2022 EC³
2022 European Conference on Computing in Construction
Ixia, Rhodes, Greece
July 24-26, 2022

EVALUATION OF LOW-COST MICROCONTROLLER-BASED ELECTRONIC SYSTEMS FOR SIMPLE SENSOR NODE APPLICATIONS

Benjamin Burse[1], Kristina Doycheva[2], Andreas Aicher[3] and Christian Walther[4]
[1]Bauhaus-University Weimar, Chair for Computing in Civil Engineering, Germany
[2]Fraunhofer IVI, Fraunhofer Institute for Transportation and Infrastructure Systems
[3]Bauhaus-University Weimar, Chair for Urban Water Management and Sanitation, Germany
[4]Bauhaus-University Weimar, Institute of Structural Mechanics, Germany

Abstract

Single-board computers like Raspberry Pi, have enabled a wide variety of monitoring applications. Microcontroller development boards offer similar functionality, but at a lower cost. Still, they are not widely used, because they are supposed to be slow. This paper aims to evaluate the abilities of modern microcontrollers and compares them to single-board computers. For this purpose, a market research and experiments are conducted. The results show that promising low-cost microcontrollers exist, which could be applied to reduce the cost per node or increase the number of nodes used simultaneously in one monitoring system.

Introduction

In our modern world, many systems observe their surroundings and may also act on them, mostly unnoticed by us or taken for granted. These processes are referred to as monitoring and control, and they are becoming increasingly indispensable. The size of systems making use of monitoring and control ranges from small, everyday items to entire factories or even cities (Su et al. 2014, Westkämper et al. 2013, Eremia et al. 2017, Zand et al. 2012). Monitoring or monitoring and control, respectively, have made some outstanding achievements of the 20th century possible. One example is magnetic resonance imaging, presented in 1973. By using strong electromagnetic fields, this procedure can create cut-through images of animal or human bodies, thereby enabling doctors to have a view into the body without the need to cut it open. This capability supports research in many fields of modern medicine, e.g., detecting and treating cancer or understanding processes in the brain (Lauterbur 1973, Morris 2002, Despotović et al. 2015).

In the classical engineering sciences of civil and mechanical engineering, monitoring is used to understand materials and their properties and thus to improve them. A reliable method to evaluate component strength is destructive testing. In such experiments, the components that have to be assessed can be tested by compressive force through a hydraulic press until failure. However, destructive testing is not always an option, as sometimes parts of the experiment can be too large, expensive, or long to produce. In such cases, monitoring is a good, non-destructive alternative. The procedure of monitoring specific criteria over the components lifetime is referred to as structural health monitoring (SHM) (Malek & Kaouther 2014, Sheng et al. 2011).

With monitoring, whether on a small or large scale, data is collected. This data can either be used to control a known process or explore an unknown one, resulting in countless application areas. Different quantities can either be perceived by humans (e.g. temperature, sound, and a partial spectrum of light) or require special sensors for their detection (magnetism or radiation) and are measured with varying sensor hardware.

A hardware review conducted by Healy et al. (2008) lists more than 40 different sensor nodes produced since 1998 together with their hardware specifications. In cases where a single measuring device is not sufficient for a task or multiple devices are required to handle a number of tasks, wired networks need to be formed. Nowadays, these networks shift towards wireless technologies, as they enable more opportunities concerning their installation. In recent years, wireless sensor networks allowed for utilization in new SHM application domains. Wireless monitoring systems are cheaper, easier to install and maintain than conventional wired systems, and, depending on the application, transmission times are negligible.

Many wireless sensor nodes are based on single-board computers like the Raspberry Pi (Crookes et al. 2021). The term single-board computer is used within this work in contrast to standard desktop computers on which each component like CPU, memory or peripheral devices such as graphic card, sound card, etc. are separate components that are plugged into a linking circuit board most often referred to as motherboard (e.g. the IBM-like PC). Single-board computers are available in different sizes and appearance. Many, but not all, are also developer boards offering access to a wide variety of input and output interfaces (Ray & Al Dhaheri 2017, Kanagachidambaresan 2021).

The work presented in this paper should assess if microcontroller-based sensor node designs could complement or even replace existing none-microcontroller-based ones and thus increase cost-efficiency. A microcontroller is a standalone unit, which comprises of a microprocessor, sometimes also referred to as Central Processing Unit

(CPU), memory and programmable input/output peripherals. All components are wrapped within a single chip.
The paper focuses on evaluating criteria relevant for project hardware selection. To accomplish this aim, a wireless sensor node system based on microcontrollers was designed and tests were carried out. Relevant criteria, such as power consumption, computational power, and connectivity are brought to the tests and results are collected, analyzed, and interpreted in the context of similar solutions based on single-board computers. Additionally, non-measurable quantities like overall system complexity and accessibility for non-computer-science or electronic-engineering users are considered.

Design of a wireless sensor node system based on microcontrollers

A wireless sensor node system consists of three major components: a sensor node consisting of a microcontroller and senors (S1, S2, etc.) attached to it through interfaces (iface), a gateway, and a server. Figure 1 presents a high-level overview of such a system. The gateway and the server are intentionally kept abstract. Furthermore, the power supply of each component is considered to be a separate building block, as it ideally could be chosen from a variety of options like batteries, solar modules and generators. It is also possible that the gateway and the server get combined within a single device. In this case, there is no Wide Area Network (WAN) connection involved to transport the data between the gateway and the server.

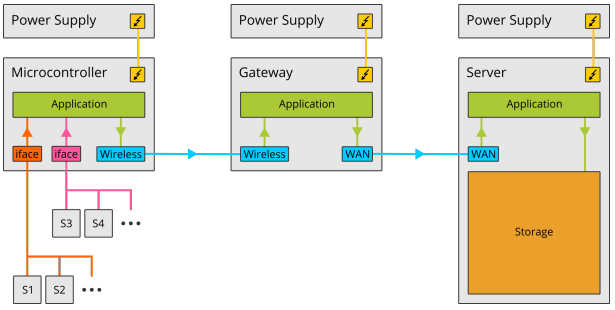

Figure 1: High-level overview and dataflow of the system. S1 - S4 denote sensors and Iface denotes an interface.

Based on this abstract model of a wireless sensor node system, we propose a specific implementation. To transmit data between the sensor node and the gateway, the wireless transmission technology wireless fidelity (WiFi) was chosen over other options like Bluetooth, as it offers a high number of possible devices and a good transmission quality over long distances. This decision directly dictates the appearance of the gateway. It would have been possible to use a Raspberry Pi as a gateway, but this would have involved numerous setups, possibly leading to an insecure device. As there was no intention to preprocess the data received from the sensor nodes, a regular WiFi router was chosen to function as a gateway. The ESP32 (Espressif Systems 2021) and the ESP8266 (Espressif Systems 2020) were selected as the microcontroller unit respectively. They offer decent performance, a wide range of analog and digital interface connections, e.g. Inter-Integrated Circuit bus I^2C (NXP Semiconductors 2021), low power consumption, and good documentation at low procurement costs. The sensor nodes were equipped to measure different physical quantities by attaching the following sensors to the I^2C interface: INA219 (current, voltage), BME280 (temperature, humidity, air pressure), and BH1750 (illuminance) (Texas Instruments 2015, Bosch Sensortec 2015, ROHM Semiconductor 2011). Additionally, the analog temperature sensor DHT11 (Sunrom Technologies 2012) was chosen to evaluate an interface without I^2C. For the server, a Raspberry Pi 3 was selected as it can be operated on a 24/7 basis with low power consumption. Furthermore, it offers enough non-volatile memory in the form of a secured digital (SD) card for all sorts of projects. There were many different options available to save the data on the SD card. These ranged from plain text, over JavaScript Object Notation (JSON) and Extensible Markup Language (XML), to databases like MariaDB or MongoDB. In the end, JSON was chosen, as it is human-readable, can be automatically parsed, is shorter to read and write compared to XML, and is easy to transfer between systems. Furthermore, JSON was also used for the data exchange between the microcontroller and the server. In terms of available programming languages, the microcontrollers and the server were quite different. The microcontrollers offer only C++ and Python as programming languages, whereas the server allows to choose from any available on Linux. Finally, for the power supply, a combination of on-grid and off-grid powering was chosen. The microcontrollers were powered by rechargeable batteries, whereas regular wall plugs powered the router and the Raspberry Pi. However, it would also be possible to operate the router and the Raspberry Pi with correspondingly large batteries. In Figure 2 the schematic of the proposed system is presented.

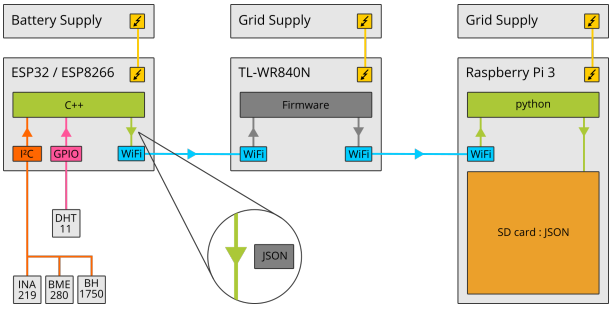

Figure 2: Detailed view of the proposed system.

Hardware benchmark tests

To evaluate the characteristics of microcontrollers and compare them to single-board computers, a hardware benchmark was performed. All experiments considered for the benchmark were carried out indoor in a regular middle European environment at approximately 22°C. Beside the previously mentioned ESP32 and the ESP8266, Raspberry Pico (Raspberry Pi Trading Ltd 2021) also rep-

resented the microcontrollers, while some already existing models of the Raspberry Pi family constituted the group of the single-board computers, namely the Raspberry Pi 2B, Raspberry Pi 3B, Raspberry Pi 3B+, Raspberry Pi 4B and the Raspberry Pi Zero W. These devices exemplarily represent the different performance classes available on the market. The Raspberry Pi 2B was added to the selection to examine if devices without a built-in WiFi module are still relevant by the usage of an additional USB WiFi dongle. The microcontrollers were run at different clock frequencies, because it is expected that they will consume less power while operating at low than at high frequencies.

Power consumption

An experiment was carried out to record the power consumption of the selected microcontrollers and single-board computers. Both the microcontrollers and the single-board computers were supplied with a voltage of 5V via USB. The flowing current in milliampere was measured between the positive output of the power supply and the positive input of the device. For this purpose, a UNI-T UT139B multimeter set to max-setting, which only displays the highest seen value, was used.

Since neither microcontrollers nor single-board computers are continuously operated under full load, the measurements for all devices were carried out in different scenarios: idle, high load, network transmission, if available without a keyboard, without a connected screen, and without a connected WiFi dongle. Some conditions were also measured in combination.

Idle For the idle state, two approaches were used. On one hand, measurements of the Raspberry Pi devices were conducted after the system had fully loaded. On the other hand, measurements of the microcontrollers were performed by executing a program consisting of only a single sleep instruction.

High load For the high load scenario, a self-built prime number search was used. This program creates different amounts of CPU stress based on the upper search limit and the number of threads. Both the single-board computers and the microcontrollers used an upper limit of one million. For the single-board computers, the maximum number of available threads was used. The microcontrollers used a slightly modified version without threading, as this functionality is not supported in the same way as in regular C++.

Network Different approaches on the single-board computers and the microcontrollers were used for the networking scenario. The single-board computers used the Linux tool iperf3 (networking speed measurement) to create a high load on the networking device. On the microcontrollers, a self-built program was used, which transmitted random numbers as quickly as possible.

Keyboard / screen / WiFi dongle All single-board computer devices were used with a keyboard and a screen attached. As the Raspberry Pi 2 has no built-in WiFi module, an additional WiFi dongle was used. By systematically unplugging these connections, their influence was measured. All of these measurements were performed in idle state.

Deep sleep One crucial feature of the microcontrollers, which is not available on the Raspberry Pi, is *deep sleep*. A Raspberry Pi has different power states as well, but is not able to wake up or restart by itself. The microcontrollers either have a timer-based wake-up, an event-based wake-up (e.g., voltage high/low on a specific pin), or both. As all manufacturers claim that their devices require a low amount of power in the deep sleep state, a test was carried out to prove this claim. Based on the microcontroller SDK documentation, a deep sleep program was created for all microcontrollers, and the flowing current was measured again.

The results of the hardware benchmarks are summarized in Table 1. They clearly show that microcontrollers require significantly less energy than single-board computers, which was expected. But the magnitude of the differences in the different scenarios is interesting. While there is only a slight difference between the idle and full load state with the microcontrollers, single-board computers require between 1.8 to 3 times more energy in the full load state as in the idle state. This also shows that the tested microcontrollers do not have any automatic power-saving mechanisms. If, similar to the case with single-board computers, energy is to be saved when the load is low, the developer himself must carry out the necessary steps. This can be achieved, for example, by dynamically adapting the CPU clock rate or by switching various components of the microcontroller off. However, such an approach leads to much more complex programs.

Execution time of programming languages

Performance with C++ This experiment was carried out to create a unified benchmark value over all devices. This value represents the computational power of a single core per device. The time needed for one run of the prime number search was used to create this value. All devices searched for primes with an upper limit of one million. The program for this benchmark was written in C++ and it is identical for die single-board computers and the microcontrollers.

Performance with Python Due to all chosen microcontrollers supporting micro python and python being less complicated in programming than C++, the speed had to be compared. Therefore, the existing prime search benchmark was ported to python, and the experiment was repeated for all devices.

Python's level of complexity is lower compared to C++ and some tools like CMake. This enables quicker development and less complex code. However, C++ is a compiled language, and python is only an interpreted one. Therefore, it should be clear that execution times will be shorter for C++. The direct comparison from the test results 2 shows that, at least for large computational tasks, python is not well suited within a microcontroller environment. Over all

Table 1: Power consumption of all devices measured in mA. All microcontrollers were benchmarked with only a single core.

device	idle	idle BT+WiFi on	idle no keyboard (BT+WiFi on)	idle no screen (BT+WiFi on)	idle no wifi dongle	high load	high load (BT+WiFi on)	network	network + benchmark	deep sleep
Pi 0W	89	93	86	88	-	161	186	276	297	-
Pi 2B	231	290	287	288	200	424	490	387	527	-
Pi 3B	225	224	221	221	-	677	687	422	823	-
Pi 3B+	368	370	367	368	-	945	936	605	1148	-
Pi 4B	388	394	389	388	-	875	889	656	1117	-
ESP8266 @160MHz	76	84	-	-	-	81	-	84	-	5
ESP8266 @80MHz	75	79	-	-	-	77	-	81	-	5
ESP32 @240MHz	51	160	-	-	-	71	-	149	-	11
ESP32 @160MHz	40	120	-	-	-	52	-	138	-	11
ESP32 @80MHz	34	102	-	-	-	40	-	120	-	11
Pico @300MHz (1.25V)	42	-	-	-	-	51	-	-	-	2
Pico @240MHz (1.10V)	35	-	-	-	-	36	-	-	-	2
Pico @160MHz (1.10V)	25	-	-	-	-	25	-	-	-	2
Pico @125MHz (1.10V)	21	-	-	-	-	21	-	-	-	2
Pico @80MHz (1.10V)	21	-	-	-	-	18	-	-	-	2

tested microcontrollers and frequencies, C++ is approximately 47 times faster on average and on the ESP32 even more than 67 times.

WiFi performance

A separate experiment was carried out to determine whether the available WiFi bandwidth is sufficient for the data transmission of the microcontrollers and how many messages can be sent per second. For this purpose, messages containing 100 or 1000 symbols were transmitted and the transmission time was recorded. The minimal, maximal and average transmission time for 100 runs were determined.

The measured values of the WiFi transmission experiment (Table 3) clearly show that both ESPs can send more than 20 messages per second on average. Interestingly, despite the older CPU architecture and only one core, the ESP8266 requires an average of 50% less time to transmit messages. In addition, it only takes an average of 20% more time to transmit a message that is 10 times longer.

Table 2: Execution time of the programming languages in milliseconds.

Pi 0W	3964	108925
Pi 2B	3613	83689
Pi 3B	2233	49851
Pi 3B+	1917	42465
Pi 4B	1284	14438
ESP8266 @160MHz	25296	504000
ESP8266 @80MHz	50592	1016000
ESP32 @240MHz	2430	165000
ESP32 @160MHz	3658	247000
ESP32 @80MHz	7400	494000
Pico @240MHz (1.10V)	4477	205000
Pico @160MHz (1.10V)	6734	307000
Pico @125MHz (1.10V)	8638	393000
Pico @80MHz (1.10V)	13575	614000

Monitoring experiments

To evaluate the application of the wireless sensor node system for monitoring purposes, additional experiments were carried out. The aim of performing these experiments was to assess the runtime available for monitoring processes

Table 3: Time required to transmit a message of 100 or 1000 symbols via WiFi in milliseconds (based on 100 runs).

Microcontroller	Frequency	$t_{100characters}[ms]$			$t_{1000characters}[ms]$		
		Min	Max	Average	Min	Max	Average
ESP32	240MHz	27.0	105.0	39.9	30.0	215.0	43.0
ESP32	160MHz	34.0	194.0	45.3	36.0	92.0	53.5
ESP32	80MHz	45.0	235.0	61.3	53.0	96.0	71.1
ESP8266	160MHz	13.0	93.0	21.8	20.0	116.0	30.7
ESP8266	80MHz	16.0	214.0	28.2	25.0	62.0	32.7

and to check whether microcontroller-based nodes are suitable for measuring environmental parameters, in particular temperature, humidity and illuminance. A schematic representation of the sensor node setup is shown in Figure 3. One node was placed in the same room as the WiFi router, whereas the other was placed in the next room to evaluate the WiFi transmission quality of the microcontrollers' small antennas. Both nodes were placed on a windowsill where the window had no louver or drapes.

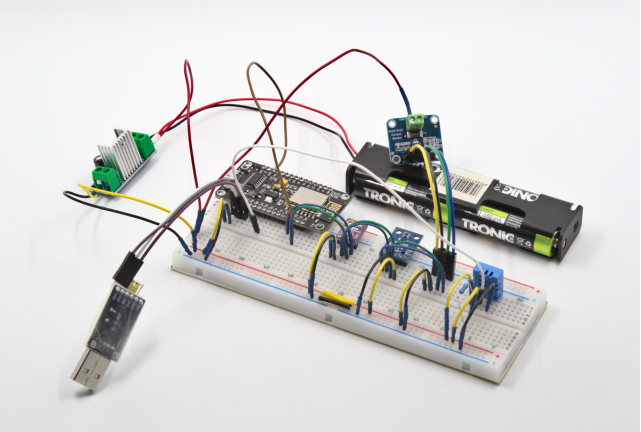

Figure 3: The ESP8266 sensor node setup.

Voltage and runtime

One of the most important questions was how long the nodes could run on battery power. While an experiment which measured the voltage of the sensor nodes was running, a rough answer can be given, as the ESP32 stopped working early on day three. This can be seen in the data evaluation in Figure 4. For better readability, the daytime of all measurements was aligned with the x-axis. For example, 6 AM on the first day of the experiment corresponds to 0.25 on the x-axis. In total, the ESP32 based node achieved an uptime of around 2.5 days, whereas the ESP8266 based one achieved a little more than six days. Based on the power consumption measurements from the previous section, the battery capacity of 2800mAh, and 5-10 seconds for a complete measurement iteration, an uptime of around one week was expected. By looking at the voltage levels of both nodes, one can see that the ESP32 was more sensitive to the voltage decrease. It stopped working at a converter input voltage of 4.9V, whereas the ESP8266 was still operational down to 4.2V. The reason for this behavior was the chosen voltage converter LM317 (Texas Instruments 2020). It was configured to deliver 3.3V using the battery's fully charged supply voltage of 5.8V. Any other configuration would have delivered a too high voltage, resulting in possibly damaging the ESPs. However, this configuration led to the earlier shutdown of the ESP32. Yet, even if not entirely successful, the results enabled many different analysis options. Some of these are now presented.

Temperature and illuminance

In the same experiment, the temperature and illumination measured by the two sensor nodes were compared. By looking at Figure 5, one can see that most measurements of the ESP32 based node are above the ones from the ESP8266 based node. This result is not surprising, as the windowsill on which the ESP32 was placed faced roughly south, whereas the ESP8266s windowsill faced roughly north. As a consequence, the ESP8266 node received more sunlight in the early hours of the day, but shortly before noon, until the later afternoon, the ESP32 received up to five times the amount of sunlight. The combination of the temperature and the illuminance shows that the temperature is directly related to the sun. If this experiment had been conducted in the winter months, it would have probably looked different. In addition, it can be seen that the coldest part of the day prevails just before sunrise. An excellent example of this behavior is the beginning of day 4. The temperature continuously decreases until around 5 AM before going back up with the rising sun.

Temperature and humidity

As already mentioned in the experimental setup, each node was equipped with a BME280 and a DHT11 sensor, which makes a comparison of the temperature and air humidity values possible. Figure 6 shows the results of measuring the same quantity with the different sensors. On the first look, the deviation between both sensors is not high. However, by additionally calculating the absolute difference between both, the picture changes slightly. Especially the ESP32-nodes measurements in the warm and sunny hours around noon reached a difference of up to 14 degrees. In comparison, the ESP8266-nodes measurements in a shady

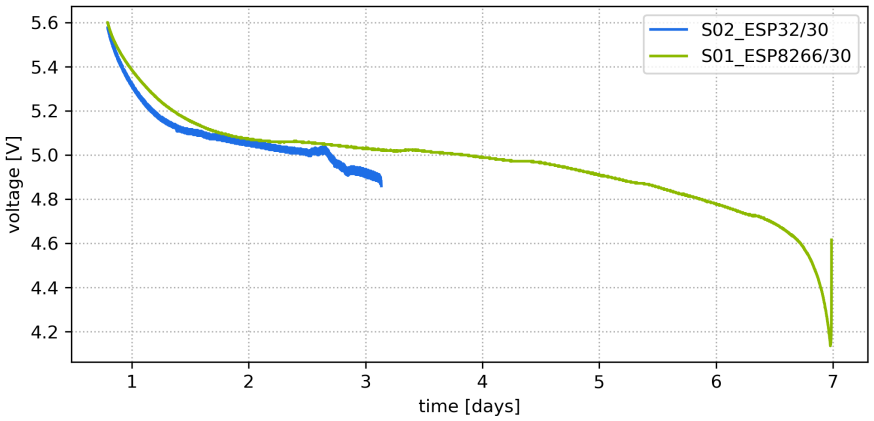

Figure 4: Results of the voltage measurement.

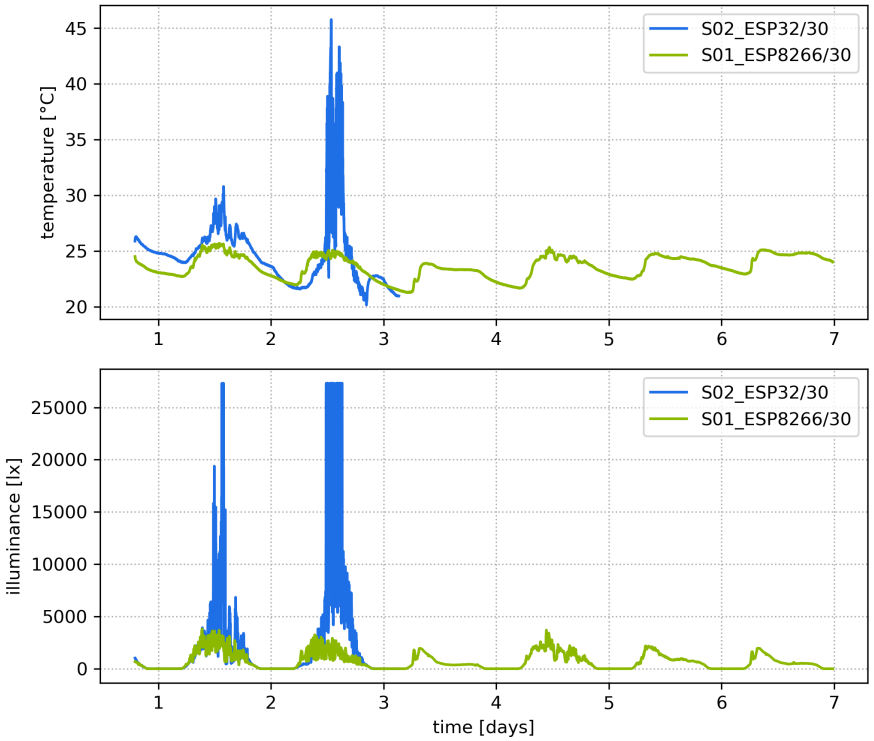

Figure 5: Results of the temperature and illuminance measurement.

location only differed by some minor degrees.

Compared to the temperature values, the humidity measurements seem to have a nearly constant difference of approximately 5% - 10% depending on the nodes location. In combination with the daytime, one can see that the humidity is highest in the night and lowest around noon. Furthermore, a high sun intensity lets the humidity drop significantly, which can be seen around noon on day two. At the end of the experiment, the DHT11 on the ESP8266 was artificially exposed to higher humidity and the measurement data reflects this (the node reached almost 100 percent humidity).

Conclusion

In this work, microcontrollers were evaluated as devices for monitoring applications. These microcontrollers were compared to single-board computers by conducting hardware benchmark tests. Moreover, a wireless sensor node system was proposed and implemented. Two sensor nodes based on different microcontrollers were used to obtain temperature, humidity and illuminance data for a small experiment.

In general, microcontrollers have proven to be an alternative of single-board computers for monitoring purposes. Although they are available at low prices, they provide a variety of interfaces via their pins, which makes the attachment of several types of sensors possible. In addi-

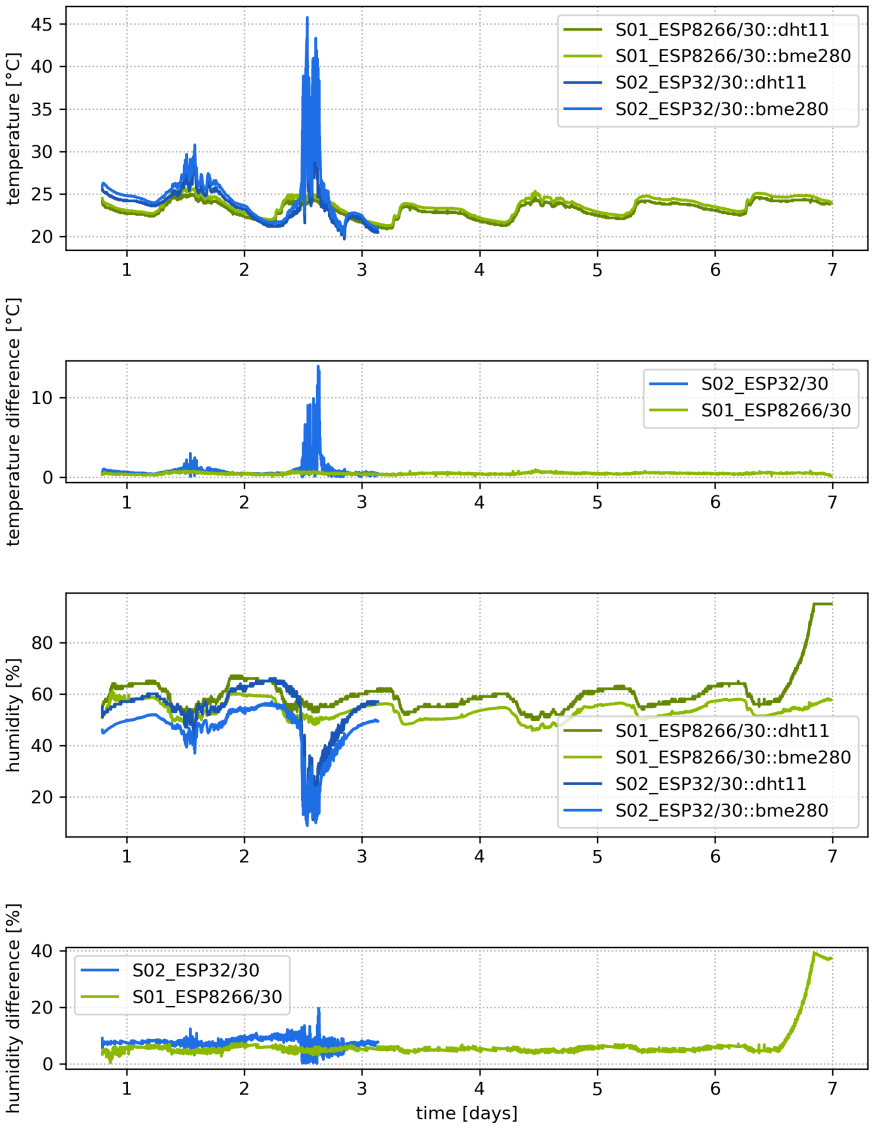

Figure 6: Results of the temperature and humidity measurement.

tion, microcontrollers have lower power consumption than single-board computers and, as a result, are more energy-efficient.

However, from a programming perspective, everything is closer to the actual hardware than in the regular operating system environments of single-board computers. But the hardware is often wrapped in some layers of abstraction offering unified access to resources of different devices. In contrast, within a microcontroller environment, solutions are mostly coupled to a specific chip or configuration. Sometimes connections on the microcontrollers have to even be hard-wired to become functional. Additionally, available resources, like time and memory, need to be considered more carefully, as some microcontrollers have such limited memory that more extensive libraries could not be used.

The focus of further research should be the lowest possible power consumption. As the experiments have shown, up to 500mA of current is required for data transmission via WiFi, which is too much for an extended battery-powered operation. Alternative wireless transmission technologies, such as ZigBee, and a minimized printed circuit board (PCB), may further reduce power consumption. Furthermore, battery recharging via solar panels for an outdoor use case will be considered. In addition, it should be possible to connect sensors that are neither purely analog nor utilizing the I^2C bus.

Acknowledgements

This work was partially supported by the Carl Zeiss Foundation. The financial support is gratefully appreciated. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the aforementioned institutions.

References

Bosch Sensortec (2015), BME280 - Combined humidity and pressure sensor (Datasheet - Revision 1.6), Bosch Sensortec GmbH.

Crookes, D., Evans, P., Halfacree, G., Hattersley, R., King, N., Monk, S. & Vanstone, M. (2021), The Official Raspberry Pi Handbook 2021, Seymour Distribution Ltd.

Despotović, I., Goossens, B. & Philips, W. (2015), MRI segmentation of the human brain: challenges, methods, and applications, Vol. 2015 of *Computational and mathematical methods in medicine*, Hindawi.

Eremia, M., Toma, L. & Sanduleac, M. (2017), The smart city concept in the 21st century, Vol. 181 of *Procedia Engineering*, Elsevier, pp. 12–19.

Espressif Systems (2020), ESP8266EX (Datasheet - Version 6.6), Espressif Systems Inc.

Espressif Systems (2021), ESP32 Series (Datasheet - Version 3.8), Espressif Systems Inc.

Healy, M., Newe, T. & Lewis, E. (2008), Wireless sensor node hardware: A review, SENSORS, 2008 IEEE, IEEE, pp. 621–624.

Kanagachidambaresan, G. R. (2021), Role of Single Board Computers (SBCs) in rapid IoT Prototyping, Springer Cham.

Lauterbur, P. C. (1973), Image formation by induced local interactions: examples employing nuclear magnetic resonance, Vol. 242 of *Nature*, Nature Publishing Group, pp. 190–191.

Malek, J. & Kaouther, M. (2014), Destructive and non-destructive testing of concrete structures, Vol. 8 of *Jordan journal of civil engineering*, pp. 432–441.

Morris, E. A. (2002), Breast cancer imaging with MRI, Vol. 40 of *Radiologic Clinics*, Elsevier, pp. 443–466.

NXP Semiconductors (2021), I^2C-bus specification and user manual (Revision 7.0), NXP Semiconductors.

Raspberry Pi Trading Ltd (2021), Raspberry Pi Pico Datasheet - An RP2040-based microcontroller board, Raspberry Pi Trading Ltd.

Ray, S. & Al Dhaheri, A. (2017), Using Single Board Computers in University Education: A Case Study, Recent Advances in Information Systems and Technologies, Springer International Publishing, Cham, pp. 371–377.

ROHM Semiconductor (2011), Ambient Light Sensor IC Series - Digital 16bit Serial Output Type Ambient Light Sensor IC (Datasheet), ROHM Co., Ltd.

Sheng, C., Li, Z., Qin, L., Guo, Z. & Zhang, Y. (2011), Recent progress on mechanical condition monitoring and fault diagnosis, Vol. 15 of *Procedia Engineering*, Elsevier, pp. 142–146.

Su, X., Tong, H. & Ji, P. (2014), Activity recognition with smartphone sensors, Vol. 19 of *Tsinghua science and technology*, TUP, pp. 235–249.

Sunrom Technologies (2012), DHT11 - Humidity and Temperature Sensor (Datasheet), Sunrom Technologies.

Texas Instruments (2015), INA219 Zerø-Drift, Bidirectional Current/Power Monitor With I^2C Interface (Datasheet), Texas Instruments Inc.

Texas Instruments (2020), LM317 3-Terminal Adjustable Regulator (Datasheet), Texas Instruments Inc.

Westkämper, E., Spath, D., Constantinescu, C. & Lentes, J. (2013), Digitale Produktion, Springer, Berlin, Heidelberg.

Zand, P., Chatterjea, S., Das, K. & Havinga, P. (2012), Wireless industrial monitoring and control networks: The journey so far and the road ahead, Vol. 1 of *Journal of sensor and actuator networks*, Molecular Diversity Preservation International, pp. 123–152.