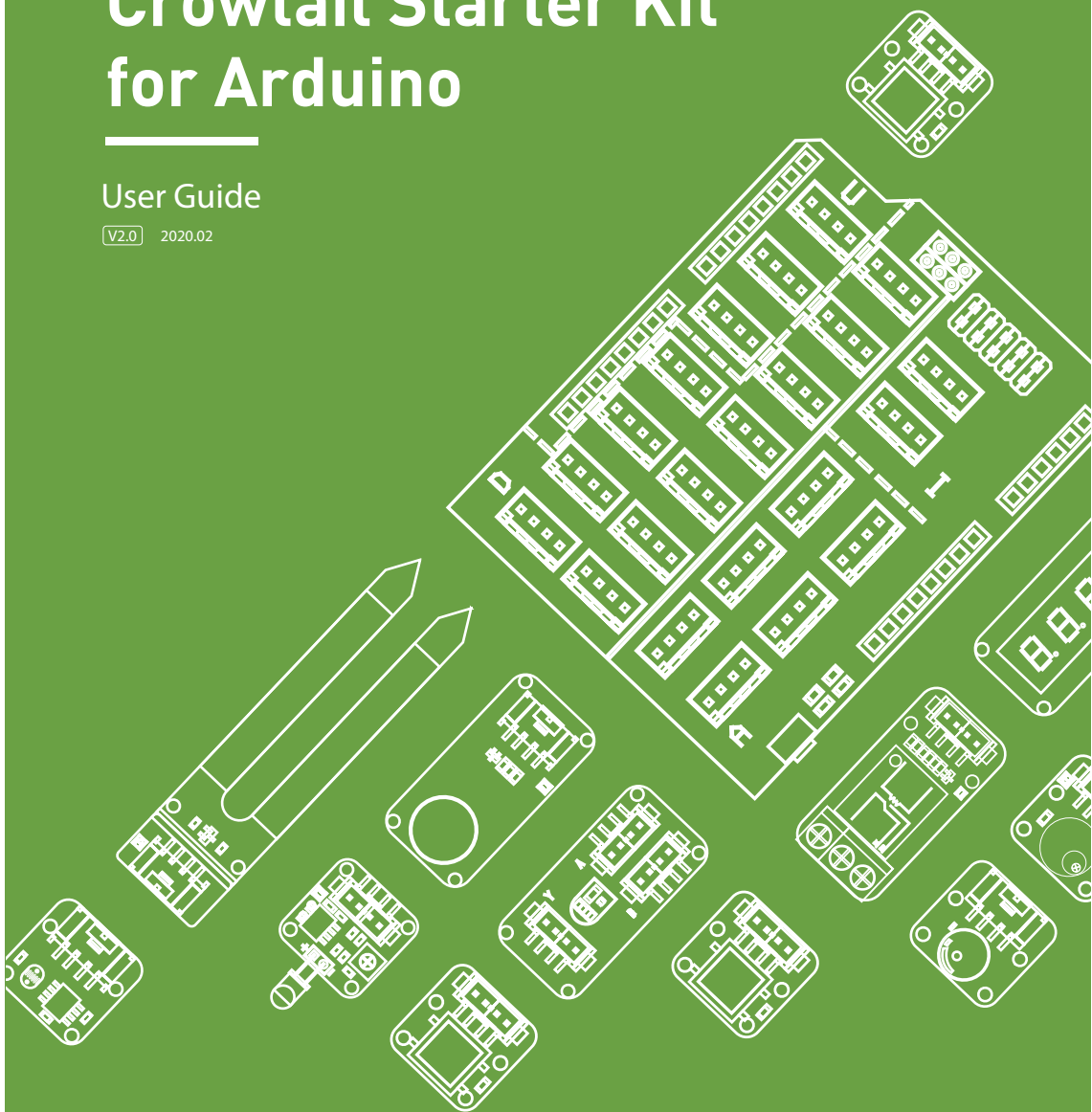


Crowtail Starter Kit for Arduino

User Guide

V2.0 2020.02



Content

Instruction	01
Crowduino Uno	02
Crowtail	02
Modules List	04
Getting Started	05
• Installing IDE	05
• How to upload program	08
• Add Libraries and Open Serial Monitor	09
Lessons	11
• Lesson 1 – LED Control	11
• Lesson 2 – Button Control LED	12
• Lesson 3 – Switch LED	14
• Lesson 4 – Touch reminder	15
• Lesson 5 – Tilt Alarm	17
• Lesson 6 – Flame Alarm	18
• Lesson 7 – Breathing LED	20
• Lesson 8 – Moisture Monitor	22
• Lesson 9 – Sound Reminder	23
• Lesson 10 – Intelligent street light	25
• Lesson 11 – Overheat alarm	26
• Lesson 12 – Logic NOT	28
• Lesson 13 – Logic OR	29
• Lesson 14 – Logic AND	31
• Lesson 15 – Automatic Control System	32
• Lesson 16 – Show number on 4-Digital display	34
• Lesson 17 – Traffic light	36
• Lesson 18 – Smart corridor light	37
• Lesson 19 – Plant watering reminder system	39
• Lesson 20 – Brightness display	41

Instruction

Welcome to the Crowtail-Starter kit for Arduino user guide. Here we are going to get started with some of the basics sensors and hardware with Arduino or Crowduino. Don't worry, this kit contains 20 interesting and instructive tutorials, from simple to difficult, step by step to familiarize you with electronic modules, exercise your logical thinking and develop your ideas, and finally use electronic modules through programming. From the installation of the Arduino to the introduction of the module to the functions used in the module programming, each step is so clear that even if you are a beginner, you can quickly master the basic Arduino programming knowledge through this starter kit.

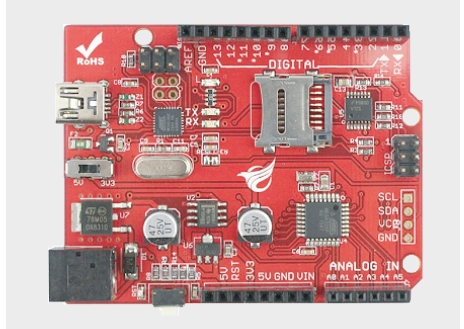
The Crowtail-Starter kit for Arduino includes 22 electronic modules, each module has its own feature and functions. Each module is carefully selected for beginners of Arduino, and is ideal for introductory learning and inspiration, such as logic module is to teach beginners programming logic thinking, light sensor is used to teach beginners the principle and method of lightcontrol in real life. In short, by learning this kit, you will learn some basic sensor principles and uses, digital, analog signal knowledge, analog-to-digital conversion knowledge, programming logic knowledge, the use of some complex electronic modules, etc. What's more important is that you will appreciate the magic of electronic modules and the fun of Arduino programming and further develop your logical thinking skills.

For the programming part, we will use the Arduino software to program. Arduino is an easy-to-use open source electronic prototyping platform. It is one of the most popular open source hardware in the world, including hardware (various models of Arduino board) and software (Arduino IDE), is the best choice for people who want to learn programming and hardware knowledge!

Crowduino Uno

The Crowduino Uno-SD mainboard is a microcontroller board that completely compatible with the Arduino UNO. It is based on the Atmega328P, which is widely also used in the Arduino Uno and other Arduino compatible boards.

The Uno-SD board has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, a reset button, and of course a Mini USB cable, simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



Compared to the original Arduino Uno, the Crowduino Uno SD board has the following upgrades:

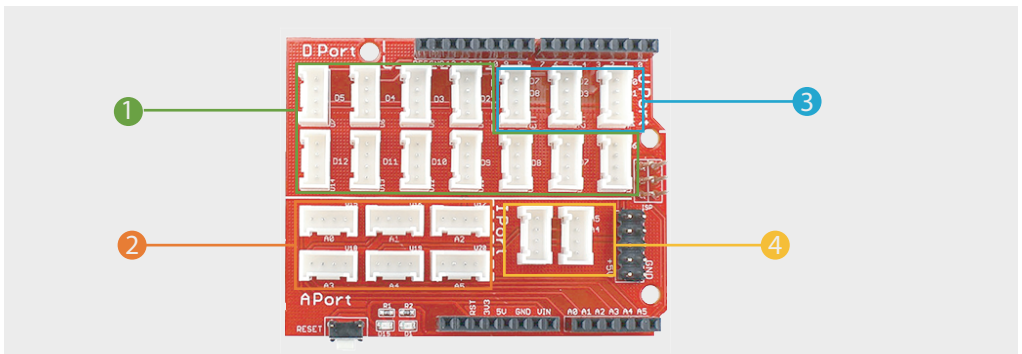
1. Reset button on the left side, which makes it more convenient to reset the system.
2. Updated power circuit. With the selector, users can select the working voltage, 3.3V or 5V, which makes the Uno-SD board compatible with more sensors that use the 3.3V logic level communication.
3. Uno-SD board uses the Mini USB connector, to avoid the potential risk of connecting to above shield.
4. On-board SD card slot. Makes this board more convenient for applications such as data logging/environment monitoring.

Crowtail

Crowtail is a series of products that we made to solve the messy jumper when connecting electronic circuits. It consists of a Base Shield and some basic Crowtail modules, helps you creating small, simple, and easy-to-assemble circuits.

Crowtail – Base Shield

The Crowtail-Base Shield is a standard IO expansion board for the Arduino. It regulates 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):

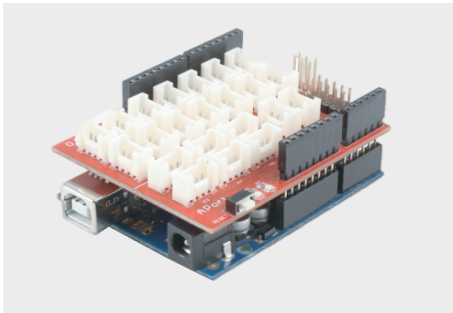


- ① 11 Digital I/O ports (D2~D12) that have a mark “D”. These ports can be used to read and control digital Crowtail modules (Crowtail modules that have a mark “D”), such as the Button and LEDs. Some of the digital I/O ports can also be used as PWM (pulse width modulation) outputs;
- ② 6 Analog ports (A0~A5) 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;
- ③ 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;
- ④ 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;

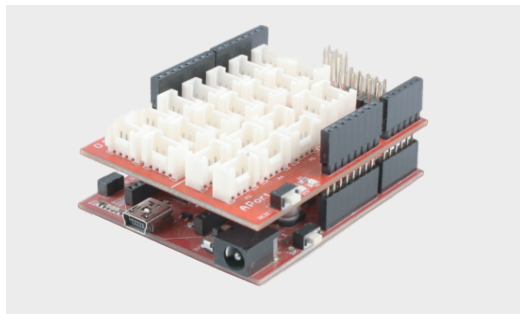
Besides, there is also a 2x5 female connector of 5V and GND, for customer usages. Users can connect any electronic modules to the Base Shield with jumper wires easily.

Compared with the traditional way of carrying out electronic projects, Crowtail has a huge performance benefit. All Crowtail has the standard 4 pin connectors. Your creative idea can be realized easier and faster just by plug and play. In addition, you don't need to debug the electronic circuits!

Connect Crowtail-Base shield with your Arduino.



Connect Crowtail-Base shield with your Crowduino.



Crowtail – Module

We make a number of electronic modules into Crowtail modules, they include a variety of sensors, displays, inputs and outputs modules, communication types include I2C, UART, digital or analog, all modules can be used by simply connecting them to the Crowtail-Base shield using a Crowtail cable, which is a huge improvement over the previously troublesome jumper connections.

Modules List

- Crowduino Uno-SD x1
- Crowtail - Base Shield x1
- Crowtail - Button x1
- Crowtail - Switch x1
- Crowtail - Buzzer x1
- Crowtail - LED(Red) x1
- Crowtail - LED(Green) x1
- Crowtail - LED(Yellow) x1
- Crowtail - Logic AND x1
- Crowtail - Logic OR x1
- Crowtail - Logic NOT x1
- Crowtail - Flame Sensor x1
- Crowtail - Moisture Sensor x1
- Crowtail - Touch Sensor x1
- Crowtail - Relay x1
- Crowtail - Light Sensor x1
- Crowtail - Thermistor Temperature Sensor x1
- Crowtail - Vibration Motor x1
- Crowtail - Linear Potentiometer x1
- Crowtail - Tilt Switch x1
- Crowtail - Sound Sensor x1
- Crowtail - 4-Digit Display x1

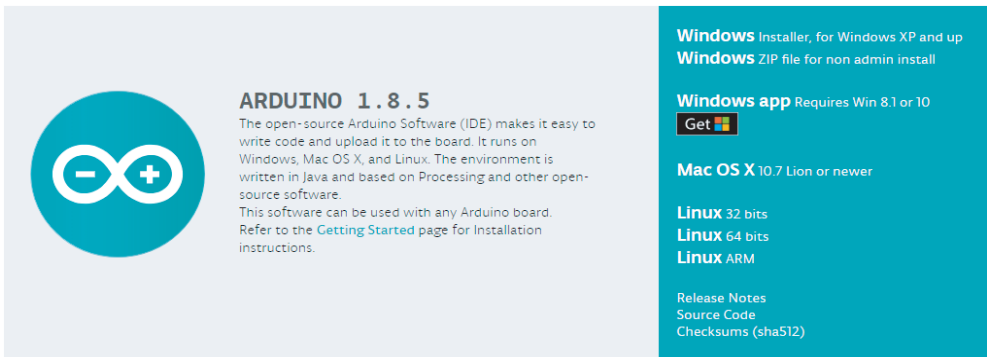
Getting Started

Installing IDE

Introduction

"Arduino" is not only the name of the microcontroller board but also the name of a programming IDE based on C/C++. After you get your Arduino board or compatible board such as Crowduino, you should install the IDE. Depending on the OS version, the specific installation varies. The Arduino software that you will use to program your Arduino is available for Windows, Mac, and Linux. In this lesson, you will learn how to set up your computer to use Arduino.

STEP1: Go to <https://www.arduino.cc/en/Main/Software> and find below page.



The screenshot shows the Arduino 1.8.5 download page. On the left is the Arduino logo (an infinity symbol with a minus and plus sign). To its right, the text reads: **ARDUINO 1.8.5**. Below this, it says: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions." On the right side of the page, there are links for different operating systems: **Windows** (Installer for Windows XP and up, Windows ZIP file for non admin install), **Windows app** (Requires Win 8.1 or 10, with a 'Get' button), **Mac OS X** (10.7 Lion or newer), **Linux** (32 bits, 64 bits, ARM), **Release Notes**, **Source Code**, and **Checksums (sha512)**.

The version available at this website is usually the latest version, and the actual version may be newer than the version in the picture.

STEP2: Download the development software that is compatible with the operating system to your computer. Take Windows as an example here.

- Click "Windows Installer" —> Click "JUST DOWNLOAD"

Click "I Agree" to see the following interface, then click "Next" and press "Browse..." to choose an installation path or directly type in the directory you want.



Click "Install" to initiate installation.

Then when finished, the Arduino icon will appear and like this



Connect Arduino/Crowduino to PC.

Connect the Arduino/Crowduino board to your computer by using the USB cable.

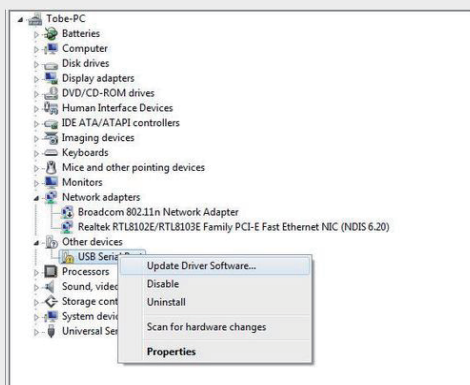
Install Driver

Installing drivers for the Crowduino In Windows.

Plug in your Crowduino, Windows will try to install the driver automatically. If you are lucky enough, the installation will be done automatically in about 1 minute. If not, follow the next steps.

STEP1: Open the Device Manager by right-clicking "My computer" and selecting control panel. Look under Ports (COM & LPT). You should see an open port named "USB Serial Port"

STEP2: Right click on the "USB Serial Port" and choose the "Update Driver Software" option.

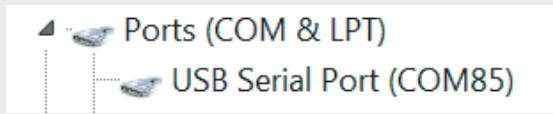


Choose the "Browse my computer for Driver software" option.

STEP3: Select the driver file named FTDI USB Drivers, located in the "Drivers" folder in the Arduino IDE.



Click Next, if you installed driver successfully, Check with serial port the Crowduino is using by opening the Windows Device Manager.



Installing Arduino (Linux)

You will have to use the make install command. If you are using the Ubuntu system, it is recommended to install Arduino IDE from the software center of Ubuntu.

Installing Arduino (Mac OS X)

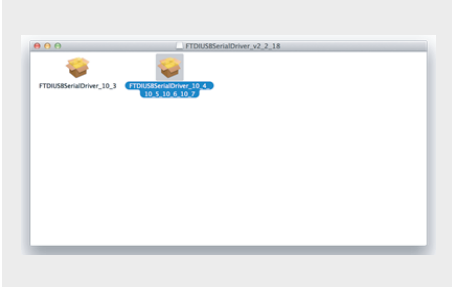
Download and Unzip the zip file, double click the Arduino.app to enter Arduino IDE; the system will ask you to install Java runtime library if you don't have it in your computer. Once the installation is complete you can run the Arduino IDE.

Installing drivers for the Crowduino with Mac OS

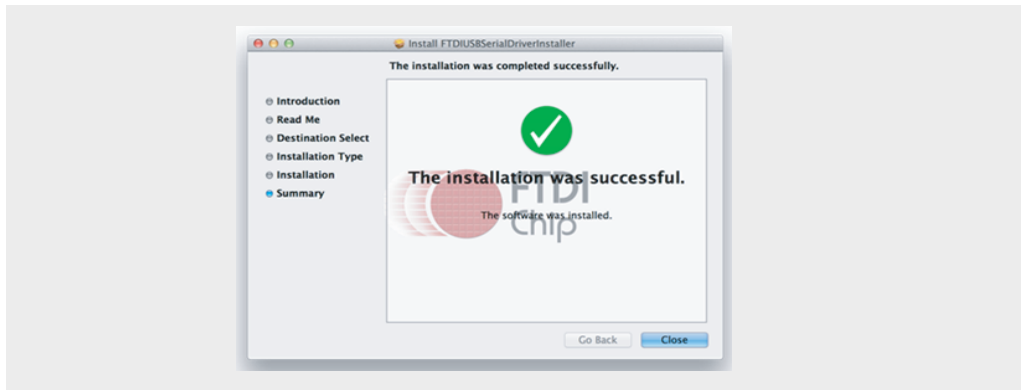
Enter page: <http://www.ftdichip.com/Drivers/VCP.htm>.

Download Driver for the Mac OS X version, and the right version for your own computer. Open the driver file which you just downloaded, and double click.

"FTDIUSBSerialDriver_10_4_10_5_10_6_10_7.mpkg" and select "Continue" option.



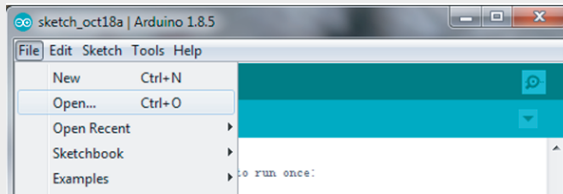
You can see the below dialog box es if you have installed driver successfully.



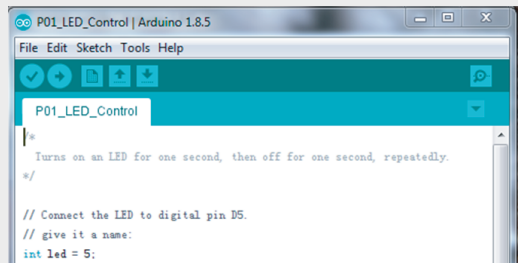
How to upload program

To upload program for Arduino, you can follow the steps as below.

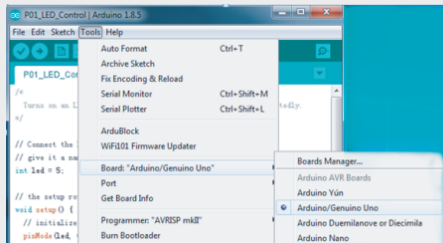
STEP1: Double click to open your Arduino IDE software and navigate “File” → “Open”



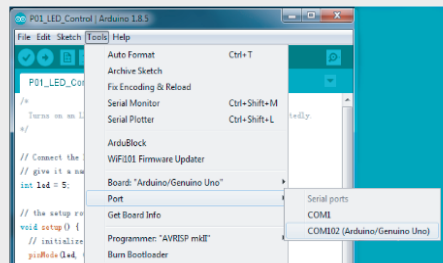
STEP2: Find the directory where you store the Arduino script and click to open it. You will then see the Arduino sketch selected for opening.



STEP3: Select the board for your Arduino, Select Tools → Board → Arduino/Genuino Uno.



STEP4: Select the port that corresponds to your board connection, Select Tools → Port → COM102(Arduino/Genuino Uno).

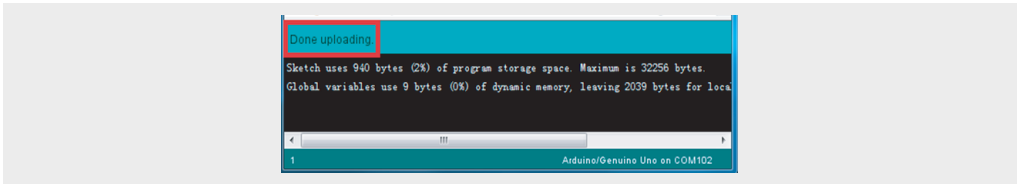


STEP5: Click the upload icon



to upload the program.

STEP6: After the program is uploaded, you will see the upload success prompt.



Add Libraries and Open Serial Monitor

What are libraries?

Libraries are a collection of code that makes it easy for you to connect a sensor, display, module, etc. There are hundreds of additional libraries available on the Internet for downloading. To use the additional libraries, you will need to install them. How to install a library? There are two functions to install the library, one is using the library manager, the other is to import a .zip library.

Importing a .ZIP Library

Libraries are often distributed as a ZIP file or folder, the name of the folder is the name of the library. Inside the folder is a .cpp file, a .h file, and often a keywords.txt file, example folder, and other files required by the library.

Process

- Unzip the zip library file, then copy and paste the extracted folder into the Arduino libraries folder (for example, E:\Arduino\libraries). Then your library file is successfully added.

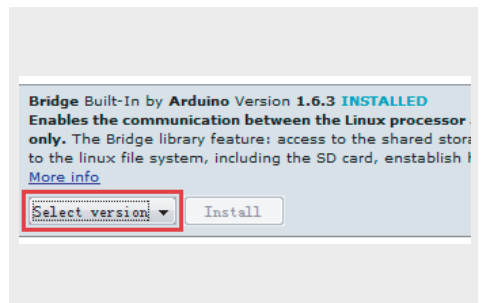
Note: the zip file will be available to use in sketches, but examples for the library will not be exposed in the “File>Examples” until after the IDE has restarted.

Using the Library Manager

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.0).

Process

- Open the IDE → click “Sketch” → click “Include Library” → click “Manage Libraries”
- Open the Library Manager, you will find a list of libraries that are already installed or ready for installation. We will install the Bridge Library in this example, scroll the list to find it, then select the version of the library you want to install
- Finally, click “install” and wait for the IDE to install the new library. Once it has finished, an installed tag should appear next to the Bridge Library.



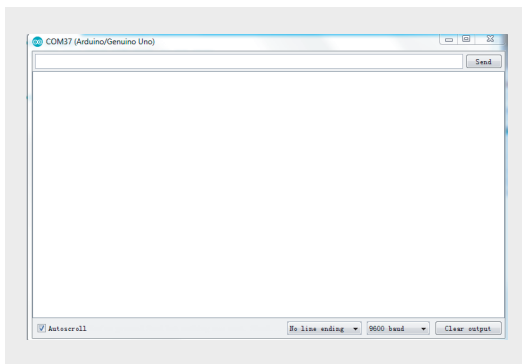
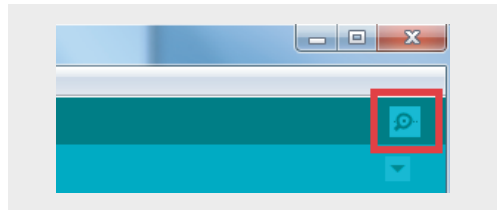
Arduino Serial Monitor (Windows, Mac, Linux)

The Arduino IDE is the software side of the Arduino platform, for it is a big part of working with Arduino and other microcontrollers, so the software includes a serial terminal. It is called the serial monitor within the Arduino environment.

The Serial Monitor is one of the Arduino IDE's many great built-in tools. It can help you understand the values that your program is trying to work with, and it can be a powerful debugging tool when you run into issues where your code is not behaving the way you expected it to. This circuit introduces you to the Serial Monitor by showing you how to print the values from your potentiometer to it.

How to use serial monitor

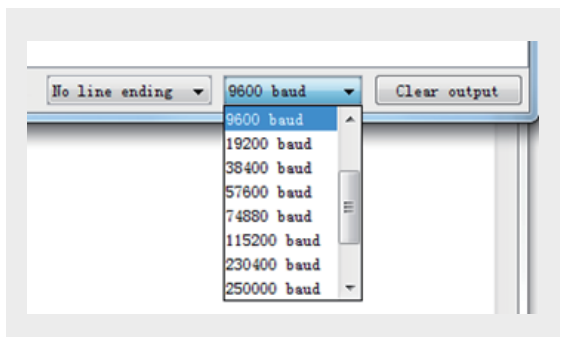
- Go to “Tools” → “Serial Port”, choose the COM the same as the one in “Device Manager”.
- Click the serial monitor icon.

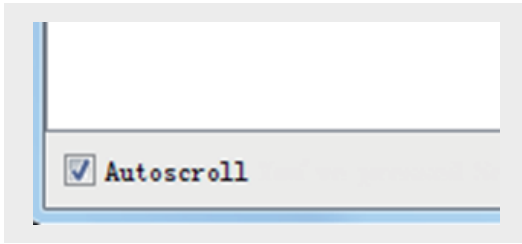


- You should see the “Serial.Monitor” window as beside.

Settings

The serial monitor has limited settings, but it's enough to meet most of your serial communication needs. The first setting you can alter is the baud rate, click on the baud rate drop-down menu to select the correct value.





You can set the terminal to autoscroll or not by checking the box in the bottom left corner.

Lessons

Lesson 1 – LED Control

Introduction

"Arduino" is not only the name of the microcontroller board but also the name of a programming IDE based on C/C++. After you get your Arduino board or compatible board such as Crowduino, you should install the IDE. Depending on the OS version, the specific installation varies. The Arduino software that you will use to program your Arduino is available for Windows, Mac, and Linux. In this lesson, you will learn how to set up your computer to use Arduino.



Required Parts

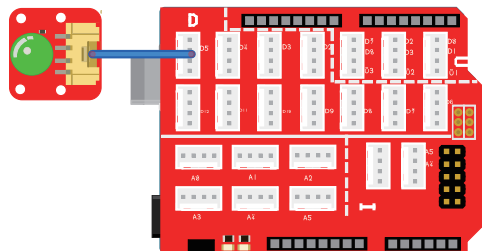
- Crowduino UNO-SD x1
- Crowtail - Base Shield x1
- Crowtail - LED x1
- Crowtail - Cable x1
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail- Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect the Crowtail- LED to the Base Shield D5 port. The complete connection is as follows:

Download the Crowtail starter kit for Arduino demo code, Open the [P01_LED_Control.ino](#) with Arduino IDE and upload it.



What will you see

The LED will light for one second and off for one second, then repeat. If not, be sure to connect the LED module correctly to the Crowtail-Base Shield interface.

Code overview and usage

```
1 // LED_Crowtail
2 // Turns on an LED for one second, then off for one second, repeatedly.
3 //
4
5 // Connect the LED to digital pin D5.
6 // give it a name:
7 int led = 5;
8
9 // the setup routine runs once when you press reset:
10 void setup() {
11   // initialize the digital pin as an output.
12   pinMode(led, OUTPUT);
13 }
14
15 // the loop routine runs over and over again forever:
16 void loop() {
17   digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
18   delay(1000);             // wait for a second
19   digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
20   delay(1000);             // wait for a second
21 }
```

Comments: //This is a comment
Comments are a great way to leave notes in your code explaining why you wrote it the way you did. You'll find many comments in the examples that further explain what the code is doing and why. Comments can be single line using `//`, or they can be multi-line using `/* */`.

Integer Variables
A variable is a placeholder for a value that may change in your code. Variables must be introduced or "declared" before using variables. Here, we declare a variable called "led" of type `int`(integer) and assign it a value of 5, that is, led corresponds to pin 5. Don't forget that variable names are case sensitive!

code to run once
The code inserted between the `setup()` braces will run once.

Input or Output
Before using one of the digital pins, you need to tell Arduino whether it is an input (INPUT) or an output (OUTPUT). We use a built-in "function" called `pinMode()` to make the pin corresponding to the led a digital output.

code to run forever
The code inserted between the `loop()` braces will run repeatedly until the Arduino resets or powers off.

Digital Output
When you're using a pin as an OUTPUT, you can command it to be HIGH (output 5 volts) or LOW (output 0 volts).

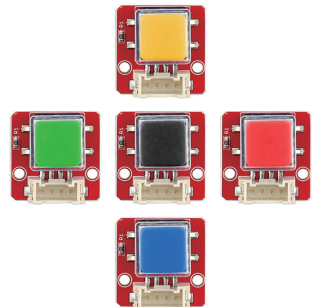
Delay
Causes the program to wait on this line of code for the amount of time in between the brackets. After the time has passed, the program will continue to the next line of code.

Lesson 2 – Button Control LED

Introduction

This momentary button output logic HIGH signal when pressed and logic LOW signal when released. The logic high and logic levels of the output can be detected by the Arduino controller, and then you can program your Arduino to do what you want after detecting the two different signals.

Since LEDs have already been learned, this lesson will let us use LED and button to make a button light.



Required Parts

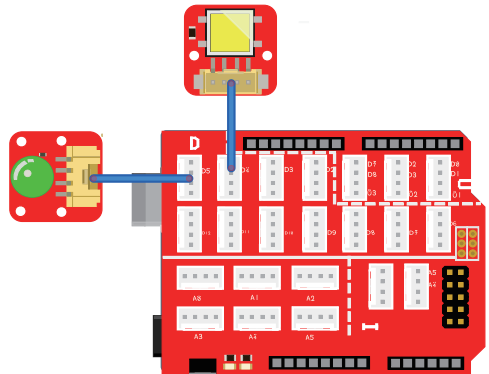
- Crowduino UNO-SD x1
- Crowtail - Base Shield x1
- Crowtail - LED x1
- Crowtail - Button x1
- Crowtail - Cable x2
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail- Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Button to Crowtail-Base Shield D4 port and Crowtail-LED to D5 port. The complete connection is as follows:

Open the `P02_Button_Control_LED` with Arduino IDE and upload it.



What will you see

If you press the button, the LED will light on, and it will light off when you release the button. If it doesn't, make sure the LED and button are properly connected to the corresponding Crowtail-Base Shield interface.

Code usage

Integer Variables: `int buttonState=0;`

A variable is a placeholder for a value that may change in your code. Variables must be introduced or "declared" before using variables. Here, we declare a variable called 'buttonState' of type int(integer) and assign it a value of 0 to record the status of the button. Don't forget that variable names are case-sensitive!

Digital Input: `buttonState = digitalRead(buttonPin);`

We use the `digitalRead()` function to read the value on a digital pin. Check to see if an input pin is reading HIGH(5V) or LOW(0V). Returns TRUE(1) or FALSE(0) depending on the reading.

If/else Statements: `if(logic statement) {code to be run if the logic statement is true}
else {code to be run if the logic statement is false }`

The if / else statement allows your code to make corresponding choices for different results, running a set of code when the logical statement in parentheses is true, and another set of code when the logical statement is false. For example, if the button is pressed, the LED will light on and when the button is released, the LED will light off.

Is equal to: `buttonState == HIGH`

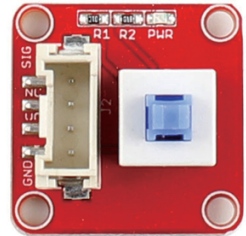
This is another logical operator. The "equal" symbol (==) can be confusing. The two equal signs are equal to ask: "The two values are equal to each other?" On the other hand, if you want to compare two values, don't forget to add a second equal sign, because if it's just a "=", it's an assignment method.

Lesson 3 – Switch LED

Introduction

The Crowtail- Switch is a Latching switch. When the switch is pressed for the first time, the switch maintains current regulation and the button outputs a HIGH signal in the self-locking state. When the switch is pressed for the second time, the switch button pops up and the switch turns off and then outputs a LOW signal. In fact, it is very similar to the button, except that the switch has a self-locking function so that it can output logic high level signal without pressing it all the time.

Since button and switch have the same thing, then in this lesson, we will use a switch and led to make a control light, which makes it easy for you to see the difference between switch and button.



Required Parts

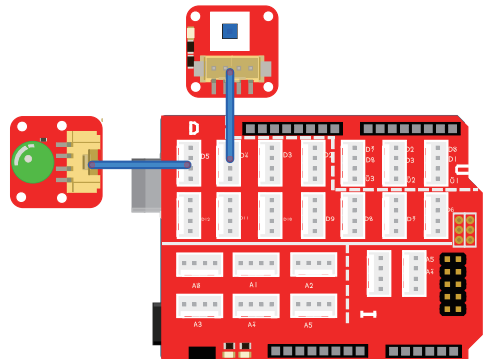
- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Switch x1
- Crowtail – LED x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail- Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Switch to Crowtail-Base Shield D4 port and Crowtail-LED to D5 port. The complete connection is as follows:

Open the [P03_Switch_Control_LED](#) with Arduino IDE and upload it.



Code usage

Integer Variables: `int switchPin=4; int ledPin=5;`

A variable is a placeholder for a value that may change in your code. Variables must be introduced or "declared" before using variables. Here, we declare a variable called 'ledPin' and 'switchPin' of type int(integer) and assign it a value of 5 and 4, that is, led corresponds to pin 5 while switch corresponds to pin 4. Don't forget that variable names are case sensitive!

Input or Output: `pinMode(ledPin,OUTPUT); pinMode(switchPin,INPUT);`

Before using one of the digital pins, you need to tell Arduino whether it is an input (INPUT) or an output (OUTPUT). We use a built-in "function" called pinMode() to make the pin corresponding to the led a digital output and the switch module as an input.

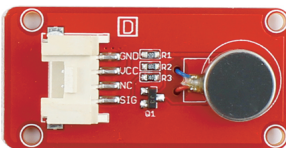
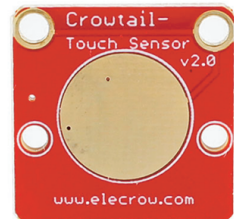
Loop: `void loop(){code to run forever}`

In the loop() function, use the digitalWrite() function to read the switch value, then use the if/else statement to determine if the switch is pressed(HIGH) or not pressed(LOW). If it is pressed, turn on the LED, otherwise, turn off the LED.

Lesson 4 – Touch reminder

Introduction

The touch sensor can detect the human touch by sensing the change of capacitance. When it detects a touch, it will output a HIGH logic level signal. Based on the touch IC TTP223-B, this module can detect human finger in 0~3mm, that is, you can place this sensor under a non metallic surface such as the glass or paper, with thickness less than 3MM, this would be useful for applications that waterproof is needed, or you want to make the buttons secret.



The vibration modules vibrate when it activated by logic HIGH level signal, like the vibration of a phone. This module is consists of a Permanent Magnet coreless DC motor, with the upgraded driver, the vibration gets much fierce, suitable for applications such as toys and alarms.

Do you have trouble getting the touch device unlocked without a prompt? Especially for the blind, it is very necessary for them to have a feedback on the touch device, so in this lesson, we will use the touch sensor and vibration motor to make a touch reminder to help those who need feedback.

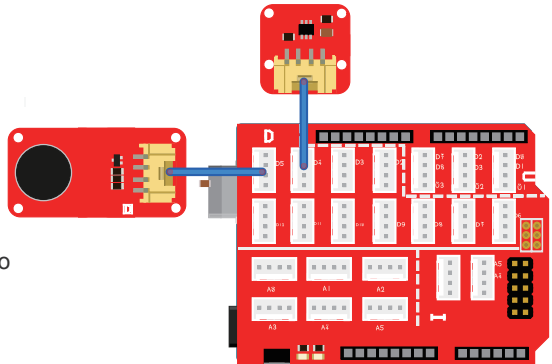
Required Parts

- Crowduino UNO-SD x1
- Crowtail – Touch Sensor x1
- Crowtail – Vibration Motor x1
- Crowtail – Base Shield x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Touch sensor to Crowtail-Base shield's D4 and Crowtail-Vibration motor to D5 port. The complete connection is as follows:



Open the [P04_Touch_Reminder](#) with Arduino IDE and upload it.

What will you see

If you touch the touch sensor, the vibration motor will vibrate. When you stop touching, the vibration motor will stop vibrating. If it doesn't, make sure the location you touch is within the sensing range of the touch sensor and sensors are properly connected to the corresponding Crowtail-Base Shield interface.

Code usage

Integer Variables: `int touchPin=4;`

A variable is a placeholder for a value that may change in your code. Variables must be introduced or "declared" before using variables. Here, we declare a variable called 'touchPin' of type int(integer) and assign it a value of 4, that is, the touch sensor corresponds to pin 4. Don't forget that variable names are case sensitive!

Input or Output: `pinMode(touchPin,INPUT);` `pinMode(vibrationPin, OUTPUT);`

Before using one of the digital pins, you need to tell Arduino whether it is an input (INPUT) or an output (OUTPUT). We use a built-in "function" called `pinMode()` to make the pin corresponding to the touch sensor a digital INPUT.

Loop: `void loop(){code to run forever}`

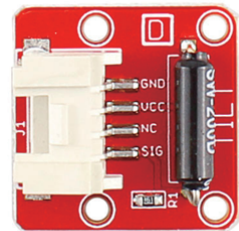
Firstly, for the reason that the touch sensor is a digital module, we use the `digitalRead()` function to read the touch sensor value, then use the `if/else` statement to determine if the touch sensor is touched(HIGH) or not touched(LOW). If it is touched, let the vibration sensor vibrate, otherwise, the vibration sensor will not vibrate.

Lesson 5 – Tilt Alarm

Introduction

The Crowtail-Tilt Switch is a sensor to detect the station of forwards. It outputs logic LOW in Horizontal direction and logic HIGH in the vertical direction. Inside the tilt switch is a pair of balls that make contact with the pins when in the case of vertical. Tilt the case over and the balls don't touch, thus not making a connection.

Do you know that the refrigerator can't tilt? Because if the refrigerator is tilted, it will affect its cooling effect, so in this lesson, we will make a tilt reminder to prevent the appliance from tilting.



Required Parts

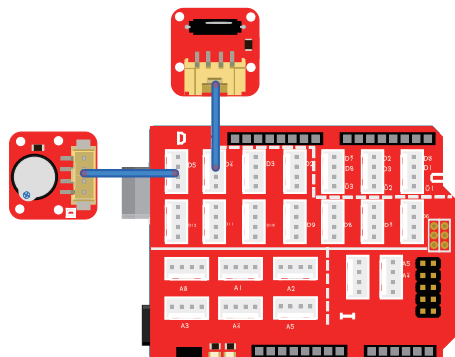
- Crowduino UNO-SD x1
- Crowtail – Tilt Sensor x1
- Crowtail – Buzzer x1
- Crowtail – Base Shield x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Tilt and Crowtail-Buzzer to Crowtail-Base shield's D4 and D5 port. The complete connection is as follows:

Open the `P05_Tilt_Alarm` with Arduino IDE and upload it.



What will you see

Let the tilt sensor lie flat. When you tilt the sensor to the left, logic high signal will be output and the buzzer will beep. When you tilt the tilt sensor back or right, logic low signal will be output and the buzzer will stop sounding.

Code usage

Integer Variables: `int tiltPin = 4; int buzzerPin = 5; int tiltState = 0;`

Here, we declare three variables `tiltPin`, `buzzerPin` and `tiltState` and assign values to them. This allows us to connect the tilt sensor to the D4 pin, the buzzer to the D5 pin, and the initial tilt sensor state to 0.

Input or Output: `pinMode(tiltPin,INPUT); pinMode(buzzerPin,OUTPUT);`

In the `setup()` method, we need to initialize the tilt sensor as an input to detect if a tilt occurs, and initialize the buzzer to an output to sound a sound when we detect the tilt.

Digital Input: `tiltState = digitalRead(tiltPin);`

We use the `digitalRead()` function to read the value on tilt sensor. Check to see if tilt sensor is reading HIGH(5V) or LOW(0V). Returns TRUE(1) or FALSE(0) depending on whether the tilt sensor is tilted.

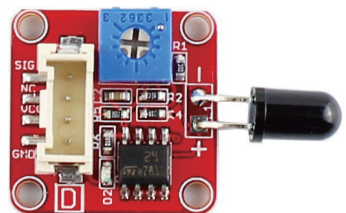
If/else Statements: `if(logic statement) {code to be run if the logic statement is true}
else {code to be run if the logic statement is false }`

Here, we use the `if / else` statement to execute the program that needs to be executed when the tilt is detected or the tilt is not detected. If the tilt is detected, Buzzer will beep, otherwise, the buzzer will not make a sound.

Lesson 6 – Flame Alarm

Introduction

The Crowtail- Flame Sensor can be used to detect fire sources 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. Due to its black epoxy, the sensor is sensitive to infrared radiation. In a fire fighting robot game, the sensor plays a very important role, it can be used as a robot's eyes to find the fire source. Fire prevention is very important in our daily life, because the fire will pose a serious threat to our lives and property, so in this lesson, we will make a fire simulation detector, and immediately warn us when the fire occurs.



Required Parts

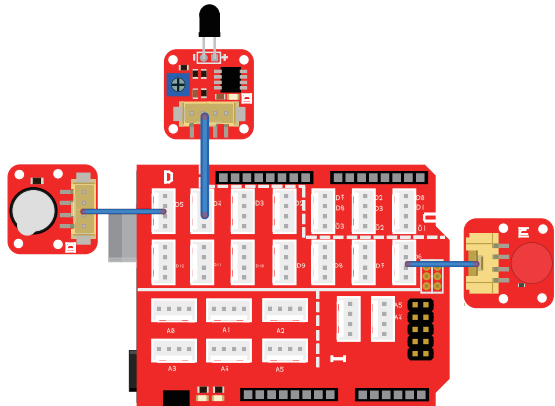
- Crowduino UNO-SD x1
- Crowtail – Flame Sensor x1
- Crowtail – Buzzer x1
- Crowtail – LED x1
- Crowtail – Base Shield x1
- Crowtail – Cable x3
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect the Crowtail-Flame sensor, Crowtail-Buzzer and Crowtail-LED to the D4, D5 and D6 ports of the Crowtail-Base shield. The complete connection is as follows:

Open the [P06_Flame_Alarm](#) with Arduino IDE and upload it.



What will you see

When the fire is close to the flame sensor, the buzzer will beep and the red LED will flash to warn of danger. When the fire goes out, the buzzer stops beeping and the LED goes out.

Code usage

Integer Variables: `const int flamePin = 4; const int buzzerPin =5; const int redPin =6; int flameState = 0;`

In the beginning, we declare four variables `flamePin`, `buzzerPin`, `redPin`, and `flameState` and assign values to them. You should notice the difference. We use “const int” and “int” to declare variables respectively. ‘const’ is the abbreviation of constants. If you use this to define variables, the variables are marked as “read-only”, that is, they cannot be changed during the program. Constants are great for declaring pin number variables that will not change throughout the program. In conclusion, we declare the flame sensor is connected to the D4 pin, the buzzer is connected to the D5 pin, the led is connected to the D6 pin and the initial `flameState` is 0.

Input or Output: `pinMode(flamePin,INPUT); pinMode(buzzerPin,OUTPUT);pinMode (redPin, OUTPUT);`

In the setup() method, we need to initialize the flame sensor as an input to detect whether a fire has occurred, and initialize the buzzer and led as outputs respectively to make the buzzer beep and led to flash when a fire is detected.

Digital Input: `flameState = digitalRead(flamePin);`

Like other digital input modules, We use the `digitalRead()` function to read the value on flame sensor. Check that the flame sensor reading is HIGH(5V) or LOW(0V). Note: It will return FALSE(0) if a fire is detected and TRUE(1) if not fire is detected.

If/else Statements: `if(logic statement) {code to be run if the logic statement is true}
else {code to be run if the logic statement is false }`

In the if/else statement, if the flame sensor detects a fire, the buzzer will beep and the led will flash. If no fire is detected, the buzzer will not beep and the led will turn off.

Lesson 7 – Breathing LED

Introduction

The Crowtail- Linear Potentiometer module uses a linear variable resistor with a maximum resistance of 10KΩ. When you move the slider from one side to the other, its output voltage will range from 0 V to the VCC you applied. The panel mount design makes this module easy to be installed in projects. In this course, we will use a linear potentiometer to control a LED light as human breathing!



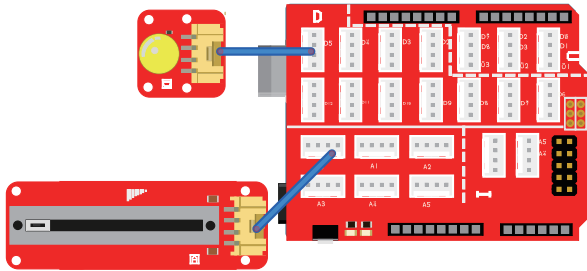
Required Parts

- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Linear Potentiometer x1
- Crowtail – LED(Yellow) x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Linear Potentiometer and Crowtail-LED to Crowtail-Base shield's A0 and D5 port. The complete connection is as follows:



Open the `P07_Breathing_LED` with Arduino IDE and upload it.

What will you see

When you move the slider on Linear Potentiometer, you can see the change of LED brightness. If you move the slider to the far left of the sliding rheostat closest to the Crowtail connector, the LED is the darkest and then the LED should be the brightest when you move the slider to the far right.

Code usage

Integer Variables: `int ledPin = 5; int analogInPin = A0; int pmeterValue = 0;`

It will always be the first to declare the pin we are going to use and assign values to them. What is different is that we use an analog pin for Linear Potentiometer in this lesson. We declare the LED pin is D5 and Linear Potentiometer is A0 and initial `pmeterValue` to 0.

Input or Output: `pinMode(pmeterPin, INPUT); pinMode(ledPin, OUTPUT);`

In the `setup()` function, initialize Linear potentiometer as an input to read value from it, and initialize the LED to an output to write the value that read from Linear potentiometer.

Analog Input: `pmeterValue = analogRead(pmeterPin);`

Different from digital pin, we use the `analogRead()` function to read the value on an analog pin. `analogRead()` takes one parameter, the analog pin you want to use, A0 in this case, and returns a number between 0 (0 volts) and 1023 (5 volts), which is then assigned to the variable: `pmeterValue`.

Arduino math function `map()`: `int gapValue = map(value, fromLow, fromHigh, toLow, toHigh)`

Remap numbers from one range to another. That is, if the value is "fromLow", the mapped value will be "toLow". If the value is "fromHigh", then the mapped value will be "toHigh". When "value" is from other values from `fromLow` to `fromHigh`, it is also mapped to between `toLow` and `toHigh` in equal proportions. So here we map the value of `pmeterValue` (between 0 and 1023) to `gapValue` (between 0 and 255). For example, if the value of `pmeter` is 200, after using the `map()` function, it will become 50 and be assigned to the variable: `gapValue`.

Analog Input: `analogWrite(ledPin, gapValue);`

For analog pin, we use the `analogWrite()` function to write the value on an analog pin. Similar to `digitalWrite()` function, it takes two parameters, But the second parameter is no longer only two high and low states, it can be any number you want to write, and each value you write will give it the corresponding state. But in fact, there is a premise, that is, the hardware must be able to divide so many levels of effect. For example, most of the led is 256.

Lesson 8 – Moisture Monitor

Introduction



This Moisture Sensor can be used to detect the moisture of soil and thus to monitor if the plants in your garden need some water. This sensor uses the two probes to pass current through the soil, and then it reads the resistance to get the moisture level. More water makes the soil conduct electricity more easily (less resistance), while dry soil conducts electricity poorly (more resistance). Compares to the other moistures sensor using the same moisture test method, this module has super long legs, making it suitable for actual applications.

Do you have any plants in your home? Now that we know the humidity sensor can detect the soil moisture, let us use the humidity sensor to detect the humidity of the plant and print it out to remind us when we need to water the plants.

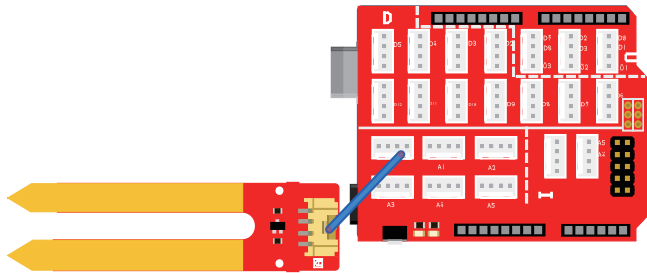
Required Parts

- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Moisture Sensor x1
- Crowtail – Cable x1
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Moisture sensor to Crowtail-Base shield's A0 port. The complete connection is as follows:



Open the [P08_Moisture_Reminder](#) with Arduino IDE and upload it.

What will you see

Open the Serial monitor, if you put the Moisture in the air, You can see that the value detected by the Moisture sensor printed by the serial monitor is 0. When you put it in the soil or water, You can see that the value keeps increasing and then reaches a continuous value.

Code usage

Integer Variables: `const int MoisturePin = A0; int sensorValue = 0;`

First, we declare two variables, MoisturePin and sensorValue which are assigned to A0 and 0 respectively. MoisturePin is the pin used to connect to the Moisture sensor. We don't want to change it in the program, so we use constants to declare it.

Serial Begin: `Serial.begin(9600);`

Serial can be used to send and receive data from your computer. This line of code tells Arduino that we want to “begin” that communication with the computer, just like the way we say “Hi” to start a conversation. “9600” is the baud rate we set, which represents the speed of communication between two devices. Notice that the baud rate must match on both sides.

Serial print: `Serial.print("sensor = "); Serial.println(sensorValue);`

Serial.print() is used to print the string or variable that you want to output, If you want to output a string, you need to wrap the string with " ". If you want to output the variable, just write the variable name in parentheses. You may have noticed that one is print and the other is println. The difference between them is that after “print” method prints the content, the information that needs to be printed can continue to be displayed in this line, and println method will open a new line after printing the content, that is, the information that needs to be printed can't continue to be displayed in this line.

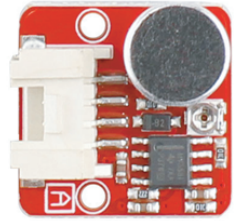
Lesson 9 – Sound Reminder

Introduction

Crowtail-Sound Sensor can detect the sound loudness of the environment. The main component of the module is a simple microphone to receive the sound signal, with band pass filter and a LMV358 amplifier to make the output signal easy to be received by microprocessor.

This module outputs an analog signal that shows the relative loudness. Besides, an on-board potentiometer can be used to adjust the output voltage.

We will use sound sensor and buzzer together to make a noise reminder, you can use this project to remind yourself not to be too noisy to affect others.



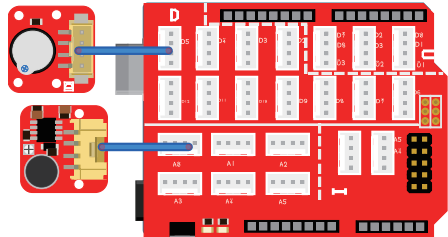
Required Parts

- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Sound Sensor x1
- Crowtail – Buzzer x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Sound sensor and Crowtail-Buzzer to Crowtail-Buzzer shield's A0 and D5 port. The complete connection is as follows:



Open the [P09_Sound_Reminder](#) with Arduino IDE and upload it.

What will you see

If the sound in your surroundings is too noisy or you are yelling at the sound sensor, the buzzer will beep for one second to remind you to lower the volume. If the sound is still loud, the buzzer will continue to beep. Otherwise, the buzzer will stop beeping.

Code usage

Integer Variables: `int soundPin = A0;` `int buzzerPin = 5;` `int soundValue = 0;`

Declare the soundPin connect to A0 pin, buzzerPin connect to D5 pin and initialize soundValue to 0.

If/else Statements: `if(logic statement) {code to be run if the logic statement is true}`
`else {code to be run if the logic statement is false }`

In the if / else statement, if the soundValue is greater than 500, we let the buzzer sound for one second, then loop () to determine if the soundValue is still greater than 500, and if so, the buzzer will continue to beep. Otherwise, the buzzer will stop beeping. Of course, if you feel that the sound is very big and the buzzer still won't beep, you can modify the value of soundValue to find the maximum volume limit that suits you best.

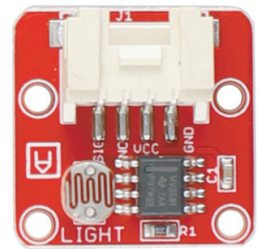
Lesson 10 – Intelligent street light

Introduction

The Light sensor module uses the GL5516 photoresistor to detect the light intensity of the environment. The resistance of the sensor decreases when the light intensity of the environment increases. The chip LMV358 is used as a voltage follower to enable you to get an accurate data.

This module outputs an analog signal that shows the light intensity. It can be used in many occasions, such as intelligent street light, intelligent corridor light, etc.

Are you paying attention to the street lights on the road? They will automatically turn on when it is dark. How about using the light sensor and LED to make our own smart street lights in this lesson!



Required Parts

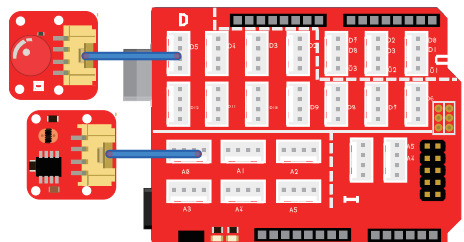
- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Light Sensor x1
- Crowtail – LED x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Light sensor to Crowtail-Base shield's A0 and Crowtail-LED to D5 port. The complete connection is as follows:

Open the [P10_Intelligent_Street_Light](#) with Arduino IDE and upload it.



What will you see

In the daytime, use your hand to block the top of the light sensor, you will see that the led will automatically light up. When you remove your hand, the led will automatically go out. You can also observe the difference between day and night, and then make an energy-saving light for your home!

Code usage

Integer Variables: `int lightPin = A0; int ledPin = 5; int lightValue = 0;`

Define the lightPin connect to A0 port of the Crowtail-Base shield and ledPin to D5 port. Define a variable called lightValue to store the value read by the light sensor and initialize the value to 0.

Analog Input: `lightValue = analogRead(lightPin);`

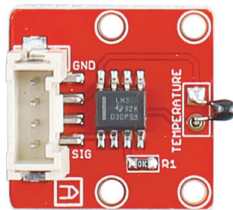
In the loop() function, use analogRead() to repeatedly read the value of light sensor and store the value to the variable named lightValue.

If/else Statements: `if(logic statement) {code to be run if the logic statement is true}
else {code to be run if the logic statement is false }`

What we want to do in if/else statement is to judge if the value of the light intensity less than 800, so for logic statement we use the code "lightValue<800" to express, if yes, it will return TRUE and execute the code in if{}, otherwise, it will return FALSE and execute the code in else{}.

Lesson 11 – Overheat alarm

Introduction



The Crowtail-Thermistor Temperature Sensor uses a thermistor to detect the ambient temperature. The resistance of a thermistor increases when the ambient temperature decreases, so the Arduino can detect the voltage and thus to calculate the current temperature. The detection range of this sensor is between -40 to 125 degrees Celsius with an accuracy of $\pm 1.5^{\circ}\text{C}$. However, it doesn't output the temperature value directly. To get the specific temperature value, we will use the formula in the code.

In daily life, some appliances may cause damage to electrical appliances if the temperature is too high, such as televisions, fans, and so on. Therefore, in this lesson we will use the Thermistor Temperature Sensor and Buzzer to make an overheating alarm for these appliances.

Required Parts

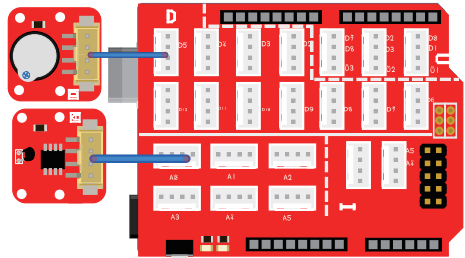
- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Thermistor Temperature Sensor x1
- Crowtail – Buzzer x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Thermistor temperature sensor to Crowtail-Base shield's A0 and Crowtail-Buzzer to D5 port. The complete connection is as follows:

Open the [P11_Overheat_Alarm](#) with Arduino IDE and upload it.



What will you see

Place this item in the environment or device you want to measure. When the temperature detected by the thermistor is higher than 35 degrees Celsius, the buzzer will beep to indicate that the temperature is overheating, otherwise, the buzzer will not beep.

Code usage

Math Library: `#include <math.h>`

Includes the math library into your program, you can call the function directly inside the library.

Float Variables: `float temperature;` `float resistance;`

The float variable, short for floating-point number, is similar to an integer except it can represent numbers that contain a decimal point. Floats are good for representing values that need to be more precise than an int. Floats allow us to measure precise temperature such as 35.58 degrees Celsius instead of just 35 degrees Celsius.

Convert to temperature: `temperature=1/(log(resistance/10000)/B+1/298.15)-273.15;`

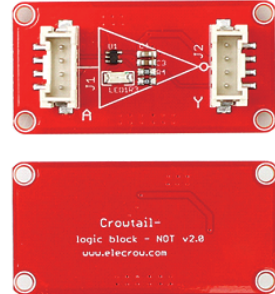
Convert to temperature via datasheet. The log is the abbreviation of Logarithmic, which is a method in the "Math.h" library. The function prototype is `Math.log()`. After using this function, the function returns the natural logarithm of the number in the parentheses, such as `log(10)`. The number returned is approximately equal to 2.3

Lesson 12 – Logic NOT

Introduction

What is logic gate?

In digital circuit applications, a logic gate is a basic logic component that acts as a switch. Under certain conditions, it can determine whether the signal passes or does not pass. A simple logic gate can be made up of transistors that perform a logic operation on the input level signal and then generate a high or low level signal. Logic gates are the basic structure of digital circuit systems. Simple logic gates can be combined into more complex logic operations, which is the basis of VLSI design. There are three basic logic gates, namely "and", "or", and "not".



Input	Output
A	A'
0	1
1	0

First, let's take a look at the Crowtail-Logic NOT module. The interface of the logic NOT module is a digital interface, it will output a logic-high or logic-low signal. The logic NOT has one input and one output, the output does a reverse operation on the input, which means if you send a logic-high signal to the input, the logic NOT will output a logic-low signal. You can check the inverting function of the Logic NOT module according to the following table.

For the Logic NOT module, we combine it with the touch light to see what's going to happen.

Required Parts

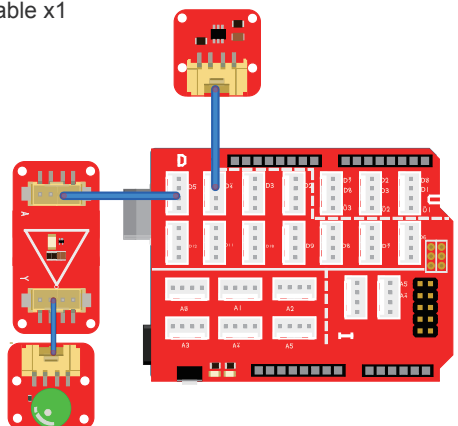
- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Logic NOT x1
- Crowtail – Touch Sensor x1
- Crowtail – LED x1
- Crowtail – Cable x3
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Touch and Crowtail-Logic NOT(A port) to Crowtail-Base shield's D4 and D5 port. Connect Crowtail-LED to Crowtail-Logic NOT(Y port). The complete connection is as follows:

Open the [P12_Logic_NOT](#) with Arduino IDE and upload it.



What will you see

After uploading the program, you will see the LED light up! This is too strange, because in the code we let the led light be on when the touch sensor is touched, but now the touch sensor is not touched at all. Don't be surprised, this is because the Logic NOT module's inverse action, when the touch sensor is not touched, our program does output a logic low level, but after the Logic NOT module is inverted, the output will become high. So, the light is lit. Now try touching the touch sensor to see what will happen.

Code usage

Integer Variables: `int touchPin = 4; int ledPin = 5; int touchState = 0;`

Declare touchPin, ledPin two variables and assign values 4 and 5 to indicate the port to which the touch sensor and led are connected. Declare that the touchState variable stores the value read by the touch sensor.

If/else Statements: `if(logic statement) {code to be run if the logic statement is true}
else {code to be run if the logic statement is false }`

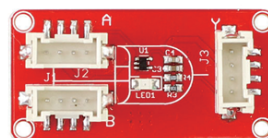
In the if/else statement, we will judge if the touch sensor is touched, and if so, turn on the LED, otherwise turn off the LED. Notice that the Logic NOT module will invert!

Lesson 13 – Logic OR

Introduction

The logic OR is a logic gate that creates rules for your circuit. But unlike NOT, the logic OR has two inputs and one output. It is a good option for projects where you want to detect two inputs but doesn't care which input is activated. The interface of the logic OR is a digital interface, it will output a logic-high or logic-low signal.

The logic OR will output a logic-high signal when either or both of the inputs are logic-high. Otherwise, the OR will output a logic-low signal when both of the inputs are logic-low.



Input		Output
A	B	F=A+B
0	0	0
0	1	1
1	0	1
1	1	1

In this lesson, we will use the two inputs of Button and Collision sensor to control the Logic OR module. Let's see what happens when its output to the vibration motor.

Required Parts

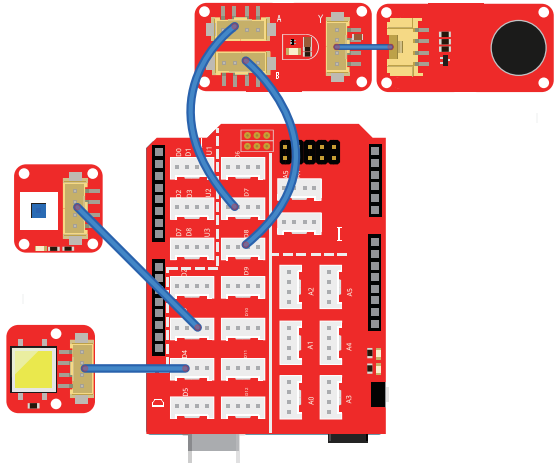
- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Logic OR x1
- Crowtail – Button x1
- Crowtail – Switch x1
- Crowtail – Vibration Motor x1
- Crowtail – Cable x5
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Switch and Crowtail-Button to Crowtail-Base shield's D3 and D4 port. Connect Crowtail-Logic OR A port and Crowtail-Logic OR B port to D7 and D8 port. Connect Crowtail-Vibration motor to Crowtail-Logic OR Y port. The complete connection is as follows:

Open the [P13_Logic_OR](#) with Arduino IDE and upload it.



What will you see

Pressing the button, the vibration motor will vibrate, press the switch, the vibration motor will also vibrate, and at the same time press the button and the switch, the vibration motor will also vibrate. That is the function of the logic OR module, as long as one of its inputs is high logic signal, then its output is a high logic signal.

Code usage

Integer Variables: `int LogicA = 7; int LogicB = 8;`

Declare two variables named "LogicA" and "LogicB" and assign values 7 and 8. This means that we need to connect the two inputs (A, B) of the logic OR module to the D7 and D8 ports.

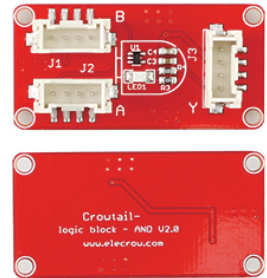
If/else Statements: `if(logic statement) {code to be run if the logic statement is true}
else {code to be run if the logic statement is false }`

Remember that the if/else statement can theoretically be used infinitely. Here we use two if/else statements, the first one for event handling when the switch is pressed, and the second for the button being pressed.

Lesson 14 – Logic AND

Introduction

The logic AND can gives you more ability to create interesting and complex interactions. It has two inputs and one output, the interface of the logic AND is a digital interface, it will output a logic-high or logic-low signal. Different from the logic OR module, the logic AND module only output a logic-high signal when both of the inputs are logic-high. Otherwise, the AND will output a logic-low signal.



Input		Output
A	B	F=A. B
0	0	0
0	1	0
1	0	0
1	1	1

In this lesson, we will use Crowtail-Logic AND implement its logical judgment function instead of using code to create smart corridor lights.

Required Parts

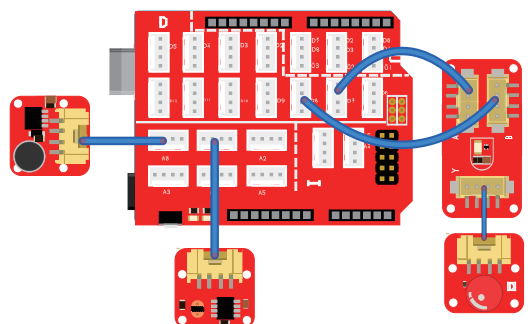
- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Logic AND x1
- Crowtail – Sound Sensor x1
- Crowtail – Light Sensor x1
- Crowtail – LED x1
- Crowtail – Cable x5
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Sound sensor and Crowtail-Light sensor to Crowtail-Base shield's A0 and A1 port. Connect Crowtail-Logic AND A port and Crowtail-Logic AND B port to D7 and D8 port. Connect Crowtail-LED to Crowtail-Logic AND Y port. The complete connection is as follows:

Open the [P14_Logic_AND](#) with Arduino IDE and upload it.



What will you see

If you put this project in a very bright environment, no matter how loud your sound is, the LED will not light up. Also, if you put it in a very quiet environment, no matter how dark your environment is, the LED will not light up. Because under the function of the Logic AND module, the two inputs must be at the same logic high level to output a logic high level. Is it amazing? The Logic AND module helps me to handle different events. In fact, this is the role of the logic module, which divides the role of complex problems into simple choices and processes.

Code usage

Integer Variables: `int soundPin = A0; int lightPin = A1;`

Define the variable `soundPin`, `lightPin` and assigned them `A0`, `A1`. Remember that our Crowtail-sound sensor and Crowtail-light sensor are analog sensors, we want the volume and brightness values read by these two sensors to help us whether Turn on the light, so we connect it to the `A0` and `A1` analog ports.

Analog Input: `lightValue=analogRead(lightPin); soundValue=analogRead(soundPin);`

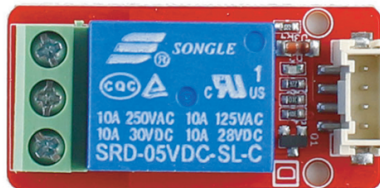
Read the value of the light sensor connected to the `lightPin` port and assign it to the variable `lightValue`. Read the value of the sound sensor connected to the `soundPin` port and assign it to the variable `soundValue`.

If/else Statements: `if(logic statement) {code to be run if the logic statement is true}
else {code to be run if the logic statement is false }`

We use two If/else statements here. The first one is to judge if the `soundValue` larger than 500, if so, output high logic signal to LogicA port, otherwise output low logic signal to LogicA port. The second one is to judge if the `lightValue` less than 800, if so, output high logic signal to LogicB port, otherwise output low logic signal to LogicB port.

Lesson 15 – Automatic Control System

Introduction



The Relay is the most commonly used module in some actual applications such as home automation, and sometimes you need to control large current such as the air conditioning/water heater. That is what this Large Current Relay Module can help.

The connector of the relay module has three sockets: common (COM), normally closed (NC), and normally open (NO). COM: common pin. NC (Normally Closed): the normally closed configuration is used when you want the relay to be closed by default, meaning the current is flowing unless you send a signal to the relay module to open the circuit and stop the current. NO (Normally Open): the relay is always open, so the circuit is broken unless you send a signal to close the circuit.

This Large Current Relay Module is a high quality Single Pole Double Throw Relay(SPST). When the relay not triggered, the 2 outputs is disconnected; when the relay triggered(Logic Low), the 2 outputs connected, the max current can be up to 30A@240VAC or 30A@30VDC, which would be enough for most of the home appliances.

Do you know how the elevator is controlled? In fact, it is an automatic control system and how does it achieve automatic control? In this lesson, We will use Relay and Thermistor temperature sensor to make an automatic high temperature indicator light.

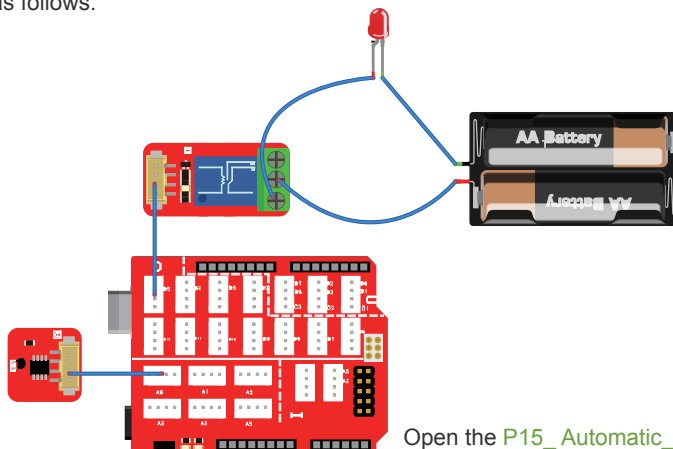
Required Parts

- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Relay x1
- Crowtail – Thermistor Temperature Sensor x1
- Crowtail – Cable x2
- LED x1
- USB Cable x1
- Battery Case x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Relay and Crowtail-Thermistor temperature sensor to Crowtail-Base shield's D5 and A0 port. Connect the power supply and LED to the relay. Connect the VCC(red wire) of the battery case to the COM port of the relay; connect the GND(black wire) of the battery case to the short leg of the LED; connect the long leg of the LED to the NO port of relay. The complete connection is as follows:



Open the [P15_Automatic_Control_System](#) with Arduino IDE and upload it.

What will you see

If your current ambient temperature is not 30 degrees Celsius, the LED will not turn on, otherwise, the LED will automatically turn on. If your environment does not reach 30 degrees Celsius, but you want to test whether the project is working, you can blow against the Thermistor Temperature Sensor and let it detect temperatures greater than 30 degrees Celsius.

Code usage

Math Library: `#include <math.h>`

Includes the math library into your program, you can call the function directly inside the library.

Float Variables: `float temperature;`

The float variable, short for floating-point number, is similar to an integer except it can represent numbers that contain a decimal point. Floats are good for representing values that need to be more precise than an int. Float allows us to measure precise temperature such as 25.58 degrees Celsius instead of just 25 degrees Celsius.

Lesson 16 – Show number on 4-Digit display

Introduction

The 4-Digit display module is a 12 pin module. But in our Crowtail gadget, we utilize a TM1650 to scale down the controlling pins into 2 Crowtail pins. It only takes I2C pins of Arduino or Crowduino to control the content, even the luminance for this display. For projects that require of the alpha-numeric display, this can be a nice choice.

The number it displays is red. In this lesson we will learn how to use the 4-Digit display module to display numbers and even display points on this module.



Required Parts

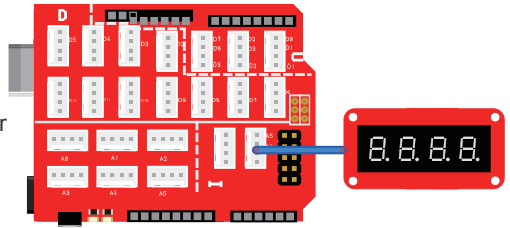
- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – 4-Digit Display x1
- Crowtail – Cable x1
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-4-Digit Display to Crowtail-Base shield's I port. The complete connection is as follows:

Open the downloaded folder "Crowtail-Starter kit for Arduino demo code", navigate to the folder lib-> TM1650, and add TM1650 to the Arduino library. If you don't know how to add it, you can skip to the "Add Libraries and Open Serial Monitor" category and check out the tutorial. Open the [P16_Show_Number_On_4-Digital_display](#) with Arduino IDE and upload it.



What will you see

You will see that the Crowtail-4-Digit Display will display the numbers 1-4 first, then the numbers 5-8 will be displayed on it.

Code usage

Import library: `#include "TM1650.h" #include <inttypes.h>`

"TM1650.h" is a 4-digit display library with very comprehensive use of functions for calling and using this module. Therefore, we should first import the TM1650.h library so that its function can be used to call the 4-Digit display. "inttypes.h" is a library that allows us to define an integer type containing information about the size of the type. In order to facilitate the definition of the certificate type later, we need to call it.

Array: `static uint8_t TubeTab[] = {} static uint8_t TubeTabwithPoit[] = {}`

Its prototype is: Array name[]. Arrays are a form of programming that organizes a set of elements of the same type in an unordered form for ease of processing. Here we create two arrays to store the numbers we want to display. If we want to get the Nth value of the array, we only need to use the array name[N-1]. "static" is a keyword. The advantage of variables declared with it is that it cannot be used by other files. Variables with the same name can be defined in other files without conflict. "uint8_t" is an unsigned character that declares that each element in the array occupies 8 bits of storage. The prerequisite for using it is to import the "inttypes.h" library.

Create an instance: `TM1650 DigitalLED(A5,A4);`

After introducing the "TM1650.h" library, we can create an instance using the function inside. The name of this instance is DigitalLED, which is controlled by the A5, A4 (I2C interface) pins.

Clear screen: `DigitalLED.clearDisplay();`

This is the method of the "TM1650.h" library, which is used to clear the display of the 4-Digit display module.

Digital Display: `DigitalLED.display(Number of digits, display content);`

This is the function of the “TM1650.h” library, which is used to display the number or char of the 4-Digit display module. This function has two parameters, the first one is the number of digits in the digital tube and the second one is the number or character that needs to be displayed.

Lesson 17 – Traffic light

Introduction

We have already finished learning all the modules, let us try to do some interesting projects. Do you remember the 4-Digit display module used above, is cool right? In this lesson, let's make a traffic light with a 4-Digit display and three color LEDs! The rule is that there are three colors of red, yellow and green LEDs. Whenever the corresponding color LED is lit, the 4-digit display will indicate the corresponding remaining time.

Required Parts

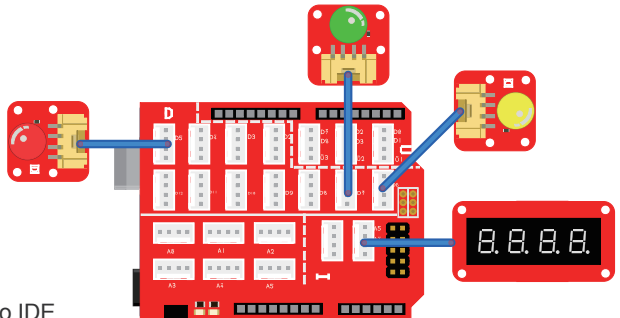
- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – 4-Digit Display x1
- Crowtail – LED(Red) x1
- Crowtail – LED(Yellow) x1
- Crowtail – LED(Green) x1
- Crowtail – Cable x4
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-4-Digit Display to Crowtail-Base shield's I port. Connect Crowtail-LED(Red) to D5 port, Crowtail-LED(Yellow) to D6 port and Crowtail-LED(Green) to D7 port. The complete connection is as follows:

Open the [P17_Traffic_Light](#) with Arduino IDE and upload it.



What will you see

The green light will light first. The remaining time of the current green light will be displayed on the 4-digit display. Then the green light goes out, the yellow light is on, and the 4-digit display will show the remaining time of the current yellow light. Finally, the yellow light goes out and the red light lights up. The 4-digit display will show the current red light remaining time prompt.

Code usage

4-Digit Display library: `#include "TM1650.h"`

"TM1650.h" is a 4-digit display library with very comprehensive use of functions for calling and using this module. Therefore, we should first import TM1650.h library so that its function can be used to call the 4-Digit display.

Integer Variables: `int redPin = 5; int yellowPin = 6 int greenPin = 7; int redValue=5; int yellowValue=2; int greenValue=5;`

Declare three colors and three values of the LEDs and assign values to them, which mean that they should be connected to D5, D6 and D7 ports of Crowtail-Base Shield. Then declare the LED lighting time of the three colors and assign a value.

Digital Display: `DigitalLED.display(Number of digits, display content);`

This is the function of the "TM1650.h" library, which is used to display the number or char of the 4-Digit display module. This function has two parameters, the first one is the number of digits in the digital tube. The second is the number or character that needs to be displayed.

Decrement: `greenValue-=1; yellowValue-=1; redValue-=1;`

The meaning of decrement is to let a value be successively subtracted from a certain number. Its code representation is generally like this: `valueName =valueName-1`, we can also abbreviate it as: `valueName-=1`.

While statement: `while(logic statement){ code to be run if the logic statement is true}`

Just like if/else statement, while statement is also a logical selection loop statement. The code will run when the logic statement inside the parentheses is true. The only difference from the if/else statement is that when the logic statement is always true inside the while statement, the code inside will always run in the while statement, will not run the outside code.

Setup and Loop: `void setup(){code to run once} & void loop(){code to run forever}`

In the `setup()` function, we need to set three LEDs as output.

In the `loop()` function, the green led will be turned on first, then in the if statement, we use the `greenValue` decrement method to make the green led light for 5 seconds, and let the 4-Digit display show the remaining time of the green light. When the green light is on for 5 seconds, turn off the green led. Then let the yellow and red lights do the same.

Lesson 18 – Smart corridor light

Introduction

When we go up the stairs at night, we all know that as soon as we make a little sound, it will automatically light up. Why? Through this lesson, you will find out the principles and make your own smart corridor lights.

In this course, we will use sound and light sensors to detect sound and brightness. When the day is dark and the sound is detected, the light will automatically turn on to illuminate. In addition, we have added another function to this corridor light, which is to help the elderly function, because the elderly may not have good eyesight and maybe too dark for them during the day, so we hope to use a touch switch to help them. When they feel too dark, they can turn on the light by pressing the touch sensor.

Required Parts

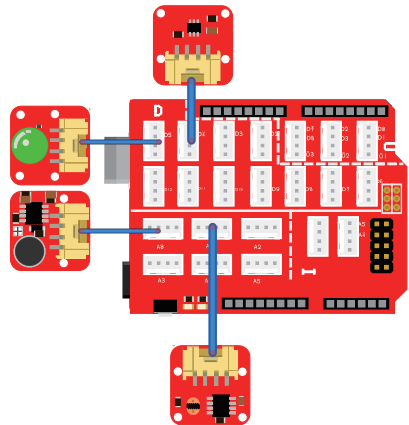
- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Sound Sensor x1
- Crowtail – Light Sensor x1
- Crowtail – Touch Sensor x1
- Crowtail – LED(Green) x1
- Crowtail – Cable x4
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Sound sensor and Crowtail-light sensor to Crowtail-Base shield's A0 and A1 port. Connect Crowtail-touch sensor to D4 port, Crowtail-LED(Green) to D5 port. The complete connection is as follows:

Open the [P18_Smart_Corridor_Light](#) with Arduino IDE and upload it.



What will you see

When you block the light sensor by hand and make a loud noise, the led will automatically turn on. When you touch the touch sensor with your hand, the led will also open. If the brightness is large, the LED will not turn on no matter how loud your sound is. Similarly, if the sound is very low, no matter how dark the brightness is, the led light will not turn on.

Code usage

Integer Variables: `int soundPin = A0; int lightPin = A1; int touchPin = 4; int ledPin = 5; int soundValue = 0; int lightValue = 0; int touchValue = 0;`

Declare the variables we will use and assign values to these variables. For the need to use the pin we usually add Pin as the end of the variable name. For the value variable that needs to be used,

we usually add Value to the variable name as the end. Here, we declare that sound sensor and light sensor are connected to analog ports A0 and A1, touch sensor and LED are connected to digital ports D4 and D5, and then all values to be used are assigned to zero.

```
Input or output: pinMode(soundPin,INPUT);    pinMode(lightPin,INPUT);  
pinMode(touchPin,INPUT);    pinMode(ledPin,OUTPUT);
```

In the setup() function, we initialize soundPin, lightPin, touchPin to input port to read the value and initialize the ledPin to output port.

```
If/else Statements: if(logic statement) {code to be run if the logic statement is true}  
else {code to be run if the logic statement is false }
```

Here, we use the if / else statement to determine if the touch sensor is touched. If yes, turn on the LED for 5 seconds, then turn it off, otherwise, read the values from lightPin and soundPin, then determine if the brightness is dark and there is sound, and if so, turn on the LED for 5 seconds, then turn it off.

Lesson 19 – Plant watering reminder system

Introduction

Is there a plant in your home? Do you often forget to water the plants and cause the death of the plants? It doesn't matter, by studying this course, you will have your own plant watering reminder system, and the plant will tell you how it is now! You will never forget to water the plants, let's do it together.

This section mainly reads the moisture value of the plant through the moisture sensor. The read moisture value will be divided into three cases and the 4-Digit Display will always display the moisture value to show you whether the plant needs watering. In addition, buzzer and two LEDs will work in corresponding plant conditions to remind you of the growth of the plant.

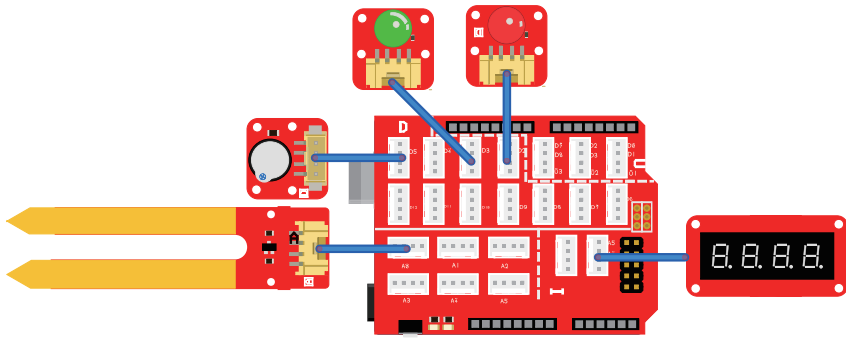
Required Parts

- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Moisture Sensor x1
- Crowtail – LED(Red) x1
- Crowtail – LED(Green) x1
- Crowtail – Buzzer x1
- Crowtail – 4-Digit Display x1
- Crowtail – Cable x5
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.

STEP2: Connect Crowtail-Moisture sensor to Crowtail-Base shield's A0 port. Connect 4-Digit Display to Crowtail-Base shield's I port. Connect Crowtail-LED(Red), Crowtail-LED(Green) and Crowtail-Buzzer to D2, D3 and D5 port. The complete connection is as follows:



Open the [P19_Plant_Watering_Reminder_System](#) with Arduino IDE and upload it.

What will you see

The moisture value read by the moisture sensor will be divided into three stages. The 4-digit display will show moisture value at any stage. The first stage is the extreme water shortage with a moisture value of less than 10%. At this time, the buzzer will beep loud to prompt water it immediately. The second is a slight water shortage with a moisture value between 10% and 20%. The LED(Red) will turn on to remind you that it needs to be watered in time. The third stage is not lacking water, and its moisture value is more than 20%. At this time, the LED(Green) will turn on to indicate that the plant does not lack water and grow well.

Code usage

4-Digit Display library: `#include "TM1650.h"`

"TM1650.h" is a 4-digit display library with very comprehensive use of functions for calling and using this module. Therefore, we should first import TM1650.h library so that its function can be used to call the 4-Digit display.

Integer Variables: `int d,t,h;`

Define three variables to store the single-digit, ten-digit, and hundred digits of the moisture value. Thus, with these three variables, we can display the corresponding number in the corresponding digital tube of the 4-digit display.

Array: `static uint8_t TubeTab[] = {}`

Its prototype is: `Array name[]`. Arrays are a form of programming in which, in order to facilitate the processing, a group of elements of the same type is organized in an unordered form. Here we create the arrays to store the numbers we want to display. If we want to get the Nth value of the array, we only need to use the array name `[N-1]`. "static" is a keyword. The advantage of variables declared with it is that it cannot be used by other files. Variables with the same name can be defined in other files without conflict. "uint8_t" is a char unsigned character, the array declared with it is an 8-bit character. The prerequisite for using it is to import the "inttypes.h" library.

Divisor and remainder: $t = \text{moistureValue}/10$; $d = \text{moistureValue}\%10$;

“t” and “d” represent the tens and single digits of the moistureValue we want to get. When the moistureValue is a two-digit number and you want to get a ten-digit value, we only need to divide the moistureValue by 10, and the result is a ten-digit value, such as $52/10=5$. If you want to get a single digit value, we only need to use the moistureValue to take the remainder of 10 to get the single digit value, such as $52\%10=2$.

Loop: `void loop(){code to run forever}`

In the loop() function, we need to read the value from the moisture sensor and then use the if / else if / else statement to determine which stage the plant's moisture value is in. For different moisture value stages, we will display the moisture value on the 4-digit display and let the buzzer or led work to tell us the plant condition.

Lesson 20 – Brightness display

Introduction

Remember the breathing lights we made? Since we can adjust the brightness of the led, can we see how the brightness of the led changes? Our 4-Digit Display will give us the answer and let the brightness level show up!

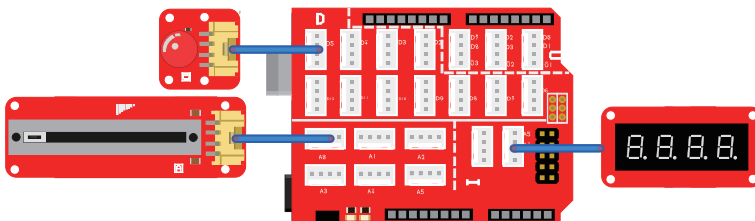
In this course, we will improve the breathing lights we've learned earlier, designed to let you know how the input module works with 4-Digit Display to display relevant input data.

Required Parts

- Crowduino UNO-SD x1
- Crowtail – Base Shield x1
- Crowtail – Linear Potentiometer x1
- Crowtail – LED(Red) x1
- Crowtail – 4-Digit Display x1
- Crowtail – Cable x3
- USB Cable x1

Hardware Connection

STEP1: Plug the Crowtail-Base Shield onto the Arduino or Crowduino Board.



STEP2: Connect 4-Digit Display to Crowtail-Base shield's I port. Connect Crowtail-LED(Red) and Crowtail-Linear Potentiometer to D5 and A0 port. The complete connection is as follows: Connect Crowtail-LED(Red), Crowtail-LED(Green) and Crowtail-Buzzer to D2, D3 and D5 port. The complete connection is as follows:

Open the `P20_Brightness_Display` with Arduino IDE and upload it.

What will you see

Slide the rheostat left and right, you will see the change in LED brightness, the corresponding LED brightness value will be displayed on the 4-digit display. When you slide to the far left and close to the Crowtail interface, led is the darkest, the minimum value is 0 on 4-Digit Display, the light is the brightest when sliding the rightmost, and the maximum value is 255 when 4-Digit Display is displayed.

Code usage

4-Digit Display library: `#include "TM1650.h"`

"TM1650.h" is a 4-digit display library with very comprehensive use of functions for calling and using this module. Therefore, we should first import TM1650.h library so that its function can be used to call the 4-Digit display.

Arduino math function `map()`: `int gapValue = map(value, fromLow, fromHigh, toLow, toHigh)`

Remap numbers from one range to another. That is, if the value is "fromLow", the mapped value will be "toLow". If the value is "fromHigh", then the mapped value will be "toHigh". When "value" is from other values from fromLow to fromHigh, it is also mapped to between toLow and toHigh in equal proportions. So here we map the value of `pmeterValue` (between 0 and 1023) to `gapValue` (between 0 and 255). For example, if the value of `pmeter` is 200, after using the `map()` function, it will become 50 and be assigned to the variable: `gapValue`.

Divisor and remainder: `h = gapValue/100; t = gapValue/10%10; d = gapValue%10;`

"h", "t" and "d" represent the hundreds digit, tens and single digits of the `gapValue` we want to get. When the `gapValue` is a three-digit number and you want to get hundred-digit value, you only need to divide the `gapValue` by 100 and the result is a hundred-digit value, such as $521/100=5$. For tens digits, you only need to divide the `gapValue` by 10, and then get the result to the remainder of 10. Then, you get a ten-digit value, such as $521/10\%10=2$. If you want to get a single-digit value, you only need to use the `gapValue` to take the remainder of 10 to get the single-digit value, such as $521\%10=1$.

Loop: `void loop(){code to run forever}`

In the loop function, we will first read the value of the sliding rheostat (0-1023), map the value to `gapValue` (0-255), and then display the value of `gapValue` based on the 4-digit display and write it as brightness to led.



MAKE YOUR MAKING EASIER



info@elecrow.com



+86 0755-23204330



www.elecrow.com



8th Floor, F-building, Fusen Industry
park, Gushu Town, Bao'an District,
Shenzhen, China.

