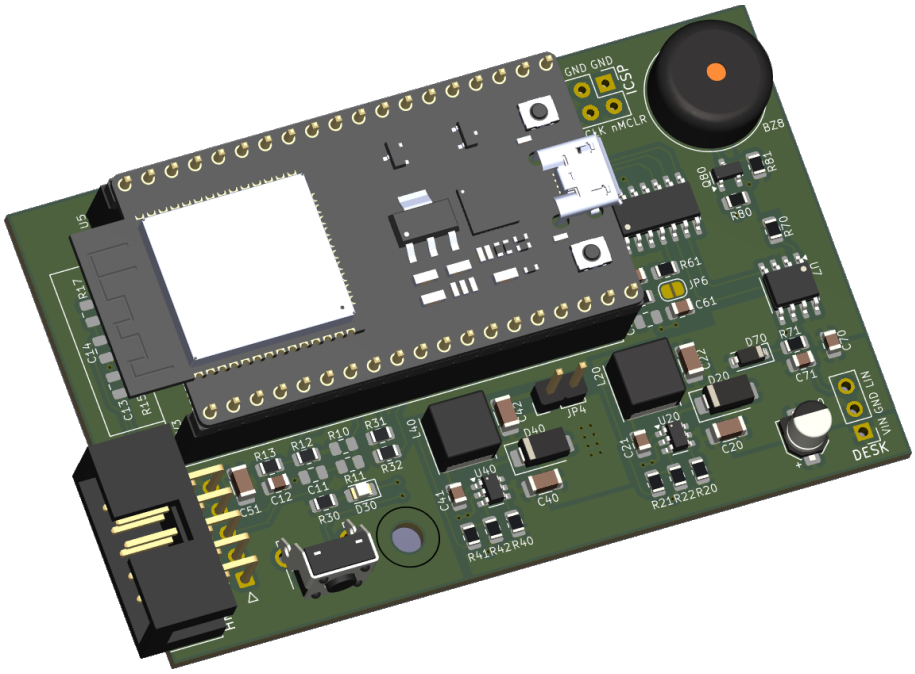
LYFT mk2



Developer Guide

Welcome to LYFT mk2 !

This is an open-source DIY project designed to bring enhanced functionality and greater convenience to the famous IKEA BEKANT standing desk. The second generation of the LYFT remote control is built with both users and developers in mind: whether you want to adapt it for your own needs or contribute new features, the project is ready for you.

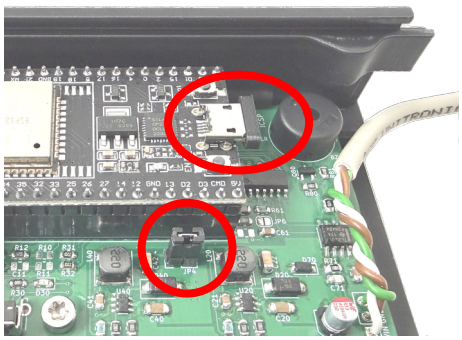
The redesigned dual-controller architecture allows you to control the desk using a simple Python API and run your own applications directly on the ESP platform — no detailed knowledge of the desk protocol is required, as the API and controller handle all tasks internally, enabling endless customization and innovation.

To facilitate a smooth start, the most important information has been summarized here for quick and efficient reference. All source files (hardware and software) are available on the GitHub repository, which can be accessed by scanning the QR code.

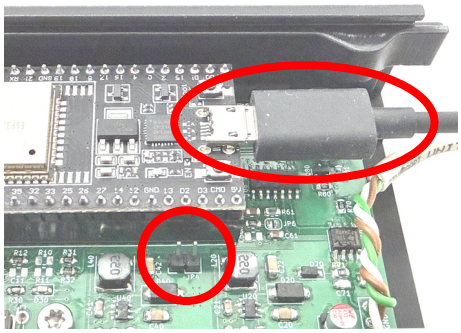
Getting Started

To upload your own software to the ESP module, connect it using a USB cable.

Warning: Before connecting the USB cable, **Jumper four (JP4) must be removed**. Failure to do so will damage the board!



Jumper closed,
no USB cable connected



Jumper opened,
USB cable connected

Once the USB cable is connected to the host controller (ESP module), a connection to the PC must be established **within 3 seconds**. After that, no further input via the USB port will be recognized.

To communicate with the desk controller, the *Bekant* library must first be imported into the project. The library is already installed on the ESP module and can also be downloaded from the GitHub repository. After that, a desk object must be instantiated.

```
import bekant

desk = bekant.Bekant()
```

The desk communication must be explicitly initiated by the host controller (ESP module) using the `startup()` command. The startup routine establishes communication with the desk and sets the motors into operating mode. After a brief startup phase (approximately one second), the desk is ready for use. Communication can also be disabled using the `shutdown()` command.

```
desk.startup()
```

To move the desk, it is recommended to determine the highest and lowest motor positions first. The following queries can be used for this purpose. Every target position must lie within these two limits.

```
desk_upper_limit = desk.get_upper_limit()
desk_lower_limit = desk.get_lower_limit()
```

The current desk position can be read using the `get_position()` command. The returned value is of type `int`. The read position must be between `upper_limit` and `lower_limit`.

```
desk_position = desk.get_position()
```

In order to move the desk to a desired position, the simple `set_position()` command can be used. The desk will automatically move to the specified position and stop once it is reached.

```
target_position = 2000
desk.set_position(target_position)
```

The desk can be stopped at any time using the `stop()` command.

```
desk.stop()
```

To avoid unexpected interruptions or error states, it is recommended to use the built-in watchdog of the desk controller. However, it should only be enabled after the startup routine, so that this routine can run without interruption. Once the watchdog is activated, **a request must be sent to the desk controller every 100 ms**. For example, the current desk position or the operating state can be queried. If no request is received, the desk controller will be reset by the watchdog. After a reset, the desk controller must be reactivated using the `startup()` function.

If necessary, the watchdog can be temporarily disabled using the `watchdog_disable()` function.

```
desk.watchdog_enable()
```

To determine the current state of the desk, the `get_state()` function can be used. It returns a unique, current operating mode (e.g. idle, moving, limit reached, ...). The exact values and their corresponding states can be found in the *Bekant* library (`class State`).

```
desk.get_state()
```

The *Bekant* library provides many additional functions. Further details can be found directly in the `bekant.py` library file or the GitHub repository. Finally, it should be noted that the existing implementation and usage of the library can serve as a good reference for how to use it effectively.

I wish you much enjoyment and success in extending and adapting the capabilities of your desk!

Happy coding!