Pomiary Automatyka Robotyka, ISSN 1427-9126, R. 26, Nr 4/2022, 43–51, DOI: 10.14313/PAR_246/43

System Architecture for Development and Supervision of Robotic Astronomical Telescope

Patryk Bartkowiak, Radosław Patelski, Marta Kwiatkowska, Dariusz Pazderski
Poznan University of Technology, Institute of Automatic Control and Robotics, Piotrowo 3a, 60-965 Poznan, Poland

Abstract: In this paper the novel control and communication scheme designed to ease the development and maintenance of the robotic astronomical telescope device is presented. The proposed solution allows the user to remotely access any signal in the controller of the telescope without imposing any additional overhead during telescope operation. The implemented scheme can be used by both an automated control system and human operators for easy supervision, control, and maintenance of the device.

Keywords: Embedded computer control systems and applications, Remote servicing, Tele-maintenance, Condition Monitoring, Fault detection and diagnosis

1. Introduction

Development of autonomous robotic telescopes poses considerable challenges concerning reliability and maintainability of the supervised system. Devices designed with the aim of deployment in distant isolated locations, with possibly harsh and changing environmental conditions, require special means of supervision and maintenance to ensure their constant and infallible operation. Ivanescu et al. [6] presented some challenges of the telescope working in such conditions, which resulted in the need for constant supervision by a human operator. The demand for development of modern systems that allow easy and automated supervision or maintenance has recently been reported by Marchiori et al. [11]. Importantly, proposed solutions have to be lightweight in terms of computational resources as the performance of the telescope controller cannot be hindered during normal operations. Moreover, the need for a simplified graphical user interface for the control of the telescopes by the final user has also been reported, e.g. Abareshi et al. [1] with growing interest in taking advantage of modern web-based technologies as used by Sadeh et al. [12] or Edwards et al. [3]. Satisfaction of all these requirements introduces significant difficulty for any team working on the development of automated robotic telescopes.

In this paper a novel framework for software architecture designed to ease the development and maintenance of autonomous telescopes is presented. Recently, this architecture has been successfully employed in the SkyLab laboratory established at the Poznan University of Technology in 2018. The progresses in development of the SkyLab were already reported by Kozlowski et al. [8]; Krysiak et al. [10]; Kozłowski et al. [9].

The proposed framework takes advantage of an internal communication scheme, which makes it possible to monitor every single signal in the software realization of the telescope controller without affecting the real-time operation of the control loop. Multiple client applications able to communicate with the controller were implemented using this approach. They include services used to automatically synchronize the clock of the controller, acquire and store the data produced during the operation of the telescope, and control of the telescope using a graphical interface, either for maintenance or regular observations. The use cases presented in this paper confirm the wide applicability of the proposed framework.

The rest of the paper is organized as follows. Section 2 presents the overall structure of the discussed telescope system. In Section 3 details of the internal communication scheme are described. Section 4 presents the client application designed mainly for astronomical operations and Section 5 describes the client application for maintenance and development. In Section 6 the example of some possibilities of the proposed system is presented. Section 7 concludes the paper.

2. Telescope System Architecture

The considered software architecture was designed for a set of astronomical telescope mounts currently developed at the Institute of Automatic Control and Robotics. The discussed collection consists of autonomous robotic mounts for a single telescope of diameter of either 0.5 m or 1 m. Currently, all of the SkyLab mounts work under the same framework presented in this paper. The example of the mount used in SkyLab is given in Fig. 1.

The developed solution consists of several interconnected devices including Mimas – Spartan 6 FPGA, microcontroller (MCU) STM32H743ZI2 with the high performance CPU ARM Cortex-M7 operating up to 480 MHz, an electronic security system with energy dissipation function (UZE), the compu-

Autor korespondujący:
Dariusz Pazderski, dariusz.pazderski@put.poznan.pl

Artykuł recenzowany
nadesłany 21.11.2022 r., przyjęty do druku 05.12.2022 r.

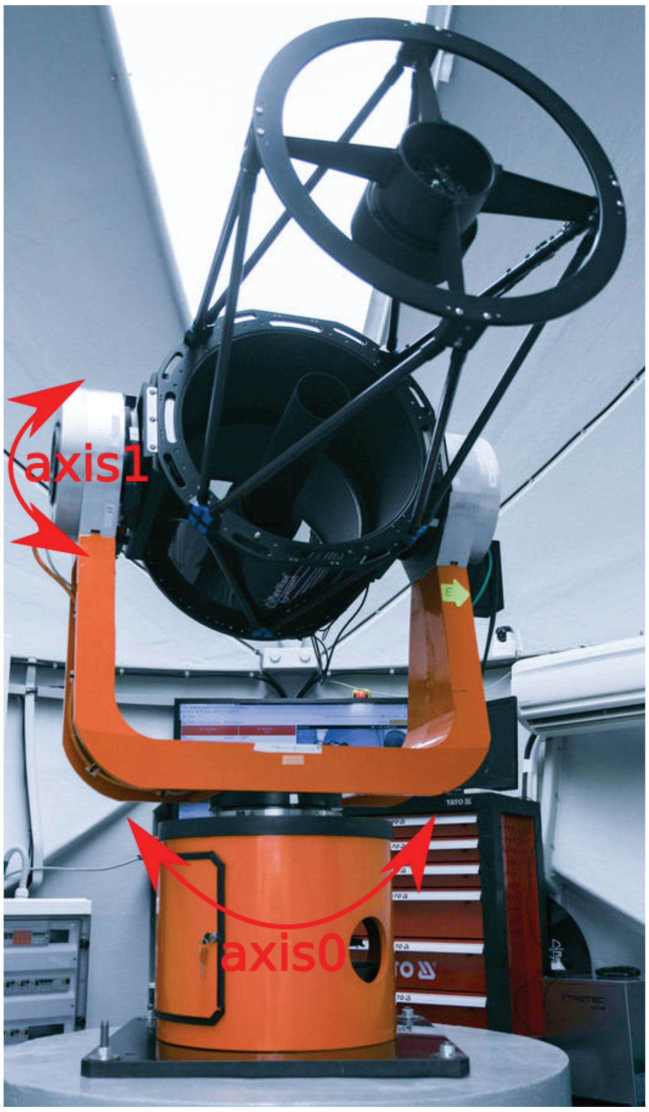


Fig. 1. The mount carrying a 0.5 m class telescope (PlaneWave CDK20 0.51-m f/6.8 Corrected Dall-Kirkham with resolving power 0.28 arcsec and camera FLI PL16803 CCD 4096 × 4096 9 μm pixels with resolution 0.54 × 0.54 arcsec/px and field of view 0.61 × 0.61 deg) used in SkyLab
Rys. 1. Montaż z teleskopem klasy 0,5 m (PlaneWave CDK20 0.51-m f/6.8 Corrected Dall-Kirkham z maksymalną zdolnością rozdzielczą 0,28 arcsec oraz camerą FLI PL16803 CCD 4096 × 4096 9 μm piksele o rozdzielczości 0,54 × 0,54 arcsec/px z polem widzenia 0,61 × 0,61 deg) stosowany w SkyLab

ter Raspberry Pi 4 (RPI) and the miniature NTP/PTP time server NTS-pico3 (GPS). The architecture of this system, with all channels of data exchange between the nodes, is graphically presented in Fig. 2.

2.1. Hardware-software structure

The STM32 is a main real-time controller which is responsible for performing low-level tasks such as processing data, computing trajectory, controlling motors and overviewing all external devices to ensure faultless operation. The control system loop is designed to work with a 10 kHz frequency for both axes of the telescope mount. Using the I²C interface, the STM32 communicates with the UZE module to obtain the temperature and current measurements of the motors and to adjust a supply voltage or energy dissipation function settings. The main controller program is written in a standard C/C++ programming language. Here, the novel programming framework introduced by Gawron and Kozłowski [4] was employed to manage the code structure and ensure the exchange of information between separate modules. The FPGA is used mainly for signal processing and gathering of measurements, including telescope positions, status of encoders, supply voltage and current of the drives, which later are sent to the main STM32 controller. The communication between FPGA and STM32 is made possible by employing a fast SPI interface with STM32 in a role of a main device. FPGA is also tasked with exercising custody over drives controller as a watchdog and, in case of failure, interrupt PWM signals which control the drives. The RPI runs Linux-based Raspbian operating system with realtime Preempt RT kernel modified to incorporate support of the Precise Time Protocol (PTP). The real-time system is crucial to guarantee a minimum latency between external interrupts and the interrupt handling, and consistent behavior of thread scheduling. Due to its networking capabilities, the RPI platform serves mainly as a relay for communication between the STM32 controller and external client applications. The communication between STM32 and RPI through the SPI interface is adjusted to use the ASTCom communication library, which is described in detail in Section 3. As the RPI is the only computer-scale node in the considered system, it also hosts several of those client services. The Relay program provides service support with the SPI interface and transmission of data between STM32 and external clients, including a Virtual Star Server (VSS), Database Relay, Robot Supervisor Panel (RSP) or Time Synchronizer (TimeS). The task of Database Relay is to intercept the data streaming from the STM32 and save all transmitted data to an external database for future reference. The Virtual Star application is designed to ensure a proper operation of the telescope in astronomical observation tasks, while diagnostic operations can be carried out through a multi-function web application RSP. The RPI communicates with the external world by Ethernet connection, taking advantage of Websocket or HTTP protocol.

2.2. Time synchronization

In a case of astronomical observations, an availability of reliable clock source is a crucial requirement to obtain a high level of tracking accuracy, as it is vital to calculate a position of an observed object in a given time instant. Thus, a custom time

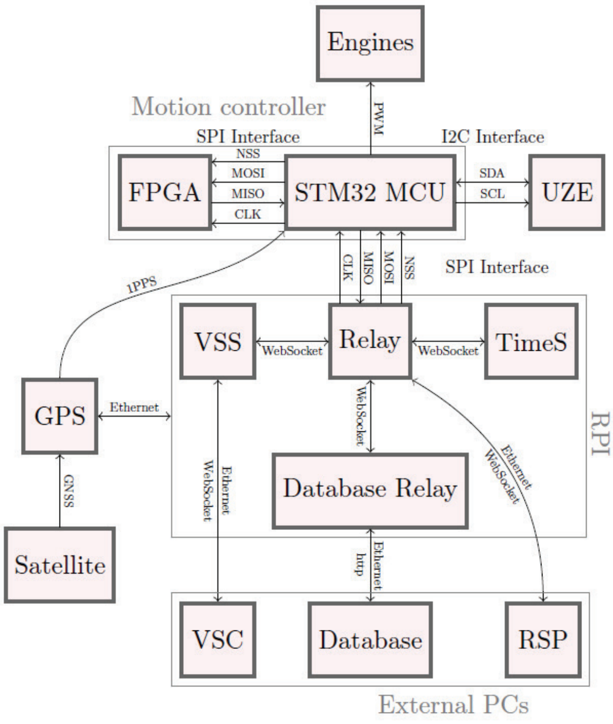


Fig. 2. Software, hardware and communication architecture of the telescope system
Rys. 2. Oprogramowanie, osprzęt i architektura systemu komunikacji w układzie teleskopu

synchronization module was developed using a proposed communication scheme. To this end, a GPS module is employed, which is connected to the RPI using an Ethernet standard. This enables the RPI to periodically update its internal system clock through the PTP protocol, as described in Cochran et al. [2]. However, in order to employ the PTP synchronization, the system kernel has to provide control of the hardware clock and packet timestamping at the hardware or software layer, which is ensured by the aforementioned kernel modifications. Once the synchronization of its system clock is performed, the RPI can be treated as the main source of time measurements in the considered system. By taking advantage of this notion, the program TimeS reads the synchronized system timestamp of the hindmost full second and transfers it in the UTC format to the STM32 using the proposed ASTCom approach, thus updating the internal clock of the telescope control system. In order to avoid time diverging, an additional one 1PPS (a pulse per second) signal is routed directly from the GPS module into one of the digital inputs of the STM, which enables the control system to precisely assign the received UTC timestamp to the time instant marked by the 1PPS pulse.

2.3. Database

Data storage during the operation of the automatic control system is an element of high importance, as it was emphasized by Story et al. [13]. In particular, thanks to recorded signals, it is possible to conduct an in-depth analysis of the conducted experiments, as well as to diagnose the cause of the possible system defects based on the data collected over a wide time horizon. An open-source time series database InfluxDB installed on an external server with plenty of storage space is used for gathering and saving data from the controlled system. As has been mentioned earlier, the program Database Relay, running on the RPI machine, catches streamed signals and saves data into the database using InfluxDB's custom command language similar to mySQL by the HTTP protocol. An arbitrary choice of stored data is given to the user of the telescope system and these can be defined using the RSP and VS applications, both of which are equipped with a user-friendly GUI. To access the database and download the data series, one can make use of the HTTP protocol directly or take advantage of a custom application that works as a module of the MATLAB environment. Any calculations, visualization, or data processing can also be done in the MATLAB program.

3. Communication Library Design

In order to facilitate the communication and data exchange between various nodes of the telescope environment, the novel communication protocol called the Astronomic Communication Library (ASTCom) was proposed and implemented. The library provides an interface for serialization and real-time exchange of data between various devices through the Websocket protocol. The serialization itself is performed using a custom format similar to the well-known MessagePack format enhanced with several extensions to accommodate features of the proposed scheme. Mainly, the possibility to dynamically configure the set of exchanged data at run-time is introduced.

The basis of the ASTCom library design is the system of variable and class registration, which allows the library to discover each and every desired variable declared in the code of the telescope controller. To this end, a conceptual ASTVariable is defined as any variable or class that can be serialized by the ASTCom library. Serialization procedures for multiple basic types (e.g. integer, float, string, array, etc.) are predefined by the library itself. Moreover, ASTVariable can also represent a function with an arbitrary signature. In order to enable serialization of custom defined classes, the special macro `AST_VARIABLE_DEFINE` is declared which, when included in the class definition, declares a set of functions used to serialize and deserialize all variables of the chosen class. The types of variables and a proper way of their processing are discovered automatically by the library. Thus, once a class is defined, a call to a single function automatically serializes an entire content of its object, including any member objects that were implemented taking advantage of the `AST_VARIABLE_DEFINE`. A globally accessible Collection class is then defined, which can be called to recursively scan and register any object of the ASTVariable type. During this process, information about name, type, address of data in physical memory and addresses of serializing and deserializing functions of the object are stored in memory of the Collection singleton. Due to to the recursiveness of this operation, it is sufficient to register only the top level object, provided that all member objects are also of the ASTVariable type. Thus, if any change in the structure of the software is made, there is no requirement for any additional modifications outside the affected classes, as their proper registration is automatically ensured by the top level class. It is of major significance that all of these operations are performed either at compile time or on the device startup and does not in any way slow down main computation tasks executed online by a CPU. Once the initial registration of objects is performed, the current state of any signal in the telescope can be easily obtained by invoking the Collection object with name and type of the required variable.

In order to make use of the proposed approach, two separate modules are implemented – the Endpoint module, to be employed on board of the telescope, and the Client module, run in each of the client applications. Both modules are derived from the single Interface class. These two are then used to establish communication between various devices in the ASTCom network. To this end, a series of hard-coded ASTCommands is defined and used to exchange basic commands between devices. These are in nature similar to functions of the Modbus protocol and are first used to establish a formal connection between the nodes, during which version compatibility is verified and access rights are granted through password verification. Each connection is represented in both parties by a separate Connection class object, which stores all information necessary to carry on the communication. It has to be noted that multiple Client devices can be simultaneously connected to a single Endpoint and their number can be limited by the Endpoint to reduce the performance impact. Once the connection is defined, the Client uses proper ASTCommands to query the Endpoint device and request it in order to define new ASTMessages – virtual structures consisting of several ASTVariables with a unique ID number assigned. Once the Endpoint receives such request, it queries the Collection for desired entries and copies pointers to serialization functions. Hence, once the ASTMessage is defined in the Endpoint, it is directly available for use, and no additional overhead is created beside brief configuration of the messages, which can be done before the proper start of operation of the telescope system. On the Client side, the newly defined ASTMessage is also bound to a chosen locally defined variable of the same type. It is noteworthy that as the serialization procedures of the basic types are hard-coded into the library, the Client device is not required to have counterparts of the defined data in its source code, as they can always be built in the runtime from objects of basic types. On both sides, defined ASTMessages are stored in the MessageRegister class object, with a notion that the Endpoint defines a single register used by all connections, while the Clients assign separate storage for each connection. The process of ASTMessage definition can be seen as creating of a bond between the local and remote variables on two devices. In case of an ASTMessage containing an ASTVariables representing the function, the bound is made between the function on one device and variables used as its arguments on the other. Such messages containing references to a function can be transferred only in one direction, as arguments can only be written to functions. A graphical scheme showing an example of this binding is given in Fig. 3.

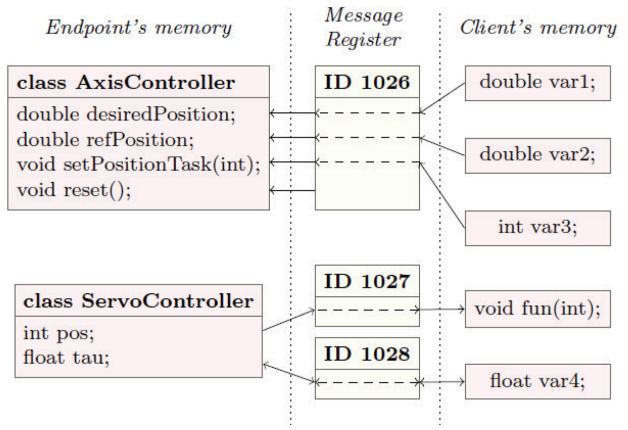


Fig. 3. Example of variable binding in ASTCom. Note that function reset() is called without any argument and the remote variable int pos is used to invoke a local callback function. Variables float tau and float var4 can be transferred in both directions

Rys. 3. Przykład łączenia zmiennych w ASTCom. Zauważyć można, że funkcja reset() wywoływana jest bez argumentów, a zdalna zmienna int pos wykorzystywana jest do wywołania funkcji lokalnej. Zmienne float tau oraz float var4 mogą być przesyłane w obu kierunkach

Two modes of data exchange are supported by the ASTCom library – the private channel communication used to transfer data between two devices and the stream channel used by the Endpoint to broadcast large amounts of data to all connected clients. All of the aforementioned ASTCommands are also sent through the private channel. Importantly, the ASTCom library does not specify a precise transport layer of the channels, and thus the communication can be carried out using several medias, including Websocket, TCP/IP or SPI communication with both channels supported by a single connection or separated between two independent routes, e.g. two separate Websocket connections. The Client can request to exchange the data through one of these communication channels. In case of the private channel, the Client requests a single exchange of data of a chosen ASTMessage in the desired direction – the data can be both read from or written to the telescope. Once the command is carried out, the values of the bound variables are identical on both devices. If the considered ASTMessage contains ASTVariables representing the function, it is remotely invoked, which can be used to control the behavior of the telescope controller.

While the private channel is designed mainly to control the telescope by the user or the supervisor, the stream channel is intended for constant acquisition of information about the state of the adevice. To this end, the Client may request the Endpoint to start recording chosen ASTMessages with a desired frequency. Currently, on the considered setup of the telescope controller, the recording with a frequency up to 10 kHz is possible. The Endpoint executes such a command, by cyclically serializing the value of requested variables into a local predefined buffer. Once a certain volume of data is acquired, the data is flushed and sent to all connected Clients, which receive the packets containing the amassed record of state of the telescope in the previous time instants. Importantly, Clients may bind the received data to some custom callback functions and this way process each sample of data separately upon arrival. Thus, a significant amount of data can be exchanged between devices to allow constant monitoring of the telescope performance.

The proposed approach was first implemented in C++ code, as this is the language of the main controller of the telescope. To enable support of various client applications, the Client class with all necessary dependencies was later ported into JavaScript (using LLVM/Clang-based Emscripten compiler), pure C and C# (using p/invoke feature).

4. Automated Control System

In order to enable the user to interact with the telescope and perform standard astronomical calculations, the automated control VirtualStar (VS) system was developed. It is responsible for real-time data acquisition and processing (e.g. axes states are continuously monitored) and, most importantly, it is a complex tool for calculation of desired telescope trajectories based on the movements of celestial objects. All these computations are possible with the use of commonly known astronomical libraries (SOFA, SGP4) as reported by Vallado et al. [14]; Hohenkerk [5]. Moreover, this software is responsible for management of the graphical presentation layer and processing of user requests. In order to ensure full system scalability and security in data storage, the VirtualStar system design consists of two separate subsystems (shown in Fig. 2 as the VSS and VSC modules). C# and .Net Core 3.1 tools were chosen to implement all subsystem components, including open source astronomical libraries – nuget packages written as wrappers for C++ source tools. The server-based subsystem part has been built specifically for the linuxarm distribution, while the client part is only available as the Windows application.

4.1. VirtualStar Server

The basis of the VirtualStar Server (VSS) structure is the ASTCom Client module which enables the exchange of the current state of any signal processed by the STM32 controller. Directly at startup, the ASTMessages are defined according to the rules specified by the ASTCom modules. A custom JSON-based configuration file contains a set of variable and function names paired with specific ASTCom library signals, which are used to properly configure the connection. Then, after a successful connection, each signal is transferred to the graphical presentation layer or sent to other subroutines within the VSS module, such as the kinematic calibration module (known as pointing model), for feature control tasks of the telescope system. The computation and data preparation are completely independent of the telescope mechanics, making it possible, for example, to continuously provide trajectory samples while tracking satellites without waiting to reach a previous point. Connected to the ASTCom Endpoint, VSS takes advantage of the private communication channel to query the Endpoint device and request it to perform the following commands: initialize/start/reset trajectory, getting a free buffer space, rotation around the horizontal axis (control of the altitude angle), rotation around the vertical axis (control of the azimuth angle), and sending next trajectory positions. In addition, using this mode, it also reads signals considered to be the general state of the device: telescope positions from encoders (instrumental altitude and azimuth angles), operating state, status and potential errors of each axis, and current time measured in microseconds. Meanwhile, the stream channel communication is used by the VSS to receive continuous tracking errors data with a specified frequency. Then, these position errors are stored in internal data arrays until they are forwarded to the VirtualStar client's presentation layer, described briefly below. It should be noted that both the mount driver and ASTCom library process requests synchronously, while the VirtualStar system handles all commands fully asynchronously. This means that at the VSS level it was necessary to include a synchronization module, thanks to which overloads and communication errors were avoided. It is a simple mutex-based implementation that always passes only one thread to the requesting function and holds all others until it is released.

4.2. VirtualStar Client

The user interface of the VirtualStar is designed as a standalone desktop VirtualStar Client application based on the Windows Presentation Foundation (WPF) framework and .Net Core 3.1. Its basic structure consists of the background communication layer and the graphical user interface with data presentation features. The exchange of data between the VSS and the VSC is performed on a separated communication layer using a Websocket protocol. All high-level application commands inputted by the user through the graphical layer are immediately converted to messages in JSON format and sent to the VSS, where they are serialized to particular commands recognized by the mount driver. Once a command is sent to the VSS, the VSC waits for the response stating that the command was correctly classified, executed, and completed successfully. Furthermore, the communication with the VSS is kept active by using a watchdog, which informs about any possible gaps in sending requests even though the Websocket state is still open. By applying a continuous time synchronization with the mount driver, both STM32 controller and VirtualStar subsystems operate in the same clearly defined time domain. Thus, all discrete points approximating the desired trajectory are fully synchronized and the motion actions are performed with a valid coordinates conversion time.

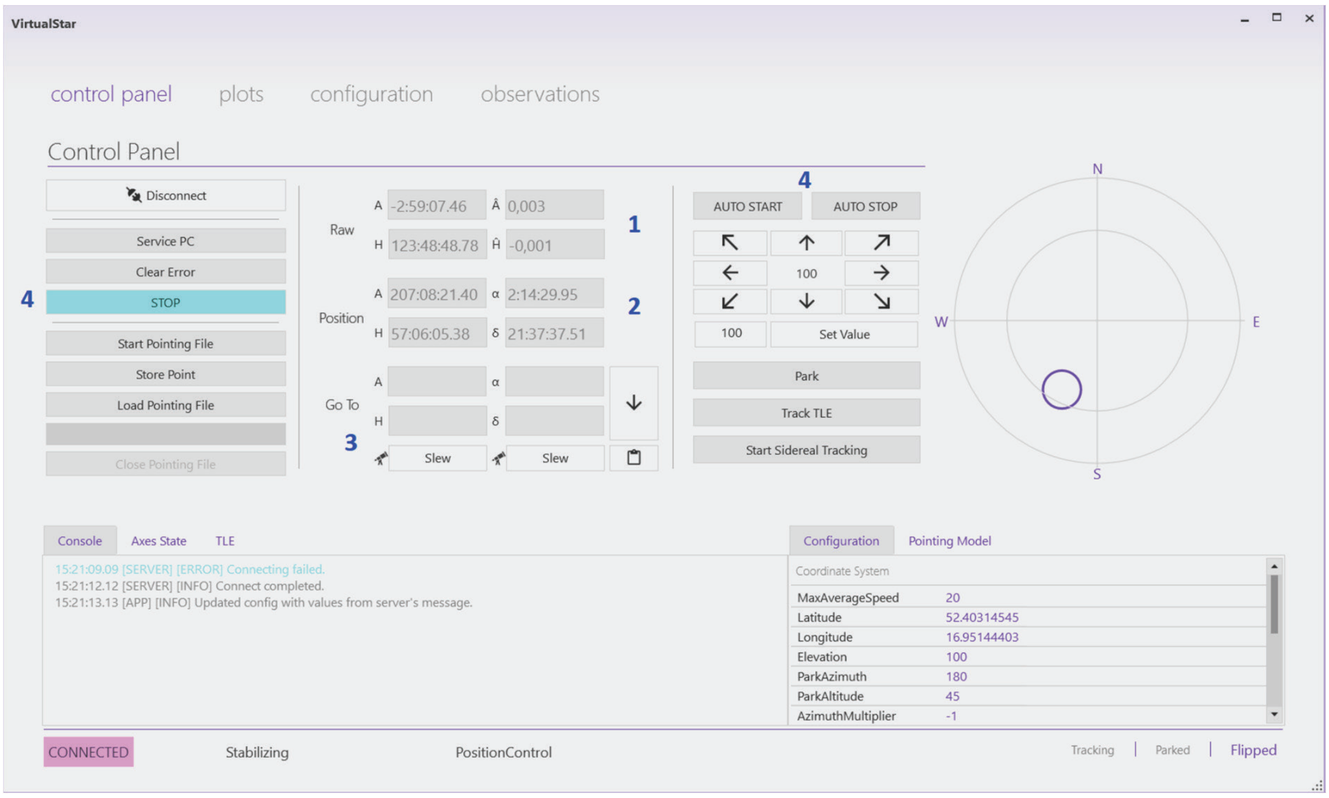
The graphical user interface layer (Fig. 4) shows the current coordinates as instrumental positions in the horizontal system (based on encoders values) (1), as well as current astronomical coordinates, both in the horizontal and equatorial systems (2). The last item in the vertical stack is the field for the input of the slewing targets (3), defined in any coordinate system. Other functionalities of the system (4) include: switching to a passive or active state, starting or stopping sidereal tracking, sending TLE orbital elements data for satellite tracking, monitoring operations of the entire system and reacting to possible errors and faults, preparing a file for the calculation of pointing model coefficients, and generating an automatic observations program.

5. Web Based Human Interface

The web application called the Robot Supervisor Panel (RSP) was created to facilitate the development and testing of the telescope system and to enable the designer to interact with the system in a simple and easy way. The main focus of the RSP is given to diagnostics, parameter tuning, online visualization with signal processing, and the possibility of conducting experiments for the telescope development and maintenance. The application is written using standard web development tools, including Hyper-Text Markup Language (HTML), Cascading Style Sheets (CSS) with Leaner Style Sheets (LESS) and Java Script (JS). Additionally, the JS front-end framework Vue.js is used for simplification of the design process and maintenance of the code. The most significant advantage of this framework is the reactivity feature. Namely, after any change of a reactive state, the interface structure is updated automatically. Moreover, this framework supports the modular structure of the application, which enables to dynamically modify the page depending on data received from the telescope system and the user's interaction.

5.1. Communication with the telescope system

The main target of RSP is an exchange of information with the telescope system by taking advantage of the ASTCom communication library. The RSP interface is divided into several panels, each responsible for a different functionality. The first panel, called Board, can be used to initiate a connection with the telescope system. Once the connection is successfully established, the application queries the telescope controller



Raw	A -2:59:07.46	Â 0,003	1
	H 123:48:48.78	Ĥ -0,001	
Position	A 207:08:21.40	α 2:14:29.95	2
	H 57:06:05.38	δ 21:37:37.51	
Go To	A	α	
	H	δ	

```
15:21:09.09 [SERVER] [ERROR] Connecting failed.
15:21:12.12 [SERVER] [INFO] Connect completed.
15:21:13.13 [APP] [INFO] Updated config with values from server's message.
```

Coordinate System

MaxAverageSpeed	20
Latitude	52.40314545
Longitude	16.95144403
Elevation	100
ParkAzimuth	180
ParkAltitude	45
AzimuthMultiplier	-1

Fig. 4. VirtualStar graphical user interface (a color inversion is applied to the image for readability)
Rys. 4. Interfejs graficzny aplikacji VirtualStar (dla zwiększenia czytelności zastosowano inwersję kolorów na ilustracji)

for a list of all available variables which are then separated into three types: streaming signals, parameters, and executable functions. An internal structures corresponding to these remote variables are then dynamically created to be used in further processing. These dynamically defined variables are used to exchange data with the telescope controller. It should be noted that for writing, reading, and streaming requests, the RSP takes advantage of a private channel to communicate with the Endpoint, and streaming channel to receive streaming data of requested signals.

5.2. Monitoring

The Board tab also shows all basic information about the device, including connection status, telescope status, and possible errors. The designed layout allows the user to keep an eye on the device and connection status without much effort. A full overview of all statuses and errors for each module of the telescope system can be found on the Diagnostics tab. In this tab, the user is able to check states of low-level controller, motors or a trajectory status. The card Signals presents remote variables available for streaming and is responsible for defining commands for a stream of chosen signals with a frequency up to 10 kHz that is equal to the sampling frequency of the main controller. The Plot card, as the name suggests, is used to visualize received data of streamed variables. Moreover, an additional tab called Compute is provided, which enables the user to perform algebraic calculations using streamer signals, the results of which can be later displayed in Plot tab. Significantly, each card offers a possibility to save its current state to a local configuration file, which can be later loaded to restore all settings with just a one click, e.g. if there is a need to recreate a past experiment.

5.3. Adjusting the control system

Using the Parameters card, one can easily read and modify the parameters values of the telescope system to perform the controller tuning. During the tuning process there is no need to restart the system to apply new settings and this process can be done online, while the telescope executes the ordered task. The online browser of signals received from the controller helps to manually tune up parameters to obtain the required performance of the control system for various conditions.

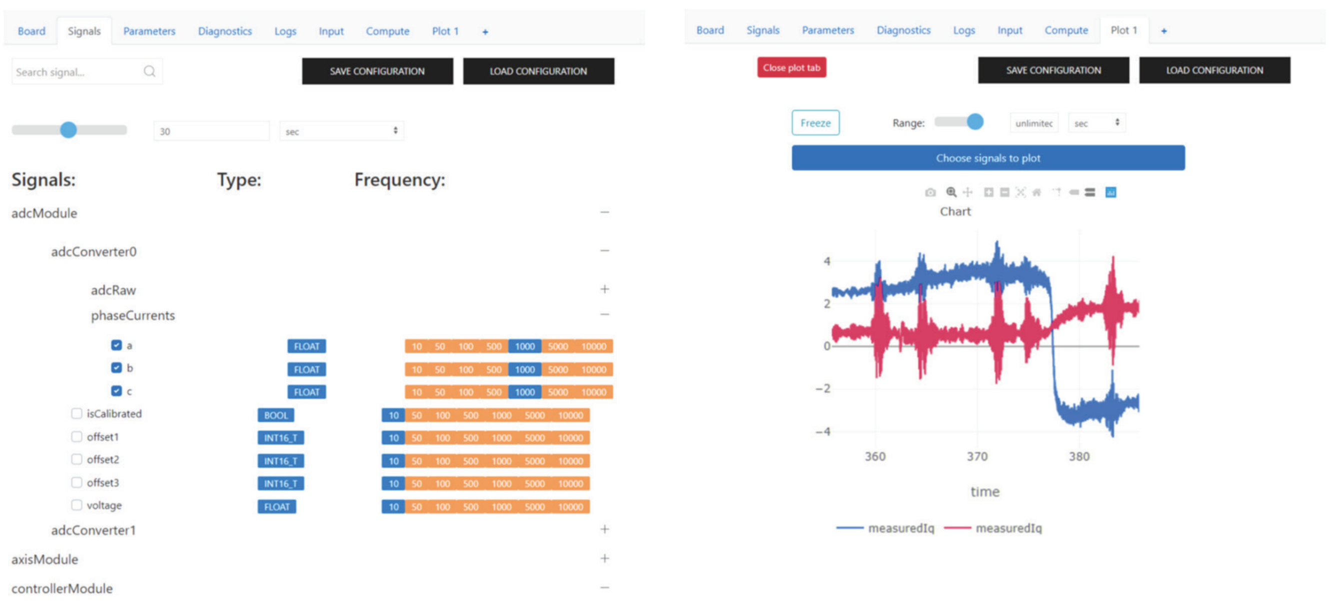
5.4. Control and experiments

The Input card enables to send predefined control commands like start, stop, or reset, and custom commands to remotely execute functions with given arguments. From this panel the user is also able to define a trajectory for each axis by giving a mathematical formula dependent on time and additionally specifying experiment duration. This function of the RSP has been used for experimental validation in Section 6.

6. Experimental Validation

In order to present the practical possibilities of the proposed software framework, the experiment was performed using the real telescope system operating in SkyLab. The experiment was conducted for a whole-night tracking of drifting sinusoidal trajectory specified for the single axis of the telescope with an average speed of approximately 0.0095 rad/min that resulted in an axis covering the distance of 6.8 rad in 12 h. Data acquisition was carried out throughout the experiment with a frequency of 1 kHz. The change of axis position from the start of experiment, motor current, and position tracking error were chosen for recording. Notably, all data was gathered directly from the drive controllers and visualizes the performance of the mount itself irrespectively from properties (e.g. resolving power) of the optical part of the telescope. Due to the simple character of the desired trajectory, the RSP module was used for its generation. Alternatively, VS application could be employed to perform the experiment with a trajectory corresponding to movement of some celestial objects. The database was used for online storage of acquired measurements. Significantly, the experiment was conducted remotely and no kind of human supervision was required throughout the experiment. The results of the experiment are given in Fig. 6.

In Fig. 7 a zoomed view of only the first two seconds of the experiment with full resolution of 1000 samples per second is given. Notably, both these figures were created using the same data from the same experiment. This exposes the possibilities of the proposed solution to monitor the performance of the telescope in different time scales while preserving high time accuracy even for long acquisition periods. In particular, one can distinguish different dynamic effects visible in Fig. 6 and 7



Signals:	Type:	Frequency:
adcModule		
adcConverter0		
adcRaw		
phaseCurrents		
☑ a	FLOAT	10 50 100 500 [✓ 1000] 5000 10000
☑ b	FLOAT	10 50 100 500 [✓ 1000] 5000 10000
☑ c	FLOAT	10 50 100 500 [✓ 1000] 5000 10000
☐ isCalibrated	BOOL	[✓ 10] 50 100 500 1000 5000 10000
☐ offset1	INT16_T	[✓ 10] 50 100 500 1000 5000 10000
☐ offset2	INT16_T	[✓ 10] 50 100 500 1000 5000 10000
☐ offset3	INT16_T	[✓ 10] 50 100 500 1000 5000 10000
☐ voltage	FLOAT	[✓ 10] 50 100 500 1000 5000 10000
adcConverter1		
axisModule		
controllerModule		

Fig. 5. Example views of different tabs of Robot Supervisor Panel web application
Rys. 5. Przykładowe widoki w różnych zakładkach przeglądarkowej aplikacji Robot Supervisor Panel

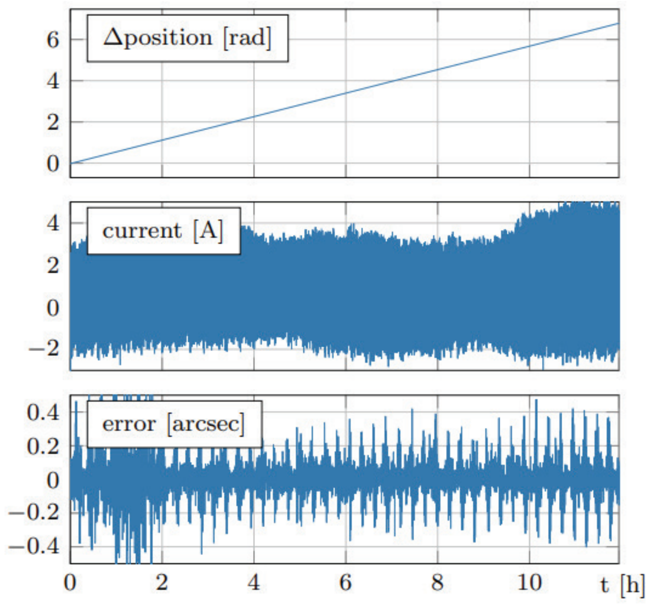


Fig. 6. Data obtained during the whole experiment
Rys. 6. Dane uzyskane w trakcie całego eksperymentu

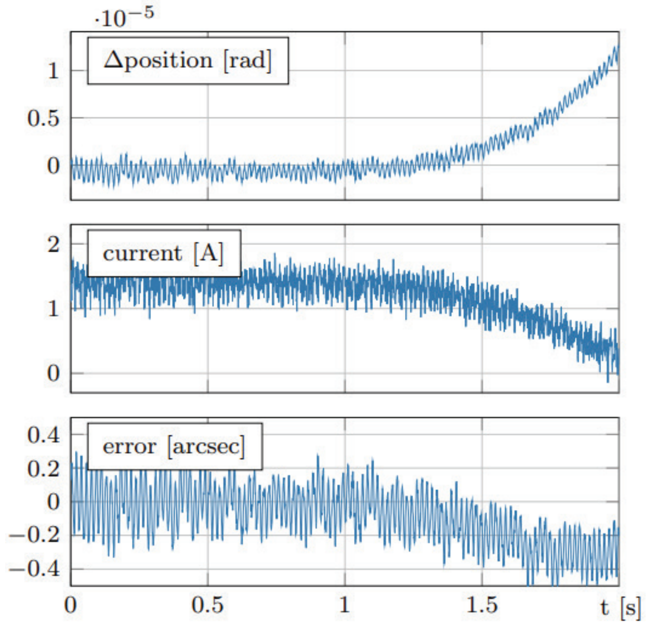


Fig. 7. Detailed view of data obtained in the first 2 s of the experiment
Rys. 7. Szczegółowy widok danych uzyskanych w pierwszych 2 s trwania eksperymentu

– vibrations visible in the wider time horizon are presumably caused by the presence of cogging torque due to the electromechanical structure of the motors, while the tracking errors observed in the short time scale are attributed to the vibrations of the mechanical mount which is not perfectly stiff. The data obtained from the whole experiment takes approximately 1 GB of memory when stored in optimized MATLAB data format and over 4 GB when written into text based .csv file.

The gathered data can be used not only for experimental purposes, but also to supervise the performance of the telescope system in order to detect failures and react as quickly as possible to any reduction in control quality, which may otherwise lead to loss of functionality of the system. In the worst case, it can help diagnose a broken element thanks to the recorded historical data. This subject was discussed in Katal et al. [7], where difficulties and possible techniques to deal with data analyze problem were presented.

7. Summary

The design of a control architecture for autonomous telescope mounts can be regarded as an interesting and challenging task due to a high complexity of the system, difficulties in achieving high performance of motion control and a reliable operation of the system over a long period of time in various environmental conditions. The tools described in the paper have been created taking into account these requirements. The implemented communication library ensures that adding new signals to the growing code of the controller will not take much effort, and no significant changes on the client side will be necessary. The designed web application allows monitoring, tuning the telescope control system and conducting experiments from one place that can be convenient for system designers and engineering scientists. This diagnostic tool can also be used during the maintenance to keep performance on the highest level and prevent failures of the system. Alternatively, the VirtualStar system enables astronomers to efficiently carry out desired observations. The database provides access to measurements gathered during observations to later conduct an analysis of the acquired data.

The future work may include extension of functionality of the presented client applications, including implementation of log viewer and adding additional conversion and calculation functions in the web application. Moreover, reliability and security of the considered communication scheme should be increased. Specifically, introduction of some security measures concerning prevention of unauthorized access to the communication network may be considered as a crucial issue for the employment of the telescope mounts working under the proposed framework. Additionally, there is still a space for the development of the error detection process to achieve better effectiveness and reliability of the telescope control system, including analysis of collected data.

Acknowledgments

We would like to express our thanks to our late leader, Professor Krzysztof Kozłowski who was the initiator of series of projects focused on the development of technologies for autonomous optical observation systems implemented at the Poznan University of Technology.We would also like to thank Bartłomiej Krysiak, PhD, and Mateusz Ochocki, MSc, who are the designers of the mechanical structure of the telescopic mount used in the SkyLab, and Tomasz Jedwabny, MSc, who designed and integrated the electronic units of the system, and Stanisław Kozłowski, PhD, for his support in the field of astronomy and SST, and active participation in the functional development of the SkyLab laboratory. We also thank Piotr Mieszała, MSc, for his technical and administrative support of the projects.

This research was funded by the Poznan University of Technology grant number 0211/SBAD/0121.

References

1. Abareshi B., Marshall R., Gott S., Sprayberry D., Cantarutti R., Joyce D., Williams D., Probst R., Reetz K., Paat A., et al., *A new telescope control software for the Mayall 4-meter telescope*, „Software and Cyberinfrastructure for Astronomy IV", Vol. 9913, 2016, 645–656. SPIE, DOI: 10.1117/12.2233087.
2. Cochran R., Marinescu C., Riesch C., *Synchronizing the Llinux system time to a PTP hardware clock*. IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, 2011, 87–92, DOI: 10.1109/ISPCS.2011.6070158.

3. Edwards P., Amy S., Brodrick D., Carretti E., Hoyle S., Indermuehle B., McConnell D., Mader S., Mirtschin P., Preisig B., et al., *Remote access and operation of telescopes by the scientific users*. „Observatory Operations: Strategies, Processes, and Systems V", Vol. 9149, 2014, International Society for Optics and Photonics, DOI: 10.1117/12.2058794.
4. Gawron T., Kozłowski K., *Semi-automated synthesis of control system software through graph search*, „Advanced, Contemporary Control", 2020, 1092–1103, Springer, DOI: 10.1007/978-3-030-50936-1_91.
5. Hohenkerk C.Y., *SOFA and the algorithms for transformations between scales & between systems*. [In:] H. Schuh, S. Boehm, T. Nilsson, N. Capitaine (eds.), „Proc. of Journées Systémes de Référence Spatiotemporels 2011", 2012, 21–24.
6. Ivanescu L., Baibakov K., O'Neill N.T., Blanchet J.P., Blanchard Y., Saha A., Rietze M., Schulz K.H., *Challenges in operating an Arctic telescope*. „Ground-based and Airborne Telescopes V", Vol. 9145, 2014, 1489–1509, SPIE, DOI: 10.1117/12.2071000.
7. Katal A., Wazid M., Goudar R.H., *Big data: Issues, challenges, tools and Good practices*. [In:] 2013 Sixth International Conference on Contemporary Computing (IC3), 2013, 404–409, DOI: 10.1109/IC3.2013.6612229.
8. Kozlowski K., Pazderski D., Krysiak B., Jedwabny T., Piasek J., Kozlowski S., Brock S., Janiszewski D., Nowopolski K., *High precision automated astronomical mount*. [In:] Conference on Automation, 2019, 299–315, Springer, DOI: 10.1007/978-3-030-13273-6_29.
9. Kozłowski S., Pazderski D., Krysiak B., Patelski R., Jedwabny T., Kozłowski K., Sybis M., *SkyLab: Research and development facility for ground based observations*. „Ground-based and Airborne Telescopes VIII", Vol. 11445, 2020, 1399–1408, SPIE, DOI: 10.1117/12.2576332.
10. Krysiak B., Pazderski D., Kozłowski S., Kozłowski K., *High efficiency direct-drive mount for space surveillance and NEO applications*. „Publications of the Astronomical Society of the Pacific", Vol. 132, No. 1015, 2020, DOI: 10.1088/1538-3873/ab9cc5.
11. Marchiori G., Formentin F., Rampini F., *Reliability-centered maintenance for ground-based large optical telescopes and radio antenna arrays*. „Groundbased and Airborne Telescopes V", Vol. 9145, 2014, 1346–1354, SPIE, DOI: 10.1117/12.2057593.
12. Sadeh I., Dezman D., Oya I., Pietriga E., Schwarz J., *The Graphical User Interface of the Operator of the Cherenkov Telescope Array*. [In:] Proc. of International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS'17), Barcelona, Spain, 8-13 October 2017, 186–191. JACoW, 2018, DOI: 10.18429/JACoW-ICALEPCS2017-TUBPL06.
13. Story K., Leitch E., Ade P., Aird K., Austermann J., Beall J., Becker D., Bender A., Benson B., Bleem L., et al., *South Pole Telescope software systems: control, monitoring, and data acquisition*. „Software and Cyberinfrastructure for Astronomy II", Vol. 8451, 2012, International Society for Optics and Photonics, DOI: 10.1117/12.925808.
14. Vallado D.A., Crawford P.S., Hujsak R., Kelso T.S., *Revisiting Spacetrack Report #3*. [In:] AIAA Astrodynamics Specialist Conference, 2006.

Architektura Systemu do zadań Rozwoju i Nadzoru Zrobotyzowanego Teleskopu Astronomicznego

Streszczenie: Artykuł przedstawia nowy system sterowania i komunikacji zaprojektowany w celu usprawnienia rozwoju i utrzymania zrobotyzowanego montażu teleskopu astronomicznego. Proponowane rozwiązanie umożliwia użytkownikowi zdalny dostęp do dowolnych sygnałów wewnątrz sterownika bez zwiększonego obciążenia podczas pracy systemu. Zaimplementowane rozwiązanie może być wykorzystywane zarówno przez automatyczny system nadzorujący, jak i przez użytkownika lub operatora, do nadzoru, sterowania i utrzymania urządzenia.

Słowa kluczowe: sterowanie w systemach wbudowanych, zdalne sterowanie, zdalne utrzymanie, kontrola warunków pracy, wykrywanie błędów i diagnoza

Patryk Bartkowiak, MSc, Eng
patryk.bartkowiak@put.poznan.pl
ORCID: 0000-0002-1675-4259

He is a doctoral student of Poznan University of Technologu (PUT) Doctoral School. He graduated from the PUT with B.E. degree obtained in 2018 and M.S. degree gained after year in area of Control and Robotics. In September 2019, he started work on a project aimed at designing, building a telescope and developing software to track objects in the sky for this device. His area of interests focus mainly on programming, control systems, control algorithms analysis and hardware implementation. The main area of his research are underactuated systems, linearization methods and robust methods of state estimation.

Radosław Patelski, MSc, Eng
radoslaw.patelski@put.poznan.pl
ORCID: 0000-0002-7301-5436

He is the Teaching/Reasearch Assistant in the Institute of Automatic Control and Robotics at the Faculty of Control, Robotics and Electrical Engineering of the Poznan University of Technology. He is also a PhD candidate in the discipline of Automation, Electronic and Electrical Engineering. His scientific interests include the state estimation, disturbance rejection and adaptive control, with special attention given to the stability properties of these systems.

Marta Kwiatkowska, MSc, Eng
marta.kwiatkowska@put.poznan.pl
ORCID: 0000-0003-4260-9909

She is an engineer in the Institude of Automatic Control and Robotics at the Poznan Unversity of Technology in the field of computer science and software engineering, including development of the system for supervising operations of an astronomical telescope. In 2019, she participated in research related to improvement of pointing model for a telescope 0.5m, as part of her M.S. thesis in area of Automatic Control and Robotics. Her scientific interest focus mainly on software development, testing and data science in the space industry.

Dariusz Pazderski, PhD, DSc
dariusz.pazderski@put.poznan.pl
ORCID: 0000-0002-8732-7350

He received his PhD degree in 2007 from the Faculty of Computing Science and Management of the Poznan University of Technology (PUT). In 2017, he obtained habilitation degree in control theory in robotics. Since 2020 he is the Head of the Institute of Automatic Control and Robotics at the Faculty of Control, Robotics and Electrical Engineering (PUT). His research interests include continuous and discontinuous nonlinear control, robust and adaptive control, state estimation techniques, motion planning, sensor systems and various applications of robotic technology including autonomous observatory systems. The results of his research have been published in many national and international conferences and journal papers.