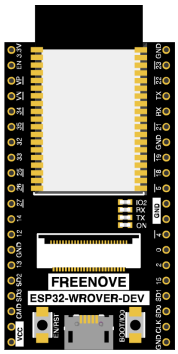
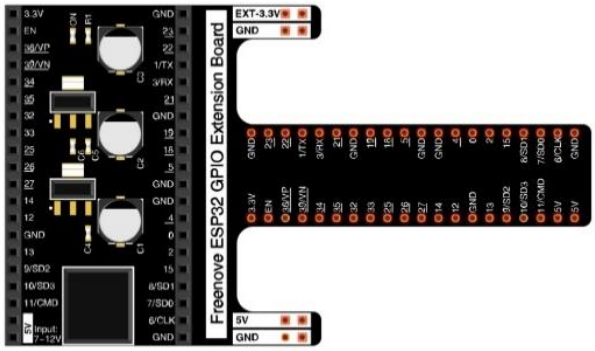
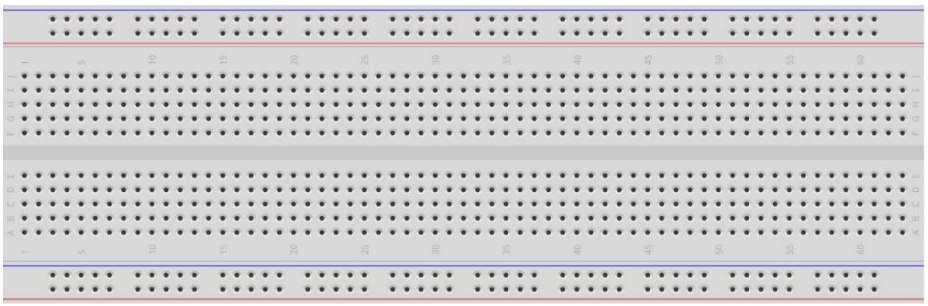
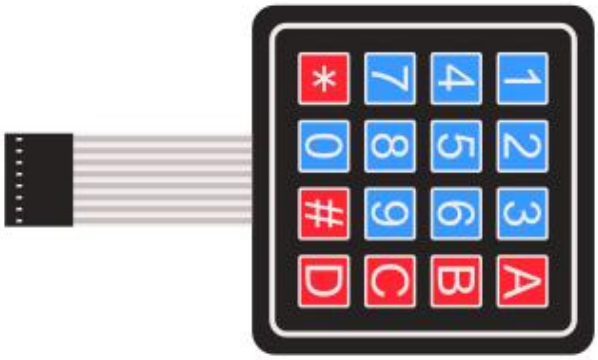
Chapter 22 Matrix Keypad

Earlier we learned about a single push button switch. In this chapter, we will learn about matrix keyboards, which integrates a number of push button switches as keys for the purposes of input.

Project 22.1 Matrix Keypad

In this project, we will attempt to get every key code on the matrix keypad to work.

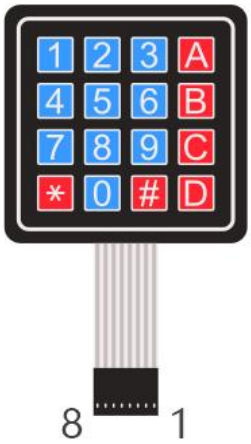
Component List

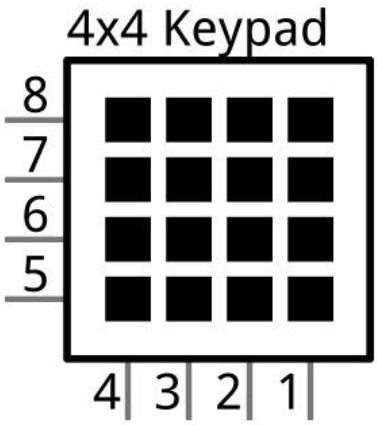
ESP32-WROVER x1 	GPIO Extension Board x1 
Breadboard x1 	
Jumper M/M x8 	4x4 Matrix Keypad x1 

Component knowledge

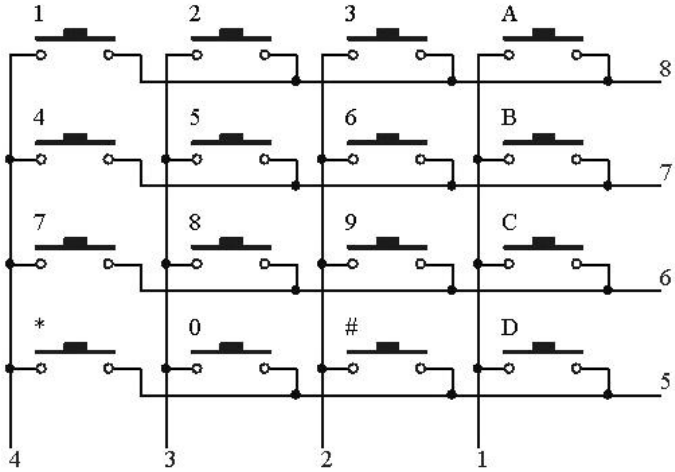
4x4 Matrix Keypad

A keypad matrix is a device that integrates a number of keys in one package. As is shown below, a 4x4 keypad matrix integrates 16 keys:





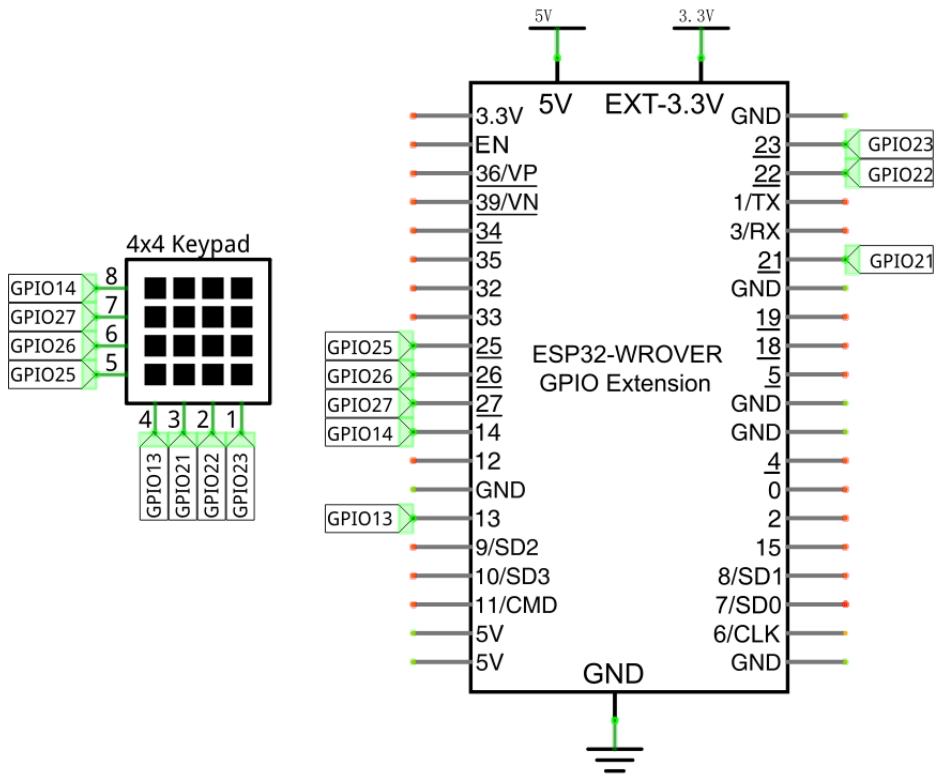
Similar to the integration of a LED matrix, the 4x4 keypad matrix has each row of keys connected with one pin and this is the same for the columns. Such efficient connections reduce the number of processor ports required. The internal circuit of the Keypad Matrix is shown below.



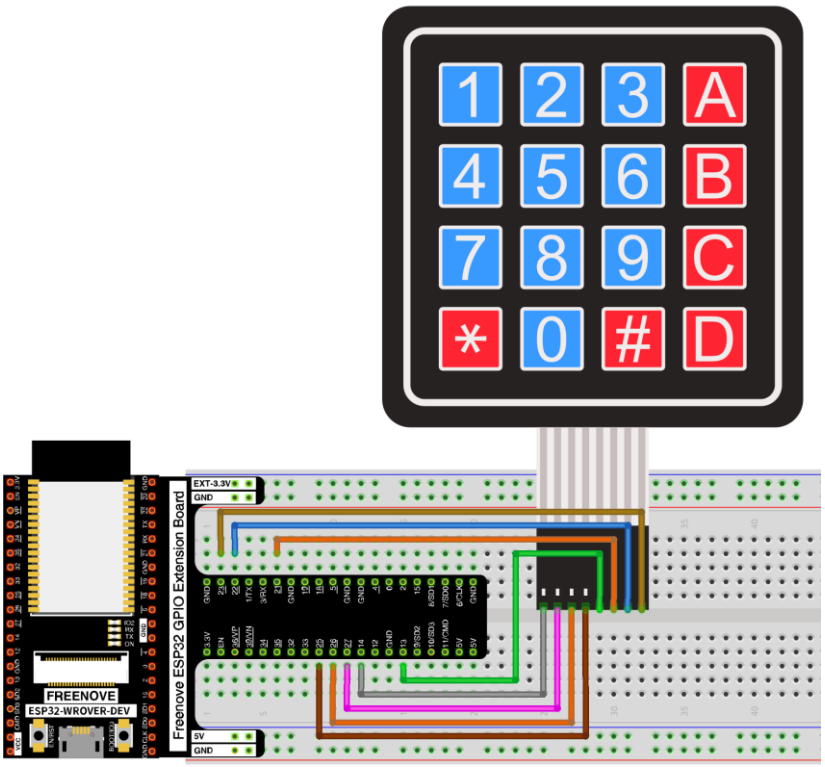
The usage is similar to the LED matrix, using a row or column scanning method to detect the state of each key's position by column and row. Take column scanning method as an example, send low level to the first 1 column (Pin1), detect level state of row 5, 6, 7, 8 to judge whether the key A, B, C, D are pressed. Then send low level to column 2, 3, 4 in turn to detect whether other keys are pressed. Therefore, you can get the state of all of the keys.

Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



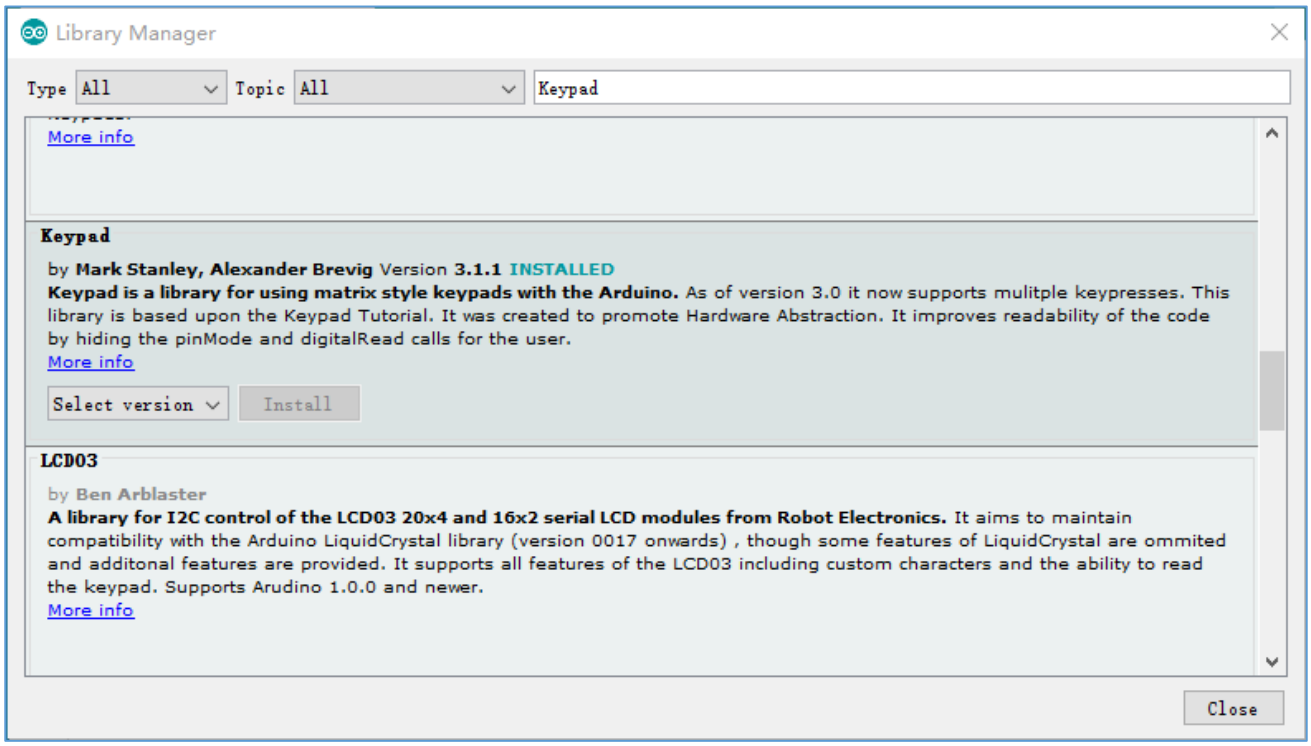
Sketch

This code is used to obtain all key codes of the 4x4 matrix keypad, when one of the keys is pressed, the key code will be printed out via serial port.

How to install the library

We use the third party library Keypad. If you haven't installed it yet, please do so before learning. The steps to add third-party Libraries are as follows: open arduino->Sketch->Include library-> Manage libraries. Enter " Keypad" in the search bar and select " Keypad " for installation.

Refer to the following operations:



Sketch_22.1_Get_Input_Characters

Sketch_22.1_Get_Input_Characters | Arduino 1.8.13 Hourly Build 2020/02/19 03:33

File Edit Sketch Tools Help

Sketch_22.1_Get_Input_Characters

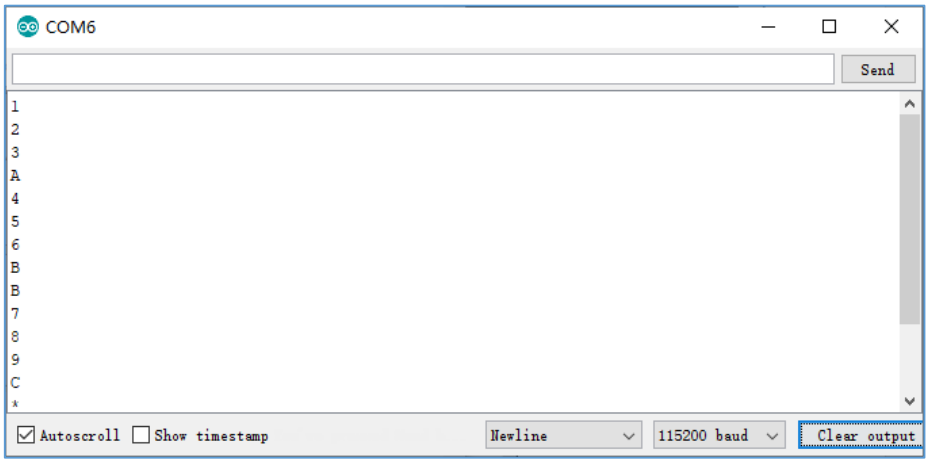
```
/*****************************************************************
  Filename    : Get Input Characters
  Description : Call the Keypad function to set the matrix keyboard and get the keys for the matri
  Auther      : www.freenove.com
  Modification: 2020-3-5
*****************************************************************/
#include <Keypad.h>

// define the symbols on the buttons of the keypad
char keys[4][4] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPins[4] = {14, 27, 26, 25}; // connect to the row pinouts of the keypad
byte colPins[4] = {13, 21, 22, 23};   // connect to the column pinouts of the keypad

// initialize an instance of class NewKeypad
Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);

void setup() {
  Serial.begin(115200); // Initialize the serial port and set the baud rate to 115200
  Serial.println("ESP32 is ready!");  // Print the string "UNO is ready!"
}

void loop() {
  // Get the character input
  char keyPressed = myKeypad.getKey();
  // If there is a character input, sent it to the serial port
  if (keyPressed) {
    Serial.println(keyPressed);
  }
}
```

Done uploading.

```
Leaving...
Hard resetting via RTS pin...
```

24 ESP32 Wrover Module on COM6

Download the code to ESP32-WROVER, open the serial port monitor, set the baud rate to 115200, press the keyboard, the value of the pressed keys will be printed out via the serial port. As shown in the following figure:



The following is the program code:

```
#include <Keypad.h>
// define the symbols on the buttons of the keypad
char keys[4][4] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[4] = {14, 27, 26, 25}; // connect to the row pinouts of the keypad
byte colPins[4] = {13, 21, 22, 23};   // connect to the column pinouts of the keypad

// initialize an instance of class NewKeypad
Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);

void setup() {
  Serial.begin(115200); // Initialize the serial port and set the baud rate to 115200
  Serial.println("ESP32 is ready! ");  // Print the string "ESP32 is ready! "
}

void loop() {
  // Get the character input
  char keyPressed = myKeypad.getKey();
  // If there is a character input, sent it to the serial port
  if (keyPressed) {
    Serial.println(keyPressed);
  }
}
```

First, add header file, define 4*4 matrix keyboard key value and the matrix keyboard pin.

```
#include <Keypad.h>
// define the symbols on the buttons of the keypad
char keys[4][4] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[4] = {14, 27, 26, 25}; // connect to the row pinouts of the keypad
byte colPins[4] = {13, 21, 22, 23};   // connect to the column pinouts of the keypad
```

Second, define a matrix keyboard object and associate the keys and pins with it.

```
Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);
```

Finally, get the key value and print it out via the serial port.

```
void loop() {
  // Get the character input
  char keyPressed = myKeypad.getKey();
  // If there is a character input, sent it to the serial port
  if (keyPressed) {
    Serial.println(keyPressed);
  }
}
```

Reference

class Keypad You need to add the library each time you use the Keypad.

Keypad(char *userKeymap, byte *row, byte *col, byte numRows, byte numCols);

Constructor, the parameters are: key code of keyboard, row pin, column pin, the number of rows, the number of columns.

char getKey();

Get the key code of the pressed key. If no key is pressed, the return value is NULL.

void setDebounceTime(uint);

Set the debounce time with a default time of 10ms.

void setHoldTime(uint);

Set the duration for the key to keep stable state after pressed.

bool isPressed(char keyChar);

Judge whether the key with code "keyChar" is pressed.

char waitForKey();

Wait for a key to be pressed, and return key code of the pressed key.

KeyState getState();

Get the state of the keys.

bool keyStateChanged();

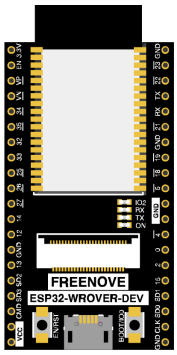
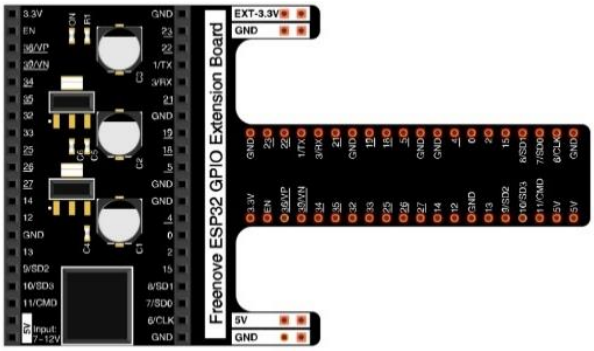
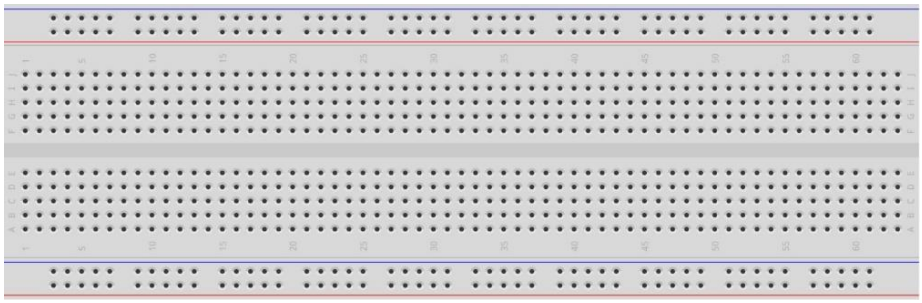
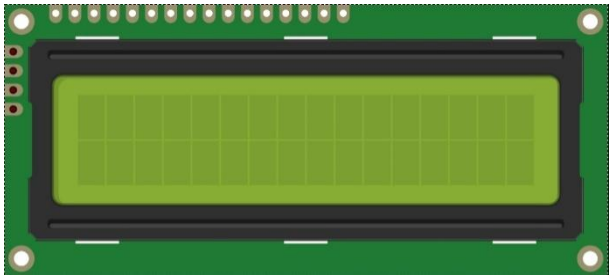
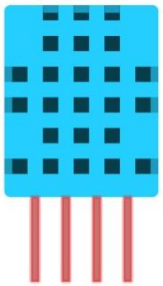
Judge whether there is a change of key state, then return True or False.

For More information about Keypad, please visit: http://playground.arduino.cc/Code/Keypad

Project 24.2 Hygrothermograph

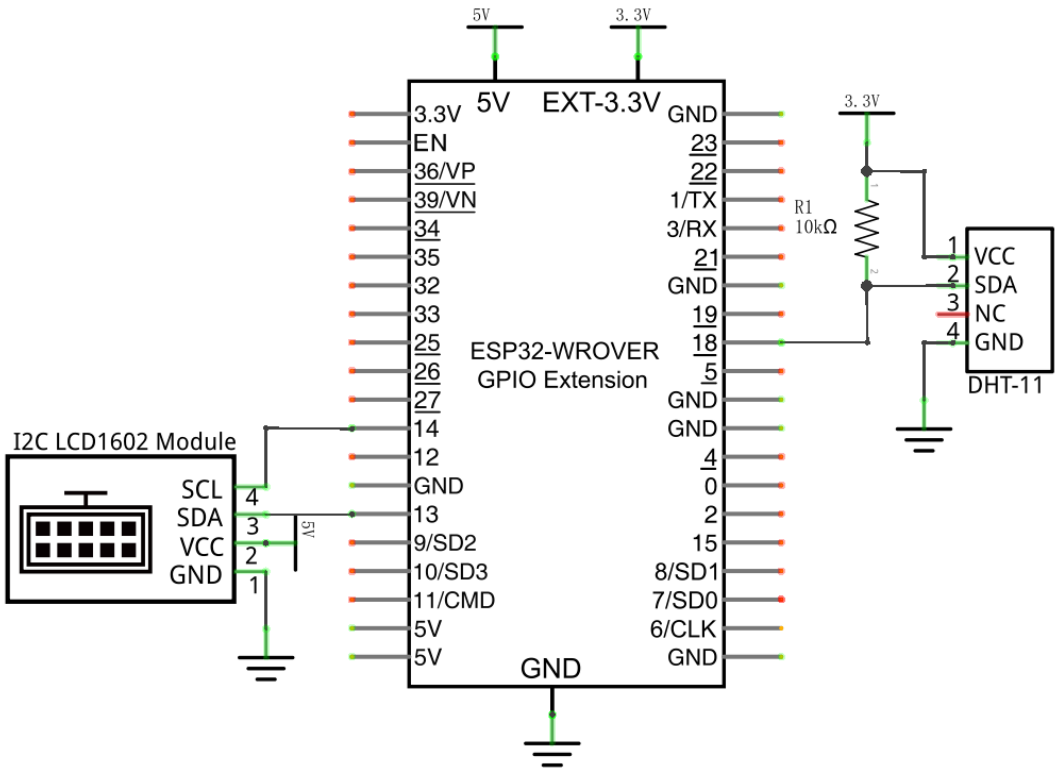
In this project, we use L2C-LCD1602 to display data collected by DHT11.

Component List

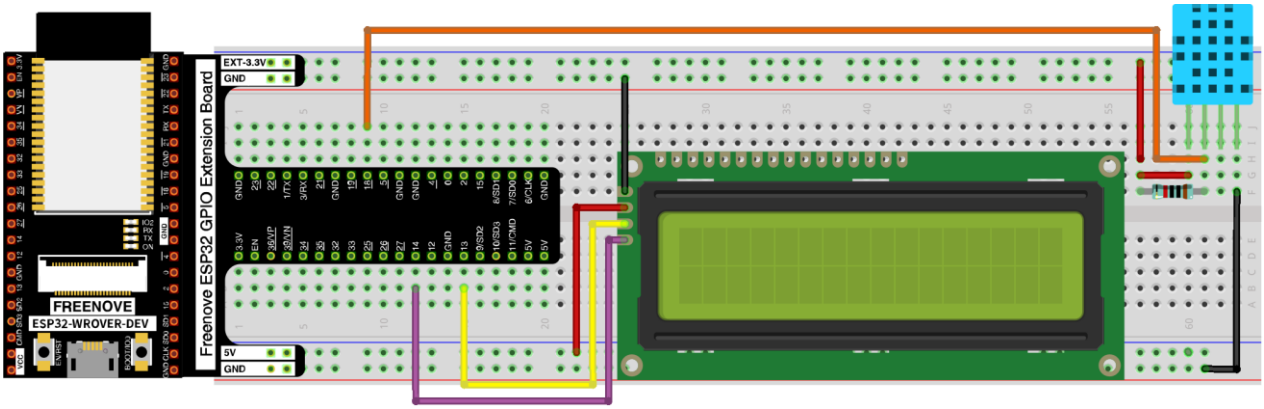
ESP32-WROVER x1 	GPIO Extension Board x1 
Breadboard x1 	
LCD1602 Module x1 	Resistor 10kΩ x1 
Jumper F/M x4 Jumper M/M x4 	DHT11 x1 

Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

This code uses the DHTesp and LiquidCrystal_I2C libraries, so make sure the relevant library files are added before writing the program.

Sketch_24.2_Temperature_and_Humidity_Sensor

Sketch_24.2_Temperature_and_Humidity_Sensor_I2C | Arduino IDE 2.0.4

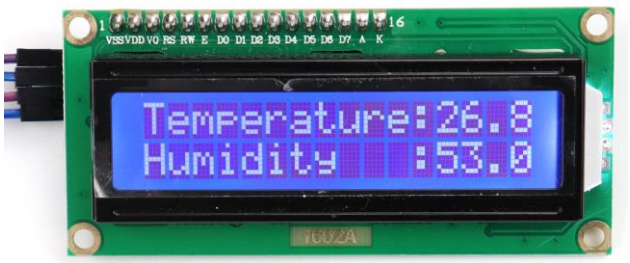
File Edit Sketch Tools Help

ESP32 Wrover Module

Sketch_24.2_Temperature_and_Humidity_Sensor_I2C.ino

```
/*********************************************************************
  Filename    : Temperature and Humidity Sensor
  Description : Use DHT11 to measure temperature and humidity.Print the result to the L
  Auther      : www.freenove.com
  Modification: 2020/07/11
*********************************************************************/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHTesp.h"

#define SDA 13                    //Define SDA pins
#define SCL 14                    //Define SCL pins

DHTesp dht;                       // create dht object
LiquidCrystal_I2C lcd(0x27,16,2); //initialize the LCD
int dhtPin = 18;                  // the number of the DHT11 sensor pin

void setup() {
  Wire.begin(SDA, SCL);           // attach the IIC pin
  if (!i2CAddrTest(0x27)) {
    lcd = LiquidCrystal_I2C(0x3F, 16, 2);
  }
  lcd.init();                     // LCD driver initialization
  lcd.backlight();                // Open the backlight
  dht.setup(dhtPin, DHTesp::DHT11); //attach the dht pin and initialize it
}

void loop() {
```

Download the code to ESP32-WROVER. The first line of LCD1602 shows the temperature value, and the second line shows the humidity value. Try to "pinch" the thermistor (without touching the leads) with your index finger and thumb for a brief time to observe the change in the LCD display value.



The following is the program code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHTesp.h"

#define SDA 13                    //Define SDA pins
#define SCL 14                    //Define SCL pins
DHTesp dht;                       // create dht object
LiquidCrystal_I2C lcd(0x27,16,2); //initialize the LCD
int dhtPin = 18;                  // the number of the DHT11 sensor pin

void setup() {
  Wire.begin(SDA, SCL);           // attach the IIC pin
  if (!i2CAddrTest(0x27)) {
    lcd = LiquidCrystal_I2C(0x3F, 16, 2);
  }
  lcd.init();                     // LCD driver initialization
  lcd.backlight();                // Open the backlight
  dht.setup(dhtPin, DHTesp::DHT11); //attach the dht pin and initialize it
}

void loop() {
  // read DHT11 data and save it
  flag:TempAndHumidity DHT = dht.getTempAndHumidity();
  if (dht.getStatus() != 0) {       //Determine if the read is successful, and if it fails, go back to flag and re-read the data
    goto flag;
  }
  lcd.setCursor(0, 0);              //set the cursor to column 0, line 1
  lcd.print("Temperature:");        //display the Humidity on the LCD1602
  lcd.print(DHT.temperature);
  lcd.setCursor(0, 1);              //set the cursor to column 0, line 0
  lcd.print("Humidity   :");        //display the Humidity on the LCD1602
  lcd.print(DHT.humidity);
  delay(2000);
}
bool i2CAddrTest(uint8_t addr) {
  Wire.begin();
  Wire.beginTransmission(addr);
  if (Wire.endTransmission() == 0) {
    return true;
  }
  return false;
}
```

First, add the library function header file.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHTesp.h"
```

Second, initialize the pins associated with the DHT11 sensor and I2C-LCD1602.

```
DHTesp dht;                        // create dht object
LiquidCrystal_I2C lcd(0x27,16,2); //initialize the LCD
int dhtPin = 18;                   // the number of the DHT11 sensor pin

void setup() {
  Wire.begin(SDA, SCL);            // attach the IIC pin
  if (!i2CAddrTest(0x27)) {
    lcd = LiquidCrystal_I2C(0x3F, 16, 2);
  }
  lcd.init();                      // LCD driver initialization
  lcd.backlight();                 // Open the backlight
  dht.setup(dhtPin, DHTesp::DHT11); //attach the dht pin and initialize it
}
```

Finally, the data of temperature and humidity sensor are obtained and displayed on LCD1602. The first row shows the temperature and the second shows the humidity.

```
flag:TempAndHumidity DHT = dht.getTempAndHumidity();
  if (dht.getStatus() ! = 0) {       //Determine if the reading is successful, and if it fails, go back to flag and re-read the data
    goto flag;
  }
  lcd.setCursor(0, 0);               //set the cursor to column 0, line 1
  lcd.print("Temperature:");         //display the Humidity on the LCD1602
  lcd.print(DHT.temperature);
  lcd.setCursor(0, 1);               //set the cursor to column 0, line 0
  lcd.print("Humidity   :");         //display the Humidity on the LCD1602
  lcd.print(DHT.humidity);
```