Laboratory guide

(To be updated during the semester)

Version: 2022.03.16

General description of the laboratory

In the laboratory, you will be using workstations running Arch Linux. Please select the Linux system in the bootloader when turning on the computer.

Your home directory on these computers is located on a network drive and uses the SAMBA (CIFS) file system. This filesystem is not Buildroot compatible (does not store correctly certain attributes critical for BR operation). Therefore, please remember not to unpack BR on it, nor copy the BR directory onto it.

You can, however, store archives on it (preferably compressed „tar") containing the contents of the BR directory ("tar" archives store internally full information about file attributes, symbolic links, etc.).

To enable you to compile BR on your workstations, a "/malina" directory is created on them. Please create your subdirectory there (preferably with a name related to your login). In this subdirectory (or its deeper subdirectory), you will be able to unpack BR and compile it.

Caution! Certain Buildroot packages do not accept too long paths for the BR directory. Please avoid using too long directory names.

Similarly, you can place the CCACHE catalog (described below) in this catalog.

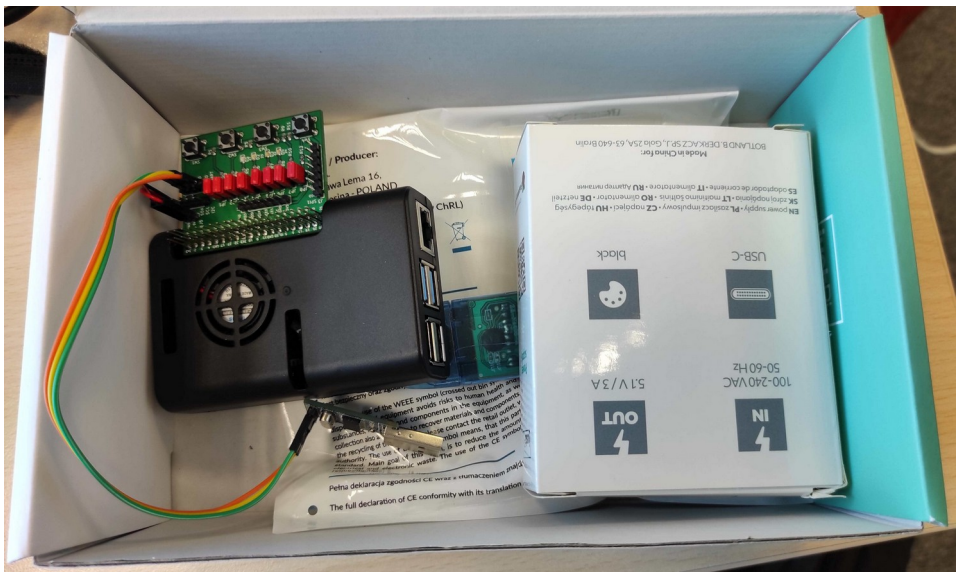
Caution! The contents of the /malina directory may be removed without warning between classes (or on request of the person currently working on a given machine during classes). The reason is that this directory has a limited capacity and is shared by all users of the computer.

Storing the laboratory set

To avoid unnecessary waste of time and unnecessary wear of components, please place the set in a box in a box without disconnecting them as in the photo below.

However, please disconnect the AC adapter and pack it in a separate box.

Similarly, if any additional modules (not UART/USB adapter) have been connected, please disconnect them and put them in the appropriate box before packing the kit.



Connecting RPi to the workstation

In order not to damage the RPi computer or the UART/USB adapter, please follow the instructions below.

1. Before touching the RPi or the connected module (or adapter), ground yourself by touching the PC (workstation).
2. Any changes to the connections should be made with the RPi power supply disconnected and the UART/USB adapter disconnected from the PC.
3. Before starting work, please make sure that the USB/UART adapter is correctly connected (GND ↔ GND, RXD ↔ TXD, TXD ↔ RXD). It may have been accidentally incorrectly connected by the person packing the kit
4. To connect the adapter to a PC, please use a USB extension cable.
5. Please always connect the USB/UART adapter to the PC first, and only then connect the RPi power supply to a power outlet.
6. Power cycling the RPi should be done by inserting/removing the power supply to/from the mains socket (extension cord). The idea is to minimize the wear of the RPi's USB-C socket.

Preparation of the Buildroot environment

Caution! In the meantime the LTS (long term support) 2022.02 version of BR has been released. We will use that version in the lab!

- Download the environment:
  $ wget https://buildroot.org/downloads/buildroot-2022.02.tar.xz
- Unpacking:

  $ tar -xJf buildroot-2022.02.tar.xz

- Preconfiguration:

  $ cd buildroot-2022.02

  $ make raspberrypi4_64_defconfig

- **Important settings:**
  - „System configuration/Run a getty.../TTY port” : make sure that it is set to „console” (in older BR it had to be set to „ttyAMA0”).
  - „Build options/Mirrors and Download locations/Primary download site”: set to „**http://192.168.137.24/dl**”
  - In Toolchain/Toolchain type – select „External toolchain”. The default version „Arm AArch64 2021.07” should work correctly.
  - Please switch on the CCACHE in Build options/Enable compiler cache” to further speed up the compilation. The ccache directory should be modified with the BR2_CCACHE_DIR option („Compiler cache location”). It should be located in your subdirectory in „/malina”. The suggested name is: „/malina/user_subdirectory/ccache-br”. The tool used to clean the „/malina” directory will TRY to preserve that directory (if it is not too big).
  - Please remember to set appropriately the format of the generated system image in „Filesystem images” (usually „initial RAM filesystem linked into linux kernel” should be the right choice, but leave ext2/3/4 selected to avoid annoying error messages. You may need to increase the size of the ext2 image though).
    Another possibility for generation of the kernel with Initramfs is disabling the creation of ext2/3/4 image and disabling the „Custom scripts to run after creating filesystem images” in „System configuration”.
    When creating the Initramfs-enabled image, please switch on the gzip image compression („Compression method (gzip)”) for the „cpio” archive.
    Caution! The script called after compilation that builds the image of the SD card may have a problem with insufficient capacity of the VFAT partition.

Specific „features” of Buildroot-2022.02

You may experience certain problems with the final step of the image building. The Buildroot environment for Raspberry Pi at the end executes the script to build the image for the SD card. This script assumes that the **ext4** file system image was previously created.
If we generate only the **Initramfs** image, then at the end we get an ugly error message. Probably we can ignore it, because all important files are created before that error, but learning to ignore errors is not a good method. There are to solutions possible. The first one is to leave enabled the creation of the ext4 file system image. However the default size of the ext4 image (60MB) may be too small for more complex setups. Please increase it if you get errors related to too small ext4 filesystem.

There may also be a problem with the insufficient size of the VFAT partition (by default it is only 32MB!). This problem requires that you manually correct the size of this partition in the file "board/raspberrypi4-64/genimage-raspberrypi4-64.cfg". We should find a fragment there:

```
image boot.vfat {
      vfat {
            files = {
                  "bcm2711-rpi-4-b.dtb",
                  "rpi-firmware/cmdline.txt",
                  "rpi-firmware/config.txt",
```

```
                "rpi-firmware/fixup4.dat",
                "rpi-firmware/start4.elf",
                "rpi-firmware/overlays",
                "Image"
            }
        }

        size = 32M
}
```

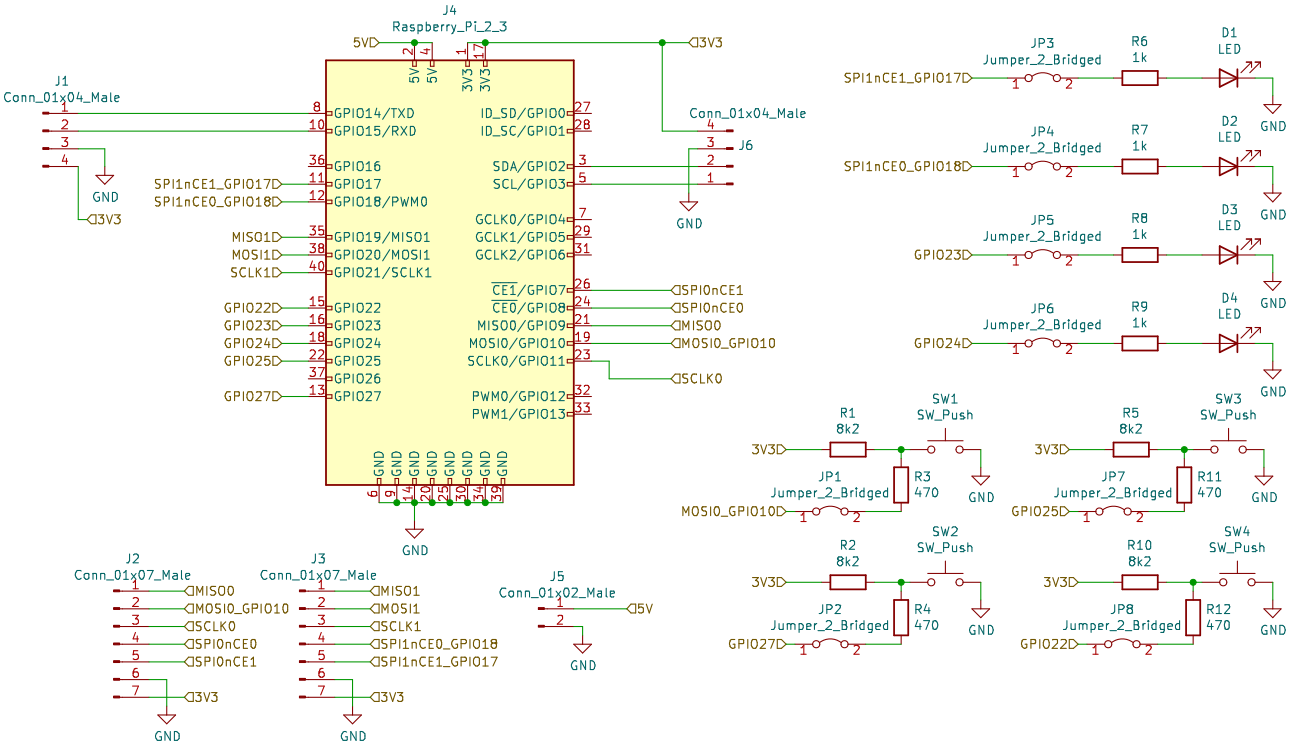
and change the size from 32M to 128M or 256M.

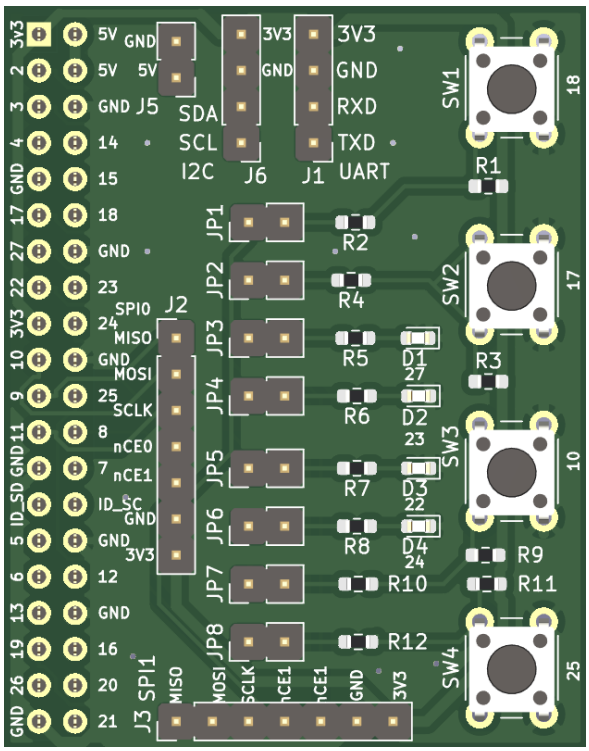
The second solution is to disable the „Custom scripts to run after creating filesystem images” in „System configuration”. Then the SD card image won’t be created.

Caution! The Debian/Linux system often uses, the "dash" interpreter as the system shell (check what "ls -lr /bin/sh" returns). In leads to compile-time errors in BR: 'rm: refusing to remove'. ' or '..' directory: skipping '(BRPATH)/output/build/buildroot-fs/ext2/target/run/..' ”. In Debian, you should then run "dpkg-reconfigure dash" and disallow the use of dash as the default shell. If that doesn't help, you can manually correct the symbolic link "/bin/sh" (please make sure that bash is installed beforehand!)

Usage of the extension board

The laboratory kit is supplemented with an expansion board, the design of which is available in the public repository https://gitlab.com/WZab/wz_edu_boards . The following pictures show the schematic diagram and top view of this board.





The board contains four LEDs of different colors, connected via current-limiting resistors to the GPIO pins (GPIO number is given next to each LED). Setting the given output to the output mode and switching on the high logic level causes the LED to light up. In addition, the board contains four switches, connected via additional resistors (protecting the RPi against damage and setting the high state when the button is not pressed) to the GPIO pins (numbers are given under the buttons). Pressing the button forces a low state on the given pin, working as an input.

The JP1 - JP8 jumpers (normally placed) allow you to disconnect the LED or the button from the GPIO pin, when it is to be used for another purpose.

The board provides a UART serial interface connector to which we should connect the USB/UART converter. The connections should be made before connecting the RPi to the power supply and the converter to the USB socket. We connect the ground pins (GND) in the UART connector of the board and in the converter. The RXD and TXD pins should be connected "crosswise". That is, the RXD of one board is connected to the TXD of the other. We leave the power contacts (3V3, 3.3V, 5V, Vcc etc.) unconnected.

In addition, the board provides connectors for I2C and SPI interfaces (SPI0 and SPI1). Their use should be consulted with the teacher.

Basic usage of minicom

- Start minicom with:
  $ minicom -D /dev/ttyUSB0 -o
- It is important to switch off the „hardware flow control” in the „serial port settings”
- The transmission speed should be set to 115200

Mirror server

The mirror server has been created for our lab, which stores the source packages needed by the Buildroot environment.

Server's IP address: **192.168.137.24**

The URL of sources repository to be written into „Build options/Mirrors and Download locations/Primary download site" in Buildroot: **http://192.168.137.24/dl**

The rescue system

In the lab, you do not have access to the administrator account on your workstations. In addition, the quality of SD card connectors in the Raspberry Pi may result in a malfunction due to frequent removal and insertion. Therefore, the cards have been configured to allow almost full management of their content from the Raspberry Pi.

Normally, the user's system starts up when the power is turned on.
The kernel, the device tree and the "cmdline.txt" file are taken from the "user" directory of the VFAT partition on the SD card.

If you keep the SW4 button (connected to GPIO 25) pressed when you turn on the power of the Raspberry Pi, the rescue system will be loaded. The kernel, device tree, and the "cmdline.txt" file are taken from the "rescue" directory of the VFAT partition on the SD card.

Initially, both of these directories contain the rescue system.
In both cases, cmdline.txt contains:

```
root=/dev/mmcblk0p2 rootwait console=tty1 console=ttyAMA0,115200
```

This line defines that the root filesystem should be mounted from the second partition of the SD card. However, if the kernel is compiled with initial ramdisk the "root = ..." option is ignored.

Initially, the SD card contains a small second partition that does not contain any file system.
If you want to create a user system compiled with ext4 as the root filesystem, you have to enlarge this partition (eg to the maximum size) with the fdisk program. ***Caution! Fdisk must be run with the "-u" option.***

In such a situation, of course, we still need to install a filesystem on this partition.
We can do this with the dd program, downloading the file system image over the network.

In the BR/output/images directory, we can run an ad hoc http server:

```
$ python3 -m http.server
```

It will listen on port 8000 and serve files from that directory.
We can then get the filesystem image by saving it immediately to the target partition:

```
# wget  -O - 192.168.145.xxx:8000/rootfs.ext4 | dd of=/dev/mmcblk0p2 bs=4096
```

Of course, we have to replace 192.168.145.xxx with the IP number of our workstation.

The rescue system provides tools to partition and format the SD card (**mkfs ..., fdisk**; please be careful not to destroy partition 1 which contains files essential for booting the system; *advanced users can store these files in ramdisk and reconstruct this partition*).
There are also tools to repair the filesystems on the card (f**sck ...**).

The rescue system is equipped with an SSH (dropbear) server, but it accepts the incoming connections only after the superuser password has been set. We should do this with the **passwd** command.
We can then copy the new system image, e.g. by:
`scp Image` root@192.168.143.xxx`:/tmp/d/user` (We assume here that our board got the IP address 192.168.143.xxx, and that the first partition of the SD card was mounted in the directory `/tmp/d`, by: "`mkdir /tmp/d; mount /dev/mmcblk0p1 /tmp/d`").

Regeneration of the SD card

If the data on the SD card is corrupted in a way that prevents the system from booting, you can restore the card. You should start the RPi from another card and boot the rescue system. After that you may replace the card with the "damaged" one. Then log in to RPi and run the command:

```
# wget -O - http://192.168.137.24/sdmini.img | dd of=/dev/mmcblk0 bs=4096
```

After successfully writing the system to the card, you can enter the command "reboot" and make sure that the system properly boots from the recreated card.

Reset Linux by minicom

If the Linux on the Raspberry Pi is started, but you can't control it (eg. no login prompt is displayed or you don't know the password), you can try to reset it with a special minicom command: "CTRL + A, F, B". Please note, that this is an emergency solution (like pressing the RESET key in the PC. It may cause corruption of the mounted filesystems).