Saana Vallius
Internet of Things
Spring 2025

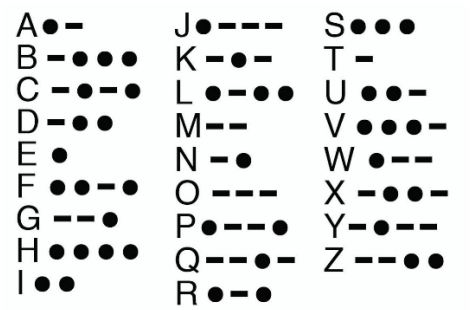
https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico

https://thonny.org/

Programming Exercises 1

Exercise 1.1

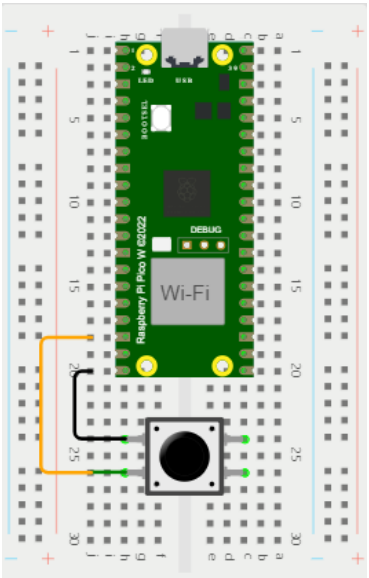
Create a program that blinks the on-board LED to show the SOS sequence in morse code. Each long character (dah) should last for 1500ms while each short character (dit) should last for 500ms. There should be a 500ms break between characters and a 1500ms break between letters. After each message, there should be a 3500ms break and the board should start showing the SOS code again.



Exercise 1.2

For this exercise you need to wire a push button to your Pico board. Make sure your Pico board is not powered before making any wiring changes.

1. Attach a push button over the junction of the breadboard so that there are two pins on both sides of the junction.
2. Connect the bottom left pin of the push button to ground (GND) on pin 18 on the Pico W.
3. Connect the top left pin of the push button to pin GP15 on the Pico W.
4. The connection should look as follows:



5. The following program sets up pin 15 as an input pin and enters a loop where the status of the button is checked on each round. If the button is pressed, “Button was pressed” is printed out to the console. Copy and run the program in Thonny to investigate how it works.

```
from machine import Pin
import time

button = Pin(15, Pin.IN, Pin.PULL_UP)

while True:
    if button.value() == 0:
        print("Button was pressed")
    time.sleep(0.1)
```

6. Modify the program so that it detects each button press exactly once. Add a variable that counts the button presses and change the print to show how many times the button was pressed.

Exercise 1.3

Write a program that waits for the push button to be pressed. When the button is pressed, the program generates a random number of the dice (1-6) and prints the number to the output console. Each button press must be detected exactly once. Documentation for the MicroPython random library can be found here: https://micropython.org/resources/docs/en/latest/library/random.html.

Programming Exercises 2

Exercise 2.1

Write a program that writes your name in the center of the OLED display when the button is pressed. When the button is pressed again, the screen is emptied. The correct center point of the display must be calculated.

Exercise 2.2

Write a program that reads user input and prints out user messages on the OLED display. Each message should appear on a new line. When the screen is full, the messages should scroll up on the display to always show the newest messages, and the latest message always appears on the bottom of the display. Pressing the button empties the display.

Exercise 2.3

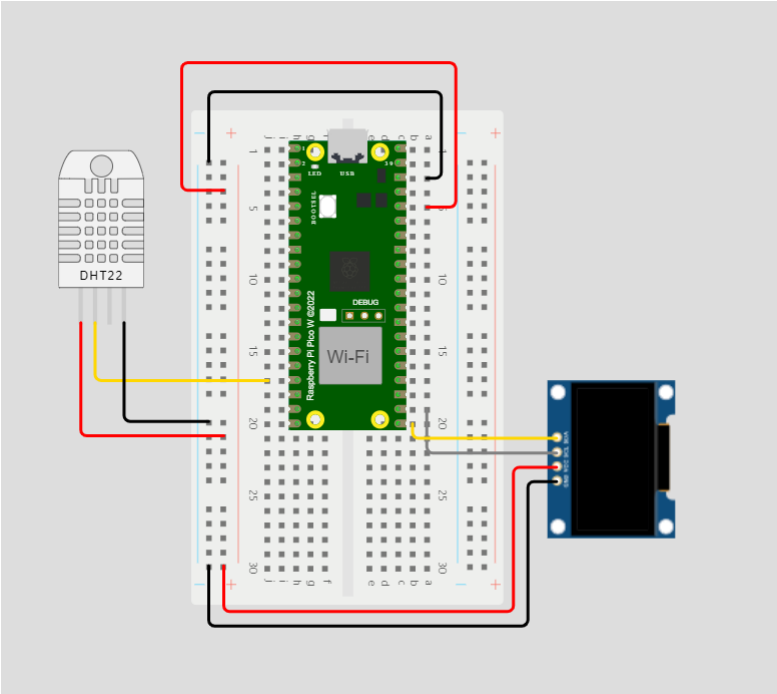
Extend the dice throwing exercise. Write a program that waits for the push button to be pressed. Each button press must be detected exactly once. When the button is pressed, the program generates a random number of the dice (1-6). This time, the side of the dice should be drawn on the OLED display.

Programming Exercises 3

Exercise 3.1

Write a program that reads the temperature and humidity values from the DHT sensor once per second and prints the values to the OLED display. The temperature and humidity values should be printed on separate lines with descriptive names.

Here is how to connect the DHT22 sensor and the OLED display to the Pico board using the power rails of the breadboard:



Exercise 3.2

The Raspberry Pi Pico board has an internal analog temperature sensor. Write a program that reads the temperature values and prints both the AD value and the temperature values to the console once every 2 seconds.

To calculate the temperature value, you must first convert the digital AD value to voltage.

- Resolution: 16 bits
- Reference voltage: 3.3 V

The you need to use the transfer function for the sensor to calculate the temperature:

$$T = 27℃ - \frac{V_{out} - 0.706V}{1.721 \frac{mV}{℃}}$$

What is the smallest detectable change in temperature measured by the sensor?

Exercise 3.3

Modify the program from the previous exercise by calculating the floating average of five consecutive temperatures. Print the result to the OLED display once per second.

Hint: Save the values into an array or list with 5 elements. Look into how you can **pop** the first item of a list to always have the 5 newest values stored.

Programming Exercises 4

For this week's exercises you will need to connect the Raspberry Pi Pico to the local network and through that to the Internet. You will also need to install an MQTT client such as MQTTX to subscribe to messages sent from the Pico board. MQTTX can be downloaded from here: https://mqttx.app/. MQTTX also has a command line tool that can be downloaded from here: https://mqttx.app/cli.

SSID: SmartIoTMQTT
Wi-Fi password: SmartIot

Note that some of the exercises only work in the local lab network specified here. You will find example code for the exercises in this GitLab repository: https://gitlab.metropolia.fi/saanapi/hardware-1-networks-final-assignment.

The repository contains the following files:

- **connect_to_wlan.py** You can use the program to connect the Raspberry Pi Pico W to the specified WLAN.
- **install_mqtt.py** You can use the program to install MQTT client to the Raspberry Pi Pico.
- **mqtt_publish_test.py** The test connects the Pico W to the specified WLAN, opens an MQTT connection and periodically sends MQTT messages.

To use the example programs, you must replace the default values of the following variables at the top of the programs:

```
SSID = "WLAN SSID"
PASSWORD = "WLAN PASSWORD"
BROKER_IP = "192.168.1.254"
```

Exercise 4.1

Connect your Raspberry Pi Pico board to the local Wi-Fi network. While the Pico board is trying to connect to the network, the message “Connecting…” should be shown on the OLED display. Once the connection has been established, display message “Connected” as well as the IP address of the Pico board on the OLED display.

Exercise 4.2

Write a program that reads temperature and humidity values from the DHT22 sensor and publishes the messages over the local MQTT connection. The messages should have a separate topic for humidity and temperature data, and you should use your name as the top-level topic. Example topic: “/saana/temperature”. The MQTT messages should be sent once every 5 seconds. Subscribe to your own topic on an MQTT client on your own computer.

Exercise 4.3

Write a program that asks the user to input a Pokémon ID number between 1 and 1025. The number is used to fetch the name of the Pokémon corresponding to the ID number. The Pokémon names are fetched from PokéAPI at https://pokeapi.co/. Use the **pokemon-species** endpoint for the API calls. Documentation for the endpoints can be found here: https://pokeapi.co/docs/v2.