Practical Course on Computer Vision and Robotics - Basic Setup Instructions

Julian Kerl and Lennart Jahn

October 2025

1 Getting started

This chapter concerns hardware and software matters to get started. It covers information from the ev3 to the raspberry pi and how to interface with your PC.

The raspberry pi does all the computation for the robot. The Lego ev3 is used only as a motor and sensor controller. You shouldn't write software for running on the ev3. Your PC is used to connect the the raspberry pi and program the pi over SSH.

Username: `pi`
Password: `raspberry`
Host Name: `raspberrypiX` where X is your pi's number

1.1 Connecting the hardware

The motors plug into the `A, B, C, D` ports of the ev3. The sensors plug into the `1, 2, 3, 4` ports of the ev3.

The lineleader sensor and the IMU (accelerometer, compass and gyro) sensor are currently not supported. If someone wants to use these they can implement a sensor class for this and then please share it with the group and make a pull-request on github :).

The ev3 has a battery pack that connects directly into the ev3. This is charged via the 10v power brick and connects via a barrel plug. The raspberry pi is powered over usb-c with the usb-c power brick. Use the usb-c power bank to power the raspberry pi when using the robot wirelessly. Connect the ev3 to the raspberry pi using the usb-A to usb-B cable. The usb-A goes into the raspberry pi and the usb-B goes into the ev3. The ev3 acts as the slave and the raspberry pi acts as the master. The micro SD card is loaded with raspberry OS with some extra software from us and it should be plugged into the raspberry pi. The ev3 runs its built in firmware and doesn't need a micro SD card.

For debugging we recommend running the ev3 via the charging cable. With the ev3 battery, the motors will only perform at their full performance for 10 to 20 minutes. The raspberry pi can run for 6 hours with a full charge of the battery bank.

The webcam connects to the raspberry pi.

1.2 Raspberry Pi

Specification: Raspberry Pi 4, 4GB running Raspberry Pi OS, 64bit with some additional software installed to get started quickly.

Useful utilities that have been installed on top of Raspberry Pi OS:

- Tailscale with Headscale server is used to easily SSH into the the pi from any network.
- `/etc/NetworkManager/system-connections/` contains a template for connecting to home WiFi networks.
- Python with automatically activated `ev3-env` virtual environment with the basic needed packages.

- libusb and udev rules for connection to the ev3.
- Message of the day is in `/etc/profile.d/motd.sh`

Warning! Don't press button on battery pack. This will stop the power to the raspberry.

To shutdown the Raspberry Pi, run:

```
$ sudo poweroff
```

After ten green flashes, the device is safe to disconnect from power.

To reboot run:

```
$ sudo reboot
```

1.2.1 Connect the Raspberry Pi to Wifi

The WiFi network configuration for the Raspberry Pi is written in the folder `/etc/NetworkManager/system-connections/` and contains one config file per network.

The contents of the configuration file for a standard home network look like this:

```
[connection]
id=DESCRIPTIVE-NAME
type=wifi
interface-name=wlan0
autoconnect=true
autoconnect-priority=20

[wifi]
mode=infrastructure
ssid=YOUR_WIFI_SSID

[wifi-security]
auth-alg=open
key-mgmt=wpa-psk
psk=YOUR-WIFI-PASSWORD-HASH

[ipv4]
method=auto

[ipv6]
addr-gen-mode=default
method=auto

[proxy]
```

Replace the CAPITAL-LETTERS with the correct values, rename the file file to `DESCRIPTIVE-NAME.nmconnection` with owner root and permission 600, reload the config with

```
$ sudo nmcli con reload
```

and your new connection is set up.

The psk is a hashed version of the password and can be generated as follows:

```
$ wpa_passphrase amazing-home-wifi YOUR-PASSWORD
```

See the man page for details.

To prioritize which network should be connected if multiple are available, you can raise or lower the autoconnect priority (higher number means higher priority) or even disable autoconnect completely.

1.2.2 Tailscale

Tailscale is used to create a tunnel between your PC and the Raspberry Pi. This is needed so that you can connect to it when you are using eduroam or hotspots. It actually lets you connect to it from anywhere. We use a self hosted instance of Headscale, a open source re-implementation of the closed source Tailscale server. We do not process any personal information and you do not have to make an account with a third party. No network traffic leaves the university network, unless you take the Raspberrypi out of it.

If you don't want to use Tailscale, you don't have to, however then you can't use the raspberry pi with eduroam unless you find another solution.

Tailscale comes pre-configured on the Raspberrypi.

On your PC install Tailscale according to the instructions for your system type. Ask Julian for a key with correct access rights and join the Tailscale network by running

```
$ tailscale up --login-server https://headscale.physik3.uni-goettingen.de
  --authkey <key>
```

We will provide you with the authentication key. You can access your Raspberry from the IP shown by

```
$ tailscale status
```

1.2.3 Connecting to the Raspberry Pi from your PC

Create a file `~/.ssh/config` and append the following to it (change the IP address as appropriate):

```
Host raspberry
    HostName 192.168.0.102 # <-- whatever your raspberry ip is
    User pi
    ForwardX11Trusted yes
    ForwardX11 yes
```

Now in your terminal you should be able to SSH into the raspberry by doing

```
$ ssh raspberry
```

The default username is `pi` and the default password is `raspberry`.

To enable passwordless ssh access to your raspberry pi, you can use a public-private key pair. On your computer first create a new ssh key (unless you have one already):

```
$ ssh-keygen -t ed25519 -C "YOUR.USERNAME@stud.uni-goettingen.de"
```

Copy your ssh public key to the `~/.ssh/authorized_keys` file on the Raspberry Pi:

```
$ ssh-copy-id -i ~/.ssh/id_ed25519.pub raspberry
```

For Windows the process is very similar, but you might have to install openSSH client first. See this guide for more information. Note: you don't need the Windows OpenSSH server.

1.2.4 Securing your Raspberry Pi

Secure your raspberry pi by disabling password logins. Open the `sshd_config` file:

```
$ sudo nano /etc/ssh/sshd_config
```

and change PasswordAuthentication from yes to no:

```
PasswordAuthentication no
```

For easy support, the tailscale access control allows course tutors to connect to your systems. However, tailscale does not provide some magic way to break into a system, one still has to have an ordinary

login on the device, be it via password or SSH-key. So to disable tutor access, you can just comment out the corresponding SSH-keys in `~/.ssh/authorized_keys`.

1.2.5 Setting up an SSH key Pair on the Raspberry Pi

It is useful to also have an SSH key pair on the Raspberry Pi so that you can pull and push your work from the Raspberry Pi to git. On the Raspberry Pi, do the following:

```
$ ssh-keygen -t ed25519
```

Make sure you use a passphrase!

1.2.6 Pushing to gitlab

The Raspberry Pi has an own gitlab account, but its configuration is inaccessible to you. You can give your SSH public key to the course tutors. They will add it to the Raspberry Pi gitlab accout and accept an invitation to your groups' gitlab repository. Then, the Raspberry Pi can push and pull from the repository without the need for storing personal keys or secrets on the device.

Create a file `~/.ssh/config` and add:

```
Host gitlab.gwdg.de
    User git
    IdentityFile ~/.ssh/id_ed25519
```

Now gitlab authentication should automatically work using the key pair.

If you still want to keep individual contributions cleanly visible in the commit history, we have included `git-su` (git switch user) to conveniently manage and switch multiple git identities (just user names and e-mail-addresses, not SSH keys [unnecessary using above setup]). Call the programm using `git-su`, the interface is self explanatory.

2 Programming on your Windows/Mac/Linux PC

Prerequisites:

1. Install Python. You can install Python directly via `apt` on debian systems, see here. On Mac and Windows you can use the installer from the Python website. Or you can use Anaconda.
2. `pip install opencv-contrib-python Pillow numpy imagezmq jsonpickle pyqt5`

3 Programming on the Raspberry Pi

Install VSCode. On Linux I would suggest installing via the apt-get package. Once in VS Code, install the remote SSH extension and access the raspberry via the status bar item in the far left corner. When accessing the raspberry pi via VS Code, you can install the python extension.

Open VS Code and connect to the raspberry pi. Go to Terminal → New Terminal, to open a terminal. The raspberry pi has python installed. Important packages that we installed for you:

- `ev3-dc`, for communicating with the Lego EV3.
- `imagezmq`, for sending images and messages to your PC.
- `jsonpickle`, for converting JSON to string for message sending.
- `numpy`, `scipy` for maths.
- `opencv-contrib-python`, for image manipulation.

3.1 The ev3-python3 Package

We use the ev3-python3 package to control the ev3 from the raspberry pi. The documentation can be found here.

The ev3-python3 package is already installed and has been done so via `pip install ev3_dc`. Here is some example code to check is motors and sensors are working correctly:

To control two motors plugged into port `A` and port `D` you can do:

```
import ev3_dc as ev3

with ev3.EV3(protocol=ev3.USB) as my_ev3:

    motor_a = ev3.Motor(ev3.PORT_A, ev3_obj=my_ev3)
    motor_d = ev3.Motor(ev3.PORT_D, ev3_obj=my_ev3)

    motor_a.start_move_by(360*2, speed=80, brake=True)
    motor_d.start_move_by(360*2, speed=80, brake=True)
```

To get the colour from the colour sensor you can do:

```
import ev3_dc as ev3

with ev3.EV3(protocol=ev3.USB) as my_ev3:
    print(my_ev3.sensors)

    my_color = ev3.Color(ev3.PORT_4, ev3_obj=my_ev3)

    print('The color is', my_color.color)
    print('The reflected intensity is ', my_color.reflected, '%')
    print("The ambient colour is", my_color.ambient)
```

To get the touch sensor value you can do:

```
import ev3_dc as ev3

with ev3.EV3(protocol=ev3.USB) as my_ev3:
    my_touch = ev3.Touch(ev3.PORT_1, ev3_obj=my_ev3)

    print('touched' if my_touch.touched else 'not touched')
```

3.1.1 X11 Forwarding (not required)

On Mac you need to install xquartz, keep it running in the background whenever you need X11, and in the preferences enable *Security* → *Authenticate Connections*, and *Allow connections from network clients*.

On Linux X11 forwarding on the client should work out-of-the-box.

For X11 on Windows you either need PuTTY client or Xming. See this guide for more details.