# Urban Farming 101

## Introduction

In this activity, we will turn our MKR IoT Carrier into an urban farming device! We will use the carrier's sensors and a moisture sensor to analyze the environment for a plant, use artificial lighting and introduce relays – an electronic component used to activate high-power devices. We will be focusing on setting up a dashboard in Arduino IoT Cloud, where we can read data, and control different components.

## *Learning Objectives*

The goals of this activity are:

- Set up an urban farming environment
- Understand how relays work
- Understand how the moisture sensor works
- Create an ideal environment for a plant

## *Activity Complexity*

This activity requires having previous knowledge in:

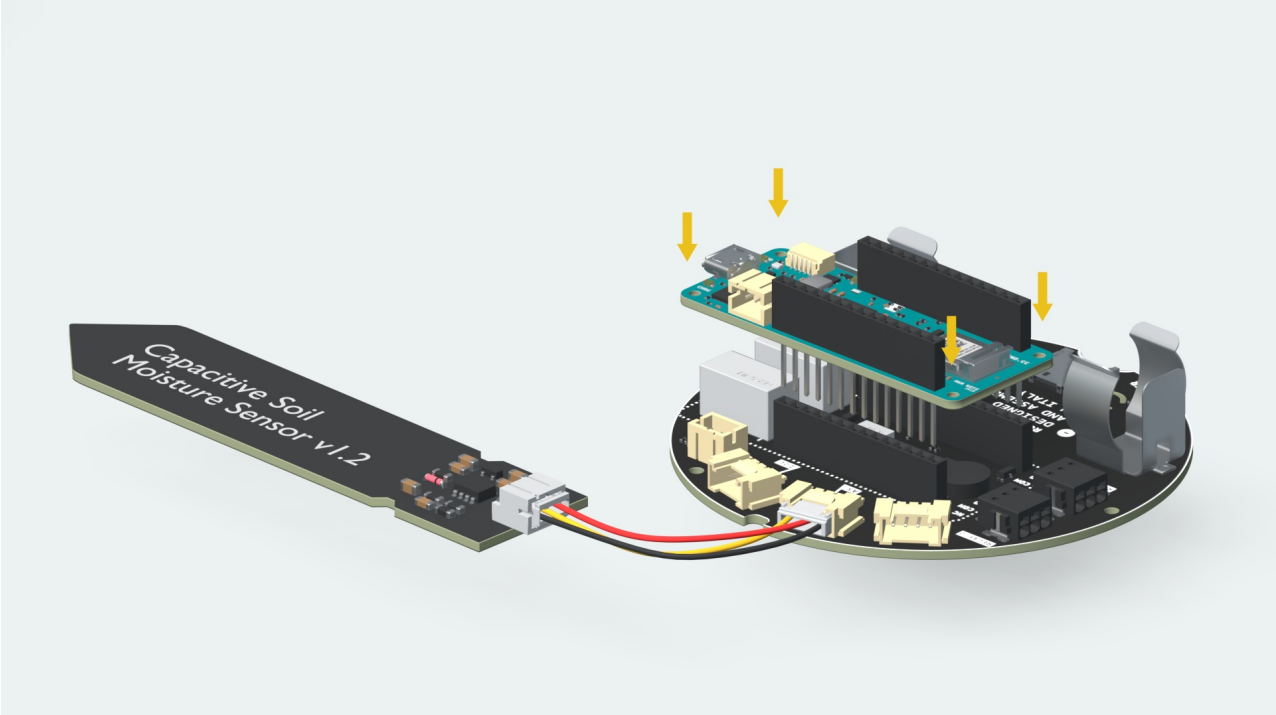
- Configuring Arduino IoT Cloud
- Working with the carrier's different sensors

## *Components Used*

- MKR WiFi 1010
- MKR IoT Carrier
- Moisture sensor
- Micro USB cable

## *Assembly*

First, we need to mount the MKR WiFi 1010 on top of the MKR IoT Carrier. We then need to plug the moisture sensor into the A5 slot on the carrier. Finally, we can connect the micro USB cable to a computer.



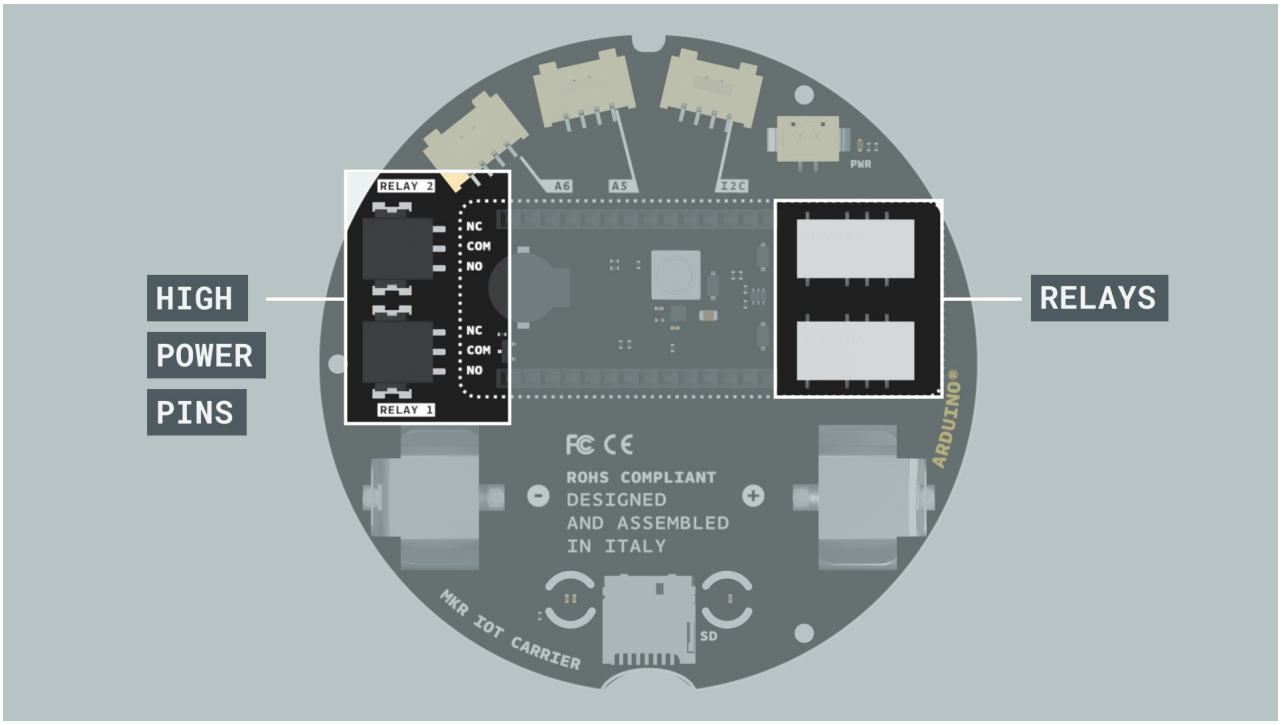

## What are relays

This activity will introduce a new component: the relay. A relay is a switch operating on electricity, in other words, relays are switches controlled electrically.

Relays allow using an Arduino board to control higher power circuits than the ones that the Arduino board is capable to control. They are typically used in industrial applications to control high power circuits, but it is also used in cars, homes and other electric applications. A very common example of the use of relays is the indicator light on cars. When the driver of the car activates the indicator light, we can hear a "tick-tack" sound, that sound is produced by a relay turning ON and OFF the light.

Relays are composed by an electromagnet that moves a tiny metallic plank, which is called **COM** terminal, between two different positions **NC** terminal and **NO** terminal. We can decide in which position the COM terminal is connected with by activating/deactivating the electromagnet by connecting a low power signal in the electromagnet control terminals.

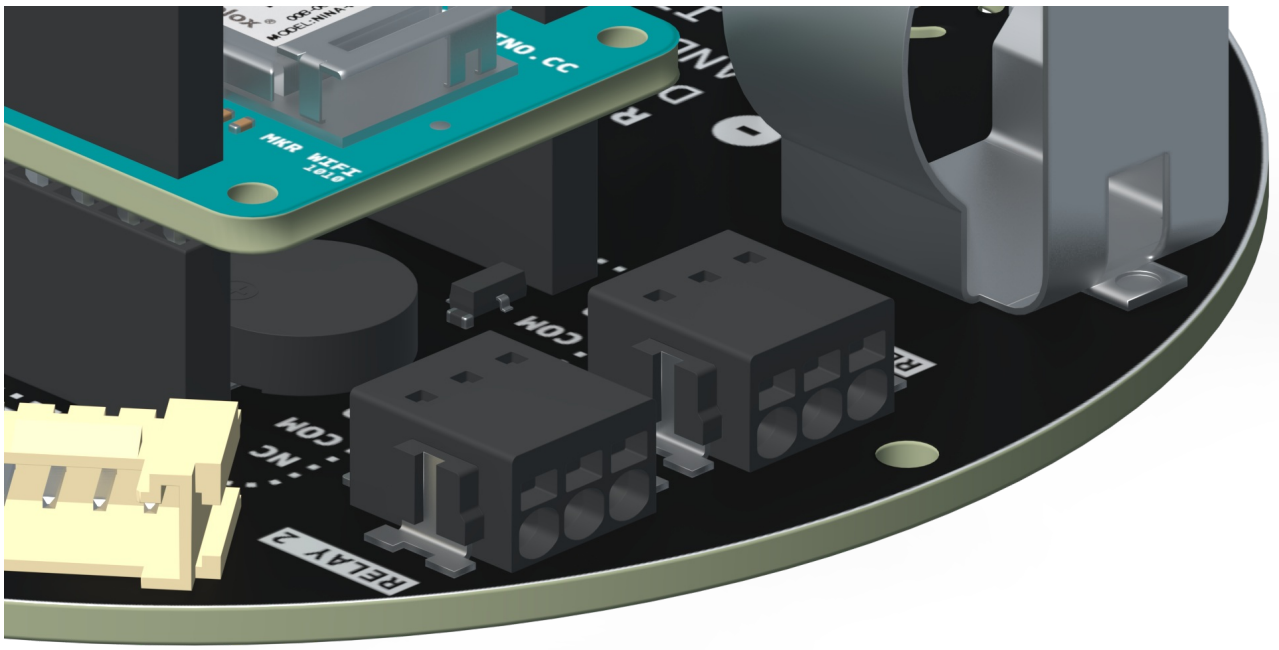
We will connect the high power signals that we want to control to the COM, NO or NC also called **high power pins**. Then the electromagnet control signals, called the **low power pins**, to the Arduino board. This is the way in which we will control high power signals by using our Arduino board. A relay separates the high power network from the low power network the Arduino is operating on, and ensures that these are never connected.




## *How to use the relays on the carrier*

As we can see in the image above, the MKR IoT Carrier has two relays capable of handling up to 24V each.

The connections between our high power circuti and the relays will be done through these connectors.



In them, the round hole at the bottom in the connector is the one that we will use to create the connections, once we introduce the cables inside them, those will be automatically locked keeping them in the connector.

When we want to remove the cable from the connector, we will need to introduce a tool inside the top square hole (it can be a flat screw driver, a hard piece of plastic, etc.), by doing that, we will unlock the cable and we will be able to remove it from the connector.

Now that we know how the realys work and how to make the connections on them, let's take a look at the definitions of the high power pins:

- **NO** - Stands for **normally open**. This means that when we write a **HIGH** state to the relay, **NO** pin is connected with **COM**.
- **NC** - Stands for **normally closed**. This means that when we write a **LOW** state to the relay, when the relay is not supplied, **NC** pin is connected with **COM**.
- **COM** - Stands for **common** and is used as the switch in a relay. When a HIGH or LOW signal is written to the relay, the common pin will be between either **NO** or **NC** depending on what configuration we are using.

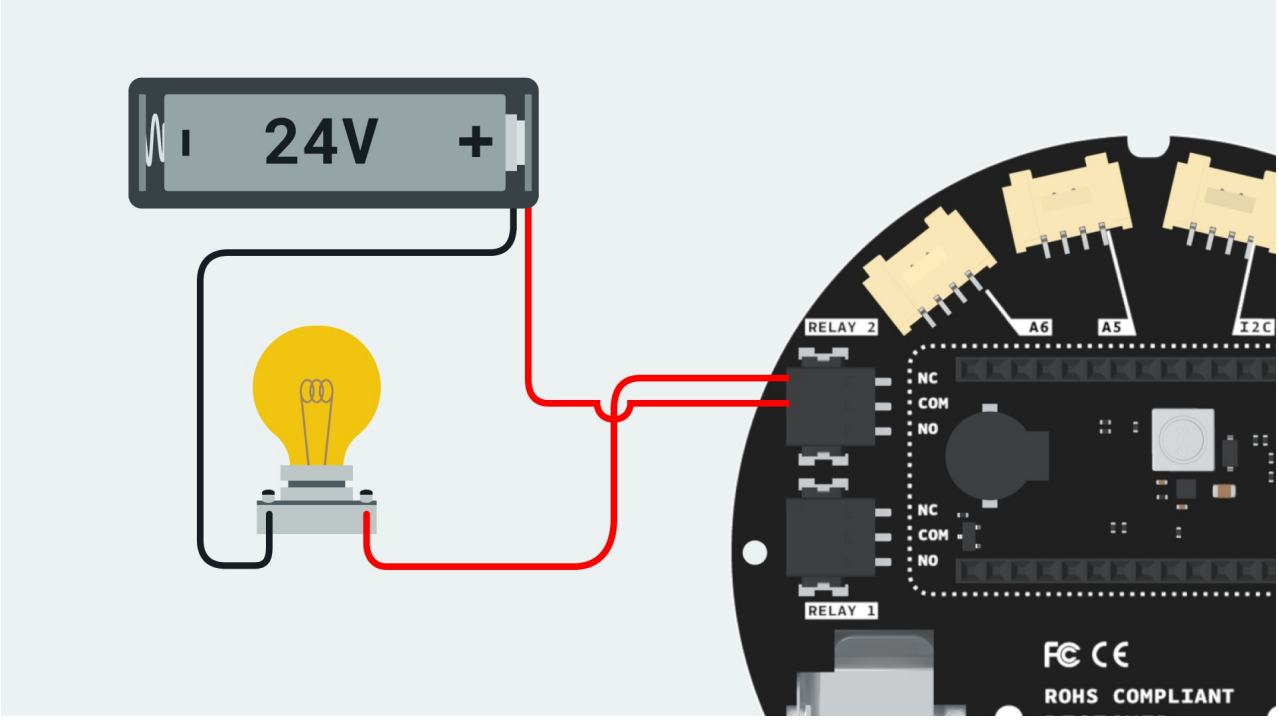
The L1 and L2 LEDs on the carrier allow us to know in a visual way what is the state of the relays, if the LED is ON, it means that the the COM and the NC terminal of the relay are connected and if the LED is OFF it means that COM and NO are connected.

We do not have to consider the low power pins, since they are already connected to the Arduino board through the MKR IoT Carrier.

## *How to set up a NC or NO circuit*

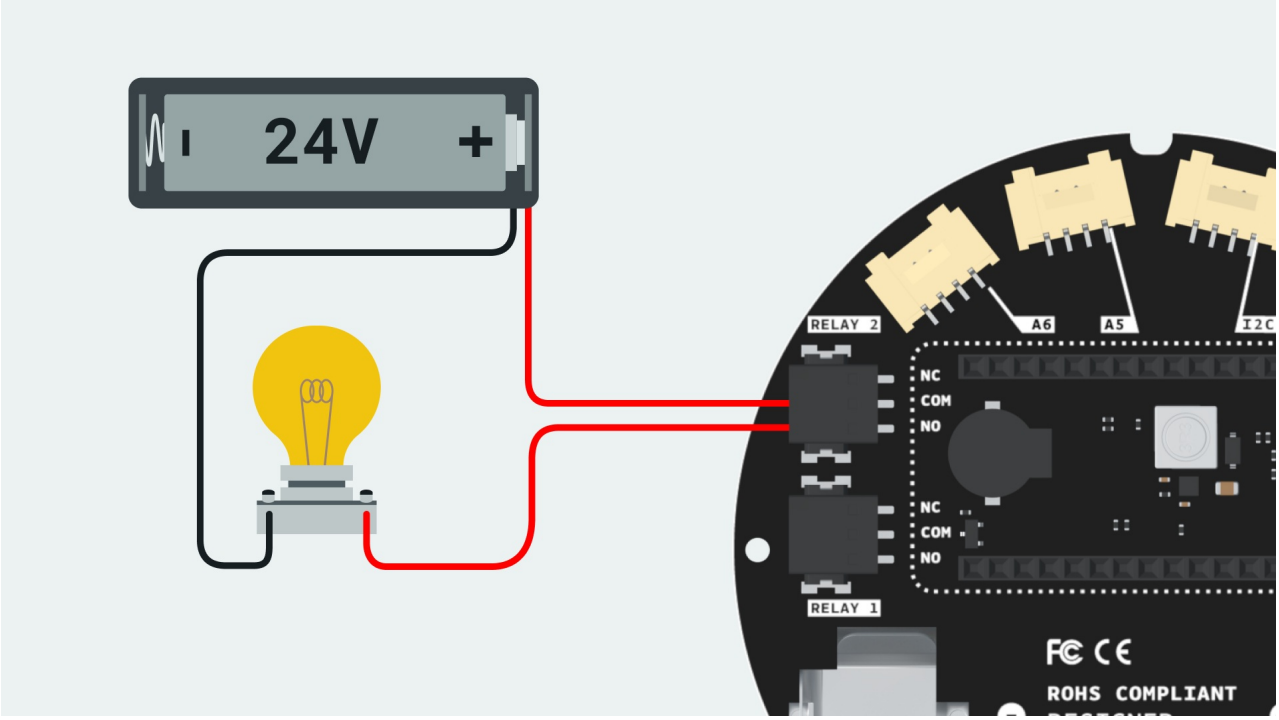
The relays we are using have a NO (normally open) configuration, meaning that the NO and COM terminals are connected by default and when we put a high level on the low power terminals we will connect the NC and COM terminals.

The following image shows how we will need to make the connections of the higher power source and components to the relay.




There are as well relays that use NC configuration, meaning that the NC and COM terminals are connected by default and when we put a high level on the low power terminals we will connect the NO and COM terminals.

If we use a normally closed (NC) configuration, the following image shows how the higher power source and components should be connected to the relay.




> **Important!** In this activity, we will only be testing out the functionality of the relay and how we can control it. The relays on the shield only support up to **24V**, meaning that they are not intended to work with higher voltages than that. Like mentioned before, a standard power outlet is typically around 220-240V, which is 10 times as much as the relay is designed for, and can cause immediate danger.

## Moisture sensor

The moisture sensor is one of the most basic, yet powerful, sensors out there. It has two exposed conductors and is basically a variable resistor. When it is exposed to water, the resistance drops, as the conductivity is increased, while less water means higher resistance.

If we place this sensor in a plant pot that has recently been watered, the resistance is lower, and if we do a reading on it, we can measure how moist the plant is. But if we place the sensor into a plant pot that has not been watered in days, it is most likely going to show 1023 (completely dry). This gives us immediate feedback: we need to water our plants, otherwise they will not survive!

This is an incredibly useful sensor for urban farming projects, and there are thousands of cool inventions that have been made using moisture sensors all over the world!

## Configuring the IoT cloud

In this activity, we will be using Ardunio IoT Cloud to both monitor sensors and control different actuators on the carrier. As we have done in the previous activities, we will first need to set up some properties in the cloud. Let's navigate back to Arduino IoT Cloud.

We will start by creating a new thing in Arduino IoT Cloud, link the thing to our board, and then we will create the properties that is shown in the table below.

| Name | Data Type | Value Range | Function | Permission |
|---|---|---|---|---|
| humidity | float | 0 - 100 | Display humidity | Read only |
| temperature | float | -40 - 100 | Display temperature | Read only |
| light | int | 0 - 5000 | Measure illuminance | Read only |
| moistValue | int | 0 - 100 | Display moist level | Read only |
| relay_1 | Boolean ON/OFF | n/a | Turn on or off relay 1 | Read & write |
| relay_2 | Boolean ON/OFF | n/a | Turn on or off relay 2 | Read & write |
| rgbColor | colored light | n/a | Control the color of the RGBs on the carrier | Read & write |

| Name | Data Type | Value Range | Function | Permission |
|---|---|---|---|---|
| updateDisplay | Boolean ON/OFF | n/a | Refresh the display on the carrier | Read & write |

When creating the properties with the permission **Read only**, we also need to change it to update every second. For the **Read & write** properties, we need to set it to update whenever the value changes. This ensures that sensor readings only happen every 1 second, providing more consistent data.

When we are done creating the properties, we can simply click **Edit sketch** and a code will be automatically generated.

## Gradually building code

Now it's time to complete the code! We will begin by including the library to control the MKR IoT Carrier. We will also set `CARRIER_CASE` to `false`. If we are going to use the plastic encasing, we can switch this variable to `true`.

Here we will also define the pin that we will connect the **moisture sensor** to, `A5`, followed by two empty strings. These strings will be updated with the state of its corresponding relays, but we will go through that later on.

In the `void setup()`, we will only add the commands `while(!Serial);` and `carrier.begin();` which allows us to initialize the carrier library by opening the Serial Monitor.

In the `void loop()`, there are two main things that happen: we create an `if/else` statement for the relays and we read each sensor in the project.

The `if/else` statements are quite simple. If we activate `relay_1` from the cloud dashboard, first the relay turns **ON**, and then changes the `relayState1` string to **ON**. If we turn it **OFF**, the `relay` turns off and changes `relayState1` to **OFF**. The exact same thing happens with `relay_2`, but it controls the second relay instead.

Then, we do readings to get the current `light`, `temperature` and `humidity`. But for this activity, we will also use the moisture sensor. When we are reading an analog component, we will get a value between 0-1023. This value range does not make much sense, so let's simplify it a little bit by using the `map()` function.

We can use this command:

```
moistValue = map(rawMoistValue, 0, 1023, 100, 0);
```

This basically changes the value range from 0 - 1023 to 100 - 0. Now, if a plant is really dry, the value will be really low, and if the plant is really moist, the value will be very high. This way, we can measure from a scale of 0 - 100 instead of 1023 - 0.

After we have finished `void loop()`, let's take a look at the functions that were generated from the Arduino IoT cloud dashboard. The first two – `onRelay1Change()` and `onRelay2Change()` – will be left empty since we already set these up in the loop. The `onRgbColorChange()` will use the same code we used in Activity 4, Remote Triggers to control the RGBs on the carrier from the cloud dashboard.

The final function of this program is to remotely update the display on the carrier with the values from the sensors and the state of the relays. These are printed in different colors on the same screen. This function is triggered from the cloud dashboard. At the end of this function, we use the command `updateDisplay = false;` which makes sure that the Boolean is reset in the dashboard.

Once we are finished with the code, we need to upload it to our board. Once that is done, we need to open the Serial Monitor to initialize the carrier library. After it has made a connection to the cloud, we can navigate to the Arduino IoT cloud dashboard.

## Testing it out

Great job! We have now created a pretty sophisticated application to measure and control different aspects of an urban farming environment. The last step before we can test it out is to create a dashboard for our activity. Navigate back to the cloud, and click on the **Dashboards** button, and create a new dashboard. We can name it something simple, such as "Activity 8".

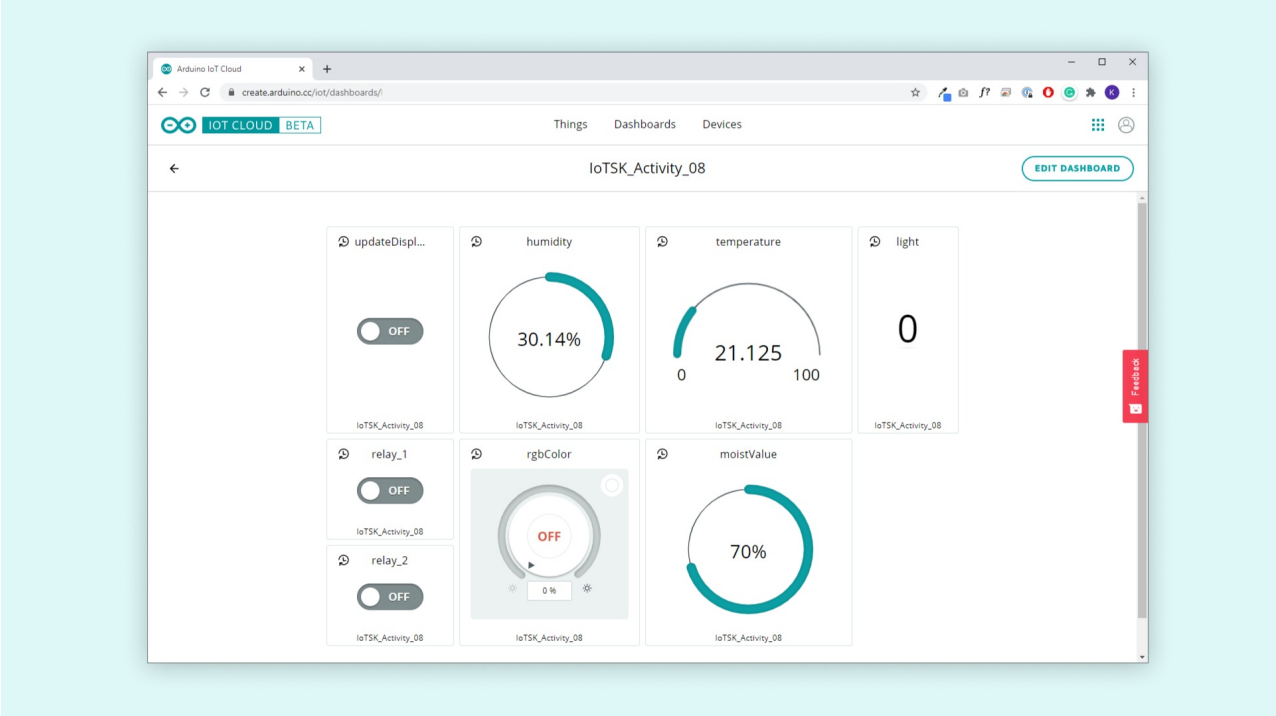
We will need to create eight widgets in total, and we can use the table below to set them up and link them to our properties properly:

| Property | Widget | Value Range |
|---|---|---|
| humidity | Percentage | 0 - 100 |
| temperature | Gauge | -40 - 100 |
| light | Value | 0 - 5000 |
| moistValue | Percentage | 0 - 100 |

| Property | Widget | Value Range |
|---|---|---|
| relay_1 | Switch | n/a |
| relay_2 | Switch | n/a |
| rgbColor | Colored Light | n/a |
| updateDisplay | Switch | n/a |

In this dashboard, we have four widgets used to monitor the sensors, and four properties used to control different actuators. Now we can place the MKR IoT Carrier in the plastic casing, and place it close to a plant that we want to monitor! In this activity, we will be using the aloe vera plant as an example.

Once we have set up the dashboard, it should look similar to the image below, but we can organize the dashboard however we want to!




So let's see what the data we record from the sensors can tell us, and what actions we can use to control the environment. But first, let's pick a popular plant and see what conditions that plant thrives in best!

## *Creating an ideal environment*

In this activity, we used an aloe vera plant, which is a cheap and durable plant that can be found in homes all over the world. But the plant has some ideal conditions; let's see how we can set that up. The table below describes the ideal conditions for the aloe vera plant.

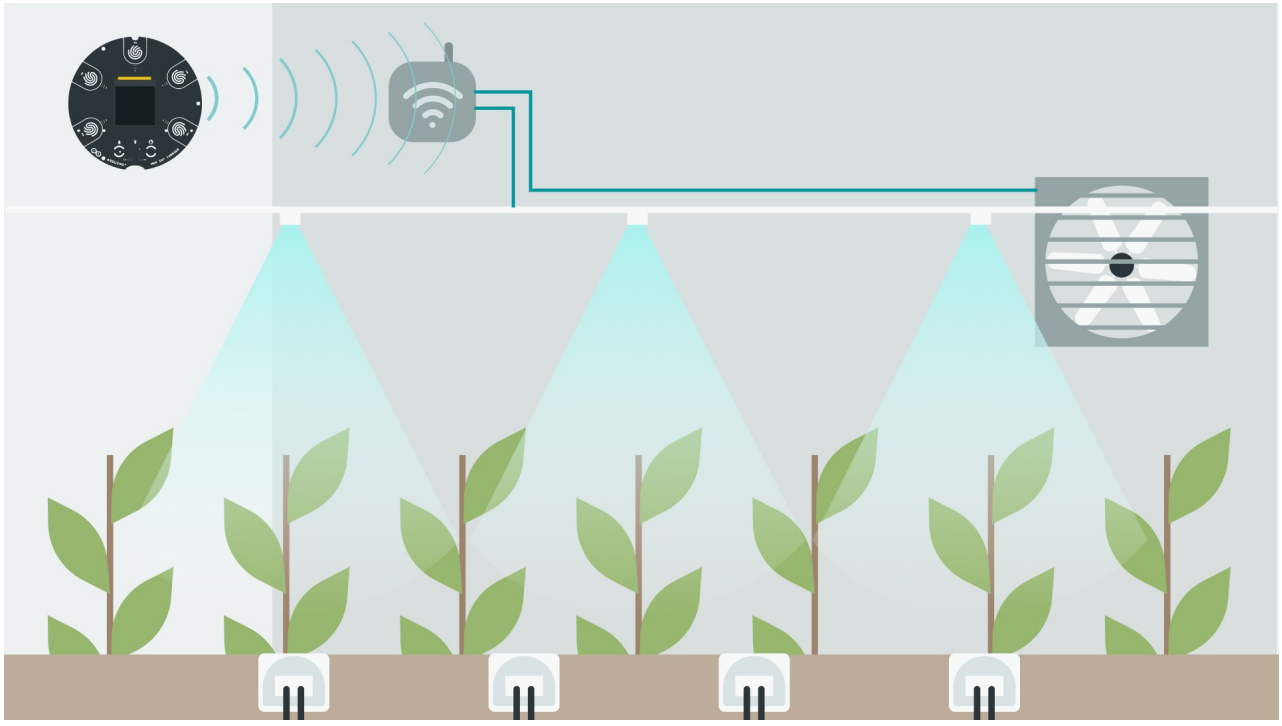
| Relative humidity | Temperature | Sun light | Moisture |
|---|---|---|---|
| 40% | 15 - 24 Celsius (60-75 Fahrenheit) | 6-8 hours a day | Low - Medium |

Using this plant as a reference, let's see how we can improve its environment.

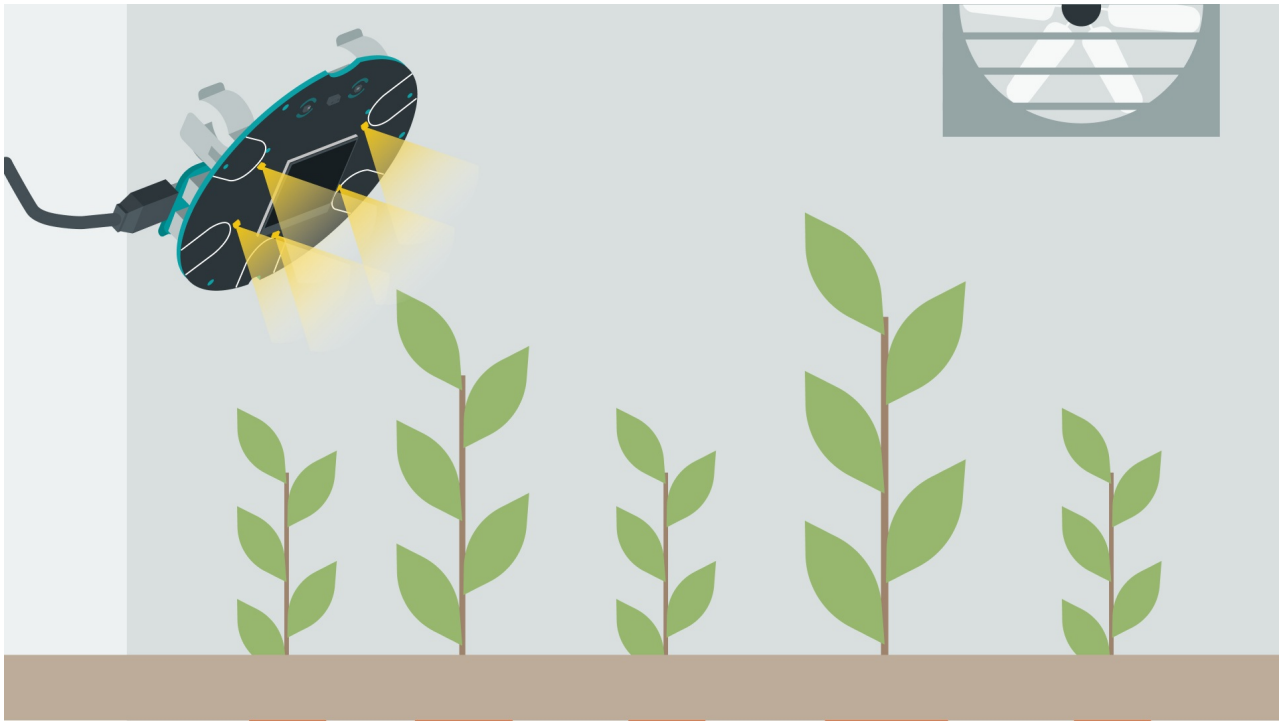
- **Humidity & temperature:** If we are above the 40% relative humidity mark, we need to take action. This can be done by either placing a fan close to it, placing it close to an air conditioner or close to an open window. This will also fix the problem of high temperature as these two measurements are closely connected to one another.
- **Moisture:** To measure the moisture of the plant, we can gently place the moisture sensor in the plant pot. We want to keep this value relatively close to 50%. This means that we should not water the plant too much, but not too little – just a little bit every so often!
- **Sunlight:** The plant does need sunlight, but can survive a bit in the shade. The recommended sun dosage per day is around 6-8 hours. If we place the MKR IoT Carrier next to the plant, the light sensor gives us information on whether the spot receives enough sunlight. The value that is recorded by the light sensor is recorded in LUX, and this should normally be over 200 (this measurement was taken from placing it in a window during sun hours).

All of the above can be measured with the sensors in the carrier, but we can also be creative on how we can use the actuators on the carrier:

> **Relays:** While there are no high power sources or components included in the kit, we can start getting creative on how to use them! There are two popular components that are typically controlled by a relay in an urban farming setting: **water pumps** and **fans**. If the moisture is low, we can remotely control a pump that is connected to a water source, and pump water into the plant! If the temperature and humidity is too high, we can enable a fan that brings cool air towards the plant.



**RGBs:** The RGBs on the carrier can be used as artificial lighting if angled correctly. This method is particularly good to use during long dark months, where sunlight is rare. If we activate the RGBs, and place them above the plant, we can simulate a UV lamp. The RGBs are of course not very powerful, and there are much more powerful lamps that we can use for this purpose, but then we need to use the relays.



## Challenge

In this activity, we have created an urban farming setup that can be remotely controlled and monitored. But what if we wanted to add a bit of personality to the project?

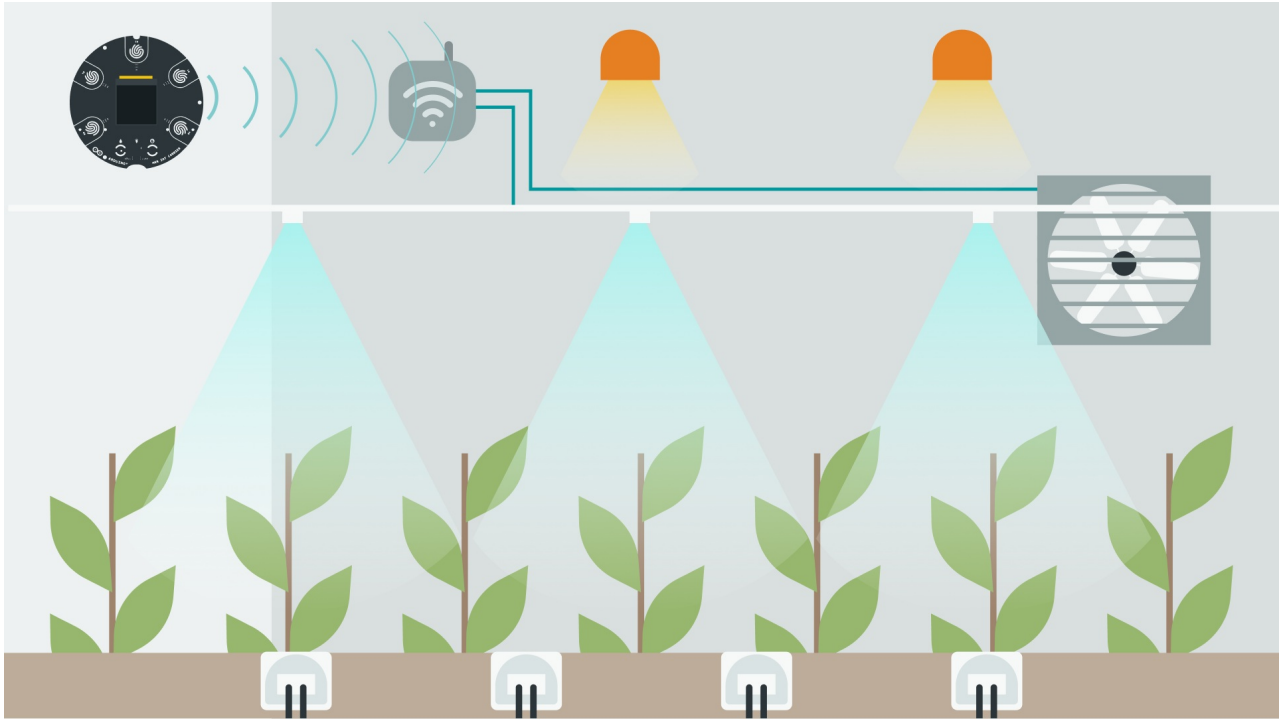
Earlier in this activity, we worked with both Strings and if/else conditionals. The challenge of this activity, is to:

- Create a property named "plant_feeling" in Arduino IoT Cloud of the String data type.
- Create a conditional statement that can express the plant's "emotions" in words. For example, if the humidity or temperature is too high, or the moisture is too low, the plant is "unhappy".
- Read the emotion of the plant in the Arduino IoT Cloud dashboard.

## Wrapping up

In this activity, we built a practical urban farming application. This setup allows us to measure not only the environment of a plant but also how we can remotely control actuators to change the environment. We learned about what relays are, what they can be used for, and how to write a program to control them. We also learned how to use the moisture sensor, an essential sensor to use for urban farming projects, and how to map the sensor values to a range that makes more sense!

By using Arduino IoT Cloud and the MKR IoT Carrier, we were able to test several aspects of an urban farming environment, but we can take things further than that! We can, for example, build a greenhouse that has artificial lighting, a cooling fan and a water pump. These three can create their own little ecosystem for plants to thrive in, and we can set it up so we don't have to physically interact with it by controlling it remotely from the cloud, or just automate the entire process!



To finish up this activity, let's put back the components we have used in this activity inside the kit box, and log out of the Arduino Create / Arduino IoT Cloud environment.