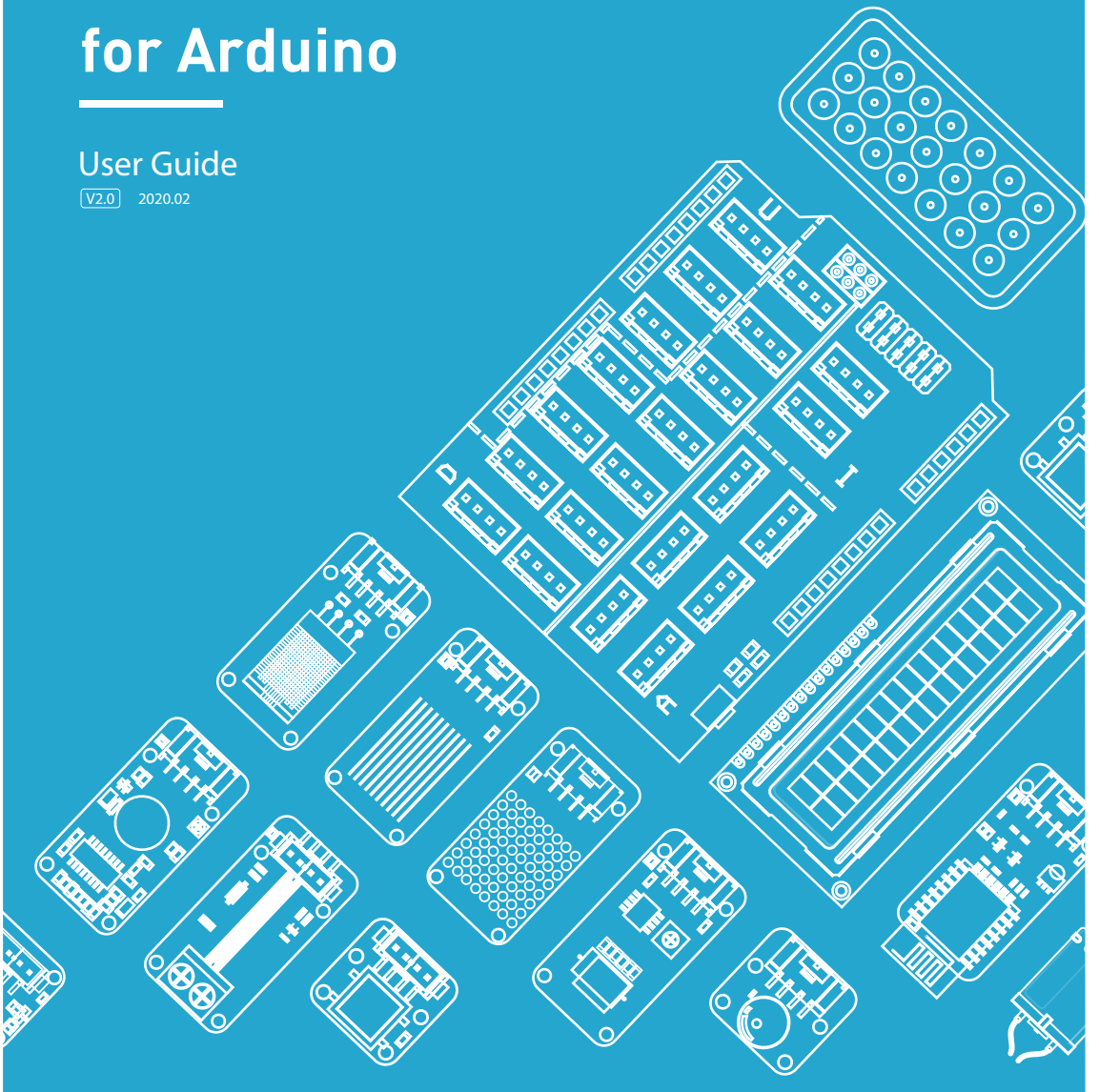
Crowtail Adavanced Kit for Arduino

User Guide

V2.0 2020.02



Content

Instruction

Welcome to the Crowtail-Advanced kit for Arduino user guide. Do you already have some knowledge of electronics and some programming skills. How do you continue to strengthen your thinking, hands-on skills, programming and innovation capabilities? This kit will lead you to a new stage! This kit contains 21 fun and creative tutorials, from simple to difficult, leading you to gradually explore and discover modules and the fun of programming and to constantly train your thinking and programming skills throughout the process to enhance your ability to innovate and confidence. Through this kit, you will learn about digital signals, analog signals, digital-to-analog conversion, automatic control systems, remote control systems, display systems, WIFI, and more. You will find that programming can create more fun, and continuously improve your ability, and finally let you create your own outstanding works, and think and solve problems more comprehensively, meticulously, and confidently!

The Crowtail-Advanced kit for Arduino includes 22 electronic modules, each module has its own feature and functions. Each module has been carefully selected from more than one hundred Crowtail modules to provide deeper learning and creative guidance for those who want a deeper understanding of hardware modules and programming knowledge. In the process, it continuously stimulates the thinking ability and creativity of learners.

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). This is the best choice for people who want to learn programming and hardware knowledge! In short, you will explore what is the core of programming and creation, and help you to have a more comprehensive thinking in the future to create your own excellent works.

Modules List

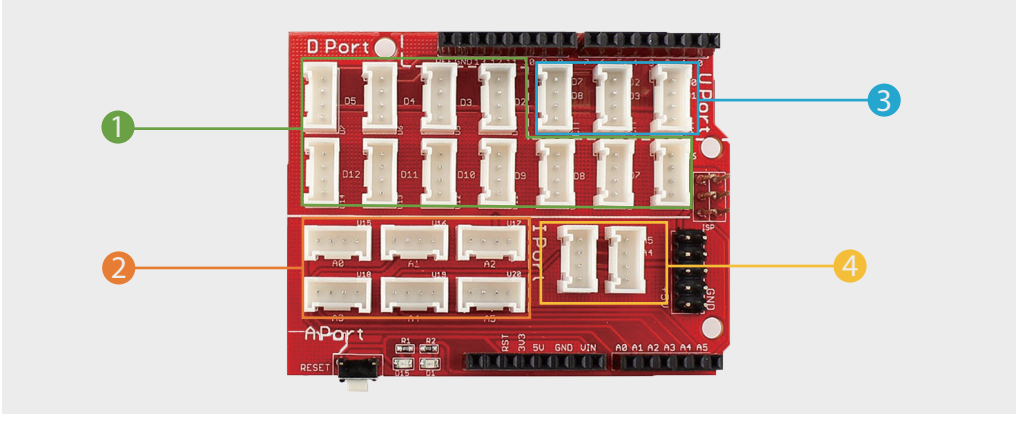
- Crowtail - Base Shield x1
- Crowtail - Button x1
- Crowtail - LED(Red) x1
- Crowtail - LED(Green) x1
- Crowtail - Vibration Sensor x1
- Crowtail - RTC x1
- Crowtail - Temperature& Humidiy Sensor x1
- Crowtail - IR Receiver x1
- Crowtail - MOSFET x1
- Crowtail - Water Sensor x1
- Crowtail - LED Matrix x1
- Crowtail - 9G Servo x1
- Crowtail - Serial Wifi x1
- Crowtail - Rotary Angle Sensor x1
- Crowtail - 3-Axis Digital Compass x1
- Crowtail - PIR Sensor x1
- Crowtail - I2C LCD x1
- Crowtail - BMP180 Barometer x1
- Crowtail - Luminance Sensor x1
- DC motor x1
- Infrared Remote Control x1
- Battery case x1

Crowtail

Welcome to the world of Crowtail! Crowtail is a modulated, ready-to-use toolset, it takes a building block approach to assemble electronics. It simplifies and condenses the learning process significantly. In our Crowtail warehouse, there are over 150 Crowtail modules and Crowtail shields!
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. It will satisfy all you need for your project!
Crowtail is a series of products that we made to solve the messy jumper wires when connecting electronic circuits. It consists of a Base Shield and some basic Crowtail modules, which helps you creating small, simple, and easy-to-assemble circuits.In other words, when you use Crowtail, your electronic project will not be a messy wiring, instead it will be a simple and easy way to manage electronic project!

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;

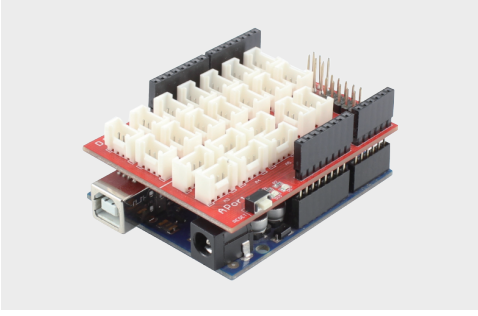
3 3 UART ports that have a mark of “U”. These interfaces can be used for UART communication such as the WIFI module or Bluetooth module;

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;

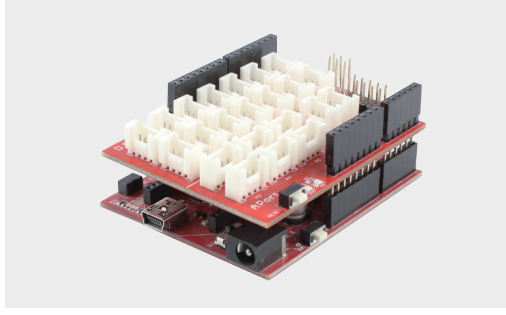
Besides, there is also a 2x5 female connector of 5V and GND for customer usages. Users can connect any electronic modules to the Base Shield with jumper wires easily.

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 realize 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 – Modules

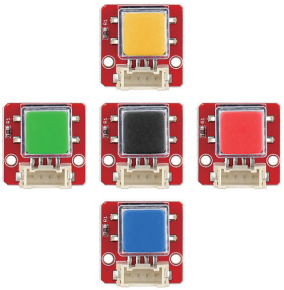
We make more than 100 kinds of electronic modules into Crowtail modules. They include a variety of sensors, displays, inputs and outputs modules, communication types include I2C, UART, digital or analog, which aim to provide more options to fully meet your electronic projects. All needs! All modules 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.

Lessons

Lesson 1 – LED Control

Introduction

The LED is the best choice to help you learn I/O pins. What you need to do is connecting the LED module to the Base Shield D ports, then download the program to the Arduino. Besides the very basic usage, you can make the LED blink with the frequency you want, thus the brightness with PWM. Actually, LED is the most popular used for human interface. In this kit, we prepared two colors of the LED, including red and green, so you can create your own LED circuit!



This momentary button outputs logic HIGH signal when pressed and logic LOW signal when released. The logic high and logic low 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.
For this lesson, we will use button and LED module to make a button control.

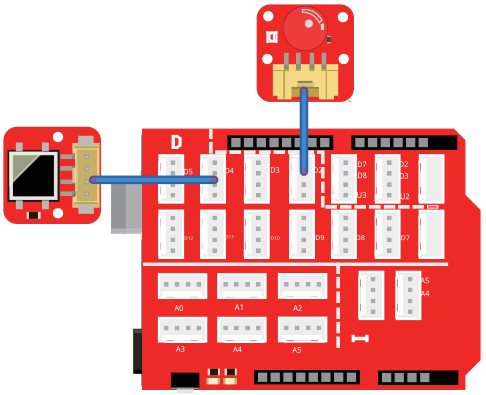
Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – LED(Red) 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 D2 port. The complete connection is as follows:



Open the P01_LED_Control 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 overview

1. Declare the variables for LED and button and assign values to them.
2. Define whether the module is output or input.
3. Read the value of the button.
4. If the button value read is HIGH(pressed), turn on the LED.
5. If the button value read is LOW(not pressed), turn off the LED.

Code usage

P01_Button_Control

```
/*
P01_Button_Control
Connect led to D2 and button to D4 port
*/
int redPin = 2;
int buttonPin = 4;
int buttonState=0;
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(redPin, OUTPUT);
  // initialize the digital pin as an input.
  pinMode(buttonPin, INPUT);
}


// the loop routine runs over and over again forever:
void loop() {
    buttonState = digitalRead(buttonPin);
      // check if the button is pressed.
    // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
      // turn LED on:
      digitalWrite(redPin, HIGH);
      }

  else {
      // turn LED off:
      digitalWrite(redPin, LOW);
  }
}
```

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 two variables that define which ports of the base shield the module should connect to and 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!

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.

Digital Input

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

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

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.

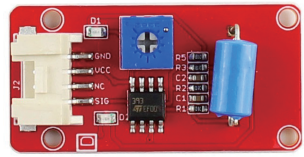
Digital Output

When you're using a pin as an OUTPUT, you can command it to be HIGH (output 5 volts) or LOW (output 0 volts). When you set it to HIGH, for digital output modules, it means work. When you set it to LOW, for digital output modules, it means don't work. For example, the led will light on(work) when it is set to HIGH and it will light off(don't work) when it is set to LOW.

Lesson 2 – Vibration detector

Introduction

The Crowtail-Vibration Sensor is a high sensitivity non-directional vibration sensor. When the module is stable, the circuit is turned on and the output is low. When the movement or vibration occurs, the circuit will be briefly disconnected and output high. At the same time, you can also adjust the sensitivity according to your own needs. It is widely used to report the theft alarm, intelligent car, earthquake alarm, motorcycle alarm, etc.
In this lesson, we will use vibration sensor and LED to make a vibration detector that you can use to detect earthquakes and theft.



Required Parts

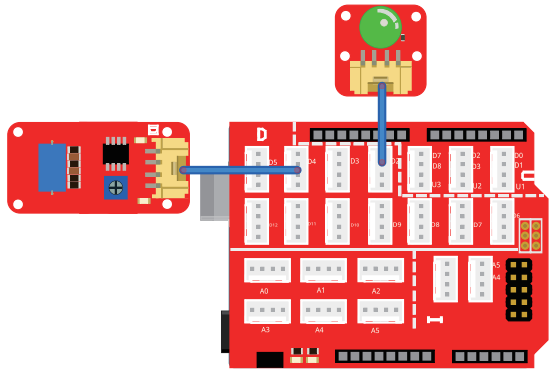
- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Vibration Sensor x1
- Crowtail – LED x1
- Crowtail – Base shield x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-Vibration sensor to Crowtail-Base Shield D4 port and Crowtail-LED to D2 port. The complete connection is as follows:

Open the P02_Vibration_Detector with Arduino IDE and upload it.



What will you see

Put the Vibration sensor on the table or hold it in your hand and shake it. You will see the LED light on as you shake. When you stop shaking, the LED will turn off.

Code overview

1. Declare the variables for vibration sensor and LED and assign values to them.

2. Define whether the module is output or input.
3. Read the value of the vibration sensor.
4. If the value of the vibration sensor is LOW(shaked), turn on the LED.
5. If the value of the vibration sensor is HIGH(not shaked), turn off the LED.

Code usage

Integer Variables: int ledPin = 2; int vibrationPin = 4; int vibrationState=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 two variables that define which ports of the base shield the module should connect to and a variable called “vibrationState” of type int(integer) and assign it a value of 0 to record the status of the vibration sensor. Don't forget that variable names are case-sensitive!

Input or Output: pinMode(vibrationPin,INPUT); pinMode(ledPin,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 and the vibration sensor module as a input.

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 here, if the vibration sensor is vibrating, the LED will light on and when the vibration sensor is not vibrating, the LED will light off.

Digital Output: digitalWrite(ledPin, HIGH); digitalWrite(ledPin, LOW);

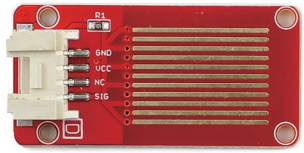
When you're using a pin as an OUTPUT, you can command it to be HIGH (output 5 volts) or LOW (output 0 volts).

Lesson 3 – Raining reminder

Introduction

The water sensor detects water by having a series of exposed traces connected to ground and interlaced between the grounded traces are the sensor traces. The sensor traces have a weak pull-up resistor of 1 MΩ. The resistor will pull the sensor trace value high until a drop of water shorts the sensor trace to the grounded trace. With the digital I/O pins of Crowduino/Arduino, you can detect the amount of water induced contact between the grounded and sensor traces.

Do you have the experience of getting clothes wet by the rain? You won't experience such a disaster again. In this course, we will use water sensor and make a rain reminder, so that it can immediately remind you to collect clothes when it starts to rain!



Required Parts

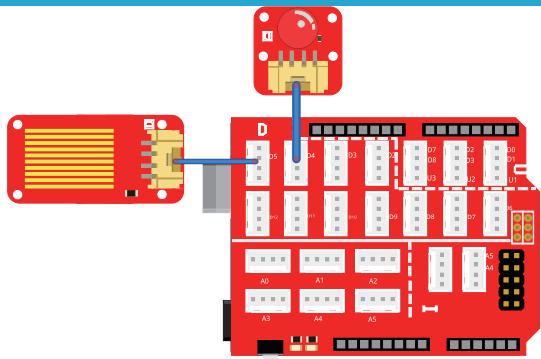
- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – LED(Red) x1
- Crowtail – Water Sensor x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

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

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

Open the P03_Raining_Reminder with Arduino IDE and upload it.



What will you see

When you put the water drop on the sensor traces of the water sensor, the led will light on. When you dry the water, the led will light off.

Code overview

1. Declare the variables for water sensor and LED and assign values to them.
2. Define whether the module is output or input.
3. Read the value of the water sensor.
4. If the value of the water sensor is LOW(water detected), led light on.
5. If the value of the water sensor is HIGH(no water detected), led light off.

Code usage

Integer Variables: int ledPin = 4; int waterPin = 5; int waterState = 0;

First, we declare two variables named ledPin and waterPin and assign them 4 and 5 respectively.

This means we can now use D4 and D5 ports for LEDs and water sensors.Then, declare a variable named waterState to store the read water sensor value.

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

In the setup() function, we initialize the water sensor as an input to detect if there is water on the water sensor and initialize the led to output to show you the result of the water 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 }

Here, we use the if/else statement to execute the program that needs to be executed when the water is detected or the water is not detected. If the water is detected, turn led on, otherwise, turn led off.

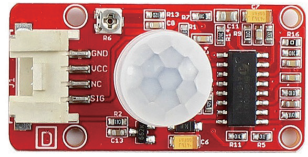
Is equal to: buttonState == HIGH

Different from touch sensor and button, when water is detected, the state of the water sensor is LOW, and the state of water not detected is HIGH. It means that led will light on when water sensor value is LOW to remind us it is raining and light off when water sensor value is HIGH.

Lesson 4 – Intelligent corridor light

Introduction

Crowtail - PIR Motion Sensor(Passive Infrared Sensor) can detect infrared signals caused by motion. If the PIR sensor notices the infrared energy, the motion detector is triggered and the sensor outputs HIGH on its SIG pin. The detecting range can be adjusted by a potentiometer soldered on its circuit board, the max detecting range of it up to 6 meters.
How about use PIR motion sensor and LED to make a smart corridor light?



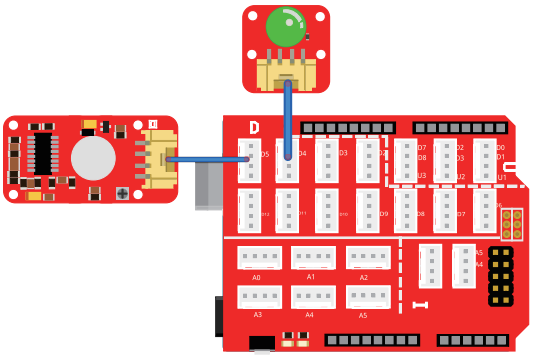
Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – PIR Motion Sensor x1
- Crowtail – LED(Green) x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-LED to Crowtail-Base shield's D4 and Crowtail-PIR motion sensor to D5 port. The complete connection is as follows:



Open the P04_Intelligent_corridor_light with Arduino IDE and upload it.

What will you see

When you wave or walk within the range of the PIR motion sensor, the LED will turn on for 5 seconds. If there is no movement of the object within the sensing range of the PIR motion sensor, the LED will not light up.

Code overview

1. Macro definitions of PIR motion sensor pin and LED pin.
2. Define whether the module is output or input.
3. Create functions to turn LED on and turn LED off separately.
4. Create a boolean type object motion detection function that returns true when the object moves, otherwise return false.
5. Determine the value returned by the object motion detection function in the loop() function to call turnOnLED() and turnOffLED functions to turn the LED on or off.
6. Repeat loop() function.

Code usage

Macro definition: #define LED 4 #define PIR_MOTION_SENSOR 5

The prototype of the macro definition constant is #define [MacroName] [MacroValue]. What is the difference between a macro definition constant and a variable? First, Macro-defined constants cannot be changed while the program is running. Variables can be changed. Second, the variable can be used inside the function defined by it, but the life cycle end when the function ends. The macro defines the constant until the entire program runs, the life cycle ends.

Boolean type: boolean isPeopleDetected(){}

Objects created with boolean types (including variables and methods) have only two values: true and

false. By judging the returned value, we can easily execute the corresponding code according to the difference of values.

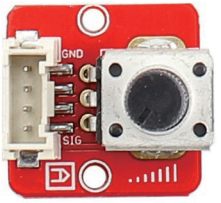
Modular programming: void turnOnLED(){} void turnOffLED(){} boolean isPeopleDetected(){}

Modular programming allows us to better manage and call our code, such as a module code problem, we only need to modify the code inside the module, without modifying the code of the entire program. In addition, modularizing the code allows us to implement functions with simpler logic.

Lesson 5 – Breathing light

Introduction

This rotary angle sensor may also be known as potentiometer twig that produces analog output between 0 and VCC (5V DC with Crowduino) on its SIG pin. The angular range is 300 degrees with a linear change in value. The resistance value is 10k ohms, perfect for Arduino use. Some applications like smart light control, volume control, only you can not think of things, no impossible things! For this lesson, we will use rotary angle sensor and led to make a breathing light, you can control the brightness of led like human breathing!



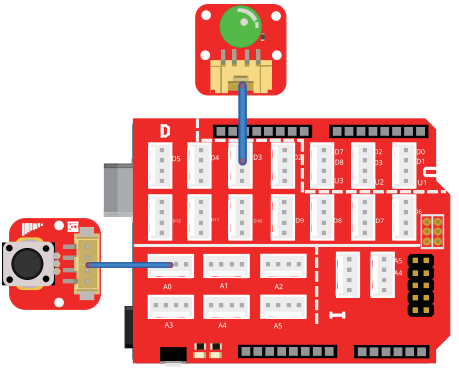
Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – Rotary Angle Sensor x1
- Crowtail – LED(Green) x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-LED to Crowtail-Base shield's D3 and Crowtail-Rotary angle sensor to A0 port. The complete connection is as follows:



Open the P05_Breathing_Light with Arduino IDE and upload it.

What will you see

When turning the knob on the rotary angle sensor, the brightness of the led will change. When you turn to the limit in one direction, if the led is off, then if you turn to the limit in the other direction, the brightness of the led is the maximum.

Code overview

1. Declare the pin of led and rotary angle sensor. Declare some constants to be used.
2. Initialize the serial monitor and initialize the rotary angle sensor to input and led to output.
3. Use getDegree() function to get the degrees of rotary angle sensor.
4. Print the degrees of rotary angle sensor and map the degrees as led brightness.

Code usage

Serial print: Serial.println("The angle between the mark and the starting position:"); Serial.println(degrees);

Serial print has two printing methods, one is Serial.print(), the other is Serial.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. 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.

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 Output: analogWrite(LED,brightness);

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 are 256.

Analog Input: int sensor_value = analogRead(ROTARY_ANGLE_SENSOR);

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 sensor_value.

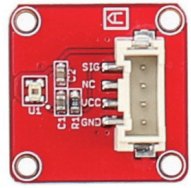
Integer function: int getDegree(){}

Unlike void function (), an integer function will return a number after running, and this number is an integer number. In short, this integer function() will return an integer number each time it is run.

Lesson 6 – Calculating light intensity

Introduction

Crowtail - Luminance Sensor using APDS-9002 as lumens Sensor, provide the linear transform lumen intensity for the output voltage levels. And APDS-9002 spectrum and human eye is extremely close to. It is very suitable for the field of AI applications.
In this lesson, we will use Crowtail-Luminance Sensor to obtain the light intensity and print it in serial monitor.



Required Parts

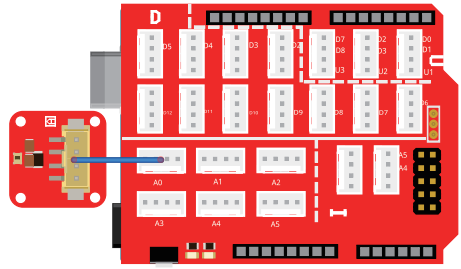
- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – Luminance Sensor x1
- Crowtail – Cable x1
- USB Cable x1

Hardware Connection

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

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

Open the P06_Calculating_Light_Intensity with Arduino IDE and upload it.



What will you see

Open the serial monitor, you will see the monitor is printing the voltage and luminance. When you block the luminance sensor with your hand, you will see that the voltage and luminance detected by the luminance sensor is significantly reduced.

Code overview

1. Create two arrays for use in the FmultiMap() function.
2. Initialize the serial monitor in the setup() function and set its baud rate to 9600.
3. Use the readAPDS9002Vout() function to get the voltage read by the luminance sensor.
4. Use the FmultiMap() function to get the real measurement point and more accurate data from the sensor.
5. Use the readLuminance() function to get the luminance of the luminance sensor.
6. In the loop() function, print the voltage and luminance of the luminance sensor and then repeat.

Code usage

Array: float VoutArray[] = {} float LuxArray[] = {}

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 and they will be used in FmultiMap() function. float means that the array we created is floating point, not integer. Pay attention to VoutArray[], it will act _in array in FmultiMap() function, so it should have increasing values.

Float function(): float readAPDS9002Vout(uint8_t analogpin){}

Unlike void function (), a float function will return a number after running, and this number is a floating point number. In short, this float function () will return a floating point number each time it is run. “analogpin” is the parameter of the function, it will be passed to the function, "uint8_t" is an unsigned character that declares that each element in the array occupies 8 bits of storage.

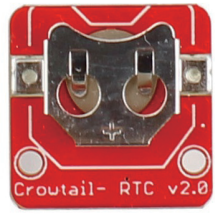
FmultiMap function(): float FmultiMap(){}

The prototype of the function is: float FmultiMap (float val, float * _in, float * _out, uint8_t size).FmultiMap () is a function can be applied to the detection of sensors that do not change linearly. Since I only need floating point numbers, the function returns a floating point number. The parameters are the int val which comes from the analogRead() function, an array of input values and an array of corresponding output values and a parameter to indicate the size of the array's used. NOTE: the input array must be a monotone increasing array of values. You can see more uses of this function here “https://playground.arduino.cc/Main/MultiMap/”.

Lesson 7 – Get current time

Introduction

This tiny RTC module is based on the clock chip DS1307 which communicates with microcontrollers with I2C protocol. The clock/calendar provides seconds, minutes, hours, day, date, month and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. Besides, this module is really low power consumption, it can serve you more than a month with a CR1220 battery. If you want to make your own electronic watch, you need to generate the correct time, so in this lesson, let's take a look at how to use RTC to get the time.



Required Parts

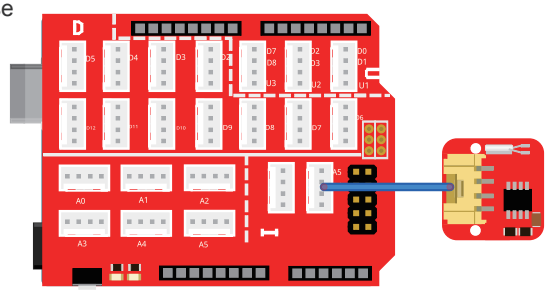
- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – RTC x1
- Crowtail – Cable x1
- USB Cable x1

Hardware Connection

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

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

Open the downloaded folder "Crowtail-Advanced kit for Arduino demo code", navigate to the folder lib-> RTC, and add RTC to the Arduino library. Open the P7_Get_Current_Time with Arduino IDE and upload it.



What will you see

Open the serial monitor of the Arduino IDE and you will see that the monitor is printing out your local time.

Code overview

1. Import I2C library and RTC standard library.

2. Initialize the serial port to set the baud rate to 9600, initialize the Wire library and the real time clock.
3. Determine if the RTC is running and initialize the time in the chip with the current date and time.
4. Get current date and time information saved by RTC.
5. Print the preliminary year, month, day, hour, minute and second information in the decimal output.

Code usage

Import library: #include <Wire.h> #include "RTClib.h"

Arduino communicates with the real-time clock through the I2C bus. In order to use this bus, the compiler must be notified to join the library. In addition, we also need to import the RTC library to get the time.

Instance: RTC_DS1307 RTC;

The meaning of this statement is to create an instance, RTC_DS1307 is a class in RTClib.h, RTC is to create an instance of RTC_DS1307 class, this instance includes some related functions and variables.

Initialization: Serial.begin(9600); Wire.begin(); RTC.begin();

Initialize the serial monitor and set the baud rate to 9600 to initialize the I2C bus and real-time clock.

Initialize the RTC chip: RTC.adjust(DateTime(__DATE__, __TIME__))

Initialize the RTC chip with the current date and time.

RTC_DS1307's now() function: DateTime now = RTC.now();

“now()” is another function of RTC_DS1307. It returns a DateTime instance and saves the current date and time information. After running this statement, we can know the current month through now.month() and get the current minutes through now.minute(). And then years, days, hours and seconds.

Serial print: Serial.print(now.year(), DEC); Serial.print('/');

Serial.print() is used to print the string or variable that you want to output, it receives two parameters, the first is the transmitted value, the second is the format of the transfer, DEC is the decimal, and OCT is the octal, and HEX is the hexadecimal. 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.

Lesson 8 – LCD display

Introduction

Crowtail-I2C LCD includes LCD1602 and MCP23008 modules. Unlike ordinary LCDs, which require

many pins, the Crowtail-I2C LCD only needs 4 pins to control everything. It's very popular where display is needed.
Here, we will use Crowtail-I2C LCD to display our first greeting.

Required Parts

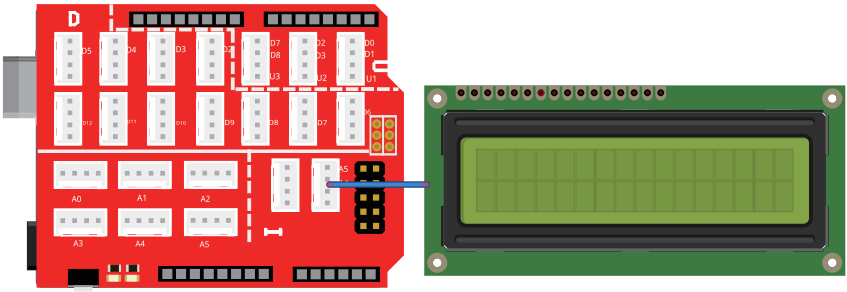
- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – I2C LCD x1
- Crowtail – Cable x1
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-I2C LCD to Crowtail-Base shield's I port. The complete connection is as follows:

Open the downloaded folder "Crowtail-Advanced kit for Arduino demo code", navigate to the folder lib-> LiquidCrystal, and add LiquidCrystal to the Arduino library. Open the P08_LCD_Display with Arduino IDE and upload it.



What will you see

You will first see the LCD light up and display the greeting "hello world" on the first line. After a second, the second line of the LCD will display "good bye" and finally the LCD will go out.

Code overview

1. Import the I2C and LCD library.
2. Create an LCD instance object.
3. Set up the LCD's number of rows and columns.
4. Print message on the LCD.
5. Clear LCD display and turn off the backlight of LCD.

Code usage

Import library: #include <Wire.h> #include "LiquidCrystal.h"

Import the library into your program so that you can work with modules using the functions built into the library. Usually, we use #include <library name> or #include "library name" to import the library. Wire.h is a library of I2C modules, before we use I2C modules, it is necessary for us to import this library. LiquidCrystal.h is a library of LCD, it can provide many useful and convenient functions for us to use LCD.

Create an instance: LiquidCrystal lcd(0);

After introducing the LiquidCrystal.h library, we can create an instance using the function inside. The name of this instance is lcd, which is connected to 0 (I2C address). Surely, if you want, you can change the name of the instance, such as LiquidCrystal I2C_LCD(0). But after changing the name of the instance object, don't forget to change the instance object below.

LCD set up: lcd.begin(16,2)

"lcd.begin()" function is built-in LiquidCrystal.h library, its prototype is lcd.begin(rows,columns). Here, we set up the LCD's number of rows is 16 and number of columns is 2.

LCD print: lcd.print("hello, world!");

"lcd.print()" function is built-in LiquidCrystal.h library, the content in parentheses is what you need to print. The content in parentheses can be a variable or a string.

LCD setCursor: lcd.setCursor(0, 1);

"lcd.setCursor()" function is built-in LiquidCrystal.h library, its prototype is lcd.setCursor(rows,columns). We use this function to set which row and column to display the message.

LCD clear and LCD backlight: lcd.clear();lcd.setBacklight(LOW);

"lcd.clear()" function and lcd.setBacklight() function are built-in LiquidCrystal.h library, lcd.clear() is to clear the display on the LCD and lcd.setBacklight() is to turn on/off the backlight of the LCD.

Lesson 9 – Electric watch

Introduction

Have you ever thought about making an electronic watch for yourself? This must be very cool! Today, we will use the electronic modules RTC and LCD learned above to make an electronic watch. Bring this watch to make yourself the coolest person on the street!

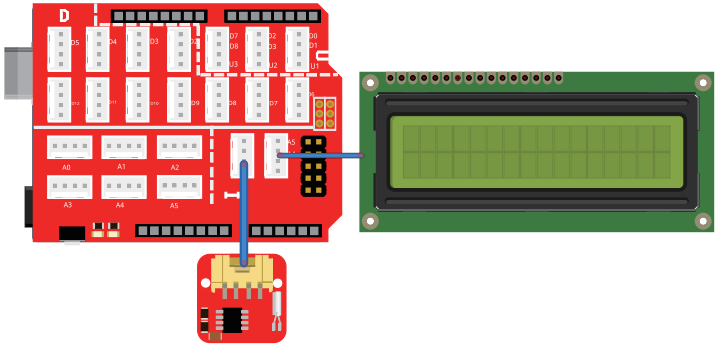
Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – I2C LCD x1
- Crowtail – RTC x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-I2C LCD and Crowtail-RTC to Crowtail-Base shield's I port. The complete connection is as follows:



Open the P09_Electric_Watch with Arduino IDE and upload it.

What will you see

You will see the current year, month, and day information displayed on the first line of the LCD, the current hour, minute, and second information displayed on the second line, and the time displayed on the LCD is constantly updated over time.

Code overview

1. Import the I2C, RTC and LCD library.
2. Create an RTC instance and lcd instance.
3. Initialize the serial monitor, I2C and RTC.
4. Determine if the RTC is running and initialize the time in the chip with the current date and time.
5. Initialize the lcd and get the current time from RTC module. Then print the current time information on the LCD.

Code usage

Import library: #include <Wire.h> #include "RTClib.h" #include "LiquidCrystal.h"

Import the necessary of using RTC and LCD. Both RTC and LCD are I2C modules, so we need to import the I2C Wire.h library. RTClib.h and LiquidCrystal.h are the libraries of RTC module and LCD module, they both provide many convenient and useful functions to use.

Create instance: RTC_DS1307 RTC; LiquidCrystal lcd(0);

After import the RTC and LCD library, we can use the function in the library to create instances for us to use RTC and LCD. Here we create an RTC instance and lcd instance.

Initialize the RTC chip: RTC.adjust(DateTime(__DATE__, __TIME__));

Initialize the RTC chip with the current date and time.

RTC_DS1307's now() function: DateTime now = RTC.now();

“now()” is another function of RTC_DS1307. It returns a DateTime instance and saves the current date and time information. After running this statement, we can know the current month through now.month() and get the current through now.minute(). The number of minutes, and so on, year, day, hour, second.

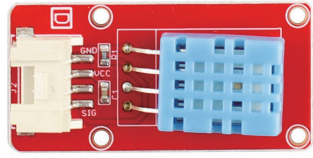
LCD print: lcd.print();

“lcd.print()” function is built-in LiquidCrystal.h library, the content in parentheses is what you need to print. The content in parentheses can be a variable or a string.

Lesson 10 – Temperature&Humidity detecting system

Introduction

This module can help you detect the temperature and humidity of the environment of your house. The module contains a DHT11 temperature & humidity sensor that is a complex sensor with a calibrated digital signal out. It uses digital module acquisition technology and the temperature&humidity sensor technology. The sens or consists of a resistance type moisture element and an NTC temperature measuring element. Because of single wire serial interface, it is easy to use the module.
This lesson, we will use Crowtail-Temperature&humidity sensor and Crowtail-I2C LCD to make a temperature and humidity detecting system, which aim to help you get the environment’s temperature and humidity around you.



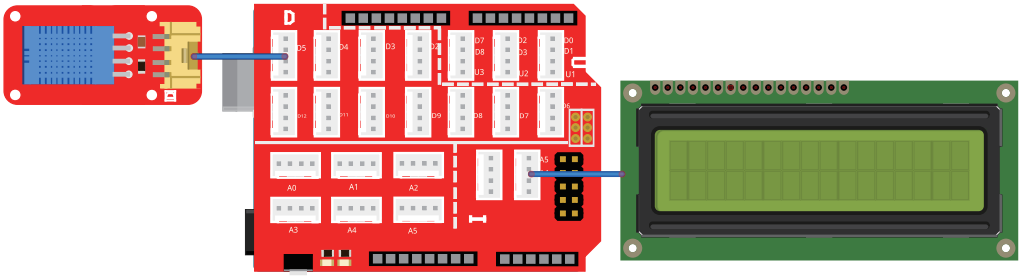
Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – I2C LCD x1
- Crowtail – Temperature&Humidity Sensor x1
- Crowtail – Cable x2
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-I2C LCD and Crowtail-Temperature&Humidity Sensor to Crowtail-Base shield's I port and D5 port. The complete connection is as follows:



Open the downloaded folder "Crowtail-Advanced kit for Arduino demo code", navigate to the folder lib-> DHT, and add DHT to the Arduino library. Open the P10_Temperature&Humidity_Detecting_System with Arduino IDE and upload it.

What will you see

You will see that the first column of the LCD displays the humidity information of the current environment, and the second column displays the temperature information of the current environment. When you blow on the Temperature & Humidity Sensor, you will see Temperature and humidity will also change on LCD.

Code overview

1. Import the LCD and Temperature&Humidity Sensor library.
2. Macro definitions of Temperature&Humidity Sensor pin and DHT11.
3. Create a DHT instance and lcd instance.
4. Initialize the serial monitor, Temperature&Humidity Sensor and lcd.
5. Read the humidity and temperature information from Temperature&Humidity Sensor.
6. Print humidity and temperature information on LCD.

Code usage

DHT library: #include "DHT.h".

DHT.h is a library based on temperature and humidity sensors. It contains multiple dht temperature and humidity sensors, such as dht11, dht22 and other sensors, so we will use the built-in function to read the temperature and humidity values.

DHT type: #define DHTTYPE DHT11.

The macro defines a DHT type. Because DHT.h contains libraries for multiple DHT sensors, we need to tell the program which type of DHT sensor we need to use when we create an instance object for DHT.

Create DHT instance: DHT dht(DHTPIN, DHTTYPE);

The prototype for creating a DHT instance object is DHT name (uin8_t pin, uin8_t type, uin8_t count). The “pin” represents the pin of the DHT sensor connected to the Arduino. The “type” represents the type of the DHT sensor. Our Crowtail–Temperature & Humidity Sensor uses DHT11. The “count” is an optional parameter.

Read humidity and temperature: float h = dht.readHumidity(); float t = dht.readTemperature();

The prototype for creating a DHT instance object is DHT name (uin8_t pin, uin8_t type, uin8_t count). The “pin” represents the pin of the DHT sensor connected to the Arduino. The “type” represents the type of the DHT sensor. Our Crowtail–Temperature & Humidity Sensor uses DHT11. The “count” is an optional parameter.

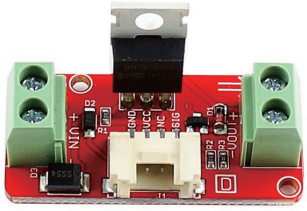
Isnan(): if (isnan(t) || isnan(h)) {}

The role of is isnan() to determine whether the number in the brackets is a number. Isnan() is a short of “is not a number”, if the number in the parentheses is not a number, it returns TRUE, otherwise, it returns FALSE.

Lesson 11 – PWM control

Introduction

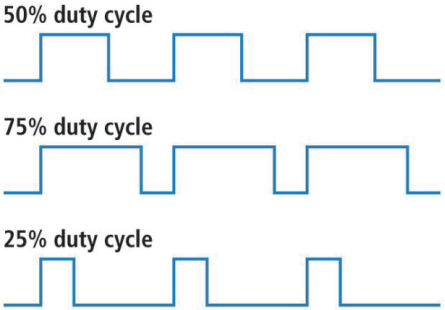
Crowtail-MOSFET enables you to control high voltage items (such as 50VDC) and low voltages (such as 5V) on a microcontroller. A MOSFET is also a switch. There are two screw terminals on the board. One for input power and the other for the device you want to control. Crowtail-MOSFET transfers power from one end to the other when closed. However, if there is no external power source, your device can still get power from the microcontroller through the Crowtail interface. Similar to relays, MOSFET is often used in automatic control systems. It uses a small power source to control a large power source so that our electricity can be controlled more safely and conveniently. In this lesson, we will use MOSFET and motor to make a PWM controlled fan, so that we can control the speed of the fan as we want!



What is PWM

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.

When talking about how long a PWM signal is on, this is referred to as duty cycle. Duty cycle is measured in percentage. The percentage of duty cycle specifically describes the percentage of time a digital signal is on over an interval or period of time. The variation in the duty cycle tells the motor how fast it should turn.



Required Parts

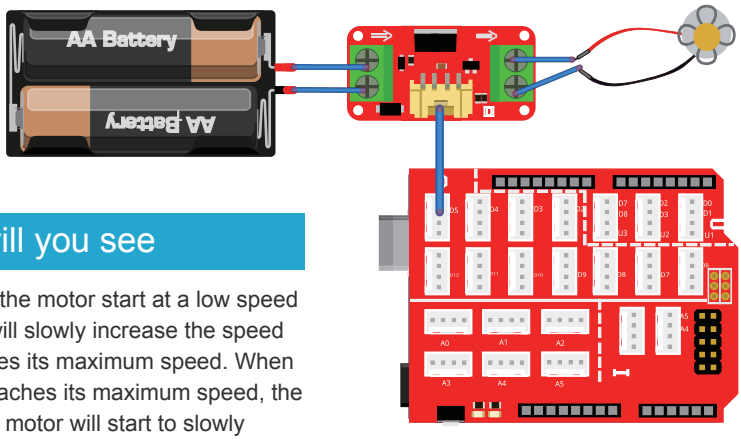
- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – MOSFET x1
- Crowtail – Cable x1
- Battery Case x1
- DC Motor x1
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-MOSFET to Crowtail-Base shield's D5 port. Connect battery case and DC motor to Crowtail-MOSFET. The complete connection is as follows:

Open the P11_PWM_Control with Arduino IDE and upload it.



What will you see

You will see the motor start at a low speed and then it will slowly increase the speed until it reaches its maximum speed. When the motor reaches its maximum speed, the speed of the motor will start to slowly decrease and reach its minimum speed.

Code overview

1. Declare the pin the MOSFET is connected to.
2. Define MOSFET as an output module.
3. Slowly increase the speed of the motor.
4. Slowly decrease the speed of the motor.

Code usage

Integer Variables: int mosfetPin = 5; int delayTime = 50;

Declare two variables named mosfetPin and delayTime, which means that we can use MOSFET through D5 port and the value of delayTime is 50.

For() statement: for (<initialiser code> ; <condition test expression> ; <iterator expression>){code to be run if condition test expression is true }

The Arduino for loop provides a mechanism to repeat a section of code depending on the value of a variable. So you set the initial value of the variable, the condition to exit the loop (testing the variable), and the action on the variable each time around the loop. Initialiser section: The initial value of the control variable. Condition Section: The condition to stop the loop. Iterator Section: The loop variable action (increment or decrement).

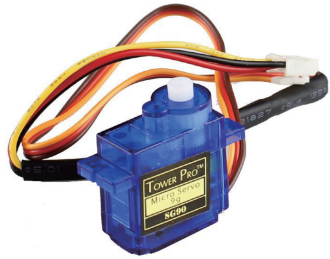
Analog output: analogWrite(mosfetPin, i);

The prototype of the analogWrite () function is analogWrite(pin,value). We know the analogWrite() function is to write the value on an analog pin. Actually, writing the analog value is controlled by PWM, so we can directly use analogWrite () to write the PWM wave. Similar to digitalWrite() function, it takes two parameter, 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. Note that value has 256 levels, from 0-255.

Lesson 12 – Servo control

Introduction

Tower Pro SG90 is a high quality, low-cost servo for all your mechatronic needs. It comes with a 4-pin power and control cable, mounting hardware. Servo is used in many intelligent situations, such as automatic doors, robots, aerial models, etc. It can be said that the servo is almost an indispensable module in the field of intelligent control, so in this lesson, we will learn how to use the servo and make you can make your project easier by using it.



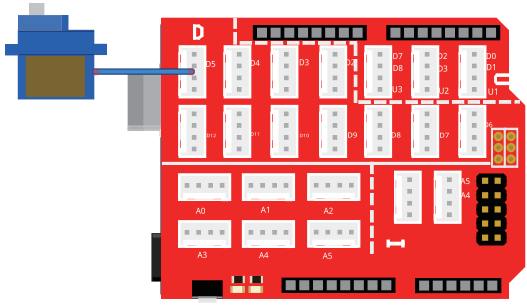
Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – 9G Servo x1
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-9G Servo to Crowtail-Base shield's D5 port. The complete connection is as follows:



Open the downloaded folder "Crowtail-Advanced kit for Arduino demo code", navigate to the folder lib-> Servo, and add Servo to the Arduino library. Open the P12_Servo_Control with Arduino IDE and upload it.

What will you see

You will see that the servo is rotated 180 degrees from 0 degrees and then rotated back to 0 degrees from 180 degrees.

Code overview

1. Import the servo library.
2. Create a servo object to control servo.
3. Declare a variable to store the servo position.
4. Attach the servo on D5 port.
5. Servo goes from 0 degrees to 180 degrees.
6. Servo goes from 180 degrees to 0 degrees.

Code usage

Servo library: #include <Servo.h>

Import the servo library. Servo.h is a wonderful library of the servo, it provides a very convenient function to control rotation of servo.

Create servo instance: Servo myservo;

Create a servo object to control the servo. After import the servo library, we need to create a servo object to tell that need to control servo.

Attach function: myservo.attach(5);

Declare which pin the servo is connected to. Different from the other modules which use variable to initialize which pin them should be connected to, servo use ServoObject.attach() function to tell which pin it is connected to. For example, our code is to connect the servo to D5 port.

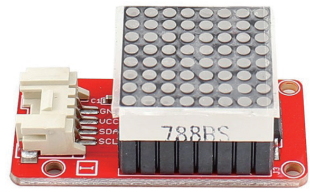
For() statement: for (<initialiser code> ; <condition test expression> ; <iterator expression>){code to be run if condition test expression is true }

The Arduino for loop provides a mechanism to repeat a section of code depending on the value of a variable. So you set the initial value of the variable, the condition to exit the loop (testing the variable), and the action on the variable each time around the loop. Initialiser section: The initial value of the control variable. Condition Section: The condition to stop the loop. Iterator Section: The loop variable action (increment or decrement). In the first for() statement, we initialize pos to 0, then judge if the pos less than 180, if yes, pos plus 1 and run the code in for statement.

Lesson 13 – Matrix display

Introduction

This Crowtail-LED Matrix uses the HT16K33 which is a neat little chip that has the ability to drive a multiplexed 8x8 matrix (that's 64 individual LED). We offer an LED matrix to you-blue. You need to pay attention to that the driver can turn LED on and off but does not have the ability to individually PWM dim them.
In this lesson, we will learn how to use Crowtail-LED Matrix to show the string. Let's use this module to make a billboard and show a personal advertisement!



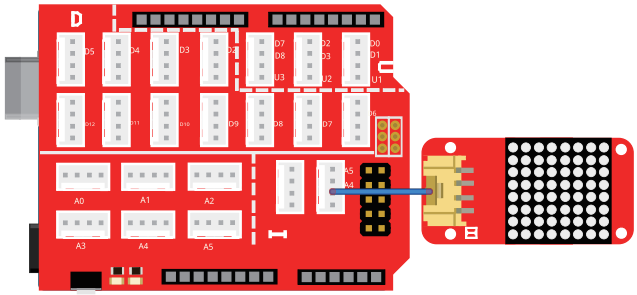
Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – LED Matrix x1
- Crowtail – Cable x1
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-LED Matrix to Crowtail-Base shield's I port. The complete connection is as follows:



Open the downloaded folder "Crowtail-Advanced kit for Arduino demo code", navigate to the folder lib-> Adafruit_LED_Backpack, and add Adafruit_LED_Backpack to the Arduino library. Open the P13_Matrix_Display with Arduino IDE and upload it.

What will you see

You can see that the LED matrix scrolls in one direction to display "Hello", and then rotates to 90 degrees, and scrolls to display "LED Matrix".

Code overview

1. Import the I2C and LED Matrix library.
2. Create a matrix instance object to control LED Matrix.
3. Initialize and pass the I2C address of LED Matrix.
4. Set text display size and scroll display on LED Matrix.
5. Use for() statement to scroll display "Hello".
6. Change text display direction.
7. Use for() statement to scroll display "LED Matrix".
8. Change to default text display orientation.

Code usage

Matrix library: #include "Adafruit_LEDBackpack.h"

"Adafruit_LEDBackpack.h" is a basic library for an 8x8 LED matrix. The library is written for the Arduino and will work with any Arduino as it just uses the I2C pins. The code is very portable and can be easily adapted to any I2C-capable micro.

Create LED Matrix instance: Adafruit_8x8matrix matrix;

Create an LED Matrix instance after we import the Adafruit_LEDBackpack.h library. The name of the instance object is matrix.

Initialize LED Matrix: matrix.begin(0x70);

Initialize the LED Matrix instance object and pass the I2C address. The instance object matrix address is 0x70. The reason that I2C modules can be cascaded is that each module above has its own I2C address so that they will not be confused when communicating.

Text size: matrix.setTextSize(1);

We use the matrix.setTextSize() function to set the text size displayed on the LED matrix. 1 is a good choice for 8x8 LED matrix display. You can try changing 1 to 2 and see what the LED matrix will look like.

Wrap of scroll: matrix.setTextWrap(false);

Using matrix.setTextWrap() function we can set whether the text display on LED Matrix is wrapped or scroll. If the parameters in the brackets are true, the display text effect is wrapped, if it is false, the display text effect is scroll.

LED Matrix ON or OFF: matrix.setTextColor(LED_ON);

Set whether the LED is on or off, when you choose “LED_ON”, you can see the effect of led light, if you choose “LED_OFF”, you will not see any effect.

Set Cursor: matrix.setCursor(x,0);

Set the position of the cursor. We use the matrix.setCursor() function to position the cursor. For example, the cursor is set to display from the horizontal x and vertical 0. Remember that 0 is the first and 1 is the second.

Display direction: matrix.setRotation(3); matrix.setRotation(0);

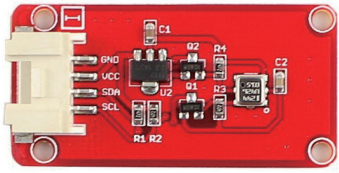
The matrix.setRotation() function is to set the direction of text display on LED Matrix. The parameters in parentheses represent the display direction, 0 is the default angle, and the value of the parameter differs by 1 for every 90 degrees.

Lesson 14 – Get atmospheric pressure

Introduction

The BMP180 offers a pressure measuring a range of 300 to 1100 hPa with an accuracy down to 0.02

hPa in advanced resolution mode. It's based on piezo-resistive technology for high accuracy, ruggedness and long term stability. These come factory-calibrated, with the calibration coefficients already stored in ROM. What makes this sensor great is that it is nearly identical to its former rev, the BMP085! This lesson we will try to use Crowtail-BMP180 Barometer to get the pressure of atmospheric.



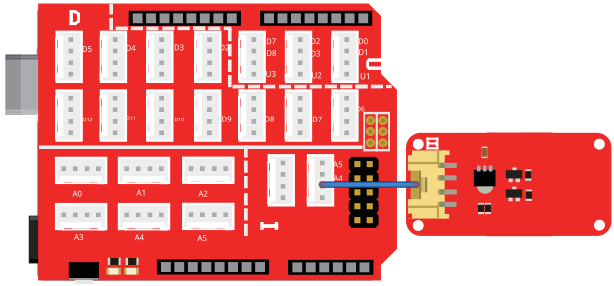
Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – BMP180 Barometer x1
- Crowtail – Cable x1
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-BMP180 Barometer to Crowtail-Base shield's I port. The complete connection is as follows:



Open the downloaded folder "Crowtail-Advanced kit for Arduino demo code", navigate to the folder lib-> SFE_BMP180, and add SFE_BMP180 to the Arduino library. Open the P14_Get_Atmospheric_Pressure with Arduino IDE and upload it.

What will you see

After uploading the program, open the serial monitor you will see the monitor is printing the altitude, temperature, absolute pressure, relative pressure and calculated altitude data.

Code overview

1. Import the BMP180 Barometer and I2C library.
2. Create a BMP180 instance to obtain the pressure data and define a macro of altitude.
3. Initialize the serial monitor and set the baud rate for it and initialize the BMP180 barometer.
4. Print the altitude and get the temperature and then print.
5. Get the absolute pressure and relative pressure and then print.

Code usage

BMP180 Barometer library: #include <SFE_BMP180.h>

Import the BMP180 library. SFE_BMP180.h is a library of BMP180 barometer, with using the function of this library, we can easy to obtain the pressure data.

Create BMP180 instance object: SFE_BMP180 pressure;

As I often mentioned before, after importing the library, if we want to use the library's functions, we need to create an instance for us so that we can use the module functions more easily.

Macro definition: #define ALTITUDE 1655.0.

The prototype of the macro definition constant is #define [MacroName] [MacroValue]. We will use the variable of altitude to get the relative pressure data. You need to change your own altitude.

Get temperature data: status = pressure.getTemperature(T);

“pressure.getTemperature” is the function of getting BMP180 temperature data. The temperature data is stored in the variable T. If successfully gets pressure data, the function returns 1, otherwise, it returns 0.

Get absolute pressure data: status = pressure.getPressure(P,T);

“pressure.getPressure()” is the function of getting BMP180 absolute pressure data. Similar to pressure.getTemperature function, the pressure data is stored in the variable P and function returns 1 if successful and 0 if failure. What you need to note is that this function requires the previous temperature data(T).

Get relative pressure data: p0 = pressure.sealevel(P,ALTITUDE);

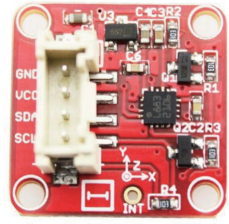
“pressure.sealevel” is the function of getting BMP180 relative pressure data. Parameter P is absolute pressure in mb. Parameter ALTITUDE is current altitude in m. The result p0 is sea-level compensated pressure in mb.

Lesson 15 – Digital compass

Introduction

Crowtail-3-Axis Compass module, a member of Crowtail family uses I²C based Honeywell HMC5883L digital compass. This ASIC is equipped with high-resolution HMC118X magneto-resistive sensors and a 12-bit ADC. It can provide an accurate compass heading. Signal conditioning like amplification, automatic degaussing strap drivers and offset cancellation are inbuilt. This Crowtail module also includes an XC6206P332MR for power supply requirement. Hence user can connect any 3.3V to 6V DC power supply.

Do you know what electronic module the ship's heading guidance uses? That's right, this is the Crowtail-3-Axis Compass module that you need to learn in this lesson. Let's start learning how to use the Crowtail-3-Axis Compass module to make a compass.



Required Parts

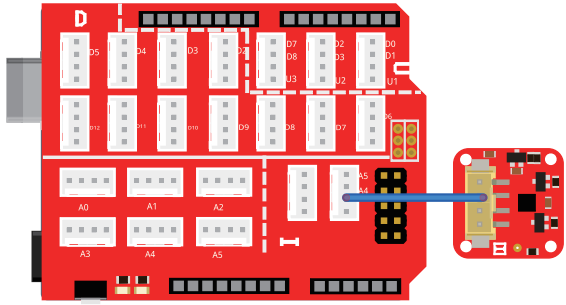
- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – 3-Axis Compass x1
- Crowtail – Cable x1
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-3-Axis Compass to Crowtail-Base shield's I port. The complete connection is as follows:

Open the downloaded folder "Crowtail-Advanced kit for Arduino demo code", navigate to the folder lib-> 3Axis_Digital_-Compass_HMC5883L, and add Digital Compass to the Arduino library. Open the P15_Digital_Compass with Arduino IDE and upload it.



What will you see

Upload the program and open the serial monitor, you can see the monitor is printing the raw values of 3-axis, the scaled values of 3-axis and the information of compass's radians and degrees.

Code overview

1. Import the I2C and HMC5883L compass library.
2. Create a compass instance and a variable to store errors that occur.
3. In the setup() function, initialize serial, I2C and check errors for compass.
4. Retrieve the raw values from the compass (not scaled)
5. Retrieved the scaled values from the compass (scaled to the configured scale).

6. Get the radians and change it into degrees, need to make sure the radians between 0-2*PI and degrees between 0-360 degrees.
7. Use Output() function to print all the data

Code usage

HMC5883L Compass Library: #include <HMC5883L.h>

Import the library of the compass. The library has many build-in functions to get raw, scaled and degree values.

Raw and scaled: MagnetometerRaw raw = compass.readRawAxis(); MagnetometerScaled scaled = compass.readScaledAxis();

“compass.readRawAxis()” is the function retrieve the raw values from the compass(not scaled). “compass.readScaledAxis” is the function retrieve the scaled values from the compass(scaled to the configured scale).

Add your “Declination Angle”: float declinationAngle = -0.0457;

Once you get your heading, you need to add your “Declination Angle”, which is the “Error” of the magnetic field in your location. You can find yours here: “http://www.magnetic-declination.com/”. Mine is -2 37’ which is -2.617 degrees and I need -0.0456752665 radians and I will use -0.0457.

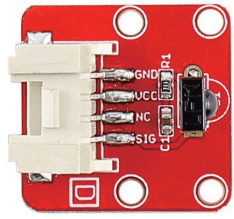
Function: Output()

Create a function called output() to print all the information of compass, there are four parameters we need to pass to this function, including raw, scaled, radians and degrees.

Lesson 16 – IR control system

Introduction

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. The IR detector has a demodulator inside that looks for modulated IR at 38 kHz. The Infrared Receiver can receive signals well within 10 meters. If more than 10 meters, the receiver may not get the signals.This lesson, we will use the IR Receiver, Infrared Remote Control and two color LEDs to make a wireless IR control project, which allows us to turn these two LEDs on or off.



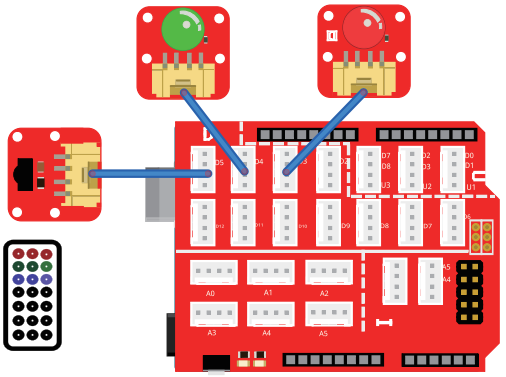
Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – IR Receiver x1
- Crowtail – Infrared Remote Control x1
- Crowtail – LED(Green) x1
- Crowtail – LED(Red) x1
- Crowtail – Cable x3
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-LED(Red), Crotail-LED(Green) and Crowtail-IR Receiver to Crowtail-Base shield's D3, D4 and D5 ports.The complete connection is as follows:



Open the downloaded folder “Crowtail-Advanced kit for Arduino demo code”, navigate to the folder lib-> IRremote, and add IRremote to the Arduino library. Open the P16_IR_Control_System with Arduino IDE and upload it.

What will you see

After the program upload is completed, when you press the "1" button on the remote control, the red LED lights on. When you press the "2" button on the remote control, the green LED light on. When the "3" button on the remote control is pressed, these two LEDs will be turned off at the same time.

Code overview

1. Import the IR remote library and declare some variable of the remote control button.
2. Declare the pin of two color LEDs and IR Receiver and create an instance of IR Receiver.
3. Initialize serial monitor and IR receiver and then declare modules are output or input.
4. Determine which button is pressed .if the pressed button is the last button pressed or the newly pressed button.
5. Use switch() statement to do the different things when different buttons are pressed.
6. Receive the next value.

Code usage

Constant: const uint16_t BUTTON_0 = 0x6897;

'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. “uint16_t” is a char unsigned character, the constant declared with it is a 16-bit character. "0x6897" is the encoding for hex button 0.

Start receiver: irrecv.enableIRIn();

Before we use the remote control to control the project, we need to make the IR Receiver start receiver so that IR Receiver can get the data we send from the remote control.

AND operation: uint16_t resultCode = (results.value & 0xFFFF);

“&” symbol means AND operation, just like logic AND modules we learned in Starter kit for Arduino, When both of the result is 1 when both numbers are 1, otherwise 0. The computer will get values by converting hexadecimal numbers to binary numbers. For example, 0xF in hexadecimal will be changed into 0x1111 in a binary number, then we operate 0x1111 and results.value(also need to convert hexadecimal number to binary number) to get the result. So we use results.value and 0xFFFF operation. When result.value is 0xFFFF (press the original button), resultCode is 0xFFFF. When result.value is not 0xFFFF (press the new button), resultCode is results. value.

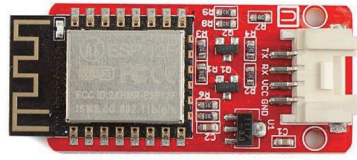
Switch case statement: switch(variable){case constantExpression1:statement1;break; case constantExpression2:statement2:break; default:statement2;break;}

Switch statement is similar to if/else statement, it is a judgment selection code. Its function is to control the flow of processes. When the quantity expressed by the variable expression matches the constant in one of the case statements, the statements following the case statement are executed, and the statements in all subsequent case statements are executed in turn, unless a break; statement is found out of the switch statement... If the amount of the constant expression does not match the constants of all case statements, the statements in the default statement are executed.

Lesson 17 – ESP8266 TCP server

Introduction

The serial wifi module based on ESP-12. which is an ultra-low-power UART-WiFi module. It has excellent dimensions and ULP technology compared to other similar modules. The module is a special design for mobile devices and the Internet of Things. Once firmware is upgraded to the appropriate version, a compatible Android device can run the IOT. APK to do the following: control the PWM, I / O pin, or Serial communication. For example, you can use this module transmit date with its serial port. It is easy to communicate with other devices
This lesson, we will use Serial wifi module to make a TCP server to print the information on the website.



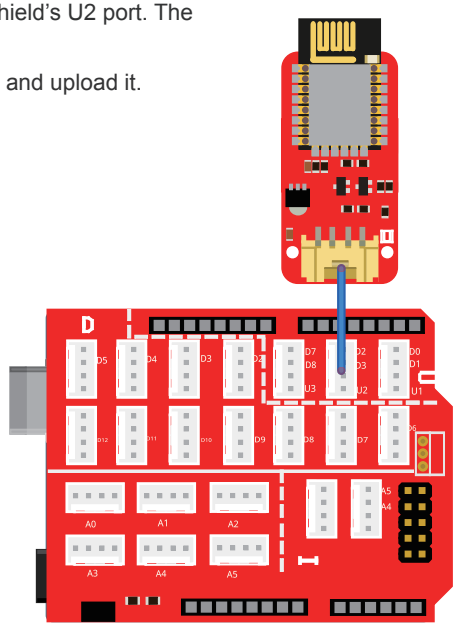
Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – Serial Wifi x1
- Crowtail – Cable x1
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-Serial Wifi to Crowtail-Base shield's U2 port. The complete connection is as follows:



Open the P17_ESP8266_TCP_Server with Arduino IDE and upload it.

What will you see

Upload the code and open the serial monitor, you can see some configuration information. Then, use your pc to connect the wifi of Serial Wifi. Once you connect the Serial Wifi module, open your browser and type the IP address you just saw from the serial monitor to visit the web server of the Serial Wifi.

Code overview

1. Macro defines a variable DEBUG to true.
2. Send the data to ESP8266 to configure access points and other connection information.
3. Check if the ESP is available, if yes, send the connection id and webpage information.
4. Create a function to send data.

Code usage

Macro definition: #define DEBUG true.

The prototype of the macro definition constant is #define [MacroName] [MacroValue]. What is the difference between a macro definition constant and a variable? Macro-defined constants cannot be changed while the program is running. We will use this macro definition constant in the function of sendData().

Send data: String sendData(String command, const int timeout, boolean debug){code to run; return response;}

This function is to send the data to ESP8266. "command" is the data/command to send. "timeout" is

the time to wait for a response. “debug” is a selection parameter, choose whether to print on serial monitor(true=yes, false=no). “return response” is to get the response from the esp8266. “boolean” is a basic type of data, it only returns true or false when print.

Html string: String webpage = "<h1>Hello World!</h1>";

This is slightly different from the normal string, that is, there are "<>" signs on both sides of the string. This is actually a string representation of a web page, h1 represents the few lines of the string on the web page.

Lesson 18 – Weather reminder

Introduction

Have you ever been in a hurry without paying attention to the outside temperature and whether it rained? I once had such an experience, because I was in a hurry and did not notice that the temperature outside was so cold, and then I was shivered by the coldness.
In this lesson, we will solve this problem for you! We will use Crowtail-Temperature & Humidity sensor, Crowtail-Water sensor, Crowtail-I2C LCD and Crowtail-LED as a weather reminder so that you can also know the weather when you are in a hurry.

Required Parts

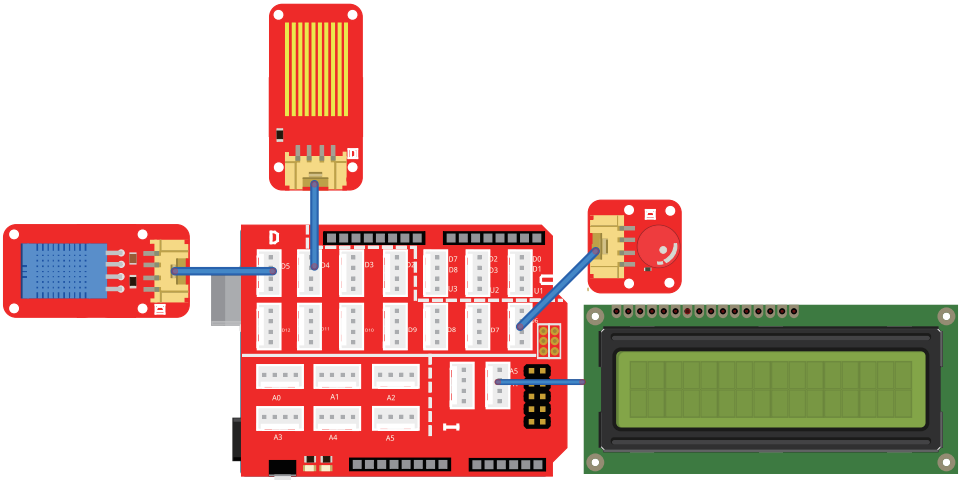
- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – Temperature&Humidity Sensor x1
- Crowtail – Water Sensor x1
- Crowtail – I2C LCD x1
- Crowtail – LED x1
- Crowtail – Cable x4
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-Water Sensor, Crowtail-Temperature&Humidity Sensor and Crowtail-LED to Crowtail-Base shield’s D4, D5 and D6 port. Connect Crowtail-I2C LCD to Crowtail-Base shield’s I port. The complete connection is as follows:

Open the P18_Weather_Reminder with Arduino IDE and upload it.



What will you see

You can see the LCD will display the information of temperature and humidity. And if there is a rain detected by water sensor, the LCD will display the "Bring umbrella" prompt. In addition, the red led will light up to make it more obvious that it is raining. You need to bring an umbrella. When the water sensor does not detect water, the LCD no longer displays the "Bring umbrella" prompt, and the red led will go out.

Code overview

1. Import I2C, LCD and DHT library.
2. Declare the pin and type of Temperature&Humidity sensor and initialization.
3. Declare I2C address of LCD and declare the pin of water sensor and LED.
4. Initialize the DHT and LCD. Declare modules are output or input.
5. Read the humidity, temperature and water state information.
6. If water is detected, show the temperature and humidity information and water remind on LCD.
7. If water is not detected, it only shows the temperature and humidity information.

Code usage

Import library: #include <Wire.h> #include "LiquidCrystal.h" #include "DHT.h"

Import the temperature and humidity sensor, LCD and I2C library. DHT.h is a library based on temperature and humidity sensors. LiquidCrystal.h is a library of LCD. Wire.h is a library of I2C modules.

Create DHT and LCD instance: DHT dht(DHTPIN, DHTTYPE); LiquidCrystal lcd(0);

The prototype for creating a DHT instance object is DHT name (uin8_t pin, uin8_t type, uin8_t count).

The “pin” represents the pin of the DHT sensor connected to the Arduino. The “type” represents the type of the DHT sensor. Our Crowtail–Temperature & Humidity Sensor uses DHT11. The “count” is an optional parameter. “LiquidCrystal lcd(0)” is to create an instance of LCD, which is connected to 0(I2C address).

Initialize : Serial.begin(9600); dht.begin(); lcd.begin(16, 2);

Before to use serial monitor, dht and lcd, we need to initialize them. Here, we initialize the baud rate of serial monitor is 9600 and we set up the LCD’s number of rows and columns.

Isnan(): if (isnan(t) || isnan(h)) {}

The role of is isnan() to determine whether the number in the brackets is a number. Isnan() is a short of “is not a number”, if the number in the parentheses is not a number, it returns TRUE, otherwise it returns FALSE.

Lesson 19 – Remote control system

Introduction

Have you ever been troubled by the need to turn off the fans yourself? Have you ever been troubled by the need to open the door yourself? Have you ever bothered to get up at night and turn on the light in the toilet? You must have thought that if you could use a simple remote control to control all of them!
Let's get started, we will use IR receiver, Infrared remote control, DC motor, Servo, MOSFET, LED to make a remote control system.

Required Parts

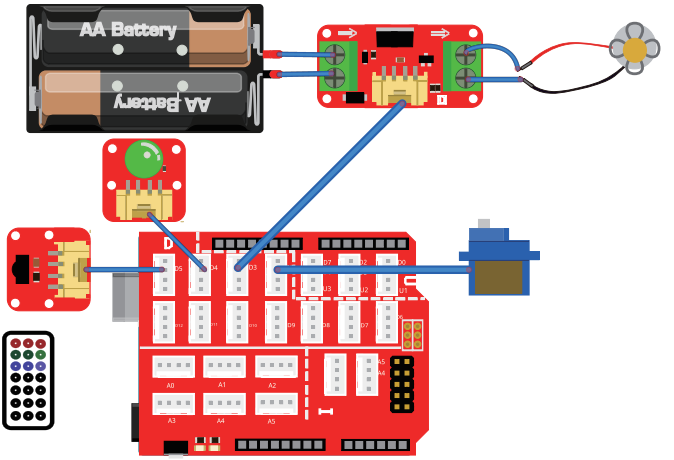
- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – IR Receiver x1
- Crowtail – MOSFET x1
- Crowtail – LED(Green) x1
- Crowtail – 9G Servo x1
- Crowtail – Cable x3
- Infrared Remote Control x1
- Battery Case x1
- DC Motor x1
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-9G Servo, Crowtail-MOSFET, Crowtail-LED and Crowtail-IR Receiver to Crowtail-Base shield’s D2, D3, D4 and D5 port. The complete connection is as follows:

Open the P19_Remote_Control_System with Arduino IDE and upload it.



What will you see

Upload the program, when you press the “0” button on Infrared remote control, you can see the servo is turning to 180 degrees (open door) and then rotate back to 0 degrees (close door). When you press the “1” button, you can see the motor will be turned on. When you press the “2” button, you can see the LED will be turned on. When you press the “3” or “4” button, the motor or LED will be turned off separately. If you turn on both the motor and LED, you can press the “5” button to close all the modules.

Code overview

1. Import servo and IR remote library.
2. Declare some constants, that is, the hexadecimal representation of the remote control buttons.
3. Create servo and IR Receiver instance object and declare the pin of modules.
4. Initialize the servo and IR receiver. Declare modules are output or input.
5. Determine which button is pressed .if the pressed button is the last button pressed or the newly pressed button.
6. Use switch() statement to do the different things when different buttons are pressed.
7. Receive the next value.

Code usage

Decode results: decode_results results;

Decoding result received by IR Receiver. The decoded result is placed in the instance object “results” constructed by “decode_results”.

AND operation: uint16_t resultCode = (results.value & 0xFFFF);

“&” symbol means AND operation, just like logic AND modules we learned in Starter kit for Arduino, When both of the results is 1 when both numbers are 1, otherwise 0. The computer will get values by converting hexadecimal numbers to binary numbers. For example, 0xF in hexadecimal will be changed into 0x1111 in

binary number, then we operate 0x1111 and results.value(also need to convert hexadecimal number to binary number) to get the result. So we use results.value and 0xFFFF operation. When result.value is 0xFFFF (press the original button), resultCode is 0xFFFF. When result.value is not 0xFFFF (press the new button), resultCode is results. value.

Switch case statement: switch(variable){case constantExpression1:statement1;break; case constantExpression2:statement2:break; default:statement2;break;}

Switch statement is similar to if/else statement, it is a judgment selection code. Its function is to control the flow of processes. When the quantity expressed by the variable expression matches the constant in one of the case statements, the statements following the case statement are executed, and the statements in all subsequent case statements are executed in turn, unless a break; statement is found out of the switch statement. . If the amount of the constant expression does not match the constants of all case statements, the statements in the default statement are executed.

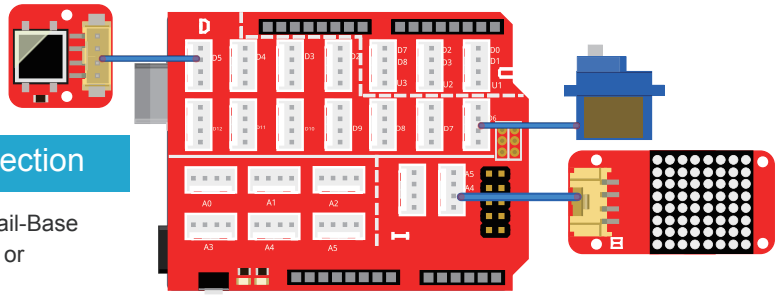
Lesson 20 – Polite automatic door

Introduction

Have you ever been doing your own thing and suddenly been asked by your parents to open the door? This is definitely a very annoying thing, and I am determined to change it! So in this course, we will use Crowtail-LED Matrix, Crowtail-9G Servo and Crowtail-Button to make a simulated automatic door opening for the door of our house. Of course, this door will be very polite to visitors.

Required Parts

- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – LED Matrix x1
- Crowtail – Button x1
- Crowtail – 9G Servo 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 and Crowtail-9G Servo to Crowtail-Base shield's D5 and D6 port. Connect Crowtail-LED Matrix to Crowtail-Base shield's I port. The complete connection is as follows:

Open the P20_Polite_Automatic_Door with Arduino IDE and upload it.

What will you see

Upload the program, when you press the button, the servo will rotate to 180 degrees(open door) and LED Matrix will show "Welcome" prompt message. When you release the button, the servo will rotate back to 0 degrees(close door). LED Matrix no longer prompts.

Code overview

1. Import the I2C, LED Matrix and servo library.
2. Create instances of LED Matrix and servo.
3. Declare variables of button's pin and state.
4. Initialize the LED Matrix and servo, pass the I2C address to LED Matrix and declare the pin of servo.
5. Read the state of the button.
6. If button is pressed, LED Matrix displays the welcome message and servo rotate to 180 degrees.
7. If button is not pressed, LED Matrix will display nothing and servo rotate to 0 degrees.

Code usage

Import library: #include <Wire.h> #include "Adafruit_LEDBackpack.h" #include <Servo.h>

This would be always the first thing to do when you need to use the external function. Crowtail-LED Matrix is an I2C module, so we need to import I2C library(Wire.h) first. Then we import LED Matrix's library "Adafruit_LEDBackpack.h" and servo's library "Servo.h".

Create instance: Adafruit_8x8matrix matrix; Servo myservo;

Create instances of LED Matrix and servo, so we can use the instance object to control the LED Matrix and servo.

Text size: matrix.setTextSize(1);

We use the matrix.setTextSize () function to set the text size displayed on the LED matrix. 1 is a good choice for 8x8 LED matrix display. You can try changing 1 to 2 and see what the LED matrix will look like.,.

Wrap of scroll: matrix.setTextWrap(false);

Using matrix.setTextWrap() function we can set whether the text display on LED Matrix is wrapped or scroll. If the parameters in the brackets are true, the display text effect is wrapped, if it is false, the display text effect is scroll.

LED Matrix ON or OFF: matrix.setTextColor(LED_ON);

Set the whether the LED is on or off, when you choose “LED_ON”, you can see the effect of led light, if you choose “LED_OFF”, you will not see any effect.

Servo rotation: myservo.write(180);

The function of servo rotation. We use instanceName.write() function to control the servo rotate. The parameters in the brackets of this function are the specific angles to which the servo is rotated. For example, here we enter the parameter in the brackets as 180 to rotate the servo to 180 degrees.

Lesson 21 – Weather station

Introduction

Remember the weather reminder we made above? Is it still not enough for you to master the weather? Well, let's make a more detailed weather station and "tell" you all the information about the weather that can be measured!
We will use Crowtail- Temperature & Humidiy Sensor, Crowtail- Water Sensor, Crowtail- BMP180 Barometer, Crowtail- Luminance Sensor, Crowtail- I2C LCD to make a rich weather information that can provide you with temperature, humidity, rain, atmospheric pressure and brightness.

Required Parts

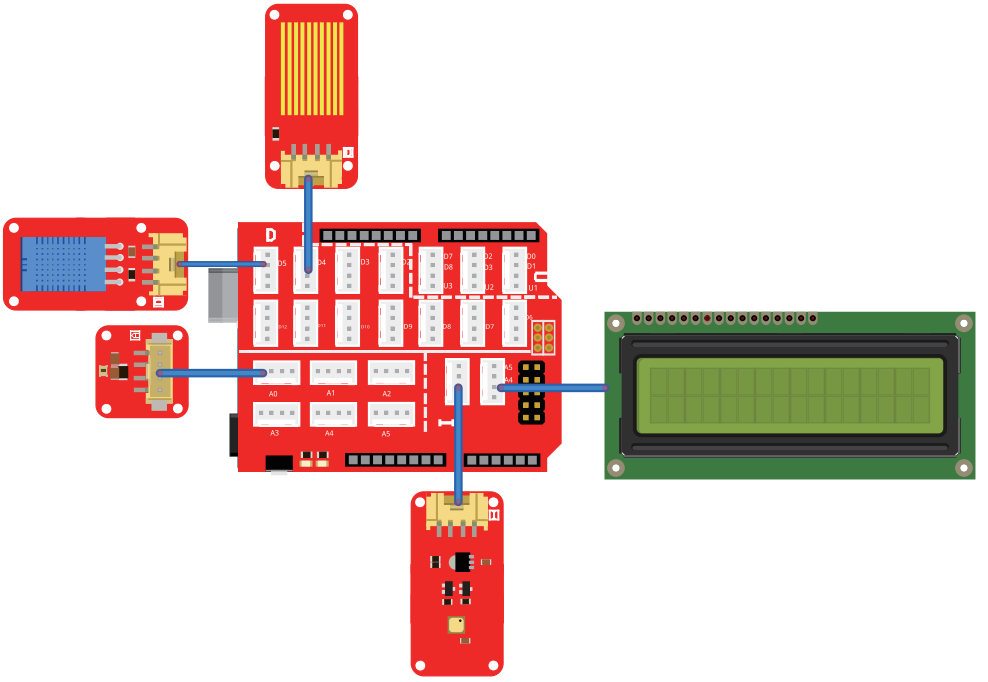
- Crowduino UNO-SD/Arduino UNO x1
- Crowtail – Base Shield x1
- Crowtail – Temperature&Humidity Sensor x1
- Crowtail – Water Sensor x1
- Crowtail – BMP180 Barometer x1
- Crowtail – Luminance Sensor x1
- Crowtail – I2C LCD x1
- Crowtail – Cable x5
- USB Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-Water Sensor and Crowtail-Temperature&Humidity Sensor to Crowtail-Base shield’s D4 and D5 port. Connect Crowtail-Luminance sensor to Crowtail-Base shield’s A0 port. Connect Crowtail-BMP180 Barometer and Crowtail-I2C LCD to Crowtail-Base shield’s I port. The complete connection is as follows:

Open the P21_Weather_Station with Arduino IDE and upload it.



What will you see

Upload the program and you will see that the LCD will first display the brightness information of the environment and whether it is raining, and then the LCD switches the screen to display the humidity and temperature information of the current environment. Finally, the LCD will switch screens again to display the current absolute and relative pressure information.

Code overview

1. Import BMP180, I2C, LCD and DHT library.
2. Create BMP180, DHT11 and LCD instances.
3. Declare the pin of modules, some variable and arrays.
4. Initialize the DHT, LCD and determine whether the modules is output or input.
5. Create the function to read luminance data and pressure data.
6. Read humidity, temperature and water information from sensor.
7. Print the luminance and whether it rains on LCD.
8. Print the humidity and temperature sensor on LCD.
9. Print the environment's pressure information on LCD.

Code usage

Import library: #include <SFE_BMP180.h> #include <Wire.h> #include "LiquidCrystal.h" #include "DHT.h"imp

As I had said before, it would be always first to import the library, so we can very convenient to use the function which is created by others. “SFE_BMP180.h” is the library file of BMP180 module. “Wire.h” is the library of I2C modules which is including BMP180 and I2C LCD. “LiquidCrystal.h” is the library of LCD and “DHT.h” is the library of DHT sensor which include dht11 sensor, dht22 sensor and other type of DHT sensors.

Create instances: SFE_BMP180 pressure; DHT dht(DHTPIN, DHTTYPE); LiquidCrystal lcd(0);

Create some instances to control the BMP180, DHT and lcd. We create a BMP180 instance object calls “pressure”. A DHT instance object calls “dht” which includes its pin and type. An LCD instance object called lcd and passes the I2C address to the object.

Initialize: Serial.begin(9600); dht.begin(); lcd.begin(16, 2); pressure.begin()

Initialize all the modules that we are going to use. We initialize the serial monitor and set its baud rate is 9600, initialize the DHT, BMP180 and LCD which is set up 16 row and 2 columns.

Functions: float readLuminance(uint8_t analogpin){} float FmultiMap(float val, float * _in, float * _out, uint8_t size){} void bmp180(){}

Create the functions to read luminance information and pressure information. When the code is relatively long, we will try to modularize the code that implements the same function. Modularity means that it is achieved by creating functions. This way the code is not easy to cause confusion, easy to manage and modify. Here we create readLuminance() function to read the luminance information. FmultiMap() is a function to calculate luminance value and we create a function call bmp180() which is to read pressure information.

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

In the loop () function, we will read the values of temperature, humidity, brightness, and air pressure through the function created by ourselves or the function of the imported library file, and then display them separately through the LCD.

ELECROW

MAKE YOUR MAKING EASIER

info@elecrow.com

www.elecrow.com

+86 0755-23204330

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