Co-funded by
the European Union

Repository of Open Educational Resources for Laboratory Support in Engineering and Natural Science-RELAB

Project Intellectual Output 6

WEB-Laboratory: Tutorial of technical design and implementation

version 1.0, March 1st 2022

"This project has been funded with support from the European Commission. This publication reflects solely the views of the author, and the Commission cannot be held responsible for any use that may be made of the information contained therein."

INTRODUCTION

Purpose of this tutorial is to introduce to main concepts required for development of WebLabs derived from existing, “off-line” Lab excercises.

Our goal is to provide path for teachers with limited resources, and with little or no dedicated IT department support.

We will primarily describe approach based on Linux Debian based Lab servers, and in followup, extended goals tutorials show similar for Windows based Lab servers.

So, basic understanding of Linux Debian OS-es is required, and some of the networking concepts. This approach should be considered in parallel with another, described in parallel series of tutorials, which is based on EJSS and RIP-Server concepts.

There are some fundamental differences in these two approaches: the one described in this and accompanied “RELAB_Weblab_IO5_OffLine2OnLine.pdf”, relies on existing off-line Lab setups, experiments and excercises that need to be quickly converted to on-line versions. By using proven and maintained open source software packages and services, we propose a method how to do this conversion – final end user experience will provide remote desktop connectivity, file sharing and clipboard sharing. So, if students are already familiar with “off-line” Lab setup, they can also do the same with modest additional learning effort. It is not a precanned lab excercise but rather one based on students understanding of remote connectivity concepts and basic understanding of Linux OS.

The other approach, based on EJSS and RIP-Server concept is more streamlined approach requiring additional effort from teacher end to prepare both Lab setup, and front-end Lab excercise. In addition, custom libraries need to be used on Arduino or RPI platform side to enable RPC methods: collecting control input variables and posting output variables (e.g. sensor reading or internal parameters). This is more pre-canned approach enabling Students to explore well defined concepts without any in-depth understanding of remote connectivity and processes on hosting Lab server. In that sense, it is better experience for excercises purely focused on e.g. topic in control theory, signal processing or similar. Students would be able to access setup just through browser. Additional steps though would be required for accessing Lab setups without public/static IP address (not specifically described in current set of tutorials).

Remotely accessible Lab Setups - WebLab

WebLab is recently introduced term, concept that describes (typically) physical Laboratory setups that can be accessed over the public Internet. There is a blurred line of distinction between Remotely Accessible Lab setups over Campus intranet infrastructure and WebLabs.

Motivation for Remotely Accessibly Lab setups over Campus is frequently different from WebLabs – it can be driven by office constraints, safety reasons and comfort/productivity reasons. Since modern VPN services offer almost identical experiences, given the network bandwidth and latencies are sufficient, public and intranet access can be considered the same.

Hence we will focus on alternative approaches, which require less-or-none IT department support and more suitable for wider audience of students. Still, security concerns need to be addressed both in terms of external intrusions, inadvertent access of students to Lab setup resources not intended for Lab excercises, as well as mutual interference of activities between the students.

Since there is a significant overlap in implementation and description of tech design of WebLabs, this document should be used together with "RELAB_Weblab_IO5_OffLine2OnLine.pdf"

Primary additional services to be provided, comparing to "offline" Lab server approach, are:

- Remote desktop connectivity
- File exchange / Clipboad exchange
- Video streaming (optional), unidirectonal (Lab setup→Student desktop)

RELAB WebLab architecture

WebLab architecture includes 3 elements.

1. **Lab Setup** has to include computer that will be used for two-fold reasons:
    - For control of locally attached equipment
    - As communication node thus enabling remote access.

    Apart from the computer, additional equipmnent is part of the Lab setup where actual experiments are done. This equipment must include mechanisms of remote control, nowadays typically based on serial ports (RS232), USB communication, LAB 100/1000Mbps access, WiFi, or PCIe. Obviously, equipment which only has manual controls/displays is not suitable for this purpose. Notable examples are various flavors of Arduino boards, ESP32, RaspberryPi 3 or 4, and many other lab instruments with embedded remote control.

    If certain instruments, or physical phenomenon need to be observed in real-time, WebCamera (USB2/3 connected), should be hooked to the Lab server.

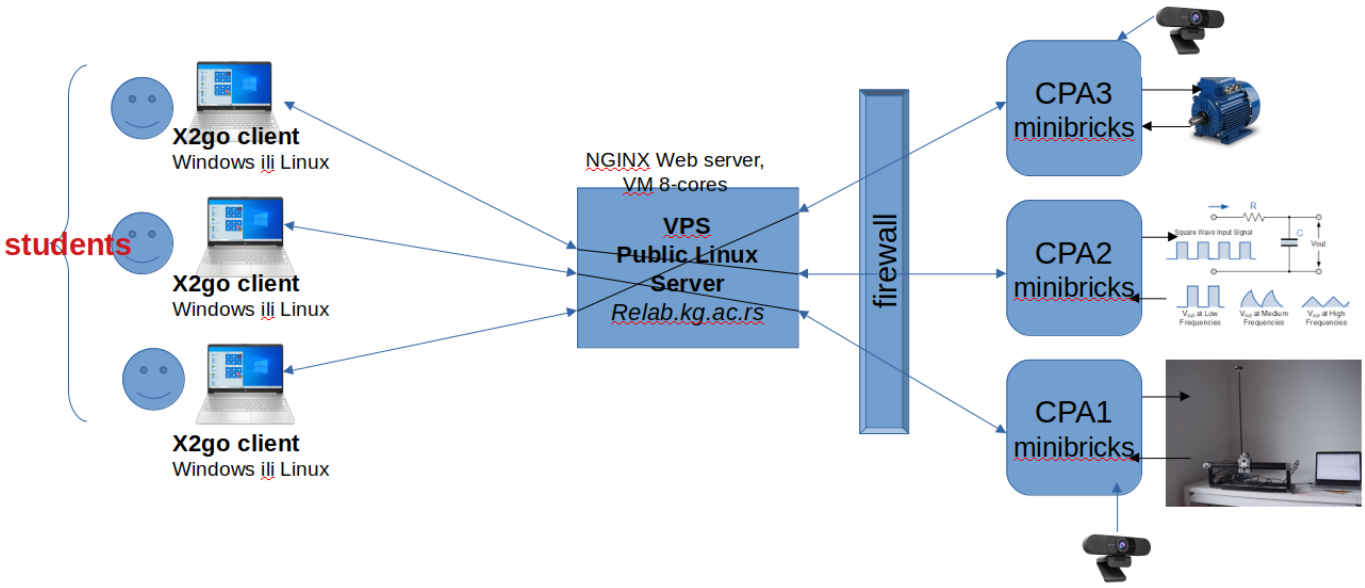
2. **Weblab Public (aka Bastion) Server** is typically VPS (Virtual Private Server). It does not need to be dedicated to this sole purpose since it is highly advised to use Open Source Linux distribution as VPS OS. In the rest of document, we assume this is one of Debian based distros, more specifically Ubuntu. This is public facing service allowing exposure

of selected assets behind the firewall. In this way, surface of attach is significantly reduced to single VPS, following standard methods of protection that need to be strictly followed. Apart from strict "Bastion" (a.k.a. "Proxy Jump") service, regarding WebLab this VPS can host many other Web applications, like one for time reservation and monitoring of accesses.

3. **Client/Student workstation** should include very few software modifications. It can be either standard Windows or Linux-based. This PC or laptop should have moderate performance requirements. It will be used primarily as remote desktop terminal, with provisions for file exchange, mainly to collect artifacts or proofs that excercise has been done. Or, to transfer code snippets, or configuration files that might have been prepared offline.

   Ideally, no additional software packages are needed, which can be accomplished using modern local browsers. But, simple <5min software installation of additional software packages is also acceptable.

Sample architecture for remote desktop and file sharing:



Important alternatives: Apache Guacamolo

This is open source (https://github.com/apache/guacamole-server) remote desktop gateway (cross-platform) supporting SSH, VNC and RDP. It has server implementation which can run on any modern Linux distro, and client that can run on any modern browser. It also has Web-based application that can provide configuration access, user count creation and monitoring of activities. This service can be installed on 'Bastion' server, as additional one, in parallel with the proposed and currently verified solution.

Like any other solution, this may suffer from vulnerabilities, so frequent updates with up-to-date releases are required.

Video streaming:

There are various methods to achieve this service: generating video stream at lab setup and making it available in real-time on student side.

Good example to follow is provided in: https://www.nginx.com/blog/video-streaming-for-remote-learning-with-nginx/

It is based on FFMPEG or VLC used on Lab server side to grab WebCamera video stream (received by Lab server over USB), and convert into RTMP (or similar, e.g. SRT) stream to same VPS hosting "Bastion" server. Additional Web service gets the stream and converts into MPEG-DASH steram of files that can be monitored by multiple users. This gives opportunity to have several students concurrently connected and watching same experiment. One drawback of MPEG-DASH / HLS streaming is inherent latency of 2-3 seconds, which may prevent high rate interactivity.

For lower-latency, if Lab server has public address (X.Y.Z.W), and one-to-one streaming is sufficient, it can be quickly acheived with FFMPEG on Lab server side, and VLC player (need to be installed on Student PC, https://github.com/videolan/vlc):

```
ffmpeg -f video4linux2 -framerate 25 -video_size 1280x720 -i /dev/video0 \
    -f alsa -ac 2 -i sysdefault:CARD=WEBCAM -c:v libx264 -b:v 1600k -preset ultrafast \
    -x264opts keyint=25 -g 25 -pix_fmt yuv420p -c:a aac -b:a 128k \
    -vf "drawtext=fontfile=/usr/share/fonts/truetype/dejavu/DejaVuSansMono.ttf: \
text='CLOUD DETECTION CAMERA 01 UTC-3 %{localtime\:%Y-%m-%dT%T}': fontcolor=white@0.6:
fontsize=12: x=10: y=10: box=1: boxcolor=black: boxborderw=6" \
    -f rtp_mpegts "rtp://X.Y.Z.W:5111?ttl=2"
```

On Student PC side, we will VLC player, with opened connection to: "rtp://X.Y.Z.W:5111".

Another approach for enabling lower latency video streaming is SRT-protocol based. This is more recent, emerging addition (e.g. comparing to RTMP), which guarantees transport latencies. It is UDP-based, and more details can be found at https://github.com/Haivision/srt .

Both FFMPEG and VLC have support for SRT protocols in more recent versions, or they can be enabled as described in https://srtlab.github.io/srt-cookbook/apps/ffmpeg/ and https://srtlab.github.io/srt-cookbook/apps/vlc-media-player/

From Lab server setup, stream can be sent with (if ffmpeg is compiled with `--enable-libsrt`):

```
ffmpeg -f video4linux2 -framerate 25 -video_size 1280x720 -i /dev/video0 \
    -f alsa -ac 2 -i sysdefault:CARD=WEBCAM -c:v libx264 -b:v 1600k -preset ultrafast \
    -x264opts keyint=25 -g 25 -pix_fmt yuv420p -c:a aac -b:a 128k \
    -vf "drawtext=fontfile=/usr/share/fonts/truetype/dejavu/DejaVuSansMono.ttf: \
text='CLOUD DETECTION CAMERA 01 UTC-3 %{localtime\:%Y-%m-%dT%T}': fontcolor=white@0.6:
fontsize=12: x=10: y=10: box=1: boxcolor=black: boxborderw=6" \
    -f mpegts "srt://X.Y.Z.W:5112?mode=caller&passphrase=mypasswd&pbkeylen=32&&oheadbw=100"
```

On Student PC side, we will VLC player, with opened connection to: "srt://X.Y.Z.W:5112". Additional options like passphrase, latency, key length need to be specified in VLC GUI, as described in https://srtlab.github.io/srt-cookbook/apps/vlc-media-player/

Lab server configuration details

Lab server computer configuration can be based on thin-client or mid-performance range PC. It is advised that at least two cores x86 CPU is available with at least 4GB of DDR, and moderate disk space. Additional requirements for performance may come from workloads that are executed locally, but for remote access above, moderate requirements are sufficient. It is worth mentioning that external GPU is of no use for this purpose. Nevertheless, if lab server is not permanently dedicated to this role, different set of constraints may apply. Also, in many cases, PC configurations that are older, 4-6 years, can be used successfully for this role.

Lab servers should preferrably have Linux OS installed. Regarding performance requirements, typically Linux OS has lower demands, and recent versions, like Ubuntu 20.04 can run successfully on older PC configurations. EOL (End-of-Life) for Ubuntu LTS (Long-Term-Support) extends to multiple years, so it can be used for years to come.

If Windows OS is used, it is advised to stay with most recent version (Win 10 or Win 11), primarily due to security concerns and lifetime of third party support (e.g. drivers). Typically, modern versions of Win OS have higher requirement for PC performances (comparing to Linux), but this should not be a concern for OEM installed OS versions, and PC setups that are few yrs old. Ease of use, required application software packages, and familiarity with OS is much more important decision factor.

Lab server network support

Lab server must be hooked to Campus internal LAN, or in rare cases, for smaller installation, directly to public Internet, via DSL, Cable, 4G or FiberOptic modem.

In former case, connection over 100/1000Mbps Ethernet to Campus internal LAN, access to public internet is enabled for Lab server, but it is not mandatory. Access to ‘Bastion’ server from the internal LAN must be enabled though.

One important detail will define many followup actions. Are public IP addresses available to Lab servers? Do we want to expose fully Lab servers with potentially sensitive lab equipment to public internet? We need to keep in mind that public IP addresses are sparse resource and for IPv4 addressing scheme, there are only few billion unique addresses available.

Strong advise is to avoid having Lab servers directly accessible on public Internet with their own IP addresses. If this is still the decision (perhaps temporary), make sure that for Windows security patches are immediately applied as soon as they are released. Using a public IP address is the same as the advantage: It allows ANYONE to (try to) connect to your device directly from the Internet. Since communication over public internet is ALWAYS two-way, when you connect to Internet, Internet connects to you. By exploiting various vulnerabilities, people without good intentions can access your files, or modify access parameters or anonimously do any action they please. There are even publicly available Internet services, not to mention custom built scanner, that regularly scan all IP addresses in public space, and seek for known vulnerabilities. Incidentally, your real IP address can be used not only to hack into your Lab network, but can be used, on your behalf as launchpad for DDoS attack, in both directions. So either your network will be bombarding others, or externally your router and network can be overwhelmed with DDoS atch.Such attacks are often carried out against gamers and streamers, for example, to knock an opponent out of the competition by sabotaging their Internet connection.

Nevertheless, temporarily (if possible), you can set public IP address, for quick fixes and most often for initial setups. E.g. if you are doing initial configuration of WebLab setup, and doing that remotely, someone locally needs to install OS and provision, e.g. in case of Linux OS, SSH access, hook to the LAN and assign static, public IP address. After that, all the rest can be done remotely, as described in details in "RELAB_Weblab_IO5_OffLine2OnLine.pdf".

In later case, regarding network connectivity, if Lab server (for some isolated setup), connects via DSL, Cable, Fiber, 4G, with dynamic IP address assigned, and being behind NAT router, there is an option to open specific port, but only if static IP address is available for your modem. In more frequent cases, this is not possible. Still by use of "Proxy Jump" server that can be provisioned remotely, we can make Lab server available to the remote students: Lab server would need to initiate reverse SSH tunnel connection (as described in "RELAB_Weblab_IO5_OffLine2OnLine.pdf").

Network bandwidth for ensuring good remote desktop connectivity with any of the above methods is not high, so conenctions with 2Mbps or more will be sufficient. Low latency is of higher concern, as any latency (ping RTT) beyond 150ms (or at the max 200ms) will make remote desktop connection experience unacceptable.

Lab server remote accessibility add-ons

If Windows OS is used, it is necessary to enable Windows Firewall / Defender or 3rd party packages, e.g. Norton 360. After that Windows Remote Desktop access need to be enabled. Please note that Windows Remote Desktop is not enabled on Windows Home editions, by default.

Very common open source alternative to this Windows RDP is VNC service (, which can be installed on all Windows versions, as well as Linux OS. Most free VNC versions do not have built-in encryption of traffic. There are multiple implementations of this service, based on common protocol defined in RFC 6143. X2GO file sharing need to be explicitly enabled on server and client side, providing mapped folders that are securely synchronized. On server side, mapped folder CLIENT-FOLDER will appear as ~/media/disk/_CLIENT-FOLDER ('/' or '\' are replaced with '_').

For Linux OS based Lab server (not available for Windows OS server), apart from VNC server option, X2GO open source software package (https://wiki.x2go.org/doku.php/download:start) allows low-latency connections, SSH based, thus fully encrypted over-the-network solution. X2GO server<=>X2GO client connection allows file sharing. X2GO is based on modified NX3 protocol, which does compression of X11 events:

```
RemoteClient (e.g. xterm) <=> nxproxyClient <=>(Web)<=> nxproxyServer<=>X server
```

Not only Linux desktop environments are suitable for X2GO, but XFCE, LXDE and MATE will always work. Hence, additional desktop environment may be required (as described in "RELAB_Weblab_IO5_OffLine2OnLine.pdf").

Both VNC (https://github.com/RealVNC, https://github.com/TigerVNC/tigervnc, https://github.com/ultravnc/UltraVNC), and X2GO (https://code.x2go.org/releases/X2GoClient_latest_mswin32-setup.exe) have Windows clients.

Bastion/Weblab Public server configuration details (Linux, Ubuntu)

VPS Linux is typically already configured for public facing roles with necessary packages installed and some level of network security measures provisioned. This is the case either for commercial solutions like AWS, Google, Digital Ocean, Linode, Azure, or University hosted cloud services. This means that SSH access is already enabled, with few accounts created, at least one of them with 'sudo-er' privilages in order to execute follow-up steps. It may reside in same LAN, or multiple LAN even behind different firewalls.

Single bastion server is sufficient for multiple groups of Lab setups, and limited by available bandwidth and compute power to less extent. Bastion server is the only one Linux server accepting public SSH connections.

If a user wants to access "hidden" (from public Internet) Lab setup machine, they need to connect to the bastion first, make another SSH connection from the bastion to the final destination. This process is sometimes called "proxy jump", and can be automated.

Details how to configure Bastion server are provided in "RELAB_Weblab_IO5_OffLine2OnLine.pdf"

Client / Student workstation configuration (Windows or Linux)

To enable Remote Desktop experience, we will use X2GO software package, by installing the server side on Lab Server, and X2GO client side on Student workstation. X2GO server side is available for Linux only.

Current approach is based on static distribution of students between groups being allocated single Lab setup (of same type) per group. As a follow-up we will provide reservation service for students so they can reserve time slot on a setup. By implementing customized X2GO session-broker service, we will also provide routing ability to available setup of selected type. In this way Student role is further simplified, so SSH keys and other X2GO session parameters will be exchanged dynamically at the beginning of X2GO session.

Additional approaches, like one based on Apache Guacamolo (https://guacamole.apache.org/) software, can enable true 'client-less' approach, i.e. without any additional software package installed on Student PC side.

This approach is limited to VNC servers and free RDP servers. In this case, student would just need to point to Guacamolo service (started on Bastion server'), and provide credentials (user name and password). After that, it will be directed to routed Lab setup. This approach will be also described as extension and alternative for Windows servers towards end of the RELAB project.