

Lesson 2 Little Light Catcher

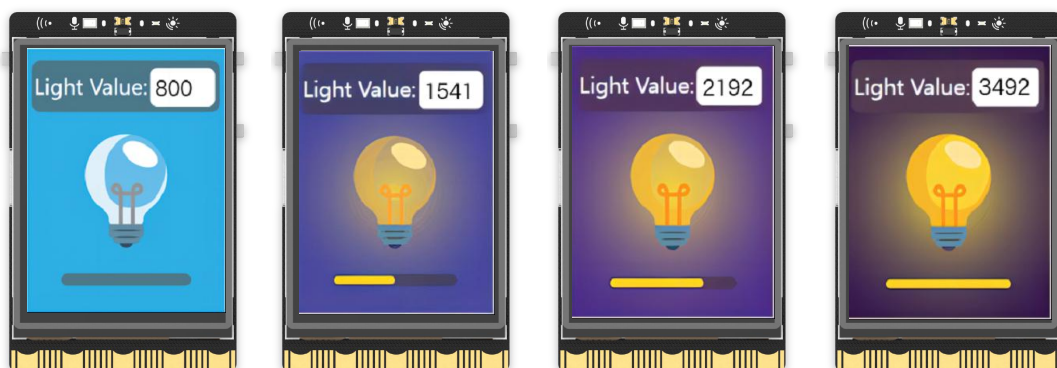
In daily life, the intensity of light has a great impact on human emotions. Whether it is positive or negative emotions, brighter light can intensify these feelings. Therefore, functional home environments such as study rooms and kitchens often use strong lighting, which can help people concentrate and be more productive. Bedrooms, on the other hand, are suitable for softer lighting, which creates a warm and comfortable feeling and helps to relax the mind and body. However, it is difficult to accurately judge the intensity of light based solely on subjective feelings. So how can we determine the size of the ambient light and then judge whether the lighting in each room of the home is reasonable?

Let's use the UNIHAKER to create a tiny light meter to better understand the ambient light intensity in our rooms!



Task Objectives

The detected light values through the light sensor will be displayed on the screen. Additionally, the strength of the light will be divided into four levels, and the corresponding brightness level of the small light background image will be displayed in different environments with varying light intensity.





Knowledge points

1. Understanding Light Sensors
2. Understanding the Pinpong Library
3. Learning how to use the Pinpong Library to detect light values
4. Learning how to use the Unihiker Library to display images, switch between text and images.

Material List

Hardware List:

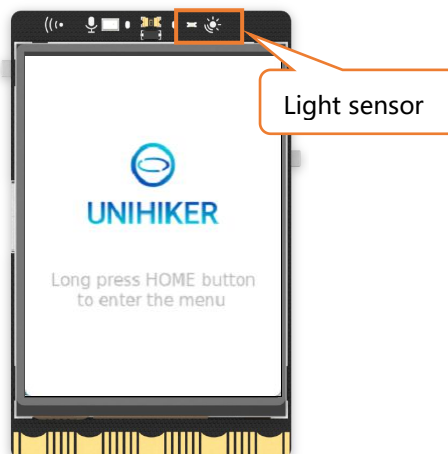
	
UNIHAKER x1	Type-C & Micro Dual-Use USB Cable x1

Software Preparation: Mind+ Programming Softwarex1

Knowledge background

1. What is a light sensor?

A light sensor is a device that can convert light signals into electrical signals, which can be used to detect the value of light. The light sensor is built into the UNIHAKER and returns an analog value when detecting light, with a numerical range of 0-4095. The brighter the light, the higher the numerical value, and the darker the light, the lower the numerical value.



2. What is Pinpong Library

Through the Mind+ software, we can connect the UNIHAKER to the computer, but how can we control the components on the UNIHAKER to run? This is where we need to use the Pinpong Library. The Pinpong Library is a Python library for controlling open-source hardware mainboards based on the Firmata protocol. It takes only five minutes to get started with using Python to control open-source hardware. The principle is to burn a specific firmware onto the open-source hardware, so that the open-source hardware can communicate with the computer through the serial port and execute various commands.

The name of the Pinpong Library is composed of "Pin" and "Pong". "Pin" refers to the pin, and "Pinpong" is a homophonic word of "Ping Pong", which refers to the back and forth of the signal.

The design of the Pinpong Library is to free developers from being restricted by complex hardware models during the development process and to focus on the implementation of software. Even if the program is developed using other boards in the initial stage, it can be modified to run on the UNIHAKER by simply modifying the hardware parameters, achieving the goal of "writing once, running everywhere".

So, can all the hardware on the UNIHAKER be controlled through the Pinpong Library?

In fact, the UNIHAKER is composed of a single board computer and a microcontroller system. The microcontroller system is similar to an Arduino board, and the onboard gold fingers, Gravity interface, I2C interface, buttons, buzzer, light sensor, accelerometer and gyroscope are all connected to the microcontroller. When in use, they can be controlled through the Pinpong Library. The single-board computer can be regarded as a small computer, and the touch screen, USB port, microphone, home button, etc. on the UNIHAKER can all be controlled through it.

3. The method of importing the Pinpong library and initializing the board

To control the UNIHAKER using the Pinpong library, we need to initialize the board first. To do this, we need to import the Board class from the "pinpong.board" package provided by the Pinpong library. Similarly, to control the various components on the UNIHAKER, we need to import the relevant modules from the "pinpong.extension.unihiker" package.

```
from pinpong.board import Board # Import the Board module from the pinpong.board package
from pinpong.extension.unihiker import * # Import all modules from the pinpong.extension.unihiker package

Board('UNIHAKER').begin() # Initialize the board by selecting the board type and port number;
if not specified, the program will automatically detect it
```

In the code snippet, the main control board model 'UNIHAKER' is specified in the argument of Board(""), and the port number is specified in the begin() function. If the model and port number are not specified, the program will automatically recognize them.

4. The instruction 'light.read()' in the Pinpong library reads the light value

"After initializing the board, we can use the 'light.read()' command in the Pinpong library to read the light value."

```
Light = light.read() # Read the light value
```

Specifically, the command "light.read()" is used to read the light value, with "Light" being a variable used to store the value read.

Tips: The Pinpong library encapsulates the functions of the onboard components in their respective classes in the "pinpong.extension.unihiker" file. Since the classes have already been instantiated in the file, a "light" object has been generated. Therefore, we can directly use "light.read()" to read the light value.

5. The draw_image() method of the Unihiker library's

GUI class can display images on the screen of the UNIHAKER. When programming, this functionality is achieved through the format "object.method()", and the method returns an image object that can be stored in a variable for subsequent object update operations.

```
img = gui.draw_image(x=0, y=0, w=240, h=320, image='light-1.png') # Display the initial back  
ground image 'light-1'
```

The parameters w and h in the method represent the width and height of the image, respectively. If omitted, the image will be scaled proportionally based on the smaller of the two dimensions. If both width and height parameters are omitted, the original resolution of the image will be maintained. Here, we will set the width and height to match the screen. The "image" parameter corresponds to the image source, which can be a path or an image object. "img" is a variable used to store the image object.

Tips: When it is necessary to set the position of the image, the horizontal and vertical coordinates can be determined by fixing the values of x and y. When the position is (0,0), it can be omitted.

6. General Knowledge and Functionality of Unihiker Library's GUI Class

(1) Updating Control Object Configurations

All the elements displayed on the screen of the UNIHAKER can be referred to as controls, such as images, text, emoticons, buttons, etc. Once generated, control objects can be modified by using the "object.config()" method.

```
img = gui.draw_image(x=0, y=0, w=240, h=320, image='light-1.png') # Display the initial bac  
kground image 'light-1'  
  
img.config(image='light-2.png') # Switch the background image to 'light-2'
```

Here, we first displayed an initial background image 'light-1.png' and stored it as a image control in the variable 'img'. Then, we updated the control object by using the 'config' method to replace the background image with 'light-2.png'.

```
value = gui.draw_text(x=155, y=30, text='0', font_size=18) # Display the initial light value  
  
value.config(text='666') # Update the displayed light value
```

Here, we first displayed a line of text at the coordinate position (153, 28) with the content "800", and stored it as a text control in the variable 'value'. Then, we updated the control object by using the 'config' method to replace the text content with "666".

Hands-on practice

Task Description 1: Display a fixed light value

Display the light value of 800 on the screen, while showing a background image of a small lamp.

1. Hardware setup

STEP 1: Connect the UNIHKER to the computer via a USB cable.



2. program coding

STEP 1: Create and Save Project File

Launch Mind+, save the project as "002 Little Light Catcher".

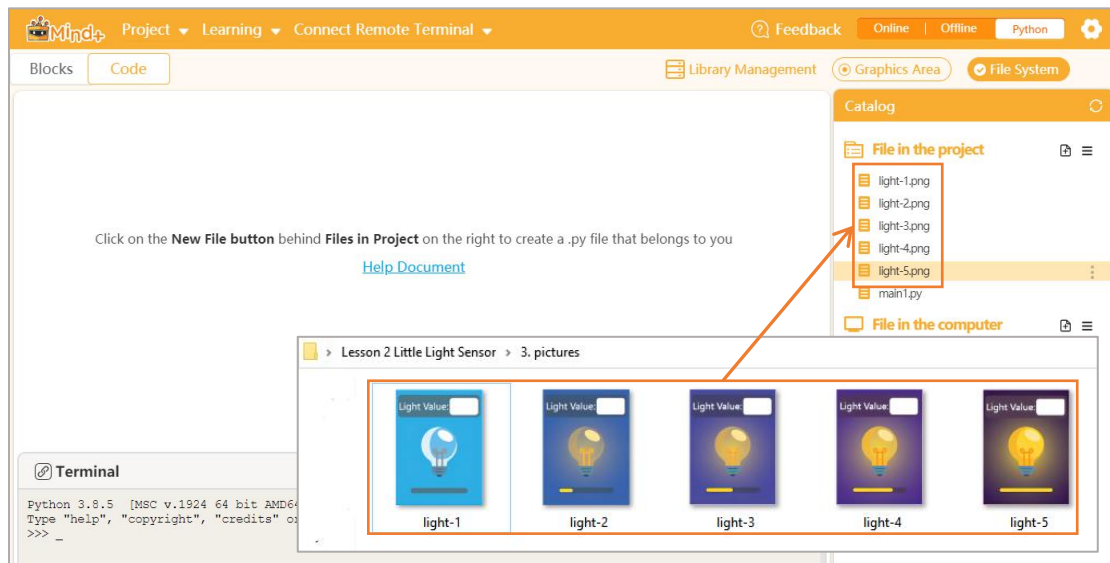
STEP2: Create and Save Python File

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

STEP 3: Importing Images

To import the required PNG images into the project folder, follow these steps:

(1) Drag and drop the images from the 'pictures' files into the project folder one by one.



STEP 4: Programming

(1) Import necessary libraries

In this task, we need to use the GUI module in the Unihiker library to display images and text on the screen. Therefore, we need to import it first. The code is as follows. Also, since we need a certain delay in the program to keep it running, we also need to import the time library.

```
from unihiker import GUI # Import the unihiker library
import time # Import the time library
```

(2) Instantiate the GUI class

Before using the GUI module in the Unihiker library to display text and images, we need to instantiate the GUI class to create an object for using various methods in the class.

```
gui = GUI() # Instantiate the GUI class and create a gui object
```

(3) Display image

After creating the gui object, we can add a background image to the blank screen.

```
img = gui.draw_image(x=0, y=0, w=240, h=320, image='light-1.png') # Display the initial background image 'light-1'
```

(4) Displaying the Light Value

Afterwards, we assigned a specific light value of "800" and displayed it on the screen.

```
value = gui.draw_text(x=155, y=30, text='888', font_size=18) # Display the initial light value
```

(5) Maintaining Display of Content

In order to ensure that the screen content is displayed for a long time, we keep the program running at the end, thus maintaining the display of content.

```
while True: # Loop  
    time.sleep(1) # Delay for 1 second
```

Tips: The complete example program is as follows:

```
from unihiker import GUI # Import the unihiker library  
import time # Import the time library  
  
gui = GUI() # Instantiate a GUI object by creating an instance of the GUI class  
  
img = gui.draw_image(x=0, y=0, w=240, h=320, image='light-1.png') # Display the initial background image 'light-1'  
value = gui.draw_text(x=155, y=30, text='800', font_size=18) # Display the default light value  
  
while True: # Loop  
    time.sleep(1) # Delay for 1 second
```

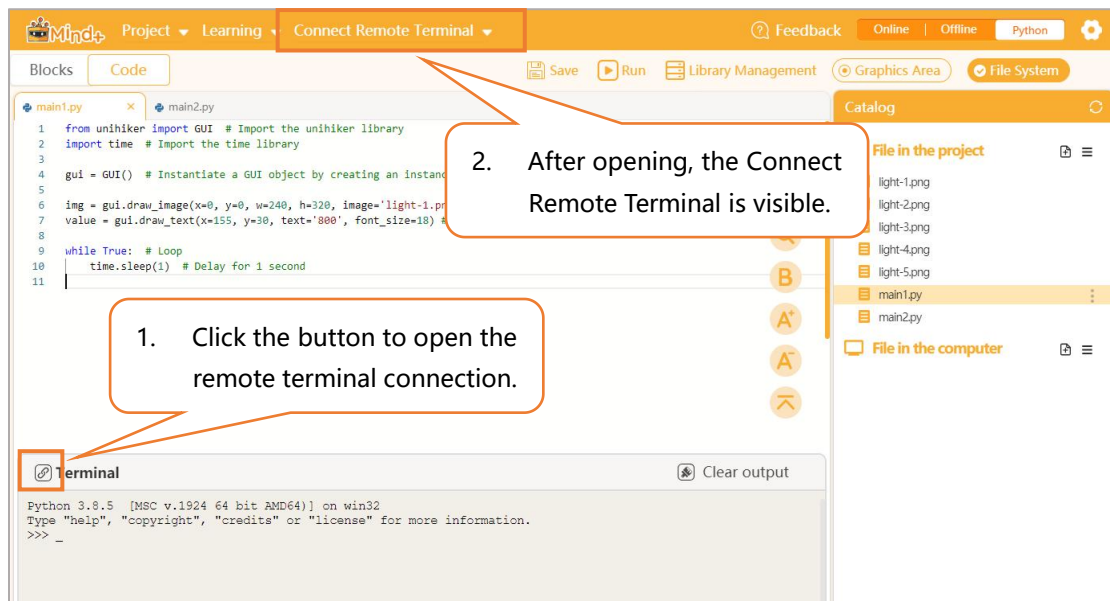
3. Running the Program

STEP 1: Remote connection to the UNIHKER

- (1) Confirm that the UNIHKER is connected and powered on.

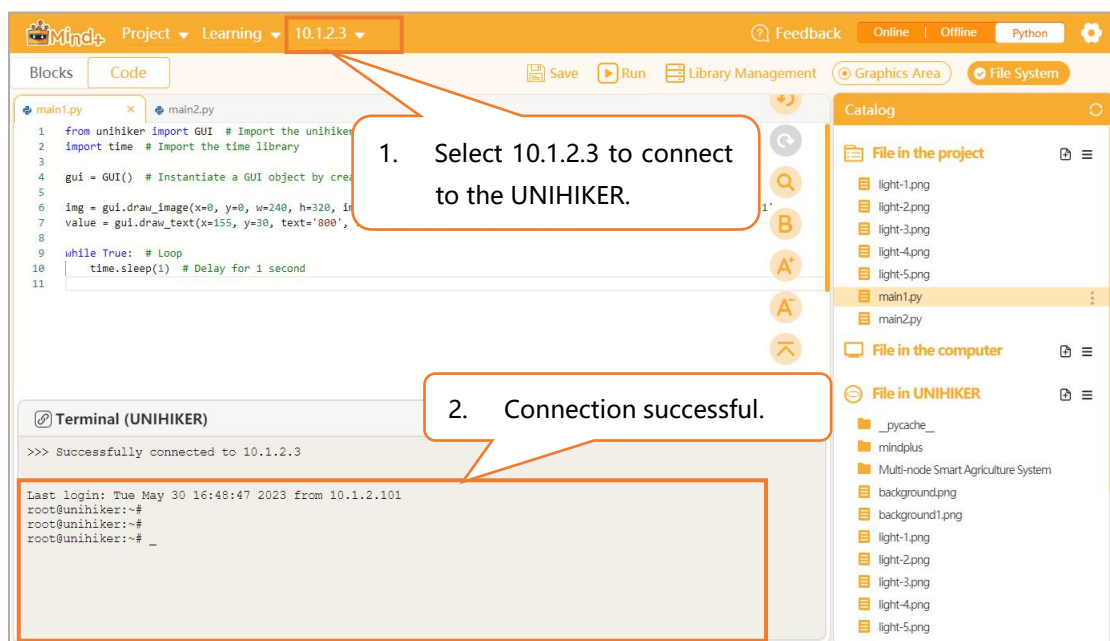


- (2) Open the remote connection terminal.

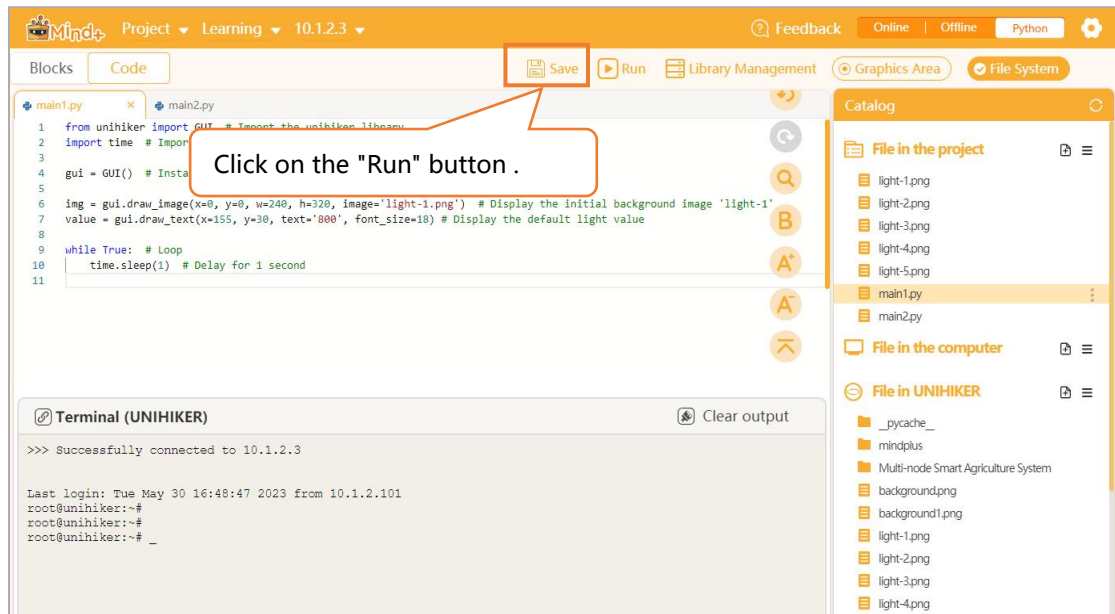


(3) Connect to the UNIHAKER.

Tips: "10.1.2.3" is the fixed IP address used for the UNIHAKER board when connected directly to a computer via USB cable.

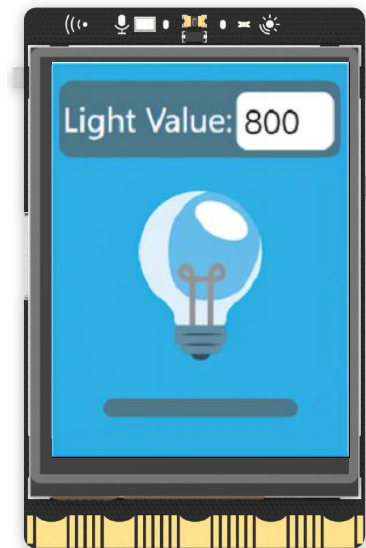


STEP 2: Click on the "Run" button in the upper right corner.



STEP 3: Observe the result.

Observing the blank space, a blue LED can be seen displayed in the center of the screen, with a value of 800 displayed in the light value bar.



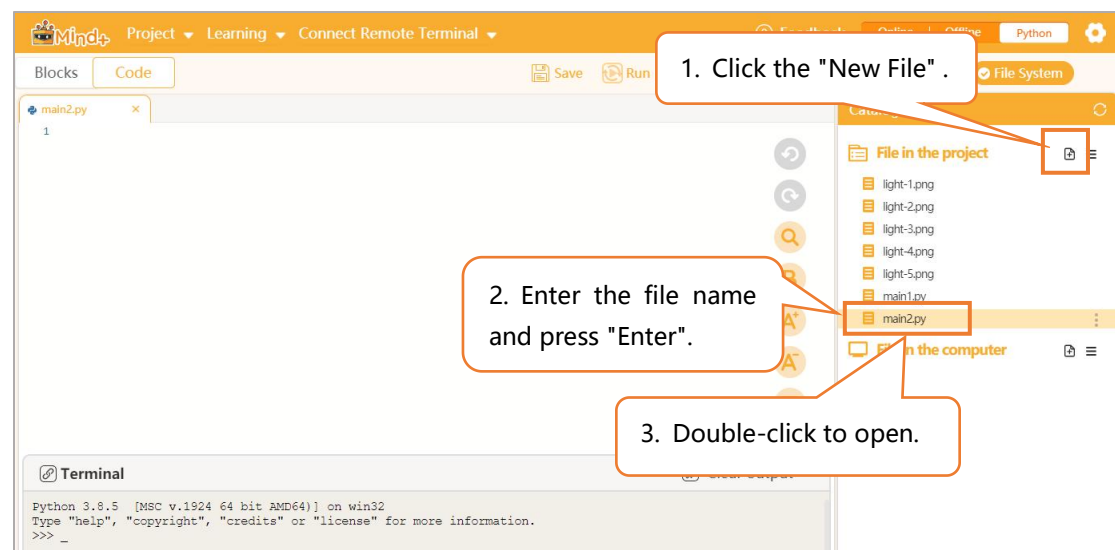
Task Description 2: Real-time detection of light values

Real-time detection of light values through the onboard light sensor, displayed on the screen, and the light values are evenly divided into four different levels, and different brightness levels of small light pictures are displayed accordingly.

1. program coding

STEP 1: Create and Save Project Files

Previously, we have already created and saved a project file. Here, we only need to create a new Python program file named "main2.py" and save it. The specific operation method is the same as before. After creating it, we double-click to open it for programming.



STEP 2: Programming

(1) Import necessary libraries

We need to import the necessary libraries to detect light values through the onboard light sensor. Therefore, we will import the Pinpong libraries and initialize the UNIHAKER.

```
from unihiker import GUI # Import the unihiker library
import time # Import the time library

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

Board().begin() # Initialize the board by selecting the board type and port number; if not specified, the program will automatically detect it
```

(2) Instantiating the GUI Class and Displaying the Background Image and Initial Light Value on Screen

```
gui = GUI() # Instantiate the GUI class and create a gui object

img = gui.draw_image(x=0, y=0, w=240, h=320, image='light-1.png') # Display the initial background image 'light-1'

value = gui.draw_text(x=155, y=30, text='0', font_size=18) # Display the initial light value
```

- (3) Loop through to detect the light value and update the numerical display and image accordingly.

In the previous task, we stored the light value in the variable "Light" and assigned it a value of 800. In this task, we will detect the light value using the onboard light sensor and assign it to the variable "Light". To continuously monitor and update the light value on the screen every second, we will use a while loop.

```
while True:
    Light = light.read() # Read the light value
    value.config(text=Light) # Update the displayed light value

    # Determine the light level
    if 0 <= Light < 1024:
        img.config(image='light-2.png') # Switch the background image to 'light-2'
    elif 1024 <= Light < 2048:
        img.config(image='light-3.png') # Switch the background image to 'light-3'
    elif 2048 <= Light < 3072:
        img.config(image='light-4.png') # Switch the background image to 'light-4'
    else:
        img.config(image='light-5.png') # Switch the background image to 'light-5'

    time.sleep(1) # Delay for 1 second to keep the screen content displayed for a longer time
```

Tips: The complete example program is as follows:

```
from unihiker import GUI # Import the unihiker library
import time # Import the time library

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

Board().begin() # Initialize the board by selecting the board type and port number; if not specified, the program will automatically detect it

gui = GUI() # Instantiate the GUI class and create a gui object

img = gui.draw_image(x=0, y=0, w=240, h=320, image='light-1.png') # Display the initial bac
```

```

kground image 'light-1'
value = gui.draw_text(x=155, y=30, text='0', font_size=18) # Display the initial light value

while True:
    Light = light.read() # Read the light value
    value.config(text=Light) # Update the displayed light value

    # Determine the light level
    if 0 <= Light < 1024:
        img.config(image='light-2.png') # Switch the background image to 'light-2'
    elif 1024 <= Light < 2048:
        img.config(image='light-3.png') # Switch the background image to 'light-3'
    elif 2048 <= Light < 3072:
        img.config(image='light-4.png') # Switch the background image to 'light-4'
    else:
        img.config(image='light-5.png') # Switch the background image to 'light-5'

    time.sleep(1) # Delay for 1 second to keep the screen content displayed for a longer time

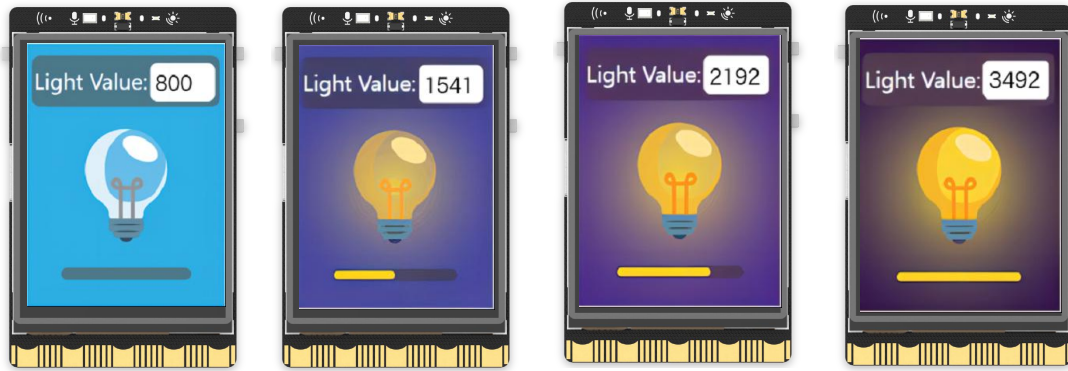
```

2. Running the Program

STEP 1: Remote connection to the UNIHAKER

STEP 2: Observe the result.

After clicking "Run", observe the UNIHAKER board and shine light on the light sensor from different angles. You will notice that the light value displayed at the top of the screen is constantly changing at regular intervals, while the brightness of the LED and the filling of the progress bar change accordingly. The stronger the light, the higher the numerical value, the brighter the LED, and the higher the filling of the progress bar.



Challenge Yourself

Consider adding a notification feature to display corresponding text information on the screen when the light value is too high (>3072) or too low (<1024). You can try it yourself!

Further reading

1. The principle of a light sensor

The principle of a light sensor is based on the interaction of two components, namely the emitter and the receiver. The emitter focuses the light through a lens and transmits it to the receiver's lens, where it is converted into an electrical signal by the receiving sensor. This electrical signal can then be used for various switching and control actions. The basic principle is to use the signal obtained by blocking the interaction between the emitter and receiver to achieve various types of automated control.