 doi:10.20944/preprints202107.0026.v1

Article

Network of Smart Tip-Timing Sensors in Distributed Blade Health Monitoring System†

Edward Rokicki[1,‡], Jerzy Kotkowski [1,‡], Radoslaw Przysowa[1*,‡] and Piotr Filipkowski [2,‡]

[1] Instytut Techniczny Wojsk Lotniczych, ul. Ksiecia Boleslawa 6, 01-494 Warsaw, Poland
[2] Institute of Information Systems and Digital Economy, Warsaw School of Economics, ul. Madalińskiego 6/8, 02-513 Warsaw, Poland; pfilip@sgh.waw.pl
* Correspondence: radoslaw.przysowa@itwl.pl
† This paper is an extended version of our paper published in AVT-357 Research Workshop on Technologies for future distributed engine control systems (DECS) organized online by the NATO Science and Technology Organization on 11–13 May 2021.
‡ These authors contributed equally to this work.

Abstract: Blade Health Monitoring (BHM) is often necessary in power plants and in aviation to prevent excessive blade vibration and cracks. This article proposes a network of blade tip timing sensors operated in a distributed BHM system. A number of cooperating agents is implemented in smart conditioning units which can autonomously operate in an adverse environment and communicate with other nodes via a serial interface. The project uses special versions of reduced instruction set chips that are able to operate near the hot section of the engine. Due to the limited number of types of microprocessors available in the extended temperature range grading, it was necessary to fully utilize the limited hardware resources and implement preemptive multitasking. For this purpose, a custom operating system and communication protocol were designed. The protocol hosts the middle layer which hides the implementation of the distributed system. The presented architecture ensures the sufficient computational capacity in individual nodes of the network operated in adverse conditions. It is scalable and resistant to transmission errors.

Keywords: smart sensor; multi-agent system; modular architecture; Blade Health Monitoring; system on chip

1. Introduction

During the operation of turbomachinery, a vital problem is the health management and prediction of critical components, which, besides bearings, are blades. Blade vibration can be measured and monitored by the non-contact blade tip-timing (BTT) method, both in development and service. For this purpose, a set of sensors is installed around the blade tips[1]. Blade Health Monitoring (BHM) systems [2,3] that warn of excessive blade vibration and the development of fatigue cracks [4] are often used in power plants and sometimes in aviation. They are designed to prevent a serious failure of the turbine caused by blade fractures.

There are usually several turbo sets in a power station that are controlled by a distributed control system [5]. There are two or more turbine stages in each set, which are monitored by BHM systems [6,7] that are connected to the Intranet of Things (IoT) network [8] and communicate with the main control system. Each of the BHM subsystems includes two or more blade vibration sensors, each with a dedicated conditioning circuit that requires configuring the gain, trigger method, and level, and checking the health of the sensor and the quality of the generated signal. The conditioning system often operates close to the turbine, so it has to be built from high-temperature components [9,10] which can operate up to 150°C.

 doi:10.20944/preprints202107.0026.v1

Installing, configuring, testing, calibrating, servicing and replacing damaged sensors is costly and often has to be performed in hazardous conditions. To simplify these activities, the control and monitoring systems use smart sensors [11,12] that contain a microcontroller and are able to communicate with the measurement system, thanks to which the system can uniquely identify and configure them. Currently, many types of sensors such as pressure transducers, strain gauges, thermometers and accelerometers are available in smart versions, compliant with the IEEE 1451 standard, which defines the communication method and the set of essential data about the sensor known as the transducer electronic data sheet (TEDS). The standard can also be used for frequency-time sensors, which include BTT / BHM sensors [13,14], but it has not been adopted widely. In addition, sensors operating at elevated temperatures [15] are often not smart due to the lack of electronics or its relocation to external conditioning systems.

A popular protocol for communication with embedded systems is Modbus [16], which is used with serial interfaces (RS232, RS485) as well as Ethernet networks. It is suitable for configuring smart sensors or conditioning systems. Its disadvantage is centralization, i.e. devices are not peers and a master is needed to initiate the transmission.

Because the blade deflection is proportional to the time of arrival (TOA), BTT sensors basically do not require calibration like voltage pressure or tip clearance sensors [17]. Nevertheless, the condition of the sensor, the conditioning of the input signal and the time measurement method (triggering) have a decisive influence on the uncertainty of the blade vibration measurement [18]. Unfortunately, previous attempts to standardize the requirements for the sensor signal and its initial processing have not been successful [19]. The issue of BTT system validation and uncertainty estimation has recently been studied under the Clean Sky 2 program [20] using the simulator [21] and rig testing.

In development, optical BTT sensors are mainly used, while in blade health monitoring - magnetic sensors. Both types require dedicated signal conditioning and triggering systems [22]. Typically, gain, polarity, arming level, and trigger threshold are configured. One of the first commercially available digitally-controlled conditioning systems was the Blade Vibration Sensor Interface (BSVI) by Hood Tech [23], which made it possible to conveniently select gain and triggering parameters for sensors of various types.

BHM conditioning units, like other embedded systems [24], are operated in the Industrial Intranet [25]. Unlike personal computers, embedded systems are designed to perform specific tasks and are implemented in hardware with very limited system resources, in particular in microcontrollers. These chips have the data and address bus inside the integrated circuit and also the fixed structure of ROM and RAM memory. For the developed system, System on Chip (SoC) microcontrollers [26] are best suited since they have a set of advanced analog, digital and communication peripherals operating independently from the CPU. As a minimum, the control system contains only one chip that measures the process variables and generates control signals in a way that ensures the required performance and safe operation.

The article proposes a distributed system for controlling BTT sensors, which is a subsystem of a BHM system. It controls a number of smart conditioning units, which can autonomously operate in an adverse environment and communicate with other nodes via a serial interface (Figure 1). Due to the limited resources of the used microcontroller, a custom operating system and communication protocol were designed. The presented architecture has to be scalable and resistant to transmission errors.

2. Materials and Methods

2.1. Microcontroller Selection

There are no universal design methodologies for digital circuits and the specific choice of components is defined by the product specifications, the skills of the project team, and available time, funding and technology. In this case, the choice of components for the conditioning system was a trade-off between the required functions, computing power, environmental requirements, cost and demand for cooling and power. As the system is to

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0026.v1

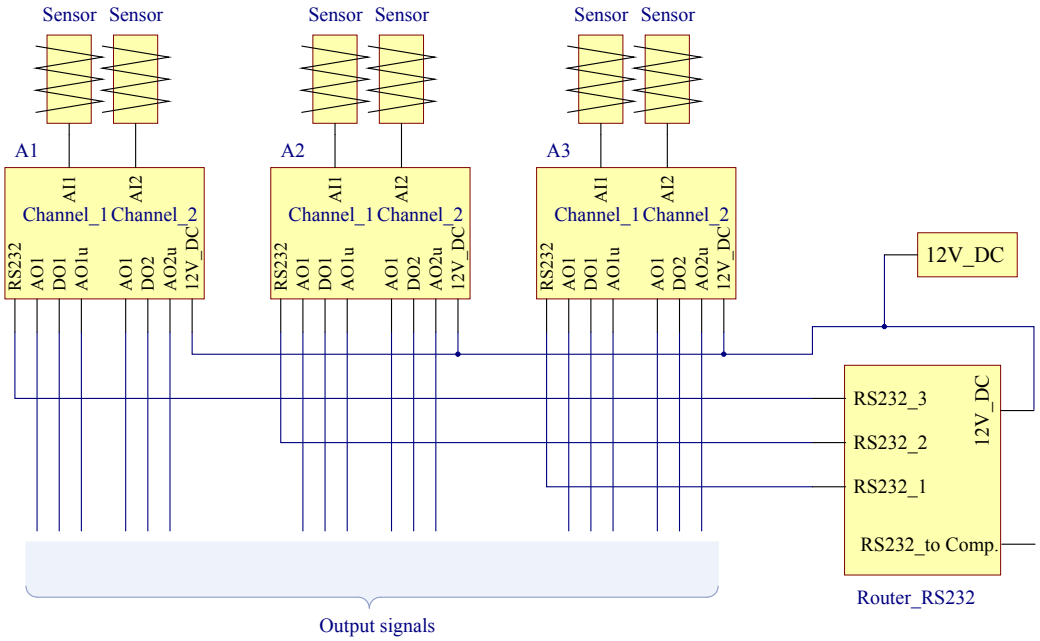

Figure 1. Conditioning system for 6 BTT sensors

operate close to the turbine, only the components were used that are available in special versions. Due to the fact that the BTT output signals are transmitted over separate cables, and communication is limited to configuration and average signal quality information, a basic serial interface (e.g. RS232) is sufficient. The Ethernet interface cannot be used because it requires the implementation of multiple software layers in the microcontroller.

In the developed system, Reduced Installation Set Computing (RISC) microprocessors such as PIC16F17XX from Microchip Technology were used [27]. They are characterized by a reduced number of instructions and addressing modes, time optimization of the instruction execution in one cycle, a high frequency of the processor clock, a large number of auxiliary registers and pipeline processing (pipelining) and reliable operation at elevated temperatures. The microcontroller uses Harvard architecture. To operate effectively, it should not use more than 50% of its resources.

The selected microcontroller should be equipped with the necessary peripherals, such as:

- Clocks and counters,
- RAM and non-volatile memory (NVM) for data
- A number of input and output ports to control peripherals
- Interfaces for external communication: EUSART, I2C and SPI
- ADC and DAC converters
- Flash memory for code

For example, 8-bit PIC16F17XX microcontrollers are equipped with Flash memory with a capacity of up to 28 KB and up to 2 KB of RAM, 16-bit and 8-bit counters, PWM as well as EUSART, I2C and SPI communication interfaces (Figure 2). The peripherals receive configuration settings, clock and synchronization pulses from the CPU. It is thanks to the peripherals that it is possible to build an independent embedded system that communicates with the world. The use of the SoC system enables miniaturization, price reduction and significantly increases the functionality and attractiveness for the user. For applications that require more processing power, more powerful Microchip or ARM microcontrollers are available, but their use is also subject to the rules outlined above.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0026.v1

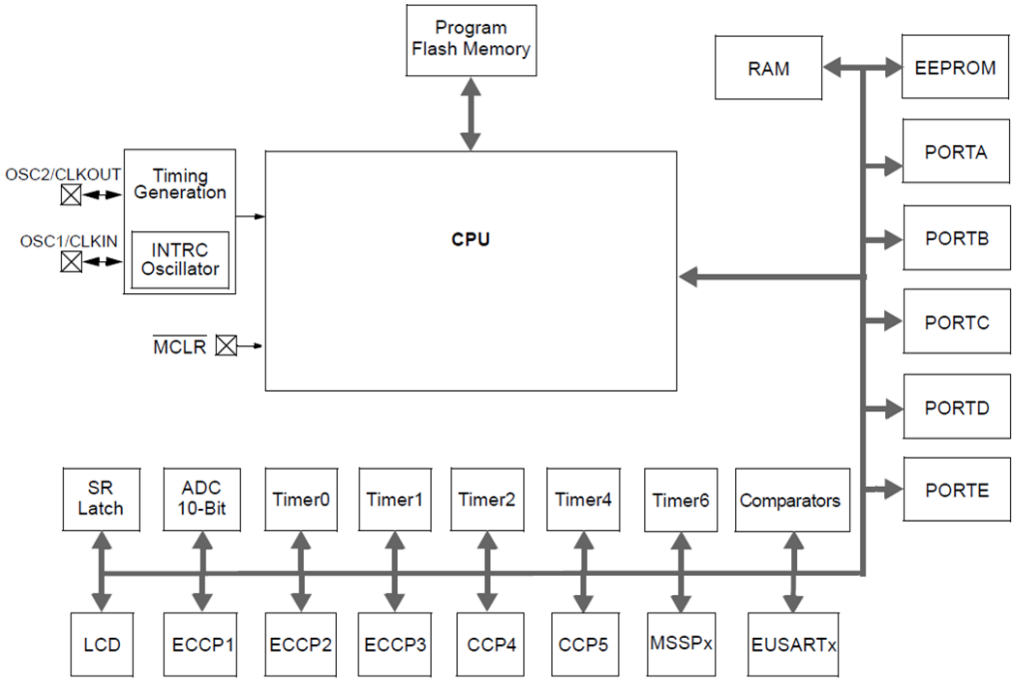

Figure 2. System on Chip - structure of a PIC microcontroller

2.2. Embedded Software

Apart from the usual requirements of accuracy and repeatability of the calculation, embedded systems must meet the requirements of limited response time because a delayed response can lead to incorrect operation of the plant. Embedded software can be implemented using the main loop only or with an interrupt-based real-time operating system which manages processor's peripheral devices, ensures access to the processor registers and IO circuits via RAM memory addressing as well as offers extensive mechanisms of errors detection and recovery (e.g. watchdog). In the presented device, due to the limited resources of the processor, it was not possible to use a ready-made operating system, but its most important functions have been implemented in a proprietary code.

The structure of an embedded application is usually based on the use of an infinite main loop where successive commands are run sequentially to perform individual tasks. The main loop implements a state machine that describes a processor in one of a finite list of states. The conditions for transition between states are strictly defined. Programming with the main loop is ideal for simple applications with single functions. This solution is simple and transparent but does not provide the ability to handle asynchronous events in real time.

The use of multithreaded programming with preemption [28] enables a low-performance microcontroller to execute advanced tasks such as voltage measurements, averaging operating parameters, controlling a programmable amplifier, counter-based measurements and a serial communication. In the proposed operating system, the main loop allocates CPU time to the threads. As a result, all threads, even those that perform compute-intensive tasks, run periodically in the background, at times determined by the counter or by external peripheral interrupts. Threads are not allowed to perform 'for' or 'while' loops. Iterative calculations must be done in the main loop or incrementally. Such software is a universal multitasking operating system featuring preemption, timing, and critical sections. It can host an application consisting of many threads working independently, having their own time and communicating with each other and with the world.

To create the code, despite the use of ANSI C, structures and principles known from object-oriented programming (OOP), i.e. abstraction, encapsulation, inheritance and polymorphism were used [29]. Each object in the system serves as an abstract "doer" that can perform the assigned task (measurements or calculations), describe and change its state,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0026.v1

and communicate with other objects in the system without revealing how the features are implemented. The object cannot change the internal fields of other objects without calling their methods. Each type of object has its own interface to interact with other objects, which determines the acceptable methods of cooperation.

At the same time, the software was developed so that it operates in the network as an autonomous software agent [30,31]. The code works independently and tries to solve problems in its environment, without the need to involve the operator if it is not necessary. The device is able to initiate communication with other nodes (without the master device). It is also able to communicate in a language that is close to natural.

The software is designed to avoid operations that, in the event of a programming error, could cause the processor to crash, thus reducing the reliability of the entire control system. In particular, the system is pre-protected against attempts to cause a device malfunction through malicious transmissions, e.g. multiple pings or notorious unauthorized transmission, which may also arise as a result of the failure of another device in the network. Such actions usually cause an overflow of the processor stack and a system hang.

The consistency of the distributed software and processed BHM data can be achieved by using a network operating system in a multitier architecture (Figure 3). The local kernels of the bottom-layer devices manage local resources (memory, data and peripherals). Some of them can be integrated and shared by the middle tier of the distributed operating system (middleware), which has to hide the fact that the underlying components are heterogeneous. It is done by defining new services and communication interfaces independent of local systems. Distributed applications use only these interfaces and should not use the services of the local operating system directly [32,33].

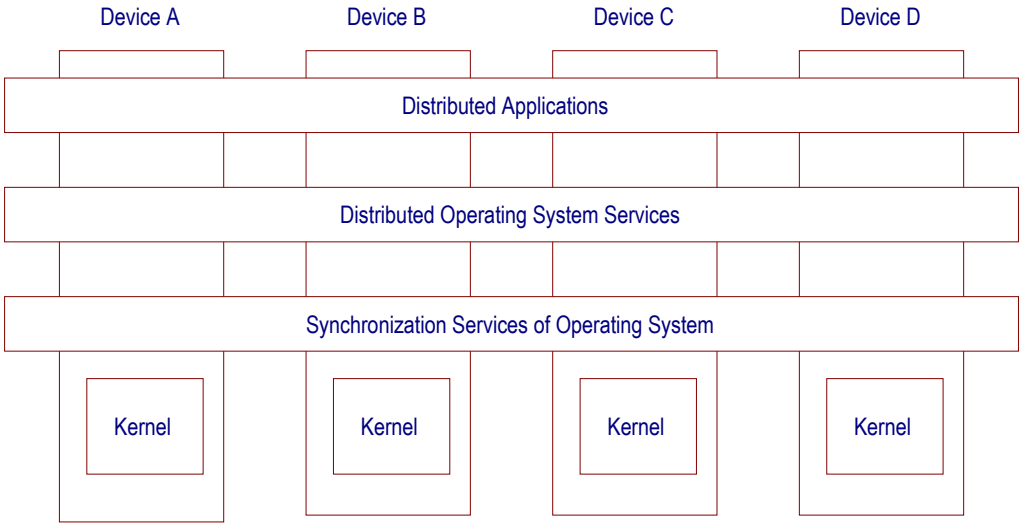

Figure 3. Architecture of distributed system

3. Results

3.1. BTT Conditioning Unit

The presented system is designed to condition signals from inductive sensors and control signal levels on a remote computer for the purpose of blade vibration data acquisition and monitoring. The previous version of the system was presented in [22]. The electronics offers the possibility of selecting the gain within wide limits, thanks to which it can effectively generate BTT signals of adequate quality even for low rotational speeds and high stand-off distances. The length of input signal cables is limited if high gains are used, so separate units are provided to enable distributed measurement configurations. The connectivity between units is carried out over the RS232 interface via a dedicated RS232 router.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0026.v1

The signal conditioning and quality, as well as BTT triggering, is controlled by a microprocessor (Figure 4). The input signals from probes are connected to analog input (AI) terminals. This signal can be attenuated in the case of high rotational speeds, bulky blade tips or low stand-off distance. Next, the signal is transmitted to the differential preamplifier with selectable gain. In the further amplifier stage, the signal can be amplified from 1 to 128 times. Then, the produced signal feeds 2 outputs: differential and unipolar as well as the trigger (tip-timing generator) and RMS detector. The output stage separates output signals from the amplifier.

The tip-timing generator produces digital BTT pulses (5 V TTL) at the digital output (DO) terminal. They correspond to zero-crossings of the analog signal. The user must activate this block in software and can select the polarity of analog signal: rising or falling edge.

The CPU controls the gains (or the attenuation ratio). RMS detector analyzes the voltage of signals from the preamplifier and amplifier to obtain their RMS levels. The averaged signals are passed to the CPU, where the analog to digital converter produces the RMS value. This measurement helps to choose the proper gain of signals and avoid noise from weak signals or saturation for strong signals. The CPU also provides bidirectional serial communication with the PC computer via RS232 needed for remote setup and control of the amplifier.

The power supply block contains AC/DC converters which ensure galvanic separation from the external power source, which is 12 V DC. However, due to the distributed structure of the measurement system, the powering and grounding of the electronics and computers has to be carefully planned to avoid ground loops and undesired current flows.

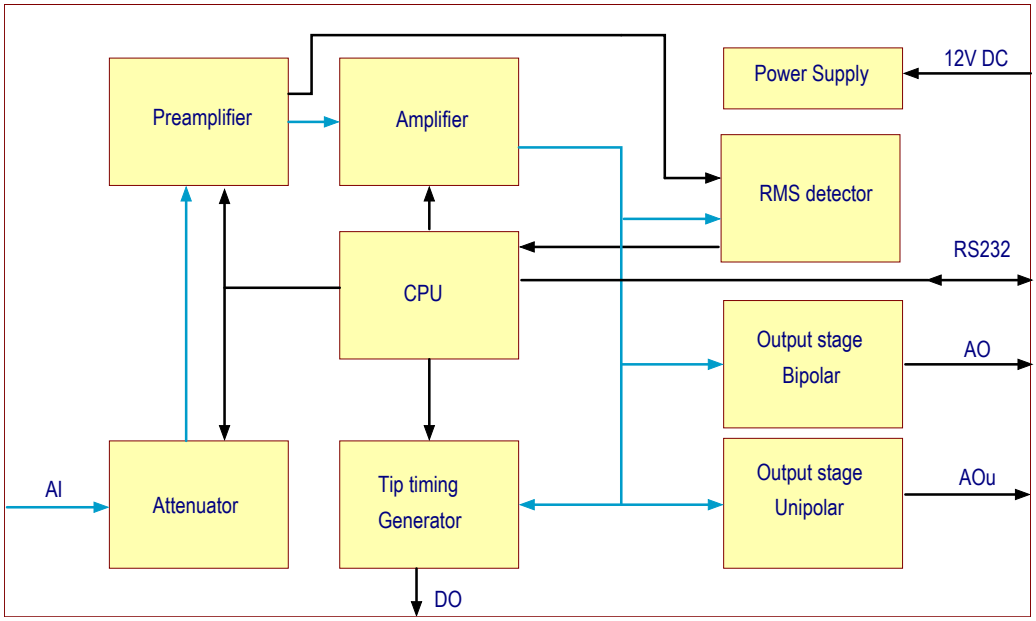

Figure 4. Block diagram of the amplifier - single channel presented

After power-up, the unconnected amplifier sets default gains (with no attenuation and disabled digital output). However, the RS232 connection and dedicated software are required to control gains and TTL output. The PC computer communicates with amplifiers via the RS232 router. The dedicated PC software has the following functions:

- Attenuation selection
- Gain selection
- Switching on / off tip-timing digital outputs
- Positive or negative triggering of tip-timing signals

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0026.v1

- Monitoring the RMS values of internal analog signals from the preamplifier and amplifier to avoid saturation

3.2. Distributed system

A network of smart conditioning units in a tree topology using RS-232 serial interface is shown in Figure 5. Communication between two devices is carried out through a custom RS232 router, which is a network node, transparent for communicating devices. Thanks to the use of routers and stacking them, the number of supported measurement channels can be large. This architecture can be upgraded to a bus topology if RS-485 is used instead of RS-232.

A set of smart conditioning units connected through a serial interface creates a loosely coupled distributed BHM system (Figure 6) in which a faulty node does not interfere with communication and will not cause a failure of the entire system. Devices in the network are autonomous and communicate peer to peer, so they form a multi-agent system. Analog and digital BTT signals are transferred from conditioning systems to data processing units via separate cables.

Despite the significant technological progress, systems based on microcontrollers have many limitations. Smart conditioning units are primarily responsible for ensuring the health and performance of sensors while more complex tasks such as BTT processing and fault detection and isolation are performed by networked high performance computers working in higher layers of the system which synthesize sensor data collected from various locations.

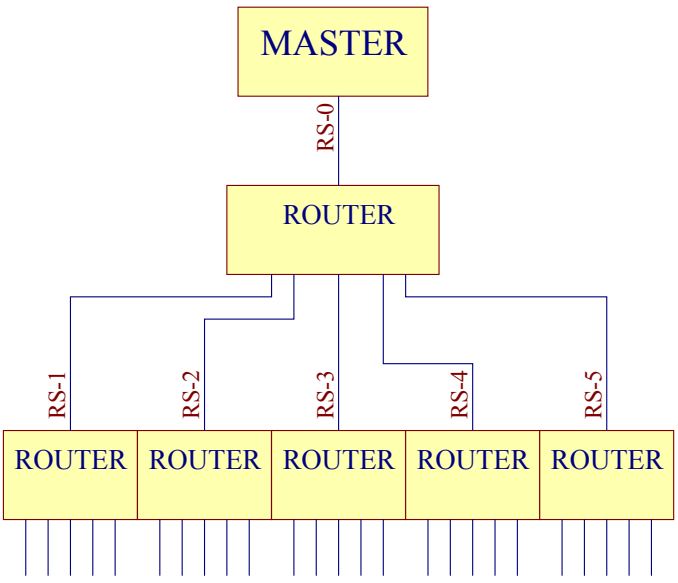

Figure 5. Tree-shaped network of embedded devices connected via RS-232 interface

3.3. Communication Protocol

A custom protocol [34] was developed to send the configuration and health of sensors between devices connected via RS232. It was inspired by DHCP and Modbus protocols. The transmitted message consists of the number of fields in the ASCII code. Similarly to the TCP/IP model, the structure of the data frame is layered (Table 1), which makes it possible to implement new applications in the future. The transport tier contains the sender and recipient identifiers, a coded function (SET, QUERY and ANSWER), CRC checksum and also hosts the intermediate tier with function-specific data (Table 2).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 July 2021 doi:10.20944/preprints202107.0026.v1

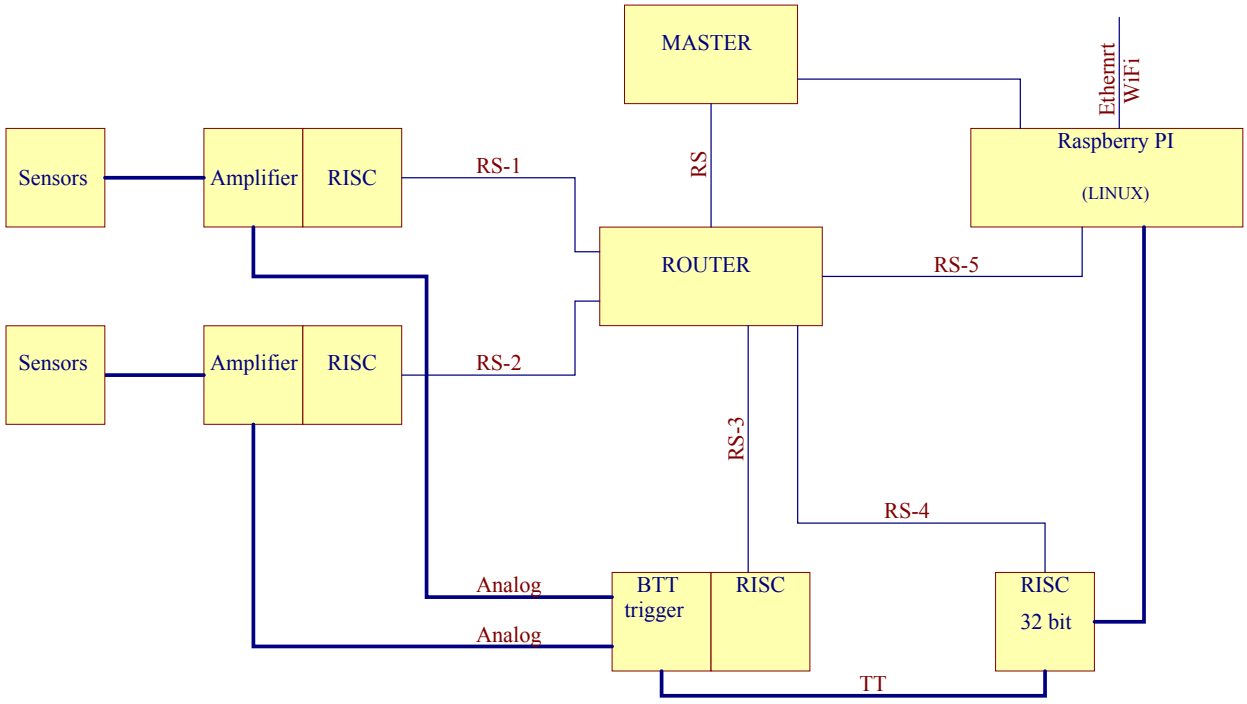

Figure 6. Loosely coupled distributed system based on IoT devices

Table 1. Abstraction layers

Layer	Task
Middleware	High level communication
Transport	Sending and receiving frames
Physical	Sending and receiving bytes

The intermediate layer of the data frame contains variables describing the configuration and health monitoring of sensors and the coded function or command specific to this layer. If necessary, it can contain another layer (the application tier) or messages / commands formulated in natural language. The protocol with a multitier data frame enables the various nodes of the distributed system to have controlled access to local resources and data.

It is possible to send up to 9999 characters in a single transmission. After detecting an error, the transmission is obligatorily abandoned and the system waits for the next message. Communication between two devices can be carried out directly via a standard RS232 cable or via a custom RS232 router that supports up to six devices now. The messages are sent in ASCII code, which means that characters with codes 0-31 are used to control the transmission.

In the code implementing the communication protocol, a software trap was introduced that simulates the correct reception of a malicious transmission, a blockade of incrementing the transmission counter and necessary interrupts. The prolonged operation of the trap causes sending information about the timeout and error code to the transmission node. The addressee of such transmission is made void in order not to forward it and not to generate unnecessary replies. The embedded system then goes to the procedure of failure minimization and concealment. Disabling the trap returns the system to its normal state.

 doi:10.20944/preprints202107.0026.v1

Table 2. Data frame

Name	Length (bytes)	Function
SOH	1	Start of header
Length	4	Frame length in bytes
Error	1	Error code
Timeout	1	Timeout value
Function	1	Function code
Sender	1	Sender address
Recipient	1	Recipient address
US	1	Unit separator
Variable	1	Variable number
Value	4	Variable value
US	1	Unit separator
ETB	1	End of transmission block
EOT	1	End of Transmission

4. Conclusions

The presented network of autonomous agents operating in a distributed BHM system was implemented in hardware suited for the adverse environment around the turbine. This architecture facilitates multi-sensor tip-timing data acquisition but can be adapted for other advanced measurements and data analysis needed in a prognosis and health management system or an engine control system operated at elevated temperature. Increasing the continuous operating temperature of the electronics from 85°C to over 150°C not only significantly increases the overall system reliability but also offers more flexibility in distributing its components around the engine. This reduces length of cables, weight and demand for cooling.

The developed custom software made it possible to implement advanced network features in low-performance microcontrollers which can operate at high temperature. The proposed communication protocol enables flexible configuration and monitoring of BTT sensors but can also host other data and functions such as encryption and permissions. It will be adopted for other engine and ground applications e.g. oil debris monitoring.

Further research on distributed health management systems will take into account the Zero Trust Architecture [35] and DevSecOps [36] methodology. This will make these systems safer and more sustainable which is indispensable in aerospace and defence.

Author Contributions: Conceptualization: E.R. and P.F.; Investigation: J.K, E.R. and R.P.; Methodology: E.R., R.P. and P.F.; Project administration: R.P.; Software: J.K. and E.R.; Supervision: R.P. and E.R.; Visualization: J.K. and E.R.; Writing—original draft: J.K., R.P., E.R. and P.F.; Writing—review & editing, R.P. All authors have read and agreed to the published version of the manuscript.

Funding: This publication includes the results of the statutory activity of ITWL 'Demonstrator of the system to measure turbine blade vibration', financed by the Ministry of Science and Higher Education in 2018-2020.

Institutional Review Board Statement: Not applicable

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

Abbreviations

The following abbreviations are used in this manuscript:

AC	Alternating current
ADC	Analog-to-Digital Converter
AI	Analog input
AO	Analog output
ASCII	American Standard Code for Information Interchange
AVT	Applied Vehicle Technology Panel
BHM	Blade Health Monitoring
BSVI	Blade Vibration Sensor Interface
BTT	Blade Tip Timing
CPU	Central Processing Unit
CRC	Cyclic Redundancy Code
DAC	Digital-to-Analog Converter
DAQ	Data Acquisition system
DC	Direct current
DECS	Distributed engine control systems
DO	Digital output
EUSART	Enhanced Universal Asynchronous Asynchronous Receiver Transceiver
I2C	Inter-Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ITWL	Air Force Institute of Technology in Warsaw
NATO	North Atlantic Treaty Organization
NVM	non-volatile memory
PC	Personal computer
RAM	Random-access memory
RISC	Reduced Instruction Set Computing
RMS	Root Mean Square
ROM	Read only memory
SoC	System on Chip
SPI	Serial Peripheral Interface
TCP/IP	Transmission Control Protocol/Internet Protocol
TOA	Time of arrival
TTL	Transistor–transistor logic

References

1. Przysowa, R.; Russhard, P. Non-Contact Measurement of Blade Vibration in an Axial Compressor. *Sensors* **2020**, *20*, 68. doi:10.3390/s20010068.
2. Badami, V. Advanced Sensors and Monitoring & Diagnostics (M & D) for Gas Turbines. EE392n: Intelligent Energy Systems, Lecture: April 24, 2012.; , 2012.
3. Russhard, P. The Use of Blade Tip Timing Technologies to Assess and Monitor Rotor Blade Health from Design to Production. STO-MP-AVT-229 - Test Cell and Controls Instrumentation and EHM Technologies for Military Air, Land and Sea Turbine Engines; NATO Science and Technology Organization: Rzeszów, 2015; pp. 1–27.
4. Wu, S.; Wang, Z.; Li, H.; Yang, Z.; Tian, S.; Yan, R.; Wang, S.; Chen, X. Blade Crack Detection Using Blade Tip Timing. *IEEE Transactions on Instrumentation and Measurement* **2021**, *70*. doi:10.1109/TIM.2020.3046926.
5. Vyatkin, V. The IEC 61499 standard and its semantics. *IEEE Industrial Electronics Magazine* **2009**, *3*, 40–48. doi:10.1109/MIE.2009.934796.
6. Yu, L.; Shrivastava, S. Distributed Real Time Compressor Blade Health Monitoring System. *PHM Society* **2016**, pp. 1–8. doi:10.36001/phmconf.2016.v8i1.2498.
7. Przysowa, R. Blade Vibration Monitoring in a Low-Pressure Steam Turbine. Volume 6: Ceramics; Controls, Diagnostics, and Instrumentation; Education; Manufacturing Materials and Metallurgy; American Society of Mechanical Engineers: Oslo, 2018; pp. 1–11. doi:10.1115/GT2018-76657.
8. Akhtar, M.M.; Rizvi, D.R.; Ahad, M.A.; Kanhere, S.S.; Amjad, M.; Coviello, G. Efficient Data Communication Using Distributed Ledger Technology and IOTA-Enabled Internet of Things for a Future Machine-to-Machine Economy. *Sensors* **2021**, *21*, 4354. doi:10.3390/s21134354.
9. Allard, B.; Buttay, C.; Martin, C.; Morel, H. Considerations for High Temperature Power Electronics. 18th International Symposium on Power Electronics Ee2015, Novi Sad, Serbia, 2015.
10. Stoica, L.; Riches, S.; Johnston, C. *High Temperature Electronics Design for Aero Engine Controls and Health Monitoring*; River Publishers: Gistrup, Denmark, 2016; pp. 1–162. doi:10.13052/rp-9788793379244.
11. Wilson, J.S., Ed. *Sensor Technology Handbook*; Newnes: Burlington, MA, USA, 2005.

12. Munera, E.; Poza-Lujan, J.L.; Posadas-Yagüe, J.L.; Simó-Ten, J.E.; Noguera, J.F.B. Dynamic Reconfiguration of a RGBD Sensor Based on QoS and QoC Requirements in Distributed Systems. *Sensors (Switzerland)* **2015**, *15*, 18080–18101. doi:10.3390/s150818080.
13. Yurish, S.Y. IEEE 1451 Standard and Frequency Output Sensors: How to Obtain a Broad-Based Industry Adoption? *Sensors & Transducers Magazine (S&T e-Digest)* **2005**, *59*, 412–418.
14. Viegas, V.; Postolache, O.; Dias Pereira, J. Transducer Electronic Data Sheets: Anywhere, Anytime, Anyway. *Electronicsweek* **2019**, *8*, 1345. doi:10.3390/electronics8111345.
15. Murugan, M.; Walock, M.; Ghoshal, A.; Knapp, R.; Caesley, R. Embedded Temperature Sensor Evaluations for Turbomachinery Component Health Monitoring. *Energies* **2021**, *14*. doi:10.3390/en14040852.
16. Modbus Organization. https://modbus.org/, accessed on 2021-04-13.
17. Gil-García, J.; Solís, A.; Aranguren, G.; Zubia, J. An Architecture for On-Line Measurement of the Tip Clearance and Time of Arrival of a Bladed Disk of an Aircraft Engine. *Sensors* **2017**, *17*, 2162. doi:10.3390/s17102162.
18. Russhard, P. Blade Tip Timing (BTT) Uncertainties. AIP Conference Proceedings 1740. 12TH International A.I.VE.LA. Conference on Vibration Measurements by Laser and Noncontact Techniques: Advances and Applications; Tomasini, E.P., Ed.; , 2016. doi:10.1063/1.4952657.
19. Hayes, B.C. ISA-RP107.1-201x Tip Timing Waveform Quality. Technical report, International Society of Automation, 2016.
20. Russhard, P. Batista - Blade Tip Timing System Validator, a Clean Sky 2 Project. Grant Agreement No 862034. Technical report, EMTD Ltd., 2019.
21. Mohamed, M.E.; Bonello, P.; Russhard, P. An Experimentally Validated Modal Model Simulator for the Assessment of Different Blade Tip Timing Algorithms. *Mechanical Systems and Signal Processing* **2020**, *136*. doi:10.1016/j.ymssp.2019.106484.
22. Przysowa, R.; Kaźmierczak, K. Triggering Methods in Blade Tip-Timing Systems. In *Proceedings of the Twelve International Conference on Vibration Engineering and Technology of Machinery, VETOMAC XII*; Rządkowski, R.; Szczepanik, R., Eds.; Instytut Techniczny Wojsk Lotniczych: Warsaw, 2016; pp. 129–138. doi:10.13140/RG.2.2.21687.52645.
23. Blade Vibration Sensor Interface BVSI 5. User's Manual. Technical report, Hood Technology Corporation, 2007.
24. Berger, A.S.; Berger, A.H. *Embedded Systems Design: An Introduction to Processes, Tools, and Techniques*; CMP Books, Taylor & Francis, 2002.
25. Hinojosa-Palafox, E.A.; Rodríguez-Elías, O.M.; Hoyo-Montaño, J.A.; Pacheco-Ramírez, J.H.; Nieto-Jalil, J.M. An Analytics Environment Architecture for Industrial Cyber-Physical Systems Big Data Solutions. *Sensors* **2021**, *21*, 4282. doi:10.3390/s21134282.
26. Jerraya, A.A.; Yoo, S.; Wehn, N.; Verkest, D. *Embedded Software for SoC*; Springer US, 2007.
27. Microchip PICmicro Mid-Range MCU Family Reference Manual. Technical report, Microchip Technology Inc, Chandler, AZ, USA, 1997.
28. Lamie, E.L. *Real-Time Embedded Multithreading: Using ThreadX and ARM*; Taylor & Francis, 2005.
29. Schreiner, A. *Object oriented programming with ANSI-C*; Axel T. Schreiner / Lulu: Morrisville, NC, USA, 2011; p. 223.
30. Negenborn, R.R. *Multi-Agent Model Predictive Control with Applications to Power Networks*; TU Delft: Delft, Netherlands, 2007; p. 173.
31. Zong, S.; Tian, Y.P. Consensus of Multi-Agent Systems with Unbounded Time-Varying Delays. *Applied Sciences (Switzerland)* **2021**, *11*. doi:10.3390/app11114944.
32. Insaurralde, C.C. Fully-Deterministic Execution of IEC-61499 Models for Distributed Avionics Applications. *Aerospace* **2018**, *5*, 1–30. doi:10.3390/aerospace5010015.
33. Zhang, W.; Liu, J.; Cheng, L.; Filho, R.S.; Gao, F. A Survey of Optimal Hardware and Software Mapping for Distributed Integrated Modular Avionics Systems. *Applied Sciences (Switzerland)* **2020**, *10*. doi:10.3390/APP10082675.
34. Filipkowski, P.; Rokicki, E. Inteligentne rozwiązania komunikacyjne w obszarze Industrial Internet of Things (Smart communication solutions in the area of Industrial Internet of Things). In *Techniczno-społeczne uwarunkowania gospodarki cyfrowej (Technical and social conditions of the digital economy)*; Kobyliński, A.; Filipkowski, P., Eds.; OW SGH: Warsaw, Poland, 2020; pp. 217–236.
35. Rose, S.; Borchert, O.; Mitchell, S.; Connelly, S. Zero Trust Architecture. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, 2020. doi:10.6028/NIST.SP.800-207.
36. Hsu, T. *Hands-On Security in DevOps: Ensure continuous security, deployment, and delivery with DevSecOps*; Packt Publishing, 2018.