technical overview - human oversaight Button Interface
14.8.2025

A - hardware, PCB and connections
B - instructions for python scripts, NOTES
C - python wrapper example, necessary libraries

PCB Overview and Connections - human oversaight Button Interface

A

Turn on:

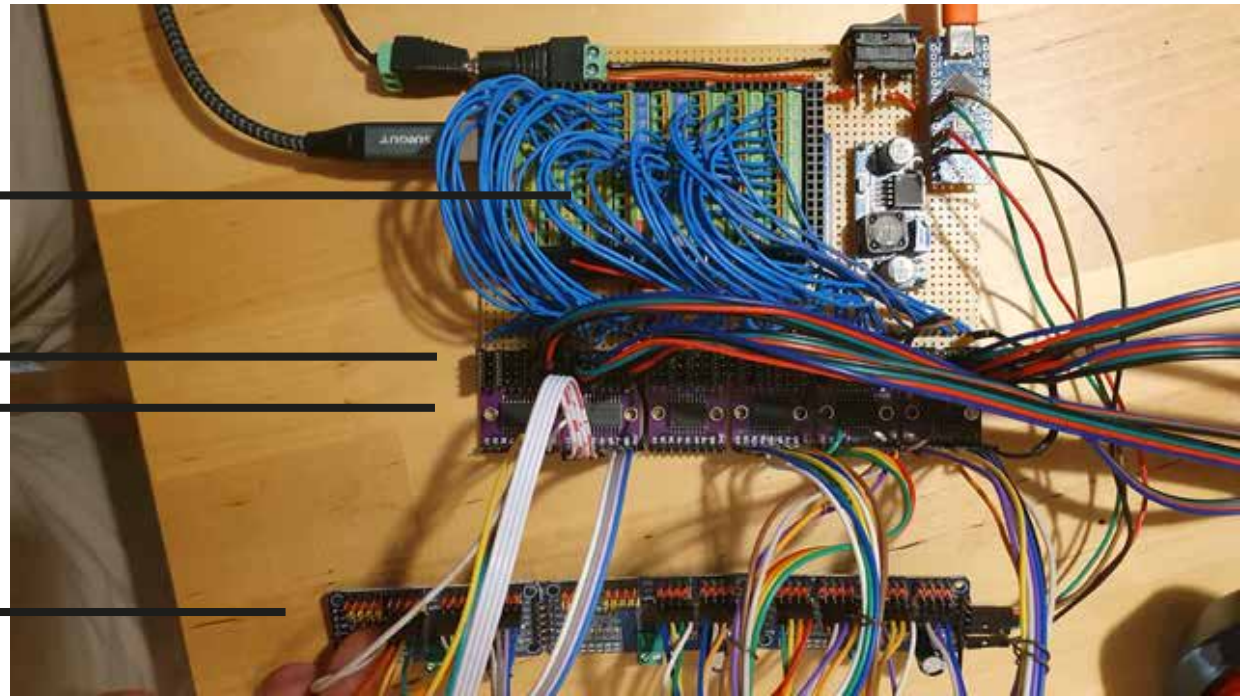
1.: Plug 12V power supply, and turn it on with the rotary knob to full.
2.: Switch ON - 12V for LED, IdleAnimation as long as there is no USB Nano (Step 3.) plugged
3.: Plug Nano USB C for LED realtime control (if not plugged already)
(`comport` from `device manager` for python script `_nano_48_led_PWM`)
4.: Plug Mega USB A for registering button presses
(`comport` from `device manager` for python script `_mega_48_buttons_read`)

Mega - 5V ButtonSwitch signal

plugs
- signal button (blue) -
- 5V (green) -
- 12V (red) -
- LED GND (black) -

ULN2803A Darlington Drivers

PCA9685 I2C PWM



There are two scripts existing, one is responsible for the button registering, the other for the LED control. They can be used and combined with any other python script in common ways. The two examples use tkinter for a fast GUI and maintenance setup. The GUI part can be deleted from final wrappers.

_mega_48_buttons_read.py

! The according COM port needs to be written in the script in the first line saying:
SERIAL_PORT = ‚COMXX‘ # your MEGA`S COMPORT found in device manager on Windows or Linux equivalent, is usually detected once and stays the same when same USB port.

- this reads the 48 buttonStates in total, in this case with a simple tkinter GUI for easy maintenance.
- there is no „framerate“ to set, since it´s just waiting for 48 serial values, as often as the Arduino sends via Serial on the COM port that belongs to the Arduino Mega. This is set to 10ms, aka 100fps button polling.

_nano_48_led_PWM.py

! The according COM port needs to be written in the script in the first line saying:
SERIAL_PORT = ‚COMXX‘ # your NANO´S COMPORT found in device manager on Windows or Linux equivalent, is usually detected once and stays the same when same USB port.

- this writes the 48 LED values in total, in this case with a simple tkinter GUI for easy maintenance.
- the values are integers ranging from 0-255. This enables brightness control.
- in case of ON/OFF binary behaviour just send 0 OR 255 from any other call (instead of 0 and 1) - binary built in.
- The Arduino Nano is update via Serial every 20ms, but this can be adjusted in the code at the top at the according variable UPDATE_INTERVAL_MS = 20.

NOTES:
- Always try to have the power for the LEDs on already before plugging the Arduinos, to not have the LEDs draw power from them only. It´s no problem to start the computer and already attached Arduinos first - but it´s sill good practice to start the 12V supply before.
- Out of the 48 LED outputs, only 40 are working, since one Darlington board was faulty. The two python scripts are used to find the ID´s of according Button LEDs or switches and by that the according plug on the PCB (PCA and Darlington) for mapping IDs if necessary.
- In this setup the PCA9685 combined with the NANO´S I2C are replacing the second Arduino Mega for 48 LED outputs > smaller PCB.

Two other scripts are an example how to use I/O simultaneously and use that wrapper (_controller_01.py) from another python script (_GUI_01.py), using tkinter for testing. So by starting _GUI_01.py after declaring the according COM ports in _controller_01.py you should be able to control the buttons from _GUI_01 via _controller_01 as „forwarder“.

C

_controller_01.py

- there is no need to start this script, it will be called by _GUI_01.py

! The according COM port needs to be written in the script in the first line saying:

```
# — Configuration —
WRITE_PORT   = ‚COMXX‘   # Port to send LED values (NANO) -> 0 OFF <-> 255 ON
READ_PORT    = ‚COMXX‘    # Port to read button states (MEGA)
```

- this forwards any values for leds and reads any buttonstate into two public arrays:

```
led_values    = [0] * NUM_CHANNELS   # 0–255 values to send
button_states  = [0] * NUM_CHANNELS   # 0/1 states read back
```

- the framerate can be set and is now 50 fps:

```
PS        = 50
INTERVAL    = 1.0 / FPS   # seconds
```

_GUI_01.py

- just start this after setting the COM ports in the _controller_01.py

NOTES:
libraries needed:
- pyserial
- tkinter

DISCLAIMER: These two scripts could not be sufficiently tested due to locality reasons, but should work as planned.