TDT4258 Low-Level Programming
HS 2024

Lab assignment 3

Embedded Device Programming

Handout: Thursday, 3rd October 2024, 12:00
Deadline: Friday, 1st November 2024, 17:00

Teaching Assistants: Mikkel Svartveit and Magnus Øvre Sygard
Assignment Coordinator: Roman K. Brunner
Professor: Lasse Natvig

Pre-amble

The labs are here for you to deepen your understanding of concepts taught in the lecture. The goal is that you not only develop a theoretical understanding of the matter, but also develop the technical skills to apply it in practice.

Each lab has a **main project**, but we also provide optional exercises for those who want to go beyond the mandatory exercise. To collect points for a lab, you only need to **hand in the solution to the main project** (in addition to filling in an AI-"statement" as described at blackboard).

The optional exercises are for possible extended learning and maybe even your entertainment.

The lab exercises are compulsory activities in the course. **You need to collect 27 points in total to be allowed to take the exam**. We assess the lab assignments such that a fully approved solution will get 10 points, and we hope that will be the normal case. Submissions with significant defects might get a reduced number of points. Since there will be at least 4 labs it will be possible to reach the exam threshold without a full score on three first labs. This is a new approach compared to earlier editions of the course — and has been developed to give students more flexibility.

As the assignments are part of the evaluation, they are subject to NTNU's plagiarism rules [1]. We have tools at our disposal and will run all submissions through plagiarism checkers. Copying code from current or past students is considered plagiarism. Hence, we advise you to not share code to prevent situations where we have to find out who copied from whom.

While copying each other is disallowed, we still encourage student discussions about your solutions. This will allow you to explore alternative approaches and solutions and learn about the advantages and challenges of particular implementations. **As a rule of thumb: Sharing and discussing ideas is fine, sharing code is not.**

1 Description

In this lab, you will program an embedded device and interact with the attached hardware. You are going to set up a Raspberry Pi 4, attach an extension board and use it to run a simplified version of TETRIS. The main goal of this assignment is to output the game state on a LED matrix and capture the joystick input to allow a user to play the game.

[1] https://i.ntnu.no/wiki/-/wiki/English/Cheating+on+exams

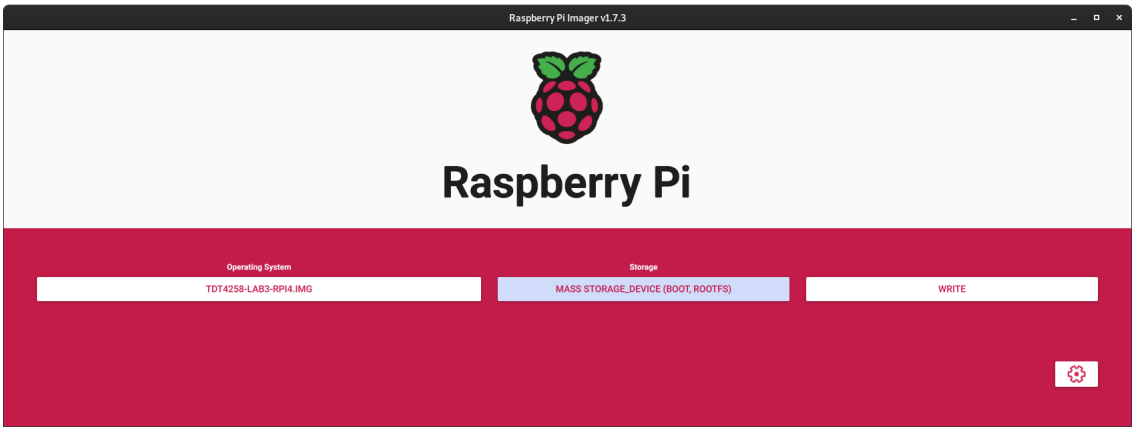


Figure 1: Configuration dialog of Pi Imager

2 Setting up the Raspberry Pi 4

You will be provided with a Raspberry Pi 4, a power supply, some additional cables and an SD card for the operating system. You have to flash an operating system image onto the SD card and find a suitable way to connect to the Raspberry Pi 4. You will have two main different options to work with the Raspberry Pi 4:

1. You can use it as a normal desktop and connect a screen, keyboard and mouse to it.

2. You can also work over the terminal on the Raspberry Pi 4, for which you need a local network to which you can connect over WiFi or Ethernet.

2.1 Raspberry Pi OS

Note: If you do not have an SD card reader or micro SD card reader, please contact us, so we can provide you with a USB SD card reader.

For the first step, you have to flash the operating system onto the SD card. Do not use whatever is installed on the SD card as there is no guarantee that it is the correct OS with the correct configuration. Use only the OS that we provide to you on Blackboard (compressed `.img` file).

To flash the image onto the SD card, go to https://www.raspberrypi.com/software/, download and install the Raspberry

Pi Imager on your local machine. Plug in your SD card and open the Raspberry Pi Imager. Set the Operating System to 'Use custom' and select the `.img` file you downloaded from Blackboard. For 'Storage', select the SD card reader (see Figure 1).

Next press CTRL+SHIFT+X (MacOS: CMD+SHIFT+X) to open the advanced image options. Tick the box to set the hostname. You can leave the default hostname or set a custom one, depending on the environment you are going to work. It is always useful that every device in your network has a unique hostname. Tick the box to enable SSH and provide a password (You need to remember this password later on). If you want to connect your Raspberry Pi 4 to your local WiFi, tick the box to configure WiFi and put in your SSID (WiFi Name) and password. Networks like eduroam are not supported, as they use other means of user authentication. Compare to Figure 2 for details.

Press the write button and confirm that you want to overwrite the content of the SD card. Unplug the SD card after writing the image has finished and insert it on the back of the Raspberry Pi 4.

2.2 Sense Hat

You are provided with a Raspberry Pi Sense Hat extension board, which you need to connect to your Raspberry Pi 4. Connect it to the GPIO pin header so that the extension board covers the Raspberry Pi 4 PCB like seen in Figure 3.

2.3 Working on the Raspberry Pi 4

Connect the power supply to the Raspberry Pi 4 and it will automatically start to boot. The Sense Hat board lights up the LED matrix in rainbow colors. After booting has finished, the Sense Hat LED matrix shows a white square.

To work on the Raspberry Pi, you have several options to connect to it. The easiest option is just to connect a screen, keyboard and mouse to use the Raspberry Pi as a normal desktop computer. You will be presented with a graphical user interface. In any way, you should connect the Raspberry Pi to the internet to install additional software if needed. This can be done over Ethernet or WiFi.

If you want to work on the Raspberry Pi from your own computer, you have to connect it to your local network via Ethernet or WiFi. You can find the IP address of the Raspberry Pi in your router or connect to it with the hostname that you have set during the flashing process. You will need an SSH

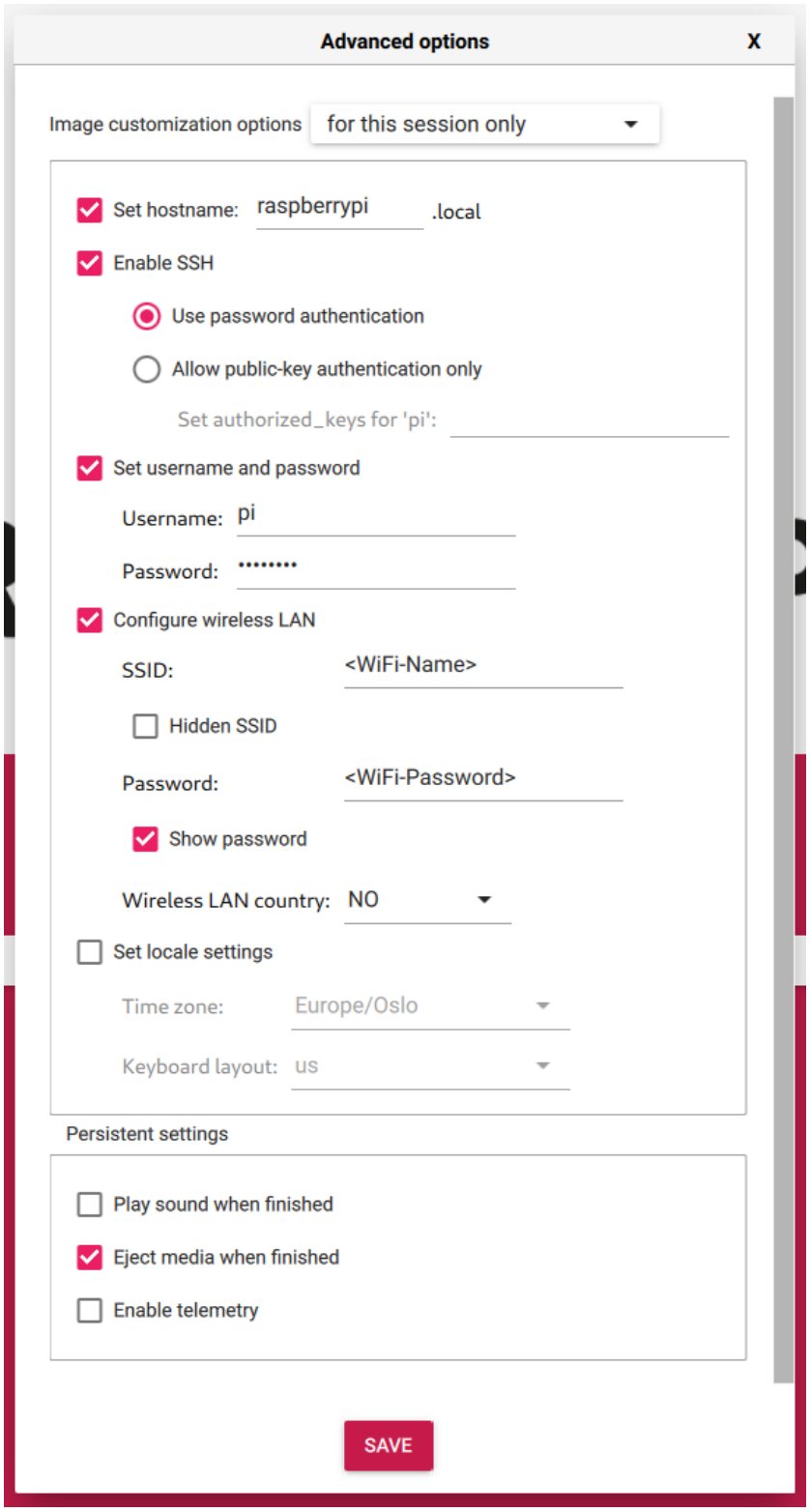


Figure 2: Further configurations can be made in the advanced options. This menu can be opened by clicking on the gear icon in the lower right corner of the window (see Figure 1) or by pressing `CTRL + SHIFT + X`, respectively `CMD + SHIFT + X` on MacOS.)

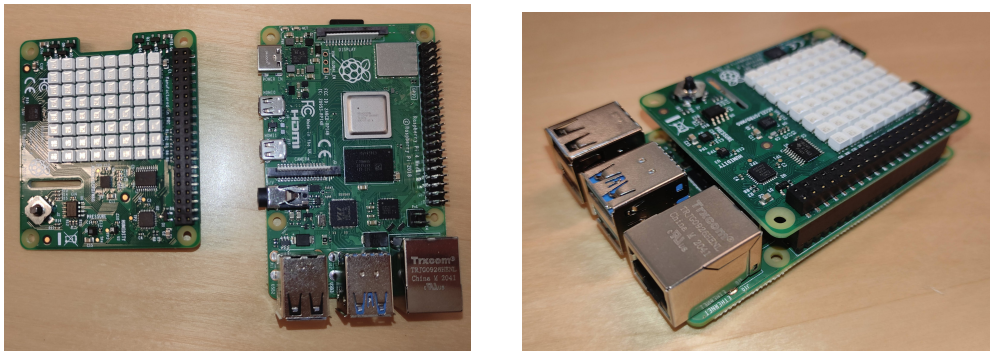


Figure 3: Connecting the Sense Hat extension board to the Raspberry Pi 4.

SSH connection via IP SSH connection via hostname

Figure 4: Connecting to the Raspberry Pi.

client on your system (preinstalled on Linux and MacOS; for Windows you can use the Linux Subsystem on Windows). If you are working on Windows and none of these options are available to you, your can install PuTTY or any other SSH client that is available for Windows. In Figure 4 you can see the different methods for connecting a Raspberry Pi. To log in use that username and password that you have entered during the SD card setup.

3 Main Task: Play TETRIS on the Sense Hat

You find the source code for a simple implementation of the TETRIS game, which runs out of the box on the Raspberry Pi 4 console and can be controlled with the arrow keys of your keyboard (Enter exits the game). Your task is to display the playing field in a colorful way on the Sense Hat LED matrix and make it playable with the Sense Hat joystick. You are not allowed to use any libraries apart from standard libraries and you are not allowed to reuse code that you find somewhere on the internet.

Note: You can extend the provided code to your liking, including data structures used for the game logic. However, some limitations apply:

- You are not allowed to remove or change the way the game is outputted

on the console.

- You are not allowed to change the game logic, timings and speeds.
- You are allowed to change data structures within the game logic as long as they do not alter the game itself.

Task Summary

- Display the playing field on the Sense Hat 8x8 RGB LED matrix. Do not display anything else than the tiles, so no scores or other statistics. A requirement for this implementation is to memory map the frame buffer of the LED matrix (documentation below).
- The game must be playable with the Sense Hat joystick, such that a left press moves the current tile to the left, a right press to the right and a down press drops it to the bottom. A center press must exit the game. The tile must also be continuously moved when a joystick is continuously pressed.
- Colourise the playing field such that each tile has a color that must stay the same during the entire time it is visible on the LED matrix. You have to vary the colors used so that not all tiles on the playing field have the same color. You are free in how your program chooses colors, but pay attention that all colors must be well distinguishable.

3.1 Documentation to Read

To interact with the Sense Hat you need to familiarise yourself with some fundamental concepts of using devices in Linux. You do not need to set up anything special, as your operating system image comes with all the necessary drivers and configurations. The LED Matrix of the Sense Hat is a configuration of 8x8 RGB LEDs of the type RGB565. That means that the color depth of each pixel is 16 bits. The red color channel has 5 bits, the green has 6 bits, and the blue has 5 bits, exactly in this order. The LED Matrix comes with a Linux kernel driver that exposes it as a normal framebuffer in the system with the identification 'RPi-Sense FB'.

To interact with a Linux framebuffer you need to read the following documentation carefully:

- https://www.kernel.org/doc/html/latest/fb/framebuffer.html up to and including the Programmers View of `/dev/fb`

- https://www.kernel.org/doc/html/latest/fb/api.html describes the `ioctl` interface

Note: You cannot assume that the framebuffer always has the same path or that there is only one single framebuffer on your system. Make sure you check the identification of the framebuffer before using it and exit with an error if you cannot find the `RPi-Sense FB`. Further, your LED Matrix is a fixed screen, so you are not required to check or change the variable screen information.

The joystick of the Sense Hat acts as a normal input device where each direction is mapped to the up, left, right, and down key. Pressing the joystick emits the enter key. You will find a raw interface for the joystick device on your system. The identification of the input device is 'Raspberry Pi Sense HAT Joystick'.

To interact with the Linux event system which exposes the joystick inputs, you need to read the following documentation:

- https://www.kernel.org/doc/html/latest/input/input.html - pay special attention to the **Event Interface**.
- https://www.kernel.org/doc/html/latest/input/event-codes.html - focus on the EV_KEY event.

Note: You cannot assume that the joystick has always the same path or that there is only one on your system. Check the identification before using it and exit with an error if you do not find `Raspberry Pi Sense HAT Joystick`. When an input event is ready, it might not be the only event in the queue. There can be multiple input events ready for reading. Finally, you have to check and read input events in a non-blocking way, so that the game does not pause between input events.

Familiarise yourself with the following functions that are helpful or necessary to interact with devices in Linux:

- `mmap` (https://man7.org/linux/man-pages/man2/mmap.2.html) and `munmap` (https://man7.org/linux/man-pages/man3/munmap.3p.html) to map and unmap device memory into the virtual address space.
- `ioctl` to perform special device actions, i.e. setting and retrieving device settings. https://man7.org/linux/man-pages/man2/ioctl.2.html https://www.kernel.org/doc/html/latest/driver-api/ioctl.html
- `poll` to poke if there is anything to do https://man7.org/linux/man-pages/man2/poll.2.html

4 Optional Tasks

- **A [EASY]:** Light up the entire LED Matrix of the Sense Hat with a single color. Change the color with the press of the joystick device (remember that this emits the enter key).
- **B [EASY]:** Use the C-preprocessor with `#define` and conditional compilation to make one C-file that implements 8x8 colored pixels on two platforms; (a) as colored LED's on the sense hat, and (b) using the function `BigPixel()` for the CPUlator VGA as presented in optional tasks in Lab1 and Lab2.
- **C [MEDIUM]:** (Generative AI)
  The Sense Hat has sensors for temperature, humidity and barometric pressure. Read out these values at regular intervals for e.g. 12 hours and store them in a text-file (A .CSV-file with comma separated values can be recommended). Visualize the data. Here we recommend writing a python-program using matplotlib. If you have not used matplotlib before, copilots most often give very good help with such tasks.
- **D [DIFFICULT]:** Indoor climate Station Project (Generative AI)
  Build on optional task C and optional task B of Lab2 (8x8 pixel font) and implement an "indoor climate station" that displays temperature, humidity and barometric pressure at regular time intervals. The read outs should be displayed as text on the Sense Hat LEDs using an 8x8 pixel font. The letters could roll slowly in from the right edge, stay on the sense hat for half a second, and then roll slowly out to the left. You are free to choose other ways of visualizing the climate information on the Sense Hat LEDs.
- **E [HARD]:** The simple TETRIS implementation uses a single simple shape. Can you extend it to multiple different shapes?

5 Submission and assessment

Submit your **commented C code file** `stetris.c` before the deadline on Blackboard. Ensure you haven't changed the console output of the game and that it still runs the provided game logic in the same speed as provided. Make sure that your program compiles and executes before you submit.

We expect all submissions to meet the following requirements:

1. The submission is on time. We provide enough time to solve the labs, but we expect you to start early.

2. The program follows the outlined requirements.

3. You have submitted an *"AI-statement"* for this lab exercise. It will be anonymized and will not influence on the assessment — but will help us all to gather experience with the use of generative AI in the course. This is further explained in lecture 0 and on blackboard.

Failing to meet any of these requirements will result in the number of points collected by the submission to be reduced, depending on the seriousness of the defect(s). No submission gives zero points.

Commenting on your code and keeping it tidy is very important. Helping us understand what you did, supports us in assessing your work – we can only give points for what we understand.

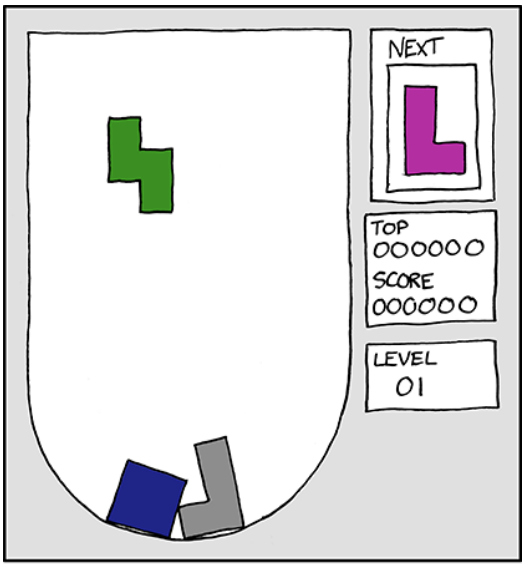
6 Similarity Checking and Plagiarism

You must submit your **own work**. You must write your **own code** and **not copy** it from anywhere else, including your classmates, internet, and automated tools.

7 Questions

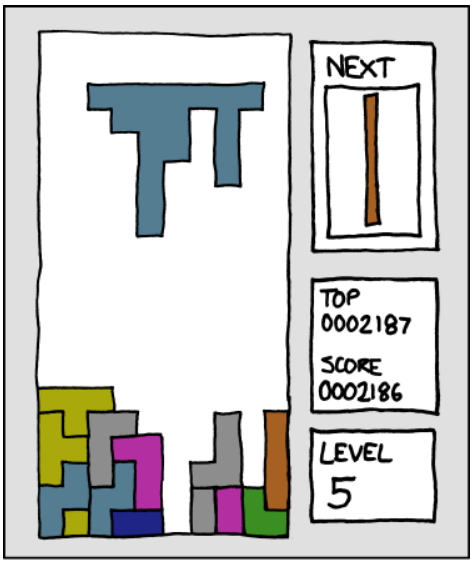
If you have any questions about this assignment, we encourage you to ask the question on the course forum on Piazza. By that, you also help other students who have the same questions in the future.

Source: https://xkcd.com/724/



HELL

Source: https://xkcd.com/888/



HEAVEN