Design and Development of a Remote Lab for Hands-On Education in Mechatronics and Control Engineering

J. Zumsande [*] **S. Bosselmann** [*] **M. Dagen** [*] **T. Ortmaier** [*]

[*] *Leibniz Universität Hannover, Institute of Mechatronic Systems, Hanover, Germany (e-mail: johannes.zumsande@imes.uni-hannover.de).*

Abstract: Demonstrators expand the theoretical lectures of control engineering by a practical hands-on experience to teach students the implementation and effects of a control algorithm and demonstrate the range of field starting with a mathematical model and ending with a controlled system. For this purpose, remotely controlled laboratories (RCL) are an effective alternative to supervised laboratories due to their advantages in 24/7 accessibility and interactivity as well as their low labor consumption. At the Institute of Mechatronic Systems (imes) a RCL was developed to extend the range of courses with a motivating and application-oriented laboratory. It was designed considering minimal latency remote control as well as portability to other environments with different testbeds and different learning management systems. To increase students' motivation a challenge and ranking system has been implemented. For each task a challenge is created with a criterion of optimisation and students compare themselves with their fellow students by an up-to-date ranking.

Keywords: Virtual and remote labs, internet based teaching of control engineering, control education using laboratory equipment

1. INTRODUCTION

The constant increasing amount of students visiting lectures, tutorials and laboratories awakens the necessity of new teaching methods in order to save time, place and money without any loss of quality or quantity. A remotely controlled laboratory (RCL) is an opportunity to handle that issue. It allows students to remotely control a testbed located anywhere in the world from e.g. their private computers, smart phones, tablets, etc. via Internet. A RCL offers a hands-on education without the need of resources like advisers or rooms. Moreover, it is accessible 24/7 so that it fits students' individual schedules. A RCL creates two interfaces, a web interface for user's input and the interface to the testbed, as well as the data processing between them.

As shown in May et al. (2016), RCL are able to increase students' motivation and their skills in personal initiative. Since they operate real devices, a limited access to the testbeds is mandatory in order to prevent concurrency while controlling them. Most of the RCL found on the Internet use a booking system to lock the laboratory for a certain time and limit the access to the reserved time period only. A booking and authentication server environment called iLab, that can be used in a RCL, was built by del Alamo et al. (2011). Unfortunately, the support is outdated since 2011. Heradio and de la Torre (2016) created a Moodle plug-in called EJSApp which combines a RCL (including a booking system) with the learning management Moodle. Ortelt and May (2016), Frerich and Kruse (2015) and Meyes (2017) developed a RCL using the iLab booking system. Roth et al. (2017) designed a RCL for educational use in school. It uses a blocking system so that only one person can control the testbed at a time. The connection to the laboratory is only established when no user is connected to the system.

In this extended abstract we present a new RCL. To enhance students' motivation a "gamifying" aspect is used. The students can gain a score, while solving special tasks, and compare themselves with their fellow students. Existing lectures at the institute already profit from a challenge and comparison system as the students show increased motivation and spend more time on lectures. Finally, the students benefit from a deeper understanding and greater skills in personal initiative. While the students use the RCL they reach predefined milestones to check their work. If all milestones are reached the task is processed completely and the students are able to submit their results.

The technical implementation was optimised regarding low latency feedback, to enhance the sense of interactivity and control for the user, and the actuation of different testbeds with different communication interfaces. For portability reasons the developed RCL is hosted by a self-contained server which offers an easy interface to different learning management systems. After an user data table (e.g. in csv format) has been transmitted, the RCL server works independent and self-contained.

In the lecture "Mechatronic Systems", usually part of the third semester, an inverted pendulum is used as an educational device to demonstrate typical topics in the field of mechatronics and control engineering, such as modelling, identification, state observers and control theory. The controlled inverted pendulum experiences a real sense of achievement and demonstrates an obvious benefit by understanding and using control loops. Therefore, it was used as the first testbed for the presented RCL.

2. WEB INTERFACES

2.1 Interfaces

The web user interface consists of the following components: a main page, information web pages, a booking system (Fig. 1), a testbed control interface (Fig. 2) and an exam web page (Fig. 3).

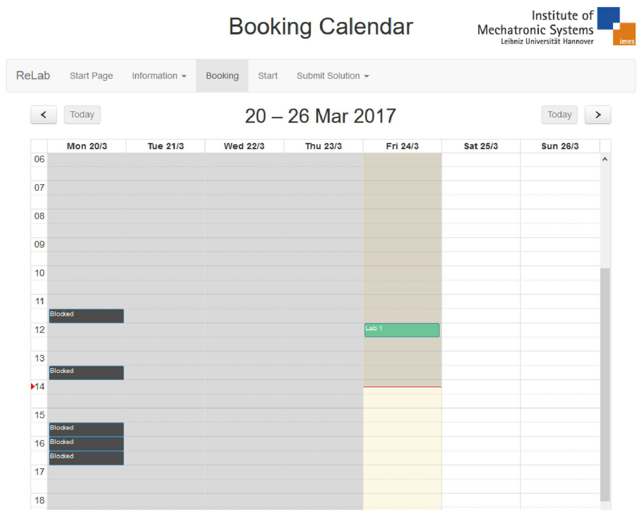
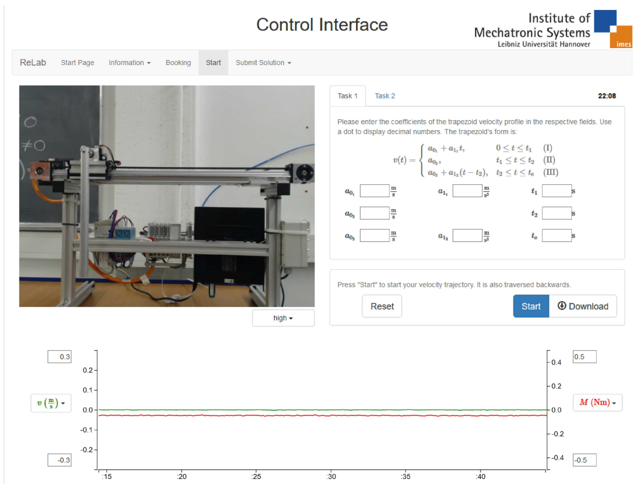


Fig. 1. Screenshot of the web user interface: booking system



Please enter the coefficients of the trapezoid velocity profile in the respective fields. Use a dot to display decimal numbers. The trapezoid's form is:

$$v(t) = \begin{cases} a_{0_1} + a_{1_1} t, & 0 \leq t \leq t_1 \quad \text{(I)} \\ a_{0_2}, & t_1 \leq t \leq t_2 \quad \text{(II)} \\ a_{0_3} + a_{1_3}(t - t_2), & t_2 \leq t \leq t_e \quad \text{(III)} \end{cases}$$

a_{0_1} $\frac{m}{s}$ a_{1_1} $\frac{m}{s^2}$ t_1 s

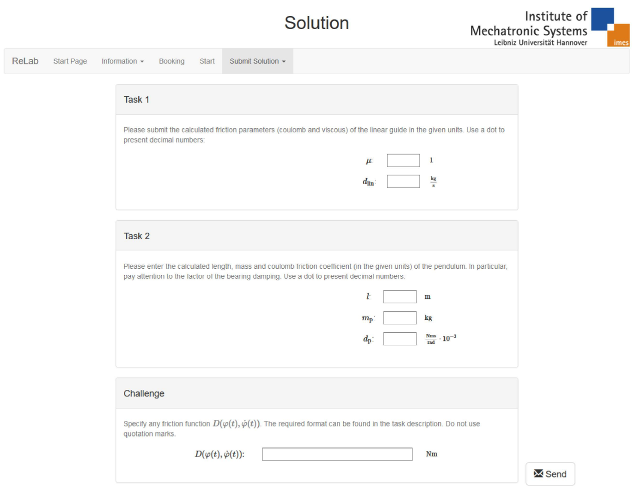
a_{0_2} $\frac{m}{s}$ t_2 s

a_{0_3} $\frac{m}{s}$ a_{1_3} $\frac{m}{s^2}$ t_e s

Press "Start" to start your velocity trajectory. It is also traversed backwards.

Fig. 2. Screenshot of the web user interface: testbed control

The main page shows the up-to-date ranking of the challenge while all relevant information about the tasks and an explanation about the RCL can be found on the information pages. With the booking system the students book the laboratory for a certain time. They are able to see existing reservations of other students or teachers and can choose a free slot for their lab. The testbed can be controlled with minimal latency using the control interface. If desired, all measured values can be exported to a csv table and



Task 1

Please submit the calculated friction parameters (coulomb and viscous) of the linear guide in the given units. Use a dot to present decimal numbers:

μ: 1

d_{lin}: $\frac{\text{kg}}{\text{s}}$

Task 2

Please enter the calculated length, mass and coulomb friction coefficient (in the given units) of the pendulum. In particular, pay attention to the factor of the bearing damping. Use a dot to present decimal numbers:

l: m

m_{p}: kg

d_{p}: $\frac{\text{Nms}}{\text{rad}} \cdot 10^{-3}$

Challenge

Specify any friction function $D(\varphi(t), \dot{\varphi}(t))$. The required format can be found in the task description. Do not use quotation marks.

$D(\varphi(t), \dot{\varphi}(t))$: Nm

Fig. 3. Screenshot of the web user interface: exam web page

analysed after the experiments are completed. To check their results, the students submit them to the RCL server and get a direct feedback.

2.2 JavaScript libraries

Common known libraries *jQuery* and *Bootstrap* are used for interactions on the web user interfaces. Other libraries used are:

- *FullCalendar* (in combination with *Moment.js*) as a calendar for scheduling used for the booking web page,
- *D3.js* for showing the real-time data graph on the control interface and
- *MathJax* for rendering LaTeX equations in html.

3. COMMUNICATION STRUCTURE

The logical sequence of using the RCL is to book the testbed, execute the lab and finally submit a solution to the server. This procedure is taken up for demonstrating the communication structure of the developed RCL, which is shown in Fig. 4.

3.1 Book the testbed

By accessing the booking web page URL with a browser an HTTP (hypertext transfer protocol, defined in Fielding et al. (1999)) request is generated and forwarded by the web server nginx to the Jinja2 templating engine. Jinja2 generates an html document, using a library with predefined html blocks, that is sent back to the client. Afterwards, the client requests all linked .js and .css files, which are served directly by nginx, as it is faster and requires less use of system's resources than delivering them with flask, a Python framework for web applications, and uWSGI, a web application server for deploying flask (see Unbit et al. (2016)).

The calendar view of the booking web page is created by using the JavaScript libraries *FullCalendar* and *Moment.js*. Besides viewing the interface, the generated web page also defines the client side interface to the server

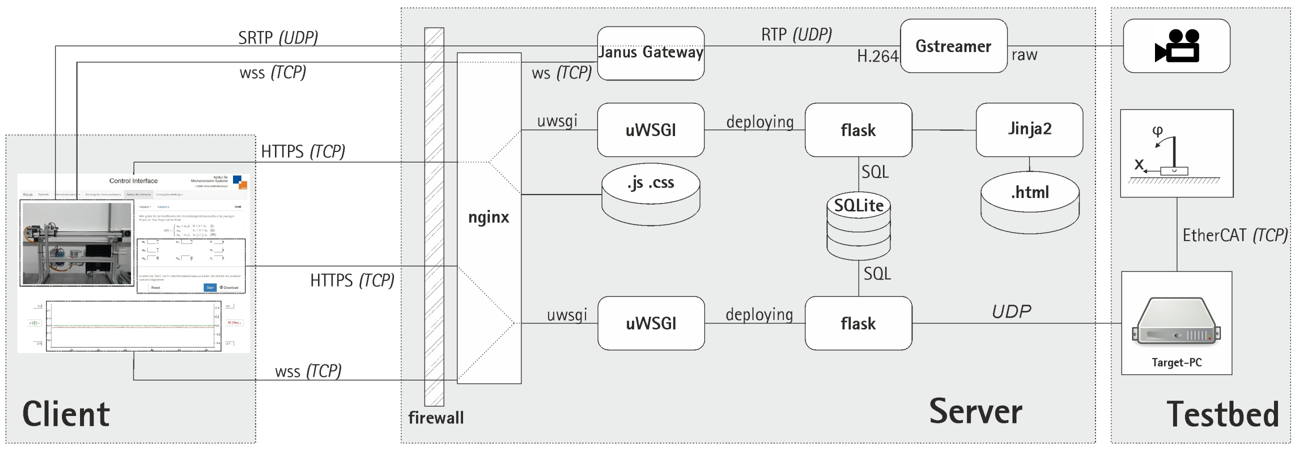


Fig. 4. The server structure consisting of the video stream, view generation and server interactions (from top to bottom)

by specifying the URLs, where server functions can be invoked.

To separate the previous mentioned view generation from dynamic server interactions (invoke functions), all following server interactions are served by a second flask application (see Fig. 4 at the bottom), which links URL addresses to Python functions. The functions are invoked by requesting the unique URLs and parameter (e.g. reservation time) can be passed within the HTTPS request body.

Two types of server interaction communication are possible: WebSocket (ws(s), defined in Fette and Melnikov (2011)) and HTTPS. While HTTPS offers a "question/answer" character between client (question) and server (answer), wss opens a persistent connection between server and client in which both (server and client) are allowed to send packages simultaneously without previous request.

After the web page is loaded completely, the client joins a common wss pool with all clients connected. When one client books the laboratory an actualisation message is sent from the server to the other connected clients in order to keep their calendar view up-to-date. Contrary to HTTPS, the client does not have to request the server for calendar view updates as wss enables to push the actualisation from server to client. HTTPS is used, when a response to the client's request is desired (e.g. a reservation request). The flask application checks a reservation request for permission and refuses it for the following reasons:

- maximum number of tries reached,
- an active, future reservation,
- a past reservation time,
- the selected slot is already booked by another group.

Valid reservations will be registered in the database. After the request is processed, a feedback is sent back to the client and the server sends an actualisation message to all other clients connected to the common wss pool.

3.2 Control the testbed

The control web page is generated similarly like the booking web page, except that the request is checked for a valid reservation by the flask application before it is transmitted to Jinja2.

The controller was designed in Simulink Real-Time to regulate the inverted pendulum and actuate the carriage. It is possible to set parameter and signals of the Simulink application and output measured signals including timestamps with Simulink's UDP (user datagram protocol, defined in Postel (1980)) blocks. Due to Python's wide communication protocol support, such as serial port and Ethernet communication via TCP (transmission control protocol, defined in Postel (1981)) and UDP, it is possible to control different testbeds with the developed RCL structure.

The control signals are sent via HTTPS to the flask application which proves them against predefined conditions like boundaries, sequence of requests and format. The signal is transmitted via UDP to the target PC and afterwards, if the request is corresponding to a predefined milestone, it is registered in the database. The measured data are sent to the client using wss and the view of the real-time graph is renewed with each frame reaching the client.

The testbed can be observed with a real-time video stream. The multimedia framework GStreamer compresses the raw video data coming from a webcam into two selectable H.264 video streams (high and low quality). They are formatted to RTP (real-time transport protocol, defined in Schulzrinne et al. (2003)) streams and sent to Janus Gateway, which secures the streams with DTLS. The following SRTP stream is forwarded to the client. Janus Gateway enables the use of WebRTC, which was developed for client to client video conferencing, for streaming real time video and audio data from server to client. The video streams are controlled (start, stop, select high or low resolution) by a WebSocket connection between the client, Janus Gateway as the backend server and the web server nginx as a reverse proxy server. The connection remains open while the video stream is running.

3.3 Submit a solution

The web page for submitting a solution is also generated using Jinja2 and user's results are checked by the second flask application against predefined boundaries. If all conditions are fulfilled the laboratory is marked as "passed" in the database.

4. EDUCATION

After the technical implementation was demonstrated in the previous chapters, this section will focus in the educational part of the RCL that is separated in tasks and an optimisation challenge.

4.1 Possible tasks

To give an impression which tasks are possible some example tasks for the inverted pendulum will be presented in the following:

- control of the carriage,
- identify the system's parameter,
- implement a state observer,
- implement a control algorithm.

To give an introduction to the RCL an easy start can be to implement a cascade control of the carriage. Students tune the velocity control loop first by stimulating the system with a velocity jump signal and when it fulfills stability conditions they can tune the position control loop of the carriage. It is a fast first success to increase students' motivation. In the associated lecture a mathematical model of the inverted pendulum is deduced and can be used to identify the system's parameter. Mass and length of the pendulum and frictional properties of the system can be identified by stimulating the system with user specified trajectories and measuring the drive torque of the guiding as well as the position and velocity of the pendulum and the carriage. To see the benefit of an identified model another task can give an introduction in state observers. With a comparison between the unfiltered, low-pass filtered, Luenberger and Kalman Filter observed velocity signal, students can observe the effects of several types of signal filtering. They are finally able to submit its own control method to stabilize the inverted pendulum. Therefore, many combination of filters and control methods (e.g. PID controller, state feedback with prefiltering, PI state feedback) are imaginable.

4.2 Challenge

In contrast to the tasks, the challenge does not have a specific solution. It is an optimisation task and the outcome depends on the effort of the students. Possible challenges for the inverted pendulum are:

- tune a control loop against overshoot and command response,
- create a frictional function that fits different trajectories the best,
- write an upswing control for reaching the upper equilibrium position as fast as possible.

In order to increase students' motivation to improve their solutions, a ranking system is established. Each student's solution gains a score and is listed in the ranking. In a second try they can improve their results in order to achieve more score points. To increase the motivation further a prize system is a possible option.

5. USER STUDY

To validate the developed system and get a first impression a user study with students and employees of the institute was designed. The RCL was online for three weeks and students as well as employees have used the laboratory simultaneously to validate the structure under intensive use. The anonymised questionnaire was designed as described in Brooke (1996) to calculate a system usability scale (SUS) with a range from 0 (worst) to 100 (best). 21 Participants (13 employees and 8 students) have tested the RCL and 12 completed the questionnaire (9 employees and 3 students).

The developed RCL reached an arithmetic mean SUS of 85. More detailed statistical evaluation can be seen in Fig. 5, using the grade and corresponding adjectives scales introduced by Bangor et al. (2009). The resulting score indicates an user-friendly and intuitive system. Nevertheless, further studies are planned in order to validate the system with more student participants.

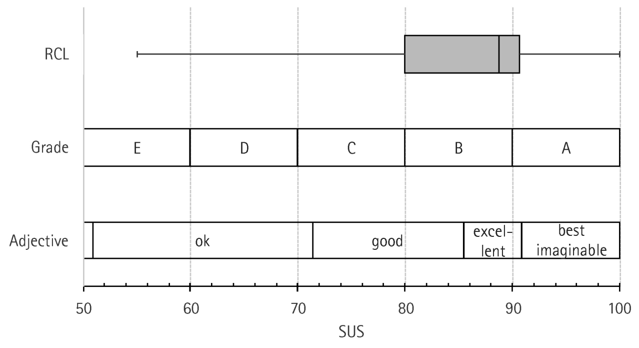


Fig. 5. RCL's SUS in comparison to US school grades and corresponding adjectives

6. CONCLUSION

In this extended abstract a new RCL with a booking system, a control interface and a ranking system to increase students' motivation was introduced. It can be integrated in different learning management systems by transmitting the user database to the RCL server and is working afterwards self-contained. Due to its wide communication protocol support, Python was used as the server side programming language in order to actuate different testbeds with different communication interfaces. Minimal latency feedback was realised by a WebRTC video stream and a data stream transmitted through wss.

In order to enhance server performance nginx delivers the static content while the uWSGI application servers are handling dynamic requests. The testbed can be controlled by specifying Simulink's signals and parameter (e.g. reference variable, gain factor, parameter of transfer functions) or functions (e.g. start/stop, reset, switch of signal flow). Due to the use of Simulink's UDP blocks, the RCL can be transmitted to other Simulink controlled testbeds easily.

While students are working on the exercises, they have to reach predefined milestones to check if the exercise was completed successfully. Finally, they can see their score for the challenge and can optimise it with a second try.

A procedure video of the developed RCL can be found on YouTube (Zumsande (2017)).

Portability to other testbeds is planned for future works. This includes an enhanced modularity and possibilities of individualisation for the developed RCL. Moreover, further user studies will be conducted to validate the obtained first impression.

REFERENCES

Bangor, A., Kortum, P., and Miller, J. (2009). Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3), 114–123.

Brooke, J. (1996). Sus: a 'quick and dirty' usability scale. In P.W. Jordan, B. Thomas, B.A. Weerdmeester, and I.L. McClelland (eds.), *Usability Evaluation in Industry*, chapter 21, 189–194. Taylor and Francis, London.

del Alamo, J., Abelson, H., and Mitchell, D. (2011). The ilab project. URL wikis.mit.edu/confluence/display/ILAB2/home.

Fette, I. and Melnikov, A. (2011). Rfc 6455: The websocket protocol. URL tools.ietf.org/html/rfc6455.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Rfc 2616: Hypertext transfer protocol – http/1.1. URL ietf.org/rfc/rfc2616.txt.

Frerich, S. and Kruse, D. (2015). Virtual learning environments. URL ruhr-uni-bochum.de/elli/virt.html.

Heradio, R. and de la Torre, L. (2016). Ejsapp. URL moodle.org/plugins/mod_ejsapp.

May, D., Terkowsky, C., T.Ortelt, and Tekkaya, A. (2016). The evaluation of remote laboratories development and application of a holistic model for the evaluation of online remote laboratories in manufacturing technology education. In *13th IEEE International Conference on Remote Engineering and Virtual Instrumentation (REV)*.

Meyes, R. (2017). Remote labs. URL www.remote-labs.rwth-aachen.de/.

Ortelt, T. and May, D. (2016). International manufacturing remote lab. URL mintrelab.tu-dortmund.de/das-remote-lab/.

Postel, J. (1980). Rfc 768: User datagram protocol. URL ietf.org/rfc/rfc768.txt.

Postel, J. (1981). Rfc 793: Transmission control protocoll. URL ietf.org/rfc/rfc793.txt.

Roth, D., Maus, S., Altherr, S., Vetter, M., Eckert, B., and Gröber, S. (2017). Remotely controlled laboratories - rcls. URL rcl-munich.informatik.unibw-muenchen.de.

Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. (2003). Rfc 3550: Rtp: A transport protocol for real-time applications. URL tools.ietf.org/html/rfc3550.

Unbit, Koskela, A., Mierzwa, L., and Churchill, D. (2016). uwsgi documentation. URL uwsgi-docs.readthedocs.io/en/latest/StaticFiles.

Zumsande, J. (2017). Relab for hands-on education. URL youtube.com/imesVideo.