BitDogLab Hardware Data Base (HDB)

BitDogLab, an initiative of the School Project 4.0 at Unicamp, is an educational tool devoted to electronics and computing. Based on Raspberry Pi Pico H or W, it allows users to explore, assemble, and program using components mounted on its board and also external ones connected in an organized and secure manner. Meticulously selected, the components foster hands-on learning, encouraging users to enhance programming and electronics skills synergistically and progressively. This enriching platform offers a vibrant experience, immersing users in a colourful, auditory, and synesthetic learning environment. Additionally, BitDogLab is optimized for programming assisted by large language models (LLM), like GPT-4, facilitating a more intuitive learning guided by a tireless tutor. Aimed at pre-university education, BitDogLab aims to catalyze the incorporation of educational technology, providing a robust and flexible tool uniquely integrated into students' learning journey.

A hallmark of BitDogLab is that its project is entirely open, allowing it to be freely copied, manufactured, assembled, and improved by users. More information at: https://github.com/Fruett/BitDogLab

Hardware Connections and Configurations:

The connections of the Raspberry Pi Pico with other components are executed as follows:

A common cathode RGB LED has the red electrode connected to GPIO 12 through a 220-ohm resistor; the green pin is connected to GPIO 13 also through a 220-ohm resistor, and the blue pin to GPIO 11 through a 150-ohm resistor.

A button, identified as Button A, is connected to GPIO5 of the Raspberry Pi Pico. The other terminal of the button is connected to the board's GND.

Another button, identified as Button B, is connected to GPIO6 of the Raspberry Pi Pico. The other terminal of this button is also connected to the board's GND.

Another button, identified as RESET, is connected to RUN (pin number 30) of the Raspberry Pi Pico. The other terminal of this button is also connected to the board's GND.

A passive buzzer, identified as Buzzer A, is connected - through a transistor - to GPIO21 of the Raspberry Pi Pico.

Another passive buzzer, identified as Buzzer B, is connected to GPIO10 of the Raspberry Pi Pico.

The in pin of a 5 rows by 5 columns 5050 RGB LED matrix, type WS2812B (Neopixel), is connected to GPIO7.

An analog joystick, type KY023, has its VRy output connected to GPIO26 and VRx output to GPIO27. Its SW button is connected to GPIO22; the other terminal of the button is on GND.

A 0.96-inch OLED display of 128 columns by 64 rows with I2C communication has its SDA pin connected to GPIO14 and the SCL pin to GPIO15. This display requires loading the ssd1306.py library on the Raspberry Pi Pico.

An electret microphone module with an analog output is connected to GP28. The average level of the output signal is 1.65 V, and the module's output voltage ranges from 0V to 3.3V.

A 14-pin Insulation-Displacement Connector (IDC) box is used for hardware expansion and is connected as follows to the Raspberry Pi Pico: pino 1 com o GND, pino 2 com o 5V, pino 3 com 3V3, pino 4 com GPIO8, pino 5 com o GPIO28, pino 6 com o GPIO9, pino 7 com GND analogico. pino 8 com o GPIO4, pino 9 com o GPIO17, pino 10 com o GND, pino 11 com o GPIO16, pino 12 com GPIO19, pino 13 com o GND, pino 14 com o GPIO18.

A terminal bar allowing the connection of alligator clips is connected to the Raspberry Pi Pico: the DIG 0, 1, 2, and 3 electrodes are connected to GPIOs 3, 2, 1, and 0, respectively. Besides, this terminal bar has five more electrodes connected to the Raspberry Pi Pico's: analog GND, GPIO 28, GND, 3V3, and 5V.

Guidance for programming in Micropython:

We suggest importing the following libraries as needed by the user:

```
from machine import PWM, Pin
import neopixel
import time
import random
from machine import Pin, SoftI2C, ADC
from ssd1306 import SSD1306_I2C
import math
# OLED Configuration
i2c = SoftI2C(scl=Pin(15), sda=Pin(14))
oled = SSD1306_I2C(128, 64, i2c)

# Number of LEDs in your 5x5 matrix
NUM_LEDS = 25
# Initialize the NeoPixel matrix on GPIO7
np = neopixel.NeoPixel(Pin(7), NUM_LEDS)
# Setting the LED matrix
LED_MATRIX = [
```