



Arduino Learners kit

Step-by-step Guide

What is Arduino?

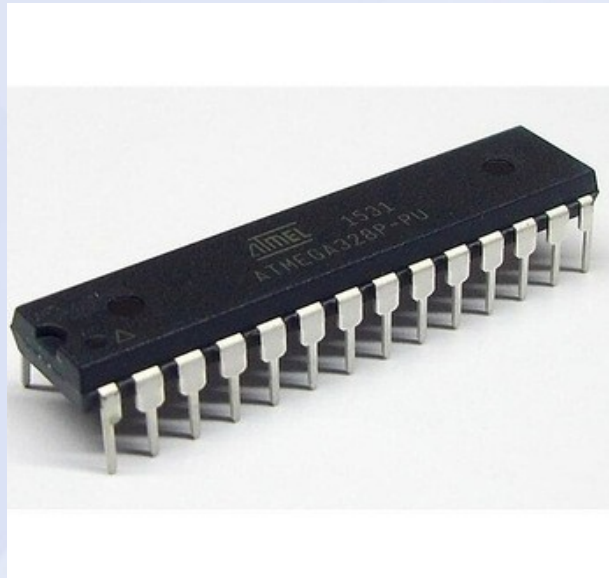
Open-Source

Hardware Platform

Software (Arduino IDE)

Website: www.arduino.cc

Atmega328P microcontroller



Very small

Very low power computer

2KB RAM

Some applications



Microwave Oven

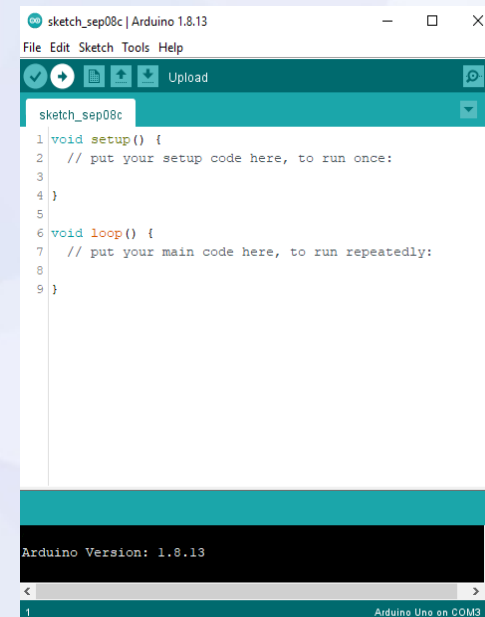


Modern Cars

Arduino board and IDE



Arduino board compatible



The IDE

Materials

GizDuino SE or Student Edition (Arduino UNO)

2 pins Tact Switch
10K Ohms 1/4W Resistors

Red,Green,Blue LED 5mm
470 Ohms 1/4W Resistors

DHT11 Temperature and Humidity sensor

HC-SR04 Distance sensor

5V Relay Module Active-Low

2-channel Tiny Motor Driver Module

DC Motor

Servo motor SG-90 160 degrees

2X16 LCD Blue back light with I2C

Passive Buzzer

Brief explanation about Microcontrollers.

I. Introduction to gizDuino Student Edition

a. Parts and descriptions

b. Specifications and Driver installation

c. Software Arduino IDE installation, board and comport select

II. Basic structure

a. void setup (pinMode, input, output)

- **Examples of Output and Input devices**

b. void loop (Digital and Analog I/O)

III. Exercises

Part 1. LED Blinking with Manual Wiring

- **How to use breadboard**
- **LED polarity**

Part 2. Multiple LEDs with Manual Wiring

Part 3. Latch and Push button switch

Part 4. PassiveBuzzer - alarm

Part 5. Relay – switching ON/OFF device

Part 6. DHT 11 Temperature and Humidity Sensor

- **How to add a DHT Library**

Part 7. HC-SR04 Distance Sensor – Distance measure

Part 8. Servo Motor

Part 8. DC motor and the driver – Motor Controls

Part 9. 2x16 Characters LCD Display Blue Backlight with I2C module

I. Introduction to gizDuino SE

What is gizDuino SE or Student Edition?

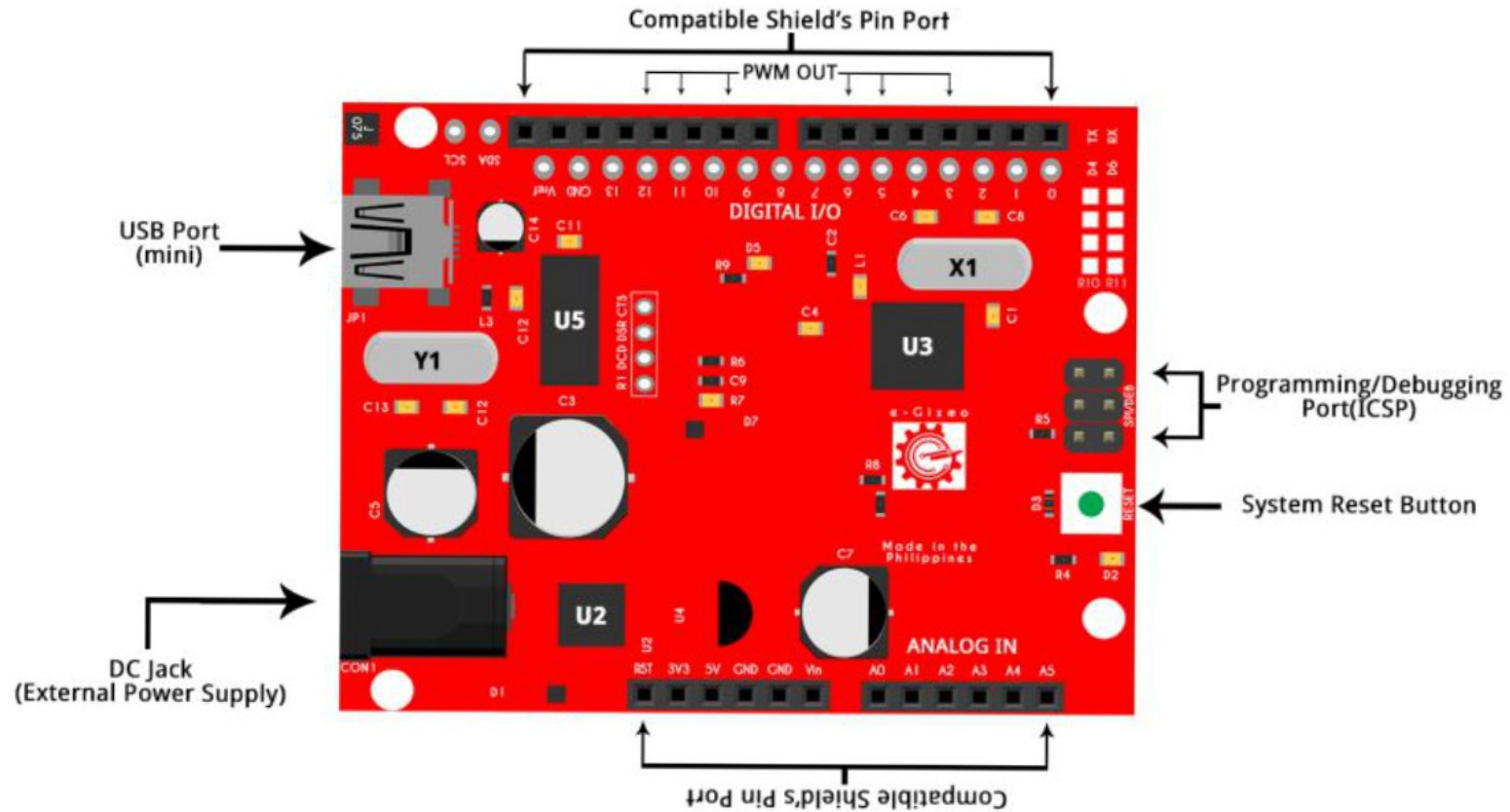
- **It is an Arduino UNO compatible board.**
- **It has CH340 Driver and needed to install manually.**
- **It is Ardino Compatible Shield**

gizDuino SE

- is 100% code compatible with Arduino UNO. It is built for the budget but with the Best possible quality. To obtain to most competitive price, we used the same USB Bridge CH340 universally used by China UNO compatible manufacturers. Hence, If you are already user of these boards, you can switch to the gizDuino UNO-SE And not notice a difference in usage.

- Genuine ATMEL AVR with ATMEGA328P MCU Core**
- 100% Code compatible**
- CH340 USB to UART bridge.**

Parts and Descriptions



- U2 - On-board Voltage Regulator**
- U3 - Atmega328p**
- U5 - CH340 Driver**
- X1 - 16MHz Crystal**
- Y1 - 12MHz Crystal**

General Specifications

Microcontroller: **ATMEGA328P**

Power Input (DC): **USB (+5V), External (7~12V)**

Input Voltage (limits): **6~20V**

DC Current per pins: **40mA**

Digital I/O Pins: **20 Digital (6 for PWM)**

Analog Input Pins: **6**

Hardware Serial Pin: **1**

Flash Memory: **32KB (2KB used by boot loader)**



Software Arduino IDE installation

Arduino IDE

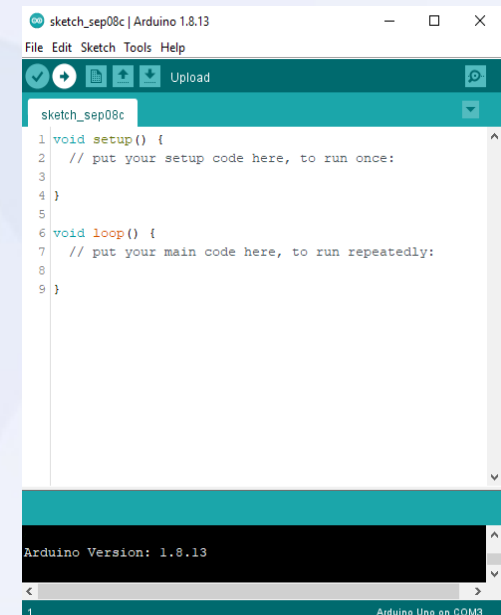
[www.e-gizmo.net/oc/kits documents/ARDUINO IDE SOFTWARES](http://www.e-gizmo.net/oc/kits%20documents/ARDUINO%20IDE%20SOFTWARES)

Choose your Arduino IDE for your OS.

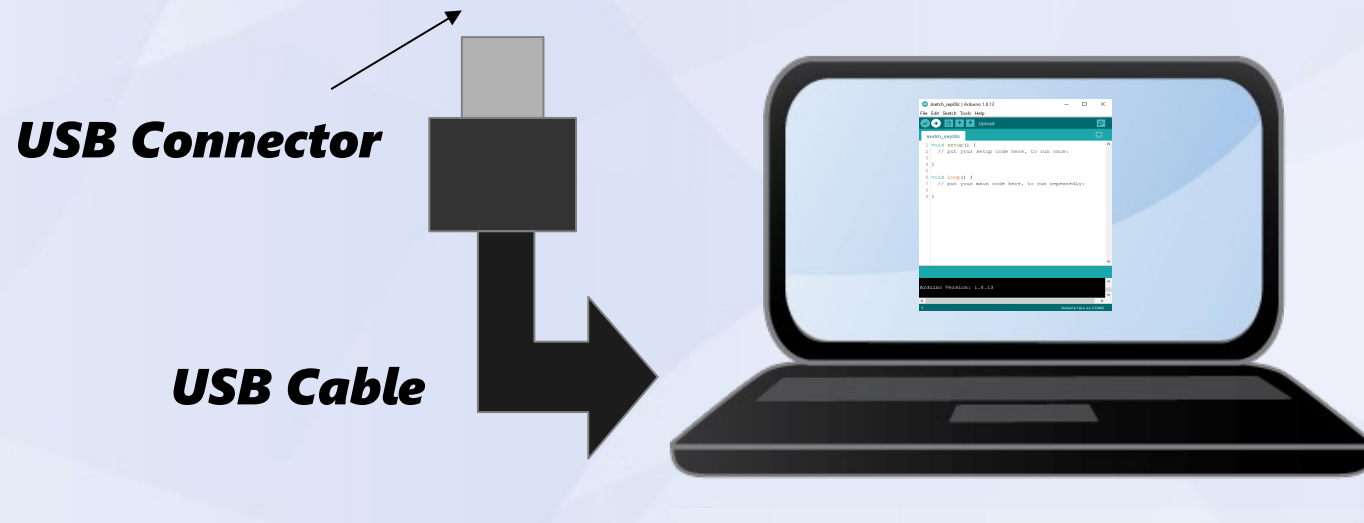
Drivers

Install this first!

<http://e-gizmo.net/oc/kits%20documents/CH340%20USB-TTL/drivers.zip>



Connect the gizDuino SE to PC

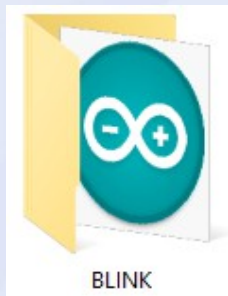


Upload BLINK.ino

On the Arduino IDE.

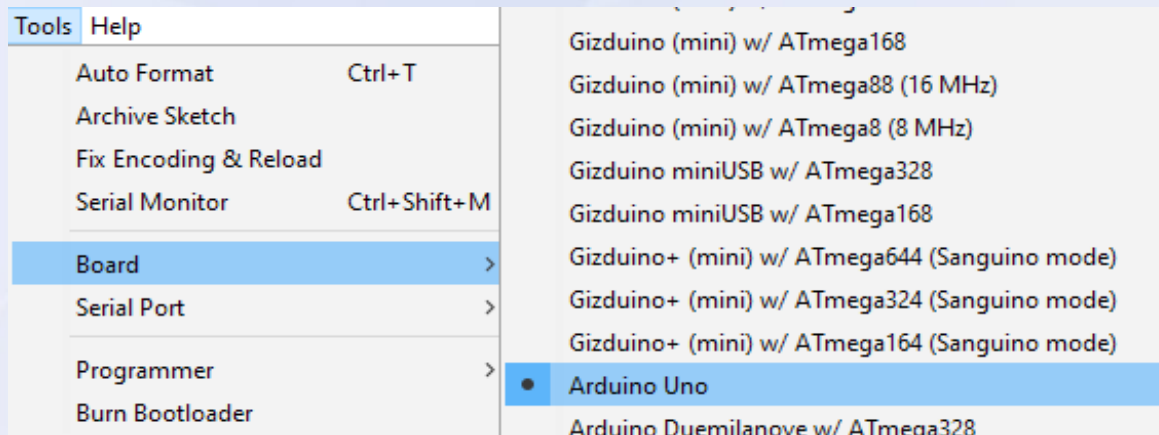
1. BLINK codes

Go to File> Open > BLINK.ino



2. Board select

Go to Tools>Boards>Arduino UNO

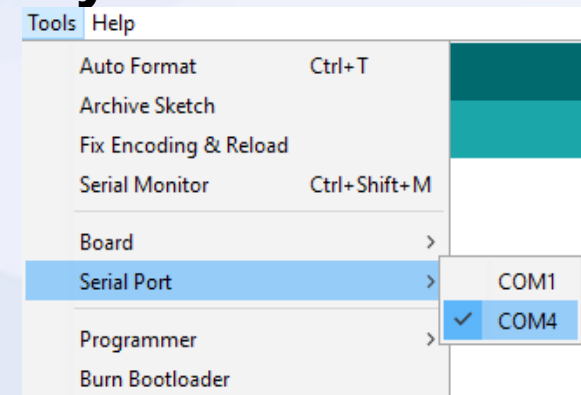


3. Port select

Go to Tools>Port>COM#

Select the correct port

**Go to Device Manager
if you're not sure.**



II. Basic structure

```
1 void setup() {  
2   // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8  
9 }
```

setup()

- is called when a sketch starts.
- it will only run once, after power up or reset the device.

Loop()

- loops consecutively.
- actively control the Arduino device.

//

- a single line comment begins with // (two adjacent slashes)

/**/

- a block or multi-line comment is marked by the symbol /* and the symbol */

Digital I/O and Time Functions

```
1 void setup() {  
2  
3   pinMode(LED_BUILTIN, OUTPUT);  
4  
5 }
```

```
8 void loop() {  
9   digitalWrite(LED_BUILTIN, HIGH);  
10  delay(1000);  
11  digitalWrite(LED_BUILTIN, LOW);  
12  delay(1000);  
13 }
```

pinmode(pin,mode);

Pin: arduino pin number to set mode of.

Mode: INPUT, OUTPUT or INPUT_PULLUP.

LED_BUILTIN

= 13 or Digital pin 13

digitalWrite(pin,value);

Pin: arduino pin number to set mode of.

Value: HIGH or LOW

delay(ms);

ms: the number of milliseconds to pause.

1000ms = 1second

What are those devices use in __?

Output devices

(Digital/PWM)

LED Lights

Transistor

Buzzer

Servo

DC Motor

Stepper

Relay

Input devices

(Digital/Analog)

Push Buttons

ADC (Analog to Digital Converter)

LDR (Light Dependent Resistor)

Temperature Sensor

Weight sensor

Compact Proximity

Voltage Measurement

Position

Serial Communication devices

(RX/TX/Modules/Shields/I2C)

GSM (Text/Call)

GPS (Location)

WiFi (ESP8266,LoRa)

Bluetooth (HC-05,HM-10..)

RF Modules (UHF)

Fingerprint

Industrial Equipment with Serial Connections

RFID

LCD,RTC I2C

SPI Communication devices

(MOSI/MISO/SCK/CS)

SD Card Shields (Data logging)

Ethernet Shield (ENC,Wiznet)

OLED Display

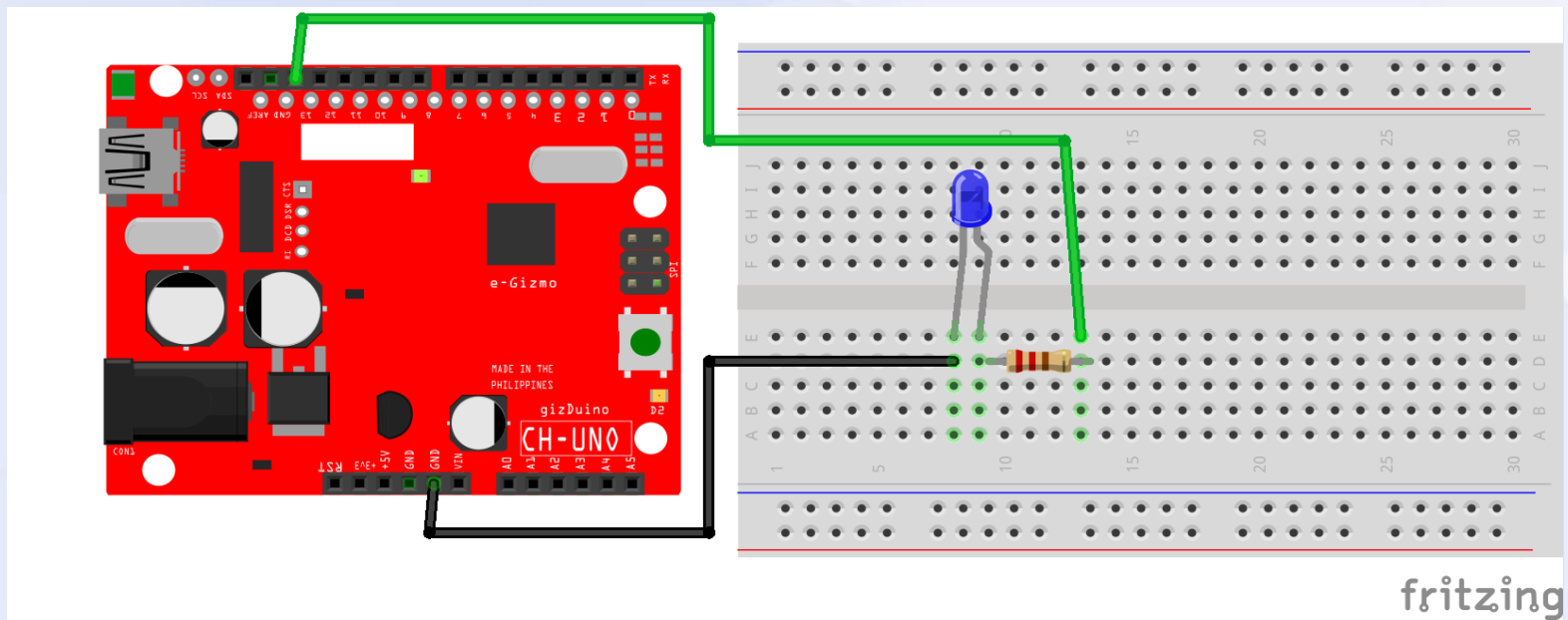


III. Exercises

Part 1. LED Blinking Manual Wiring

**This part shows you how to manually
Wiring the LED, Resistor to gizDuino.**

Upload the LEDBlinking.ino.



The code

```
8 #define LED 13 // assign a variable name in D13
```

```
10 void setup() {  
11  
12   pinMode(LED, OUTPUT); // Set D13 as Output  
13  
14 }
```

```
16 void loop() {  
17  
18   digitalWrite(LED, HIGH); // Turn ON LED  
19   delay(1000); // delay 1 sec = 1000 ms  
20   digitalWrite(LED, LOW); //Turn OFF LED  
21   delay(1000);
```

#define constantName value

constantName: the name of the macro to define.

value: the value to assign to the macro.

pinmode(pin,mode);

Pin: arduino pin number to set mode of.

Mode: INPUT,OUTPUT or INPUT_PULLUP.

digitalWrite(pin,value);

Pin: arduino pin number to set mode of.

Value: HIGH or LOW

delay(ms);

ms: the number of milliseconds to pause.

1000ms = 1second

How to use the breadboard.

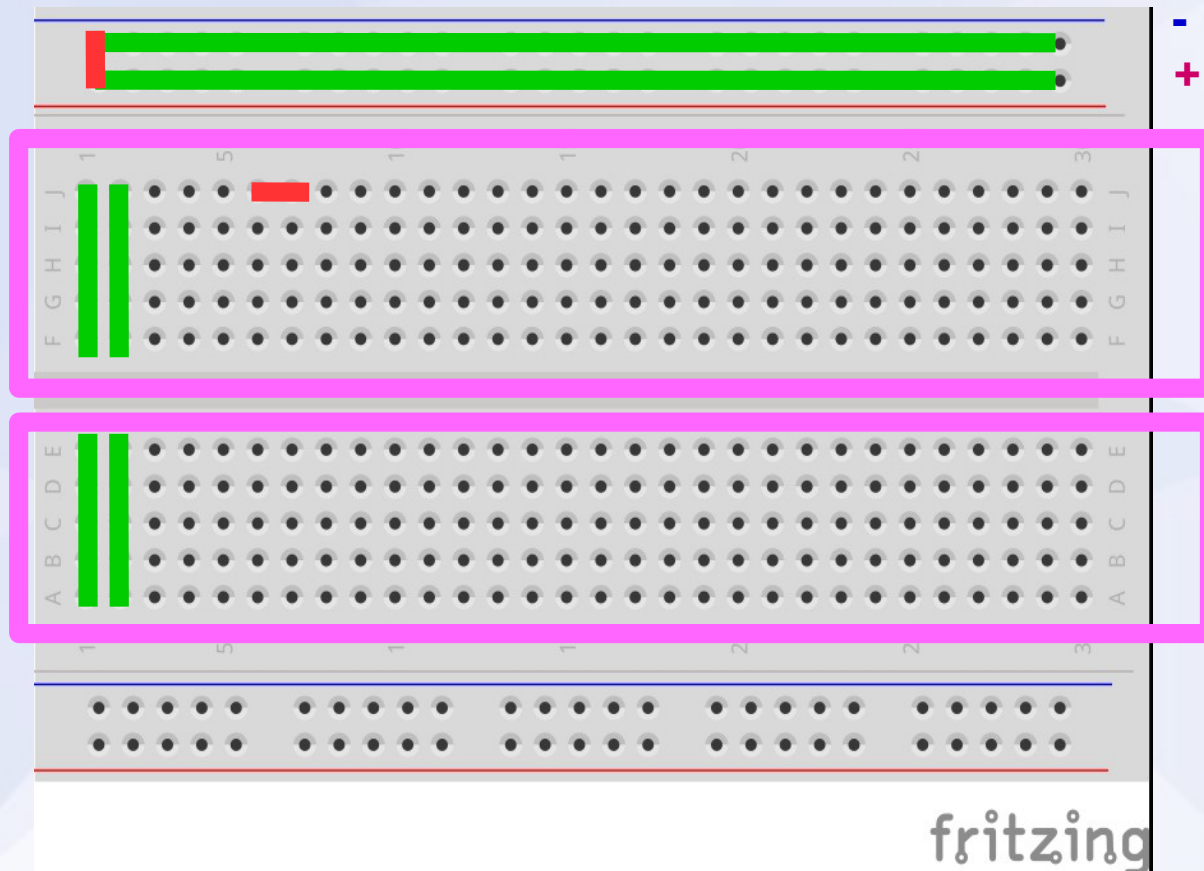
Horizontally Connected =

For Power Supply Extensions

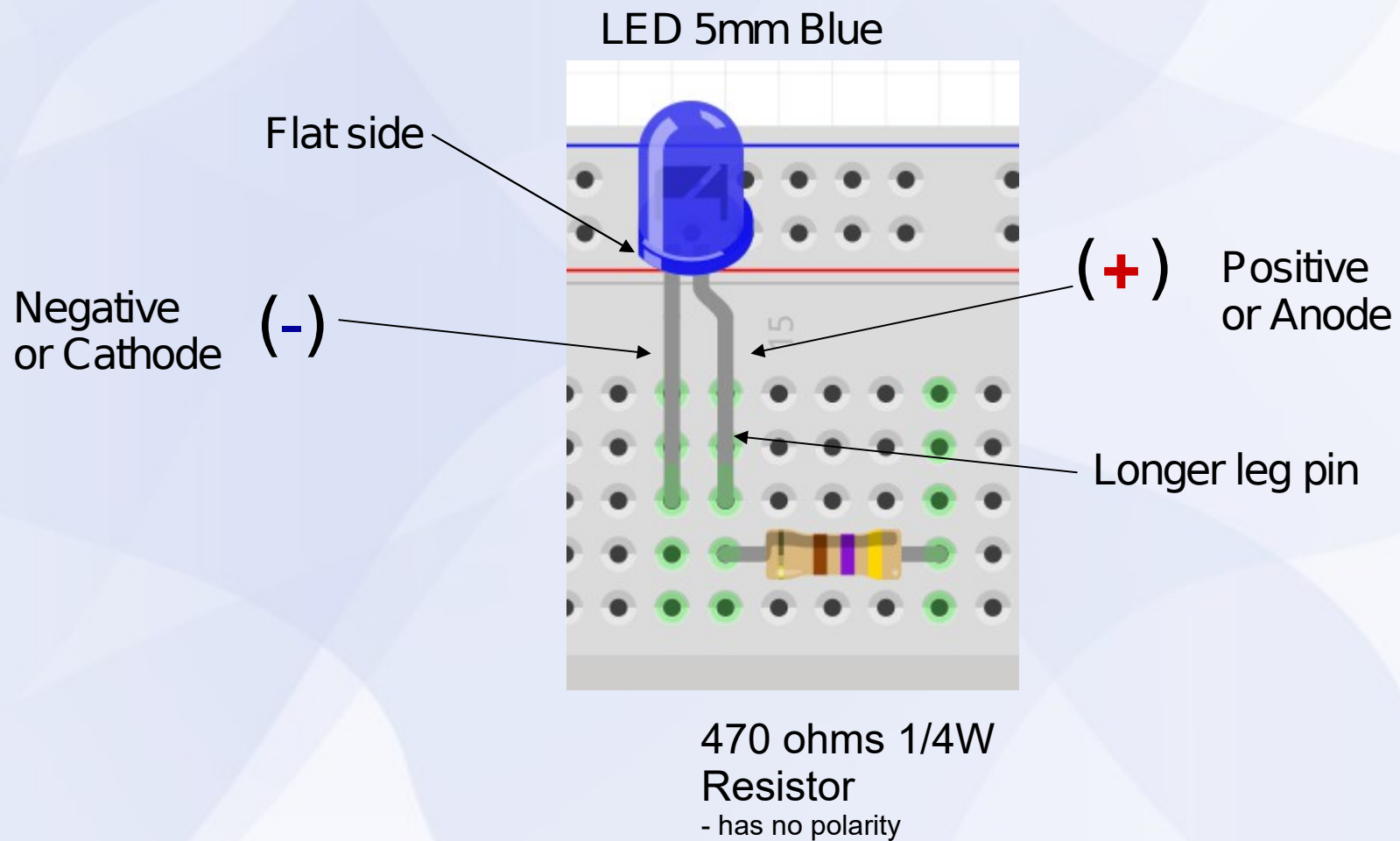
- **GND**
+ **VCC**

Vertically 
connected

For Components
Space such as
LED, resistor,
Sensors with male
Connectors and
Jumper wires.



LED polarity





e-Gizmo
MECHATRONIX CENTRAL

Part 2. Multiple LEDs

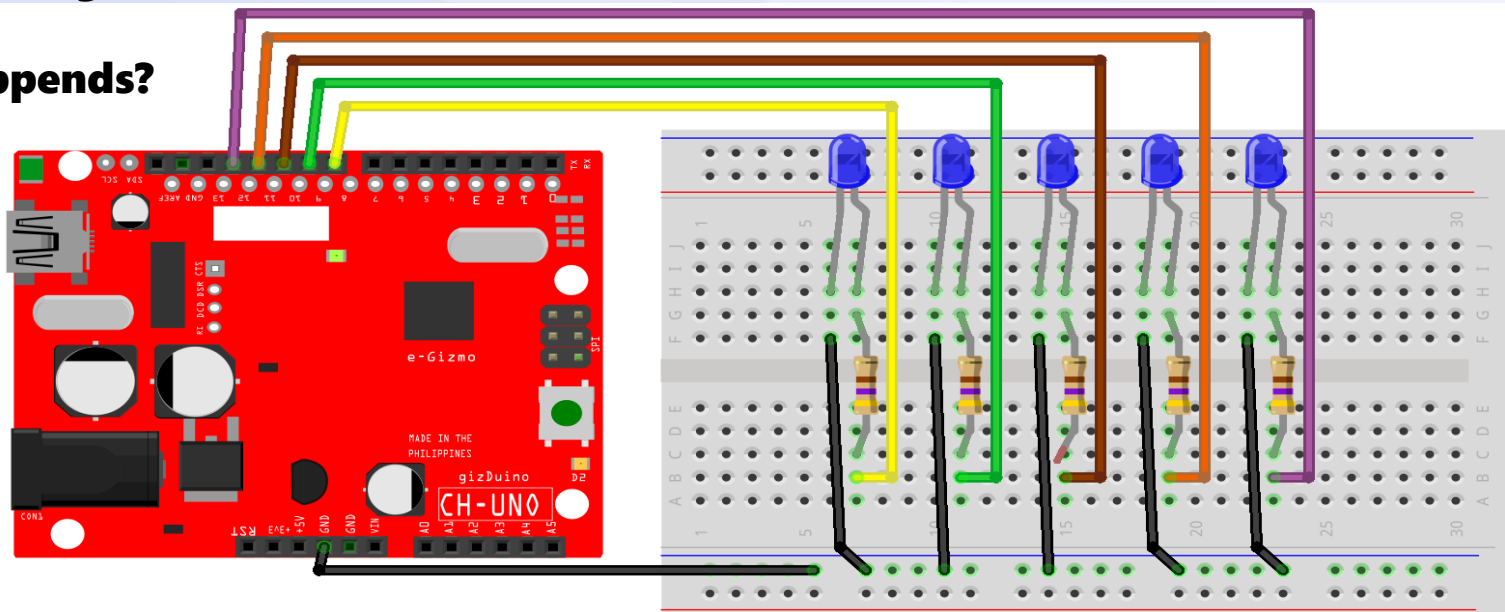
DigitalPin 8,9,10,11,12

- 1. Blink all the LEDs**
- 2. Do the Running Light**

Upload the code

- 1. MultipleLEDs2.ino**
- 2. RunningLEDLights2**

See what happens?



fritzing

The code (multiple use)

```
8 #define LED_1_PIN 8 // assign a variable name in D8
9 #define LED_2_PIN 9 // assign a variable name in D9
10 #define LED_3_PIN 10 // assign a variable name in D10
11 #define LED_4_PIN 11 // assign a variable name in D11
12 #define LED_5_PIN 12 // assign a variable name in D12
```

```
14 void setup() {
15
16   pinMode(LED_1_PIN, OUTPUT); // Set D8 as Output
17   pinMode(LED_2_PIN, OUTPUT); // Set D9 as Output
18   pinMode(LED_3_PIN, OUTPUT); // Set D10 as Output
19   pinMode(LED_4_PIN, OUTPUT); // Set D11 as Output
20   pinMode(LED_5_PIN, OUTPUT); // Set D12 as Output
21
22 }
```

#define constantName value

constantName: the name of the macro to define.

value: the value to assign to the macro.

pinMode(pin, mode);

Pin: arduino pin number to set mode of.

Mode: INPUT, OUTPUT or INPUT_PULLUP.

Note:

Always add the pinMode on each pin you will use either OUTPUT or INPUT.

The code

```
24 void loop() {  
25  
26   // Turn ON ALL LED  
27   digitalWrite(LED_1_PIN, HIGH);  
28   digitalWrite(LED_2_PIN, HIGH);  
29   digitalWrite(LED_3_PIN, HIGH);  
30   digitalWrite(LED_4_PIN, HIGH);  
31   digitalWrite(LED_5_PIN, HIGH);  
32   delay(1000);  
33   // Turn Off ALL LED  
34   digitalWrite(LED_1_PIN, LOW);  
35   digitalWrite(LED_2_PIN, LOW);  
36   digitalWrite(LED_3_PIN, LOW);  
37   digitalWrite(LED_4_PIN, LOW);  
38   digitalWrite(LED_5_PIN, LOW);  
39   delay(1000);  
40  
41 }
```

digitalWrite(pin,value);

Pin: arduino pin number to set mode of.

Value: HIGH or LOW

delay(ms);

ms: the number of milliseconds to pause.

1000ms = 1second

Note:

*If you are using Digital pin as Output use **digitalWrite**.*

The code (for loop)

```
8 int LED_NUMBER[] = {8, 9, 10, 11, 12};
9 int DEL1 = 100;
10 int DEL2 = 100;
```

```
12 void setup() {
13
14     for (int i = 0; i <= 4; i++) {
15         pinMode(LED_NUMBER[i], OUTPUT);
16     }
17
18 }
```

```
20 void loop() {
21
22     ASCEND();
23     delay(DEL1);
24     DESCEND();
25     delay(DEL1);
26
27 }
```

int var = val;

var: variable name

val: the value you assign to that variable.

for(initialization; condition; increment){
// statement(s);
}
|

Where:

Initialization: i = 0; //start

Condition: < (less than), > (greater than), <= (less than equal), >= (greater than equal)

Increment: ++, --, +=, -=

***Inside the loop, you can add
The Functions()/ statements***

The code (add more functions)

```
29 void ASCEND() {  
30   for (int i = 0; i <= 4; i++) {  
31     digitalWrite(LED_NUMBER[i], HIGH);  
32     delay(DEL2);  
33   }  
34   for (int i = 0; i <= 4; i++) {  
35     digitalWrite(LED_NUMBER[i], LOW);  
36     delay(DEL2);  
37   }  
38 }
```

```
39 void DESCEND() {  
40   for (int i = 4; i >= 0; i--) {  
41     digitalWrite(LED_NUMBER[i], HIGH);  
42     delay(DEL2);  
43   }  
44   for (int i = 4; i >= 0; i--) {  
45     digitalWrite(LED_NUMBER[i], LOW);  
46     delay(DEL2);  
47   }  
48 }
```

int var = val;

var: variable name

val: the value you assign to that variable.

for(initialization; condition; increment){
// statement(s);
}
|

Where:

Initialization: i = 0; //start

Condition: < (less than), > (greater than), <= (less than equal), >= (greater than equal)

Increment: ++, --, +=, -=

The 5 LED was assigned in 8,9,10,11,12 as 0,1,2,3,4 for loop.

Part 3. Latch and Push Button switch

Connections:

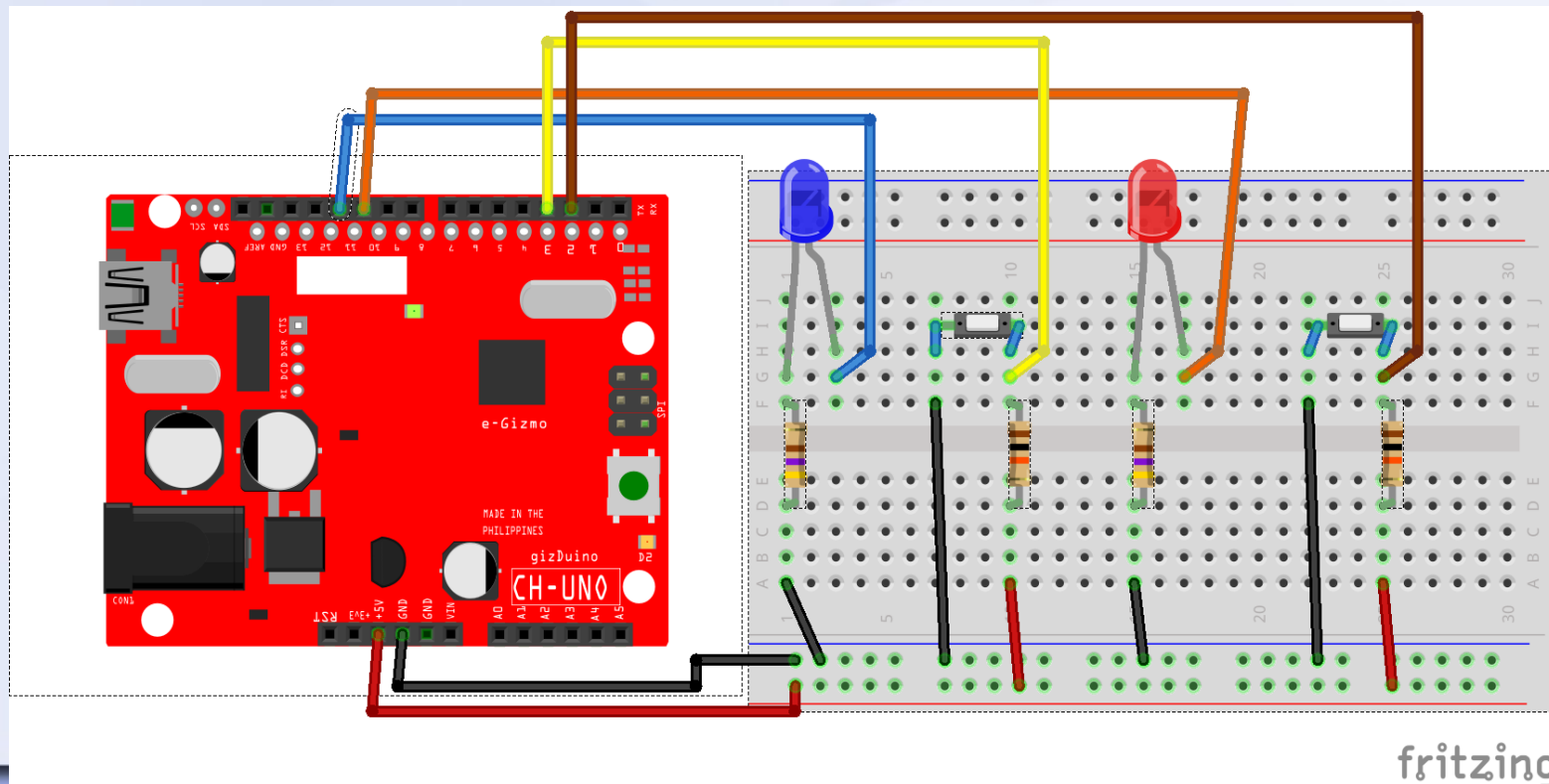
LED Blue → **D11**

LED Red → **D10**

Button 1: D3

Button 2: D2

Upload the Latch Switch Button.ino.



The code

```
8 int LED_PIN1 = 11;
9 int LED_PIN2 = 10;
10
11 int SWITCHPIN1 = 3;
12 int SWITCHPIN2 = 2;
13
14 int SWITCH_STATE1 = 0;
15 int SWITCH_STATE2 = 0;
16
17 int LED_STATE = 0;
```

int var = val;

var: variable name

val: the value you assign to that variable.

LEDs pin 11 and 10

Switches pin 3 and 2

The switch state is 0 or LOW

If it is pressed,

**But in normal situation the
Switch value is HIGH.**

**LED state is 0, for switch
case statement.**

**- controls the flow of
programs by allowing
programmers to specify
different code that executed
in various conditions.**

The code

```
19 void setup() {  
20  
21   pinMode(LED_PIN1, OUTPUT);  
22   pinMode(LED_PIN2, OUTPUT);  
23  
24   pinMode(SWITCHPIN1, INPUT);  
25   pinMode(SWITCHPIN2, INPUT);  
26  
27   digitalWrite(LED_PIN1, LOW);  
28 }
```

pinmode(pin,mode);

Pin: arduino pin number to set mode of.

Mode: INPUT,OUTPUT or INPUT_PULLUP.

```
30 void loop() {  
31  
32   SWITCH_STATE1 = digitalRead(SWITCHPIN1);  
33   SWITCH_STATE2 = digitalRead(SWITCHPIN2);  
34  
35   if (SWITCH_STATE1 == 0) {  
36     while (digitalRead(SWITCHPIN1) == 0);  
37     switch (LED_STATE) {  
38       case 0:  
39         digitalWrite(LED_PIN1, HIGH);  
40         LED_STATE = 1;  
41         break;  
42       case 1:  
43         digitalWrite(LED_PIN1, LOW);  
44         LED_STATE = 0;  
45         break;  
46     }  
47   }  
48   if (SWITCH_STATE2 == 0) {  
49     digitalWrite(LED_PIN2, HIGH);  
50   }  
51   if (SWITCH_STATE2 == 1) {  
52     digitalWrite(LED_PIN2, LOW);  
53   }  
54 }
```

Switch Case:

- controls the flow of programs by allowing programmers to specify different code that executed in various conditions.

while:

- while loop will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false.

If statement:

- checks for a condition and executes the following statement or set of statements if the condition is 'true'.

Part 4. Passive Buzzer

Connections

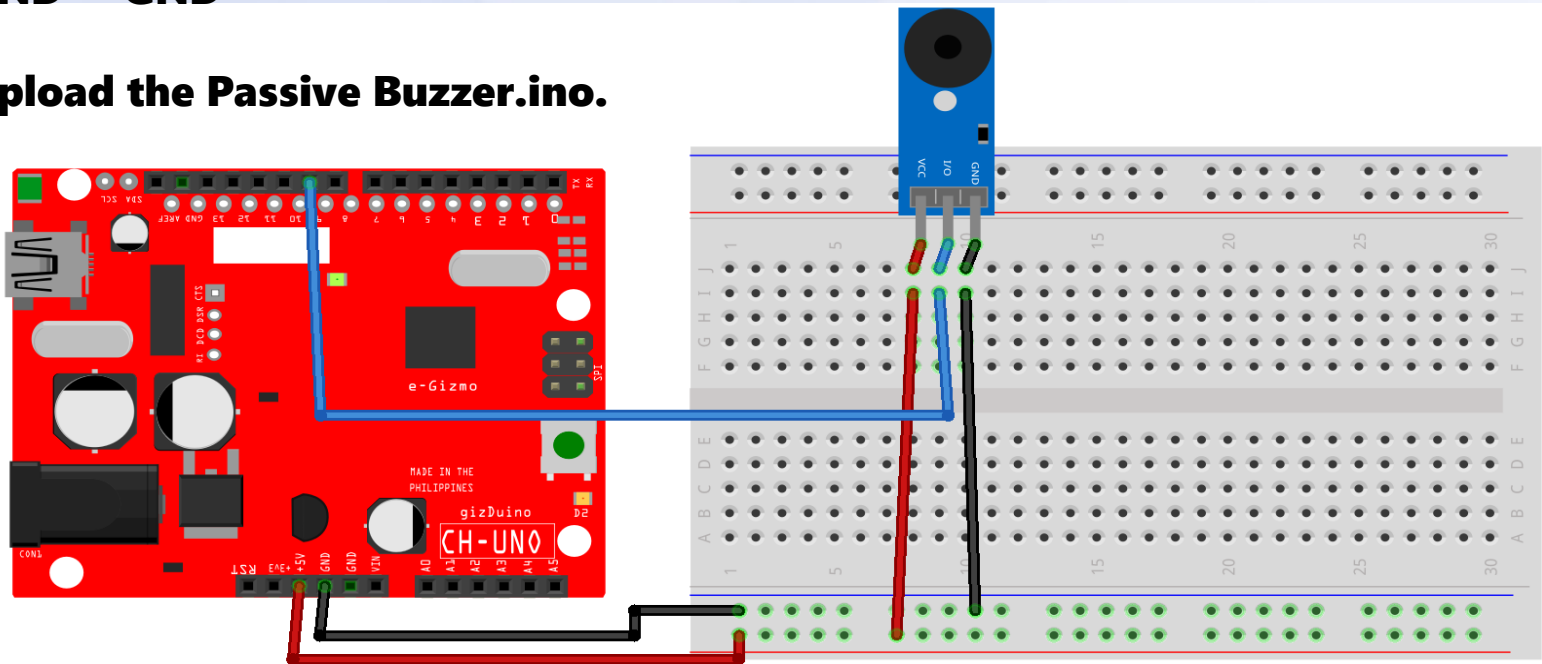
Passive Buzzer

VCC → **5V**

I/O → **D9**

GND → **GND**

Upload the Passive Buzzer.ino.



The code

```
6 #include "pitches.h"
7 #define BUZZER 9
```

```
26 tone(BUZZER, NOTE_B5);
27 delay(500);
28 tone(BUZZER, NOTE_A5);
29 delay(500);
30 tone(BUZZER, NOTE_G5);
31 delay(500);
```

#include "pitches.h"

- is used to include outside libraries in your sketch.

#define constantName value

constantName: the name of the macro to define.

value: the value to assign to the macro.

tone(pin, frequency)

tone(pin, frequency, duration)

frequency: the frequency of the tone in hertz

duration: the value to assign to the macro.

Part 5. Relay

Connections:

Relay → **gizDuino**

IN → **Digital pin 5**

VCC → **+5V**

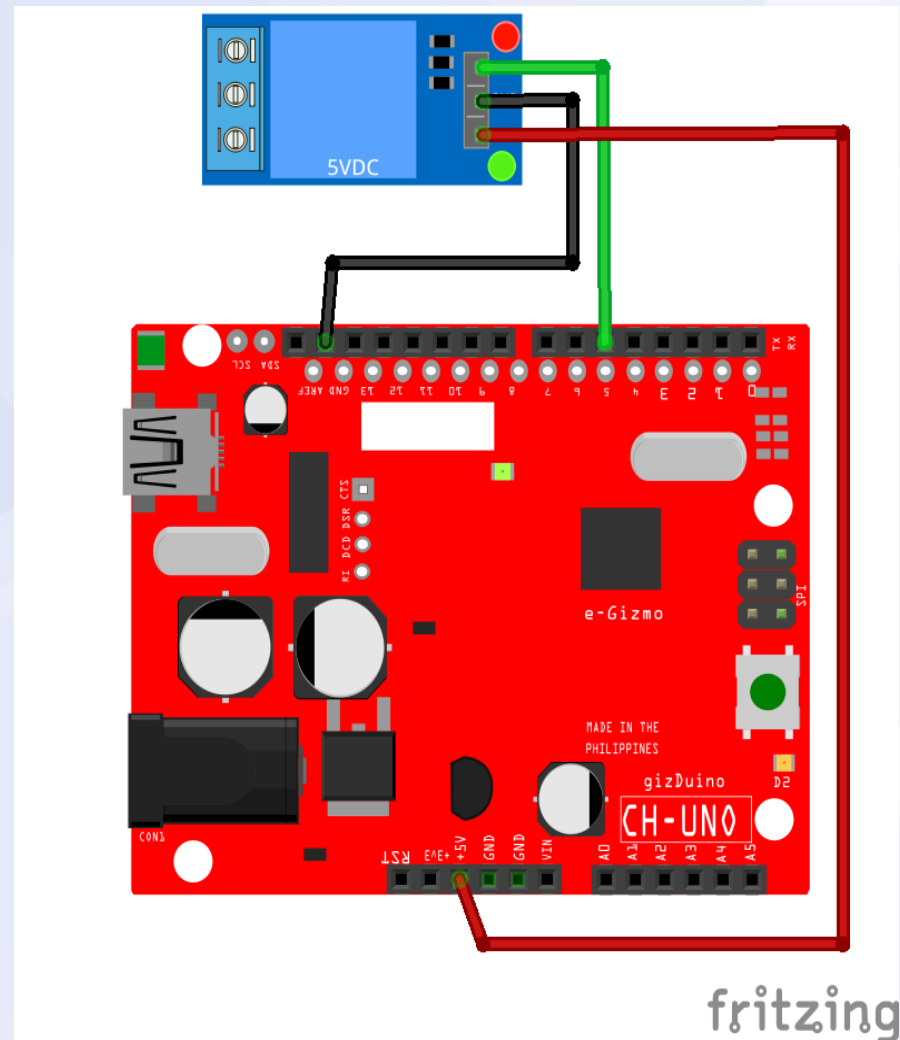
GND → **GND**

Upload the code RELAY.ino.

Its actually a blinking LED on Digital Pin 5. Once the Output Is LOW the relay will trigger its Contact.

See what happens?

Relay is Active-LOW.



The code

```
6 int RELAY_PIN = 5; // RE
7 int LED_INDICATOR = 4; /
```

```
9 void setup() {
10   // initialize the digital pin as an output.
11   pinMode(RELAY_PIN, OUTPUT);
12   pinMode(LED_INDICATOR, OUTPUT);
13
14
15 }
```

```
16 void loop() {
17   digitalWrite(LED_INDICATOR, LOW); // LE
18   digitalWrite(RELAY_PIN, HIGH); // turn
19   delay(1000); //delay in 1sec
20   digitalWrite(LED_INDICATOR, HIGH); //
21   digitalWrite(RELAY_PIN, LOW); // turn t
22   delay(1000);
23
24 }
```

Relay pin is on Digital Pin 5 LED indicator is Digital Pin 4

Check/test the relay, if it is Active-Low when you put 0/GND to the Input pin, it will trigger the relay contacts from Normally Open to Closed. Vice versa, with Active-High.

On this sample code, every 1 sec the pin 5 is blinking. If you see the LED indicator ON the relay triggered. OFF if not.

Part 6. Temperature and Humidity Sensor

Connections:

DHT11 → **gizDuino**

VCC → **+5V**

OUT → **Digital pin 12**

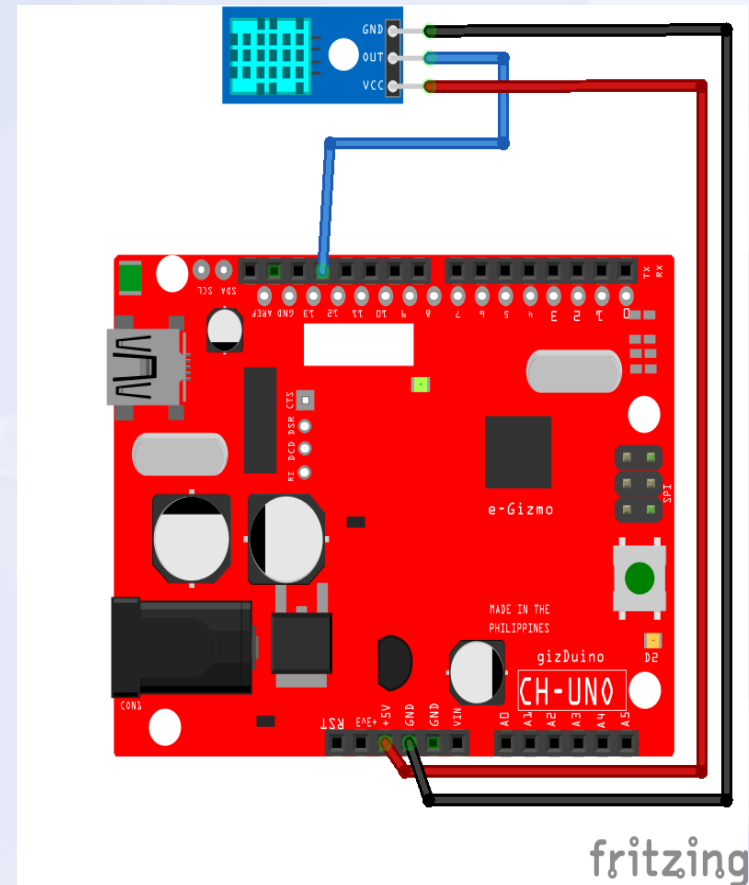
GND → **GND**

**First, read the nextslide on how to add
A DHT11 library.**

Upload the DHT11breakout_sample_sketch.

**Check the temperature and humidity
Value on the Serial Monitor.**

See what's the results?



How to add a DHT Library

There are two library folder to choose from and here's the path. (Choose Only 1)

1. My Documents > Arduino > libraries > DHT

2. Arduino 1.8.13 > libraries > DHT

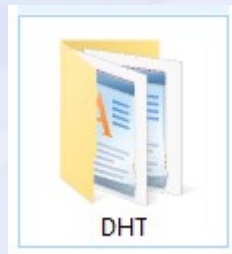
Libraries is available in our site.

When you download a library,
The file is compressed in zip
file.

So, Extract the file.

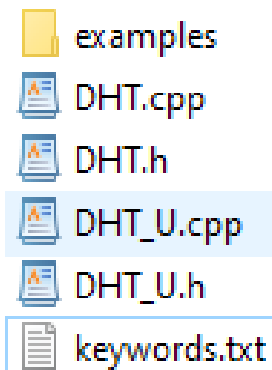
Get the DHT folder which
contains :

- **examples** > .ino file
- **DHT.cpp**
- **DHT.h**
- Keywords.txt



Note:

The *filename* and its file inside
must be the **same name** in *cpp*
and *h* files.



The code

```
16 #include "DHT.h"
17
18 #define DHTPIN 12
```

```
43 void setup() {
44     Serial.begin(9600);
45     Serial.println("DHTXX TEST!");
46
47     dht.begin();
48 }
```

dht.begin();

- the included library has need to
Set the begin function inside the setup.

#include "DHT.h"

- is used to include outside libraries in
your sketch.

Next to pin assigning DHTPIN is on
Digital Pin 12

In setup section:

Serial.begin(9600);

Serial.begin(speed)

Serial.begin(speed, config)

- sets the data rate in bits per seconds
(baud) for serial data transmission. For
communicating with Serial Monitor.

Serial.println("DHTXX TEST!");

Serial.println(val)

Serial.println(val, format)

- prints data to the serial port as human-
readable ASCII text followed by a
carriage return character (ASCII 13, or '\r')
and a newline character (ASCII 10, '\n').

The code

```
50 void loop() {  
51   // Wait a few seconds between measurements.  
52   delay(2000);  
53  
54   // Reading temperature or humidity takes about 250 millise  
55   // Sensor readings may also be up to 2 seconds 'old' (its  
56   float h = dht.readHumidity();  
57   // Read temperature as Celsius  
58   float t = dht.readTemperature();  
59   // Read temperature as Fahrenheit  
60   float f = dht.readTemperature(true);  
61  
62   // Check if any reads failed and exit early (to try again)  
63   if (isnan(h) || isnan(t) || isnan(f)) {  
64     Serial.println("FAILED TO READ FROM DHT SENSOR!");  
65     return;  
66   }  
67  
68   Serial.print("HUMIDITY: ");  
69   Serial.print(h);  
70   Serial.print(" %\t");  
71   Serial.print("TEMPERATURE: ");  
72   Serial.print(t);  
73   Serial.print(" *C ");  
74   Serial.print(f);  
75   Serial.println(" *F\t");  
76  
77 }  
78
```

In loop section:

float var = val;

var: variable name

val: the value you assign to that variable.
- datatype for floating-point numbers, a number that has a decimal point.

if (condition){
//statements(s)
}

Condition: a boolean expression (true or false)

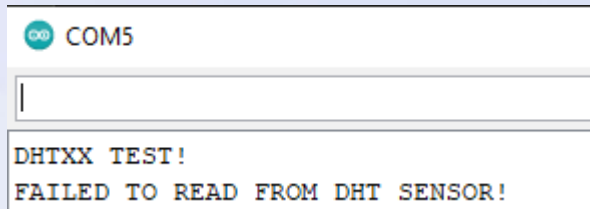
return;

- terminate a function and return a value from a function to the calling function, if desired.

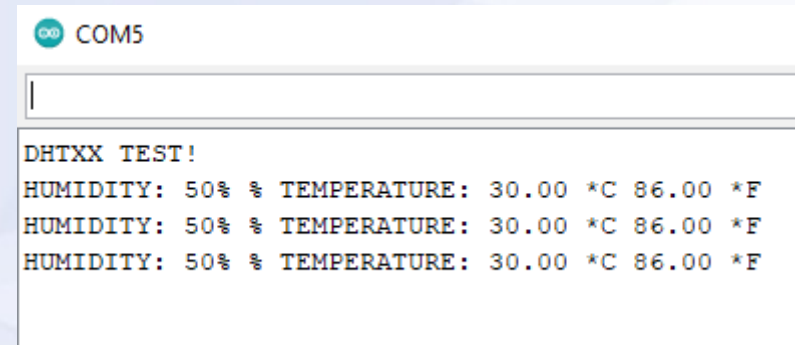
Serial.println(); - refer to previous slide.

Serial Monitor

Open the Serial Monitor by clicking the icon on the right top corner in Arduino Ide.



If the sensor is not properly connected it will display failed to read from dht sensor.



This should be the correct output displayed on the Serial Monitor.

Part 7. Distance Sensor

Distance Sensor → gizDuino

Vcc → +5V

Trigger → Digital Pin 11

Echo → Digital Pin 12

Gnd → GND

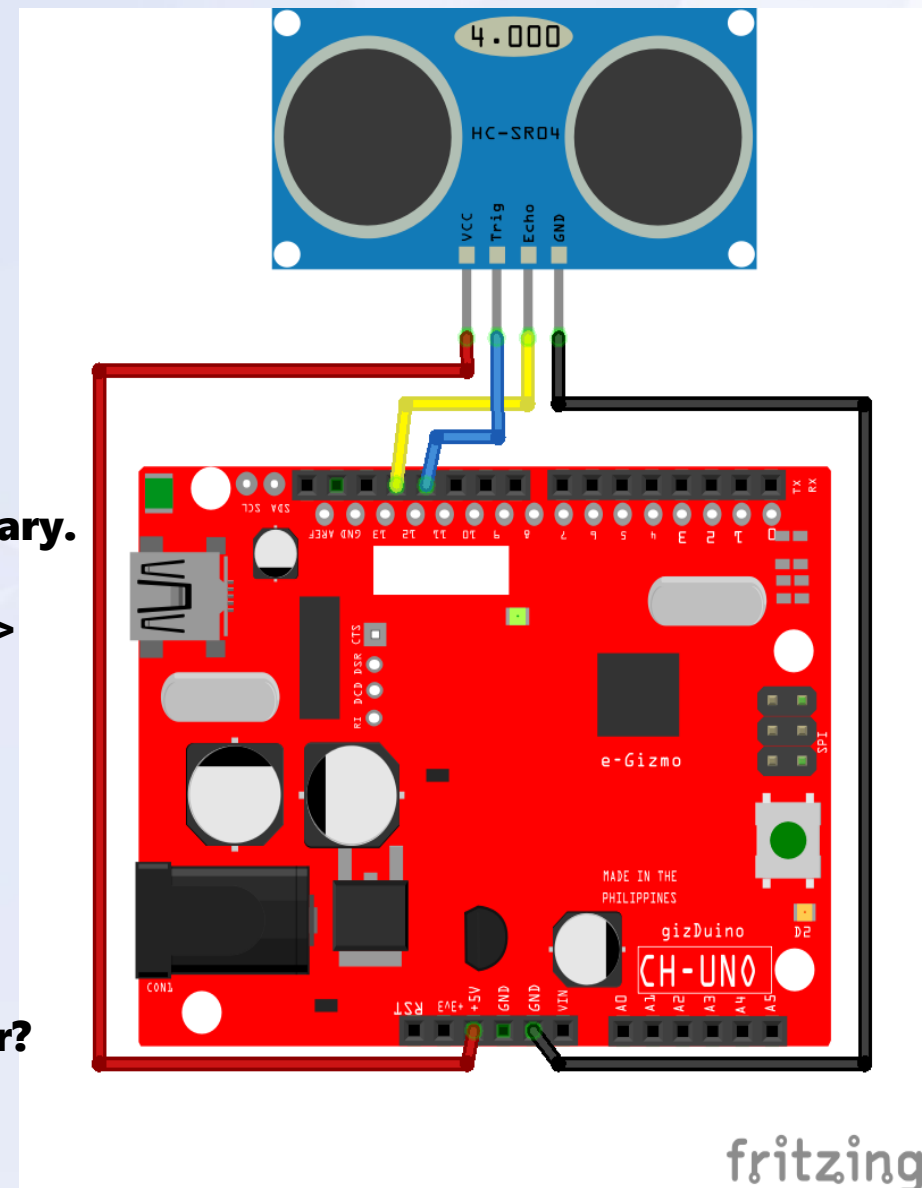
Make sure you add the NewPing library.

**File>Examples>NewPing>examples>
NewPingExample.**

Open the Serial Monitor.

Check the value.

**See what happens, if you block
Something on the front of the sensor?**



The code

```
1 #include <NewPing.h>
```

```
3 #define TRIGGER_PIN 12  
4 #define ECHO_PIN 11  
5 #define MAX_DISTANCE 200
```

#include <NewPing.h>

- is used to include outside libraries in your sketch.

Trigger pin is Digital Pin 12

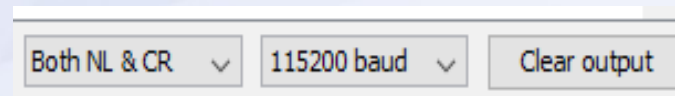
Echo pin is Digital Pin 11

Maximum Distance is 200 (cm) unit

The code

```
9 void setup() {  
10   Serial.begin(115200); // Open serial  
11 }  
12  
13 void loop() {  
14   delay(50); // V  
15   unsigned int uS = sonar.ping(); // S  
16   Serial.print("Ping: ");  
17   Serial.print(uS / US_ROUNDTRIP_CM);  
18   Serial.println("cm");  
19 }
```

Serial.begin sets to 115200 baudrate.
When you open the serial monitor
Go to this section select the correct
baudrate **115200** bps

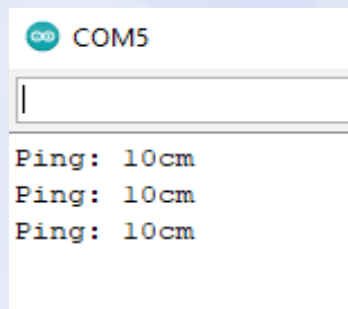


Both NL & CR ▾ 115200 baud ▾ Clear output

unsigned int var = val;

var: variable name

val: the value you assign to that variable



COM5

Ping: 10cm
Ping: 10cm
Ping: 10cm

Serial Monitor display

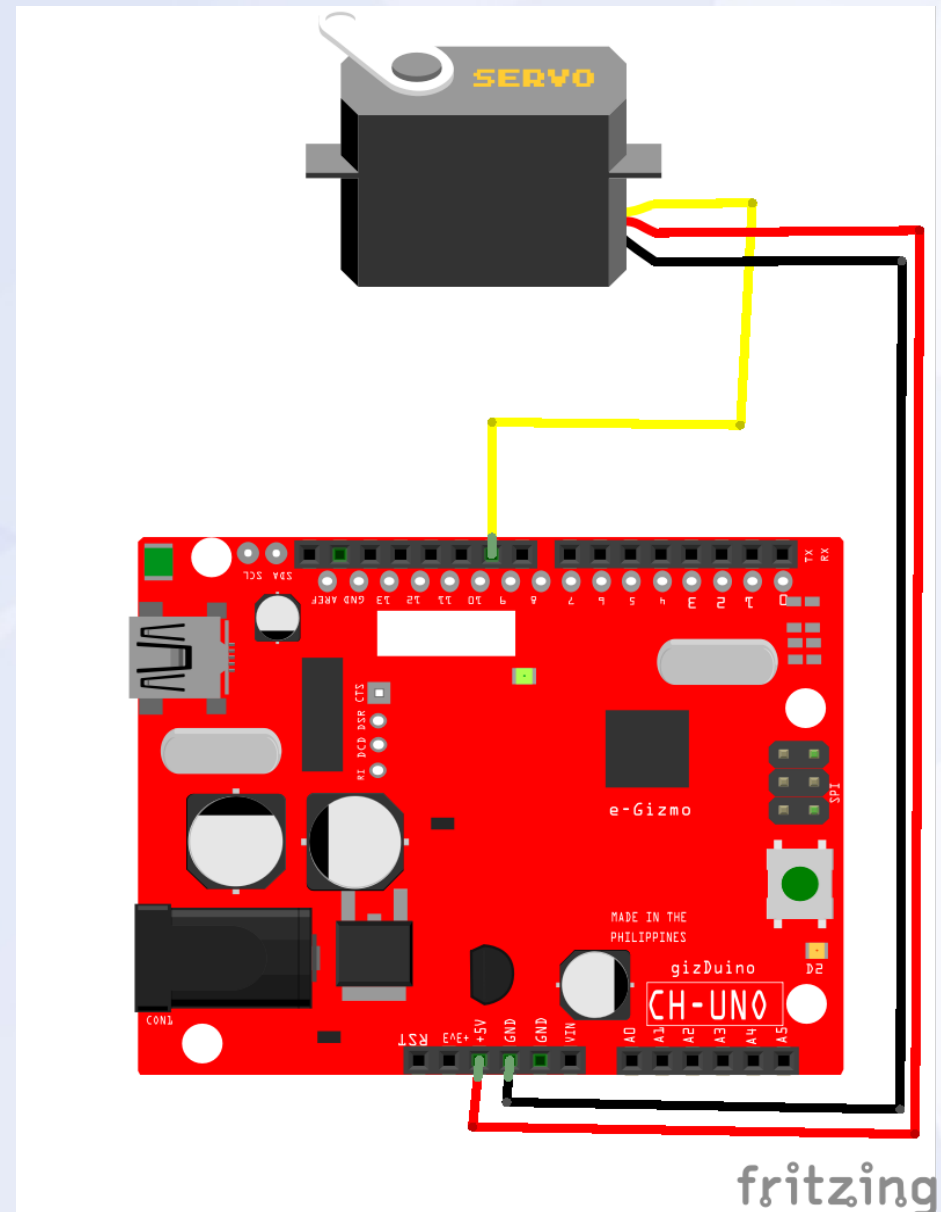
Part 8. Servo Motor

Servo → **gizDuino**
SIGNAL (Yellow/orange) → **Digital Pin 9**
GND(Black/brown) → **GND**
VCC(Red) → **+5V**

Upload the ServoSweep.ino

Question? Why do we need a Servo Motor?

See what happens?



The code

```
10 #include <Servo.h>
11
12 Servo myservo; // create servo object to control a servo
13 // twelve servo objects can be created on most boards
14
15 int pos = 0;    // variable to store the servo position
```

```
17 void setup() {
18     myservo.attach(9);
19 }
```

#include <Servo.h>

- is used to include outside libraries in your sketch.

Servo object name;

- name it your servo

int pos = 0;

- we create variable with start to 0 degree position.

servo.attach(pin);

servo.attach(pin,min,max);

- attach the servo variable to a pin.

The code

```
21 void loop() {  
22   for (pos = 0; pos <= 180; pos += 1) { //  
23     // in steps of 1 degree  
24     myservo.write(pos);                // to  
25     delay(15);                          // w  
26   }  
27   for (pos = 180; pos >= 0; pos -= 1) { //  
28     myservo.write(pos);                // to  
29     delay(15);                          // w  
30   }  
31 }
```

**for(initialization; condition;
increment){
// statement(s);
}
|**

Where:

Initialization: pos = 0; //start

Condition: <(less than), >(greater than), <=(less than equal), >=(greater than equal)

Increment: ++, --, +=, -=

servo.write(angle);

angle: the value to write to the servo, from 0 to 180 degrees.

Part 9. Servo Motor

Motor Driver → gizDuino

IN1 → Digital Pin 8

IN2 → Digital Pin 9

+VM → +5V

GND → GND

DC Motor → Motor Driver

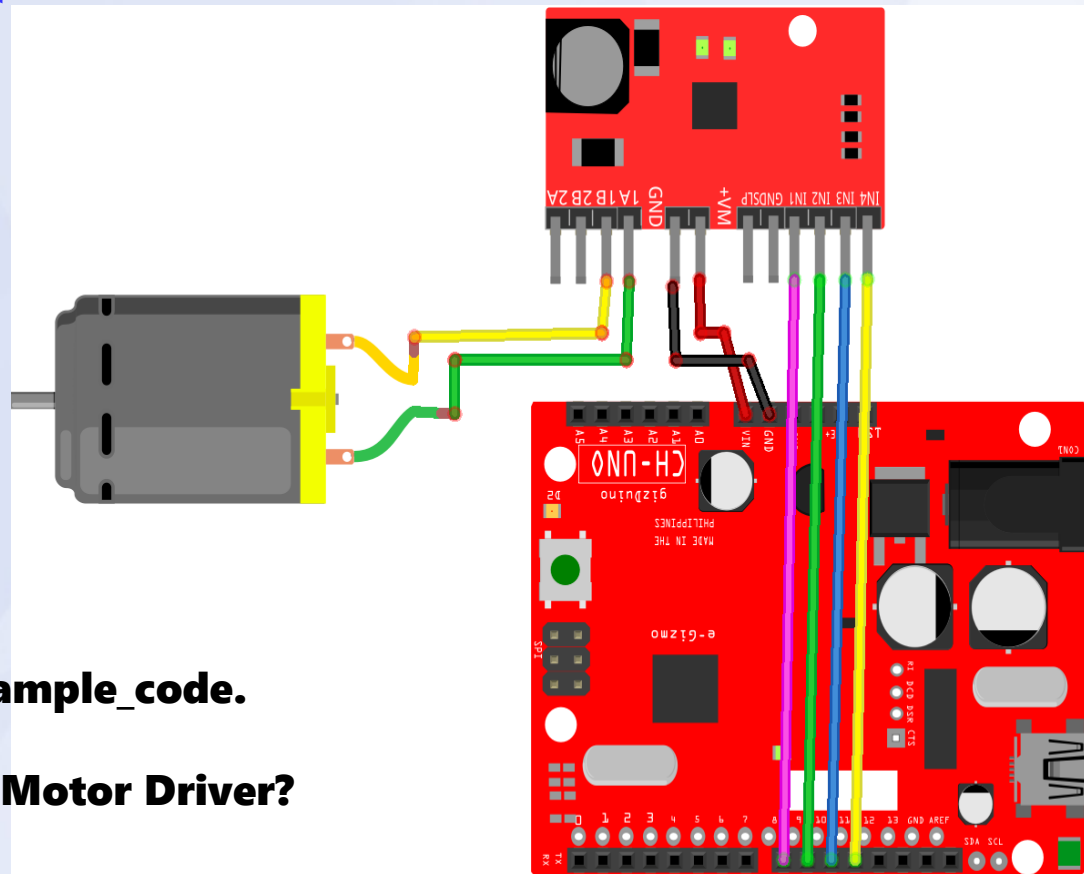
Pin1 → 1A

Pin2 → 1B

Upload the 2ch_tiny_motor_sample_code.

Question? Why do we need a Motor Driver?

See what happens?



The code

```
58 void setup()
59 {
60   pinMode(8, OUTPUT); //M1DIR
61   pinMode(9, OUTPUT); //M1RUN
62   pinMode(10, OUTPUT); //M2RUN
63   pinMode(11, OUTPUT); //M2DIR
64 }
```

```
65 void loop()
66 {
67   //Forward
68   digitalWrite(8, 0);
69   analogWrite(9, 128);
70   analogWrite(10, 128);
71   digitalWrite(11, 0);
72   delay(1000);
```

In gizDuino LIN-UNO the PWM pins are Digital Pin 3,5,6,9,10,11.

9 and 10 are PWM pins for controlling the speed from 0-255. 0 means stop, 255 full-speed

While 8 and 11 are assigned for switching the directions of the motor.

On this example, used digitalWrite for the direction and analogWrite for the speed.



e-Gizmo
MECHATRONIX CENTRAL

Part 10. LCD Display with I2C module

LCD w/ I2C → gizDuino

Gnd → GND

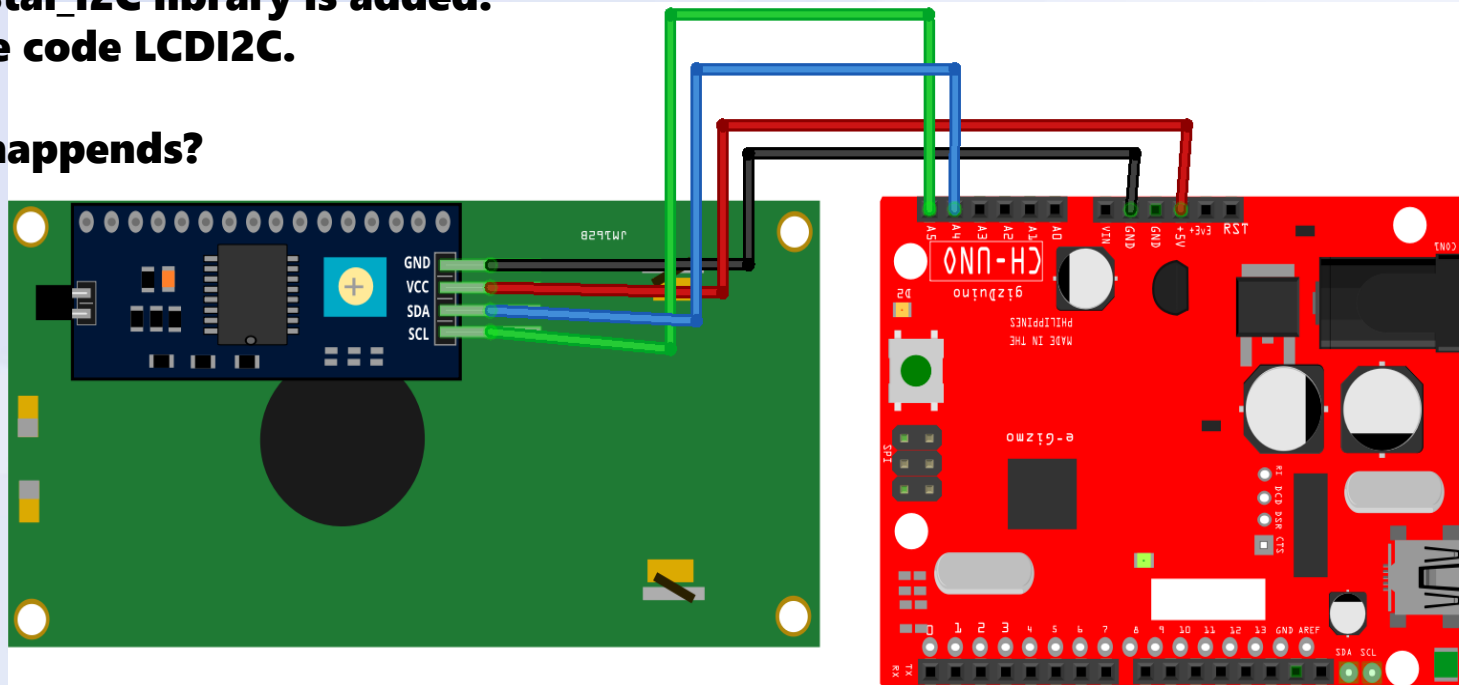
Vcc → +5V

SDA → A4/Digital Pin 18/SDA

SCL → A5/Digital Pin 19/SCL

**Make sure the
LiquidCrystal_I2C library is added.
Upload the code LCDI2C.**

See what happens?



fritzing

The code

```
6 #include <Wire.h>
7 #include <LiquidCrystal_I2C.h>
```

Wire and LiquidCrystal_I2C added.

```
13 LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
```

This is with I2C module so the
i2c address is **0x27**

```
15 void setup() {
16     Serial.begin(9600);
17
18     lcd.begin(16,2);
19 }
```

Set the baudrate to 9600 bps.

lcd.begin(16,2);

16: number of columns

2: number of rows

With the 32 characters display in total.

The code

```
20  for(int i = 0; i< 3; i++)
21  {
22      lcd.backlight();
23      delay(250);
24      lcd.noBacklight();
25      delay(250);
26  }
```

LCD backlight blinks 4 times every 250ms delay

```
27  lcd.backlight();
28
29  lcd.setCursor(0,0); //Start at
30  lcd.print("Hello, world!");
31  delay(1000);
32  lcd.setCursor(0,1);
33  lcd.print("I2C Module Disp");
34  delay(8000);
```

lcd.backlight();

- it stick to backlight on.

lcd.setCursor(0,0);

- set the cursor on the origin (0,0), column, row.

lcd.print("Hello, world!");

- it displays on the LCD.

```
36  lcd.clear();
37  lcd.setCursor(0,0); //Start at
38  lcd.print("Use Serial Mon");
39  lcd.setCursor(0,1);
40  lcd.print("Type to display");
41
```

lcd.clear();

- clear out/erase all the display.

The code

```
46 void loop()  
47 {  
48   {  
49     if (Serial.available()) {  
50       delay(100);  
51       lcd.clear();  
52       while (Serial.available() > 0) {  
53         lcd.write(Serial.read());  
54       }  
55     }  
56   }  
57  
58 }
```

Serial.available()

- get the number of bytes (characters) available for reading from the serial port.

lcd.write(Serial.read());

- writes binary data to the serial port.
- it prints the read data to lcd display.

Serial.read();

- reads incoming serial data



For more info:

- Website: www.e-gizmo.net
- Egizmo Tech blog: <https://www.e-gizmo.net/oc/index.php?route=journal3/blog>
- Facebook: eGizmoMechatronics
- Youtube Channel: e-Gizmo Mechatronics Central
- Shopee: eGizmo