# 4. Extend Arduino's capabilities with sensors

Difficult Level：

## A. Digital sensor: Temperature & Humidity Sensor

### What is digital sensor?

Digital sensor is a kind of sensor with signal conversion and digital output function. It is able to convert physical quantities (such as temperature, humidity, pressure) or chemical quantities (such as gas concentration, PH) in the environment into digital signals and communicate with other systems through interfaces. Digital sensors are usually composed of sensing elements, analog-to-digital converters (ADC), processors and interface circuits.

In digital sensors, the sensing element plays the role of collecting environmental information, which is subject to changes in physical or chemical quantities and changes accordingly. The analog to digital converter (ADC) converts the analog signal output by the sensing element into a digital signal. The processor is responsible for processing and analyzing the digital signal and transmitting the results to the external system through the interface circuit.

Digital sensors are characterized by high accuracy, high stability and low power consumption. They provide accurate measurements and have a long service life.

### Temperature & Humidity Sensor

Why we can just get the date? Because this is a digital sensor, the analog part is already taken care of on the sensor side. And now let’s start the Temperature & Humidity Sensor project.

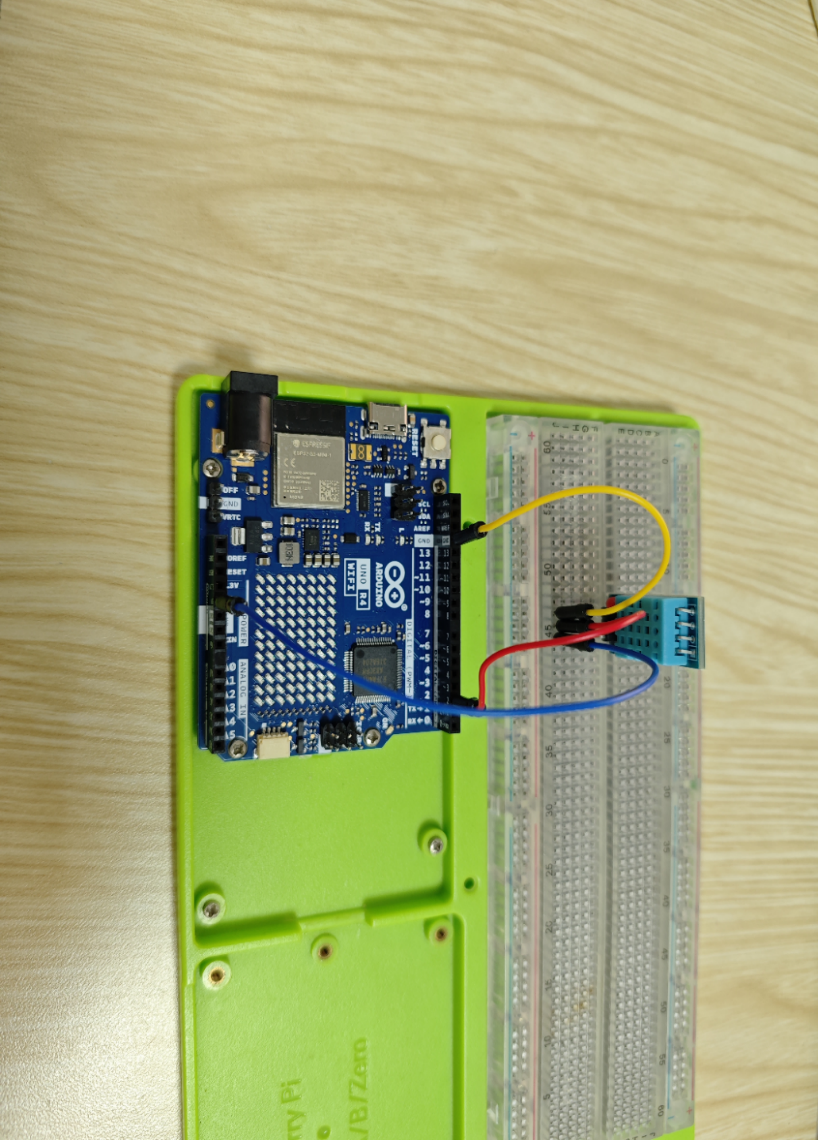
#### What you need

* Arduino IDE software
* 1 x Arduino UNO R4 Wi-Fi
* 1 x USB 2.0 cable type C
* 1 x Temperature and humidity sensor Module
* 3 x Jumper wires
* 1 x 52Pi Bread Board

#### How to wiring circuit diagram？

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

You can look at the circuit connection diagram below to connect our own devices.



Open the Arduino IDE software and connect the Arduino UNO R4 WIFI with the PC.

Copy the code and upload it to your Arduino UNO R4 WIFI.

**#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();  // Initialize the DHT sensor**

**}**

**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 {**

**// Print the humidity, temperature to the serial monitor**

**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");**

**}**

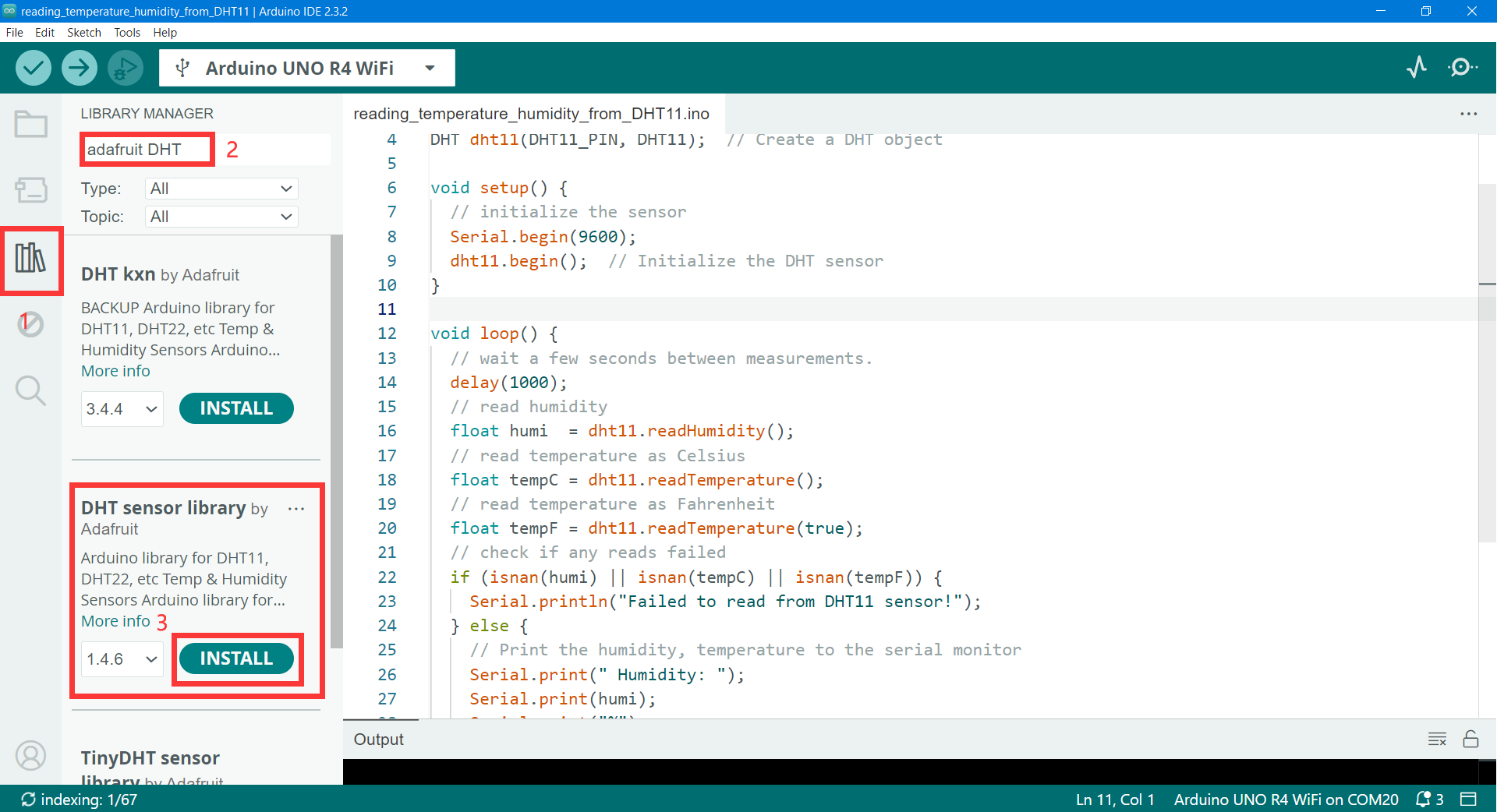
**}**

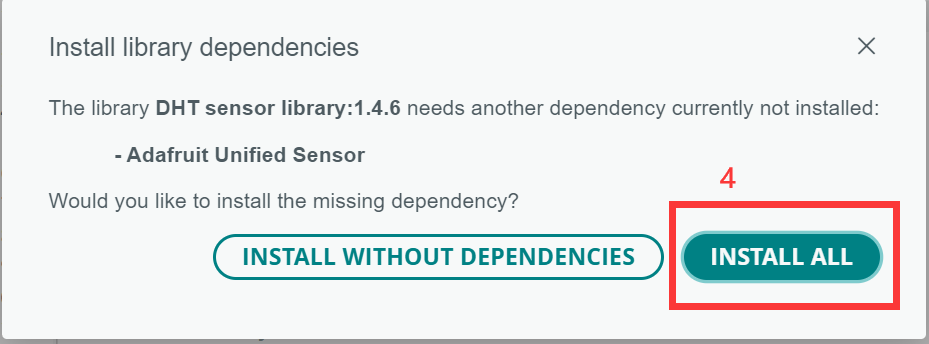
Inclusion of necessary libraries. This part of the code includes the DHT sensor library and defines the pin number and sensor type used in this project.

**#include "DHT.h"**

**#define Dht11\_PIN 2  // Define the pin used to connect the sensor**

**Note**: Check if you have the DHT library installed.





Creation of DHT object. Here we create a DHT object using the defined pin number and sensor type.

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

Then we initialize in setup (), such as serial port initialization, DHT initialization.

**void setup() {**

**// initialize the sensor**

**Serial.begin(9600);**

**dht11.begin();  // Initialize the DHT sensor**

**}**

Main loop. The loop () function runs continuously after the setup function. Here, we read the humidity and temperature values, and print these values to the serial monitor. If the sensor read fails (returns NaN), it prints an error message.

**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!");**

**} else {**

**// Print the humidity, temperature to the serial monitor**

**Serial.print(" Humidity: ");**

**Serial.print(humi);**

**Serial.print("%");**

**Serial.print("   ");**

**Serial.print(" Temperature: ");**

**Serial.print(tempC);**

**Serial.print("°C");**

**Serial.print("   ");**

**Serial.print(" Fahrenheit: ");**

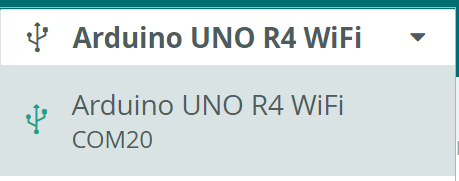
**Serial.print(tempF);**

**Serial.println("°F");**

**}**

**}**

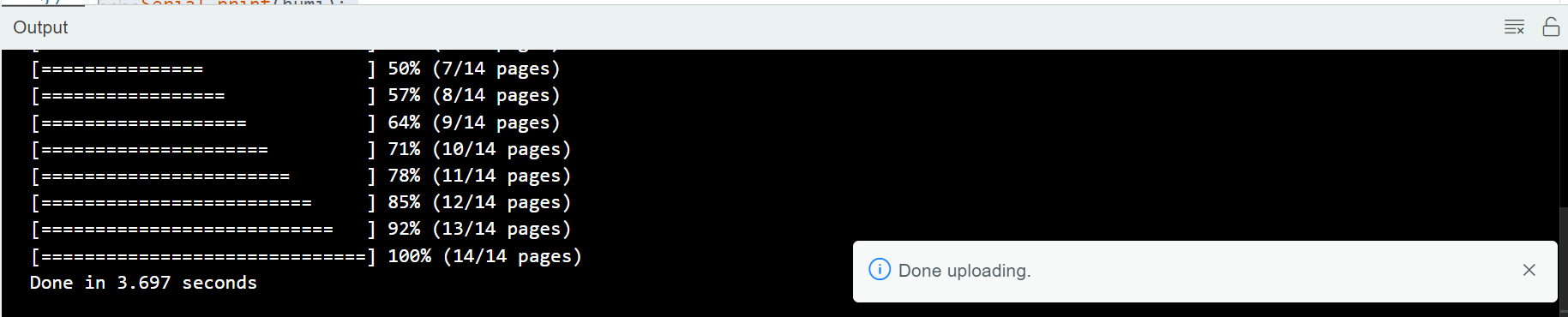
Then select the com port where the Arduino UNO R4 WIFI is located in the software.



We can click the upload button to upload our program into the Arduino UNO R4 WIFI.



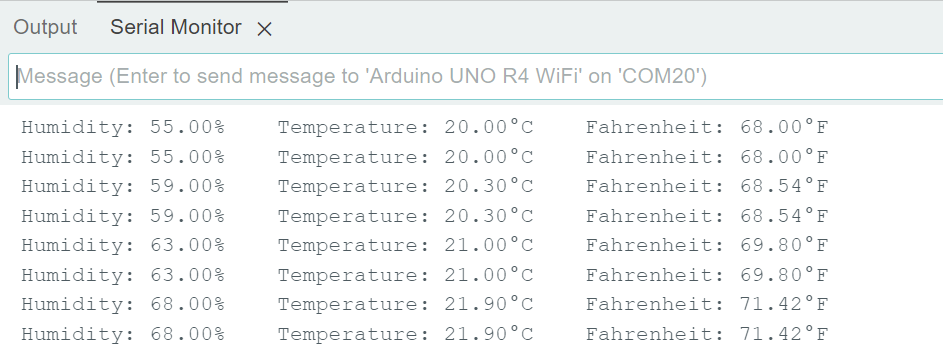
Upload successful.



Open the Serial Monitor.



We can see temperature and humidity on the serial monitor.



## B. Other digital sensor and code

### MPU6050 Gyroscope & accelerate

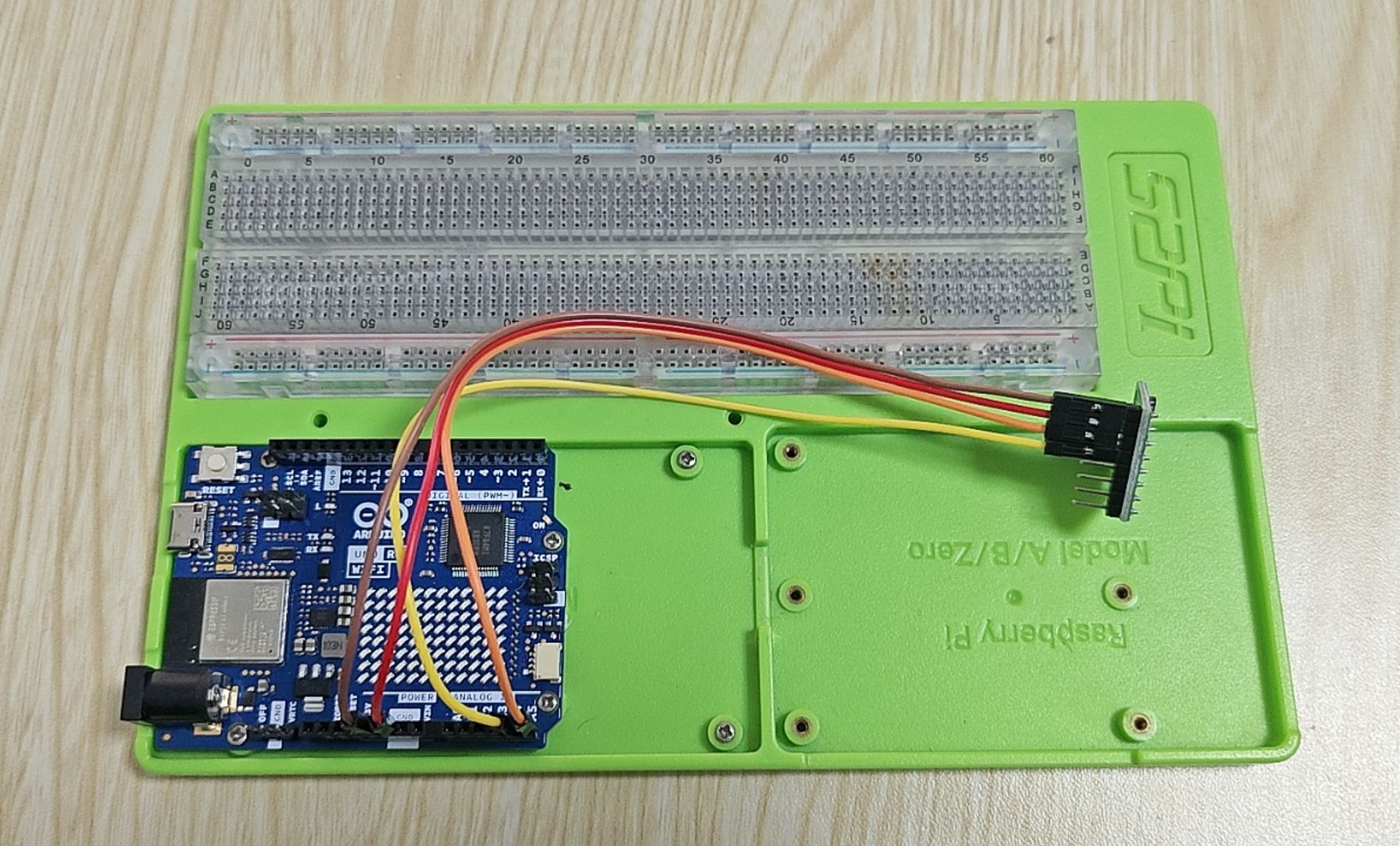
#### What you need

* Arduino IDE software
* 1 x Arduino UNO R4 Wi-Fi
* 1 x USB 2.0 cable Type C
* 1 x Accelerometer & Gyroscope Module(MPU6050)
* 4 x Female-Male Jumper wires
* 1 x 52Pi Bread Board

#### How to wiring circuit diagram?

|  |  |
| --- | --- |
| Arduino UNO R4 WIFI | Accelerometer & Gyroscope Module(MPU6050) |
| 5V | VCC |
| GND | GND |
| A5 | SCL |
| A4 | SDA |

You can look at the circuit connection diagram below to connect our own devices.



Then connect the computer with the Arduino UNO R4 WIFI.

Open the Arduino IDE, copy and paste the code.

**#include <Adafruit\_MPU6050.h>**

**#include <Adafruit\_Sensor.h>**

**#include <Wire.h>**

**// Create an object for the MPU6050 sensor**

**Adafruit\_MPU6050 mpu;**

**void setup(void) {**

**// Initialize the serial communication**

**Serial.begin(9600);**

**// Check if the MPU6050 sensor is detected**

**if (!mpu.begin()) {**

**Serial.println("Failed to find MPU6050 chip");**

**while (1) {**

**delay(10);**

**}**

**}**

**Serial.println("MPU6050 Found!");**

**// set accelerometer range to +-8G**

**mpu.setAccelerometerRange(MPU6050\_RANGE\_8\_G);**

**// set gyro range to +- 500 deg/s**

**mpu.setGyroRange(MPU6050\_RANGE\_500\_DEG);**

**// set filter bandwidth to 21 Hz**

**mpu.setFilterBandwidth(MPU6050\_BAND\_21\_HZ);**

**// Add a delay for stability**

**delay(100);**

**}**

**void loop() {**

**/\* Get new sensor events with the readings \*/**

**sensors\_event\_t a, g, temp;**

**mpu.getEvent(&a, &g, &temp);**

**/\* Print out the values \*/**

**Serial.print("Acceleration X: ");**

**Serial.print(a.acceleration.x);**

**Serial.print(", Y: ");**

**Serial.print(a.acceleration.y);**

**Serial.print(", Z: ");**

**Serial.print(a.acceleration.z);**

**Serial.println(" m/s^2");**

**// Print out the rotation readings in rad/s**

**Serial.print("Rotation X: ");**

**Serial.print(g.gyro.x);**

**Serial.print(", Y: ");**

**Serial.print(g.gyro.y);**

**Serial.print(", Z: ");**

**Serial.print(g.gyro.z);**

**Serial.println(" rad/s");**

**// Print out the temperature reading in degrees Celsius**

**Serial.print("Temperature: ");**

**Serial.print(temp.temperature);**

**Serial.println(" degC");**

**// Add a blank line for readability**

**Serial.println("");**

**// Add a delay to avoid flooding the serial monitor**

**delay(1000);**

**}**

The code starts by including the necessary libraries and creating an object for the MPU6050 sensor. This code uses the Adafruit\_MPU6050 library, Adafruit\_Sensor library, and Wire library. The Adafruit\_MPU6050 library is used to interact with the MPU6050 sensor and retrieve acceleration, rotation, and temperature data. The Adafruit\_Sensor library provides a common interface for various types of sensors. The Wire library is used for I2C communication, which is necessary to communicate with the MPU6050 sensor.

**#include <Adafruit\_MPU6050.h>**

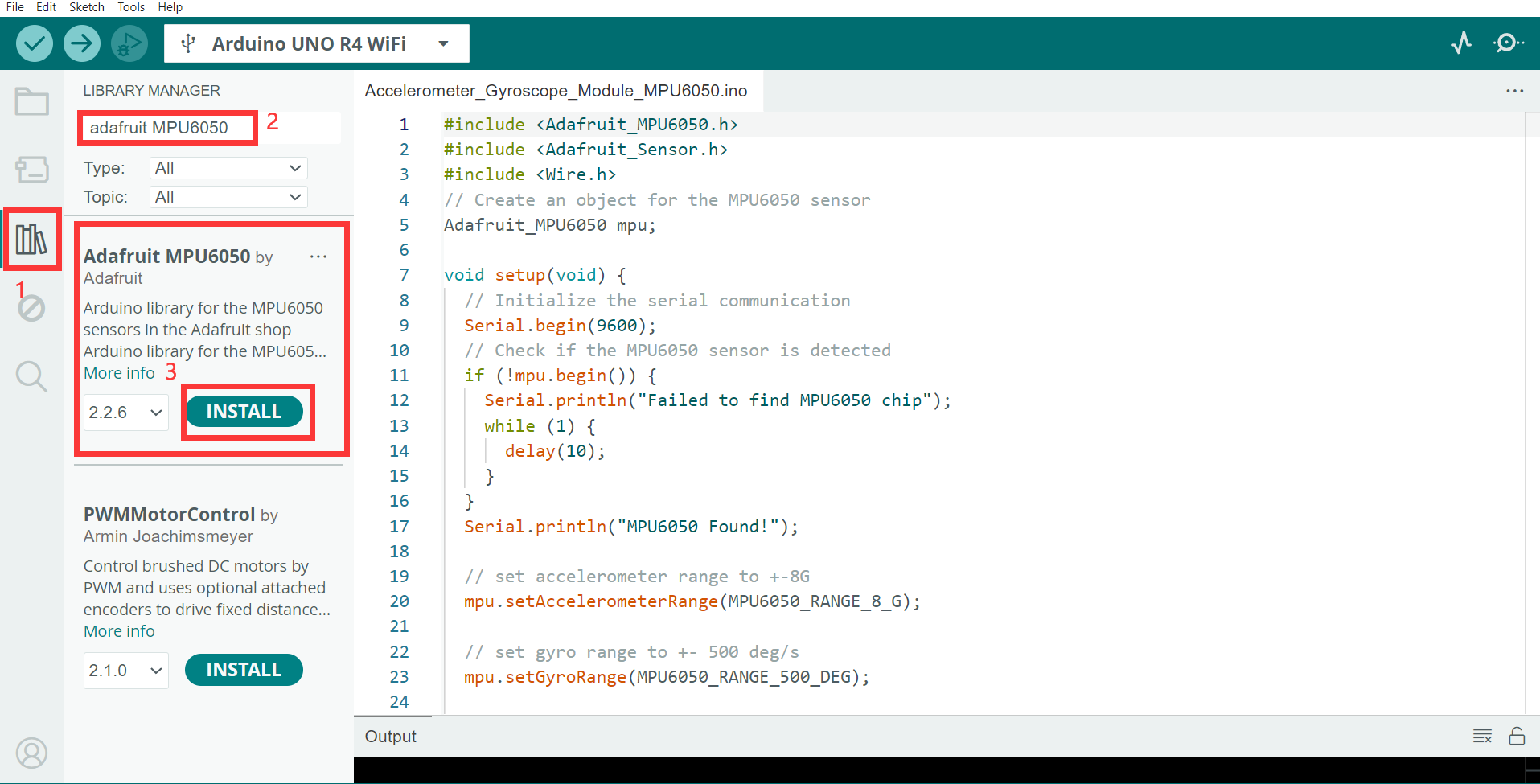
**#include <Adafruit\_Sensor.h>**

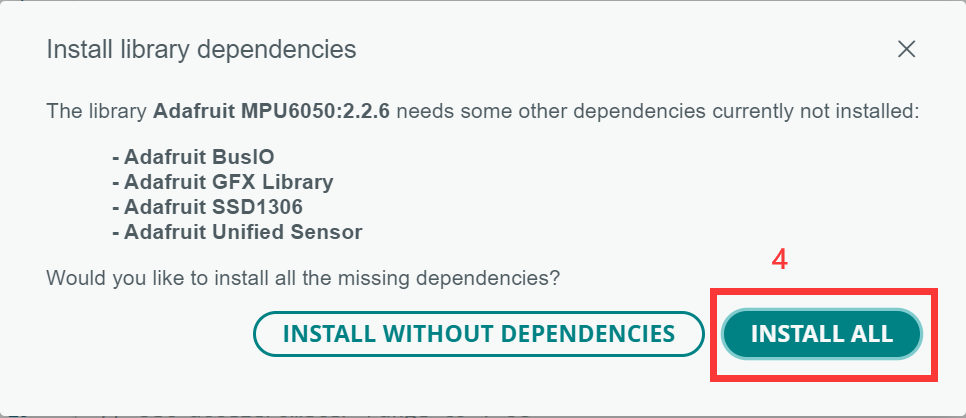
**#include <Wire.h>**

**// Create an object for the MPU6050 sensor**

**Adafruit\_MPU6050 mpu;**

**Note**: Check if you have the library installed. If not, follow these steps to install it.





The setup () function initializes the serial communication and checks if the sensor is detected. If the sensor is not found, the Arduino enters an infinite loop with a “Failed to find MPU6050 chip’ message. If found, the accelerometer range, gyro range, and filter bandwidth are set, and a delay is added for stability.

**void setup(void) {**

**// Initialize the serial communication**

**Serial.begin(9600);**

**// Check if the MPU6050 sensor is detected**

**if (!mpu.begin()) {**

**Serial.println("Failed to find MPU6050 chip");**

**while (1) {**

**delay(10);**

**}}**

**Serial.println("MPU6050 Found!");**

**// set accelerometer range to +-8G**

**mpu.setAccelerometerRange(MPU6050\_RANGE\_8\_G);**

**// set gyro range to +- 500 deg/s**

**mpu.setGyroRange(MPU6050\_RANGE\_500\_DEG);**

**// set filter bandwidth to 21 Hz**

**mpu.setFilterBandwidth(MPU6050\_BAND\_21\_HZ);**

**// Add a delay for stability**

**delay(100);**

**}**

In the loop () function, the program creates events to store the sensor readings and then retrieves the readings. The acceleration, rotation, and temperature values are then printed to the serial monitor.

**void loop() {**

**// Get new sensor events with the readings**

**sensors\_event\_t a, g, temp;**

**mpu.getEvent(&a, &g, &temp);**

**// Print out the acceleration, rotation, and temperature readings**

**// ...**

**// Add a delay to avoid flooding the serial monitor**

**delay(1000);**

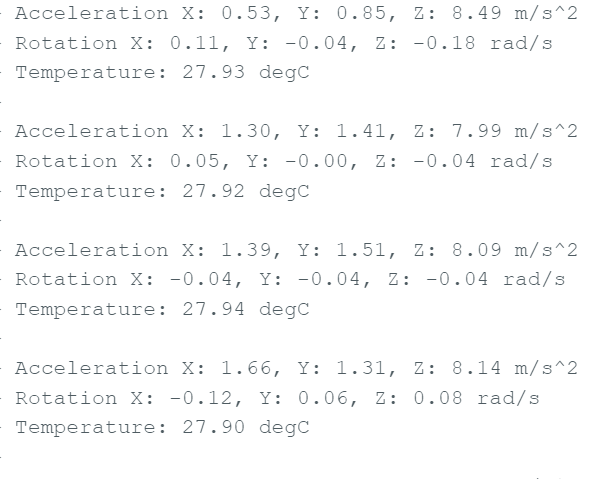
**}**

Then we can upload our program into the Arduino UNO R4 WIFI. After upload successful.

Open the Serial Monitor.



Then we can see the values of acceleration, rotation, and temperature.



### Atmospheric Pressure Sensor (BMP280)

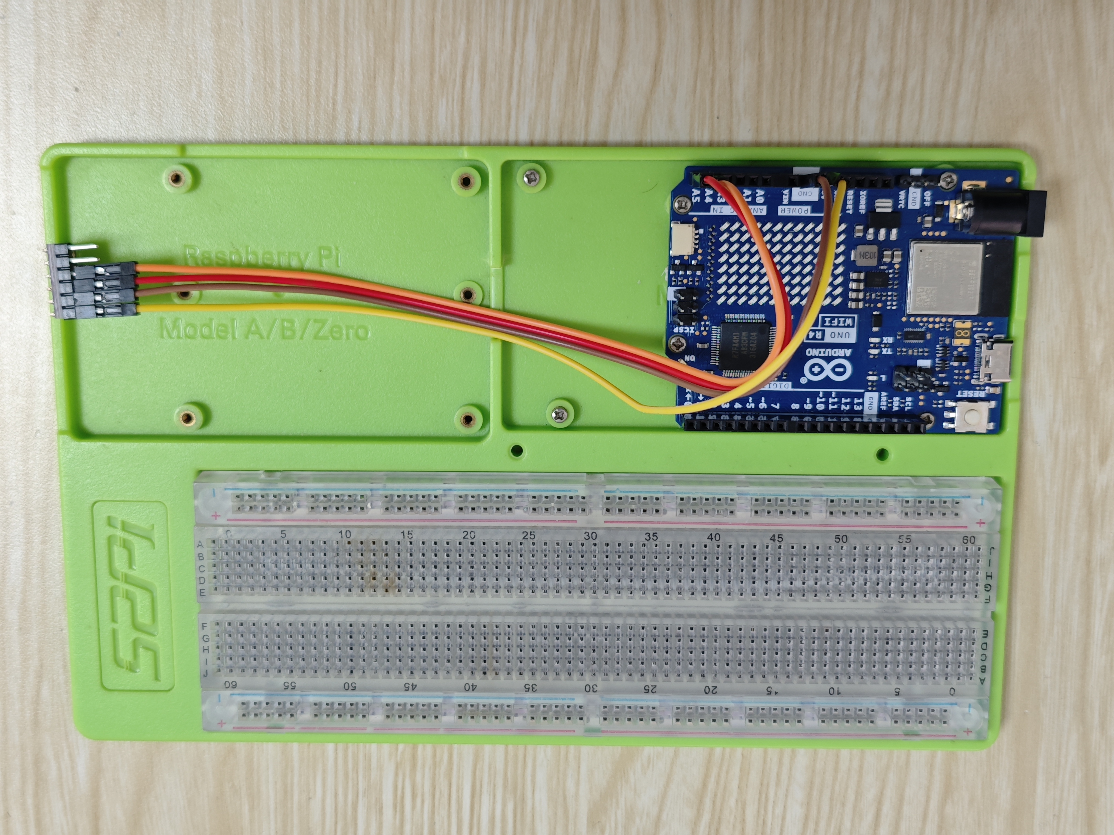
#### What you need

* Arduino IDE software
* 1 x Arduino UNO R4 Wi-Fi
* 1 x USB 2.0 cable type C
* 1 x Atmospheric Pressure Sensor Module
* 4 x Female-Male Jumper wires
* 1 x 52Pi Bread Board

#### How to wiring circuit diagram?

|  |  |
| --- | --- |
| Arduino UNO R4 WIFI | Atmospheric Pressure Sensor |
| 3.3V | VCC |
| GND | GND |
| A5 | SCL |
| A4 | SDA |

You can look at the circuit connection diagram below to connect our own devices.



Then connect the computer with the Arduino UNO R4 WIFI. Open the Arduino IDE, copy and paste the code.

**#include <Wire.h>**

**#include <SPI.h>**

**#include <Adafruit\_BMP280.h>**

**#define BMP280\_ADDRESS 0x76**

**Adafruit\_BMP280 bmp;  // use I2C interface**

**void setup() {**

**Serial.begin(9600);          // initialize serial communication with baud rate of 9600**

**while (!Serial) delay(100);  // wait for native usb**

**Serial.println(F("BMP280 test"));**

**unsigned status;**

**status = bmp.begin(BMP280\_ADDRESS);**

**if (!status) {**

**Serial.println(F("Could not find a valid BMP280 sensor, check wiring or "**

**"try a different address!"));**

**while (1) delay(10);  // Stop code execution if the sensor is not found.**

**}**

**/\* Default settings from datasheet. \*/**

**bmp.setSampling(Adafruit\_BMP280::MODE\_NORMAL,     /\* Operating Mode. \*/**

**Adafruit\_BMP280::SAMPLING\_X2,     /\* Temp. oversampling \*/**

**Adafruit\_BMP280::SAMPLING\_X16,    /\* Pressure oversampling \*/**

**Adafruit\_BMP280::FILTER\_X16,      /\* Filtering. \*/**

**Adafruit\_BMP280::STANDBY\_MS\_500); /\* Standby time. \*/**

**}**

**void loop() {**

**// Read and print temperature in degrees Celsius.**

**Serial.print(F("Temperature = "));**

**Serial.print(bmp.readTemperature());**

**Serial.println(" °C");**

**//Read and print atmospheric pressure in hectopascals (hPa).**

**Serial.print(F("Pressure = "));**

**Serial.print(bmp.readPressure());**

**Serial.println(" hPa");**

**//Read and print approximate altitude based on standard pressure (1013.25 hPa).**

**Serial.print(F("Approx altitude = "));**

**Serial.print(bmp.readAltitude(1013.25));**

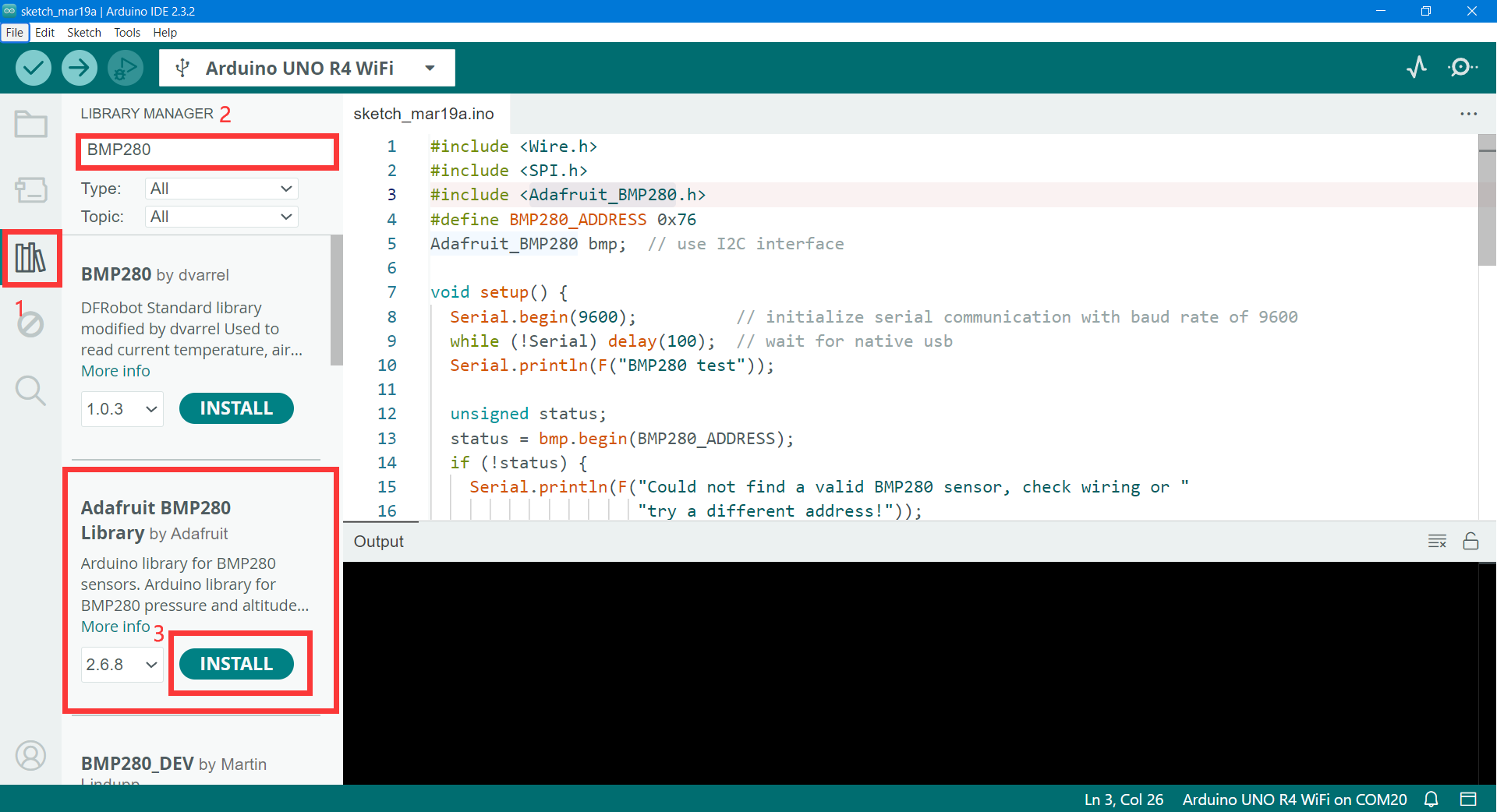
**Serial.println(" m");**

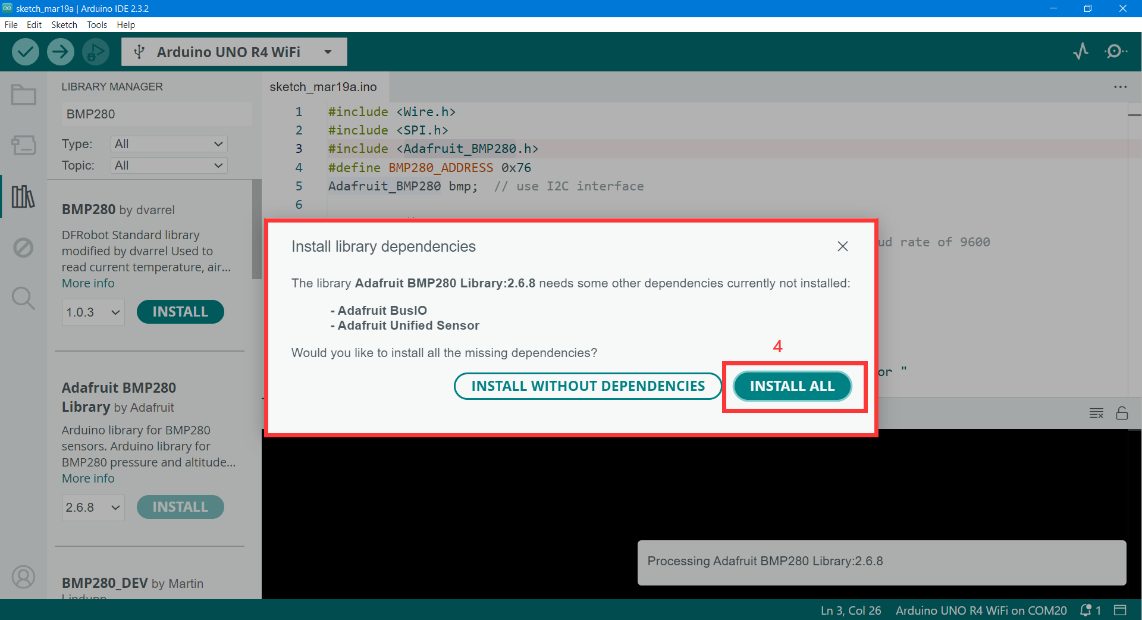
**Serial.println();  // Print a blank line to separate readings.**

**delay(2000);       // Wait for 2 seconds before taking the next set of readings.**

**}**

**Note**: To install the library, use the Arduino Library Manager and search for **“Adafruit BMP280”** And install it.





Including Libraries and Initialization. Necessary libraries are included and the BMP280 sensor is initialized for communication using the I2C interface.

Adafruit BMP280 Library: This library provides an easy-to-use interface for the BMP280 sensor, allowing the user to read temperature, pressure, and altitude.

Wire.h: Used for I2C communication.

**#include <Wire.h>**

**#include <Adafruit\_BMP280.h>**

**#define BMP280\_ADDRESS 0x76**

**Adafruit\_BMP280 bmp; // use I2C interface**

The setup () function initializes the Serial communication, checks for the BMP280 sensor, and sets up the sensor with default settings.

**void setup() {**

**Serial.begin(9600);**

**while (!Serial) delay(100);**

**Serial.println(F("BMP280 test"));**

**unsigned status;**

**status = bmp.begin(BMP280\_ADDRESS);**

**// ... (rest of the setup code)**

The loop () function reads data from the BMP280 sensor for temperature, pressure, and altitude. This data is printed to the Serial Monitor.

**void loop() {**

**// ... (read and print temperature, pressure, and altitude data)**

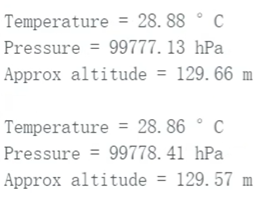
**}**

Then we can upload our program into the Arduino UNO R4 WIFI. After upload successful,

Open the Serial Monitor.



We can see the temperature, pressure, and approximate altitude in the serial monitor.



**Congratulations！**



You have already got this new skills!!! and let’s get into next chapter!