IoT4Schools

"Bringing the Internet of Things in school education as a tool to address 21st century challenges"

Using IoT Devices for Water Conservation and Preservation of School Gardens

Teachers' guidelines

Kofteros, A & Tsaliki, F

Heron, Digital Education & Mathisis

Co-funded by the European Union

Table of contents

1 Introduction to the project

1.1 Scenario and Scope of the project

The scope of the project is for the students to research the needs of sustaining a school garden, especially during the weeks and months of school holidays. Additionally, students can maximise the efficiency of watering plants using an automated IoT based system, that continuously measures the humidity in the soil and administering water when and if necessary, conserving valuable resources that are scarce, especially in southern Europe (Greece, Cyprus, Spain etc.). The automated system, based on Micro:bit, can also be solar powered to further reduce the cost of operation.

Through this scenario, a model automated gardening system can be created, continuously measuring the humidity in the soil, the temperature of the environment and (as an optional) the quantity of the water in the water reserve tank. The programming of the Micro:bit will allow the device to make decisions to administer water, including the quantity.

1.2 Learning objectives

Students are expected to:

- Research the watering needs of specific plants included in school gardens
- Identify the problems of irrigation in school gardens during the school holidays
- Identify the minimum required components to build a working prototype of an automated watering system
- Assemble and program an irrigation system based on Micro:bit

1.3 Learning pathway – Stages of implementation

The project builds on the premise of the pedagogical value of school gardens and allows students to identify the problem of irrigation during the school breaks, and especially the long summer break. Students will be required to suggest an automated system that delivers water in the most efficient way, conserving it when the soil is already humid, based on weather conditions (i.e. if rainfall occurred). Then, students are required to assemble, program and test their system to check it under real (lab) conditions.

Classroom arrangement: Students will work in groups of 3 or 4.

Problem identification: Students discuss the benefits of having a school garden (assumptions: schools have already covered this in Environmental Education) and identify the problem of watering the plants during the long weeks and months of school holidays. Students are enabled to suggest ways of keeping the plants irrigated, with the minimum possible cost and resources.

Discussion and assignment of the activity: Students work in teams to identify both the benefits (skill-building, creativity, sustainability) of creating school gardens. They are encouraged to research websites of schools that have their own school gardens, as well as articles online about their maintenance. Depending on the time allocated by the teacher(s) to the project, video conferences with other schools to share experiences might be encouraged.

After researching school gardens, students will focus on the problem of maintenance (mainly irrigation) during the school holidays. It is expected that students will research automated solutions that can be reproduced at school, with minimum cost. Since the scope of the project is to understand the complexities of such a project, and create a prototype (not an actual irrigation system), students will focus on creating a working system using IoT devices.

Planning: Students focus on the necessary equipment to create a working prototype of an automated system for watering a school garden. They will have to make suggestions on what type of devices, including sensors, are required, and the minimum number and type of components, to sustain a low entry-level of adoption. Students are encouraged to search online and suggest the component models, as well as the cost of each system. Students can suggest various types of each component, discuss them in their groups, and then, through a class discussion, decide on the final parts of their system. To help teachers with this process, a suggested list of components is included in the Appendix, and are the same components used in this guide.

Creation: The teacher uses a set of components that have been selected either by the students or through the school. It is encouraged that the students are part of the selection process, however, schools can still use components of their own selection to lower costs.

Students assemble their components, based on the steps given by the teacher, and each team is encouraged to work on its own, on a prototype that includes breadboards to avoid soldering, thus encouraging exploration, ease of switching components and wiring, and reducing the time required to reach the first working model.

Testing - optimization: Each team creates a working prototype model of an automated irrigation system and tests them in small-size pots that contain soil. Their code must be easily readable (block-based) and students must be able to explain how they created the system and how their code works. Each team must be given enough time to make changes to the hardware and code of their project and make an initial presentation to their teacher.

Presentation - sharing: After the teachers give their approval of the first operational prototype of each team, students are encouraged, one by one, to present their project and explain how their code works, how the components were connected, how it works and any suggestions for improvements they have. After each team presents their project, the other teams are encouraged to offer constructive feedback and suggestions. After the presentations, students are encouraged to implement the suggestions of the other teams, to improve their project.

1.4 Learning prerequisites

All students are expected to be familiar with the basic functions and sensors embedded on the Micro:bit device, including working knowledge of the block-based coding environment of MakeCode.

1.5 Hardware and software

Hardware:

- The BBC Micro:bit
- External power supply (4xAA batteries) and alternatively 5V solar panels
- RUNCCI-YUN Mini Water Pump Submersible Pump Micro Motor Pump DC 3V 5V

- AZDelivery Soil Moisture Sensor Hygrometer Module V1.2
- Zalati IOBIT Adapter for BBC Micro Expansion
- ESP8266 ESP-01S WiFi Module

Software:

- Microsoft MakeCode (block coding, alternatively Python)

1.6 Time plan

The project is estimated to have an average completion time of 6 to 8 periods (each period in Cyprus has a duration of 45'). Time is allocated in the following activities:

- 2 periods for research of school gardens to identify the problem and suggest solutions
- 1 period for selecting the equipment to be used (minimum number of items)
- 3 periods for assembling and programming the basic prototype setup with the equipment
- 1 period to present the project to the class (additional step)
- 1 period to make improvements, based on peer feedback (additional step)

2 Implementation of the project

2.1 Create a prototype system for garden irrigation

Students create a working prototype model of their garden irrigation system, that is completely functional based on the specifications set during the identification of the problem. All students should be familiar with the basic programming concept of the Micro:bit as well as the MakeCode programming environment.

Students are expected to be able to program the device (Micro:bit), at least for simple tasks such as temperature measurements and display of information.

Suggested activities:

- Identify what quantity of water a plant / garden requires (needs vary based on the type of plant)
- Suggest the most efficient use of school gardens, based on the soil and water availability of the school
- Understand terms such as soil humidity, air humidity, temperature, and how the two (temperature and humidity) are related.
- Suggest the most efficient use of components to create a stand-alone sustainable system
- Elaborate on how the prototype system could be expanded to cover the needs of an actual school garden

2.1.1 Connecting the Soil Moisture Sensor

Students are required to have working knowledge of Scratch / block based coding, as previously mentioned, as well as a familiarity with the (integrated) sensors of Micro:bit and its operation. To create a more complex system such as the one proposed in this guide, we will follow an iterative procedure that (a) allows students to familiarize themselves with the use of the external connectors of the Micro:bit and (b) understand the specific needs of connecting motors and external sensors to the device.

As a first step, students will connect the soil moisture sensor to the Micro:bit, and program it using MakeCode.

Figure 1: Capacitive Soil Moisture Sensor

The Soil Moisture Sensor has 3 pins, that must be connected (at this point) to the Micro:bit. The 3 pins are labeled as: VCC (Power), GND (Ground) and AO (Analog Output). For the first part of the guide, we will connect the VCC pin to the 3V connector of the Micro:bit, the GND pin to the respective GND pin, and the AO pin to the Pin 0 (P0). This last pin is responsible for reading the moisture value.

Sensor Pin	Micro:bit Pin	Explanation
VCC	3V	Powers the sensor
GND	GND	Grounds the circuit
AO	Pin 0	Sends the value of moisture to the Micro:bit

Table 1: Connectivity of Sensor and Micro:bit

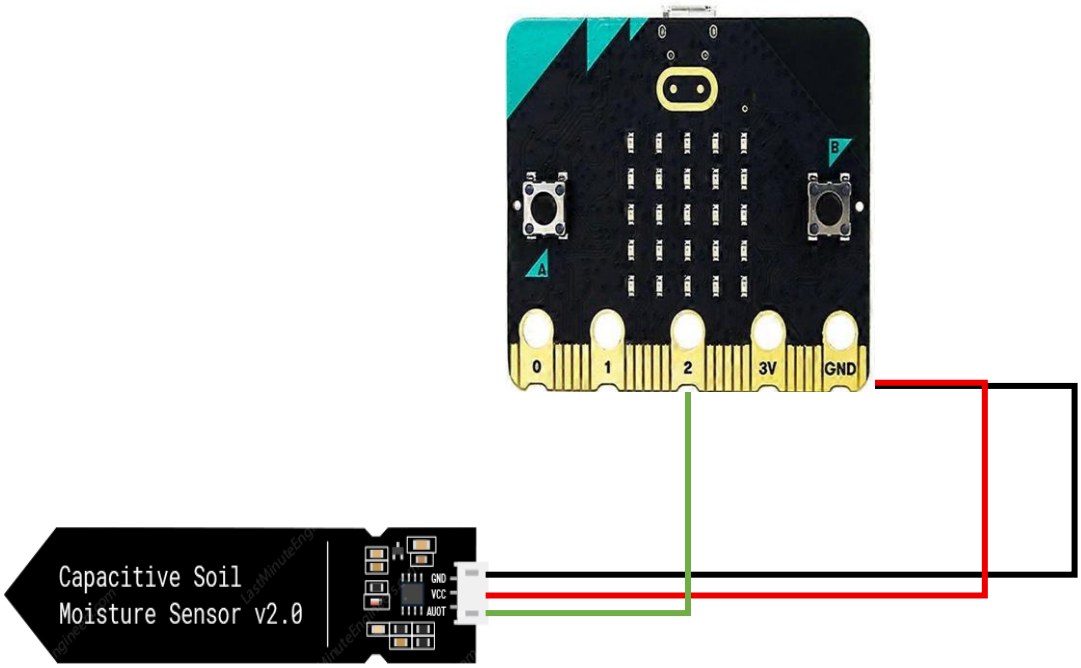


Figure 2: Connecting the Soil Moisture sensor to the Micro:bit

Next step is to program the Micro:bit to detect the moisture in the soil. Since we are using a capacitive (analogue) soil moisture sensor, it can provide a range of values from 0 (very wet) to 1023 (dry). Therefore, we can quantify the moisture level in the soil, rather than simply detecting its presence. This has significant implications, since we can program our irrigation system to provide the appropriate amount of water, not just turn on or off the water pump, something we will learn in the following stages.

As an initial testing of the setup, we will create code that detects soil moisture. For this purpose, we will place the sensor in real soil and develop a simple program to determine whether moisture is present.

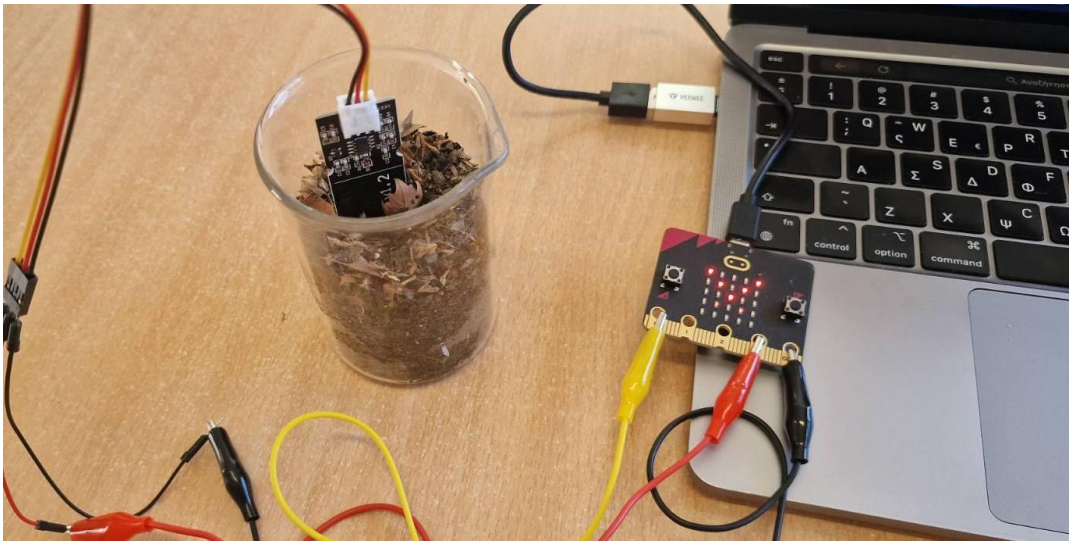


Figure 3: Test setup of Micro:bit in soil

2.1.2 Programming the sensor

Next, we code on MakeCode! We must start by creating a new variable called 'Moisture', which will store the value received from the sensor. This variable will help us determine not only whether moisture is present in the soil, but also how much.

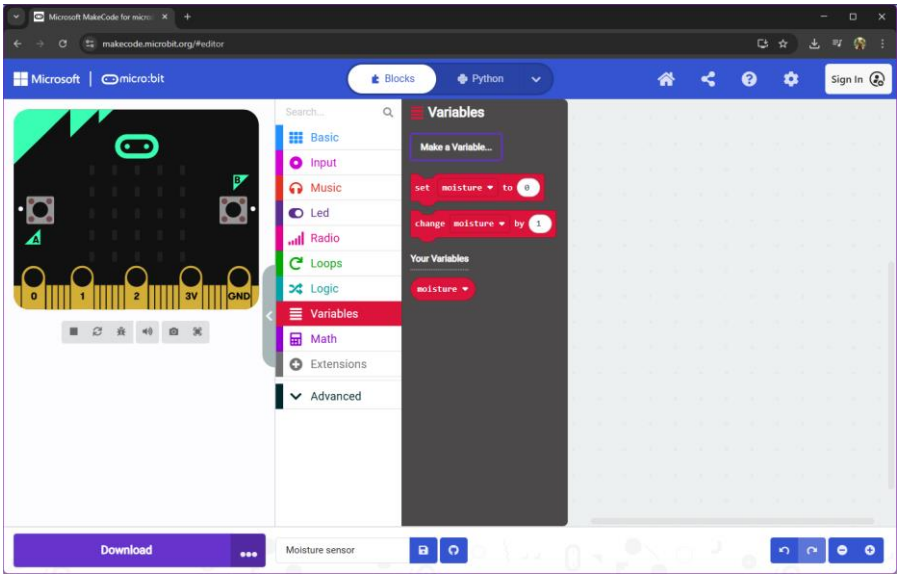


Figure 4: Creating the moisture variable

Our next step is to create the logic that will show the amount of moisture in the soil. First, we will need to store the value from the analog pin (P0) to our new variable. This is the value that is taken from the soil moisture sensor, as seen earlier.

For this, we will need to use the “Advanced” command set of MakeCode, and specifically the “Pins” section. We will read the analog pin (P0) and store its value in the “Moisture” variable.

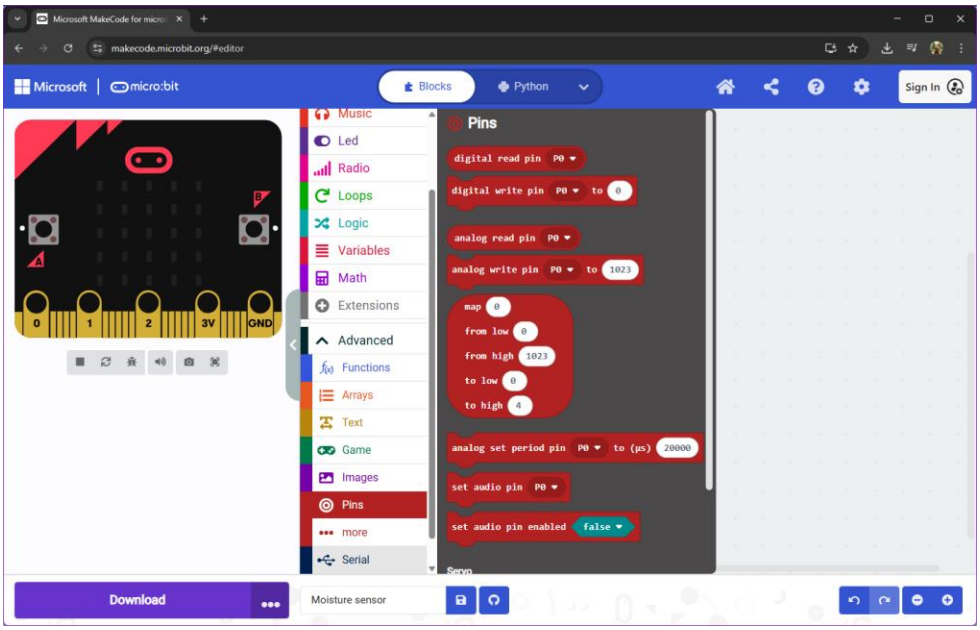


Figure 4: Advanced MakeCode commands

We drag and drop the blocks to create a first “binary” version of our code – in this, we will detect if there is moisture in our soil.

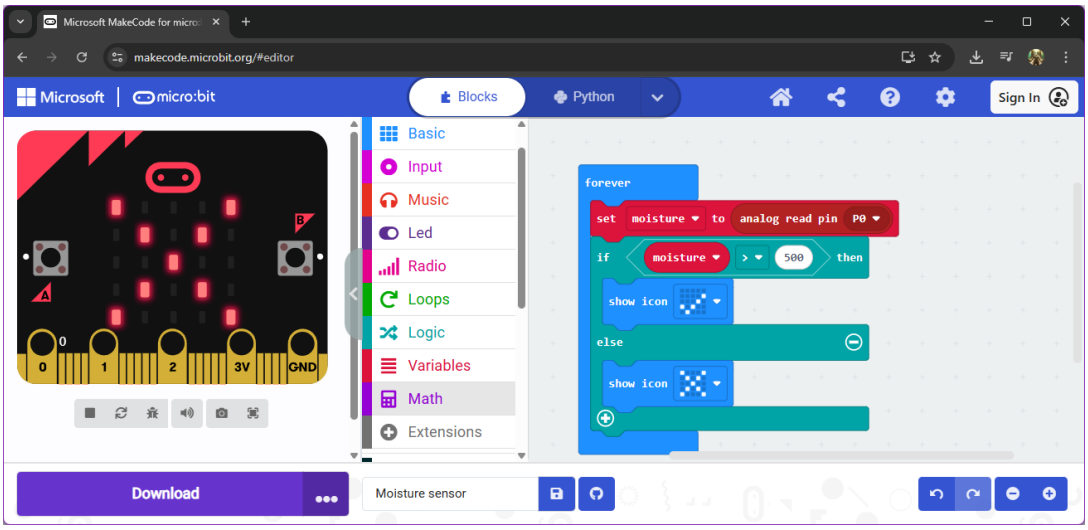
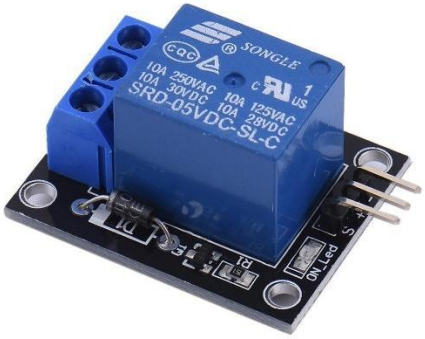


Figure 5: Checking the moisture of the soil

As we have seen earlier, the soil moisture sensor gives a value from 0 to 1023 (please note: some models give a value of 0 for wet, and 1023 for dry, other models give the opposite – check first your capacitive soil moisture sensor and make changes to your code accordingly).

Our next step is to connect the water pump. In the next section, we will understand how to connect a water pump to a Micro:bit and how to use it in conjunction with the soil moisture sensor.

2.1.3 Connecting the Water Pump



Connecting a water pump directly to a Micro:bit is not recommended, as it requires 5V for operation. A Micro:bit can provide 3V, therefore we will need to use a relay, connected to an external power source (4XAA batteries) or, alternatively, to solar panels for creating an autonomous, completely sustainable system.

Our relay has 3 pins (“S”, “+”, “-”). These are, essentially, the In (signal pin), VCC (for power) and GND (Ground). Table 2, below, shows the connectivity to the Micro:bit, and how each one of the pins works. Essentially, the relay acts as a switch between the Micro:bit and the pump, using an external power source to switch the pump on and off during operation.

Relay Pin	Micro:bit Pin	Explanation
In (S)	P1	Turns Relay on or off
VCC (+)	Power for relay	Powers the relay
GND (-)	GND	Grounds the circuit

Table 2: Connectivity of Relay and Micro:bit

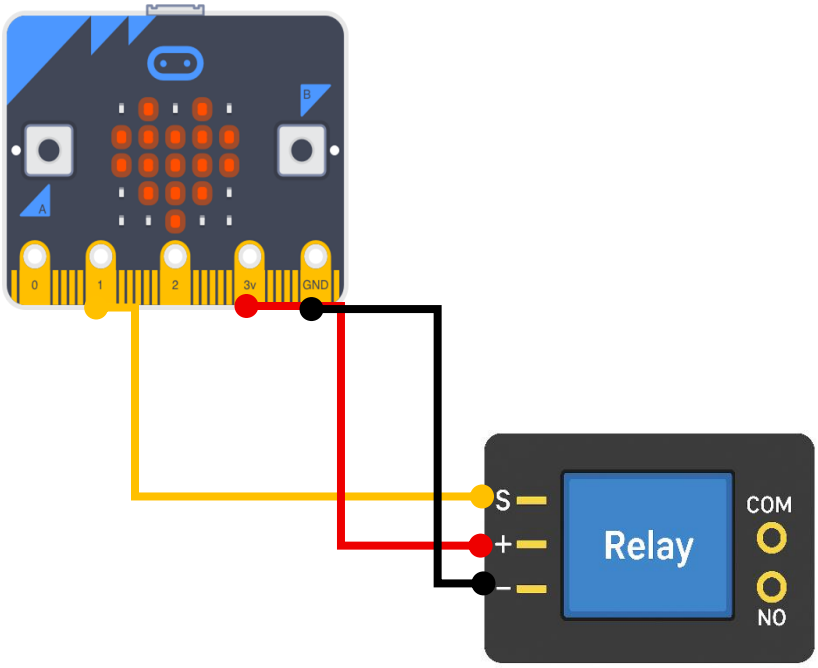


Figure 6: Connecting the relay to Micro:bit

Please note: Even though the relay will be connected to an external battery pack, it still needs to be connected to the Micro:bit. Our next step is to connect the relay to the water pump and the external battery pack.

For connecting to the water pump, we will use the two pins on the other side of the relay even though we have three available: COM, (Common Pin), NO (Normally Open) and NC (Normally Closed). For connecting to the battery pack and the water pump, we will only use two of the pins (COM and NO).

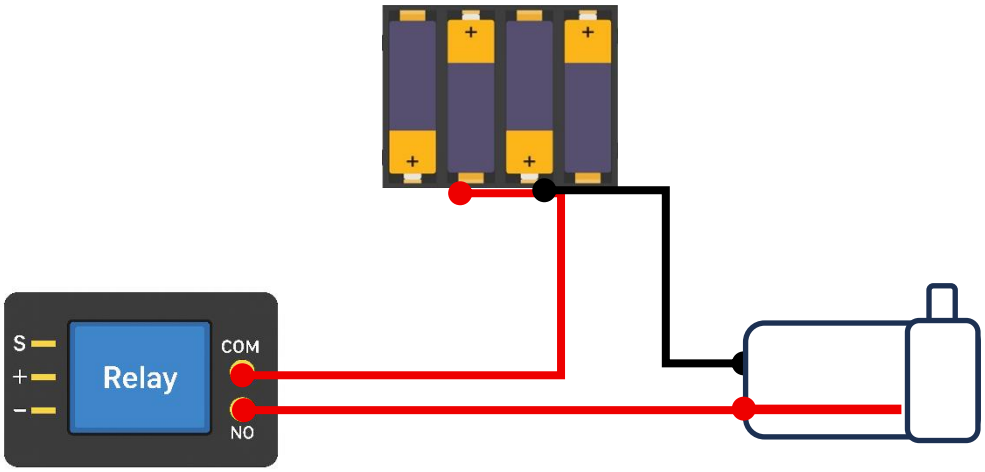


Figure 7: Connecting the water pump

2.1.4 Programming the water pump

We will program the Micro:bit to switch on and off the pump, using an external battery pack as a power source. We will need to be able to switch on the pump with the press of Button A, and again switch it off with the press of Button B on the Micro:bit.

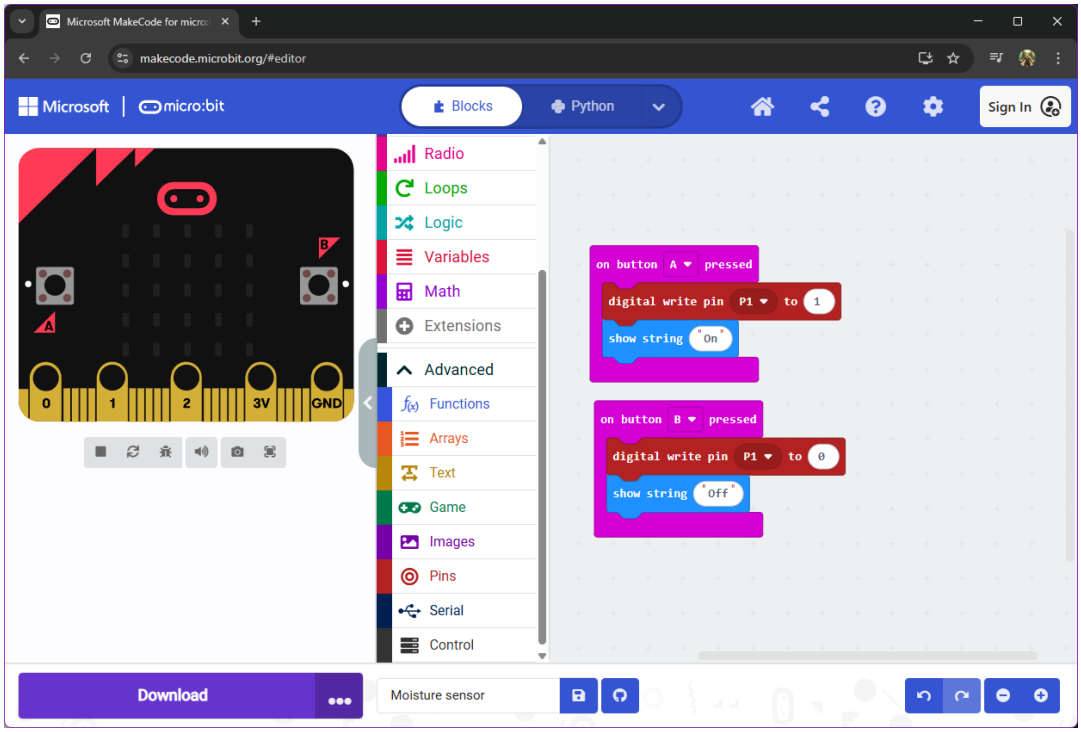


Figure 8: A simple code to switch on and off the pump

The code above is quite basic: pressing Button A on the Micro:bit turns the pump on indefinitely, while pressing Button B turns it off. Although simple, this setup is not very practical. For instance, what if the pump is already running and Button A is pressed again (regardless of what the Micro:bit LED display shows). The improved version below offers more functionality: it checks the current state of the pump (via the value on P1) and toggles it between 0 and 1 each time Button A is pressed, effectively switching the pump on or off with every click.

We will create a more efficient code which detects the pressing of Button A, and switches On or Off the pump accordingly. We will create the code in a Forever loop, since we want the Micro:bit to continuously check the pressing of button(s).

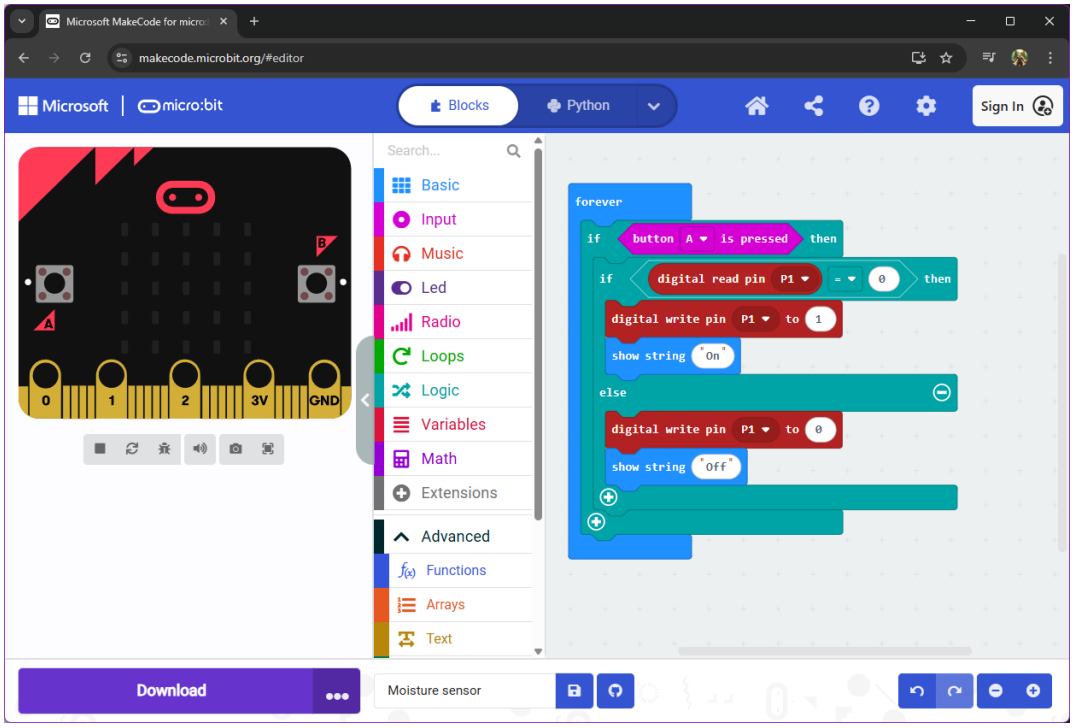


Figure 9: Switching the pump on and off

The code above is more advanced than the first example. In this case, when Button A is pressed, Micro:bit checks the state of the pump (the value of P1). If the pump is not working (P1=0), then it switches it on (P1=1). If the pump is working when we press the button, it switches the pump off.

Even though the above code is much more efficient than the previous one, it still has one significant problem: the pump will operate until we press Button A again. To conserve water, we will improve our code to let the pump work only for a set amount of time (for our example, 5 seconds).

To achieve this, we use the “pause” command (Figure 10), and we set the time to 5000ms (milliseconds). After the time elapses, the value of P1 will become 0, thus stopping the pump, and displaying the “Off” string on the LEDs of the Micro:bit.

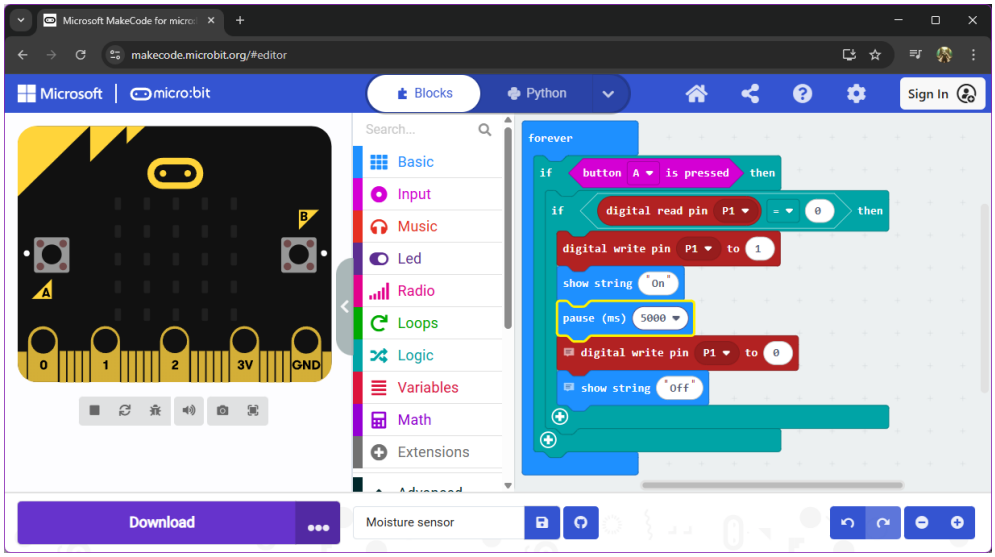


Figure 10: Switching the pump on for 5 seconds (5000 miliseconds)

Of course, this is not the desired result. Our goal is to create a fully automated system that activates the pump—and delivers water—only when it is truly needed. In the next section, we will connect all the components, including the soil moisture sensor, and program the Micro:bit to supply water when required, and only for a short period of time.

2.1.5 Connecting the components

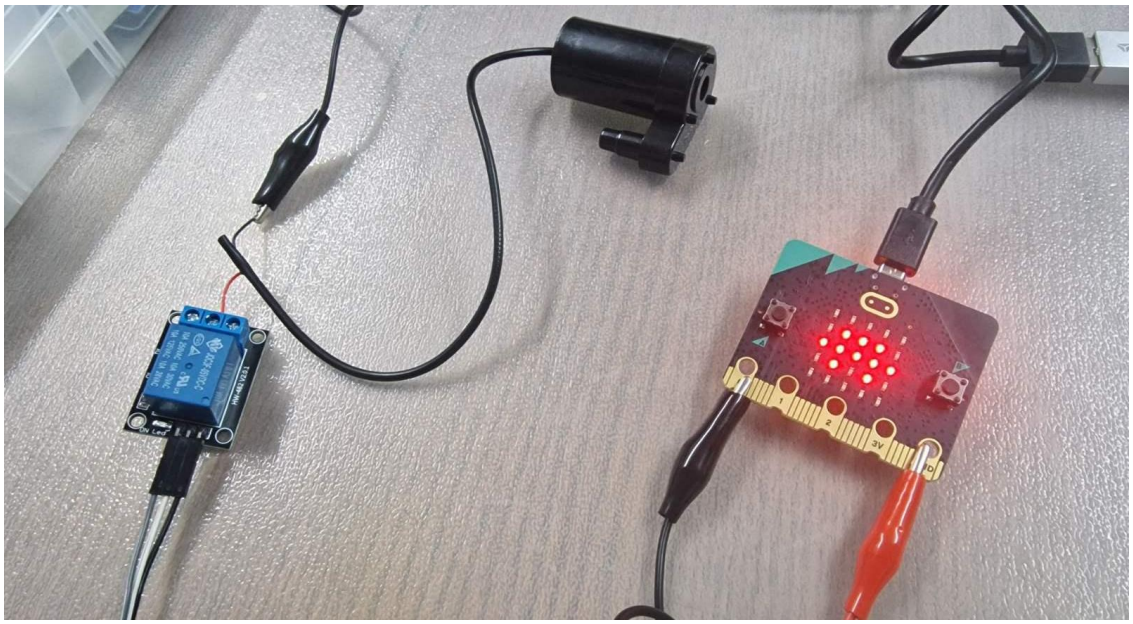


Figure 11: Attaching the relay to the Micro:bit

In our automatic irrigation system, we use the Micro:bit as the control unit. We connect the capacitive soil moisture sensor to the Micro:bit by attaching its VCC pin to the Micro:bit 3V, its GND pin to the Micro:bit GND, and its signal (S) pin to Micro:bit pin P0, so it can send soil moisture readings. The relay has two sides: the control side and the switch side. On the control side, we connect the S pin to Micro:bit pin P1, the + pin to Micro:bit 3V, and the – pin to Micro:bit GND. We make sure to connect the Micro:bit GND and the battery pack's negative terminal together so the relay works correctly. On the switch side, we connect the COM pin to the positive terminal of the external battery pack, and the NO (Normally Open) pin to the positive terminal of the water pump. Then we connect the pump's negative terminal to the battery pack's negative terminal. We power the Micro:bit separately via USB or its own battery pack, while the pump is powered by the external battery pack. When the Micro:bit reads that the soil is dry, we send a signal to the relay via pin P1, causing COM and NO to connect, which powers the pump. When the signal stops, the relay opens and the pump turns off.

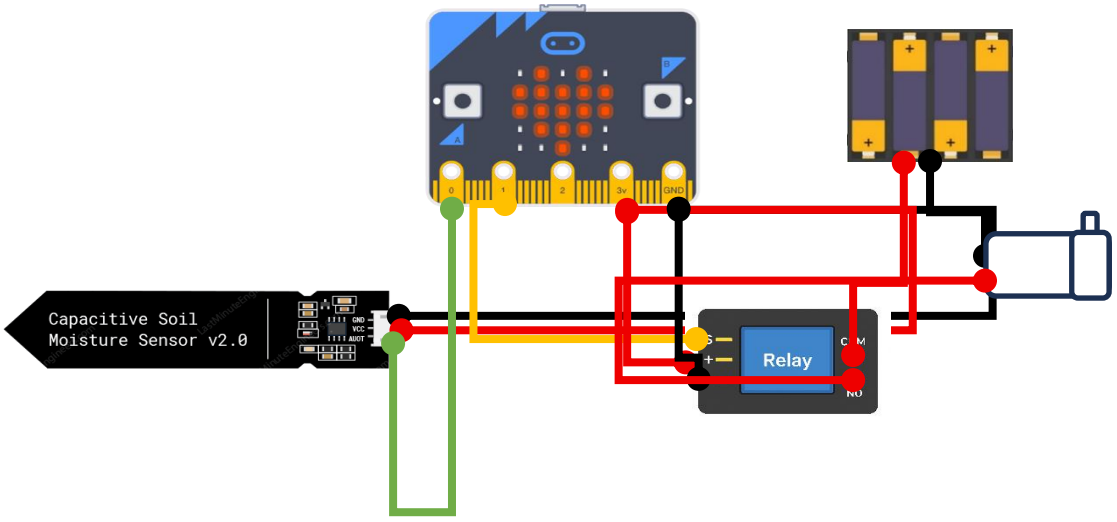


Figure 12: The complete setup

Once all the components are connected, we can proceed with the final code that will allow our system to be completely automated.

Please note: for the development of the system, and while in the testing stage, it is important to use both crocodile clips as well as (if deemed appropriate) at least one breadboard. However, since the setup is very basic, to test the functionality of the components, we can use only the crocodile clips. In some instances, different accessories giving access to more pins are available for the Micro:bit. However, to keep the cost low, our proposed setup requires the minimum number of components to be functional.

To test our setup, we will need to program our Micro:bit. We will program our device to decide when to switch the pump on and provide water. As such, the capacitive soil sensor will decide when to operate the pump and for how long. For the next example, we assume that the condition of the soil is either dry or wet, and therefore the pump will either operate or not. Essentially, we are replacing the pressing of Button A, with the value of the variable "Moisture". On certain soil moisture sensors, a reading of 0 indicates dry soil, while 1023 represents very wet conditions, with intermediate values reflecting varying moisture levels. However, on other sensors, this scale may be reversed—where 0 corresponds to wet soil and 1023 to dry. In our case (example below), if the value of moisture is over 400, then it switches on the pump for 5 seconds

(5000 ms). All the code is within a Forever loop, ensuring continuous monitoring of the soil's condition and moisture level.

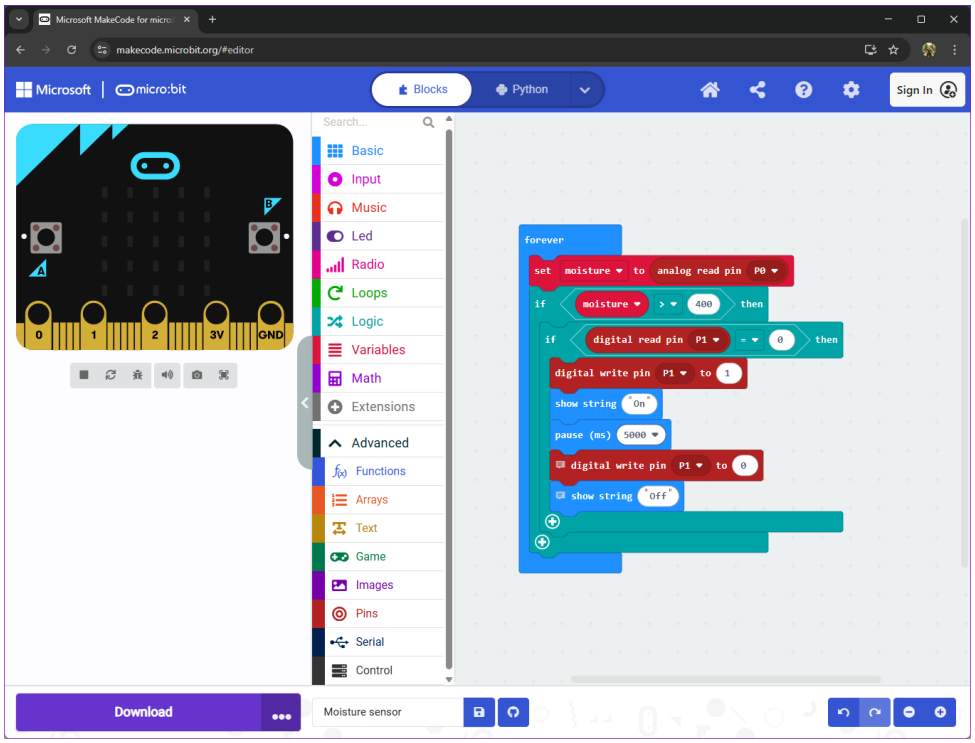


Figure 13: The code for the irrigation system

Next, we will take our setup to the next level: we will turn this simple irrigation system into a completely autonomous, remotely working system that can send data through the internet, regarding the soil humidity. For that, we will need to connect the Wi-Fi module, and link our system to an online data logging platform.

2.1.6 Connecting Wi-Fi

The Micro:bit is very capable for its size and price. However, despite all its features, it does not (currently, at least for version 2) have built-in Wi-Fi capabilities. Fortunately, Wi-Fi modules such as the ESP8266 are available, and are very inexpensive (around 3-4 euro each).

The Micro:bit cannot connect directly to a Wi-Fi module such as the ESP8266 because their communication systems are not fully compatible on their own. The Micro:bit operates with a 3.3V logic level but cannot provide enough current or use the correct serial communication pins required by the Wi-Fi module. Additionally, the ESP8266 requires a stable power source and a proper serial interface to exchange data, which the Micro:bit alone cannot handle. For this reason, we use an expansion board that acts as a bridge, providing the correct voltage levels, stable power supply, and appropriate connections. The expansion board also makes it much easier to connect multiple components, including sensors and modules, without damaging the Micro:bit.

For this tutorial, we have selected the ESP8266 ESP-01S Wi-Fi Module, and the Zalati IOBIT Adapter for Micro:bit.

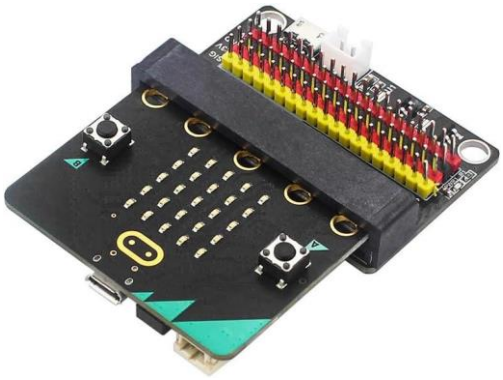


Figure 14: The Micro:bit attached to the zalati IOBIT adapter

The Micro:bit easily attaches to the zalati adapter board and provides access to all the input/output pins, power lines, and ground connections through clearly labeled headers. This board allows the Micro:bit to communicate with external modules, such as the Wi-Fi module, while also providing access for additional components.

With the Micro:bit connected, the board allows us to connect external modules and other components without the need of soldering or complex wiring. Our next step is to prepare the Wi-Fi module, in this case the ESP8266 ESP-01S. The module requires a stable 3.3V power supply capable of providing sufficient power for Wi-Fi transmissions.

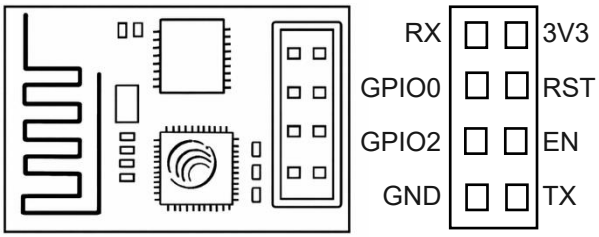


Figure 15: The Wifi module pins

IMPORTANT: The Micro:bit on its own, cannot provide sufficient power to run the ESP8266 module. However, the expansion board can provide the additional power (see Table 3).

ESP8266 Pin	IOBIT adapter	Explanation
3V3	3V	Supplies 3.3V power to the module.
GND	GND	Connects ground to module
EN (enable)	3V	Must be connected to 3.3V to keep the module powered
TX (Transmit)	P1 Signal Pin	Data transmission from the ESP8266 to the Micro:bit
RX (Receive)	P0 Signal Pin	Data reception by the ESP8266 from the micro:bit (micro:bit TX).

Table 3: Connectivity of Wi-Fi Module and Micro:bit expansion board

As an external source, it is recommended to either use a separate power pack connected to the board, or a power bank. Similarly, the Micro:bit should be connected to an external battery pack itself.

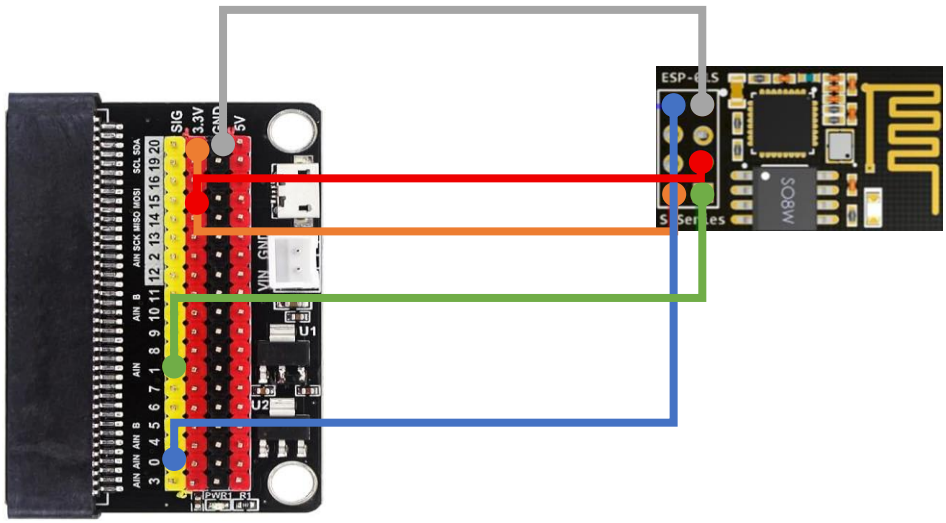


Figure 16: Connecting the Wi-Fi module to the expansion board

2.1.7 Connecting all the components

Now that we have connected the Wi-Fi module, it is time to finalise our setup: we will add the rest of the components on the expansion board! In the previous sections (2.1.1 to 2.1.5), we were introduced to how each component works on its own. Now, we will finally put everything together!

First, we need to add the most important component: the Soil Moisture Sensor! We connect the VCC pin from the sensor to the 3V pin on the board. Then, we connect the GND pin to the respected GND pin on the board. Finally, the A0 pin from the sensor to the P2 Signal Pin on the board. This is the pin that will read the moisture level!

Soil Moisture Pin	IOBIT adapter	Explanation
VCC	3V	Power the sensor module
GND	GND	Connects to Ground
A0	P2 Signal Pin	Read the moisture value

Table 4: Connectivity of Soil Moisture Sensor to Micro:bit expansion board

Next, we will have to connect the relay that will help us control (on and off) the water pump. We connect the S pin of the relay to the P8 Signal Pin on the board. We then connect the VCC pin to the 3V (or 5V) pin on the expansion board. Finally, we connect the GND pin from the relay to the GND pin on the board, as shown in Table 5 below.

Relay Pin	IOBIT adapter	Explanation
In (S)	P8 Signal Pin	The control pin that receives the digital signal from the Micro:bit to turn the pump On and Off
VCC (+)	3V or 5V	Supplies power for the relay module
GND (-)	GND	Connects to Ground

Table 5: Connectivity of Relay Pin to Micro:bit expansion board

Please note: the pump will still have to be powered by an external power source, as shown in earlier paragraph.

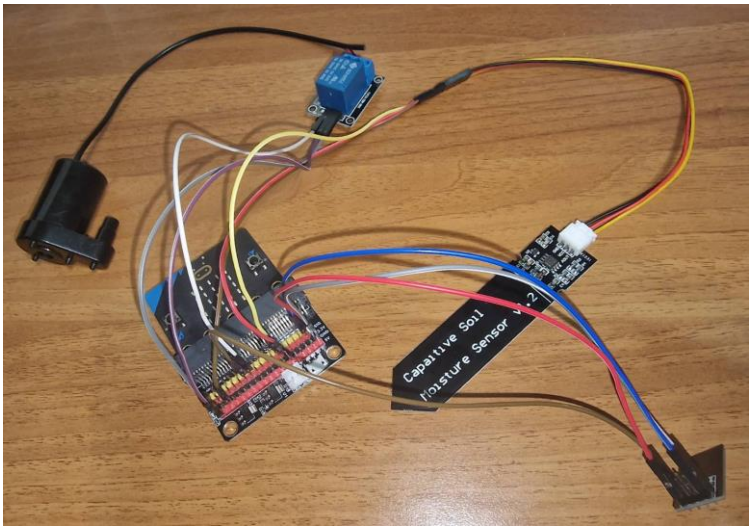


Figure 17: The components including the Wi-Fi module

2.1.8 Connecting to an online IoT Platform

In section 2.1.5, we programmed the basic setup of our system to check if the soil is dry, and turn the pump on for a specific time. Next, we will connect our system, via the Wi-Fi module, to an online platform that will allow our Micro:bit to transmit data from the moisture sensor. There are various platforms that offer basic free accounts, like ThingSpeak, Blynk, Adafruit IO and others.

In the next pages, we will examine three of these platforms, and we will explain how to connect to the ThingVerse for our project.

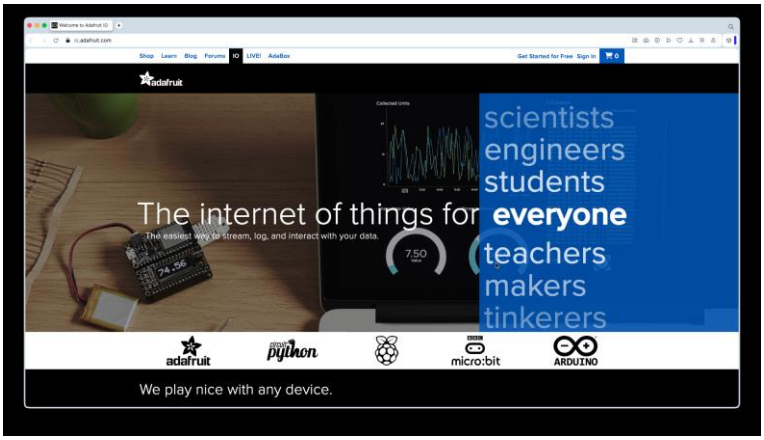


Figure 18: The Adafruit IO Platform

One such platform is Adafruit IO (https://adafruit.com), an easy to use platform for connecting devices such as Micro:bit. First, we need to create a free account. This will allow us to connect up to 5 devices (enough for 5 groups) to the platform. Once we create our account, we click on the “IO” link on the top of the page to view our dashboard (Figure 18).

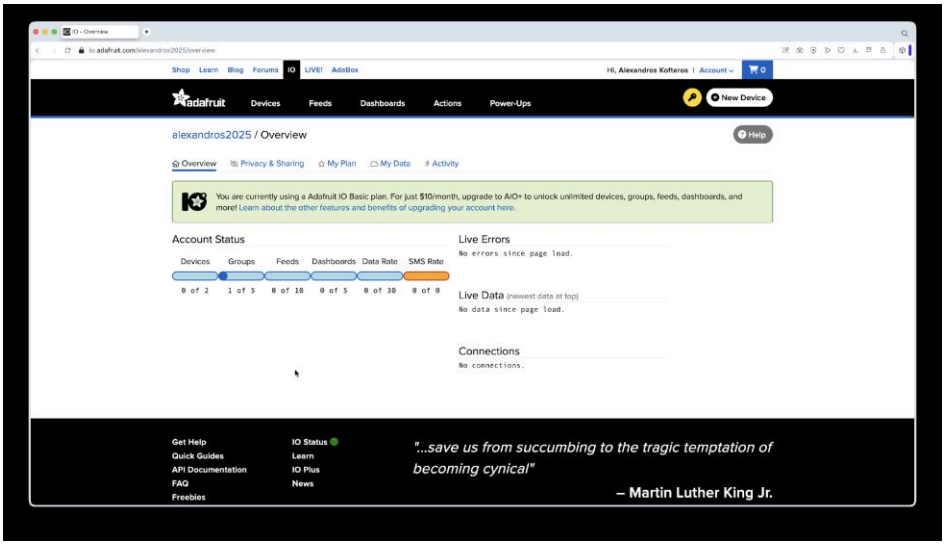


Figure 19: The dashboard with our devices

Our next step is to create a new Feed. We click on the “Feeds” menu, to view our feeds. A feed is essentially a stream of data that we publish to the Adafruit IO platform. This is where all the data from our soil moisture sensor will appear. We click on “New Feed” to -surprise!- create a new feed, as we can see in the image below (Figure 19).

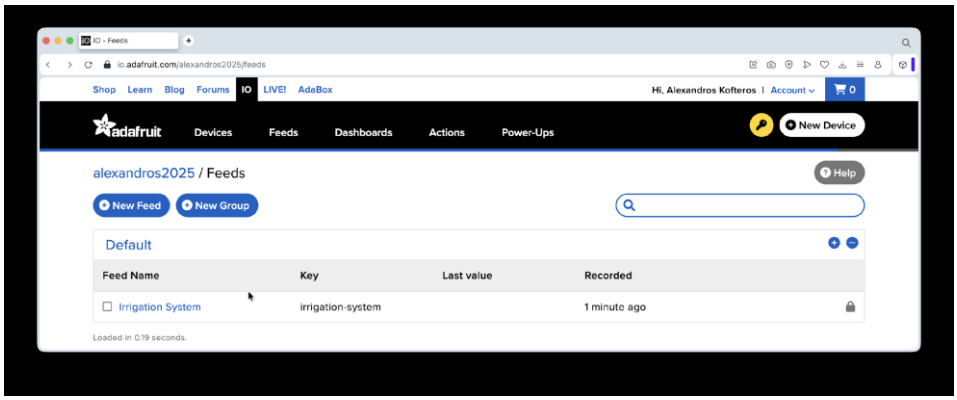


Figure 20: Our feeds

We will now program our system to receive data from the soil moisture sensor and transmit, via the Wi-Fi module, to our feed on the Adafruit IO platform!

Another solution that allows great flexibility is Blynk (https://blynk.io), an IoT platform that supports free educational (albeit a bit limited) accounts.

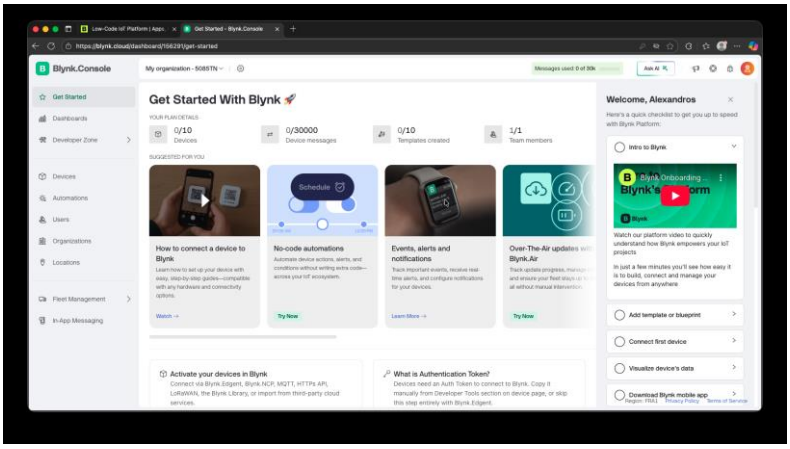


Figure 21: The Blynk IO page

From the main screen, we click on “Developer Zone” (on the left side) and then we click on “New template”. We select the Micro:bit, and then we click “Create”.

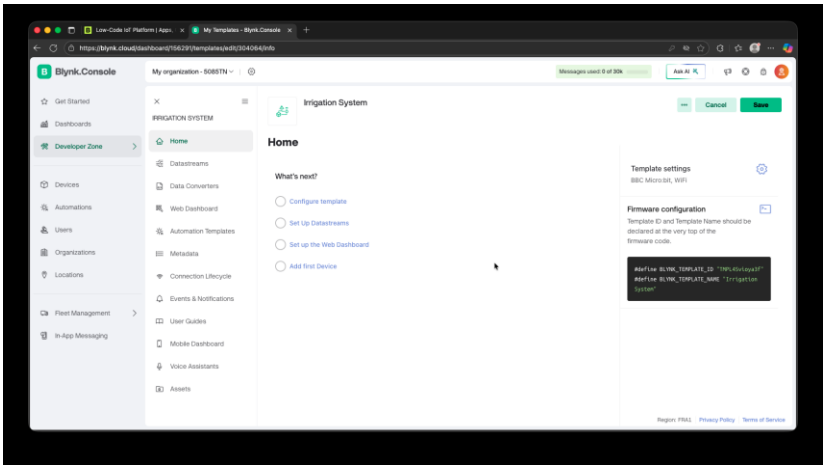


Figure 22: Creating a new Datastream

Next, we click on Datastreams (image above) and create a new Datastream.

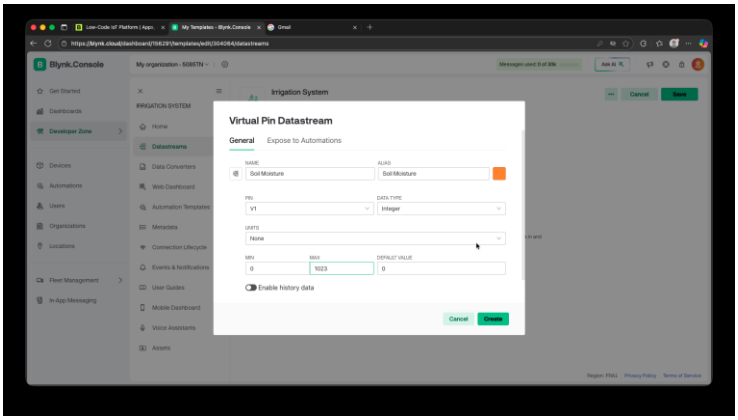


Figure 23: Editing the Datastream

It is important to name our Datastream with a name that signifies the data it will send. We choose one of the available pins, the data type (soil moisture level is integer) and the range (minimum value for soil moisture is 0 and maximum is 1023). We click on “Create” to finish this part of the procedure.

Next, we go to the Web Dashboard to add the chart for our data. We click on the Widget Box and locate the Chart. We drag and drop the chart widget in the right part of the screen, and then we click on the wheel to add the Datastream we created earlier. We click Save to finalise our changes.

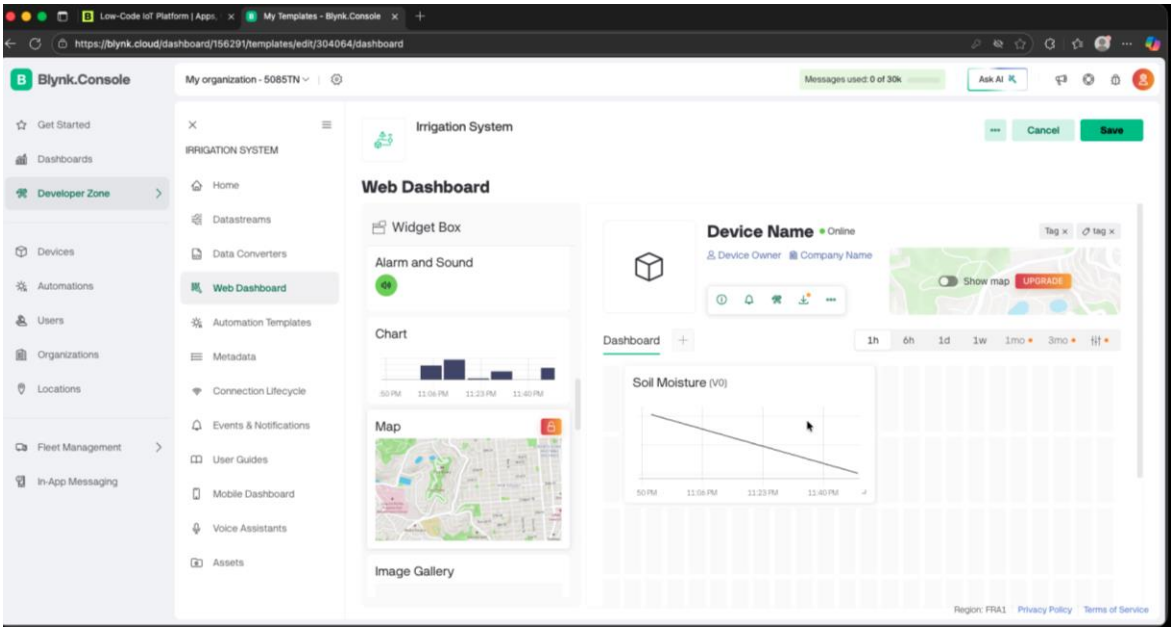


Figure 24: The Chart Widget

The platform we will use for our examples is ThingSpeak. It offers the best possible combination of ease of use with free tools that are sufficient for our needs. We must first create a (free) account, in a few very simple steps, from page (https://thingspeak.mathworks.com/).

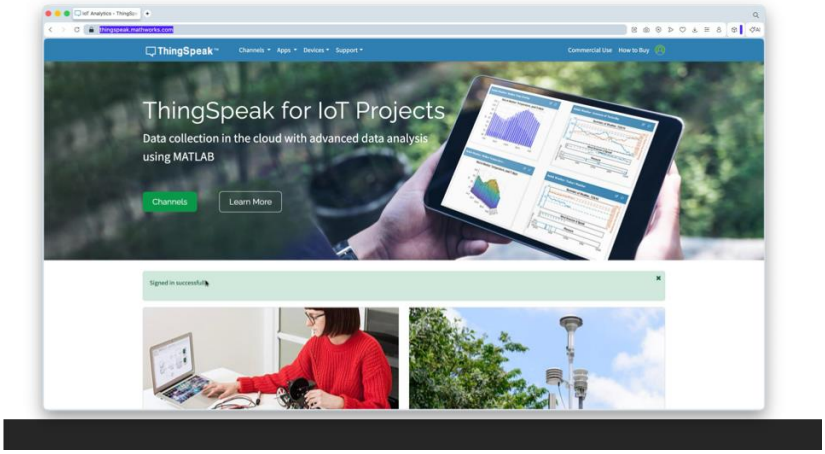


Figure 25: The ThingSpeak platform

After we create our account, we can log into our Channels (or create a new one). Each channel can display information from different devices. For our example, we will click on “New Channel” (image below left). Once we create a new channel, we need to give it a name that is related to the project we are created. Then, we decide what information we transmit to the platform. In our case, we will just send data for the soil humidity. We click on the tick box in Field 1 to enable it, and then we type the name of the field. This is very important, since we will connect this to our code later on! At the bottom of the page we can find the “Save” button, to complete the process. We will then be presented with the Channel area (Figure 27).

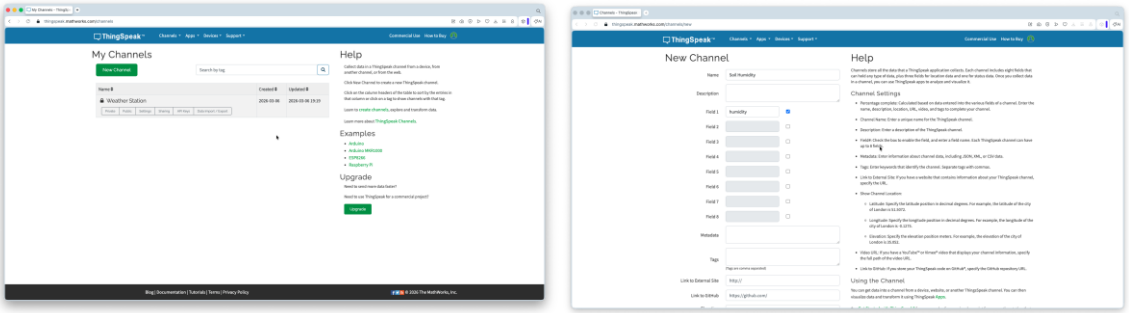


Figure 26: Creating a new channel

This part is very important – we can have multiple visualisations, based on the fields we created and, of course, the data we will send via our wifi module. For the time being, we will only use the soil moisture sensor. Therefore, we will only have one chart. Our next step is to get the API key we will use to connect to our platform and to the specific channel. We click on the link “API Keys” (image below).

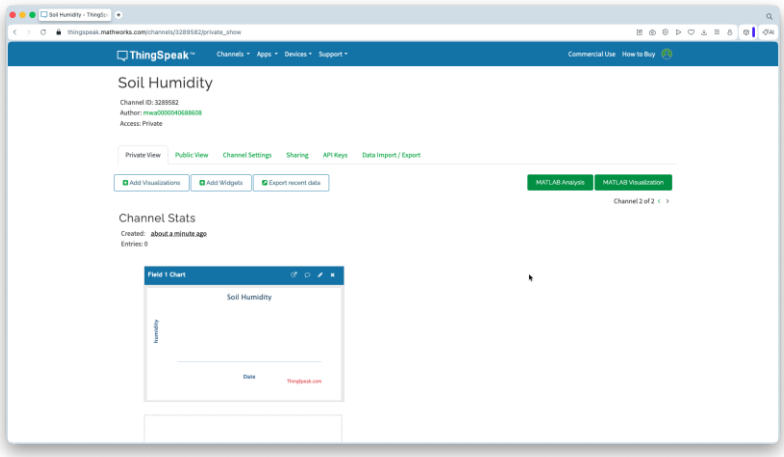


Figure 27: Visualisations and API Keys

As we are going to send data to the platform, we copy the Write API Key (image below). We will use this in the code we will create in the next pages.

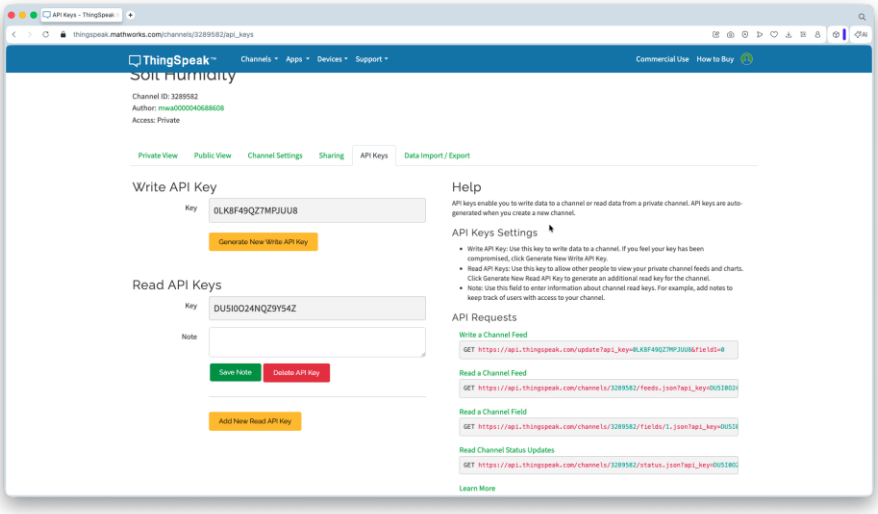


Figure 28: The API Keys

2.1.9 Programming the system

Now is the time to finalise our setup and get everything working: the soil moisture sensor to check the humidity of our garden, the pump to be switched on or off accordingly, and the data from the sensor to be sent to our feed on the ThingSpeak platform. We need to add an extension to our MakeCode environment. We search for “8266” to provide drivers for our Wi-Fi module. From the list, we select the “ThingSpeak” module (Figure 29).

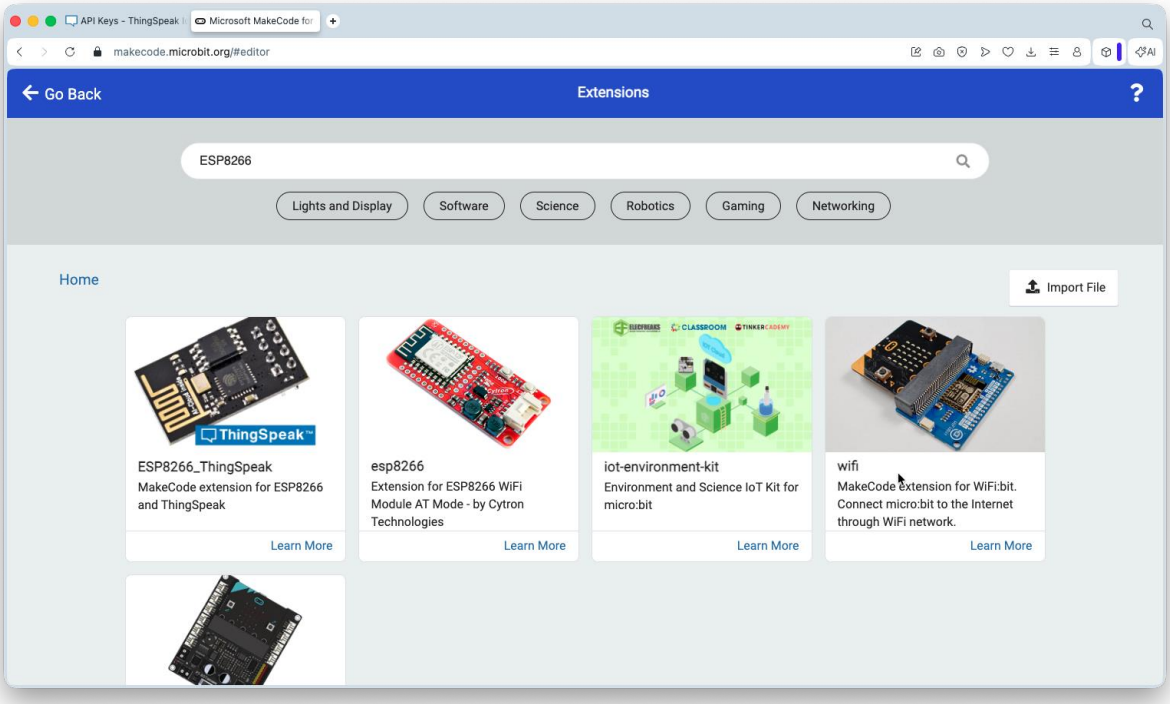


Figure 29: The Wi-Fi Module

Once we have added the extension, we can see the new commands available in our MakeCode screen (Figure 30). We also see that Blynk (and other platforms) have been automatically added.

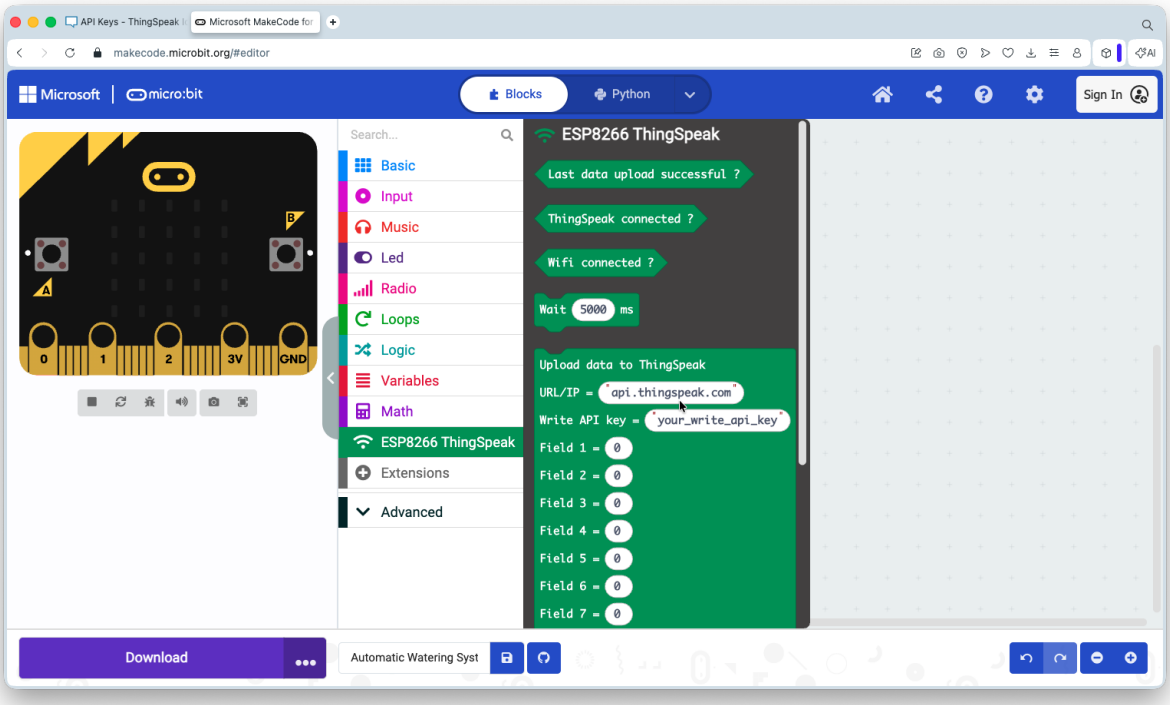


Figure 30: The new ESP8266 ThingSpeak commands

Before we proceed, we will test and initialize our Wi-Fi Module. Drag and drop the “initialize ESP8266” module to the “on Start” block and make sure you change the pins according to how each part was connected to the system. Next, we will add the SSID and password for our Wi-Fi. Now that we have initialized our Wi-Fi, we can start receiving data from our Soil Moisture Sensor!

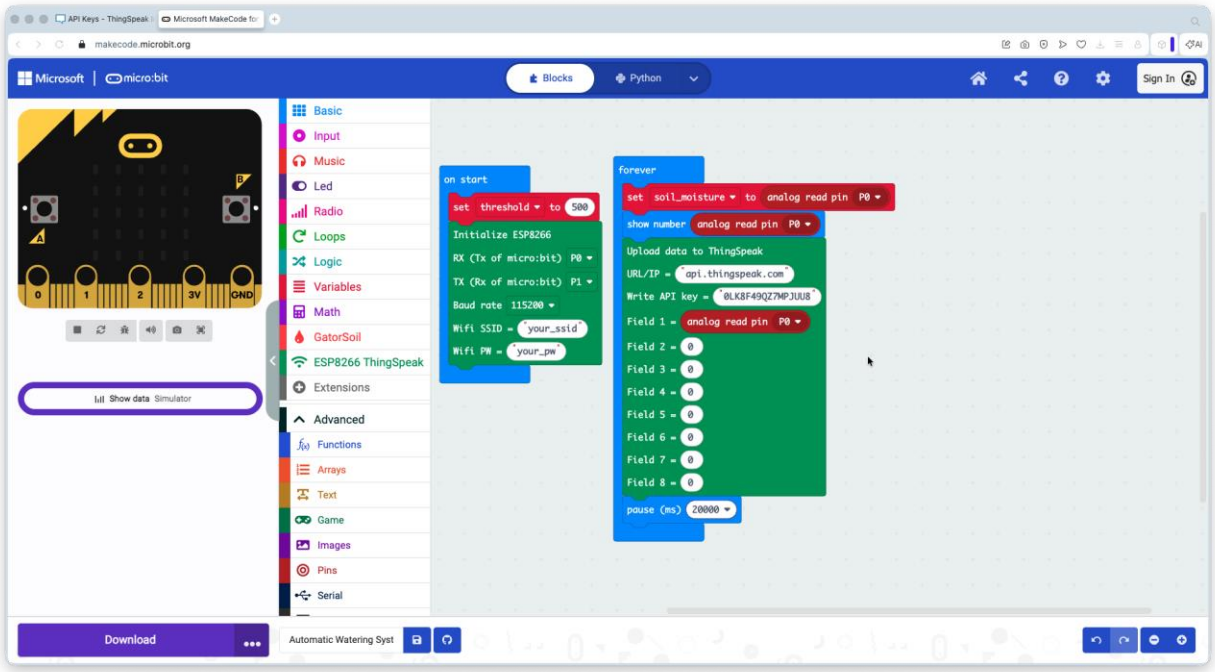


Figure 31: Writing the code

The code above has similarities in its logic with the earlier code we examined earlier. However, in this code, we initiate the Wi-Fi module (on start) and we connect to the Wi-Fi network.

Next, inside the “on forever” block, we read the indication from the soil moisture sensor (pin P2) and we display it on the LEDs of the Micro:bit. As you can see in the above image, we send the data that is read from the analog pin on which we have connected the soil moisture sensor. This data is transmitted to the ThingSpeak platform constantly. Of course, we need to do refinements to that code, but that will happen in the next steps.

To initiate the water pump, we will have to check if the moisture of the soil is below the threshold we have set. The indication from the soil moisture is compared to the threshold we have set as a variable earlier (on start). If the indication is lower than the threshold, the pump is switched on to provide water. Otherwise, the pump is either switched off or remains off. The data from the moisture sensor is sent to the ThingSpeak platform, with a pause of 20000ms between readings.

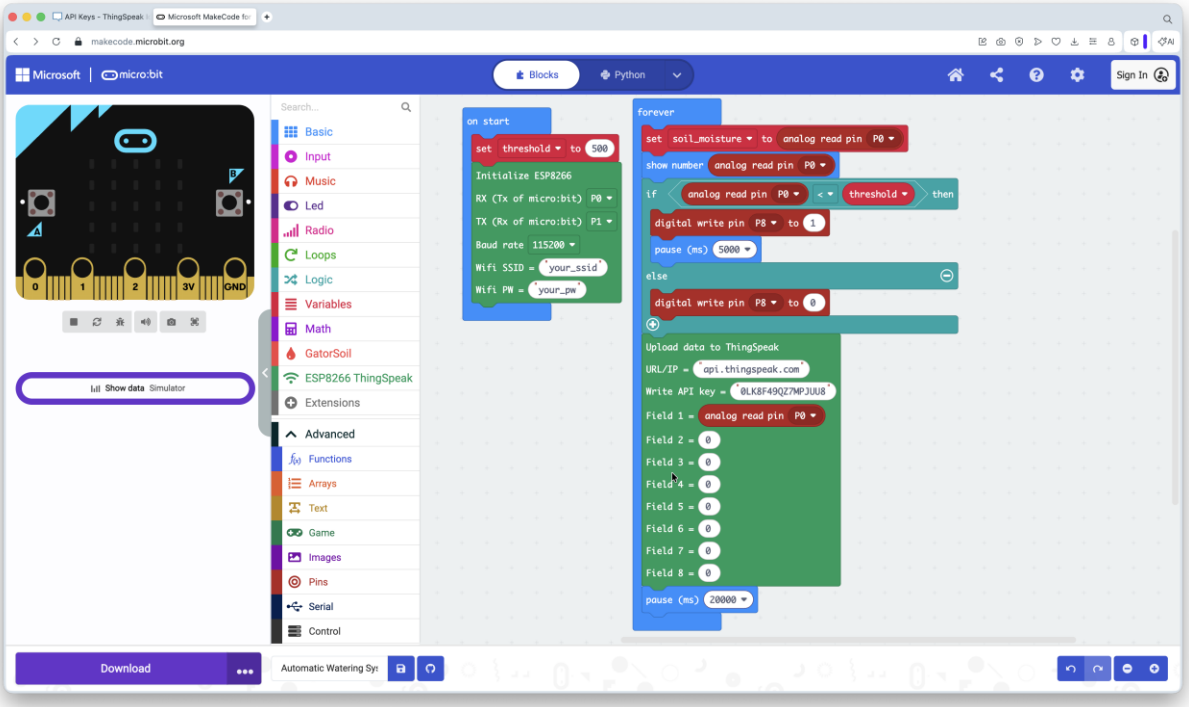


Figure 32: The final code

2.1.10 Designing a case for the system

Automated watering systems (or irrigation systems) already exist as kits sold online, and in many cases there are also tutorials, both for Arduino and for Micro:bit devices. Some of these are very simple (measuring the humidity of the soil), others are more complex, featuring even more sensors (for example, measuring the quantity of the water in the water tank). A selection of ready-made projects are featured in the Appendix of this document.

Even though designing and creating a case for the irrigation system is not part of the scope of this guide, and it would require allocation of more time, this is an important activity that can be applied in other subjects (Design & Technology, Arts) as part of a more complete, cross-curricular STEAM approach.

Students can use ready-made materials that are water-resistant (or at least are not damaged by water) to encase the Micro:bit and its components. Acrylic cases are readily available and come in all shapes and sizes and can be used effectively to store all the components of a Micro:bit and its motor and relay (Figure 15). At the same time, if there is access to a 3D printer at school, students could use TinkerCad to design their own case as part of a team project, to encase the Micro:bit. This could be assigned to the teams of students as a follow-up project (see Section 3 of this document).

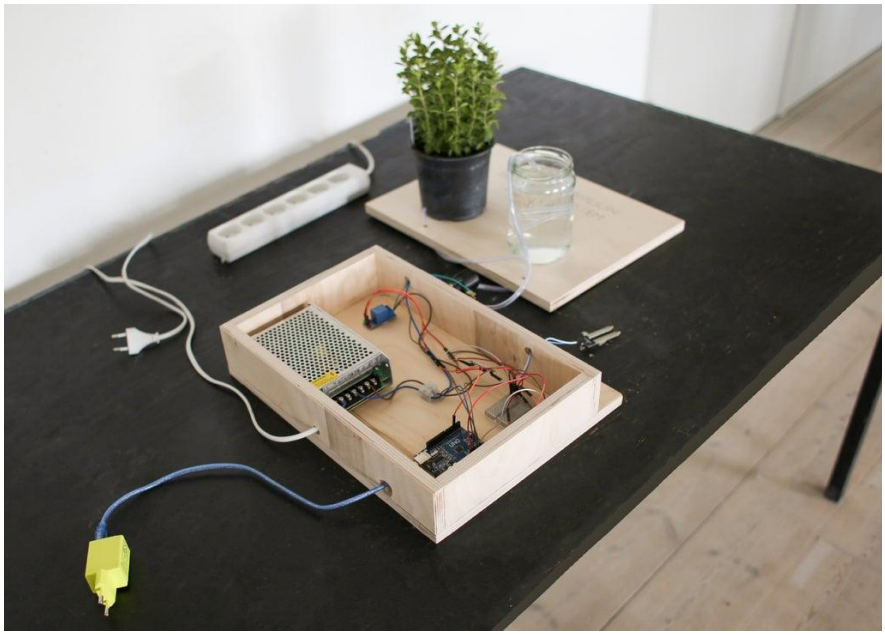


Figure 33: Wooden case for Arduino-based automatic watering system (Image from: https://www.instructables.com/Arduino-Plant-Watering-System/)

Figure 34: Acrylic case for Micro:bit-based automatic watering system (Image from: https://www.instructables.com/Automatic-Plant-Watering-System-Using-a-Microbit/

3 Tips and recommendations

3.1 Further expansion of the project

It is recommended that students discuss how to improve the system, and ensure that it can be functional, under all conditions. They can identify further problems, such as the need to keep the components safe from the weather conditions and even test the system in the actual garden (but, with more modest expectations on the watering of a single plant instead of an actual, full-size school garden). Students can be expected to identify possible solutions, and the main specifications of their updated and upgraded system (for example, use of water-resistant material to encase the circuits, more functional setup with less cables depending on crocodile clips, etc).

Additionally, students can suggest new features of the system, such as (as mentioned earlier in this document) a sensor to measure the quantity of the water in the tank used by the pump to water the plants. For example, they could use a water level sensor (Figure 16). These sensors are placed in a water tank, and they give a reading on the height of the water (thus measuring the quantity available). Even though they have some similarities with soil moisture sensors, they are fundamentally different as these are placed in water, and not in the soil.

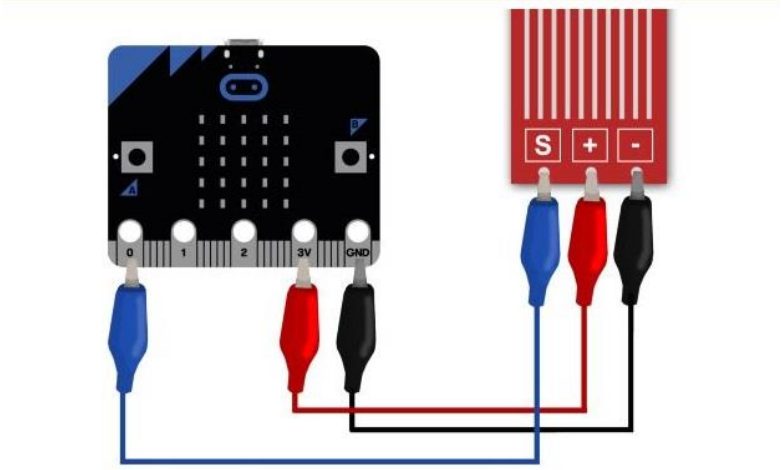


Figure 35: water level sensor (Image from: https://pishop.ca)

By using water level sensors, a Micro:bit can show the level of water using its LED lights or make warning sounds using its speaker when the quantity of water is low (i.e. below 50%). Students should be allowed to experiment and make their own suggestions.

The overall code of the system can also be improved. For example, in the code we used for the examples in this guide, the system checks a specific value of the variable “Moisture” we have set (400) and then it switches the pump on for 5 seconds. Students can program the system to be more efficient. For example, they can detect the amount of moisture in the soil and then let the pump work for a specific (instead of a set) amount of time.

3.2 Sustainability of the system

The above system works on an external battery pack (4XAA) that requires replacement or recharging (depending on the batteries used). However, this might pose a problem when it is required to replace them regularly, or charge them, making the system less sustainable. The use of solar panels, especially (at least)

4x1,5 V connected in series, can provide enough power for the pump to work. An additional array of solar panels can be used to power the Micro:bit as well. For powering the Micro:bit, at least 2 solar panels (1.5V, 100mA) connected in series can be used. Again, this can be an added project for the students, to measure the power requirements of the proposed setup and turn it into a completely autonomous and sustainable irrigation system.

3.3 3D printing system case

More schools acquire 3D printers, for use in various subjects. In case your school has access to a 3D printer, either in-school or as a service (some schools email 3D models and are printed by other companies/organisations), you can experiment with creating cases for the Micro:bit and the rest of the components.

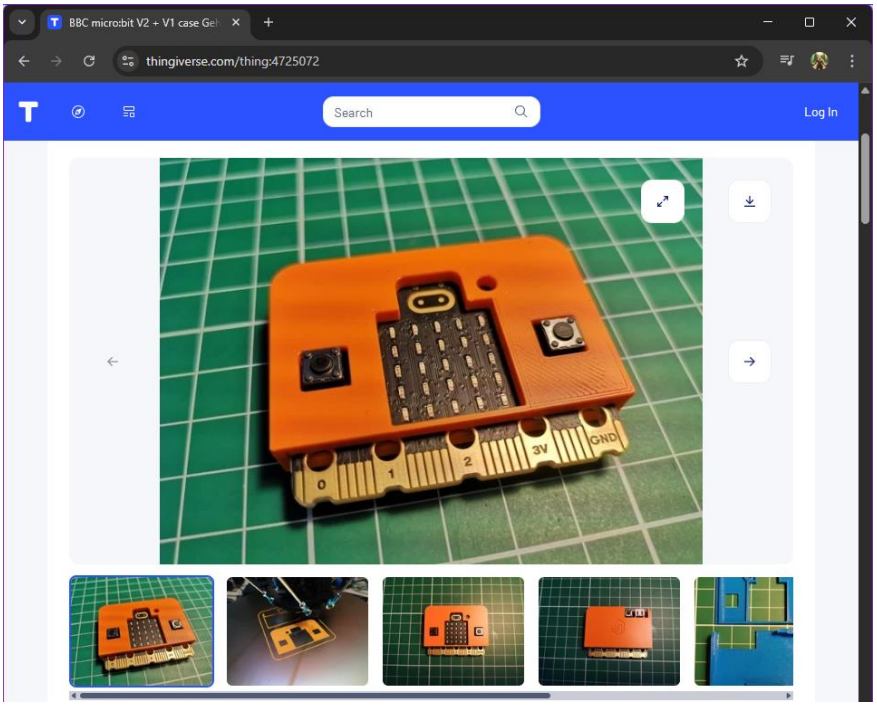


Figure 36: Example of 3D printed Micro:bit case (Image from: https://thingverse.com)

Many models are already available online, and even included in easy-to-use software such as Tinkercad. Students can design the case for their system, including any and all holes for the wiring, and then send them to a 3D printer. Students can experiment with designing cases and improving their design under real conditions.

3.4 Discussion

When creating complex projects that require the acquisition of new skills and knowledge, students, as active learners, have a lot to share. During the process, we encourage students to make comments on their system, as well as provide constructive feedback to their peers. At the end of the project, where students have a more thorough understanding of the process and the end results, it is imperative that we allow and encourage them to express their ideas, suggestions, objections. Through this discussion, which might even prove to be the most important part of the entire process, students solidify their ideas and their own role in the project.

4 References

- **BBC Micro:bit Foundation.** (n.d.). *Powering your micro:bit.* https://support.microbit.org/support/solutions/articles/19000013982-connecting-a-power-supply-to-the-micro-bit
  It describes different ways to power the micro:bit using batteries or USB, and how to avoid overloading the board.
- **BBC Micro:bit Foundation.** (2020). *Automatic plant watering system project idea.* Retrieved October 10, 2025, from https://microbit.org/ideas/
  A similar micro:bit-based plant watering project that combines coding and environmental monitoring.
- **Kitronik Ltd.** (n.d.). *Using a relay board with the BBC micro:bit.* https://kitronik.co.uk/
  A clear guide on connecting a relay module to the micro:bit to safely control devices like pumps or lights.
- **Microsoft MakeCode.** (n.d.). *micro:bit power and accessories guide.* https://makecode.microbit.org/
  Explains how to power the micro:bit, use its accessories, and program it using the MakeCode editor.
- **BBC Micro:bit Foundation.** (n.d.). *Plant watering*
  https://makecode.microbit.org/projects/plant-watering
  Classroom materials and project ideas for building a plant watering system with sensors and code.

5 Appendix

5.1 Glossary – Definitions of key concepts

1. Micro:bit
   A small programmable microcontroller board used for learning coding and electronics. It controls the watering system by reading sensor data and sending commands to the relay.

2. Relay
   An electronic switch that allows the Micro:bit to control high-power devices (like a water pump) using a low-power signal. It has control pins (S, +, -) and switch pins (COM, NO/NC). The relay operates based on electrical voltage and can switch on/off in milliseconds.

3. Water Pump
   A device that moves water from a source (e.g., a water tank) to plants. Controlled via the relay, it turns on or off depending on the moisture level detected by the soil sensor. The pump's operation time is often measured in milliseconds or seconds.

4. Crocodile Clips
   Metal clips with insulated handles used to connect electrical components together without soldering. They allow flexible, temporary connections between the Micro:bit, relay, sensors, and power source.

5. Battery Pack
   A container with batteries (e.g., AA or AAA) that stores and supplies electrical energy to power the Micro:bit, relay, and water pump when solar energy is not available. The battery pack provides a certain voltage (e.g., 4.5V or 6V) to power the system.

6. Solar Panels
   Devices that convert sunlight into electrical energy. They can be used to recharge the battery pack or directly power parts of the system for eco-friendly operation. Solar panels support sustainability by reducing the need for fossil fuels and lowering energy costs.

7. Soil Moisture Sensor
   A sensor that detects the amount of water in the soil and sends this data as an electrical signal to the Micro:bit. It measures humidity levels in the soil, usually expressed as a percentage.

8. Voltage
   The electrical force that drives current through a circuit, measured in volts (V). Each component in the watering system requires a specific voltage to function correctly.

9. Milliseconds
   A unit of time equal to one thousandth of a second (0.001 s). In the watering system, milliseconds describe how fast the Micro:bit can send control signals to the relay or how quickly sensors respond.

10. Humidity

The amount of water vapor in the air or soil, usually expressed as a percentage. In this system, soil humidity is detected by the soil moisture sensor to determine whether watering is needed.

11. Sustainability / Sustainable
    Practices that meet present needs without harming the environment or depleting resources for future generations. Using solar panels and efficient irrigation systems makes watering systems more sustainable.

12. Irrigation
    The artificial application of water to soil to assist plant growth. In this system, irrigation is automated based on soil humidity readings, saving water and improving plant health.

13. Power Connections
    Electrical paths that deliver energy from the battery pack or solar panels to the Micro:bit, relay, and pump. These are typically made using crocodile clips or wires.

14. Control Connections
    Wires that carry signals from the Micro:bit to the relay and from the soil moisture sensor to the Micro:bit. They allow the system to “decide” when watering is needed.

5.2 Ready-made watering systems:

https://makecode.microbit.org/projects/plant-watering

https://www.instructables.com/Automatic-Plant-Watering-System-Using-a-Microbit/

https://www.instructables.com/Microbit-Enabled-Plant-Watering-System/