

## Lesson 8 Smart Agriculture Visualization System

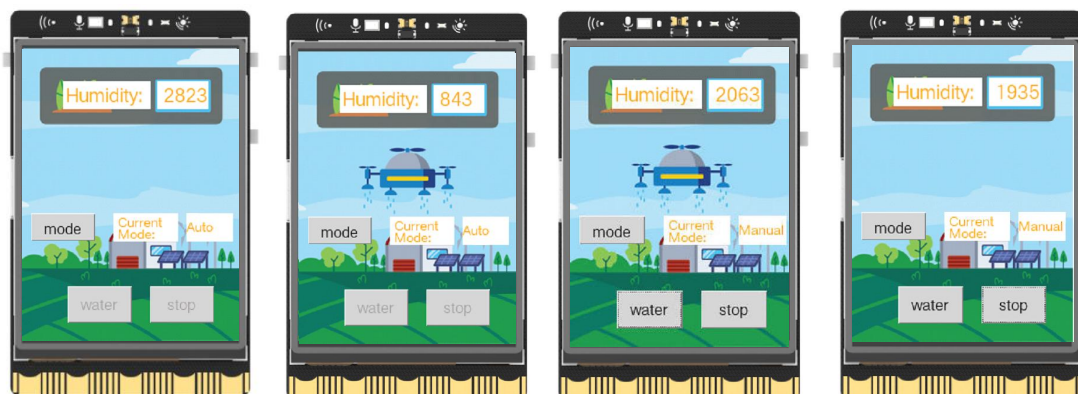
Nowadays, with the continuous advancement of modern technology, the level of agricultural automation is constantly improving. An increasing number of electronic devices such as sensors and displays are being applied in agriculture to monitor crop growth anytime and anywhere, enabling improvements to be made when the environmental conditions are unfavorable.

In this lesson, let's combine UNIHAKER to design a smart agriculture visualization system and simulate the monitoring and improvement of crop growth environments in modern agriculture!



### Task Objectives

Display the soil moisture value detected by the soil moisture sensor on the UNIHAKER. Additionally, set up two modes: automatic and manual. In the automatic mode, when the detected soil moisture value is too low, water can be automatically irrigated using an external relay and a water pump. At the same time, a watering icon appears on the UNIHAKER. In the manual mode, watering and turning off the water can be done by clicking buttons on the screen. Clicking the "Switch Mode" button allows switching between manual and automatic modes.


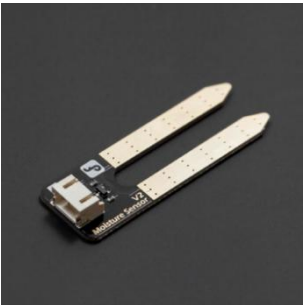




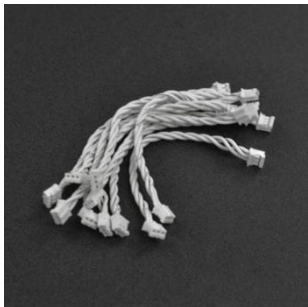


### Knowledge points

1. Understand the soil moisture sensor, relay, and water pump.
2. Learn how to use the unihiker library to display buttons.
3. Learn how to use the Pinpong library to retrieve data from the soil moisture sensor.
4. Learn how to use the Pinpong library to control the relay.
5. Learn how to use the relay to control the water pump for irrigation.

## Material List

### Hardware List:

		
UNIHAKER x1	Soil Moisture Sensor x1	Easy Relay Module x1
		
Type-C & Micro Dual-Use USB Cable x1	Wall Adapter Power Supply 12VDC 1A x1	Immersible Pump & WaterTube x1
		
PH2.0-3P white silicone twisted wire at both ends x2		

**Software Preparation:** Mind+ Programming Softwarex1

### Others:

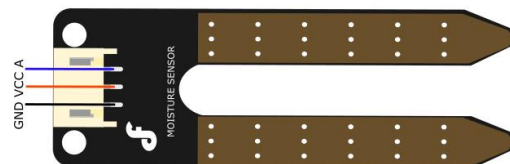
1. Plant pot with a plant x1
2. Beaker with water x1
3. Cross/flathead screwdriver x1

## Knowledge background

### 1. Soil Moisture Sensor

This is a simple moisture sensor used to detect the moisture content in soil. When the soil is dry, the sensor output value decreases, and vice versa. The surface of the sensor is gold-plated, which extends its lifespan.

Number	Name	Description
1	GND	GND(Black wire)
2	VCC	Power(Red wire)
3	A	Analog output(Blue wire)



To use the soil moisture sensor, insert the two metal probes into the soil fully and wait for a few seconds until the data stabilizes. The corresponding value of soil moisture can be read through a display module or software.

**Tips 1:** Measurement values may vary depending on different soil mediums. Moisture distribution in the soil can be uneven, so the data represents local humidity only.

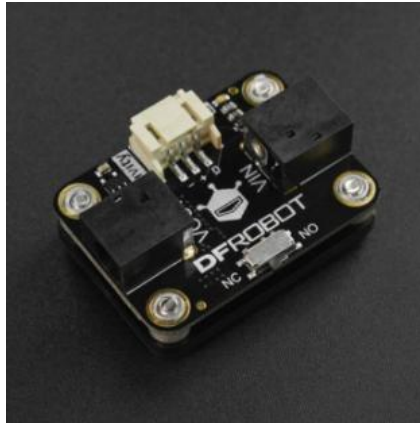
**Tips 2:** The plastic part on the top of the sensor is not waterproof, so please avoid burying the entire sensor in the soil.

### 2. Relay

A relay is an automatic switch component that uses a small current to control the operation of a larger current. It also provides isolation functionality.

Although the supply voltage of the UNIHAKER board is 5V, which is higher than the minimum operating voltage of the water pump (4.5V), directly driving the pump with the board may result in unstable operation. Therefore, in this case, we will use a relay module and an external power source to drive the water pump.

**Tips:** The working principle of a relay can be referred to in further reading.



### 3. Water Pump

A water pump is a mechanical device used for conveying liquids or increasing liquid pressure. It can be directly placed in water and has a wide voltage supply range of 4.5 to 12V. It can be used for watering plants or even for changing the water in an aquarium.



### 4. The add\_button() Method of the UNIHAKER Library's GUI Class for Displaying Buttons

The add\_button() method in the GUI class allows you to display a button widget on the UNIHAKER screen. In programming, you can achieve this functionality by using the syntax "object.method\_name()". Additionally, this method returns a control object, which can be stored in a variable for convenient future object update operations.

```
from unihiker import GUI # Importing the GUI module from the unihiker library
gui = GUI() # Instantiating the gui object
# Defining callback functions
def click_A(): # Define the operation when button A is clicked - image switch
    img.config(w=240, h=320, image='img/water1.png')
```

```
button_A = gui.add_button(x=50, y=260, w=70, h=40, text="water", onclick=click_A,  
state='disabled')
```

Among them, the parameters 'x' and 'y' represent the horizontal and vertical coordinates of the button's position, 'w' and 'h' represent the width and height of the button, 'text' refers to the text displayed on the button. 'onclick' represents the functionality triggered when the button is clicked. Here, we need to set the functionality as a callback function beforehand. 'state' indicates the state of the button, which can have two values: 'normal' represents the normal clickable state, and 'disabled' represents the disabled state when the button appears grayed out and cannot be clicked.

#### 5. The write\_digital() Method of the Pin Class in the Pinpong Library for Digital Output

The write\_digital() method in the Pin class of the Pinpong library allows for digital output. Before using it, we need to import the Pinpong library, initialize the UNIHAKER, and instantiate the Pin class to create a pin object.

```
from pinpong.board import Board, Pin # Import the Board and Pin modules from the pinpong  
library  
Board().begin() # Initialize the UniHiker board  
relay = Pin(Pin.P23, Pin.OUT) # Initialize pin 23 as digital output mode  
relay.write_digital(1) # Set the relay pin to high level, turn on the relay
```

Among them, Pin and Board are similar and both are modules under the pinpong.board package. "Pin.P23" refers to the 23rd pin where the relay is located on the board. "(Pin.P23, Pin.OUT)" is used to define the P23 pin as a digital output mode. It is then passed to the Pin class for instantiation, resulting in a pin object stored in the variable relay. Finally, the write\_digital() method is used to output a specific level to the pin. "1" represents a high level, while "0" represents a low level.

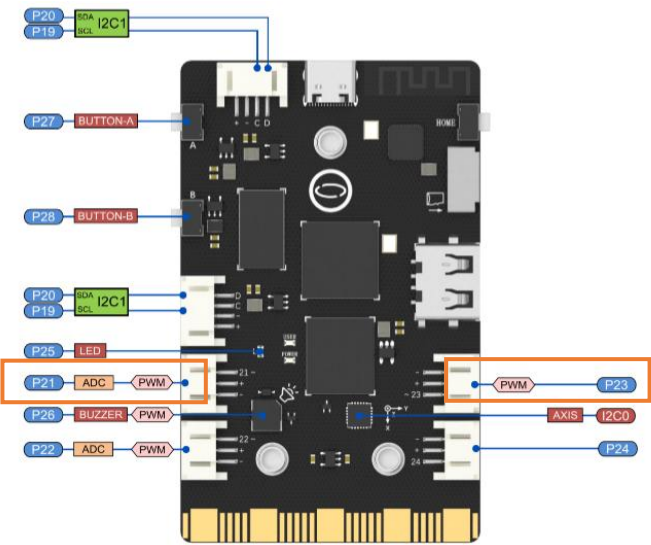
#### 6. The read\_analog() Method of the Pin Class in the Pinpong Library for Analog Input

The read\_analog() method in the Pin class of the Pinpong library allows for analog input. Before using it, we need to import the Pinpong library, initialize the UNIHAKER board, and instantiate the Pin class to create a pin object.

```
from pinpong.board import Board, Pin # Import the Board and Pin modules from the pinpong  
library  
Board().begin() # Initialize the UniHiker board  
adc0 = Pin(Pin.P21, Pin.ANALOG) # Initialize pin 21 as analog input mode  
Soil_moisture_value = adc0.read_analog() # Read analog value
```

Among them, Pin and Board are similar and both are modules under the pinpong.board package. "Pin.P21" refers to the 21st pin where the soil moisture sensor is located on the board. "(Pin.P21, Pin.ANALOG)" is used to set the P21 pin as an analog input mode. It is then passed to the Pin

class for instantiation, resulting in a pin object stored in the variable `adc0`. Finally, the `read_analog()` method is used to detect and read the analog value, which is stored in the variable `Soil_moisture_value`.



Pin Function Diagram

**Tips:** Not all modes are supported by every pin on the UNIIKER. For specific information, please refer to the table below.

Hardware Pin	Function
ADC	P0 P1 P2 P3 P4 P10 P21 P22
PWM	P0 P2 P3 P10 P16 P21 P22 P23
Digital Input、 Digital Output	All Pins Configurable

Hands-on practice

Task Description 1: Displaying a Fixed Moisture Value

Add a smart agriculture background image on the screen and display an initial set humidity value of 50. Also, add two buttons to simulate the functionality of watering control: pressing the "Water" button will switch to a watering background image, and pressing the "Stop Water" button will switch back to the original background image.

1. Hardware setup

Connect the UNIIKER to the computer via a USB cable.



## 2. program coding

### STEP1: Creating and Saving Project Files

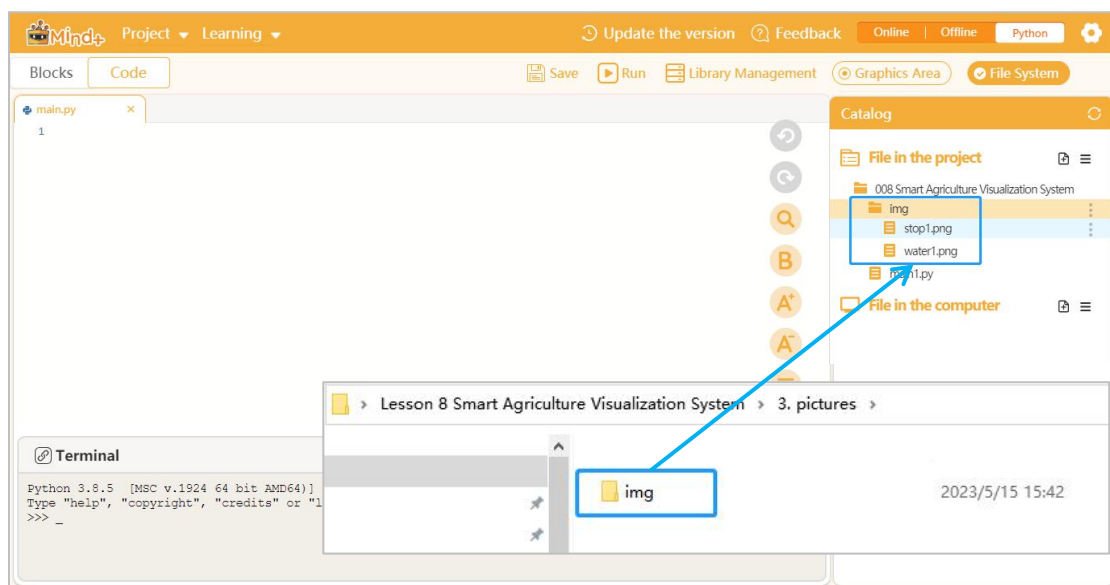
Launch Mind+ and save the project as "008 Smart Agriculture Visualization System".

### STEP2: Creating and Saving Python Files

Create a Python program file named "main1.py" and double-click to open it.

### STEP3: Import image folder

From the 'pictures' folder, drag and drop the 'img' folders into this area.



### STEP 4: Programming

#### (1) Importing the Required Libraries

In this task, we will be using the GUI module from the unihiker library to display text on the screen. Therefore, we need to import it first. Additionally, since we will need to introduce some delays during the program execution, we also need to import the time library.

```
from unihiker import GUI # Importing the GUI module from the unihiker library
import time # Importing the time library
```

#### (2) Instantiate the GUI Class and Display the Initial Background Image

Before using the GUI module from the unihiker library, we need to instantiate the GUI class to create an object, which allows us to utilize various methods within the class.

Afterward, we display an initial background image on the screen.

```
gui = GUI() # Instantiating the gui object
```

```
# Displaying the background image
```

```
img = gui.draw_image(w=240, h=320, image='img/stop1.png')
```

### (3) Define the Functionality of Button Clicks

Since we will be adding two buttons on the screen and using them to trigger the watering and stopping watering functionalities, we need to define two functions. In the function "click\_A()", we set the functionality to switch to the watering image, and in the function "click\_B()", we set the functionality to switch to the stopping watering image. These functions will be called later in the program.

```
# Defining callback functions
```

```
def click_A(): # Define the operation when button A is clicked - image switch
```

```
    img.config(w=240, h=320, image='img/water1.png')
```

```
def click_B(): # Define the operation when button B is clicked - image switch
```

```
    img.config(w=240, h=320, image='img/stop1.png')
```

### (4) Define the Button Display Parameters

Next, we can display two buttons on the UNIIKER, representing "Water" and "Stop Water," and set the functionality to be triggered when the buttons are clicked.



```
# Display buttons
```

```
'''Display buttons and set their functions triggered by clicking'''
```

```
button_A = gui.add_button(x=50, y=260, w=70, h=40, text="water", onclick=click_A)
```

```
button_B = gui.add_button(x=140, y=260, w=70, h=40, text="stop", onclick=click_B)
```

### (4) Display the Soil Moisture Value

we will draw two filled rectangles on the screen and display the text "Humidity Value" along with the given soil moisture value "50".

```
# Draw filled rectangles and display text inside them
```

```
gui.fill_rect(x=45, y=35, w=95, h=30, color="white")
```

```
gui.fill_rect(x=148, y=35, w=55, h=30, color="white")
```



```
gui.draw_text(x=48, y=36, color="orange", text='value:')  
  
text = gui.draw_text(x=160, y=36, color="orange", text="50") # Display initial humidity value:  
50
```

#### (5) Display Emojis

Finally, to ensure that the screen content remains displayed for a long time, we keep the program running.

```
while True: # loop  
    time.sleep(1) # Wait for 1 second
```

**Tips:** The complete example program is as follows:

```
from unihiker import GUI # Importing the GUI module from the unihiker library  
import time # Importing the time library  
  
gui = GUI() # Instantiating the gui object  
  
# Displaying the background image  
img = gui.draw_image(w=240, h=320, image='img/stop1.png')  
  
# Defining callback functions  
def click_A(): # Define the operation when button A is clicked - image switch  
    img.config(w=240, h=320, image='img/water1.png')  
  
def click_B(): # Define the operation when button B is clicked - image switch  
    img.config(w=240, h=320, image='img/stop1.png')  
  
# Display buttons  
"""Display buttons and set their functions triggered by clicking"""  
button_A = gui.add_button(x=50, y=260, w=70, h=40, text="water", onclick=click_A)  
button_B = gui.add_button(x=140, y=260, w=70, h=40, text="stop", onclick=click_B)  
  
# Draw filled rectangles and display text inside them  
gui.fill_rect(x=45, y=35, w=95, h=30, color="white")  
gui.fill_rect(x=148, y=35, w=55, h=30, color="white")  
  
gui.draw_text(x=48, y=36, color="orange", text='humidity:')  
  
text = gui.draw_text(x=160, y=36, color="orange", text="50") # Display initial humidity value:
```

50

```
while True: # loop
    time.sleep(1) # Wait for 1 second
```

### 3. Running the Program

**STEP 1:** Remote connection to UNIIKER

**STEP 2:** Observe the Effect

Observing the UNIIKER, you can see that on the smart agriculture background image, a fixed soil moisture value of 50 is displayed, and there are two buttons below. When you click the "Water" button, a sprinkler drone appears, simulating an actual watering scene. When you click the "Stop Water" button, the drone disappears, and watering is stopped.



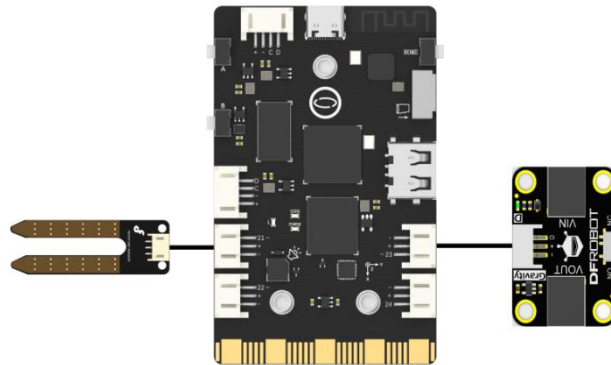
### Task Description 2: Real-time Humidity Detection and Mode

#### Switching

By using an external soil moisture sensor, we will continuously monitor the humidity values and display them on the screen. Additionally, we will set two modes: automatic and manual. In the automatic mode, when the detected soil moisture value is too low, we will automatically water the plants using an external relay and water pump. In the manual mode, watering can be initiated by clicking the buttons on the screen. It will also be possible to manually switch between the modes.

## 1. Hardware setup

**STEP1:** Connect the soil moisture sensor to the P21 pin of the UNIHAIKER, and connect the relay to the P23 pin of the UNIHAIKER.



**STEP2:** Use a screwdriver to connect the positive and negative wires of the water pump to the adapter. Follow the steps and illustration below:

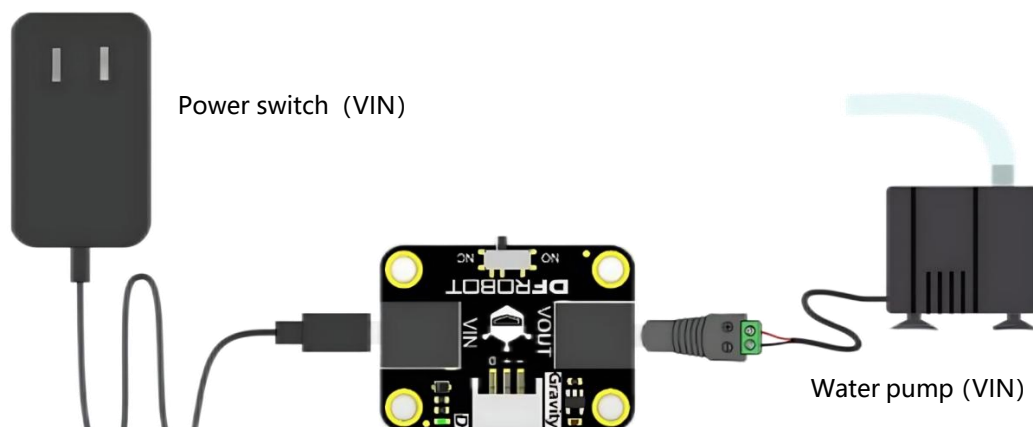
- (1) Loosen the screws on the adapter.
- (2) Insert the wires, paying attention to the polarity (brown/red - positive; blue/black - negative).
- (3) Tighten the screws.

The red/brown wire is connected to the positive terminal.

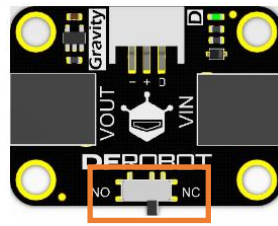


The blue/black wire is connected to the negative terminal.

**STEP3:** Connect the 12V power switch to the water pump's adapter using a relay.



**STEP4:** Set the relay switch to the NC (Normally Closed) terminal.



Switch to the NC (Normally Closed) terminal.

**STEP5:** Secure the water pump in a filled beaker

**Tips:** The water pump should not run dry. Make sure to immerse the black pump head in water to prevent potential hardware damage.



**STEP6:** Insert the water pipe and soil moisture sensor into the flowerpot.



## 2. program coding

### STEP1: Creating and Saving Python Files

Create a Python program file named "main2.py" and double-click to open it.

### STEP 2: Programming

(1) Import the Pinpong library and initialize the UNIHAKER and pins.

Here, we will be using an external soil moisture sensor and relay, so we need to import the Board and Pin modules from the Pinpong library and initialize the UNIHAKER and pins.

```
from pinpong.board import Board, Pin # Import the Board and Pin modules from the pinpong library
Board().begin() # Initialize the UniHiker board
adc0 = Pin(Pin.P21, Pin.ANALOG) # Initialize pin 21 as analog input mode
relay = Pin(Pin.P23, Pin.OUT) # Initialize pin 23 as digital output mode
```

(2) Update the functionality of button clicks.

Next, in the callback functions for buttons A and B, we need to add the relay output to enable watering when button A is clicked by setting the relay output to a high level, and to stop watering when button B is clicked by setting the relay output to a low level.

```
# Defining callback functions
def click_A(): # Define the operation when button A is clicked - image switch
    img.config(w=240, h=320, image='img/water1.png')
    relay.write_digital(1) # Set the relay pin to high level, turn on the relay

def click_B(): # Define the operation when button B is clicked - image switch
    img.config(w=240, h=320, image='img/stop1.png')
    relay.write_digital(0) # Set the relay pin to high level, turn off the relay
```

(3) Define the functionality of the mode switch button.

Since we will be adding a button on the screen to switch between modes, we need to define a callback function for that button. Here, we first define a flag for the automatic mode and use 1 and 0 to represent its on and off states. In the callback function, we check the current mode. If the current mode is manual, then when the "Switch Mode" button is clicked, the mode is updated to automatic. At this point, buttons A and B are disabled, and the flag is set to 1. Conversely, if the current mode is automatic, clicking the "Switch Mode" button updates the mode to manual, buttons A and B return to normal, and the flag is set to 0.

```
def click_C(): # Define the operation when button C is clicked -- mode switching
```

```
global is_auto_mode
if is_auto_mode == 0: # If the auto mode flag is 0, indicating manual mode
    text_mode.config(text="Auto")
```

(4) Display filled rectangles and text

Next, we will display two filled rectangles and text on the screen to represent the current mode.

```
gui.fill_rect(x=100, y=180, w=70, h=30, color="white") # Draw rectangle to display "Current
Mode"
gui.fill_rect(x=180, y=180, w=50, h=30, color="white") # Draw rectangle to display mode typ
e
text_2 = gui.draw_text(x=105, y=180, color="orange", text='Current', font_size=10) # Display
text "Current"
text_3 = gui.draw_text(x=105, y=193, color="orange", text='Mode:', font_size=10) # Display
text "Mode:"
text_mode = gui.draw_text(x=180, y=185, color="orange", text="Auto", font_size=10) # Disp
lay mode type
```

(5) Define button display parameters

After that, we set the initial mode to be automatic. Therefore, we need to adjust the state of buttons A and B to be disabled. Additionally, we add a "Switch Mode" button on the screen for switching between automatic and manual modes.

```
# Display buttons
"""Display buttons and set their functionalities, along with the initial state as disabled"""
button_A = gui.add_button(x=50, y=260, w=70, h=40, text="water", onclick=click_A, state='di
sabled')
button_B = gui.add_button(x=140, y=260, w=70, h=40, text="stop", onclick=click_B, state='di
sabled')
button_C = gui.add_button(x=10, y=180, w=70, h=30, text="mode", onclick=click_C)
```

(6) Define the automatic watering function

In the automatic mode, we need to set up the automatic watering functionality. Therefore, here we define a function where we set a threshold and check the soil moisture value. When the moisture value is below the set threshold, we water the plants for 3 seconds.

```
# Define automatic watering for 3 seconds when the humidity value is low
def auto_watering():
    """Define automatic watering for 3 seconds when the humidity value is low"""
    if Soil_moisture_value < 2120:
        click_A()
```

```
time.sleep(3)
click_B()
```

(7) Use the sensor to detect soil moisture

Lastly, we set up a loop to detect data from the soil moisture sensor every second, and perform automatic watering in the automatic mode.

```
while True: # Loop
    Soil_moisture_value = adc0.read_analog() # Read analog value
    print(Soil_moisture_value) # Print and display humidity value
    if is_auto_mode == 1: # If in auto mode
        auto_watering() # Perform automatic watering
    text_value.config(text=Soil_moisture_value) # Update humidity value
    time.sleep(1) # Delay for 1 second
```

**Tips:** The complete example program is as follows:

```
from unihiker import GUI # Importing the GUI module from the unihiker library
from pinpong.board import Board, Pin # Import the Board and Pin modules from the pinpong library
import time # Importing the time library

gui = GUI() # Instantiating the gui object

Board().begin() # Initialize the UniHiker board
adc0 = Pin(Pin.P21, Pin.ANALOG) # Initialize pin 21 as analog input mode
# adc0 = ADC(Pin(Pin.P21, Pin.IN)) # Initialize pin 21 as analog input mode
relay = Pin(Pin.P23, Pin.OUT) # Initialize pin 23 as digital output mode

# Displaying the background image
img = gui.draw_image(w=240, h=320, image='img/stop1.png')

# Defining callback functions
def click_A(): # Define the operation when button A is clicked - image switch
    img.config(w=240, h=320, image='img/water1.png')
    relay.write_digital(1) # Set the relay pin to high level, turn on the relay

def click_B(): # Define the operation when button B is clicked - image switch
    img.config(w=240, h=320, image='img/stop1.png')
```



```

relay.write_digital(0) # Set the relay pin to high level, turn off the relay

is_auto_mode = 1 # Define a flag for auto mode, 1 for on, 0 for off

def click_C(): # Define the operation when button C is clicked -- mode switching
    global is_auto_mode
    if is_auto_mode == 0: # If the auto mode flag is 0, indicating manual mode
        text_mode.config(text="Auto")
        button_A.config(state='disabled') # Set button state to disabled, indicating not clickable
        button_B.config(state='disabled')
        is_auto_mode = 1 # Set auto mode flag to 1, indicating auto mode is on
    elif is_auto_mode == 1: # If the auto mode flag is 1, indicating auto mode
        text_mode.config(text="Manual")
        button_A.config(state='normal') # Set button state to normal, indicating clickable
        button_B.config(state='normal')
        is_auto_mode = 0 # Set auto mode flag to 0, indicating manual mode is on

# Draw filled rectangles and display text inside the rectangles
gui.fill_rect(x=45, y=35, w=95, h=30, color="white") # Draw rectangle to display "Humidity"
gui.fill_rect(x=148, y=35, w=55, h=30, color="white") # Draw rectangle to display humidity data
ata
gui.fill_rect(x=100, y=180, w=70, h=30, color="white") # Draw rectangle to display "Current
Mode"
gui.fill_rect(x=180, y=180, w=50, h=30, color="white") # Draw rectangle to display mode type
e

text_1 = gui.draw_text(x=48, y=36, color="orange", text='Humidity:') # Display text "Humidity:"
text_value = gui.draw_text(x=155, y=36, color="orange", text="") # Display humidity data
text_2 = gui.draw_text(x=105, y=180, color="orange", text='Current', font_size=10) # Display
text "Current"
text_3 = gui.draw_text(x=105, y=193, color="orange", text='Mode:', font_size=10) # Display t
ext "Mode:"
text_mode = gui.draw_text(x=180, y=185, color="orange", text="Auto", font_size=10) # Disp
lay mode type

# Display buttons
""Display buttons and set their functionalities, along with the initial state as disabled""

```

```

button_A = gui.add_button(x=50, y=260, w=70, h=40, text="water", onclick=click_A, state='disabled')
button_B = gui.add_button(x=140, y=260, w=70, h=40, text="stop", onclick=click_B, state='disabled')
button_C = gui.add_button(x=10, y=180, w=70, h=30, text="mode", onclick=click_C)

# Define automatic watering for 3 seconds when the humidity value is low
def auto_watering():
    """Define automatic watering for 3 seconds when the humidity value is low"""
    if Soil_moisture_value < 2120:
        click_A()
        time.sleep(3)
        click_B()

while True: # Loop
    Soil_moisture_value = adc0.read_analog() # Read analog value
    print(Soil_moisture_value) # Print and display humidity value
    if is_auto_mode == 1: # If in auto mode
        auto_watering() # Perform automatic watering
    text_value.config(text=Soil_moisture_value) # Update humidity value
    time.sleep(1) # Delay for 1 second

```

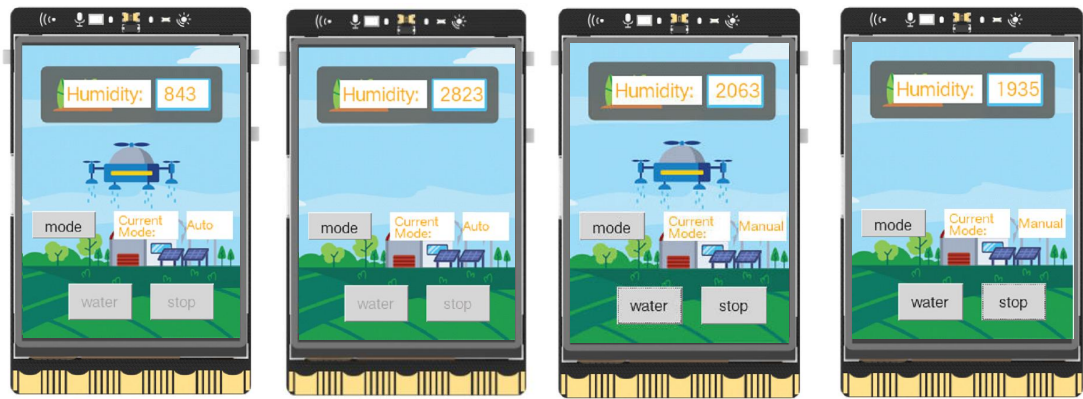
### 3. Running the Program

**STEP1:** Plug the 12V power switch into a power outlet.

**STEP 2:** Remote Connect to the UNIHAKER, Run the Program, and Observe the Results

After clicking "Run", observe the UNIHAKER. Initially, the mode is set to automatic, and the "Water" and "Stop" buttons are grayed out and cannot be clicked. The detected soil moisture value is around 843, which is lower than the threshold value we set at 2120. As a result, the water pump starts automatically and pumps water into the flower pot. Soon, the moisture value increases and reaches 2823, at which point the water pump stops.

After clicking the "Switch Mode" button, the current mode changes to manual, and the buttons return to their normal state. Now, when you click the "Water" button, the water pump starts working again. And when you click the "Stop" button, the water pump stops.



**Tips:** The threshold value "2120" in the program is a critical value we set based on experimental conditions for plant dehydration. Different plants have different water requirements and heat tolerance. When there is insufficient water, it is necessary to add water to the beaker.

## Challenge Yourself

Think about it, for this experiment, do you think there is anything that can be further optimized? Give it a try yourself! Alternatively, leave us a message with your thoughts!

## Extended Reading

### 1. Working Principle of Relays

Relays are electrical control switches that can control the on/off state of a larger current using a smaller current. Their operation principle is based on electromagnetic induction.

A relay consists of a coil and contacts. The coil is usually made of fine wire, and when current flows through the coil, a magnetic field is generated inside the relay. The presence of the magnetic field attracts or releases the contacts, thereby achieving the on/off function.

The working process of a relay is as follows:

1. When no current flows through the coil, the contacts are in the normally closed (NC) state, which means they are closed and current can flow through.

2. When current is applied to the coil, a magnetic field is generated, and the magnetic attraction forces the contacts to open, placing them in the normally open (NO) state, where current cannot flow through.
3. When the current to the coil is cut off, the magnetic field disappears, and the contacts return to the normally closed state.

Relays have the advantage of isolating the control signal from the controlled circuit, thereby protecting circuits and devices. They are widely used in automatic control systems, power systems, and industrial equipment.

