I·A·A·S

Institute of Architecture of Application Systems

Internet of Things Patterns for Devices: Powering, Operating, and Sensing

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:

```
@article{Reinfurt2017,
  author    = {Reinfurt, Lukas and Breitenb\{"u}ucher, Uwe and Falkenthal,
               Michael and Leymann, Frank and Riegg, Andreas},
  title     = {Internet of Things Patterns for Devices:
               Powering, Operating, and Sensing},
  journal   = {International Journal on Advances in Internet Technology},
  year      = {2017},
  pages     = {106--123},
  publisher = {IARIA}
}
```

The full version of this paper can be retrieved from
http://www.iariajournals.org/software

Universität Stuttgart
Germany

Internet of Things Patterns for Devices:
Powering, Operating, and Sensing

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—**A central part of the Internet of Things are devices. By collecting data about themselves and their environment using sensors, they provide the raw resources for later analytics stages. Based on the results of these analytics they can also act back on their environment through actuators. Depending on their use case, these devices come in all shapes and sizes, are placed in various environments, and often have to operate under constraints such as limited access to energy or requirements for mobility. All these factors have an impact on how they are supplied with energy, how they operate, and how they sense. In this paper, we describe the resulting types of energy supplies, operating modes, and sensing techniques as Internet of Things Patterns based on existing terminology and known implementations. We show that these patterns are interconnected with others and that they form the beginning of an Internet of Things Pattern Language, which allows readers to find and navigate through abstract solutions for often reoccurring problems.**

Keywords—*Internet of Things; Patterns; Devices; Constraints; Energy Supply; Operation Mode; Sensing.*

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. This work is an extension of our paper "Internet of Things Patterns for Devices" we presented at the ninth International Conferences on Pervasive Patterns and Applications (PATTERNS) 2017 [1]. We also have published other patterns in several categories in [2][3]. 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 their own problems.

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 are located not only in controlled indoor environments but also outside and in harsh conditions. For example, some of them are required to be mobile, are located off the power grid, or are placed under water.

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 [4]. These choices and other factors then also have an impact on how devices are able to operate and use their sensors.

Based on the terminology by Bormann et al. [4] and additional sources describing the application of IoT devices in real world scenarios, we created eight patterns for IoT devices with different energy sources, operation modes, and sensing techniques. 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. Three of these patterns have been described in detail in [1], namely PERIOD ENERGY-LIMITED DEVICE, ENERGY-HARVESTING DEVICE, and NORMALLY-SLEEPING DEVICE. This paper expands on [1] by adding detailed descriptions of the other three patterns, which where only shortly summarized in [1]. Moreover, we present two new patterns for SCHEDULE-BASED SENSING and EVENT-BASED SENSING.

The rest of this paper is structured as follows: Section II provides a short overview of previous work related to patterns in general and to our IoT Patterns. Section III briefly summarizes our understanding of patterns, our pattern format, and our previously published IoT Patterns. Section IV introduces the eight IoT Patterns for devices in three categories: The four energy supply type patterns ALWAYS-ON DEVICE, PERIOD ENERGY-LIMITED DEVICE, LIFETIME ENERGY-LIMITED DEVICE, and ENERGY-HARVESTING DEVICE, the two operation mode patterns ALWAYS-ON DEVICE and NORMALLY-SLEEPING DEVICE, and the two sensing patterns SCHEDULE-BASED SENSING, and EVENT-BASED SENSING. We also show how they are connected among themselves and to the already presented IoT Patterns. Section V describes these eight patterns in detail following the pattern format described in Section III. Finally, Section VI provides a summary and an outlook on our planned future work.

II. RELATED WORK

The pattern concept was first introduced by Alexander et al. in the architecture domain [5]. Since then, the concept has been applied in other domains. Examples from IT include the Messaging Patterns by Hohpe et al. [6] or the Cloud Computing Patterns by Fehling et al. [7]. There has also been work on the pattern writing process itself [8][9][10][11]. Others are working on making abstract patterns more usable by linking them to technology specific patterns [12] or to solutions implementations [13][14] and, thus, building solutions languages [15].

We presented our first five IoT Patterns, DEVICE GATEWAY, DEVICE SHADOW, RULES ENGINE, DEVICE WAKEUP TRIGGER, and REMOTE LOCK AND WIPE [2]. We later added more patterns in [3]. These patterns are not concerned with IoT devices themselves but do already mention the terminology by Bormann et. al [4]. 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 [16]. 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 [17]. Another publication collected other existing patterns, many just blog posts without much substance, and categorized them into a pattern language [18]. Many of these patterns are just one or two sentences long and they all are lacking the interconnections between them.
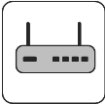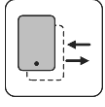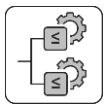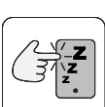
III. IOT PATTERNS OVERVIEW

The IoT Patterns presented in this paper and our previous work follow the ideas of Alexander [5] and others [8][9][10][11]. As described in more detail in [2][3], 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 [2][3], but, in short, is made up of the following elements: The **Name**, **Icon**, and **Aliases** help to identify the pattern. The icons are also intended as a visual summary of the core solution and should be usable in architecture diagrams to represent the patterns. A brief summary section gives an overview. 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 **Solution Details** section gives more de-

tails on the solution. Other relevant patterns are listed as **Related Patterns**. Existing products, which implement the pattern and are used as sources for the pattern writing process, are summarized under **Known Uses**.

Table I provides an overview of our earlier patterns, including a short summary of the problems (P) they are solving and a brief description of how they solve it (S). These and future patterns are available online in shortened versions at http://www.internetofthingspatterns.com.

TABLE I. OVERVIEW OF OUR PREVIOUS IOT PATTERNS

DEVICE GATEWAY [2]	**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 the communication technology of the network and vice-versa.
DEVICE SHADOW [2]	**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 representation. Synchronize the virtual representation with the actual device state when the device is online.
RULES ENGINE [2]	**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 to 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 [2]	**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 [2]

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 features. Execute triggered functions as soon as the device receives them and provide an acknowledgment to the backend.

DELTA UPDATE [3]

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 [3]
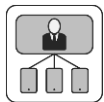
P.: You want to manage a large number of devices remotely.

S.: Set up a management server on the backend. Install management clients on 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 [3]

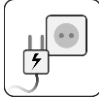**P.:** You need to use wireless communication in a crowded area, but you cannot use the crowded radio spectrum.

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. IoT Patterns for Devices

In this paper, we add eight IoT Patterns for devices. This section presents an overview of all of them in Table II, Table III, and Table IV. Related patterns are organized into three groups. The first group, *Energy Supply Types*, is summarized in Table II 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. However, 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 III 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 II. OVERVIEW OF THE NEW IOT PATTERNS CONCERNED WITH DEVICE ENERGY SOURCES

Energy Supply Types	
MAINS-POWERED DEVICE (Section V.A)	**P.:** You need to power a stationary device, which requires a lot of energy. **S.:** Connect the device to mains power. Transform higher voltages to a low voltage, which you can use. Convert alternating current to direct current and filter it to get smooth voltage.
PERIOD ENERGY-LIMITED DEVICE (Section V.B)	**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 (Section V.C)	**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.
ENERGY-HARVESTING DEVICE (Section V.D)	**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.

The third group, *Sensing*, shown in Table IV, contains patterns which describe different sensing techniques. These are influenced by the energy supply type and operation mode used, as well as by other forces, for example the preexisting knowledge about the domain. For example, EVENT-BASED SENSING is a good option for NORMALLY-SLEEPING DEVICES, but requires the events that should be observed to be known and understood beforehand.

PERIOD ENERGY-LIMITED DEVICE, ENERGY-HARVESTING DEVICE, and NORMALLY-SLEEPING DEVICE have been presented in detail before in [1], while the other energy supply and operation mode patterns where only briefly summarized. This work adds detailed descriptions of these patterns as well as the new sensing patterns.

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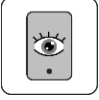CONCERNED WITH DEVICE OPERATION MODES

Operation Modes	
ALWAYS-ON DEVICE (Section V.E)	**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.
NORMALLY-SLEEPING DEVICE (Section V.F)	**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.

TABLE IV. OVERVIEW OF THE NEW IOT PATTERNS
CONCERNED WITH SENSING

Sensing	
SCHEDULE-BASED SENSING (Section V.G)	**P.:** You do not know what kinds of events you are looking for in your measurements or you need a general overview of the trend of the phenomenon that you are measuring.
	S.: Define a schedule for sensor reading which fits with your use case's requirements. Program the device to power up its sensors and read their values according to this schedule. Power down the sensors after reading.
EVENT-BASED SENSING (Section V.H)	**P.:** You have a use case where you are interested in irregularly occurring events that are represented in sensor values. Reading the sensor in regular intervals is wasteful because those events occur rarely and the other values are of no interest to you. Besides, it is possible that you miss such an event if it falls between two sensor readings.
	S.: Define the events that are of interest to you in the form of sensor values. Use low-power sensors and power them continuously. Program low energy comparator circuits to read those sensors and watch for the events. Only propagate measurements to the device if a sensor value triggers such a comparator circuit.

These patterns do not exist in a vacuum. They are connected among themselves and to the patterns we previously presented [2][3]. 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).
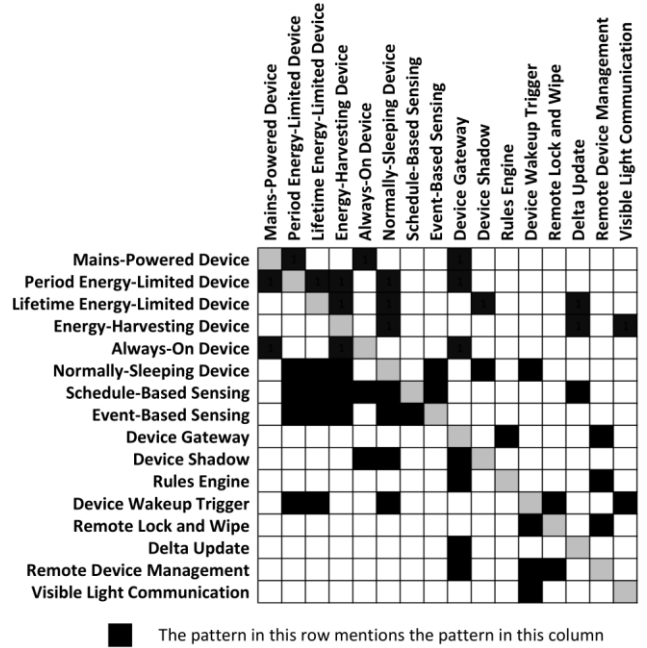

Figure 1. Connections between IoT Patterns.

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.

As the applicability of the IoT Patterns for devices 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 the IoT device patterns presented in this paper in more detail. We start with the *Energy Supply Types* category, followed by the *Operation Modes*, and then the *Sensing* category.

A. MAINS-POWERED DEVICE
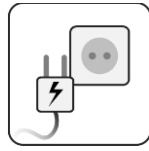
Some devices require a lot of power or are stationary. Power them by connecting them to mains power.

Aliases: Mains-operated

Context: You have a device that needs to be powered. It may be an ALWAYS-ON DEVICE that has to run continuously to fulfill its intended function. It may also be a device that does a lot of local processing which requires large amounts of energy.

Problem: You need to power a stationary device, which requires a lot of energy over its lifetime.

Forces:
- **Energy Requirements:** The device needs a large amount of energy over its lifetime. Making it a LIFETIME ENERGY-LIMITED DEVICE is not an option because the device needs more energy over its lifetime than current batteries can provide in a reasonable form factor without being replaced or recharged. Making it an ENERGY-HARVESTING DEVICE is not an option because ambient energy does not deliver the required power.
- **Environmental Constraints:** The environment of the device may have some constraints, such as not having a suitable source of ambient energy, thus making it not possible to use an ENERGY-HARVESTING DEVICE.
- **Maintenance:** Making it a PERIOD ENERGY-LIMITED DEVICE is not an option because replacing or recharging a battery in frequent intervals is too much effort.
- **Mobility:** The device does not have to be mobile.
- **Infrastructure:** Mains power is available in many places, but voltages and outlet designs vary depending on location.

Solution: Connect the device to mains power. Transform higher voltages to a lower voltage that you can use. Convert alternating current to direct current and filter it to get smooth voltage.

Solution Details: A MAINS-POWERED DEVICE requires a bit more than adding a cable to a power outlet. As many devices operate on lower voltages than provided by the power grid, a transformer, which converts high voltages to lower voltages, is required. Moreover, often alternating current (AC) has to be converted to direct current (DC) using a rectifier.
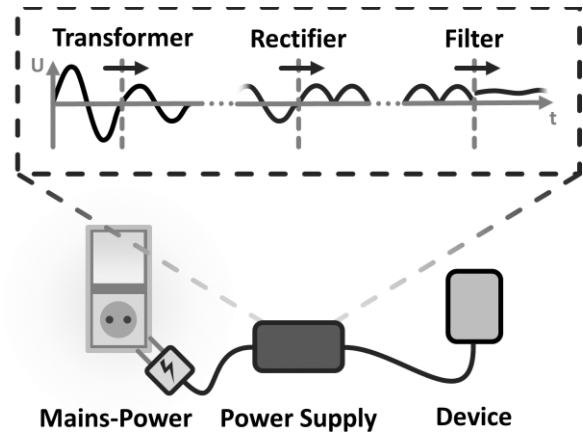

Figure 2. Sketch of the MAINS-POWERED DEVICE pattern.

It should also be filtered to provide a smooth voltage that does not damage components. Fig. 2 shows this process with an external power supply. Besides, a regulator might be necessary to receive a constant voltage level.

All these components have to be added to the device, either externally or internally. Both options have different benefits and drawbacks. An external power supply increases portability as the weight and size of the device can be kept small. The device can be used with different adapters to connect it to different power sources and if an adapter fails, it can be easily replaced. Besides, heat and electrical noise production is reduced and kept away from the device itself. In addition, production and inventory is simplified. The drawbacks of an external power supply are that it still uses some power if the device is not connected or powered, and possible confusion and compatibility problems when looking for an appropriate power supply for a device. An internal power supply reverses these benefits and drawbacks. These aspects should be taken into account when designing a MAINS-POWERED DEVICE.

Benefits:
- **Energy Requirements:** The device has large amounts of energy at its disposal allowing it to perform energy intensive tasks.
- **Maintenance:** There are no batteries that have to be recharged or replaced.

Drawbacks:
- **Dependence:** The device is dependent on the power grid. If there is a power outage, it will stop functioning. Additionally making the device a LIFETIME ENERGY-LIMITED DEVICE or PERIOD ENERGY-LIMITED DEVICE provides it with some backup power in such cases.
- **Mobility:** As the device is bound to the power grid, its mobility is severely limited.
- **Infrastructure:** If power lines are not already available it might be expensive and a lot of effort to build the required infrastructure.

Related Patterns:

- **LIFETIME ENERGY-LIMITED DEVICE:** A built-in battery can be used as backup power in case of a power outage.
- **PERIOD ENERGY-LIMITED DEVICE:** A rechargeable or replaceable power source can be used to provide backup power in case of a power outage. The MAINS-POWERED DEVICE can recharge it once the power is up again.
- **NORMALLY-OFF:** A MAINS-POWERED DEVICE might also be a NORMALLY-OFF DEVICE to reduce its energy consumption.
- **ALWAYS-ON:** A MAINS-POWERED DEVICE can be an ALWAYS-ON DEVICE if it is required.

Known Uses: Common examples for MAINS-POWERED DEVICES are those that offer functionality that requires larger amounts of energy or that is used very frequently. Examples are *Philips Hue* lights [19], the *Xiaomi Mi Air Purifier 2* [20], the *Airboxlab Foobot* [21], or the many other home appliances that are increasingly connected to the internet. DEVICE-GATEWAYS are also often MAINS-POWERED DEVICES, like the *WeR@Home Hub*, which is powered with a power supply, but is also a PERIOD ENERGY-LIMITED DEVICE as it has a replaceable battery as backup to make it resilient against power outages [22]. Similarly, the *Afero Hub* is a MAINS-POWERED DEVICE but is also equipped with a port where a backup battery can be connected [23].

B. PERIOD ENERGY-LIMITED DEVICE
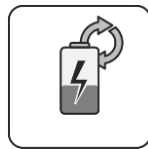
Devices that are mobile or located in remote places cannot rely on fixed infrastructure. When they require a fair amount of power, use a replaceable or rechargeable energy source and renew it regularly.

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 that 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 that 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.

Solution Details: 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 that 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. 3 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 re-

placement 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 the outside.

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 and 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. 3. 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. To minimize downtime, you have to choose the notification threshold to allow time for power source renewal before it runs out.

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.
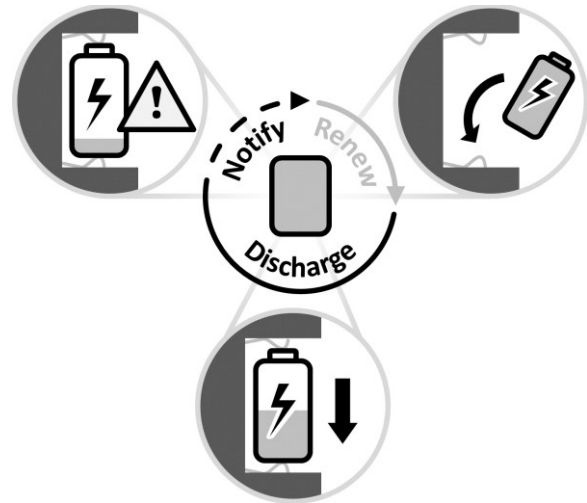

Figure 3. Sketch of the PERIOD ENERGY-LIMITED DEVICE pattern.

- **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 may 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 that allows sealing the device.
- **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 [24]. A similar device, *Logitech's POP Home Switch*, claims up to 5 years battery life from its replaceable battery [25]. *Sen.se's ThermoPeanut* is a wireless temperature sensor with a replaceable battery that lasts up to 6 months, depending on the frequency of sensor reading [26]. Another example is the *Nest Learning Thermostat,* which comes with a rechargeable lithium-ion battery [27]. 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 [28]. 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 [22][23][29].

C. LIFETIME ENERGY-LIMITED DEVICE
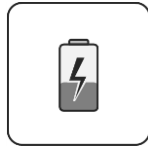
Devices that are mobile or located in remote places cannot rely on fixed infrastructure. When they require a small amount of power and need a robust construction, use a built-in power source.

Aliases: Non-replaceable Battery

Context: You have a device that needs to be powered. The device needs only a small amount of energy to function.

> **Problem:** 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.

Forces:
- **Mobility:** The device has to be mobile, thus, being a MAINS-POWERED DEVICE is not an option.
- **Location:** The device may be placed in a remote location far of the grid, where common infrastructure such as powerlines are not available. Thus, being a MAINS-POWERED DEVICE is not an option. The location may also provide no suitable form of ambient energy for an ENERGY-HARVESTING DEVICE.
- **Maintenance:** A PERIOD ENERGY-LIMITED DEVICE is not an option because replacing or recharging a battery in frequent intervals is too much effort.
- **Energy Requirements:** Being an ENERGY-HARVESTING DEVICE on its own is not an option, because it does not deliver the required power.

- **Lifetime:** You know the maximum lifetime of the device.
- **Ruggedness:** The device is intended to be used in environments where a rugged device enclosure is needed.

> **Solution:** Build an energy source into the device, which will last for the entire expected lifetime of the device.

Solution Details: For many applications, being MAINS-POWERED DEVICE or a PERIOD ENERGY-LIMITED DEVICE would be too complicated or simply not possible. For example, a sensor network might be located deep down at the bottom of the ocean, which would make these tasks very difficult. In such cases, using a built-in battery that has enough capacity to last for the expected lifetime of the device would be a better solution. Even in cases where mains power would be available or replacing a battery would be possible, using a built-in battery can make the design, manufacturing, and usage of a device much simpler.

Fig. 4 shows the different phases of a LIFETIME ENERGY-LIMITED DEVICE. During production, a charged battery is wired or soldered into the device. Then, during the normal operation of the device, the battery is slowly discharged. At some point, the battery level reaches a critical point where it may make sense to notify a responsible person about the imminent end of life of the device. Once the battery is fully discharged, the device has reached its end of life.

Benefits:
- **Simplicity:** A LIFETIME ENERGY-LIMITED DEVICE is simple and cheap to build. Only a battery has to be soldered into the device. There is no need for charging circuits, charging ports, access to the battery through the enclosure, or energy-harvesting components.
- **Maintenance:** There is no maintenance required for the built-in battery, since it cannot be replaced or recharged.
- **Costs:** There are no additional costs apart from the onetime costs of building the battery into the device.
- **Independence:** The device is independent of existing infrastructure.
- **Location:** The device is not bound to a particular location, as it does not depend on its environment.
- **Mobility:** The device can be freely moved, as it is not bound to the power grid or other environmental factors.
- **Ruggedness:** The device enclosure can be build more rugged since access to the battery is not required and all power related parts are internal.

Drawbacks:
- **Lifetime:** The overall lifetime of the device is limited by the energy supply. Once the energy supply is depleted the device becomes useless and the whole device has to be replaced which require some effort, especially if the device is located at a remote place.
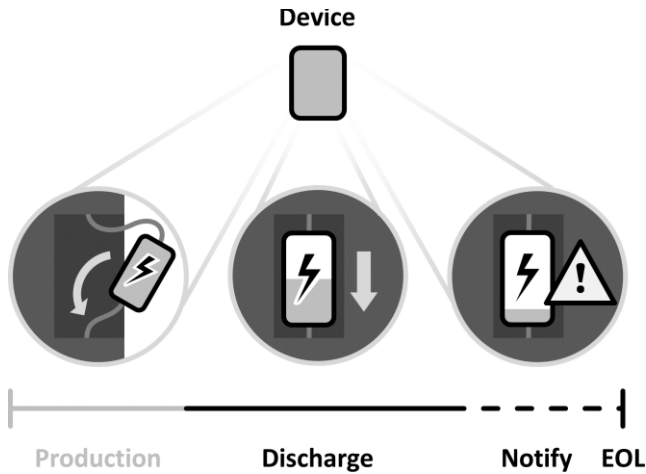
Figure 4. Sketch of the LIFETIME ENERGY-LIMITED DEVICE pattern.

The overall lifetime of the energy supply can be increased by also being an ENERGY-HARVESTING DEVICE to trickle charge the battery. However, this then also negates some of the benefits of a pure LIFETIME ENERGY-LIMITED DEVICE.

- **Energy Requirements:** The total available energy is limited and dictates the use cases where a LIFETIME ENERGY-LIMITED DEVICE makes sense. In general, using a built-in power source only makes sense for devices with very low energy requirements. Besides, to make the most out of its limited energy supply, the device should also be a NORMALLY-SLEEPING DEVICE that turns off most of the time.

Related Patterns:
- **ENERGY-HARVESTING DEVICE:** Energy-harvesting can be used to extend the lifetime of LIFETIME ENERGY-LIMITED DEVICES.
- **NORMALLY-OFF DEVICE:** LIFETIME ENERGY-LIMITED DEVICES may also be NORMALLY-OFF DEVICES to extend the lifetime of their energy source.
- **MAINS-POWERED DEVICE:** A MAINS-POWERED DEVICE can have an additional non-replaceable and non-rechargeable backup battery in case of a power outage.

Known Uses: *Amazon's Dash Button* has a built-in battery that is claimed to last 1000 button presses. It cannot be replaced or recharged once empty [30]. The *Tile Mate* is a small Bluetooth tag, which can be used to keep track of your belongings. The battery is guaranteed to last for a year after which a replacement Tile (which may include updated technology) can be automatically ordered for a reduced price [31]. The *Chipolo Plus* tag is sold in a similar fashion [32].

D. Energy-Harvesting Device
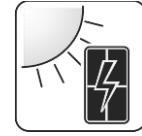
Devices that are mobile or located in remote places cannot rely on mains power infrastructure. When they require a small amount of power and the environment allows it, use energy-harvesting to gather its required power.

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.

Solution Details: An ENERGY-HARVESTING DEVICE transforms ambient energy into electrical energy, as depicted in Fig. 5. 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.
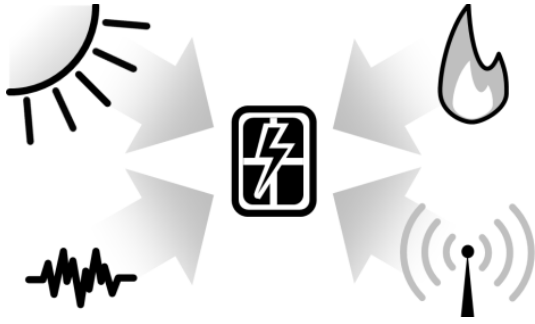
Figure 5. Sketch of the Energy-Harvesting Device Pattern.

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. However, 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 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 that 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 [33], 6loWPAN [34], or CoAP [35].

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 many 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 that use passive RFID for communication. Here, the RF signal generated by the reader also powers the device [36]. 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 [37]. A device using WISP is the *WISPCam*, a passive RFID powered camera tag [38]. *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 [39][40]. *Freevolt* is another technology that harvests energy for low power devices from radio frequencies produced by broadcast networks, such as 2g, 3g, 4g, WiFi, and digital TV. The *CleanSpace Tag* is an air quality sensor that uses this technology to generate perpetual power for its lifetime [41].

E. ALWAYS-ON DEVICE
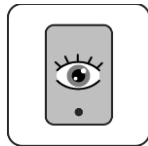
Some devices have to be available and responsive at all times. If the energy supply allows it, leave the device running and connected at all times.

Context: You have a MAINS-POWERED DEVICE or a device that has plenty of energy available, even though it is a PERIOD-ENERGY-LIMITED DEVICE, LIFETIME-ENERGY-LIMITED DEVICE, or ENERGY-HARVESTING DEVICE.

Problem: You have a device with an inexhaustible energy supply and need to have it available and responsive at all times.

Forces:
- **Energy Savings:** Saving energy is not the top priority in your situation.
- **Reachability:** The device has to be online and reachable all the time.
- **Reactivity:** The device has to react to commands instantly.
- **Functionality:** All of the device's functionality has to be available all the time.

Solution: Let the device be always on so that it is permanently connected to the network and in a state where it can receive and execute commands as soon as they arrive.

Solution Details: ALWAYS-ON DEVICES are simple. They do not have to consider a limited energy supply. Thus, they can be left running at all times with their full functionality enabled, as shown in Fig. 6. However, if their availability is a critical factor then additional precautions have to be taken to ensure availability despite possible problems. One problem might be malfunctioning or unavailable energy supply, for example, if the electrical grid is down due to a power outage. In such cases, a secondary backup power supply can be helpful, for example, in form or a backup battery. To help with a faulty communication mechanism, an ALWAYS-ON DEVICE could be equipped with multiple communication technologies to provide fallback mechanisms if necessary. To protect against failures in the device itself, multiple devices could be used to provide redundancy.

ALWAYS-ON DEVICES also benefit from an energy efficient design. As they are always running and connected, they use a lot of energy. By using energy efficient components to build these devices, the overall cost of running them can be reduced while still having them always on and connected.

Benefits:
- **Reachability:** The device will be reachable all the time as long as there are no network or power outages.
- **Reactivity:** The device will be able to react instantly because it does not have to wake up from any power saving operational states.

Drawbacks:
- **Power Consumption:** Being always on might lead to high power consumption, which could be reduced by using energy-efficient hardware.
- **Maintenance Cycles:** An ALWAYS-ON DEVICE that is also a PERIOD-ENERGY-LIMITED DEVICE or LIFETIME-ENERGY-LIMITED DEVICE will have shorter maintenance cycles than a NORMALLY-OFF DEVICE, which increases costs.
- **Feasibility:** An ALWAYS-ON DEVICE powered by Energy-Harvesting might not be feasible with current technology or only with a considerable cost.
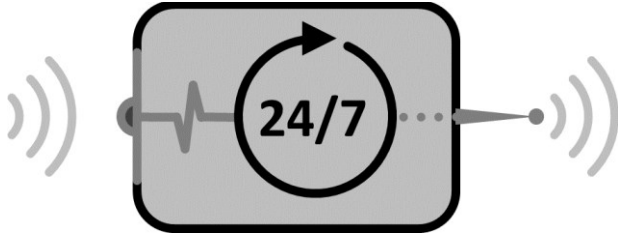
Figure 6. Sketch of the ALWAYS-ON DEVICE Pattern.

Related Patterns:
- **MAINS-POWERED DEVICE:** Devices powered by mains power are usually a good candidate for being ALWAYS-ON DEVICES since they do not have any limitations in the power available to them.
- **PERIOD-ENERGY-LIMITED DEVICE:** ALWAYS-ON DEVICES that are MAINS-POWERED DEVICES sometimes execute critical functionality and should have high availability. Such devices may also be PERIOD ENERGY-LIMITED DEVICES and use a backup power supply, such as a replaceable battery, to guard themselves against power outages.

Known Uses: Devices that use the *Z-Wave* standard and are mains-powered are referred to as listening devices. These devices keep on their receivers at all times to extend the *Z-Wave* mesh network by acting as repeaters for messages from other devices in the network [42]. Device Gateways, such as the *Samsung SmartThings Hub* [29] or the *Wink Hub* [43], are also often permanently on so that they can effectively fulfill their function. This is also enabled by being mains-powered.

F. NORMALLY-SLEEPING DEVICE
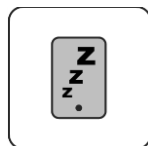
Devices with a limited supply of energy have to use it wisely. Disable most of the components of the device for long periods and only enable them when required.

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

Context: You have a use case that 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. However, if these components have long startup times, the responsiveness of the device suffers.
- **Communication:** Turning of the communication module when not needed saves energy. However, 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.

Solution Details: A NORMALLY-SLEEPING DEVICE cuts power to its main components for long stretches of time, as shown in Fig. 7. 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. 7. 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. 7. One option to do this is a small circuit that monitors a sensor and reactivates power when it reaches a predefined threshold. Alternatively, 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. 7. For example, it saves the current sensor values and reestablishes a connection to a backend server. It uploads its state and processes messages that 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 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.
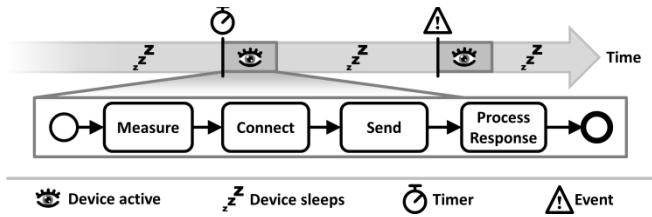
Figure 7. Sketch of the NORMALLY-SLEEPING DEVICE pattern.

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 that 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.

- **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 to which it listens.

- **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 that 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 [42]. *Libelium's Waspmotes* support different operation modes to save power, including sleep and deep sleep modes that 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 hibernation 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 [44]. 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 [45]. 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 [46].

G. SCHEDULE-BASED SENSING
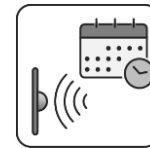
Some situations are not well understood or require a broad overview of measurable data points. Read the sensor based on a schedule. Turn it off between readings if the interval length allows it.

Aliases: Survey-Based Sensing

Context: You have a device with sensors built into it or attached to it and need to use these sensors to acquire measurements over long periods.

Problem: You do not know what kinds of events you are looking for in your measurements or you need a general overview of the trend of the phenomenon that you are measuring.

Forces:

- **Energy Requirements:** Leaving sensors running all the time allows them to record detailed measurements but uses more energy.

- **Initialization Delay:** Sensors can be turned off to save energy but need some time to initialize and produce sensible values when turned on again.
- **Size:** Measuring and sending many values over a long period enables detailed analysis but takes up storage space and communication bandwidth.
- **Fluctuation:** Sensor measurements fluctuate but for your use case, long-term trends are more important than accurately measuring every spike.
- **Knowledge:** With some knowledge about what you want to measure, you can get better results by applying techniques specifically tailored to your use case. However, in some cases, you do not precisely know what you are looking for.
- **Relevance:** In some cases, only specific events are of relevance to you, but in other cases, you may be more interested in general long-term trends.

Solution: Define a schedule for sensor reading which fits with your use case's requirements. Program the device to power up its sensors and read their values according to this schedule. Power down the sensors after reading.

Solution Details: SCHEDULE-BASED SENSING allows you to get a general overview of the values you are measuring and how they evolve over time. Fig. 8 shows the basic process. The real state of the phenomenon that you want to measure is a continuum of values, which are constantly fluctuating over time. SCHEDULE-BASED SENSING has the device turn on its sensors based on a schedule to take measurements. Some sensors will take a certain amount of time after being turned on to initialize or calibrate before they produce sensible readings. After a certain delay, they will produce a measurement, which is stored or communicated by the device. These values can be used to interpolate an approximation of the real state. After taking a measurement, the sensor can be turned off again to safe energy.

Note that depending on the interval length between measurements some important or interesting events may be missed, as shown by the spike in Fig. 8, which is not represented in the interpolated state. By choosing a suitable sampling rate and adjusting the schedule accordingly to decrease the interval length between measurements and thereby taking more measurements per unit of time, more sampling points can be generated which lead to a more precise approximation. However, this does not solve the problem that some events might still lie outside the measurement windows. Besides, at some point the interval length may be so short that turning the sensor off between readings is no longer possible (because of the initialization delay) and you lose the energy saving benefits of SCHEDULE-BASED SENSING.

Benefits:
- **Energy Requirements:** The sensors can be turned off and the device itself can go to sleep (see NORMALLY-SLEEPING DEVICE) in the time between the sensor readings to save energy.
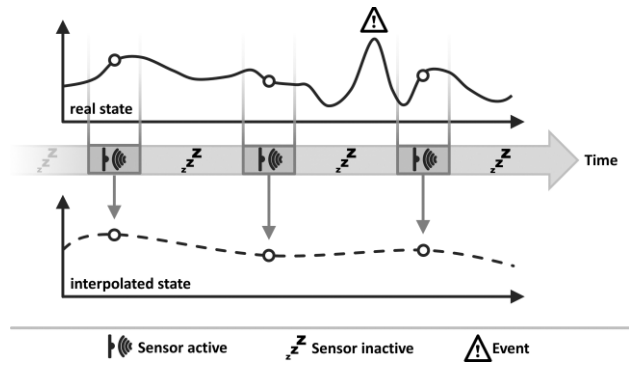

Figure 8. Sketch of the SCHEDULE-BASED SENSING pattern.

- **Knowledge:** You do not need to know anything about the values you are measuring.
- **Size:** The measuring schedule can be tuned to provide a good trade-off between accuracy, required storage space, and communication bandwidth

Drawbacks:
- **Fluctuation:** Depending on the selected schedule, the device will miss potentially interesting measurements. Use EVENT-BASED SENSING if you are interested in these spikes and know how to recognize them.
- **Size:** When a high accuracy is required and, thus, a short measuring interval is selected, many measurements will be created that have to be stored or communicated. Use DELTA UPDATE to communicate measurement values that have changed since the last measurement.
- **Relevance:** Even if you are interested in a long-term general overview of the measured event, a lot of the data collected may be not that relevant to your use case but still takes up a lot of storage space or communication bandwidth.
- **Energy Requirements:** The device has to wake up for each measurement even if nothing interesting happened, which consumes energy.

Related Patterns:
- **NORMALLY-SLEEPING DEVICE:** A NORMALLY-SLEEPING DEVICE can be programmed to wake up according to a schedule and take measurement, thus, implementing SCHEDULE-BASED SENSING.
- **ALWAYS-ON DEVICE:** Devices that are always running can also benefit from SCHEDULE-BASED SENSING as it can reduce the overall energy, storage space, and communication bandwidth required making such devices more efficient.
- **EVENT-BASED SENSING:** If certain well-defined events are more interesting than a general history of measurement values, EVENT-BASED SENSING is a good alternative to SCHEDULE-BASED SENSING.

- **ENERGY-HARVESTING DEVICE:** These devices can be programmed to take a measurement whenever they have gathered enough energy. Thus, they would implement SCHEDULE-BASED SENSING with an interval length that depends on environmental factors and, thus, can be irregular.
- **PERIOD ENERGY-LIMITED DEVICE:** The energy saved by turning the sensors off most of the time by using SCHEDULE-BASED SENSING can increase the time between replacing or recharging the batteries of these devices, which lowers their maintenance costs.
- **LIFETIME ENERGY-LIMITED DEVICE:** These devices can increase their maximum time to life by using SCHEDULE-BASED SENSING with long measurement intervals.

Variants:
- **PERIODIC SENSING:** A common variant of SCHEDULE-BASED SENSING uses a regular schedule with evenly spaced intervals to take the measurements.

Known Uses: Periodic sensor reading is mentioned in the *Libelium Waspmote* technical guide as an option for sensors with a high power consumption and for use cases where continuous monitoring to generate alarms is not required [44]. *NXP's MMA955xL* platform can take sensor readings at evenly spaced points in time [47]. Like many other microcontroller boards, the *Arduino* supports reading the current value of sensors attached to its analog pins in a loop [48].

H. EVENT-BASED SENSING

If you are interested in specific events, reading a sensor in regular interval can be wasteful. Besides, some events may be missed if they happen outside the measurement window. Implement a low-energy event detection circuit and only generate a measurement if an event is detected.

Context: You have a device with sensors built into it or attached to it and you need to use these sensors to acquire measurement and react to certain events.

Problem: You have a use case where you are interested in irregularly occurring events that are represented in sensor values. Reading the sensor in regular intervals is wasteful because those events occur rarely and the other values are of no interest to you. Besides, it is possible that you miss an event if it falls between two sensor readings.

Forces:
- **Energy Requirements:** Leaving sensors running all the time allows them to record detailed measurements but uses more energy.
- **Sampling:** The events you are interested in can be of short duration but measuring them requires a high enough sampling rate.
- **Size:** Measuring and sending many values over a long period enables detailed analysis but takes up storage space and communication bandwidth.
- **Knowledge:** You do not always have prior knowledge about what you want to measure, but if you have, you can get better results by applying techniques specifically tailored to your use case.
- **Relevance:** In some cases, only specific events are of relevance to you, but you do not need to collect all the other uninteresting measurements.

Solution: Define the events that are of interest to you in the form of sensor values. Use low-power sensors and power them continuously. Program low energy comparator circuits to read those sensors and watch for the events. Only propagate measurements to the device if a sensor value triggers such a comparator circuit.

Solution Details: EVENT-BASED SENSING requires that you know beforehand what events you are interested in and that you can formalize the specific features that distinguish these events from others. This allows you to implement comparator logic that can detect these events (see Fig. 9), which often can be done with very energy-efficient hardware circuits. These comparators change their output once they detect an event, which can be used to signal an interrupt pin of the device's processor. This tells the device to wake up, if necessary, and process the event at once. If the device should react to different events, it either requires multiple interrupt pins, one for each event, or the different comparator circuits have to be combined with an or gate before being put on the interruption pin. The value that triggered the comparator circuit and, thus, the interrupt should be stored in a register to allow the device to read it once it has started up.

Benefits:
- **Relevance:** You only get the measurement that you previously defined as relevant to you.
- **Size:** If the events that you are interested in do not occur very often, these measurements will not cost much storage space or communication bandwidth.
- **Energy Requirements:** The device can be turned off and is only turned on once something interesting happened, which can save a lot of energy.

Drawbacks:
- **Relevance:** You do not get a continuous history of measurement values. If you need such a history, use SCHEDULE-BASED SENSING.
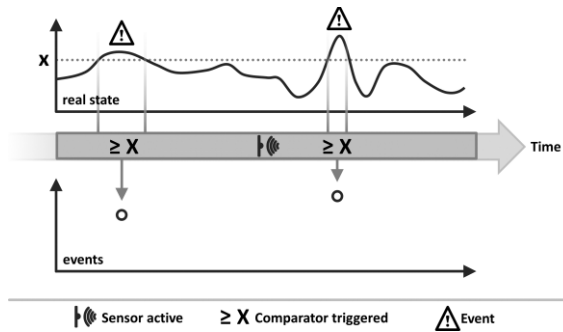
Figure 9. Sketch of the EVENT-BASED SENSING pattern.

- **Energy Requirements:** The sensors and the comparator components have to be running all the time to be able to catch the interesting events.

Related Patterns:
- **NORMALLY-SLEEPING DEVICE:** EVENT-BASED SENSING can be used to wake up a NORMALLY-SLEEPING DEVICE only when it is necessary to react to a certain event.
- **SCHEDULE-BASED SENSING:** If not only certain events are important, SCHEDULE-BASED SENSING is an alternative that provides regular sensor readings.
- **PERIOD ENERGY-LIMITED DEVICE, LIFETIME ENERGY-LIMITED DEVICE, and ENERGY HARVESTING DEVICE:** SCHEDULE-BASED SENSING is one option to lower the energy consumption of these devices by only powering the rest of the device on when an event is detected. This can increases their time between maintenance or their total time of life.

Variants:
- **CHANGE-BASED SENSING:** A common variant of the EVENT-BASED SENSING pattern is CHANGE-BASED SENSING. Here, new measurements are only propagated when the sensor value has changed by a certain significant amount, which can be configured according to the use case. Thus, messages are only sent for relevant changes. Unnecessary communication caused by small and insignificant fluctuations in the measured value is avoided.

Known Uses: *Libelium's Waspmotes* have interruption pins to which comparators can be connected. Once a comparator witnesses an event, its changing output send to the pin triggers the microprocessor to wake up and take further action [44]. *Fibaro's CO Sensor* has configuration settings to define the minimum change in sensor values that is required so that a message will be send [49]. *NXP's PCT2202* temperature sensor runs either in comparator or interrupt mode. In both modes, an alert pin is activated if the temperature remains higher than a configurable threshold for multiple readings [50]. The *Arduino*, like many other microcontroller boards, allows interrupts to be set on digital pins, to which sensors can be connected. These interrupts fire when the pin is low or high, or changes its value [51].

VI. CONCLUSION AND FUTURE WORK

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 the particular use case and the environment directly influence them. 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 eight IoT device patterns in three categories. The energy source type patterns describe different ways a device can be powered depending on factors such as energy requirement, mobility, or the environment. The operation mode patterns explain how a device might operate under different constraints such as limited energy. The sensing patterns describe different sensing techniques that devices can use.

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. We already added patterns for device bootstrapping and registration [52] but we will also add new patterns in several other categories, such as communication between devices and platforms, data processing, security, and more.

We also plan to do more work to support implementations based on patterns. For this, generic patterns, like the IoT Patterns presented in this work, can be refined into technology specific patterns [12]. These in turn can be linked to solution languages[53], which can provide practitioners with concrete guidance on how to implement a system based on patterns [15] and further provide implementations of the patterns [13][14]. These concepts can also be applied to IoT Patterns.

REFERENCES

[1] L. Reinfurt, U. Breitenbücher, M. Falkenthal, F. Leymann, and A. Riegg, "Internet of Things Patterns for Devices," *Proceedings of the Ninth International Conferences on Pervasive Patterns and Applications (PATTERNS) 2017*: Xpert Publishing Services, pp. 117–126, 2017.

[2] 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.

[3] 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.

[4] 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 2017.11.14

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

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

[7] 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.

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

[9] N. B. Harrison, "The Language of Shepherding: A Pattern Language for Shepherds and Sheep," in *Software patterns series, Pattern languages of program design 5*1st ed., Upper Saddler River, NJ: Addison-Wesley, 2006, pp. 507–530.

[10] N. B. Harrison, "Advanced Pattern Writing: Patterns for Experienced Pattern Authors," in *Software patterns series, Pattern languages of program design 5*1st ed., Upper Saddler River, NJ: Addison-Wesley, 2006, pp. 433–452.

[11] 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.

[12] M. Falkenthal *et al.,* "Leveraging Pattern Application via Pattern Refinement," *Proceedings of the International Conference on Pursuit of Pattern Languages for Societal Change (PURPLSOC)*, 2016.

[13] M. Falkenthal, J. Barzen, U. Breitenbücher, C. Fehling, and F. Leymann, "Efficient Pattern Application: Validating the Concept of Solution Implementations in Different Domains," (Englisch), International Journal on Advances in Software, vol. 7, no. 3&4, pp. 710–726, 2014.

[14] M. Falkenthal, J. Barzen, U. Breitenbücher, C. Fehling, and F. Leymann, "From Pattern Languages to Solution Implementations," *Proceedings of the Sixth International Conferences on Pervasive Patterns and Applications (PATTERNS 2014)*: IARIA, pp. 12–21, 2014.

[15] M. Falkenthal and F. Leymann, "Easing Pattern Application by Means of Solution Languages," *Proceedings of the Ninth International Conferences on Pervasive Patterns and Applications (PATTERNS) 2017*: Xpert Publishing Services, pp. 58–64, 2017.

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

[17] 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.

[18] G. S. Chandra, *Pattern language for IoT applications,*

[19] Philips, *Hue White and color ambiance Single bulb BR30.* [Online]. Available from: http://www2.meethue.com/en-us/p/hue-white-and-color-ambiance-single-bulb-br30/46677468941 2017.11.14

[20] Xiaomi, *Xiaomi Mi Air Purifier 2.* [Online]. Available from: https://xiaomi-mi.com/air-and-water-purifiers/xiaomi-mi-air-purifier-2/ 2017.11.14

[21] Airboxlab, *Install & setup your Foobot.* [Online]. Available from: http://help.foobot.io/hc/en-us/articles/204713582-Install-setup-your-Foobot 2017.11.14

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

[23] Afero, "Hub Secure Hub Product brief," 2016. [Online]. Available from: https://developer.afero.io/static/custom/files/HubProductBrief.pdf 2017.11.14

[24] Shortcut Labs, *Flic: The Wireless Smart Button.* [Online]. Available from: https://start.flic.io/ 2017.11.14

[25] 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 2017.11.14

[26] Sen.se, *ThermoPeanut.* [Online]. Available from: https://sen.se/store/thermopeanut/ 2017.11.14

[27] Nest, *Nest Learning Thermostat - Install & Explore.* [Online]. Available from: https://nest.com/thermostats/nest-learning-thermostat/tech-specs/ 2017.11.14

[28] Roost, *Roost Wi-Fi battery for smoke and CO alarms.* [Online]. Available from: http://www.getroost.com/product-battery 2017.11.14

[29] SmartThings, *Architecture.* [Online]. Available from: http://docs.smartthings.com/en/latest/architecture/index.html 2017.11.14

[30] Amazon Web Services, *AWS IoT Button.* [Online]. Available from: https://aws.amazon.com/iotbutton/ 2017.11.14

[31] Tile, *What is reTile?* [Online]. Available from: https://support.thetileapp.com/hc/en-us/articles/200550678-ReTile-How-do-I-renew-or-replace-my-Tile-after-it-expires-2017.11.14

[32] Chipolo, *Chipolo Plus Renewal Program.* [Online]. Available from: https://chipolo.net/pages/renewal 2017.11.14

[33] *IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs)*, 2016.

[34] *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, 2007.

[35] *The Constrained Application Protocol (CoAP)*, 2014.

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

[37] J. R. Smith, *Wireless Identification Sensing Platform (WISP).* [Online]. Available from: http://sensor.cs.washington.edu/WISP.html 2017.11.14

[38] 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.

[39] 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 2017.11.14

[40] 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 2017.11.14

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

[42] SmartThings, *Z-Wave Primer.* [Online]. Available from: http://docs.smartthings.com/en/latest/device-type-developers-guide/z-wave-primer.html 2017.11.14

[43] Wink, *Wink Hub.* [Online]. Available from: http://www.wink.com/products/wink-hub/ 2016.01.20

[44] Libelium, "Waspmote Technical Guide," 2016. [Online]. Available from:

http://www.libelium.com/downloads/documentation/waspmote_technical_guide.pdf 2017.01.13

[45] T. Benson, *How I Hacked Amazon's $5 WiFi Button to track Baby Data — Medium.* [Online]. Available from: https://blog.cloudstitch.com/how-i-hacked-amazon-s-5-wifi-button-to-track-baby-data-794214b0bdd8 2017.11.14

[46] PawTrax, *Welcome to PawTrax.* [Online]. Available from: http://www.pawtrax.co.uk/ 2017.11.14

[47] NXP, *MMA955xL Intelligent, Motion-Sensing Platform Hardware Reference Manual.* [Online]. Available from: http://www.nxp.com/docs/en/reference-manual/MMA955xLRM.pdf 2017.07.20

[48] Arduino, *analogRead.* [Online]. Available from: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/ 2017.11.14

[49] Fibaro, *CO Sensor.* [Online]. Available from: http://manuals.fibaro.com/co-sensor/ 2017.11.14

[50] NXP, *PCT2202: Ultra low power, 1.8 V, 1 deg. C accuracy, digital temperature sensor with I2C-bus interface.* [Online]. Available from: http://www.nxp.com/docs/en/data-sheet/PCT2202.pdf 2017.11.14

[51] Arduino, *attachInterrupt.* [Online]. Available from: https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/ 2017.11.14

[52] L. Reinfurt, U. Breitenbücher, M. Falkenthal, F. Leymann, and A. Riegg, "Internet of Things Patterns for Device Bootstrapping and Registration," in *Proceedings of the 22nd European Conference on Pattern Languages of Programs (EuroPLoP)*: ACM, 2017

[53] M. Falkenthal, J. Barzen, U. Breitenbücher, and F. Leymann, "Solution Languages: Easing Pattern Composition in Different Domains," International Journal on Advances in Software, vol. 10, no. 3 & 4, 2017. in press.