applied sciences

MDPI

Article

A Novel Software Architecture Solution with a Focus on Long-Term IoT Device Security Support

Ivica Dodig [1,2,*], Davor Cafuta [1,2,*], Tin Kramberger [1] and Ivan Cesar [1]

[1] Department of Information Technology and Computing, Zagreb University of Applied Sciences, 10000 Zagreb, Croatia; tin.kramberger@tvz.hr (T.K.); ivan.cesar@tvz.hr (I.C.)
[2] Multimedia, Design and Application Department, University North, 42000 Varaždin, Croatia
* Correspondence: ivica.dodig@tvz.hr (I.D.); davor.cafuta@tvz.hr (D.C.)

Abstract: This paper presents a solution for upgrading a previous device model to an Industry 4.0 smart device, with the goal of maintaining high compatibility. A novel IoT architecture is presented that satisfies the characteristics of a smart device. We analysed existing IoT architectures and proposed a new architecture to achieve long-term security and usability. To ensure long-term security, we eliminated the possibility of device configuration outside the immediate vicinity of the device with a dedicated protocol. The security concepts of the existing architectures were also analysed and further modified. To improve compatibility with previous device models, we propose a new method to collect data from sensors by introducing a multithreaded microcontroller. We propose additional software components to ensure factory programming, maintenance, and cloud Big Data analysis. Based on our experiments, we adapted the algorithm to increase the accuracy of the temperature and flow sensors by using a temperature calibration device and known flow cycles. Measurement results are presented to confirm the successful upgrade. We designed a hardware architecture to ensure compatibility with previous and future device models. Issues with previous sensors encountered during the upgrade were discussed and resolved. A novel software architecture based on security for long-term IoT devices is proposed.

Keywords: smart device; sensors; beverage cooling device; Industry 4.0

Citation: Dodig, I.; Cafuta, D.; Kramberger, T.; Cesar, I. A Novel Software Architecture Solution with a Focus on Long-Term IoT Device Security Support. *Appl. Sci.* **2021**, *11*, 4955. https://doi.org/10.3390/app11114955

Academic Editor: Ioannis Chatzigiannakis

Received: 23 April 2021
Accepted: 24 May 2021
Published: 27 May 2021

1. Introduction

Smart devices and the presence of the Internet provide innovative solutions to various challenges, such as in business, government, public and private industries, and quality of life in general [1]. The IoT is multidisciplinary in a variety of applications such as smart surveillance systems, smart agriculture, smart monitoring and security, smart energy consumption, and smart cities and homes [2–4]. The IoT [5] concept is increasingly mentioned in the context of the connectivity of multiple electronic devices over the Internet. It is a platform that can process data and communicate with each other independently or in interaction with humans over a wireless or wired connection. In convergence with electronic systems and micro-services based on cloud technology, IoT devices contribute to the overall compatibility of the intelligent ecosystem [6]. They also enable the interconnection of ubiquitous devices equipped with various sensors, enabling innovative new services and contributing to save time and money while improving the quality of life [7].

The IoT ecosystem consists of networked smart devices that collect and respond to data from the environment using built-in processors, sensors and communication hardware. One of these devices is our beverage cooling device equipped with temperature sensors, flow sensors, and a unit that measures energy consumption. The device interacts with the main components: compressor, cooling fan, and pump via the main board to form the sensitive IoT ecosystem. All collected sensor data are shared by IoT devices through a connection to an IoT gateway or other edge device, through which the data are sent to the

cloud for storage and further analysis. The data are partially analysed in the edge device layer before being sent to the cloud [8].

To analyse large amounts of collected data from sensors or other smart communication devices in real time, Big Data technology combined with artificial intelligence and business intelligence is essential [9]. The efficient analysis of collected data inherently improves the quality of life, enables more informed decisions [10], reduces production costs and generates higher revenues.

When automation is applied to industry, it is referred to as industrial IoT (IIoT)—a subset of IoT [11,12]. In the IIoT concept, one of the main challenges is to ensure a high standard of reliability and security while working within the resource constraints of internet availability and computing power [7]. The telemetry of the product implemented with IIoT can be used to intelligently monitor the internal processes of the device throughout its life cycle. The life cycle consists of the manufacturing process, commissioning, service cycle and finally, the decommissioning of the product. The enhanced life cycle is part of the goals of Industry 4.0 [13] and allows the manufacturer to constantly monitor product performance in different environments. Life cycle monitoring, used for high-end products such as the ones in the automotive industry, can also be applied in other areas, in our case, manufacturing a beverage cooling device.

The focus shifts from the production process to the life cycle of a product. Industry 4.0 refers to smart manufacturing, but also to the constant monitoring of the product during its life cycle in order to make decisions based on the information obtained. In Industry 3.0, the product was mostly improved through experience. Figure 1 illustrates the main changes from Industry 3.0 to Industry 4.0 [14].

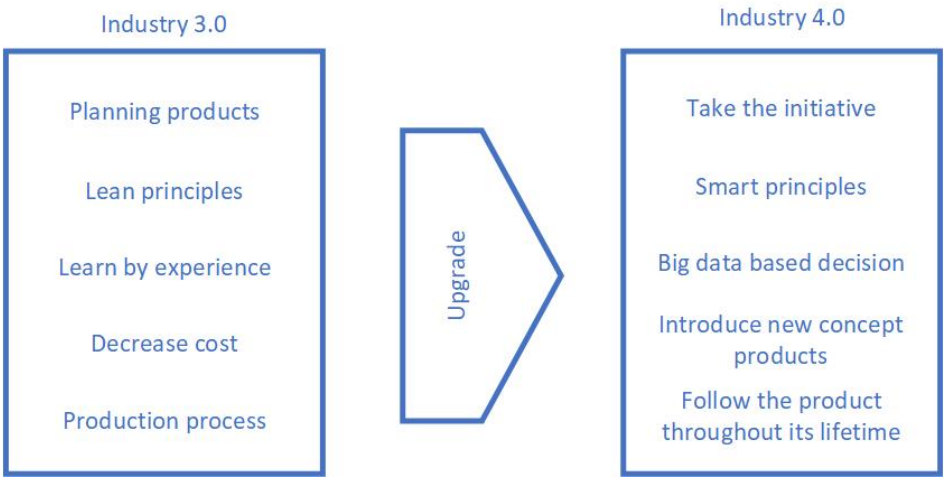

Figure 1. Industry 3.0 vs. Industry 4.0: Industry 4.0 focuses on taking the initiative and following the product throughout its lifetime, often based on informed, data-driven decisions.

In the modern world, one of the most important issues in implementing Industry 4.0 is security. In this paper, we propose a novel security-oriented software architecture concept suitable for a device with a lifetime of 15 years or more.

Standard IoT architectures define three common layers: perception layer (device), network layer (connect), and application layer (manage) [15]. This architecture provides top–down and bottom–up communication of from layers with implemented security principles. Implemented security, especially cryptography, can become the issue considering long-term device usage. Unlikely device hardware updates combined with exploits in network security protocols pose a serious problem. We propose a new software architecture to address this problem by limiting top-down communication and preventing the possibility of external device reconfiguration from the cloud. To provide additional security, the network layer is enhanced with two new layers that ensure strict separation between

device operation and communication. Therefore, a potential exploit in the microcontroller thread of device communication will not affect device operation.

Guided by the newly proposed software architecture, an upgrade concept of the existing device to a smart network-connected Industry 4.0 device is presented. One of the main objectives was to seamlessly upgrade all previous version devices manufactured by Oprema d.d. and used in the beverage industry today to a telemetry-based smart device. To achieve compatibility, a minimal change is proposed. Problems introduced with the new components are discussed in detail and the necessary adjustments are presented. New algorithms were developed to optimise the output of temperature and flow sensors. An experiment was conducted to further adjust the algorithms to achieve accurate sensor calibration.

This paper is organised as follows. In Section 2, we discuss the current software architecture models and the security problem in their implementation, describing related work and software architectures in the field. Section 3 presents a new proposed software architecture and discusses the rationale behind the proposal. Later, in Section 4, the concept of smart device extension is presented along with the problems of temperature measurement, flow sensor measurement and the security concept of the device. The results of the device measurement are presented. The paper is concluded with Section 5, where the advantages of our proposed system are presented and discussed.

2. Related Work

With the expanding and growing IoT usage, scientific research has focused on improving IoT architecture to improve scalability and mitigate security threats in complex IoT systems.

As one of the main drivers of Industrial Revolution 4.0, the goal of IoT is to provide connectivity through the application of a computer network, path, or service. Today, the IoT provides innovative solutions for advanced use in healthcare, food supply and production, mining, firefighting, transportation, and logistics [16].

Network connectivity allows for longer process monitoring with the goal of faster response and detailed information about the operation of the entire monitored system. An example could be a greenhouse system that enables sensor monitoring to manage greenhouse processes [17].

Kalsoom et al. discusses the different types of sensor technology used in the manufacturing industry. This paper provides a broad overview of the existing literature on Industry 4.0 [18].

Non-thermal processing in food Industry 4.0 and sustainability is discussed in [19]. It gives an overview of the application of non-thermal processing with optimisation to enable the concept of smart factories in terms of Industry 4.0 (sensors, IoT, Big Data, AI). The conclusion welcomes interdisciplinary research and contributions in this field.

Petrillo et al. addresses a problem of designing innovative Integrated Vehicle Health Maintenance (IVHM) systems leveraging the potential of the Internet of Vehicles (IoV). The authors present a novel cloud-based architecture that allows ingesting a large amount of data from multiple vehicles for the prognostic analysis of automotive components [9].

Barriga et al. presented a review paper on smart parking technologies. The paper provides a comprehensive classification of sensor solutions according to the ease of installation, detection autonomy and economy cost. Network security is considered as an issue since it significantly affects the reliability of the system [20].

Tastan et al. proposed an IoT-based personalised air quality measurement and monitoring system using air quality sensors and DIY approach. The measurement data are sent to the cloud via Blynk mobile interface. The paper validates the Industry 4.0 concept of a simple but effective smart device [21].

Similarly, IoT-based indoor air quality monitoring platform is proposed in [22]. The platform is equipped with an LTE modem that enables direct connection to the cloud without local storage and analysis. Amazon Web Service (AWS) was used as a web server

to analyse, visualise and present the collected data. An alert system which signals the air quality after web analysis is presented.

Sunny et al. presented a case study of an IoT device for monitoring harsh environments. The paper proposes a low-cost sensor system based on commercial off-the-shelf components. The proposed system provides a remote sensing capability for a nuclear waste storage facility [23].

A similar research by Francisco Maroto-Molina et al. presents a low-cost solution to enable the motorisation of an entire herd. The paper uses various tags/collars to create a low-cost system with off-the-shelf components [24].

In their paper, Matteo D'Aloia et al. proposed a low-cost IoT system for real-time monitoring based on the microservice paradigm, offering modularity, scalability and integration of heterogeneous and legacy systems [25].

As part of the Convergence project research, Elise Saoutieff et al. proposed a low-power sensing platform for health and environmental monitoring with embedded sensors and external sensors. The paper discusses the development and testing process of the wearable monitoring device [26].

Mohammad Shahidul Islam et al. presented a paper on the overall monitoring system of human body signal through biomedical sensors, MySignals and LoRa wireless network system. The standard software architecture was used: physical access layer, network layer and service and application layer. The paper presents security issues and solutions for each software architecture layer [27].

The indoor thermal comfort monitoring system [28] utilises standard IoT software architecture: perception layer (device), network layer (connect) and application layer (manage). The application layer does not provide an application, but a service used to collect data to run the desktop application locally. Security has not been considered in this work.

Chu et al. proposed an extensible modular IoT-based power extender for temperature control. The paper proposes a master-slave software architecture based on public or private cloud connectivity. The paper focuses on applying the concept to legacy devices and equipment to become smart machines [29].

A four-layer IoT architecture has been proposed, consisting of application, processing, transport and perception layer [30]. This architecture is the backbone of a smart and intelligent irrigation system based on edge computing proposed by Munir et al. The perception layer is a physical layer. The transport layer is used to transport data across networks. The processing layer stores, inspects, and processes large amounts of data coming from the transport layer. Finally, the application layer provides application-specific functions to the end user.

Ungurean et al. proposed a new software architecture composed of four layers. The things layer uses industrial interconnected buses to connect devices. The data provider layer is used to collect data from the field buses and provide it in a packaged form. The middleware layer is used to exchange information over the Internet. The protocol is based on the publisher–subscriber paradigm. Finally, the application layer provides the connection between the user and things. All communication is handled bilaterally. Security is not considered in this work [31].

In their paper, Yelamarthi et al. proposed a five-layer architecture that is application-driven, focusing on the lower layers in the agricultural environment. The sensor and actuator layers ingest raw data and provide digitised environmental data. The embedded low-power processor layer provides data tagged with a source identifier and enables coarse data analysis. The wireless transceiver layer sends data packets for detailed analysis. The internet gateway sends data for storage and retrieval. Finally, the last layer provides the data to the end user as an application management cloud server [32].

On the other hand, Kraijak et al. propose a five-layer software architecture: perception, network, middleware, application and business. The business layer covers all IoT application and service management [6].

Sattar et al. proposed an android-based wound care solution to detect the level of feasibility of the environment for healing. The sensor data are analysed by a trained model to predict the environment feasibility for wounds in three different environments with excellent accuracy. In this work, a modified software architecture has been developed which is more cloud oriented [33].

Malhotra et al., in the review paper "Internet of Things: Evolution, Concerns and Security Challenges", presented various software architectures and security threats on standard IoT layers. The paper classifies and discusses IoT vulnerabilities [34].

Security challenges are discussed in detail in [35], with a focus on IoT edge computing. In their review, the problems in a standard software architecture consisting of perception, network and application layers are addressed. Conventional and AI-driven privacy protection of edge-based IoT are compared and elaborated. The paper also considers open challenges and issues for securing IoT services in edge computing [35].

Ahanger et al. proposed an IoT-inspired intrusion detection framework for a smart home security system. The modular framework is based on Fog layer as multiple devices communicate in the private network. The authors propose the network security as an important domain to be considered in future work [36].

The majority of the analysed papers are based on the current usability of IoT devices and provide a way to improve the features of the proposed device. The papers do not consider the long-term usability of the simple smart IoT device. For simple smart devices, hardware upgrade is not an option, security protocols can easily become obsolete and the hardware eventually becomes too old for a software upgrade. We proposed a software architecture to solve the problem by separating device operation and device communication in both software and hardware architecture.

3. Software Architecture

This chapter proposes a novel software architecture based on existing software architectures presented in related work. Modern software architecture for the Internet of Things requires compatibility with other devices, operating systems, and the cloud. Modularity and scalability should also be considered to achieve easy upgrade and service process. In today's world, we believe that one of the most important factors should be security.

The data collected are not considered public, but due to the network dispersion of the devices, the main focus was to prevent the data from being tempered with rather than viewed. Invalid configuration can render the device unusable. For these reasons, we propose a change in the software architecture to disable device control from the cloud. The device can only be controlled by the service application through the authorised service or device owner.

The proposed system architecture is shown in Figure 2. The lowest layer is the physical layer where all the sensors of the device are located. The sensors used on this layer are: temperature, flow sensors and sensors measuring current consumption and nominal voltage. This layer is the first layer that forms the set of sensors needed for the operation of the device and telemetry.

The operation and data processing layer was implemented as the first thread of the two-threaded ESP32 microcontroller. Data processing collects and processes the data received from the physical layer and verifies the data integrity according to the expected range. The specified sensor data were used and processed in the device operation algorithm along with the configuration parameters to make the device operational.

These two layers must be functional for the device to operate. In the original device model, these two layers were implemented as part of outdated microcontroller that used the serial port as a debugging point, and the device was configured via three buttons using the two-line display LCD. The configuration is now enabled through an additional service and transport layer.

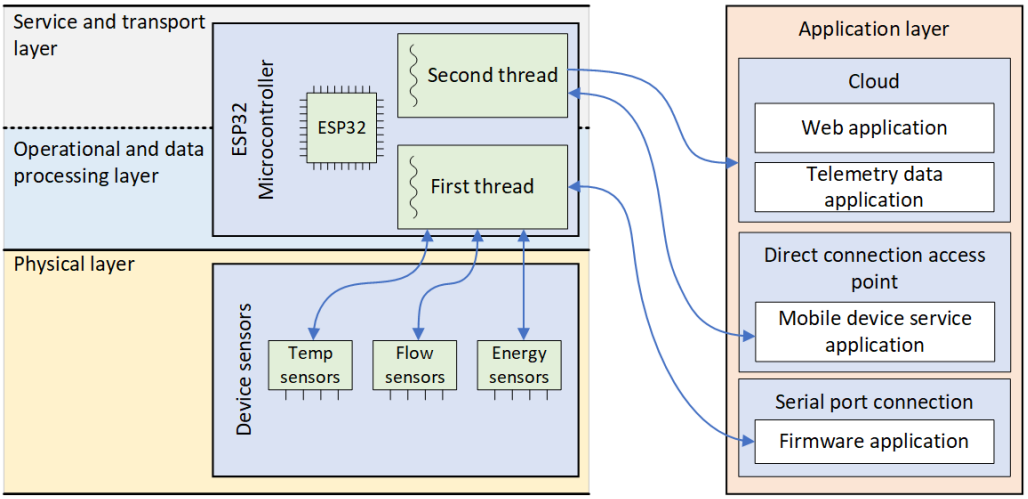

Figure 2. Proposed IoT architecture. The connection between the service and transport layer and the mobile device service application is achieved via direct connection access point. The connection between the cloud and service transport layer can be achieved via WiFi connection and GSM connection.

Completely independently, the second microcontroller thread is responsible for communicating with the mobile service application and sending telemetry data to the cloud. This thread implements a service and transport layer. The mobile service application can only connect to the device through a direct connection to the service and transport layer. The service and transport layer exposes a WiFi access point in a short period of time after booting the device. Device configuration is set through the mobile service application, as the new device model no longer has a set up interface with the display unit. In addition to setting the parameters, the mobile service application calibrates the flow sensors since the installation parameters affect the actual calculation of the flow sensors. This process is performed by authorised service personnel during device installation. The user role in mobile service application is configured in the cloud application. Simultaneously, the service and transport layer groups the telemetry measurement data and forwards them to the cloud. This process is performed via a local wireless WiFi client (part of the local network at the device installation site) or using a built-in modem with GSM access.

To ensure the security of the device installation prior to configuration by the mobile service application, the device acts as a wireless access point to which the mobile service application connects via a security access point. Security access point settings are unique to every device and an authentication QR code is provided in the device manual and on the device casing. Additional security is achieved by activating the access point only at a short time interval when the device has started.

Since it is necessary for the service and transport layer to communicate simultaneously with the mobile service application and the cloud over the Internet, it should be possible for the device to simultaneously be a WiFi client on the local network and provide a secure wireless access point service in the defined time frame upon booting. For these reasons, an ESP32 microcontroller is used as it supports simultaneous operation of the wireless network in client mode and access point mode. In case a different connectivity option is required in client mode, the device enables the usage of WiFi, or the direct connection to the GSM network through a serial connection.

The dual-thread operation of the ESP23 microcontroller allows the operation and data processing layers to operate simultaneously with service and transport layers. The device can be used without network connectivity configured through an access point. Once configured, the device can also be used without telemetry or any other smart features if needed. However, with the telemetry option enabled, the device becomes part of a network that sends telemetry data to the cloud. The wireless client connection used to send data to the cloud does not allow inbound connections, and therefore reconfiguration via LAN is not possible.

Dual-thread operation additionally improves security in the event of an exploit running on the service and transport layer thread. A successful exploit blocks the thread, disabling the telemetry sending process in the worst case scenario (configuration time connectivity has likely expired). The operation and data processing layers continue to run as an independent thread, so the device is fully functional.

The second thread is very similar to the performance of an edge layer, which allows the reconfiguration of devices and communication with the cloud at the defined frequency. The telemetry data were sent to the cloud over an insecure communication channel. The data are protected from tampering with digital signatures by providing a unique communication key to each device. The address of the cloud server is determined by the mobile service application. The address may vary depending on the location and client. For example, breweries being larger clients may require the installation of a separate telemetry cloud.

When the device is manufactured, a factory programming application is used to initialise the microcontroller. Programming directly affects the first three layers of the architecture. The factory application is responsible for calibrating the sensor and verifying the device components. The specified application generates unique communication keys to ensure the security of the mobile service application. The security protocols are described in the next section in detail.

The device can also be upgraded through a mobile service application during regular sanitation service or in case of emergency service caused by the device failure. Upgrades are not available through the cloud due to possible disruptions during the upgrade process. There is no indicator or display unit on the device to notify the user that an upgrade is in progress. Disruption of the device during the upgrade may result in the permanent failure of the device.

The cloud as an application layer receives telemetry data over the Internet. Telemetry data can be inspected by the end user of the device, the owner of the device, and the manufacturer. From the telemetry data, data on beverage consumption, hygiene, regular service, and the energy consumption of the device at various specific locations can be obtained. The cloud analyses and forwards errors about the operation of the device and ensures timely service of the device.

In our proposed architecture, the cloud communication with the device is disabled due to the risk of damaging the device or its environment by setting undesirable operating parameters of the device. For example, decreasing the temperature of the bath may lead to the complete freezing of the device and consequently, flooding the environment once the device temperature increases.

We believe that as much as the security protocols and communication keys provide adequate security at the time of device production, there is a high probability that they will be compromised during an average device lifetime of 15 years. Continuous security protocol upgrades can be performed during maintenance intervals up to the limits of hardware resilience on the device motherboard. Stronger security protocols require hardware upgrades within the average lifetime of the device, so we concluded that cloud-to-device connectivity should not be implemented (the device's connectivity to the cloud is used to send telemetry data). For this reason, we propose a modified software architecture.

4. Device Architecture

When designing a smart beverage cooler, many factors had to be considered to ensure that the device would function as smoothly as possible during operation. We modelled the architecture after the standard IoT architecture, but introduced several important changes due to specific requirements. Some of the primary requirements that needed to be addressed were: the beverage cooling device should send data to the cloud and be configured and maintained using the mobile service application; it should be able to modify device operating parameters in the event of device part replacements should be ensured; and device control or upgrade via the cloud for security reasons should be disabled. The

aim of the project is to improve the device to a smart IoT device according to Industry 4.0 standards.

For this reason, we propose a different software architecture that meets all the device requirements. In this Section, the physical device architecture is presented. Each part of this Section presents different parts of the device in the system architecture model.

4.1. Hardware Layer

The device consists of six separate copper tubes passing through the bath filled with the cooled substance. The copper tubes are filled with the beverage liquid to be cooled and are formed in a coil structure to efficiently increase the tube length. The device uses a bath temperature sensor to directly measure the temperature of the cooled substances. The cooling of the refrigerated substance is achieved by circulating the refrigerant inside the aluminium tube formed as a coil, which is inserted into the bath in parallel with the copper tubes. The cooling effect is based on the transformation of the refrigerant from a liquid state to a gas. This process, known as evaporation, cools the substance in the bath and the copper tubes immersed in the bath. This is the basic principle on which most refrigerators work.

To start the evaporation process and convert the refrigerant from a liquid state to a gas, the pressure on the refrigerant must be reduced through an outlet called the capillary tube. To keep a refrigerator running, the gaseous refrigerant must be converted back to a liquid state, so the refrigerant must be compressed to a higher pressure and temperature. Once the refrigerant is compressed to a higher pressure and temperature, the refrigerant is under high pressure and becomes hot. It must be cooled in the condenser, which is located at the back of the refrigerator, so that the contents can be cooled by the ambient air. This cooling process can be aided by a fan. As the refrigerant in the condenser cools (still under high pressure), it returns to a liquid state. This refrigerant can be reused in the evaporation process. The process described is shown in Figure 3. To monitor the compressor activity, a temperature sensor measures the temperature of the compressor and condenser to turn on the fan when needed to reduce the unit temperature, thus extending the unit lifetime. In addition, a time delay is added if there is a repeated request for compressor activity.

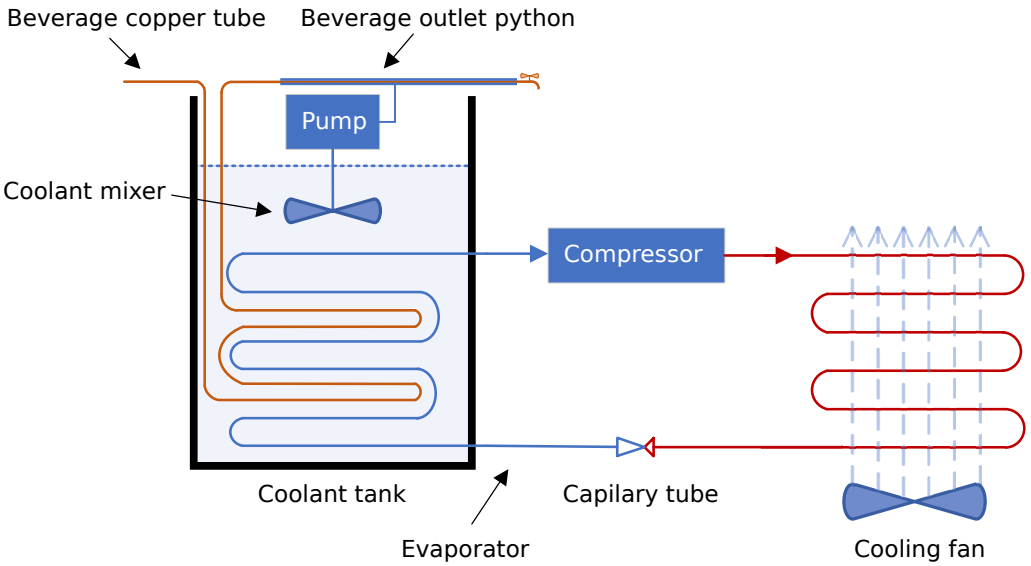

Figure 3. Working principle of our smart beverage cooling device.

The cooling process is responsible for the cold drink in the pipes. The user of the device sets the desired beverage temperature through the configuration, which is verified by the temperature sensor immersed in the bath. In addition, a tolerable deviation from the desired temperature is configured. Only when the temperature rises from the desired temperature by the tolerable deviation is the cooling process switched on. Similarly,

the cooling process is stopped when the desired liquid temperature in the bath is reached. The frequency of switching on the cooling process depends on the external environmental temperature and the device insulation. It is therefore recommended to store the device in a darker and cooler place.

The inlets of the copper tubes with beverages are connected to the beverage sources. The outlets of the copper tubes are connected to the rotary flow meters. The outlets of the flow meter are connected to tubes arranged to form the python tube. The python tube with six beverage lines and a central return tube is shown in Figure 4. The python tube connects the copper tubes to the beverage taps, which are mounted for serving beverages. To ensure the cooling of the python tube, the cooling substance is pumped in circulation in the central tubes which are integrated into the python tube.

Figure 4. Python tubing with six beverage lines and a return central line.

A temperature sensor is installed at the inlet and outlet of the central tubes to measure the temperature difference. The pump (mixer) is responsible for achieving circulation and to cool the copper tubes in the bath evenly, by distributing the liquid and pressing it into the closed circuit. The temperature difference at the central tubing is used to regulate the power of the pump (mixer) and the operation of the compressor to reduce power consumption. Finally, the last temperature sensor is mounted on the unit casing to measure the ambient temperature. The ambient temperature affects the running time of the compressor and the fan, so this information can be used to detect the malfunctions of the unit.

Temperature and flow sensors are connected to the main processing unit. The passive sensors are connected directly to the 8-bit microcontroller. The temperature sensors are NTC thermistors connected to a 10-bit analogue-to-digital converter, which passes the data to the microcontroller via the I2C bus. Rotating flow sensors are directly connected and counted via the digital input line of the microcontroller.

The microcontroller executes the device algorithm to provide temperature hysteresis cooling system based on the bath temperature sensor. The device has a two-line liquid crystal display that shows the readings from the sensors and allows the reading and setting of the bath base temperature and hysteresis differential temperature. Cumulative flow sensor readings and settings are stored in EEPROM and are retained even if the unit loses power. Other sensor readings are displayed in real time and are not stored or transmitted to an output device. The unit uses a simple fault detection algorithm based on compressor temperature and the deviation of the average python tube temperature from the bath temperature.

For power failure detection, the motherboard is equipped with current sensors that provide information about compressor, pump or fan failure. Due to the low resolution of the current sensors, they cannot provide a detailed current measurement or energy footprint of the device.

4.2. Improving the Hardware Layer

The intelligent beverage cooling system was developed in cooperation between the company Oprema d.d. and Zagreb University of Applied Sciences within the project "Development of the new generation Eco Smart product of the company Oprema d.d.", funded by European Union [37]. One of the final versions of the device is shown in Figure 5:

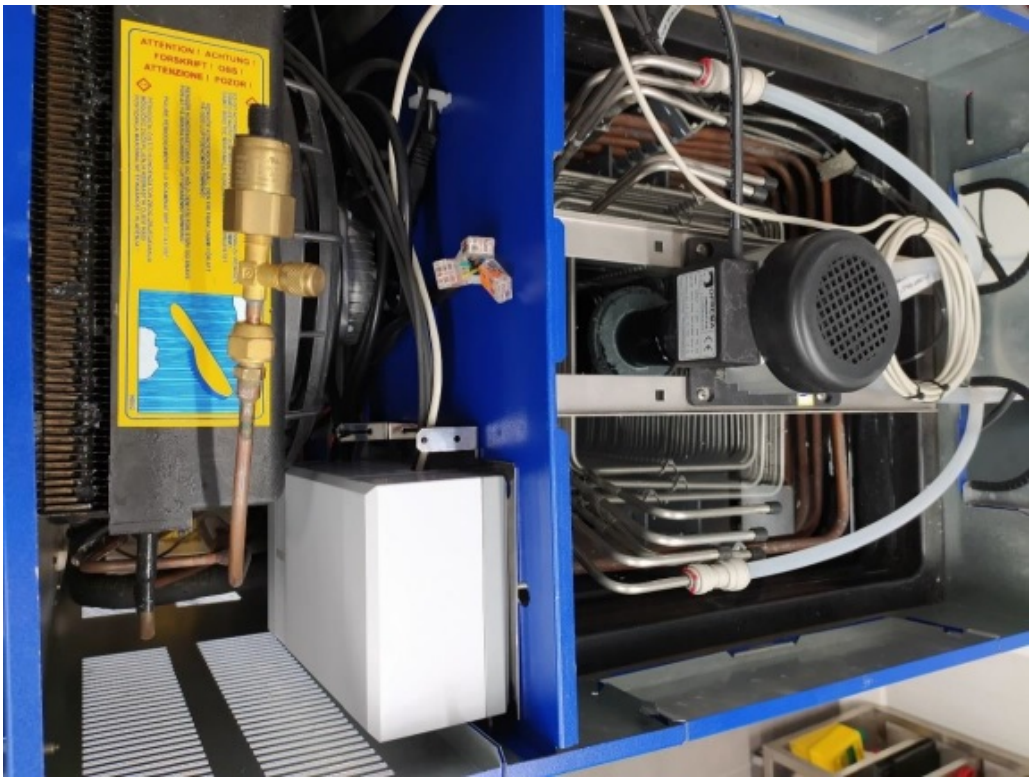

Figure 5. Beverage cooling device upgraded with smart option produced by Oprema d.d.

A new type of microcontroller is introduced for the implementation of telemetry in the device. The motherboard is introduced with the ESP32 microcontroller, as it supports two modes of operation of the wireless module and has two CPU cores, one of which is responsible for the operation of the device and data acquisition and processing, and the other is responsible for the connection with the cloud or mobile application. The idea was to introduce a new microcontroller to an existing motherboard, avoiding additional cost. The test phase is shown in Figure 6.

During implementation, additional fixes were made to the motherboard as the new microcontroller eliminated the need for other electronics required by the previous microcontroller. For example, the new microcontroller has integrated multiple 12-bit analogue/digital converters and has fast EEPROM memory. In order to provide a smart upgrade for previous version devices and simplify service procedures, the goal of the project was to retain all sensors and wiring from the previous device. The upgraded motherboard is shown in Figure 7. This chapter presents the upgrades that were achieved to meet the proposed goals.

The device uses six NTC thermistor sensors. The testing of temperature deviation was carried out at the Zagreb University of Applied Sciences and in the laboratories of the company Oprema d.d. The measurement was performed by reading data from the microcontroller delivered to a database server. Simultaneously, with the operation of the device, calibrated temperature sensors DS18B20 [38] were installed via the I2C bus. The sensors were connected to a microcontroller and delivered data to the same database server over the wireless network. The temperature obtained from the device was compared with the actual temperature measured by the calibrated sensor.

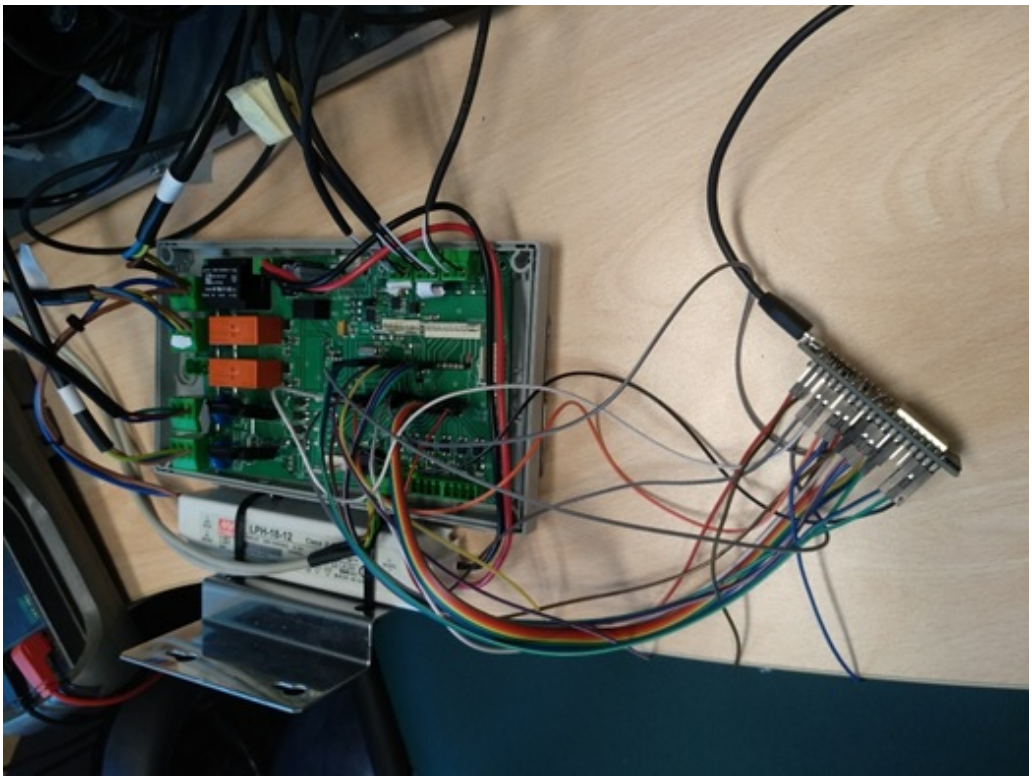

Figure 6. Testing concept: introduce a new microcontroller on previous device version motherboard.

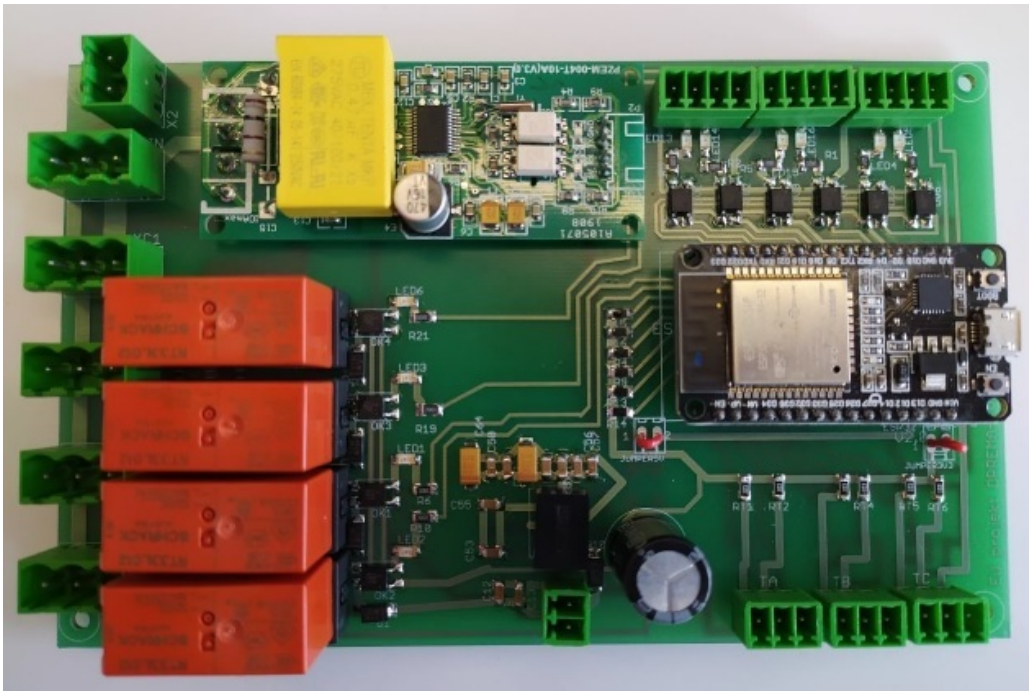

Figure 7. New motherboard with relay switching option.

Experiments showed that a previous model of the device had slight deviations around the reference temperature of the NTC thermistor (25 degrees Celsius) and significant deviations (several degrees Celsius) around the expected temperature of the bath-cooled substance. The NTC thermistor used by the device is ELIWELL SN8DED11502C0 [39]. In the component data sheet, the resistance is given as 10 kOhm at 25 degrees. The resistance changes depending on the temperature according to the specification table given by the manufacturer. The slope of the curve depends on the coefficient of the NTC thermis-

tor. To calculate the temperature accurately, the previous device uses the Steinhart–Hart Equation (1):

$$\frac{1}{T} = A + Bln(R) + C(ln(R))^3 \quad (1)$$

The coefficients A, B, C are the Steinhart–Hart coefficients determined by the factory settings of the sensor. R is the resistance value that will be calculated by the microcontroller. T is the determined temperature. The voltage on the pin of the microcontroller generated on a resistor of the parallel divider is proportional to the value R. In the divider, one of the resistors is fixed and accurate while the other resistor is an NTC thermistor. In the device schematic, the NTC thermistor is connected in a divider with a fixed and accurate resistance of 10,000 ohms. The read value is multiplied by 500 (the referenced voltage on previous version microcontroller increased by the required accuracy of two decimal places) and divided by the maximum number value of the analogue/digital converter. The 12-bit analogue/digital converter provides number values ranging from 0 to 4095. The *measured voltage* equation is given in (2):

$$measuredVoltage = (int)\frac{500 * readValue}{4095} \quad (2)$$

The *resistance* is determined as the maximum magnitude of the analogue/digital converter divided by the *read value* from the analogue/digital converter as stated in (3):

$$Resistance = \frac{4095}{readValue} - 1 \quad (3)$$

Since the divider fixed value is set to 10,000 ohm, the *resistance* is corrected by the fixed amount in (4):

$$Resistance = \frac{10,000}{Resistance} \quad (4)$$

The amount is then divided by the factory *resistance* of the NTC thermistor at a temperature of 25 degrees (10,000 ohms) in (5):

$$Kelvin = \frac{Resistance}{10,000} \quad (5)$$

Finally, the Steinhart–Hart formula (temperature correction) is applied where B is the coefficient of the NTC thermistor (B = 3950 according to datasheet) and 25C is the nominal temperature at which the resistance of the thermistor is expressed. The result is obtained in *Kelvin* (6) and degrees *Celsius* (7):

$$Kelvin = log(Kelvin) * \frac{1}{B} + \frac{1}{25 + 273.15} \quad (6)$$

$$Celsius = \frac{1}{Kelvin} - 273.15 \quad (7)$$

If the Formula (7) is applied in the range around the nominal temperature (25 degrees Celsius), minimal deviations are obtained. However, when applied in the range of 0 degrees Celsius, deviations of up to 5 degrees Celsius may occur. Since the operating temperature of the device is 0 degrees and the allowable deviations, per project requirements, are two degrees Celsius, the error of the sensor operation is almost twice the expected nominal operation. The error results from the exponential curve of the thermistor, where it can be observed that there is an exponential change in resistance at much lower or higher temperatures. This can be confirmed by analysing the factory table of resistance change with the temperature data given in the sensor data sheet.

The measured sensor temperature according to real temperature for a previous device model is presented in Figure 8.

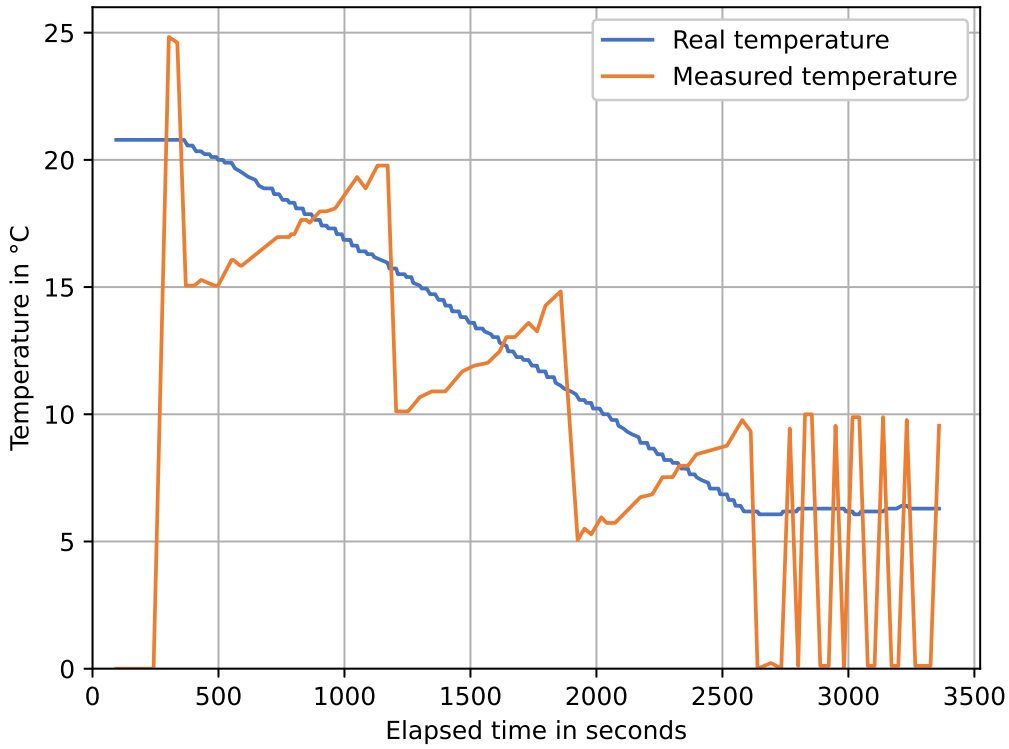

Figure 8. The measured temperature according to real temperature for the previous device model.

According to the graph shown, using the Steinhart–Hart equation near 0 degrees Celsius is not an option for accurate temperature telemetry. Since the factory resistance settings in the data sheet are in increments of 5 degrees Celsius, the interpolation of the values between each 5 degrees Celsius is performed. The code shown in Listing 1 first determines the membership of a set of two points: the upper value of the resistor and the lower value of the thermistor temperature resistance. An interpolation of the values between these two points is performed to determine the exact temperature. In Figure 9, it can be seen that after our corrections, the calculated temperatures follow relatively well the real temperature measured with the reference sensor. We calculated the mean absolute error (MAE) for each measured data point. The MAE for the graph represented by Figure 8 was calculated as 3.802, while the MAE for a graph represented by Figure 9 is 0.341, which corresponds to a 11.34-fold temperature precision measurement increase.

The most important group of sensors are flow sensors, which enable flow measurement on all copper tubing inside the python tube. With the help of these sensors, it is possible to measure the consumption of beverages and determine the amount of liquid that has flown through the device. The device is equipped with flow sensors shaped in the form of blades that rotate under water pressure and send a pulse to the microcontroller when a complete rotation is achieved. The microcontroller allows the support of interrupts on all 30 pins. Pins 5, 17, 16, 4, 2 and 15 were chosen as the digital input pins for pulse detection on the flow sensors since the device can support six simultaneous beverage lines. These pins were chosen because they have no other important role (I2C, SPI or Serial) on the microcontroller.

The previous version of the device used interrupt routines. Each pin triggered its interrupt routine upon the detection of a signal change (pulse) on the pin itself. The signal change was due to a pulse from the flow sensor. In the interrupt routine, the value of the variable was increased. In the new version of the code with the specified method, a large number of measurements were made that did not correspond to the actual flow during the measurements. A detailed experimental investigation of the signal and the microcontroller work traced the problem to the speed of the microcontroller itself. The previous version microcontroller operated at a speed of 8 MHz and detected the signal pulse much better. The new microcontroller detected many more signals in the same amount of time for the

same amount of fluid. At the operating frequency of 250 MHz, the microcontroller manages to register the transit noise in the pulse. An example of the noise visible on the oscilloscope is shown in Figure 10.

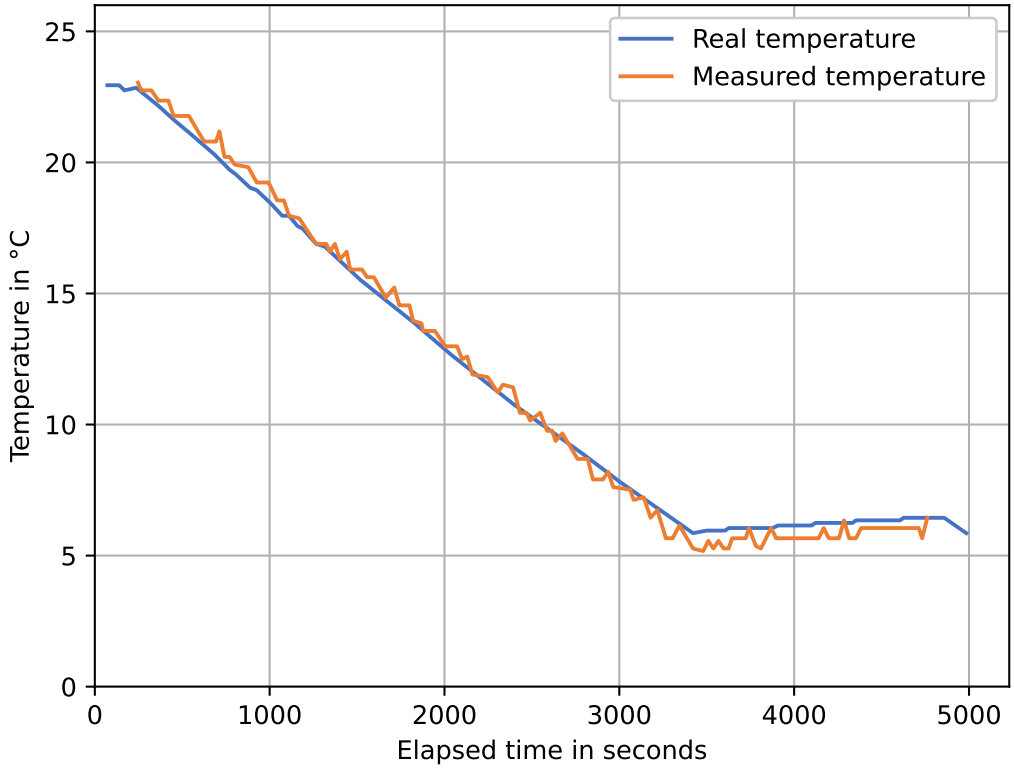

Figure 9. The measured temperature according to real temperature in the smart IoT device.

Listing 1. Program code snippet for temperature interpolation.

```
float x[] = {329.5, 247.7, 188.5, 144.1, 111.3, 86.43, 67.77,
    53.41, 42.47, 33.9, 27.28, 22.05, 17.96, 14.69, 12.09,
    10, 8.313, 6.94, 5.827, 4.911, 4.16, 3.536, 3.02, 2.588,
    2.228, 1.924, 1.668, 1.451, 1.266, 1.108, 0.9731, 0.8572,
    0.7576};
float y[] = {-50,-45,-40,-35,-30,-25,-20,-15,-10,-5,0,5,10,
    15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,
    105,110};

for(i = 0; i < 5; i++){
    measTempU[i] = (int)(( measTempRaw[i] * 330.0) / 4095.0);
    diff = (float)measTempU[i];
    readvalue = 10 * (diff/(330 - diff)); //max voltage 3.3V
    int k;
    for (k = 0; k < 33 - 1; k++)
    {
        if (readValue < x[k] && readValue > x[k + 1])
            break;
    }
    readValue = (y[k] + (y[k + 1] - y[k])/(x[k + 1] - x[k]) *
        (readValue - x[k])) * 100;
    measTemp[i] = (int)readValue;
}
```

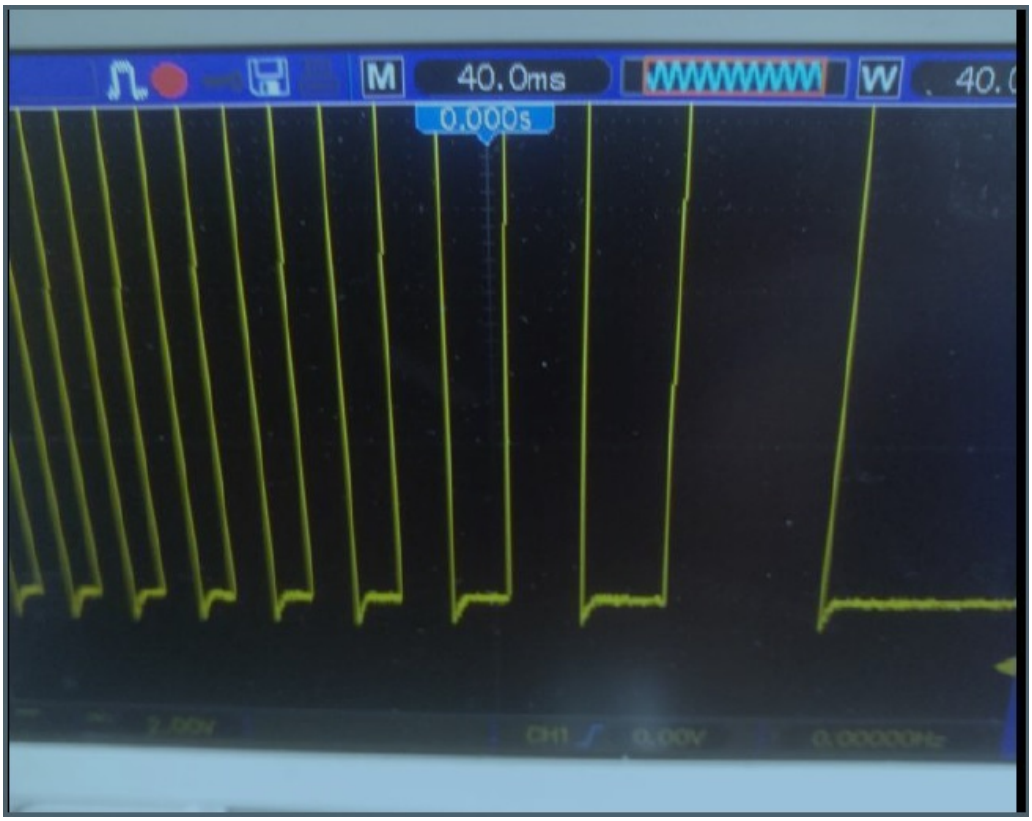

Figure 10. The noise on pulse image of the flow sensor.

Resetting the number of microseconds to zero for every pulse (subtracting the first value from all values) gives the timeline of detected pulses. Based on the information obtained, two solutions were proposed based on: begin itemize item double pulse isolation item pulse isolation according to time base end itemize.

The expected order of correct signal readout is the successive iteration of high value and low value. Double pulse isolation operates by discarding pulses that have reported a high value without going to a low value in the meantime. These are extremely fast pulses that occur in immediate succession. The isolation of double pulses has not proved satisfactory and cannot be improved.

Time-based isolation ignores detected high-value pulses for a specified period of time after the occurrence of a high-value pulse. All pulses detected immediately after the occurrence of the first pulse were discarded for the pause period. This approach improved the results somewhat, but the effectiveness depended on the time pause value. The time pause value of 15,000 µs approached the theoretical maximum. This time is defined by the number of pulses per litre of fluid during maximum turbine flow. At faster tapping, the results are satisfactory, but at slower tapping, the error becomes larger. Slower tapping is defined by 1 L metered in 60 s, while faster tapping is defined by 1 L metered in 10 s. This definition is based on experimental results with previous Oprema d.d. device models.

To solve the problem, a new algorithm is proposed based on the time prediction of the pulses and the previous interrupt logic on pins is abandoned. The proposed algorithm is shown in Listing 2.

Listing 2. Program code snippet for flow measurement.

```
//TIMER
timer = timerBegin(0, 80, true);
timerAttachInterrupt(timer, &onTimer, true);
timerAlarmWrite(timer, 4000, true); //timer every 4ms
timerAlarmEnable(timer);

void IRAM_ATTR onTimer() { //every 4ms
    portENTER_CRITICAL_ISR(&timerMux);
    pulse_number[0]=digitalRead(V1);
    pulse_number[1]=digitalRead(V2);
    pulse_number[2]=digitalRead(V3);
    pulse_number[3]=digitalRead(V4);
    pulse_number[4]=digitalRead(V5);
    pulse_number[5]=digitalRead(V6);
    for(int e=0;e<6;e++){
        last_last_pulse_number[e]=last_pulse_number[e];
        last_pulse_number[e]=pulse_number[e];
        if(pulse_number[e]==0 and last_pulse_number[e]==1){
            if (last_last_pulse_number[e]==1){
                flow[e]++;
            }
        }
    }
    portEXIT_CRITICAL_ISR(&timerMux);
}
```

The timer is set to 4 ms, which was calculated to be the shortest signal period on the fastest tap, according to the flow meter specifications from the datasheet [40]. Interrupt timer time is calculated upon maximum flow measurement based on device maximum python pipe capacity. This capacity determines the number of impulses per second using the datasheet data for the number of litres per second in maximum flow. When interrupts occur (every 4 ms), the flow line state is read and the previous two states are taken into account. A flow pulse is added only if the previous two states were in a high logical unit. In this way, errors caused by noise when a logic zero occurs are ignored. Satisfactory measurement accuracy was achieved using the above algorithm. The results are shown in Tables 1 and 2. The results presented are satisfactory over the entire operating range of the flow meter (slow and fast tapping). Positive deviation ranges from 2.12% to 4.5% for slow tapping and from 4.2% to 6.8% for fast tapping. The results shown are averages for multiple measurements taken on multiple flow meters. The minimum (Python #1) and maximum (Python #2) flow meter deviations are presented. The flow meter deviations are compared to calibrated flow meters (Multicon V7x [41]).

In order to store the instrument settings and recorded flow on all pipes, the previous model had a fast EEPROM on the main board. This EEPROM was added because the microcontroller internal EEPROM was too slow to allow memory operation during power failure. The new device model uses a microcontroller EEPROM because the writing speed was significantly improved.

Table 1. Number of impulses in slow tapping during 60 s period for 1 L.

Calibrated Flow Meter		Telemetry Results		Deviation	
Python #1	**Python #2**	**Python #1**	**Python #2**	**Python #1**	**Python #2**
380	373	389	390	9	17
375	374	384	391	9	17
377	373	386	390	9	17
378	375	386	391	8	16
377	377	385	393	8	16

Table 2. Number of impulses in fast tapping during 10 s period for 1 L.

Calibrated Flow Meter		Telemetry Results		Deviation	
Python #1	**Python #2**	**Python #1**	**Python #2**	**Python #1**	**Python #2**
379	382	396	409	17	27
380	375	396	399	16	24
383	383	399	407	16	24
382	380	398	402	16	22
384	383	401	406	17	23

The previous device model had sensors to establish the energy balance of the device. A voltage transformer was used to measure the main voltage and frequency. Current sensors were used to measure fan, compressor and pump (mixer) current. The voltage transformer and current sensors were not able to measure voltage and current accurately. In order to improve the motherboard reliability, the values for the maximum current range were predefined, which resulted in a relatively large error in the current measurements. In addition, the voltage transformer used was not sufficiently calibrated, so detailed voltage information was missing. Measurement accuracy was not a problem as it was not used for precise telemetry. Current sensors allowed the implementation of compressor, fan, and pump (mixer) monitoring that provided information on the current flow present. Alarming events were shown on the unit display.

The new motherboard has a connection capability for current meters between the main power supply and the device. The current meter provides voltage, current, frequency, and power factor information over the serial line. This meter provides accurate measurements that can be used as telemetry for the unit. For each unit, compressor, fan, and pump (mixer) startup is performed to calculate the average current flow in various combinations. This information is later used to detect compressor, fan or pump (mixer) malfunctions. These malfunctions are introduced to the system as alerts through cloud connectivity. Based on the alert, it can additionally shut down the unit before the malfunction causes major damage. This adjustment simplifies the motherboard, provides an accurate power balance, and maintains fault detection and alerting.

An important concept related to ecology and health is the sanitation of the device. The decision about the required sanitation is made based on previous sanitation schedule, the length of the python, the wrong temperatures in the system (leading to the possible growth of bacteria), the flow rate and the summary flow statistics. Based on the information obtained, the mandatory sanitation is ordered.

The flow sensors monitor the sanitation process and ensure that it is performing correctly by measuring the flow rate through the device itself during sanitation. Since the sanitation is performed with aggressive liquids, it is necessary to cleanse the device tubes by sending a significant volume of liquid before the device is restored to general use. The flow measurements obtained during sanitation are not included in the total flow statistics.

4.3. Security Protocol

The security concept is presented in Figure 11. The smart beverage cooling device uses a public and secret key for communication. Both keys are generated separately for each device during manufacturing and inserted into the device EEPROM on the production line. On the last step of the production line, the device was registered with the cloud. During that process, a serial number is generated and placed together with both keys in the cloud database.

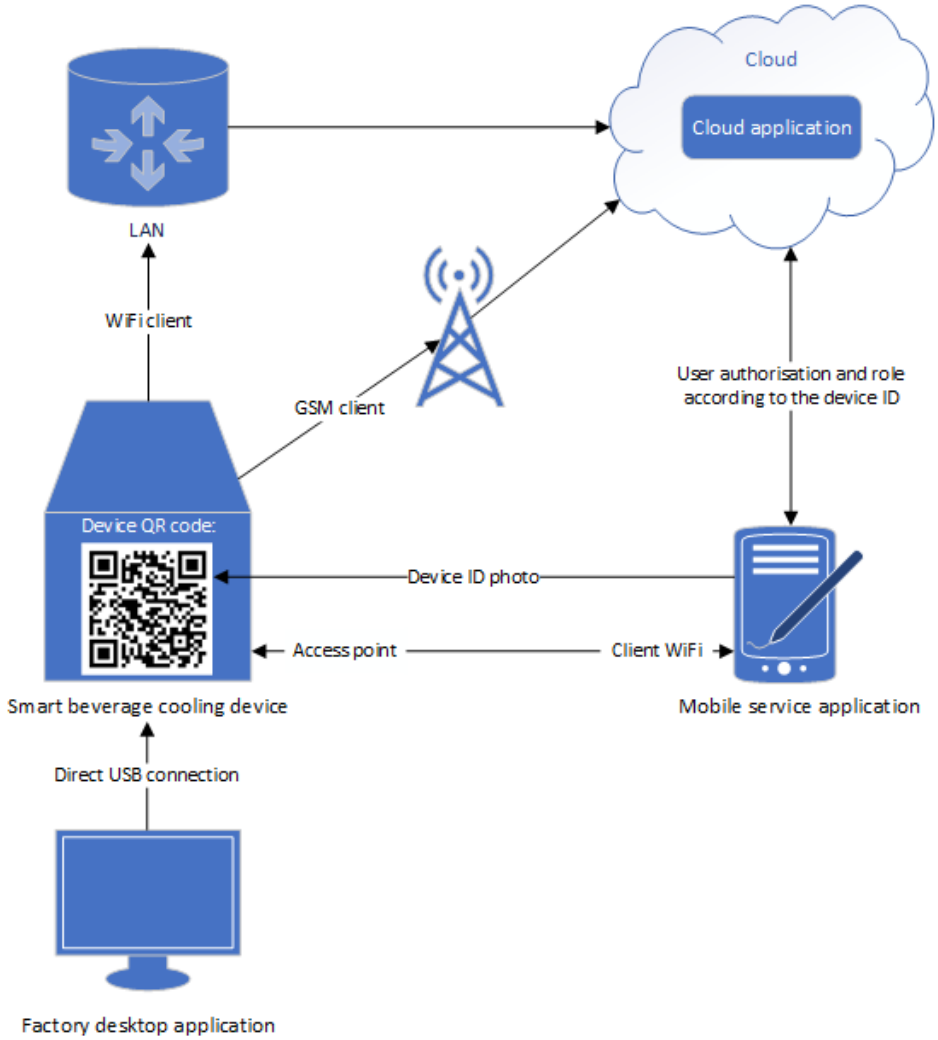

Figure 11. The security concept of the device.

The public key can be easily retrieved as it is printed on each device (QR code), while the secret key is only stored in the cloud and on the erasable programmable read-only memory (EPROM) of the device itself. The mobile application connects to the smart beverage cooling device using wireless technology, with the smart beverage cooling device acting as the access point. The access point is protected by the Wireless Fidelity Protected Access II (WPA2) encryption standard. Before connecting the mobile application to the device, it is necessary to read the public key of the device in the form of Quick Response (QR) code printed on the smart beverage cooling device through the mobile device application to retrieve the wireless password and secret key of the smart beverage cooling device from the cloud. Once connected, the smart beverage cooling device communicates with the mobile application through the Representational State Transfer (RESTful) service. All the communication takes place over a JavaScript Object Notation (JSON) data structure. The communication protocol used is Hypertext Transfer Protocol Secure (HTTPS). To gain further security, each JSON message that is sent to the device must be signed with a secret key that is not permanently stored anywhere on the mobile device for security reasons.

For a mobile application to access RESTful cloud services, a service technician must have created a cloud account. This account can only be assigned by a person with elevated cloud privileges (e.g., cloud administrator). This eliminates the problem of unauthorised login of service technicians to the cloud. All communication between the cloud and the mobile application is encrypted using the HTTPS protocol. After the person logs in to the application with a username and password, all further communication is signed with

a token that is issued by the cloud after each login. This token expires after predefined period of time and the service technician must log in to the application again to confirm their identity.

When the smart beverage cooling device communicates with the cloud, the public and private keys are used again. Each message must contain the device identifier which identifies the device on the cloud, and the message is signed with a private key to ensure authenticity. Since only the smart beverage cooling device and the cloud possess the private key, it is simply validated on the cloud. All data are sent to the RESTful service encrypted using the HTTPS protocol, which provides additional data protection. The communication between the cloud and the smart beverage cooling device is always one-way, in a sense that the smart beverage cooling device only sends data to the cloud. This behaviour avoids the possibility of unauthorised device adjustment over the Internet.

4.4. Factory Application

In the previous model of the device, the microcontroller was programmed during the production of the motherboard. The final stage of production consists of quality control, which verifies the correct operation of the sensors. When setting up the device, it was necessary to calibrate the flow sensor and enter the calibration coefficient along with the temperature and temperature hysteresis settings using three buttons and a small two-line display.

The new model of the device does not involve any significant changes in the production process. A new factory application was developed for the new device model, replacing the microcontroller programming tool previously used with the same code. When programming the microcontroller, two keys are generated and a specific program is entered for each device. In the future, if needed, a specific code can be entered for a device model with a different hardware. When the quality is checked in the final stage of production, the device is started for the first time and reported through the service cloud applications. At this stage, all sensors are verified. As with the previous model, the flow sensor is calibrated via a mobile field application when the device is set up. When entered, the device settings are also entered into the service mobile application. Thus, the service mobile application replaces the three buttons and two-line display in communication with the new model. The factory application uses command line commands to program the microcontroller via a standard USB port.

4.5. Cloud

Implementing the application and business logic in the cloud enables high availability, scalability and easily supports the increase in the number of supported devices. The cloud application provides endpoints for data collection as well as device management capabilities, such as managing device location, generating usage reports, and providing real-time alerts in case of anomalous behaviour. The application picture is shown in Figure 12.

In particular, the sanitation expectancy period is calculated and enforced through alerting system. Therefore, it consists of the user interface part for management and the REST API segment for data ingestion and service mobile application authentication operation. The user interface operations are authorised by role-based security and the permissions are granulated based on the organisation structure. Multiple devices can be assigned to a single organisation, and each organisation can have multiple administrators. User and device data are stored in the relational database due to referential integrity requirements. The service mobile application intended for service operations uses the protected endpoint and requires standard JWT token authentication. The token is obtained on a separate endpoint using the assigned credentials for the waiting user via a common username and password authentication scheme. Based on the permissions and organisational affiliation, each service user is separately authenticated for each device using cloud authentication and device-based authentication. Data entry does not pose a high security risk, so no dedicated

authentication is performed other than signing the payload data with the device's private key and verifying it in the cloud with the device's specified public key. Due to the large amount of ingested data, a NoSQL database is used for the efficient storage of sensor measurement data. The data aggregation component processes the data in near real time (NRT) to provide an efficient reporting and alerting system.

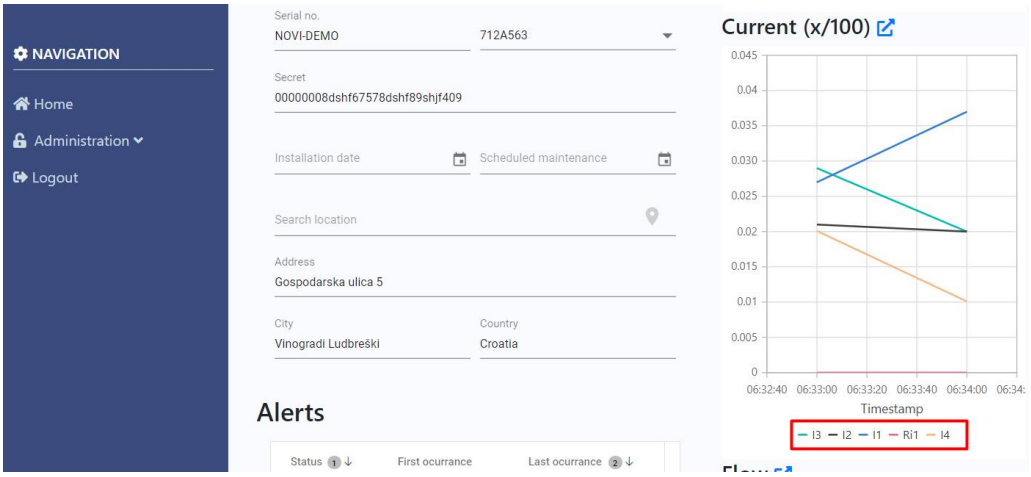

Figure 12. Cloud application interface.

The cloud system allows three levels of device monitoring:

- device user (restaurant, hotel, private person);
- device owner (brewery, juice factory, hotel chain);
- device manufacturer Oprema d.d.

Each level has its own monitoring needs and accordingly monitors the device from its own point of view. The equipment user is interested in the energy balance and the equipment flows. The sanitation and fault alerts determine the maintenance schedule. The equipment owner is interested in the equipment flow because most equipment is rented to introduce new beverages on tap in restaurants and stores. Finally, the device manufacturer has information about the reliability of the equipment, which allows it to introduce new hardware and software innovations in equipment models or to create an equipment upgrade specification in the service cycle. This is one of the primary goals of Industry 4.0.

4.6. Service Mobile Application

A service mobile application was developed to support the service technician in the field. The application allows the technician to read raw values from devices that have not yet been processed by the cloud, which can help the service technician troubleshoot the smart device. For example, the liquid flow sensor is stuck. After the service technician drains a certain amount of fluid through the python tube, a fault can be easily detected. A service mobile application is shown in Figure 13.

In order for the service mobile app to connect to the device, a cloud service technician account is required to retrieve the secret key for the device so that no one can access it without authorisation. An additional security feature is the mobile application connection window of only 5 min after the device was powered on. This creates a new application layer that is the only access point for changing device parameters and prevents unauthorised access to the device through two-tier security.

Using the mobile application, it is possible to set all the internal parameters of the device as well as update the firmware running on the device. After launching the mobile application, it is necessary to read the public key of the smart device, which is located on the device in the form of a QR code, after which the application contacts the cloud to see if the service technician logged into the application has the authorisation to service

the device. When the mobile application is connected and the smart device is in service mode, any action from the mobile device to the smart device must be signed with the public and private keys. Parameters that can be loaded and configured on the device are wireless connection settings, GSM connection settings, firmware update, manual device control, sensor constants, and cloud connection settings. Parameters that cannot be set, but only loaded, are real-time sensor data. All communication between the service mobile application and the cloud is done over an HTTPS connection via RESTful web services.

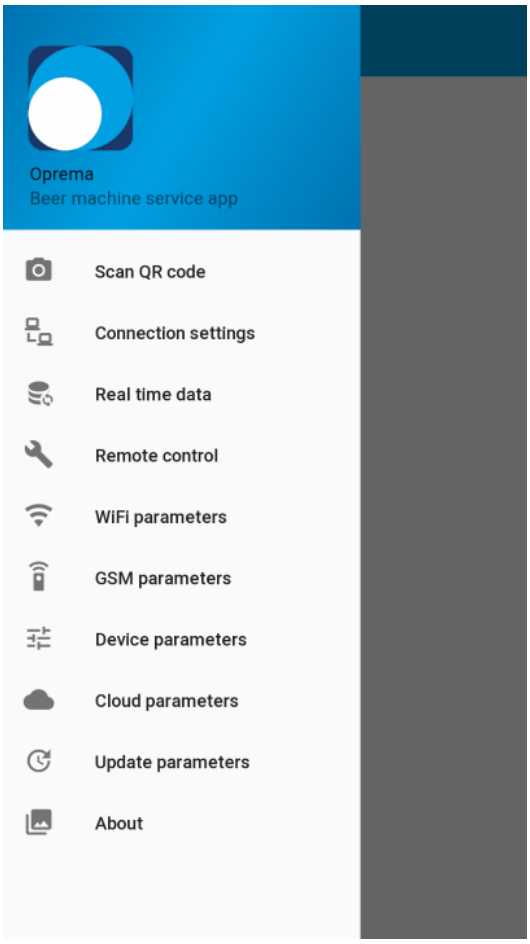

Figure 13. Service mobile application interface.

The service mobile application is not only used for service purposes but also for the regular maintenance of the device. The regular maintenance of the device mainly includes the sanitation of the device and calibration of the flow sensor of the device. During the periodic maintenance, which is required by the authorities, the service technician enters the date of maintenance and the actions performed through the application, which are sent to the cloud. In this way, the condition of the device is monitored and the organisation of the regular maintenance of the device is simplified. The regular maintenance of the device must take into account the flow that has occurred since the last maintenance, in addition to the time interval.

5. Conclusions

In this paper, we presented the final results of the project European Regional Development Fund-Operational Programme: Competitiveness and Cohesion; Project name: Development of the new generation Eco Smart product of the company Oprema d.d. in the computing field. The aim of the project was to introduce smart support for the device following the resolution of Industry 4.0. According to the request of Oprema d.d., the proposed extension should be able to update a previous, non-smart version of the device in

the simplest and most economical way. According to the requirement, the existing device sensors and other electronic parts should be retained as much as possible.

As security becomes an important issue in Industry 4.0, a secure one-way communication to the cloud is proposed. Moreover, a mobile service application intended for personal and custom device configuration is available shortly after the device is registered on its private wireless network. The device configuration is not available via local LAN or GSM modem connection.

To achieve the proposed security changes, a new software architecture is proposed based on the standard software architecture of IoT devices. The new software architecture introduces an additional layer to separate the device operation from the IoT connectivity and configuration of the device. The implementation of the software architecture is analysed and necessary security protocols and adjustments are described in detail. The mobile service application is also described along with factory application.

In this paper, the concept of a dual-thread microcontroller is used to separate the device operation from IoT communication. In the case of exploiting a security vulnerability in the IoT communication thread (service and transport layer) and blocking it, a device would be fully functional without telemetry in the second thread. Additionally, the paper discusses the temperature and flow sensors used in the previous version of the device. During the adaptation to a new controller as part of the new device, several problems emerged. The solution was presented and discussed in the previous section. The obtained results confirm the proper operation of the temperature sensor with an 11.34-fold increase in precision compared to the original algorithm. We introduced an efficient mechanism to reduce the noise in the readout of the flow sensor to enable reliable measurements. The results show that the number of pulses of the flow sensor for one litre at slow and fast tapping is consistent with that of the calibrated measurement device. The range of positive deviation in the pulse count is between 2.4% and 6.8%.

Future research approaches will include optimising the device processes when additional long-term usage data will be available. Continuous monitoring of the devices in line with Industry 4.0 offers the best opportunities for further improvements.

Author Contributions: Conceptualisation, I.D. and D.C.; methodology, I.D. and D.C.; software, I.C. T.K. and D.C.; validation, I.D.; formal analysis, I.D. and D.C.; investigation, I.D. and D.C.; resources, I.D., D.C., T.K. and I.C.; data curation, I.C. and T.K.; writing—original draft preparation, I.D., D.C. and T.K.; writing—review and editing, T.K. and I.C.; visualisation, T.K. and I.D.; supervision, I.D. and D.C.; project administration, D.C.; funding acquisition, I.D., D.C., T.K. and I.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by European Union Project-European Regional Development Fund-Operational Programme: Competitiveness and Cohesion-Project name: Development of the new generation Eco Smart product of the company Oprema d.d., Project number: KK.01.2.1.0104. The APC was funded by Zagreb University of Applied Sciences.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This paper is a part of the project Development of the new generation Eco Smart product of the company Oprema d.d. The authors would like to thank all project participants for their cooperation and support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cha, S.; Hsu, T.; Xiang, Y.; Yeh, K. Privacy Enhancing Technologies in the Internet of Things: Perspectives and Challenges. *IEEE Internet Things J.* **2019**, *6*, 2159–2187. [CrossRef]
2. Cirillo, F.; Gómez, D.; Diez, L.; Maestro, I.E.; Gilbert, T.B.J.; Akhavan, R. Smart City IoT Services Creation Through Large-Scale Collaboration. *IEEE Internet Things J.* **2020**, *7*, 5267–5275. [CrossRef]

3. Ungurean, I.; Brezulianu, A. An Internet of Things Framework for Remote Monitoring of the HealthCare Parameters. *Adv. Electr. Comput. Eng.* **2017**, *17*, 11–16. [CrossRef]
4. Ayoub, W.; Samhat, A.E.; Mroue, M.; Joumaa, H.; Nouvel, F.; Prévotet, J.C. Technology Selection for IoT-Based Smart Transportation Systems. In *Vehicular Ad-Hoc Networks for Smart Cities. Advances in Intelligent Systems and Computing*; Laouiti, A., Qayyum, A., Mohamad Saad, M., Eds.; Springer: Singapore, 2020; Volume 1144. [CrossRef]
5. Yaqoob, I.; Hashem, I.A.T.; Ahmed, A.; Kazmi, S.M.A.; Hong, C.S. Internet of things forensics: Recent advances, taxonomy, requirements, and open challenges. *Future Gener. Comput. Syst.* **2019**, *92*, 265–275. [CrossRef]
6. Kraijak, S.; Tuwanut, P. A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends. In Proceedings of the 2015 IEEE 16th International Conference on Communication Technology (ICCT), Hangzhou, China, 18–20 October 2015; pp. 26–31. [CrossRef]
7. Sisinni, E.; Saifullah, A.; Han, S.; Jennehag, U.; Gidlund, M. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4724–4734. [CrossRef]
8. Ungurean, I.; Gaitan, N.C. A Software Architecture for the Industrial Internet of Things—A Conceptual Model. *Sensors* **2020**, *20*, 5603. [CrossRef]
9. Petrillo, A.; Picariello, A.; Santini, S.; Scarciello, B.; Sperlí, G. Model-based vehicular prognostics framework using Big Data architecture. *Comput. Ind.* **2020**, *115*, 103177. [CrossRef]
10. De santo, A.; Galli, A.; Gravina, M.; Moscato, V.; Sperli, G. Deep Learning for HDD health assessment: An application based on LSTM. *IEEE Trans. Comput.* **2020**. [CrossRef]
11. Kaur, K.; Garg, S.; Aujla, G.S.; Kumar, N.; Rodrigues, J.J.P.C.; Guizani, M. Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay. *IEEE Commun. Mag.* **2018**, *56*, 44–51. [CrossRef]
12. Bader, S.R.; Maleshkova, M.; Lohmann, S. Structuring Reference Architectures for the Industrial Internet of Things. *Future Internet* **2019**, *11*, 151. [CrossRef]
13. Oztemel, E.; Gursev, S. Literature review of Industry 4.0 and related technologies. *J. Intell. Manuf.* **2020**, *31*, 127–182. [CrossRef]
14. Marina, C.; Ivica, V.; Nikola, B. From concept to the introduction of industry 4.0. *Int. J. Ind. Eng. Manag.* **2017**, *8*, 21–30.
15. Lombardi, M.; Pascale, F.; Santaniello, D. Internet of Things: A General Overview between Architectures, Protocols and Applications. *Information* **2021**, *12*, 87. [CrossRef]
16. Li, S.; Xu, L.D.; Zhao, S. The internet of things: A survey. *Inf. Syst. Front.* **2015**, *17*, 243–259. [CrossRef]
17. Cafuta, D.; Dodig, I.; Cesar, I.; Kramberger, T. Developing a Modern Greenhouse Scientific Research Facility—A Case Study. *Sensors* **2021**, *21*, 2575. [CrossRef]
18. Kalsoom, T.; Ramzan, N.; Ahmed, S.; Ur-Rehman, M. Advances in Sensor Technologies in the Era of Smart Factory and Industry 4.0. *Sensors* **2020**, *20*, 6783. [CrossRef]
19. Režek Jambrak, A.; Nutrizio, M.; Djekić, I.; Pleslić, S.; Chemat, F. Internet of Nonthermal Food Processing Technologies (IoNTP): Food Industry 4.0 and Sustainability. *Appl. Sci.* **2021**, *11*, 686. [CrossRef]
20. Barriga, J.J.; Sulca, J.; León, J.L.; Ulloa, A.; Portero, D.; Andrade, R.; Yoo, S.G. Smart Parking: A Literature Review from the Technological Perspective. *Appl. Sci.* **2019**, *9*, 4569. [CrossRef]
21. Taştan, M.; Gökozan, H. Real-Time Monitoring of Indoor Air Quality with Internet of Things-Based E-Nose. *Appl. Sci.* **2019**, *9*, 3435. [CrossRef]
22. Jo, J.; Jo, B.; Kim, J.; Kim, S.; Han, W. Development of an IoT-Based Indoor Air Quality Monitoring Platform. *J. Sens.* **2020**, *2020*, 8749764. [CrossRef]
23. Sunny, A.I.; Zhao, A.; Li, L.; Kanteh Sakiliba, S. Low-Cost IoT-Based Sensor System: A Case Study on Harsh Environmental Monitoring. *Sensors* **2021**, *21*, 214. [CrossRef]
24. Maroto-Molina, F.; Navarro-García, J.; Príncipe-Aguirre, K.; Gómez-Maqueda, I.; Guerrero-Ginel, J.E.; Garrido-Varo, A.; Pérez-Marín, D.C. A Low-Cost IoT-Based System to Monitor the Location of a Whole Herd. *Sensors* **2019**, *19*, 2298. [CrossRef] [PubMed]
25. D'Aloia, M.; Longo, A.; Guadagno, G.; Pulpito, M.; Fornarelli, P.; Laera, P.N.; Rizzi, M. Low Cost IoT Sensor System for Real-time Remote Monitoring. In Proceedings of the 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Roma, Italy, 3–5 June 2020; pp. 576–580. [CrossRef]
26. Saoutieff, E.; Polichetti, T.; Jouanet, L.; Faucon, A.; Vidal, A.; Pereira, A.; Boisseau, S.; Ernst, T.; Miglietta, M.L.; Alfano, B.; et al. A Wearable Low-Power Sensing Platform for Environmental and Health Monitoring: The Convergence Project. *Sensors* **2021**, *21*, 1802. [CrossRef] [PubMed]
27. Shahidul Islam, M.; Islam, M.T.; Almutairi, A.F.; Beng, G.K.; Misran, N.; Amin, N. Monitoring of the Human Body Signal through the Internet of Things (IoT) Based LoRa Wireless Network System. *Appl. Sci.* **2019**, *9*, 1884. [CrossRef]
28. Sung, W.-T.; Hsiao, S.-J.; Shih, J.-A. Construction of Indoor Thermal Comfort Environmental Monitoring System Based on the IoT Architecture. *J. Sens.* **2019**, *2019*, 2639787. [CrossRef]
29. Chu, H.-M.; Lee, C.-T.; Chen, L.-B.; Lee, Y.-Y. An Expandable Modular Internet of Things (IoT)-Based Temperature Control Power Extender. *Electronics* **2021**, *10*, 565. [CrossRef]
30. Munir, M.S.; Bajwa, I.S.; Ashraf, A.; Anwar, W.; Rashid, R. Intelligent and Smart Irrigation System Using Edge Computing and IoT. *Complexity* **2021**, *2021*, 6691571. [CrossRef]

31. Ungurean, I.; Gaitan, N.C.; Gaitan, V.G. A Middleware Based Architecture for the Industrial Internet of Things. *KSII Trans. Internet Inf. Syst.* **2016**, *10*, 2874–2891. [CrossRef]
32. Yelamarthi, K.; Aman, M.S.; Abdelgawad, A. An Application-Driven Modular IoT Architecture. *Wirel. Commun. Mob. Comput.* **2017**, *2017*, 1350929. [CrossRef]
33. Sattar, H.; Bajwa, I.S.; Ul-Amin, R.; Mahmood, A.; Anwar, W.; Kasi, B.; Kazmi, R.; Farooq, U. An Intelligent and Smart Environment Monitoring System for Healthcare. *Appl. Sci.* **2019**, *9*, 4172. [CrossRef]
34. Malhotra, P.; Singh, Y.; Anand, P.; Bangotra, D.K.; Singh, P.K.; Hong, W.-C. Internet of Things: Evolution, Concerns and Security Challenges. *Sensors* **2021**, *21*, 1809. [CrossRef]
35. Xu, Z.; Liu, W.; Huang, J.; Yang, C.; Lu, J.; Tan, H. Artificial Intelligence for Securing IoT Services in Edge Computing: A Survey. *Secur. Commun. Netw.* **2020**, *2020*, 8872586. [CrossRef]
36. Ahanger, T.A.; Tariq, U.; Ibrahim, A.; Ullah, I.; Bouteraa, Y. IoT-Inspired Framework of Intruder Detection for Smart Home Security Systems. *Electronics* **2020**, *9*, 1361. [CrossRef]
37. Development of the New Generation Eco Smart Product of the Company Oprema d.d. Available online: https://www.oprema.com/?A=ABO&S=ERE (accessed on 12 May 2021).
38. Maxim Integrated: DS18B20. Available online: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf (accessed on 10 May 2021).
39. Eliwell Product Documentation. Available online: https://www.eliwell.com/en/Partnumber/SN8DED11502C0.html (accessed on 10 May 2021).
40. Digmesa Flow Meters. Available online: https://www.digmesa.com/wp-content/uploads/938-3570_01_GB.pdf (accessed on 10 May 2021).
41. Oprema d.d.: MULTICON. Available online: https://support.oprema.com/docs/manuals/Oprema_MULTICON-hr-en-de.pdf (accessed on 10 May 2021).