Betriebsdokumentation

System Requirements

Hardware

Prototype 1

- Raspberry Pi 4 8GB
- Alphabot II

Prototype 2

- NVIDIA Jetson Nano
- Arduino Uno
- Adafruit Motor Shield v2.3
- 2 x DC Motors
- USB-Webcam
- WLAN-Module
- Battery Pack 4 AA-Bateries
- Battery Pack 5V 4A
- Bluetooth HC-05 Module (only used for testing)

Software

- Ubuntu Server 22.04.3 LTS (64-bit)
- ROS 2 Humble

System Setup

Introduction

This guide provides instructions on how to set up a ROS 2 environment on a Raspberry Pi or Jetson Nano. It also includes instructions on how to install the necessary packages for the Follow-Me Robot project.

The project offers two different platforms for running the Follow-Me Robot, which can be partially combined.

Choose the main processing computer:

- Raspberry Pi 4
- Jetson Nano

Choose the platform for the Robot:
- Waveshare Alphabot2
- Arduino Uno and Adafruit Motor Shield v2.3

Recommended Setup:
- Jetson Nano + Arduino Uno & Adafruit Motor Shield v2.3
or
- Raspberry Pi 4 + Waveshare Alphabot2

Raspberry Pi 4 with the Arduino Uno and Adafruit Motor Shield v2.3 is also possible without any problems. It is not recommended to use the Jetson Nano with the Waveshare Alphabot2 because Waveshare Alphabot2 isn't built for the Jetson Nano.

Depending on the chosen platform, follow the corresponding instructions in the guide.

Setting up Raspberry Pi (skip if using Jetson Nano)

Installing Ubuntu Server 22.04.3 LTS (64-bit) on Raspberry Pi

Download and install the official Raspberry Pi Imager from (https://www.raspberrypi.com/software).
Choose the following parameters:

- Operating System: `Other general-purpose OS/Ubuntu/Ubuntu Server 22.04.3 LTS (64-bit)`.

Setting up Jetson Nano (skip if using Raspberry Pi)

Installing Jetson Nano - Ubuntu 20.04 image

Follow this guide to install the image Jetson Nano with Ubuntu 20.04 OS image

Connecting to Your Raspberry Pi or Jetson Nano Using SSH

Connect the Raspberry Pi or Jetson Nano to Ethernet, either directly to your PC or to your local network.

Find out the IP address of your Pi or Jetson Nano by using one of the following methods:

- Using the Router Interface:
  1. Log in to your router's web interface. This typically involves entering the router's IP address into a web browser.
  2. Look for a section named "Connected Devices," "DHCP Clients," or similar. This section should list all devices connected to the network along with their assigned IP addresses.

3. Find the entry corresponding to your Raspberry Pi.

- Using a Network Scanner App (if connected to the same network):
  1. Use a network scanner app on your smartphone, tablet, or PC.
  2. Scan the local network for connected devices, and look for the Raspberry Pi.

Open the command prompt or terminal and type the following command:

for Raspberry Pi:

```
ssh ubuntu@<ip address of the device>
```

Enter the password when prompted (default password: ubuntu).

for Jetson Nano:

```
ssh jetson@<ip address of the device>
```

Enter the password when prompted (default password: jetson)

Setting up Wi-Fi

The instructions are based on this guide and modified for our use case.

Install required packages

```
sudo apt-get install hostapd dnsmasq
```

Hostapd

The purpose of Hostapd is to set WiFi as an access point.

Write a new config file for Hostapd:

```
sudo vi /etc/hostapd/hostapd.conf
```

Add the following content:

```
interface=wlan0
driver=nl80211
ssid=FollowMeRobot
hw_mode=g
channel=7
wmm_enabled=0
```

```
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=ubuntu1234
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Tell Hostapd to use our config file by editing `/etc/default/hostapd` and changing the line that starts with `DAEMON_CONF` (`remove`).

```
sudo vi /etc/default/hostapd
```

It should look like this:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Start Heostapd:

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl start hostapd
```

dnsmasq

Dnsmasq acts as a DHCP Server, so when a device connects to Raspberry Pi or Jetson Nano, it can get an IP assigned to it.

Make a backup of the default config:

```
sudo cp /etc/dnsmasq.conf /etc/dnsmasq.conf.org
```

Create a new config file:

```
sudo vi /etc/dnsmasq.confi
```

Add the following content:

```
interface=wlan0
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
```

Reload dnsmasq config:

```
sudo systemctl reload dnsmasq
```

Solving Startup Error

On system startup, dnsmasq will not wait for the wlan0 interface to initialize and will fail with error `wlan0 not found`.

We need to tell systemd to launch it after the network gets ready, so we will modify dnsmasq service file by adding `After=` and `Wants=` under `[Unit]` section.

```
sudo vi /lib/systemd/system/dnsmasq.service
```

Add the following lines under `[Unit]`:

```
[Unit]
...
After=network-online.target
Wants=network-online.target
```

Config Static IP

Ubuntu uses cloud-init for initial setup, following file needs to be modified to set the wlan0 IP.

Modify the cloud-init file:

```
sudo vi /etc/netplan/50-cloud-init.yaml
```

Add the following content to the file:

```
        wlan0:
            dhcp4: false
            addresses:
            - 192.168.4.1/24
```

The final file should look like this:

```
network:
    version: 2
    ethernets:
        eth0:
            dhcp4: true
            match:
                macaddress: 12:34:56:78:ab:cd
            set-name: eth0
```

```
        wlan0:
            dhcp4: false
            addresses:
            - 192.168.4.1/24
```

then run the following command to apply the changes:

```
sudo netplan apply
```

Finally, reboot your Raspberry Pi or Jetson Nano and check if you can connect to it via WiFi and SSH.

for troubleshooting check the documentation

Setting up ROS 2

Installing ROS 2 Humble on the Raspberry Pi

Follow the installation guide at (https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html.)
Alternatively, run the installation script.

Installing ROS 2 Humble on the Jetson Nano

first we need to install the corect python version since Ros2 Humble is only compatible with python 3.8.

```
sudo apt install python3.8
sudo apt install python3.8-dev
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.8 1
sudo update-alternatives --config python3
```

select the python3.8 version

and the check the version with the following command

```
python3 --version
```

Because the Jetson Nano image we are using runs on Ubuntu 20.04, we can follow the same installation guide as for the Raspberry Pi and we have to instal from source

Follow the installation guide at (https://docs.ros.org/en/humble/Installation/Alternatives/Ubuntu-Development-Setup.html)

Creating the Workspace

To manually create the workspace, execute the following commands:

```
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws/src
```

Alternatively, run the installation script.
If you encounter issues, follow this tutorial (https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Colcon-Tutorial.html).

Install Dependencies

Installing OpenCV and cv_bridge

Use the *vision_opencv* repository from this GitHub repository(https://github.com/ros-perception/vision_opencv). Install the dependencies:

```
sudo apt install python3-numpy
sudo apt install libboost-python-dev
```

Clone the repository:

```
cd ~/ros2_ws/src
git clone https://github.com/ros-perception/vision_opencv.git -b humble
cd ~/ros2_ws
colcon build --symlink-install
```

Install *Python3-opencv*:

```
sudo apt install python3-opencv
```

Install Rpi.GPIO

Install the Python GPIO Library (allows access to the GPIO Pins of the Raspberry Pi):

```
pip3 install RPi.GPIO
```

Installing the Servo Library

Install the python libraries to enable communication with the PCA9685 servo:

```
sudo pip install smbus
```

Installing falsk

```
sudo pip3 install flask
sudo pip3 install flask-socketio
sudo pip3 install flask-cors
```

Installing Packages

Installing the Camera Package

Clone the camera_package repository:

```
cd ~/ros2_ws/src
git clone https://github.com/cl-ire/camera_package.git NOTE: Placeholder, update the
path later
cd ~/ros2_ws
colcon build
source install/setup.bash
```

Uploading Yolo Config Files

Create the yolo_config folder in the src folder:

```
cd ~/ros2_ws/src
mkdir yolo_config
```

Download the yolo config files from google drive.

Upload the files to the yolo_config folder using WinSCP or similar tools.

If you use `ls`, you should see the files `yolov3.cfg` and `yolov3.weights` in the yolo_config folder.

Installing the ros2_for_waveshare_alphabot2 Package (skip if using Arduino)

This repository is a ROS II version based upon the ROS for Waveshare Alphabot2 Repository by Shaun Price.

Download and build the Repository:

```
cd ~/ros2_ws/src
git clone https://github.com/cl-ire/ros2_for_waveshare_alphabot2.git
```

```
cd ~/ros2_ws
colcon build
source install/setup.bash
```

Building the Waveshare Alphabot2 (skip if using Arduino)

Follow this tutorial (https://www.waveshare.com/wiki/AlphaBot2) to build the Waveshare Alphabot2.

Setting up Arduino Uno and Adafruit Motor Shield v2.3 (skip if using Alphabot2)

Setting up the Arduino IDE

Follow this tutorial (https://funduino.de/hardware-software) to install the Arduino IDE.

Build the Robot

Follow this tutorial (https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/install-headers) to build the Adafruit Motor Shield.

Connect the DC motors to Port M3 and M4 of the Adafruit Motor Shield. Connect a 6V power source (e.g., 4 AA batteries) to the power input of the Adafruit Motor Shield. Connect the Adafruit Motor Shield power input to the Arduino Uno via a cable with a barrel jack and open contacts.

If you want to use the Arduino independently from the Jetson Nano, follow this guide (https://funduino.de/tutorial-hc-05-und-hc-06-bluetooth) to install the Bluetooth module HC-05. This is necessary to run certain tests wirelessly.

Installing the Adafruit Motor Shield Library

Follow this tutorial (https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/install-software) to install the Motor Shield V2 library.

Upload the Script to the Arduino

Connect the Arduino Uno to the PC via USB cable. Open the Arduino Script in the Arduino IDE and choose the COM port the Arduino is connected to (Tools → Port → COMX). Upload the Script to the Arduino (Upload).

To test the script, open the serial monitor (Tools → Serial Monitor) and send the following commands:

```
100,100,1000
```

This should move the robot forward for 1 second.

If you want to test it more extensively, you can use the Python script (https://github.com/tBuddy00/Follow-Me-Roboter/blob/main/src/Arduino/Test/arduino.py)

Build the Arduino Setup (skip if using Alphabot2)

Connecting the Jetson Nano or Raspberry Pi and Arduino

Connect the Jetson Nano or Raspberry Pi to the Arduino Uno via USB cable.

Connect the Jetson Nano (5v 4A) or Raspberry Pi (5V 2A) to a power source.

Usage via SSH

Raspberry Pi 4B

Connect to WiFi

```
ssid: Alphabot2

Password: ubuntu1234
```

Connect with SSH

```
ssh ubuntu@192.168.4.1

Pw: ubuntu1234
```

Jetson Nano

Connect to Wi-Fi

```
ssid: JetsonNano

Pw: jetson1234
```

Connect with SSH

```
ssh jetson@192.168.5.1
```

```
Pw: jetson
```

Run Project from Console

```
cd ~/ros2_humble/ && source install/setup.bash
cd ~/ros2_ws/ && source install/setup.bash
cd ~/ros2_ws/src/camera_package/launch

ros2 launch follow_me_arduino_launch.py
```

there are different launch files

- Launch for the Arduino version `follow_me_arduino_launch.py`
- Launch for the Alphabot2 version `follow_me_launch.py`
- test the Arduino version `arduino_test_launch.py`

Debuging Comands

reinstall the camera_package

```
cd ~/ros2_ws/src/ && rm -r -f camera_package && git clone  https://github.com/cl-
ire/camera_package.git
cd ~/ros2_ws && colcon build --packages-select camera_package
```

reinstall the ros2_for_waveshare_alphabot2 package

```
cd ~/ros2_ws/src/ && rm -r -f ros2_for_waveshare_alphabot2 && git clone
https://github.com/cl-ire/ros2_for_waveshare_alphabot2.git
cd ~/ros2_ws && colcon build --packages-select ros2_for_waveshare_alphabot2
```

Troubleshooting

Problems with the Wi-Fi

First, check what isn't working by running the following commands:

```
# Check the status of the dnsmasq service
sudo systemctl status dnsmasq

# Check for the softblock issue
sudo rfkill list
```

```
# other useful commands
sudo systemctl status check-dnsmasq.service
sudo systemctl status unblock-wifi.service
```

fix softblock issue

If your Wi-Fi network doesn't boot properly, check the following commands:

```
sudo rfkill list
sudo systemctl status dnsmasq
```

if it looks like this

```
0: phy0: Wireless LAN
        Soft blocked: yes
        Hard blocked: no
```

Then you have to fix the softblock issue. We have to create a script to unblock wlan0 on boot.

Run the following command to create the script:

```
sudo vi /etc/systemd/unblock-wifi.sh
```

Add the following content:

```
#!/bin/bash

# Ensure that wlan0 is unblocked
sudo rfkill unblock wlan
```

Run the following command to make the script executable:

```
sudo chmod +x /etc/systemd/unblock-wifi.sh
```

Run the following command to create a service that runs the script on boot:

```
sudo vi /etc/systemd/system/unblock-wifi.service
```

Add the following content:

```
[Unit]
Description=Unblock wifi on boot
After=network-online.target
```

```
Before=dnsmasq.service

[Service]
Type=oneshot
ExecStart=/etc/systemd/unblock-wifi.sh

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

Run the following commands to enable the service:

```
sudo systemctl enable unblock-wifi.service
sudo systemctl start unblock-wifi
```

Edit the dnsmasq service file to make sure that the unblock-wifi service runs before the dnsmasq service:

```
sudo vi /lib/systemd/system/dnsmasq.service
```

Add the following lines under `[Unit]`:

```
[Unit]
...
After=wlan0.service
After=unblock-wifi.service
```

Reboot the device:

```
sudo reboot
```

if softblock issue still persists

If the softblock issue still persists, we can try implementing a service that restarts dnsmasq service after boot.

1. Create a script to check dnsmasq status and restart if necessary:

```
sudo vi /usr/local/bin/check_dnsmasq.sh
```

Add the following content to the file:

```
#!/bin/bash

# Check if dnsmasq is active
```

```
if systemctl is-active --quiet dnsmasq; then
    echo "dnsmasq is already running"
else
    echo "dnsmasq is not running, attempting to restart services"
    systemctl restart unblock-wifi.service
    if systemctl restart dnsmasq; then
        echo "Successfully restarted dnsmasq"
    else
        echo "Failed to restart dnsmasq"
    fi
fi
```

Save and exit the editor (`Esc`, then `:wq`).

1. Make the script executable:

```
sudo chmod +x /usr/local/bin/check_dnsmasq.sh
```

1. Create a systemd service unit file:

```
sudo vi /etc/systemd/system/check-dnsmasq.service
```

Add the following content:

```
[Unit]
Description=Check if dnsmasq is running and restart services if necessary
After=multi-user.target

[Service]
Type=oneshot
ExecStart=/usr/local/bin/check_dnsmasq.sh
StandardOutput=journal
StandardError=journal

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

Save and exit the editor (`Esc`, then `:wq`).

1. Enable the service to run at boot:

```
sudo systemctl enable check-dnsmasq.service
```

1. Test the service:

```
sudo systemctl start check-dnsmasq.service
sudo systemctl status check-dnsmasq.service
```

You should see the status of the service, including the echo statements from the script.