Co-funded by
the European Union

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

Project Intellectual Output 5

WEB-Laboratory: design, implementation, maintenance

“Enable conversion of off-line Lab setups to remotely accessible WebLabs”

version 1.0, Jan 31st 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

It is frequently the case that Laboratory experiments are developed first in ad-hoc manner using available equipment and familiar computer resources, for on-the-site Lab work. While not ideal, this approach is very common.

But global pandemics, as well as growing interest in distant learning, are pushing requirements for Lab setups even further. Our goal in this part of RELAB project is to offer incremental approach: helping teachers leverage previous efforts in creation of Lab setups, but still make them remotely accessible as long as safety standards are met.

Under the above assumption, this document won't cover dedicated WebLabs, created from ground-up having in mind remote accessibility. These WebLabs should take into account this requirement from the very beginning, select appropriate platform/framework/APIs at several levels: Lab setup server, Proxy Server and the Client (Student's workstation). There are many benefits, e.g. well controlled environment allowing productive turnaround, also increased security aspects. Also, proposed solution is meant to be used by a teacher or small group of teachers without extensive IT support – this is the primary driver of our architectural decisions (since many different paths are possible).

Incremental approach will include three elements based on standard, open-source, off-the-shelf technologies. Still, some elements like Lab Setup registration, Student registration, real-time resource allocation, video streaming proxying, and admin ssh access via browser would require additional development effort.

This intelectual output will be provided under terms of liberal licenses (final selection of MIT,BSD, or GPL licenses will be discussed with TEMPUS office). Necessary scripts, software packages and documentation will be hosted on GITHUB and relab.kg.ac.rs, to enable easy dissemination.

For Lab Setup configuration, intermediate familiarity with Linux OS is required. For WebLab/Bastion server, familiarity with Linux OS should be at intermediate/advanced level, but this host is unique and can serve many Lab Setups and even multiple institutions within same or even few Universities.

Introduction to RELAB WebLab architecture for online access

This solution is created with following criteria in mind:

- Simplicity in mind for end users, i.e. Students, in order to allow quick access, easy installation, yet similar experience as being physically present.
- There are already created Lab setups using Linux boxes, with attached lab equipment, EVMs or similar – used for offline work, with student physical presence.
- Conversion of offline Lab setup (as described in previous bullet) to online Lab setup, should not take much time and great IT expertise of teacher
- Solution should use only open source software packages with liberal licenses – not freeware or commercial options
- WebLab/Bastion server installation and setup can be more complex, but it can be shared between multiple Lab setups, including Lab setups from different location. This is one-time effort, based on steps and descriptions in this document.
- Avoid putting multiple Lab setups on public Internet, to reduce "surface-of-attack" to single Bastion server only. Since various pieces of Lab equipment might reside on internal LAN, just exposing Lab setups with the public IP, can also create network configuration issues
- Among various open-source, free to use remote desktop solutions, we selected X2GO, as reliable, secure and low-latency option. All network connections are established are either SSH or reverse SSH based.

RELAB WebLab architecture elements

Proposed architecture includes 3 elements.

1. **Lab Setup** configuration with all the necessary equipment attached, as required for on-the-side student lab tasks, e.g. attached Arduino, motors, sensors, camera, etc. This is typically already available and beyond scope of this document.
    - **Lab Server** is part of the Lab setup. We assume this is Ubuntu/Debian based lightweight computer (x86-based or ARM-based like RPi). We will provide detailed explanation about its configuration and necessary software add-ons. Important assumption here is that Lab Server does not have (IPv4 or IPv6) public address. We assume this setup is behind NAT, in most typical Lab LAN environment. In addition we won't discuss VPN option as it requires more complex IT setups, which may also create other security concerns.
2. **Weblab Public (aka Bastion) Server** is somewhat arbitrary term, but this is typically VPS (Virtual Private Server) available as Cloud asset, provided by University Campus IT service, or rented from Public Cloud Provider (AWS, DigitalOcean, Azure, Google, or some local cloud provider). It can be used for multiple purposes, like Web server, Chat/Blog Server, File Server, Streaming server, etc. We will limit the scope of discussion in this document to additional (very few) services/configuration modifications required for the proposed solution. This is also typically Linux-based server.

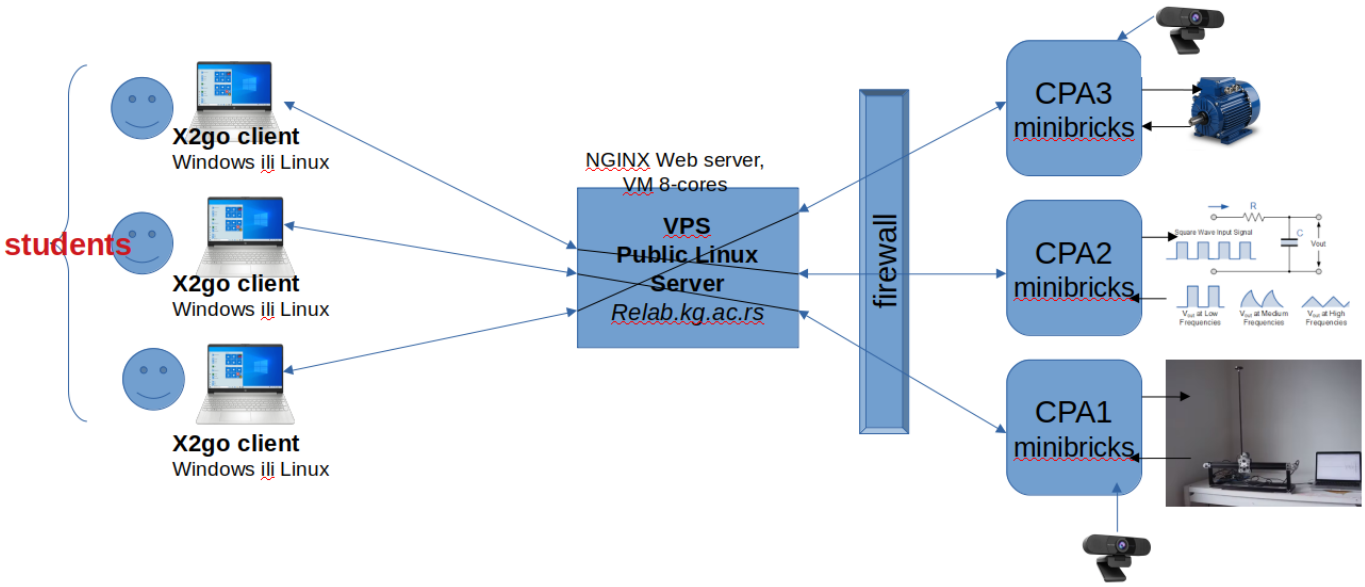
- In follow-up step we will provide software solution for web-based registration of new Lab Setup resources and new Students. Resource (time slot) allocation for the given student and given setup will be provided. Necessary logging/journalling of activities for monitoring student work will be included and review mechanism provided. Finally carefully tuned (chroot/jailed ssh) security solution is needed and is also subject of this solution.
- In this specific role of a server which does public facing, it is called 'Bastion' or 'Proxy Jump' server.
- Weblab / Bastion server specific requirements are mainly on network bandwidth side, as it mostly acts as communication node.

3. **Client/Student workstation** needs very few software modifications. It can be either Windows or Linux-based. Only two additional software packages that can be easily installed (<5mins), are required.
   - As a stretch goal, we will provide browser based solution, that should enable students to access remote Lab setup without any new software installed, apart from signing up and providing supplied credentials. In next phase of project, as an alternative, we will provide Apache Guacamole based solution, with necessary modifications included.

RELAB WEBLAB WITH MULTIPLE LINUX+ARDUINO BOXES

Configuration depicted below is created in Laboratory of University of Kragujevac.

Access to these lab setups have been implemented via VPS, which also runs relab.kg.ac.rs web site. Thanks to Linux OS, multiple services can run concurrently.



This pilot installation consists of:

- 5 Lab setups
  - based on Gigabyte Minibricks mini PCs with Ubuntu 20.04, each with Arduino board and protoboard connected to Arduino I/O (analog and digital inputs and outputs). Additional details are provided in separate document
  - Each setup has USB camera hooked to mini PC so the students can observe real behavior, like LEDs being turned on/off, DC motor movement, etc.
- Bastion server running relab.kg.ac.rs
  - VPS with Ubuntu 20.04 LTS also acting as web server. This is configuration with 4 dedicated cores and 8GB of RAM, along with 80GB HDD/SSD.
- Student client PCs/laptops
  - Using this intial solution, close to 100-120 students have remotely accessed Lab setups instead of coming to the lab and executing short Lab experiments (<10-15mins).
  - There are no special requirements for these setups: they can be either standard Windows or Linux desktops or laptops. Current solution is based on X2GO server/client solution, so access from IoS or Android devices is not possible.

Brief description of Lab server configuration (Linux, Ubuntu)

Through the rest of this document we will assume lab setup which includes Linux-based thin server (e.g. x86 MiniBricks, or RPI4). Debian based distro Ubuntu will be used as example. Below instructions are provided to depict overall process – exact commands are provided in the appendix.

- First step would be to configure, connect and verify initial Lab setup that is used by students coming to the lab. This includes installation of multiple standard debian packages on top of Ubuntu 20.04LTS
- Second step is generation of ssh keys and installation on 'Bastion' server.
  - This step creates id_rsa private keys, and id_rsa.pub public keys
  - These keys need to be transfered to the WebLab Public Server and included in authorized_keys of weblab1 user (on WebLab Public Server)

Bried description of Bastion/Weblab Public server configuration (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.
- As many specific user accounts (named weblabcpaX) as Lab setups are available with restricted privilages need to be created.
- We should verify network connectivity between from Bastion/Weblab server to Lab setups, and vice versa.

Brief description of 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, whereas X2GO client is available for both Windows and Linux OS-es.
- Additional ZIP package is provided with necessary batch and shell scripts and configuration files, to allow one-click connectivity from Student workstation to Lab setup.

Detailed Bastion (aka 'proxy jump') server configuration

Bastion server is public-facing (i.e. host with public IP), hardened systems that serve as an entry point to systems behind a firewall or other restricted location. These systems 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. This reduces "surface-of-attack" to single server only.

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.

In our solution Lab setups also establish SSH reverse tunnel (i.e. Lab setup initiates and maintains connection to the Bastion server), in such way creating opportunity for external users (Student/Client machines) to access Lab setups not exposed to the public internet.

Lab Setup dedicated user accounts

Bastion server has dedicated user accounts just for Lab setup connections, specially created with limited set of permissions: "weblabcpa", "weblabcpa2", …, "weblabcpa5".

This is accomplished using Linux standard 'chroot' commands. In this way only a subset of commands on Bastion server can be used by these user accounts.

In normal operation, it is not expected that Students would log into Bastion server at all. They would rather use automated scripts, that we provide, to access Lab setups using single command (i.e. provided batch files for Windows student client or shell scripts for Linux student client).

This is accomplished with Linux command, adduser:

```
sudo adduser weblabcpa
sudo adduser weblabcpa2
…
sudo adduser weblabcpa5
```

For each user, we need to provide password, but for these accounts, we will allow only passwordless login, i.e. w/o password but with SSH key (hence strong password as good habit are OK, but not mandatory).

In order to allow logging with restricted privilages, we will create special "`chroot`" folder, in this case, "`/home/testjailed`".

A *chroot* is a way of isolating applications from the rest of your computer, by putting them in a *jail*. This is particularly useful if you are testing an application which could potentially alter important system files, or which may be insecure. This can be done for subset of user account on given setup. This will be used for weblabcpaX accounts.

```
mkdir –p /home/testjailed/{bin,lib64,etc,home}
cp –v /bin/bash /home/testjailed/bin
```

Now we need to find all libraries required by bash. This is done with ldd command:

```
ldd /bin/bash
```

In Ubuntu 20.04LTS case we get this list:

```
linux-vdso.so.1 (0x00007ffe901d7000)
libtinfo.so.6 => /lib/x86_64-linux-gnu/libtinfo.so.6 (0x00007fdbb22a7000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007fdbb22a1000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fdbb20af000)
/lib64/ld-linux-x86-64.so.2 (0x00007fdbb2416000)
```

All the above libraries need to be copied (from /lib) to "/*home*/testjailed/lib" folder :

```
cp -v /lib64/{libtinfo.so.5,libdl.so.2,libc.so.6,ld-linux-x86-64.so.2}
/home/testjailed/lib64/
```

Then we need to copy account files:

```
mkdir /home/testjailed/etc
cp -vf /etc/{passwd,group} /home/testjailed/etc/
```

Finally we can modify *"/etc/ssh/sshd_config"* file, to include, to configure SSH to use chroot jail (i.e. "testjailed" in this case):

```
Match User weblabcpa
    ChrootDirectory /home/testjailed
    PasswordAuthentication no

Match User weblabcpa2
    ChrootDirectory /home/testjailed
    PasswordAuthentication no

Match User weblabcpa3
    ChrootDirectory /home/testjailed
    PasswordAuthentication no

Match User weblabcpa4
    ChrootDirectory /home/testjailed
    PasswordAuthentication no

Match User weblabcpa5
    ChrootDirectory /home/testjailed
    PasswordAuthentication no
```

And restart ssh service: `sudo systemctl restart ssh.service`

In oder to enable passwordless access, we need to copy id_rsa.pub public keys, generated on Lab Setups (with ssh-keygen) to `.ssh/authorized_keys` files, for all weblabcpaX accounts.
These keys can be copy/pasted/transferred from Lab Setups, either as regular (scp based) file transfer, or using ssh-copy-id utility.

To make sure that all SSH connections are enabled, you can manually check using cmd line SSH command (from Lab Setup to WebLab/Bastion server).

Monitoring of operation on Bastion server

Currently established connections can be monitored in "*/var/log/auth.log*" standard files

```
grep "weblabcpa" /var/log/auth.log
```

and number of connections successfully created by Lab Setups using following command:

```
relab_user@relabserver01:~/LOGS$ sudo lsof -iTCP -sTCP:LISTEN | grep sshd
[sudo] password for relab_user:
sshd      2142135            root    3u  IPv4 26149991      0t0  TCP *:ssh (LISTEN)
sshd      2142135            root    4u  IPv6 26149993      0t0  TCP *:ssh (LISTEN)
sshd      2340265      weblabcpa2   10u  IPv6 28404097      0t0  TCP ip6-localhost:42287
(LISTEN)
sshd      2340265      weblabcpa2   11u  IPv4 28404098      0t0  TCP localhost:42287 (LISTEN)
sshd      2340265      weblabcpa2   12u  IPv6 28404101      0t0  TCP ip6-localhost:22026
(LISTEN)
sshd      2340265      weblabcpa2   13u  IPv4 28404102      0t0  TCP localhost:22026 (LISTEN)
sshd      2340967       weblabcpa   10u  IPv6 28409371      0t0  TCP ip6-localhost:41310
(LISTEN)
sshd      2340967       weblabcpa   11u  IPv4 28409372      0t0  TCP localhost:41310 (LISTEN)
sshd      2340967       weblabcpa   12u  IPv6 28409375      0t0  TCP ip6-localhost:22024
(LISTEN)
sshd      2340967       weblabcpa   13u  IPv4 28409376      0t0  TCP localhost:22024 (LISTEN)
sshd      2425492      weblabcpa4   10u  IPv6 29142437      0t0  TCP ip6-localhost:56902
(LISTEN)
sshd      2425492      weblabcpa4   11u  IPv4 29142438      0t0  TCP localhost:56902 (LISTEN)
sshd      2425492      weblabcpa4   12u  IPv6 29142441      0t0  TCP ip6-localhost:22030
(LISTEN)
sshd      2425492      weblabcpa4   13u  IPv4 29142442      0t0  TCP localhost:22030 (LISTEN)
sshd      2510791      weblabcpa5   10u  IPv6 29690310      0t0  TCP ip6-localhost:49842
(LISTEN)
sshd      2510791      weblabcpa5   11u  IPv4 29690311      0t0  TCP localhost:49842 (LISTEN)
sshd      2510791      weblabcpa5   12u  IPv6 29690314      0t0  TCP ip6-localhost:22032
(LISTEN)
sshd      2510791      weblabcpa5   13u  IPv4 29690315      0t0  TCP localhost:22032 (LISTEN)
sshd      2550356      weblabcpa3   10u  IPv6 29982993      0t0  TCP ip6-localhost:33894
(LISTEN)
sshd      2550356      weblabcpa3   11u  IPv4 29982994      0t0  TCP localhost:33894 (LISTEN)
sshd      2550356      weblabcpa3   12u  IPv6 29982997      0t0  TCP ip6-localhost:22028
(LISTEN)
sshd      2550356      weblabcpa3   13u  IPv4 29982998      0t0  TCP localhost:22028 (LISTEN)
```

Lab setup server configuration (Linux based)

Lab setup servers are affordable mini PCs with 4 GB of RAM, 2 cores and 80GB of SSD storage. Same approach can be done using Raspberry PI4, whose Linux distro is also Debian based.

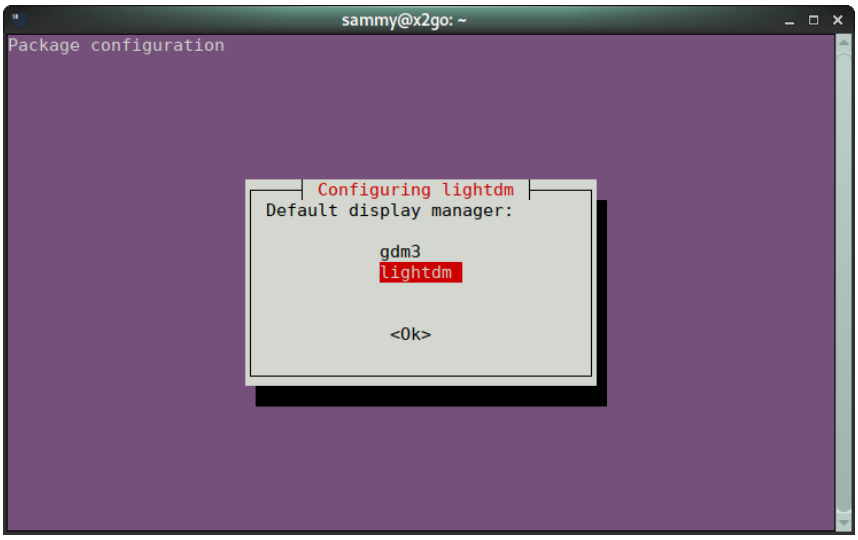
Linux OS used for Lab setups, is Ubuntu 20.04.3 LTS. In order to prepare miniPC for this role, following modifications are required:

SSH service

- Console (terminal, keyboard, mouse) are only temporarily needed, allowing setups to be more tightly packed (so called headless configuration), but they need to be available for OS installation and next step (SSH enablement)
- Connect Lab setup miniPC to the local LAN, then update packages to the most recent
  - `sudo apt update`
- SSH need to be enabled
  - `sudo apt install openssh-server`
  - After this step, you can switch to 'headless' configuration, i.e. remove monitor, keyboard and mouse

Add Desktop environment

- Install Desktop environment to allow unrestricted remote use of applications, on Lab setup miniPC:
  - Install and configure full desktop environment (opportunistic option, to avoid many explicitly selected packages). Please note that this step may take up to 5 minutes, due to many dependent packages being installed:
    - `sudo apt-get install xubuntu-desktop`
  - When prompted to choose display manager, select `lightdm`:



X2GO service

- Install X2GO server on Lab setup miniPC:

- `sudo apt-get install x2goserver x2goserver-xsession`

After this step, you should verify connectivity using X2GO (from local Linux or Windows setup).

- Next step is to enable permanent SSH reverse tunnel, by enabling autossh service. This service opens and restores reverse tunnel at boot time, and also during up time if it fails for any reason. In this way we have permanent connection to Bastion server (relab.kg.ac.rs)

Setup procedure is as follows:

- We need first to prepare SSH keys for passwordless connection, from Lab setup server to Bastion server. These keys are generated using:
  - `mkdir -p $HOME/setupPROXY/sshkeys/weblabcpa2/`
  - `ssh-keygen -f $HOME/setupPROXY/sshkeys/weblabcpa2/ -t rsa`
  - `ssh-keygen -t rsa -b 4096 -C "weblabcpa2@example.com" -f $HOME/setupPROXY/sshkeys/weblabcpa2/id_rsa`
  - Now copy keys to Bastion server, either manually or using:
    - `ssh-copy-id -i $HOME/setupPROXY/sshkeys/weblabcpa2/id_rsa.pub`
  - Verify that SSH connection from Lab setup to Bastion can be established
    - `ssh -i $HOME/setupPROXY/sshkeys/weblabcpa2/id_rsa weblabcpa2@relab.kg.ac.rs`
  - Also, you can verify on Bastion server, that in `/home/weblabcpa2/.ssh/authorized_keys` file, keys for CPA2 have been added

- Install AUTOSSH package:
  - `sudo apt install -y autossh`
- Create autossh.service file in home folder (vi ~/autossh.service)

```
[Unit]
Description=Autossh, keeps a reverse tunnel to 'relab.kg.ac.rs' open
After=network.target
[Service]
User=cpa2
Environment="AUTOSSH_GATETIME=0"
Restart=always
RestartSec=30
Type=simple
ExecStart=/usr/bin/autossh -o "ServerAliveInterval 10" -o "ServerAliveCountMax 3" -
o "ExitOnForwardFailure=yes" -i /home/cpa2/setupPROXY/sshkeys/weblabcpa2/id_rsa  -N
-R 22026:localhost:22 weblabcpa2@relab.kg.ac.rs
StandardOutput=journal

[Install]
WantedBy=multi-user.target
```

- `cp ~/autossh.service /lib/systemd/system/autossh.service`
- `sudo ln -s /lib/systemd/system/autossh.service \ /etc/systemd/system/autossh.service`
- `sudo systemctl daemon-reload`
- `sudo systemctl start autossh`
- `sudo systemctl status autossh`
  - Previous commands verifies the status of autossh service, and shows output like this:

```
● autossh.service - Autossh, keeps a reverse tunnel to 'relab.kg.ac.rs' open
     Loaded: loaded (/lib/systemd/system/autossh.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2022-01-13 20:47:08 CET; 2 weeks 4 days ago
   Main PID: 682 (autossh)
```

```
     Tasks: 2 (limit: 4545)
    Memory: 1.5M
    CGroup: /system.slice/autossh.service
            ├─ 682 /usr/lib/autossh/autossh -o ServerAliveInterval 10 -o ServerAliveCountMax
3 -o ExitOnForwardFailure=yes -i /home/cpa2/setupPROXY/sshkeys/weblabcpa2/id_rsa -N -R
22026:localhost:22 webl>
            └─1118 /usr/bin/ssh -L 42287:127.0.0.1:42287 -R 42287:127.0.0.1:42288 -o
ServerAliveInterval 10 -o ServerAliveCountMax 3 -o ExitOnForwardFailure=yes -i
/home/cpa2/setupPROXY/sshkeys/weblabcpa>
...
јан 13 20:47:19 cpa2-GB-BACE-3000 autossh[682]: starting ssh (count 8)
јан 13 20:47:19 cpa2-GB-BACE-3000 autossh[682]: ssh child pid is 1118
```

- `sudo systemctl enable autossh`
  - Previous command is needed to enable service at boot time

In this example we used port 22026 for reverse SSH connection to Bastion server side. It can be verified using monitoring commands on Bastion server side, as described in previous chapter.

Also from Bastion server side, you should be able to connect to CPA2 using:

`ssh -p 22026` cpa2@127.0.0.1, and providing password

Ubuntu 20.04 updates for known issues (Ubuntu-specific)

In order to avoid screensaver related lockup during remote session – add xorg.conf to /etc/X11/ folder:

```
Section "Monitor"
Option          "DPMS"
EndSection

Section "ServerLayout"
Option          "BlankTime"     "0"
Option          "StandbyTime"   "0"
Option          "SuspendTime"   "0"
Option          "OffTime"       "0"
EndSection
```

One more Ubuntu 20.04/20.10 specific issue pops up only at remote desktop connection (X2GO in this case), always (at login) prompting "`Authentication is required to create a color profile/managed device`".

This is triggered by Polkit, which is an application authorization framework. When you connect to Ubuntu remotely you will see the above errors because the Polkit Policy file cannot be accessed without superuser authentication.

Remedy for this issue is:

```
sudo vi /etc/polkit-1/localauthority.conf.d/02-allow-colord.conf
```

and include the following:

```
polkit.addRule(function(action, subject) {
 if ((action.id == "org.freedesktop.color-manager.create-device" ||
 action.id == "org.freedesktop.color-manager.create-profile" ||
 action.id == "org.freedesktop.color-manager.delete-device" ||
 action.id == "org.freedesktop.color-manager.delete-profile" ||
 action.id == "org.freedesktop.color-manager.modify-device" ||
 action.id == "org.freedesktop.color-manager.modify-profile") &&
 subject.isInGroup("{users}")) {
 return polkit.Result.YES;
 }
});
```

Pilot setup at University of Kragujevac:

Limit X2GO session time

Finally in order to limit X2GO session time, we need to add CRON task, executed every 5 minutes. It will check if X2GO session is established, calculate its duration and terminate after ~15minutes. This is achieved by first adding special script (x2golimittime) along with other X2GO utilities (upon X2GO server package installation, as described earlier) – into folder */usr/lib/x2go/*

```
#!/bin/bash
# Below constant defines timeout
maxSESSION=800
string1=`/usr/bin/x2golistsessions | cut -f6 -d'|'`
if [ -z "$string1" ]
then
    echo "x2go session list is empty" > /dev/null
else
    string2=`date -Iseconds | cut -f2 -d'T' | cut -f1 -d'+'`
    StartDate=$(date -u -d "$string1" +"%s")
    FinalDate=$(date -u -d "$string2" +"%s")
    tdiff=`date -u -d "0 $FinalDate sec - $StartDate sec" +"%H:%M:%S"`
    #echo $tdiff
    tdiffIN=(${tdiff//:/ })
    tHOURS=${tdiffIN[0]}
    tMINS=${tdiffIN[1]}
    tSECONDS=${tdiffIN[2]}
    #echo "$tHOURS $tMINS $tSECONDS"
    totSECONDS="$(($tHOURS*3600 + $tMINS*60 + $tSECONDS))"
    #echo $totSECONDS
    if [ "$totSECONDS" -gt "$maxSESSION" ]; then
      stringSESSION=`/usr/bin/x2golistsessions | cut -f2 -d'|'`
      #echo "Terminate session $stringSESSION"
      /usr/bin/x2goterminate-session $stringSESSION
    fi
fi
```

Then we need to add this task to list of CRON tasks, using: `sudo crotab -e`

```
...
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
*/5 * * * * /usr/lib/x2go/x2golimittime
```

The rest of remote Lab server setup is identical to local Lab server setup, e.g. adding Arduino IDE, installing USB camera and application like 'cheese' to capture images from USB camera.

Client / Student workstation setup (Windows and Linux)

Student setup requires installation of two packages only: X2GO client and specific set of files with custom SSH keys, X2GO session file and BATCH / SHELL script files.

In this way, single click or command is sufficient to initiate connection to remote Lab setup – only password needs to be entered before remote desktop connection, via Bastion server is established.

X2GO client installation

- Windows client setup: http://code.x2go.org/releases/X2GoClient_latest_mswin32-setup.exe
- Linux client setup: it is available in standard Ubuntu repositories, so following command is sufficient: `sudo apt-get install x2goclient`
- It is important to select XFCE as session type (since this was selected in X2GO server type)

Installation of custom files and login procedure is described in separate document “Manual for accessing Lab Setups over X2GO server” (in Serbian “UPUTSTVO ZA PRISTUPANJE LABORATORIJSKIM VEŽBAMA PREKO X2GO SERVERA”).

Custom package includes following files:

- Predefined X2GO sessions (for all 5 setups)
  - `cpa_sessions.txt (below is listed only for the 1st session i.e. setup)`

```
[20211118132048980]
applications=WWWBROWSER, MAILCLIENT, OFFICE, TERMINAL
autologin=true
clipboard=both
command=XFCE
defsndport=true
directrdp=false
directrdpsettings=
directxdmcp=false
directxdmcpsettings=
display=1
dpi=142
export=
fstunnel=true
fullscreen=false
height=600
host=127.0.0.1
icon=:/img/icons/128x128/x2gosession.png
iconvfrom=ISO8859-1
iconvto=UTF-8
key=
krbdelegation=false
krblogin=false
maxdim=false
multidisp=false
name=RelabMiniBricsCPA
pack=16m-jpeg
print=true
published=false
quality=9
rdpclient=rdesktop
rdpoptions=
rdpport=3389
rdpserver=
rootless=false
setdpi=true
sndport=4713
sound=true
soundsystem=pulse
soundtunnel=true
```

```
speed=2
sshport=22024
sshproxyautologin=true
sshproxyhost=147.91.209.60
sshproxykeyfile=id_rsa_weblabcpa.txt
sshproxykrblogin=false
sshproxyport=22
sshproxysamepass=false
sshproxysameuser=false
sshproxytype=SSH
sshproxyuser=weblabcpa
startsoundsystem=true
type=auto
useiconv=false
usekbd=true
user=cpa
usesshproxy=true
width=800
xdmcpclient=Xnest
xdmcpserver=127.0.0.1
xinerama=false

[20211118132048981]
applications=WWWBROWSER, MAILCLIENT, OFFICE, TERMINAL
autologin=true
clipboard=both
command=XFCE
defsndport=true
directrdp=false
directrdpsettings=
directxdmcp=false
directxdmcpsettings=
display=1
dpi=142
export=
fstunnel=true
fullscreen=false
height=600
host=127.0.0.1
icon=:/img/icons/128x128/x2gosession.png
iconvfrom=ISO8859-1
iconvto=UTF-8
key=
krbdelegation=false
krblogin=false
maxdim=false
multidisp=false
name=RelabMiniBricsCPA2
pack=16m-jpeg
print=true
...
```

- SSH public keys, different keys for each Lab setup (generated on Lab setup using ssh-keygen)
  - `id_rsa_weblabcpa2.txt`
  - `id_rsa_weblabcpa3.txt`
  - `id_rsa_weblabcpa4.txt`
  - `id_rsa_weblabcpa5.txt`
  - `id_rsa_weblabcpa.txt`
- Windows batch files for direct access to Lab Setup 1,2,..,5
  - `to_cpa.bat`
  - `to_cpa2.bat`
  - `to_cpa3.bat`
  - `to_cpa4.bat`
  - `to_cpa5.bat`
- Linux batch files for direct access to Lab Setup 1,2,..,5
  - `to_cpa.sh`
  - `to_cpa2.sh`
  - `to_cpa3.sh`
  - `to_cpa4.sh`
  - `to_cpa5.sh`

Connection is initiated from command prompt/terminal: to_cpa.bat (or to_cpa2.bat ...)
Only previously set Lab Setup password need to be provided at login, and Student will access Lab setup desktop.