific location for this device which restricts available energy sources. For example, the device has to be mobile, wearable, or in a remote location.

Problem: You need to power a device which requires a fair amount of power. The device is mobile or located in a remote place. Moreover, mains-power is not available.

Forces:

- **Energy Needs:** The device needs a fair amount of energy to work. A Lifetime Energy-Limited Device is not an option if it needs more in its lifetime than current batteries offer in a reasonable form factor. An Energy-Harvesting Device is not an option if the device needs more power for a cycle than the harvesting generates between cycles.
- **Environmental Constraints:** Your use case enforces a specific location for the device. For example, the device has to be mobile or wearable, or the device location is in an area where mains power is not available. Thus, being a Mains-Powered Device is not an option. Besides, an Energy-Harvesting Device is not an option if no suitable form of ambient energy source is available at the device's location.
- **Costs:** Replacing or recharging the power source is an option but has a cost associated with it, especially if the device is located in a remote or inaccessible location. For your use case, it makes economically and physically sense to do this in the time frame which allows the device to sustain its functionality.
- **Uptime:** You want to minimize the periods where the device is not operating because of power source renewal.

Solution: Use a replaceable or rechargeable source of energy to power the device. Implement a notification mechanism that informs you when the power source is nearly empty. Replace or recharge the power source when needed.

Result: Using a replaceable or rechargeable power source is a common occurrence in today's devices. Increasingly energy efficient electronic components now allow manufacturers to build devices which run on one charge for weeks to months, if not years. For the rest of this text, we equate a Period Energy-Limited Device with using batteries, as they are common in the domain of IoT. But for example, fuel for a generator is another valid form of a power source for a Period Energy-Limited Device.

Fig. 2 shows the lifecycle of a Period Energy-Limited Device. It can be roughly divided into three phases: Most of the time, the device operates normally and, thus, *discharges* the power source, as shown at the bottom. Once a certain threshold is reached, the device starts to *notify*, as shown at the top left. Then, the depleted power source is *renewed*, as shown at the top right, before the cycle begins again.

Batteries come in different forms and sizes and are renewable in two ways. The first way to renew power for a Period Energy-Limited Device is to replace depleted batteries with full ones. The replacement battery is either a new non-rechargeable battery or a recharged battery. In this case, it makes no difference to the device if the battery is rechargeable or not. If you recharge the battery, it happens outside of the device through a separate charger. Integrating this replacement mechanism into a device is straightforward. It requires a connector to which you attach the battery. An optional compartment housing this connector offers protection for the battery and the device internals from outside influences. The second way to renew the battery is to allow it to be recharged inside the device. This requires integrating a charging circuit into the device. When the battery is empty, you connect another energy source to the device to recharge the battery, for example, a power bank. Alternatively, you bring the device near to mains power where you can plug in a power supply. The complexity of the charging circuit varies depending on the type of battery and the desired recharge time. A slow charge circuit is simple because it cannot damage the battery and thus requires no end-of-charge detection. A fast charge circuit has to detect end-of-charge through voltage or temperature to prevent overcharging the battery. In this case, the battery has to be rechargeable but not replaceable. If it is rechargeable and not replaceable, replacing the battery when it malfunctions becomes difficult, but it allows for a tighter integration and closed housing. Depending on the intended use case of the device, you have to take care to shield it from its environment. Dust or waterproof battery compartments offer protection from outside elements. For integrated rechargeable batteries, nothing but the charging contact has to be accessible from the outside. This further prevents environmental factors of deteriorating the device.

Since the power renewal of the Period Energy-Limited Device requires another entity to act, the device needs a notification mechanism to trigger power source renewal, as shown at the top left in Fig. 2. If the device sends out messages, adding the battery status to these messages is one way to inform others about the device's battery status. Besides, a repeating light or sound indicating low energy is another option. You have to choose the notification threshold to allow time for power source renewal before it runs out to minimize downtime.

Benefits:

- **Independence:** The device is independent of the grid and of its environment. It has power regardless of power outages or bad weather as long as you replace its energy source in time.
- **Lifetime:** The power source does not limit the lifetime of the device if it is replaceable.
- **Cost:** The costs for the device itself and for its installation are low. A battery connector and compartment or a charging circuit do not add high costs and wires are not required.

Drawbacks:

- **Lifetime:** The power source limits the lifetime of a device if it is a rechargeable but not replaceable battery because aging batteries deteriorate with time and batteries have a maximum charge cycle count. This is not a problem if the maximum number of charge cycles allows the device to run until its intended end of life. Otherwise, making the battery replaceable solves this problem.
- **Costs:** You need to replace or recharge the power source in regular intervals which increase maintenance costs. Also being an Energy-Harvesting Device or a Normally-Sleeping Device increases the interval length.
- **Durability:** The device has to support replacing or recharging the power source which requires access to the power source or the recharging contacts. If the device exposes these points to the environment they deteriorate in harsh conditions. One option is to build these points dust or waterproof but doing this does not offer full protection and increases costs. Wireless charging is another option which allows sealing the device.

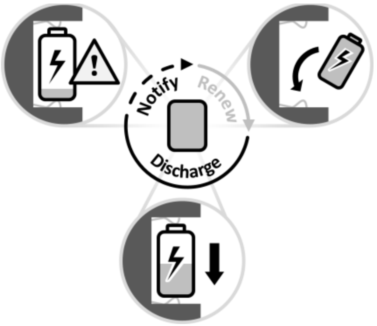


Figure 2. Sketch of the Period Energy-Limited Device Pattern.

- **Uptime:** The device is not operational when you replace its power source instead of recharging it. Making the power source rechargeable besides being replaceable is one way to guarantee uptime. Another option is to have two power sources in the device, where it uses one as a backup while you replace the other one.

Related Patterns:

- **Energy-Harvesting Device:** One way to increase the time needed between power source replacements or recharging is energy harvesting. An example is adding a small solar cell, which trickle charges the battery.
- **Normally-Sleeping Device:** A Normally-Sleeping Device saves energy when the device is not needed. This can increase the interval length between power source replacement or recharging for Period Energy-Limited Devices.
- **Mains-Powered Device:** A Mains-Powered Device can also be a Period Energy-Limited Device if it uses a battery as a backup in case of power outage.

Known Uses: One example of a Period Energy-Limited Device is the *Flic Wireless Smart Button*. It claims to last one year or more on its replaceable battery [13]. A similar device, *Logitech's POP Home Switch*, claims up to 5 years battery life from its replaceable battery [14]. *Sen.se's ThermoPeanut* is a wireless temperature sensor with a replaceable battery which lasts up to 6 months, depending on the frequency of sensor reading [15]. Another example is the *Nest Learning Thermostat,* which comes with a rechargeable lithium-ion battery [16]. The *Roost Smart Battery* is a replacement battery, which adds WiFi connectivity to smoke detectors. It notifies users via an app when the alarm is triggered or the battery runs low [17]. Besides, some Mains-Powered Devices are also Period Energy-Limited Devices as they use batteries as a backup to increase their resilience against power outages. Examples include the Device Gateways from *SmartThings*, *Essence*, or *Afero*. They either include a backup battery or offer connection options for external batteries [18][19][20].

B. *Energy-Harvesting Device*

Aliases: Ambient Energy, Event Energy-Limited, Event-Based Harvesting

Context: You have a device that needs to be powered. The device needs only a small amount of energy to function. Besides, your use case dictates a specific location for this device which restricts available energy sources. For example, the device has to be mobile, wearable, or in a remote location.

Problem: You need to power a device with very little power needs. The device is mobile or located in a remote place. Its environment is stable and predictable.

Forces:

- **Location:** The device has to be mobile or is located at a remote place. Thus, it cannot be a MAINS-POWERED DEVICE.
- **Effort:** Replacing or recharging a battery in frequent intervals is too much effort or not possible at all. Thus, using a PERIOD ENERGY-LIMITED DEVICE is not an option.
- **Energy Requirements:** The device needs very little energy to function.
- **Lifetime Energy Requirements:** The device needs more energy over its lifetime than current batteries can provide in a reasonable form factor without being replaced or recharged. Thus, using a LIFETIME ENERGY-LIMITED DEVICE is not an option.

Solution: Integrate an energy harvesting component, such as a solar cell, into the device. Use it to turn the energy available in the device's surroundings into power for the device. Use components and technologies optimized for low-power usage to make the most of the harvested energy.

Result: An ENERGY-HARVESTING DEVICE transforms ambient energy into electrical energy, as depicted in Fig. 3. Ambient energy can be in form of radiant energy (solar, infrared, radio-frequency), thermal energy, mechanical energy, or biomechanical energy. Each of these energy forms comes with its own benefits and drawbacks that have to be taken into account for each use case separately.

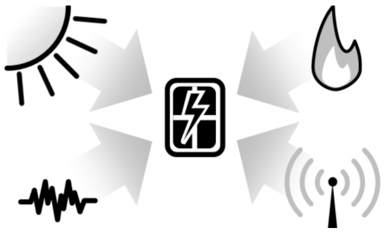


Figure 3. Sketch of the Energy-Harvesting Device Pattern.

Radiant energy in form of sunlight or other light sources is a common source of energy for ENERGY-HARVESTING DEVICES. Miniature solar modules are able to harvest enough energy, even from indoor lights, to perpetually transmit a measurement a few times per hour. But especially when using sunlight, it has to be taken into account that it is only available for a limited time each day. Another form of radiant energy, radio-frequency, is produced by the many wireless communication technologies we use today and can also be harvested. Because it is purposely generated and heavily regulated, it is more predictable than other forms of ambient energy. However, to be usable, a sufficient level of energy density is required in the environment which might only be given in more populated areas. Mechanical energy can also be harvested. For example, a switch may generate enough energy when activated to be able to send several radio telegrams. Another example is a thermoelectric generator which is able to collect and transform thermal energy in form of temperature differences into electricity.

The availability of each of these forms of ambient energy depends on the environment of the use case. Not all forms might be available in all locations and the available energy might be too small to power a particular device. Besides, mobility has to be taken into account. If the device is fixed, then the availability of ambient energy can be measured and is fairly predictable. If the device is mobile, then the form and amount of available ambient energy can fluctuate widely.

Even though it might only supply a very small amount of energy, ambient energy can be used to power very energy efficient circuits and sensors and to transmit and receive small messages. An ENERGY-HARVESTING DEVICE can be powered directly if it uses very energy efficient components, but in many cases, the harvested energy will not be enough for sustained operation. In such cases, the ambient energy can be used to trickle charge a battery or capacitor. Once sufficient energy is collected, the device can then turn on and use it for a short period of operation. Another use is to supplement PERIOD ENERGY-LIMITED DEVICES to increase the intervals between recharging.

As the harvested power is often so small, it is necessary for the device to use technologies which are optimized for ultra-low energy. This includes using components, such as microchips or sensors, which are very energy efficient. It also includes using communication technologies, such as wireless modules and even protocols and payload formats, which are optimized for ultra-low energy. Often, technologies are specifically created for this in mind, for example, the ISO/IEC 14543-3-10:2012 standard. But there are also examples of existing technologies, which have been adapted to be more energy-saving, such as IEEE 802.15.4, 6loWPAN, or CoAP.

Benefits:

- **Independence:** The device is independent of the electrical grid. Besides, it can be flexibly positioned because it does not require any wire.

- **Perpetual Energy:** Devices with very low energy requirements can be powered for as long as the energy harvesting components do not fail.
- **Cost:** The total cost of ownership of Energy-Harvesting Devices, which includes installation, operation, and management costs, is low. No cables have to be added during installation and battery replacement or recharging are either reduced in frequency or not necessary at all. Besides, the power used by the device is also free. Because there are no special infrastructure requirements, retrofitting an Energy-Harvesting Device is also easy.
- **Maintenance:** Maintenance can be reduced or is not necessary at all. This is especially beneficial if the device is located in inaccessible areas or if a lot of devices are operated.
- **Environmental Impact:** Energy-Harvesting Devices have a low environmental impact. The energy they harvest is freely available and energy wasting is not a problem. They also do not produce as much hazardous waste in form of old batteries as Period Energy-Limited Devices, but other components, including the energy harvesting components, might still be hazardous.

Drawbacks:

- **Dependence:** The device depends on the availability characteristics of the ambient energy source and its environment. These might be hard to accurately predict and control. If the environment changes it might no longer provide enough ambient energy for the device.
- **Energy:** Depending on the used technology, only small amounts of energy may be harvested from the environment. To get the most out of the available energy, high energy efficiency is necessary. This requires the device to consume very little energy during idle times, which can be achieved by making it a Normally-Sleeping Device. It also requires the device to be efficient when it is awake, which can be done by using low power components and technologies.
- **Fragility:** Depending on the form of ambient energy used, the components needed for energy harvesting might be fragile and not suited for all environments.

Related Patterns:

- **Period Energy-Limited Device:** Energy harvesting can be used to extend the intervals between recharging or replacing the energy source of a Period Energy-Limited Device.
- **Normally-Sleeping Device:** Energy harvesting may power Normally-Sleeping Devices if the harvested energy is only enough for short bursts of activity.

Known Uses: A common use of energy harvesting is found in devices which use passive RFID for communication. Here, the RF signal generated by the reader also powers the device [21]. Researchers are working on extending the capabilities of RFID powered device beyond responding with fixed data. An example is the *Wireless Identification and Sensing Platform* (WISP). It allows fully programmable 16-bit microcontrollers with attached sensors to be powered by RFID [22]. A device using WISP is the WISPCam, a passive RFID powered camera tag [23]. *EnOcean* created a patented wireless communication technology that is now standardized as ISO/IEC 14543-3-10:2012. It uses kinetic motion, solar, and thermal converters to create enough power for transmitting wireless signals. *EnOcean* also produces modules and products (mainly in the home automation sector) that utilize this technology. Many other companies have licensed the *EnOcean* technology and offer products [24][25]. *Freevolt* is another technology that harvests energy for low power devices from radio frequencies produces by broadcast networks, such as 2g, 3g, 4g, WiFi, and digital TV. The *CleanSpace Tag* is an air quality sensor which uses this technology to generate perpetual power for its lifetime [26].

C. Normally-Sleeping Device

Aliases: Sleepy, Deep Sleep, Hibernate, Duty-cycled, Normally-Off

Context: You have a use case which comes with size, weight, cost, or energy restrictions. For example, this is the case when the use case needs mobility or wearability. You use devices optimized to fit these restrictions. These devices are Lifetime Energy-Limited Devices, Period Energy-Limited Devices, or Energy-Harvesting Devices.

Problem: You have a device with a limited energy supply. You want to minimize the power used by the device.

Forces:

- **Limited Energy:** Having an Always-On Device is not an option since the device has a limited power source.
- **Energy Saving:** Saving energy decreases costs and is good for the environment but leads to constraints.
- **Component Use:** The device does not use every component continuously. Turning them off when not needed saves energy. But if these components have long startup times, the responsiveness of the device suffers.
- **Communication:** Turning of the communication module when not needed saves energy. But doing this manually takes too much effort, especially for remotely placed or large amounts of devices.

Solution: Program the device to disable its main components when they are not needed. Leave a small circuit powered which reactivates the components after a predefined amount of time has passed or when an event occurs.

Result: A NORMALLY-SLEEPING DEVICE cuts power to its main components for long stretches of time, as shown in Fig. 4. Good candidates for saving energy among these components are wireless communication modules, as they drain large amounts of power. Thus, NORMALLY-SLEEPING DEVICES are not able to communicate during their off periods. Other components, from processing units to individual sensors or actuators, are also disabled to add to these energy savings.

One component has to be active continuously to wake up the device. A clock component is able to reactivate power to the other components after a predefined amount of time, shown as the first active period in Fig. 4. This time is either absolute, for example, every full hour, or relative, for example, 5 minutes after the last active period ended. Another way to reactivate the turned off components is on events, shown as the second active period in Fig. 4. One option to do this is a small circuit which monitors a sensor and reactivates power when it reaches a predefined threshold. Or a DEVICE WAKEUP TRIGGER can be used to create such an event.

Once reactivated, the device resumes normal operation, as shown at the bottom of Fig. 4. For example, it saves the current sensor values and reestablishes a connection to a backend server. It uploads its state and processes messages which are waiting for it on the server. After the device has finished this process it returns to the sleeping state until the next period of activity.

Benefits:

- **Efficiency:** The device is more energy efficient because it is active only when needed.
- **Longevity:** Sleeping for long periods of time saves energy. This increases the maximum lifetime of LIFETIME ENERGY-LIMITED DEVICES. Besides, it increases the interval length between replacing or recharging the power source in PERIOD ENERGY-LIMITED DEVICES.

Drawbacks:

- **Intermittent Connectivity:** Communication with the device is intermittent. When it is sleeping, other communication partners cannot reach it. A DEVICE SHADOW is one option to allow others to communicate with an eventually consistent version of the device. On the device itself, not every component has to be off when it is sleeping. An example is a sensor which keeps collecting measurements that need to be sent to the backend eventually. The device has to store these measurements in a queue and sends them later when it activates the next time.

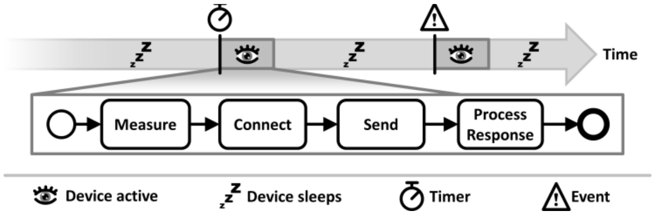


Figure 4. Sketch of the NORMALLY-SLEEPING DEVICE Pattern.

- **Timing:** In important cases, for instance for critical security updates, another component has to contact the device at once. Waiting for the NORMALLY-SLEEPING DEVICE to reconnect during its next activity window is not an option. A DEVICE WAKEUP TRIGGER is one way to get the NORMALLY-SLEEPING DEVICE to reconnect at once by creating an event that it listens to.
- **Energy:** Establishing a new connection for communication needs power. Sometimes it is more efficient to sustain an existing connection than creating a large number of new ones. This point depends on the chosen technology and the required communication frequency. You have to choose sleep schedules with this in mind.

Related Patterns:

- **ENERGY-HARVESTING DEVICE:** Devices which use energy harvesting as their source of power often also are NORMALLY-SLEEPING DEVICES. They sleep until they harvested the energy they need for a short period of activity.
- **DEVICE WAKEUP TRIGGER:** In situations when it is necessary to communicate with a NORMALLY-SLEEPING DEVICE outside of its regular communication windows, a DEVICE WAKEUP TRIGGER is one option. The DEVICE WAKEUP TRIGGER tells a disconnected device to reconnect at once.
- **PERIOD ENERGY-LIMITED DEVICE:** Being a NORMALLY-SLEEPING DEVICE extends the interval between replacing or recharging the power source in PERIOD ENERGY-LIMITED DEVICES.
- **LIFETIME ENERGY-LIMITED:** Being a NORMALLY-SLEEPING DEVICE extends the maximum lifetime of LIFETIME ENERGY-LIMITED DEVICES.
- **DEVICE SHADOW:** Using a DEVICE SHADOW allows other communication partners to retrieve the latest known state and to send commands to a currently sleeping device.

Known Uses: *Z-Wave* has so-called *sleepy devices*, which turn off to save energy and periodically wake up and reconnect. When reconnected, they inform other devices that they are listening for commands for the next seconds [27]. *Libelium's Waspmotes* support different operation modes to save power, including sleep and deep sleep modes which last from milliseconds to days. In these modes, they pause the main program and the microcontroller. Synchronous interrupts (periodic and relative programmed timers), or asynchronous interrupts (sensor readings or XBee activity) end these modes. Besides, they support a hibernate mode, where they cut power off from every part except the clock. The clock ends this mode after a predefined time with a synchronous interruption [28]. Other devices turn on for a brief moment if an event occurs. For example, the *Amazon Dash Button* turns on once a person presses the button. It connects to a WiFi network, places an order, and shuts off as soon as it receives a response [29]. The *PawTrax* pet tracker wakes up when it receives a text message, gets the current GPS position and returns it before it goes back to sleep. Besides, it has an option to return position data in set intervals [30].

VI. Summary and Outlook

Devices are a central point of any IoT system, as they link the physical with the digital world through their sensors and actuators. They are also a starting point when designing IoT systems because they are directly influenced by the particular use case and the environment. Their selection then further influences the design of the IoT system as it has to cater to the different device characteristics.

To help individuals to design IoT systems that work with different kinds of devices, we presented six IoT device Patterns. Three of them were described in more detail. One of the energy source Patterns, Period Energy-Limited Device, describes a device which uses a replaceable or rechargeable power source. This allows it to be mobile but also requires some maintenance. Another Pattern in this group, the Energy-Harvesting Device, explains how devices can use harvest ambient energy for their power needs. From the category of operating modes, a Normally-Sleeping Device can disable most of its components and sleep for some time to save energy. We also showed how these Patterns are interconnected, also with our previous Patterns.

In the future, we want to expand this selection of Patterns into a full IoT Pattern catalog and further refine their interrelations to form an IoT Pattern Language. This will also include new Patterns, which are concerned with device bootstrapping, device registration, communication between devices and platforms, data processing, and more.

Acknowledgment

This work was partially funded by the BMWi projects NEMAR (03ET4018B), SmartOrchestra (01MD16001F) and SePiA.Pro (01MD16013F).

References

[1] L. Reinfurt, U. Breitenbücher, M. Falkenthal, F. Leymann, and A. Riegg, “Internet of Things Patterns,” in *Proceedings of the 21st European Conference on Pattern Languages of Programs (EuroPLoP)*: ACM, 2016. in press.

[2] L. Reinfurt, U. Breitenbücher, M. Falkenthal, F. Leymann, and A. Riegg, “Internet of Things Patterns for Communication and Management,” LNCS Transactions on Pattern Languages of Programming, 2017. unpublished.

[3] C. Bormann, M. Ersue, and A. Keranen, “Terminology for Constrained-Node Networks,” IETF, 2014. [Online]. Available from: http://www.rfc-editor.org/rfc/pdfrfc/rfc7228.txt.pdf 2017.01.13

[4] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press, 1977.

[5] G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, Massachusetts: Addison-Wesley, 2004.

[6] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*. Wien: Springer, 2014.

[7] G. Meszaros and J. Doble, “Metapatterns: A Pattern Language for Pattern Writing,” *Third Pattern Languages of Programming Conference*: Addison-Wesley, 1996.

[8] N. B. Harrison, “The Language of Shepherding: A Pattern Language for Shepherds and Sheep,” in *Software patterns series, Pattern languages of program design 5*, D. Manolescu, M. Voelter, and J. Noble, Eds. 1st ed., Upper Saddler River, NJ: Addison-Wesley, 2006, pp. 507–530.

[9] N. B. Harrison, “Advanced Pattern Writing: Patterns for Experienced Pattern Authors,” in *Software patterns series, Pattern languages of program design 5*, D. Manolescu, M. Voelter, and J. Noble, Eds. 1st ed., Upper Saddler River, NJ: Addison-Wesley, 2006, pp. 433–452.

[10] T. Wellhausen and A. Fießer, “How to write a pattern?: A rough guide for first-time pattern authors,” *Proceedings of the 16th European Conference on Pattern Languages of Programs*: ACM, 2012.

[11] V.-P. Eloranta, J. Koskinen, M. Leppänen, and V. Reijonen, *Designing distributed control systems: A pattern language approach*. Hoboken, NJ: Wiley, 2014.

[12] S. Qanbari *et al.*, “IoT Design Patterns: Computational Constructs to Design, Build and Engineer Edge Applications,” *Proceedings of the First International Conference on Internet-of-Things Design and Implementation (IoTDI)*: IEEE, pp. 277–282, 2016.

[13] Shortcut Labs, *Flic: The Wireless Smart Button.* [Online]. Available from: https://start.flic.io/ 2017.01.13

[14] Logitech, *POP Home Switch Simple smart home control for the whole family.* [Online]. Available from: http://www.logitech.com/en-us/product/pop-home-switch 2017.01.13

[15] Sen.se, *ThermoPeanut.* [Online]. Available from: https://sen.se/peanut/thermo/ 2017.01.13

[16] Nest, *Nest Learning Thermostat - Install & Explore.* [Online]. Available from: https://nest.com/thermostat/install-and-explore/ 2017.01.13

[17] Roost, *Roost Wi-Fi battery for smoke and CO alarms.* [Online]. Available from: http://www.getroost.com/product-battery 2017.01.13

[18] SmartThings, *Architecture.* [Online]. Available from: http://docs.smartthings.com/en/latest/architecture/index.html 2017.01.13

[19] Essence, *WeR@Home Installation Guide* 2017.01.13

[20] Afero, “Hub Secure Hub Product brief,” 2016. [Online]. Available from: https://developer.afero.io/assets/HubProductBrief.pdf 2017.01.13

[21] R. Want, “An introduction to RFID technology,” IEEE Pervasive Computing, vol. 5, no. 1, pp. 25–33, 2006.

[22] J. R. Smith, *Wireless Identification Sensing Platform (WISP).* [Online]. Available from: http://sensor.cs.washington.edu/WISP.html 2017.01.13

[23] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith, “WISPCam: A Battery-Free RFID Camera,” *2015 IEEE International Conference on RFID (RFID)*, pp. 166–173, 2015.

[24] EnOcean, *EnOcean – The World of Energy Harvesting Wireless Technology*. [Online]. Available from: https://www.enocean.com/fileadmin/redaktion/pdf/white_paper/WhitePaper_Getting_Started_With_EnOcean_v1.0.pdf 2017.01.13

[25] EnOcean, *Energy Harvesting Wireless Power for the Internet of Things*. [Online]. Available from: https://www.enocean.com/fileadmin/redaktion/pdf/white_paper/White_Paper_Internet_of_Things_EnOcean.pdf 2017.01.13

[26] drayson, *RF Energy Harvesting for the Low Energy Internet of Things* 2017.01.13

[27] SmartThings, *Z-Wave Primer*. [Online]. Available from: http://docs.smartthings.com/en/latest/device-type-developers-guide/z-wave-primer.html 2017.01.13

[28] Libelium, “Waspmote Technical Guide,” 2016. [Online]. Available from: http://www.libelium.com/downloads/documentation/waspmote_technical_guide.pdf 2017.01.13

[29] T. Benson, *How I Hacked Amazon’s $5 WiFi Button to track Baby Data — Medium*. [Online]. Available from: https://medium.com/@edwardbenson/how-i-hacked-amazon-s-5-wifi-button-to-track-baby-data-794214b0bdd8 2017.01.13

[30] PawTrax, *Welcome to PawTrax*. [Online]. Available from: http://www.pawtrax.co.uk/ 2017.01.13