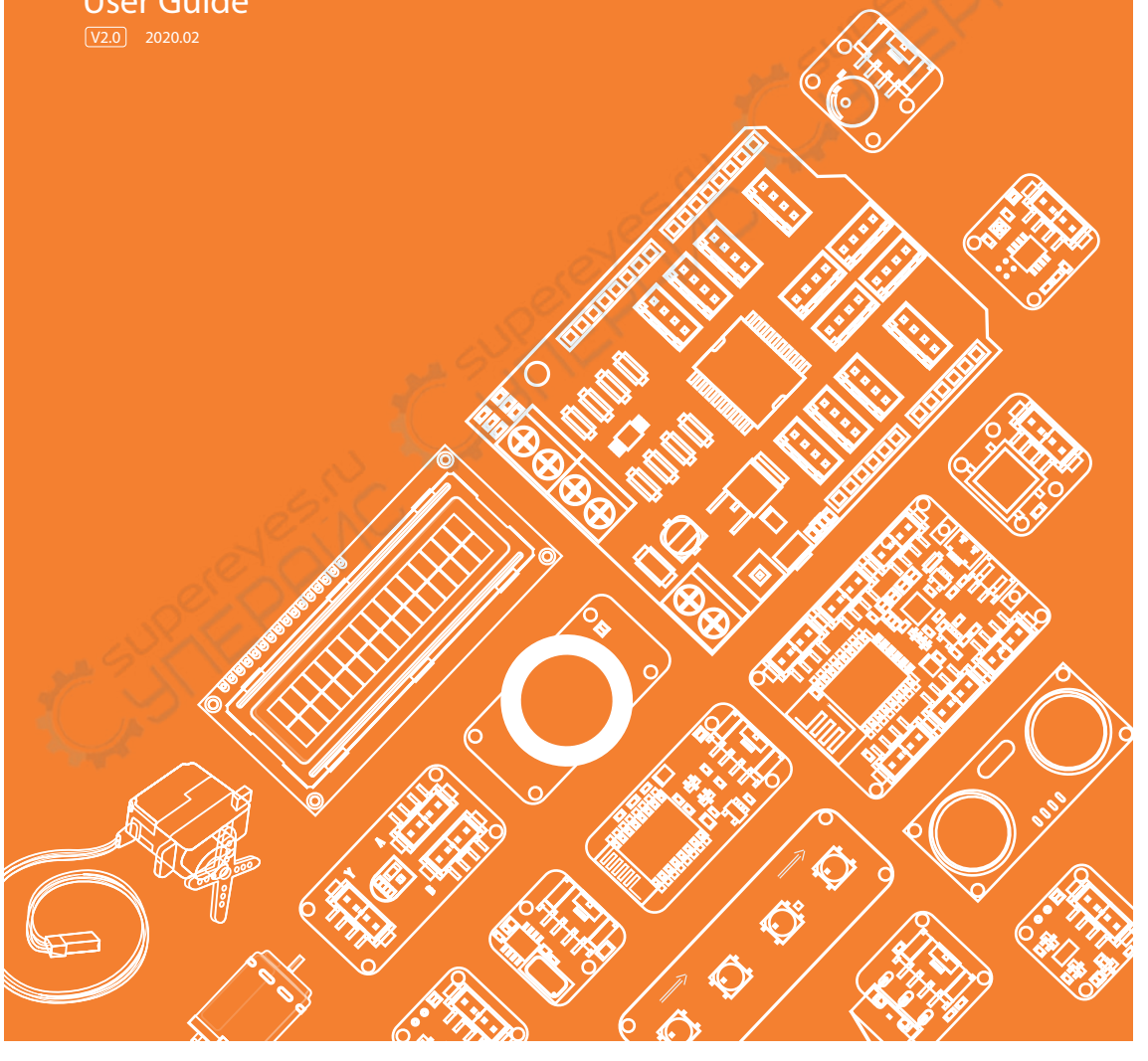
Crowtail Deluxe Kit for Arduino

User Guide

V2.0 2020.02



Content

Instruction

Welcome to the user manual of the Crowtail-Deluxe kit for Arduino. The electronic world is so exciting and amazing! Probably you can't wait to create more interesting works or even realize your electronic dream. Just wait a minutes, let’s get to know this amazing kit first. The Crowtail-Deluxe Kit for Arduino is a kit for advanced learners with a basic programming foundation. It is designed to help learners diverge their ideas, learn how more complex electronic components work and program, and ultimately help them achieve their dream electronic project! The kit contains more than 20 creative and inspiring courses, from simple to difficult, to guide you in detail to learn how these complex electronic modules work and how to use them, and then program to achieve challenging and creative electronic projects that will even teach you how to create your own artificial intelligence work! In all, by learning this kit, you will learn how to build a smart car completely, including wireless remote control, intelligent tracking or obstacle avoidance, etc. Of course, there are more complex movement data acquisition, wifi and Bluetooth use!

The Crowtail-Deluxe kit for Arduino includes more than 20 electronic modules, which are selected one by one from more than 100 Crowtail electronic modules. The kit is designed to let you learn some complex electronic modules and learn how they work and how to use them. In addition, the kit will broaden your thinking, help you inspire more novel ideas and creations, and ultimately help you achieve your dream work!

For the programming part, we will use Arduino software for programming. As you know, 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 Arduino board models) and software (Arduino IDE). This is the easiest and best choice for you to learn programming and hardware and achieve creative work. In short, you will explore the core of electronic device programming and creation, and help you create your own great work with a more comprehensive mind.

Modules List

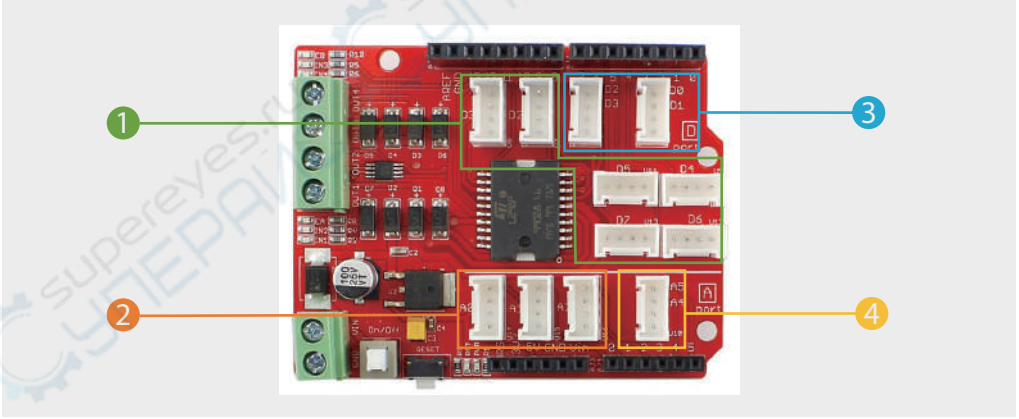
- Crowtail - Motor Base shield x1
- Crowtail - Buzzer x1
- Crowtail - Switch x1
- Crowtail - LED(Red) x1
- Crowtail - Bright LED x1
- Crowtail - RGB-LED x1
- Crowtail - OLED x1
- Crowtail - IR Reflective Sensor x3
- Crowtail - Thumb Joystick x1
- Crowtail - IR Receiver x1
- Crowtail - IR Emitter x1
- Crowtail - Hall Sensor x1
- Crowtail - One Wire Waterproof Temperature Sensor x1
- Crowtail - Collision Sensor x1
- Crowtail- Ultrasonic Ranging Sensor x1
- Crowtail- 3-Axis Digital Accelerometer(±16g) x1
- Crowtail - 9G Servo x1
- Crowtail - ESP8266 nodeMCU x1
- Crowtail - Analog Gyro x1
- Crowtail - I2C EEPROM x1
- Crowtail - Bluetooth Low Energy Module x1
- Micro-Speed Motor x2
- Small magnet 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 when connecting electronic circuits. It consists of a Base Shield and some basic Crowtail modules, which help you create small, simple, and easy-to-assemble circuits. In other words, when you use Crowtail, your electronic project will not be messy wiring, instead, it will be a simple and easy to manage an electronic project!

Crowtail – Base Shield

For those who want to use Arduino to build their own DIY mobile platform, motor drive has always been a problem. It need 2 DC motors at least to control their speed& direction of rotation because you will want your platform forward, turn back, turn left or right. Besides, motors always need large current, so, you will need some modules to help you control the large current, with a microcontroller such as Arduino. Then, the Crowtail-Motor Base shield appears, it can meet your all demand to DIY your mobile platform! The Crowtail-Motor Base Shield is a standard IO expansion motor driver board for the Arduino. It regulate the IOs of Arduino to the standard Crowtail interface and have two interfaces to drive motors or a interface to drive 4-wire stepper, which can be sorted into 4 kinds: Analog(A), Digital(D), UART(D0&D1, D2&D3) and IIC(A4&A5):



1 6 Digital I/O ports (D2~D7) that have a mark “D”. These ports can be used to read and control digital Crowtail modules (Crowtail modules that have a mark “D”), such as the Button and LEDs. Some of the digital I/O ports can also be used as PWM (pulse width modulation) outputs;

2 3 Analog ports (A0~A2) that have a mark of “A”. Besides the functional of digital, these A ports can read the analog signal, such as a potentiometer or light sensor;

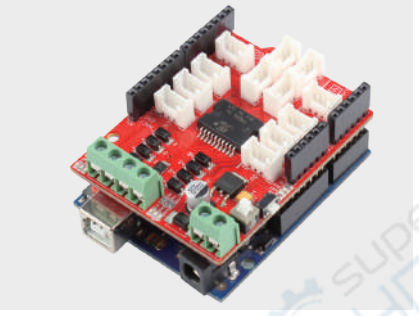
3 2 UART ports(D0&D1, D2&D3) that have a mark of “U”. These interfaces can be used for UART communication such as the WIFI module or Bluetooth module;

4 1 IIC ports(A4&A5) that have a mark of “I”. These interfaces are for the IIC Communication;

Besides, there is also two motor connector, which you can drive 2 motors simultaneously or drive a 4-wire stepper. Users can connect any electronic modules to the Base Shield with Crowtail cable 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- Motor Base shield with your Arduino



Connect Crowtail- Motor Base shield with your Crowduino

Crowtail – Modules

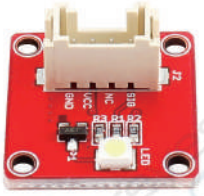
We make more than 100 kinds of electronic modules into Crowtail modules. They include a variety of sensors, displays, inputs and outputs modules, communication types include I2C, UART, digital or analog, which aim to provide more options to fully meet all needs for your electronic projects! All modules can be used by simply connecting them to the Crowtail-Motor Base shield using a Crowtail cable, which is a huge improvement over the previously troublesome jumper connections.

Lessons

Lesson 1 – LED Control

Introduction

Crowtail bright LED is a bright LED light with high brightness and a large angle. Then its low terminal resistance can be applied to indoor lighting, commercial lighting. Unlike colored LEDs, Crowtail bright led light is white, very similar to our daily light source and you can use it to make your own table lamp. It can provide a strong light source for your project, lighting, or so on.



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 a 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.

In this lesson, we are going to use bright LED module and switch to make a corridor night lights, which will produce a strong illumination light for those who need it. Let's start playing and see the difference form colored LEDs.

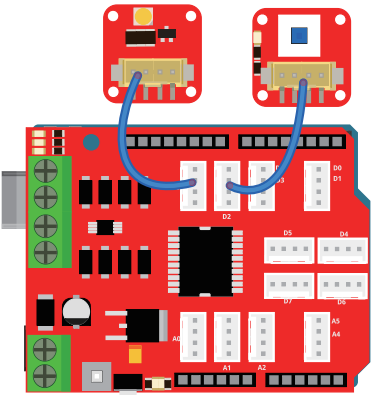
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - Bright LED x1
- Crowtail - Switch x1
- Crowtail - Cable x2

Hardware Connection

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

STEP2: Connect the Crowtail-Switch and Crowtail-Bright LED to the D2 and D3 ports of the Crowtail-Motor Base Shield. The complete connection is as follows.



Open the P01_LED_Control with Arduino IDE and upload it.

What will you see

When you press the switch, the bright LED will immediately light up, and the LED will go out when you press the switch again to make the switch button pop up.

Code overview

1. Declare the LED and switch pins, and the “switchstate” variable to store the state of the button
2. Setup the LED as output and switch as input.
3. Read the state of the switch.
4. If the switch is pressed(HIGH), turn the LED on.
5. If the switch is pressed again and switch button is pop up(LOW), turn the LED off.

Code usage

P01_LED_Control

```
/*
P01_LED_Control
Crowtail - bright led connect to D3 port
Crowtail - switch connect to D2 port
*/

int ledPin = 3;    //declare the variable of led pin
int switchPin = 2;  //declare the variable of switch pin
int switchState=0;  //declare the variable of switch state
// the setup routine runs once when you press reset:
void setup() {
  // initialize the led pin as an output.
  pinMode(ledPin, OUTPUT);
  // initialize the switch pin as an input.
  pinMode(switchPin, INPUT);
}
// the loop routine runs over and over again forever:
void loop() {
    switchState = digitalRead(switchPin);
    // check if the switch is pressed.
    // if it is, the switchState is HIGH:
  if (switchState == HIGH) {
      // turn LED on:
      digitalWrite(ledPin, HIGH);
    }
  else {
      // turn LED off:
      digitalWrite(ledPin, 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 ‘switchState’ of type int(integer) and assign it a value of 0 to record the status of the switch. 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 as an output and switch as an input.

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 switch is pressed, the LED will light on and when the switch is pressed a second time and the switch button is pop up, 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.

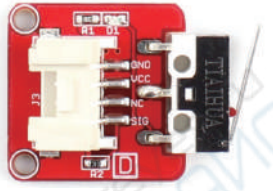
Digital Output

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 – Collision experiment

Introduction

This is a Robot model car crash or collision switch sensor module for Arduino, it normally outputs a logic HIGH signal, but when the sensor crash something such as the wall, the on-board switch will be pressed, and the module outputs a logic LOW signal. This module can be installed into any mobile platform to achieve collision detection function via 4 pin sensor cable and Arduino sensor expansion board connector. Do you know why the car opens the airbag immediately after the impact? In fact, the collision sensor is used to detect collisions in the car, so we will simulate a car detection project in this lesson, which will be absolutely cool if installed in your own smart car.



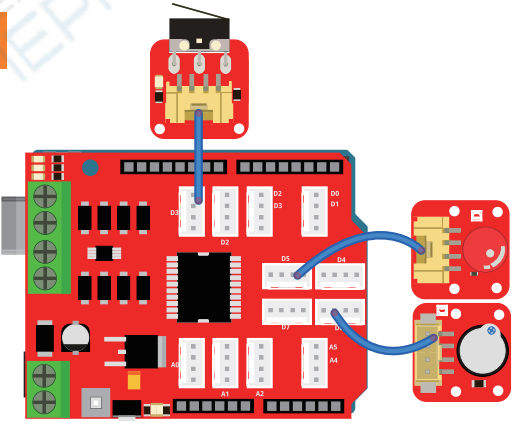
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - Collision Sensor x1
- Crowtail - LED x1
- Crowtail - Buzzer x1
- Crowtail - Cable x3

Hardware Connection

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

STEP2: Connect the Crowtail-Collision Sensor to the D3 port of the Crowtail-Motor Base Shield. Connect the Crowtail-LED and Crowtail-Buzzer to the D5 and D6 ports of the Crowtail-Motor Base Shield. The complete connection is as follows.



Open the P02_Collision_Experiment with Arduino IDE and upload it.

What will you see

When you press the onboard switch, you can see the red indicator light on the collision sensor, led and buzzer will work for 0.5 seconds, when you release the switch, the red indicator light on the board Off, led and buzzer no longer light up and beep.

Code overview

1. Macro definitions of PIR motion sensor pin, LED pin and buzzer pin.
2. Declare a variable to store the state of the collision sensor.
3. Setup led and buzzer as output.
4. Read the state of the collision sensor and store it to the variable collisionState.
5. If collision is detected(LOW), turn led and buzzer on for 0.5 seconds.
6. If a collision is not detected(HIGH), turn led and buzzer off.

Code usage

Macro definition: #define collisionPin 3 #define ledPin 5 #define buzzerPin 6

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.

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 the collision sensor, when the on-board switch of the collision sensor is pressed, the state of the collision sensor is LOW which is different from most modules and it will be HIGH when the on-board switch of collision sensor is not pressed.

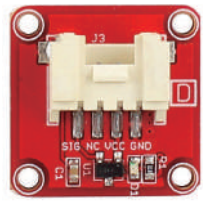
Delay: delay(500); delay(200);

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

Lesson 3 – Hall speed test

Introduction

The Crowtail-Hall Sensor uses the Allegro™ A1101 Hall-effect switches are next-generation replacements for the popular Allegro312x and 314x lines of unipolar switches. It measures the Hall Effect, which is a production of a voltage difference across an electrical conductor, transverse to an electric current in the conductor as well as a magnetic field perpendicular to the current.



The output of the continuous-time switch Hall sensor output logic low(turns on) when a magnetic field (south polarity) perpendicular to the Hall sensor exceeds the BOP threshold, and it output logic high(turn off) when the magnetic field disappears.
The Hall sensor is a very practical electronic module. It can be applied in many aspects, such as measuring displacement, measuring speed and so on. In this lesson, we will do a simple speed measurement experiment with a hall sensor, let's start turning.

Required Parts

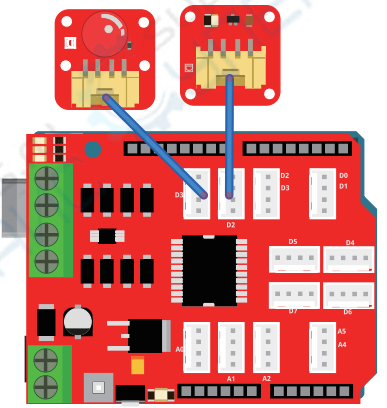
- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - Hall Sensor x1
- Crowtail - LED x1
- Crowtail - Cable x2

Hardware Connection

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

STEP2: Connect the Crowtail-Hall Sensor to the D2 port of the Crowtail-Motor Base Shield. Connect the Crowtail-LED to the D3 port of the Crowtail-Motor Base Shield. The complete connection is as follows.

Open the P03_Hall_Speed_Test with Arduino IDE and upload it.



What will you see

Open the serial port monitor. When you use the magnet's S polar to approach the hall sensor, the red indicator light on the hall sensor will light up and the speed count will increase by one. When you repeat the action within the 5 second count time, the count will follow the number of times the S-Class approaches the Hall sensor are superimposed and printed out after five seconds.If the count of speed more than 20, the LED will lights on, otherwise, the LED will lights off.

Code overview

1. Declare the hall sensor and led pins.
2. Define the time interval of each measurement, a variable to timing and a variable to count.
3. Initialize the serial monitor and set the baud rate.
4. Call interrupt function to count speed.
5. Start timing and get the current system time.
6. If the count of speed more than 20, print the count of speed every 5 seconds and turn led on.
7. If the count of speed less than 20, print the count of speed every 5 seconds.

Code usage

Constant: const int taketime = 5000;

'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. Here we declare the “taketime” constant and it assigned to 5000, which mean we will get the count of speed every 5 seconds.

Unsigned long integer: unsigned long time;

Each type of integer is divided into two types: unsigned and signed (float and double are always signed). In data types other than char, integers declared by default Type variables are all signed types; char is always unsigned by default. The long keyword represents a long integer type, which is a basic data type in a programming language. It is an abbreviation of long int. The default is a signed long integer type.

Interrupt function: attachInterrupt(digitalPinToInterrupt(interruptPin), count, FALLING);

The prototype of the interrupt function is: attachInterrupt(interrupt，function，mode). “Interrupt”: Interrupt source (In Arduino, the selectable value of the interrupt source is 0 or 1, generally corresponding to pins 2 and 3 respectively). “function”: the name of the function that needs to be interrupted. “mode”: LOW (low level trigger), CHANGE (trigger when change), RISING (low-level to high-level trigger), FALLING (high-level to low-level trigger). We use a Hall sensor as the interrupt source connected to the D3 port. When the interrupt occurs (the S level of the magnet is close to the Hall sensor), this means that the high level signal of the Hall sensor will become a low level signal, and then we run the code in the count function.

Millis() function: time = millis();

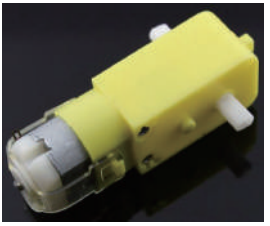
This function returns the number of milliseconds when the Arduino board starts running the current program. This number overflows after about 50 days and returns to zero. By using this function, we can precisely control the program to calculate the count of speed every 5 seconds and print it out in the serial monitor.

Lesson 4 – Motor control

Introduction

The 5V, 83rpm Micro DC Geared Motor with Back Shaft is ideal for DIY enthusiasts. This motor is inexpensive, small, easy to install, and ideally suited for use in a mobile robot car. Its no-load speed is 50rpm under 3V voltage operation and 83rpm under 5V voltage operation.

Our Crowtail-Motor Base Shield includes 2 motor interfaces, which can drive two motors at the same time! Of course, a 4-wire stepper motor can also be driven. After inserting the Crowtail-Motor Base Shield into Arduino/Crowduino, the specific method of controlling the motor is as follows:

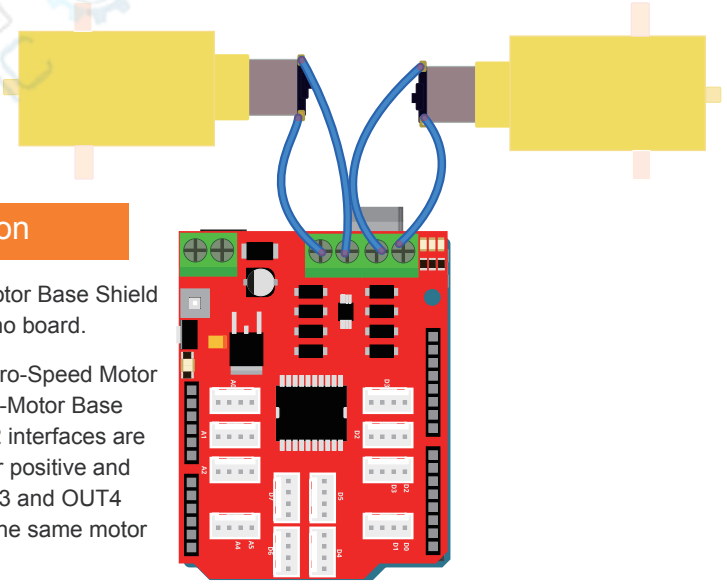


Motor	Pin Name	Arduino pin	Description
Motor_1	IN1	D8	D8=0,D11=1 -> clockwise;
	IN2	D11	D8=1,D11=0 -> anticlockwise;
	ENA	D9	Motor_1 speed control, duty can be 0%~100%
Motor_2	IN3	D12	D12=0,D13=1 -> clockwise;
	IN4	D13	D12=1,D13=0 -> anticlockwise;
	ENB	D10	Motor_2 speed control, duty can be 0%~100%

In this course, we will use the Crowtail-Motor Base Shield motor interface to control two TT motors simultaneously. This includes not only controlling the on or off of the motor, but also the speed of the motor, which means that you can implement controls such as turning. Add "legs" to your smart car!

Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Micro-Speed Motor x2
- Jumper wire x4



Hardware Connection

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

STEP2: Connect the two Micro-Speed Motor to motor interface of Crowtail-Motor Base Shield. The OUT1 and OUT2 interfaces are connected to the same motor positive and negative poles, and the OUT3 and OUT4 interfaces are connected to the same motor positive and negative poles.

Be careful not to connect the motor to the interval interface, such as OUT1 and OUT3, otherwise, the motor will not work properly. The complete connection is as follows.

Open the P04_Motor_Control with Arduino IDE and upload it.

Note: Because the motor consumes more power, after uploading the code, you need to provide additional power for the Crowtail-Motor Base Shield (usually connect 9V power supply for the power input port of the motor base shield) and then turn on the onboard power switch(you don't have to power Crowduino/Arduino, because the motor base shield will share the power with Crowduino/Arduino).

What will you see

Assume that motors 1 connected to OUT1 and OUT3 are mounted on the left tire of the car, and motors 2 connected to OUT3 and OUT4 are mounted on the right tire of the car. At the beginning, you will see that motor 1 is faster than motor 2 so that the car can turn left. Then, the two motors stopped rotating for two seconds. Finally, the motor1 motor is slower than the motor2 motor. This way the car can turn right.

Code overview

1. Declare the pins to control the speed and direction of rotation of the two motors.
2. Setup the pins which are used to control motors as output.
3. Create three functions to control the two motors to turn left, turn right and stop.
4. Call left, stop and right functions we just created to control motor rotation.

Code usage

Constant: const int pinI1=8; const int pinI2=11; const int speedpinA=9; const int pinI3=12; const int pinI4=13; const int speedpinB=10;

'const' is the abbreviation of constants. Here, we will create 6 constant which mean the pins we use to control the two motors. You can control the steering and speed of the motor through our motor control usage chart.

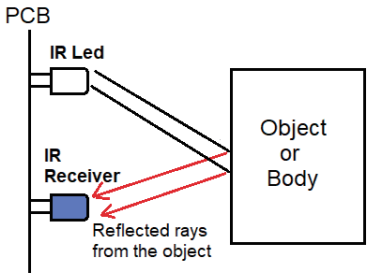
Modular programming: void Test_Load_Left(){} void Test_Load_Right(){} void clean_Output(){}

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. We call these three functions in the loop() function to control the motor to turn left, stop and turn right.

Lesson 5 – Tracing experiment

Introduction

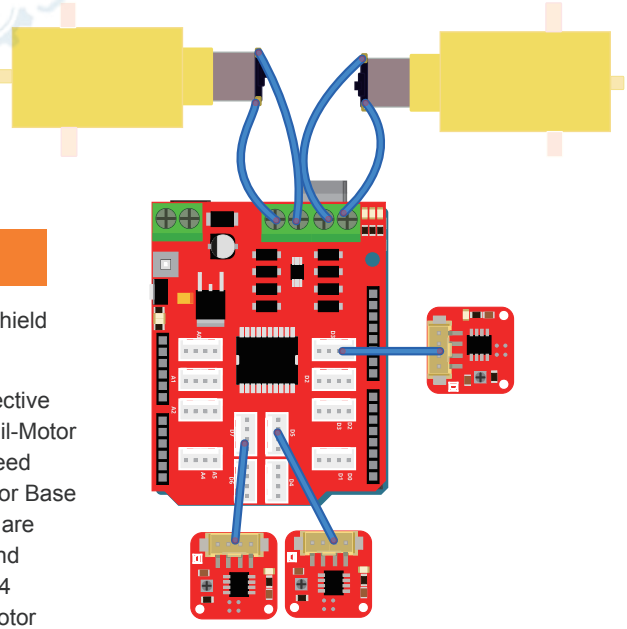
This IR reflective module emits the infrared light and then detects if the echo received, to estimates if there is an obstacle or not. It utilizes RPR-220 reflective photosensor modules, When no infrared light echo received, that is, there is no black obstacle in front of the sensor, this module output logic HIGH, and vice versa. There is also an on-board potentiometer to adjust the sensitivity. This sensor is a basic and widely used part in applications such as line-following cars, rotary speed detection, auto data logging on utility meters.



This IR reflective module emits the infrared light and then detects if the echo received, to estimates if there is an obstacle or not. It utilizes RPR-220 reflective photosensor modules, When no infrared light echo received, that is, there is no black obstacle in front of the sensor, this module output logic HIGH, and vice versa. There is also an on-board potentiometer to adjust the sensitivity. This sensor is a basic and widely used part in applications such as line-following cars, rotary speed detection, auto data logging on utility meters.

Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - IR Reflective Sensor x3
- Crowtail - Cable x3
- Micro-Speed Motor x2
- Jumper wire x4



Hardware Connection

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

STEP2: Connect three Crowtail-IR Reflective Sensor to D3, D5 and D7 port of Crowtail-Motor Base Shield. Connect the two Micro-Speed Motor to motor interface of Crowtail-Motor Base Shield. The OUT1 and OUT2 interfaces are connected to the same motor positive and negative poles, and the OUT3 and OUT4 interfaces are connected to the same motor

positive and negative poles, and the OUT3 and OUT4 interfaces are connected to the same motor positive and negative poles. Be careful not to connect the motor to the interval interface, such as OUT1 and OUT3, otherwise the motor will not work properly. The complete connection is as follows.

Open the P05_Tracing_Experiment with Arduino IDE and upload it.

Note: Because the motor consumes more power, after uploading the code, you need to provide additional power for the Crowtail-Motor Base Shield (usually connect 9V power supply for the power input port of the motor base shield) and then turn on the onboard power switch(you don't have to power Crowduino/Arduino, because the motor base shield will share the power with Crowduino/Arduino).

What will you see

Assume that motors 1 connected to OUT1 and OUT3 are mounted on the left tire of the car, and motors 2 connected to OUT3 and OUT4 are mounted on the right tire of the car. Place the IR reflective sensors connected to D3, D5, and D7 at the left, middle, and right positions of the bottom of the cart, respectively. When a black object is held only by the sensor of D3, motor2 rotates faster than motor1(turn left). When using a black object to block only the D5 sensor, motor1 rotates as fast as motor2 (forward). When using a black object to block only the sensor of D7, motor1 rotates faster than motor2 (turn right). When none of the three sensors are blocked by a black object, the speed of motor1 and motor2 becomes 0 (stopped).

Code overview

1. Declare the pins of three IR Reflective Sensors and variables to store the state of them.
2. Declare the pins to control the speed and direction of rotation of the two motors.
3. Setup the pins which are used to control motors as output.
4. Create functions to control the two motors to turn left, turn right, stop and forward.
5. Read the state of the three IR Reflective Sensors.
6. Use if statement to judge which function should be called to control motor.

Code usage

Constant and integer Variables: const int irPin1 = 3; const int irPin2 = 5; const int irPin3 = 7; int ir1State = 0; int ir2State = 0; int ir3State = 0;

'const' is the abbreviation of constants. The biggest difference between constants and variables is that constants cannot change values while the program is running, while variables can be changed. We use three constants to declare three IR reflective sensor pins because the pins of the three sensors do not need to be changed during the program running. Then we use three variables to declare the status of the three IR reflective sensors so that we can continuously read the status values of the IR reflective sensors and update them.

AND operation: if((ir1State == 0)&(ir2State != 0)&(ir3State != 0)){}

"&&" stands for "AND", that is, the AND operation is performed on the pre and post operands. If the before and after operands are true, the expression evaluates to true. In the formula, if the left operand is false, the right operand is no longer calculated.

Lesson 6 – Over temperature reminder

Introduction

This is a waterproof version of the DS18B20 Temperature sensor. Handy for when you need to measure something far away, or in wet conditions. While the sensor is good up to 125 degrees, the cable is jacketed in PVC so we suggest keeping it under 100 degrees. Because they are digital, you don't get any signal degradation even over long distances! The DS18B20 provides 9 to 12-bit (configurable) temperature readings over a 1-Wire interface so that only one wire (and ground) needs to be connected from a central microprocessor. Usable with 3.0-5.5V systems. Because each DS18B20 contains a unique silicon serial number, multiple DS18B20s can exist on the same 1-Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and process monitoring and control. Today, we will use this temperature sensor and LED as a temperature warning device. When the temperature exceeds a certain value, the LED lights up to indicate that the water temperature is too hot.

Note: Please pay attention to safety during use.

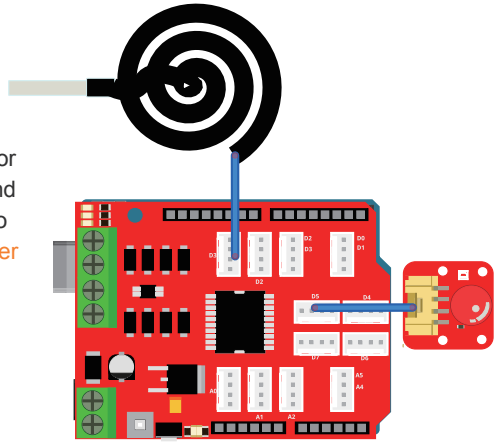
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - One Wire Waterproof Temperature Sensor x1
- Crowtail - LED
- Crowtail - Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-One Wire Waterproof Temperature Sensor and Crowtail-LED to D3 and D5 port of Crowtail-Motor Base Shield.The complete connection is as follows.



Open the downloaded folder “Crowtail Deluxe Kit for Arduino demo code”, navigate to the folder “lib”, and add “Ds18b20” and “OneWire” folder to the Arduino library. Open the P06_Over_Temperature_Reminder with Arduino IDE and upload it.

What will you see

Upload the program and then open the serial monitor, you will see the temperature of one wire waterproof temperature sensor is printing. When the temperature is higher than 40℃.the led will light on and it will light off when the temperature is lower than 40℃.

Code overview

1. Import “OneWire.h” and “DallasTemperature.h” library for one wire waterproof temperature sensor.
2. Declare the pins of one wire waterproof temperature sensor and led.
3. Create a “oneWire” instance to communicate with any OneWire devices.
4. Pass instance we just created to Dallas temperature.
5. Initialize the serial and one wire waterproof temperature sensor and setup led as output.
6. Get temperature information and print.
7. Turn led on if temperature higher than 40, otherwise, turn led off.

Code usage

Import libraries: #include <OneWire.h> #include <DallasTemperature.h>

“OneWire.h” is a library for 1-Wire bus communication. The 1-Wire bus is a bus technology introduced by Dallas Semiconductor (which has been acquired by Maxim). One line completes two-way data communication. “DallasTemperature.h” is a library that has a layer on top of the “OneWire.h” library, which is convenient for directly using temperature sensors such as DS18B20. Therefore, using these two libraries, it will be very convenient for us to use one wire waterproof temperature sensor.

Create an instance: OneWire oneWire(ONE_WIRE_BUS);

Create an instance called “oneWire”. Therefore, we can use this instance to communicate with any OneWire devices.

Dallas Temperature: DallasTemperature sensors(&oneWire);

Pass the OneWire instance we just create to Dallas Temperature. This means that we can get the temperature of the one wire waterproof temperature sensor connected to the OneWire bus.

Initialize serial and sensor: Serial.begin(9600); sensors.begin();

Initialize the serial monitor and set the baud rate to 9600, you can set other baud rates, but remember to adjust the baud rate set in your own code in the serial monitor, otherwise garbled characters may appear. Initialize the sensor to make the sensor start work.

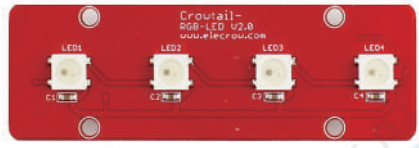
Get temperature: t = sensors.getTempCByIndex(0);

“getTempCByIndex()” is a function build-in “DallasTemperature.h” library. Using this function, we can easily get temperatures for one wire waterproof temperature sensor. The 0 in the brackets means that we only get the temperature from the first sensor.

Lesson 7 – Colorful RGB light

Introduction

The Crowtail- RGB-LED module with 4 pcs of WS2812B which is a Chainable & Addressable LED. Users can control all the LEDs with only one pin! Besides, the LED bar can be also chainable, that is, you can connect more than one LED bar to make your project more dreamful. In this module, you can control every LED with different colors at the same time.



Have you ever imagined that you have a colorful magic ball? Put this Crowtail-RGB-LED into a transparent sphere to achieve this effect! Maybe with some music, you will feel like being at the concert.

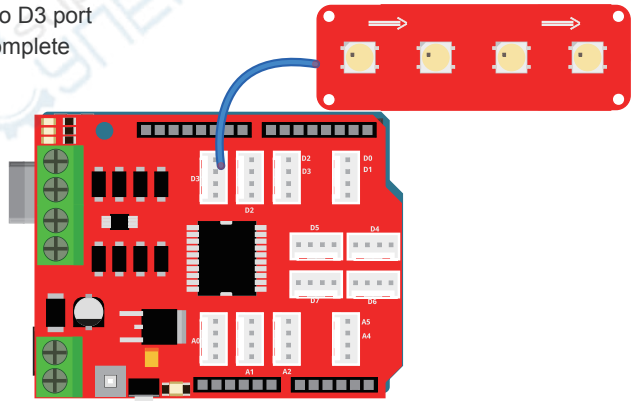
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - RGB-LED x1
- Crowtail - Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-RGB-LED to D3 port of Crowtail-Motor Base Shield. The complete connection is as follows.



Open the downloaded folder “Crowtail Deluxe Kit for Arduino demo code”, navigate to the folder “lib”, and add “Adafruit_NeoPixel” folder to the Arduino library. Open the P07_Colorful_RGB_Light with Arduino IDE and upload it.

What will you see

You will see that the dots on the Crowtail-RGB-LED will be filled with the same color in turn, first red, then green, and finally blue. Next, the RGB LED will light up in different colors like a rainbow.

Code overview

1. Import the RGB library.
2. Declare the pin of Crowtail-RGB-LED.
3. Create an RGB LED instance and select parameters for it.
4. Initialize the RGB strip.
5. Create functions that light up the RGB dots.
6. Call the function we create to show the RGB light.

Code usage

RGB library: #include <Adafruit_NeoPixel.h>

Import the RGB library. “Adafruit_NeoPixel.h” import many convenient functions to use the RGB, so we just need to import the library and use.

Create instance: Adafruit_NeoPixel strip = Adafruit_NeoPixel(4, PIN, NEO_GRB + NEO_KHZ800);

Create an RGB instance called strip. There are three parameters we need to pass to the instance. Parameter 1 is the number of pixels in strip. Parameter 2 is the pin number. Parameter 3 is the pixel type flags, you can add together as needed:
NEO_KHZ800 800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
NEO_KHZ400 400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
NEO_GRB Pixels are wired for GRB bitstream (most NeoPixel products)
NEO_RGB Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)

Choose RGB color: strip.Color(20, 0, 0);

“strip.Color()” is the build-in function of “Adafruit_NeoPixel.h” library. You can use this function to set the color of the RGB. For example, strip.Color(20,0,0) mean set RGB strip to red color. In addition, you can change the brightness of the RGB dots by changing the first parameter which we set to 20 in here.

Note: RGB is very bright, please protect your eyes and avoid looking directly.

Set pixel color: strip.setPixelColor(i, c);

“strip.setPixelColor()” is a built-in function of “Adafruit_NeoPixel.h” library. We use this function to set which pixel or color of RGB strip we need to show. The first parameter represents the number of pixels of the RGB strip, and the second parameter represents the color.

RGB show function: strip.show();

After we have set up how to display the RGB strip, we need to call this function to display the RGB strip we have set.

Lesson 8 – Show text on OLED

Introduction

Crowtail- OLED is constructed from the 128 x 64 dot-matrix OLED module. The display offers high brightness, self-emission, high contrast ratio, slim/thin outline, wide viewing angle, wide temperature range and low power consumption. You can almost display anything with this OLED module, such as text and pictures! This kind of OLED is very suitable to use in your project which needs a small display to show data or pictures.

In this lesson, we will learn how to display text using OLED. Then let's use this tiny display at the door of the company as a company's electronic business card, but a little different from ordinary business cards is that you can modify it when you have any new ideas!

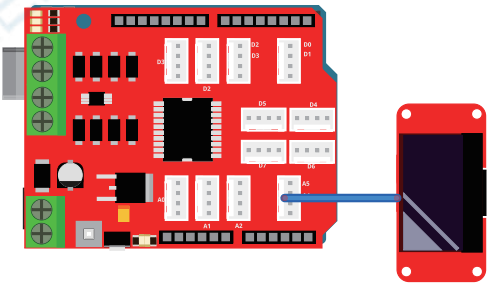
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - OLED x1
- Crowtail - Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-OLED to A4&A5 port of Crowtail-Motor Base Shield. The complete connection is as follows.



Open the downloaded folder “Crowtail Deluxe Kit for Arduino demo code”, navigate to the folder “lib”, and add “U8glib” folder to the Arduino library. Open the P08_Show_Text_On_OLED with Arduino IDE and upload it.

What will you see

You can see a larger "Elecrow" LOGO in the middle of the OLED and a smaller "elecrow" URL in the lower half of the OLED.

Code overview

1. Import the OLED library and create an instance of OLED.
2. Create a function to draw a big logo.
3. Create a function to draw the URL.
4. Call the function we created to show the Text.

Code usage

OLED Library: #include "U8glib.h"

“U8glib.h” is a library of OLED, it has very comprehensive usage methods and functions for you to call and use OLED. So, first to includes the U8glib.h library into your program and then you can call the function directly inside the library to using the OLED.

Create instant object: U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE);

Before you start using OLED, you need to initialize the OLED object. In fact, you need to tell which type of OLED is used(SSD1306), what is the pixel(128X64), and what is the type of communication between Arduino and OLED(U8G_I2C_OPT_NONE). Therefore, your initialized OLED instance can be used correctly by you.

Draw text: void drawLogo(uint8_t d){//draw text you want} void drawURL(void) {//draw text you want}

We created a drawLogo() method to draw our logo and a drawURL() method to draw our URL. In these two methods, we use the u8g.setFont() method to select the font we want to display and use the u8g.drawStr() method to set what position and character we want to draw. There are three parameters in the u8g.drawStr() function. The first parameter represents the abscissa position of the character we need to draw, the second represents the ordinate position of the character to be drawn, and the third represents the character to be drawn.

Show the text you draw: u8g.firstPage();do{//put what you want to draw here} while (u8g.nextPage()) ;

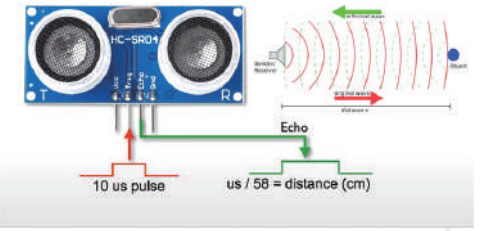
The drawing process must be performed in accordance with this framework, and the code displayed by the drawing is executed in a loop composed of firstPage and nextPage.

Lesson 9 – Ultrasonic ranging display

Introduction

This HC-SR04 has stable performance and high ranging accuracy. Compared to the Shap IR ranging module, HC-SR04 is cheaper than it. But it has the same ranging accuracy and longer ranging distance.

Ultrasonic pulses travel outward until they encounter an object, The object causes the wave to be reflected back towards the unit. The ultrasonic receiver would detect the reflected wave and stop the stop timer. Now read the time of the counter, which is the ultrasonic propagation time in the air. According to the formula: Distance = ECHO high level time X ultrasonic velocity (Speed of Sound in air 340m/sec) / 2, you can calculate the distance to the obstacle.



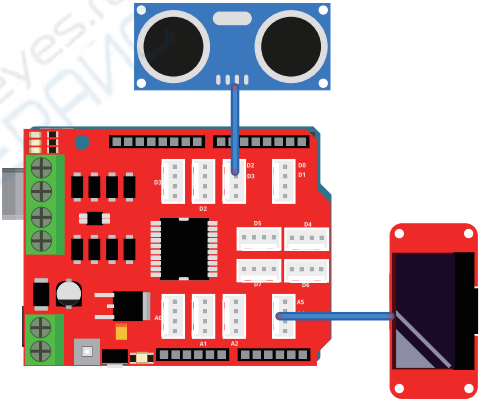
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - Ultrasonic Ranging Sensor x1
- Crowtail - OLED x1
- Crowtail - Cable x2

Hardware Connection

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

STEP2: Connect Crowtail-Ultrasonic Ranging Sensor to D2&D3 port of Crowtail-Motor Base Shield. Connect Crowtail-OLED to A4&A5 port of Crowtail-Motor Base Shield. The complete connection is as follows.



Open the downloaded folder “Crowtail Deluxe Kit for Arduino demo code”, navigate to the folder “lib”, and add “Ultrasonic” folder to the Arduino library. Open the P09_Ultrasonic_Ranging_Display with Arduino IDE and upload it.

What will you see

Move the object that the ultrasound is facing. You can see that the distance displayed by the OLED will change as the distance between the ultrasound and the object facing changes.

Code overview

1. Import the libraries of ultrasonic ranging sensor and OLED.
2. Create instances of ultrasonic ranging sensor and OLED.
3. Create variables to store distance and characters array to change distance using characters.
4. Create a function to draw the distance on OLED.
5. Read the distance from the ultrasonic ranging sensor.

6. If the distance is different from the previous distance, clear OLED and displays the new distance on OLED.
7. If the distance is the same as the previous distance, don't need to upgrade the display on OLED.

Code usage

Import libraries: #include <Wire.h> #include <Ultrasonic.h> #include "U8glib.h"

Import the libraries for using the ultrasonic ranging sensor and OLED. "Wire.h" is the library of I2C devices, we need to import this library to tell Arduino IDE before we use I2C devices. "Ultrasonic.h" is a library of ultrasonic ranging sensors. With the built-in method of this library, we no longer need to perform complex distance calculations to obtain the distance. "U8glib.h" is the library of OLED.

Create ultrasonic instance: Ultrasonic ultrasonic(2,3);

Create an ultrasonic instance object so we can use the function of the ultrasonic library to get the distance. Inside the parentheses, there are two parameters we need to pass. The first one is the pin of trig and the second one is the pin of echo.

Character array: char distance_value_temp[5] = {0};

Char is short for Character, which is to declare that we want to create character-related."distance_value_temp[5] = {0}" means that we are creating an array that can contain 5 elements. Since we used char to declare it, it is 5 characters. Finally, we assign it a value of 0.

Convert numbers to strings: itoa(int value, char *string,int radix);

Since we need to display the distance value on the OLED, the OLED can only display strings, but the distance value is an integer variable, so we have to convert it to a string to display on the OLED. "value": the data to be converted; "string": the address of the target string; "radix": the converted hexadecimal number, which can be decimal, hexadecimal, etc.

Get distance: Distance=ultrasonic.Ranging(CM);

"Ultrasonic.Ranging ()" is the function used to obtain the ultrasonic distance. You need to fill in the brackets in which units you want to get the distance value. Optional parameters are CM (centimeter), INC (feet). After getting the distance, we assign the distance to the variable "Distance".

Lesson 10 – Use of the joystick

Introduction

Crowtail- Thumb Joystick is a Crowtail compatible module that is very similar to the 'analog' joystick on PS2 (PlayStation 2) controllers. Two direction movements will output different analog signals as

they are actually two potentiometers. The resistor is 10k for each. The joystick also has a push-button that is could be used for special applications. When the module is in the working mode it will output two analog values representing two directions. The value is restricted in a little smaller range (e.g 200~800)compared to the normal joystick, while it is around 1023 when the button is pushed so that the MCU can detect the action of pressing.In this lesson,we will print the working status of the joystick in different positions on the serial monitor,

so that we can use it later to control the small smart car made by yourselves, such as the head rotation (servo), movement (motor), horn (buzzer), light (led), etc.

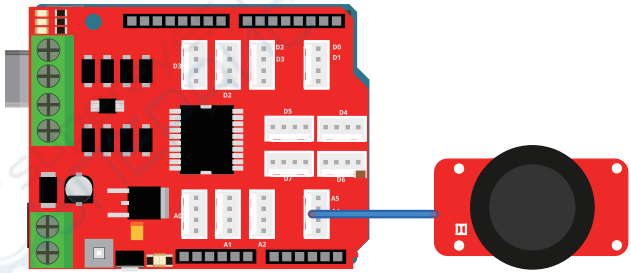
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - Thumb Joystick x1
- Crowtail - Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-Thumb Joystick to A4&A5 port of Crowtail-Motor Base Shield. The complete connection is as follows.



Open the P10_Use_Of_The_Joystick with Arduino IDE and upload it.

What will you see

Open the serial monitor, when you drag the joystick to different positions, you can see that the joystick will print the corresponding value, and you will also see the direction prompts, including up and down, left and right. But in fact, It can also detect the direction of the two axes at the same time! For example, the upper right direction.

Code overview

1. Declare two variables to store x, y axes value of joystick.
2. Initialize the serial monitor and set the baud rate for it.
3. Read the value of x, y axes of the joystick and store them in two variables.
4. Print the x, y axes value of joystick in decimal.
5. Use if/else statement to judge the direction of the joystick and print them.

Code usage

Analog input: xValue = analogRead(A5); yValue = analogRead(A4);

Read the analog input of the thumb joystick. Why we don't need to declare A4, A5 port as INPUT? As I said before, because Arduino pins are the default input mode, when the pin is input, we do not need to initialize the pin as input. Of course, you can also declare the pin as an input, which will be more standardized and easier to understand for others.

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

We use 4 if statements to determine the values of the x and y axes so that we can print a direction hint based on the position of the joystick. You may be confused by the value in the logical statement, it is just the value that I distinguish and set when I observe the change in the value of the joystick after changing the position of the joystick. So when you have a better idea, you can change your value.

Lesson 11 – 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. Have you thought about adding a rotatable part to your smart product, such as the robot's head and hands. In this lesson, we will use servo and joystick to make a servo controller that allows you to use the joystick to control the rotation of servo.

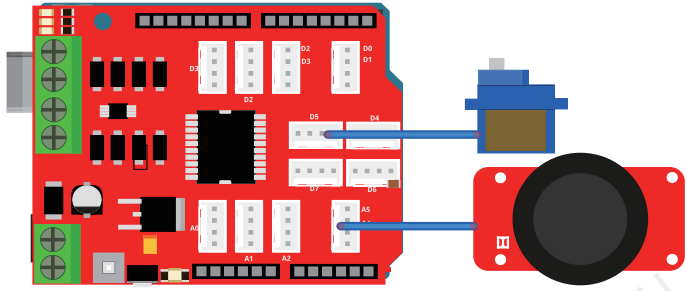
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - 9G Servo x1
- Crowtail - Thumb Joystick x1
- Crowtail - Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-9G Servo to D5 port of Crowtail-Motor Base Shield. Connect Crowtail-Thumb Joystick to A4&A5 port of Crowtail-Motor Base Shield. The complete connection is as follows.



Open the downloaded folder “Crowtail Deluxe Kit for Arduino demo code”, navigate to the folder “lib”, and add “Servo” folder to the Arduino library. Open the P11_Servo_Control with Arduino IDE and upload it.

What will you see

At first, the servo will rotate to 90 degrees of its midpoint. When you rock the joystick left and right along the x-axis, you can see that the servo will rotate to the corresponding angle as the joystick moves. Whenever you press the joystick cap, the servo will rotate its midpoint 90 degrees.

Code overview

1. Import the servo library and create an instance of the servo.
2. Declare two variables to store the joystick value and servo angle value.
3. Attaches the servo instance object on pin 5 and initialize it with 90 degrees.
4. Read the joystick value
5. If the joystick button is pressed, let the servo rotate to 90 degrees.
6. If the joystick button is not pressed, let the servo rotate to the corresponding angle as the joystick moves using map() function.

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 the rotation of servo. With using this library, we don't need to bother to check the contrast between the rotation angle of the PWM and the servo, we only need to input the angle we want the servo to rotate in the rotation function.

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 we need to control this 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.

Servo rotation function: myservo.write(90); myservo.write(angle);

The function of servo rotation. The prototype of this function is “ServoObject.write()”, The parameter in parentheses is the angular position you need the servo to rotate to, for our 180 degrees servo, the range of parameters you can enter is 0-180 degrees.

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 pos (between 200 and 800) to angle(between 0 and 180). For example, if the value of joystick pos is 300, after using the map() function, it will become about 30 and be assigned to the variable: angle.

Lesson 12 – Smart door

Introduction

When we go to some commercial buildings, the gate will open automatically when we walk to the door. Very convenient, right? In this lesson, we will use our own way to make a mini smart door using ultrasonic, servo and led modules. When we stand in front of the door, the door can open automatically. In addition, the red led can indicate the opening and closing of the door.

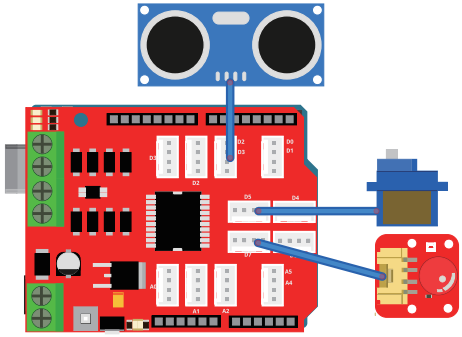
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - 9G Servo x1
- Crowtail - Ultrasonic Ranging Sensor x1
- Crowtail - LED x1
- Crowtail - Cable x2

Hardware Connection

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

STEP2: Connect Crowtail-9G Servo and Crowtail-LED to D5 and D7 port of Crowtail-Motor Base Shield. Connect Crowtail-Ultrasonic Ranging Sensor to D2&D3 port of Crowtail-Motor Base Shield. The complete connection is as follows.



Open the P12_Smart_Door with Arduino IDE and upload it.

What will you see

When you put your hand in front of the ultrasonic ranging sensor and make sure the distance between your hand and ultrasonic ranging sensor should less than 10 cm, you will see the servo will rotate to 170 degrees. And the LED will light off to indicate the door is opened. Until you move your hand away in front of the ultrasonic ranging sensor, the servo will rotate to 10 degrees and the LED will light on to indicate the door is closed.

Code overview

1. Import the ultrasonic ranging sensor and servo library.
2. Create instances of ultrasonic ranging sensor and servo.
3. Declare the port of led and some variables to store the distance and servo angle.
4. Attach the servo to D5 port and initialize the servo and setup the led as output.
5. Read the distance from the ultrasonic ranging sensor.
6. If the value of distance less than 10 cm, make the servo rotate to 170 degrees and turn led off.
7. If the value of distance more than 10 cm, make the servo rotate to 10 degrees and turn led on.

Code usage

Instances: Ultrasonic ultrasonic(2,3); Servo myservo;

Create an ultrasonic ranging sensor and servo instance object so we can get the distance and make the servo rotate to the angle we want. For ultrasonic object, there are two parameters inside the parentheses we need to pass. The first one is the pin of trig and the second one is the pin of echo.

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.

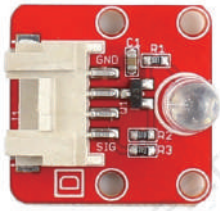
If/else Statements: if(logic statement) {code to be run if the logic statement is true} else {code to be run if the logic statement is false }

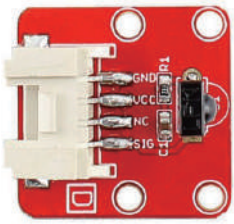
We use if / else statements to run the corresponding code based on the different distances detected by the ultrasonic ranging sensor. When the distance is less than 10 cm(no people are detected), first determine that the servo is at 170 degrees. If not, rotate the server to a 170-degree angle. If it is, then there is nothing to do. When the distance is more than 10 cm(people are detected), first determine that the servo is at 10-degrees. If not, rotate the server to a 10 degree angle. If it is, then there is nothing to do.

Lesson 13 – IR control

Introduction

The Infrared Emitter is an LED make from gallium arsenide, with its color centered around 940nm. It's used to transmit infrared signals through an infrared LED, while there is an Infrared receiver to get the signals on the other side. We can not only use the emitter to transmit data or commands, but also to emulate remotes to control your home appliance using an Arduino.





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. We often use the Crowtail-Infrared Receiver and the Crowtail-Infrared Emitter to work together.

Infrared control allows people to wirelessly control various electronic devices, especially widely used by household appliances, such as your TV remote control, air conditioner remote control, etc. In addition, it would be a good idea to use Infrared control to control your smart car. Therefore, in this lesson, we will use Crowtail- Infrared Emitter and Crowtail- Infrared Receiver to make our own IR control system to control led and buzzer.

Required Parts

- Arduino Uno/Crowduino Uno x2
- Crowtail - Motor Base Shield x2
- Crowtail - Infrared Emitter x1
- Crowtail - Infrared Receiver x1
- Crowtail - Collision Sensor x1
- Crowtail - LED x1
- Crowtail - Buzzer x1
- Crowtail - Cable x5

Hardware Connection

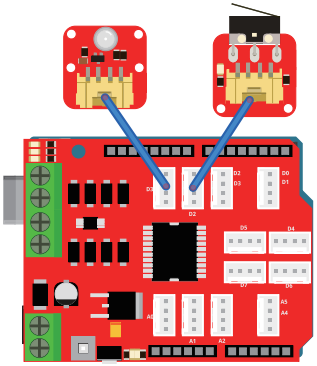
STEP1: Plug two Crowtail-Motor Base Shields onto two Arduino or Crowduino boards respectively.

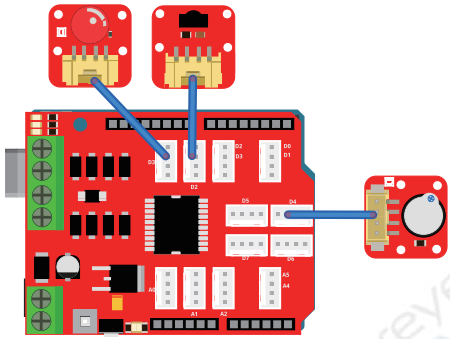
STEP2: Open the downloaded folder “Crowtail Deluxe Kit for Arduino demo code”, navigate to the folder “lib”, and add “IRSendRev” folder to the Arduino library.

STEP3: Connect Crowtail-Infrared Emitter and Crowtail-Collision Sensor to D3 and D2 port of Crowtail-Motor Base Shield. Open the P13_IR_Emitter inside the P13_IR_Control folder with Arduino IDE and upload it.

STEP4: Connect Crowtail-Infrared Receiver, Crowtail-LED and Crowtail-Buzzer to D2, D3 and D4 port of Crowtail-Motor Base Shield. Open the P13_IR_Receiver inside the P13_IR_Control folder with Arduino and upload it.

The complete connection is as follows.





What will you see

Point the infrared transmitter towards the infrared receiver. When you press the collision sensor for the first time, the LED is on. When you press the collision sensor a second time, the buzzer will sounds. When you press the collision sensor a third time, the led goes out and the buzzer no longer beeps.

Code overview

Infrared Emitter code:

1. Import the IR library and declare the pin of the collision sensor.
2. Declare three arrays to store the data we want to send.
3. Setup the collision sensor as input.
4. Create a function to scan if the collision sensor is pressed.
5. If the collision sensor is pressed for the first time, the data LED1 is sent.
6. If the collision sensor is pressed for the second time, the data LED2 is sent.
7. If the collision sensor is pressed for the third time, the data OFF is sent.

Infrared Receiver code:

1. Import IR library and declare led, buzzer and IR receiver pins.
2. Set the led and buzzer as outputs and initialize the IR receiver and serial monitor.
3. Receive data.
4. If the seventh bit of the data is equal to 1, turn on the LED.
5. If the seventh bit of the data is equal to 2, make the buzzer work.
6. If the seventh bit of the data is equal to 3, turn off the LED and stop the buzzer.

Code usage

IR library: #include <IRSendRev.h>

Import the IR library. Calling this library file can greatly simplify the complicated process when the infrared transmitter and infrared receiver send and receive data. It enables us to implement infrared data transmission and reception with simple functions.

Array: unsigned char LED1[] = {15, 70, 70, 20, 60, 5, 1, 0, 0, 0,0};

Create unsigned char array. 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. Unsigned char is an unsigned byte type. The size of a char type variable is usually 1 byte and it is an integer. Here we create three unsigned char arrays to store the data we want to send.

Send data: IR.Send(LED1, 38); IR.Send(LED2, 38); IR.Send(OFF, 38);

“IR.Send(Parameter1, Parameter2)” is an infrared send data function built-in the IR library. The parameter1 is the data we want to send and parameter2 is the frequency we send this data. You may confuse why IR emitter is connected to D3 port(pin3), that is because the D3 port is the PWM output port.

IR Receiver initialize: IR.Init(IR_Receiver);

“IR.init()” is the initialize function of the IR receiver. Before we receiver the data sent by the emitter, we have to initialize the IR receiver to tell which pin to obtain the data. Therefore, we need to enter the pin of the receiver in the parentheses.

Receive data: IR.Recv(dta);

“IR.Recv()” is the IR receive function of IR receiver that built-in the IR library. The parameter in parentheses is the address where the data is stored after the receiver receives the data. Here we store the data in the “dta” array we created.

Lesson 14 – Store data

Introduction

If you need to do some data storage in Arduino but found that the EEPROM in ATmega chip too limited, then this Crowtail-I2C EEPROM is your best choice. This module is based on the EEPROM chip AT24C256, which has 256k bit capacity. It communicates with Arduino with the I2C bus, helps you do much more data storage easily.
In this lesson, we will learn how to use it as a memory, store some data in it and read it out.

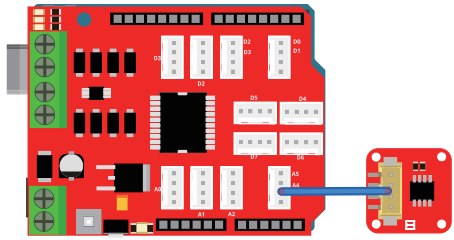
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - I2C EEPROM x1
- Crowtail - Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-I2C EEPROM to A4&A5 port of Crowtail-Motor Base Shield. The complete connection is as follows.



Open the P14_Store_Data with Arduino and upload it.

What will you see

Open the serial monitor and you will see 20 "." are printed in the writing test. In the reading test, you can see that a total of 20 letters were printed from A to T. These 20 letters are exactly the data we write in the EEPROM.

Code overview

1. Import the I2C library and macro define an I2C address for EEPROM.
2. Initialize the I2C and serial monitor and set baud to 9600.
3. Create the function of writing data and reading data.
4. Call the writing function to write the data into the EEPROM.
5. Call the reading function to read the data from the EEPROM and print.

Code usage

Micro define: #define EEPROM_ADDR 0x50

Micro defines the I2C address of EEPROM. The reason why I2C devices can be cascaded is that the address of each I2C device is unique, which allows the I2C device to not cause confusion when communicating. Here, the EEPROM default I2C address is 0x50.

Write data: void i2c_eeprom_write_byte(int deviceaddress, unsigned int eeaddress, byte data){ }

This is the function we create to write the data into the EEPROM. There are three parameters we need to pass for this function. Here, we will write 20 letters from A-T into the EEPROM.

Read data: byte i2c_eeprom_read_byte(int deviceaddress, unsigned int eeaddress){ }

This function is created to read the data we write in the EEPROM. We need to pass two parameters for this function. After we read the data from EEPROM, we will print them in the serial monitor.

Lesson 15 – Calculation of acceleration

Introduction

The Crowtail- 3-Axis Digital Accelerometer(±16g) is base on an advanced 3-axis IC ADXL345. This is a high-resolution digital accelerometer providing you at max 3.9mg/LSB resolution and large ±16g measurement range. Have no worry to implement it into your free-fall detection project, cause it's robust enough to survive up to 10,000g shock. Meanwhile, it's agile enough to detect single and double taps. It's ideal for motion detection, Gesture detection as well as robotics.this digital 3-axis accelerometer has excellent EMI protection.

Its variable output makes it suitable for a wide range of applications, such as HDD shock protection, Robotics, Smart vehicles and etc. Therefore, in this lesson, we will learn how to use this accelerometer and apply this accelerometer data to our project.

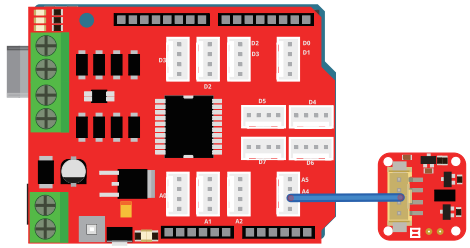
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - 3-Axis Digital Accelerometer(±16g) x1
- Crowtail - Cable x1

Hardware Connection

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

STEP2: Connect Crowtail-3-Axis Digital Accelerometer(±16g) to A4&A5 port of Crowtail-Motor Base Shield. The complete connection is as follows.



Open the downloaded folder “Crowtail Deluxe Kit for Arduino demo code”, navigate to the folder “lib”, and add “DigitalAccelerometer_ADXL345” folder to the Arduino library. Open the P15_Calculation_Of_Acceleration with Arduino and upload it.

What will you see

Open the serial monitor, and you will see that the accelerometer values for the X, Y, and Z axes are printed first, and then the acceleration values in the three axis directions are converted into G units and printed. When you swipe the accelerometer quickly along an axis, you will see the accelerometer value of that axis will change.

Code overview

1. Import the I2C and accelerometer libraries and create an instance of ADXL345 accelerometer.
2. Initialize serial monitor and ADXL345 accelerometer.
3. Read the accelerometer values of three axis and store them in variables.
4. Print the accelerometer values of three axis.
5. Get the accelerometer values of three axis in G units and print.

Code usage

ADXL345 library: #include <ADXL345.h>

Import the library of ADXL345 accelerometer. The calculation of acceleration is relatively complicated. Fortunately, we can use the functions in this library to simplify the difficulty of using the accelerometer. The I2C address of the accelerometer has been defined in the library file, so we will not declare the I2C address of the accelerometer again in the program.

Activity and inactivity threshold: adxl.setActivityThreshold(75); adxl.setInactivityThreshold(75);

Sets the byte which holds the threshold value for detecting activity/inactivity. The data format is unsigned, so the magnitude of the activity/inactivity event is compared with the value is compared with the value in the THRESH_ACT/THRESH_INACT register. The scale factor is 62.5mg/LSB. A value of 0 may result in undesirable behavior if the activity/inactivity interrupt is enabled. The maximum value is 255.

Read acceleration data: adxl.readXYZ(&x, &y, &z);

The function of reading the acceleration data from the ADXL345. We need to pass the address of three axis to this function. Each axis reading comes in 10-bit resolution, ie 2 bytes. Least Significant Byte first!! Thus we are converting both bytes into one int.

Read acceleration in G unit: adxl.getAcceleration(xyz);

Read the acceleration data in G unit. The parameters filled in the brackets are the arrays in which we need to store the accelerometers of the three axes read. The first, second and third elements of the array store the x-axis, y-axis and z Acceleration data.

Lesson 16 – Use of gyroscope

Introduction

The Crowtail- Analog Gyro is based on an angular velocity sensor(Murata-ENC-03R) that uses the phenomenon of Coriolis force, which is generated when a rotational angular velocity is applied to. It provides 1-axis velocity with analog output. with precision and quick-response measurements, this module suitable for applications with position& posture controlling.

This Crowtail- Analog Gyro is very easy to use, you can get the gyro angle data after transfer the analog values read from the gyro. The gyroscope is widely used in many areas, such as balance trolley, mobile phone and etc. If you have such a gyroscope, what would you use it for? Well, before thinking about this, let's start the subject of this lesson, that is how to use this analog gyro to get the angular velocity.

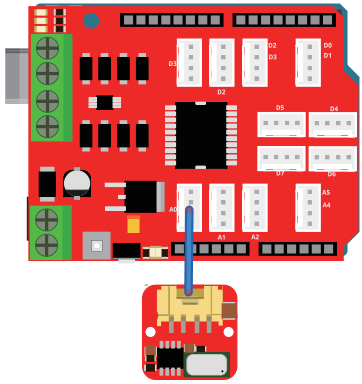
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - Analog Gyro x1
- Crowtail - Cable x1

Hardware Connection

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

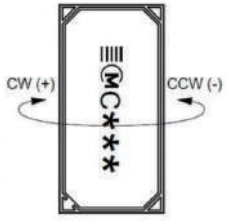
STEP2: Connect Crowtail-Analog Gyro to A0 port of Crowtail-Motor Base Shield. The complete connection is as follows.



Open the P16_Use_Of_Gyroscope with Arduino and upload it.

What will you see

Put the analog gyro on your desk horizontally and press the reset button on the Crowduino/Arduino to calibrate the sensor. Open the serial monitor, as you see the prompt “Now you can begin your test”, that means the calibration done. Then you will see the monitor is printing the angular velocity of the analog gyro. Try to turn the analog gyro sensor and you will see the angular velocity changing.
Rotating direction can reference the following picture:



Code overview

1. Import the math library and declare the pin of the analog gyro.
2. Declare variables to store the values of the analog gyro.
3. Setup the analog gyro as input and initialize the serial monitor.
4. In the setup() function, read the value from the analog gyro and use it as a reference value.
5. Read the value from analog gyro and convert it to angular velocity and print.

Code usage

Float and double: float reference_Value=0; double angularVelocity;

"Float" is a floating-point data type. The FLOAT data type is used to store single-precision floating-point numbers or double-precision floating-point numbers. Single-precision values of floating-point types have 4 bytes. "Double" (double precision floating point) is a data type used by computers. Rather than a single-precision floating-point number (float), a double (double-precision floating-point number) uses 64 bits (8 bytes) to store a floating-point number.

Reference value: reference_Value = sum/1000.0;

To obtain the reference value is the reason why we need to calibrate the analog gyro in beginning. We use for() statement to get the analog gyro data 1000 time in 2 seconds and then get the average of the value as a reference value. Therefore, within 2 seconds after uploading the program, you should not move the sensor because the sensor is being calibrated.

Lesson 17 – Use of Bluetooth

Introduction

Crowtail- Bluetooth Low Energy Module HM-13 is an easy module to use and designed for transparent wireless serial connection setup. The serial port Bluetooth module is fully qualified BT Version V4.0 EDR + BLE dual mode with complete 2.4GHz radio transceiver and baseband.

Bluetooth devices are widely used in our daily life, and they are powerful! Such as Bluetooth function of your mobile phone, Bluetooth headset, Bluetooth speaker and so on. In this lesson, we will learn how to use this bluetooth to control the led.

Required Parts

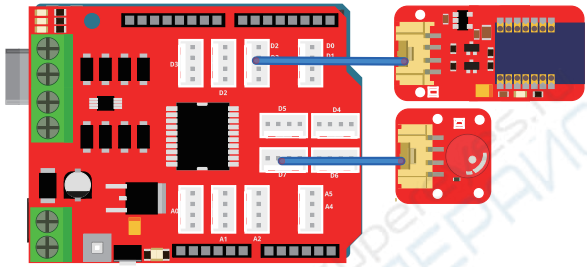
- Arduino Uno/Crowduino Uno x1
- Crowtail - LED x1

- Crowtail - Motor Base Shield x1
- Crowtail - Bluetooth Low Energy Module x1
- Crowtail - Cable x2

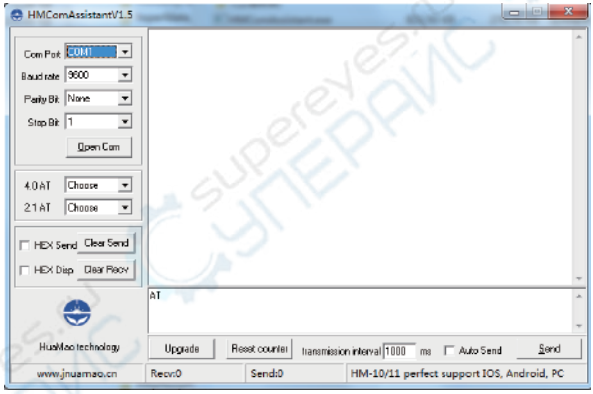
Hardware Connection

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

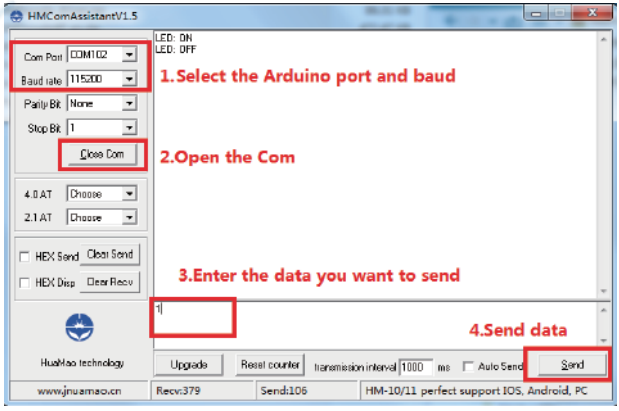
STEP2: Connect Crowtail-Bluetooth Low Energy Module to D2&D3 port of Crowtail-Motor Base Shield. Connect Crowtail-LED to D7 port of Crowtail-Motor Base shield. The complete connection is as follows.



Open the P17_Use_Of_Bluetooth with Arduino and upload it. Extract the compressed package named "HMPCComAssistant_en.rar" and open the serial port assistant software "HMComAssistant.exe" in the unzipped folder. You will see the software interface as shown below:



Then you should set the software as the following picture to control the led with bluetooth:



What will you see

Open the serial port assistant software "HMComAssistant.exe" and select the Arduino port and set baud(115200) for bluetooth, then press “Open Com” and enter the data you want to send to bluetooth(1 or 0 in our demo code). Finally, press “Send” to send the data to control the LED on and off. When you enter the data ‘1’ and send it, the LED will light on and receive the prompt “LED ON”. When you send the data "0", the LED will lights off and receive the prompt "LED OFF".

Code overview

1. Declare the pin of led, and a variable to store the data we send from the serial port.
2. Setup the led as output and initialize the serial monitor and set the baud(115200) that correspond to our bluetooth.
3. Check if the serial port has received the data from bluetooth. If yes, store the data in the variable “state”.
4. If data received is “0”, turn LED off.
5. If data received is “1”, turn LED on.

Code usage

Serial baud rate: Serial.begin(115200);

Set the baud for serial monitor correspond to our bluetooth module. “115200” is the default baud rate for our bluetooth module. If we set the wrong baud for Arduino serial monitor, it may cause communication failure or garbled communication between Bluetooth and Arduino.

Received data: Serial.available()>0

Determine if the serial monitor has received data. We send the data to Bluetooth, and then Bluetooth sends the data to the Arduino serial monitor. If we send data, the expression "Serial.available ()> 0" is true, otherwise it is false.

Read serial data: state = Serial.read();

Function to read data from the serial monitor. Then, we store the data in the variable "state", and we turn on or off the led based on the value of this variable.

Lesson 18 – ESP8266 web server

Introduction

This is Crowtail- ESP8266 Node MCU, it adds six crowtail interfaces on the board, so you can easily use this board with other crowtail modules. We intergrate a USB-Serial chip that can upload code.

It also has auto-reset so no noodling with pins and reset button pressings. To make it easy to use for portable projects, we added a connector for 3.7V Lithium polymer batteries and built-in battery charging. You don't need a battery, it will run just fine straight from the micro USB connector. But, if you do have a battery, you can take it on the go, then plug in the USB to recharge. The board will automatically switch over to USB power when it's available. Comes fully assembled and tested, with a USB interface that lets you quickly use it with the Arduino IDE or NodeMCU Lua. (It comes preprogrammed with the Lua interpreter) We also toss in some header so you can solder it in and plug into a solderless breadboard. In this lesson, we will learn how to use the wifi function of Crowtail- ESP8266 Node MCU module to control the LED and buzzer, but actually you can control more other devices you want.

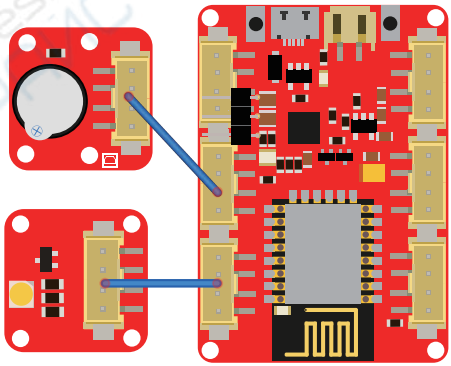
Required Parts

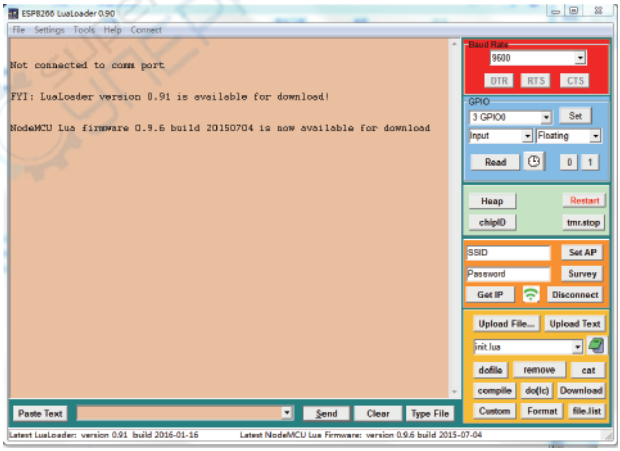
- Crowtail - ESP8266 Node MCU x1
- Crowtail - Bright LED x1
- Crowtail - Buzzer x1
- Crowtail - Cable x2
- Micro USB Cable x1

Hardware Connection

STEP1: Connect Crowtail-Bright LED and Crowtail-Buzzer to D1 and D2 port of Crowtail-ESP8266 Node MCU.

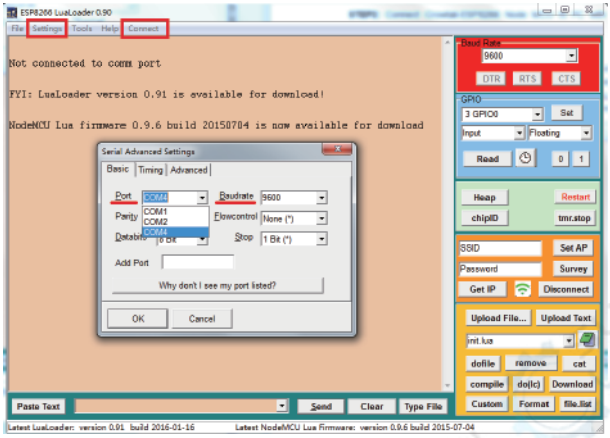
STEP2: Connect Crowtail-ESP8266 Node MCU to PC with a micro USB cable. The complete connection is as follows.



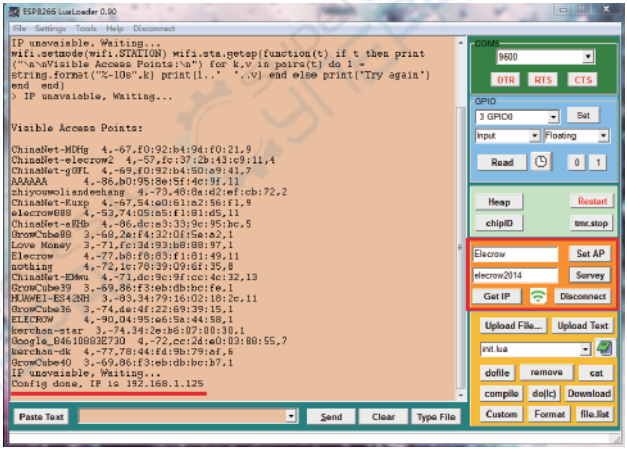


For ESP8266 Node MCU, We will use luaEditor to edit the code and use LuaLoader to compile and upload the code file. Go to the "P18_ESP8266_Web_Server" directory, then go to the "Tool" directory and open the application named "LuaLoader.exe" located in the "LuaLoader (version 0.90)" folder. Then, you will see the software window as below:

1. Click the menu “Setting”, choose the ‘Comm Port Settings”, and it’ll popup an interface “Serial Advanced Setting”, you should set the port(The port of my board is 4, but yours may be different, you can check the port of the board in the port of your computer) for Crowtail-ESP8266 Node MCU and baud rate there. After correctly set, click the “Connect” option on the top of the menu.



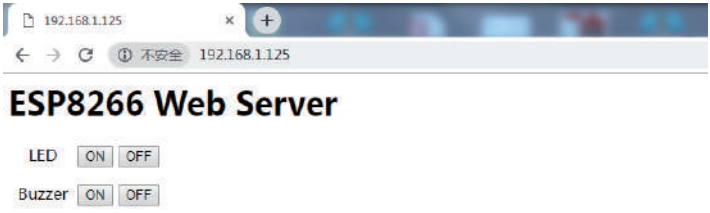
2. Connect the wireless networks around you. We can click the “Survey” option to see all WIFI can obtain around you. Then enter the WIFI name and password and click “Set AP” option. Finally, click “Get IP” option to get the IP for Crowtail-ESP8266 Node MCU.



3. Click “Upload File...” option, open “P18_ESP8266_Web_Server" folder, then choose “init.lua” and “webserver.lua” in the “Program” folder for uploading.

What will you see

Copy the IP of Crowtail-ESP8266 Node MCU and open it with browser, you could see the web page shown below. You can use the button on the web page to control the LED and buzzer:



Use overview

1. Open “LuaLoader.exe” software and set port and baud for Crowtail-ESP8266 Node MCU.
2. We can use “luaEditor.exe” software to open .lua file and edit code to control buzzer and LED through Crowtail-ESP8266 Node MCU. Of course, you can also program and control other electronic modules.
3. Set WIFI for Crowtail-ESP8266 Node MCU and get it IP.
4. Upload the "webserver.lua" program to Crowtail-ESP8266 Node MCU.
5. Copy the IP and use the browser to web page of Crowtail-ESP8266 Node MCU server.
6. Click the button on the web page to turn LED and buzzer on or off.

Lesson 19 – A naughty robot

Introduction

Can OLED display only strings? Of course not! OLED can also display pictures, just need to simply convert the picture into a bitmap to display the picture. So in this course, we will learn how to display pictures on an OLED module. Used in combination with Crowtail-ultrasonic distance sensor and Micro-Speed Motor, it can create a naughty robot. If nothing stops him in front, he will run endlessly.

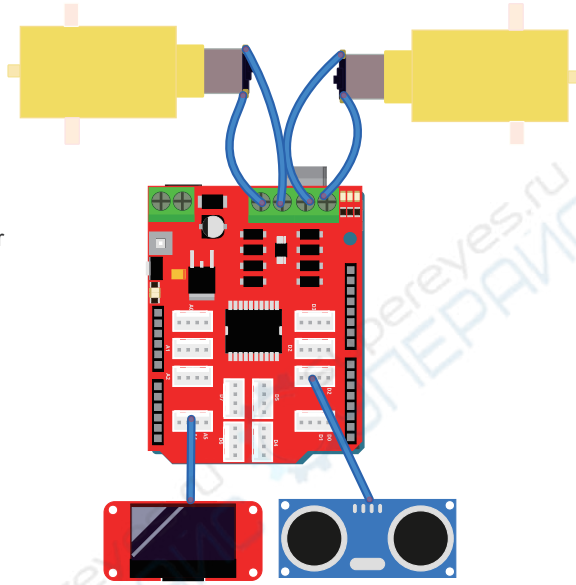
Required Parts

- Arduino Uno/Crowduino Uno x1
- Crowtail - Motor Base Shield x1
- Crowtail - OLED x1
- Crowtail - Ultrasonic Ranging Sensor x1
- Micro-Speed Motor
- Crowtail - Cable x2
- Jumper Wire x4

Hardware Connection

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

STEP2: Connect Crowtail-OLED to A4&A5 port of Crowtail-Motor Base Shield. Connect Crowtail-Ultrasonic Ranging Sensor to D2&D3 port of Crowtail-Motor Base Shield. Connect the two Micro-Speed Motor to the motor interface of Crotail-Motor Base Shield. The OUT1 and OUT2 interfaces are connected to the same motor positive and negative poles, and the OUT3 and OUT4 interfaces are connected to the same motor positive and negative poles. Be careful not to connect the motor to the interval interface, such as OUT1 and OUT3, otherwise the motor will not work properly. The complete connection is as follows.



Open the P19_A_Naughty_Robot with Arduino and upload it.

Note: Because the motor consumes more power, after uploading the code, you need to provide additional power for the Crowtail-Motor Base Shield (usually connect 9V power supply for the power input port of the motor base shield) and then turn on the onboard power switch(you don't have to power Crowduino/Arduino, because the motor base shield will share the power with Crowduino/Arduino).

What will you see

If there is nothing in front of the ultrasonic ranging sensor, the two micro-speed motors will always run, and a smiling face will be displayed on the OLED, just like a naughty robot. Put your hand or other obstacles in front of the ultrasonic ranging sensor, you will stop the two micro-speed motors and an angry expression will be displayed on the OLED.

Code overview

1. Import the I2C, ultrasonic ranging sensor and OLED libraries.
2. Create an instance object for ultrasonic ranging sensor and OLED.
3. Declare the bitmap of a happy and angry image.
4. Declare some variables for the motor pin and ultrasonic ranging sensor.
5. Setup all the pins of the motor as output.
6. Create functions to control the motor and draw images on OLED.
7. Read the value of the distance from the ultrasonic ranging sensor.
8. If the distance's value lower than 10 cm, draw an angry image and make the two motors stop.
9. If the distance's value more than 10 cm, draw a smile image and make the two motors keep running.

Code usage

Create instant object: U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE);

Before you start using OLED, you need to initialize the OLED object. In fact, you need to tell which type of OLED is used(SSD1306), what is the pixel(128X64), and what is the type of communication between Arduino and OLED(U8G_I2C_OPT_NONE). Therefore, your initialized OLED instance can be used by you normally.

Bitmap: static const unsigned char happy[] U8G_PROGMEM={} static const unsigned char angry[] U8G_PROGMEM={}

Declare the bitmap and assign it. This is the U8glib library method. Generally, we will make a modification to the name, such as happy and angry, and then the content in {} is the font of the image you generated with the bitmap software. You only need to put The generated font is copied here and the bitmap can be called in the program to display on the OLED.

Draw a bitmap: u8g.drawXBMP(0,0,128,64,happy); u8g.drawXBMP (0,0,128,64,angry);

“u8g.drawXBMP()” is a function of draw bitmap, Its function prototype is: void drawXBMP(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h, const u8g_pgm_uint8_t * bitmap), x: the abscissa of the upper left corner of the bitmap, y: the ordinate of the upper left corner of the bitmap, w: the width of the bitmap, h: the height of the bitmap, *bitmap: bitmap object.
Here we use this function to draw the two bitmaps happy and angry in the area where the abscissa is 0 pixels, the ordinate is 0 pixels to the abscissa is 128 pixels, and the ordinate is 68 pixels (ie, the entire OLED screen area).

Lesson 20 – A remote control car

Introduction

If you have a good understanding of our electronic modules, then you must know that our modules can almost meet all your needs for making your own smart car! It's time to start making our own car! In this lesson, we will use the infrared remote control method to control the car, and make a simple remote control car through the infrared emitter and receiver, button, buzzer, motors, LED and so on. Less gossip, let's start making!

Required Parts

- Arduino Uno/Crowduino Uno x2
- Crowtail - Motor Base Shield x2
- Crowtail - Buzzer x1
- Crowtail - Switch x1

- Crowtail - IR Emitter x1
- Crowtail - IR Receiver x1
- Crowtail - Thumb Joystick x1
- Crowtail - Bright LED x1
- Crowtail - Collision Sensor x1
- Crowtail - Cable x7
- Micro-Speed Motor x2
- Jumper Wire x4

Hardware Connection

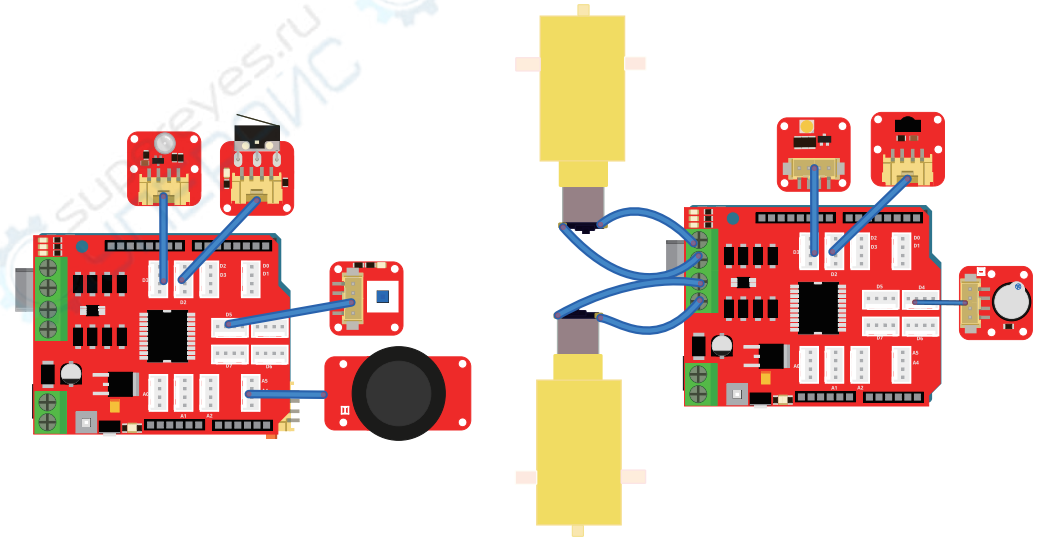
STEP1: Plug the Crowtail-Motor Base Shield onto the Arduino or Crowduino board.

STEP2: Connect Crowtail-Thumb Joystick to A4&A5 port of Crowtail-Motor Base Shield. Connect Crowtail-IR Emitter, Crowtail-Collision Sensor and Crowtail-Switch to D3, D2 and D5 port of Crowtail-Motor Base Shield. Open the P20_IR_Emitter_Code inside the P20_A_Remote_Control_Car folder with Arduino IDE and upload it.

STEP3: Connect Crowtail-IR Receiver, Crowtail-Bright LED and Crowtail-Buzzer to D2, D3 and D4 port of Crowtail-Motor Base Shield. Connect the two Micro-Speed Motor to the motor interface of Crowtail-Motor Base Shield. The OUT1 and OUT2 interfaces are connected to the same motor positive and negative poles, and the OUT3 and OUT4 interfaces are connected to the same motor positive and negative poles. Be careful not to connect the motor to the interval interface, such as OUT1 and OUT3, otherwise the motor will not work properly. Open the P20_IR_Receiver_Code inside the P20_A_Remote_Control_Car folder with Arduino IDE and upload it.

Note: Because the motor consumes more power, after uploading the code, you need to provide additional power for the Crowtail-Motor Base Shield (usually connect 9V power supply for the power input port of the motor base shield) and then turn on the onboard power switch(you don't have to power Crowduino/Arduino, because the motor base shield will share the power with Crowduino/Arduino).

The complete connection is as follows.



What will you see

Point the infrared transmitter towards the infrared receiver. When you press the collision sensor, the buzzer will beep and when the collision sensor is released, the buzzer will stop beeping. When you press the switch, the bright led will light on and when the switch button is pop up, the bright led will light off. When you push the joystick forward along the y-axis, you will see two micro-speed motors turn clockwise. When you push the joystick backward along the y-axis, you will see two micro-speed The motor will rotate counterclockwise. When you release the joystick and the joystick returns to the middle position, you will see that the two micro-speed motors will stop rotating.

Code overview

IR_Emitter_Code:

1. Import IR library and declare the pin of collision sensor and switch.
2. Declare some variables to store the joystick, collision sensor and switch state.
3. Declare some data we going to send.
4. Setup the collision sensor and switch as output.
5. Create functions to check if the switch and collision sensor are pressed.
6. Send related data according to the actions of collision sensor, switch and joysticks.

IR_Receiver_Code:

1. Import IR library and define the pin of IR receiver, led and buzzer.
2. Declare the pins of the motor pin and array to store the data received.
3. Setup all motor pins as output and initialize the IR receiver.
4. Create some motor working functions to be called when different data is received.
5. According to the received data, the motor working method is called.
6. The motor working functions are called according to the received data.

Code usage

Array: unsigned char LED[] = {15, 70, 70, 20, 60, 5, 1, 0, 0, 0,0};

Create unsigned char array. 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. Unsigned char is an unsigned byte type. The size of a char type variable is usually 1 byte and it is an integer. Here we create seven unsigned char arrays to store the data we want to send.

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.