Lesson 10 Multi-Node Intelligent Agricultural System

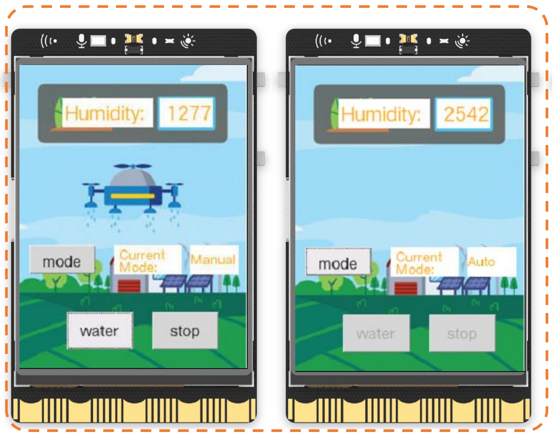
In our previous class, we embarked on designing an intelligent agricultural IoT system, which incorporated the use of a UNIHIKER for monitoring soil moisture levels during plant growth. Subsequently, the soil humidity data was uploaded to the SIoT IoT platform, enabling remote access and control of the watering system when necessary.

Nevertheless, in real-life agricultural scenarios, it is often vital to detect diversified data from various locations and aggregate it into a centralized platform for effortless remote access.

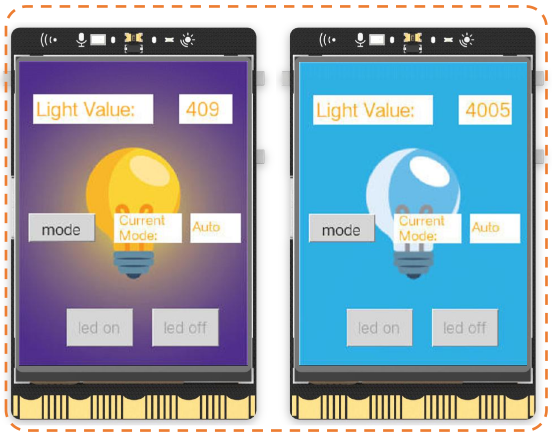
Therefore, in this class, we shall utilize three UNIHIKER to create an advanced multi-node intelligent agricultural network that simulates a practical agricultural setting.

Task Objectives

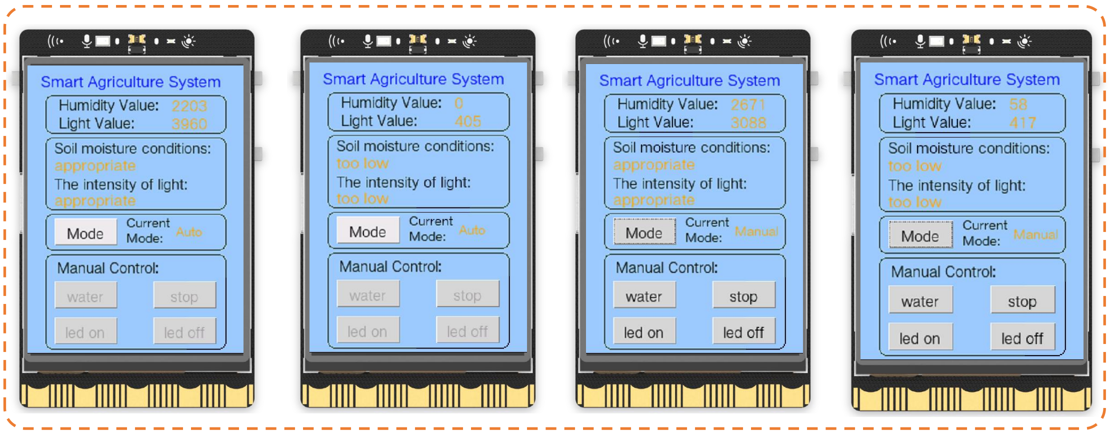
Prepare multiple UNIHIKER(taking 3 as an example) and connect all the boards to the same LAN segment of the computer. Then, start the Service Toggle as a server on the third UNIHIKER separately. It will display the soil moisture data detected by the first board and the light intensity value data detected by the second board on its screen, while also sending them to the SIoT platform of the third board. Finally, make the third board subscribe to the messages received from the IoT platform, displaying the soil humidity status of board 1 and the light intensity status of board 2 together on the screen. Moreover, when the detected environmental data is not good, it will automatically send an alert email to our mailbox to remind us to water and supplement light in time.



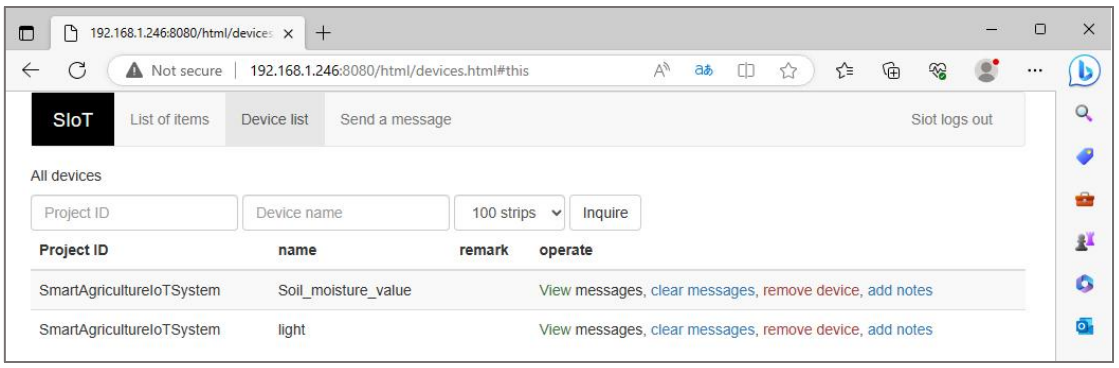
UNIHIKER 1



UNIHIKER 2



UNIHIKER 3

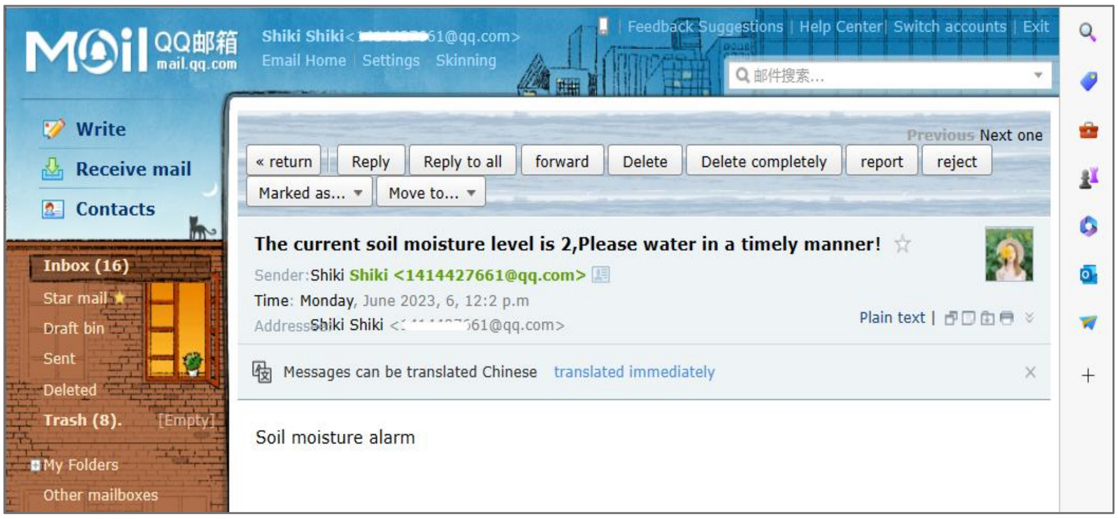

192.168.1.246:8080/html/devices.html#this

SIoT | List of items | Device list | Send a message | Siot logs out

All devices

Project ID | Device name | 100 strips | Inquire

Project ID	name	remark	operate
SmartAgricultureIoTSystem	Soil_moisture_value		View messages, clear messages, remove device, add notes
SmartAgricultureIoTSystem	light		View messages, clear messages, remove device, add notes



Shiki Shiki<[redacted]61@qq.com>

The current soil moisture level is 2,Please water in a timely manner!

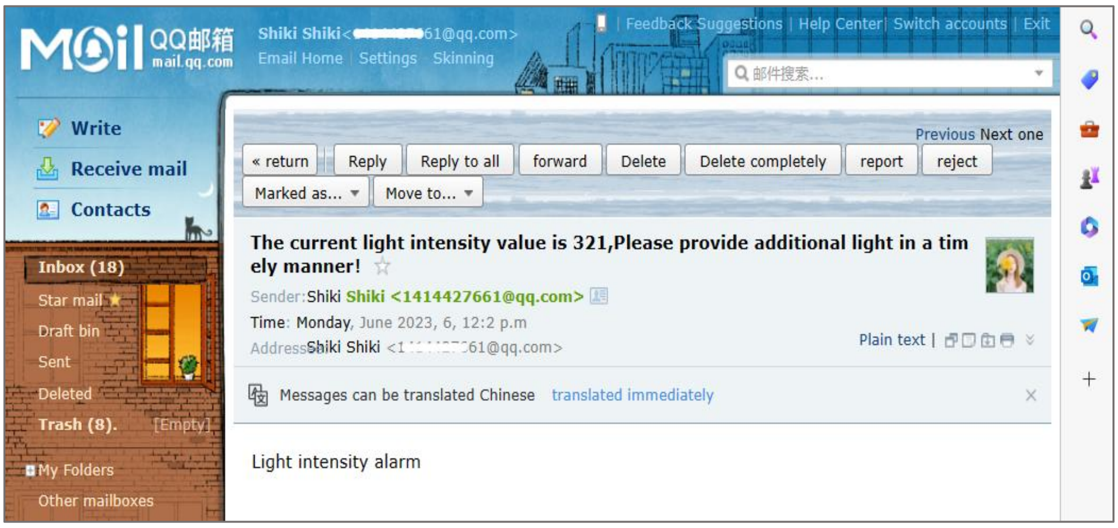
Sender: Shiki **Shiki <1414427661@qq.com>**

Time: Monday, June 2023, 6, 12:2 p.m

Address Shiki Shiki <[redacted]61@qq.com>

Messages can be translated Chinese translated immediately

Soil moisture alarm



Shiki Shiki<[redacted]61@qq.com>

The current light intensity value is 321,Please provide additional light in a timely manner!

Sender: Shiki **Shiki <1414427661@qq.com>**

Time: Monday, June 2023, 6, 12:2 p.m

Address Shiki Shiki <1[redacted]61@qq.com>

Messages can be translated Chinese translated immediately

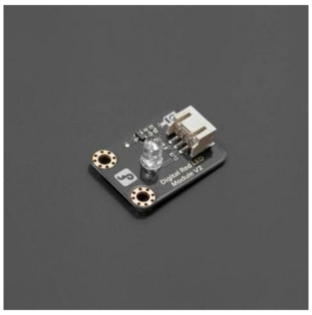
Light intensity alarm

Knowledge points

1. Learn how to use three UNIHIKER to build a multi-node IoT system.
2. Master various methods of supplying power to the UNIHIKER.
3. Learn different methods of remotely connecting to the UNIHIKER and running programs.
4. Learn the process of sending emails using the smtplib library.
5. Learn the specific methods of sending emails using the smtplib library.

Material List

Hardware List:

UNIHIKER 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	Numeric Red LED Display Module x1	

Software Preparation: Mind+ Programming Softwarex1

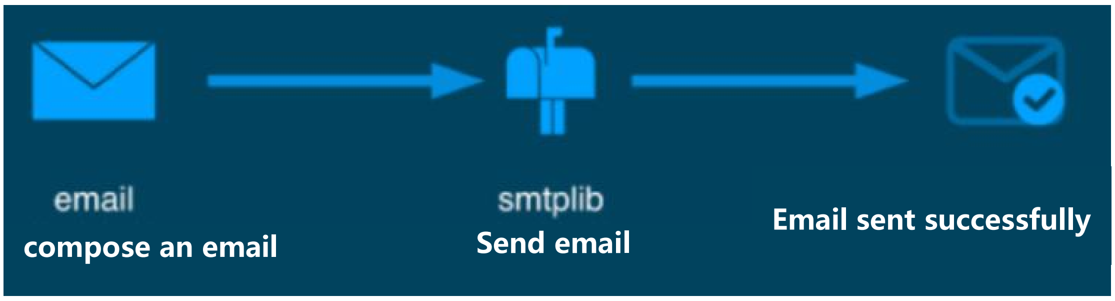
Others:

1. Plant pot with a plant x1
2. Beaker with water x1
3. Cross/flathead screwdriver x1
4. Type-C or USB-A Power Adapter or Power Bank x3

Knowledge background

1. smtplib and email libraries

Sending emails using Python requires two key steps: constructing the email content and sending the email. For the former, we can utilize the email library, and for the latter, we can use the smtplib library.



2. Creating Email Text with MIMEText Module in email.mime.text Package

The MIMEText module in the email.mime.text package allows us to create text content when constructing an email. To use it, we need to import the module first and then instantiate the MIMEText() class to create a text object.

```
from email.mime.text import MIMEText # Import the MIMEText module from the email.mime.text package, responsible for handling text

from email.utils import formataddr # Import the formataddr module from the email.utils package, responsible for formatting the input content
```

In this context, "Hello World" refers to the text content to be sent, "plain" indicates the format of the text, and "utf-8" specifies the encoding.

3. Formatting Content with the formataddr Module in the email.utils Package

The formataddr module in the email.utils package allows formatting the input content when constructing an email, ensuring that it is recognized correctly by the mail server. To use this module, it needs to be imported first.

```
from email.utils import formataddr # Import the formataddr module from the email.utils package, responsible for formatting the input content

msg['From'] = formataddr([my_name, my_sender]) # Define the sender information: the name and email address of the sender

    msg['To'] = formataddr([my_user_name, my_user]) # Define the recipient information: the name and email address of the recipient

    msg['Subject'] = title # Define the subject of the email
```

In this case, "msg['From']" represents the sender's information in the email, including the email address and nickname; "msg['To']" represents the recipient's information in the email, including the email address and nickname; "msg['Subject']" represents the subject of the email, and the text content "邮件测试" is the specific content of the email.

Here, we use the "formataddr()" function to format the text of the sender's information and recipient's information in the email.

4. Functions of the smtplib Library

The smtplib library enables the functionality to send emails. To use this library, first import it. Then, create an SMTP service object using the "SMTP_SSL()" function to connect to the email server. Next, use the "login()" function to log in to the email account. After that, the "sendmail()" function can be used to send the email. Finally, use the "quit()" function to disconnect from the email server.

(1) Creating SMTP Service Object with smtplib Library's SMTP_SSL() Function

The SMTP_SSL() function in the smtplib library is used to create an SMTP service object for connecting to an email server. To use this function, you need to import the smtplib library first.

```
import smtplib # Import the smtplib library

server = smtplib.SMTP_SSL("smtp.qq.com", 465) # Create an SMTP service and connect to the q
q email server. The SMTP protocol encryption port is 465.
```

Here, "smtp.qq.com" refers to the QQ email server, and "465" represents the encrypted port for the SMTP protocol.

(2) Logging into Email with the login() Function of the smtplib Library

The login() function in the smtplib library allows you to log into the sender's email account. To use this function, you need to set up the account credentials and the third-party login authorization code for the email account you want to log in to.

```
my_sender = '1414427661@qq.com' # Set the sender's email account, enter your
own email

my_pass = 'btxgotoklvfxfeba' # Set the sender's email authorization code, cannot
be empty

server.login(my_sender, my_pass) # Log in to the email account with the sender's
email address and password
```

In this case, "my_sender" refers to the sender's email account, and "my_pass" refers to the sender's email authorization code.

(3) Sending Emails with the sendmail() Function of the smtplib Library

The sendmail() function in the smtplib library allows for sending emails. To use this function, you need to set up the sender's email account, recipient's email account, and the content of the email to be sent.

```
msg = MIMEText(content, 'plain', 'utf-8') # Create a message text object, with content as the em
ail content, 'plain' as the text format, and 'utf-8' as the encoding

my_sender = '1414427661@qq.com' # Set the sender's email account, enter your own email

my_user = '1414427661@qq.com' # Set the recipient's email account, I send it to myself here

server.sendmail(my_sender, my_user, msg.as_string()) # Send the email with the sender's email a
ddress, recipient's email address, and the message in string format
```

In this case, "my_sender" refers to the sender's email account, "my_user" refers to the recipient's email account, and "msg.as_string()" is used to convert the email into a string format.

(4) Closing the Connection with the quit() Function of the smtplib Library

The quit() function in the smtplib library is used to close the connection between the email server and the client.

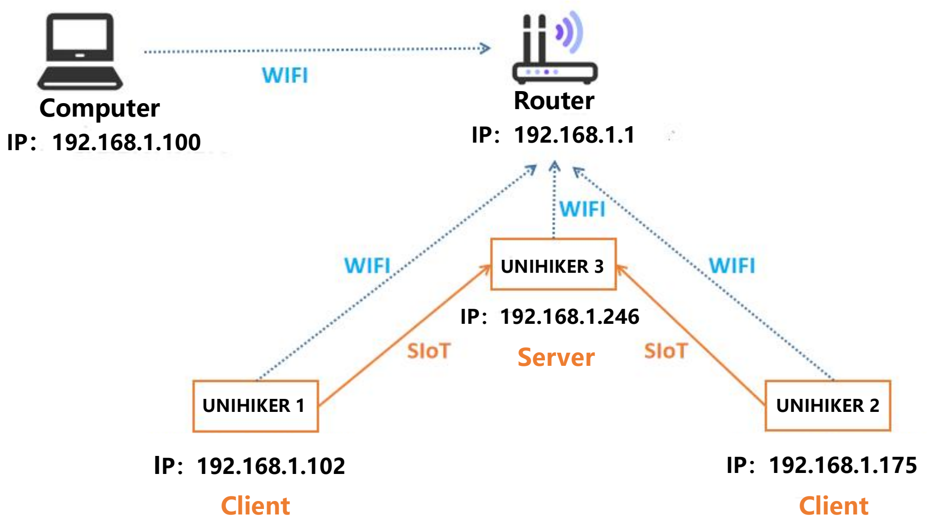
```
server.quit() # Close the connection
```

Hands-on practice

Task Description 1: Preparations

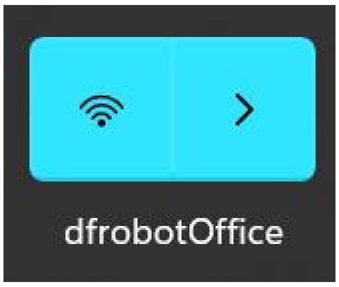
1. Analysis and Design

In this project, we will accomplish the design of a multi-node smart agriculture system. Firstly, we need to connect three UNIHIKER to the computer within the same local area network (using a router or mobile hotspot). Next, we will use a USB cable to connect each board to the computer one by one for network configuration. Finally, we will start the Service Toggle on one of the boards as a server, while the other boards will act as clients. Through programming, we will send data to the SIoT IoT platform.



2. Network Configuration

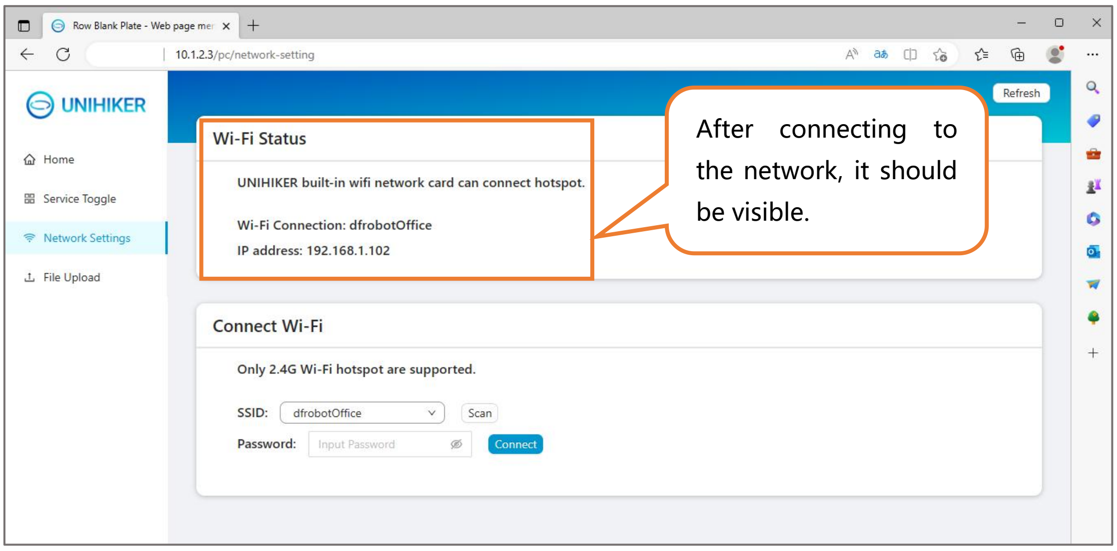
STEP 1: Connect the computer to the WiFi of the router



Tip: If you don't have a router, you can also connect to the internet by using the hotspot feature on your mobile phone.

STEP 2: Take one UNIHIKER and connect it to the computer using a USB cable.

STEP 3: Open a browser and log in to the management interface, the web menu, of the UNIHIKER by entering "10.1.2.3". In the web menu, configure the network settings to connect the UNIHIKER to the same Wi-Fi network as the computer.


Row Blank Plate - Web page me

10.1.2.3/pc/network-setting

UNIHIKER

Home

Service Toggle

Network Settings

File Upload

Refresh

Wi-Fi Status

UNIHIKER built-in wifi network card can connect hotspot.

Wi-Fi Connection: dfrobotOffice

IP address: 192.168.1.102

After connecting to the network, it should be visible.

Connect Wi-Fi

Only 2.4G Wi-Fi hotspot are supported.

SSID: dfrobotOffice Scan

Password: Input Password Connect

STEP 4: Once done, unplug the USB cable and use the same method to configure the network for the other two UNIHIKER.

STEP 5： Power on all three UNIHIKER and keep them in the powered-on state. Here, there are three ways to power the boards, and you can choose any one of them:

(1) Use a power adapter that connects to the USB or Type-C port.

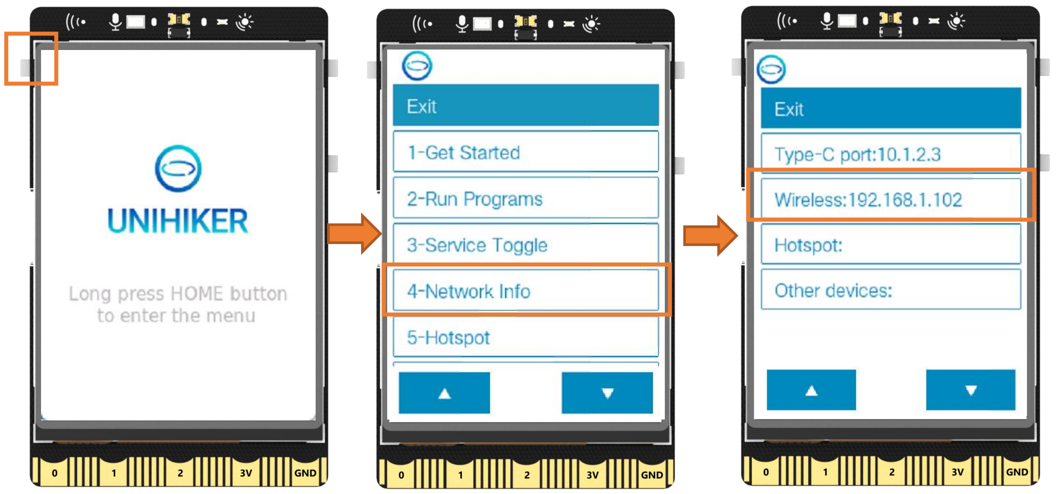
(2) Use a power bank that connects to the USB or Type-C port.

(3) Use the USB port on the computer to power the boards.

STEP 6： Check the IP addresses of the wireless connections for each board.

Check and record the IP addresses of the wireless connections for the three boards in sequence. As shown in the figure below, the first three sets of numbers should be the same, indicating that we have successfully connected the three boards to the same local network segment.

Tips: Here, the wireless connection IP addresses for the three UNIHIKER are as follows:” 192.168.1.102” , “192.168.1.246” , “192.168.1.175” .



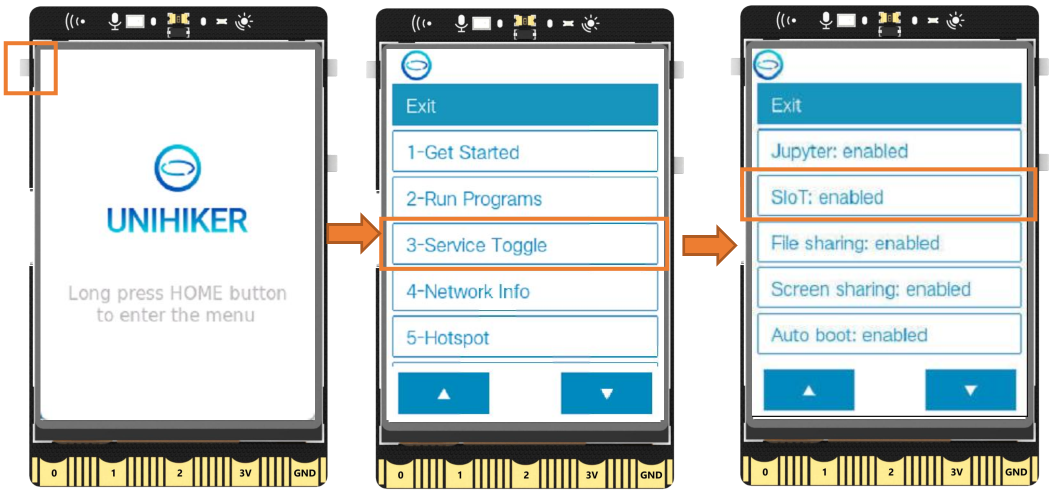
3. Start the SIoT

STEP 1: Start the SIoT service on one of the boards separately.

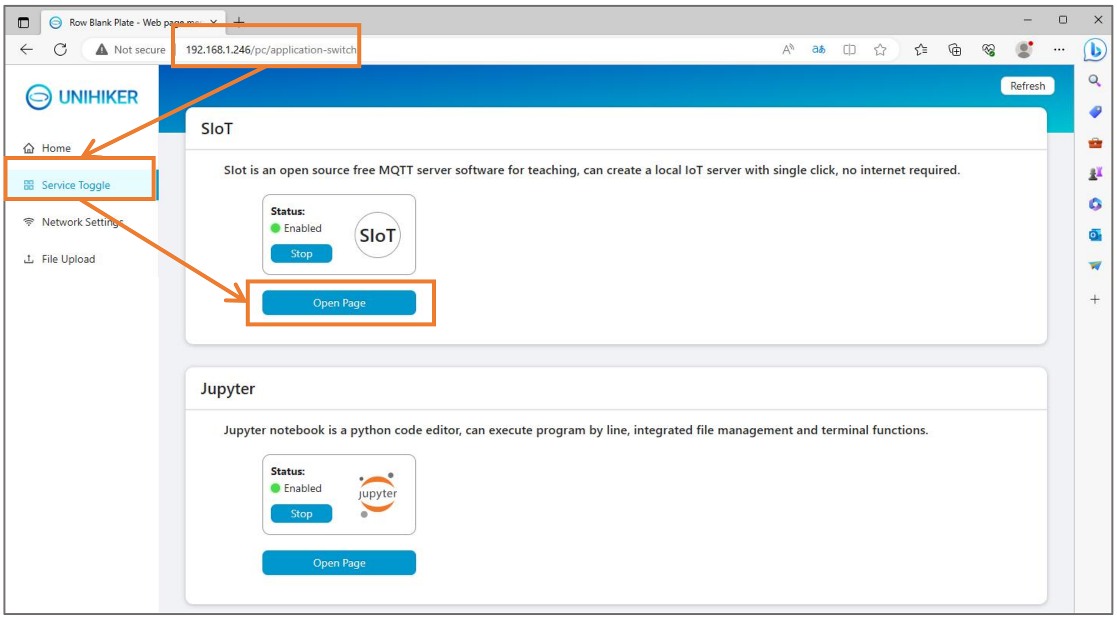
Find one of the UNIHIKER and press the HOME button to enter the menu. Click on "Service Toggle" and find the SIoT. Enable it by clicking on it, as shown in the figure below. At the same time, make sure that the SIoT is closed on the other two UNIHIKER.

Tip1: Here, we have started the SIoT service on the board with the IP address "192.168.1.246".

Tip2: Closing the SIoT on other UNIHIKER is to avoid data being sent to the wrong location in case of incorrect address input.



STEP 2: Check the computer's network connection. Open a web browser, enter the IP address "192.168.1.246", and press Enter to access the UNIHIKER web menu. Find the SIoT service under the "Service Toggle" section and click on "Open Page".



SIoT

Slot is an open source free MQTT server software for teaching, can create a local IoT server with single click, no internet required.

Status:
Enabled
Stop
SIoT
Open Page

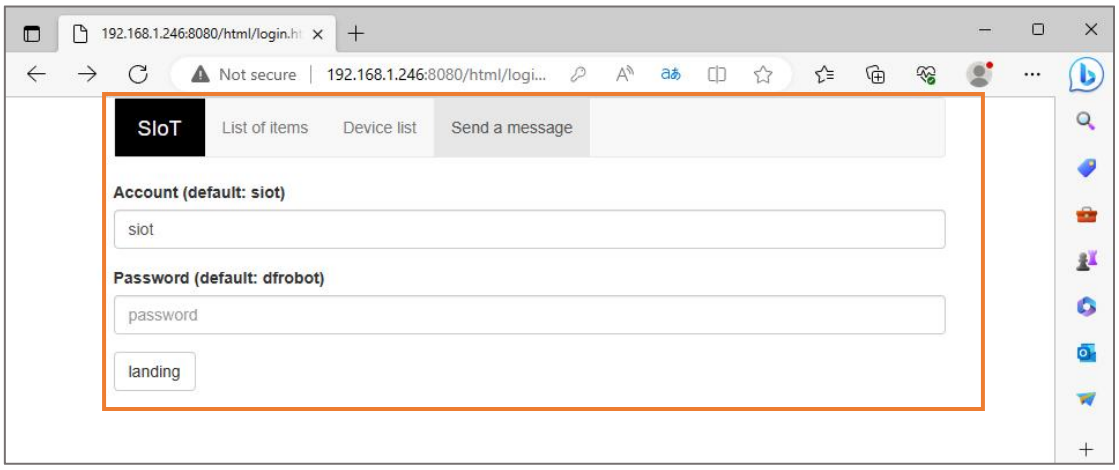
Jupyter

Jupyter notebook is a python code editor, can execute program by line, integrated file management and terminal functions.

Status:
Enabled
Stop
jupyter
Open Page

Tips: Here, you need to replace "192.168.1.246" with your actual IP address.

If you see the following interface (IoT platform web page), it indicates that your computer and the board are connected to the same local area network, and the Service Toggle has been successfully launched.


192.168.1.246:8080/html/login.ht
Not secure
192.168.1.246:8080/html/logi...
SIoT
List of items
Device list
Send a message
landing

Account (default: siot)

siot

Password (default: dfrobot)

password

Task Description 2: UNIHIKER 1 Soil Moisture Detection

In this task, for ease of distinction, we will refer to the board with the activated Service Toggle as UNIHIKER 3, while the other two boards will be UNIHIKER 1 and UNIHIKER 2, respectively.

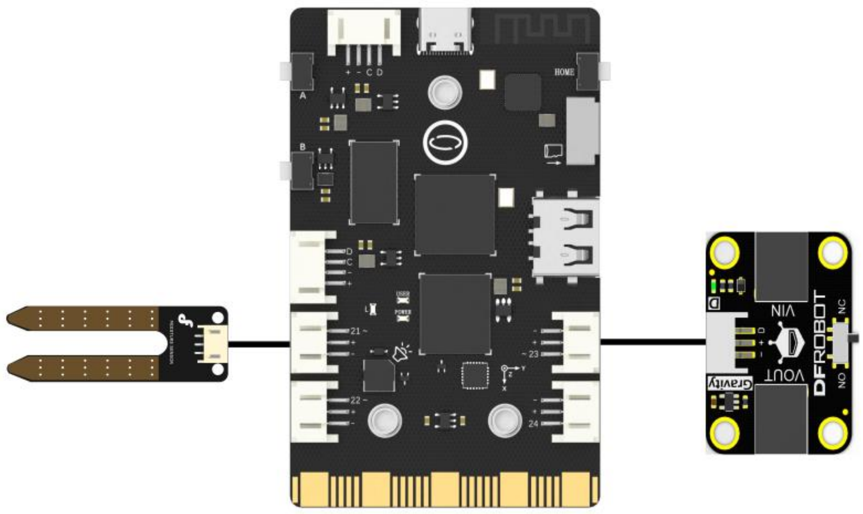
In this task, our objective is to display a smart agriculture background image on the screen of UNIHIKER 1. We will connect an external soil moisture sensor to the board to measure the moisture data. The measured data will be displayed on the screen and simultaneously sent to the SIoT platform on Board 3. The overall functionality is the same as in the previous lesson.

Furthermore, UNIHIKER 1 is equipped with both automatic and manual modes. In automatic mode, when the detected soil moisture value is too low, watering is initiated automatically using an external relay and water pump. In manual mode, watering can be performed by clicking a button on the screen. The modes can be manually switched using the buttons.

Additionally, it is possible to remotely control watering, stop watering, switch to automatic mode, or switch to manual mode by sending the following instructions through the SIoT IoT platform web interface: "relay on", "relay off", "auto" and "manual.

1. Hardware setup

STEP 1: Connect the relay to Pin P23 on the UNIHIKER.

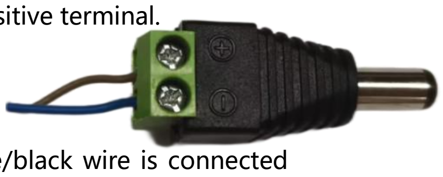


STEP 2: Use a screwdriver to connect the positive and negative wires of the water pump to the adapter. Follow the steps and illustration below:

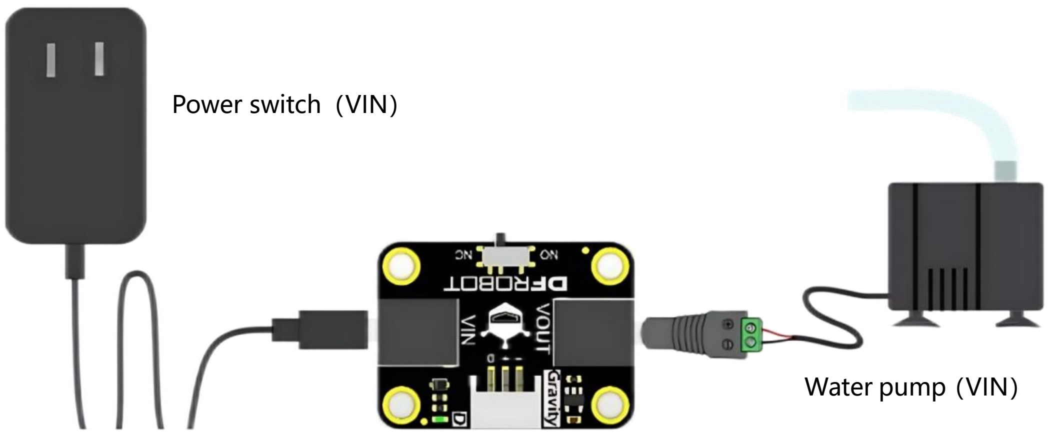
(1) Loosen the screws on the adapter.

(2) Insert the wires, paying attention to the polarity (brown/red - positive; blue/black - negative).

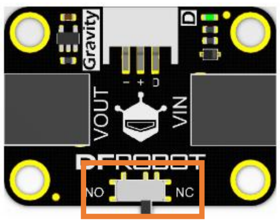
(3) Tighten the screws.


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

STEP 3: Connect the 12V power switch to the water pump's adapter using a relay.



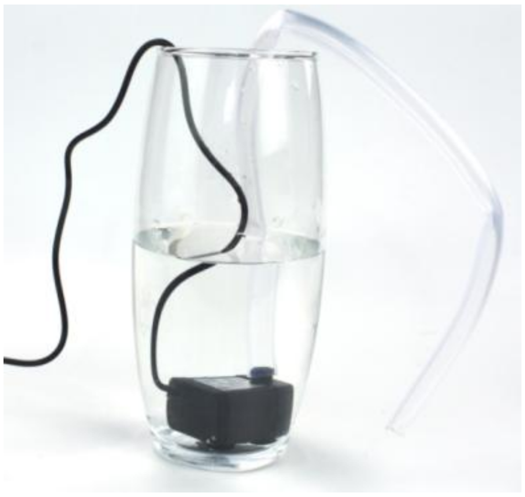
STEP 4: Set the relay switch to the NC (Normally Closed) terminal.



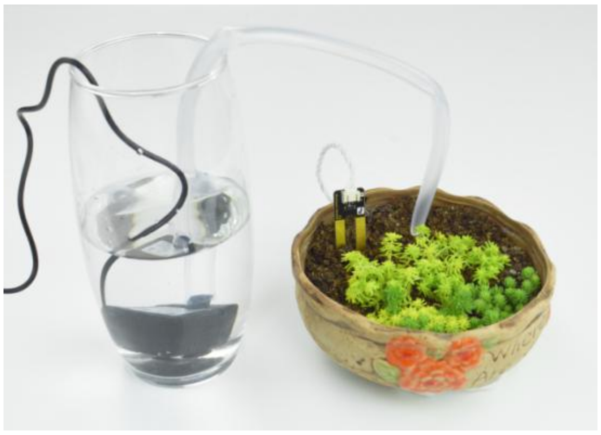
Switch to the NC (Normally Closed) terminal.

STEP 5： 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.



STEP 6： Insert the water pipe and soil moisture sensor into the flowerpot.



Tips: The specific steps mentioned above can be referred to in Lesson 8 for detailed instructions.

2. program coding

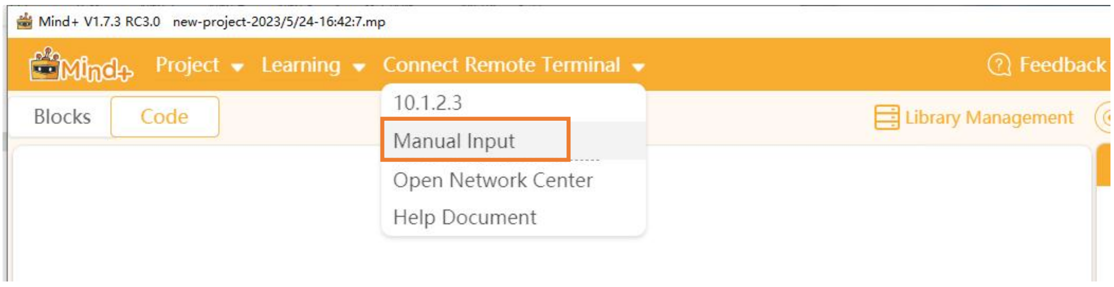
STEP 1： Creating and Saving Python Files

To start Mind+ and save the project with the name "010 Multi-node Smart Agriculture System 01” .

STEP 2: Remote connection to UNIHIKER

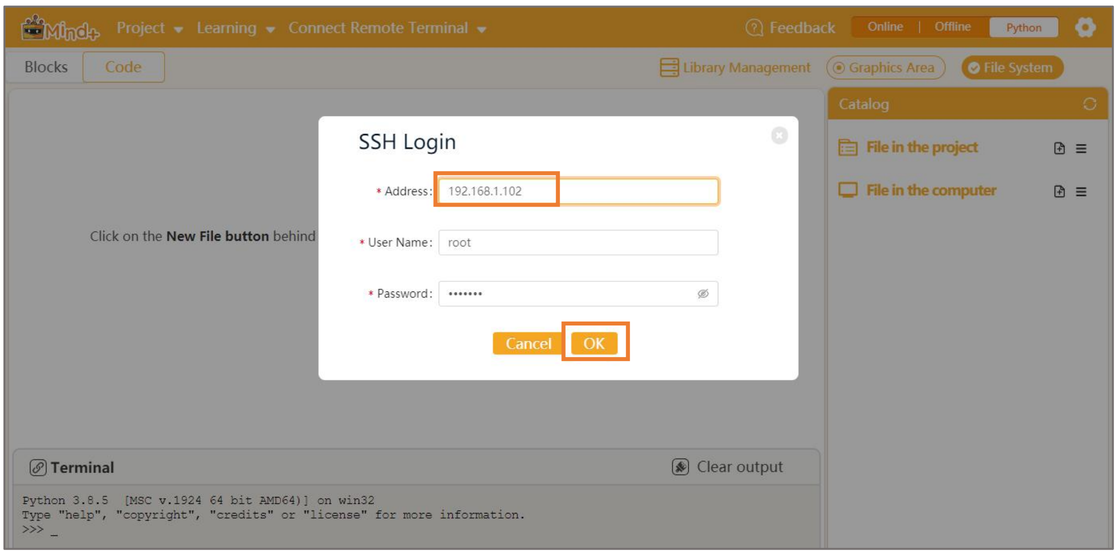
To program different UNIHIKER, we can open multiple instances of Mind+ simultaneously. In each Mind+ instance, manually enter a different IP address to connect to and access different boards.

(1) Select "Manual Input"



(2) Enter the wireless connection IP address of UNIHIKER 1 as "192.168.1.102"

Tips: Please replace it with the actual IP address of your UNIHIKER 1. Also, make sure that the board is powered on and functioning properly, and both the board and your computer are connected to the same Wi-Fi network to establish a successful connection.



* Address: 192.168.1.102

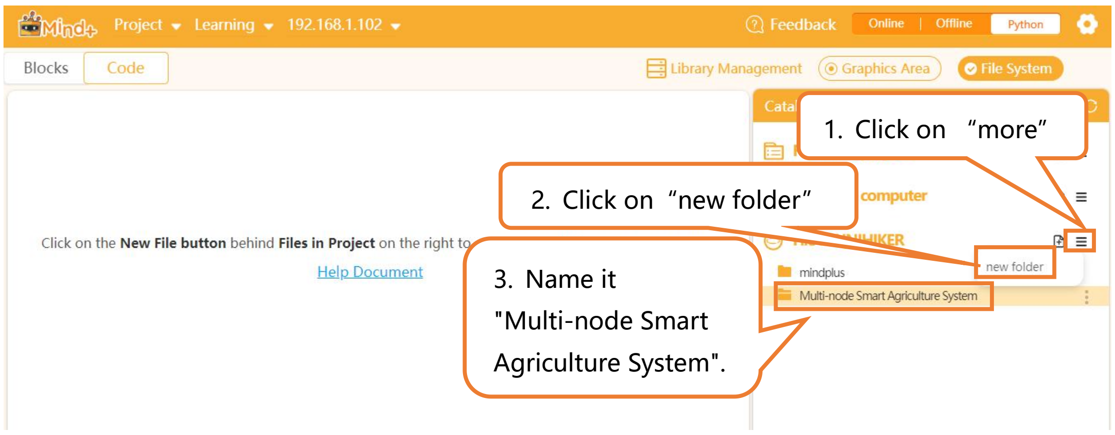
* User Name: root

* Password: •••••••

```
Python 3.8.5  [MSC v.1924 64 bit AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

STEP 3: Create a new file in UNIHIKER 1 and name it "Multi-node Smart Agriculture System".

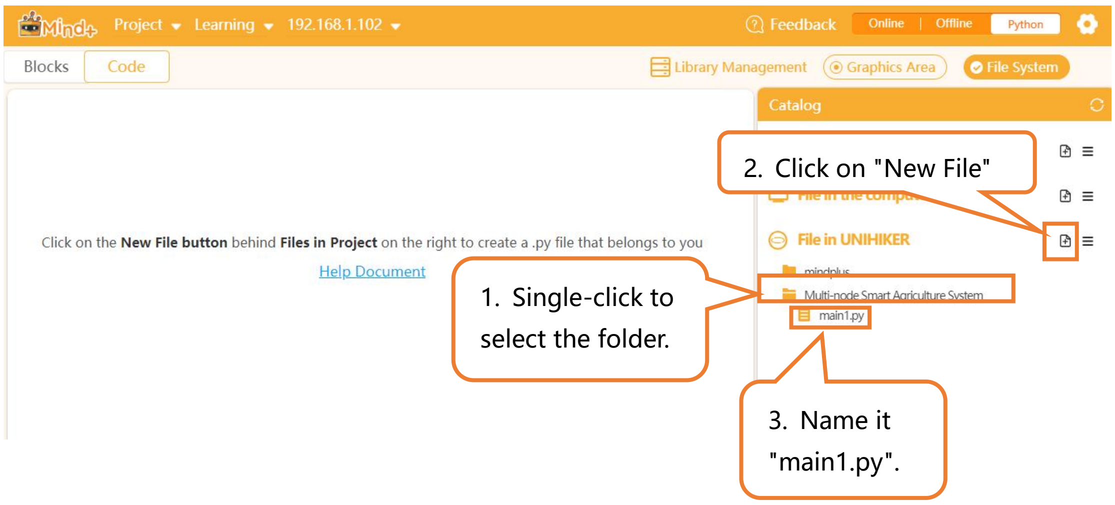
Tips: Here, we will directly create the program on the board itself. This allows us to run the program directly from the UNIHIKER Home menu without the need to launch Mind+. The instructions for operating this will be provided later.



Click on the **New File button** behind **Files in Project** on the right to

STEP 4: Create and save a Python file

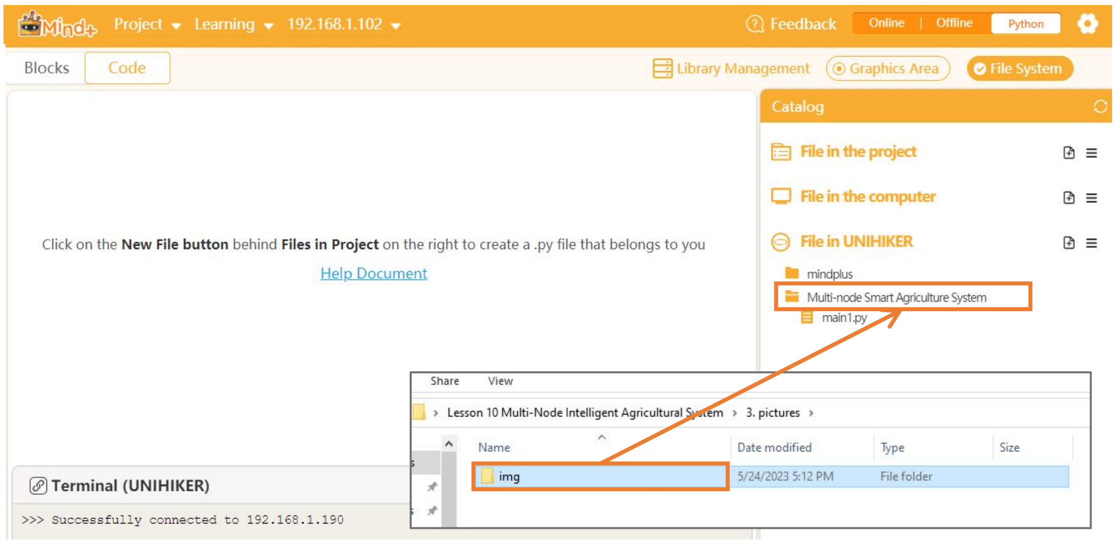
In the "Multi-node Smart Agriculture System" folder, create a Python program file named "main1.py" and double-click to open it.



Click on the **New File button** behind **Files in Project** on the right to create a .py file that belongs to you

STEP 5: Import image folder

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



Click on the **New File button** behind **Files in Project** on the right to create a .py file that belongs to you

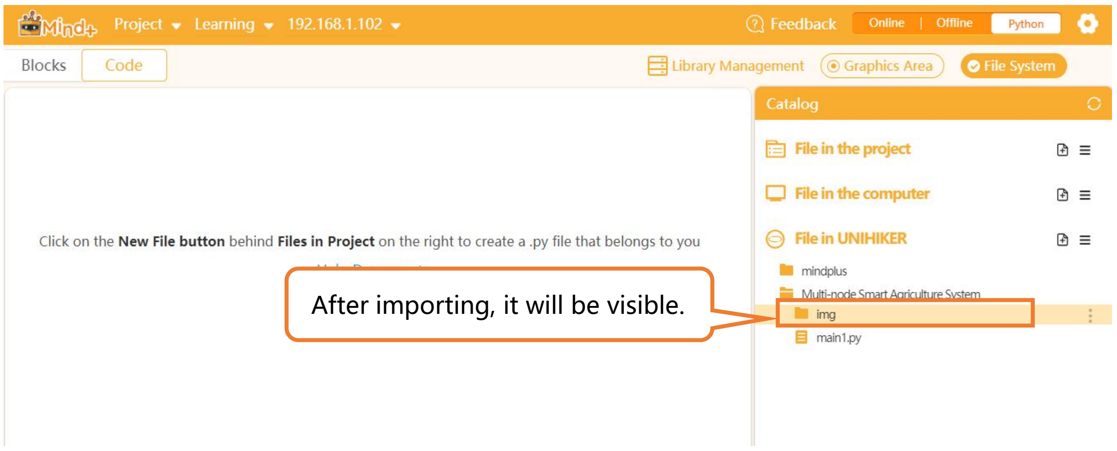
Help Document

Terminal (UNIHIKER)

```
>>> Successfully connected to 192.168.1.190
```

Lesson 10 Multi-Node Intelligent Agricultural System › 3. pictures ›

Name	Date modified	Type	Size
img	5/24/2023 5:12 PM	File folder	



Click on the **New File button** behind **Files in Project** on the right to create a .py file that belongs to you

STEP 6: Programming

(1) Import the required libraries

```
from unihiker import GUI  # Import the GUI module from the unihiker lib
rary

from pinpong.board import Board, Pin  # Import the Board and Pin module
s from the pinpong library

import time  # Import the time library

import siot  # Import the siot library
```

(2) Initialize the UNIHIKER and pins.

```
Board().begin()  # Initialize the UniHiker board

adc0 = Pin(Pin.P21, Pin.ANALOG)  # Initialize pin 21 as analog input mo
de
```

```
relay = Pin(Pin.P23, Pin.OUT)  # Initialize pin 23 as digital output mo
de
```

(3) Connect to the SIoT IoT platform and configure the functionality for sending and subscribing to message data.

Tips: In the IP address field, enter the wireless hotspot IP address of UNIHIKER 3 that was recorded earlier.

```
'''Set IoT platform connection parameters'''

SERVER = "192.168.1.246"  # MQTT server IP address, enter your actual I
P address

CLIENT_ID = ""  # CLIENT_ID can be left empty on the SIoT platform

IOT_UserName = 'siot'  # Username

IOT_PassWord = 'dfrobot'  # Password

IOT_pubTopic1 = 'SmartAgricultureIoTSystem/Soil_moisture_value'  # Topi
c for Soil_moisture_valuet data, "project_name/device_name"


def sub_relay(client, userdata, msg):

    topic = msg.topic

    payload = msg.payload.decode()

    '''Define the operations when receiving commands'''

    print("\nTopic:" + topic + " Message:" + payload)  # Print the rece
ived message

    if payload == 'relay on':  # If "relay on" is received

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

        relay.write_digital(1)  # Set the relay to high level

    elif payload == 'relay off':  # If "relay off" is received

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

        relay.write_digital(0)  # Set the relay to low level

    elif payload == 'auto':  # If "auto" is received

        print("auto")

        click_C()  # Switch to auto mode
    elif payload == 'manual':  # If "manual" is received
        print("manual")
        click_C()  # Switch to manual mode


siot.init(CLIENT_ID, SERVER, user=IOT_UserName, password=IOT_PassWord)
 # Initialize with the correct username and password
siot.connect()  # Connect to the siot IoT platform
siot.subscribe(IOT_pubTopic1, sub_relay)  # Subscribe to the IoT platfo
rm message
siot.loop()  # Start the loop to receive messages
```

(4) Display the screen interface.

```
'''The display screen interface'''
gui = GUI()  # Instantiate the GUI object
img = gui.draw_image(w=240, h=320, image='img/stop.png')  # Display the
 background image


def click_A():
    '''Define the operations when button A is clicked - switch to displ
ay watering image'''
    img.config(w=240, h=320, image='img/water.png')
    relay.write_digital(1)  # Set the relay to high level


def click_B():
    '''Define the operations when button B is clicked - switch to displ
ay stop watering image'''
    img.config(w=240, h=320, image='img/stop.png')
    relay.write_digital(0)  # Set the relay to low level


is_auto_mode = 1  # Initialize the mode to auto
```

```
def click_C():  # Define the operations when button C is clicked - swit
ch mode
    global is_auto_mode
    if is_auto_mode == 0:  # If the auto mode flag is 0, i.e., original
ly in manual mode, then when clicked
        text_mode.config(text="Auto")
        button_A.config(state='disabled')  # Set the button state to 'd
isabled' to prevent clicking
        button_B.config(state='disabled')
        is_auto_mode = 1  # Set the auto mode flag to 1 to enable auto
mode
    elif is_auto_mode == 1:  # If the auto mode flag is 1, i.e., origin
ally in auto mode, then when clicked
        text_mode.config(text="Manual")
        button_A.config(state='normal')  # Set the button state to 'nor
mal' to enable clicking
        button_B.config(state='normal')
        is_auto_mode = 0  # Set the auto mode flag to 0 to enable manua
l mode


# Draw filled rectangles
gui.fill_rect(x=45, y=35, w=95, h=30, color="white")  # Draw a rectangl
e to display "Humidity value"
gui.fill_rect(x=148, y=35, w=55, h=30, color="white")  # Draw a rectang
le to display humidity value data
gui.fill_rect(x=100, y=190, w=70, h=30, color="white")  # Draw a rectan
gle to display "Current mode"
gui.fill_rect(x=180, y=190, w=50, h=30, color="white")  # Draw a rectan
gle to display mode type


# Display text within the rectangles
gui.draw_text(x=48, y=36, color="orange", text='Humidity:')  # Display
the text "Humidity:"
text_value = gui.draw_text(x=155, y=36, color="orange", text="")  # Dis
```

```
play humidity value data

text_2 = gui.draw_text(x=105, y=190, color="orange", text='Current', fo
nt_size=10)  # Display text "Current"

text_3 = gui.draw_text(x=105, y=203, color="orange", text='Mode:', font
_size=10)  # Display text "Mode:"

text_mode = gui.draw_text(x=180, y=195, color="orange", text="Auto", fo
nt_size=10)  # Display mode type



# Display buttons

'''Display buttons and set the operations triggered by clicking the but
tons, initially in a disabled state'''

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

button_B = gui.add_button(x=140, y=260, w=70, h=40, text="stop", onclic
k=click_B, state='disabled')

button_C = gui.add_button(x=10, y=190, w=70, h=30, text="mode", onclick
=click_C)
```

(5) Define the automatic watering function.

```
# Define automatic watering for 3 seconds when the humidity value is lo
w

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()

    else:

        click_B()
```

(6) Loop to continuously monitor the humidity value and send data to the IoT platform.

```
while True:  # Loop
```

```
    Soil_moisture_value = adc0.read_analog()  # Read the analog value

    #print(Soil_moisture_value)  # Print and display the humidity value

    siot.publish(IOT_pubTopic1, Soil_moisture_value)  # Publish informa
tion to the IoT platform

    if is_auto_mode == 1:

        auto_watering()  # Perform automatic watering

    text_value.config(text=Soil_moisture_value)  # Update the humidity
value

    time.sleep(1)  # Delay for 1 second
```

Tips: The complete example program is as follows:

```
#Board 1 detects soil moisture data and sends it to the SIoT IoT platform hosted on Board 3


from unihiker import GUI  # Import 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  # Import the time library

import siot  # Import the siot library


'''Set IoT platform connection parameters'''

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


SERVER = "192.168.1.246"  # MQTT server IP address, enter your actual IP address

CLIENT_ID = ""  # CLIENT_ID can be left empty on the SIoT platform

IOT_UserName = 'siot'  # Username

IOT_PassWord = 'dfrobot'  # Password

IOT_pubTopic1 = 'SmartAgricultureIoTSystem/Soil_moisture_value'  # Topic for Soil_moisture_v
aluet data, "project_name/device_name"


def sub_relay(client, userdata, msg):
    topic = msg.topic
    payload = msg.payload.decode()
    '''Define the operations when receiving commands'''
    print("\nTopic:" + topic + " Message:" + payload)  # Print the received message
    if payload == 'relay on':  # If "relay on" is received
        img.config(w=240, h=320, image='img/water.png')
        relay.write_digital(1)  # Set the relay to high level
    elif payload == 'relay off':  # If "relay off" is received
        img.config(w=240, h=320, image='img/stop.png')
        relay.write_digital(0)  # Set the relay to low level
    elif payload == 'auto':  # If "auto" is received
        print("auto")
        click_C()  # Switch to auto mode
    elif payload == 'manual':  # If "manual" is received
        print("manual")
        click_C()  # Switch to manual mode
```

```
siot.init(CLIENT_ID, SERVER, user=IOT_UserName, password=IOT_PassWord)  # Initialize with the correct username and password
siot.connect()  # Connect to the siot IoT platform
siot.subscribe(IOT_pubTopic1, sub_relay)  # Subscribe to the IoT platform message
siot.loop()  # Start the loop to receive messages
```

```
gui = GUI()  # Instantiate the GUI object
img = gui.draw_image(w=240, h=320, image='img/stop.png')  # Display the background image
```

```
def click_A():
    '''Define the operations when button A is clicked - switch to display watering image'''
    img.config(w=240, h=320, image='img/water.png')
    relay.write_digital(1)  # Set the relay to high level
```

```
def click_B():
    '''Define the operations when button B is clicked - switch to display stop watering image'''
    img.config(w=240, h=320, image='img/stop.png')
    relay.write_digital(0)  # Set the relay to low level
```

```
is_auto_mode = 1  # Initialize the mode to auto
```

```
def click_C():  # Define the operations when button C is clicked - switch mode
    global is_auto_mode
    if is_auto_mode == 0:  # If the auto mode flag is 0, i.e., originally in manual mode, then whe
n clicked
        text_mode.config(text="Auto")
        button_A.config(state='disabled')  # Set the button state to 'disabled' to prevent clicking
        button_B.config(state='disabled')
        is_auto_mode = 1  # Set the auto mode flag to 1 to enable auto mode
    elif is_auto_mode == 1:  # If the auto mode flag is 1, i.e., originally in auto mode, then when
clicked
        text_mode.config(text="Manual")
        button_A.config(state='normal')  # Set the button state to 'normal' to enable clicking
        button_B.config(state='normal')
        is_auto_mode = 0  # Set the auto mode flag to 0 to enable manual mode
```

```
# Draw filled rectangles
gui.fill_rect(x=45, y=35, w=95, h=30, color="white")  # Draw a rectangle to display "Humidity v
alue"
gui.fill_rect(x=148, y=35, w=55, h=30, color="white")  # Draw a rectangle to display humidity v
alue data
gui.fill_rect(x=100, y=190, w=70, h=30, color="white")  # Draw a rectangle to display "Current
mode"
```

```
gui.fill_rect(x=180, y=190, w=50, h=30, color="white")  # Draw a rectangle to display mode type
```

```
# Display text within the rectangles
gui.draw_text(x=48, y=36, color="orange", text='Humidity:')  # Display the text "Humidity:"
text_value = gui.draw_text(x=155, y=36, color="orange", text="")  # Display humidity value data
text_2 = gui.draw_text(x=105, y=190, color="orange", text='Current', font_size=10)  # Display text "Current"
text_3 = gui.draw_text(x=105, y=203, color="orange", text='Mode:', font_size=10)  # Display text "Mode:"
text_mode = gui.draw_text(x=180, y=195, color="orange", text="Auto", font_size=10)  # Display mode type
```

```
# Display buttons
'''Display buttons and set the operations triggered by clicking the buttons, initially in a disabled state'''
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=190, w=70, h=30, text="mode", onclick=click_C)
```

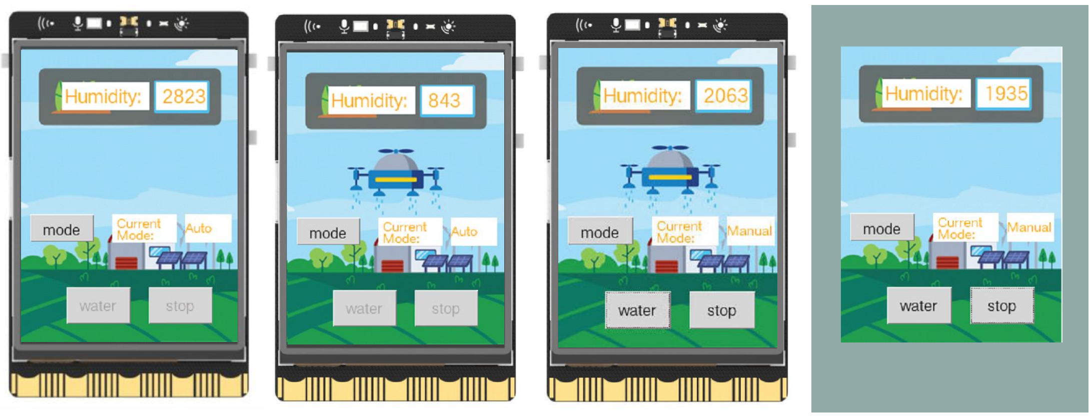
```
# Define automatic watering for 3 seconds when the humidity value is low
def auto_watering():
    '''Define automatic watering for 3 seconds when the humidity value is low'''
    if Soil_moisture_value < 2120:
        click_A()
        time.sleep(3)
        click_B()
    else:
        click_B()
```

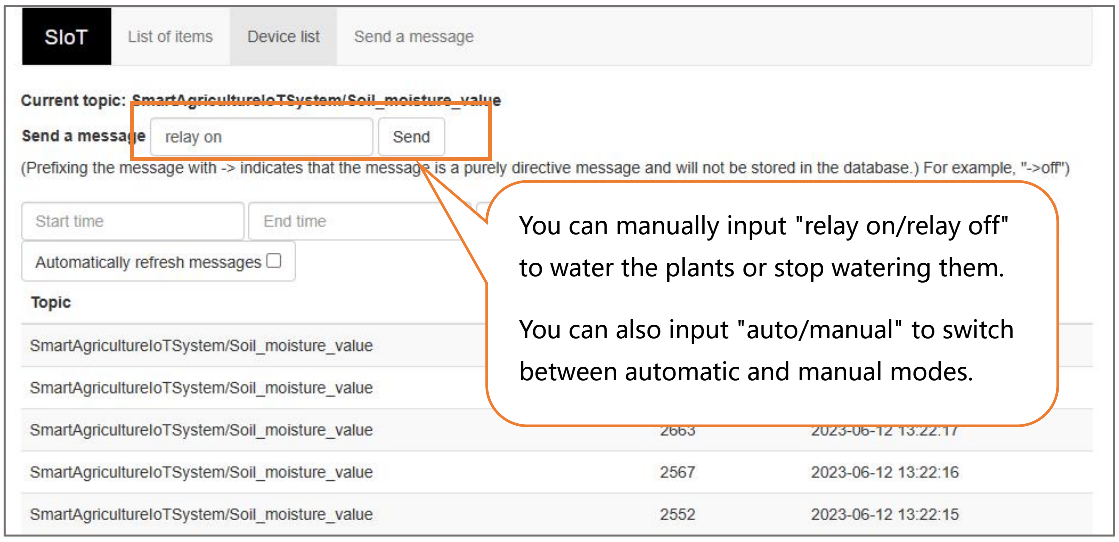
```
while True:  # Loop
    Soil_moisture_value = adc0.read_analog()  # Read the analog value
    #print(Soil_moisture_value)  # Print and display the humidity value
    siot.publish(IOT_pubTopic1, Soil_moisture_value)  # Publish information to the IoT platform
    if is_auto_mode == 1:
        auto_watering()  # Perform automatic watering
    text_value.config(text=Soil_moisture_value)  # Update the humidity value
    time.sleep(1)  # Delay for 1 second
```

3. Running the Program

STEP 1: Run and observe the results

Click the "Run" button in the Mind+ software and observe UNIHIKER 1. You will see the following interface on Board 1, where you can control watering in both manual and automatic modes. You can also open a browser and enter the IP address of the UNIHIKER where the SIoT server is located (in this case, "192.168.1.102"). Open the Service Toggle switch in the web menu and remotely view humidity data from the IoT platform and perform watering operations.





Current topic: SmartAgricultureIoTSystem/Soil_moisture_value

Send a message: relay on Send

(Prefixing the message with -> indicates that the message is a purely directive message and will not be stored in the database.) For example, "->off")

Topic		
SmartAgricultureIoTSystem/Soil_moisture_value		
SmartAgricultureIoTSystem/Soil_moisture_value		
SmartAgricultureIoTSystem/Soil_moisture_value	2663	2023-06-12 13:22:17
SmartAgricultureIoTSystem/Soil_moisture_value	2567	2023-06-12 13:22:16
SmartAgricultureIoTSystem/Soil_moisture_value	2552	2023-06-12 13:22:15

You can manually input "relay on/relay off" to water the plants or stop watering them.

You can also input "auto/manual" to switch between automatic and manual modes.

Task Description 3: UNIHIKER 2 Light Detection

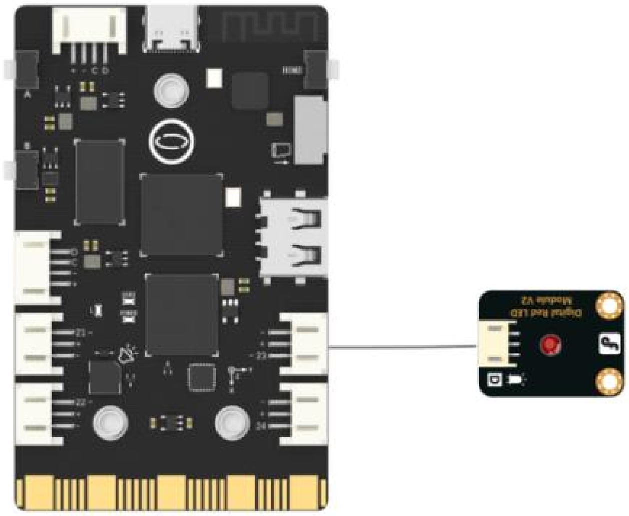
In this task, we will display a small lamp background image on the screen of UNIHIKER 2. The onboard light sensor will detect the light intensity, and the measured data will be displayed on the screen and sent to the SIoT platform on UNIHIKER 3.

Additionally, we will set up both automatic and manual modes for UNIHIKER 2. In automatic mode, when the detected light intensity is too low, an external LED will be automatically activated for supplemental lighting. In manual mode, the LED can be controlled by clicking a button on the screen. The modes can be manually switched using the buttons.

Furthermore, instructions can be sent through the SIoT IoT platform web interface to remotely control the LED, turn it on or off, switch to automatic mode, or switch to manual mode. The instructions include "led on", "led off", "auto" and "manual".

1. Hardware setup

Connect the LED module to pin P23 of UNIHIKER 2.



2. program coding

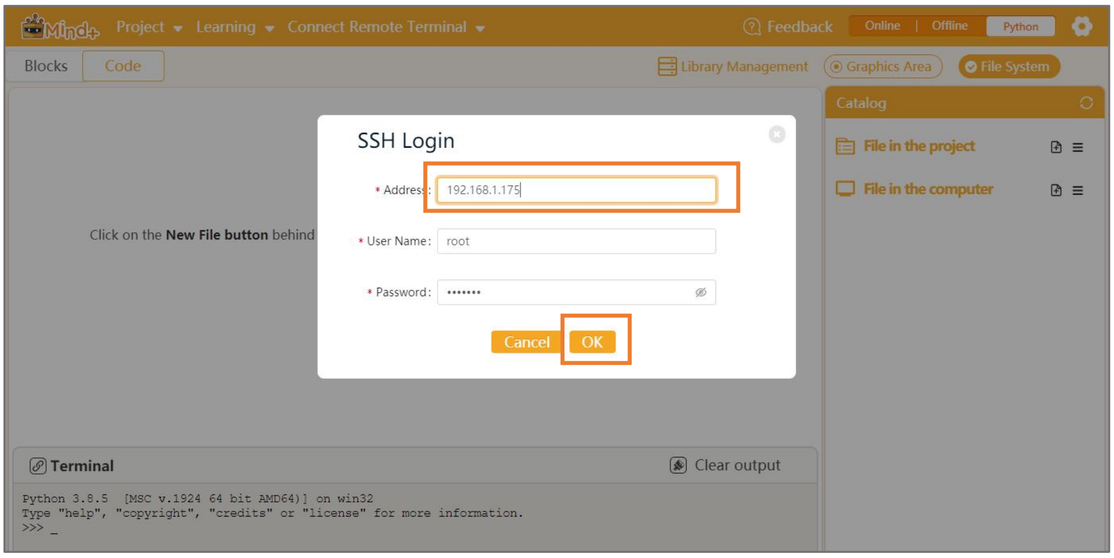
STEP 1: Creating and Saving Python Files

Start another instance of the Mind+ software, save the project with a new name as "010, Multi-node Smart Agriculture System 02" .

STEP 2: Remote connection to UNIHIKER

(1) Select "Manual Input"

(2) Enter the wireless connection IP address of UNIHIKER 2 as "192.168.1.175".



SSH Login

* Address: 192.168.1.175
* User Name: root
* Password: •••••••

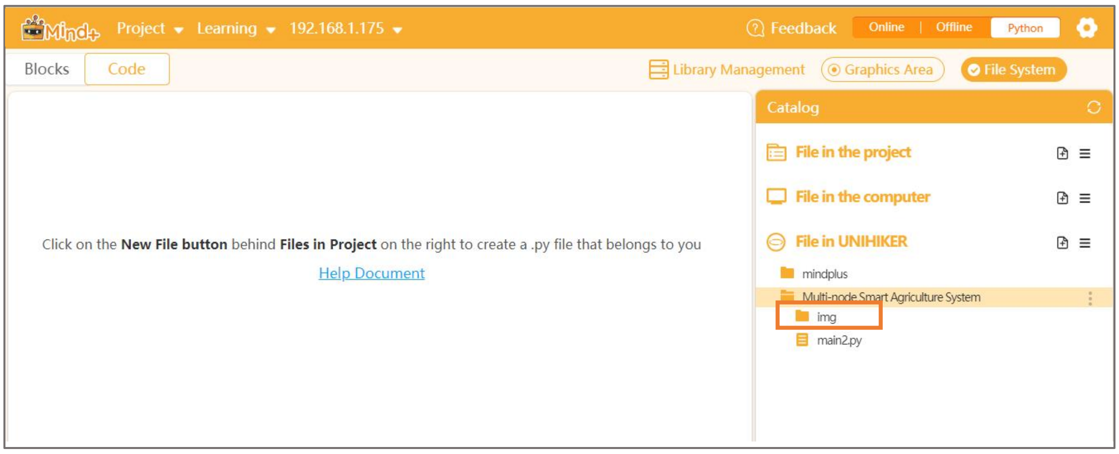
```
Python 3.8.5  [MSC v.1924 64 bit AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

STEP 3: Just like in UNIHIKER 1, create a new folder directly in UNIHIKER 2 and name it "Multi-node Smart Agriculture System"

STEP 4: Create and save a Python file

In the "Multi-node Smart Agriculture System" folder, create a Python program file named "main2.py" and double-click to open it.

STEP 5: Create and save a Python file



Click on the **New File button** behind **Files in Project** on the right to create a .py file that belongs to you

STEP 6: Programming

In general, the programming logic in this step is similar to that of board 1 mentioned above, so I won't go into great detail. However, it is important to ensure that the actual IP address of HiLink board 3 is written in the code.

Tips: The complete example program is as follows:

```python
#Board 2 detects light data and sends it to the SIoT IoT platform opened on Board 3


from unihiker import GUI  # Import the unihiker library

from pinpong.board import Board, Pin  # Import the Board and Pin modules from the pinpong library

from pinpong.extension.unihiker import *  # Import all modules from the pinpong.extension.unihiker package

import time  # Import the time library

import siot  # Import the siot library


Board().begin()  # Initialize the UniHiker board

led = Pin(Pin.P23, Pin.OUT)  # Initialize pin 23 as a digital output


SERVER = "192.168.1.246"  # MQTT server IP address, enter your actual IP address

CLIENT_ID = ""  # CLIENT_ID can be left empty on the SIoT platform
```

```python
IOT_UserName = 'siot'  # Username

IOT_PassWord = 'dfrobot'  # Password

IOT_pubTopic2 = 'SmartAgricultureIoTSystem/light'  # Topic for light data, "project_name/device_name"


def sub_led(client, userdata, msg):

    topic = msg.topic

    payload = msg.payload.decode()

    '''Define the operations when receiving commands'''

    print("\nTopic:" + topic + " Message:" + payload)  # Print the received message

    if payload == 'led on':  # If "led on" is received

        img.config(w=240, h=320, image='img/turn on.png')

        led.write_digital(1)  # Turn on the LED

    elif payload == 'led off':  # If "led off" is received

        img.config(w=240, h=320, image='img/turn off.png')

        led.write_digital(0)  # Turn off the LED

    elif payload == 'auto':  # If "auto" is received

        print("auto")

        click_C()  # Switch to auto mode

    elif payload == 'manual':  # If "manual" is received

        print("manual")

        click_C()  # Switch to manual mode


siot.init(CLIENT_ID, SERVER, user=IOT_UserName, password=IOT_PassWord)  # Initialize with the correct username and password

siot.connect()  # Connect to the SIoT platform

siot.subscribe(IOT_pubTopic2, sub_led)  # Subscribe to the topic for light data

siot.loop()  # Start the SIoT loop


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

```
img = gui.draw_image(w=240, h=320, image='img/turn off.png')  # Display the background image

def click_A():
    '''Define the operations when button A is clicked - switch to the image of the light turned on'''
    img.config(w=240, h=320, image='img/turn on.png')
    led.write_digital(1)  # Turn on the LED

def click_B():
    '''Define the operations when button B is clicked - switch to the image of the light turned off'''
    img.config(w=240, h=320, image='img/turn off.png')
    led.write_digital(0)  # Turn off the LED

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, i.e., it was in manual mode, then after clicking
        text_mode.config(text="Auto")
        button_A.config(state='disabled')  # Set the button state to 'disabled' to indicate it cannot be clicked
        button_B.config(state='disabled')
        is_auto_mode = 1  # Set the auto mode flag to 1, i.e., turn on auto mode
    elif is_auto_mode == 1:  # If the auto mode flag is 1, i.e., it was in auto mode, then after clicking
        text_mode.config(text="Manual")
        button_A.config(state='normal')  # Set the button state to 'normal' to indicate it can be clicked
        button_B.config(state='normal')
    is_auto_mode = 0  # Set the auto mode flag to 0, i.e., turn on manual mode

# Draw filled rectangles
gui.fill_rect(x=15, y=35, w=125, h=30, color="white")  # Draw a rectangle to display "Light Value"
gui.fill_rect(x=168, y=35, w=55, h=30, color="white")  # Draw a rectangle to display light value data
gui.fill_rect(x=100, y=160, w=70, h=30, color="white")  # Draw a rectangle to display "Current Mode"
gui.fill_rect(x=180, y=160, w=52, h=30, color="white")  # Draw a rectangle to display mode type

# Display text inside the rectangles
gui.draw_text(x=18, y=36, color="orange", text='Light Value:')  # Display the text "Light Value"
text_value = gui.draw_text(x=175, y=36, color="orange", text="")  # Display the light value data
text_2 = gui.draw_text(x=105, y=157, color="orange", text='Current', font_size=10)  # Display text "Current"
text_3 = gui.draw_text(x=105, y=173, color="orange", text='Mode:', font_size=10)  # Display text "Mode:"
text_mode = gui.draw_text(x=182, y=164, color="orange", text="Auto", font_size=10)  # Display the mode type

# Display buttons
'''Display buttons and set the functionality triggered when the buttons are clicked, and initially disable them'''
button_A = gui.add_button(x=50, y=260, w=70, h=40, text="led on", onclick=click_A, state='disabled')
button_B = gui.add_button(x=140, y=260, w=70, h=40, text="led off", onclick=click_B, state='disabled')
button_C = gui.add_button(x=10, y=160, w=70, h=30, text="mode", onclick=click_C)

# Define the operation of automatically turning on the light when the light value is low
```

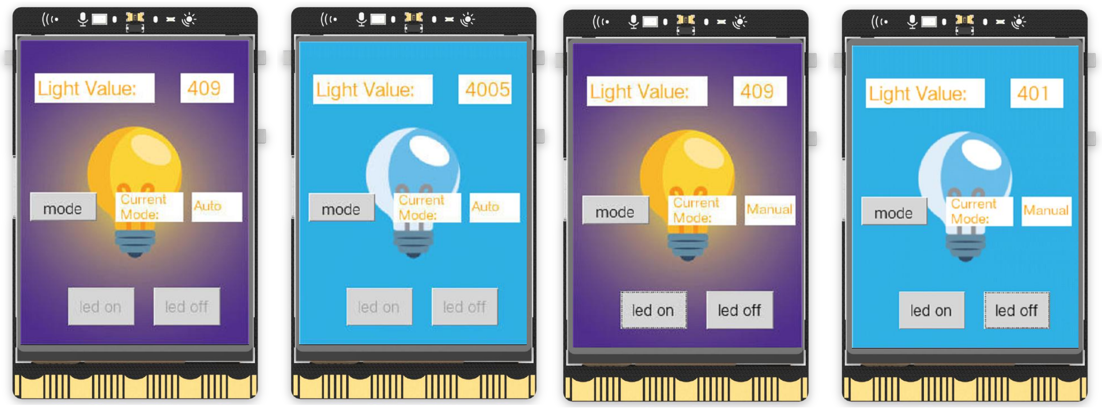
```
def auto_light():
    '''Define the operation of automatically turning on the light when the light value is low'''
    if Light < 666:
        click_A()
    else:
        click_B()

while True:  # Loop
    Light = light.read()  # Read the light value
    print(Light)  # Print and display the light value
    siot.publish(IOT_pubTopic2, Light)  #
    if is_auto_mode == 1:
        auto_light()  # Automatically turn on the light when the light value is low
    text_value.config(text=Light)  # Update the light value
    time.sleep(1)  # Delay for 1 second
```

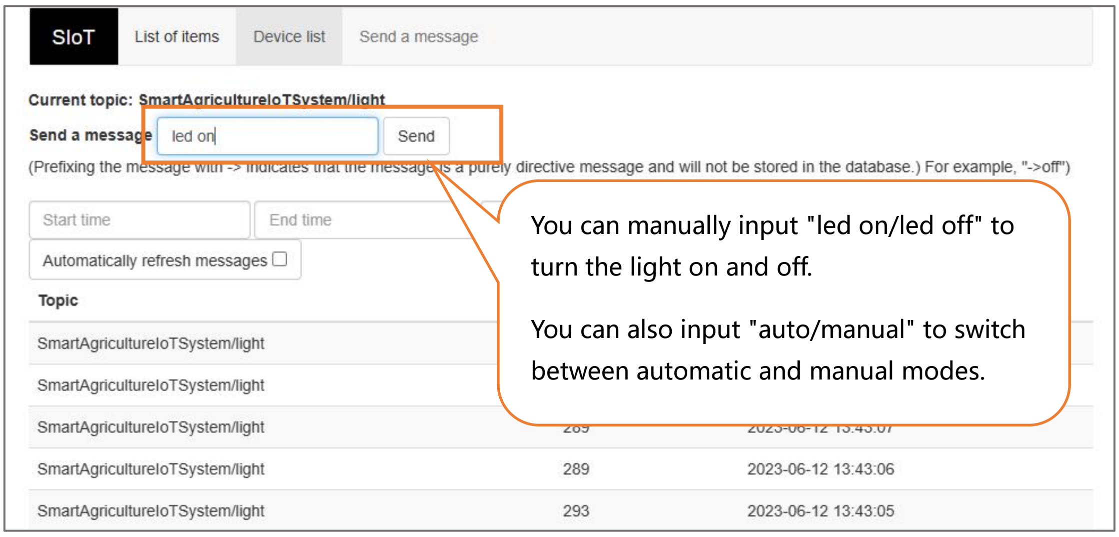
3. Running the Program

STEP 1: Run and observe the results

Clicking on the "Run" button in the Mind+ software and observing the UNIHIKER 2, you will see the interface as shown below. It allows control of the fill light (turn on the LED) manually and automatically in both modes. You can also open the SIoT in the web application by entering the IP address of the UNIHIKER where the SIoT server is located (in this case, it is "192.168.1.125") in a web browser to remotely view the light value data and perform fill light operations from the IoT platform.

Please note that the topic of UNIHIKER 2 is different from that of UNIHIKER 1, and you need to return to the device list and select the "light" project for operation.





Current topic: SmartAgricultureIoTSystem/light

Send a message: led on

(Prefixing the message with -> indicates that the message is a purely directive message and will not be stored in the database.) For example, "->off")

Automatically refresh messages ☐

Topic		
SmartAgricultureIoTSystem/light		
SmartAgricultureIoTSystem/light		
SmartAgricultureIoTSystem/light	289	2023-06-12 13:43:07
SmartAgricultureIoTSystem/light	289	2023-06-12 13:43:06
SmartAgricultureIoTSystem/light	293	2023-06-12 13:43:05

You can manually input "led on/led off" to turn the light on and off.

You can also input "auto/manual" to switch between automatic and manual modes.

Task Description 4: Remote Monitoring with UNIHIKER 3

Here we use UNIHIKER 3 as the main station to receive the soil moisture and light value data sent to the SIoT IoT platform by UNIHIKER 1 and UNIHIKER 2. The data is then summarized and displayed on the screen of UNIHIKER 3, and the buttons on UNIHIKER 3 are used to control the watering, stopping watering, turning on and turning off the lights of UNIHIKER 1 and UNIHIKER 2 respectively.

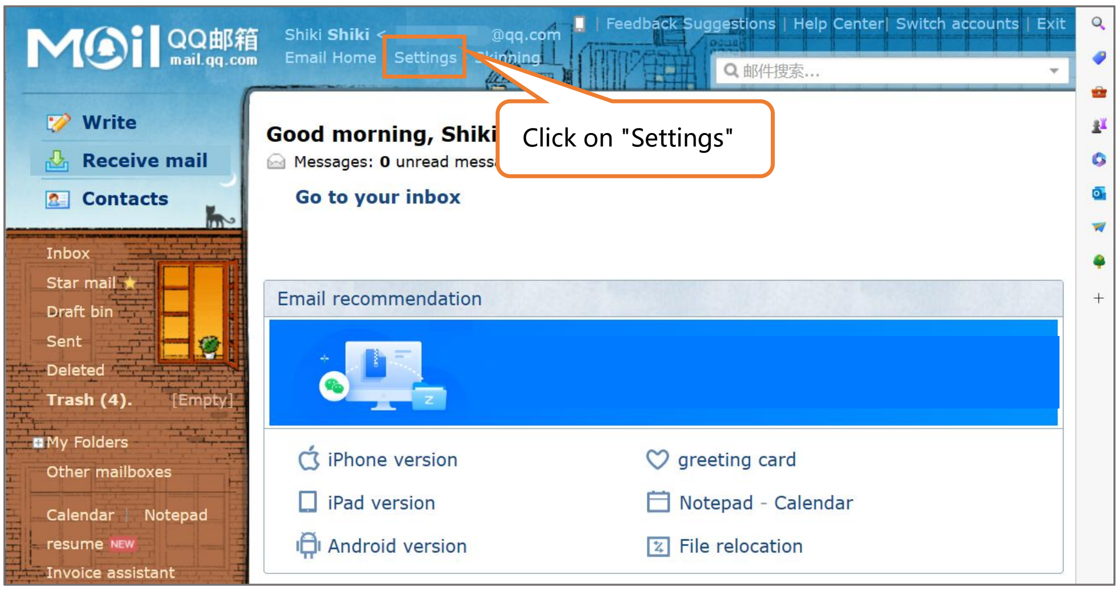
At the same time, two modes, automatic and manual, are set on UNIHIKER 3 with a switch button for each mode, used to control the mode switching of UNIHIKER 1 and UNIHIKER 2 respectively, and the initial mode is set to automatic.

Most importantly, the main station of UNIHIKER 3 can also make real-time judgments on the received data. When certain environmental data is poor, it will remotely send an email to the mailbox as an alarm prompt, so that we can obtain information promptly and perform watering and lighting.

1. Email Settings

STEP 1: Record QQ Mail Authorization Code

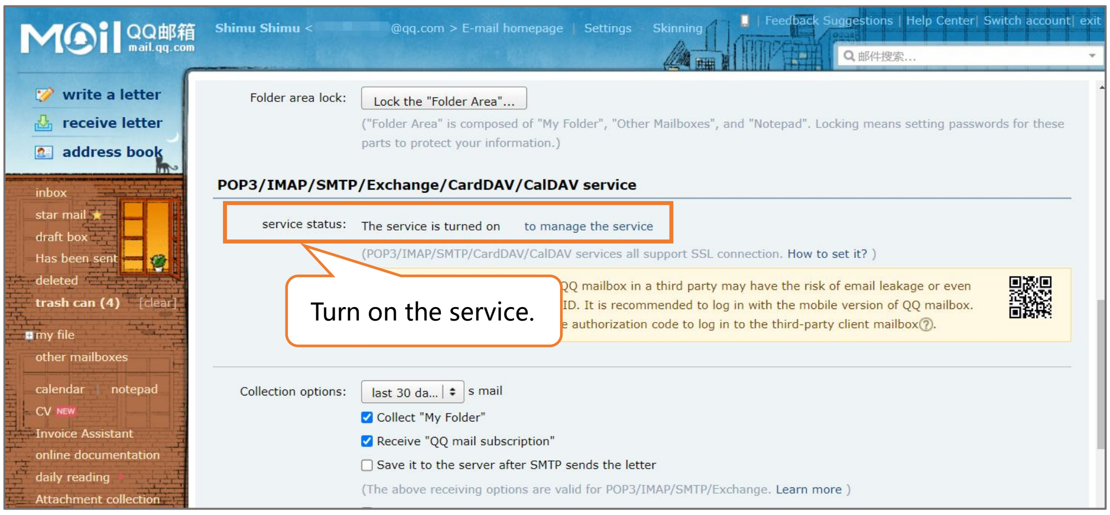
(1) Open QQ Mail and click "Settings" at the top of the page.



(2) Enable SMTP service, generate authorization code and record it

Click on "Account" and locate the "POP3/SMTP service" and "IMAP/SMTP service" options, then click "enable".

Tips: A verification code needs to be sent via phone message to generate the authorization code.



Folder area lock: Lock the "Folder Area"...
("Folder Area" is composed of "My Folder", "Other Mailboxes", and "Notepad". Locking means setting passwords for these parts to protect your information.)

POP3/IMAP/SMTP/Exchange/CardDAV/CalDAV service

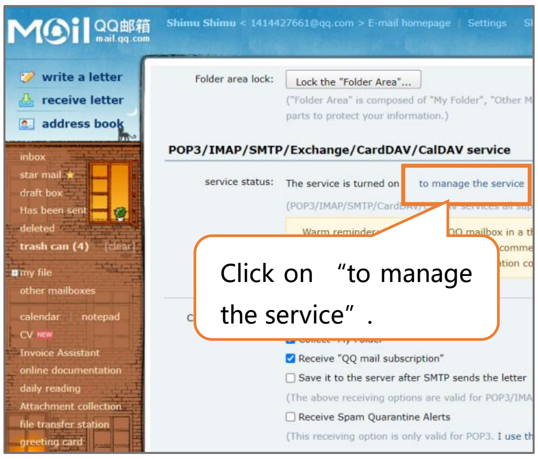
service status: The service is turned on to manage the service
(POP3/IMAP/SMTP/CardDAV/CalDAV services all support SSL connection. How to set it?)

QQ mailbox in a third party may have the risk of email leakage or even ID. It is recommended to log in with the mobile version of QQ mailbox. e authorization code to log in to the third-party client mailbox.

Collection options: last 30 da... s mail
- [x] Collect "My Folder"
- [x] Receive "QQ mail subscription"
- [ ] Save it to the server after SMTP sends the letter

(The above receiving options are valid for POP3/IMAP/SMTP/Exchange. Learn more)

Click on "to manage the service" and find the "POP3/IMAP/SMTP/Exchange/CardDVA service" on the "Security Settings" page. Click on "Generate authorization code".



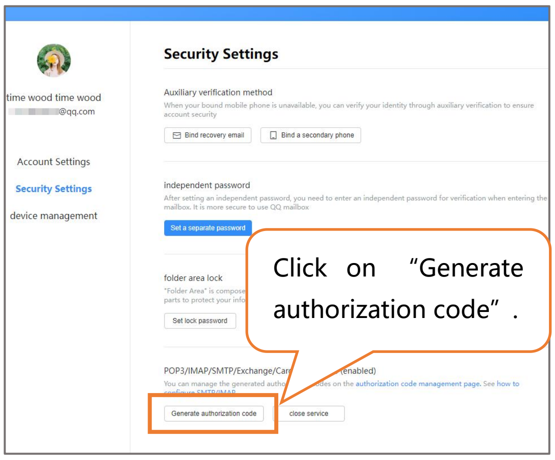
Folder area lock: Lock the "Folder Area"...
("Folder Area" is composed of "My Folder", "Other M
parts to protect your information.)

POP3/IMAP/SMTP/Exchange/CardDAV/CalDAV service

service status: The service is turned on | to manage the service
(POP3/IMAP/SMTP/CardDAV/CalDAV services all sup

Warm reminder ... QQ mailbox in a t

☑ Receive "QQ mail subscription"
☐ Save it to the server after SMTP sends the letter
(The above receiving options are valid for POP3/IMA
☐ Receive Spam Quarantine Alerts
(This receiving option is only valid for POP3. I use th



Security Settings

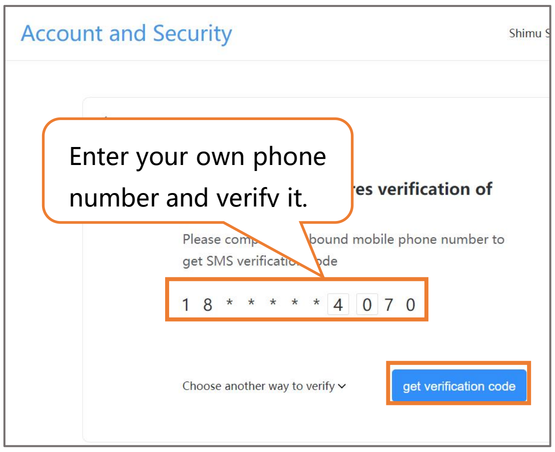
Auxiliary verification method
When your bound mobile phone is unavailable, you can verify your identity through auxiliary verification to ensure account security

independent password
After setting an independent password, you need to enter an independent password for verification when entering the mailbox. It is more secure to use QQ mailbox

folder area lock
"Folder Area" is compose
parts to protect your info

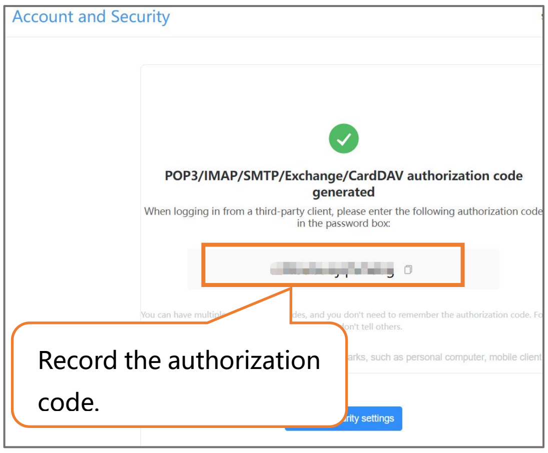
POP3/IMAP/SMTP/Exchange/Car ... (enabled)
You can manage the generated autho ... des on the authorization code management page. See how to

Select the method of receiving the verification code via your phone number, verify it successfully, and record the authorized code.


Account and Security
Shimu S
Enter your own phone number and verify it.
1 8 * * * * * 4 0 7 0
Choose another way to verify
get verification code

...es verification of

Please comp... bound mobile phone number to get SMS verificatio... ode



POP3/IMAP/SMTP/Exchange/CardDAV authorization code generated

When logging in from a third-party client, please enter the following authorization code in the password box:

You can have multipl... des, and you don't need to remember the authorization code. Fo... n't tell others.

...arks, such as personal computer, mobile client

2. program coding

STEP 1: Creating and Saving Python Files

Start another instance of the Mind+ software, save the project with a new name as "010, Multi-node Smart Agriculture System 03" .

STEP 2: Remote connection to UNIHIKER

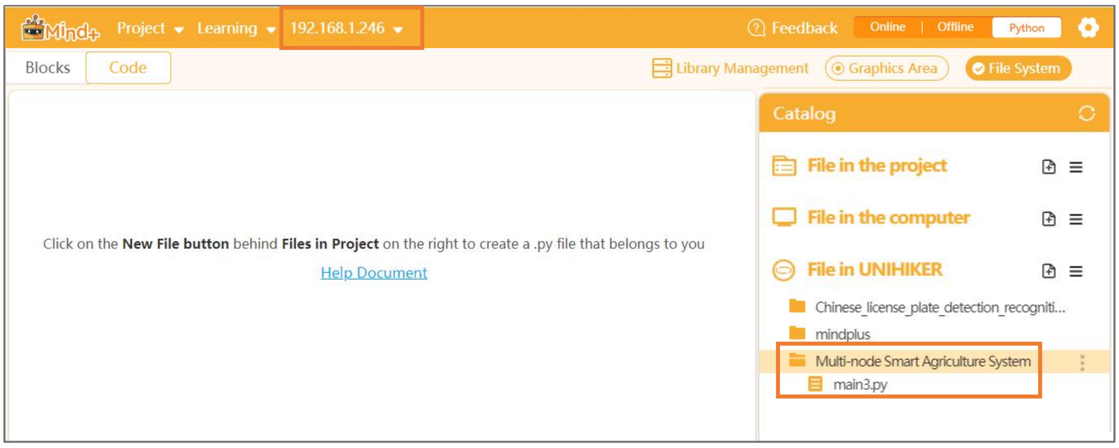
(1) Select "Manual Input"

(2) Enter the wireless connection IP address of UNIHIKER 3 as "192.168.1.246".

STEP 3: Just like in UNIHIKER 1, create a new folder directly in UNIHIKER 3 and name it "Multi-node Smart Agriculture System 03"

STEP4: Create and save a Python file

In the "Multi-node Smart Agriculture System" folder, create a Python program file named "main3.py" and double-click to open it.



Click on the **New File button** behind **Files in Project** on the right to create a .py file that belongs to you

Help Document

STEP 6: Programming

(1) Importing Required Libraries

To successfully send an email, we first need to import the smtplib library. Additionally, when constructing an email, we need to process text and format the input content. Therefore, we also need to import the MIMEText module and the formataddr module.

```
from unihiker import GUI   # Import the GUI module from the unihiker li
brary

import time  # Import the time library

import siot  # Import the siot library

import smtplib # Import the smtplib library

from email.mime.text import MIMEText # Import the MIMEText module from
the email.mime.text package, responsible for handling text

from email.utils import formataddr # Import the formataddr module from
the email.utils package, responsible for formatting the input content
```

(2) Setting IoT Platform Connection Parameters

Next, since we will be sending data from UNIHIKER 1 and UNIHIKER 2 to the IoT platform, we need to first set up the connection parameters for the IoT platform in order to subscribe later on.

```
'''Set IoT platform connection parameters'''

SERVER = "192.168.1.246"  # MQTT server IP address, enter personal actu
al IP

CLIENT_ID = ""  # On SIoT, CLIENT_ID can be left blank

IOT_UserName = 'siot'  # Username

IOT_PassWord = 'dfrobot'  # Password

IOT_pubTopic1 = 'SmartAgricultureIoTSystem/Soil_moisture_value'  # Humi
```

```
dity topic, "project name/device name"

IOT_pubTopic2 = 'SmartAgricultureIoTSystem/light'  # Light intensity to
pic, "project name/device name"
```

(3) Displaying Screen Page

Subsequently, we will exhibit four distinct sections of content on the screen of Board 3. Initially, we will showcase the humidity and light value data of Board 1 and Board 2; secondly, we will analyze the data by discrimination, and display appropriate messages such as "current soil humidity is within acceptable levels" and "current light intensity is optimal" on the screen. Following this, we have the mode switching button, which also shows the current mode being operated; ultimately, in manual mode, there exist four button controls. We will configure their functionalities utilizing callback functions.

```
# Define the callback functions for five buttons

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

    siot.publish(IOT_pubTopic1, 'relay on')  # Publish the message 'relay on' to the IoT platform


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

    siot.publish(IOT_pubTopic1, 'relay off')  # Publish the message 'relay off' to the IoT platform


def click_C():  # Define the operation when button C is clicked -- switch image

    siot.publish(IOT_pubTopic2, 'led on')  # Publish the message 'led on' to the IoT platform


def click_D():  # Define the operation when button D is clicked -- switch image

    siot.publish(IOT_pubTopic2, 'led off')  # Publish the message 'led off' to the IoT platform


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


def click_E():  # Define the operation when button E is clicked -- switch mode

    global is_auto_mode

    if is_auto_mode == 0:

        text_mode.config(text="Auto")

        button_A.config(state='disabled')
        button_B.config(state='disabled')
        button_C.config(state='disabled')
        button_D.config(state='disabled')
        siot.publish(IOT_pubTopic1, 'auto')  # Publish the message 'auto' to the IoT platform
        siot.publish(IOT_pubTopic2, 'auto')  # Publish the message 'auto' to the IoT platform
        is_auto_mode = 1
    elif is_auto_mode == 1:
        text_mode.config(text="Manual")
        button_A.config(state='normal')
        button_B.config(state='normal')
        button_C.config(state='normal')
        button_D.config(state='normal')
        siot.publish(IOT_pubTopic1, 'manual')  # Publish the message 'manual' to the IoT platform
        siot.publish(IOT_pubTopic2, 'manual')  # Publish the message 'manual' to the IoT platform
        is_auto_mode = 0

'''Display screen page'''
gui=GUI() # Instantiate the gui object


# Display filled rectangle
gui.fill_rect(x=0, y=0, w=240, h=320, color="#99CCFF") # Draw a filled rectangle as the backgro
und
# Display title
title = gui.draw_text(x=15, y=5, text='Smart Agriculture System', font_size=12, color='blue')


# Display part1 data (rounded rectangle, text)
gui.draw_round_rect(x=20, y=33, w=200, h=42, r=8, width=1) # Display rounded rectangle (start
ing point coordinates (20,33), width 200, height 42, radius 8, line width 1)
gui.draw_text(x=35, y=32, text='Humidity Value:', font_size=11)  # Display text "Humidity Value:
"
text_value = gui.draw_text(x=160, y=32, color="orange", text="", font_size=12)  # Display soil va
```

```
lue data

gui.draw_text(x=35, y=52, text='Light Value:', font_size=11) # Display text "Light Value:"

text_value_2 = gui.draw_text(x=160, y=52, color="orange", text="", font_size=12) # Display light
 value data


# Display part2 data (rounded rectangle, text)

gui.draw_round_rect(x=20, y=80, w=200, h=80, r=8,width=1)

text1 = gui.draw_text(x=30, y=80, text='Soil moisture conditions:', font_size=11, color='black') #
 Display "Soil moisture conditions:" at coordinate (30, 60), font size 11, color black

text2 = gui.draw_text(x=30, y=98, text='appropriate', font_size=12, color='orange')

text3 = gui.draw_text(x=30, y=120, text='The intensity of light:', font_size=11, color='black')

text4 = gui.draw_text(x=30, y=137, text='appropriate', font_size=12, color='orange')


# Display mode switching function (rounded rectangle, button, text) for Part 3

gui.draw_round_rect(x=20, y=165, w=200, h=40, r=8, width=1)

button_E = gui.add_button(x=30, y=170, w=70, h=30, text="Mode", onclick=click_E)

gui.draw_text(x=110, y=165, text='Current', font_size=10) # Display "Current Mode:"

gui.draw_text(x=110, y=182, text='Mode:', font_size=10) # Display "Mode:"

text_mode = gui.draw_text(x=165, y=175, color="orange", text="Auto", font_size=10) # Display
the mode type


# Display manual control mode (rounded rectangle, button, text) for Part 4

gui.draw_round_rect(x=20, y=210, w=200, h=105, r=8, width=1)

gui.draw_text(x=33, y=213, text='Manual Control:', font_size=11) # Display "Manual Control:"

button_A = gui.add_button(x=30, y=240, w=70, h=30, text="water", onclick=click_A, state='disa
bled')

button_B = gui.add_button(x=140, y=240, w=70, h=30, text="stop", onclick=click_B, state='disa
bled')

button_C = gui.add_button(x=30, y=280, w=70, h=30, text="led on", onclick=click_C, state='dis
abled')

button_D = gui.add_button(x=140, y=280, w=70, h=30, text="led off", onclick=click_D, state='di
sabled')
```

(4) Email Settings

Next, we need to configure the email sending function. First, make sure the following five parameters are set up: sender's email account, sender's email authorization code, sender's email nickname, recipient's email account, and recipient's email nickname. Afterwards, we will define a function to send emails, which will take the email content and title as parameters to enable multiple calls for sending emails.

Tips: Make sure to fill in the actual sender's account and authorization code information. The nickname can be set as desired, but cannot be left blank.

```
# Define a function for sending email alerts
def mail(content, title):
    ret = True # Define a flag variable ret to record the event of sending email, with an initial value of True
    try:
        msg = MIMEText(content, 'plain', 'utf-8') # Create a message text object, with content as the email content, 'plain' as the text format, and 'utf-8' as the encoding
        '''Three header information: sender, recipient, subject'''
        msg['From'] = formataddr([my_name, my_sender]) # Define the sender information: the name and email address of the sender
        msg['To'] = formataddr([my_user_name, my_user]) # Define the recipient information: the name and email address of the recipient
        msg['Subject'] = title # Define the subject of the email


        server = smtplib.SMTP_SSL("smtp.qq.com", 465) # Create an SMTP service and connect to the qq email server. The SMTP protocol encryption port is 465.
        server.login(my_sender, my_pass) # Log in to the email account with the sender's email address and password
        server.sendmail(my_sender, my_user, msg.as_string()) # Send the email with the sender's email address, recipient's email address, and the message in string format
        server.quit() # Close the connection
    except Exception: # If the try statement does not execute, ret is set to False
        ret = False # Set the ret flag to False
    return ret # Return the ret flag
```

(5) Define Callback Function

Since sending emails is performed when the data from UNIHIKER 1 and UNIHIKER 2 are not optimal, we need to first set the functional operation when receiving data from the SIoT IoT platform.

When receiving data on soil humidity under Topic1, we will perform three steps: first, display the data on the screen; second, discriminate the data and display the analysis result on the screen; and finally, if the discrimination result is poor, set up to send an alert email, with no less than 10 seconds interval between each email.

Similarly, when receiving data on light intensity under Topic2, we will perform the same processing.

Tips: The email sending frequency here can be modified as desired.

```
soil_mail_enable = True  # Define the flag for enabling soil alarm email sending
light_mail_enable = True  # Define the flag for enabling light alarm email sending

soil_mail_time = time.time()  # Flag for recording the scheduled time for sending soil alarm email
light_mail_time = time.time()  # Flag for recording the scheduled time for sending light alarm email

# Define callback function
def sub_cb(client, userdata, msg):
    global soil_mail_enable, light_mail_enable, soil_mail_time, light_mail_time
    topic = msg.topic # Store topic data in variable
    payload = msg.payload.decode() # Store payload message data as string in variable
    print("\nTopic: " + topic + " Message: " + payload) # Print data on the terminal

    if topic == IOT_pubTopic1:  # Define the operation when receiving Soil_moisture_value
        if payload.isdigit(): # If the received message is a number, it is a humidity value rather than a control command
            Soil_moisture_value = int(payload)  # Convert to a numerical type for easy judgment later
            text_value.config(text = str(Soil_moisture_value))  # Update the screen display data
            if Soil_moisture_value < 2120: # The threshold is set to 2120, which can be adjusted according to the actual situation
                text2.config(text="too low")  # Update the text warning
```

```python
                if soil_mail_enable == True and (time.time()-soil_mail_time) > 10:  # Only send emails if allowed and 10 seconds have passed since the last email was sent
                    ret1=mail("Soil moisture alarm","The current soil moisture level is " + str(Soil_moisture_value) + ", please water in a timely manner!")
                    if ret1:  # If there is a send email event
                        print("The email has been sent successfully 1")
                        soil_mail_enable = False  # The email has been sent, change the flag to avoid duplicate sending
                        soil_mail_time = time.time()   # Record the time of this email sending
                    else: # Otherwise
                        print("The email failed to send 1")
            else:
                text2.config(text="appropriate")  # Update the text
                soil_mail_enable = True   # If the alarm is lifted, allow subsequent emails to be sent
    elif  topic == IOT_pubTopic2:  # Define the operation when receiving Light
        if payload.isdigit(): # If the received message is a number, it is a light value rather than a control command
            light_value = int(payload)  # Convert to a numerical type for easy judgment later
            text_value_2.config(text = str(light_value))  # Update the screen with the light value
            if light_value < 666:   # The threshold is set to 666, which can be adjusted according to the actual situation
                text4.config(text="too low")  # Update the text warning
                if light_mail_enable == True and  (time.time()-light_mail_time) > 10:  # Only send emails if allowed
                    ret2=mail("Light intensity alarm","The current light intensity value is " + str(light_value) + ", please provide additional light in a timely manner!")
                    if ret2:  # If there is a send email event
                        print("The email has been sent successfully 2")
                        light_mail_enable = False  # The email has been sent, change the flag to avoid duplicate sending
                        light_mail_time = time.time()  # Record the time of this email sending
                    else:  # Otherwise
                        print("The email failed to send 2")
            else:
                text4.config(text="appropriate") # Update the text warning
                light_mail_enable = True  # If the alarm is lifted, allow subsequent emails to be sent
```

(6) Subscribe to IoT Platform Messages

After defining the callback function, we set board 3 to subscribe to messages from the IoT platform, and send "auto" to both topics at the beginning, in order to set the default starting mode to automatic.

```
siot.init(CLIENT_ID, SERVER, user=IOT_UserName, password=IOT_PassWord) # Initialize and veri
fy the correctness of the input username and password.
siot.connect() # Connect to the siot IoT platform.
siot.subscribe(IOT_pubTopic1, sub_cb) # Subscribe to Topic1 for soil moisture data.
siot.subscribe(IOT_pubTopic2, sub_cb) # Subscribe to Topic2 for light intensity data.
siot.publish(IOT_pubTopic1, 'auto') # Publish the message 'auto' to the IoT platform.
siot.publish(IOT_pubTopic2, 'auto') # Publish the message 'auto' to the IoT platform.
siot.loop() # Keep the program running in a loop.
```

(7) Loop to keep the program running

```
while True: # Keep the program running in a loop.
    time.sleep(0.5) # Delay for 0.5 seconds.
```

Tips: The complete example program is as follows:

```
'''Subscribe to Board 1 and Board 2 and send messages to the IoT platform, displaying them o
n Board 3. Send an email alert when the value is not good.'''
from unihiker import GUI   # Import the GUI module from the unihiker library
import time  # Import the time library
import siot  # Import the siot library
import smtplib # Import the smtplib library
from email.mime.text import MIMEText # Import the MIMEText module from the email.mime.t
ext package, responsible for handling text
from email.utils import formataddr # Import the formataddr module from the email.utils packa
ge, responsible for formatting the input content
```

```python
'''Set IoT platform connection parameters'''
SERVER = "192.168.1.246"  # MQTT server IP address, enter personal actual IP
CLIENT_ID = ""  # On SIoT, CLIENT_ID can be left blank
IOT_UserName = 'siot'  # Username
IOT_PassWord = 'dfrobot'  # Password
IOT_pubTopic1 = 'SmartAgricultureIoTSystem/Soil_moisture_value'  # Humidity topic, "project name/device name"
IOT_pubTopic2 = 'SmartAgricultureIoTSystem/light'  # Light intensity topic, "project name/device name"

# Define the callback functions for five buttons
def click_A():  # Define the operation when button A is clicked -- switch image
    siot.publish(IOT_pubTopic1, 'relay on')  # Publish the message 'relay on' to the IoT platform

def click_B():  # Define the operation when button B is clicked -- switch image
    siot.publish(IOT_pubTopic1, 'relay off')  # Publish the message 'relay off' to the IoT platform

def click_C():  # Define the operation when button C is clicked -- switch image
    siot.publish(IOT_pubTopic2, 'led on')  # Publish the message 'led on' to the IoT platform

def click_D():  # Define the operation when button D is clicked -- switch image
    siot.publish(IOT_pubTopic2, 'led off')  # Publish the message 'led off' to the IoT platform

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

def click_E():  # Define the operation when button E is clicked -- switch mode
    global is_auto_mode
    if is_auto_mode == 0:
        text_mode.config(text="Auto")
        button_A.config(state='disabled')
        button_B.config(state='disabled')
        button_C.config(state='disabled')
        button_D.config(state='disabled')
        siot.publish(IOT_pubTopic1, 'auto')  # Publish the message 'auto' to the IoT platform
        siot.publish(IOT_pubTopic2, 'auto')  # Publish the message 'auto' to the IoT platform
        is_auto_mode = 1
    elif is_auto_mode == 1:
        text_mode.config(text="Manual")
        button_A.config(state='normal')
        button_B.config(state='normal')
        button_C.config(state='normal')
        button_D.config(state='normal')
        siot.publish(IOT_pubTopic1, 'manual')  # Publish the message 'manual' to the IoT platform
        siot.publish(IOT_pubTopic2, 'manual')  # Publish the message 'manual' to the IoT platform
        is_auto_mode = 0
```

```
'''Display screen page'''
gui=GUI() # Instantiate the gui object
```

```
# Display filled rectangle
gui.fill_rect(x=0, y=0, w=240, h=320, color="#99CCFF") # Draw a filled rectangle as the backgr
ound
# Display title
title = gui.draw_text(x=15, y=5, text='Smart Agriculture System', font_size=12, color='blue')
```

```
# Display part1 data (rounded rectangle, text)
gui.draw_round_rect(x=20, y=33, w=200, h=42, r=8, width=1) # Display rounded rectangle (sta
rting point coordinates (20,33), width 200, height 42, radius 8, line width 1)
gui.draw_text(x=35, y=32, text='Humidity Value:', font_size=11)  # Display text "Humidity Valu
```

```
e:"

text_value = gui.draw_text(x=160, y=32, color="orange", text="", font_size=12)  # Display soil
value data

gui.draw_text(x=35, y=52, text='Light Value:', font_size=11) # Display text "Light Value:"

text_value_2 = gui.draw_text(x=160, y=52, color="orange", text="", font_size=12) # Display lig
ht value data
```

```
# Display part2 data (rounded rectangle, text)

gui.draw_round_rect(x=20, y=80, w=200, h=80, r=8,width=1)

text1 = gui.draw_text(x=30, y=80, text='Soil moisture conditions:', font_size=11, color='black')
# Display "Soil moisture conditions:" at coordinate (30, 60), font size 11, color black

text2 = gui.draw_text(x=30, y=98, text='appropriate', font_size=12, color='orange')

text3 = gui.draw_text(x=30, y=120, text='The intensity of light:', font_size=11, color='black')

text4 = gui.draw_text(x=30, y=137, text='appropriate', font_size=12, color='orange')
```

```
# Display mode switching function (rounded rectangle, button, text) for Part 3

gui.draw_round_rect(x=20, y=165, w=200, h=40, r=8, width=1)

button_E = gui.add_button(x=30, y=170, w=70, h=30, text="Mode", onclick=click_E)

gui.draw_text(x=110, y=165, text='Current', font_size=10) # Display "Current Mode:"

gui.draw_text(x=110, y=182, text='Mode:', font_size=10) # Display "Mode:"

text_mode = gui.draw_text(x=165, y=175, color="orange", text="Auto", font_size=10) # Displa
y the mode type
```

```
# Display manual control mode (rounded rectangle, button, text) for Part 4

gui.draw_round_rect(x=20, y=210, w=200, h=105, r=8, width=1)

gui.draw_text(x=33, y=213, text='Manual Control:', font_size=11) # Display "Manual Control:"

button_A = gui.add_button(x=30, y=240, w=70, h=30, text="water", onclick=click_A, state='dis
abled')

button_B = gui.add_button(x=140, y=240, w=70, h=30, text="stop", onclick=click_B, state='dis
abled')

button_C = gui.add_button(x=30, y=280, w=70, h=30, text="led on", onclick=click_C, state='di
sabled')
```

```
button_D = gui.add_button(x=140, y=280, w=70, h=30, text="led off", onclick=click_D, state='disabled')
```

```
'''Email settings'''

my_sender = '1414427661@qq.com' # Set the sender's email account, enter your own email

my_pass = 'btxgotoklvfxfeba' # Set the sender's email authorization code, cannot be empty

my_name = 'ShadowNie' # Set the sender's email nickname

my_user = '1414427661@qq.com' # Set the recipient's email account, I send it to myself here

my_user_name = 'ShadowNie' # Set the recipient's email nickname
```

```
# Define a function for sending email alerts

def mail(content, title):

    ret = True # Define a flag variable ret to record the event of sending email, with an initial value of True

    try:

        msg = MIMEText(content, 'plain', 'utf-8') # Create a message text object, with content as the email content, 'plain' as the text format, and 'utf-8' as the encoding

        '''Three header information: sender, recipient, subject'''

        msg['From'] = formataddr([my_name, my_sender]) # Define the sender information: the name and email address of the sender

        msg['To'] = formataddr([my_user_name, my_user]) # Define the recipient information: the name and email address of the recipient

        msg['Subject'] = title # Define the subject of the email
```

```
        server = smtplib.SMTP_SSL("smtp.qq.com", 465) # Create an SMTP service and connect to the qq email server. The SMTP protocol encryption port is 465.

        server.login(my_sender, my_pass) # Log in to the email account with the sender's email address and password

        server.sendmail(my_sender, my_user, msg.as_string()) # Send the email with the sender's email address, recipient's email address, and the message in string format

        server.quit() # Close the connection

    except Exception: # If the try statement does not execute, ret is set to False
```

```
        ret = False # Set the ret flag to False
    return ret # Return the ret flag

soil_mail_enable = True  # Define the flag for enabling soil alarm email sending
light_mail_enable = True  # Define the flag for enabling light alarm email sending

soil_mail_time = time.time()  # Flag for recording the scheduled time for sending soil alarm email
light_mail_time = time.time()  # Flag for recording the scheduled time for sending light alarm email

# Define callback function
def sub_cb(client, userdata, msg):
    global soil_mail_enable, light_mail_enable, soil_mail_time, light_mail_time
    topic = msg.topic # Store topic data in variable
    payload = msg.payload.decode() # Store payload message data as string in variable
    print("\nTopic: " + topic + " Message: " + payload) # Print data on the terminal

    if topic == IOT_pubTopic1:  # Define the operation when receiving Soil_moisture_value
        if payload.isdigit(): # If the received message is a number, it is a humidity value rather than a control command
            Soil_moisture_value = int(payload)  # Convert to a numerical type for easy judgment later
            text_value.config(text = str(Soil_moisture_value))  # Update the screen display data
            if Soil_moisture_value < 2120: # The threshold is set to 2120, which can be adjusted according to the actual situation
                text2.config(text="too low")  # Update the text warning
                if soil_mail_enable == True and (time.time()-soil_mail_time) > 10:  # Only send emails if allowed and 10 seconds have passed since the last email was sent
                    ret1=mail("Soil moisture alarm","The current soil moisture level is " + str(Soil_moisture_value) + ", please water in a timely manner!")
                    if ret1:  # If there is a send email event
```

```
                        print("The email has been sent successfully 1")
                        soil_mail_enable = False  # The email has been sent, change the flag to avoid duplicate sending
                        soil_mail_time = time.time()   # Record the time of this email sending
                    else: # Otherwise
                        print("The email failed to send 1")
            else:
                text2.config(text="appropriate")  # Update the text
                soil_mail_enable = True   # If the alarm is lifted, allow subsequent emails to be sent
    elif  topic == IOT_pubTopic2:  # Define the operation when receiving Light
        if payload.isdigit(): # If the received message is a number, it is a light value rather than a control command
            light_value = int(payload)  # Convert to a numerical type for easy judgment later
            text_value_2.config(text = str(light_value))  # Update the screen with the light value
            if light_value < 666:   # The threshold is set to 666, which can be adjusted according to the actual situation
                text4.config(text="too low")  # Update the text warning
                if light_mail_enable == True and  (time.time()-light_mail_time) > 10:  # Only send emails if allowed
                    ret2=mail("Light intensity alarm","The current light intensity value is " + str(light_value) + ", please provide additional light in a timely manner!")
                    if ret2:  # If there is a send email event
                        print("The email has been sent successfully 2")
                        light_mail_enable = False  # The email has been sent, change the flag to avoid duplicate sending
                        light_mail_time = time.time()  # Record the time of this email sending
                    else:  # Otherwise
                        print("The email failed to send 2")
            else:
                text4.config(text="appropriate") # Update the text warning
                light_mail_enable = True  # If the alarm is lifted, allow subsequent emails to be sent
```

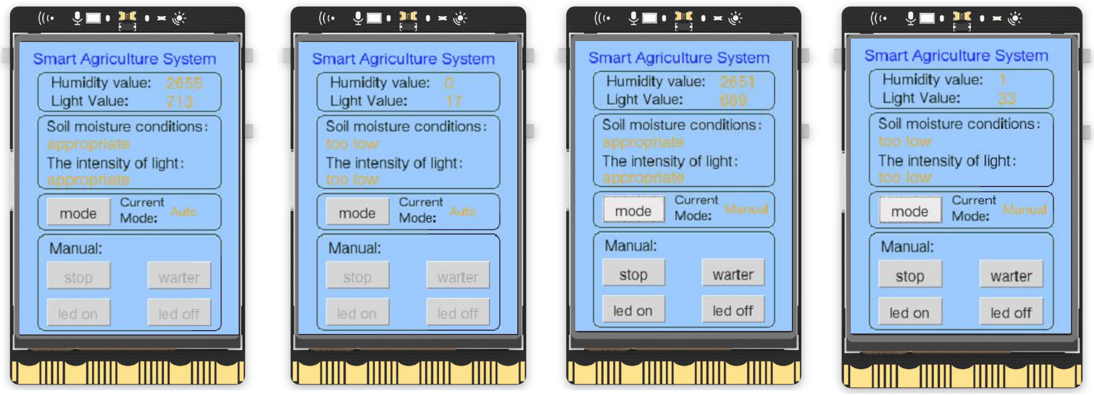
```
siot.init(CLIENT_ID, SERVER, user=IOT_UserName, password=IOT_PassWord) # Initialize and ve
rify the correctness of the input username and password.

siot.connect() # Connect to the siot IoT platform.

siot.subscribe(IOT_pubTopic1, sub_cb) # Subscribe to Topic1 for soil moisture data.

siot.subscribe(IOT_pubTopic2, sub_cb) # Subscribe to Topic2 for light intensity data.

siot.publish(IOT_pubTopic1, 'auto') # Publish the message 'auto' to the IoT platform.

siot.publish(IOT_pubTopic2, 'auto') # Publish the message 'auto' to the IoT platform.

siot.loop() # Keep the program running in a loop.


while True: # Keep the program running in a loop.

    time.sleep(0.5) # Delay for 0.5 seconds.
```

Tips: Please provide actual sender account and authorization code, these information cannot be empty.

3. Running the Program

STEP 1: Run and observe the results

Click the "Run" button on the Mind+ software and observe UNIHIKER 3. The initial mode is set to "Auto", and at this point, the four buttons for watering, stopping water, turning on the light, and turning off the light are grayed out and cannot be clicked. After clicking the mode switch button, the buttons return to normal state, and the detected soil humidity and light values are also displayed on the screen. Then, when you click on "start" (watering) and "on" (turning on the light) on UNIHIKER 3, the water pump on UNIHIKER 1 starts to work, and the LED on UNIHIKER 2 lights up. Finally, when you click on "stop" (stopping water) and "off" (turning off the light), the water pump and LED stop working.

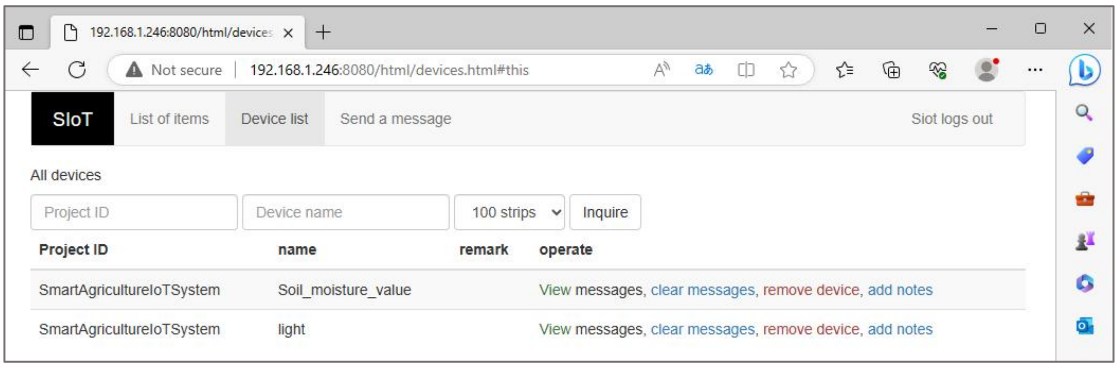

Smart Agriculture System
Humidity value: 2655
Light Value: 713
Soil moisture conditions: appropriate
The intensity of light: appropriate
Current Mode: Auto

Smart Agriculture System
Humidity value: 0
Light Value: 17
Soil moisture conditions: too low
The intensity of light: too low
Current Mode: Auto

Smart Agriculture System
Humidity value: 2651
Light Value: 689
Soil moisture conditions: appropriate
The intensity of light: appropriate
Current Mode: Manual

Smart Agriculture System
Humidity value: 1
Light Value: 33
Soil moisture conditions: too low
The intensity of light: too low
Current Mode: Manual

Observing the IoT platform, it can be seen that data is being transmitted on two different topics.

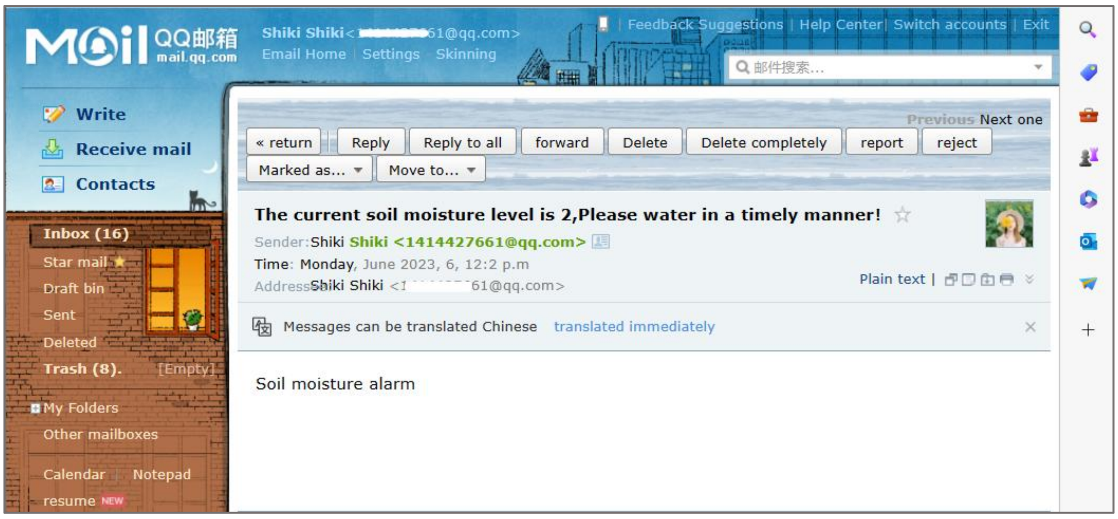

192.168.1.246:8080/html/devices
Not secure | 192.168.1.246:8080/html/devices.html#this
SIoT
List of items
Device list
Send a message
SIot logs out

All devices

Project ID | Device name | 100 strips | Inquire

Project ID	name	remark	operate
SmartAgricultureIoTSystem	Soil_moisture_value		View messages, clear messages, remove device, add notes
SmartAgricultureIoTSystem	light		View messages, clear messages, remove device, add notes

Similarly, we will receive email notifications from QQ Mail when the soil moisture or light intensity data is inadequate.



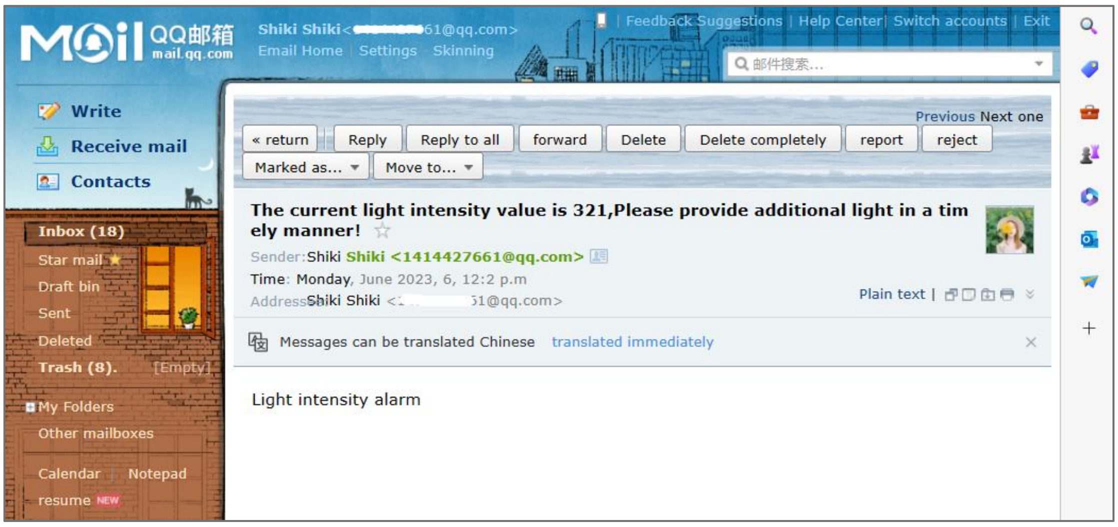
The current soil moisture level is 2,Please water in a timely manner!

Sender: Shiki **Shiki <1414427661@qq.com>**

Time: Monday, June 2023, 6, 12:2 p.m

Address Shiki Shiki <1 61@qq.com>

Soil moisture alarm



The current light intensity value is 321,Please provide additional light in a timely manner!

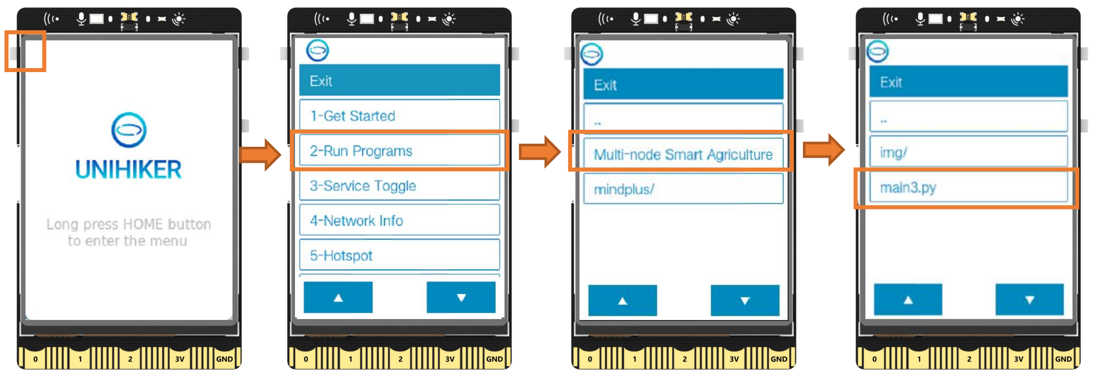
Sender: Shiki **Shiki <1414427661@qq.com>**

Time: Monday, June 2023, 6, 12:2 p.m

Address Shiki Shiki < 51@qq.com>

Light intensity alarm

Tips: Here, since we have directly created the Python program file in the memory of the Row Free board, we can also run the program directly on the board. Taking UNIHIKER 3 as an example, the operation method is as follows:



Tips: To ensure the ability to send emails, it's necessary to make sure that the connected router or mobile hotspot is properly connected to the network.

Challenge Yourself

Think about which of the three ways of supplying power to the UNIHIKER (power adapter, power bank, computer USB port) is more suitable for scenarios that require long-term program operation, and which is more suitable for outdoor use scenarios?

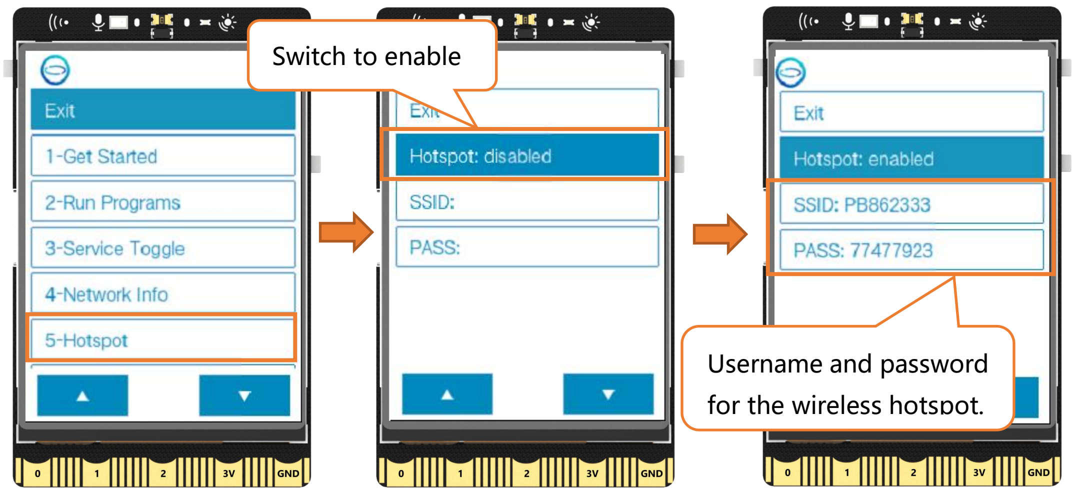
In addition to email alerts, what other feasible remote notification methods are there?

In a scenario where there is neither a router nor a mobile hotspot, can we still build a multi-node IoT system?

Tips: See Appendix for the answers.

Appendix

In a scenario where there is neither a router nor a mobile hotspot, we can still create a Wi-Fi hotspot by enabling the hotspot function of one of the UNIHIKER, and other UNIHIKER can connect to this Wi-Fi hotspot to connect to the same local area network. The method is as follows.


Exit
1-Get Started
2-Run Programs
3-Service Toggle
4-Network Info
5-Hotspot
Switch to enable
Exit
Hotspot: disabled
SSID:
PASS:
Exit
Hotspot: enabled
SSID: PB862333
PASS: 77477923
Username and password for the wireless hotspot.
0
1
2
3V
GND