Exploring the Wallaby

Leo Halbritter
tgm
Nothing To C Here
Vienna, Austria
lhalbritter@student.tgm.ac.at

Ferris Bartak
tgm
Nothing To C Here
Vienna, Austria
fbartak@student.tgm.ac.at

Julia Pöschl
tgm
Nothing To C Here
Vienna, Austria
jpoeschl@student.tgm.ac.at

Sarah Breit
tgm
Nothing To C Here
Vienna, Austria
sbreit@student.tgm.ac.at

Manuel Kisser
tgm
Nothing To C Here
Vienna, Austria
mkisser@student.tgm.ac.at

Fabio Fuchs
tgm
Nothing To C Here
Vienna, Austria
ffuchs@student.tgm.ac.at

***Abstract*—This paper shows, which possibilities you have by accessing the wallaby's operating system by terminal. The main aim is to show up a way to connect the wallaby to the internet and then create an automatic backup system. Although we figured out how to implement such a system, we discovered many other things on the controller which are worth to be mentioned. Therefore, we went from our initial approach, to 'Exploring the Wallaby'. This paper shows how you can modify sort of useful things on the wallaby controller, how to connect the wallaby to an access point and how to push your projects on to github with the help of a Raspberry Pi.**

***Keywords*—Network, Version, Backup, Scripting, Connection, Teamwork**

I. Introduction

We thought about the possibility of automatically backing up the code projects on the wallaby using a versioning software. So first we had to connect our wallaby with a network. We reconfigured one interface of the Raspberry Pi to behave like an access point. Another network interface is connected to the internet. Then we had to connect the wallaby with the previously created network access point – the difficult part, since the wallaby does not have most services used to automatically get a network connection. Once that was set up we had to configure the git-service on the Raspberry Pi, so that it takes every hour the code projects of the 'Default User" directory and pushes them onto a github repository.

II. Motivation

While working with the wallaby and its web interface, some issues occurred, which we tried to get rid of with the networking and backup system - the overall purpose of this paper. The most trivial problem with the network of the wallaby is, besides that it is not safe at all, that its range is not that big. In fact, a laptop has to be at most one meter away of the wallabies.

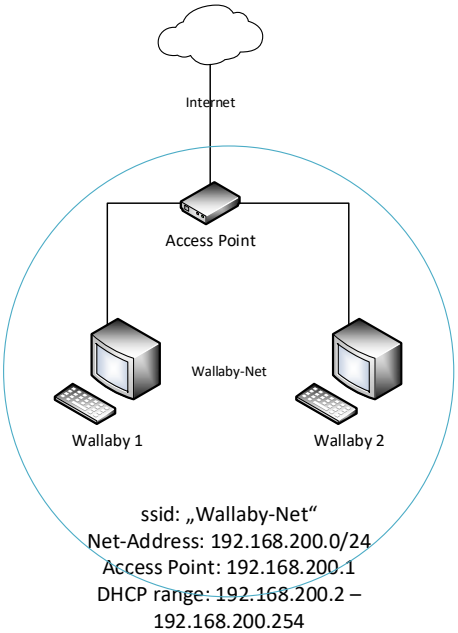
Additionally, if you move your laptop to near to another wallaby which has been connected to your laptop before, it can be possible that it connects to this wallaby without your knowledge. If you then compile some changes they will be sent to the false wallaby. And if you have two different versions of the same program on both wallabies, no backups and a huge amount of bad luck, you overwrite your newer version with the older one. After this happened to us, we recognized, that an automatic backup system is needed.

To get back to the scientific part, the first logical step to get a better connection is to use the cable network connection instead. Trouble here is, that the operating system we used (we all use windows 10) is not able to establish a connection to the wallabies. We tried to get the driver software several times, also by following the official tutorial on the homebase website[1], we could not accomplish it. After all we had to use different operating systems. But the range of the network is still not improved by this at all.

Our first try to improve the wireless network connection was to use a Raspberry Pi as a kind of network repeater. The Raspberry Pi was configured to connect with the native Wi-Fi of a wallaby and was mounted next to it on the robot. Then we accessed the Raspberry Pi by remote desktop and opened the web interface there. The advantage of this is that there is no need to reconfigure the wallaby controller. The cons are that each wallaby needs its own Raspberry Pi and it is completely inefficient. Furthermore, it isn't allowed to use additional controllers in Botball game.

And This leads us to our final solution.

III. Concept



1 Network plan of the 'wallaby net'

A. *Setting up the access point on the Raspberry Pi*

We started with the easy part of our goal, setting up one interface of the Raspberry Pi as an access point. We just connected the onboard network card with the internet and an external one (a WiFi USB-Stick) got configured so it creates an access point for the wallaby. Since it would be too unsecure to connect the wallaby with a public network – clients could find it – we decided to build our own network with the Raspberry Pi.

As an alternative way it is also possible to use any other device to create a wireless LAN network for the wallabies. For example, we tried out to use the built-in functionality of a windows laptop and an android smartphone to create a hotspot. It worked well, too. In fact, each device that has two network adapters is fine. In those cases, we have to connect our Raspberry Pi, which runs the backup service, to our LAN as a host or kind of server, too.

B. *Connecting the wallaby with the Raspberry Pi*

This is the tricky part since the wallaby does not come with all the necessary network configurations or any ability to change the network settings via GUI or web interface. For this we had to use the command line program `wpa_cli` with the service `wpa_supplicant` (Tutorial for this follows). While trying to get out this we decided to change our main topic.

C. *Using git via commandline/ssh*

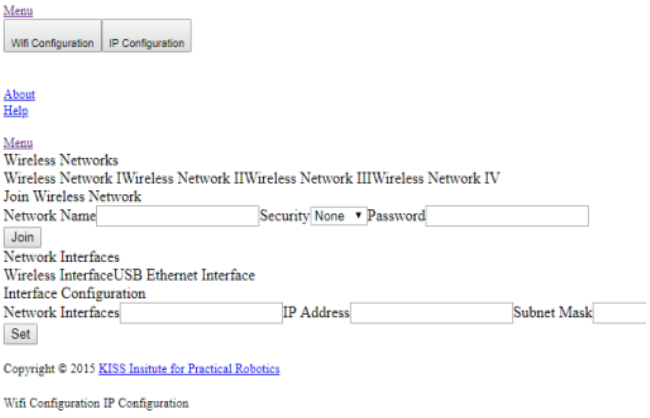
Once the wallaby established a connection with the Raspberry Pi the backup-system should automatically work by using git. The Raspberry Pi should save the sub-directories of the 'Default User'-directory locally and then push them onto a github repository.

IV. The Exploration

When we looked up how to change the network settings of the wallaby, we found many interesting things already located on the wallaby.

A. *Hidden pages of the web interface*

Previously we wanted to change the wallaby's website layout. To achieve that we downloaded the whole 'harrogate' directory where various '.jade' and '.php' files are located. We looked at the purpose of those files and found a few interesting things, which are nowhere to be found on the IDE right now. Like a website to let the wallaby join a network.



2 The network configuration page

Sadly, we couldn't use this site, because it had no implemented functionality which would do, what we expected the site to do.

B. *The WiFi-Configuration*

Close to what we wanted to achieve was a custom wifi-service. It is based on a python-script which creates the network of the wallaby. We got a hint of its existence by an previous paper of the team 'Almighty 7', 'Hacking the wallaby'[2]. The python-script '`wifi_configurator.py`' contains a lot of code, like this section, which appears to be related to our paper topic.

```
# ===== Start Wi-Fi =====

os.system('ifconfig wlan0 192.168.125.1')
time.sleep(1)
os.system('wpa_cli ter')
time.sleep(1)
os.system('hostapd /etc/hostapd_wallaby.conf &')

time.sleep(10)
os.system('wall -n \"Wallaby Name: ' + wallaby_id + '-$

#start DHCP
os.system('/usr/sbin/udhcpd /etc/udhcpd.conf')
```

3 wifi_configurator.py script

As the name states, it is used to configure the WiFi. The `os.system` lines are executing a linux command, while the `time.sleep` commands let the system wait a specific amount of time in seconds. With the first `os.system` the interface 'wlan0' is configured with the IP-address 192.168.125.1, '`wpa_cli ter`' terminates the running wpa-service and '`hostapd …`' starts the hoastapd-service (the same service is used on the Raspberry Pi to create an access point) in background. There are many new files to explore, like '`hostapd-wallaby.conf`'. We quickly looked through those files and found out that most of them are not useful for us, but the '`welcome.wav`' file piqued our curiosity.

C. *Playing with the Audio*

As you may or may not know the notification sound played when the wallaby's battery is about to die is a little bit irritating. For two seconds it loudly proclaims, 'Low voltage detected, please turn off the controller' and this can get very annoying. With our knowledge of where the '.wav' files are located, we chose to replace them with our own audio files. Now the 'Low voltage detected …' sound is a short trumpet chime, which we edited in school. We achieved this by replacing the '`turn_off_wallaby.wav`' file with our very own version.

We also found a '`wallaby.wav`' file which appears to not be used – or at least the wallaby never played before. We decided to use that instead of the normal 'Welcome' sound when the WiFi is turned on – or in our case, when the controller established a connection with the WiFi.

D. *Stopping the wifi-service*

One attempt of our attempts at changing the WiFi-settings of the wallaby was to stop all services which were related to networks or WiFi in general. With the command

```
ls -l /lib/systemd/system | grep .service$
```

we found all services and discovered 'wifi.service'. As it turned out, it was a custom service, which was the only option we could think of after we could not find it while researching its origin. Once we figured that out we immediately tried to dig up more. After going down the rabbit hole we were disappointed. It literally just executes the Python-script we mentioned earlier.

Next, we finally tried to stop all services. The command

```
systemctl stop *
```

looked like it would have been the one to accomplish that. But the results did not look like what we wanted to accomplish. Everything failed and in the end, it would have changed absolutely nothing, for the simple reason that the 'wifi.service' only runs the script, but if the script has been run before it is irrelevant if the service is stopped or not.

V. Connect the Wallaby to the created network (Tutorial)

This is a step-by-step tutorial on how to configure your wallaby, so it connects with the specified network. Keep in mind that you do need an access point for the wallaby to connect to. In this tutorial to activate the hostapd service we used a Raspberry Pi.

A. ***First step:*** *Getting access to the wallaby's terminal.*

There are many ways to do this. Choose the one that suits you the most. You can connect your computer via cable or the network. We do not recommend the latter, because as soon as you are done, the network will not be available anymore. You can use a SSH client like PuTTY and connect via SSH (ip-address is same as the one of the web interface). On the Ubuntu distribution of Linux this is achieved by typing `ssh root@<ip-addr of wallaby>` into your terminal. In our opinion the best method for this is to just connect a keyboard to the wallaby and make the terminal visible by clicking following buttons: 'Settings→Hide UI". The reason why we recommend this is because of it being a simpler way with less failure risk than other options. If the network were unreachable this option would still be valid. It also works through reboots, which close your SSH session.

B. ***Second step:*** *Comment out the whole 'wifi_configurator.py"-file.*

Open the file `/usr/bin/wifi_configurator.py` with your favorite text editing tool.

IMPORTANT! Backup files and code before changing or deleting any of them! Some might be critical for the wallaby'sunimpeded functionality! If you are not able to fix your mistakes you would have to reinstall the wallaby and start from scratch. You can also download the newest firmware of the wallaby and if something critical happens, just follow the tutorial on the kipr-website[3].

Instead of commenting out single lines with the '#" character you might want to use three quote signs instead. With that you can comment out whole blocks of code, which frees up a lot of time that could be spent elsewhere. To close a comment block just write three more quote signs. In our case we just placed three quotes at the beginning and at the end of the file. When you are done it should resemble something like this:

```
"""
<code of the file>
...
...
...
"""
```

Then you effectively prevent the wallaby's default network creation on start up.

C. ***Third step:*** *Apply the new configuration.*

If you have worked with a remote SSH connection you now have to plug-in a keyboard. Once you have done that type in 'sudo reboot". This will reboot your wallaby and after it booted completely you will notice, that the wallaby's status LED is not going shine with a blue light. This is because the wallaby will not have created a WiFi network. At the moment you are not able to access the wallaby's web interface but that won't be a problem for long.

D. ***Fourth step:*** *Connect the wallaby with the Raspberry Pi's network.*

Now comes the tricky part. On a normal Linux distribution you would use something similar to `iwconfig <if> essid <ssid> key s:<psk>` but if you use this command you will receive an error message in response. Apparently, this command is not supported. After a long time of looking for another way we found out that the network is configured with 'wpa_cli" which is using the 'wpa_supplicant"-service. Following this discovery we tried to figured out how to make our own wpa_cli configuration script:

```
#!/bin/bash
#Client-configuration
wpa_cli ter
killall hostapd
wpa_supplicant -B -I wlan0 -c
/etc/wpa_supplicant/wpa_supplicant-
wlan0.conf
sleep 3s
#old network is deleted, new one added
wpa_cli remove_network 0
wpa_cli add_network
sleep 3s
#Setting the SSID and PSK
wpa_cli set_network 0 ssid '"Wallaby-
Net"'
wpa_cli set_network 0 psk
'"SomePasskey"'
sleep 3s
#Enable the config and save it
wpa_cli enable_network 0
wpa_cli save
sleep 3s
#new status is printed
echo `wpa_cli status`
sleep 3s
```

4 Script to connect to a network

This script is inspired by the code of the github repository 'Wallaby-Communication-Library"[4].

This script can be inserted into the terminal as it is but we created it with one requirement in mind. It should be an executable script as it is easier to debug and faster to initiate. Let us begin with the very first command `wpa_cli ter`. It terminates the 'wpa_supplicant"-service that is running right now. The second line, `killall hostapd` kills all the hostapd processes. Actually this is only necessary, if you want to keep the wifi_configurator-file pristine and not comment out most of it. The next line, `wpa_supplicant -B -I -wlan0 -c /etc/wpa_supplicant/wpa_supplicant-wlan0.conf` is starting a 'wpa_supplicant"-service on the interface 'wlan0" and sets its configuration file to 'wpa_supplicant-wlan0.conf" in the directory '/etc/wpa_supplicant/". The next line simply lets the system wait 3 seconds which gives the system more time to finish the previous process. Next you have to remove the existing network 0 with the `wpa_cli remove_network 0` command. This is the network the system previously ran on. With

`wpa_cli add_network 0` we add a new network 0. Afterwards we wait another three seconds. Next you have to tell the network which SSID and passkey to use. This is done with the commands: `wpa_cli set_network 0 ssid'"<SSID>"'` `wpa_cliset_network 0 psk '"<passkey>"'` And now we wait an additional three seconds. At last we enable the created network with `wpa_cli enable_network 0` and save the configuration to the file system via `wpa_cli save` and we are done. The last few lines are just for debuging and serve no additional function. If you have done everything correctly the wallaby's status LED should be shining with a blue light and your wallaby should get an IP address on the network interface 'wlan0". To confirm the latter simply type in 'ipaddr". After your computer is connected with the wallaby and you looked up its IP address they should be on the same network you can now access the web interface by using the new address.

VI. Setting up the backup system on the Raspberry Pi (Tutorial)

You will have to follow this tutorial to be able to use our script. This only must be done one time. (We assume you have already connected your Wallaby with a RaspberryPI which also is connected to the Internet, if not follow the steps of the point above).

Replace all the URLs, paths and IP addresses with your own!

At first you will have to install git and create a repository on the versioning service of your choice. We used github for simplicity.

Next up is the execution of the script. You must execute the script on a Raspberry Pi. Clone your repository to the folder where the local Backups should be saved. E.g. `git clone https://github.com/sbreit/Raspberry.git /home/pi/WallabyBackup` To improve the quality of life while using git you should use the following command: `git config credential.helper store`. Its purpose is the storage of your username and password, so that you do not have to enter them every single time you want to backup. The script is the following:

```
#!/bin/bash

#Copy all relevant files from the
wallaby #to the Raspberry
scp -r
root@<ip>:/home/root/Documents/KISS/Defa
ult\ User/* <dir on raspberry>

#Pull the Repository
git pull <repository> --rebase

#Add all files to the local repositories
git add -A

#Commit all files that were changed
git commit -m '<msg for commit>"

#Push the repository
git push <repository> master
```

5 Script to backup your code projects

To make it work automatically add the script to the crontab.

Note: If you simply hardcode the ip address in the script like we did, you should use a DHCP-Server with reserved IP addresses for your network devices.

VII. Conclusion

There are many ways to improve this solution. One thing that could be added, is a system, that automatically checks for new commits on your repository and puts them on the wallaby. For this the commands `git fetch origin` or `git clone` could be useful. Since we had not enough time to look deeper into this, we cannot show you a script.

Furthermore, there is a possibility to switch the network card of the wallaby, so the connection is more stable.

If you want to change the network the wallaby connects too, you still would have to go into the terminal and change the setting manually or change our script and run it again. An useful extension to the wallaby's GUI would therefore be to add more functionality to the network section.

To improve the security of the wallaby you could add a password for the root-user, so not everyone on the same network can access the wallaby.

To conclude this paper, we can say, that it is not easy to accomplish a connection to a network with the wallaby, if you haven't done it before, since it is not set up to do that. This method of connecting the wallaby to a network is still not perfect, because it lacks of a stable connection, but to accomplish that you would need to switch the hardware.

Anyways, this paper shows a good method to back up all your code of the wallaby, without copy and pasting it and then connect to another network just to upload it. Our team uses this method now and it really is much easier to work like this, than before. Not only is the network's range much more, than the wallaby's network range, it also makes working as a team easier.

[1] KIPR, 'Connecting to Windows with a USB Cable' http://homebase.kipr.org/team/mod/resource/view.php?id=395, last visit: 08.04.2018

[2] Almighty 7, 'Hacking the wallaby' http://webspace.pria.at/ecer2016/papers/paper_16-0385.pdf, last visit: 08.04.2018

[3] KIPR, 'Wallaby firmware' http://www.kipr.org/kiss-web-wallaby-firmware, last visit: 09.04.2018

[4] LACT-Botball-0636, 'Wallaby Communication Library' https://github.com/LACT-Botball-0636/Wallaby-Communication-Library, last visit: 08.04.2018