M-O Control ESP NAW – Sedláček

I use the **ESP NAW protocol** for control. The eye displays process their portion of data as numbers 1-7 and display an image accordingly. The board in the head processes up/down movement (0-180 from the joystick) and controls the beacon based on the received number (1-7). The board in the body controls the driving direction, speed, head rotation, and plays sounds.

I am attaching the Fritzing files, where the wiring can be seen more clearly.

The first essential prerequisite for communication is identifying the **MAC addresses** of all ESP32 and ESP8266 units. The attached files contain Arduino sketches named **"MAC address code"** which will print the MAC address to the Serial Monitor. **Make sure to write these addresses down!** You will need them later in your code.

For the "Eyes" and the ESPs controlling the "Head" and "Body," you will need to record the MAC address of the **Controller**. Conversely, you must program the MAC addresses of the Eyes, Body, and Head into the Controller. This acts as a security feature to prevent accidental control of a different robot.

The obtained MAC addresses follow a structure similar to this example:

```
A4:B0:0F:83:0C:8C
```

You must rewrite them into the following format:

```
0xA4, 0xB0, 0x0F, 0x83, 0x0C, 0x8C
```

Instructions for implementation:

- **Body Code:** Find **line 7** and replace `{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}` with your specific MAC address.
- **Head Code:** Follow the same step on **line 7**.
- **Eyes Code:** Follow the same step on **line 12**.
- **Transmitter (Controller) Code:** We will do the same here by recording the MAC addresses of all other ESPs. Enter the required MAC addresses into **lines 30–33**.

For the eyes and the controller, I used this display: ESP32-2432S028R You will need **3 units** ESP32-2432S028R.
https://www.aliexpress.com/item/1005007730940177.html?spm=a2g0o.order_list.order_list_main.40.dc5c1802RcEPQo (**ESP32 for Arduino LVGL WIFI&Bluetooth Development Board 2.8 240*320 Smart Display Screen 2.8inch LCD TFT Module With Touch WROOM)**

Note: These displays use two different drivers... this specific one uses the less common one. I am attaching the configuration file, which must be added to the TFT_eSPI-master library.

I won’t list all the libraries here, but a few details are essential:

- When uploading, select **ESP32 Dev Module**. (Guides on how to add ESP32 and ESP8266 boards can be found online).
- To ensure library compatibility, you must use these specific versions:
    - **ESP32 Board Core:** version 3.3.7
    - **TFT_eSPI:** version 2.5.43
    - **Tjpg_Decoder:** 1.1.0
    - **SD:** 1.3.0

I had to fine-tune the colors and image orientation. **Gemini AI** was a huge help here, as there were issues with the User_Setup.h settings and the display driver version (due to inconsistent manufacturing across different vendors).

Wiring and Setup

- **Powering the Eyes:** I use the included cable. On my prototype board, I created a lead for 5V and GND. I used the connector where RX and TX are located—this is a **5V input**; other pins run on 3.3V! Do not connect the RX and TX pins
- **SD Card:** Upload the images from the folder directly to the SD card. The images are different for the right and left eyes. Do not use folders—place them in the **root directory**.
- **Code Adjustment:** The code for both eyes is identical except for **line 75**: tft.setRotation(3); // 1 for the right eye or 3 for the left eye (or vice versa? :))
- **Transmitter:** Follow the same process—upload the corresponding image files to the SD card and upload the Arduino code. I set the display orientation so the SD card faces me (there is more space inside the controller enclosure that way).
- **Testing:** I am attaching a file for touch testing called „testdotyku“.
- **Joysticks:** Connected via **I2C** using the **ADS1115** (approx. €2). To connect the ADS1115 to the ESP32-2432S028R, I used the included wire and **pins 22 and 27** (I2C bus).
- **Battery Power:** Since I didn't have an extra cable for the battery connector, I cut the end off a USB to USB-C cable and powered the board via the USB-C port. (See photos).

Body and Sound System

I am using two **ESP8266 Wemos D1 mini Pro** boards—one in the head and one in the body. I built custom PCBs for both using prototype perforated boards. Schematics are provided below.

- **Audio:** I use a **DFPlayer Mini MP3 player** (approx. €3) and a mini amplifier (**PAM8403**).

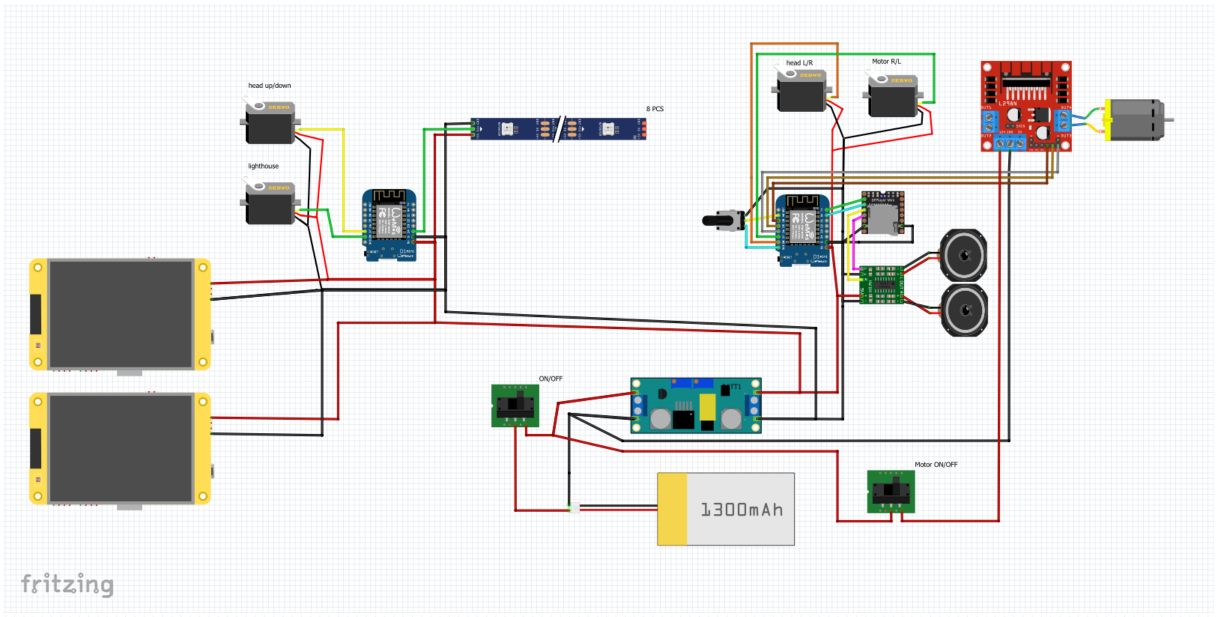
- **Sounds:** I used the original sounds from the manual, but renamed them as required by the MP3 player (**0001 – 0050**).

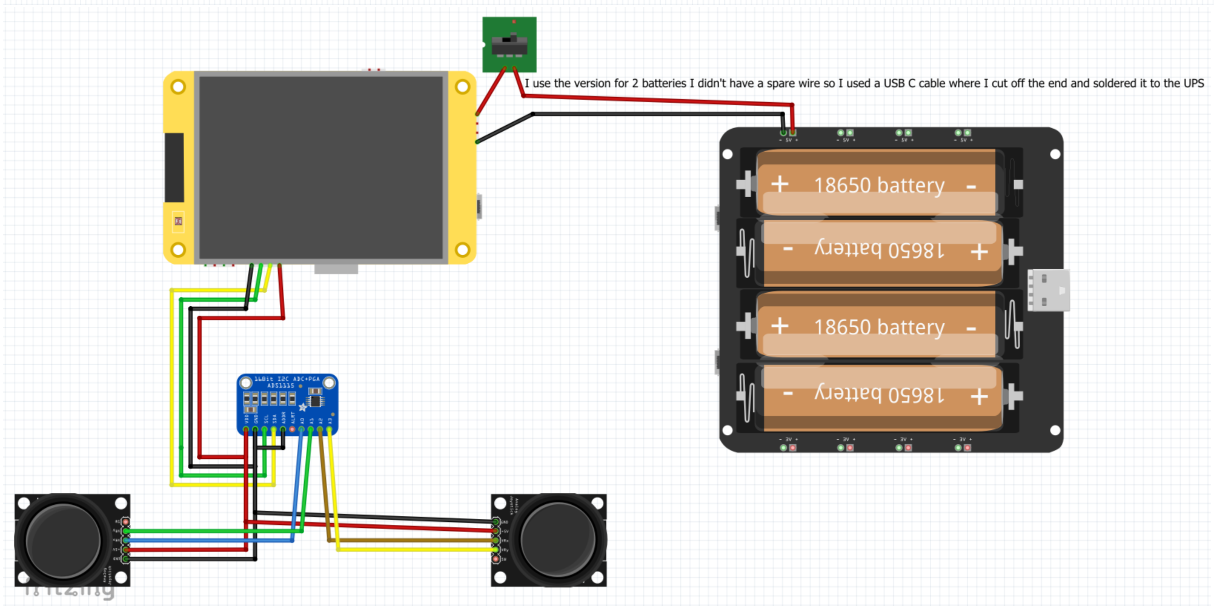
  Warning: These must be in a folder named mp3 on the SD card. (Yes, the files aren't actually MP3s, but once I made that mistake, I didn't want to go back and fix it... SORRY).

- **Volume:** Controlled via a potentiometer.
- Beacon: The beacon in the head consists of an RGB LED strip (5V, 8 LEDs) left over from an old project. Tips and Observations

- I chose this solution to minimize wiring between the body and the head—only 2 wires (**GND and 5V**) are needed.

- **Weight:** Do not 3D print the head too heavy; use a maximum of **2 perimeters**. I printed mine with solid walls—a big mistake—the head is too heavy, and the robot tips over. It helped that the head rotates into the direction of travel. Also, I reduced the maximum speed when the robot turns left or right.
- **Mounting:** I used **Velcro** to attach all components inside the robot.
- **Stability:** I placed the battery in the lower blue rear cover to keep the center of gravity low. I use a **3S LiPo**.
- **Safety/Demo Mode:** I added a power switch for the motor driver. This allows children to "control" the robot (eyes/sounds) without it actually moving—perfect for tabletop presentations.
- **Future Tweak:** I might add some weight to the arms for better balance.




I use the version for 2 batteries I didn't have a spare wire so I used a USB C cable where I cut off the end and soldered it to the UPS