

Crowtail Starter Kit for Raspberry Pi

User Guide

V2.1 2022

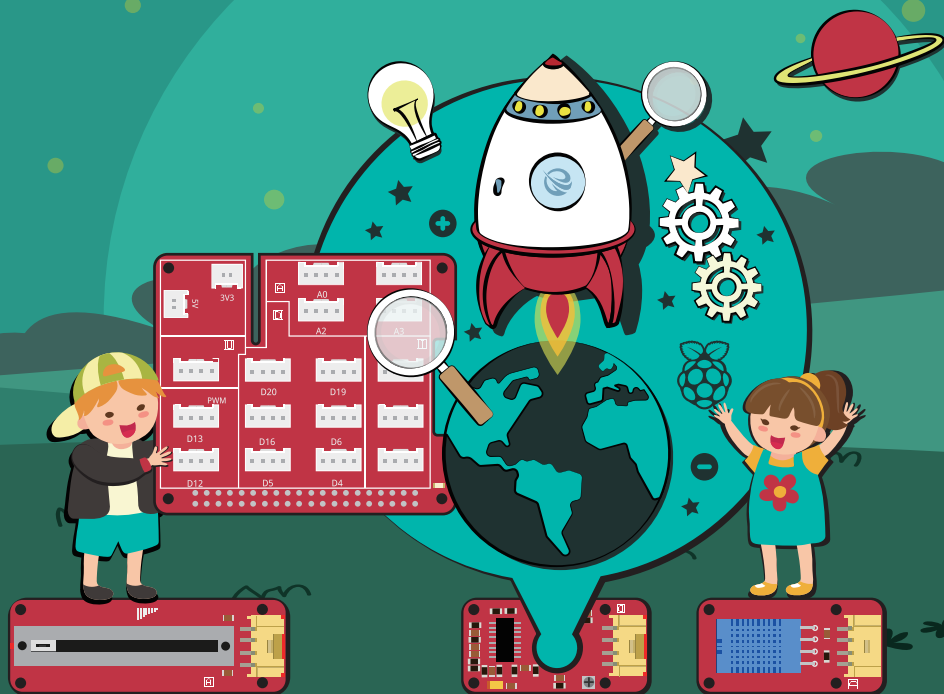


Content

Instruction	02
Meet the Raspberry Pi	03
Python	06
Basic python and linux usage	08
What is Crowtail	10
Modules list	12
Learning journey	13
• Lesson 1 Hello world	13
• Lesson 2 Blinking light	14
• Lesson 3 Control LED	16
• Lesson 4 Doorbell	18
• Lesson 5 Secret of the treasure chest	20
• Lesson 6 Bright and dark	22
• Lesson 7 Smart light	24
• Lesson 8 Collision alert	25
• Lesson 9 Car tracking	27
• Lesson 10 Plant doctor	29
• Lesson 11 Dimmer	31
• Lesson 12 Variable speed fan	33
• Lesson 13 Servo control	35
• Lesson 14 Obstacle alert	38
• Lesson 15 Smart lantern	41
• Lesson 16 Distance display	43
• Lesson 17 Weather station	45
• Lesson 18 Remote control	48
• Lesson 19 Speed measurement	50
• Lesson 20 Plant cultivation expert	52
• Lesson 21 Remote control door	54

Instruction

Have you ever wanted to start learning and building your Raspberry Pi project but found no toolkit to get started? Have you ever try to use your Raspberry Pi to learn Python but found that there are few kits and tutorials on the market to get started? This Crowtail-Starter Kit for Raspberry Pi will be one of the best kits for you to get started with Raspberry Pi and Python! This kit includes a Crowtail-Base Shield for Raspberry Pi and more than 22 modules like RGB LED, ultrasonic ranging sensor, IR and etc. Each module is selected from hundreds of Crowtail module sets, designed to provide you with the most suitable and introductory courses for learning. With this kit, you will learn how to control digital and analog modules, digital-to-analog conversion, PWM control, serial communication, I2C communication, and more. In addition, this kit provides rich, interesting and open-minded courses, so that you can not only learn hardware and programming knowledge but also can distribute your thinking and guide you to think further. There are no complicated jumpers and soldering, so you can focus on learning Raspberry Pi, hardware principles and Python programming. All you need to do is to plug the modules into the base shield and start the project with Raspberry Pi quickly and comprehensively.

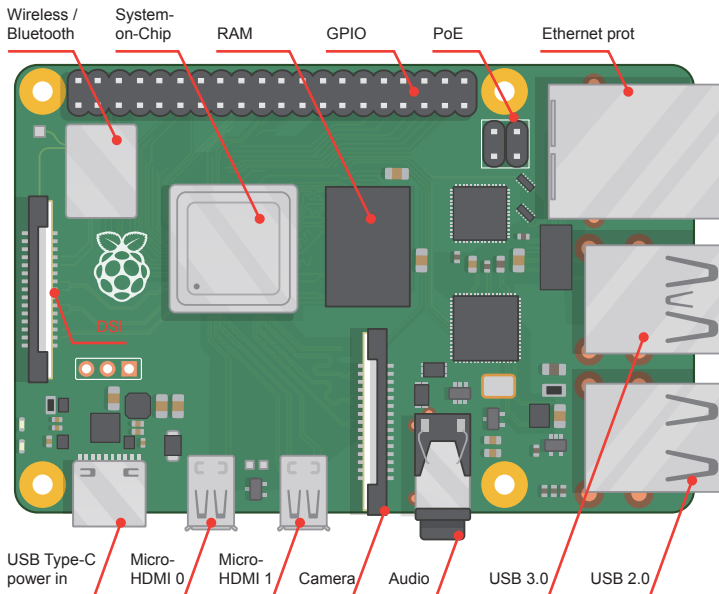


Meet the Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and works with a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

What's more, the Raspberry Pi has the ability to interact with the outside world and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We hope to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work. You can view more information on this link "<https://www.raspberrypi.org/>".

Raspberry Pi 4 Hardware Overview



The Raspberry Pi 4 is the newest version of Raspberry Pi, which takes Pi to another level, with performance that's good enough to use in a pinch as a desktop PC. Do you only have Raspberry Pi 3? Don't worry, this kit can also be used with Raspberry Pi 3B, 3B + and even Raspberry Pi Zero.

You have a few options when it comes to interacting with the Raspberry Pi.

Option 1: Full Desktop Setup

The first and most common is to use it like you would a full desktop computer (just smaller). This involves connecting a keyboard, mouse, and monitor. With this setup, you are likely best served by installing Raspbian with Desktop, which gives you a full graphical user interface (GUI) to work with.

This is the best option if you want an experience similar to working with other operating systems (OS), such as Windows, macOS, or other popular Linux flavors, like Ubuntu.



Option 2: Headless Pi

The other option is to create a headless setup, which means you can skip the monitor, keyboard, and mouse. While this is the cheaper way to go, it means you'll need to be open to performing all your actions in the command line interface. For this, you will want either Raspbian with Desktop or Raspbian Lite operating systems.

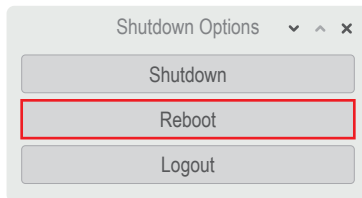
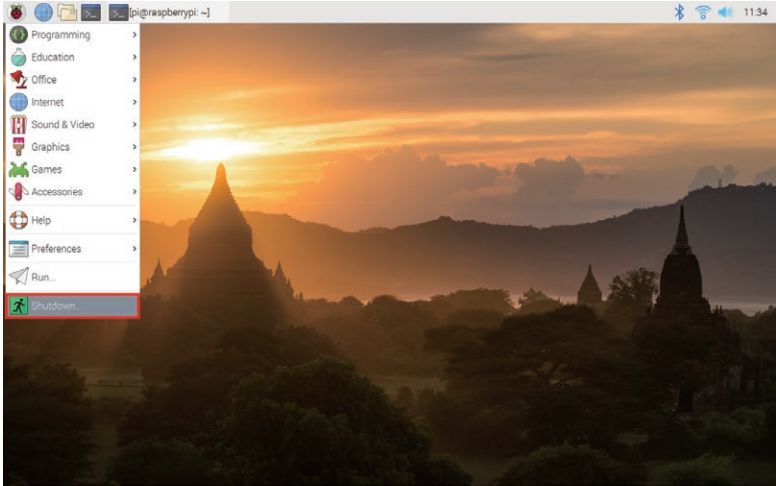


In this guide, we will focus on the first option (use Raspberry Pi like a full desktop computer).

• **Step 3:**

Restart your Pi to ensure that the changes to take effect. Click on Pi Applications menu-->Shutdown.

Since we just need to restart, click on Reboot button.



Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. It is a beginner-friendly programming language that is used in schools, web development, scientific research, and in many other industries.



Why is python? There are the following reasons.

1. Python Has a Healthy, Active and Supportive Community

Python is been around for quite some time, so there's plenty of documentation, guides, tutorials and more.

2. Python Has Some Great Corporate Sponsors

It helps big time when a programming language has a corporate sponsor. In Google's case, they created a vast quantity of guides and tutorials for working with Python.

3. Python Has Big Data

It is one of the most popular languages used in data science, second only to R. It's also being used for machine learning and AI systems and various modern technologies.

4. Python Has Amazing Libraries

Python has an excellent selection of libraries. In other words, there are library-like tools that offer cross-platform support, which is a huge benefit.

5. Python Is Reliable and Efficient

Python is speedy, reliable and efficient. You can work with and deploy Python applications in nearly any environment, and there's little to no performance loss no matter what platform you work with.

6. Python Is Accessible

For newcomers and beginners, Python is incredibly easy to learn and use. In fact, it's one of the most accessible programming languages available

Python is usually pre-installed on the Raspberry Pi system. Python includes two versions, python2 and python3. In this guide, we will use python2. There are two ways to interact with python. One uses the interpreter and the other runs the python program from a file. Python is an interpreted language, and using an interpreter means that we can run programs without running a separate compilation step. But usually we want to save the commands together in one or more files so that we can run them all at once. In this tutorial, we will focus on run python from a file.

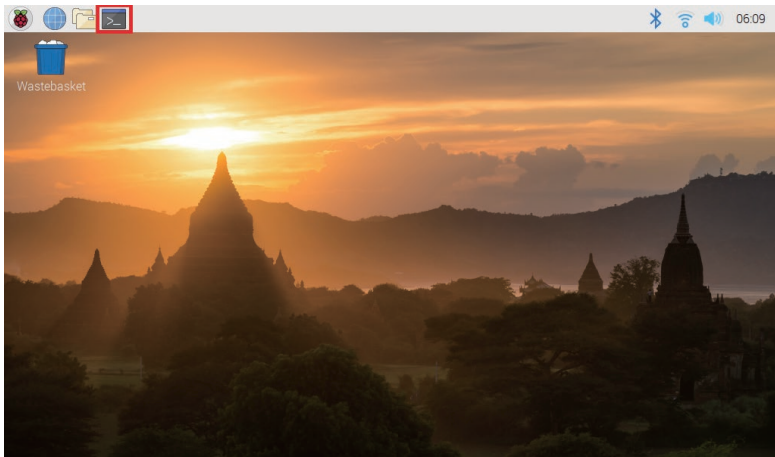
Basic python and linux usage

When start to use Raspberry Pi, we can use an incredible way to execute scripts without the need to download each one separately, which is called “git cloning”.

But the premise is that the network must be connected for the Raspberry Pi.

In simple terms: we will clone the GitHub directory where all the example scripts are located into our desktop environment so we won't need to download every single script every time. And how we gonna do that? Follow the steps below and using the git clone command:

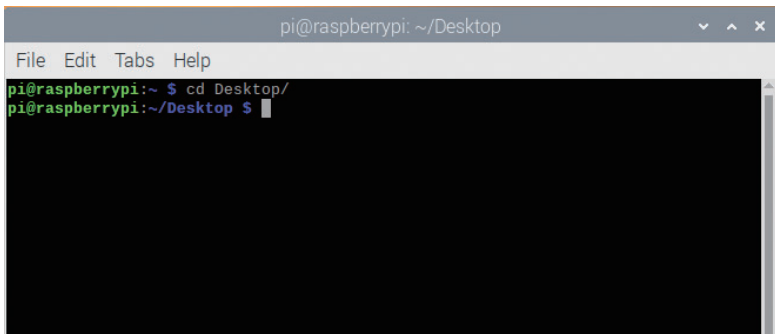
1. Open “terminal”. It looks like a black screen and we will use it to execute most of our python scripts and to download extensions and scripts from GitHub.



2. After opening terminal successfully we need to clone the scripts directory into our desktop.

To do so you need to type in the terminal: `cd Desktop`

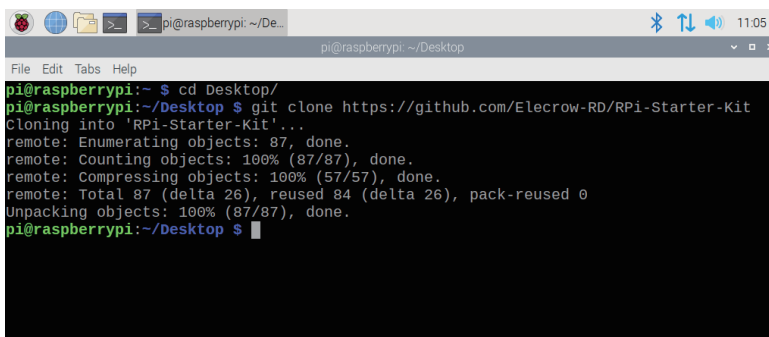
3. Press “Enter” on your keyboard. Now you should be in your desktop folder which is your desktop that you see.



4. Inside the terminal type the following command:

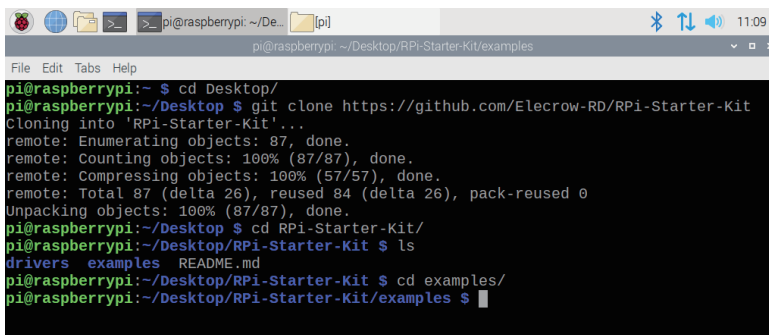
```
git clone https://github.com/Elecrow-RD/RPi-Starter-Kit
```

5. Press ENTER and wait till the cloning process is completed.



```
pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~$ cd Desktop/
pi@raspberrypi:~/Desktop$ git clone https://github.com/Elecrow-RD/RPi-Starter-Kit
Cloning into 'Rpi-Starter-Kit'...
remote: Enumerating objects: 87, done.
remote: Counting objects: 100% (87/87), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 87 (delta 26), reused 84 (delta 26), pack-reused 0
Unpacking objects: 100% (87/87), done.
pi@raspberrypi:~/Desktop$
```

6. After cloning the git repository, it will show up on our desktop as a “RPi-Starter-Kit” folder. Let’s go into that folder by “cd” command so we can use the scripts that are inside.



```
pi@raspberrypi: ~/Desktop/RPi-Starter-Kit/examples
File Edit Tabs Help
pi@raspberrypi:~$ cd Desktop/
pi@raspberrypi:~/Desktop$ git clone https://github.com/Elecrow-RD/RPi-Starter-Kit
Cloning into 'Rpi-Starter-Kit'...
remote: Enumerating objects: 87, done.
remote: Counting objects: 100% (87/87), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 87 (delta 26), reused 84 (delta 26), pack-reused 0
Unpacking objects: 100% (87/87), done.
pi@raspberrypi:~/Desktop$ cd RPi-Starter-Kit/
pi@raspberrypi:~/Desktop/RPi-Starter-Kit$ ls
drivers  examples  README.md
pi@raspberrypi:~/Desktop/RPi-Starter-Kit$ cd examples/
pi@raspberrypi:~/Desktop/RPi-Starter-Kit/examples$
```

7. Write the command “sudo python <script name>” in order to execute the python script, for example, “sudo python 1_hello_world.py”.

```
sudo python 1_hello_world.py
```

The “sudo” command gives us root permissions (admin permissions) which are required by the GPIO library, afterward we write “python” to tell the system that we want to execute a command using the python library. In the end, we will write the script name as we downloaded it to the desktop, and that’s it! You’ve successfully executed the python script.

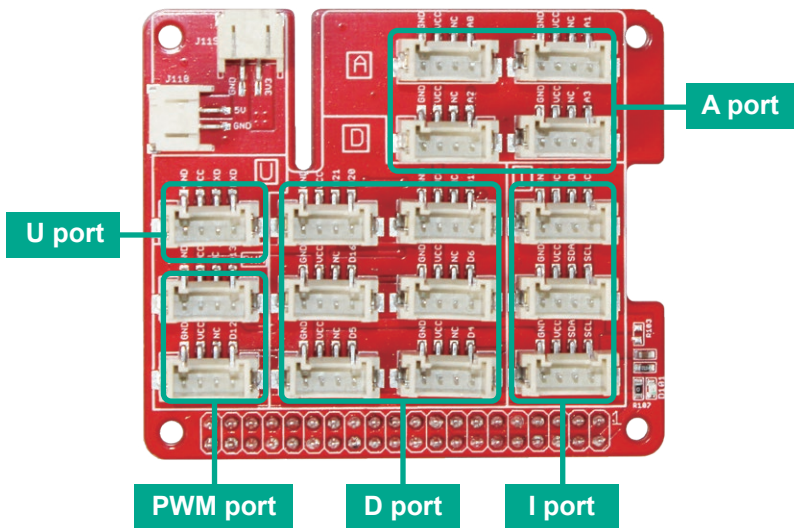
What is Crowtail

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 Base Shields 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 simple and easy to manage an electronic project!

Crowtail – Base Shield for Raspberry Pi



- **U port:** 1 UART port that has a mark of “**U**”.

These interfaces can be used for UART communication such as the WIFI module or Bluetooth module;

- **I port:** 3 IIC ports that have a mark of “**I**”.

These interfaces are for IIC Communication, users can utilize 2 IIC modules at the same time;

- **D port:** 6 Digital I/O ports that have a mark “**D**”.

These ports can be used to read and control digital Crowtail modules, such as the Button and LEDs;

- **A port:** 4 Analog ports (A0~A3) that have a mark of “**A**”.

Besides the function of digital, these A ports can read the analog signal, such as a potentiometer or light sensor;

- **PWM port:** 2 PWM ports labeled “**PWM**”.

There are two hardware PWM ports on the basic shield board, which can be used to adjust the brightness of the LED, use the buzzer to play different tones, rotate the servo, etc.

This is a Crowtail base shield for Raspberry Pi with UART/I2C/Analog/Digital interface. All the Crowtail modules can be plugged into Raspberry Pi through this base shield. Moreover, this shield has an on-board ADC chip so that analog output modules can be used on Raspberry Pi. The ADC chip used is MCP3008. It talks to Raspberry Pi using the SPI interface.

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 - Base Shield for Raspberry Pi using a Crowtail cable, which is a huge improvement over the previously troublesome jumper connections.

Modules list

- Crowtail - Base Shield for Raspberry Pi x1
- Crowtail - Button x1
- Crowtail - Buzzer x1
- Crowtail - LED Green) x1
- Crowtail - LED Red) x1
- Crowtail - Touch Sensor x1
- Crowtail - Hall Sensor x1
- Crowtail - PIR Motion Sensor x1
- Crowtail - Moisture Sensor x1
- Crowtail - Light Sensor x1
- Crowtail - Rotary Angle Sensor x1
- Crowtail - IR Reflective Sensor x1
- Crowtail - Collision Sensor x1
- Crowtail - MOSFET x1
- Crowtail - 9G Servo x1
- Crowtail - Ultrasonic Ranging Sensor x1
- Crowtail - Temperature& Humidity Sensor x1
- Crowtail - RGB-LED x1
- Crowtail - I2C LCD x1
- Crowtail - IR Receiver x1
- Infrared Remote Control x1
- Micro-Speed Motor x1

Learning journey

Lesson 1 – Hello world

Instruction

Do you know what the first program is usually for people learning code? That must be the program "Hello, World"! It can also be used as a sanity test to make sure that a computer language is correctly installed, and that the operator understands how to use it. Well, let's print out your first greeting with python!

Target

- Learn how to run a python program from a file.
- Learn how to print the phrase with python.
- Learn how to comment out a text.

Programming and note

• Print function:

We use the print function to print the text "Hello World".

Note that the text you need to print should be inside double quotes when you use python 3.

• Comments:

A comment is any text to the right of the pound (or hash) symbol #.

The Python interpreter ignores this text, and it can be useful for writing notes to yourself or other programmers about what's going on in the code.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 1_hello_world.py
```

Result

You can see that your Python first greeting "Hello World" is printed.

Thinking

How do we print when we want to print a changing value instead of text?

Lesson 2 – Blinking light

Instruction

If "Hello World" is the first program in the software section, what is the first program in the hardware? That must be a blinking LED. When a hardware problem occurs, we usually use this program to determine what the problem is.

Target

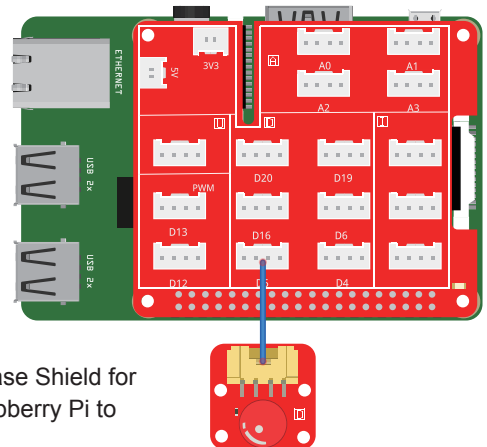
- Learn how the LED work and use it to make a blinking light.
- Learn how to use GPIO of Raspberry Pi.
- Learn how to create variables.
- Learn how to control output signals.
- Learn how to pause code.

Required materials

- Crowtail - Base Shield for Raspberry Pi x1
- Crowtail - LED (Red) x1
- Crowtail - Cable x1

Hardware learning and connection

The LED is the best choice to help you learn I/O pins. What you need to do is connecting the LED module to the Base Shield D ports, then download the program to the Arduino. Besides the very basic usage, you can make the LED blink with the frequency you want, thus the brightness with PWM. Actually, LED is the most popular used for human interface.



Connect Crowtail - LED to D5 port of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.

Programming and note

• Import library:

“import” is the statement for importing a library(also can be called a module). By importing a module, we can use the programmed functions in the module. We can use the import statement to import modules with the following syntax:

```
import module1 [, module2 [, ... moduleN]]
```

You can name the module file whatever you like. For example, create an alias for RPi.GPIO module called GPIO:

```
import RPi.GPIO as GPIO
```

In this case, the two modules RPi.GPIO and time are pre-packaged with Raspbian, so we can directly import these two modules and rename RPi.GPIO to GPIO. After introducing these two modules, we can control the Raspberry Pi GPIO and runtime.

• Creating variables: led_pin = 5

Variables are containers for storing data values. Unlike other programming languages, Python has no command for declaring a variable. A variable is created the moment you first assign a value to it. Here, we will create a variable to store the pin of the LED.

• GPIO schemes: GPIO.setmode(GPIO.BCM)

There are two possible Raspberry Pi GPIO schemes: GPIO.BOARD and GPIO.BCM. Before using the Raspberry Pi GPIO port, we need to simply use "GPIO.setmode ()" to set the pin mode. In this guide, we will focus on GPIO.BCM.

• Setup pin: GPIO.setup(led_pin, GPIO.OUT)

After setting the mode of GPIO, we will use GPIO.setup() to set the specified GPIO port as input or output. Here, we will set led as an output pin.

• Output: GPIO.output(led_pin, GPIO.HIGH) GPIO.output(led_pin, GPIO.LOW)

By using GPIO.output() function, we can set a pin to high or low logic signal. There are two parameters of GPIO.output() function, the first one is the pin we want to set, the second one is the state(HIGH or LOW) we set for the pin. We can also use 1 and 0 to represent HIGH and LOW.

• Delay: time.sleep(0.5)

We use time.sleep() to delay the running of the code. In parentheses, we need to fill in the time we want to delay, expressed in seconds.

• Cleanup (Release Occupation): GPIO.cleanup()

Here we need to develop a good habit. When we are not using GPIO signals, we should use cleanup () to set all pins to a low state to release occupation. Otherwise, the pin logic level may always be high.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 2_blinking_light.py
```

Result

You will see the LED will light up for 0.5 seconds and then the LED will go out.

Thinking

Try to make it like a home light, use the switch to turn the light on and off.

Lesson 3 – Control LED

Instruction

Now that we have learned how to light up the LED, can we use our module to control the light on and off? Of course, in order to call for energy saving, let's make a switch light that only controls the light for a while each time.

Target

- learn how the button work and use it to control the LED.
- Learn how to read the digital input.
- Learn how to run the code forever.
- Learn how to use if/else statement.
- Learn how to catch and handle errors.

Required materials

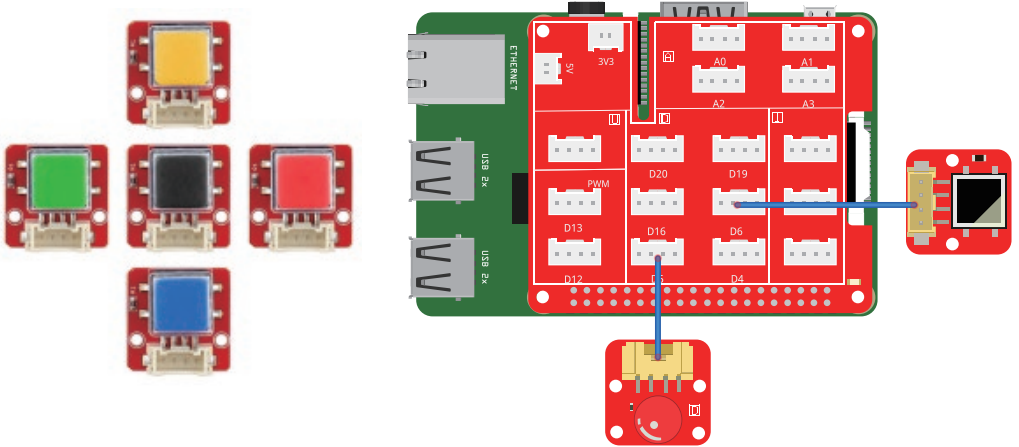
Crowtail - Base Shield for Raspberry Pi x1
Crowtail - Button x1

Crowtail - LED (Red) x1
Crowtail - Cable x2

Hardware learning and connection

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

Connect Crowtail - LED and Crowtail - Button to D5 and D6 port of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

- **Setup pin:** `GPIO.setup(button_pin, GPIO.IN)`

We use `GPIO.setup()` to set the specified GPIO port as input or output.

Here, we will set led as an output pin and we will set the button as an input pin to detect if the button is pressed.

- **Loop code forever:**

With the while loop, we can execute a set of statements as long as a condition is true.

In this case, we use "While True" here which means the condition will true(1) forever, so the code inside of while True will execute forever.

- **If/else statement:**

You can use this if / else statement when there are two cases and you want different code to be executed in different cases. If the "if" conditional statement is true, the code inside the "if" is executed; if the conditional statement is false, the code inside the else is executed. When we detect the button_pin is true(pressed), then the code inside of "if" will be executed, otherwise, the code inside of else will be executed.

- **Input:** `GPIO.input(button_pin)`

By using the `GPIO.input ()` function, we can detect whether the pin is a high or low logic signal. This function requires a parameter, which is the pin you want to detect.

- **Equal to: ==**

You might be confused by the "==" sign, which means equal (to compare whether the objects are equal), return true (1) if they are equal, otherwise false (0). This condition can be used in several ways, most commonly in "if statements" and loops. For example "1 == 2", this returns 0.

• Try Except:

The try block lets you test a block of code for errors.

The except block lets you handle the error.

“KeyboardInterrupt” mean user interrupts execution(usually enter key “control+c”).

When you press “control+c”, the program runs the code in except block. In this lesson, we will use this release occupation of GPIO when user interrupt(key “control+c”) is detected.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 3_control_led.py
```

Result

When you press the button, the LED is on; when the button is released, the LED is off.

Thinking

Can you make a more intelligent light, so that this light will automatically light up when someone is there, such as a corridor light?

Lesson 4 – Doorbell

Instruction

I remembered an awkward experience when my friend came to visit my house and knocked on the door, I was listening to the songs. But the knock on the door was too low for me to hear. It wasn't until five minutes later that I heard the knock and opened it. It was really an awkward experience, so I decided to make a doorbell so that I could hear it more clearly when there were visitors.

Target

- Learn how the buzzer and touch sensor work and use them to make a doorbell.

Required materials

Crowtail - Base Shield for Raspberry Pi x1

Crowtail - Touch Sensor x1

Crowtail - Buzzer x1

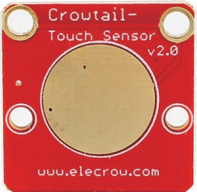
Crowtail - Cable x2

Hardware learning and connection



Buzzer

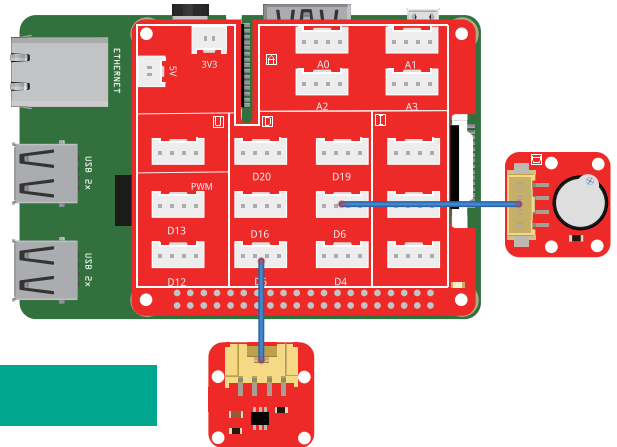
The buzzer module is for making sound in your project. It sounds when activated by a logic HIGH signal. Connect the buzzer to digital ports of Crowtail - Base Shield for Raspberry Pi, you can easily make it sounds with setting the related ports to logic HIGH. The buzzer module can be also connected to an analog pulse-width modulation(PWM) output to generate various tones, which means you can use it to compose your own melody!



Touch Sensor

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

Connect Crowtail - Touch Sensor and Crowtail - Buzzer to D5 and D6 ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

- **Input: `GPIO.input(touch_pin)`**

Use `GPIO.input()` function to detect the state of the touch sensor, if the touch sensor is touched, the state of it is 1(HIGH); if the touch sensor is not touched, the state of it is 0(LOW).

- **If/else statement:**

Use if / else statements to execute the corresponding code based on whether the state of the touch sensor is equal to 1. If yes, make the buzzer sound for 3 seconds; if not, make the buzzer stops beeping.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 4_doorbell.py
```

Result

When you touch the touch sensor, the buzzer will keep beeping for 3 seconds. After 3 seconds, when you stop touching the touch sensor, the buzzer will stop beeping.

Thinking

What if someone rings the doorbell while we listen to music in the room or go to the bathroom? We may not hear even such a loud doorbell, how to solve it?

Lesson 5 – Secret of the treasure chest

Instruction

Have you ever seen a treasure chest? Once the treasure chest is opened, it will dazzle and cool. Let's do a project with similar effects in this lesson. Put this in your box and let it change from a simple box to a radiant treasure chest!

Target

- Learn how the hall sensor work and use it to make a Luminous treasure box with LEDs.

Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - LED(Green) x1
Crowtail - LED(Red) x1

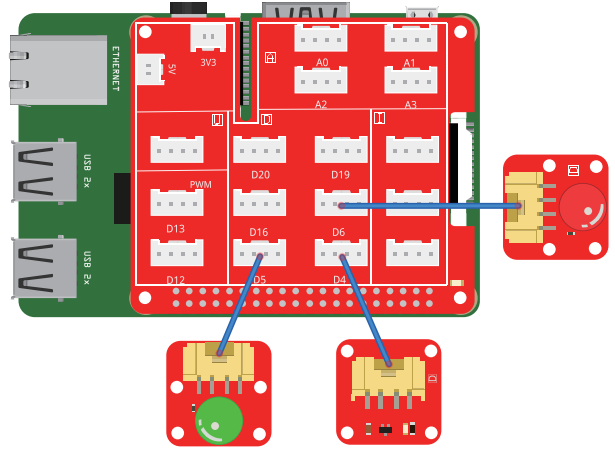
Crowtail - Hall Sensor x1
Crowtail - Cable x3

Hardware learning and connection

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 its output logic high(turn off) when the magnetic field disappears.



Connect Crowtail - Hall Sensor, Crowtail - LED(Green) and Crowtail - LED (Red) to D4, D5 and D6 ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

• If/else statement:

Compared to the if / else statements you used before, you can find that the conditions of the if statement is slightly different. This is because the Hall sensor outputs a low level when the south pole of the magnet is detected and outputs a high level when the south pole of the magnet is not detected. Therefore, in this case, when the Hall sensor detects the south pole of the magnet, we will make the two LEDs flash (like the treasure inside glitters when the treasure box is opened), otherwise, the two LEDs are off (the treasure box close).

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 5_treasure_chest.py
```

Result

Use the south pole(S) of the magnet close to the hall sensor, you will find that the two colors LEDs will always flash. When you remove the magnet or use the magnet north pole(N) near the hall sensor, you will find that the two LEDs will light off.

If you want to apply the project to your box, don't forget to modify the condition of the if statement, that is, the two LEDs need to be turned on when the state of the hall sensor is 1 while turned off when the state of the hall sensor is 0.

Thinking

Do you know any other applications for Hall sensors? Can we use it to test speed?

Lesson 6 – Bright and dark

Instruction

Do you remember the thinking problem of lesson 3?

How to make a smart light that can automatically light on when the day is dark and light off when the day is bright? Don't worry, this is what we will do in this lesson.

Target

- Learn how the light sensor work and use it to make an automatically light with LED.
- Learn how to read the analog values from an analog module.
- Learn how to set up the SPI bus.

Required materials

Crowtail - Base Shield for Raspberry Pi x1

Crowtail - Light Sensor x1

Crowtail - LED(Green) x1

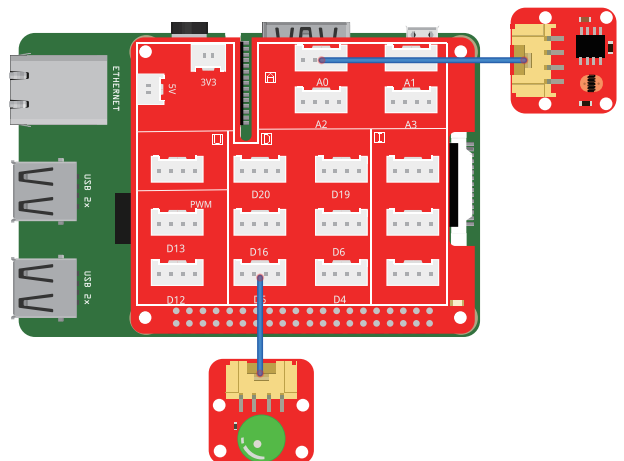
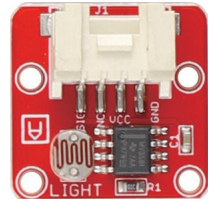
Crowtail - Cable x2

Hardware learning and connection

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

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

Connect Crowtail - Light Sensor and Crowtail - LED to A0 and D5 ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

• SPI module: import spidev

Thanks to the on-board ADC chip of base shield, analog modules can be used on Raspberry Pi. But before we import the SPI module and use this chip to communicate with Raspberry Pi through SPI, we need to activate the SPI-interface of the Raspberry Pi (we have already done this before starting lessons).

• Functions: def readadc(parameter1,...)

A function is a block of code that only runs when it is called. You can pass data, known as parameters, into a function and function can return data as a result. We add only parameter in this function, but inside the parentheses, you can add many parameters as you want, just separate them with a comma. Here, we will use this function to read SPI data (analog value). Analog value can be any value from 0-1023 (1024 numbers)

• Return values: return adcout

To let a function return a value, use the return statement. In this function, we set the function returns the analog values read from SPI when we call.

• Calling a function: value = readadc(0)

To call a function, use the function name followed by a parenthesis. We will get the analog values from A0 port and store it in the variable "value".

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 6_bright_and_dark.py
```

Result

Use your hand to block the light sensor or when the day is dark, the LED will light on automatically, and it will light off when you remove your hand from the light sensor or the day is bright.

Thinking

When the light sensor detects that the current brightness is dark, the LED will light up, and it will stay on even if there is no one there, which is not energy efficient or not smart enough. How can we solve this problem?

Lesson 7 – Smart light

Instruction

Thinking back to the question left over from the previous lesson, how to make a more energy-efficient and "smart" lamp? There are usually several different solutions to a project. In this lesson, we will change the way to make a smart light, try to find out the difference between them to see which is better.

Target

- Learn how the PIR motion sensor work and use it to make a smart light with LED.

Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - PIR Motion Sensor x1

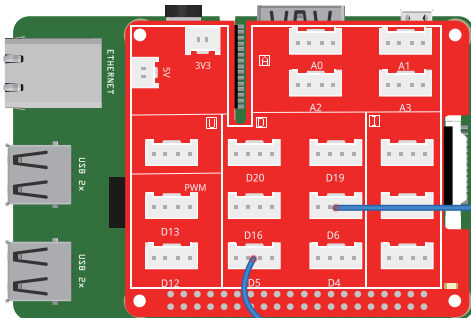
Crowtail - LED(Green) x1
Crowtail - Cable x2

Hardware learning and connection

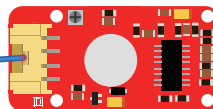
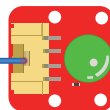
Crowtail - PIR Motion Sensor(Passive Infrared Sensor) can detect infrared signals caused by motion. If the PIR sensor notices the infrared energy, the motion detector is triggered and the sensor outputs HIGH on its SIG pin. The detecting range can be adjusted by a potentiometer soldered on its circuit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



Note: We design that when the PIR module detects an object movement, it will change from low level to high level and output high level continuously for 5 seconds, which means you don't need to set extra delay.



Connect Crowtail - PIR Motion Sensor and Crowtail - LED to D5 and D6 ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

- **If/else statement:**

If the statement of pir motion sensor is 1(detect movement), turn on the LED for 5 seconds, otherwise, turn LED off.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 7_smart_light.py
```

Result

Wave your hand or make a movement within the detection range of the PIR motion sensor, the LED will light for 5 seconds; after 5 seconds, if you stop waving or moving, the LED will turn off.

Thinking

No matter the day or night, this LED will still light up whenever someone moves, so it is still defective. How can we combine this lesson with lesson 6 to form a smart energy-saving light that turns on only when someone's movement is detected and it's dark?

Lesson 8 – Collision alert

Instruction

Do you know how the car triggers a collision alert and opens the airbag to protect us after detecting a collision? In fact, what a car uses to detect a collision is a collision sensor. In this lesson, we will use this sensor to simulate the process of a car collision and alarm.

Target

- Learn how the collision sensor work and use it to make a collision alert with buzzer and LED.

Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - Collision Sensor x1
Crowtail - Buzzer x1

Crowtail - LED(Red) x1
Crowtail - Cable x3

Thinking

Collision sensors do alert us after a collision, but we need to take precautions, such as how do we remind us before the car hits an obstacle?

Lesson 9 – Car tracking

Instruction

Have you ever played a car that tracks? Do you know how the car realize the function of tracking? Let me tell you, the tracking function of the car is generally inseparable from the infrared reflection module. This module will output different levels of signal when it encounters white and black objects. Let's take a look.

Target

- Learn how the IR reflective sensor work and use it to make a tracking experiment with two LEDs.

Required materials

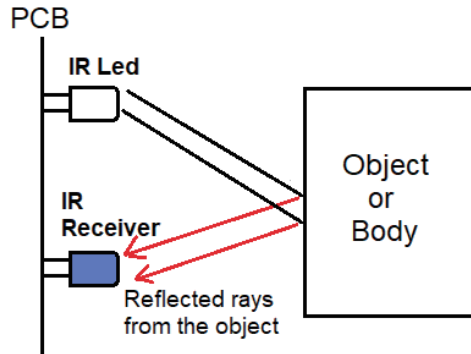
Crowtail - Base Shield for Raspberry Pi x1
Crowtail - IR Reflective Sensor x1
Crowtail - Cable x3

Crowtail - LED(Green) x1
Crowtail - LED(Red) x1

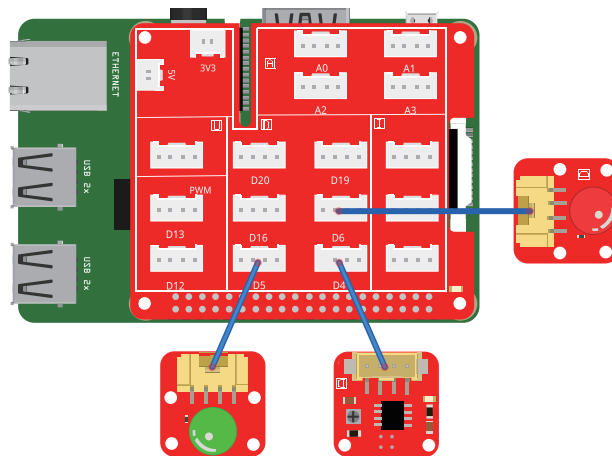
Hardware learning and connection

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 means, there is no black obstacle in front of the sensor, this module output logic LOW, and vice versa. There is also an on-board potentiometer to adjust the sensitivity, and the effective workingdistance of the sensor can be adjusted to 4-15mm. 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.





You may wonder why IR Reflective Sensor can achieve tracking? Let's introduce the principle of IR Reflective Sensor tracking. When the infrared light emitted from the infrared led of the IR Reflective Sensor hits the surface of a white object, most of the infrared light is reflected back to the IR Receiver, but when the light hits the surface of a black object, black (the color that absorbs light) takes all the infrared light is absorbed, so the infrared will not be reflected back to the IR Receiver.



Connect Crowtail - IR Reflective Sensor, Crowtail - LED(Green) and Crowtail - LED(Red) to D4, D5 and D6 ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.

Code overview

1. Import the GPIO and time libraries.
2. Set GPIO mode, define variables to store pins and set up.
3. If the state of IR reflective sensor is 1, turn on the green LED and turn off the red LED, then print "on track" prompt.
4. If the state of IR reflective sensor is 0, turn off the green LED and turn on the red LED, then print "off track" prompt.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 9_car_tracking.py
```

Result

Use a black object to block the infrared reflection sensor, and you will see that the green LED is on, and the red LED is off, and the "on track" prompt is printed on the terminal; use a white object to block the infrared reflection sensor, and you will see that the red LED is on and the green LED is on Off and an "off track" prompt is printed on the terminal.

Thinking

Can an infrared reflectance sensor achieve tracking? If not, how many of these sensor carts are needed to achieve the tracking function?

Lesson 10 – Plant doctor

Instruction

As we all know, the main substances are needed by plants include water, light, minerals and so on. They are so fragile and we need to take good care of them so that they can grow healthy into a big plant. In this course, we will be the doctors of plants, and when the plants are "thirsty" we will know in time.

Target

- Learn how the moisture sensor work and use it to make a plant water shortage reminder with buzzer and LED.

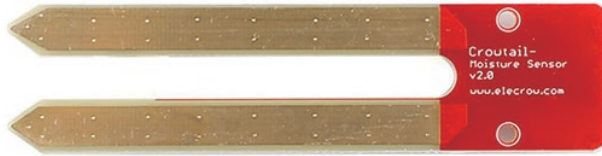
Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - Moisture Sensor x1
Crowtail - Buzzer x1

Crowtail - LED(Green) x1
Crowtail - Cable x3

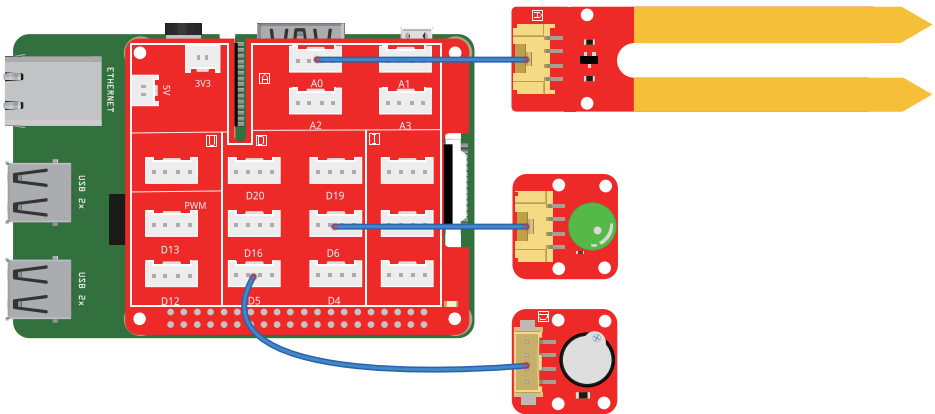
Hardware learning and connection

This Moisture Sensor can be used to detect the moisture of soil and thus to monitor if the plants in your garden need some water.



This sensor uses the two probes to pass current through the soil, and then it reads the resistance to get the moisture level. More water makes the soil conduct electricity more easily (less resistance), while dry soil conducts electricity poorly (more resistance). Compared to the other moisture sensors using the same moisture test method, this module has super long legs, making it suitable for actual applications.

Connect Crowtail - Moisture Sensor, Crowtail - Buzzer and Crowtail - LED (Green) to A0, D5 and D6 ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Code overview

1. Import all the libraries we need.
2. Set GPIO mode, define variables to store pins and set up.
3. Open the SPI bus.
4. Define a function to read SPI data.
5. Calling the function to get the analog value and store in variable "value".
6. If "value" lower than 300, make the buzzer work and the LED lights off.
7. If "value" greater than 300, make the buzzer stop working and the LED lights on.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 10_plant_doctor.py
```

Result

When the soil moisture detected by the sensor is less than 300 (1023 in total), the buzzer works and the green LED is turned off (water shortage); when the soil moisture detected by the sensor is 300 or more, the buzzer stops working, The green LED lights up (no water shortage).

Thinking

This can indeed detect whether a plant is lacking water, but how can we solve the problem when the plant is lacking light or other factors?

Lesson 11 – Dimmer

Instruction

We must not read in bright or dim light, as both environments can harm our eyes. Our eyes have different requirements for light brightness in different environments, so it is necessary to adjust the brightness of the lamp so that we can adapt to different brightness environments. Now let's make a mini dimmer!

Target

- Learn how the rotary angle sensor work and use it to make a mini dimmer with LED.
- Learn how to use PWM to control the brightness of the LED.

Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - Rotary Angle Sensor x1

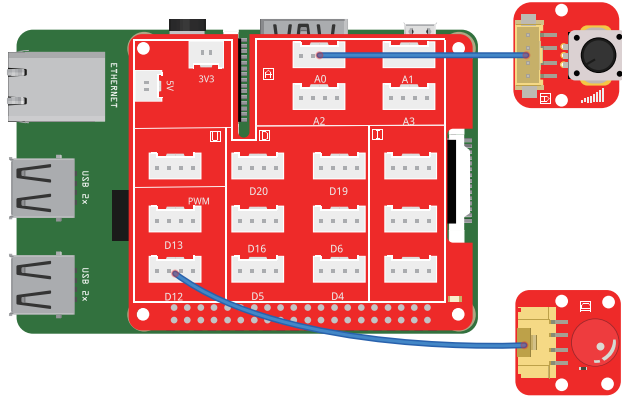
Crowtail - LED(Red) x1
Crowtail - Cable x2

Hardware learning and connection



This rotary angle sensor may also be known as a potentiometer twig that produces analog output between 0 and Vcc on its SIG pin. The angular range is 300 degrees with a linear change in value. The resistance value is 10k ohms, perfect for any platform use. Some applications like smart light control, volume control, only you can not think of things, no impossible things!

Connect Crowtail - Rotary Angle Sensor and Crowtail - LED(Red) to A0 and D12 ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



What is PWM

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

50% duty cycle



75% duty cycle



25% duty cycle



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

Programming and note

- **Pwm instance:** `pwm = GPIO.PWM(len_pin,80)`

Create a PWM instance object so that we can use this to control the PWM.

There are two parameters of this function, the first one is the pin you want to use PWM to control, and the second one is frequency.

- **Function: `brightness_func()`**

Create this function to change the analog value to a PWM duty cycle.

Because the analog value can be 0-1023, and the LED brightness has 256 levels (0-255), but the duty cycle can only be 0% -100%, so we use this function to change the analog value to PWM duty ratio.

- **Start PWM: `pwm.start(0)`**

Start the PWM. After we set the PWM, we need to turn it on.

Here we set the PWM at startup to 0 (the PWM interval can only be 0-100)

- **Duty cycle: `pwm.ChangeDutyCycle(brightness)`**

Change the duty cycle of the PWM. The duty cycle is in the range of 0% -100%.

The larger the duty cycle, the larger the current will be, and the brighter the LED will be.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 11_dimmer.py
```

Result

Turn the knob of the rotary angle sensor and you will find that the brightness of the LED will also change. If the LED becomes lighter when you rotate the rotary angle sensor clockwise; then, when you rotate the rotary angle sensor counterclockwise, the LED will become darker.

Thinking

Do you know how the volume is adjusted by the speaker?

Lesson 12 – Variable speed fan

Instruction

Here comes the hot summer! If you have a small personal fan, especially in the hot outdoors, it will be a good choice! Let's start making a small Raspberry Pi variable speed fan!

Target

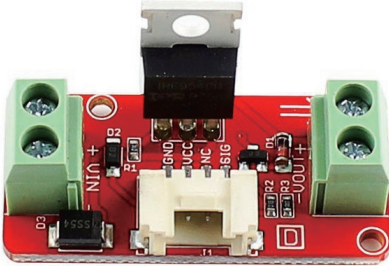
- Learn how the MOSFET work and use it to make a small fan with a micro-speed motor and rotary angle sensor.

Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - MOSFET x1
Crowtail - Rotary Angle Sensor x1
Crowtail - Cable x2

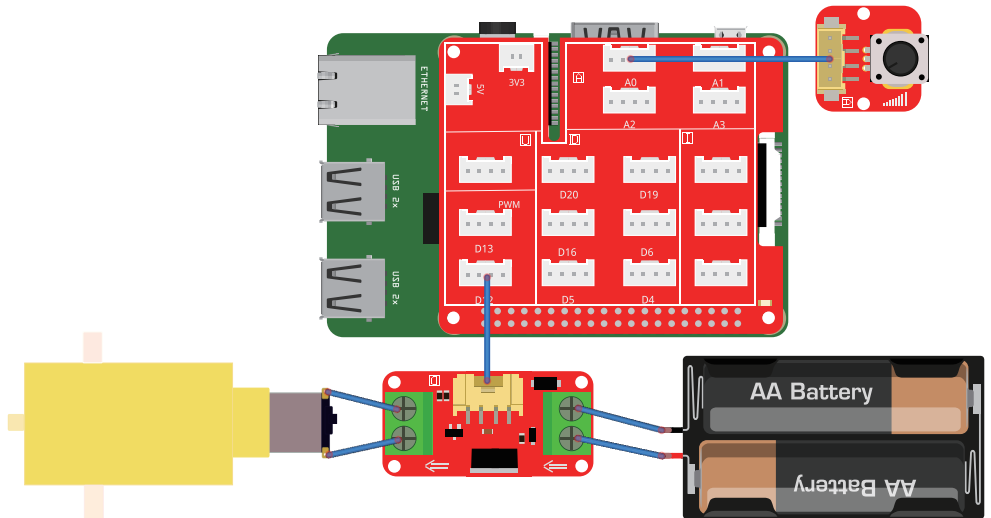
Micro-speed motor x1
Battery case x1
Jumper Wire x3

Hardware learning and connection



Crowtail - MOSFET enables you to control high voltage items (such as 50VDC) and low voltages (such as 5V) on a microcontroller. A MOSFET is also a switch. There are two screw terminals on the board. One for input power and the other for the device you want to control. Crowtail - MOSFET transfers power from one end to the other when closed. However, if there is no external power source, your device can still get power from the microcontroller through the Crowtail interface.

Connect Crowtail - Rotary Angle Sensor and Crowtail - MOSFET to A0 and PWM(D12) ports of Crowtail - Base Shield for Raspberry Pi; connect the battery case VCC(red wire) to VIN + and GND(black wire) to VIN - of the Crowtail - MOSFET; connect the two wires of the micro-speed motor to VOUT + and VOUT - of the Crowtail - MOSFET. plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Code overview

1. Import all the libraries needed.
2. Set GPIO mode, define variables to store pins and set up.
3. Open the SPI bus, create a PWM instance and start it.
4. Create functions to read SPI data and change analog data into a duty cycle.
5. Calling the function to get the analog value and store in variable “value”.
6. Calling the function to change “value” into a duty cycle.
7. Change the duty cycle of the PWM to control the speed of the fan.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 12_speed_fan.py
```

Result

When you turn the knob on the rotary angle sensor, you will notice that the speed of the micro-speed motor will also change. When you turn the rotary angle sensor clockwise, the speed of the motor increases; then when you turn the rotary angle sensor counterclockwise, the speed of the motor will decrease.

Note: Because the power supply we have is a 3V battery case, the driving ability is weak, so there is a special method to start the motor, that is, to quickly turn the angle sensor to the maximum(Turn clockwise), the maximum PWM (100), and the motor speed to the fastest to drive the motor. Then you can adjust the speed of the motor to what you want. If you want to see the motor speed change more obviously, you can try to connect the 5V power supply, but be careful not to connect the larger power supply, otherwise it may be dangerous!

Thinking

This fan is very cool, isn't it? But can we make a wireless remote control fan so that we can remotely control the fan?

Lesson 13 – Servo control

Instruction

Do you know how the robot turns its head and uses its arms to perform difficult gestures, such as grabbing things, dancing, etc.? The secret is the servo. Let's take a look at the servo and see what we will do with this servo!

Target

- Learn how the servo works and control its rotation.
- Learn the for loops.
- Learn the range() function.

Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - 9G Servo x1

Crowtail - Cable x1

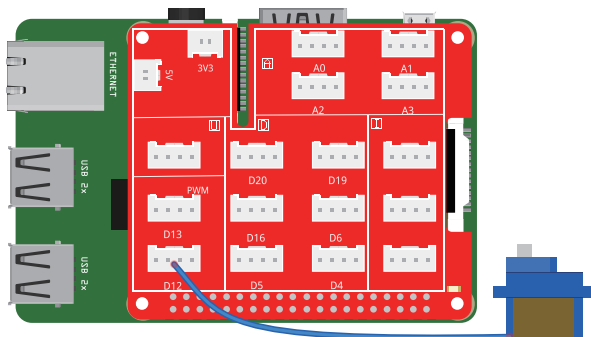
Hardware learning and connection

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 automation field, robot's head and hand.

Connect Crowtail - 9G Servo to PWM(D12) port of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

• Global variables: global pwm

It is a global variable identifier that defines the variable as a global variable. It means you can change the value of a variable within a function.

• Functions: def setup() def setDirection(direction)

We create two functions here, the setup () function is used to set the PWM of the servo pin; the setDirection () function is used to rotate the servo to an angle based on the duty cycle. Note that the a and b variables reflect the relationship between the duty cycle and the rotation angle. They must match the type of servo you are using. Therefore, a and b here are based on the SG90 servo we use.

• For loops: for direction in range(0, 181, 10)

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). With the for loop we can execute a set of statements, once for each item in a list, tuple, set, etc.

• The range() function: for direction in range(0, 181, 10)

To loop through a set of code a specified number of times, we can use the range() function. The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number. In this case, we loop the value of direction from 0 to 180(181 is not included), with each increment of 90, such as 0 to 90, 90 to 180.

Code overview

1. Import the GPIO and time libraries.
2. Set up the variables for servo pin and PWM frequency.
3. Set a and b variables which reflect the relationship between the duty cycle and the rotation angle.
4. Define two functions to set up PWM for servo and rotate the servo to an angle based on the duty cycle.
5. Print start prompt and calling two functions to rotate the servo.
6. Make the servo rotate to 0 degrees and print the end promptly.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 13_servo_control.py
```

Result

You will see the screen will print the start promptly and then the servo will slowly rotate from 0 degrees to 180 degrees. Finally, the servo rotates back to 0 degrees quickly and stops and prints the end promptly.

Thinking

How many servos the robot arm needs at least to control grasping object? How do multiple servos work together?

Lesson 14 – Obstacle alert

Instruction

Have you found your answer about the thinking question of lesson 8? How can the alarm remind us before the car hit an obstacle? Maybe this lesson will give you an answer.

Target

- Learn how the ultrasonic ranging sensor works and use it to make an obstacle alarm with buzzer.

Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - Ultrasonic Ranging Sensor x1

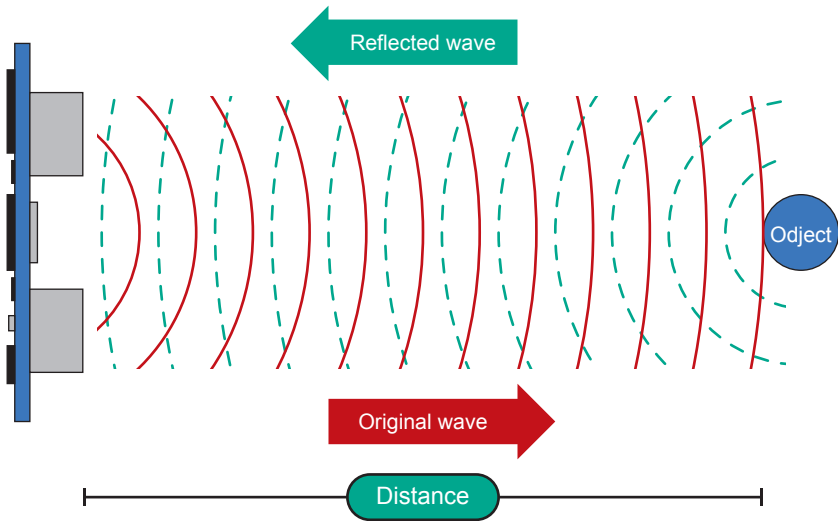
Crowtail - Buzzer x1
Crowtail - Cable x2

Hardware learning and connection

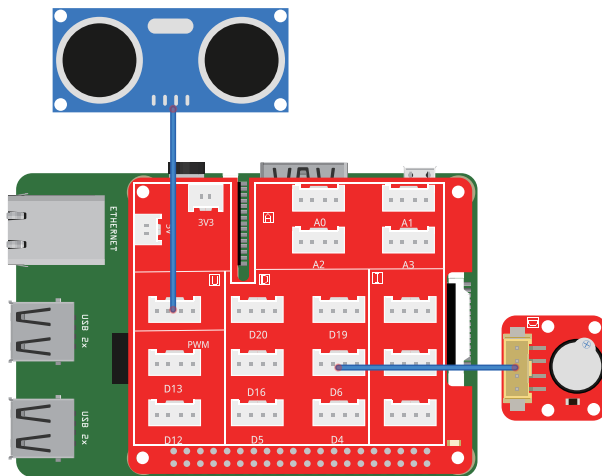
This HC-SR04 has stable performance and high ranging accuracy.

The process of ultrasonic ranging: 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.





Connect Crowtail - Ultrasonic Ranging Sensor and Crowtail - Buzzer to U port and D6 port of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

- **Time time function: `time.time()`**

When we use this `time.time()` function, it will return a merge of the current time(floating point seconds since the 1970s). For example, if a pin goes from low to high, and we're recording the low condition using the `time.time()` function, the recorded timestamp will be the latest time at which that pin was low. Once a signal is received by the echo pin, the value changes from low(0) to high(1). So, here we use this function to record the last low timestamp for echo pin(`pulse_start`) and last high timestamp for echo(`pulse_end`).

• **Distance: distance = pulse_duration * 17150**

The speed of sound is variable, depending on what medium it's traveling through, here we will take the speed of sound in air as an example of 343m / s. But the distance we output is in cm, so we change it to cm In terms of units, it should be 34300cm / s. Then we can know how to calculate the distance through the evolution of the formula.

$$34300 = \frac{\text{Distance}}{\text{Time}/2}$$

$$17150 = \frac{\text{Distance}}{\text{Time}}$$

$$17150 \times \text{Time} = \text{Distance}$$

• **Round function: distance = round(distance, 2)**

The round () function returns the rounded value of the floating point number x. This function takes two parameters, the first is a floating point number to be rounded, and the second is the number of digits to be rounded to the decimal point.

Code overview

1. Import the GPIO and time libraries.
2. Set mode for GPIO and initialize pin
3. Initialize the trig pin and sent pulse signals.
4. Set an echo pin to listen to the pulse signal we've sent and calculated the time.
5. Use the formula to calculate the distance.
6. Print the distance.
7. If the distance less than 20 cm, make the buzzer work.
8. Stop the buzzer if the distance is equal to or greater than 20 cm.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples  
sudo python 14_obstacle_alert.py
```

Result

Point the ultrasonic ranging sensor at an object. You can see the distance measured by the ultrasonic ranging sensor is printed ever 0.5 seconds.

Change the distance between ultrasonic ranging sensor and object, if the distance less than 20 cm, the buzzer will make a noise, otherwise, the buzzer will stop working.

Thinking

Try to make a smart door with an ultrasonic ranging sensor and servo, and let the door open automatically when someone is in front of the door.

Lesson 15 – Smart lantern

Instruction

I've been to such an amusement park before, the pirate ship only lights up and swings when someone is nearby, which is really smart, energy-efficient and cool! So I got an idea, let's try to make such a smart lantern in this lesson.

Target

- Learn how the RGB work and use it to make a smart lantern with a PIR motion sensor.
- Learn how to use the pip tool to import libraries.
- Learn how to create and use lists.

Required materials

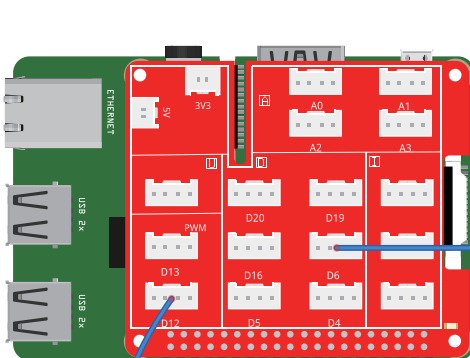
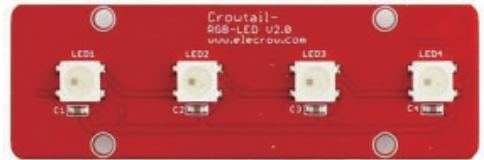
Crowtail - Base Shield for Raspberry Pi x1
Crowtail - RGB-LED x1

Crowtail - PIR Motion Sensor x1
Crowtail - Cable x2

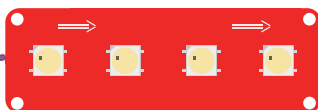
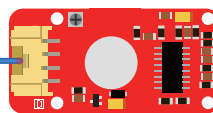
Hardware learning and connection

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.



Connect Crowtail - RGB-LED and Crowtail - PIR Motion Sensor to PWM(D12) and D6 ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

- **Ws281x library: `from rpi_ws281x import *`**

This is the official Python distribution of the ws281x library. The library provides many functions so we can control the RGB LED module. We can simply use the following steps to install this library:

- **Step 1:** Make sure you have installed pip tool(make sure you have connected network, if it is already installed, you can skip this step):

`sudo apt-get install python-pip`

- **Step 2:** install the ws281x from pip:

`sudo pip install rpi_ws281x`

- **LED strip configuration:**

We need to configure the LED strip and then pass them into the NeoPixel instance object, so we can control the RGB light correctly.

- **List: `colors = [Color(220,20,60),Color(178,34,34),Color(255,0,0),Color(139,0,0)]`**

A list is a collection that is ordered and changeable. In Python, lists are written with square brackets. In this case, we create a list named colors, which include 4 colors created by the Color() function.

- **Strip number: `strip.numPixels()`**

This function is used to get the number of lights of the RGB LED module. It will return the value of LED_COUNT that we configured.

- **Set color: `strip.setPixelColor(i,color)`**

Set the color of the i-th LED to "color" value. Note that the number of the first LED is 0.

- **Update display: `strip.show()`**

This function is to update the display with the data from the LED buffer. Every time we update the LED color settings, we need to call this function to update the display.

Code overview

1. Import GPIO, time and RGB LED libraries.
2. Set mode for GPIO and set up pin for the PIR motion sensor.
3. Configure the RGB LED strip.
4. Create a NeoPixel instance object with the appropriate configuration.
5. Create a list to store the value of colors we set.
6. Create two functions to light up and wipe the RGB LED.
7. If the state of the PIR motion sensor is 1, calling the function to display the RGB LED.
8. Use try and except to catch and handle the error.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 15_smart_lantern.py
```

Result

Wave your hand or walk around within the detection range of the PIR motion sensor, you will see 4 lights on the RGB LED light up and change 4 colors in turn. If you stop your movement, the RGB LED will lights off.

Thinking

When there are enough RGB LEDs, can we make it show a pattern?

Lesson 16 – Distance display

Instruction

Seeing is believing, when the car is approaching an obstacle, only the danger reminder is still a little worrying. If we can add a display to show the distance of the obstacle, it is definitely a very good choice! Presumably, you know what we need to learn in this lesson, let's get started.

Target

- Learn how the LCD works and use it to make a distance display with an ultrasonic ranging sensor.
- Learn how to use I2C modules.
- Learn how to display messages on the LCD and control the backlight.

Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - Ultrasonic Ranging Sensor x1

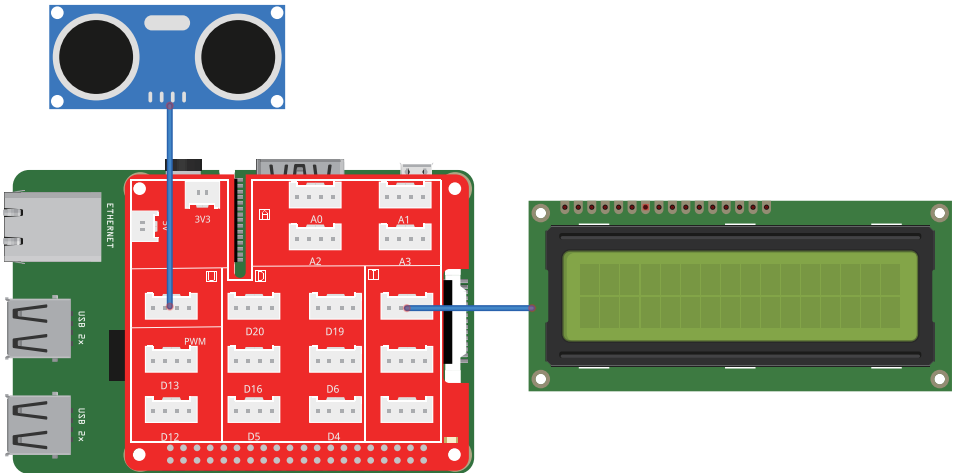
Crowtail - I2C LCD x1
Crowtail - Cable x2

Hardware learning and connection

Crowtail - I2C LCD includes LCD1602 and MCP23008 modules. Unlike ordinary LCDs, which require many pins, the Crowtail - I2C LCD only needs 4 pins to control 16x2 and the backlight. It's very popular where the display is needed, you can use this LCD to display numbers, string, symbol and so on.



Connect Crowtail - Ultrasonic Ranging Sensor and Crowtail - I2C LCD to U and I ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

- **LCD library: import Adafruit_CharLCD as LCD**

This library is for accessing LCD from a Raspberry Pi or BeagleBone Black. You have to install the following dependencies(make sure you have a connected network).

```
sudo apt-get install python-smbus
```

Then, follow the commands below to install the library.

```
cd starter_kit_for_pi/drivers/Adafruit_Python_CharLCD
```

```
sudo python setup.py install
```

- **Set LCD: lcd_columns = 16 lcd_rows = 2**

Defines the LCD column and row size for a 16x2 LCD. Our LCD is 16x2 characters, and we need to set it correctly according to the LCD used.

- **LCD instance: lcd = LCD.Adafruit_CharLCDBackpack(address = 0x20)**

Create an LCD instance and pass the I2C address for it. You can use the following command to check the I2C address for I2C devices.

```
sudo i2cdetect -y 1
```

• LCD Backlight: `lcd.set_backlight(0)` `lcd.set_backlight(1)`

Function to control the backlight of LCD. Note that 0 represents turn on the backlight while 1 represents to turn off.

Code overview

1. Import all the libraries we need.
2. Create a function to detect the distance and return it.
3. Define the LCD column and row size.
4. Initialize the I2C address for LCD.
5. Turn backlight on.
6. Display the distance message on LCD.
7. If a user interrupts occur, clean LCD and turns the backlight off.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 16_distance_display.py
```

Result

You will see the distance detected by the ultrasonic ranging sensor is displayed on the LCD, and the distance is updated every 2 seconds.

Thinking

Try to apply this project to obstacle avoidance robots. In addition, when an obstacle is detected, how to turn the robot in a direction without obstacles.

Lesson 17 – Weather station

Instruction

The weather changes too frequently. Yesterday was a hot day. Today may be a chilly day. Changeable weather is the most likely to cause illness. We must always pay attention to changes in the weather, carried with umbrellas and so on according to the weather.

Well, let's make a weather system in this course so that we can know the weather temperature and other information at any time.

Target

- Learn how the dht11 module work and use it to make a weather display station with I2C LCD and LED.
- Learn how to read humidity and temperature information form the dht11 sensor.

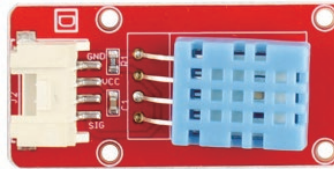
Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - Temperature&Humidity Sensor x1
Crowtail - I2C LCD x1

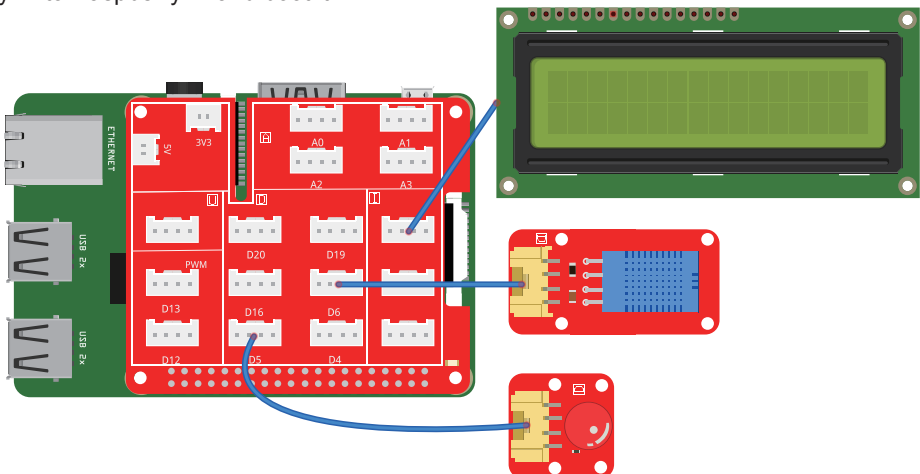
Crowtail - LED(Red) x1
Crowtail - Cable x3

Hardware learning and connection

This module can help you detect the temperature and humidity of the environment of your house. The module contains a DHT11 temperature & humidity sensor that is a complex sensor with a calibrated digital signal out. It uses digital module acquisition technology and temperature& humidity sensor technology. The sensor consists of a resistance type moisture element and an NTC temperature measuring element. Because of the single-wire serial interface, it is easy to use the module.



Connect Crowtail - Temperature&Humidity Sensor, Crowtail - LED and Crowtail - I2C LCD to D5, D6 and I ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

• DHT library: `import Adafruit_DHT`

This python library is read the DHT series of humidity and temperature sensors on a Raspberry Pi or BeagleBone Black.

We can use the following commands to install the library.

```
cd starter_kit_for_pi/drivers/Adafruit_DHT  
sudo python setup.py install
```

• Read data: `Adafruit_DHT.read_retry(sensor_type, sensor_pin)`

Reads the DHT sensor of the specified sensor type (DHT11, DHT22 or AM2302) on the specified pin and returns a tuple of humidity (floating point value expressed as a percentage) and temperature (floating point value expressed in degrees Celsius). We need to pass two parameters, the first is the sensor type, for example, our sensor is DHT11, we only need to pass 11. The second is the sensor pin we connected. This function will return two floating-point values and we could store them in humidity and temperature these two variables.

Code overview

1. Import DHT and other libraries we need.
2. Set mode for GPIO and create some variables to store pins for dht11 and led.
3. Set up LCD and turn the backlight on.
4. Read humidity and temperature from the dht11 sensor.
5. If data available from the sensor, print them on LCD
6. If the temperature is lower than 10 or more than 35 degrees in Celsius, turn LED on. Otherwise, turn LED off.
7. If a user interrupts occur, clean LCD, turn the backlight off and clean up the GPIO.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples  
sudo python 17_weather_station.py
```

Result

You will see the first column of the LCD shows the current temperature information and the second column shows the humidity information. When the temperature is less than 10 degrees Celsius or greater than 35 degrees Celsius, the LED will light up; otherwise, the LED will go out.

Thinking

If we want to build a more comprehensive weather station, such as obtaining air quality information, ultraviolet data, etc., how should we implement it on the basis of this project?

Lesson 18 – Remote control

Instruction

Nowadays, most home appliances in the home will have a remote control, which makes it easier and more convenient for us to control home appliances. Can Raspberry Pi have its own remote control? Let's try it out!

Target

- Learn how the IR receiver work and use the infrared remote control to communicate with it.

Required materials

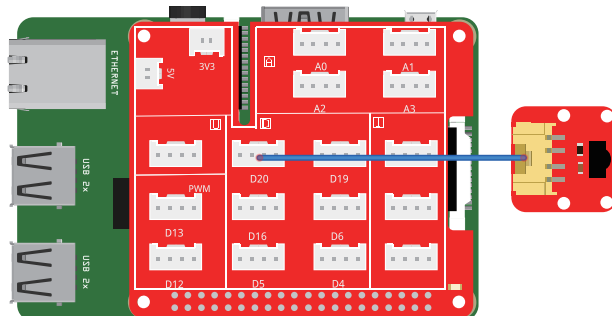
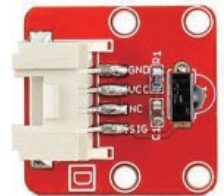
Crowtail - Base Shield for Raspberry Pi x1
Crowtail - IR Receiver x1

Infrared Remote Control x1
Crowtail - Cable x1

Hardware learning and connection

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, we can use our infrared remote control to control this module well. The Infrared Receiver can receive signals well within 10 meters. If more than 10 meters, the receiver may not get the signals.

Connect Crowtail - IR Receiver to D20 port of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

- **Install the LIRC library**

sudo apt-get install lirc

Make sure you have a connected network.

If it gives you errors, don't worry. It will be fixed later.

- **Set up lirc:**

We need to configure the LIRC so that we can use the GPIO of the Raspberry Pi for the IR receiver.

- **Step 1:** Inside starter kit for raspberry/drivers folder that we cloned, there are 3 files you need to copy the files to the LIRC configuration directory, from the starter kit for raspberry folder run the following commands:

sudo cp drivers/LIRC/* /etc/lirc

- **Step 2:** After you moved the configuration file, edit your boot config file with the command "sudo nano /boot/config.txt" and where it says(previous image)

Uncomment this to enable the lirc-rpi module

#dtoverlay=lirc-rpi

For the latest image, you should find where it says

Uncomment this to enable infrared communication

dtoverlay=gpio-ir,gpio_pin=17

dtoverlay=gpio-ir-tx,gpio_pin=18

Change it to this:

Uncomment this to enable the lirc-rpi module

dtoverlay=gpio-ir,gpio_pin=20

Press Ctrl +O and Ctrl +X, then enter to save and exit.

- **Step 3:** Execute the following commands to copy the configuration files

sudo cp /etc/lirc/lirc_options.conf.dist /etc/lirc/lirc_options.conf

sudo cp /etc/lirc/lircd.conf.dist /etc/lirc/lircd.conf.

- **Step 4:** Edit /etc/lirc/lirc_options.conf by writing the command "sudo nano /etc/lirc/lirc_options.conf" and modify the following lines to be exact as here:

driver = default

device = /dev/lirc0

Press Ctrl +O and Ctrl +X, then enter to save and exit.

- **Step 5:** Then add the two lines below to /etc/modules (use command "sudo nano /etc/modules"). This will start the modules up on boot. Pin 20 below will be used to take the output from the IR sensor.

lirc_dev

lirc_rpi gpio_in_pin=20

- **Step 6:** Now reboot the Raspberry Pi for the configuration to take effect

`sudo reboot`

Now run apt-get install lirc once again to fix the previous errors if any

`sudo apt-get install lirc`

- **Step 7:** The last step, stop the LIRC library to we could use the IR driver with our python script

`sudo /etc/init.d/lirc stop`

- **Note: if you get a runtime error that the command cannot be found, maybe you have a different version of LIRC, try this command instead:**

`sudo /etc/init.d/lircd stop`

- **A quick test**

To perform a quick test to see if LIRC is working, we need to stop the LIRC daemon and start mode2. mode2 shows the pulse/space length of infrared signals.

`mode2 -d /dev/lirc0`

Result

When buttons are pressed on your remote, the monitor will print the pulse/space length of infrared signals.

Thinking

How to use the remote control to control the Raspberry Pi and the hardware module connected to it?

Lesson 19 – Speed measurement

Instruction

Just like the thinking question of lesson 5, how can we use hall sensors in other applications? Well, this lesson will let you know how to measure the speed by using the hall sensor.

Target

- Try to make a speedometer with a hall sensor and display the speed on LCD.

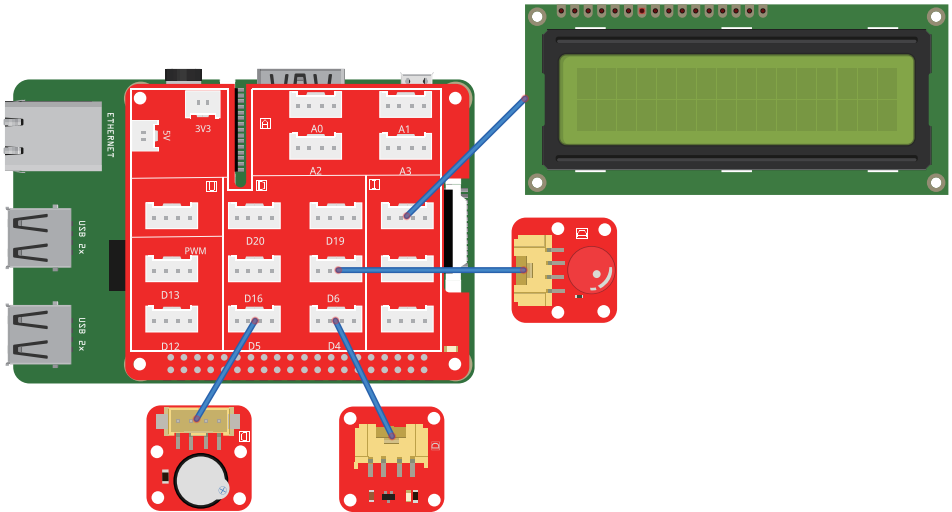
Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - Hall Sensor x1
Crowtail - I2C LCD x1

Crowtail - Buzzer x1
Crowtail - LED(Red) x1
Crowtail - Cable x4

Hardware learning and connection

Connect Crowtail - Hall Sensor, Crowtail - Buzzer and Crowtail - LED to D4, D5 and D6 ports of Crowtail - Base Shield for Raspberry Pi; connect Crowtail - I2C LCD to I port of Crowtail - Base Shield for Raspberry Pi; plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

- **Pull-up and pull-down:** `GPIO.setup(hall_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)`

Different from the previous GPIO configuration, we configure the hall sensor as a pull-up, which means that the input of this pin is high by default. Mainly for protection.

- **Interrupt function:** `GPIO.add_event_detect()`

This is an interrupt function, it can listen to a pin.

Once the pin input has changed, calling `eventdetected()` will return `True`. We can add a callback function into this interrupt function and ignore edge operations due to switching jitter. In this case, we use this interrupt function to detect whether the input of the Hall sensor has changed (`GPIO.LOW`). If so, call back the `hall_pulse()` function and add 50ms to prevent the sensor from shaking.

Code overview

1. Import all the libraries we need.
2. Set up the pin for hall sensor, buzzer and LED.
3. Set up the LCD and turn the backlight on.
4. Create a callback function to count the rpm.
5. Create an interrupt function to detect if the output of the hall sensor pin changes.
6. When the counting time reaches 10 seconds, rpm is displayed on the LCD.
7. If the rpm is less than or equal to 20, neither the buzzer nor the LED will work.
8. If the rpm is more than 20 and less than 30, the LED lights up and the buzzer does not work.
9. If the rpm is 30 or more, both LED and Buzzer work.
10. Then, start a new rpm count.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 19_speed_measurement.py
```

Result

Use the south pole of the magnet close to the hall sensor at different frequencies. The LCD will display the number of times the south pole of the magnet has approached the Hall sensor within 10 seconds. When the number is 20 or less, neither the LED nor the buzzer will work; When the number is more than 20 and less than 30, the LED lights up and the buzzer does not work; when the number is 30 or more, the LED and the buzzer will work.

Thinking

Now that it can be used to measure rpm, how do we get the speed or distance?

Lesson 20 – Plant cultivation expert

Instruction

There are many factors that can affect plant growth, such as soil moisture, light, weather, etc. You will find that focusing on only one aspect of plant growth cannot support plants. We need to take into account all the factors required for plant growth and create the most suitable environment for plants so that plants can grow healthy. Alright, let's be a planting expert now.

Target

- Try to make a plant growth monitor to monitor all the growth factors of the plants.

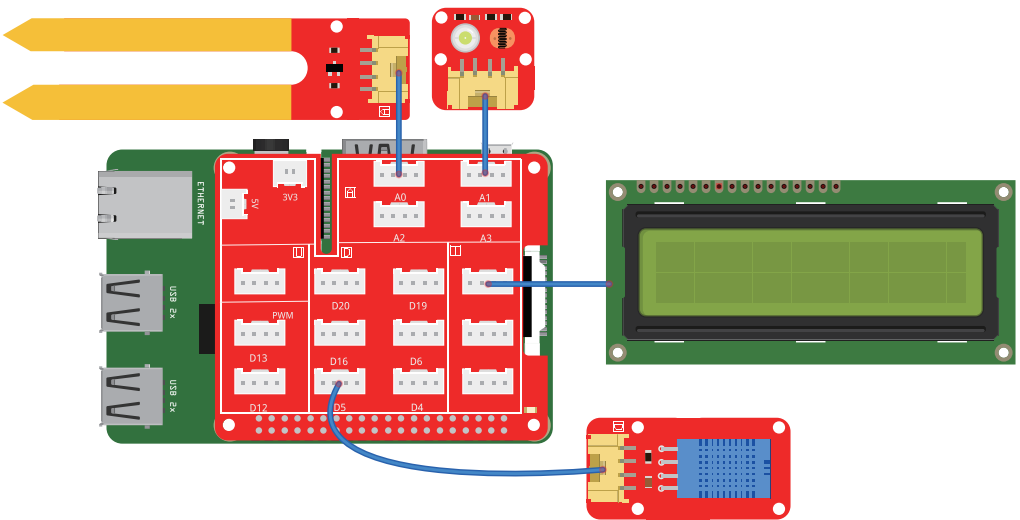
Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - Temperature&Humidity Sensor x1
Crowtail - Moisture Sensor x1

Crowtail - Light Sensor x1
Crowtail - I2C LCD x1
Crowtail - Cable x4

Hardware learning and connection

Connect Crowtail - Temperature&Humidity Sensor, Crowtail - Moisture Sensor and Crowtail - Light Sensor to D5, A0 and A1 ports of Crowtail - Base Shield for Raspberry Pi; connect Crowtail - I2C LCD to I port of Crowtail - Base Shield for Raspberry Pi; plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Code overview

1. Import all the libraries we need.
2. Set mode for GPIO and create variables to store the pins.
3. Set up LCD and turn the backlight on.
4. Initialize SPI and create a function to read the analog value from SPI.
5. Read the moisture value from moisture sensor and display on LCD.
6. Read the brightness value from the light sensor and display on LCD.
7. Read the temperature and humidity from sensor and display on LCD.

Execute the following commands and try it by yourself

```
cd RPi-Starter-Kit/examples
sudo python 20_plant_expert.py
```

Result

Plug the moisture sensor into the soil, you will see the LCD display the moisture value of the soil. About 2 seconds later, the LCD displays the brightness value. And then about 2 seconds later, the LCD will print the temperature and humidity information.

Thinking

After we get these data that can affect plant growth, how to use these data to build an automated system? For example, watering the plants automatically when the moisture is low.

Lesson 21 – Remote control door

Instruction

How do we solve this when someone is knocking on the door and we are too busy with other things? It would be great if we could use the remote control to open and close doors! Okay, so in this lesson, we're going to use the remote control to control the servo, just like opening or closing a door.

Note: The latest image of the Raspberry Pi has problems working with IR. We recommend that you download the image directly from our website to use IR.

Target

- Use the infrared remote control to control the servo.
- Learn the basic usage of the socket.

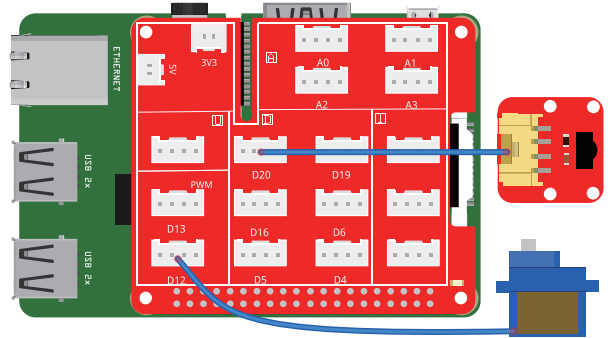
Required materials

Crowtail - Base Shield for Raspberry Pi x1
Crowtail - IR Receiver x1
Crowtail - 9G Servo x1

Crowtail - Cable x1
Infrared Remote Control x1

Hardware learning and connection

Connect Crowtail - IR Receiver and Crowtail - 9G Servo to D20 and PWM(D12) ports of Crowtail - Base Shield for Raspberry Pi, plug Crowtail - Base Shield for Raspberry Pi to Raspberry Pi and boot it.



Programming and note

• A quick test

Just like lesson 18, in order to use the IR receiver, we need to set up IR receiver. You can follow skip to lesson 18 if you don't how to set up.

• Install python-lirc:

Python bindings for LIRC, it allows us to use python to access IR. We can use the following command to install this(make sure you have a connected network).

```
sudo apt-get install liblircclient-dev  
sudo pip install python-lirc
```

• Socket library: import socket, signal

We will use the raspberry pi as a server here. So we need to import the socket library to use the infrared remote control to control the raspberry pi. Signal package is responsible for processing signals inside python programs, that is once the IR receiver receives the specific key, the program can interrupt the original program execution flow to process the signal.

• Socket setup:

Set up Raspberry Pi as a server.

We need to set up HOST, PORT and define socket type(network communication).

• Close function: def handler()

Create a function to close the socket after the transfer is complete and clean up the output of GPIO.

• Read key: output = lirc.nextcode()[0]

Read the key pressed on the infrared remote control. If the key is "-"(volume down) make the servo rotate to 180 degrees. If the key is "+"(volume up) make the servo rotate to 0 degrees.

Code overview

1. Import all the libraries we need.
2. Define some variables for using the servo.
3. Create functions to set up and rotate the servo.
4. Set up socket for raspberry pi.
5. Create functions to close the socket and send commands.
6. Read the command pressed on the infrared remote control.
7. If the key is "-"(volume down) make the servo rotate to 180 degrees.
8. If the key is "+"(volume up) make the servo rotate to 0 degrees.

Execute the following commands and try it by yourself

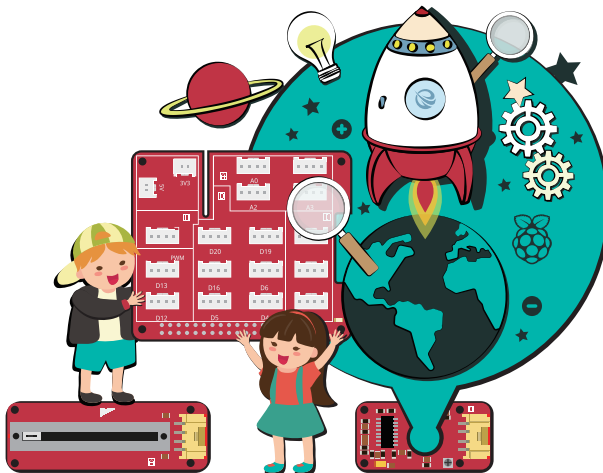
```
cd RPi-Starter-Kit/examples  
sudo python 21_remote_door.py
```

Result

When you press the "+" key on the infrared remote control, the servo shaft will rotate to 180 degrees. When you press the "-" key on the infrared remote control, the servo shaft will rotate to 0 degrees.

Thinking

How to add more hardware modules to build your own remote control home? Such as control LEDs, servo, LCD, buzzer and so on.





MAKE YOUR MAKING EASIER



info@elecrown.com



+86 0755-23204330



www.elecrown.com



5F, Fengze Building B,
Nanchang Huafeng Industrial Park,
Bao'an District, Shenzhen, China.

