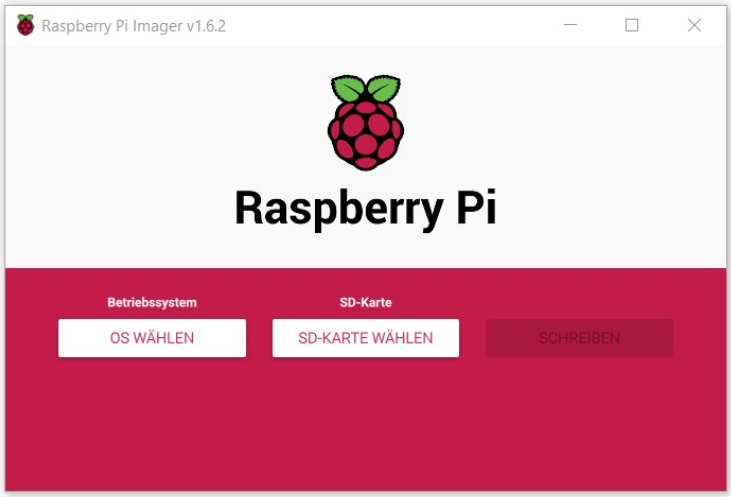
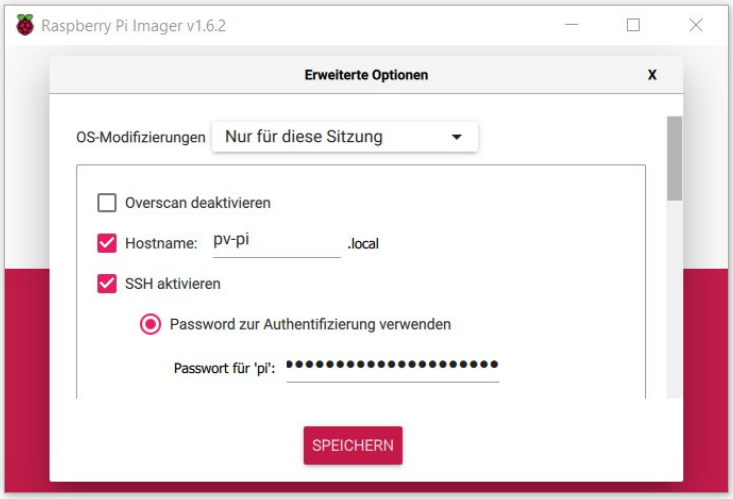
1. Wiring

2. Set Up a Base Raspberry Pi Image

- Download and install the Raspberry Pi imager from https://www.raspberrypi.com/software/
    - Note: Raspberry Pi Imager v1.6.2 was used in this manual.
- Insert the Micro SD Card into the computer.
    - Note: a SanDisk Extreme 64 GB Micro SDXC card was used in this manual.
- Start the Raspberry Pi Imager to setup the Raspberry Pi Image.



- Choose OS: Raspberry Pi OS (32-bit).
    - Note: Debian Bullseye 2021-10-30 was the default image at time of writing.
- Select SD card: Choose your SD card volume.
- Press Ctrl+Shift+X
- A popup comes up, allowing you to change some options for your raspberry pi.



- Make the following changes:
    - Choose a hostname for your pi. In this manual, the hostname “pv-pi” is used.
    - Activate SSH.

        - Choose your authentication method. Password authentication was used in this case. Do not use the standard credentials (user: pi; password: raspberry), because that will trigger a warning on every startup.
    - Setup Wi-Fi: Enter The SSID and the password for the Wi-Fi in your test-environment.
    - Choose your appropriate country, language and keyboard settings.
- Select “Save”.
- Back in the main screen, click on “write” now and confirm.
- The Raspberry Pi image is now written to the micro SD card. Windows might offer you to reformat the SD card after the imager is finished. Do not do that.
- After the image is written and verified, you can remove the SD card.

3. First Boot and Force HDMI

- Disconnect the display. Insert the SD card into the Raspberry Pi and power up the Pi.
- Open your SSH tool on PC and connect to your raspberry pi by using the hostname you previously defined. Enter your raspberry pi user credentials.
- Open up the boot config.txt

```
$ sudo nano /boot/config.txt
```

- Uncomment the following line and then save the file:

```
hdmi_force_hotplug=1
```

- reboot the Raspberry Pi, connect the display and repower the Pi. You should now get a picture on your display. Enabling force HDMI averts a possible startup race between the raspberry pi and the display. Normally, the display would need to be powered on before the raspberry pi.

```
$ sudo reboot
```

4. Mount your Storage device

- Switch to root user.

```
$ sudo su root
```

- The control unit stores measurement data on a separate storage device, such as an SSD. To be able to use your storage device it needs to be mounted. First, install some filesystem drivers:

```
$ apt-get -y install ntfs-3g hfsutils hfsprogs exfat-fuse
```

- Now we create the directory to mount the device to. Use the /mnt/ssd directory for this:

```
$ mkdir /mnt/ssd
```

- Use blkid to find the device id and path you want to mount.

```
$ blkid -o list -w /dev/null
```

```
pi@pv-pi:~ $ sudo blkid -o list -w /dev/null
device                          fs_type     label       mount point                 UUID
------------------------------------------------------------------------------------------------------------------------------------
/dev/mmcblk0p1                  vfat        boot        /boot                       E183-6233
/dev/mmcblk0p2                  ext4        rootfs      /                           1232a209-2596-48f0-a078-731d10b918ad
/dev/sda1                       vfat                    /media/pi/64B9-2277         64B9-2277
```

- In this case, the device I want to mount is /dev/sda1. Now mount the device to /mnt/ssd. Adjust the following command to fit your setup.

```
$ mount -t vfat -o utf8,uid=root,gid=root,noatime /dev/sda1 /mnt/ssd
```

- Create an entry in fstab to automatically mount the device on boot.

```
$ nano -w /etc/fstab
```

- Add the following line with the uuid from above: (adjust according to your setup: uuid and file system)

```
UUID=64B9-2277 /mnt/ssd/ vfat utf8,uid=root,gid=root,noatime 0
```

- More detailed information on mounting devices: https://jankarres.de/2013/01/raspberry-pi-usb-stick-und-usb-festplatte-einbinden/

5. Set Up SSH keys and Git

- Make sure you are acting as root user

```
$ sudo su root
```

- Generate SSH keys for the root user. Leave the default location for the key and use a passphrase if you want. In this manual, a passphrase is not used.

```
$ ssh-keygen
```

- Output the public ssh key onto the screen. The public key is not a secret. It will be used to grant your raspberry pi rights to clone the SunshadeCorp git repositories.

```
$ cat ~/.ssh/id_rsa.pub
```

- In your web browser, visit the ssh key configuration page of your github account. https://github.com/settings/keys
- Click “New SSH key”. Choose a name for this key. Paste the output of the cat command into the key field and save. Your raspberry pi is now able to access the SunshadeCorp repositories.

6. Install EasyBMS-master

- Make sure you are acting as root user.

```
$ sudo su root
```

- Clone the control-pi-docker repository to the /docker directory.

```
$ git clone git@github.com:SunshadeCorp/control-pi-docker.git /docker
$ cd /docker
```

- If there is a specific branch you want to use, then get the branch using git checkout. In this case, the branch I want to use is called pv-hornbostel.

```
$ git checkout pv-hornbostel
```

- Now you need to create the configuration file that contains the mqtt credentials for your setup. Copy the example credentials file to credentials.yaml and edit the contents of the file before you continue. In this example, empty strings are used for username and password.

```
$ cp credentials.example.yaml credentials.yaml
$ nano credentials.yaml
```

    - Note: If you want to use an empty string in your credentials file, use double single quotes (‘’)
- The same MQTT credentials need to be passed to mosquitto and modbus4mqtt. For them, the credentials are passed as docker-compose environment variables. Create the .env file for docker-compose.

```
$ touch /docker/.env && nano /docker/.env
```

- Enter the MQTT credentials:

```
MQTT_USER=""
MQTT_PASSWORD=""
```

- Execute the install script. The install script downloads and installs docker and docker-compose. Also it clones the rest of the SunshadeCorp repositories into its appropriate directories.

```
$ ./install.sh
```

- If you want to use a specific branch in any of the sub repositories, then now go check these branches out inside the build directory.
- Edit the slave mapping for the BMS master according to your configuration

```
$ cd /docker/build/easybms-master
$ cp slave_mapping.example.yaml slave_mapping.yaml
```

```
$ nano slave_mapping.yaml
```

    - TODO: Explain this in its own chapter
- You can now start the EasyBMS-master and its services with docker-compose. The first startup will take much longer because the containers are being downloaded or built. This might take a while. Remove the –d flag if you want to see the docker-compose log as the services are starting up.

```
$ cd /docker && docker-compose up -d
```

7. Configure the BMS slaves

8. Set Up Remote Access via VPN

9. Configure EasyBMS-master

10. Notes

- In order to be able to pull branches, you may need to specify the default pull behavior. You can do this by typing:

```
$ git config pull.rebase false
```