ME 461 - Let Pico move it

Recall:

Pinout of Pi Pico is also provided as a reference in Figure 1.

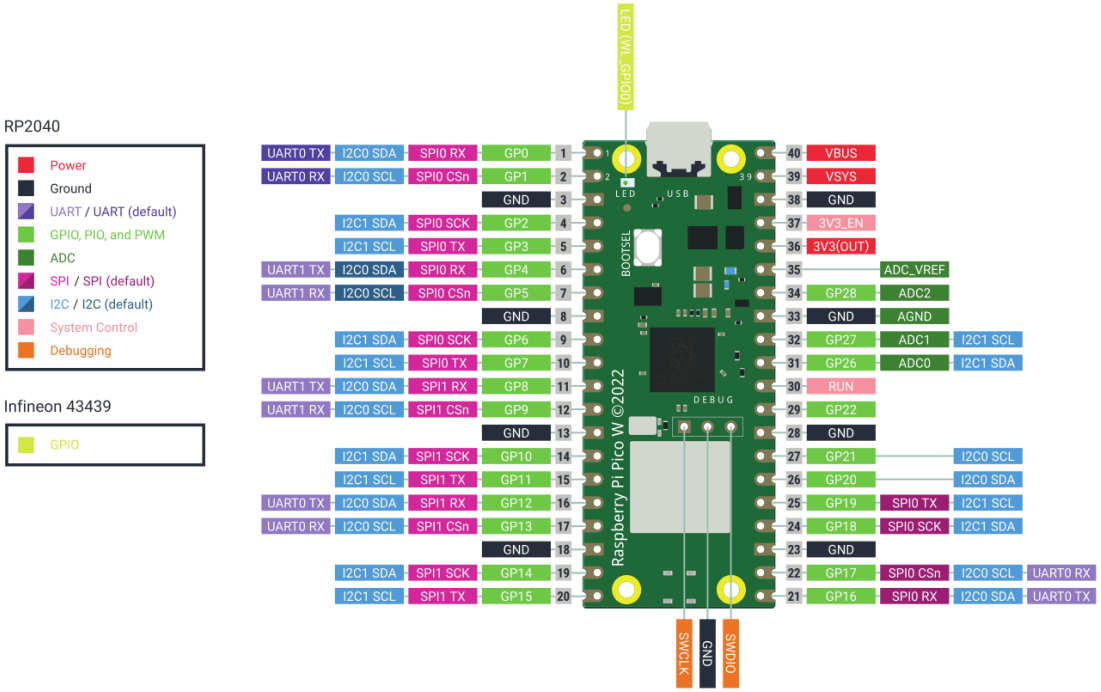

Fig 1: Raspberry Pi Pico W pinout

Rules of thumb: - would not hurt to remember 🙂

- Your code should
  - not crash no matter what
  - include sufficient comments
  - be as lean and short as possible
  - not have self-repeating parts
  - run smoothly and be responsive
- Try to use cut to length wires. Slayed network cables are a good single strand source.
- Everytime you take your breadboard etc out from your backpack or its container, manually check connections to make sure nothing is loose or disconnected. You will get faster in time :)
- Do not modify your circuit while it is powered on
- Do not rush into anything
- If something does not work as expected:
  - it is most probably not a bug but you are missing something.
  - Re-check the cabling and make sure that there are no loose connections
- If you feel a different odor coming from around your breadboard, power it down right away, and do not directly touch things to find out what happened.
- Based on definitions in Labs / Assignments, get used to taking incentives and making reasonable assumptions when certain details are not provided.

What 2 do?

STEP 1: Get ready

Do a literature survey (in this case it is mostly a web search) on how people use L293D, quad half-H-bridge driver.

The pin layout of L293(D) is given in Figure 2. As mentioned in the datasheet (where is the datasheet? hmmm…), L293 is a quadruple half h-drive. By using two half-drives you can implement one full H-bridge. L293 and L293D do have the same footprint, however, L293 has an output current of 1A per channel whereas the limit is 600mA for L293D. The 'D' indicates that *flyback diodes* are included in the package; therefore, you do not need them in your circuit. By the way, what is a flyback diode?

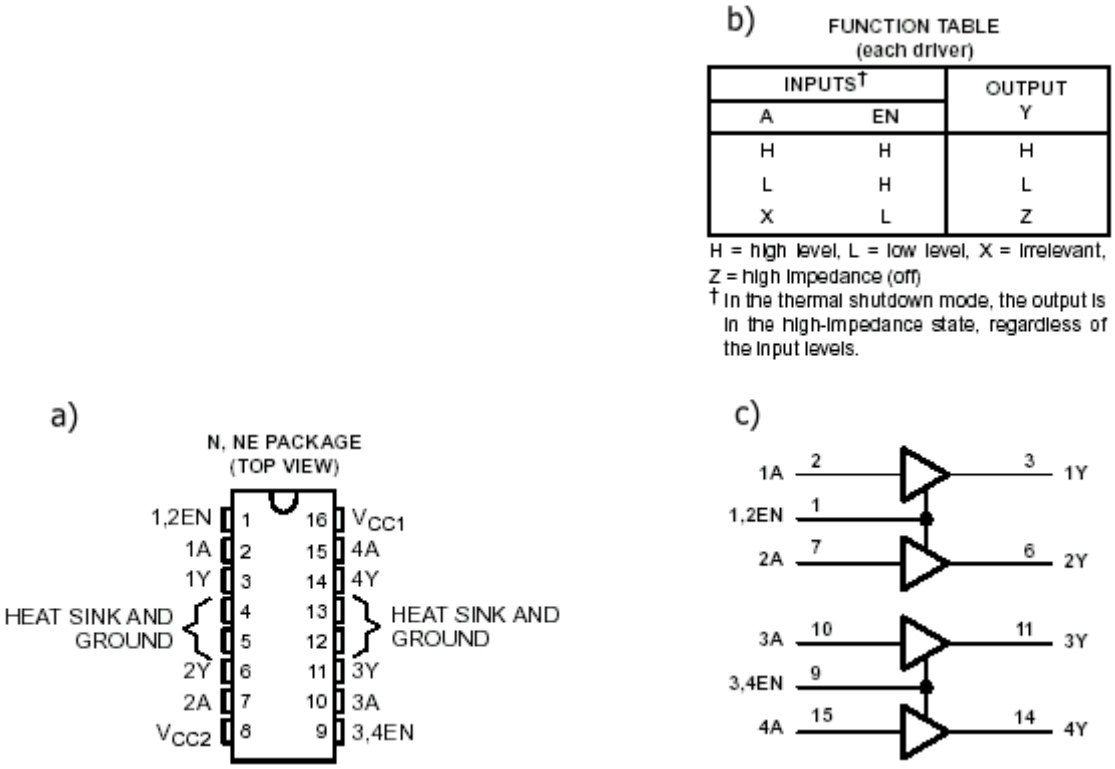

Figure 2. L293(D) a) Pin Layout b) function table c) logical diagram

With one half H-drive (or half-bridge) you can turn an inductive load ON or OFF. This means that if you are controlling a motor, you can only control the motor in one direction. This implies that you can achieve unidirectional speed control of 4 motors with a single L293 IC, yet you cannot change their directions. This also implies that you can control 4 relays, or a 4 coil stepper motor by using these 4 half bridges separately. Or you can use two full bridges, interface two motors to this IC and control both the speed and the direction of rotation of these motors. The pin layout of the IC is organized nicely so that two half bridges are located on opposite sides. On each side, these half bridges are connected through a common enable pin by the use of which you can create a full-bridge when needed. The functional table is not very complicated. If the bridges are enabled, the logic value at the input pins (A) are reflected to the corresponding output pins (Y). In case the pins are disabled (by pulling the enable pin to low), the outputs are inhibited.

For small-scale robotic implementations, L293(D) is a practical and a low-cost solution. One way to drive loads that demand greater amounts of currents, you can combine bridges or choose another IC. One common way to combine bridges is to solder two ICs on top of each other. However, if you need to go over 1-1.5 A, then consider a proper driver designed for such

loads since cooling might easily become your major problem if you use L293. If you push your L293 to its limits, cooling might easily become an issue. In such a case check the datasheet, find out proper pins, and find a creative way to dissipate heat through these pins. For dealing with more powerful motors, L298 or LMD18200 driver chips are two common alternatives among several others. The worst-case scenario is that you might need to build your own H-bridge(s), yet this solution is NOT recommended.

Let's take a look at the practical side and check out the application information given Figure 3. Use of both half- and full-bridges are illustrated in this example to drive motors in either unidirectional or bidirectional fashion. As mentioned above, the flyback diodes are not necessary along with L293D since they are present inside this IC.

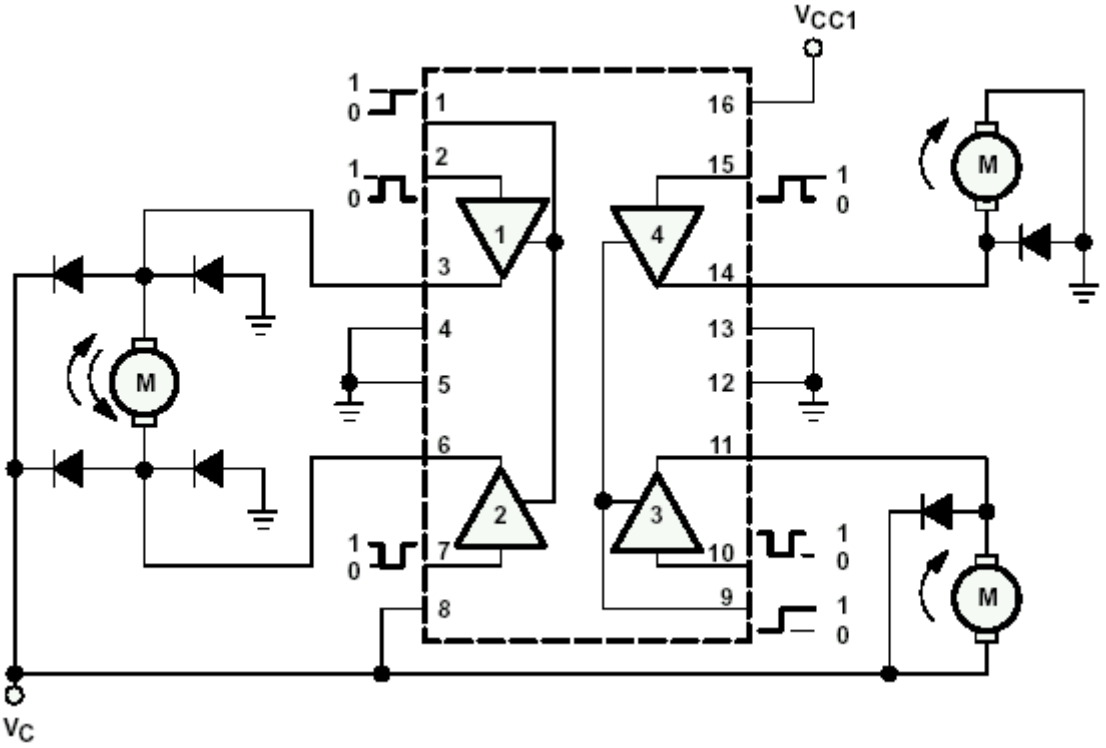

Figure 3. Possible use of half- and full-bridges.

In this lab, you will implement a full-bridge and control the speed of a motor via pulse-width-modulation (PWM). Before starting the lab, please make sure that everything is clear about what have been presented so far.

STEP 2: Safety first - LED motor

Despite the appeal, do NOT start playing with the DC motor right away. A safe play is, at least to visually test your code, is to use a simple circuit, i.e an LED motor as shown in Figure 4.

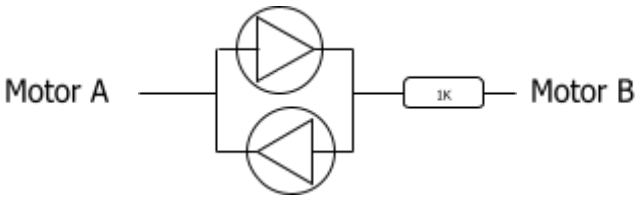

Fig 4. Schematic of a LED motor.

Build an LED motor and check if your motor works: Interface it to L293D as Motor A on our PCB board. Trace the pins from the board to figure out which Pi Pico pins are mapped to which L293D pins. Using ButtonL and ButtonR switches (recall from past labs) change direction, using the Pot, generate a PWM signal to change the intensity of the LEDs on the LED motor. If everything works as expected, attach the real motor as Motor B and test it as well.

STEP 3: Seemingly easy part - GUI

On your PC develop a Python GUI that is similar to what is shown in Figure 5.

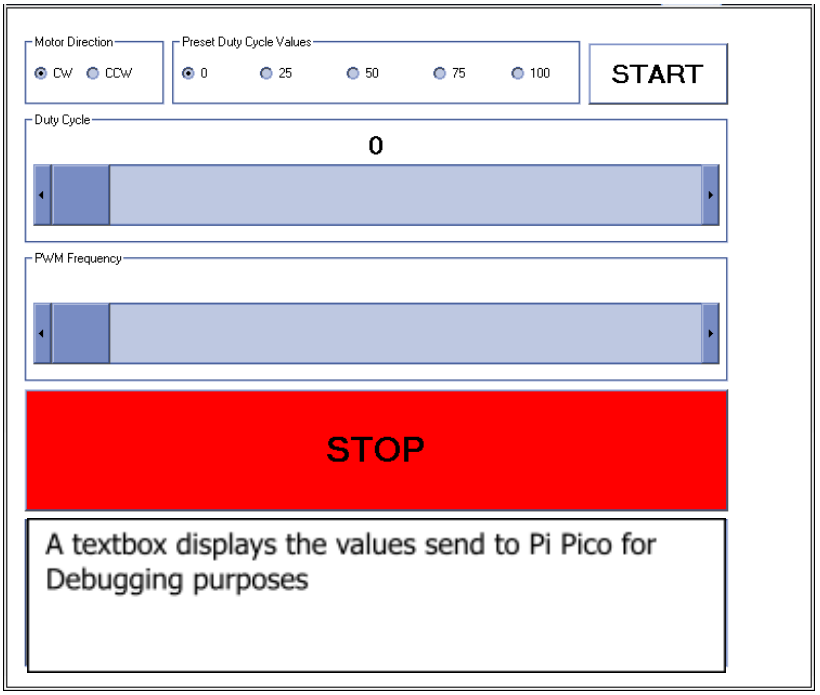

Figure 5. Graphical User Interface to be used in this experiment.

Now shown in this GUI but you can also add a selector for Motor A and Motor B as a bonus. If not added, this GUI is to work with Motor A by default.

STEP 4: Play with the GUI

Now play with the GUI and make sure it works. Among all settings you can change via the GUI the most interesting one should be the PWM frequency. Try to figure out what happens when you change it? What are the meaningful ranges for both L293D and the motor you use?

RULES of the GAME: First thing last

Have fun!