

Directory

Preface – Getting Started.....	3
Part 1 Modules Introduction.....	5
1. Crowtail – Motor Base Introduction.....	5
2. Crowtail- LED.....	5
3. Crowtail- Button.....	6
4. Crowtail- Switch.....	6
5. Crowtail- Buzzer.....	7
6. Crowtail- Hall Sensor.....	7
7. Crowtail- Vibration Sensor.....	8
8. Crowtail- One Wire Waterproof Temperature Sensor.....	8
9. Crowtail- RGB-LED.....	9
10. Crowtail- Rotary Angle Sensor.....	9
11. Crowtai- RTC.....	9
12. Crowtail- Solid-State Relay.....	10
13. DC Toy / Hobby Motor.....	10
14. Crowtail- Ultrasonic Ranging Sensor.....	10
15. Crowtail- IIC EEPROM.....	11
16. Crowtail- IR Emitter.....	11
17. Crowtail- IR Receiver.....	12
18. Crowtail- 9G Servo.....	13
19. Crowtail- I2C LCD.....	13
20. Crowtail- 3-Axis Digital Accelerometer($\pm 16g$).....	13
21. Crowtail- 3-Axis Digital Gyro.....	14
22. Crowtail- 3-Axis Digital Compass.....	14
23. Crowtail- A6 GPRS/GSM Module.....	15
24. Crowtail- ESP8266 nodeMCU.....	16
Part2: Crowtail Applications.....	16
Lesson1: LED Control.....	16
Lesson2: Hall Effect Switch.....	22
Lesson3: PWM Control LED.....	23

Lesson4: Colorful RGB LED.....	24
Lesson5: Vibration Alarm.....	26
Lesson6: Make an electric watch.....	27
Lesson7: Temperature display system.....	28
Lesson8: Ultrasonic ranging.....	30
Lesson9: Motor Control.....	32
Lesson10: Servo control.....	33
Lesson11: MINI Fan.....	34
Lesson12: IR Control.....	36
Lesson13: IIC EEPROM.....	39
Lesson14: 3-Axis Digital Accelerometer.....	40
Lesson15: 3-Axis Digital Gyro.....	41
Lesson16: 3-Axis Digital Compass.....	43
Lesson17: Send data to your Phone.....	47
Lesson18: ESP8266 Web Server.....	48



ELECROW

Preface – Getting Started

Welcome to the world of Crowtail! Crowtail is a modulated, ready-to-use toolset, it takes a building block approach to assembling electronics. It simplifies and condenses the learning process significantly.

The Crowtail products are basic-functional modules that *consist* of a Base Shield and various modules with standardized connectors, each Crowtail module has its specific functions, such as light sensing and temperature sensing. With these Crowtail modules, users do not need to deal with the mess jumper wires or debug the electronic circuits, they can just plug the Crowtail modules to the base shield and then play!

Before we discuss those Crowtail modules one by one, you need to seat yourself and finish some preparations.

1. What's Arduino?

Arduino is a flexible and easy-to-learn open source development platform that enjoys great fame among makers, geeks and interactive artists. We have released two versions of kit- The Elecrow Advanced Kit and Elecrow Starter Kit. They are contain the most popular and basic accessories for DIY projects, such as LED, Button, PIR sensor, Flame sensor, serial WIFI, 4-Digit Display, IIC LCD, etc. But if you want to do something more creative and difficult, recommend this Crowtail Deluxe kit to you. It has lots of powerful module like Crowtail- ESP8266 nodeMCU, Crowtail- A6 GPRS/GSM Module, Crowtail- 3-Axis Digital Gyro and so on. You can climb more higher than before!

2. Arduino IDE Installation

Arduino is also the name of a programming IDE based on C/C++. After you get your Arduino, you should install the IDE. Depending on OS version, the specific installation varies. Thankfully Arduino team provides us a detailed installation guide for most OS systems:

<http://arduino.cc/en/Guide/HomePage>

Download

Arduino 1.0.5 ([release notes](#)), hosted by [Google Code](#):

- + [Windows Installer, Windows \(ZIP file\)](#)
- + [Mac OS X](#)
- + [Linux: 32 bit, 64 bit](#)
- + [source](#)

Next steps

- [Getting Started](#)
- [Reference](#)
- [Environment](#)
- [Examples](#)
- [Foundations](#)
- [FAQ](#)

Shenzhen Elecrow Tech. Deve. Co., Ltd.

Website: www.elecrow.com

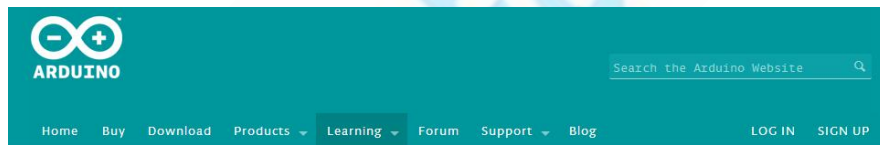
Then, connect your Arduino board to PC via USB. Install the Driver and the computer will recognize the Arduino board as a COM port:



3. Language Reference

Arduino team also provides a good and comprehensive website for you to learn:

<http://arduino.cc/en/Reference/HomePage>



Reference [Language](#) | [Libraries](#) | [Comparison](#) | [Changes](#)

Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure

- setup()
- loop()

Control Structures

- if
- if...else

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT | INPUT_PULLUP
- LED_BUILTIN
- true | false
- integer constants

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

Part 1 Modules Introduction

1. Crowtail – Motor Base Introduction

The Crowtail- Motor Base Shield integrate the Crowtail-base shield and motor & stepper shield. It will bring convenience to your work with Arduino. It regulate the IOs of Arduino to the standard Crowtail interface, which can be sorted into 4 kinds: Analog (A), Digital (D), UART (U) and IIC (I):



6 Digital I/O ports (D2~D7) that have a mark “D”. These ports can be used to read and control digital Crowtail modules (Crowtail modules that have a mark “D”), such as the Button and LEDs. Some of the digital I/O ports can also be used as PWM (pulse width modulation) outputs;

3 Analog ports (A0~A2) that have a mark of “A”. Besides the functional of digital, these A ports can read the analog signal, such as a potentiometer or light sensor;

1 Hardware Serial ports (D0 D1) and 1 Software Serial port (D2 D3). There can be used for UART communication such as the WIFI module or Bluetooth module;

1 IIC ports (D4 D5). The IIC interfaces are for the IIC Communication such as communication with IIC LCD;

2 Motor output interface. There can control two motor at the same time.

1 External Power input port. Could be 6~22V depending on the motor you used.

1 Power switch. The switch helps to power on/off the Motor Base Shield.

2. Crowtail- LED



The Crowtail-LED is a simple LED indicator with resistor. The LED would be on when activated by logic HIGH, and off by logic LOW. It is the most common used indicator for human interfacing, and the best way for users stepping into the Arduino.

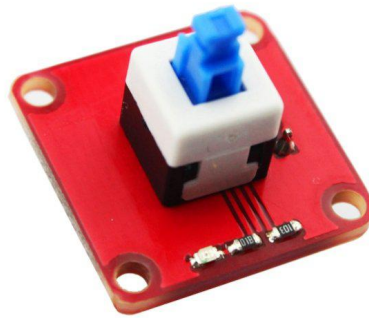
There are 3 LED modules in the Crowtail- Starter Kit, with Red/Yellow/Green color.

3. Crowtail- Button



The Crowtail-Button is a momentary push button which rebounds on its own position after released. The button outputs a logic HIGH signal when pressed, and logic LOW when released.

4. Crowtail- Switch



The Crowtail- Switch is a self-lock button. The switch outputs a logic HIGH signal when pressed.

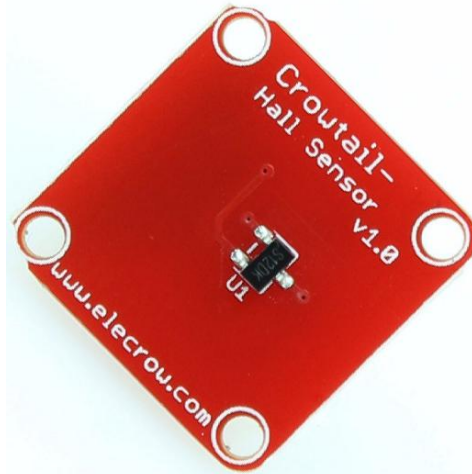
5. Crowtail- Buzzer



The Crowtail- Buzzer module is for making sound in your project. It sounds when activated by a logic HIGH signal. Connect the buzzer to any of the D (digital) ports of Crowtail- Base Shield, you can easily make it sounds with setting the related ports to logic HIGH.

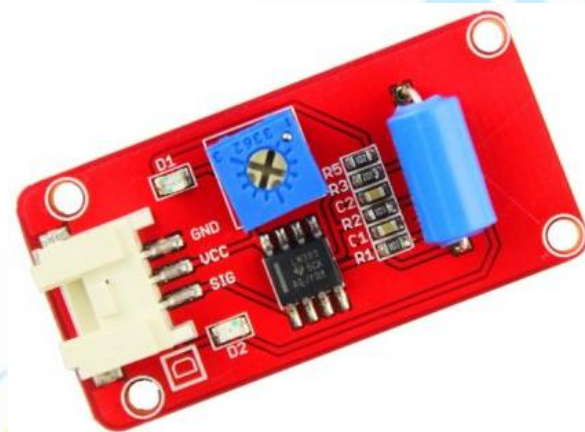
The buzzer module can be also connected to an pulse-width modulation (PWM) output to generate various of tones.

6. Crowtail- Hall Sensor



The Crowtail- Hall Sensor uses the Allegro™ A1101 Hall-effect switches. It measures the Hall Effect, which is a production of a voltage difference across an electrical conductor, transverse to an electric current in the conductor as well as a magnetic field perpendicular to the current.

Crowtail- Vibration Sensor



This Vibration sensor is based on the high precision vibration sensor SW-420. When have the vibration, the sensor will output the pulse. It is widely used to reported the theft alarm, intelligent car, earthquake alarm, motorcycle alarm, etc

7. Crowtail- One Wire Waterproof Temperature Sensor

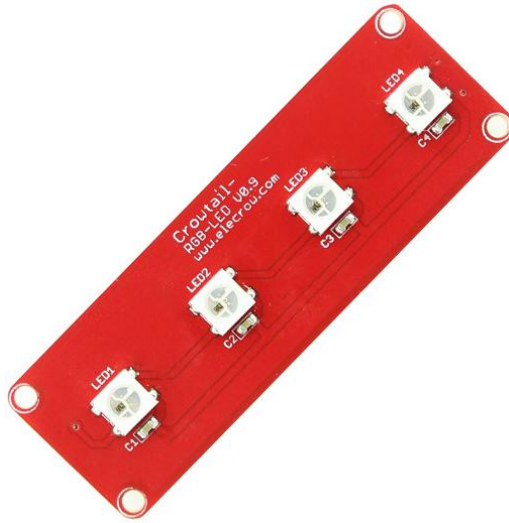


Shenzhen Elecrow Tech. Deve. Co., Ltd.

Website: www.elecrow.com

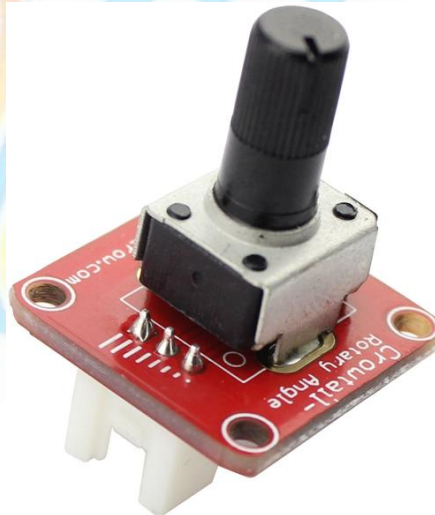
This One Wire Temperature Sensor has a waterproof probe and long wire shape, suitable for immersive temperature detection. The chip inside this sensor is the widely adopted DS18B20.

8. Crowtail- RGB-LED



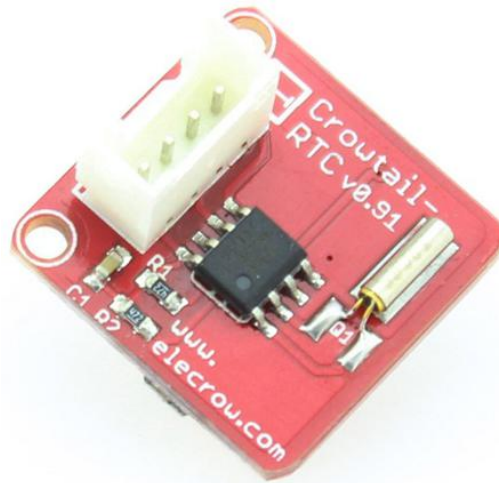
The Crowtail- RGB-LED module with 4 PCS WS2812B which is a Chainable & Addressable LED. You can control all the LED with only one microcontroller pin! Besides, the LED bar can be also chainable, that is, you can connect more than one LED bar together to make your project more dreamful.

9. Crowtail- Rotary Angle Sensor



This rotary angle sensor may also be known as potentiometer twig that produces analog output between 0 and VCC on its SIG pin. The angular range is 300 degrees with a linear change in value.

10. Crowtai- RTC



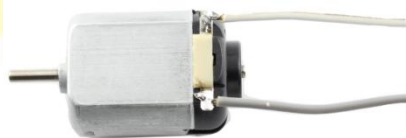
The Crowtail- RTC is based on the clock chip DS1307 which communicates with microcontroller with I2C protocol.

11. Crowtail- Solid-State Relay



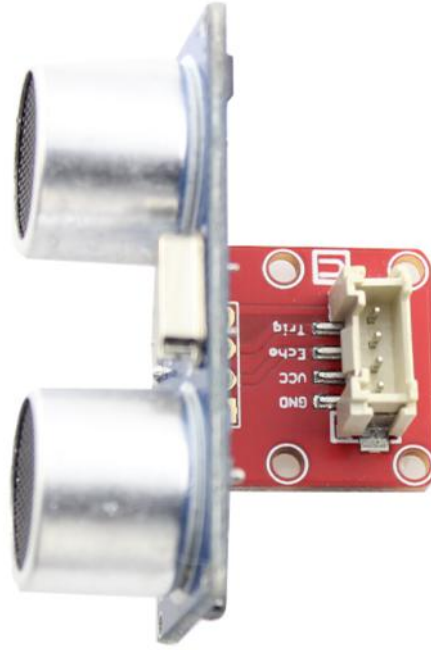
This Solid State Relay Module is based on Omron G3MB, to control current up to 2A@240VAC.

12. DC Toy / Hobby Motor



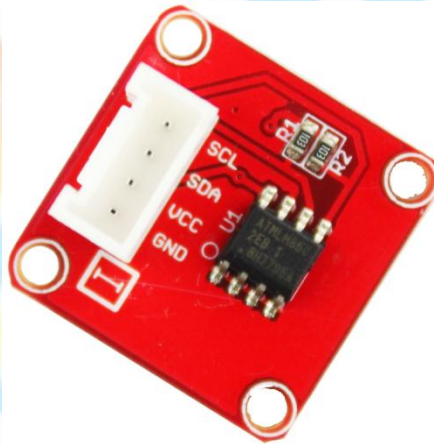
This is a standard '130 size' DC motor. It comes with a wider operating voltage range than most toy motors: from 4.5V to 9V DC.

13. Crowtail- Ultrasonic Ranging Sensor



This HC-SR04 has stable performance and high ranging accuracy. It used for detecting the obstacle and distance.

14. Crowtail- IIC EEPROM



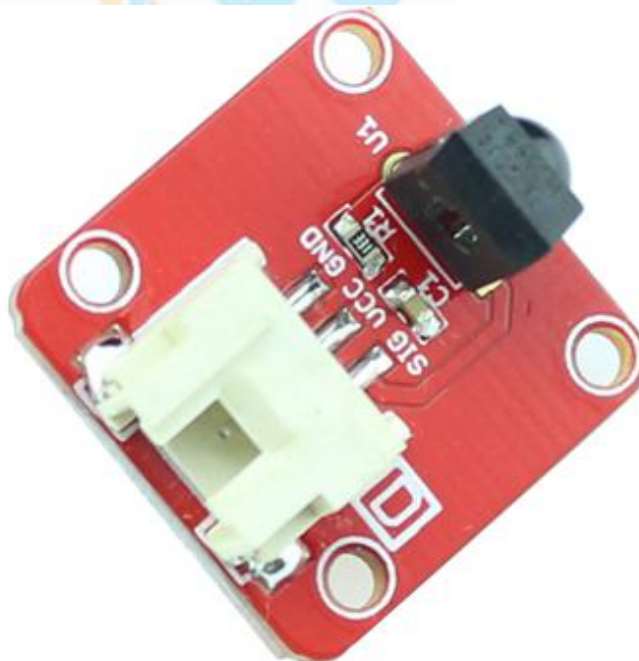
This module is based on the EEPROM chip AT24C256, which has 256k bit capacity. It communicate with Arduino with I2C bus, helps you do much more data storage easily.

15. Crowtail- IR Emitter



The Infrared Emitter is an LED made from gallium arsenide, with its color centered around 940nm. It's used to transmit infrared signals through an infrared LED, while there is an Infrared receiver to get the signals on the other side.

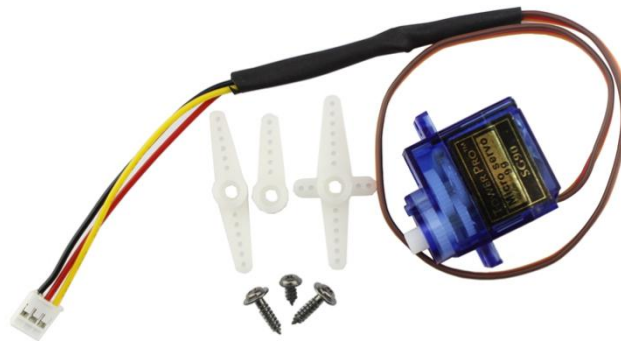
16. Crowtail- IR Receiver



The Crowtail- IR Receiver module uses the HS0038B which is miniaturized receivers for infrared remote control systems and it is the standard IR remote control receiver series,

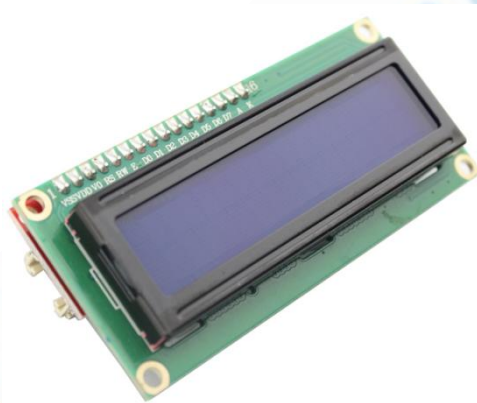
supporting all major transmission codes.

17. Crowtail- 9G Servo



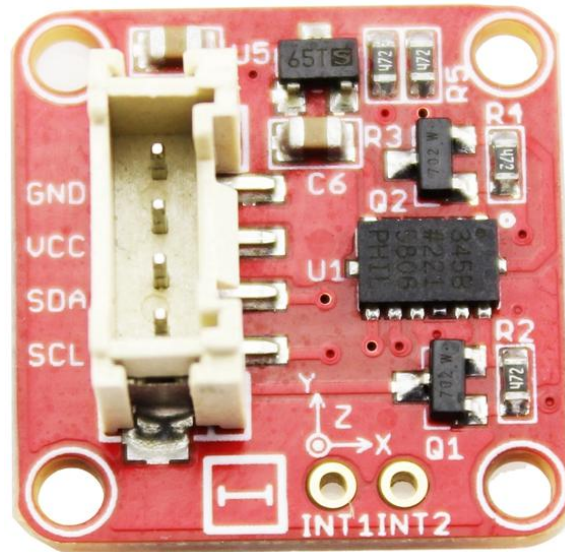
Tower Pro SG90 is a high quality, low-cost servo for all your mechatronic needs. It comes with a 3-pin power and control cable.

18. Crowtail- I2C LCD



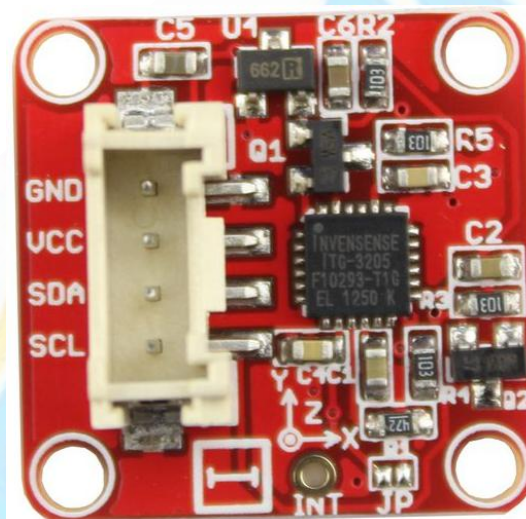
The Crowtail I2C LCD can display a max of 16x2 characters. With the help of the I2C bus convertor and related library, you can easily use this module with IIC interface.

19. Crowtail- 3-Axis Digital Accelerometer($\pm 16g$)



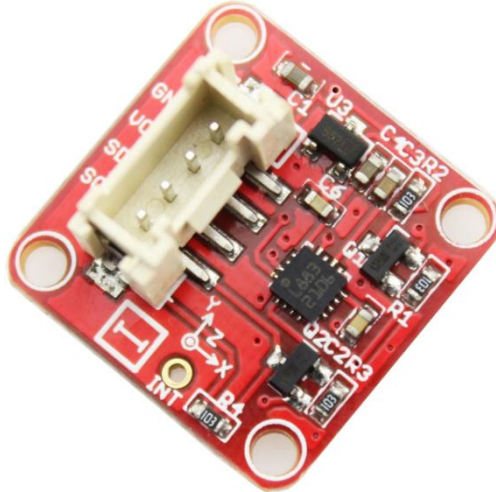
The Crowtail- 3-Axis Digital Accelerometer based on the excellent ADXL-345, this digital 3-axis accelerometer has excellent EMI protection.

20. Crowtail- 3-Axis Digital Gyro



Crowtail- 3-axis Gyro module based on ITG 3200. It is the world's first single-chip, digital-output, 3-axis MEMS motion processing gyro optimized for gaming, 3D mice, and motion-based remote control applications for Internet connected Digital TVs and Set Top Boxes.

21. Crowtail- 3-Axis Digital Compass



Crowtail- 3-Axis Digital Compass uses I²C based Honeywell HMC5883L digital compass. This ASIC is equipped with high resolution HMC118X magneto-resistive sensors and a 12-bit ADC. It provides compass heading accuracy up to 1° to 2°.

22. Crowtail- A6 GPRS/GSM Module



How about “DIY” a phone by using the Crowtail module? This is a new Crowtail- A6 GPRS/GSM Module, which is use the newest A6 gprs/gsm chip, A6 is a low-cost but efficient chip. With its ultra-small size, low power consumption and wide operating temperature range, A6 is an ideal solution for M2M applications, for automotive, industrial and PDA, personal tracking, power environmental monitoring, wireless POS, smart metering, and other M2M applications, to provide comprehensive GSM / GPRS text messaging, voice and data transmission services. Our module has specially Crowtail interface and we design this module for making it work with Arduino/Crowduino more convenient.

23. Crowtail- ESP8266 nodeMCU



Crowtail- ESP8266 Node MCU, it adds six Crowtail interface on the board, so you can easily to use the ESP8266 IOT board with other Crowtail modules. We wired up a USB-Serial chip that can upload code. It also has auto-reset so no noodling with pins and reset button pressings. To make it easy to use for portable projects, we added a connector for 3.7V Lithium polymer batteries and built in battery charging. You don't need a battery, it will run just fine straight from the micro USB connector. But, if you do have a battery, you can take it on the go, then plug in the USB to recharge.

Part2: Crowtail Applications

Lesson1: LED Control

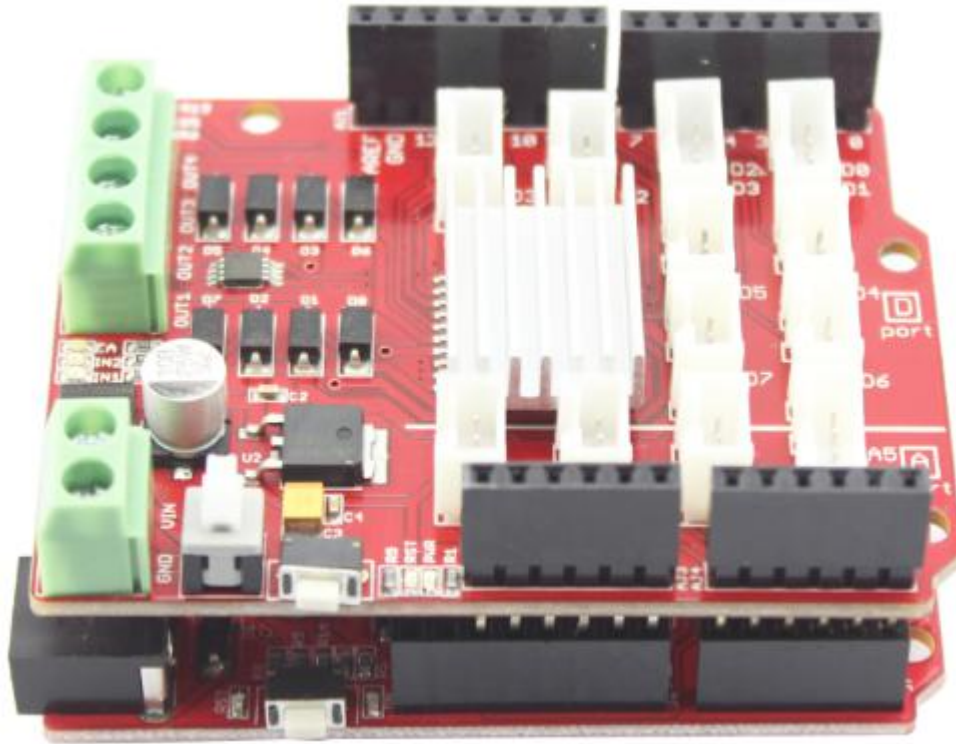
First we need to achieve some simple control, to make you more familiar with this kit. LED control is based on Arduino. In this lesson, you can learn how to control the LED with Crowtail button.

Material:

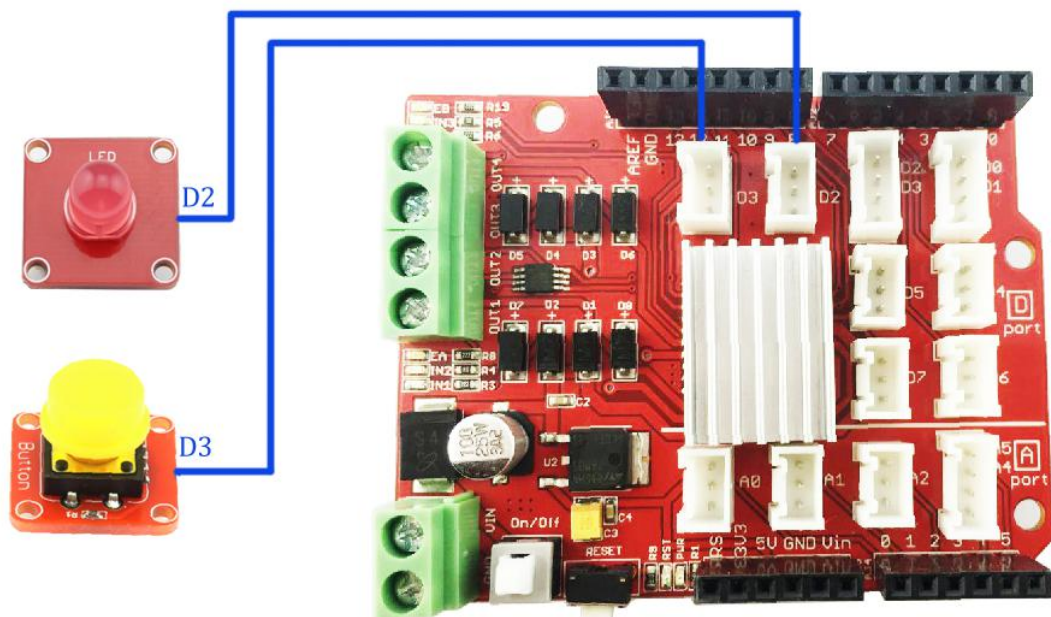
- Crowduino UNO x 1
- Crowtail- Motor Base Shield x 1
- Crowtail- LED x 1
- Crowtail- Button x1
- USB Cable x 1

Hardware Connection

Plug the Crowtail- Base Shield onto the Crowduino UNO:



Then connect the Crowtail- LED to the D2 port of Motor Base Shield, Crowtail- Button to D3 port, as following:



Firmware:

In this demo code, firstly we defined the Pin2, Pin3 as output with the *pinMode()* function and defined the Pin4,Pin5 as input with the *pinMode()* in the *setup()*, which will run once when the program start:

```
// the setup routine runs once when you press reset  
void setup() {
```

```
// initialize the digital pin as an output.  
pinMode(led, OUTPUT);  
pinMode(button, INPUT);  
}
```

And then, the Arduino read the status of the button with the function *digitalRead()*:

```
// read the state of the pushbutton value:  
buttonState = digitalRead(button);
```

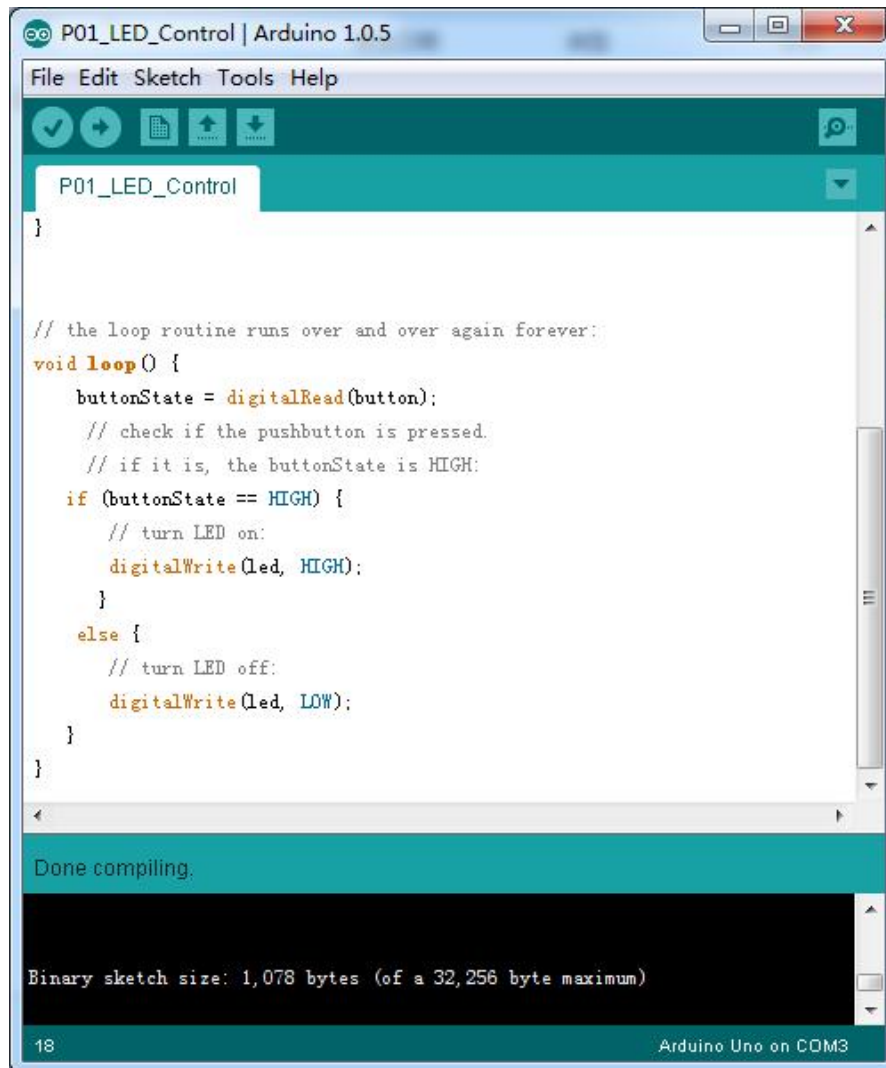
And thus to decide make the LED on or off, with the function *digitalWrite()*:

```
// check if the pushbutton is pressed.  
// if it is, the buttonState is HIGH:  
if (buttonState == HIGH) {  
    // turn LED on:  
    digitalWrite(led, HIGH);  
}  
else {  
    // turn off LED:  
    digitalWrite(led, LOW);  
}
```

Program downloading:

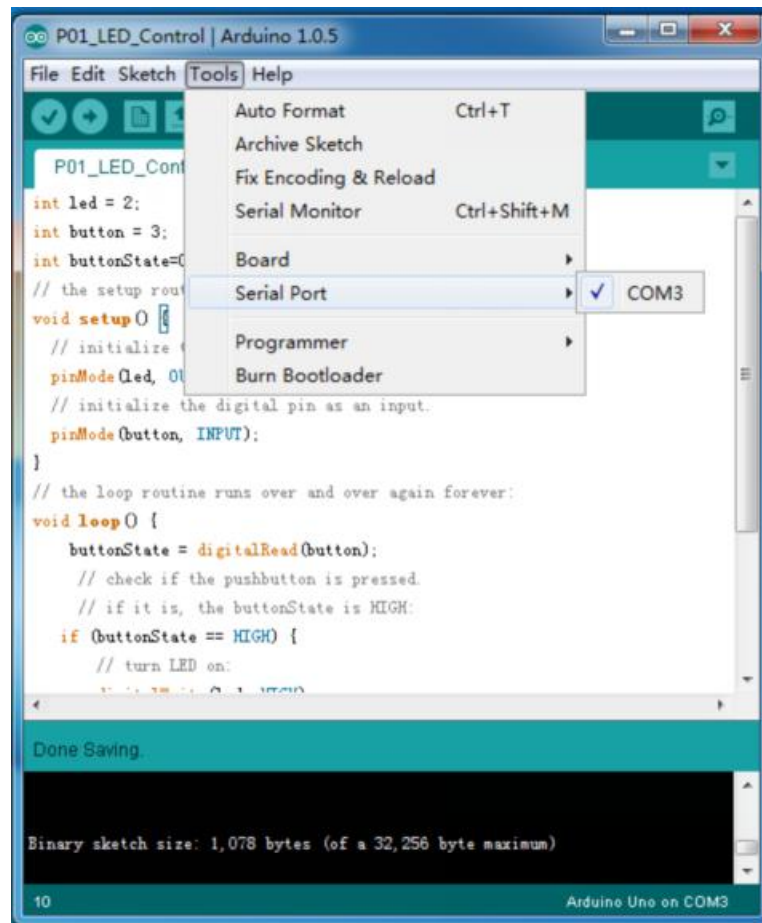
Download the [Crowtail Deluxe Kit Demo Code](#), Open the *P01_LED_Control.ino* with Arduino IDE as below:



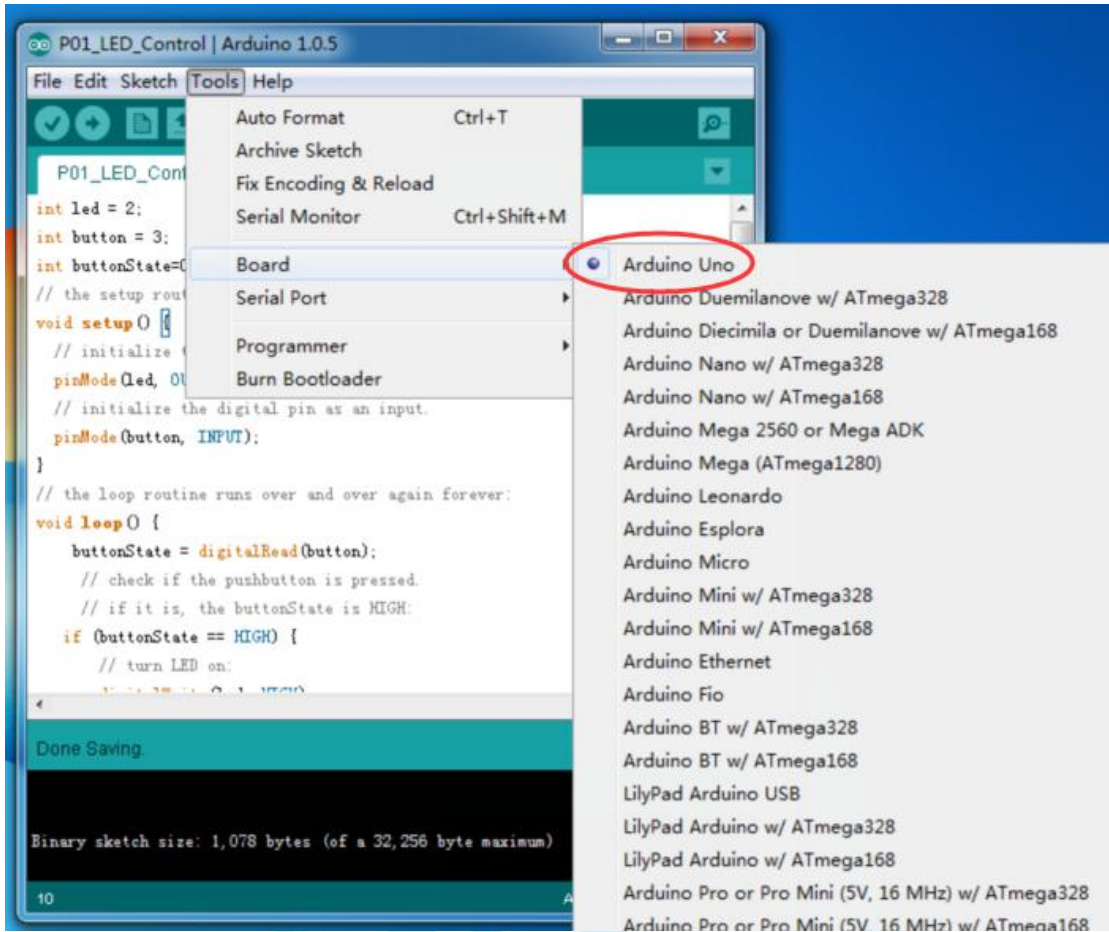


```
Arduino IDE - P01_LED_Control | Arduino 1.0.5
File Edit Sketch Tools Help
P01_LED_Control
}
// the loop routine runs over and over again forever:
void loop() {
  buttonState = digitalRead(button);
  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(Led, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(Led, LOW);
  }
}
}
Done compiling.
Binary sketch size: 1,078 bytes (of a 32,256 byte maximum)
18 Arduino Uno on COM3
```

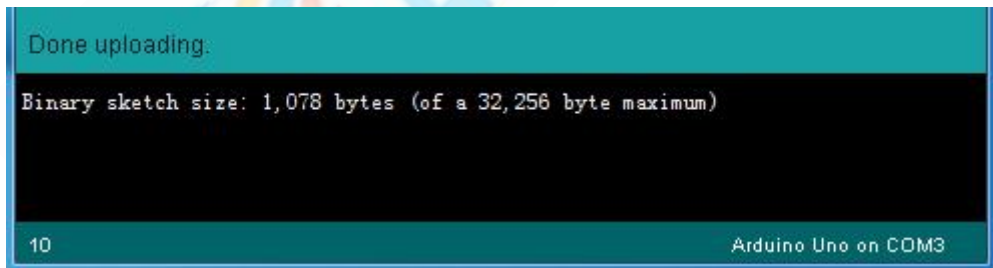
Click *Tools->SerialPort*, and choose the right com port, which will list after the driver successfully installed:



Click *Tools->Board*, choose the Arduino board you are using, such as the Arduino Uno



Click *Upload*, the code will be compiled and uploaded to the board. The down area of IDE shows following information if download succeed:



When you press the button, the LED turns ON. Otherwise, the LED turns OFF.
(Explore more Arduino functions here: <http://www.arduino.cc/en/Reference/HomePage>)

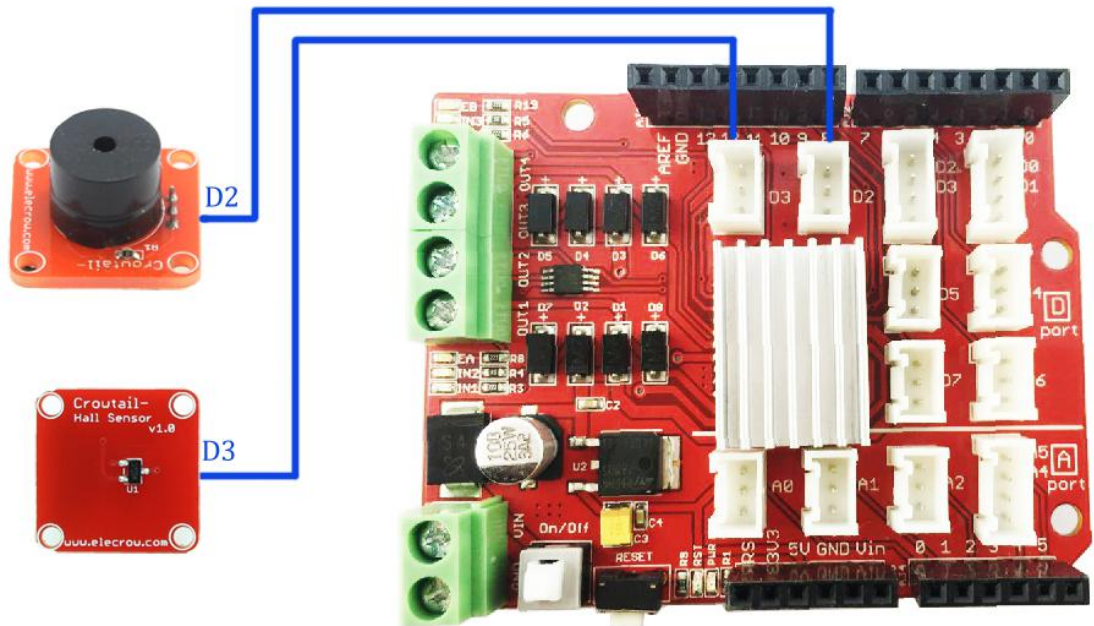
Lesson2: Hall Effect Switch

The hall effect switch is in common use in our daily life, such as the lock of the automatic door. In this lesson, we will tell you how to use the hall sensor. When you put the magnet approach to hall sensor, the buzzer will alarm.

Material:

- Crowtail- Buzzer x 1
- Crowtail- Hall sensor x 1

Hardware Connection



Open the Arduino Code [P02_Hall_Effect_Switch.ino](#), and upload it to Arduino board. When the magnet close to hall sensor, the buzzer will alarm.

Firmware:

In the programs, the Arduino first read the status of the hall sensor with the function *digitalRead()*:

```
// read the state of the hall sensor:
HallState = digitalRead(hallsensor);
```

And thus to decide make the buzzer alarm or not, with the function *digitalWrite()*:

```
// if have magnet near by hall sensor, the HallState is LOW, the buzzer will alarm:
if (HallState == LOW) {
    digitalWrite(buzzerPin, HIGH);
}
else {
    digitalWrite(buzzerPin, LOW);
}
```

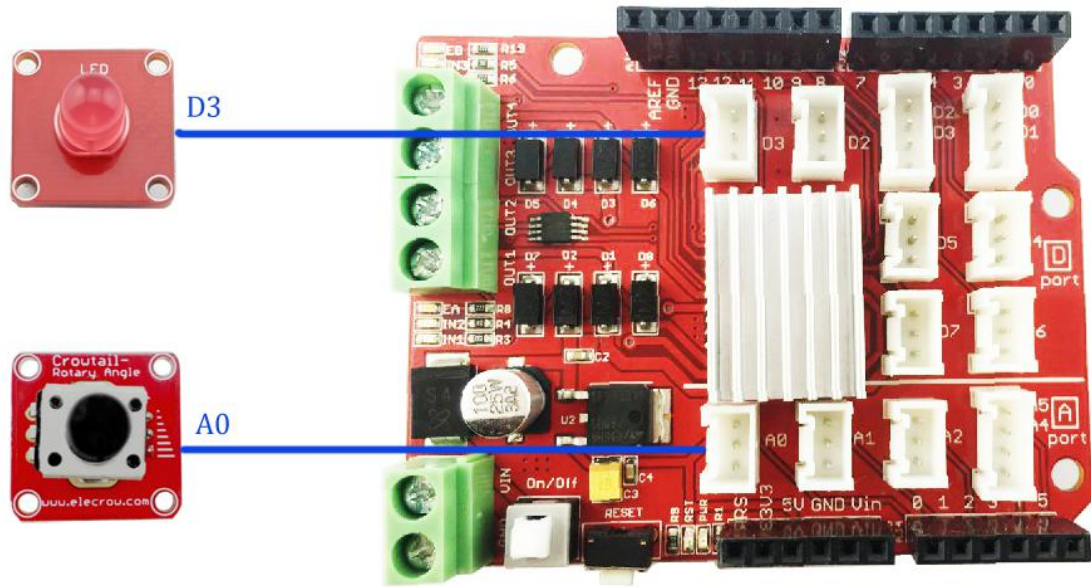
Lesson3: PWM Control LED

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. It is very useful in our electronic world and we need to learn how to use it.

Material:

- Crowtail- LED x 1
- Crowtail- Rotary Angle Sensor x 1

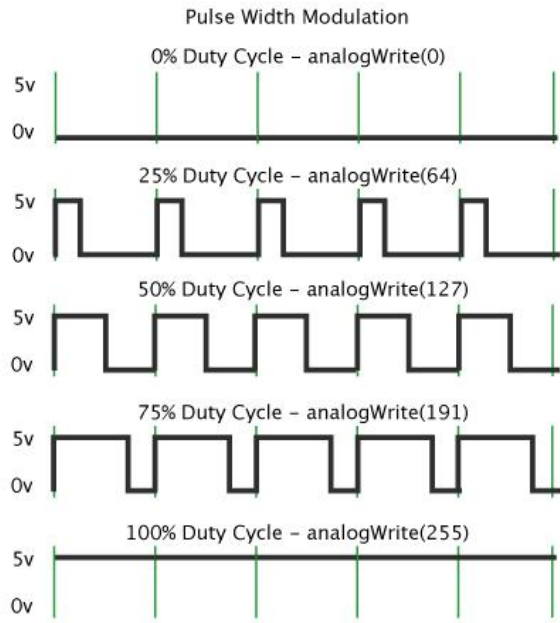
Hardware Connection



About the PWM

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to `analogWrite()` is on a scale of 0 - 255, such that `analogWrite(255)` requests a 100% duty cycle (always on), and `analogWrite(127)` is a 50% duty cycle (on half the time) for example.



Note!!!

Arduino UNO PWM pins: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

In this lesson, we adjust the Rotary Angle Sensor to control the PWM output then control the brightness of LED.

Open the Arduino code [P03_PWM_Control_LED.ino](#) and download it to the Arduino.

When adjust the Rotary angle sensor, the brightness of LED will change. Also you can open the serial monitor to see the angle value.



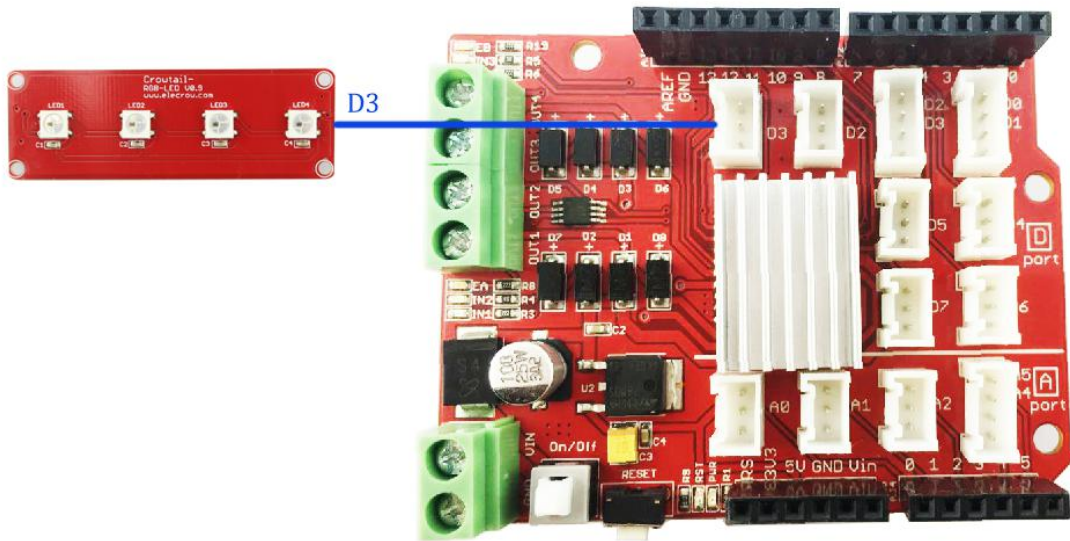
Lesson4: Colorful RGB LED

In this lesson we will learn how to control the one wire RGB LED WS2812B which is a Chainable & Addressable LED. You can control all the LED with only one microcontroller pin and control every LED with different color at the same time. Let's begin!

Material:

- Crowtail- RGB-LED x 1

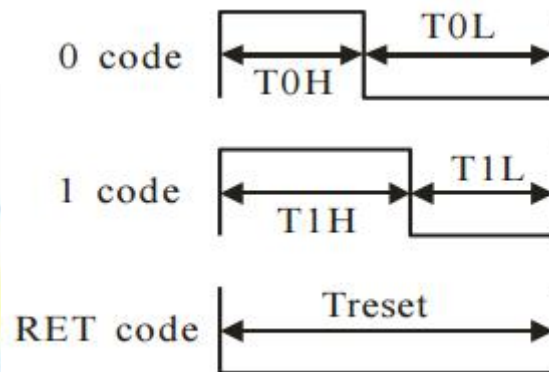
Hardware Connection



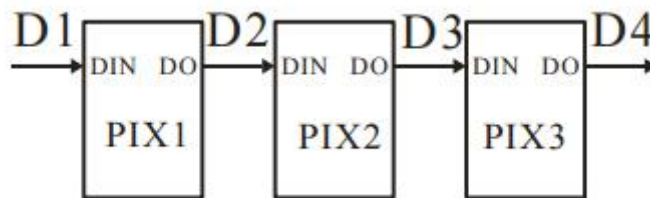
About the WS2812B

WS2812B is an intelligent control LED light source that the control circuit and RGB chip are integrated in a package of 5050 components. Its internal includes intelligent digital port data latch and signal reshaping amplification drive circuit.

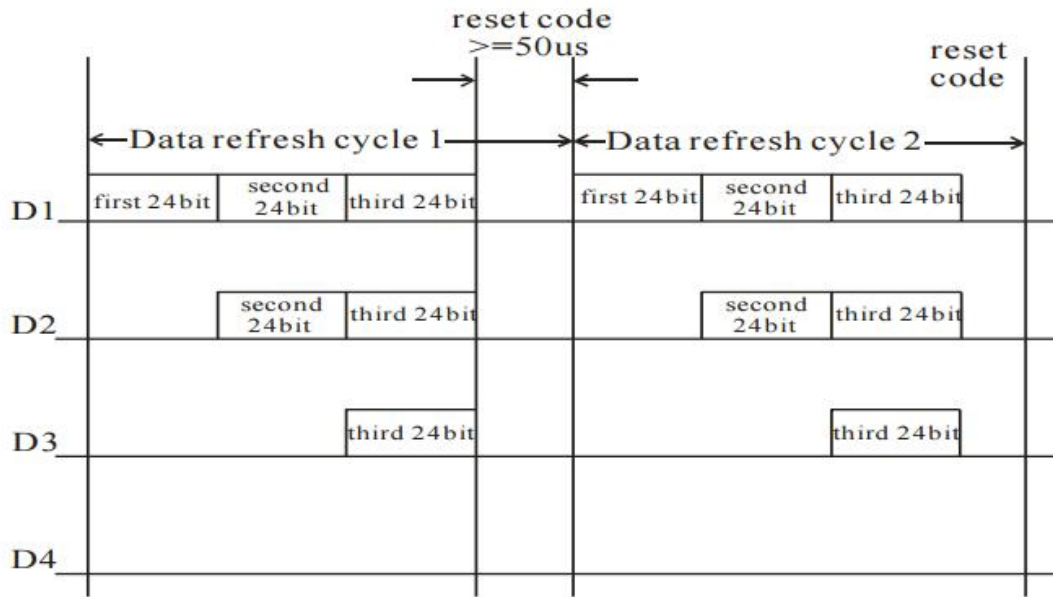
Sequence chart:



Cascade method:



Data transmission method:



Now, Let's run the colorful RGB LED.

For this application, an Arduino library "**Adafruit_NeoPixel**" is needed. Firstly, copy the folder of "**Adafruit_NeoPixel**" to the Arduino library file: ... [\Arduino\libraries](#)

Open the [P04_RGBLED.ino](#):

Upload the program to Arduino or Crowduino, have a try and you can see the LED flashing.

=====

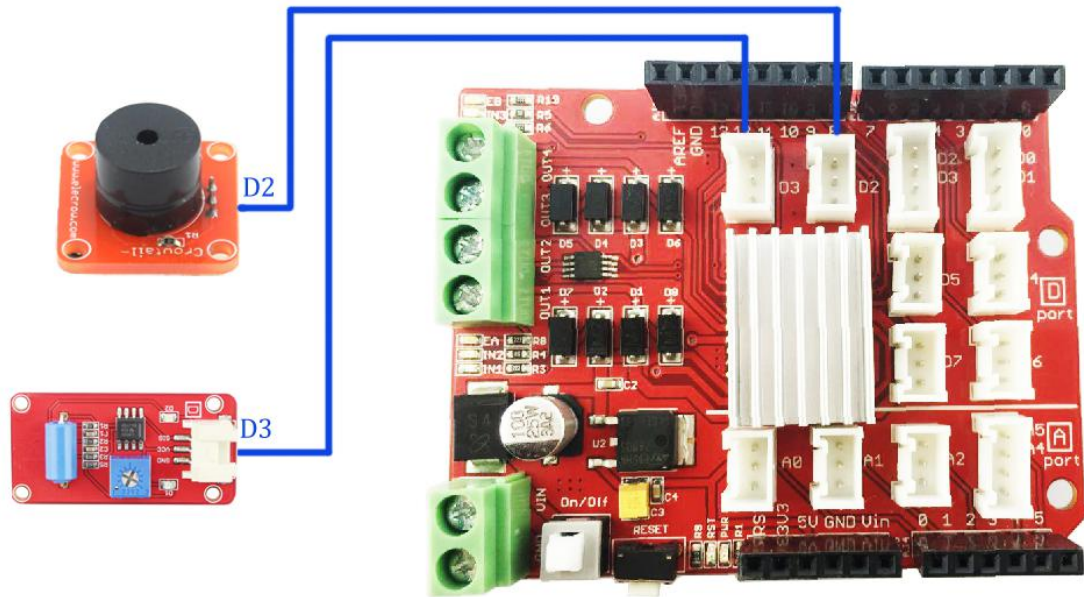
Lesson5: Vibration Alarm

Do you want to make a vibration alarm? Sometime you do not want other people to touch your things, so you can make an alarm, if someone touch it, it will alarm .

Material:

- Crowtail- Vibration Sensor x 1
- Crowtail- Buzzer x 1

Hardware Connection



Firmware

Open the [P05_Vibration_Alarm.ino](#)

In the program, the Arduino firstly detects whether have vibration:

```
vibrationState = digitalRead(vibrationsensor);
```

And if have vibration, the Arduino board set a logic HIGH to the Buzzer, alarm!

```
if (vibrationState == HIGH) {  
    // buzzer alarm  
    digitalWrite(buzzerPin, HIGH);  
    delay(500);  
}  
else {  
    // turn buzzer off:  
    digitalWrite(buzzerPin, LOW);  
}
```

Upload successfully, when you swing the sensor, the buzzer will alert.



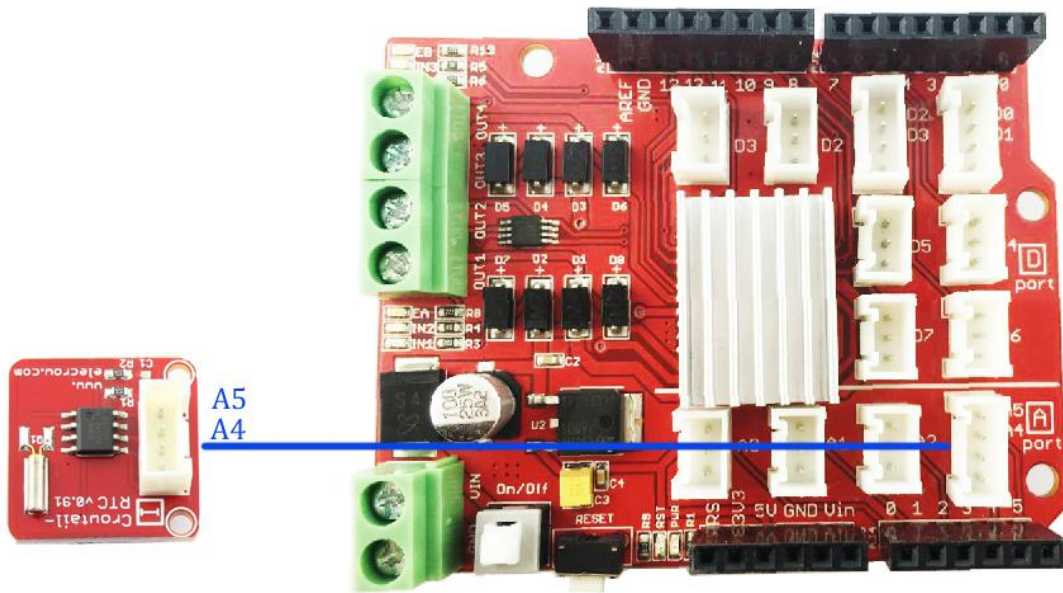
Lesson6: Make an electric watch

Do you want to make an electric watch? It is funny and useful. In this lesson, we will tell you how to make it, now let's begin.

Material:

- Crowtail- RTC x 1

Hardware Connection



Firmware:

For this application, an Arduino library “RTC” is needed. Firstly, copy the folder of “RTC” to the Arduino library file: ... \Arduino\libraries

Open the Arduino code [P06_RTC.ino](#)

In the function *Loop()*, Arduino initialize Serial port, RTC and adjust the time:

```
void setup () {  
  Serial.begin(9600);  
  Wire.begin();  
  RTC.begin();  
  if (! RTC.isrunning()) {  
    Serial.println("RTC is NOT running!");  
    // following line sets the RTC to the date & time this sketch was compiled  
    RTC.adjust(DateTime(__DATE__, __TIME__));  
  }  
}
```

Upload the program to Arduino or Crowduino, open the monitor of the Arduino, after running this program, the RTC module will synchronized the time from the computer, you can see the time now.



Lesson7: Temperature display system

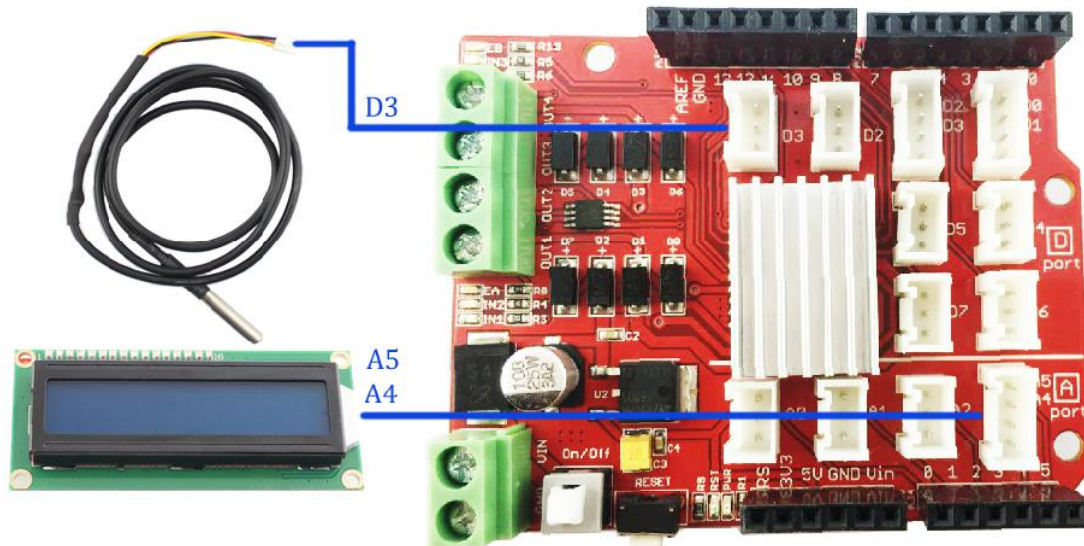
In our daily life, temperature is a important information. In this lesson, we will learn how to get the temperature digital data with DS18B20 and display it in the LCD.

Material:

- Crowtail- One Wire Waterproof Temperature Sensor x 1
- Crowtail- IIC LCD x 1

Hardware Connection

Connect the Crowtail- One Wire Waterproof Temperature Sensor to the **D3**, and IIC LCD to **A5 /A4**:

**Firmware**

For this application, Arduino library **"DallasTemperature"**, **"OneWire"** and **"LiquidCrystal"** are needed. Firstly, copy the folder file of **"DallasTemperature"**, **"OneWire"** and **"LiquidCrystal"** to the Arduino library file: ... [\Arduino\libraries](#)

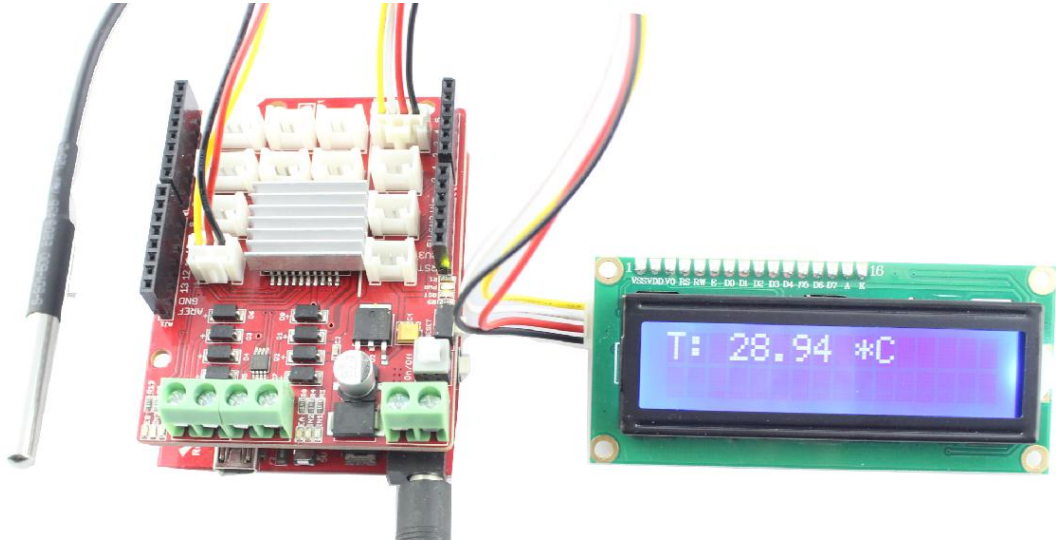
Open the Arduino code: [P07_Temperature_display_system.ino](#)

After getting the temperature data from DS18B20, the Arduino will send data to IIC LCD and display them on it. In this lesson, the temperature information also send to the monitor, you can open the Arduino IDE monitor to see it.

```
void loop(void)
{
  // call sensors.requestTemperatures() to issue a global temperature
  // request to all devices on the bus
  Serial.print("Requesting temperatures...");
  sensors.requestTemperatures(); // Send the command to get temperatures
  Serial.println("DONE");
  // After we got the temperatures, we can print them here.
  // We use the function ByIndex, and as an example get the temperature from the first sensor only.
  Serial.print("Temperature for the device 1 (index 0) is: ");
  Serial.println(sensors.getTempCByIndex(0));
  float t=sensors.getTempCByIndex(0);
  // set the cursor to column 0, line 0
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 0);
  lcd.print("T: ");
  // print the number of seconds since reset:
  lcd.print(t);
}
```

```
lcd.print(" *C");  
lcd.setBacklight(HIGH);  
delay(1000);  
}
```

Upload the program to Arduino or Crowduino, you can see the digital data of temperature in the LCD.



Lesson8: Ultrasonic ranging

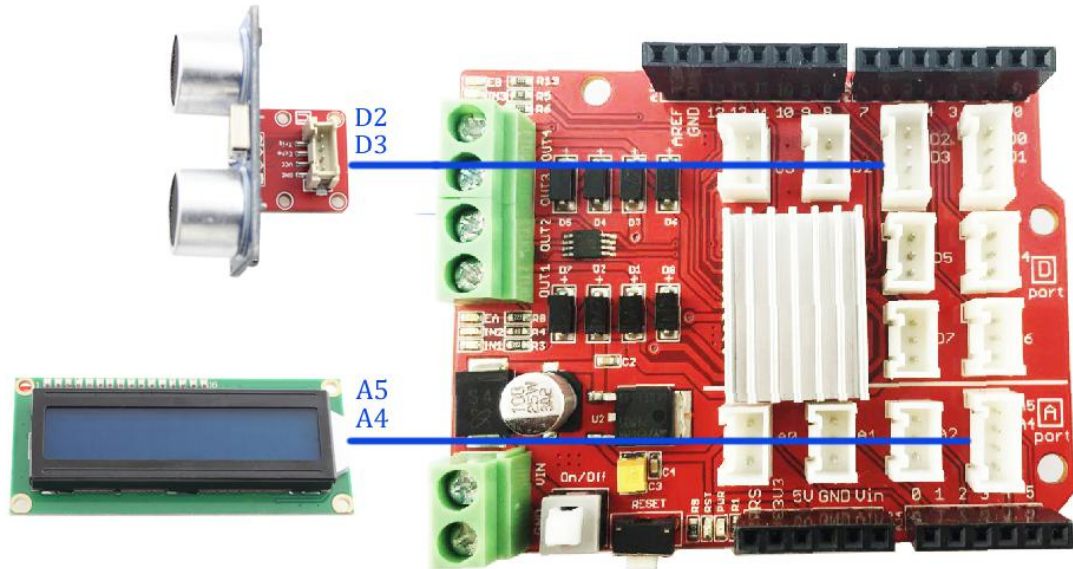
In this lesson, we will tell you how to detect the distance with ultrasonic and display the data with IIC LCD.

Material:

- Crowtail- Ultrasonic sensor x 1
- Crowtail- IIC LCD x 1

Hardware Connection

Connect the Crowtail- Ultrasonic sensor to D2/D3, and the IIC LCD to the "A4/A5" as below:



Firmware

For this application, Arduino library “**Ultrasonic**” and “**LiquidCrystal**” are needed. Firstly, copy the folder file of “**Ultrasonic**” and “**LiquidCrystal**” to the Arduino library file: ... [\Arduino\libraries](#)

Open the Arduino Code: [P08_Ultrasonic_ranging.ino](#)

In the firmware, we initialize the ultrasonic and LCD:

```
Ultrasonic ultrasonic(2,3);//Init an Ultrasonic object
// Connect via i2c, default address #0 (A0-A2 not jumpered)
LiquidCrystal lcd(0);
int Distance=0;
intPre_Distance=0;
void setup() {
    // set up the LCD's number of rows and columns:
    lcd.begin(16, 2);
}
```

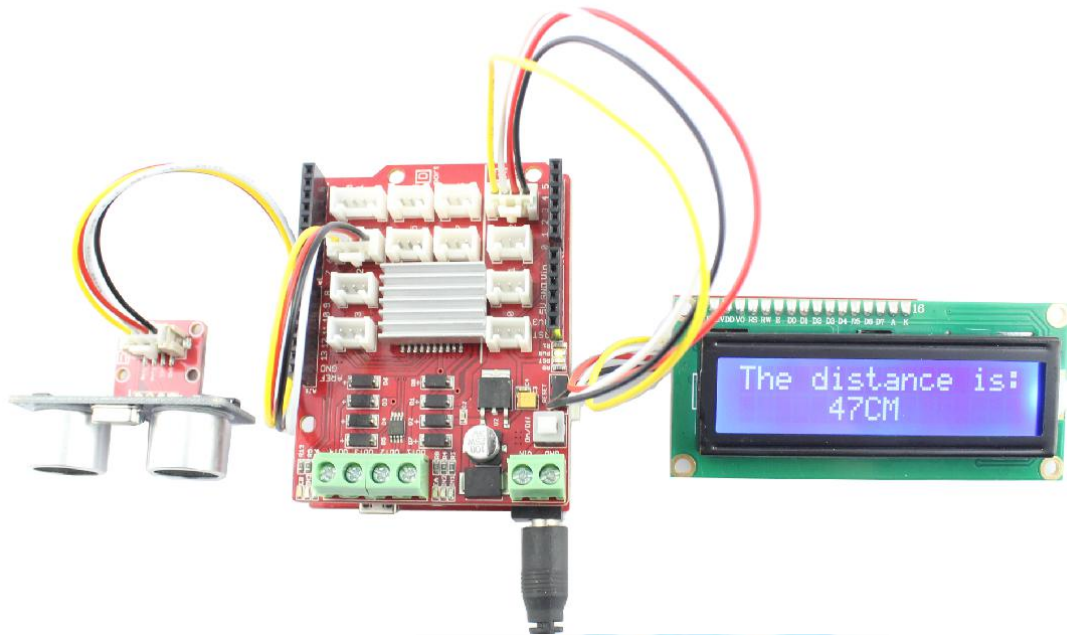
And get the distance:

```
Distance=ultrasonic Ranging(CM);//get the current result;
```

If the Current distance is different with the pre-distance, we display it in the LCD:

```
if(Distance!=Pre_Distance)
{
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("The distance is:");
    lcd.setCursor(5, 1);
    lcd.print(Distance);
    lcd.print("CM");    //the uint of distance
    Pre_Distance=Distance;
}
```


After successfully upload the code, the LCD will display the current distance.



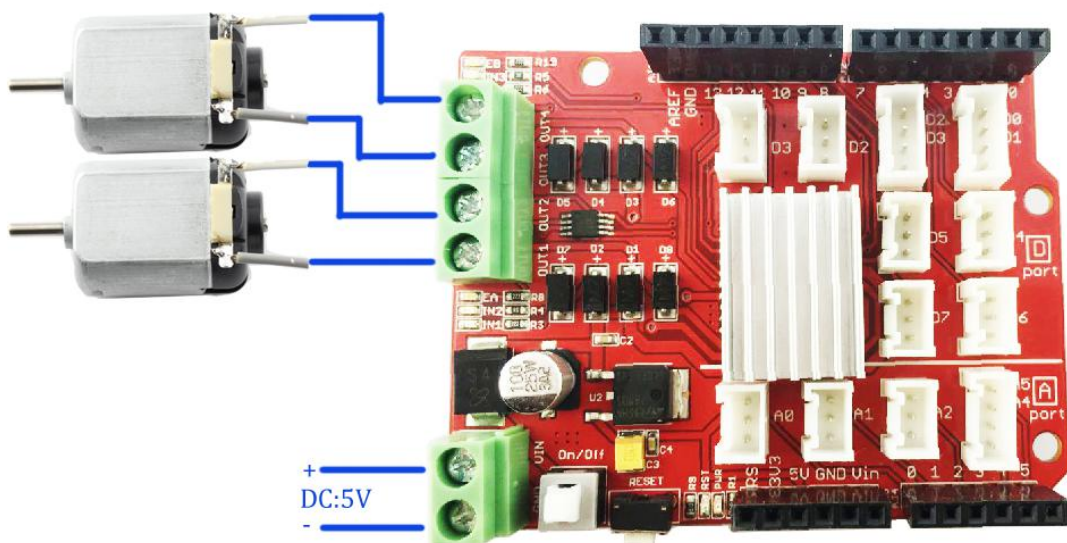
Lesson9: Motor Control

DC motors in the electronic world is very common, for example, you want to make a robot or toy car, it should be arrange to play a role, so learn to control the motor is necessary.

Material:

- DC motor x 2

Hardware Connection



Firmware

And then open the [P09_Motor_Control.ino](#)

With this Motor Base Shield, you can simultaneously control 2 DC motors. 6 Arduino pins are needed to control the 2 DC motors. D8/D9/D11 pins are for controlling motor_1, and D10/D12/D13 are for controlling the motor_2.

Initial in the *setup()*, show as below:

```
pinMode(pinI1,OUTPUT); //set to output
pinMode(pinI2,OUTPUT);
pinMode(speedpinA,OUTPUT);
pinMode(pinI3,OUTPUT);
pinMode(pinI4,OUTPUT);
pinMode(speedpinB,OUTPUT);
```

Upload the program to Arduino & Crowduino, you can see the motor start to run quickly.



Lesson10: Servo control

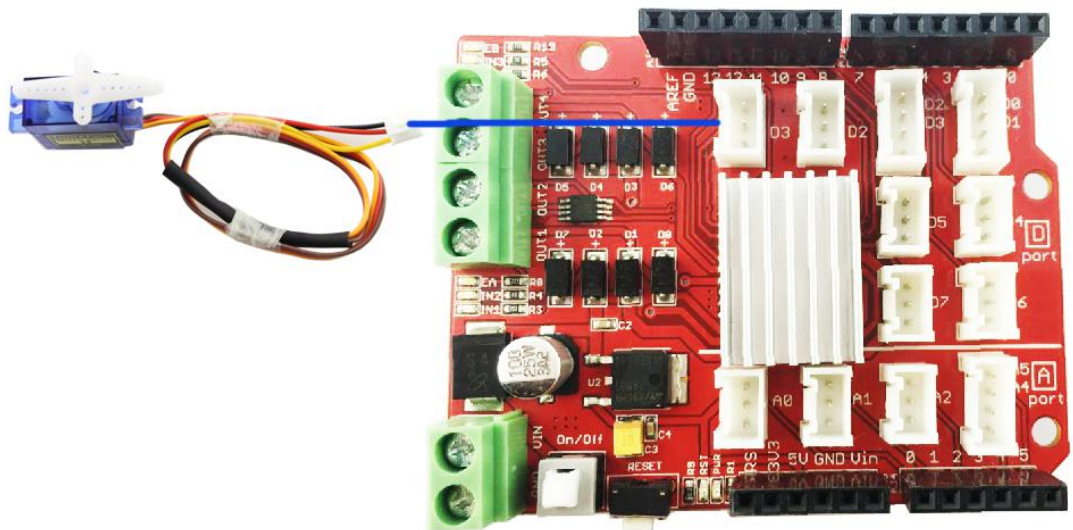
Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds. In this lesson, we will tell you how to control a servo.

Material:

Crowtail- Servo x 1

Hardware Connection

Connect the Servo to “D” port as below:



Firmware

For this application, an Arduino library “**Servo**” is needed. Firstly, copy the folder file of “**Servo**” to the Arduino library file: ... [\Arduino\libraries](#)

Open the Arduino Code [P10_Servo_control.ino](#)

In the firmware, we define the Pin 5 to connect the servo.

```
myservo.attach(5); // attaches the servo on pin 5 to the servo object
```

And in the function *loop()*, we control the spiral arm of servo rotate from 0 degrees to 180 degrees and then rotate from 180 degrees to 0 degrees.

```
for(pos = 0; pos< 180; pos += 1) // goes from 0 degrees to 180 degrees
{ // in steps of 1 degree
myservo.write(pos);           // tell servo to go to position in variable 'pos'
delay(15);                    // waits 15ms for the servo to reach the position
}
for(pos = 180; pos>=1; pos-=1) // goes from 180 degrees to 0 degrees
{
myservo.write(pos);           // tell servo to go to position in variable 'pos'
delay(15);                    // waits 15ms for the servo to reach the position
}
```

After successfully uploading the code, you can see that the spiral arm rotate in steps of 1 degree.



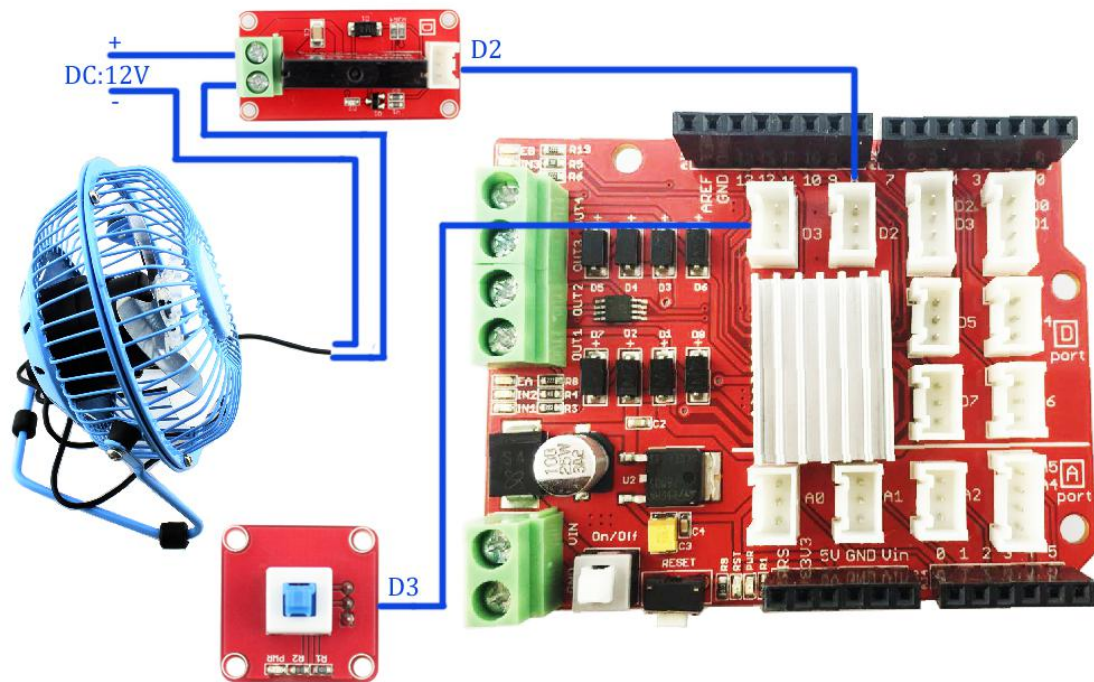
Lesson11: MINI Fan

When the summer coming, all of us have to bear the torture of hot. So how about diy a “mini fan”? But if we want to In this lesson, we will tell you how to control a fan with Solid-State Relay . The crowtail solid-state relay module, it acts as a “low voltage controlled switch to control high voltage”, For example, if you want to control power of your washer or air conditioner, which is oftern110v or 220v,by microchip such as AVR or PIC, it is necessary and safe to make your microchip control the relay first, and then control the power supply of those device with the relay.

Material:

- Crowtail- Solid-State Relayx 1
- Crowtail- Switch x1
- Fan x1

Hardware Connection



Firmware

Open the Arduino Code: [P11_MINI_Fan.ino](#)

In the function `setup()`, firstly we initialize the output and input pins of Arduino:

```
void setup()
{
  // initialize the digital pin as output.
  pinMode(relay, OUTPUT);
  // initialize the digital pin as input.
  pinMode(Crowtailswitch, INPUT);
}
```

We can read the switch state in the main loop, and then control the fan to run or stop by the solid-state relay module.

```
void loop() {
  State = digitalRead(Crowtailswitch);
  // if it is, the State is HIGH:
  if (State == HIGH) {
    // turn Fan on:
    digitalWrite(relay, HIGH);
  }
  else {
    // turn Fan off:
    digitalWrite(relay, LOW);
  }
}
```

After successfully uploading the code, when you press the switch, the fan will running, you

could enjoy the cool now !



Lesson12: IR Control

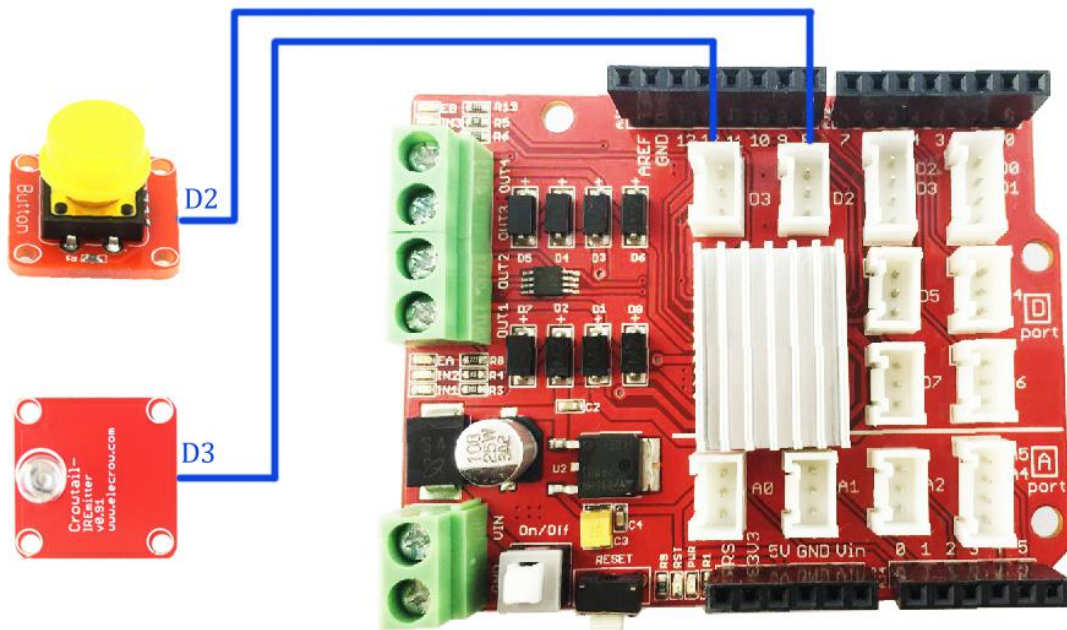
The cheapest way to remotely control a device within a visible range is via Infra-Red light. Almost all audio and video equipment can be controlled with it in nowadays. Due to this wide spread use and required components are quite cheap, thus making it ideal for us hobbyists to use IR control for our own projects. In this lesson, we will tell you how to use the IR Emitter to control LED with Arduino.

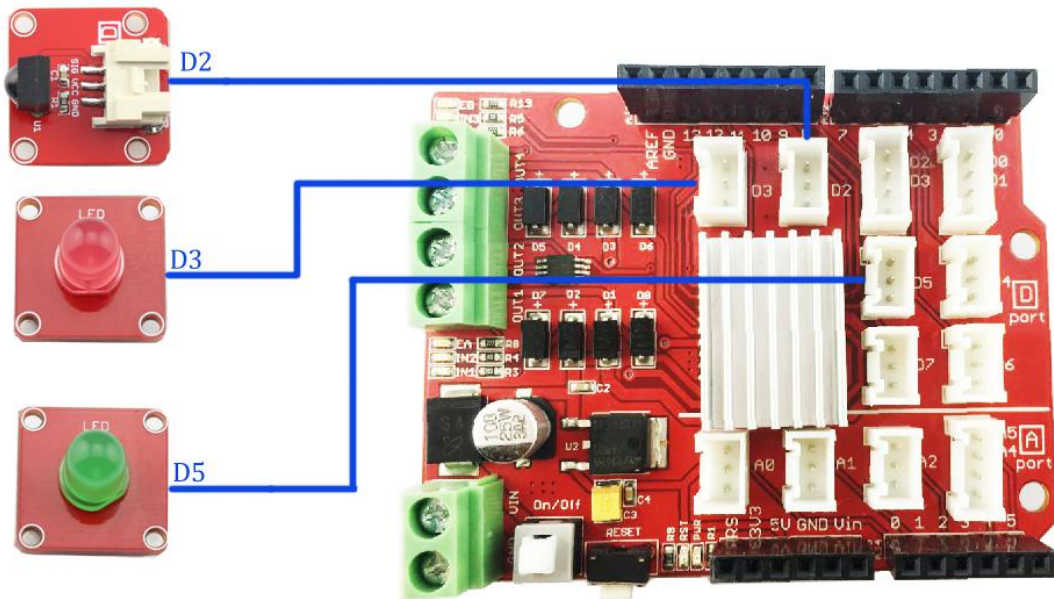
Material:

- Crowtail- LED x 2
- Crowtail- IR Receiver x 1
- Crowtail- IR Emitter x1
- Crowtail- Button

Hardware Connection

In this lesson, you need two Arduino or Crowduino, and two motor base shield. As below:





Firmware

For this application, an Arduino library “**IRremote**” is needed. Firstly, copy the folder file of “**IRremote**” to the Arduino library file: ... \Arduino\libraries

Open the Arduino Code [P12_IR_Send.ino](#)

In the firmware, we define the button connect to D2 pin and IR Emitter connect to D3 Pin, and define code we send.

```
int button=2;
int k = 0;
//unsigned char d[] = {9, 90, 91, 11, 31, 4, 1, 2, 3, 4};
unsigned char LED1[] = {15, 70, 70, 20, 60, 5, 1, 0, 0, 0,0};
unsigned char LED2[] = {15, 70, 70, 20, 60, 5, 2, 0, 0, 0,0};
unsigned char OFF[] = {15, 70, 70, 20, 60, 5, 3, 0, 0, 0,0};
```

In the function *setup()*, we initialize all of the Pins we used:

```
pinMode(button,INPUT);
```

In the main loop, we scan the button first, and according to the times of press button and then send different code:

```
void loop()
{
  if(keyscan())
  {
    k=k+1;
    if(k>3)
    {
      k=1;
    }
    switch(k)
    {
```

```

    case 1:IR.Send(LED1, 38);break;//sent the data via 38Kz IR
    case 2:IR.Send(LED2, 38);break;//sent the data via 38Kz IR
    case 3:IR.Send(OFF, 38);break;//sent the data via 38Kz IR
  }
}
}

```

Upload this code to the Arduino board of IR Emitter.

Open the Arduino Code: [P12_IR_Receive.ino](#)

In the firmware, we define the LED1 connect to D3 pin, LED2 connect to D5 Pin, and IR Receiver connect to D2.

```

int IR_Receiver=2;
int led1 = 3;
int led2 = 5;

```

In the function *setup()*, we initialize all of the Pins we used:

```

void setup()
{
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  IR.Init(IR_Receiver);
  Serial.begin(9600);
  Serial.println("init over");
}

```

In the main loop, IR Receiver detect and read value from the IR Emitter, and then Arduino will control the LED according to the value of the IR Emitter send:

```

void loop()
{
  if(IR.IsDta())
  {
    IR.Recv(dta);
    Serial.println(dta[6]);
    if(dta[6]==1)
    {
      digitalWrite(led1, HIGH);
    }
    if(dta[6]==2)
    {
      digitalWrite(led2, HIGH);
    }
    if(dta[6]==3)
    {
      digitalWrite(led1, LOW);
      digitalWrite(led2, LOW);
    }
  }
}

```

```
}
}
```

After successfully upload this code to Arduino board of IR Receiver part, when we first press the button, Arduino will light up the red LED, and the second press will light up the green LED, the third time, they will turn off two LED. Let us have a try.



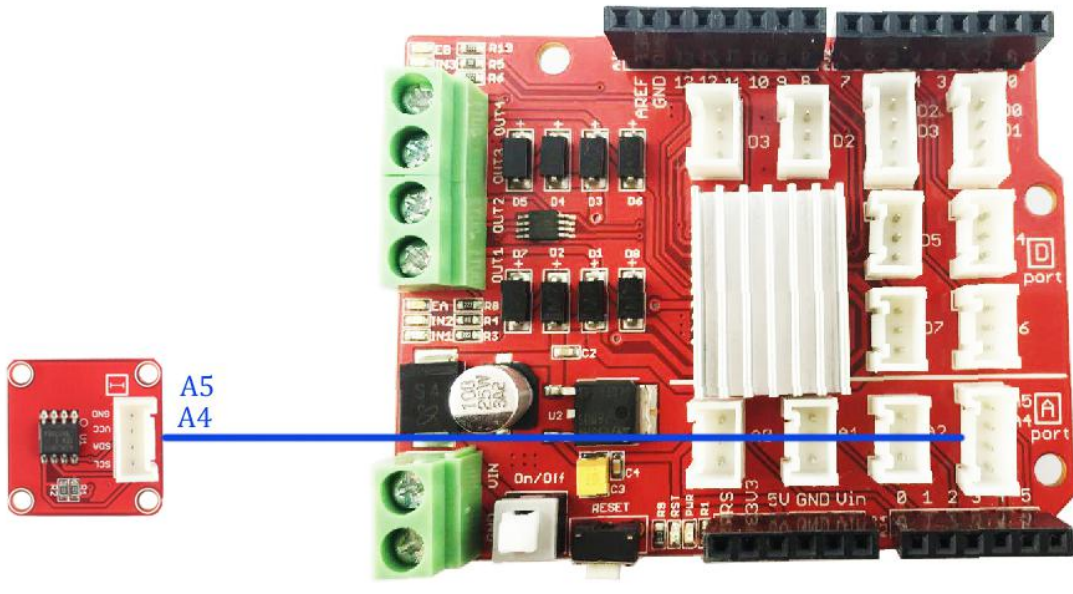
Lesson13: IIC EEPROM

Sometimes we need to save some useful data. For example, you need to save data from your Arduino, the data need to be used at next time, now you can use the IIC EEPROM. It can save the data and avoid losing the data when Arduino restart.

Material:

- Crowtail- IIC EEPROM x 1

Hardware Connection



Firmware

Open the Arduino code [P13_IIC_EEPROM.ino](#)

In this lesson, we tell you how to write the data to IIC EEPROM and read the data from IIC EEPROM. And all the operations we can achieve in the setup() function.

In the function *setup()*, we initialize the IIC library and the serial:

```
Wire.begin();
Serial.begin(9600);
// TESTS FOR EACH FUNCTION BEGIN HERE
Serial.println("Writing Test:");
for (int i=0; i<20; i++){           // loop for first 20 slots
    i2c_eeprom_write_byte(EEPROM_ADDR,i+65); // write address + 65 A or 97 a
    Serial.print(". ");
    delay(10);                       // NEED THIS DELAY!
}
}
```

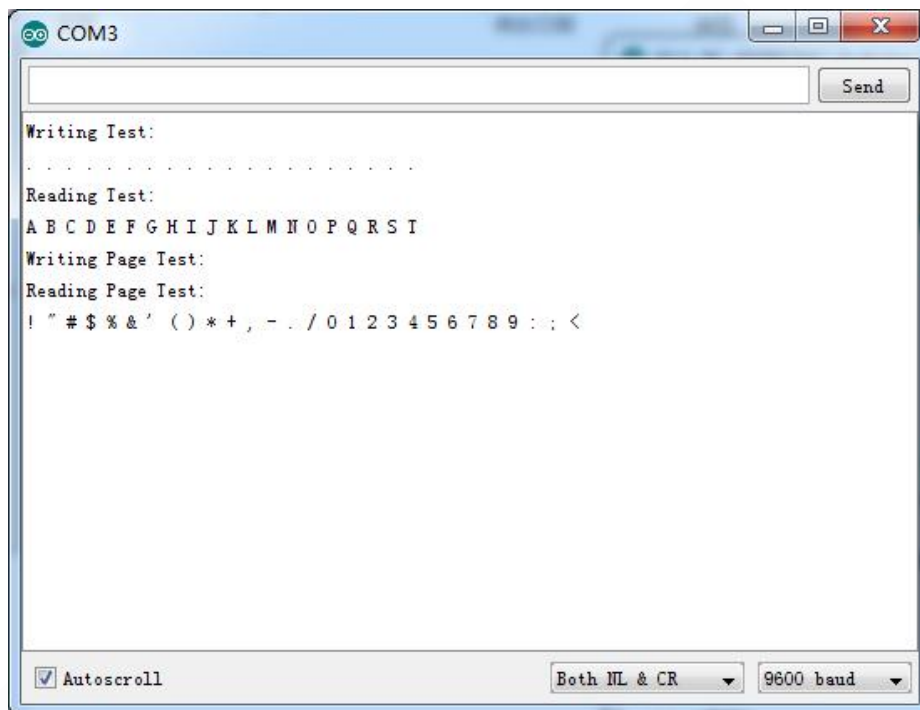
```

Serial.println("");
delay(500);

Serial.println("Reading Test:");
for (int i=0; i<20; i++){           // loop for first 20 slots
  Serial.write(i2c_eeprom_read_byte(EEPROM_ADDR, i));
  Serial.print(" ");
}

```

After successfully upload the code. You can open the monitor and see the data from Arduino, as below:



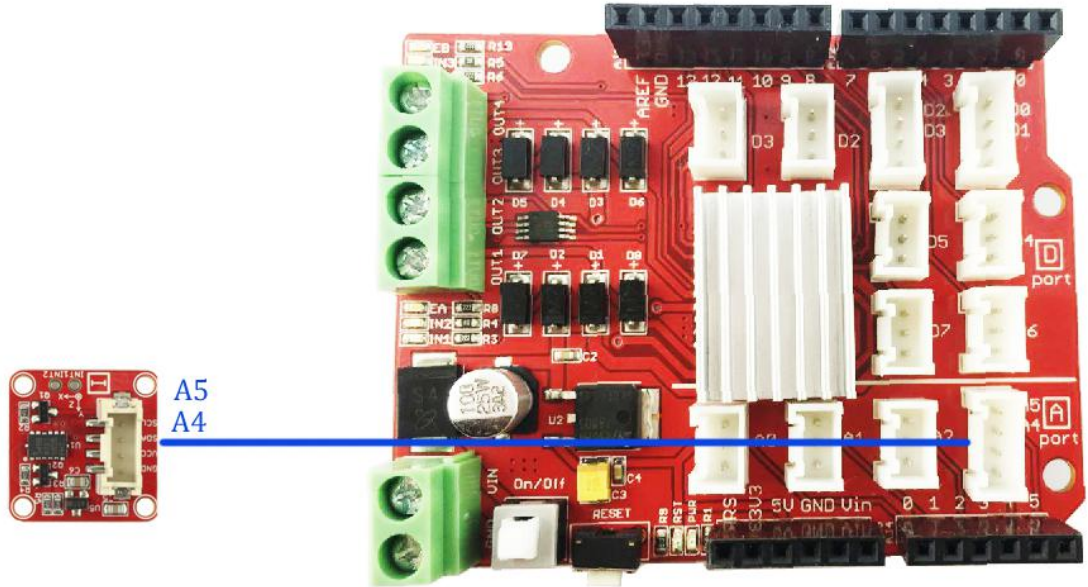
Lesson14: 3-Axis Digital Accelerometer

The 3-Axis Digital Accelerometer is a high resolution digital accelerometer providing you at max 3.9mg/LSB resolution and large $\pm 16g$ measurement range. It is based on an advanced 3-axis IC ADXL345. Have no worry to implement it into your free-fall detection project, cause it's robust enough to survive up to 10,000g shock. Meanwhile, it's agile enough to detect single and double taps. It's ideal for motion detection, Gesture detection as well as robotics. It can apply in many occasions, like a self-balancing robot car or quadcopter.

Material:

- Crowtail- 3-Axis Digital Accelerometer x 1

Hardware Connection

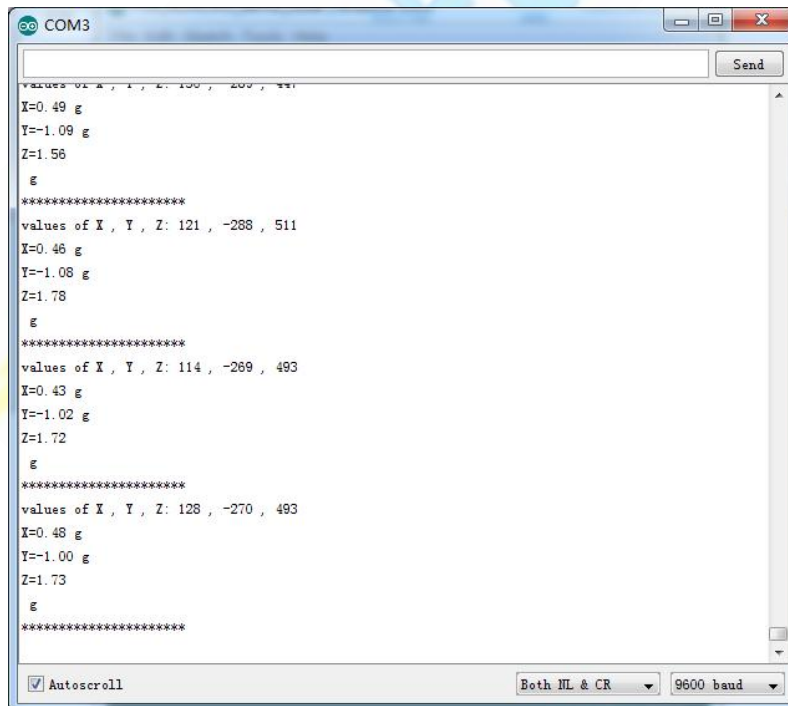


Firmware

Open the Arduino code: [P14_ADXL345_demo_code.ino](#)

This demo code shows how to get the accelerated speed of X/Y/Z axis and print out on the serial monitor.

After successfully uploading the code. Rotating or moving the accelerometer, you can see the change of the X/Y/Z axis value show as below:



Lesson15: 3-Axis Digital Gyro

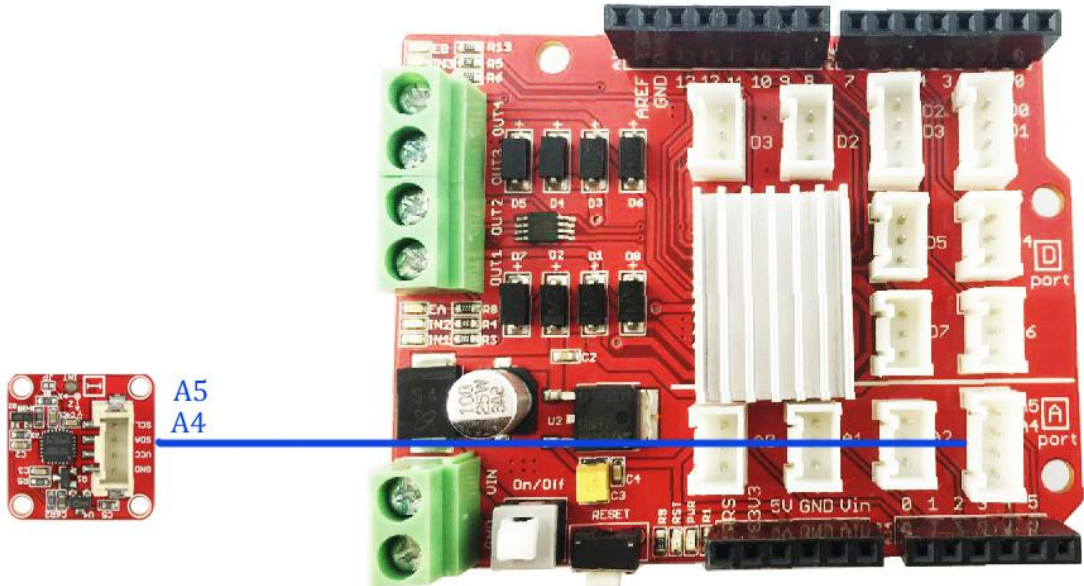
ITG-3200 is the world's first single-chip, digital-output, 3-axis MEMS motion processing gyro optimized for gaming, 3D mice, and motion-based remote control applications for Internet connected Digital TVs and Set Top Boxes. It also can apply on a self-balancing robot car or

quadcopter.

Material:

- Crowtial- 3-Axis Digital Gyro x 1

Hardware Connection



Firmware

Open the Arduino code: [P15_ITG3200_gyro.ino](#)

In the function *setup()*, we initialize the serial port and gyro:

```
void setup()
{
  Serial.begin(9600);
  gyro.init();
  gyro.zeroCalibrate(200,10);//sample 200 times to calibrate and it will take 200*10ms
}
```

In the main loop, Arduino get the X Y Z axis value from the ITG3200 and print to the monitor:

```
void loop()
{
  Serial.print("Temperature = ");
  Serial.print(gyro.getTemperature());
  Serial.println(" C");

  int16_t x,y,z;
  gyro.getXYZ(&x,&y,&z);
  Serial.print("values of X , Y , Z: ");
  Serial.print(x);
  Serial.print(" , ");
  Serial.print(y);
```

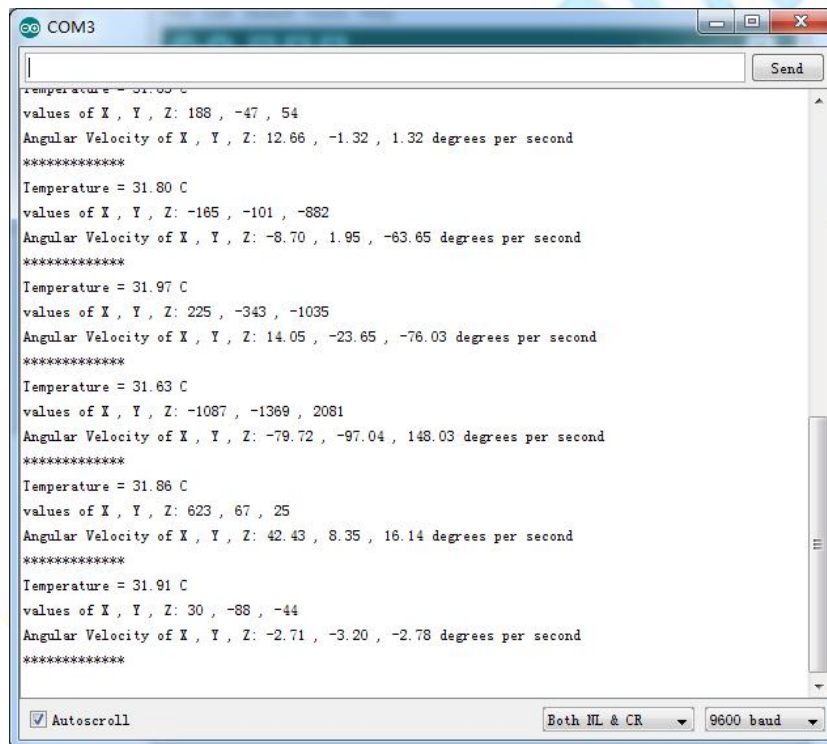
```

Serial.print(" ");
Serial.println(z);

float ax,ay,az;
gyro.getAngularVelocity(&ax,&ay,&az);
Serial.print("Angular Velocity of X , Y , Z: ");
Serial.print(ax);
Serial.print(" ");
Serial.print(ay);
Serial.print(" ");
Serial.print(az);
Serial.println(" degrees per second");
Serial.println("*****");
delay(1000);
}

```

After successfully uploading the code. Open the serial monitor to check the result.



Lesson16: 3-Axis Digital Compass

We often losing the direction when explore in the forest or sailing on the sea, so both of us need a electric compass. In this lesson we will tell you how to achieve this function by our Crowtail- 3-Axis Digital Compass.

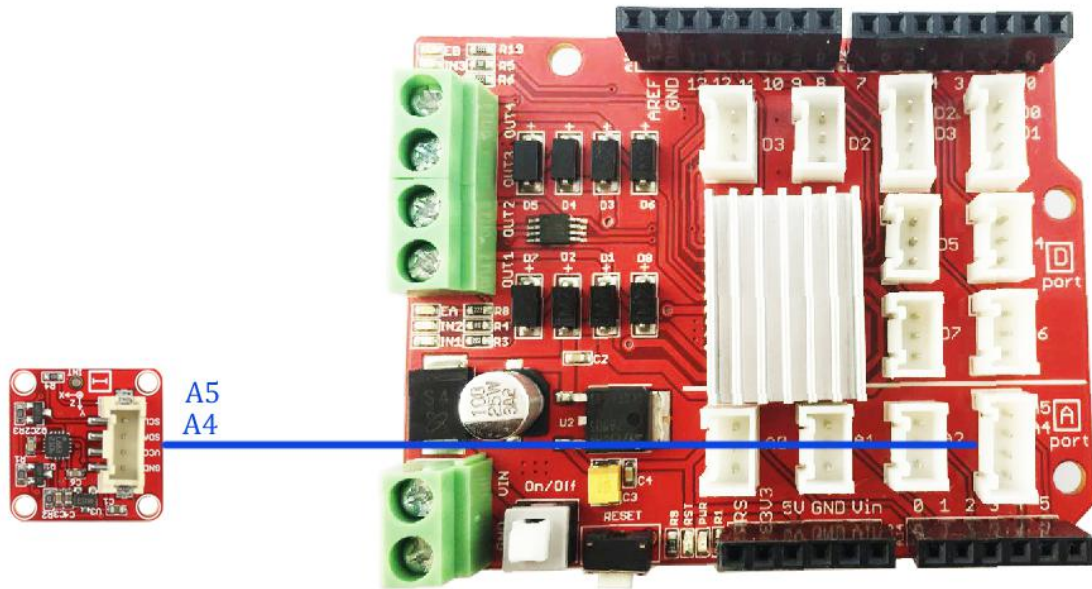
Material:

- Crowtail- 3-Axis Digital Compass x 1

43

Address: 5thFloor, Building F1, Huafeng Industry Park, Gushu Town, Bao' an District, Shenzhen, CN.

E-mail: wilson@elecrow.com

Hardware Connection**Firmware**

Open the Arduino code: [P16_HMC5883L_demo_code.ino](#)

In the function `setup()`, we initialize the IIC interface and print some information:

```
// Initialize the serial port.
Serial.begin(9600);

Serial.println("Starting the I2C interface.");
Wire.begin(); // Start the I2C interface.

Serial.println("Constructing new HMC5883L");

Serial.println("Setting scale to +/- 1.3 Ga");
error = compass.setScale(1.3); // Set the scale of the compass.
if(error != 0) // If there is an error, print it out.
    Serial.println(compass.getErrorText(error));

Serial.println("Setting measurement mode to continuous.");
error = compass.setMeasurementMode(MEASUREMENT_CONTINUOUS); // Set the measurement mode to
Continuous
if(error != 0) // If there is an error, print it out.
    Serial.println(compass.getErrorText(error));
```

In the main loop, Arduino get the Scaled Axis and print to the minitor:

```
void loop()
{
    // Retrieve the raw values from the compass (not scaled).
    MagnetometerRaw raw = compass.readRawAxis();
    // Retrieved the scaled values from the compass (scaled to the configured scale).
```

```

MagnetometerScaled scaled = compass.readScaledAxis();

// Values are accessed like so:
int MilliGauss_OnThe_XAxis = scaled.XAxis;// (or YAxis, or ZAxis)

// Calculate heading when the magnetometer is level, then correct for signs of axis.
float heading = atan2(scaled.YAxis, scaled.XAxis);

// Once you have your heading, you must then add your 'Declination Angle', which is the 'Error' of the magnetic
field in your location.
// Find yours here: http://www.magnetic-declination.com/
// Mine is: -2°37' which is -2.617 Degrees, or (which we need) -0.0456752665 radians, I will use -0.0457
// If you cannot find your Declination, comment out these two lines, your compass will be slightly off.
float declinationAngle = -0.0457;
heading += declinationAngle;

// Correct for when signs are reversed.
if(heading < 0)
    heading += 2*PI;

// Check for wrap due to addition of declination.
if(heading > 2*PI)
    heading -= 2*PI;

// Convert radians to degrees for readability.
float headingDegrees = heading * 180/M_PI;

// Output the data via the serial port.
Output(raw, scaled, heading, headingDegrees);

// Normally we would delay the application by 66ms to allow the loop
// to run at 15Hz (default bandwidth for the HMC5883L).
// However since we have a long serial out (104ms at 9600) we will let
// it run at its natural speed.
delay(66);//of course it can be delayed longer.
}

// Output the data down the serial port.
void Output(MagnetometerRaw raw, MagnetometerScaled scaled, float heading, float headingDegrees)
{
    Serial.print("Raw:\t");
    Serial.print(raw.XAxis);

```

```
Serial.print(" ");
Serial.print(raw.YAxis);
Serial.print(" ");
Serial.print(raw.ZAxis);
Serial.print(" \tScaled:\t");

Serial.print(scaled.XAxis);
Serial.print(" ");
Serial.print(scaled.YAxis);
Serial.print(" ");
Serial.print(scaled.ZAxis);

Serial.print(" \tHeading:\t");
Serial.print(heading);
Serial.print(" Radians \t");
Serial.print(headingDegrees);
Serial.println(" Degrees \t"); }
```

After successfully upload the code. Open the serial monitor and rotating the sensor to check the result:


```

COM3
Raw: -39 192 -235 Scaled: -35.88 176.64 -216.20 Heading: 1.73 Radians 98.86 Degrees
Raw: -44 190 -233 Scaled: -40.48 174.80 -214.36 Heading: 1.75 Radians 100.42 Degrees
Raw: -50 188 -234 Scaled: -47.84 172.04 -213.44 Heading: 1.80 Radians 102.92 Degrees
Raw: -58 189 -233 Scaled: -53.36 173.88 -214.36 Heading: 1.82 Radians 104.44 Degrees
Raw: -81 179 -233 Scaled: -74.52 164.68 -214.36 Heading: 1.95 Radians 111.73 Degrees
Raw: -88 177 -236 Scaled: -80.96 162.84 -217.12 Heading: 1.99 Radians 113.82 Degrees
Raw: -95 176 -235 Scaled: -87.40 161.92 -216.20 Heading: 2.02 Radians 115.74 Degrees
Raw: -95 174 -233 Scaled: -87.40 160.08 -214.36 Heading: 2.02 Radians 116.02 Degrees
Raw: -96 174 -232 Scaled: -88.32 160.08 -213.44 Heading: 2.03 Radians 116.27 Degrees
Raw: -102 169 -234 Scaled: -93.84 155.48 -215.28 Heading: 2.07 Radians 118.49 Degrees
Raw: -107 166 -235 Scaled: -98.44 152.72 -216.20 Heading: 2.10 Radians 120.19 Degrees
Raw: -113 163 -232 Scaled: -103.96 149.96 -213.44 Heading: 2.13 Radians 122.11 Degrees
Raw: -118 160 -235 Scaled: -108.56 147.20 -216.20 Heading: 2.16 Radians 123.79 Degrees
Raw: -124 155 -235 Scaled: -114.08 142.60 -216.20 Heading: 2.20 Radians 126.04 Degrees
Raw: -129 158 -234 Scaled: -118.68 145.36 -215.28 Heading: 2.21 Radians 126.61 Degrees
Raw: -132 152 -232 Scaled: -121.44 139.84 -213.44 Heading: 2.24 Radians 128.35 Degrees
Raw: -145 145 -235 Scaled: -133.40 133.40 -216.20 Heading: 2.31 Radians 132.38 Degrees
Raw: -144 144 -236 Scaled: -132.48 132.48 -217.12 Heading: 2.31 Radians 132.38 Degrees
Raw: -150 143 -234 Scaled: -138.00 131.56 -215.28 Heading: 2.33 Radians 133.75 Degrees
Raw: -151 138 -235 Scaled: -138.92 126.96 -216.20 Heading: 2.36 Radians 134.96 Degrees
Raw: -159 130 -236 Scaled: -146.28 119.60 -217.12 Heading: 2.41 Radians 138.11 Degrees
Raw: -200 91 -235 Scaled: -184.00 83.72 -216.20 Heading: 2.67 Radians 152.92 Degrees
Raw: -231 48 -233 Scaled: -212.52 44.16 -214.36 Heading: 2.89 Radians 165.64 Degrees
Raw: -246 4 -236 Scaled: -226.32 3.68 -217.12 Heading: 3.08 Radians 176.45 Degrees
Raw: -263 -39 -236 Scaled: -241.96 -35.88 -217.12 Heading: 3.24 Radians 185.82 Degrees
Raw: -261 -56 -237 Scaled: -240.12 -51.52 -218.04 Heading: 3.31 Radians 189.49 Degrees
Raw: -264 -75 -238 Scaled: -242.88 -69.00 -218.96 Heading: 3.37 Radians 193.24 Degrees
Raw: -267 -90 -236 Scaled: -245.64 -82.80 -217.12 Heading: 3.42 Radians 196.01 Degrees
Raw: -265 -96 -235 Scaled: -243.80 -88.32 -216.20 Heading: 3.44 Radians 197.30 Degrees
Raw: -268 -95 -239 Scaled: -246.56 -87.40 -219.88 Heading: 3.44 Radians 196.90 Degrees
Raw: -267 -97 -236 Scaled: -245.64 -89.24 -217.12 Heading: 3.44 Radians 197.35 Degrees
Raw: -266 -99 -238 Scaled: -244.72 -91.08 -218.96 Heading: 3.45 Radians 197.80 Degrees
Raw: -266 -103 -239 Scaled: -244.72 -94.76 -219.88 Heading: 3.47 Radians 198.55 Degrees
Raw: -268 -106 -239 Scaled: -246.56 -97.52 -219.88 Heading: 3.47 Radians 198.96 Degrees
Raw: -269 -109 -237 Scaled: -247.48 -100.28 -218.04 Heading: 3.48 Radians 1

```

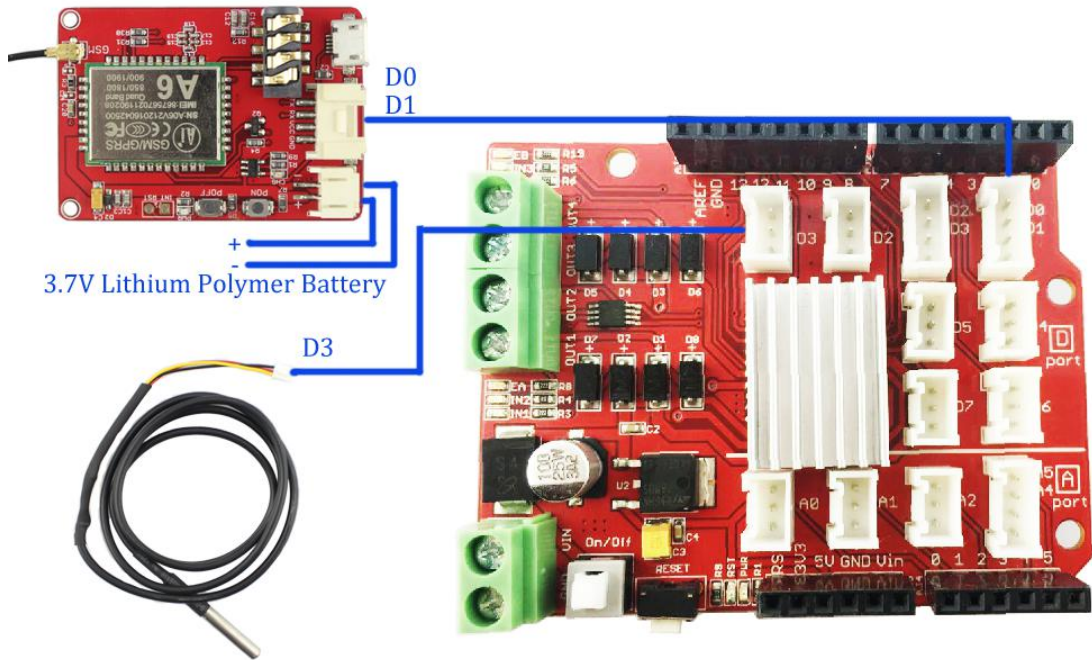
Lesson17: Send data to your Phone

Now is an era of "Internet of things", so we have to keep up with the pace of the time. In the lesson, we will tell you how to use the GPRS/GSM module—Crowtail- A6 GPRS/GSM Module. You can control your project with it anywhere that have GPRS/GSM network. In this lesson, we just teach you how to send the temperature information to your phone.

Material:

- Crowtail- A6 GPRS/GSM Module x 1
- Crowtail- One Wire Waterproof Temperature Sensor x 1

Hardware Connection



Firmware

Open the Arduino code: [P17_A6_SMS.ino](#)

In the function `setup()`, we initialize the serial port and DS18B20:

```
Serial.begin(115200);
sensors.begin();
```

In the main loop, Arduino read the temperature and send it to your phone:

```
sensors.requestTemperatures(); // Send the command to get temperatures
float t=sensors.getTempCByIndex(0);
sendData("AT\r\n",1000,DEBUG);
sendData("AT+CMGF=1\r\n",2000,DEBUG);
sendData("AT+CIPMUX=1\r\n",1000,DEBUG);
sendData("AT+CMGS=136xxxxxxxx;\r\n",1000,DEBUG);// 136xxxxxxxx change to your phone number
sendData("Now Temperatures is:",1000,DEBUG);
Serial.print(t);
sendData("°C",1000,DEBUG);
Serial.write(0x1A);
while(1);
```

After successfully upload the code. Connect the A6 GPRS/GSM module to the Motor base shield, then press the reset button, you can receive the SMS on your phone.

Lesson18: ESP8266 Web Server

If you are interesting with "IOT", you should not miss this module that called ESP8266Node MCU. In this lesson, we will tell you how to use the Crowtail- ESP8266 NodeMCU which is based on ESP8266-12E. And how to achieve remote control with it. It comes preprogrammed with the Lua interpreter, you can easy to use Lua to control your IOT project.

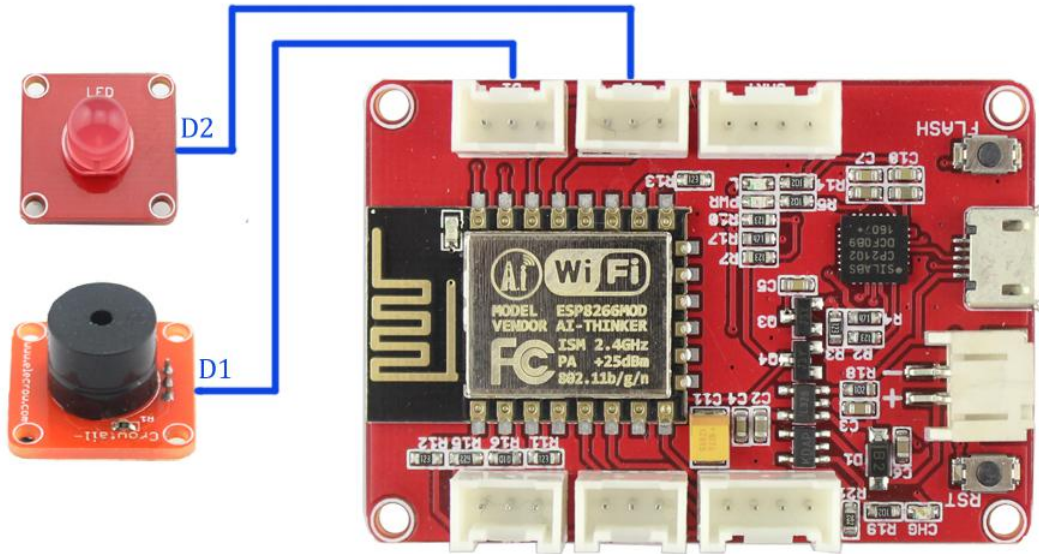
Material:

- Crowtail- ESP8266 NodeMCU x 1
- Crowtail- LED
- Crowtail- Buzzer

Hardware Connection

In this lesson, we just control the LED and Buzzer, but actually you can control other device you need.

Connect the nodeMCU to PC with micro USB cable and connect the LED and buzzer to nodeMCU. As below:



Firmware

We can control the led/ buzzer by ESP8266 Web Server.

First, In the init.lua, we set Wifi module and connect to the wireless router :

```
--init.lua
print("Setting up WIFI...")
wifi.setmode(wifi.STATION)
--modify according your wireless router settings
wifi.sta.config("Elecrow803","elecrow2014")
wifi.sta.connect()
tmr.alarm(1, 2000, 1, function()
if wifi.sta.getip()== nil then
print("IP unavaiable, Waiting...")
else
tmr.stop(1)
print("Config done, IP is "..wifi.sta.getip())
dofile("webservice.lua")
end
end)
```

In the webservice.lua, we initialize GPIO Pin we use and set the ESP8266 NodeMCU as web

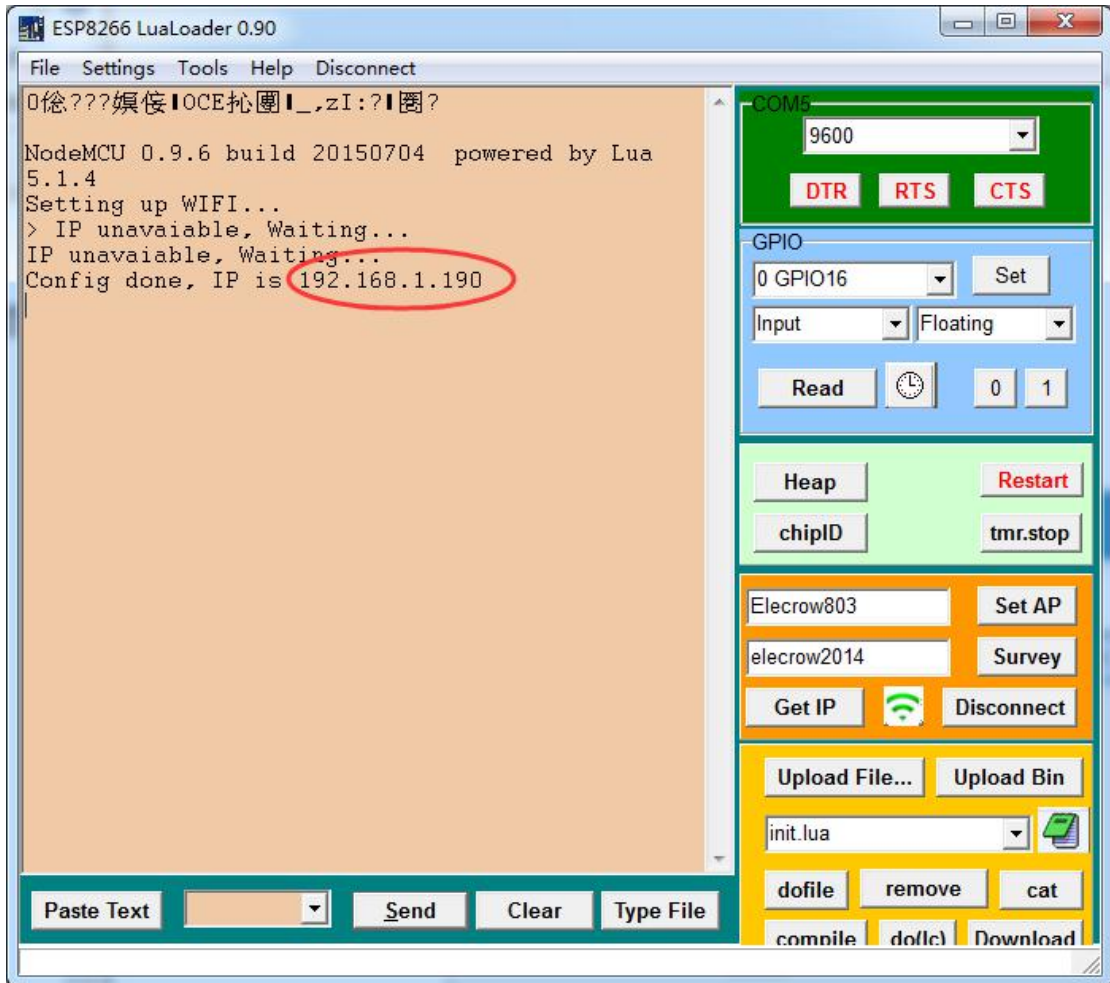
server:

```

led = 1
buzzer = 2
gpio.mode(led, gpio.OUTPUT)
gpio.mode(buzzer, gpio.OUTPUT)
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
  conn:on("receive", function(client,request)
    local buf = "";
    local _, _, method, path, vars = string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
    if(method == nil)then
      _, _, method, path = string.find(request, "([A-Z]+) (.+) HTTP");
    end
    local _GET = {}
    if (vars ~= nil)then
      for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
        _GET[k] = v
      end
    end
    buf = buf.."<h1> ESP8266 Web Server</h1>";
    buf = buf.."<p><span
style='width:70px;display:inline-block;text-align:center;'>LED</span><a
href='\"?pin=ON1\"><button>ON</button></a> <a href='\"?pin=OFF1\"><button>OFF</button></a></p>";
    buf = buf.."<p><span style='width:70px;display:inline-block;text-align:center;'>Buzzer</span><a
href='\"?pin=ON2\"><button>ON</button></a> <a href='\"?pin=OFF2\"><button>OFF</button></a></p>";
    local _on,_off = "", ""
    if(_GET.pin == "ON1")then
      gpio.write(led, gpio.HIGH);
    elseif(_GET.pin == "OFF1")then
      gpio.write(led, gpio.LOW);
    elseif(_GET.pin == "ON2")then
      gpio.write(buzzer, gpio.HIGH);
    elseif(_GET.pin == "OFF2")then
      gpio.write(buzzer, gpio.LOW);
    end
    client:send(buf);
    client:close();
    collectgarbage();
  end)
end)

```

Upload the code by ESP8266 LuaLoader.exe. After successfully upload the code. Press the dofile button and catch the IP address as below:



Copy the IP and open the browser, past the IP and visit, you could the web page shows as below:



Press the LED ON, it will turn on the LED. Also the buzzer can be controlled. Please have a try!