# Basic Project

## Project 1 Button controls led

### Description

This project is a beginner-friendly introduction to Arduino programming and basic electronics. It demonstrates how to control an LED using a pushbutton switch. When the button is pressed, the LED turns on; when released, the LED turns off. The project helps beginners understand digital input and output, as well as how to interface hardware components with the Arduino Uno R4.

### How-To Guide

#### 1. Gather Components:

* 1 x Arduino Uno R4 Minima
* 1 x Breadboard
* 1 x LED
* 1 x Pushbutton
* 1 x Resistors (220-330 ohms for the LED, 10k ohms for the pushbutton)
* 6 x Jumper wires

#### 2. Assemble the Circuit:

* Connect the components on the breadboard as shown in the circuit diagram:
* Connect one leg of the pushbutton to digital pin 2 on the Arduino Uno R4.
* Connect the other leg of the pushbutton to the ground (GND) rail on the breadboard.
* Connect one leg of the resistor (10k ohms) to the same leg of the pushbutton connected to the ground.
* Connect the other leg of the resistor to the positive rail on the breadboard (5V).
* Connect the positive (longer) leg of the LED to digital pin 13 on the Arduino Uno R4.
* Connect the negative (shorter) leg of the LED to a resistor (220-330 ohms), then connect the other leg of the resistor to the ground (GND) rail on the breadboard.

Figure of wiring diagram



#### 3. Write the Arduino Code:

Open the Arduino IDE on your computer.

##### Steps

* **Setup the Circuit:** Connect the components as shown in the circuit diagram.
* **Code the Arduino:** Write the code to control the LED with the button. Here's a simple example:

const int buttonPin = 7; // Pin connected to the pushbutton

const int ledPin = 2;   // Pin connected to the LED

int buttonState = 0;     // Variable for storing the button's state

void setup() {

  pinMode(ledPin, OUTPUT);    // Set the LED pin as an output

  pinMode(buttonPin, INPUT);  // Set the button pin as an input

}

void loop() {

  buttonState = digitalRead(buttonPin); // Read the state of the button

  if (buttonState == HIGH) { // If button is pressed

    digitalWrite(ledPin, HIGH); // Turn on the LED

  } else {

    digitalWrite(ledPin, LOW);  // Turn off the LED

  }

}

Upload the code to your Arduino Uno R4 by clicking the upload button in the IDE.

#### 4. Understand the Code:

The code initializes two variables **buttonPin** and **ledPin** to store the pin numbers connected to the pushbutton and LED, respectively.

In the **setup()** function, it sets the pin modes: **ledPin** as an **output** and **buttonPin** as an **input**.

The **loop()** function continuously checks the state of the button using:

**digitalRead(buttonPin)**.

If the button state is **HIGH (pressed)**, it turns on the LED by setting **ledPin** to **HIGH**.

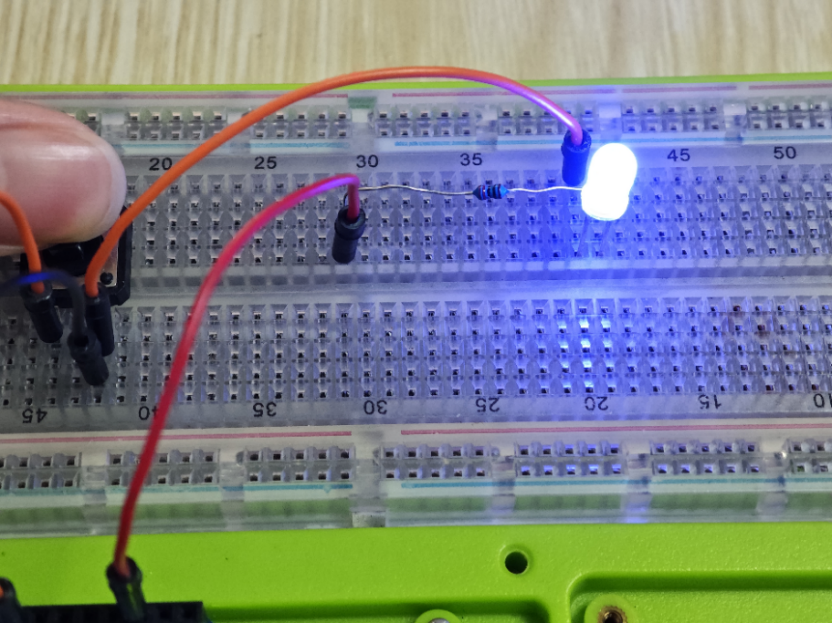
If the button state is **LOW (released)**, it turns off the LED by setting **ledPin** to **LOW**.

#### 5. Test the Project:

Once the code is uploaded, press the **pushbutton**.

The LED should turn on while the button is pressed and turn off when released.

If it doesn't work as expected, double-check the connections and code.



#### 6. Experiment and Learn:

Experiment with the code by modifying it to achieve different behaviors.

Try adding more LEDs and buttons to create more complex interactions.

Explore other Arduino functions and components to expand your knowledge.

## Project 2 Read data from temperature & humidity sensor

### Description

This project aims to read data from a temperature and humidity sensor using an Arduino Uno R4. The DHT11 (temperature & humidity) sensor is commonly used for this purpose due to its simplicity and accuracy. The sensor provides digital output, making it easy to interface with the Arduino. By reading temperature and humidity data, you can monitor environmental conditions in various applications, such as home automation, weather stations, and greenhouse monitoring systems.

### How-To Guide

#### 1. Gather Components:

* 1 x Arduino Uno R4
* 1 x Breadboard
* 1 x DHT11 Temperature and Humidity Sensor
* 3 x Jumper wires

#### 2. Assemble the Circuit:

Connect the components on the breadboard as per the provided circuit diagram.

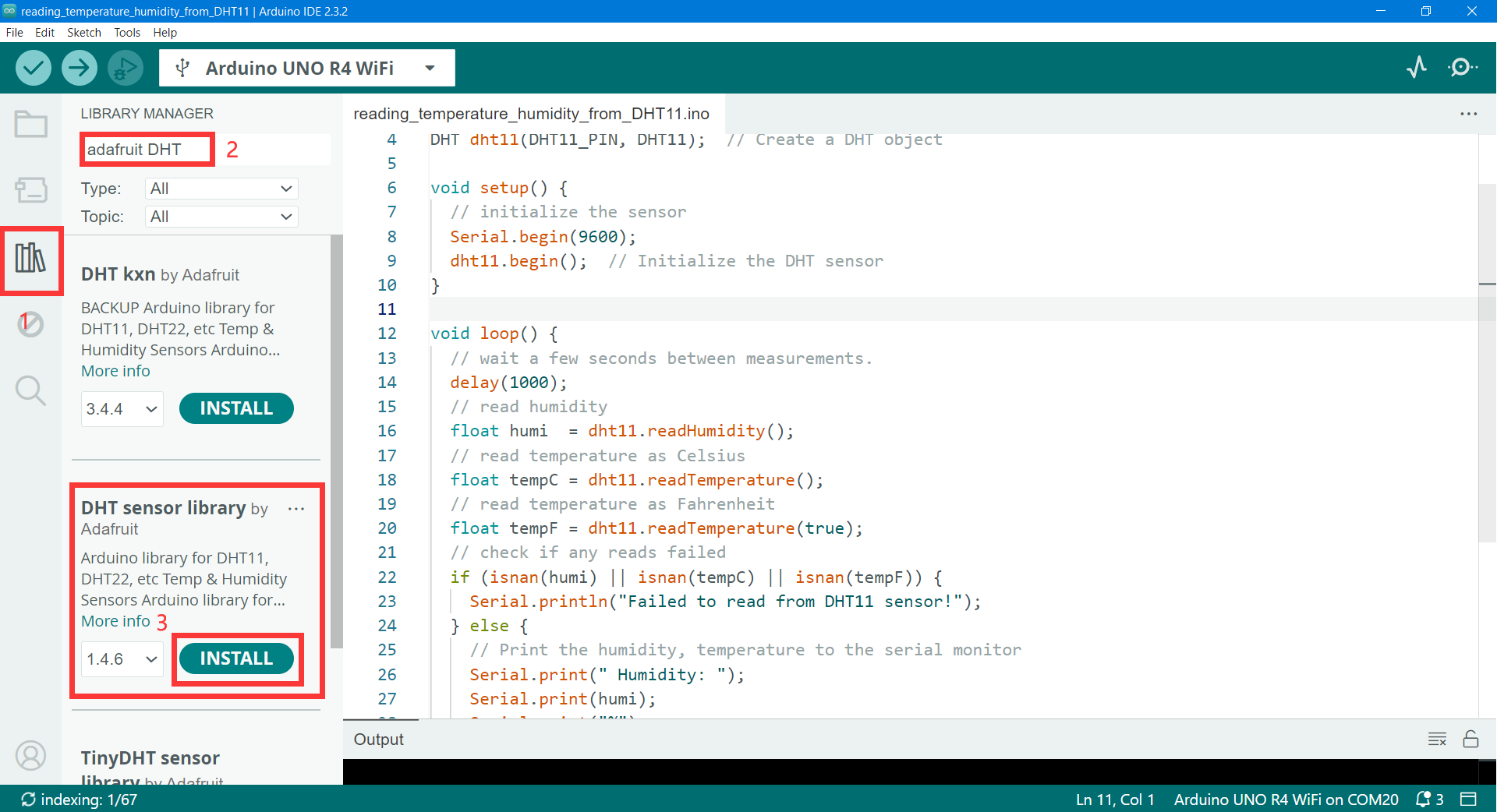
|  |  |
| --- | --- |
| Arduino UNO R4 minima | Temperature and humidity sensor |
| 3.3V/5V | VCC |
| GND | GND |
| Pin 2 | DATA |

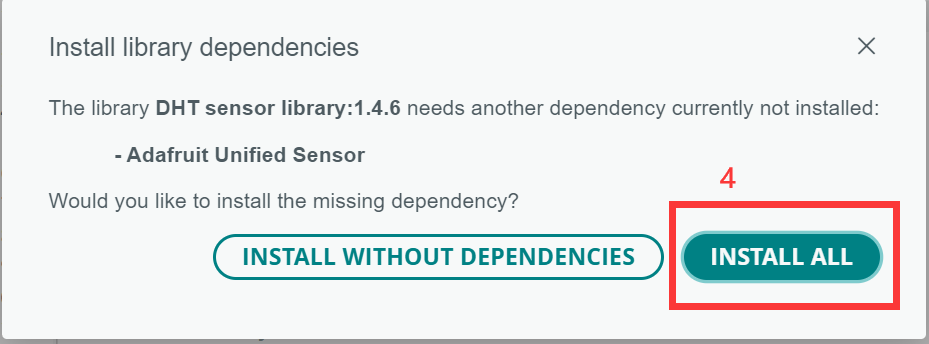
Ensure the connections are secure and there are no loose wires.

#### 3. Install Library:

Open your Arduino IDE on your computer.

Go to Sketch -> Include Library -> Manage Libraries.





Search for "DHT" and install the library.

This library provides functions to easily communicate with the DHT sensors.

#### 4. Write the Arduino Code:

Copy the provided Arduino code into your Arduino IDE.

#include "DHT.h"

#define DHT11\_PIN 2 // Define the pin used to connect the sensor

DHT dht11(DHT11\_PIN, DHT11);  // Create a DHT object

void setup() {

  // initialize the sensor

  Serial.begin(9600);

  dht11.begin();

}

void loop() {

  // wait a few seconds between measurements.

  delay(1000);

  // read humidity

  float humi  = dht11.readHumidity();

  // read temperature as Celsius

  float tempC = dht11.readTemperature();

  // read temperature as Fahrenheit

  float tempF = dht11.readTemperature(true);

  // check if any reads failed

  if (isnan(humi) || isnan(tempC) || isnan(tempF)) {

    Serial.println("Failed to read from DHT11 sensor!");

  } else {

    Serial.print(" Humidity: ");

    Serial.print(humi);

    Serial.print("%");

    Serial.print("    ");

    Serial.print("Temperature: ");

    Serial.print(tempC);

    Serial.print("°C");

    Serial.print("    ");

    Serial.print("Fahrenheit: ");

    Serial.print(tempF);

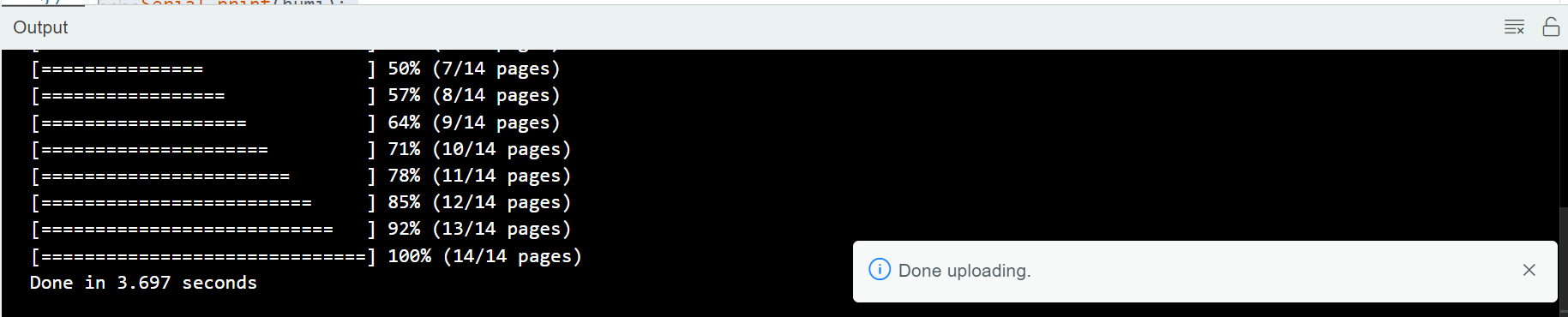
    Serial.println("°F");

  }

}

#### 5. Upload the Code:

* Connect your Arduino Uno R4 Minima to your computer via USB.
* Select the correct board and port in the Arduino IDE.
* Click the upload button in the IDE to upload the code to your Arduino.

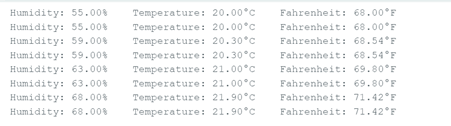


#### 6. Monitor the Data:

* Open the Serial Monitor in the Arduino IDE (Tools -> Serial Monitor).



You should see temperature and humidity readings printed every few seconds.



Verify that the readings change when you blow on the sensor or change the surrounding environment.

#### 7. Experiment and Learn:

Experiment with different sensors and sensor placements to monitor temperature and humidity in various environments.

Explore advanced features of the DHT sensor library and implement additional functionalities, such as data logging or setting thresholds for alerts.

## Project 3 Read data from the Rain Drop Sensor

### Description

This project focuses on reading data from a raindrop sensor using an Arduino Uno R4 minima. The raindrop sensor detects the presence of water or raindrops on its surface and provides an analog output signal that varies based on the amount of water detected. By interfacing the raindrop sensor with an Arduino, you can monitor rainfall or detect the presence of water in various applications, such as weather stations, automated irrigation systems, and leak detection systems.

### How-To Guide

#### 1. Gather Components:

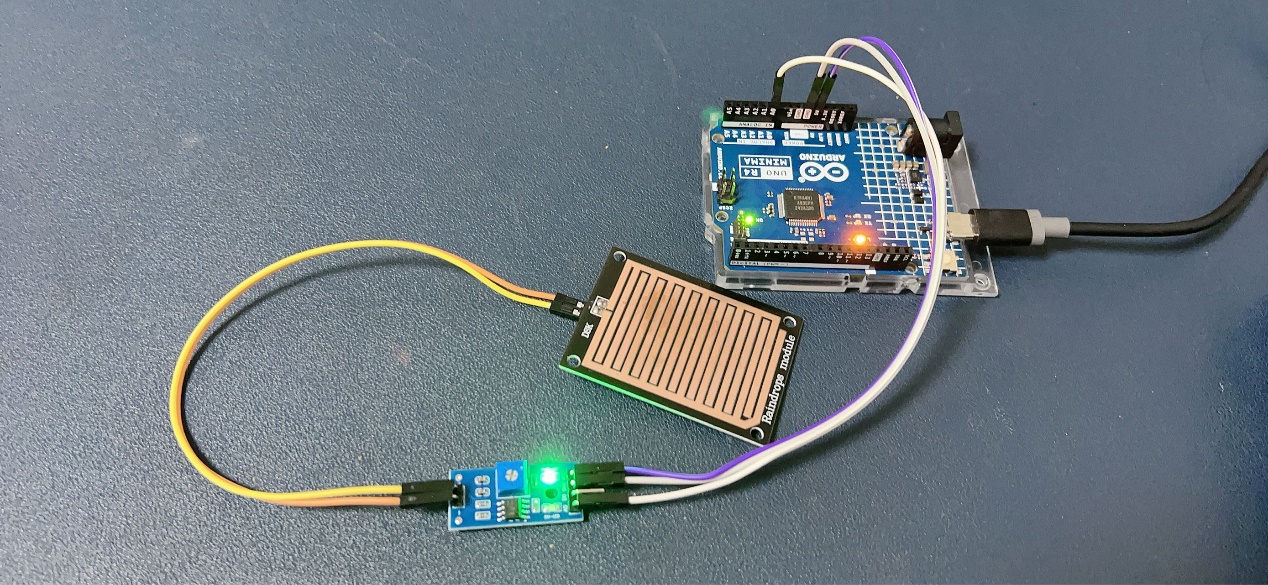
* 1 x Arduino Uno R4 Minima
* 1 x Breadboard
* 1 x Raindrop Sensor Module
* 1 x Buzzer
* 8 x Jumper wires

#### 2. Assemble the Circuit:

* Connect the raindrop sensor module to the Arduino Uno R4 as follows:
* Connect the VCC pin of the raindrop sensor to the 5V pin on the Arduino.
* Connect the GND pin of the raindrop sensor to the GND pin on the Arduino.
* Connect the A0 pin of the raindrop sensor to an analog input pin on the Arduino (e.g., A0).

|  |  |
| --- | --- |
| Arduino UNO R4 minima | Raindrop Sensor Module |
| 5V | VCC |
| GND | GND |
| 7 | DO |
| NC | AO |

|  |  |
| --- | --- |
| Arduino UNO R4 minima | buzzer |
| 8 | + |
| GND | GND |



#### 3. Write the Arduino Code:

Here's a basic example code to read data from the raindrop sensor and print it to the serial monitor:

const int raindropPin = A0; // Analog pin connected to the raindrop sensor

void setup() {

  Serial.begin(9600); // Initialize serial communication

}

void loop() {

  int raindropValue = analogRead(raindropPin); // Read analog value from the sensor

  Serial.print("Raindrop Value: ");

  Serial.println(raindropValue); // Print the value to the serial monitor

  delay(1000); // Delay for stability

}

#### 4. Upload the Code:

* Connect your Arduino Uno R4 to your computer via USB.
* Open the Arduino IDE, copy the provided code, and paste it into a new sketch.
* Select the correct board and port in the Arduino IDE.
* Click the upload button to upload the code to your Arduino.

##### Code Explanation:

The code initializes a variable **raindropPin** to store the analog pin connected to the raindrop sensor.

In the **setup()** function, it initializes serial communication at a baud rate of **9600**.

The **loop()** function continuously reads analog values from the raindrop sensor using **analogRead(raindropPin).**

It prints the analog values to the serial monitor, allowing you to monitor the sensor readings in real-time.

The **delay(1000)** function adds a 1-second delay between readings for stability.

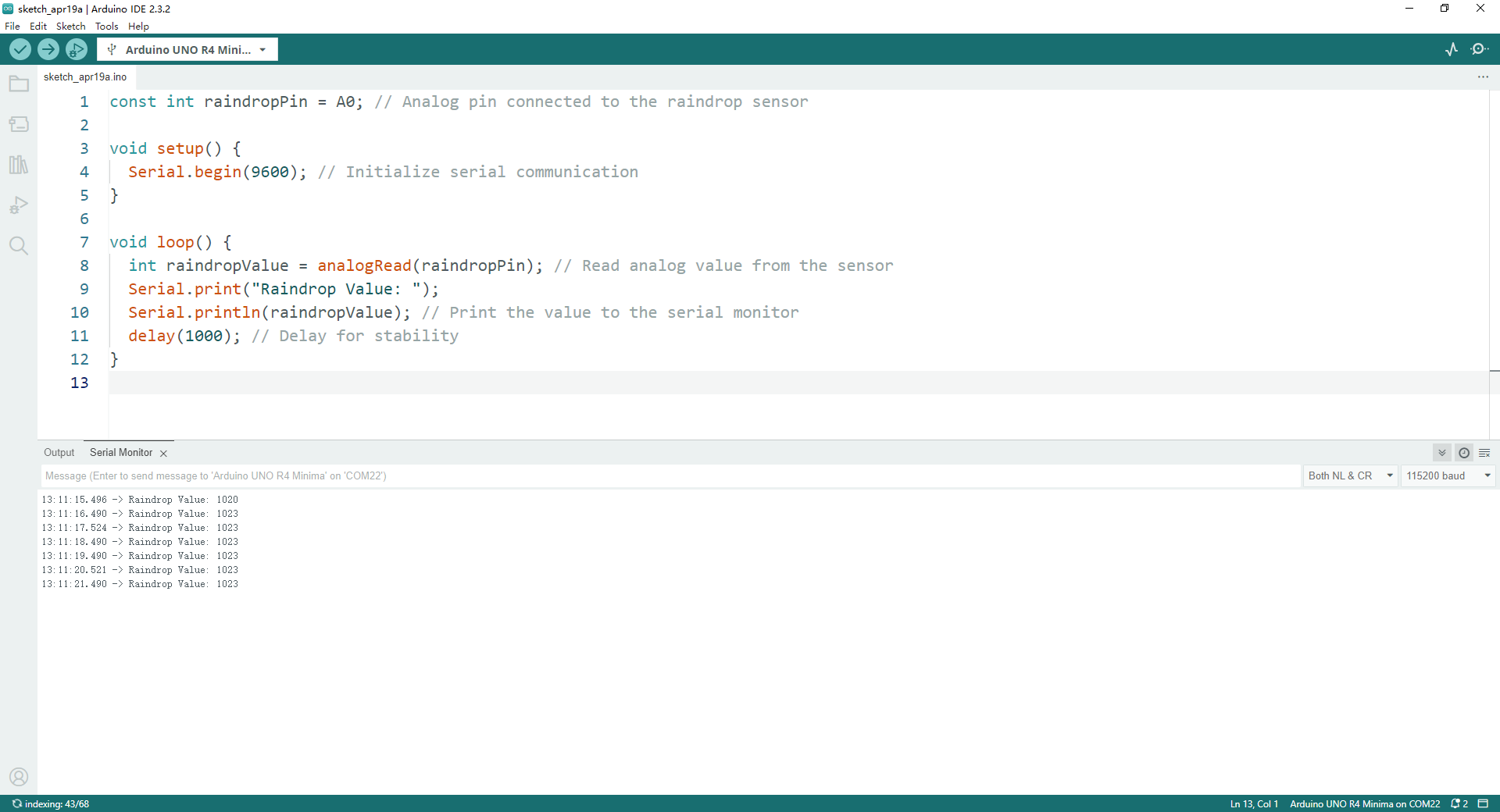
#### 5. Monitor the Data:

Open the Serial Monitor in the Arduino IDE (Tools -> Serial Monitor).

You should see the analog values from the raindrop sensor being printed to the serial monitor.

Observe how the values change when water is present or absent on the sensor surface.

If no rain drop on the sensor pad, it will shows:



If there is some water on the sensor pad:

The value will be changed.



Through experiments, it can be found that when raindrops fall, the value of the analog quantity will drop from 1023 to about 500, which means that we can judge the amount of rainfall through the number of the analog quantity and the amount of water!

Of course, if you only want the amount of one state, you can directly connect the digital pin to the DO pin, which will only provide high and low level messages, such as 1 for high level and 0 for low level.

#### 6. Experiment and Learn:

Experiment with different sensitivity settings of the raindrop sensor, if available.

Explore ways to convert the analog values to meaningful rainfall measurements.

Consider integrating the raindrop sensor into larger projects, such as weather stations or automated irrigation systems. The following experiment uses digital pins and buzzers to continue raindrop detection and judgment and issue warning sounds through the buzzer.

#### 7. Adding a buzzer to alarm.

Modify the code and upload it again.

#define SIGNAL\_PIN 7     // Define the output signal

int buzzer = 8;

int value = 0; // variable to store the sensor value

void setup() {

  Serial.begin(9600);

  pinMode(SIGNAL\_PIN, INPUT);   // configure D7 pin as an OUTPUT

  pinMode(buzzer, OUTPUT);

}

void loop() {

  value = digitalRead(SIGNAL\_PIN);    // read the analog value from sensor

  Serial.print("Digital Output: ");   // Print output to the serial port

  Serial.println(value);

  if(value == 1){                   // The high level is dry

  Serial.println("dry");

  digitalWrite(buzzer, LOW);

  }

  else{      // The low level is wet

  Serial.println("raining!!");

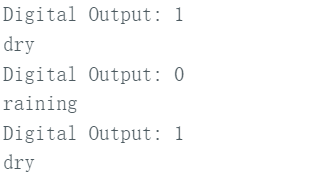
  digitalWrite(buzzer, HIGH); //Make a sound

  }

  delay(1000);

}

When a raindrop falls on the sensor, the buzzer goes off. Then open the serial monitor you can see this.



This project provides practical experience in sensor interfacing and data acquisition with Arduino. By reading data from the raindrop sensor, you can gain insights into environmental conditions and develop solutions for various applications. Have fun experimenting with your raindrop sensor project!

## Project 4 1.3-inch IPS 240x240 pixels RGB Display

### Description

This project focuses on interfacing a 1.3-inch IPS (In-Plane Switching) display with a resolution of 240x240 pixels with an Arduino Uno R4 Minima. The display provides a vibrant color screen suitable for displaying graphics, text, and images. By connecting the display to an Arduino, you can create various projects such as wearable devices, small gaming consoles, data loggers, and more.

### How-To Guide

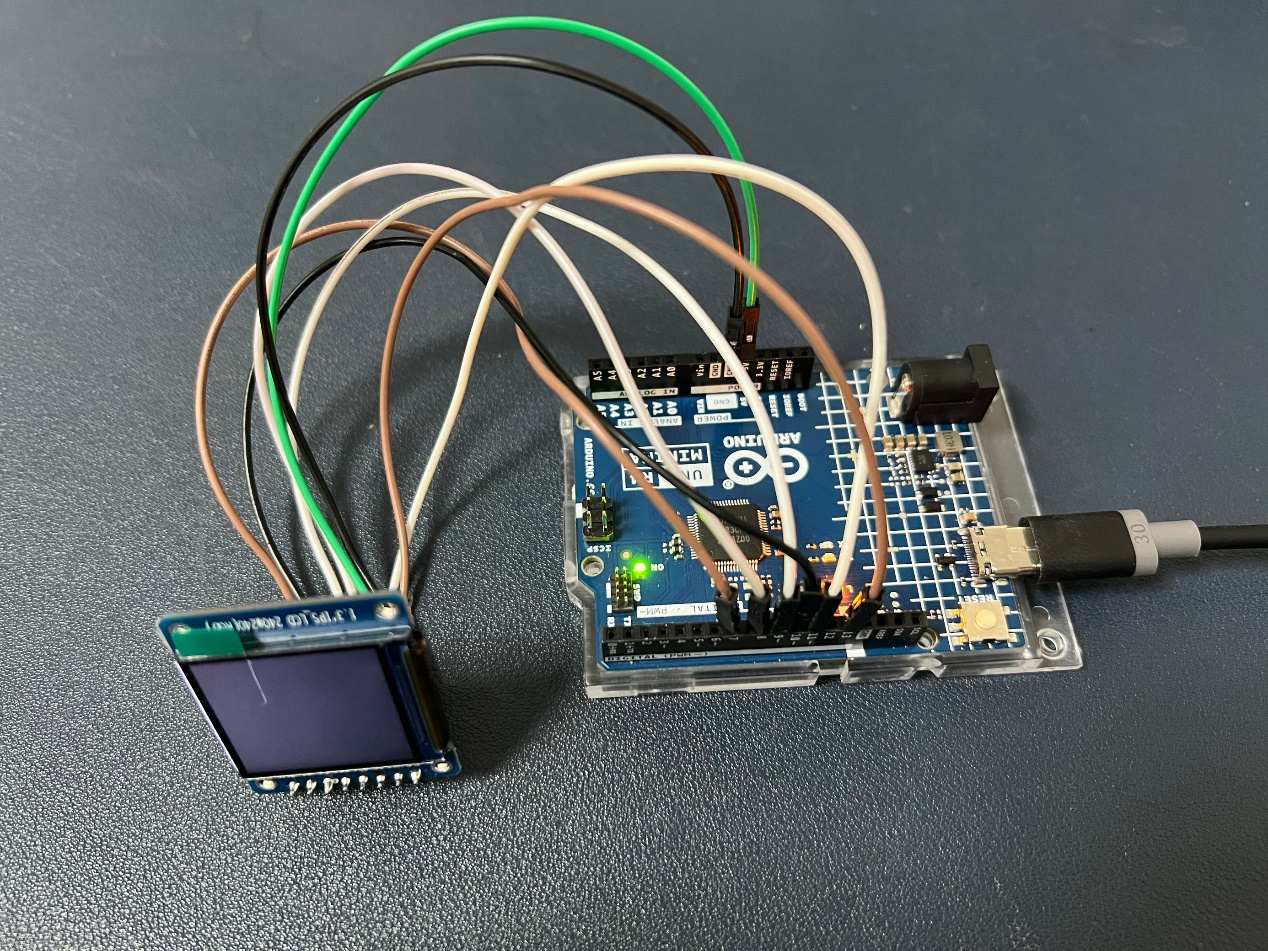
#### 1. Gather Components:

* 1 x Arduino Uno R4 minima
* 1 x 1.3-inch IPS 240x240 pixels RGB Display (e.g., ST7789 driver)
* 1 x Breadboard (optional)
* 8 x Jumper wires

#### 2. Assemble the Circuit:

* Connect the display to the Arduino Uno R4 as follows:
* VCC: Connect to the 5V pin on the Arduino.
* GND: Connect to the GND pin on the Arduino.
* SCK: Connect to digital pin 13 (SCK or SPI Clock).
* SDA: Connect to digital pin 11 (MOSI or SPI Data).
* DC: Connect to digital pin 9 (for the Data/Command selection).
* RESET: Connect to digital pin 8 (for reset).
* CS: Connect to digital pin 10 (for chip select).

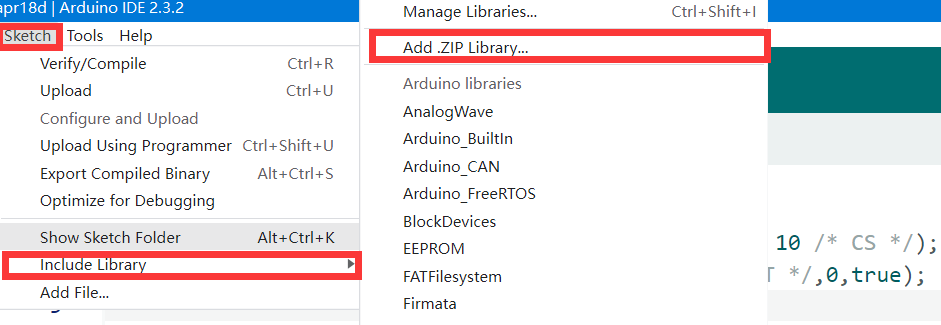
|  |  |
| --- | --- |
| 1.3-inch IPS 240x240 pixels RGB Display | Arduino UNO R4 Minima |
| GND | GND |
| VCC | 5V |
| SCL | 13 |
| SDA | 11 |
| RES | 8 |
| DC | 9 |
| CS | 10 |
| BLK | 7/NC |



#### 3. Add .Zip Library:

Here's a basic example code to initialize the display and draw a simple shape (e.g., a rectangle) on the screen, before you doing that, you need to adding library from a zip file which coming with package’s USB dongle, it stored in libraries folder in the USB dongle.

Then you can copy the code to the Arduino IDE and you need to add the library to the project, open the flash drive you buy in this kit, you will see the library. Then find the .zip file we can add it as a library.



#### 4. Write the Arduino Code:

#include <Arduino\_GFX\_Library.h>

Arduino\_DataBus \*bus = new Arduino\_HWSPI(9 /\* DC \*/, 10 /\* CS \*/);

Arduino\_GFX \*gfx = new Arduino\_ST7796(bus, 8 /\* RESET \*/,0,true);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* End of Arduino\_GFX setting

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup(void)

{

  Serial.begin(9600);

  pinMode(7, OUTPUT);

  digitalWrite(7, LOW);

  // Serial.setDebugOutput(true);

  // while(!Serial);

  Serial.println("Arduino\_GFX Hello World example");

  pinMode(7, OUTPUT);

  digitalWrite(7, HIGH);

  // Init Display

  if (!gfx->begin())

  {

    Serial.println("gfx->begin() failed!");

  }

  gfx->fillScreen(BLACK);

  gfx->setCursor(10, 10);

  gfx->setTextColor(PURPLE);

  gfx->setTextSize(3, 3);

  gfx->println("Hello World!");

  gfx->setCursor(10, 50);

  gfx->setTextColor(NAVY);

  gfx->setTextSize(3, 3);

  gfx->println("5");

  gfx->setCursor(30, 50);

  gfx->setTextColor(DARKGREEN);

  gfx->setTextSize(3, 3);

  gfx->println("2");

  gfx->setCursor(50, 50);

  gfx->setTextColor(DARKCYAN);

  gfx->setTextSize(3, 3);

  gfx->println("P");

  gfx->setCursor(70, 50);

  gfx->setTextColor(MAROON);

  gfx->setTextSize(3, 3);

  gfx->println("i");

  // Draw the box (if 1pix is a line)

  gfx->fillRect(115,65,10,10,WHITE);

  // It's equal to draw a line

  gfx->fillRect(150,0,1,240,WHITE);

  gfx->fillRect(0,100,240,1,WHITE);

  // Draw a circle

  gfx->drawEllipse(120,70,20,20,MAGENTA);

}

void loop()

{

  delay(1000); // 1 second

}

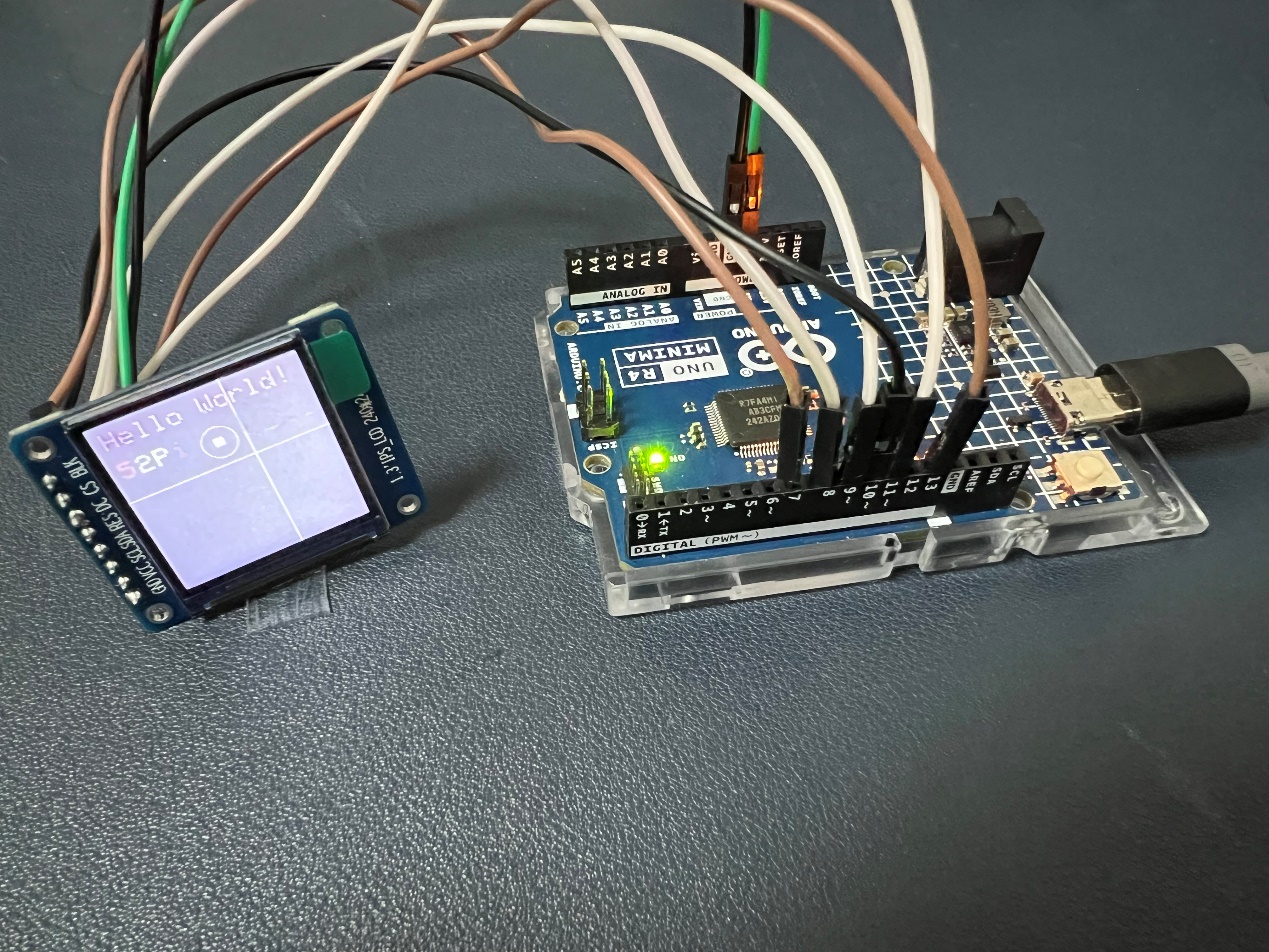
#### 5. Upload the Code:

* Connect your Arduino Uno R4 to your computer via USB.
* Open the Arduino IDE, copy the provided code, and paste it into a new sketch.
* Select the correct board and port in the Arduino IDE.
* Click the upload button to upload the code to your Arduino.

#### 6. Observe the Display:

Once the code is uploaded successfully, you should see a red rectangle being drawn on the display, followed by a black screen, with a delay of 2 seconds between each action.

If the display shows unexpected behavior or doesn't respond, double-check the connections and code.



#### 7. Code Explanation:

Let's break down the code:

#include <Arduino\_GFX\_Library.h>

This line includes the Arduino GFX Library, which provides graphics functions for drawing shapes, text, and more on displays.

Arduino\_DataBus \*bus = new Arduino\_HWSPI(9 /\* DC \*/, 10 /\* CS \*/);

Arduino\_GFX \*gfx = new Arduino\_ST7796(bus, 8 /\* RESET \*/,0,true);

These lines initialize the display interface and graphics objects:

* **bus:** It defines the communication bus used by the display. In this case, it's initialized as a **Hardware SPI bus**, specifying the pins for **Data/Command (DC)** and **Chip Select (CS)**.
* **gfx:** It initializes the graphics object using the ST7796 driver, specifying the communication **bus (bus)**, the pin for **Reset (RESET)**, the **address (0)**, and whether the display is vertically **flipped (true)**.

void setup(void)

{

  Serial.begin(9600);

  pinMode(7, OUTPUT);

  digitalWrite(7, LOW);

In the setup function:

* Serial communication is initialized at a baud rate of 9600.
* Pin 7 is set as an output and is set LOW. This pin seems to control something external to the display, perhaps a power supply or backlight.

 Serial.println("Arduino\_GFX Hello World example");

  pinMode(7, OUTPUT);

  digitalWrite(7, HIGH);

* A message is printed to the serial monitor indicating the start of the program.
* Pin 7 is set HIGH, possibly turning on the external device (e.g., backlight).

// Init Display

  if (!gfx->begin())

  {

    Serial.println("gfx->begin() failed!");

  }

  gfx->fillScreen(BLACK);

* The display is initialized. If initialization fails, an error message is printed to the serial monitor.
* The display is filled with black color using the **fillScreen()** function.

  gfx->setCursor(10, 10);

  gfx->setTextColor(PURPLE);

  gfx->setTextSize(3, 3);

  gfx->println("Hello World!");

* Text "Hello World!" is printed on the screen with purple color, at coordinates (10, 10), and with text size 3.

  // Additional text is printed similarly with different colors and positions.

  // Draw the box (if 1 pixel is a line)

  gfx->fillRect(115, 65, 10, 10, WHITE);

* A white box (rectangle) is drawn on the screen at coordinates (115, 65) with a width and height of 10 pixels each.

// It's equal to draw a line

  gfx->fillRect(150, 0, 1, 240, WHITE);

  gfx->fillRect(0, 100, 240, 1, WHITE);

Two white lines are drawn on the screen: one vertical line at x-coordinate 150 spanning the entire height of the display, and one horizontal line at y-coordinate 100 spanning the entire width.

// Draw a circle

  gfx->drawEllipse(120, 70, 20, 20, MAGENTA);

A magenta-colored ellipse (circle) is drawn on the screen with center coordinates (120, 70) and radii of 20 pixels each.

void loop()

{

  delay(1000); // 1 second

}

In the loop function, there's a 1-second delay, which keeps the program looping with a delay of 1 second between iterations. This is often used to keep the program running without executing any specific tasks repeatedly.

#### 8. Experiment and Learn:

This project provides a starting point for interfacing the 1.3-inch IPS 240x240 pixels RGB display with an Arduino Uno R4. You can further explore the capabilities of the display by adding text, images, animations, and interactive elements. Enjoy experimenting with your display project!