



Lesson 4:

Environmental Monitoring

Block-based Programming

Drones4STEM © 2023 by Drones4STEM is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-sa/4.0/>

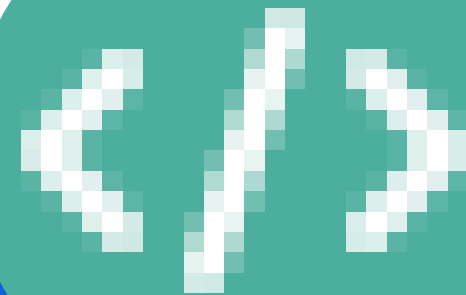


Co-funded by
the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Lesson Objectives

Learning the basics of
block-based programming



SCRATCH

Learn how to use Scratch and Mind+ to
program the mission!

Test yourself by programming a drone
mission!



Introduction to Block-based Programming

What is block-based programming?

Definition: Visual coding method for programming without syntax, by connecting instructions in the form of blocks

Main Purpose: Being a beginner-friendly approach to programming by focusing on the core concepts of structure, logic and problem solving



Why use block-based programming?

1 Simplifies coding

- Lets beginners write perfectly complete programs without worrying about the specifics of syntax and other complexities of text based programming.

2 Visual Learning

- Allows coders of any age to visually understand programming structure and connections through block representation

3 Encourages experimentation and creativity

- With such an easy to use tool, inexperienced coders are quickly free to try new ideas and test their problem solving skills

Tools for Block-Based Programming

1 [Scratch](#)

- A popular block-based programming language with its own development environment. Free to use, beginner-friendly, with many pre-made applications that can be found on the official website!

2 [Mind+](#)

- Mind+ is a free to use, Scratch 3.0-based programming software that supports various open-source hardware such as Arduino, micro: bit, mPython-esp32 and all kinds of controller boards.
- Mind + supports dozens of commonly-used extension module libraries, which make it suitable for drone programming.

Types Of Blocks

1

MOTION

It allows you to position and move the sprite within the work area.

2

LOOKS

It allows you to change the appearance and let the sprite speak.

3

SOUND

It allows you to introduce sound effects.

4

EVENTS

It allows to start the program according to the different events and manages the messages.

5

CONTROL

It allows you to make loops and insert conditional blocks.

6

SENSING

It allows you to manage interactions.

7

OPERATORS

It allows you to perform mathematical and comparison operations.

8

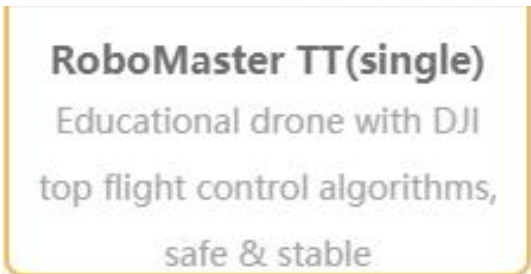
VARIABLES

It allows you to create and manage variables.

Block-based Programming and Drones

Block-based programming can be used to program drone devices, directing them to take different flight paths, perform take-offs and landings, use sensors, take pictures and more.

To program the drone Robomaster TT in Mind+ you must first install the correct extension:

Choose  then  then 

In Mind+, the Blocks used to program drones are

and can be found in the  section after installing the extension.

Simple Example of Block-based Drone Programming

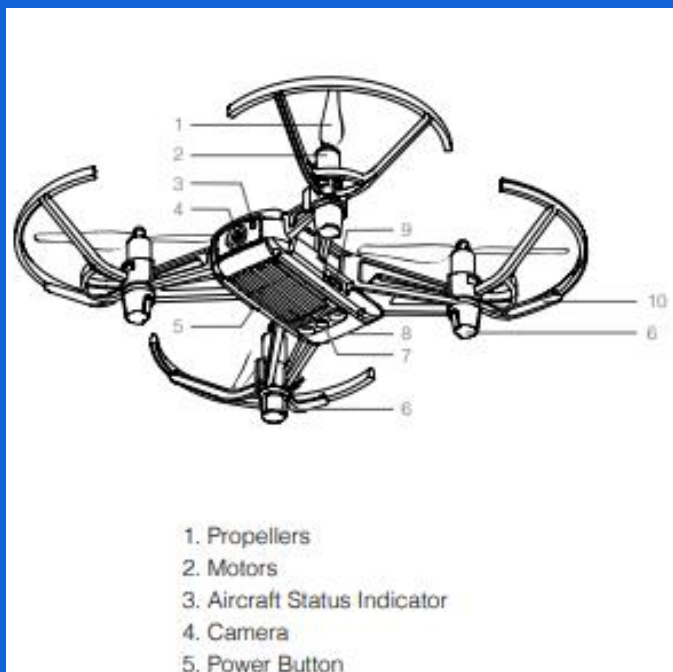
Blocks connect with each-other like a puzzle.

The programmer can drag and drop blocks in order to create their program.

This simple example shows how easily blocks can be used to program a drone to: take off, fly left for 50cm, fly right for 50cm, repeat that 10 times, and then land.



Use Case 2 Mission: Connecting the Drone



Step 1

Press the power button of the drone.

Connect to the drone through Wifi, to the network TELLO or RMTT



Step 2

Step 3

Go to



in Mind+ and press the



Connect to the device through Mind+



Step 4

Use Case 2 Mission: Drone programming and Sensors

Purpose:

The goal of the mission is to:

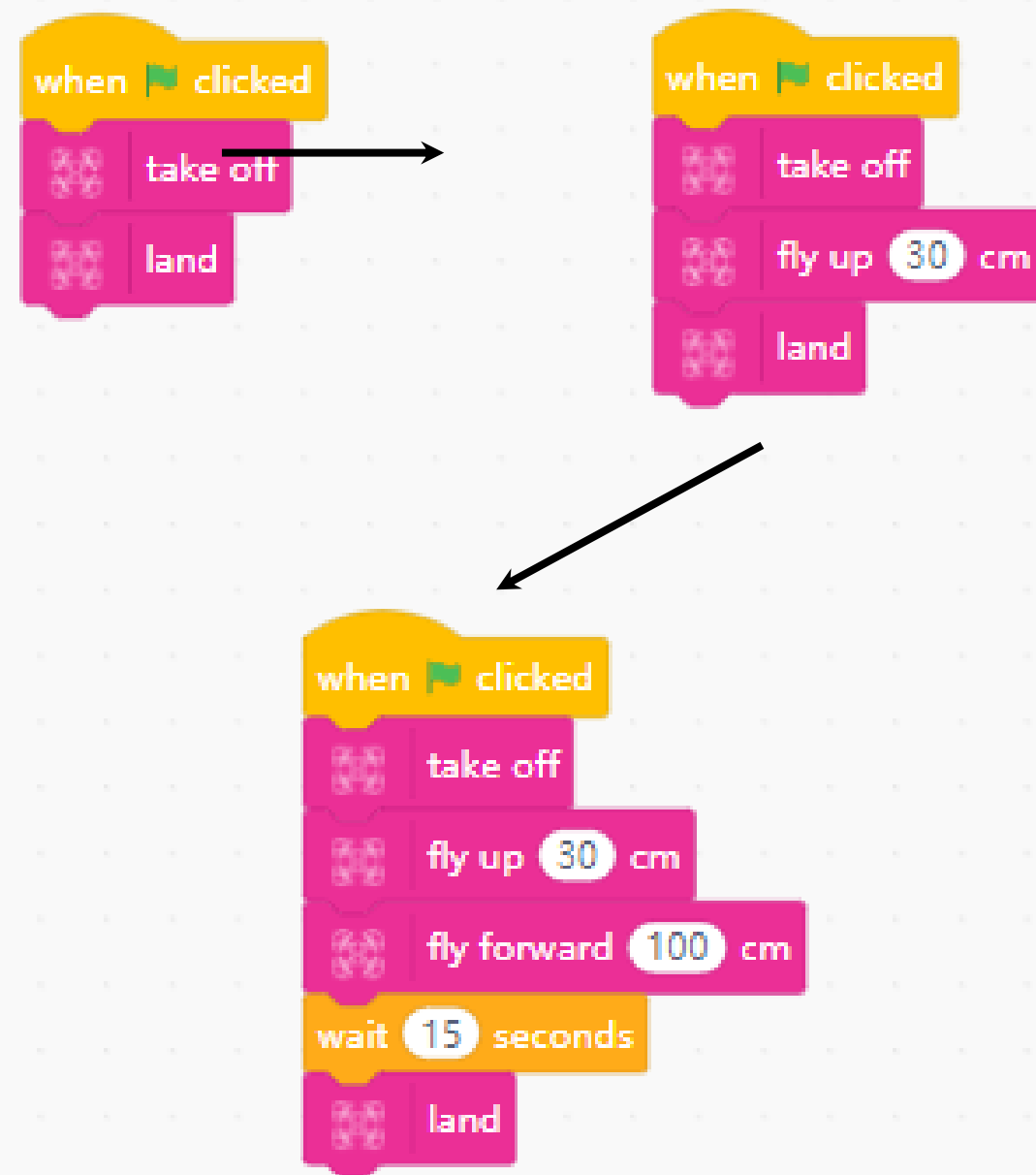
- Teach block-based drone programming
- Teach use of sensors with drones
- Teach data collection and analysis of sensor data

In this programming lesson we will:

- Program a Robomaster TT drone using Mind+
- Collect data from the Drone's Internal sensors
- Organise the data in lists to help data analysis

Use Case 2 Mission: Drone Flight basics

(1)



Starting off

Using simple commands found in the



and

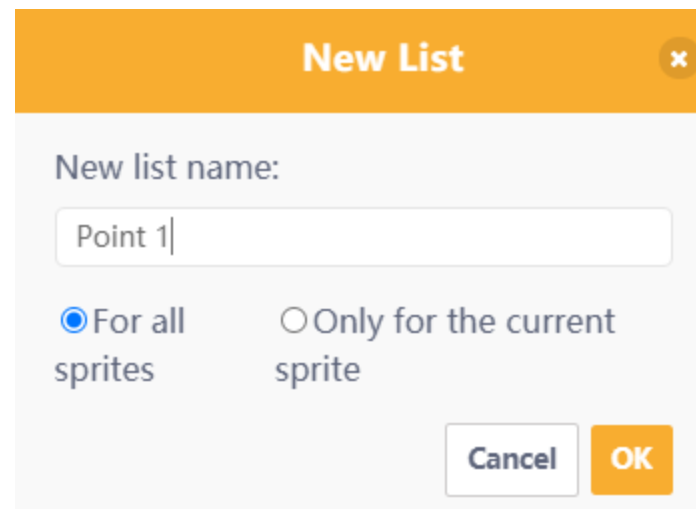


we slowly build a simple program for the drone to:

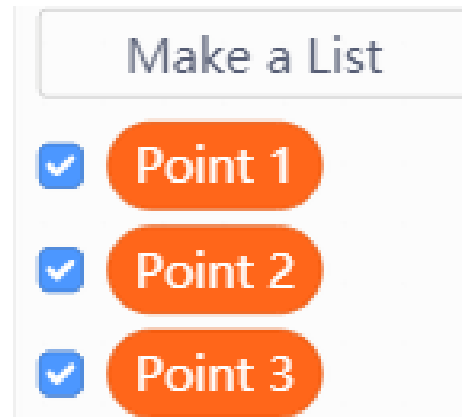
- Take off, then fly up 30cm
- Fly forward 100cm, then wait 15 seconds
- Land

Use Case 2 Mission: Creating Lists

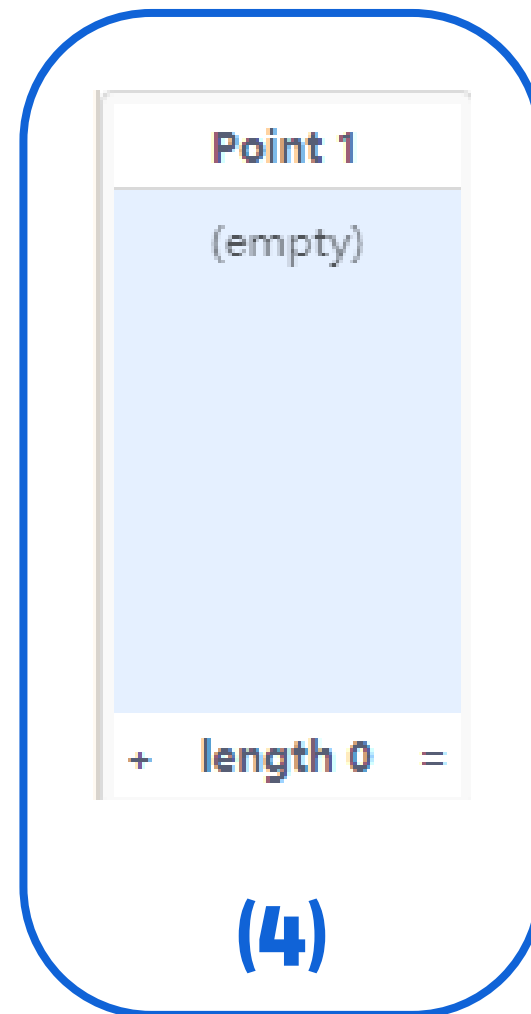
(2)



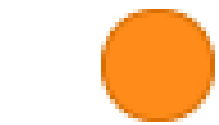
(3)



(4)



Creating Lists

Before using Lists to store data, we have to create them. In the  **Variables** section we

Make a List

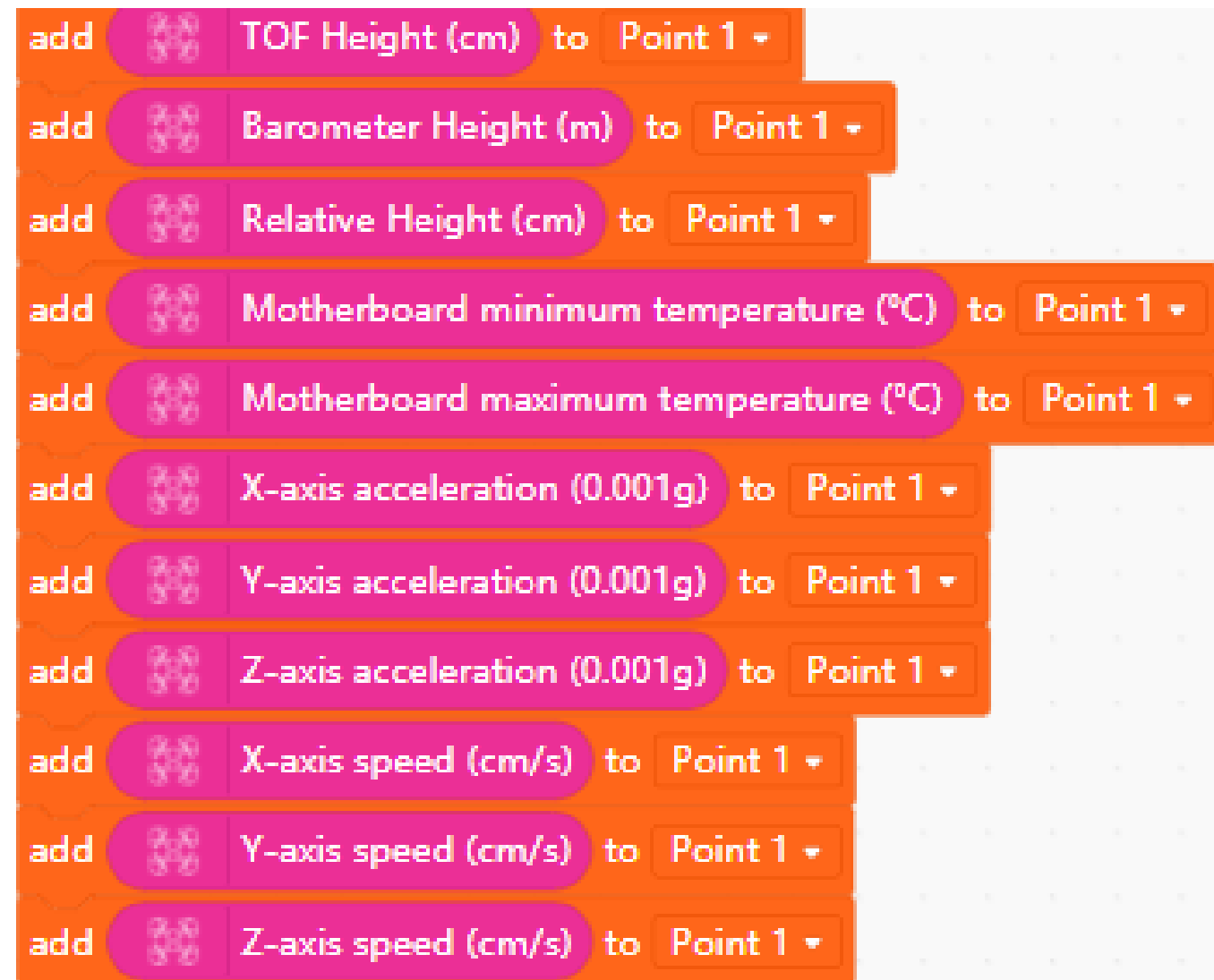
find the command  which causes the pop up (2).

We use this command to make three lists: Point 1, Point 2, Point 3, seen in (3).

These lists show up empty in the sprite

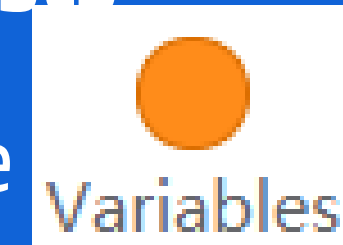
Use Case 2 Mission: Filling lists with Data

(5)



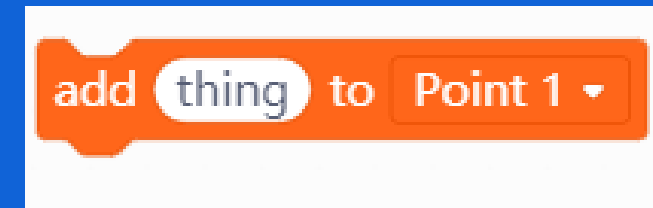
Data in Lists

After creating the lists, in the



section

we find the command



, which

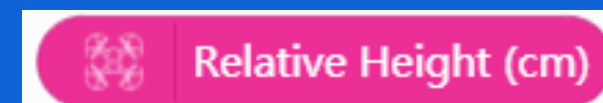
we can use to put different variables in the lists.

In the



section, we find variables such

as



, which we use in (5) to put

data gathered from the drone's internal

Use Case 2 Mission: Final Program

The image displays three columns of Scratch code blocks, representing a drone mission program. The first column (left) starts with a 'when clicked' event, followed by 'take off', 'fly up 30 cm', 'wait 15 seconds', 'fly forward 100 cm', and a series of 'add' blocks for Point 1: TOF Height (cm), Barometer Height (m), Relative Height (cm), Motherboard minimum temperature (°C), Motherboard maximum temperature (°C), X-axis acceleration (0.001g), Y-axis acceleration (0.001g), Z-axis acceleration (0.001g), X-axis speed (cm/s), Y-axis speed (cm/s), and Z-axis speed (cm/s). It ends with 'fly up 50 cm' and 'fly backward 100 cm'. The second column (middle) continues with 'add' blocks for Point 2: TOF Height (cm), Barometer Height (m), Relative Height (cm), Motherboard minimum temperature (°C), Motherboard maximum temperature (°C), X-axis acceleration (0.001g), Y-axis acceleration (0.001g), Z-axis acceleration (0.001g), X-axis speed (cm/s), Y-axis speed (cm/s), and Z-axis speed (cm/s). It then includes 'fly down 60 cm', 'flip forward(f)', and 'set speed to 50 cm/s'. The third column (right) continues with 'add' blocks for Point 3: TOF Height (cm), Barometer Height (m), Relative Height (cm), Motherboard minimum temperature (°C), Motherboard maximum temperature (°C), X-axis acceleration (0.001g), Y-axis acceleration (0.001g), Z-axis acceleration (0.001g), X-axis speed (cm/s), Y-axis speed (cm/s), and Z-axis speed (cm/s). It concludes with a 'land' block.

Combination

We repeat the process in (5) for lists Point 2 and Point 3. Then we combine those commands with our simple program in (1).

In between the measurements, we add different drone actions to ensure that there is variety in the collected data!

Use Case 2 Mission: Final Result

Point 1		Point 2		Point 3	
1	107	1	220	1	154
2	37.84	2	38.22	2	38.02
3	80	3	130	3	120
4	71	4	71	4	69
5	74	5	73	5	72
6	-32.00	6	-32.00	6	-34.00
+ length 11 =		+ length 11 =		+ length 11 =	

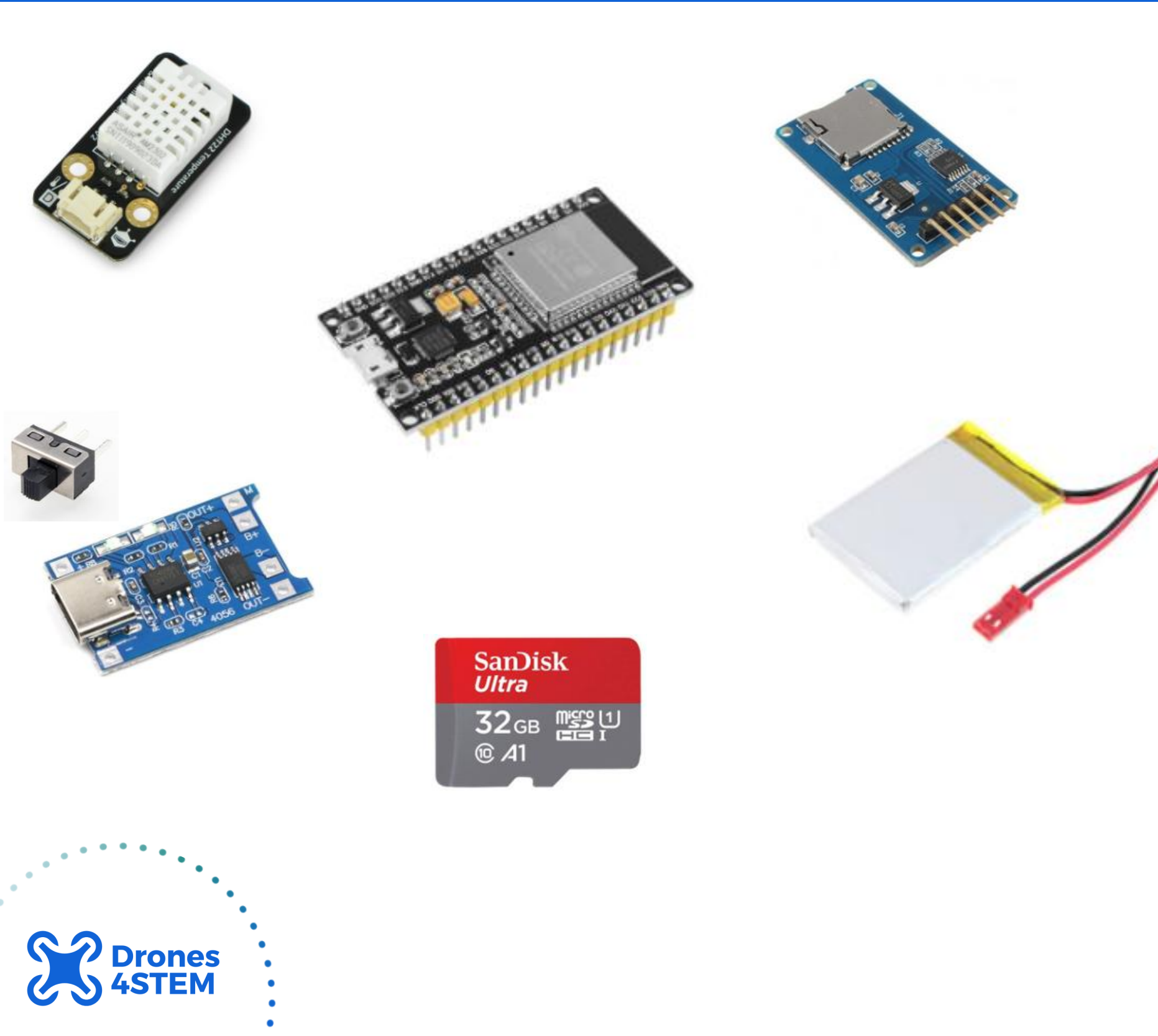
End of the Program

After running the program, we have 3 different lists with measurements coming from the drone's sensors.

We will use this data in the following stages of the mission!

The process can be replicated using external sensors instead of internal, to collect

Use Case 2 Mission: Bonus



Using External Sensors

In order to use external sensors, we will add a bonus section to the Use Case. This requires:

- A DHT11 Temperature and Humidity Sensor
- An ESP32 Development Board
- A Breakout Board for Micro SD Card Mini
- A Battery (Li-Po - 3.7v 800mAh)
- A battery charger (1A USB-C - TP4056)
- A switch (Mini SPDT - 1A/250VAC)

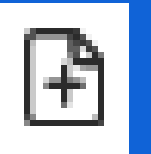
Use Case 2 Mission: Bonus



Programming

For using an external sensor, we will move to the Code section of Mind+ in order to write Python Code.

The file main.py of the use case is the one we will be using.



To create a new Python file, click the icon next to “File in the project”, and then write or paste your code.

The main.py file will program the data collection

Use Case 2 Mission: Bonus

```
import machine
import dht
import time
import os
from machine import SPI, Pin
import sdcard

# DHT22 Sensor Setup
DHT_PIN = 4
dht_sensor = dht.DHT22(machine.Pin(DHT_PIN))

# SD Card Configuration
CS_PIN = 5 # Confirm this matches your wiring
spi = SPI(1,
          baudrate=400000, # Reduced baudrate for stability
          sck=Pin(13),
          mosi=Pin(15),
          miso=Pin(14))

try:
    sd = sdcard.SDCard(spi, Pin(CS_PIN))
    os.mount(sd, '/sd')
    print("SD card mounted successfully")
    print("Root directory:", os.listdir('/'))
    print("SD card contents:", os.listdir('/sd'))
except Exception as e:
    print("SD card initialization failed:", e)
    machine.reset()

# File Configuration
FILE_PATH = '/sd/data.csv'

# Test file operations
try:
    with open(FILE_PATH, 'a') as f:
        f.write('Test entry\n')
        print("File write test successful")
except Exception as e:
    print("File operation failed:", e)

def log_data(temp, hum):
    try:
        timestamp = "{:02d}:{:02d}:{:02d}".format(*time.localtime())
        data = f"Temp {temp:.1f},Hum {hum:.1f}\n"
        with open(FILE_PATH, 'a') as f:
            f.write(data)
        print(f"Data logged: {data.strip()}")
    except Exception as e:
        print("Logging failed:", e)

while True:
    try:
        dht_sensor.measure()
        temp = dht_sensor.temperature()
        hum = dht_sensor.humidity()
        log_data(temp, hum)
    except OSError as e:
        print("Sensor read error:", e)

    time.sleep(10)
```

The code

The python code in main.py takes care of the following things:

- Setting up the DHT11 sensor
- Configuring the SD Card
- Creating a .csv file for the data
- Logging the sensor

Use Case 2 Mission: Bonus

File in the project



 main.py

 sdcard.py

SD Library

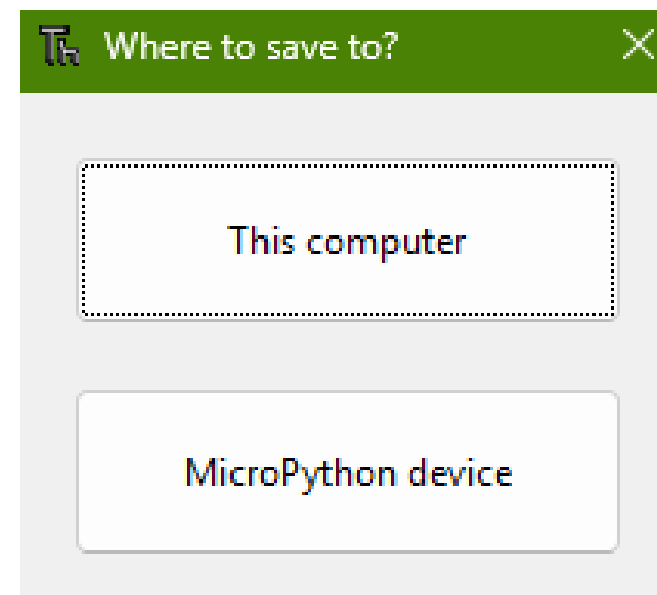
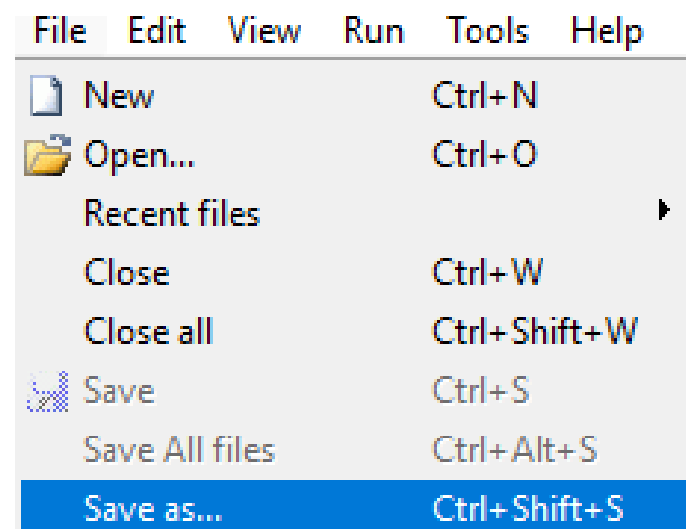
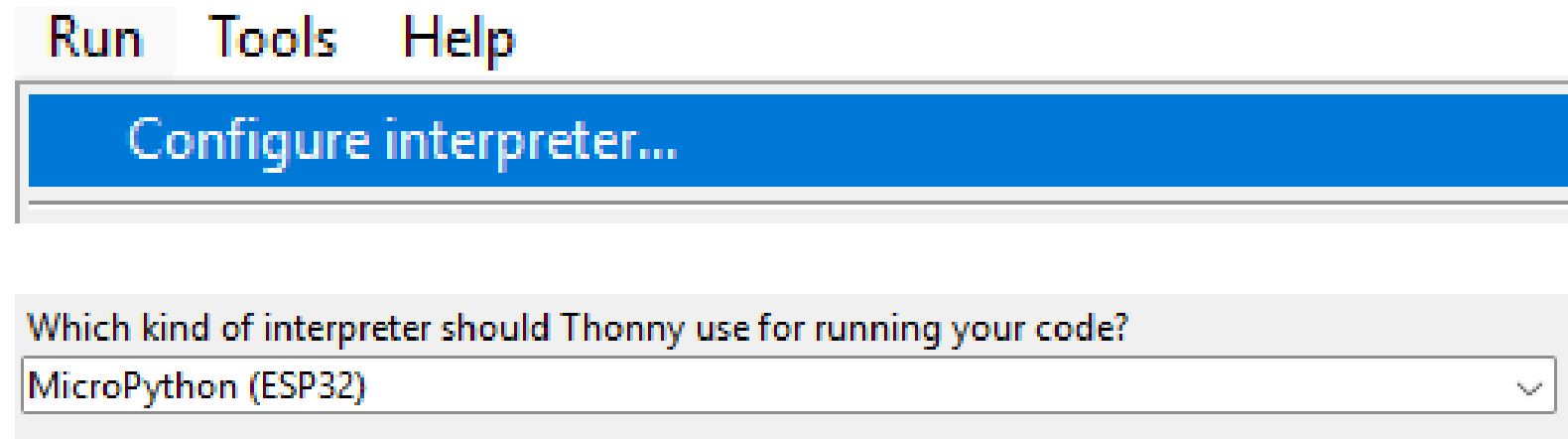
For the code in main.py to work, an additional library is needed.

The sdcard.py library can be found in this link:

<https://github.com/micropython/micropython-lib/blob/master/micropython/drivers/storage/sdcard/sdcard.py>

Copy the code and paste it in a new python file named sdcard.py

Use Case 2 Mission: Bonus



Loading code into the development board

One way to save our main.py and sdcard.py files in the development board is through the IDE Thonny, through the following steps:

- Step 1: Open the main.py file with Thonny.
- In Thonny, click Run, then Configure Interpreter, and choose MicroPython ESP32. Press OK.
- Then, click File, Save As, and choose the MicroPython option.
- Repeat for the sdcard.py file.

Questions

- How does block based programming make programming more accessible?
- Why does the drone have internal sensors?
- How can block-based drone programming be used in environmental monitoring?

What are your questions?



Additional Resources

Further Reading

<https://mindplus.dfrobot.com/RMTT>

Videos

[Introducing Mind+ : A Sctrach 3.0 Programming Software](#)



Thank you for your
attention !



Co-funded by
the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.



