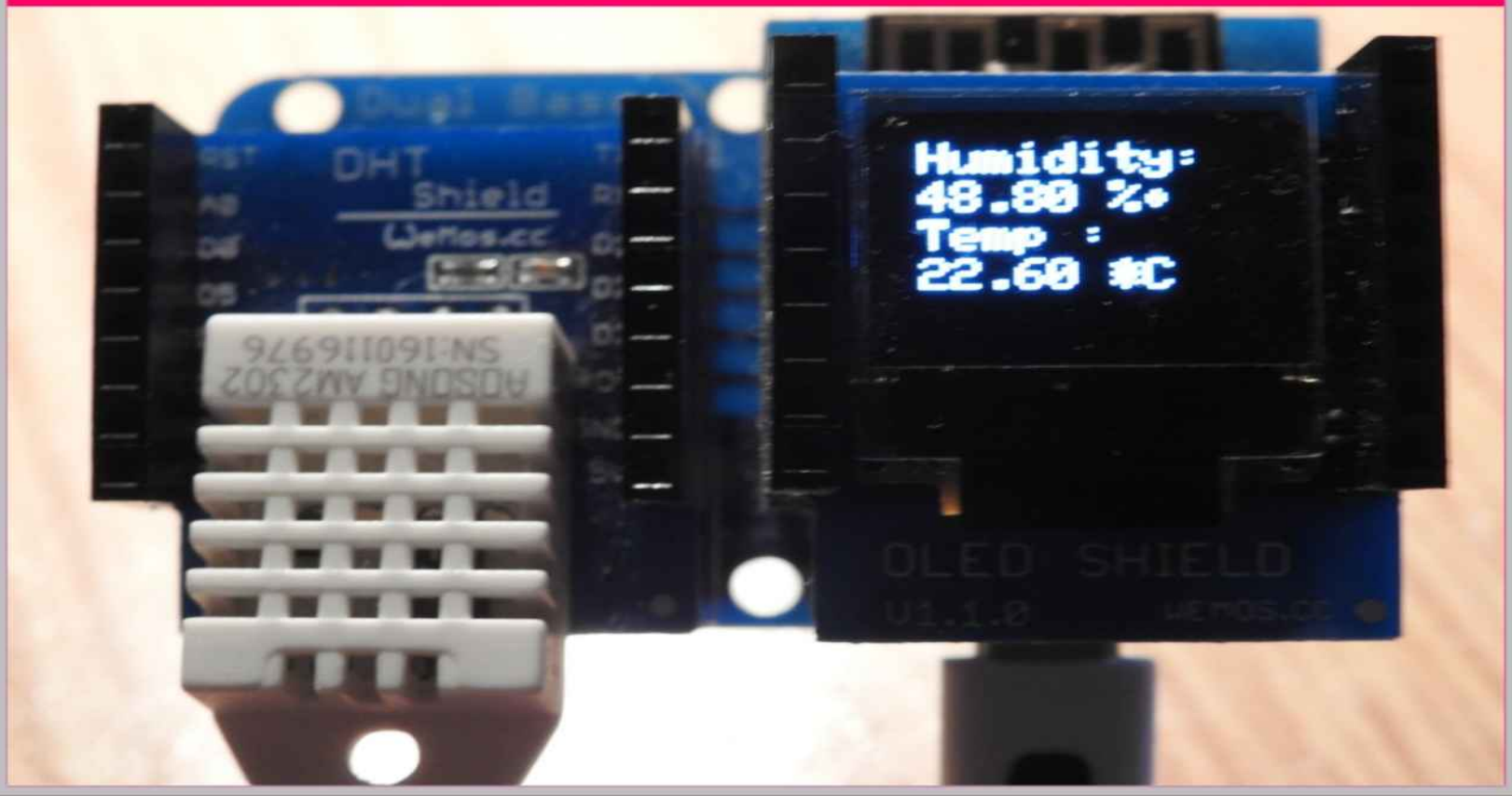
Learn about the ESP8266 using Wemos shields

IAIN HENDRY



Learn about the ESP8266 using Wemos shields

by

Iain Hendry

Contents

out the ESP8266

The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by manufacturer Espressif Systems in Shanghai, China.

The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. However, at first there was almost no English-language documentation on the chip and the commands it accepted.

The very low price and the fact that there were very few external components on the module, which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, chip, and the software on it, as well as to translate the Chinese documentation.

The ESP8285 is an ESP8266 with 1 MiB of built-in flash, allowing for single-chip devices capable of connecting to Wi-Fi

Processor: L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz

Memory:

32 KiB instruction RAM
32 KiB instruction cache RAM
80 KiB user-data RAM
16 KiB ETS system-data RAM
External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)

IEEE 802.11 b/g/n Wi-Fi

- Integrated TR switch, balun, LNA, power amplifier and matching network WEP or WPA/WPA2
- authentication, or open networks

16 GPIO pins
SPI
I²C (software implementation)[6]
I²S interfaces with DMA (sharing pins with GPIO)
UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
10-bit ADC (successive approximation ADC)

The pinout is as follows for the common ESP-01 module:

VCC, Voltage (+3.3 V; can handle up to 3.6 V)
GND, Ground (0 V)
RX, Receive data bit X
TX, Transmit data bit X
CH_PD, Chip power-down
RST, Reset
GPIO 0, General-purpose input/output No. 0
GPIO 2, General-purpose input/output No. 2

e Wemos Mini and shields

The WeMos D1 mini is a miniature wireless Wifi microcontroller development board. It uses the popular ESP8266 module and puts it into a fully fledged development board. Programming the D1 mini is as simple as programming any other Arduino based microcontroller as the module includes a built in micro USB interface allowing the module to be programmed directly from the Arduino IDE this does require that you add the ESP8266 support to via the board manager but that is a fairly easy task that we will show you how to do that in another chapter.

The D1 mini is designed to allow Wemos compatible shields to be plugged into the board in a similar way to the Arduino development board platform which greatly expands its capabilities. Included with the module is a set of headers that require soldering that allow the shields to be easily added or removed from the D1 mini.

Other features of the D1 Mini include

11 digital input/output pins, all pins have interrupt/pwm/I2C/one-wire supported (except D0)
1 analog input(3.2V max input)
a Micro USB connection
Compatible with MicroPython, Arduino, nodemcu

The Wemos D1 Mini exposes the serial port of the ESP8266 module via CH340 USB to UART chip which is also connected to the boot pins of the module, allowing for a seamless virtualization of the device. You will need to install the relevant driver for this chip on your operating system, I have tried several Windows 7 and Windows 10 systems and also a Mac mini and had no issues.

There are multiple Wemos Mini variants, the main differences are memory and connectors on the board

ting up the Arduino IDE

This is the suggested installation method for end users.

Prerequisites

Arduino 1.6.8, download this from Arduino website.
Internet connection

Instructions

Start Arduino IDE and open the Preferences window.
Enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into Additional Board Manager URLs field. You can add multiple URLs, separating them with commas.
Open Boards Manager from Tools > Board menu and find esp8266 platform.
Select the version you need from a drop-down box.
Click install button.
Don’t forget to select your ESP8266 board from Tools > Board menu after installation.

You may optionally use staging boards manager package link: http://arduino.esp8266.com/staging/package_esp8266com_index.json.

This may contain some new features, but at the same time, there may be problems that need fixed. I stick to the stable version.

;ic examples

sh an LED example

Description

This is the most basic example where we flash the on-board LED of the Wemos mini. We use the LED_BUILTIN macro which is connected to D4

Code Example

```
// the setup function runs once when you press reset or power the board
void setup()
{
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the
voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the
voltage LOW
  delay(1000);                       // wait for a second
}
```

Output

The LED will flash on and off

sic Wifi example

Description

This is a very basic example which connects your ESP8266 to Wifi, once you can do this there are many follow on projects that you can run.

Code Example

```
#include <ESP8266WiFi.h>

void setup()
{
  //init serial
  Serial.begin(115200);
  Serial.println();
  //change these
  WiFi.begin("username", "password");

  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {}
```

Output

Open the serial monitor and you should see something like this, your IP may be different

Connecting.....
Connected, IP address: 192.168.1.161

Scanner example

Description

This is a basic example which will display via the serial monitor the address of any devices connected to your Wemos.

A very useful code snippet to have in your collection.

Code Example

```
#include <Wire.h>

void setup()
{
  Wire.begin();

  Serial.begin(9600);
  Serial.println("\nI2C Scanner");
}

void loop()
{
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {

    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
```

```
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknow error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);           // wait 5 seconds for next scan
}
```

sic Webserver example

In this example we will create a basic webserver example using our Wemos, it will connect to your Wifi network and then you will navigate to a URL and a basic page will appear.

This page will display 2 links , one will switch on an led connected to D5 and the other option will switch the led off.

Parts

1 x Wemos D1 or D2
1 x USB cable
1 x LED and resistor or use a module

Code

Remember and add your own username and password

```
#include <ESP8266WiFi.h>

const char* ssid = "ssid name";
const char* password = "ssid password";

int ledPin = D5;
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  delay(10);


  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
```

```
  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.print("Use this URL : ");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");

}

void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()){
```

```
  delay(1);
}

// Read the first line of the request
String request = client.readStringUntil('\r');
Serial.println(request);
client.flush();

// Match the request

int value = LOW;
if (request.indexOf("/LED=ON") != -1) {
  digitalWrite(ledPin, HIGH);
  value = HIGH;
}
if (request.indexOf("/LED=OFF") != -1){
  digitalWrite(ledPin, LOW);
  value = LOW;
}



// Return the response
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(""); //  do not forget this one
client.println("<!DOCTYPE HTML>");
client.println("<html>");

client.print("Led pin is now: ");

if(value == HIGH) {
 client.print("On");
} else {
  client.print("Off");
}
client.println("<br><br>");
```

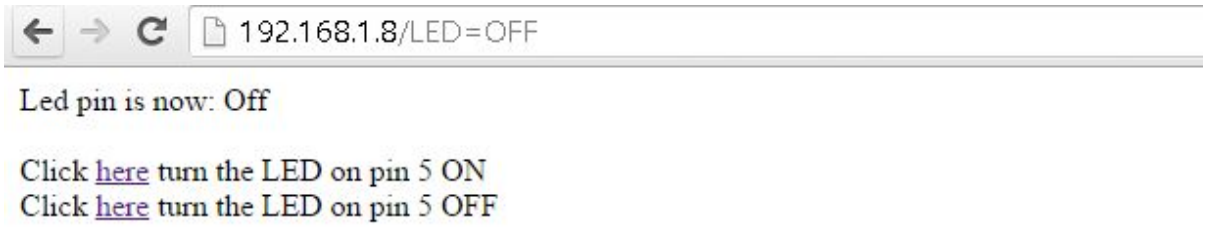
```
  client.println("Click <a href=\"/LED=ON\">here</a> turn the LED on pin 5 ON<br>");
  client.println("Click <a href=\"/LED=OFF\">here</a> turn the LED on pin 5 OFF<br>");
  client.println("</html>");

  delay(1);
  Serial.println("Client disconnected");
  Serial.println("");

}
```

Results

Open the serial monitor , all going well and you will see the IP address and messages like the following

```
Connecting to
.........
WiFi connected
Server started
Use this URL to connect: http://192.168.1.8/
```

Using your favourite web browser navigate to the IP above



I tried this out and it uses a static IP address

```
//This example will set up a static IP - in this case 192.168.1.99

#include <ESP8266WiFi.h>

const char* ssid = "ssid name";
const char* password = "ssid password";

int ledPin = D5;
WiFiServer server(80);
IPAddress ip(192, 168, 1, 99); // where xx is the desired IP Address
IPAddress gateway(192, 168, 1, 1); // set gateway to match your network

void setup() {
  Serial.begin(115200);
  delay(10);


  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);

  Serial.print(F("Setting static ip to : "));
  Serial.println(ip);

  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  IPAddress subnet(255, 255, 255, 0); // set subnet mask to match your
network
  WiFi.config(ip, gateway, subnet);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.print("Use this URL : ");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");

}

void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }

  // Read the first line of the request
  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();

  // Match the request

  int value = LOW;
  if (request.indexOf("/LED=ON") != -1) {
```

```
    digitalWrite(ledPin, HIGH);
    value = HIGH;
  }
  if (request.indexOf("/LED=OFF") != -1){
    digitalWrite(ledPin, LOW);
    value = LOW;
  }



  // Return the response
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println(""); //  do not forget this one
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");

  client.print("Led pin is now: ");

  if(value == HIGH) {
    client.print("On");
  } else {
    client.print("Off");
  }
  client.println("<br><br>");
  client.println("Click <a href=\"/LED=ON\">here</a> turn the LED on pin
5 ON<br>");
  client.println("Click <a href=\"/LED=OFF\">here</a> turn the LED on
pin 5 OFF<br>");
  client.println("</html>");

  delay(1);
  Serial.println("Client disconnected");
  Serial.println("");

}
```

ad and write to the eeprom

Description

The ESP8266 has 512 bytes of internal EEPROM, this could be useful if you need to store some settings, such as IP address or Wifi details

Code

The write example first

```
#include <EEPROM.h>

int addr = 0;

void setup()
{
  EEPROM.begin(512);  //Initialize EEPROM

  // write to EEPROM.
  EEPROM.write(addr, 'a');
  addr++;                    //Increment address
  EEPROM.write(addr, 'b');
  addr++;                    //Increment address
  EEPROM.write(addr, 'C');

  //Write string to eeprom
  String sample = "testing eeprom";
  for(int i=0;i<sample.length();i++)
  {
    EEPROM.write(0x0F+i, sample[i]); //Write one by one with starting address of 0x0F
  }
  EEPROM.commit();    //Store data to EEPROM
}

void loop()
```

```
{
}
```

And now the read example

```
#include <EEPROM.h>

int addr = 0;

void setup()
{
  EEPROM.begin(512);  //Initialize EEPROM
  Serial.begin(9600);
  Serial.println("");
  Serial.print(char(EEPROM.read(addr)));
  addr++;                    //Increment address
  Serial.print(char(EEPROM.read(addr)));
  addr++;                    //Increment address
  Serial.println(char(EEPROM.read(addr)));

  //Read string from eeprom (testing eeprom)
  String strText;
  for(int i=0;i<14;i++)
  {
    strText = strText + char(EEPROM.read(0x0F+i)); //Read one by one with
starting address of 0x0F
  }

  Serial.print(strText);  //Print the text
}

void loop()
{
}
```

Output

Open the serial monitor

abC
testing eeprom

P specific APIs

Description

Some ESP-specific APIs related to deep sleep, RTC and flash memories are available in the ESP object. These may be useful, we have already looked at the restart one

Code

```
/*
ESP.restart() restarts the CPU.
ESP.getResetReason() returns a String containing the last reset reason in human readable format.
ESP.getFreeHeap() returns the free heap size.
ESP.getHeapFragmentation() returns the fragmentation metric (0% is clean, more than ~50% is not harmless)
ESP.getMaxFreeBlockSize() returns the maximum allocatable ram block regarding heap fragmentation
ESP.getChipId() returns the ESP8266 chip ID as a 32-bit integer.
ESP.getCoreVersion() returns a String containing the core version.
ESP.getSdkVersion() returns the SDK version as a char.
ESP.getCpuFreqMHz() returns the CPU frequency in MHz as an unsigned 8-bit integer.
ESP.getSketchSize() returns the size of the current sketch as an unsigned 32-bit integer.
ESP.getFreeSketchSpace() returns the free sketch space as an unsigned 32-bit integer.
ESP.getSketchMD5() returns a lowercase String containing the MD5 of the current sketch.
ESP.getFlashChipId() returns the flash chip ID as a 32-bit integer.
ESP.getFlashChipSize() returns the flash chip size, in bytes, as seen by the SDK (may be less than actual size).
ESP.getFlashChipRealSize() returns the real chip size, in bytes, based on
```

the flash chip ID.
ESP.getFlashChipSpeed(void) returns the flash chip frequency, in Hz.
ESP.getCycleCount() returns the cpu instruction cycle count since start as an unsigned 32-bit. This is useful for accurate timing of very short actions like bit banging.
ESP.getVcc() may be used to measure supply voltage. ESP needs to reconfigure the ADC at startup in order for this feature to be available. Add the following line to the top of your sketch to use
*/

```
void setup() {

  Serial.begin(115200);

  Serial.print("SDK version:");
  Serial.println(ESP.getSdkVersion());

  Serial.print("Core version:");
  Serial.println(ESP.getCoreVersion());

  Serial.print("CPU ferquency:");
  Serial.println(ESP.getCpuFreqMHz());

  Serial.print("Chip ID:");
  Serial.println(ESP.getChipId());

  Serial.print("Reset Reason:");
  Serial.println(ESP.getResetReason());

  Serial.print("Free heap:");
  Serial.println(ESP.getFreeHeap());

}

void loop() {}
```

Output

Open the serial monitor

SDK version:2.2.1(cfd48f3)
Core version:2_4_2
CPU ferquency:80
Chip ID:1947093
Reset Reason:External System
Free heap:51880

ook at SHA-1

Description

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value known as a message digest – typically rendered as a hexadecimal number, 40 digits long

more - https://en.wikipedia.org/wiki/SHA-1

Code

```
#include "Hash.h"

void setup()
{
  Serial.begin(115200);

  String result = sha1("testing sha1");

  Serial.println();
  Serial.print(result);

}

void loop() {}
```

Output

I saw the following in the serial monitor

47d4387d5b2dcf196e295d48219b2c535c023eea

e ticker library

```
/*
  Basic Ticker usage
  Ticker is an object that will call a given function with a certain period.
  Each Ticker calls one function. You can have as many Tickers as you like,
  memory being the only limitation.
  A function may be attached to a ticker and detached from the ticker.
  There are two variants of the attach function: attach and attach_ms.
  The first one takes period in seconds, the second one in milliseconds.
  The built-in LED will be blinking.
*/

#include <Ticker.h>

Ticker flipper;

int count = 0;

void flip() {
  int state = digitalRead(LED_BUILTIN);  // get the current state of GPIO1
pin
  digitalWrite(LED_BUILTIN, !state);     // set pin to the opposite state

  ++count;
  // when the counter reaches a certain value, start blinking like crazy
  if (count == 20) {
    flipper.attach(0.1, flip);
  }
  // when the counter reaches yet another value, stop blinking
  else if (count == 120) {
    flipper.detach();
  }
}
```

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  // flip the pin every 0.3s
  flipper.attach(0.3, flip);
}

void loop() {
}
```

npling analog data

Description

Sampling data is particularly useful for analog sensors such as an LDR or a thermistor. Another good example is one of the MQ gas sensors that can be bought which take a period of time to 'warm up'.

You can change the amount of samples to take and the interval which is in millioseconds Here is a simple code example which shows this technique

Code

```
int sampleData(int analogPin, int samples = 10, int sampleInterval = 100)
{
int sampleData[samples];
int val = 0;
for (int i = 0; i<samples; i++)
{
sampleData[i] = analogRead(analogPin);
val = val + sampleData[i];
delay(sampleInterval);
}
val = (val / samples);
return map(val, 550, 1023, 100, 0);
}
```

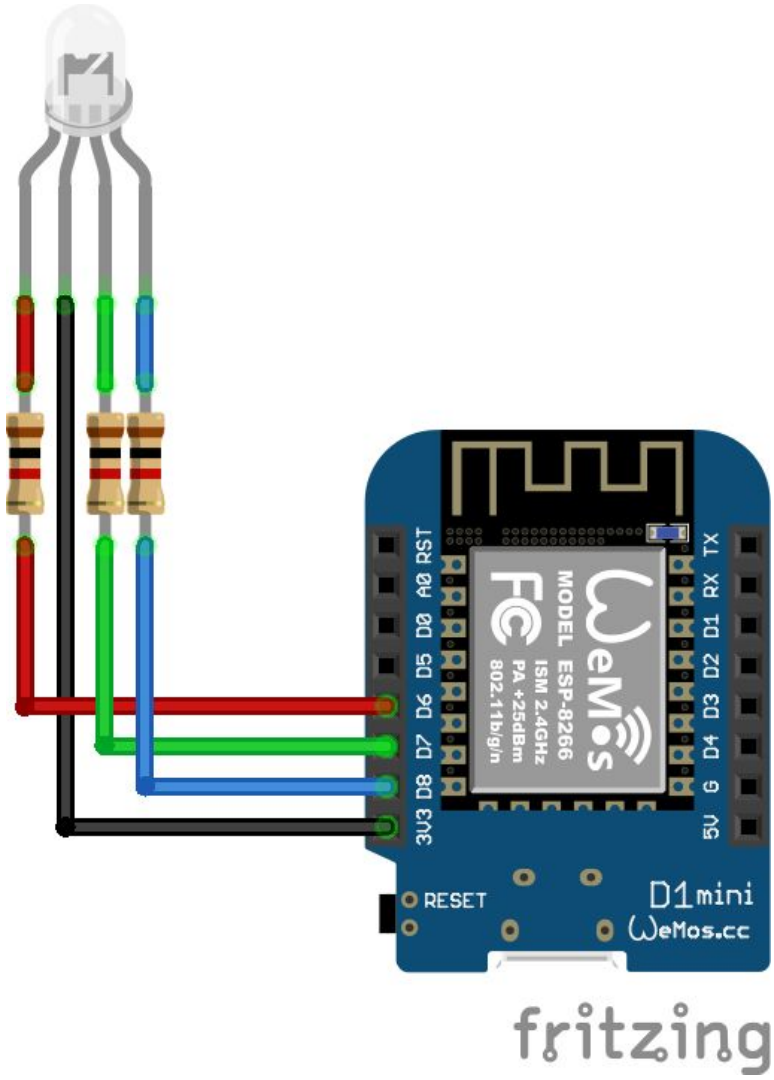
ntrol a Wemos using an application on your PC

Description

In this example we will control an RGB led connected to our Wemos Mini from an application running on our PC. In this example I will use VB.net to write the application but you can use any language that has the ability to use the serial port. I wanted to have a GUI but you can easily have a command line, also the app could be created in C#. I don't really have a preference in that area. I just decided to do it in VB.net for a change rather than C#

Layout

Here is the layout, the LED is connected to pins 6, 7 and 8 of the Wemos Mini and we use 3v3 to power it. I had an RGB led breakout, common anode type



Arduino Code

The Arduino part is quite straightforward, basically it listens for a character on the serial port and then performs an action based on the character recieved. Our PC application will send numbers 1 to 6, each of these numbers will tie in with either switching an individual LED on or off on the RGB led.

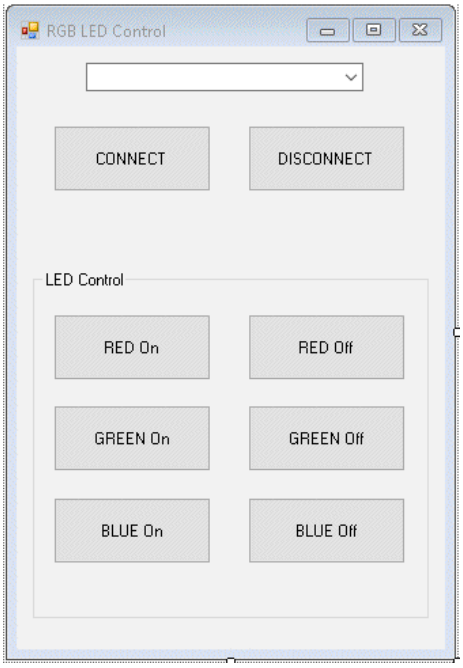
```
int redPin = D6;
int greenPin = D7;
int bluePin = D8;

void setup()
```

```
{
  // declare the serial comm at 9600 baud rate
  Serial.begin(9600);

  // output pins
  pinMode(redPin, OUTPUT); // red
  pinMode(greenPin, OUTPUT); // green
  pinMode(bluePin, OUTPUT); // blue
  digitalWrite(redPin, HIGH);
  digitalWrite(greenPin, HIGH);
  digitalWrite(bluePin, HIGH);
}

void loop()
{
  // call the returned value from GetFromSerial() function
  switch(GetFromSerial())
  {
  case '1':
   digitalWrite(redPin, LOW);
    break;
  case '2':
    digitalWrite(redPin, HIGH);
    break;
  case '3':
    digitalWrite(greenPin, LOW);
    break;
  case '4':
    digitalWrite(greenPin, HIGH);
    break;
  case '5':
   digitalWrite(bluePin, LOW);
    break;
  case '6':
    digitalWrite(bluePin, HIGH);
    break;
  }
```

```
}

// read the serial port
int GetFromSerial()
{
  while (Serial.available()<=0) {
  }
  return Serial.read();
}
```

Application

This was written in Visual Studio 2010. I wanted to have a combo box with a list of com ports, the user would then select the correct one and press a button to connect, there would also be a button to disconnect. There would be 6 more buttons on the form these would switch the red, green and blue leds on and off This is the GUI I designed in Visual Studio



Now for the code for the application

```
Imports System.IO.Ports

Public Class frmRGBLed
    Dim WithEvents serialPort As New SerialPort


    Private Sub btnConnect_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnConnect.Click
        Try
            serialPort.BaudRate = 9600
            serialPort.PortName = cboPorts.SelectedItem.ToString
            serialPort.Parity = Parity.None
            serialPort.DataBits = 8
            serialPort.StopBits = 1
            serialPort.Open()
            serialPort.Encoding = System.Text.Encoding.Default
            If serialPort.IsOpen Then
                btnConnect.Visible = False
                cboPorts.Enabled = False
                btnDisconnect.Visible = True
            End If
        Catch
            serialPort.Close()
        End Try
    End Sub

    Private Sub btnDisconnect_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnDisconnect.Click
        Try
            serialPort.Close()
            btnConnect.Visible = True
            btnDisconnect.Visible = False
            cboPorts.Enabled = True
            Exit Sub
        Catch
            MessageBox.Show("Problem closing port")
        End Try
```

```
    End Sub

    Private Sub blnRedOn_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles blnRedOn.Click
        serialPort.Write("1")
    End Sub

    Private Sub btnRedOff_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnRedOff.Click
        serialPort.Write("2")
    End Sub

    Private Sub btnGreenOn_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnGreenOn.Click
        serialPort.Write("3")
    End Sub

   Private Sub btnGreenOff_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnGreenOff.Click
        serialPort.Write("4")
    End Sub

    Private Sub btmBlueOn_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btmBlueOn.Click
        serialPort.Write("5")
    End Sub

    Private Sub btnBlueOff_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnBlueOff.Click
        serialPort.Write("6")
    End Sub


    Private Sub GetSerialPortNames()
        For Each sport As String In My.Computer.Ports.SerialPortNames
            cboPorts.Items.Add(sport)
        Next
```

```
    End Sub

    Private Sub frmRGBLed_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
        Try
            GetSerialPortNames()
            cboPorts.SelectedIndex = 0
        Catch
            MsgBox("No ports connected.")
        End Try
    End Sub

End Class
```

You could add any other functionality you wanted, maybe a status bar. Automatic detection of the Wemos rather than using a combobox and buttons

Links

The complete code is in the following github location, I always find it easier to check out the code on your own machine.

There is also the fritzing schematics and code for the Arduino IDE

https://github.com/esp8266learning/projects/tree/master/Wemos%20RGB%20Led

ook at SPIFFS

The following diagram illustrates flash layout used in Arduino environment:

File system size depends on the flash chip size. Depending on the board which is selected in IDE, you have the following options for flash size:

Board	**Flash chip size, bytes**	**File system size, bytes**
Generic module	512k	64k, 128k
Generic module	1M	64k, 128k, 256k, 512k
Generic module	2M	1M
Generic module	4M	1M, 2M, 3M
Adafruit HUZZAH	4M	1M, 2M, 3M
ESPresso Lite 1.0	4M	1M, 2M, 3M
ESPresso Lite 2.0	4M	1M, 2M, 3M
NodeMCU 0.9	4M	1M, 2M, 3M
NodeMCU 1.0	4M	1M, 2M, 3M

Board	Flash chip size, bytes	File system size, bytes
Olimex MOD-WIFI-ESP8266(-DEV)	2M	1M
SparkFun Thing	512k	64k
SweetPea ESP-210	4M	1M, 2M, 3M
WeMos D1 R1, R2 & mini	4M	1M, 2M, 3M
ESPDuino	4M	1M, 2M, 3M
WiFiduino	4M	1M, 2M, 3M

The filesystem implementation for ESP8266 had to accomodate the constraints of the chip, among which its limited RAM. SPIFFS was selected because it is designed for small systems, but that comes at the cost of some simplifications and limitations.

First, behind the scenes, SPIFFS does not support directories, it just stores a “flat” list of files. But contrary to traditional filesystems, the slash character '/' is allowed in filenames, so the functions that deal with directory listing (e.g. openDir("/website")) basically just filter the filenames and keep the ones that start with the requested prefix (/website/). Practically speaking, that makes little difference though.

Second, there is a limit of 32 chars in total for filenames. One '\0' char is reserved for C string termination, so that leaves us with 31 usable characters.

Combined, that means it is advised to keep filenames short and not use deeply nested directories, as the full path of each file (including directories, '/' characters, base name, dot and extension) has to be 31 chars at a maximum. For example, the filename /website/images/bird_thumbnail.jpg is 34 chars and will cause some problems if used, for example in exists() or in case another file starts with the same first 31 characters.

ESP8266FS is a tool which integrates into the Arduino IDE. It adds a menu item to Tools menu for uploading the contents of sketch data directory into ESP8266 flash file system.

Warning: Due to the move from the obsolete esptool-ck.exe to the supported esptool.py upload tool, upgraders from pre 2.5.1 will need to update the ESP8266FS tool referenced below to 0.4.0 or later. Prior versions will fail with a “esptool not found” error because they don’t know how to use esptool.py.

Download the tool: https://github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.4.0/ESP8266FS-0.4.0.zip
In your Arduino sketchbook directory, create tools directory if it doesn’t exist yet
Unpack the tool into tools directory (the path will look like <home_dir>/Arduino/tools/ESP8266FS/tool/esp8266fs.jar) If upgrading, overwrite the existing JAR file with the newer version.
Restart Arduino IDE
Open a sketch (or create a new one and save it)
Go to sketch directory (choose Sketch > Show Sketch Folder)
Create a directory named data and any files you want in the file system there
Make sure you have selected a board, port, and closed Serial Monitor

Select Tools > ESP8266 Sketch Data Upload. This should start uploading the files into ESP8266 flash file system. When done, IDE status bar will display SPIFFS Image Uploaded message.

Code

```
#include <FS.h>

const char* filename = "/testfile.txt";

void setup()
{
  //init serial
  Serial.begin(115200);
  Serial.println();
  //Initialize File System
  if(SPIFFS.begin())
  {
    Serial.println("SPIFFS Initialize....ok");
  }
  else
  {
    Serial.println("SPIFFS Initialization...failed");
  }

  //Format File System
  if(SPIFFS.format())
  {
    Serial.println("File System Formated");
  }
  else
  {
    Serial.println("File System Formatting Error");
  }
```

```
  //Create New File And Write Data to It
  File f = SPIFFS.open(filename, "w");

  if (!f)
  {
    Serial.println("file open failed");
  }
  else
  {
      //Write data to file
      Serial.println("Writing Data to File");
      f.print("This is sample data which is written to sample file");
      f.close();  //Close file
  }

}

void loop()
{
  int i;

  //Read File data
  File f = SPIFFS.open(filename, "r");

  if (!f)
  {
    Serial.println("file open failed");
  }
  else
  {
      Serial.println("Reading Data from File:");
      //Data from file
```

```
    for(i=0;i<f.size();i++) //Read upto complete file size
    {
      Serial.print((char)f.read());
    }
    f.close();  //Close file
    Serial.println("File Closed");
  }
  delay(5000);
}
```

Output

Open the serial monitor

```
SPIFFS Initialize....ok
File System Formated
Writing Data to File
Reading Data from File:
This is sample data which is written to sample fileFile Closed
```

up your ESP8266 as an Action point

An access point (AP) is a device that provides access to Wi-Fi network to other devices (stations) and connects them further to a wired network. ESP8266 can provide similar functionality except it does not have interface to a wired network. Such mode of operation is called soft access point (soft-AP). The maximum number of stations that can simultaneously be connected to the soft-AP can be set from 0 to 8, but defaults to 4.

The soft-AP mode is often used and an intermediate step before connecting ESP to a Wi-Fi in a station mode. This is when SSID and password to such network is not known upfront. ESP first boots in soft-AP mode, so we can connect to it using a laptop or a mobile phone. Then we are able to provide credentials to the target network. Once done ESP is switched to the station mode and can connect to the target Wi-Fi.

Another handy application of soft-AP mode is to set up mesh networks. ESP can operate in both soft-AP and Station mode so it can act as a node of a mesh network.

To configure the ESP8266 as an access point, to allow other devices to connect to it, you can use the softAP function

Code

```
#include <ESP8266WiFi.h>        // Include the Wi-Fi library

const char *ssid = "ESP8266 Access Point"; // The name of the Wi-Fi network that will be created
```

```
const char *password = "thereisnospoon";   // The password
required to connect to it, leave blank for an open network

void setup() {
  Serial.begin(115200);
  delay(10);
  Serial.println('\n');

  WiFi.softAP(ssid, password);             // Start the access point
  Serial.print("Access Point \"");
  Serial.print(ssid);
  Serial.println("\" started");

  Serial.print("IP address:\t");
  Serial.println(WiFi.softAPIP());         // Send the IP address of the
ESP8266 to the computer
}

void loop() { }
```

To see if it works, open the Wi-Fi settings on your computer, look for a network called "ESP8266 Access Point", enter the password "thereisnospoon", and connect to it. Then open a terminal, and ping to 192.168.4.1 (this is the default IP address of our ESP AP). You'll see that the ESP responds to your pings.

ɜic examples with shields

In these examples we will perform some basic tasks with various shields, we will only connect one shield to a wemos mini.

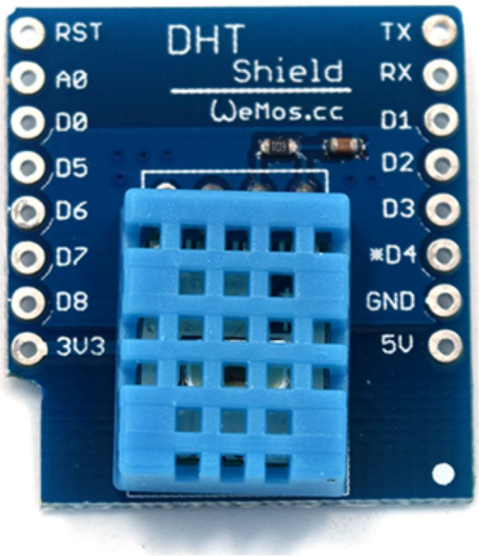
IT Shield example

Description

In this example we look at the DHT shield, there are several variations of this shield. The older one features the DHT11, newer variants use the DHT12 which is a slightly better device

The DHT Shield version 1 is a temperature and humidity sensor that uses the DHT11 sensor. The temperature range is 0 to 60C and the humidity range is 20 90%RH

The temperature and humidity data read from the sensor is the result of the last measurement. The sensor reading interval should be more than 2 seconds to ensure accurate measurements. The v1 sensor uses D4 on the D1 mini so this would be unavailable for other devices.



Code Example

There are several libraries you can use which do most of the hard work for you, this uses the Adafruit one.

```
/* DHT Shield - Simple
*
* Example testing sketch for various DHT humidity/temperature sensors
* Written by ladyada, public domain
*
* Depends on Adafruit DHT Arduino library
* https://github.com/adafruit/DHT-sensor-library
*/

#include "DHT.h"

#define DHTPIN D4     // what pin we're connected to
#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
Serial.println("DHTxx test!");

  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow
sensor)
  float h = dht.readHumidity();
```

```
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(f);
  Serial.print(" *F\t");
  Serial.print("Heat index: ");
  Serial.print(hic);
  Serial.print(" *C ");
  Serial.print(hif);
  Serial.println(" *F");
}
```

Output

Open the serial monitor and you should see something like this all going well

Humidity: 20.00 % Temperature: 21.00 *C 69.80 *F Heat index: 19.68 *C 67.42 *F
Humidity: 20.00 % Temperature: 22.00 *C 71.60 *F Heat index: 20.78 *C 69.40 *F
Humidity: 20.00 % Temperature: 22.00 *C 71.60 *F Heat index: 20.78 *C 69.40 *F
Humidity: 20.00 % Temperature: 22.00 *C 71.60 *F Heat index: 20.78 *C 69.40 *F
Humidity: 20.00 % Temperature: 22.00 *C 71.60 *F Heat index: 20.78 *C 69.40 *F
Humidity: 19.00 % Temperature: 23.00 *C 73.40 *F Heat index: 21.85 *C 71.33 *F

e 1 button shield

Description

A simple shield that comprises a button that uses the D3 pin, so this would be unavailable for other devices. Here is a picture of the shield.

Code Example

```
const int buttonPin = D3;
const int ledPin = BUILTIN_LED;

int buttonState = 0;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);

  // set initial state, LED off
  digitalWrite(ledPin, buttonState);
}

void loop() {
  // read button state, HIGH when pressed, LOW when not
  buttonState = digitalRead(buttonPin);

  // if the push button pressed, switch on the LED
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);  // LED on
```

```
  } else {
    digitalWrite(ledPin, LOW); // LED off
  }
}
```

Output

S2812B RGB Shield

Description

The WS2812 is a intelligent control LED light source that the control circuit and RGB chip are integrated in a package of 5050 components. It internal include intelligent digital port data latch and signal reshaping amplification drive circuit. Also include a precision internal oscillator and a 12V voltage programmable constant current control part, effectively ensuring the pixel point light color height consistent.

The data transfer protocol use single NZR communication mode. After the pixel power-on reset, the DIN port receive data from controller, the first pixel collect initial 24bit data then sent to the internal data latch, the other data which reshaping by the internal signal reshaping amplification circuit sent to the next cascade pixel through the DO port. After transmission for each pixel， the signal to reduce 24bit. pixel adopt auto reshaping transmit technology, making the pixel cascade number is not limited the signal transmission, only depend on the speed of signal transmission.

LED with low driving voltage, environmental protection and energy saving, high brightness, scattering angle is large, good consistency, low power, long life and other advantages. The control chip integrated in LED above becoming more simple circuit, small volume, convenient installation.

These are also sold as Neopixels


WS2812B RGB
SHIELD
WEMOS.CC
RST
A0
D0
D5
D6
D7
D8
3V3
TX
RX
D1
D2
D3
D4
GND
5V
WeMos

Code Examples

You need to install the Adafruit Neopixel library, this basic example will flash red, green and blue colours individually for 1 second

Example 1

```
#include <Adafruit_NeoPixel.h>

#define PIN          D2

// When we setup the NeoPixel library, we tell it how many pixels, and which pin to use to send signals.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(1, PIN, NEO_GRB + NEO_KHZ800);

void setup()
{
  pixels.begin(); // This initializes the NeoPixel library.
}

void loop()
{
  // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
  pixels.setPixelColor(0, pixels.Color(2, 0, 0)); // red
  pixels.show(); // This sends the updated pixel color to the hardware.
  delay(1000); // Delay for a period of time (in milliseconds).
pixels.setPixelColor(0, pixels.Color(0, 2, 0)); // green
pixels.show(); // This sends the updated pixel color to the hardware.
  delay(1000); // Delay for a period of time (in milliseconds).
  pixels.setPixelColor(0, pixels.Color(0, 0, 2)); // blue
  pixels.show(); // This sends the updated pixel color to the hardware.
  delay(1000); // Delay for a period of time (in milliseconds).
}
```

Example 2

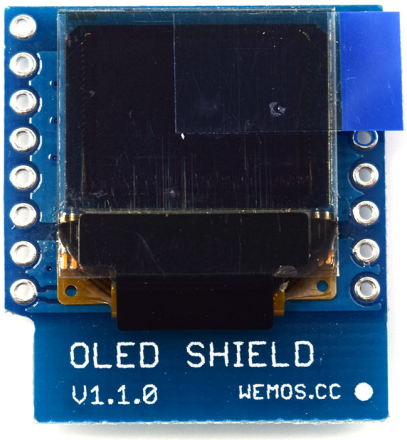
This example will create some random numbers from 0 to 4 and then flash the RGB LED with the values.

```
#include <Adafruit_NeoPixel.h>

#define PIN           D2

long randomRed;
long randomGreen;
long randomBlue;
// When we setup the NeoPixel library, we tell it how many pixels, and which pin to use to send signals.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(1, PIN, NEO_GRB + NEO_KHZ800);

void setup()
{
  pixels.begin(); // This initializes the NeoPixel library.
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop()
{
  // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
  randomRed = random(0, 4);
  randomGreen = random(0, 4);
  randomBlue = random(0, 4);
  pixels.setPixelColor(0, pixels.Color(randomRed, randomGreen, randomBlue)); // red
  pixels.show(); // This sends the updated pixel color to the hardware.
  delay(1000); // Delay for a period of time (in milliseconds).
}
```

,ED Shield

Description

In this example we look at another terrific little low cost shield for the Wemos mini, this time its the OLED shield. Lets look at the shield and some specs



- **Screen Size:** 64x48 pixels (0.66” Across)
- **Operating Voltage:** 3.3V
- **Driver IC:** SSD1306
- **Interface:** IIC(I2C)
- **IIC Address:** 0x3C or 0x3D

The shield uses the I2C pins, so you can still connect another I2C device (if it uses a different address) and the other pins are available

D1 mini	**Shield**
D1	SCL
D2	SDA

Code Example

You will need to add the https://github.com/mcauser/Adafruit_SSD1306 library The following code example is a simple hello world type example

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// SCL GPIO5
// SDA GPIO4
#define OLED_RESET 0  // GPIO0
Adafruit_SSD1306 display(OLED_RESET);

#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2


#define LOGO16_GLCD_HEIGHT 16
#define LOGO16_GLCD_WIDTH  16


void setup()  {
  Serial.begin(9600);

  // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3C (for the 64x48)
  // init done

display.display();
```

```
  delay(2000);

  // Clear the buffer.
  display.clearDisplay();

  // text display tests
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.println("Hello, world!");
  display.display();
  delay(2000);
  display.clearDisplay();

}


void loop() {

}
```

rometric Pressure Shield

Description

This shield uses a HP303B, here is some information about the sensor

The HP303B is a miniaturized Digital Barometric Air Pressure Sensor with a high accuracy and a low current consumption, capable of measuring both pressure and temperature.

The pressure sensor element is based on a capacitive sensing principle which guarantees high precision during temperature changes. The small package makes the HP303B ideal for mobile applications and wearable devices.

The internal signal processor converts the output from the pressure and temperature sensor elements to 24 bit results. Each unit is individually calibrated, the calibration coeicients calculated during this process are stored in the calibration registers. The coeicients are used in the application to convert the measurement results to high accuracy pressure and temperature values.

The result FIFO can store up to 32 measurement results, allowing for a reduced host processor polling rate. Sensor measurements and calibration coeicients are available through the serial I2C or SPI interface. The measurement status is indicated by status bits or interrupts on the SDO pin.

Features

• Operation range: Pressure: 300 –1200 hPa. Temperature: -40 – 85 °C.
• Pressure sensor precision: ± 0.005 hPa (or ±0.05 m) (high precision mode).
• Relative accuracy: ± 0.06 hPa (or ±0.5 m)
• Absolute accuracy: ± 1 hPa (or ±8 m)
• Temperature accuracy: ± 0.5°C.
• Pressure temperature sensitivity: 0.5Pa/K
• Measurement time: Typical: 27.6 ms for standard mode (16x). Minimum: 3.6 ms for low precision mode.
• Average current consumption: 1.7 µA for Pressure Measurement, 1.5uA for Temperature measurement @1Hz sampling rate, Standby: 0.5 µA.
• Supply voltage: VDDIO: 1.2 – 3.6 V, VDD: 1.7 – 3.6 V.
• Operating modes: Command (manual), Background (automatic), and Standby.
• Calibration: Individually calibrated with coeicients for measurement correction.
• FIFO: Stores up to 32 pressure or temperature measurements.
• Interface: I2C and SPI (both with optional interrupt)

Code Example

The example requires the https://github.com/wemos/LOLIN_HP303B_Library This examples works OK

```
include <LOLIN_HP303B.h>

    // HP303B Opject
    LOLIN_HP303B HP303BPressureSensor;
```

```
void setup()
{
  Serial.begin(115200);
  while (!Serial);
  HP303BPressureSensor.begin();
  Serial.println("Init complete!");
}


void loop()
{
  int32_t temperature;
  int32_t pressure;
  int16_t oversampling = 7;
  int16_t ret;
  Serial.println();

  ret = HP303BPressureSensor.measureTempOnce(temperature,
oversampling);

  if (ret != 0)
  {
    //Something went wrong.
    //Look at the library code for more information about return codes
    Serial.print("FAIL! ret = ");
    Serial.println(ret);
  }
  else
  {
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println(" degrees of Celsius");
  }

  //Pressure measurement behaves like temperature measurement
  //ret = HP303BPressureSensor.measurePressureOnce(pressure);
  ret = HP303BPressureSensor.measurePressureOnce(pressure,
```

```
oversampling);
  if (ret != 0)
  {
    //Something went wrong.
    //Look at the library code for more information about return codes
    Serial.print("FAIL! ret = ");
    Serial.println(ret);
  }
  else
  {
    Serial.print("Pressure: ");
    Serial.print(pressure);
    Serial.println(" Pascal");
  }

  //Wait some time
  delay(500);
}
```

Output

)pen the serial monitor and you should see something like this

```
Temperature: 31 degrees of Celsius
Pressure: 100469 Pascal

Temperature: 31 degrees of Celsius
Pressure: 100471 Pascal

Temperature: 31 degrees of Celsius
Pressure: 100470 Pascal
```

T 2.4 Touch Shield

Description

This is a 2.4 inch TFT shield for a Wemos, its a large shield thats more than twice the size of a D1 mini but its an excellent little TFT screen and there are a couple of example to let you get up and running quick unlike some of the Arduino TFT shields which I have had trouble with in the past This is a picture of the shield

Features

- 2.4” diagonal LCD TFT display
- 320×240 pixels
- TFT Driver IC: ILI9341
- Touch Screen controller IC: XPT2046

Code Example

You need to install various libraries – https://github.com/adafruit/Adafruit-GFX-Library, https://github.com/adafruit/Adafruit_ILI9341 and https://github.com/PaulStoffregen/XPT2046_Touchscreen Libraries

Touchscreen example

```
#include <SPI.h>
#include <XPT2046_Touchscreen.h>

#define TS_CS D3 //for D1 mini or TFT I2C Connector Shield (V1.1.0 or later)
// #define TS_CS  12 //for D32 Pro

XPT2046_Touchscreen ts(TS_CS);

void setup()
{
  Serial.begin(115200);
  ts.begin();
  ts.setRotation(1);
  while (!Serial && (millis() <= 1000))
    ;
}

void loop()
{
  if (ts.touched())
  {
    TS_Point p = ts.getPoint();
    Serial.print("Pressure = ");
    Serial.print(p.z);
    Serial.print(", x = ");
```

```
    Serial.print(p.x);
    Serial.print(", y = ");
    Serial.print(p.y);
    delay(30);
    Serial.println();
  }
}
```

Default graphics example

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_ILI9341.h>

#define TFT_CS D0  //for D1 mini or TFT I2C Connector Shield (V1.1.0 or later)
#define TFT_DC D8  //for D1 mini or TFT I2C Connector Shield (V1.1.0 or later)
#define TFT_RST -1 //for D1 mini or TFT I2C Connector Shield (V1.1.0 or later)
#define TS_CS D3   //for D1 mini or TFT I2C Connector Shield (V1.1.0 or later)

// #define TFT_CS 14  //for D32 Pro
// #define TFT_DC 27  //for D32 Pro
// #define TFT_RST 33 //for D32 Pro
// #define TS_CS  12 //for D32 Pro

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_RST);

void setup()
{
  Serial.begin(115200);
  Serial.println("ILI9341 Test!");

  tft.begin();

  // read diagnostics (optional but can help debug problems)
```

```
  uint8_t x = tft.readcommand8(ILI9341_RDMODE);
  Serial.print("Display Power Mode: 0x");
  Serial.println(x, HEX);
  x = tft.readcommand8(ILI9341_RDMADCTL);
  Serial.print("MADCTL Mode: 0x");
  Serial.println(x, HEX);
  x = tft.readcommand8(ILI9341_RDPIXFMT);
  Serial.print("Pixel Format: 0x");
  Serial.println(x, HEX);
  x = tft.readcommand8(ILI9341_RDIMGFMT);
  Serial.print("Image Format: 0x");
  Serial.println(x, HEX);
  x = tft.readcommand8(ILI9341_RDSELFDIAG);
  Serial.print("Self Diagnostic: 0x");
Serial.println(x, HEX);

  Serial.println(F("Benchmark                Time (microseconds)"));
  delay(10);
  Serial.print(F("Screen fill              "));
  Serial.println(testFillScreen());
  delay(500);

  Serial.println(F("Done!"));
}

void loop(void)
{
  for (uint8_t rotation = 0; rotation < 4; rotation++)
  {
    tft.setRotation(rotation);
    testText();
    delay(1000);
  }
}

unsigned long testFillScreen()
{
```

```
  unsigned long start = micros();
  tft.fillScreen(ILI9341_BLACK);
  yield();
  tft.fillScreen(ILI9341_RED);
  yield();
  tft.fillScreen(ILI9341_GREEN);
  yield();
  tft.fillScreen(ILI9341_BLUE);
  yield();
  tft.fillScreen(ILI9341_BLACK);
  yield();
  return micros() - start;
}

unsigned long testText()
{
  tft.fillScreen(ILI9341_BLACK);
  unsigned long start = micros();
tft.setCursor(0, 0);
  tft.setTextColor(ILI9341_WHITE);
  tft.setTextSize(1);
  tft.println("Hello World!");
  tft.setTextColor(ILI9341_YELLOW);
  tft.setTextSize(2);
  tft.println(1234.56);
tft.setTextColor(ILI9341_RED);
  tft.setTextSize(3);
  tft.println(0xDEADBEEF, HEX);
  tft.println();
  tft.setTextColor(ILI9341_GREEN);
  tft.setTextSize(5);
  tft.println("Groop");
  tft.setTextSize(2);
  tft.println("I implore thee,");
  tft.setTextSize(1);
  tft.println("my foonting turlingdromes.");
  tft.println("And hooptiously drangle me");
```

```
  tft.println("with crinkly bindlewurdles,");
  tft.println("Or I will rend thee");
  tft.println("in the gobberwarts");
  tft.println("with my blurglecruncheon,");
  tft.println("see if I don't!");
  return micros() - start;
}

unsigned long testLines(uint16_t color)
{
  unsigned long start, t;
  int x1, y1, x2, y2,
      w = tft.width(),
      h = tft.height();

  tft.fillScreen(ILI9341_BLACK);
  yield();

  x1 = y1 = 0;
  y2 = h - 1;
  start = micros();
  for (x2 = 0; x2 < w; x2 += 6)
    tft.drawLine(x1, y1, x2, y2, color);
  x2 = w - 1;
  for (y2 = 0; y2 < h; y2 += 6)
    tft.drawLine(x1, y1, x2, y2, color);
  t = micros() - start; // fillScreen doesn't count against timing

  yield();
  tft.fillScreen(ILI9341_BLACK);
  yield();

  x1 = w - 1;
  y1 = 0;
  y2 = h - 1;
  start = micros();
  for (x2 = 0; x2 < w; x2 += 6)
```

```
  tft.drawLine(x1, y1, x2, y2, color);
x2 = 0;
for (y2 = 0; y2 < h; y2 += 6)
  tft.drawLine(x1, y1, x2, y2, color);
t += micros() - start;

yield();
tft.fillScreen(ILI9341_BLACK);
yield();

x1 = 0;
y1 = h - 1;
y2 = 0;
start = micros();
for (x2 = 0; x2 < w; x2 += 6)
  tft.drawLine(x1, y1, x2, y2, color);
x2 = w - 1;
for (y2 = 0; y2 < h; y2 += 6)
  tft.drawLine(x1, y1, x2, y2, color);
t += micros() - start;

yield();
tft.fillScreen(ILI9341_BLACK);
yield();

x1 = w - 1;
y1 = h - 1;
y2 = 0;
start = micros();
for (x2 = 0; x2 < w; x2 += 6)
  tft.drawLine(x1, y1, x2, y2, color);
x2 = 0;
for (y2 = 0; y2 < h; y2 += 6)
  tft.drawLine(x1, y1, x2, y2, color);

yield();
return micros() - start;
```

```
}

unsigned long testFastLines(uint16_t color1, uint16_t color2)
{
  unsigned long start;
  int x, y, w = tft.width(), h = tft.height();

  tft.fillScreen(ILI9341_BLACK);
  start = micros();
  for (y = 0; y < h; y += 5)
    tft.drawFastHLine(0, y, w, color1);
  for (x = 0; x < w; x += 5)
    tft.drawFastVLine(x, 0, h, color2);

  return micros() - start;
}

unsigned long testRects(uint16_t color)
{
  unsigned long start;
  int n, i, i2,
      cx = tft.width() / 2,
      cy = tft.height() / 2;

  tft.fillScreen(ILI9341_BLACK);
  n = min(tft.width(), tft.height());
  start = micros();
  for (i = 2; i < n; i += 6)
  {
    i2 = i / 2;
    tft.drawRect(cx - i2, cy - i2, i, i, color);
  }

  return micros() - start;
}

unsigned long testFilledRects(uint16_t color1, uint16_t color2)
```

```
{
  unsigned long start, t = 0;
  int n, i, i2,
      cx = tft.width() / 2 - 1,
      cy = tft.height() / 2 - 1;

  tft.fillScreen(ILI9341_BLACK);
  n = min(tft.width(), tft.height());
  for (i = n; i > 0; i -= 6)
  {
    i2 = i / 2;
    start = micros();
    tft.fillRect(cx - i2, cy - i2, i, i, color1);
    t += micros() - start;
    // Outlines are not included in timing results
    tft.drawRect(cx - i2, cy - i2, i, i, color2);
    yield();
  }

  return t;
}

unsigned long testFilledCircles(uint8_t radius, uint16_t color)
{
  unsigned long start;
  int x, y, w = tft.width(), h = tft.height(), r2 = radius * 2;

  tft.fillScreen(ILI9341_BLACK);
  start = micros();
  for (x = radius; x < w; x += r2)
  {
    for (y = radius; y < h; y += r2)
    {
      tft.fillCircle(x, y, radius, color);
    }
  }
```

```
  return micros() - start;
}

unsigned long testCircles(uint8_t radius, uint16_t color)
{
  unsigned long start;
  int x, y, r2 = radius * 2,
          w = tft.width() + radius,
          h = tft.height() + radius;

  // Screen is not cleared for this one -- this is
  // intentional and does not affect the reported time.
  start = micros();
  for (x = 0; x < w; x += r2)
  {
    for (y = 0; y < h; y += r2)
    {
      tft.drawCircle(x, y, radius, color);
    }
  }

  return micros() - start;
}

unsigned long testTriangles()
{
  unsigned long start;
  int n, i, cx = tft.width() / 2 - 1,
          cy = tft.height() / 2 - 1;

  tft.fillScreen(ILI9341_BLACK);
  n = min(cx, cy);
  start = micros();
  for (i = 0; i < n; i += 5)
  {
    tft.drawTriangle(
        cx, cy - i,     // peak
```

```
        cx - i, cy + i, // bottom left
        cx + i, cy + i, // bottom right
        tft.color565(i, i, i));
  }

  return micros() - start;
}

unsigned long testFilledTriangles()
{
  unsigned long start, t = 0;
  int i, cx = tft.width() / 2 - 1,
         cy = tft.height() / 2 - 1;

  tft.fillScreen(ILI9341_BLACK);
  start = micros();
  for (i = min(cx, cy); i > 10; i -= 5)
  {
    start = micros();
    tft.fillTriangle(cx, cy - i, cx - i, cy + i, cx + i, cy + i,
                     tft.color565(0, i * 10, i * 10));
    t += micros() - start;
    tft.drawTriangle(cx, cy - i, cx - i, cy + i, cx + i, cy + i,
                     tft.color565(i * 10, i * 10, 0));
    yield();
  }

  return t;
}

unsigned long testRoundRects()
{
  unsigned long start;
  int w, i, i2,
      cx = tft.width() / 2 - 1,
      cy = tft.height() / 2 - 1;
```

```
  tft.fillScreen(ILI9341_BLACK);
  w = min(tft.width(), tft.height());
  start = micros();
  for (i = 0; i < w; i += 6)
  {
    i2 = i / 2;
    tft.drawRoundRect(cx - i2, cy - i2, i, i, i / 8, tft.color565(i, 0, 0));
  }

  return micros() - start;
}

unsigned long testFilledRoundRects()
{
  unsigned long start;
  int i, i2,
      cx = tft.width() / 2 - 1,
      cy = tft.height() / 2 - 1;

  tft.fillScreen(ILI9341_BLACK);
  start = micros();
  for (i = min(tft.width(), tft.height()); i > 20; i -= 6)
  {
    i2 = i / 2;
    tft.fillRoundRect(cx - i2, cy - i2, i, i, i / 8, tft.color565(0, i, 0));
    yield();
  }

  return micros() - start;
}
```

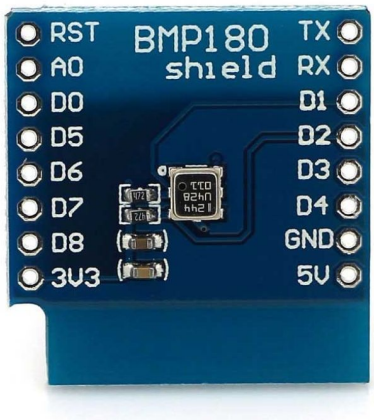
1P180 Shield

Description

The BMP180 is the new digital barometric pressure sensor of Bosch Sensortec, with a very high performance, which enables applications in advanced mobile devices, such as smartphones, tablet PCs and sports devices. It follows the BMP085 and brings many improvements, like the smaller size and the expansion of digital interfaces.

The ultra-low power consumption down to 3 µA makes the BMP180 the leader in power saving for your mobile devices. BMP180 is also distinguished by its very stable behavior (performance) with regard to the independency of the supply voltage.

The BMP180 shield uses D1 and D2 and is not an official wemos shield but is sturdy construction, good quality and comes with a selection of headers that you can solder yourself to it. Here is a picture of the shield



Code Example

We had a previous code example for this module, this was tested and worked with the shield perfectly. As you can see the Adafruit library is required

```
#include <Wire.h>
#include <Adafruit_BMP085.h>

Adafruit_BMP085 bmp;

void setup()
{
  Serial.begin(9600);
  //Wire.begin (4, 5);
  if (!bmp.begin())
  {
    Serial.println("Could not find BMP180 or BMP085 sensor at 0x77");
    while (1) {}
  }
}

void loop()
{
  Serial.print("Temperature = ");
  Serial.print(bmp.readTemperature());
  Serial.println(" Celsius");

  Serial.print("Pressure = ");
  Serial.print(bmp.readPressure());
  Serial.println(" Pascal");


  Serial.println();
```

```
  delay(5000);
}
```

Output

Open the serial monitor and you should see something like this

Temperature = 27.30 Celsius
Pressure = 99938 Pascal

Temperature = 27.30 Celsius
Pressure = 99934 Pascal

Temperature = 32.20 Celsius
Pressure = 99965 Pascal

18B20 Shield

Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

This shield is not an official Wemos shield but works fine.

Code Example

A lengthy code example which uses the OneWire library,

```
#include <OneWire.h>

// OneWire DS18S20, DS18B20, DS1822 Temperature Example
//
// http://www.pjrc.com/teensy/td_libs_OneWire.html
//
// The DallasTemperature library can do all this work for you!
// http://milesburton.com/Dallas_Temperature_Control_Library

OneWire  ds(D2);  // on pin D2 (a 4.7K resistor is necessary)
```

```
void setup(void) {
  Serial.begin(9600);
}

void loop(void)
{
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float celsius, fahrenheit;

  if ( !ds.search(addr))
  {
    Serial.println("No more addresses.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return;
  }

  Serial.print("ROM =");
  for( i = 0; i < 8; i++)
  {
    Serial.write(' ');
    Serial.print(addr[i], HEX);
  }

  if (OneWire::crc8(addr, 7) != addr[7])
  {
      Serial.println("CRC is not valid!");
      return;
  }
  Serial.println();
```

```
  // the first ROM byte indicates which chip
  switch (addr[0]) {
    case 0x10:
      Serial.println("  Chip = DS18S20");  // or old DS1820
      type_s = 1;
      break;
    case 0x28:
      Serial.println("  Chip = DS18B20");
      type_s = 0;
      break;
    case 0x22:
     Serial.println("  Chip = DS1822");
      type_s = 0;
      break;
    default:
      Serial.println("Device is not a DS18x20 family device.");
      return;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);        // start conversion, with parasite power on at the
end
  delay(1000);
  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);         // Read Scratchpad

  Serial.print("  Data = ");
  Serial.print(present, HEX);
  Serial.print(" ");
  for ( i = 0; i < 9; i++) {           // we need 9 bytes
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  Serial.print(" CRC=");
```

```
  Serial.print(OneWire::crc8(data, 8), HEX);
  Serial.println();

  // Convert the data to actual temperature
  // because the result is a 16 bit signed integer, it should
  // be stored to an "int16_t" type, which is always 16 bits
  // even when compiled on a 32 bit processor.
  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      // "count remain" gives full 12 bit resolution
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    // at lower res, the low bits are undefined, so let's zero them
    if (cfg == 0x00) raw = raw & ~7; // 9 bit resolution, 93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
    //// default is 12 bit resolution, 750 ms conversion time
  }
  celsius = (float)raw / 16.0;
  fahrenheit = celsius * 1.8 + 32.0;
  Serial.print("  Temperature = ");
  Serial.print(celsius);
  Serial.print(" Celsius, ");
  Serial.print(fahrenheit);
  Serial.println(" Fahrenheit");
}
```

Output

Open the serial monitor and you should see something like this

ROM = 28 FF B7 27 A1 16 3 59

```
  Chip = DS18B20
  Data = 1 A5 1 4B 46 7F FF C 10 99  CRC=99
  Temperature = 26.31 Celsius, 79.36 Fahrenheit
No more addresses.

ROM = 28 FF B7 27 A1 16 3 59
  Chip = DS18B20
  Data = 1 A5 1 4B 46 7F FF C 10 99  CRC=99
  Temperature = 26.31 Celsius, 79.36 Fahrenheit
No more addresses.
```

২ Shield

Description

The PIR shield is a simple PIR that uses an AS312, this shield uses D3

AS312 is a newest smart digital motion detector with a small window size. It offers a complete motion detector solution, with all electronic circuitry built into the detector housing. Only a power supply and power-switching components need to be added to make the entire motion switch. The AS312 delay time is 2 seconds.

Code Example

Very simple stuff here, we will output via the serial monitor.

```
const int PIR = D3;
    int PIRState = 0;

    void setup()
    {
      Serial.begin(9600);
      pinMode(PIR, INPUT);
    }
```

```
void loop()
{
  PIRState = digitalRead(PIR);

  if (PIRState == HIGH)
  {
    Serial.write("INTRUDER DETECTED\n");
  }
  else
  {
    Serial.write("NOWT GOING ON\n");
  }
  delay(1000);
}
```

Output

Open the serial monitor and move your hand in front of the PIR and then move away

NOWT GOING ON
NOWT GOING ON
NOWT GOING ON
INTRUDER DETECTED
INTRUDER DETECTED

cro SD Card Shield

Description

The shield provides a way of allowing a WeMos D1 mini to directly communicate with microSD cards. The shield is compatible with the standard Arduino SD card library.

Here is the shield

Configurable CS pin, Default: D4 (GPIO0)

Pins

D1 mini Shield
D5 CLK
D6 MISO
D7 MOSI
D4 (Default) CS

Code Example

```
#include <SPI.h>
#include <SD.h>

// change this to match your SD shield or module;
// WeMos Micro SD Shield V1.0.0: D8
// LOLIN Micro SD Shield V1.2.0: D4 (Default)
const int chipSelect = D4;
```

```
File myFile;

void setup()
{
 // Open serial communications and wait for port to open:
 Serial.begin(9600);
 while (!Serial) {
  ; // wait for serial port to connect. Needed for Leonardo only
 }

 Serial.print("Initializing SD card...");

 if (!SD.begin(chipSelect)) {
  Serial.println("initialization failed!");
  return;
 }
 Serial.println("initialization done.");

 // open the file. note that only one file can be open at a time,
 // so you have to close this one before opening another.
 myFile = SD.open("test.txt", FILE_WRITE);

 // if the file opened okay, write to it:
 if (myFile) {
  Serial.print("Writing to test.txt...");
  myFile.println("testing 1, 2, 3.");
  // close the file:
  myFile.close();
  Serial.println("done.");
 } else {
  // if the file didn't open, print an error:
  Serial.println("error opening test.txt");
 }

 // re-open the file for reading:
 myFile = SD.open("test.txt");
 if (myFile) {
```

```
    Serial.println("test.txt:");

    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    // close the file:
    myFile.close();
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
}

void loop()
{
  // nothing happens after setup
}
```

Output

Controller Shield

Description

In this small article we take a look at the IR controller shield, this is a shield which can act as an IR receiver and sender, here are some details about the shield

- 1x IR receiver (38kHz)
- Configurable IO (Default: Sender - D3/GPIO0, Receiver - D4/GPIO2)
- 4x IR LEDs emitter (940nm)

Lets see a picture of the shield, you can see the IR reciever and the 4 emitters around the outside, so quite a good design this.

The shield uses the following pins, so they would be unavailable for other devices that you use

D1 mini	Shield
D3	Send
D4	Receiver

Code Example

You need to install the https://github.com/markszabo/IRremoteESP8266 first.

There are a few test examples but i simply wanted to see the code returned from a keypress. The default test examples had a bit more data than I really wished. I have also noticed false codes getting returned at times, I have not figured this out yet. Looking at possible solutions although a few mention capacitors and noise.

```
#ifndef UNIT_TEST
#include <Arduino.h>
#endif
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>
#if DECODE_AC
#include <ir_Daikin.h>
#include <ir_Fujitsu.h>
#include <ir_Gree.h>
#include <ir_Haier.h>
#include <ir_Kelvinator.h>
#include <ir_Midea.h>
#include <ir_Toshiba.h>
#endif  // DECODE_AC

#define RECV_PIN D4
#define BAUD_RATE 115200
#define CAPTURE_BUFFER_SIZE 1024
```

```
#if DECODE_AC
#define TIMEOUT 50U
#else
#define TIMEOUT 15U  // Suits most messages, while not swallowing
many repeats.
#endif

#define MIN_UNKNOWN_SIZE 12


// Use turn on the save buffer feature for more complete capture
coverage.
IRrecv irrecv(RECV_PIN, CAPTURE_BUFFER_SIZE, TIMEOUT,
true);

decode_results results;  // Somewhere to store the results


void setup()
{
  Serial.begin(BAUD_RATE, SERIAL_8N1, SERIAL_TX_ONLY);
  while (!Serial)  // Wait for the serial connection to be establised.
    delay(50);
  Serial.println();
  Serial.print("IRrecvDumpV2 is now running and waiting for IR input
on Pin ");
  Serial.println(RECV_PIN);

#if DECODE_HASH
  // Ignore messages with less than minimum on or off pulses.
  irrecv.setUnknownThreshold(MIN_UNKNOWN_SIZE);
#endif  // DECODE_HASH
  irrecv.enableIRIn();  // Start the receiver
}


void loop()
```

```
{
  if (irrecv.decode(&results))
  {
    // print() & println() can't handle printing long longs. (uint64_t)
    serialPrintUint64(results.value, HEX);
    Serial.println("");
    irrecv.resume();  // Receive the next value
  }
  delay(200);
}
```

Output

Go ahead and press various keys on your remote, this is a couple of keys that I tested. They were 3, 4 and 5 and I did press them multiple times this was not bounce or multiple return codes

FF7A85
FF7A85
FF7A85
FF7A85
FF10EF
FF10EF
FF10EF
FF38C7

ıbient light Shield (BH1750)

Description

This is a simple light sensor which provides reasonably accurate, quantitative values for the light level. The BH1750 can be operated in continuous or single measurement mode, and has resolution settings of 0.5 lx, 1 lx, and 4 lx, and communicates over I2C. This offers a way to measure light levels in a more accurate and efficient way than a light-dependent resistor.

Features

1) I2C bus Interface (f / s Mode Support)
2) Spectral responsibility is approximately human eye response
3) Illuminance to Digital Converter
4) Wide range and High resolution. (1 - 65535 lx)
5) Low Current by power down function
6) 50Hz / 60Hz Light noise reject-function
7) 1.8V Logic input interface
8) No need any external parts
9) Light source dependency is little. (ex. Incandescent Lamp. Fluorescent Lamp. Halogen Lamp. White LED. Sun Light)
10) It is possible to select 2 type of I2C slave-address.

11) Adjustable measurement result for influence of optical window (It is possible to detect min. 0.11 lx, max. 100000 lx by using this function.)
12) Small measurement variation (+/- 20%)
13) The influence of infrared is very small.

Code Example

You need to install the following library for this example :

https://github.com/claws/BH1750

```
#include <Wire.h>
#include <BH1750.h>


BH1750 lightMeter(0x23);

void setup(){

  Serial.begin(9600);

  // Initialize the I2C bus
  Wire.begin();

  if (lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE))
  {
    Serial.println(F("BH1750 Advanced begin"));
  }
  else
  {
    Serial.println(F("Error initialising BH1750"));
  }

}

void loop()
{
  uint16_t lux = lightMeter.readLightLevel();
  Serial.print("Light: ");
  Serial.print(lux);
  Serial.println(" lx");
```

```
  delay(1000);
}
```

Output

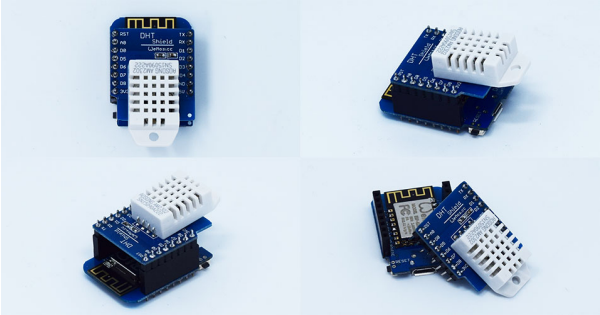
Open the serial monitor, cover the sensor, shine light on the sensor and so on to see the readings change, you can see where I covered it completely reading 0 lx

Light: 29 lx
Light: 29 lx
Light: 22 lx
Light: 0 lx
Light: 0 lx

IT Pro Shield

Description

The DHT Pro Shield is a digital temperature and humidity sensor shield based on the DHT22 sensor for your WeMos D1 mini. Measurements are read using one single pin. The measurement interval should be larger than 2 seconds.



Specification:

Temperature: -40..80°C (±0.5°C)
Humidity: 0..99.9%RH (±2%RH)

Code Example

There are basic examples , the code below is based on the example from https://github.com/wemos/D1_mini_Examples.git

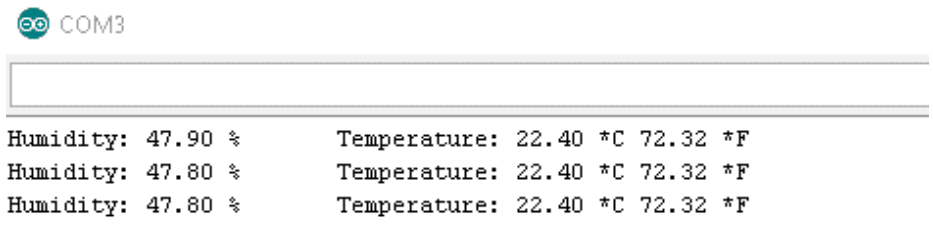
```
#include "DHT.h"
#define DHTPIN D4     // what pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302)

DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  Serial.begin(9600);
```

```
  Serial.println("DHTxx test!");
  dht.begin();
}

void loop()
{
  // Wait a few seconds between measurements.
  delay(2000);

  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f))
  {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(f);
  Serial.println(" *F");

}
```

Output

Open the serial monitor with Ctrl+Shift M or Tools -> Serial Monitor, and you should see the values for the humidity in percent

as well as the temperature and heat index in Celcius and Fahrenheit. Here is my hardware running



iB LED Shield

Description

The RGB shield contains 7 RGB LEDs these are WS2812B-mini each with 24-bit RGB color. There are 9 optional control pins, the default is D4.

Code Examples

You need to add the Adafruit Neopixel library for these examples, there are other libraries you can try of course -

Example 1

```
#include <Adafruit_NeoPixel.h>

#define PIN   D4
#define LED_NUM 7

// When we setup the NeoPixel library, we tell it how many pixels, and which pin to use to send signals.
Adafruit_NeoPixel leds = Adafruit_NeoPixel(LED_NUM, PIN, NEO_GRB + NEO_KHZ800);


void setup()
{
  leds.begin(); // This initializes the NeoPixel library.
}



void led_set(uint8 R, uint8 G, uint8 B)
{
  for (int i = 0; i < LED_NUM; i++)
  {
```

```
    leds.setPixelColor(i, leds.Color(R, G, B));
    leds.show();
    delay(150);
}
}

void loop() {

  led_set(10, 0, 0);//red
  led_set(0, 0, 0);

  led_set(0, 10, 0);//green
  led_set(0, 0, 0);

  led_set(0, 0, 10);//blue
  led_set(0, 0, 0);

}
```

Example 2

```
#include <Adafruit_NeoPixel.h>

#define PIN   D4
#define LED_NUM 7

// When we setup the NeoPixel library, we tell it how many pixels, and
which pin to use to send signals.
Adafruit_NeoPixel leds = Adafruit_NeoPixel(LED_NUM, PIN, NEO_GRB
+ NEO_KHZ800);


void setup()
{
 leds.begin(); // This initializes the NeoPixel library.
}



void loop()
{
 for (int i = 0; i < LED_NUM; i++)
 {
 leds.setPixelColor(i, leds.Color(10, 0, 0));
 leds.show();
 delay(250);

 leds.setPixelColor(i, leds.Color(0, 10, 0));
 leds.show();
 delay(250);

 leds.setPixelColor(i, leds.Color(0, 0, 10));
 leds.show();
 delay(250);
 }
}
```

Example 3

My favourite this one

```
#include <Adafruit_NeoPixel.h>

#define PIN   D4
#define LED_NUM 7

// When we setup the NeoPixel library, we tell it how many pixels, and
which pin to use to send signals.
Adafruit_NeoPixel leds = Adafruit_NeoPixel(LED_NUM, PIN, NEO_GRB
+ NEO_KHZ800);


void setup()
{
  leds.begin(); // This initializes the NeoPixel library.
  Serial.begin(9600);
  randomSeed(analogRead(A0));
}


void loop()
{
  //you can make the colours 255 but that is bright
  int redRandom = random(10);
  int greenRandom = random(10);
  int blueRandom = random(10);
  int ledRandom = random(7);
  leds.setPixelColor(ledRandom, leds.Color(redRandom, greenRandom,
blueRandom));
  leds.show();
  delay(250);
}
```

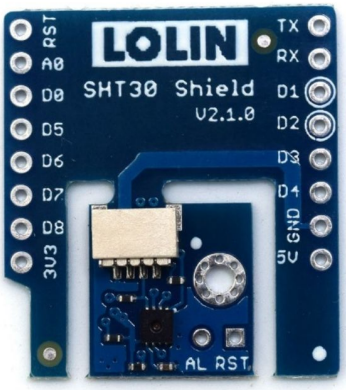
T30 Shield

Description

The digital SHT3x humidity sensor series takes sensor technology to a new level. As the successor of the SHT2x series it is determined to set the next industry standard in humidity sensing. The SHT3x humidity sensor series combines multiple functions and various interfaces (I2C, analog voltage output) with a applications-friendly, very wide operating voltage range (2.15 to 5.5 V). The SHT3x humidity sensor is available in both large and small volumes.

The SHT3x builds on a completely new and optimized CMOSens® chip, which allows for increased reliability and improved accuracy specifications. The SHT3x offers a range of new features, such as enhanced signal processing, two distinctive and user-selectable I2C addresses, an alert mode with programmable humidity and temperature limits, and communication speeds of up to 1 MHz.

The DFN package has a footprint of 2.5 × 2.5 mm2 with a height of 0.9 mm. This allows for integration of the SHT3x into a great variety of applications. Additionally, the wide supply voltage range of 2.15 to 5.5 V and variety of available interfaces guarantee compatibility with diverse integration requirements. All in all, the SHT3x humidity sensor series incorporates 15 years of knowledge from Sensirion, the leader in the humidity sensor industry.



Code Example

Adafruit again do the bulk of the work with the Adafruit_SHT31 library from https://github.com/adafruit/Adafruit_SHT31 . You can also install this using the library manager.

```
/***************************************************
  This is an example for the SHT31-D Humidity & Temp Sensor

  Designed specifically to work with the SHT31-D sensor from Adafruit
  ----> https://www.adafruit.com/products/2857

These sensors use I2C to communicate, 2 pins are required to
  interface
****************************************************/

#include <Arduino.h>
#include <Wire.h>
#include "Adafruit_SHT31.h"

Adafruit_SHT31 sht31 = Adafruit_SHT31();

void setup() {
  Serial.begin(9600);
```

```
  Serial.println("SHT30 test");
  if (! sht31.begin(0x45)) {   // Set to 0x44 for alternate i2c addr
    Serial.println("Couldn't find SHT30");
    while (1) delay(1);
  }
}


void loop() {
  float t = sht31.readTemperature();
  float h = sht31.readHumidity();

  if (! isnan(t)) {  // check if 'is not a number'
    Serial.print("Temp *C = "); Serial.println(t);
  } else {
    Serial.println("Failed to read temperature");
  }

  if (! isnan(h)) {  // check if 'is not a number'
    Serial.print("Hum. % = "); Serial.println(h);
  } else {
    Serial.println("Failed to read humidity");
  }
  Serial.println();
  delay(1000);
}
```

Output

Open the serial monitor and you should see something like this

Temp *C = 22.76
Hum. % = 43.25

Temp *C = 23.60

Hum. % = 44.41

Temp *C = 27.33
Hum. % = 54.18

T 1.4 Shield

Description

In this example we look at the 1.44“ inch TFT LCD shield from Wemos. It can display 128×128 resolution and 18-bit color The TFT is based on an ST7735S driver. Here is a picture of the LCD shield

Pins

The shield uses the following pins

D1 mini	Shield
RST*(D0/D3/D4))	TFT_RST
D3*(D0/D4/D8)	TFT_DC
D4*(D0/D3/D8)	TFT_CS
D7	MOSI
D5	SCK
NC*(D0/D3/D4/D8)	TFT_LED
*default	

Code Example

You need to install the https://github.com/adafruit/Adafruit-GFX-Library and https://github.com/adafruit/Adafruit-ST7735-Library libraries. You can do this in the Arduino libraries

Here are several examples

Example 1

```
#include <Adafruit_GFX.h> // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>

#define TFT_RST -1
#define TFT_CS D4
#define TFT_DC D3

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup(void)
{
tft.initR(INITR_144GREENTAB);
tft.setTextWrap(false); // Allow text to run off right edge
tft.fillScreen(ST7735_BLACK);
}

void loop(void)
{
tft.fillScreen(ST7735_BLACK);
tft.setCursor(0, 0);
tft.setTextColor(ST7735_BLUE);
tft.setTextSize(2);
tft.println("ESP8266");
tft.setRotation(tft.getRotation() + 1);
delay(3000);
}
```

Example 2

```
//triangles
#include <Adafruit_GFX.h> // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>

#define TFT_RST -1
#define TFT_CS D4
#define TFT_DC D3

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup(void)
{
tft.initR(INITR_144GREENTAB);
}

void loop(void)
{
tft.fillScreen(ST77XX_BLACK);
int color = 0xF800;
int t;
int w = tft.width()/2;
int x = tft.height()-1;
int y = 0;
int z = tft.width();
for(t = 0 ; t <= 15; t++)
{
tft.drawTriangle(w, y, y, x, z, x, color);
x-=4;
y+=4;
z-=4;
color+=100;
}
delay(500);
}
```

Example 3

```
//round rectangles
#include <Adafruit_GFX.h> // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>

#define TFT_RST -1
#define TFT_CS D4
#define TFT_DC D3

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup(void)
{
tft.initR(INITR_144GREENTAB);
}

void loop(void)
{
tft.fillScreen(ST77XX_BLACK);
int color = 100;
int i;
int t;
for(t = 0 ; t <= 4; t+=1)
{
int x = 0;
int y = 0;
int w = tft.width()-2;
int h = tft.height()-2;
for(i = 0 ; i <= 16; i+=1)
{
tft.drawRoundRect(x, y, w, h, 5, color);
x+=2;
y+=3;
w-=4;
h-=6;
color+=1100;
}
```

```
color+=100;
}
delay(500);
}
```

Example 4

```
//text examples
#include <Adafruit_GFX.h> // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>

#define TFT_RST -1
#define TFT_CS D4
#define TFT_DC D3

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup(void)
{
tft.initR(INITR_144GREENTAB);
}

void loop(void)
{
float p = 3.1415926;
tft.setTextWrap(false);
tft.fillScreen(ST77XX_BLACK);
tft.setCursor(0, 30);
tft.setTextColor(ST77XX_RED);
tft.setTextSize(1);
tft.println("Hello World!");
tft.setTextColor(ST77XX_YELLOW);
tft.setTextSize(2);
tft.println("Hello World!");
tft.setTextColor(ST77XX_GREEN);
tft.setTextSize(3);
```

```
tft.println("Hello World!");
tft.setTextColor(ST77XX_BLUE);
tft.setTextSize(4);
tft.print(1234.567);
delay(2500);
tft.setCursor(0, 0);
tft.fillScreen(ST77XX_BLACK);
tft.setTextColor(ST77XX_WHITE);
tft.setTextSize(0);
tft.println("Hello World!");
tft.setTextSize(1);
tft.setTextColor(ST77XX_GREEN);
tft.print(p, 6);
tft.println(" Want pi?");
tft.println(" ");
tft.print(8675309, HEX); // print 8,675,309 out in HEX!
tft.println(" Print HEX!");
tft.println(" ");
tft.setTextColor(ST77XX_WHITE);
tft.println("Sketch has been");
tft.println("running for: ");
tft.setTextColor(ST77XX_MAGENTA);
tft.print(millis() / 1000);
tft.setTextColor(ST77XX_WHITE);
tft.print(" seconds.");
delay(2500);
}
```

Example 5

```
//rectangle examples
#include <Adafruit_GFX.h> // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>

#define TFT_RST -1
#define TFT_CS D4
```

```
#define TFT_DC D3

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup(void)
{
tft.initR(INITR_144GREENTAB);
}

void loop(void)
{
tft.fillScreen(ST77XX_BLACK);
//rectangles
for (int16_t x=0; x < tft.width(); x+=6)
{
tft.drawRect(tft.width()/2 -x/2, tft.height()/2 -x/2 , x, x, ST77XX_WHITE);
}
delay(2500);
//filled rectangles
tft.fillScreen(ST77XX_BLACK);
for (int16_t x=tft.width()-1; x > 6; x-=6)
{
tft.fillRect(tft.width()/2 -x/2, tft.height()/2 -x/2 , x, x, ST77XX_YELLOW);
tft.drawRect(tft.width()/2 -x/2, tft.height()/2 -x/2 , x, x,
ST77XX_MAGENTA);
}
delay(2500);
}
```

Example 6

```
//circle examples
#include <Adafruit_GFX.h> // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>
```

```
#define TFT_RST -1
#define TFT_CS D4
#define TFT_DC D3

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup(void)
{
tft.initR(INITR_144GREENTAB);
}

void loop(void)
{
int radius = 10;
tft.fillScreen(ST77XX_BLACK);
//filled circle
for (int16_t x=radius; x < tft.width(); x+=radius*2)
{
for (int16_t y=radius; y < tft.height(); y+=radius*2)
{
tft.fillCircle(x, y, radius, ST77XX_YELLOW);
}
}
delay(2500);
//circles
tft.fillScreen(ST77XX_BLACK);
for (int16_t x=0; x < tft.width()+radius; x+=radius*2)
{
for (int16_t y=0; y < tft.height()+radius; y+=radius*2)
{
tft.drawCircle(x, y, radius, ST77XX_MAGENTA);
}
}
delay(2500);
}
```

Example 7

```
//play button
#include <Adafruit_GFX.h> // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>

#define TFT_RST -1
#define TFT_CS D4
#define TFT_DC D3

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup(void)
{
tft.initR(INITR_144GREENTAB);
}

void loop(void)
{
tft.fillScreen(ST77XX_BLACK);
tft.fillRoundRect(25, 10, 78, 60, 8, ST77XX_WHITE);
tft.fillTriangle(42, 20, 42, 60, 90, 40, ST77XX_RED);
delay(500);
// play color
tft.fillTriangle(42, 20, 42, 60, 90, 40, ST77XX_BLUE);
delay(500);
// play color
tft.fillTriangle(42, 20, 42, 60, 90, 40, ST77XX_GREEN);
}
```

Example 8

```
#include <Adafruit_GFX.h> // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>

#define TFT_RST -1
#define TFT_CS D4
```

```
#define TFT_DC D3

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup(void)
{
tft.initR(INITR_144GREENTAB);
}

void loop(void)
{
uint16_t color = ST77XX_YELLOW;
tft.fillScreen(ST77XX_BLACK);
for (int16_t x=0; x < tft.width(); x+=6)
{
tft.drawLine(0, 0, x, tft.height()-1, color);
delay(0);
}
for (int16_t y=0; y < tft.height(); y+=6)
{
tft.drawLine(0, 0, tft.width()-1, y, color);
delay(0);
}
}
```

Example 9

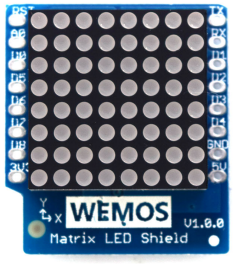
```
#include <Adafruit_GFX.h> // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>

#define TFT_RST -1
#define TFT_CS D4
#define TFT_DC D3

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);
```

```
void setup(void)
{
tft.initR(INITR_144GREENTAB);
}

void loop(void)
{
for (uint8_t i=0; i<4; i++)
{
tft.fillScreen(ST77XX_BLACK);
tft.drawCircle(10, 30, 10, ST77XX_YELLOW);
tft.setRotation(tft.getRotation()+1);
}
delay(500);
for (uint8_t i=0; i<4; i++)
{
tft.fillScreen(ST77XX_BLACK);
tft.fillRect(10, 20, 10, 20, ST77XX_GREEN);
tft.setRotation(tft.getRotation()+1);
}
delay(500);
}
```

trix LED Shield

Description

The matrix led shield is an 8×8 dot matrix LED which has 8 step adjustable intensity and is controlled by a TM140. The shield uses D5 and D7.



Code Examples

his requires the library from
https://github.com/wemos/WEMOS_Matrix_LED_Shield_Arduino_Library

here are a couple of built in examples but here are a few examples that will flash various LEDs

Example 1

```
include <WEMOS_Matrix_LED.h>

MLED mled(5); //set intensity=5

void setup()
```

```
{

}

void loop() {

  for(int y=0;y<8;y++)
  {
    for(int x=0;x<8;x++)
    {
       mled.dot(x,y); // draw dot
       mled.display();
       delay(200);
    }
}
}
```

Example 2

```
#include <WEMOS_Matrix_LED.h>

MLED mled(5); //set intensity=5

void setup()
{

}

void loop() {

for(int y=0;y<8;y++)
{
mled.dot(0,y); // draw dot
mled.display();
delay(200);
}
}
```

Example 3

```
#include <WEMOS_Matrix_LED.h>

    MLED mled(5); //set intensity=5
    int randX, randY;

    void setup()
    {
    Serial.begin(9600);
    randomSeed(analogRead(0));

    }

    void loop()
    {
    randX = random(8);
    randY = random(8);
    mled.dot(randX,randY); // draw dot
    mled.display();
    delay(200);
    }
```

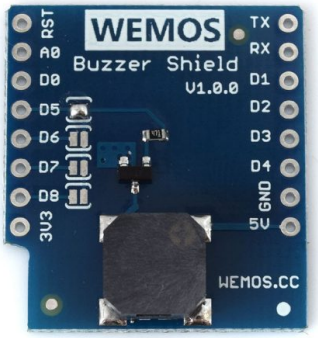
Example 4

```
#include <WEMOS_Matrix_LED.h>

    MLED mled(5); //set intensity=5
    int randX, randY;

    void setup()
    {
    Serial.begin(9600);
    randomSeed(analogRead(0));

    }

    void loop()
    {
    randX = random(8);
    randY = random(8);
    mled.dot(randX,randY); // draw dot
    mled.display();
    delay(200);
    mled.dot(randX,randY,0);//clear dot
    mled.display();
    delay(200);
    }
```

zzer Shield

Description

The buzzer shield uses an MLT-8540 buzzer



Frequency: 1kHz-3kHz
4 optional control ports - the default is D5 but D6,D7 and D8 can also be used

Code Example

This example plays a few notes from a popular song

```
ıt buzzerPin=D5; //Buzzer control port, default D5
    const int songLength = 18;
    char notes[] = "cdfda ag cdfdg gf ";
    int beats[] = {1,1,1,1,1,1,4,4,2,1,1,1,1,1,1,4,4,2};
    int tempo = 150;


    void setup()
    {
      pinMode(buzzerPin, OUTPUT);
    }


    void loop()
    {
```

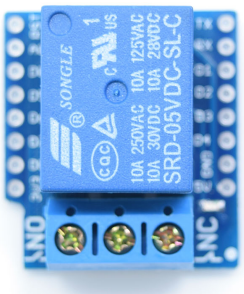
```
  int i, duration;

  for (i = 0; i < songLength; i++) // step through the song arrays
  {
    duration = beats[i] * tempo;  // length of note/rest in ms

    if (notes[i] == ' ')          // is this a rest?
    {
      delay(duration);            // then pause for a moment
    }
    else                          // otherwise, play the note
    {
      tone(buzzerPin, frequency(notes[i]), duration);
      delay(duration); // wait for tone to finish
    }
    delay(tempo/10);              // brief pause between notes
  }

  while(true){}

}


int frequency(char note)
{

  int i;
  const int numNotes = 8;  // number of notes we're storing
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
  int frequencies[] = {262, 294, 330, 349, 392, 440, 494, 523};


  for (i = 0; i < numNotes; i++)  // Step through the notes
  {
    if (names[i] == note)         // Is this the one?
    {
      return(frequencies[i]);     // Yes! Return the frequency
```

```
    }
  }
  return(0);
}
```

Relay Shield

Description

This relay shield provides the ability to switch high voltage, or high power devices from a single digital pin.



NO: 5A(250VAC/30VDC), 10A(125VAC), MAX:1250VA/150W
NC: 3A(250VAC/30VDC), MAX:750VA/90W
7 configurable IO (Default: D1)

Code Example

```
#define PIN D1 // Relay shield is controlled by digital pin D1

void setup()

{
  pinMode(PIN, OUTPUT); // Set the pin to an output
}

void loop()
{
  digitalWrite(PIN, HIGH); // switch on the relay
  delay(1000); // Wait for another second
  digitalWrite(PIN, LOW); // switch off the relay
```

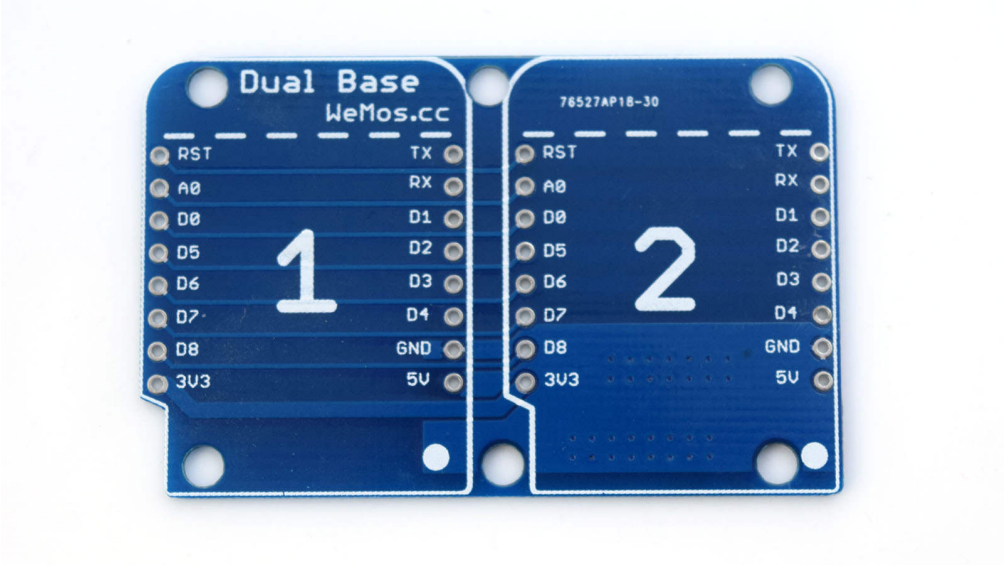
```
  delay(1000); // Wait for a second
}
```

All you will hear unless you connect an external source is the relay click on and off

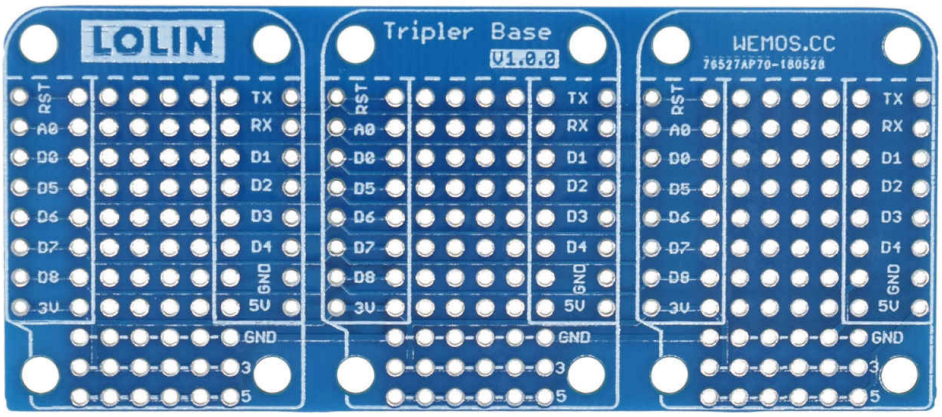
ıer Shields

These are some other shields which may be useful, we use the dual base and tripler shield when we want to connect multiple shields to a wemos without stacking the shields on top of each other

This is the dual base

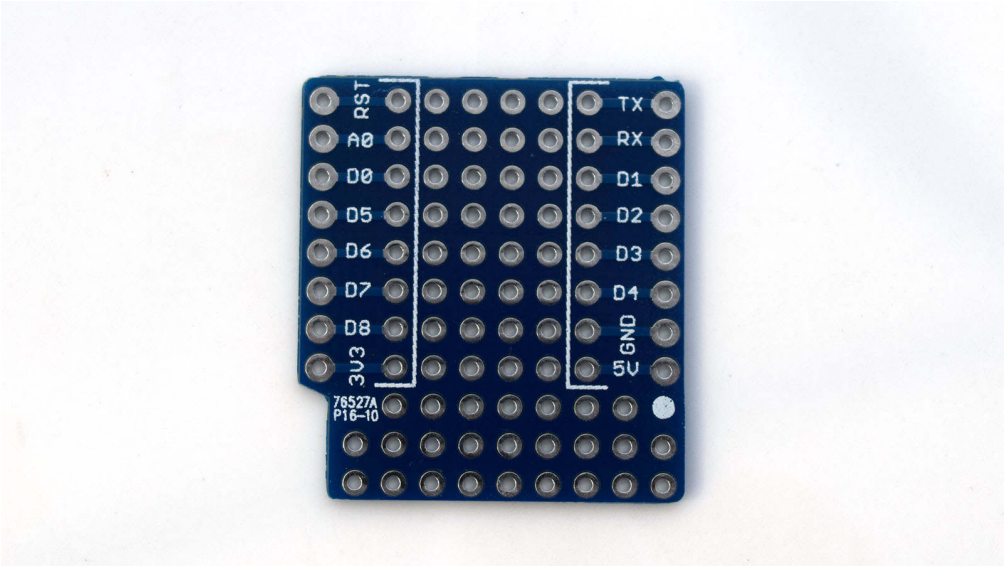


This is the tripler

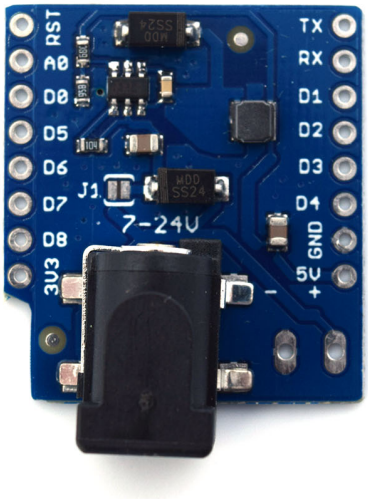


Another 2 shields are the protoboard and the DC shield. The protoboard allows you to create small prototype circuits in the wemos dimensions and the dc power shield allows you to power your wemos mini and shields using an external power source such as a 12v wall PSU.

This is the protoboard



This is the DC power shield



nbinations of shields

IT22 readings on an OLED display simple project

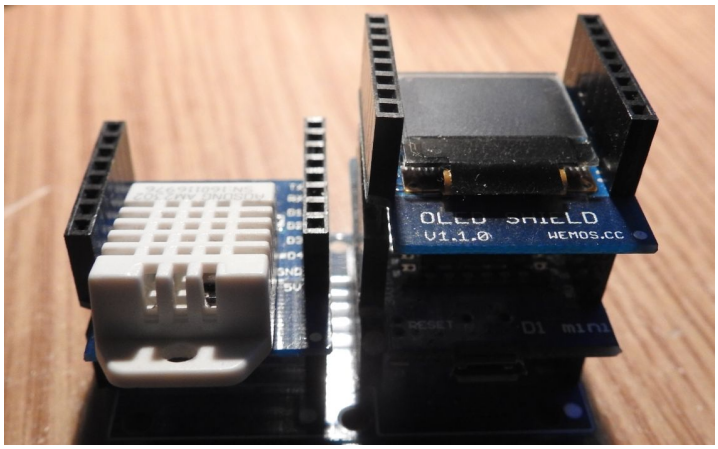
Description

In this particular example we are going to simply display temperature. pressure and altitude readings from a DHt22 sensor on an OLED display. These are all the shields used in making this a simple project to build.

Requirements

1 x Wemos Mini
1 x Wemos Dual Base
1 x OLED Shield
1 x DHT Pro Shield

You can see what I assembled in the picture below, I could have stacked in one column but I wanted to see the OLED and also leave the DHt22 uncovered.



Code Example

Various libraries required - you can install these via the library manager, here are links to them

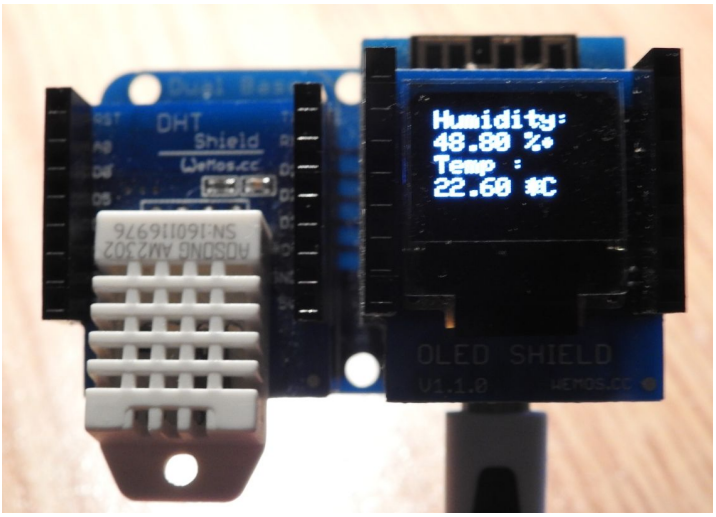
https://github.com/adafruit/DHT-sensor-library
https://github.com/adafruit/Adafruit_Sensor
https://github.com/sparkfun/SparkFun_Micro_OLED_Arduino_Library

```
#include "DHT.h"
#include <Wire.h>  // Include Wire if you're using I2C
#include <SFE_MicroOLED.h>  // Include the SFE_MicroOLED library

#define DHTPIN D4     // what pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302)
#define PIN_RESET 255  //
#define DC_JUMPER 0  // I2C Addres: 0 - 0x3C, 1 - 0x3D

MicroOLED oled(PIN_RESET, DC_JUMPER); // Example I2C declaration
DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  Serial.begin(9600);
  dht.begin();
  oled.begin();
  oled.clear(ALL);  // Clear the display's memory (gets rid of artifacts)
  oled.display();
}

void loop()
{
  // Wait a few seconds between measurements.
  delay(2000);
```

```
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t))
  {
   Serial.println("Failed to read from DHT sensor!");
   return;
  }
  oled.clear(PAGE);
  oled.setFontType(0); // set font type 0, please see declaration in
SFE_MicroOLED.cpp
  oled.setCursor(1, 3);
  oled.print("Humidity: ");
  oled.setCursor(1, 12);
  oled.print(h);
  oled.print(" %\t");
  oled.setCursor(1, 21);
  oled.print("Temp :");
  oled.setCursor(1, 30);
  oled.print(t);
  oled.print(" *C ");
  oled.display();
}
```

Output

Here you can see the output



ED shield bitcoin ticker example

Description

In this example we will display the price in USD of 1 bitcoin, the data is taken from coindesk's API and displayed on an oled display.

Requirements

1 x Wemos Mini
1 x Wemos Dual Base
1 x OLED Shield
1 x DHT Pro Shield

Code Example

OLED library required - you can install this via the library manager, here are links to them

https://github.com/sparkfun/SparkFun_Micro_OLED_Arduino_Library

```
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <SFE_MicroOLED.h>
#include <ArduinoJson.h>

const char* ssid = "iainhendry";
const char* pass = "iain061271";
#define PIN_RESET 255 //
#define DC_JUMPER 0 // I2C Addres: 0 - 0x3C, 1 - 0x3D
String id;
String value;
String json;

MicroOLED oled(PIN_RESET, DC_JUMPER); // I2C Example
```

```
WiFiClient client;

// delay between updates
const unsigned long postingInterval = 60L * 1000L;
unsigned long lastConnectionTime = 0;

void setup()
{
delay(100);
Serial.begin(115200);
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, pass);
while (WiFi.status() != WL_CONNECTED)
{
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

Serial.print("Connecting to ");

oled.begin();
oled.clear(ALL);
oled.setCursor(0,0);
oled.display();
oled.clear(PAGE);
oled.clear(ALL);
oled.print("Bitcoin price");
oled.setCursor(0,1);
oled.print("loading..");
```

```
oled.display(); // Display what's in the buffer (splashscreen)
delay(50);

}

int check_connect = 0;

void httpRequest()
{
client.stop();

// if there's a successful connection:
if (client.connect("api.coindesk.com", 80))
{
Serial.println("connecting...");
client.println("GET /v1/bpi/currentprice.json HTTP/1.1");
client.println("Host: api.coindesk.com");
client.println("User-Agent: ESP8266/1.1");
client.println("Connection: close");
client.println();
lastConnectionTime = millis();
}
else
{
// if you couldn't make a connection:
Serial.println("connection failed");
}
}

void loop()
{
int cnt;

if (cnt++ == 10000)
{
cnt = 0;
if (check_connect++ == 50)
```

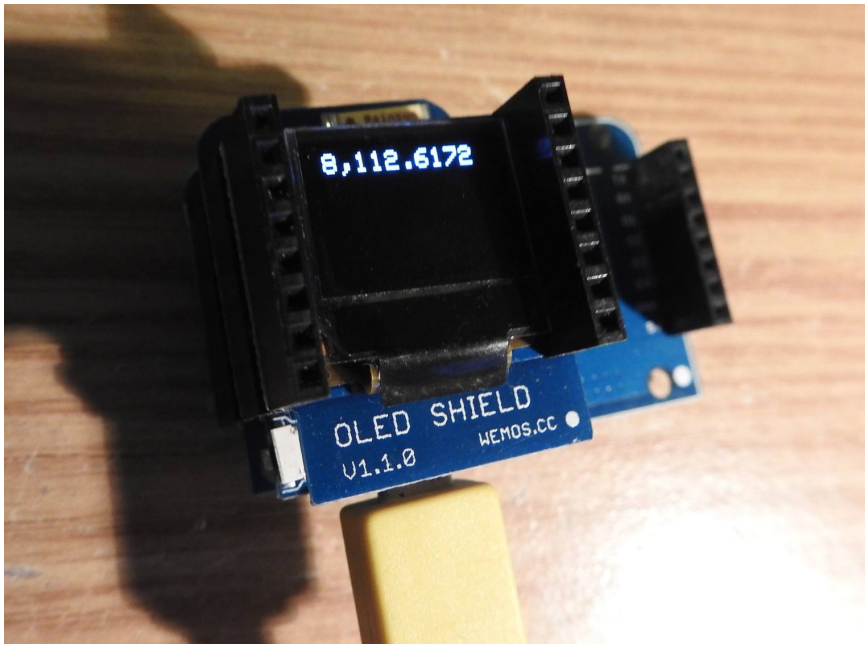
```
{
check_connect = 0;
if (WiFi.status() != WL_CONNECTED)
{
}
}
}

if (millis() - lastConnectionTime > postingInterval)
{
httpRequest();
unsigned int i = 0; //timeout counter
int n = 1; // char counter
char json[500] ="{";

while (!client.find("\"USD\":{")){}

while (i<20000)
{
if (client.available())
{
char c = client.read();
json[n]=c;
if (c=='}') break;
n++;
i=0;
}
i++;
}

StaticJsonBuffer<500> jsonBuffer;
JsonObject& root = jsonBuffer.parseObject(json);

String newjson = root["code"];
String value = root["rate"];
id = newjson.substring(9,12);
```

```
// value = newjson.substring(41,51);
oled.display();
oled.clear(PAGE); // Clear the display's internal memory
oled.clear(ALL); // Clear the library's display buffer
oled.setCursor(0,1);
oled.print(value);
oled.display();

id="";
value="";
}
}
```

Output

As you can see I originally wrote this example a while ago, Bitcoins price is no longer that high

You should see something like this on the oled display



۲ shield warning example

Description

In this example we will use a PIR shield and a buzzer shield, when an intruder is detected then the buzzer will sound. A basic alarm system

Requirements

1 x Wemos Mini
1 x Wemos Dual Base
1 x PIR Shield
1 x Buzzer shield

Code Example

```
const int PIR = D3;
int PIRState = 0;
int buzzer=D5; //Buzzer control port, default D5

void setup()
{
  pinMode(PIR, INPUT);
  pinMode(buzzer, OUTPUT);
  digitalWrite(buzzer, LOW);
  // set initial state, LED off
  digitalWrite(BUILTIN_LED, HIGH);
  Serial.begin(9600);
}

void loop() {

  PIRState = digitalRead(PIR);
```

```
  if (PIRState == HIGH)
  {
    analogWrite(buzzer, 512); // buzzer on
    Serial.write("INTRUDER DETECTED\n");
  }
  else
  {
   digitalWrite(buzzer, LOW); // buzzer off
    Serial.write("NOWT GOING ON\n");
  }
  delay(1000);
}
```

1P180 versus SHt30 temperature comparison

Description

In this project idea we compare the temperature readings from the BMP180 and an SHT30

There are a few temperature sensors we could have picked but these 2 worked fine

Requirements

1 x Wemos Mini
1 x Wemos triple Base
1 x BMP180 shield
1 x SHT30 shield

Code Example

```
#include <Wire.h>
#include <Adafruit_BMP085.h>
#include "Adafruit_SHT31.h"
#include <Arduino.h>

Adafruit_BMP085 bmp;
Adafruit_SHT31 sht31 = Adafruit_SHT31();

void setup()
{
 Serial.begin(9600);
 //start bmp180
 Serial.println("SHT30 and BMP180 comparison test");
 if (!bmp.begin())
 {
  Serial.println("Could not find BMP180 or BMP085 sensor at 0x77");
  while (1) {}
 }
 //start sht30
 if (! sht31.begin(0x45)) {   // Set to 0x44 for alternate i2c addr
  Serial.println("Couldn't find SHT30");
```

```
    while (1) delay(1);
  }
}

void loop()
{
  float t = sht31.readTemperature();

  Serial.print("BMP180 Temperature = ");
  Serial.print(bmp.readTemperature());
  Serial.println(" Celsius");

  Serial.print("SHT30 Temperature = ");
  Serial.print(t);
  Serial.println(" Celsius");

  Serial.println();
  delay(1000);
}
```

ve sensor values to an SD card

Description

This is another simple project using a Wemos MIni and a couple of shields. This time we will connect a DHT11 shield and an SD card shield and we will log some temperature data from the dht11 to an sd card. The data will be saved as a csv file, this means you can quite easily open in Microsoft Excel.

The 2 shields use the following I/O pins, the DHT uses D4 and the SD card uses the SPI pins D5, D6, D7 and D8

D1 mini	**Shield**
D4	DHT Pin
D5	SD CLK
D6	SD MISO
D7	SD MOSI
D8	SD CS

Requirements

Wemos Mini
Micro SD Shield
DHT11 Shield
1 SD card (I used an 8Gb one)

Code Example

```
#include "DHT.h"
#include <SPI.h>
#include <SD.h>

#define DHTPIN D4
#define DHTTYPE DHT11
```

```
DHT dht(DHTPIN, DHTTYPE);

const int chipSelect = D8;
File myFile;

void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }

  Serial.print("Initializing SD card...");

  if (!SD.begin(chipSelect))
  {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");
  dht.begin();

}

void loop()
{
  delay(2000);
  myFile = SD.open("dht11.csv", FILE_WRITE);
      // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow
sensor)
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f))
```

```
  {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // if the file opened okay, write to it:
  if (myFile)
  {
    Serial.print("opened dht11.csv...");
    myFile.print(h);
    myFile.print(",");
    myFile.print(t);
    myFile.print(",");
    myFile.println(f);
    // close the file:
    myFile.close();
    Serial.println("closed dht11.csv.");
  }
  else
  {
    // if the file didn't open, print an error:
    Serial.println("error opening dht11.csv");
  }
}
```

Output

Open the csv file and you should see something like the following

34.00,25.00,77.00
34.00,25.00,77.00
34.00,25.00,77.00
34.00,25.00,77.00
33.00,25.00,77.00
33.00,25.00,77.00

nclusion

We have just scratched the surface of what you can do with the wemos mini, this ebook focuses on shields, the next one will look at various sensors which can be connected to a wemos.

We recommend looking at the examples that are installed when you add support to the Arduino IDE, there are many wifi examples and other network examples that are worth looking at

You can view code examples from this book at

https://github.com/getelectronics/wemos-basics-ebook