Automation in Construction 111 (2020) 103043

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Wireless sensor network for estimating building performance

Mario Frei[a,*], Chirag Deb[a], Ruben Stadler[a], Zoltan Nagy[b], Arno Schlueter[a]

[a] Chair of Architecture and Building Systems at ETH Zurich, Stefano-Franscini-Platz 1, CH-8093 Zurich, Switzerland
[b] Intelligent Environments Laboratory at the University of Texas at Austin, 301 E Dean Keeton Street St C1700, TX 78712-0273, USA

ARTICLE INFO

Keywords:
Wireless sensor networks
Remote monitoring
Open-source hardware
Sensors
Building performance
Indoor environment
In-situ measurements
Performance gap

ABSTRACT

Accurate building energy assessments are often limited by inaccurate assumptions about the buildings' properties and occupant behavior. Many assessment methods use assumptions instead of measured data because it is cumbersome, time-intensive, and cost-intensive to acquire up-to-date, on-site measured data. This work bridges this gap by presenting the design of an open-source, wireless sensor network (WSN) that is easy to program, manufacture, modify, and deploy. The version of the WSN presented here measures air temperature, relative humidity, supply and return temperature of hot water from the heating system and the radiators, heat flux through the walls and the windows, luminosity, oil flow, electricity, window opening times, and CO_2-concentration. The modularity of the WSN allows it to be extended with the addition of new types of sensors to the existing hardware framework. The data is streamed to an online database, allowing monitoring of the measurements in near real-time. The cost and performance of the sensor network are intended to be as good as or better than the existing solutions. The current cost for one temperature sensor node starts at 116 USD. The observed battery lifetime reaches up to a year for a five-minute sampling interval. The average installation time for a sensor node is 7 min. We describe the hardware architecture, software architecture, cost of the WSN, along with results from co-location tests, and a deployment in an occupied building.

1. Introduction

Buildings account for 35% of the total global final energy consumption [1]. If the necessary steps are not taken to improve energy efficiency, the energy demand in buildings is expected to increase by 50% by 2050 [1]. In the context of the Swiss residential sector, a 64% reduction in space heating demand has been identified as a key element of the Swiss Energy Strategy 2050 [2]. Given this strategy, energy retrofits are an essential step forward. To execute a successful energy retrofit requires rich knowledge of the existing conditions in buildings. However, such knowledge about existing buildings is often based on visual inspections and educated guesses by the assessors, due to a lack of other feasible means. This method of estimating the current condition of buildings is prone to errors and contributes to the deviation between calculated and actual energy consumption. Once the building information has been gathered, a series of analyses and simulations for post-retrofit scenarios are performed, e.g., [3–5]. However, because the analyses are based on imprecise input information, the proposed retrofit measures often do not perform as expected, again adding to the performance gap. Being able to obtain measured data quickly and effectively can improve the accuracy of the initial assessment of the building, leading to more efficient retrofit measures and reduced CO_2 emissions from retrofitted buildings.

The last few decades have witnessed a transition of on-site data acquisition techniques from wall-powered data-loggers to wireless sensor networks [6]. Advances in battery technology, electronics, and wireless communication now allow for remote deployment of a large number of sensors. At the same time, the measurement results can be analyzed while the measurement campaign is still in progress.

Until recently, wireless sensor networks (WSN) technology was apparently handled mostly by experts from an electrical engineering background. It has been challenging for experts from other domains to familiarize themselves with and use this technology. As we demonstrate, the Arduino microcontroller platform can fill this gap. Arduino is an open-source hardware and software framework. Its use is versatile, including in education from primary to university level [7–10], by hobbyists and in industry [11,12]. With its flexibility and standardized pin header layout, the approach of circuit boards (shields) that are stacked on top of the Arduino board is ideal for the rapid prototyping of electronics. However, it does not scale well for applications requiring a multitude of devices, because the flexibility and standardized pin headers come with increased size and cost.

Open-source software is well established in industry and academia [13]. In contrast, open-source hardware is still maturing. Since by

* Corresponding author.
E-mail address: mario.frei@arch.ethz.ch (M. Frei).

https://doi.org/10.1016/j.autcon.2019.103043
Received 19 June 2019; Received in revised form 20 November 2019; Accepted 2 December 2019

definition, all design files for open source hardware are available, it can be studied and modified. Furthermore, vendor lock-in is not possible, i.e., if the product is not available, it can still be manufactured by someone else.

With the advent of online printed circuit board (PCB) manufacturing services, it is now possible to order three pieces of a PCB for less than 10 USD. Furthermore, custom casings for the electronics can be produced with desktop 3D printers. Hence, it is possible to prototype and produce custom high-quality PCBs and casings quickly and economically.

1.1. Context and justification

There is a plethora of WSNs identified in the literature, including MICA2, MICAz, TelosB, iMote, Wispes, etc. Despite the frequent mentions of these WSNs in the literature, the source files for the software and hardware of these WSNs are seldom completely publicly available. Moreover, these WSNs themselves are the focus of research in which researchers assess and improve the performance of the WSN (i.e., power consumption, range, networking, security, etc.). The actual measured value of a physical characteristic is often treated as a byproduct and is often neglected, e.g. [14]. We argue that while this leads to continuous improvement of the sensor networks, it remains challenging for these sensor networks to be used in research in other fields. Nevertheless, there are research projects using WSNs that focus on the measured data. There are many examples that use a combination of Arduino and XBee modules to acquire measured data. Every team developed the hardware assembly and the entire software from scratch because the source files have not been shared, e.g., [13,15–20]. In [19,21–25], the authors assess indoor air quality using a low-cost WSN. They all measure a subset of air temperature, relative humidity, CO_2 concentration, CO concentration, particulate matter, and VOC with different systems. Despite the low-cost claim, all of the authors omit the discussion of the actual cost of the presented systems, nor do they share the design files. A noteworthy counterexample is the Open-Source Building Science Sensors (OSBSS) by Ali et al. [26]. OSBSS an open-source Arduino based sensor platform for indoor data collection. Documentation of the work is available online including tutorials. However, the sensor platform is logger based, therefore offline, and it is still in a prototype stage.

Ahmad et al. and Kumar et al. highlight the benefits of WSN in the built environment, being low cost and flexibility during installation [20,27]. They also mention that the benefits come at the cost of potentially lower reliability, potentially higher technical complexity, and data privacy concerns. Martín-Garín et al. demonstrate the deployment of wireless sensor networks in occupied apartments for one week, measuring temperature, windows opening state, CO_2 concentration, atmospheric pressure, and relative humidity [13]. They also demonstrate the calibration of the sensors. Calibration of sensors is also highlighted in [18], where an Arduino based system is connected to an open-source software platform for communication and data analysis. The system is tested for ten days in an office space. It is capable of measuring air velocity, CO_2 concentration, relative humidity, occupancy, air temperature, illuminance, VOC, and particulate matter. Both aforementioned sensor systems are still in a prototype stage and do not discuss the battery lifetime.

It appears to be an unachievable task to develop a sensor network that incorporates all types of sensors needed by researchers from different domains. However, modularity allows researchers from many domains to add their desired sensors without having to develop an entirely new sensor network. This modularity is an essential element of the framework's appeal to others since they can reuse some hardware and/or software components and add or change others, thereby saving resources.

In summary, to establish a remote sensing platform, one has to redo previous work or buy closed-source hardware that can only with difficulty incorporate all the desired sensors. One may need to resort to using more than one sensing solution because one solution alone might not encompass all types of the required sensors. Both approaches lead to a waste of resources.

1.2. Objective

The objective of this work is to design an open-source WSN that is easy to program, manufacture, modify, and deploy. By 'easy' we mean with few resources and without professional training. The cost and performance of the sensor network are intended to be as good as or better than existing solutions. This sensor network is meant to be used by researchers from all fields who must take remote measurements. Making this work open-source ensures that it can be reused entirely or in parts, avoiding repeated development of the same tools without added benefits.

2. Sensor network design

In this section, the design of the hardware and software of the WSN is described. All design files are available on GitHub [28]. This allows future developers to use existing work and contribute to the field of their respective expertise. Only new contributions need to be tested; the other tested components can be relied on. We encourage all readers to use, copy, derive, and contribute to the repository. The goal was to develop a sensor network that is easy to use and reuse at a reasonable cost. The WSN is embedded within the Arduino ecosystem, giving its users access to the existing integrated development environment (IDE), the plethora of software libraries to interface numerous sensors, and tutorials. In addition, it allows prototyping using standard Arduino devices.

The main components of the WSN are the sensor nodes, router nodes, gateway, and web server. The sensor nodes carry out the measurements and send the measured data to the gateway. If there is no direct connection (not to be confused with line of sight) between a sensor node and the gateway, the data will be routed through one or more router nodes. This data transfer takes place via the ZigBee protocol. Then, the data is relayed from the gateway to the web server using General Packet Radio Service (GPRS), as shown in Fig. 1. On the web server, the data is stored in a MySQL database. The database can serve the data to any arbitrary software environment, e.g., MATLAB, Python, PHP, etc.

2.1. Hardware

This section describes the hardware of the WSN, i.e., the sensor nodes, router nodes, and gateway (Fig. 2, left). A sensor kit consists of 18 sensor nodes, five router nodes, and one gateway. The sensor kit fits in a box (40 cm × 30 cm × 13 cm) for convenient transport (Fig. 2, right).

2.1.1. Sensor node

The sensor nodes integrate sensors, power management, and wireless communication. The sensor nodes themselves are modular, consisting of one main module (MM) and one sensor module (SM). The sensor module and the battery are connected to the main module, and the entire assembly is protected by a 3D-printed casing (see Fig. 3, top).

The main modules and sensor modules can be exchanged independently and are connected with a connector integrating a power bus and a serial communication bus. The sensors on the sensor module are interfaced with a microcontroller. The microcontroller on the sensor module transfers the measurement data in a standardized format via the serial communication bus to the microcontroller of the main module. On the main module, the measurement readings from the sensor module are saved to the SD memory card and/or sent to the gateway using the ZigBee radio module (see Fig. 3, bottom).

Separating the sensing functionality from the communication

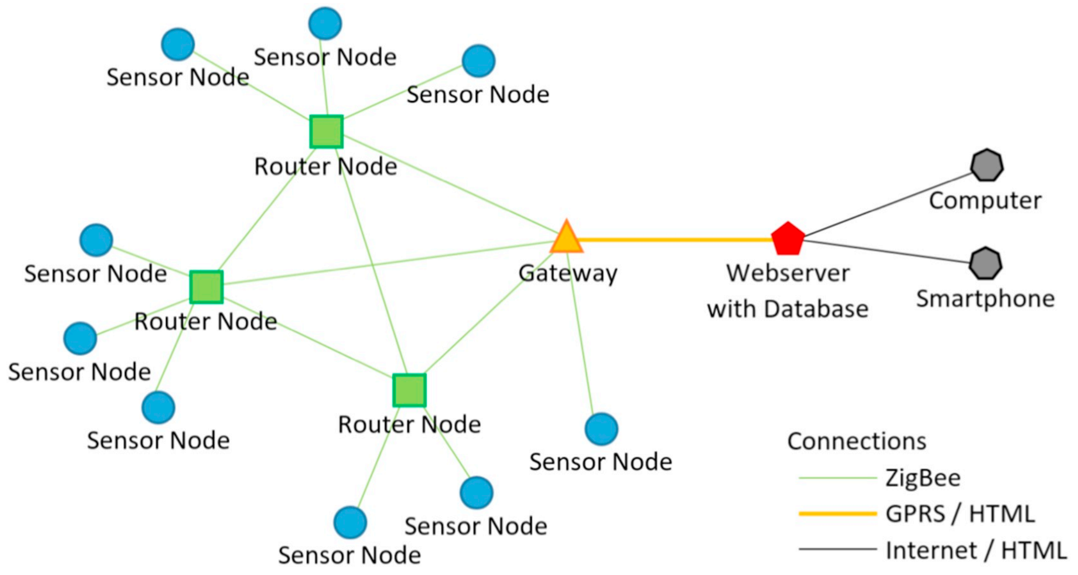


Fig. 1. Schematic of the communication paths within the WSN.

functionality into two PCBs makes independent development of the two modules possible. New sensor modules can be added to the WSN framework without altering the communication and power management, provided the boards adhere to the specified communication and power link between the two boards. Also, new means of communication can be added to the framework, without the need to alter the sensor modules.

Development is simplified because the microcontroller and the related components (e.g., resonator) are identical on the main module and sensor module. The settings of the IDE do not need to be changed for programming, and developers must familiarize themselves with the peculiarities of only one microcontroller. Also, the number of unique items on the bill of materials is kept as low as possible.

The splitting of the sensor node into a main module and a sensor module comes at the cost of added hardware (e.g., microcontroller, connectors, etc.). Nevertheless, the increased flexibility of development outweighs the added cost of components.

The microcontroller for the main module and the sensor module is a Microchip ATmega328p. The same microcontroller is used in the Arduino Pro Mini. Hence, the same settings in the IDE can be used to program the main module. Furthermore, Arduino Pro Minis can be used to set up prototypes. The microcontroller runs on 3.3 V/8 MHz and can be programmed using the in-system programming (ISP) pin header and a programmer or an Arduino set up as an ISP programmer.

2.1.2. Main module

The main module interfaces the sensor module. It triggers the measurement and sends the measured data to the gateway. Its main components are labeled in Fig. 4. It measures 57 mm by 51 mm.

The XBee transceiver on the main module is a ‘XBee S2C 802.15.4’ module manufactured by Digi International. It uses the 2.4 GHz frequency band and ZigBee protocol for communication. These radio modules provide mesh networking capability out of the box. The radio module and networking settings must be directly programmed into the radio module using the free accompanying software, XCTU. No additional programming is required for networking. The mesh network is established ad-hoc, and the routing is done by the XBee modules.

The ZigBee network protocol requires that exactly one radio module is configured as coordinator for every network. The other modules can be configured as a router or as an end device. For this framework, the XBee module of the gateway was configured as a coordinator. The XBee modules on router nodes are configured as a router, and the XBee modules on sensor nodes are configured as end devices (see Table 1).

Mesh networking allows data packages to be routed from sensor nodes through router nodes to the gateway. XBee modules configured as end devices send data to the gateway. If there is no direct line of communication, the sensor node can send the data packets to a router, which then sends it to the gateway or to another router if necessary (see Fig. 5). Mesh networks are practical in buildings with structures that

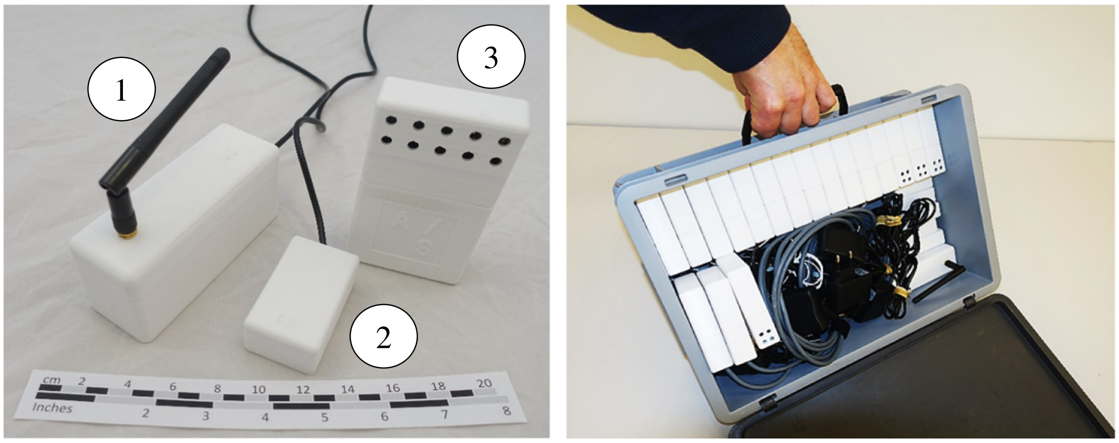
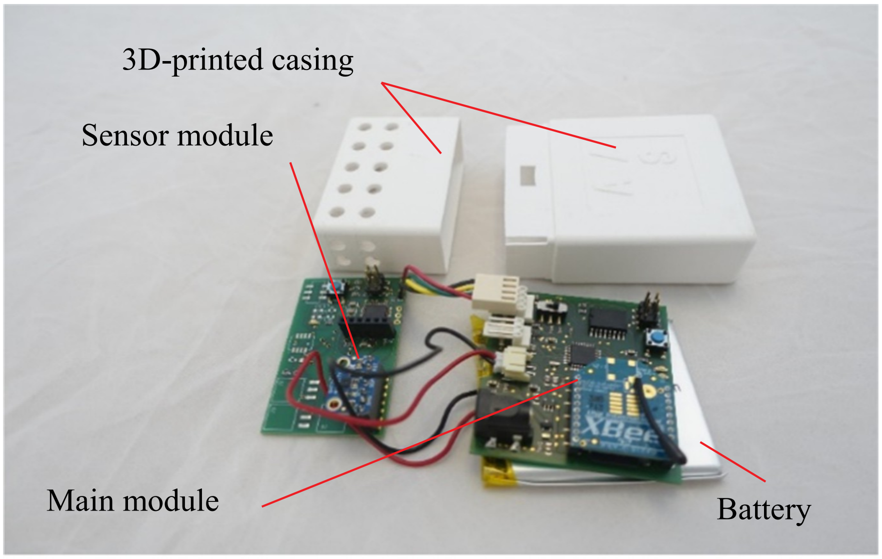


Fig. 2. Left: the three components of the WSN (1. Gateway, 2. Router node, 3. Sensor node), Right: sensor-kit packaged in a box for convenient transport.



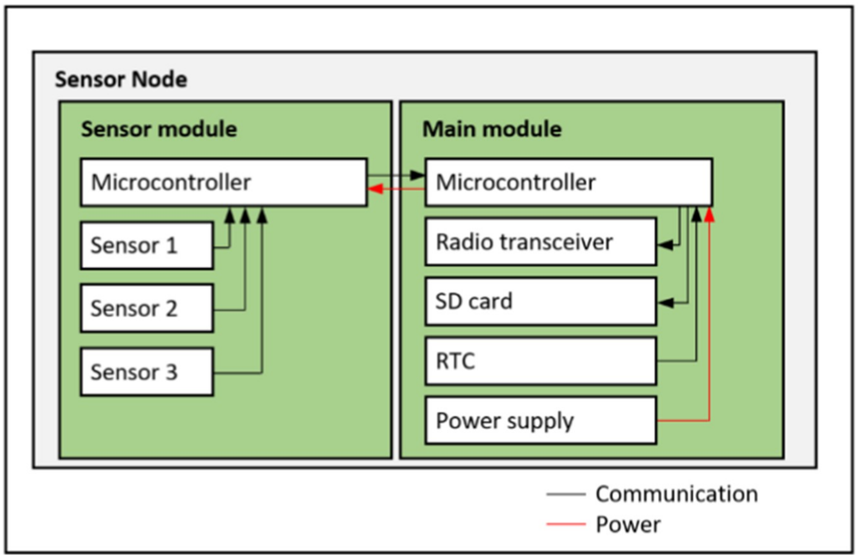


Fig. 3. Top: Photo of sensor node: 3D-printed casing with PCBs of sensor module, communication module, and LiPo-battery, bottom: General schematic of the sensor node.

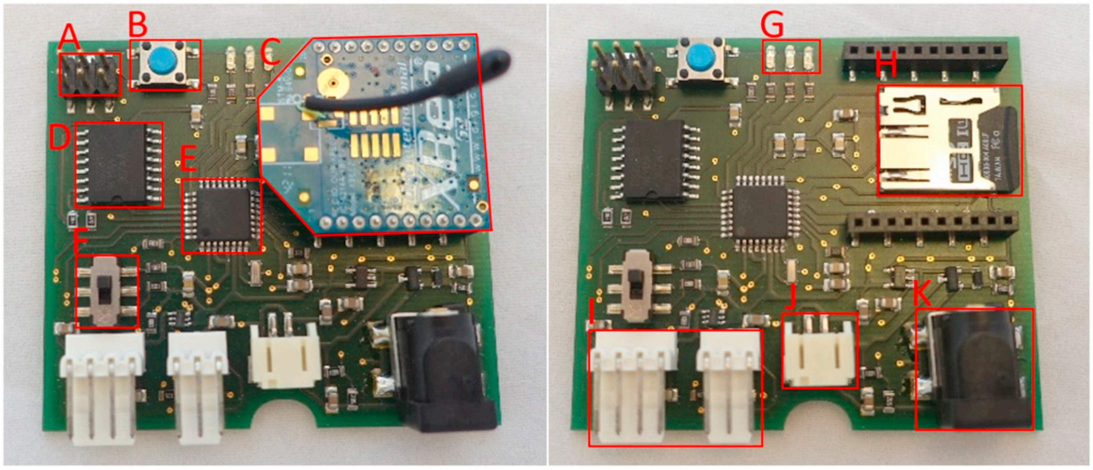


Fig. 4. PCB of the main module, left: with XBee radio, right: without XBee radio, A: ISP pin header, B: Reset button, C: XBee module, D: Real-time clock (RTC), E: Microcontroller, F: Debugging switch, G: Debugging LEDs (left to right: green, orange, red), H: SD card, I: Connectors to sensor module, J: Battery connector, K: Power jack. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Framework designation vs. ZigBee networking designation.

A/S-WSN device	ZigBee designation of XBee module
Gateway (G)	Coordinator
Router node (R)	Router
Sensor node (S)	End Device

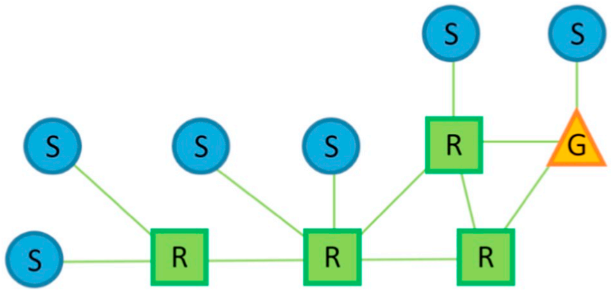


Fig. 5. Mesh networking topology: S: Sensor node, R: Router node, G: Gateway.

limit the reach of the radio modules. Router nodes make it possible for sensor nodes to be placed in every corner of the building. End devices can be put into sleep mode to conserve energy. Routers and coordinators cannot be put into sleep mode since data packets could arrive at any instant.

The manufacturer claims a range of 'up to' 60 m indoor and 'up to' 1200 m outdoor. However, the range depends on many factors, e.g., orientation of modules, height of modules, weather conditions, obstacles between the modules, interference sources, etc.

With the devices used in this work, 10 end devices or routers can be connected to a coordinator, and 11 end devices or routers can be connected to another router. However, chains, e.g., C-R-R-R-R, can be built to extend the network to a maximum number of 65,536 devices before reaching the network address limit. However, in practice, the useful number of devices will be limited by the sampling rate of the sensor nodes and the usable bandwidth of the XBee network, which is approximately 40 kilobytes per second. The useable bandwidth decreases by 50% for each time the data is relayed through a router node.

The debugging switch enables switching between two operation modes: 'debugging' and 'normal operation'. In normal operation, the LEDs flash during the booting sequence, indicating a successful initialization process, and then remain dark. In debugging mode, the LEDs flash both during the booting sequence and every time a measurement is taken. In this mode, a measurement is performed every 10 s, regardless of the previously set measurement interval. This feature can be used to diagnose faulty communication, either between the main module and the sensor module or between the main module and the gateway, without the immediate need for additional debugging tools during deployment.

The main module can be powered by a battery or by a mains adapter. Both power inputs are regulated to 3.3 V. We used Lithium polymer (LiPo) batteries with a capacity of 2000 mAh and with a nominal voltage of 3.7 V. When the battery reaches the threshold voltage of 3.4 V, the main module sends a final 'low battery' message to the gateway and then resorts to a continuous deep sleep mode.

The 4-pin connector to the sensor module provides ground and 3.3 V that can be switched on and off by the main module using a high side switch. The other two lines are for serial communication. The 4-pin connector is always needed. The 3-pin connector is an extra connector that can be used if required. It provides a continuous 3.3 V to the sensor board, even if the main module is in a sleep state. It also supplies raw power from the power jack on the main module to the sensor module. Continuous powering of the sensor module is necessary for event-based measurements such as windows opening or for sensors that need continuous power or more than 3.3 V, such as the CO_2 sensor, which requires a continuous 5 V. The third pin is an interrupt input to the main module's microcontroller. For event-based measurements, the microcontroller is woken up by an interrupt signal from the sensor board instead of the interrupt signal provided by RTC.

2.1.3. *Sensor modules*

There is a dedicated microcontroller on the sensor modules. The microcontroller has a non-volatile memory on which calibration data of sensors could be permanently stored in the future, increasing the accuracy of sensors. The dedicated microcontroller on the sensor modules can interface multiple sensors that have various communication protocols (e.g., UART, SPI, I2C, etc.), without any address conflicts between the devices on the main module and the sensors on the sensor modules. Moreover, the specific requirements of the sensors embedded in the sensor module can be handled by the dedicated microcontroller, without any changes needed on the main module. We designed sensor boards to interface eight different sensor configurations (see Table 2). The choice of sensors configurations is based on previous work [5] and the literature [26]. In [5], the authors concluded that heat flux sensors would improve the U-value estimations and sensors to assess heat losses through ventilation (S8, Reed switch) could improve the thermal assessment of the building.

The sensor for air temperature and relative humidity SHT31 measures the room temperature in buildings. This sensor is the replacement product for the SHT15 sensor used in [5,26].

The temperature sensor DS18B20 is used pairwise to measure inside and outside air temperatures next to building elements, (e.g., wall, window, roof) or supply temperatures and return temperatures of heating systems or radiators. These sensor has also been used in [5,26].

The luminosity sensor TSL2561 is used to assess the visual comfort of building occupants. These sensor has also been used in [5,26].

The volumetric oil flow sensor HZ6-DR is used to measure the oil flow rate of oil-based heating systems. The oil flow sensor briefly closes a relay for every 0.02 l of oil flow resulting in a high pulse. The counts are summed up and transmitted to the main module upon request. This is the replacement product for the sensor HZ-5-RR used in [5].

Access to electricity consumption data is essential to assess heat input into a building in cases where the heating system is all-electric

Table 2
List of sensors currently used with the WSN.

No	Measured entity	Manufacturer	Sensor	Range	Accuracy	Interface
1	Air temperature, relative humidity	Sensirion	SHT31	−40 °C to +125 °C 0% to 100%	± 0.3 °C ± 2%	I^2C
2	Temperatures (2×)	Maxim Integrated	DS18B20	−55 °C to +125 °C	± 0.5 °C	1-Wire
3	Luminosity	AMS	TSL2561	0.1 lx to 40,000 lx	–	I^2C
4	Volumetric oil flow	Braun Messtechnik	HZ6-DR	1 l/h to 60 l/h	± 1%	Pulse
5	Light pulse	AMS	TSL257	–	–	Pulse
6	Reed switch	C&K		–	–	Pulse
7	CO_2 concentration	SenseAir	S8	400 ppm to 20,000 ppm	± 40 ppm	Serial
8	Heat flux	greenTEG	gSKIN-XO 667C	-25 kW/m^2 to 25 kW/m^2	± 3%	Analog

[29]. The light-to-voltage converter TSL257 detects the flashes of indicator LEDs on modern residential electricity meters, which indicate the current electric power consumption, usually one pulse per Wh. It converts the light flashes to voltage pulses, which can be detected by the microcontroller. The counts are summed up and transmitted to the main module upon request.

Reed switches and magnets are used to detect the opening of windows and doors. When a window opens, the magnet on the window is separated from the reed switch on the window frame. The reed switch electrically closes, causing a change in voltage that triggers an interrupt on the main module's microcontroller. In turn, the main module will power on the microcontroller on the sensor module. The duration of the absence of the magnet near the reed switch is measured in seconds on the sensor module. When the window is closed again, the change in voltage is detected by the sensor module's microcontroller, and the duration for which the window is open is transmitted to the main module.

The CO_2 sensor S8 is used to assess the CO_2 concentration of the indoor air. This sensor requires 5 V continuous power. For that reason, the 3pin-connector is used to supply power to the sensor. In [5], the authors concluded the need for insight into heat losses through ventilation. The air change rate can be derived from the CO_2 concentration of indoor environments [30]. Furthermore, occupancy can also be derived from the CO_2 concentration of indoor environments. The use of this particular sensor has been demonstrated in [26].

The heat flux sensor gSKIN-XO 667C is used to measure the heat flux through building elements. The U-value of a building element can be estimated from a heat flux measurement and a temperature measurement from each side of the building element. The heat flux sensor outputs a voltage in the order of microvolts, which is more challenging to handle than the other sensors, which are all digital or binary. Due to this, the electronics workshop of our institution's physics department developed the sensor module with the heat flux sensor. Because their workshop's infrastructure is centered on PIC microcontrollers from Microchip Technology, this sensor module uses a PIC16F1825-I microcontroller instead of the ATmega328p. Even though this deviation from the standard microcontroller choice is suboptimal, it demonstrates the flexibility of the two-module approach for sensor nodes. An analog front end (MCP3911 from Microchip Technology) is used to read out the small voltages of the heat flux sensor.

2.1.4. Router node

The router nodes consist of an XBee module stacked on a regulated SparkFun XBee Explorer breakout board. No additional microcontroller is needed. A 3D-printed casing protects the electronics (see Fig. 6). The XBee module is configured as a ‘Router’. Router nodes need to be powered continuously since data packets for re-transmission can arrive at any time. Therefore it is powered by a 5 V mains adapter power supply.

2.1.5. Gateway

The gateway is the sink node of the entire mesh network. Sensor nodes send all measured data to the gateway, which then relays the data over GPRS to the web server. There must be precisely one gateway for every network.

The gateway PCB has one slot for the XBee radio module and one for the Adafruit Feather 32u4 FONA microcontroller board (see Fig. 7). The XBee module is configured as a coordinator and is connected to the microcontroller board via a serial bus. The microcontroller board features an ATmega32u4 microcontroller and a SIM800H GPRS modem. This is a second deviation from the standard microcontroller choice for this framework. However, the ATmega32u4 is still Arduino compatible, and we could not find a similarly priced microcontroller and modem combination using the ATmega328p.

A battery connector and battery charger circuit are integrated into the Feather 32u4 FONA as well. There must always be a battery connected, even when powered over USB because, without one, the GPRS modem's high current spikes could cause random resets of the microcontroller. We used a 1000 mAh LiPo battery for the gateway. In addition to dealing with the current spikes, the battery allows for a few hours of wireless operation. However, the normal operation mode is to power the gateway with a USB power supply and use the battery only to provide for the high current peaks of the modem. The 3D-printed casing protects the electronics and holds the antenna for the modem.

2.1.6. Manufacturing

To meet the self-imposed requirement of ‘easy manufacturing’, we opted for surface-mount devices (SMD) instead of through-hole technology (THT). We used the 0805 packages (2.0 mm × 1.25 mm) for most SMD parts. These packages can be soldered by hand without the need for visual aids. In addition, the size of the PCB can be kept small, and batches of several dozens of PCBs can be produced in a timely manner. The only exceptions are the connectors between the sensor module and the main module. Those parts are still THT.

The PCBs have two layers and are populated on one side. The PCBs and solder paste stencils can be ordered via online PCB production services. Because all components are on the same side, the PCBs can be reflow soldered in one pass. The average assembly time for one main module was 12 min.

2.2. Software architecture

This section describes the programming code that is required to take a measurement and store it in the database. Only the part of the code on

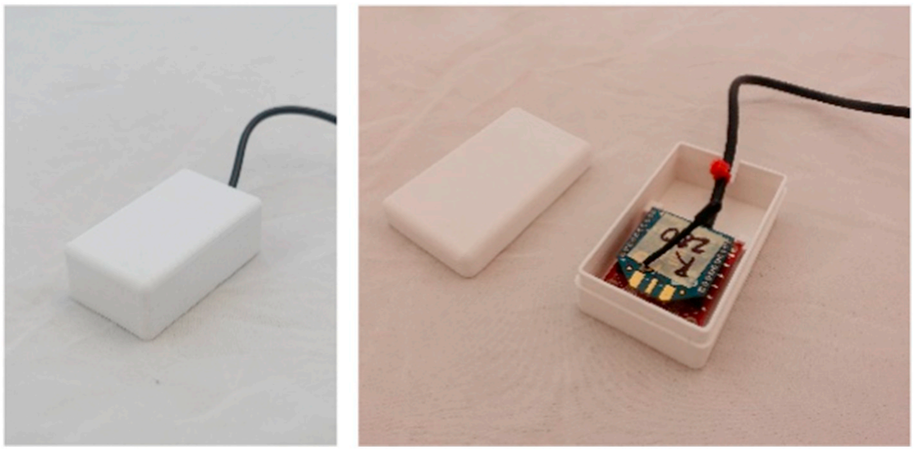


Fig. 6. Router node with 3D-printed casing; left: closed lid, right; open lid.

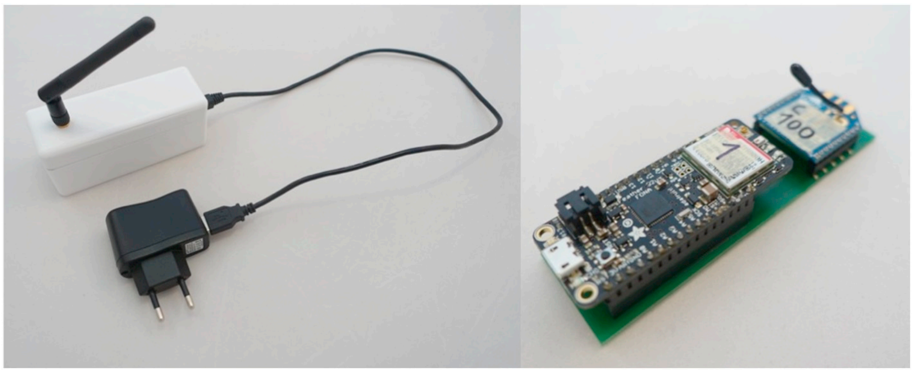


Fig. 7. Gateway: left: Gateway in 3D-printed casing including power supply, right: Gateway PCB populated with Adafruit Feather FONA 32u4 and XBee module without power supply, antenna and battery.

the various sensor modules that interfaces the actual sensors varies; the rest of the code on the sensor modules is in common. The code on all main modules is identical, except for the hardcoded gateway addresses and sensor node IDs.

2.2.1. Main module

During normal operation, if the sensor module takes measurements in regular intervals, the microcontroller on the main module is woken up by interrupts from the RTC. If the sensor module takes event-based (irregular) measurements, the main module is woken up by interrupts from the sensor module. When an interrupt is triggered on the main module, the power supply to the sensor module is activated, and the sensor module responds by sending the measured data. The received data is then checked for validity using a checksum and is repackaged into a suitable format for transmission to the gateway. The repackaging of the payload includes appending the sensor node ID to measured data. The repackaged payload is then sent to the gateway. Only one transmission is attempted. If there is an SD card present, the payload is then written to it. Finally, the microcontroller is again set to sleep mode and the cycle restarts.

Table 3 shows the structure of the payload message that is sent from the main module via the XBee network to the gateway. Its header includes the main module ID, number of measurements, and sensor module type. To save energy, multiple measurements can be collected on the main module and sent to the gateway as a bundle. One measurement may contain one or more values. The last parameter of the header is the sensor module type, which describes how many and what kind of sensors are built into the sensor module. Following the message header, the measurement data is appended. A checksum is not necessary for these particular radio modules since the XBee modules already check the validity of the message.

2.2.2. Sensor module

First, the serial bus is initialized. If the sensor module is one that takes event-based measurements (e.g., duration of windows open), it will check the serial bus for a “0xBB” byte sent from the main module. This byte is sent from the main module during its setup phase to determine whether the sensor module is a sensor module that takes periodic measurements or one that takes event-based measurements. An event-based sensor module will respond with a “0xBB” byte in order to signal the dependency on events and leave the setup function. Sensor modules for periodic measurement would ignore this negotiation.

During normal operation, i.e., not during the setup phase of the main module, the initialization of the serial bus is followed by the initialization of the sensors. The measurements are then carried out. For the transmission of the data to the main module, firstly, sends a message header. It contains a frame start byte to indicate the transmission of measured data, followed by the sensor module type, which indicates how many and what kind of sensors are installed. The measured values are then appended. A check byte is appended to the end of the payload. The structure of the payload is shown in Table 4.

Table 3
Structure of the main module payload message (MM to Gateway) with an example.

Designation	Bytes	Human readable	Format	Description
Main module ID	0xA5, 0x00	165	16-bit integer, little endian	Identifies the unique hardcoded sensor node ID
Number of measurements	0x × 02	2	8-bit integer	Indicates how many measurements are bundled together in this payload
Sensor module type	0x02	2	8-bit integer	Sensor module with temperature sensor and relative humidity sensor
Measurement 1, Value 1	0x00, 0x00, 0xC5, 0x41	24.625	4-byte float, little endian	Air temperature in °C
Measurement 1, Value 2	0xC7, 0x7A, 0x36, 0x42	45.6199	4-byte float, little endian	Relative humidity in %
Measurement 2, Value 1	0x00, 0x × 00, 0xC5, 0x41	24.625	4-byte float, little endian	Air temperature in °C
Measurement 2, Value 2	0xC7, 0x7A, 0x36, 0x42	45.6199	4-byte float, little endian	Relative humidity in %

Table 4
Structure of the sensor module payload message (SS to MM).

Designation	Bytes	Human readable	Format	Description
Frame start byte	0xAA	–	1 byte	Indicates the start of the transmission of a payload containing measured data
Sensor module type	0x02	2	8-bit integer	Sensor module type, e.g., sensor module with two temperature sensors
Value 1	0x00, 0x00, 0xC5, 0x41	24.625	4-byte float, little endian	First measured value, e.g. air temperature in °C
Value 2	0xC7, 0x7A, 0x36, 0x42	45.6199	4-byte float, little endian	Second measured value, e.g. relative humidity in %
Check byte	0x94	148	8-bit integer	Check byte for transmission error detection, (0xFF-checksum)

2.2.3. Gateway

There is no disaggregation of the data. The payloads received as XBee messages are translated from bytes to hexadecimal characters, so as to comply with the requirement of the URL. The data can be sent to the web server attached as a GET parameter, e.g., https://[mywebserver.com]?payload=[payload]&g_id=[gateway_id], where [mywebserver.com] is the URL or IP of the web server, [payload] is the data to be transmitted, and [gateway_id] is the ID of the gateway. The LED on the microcontroller board blinks twice every 1.5 s or 5 times if a message is transmitted successfully. In case of an unsuccessful data transmission, the data is lost. There is no attempt to resend the data, and there is no local data storage on the gateway.

2.2.4. Web server

The web server serves three purposes: 1. it receives and stores the data from the gateways, 2. it provides a graphical user interface (GUI) for on-site debugging of the sensor network, 3. It serves the measurement data to other applications (e.g., Matlab). It has a public URL and hosts PHP scripts and a MySQL database. One PHP script takes the payload and inserts it into the MySQL database. Another shows a table with the sensor node ID and indicates whether the nodes are online or offline. Access to the user interface was controlled through a hypertext access (.htaccess) file.

We use a MySQL database to store the measurement data. There are only three tables: one to store the information about the different sensor module types, one to store information about the measurement campaign, and one to store the measured data. The tables with example values are shown in Fig. 8. The 'Id' field in each table is the primary key of the respective table must, therefore, be unique. The 'Id' field of the 'wsn_sensor_module_type' is used as a foreign key in the 'wsn_input' table and the 'Id'.

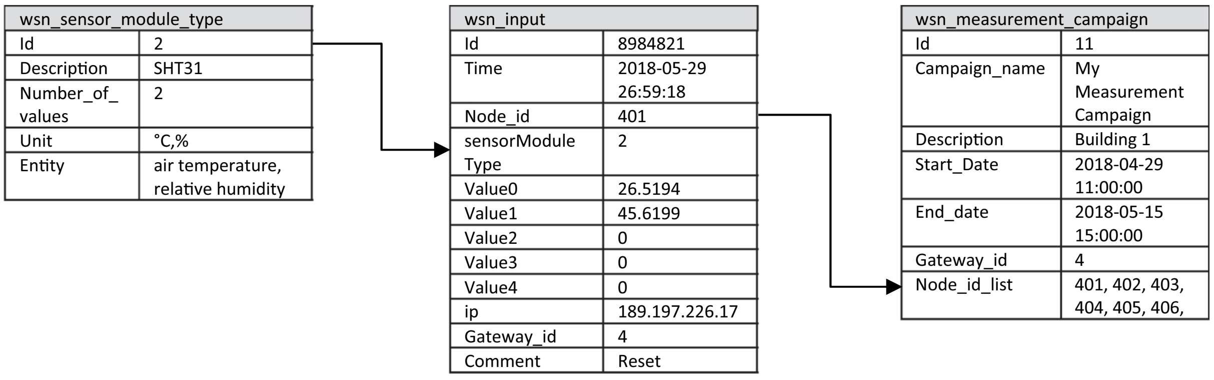
The 'gateway to MySQL' PHP script receives the payload from the gateway as a GET parameter, disaggregates it into separate values, and inserts it into the MySQL database, adding a timestamp.

There is a GUI in the form of a website. The page lists the sensor node ID, the sensor module type, the elapsed time since the last data package was received, and a status field for all sensor nodes from one sensor kit (see Fig. 9, top). When a sensor node ID is selected from the page with the tables, a line plot for the sensors of sensor nodes is shown. The line plot lets the user see whether the sensors report meaningful data as an indicator of whether the sensors were installed correctly (see Fig. 9, bottom).

The web GUI is intended for an on-site functional test. When deploying the WSN on-site, it is crucial to know whether data is received from all the sensor nodes. The website can be accessed with a computer or a smartphone. If some sensor nodes appear to be offline, additional router nodes need to be put in place between the gateway and the respective sensor nodes.

3. Performance of the sensor kit

Sensor kits consisting of up to 18 sensors were deployed in eight single-family residential buildings in Switzerland between January 2017 and May 2017. On average, it took 2.5 h to deploy one sensor kit, including an occupant questionnaire and excluding the installation of the oil flow meter, which was done by a third party. On average, it took 1 h to retrieve the installed sensors, including a second occupant questionnaire. The occupant questionnaires included questions about the building, its systems, thermal comfort, and the experience of living with the WSN installed in the house. The oil flow sensors were left in place after the campaign. For each building, an automated report of the measured data was generated. Fig. 10 shows a sample page depicting the information about one air temperature and relative humidity sensor. The report was used to inspect the quality of the data manually before further processing. For example, the top right plot shows the quality of data in terms of lost data packets. Further, validity of the measurement that can be checked, e.g., the minimum value in the second row of plots in Fig. 10 was helpful to identify erroneous data,

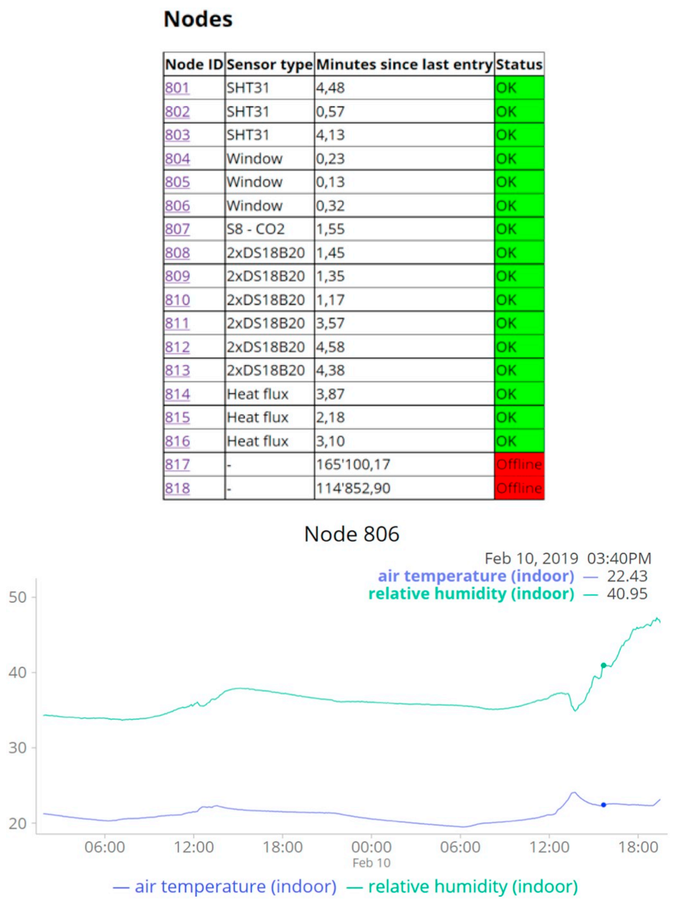


wsn_sensor_module_type	
Id	2
Description	SHT31
Number_of_values	2
Unit	°C,%
Entity	air temperature, relative humidity

wsn_input	
Id	8984821
Time	2018-05-29 26:59:18
Node_id	401
sensorModule Type	2
Value0	26.5194
Value1	45.6199
Value2	0
Value3	0
Value4	0
ip	189.197.226.17
Gateway_id	4
Comment	Reset

wsn_measurement_campaign	
Id	11
Campaign_name	My Measurement Campaign
Description	Building 1
Start_Date	2018-04-29 11:00:00
End_date	2018-05-15 15:00:00
Gateway_id	4
Node_id_list	401, 402, 403, 404, 405, 406,

Fig. 8. Tables of the database and their relations.


Nodes

Node ID	Sensor type	Minutes since last entry	Status
801	SHT31	4,48	OK
802	SHT31	0,57	OK
803	SHT31	4,13	OK
804	Window	0,23	OK
805	Window	0,13	OK
806	Window	0,32	OK
807	S8 - CO2	1,55	OK
808	2xDS18B20	1,45	OK
809	2xDS18B20	1,35	OK
810	2xDS18B20	1,17	OK
811	2xDS18B20	3,57	OK
812	2xDS18B20	4,58	OK
813	2xDS18B20	4,38	OK
814	Heat flux	3,87	OK
815	Heat flux	2,18	OK
816	Heat flux	3,10	OK
817	-	165'100,17	Offline
818	-	114'852,90	Offline

Node 806
Feb 10, 2019 03:40PM
air temperature (indoor) — 22.43
relative humidity (indoor) — 40.95
50
40
30
20
06:00
12:00
18:00
00:00
Feb 10
06:00
12:00
18:00
— air temperature (indoor) — relative humidity (indoor)

Fig. 9. Graphical User Interface: top: overview of one entire sensor kit, bottom: line plot of one sensor module showing the most recent data.

i.e., sensor modules populated with DS18B20 temperature sensors return −127 °C as measurement value in cases where the temperature sensor is disconnected from the sensor module. However, a temperature value −127 °C inside and outside of domestic building is unrealistic needs to be rejected. The histogram in the third row of Fig. 10 is helpful to identify sensors that fell off, e.g., heat flux sensors and pulse detectors for electricity meters. Heat flux sensors with both sides exposed to air in the same volume have virtually no heat flux flowing through them. Therefore, they only measure noise, which is centered around zero. The same holds true for light pulse detectors. If they fall off from the energy meters, they do not measure the light pulses from the energy meter anymore. This failure mode can be seen in the histogram as a dominant peak around zero, as well as in the plot of the time-series (Fig. 10, top-left plot). Lastly, the carpet plots in the bottom row of Fig. 10 reveal patterns, e.g., temperature drops around 7:30 or three distinct humidity regimes; first half of January (~25%), second half of January until end of February (34%), and march (40%).

3.1. Battery lifetime

To assess the power consumption of a sensor node, we investigated the case of one main module connected to one sensor module integrating two DS18B20 temperature sensors. A fully charged 2000 mAh LiPo battery was used as a power source. The sampling interval was set to 1 min. The temperature and battery voltage were measured and transmitted. The data was also written to an SD card. Fig. 11 shows the battery discharge curve of the setup. The sensor node lasted for 112 days.

During a measurement campaign, eight sensor kits, including up to 18 sensors were deployed in eight buildings. The sampling time was set to 5 min, and the sensor nodes were deployed without an SD card onboard. The sensor module populated with two DS18B20 temperature sensors had an average battery life of 10 months. Sensor nodes equipped with SHT31 (temperature, humidity) and heat flux sensors lasted, on average, 12 and 13 months, respectively. Sensor nodes with reed switches, used for measuring opening times of windows, lasted for only 4 months on average. The reduced battery life of sensor nodes with reed switches indicates the potential to improve the design of the corresponding sensor module.

3.2. Testing and validation

Before deploying the sensor-kits in occupied single-family buildings, we compared the sensors of the WSN to commercial sensors in our office spaces and in an apartment.

3.2.1. Temperature and relative humidity

We used an Onset HOBO U12 data logger as a reference to compare to the measurements from the A/S-WSN. We used four SHT31 and four DS18B20 from the sensor kit for air temperature measurements. The SHT31 sensors were also used for the measurement of relative humidity. The sensors were set up in an apartment. All sensors were placed on a tray, 70 cm above the floor, in the hallway without windows, protected from direct solar radiation.

Fig. 12 shows the time-series of temperature measurements from the four DS18B20 sensors recorded with the A/S-WSN and the temperature readings from the HOBO-U12. The measurements are strongly correlated ($R^2 = 0.997$, slope = 0.995). The maximum deviation is 0.45 °C. However, there is a noticeable offset between the HOBO U12 measurements and the DS18B20 measurements, where the HOBO U12 temperature is consistently 0.3 °C higher.

Fig. 13 depicts the temperature measurements of four SHT31 temperature sensors of the sensor-kit and the temperature readings of the HOBO U-12 datalogger. The data is strongly correlated ($R^2 = 0.996$, slope = 0.995). The maximal temperature deviation is 0.27 °C. Similar to the DS18B20 sensors, the SHT31 yield consistently lower temperature, with an offset of approximately 0.06 °C.

Fig. 14 shows the relative humidity measurement data of four STH31 sensors form the sensor-kit compared to the measurement data from the HOBO-U12 data logger. The data is strongly correlated ($R^2 = 0.986$, slope = 0.860). The maximal deviation is 4.2%. The measurement data of the commercial sensor is consistently 1.8% lower than the data from the four SHT31 sensors.

3.2.2. CO_2 concentration

We used a Wöhler CDL210 data logger to compare against the four S8 CO_2 sensors from the sensor-kit. All sensors were placed on a table in the center of a large office space. The measurement data shown in Fig. 15 is strongly correlated ($R^2 = 0.97$, slope = 1.04). The maximum deviation is 124 ppm.

3.2.3. Heat flux

For the comparison of heat flux measurements, we used five gSKIN-XO 667C heat flux sensors. Four were connected to sensor modules of the sensor-kit, and one was connected to a gSKIN data logger. The heat flux sensors were taped to the center of a window in our office space to ensure uniform conditions. The measurements were carried out during the night in order to avoid disturbances of solar radiation Fig. 16 shows the strong correlation between the sensor-kit data and the gSKIN data logger measurement data. The maximal deviation is 1.48 W/m^{2}K.

3.3. Deployment

We deployed sensor-kits in eight single-family building in eastern Switzerland from January to May 2017. Fig. 17 shows the floor plan of one of the case study building and the position of the devices of the

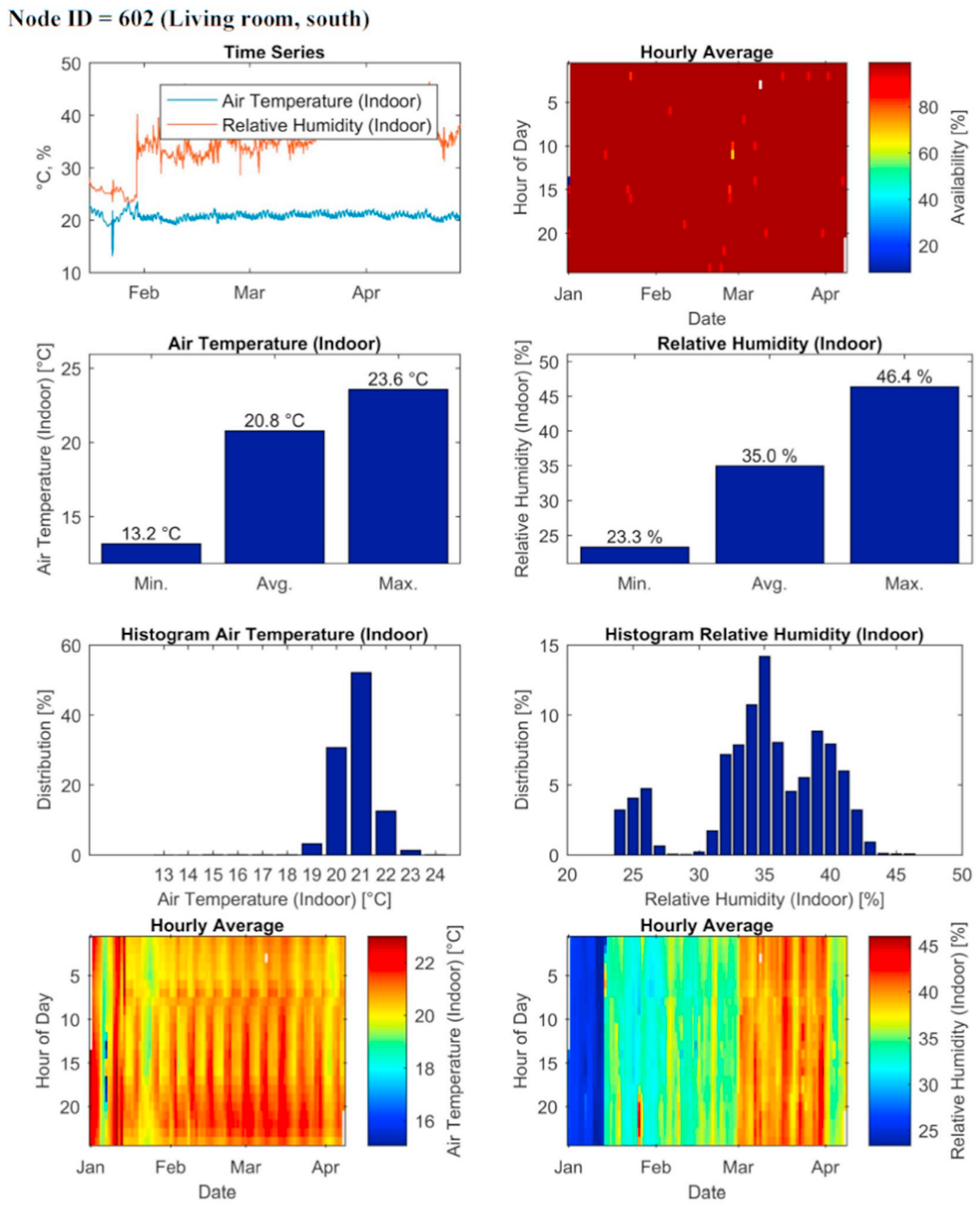


Fig. 10. Sample page for one temperature and humidity sensor from the automated measurement data report.

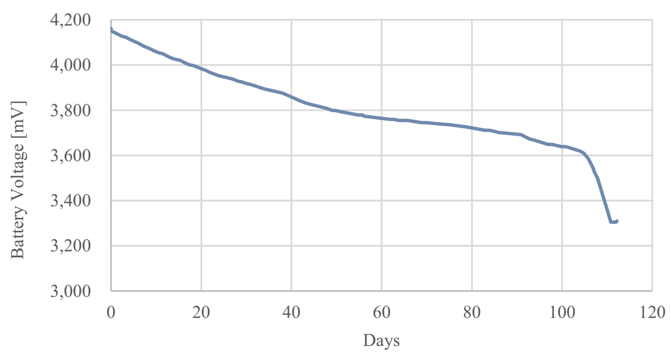


Fig. 11. Battery discharge curve for sensor node equipped with two temperature sensors.

sensor-kit. The building is a single-family house, which was constructed in 1973. The total heated area is 238 m^2. The maximum dimensions of the footprint are 17.5 m by 14 m. The floor height is 2.5 m. All floors consist of massive concrete, and the envelope consists of a cavity wall made out of clay bricks. The roof is made out of clay roof tiles. Four people live in the building. We installed 16 sensors and one gateway in total. No router nodes were deployed. Two sensor nodes for supply and return temperatures could not be installed due to a lack of access to the supply and return temperatures of the floor heating system. The measurement period started on January 19, 2017 and ended on May 11, 2017 (111 days). However, all sensors were only working from March 9, 2017. Initially, there were issues with the pulse counter sensor module regarding the pulse lengths of the electricity meter. The pulse lengths from the on-site electricity meter were much shorter compared to the pulses from the test devices. Hence, we will focus on the 62 days, where all sensors were operational.

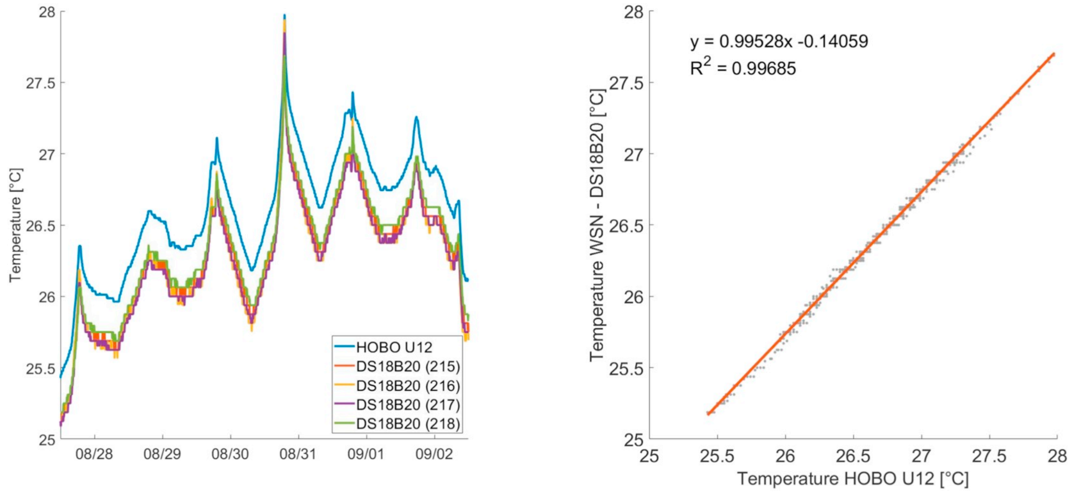


Fig. 12. Left: Temperature measurements in an apartment using the DS18B20 temperature sensor from the A/S-WSN and a commercial data logger. Right: Correlation between one DS18B20 temperature sensor from A/S-WSN and the HOBO U12 data logger.

3.3.1. Time savings

The installation of the sensor nodes in all eight case study buildings took on average 7 min. The installation time of gateways and router is lumped in these 7 min. Further, the installation of the oil meter is not included since it was carried out by a third party. However, connecting the sensor node to the oil meter was considered. During the installation of the sensor nodes, most time is spent on cable management, i.e., carefully arranging cables in order to avoid occupants interfering with cable and for visual comfort. Hence, the sensor node for air temperature and humidity can be deployed in less than a minute, since there are no cables to manage. In contrast, sensor nodes for indoor temperature and outdoor temperature, take up 20 min for the installation, because two cables have to be routed. Even though the router nodes need a wall plug for power supply, the cable of the wall plug was often coiled up, and the router was taped to the power supply since fine-positioning of the router note was not necessary.

The dismantling of the sensor-kit was much faster and only took 3 min on average per sensor node. Most time was spent on removing adhesives from walls and windows.

An earlier version of the sensor kit [31] relied on wall adapters as power supplies, and the data transmission was wireless. Hence, an extension cord had to be routed to each sensor node. That version of the sensor kit was deployed in 3 buildings. The average installation time for one sensor node was 45 min.

3.3.2. Data loss

There were two kinds of gaps. One type is marked with the circles (A) in Fig. 18, where continuous data is available for some sensor nodes, while other sensor nodes show significant gaps. We suspect that the signal strength of the XBee network was not sufficient at all times since sensor nodes close to the gateway did not show gaps. Therefore, we expect that deploying additional router nodes should increase the reliability of the XBee network.

For the second kind of gap, there is no data available for any sensor nodes (see Fig. 18, rectangle B). We assume that this is caused by an issue with the gateway. Possible causes could be a lack of power (e.g., power outage, unplugged mains adapter), lack of reception in the cellular network (e.g., varying signal strength, unplugged antenna), or other unanticipated issues. When investigating these long-lasting total data outages, we encountered unplugged gateways and unplugged antennas. However, in the example in Fig. 18, the sensor network resumed normal operation without intervention. We suspect that an increase in transmit power of the modem on the gateway during periods with weak cellular reception caused interference on the PCBs, triggering resets on

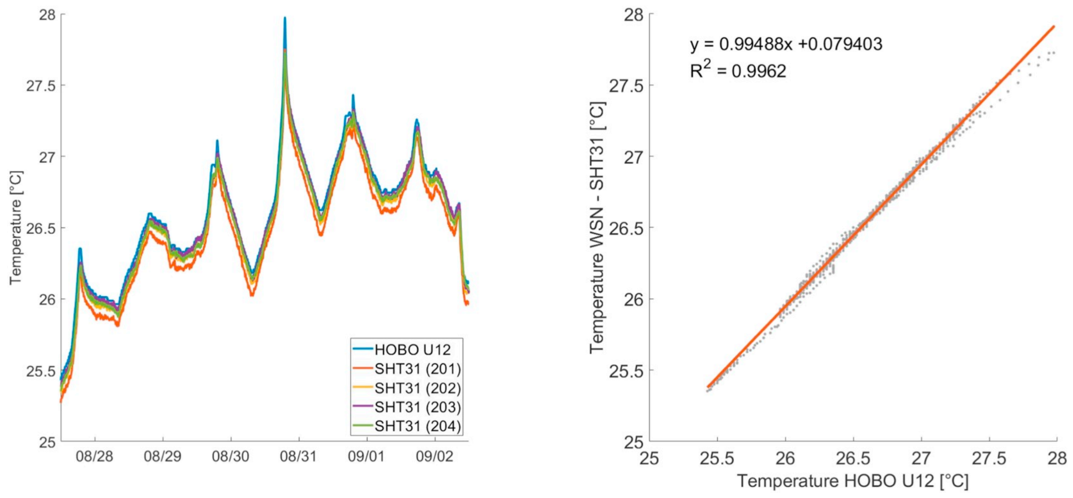


Fig. 13. Left: Temperature measurements in an apartment using the SHT31 temperature and humidity sensor from the A/S-WSN and a commercial data logger. Right: Correlation between one SHT31 temperature and humidity sensor from A/S-WSN and the HOBO U12 data logger.

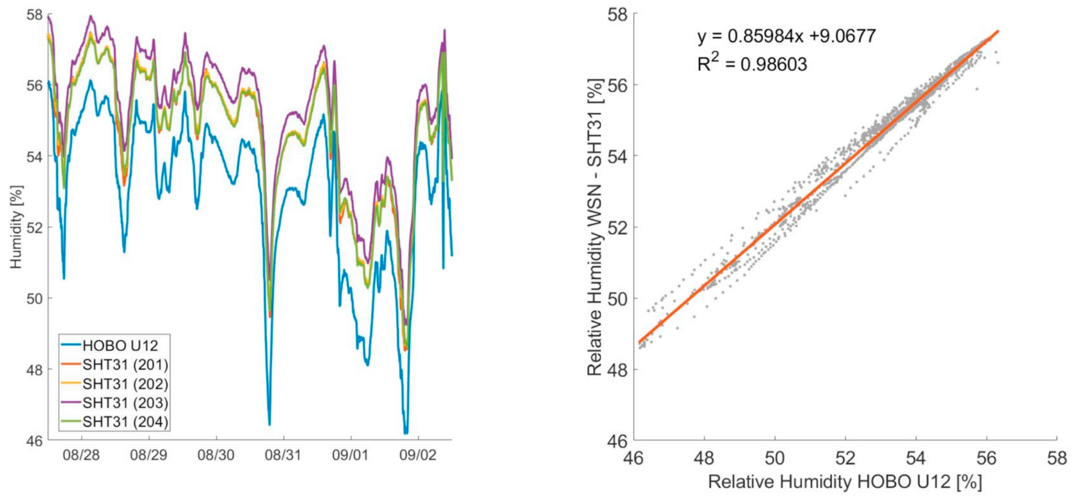


Fig. 14. Left: Humidity measurements in an apartment using the SHT31 temperature and humidity sensor from the A/S-WSN and a commercial data logger. Right: Correlation between one SHT31 temperature and humidity sensor from A/S-WSN and the HOBO U12 data logger.

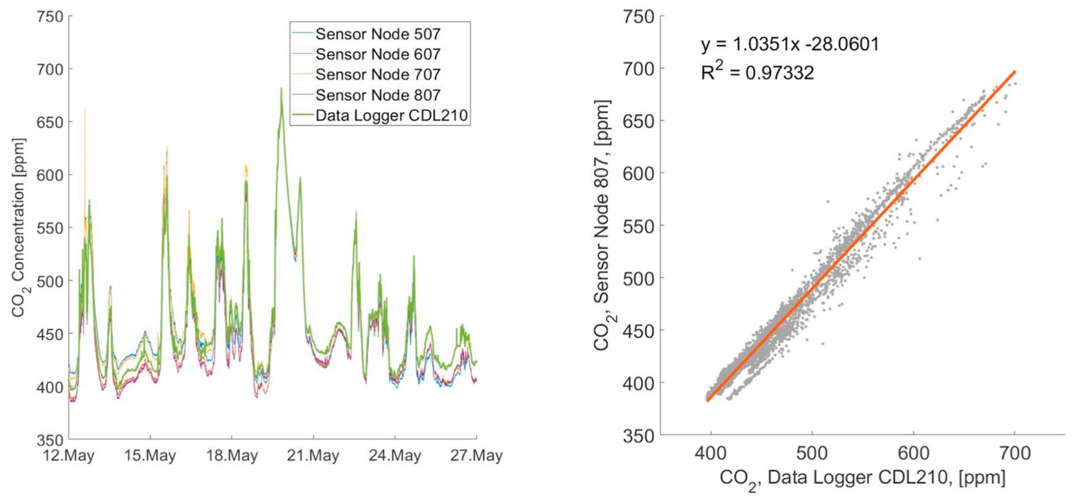


Fig. 15. Left: CO$_2$ measurements in an office space using the S8 CO$_2$ sensor from the A/S-WSN and a commercial data logger CDL210, Right: Correlation between one S8 sensor from the A/S-WSN and the HOBO U12 data logger.

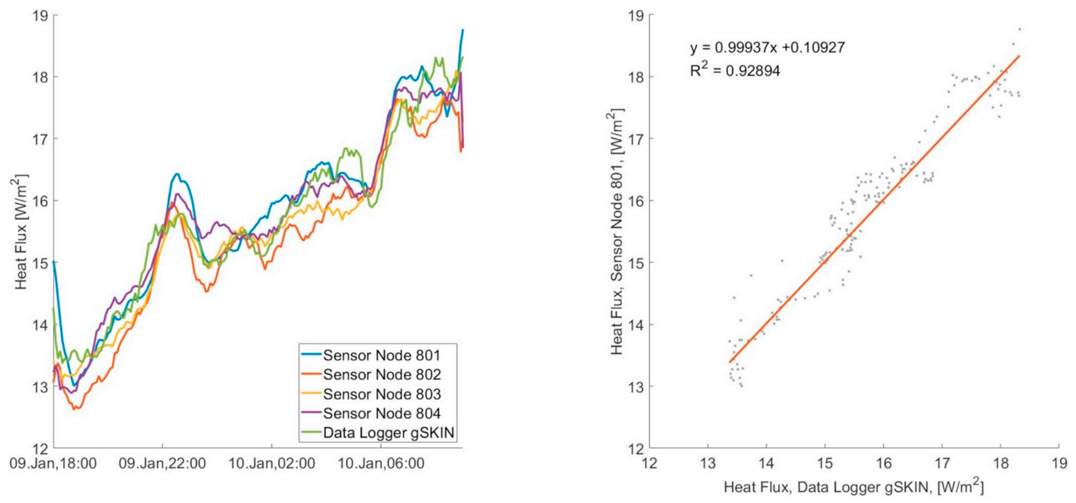


Fig. 16. Left: Overnight heat flux measurements on a windows surface in an office space using gSKIN heat flux sensor with the A/S-WSN and gSkin data logger. Right: Correlation between A/S-WSN and gSkin data logger.

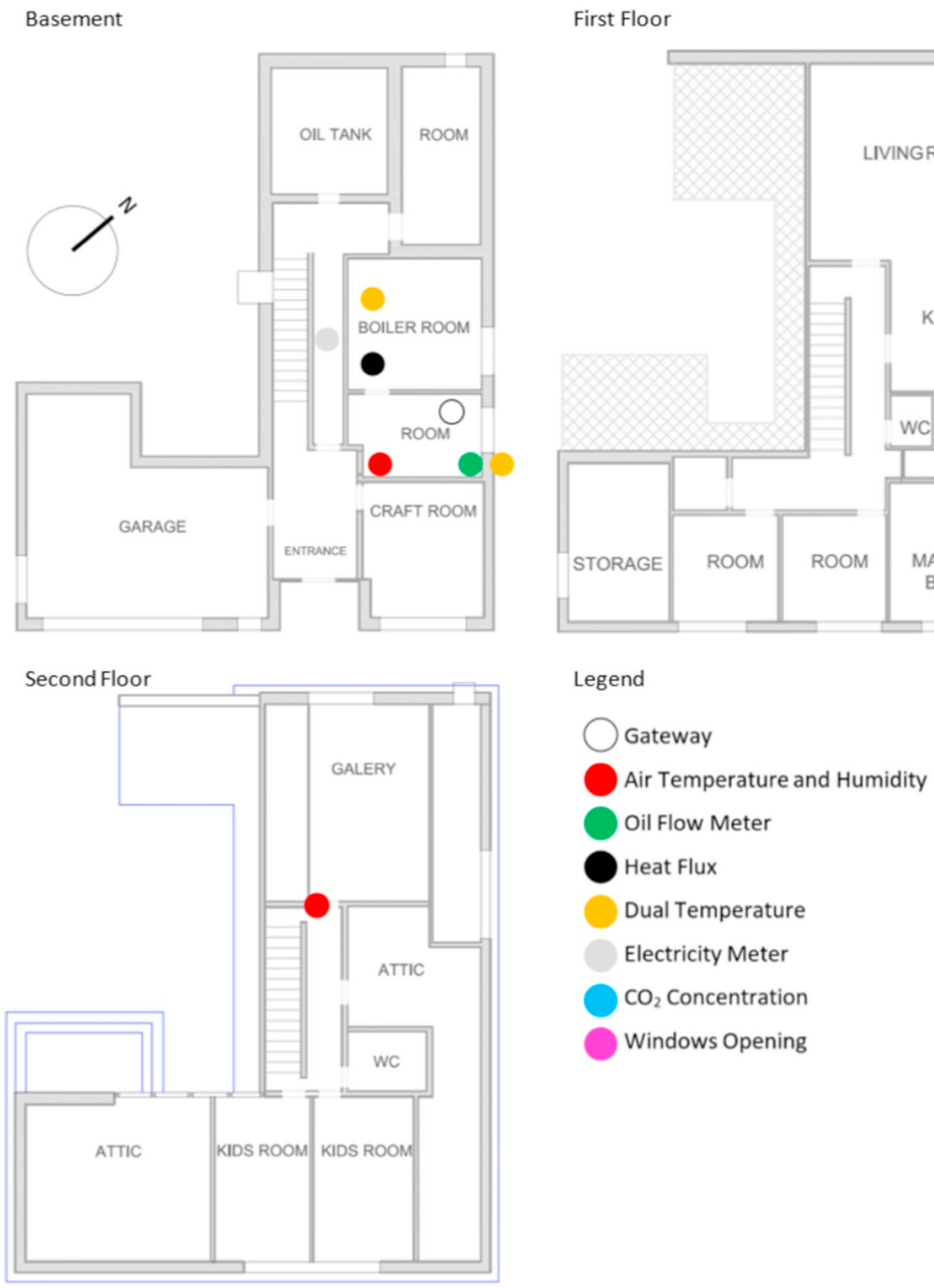


Fig. 17. Floor plan of case study building with positions of sensor nodes and gateway.

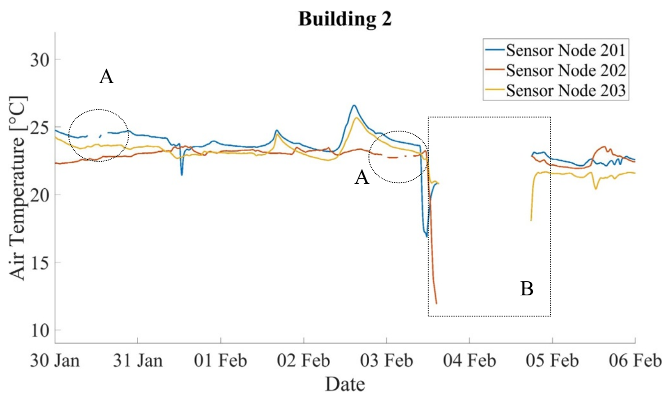


Fig. 18. Examples of different gaps in acquired measured data.

the microcontroller. A similar issue occurred during deployment of the sensor kits, in which the gateway would continuously reset. Usually, repositioning the gateway resolved the issue. However, we were not able to definitively determine the causes of these incidents.

As an example of the data loss, we investigate the electricity consumption of building 2 in Fig. 19. The top left of Fig. 19 shows the raw time-series of the electricity consumption. The bottom left depicts the number of lost data packets at the time of occurrence. On the right-hand side of Fig. 19, all packet losses are summarized in a histogram. One data packet equals 5 min of data.

For comparison, Fig. 20 shows all lost packets from the electricity consumption measurement caused by the gateway. The total data loss rate for the electricity consumption data is 10.8%. The median gap duration is 10.8 min. Only 5.2% of all gaps in the electricity consumption data were caused by the gateway. In this deployment, no router nodes were deployed due to the initial impression on-site of a good network link of the sensor nodes. However, the above data

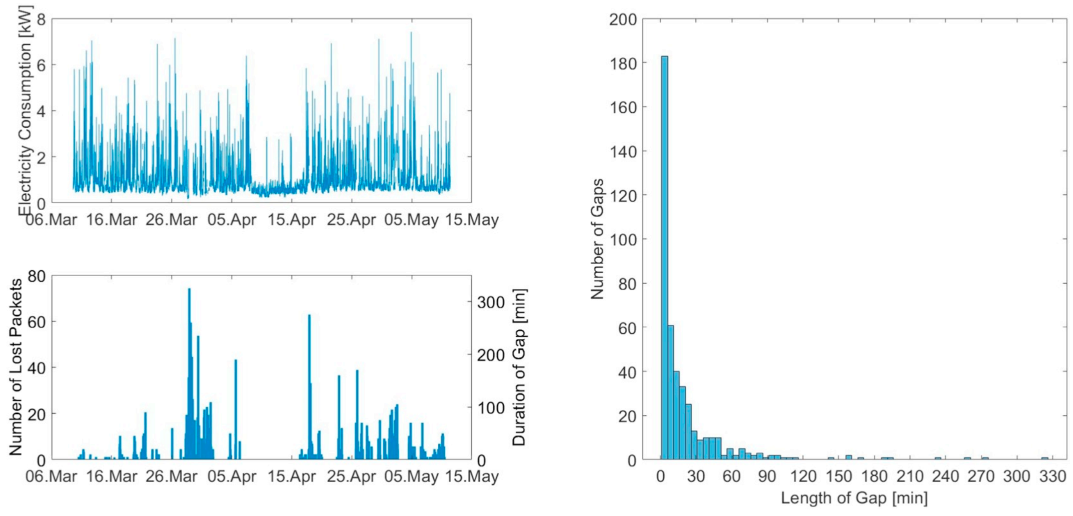


Fig. 19. Top left: electricity consumption data of case study building; bottom left: number of lost packets of electricity consumption data; right: histogram of gap length.

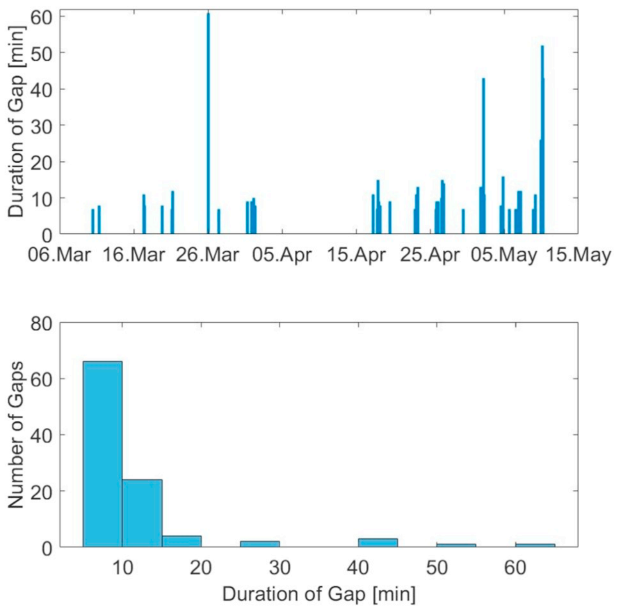


Fig. 20. Gaps caused by gateway in case study building; top: duration of gap versus time; bottom: histogram of gaps.

analysis reveals a substantial data loss, of which only a minor amount was caused by the gateway. Hence, it is advisable to deploy more router nodes.

4. Discussion

The A/S-WSN is a simple, modular, and low-cost wireless sensor network for remote sensor deployment. It can be quickly deployed and can operate for up to a year on a 2000 mAh battery. It is a valuable tool for building performance assessment and can also be of use in other domains. Due to its modularity, some components (hardware and software) can be reused or copied, while others might be replaced or added, yielding both savings in resources and increases in reliability. In the following sub-sections, the design, cost, and application of the A/S-WSN are discussed.

4.1. Cost

The costs of unique items for one sensor kit are listed in Table 5. One sensor kit consisted of one gateway, five router nodes, 18 main modules, 18 sensor modules, three SHT31 sensors, three reed switches, one S8 sensor, six pairs of DS18B20 sensors, one HZ6-DR sensor, one TSL257 sensor, and three gSKIN-XO 667C sensors. The total cost of such a kit is 4174 USD. The heat flux sensors (gSKIN-XO 667C) account for the most substantial part of the cost, 47% for this specific sensor kit. This is followed, at 35%, by the sensor node, i.e., the main module and sensor module without sensors. The other sensors account for 11% of the total cost. The router nodes and the gateway make up only 7% of the total cost. The cost for the sensor module (PCB, microcontroller, connector, etc.) is listed separately from the sensors. The costs are for single quantities; ordering in bulk might lead to significant cost savings. Costs of labor (e.g., assembly) and additional hardware components (e.g., cables and power supplies) are omitted. The costs for network services and web-hosting are excluded, and so are the costs for 3D-printing. The prices are subject to fluctuations. Notably, the prices of the batteries have dropped significantly since the start of this endeavor.

In order to compare the A/S-WSN to other products, we consider a sensor node consisting of one main module (72 USD) and one sensor module for air temperature and relative humidity (32 USD), totaling 104 USD. The measured production time for the main module was on average 12 min. Since the main module is the PCB with the highest parts count, we make the conservative assumption that assembling the

Table 5
Cost of WSN components.

Item	Unit cost	Pcs.	Total cost	Pct.
Sensor TSL2561	5.95 USD	0	0 USD	0.0%
Sensor TSL257	2.91 USD	1	2.91 USD	0.1%
Sensor SHT31	13.95 USD	3	41.85 USD	1.0%
Sensor S8	85.00 USD	1	85.00 USD	2.0%
Sensor DS18B20	9.95 USD	12	119.4 USD	2.9%
Gateway	122.78 USD	1	122.78 USD	2.9%
Router node	32.43 USD	5	162.15 USD	3.9%
Sensor HZ6-DR	224.85 USD	1	224.85 USD	5.4%
Sensor module (w/o sensors)	8.88 USD	18	159.84 USD	3.8%
Main module	71.83 USD	18	1292.94 USD	31.0%
Sensor gSKIN-XO 667C	654.10 USD	3	1962.30 USD	47.0%
Total			4174.02 USD	100.0%

Table 6
Parts cost of the main module.

Item	Unit cost	Pcs.	Total cost	Pct.
Molex 4 pin header	0.84 USD	1	0.84 USD	1.2%
Molex 3 pin header	1.04 USD	1	1.04 USD	1.4%
Power jack	1.84 USD	1	1.84 USD	2.6%
PCB	2.00 USD	1	2.00 USD	2.8%
Female pin header XBee	1.48 USD	2	2.96 USD	4.1%
Microcontroller	3.58 USD	1	3.58 USD	5.0%
microSD socket	3.95 USD	1	3.95 USD	5.5%
Various SMD components	–	–	5.81 USD	8.1%
RTC	6.22 USD	1	6.22 USD	8.7%
XBee Series 2	18.19 USD	1	18.19 USD	25.3%
Battery 2000 mAh	25.40 USD	1	25.40 USD	35.4%
Total			71.83 USD	100.0%

other PCBs took the same amount of time. At an hourly rate of 30 USD, the labor cost amounts to 6 USD per PCB or 12 USD for the sensor node. In total, then, the sensor node costs 116 USD. Further, we consider the gateway, which costs approximately 123 USD without the SIM card and data plan. Labor for the gateway PCB amounts to 6 USD. The final cost of the gateway is 129 USD. For comparison, a Waspmote ZigBee by Libelium [32] equipped with sensors for air temperature and relative humidity costs approximately 320 USD, and a corresponding gateway costs approximately 310 USD. The cost of the selected A/S-WSN components is less than that of the Waspmote. However, for a more comprehensive comparison, performance aspects such as size, battery lifetime, and quality of service should be considered as well. Furthermore, service aspects such as cloud web applications, support, (paid) access to data, and the cost of the aforementioned were not considered. Nevertheless, this comparison indicates significant savings compared to a commercial, closed-source, and modular solution. When the heat flux sensor, which is very specific to this setup, is left out, the main module makes up the bulk of the cost. A breakdown of the component cost of the main module is shown in Table 6. The primary drivers of cost are the battery, at 35% of the cost, and the radio module, at 25% of the cost. The RTC and the micro SD socket make up another 14% of the cost.

The splitting of the sensor node into a main module and a sensor module required additional components (microcontroller on sensor module, connectors, etc.), which added costs of 14 USD. 41% of the additional cost is for connectors, and 31% of the additional cost is for the additional microcontroller on the sensor module.

Significant cost savings are still achievable for this design. In the current design, the battery is the most substantial cost of the main module. Between the start of the development of the A/S-WSN and the time of writing this article, battery prices have declined by 50%, already yielding cost reductions of 18% for the main module. Further, the capacity of the battery could be reduced, depending on the duration of the measurement campaign. Moreover, the rechargeable battery could be replaced by lower-cost single-use alkaline batteries, potentially lowering the cost for three AAA alkaline batteries and a battery holder to 4 USD, a cost saving of 28% for the main module. If local logging is not used, the RTC and micro SD card socket could be replaced with a more inexpensive timer solution to save 13% of the cost of the main module. The XBee radio modules provide reliable and out-of-the-box mesh networking. However, they are expensive and make up 29% of the cost of the main module, including the needed female pin headers. Switching to a more inexpensive radio transceiver (e.g., ESP32, ESP8266, or RFM69HCW) could save 21% of the total cost of the main module. Changes in the code with regard to a new radio module are straight forward since the concerning code is encased in specific functions. In combination, different batteries, no local logging, and different radio module could reduce the parts cost of the main module by 64% to 26 USD. The cost of the sensor node with a cost-optimized main module and a sensor module for temperature and humidity could be 70 USD (cost of labor included).

For sensor modules, the cost could be reduced by integrating the integrated circuits of the sensors directly to the sensor modules instead of using breakout boards.

4.2. Achievements and challenges

The current revision of the sensor network is inherently modular and expandable. New sensor modules can be added to the framework while reusing the rest of the infrastructure. Even during deployment, sensor modules can be exchanged quickly and easily. New means of communication could be implemented using pin-compatible radio modules with an XBee footprint with the existing main module design. Alternatively, entirely new main modules could be designed and the existing sensor modules reused. New gateways with more connectivity options (Ethernet, Wi-Fi, local storage) could be added as well. Further, changes to the online storage or web interface are possible without the need to alter the rest of the network.

4.2.1. Simplicity

Manufacturing the sensor network requires only a few simple tools and training for only a few hours. Design files and bill of materials are available online. The same is valid for loading the firmware onto the microcontrollers of various components. The software is of a very concise nature. The firmware scripts for the sensor modules are each less than 130 lines of code, not including libraries. The scripts for the gateway and web server are less than 220 lines of code, not including libraries. The only exception is the script for the main module, which encompasses about 1220 lines of code without libraries. Due to the concise nature of the software, most of it can be understood and adopted easily. The first exception is the firmware for the main module. In the current state, it is only partially modular and is quite comprehensive. Succeeding revisions should cut down features in the software that haven't been used and break the code up into modules. The second exception is the sensor module for heat flux sensors, which are currently based on PIC microcontrollers, which are not programmable using the Arduino IDE. We suggest redesigning this sensor module type to be compliant with the Arduino ecosystem.

4.2.2. Ease of use

During the deployment, it is difficult to locate the source of a problem when a sensor node appears to be offline. To reduce this difficulty, we suggest implementing a heartbeat message for the sensor nodes and gateway containing information about the signal strength and battery charge state. In the current version, the only battery charge state transmitted is the one in the low battery warning message. However, regular updates about the health of the sensor node and gateway could increase confidence in the network and help to locate issues faster.

Battery lifetimes of 10 to 13 months were achieved, which are adequate for measurements in buildings during the heating season. The sensor modules for window openings exhibited battery lifetimes of 4 months. In the future, this could be improved with a more power-efficient design of the sensor board. Sensor nodes for the measurement of CO_2-concentration, oil flow, and electricity were powered with a mains adapter, which could easily be installed in most cases. However, when there was no mains power outlet nearby, it would have been preferable to have a wireless sensor node for electricity meters.

Currently, there are four different structures for payload messages: standard sensor module to main module, heat flux sensor module to main module, main module to gateway, and low battery warning. We suggest unifying these message structures for the sake of simplicity.

All components are kept simple, and the necessary skills to modify, manufacture, and program them are easy to obtain. However, the sum of skills and their spread in different domains (electronic and mechanical CAD, reflow soldering, 3D printing, coding in multiple languages, programming various devices, and handling various communication protocols) remain a challenge. Nonetheless, the modularity of the sensor network provides the option to change one part of the sensor network without altering the others. Therefore, to change one part of the network, not all skills are required, e.g., in order to add a new sensor module, one only needs to know soldering and programming Arduino. Furthermore, once the sensor kit is manufactured and the web server is set up, it can be deployed and dismounted by personnel that does not require all skills. The current limitation, however, is the reliability of the gateway. The unreliable performance of the gateway corrupts the measurement data of all associated sensor nodes. This could be improved by including local data storage on the gateway for resilience in case of a server failure. The gaps in the time series data must be reduced.

4.2.3. Data privacy and security

Data privacy and security are limited in the current setup. For the sake of simplicity, we did not use encryption or secure data transfer. We justify this for our application because personal data, e.g., contact information, was not stored on the web server, and the measurement data does not reveal information of a single person in the household.

However, the XBee transceiver offers AES128-bit encryption, which could encrypt the communication from the sensor node to the gateway. For the communication from the gateway to the web server, secure communication over HTTPS is not possible on the 32u4 microcontroller. However, the payload could be encrypted on the current gateway.

Further, an access token as part of the web server URL may be needed to avoid the injection of invalid data into the database by a third party. Security concerns need to be assessed depending on the application of the WSN and might require modification to the hardware to avoid tampering with it.

4.2.4. Further limitations

The accuracy of the sensors could be improved by calibration. For the sake of simplicity, we refrained from sensor calibration. Otherwise, more accurate and more expensive sensors could be chosen and integrated as new sensor modules.

Only whole-house energy consumption was considered in the presented work. In cases, where appliance level energy consumption data is needed, one could use a commercially available plug load monitors than is able to output LED pulses proportional to the energy consumption and use it with the current sensor module for pulse detection. Alternatively, one could design a new sensor module with an integrated power monitor.

5. Conclusion

This sensor-kit is able to acquire a multitude of different measurement inputs. As shown in the testing and validation sub-section, three different data loggers were used to compare against the measurement from the sensor-kit presented-in this work. Furthermore, it is possible to duplicate or expand upon this work because we provide all the documentation needed. Hence, there is no dependency on manufacturers or service providers. There are no hidden or late fees, e.g., for access to data. Furthermore, in this work, we provide full transparency about the cost and demonstrate lower cost compared to commercial counterparts.

The sensor network is used for building performance assessment and building retrofit assessment, including individual building model calibration using measured data from the sensor network [29]. It complements the shift in data collection technologies using remote monitoring systems in building performance assessment [33]. However, the use of the sensor network presented in this study is not limited to the domain of building technology, and we are looking forward to seeing applications in other domains as well, e.g., we suggest integrating radiant temperature sensors and occupancy sensors for more occupant centric research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We thank our colleagues Martin Gähwiler and Lukas Gautschi from the electronics workshop of the physics department of our institution. We further thank the students who assisted in the development and manufacturing of the electronics: Marc Beusch and Alexander Staub. This work was in part supported by the Office for Water and Energy of the Canton St. Gallen (AWE). This research project is financially supported by the Swiss Innovation Agency Innosuisse and is part of the Swiss Competence Center for Energy Research SCCER FEEB&D.

References

[1] International Energy Agency, Organisation for Economic Co-operation and Development, Transition to Sustainable Buildings Strategies and Opportunities to 2050, OECD/IEA, Paris, 2013, https://doi.org/10.1787/9789264202955-en (accessed March 15, 2016).
[2] Energiestrategie 2050, (n.d.). http://www.bfe.admin.ch/energiestrategie2050/ (accessed February 4, 2019).
[3] R. Wu, G. Mavromatidis, K. Orehounig, J. Carmeliet, Multiobjective optimisation of energy systems and building envelope retrofit in a residential community, Appl. Energy 190 (2017) 634–649, https://doi.org/10.1016/j.apenergy.2016.12.161.
[4] A. Martinez, J.-H. Choi, Analysis of energy impacts of facade-inclusive retrofit strategies, compared to system-only retrofits using regression models, Energ. Buildings 158 (2018) 261–267, https://doi.org/10.1016/j.enbuild.2017.09.093.
[5] Z. Nagy, D. Rossi, C. Hersberger, S.D. Irigoyen, C. Miller, A. Schlueter, Balancing envelope and heating system parameters for zero emissions retrofit using building sensor data, Appl. Energy 131 (2014) 56–66, https://doi.org/10.1016/j.apenergy.2014.06.024.
[6] M. Jia, A. Komeily, Y. Wang, R.S. Srinivasan, Adopting internet of things for the development of smart buildings: a review of enabling technologies and applications, Autom. Constr. 101 (2019) 111–126, https://doi.org/10.1016/j.autcon.2019.01.023.
[7] P. Jamieson, J. Herdtner, More missing the Boat-Arduino, Raspberry Pi, and small prototyping boards and engineering education needs them, 2015 IEEE Frontiers in Education Conference (FIE), 2015, pp. 1–6, , https://doi.org/10.1109/FIE.2015.7344259.
[8] D. Hernández, H. Trejo, E. Ordoñez, Development of an exploration land robot using low-cost and open source platforms for educational purposes, J. Phys. Conf. Ser. 582 (2015) 012007, , https://doi.org/10.1088/1742-6596/582/1/012007.
[9] Š. Kubínová, J. Šlégr, Physics demonstrations with the Arduino board, Phys. Educ. 50 (2015) 472–474, https://doi.org/10.1088/0031-9120/50/4/472.
[10] H. Tsukamoto, Y. Takemura, H. Nagumo, I. Ikeda, A. Monden, K. i Matsumoto, Programming education for primary school children using a textual programming language, 2015 IEEE Frontiers in Education Conference (FIE), 2015, pp. 1–7, , https://doi.org/10.1109/FIE.2015.7344187.
[11] Industrial IoT - PC-based Automation - Siemens, (n.d.). http://w3.siemens.com/mcms/pc-based-automation/en/industrial-iot/Pages/Default.aspx?tabcardname=simatic%20iot2000%20io-shield (accessed March 19, 2018).
[12] FluxDAQ+|Heat flux sensor and thermocouple data logger, FluxTeq heat flux sensors|national lab-approved heat flux sensors. (n.d.). http://www.fluxteq.com/heat-flux-thermocouple-data-logger (accessed March 19, 2018).
[13] A. Martín-Garín, J.A. Millán-García, A. Baïri, J. Millán-Medel, J.M. Sala-Lizarraga, Environmental monitoring system based on an open source platform and the internet of things for a building energy retrofit, Autom. Constr. 87 (2018) 201–214, https://doi.org/10.1016/j.autcon.2017.12.017.
[14] A. Chakraborty, R.R. Rout, A. Chakrabarti, S.K. Ghosh, On network lifetime expectancy with realistic sensing and traffic generation model in wireless sensor networks, IEEE Sensors J. 13 (2013) 2771–2779, https://doi.org/10.1109/JSEN.2013.2260147.
[15] N.K. Suryadevara, S.C. Mukhopadhyay, Wireless sensor network based home monitoring system for wellness determination of elderly, IEEE Sensors J. 12 (2012) 1965–1972, https://doi.org/10.1109/JSEN.2011.2182341.
[16] W.-S. Jang, W.M. Healy, M.J. Skibniewski, Wireless sensor networks as part of a web-based building environmental monitoring system, Autom. Constr. 17 (2008) 729–736, https://doi.org/10.1016/j.autcon.2008.02.001.
[17] S. Noye, R. North, D. Fisk, A wireless sensor network prototype for post-occupancy troubleshooting of building systems, Autom. Constr. 89 (2018) 225–234, https://doi.org/10.1016/j.autcon.2017.12.025.
[18] M. Karami, G.V. McMorrow, L. Wang, Continuous monitoring of indoor environmental quality using an Arduino-based data acquisition system, J. Build. Eng. 19 (2018) 412–419, https://doi.org/10.1016/j.jobe.2018.05.014.
[19] S. Abraham, X. Li, Design of a low-cost wireless indoor air quality sensor network system, Int. J. Wireless Inf. Networks 23 (2016) 57–65, https://doi.org/10.1007/

s10776-016-0299-y.
[20] P. Kumar, C. Martani, L. Morawska, L. Norford, R. Choudhary, M. Bell, M. Leach, Indoor air quality and energy management through real-time sensing in commercial buildings, Energ. Buildings 111 (2016) 145–153, https://doi.org/10.1016/j.enbuild.2015.11.037.
[21] S.S. Bhunia, S. Roy, N. Mukherjee, IEMS: indoor environment monitoring system using ZigBee wireless sensor network, Proceedings of the 2011 International Conference on Communication, Computing & Security, ACM, New York, NY, USA, 2011, pp. 142–145, , https://doi.org/10.1145/1947940.1947971.
[22] R. du Plessis, A. Kumar, G. Hancke, B. Silva, A wireless system for indoor air quality monitoring, IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society, 2016, pp. 5409–5414, , https://doi.org/10.1109/IECON.2016.7794087.
[23] D. Brunelli, I. Minakov, R. Passerone, M. Rossi, POVOMON: an ad-hoc wireless sensor network for indoor environmental monitoring, 2014 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems Proceedings, 2014, pp. 1–6, , https://doi.org/10.1109/EESMS.2014.6923287.
[24] J.-J. Kim, S.K. Jung, J.T. Kim, Wireless monitoring of indoor air quality by a sensor network, Indoor Built Environ. 19 (2010) 145–150, https://doi.org/10.1177/1420326X09358034.
[25] Y. Huang, L. Hu, D. Yang, H. Liu, Air-sense: indoor environment monitoring evaluation system based on ZigBee network, IOP Conference Series: Earth and Environmental Science 81 (2017) 012208, , https://doi.org/10.1088/1755-1315/81/1/012208.
[26] A.S. Ali, Z. Zanzinger, D. Debose, B. Stephens, Open source building science sensors (OSBSS): a low-cost Arduino-based platform for long-term indoor environmental data collection, Build. Environ. 100 (2016) 114–126, https://doi.org/10.1016/j.buildenv.2016.02.010.
[27] M.W. Ahmad, M. Mourshed, D. Mundow, M. Sisinni, Y. Rezgui, Building energy metering and environmental monitoring – a state-of-the-art review and directions for future research, Energ. Buildings 120 (2016) 85–102, https://doi.org/10.1016/j.enbuild.2016.03.059.
[28] Architecture-building-systems/wireless-sensor-network, GitHub. (n.d.). https://github.com/architecture-building-systems/Wireless-Sensor-Network (accessed April 12, 2017).
[29] C. Deb, M. Frei, J. Hofer, A. Schlueter, Automated load disaggregation for residences with electrical resistance heating, Energ. Buildings 182 (2019) 61–74, https://doi.org/10.1016/j.enbuild.2018.10.011.
[30] C.-A. Roulet, F. Foradini, Simple and cheap air change rate measurement using CO_2 concentration decays, Int. J. Vent. 1 (2002) 39–44, https://doi.org/10.1080/14733315.2002.11683620.
[31] M. Frei, Z. Nagy, A. Schlueter, Towards data-driven building retrofit, Proceedings of International Conference CISBAT 2015 Future Buildings and Districts Sustainability from Nano to Urban Scale, LESO-PB, EPFL, 2015, pp. 963–968.
[32] Waspmote-open source sensor node for the internet of things|ZigBee, Sigfox, LoRaWAN, 3G/4G compatible|Libelium, (n.d.). http://www.libelium.com/products/waspmote/(accessed July 12, 2018).
[33] B.R.K. Mantha, C.C. Menassa, V.R. Kamat, Robotic data collection and simulation for evaluation of building retrofit performance, Autom. Constr. 92 (2018) 88–102, https://doi.org/10.1016/j.autcon.2018.03.026.