Development Guidelines

The BalcCon Cyberdeck 0o27 (BCD-0o27) comes with a development framework that makes it easier to add functionality to the cyberdeck. This development guide introduces the framework, advises coding best practices and provides examples for adding functionality to the BCD.
Of course, you can ignore all of that and program the BCD either with esp-idf, rust, or any other framework that allows you to program ESP32 chips directly. However, this is not covered in this guideline.

Introduction to the BCD-0o27 Framework

Before reading this chapter it is advisable to get a local copy of the framework (see Getting the Framework) and set up the development environment (see Setting up the Development Toolchain).
The framework consists of the following components:

- **Drivers**: The drivers for the hardware, namely the ST7735 display, the WS2812 LEDs and the input buttons (aka controller) as well as the serial console. (Sometimes, drivers and libraries are not distinguished)
- **Libraries**: Libraries are provided for common tasks like opening a console on the serial port, establishing ssl connections or displaying graphical elements.
- **Modules**: Modules are programs that can be launched by the user. They can be simplest demo applications or complex programs.
- **Console Commands**: Console commands are programs that can be launched from the console. As modules, console commands can have any level of complexity. While usually they mainly interact with the serial console, they may - just as modules - access other hardware as well.

This document is currently being developed. For the time being, it is recommended to check the appendix to clone the repository. Then use the example firmware provided in the directory.

> **Info**
>
> You can connect to the badge using a serial console and a usb-c data cable. Make sure you use a terminal that understands ansi sequences. You can use the integrated serial monitor of platformio or espressif, but its highly recommended to use syncterm or minicom.

> **Warning**

There currently is a bug in the serial console that will not give you a prompt when you connect using a serial terminal. In that case, leave the connection open and simply reset the badge by pushing the reset button - left button under the display.

Appendix

Getting the Framework

The framework is hosted as a git repository at https://gitlab.com/fschuetz/bcd-0o27.git. The repository makes heavy use of submodules, as many components for the BCD-0o27 are developed in parallel.
To create a local repository using git follow these steps:

```
git clone --recurse-submodules https://gitlab.com/fschuetz/bcd-0o27
```

Setting up the Development Toolchain

In this chapter we will show how to set up the development environment consisting of git, espidf, Microsoft Visual Studio Code and Plattformio. If you are an experienced user, feel free to use the IDE of your choice.

Installing GIT and cloning the repository

We will use git from the command line, as we have to deal with submodules, which are not well supported in the Visual Studio Code IDE.
If you have git already installed, you can skip this step.

1. Install git for your platform. Follow the guide at Gitlab
2. Clone the repository:
   ```
   bash git clone --recurse-submodules https://gitlab.com/fschuetz/bcd-0o27
   ```
   Some tips on working with submodules, in case you run into a head detached situation: https://stackoverflow.com/questions/18770545/why-is-my-git-submodule-head-detached-from-master

Install Microsoft Visual Studio Code (VSCode)

1. Install VSCode for your operating system.
2. Open VSCode.

Install C/C++ and PlatformIO Extensions

1. In VSCode: Open the "Extensions" side bar on the left side.
2. Search for the "C/C++" extension by Mircrosoft and install it.

3. Search for the PlatformIO IDE by PlatformIO and install it. See PlatformIO IDE for VSCode
4. Once installation has finished, restart VSCode

Build and Upload the Firmware

In this chapter we will build the filesystem image and the firmware and upload both to the badge. If this works, then you can start developing your own firmware using the framework. For testing we highly recommend, taking the example in the directory `firmware/example` rather than the framework, as the example will give you running firmware that allows you to see something actually happening.

1. Start Microsoft Visual Studio and make sure you are in a new window, with no project loaded.
2. Select the alien head in the left bar (Platformio menu)
3. A left band appears. Select the blue button "Pick a folder"
4. From the cloned repository choose the `firmware/example` or the `firmware/framework` folder (as written above, its recommended to start with the example rather than the plain framework).
5. Click open.
6. Wait for platformio to configure itself. This is IMPORTANT: Between now and building the file there might raise popups in the bottom left corner asking you to install extensions or configurations. Usually you are asked to intall C++ specific extensions and CMake specific extensions. Confirm all of them. For CMake, if asked, you can "scan for kits". Scan for kits may also go away on itself, then everything is fine as well.
7. If all went well so far you will recognise a blue ribbon / button on the bottom and some symbols like a home, a checkmark, roght pointing arrow etc.. If there is the message "Platformio: Loading tasks..." wait until it disapears. (this may take a long while). PlatformIO will automatically configure the project.
8. Open the file main.cpp in the "src" folder.
9. In the bottom task list (with the blue ribbon / button and the checkmark etc...), first choose the build environment by clicking on the symbol that looks like a folder and says "Switch Platformio Project Environment" if you hover over it. Usually it reads default right to the symbol. Choose `env:BCD-0o27-5KY` (or the `env:BCD-0o27-5KY-release` if you want to build for release rather than debug - if unsure make a debug build).
10. Now select the checkmark to build the project. This builds the project. Dependencies are pulled in automatically. In case Platformio does not pull in the Espressif 32 framework, follow the steps in the note below to install it manually.
11. If the project builds successfully, connect your badge to the computer using a usb-c cable (make sure it is actually a data cable and not just a charging cable)
12. Make sure the device is recognised. If the device is not recognised and a com port assigned, you might have to install a driver for the CH340 serial bridge. Follow the instructions from Sparkfun

13. Once the device is recognised, in VSCode we will build the file system image, upload it and then upload the code.
14. From the side bar choose the PlatformIO icon (alien head)
15. Select BCD-0o27 -> Plattform -> Build Filesystem Image
16. When finished, select BCD-0o27 -> Platform -> Upload Filesystem Image (BCD-0o27)
17. When finished, select the right pointing arrow ("Platformio: Upload") in the blue ribbon at the bottom. The project will be built and uploaded to the badge.
18. Either launch the serial console from the blue ribbon by selecting the plug icon ("Platformio: Serial Monitor") or use minicom (preferred). Note, that the Platformio Serial Monitor may display some strange characters or double print the prompt as it does not fully support the output mode. IMPORTANT: Due to a bug, the serial console only works for input if the connection was made at badge startup. If you do not see a prompt, leave the console on and restart the badge by pushing the power button (left between display and battery holder).
19. CRITICAL: Make sure, that you DO NOT have a serial monitor running when trying to flash the badge. This will lead to flash failure (for both, filesystem and firmware).

Note: The Espressif 32 framework should install automatically when building the project. If this is not the case, you can install it manually:

1. Open the Platformio sidebar (left side, alien head icon).
2. Select PIO Home -> Platforms
3. Click on the "Embedded" tab. Search for "Espressif 32" and click it.
4. Install the latest version. (At the time of writing this is version 5.1.0. While we plan to adapt the framework to newer versions, you might want to select this version if the newest version is not working due to breaking changes.)
5. Once the installation has completed confirm the success dialog.

Helpful Resources

Espressif 32 for Platformio

Misc notes - ignore please

Updating mirrors

Go to lib directory of mirror

```
git fetch upstream
git merge upstream/master
git push
```

Updating submodules

```
git fetch
git pull
```

Coding Style

```
ESP_LOGE(TAG_MODULE_NAME, "This is a rather long error message."
                "It should be spred on two lines");
```

If functions too long, spread on multiple lines. Two lines, indent second line by one tab and finish the line with closing bracket and ;. If more than two lines use one argument per line and close on a separate line as follows:

```
myfunction_with_many_arguments(
        argument 1,
        argument 2,
        argument 3
```

TO BE CONTINUED