

e-skin lab @ Humanoids

# What we will make

- e-skin based on Velostat foil



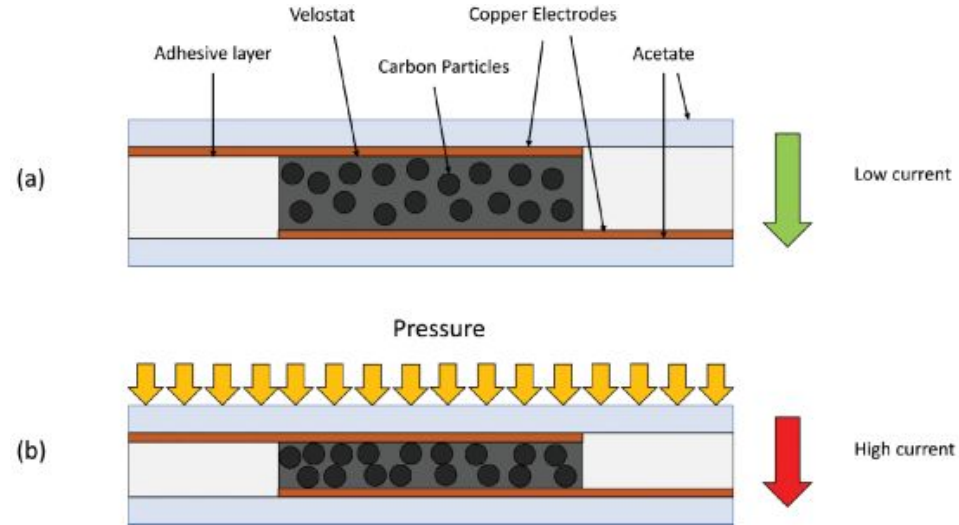
# Velostat

- Velostat = Polymeric foil filled with conductive carbon particles
- Primary developed for packaging
- Piezoresistive = Resistance dependent on deformation



# Velostat as a sensor

- Electric conductivity thanks to
  - Percolation traces (randomly formed conductive paths)
  - Tunneling of electrons (even though particles are insulated by the polymer, they can still tunnel through with certain probability)
- **When pressed:**
  - particles moves closer together → more percolation traces + smaller distance between insulated particles → more current flowing → **lower resistance**



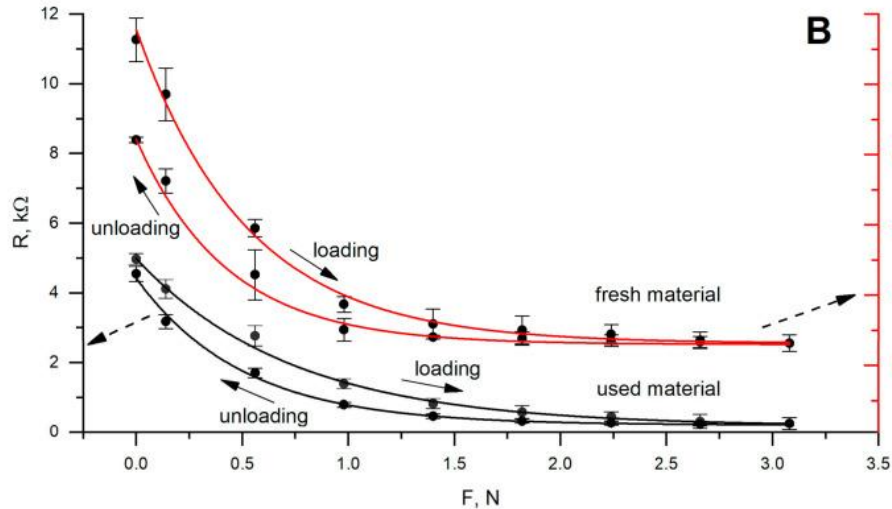
# Additional sources of piezoresistivity - Surface effects

- Skin patch is warped
  - with pressure contact area between velostat and electrodes increases
- Roughness of surfaces
  - with pressure microscopic structures are deformed increasing contact area

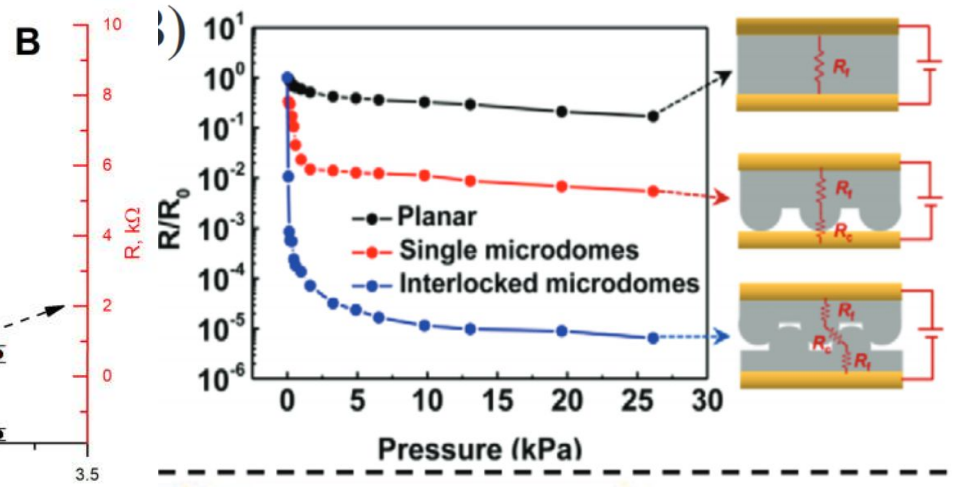
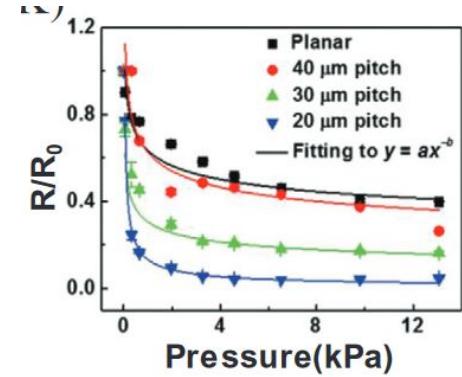


# Typical response

- Generally shown as R to pressure or force
- Typically relative resistance  $R/R_0$  is used



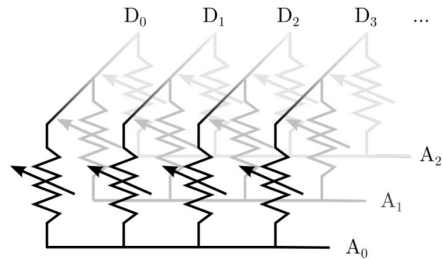
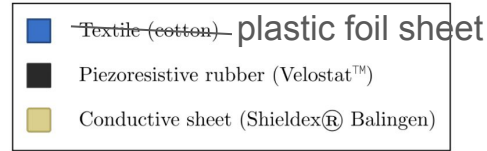
Velostat



SoTA microstructural sensors

# Construction - Resistive array

- Sandwich structure
- Row and column electrodes
- One sheet of velostat
- Insulation layers



Custom readout PCB secured on PLA holder, powered by 160 mAh LiPo battery

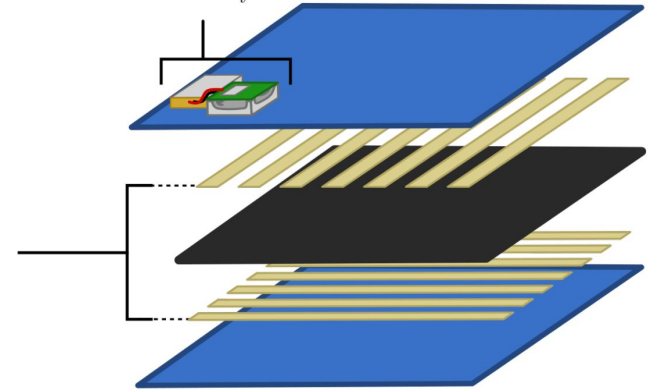
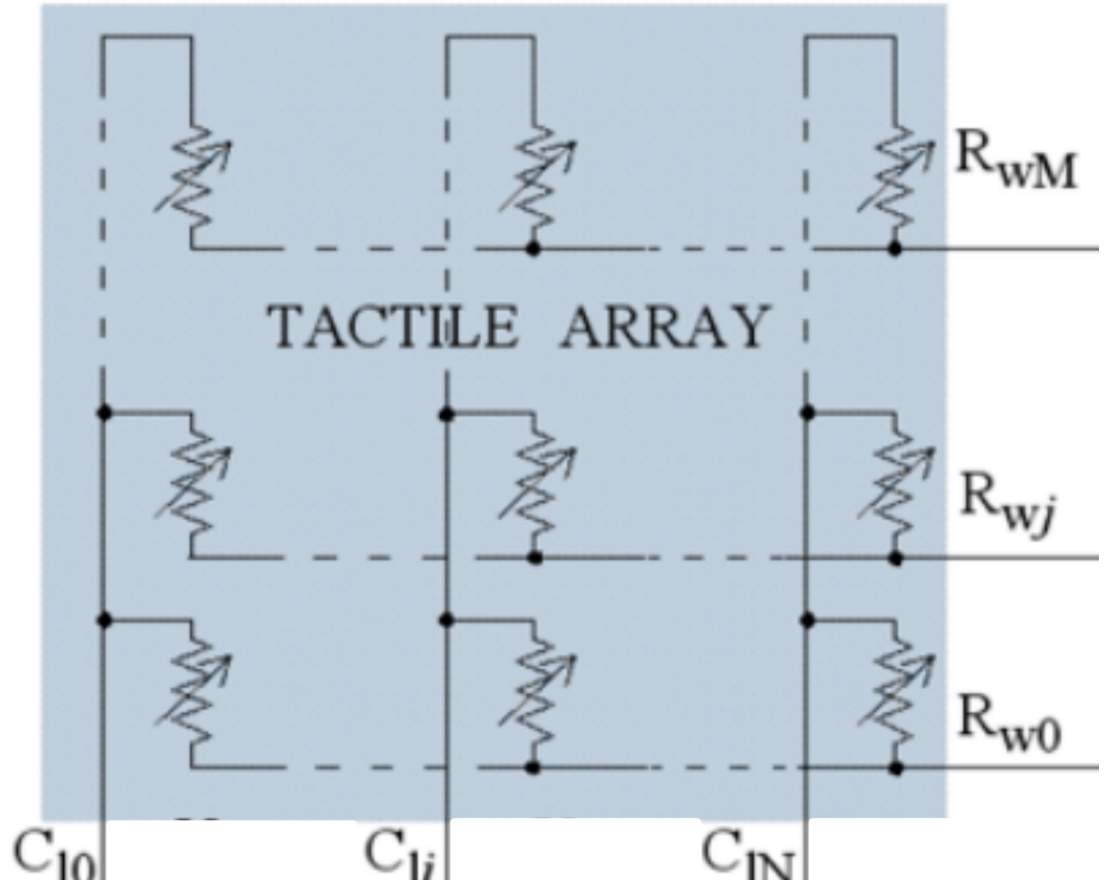


Image illustrative, taken from literature

# Equivalent schema

- Array of row and column lines
- Similar to keyboard or memories
- Special electronics to measure individual taxels (later in presentation)



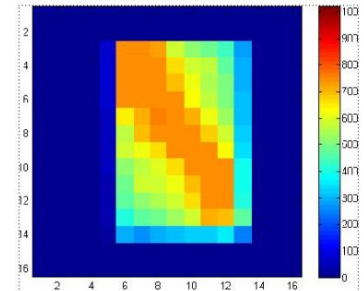
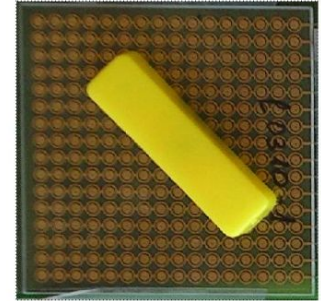
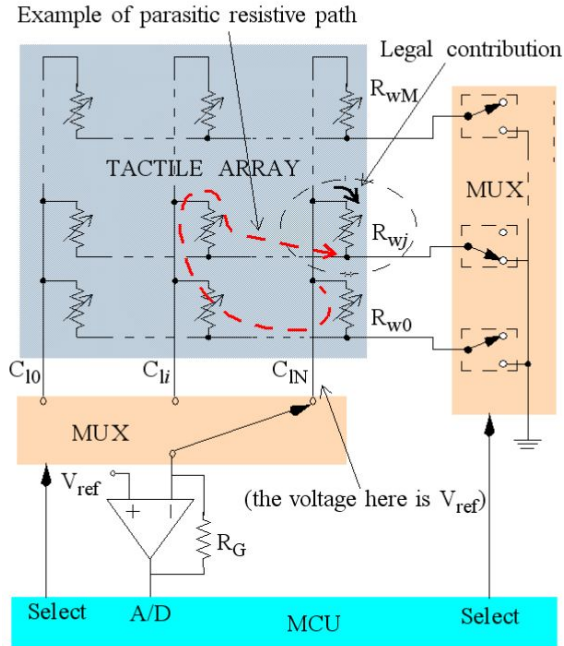


# Manufacturing video

This place will be used soon

# Naive measurement - Crosstalk current

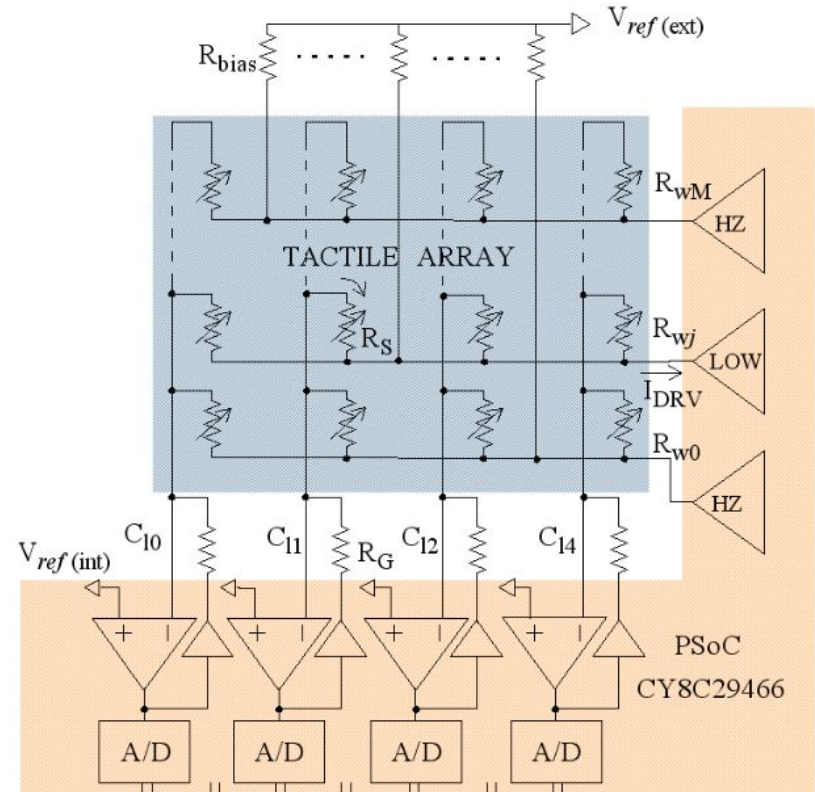
- Row and columns selecting multiplexers
- Pass current through circled resistor (black dashed)
- Current however can flow also through all the other resistors (red dashed line)
- Usable but we get distorted result





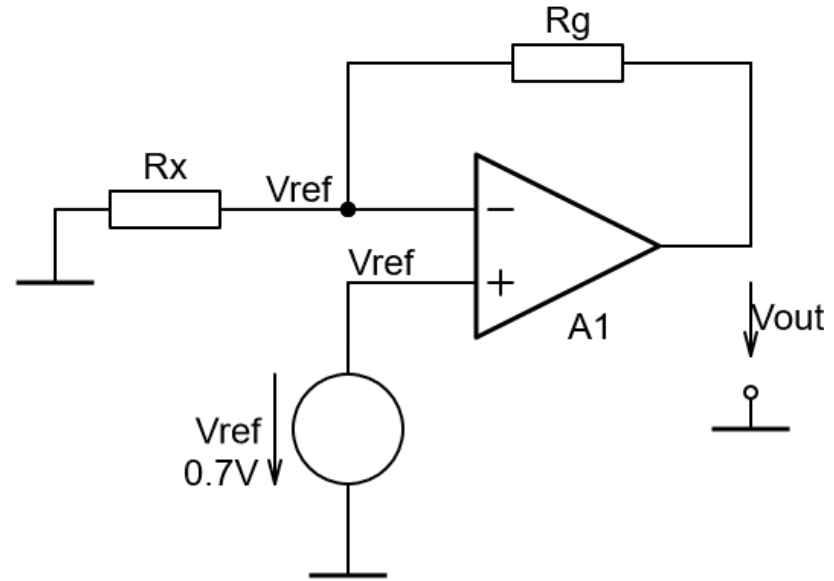
# Real Implementation

- Pull-up resistors to  $V_{ref}$
- Mux switches between GND and High Impedance
- Each op amp sampled by analog to digital converter



# What you really need to know

- We switch between rows using multiplexer
- To select row in MicroPython use function `set\_row`
- Columns are sampled by ADC
- Thanks to operational amplifiers (assuming ideal) we can simplify to single resistor measurement



$$V_{out} = \left( \frac{R_g}{R_x} + 1 \right) V_{ref}$$

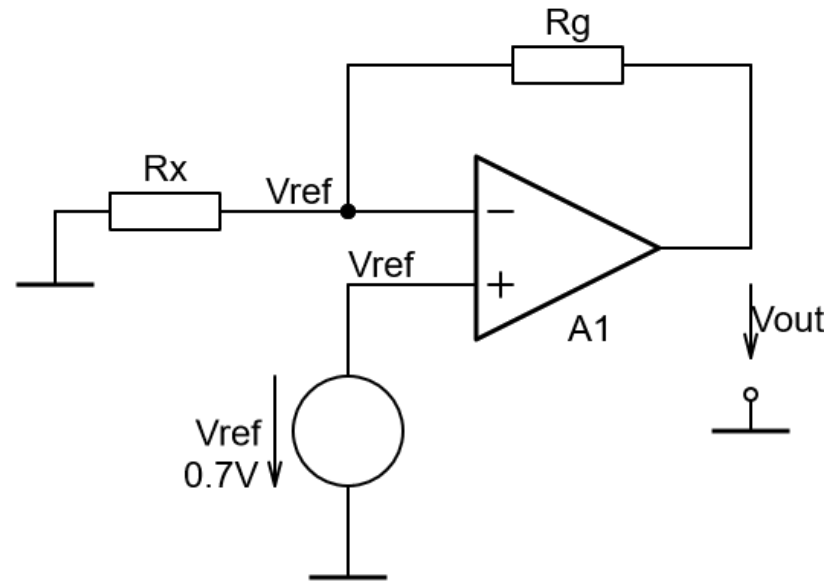
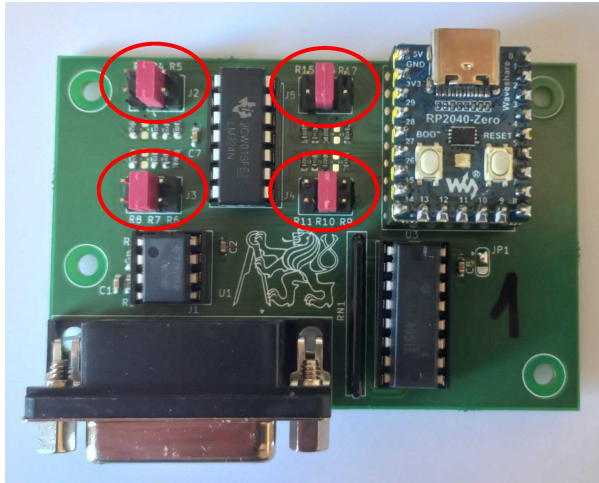
$R_x$  → Measured resistor

$R_g$  → Known (gain) resistor

$V_{ref}$  → Voltage reference (**0.7V**)

# What you really need to know

- Selection of  $R_n$  can be done using jumper on board
- Values (from center → out):
  - 1 k $\Omega$
  - 4.7 k $\Omega$
  - 10 k $\Omega$



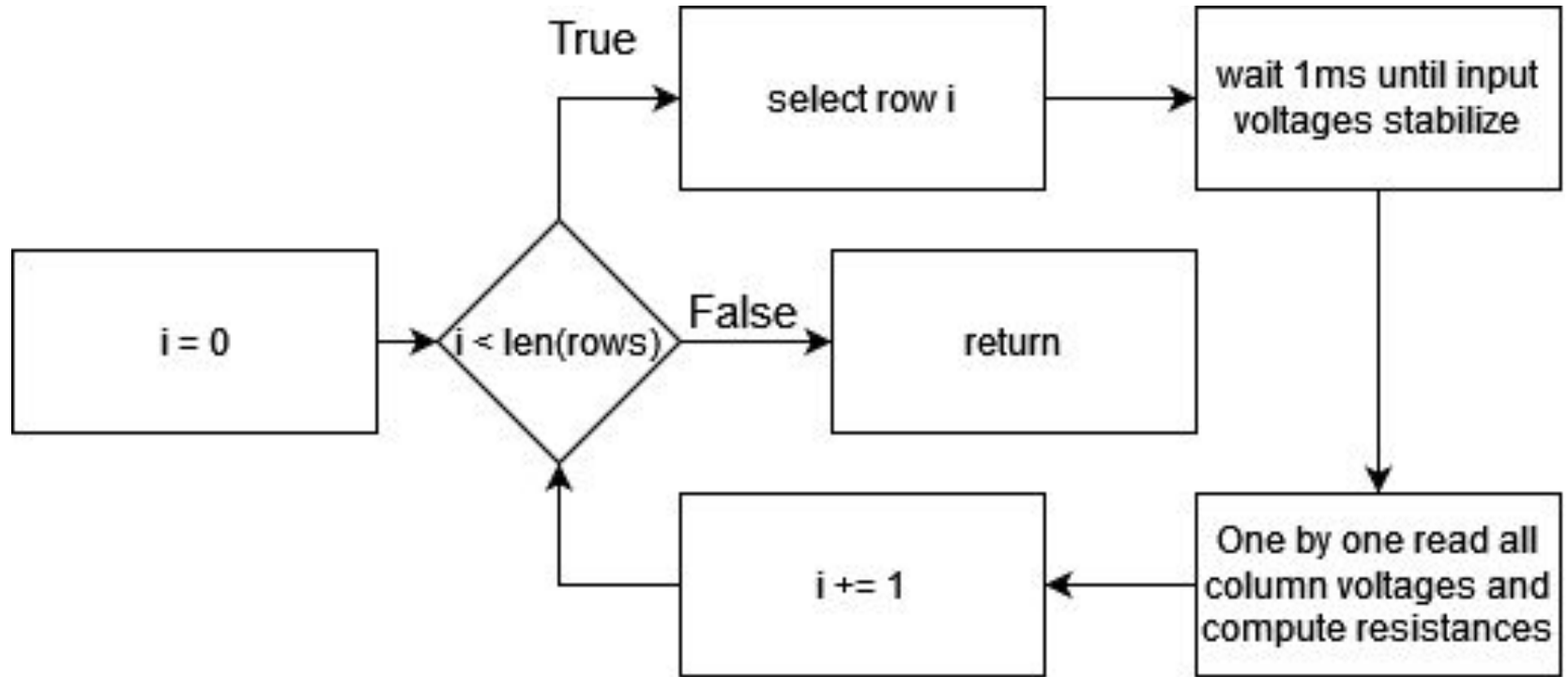
$$V_{out} = \left( \frac{R_g}{R_x} + 1 \right) V_{ref}$$

$R_x$  → Measured resistor

$R_g$  → Known (gain) resistor

$V_{ref}$  → Voltage reference (**0.7V**)

# Read out algorithm and timing



# Crash Course to MicroPython (MP)

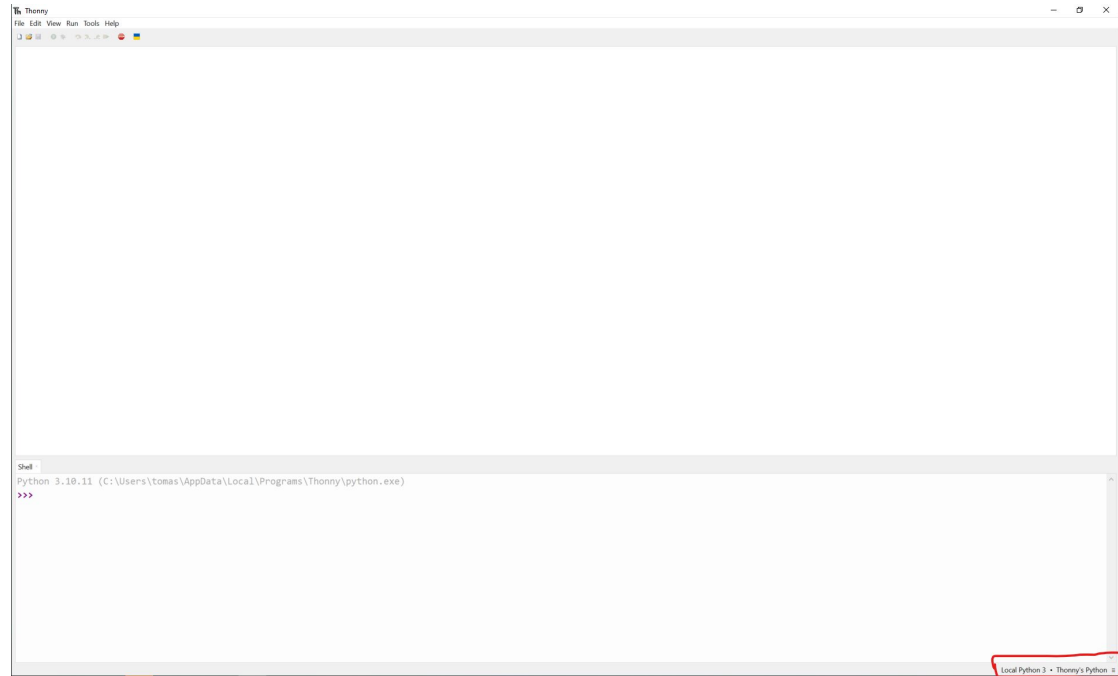
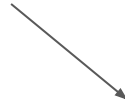
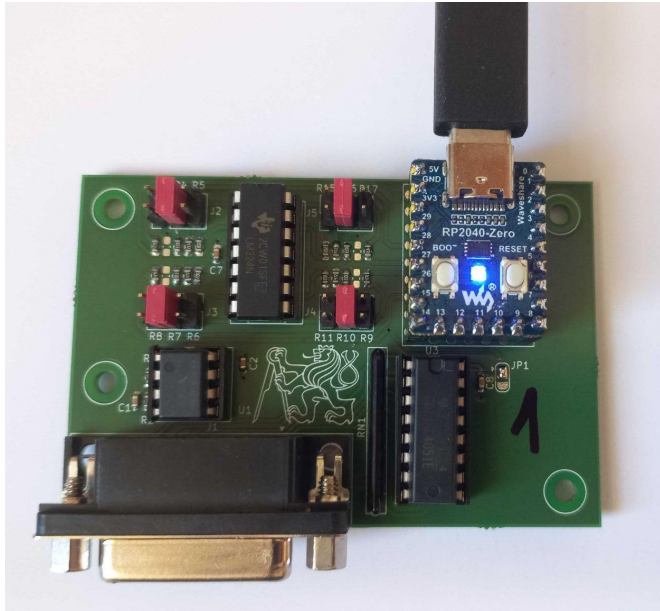
- MP = simple Python compiled for embed HW
- All hardware peripherals are objects with certain methods from `machine` module
- **Sadly no standard objects (things differ between platforms)**
- For your script to run automatically, it must be named **main.py**
- To edit script loaded in chip's memory we use Thonny IDE



**MicroPython**

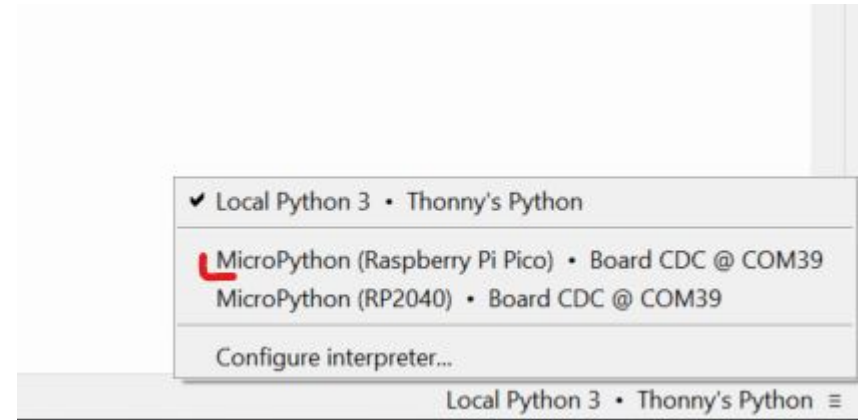


# How to edit MP script on HW using Thonny

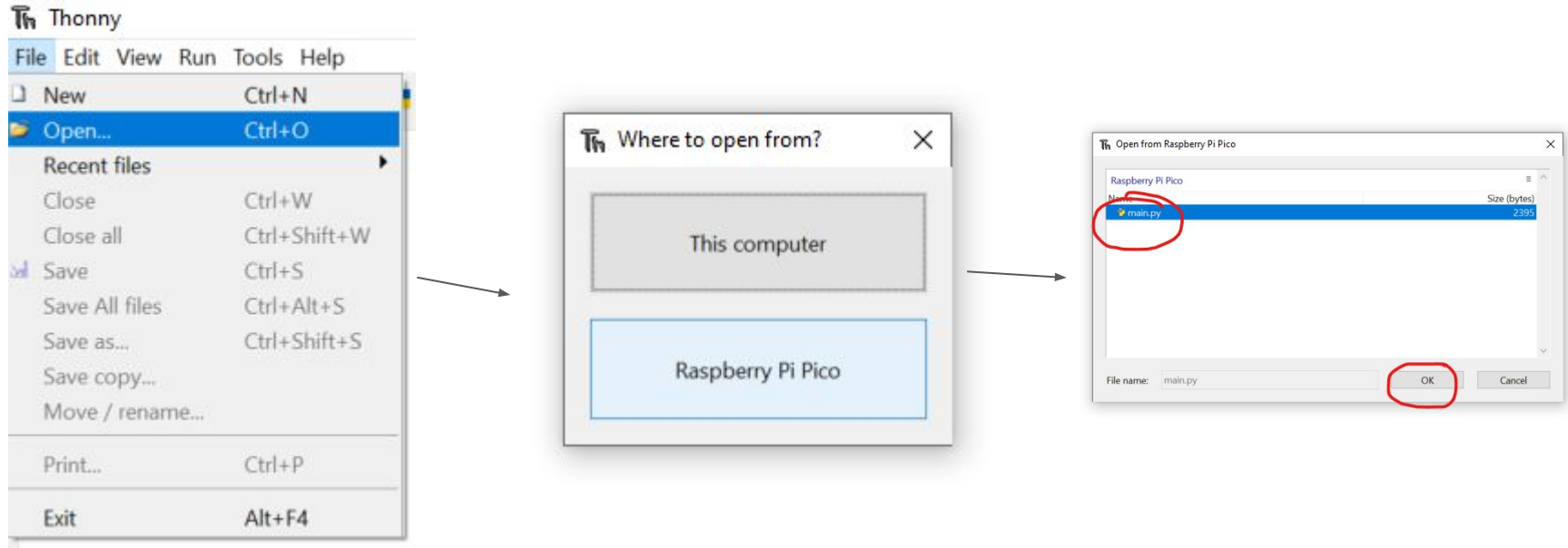


# How to edit MP script on HW using Thonny

Select marked interpreter and note what is RP2040 listed as (COM39 in example), you will need this for Python scripting in your homework

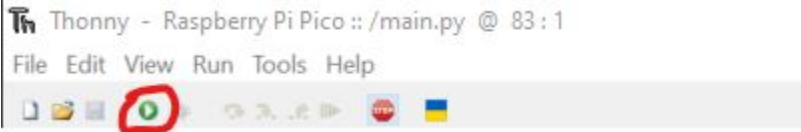


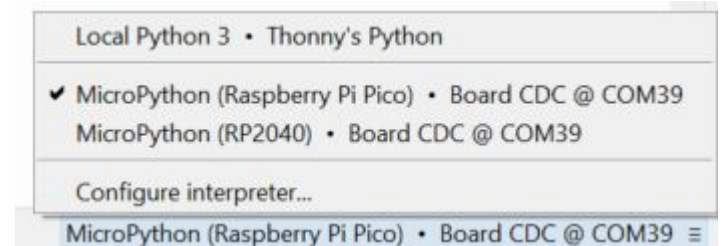
# How to edit MP script on HW using Thonny



To store your progress, just hit Ctrl+s, it will store to RP2040 memory

# How to run MicroPython script

-  Thonny - Raspberry Pi Pico :: /main.py @ 83:1  
File Edit View Run Tools Help  
The screenshot shows the Thonny IDE interface. The title bar reads "Thonny - Raspberry Pi Pico :: /main.py @ 83:1". Below the title bar is a menu bar with "File", "Edit", "View", "Run", "Tools", and "Help". Underneath the menu bar is a toolbar with several icons. The "Run" icon, which is a green circle with a white play button symbol, is circled in red.
- Press reset button on board (this restarts board and main.py is automatically executed)
  - note resetting board causes it to disconnect from PC and must be manually reconnect even though Thonny shows board being connected
- While running Python script reading data from board (f.e.: example01\_visualizaton.py) either close Thonny or set it to local interpreter to free serial port



# General Purpose Input Output (GPIO) in MP

- used for: buttons, digital communication, LEDs

```
1 # example code to blink led with pin 14
2 import machine
3 import time
4
5 led = machine.Pin(14, machine.Pin.OUT) # set pin no. 14 as digital output
6 while(true): # infinite loop sou our code never stops
7     led.value(true) # drive output pin High (light on)
8     time.sleep(0.1) # sleep for 100ms
9     led.value(false) # drive output pin Low (light off)
10    time.sleep(0.1) # sleep for 100ms
```

## Read analog values in MP

$$V = \frac{n}{2^N - 1} V_{ref}$$

- for some weird reason ADCs in MP returns 16 bit numbers with maximum of 65535 even if given ADC is 12 bit

$n$  → ADC output number

$N$  → ADC bit resolution (**16 bit** even though ADC is only 12 bit)

$V_{ref}$  → ADC reference (**3.3V**)

```
1 # periodically reads voltage values from 4 analog inputs
2 import machine
3 import time
4
5 adcs = [] # array of ADCs objects
6 adcs.append(machine.ADC(3)) # add adc channel into a list
7 adcs.append(machine.ADC(0))
8 adcs.append(machine.ADC(1))
9 adcs.append(machine.ADC(2))
10
11 voltage = [0,0,0,0] # prepare space for masured voltage
12 while(true):
13     for i in range(4):
14         voltage[i] = adcs[i].read_u16()/65535*3.3 # read analog value and convert to voltage
15         time.sleep(10e-3) # sleep for 10ms between samples
```

# Bibliography

- M. Hopkins, R. Vaidyanathan and A. H. McGregor, "Examination of the Performance Characteristics of Velostat as an In-Socket Pressure Sensor," in *IEEE Sensors Journal*, vol. 20, no. 13, pp. 6992-7000, 1 July 1, 2020, doi: 10.1109/JSEN.2020.2978431. keywords: {Loading;Sensor phenomena and characterization;Sockets;Prosthetics;Temperature sensors;Mechanical sensors;Piezoresistive measurement;pressure sensing;prosthetics;prosthetic fitting;Velostat;wearable sensors}
- Dzedzickis A, Sutinyas E, Bucinskas V, Samukaite-Bubniene U, Jakstys B, Ramanavicius A, Morkvenaite-Vilkonciene I. Polyethylene-Carbon Composite (Velostat®) Based Tactile Sensor. *Polymers (Basel)*. 2020 Dec 3;12(12):2905. doi: 10.3390/polym12122905. PMID: 33287414; PMCID: PMC7761878.
- Tang, R., Lu, F., Liu, L., Yan, Y., Du, Q., Zhang, B., Zhou, T., & Fu, H. (2021). Flexible pressure sensors with microstructures. In *Nano Select* (Vol. 2, Issue 10, pp. 1874–1901). Wiley. <https://doi.org/10.1002/nano.202100003>
- Proesmans, R.; Verleysen, A.; Vleugels, R.; Veske, P.; De Gusseme, V.-L.; Wyffels, F. Modular Piezoresistive Smart Textile for State Estimation of Cloths. *Sensors* **2022**, 22, 222. <https://doi.org/10.3390/s22010222>
- VIDAL-VERDÚ, Fernando, et al. Three realizations and comparison of hardware for piezoresistive tactile sensors. *Sensors*, 2011, 11.3: 3249-3266.