Directory

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.

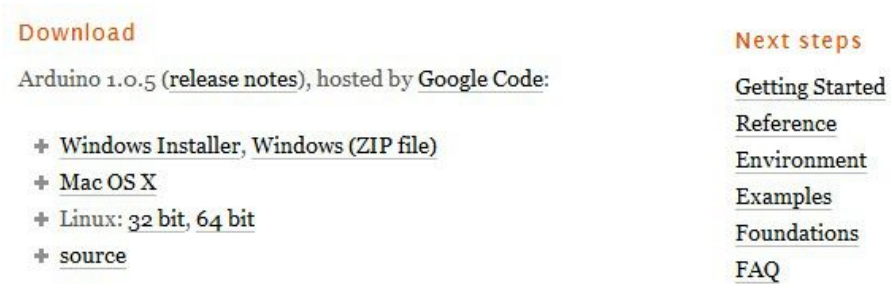
1. What's Arduino?

Arduino is a flexible and easy-to-learn open source development platform that enjoys great fame among makers, geeks and interactive artists. The Elecrow Advanced Kit for Arduino helps you get the more knowledge of Arduino then the Elecrow start kit for Arduino. It contains the most popular accessories for DIY projects such as PIR sensor, Flame sensor, serial WIFI, 4-Digit Display, IIC LCD, etc. With the straight forward instructions with the kit, you can easily dig into the Arduino world and start your own Arduino project.

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

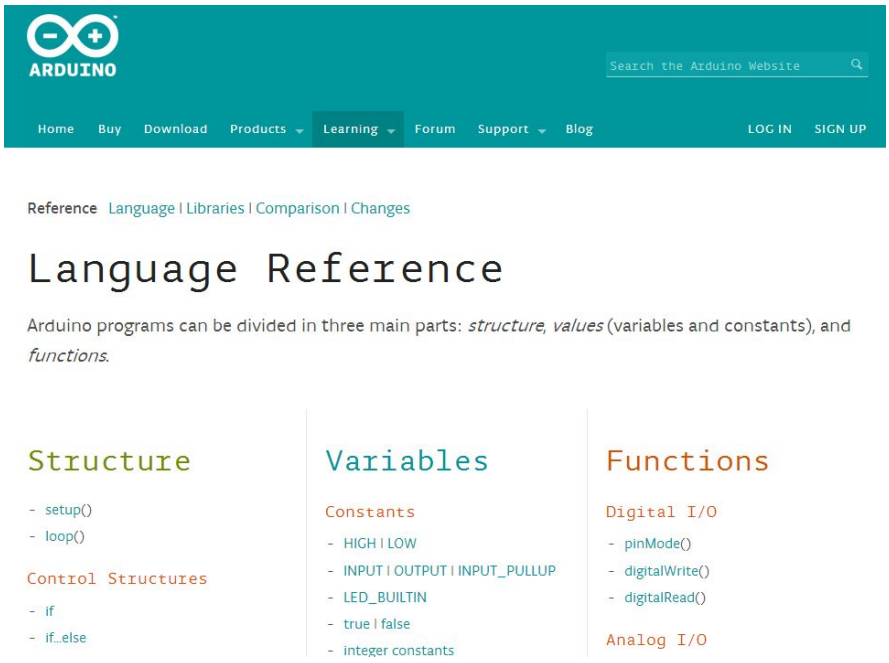


Then, connect your Arduino board to PC via USB. Install the Driver and the computer will recognize the Arduino board as a COM port:

3. Language Reference

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

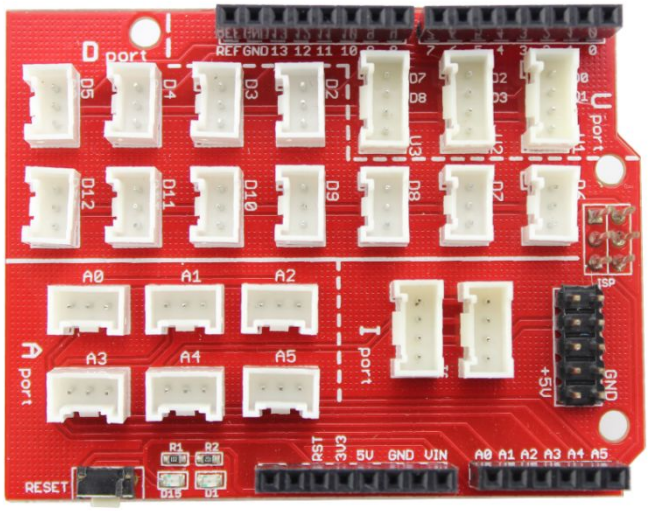
http://arduino.cc/en/Reference/HomePage



Part 1 Modules Introduction

1. Crowtail - Base Shield Introduction

The Crowtail - Base Shield is a standard IO expansion board for the 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):



11 Digital I/O ports (D2~D12) 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;

6 Analog ports (A0~A5) that have a mark of "A". Besides has the functional as digital port, these **A** ports can read the analog signal too, such as a potentiometer or light sensor;

3 UART ports that have a mark of "U". These interfaces can be used for UART communication such as the WIFI module or Bluetooth module;

2 IIC ports that have a mark of "I". These interfaces are for the IIC Communication, users can utilize 2 IIC modules at the same time;

Besides, there is also a 2x5 male connector of 5V and GND. Users can connect any electronic modules to the Base Shield with jumper wires easily.

2. Crowtail- LED

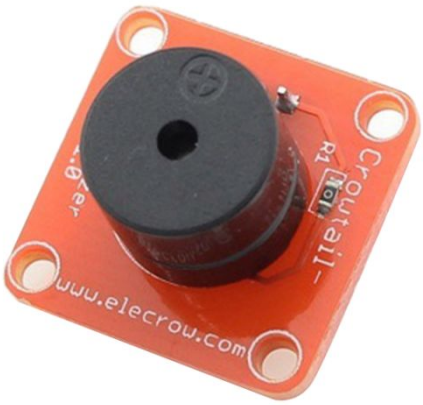
The Crowtail-LED is a simple LED indicator with resistor. The LED would be on when activated by logic HIGH, and off by logic LOW. It is the most common used indicator for human interfacing, and the most basic module for users stepping into the Arduino.

There are 2 LED modules in the Crowtail- Advanced kit, with Red /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.

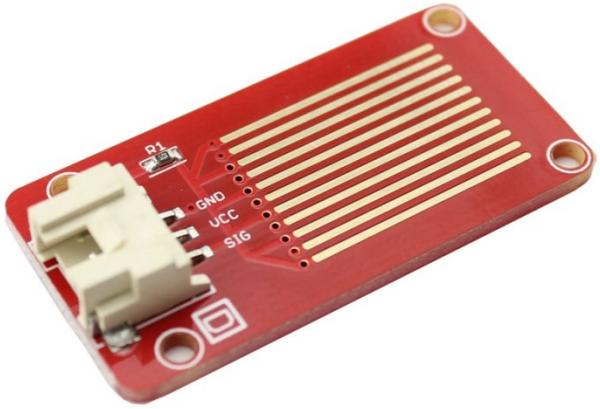
4. Crowtail- Buzzer



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 HIHG.

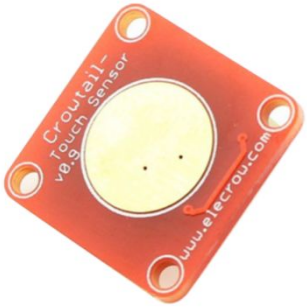
The buzzer module can be also connected to a pulse-width modulation (PWM) output to generate various of tones.

5. Crowtail- Water Sensor



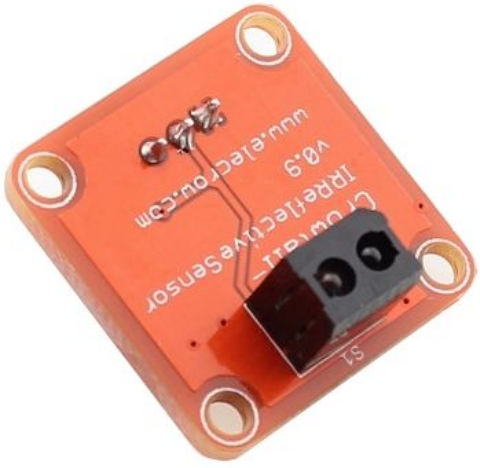
The Crowtail- water sensor detects water by having a series of exposed traces. The resistor will pull the sensor trace value high until a drop of water shorts the sensor trace to the grounded trace.

6. Crowtail- Touch Sensor



The Crowtail- Touch sensor detects the human fingers. When human finger touches, or nearby, it reports to the Arduino:”hey, it seems someone touches me”.

7. Crowtial- IR Reflective Sensor



This IR reflective module emits the infrared light and then detects if the echo received, to estimates if there is a obstacle or not.

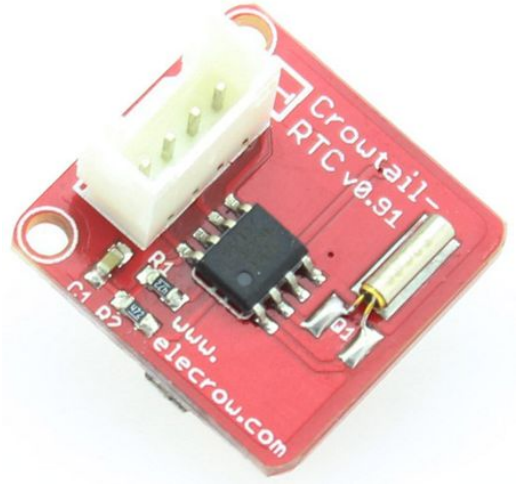
8. Crowtail- PIR Sensor

The Crowtail- PIR Motion sensor is simply to use, just connect it to Crowtail base shield and program it. When someone moves in its detecting range, the sensor outputs HIGH on its SIG PIN.

9. Crowtail- Flame Sensor

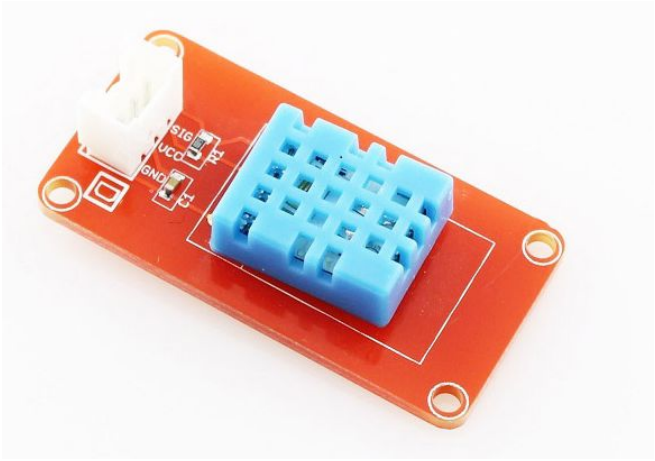
The Crowtail- Flame Sensor can be used to detect fire source or other light sources of the wavelength in the range of 760nm - 1100 nm. It is based on the YG1006 sensor which is a high speed and high sensitive NPN silicon phototransistor.

10. Crowtai- RTC



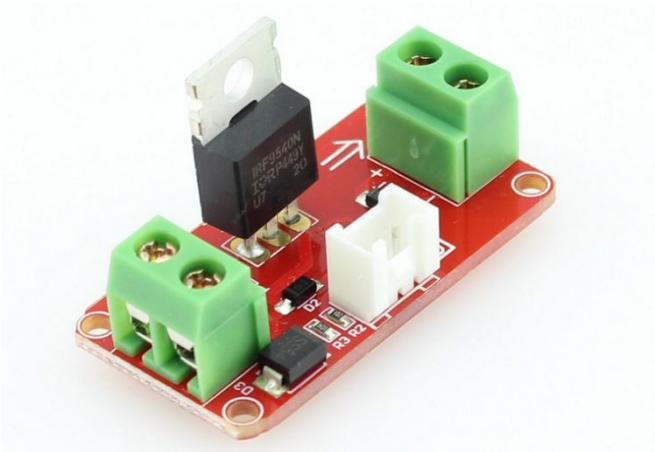
The Crowtail- RTC is based on the clock chip DS1307 which communicates with microcontrollers by I2C protocol.

11. Crowtail- Temperature& Humidity Sensor



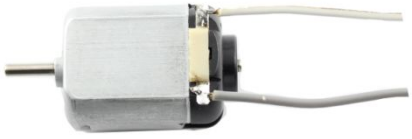
The Crowtail- Temperature& Humidity Sensor is based on DH11 which is a complex sensor with a calibrated digital signal out. It used to detect the temperature & humidity.

12. Crowtail- MOSFET



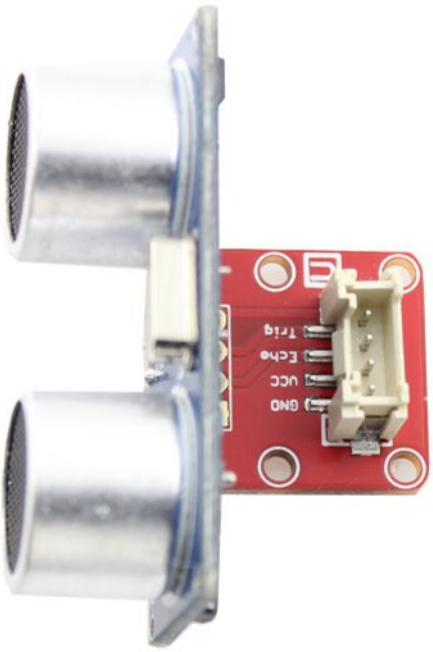
Crowtail- MOSFET enables you to control higher voltage project, such as 50V DC, MOSFET is also a kind of switch and there are two screw terminals on the board.

13. 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.

14. Crowtail- Ultrasonic Ranging Sensor

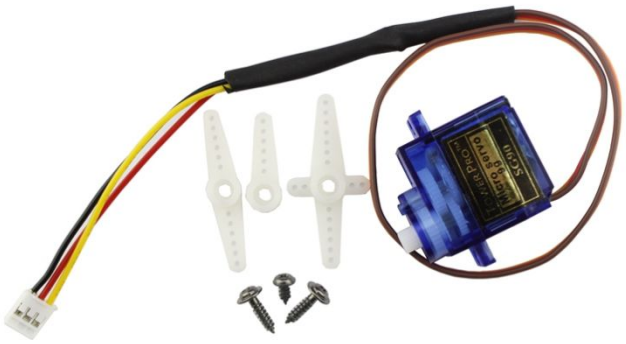


This HC-SR04 has stable performance and high ranging accuracy. It’s used for detecting the obstacle and distance.

15. Crowtail- Thumb Joystick

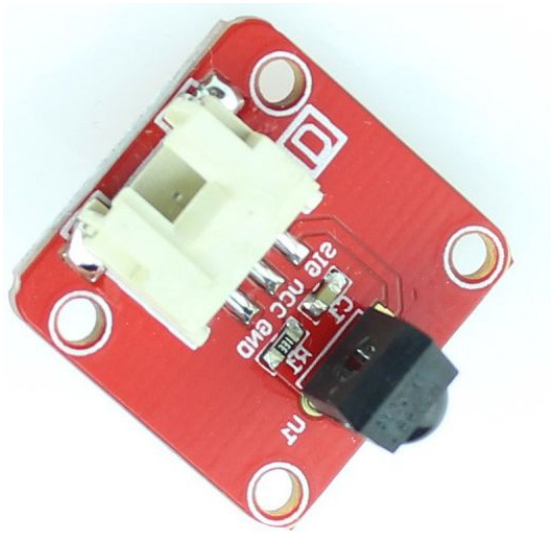
Crowtail- Thumb Joystick is very similar to the 'analog' joystick on PS2 (PlayStation 2) controllers. Two direction movements will output different analog signals as they are actually two potentiometers.

16. Crowtail- 9G Servo



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

17. 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.

18. Infrared Remote Controller

This is a cheap infrared controller, with it you can easily make remote control projects.

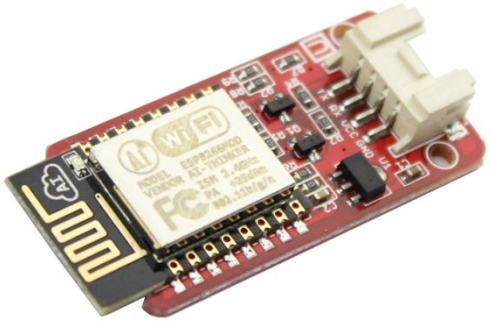
19. Crowtail- 4-Digit Display

The Crowtail- 4-digit display based on the TM1650, which only take 2 digital pin of Arduino to control the content, even the luminance of this display.

20. 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.

21. Crowtail- Serial Wifi



The Crowtail- serial Wifi based on ESP-12E which is an ultra-low power UART- WiFi module. It has excellent dimensions and ULP technology compared to other similar modules.

Part2: Crowtail Applications

Lesson1: LED Control

LED control is basic on Arduino. In this lesson, you can learn how to control the LED with button and touch sensor.

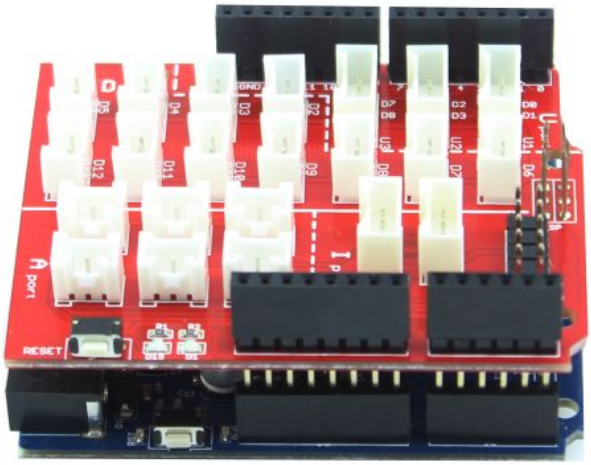
Material:

- Arduino UNO x 1
- Crowtail- Base Shield x 1

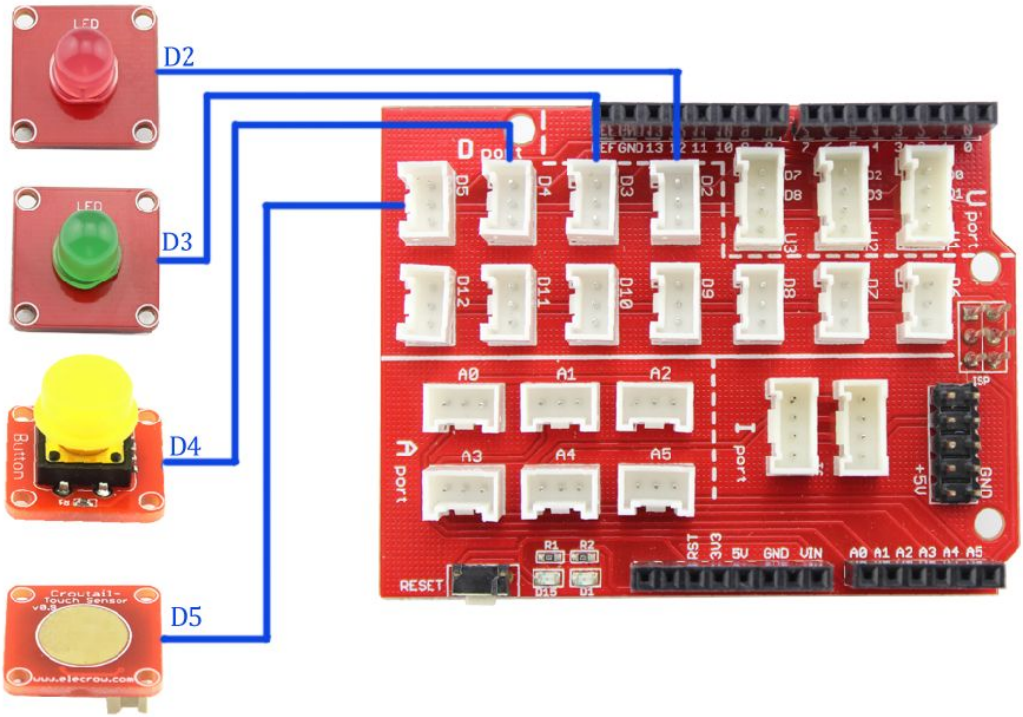
- Crowtail- LED x 2
- Crowtai- Button x1
- Crowtial- Touch sensor x1
- Crowtail- Cable x 4
- USB Cable x 1

Hardware Connection

Plug the Crowtail- Base Shield onto the Arduino or Crowduino Board:



Then connect the Crowtail- LED (Red) to the D2 port of base shield, Crowtail- LED (Green) to the D3 port, Crowtail- Button to D4 port, Crowtial- Touch sensor to D5, 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(led1, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(button, INPUT);
  pinMode(touch, INPUT);
}
```

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

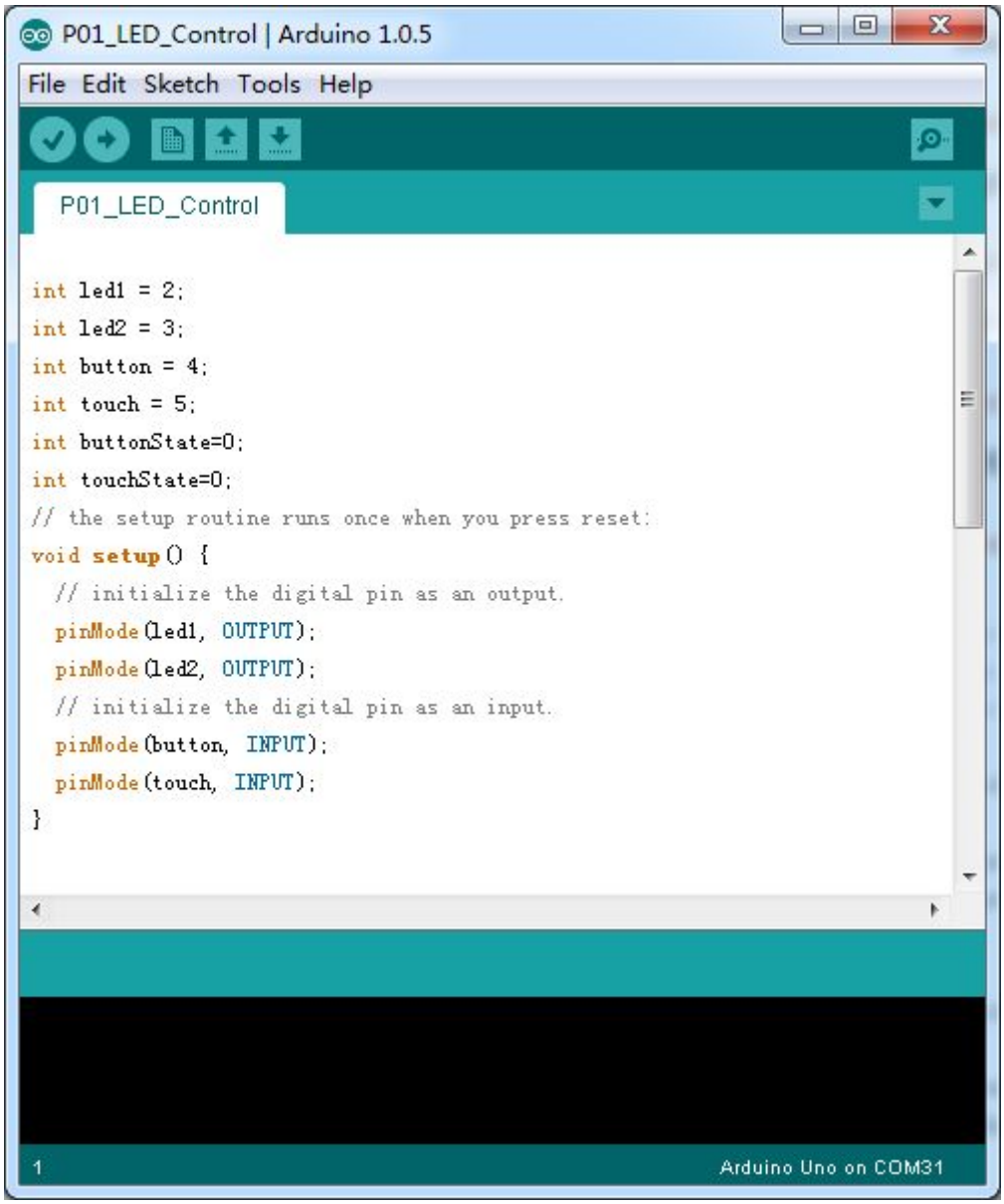
```
  // read the state of the pushbutton value:
  buttonState = digitalRead(button);
  touchState = digitalRead(touch);
```

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

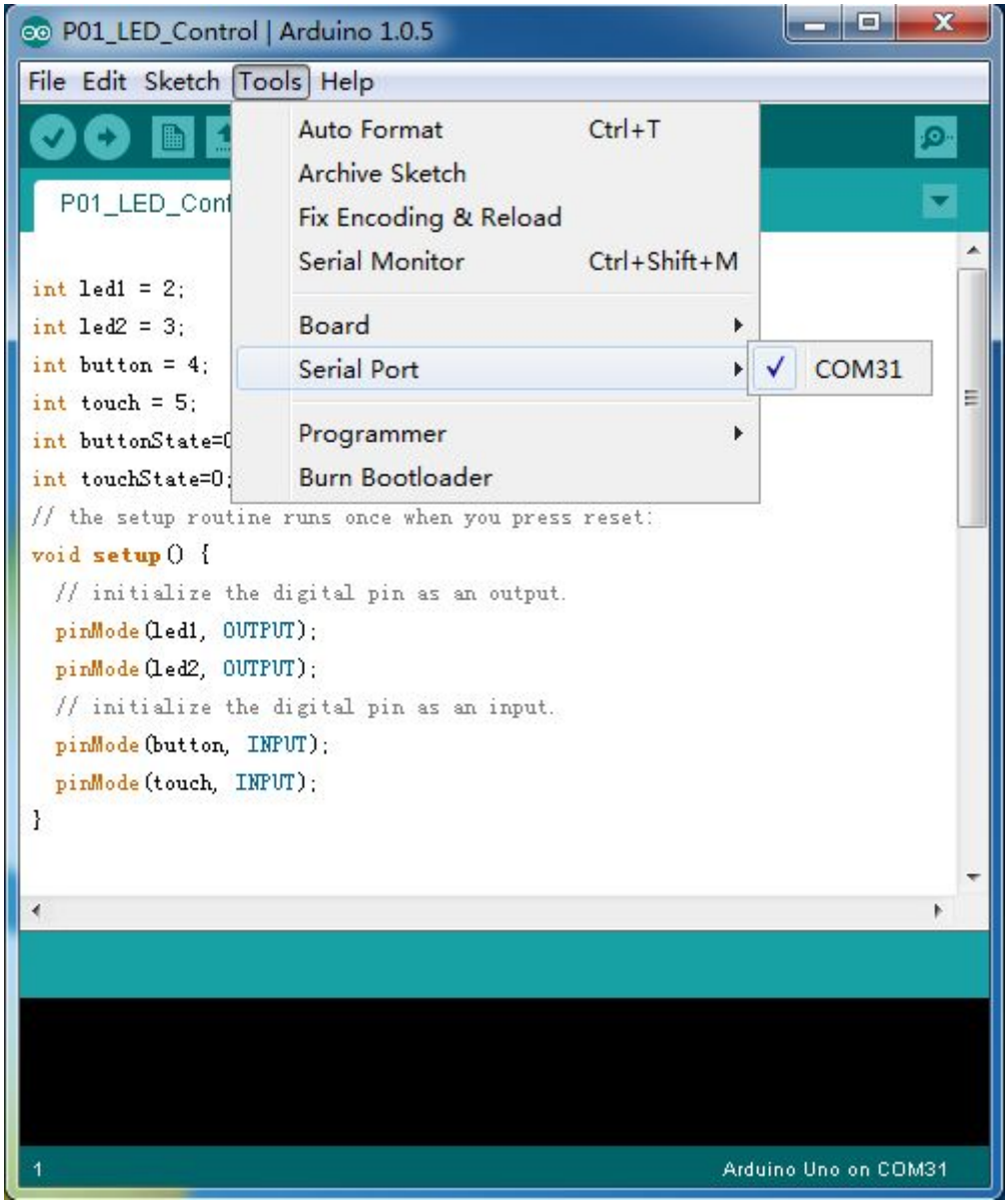
```
// check if the pushbutton or touch is pressed.
  // if it is, the buttonState and touchState is HIGH:
  if (buttonState == HIGH) {
    // turn LED1 on:
    digitalWrite(led1, HIGH);
  }
   if (touchState==HIGH){
    // turn LED2 on:
    digitalWrite(led2, HIGH);
  }
  else {
    // turn LED1 and LED2 off:
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
   }
```

Program downloading:

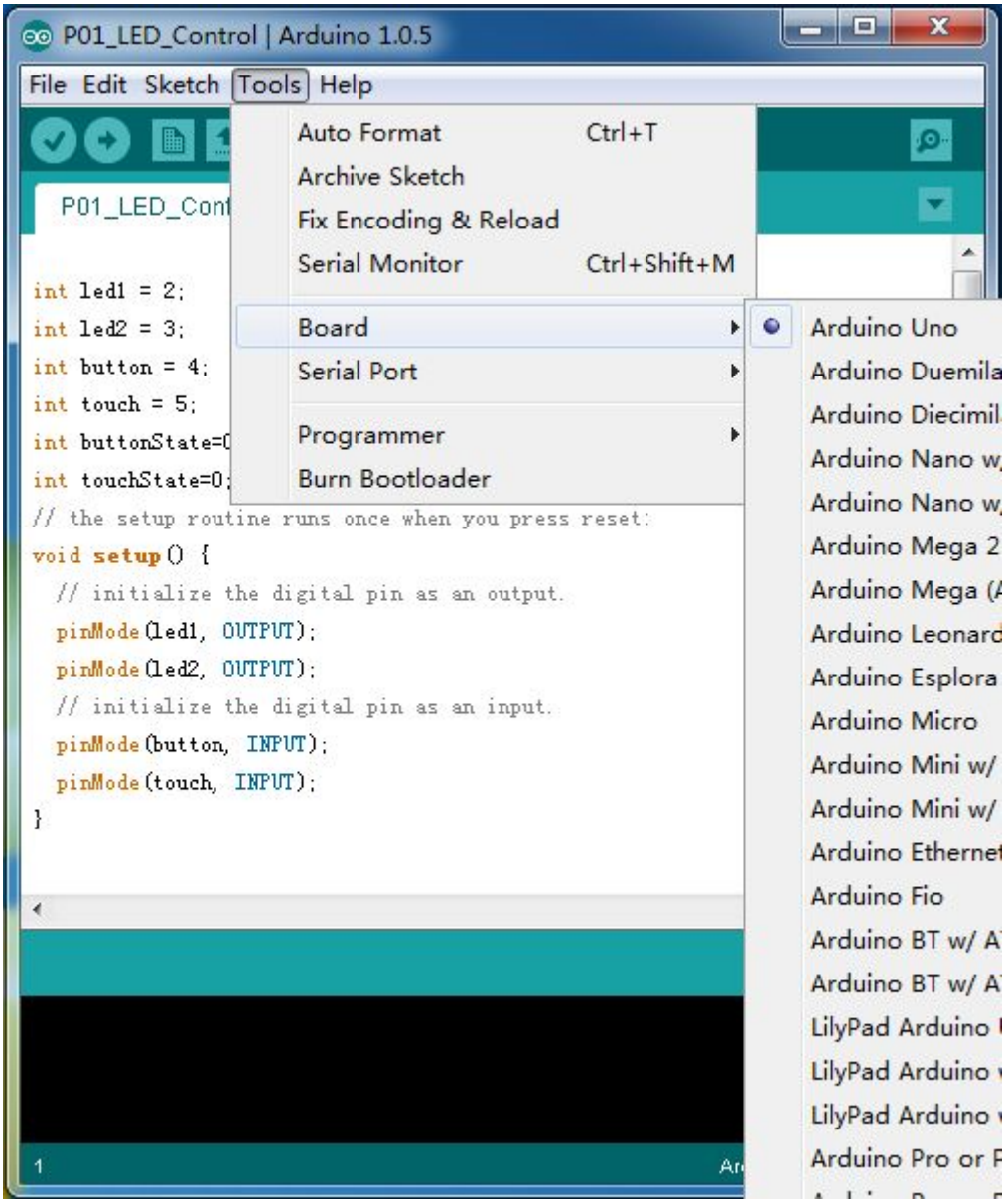
Download the **Crowtail Adcanced Kit Demo Code**, Open the *P01_LED_Control.ino* with Arduino IDE as below:



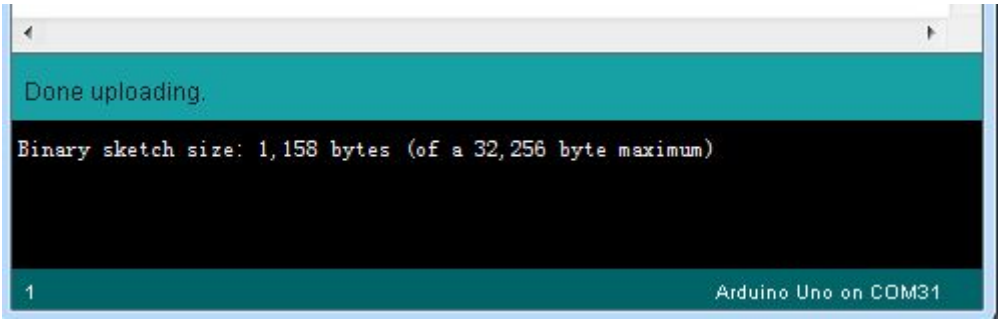
Click *Tools->Serial Port*, and choose the right serial port, which will list after the driver successfully installed:



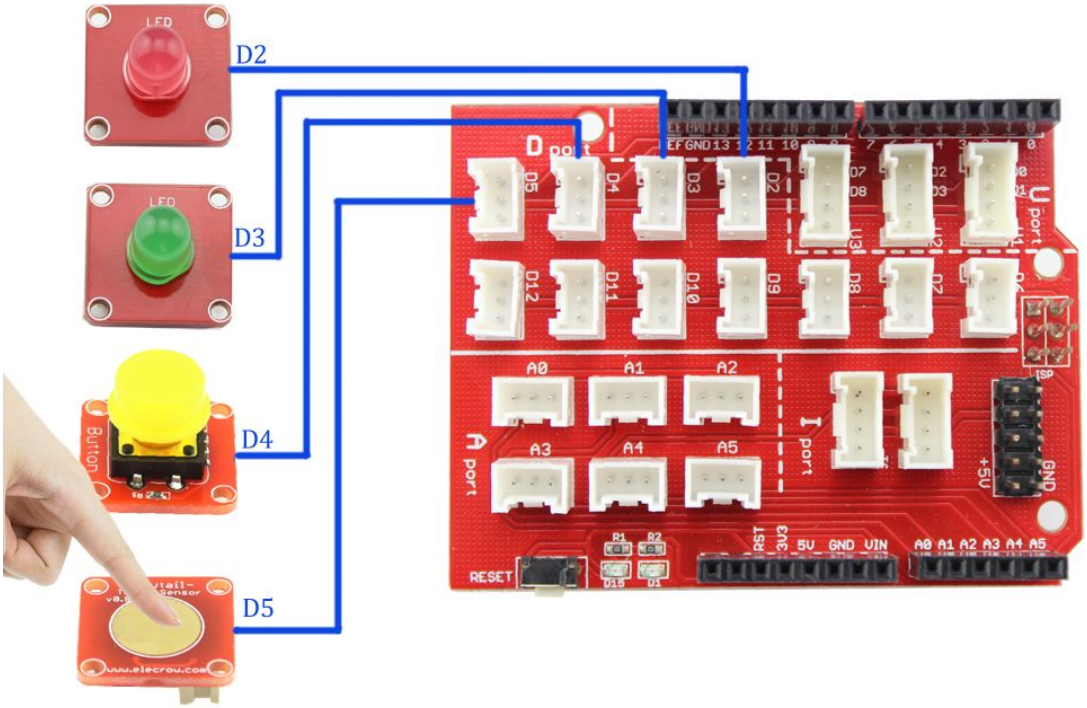
Click *Tools->Board*, choose the Arduino board you are using, such as the Arduino Uno



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 or touch the pad, the LED turns ON. Otherwise, the LED turns OFF. You can also put some things on the sensor such as a cardboard, and try again, the sensor can still sense your fingers, within about 2mm distance.



(Explore more Arduino functions here: http://www.arduino.cc/en/Reference/HomePage)

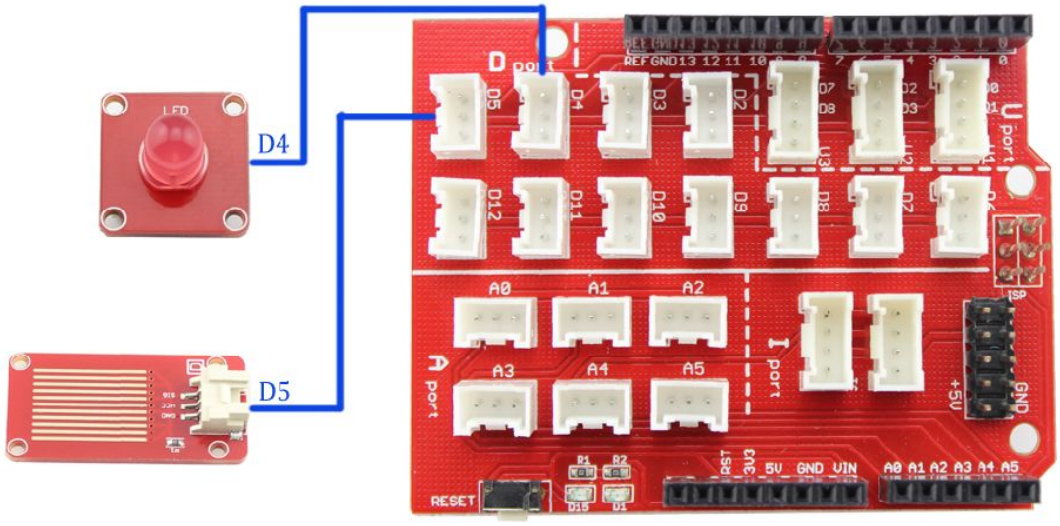
Lesson2: Raining Detecting System

In our daily life, we play computer game or watch TV all-possessed indoor, sometimes it's raining outdoor, our clothes usually be wetted, but if there is a sensor that it can remind us when it begin to rain, tragedy will not happen to us. In this lesson, we will tell you how to build it. When raining outside, the LED inside will be lighted.

Material:

- Crowtail- LED x 1
- Crowtail- Water sensor x 1

Hardware Connection



Open the Arduino Code: *P02_Raining_detect.ino*, and upload it to Arduino board. When raining, the LED turns ON.

Firmware:

Firstly, the Arduino read the status of the water sensor with the function *digitalRead(): i*n the program.

```
  // read the state of the water sensor:
waterState = digitalRead(watersensor);
```

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

```
  // if it is, the waterState is LOW, name there is raining,the LED will turn on to alarm you:
  if (waterState == LOW) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
digitalWrite(ledPin, LOW);
  }
```

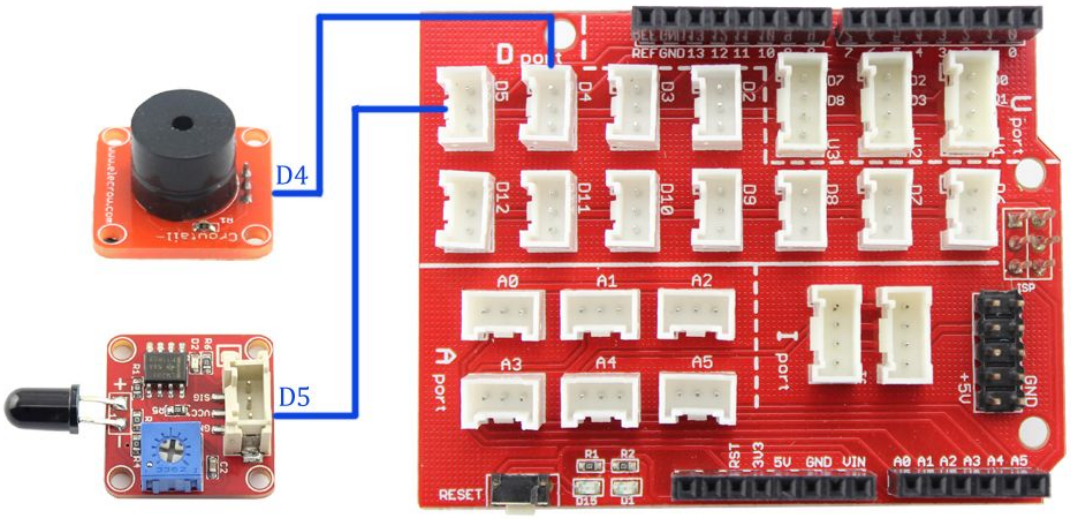
Lesson3: Flame Detecting

Have you ever seen the fire-fighting robots? Why it can find the source of fire? In this lesson, we will tell you about it and you can make a robot like that.

Material:

- Crowtail- Flame Sensor x 1
- Crowtail- buzzer x 1

Hardware Connection



Open the Arduino code *P03_Fleame_detect* and download it to the Arduino.

When the flame sensor near by the fire, the buzzer will ring out.

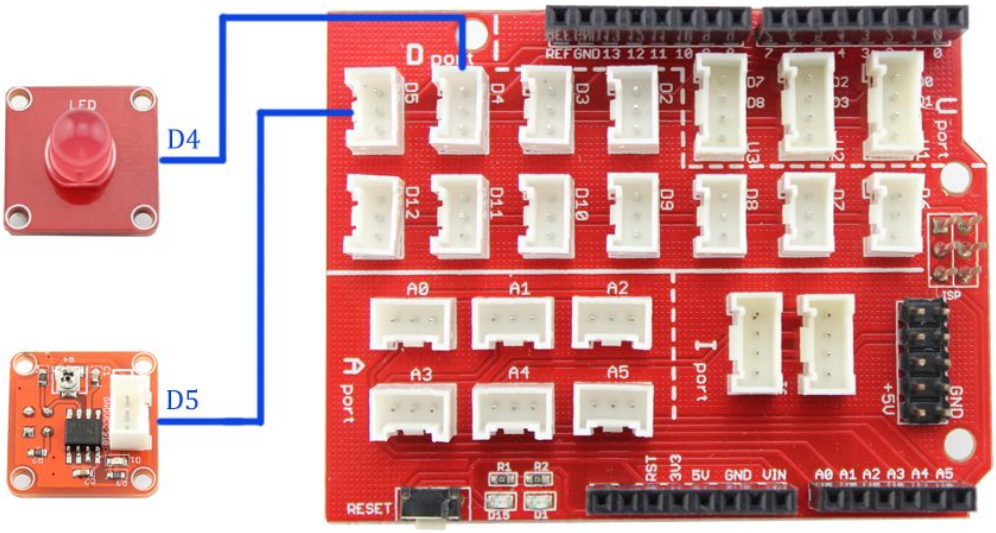
Lesson4: Tracking Experiment

In this lesson we will learn how to use the Crowtial- IR Reflective Sensor. It's use in tracking robot car.

Material:

- Crowtail- IR Reflective Sensor x 1
- Crowtail- LED x 1

Hardware Connection



Open the *P04_Line_Follower.ino*:

Firstly we define the IR Reflective Sensor and LED pins, and set a variable to store the IR Reflective Sensor status:

```
int LEDPin =4;          // the number of the vibration motor pin
int IRPin = 5;          // the number of the IR Reflective Sensor pin
int IRState = 0;            // variable for reading the IR Reflective Sensor status
```

And define the LED as OUTPUT, IR Reflective Sensor as INPUT:

```
void setup() {
  pinMode(LEDPin, OUTPUT); // initialize the LED pin as an output:
  pinMode(IRPin, INPUT);    // initialize the IR Reflective Sensor pin as an input:
}
```

If the infrared of IR Reflective Sensor be absorbed by the black line, it will output LOW, and the LED turn ON, if the IR Reflective Sensor miss the black line, and the white ground reflect the infrared back to IR Reflective Sensor, it will output HIGH, then the LED turn OFF:

```
void loop(){
IRState = digitalRead(IRPin); // read the state of theIR Reflective Sensor:
  // if it is, the IRState is HIGH:
  if (IRState == HIGH) {
    digitalWrite(LEDPin, LOW);    // turn off LED
  }
  else {
    digitalWrite(LEDPin,HIGH);    // turn on LED
}
}
```

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

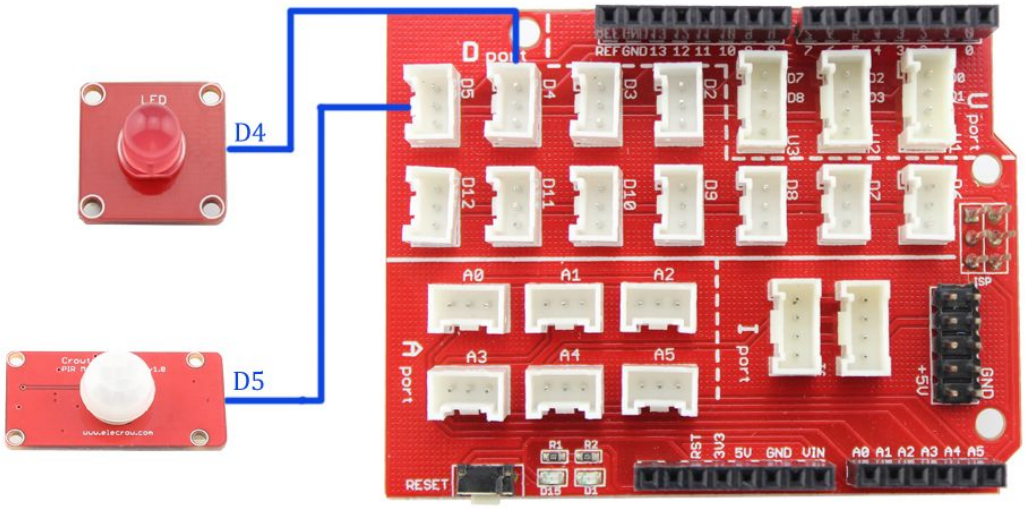
Lesson5: Stair Smart Light

As everyone knows, when we go up the stairs, the lamp of the stair automatic light up. Why it can do that? Mostly they install a PIR sensor, it can detect somebody come in then light up the LED.

Material:

- Crowtail- PIR motion sensor x 1
- Crowtail- LED x 1

Hardware Connection



Firmware

Open the *P05_Simulation_stair_light.ino*

In the program, the Arduino firstly detects whether there somebody come in:

```
isPeopleDetected();
```

And if somebody come in, the Arduino board set a logic HIHG to the LED, turn on the LED:

```
If (isPeopleDetected())//if it detects the moving people?
        turnOnLED();
else
        turnOffLED();
```

After successfully upload, you can adjust the potentiometers to change the detecting range.

Lesson6: Electric Watch

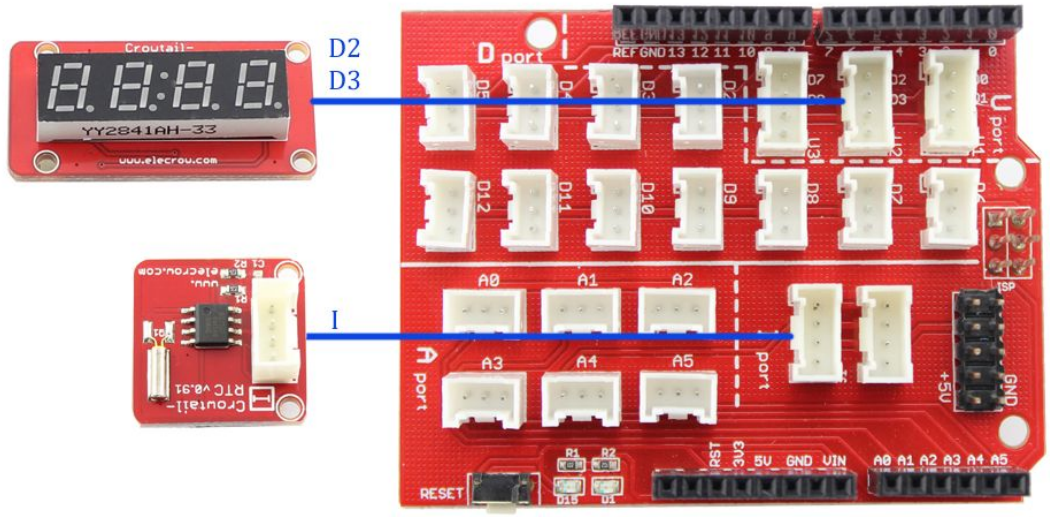
Do you want to make an electric watch? It is funny and useful. In this lesson, we will tell you how to make it, now let's begin.

Material:

- Crowtail- RTC x 1

- Crowtail- 4- Digital display x 1

Hardware Connection



Firmware:

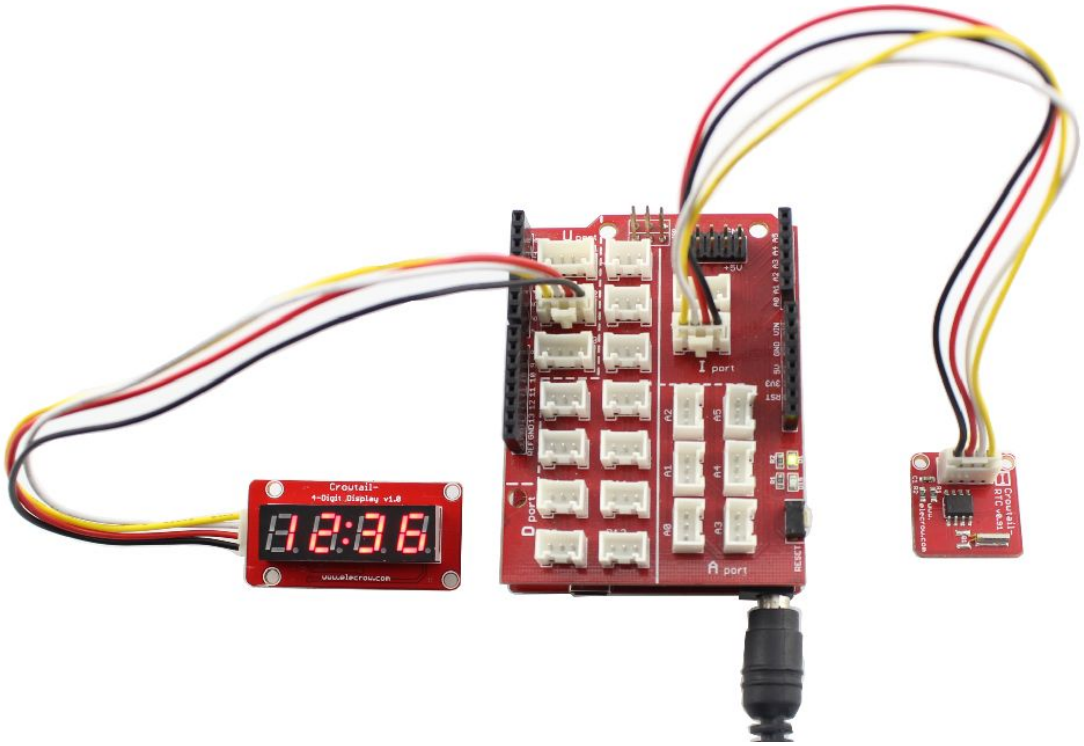
For this application, an Arduino library "**RTC**" and **"TM1650"**is needed. Firstly, copy the folder of "**RTC**" and **"TM1650"** to the Arduino library file: *… \Arduino\libraries*

Open the Arduino code *P06_RTC.ino*

In the function *Loop()*, Arduino clear the data of 4- digit display and get the time from RTC then display in the 4–digit display:

```
DigitalLED.clearDisplay();//clear the display
    DateTime now = RTC.now(); //get current time
    leddisplaymd(1,now.month());   //display month
    leddisplaymd(2,now.day());      // display day
    delay(3000);
    DigitalLED.clearDisplay();        //clear the display
    leddisplayhm(1,now.hour());       //display the hour
    leddisplayhm(2,now.minute());     // display the minute
    delay(3000);
```

Upload the program into Arduino or Crowduino, you can see the time on the digit display.



Lesson7: Temperature& Humidity Detecting System

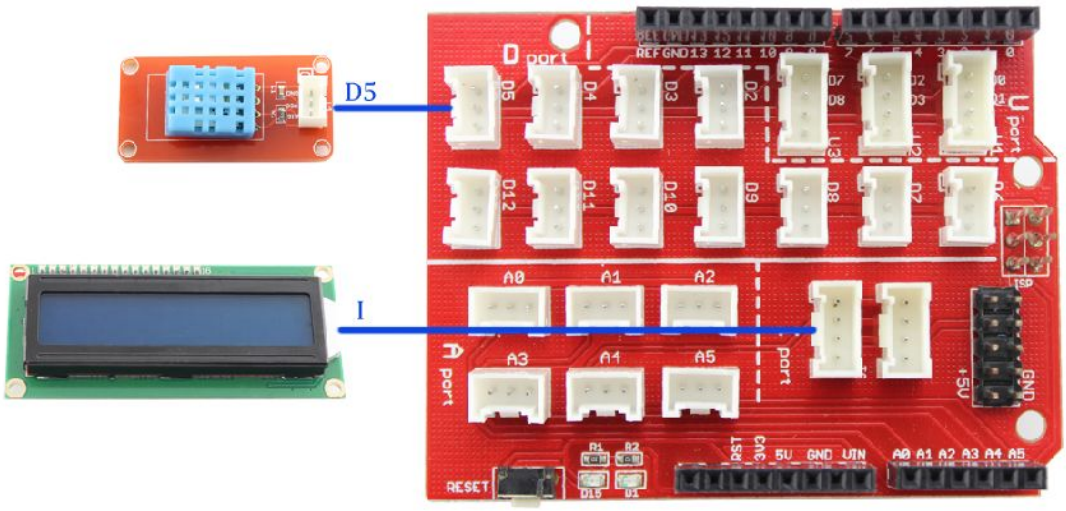
In our daily life, temperature and humidity are important information and it's closely related to our health. In this lesson, we will learn how to get the temperature and humidity digital data with DH11 and display it in the LCD.

Material:

- Crowtail- Temperature& Humidity sensor x 1
- Crowtail- IIC LCD x 1

Hardware Connection

Connect the temperature & humidity sensor to one of the "**D"** ports, and IIC LCD to **I** ports:



Firmware

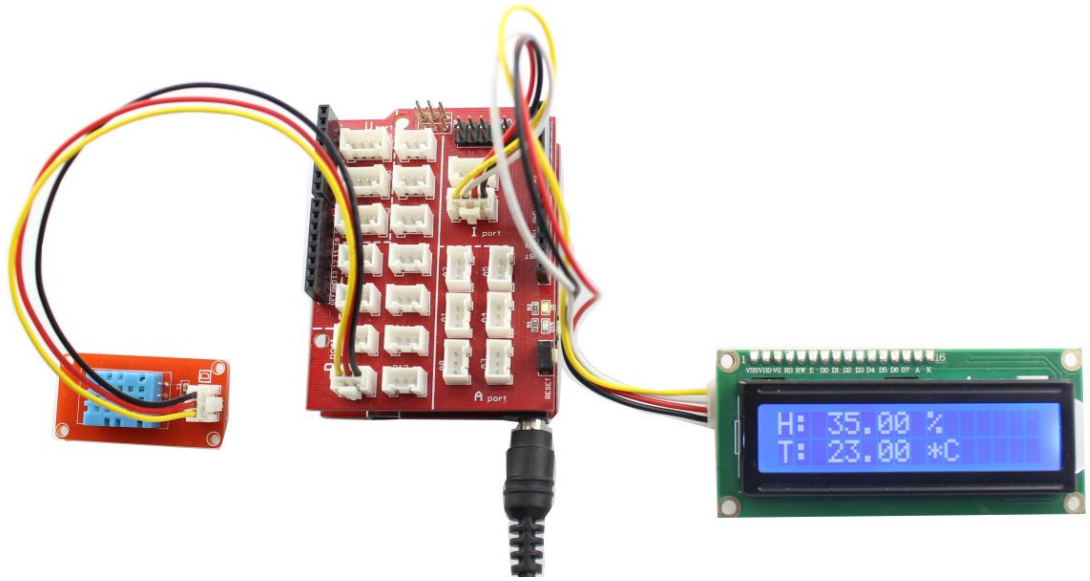
For this application, an Arduino library "**DHT**" and **"LiquidCrystal"** are needed. Firstly, copy the folder of "**DHT**" and **"LiquidCrystal"** to the Arduino library file: *… \Arduino\libraries*

Open the Arduino code *P07_Temperature_Humidity_display.ino*

After getting the temperature and humidity data from DHT11, the Arduino send data to IIC LCD and display them on it.

```
void loop() {
      float h = dht.readHumidity();
       float t = dht.readTemperature();
       // check if returns are valid, if they are NaN (not a number) then something went wrong!
       if (isnan(t) || isnan(h)) {
                lcd.println("Failed to read from DHT");
       } else {
                 // 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("H: ");
                 // print the number of fist since reset:
                 lcd.print(h);
                 lcd.print(" %");
                 // set the cursor to column 0, line 1
                 // (note: line 1 is the second row, since counting begins with 0):
                 lcd.setCursor(0, 1);
                 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 and humidity in the LCD.



Lesson8: Ultrasonic Ranging System

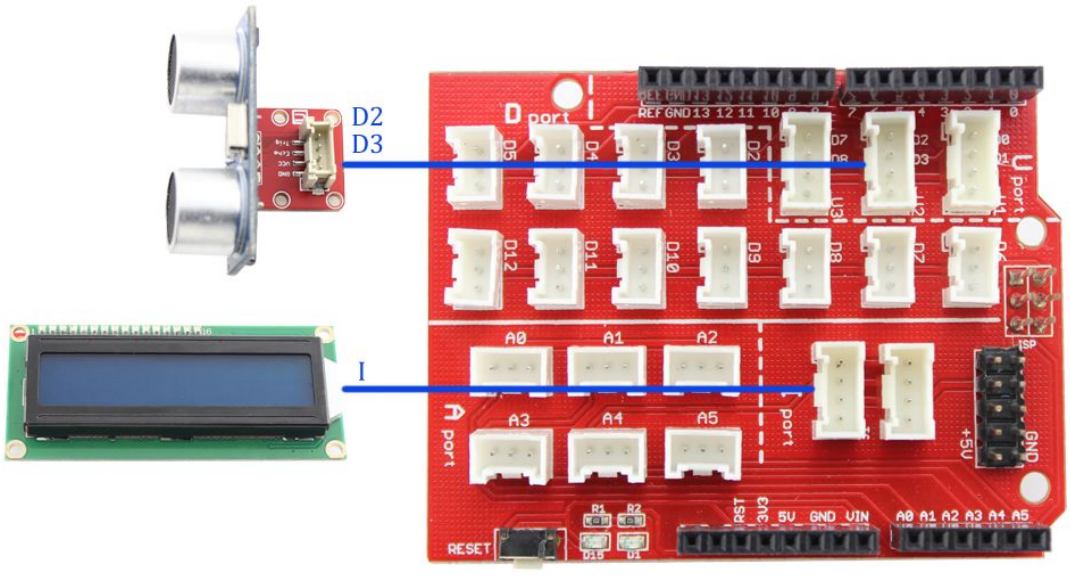
In our daily life, sometimes we want to measure the distance but we don't have ruler in hand In this lesson, we will tell you how to detect the distance with ultrasonic and display with IIC LCD.

Material:

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

Hardware Connection

Connect the Crowtail- Ultrasonic sensor to "**U**" port such as D2/D3, and the IIC LCD to the "**I**" port as below:



Firmware

For this application, an Arduino library "**Ultrasonic**" is needed. Firstly, copy the folder of "**Ultrasonic**" 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;
int Pre_Distance=0;
void setup() {
      // set up the LCD's number of rows and columns:
lcd.begin(16, 2);
}
```

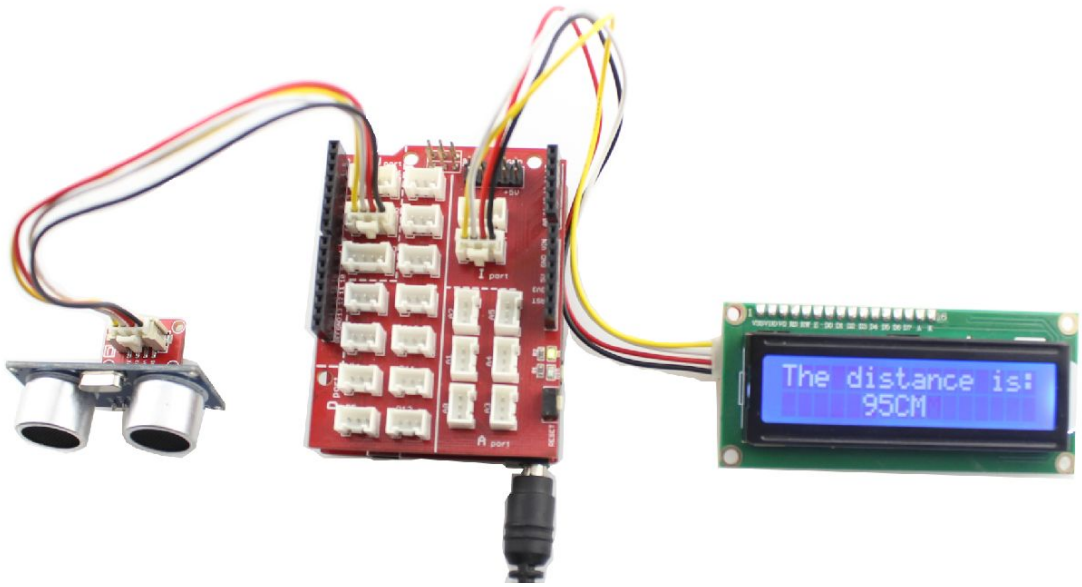
And get the distance:

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

If the Current distance different the pre-distance, we display it on 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: PWM Control

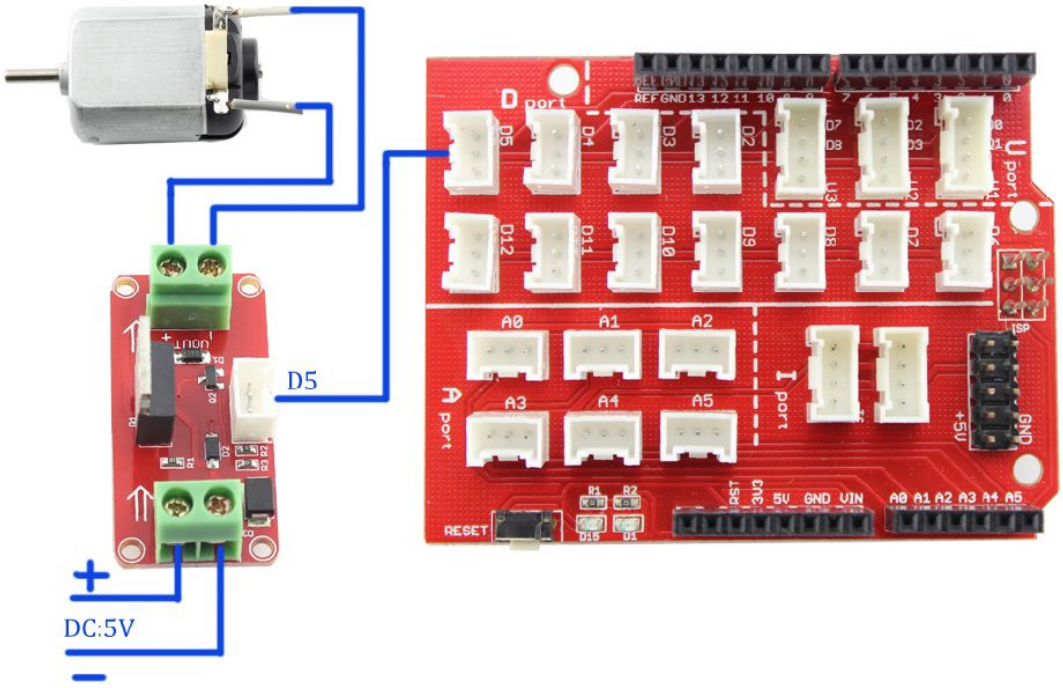
Pulse-width modulation (PWM) is a modulation technique used to encode a message into a pulsing signal. Although this modulation technique can be used to encode information for transmission, its main use is to allow the control of the power supplied to electrical devices, especially to inertial loads such as motors. In this lesson, we will tell you how to control a motor with PWM.

Material:

- Crowtail- MOFET x 1
- DC motor x 1

Hardware Connection

Connect the Crowtail- MOSFET to Port "**D**", we provide power for it with external power, but if your device's working current less than 300mA, crowduino can fully support it with no extra power . As below:



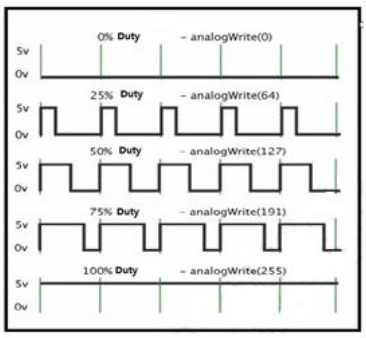
Firmware

And then open the ***P09_PWM_Control.ino***

The PWM wave can be only generated with such pins: 3/5/6/9/10/11, for the Arduino Uno or Crowduino, with the function *analogWrite()*:

```
analogWrite(motorPin, i);       // PWM control the speed of LED
```

The MOSFET control the speed of motor with PWM wave, the higher duty of the logic HIGH, the faster the motor would be.



Upload the program to Arduino & Crowduino, you can see the speed of the motor changes with the PWM.

==========

Lesson10: Servo Control

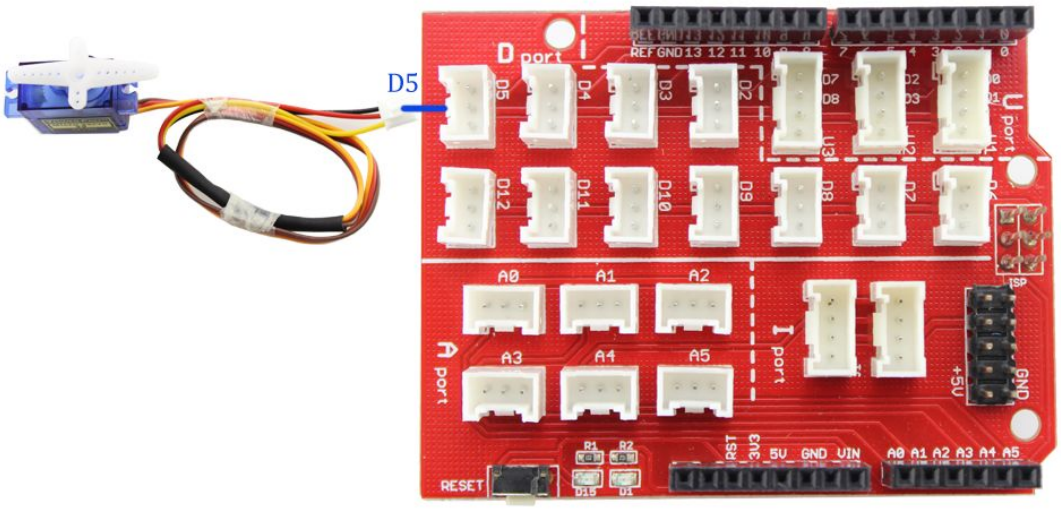
Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. In this lesson, we will tell you how to control a servo.

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

Lesson11: Joystick Servo Control

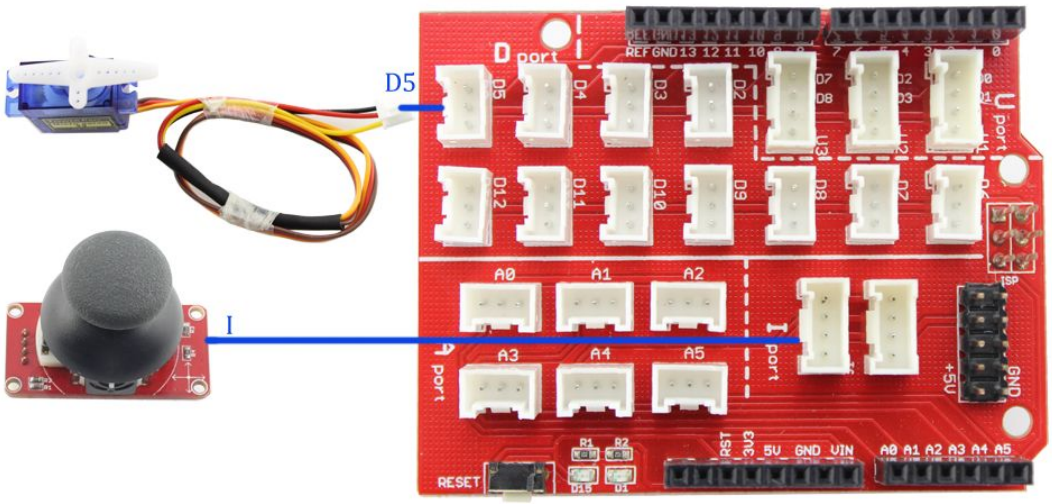
In this lesson, we will tell you how to control a servo with joystick.

Material:

- Crowtail- Servo x 1
- Crowtail- Thumb Joystick x1

Hardware Connection

Connect the Crowtail- servo to Base Shield Port "**D**", and the Crowtail- Joystick to "**I**" Port. As below:



Firmware

Open the Arduino Code*P11_Joystick_Servo_Control.ino*

In the function *setup()*, firstly we initialize the servo, connect the servo to the pin 5 and set the servo at 90 degrees:

```
void setup()
{
    myservo.attach(5);    // attaches the servo on pin 5 to the servo object
    myservo.write(90);      // set the servo at 90 degreed
}
```

In the main loop, read the analog value from the joystick, we just use one axis of the

joystick(X axis).

```
pos=analogRead(A5);    //read the pos value
```

Then the value is mapped so the range is now between 0 and 180, which will correspond to the degree angle of the servo arm:

```
int angle = map(pos,200,800,0,180); //Map the cvalues from 0 to 180 degrees
```

Then you take your servo object and write the appropriate angle, in degrees, to it (the angle must be between 0 and 180 degrees):

```
myservo.write(angle);    //write the angle to the servo
```

Finally, a delay of 15ms is programmed to allow the servo time to move into position:

```
delay(15);      // Delay of 15ms to allow servo to reach position
```

After successfully uploading the code, you can control the servo with the joystick.

Lesson12:IR Control

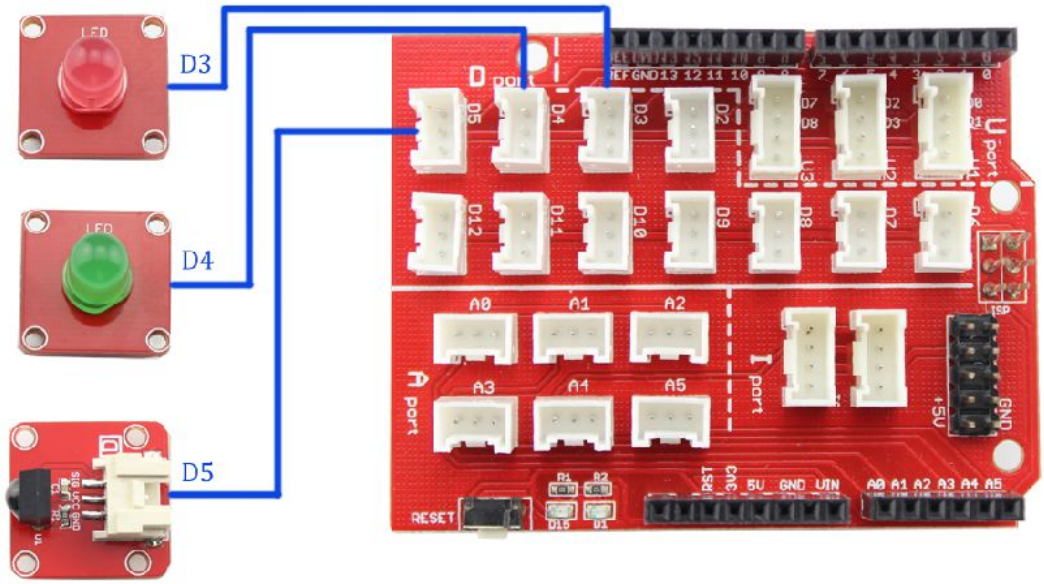
The cheapest way to remotely control a device within a visible range is via Infra-Red light. Almost all audio and video equipment can be controlled by it nowadays, due to this quite cheap components are be widely used, thus making it ideal for us hobbyists to use IR control for our own projects. In this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

Material:

- Crowtail- LED x 2
- Crowtail- IR Receiver x 1
- Infrared Remote Controllerx1

Hardware Connection

Connect the LED and IR Receiver to "**D**" port such as D3, D4, D5 as below:



Firmware

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

Open the Arduino Code: *P12_IR_Control.ino*

In the firmware, we define the Pins that we connected, and some Variable to store the data received from the sensor:

```
int LED_R=3;
int LED_G=4;
int RECV_PIN = 5;
IRrecv irrecv(RECV_PIN);
decode_results results;
uint16_t lastCode = 0;
```

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

```
irrecv.enableIRIn(); // Start the receiver
pinMode(LED_R, OUTPUT);
pinMode(LED_G, OUTPUT);
```

In the main loop, detect and read the Infrared remote control value from the IR Receive rand then Arduino control the LED according to value of the Infrared remote control:

```
void loop() {
  if (irrecv.decode(&results)) {
    uint16_t resultCode = (results.value & 0xFFFF);
   /* The remote will continue to spit out 0xFFFFFFFF if a
      button is held down. If we get 0xFFFFFFF, let's just
      assume the previously pressed button is being held down */
    if (resultCode == 0xFFFF)
      resultCode = lastCode;
    else
      lastCode = resultCode;
    // This switch statement checks the received IR code against
    // all of the known codes. Each button press produces a
    // serial output, and has an effect on the LED output.
    switch (resultCode)
    {
      case BUTTON_0:
        Serial.println("0");
        break;
      case BUTTON_1:
        Serial.println("1");
        digitalWrite(LED_R, HIGH);
        break;
      case BUTTON_2:
        Serial.println("2");
        digitalWrite(LED_G, HIGH);
        break;
      case BUTTON_3:
        Serial.println("3");
        digitalWrite(LED_R, LOW);
        digitalWrite(LED_G, LOW);
```

```
            break;
        case BUTTON_4:
            Serial.println("4");
            break;
        case BUTTON_5:
            Serial.println("5");
            break;
        case BUTTON_6:
            Serial.println("6");
            break;
        case BUTTON_7:
            Serial.println("7");
            break;
            case BUTTON_8:
            Serial.println("8");
            break;
            case BUTTON_9:
            Serial.println("9");
            break;
        default:
            Serial.print("Unrecognized code received: 0x");
            Serial.println(results.value, HEX);
            break;
    }
    irrecv.resume(); // Receive the next value
  }
}
```

Note: In this demo code, we just use the BUTTON_1,BUTTON_2 and BUTTON_3.

After successfully upload the code, when we press the button 1 on the Infrared Remote Controller, Arduino will light up the red LED, and press the button 2 will light up the green LED. Finally when you press the button 3, you will turn off all the LEDs. Let's have a try.

Lesson13: ESP8266 TCP Server

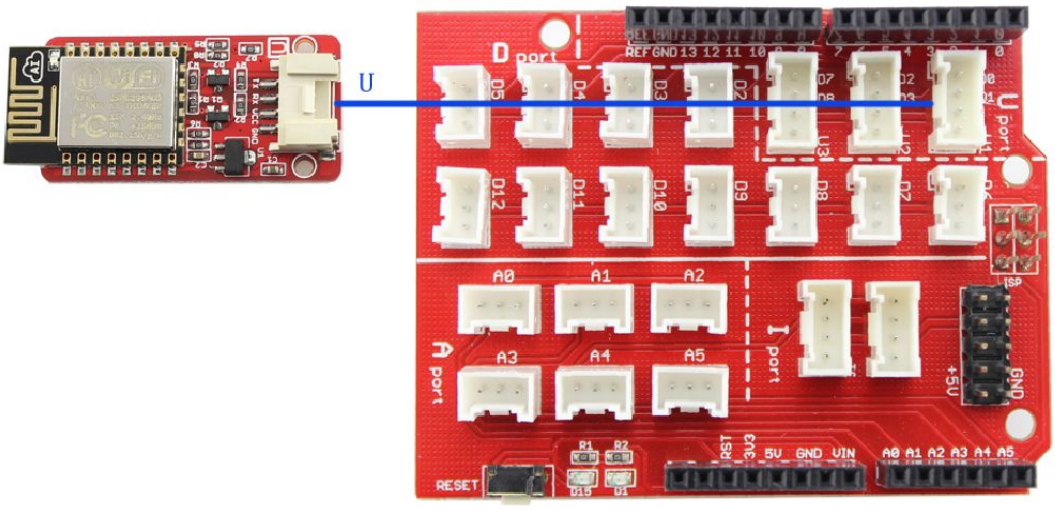
Now is an era of "Internet of things", so we have to keep up with the pace of the time. In the lesson, we will tell you how to use the serial Wi-Fi, which is based on ESP8266-12E. It is a complete and self-contained Wi-Fi network solution that can carry software applications, or through another application processor uninstall all Wi-Fi networking capabilities.

Material:

- Crowtail- Serial Wi-Fi x 1

Hardware Connection

Connect the Crowtail- Serial Wi-Fi to Base Shield Port "**U**". As below:



Firmware

Open the Arduino code*P13_ESP8266_TCPServer.ino*

In the function *setup()*, we initialize the Wi-Fi as a TCP Server:

```
Serial.begin(115200); // your esp's baud rate might be different
sendData("AT+CWMODE=3\r\n",1000,DEBUG); // configure as access point
sendData("AT+RST\r\n",2000,DEBUG); // reset module
sendData("AT+CIPMUX=1\r\n",1000,DEBUG); // configure for multiple connections
sendData("AT+CIPSERVER=1\r\n",1000,DEBUG); // turn on server,the default port is 333
```

In the main loop, Arduino check whether somebody visit the server, if it was visited, you can received “Hello World!” :

```
if(Serial.available()) // check if the esp is sending a message
 {
   if(Serial.find("+IPD,"))
   {
    delay(1000);

    int connectionId = Serial.read()-48; // subtract 48 because the read() function returns
                                         // the ASCII decimal value and 0 (the first decimal number) starts at 48
    String webpage = "<h1>Hello World!</h1>";

    String cipSend = "AT+CIPSEND=";
    cipSend += connectionId;
    cipSend += ",";
    cipSend +=webpage.length();
    cipSend +="\r\n";
    sendData(cipSend,1000,DEBUG);
 sendData(webpage,1000,DEBUG);
    String closeCommand = "AT+CIPCLOSE=";
    closeCommand+=connectionId;
    closeCommand+="\r\n";
```

```
        sendData(closeCommand,3000,DEBUG);
    }
}
```

The default IP is:192.168.4.1:333. After successfully upload the code. You can visit the TCP Server as below: