

CONSTRUCTION OF METEOROLOGICAL STATIONS WITH THE POSSIBILITY OF MEASURING AIR QUALIT

Židlochovice Elementary school, district Brno – rural, contribution organization

Tyršova 611, Židlochovice

Graphical user interface, website

Description automatically generated

Creativity, 3D printing, Electronics

Contents:

[Information about the project 3](#_Toc116852677)

[Project description: 3](#_Toc116852678)

[What age group is the project intended for? 3](#_Toc116852679)

[What skills should students master before starting the project? 3](#_Toc116852680)

[What skills will the students learn as part of the project: 3](#_Toc116852681)

[Material equipment needed to implement the project 4](#_Toc116852682)

[Device: 4](#_Toc116852683)

[Parts and Consumables: 4](#_Toc116852684)

[Alternative and expansion components: 5](#_Toc116852685)

[Financial complexity of the project 5](#_Toc116852686)

[**Basic components and material:** 6](#_Toc116852687)

[Additional components and material (expansion of the project, facilitation of work, alternative components) - not absolutely necessary for implementation 6](#_Toc116852688)

[Problems that the students will have to solve as part of the project 7](#_Toc116852689)

[Proper connection of components. 7](#_Toc116852690)

[Assembling complete wiring, programming and sending data. 7](#_Toc116852691)

[Optimization of consumption 8](#_Toc116852692)

[Weather station packaging design in 3D software, 3D printing and assembly 8](#_Toc116852693)

[Testing the entire weather station assembly 10](#_Toc116852694)

[Programming in the MakeCode environment 10](#_Toc116852695)

[Connection of individual components and their programming 13](#_Toc116852696)

[Wi-Fi module settings 13](#_Toc116852697)

[Connection of the shield (module for the internet of things) 14](#_Toc116852698)

[Connecting and programming the BME280 sensor. 15](#_Toc116852699)

[Sending data to ThingSpeak 17](#_Toc116852700)

[PM2.5 flying dust sensor 18](#_Toc116852701)

[Optimization of consumption – use of a relay to switch off the sensor. 19](#_Toc116852702)

[Sending data and its visualisation 21](#_Toc116852703)

[Create a ThingSpeak account 21](#_Toc116852704)

[Establishing a channel to receive data 23](#_Toc116852705)

[Complete assembled product 25](#_Toc116852706)

[List of links 27](#_Toc116852707)

[Programming in the MakeCode environment: 27](#_Toc116852708)

[Documentation for individual components (in English): 27](#_Toc116852709)

[E-shops for purchasing parts and components: 27](#_Toc116852710)



# Information about the project

## Project description:

As part of the project, the students will build a weather station, which should be able to record pressure, temperature, humidity and the amount of airborne dust PM2.5.

## What age group is the project intended for?

The project is quite complex and is suitable for 8th and 9th grade students in elementary school. It can be the final part of informatics subject and technical education at elementary school, for example as a final student project.

## What skills should students master before starting the project?

Basics of 3D modeling and 3D printing

Basics of programming (knowledge of no particular programming language is required)

Ability to work in a team

## What skills will the students learn as part of the project:

Creation of technical drawing, planning, prototyping

3D modelling in CAD software

Basics of working with electronic components

Programming in a block environment

Sending data and their further processing



# Material equipment needed to implement the project

## Device:

3D Printer

Filament for printing weather station packaging

Soldering station

Drill and model sander

Melt gun

## Parts and Consumables:

BBC Micro:bit V1

<https://www.hwkitchen.cz/bbc-microbit-mikropocitac-pro-vyuku-programovani/>

Shield for Microbit with Wifi and RTC

<https://www.hwkitchen.cz/iot-bit-pro-microbit-modul-pro-internet-veci-iot/>

Sensor for measuring temperature, pressure and humidity – BME280

<https://www.hwkitchen.cz/octopus-bme280-snimac-tlaku-teploty-vlhkosti/>

PM2.5 airborne dust sensor

<https://www.hwkitchen.cz/octopus-senzor-kvality-vzduchu-pm2-5/>

Relay module for microbit – 3V

<https://www.hwkitchen.cz/octopus-3v-rele-modul-pro-micro-bit/>

Power bank for power supply

Other necessary accessories:

Du Pont wires, USB cable

## Alternative and expansion components:

Battery holder for microbit:

<https://www.hwkitchen.cz/drzak-baterii-aaa-s-vypinacem-pro-micro-bit/>

Water level sensor:

<https://www.hwkitchen.cz/octopus-snimac-vodni-hladiny/>

Temperature and humidity sensor DHT11

https://www.hwkitchen.cz/snimac-teploty-a-vlhkosti-dht11/

# Financial complexity of the project

In the event that you have the necessary devices and tools available, the cost of the project is only the cost of purchasing the components, wiring and filament for the 3D printer. It should be noted here that it is possible to disassemble and reuse all components from the weather stations after the end of the project. There is also absolutely minimal soldering in the project, so it is possible to fully reuse the components for another project, or for repeating the same project with another group of students, for example in the following school year. In our case, the costs were higher, mainly because we wanted to provide the students with a larger number of different components, so that they could also test their suitability for use in the project, or come up with more different solutions. In the following budget I only list the components that are used in the final version of the product.

.



**Basic components and material:**

|  |  |
| --- | --- |
| BBC Micro:bit V1 | 550 CZK |
| Shield for Microbit with Wifi and RTC | 449 CZK |
| Sensor for measuring temperature, pressure and humidity – BME280 | 478 CZK |
| PM2.5 airborne dust sensor | 794 CZK |
| Relay module for microbit – 3V | 103 CZK |
| Power bank for power supply | 629 CZK |
| Du Pont wires, USB cable | 89 CZK |
| Filament (calculated with approx. 3 prototypes before the final product) | 599 CZK |
| In total | 3,691 CZK |

**Additional components and material (expansion of the project, facilitation of work, alternative components) - not absolutely necessary for implementation**

|  |  |
| --- | --- |
| Battery holder for microbit | 42 CZK |
| Water level sensor | 77 CZK |
| Temperature and humidity sensor DHT11 | 39 CZK |
| OLED display | 151 CZK |
| Alternative airborne dust sensor (ME Air Quality Sensor) | 1 139 CZK |



# Problems that the students will have to solve as part of the project

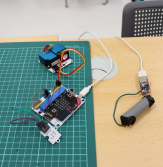
## Proper connection of components.

This is a relatively simple problem, since the correct connection of the components is clear and well documented in the manuals for the individual components and sensors.

Recommendation for teachers: Let the students first connect the individual sensors and components separately. Without sending data to ThingSpeak, only displaying data on the microbit itself. If they have never worked with similar sensors, but have experience with the basics of programming, you can expect the activities to take about 4 hours. The microbit's built-in display can be used to display data, but you can make your work much easier by using a small OLED display. You will not use the display in the final weather station (it would unnecessarily increase the power consumption of the assembly), but it is a great helper during teaching and testing.

**Assembling complete wiring, programming and sending data.**

In programming, it's just about using individual pieces of code together in one project.



Teacher's recommendation: Get the students to make at least some documentation for the project – for example, a spreadsheet of which pin of the microbit is connected to which device and a list of the variables used in the project. It is also a good idea to push students into sensible variable naming to keep the code clear. Leave the programming itself to the students, but it is again good to connect and test the individual components gradually – this will make it much easier to find errors. In these activities, students learn to work on a larger project, cooperation and consistency. The time required here is very dependent on the experience of the students.



## Optimization of consumption

Optimising electricity consumption is one of the biggest challenges for a weather station located outside the reach of mains power. Its solution and subsequent testing is very time-consuming and requires long-term planning and additional knowledge in the field of physics (voltage, current measurement, power consumption calculation) from the students. However, it is an excellent connection with the subject matter that pupils in this age group are discussing in physics.

Recommendation for teachers: This part of the project can be completely left out (replace the battery source with a mains adapter). This will save you a lot of time and resources (in our case, it took more than six weeks to test different power supply options). However, from my point of view, this is an essential part of the project, which will force students to think about a complex problem that has many possible solutions and requires knowledge from other related fields. Pupils were forced to try different power sources, measure their real capacities and disconnect different components of the assembly in time using relays in an attempt to save energy. This is exactly the type of problem that develops students the most, and by solving it they move forward in the field.

## Weather station packaging design in 3D software, 3D printing and assembly

This is by far the most time-consuming activity within the project. At first there should be a discussion with the pupils about what conditions must the weather station cover meet in order for the whole project to be functional. Subsequently, it is necessary to lay out the entire assembly in the space, measure and draw the sizes and positions of individual components and holes for cabling. Subsequently, convert the entire model to CAD and print it on a 3D printer and place the components in it. 

Recommendation for teachers: This is really by far the most complicated part of the project and cannot be rushed. Students learn very important skills through this part of the project: prototyping, problem solving, the ability to evaluate the solution, and propose its improvement. Before the first draft, it is good to brainstorm, write ideas on the board, then go through them with the pupils, but do not impose your solutions on them.

Expect the first model that "falls out" of the 3D printer to be completely unusable, but that's part of the learning process. The students will evaluate it and each next iteration will be better and better. Some problems in the design of the chassis will not be discovered by the students until they test the product in operation (for example, incorrect measurements of sensors located close to other components due to the accumulation of waste heat.) However, it is good to let them discover and eliminate these shortcomings on their own - mistakes are the best way to learn.A picture containing text, indoor

Description automatically generated

The main problems that the students will have to solve:

* **Access and extraction of air from the sensor for measuring the amount of airborne dust** PM2.P. The sensor works on the principle of beam scattering on dust particles in the sensor chamber. It is therefore necessary to ensure smooth passage of air through the sensor – the assembly must ensure trouble-free air intake and exhaust to the sensor.
* **Protection of the temperature sensor from waste heat from the rest of the assembly.** If the sensor is placed together with other components in the same space, the temperature measurement results are significantly distorted. Waste heat from other components causes a measurement error of up to 4°C. It is therefore necessary to design a separate space for this sensor, however cables must be connected to it.
* **Water resistance of the entire assembly**
* **Protection against overheating of the assembly in direct sunlight**

## Testing the entire weather station assembly

Although the students will test the entire connection "on the table", it is necessary to test the entire connection of the weather station for a long enough time, even in the long term. It is also a good idea to compare the measured values with one of the professional measuring stations. We also verified the functioning of the sensors directly at the source of pollution – in our case, at the exhaust of an older car with a diesel engine.



Air quality data from professional measuring stations can be found here:

<https://www.chmi.cz/aktualni-situace/stav-ovzdusi/prehled-stavu-ovzdusi>

# Programming in the MakeCode environment

The visual programming environment for programming MicroBit microcontrollers can be found at <https://makecode.microbit.org/>.

It is a simple block editor in the style of the well-known educational programming language Scratch, which most students have already encountered at school. This fact greatly facilitates and speeds up work in the environment. The entire environment can be switched to Czech, however, extensions (libraries) for individual additional components are only available in English.



Graphical user interface, application

Description automatically generatedThe description of working with the programming environment is not the purpose of this document, it is quite intuitive and it is possible to find a large number of educational materials for it. Considering the age group for which this project is intended, it can also be assumed that the pupils have already met it.

The only thing I would like to point out here is uploading the program to the microbit. The procedure is simple, just press the Download button on the finished program, the program will be saved and then copy it to the connected Microbit. At first, students are confused by the fact that the copied code cannot be seen on the microbit (which appears as a USB drive on the computer). The newer version of the microbit V2 allows pairing the microbit with a computer, where recording is then faster and easier. Unfortunately, according to our experience, this function is not yet reliable and we have not used it in practice.

To program extension components, it is also necessary to add extensions (libraries) for individual components that we will connect to the microbit to the programming environment. Specifically, apart from the basic commands, we will need libraries for the BME280 sensor, a sensor for measuring the amount of dust particles in the air, a relay for switching the sensor and a Wi-Fi module.

Graphical user interface, text, application

Description automatically generatedTo add an extension, click Advanced> + extension, this will take you to the screen where individual extensions are searched for.

Graphical user interface, website

Description automatically generated

Unfortunately, there is an unpleasant problem associated with importing extensions into the MakeCode environment – there is unfortunately quite a large number of extensions for the components that we are using, they have very similar, sometimes even the same names, and it is very difficult to distinguish them from each other. At the same time, different libraries work differently, some dont work with certain components at all, and some are not compatible with each other. If you encounter a problem e.g. "I do everything exactly according to the instructions, but the sensor does not work", the problem is very often in the wrong library.

Therefore, it is better to add libraries using the links you send to students, rather than using the search box in the extension section. There, you will be sure that all students work with the same library as you, a teacher, have verified and tested. Just copy the link to the library into the search field in the extension section.

For simplicity, here are links to the extensions we used in the project. These extensions are current for the sensors and modules used as of November 2021. Of course, libraries can change over time and you will need different extensions if you use alternative components.

Links to used libraries:

https://makecode.microbit.org/12007-47281-99192-97327

https://makecode.microbit.org/06200-34912-09077-56827

https://github.com/tinkertanker/pxt-iot-environment-kit



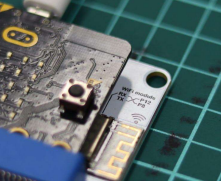
# Connection of individual components and their programming

Students should come up with the individual parts of the code themselves, the teacher's role within the project should be more of assistant (recommend suitable tools, software etc.) and help in connecting individual parts of the project. Therefore, students should definitely not see the following solution of individual parts of the code before the actual programming – they are more for teachers, or for guidance in case of an unsolvable "jam".

## Wi-Fi module settings

For wireless transmission to work, you must first set up the ESP8266 chip, which is mounted on the shield used in the project. You can find commands to use it in many libraries. And in the Makecode menu they appear under the name of the sensor (ESP8266). Graphical user interface, text, application

Description automatically generated

In the left part of the code, the ESP8266 chip is set. The parameters RX (received – transmitted) and TX (sent – transmitted) set the pins to which the ESP8266 is connected - in our case, they are given by the shield and cannot be physically changed (the chip is powered on the board). You can find the values ​​on the board itself (see photo). If you have an ESP8266 chip separately, these values ​​depend on where you connect it.

Next, the SSID of the network to which you will connect the microbit and the password are set. In practice, it is a good idea to supplement the program with some form of confirmation that the microbit has been successfully connected to the wireless network – this is provided by the part of the code on the right. If Wi-Fi is connected, a check will appear on the display.



## Connection of the shield (module for the internet of things)

The Internet of things module extends the capabilities of the microbit itself. These components, to which a microcontroller is simply connected, are colloquially called shields, there are a large number of different shields for microbits and most of them are usable for our project.

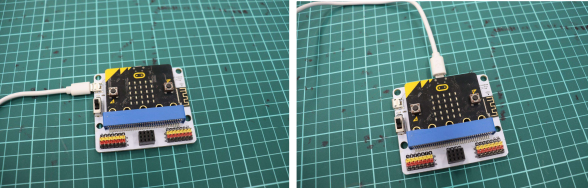
Graphical user interface

Description automatically generated

Why do we use shield?

* It allows easy connection of components, without the need to use clamps and soldering, everything is easily connected using dupont cables. Projects can be assembled and disassembled and changed very quickly. Damage to components is also prevented.
* The Shield contains other components that the microbit itself does not have. In our case, it is primarily an integrated wi-fi chip, which we will need to send data from the weather station to the internet.

The only thing we have to be careful about when working with the shield is connecting the USB cable. For the wi-fi chip to work properly, it is necessary to power the shield with the USB connector, which is directly on the shield (not the microbit connector). However, it is not possible to program the microbit with this connector. Unfortunately, it is therefore necessary to constantly switch the cable from one connector to another, or to use an external power source.



The photo on the left shows the correct connection for the wifi chip on the board to work. On the right photo is the connection for transferring the program from microbit.

Diagram

Description automatically generatedFor further connection of the components to the shield, we will also need a diagram of the output of the individual pins to the connectors in the lower part of the board:

## Connecting and programming the BME280 sensor.

The BME280 sensor connects via the I2C bus, so you will need four wires to connect it. We connect the power supply (VCC and GND) to any corresponding pins (V and G) on the shield. We connect the clock signal of the I2C bus (SCL) to pin 19 marked on the CL shield, and the data channel (SDA) to pin 20 marked on the DA shield. On the shield, these pins are brought out in two places – in the middle as a special connector for the I2C bus and on the far right as separate pins. I recommend leaving the middle position free for

connecting the OLED display during testing (it fits exactly here, without the need to use wires) and rather connect the BME280 sensor to the right side of the microbit.

Diagram, schematic

Description automatically generated

|  |  |  |
| --- | --- | --- |
|  | BM280 Na shieldu | BM280 Na shieldu |
| Red | VCC | V |
| Black | GND | G |
| Blue | SDA | Pin 19, DA |
| Green | SCL | Pin 20, CL |

In the following program, we display the values ​​read from the sensor on the OLED display. The program is simple and intuitive. Of course, an OLED display is not necessary, and data can also be written to the microbit display, or via a serial line. However, the use of the display makes the list clearer. The display must be initialised at the startup.



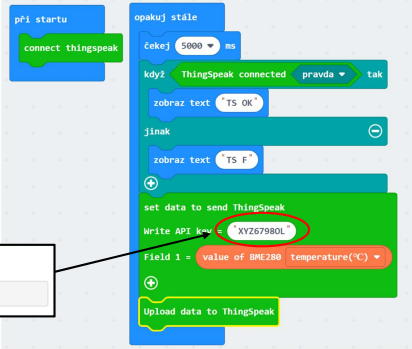
## Sending data to ThingSpeak

Graphical user interface, text, application, chat or text message

Description automatically generated

First, the connection to ThingSpeak needs to be initialised. As with the wifi connection, it is a good idea to program feedback into the microbit, whether the connection to thingspeak was successful (in our code, it is a statement of TS OK/TS F on the microbit display). For the recording itself, you need to copy the API Write key from your channel (see the section on ThingSpeak settings) and write the quantities you want to send in the corresponding fields.

The data is sent only after the "upload data to ThingSpeak" command. In the same way, we enter the sending of other values ​​to other fields. Here we found it useful to send some specific value that is not measured – for example, a constant. We can then easily verify on the ThingSpeak side whether there is a possible problem in sending data or in its collection.





Teacher Note: We ran into a problem when sending data to ThingSpeak that we were unable to resolve. In the case when we worked simultaneously with a large number of microbits (around ten or more devices) sending data over one wifi network, ThingSpeak refused to receive data from some devices. The problem may be on the ThingSpeak side (perhaps protection against abuse, using a large number of unpaid accounts at the same time), but also on the side of our network. It was not a complication when solving the project itself, but it is an inconvenience when teaching the whole class.

## Diagram Description automatically generated with medium confidencePM2.5 flying dust sensor

The sensor is connected via the GVS interface, so the connection is very easy – using a GVS cable (a trio of dupont cables). The sensor has its own electronics and can directly read the values of airborne dust PM2.5.

Graphical user interface, text, application, chat or text message

Description automatically generated

Connecting and programming the sensor is therefore very simple, but for proper operation it is necessary to understand how the sensor works. It is actually a small closed chamber through which a constant flow of air is maintained using a fan. The inside of the chamber is illuminated

by a beam that is refracted on dust particles and this refraction is recorded by a sensor. Two facts follow from this:

- It is necessary to ensure an unblocked supply of air and its removal from the sensor, which imposes requirements on the construction of the weather station and the location of the sensor in it.

The sensor has a significant consumption of electricity, if it is turned on all the time it significantly reduces the battery life of the whole assembly. On the other hand, it is not necessary to measure the dust values ​​very often, similar to the temperature, these values ​​do not change too dramatically and, for example, one measurement every 15 minutes is quite sufficient. It is therefore possible to disconnect the sensor and significantly increase the durability of the entire assembly – see consumption optimization part

Note for teachers: The sensor needs to be in operation for a certain time to start giving correct values. The first few values ​​are always ridiculously low, after about a minute of operation the values ​​stabilise and the measurement is relatively accurate. This fact is especially important when trying to optimise consumption – it is not possible to connect the sensor and immediately read values ​​from it. It needs to be in operation for a certain amount of time to start working properly.

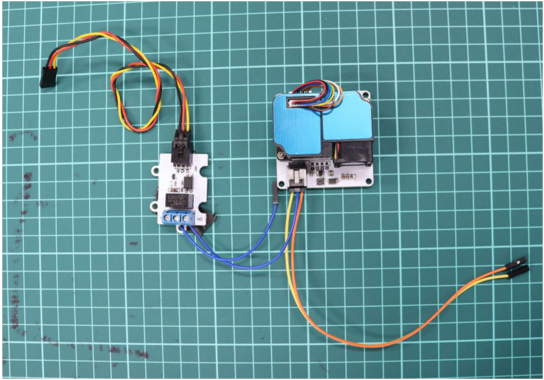
## Diagram Description automatically generatedOptimization of consumption – use of a relay to switch off the sensor.

The biggest obstacle to the operation of a weather station outside the range of mains power is the limited battery capacity. As part of the gradual testing, we went from a small battery pack with two AAA batteries to the power bank with a capacity of 20,000 mAh. Unfortunately, even this source is only capable of powering the entire set of the weather station for a maximum of three days.

Given that the microbit has a suitable technical solution for reducing consumption (some form of sleep mode), the students suggested disconnecting the sensor with the highest consumption using a relay.





Connecting and controlling the relay is again quite simple. The relay is switched by writing a logical one to the pin of the microbit, into which we connect the wire leading from the pin marked S (signal). Powering the sensor which we want to control this way, we lead through the terminals on the left side of the relay.

For the reasons stated above, it is necessary for the air quality sensor to be triggered by the relay for at least 30 seconds

Graphical user interface

Description automatically generatedbefore the actual measurement. Then a measurement is taken, the data is sent to ThingSpeak and the sensor is disconnected from the power supply using the relay.

# Sending data and its visualisation

For receiving and collecting data, we chose the ThingSpeak cloud tool designed for work in the internet of things environment. The reason for use within school projects is mainly:

* reliability
* the basic account is free for student projects, although it has limitations regarding the number of recorded values ​​(maximum of 3 million records, 4 channels and uploading data every 15 seconds), but for the purposes of this project it is completely sufficient.
* In case of more intensive use, it offers an affordable academic licence (more about free licence restrictions here: https://thingspeak.com/pages/license\_faq)
* a simple and intuitive environment that students can easily master
* it is a frequently used environment – there are plenty of tutorials and documentation
* in the make code environment, libraries are available for uploading data from microbit to ThingSpeak
* the possibility of displaying data on mobile devices

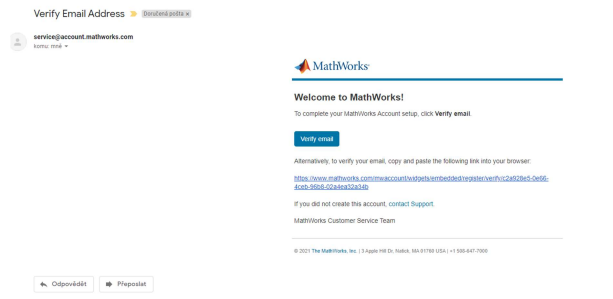


## Create a ThingSpeak account

At the address https://thingspeak.com/ we first select Sign In at the top right and on the next page Create Account.

Graphical user interface, text, application, email

Description automatically generatedWe enter the registration email, country, name and surname. The first and last name must Not contain diacritics.

Graphical user interface, diagram, application

Description automatically generatedAfter confirmation of registration, students will receive a confirmation email in their mailbox, where they will verify their account using the Verify email button.

In the ThingSpeak environment, they just select that it is a student account

## Graphical user interface, application Description automatically generatedEstablishing a channel to receive data

In the ThingSpeak environment, we first create a new channel on the Channels tab. A channel is a place to store data from one weather station. Each channel can contain up to eight fields for eight different recorded variables. In our case, we will need one field for each recorded quantity – pressure, temperature, humidity and amount of dust particles PM2.5. It is a good idea to name the field appropriately. The only other information we need to enter is the channel name. Table

Description automatically generated with medium confidence



On the Private view tab we can see the recorded data itself and its rendering in graphs. It is also possible to set the form in which the data will be displayed, or the possibility to analyse and visualise the data using other tools.

In the channel settings tab, we can change the settings made when creating the channel, and there is also an important button at the bottom of the page – clear channel. Pressing the button deletes the data recorded so far from all fields in the channel. We will often use this option when testing the device and recording data.

In the sharing tab, we can find data publication settings. To publish data from the weather station on the website, you must select the "Share channel view with everyone" option. Furthermore, there is an option to make data available only to selected users – however, they must have an account on the ThingSpeak platform.

Graphical user interface, text, application, chat or text message

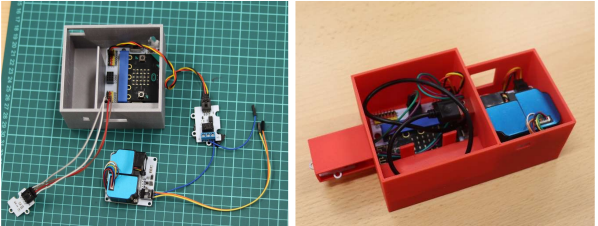
Description automatically generated

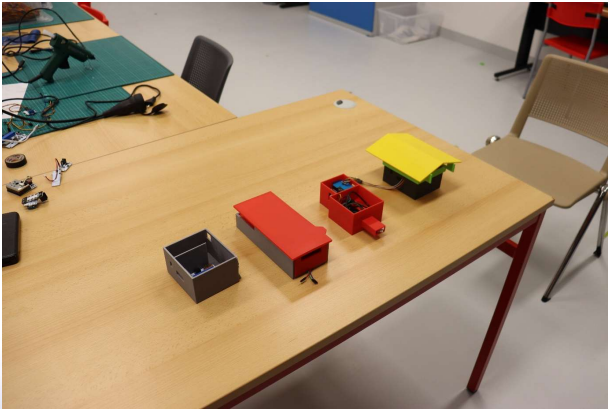
The API keys tab primarily contains the data write key that needs to be entered into Microbit to send data to the correct channel in ThingSpeak. The Write API key represents a combination of name and password for writing to the channel. If at any time during work on the project we change the write API key using the button on this tab, it is necessary to modify this item in the microbit program as well.Graphical user interface, text, application

Description automatically generated

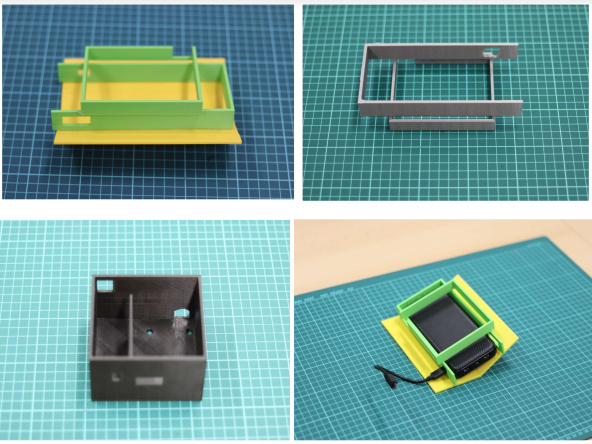
The last Data Import / Export tab allows you to export measured values ​​in CSV format, for example for further use in a spreadsheet, creating graphs, etc.

# Complete assembled product

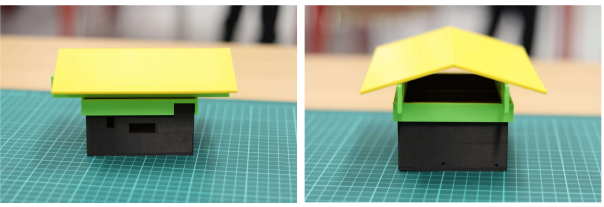
Complete connection of individual components:

Different technical solutions to the same task, the pupils arrived at different technical solutions:

Individual 3D printed components of the weather station package:





Complete weather station chassis:

Record of the operation of two functional prototypes located on the school premises:

Graphical user interface, application

Description automatically generated

# List of links

## Programming in the MakeCode environment:

MakeCode programming environment for microbit:

<https://makecode.microbit.org/>

Programming textbook with microbit - age for 8th and 9th grade (target group of this project):

<https://imysleni.cz/images/vzdelavaci_materialy/microbit_makecode/Microbit_s_Makecode_metodika_pro_ucitele.pdf>

A nice site with exercises from complete beginners to intermediate:

<https://www.microbiti.cz/>

## Documentation for individual components (in English):

Microbit V1.5:

<https://tech.microbit.org/hardware/1-5-revision/>

Shield for microbit:

<https://www.elecfreaks.com/learn-en/microbitKit/iot_kit/iot_bit.html>

BME280 sensor:

<https://www.elecfreaks.com/learn-en/microbitOctopus/sensor/octopus_ef04087.html>

Air quality sensor - PM2.5:

<https://www.elecfreaks.com/learn-en/microbitOctopus/sensor/octopus_ef04090.html>

DHT11 sensor:

<https://www.elecfreaks.com/learn-en/microbitOctopus/sensor/octopus_ef04019.html>

Water level measurement sensor:

<https://www.elecfreaks.com/learn-en/microbitOctopus/sensor/octopus_ef04019.html>

OLED display:

<https://www.elecfreaks.com/learn-en/microbitOctopus/output/octopus_ef03155.html>

Relay 3V:

<https://www.elecfreaks.com/learn-en/microbitOctopus/output/octopus_ef04086.html?highlight=relay>

## E-shops for purchasing parts and components:

<https://www.hwkitchen.cz/>

<https://rpishop.cz/>

<https://www.gme.cz/e>