Manual for Creating and Using the V0.2 Motor Driver

This manual closely describes the process of creating the version 0.2 of the motor driver. Additionally, the manual includes the necessary steps to install custom blocks in the EV3 programming software and use them to communicate with the driver and control motors connected to it.

Printed Circuit Board

The repository includes the necessary files to get the printed circuit board manufactured. The following components are necessary for the board to be functional:

- ESP32-WROOM-32E microcontroller module
- AMS1117-3.3 linear voltage regulator
- Microchip TC4427COA MOSFET motor driver (x4)
- 22μF SMD capacitor
- 330Ω SMD resistor
- 10kΩ SMD resistor (x3)
- 120kΩ SMD resistor
- 3.3V SMD LED diode
- RJ12 socket with a horizontal layout
- 2-pin screw terminal with a 5mm pitch, capable of handling at least 6A (x5)
- Pin header with a 2.54mm pitch (x6 min., OPTIONAL)
- 82KΩ SMD resistor (x2, OPTIONAL)

The resistors, the diode, and the capacitor are all of an SMD (surface mount device) 1206 footprint. For the component placement, use the schematic in `Driver_V0.2/V0.2_CAD_Files/schematic_V0.2.pdf` and the silkscreen of the PCB for guidance. Begin the soldering process with the SMD components first. Start off with the resistors, the capacitor and the LED, followed by the voltage regulator, motor drivers, and finally the ESP32 MCU. The board has thermal relief polygons connected to the ground, so heating

up the GND pins of the components might take longer compared to other pins. After all of the SMD components have been soldered onto the board, solder the THT (through-hole technology) components, beginning with the shortest components, the screw terminals. Soldering pins headers is optional, however, it is strongly recommended as it makes the programming of the microcontroller significantly easier. The minimum number of pin headers that I recommend solder is six, for the GND, 3V3, EN, IO0, TX0, and RX0 pins, however, soldering pin headers for all GPIO pins can help with a diagnostic or debugging process, should such circumstances occur. Lastly, solder the RJ12 socket in place. The R1 and R2 82K pull-up resistors need not be soldered in place, unless you plan to communicate with the EV3 via the I2C protocol. These resistors are necessary for the communication to work correctly according to the EV3 Hardware Developer Kit[1].

3D Printed Driver Enclosure and Driver Assembly

The `Driver_V0.2/V0.2_Print_Files` directory contains the **Top.3MF** and **Bottom.3MF** files for printing the enclosure of the driver. Use a freely available software, such as UltiMaker Cura to prepare the files for printing. As this version of the casing has notably narrower walls on it lower portion, I recommend choosing the cubic infill pattern with a 20% infill ratio to retain the structural integrity of the enclosure. If the sliding and locking mechanism does not work correctly, I recommend smoothing the inner surface of the rails with sandpaper. The battery I used for this driver is KAVAN Li-Po 850mAh/11,1V 40/80C 9,4Wh. After placing the battery in the lower part of the driver, place the PCB onto the platform above the battery (Figure 1). For switching the driver on and off, use a KDC1-101 rocker switch and put it into the opening above the PCB. Solder the positive (red) cable onto one pin and a wire on the second pin. Use a heat-shrinking tube or tape to isolate the connections. Screw the other end of the wire into the positive terminal of the PCB, denoted by the "+" sign of the battery terminal. Afterwards, screw the negative (black) cable of the battery to the "-" terminal.

For the Power Functions connectors on top of the enclosure, cut two PF cables into half. Strip the two middle wires and increase the rigidity of the stripped wire by using crimping pliers with wire terminals, or putting a small amount of solder onto the wire. Push the wires through the openings on the upper part of the enclosure, then screw the cables into the M1, M2, M3, and M4 terminals. Finally, slide the driver case closed. After turning on the driver, it automatically enters pairing mode with the EV3, shown by the slow flashing of the status LED. If the device is connected, the LED stays on in a dimmer mode and if the battery of the driver is low, the LED flashes fast for ten seconds before the MCU enters a deep sleep mode. Afterwards, it is necessary to charge the battery. Turning the driver on and off resets the drivers, which then resumes regular operation.

While the EV3 uses RJ12 ports, the latch on the connector is offset to the side. To create a compatible cable, cut the end of an EV3 cable, strip the wires, then place them in an RJ12 connector with a centered latch, and use a crimping tool to secure the cables in the connector. Alternatively, you can remove the latch of the EV3 connector and use this to connect to the driver, however, note that the connector will not be secured. An image of the driver connected to the EV3 and the motors can be seen in Figure 3.

[1] https://education.lego.com/en-us/product-resources/mindstorms-ev3/downloads/developer-kits/

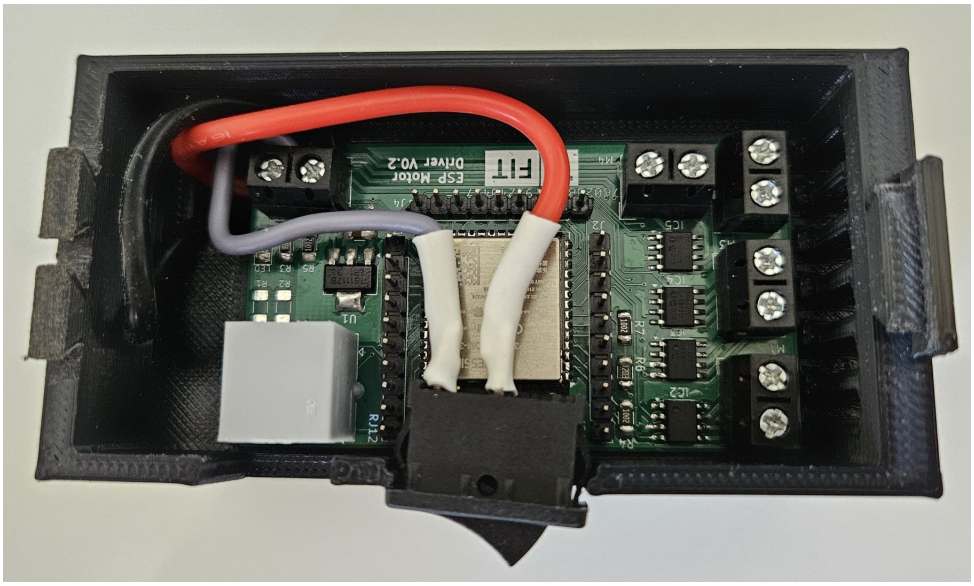

Figure 1: Assembly of the motor driver

Programming the Microcontroller

To program the microcontroller, use an Espressif ESP-Prog programmer or an equivalent alternative device. Connect the 6 pins (GND, V_{PROG}, EN, IO0, TX0, RX0) to the corresponding pins of the MCU (Figure 2). Do not forget to remove the jumper for the V_{PROG} pin if you program the MCU while connected to an external power source. Use Arduino IDE software or the Visual Studio Code editor with *PlatformIO* IDE extension to flash the files located in the `Driver_V0.2/Driver_Code` directory of the repository after choosing the correct device in the board menu. When the ESP-Prog is connected, it uses two COM ports, use the latter one to program the board.

Using the EV3 Software

Download the EV3 Lab Software[2]and install the app on your device. To import new blocks, firstly create a new project program, then navigate to the **Tools** tab. There, select **Block Import**, find the motor block in the wizard (`EV3-G_Blocks/Thesis_Block.ev3b` in the repository), then press *Import*. The software then prompts you to restart it. Afterwards, the motor control block will be available for use in the *Sensor* section of the blocks.

Create a program for the brick, which should begin with the Start block and conclude with the Stop Program block. Add the motor block to the program and choose one of the UART modes of the block to control the motors, filling out the desired parameters. Connect the EV3 brick to your computer via cable, Bluetooth, or Wi-Fi, then download the program to the brick and run it. A picture of the EV3 software interface can be seen in Figure 4. Object number one shows a program example, where all motors are set to run with a duty cycle of 10 out of 15 for ten seconds, then stop for three seconds in an indefinite loop. Number two shows the command block tray with Sensor blocks selected, while number three points to the Motor Control block. Buttons for downloading the program to the brick

[2] https://education.lego.com/es-mx/downloads/retiredproducts/mindstorms-ev3-lab/software/

and running it are located by marker number four. Marker number five shows the *Tools* tab, where the Block Import wizard is located.

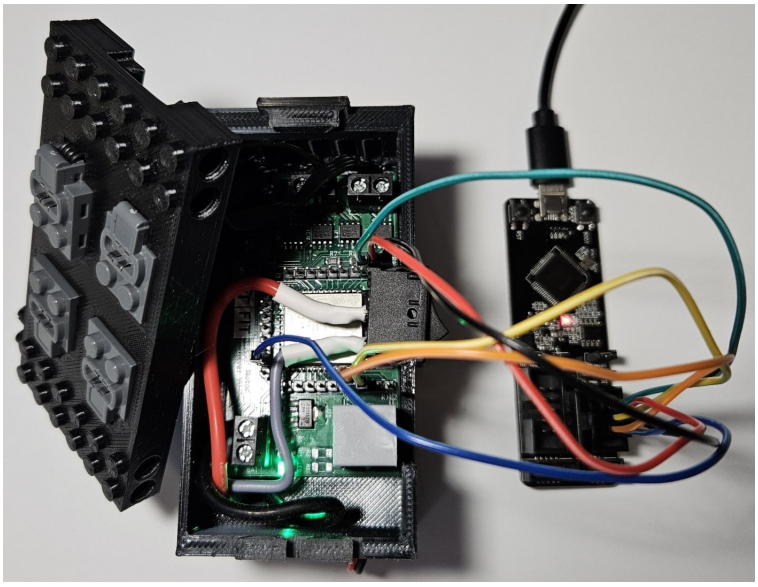

Figure 2: Programming the motor driver V0.2 with an Espressif ESP-Prog

Figure 3: Driver V0.2 connected to the EV3 and four Power Functions motors

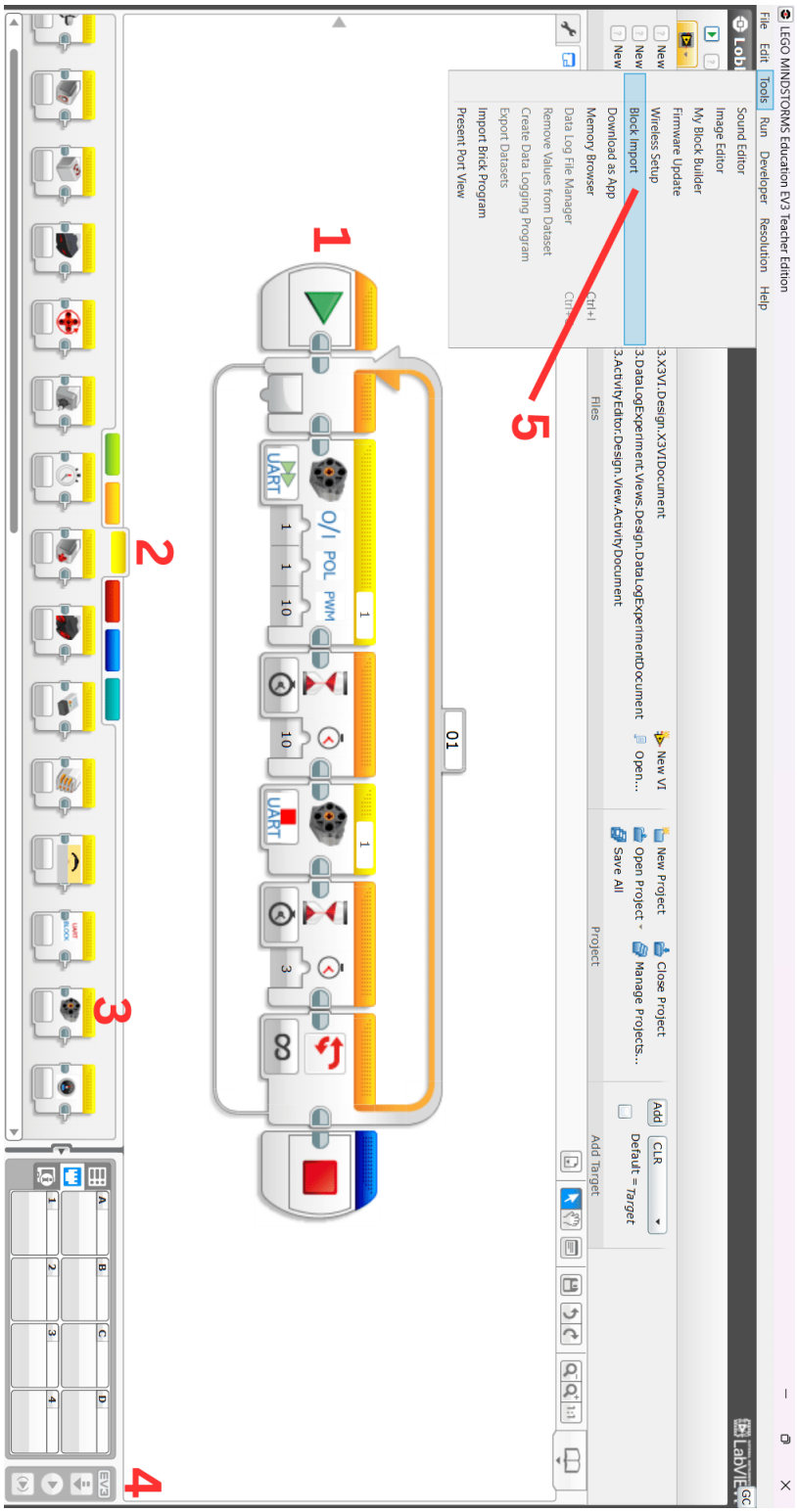

Figure 4: EV3 Lab Software