electronics

MDPI

Article

Remote Laboratory Offered as Hardware-as-a-Service Infrastructure

Wojciech Domski

Department of Cybernetics and Robotics, Faculty of Electronics, Wrocław University of Science and Technology, ul. Janiszewskiego 11/17, 50-372 Wrocław, Poland; wojciech.domski@pwr.edu.pl

Abstract: This paper presents a solution for remote classes where hardware is offered as a service. The infrastructure was based on Raspberry Pi mini computers to which a set of different developments boards were connected. The proposed software architecture allows students to connect to remote resources and interact with them. Moreover, the services monitoring status of remote resources were introduced to facilitate software development and the learning process. Furthermore, live video feedback is available to visually monitor operation of the resources. Finally, a debugging server was deployed allowing us to establish a remote debugging session between a user's PC and the dev board on the server premises. The solution offers a comprehensive remote service including user management. Safety risks of the Internet-exposed infrastructure and safety precautions were discussed. The presented RemoteLab system allows students of WUST to gain knowledge, practise and realize exercises in scope of academic courses such as robot controllers and advanced robot control. Thanks to advances in remote education and utilized tools, the RemoteLab was designed and deployed, allowing stationary classes to be substituted with remote ones, while maintaining a high level of class knowledge transfer. Up to the present, the system has been utilized by over 100 students who could realize exercises and prepare for classes thanks to 24 h system availability.

Keywords: remote laboratory; remote education; hardware as a service; STM32; Raspberry Pi; micro services; user management; security; open source

Citation: Domski, W. Remote Laboratory Offered as Hardware-as-a-Service Infrastructure. *Electronics* **2022**, *11*, 1568. https://doi.org/10.3390/electronics11101568

Academic Editors: Yoshiyasu Takefuji, Subhas Mukhopadhyay and Enrico Vezzetti

Received: 29 March 2022
Accepted: 12 May 2022
Published: 13 May 2022

1. Introduction

Nowadays, remote education is gaining more and more popularity due to its numerous advantages. One such advantage is 24 h availability [1]. Thanks to this, students can access remote laboratories any time and from any place. Furthermore, the worldwide situation caused by the COVID-19 pandemic had an impact on modern education. As a result, universities from all over the world have had to adjust. This caused rapid development of different kinds of educational aids. Computer-science-related courses were particularly susceptible to the change due to their character [2–4]. For example, online programming classes are virtually the same as the stationary counterpart. Students usually write a piece of software which is a part of an assignment and then their work is assessed. Furthermore, additional aids are already present which allow faculty to assess the assignments more efficiently thanks to tools such as version control systems, e.g., GIT and CI/CD pipelines. This allows staff to evaluate work quicker and at the same time be impartial to the final result. Only in some cases is direct involvement from a teacher required. On the other hand, classes which involve hardware interaction of any kind are not trivial to move to a remote domain. Remote laboratories that are based on hardware can be compared to an industrial trend called "Hardware-as-a-Service". This specific approach offers access to hardware in the scope of remote interaction. Thanks to this, through remote connection, hardware can be accessed to perform some measurements and even trigger an action.

The aforementioned trend was adopted in academia also, where multiple remote laboratories were established to aid students in education. Classes teaching about industrial

systems, actuators, PLCs, etc., were also successfully digitized. Students can work with a variety of equipments, sensors, and actuators [5–7]. Through combinations of electronic components, a web camera, multimeters, and web servers, remote access to a laboratory for electronic engineering students was created [8]. Another branch of education deserves particular attention, which aims to deliver a remote environment concerning wide range of microcontrollers [9], programmable devices, such as FPGAs [10], and data collection and monitoring equipment [11,12]. Over the years, a variety of solutions has emerged which usually delivers a web-based system with user interface that facilitates interaction with hardware, and in particular with microcontrollers [13–15]. Some of the remote laboratories are not exclusively dedicated to microcontrollers, but offer help in the investigation of physical phenomenons [16] or give the ability to control a robotic arm [17].

The solution proposed in this paper is similar to those which are already available. However, certain conditions had to be met, including the adaptation of already used equipment and tools, thus STM32 microcontrollers were selected as the hardware base. The stationary classes are based on that solution. Moreover, using Raspberry Pi computers as the core of RemoteLab allowed us to decrease the physical space required to outline development boards and connect them together. The RemoteLab is depicted in Figure 1. The whole infrastructure is spread over a desk on which a number of development boards are located. The boards are connected to USB hubs which in turn are connected to Raspberry Pi computers. Over the table, a setup with Raspberry Pi cameras was installed to offer a view of the desk with the development boards.

Figure 1. RemoteLab laboratory based on Raspberry Pi computers with cameras and STM32 Nucleo boards.

Because of a remote connection it is of utmost importance to ensure a secure and an encrypted channel, the service of the remote laboratory is accessible from the global network and this results in variety of challenges [18,19]. It is important to be aware of different threads [20] which can impact the functionality of the service. Moreover, sensitive data transferred between a server and the end user needs an additional layer of protection. Even a separate branch of monitoring and analysis of edge data was developed where particular emphasis was placed on the collected data [21]. Within the RemoteLab initiative, no user-specific data are collected. Nevertheless, the communication channel has to be secured to deliver highest possible level of security.

The motivation behind the work presented in this article was to help students during the hard time of the COVID-19 pandemic. Many countries were impacted by the outbreak of the virus. Stationary classes were temporally cancelled in the hope of return to the day-to-day work. However, as it turned out, the current situation created a new reality. To help students to still gain knowledge and proficiency in scope of microcontrollers programming that is part of Robot Controllers and Advanced Robot Control classes, the RemoteLab initiative was introduced. First, it was intended to offer a substitute for stationary laboratory classes. Thanks to it, students could learn and practise. Soon, it was clear that the capabilities of RemoteLab were broader. Students were not only performing exercises but were enabled to practice for classes. Moreover, after receiving feedback, the infrastructure

was also used for personal projects related to academia, proving that the initial purpose was fully accomplished.

This article's goal is to offer a closer look at RemoteLab—a comprehensive remote laboratory for robot control classes held at Wrocław University of Science and Technology. The hardware components used to build the network of development boards and sensors are presented in Section 2. In turn, the software architecture of the RemoteLab is discussed in detail in Section 3. This part of the paper contains an outlook of the main system components as well describing the interactions between them. Safety and security aspects are discussed in Section 4. Since the RemoteLab is offered to WUST students as a server where they can log into and work, user management had to be incorporated. The details were presented in Section 5. Discussion of the achieved results, which were possible thanks to the RemoteLab infrastructure, and the introduction in scope of academic teaching are discussed in Section 6. Conclusions and directions of further evolution of the service are discussed in detail in Section 7.

2. Hardware Architecture

The architecture of the presented solution is depicted in Figure 2.

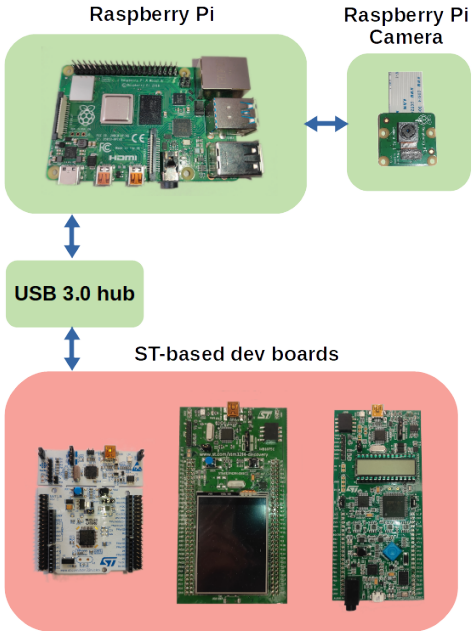


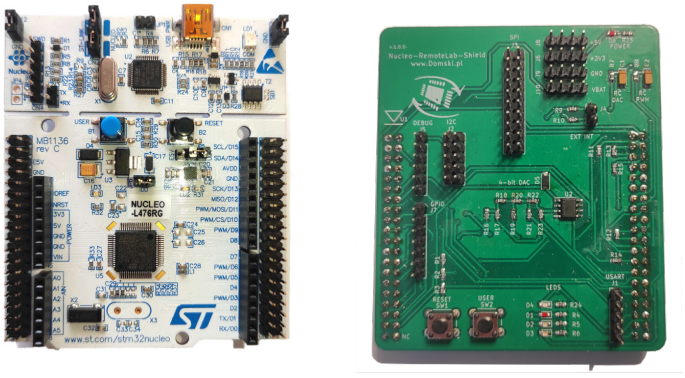
Figure 2. RemoteLab hardware architecture.

RemoteLab is based on Raspberry Pi, which is an SBC. Three units of this mini computer were used to deliver the service described in this article. The popularity of this mini computer is constantly increasing. Moreover, the community support is clear and, most of all, the Raspberry Pi ecosystem has a rich software library. For these reasons, it was selected as the host for RemoteLab.

The hardware requirements were partially dictated by the needs and curriculum of the robot control course. Since the class is organised around ST microcontrollers, it was required that the development boards should also be based on the same microcontrollers. Given the requirements, three types of test boards were selected for the laboratory.

The first set of development boards was *Nucleo-L476RG* (Figure 3a) that facilitates STM32 microcontroller—STM32L476RGT6 (ARM Cortex-M4, 1024 kB Flash and 128 kB RAM). This MCU is equipped with a selection of advanced timers, analogue-to-digital and digital-to-analogue converters, DMA, SPI, I2C, and many more. This evaluation board offers basic features in the scope of the present work with embedded systems. Additionally, it is equipped with an on-board ST-LINK debugger/programmer, accessible via a mini USB port. Availability of this component is crucial in terms of flashing the MCU located on

the *Nucleo-L476RG* board; thus, new firmware can be downloaded and tested. Moreover, this feature is commonly used when traditional flashing via a programmer is considered. However, when connected with a dedicated debugger server, remote capabilities can be achieved. In other words, the presence of the ST-LINK debugger on the board allows it to be programmed remotely while using additional software. Besides the aforementioned debugger/programmer, the *Nucleo-L476RG* board was equipped with a handful of external peripherals.



(**a**) (**b**)

Figure 3. *Nucleo-L476RG* development board with expansion shield. (**a**) Photograph of the *Nucleo-L476RG* development board; (**b**) shield for *Nucleo-L476RG*.

The user can utilize a LED, through which, the state of the board can be visualized. Furthermore, a push button is present, which can act as a trivial input interface. In addition to already mentioned components, pin headers are available, through which external sensors and actuators can be connected. The board is compatible with the so-called Arduino Uno shields due to the presence of carefully placed connectors. Thanks to this, the board can be extended with modules which are available on the market. As was already mentioned, the board has a built-in debugger/programmer. The presence of the ST-LINK offers a feature in the form of a virtual COM port (VCP) that acts as a bridge between the MCU and the PC, connected via the mini USB port.

Each *Nucleo-L476RG* development board is accompanied by an additional expansion shield. This shield can be seen in Figure 3b. It contains some internal connections between a series of selected microcontroller ports. All of the aforementioned connections are made with a 1 k resistor to prevent short circuits when the MCU is configured improperly. These connections can then be used to implement various functions. For example, GPIO is connected with a timer output. Thanks to this, with additional implementation, external interruptions can be generated with help of the timer output. This, in turn, allows one to simulate real-life external interrupts on the GPIO port. In addition to internal connections, four LEDs are available to the user. Moreover, a voltage divider is present to exercise ADC measurements. A 2R-D DAC, based on a series of resistors and voltage amplifier, is available. This allows students to familiarise themselves with 2R-R DAC operation principles. Finally, two RC filters were designed to simulate a DC motor operation. Thanks to this, students can practise implementation of a PID controller for a DC motor as well as acquire measurements to perform characteristics of these low-pass filters. The first RC filter is driven with a DAC output, while the second one is driven with a PWM output. The shield offers a number of connectors that can be utilized for external modules communicating through either the SPI or the I2C interface. More information about the expansion shield and external sensors is given in Section 2.

The second set of development boards is *32L476GDISCOVERY*, as shown in Figure 4. The main MCU of the board is STM32L476VGT6 (ARM Cortex-M4), equipped with 1024 kB of Flash and 128 kB RAM memory. This microcontroller offers a vast variety of internal peripherals, such as timers, analogue converters, communication interfaces (UARTs, SPIs,

I2Cs), and others. Similar to the *Nucleo-L476RG*, this piece of hardware has a built-in ST-LINK debugger/programmer which delivers the same functionality in the scope of RemoteLab. It allows one to flash new versions of firmware, debug it, and offers VCP for communication. The board also has a number of on-board sensors which offer a good testing environment for students. It was equipped with two LEDs for user disposal and a joystick. This joystick can generate inputs in four directions and can perform a confirmation event by being pressed down. A 24-segment LCD is available to act as a user output interface. The student can use SAI Audio DAC to generate sound and a speaker can be connected to an accompanying jack output. Audio signal can be acquired through a digital MEMS microphone. In addition, two other MEMS sensors—an accelerometer with a magnetometer and a gyroscope—can be used. This sensors can be used to, e.g., create AHRS to estimate position and orientation in the global frame. The gathered data can be stored on external Flash memory connected via Quad-SPI interface. Therefore, features offered by the board can be used to create a robotic system and made the board suitable for robot control classes.

Figure 4. Photograph of the *32L476GDISCOVERY* development board.

The last set of evaluation boards, remotely accessible via the RemoteLab infrastructure, is *STM32F429I-DISC1*, shown in Figure 5. The board contains an STM32F429ZIT6 microcontroller based on the ARM Cortex-M4 core, that offers 2 MB of Flash memory and 256 kB RAM. It offers similar set of built-in peripherals as the other two boards. However, in comparison with the previously described boards, this one offers richer user interface peripherals. It contains two user LEDs and one push button. In addition, a 2.4 inchTTF colour LCD screen with a touch panel can be found. Thanks to this, students can practise the development of graphical drivers and familiarize themselves with graphic techniques, such as double-frame buffer, commonly used for increasing performance. For better resource management, the user has an external SDRAM module which can acts as a graphic buffer and leads to the mentioned increase in performance. Similarly to *32L476GDISCOVERY*, the development board has a MEMS sensor which, in this case, is a gyroscope. This test board

allows for easy implementation of 2D and 3D algorithms which can be then visualized on a built-in display.

Figure 5. Photograph of the *STM32F429I-DISC1* development board.

Modules

In order to give students wide range of external hardware selection on which they can work, the RemoteLab was extended with external modules. Each of the *Nucleo-L476RG* evaluation boards was equipped with a shield board (Figure 3b), on which additional components were placed. Standard equipment includes LED diodes to display the binary state through GPIO ports. This allows students to become familiar with digital outputs by observing whether the LED emits light. Additionally, each board was equipped with RC filters. The RC filters act as DC motor simulators. The input of the filter is driven via DAC output or via PWM output. In turn, the output of the RC filter is connected to the microcontroller's ADC input channel, which allows the voltage to be measured. Thanks to these RC filters, students can practise the implementation of PID controllers. As mentioned earlier, the second RC filter is available but it is driven with a PWM signal. The output of this subsystem can be measured through a separate ADC input channel. Another component composed of a series of resistors connected to ADC inputs helps one's understanding of different ADC operation modes. Moreover, the board offers a 2R-R DAC. This digital-to-analogue converter has 4 bits of resolution, and the generated value can be set through 4 inputs connected to the MCU's GPIOs. The output of the 2R-R DAC is conditioned with a unit-gain amplifier and passes to an ADC input channel located in the microcontroller. Finally, when using a pair of development boards, a direct connection between them can be made. This connection is used to present how external interruptions are working. Additionally, a simple protocol based on a single wire, similar to the 1-Wire protocol, can be devised to enable communication between two boards. Finally, timer inputs and outputs in a form of PWM outputs and input capture (IC) inputs are connected with digital inputs in different configurations. Such connections allow one to gain comprehensive knowledge on the configuration of timer peripherals and learn the operation of those common interfaces. To sum up, through a series of connections supported with the presence of discrete elements, students can remotely access digital inputs and outputs, analogue inputs (ADC) and outputs (DAC), timer channels (PWM, IC), and external interruption inputs.

Moreover, besides the presence of the aforementioned peripherals, each board is equipped with different sensors capable of serial communication through a serial peripheral interface (SPI) and/or inter-integrated circuit (I2C) interfaces. In these modules, the following can be included: accelerometer, gyroscope, magnetometer, barometer, thermometer, and light intensity sensor. During the classes were RemoteLab is being used, communication with the sensors can be practised, and a selection of digital signal processing techniques can be implemented and executed based on life data provided in real-time by external sensors.

3. Software Architecture

The overall architecture for RemoteLab was built upon the component paradigm and the SaaS approach ("Software-as-a-Service") [22–24]. Thanks to this, the development and, most importantly, the maintenance are simplified due to the fact that each service is independent [25]. Four areas were identified (Figure 6), in which a dedicated service had to be created. All of the services are run on Raspberry Pi, which acts as a server. The first system is responsible for providing a connection between the end-user and the physical development board connected to the server (the debugger service). Then, monitoring of the health status of both the debugger and the boards was identified as a separate filed status service. To provide video feedback of the prototype boards, a video server was implemented through which students have a direct, almost real-time video feedback service. The last area, which required careful planning, was a user management service. Since multiple users can connect to the RemoteLab simultaneously, access management had to be introduced.

Each of described areas has been collected below, offering a more comprehensive understanding of the role each service plays in the discussed system.

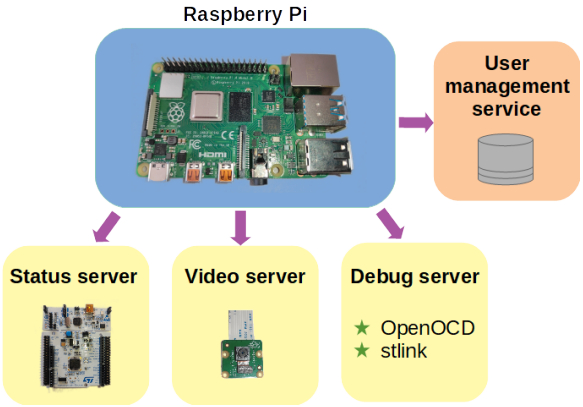


Figure 6. RemoteLab software architecture.

3.1. Debugger Server Service

Remote debugging is not a new concept. However, it allows one to distribute resources or even entire systems across a network while providing an interface through which they can be interacted with [26,27]. The debugger server is responsible for providing connection between the end-user and the development board. Each board is equipped with on-board debugger ST-Link. The hardware debugger is directly connected to the microcontroller (MCU). The ST-Link provides a USB interface through which it is connected to the server. This interface provides a communication channel as well as powering the board. At the server, there is a dedicated application called the debugger server, based on *OpenOCD* or *stlink* [28]. Since the software is required to run on Raspberry Pi, a software version suitable for this platform had to be installed. Due to differences in development board firmware, there was a need to use different debugger software (*OpenOCD* or *stlink*). This, in turn, allowed us to increase the reliability of the solution. The debugger server allows one to connect to the board via a built-in ST-Link. Moreover, the server provides a network

interface through which the development board can be controlled. At the user side, when a debug session is being created, one is required to select a remote debug server. Thanks to this, the new firmware can be uploaded to the development board MCU.

Working with the remote development board is virtually the same as working with a board which would be directly connected to the client's PC, allowing for flawless software development. Moreover, to maintain high level of security, the debug session is encrypted—SSH tunnelling was used.

3.2. Status Server

In addition to the debug server, the student is required to monitor the status of the board. The status server is capable of delivering such functionality. The service was based on Python and Flask frameworks, which can be used to build web applications. Through the web interface, a student can control the MCU status, in particular, the MCU can be halted, resumed, and restarted. Moreover, the debug server at the remote host can be restarted in cases where issues arise with the debugger itself. Additionally, the status of the board is displayed through the web interface. The issue of whether a board is connected to the remote host, and whether it is active or not, can be assessed. An example overview page is presented in Figure 7.

Server: aries

ID	Reset	Halt	Resume	Restart debugger service	Service type	Port	Status	Board	Serial	Features
1	Reset	Halt	Resume	Restart	openocd	3001	☺	Nucleo-L476RG	/dev/ttyACM1	DAC1, ADC1, TIMER3, ADC2, ADC3, ADC4, LED1, LED2, TIMER4, INT1-2, TIMER1, TIMER5, GPIO4, ADC5, LED_ON_BOARD, SPI_SCK, SPI_MISO, SPI_MOSI, SPI_CS1, SPI-CS1-ADXL345, I2C_SCL, I2C_SDA, I2C-MPU6050
2	Reset	Halt	Resume	Restart	openocd	3002	☺	Nucleo-L476RG	/dev/ttyACM4	DAC1, ADC1, TIMER3, ADC2, ADC3, ADC4, LED1, LED2, TIMER4, INT1-1, TIMER1, TIMER5, GPIO4, ADC5, LED_ON_BOARD, SPI_SCK, SPI_MISO, SPI_MOSI, SPI_CS1, SPI-CS1-ADXL345
3	Reset	Halt	Resume	Restart	openocd	3003	☺	Nucleo-L476RG	/dev/ttyACM6	DAC1, ADC1, TIMER3, ADC2, ADC3, ADC4, LED1, LED2, TIMER4, INT1-4, TIMER1, TIMER5, GPIO4, ADC5, LED_ON_BOARD, I2C_SCL, I2C_SDA, I2C-ADXL345
4	Reset	Halt	Resume	Restart	openocd	3004	☺	Nucleo-L476RG	/dev/ttyACM7	DAC1, ADC1, TIMER3, ADC2, ADC3, ADC4, LED1, LED2, TIMER4, INT1-3, TIMER1, TIMER5, GPIO4, ADC5, LED_ON_BOARD, SPI_SCK, SPI_MOSI, SPI_CS1, SPI-CS1-ADXL345
5	Reset	Halt	Resume	Restart	openocd	3005	☺	Nucleo-L476RG	/dev/ttyACM3	DAC1, ADC1, TIMER3, ADC2, ADC3, ADC4, LED1, LED2, TIMER4, INT1-6, TIMER1, TIMER5, GPIO4, ADC5, LED_ON_BOARD, SPI_SCK, SPI_MISO, SPI_MOSI, SPI_CS1, SPI-CS1-LPS25HB
6	Reset	Halt	Resume	Restart	openocd	3006	☺	Nucleo-L476RG	/dev/ttyACM5	DAC1, ADC1, TIMER3, ADC2, ADC3, ADC4, LED1, LED2, TIMER4, INT1-5, TIMER1, TIMER5, GPIO4, ADC5, LED_ON_BOARD, SPI_SCK, SPI_MOSI, SPI_CS1, SPI-CS1-LPS25HB
19	Reset	Halt	Resume	Restart	openocd	3019	☺	STM32L476-Discovery	/dev/ttyACM0	
20	Reset	Halt	Resume	Restart	openocd	3020	☺	STM32L476-Discovery	/dev/ttyACM2	

Figure 7. Status server view for Aries server.

Each development board is equipped with a serial interface for direct communication with the MCU. This is achieved via an on-board debugger which acts as a pass-through system, delivering a serial port at the remote host's side. The information about which serial port was assigned to the specific dev boards is displayed via the web interface.

As mentioned earlier, each test board has its own set of modules as well as some internal connections. These are called features. Through it, the student is able to identify whether, e.g., an ADC2 connection is available. Thanks to the notation used, in listing features, external modules connected to the dev board can be identified along with the kind of interface being used to communicate with them. For example, if inside features *I2C-MPU6050* entry is present, then the specific board was equipped with an MPU6050 IMU sensor connected via I2C. Details on which microcontroller pins were used can be identified from analysis of the expansion shield schematics.

3.3. Video Server

One of important features of RemoteLab is the ability to have a visual feedback on the test bench [29,30]. This ability is crucial during laboratory exercises. Thanks to this, a student can almost instantly retrieve feedback. For example, during the first steps of considering embedded programming, it is of the utmost importance to be able to observe

the effects of one's work. One of the first exercise involves LED blinking. Without video feedback, it would be impossible to observe how the flushed firmware is behaving on the development board. A sample frame from the video feedback is presented in Figure 8.

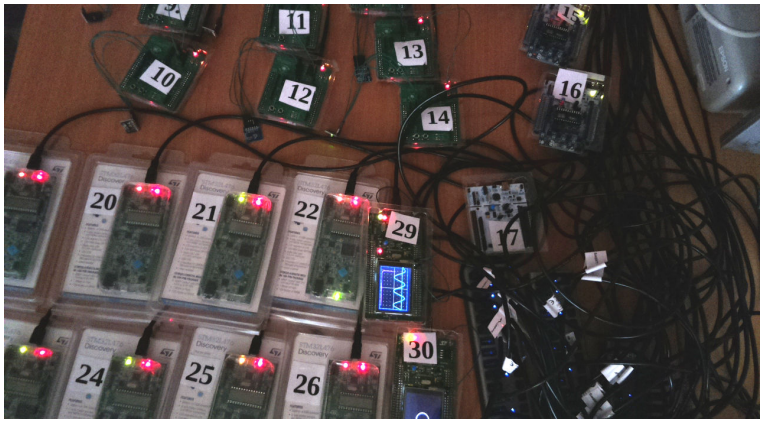


Figure 8. View from Raspi Cam connected to Eridanus server.

RemoteLab offers almost-real-time video feedback due to the fact of limited hardware resources of the server. Video streaming is a demanding task which requires both high bandwidth and computation resources [31]. RemoteLab is based on Raspberry Pi boards. Each SBC offers a dedicated interface for the camera sensor (MIPI CSI). Raspberry Pi is capable of delivering relatively high bandwidth but limited computation capabilities [32]. To remedy this inconvenience, a software solution was proposed. Instead of live video streaming to the students' computers, a picture is taken every few seconds. Along with the caching mechanism, this allows the delivery of high performance, while retaining significantly lower CPU workload when compared with native video streaming.

The performance factor is highly important since, after carrying out performance stress tests, it turned out that video streaming with a single recipient was a resource-intensive task. The proposed solution, which combines taking pictures instead of live-streamed video, and a caching mechanism, proved to be more efficient and capable of delivering video feedback for more then 10 students at the same time. During the conducted tests, it was determined that taking a picture of the work bench every two seconds was sufficient in terms of performing all exercised foreseen during RemoteLab classes.

3.4. Service Management

As mentioned at the beginning, each service is a separate component working independently. This increases the stability and maintainability of the proposed solution. In case of the debugger server service, third-party software was used which creates a separate session per board. It allows for port creation through which user debug session can communicate. Only a single connection is acceptable due to the fact that only one debug session can be connected to the debugger, and thus to a single development board. This, in turn, ensurers that only a single student can be connected to a single board. Another connection from a different debugging session is automatically rejected.

A custom solution was derived for status server and video server. Both servers were modelled as micro services using Python and Flask. Such a combination allowed us to introduces new changes to the system architecture and ensured that each service is independent.

4. Security

Nowadays, security is a subject of the utmost importance. The security of RemoteLab can be considered in two aspects: the first is the security of the RemoteLab servers themselves, while the second one is the security of an already established connection with the RemoteLab server.

In order to provide access to a server it has to allow for incoming connections. Moreover, the server has to be reachable from outside of the local network, and thus assigned a global IP address [33]. This leads to constant exposure of the server, and appropriate measures had to be introduced. To limit the possibility of an attack from outside, only a single port per server was opened and exposed to the outside network [34]. This allowed us to limit the number of attack angles and focus on securing a single channel. The SSH server listens to the exposed port. It offers a secure connection for each session; although the SSH server is secure, additional restrictions had to be introduced. The server is constantly monitoring incoming connections and filters logs for login attempts. In a case of multiple login attempts with failed authentication, the IP address of the incoming connection is blocked. This prevents so-called dictionary attacks, where the goal is to try different username and password combinations in the hope that a given combination of username and generated password is valid. Another measure for increasing security was to introduce a password policy. When a student wants to change the automatically generated password, they are allowed. However, due to the password policy, a newly set password has to be a strong one. This is a necessity for exposed servers.

Only a few mechanisms were presented to highlight the problem of internet connections and accessing remote servers. More prevention mechanisms have been introduced but it is not in the scope of this article to describe all of them. The second aspect of RemoteLab security is the security of an established communication channel itself [35]. This is important in terms of, e.g., determining where an attacker is listening to an ongoing communication in hope of intercepting a portion of information which could be exploited. Due to the fact that only SSH connection is being used, the communication channel is secure and encrypted, and thus the information transferred through the channel is secure [36].

5. User Management

To facilitate user management, a set of Python scripts was introduced. The scripts allow the management of OS users, allowing us to create, update, and remove student accounts. Due to different groups of students accessing the hardware, it is possible to assign users (students) to separate groups. This allows for better maintenance and management of user accounts.

The Raspberry-Pi-based server has limited resources, including storage space [37]. Because of this limitation, each user account has been assigned a disk quota. This allows for the management of storage space per user and per group. Furthermore, it is possible to set two kinds of limits. The first limit is called the soft limit. After exceeding this threshold, the user is notified that the assigned storage space is running out. The user is requested to free disk space to sustain normal operation. The second threshold is a hard limit. It is not possible to exceed this amount of assigned storage space. As a result, a user can no longer create new files or run tasks which require disk resources to operate. Furthermore, the quota does not affect the normal operation of the main purpose of RemoteLab. Despite exceeding quota limits, it is still possible to start debugging session, check the status server, or preview the live stream.

6. Results Discussion

RemoteLab offers remote access to the infrastructure of multiple development boards, allowing students to work with hardware as they would during stationary classes. Thanks to additional hardware—which was designed specifically for robot controllers classes—students can practise and realize exercises as they would during in-person laboratories. The presented infrastructure consists of 16 *Nucleo-L476RG* boards, 10 *STM32L476-Discovery* boards, and 2 *STM32F429I-DISC1* boards.

The ordinary workflow for robot controllers laboratories is as follows: students are assigned a single *Nucleo-L476RG* board, and they can work independently. Usually, laboratories facilitate 14 students at a time. This gives a buffer of a few spare boards. Occasionally a board is unresponsive, and then the student is assigned a different board during the

classes. This solution is the most efficient since it was discovered that the cause for this behaviour is sometimes the instability of the debugger software. However, restarting the service always resulted in the recovery of full functionality. The curriculum of robot controllers courses consists of exercises which gradually increase in complexity. First, classes are focused around connecting to the RemoteLab and using it efficiently. To familiarise students with the infrastructure, they are requested to successfully connect to the network, retrieve information from the status server, and obtain an image from the video service. This exercise is followed by a so-called "blinky project", that involves toggling the GPIO output connected to a LED in order to learn how this peripheral element works and how it can be used. In addition to this, students need to print out information about the action through the serial port. This allows them to familiarize with the remote laboratories' infrastructure. Subsequent classes involve the implementation of multiple projects that use timers, external interrupts, ADC and DAC converters, and DMA. The classes are concluded with exercises involving I2C and SPI communication interfaces, thus configuring and reading data from external modules. In comparison with the stationary classes, the exercises needed to be altered in such a way that some internal connections were already present and external modules were already present and connected. This, in turn, completely allowed us to the abandon requirement of physical access to the hardware. Thanks to this, it was possible to fully convert conventional classes to this remote counterpart.

On the other hand, robot controller classes projects involve usage of *STM32L476-Discovery* and *STM32F429I-DISC1* boards. Due to the fact that these boards are equipped with on-board external modules, such as sensors or LCD screens, they are suitable for project-type classes. During this type of academic activity, students prepare individual projects, during which they propose a subject which is then realized during the semester using provided remote resources. Examples of such projects are logging devices that periodically monitor a given physical quantity, e.g., linear acceleration. These data are then stored on Flash memory with a time stamp. Additionally, an alarm can be triggered when a certain threshold is exceeded. An example of a more advanced project is a 3D graphic driver. During the course of the semester, a group of students created a 3D graphic driver which is capable of creating 3D shapes and animations. A different group focussed on deploying techniques which increase the performance of the mentioned 3D graphic driver. These examples are provided to highlight the kinds of activities students are involved in, and the achievements which are possible thanks to RemoteLab.

Presently, there is one more academic course entitled "Advanced Robot Control". This course covers advanced aspects of robotics, both on a theoretical level, e.g., input–output decoupling methods [38], and practical aspects, such as using FreeRTOS in robotic applications. In this scope, students deploy firmware using the RemoteLab infrastructure to practise usage of FreeRTOS mechanisms. These mechanisms are then used to implement, e.g., a PID controller for a DC motor simulator, represented as a RC low-pass filter present on the expansion shield designed for *Nucleo-L476RG* board. Besides pure academic activities, the RemoteLab infrastructure was used by students to prepare their own projects, which involved usage of STM32 microcontrollers.

As mentioned earlier, sometimes the debugger service was unresponsive and it needed to be restarted to recover. Another issue emerged while RemoteLab was being used, which was connected to the network interface. Initially, each Raspberry-Pi-based server was accessed using a built-in WiFi adapter. This was dictated by the necessity to limit the number of cables used to create the network. It turned out that the wireless network connection was sometimes unstable, leading to a situation where the Raspberry Pi was not accessible. It required a hard reboot in order to restore full operation. To solve this issue, the Ethernet-wired connection was used and the wireless connection served as a backup. After this changed, no further incidents were reported related to network connection. It is worth mentioning that other incidents were reported but they cannot be classified as RemoteLab-related issues. During the first classes, it was noticed that some of the students were banned by servers. However, the root cause of this phenomenon was the human

factor. As mentioned earlier, to prevent brute-force attacks, a policy of three failed login attempts was introduced. After that, the IP address was banned for a certain period of time. In all reported cases, it turned out that the password was wrongly entered, which lead to a failed login attempt. As a solution, it was proposed to students who had that problem that they deploy a SSH key to the server in order to automate the authentication process.

7. Conclusions

In this article, both the concept and the realization of the remote laboratory was presented. The research presented in this paper was primarily focused on students and enabling them to use remote resources while not having the ability to carry out their learning on site.

The hardware architecture of the solution was presented, involving three independent servers running on Raspberry Pi mini computers. Each server was equipped with a number of development boards integrated with external modules to further widen the offered capabilities. Two of Raspberry-Pi-based servers were equipped with dedicated Raspberry Pi cameras. Thanks to this, a life-like video preview was possible, enabling students to observe the state of the tested boards.

The hardware infrastructure was accompanied with specialized software, enabling it to be a hardware-as-a-service infrastructure. Each server runs a status service, allowing the monitoring of the health and the state of boards. Moreover, if a server has a camera connected to its interface, then video streaming is also available. The last service from the client's perspective is the debug server, that allows a user to connect its debug session directly to the development board, which in turn is connected to the server itself. Moreover, a user management service is present. This maintains the client register, and ensures that resources are limited only to the use of authorised groups of users. Furthermore, this service allows us to maintain the remote management of the server in the scope of not only the users but keeping the system up to date.

Finally, safety and security aspects of transferred data were discussed. Since the infrastructure is accessible from outside the local network, some precautions had to be introduced. Restricting the access to network resources, storage, and connected devices is crucial. Exposing resources without previous precautions can lead to unnecessary risks which can lead to, e.g., data leaks or infrastructure takeover.

In the scope of hardware, modification is possible which involves different ways of interaction with the environment. In particular, an interface which could turn on a lamp or a speaker would be helpful to verify whether a light sensor or a microphone is working correctly. Furthermore, improvements in the server status would be beneficial, such as an online serial console, allowing users to directly communicate with a built-in serial interface via nothing more than a web browser.

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

I2C	inter-integrated circuit
LED	light-emitting diode
MEMS	microelectromechanical systems
MCU	microcontroller unit
MIPI CSI	mobile industry processor interface camera serial interface
PLC	programmable logic controller
PWM	pulse width modulation
SPI	serial peripheral interface
SBC	single board computer
WUST	Wrocław University of Science and Technology

References

1. Fernández-Pacheco, A.; Martin, S.; Castro, M. Implementation of an Arduino Remote Laboratory with Raspberry Pi. In Proceedings of the 2019 IEEE Global Engineering Education Conference (EDUCON), Dubai, United Arab Emirates, 8–11 April 2019; pp. 1415–1418.
2. Gomes, L.; Bogosyan, S. Current Trends in Remote Laboratories. *IEEE Trans. Ind. Electron.* **2009**, *56*, 4744–4756. [CrossRef]
3. Macho, A.; Baizán, P.; Blazquez, M.; García-Loro, F.; Sancristobal, E.; Díaz, G.; Gil, R.; Perez, C.; Castro, M. Work in progress: Proof of concept: Remote Laboratory Raspberry Pi + FPAA. In Proceedings of the 2019 IEEE World Conference on Engineering Education (EDUNINE), Lima, Peru, 17–20 March 2019; pp. 1–4.
4. Vavrenyuk, A.B.; Shishov-Turchin, D.B.; Alexeev, A.N.; Makarov, V.V. Multi-User System for Remote Access to the Resources of the Educational Computer Cluster Based on Single Board Diskless Computer Raspberry PI 3 Model B as a Service. In Proceedings of the 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), Moscow, Russia, 26–29 January 2021; pp. 731–734.
5. Letowski, B.; Lavayssière, C.; Larroque, B.; Schröder, M.; Luthon, F. A Fully Open Source Remote Laboratory for Practical Learning. *Electronics* **2020**, *9*, 1832. [CrossRef]
6. Chand, P.; Al-Rawi, M.; James, S.; Antony, J.; Jose, J. A Low-Cost System for Remote Access and Control of Automation Equipment. *Machines* **2021**, *9*, 138. [CrossRef]
7. Mejías, A.; Herrera, R.S.; Márquez, M.A.; Calderón, A.J.; González, I.; Andújar, J.M. Easy Handling of Sensors and Actuators over TCP/IP Networks by Open Source Hardware/Software. *Sensors* **2017**, *17*, 94. [CrossRef] [PubMed]
8. Otoakhia, E.; Jenmanachaiyakun, T.; Afaneh, A.; Alzebda, S.; Mani, M.; Sonbul, O.; Kalashnikov, A. Embedded web server for remote laboratory access for undergraduate students studying electronic engineering. In Proceedings of the 2011 IEEE International Symposium of Circuits and Systems (ISCAS), Rio de Janeiro, Brazil, 15–18 May 2011; pp. 337–340.
9. García-Zubia, J.; Angulo, I.; Hernandez, U.; Castro, M.; Sancristobal, E.; Orduña, P.; Irurzun, J.; de Garibay, J.R. Easily Integrable platform for the deployment of a Remote Laboratory for microcontrollers. In Proceedings of the IEEE EDUCON 2010 Conference, Madrid, Spain, 14–16 April 2010; pp. 327–334.
10. Garcia-Zubia, J.; Diego, L.d.I.; Pablo, O. Evolving towards better architectures for remote laboratories: A practical case. *Int. J. Online Eng.* **2005**, *1*. [CrossRef]
11. Othman, N.A.; Zainodin, M.R.; Anuar, N.; Damanhuri, N.S. Remote monitoring system development via Raspberry-Pi for small scale standalone PV plant. In Proceedings of the 2017 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 24–26 November 2017; pp. 360–365.
12. Saha, S.; Singh, A.; Bera, P.; Kamal, M.N.; Dutta, S.; Gorian, U.; Pramanik, S.; Khan, A.; Sur, S. GPS based smart spy surveillance robotic system using Raspberry Pi for security application and remote sensing. In Proceedings of the 2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 3–5 October 2017; pp. 705–709.
13. Limpraptono, F.Y.; Sudibyo, H.; Ratna, A.A.P.; Arifin, A.S. The design of embedded web server for remote laboratories microcontroller system experiment. In Proceedings of the TENCON 2011-2011 IEEE Region 10 Conference, Bali, Indonesia, 21–24 November 2011; pp. 1198–1202.
14. Yudi Limpraptono, F.; Putri Ratna, A.A.; Sudibyo, H. Remote laboratories multiuser based on embedded web server. In Proceedings of the 2012 9th International Conference on Remote Engineering and Virtual Instrumentation (REV), Bilbao, Spain, 4–6 July 2012; pp. 1–7.
15. Zenzerović, P.; Sučić, V. Remote laboratory for microcontroller systems design. In Proceedings of the 2011 Proceedings of the 34th International Convention MIPRO, Opatija, Croatia, 23–27 May 2011; pp. 1685–1688.
16. Lowe, D.; Newcombe, P.; Stumpers, B. Evaluation of the Use of Remote Laboratories for Secondary School Science Education. *Res. Sci. Educ.* **2013**, *43*, 1573–1898. [CrossRef]
17. Garófano, F.; Gallardo, J.; Guasch, A.; Sánchez, B.; Bragós, R. iLabRS: A remote laboratory for science & technology in secondary education. In Proceedings of the 2012 9th International Conference on Remote Engineering and Virtual Instrumentation (REV), Bilbao, Spain, 4–6 July 2012; pp. 1–4.
18. Karie, N.M.; Sahri, N.M.; Yang, W.; Valli, C.; Kebande, V.R. A Review of Security Standards and Frameworks for IoT-Based Smart Environments. *IEEE Access* **2021**, *9*, 121975–121995. [CrossRef]
19. Iqbal, W.; Abbas, H.; Daneshmand, M.; Rauf, B.; Bangash, Y.A. An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security. *IEEE Internet Things J.* **2020**, *7*, 10250–10276. [CrossRef]
20. Caprolu, M.; Di Pietro, R.; Lombardi, F.; Raponi, S. Edge Computing Perspectives: Architectures, Technologies, and Open Security Issues. In Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 116–123.
21. Roukounaki, A.; Efremidis, S.; Soldatos, J.; Neises, J.; Walloschke, T.; Kefalakis, N. Scalable and Configurable End-to-End Collection and Analysis of IoT Security Data: Towards End-to-End Security in IoT Systems. In Proceedings of the 2019 Global IoT Summit (GIoTS), Aarhus, Denmark, 17–21 June 2019; pp. 1–6.
22. Chou, W. Web Services: Software-as-a-Service (SaaS), Communication, and Beyond. In Proceedings of the 2008 IEEE Congress on Services Part II (services-2 2008), Beijing, China, 23–26 September 2008.

23. Yau, S.S.; Ye, N.; Sarjoughian, H.S.; Huang, D.; Roontiva, A.; Baydogan, M.; Muqsith, M.A. Toward Development of Adaptive Service-Based Software Systems. *IEEE Trans. Serv. Comput.* **2009**, *2*, 247–260. [CrossRef]
24. Gomes, R.L.; Bittencourt, L.F.; Madeira, E.R.M.; Cerqueira, E.C.; Gerla, M. Software-Defined Management of Edge as a Service Networks. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 226–239. [CrossRef]
25. Martínez, E.; Cambronero, M.E.; Díaz, G.; Valero, V. Design and Verification of Web Services Compositions. In Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services, Venice/Mestre, Italy, 24–28 May 2009; pp. 395–400.
26. Wenyu, C.; Dongpu, H.; Dongcheng, T.; Zongbo, H. A model of remote debugger supporting multiple types of connection. In Proceedings of the 2011 International Conference on Electronics, Communications and Control (ICECC), Ningbo, China, 9–11 September 2011; pp. 642–645.
27. Rössler, P.; Höller, R. A novel debug solution for distributed embedded applications and implementation options. In Proceedings of the IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society, Melbourne, VIC, Australia, 7–10 November 2011; pp. 2796–2801.
28. Pomante, L.; De Cesaris, I.; Dell'Osa, S. OATS: An automated test system for OpenOCD. In Proceedings of the 2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), Turin, Italy, 16–18 September 2015; pp. 502–507.
29. Yansong, C.; Weizheng, R.; Wenshen, Z. An Embedded Multi-channel H.264 Video Server/Client Designed for Remote Experiment Education. In Proceedings of the 2010 Second International Workshop on Education Technology and Computer Science, Wuhan, China, 6–7 March 2010; Volume 3, pp. 688–691.
30. Lee, Y.C.; Lee, C.M. Real-Time Smart Home Surveillance System of Based on Raspberry Pi. In Proceedings of the, 2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), Yunlin, Taiwan, 23–25 October 2020; pp. 72–74.
31. Tobagi, F.; Long, J. Client-server challenges for digital video. In Proceedings of the COMPCON Spring 1992, San Francisco, CA, USA, 24–28 February 1992; pp. 88–91.
32. Daros, M.R.; de Lima, J.P.C.; Rochadel, W.; Bento Silva, J.; Simão, J.S. Remote experimentation in basic education using an architecture with Raspberry Pi. In Proceedings of the 2015 3rd Experiment International Conference, Ponta Delgada, Portugal, 2–4 June 2015; pp. 75–78.
33. Kyuchukova, D.; Hristov, G.; Zahariev, P.; Borisov, S. A study on the possibility to use Raspberry Pi as a console server for remote access to devices in virtual learning environments. In Proceedings of the 2015 International Conference on Information Technology Based Higher Education and Training (ITHET), Lisbon, Portugal, 11-13 June 2015; pp. 1–4.
34. Garg, H.; Dave, M. Securing IoT Devices and SecurelyConnecting the Dots Using REST API and Middleware. In Proceedings of the 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), Ghaziabad, India, 18–19 April 2019; pp. 1–6.
35. Wu, W.; Zhang, Q.; Wang, H.J. Edge Computing Security Protection from the Perspective of Classified protection of Cybersecurity. In Proceedings of the 2019 6th International Conference on Information Science and Control Engineering (ICISCE), Shanghai, China, 20–22 December 2019; pp. 278–281.
36. Ozcan, M.O.; Odaci, F.; Ari, I. Remote Debugging for Containerized Applications in Edge Computing Environments. In Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 30–32.
37. Balula, S.; Henriques, R.; Fortunato, J.; Pereira, T.; Borges, H.; Amarante-Segundo, G.; Fernandes, H. Distributed e-lab setup based on the Raspberry Pi: The hydrostatic experiment case study. In Proceedings of the 2015 3rd Experiment International Conference, Ponta Delgada, Portugal, 2–4 June 2015; pp. 282–285.
38. Domski, W.; Mazur, A. Tracking of numerically defined trajectory by free-floating 3D satellite. In Proceedings of the 2019 12th International Workshop on Robot Motion and Control (RoMoCo), Poznan, Poland, 8–10 July 2019; pp. 178–183.