

ESP32 Lite

Contents

[1. Overview 3](#__RefHeading___Toc7725_733998677)

[2. Setup Arduino IDE for Arduino & environment for Espressif devices. 4](#__RefHeading___Toc19948_1323753780)

[3. Blinking a led with ESP32 8](#__RefHeading___Toc4780_733998677)

[4. Analog read serial 10](#__RefHeading___Toc7305_733998677)

[5. Blink led without delay 11](#__RefHeading___Toc7307_733998677)

[6. Digital read serial 12](#__RefHeading___Toc7309_733998677)

[7. Use button to control the LED 14](#__RefHeading___Toc7311_733998677)

[8. State change detection(edge detection) for pushbutton 15](#__RefHeading___Toc7313_733998677)

[9. LED PWM Fading 17](#__RefHeading___Toc7315_733998677)

[10. Use DHT-11 Sensor to check humidity and temperature 19](#__RefHeading___Toc7317_733998677)

[11. Use a LDR Sensor 20](#__RefHeading___Toc7319_733998677)

[12. Use Soil Moisture Sensor 21](#__RefHeading___Toc7321_733998677)

[13. Use Sound Sensor 22](#__RefHeading___Toc7323_733998677)

[14. Use Buzzer 23](#__RefHeading___Toc7325_733998677)

[15. Matrix Led Neopixel 24](#__RefHeading___Toc7327_733998677)

[16. More example about Matrix Led Neopixel 27](#__RefHeading___Toc7731_733998677)

[17. Using OLED 0,96" 33](#__RefHeading___Toc7331_733998677)

# Overview

This is a basic tutorial of ESP32 Lite. It is a General Introduction of ESP32 Lite.

In this document, we will discuss all (well, most) of the popular features of ESP32 Lite as well as how to use other modules and connect them with ESP32 Lite,

GPIO

Blink LED

Analog and Digital

Button

PWM

Sensor

Buzzer

LED Matrix

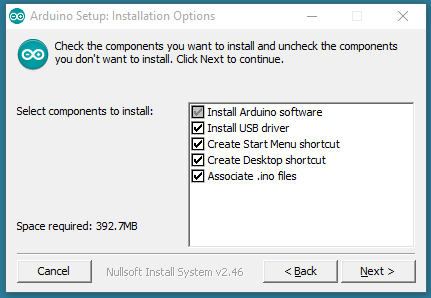
OLED 0.96"

# **Setup Arduino IDE for Arduino & environment for Espressif devices.**

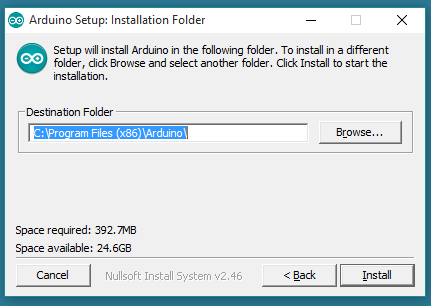
**Install Arduino IDE:**

Get the latest version from the [**download page**](https://www.arduino.cc/en/Main/Software). You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually.

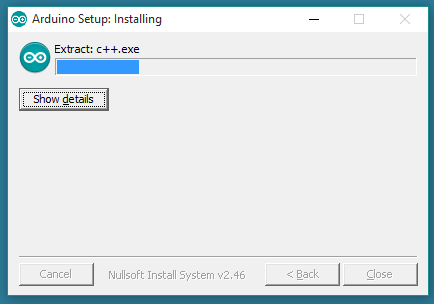
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



*Choose the components to install*



*Choose the installation directory (we suggest to keep the default one)*



*The process will extract and install all the required files to execute properly the Arduino Software (IDE)*

***Attension: If have any pop-up showing in the setup progress, please Click YES/OK/ACCEPT to install driver for arduino device.***

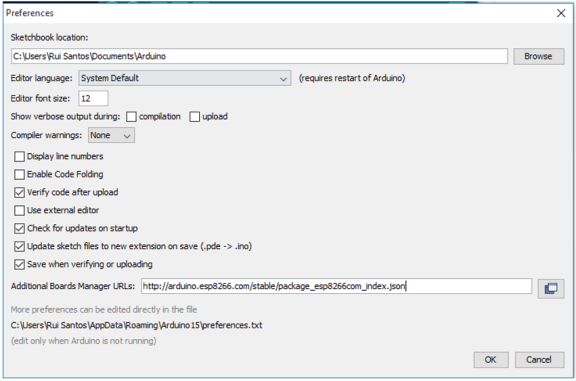
***Setup Environment for Espressif device***

1) Open the preferences window from the Arduino IDE. Go to File > Preferences

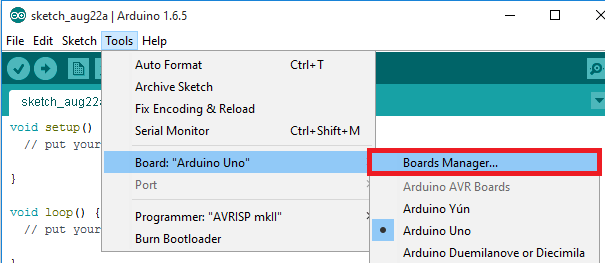
2) Enter

<http://arduino.esp8266.com/stable/package_esp8266com_index.json>

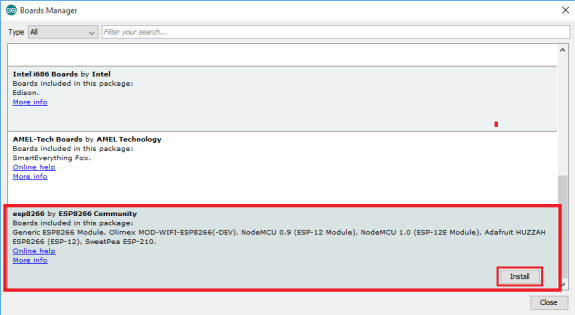
into Additional Board Manager URLs field and click the “OK” button



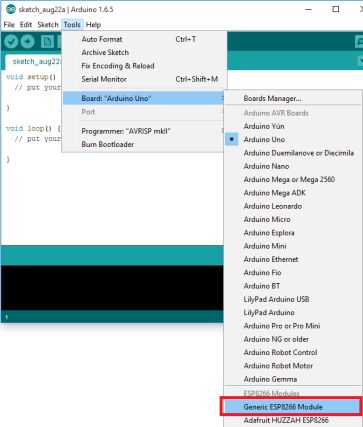
3) Open boards manager. Go to Tools > Board > Boards Manager…



4) Scroll down, select the ESP8266 board menu and install “esp8266 platform”



5) Choose your ESP8266 board from Tools > Board > Generic ESP8266 Module



6) Finally, re-open your Arduino IDE

# Blinking a led with ESP32

*Introduction*

This example shows the simplest thing you can do with an Arduino or Genuino to see physical output: it blinks the on-board LED.

The 'Blinking an LED' project uses the ESP32 Development Board will be used to blink an LED at a specific timed interval, infinitely. It is the essential fundamental tutorial for any microcontroller board, as it is the hardware equivalent of the classic "Hello World" tutorial

*Hardware Required*

Lolin32 Lite board

Lolin32 Lite shield

Module blocky LED (optional)

*Circuit*

This example uses the built-in LED that most Arduino and Genuino boards have. This LED is connected to a digital pin and its number may vary from board type to board type

*Code*

After you build the circuit plug your LoLin32 Lite board into your computer, start the Arduino Software (IDE) and enter the code below. You may also load it from the menu File/Examples/01.Basics/Blink . Remeber check sure your board connect is LoLin32 Lite in Arduino IDE “Tool → board “WEMOS LOLIN32””. The first thing you do is to initialize pin 22 as an output pin with the line

*pinMode(22, OUTPUT);*

In the main loop, you turn the LED on with the line:

*digitalWrite(22, HIGH);*

This supplies 3.3 volts to the LED anode. That creates a voltage difference across the pins of the LED, and lights it up. Then you turn it off with the line:

*digitalWrite(22, LOW);*

That takes the pin 22 back to 0 volts, and turns the LED off. In between the on and the off, you want enough time for a person to see the change, so the delay() commands tell the board to do nothing for 1000 milliseconds, or one second. When you use the delay() command, nothing else happens for that amount of time.

You can get code at link

------------------------------------------------------------------------------------------------------------------------------

*void* **setup**() {  
 *// initialize digital pin 22 as an output.*  
pinMode(22, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void **loop**() {  
 digitalWrite(22, HIGH); *// turn the LED on (HIGH is the voltage level)*  
delay(1000); *// wait for a second*  
digitalWrite(22, LOW); *// turn the LED off by making the voltage LOW*  
delay(1000); *// wait for a second*  
*}*

*//This code will blink your Led 22 on LoLin32 Board*

*-----------------------------------------------------------------------------------------------------------------------------*

You can also use board with connect to module Bloky-LED.

Fist you have to connect module with Digital port, we have 4 port for Digital. With pinMode from D1 to D4 corresponding to Digtal output I/O pin 27 13 14 25

*-----------------------------------------------------------------------------------------------------------------------------*

*/\* Blink led with LoLin32 Board\*/*

*#define LED\_IN 27 // assign input digital pin for led module with port D1*

*void* ***setup****() {  
 // initialize digital pin LED\_BUILTIN as an output.  
 pinMode(LED\_IN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void* ***loop****() {  
 digitalWrite(LED\_IN, HIGH); // turn the LED on (HIGH is the voltage level)  
 delay(1000); // wait for a second  
 digitalWrite(LED\_IN, LOW); // turn the LED off by making the voltage LOW  
 delay(1000); // wait for a second  
}*

*-----------------------------------------------------------------------------------------------------------------------------*

# Analog read serial

*Introduction*

This example shows you how to read analog input from the physical world using a potentiometer. A potentiometer is a simple mechanical device that provides a varying amount of resistance when its shaft is turned. By passing voltage through a potentiometer and into an analog input on your board, it is possible to measure the amount of resistance produced by a potentiometer (or pot for short) as an analog value. In this example you will monitor the state of your potentiometer after establishing serial communication between your LoLin32 Lite Board and your computer running the Arduino Software (IDE).

*Hardware Required*

Lolin32 Lite board

Lolin32 Lite shield

Module Blocky Rotary

*Circuit*

Connect the module Blocky Rotary to your board with Blocky Shield corresponding to Analog A1 port.

The LoLin32 Lite boards have a circuit inside called an analog-to-digital converter or ADC that reads this changing voltage and converts it to a number between 0 and 4095 (12 bits ADC). In between, [analogRead](https://www.arduino.cc/en/Reference/AnalogRead)() returns a number between 0 and 4095 that is proportional to the amount of voltage being applied to the pin.

*Code*

You can get this code at link below

*-----------------------------------------------------------------------------------------------------------------------------*

/\*

AnalogReadSerial

Reads an analog input on pin 34, prints the result to the serial monitor.

Graphical representation is available using serial plotter (Tools > Serial Plotter menu)

Attach the center pin of a potentiometer to pin 34 with port A1,

and the outside pins to +3.3V and ground.

\*/

#define PIN\_ANALOG\_PORT\_1 34

// the setup routine runs once when you press reset:

void **setup()** {

// initialize serial communication at 115200 bits per second:

Serial.begin(115200);

}

// the loop routine runs over and over again forever:

void **loop()** {

// read the input on analog pin 34:

int sensorValue = analogRead(PIN\_ANALOG\_PORT\_1);

// print out the value you read:

Serial.println(sensorValue);

delay(100); // delay in between reads for stability

}

*-----------------------------------------------------------------------------------------------------------------------------*

# Blink led without delay

*Introduction*

This sketch demonstrates how to blink an LED without using delay(). It turns on the LED on and then makes note of the time. Then, each time through loop(), it checks to see if the desired blink time has passed. If it has, it toggles the LED on or off and makes note of the new time. In this way the LED blinks continuously while the sketch execution never lags on a single instruction.

*Hardware Required*

Lolin32 Lite board

Lolin32 Lite shield

Module Blocky-LED (optional)

*Circuit*

Connect the module Blocky-LED to your board with Blocky Shield corresponding to port D1.

You can use LED 22 on LoLin32 Lite board

*Code*

The code below uses the [millis()](https://www.arduino.cc/en/Reference/Millis) function, a command that returns the number of milliseconds since the board started running its current sketch, to blink a LED

You can get this code at link below

*-------------------------------------------------------------------------------------------------------------------*

// constants won't change. Used here to set a pin number:

const int ledPin = 22;// the number of the LED pin

const int ledPortD1 = 27;

// Variables will change:

int ledState = LOW; // ledState used to set the LED

// Generally, you should use "unsigned long" for variables that hold time

// The value will quickly become too large for an int to store

unsigned long previousMillis = 0; // will store last time LED was updated

// constants won't change:

const long interval = 1000; // interval at which to blink (milliseconds)

void ***setup()*** {

// set the digital pin as output:

***pinMode***(ledPin, OUTPUT);

}

void ***loop()*** {

// here is where you'd put code that needs to be running all the time.

// check to see if it's time to blink the LED; that is, if the difference

// between the current time and last time you blinked the LED is bigger than

// the interval at which you want to blink the LED.

unsigned long currentMillis = ***millis()***;

if (currentMillis - previousMillis >= interval) {

// save the last time you blinked the LED

previousMillis = currentMillis;

// if the LED is off turn it on and vice-versa:

if (ledState == LOW) {

ledState = HIGH;

} else {

ledState = LOW;

}

// set the LED with the ledState of the variable:

//digitalWrite(ledPortD1, ledState); // un-comment if you use the blocky-led module

***digitalWrite***(ledPin, ledState);

}

}

*-------------------------------------------------------------------------------------------------------------------*

# Digital read serial

*Introduction*

This example shows you how to monitor the state of a switch by establishing [serial communication](https://www.arduino.cc/en/Reference/Serial) between your LoLin32 Lite board and your computer over USB.

*Hardware Required*

Lolin32 Lite board

Lolin32 Lite shield

Module Button

*Circuit*

Connect module button to the board.

Pushbuttons or switches connect two points in a circuit when you press them. When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and reads as LOW, or 0.When the button is closed (pressed), it makes a connection between its two legs, connect the pin to 3.3 volts, so that the pin reads as HIGH, or 1.

*Code*

You can get this code at link below

*-------------------------------------------------------------------------------------------------------------------*

/\*

DigitalReadSerial

Reads a digital input on pin 27, prints the result to the serial monitor

This example code is in the public domain.

\*/

// digital pin 27 has a pushbutton attached to it. Give it a name:

// you must connect module button into port D1, with input pin is 27

int const pushButton = 27;

// the setup routine runs once when you press reset:

void ***setup***() {

// initialize serial communication at 115200 bits per second:

***Serial.begin***(115200);

// make the pushbutton's pin an input:

***pinMode***(pushButton, INPUT);

}

// the loop routine runs over and over again forever:

void ***loop***() {

// read the input pin:

int buttonState = digitalRead(pushButton);

// print out the state of the button:

***Serial.println***(buttonState);

***delay***(1); // delay in between reads for stability

}

*-------------------------------------------------------------------------------------------------------------------*

# Use button to control the LED

Introduction

Pushbuttons or switches connect two points in a circuit when you press them. This example turns on the built-in LED on pin 27 of port D1 when you press the button on pin 13 of port D2.

Hardware Required

Lolin32 Lite board

Lolin32 Lite shield

Module Button

Module Blocky-LED

Circuit

Connect module blocky-LED to port D1, and module button to port D2 on LoLin32 Lite shield.

When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 3.3V, so that we read a HIGH.

*Code*

You can get this code at link below

*-----------------------------------------------------------------------------------------------------------------------------*

*// constants won't change. They're used here to set pin numbers:*  
*const* int buttonPin = 13; *// the number of the pushbutton pin*  
*const* int ledPin = 27; *// the number of the LED pin*  
  
*// variables will change:*  
*int* buttonState = 0; *// variable for reading the pushbutton status*  
  
*void* **setup**() {  
 *// initialize the LED pin as an output:*  
pinMode(ledPin, OUTPUT);  
 *// initialize the pushbutton pin as an input:*  
pinMode(buttonPin, INPUT);  
}  
  
void **loop**() {  
 *// read the state of the pushbutton value:*  
 *buttonState =* digitalRead(buttonPin);  
  
 *// check if the pushbutton is pressed. If it is, the buttonState is HIGH:*  
if (buttonState == HIGH) {  
 *// turn LED on:*  
digitalWrite(ledPin, HIGH);  
 } else {  
 *// turn LED off:*  
digitalWrite(ledPin, LOW);  
 }  
}

------------------------------------------------------------------------------------------------------------------------------

# State change detection(edge detection) for pushbutton

Introduction

Once you've got a [pushbutton](https://www.arduino.cc/en/Tutorial/Button) working, you often want to do some action based on how many times the button is pushed. To do this, you need to know when the button changes state from off to on, and count how many times this change of state happens. This is called state change detection or edge detection.

Hardware Required

Lolin32 Lite board

Lolin32 Lite shield

Module Button

Circuit

Connect module button to LoLin32 Lite Board on port D1 of LoLin32 Lite shield

When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to voltage, so that we read a HIGH. (The pin is still connected to ground, but the resistor resists the flow of current, so the path of least resistance is to +3.3V)

*Code*

You can get this code at link below

The sketch below continually reads the button's state. It then compares the button's state to its state the last time through the main loop. If the current button state is different from the last button state and the current button state is high, then the button changed from off to on. The sketch then increments a button push counter.

The sketch also checks the button push counter's value, and if it's an even multiple of four, it turns the LED on pin 22 ON. Otherwise, it turns it off.

-------------------------------------------------------------------------------------------------------------------

*// this constant won't change:*  
*const* int buttonPin = 27; *// the pin that the pushbutton is attached to*  
*const* int ledPin = 22; *// the pin that the LED is attached to*  
  
*// Variables will change:*  
*int* buttonPushCounter = 0; *// counter for the number of button presses*  
*int* buttonState = 0; *// current state of the button*  
*int* lastButtonState = 0; *// previous state of the button*  
  
*void* **setup**() {  
 *// initialize the button pin as a input:*  
pinMode(buttonPin, INPUT);  
 *// initialize the LED as an output:*  
pinMode(ledPin, OUTPUT);  
 *// initialize serial communication:*  
Serial.begin(115200);  
}

*void* ***loop****() {  
 // read the pushbutton input pin:  
 buttonState = digitalRead(buttonPin);  
  
 // compare the buttonState to its previous state  
 if (buttonState != lastButtonState) {  
 // if the state has changed, increment the counter  
 if (buttonState == HIGH) {  
 // if the current state is HIGH then the button went from off to on:  
 buttonPushCounter++;  
 Serial.println("on");  
 Serial.print("number of button pushes: ");  
 Serial.println(buttonPushCounter);  
 } else {  
 // if the current state is LOW then the button went from on to off:  
 Serial.println("off");  
 }  
 // Delay a little bit to avoid bouncing  
 delay(50);  
 }  
 // save the current state as the last state, for next time through the loop  
 lastButtonState = buttonState;  
  
 // turns on the LED every four button pushes by checking the modulo of the  
 // button push counter. the modulo function gives you the remainder of the  
 // division of two numbers:  
 if (buttonPushCounter % 4 == 0) {  
 digitalWrite(ledPin, HIGH);  
 } else {  
 digitalWrite(ledPin, LOW);  
 }  
}*

-------------------------------------------------------------------------------------------------------------------

# LED PWM Fading

Introduction

The objective of this post is to explain how to fade a LED with the ESP32, using the LED PWM functionalities of the microcontroller. I will be using LoLin32 Lite board to perform the tests. Since the board as a built in LED, no external hardware will be needed.

Note that at the time of writing, the commonly used analogWrite Arduino function was not yet available for the ESP32 Arduino environment support. Thus, we will need to go to lower level functions in this tutorial. Nevertheless, we will also have more control and flexibility in the PWM functionality, which is good.

In terms of hardware, the LED PWM of the ESP32 is composed of 16 independent channels, with configurable duty cycles and wave periods. The accuracy of the duty cycle can be configured until 16 bits of resolution.

Hardware Required

Lolin32 Lite board

Lolin32 Lite shield

Module Blocky-LED (optional)

Circuit

Connect module Blocky-LED to LoLin32 Lite Board on port D1 of LoLin32 Lite shield.

You can use the LED 22 on the board if you don’t have module blocky-led.

Code

You can get this code at link below

----------------------------------------------------------------------------------------------------------

#define LED\_IN 27

int freq = 5000;

int ledChannel = 0;

int resolution = 8;

void setup() {

ledcSetup(ledChannel, freq, resolution);//setup the channel, frequency and resolution we specified

ledcAttachPin(LED\_IN, ledChannel);//attach the pin to the PWM channel

//ledcAttachPin(22, ledChannel); //un-comment this line if you LED 22 on board

}

void loop() {

for (int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){

ledcWrite(ledChannel, dutyCycle);

delay(7);

}

for (int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){

ledcWrite(ledChannel, dutyCycle);

delay(7);

}

}

-------------------------------------------------------------------------------------------------------------------

# **Use DHT-11 Sensor to check humidity and temperature**

*Introduction*

The DHT11 humidity and temperature sensor makes it really easy to add humidity and temperature data to your DIY electronics projects. It’s perfect for remote weather stations, home environmental control systems, and farm or garden monitoring systems.  
In this tutorial you will learn how to use this sensor with ESP. The room temperature & humidity will be printed to serial monitor. So, let's get started!

*Hardware Required*

LoLin32 Lite Board

Module DHT11 Sensor

*Circuit*

Connect the module DHT11 Sensor to your board with Blocky Shield corresponding to port Analog A1.

*Code*

Here's the code, don’t forget add DHT library.

------------------------------------------------------------------------------------------------------------------------------

//Libraries

#include "DHT.h"

//Constants

#define DHTPIN 22 // what pin we're connected to

#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE); //// Initialize DHT sensor for normal 16mhz Arduino

//Variables

int chk;

float hum; //Stores humidity value

float temp; //Stores temperature value

void setup()

{Serial.begin(9600);

dht.begin();

}

void loop()

{

//Read data and store it to variables hum and temp

hum = dht.readHumidity();

temp= dht.readTemperature();

//Print temp and humidity values to serial monitor

Serial.print("Humidity: ");

Serial.print(hum);

Serial.print(" %, Temp: ");

Serial.print(temp);

Serial.println(" Celsius");

delay(2000); //Delay 2 sec.

}

------------------------------------------------------------------------------------------------------------------------------

# Use a LDR Sensor

*Introduction*

Wouldn’t it be cool if we could eliminate darkness? In this Arduino project, I have posted a very simple project that focuses on eliminating darkness. Whenever a room gets dark due to a fused bulb or any other factors, a light bulb automatically turns ON. This can even be used as an emergency lighting system. It can be used to automatically turn a light ON whenever there isn’t sufficient light in a room.

In order to detect the intensity of light or darkness, we use a sensor called an LDR (Light Dependent Resistor). The LDR is a special type of resistor which allows higher voltages to pass through it (low resistance) whenever there is a high intensity of light, and passes a low voltage (high resistance) whenever it is dark. We can take advantage of this LDR property and use it in our DIY Arduino LDR sensor project

*Hardware Required*

LoLin32 Lite Board

Module Light Sensor

*Circuit*

Connect the module Light Sensor to your board with Blocky Shield corresponding to port Analog A1.

*Code*

Here's the code.

------------------------------------------------------------------------------------------------------------------------------

int sensorPin = 22; // select the input pin for LDR

int sensorValue = 0; // variable to store the value coming from the sensor

void setup() {

Serial.begin(9600); //sets serial port for communication

}

void loop() {

sensorValue = analogRead(sensorPin); // read the value from the sensor

Serial.println(sensorValue); //prints the values coming from the sensor on the screen

delay(5000); //Read data every 5s

}

------------------------------------------------------------------------------------------------------------------------------

# Use Soil Moisture Sensor

*Introduction*

Have you ever wanted your plants to tell you when they need watered? Or know how saturated the soil in your garden is? With the Soil Moisture Sensor, you can do just that! This tutorial will show you how to get started using the Soil Moisture Sensor.

*Hardware Required*

LoLin32 Lite Board

Module Soil Moisture Sensor

*Circuit*

Connect the module Soil Moisture Sensor to your board with Blocky Shield corresponding to port Analog A1.

*Code*

Here's the code,

------------------------------------------------------------------------------------------------------------------------------

int sensorPin = 22; // select the input pin for Soil Moisture Sensor

int sensorValue = 0; // variable to store the value coming from the sensor

void setup() {

Serial.begin(9600); //sets serial port for communication

}

void loop() {

sensorValue = analogRead(sensorPin); // read the value from the sensor

Serial.println(sensorValue); //prints the values coming from the sensor on the screen

delay(5000); //Read data every 5s

}

------------------------------------------------------------------------------------------------------------------------------

# Use Sound Sensor

*Introduction*

Sound sensors can be used for a variety of things, one of them could be turning lights off and on by clapping.

This tutorial will show you how to turn on, turn off LED by clap your hands.

*Hardware Required*

LoLin32 Lite Board

Module Sound Sensor

Module LED

*Circuit*

Connect the module Sound Sensor to your board with Blocky Shield corresponding to port Analog A4, module to port Digital D2.

*Code*

Here's the code,

------------------------------------------------------------------------------------------------------------------------------

int soundSensor = 4; //Pin 4 for Sound Sensor

int LED = 13; //Pin 13 for LED

int statusSensor = 0;

bool switchLED = false;

void setup()

{

pinMode (LED, OUTPUT);

}

void loop()

{

statusSensor = analogRead (soundSensor);

if(statusSensor > 500)

{

switchLED = !switchLED;

}

if (switchLED)

{

digitalWrite(LED, HIGH);

}

else

{

digitalWrite(LED, LOW);

}

delay(100);

}

------------------------------------------------------------------------------------------------------------------------------

# Use Buzzer

*Introduction*

Buzzers can be found in alarm devices, computers, timers and confirmation of user input such as a mouse click or keystroke. In this tutorial you will learn how to use a buzzer or piezo speaker with ESP32.

*Hardware Required*

LoLin32 Lite Board

Module Buzzer

*Circuit*

Connect the module Buzzer to your board with Blocky Shield corresponding to port Digital D1.

*Code*

Here's the code,

------------------------------------------------------------------------------------------------------------------------------

#define speakerPin 27

void setup() {

// put your setup code here, to run once:

pinMode (speakerPin, OUTPUT);

}

void loop() {

// put your main code here, to run repeatedly:

digitalWrite(speakerPin, HIGH);

delay(200);

digitalWrite(speakerPin, LOW);

delay(1000);

}

------------------------------------------------------------------------------------------------------------------------------

# Matrix Led Neopixel

*Introduction*

This demonstration show you how to connect a LED Matrix module to LoLin32 Lite Board via Neopixel module to display information from ESP32. A Simple example how to turn on all led of NeoPixel module.

Arduino library for controlling single-wire-based LED pixels and strip such as the [Adafruit 60 LED/meter Digital LED strip](http://adafruit.com/products/1138), the [Adafruit FLORA RGB Smart Pixel](http://adafruit.com/products/1060), the [Adafruit Breadboard-friendly RGB Smart Pixel](http://adafruit.com/products/1312), the [Adafruit NeoPixel Stick](http://adafruit.com/products/1426), and the [Adafruit NeoPixel Shield](http://adafruit.com/products/1430).

*Hardware Required*

LoLin32 Lite Board

Lolin32 Lite shield

Module Neopixel

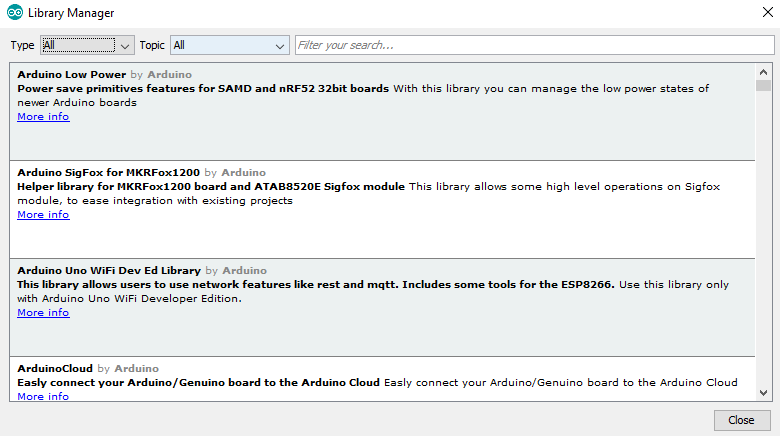
*Circuit*

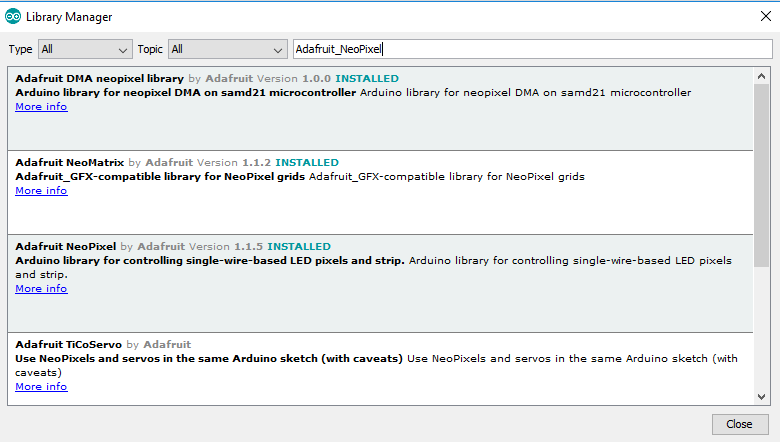
Connect the module Neopixel to your board with Blocky Shield corresponding to port digital D1.

*Note*

Before you start this example, you must include the library of this module Neopixel.

Click “Sketch→Include library → Manage libraries”

 After that search “ Adafruit\_NeoPixel” in Filter fill, and install “Adafruit NeoPixel”

*Code*

You can get this code at link below

--------------------------------------------------------------------------------------------------------------------

#include <Adafruit\_NeoPixel.h>

#ifdef \_\_AVR\_\_

#include <avr/power.h>

#endif

// Which pin on the Arduino is connected to the NeoPixels?

// On a Trinket or Gemma we suggest changing this to 1

#define PIN 27

// How many NeoPixels are attached to the Arduino?

#define NUMPIXELS 9

// When we setup the NeoPixel library, we tell it how many pixels, and which pin to use to send signals.

// Note that for older NeoPixel strips you might need to change the third parameter--see the strandtest

// example for more information on possible values.

Adafruit\_NeoPixel pixels = Adafruit\_NeoPixel(NUMPIXELS, PIN, NEO\_GRB + NEO\_KHZ800);

int delayval = 500; // delay for half a second

void setup() {

#if defined (\_\_AVR\_ATtiny85\_\_)

if (F\_CPU == 16000000) clock\_prescale\_set(clock\_div\_1);

#endif

// End of trinket special code

pixels.begin(); // This initializes the NeoPixel library.

}

void loop() {

// For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way up to the count of pixels minus one.

for(int i=0;i<NUMPIXELS;i++){

// pixels.Color takes RGB values, from 0,0,0 up to 255,255,255

pixels.setPixelColor(i, pixels.Color(50,50,50)); // Moderately bright green color.

pixels.show(); // This sends the updated pixel color to the hardware.

delay(delayval); // Delay for a period of time (in milliseconds).

}

}

--------------------------------------------------------------------------------------------------------------------

# More example about Matrix Led Neopixel

*Introduction*

This section show you more example to use this module led matrix Neopixel. Having fun.

*Hardware Required*

LoLin32 Lite Board

Lolin32 Lite shield

Module Neopixel

*Circuit*

Connect the module Neopixel to your board with Blocky Shield corresponding to port digital D1.

*Code*

You can get this code at link below

/// This example will blink a random rainbow led

#include <Adafruit\_NeoPixel.h>

#ifdef \_\_AVR\_\_

#include <avr/power.h>

#endif

#define PIN 27 // responding to Digial pin of port D1, you can use another port

#define NUM\_LEDS 9

#define BRIGHTNESS 50

// setup variable for neopixel module

Adafruit\_NeoPixel strip = Adafruit\_NeoPixel(NUM\_LEDS, PIN, NEO\_GRBW + NEO\_KHZ800);

byte neopix\_gamma[] = {

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,

2, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5,

5, 6, 6, 6, 6, 7, 7, 7, 7, 8, 8, 8, 9, 9, 9, 10,

10, 10, 11, 11, 11, 12, 12, 13, 13, 13, 14, 14, 15, 15, 16, 16,

17, 17, 18, 18, 19, 19, 20, 20, 21, 21, 22, 22, 23, 24, 24, 25,

25, 26, 27, 27, 28, 29, 29, 30, 31, 32, 32, 33, 34, 35, 35, 36,

37, 38, 39, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 50,

51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68,

69, 70, 72, 73, 74, 75, 77, 78, 79, 81, 82, 83, 85, 86, 87, 89,

90, 92, 93, 95, 96, 98, 99,101,102,104,105,107,109,110,112,114,

115,117,119,120,122,124,126,127,129,131,133,135,137,138,140,142,

144,146,148,150,152,154,156,158,160,162,164,167,169,171,173,175,

177,180,182,184,186,189,191,193,196,198,200,203,205,208,210,213,

215,218,220,223,225,228,231,233,236,239,241,244,247,249,252,255 };

void setup() {

#if defined (\_\_AVR\_ATtiny85\_\_)

if (F\_CPU == 16000000) clock\_prescale\_set(clock\_div\_1);

#endif

// End of trinket special code

strip.setBrightness(BRIGHTNESS);

strip.begin();

strip.show(); // Initialize all pixels to 'off'

}

void loop() {

// Some example procedures showing how to display to the pixels:

colorWipe(strip.Color(255, 0, 0), 50); // Red

colorWipe(strip.Color(0, 255, 0), 50); // Green

colorWipe(strip.Color(0, 0, 255), 50); // Blue

colorWipe(strip.Color(0, 0, 0, 255), 50); // White

whiteOverRainbow(20,75,5);

pulseWhite(5);

fullWhite();

delay(2000);

rainbowFade2White(3,3,1);

}

// Fill the dots one after the other with a color

void colorWipe(uint32\_t c, uint8\_t wait) {

for(uint16\_t i=0; i<strip.numPixels(); i++) {

strip.setPixelColor(i, c);

strip.show();

delay(wait);

}

}

void pulseWhite(uint8\_t wait) {

for(int j = 0; j < 256 ; j++){

for(uint16\_t i=0; i<strip.numPixels(); i++) {

strip.setPixelColor(i, strip.Color(0,0,0, neopix\_gamma[j] ) );

}

delay(wait);

strip.show();

}

for(int j = 255; j >= 0 ; j--){

for(uint16\_t i=0; i<strip.numPixels(); i++) {

strip.setPixelColor(i, strip.Color(0,0,0, neopix\_gamma[j] ) );

}

delay(wait);

strip.show();

}

}

void rainbowFade2White(uint8\_t wait, int rainbowLoops, int whiteLoops) {

float fadeMax = 100.0;

int fadeVal = 0;

uint32\_t wheelVal;

int redVal, greenVal, blueVal;

for(int k = 0 ; k < rainbowLoops ; k ++){

for(int j=0; j<256; j++) { // 5 cycles of all colors on wheel

for(int i=0; i< strip.numPixels(); i++) {

wheelVal = Wheel(((i \* 256 / strip.numPixels()) + j) & 255);

redVal = red(wheelVal) \* float(fadeVal/fadeMax);

greenVal = green(wheelVal) \* float(fadeVal/fadeMax);

blueVal = blue(wheelVal) \* float(fadeVal/fadeMax);

strip.setPixelColor( i, strip.Color( redVal, greenVal, blueVal ) );

}

//First loop, fade in!

if(k == 0 && fadeVal < fadeMax-1) {

fadeVal++;

}

//Last loop, fade out!

else if(k == rainbowLoops - 1 && j > 255 - fadeMax ){

fadeVal--;

}

strip.show();

delay(wait);

}

}

delay(500);

for(int k = 0 ; k < whiteLoops ; k ++){

for(int j = 0; j < 256 ; j++){

for(uint16\_t i=0; i < strip.numPixels(); i++) {

strip.setPixelColor(i, strip.Color(0,0,0, neopix\_gamma[j] ) );

}

strip.show();

}

delay(2000);

for(int j = 255; j >= 0 ; j--){

for(uint16\_t i=0; i < strip.numPixels(); i++) {

strip.setPixelColor(i, strip.Color(0,0,0, neopix\_gamma[j] ) );

}

strip.show();

}

}

delay(500);

}

void whiteOverRainbow(uint8\_t wait, uint8\_t whiteSpeed, uint8\_t whiteLength ) {

if(whiteLength >= strip.numPixels()) whiteLength = strip.numPixels() - 1;

int head = whiteLength - 1;

int tail = 0;

int loops = 3;

int loopNum = 0;

static unsigned long lastTime = 0;

while(true){

for(int j=0; j<256; j++) {

for(uint16\_t i=0; i<strip.numPixels(); i++) {

if((i >= tail && i <= head) || (tail > head && i >= tail) || (tail > head && i <= head) ){

strip.setPixelColor(i, strip.Color(0,0,0, 255 ) );

}

else{

strip.setPixelColor(i, Wheel(((i \* 256 / strip.numPixels()) + j) & 255));

}

}

if(millis() - lastTime > whiteSpeed) {

head++;

tail++;

if(head == strip.numPixels()){

loopNum++;

}

lastTime = millis();

}

if(loopNum == loops) return;

head%=strip.numPixels();

tail%=strip.numPixels();

strip.show();

delay(wait);

}

}

}

void fullWhite() {

for(uint16\_t i=0; i<strip.numPixels(); i++) {

strip.setPixelColor(i, strip.Color(100,100,100 ) );

}

strip.show();

}

// Slightly different, this makes the rainbow equally distributed throughout

void rainbowCycle(uint8\_t wait) {

uint16\_t i, j;

for(j=0; j<256 \* 5; j++) { // 5 cycles of all colors on wheel

for(i=0; i< strip.numPixels(); i++) {

strip.setPixelColor(i, Wheel(((i \* 256 / strip.numPixels()) + j) & 255));

}

strip.show();

delay(wait);

}

}

//make rainbow led

void rainbow(uint8\_t wait) {

uint16\_t i, j;

for(j=0; j<256; j++) {

for(i=0; i<strip.numPixels(); i++) {

strip.setPixelColor(i, Wheel((i+j) & 255));

}

strip.show();

delay(wait);

}

}

// Input a value 0 to 255 to get a color value.

// The colours are a transition r - g - b - back to r.

uint32\_t Wheel(byte WheelPos) {

WheelPos = 255 - WheelPos;

if(WheelPos < 85) {

return strip.Color(255 - WheelPos \* 3, 0, WheelPos \* 3,0);

}

if(WheelPos < 170) {

WheelPos -= 85;

return strip.Color(0, WheelPos \* 3, 255 - WheelPos \* 3,0);

}

WheelPos -= 170;

return strip.Color(WheelPos \* 3, 255 - WheelPos \* 3, 0,0);

}

uint8\_t red(uint32\_t c) {

return (c >> 16);

}

uint8\_t green(uint32\_t c) {

return (c >> 8);

}

uint8\_t blue(uint32\_t c) {

return (c);

}

------------------------------------------------------------------------------------------------------------------------------

*Code*

This code demo about advertising led. Haing-fun!

You can try more example at link below:

------------------------------------------------------------------------------------------------------------------------------

# Using OLED 0,96"

*Introduction*

This OLED 0.96" is built-in SSD1306 controller IC, it communicates via 6800/8080 8-bit parallel and I2C/4-wire serial interface. This model is also very suitable for wearable device. These OLED COG modules are ultra thin, lightweight and low power consumption which is great for handheld instruments, meters, smart grid, wearable device, medical device, IoT, mp3, etc.

In this tutorial will help you how to communicate OLED 0.96" with ESP32 Lite.

*Hardware Required*

LoLin32 Lite Board

Module OLED 0,96”

*Circuit*

Connect the Module OLED 0,96” to your board with Blocky Shield corresponding to Digital port D1.

*Code*

Here's the code, don’t forget add esp8266 oled library (esp8266-oled-ssd1306-master.zip).

--------------------------------------------------------------------------------------------------------------------

#include <Wire.h>

#include <SSD1306.h>

// Initialize the OLED display using Wire library

SSD1306 display(0x3c, 27, 19); //Pin 19: SCL, pin 27:SDA

// Adapted from Adafruit\_SSD1306

void drawLines() {

for (int16\_t i=0; i<DISPLAY\_WIDTH; i+=4) {

display.drawLine(0, 0, i, DISPLAY\_HEIGHT-1);

display.display();

delay(10);

}

for (int16\_t i=0; i<DISPLAY\_HEIGHT; i+=4) {

display.drawLine(0, 0, DISPLAY\_WIDTH-1, i);

display.display();

delay(10);

}

delay(250);

display.clear();

for (int16\_t i=0; i<DISPLAY\_WIDTH; i+=4) {

display.drawLine(0, DISPLAY\_HEIGHT-1, i, 0);

display.display();

delay(10);

}

for (int16\_t i=DISPLAY\_HEIGHT-1; i>=0; i-=4) {

display.drawLine(0, DISPLAY\_HEIGHT-1, DISPLAY\_WIDTH-1, i);

display.display();

delay(10);

}

delay(250);

display.clear();

for (int16\_t i=DISPLAY\_WIDTH-1; i>=0; i-=4) {

display.drawLine(DISPLAY\_WIDTH-1, DISPLAY\_HEIGHT-1, i, 0);

display.display();

delay(10);

}

for (int16\_t i=DISPLAY\_HEIGHT-1; i>=0; i-=4) {

display.drawLine(DISPLAY\_WIDTH-1, DISPLAY\_HEIGHT-1, 0, i);

display.display();

delay(10);

}

delay(250);

display.clear();

for (int16\_t i=0; i<DISPLAY\_HEIGHT; i+=4) {

display.drawLine(DISPLAY\_WIDTH-1, 0, 0, i);

display.display();

delay(10);

}

for (int16\_t i=0; i<DISPLAY\_WIDTH; i+=4) {

display.drawLine(DISPLAY\_WIDTH-1, 0, i, DISPLAY\_HEIGHT-1);

display.display();

delay(10);

}

delay(250);

}

// Adapted from Adafruit\_SSD1306

void drawRect(void) {

for (int16\_t i=0; i<DISPLAY\_HEIGHT/2; i+=2) {

display.drawRect(i, i, DISPLAY\_WIDTH-2\*i, DISPLAY\_HEIGHT-2\*i);

display.display();

delay(10);

}

}

// Adapted from Adafruit\_SSD1306

void fillRect(void) {

uint8\_t color = 1;

for (int16\_t i=0; i<DISPLAY\_HEIGHT/2; i+=3) {

display.setColor((color % 2 == 0) ? BLACK : WHITE); // alternate colors

display.fillRect(i, i, DISPLAY\_WIDTH - i\*2, DISPLAY\_HEIGHT - i\*2);

display.display();

delay(10);

color++;

}

// Reset back to WHITE

display.setColor(WHITE);

}

// Adapted from Adafruit\_SSD1306

void drawCircle(void) {

for (int16\_t i=0; i<DISPLAY\_HEIGHT; i+=2) {

display.drawCircle(DISPLAY\_WIDTH/2, DISPLAY\_HEIGHT/2, i);

display.display();

delay(10);

}

delay(1000);

display.clear();

// This will draw the part of the circel in quadrant 1

// Quadrants are numberd like this:

// 0010 | 0001

// ------|-----

// 0100 | 1000

//

display.drawCircleQuads(DISPLAY\_WIDTH/2, DISPLAY\_HEIGHT/2, DISPLAY\_HEIGHT/4, 0b00000001);

display.display();

delay(200);

display.drawCircleQuads(DISPLAY\_WIDTH/2, DISPLAY\_HEIGHT/2, DISPLAY\_HEIGHT/4, 0b00000011);

display.display();

delay(200);

display.drawCircleQuads(DISPLAY\_WIDTH/2, DISPLAY\_HEIGHT/2, DISPLAY\_HEIGHT/4, 0b00000111);

display.display();

delay(200);

display.drawCircleQuads(DISPLAY\_WIDTH/2, DISPLAY\_HEIGHT/2, DISPLAY\_HEIGHT/4, 0b00001111);

display.display();

}

void printBuffer(void) {

// Initialize the log buffer

// allocate memory to store 8 lines of text and 30 chars per line.

display.setLogBuffer(5, 30);

// Some test data

const char\* test[] = {

"Hello",

"World" ,

"----",

"Wellcome",

"to",

"TIC"

};

for (uint8\_t i = 0; i < 11; i++) {

display.clear();

// Print to the screen

display.println(test[i]);

// Draw it to the internal screen buffer

display.drawLogBuffer(0, 0);

// Display it on the screen

display.display();

delay(500);

}

}

void setup() {

display.init();

// display.flipScreenVertically();

display.setContrast(255);

drawLines();

delay(1000);

display.clear();

drawRect();

delay(1000);

display.clear();

fillRect();

delay(1000);

display.clear();

drawCircle();

delay(1000);

display.clear();

printBuffer();

delay(1000);

display.clear();

}

void loop() { }

--------------------------------------------------------------------------------------------------------------------