CrowPi
by Elecrow

CrowPi Lessons with Python 2 and Python 3

Quick Start Guide(V2.1 2019.11)

CrowPi Lessons with Python 2 and Python 3

- Lesson 1 - Using the Buzzer for alert sound or notification.
- Lesson 2 - Get input from button to control the Buzzer.
- Lesson 3 - How Relay works and how to control it.
- Lesson 4 - Send vibration signal to the vibration sensor.
- Lesson 5 - Detect sound using the sound sensor.
- Lesson 6 - Detect low or bright light using the Light sensor.
- Lesson 7 - Detect room temperature and humidity using the DHT11sensor.
- Lesson 8 - Detect motion using the motion sensor.
- Lesson 9 - Getting distance information using the Ultrasonic sensor.
- Lesson 10 - Controlling the LCD Display.
- Lesson 11 - Read / Write RFID card using the RFID module.
- Lesson 12 - Using the step motor and making step movements.
- Lesson 13 - Controlling servos motors using the servo interfaces.
- Lesson 14 - Controlling the 8x8 Matrix LED.
- Lesson 15 - Controlling the 7 Segment Display.
- Lesson 16 - Detecting touch using the Touch Sensor.
- Lesson 17 - Detecting tilt using the Tilt Sensor.
- Lesson 18 - Using and controlling the Button Matrix.
- Lesson 19 - Using and controlling the IR sensor
- Lesson 20 - Making your own circuit board using the Bread Board
- Lesson 21- Using the CrowPi Camera

Introduction

Thank you for backing CrowPi on Kickstarter and supporting our continues work on making everyone's making easier.

In the following file you'll find 20 lessons to get your hands on over the CrowPi.

The lessons are variant from complete beginner to more advanced, no matter what's your level of understanding we can promise you'll enjoy it.

During the lessons, the examples requires to download a file and execute it on the CrowPi, for that reason we've made 2 brief explanations on basic python and linux usage.

Both short tutorials explain how to download scripts from GitHub(code sharing platform where we keep our codes and scripts) and also executing the python scripts to be able to run the samples with ease.

The lessons begin with explaining briefly about the Raspberry Pi and it's GPIO methods, afterwards during the tutorials we'll use GPIO.IN and GPIO.OUT to control our sensors.

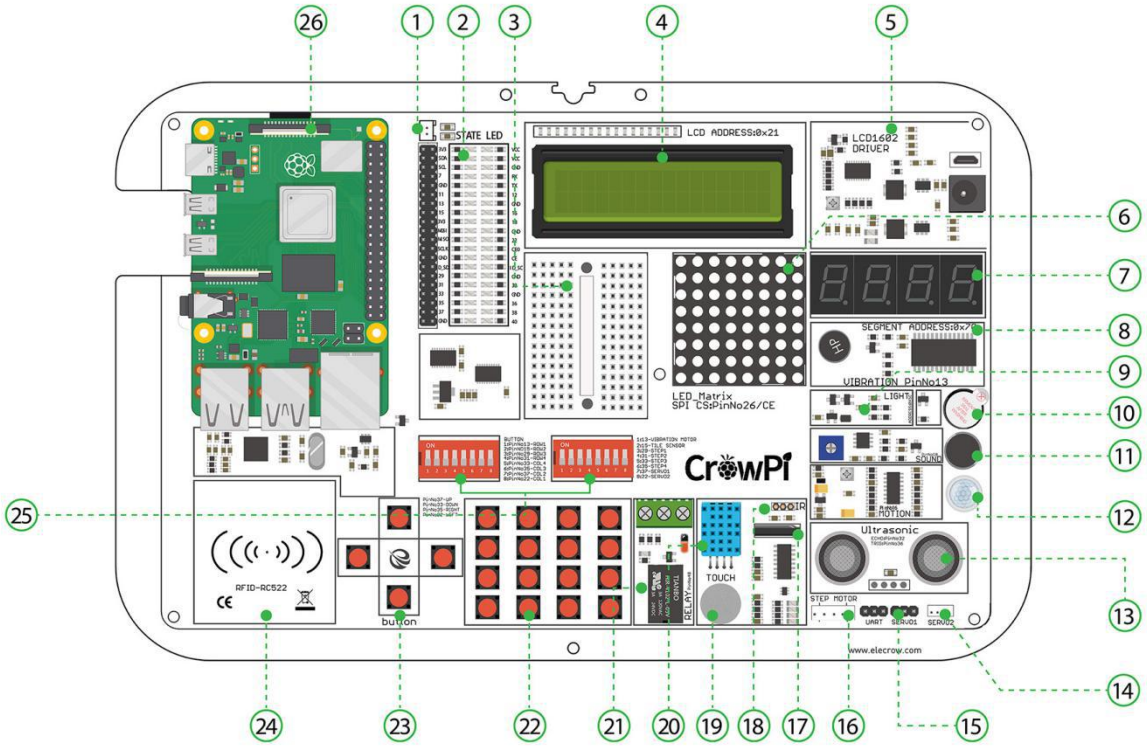
Some sensors are controllable by ports called SPi, we'll also explain how to use those in order to execute our scripts.

This tutorials are just the beginning.

We are looking forward to see what you will make with our CrowPi.

Sensors & modules list

The sensors below are listed to match the numbers on the CrowPi board:

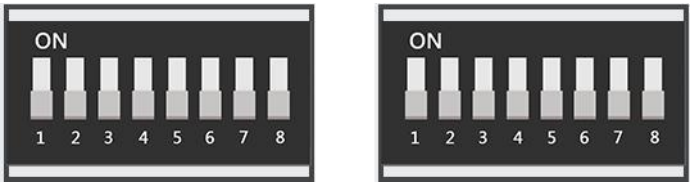


- 1 - 2-pin fan interface, the upper pin is "+" and the lower pin is "-"
- 2 - GPIO LED Indicator
- 3 - Breadboard - Used to make custom circuits using outside modules which the CrowPi doesn't include
- 4 - LCD Module - Used to show multi line information and text
- 5 - Power Circuit
- 6 - Matrix LED - Used to show text and other sorts of data
- 7 - Segment LED - Used to show numbers and data
- 8 - Vibration module - Used to make a strong vibration throughout the CrowPi board
- 9 - Light sensor - Used to detect and measure the strength of the light in the room
- 10 - Buzzer - Used to make a really loud buzzing alarm!
- 11 - Sound Sensor - Used to detect loud noise in the room

* 12 - Motion sensor - Used to detect motion in the room and around the CrowPi
* 13 - Ultrasonic sensor - Used to measure distance between the sensor and an object on top of it!
* 14,15 - Servos interface-2 interfaces to connect servos and be able to rotate them in any direction you want!
* 16 - Step motor - Used to control robots and machines in the industry, learn how to make step movements using this module!
* 17 - Tilt Sensor - Used to detect a tilt in the CrowPi; whenever you tilt it to the right or to the left, you'll know!
* 18 - IR Receiver interface - Plug the IR receiver into it. Used to receive IR signals.
* 19 - Touch Sensor - Works like a button but instead of clicking it detects touch!
* 20 - DH11 Sensor - Used to check and measure the humidity and the temperature in the room!
* 21 - Relay - Used to open and close electronic circuits to control electronics such as LED's!
* 22 - Matrix Buttons - Can be used as a keypad or as multiple options buttons!
* 23 - Independent buttons - Can be used to play games or control a robot!
* 24 - RFID Module - Used to Read and Write data over RFID/NFC cards!
* 25 - Switches - Used to switch between the sensors and modules as the Raspberry Pi doesn't contain enough GPIO pins
* 26 - Raspberry Pi

Switching between the modules

Understanding how to switch between the CrowPi sensors



The CrowPi board contains 2 “switches”, and each switch contains 8 pins, total of 16 pins.

The switches enable us to change between usage of sensors and modules, as the Raspberry Pi supports only limited number of GPIO pins, but using those switches we can extend our capability to much more.

Using those switches is pretty easy and will be required in some of the lessons

above. The following sensors require using switches and changing them

accordingly:

* **Button array (the whole left switch is dedicated to the button array & independent button)**
* **Independent button (the whole left switch is dedicated to the button array & independent button)**
* **Vibration sensor (the right switch, first pin)**
* **Tilt sensor (the right switch, second pin)**
* **Step motor (the right switch, pins number: 3,4,5,6)**
* **Servos (the right switch, number 7,8)**

When you need to use one of those sensors, you’ll need to switch to using the switch according to the right number. After the class or if you aren't using that sensor it’s good practice to turn off the switch as some other sensors use those GPIO pins as well.

If you don’t turn off unused pins and keep the switch on, that might cause conflict between sensors and the CrowPi might not work the way it should.

If you encounter any issue in the tutorials - make sure that the switch is turned on or off accordingly.

Basic Python and Linux usage

Cloning the git repository

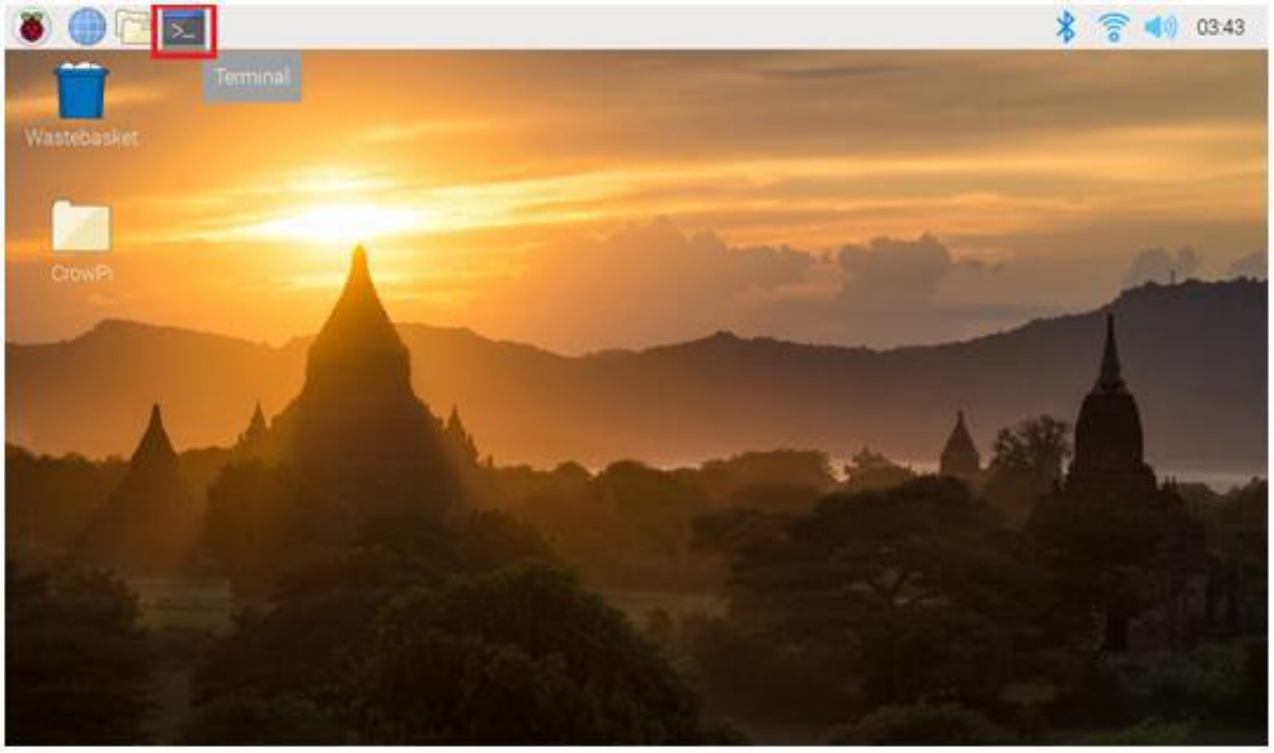
This step is optional but makes it easier to execute scripts without the need to download each one separately.

All the scripts already exist on the desktop inside the "CrowPi" Folder, if you want to update the folder for any future changes, you can follow the following steps.

This incredibly useful way called "git cloning".

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 are we gonna do that? 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

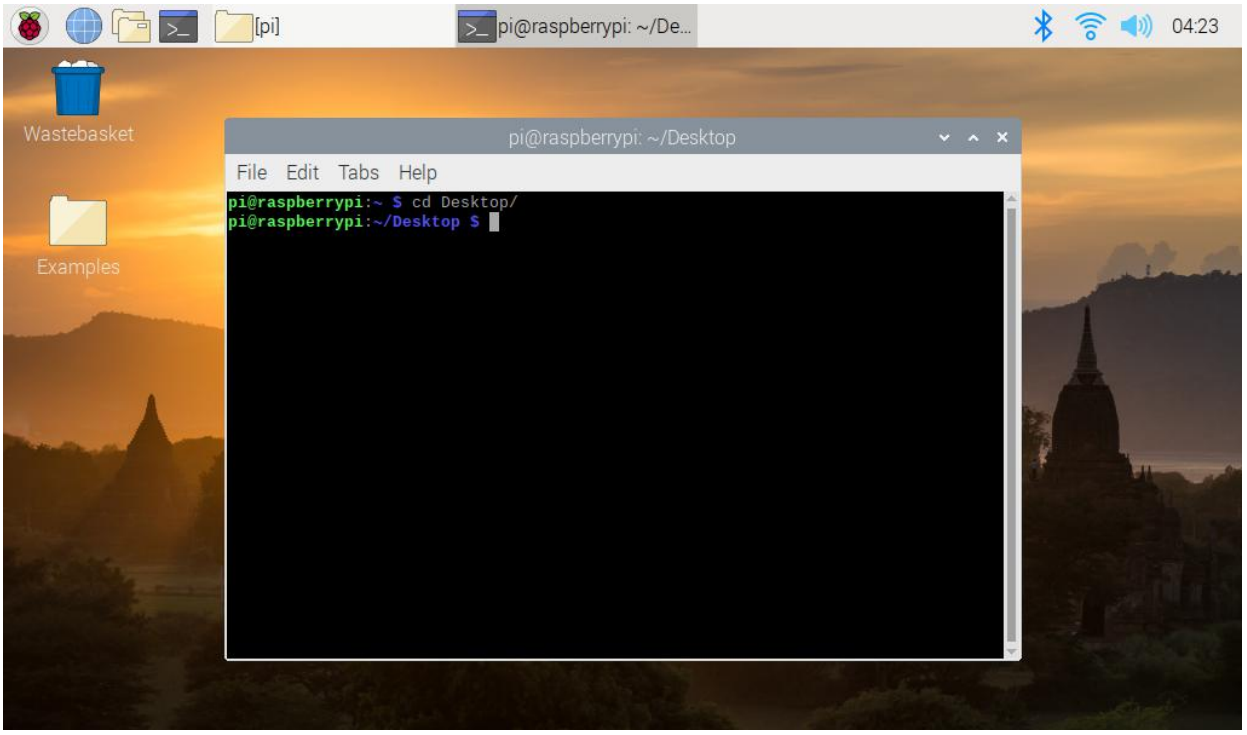


2. After opening terminal successfully we'll need to clone the scripts directory into our desktop.

```
cd Desktop
```

To do so you need to type in the terminal:

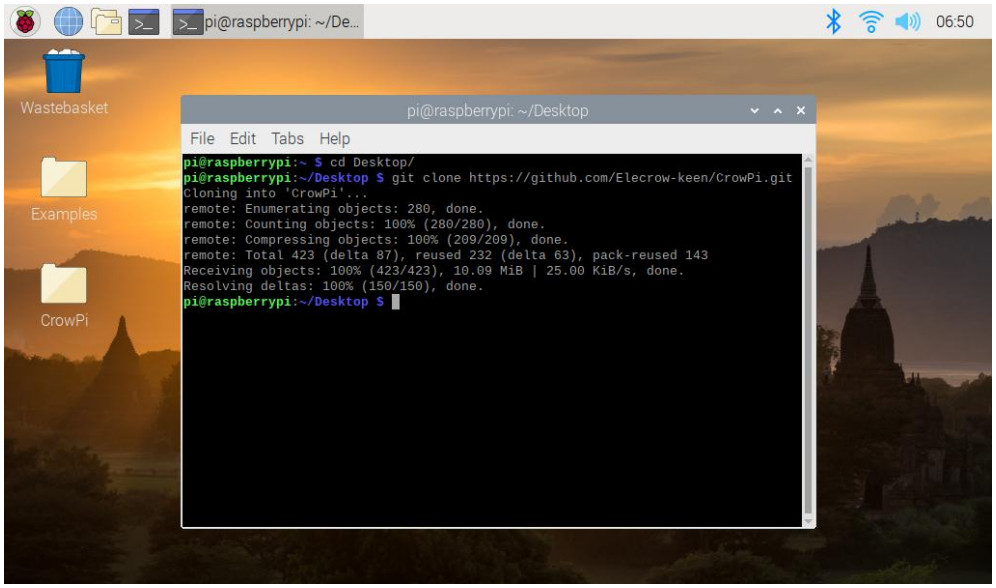
3. Press "Enter" on your keyboard. Now you should be in your desktop folder which is your desktop that you see.



4. Inside the terminal type the following command:

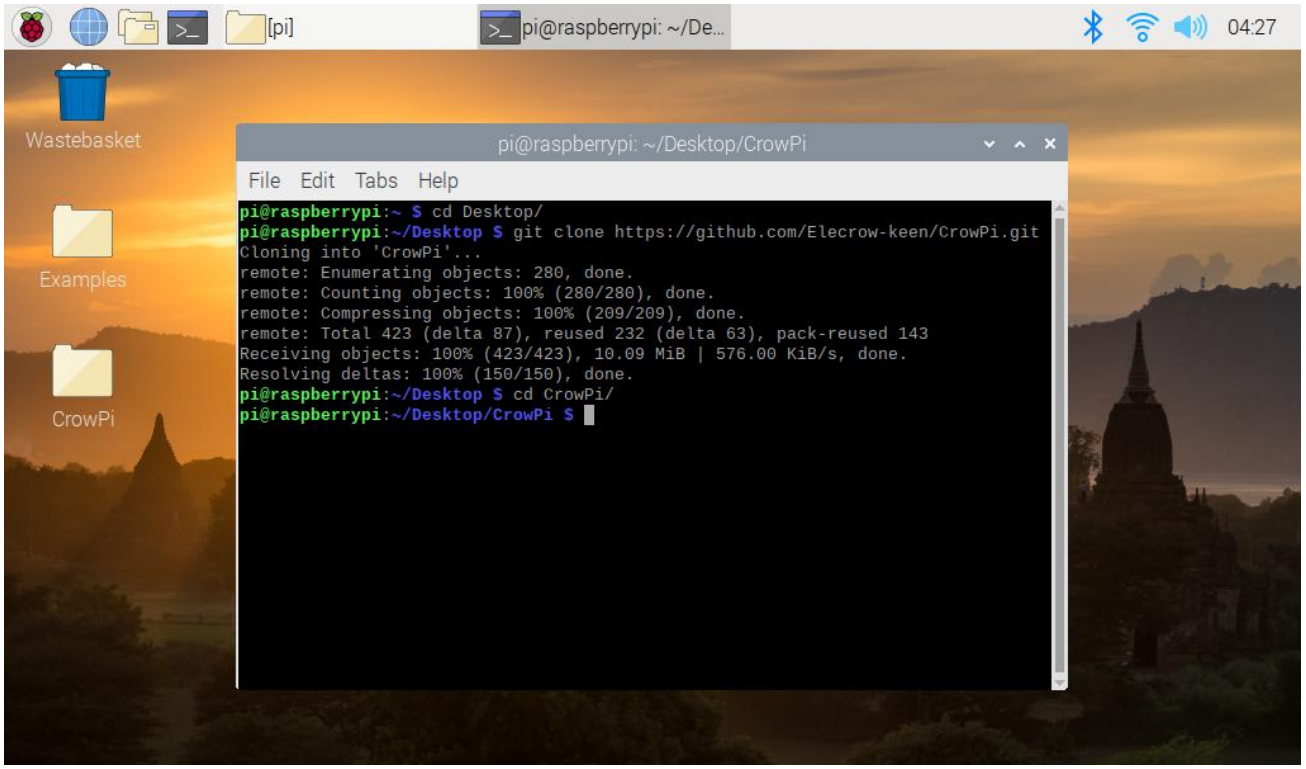
```
git clone https://github.com/Elecrow-keen/CrowPi.git
```

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



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

```
1   cd CrowPi
```



Now, during the tutorials all you need to do is execute the script. Learn how to execute the scripts on the next page!

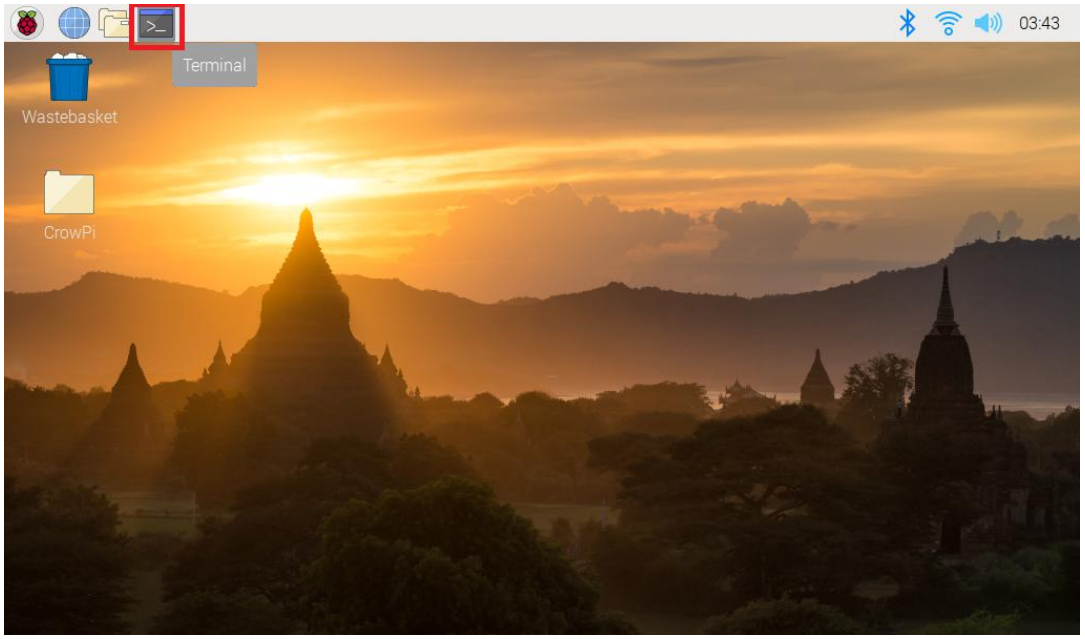
*** Note: every time you turn off your CrowPi you’ll need to repeat the steps of going into the folder using the “cd” command. BUT you don’t need to repeat the step of cloning or downloading the scripts from GitHub as they are already there on your desktop!**

Basic Python and Linux usage

Executing Python scripts

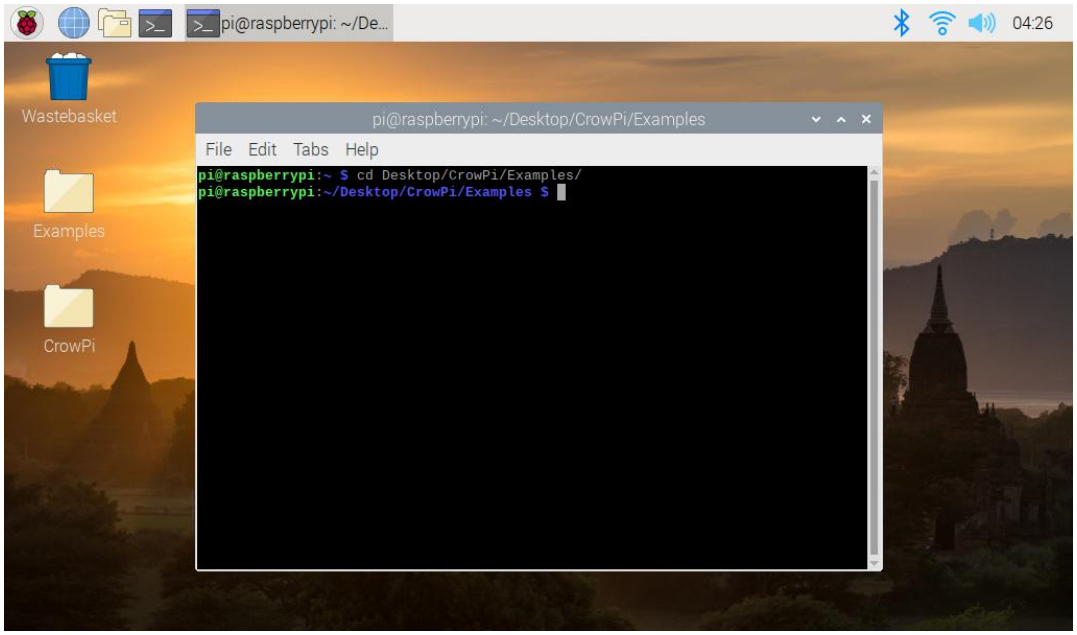
After we've successfully downloaded our script from GitHub, we might want to execute it now. In order to execute the script, we'll need to run the "python" command inside the terminal. Follow the following steps in order to get the script running through the terminal:

1) Open "terminal". It looks like a black screen and we will use it to execute most of our python script sand to download extension sand scripts from GitHub.



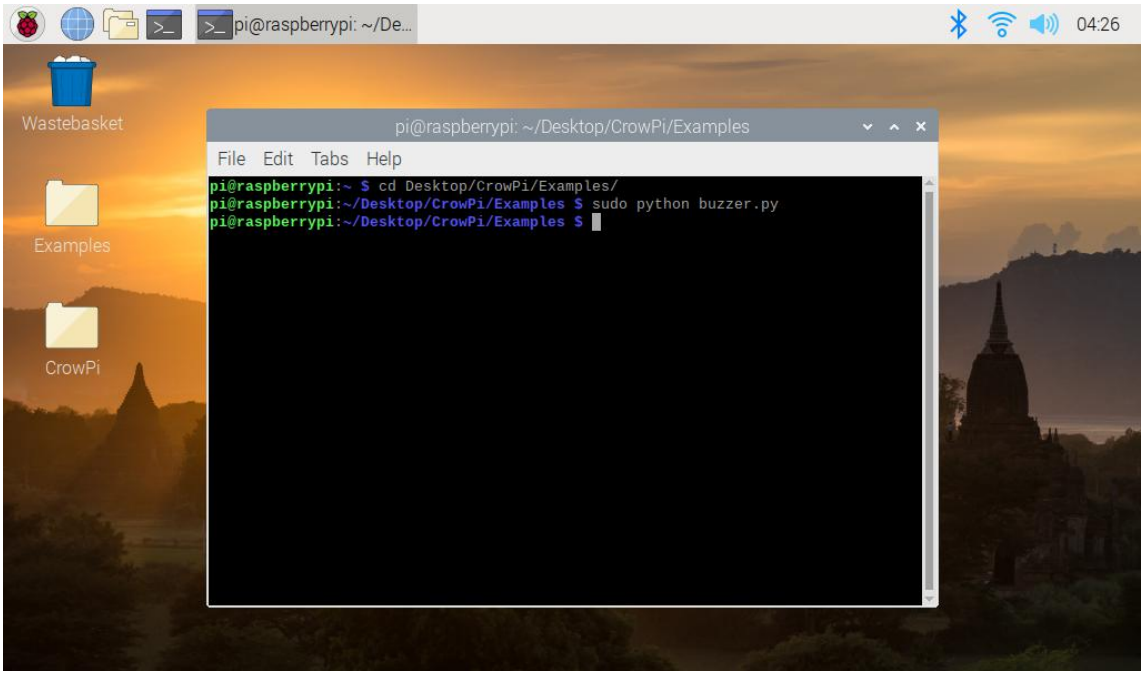
2) Change folder into the desktop folder using "cd Desktop" as the command inside the terminal

```
1 cd Desktop/CrowPi/Examples/
```



3) Write the command “sudo python <script name>” in order to execute the python script, for example “sudo python buzzer.py”

```
1   sudo python buzzer.py
```

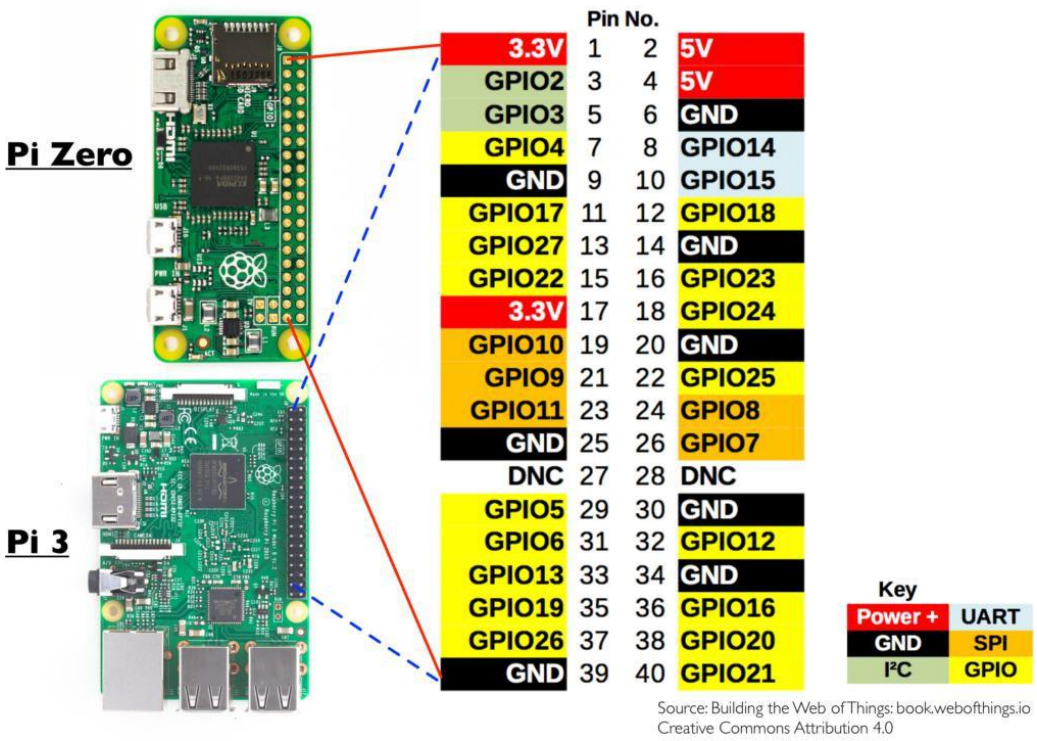


The sudo command gives us root permissions (admin permissions) which are required by the GPIO library, afterwards we write “python” to tell the system that we want to execute command using the python library. At the end, we write the script name as we downloaded it to the desktop, and that’s it! You’ve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

Now that you know how to download scripts and execute them, we are ready to start the tutorials!

GPIO Usage introduction

Basics of GPIO and how to use GPIO Input/Output



In this lesson we'll go through understanding what GPIO is, how it works and how to control it.

What will you learn

At the end of this lesson you'll be able to:

* Understand what GPIO is and how it works

* Know the difference between the BCM and BOARD GPIO

* Understand and be able to control both GPIO input and GPIO output

What will you need

* CrowPi Board after initial installation

Requires switching modules using the switch

* No

What is GPIO

"General-purpose input/output (GPIO) is a generic pin on an integrated circuit or computer board whose behavior—including whether it is an input or output pin—is controllable by the user at run time." ~Wikipedia

In simple language:

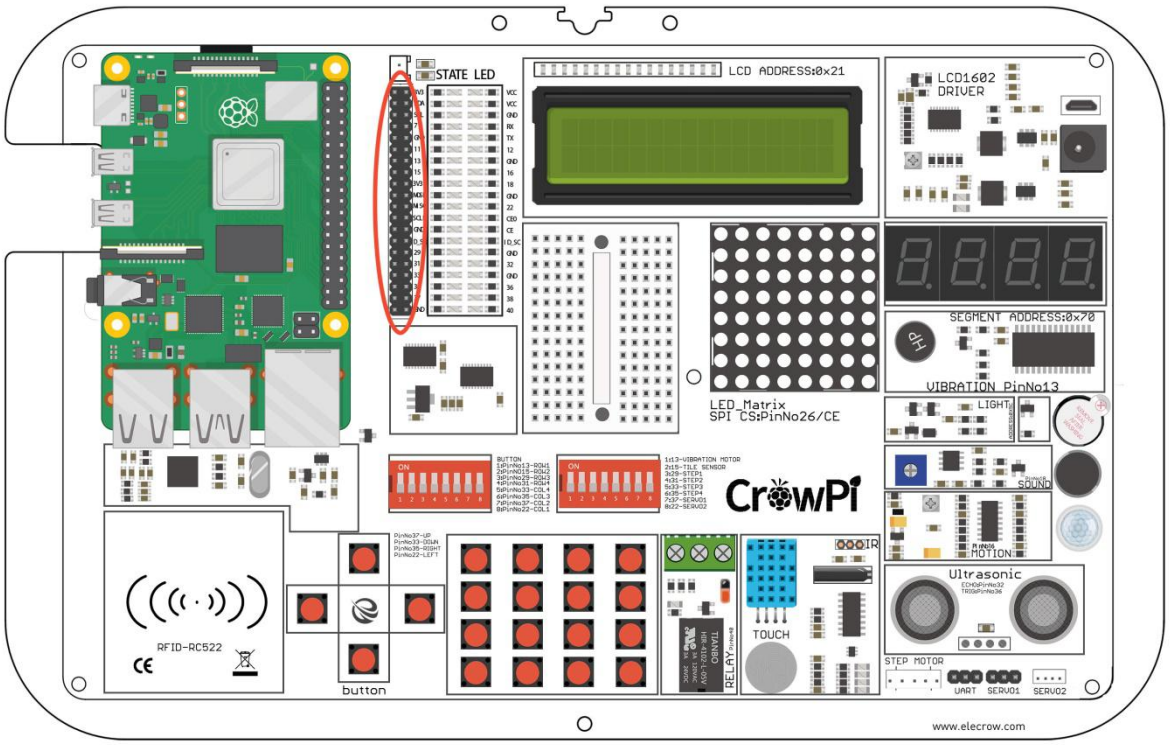
GPIO pins have no specific dedicated purpose, they can either be configured as input pins or output pins and their general purpose depends on what you want to accomplish.

Example for input pin: a button would be considered input, because you click on it.

Example for output pin: a buzzer would be considered output, because it receives a signal to buzz.

The location of the GPIO pins

The GPIO pins located on the right side of the Raspberry Pi board if you're looking from the CrowPi perspective



GPIO Schemes

There are 2 possible Raspberry Pi GPIO Scheme: GPIO.BOARD and GPIO.BCM

The GPIO.BOARD option specifies that you are referring to the pins by the number of the pin and the plug - i.e the numbers printed on the board (inside the circles on the diagram below)

The GPIO.BCM option means that you are referring to the pins by the "Broadcom SOC channel" number, these are the numbers after "GPIO" in the green rectangles around the outside of the below diagrams:

We will use the BCM numbers in the lessons.

How to use the GPIO pins

In our examples we use Python language to control the GPIO pins.
In python, we have a strong magical library called "RPi.GPIO",which is a library that helps us to control the pins programmatically using Python.

Check the below example and the comments inside the code for a better understanding of how it works in the physical world.

The first step will be to import the library by the command “RPi.GPIO as GPIO”; afterwards comes the “time” library by “import time”.

Then we setup the GPIO mode to GPIO.BCM
(not BOARD as we discussed before)

We declare the input pin as pin number **11** for our example and output pin as pin

12. (the input is the touch sensor and the output is the buzzer)

We send output to the output pin wait **1** second and then turn it off.

Then to confirm the input, we go through a loop till the GPIO.input receives the input signal, we print “Input Given” to make sure the click was confirmed, clean the GPIO using GPIO.cleanup() and quit the script.

In order to understand and grow your knowledge about GPIO's purpose and usage we highly recommend reading the official documentation about the RFID GPIO modules.

Follow this link to read more about the official Raspberry Pi GPIO Documentation:

https://pythonhosted.org/RPIO/

Lesson 1

Using the buzzer as an alert notification

After the previous class, we understand how to use the GPIO pin both as output and input. To test it, we will go with real-life example and apply our knowledge from the previous class into one of the modules over the board.

The Module we will use is the “buzzer”. The buzzer, as the name states, buzzes.

We will use the GPIO output to send a signal to the buzzer and close the circuit to make a loud buzzing noise, then we will send another signal to turn it off and close the circuit.

What will you learn

At the end of this lesson you'll be able to:

* Be able to control the buzzer module using GPIO output

What will you need

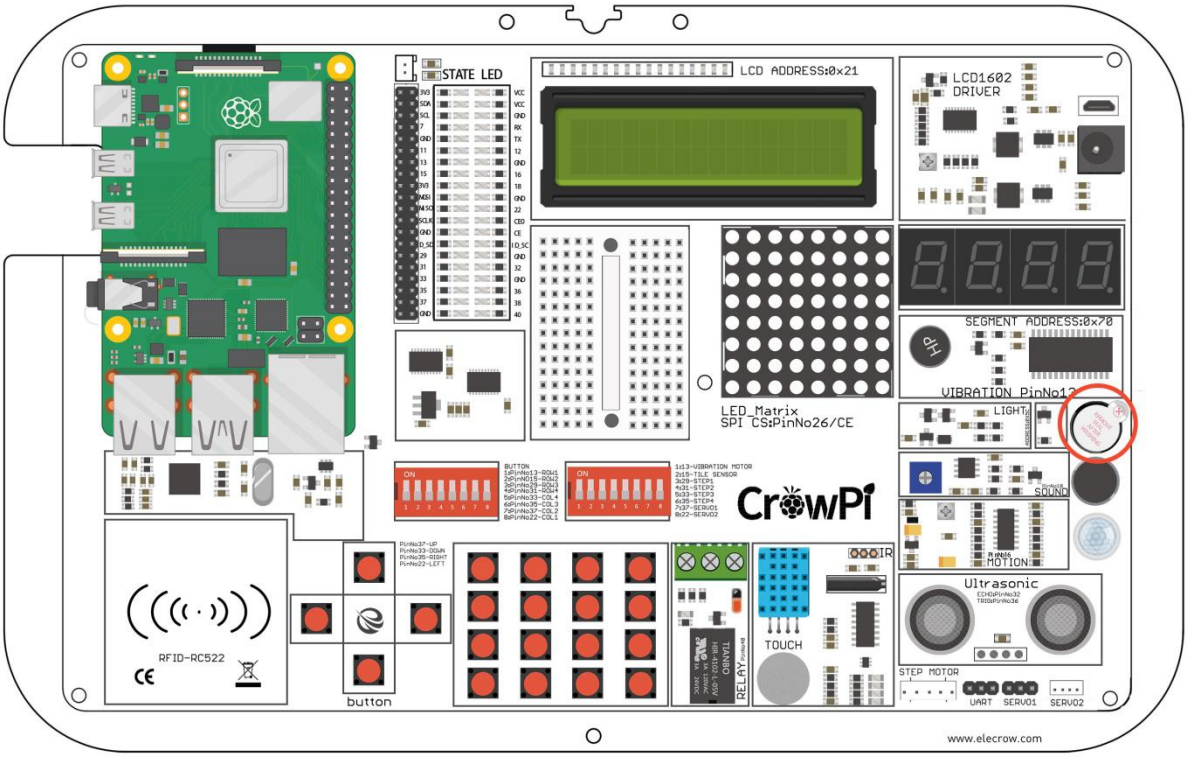
* CrowPi Board after initial installation

Requires switching modules using the switch

* No

Location of the buzzer on the CrowPi

The The buzzer is located on the right side of the CrowPi board, it’s easily detected by the loud noise it makes when activated.



The first time you use your Raspberry Pi, the Buzzer sensor might be sealed with a protective sticker.

Make sure to unseal the sticker by simply tearing it off and expose the buzzer itself.

Activating the Buzzer

Just as in the previous example, we've prepared a special script with detailed comments that will explain how the whole buzzing process works and how we are able to control the buzzer using GPIO output.

At first we import RPi.GPIO library and the time library for sleeping. Then we configure the buzzer at pin 12, we will be setting up the mode of GPIO to GPIO BOARD and setting up the pin as OUTPUT pin.

We will output a buzzing signal for 0.5 seconds and then turn it off to prevent ongoing loud noise.

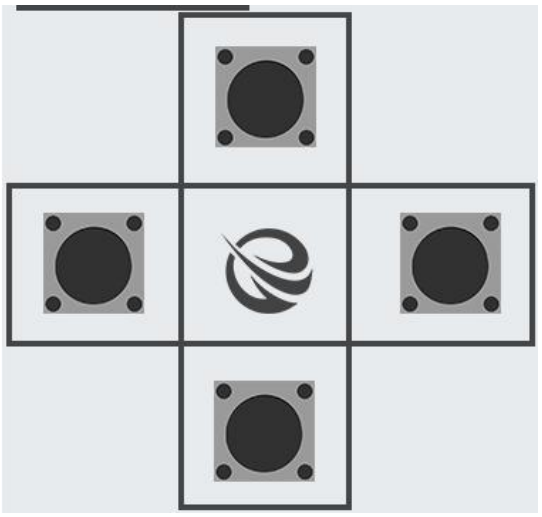
```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# http://elecrow.com/

import RPi.GPIO as GPIO
import time

buzzer_pin = 18

GPIO.setmode(GPIO.BCM)
GPIO.setup(buzzer_pin, GPIO.OUT)

# Make buzzer sound
GPIO.output(buzzer_pin, GPIO.HIGH)
time.sleep(0.5)
# Stop buzzer sound
GPIO.output(buzzer_pin, GPIO.LOW)

GPIO.cleanup()

```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  buzzer.py
```

Lesson 2

Get input from a button to control the Buzzer.



After successfully demonstrating how to turn on and off the buzzer now it's time to get things a bit more exciting. In this lesson we'll combine a button with the buzzer so the buzzer will turn on only by pressing the button.

This time we'll use 2 GPIO setups.

One will be the GPIO.INPUT which will take charge of the button as an input way to get the "press", another one will be the GPIO.OUTPUT which will send a signal to the buzzer to make some noise.

What will you learn

At the end of this lesson you'll be able to:

* Be able to activate the buzzer by pressing a button

What will you need

* CrowPi Board after initial installation

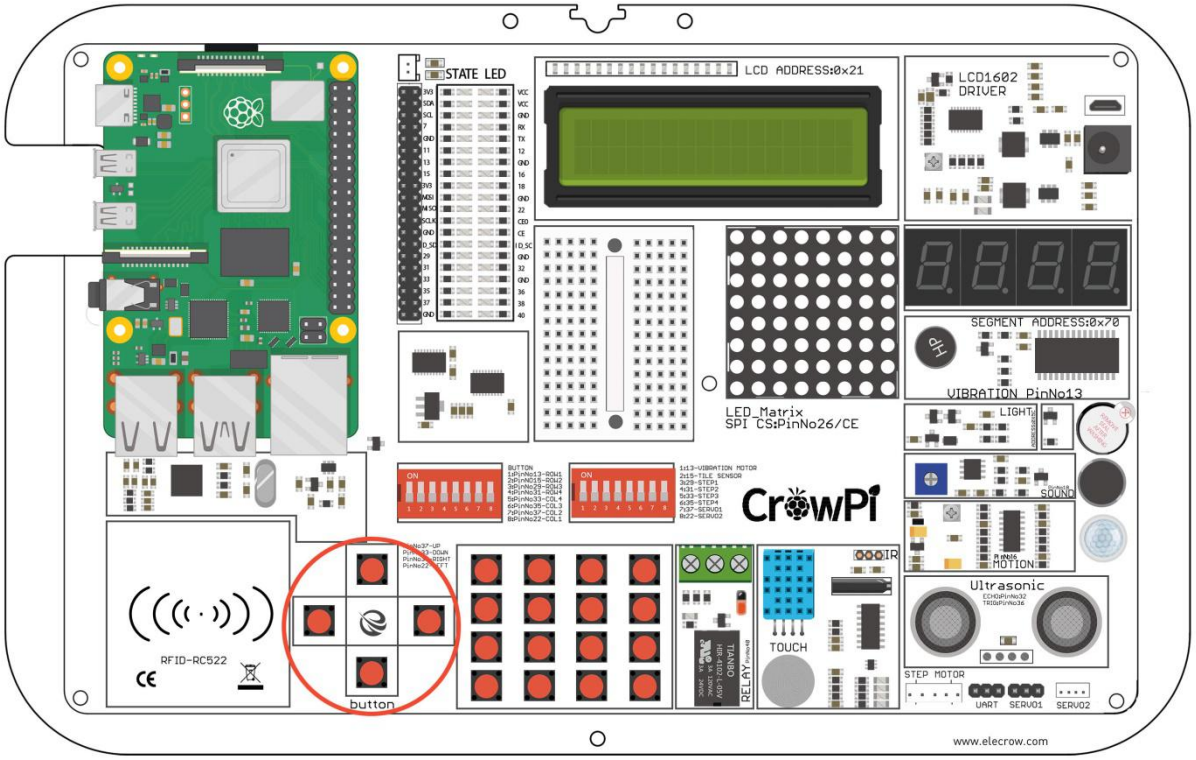
Requires switching modules using the switch

* **Yes**, the left switch - turn ALL the pins ON by turning them UP (refer to page number 5 if you forgot how to switch the sensors)

Activation button location

For our example we'll use the top button of the 4 buttons on the down left side next to the

NFC module to activate the buzzer by pressing it



Technically, we can choose any of the 4 buttons out of the independent buttons, for convenience we'll use the top one of the 4 independent buttons.

In order to change which button we would like to use, we need to set a different GPIO pin that is suitable for the button we chose.

The 4 buttons GPIO pins numbers are:

GPIO37 - Up

GPIO27 - Down

GPIO35 - Right

GPIO22 - Left

Try switching to any other of those GPIO numbers and see how it works!

Activating the Buzzer by pressing a button

For this part of our tutorial we'll need to use 2 GPIO settings, one is input and one is output.

The GPIO input will be used to determine whenever a button has been clicked or not, and the GPIO output will be used to activate the buzzer as soon as the button is pressed.

As you can see in the image below - we set up 2 pins; one is buzzer_pin and one is button_pin, the action will be set to forever until CTRL+C is pressed,

If you push the up button on the CrowPi board, the buzzer will make sound! Stop pressing it and the buzzer sound will stop.

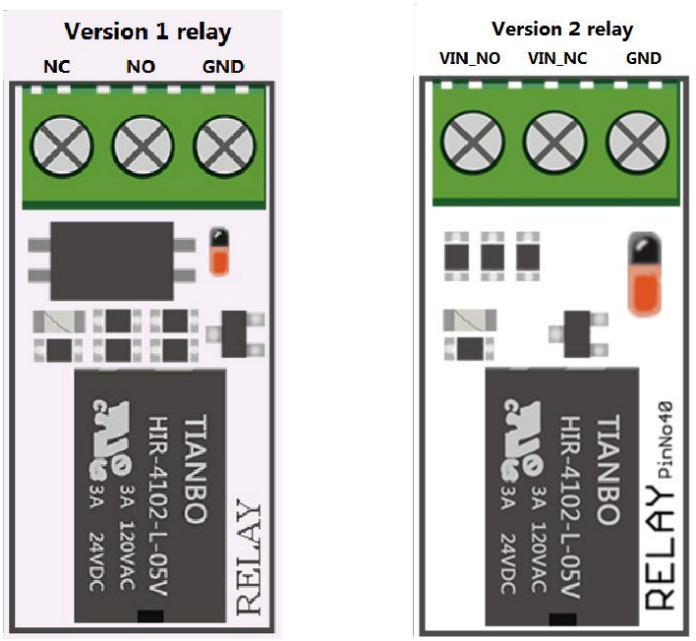
```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# http://elecrow.com/

import RPi.GPIO as GPIO
import time

# configure both button and buzzer pins
button_pin = 26
buzzer_pin = 18

# set board mode to GPIO.BCM
GPIO.setmode(GPIO.BCM)

# setup button pin asBu input and buzzer pin as output
GPIO.setup(button_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(buzzer_pin, GPIO.OUT)

try:
    while True:
        # check if button pressed
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  button_buzzer.py
```

Lesson 3

How Relay works and how to control it.



There are two version of Relay in our CrowPi, you can distinguish the version of your CrowPi by the pin name on the relay. Version 1 relay: The names of the three pins on the relay are NO, NC, COM. Version 2 relay: The names of the three pins on the relay are VIN_NC, VIN_NO, GND. **Note: The method used by different versions of the relay will be different, please choose the method corresponding to your own relay version is to use the relay, otherwise the board may be burned out!**
After we've completely finished with the buzzing part, it's time to move on.

In lesson 3 we are going learn about how to use Relay, what's the function of the Relay and how to control it.
A relay is an electrically operated switch. Many *relays* use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state *relays*. *Relays* are used where it is necessary to control a circuit via a separate low-power signal, or where several circuits must be controlled by one signal.

In our example we will show how to send a GPIO signal to open the relay and enable a custom circuit, and how to send another signal to close the relay, and close the circuit.

What will you learn

At the end of this lesson you'll be able to:

* Control a relay - open and close it using GPIO signal

What will you need

* CrowPi Board after initial installation

Requires switching modules using the switch

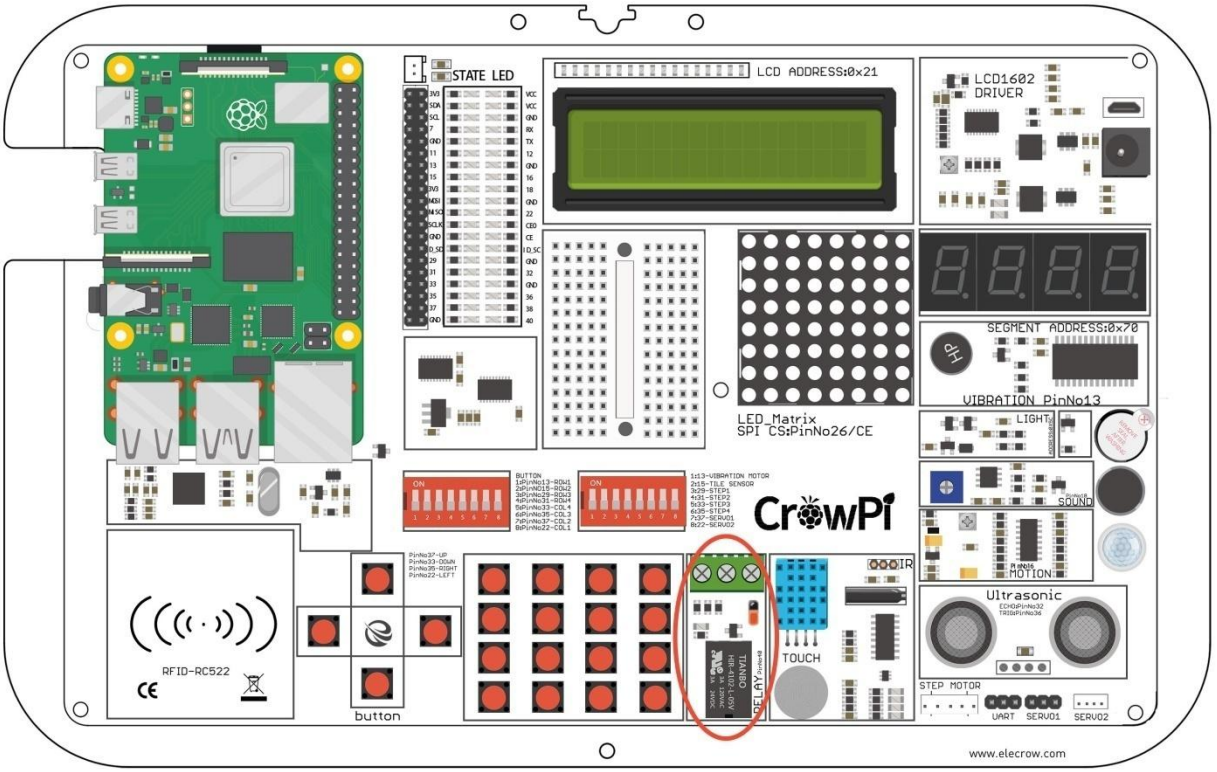
* No

Relay location on the CrowPi

The relay is located in the middle bottom part of the board, right after the button matrix, it can be easily detected by looking for a black object with 3 inserts pins, of which we'll use 2.

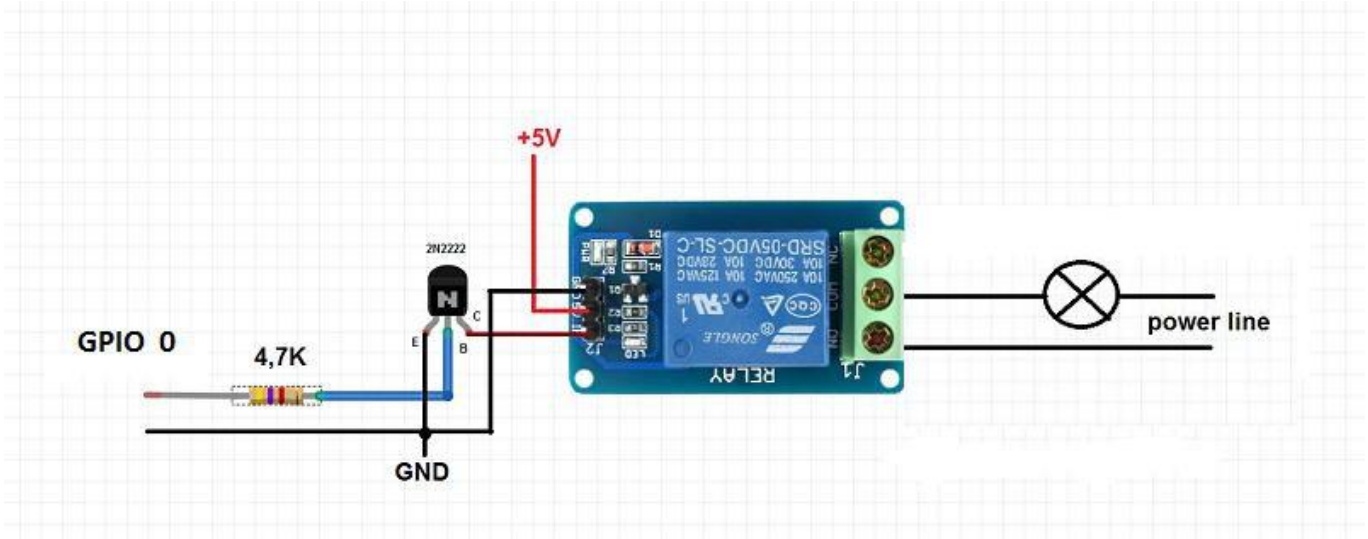
For version 1 relay, the pins we are going to use are marked with NO and COM; the NO pin will be left for now and we won't use it in our lesson. We'll need to connect the desired circuit to the right pin of the relay, and to the middle one; the left one is NC and would be left untouched.

For version 2 relay, the pins we are going to use are marked with VIN_NO and GND; the VIN_NC pin will be left for now and we won't use it in our lesson. We'll need to connect the desired circuit to the right pin of the relay, and to the middle one; the left one is VIN_NC and would be left untouched.



Controlling the Relay

Before we dive into our code, it's important to understand how relay works and what's the purpose of it. Let's look at the diagram below:



Let's look at the right side of the diagram, we can see that there are 2 black lines. Actually, those lines are the same line (eventually, they will connect together) they go through the relay so we can separate them with a "switch" and when we open the relay it will allow the flow through. So for example, if we are having an LED that has 2 wires, red and black (positive and negative), we can use the red wire or the black wire to cut it. **For version 1 relay**, connect one wire from the LED directly to the right side(NO) of the relay and the other wire from the center(COM) of the relay to the GND or to the VCC (positive) depending on which wire we are using. **For version 2 relay**, connect one wire from the LED directly to the center(VIN_NO) of the relay and the other wire from the right side(GND). Different from version 1 relay, you don't need to connect additional power to the relay, because it connects the relay directly through the power of the CrowPi, which means how much power you connect to the CrowPi, and how much power will be available when the relay is turned on! **So pay attention to the size of the power supply when connecting the module to the relay, otherwise the module will be burned out and even a fire!**

Using the relay you'll be able to close and open the circuit allowing the electricity to flow or blocking it, the result will be enabling or disabling certain electronic circuits for example LED light, electric fan and so on ....

It's VERY important that you DO NOT try to connect HIGH VOLTAGE equipment to the relay such as a table lamp, a coffee machine and other everyday household objects ...

This might cause electric shock and serious injury. Only follow the instructions in this guide for your own safety.

During the next lessons we'll learn how to create your custom circuit using the Bread Board where you can use the knowledge you gained from this page.

Now when we understand what is relay and how it works, after making sure that we followed **the safety guidelines and we didn't connect any high voltage device to it**, let's see the code:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# http://elecrow.com/

import RPi.GPIO as GPIO
import time

# define relay pin
relay_pin = 21

# set GPIO mode as GPIO.BCM
GPIO.setmode(GPIO.BCM)
# setup relay pin as OUTPUT
GPIO.setup(relay_pin, GPIO.OUT)

# Close Relay
GPIO.output(relay_pin, GPIO.LOW)
# Wait half a second
time.sleep(0.5)
# Open Relay
GPIO.output(relay_pin, GPIO.HIGH)
```

For version 1 relay, GPIO.LOW is to open the relay and GPIO.HIGH is to close the relay.

For version 2 relay, GPIO.LOW is to close the relay and GPIO.HIGH is to open the relay.

Again, different versions of relays are used in different ways. Be sure to use the relays in the correct way. Please pay attention to safety during the use process, safety first!

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  relay.py
```

Lesson 4

Cause vibrations using the vibration sensor

Have you always been wondering how does your phone vibrate when someone is calling you or when you get an SMS message?

Well, we included the exact same module into our CrowPi and now we are going to learn how to use it - the vibration module.

Vibration sensor - its internal structure is like a metal ball that is fixed to a special spring as one pole, and around it is the other pole. When the vibration gets to an certain level, the two poles are connected so as to determine when and how the shock occurs, which makes the vibration sensation.

What will you learn

At the end of this lesson you'll be able to:

* How to control the vibration sensor and send vibration signal

What will you need

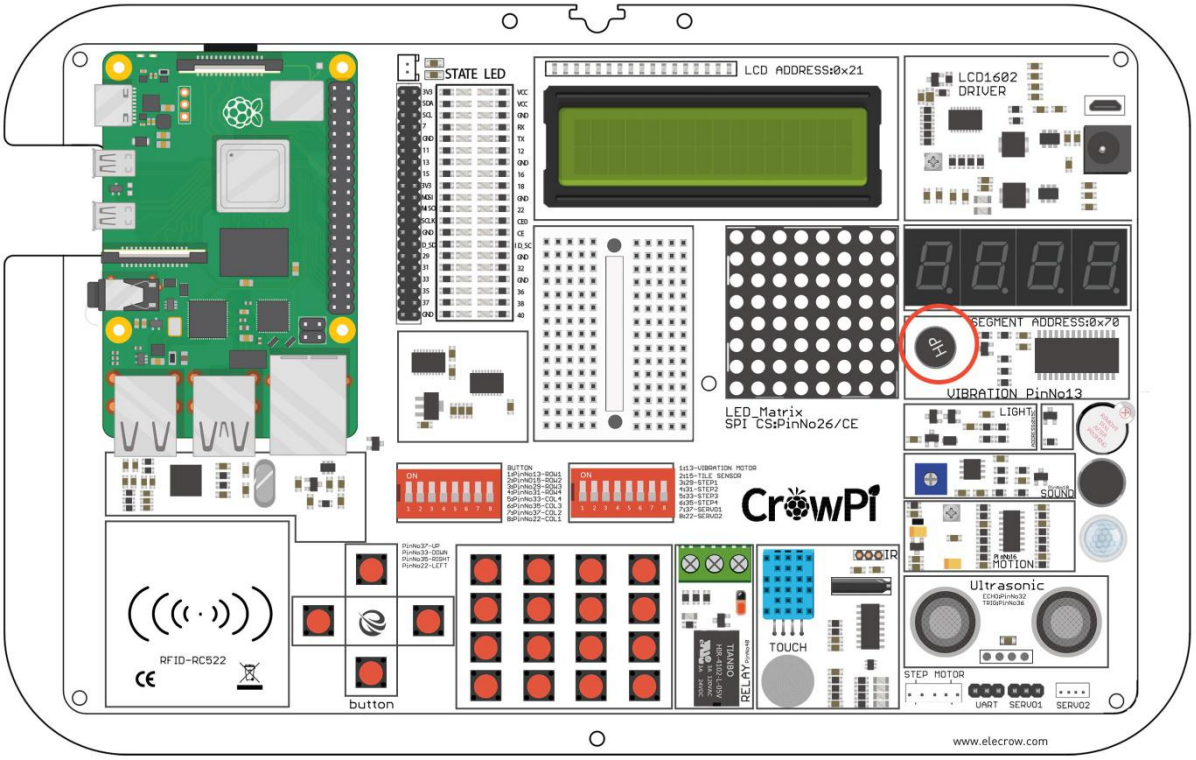
* CrowPi Board after initial installation

Requires switching modules using the switch

* **Yes**, The Right Switch, Pin number 1 - Make sure it’s on by switching it UP (refer to page number 5 if you forgot how to switch the sensors)

Vibration location on the CrowPi

The vibration sensor located on the right side of the matrix LED and under the segment LED, sometimes when it's on it's difficult to detect where the vibration comes from as it feels like the



whole CrowPi board is shaking and moving !

Activating the vibration sensor

The vibration sensor uses a GPIO.OUTPUT signal just as the buzzer and other modules we showed before. By sending an output signal the vibration sensor will vibrate, by stopping the signal with GPIO.LOW the vibration will stop.

This can be programmed through intervals of different **time.sleep()** which can make the vibration module vibrate in different tempos … have a try yourself and see what sort of vibration you can make!

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# http://elecrow.com/

import RPi.GPIO as GPIO
import time

# define vibration pin
vibration_pin = 27

# Set board mode to GPIO.BCM
GPIO.setmode(GPIO.BCM)

# Setup vibration pin to OUTPUT
GPIO.setup(vibration_pin, GPIO.OUT)

# turn on vibration
GPIO.output(vibration_pin, GPIO.HIGH)
# wait half a second
time.sleep(0.5)
# turn off vibration
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  vibration.py
```

Lesson 5

Detect sound using the sound sensor.

Sound is very interesting thing, we can figure out what sound is too noisy and which one is too low for us … but can the Raspberry Pi do that as well?

In this lesson we'll learn how to get input through the sound sensor, detect loud sounds and react accordingly. This is a great way to build your own alarm system that detects loud noise or maybe to turn on the LED by clapping!

What will you learn

At the end of this lesson you'll be able to:

* You will learn how to detect sound using the sound sensor module

What will you need

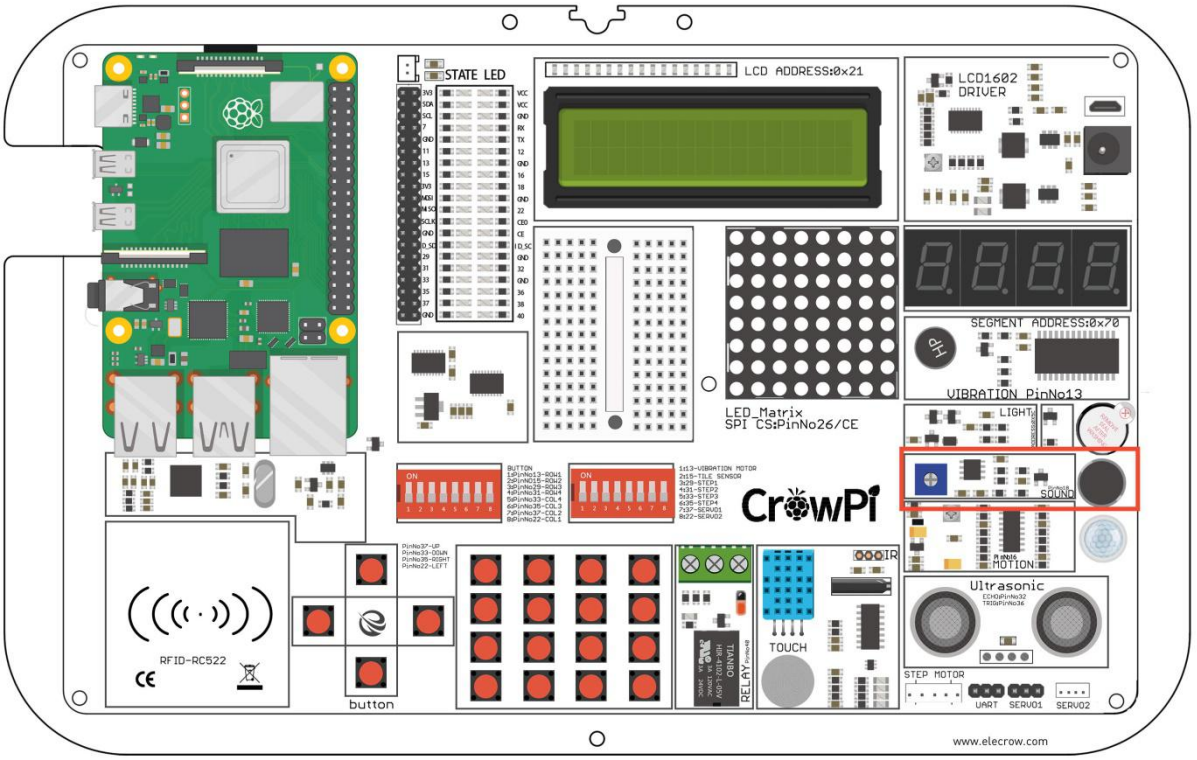
* CrowPi Board after initial installation

Requires switching modules using the switch

* No

Sound sensor location on the CrowPi

The sound sensor can be see next to the blue configuration square and the sensor itself is right under the buzzer. The sound sensor is made of 2 parts, one of which is the blue square to configure and regulate the sensitivity, and the other being sensor itself which detects loud noise.



Configuring the sensitivity

Our sensor contains a small controller that allows us manually to control the sensitivity of the noise for either too quiet or too loud.

In order to make our script work we first must learn how to control that sensitivity option … take a look at the picture above:

We'll need to turn the little blue square that is circled with a red oval in the picture to either left or right in order to control its sensitivity.

Turning the sensitivity controller can be done by simply using any Phillips screw driver.

The best way to know what sensitivity level is suitable for you is by running the script and clapping your hands once in a while or shout so you can see if there is an INPUT reaction from the sensor which means it detects the loud noise.

If the sensor doesn't detect the loud noise it means the sensitivity is too low and he won't react to it.

Increasing the sensitivity by turning the blue square will solve this issue immediately.

Getting input from the sound sensor

After we learned how to control the sensor sensitivity it's time to test it in real time and see how it works, take a look at the following code:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# http://elecrow.com/

import RPi.GPIO as GPIO
import time

# define sound pin
sound_pin = 24
# set GPIO mode to GPIO.BCM
GPIO.setmode(GPIO.BCM)
# setup pin as INPUT
GPIO.setup(sound_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

try:
    while True:
        # check if sound detected or not
        if(GPIO.input(sound_pin)==GPIO.LOW):
            print('Sound Detected')
            time.sleep(0.1)
except KeyboardInterrupt:
```

We first define our pin which is GPIO 18, then we set a while loop to keep this script running forever, we check if we got input from the sound sensor which indicates loud noise has been detected, and then we'll print "Sound Detected"

If we press CTRL+C we'll interrupt the script and clean our GPIO pins.

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  sound.py
```

Lesson 6

Detect low or bright light using the Light sensor.

The Light sensor is one of our favorites. It’s extremely useful in many projects and situations for example if you’d like to control something by detecting if there is bright light around, the light sensor is a great module for that exact case.

By using the light sensor we’ll be able to detect how much light is hitting, as each case is different we’ll need to play around to figure out which configuration is the most suitable for us.

What will you learn

At the end of this lesson you'll be able to:

* You will learn how to detect low light and bright light using the light sensor

What will you need

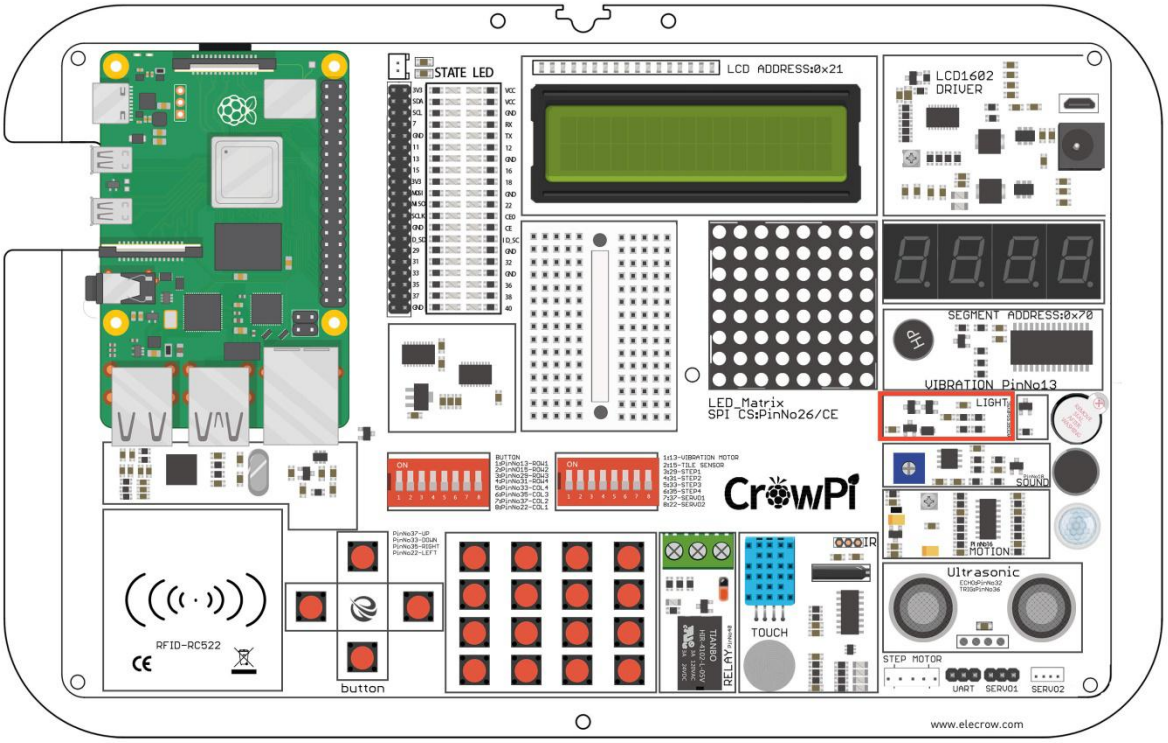
* CrowPi Board after initial installation

Requires switching modules using the switch

* No

Light sensor location on the CrowPi

The light sensor is almost invisible as it contains a very small part which is in charge of detecting the light. The light sensor located on the left side of the buzzer, if you cover that part with your hand you'll realize the output of the light sensor will be close to 0 as there is no light coming in …



Working with the light sensor

After we learned how to control the sensor sensitivity it's time to test it in real time and see how it works. The light sensor is a bit different from other sensors as it works using I2C and not using the normal GPIO the way we did before.

The script is longer than other scripts and more difficult to explain.

In short: we set binary data to control the light sensor like turning it on, off, getting high input and low input. Afterwards we use this function to "talk" with the light sensor and get the output we want depending on the light sensitivity around.

Then we simply use the function we've made of "sensor = LightSensor()" to get the light and then

we use "sensor.readLight()" to convert the binary data into readable brightness numbers (the higher the number, the brighter it is).

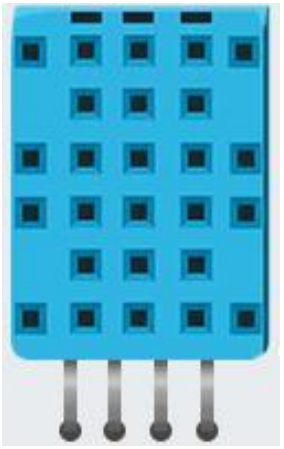
```
49          # into a decimal number
50          return ((data[1] + (256 * data[0])) / 1.2)
51
52      def readLight(self):
53
54          data = bus.read_i2c_block_data(self.DEVICE,self.ONE_TIME_HIGH_RES_MODE_1)
55          return self.convertToNumber(data)
56
57  def main():
58
59      sensor = LightSensor()
60      try:
61          while True:
62              print("Light Level : " + str(sensor.readLight()) + " lx")
63              time.sleep(0.5)
64      except KeyboardInterrupt:
65          pass
66
67  if __name__ == "__main__":
68      main()
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  light_sensor.py
```

Lesson 7

Detect room temperature and humidity



The DH11 is a very interesting and unique sensor as it not only combines one functionality but two! It contains both a humidity sensor and a temperature sensor which are pretty accurate which are great for any weather station project, or if you'd like to check the temperature and humidity in the room and make a smart home project! It can also be used to check the garden humidity and temperature to know if your flowers are in danger or they are cool.

What will you learn

At the end of this lesson you'll be able to:

* Control and get a reading of the humidity and the temperature from the DH11 sensor

What will you need

* CrowPi Board after initial installation

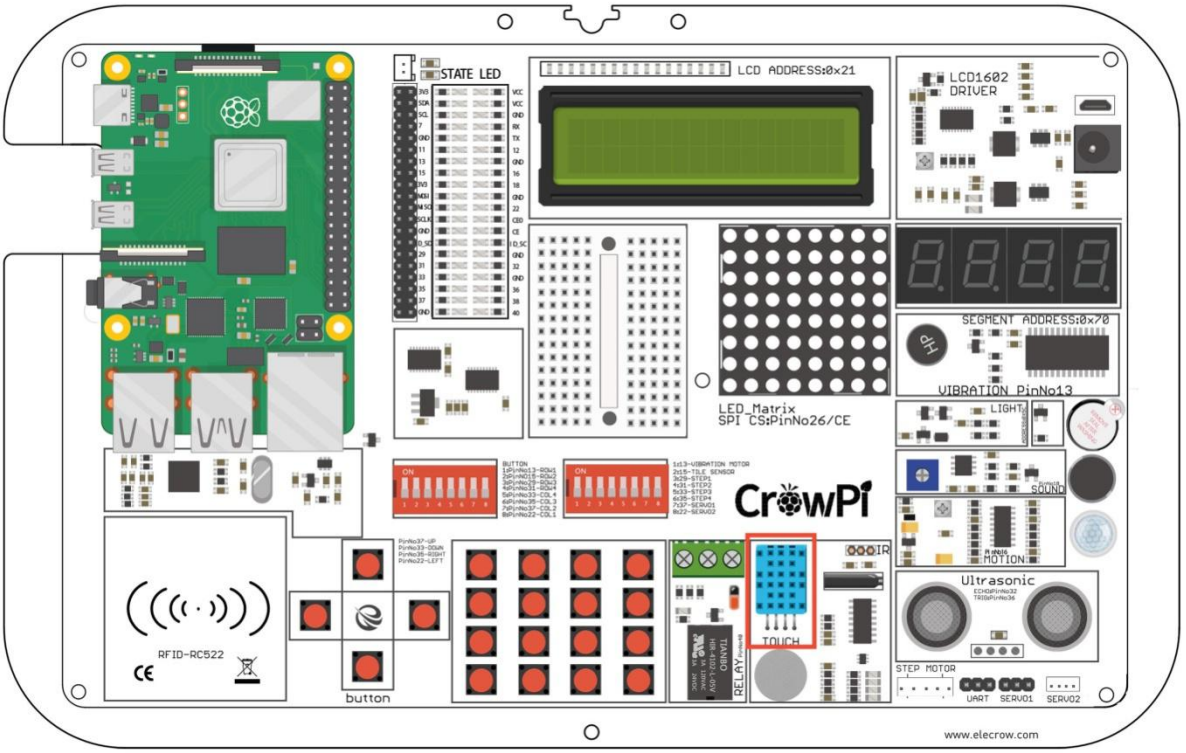
Requires switching modules using the switch

* No

DH11 sensor location on the CrowPi

The DH11 is very easy to detect, a small blue sensor with many little holes inside. it’s located right on

Top of the touch sensor and on the right side of the relay. The sensor doesn’t light up any LED or make any sound when it works but you will know it does by the information output you’ll get!



Working with the DH11 sensor

Working with DH11 is very easy using the Adafruit_DHT library. The library will return Temperature and Humidity as values which don't require any complicated math calculations and functions!

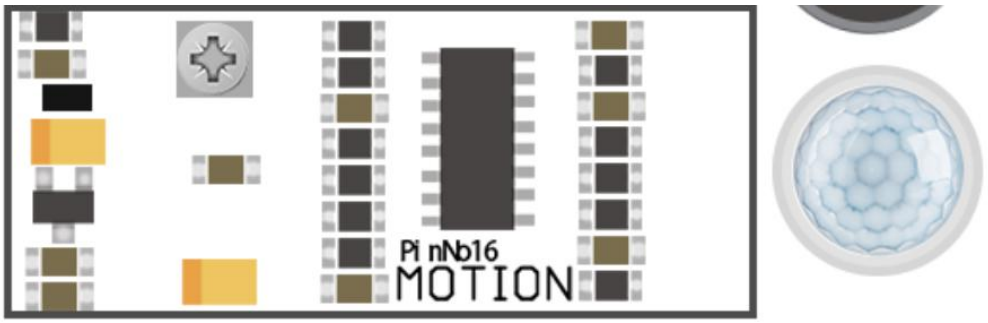
```
26 # set type of the sensor
27 sensor = 11
28 # set pin number
29 pin = 4
30
31 # Try to grab a sensor reading.  Use the read_retry method which will retry up
32 # to 15 times to get a sensor reading (waiting 2 seconds between each retry).
33 humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
34
35 # Un-comment the line below to convert the temperature to Fahrenheit.
36 # temperature = temperature * 9/5.0 + 32
37
38 # Note that sometimes you won't get a reading and
39 # the results will be null (because Linux can't
40 # guarantee the timing of calls to read the sensor).
41 # If this happens try again!
42 if humidity is not None and temperature is not None:
43     print('Temp={0:0.1f}*  Humidity={1:0.1f}%'.format(temperature, humidity))
44 else:
45     print('Failed to get reading. Try again!')
46
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  dh11.py
```

Lesson 8

Detect motion using the motion sensor.



The motion sensor is one of the most useful and often used sensors out there. As you've seen in our kick-starter video, we used the motion sensor to detect a thief coming in and trying to steal our CrowPi … by detecting their motion using infra-red light, we were able turn on a buzzer alarm and make the thief run away!

What will you learn

At the end of this lesson you'll be able to:

* Get output from the motion sensor and detect movement around the CrowPi

What will you need

* CrowPi Board after initial installation

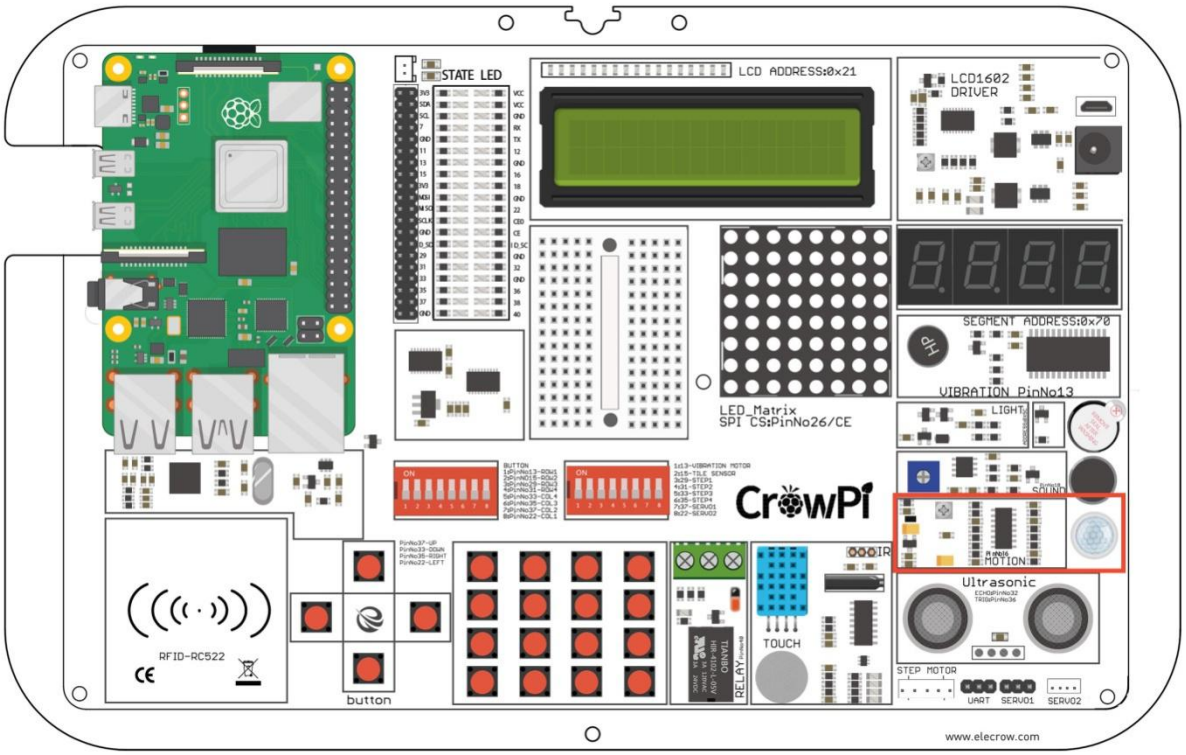
Requires switching modules using the switch

* No

Motion sensor location on the CrowPi

The motion sensor is located right under the sound sensor and contains a small white transparent cup to cover it. The little protector cup helps the sensor detect more surrounding movement by better dispersing the red infra-light.

The motion sensor doesn't make any sound or light, but it's very easy to see if it works by moving your hand on top of the CrowPi and see if it can detect movement!



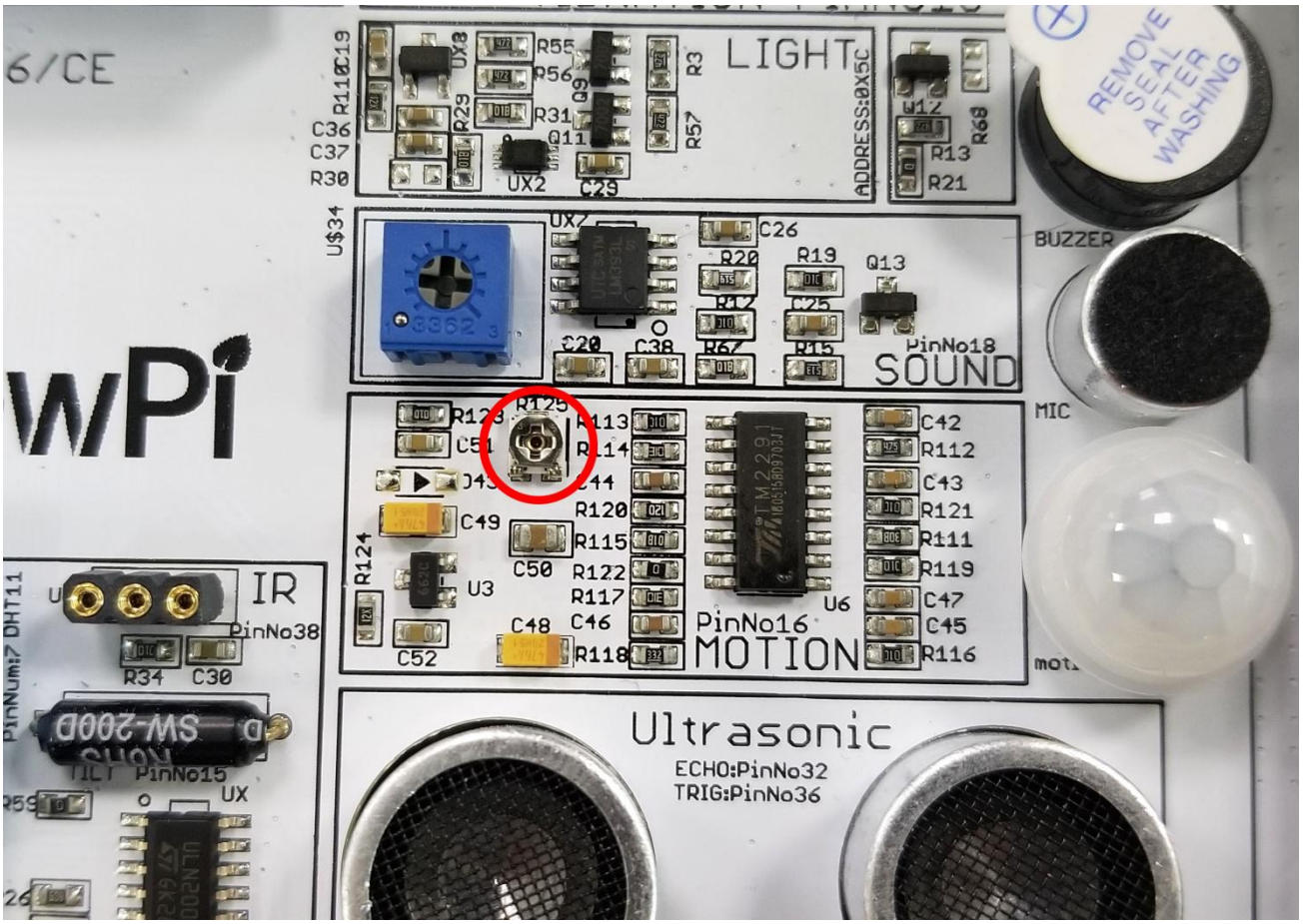
Configuring the sensitivity

The motion sensor includes a tiny screw right under the sound sensor potentiometer (the blue thing to configure the sound sensitivity)

We'll use that tiny potentiometer in order to adjust the sensitivity of the motion sensor.

By adjusting the sensitivity of the motion sensor we'll be able to let the motion sensor know from what

distance we would like to detect a motion (far away or close up) and that would allow us to have better control over our application.



By using a standard Philips flat head screwdriver, rotate the screw to the right or to the left when running the motion example script in order to find a suitable distance for the motion sensor.

Working with the motion sensor

The motion sensor is controlled by the GPIO pins. If movement is detected the motion sensor will send an input signal which will alert us that something is moving around. After couple of seconds it will turn the input off and wait for the next movement to happen …

```
# -*- coding: utf-8 -*-
# http://elecrow.com/

import RPi.GPIO as GPIO
import time

# define motion pin
motion_pin = 23

# set GPIO as GPIO.BCM
GPIO.setmode(GPIO.BCM)
# set pin mode as INPUT
GPIO.setup(motion_pin, GPIO.IN)

try:
    while True:
        if(GPIO.input(motion_pin) == 0):
                print("Nothing moves ...")
        elif(GPIO.input(motion_pin) == 1):
                print("Motion detected!")
        time.sleep(0.1)
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  motion.py
```

Lesson 9

Getting distance information using the Ultrasonic sensor.

Distance is a very useful thing in our daily life ..we use it to measure walls before we buy new furniture and when driving to make sure we won't run into the other cars! Cars by the way ... use the same ultrasonic sensor in their distance measuring functions that we will use in our demonstration! In this tutorial we will learn how to use an ultrasonic sensor to measure the distance and output it on the CrowPi screen

What will you learn

At the end of this lesson you'll be able to:

* Control the Ultrasonic sensor and get distance as output

What will you need

* CrowPi Board after initial installation

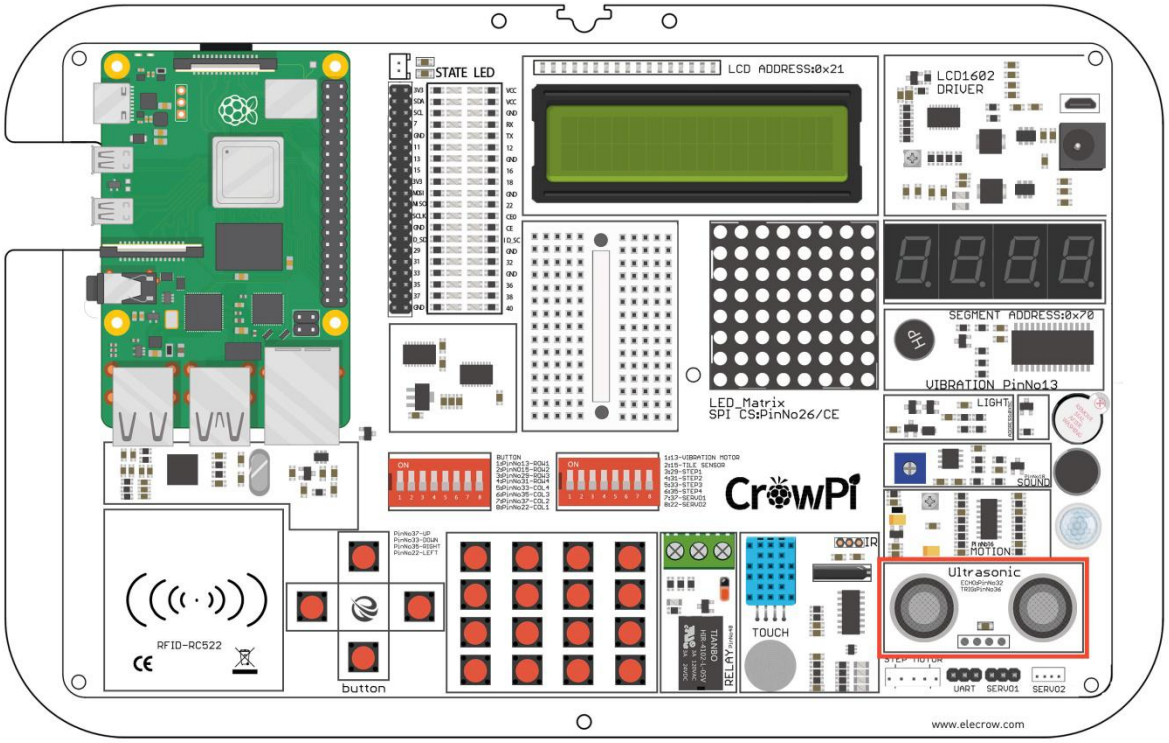
Requires switching modules using the switch

* No

Ultrasonic sensor location on the CrowPi

The ultrasonic sensor is located on the bottom right of the CrowPi board right on top of the servo and UART pins, it's easily recognized by its 2 giant circles which looks like robotic eyes!

We will use our hands to move them up and down on top of the distance sensor in order to measure the distance between our hands and the CrowPi!



Working with the ultrasonic distance sensor

The distance sensor works using GPIO INPUT but it's a bit different from what we did in our previous lessons.

The distance needs some interval to be able to detect the distance in an accurate way, it uses pulses of HIGH and LOW in order to do so. After we know the duration of the pulse we'll be able to use Math and Science to calculate the distance between the distance sensor and the object that is standing in front of it.

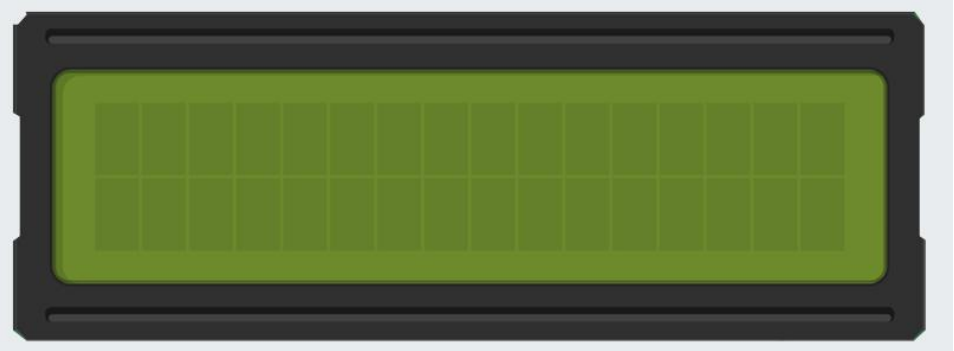
```
22
23 GPIO.output(TRIG, True)
24 time.sleep(0.00001)
25 GPIO.output(TRIG, False)
26
27 while GPIO.input(ECHO)==0:
28   pulse_start = time.time()
29
30 while GPIO.input(ECHO)==1:
31   pulse_end = time.time()
32
33 pulse_duration = pulse_end - pulse_start
34
35 distance = pulse_duration * 17150
36
37 distance = round(distance, 2)
38
39 print("Distance: %scm" % distance)
40
41 GPIO.cleanup()
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  distance.py
```

Lesson 10

Controlling the LCD Display



LCD (and matrix display) is probably the most fun and exciting part when building projects using the CrowPi. Using the LCD display you could show data that you collect using your CrowPi sensors and also update it in real time depending on the changes that the modules go through!

For example: yesterday it was really hot but today it’s really cold - let the CrowPi LCD change itself automatically with the latest and most updated info so you will not accidentally wear the wrong clothes for school / work!

What will you learn

At the end of this lesson you'll be able to:

* What you will learn how to control the LCD display and write data into it

What will you need

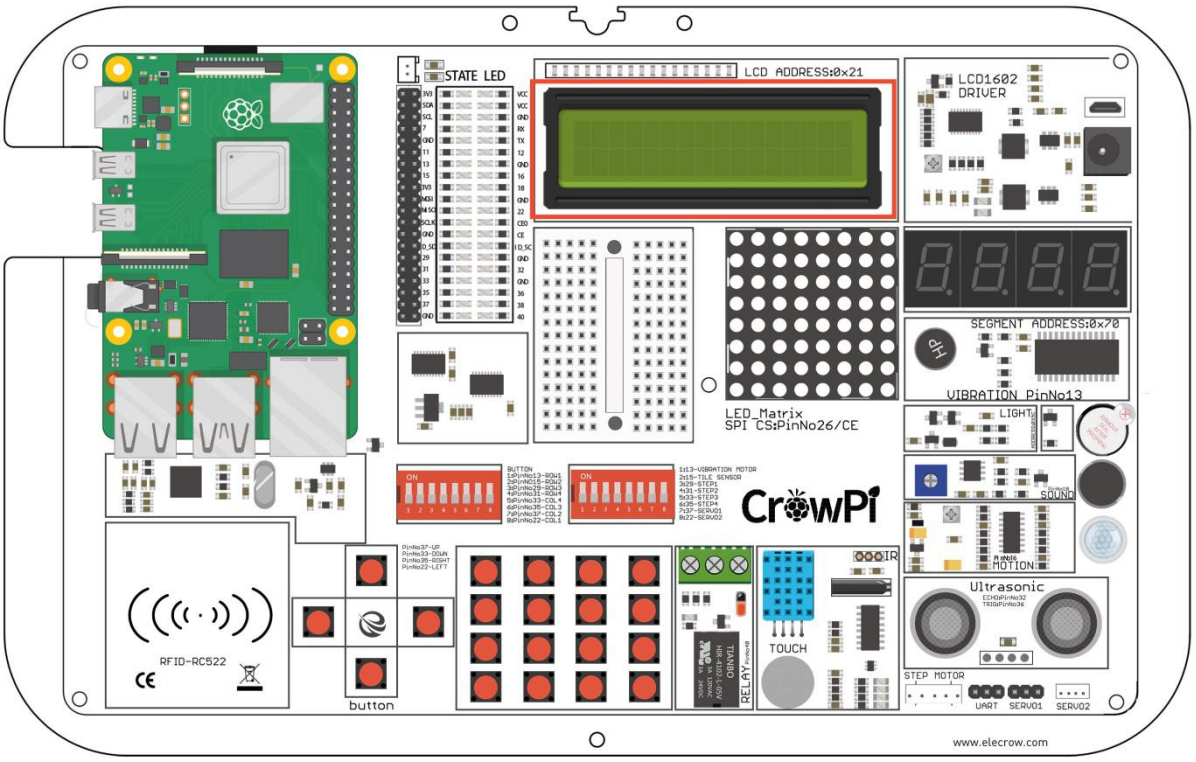
* CrowPi Board after initial installation

Requires switching modules using the switch

* No

LCD Screen location on the CrowPi

The LCD screen takes up the biggest part of the CrowPi board so we are sure you noticed it immediately! As soon as you are running the demo script and the examples, the CrowPi will turn on with beautiful background light that can be seen even when all the lights in the room are turned off.



Configuring the LCD brightness

The LCD contains a tiny screw on the left side of the power circuit on the CrowPi, this screw will help us to adjust the brightness and the contrast of the LCD screen in case we need to.

Learning how to use this is incredibly important - if you experience a dark LCD screen, know that adjusting this potentiometer will solve the problem.

By using a standard Philips screwdriver, rotate the screw to the right or to the left when running the LCD example script in order to find the suitable brightness for you.

Working with the LCD

The I2C as with some other sensors also doesn't work on GPIO Technology instead we use something called "I2C" (The same I2C we used for the light sensor in our previous examples), the address we'll use for the LCD screen is 21, by connecting to this I2C address we'll be able to send commands for example: writing text or numbers, turning on the backlight of the LCD, turning it off, enabling cursor, etc …

For controlling the LCD we'll use Adafruit_CharLCDBackpack which is Adafruit framework, and makes things a lot of easier for us when working with such a complicated product!

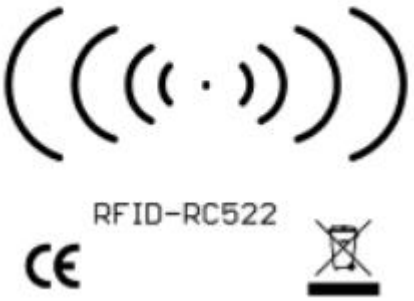
```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Example using a character LCD backpack.
import time
import Adafruit_CharLCD as LCD

# Define LCD column and row size for 16x2 LCD.
lcd_columns = 16
lcd_rows    = 2

# Initialize the LCD using the pins
lcd = LCD.Adafruit_CharLCDBackpack(address=0x21)

try:
    # Turn backlight on
    lcd.set_backlight(0)

    # Print a two line message
    lcd.message('Hello\nworld!')

```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  lcd.py
```

Lesson 11

Read / Write an RFID card using the RFID module.



The RFID module is one of the most interesting and useful modules in the market, used world wide in a wide variety of solutions such as: smart door locks, employee entry cards, business cards, and even on dog collars!?

No matter what kind of project you're into - an RFID module will definitely come into play!

What will you learn

At the end of this lesson you'll be able to:

* Control the RFID, Read and Write Data from it and recognize the chips

What will you need

* CrowPi Board after initial installation
* RFID Chip (included with the CrowPi)

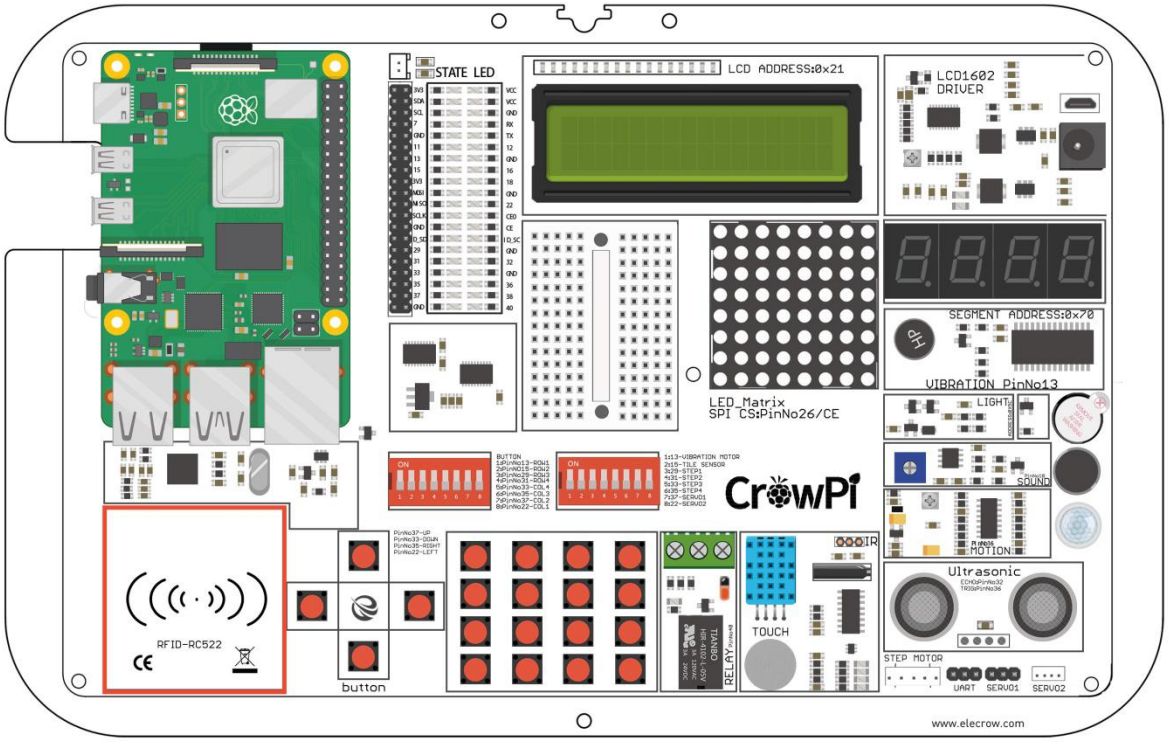
Requires switching modules using the switch

* No

RFID Module location on the CrowPi

The RFID module is located right below the Raspberry Pi (either zero or 3) it looks like a small chip with “wifi” illustration coming out of it which means wireless connectivity (which is what RFID does) In order to use it we’ll need to take the chip or the card that comes with the CrowPi and pass it over the CrowPi RFID Chip area close enough for our script to detect it.

2-4cm should be close enough, have a try!



Working with the RFID

Working with the RFID module is pretty straight forward.

We have 3 functions: Authorizing, Read, Write and Deauthorizing.

First step will be when you touch the NFC. At that time the module and our script will try to Authorize the chip using the default password configuration (if you haven't changed it, it should work) afterwards, when authorization is successful, it will read the data and display it on the screen. After it has finished it will Deauthorize and quit the script.

In another script example we'll be able to Authorize, Read, Re-write the data, and then Deauthorize.

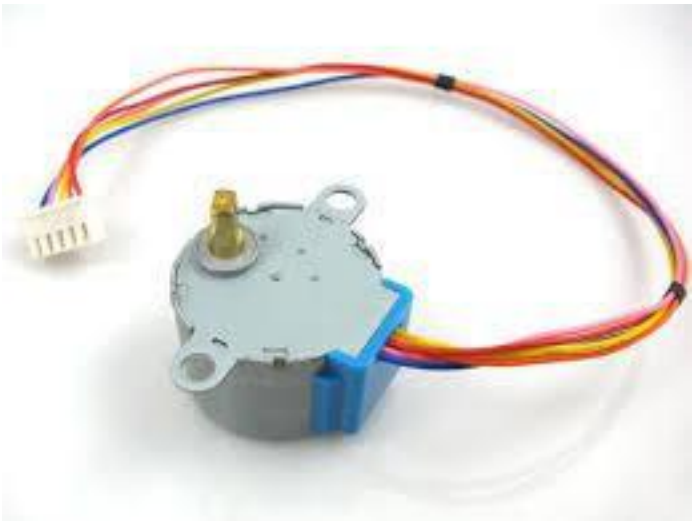
```
#!/usr/bin/env python
# -*- coding: utf8 -*-
# Copyright 2014,2018 Mario Gomez <mario.gomez@teubi.co>
import RPi.GPIO as GPIO
import MFRC522
import signal

continue_reading = True
# Incase user wants to terminate, this function is exactly for that reason.
def end_read(signal,frame):
    global continue_reading
    print("Ctrl+C captured, ending read.")
    continue_reading = False
    GPIO.cleanup()

signal.signal(signal.SIGINT, end_read)
# create the reader object
MIFAREReader = MFRC522.MFRC522()

# Welcome greeting
print("Welcome to MFRC522 RFID Read example")
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  RFID_Read.py
```

Lesson 12

Using the step motor and making step movements.



Step motor is a great way to control movements, used in wide variety of projects such robots, automatic coffee machines, 3D printers, and more! Learning how to use the step motor can give you a huge boost in practical coding abilities!

What will you learn

At the end of this lesson you'll be able to:

* Control the step motor and make step movements

What will you need

* CrowPi Board after initial installation
* Step motor (included with CrowPi kit)

Requires switching modules using the switch

* **Yes**, Right switch - pins number 3,4,5,6 - turn it on by switching it UP (refer to page number 5 if you forgot how to switch the sensors)

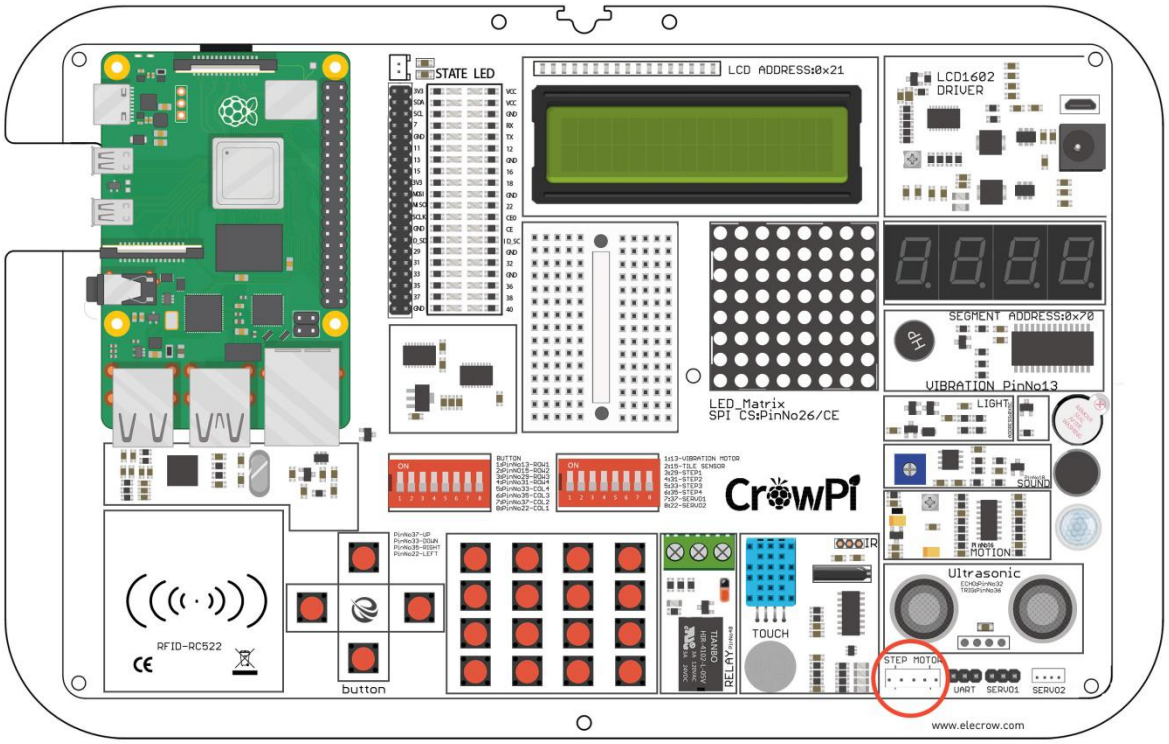
Step Motor Module location on the CrowPi

The step motor is an independent module that we'll need to connect to the board

We'll need to take the step motor which is included with the kit and connect it to our CrowPi into a dedicated place that is located on the CrowPi Board.

Refer to the following picture, connect the Step motor where the red circle is:

the module gets a bit hot don't worry! That's totally normal and not dangerous at all.



Working with the Step motor

Working with the Step motor module is pretty straight forward.

The step motor is connected to 4 GPIO pins which, each time we turn them on, one by one move very quickly which makes the step motor “push” forward and take a step. We can use the function turn Steps to tell how many steps we want to make and turn Degrees to make quarter (90 degree) turns.

```
118                 # (supply with distance to move and radius in same metric)
119                 self.turn(round(512*dist/(2*math.pi*rad),0))
120
121 def main():
122
123         print("moving started")
124         motor = Stepmotor()
125         print("One Step")
126         motor.turnSteps(1)
127         time.sleep(0.5)
128         print("20 Steps")
129         motor.turnSteps(20)
130         time.sleep(0.5)
131         print("quarter turn")
132         motor.turnDegrees(90)
133         print("moving stopped")
134         motor.close()
135
136 if __name__ == "__main__":
137     main()
138
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  stepmotor.py
```

Lesson 13

Controlling servos motors using the servo interfaces.

Servo is a really cool module that lets us mechanically control devices and move parts. Using this servo we can make a smart trash bin, a candy box with smart opening / closing door, and many other interesting projects!

In this tutorial we'll learn how to use the servo using the interface that we installed on top of the CrowPi.

What will you learn

At the end of this lesson you'll be able to:

* Connect and control the servo in multiple directions

What will you need

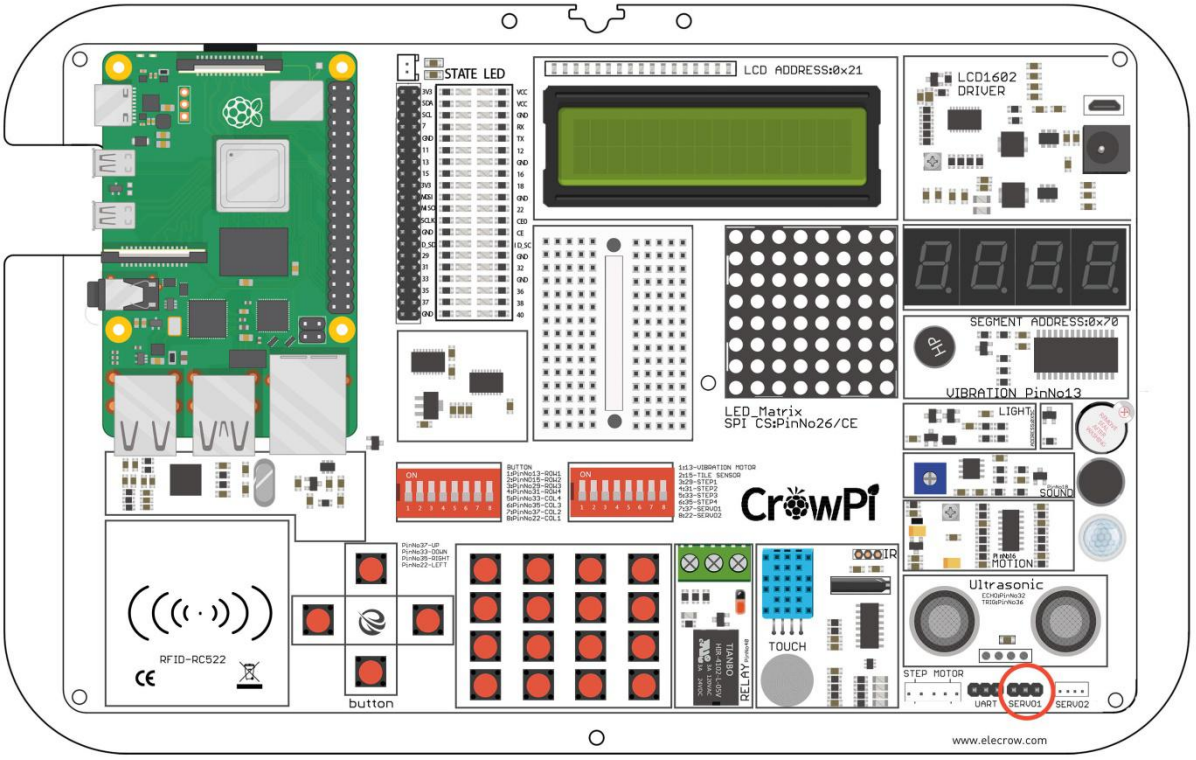
* CrowPi Board after initial installation
* Micro servo (comes with the CrowPi)

Requires switching modules using the switch

* **Yes**, Right switch - pins numbers 7,8 - turn it on by switching it UP (refer to page number 5 if you forgot how to switch the sensors)

Servo Module location on the CrowPi

The CrowPi contains 2 servos interfaces which both can be used for the purpose of controlling the servo. In this tutorial, we'll use interface number one which is marked as "Servo1".



You can always use the other interface as well but you'll need to modify the example script to the suitable GPIO pins.

The servo contains 3 pins: Positive, Negative and DATA.

The positive pin is always the red cable, the negative pin is the black one (also called ground), and the data cable is usually the colorful one.

Connect it as follows:

* **Red Cable (Positive) goes to the middle pin of the first servo**
* **Black Cable (Negative also called Ground) goes to the right pin of the first servo**
* **Colorful cable (might be blue or orange) goes to the GPIO pin on the left side of the first servo**

Working with the Servo

Working with the servo is pretty simple. Let's overview our example code to understand it

further: The servo pin is number 37 (the one we connected to the far left, which is the GPIO)

Every time the script will send direction to the servo module to turn left, we set degree of 100,10 to turn right we set degree of -100, -10.

The servo uses GPIO BOARD pin number 37.

You can play with the degrees and see where the servo will turn !

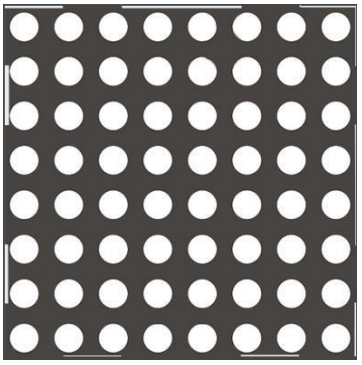
```
43      self.servo.ChangeDutyCycle( self._henkan( direction ) )
44      self.direction = direction
45
46  def main():
47
48      s = sg90(0)
49
50      try:
51          while True:
52              print("Turn left ...")
53              s.setdirection( 100, 10 )
54              time.sleep(0.5)
55              print("Turn right ...")
56              s.setdirection( -100, -10 )
57              time.sleep(0.5)
58      except KeyboardInterrupt:
59          s.cleanup()
60
61  if __name__ == "__main__":
62      main()
63
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  servo.py
```

Lesson 14

Controlling the 8x8 Matrix LED.



The matrix LED plays important rule in every blinking LED project.

Even though you might not see it at first glance, the Matrix LED can do way more than blinking red shiny LED's. It can be used to show information, text, emojis and even … Chinese characters! Perfect to show information in a fun and unique way and maybe even make a game like snake or a count down timer!

What will you learn

At the end of this lesson you'll be able to:

* Control Matrix LED, show text, control the speed and information flow

What will you need

* CrowPi Board after initial installation

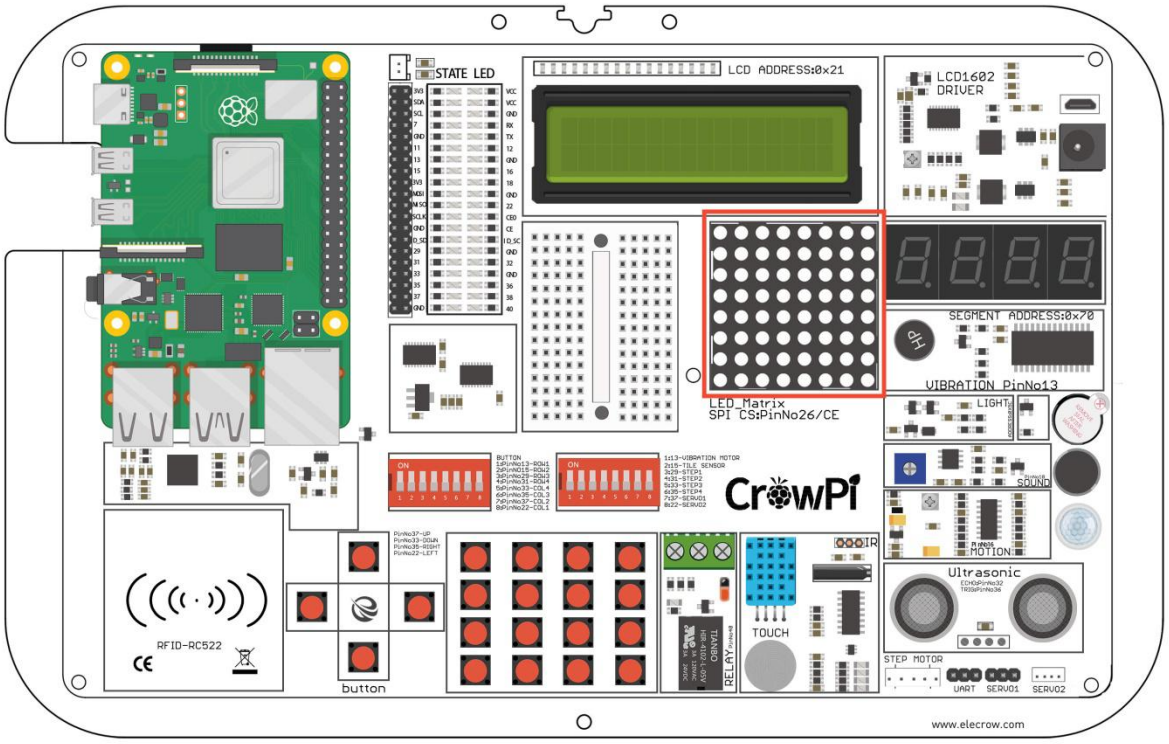
Requires switching modules using the switch

* No

Matrix Module location on the CrowPi

The matrix module is a big square module located on the left side of the segment LED and right under the LCD display. It can be easily recognized by the many small white dots which function as micro LED's.

Don't let the small size of white circles mislead you, this matrix LED can light up a dark room with ease!



Working with the Matrix Display

The Matrix LED demo we prepared contains everything for everything. Each line contains an example of demo text including show long text, controlling the speed of the text, choosing random words from a list, and much more.

In the script we make a string with a message, for example “Hello World”, and then we use the function show_message() to show the message on the Matrix Display.

We can control properties like delay which makes the message go faster or slower ..for example scroll_delay 0 will be pretty quick while delay of 0.1 will slow the message flow a bit …

The Matrix LED, unlike other modules, is controlled from the SPI interface.

Try multiple examples and change the code to see what happens!

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Copyright (c) 2017-18 Richard Hull and contributors
# License: https://github.com/rm-hull/luma.led_matrix/blob/master/LICENSE.rst
# Github link: https://github.com/rm-hull/luma.led_matrix/

# Import all the modules
import re
import time
from luma.led_matrix.device import max7219
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.core.virtual import viewport
from luma.core.legacy import text, show_message
from luma.core.legacy.font import proportional, CP437_FONT, TINY_FONT, SINCLAIR_FONT


def main(cascaded, block_orientation, rotate):

    # create matrix device
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  matrix_demo.py
```

Note: If Matrix_LED suddenly turns on without running a Python script, don't worry, you only need to run the Matrix_LED python script to solve this problem. This problem occurs because the signal of the Raspberry Pi easily interfered.

Lesson 15

Controlling the 7 Segment Display.

The segment LED is a really useful display when it comes to numbers and data

It can show us the time, count how many times we did certain things, and even be used to scare our friends with fake time bomb!

Segment LED is used in many industrial solutions such as elevators! And using it will definitely be useful to you in the future!

What will you learn

At the end of this lesson you'll be able to:

* Control the segment LED and show different numbers and data on it

What will you need

* CrowPi Board after initial installation

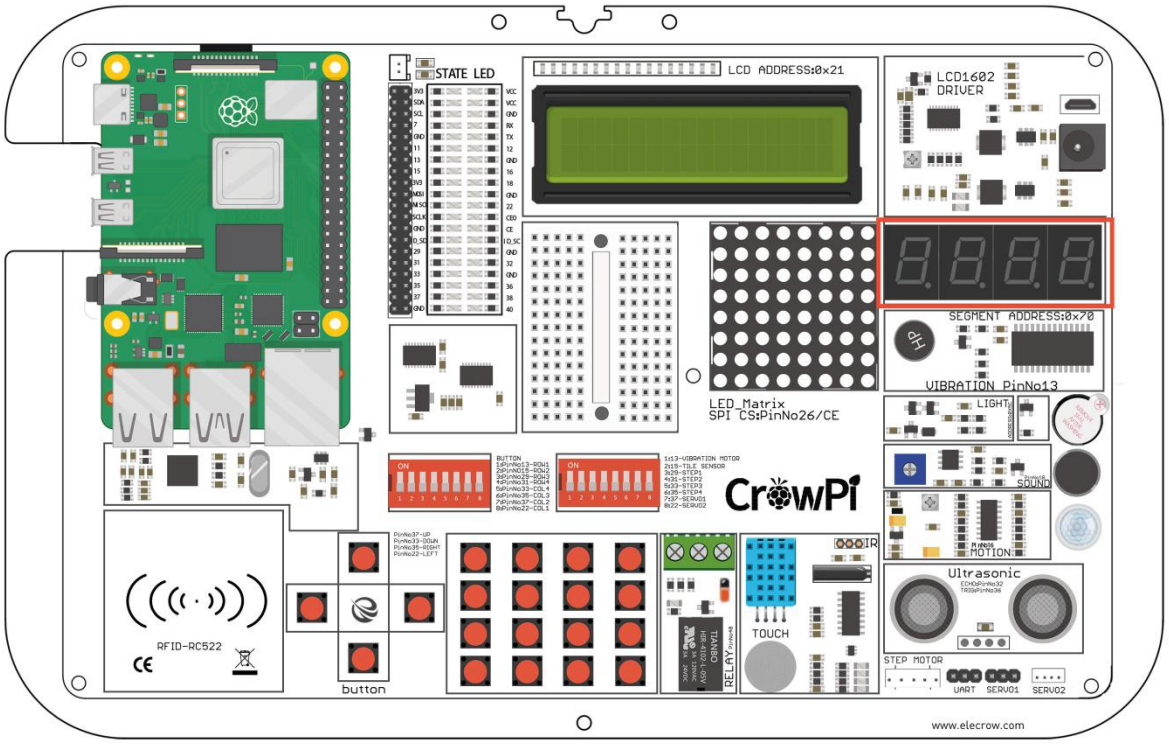
Requires switching modules using the switch

* No

Segment Display location on the CrowPi

The segment LED is located right on top of the vibration sensor and beside the Matrix LED. When turned off it can be easily recognized by its display of four blank numbers, which appear as dull "8s"…

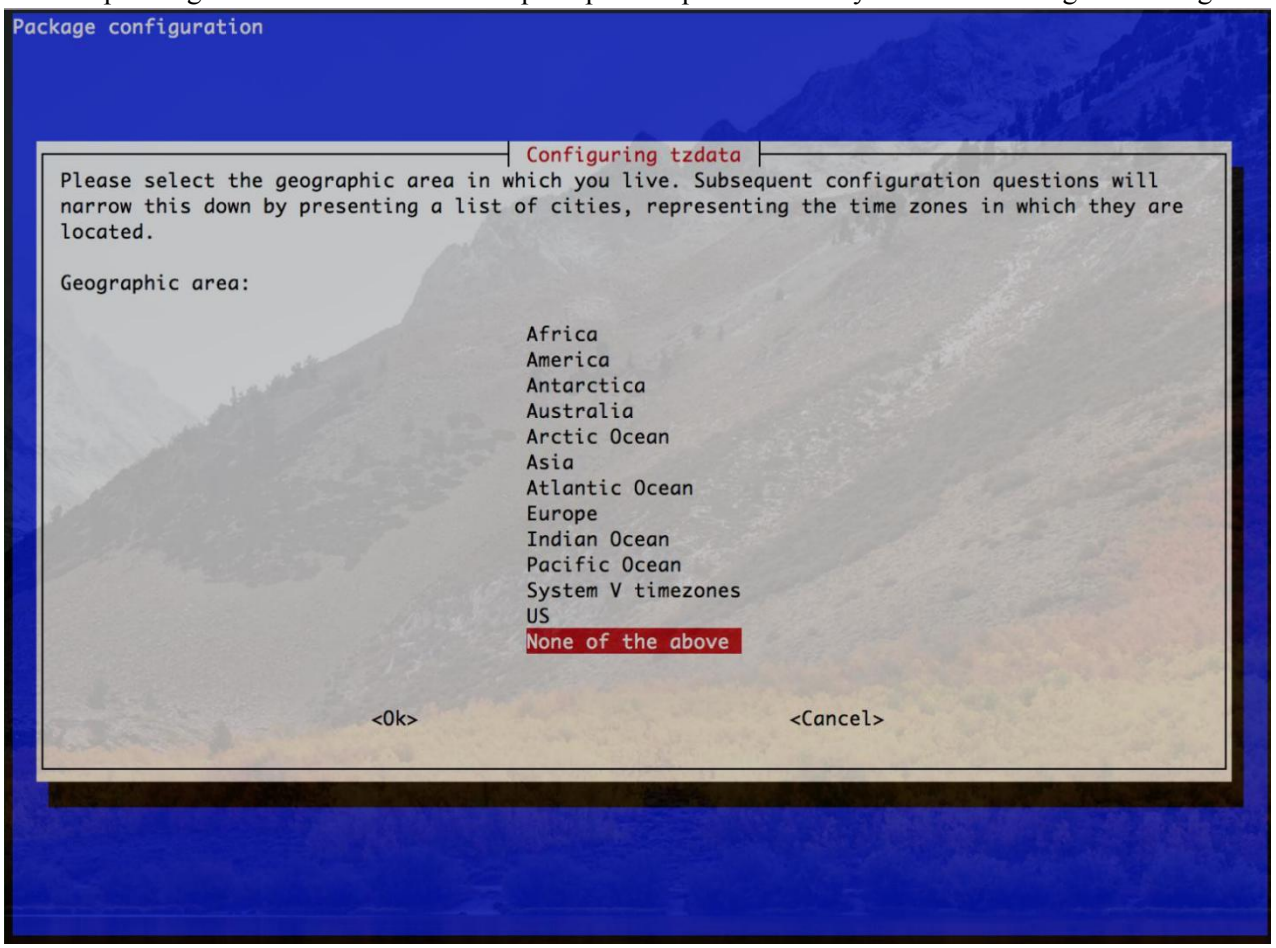
As soon as you use the Segment LED module, the dark color will turn into shiny and bright Red, and you will be able to control what data it displays!



Adjusting The Raspberry Pi System timezone

In our example we'll illustrate how to show the time using the segment display, in order to do that, `pi@raspberrypi:~/Desktop/CrowPi $ sudo dpkg-reconfigure tzdata` first we'll need to have the right time configured in the system, otherwise it will display the wrong time. We'll set it by opening the "Terminal" and running the following command: "sudo dpkg-reconfigure tzdata"

After pressing enter a new window will open up with options for many countries and regions. Navigate using



the keyboard arrows for the country that is suitable for your time zone.

After choosing the right country / region - navigate to the <OK> button and press enter to accept the new configuration - that's it!

You're ready to go with your clock which is now configured to the right timezone.

Working with the Segment Display

In our example we'll demonstrate a clock.

We'll use the time and date time modules to get the Raspberry Pi system time, we'll set the segment I2C address to 70, and then we'll set the current time of the system on the segment display using segment.write_display(), we use the function set_digit() with the number of the digit 0,1,2,3 to set the location where we want to show the number.

Try to play around with this clock example and see what you can make by yourself!

```
#!/usr/bin/python

import time
import datetime

from Adafruit_LED_Backpack import SevenSegment

# ===========================================================================
# Clock Example
# ===========================================================================
segment = SevenSegment.SevenSegment(address=0x70)

# Initialize the display. Must be called once before using the display.
segment.begin()

print "Press CTRL+Z to exit"

# Continually update the time on a 4 char, 7-segment display
while(True):
  now = datetime.datetime.now()
  hour = now.hour
  minute = now.minute
  second = now.second

  segment.clear()
  # Set hours
  segment.set_digit(0, int(hour / 10))     # Tens
  segment.set_digit(1, hour % 10)          # Ones
  # Set minutes
  segment.set_digit(2, int(minute / 10))   # Tens
  segment.set_digit(3, minute % 10)        # Ones
  # Toggle colon
  segment.set_colon(second % 2)              # Toggle colon at 1Hz

  # Write the display buffer to the hardware.  This must be called to
  # update the actual display LEDs.
  segment.write_display()

  # Wait a quarter second (less than 1 second to prevent colon blinking getting$
  time.sleep(0.25)
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  segment.py
```

Lesson 16

Detecting touch using the Touch Sensor

The touch sensor is pretty useful when it comes to buttons' functionality.

Sometimes you are not willing or able to push a regular button, but you feel like carrying out a process via a touch movement the same as with your mobile phone screen or your iPad - the touch sensor was made for this purpose exactly.

The touch sensor comes in handy in games like who can touch the pad first, or when you want to activate something by touch and not by pressing a button.

Many products in the market use touch instead of button pressing so learning how to use that type of sensor can definitely be practical!

What will you learn

At the end of this lesson you'll be able to:

* Use and detect a touch over the touch sensor surface

What will you need

* CrowPi Board after initial installation

Requires switching modules using the switch

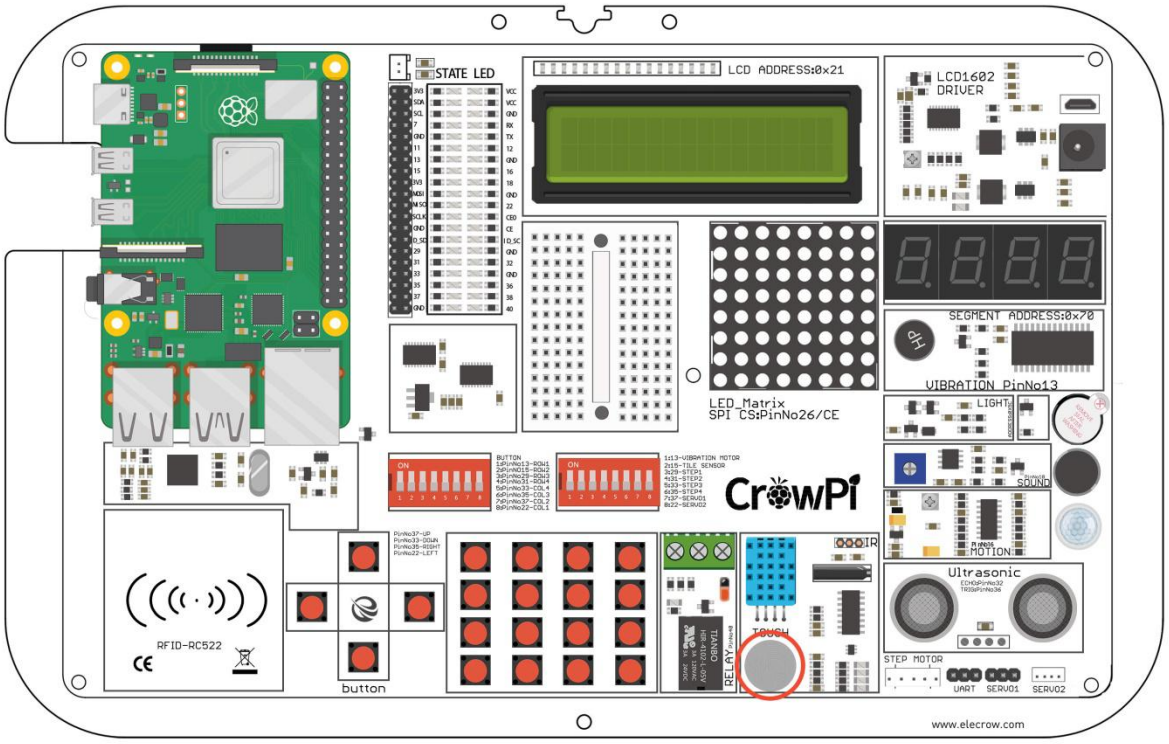
* No

Touch Sensor location on the CrowPi

The touch sensor is located right under the DH11 sensor and next to the relay.

This good location on the CrowPi allows us to easily access this convenient element.

Make sure to keep your hands off the sensor when not in use to avoid false positive alerts!



Working with the touch sensor

The touch sensor operates like any other button module with one difference that instead of pushing or clicking - we touch!

By touching the touch sensor the module will close a circuit which will indicate GPIO Input as HIGH, when you release the finger from the sensor the GPIO will go back to low indicating that currently nothing is touching the sensor.

The touch sensor uses the GPIO BOARD 11 pin.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# http://elecrow.com/

import RPi.GPIO as GPIO
import time

# define touch pin
touch_pin = 17

# set board mode to GPIO.BCM
GPIO.setmode(GPIO.BCM)

# set GPIO pin to INPUT
GPIO.setup(touch_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

try:
    while True:
        # check if touch detected
        if(GPIO.input(touch_pin)):
            print('Touch Detected')
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  touch.py
```

Lesson 17

Detecting tilt using the Tilt Sensor.

The tilt sensor is another one of my favorite sensors on the CrowPi.

The tilt sensor allows us to detect a tilt to either right or left, it can be extremely useful in scenarios where you want to know if the surface is straight or tilted and also to which side it's tilted if at all.

Tilt sensors are used in robotics and other industries to make sure things are kept straight, what kind of project would you choose to do with it?

What will you learn

At the end of this lesson you'll be able to:

* Control the tilt sensor and recognize a right side or left side tilt.

What will you need

* CrowPi Board after initial installation

Requires switching modules using the switch

* **Yes**, the right switch - pin number 2 - switch it on by turning it UP (refer to page number 5 if you forgot how to switch the sensors)

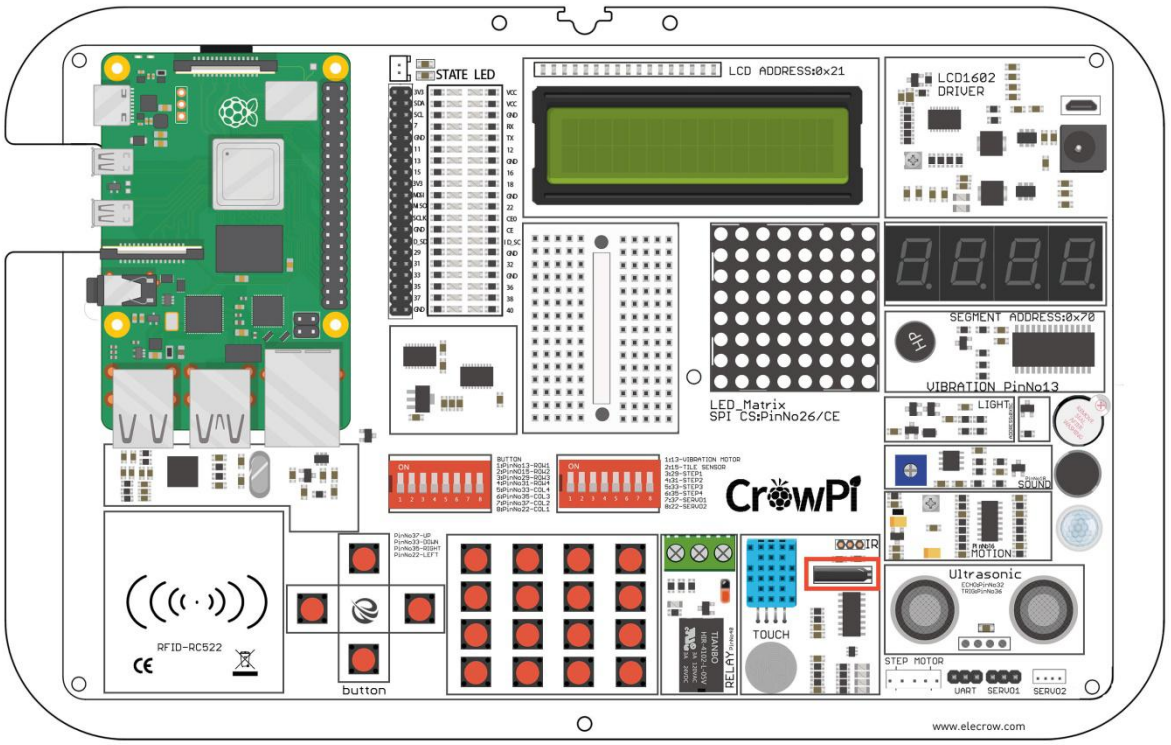
Tilt Sensor location on the CrowPi

The tilt sensor is a small long black sensor located next to the DH11 sensor and the Ultrasonic.

It can be easily recognized by the sound it makes when you shake the board to the sides, like a small ball is rolling from side to side …

Sometimes you might confuse the sound as if something broken inside the CrowPi but let us assure you that it's perfectly normal!

If the tilt sensor doesn't make any sound when tilted to right or left, it's something we should probably worry about.



Working with the tilt sensor

Working with the tilt sensor is fairly easy. When the tilt sensor tilted to the left side, it will activate a circuit which will send an INPUT signal of GPIO HIGH.

When the sensor is tilted to the right side, the circuit will open and the INPUT would be GPIO LOW.

That way we can use this data and display if the tilt is to the left side or the right side!

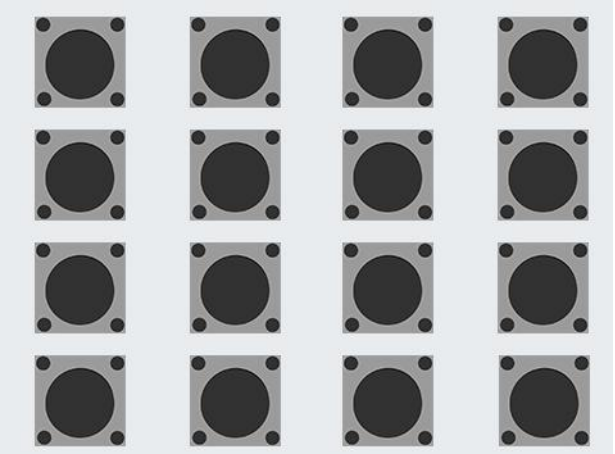
```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# http://elecrow.com/

import time
import RPi.GPIO as GPIO

# define tilt pin
tilt_pin = 22

# set GPIO mode to GPIO.BCM
GPIO.setmode(GPIO.BCM)
# set puin as input
GPIO.setup(tilt_pin, GPIO.IN)

try:
    while True:
        # positive is tilt to left negative is tilt to right
        if GPIO.input(tilt_pin):
            print("[-] Left Tilt")
        else:
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  tilt.py
```

Lesson 18

Using and controlling the Button Matrix.



The matrix button is a module with 16 independent buttons on the CrowPi board, and it can be used for a wide range of projects like secret code keypad, a memory game, or just controlling anything you want!

The full potential of the buttons allows you to make anything you want!

What will you learn

At the end of this lesson you'll be able to:

* Understand how a button matrix works and how to control it through the GPIO pins

What will you need

* CrowPi Board after initial installation

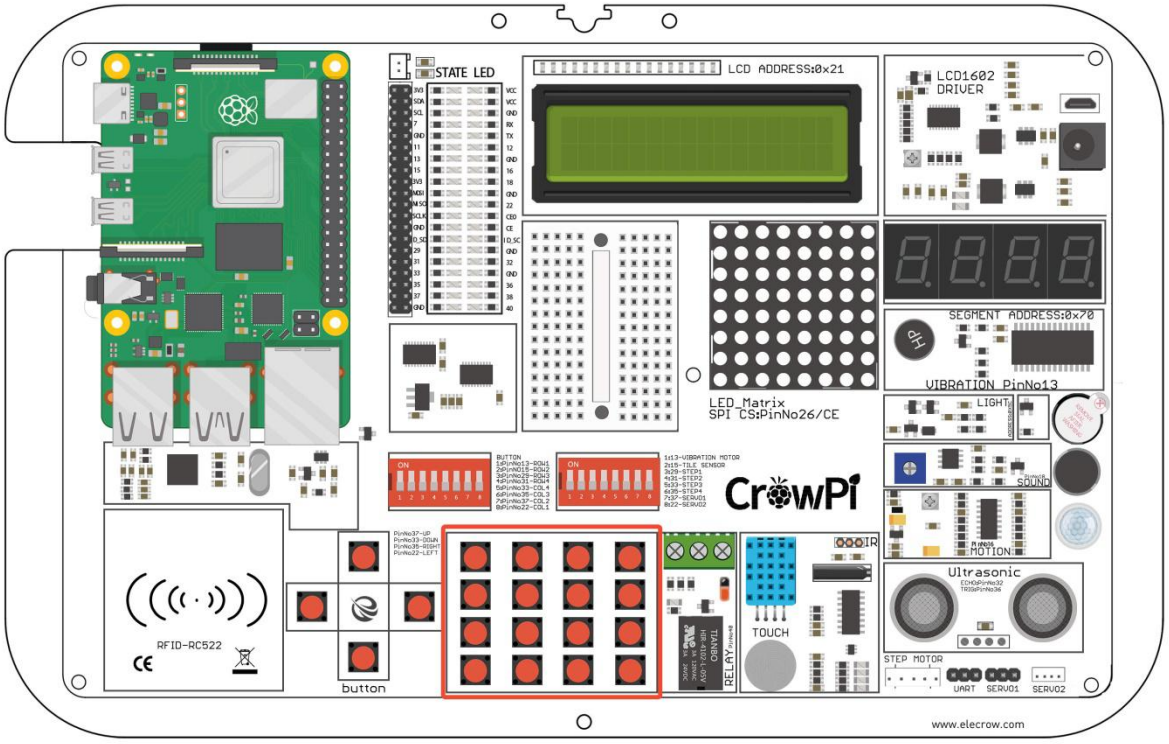
Requires switching modules using the switch

* **Yes**, the left switch - turn ALL the pins ON by turning them UP (refer to page number 5 if you forgot how to switch the sensors)

Button Matrix location on the CrowPi

The button Matrix is located near the independent buttons and the relay, and can easily be detected by the “Matrix” or "grid" shape of many buttons together, don’t accidentally think that all those buttons are separate modules ..even though they might look like they are separate - they all work as one!

The buttons' great location allows for easy access with the tip of the fingers while still being able to use and see the other sensors across the CrowPi board.



Working with Button Matrix

The button matrix is built into columns and rows of 4 each, we configure the matrix rows and columns with their GPIO pins and initialize the object called ButtonMatrix() into buttons variable.

Afterwards, we are able to go for each button on the buttons and detect if it's been clicked.

In our example if the button has been pressed we activate activateButton(), which will print the number of the button which has been pressed. You can modify this module to be able to do anything you could imagine if the button has been pressed or not!

```
44  def main():
45
46      # initial the button matrix
47      buttons = ButtonMatrix()
48      try:
49          while(True):
50              for j in range(len(buttons.columnPins)):
51                  # set each output pin to low
52                  GPIO.output(buttons.columnPins[j],0)
53                  for i in range(len(buttons.rowPins)):
54                      if GPIO.input(buttons.rowPins[i]) == 0:
55                          # button pressed, activate it
56                          buttons.activateButton(i,j)
57                          # do nothing while button is being held down
58                          while(buttons.buttonHeldDown(i)):
59                              pass
60                  # return each output pin to high
61                  GPIO.output(buttons.columnPins[j],1)
62      except KeyboardInterrupt:
63          GPIO.cleanup()
64
65  if __name__ == "__main__":
66      main()
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  button_matrix.py
```

Lesson 19

Controlling and using the IR sensor

In this lesson we'll learn how to use the IR receiver and receive IR (infra red) codes by air from our mini IR remote.

Using this method is extremely useful as we can set different actions for different buttons over the IR remote. For example: we can turn on different LEDs with each button press, or maybe control the servo movements! How about … turning off the alarm? Anything is possible after learning this lesson.

What will you learn

At the end of this lesson you'll be able to:

* Control the IR receiver and send signals via the IR remote

What will you need

* CrowPi Board after initial installation
* IR remote (comes with the CrowPi Kit)

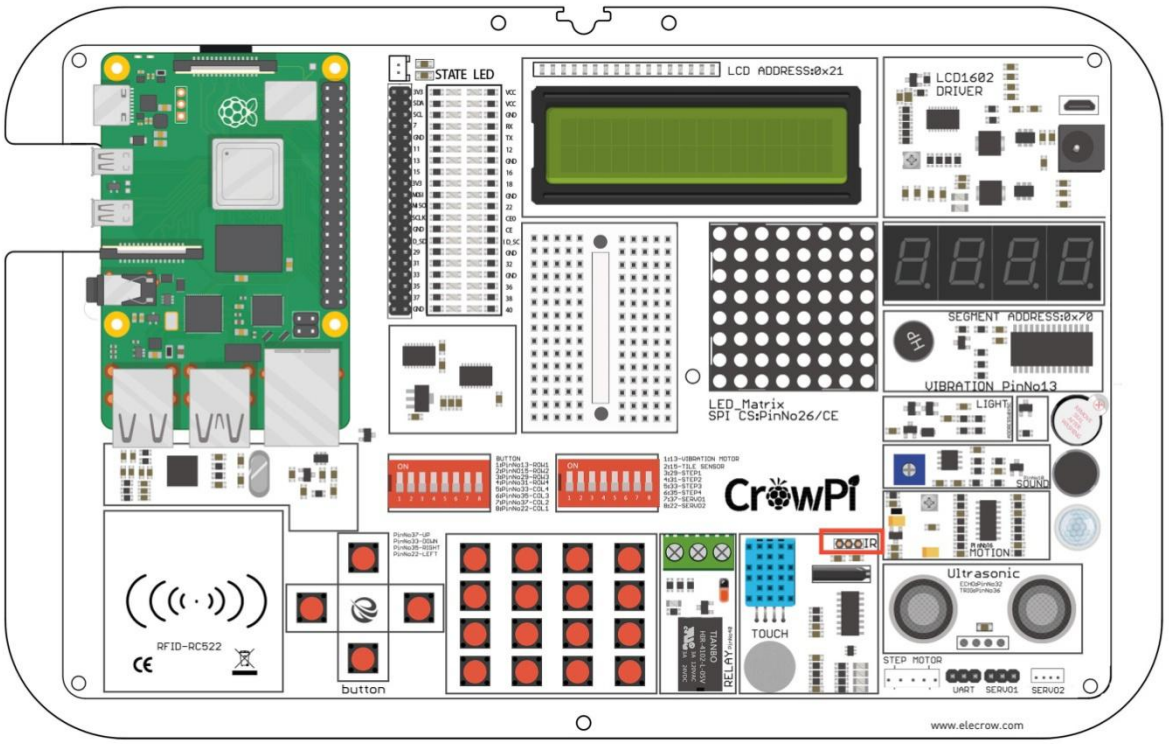
Requires switching modules using the switch

* No

IR sensor location on the CrowPi

The interface of the IR receiver is located on the right side of the DH11 sensor and right under the CrowPi logo. You need to plug the IR receiver into that interface to receive the codes from our remote. Note that the arc surface of the IR receiver should face us(that is face to the tilt sensor but not the LED matrix)

We'll also need the IR remote that comes included with the CrowPi kit, make sure to get it out and ready for our lesson!



Working with IR Receiver

Working with the IR receiver isn't as difficult as it sounds.

The code is a little bit long but let's understand how it works, we need to count the pulses that sent from the transmitter to our receiver, by counting the pulses we can determine what button was pressed after we covert the pulses into byte code.

Previously we used the IR library Python-LIRC but now it seems to be unsupported as we moved from python2 to python3.

The code is longer than the picture below, in the entire code we will perform the main code and then determine based on the bytes what button was pressed on the controller.

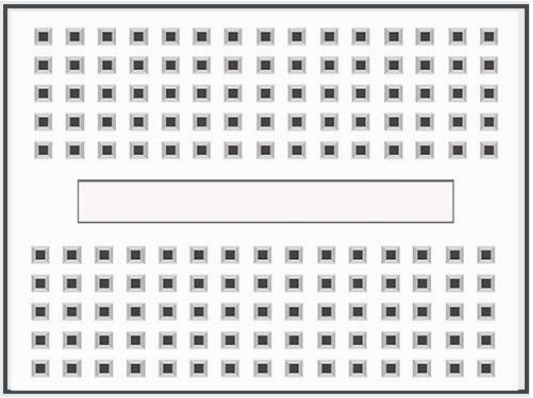
```
while True:
    if GPIO.input(PIN) == 0:
        count = 0
        while GPIO.input(PIN) == 0 and count < 200:
            count += 1
            time.sleep(0.00006)

        count = 0
        while GPIO.input(PIN) == 1 and count < 80:
            count += 1
            time.sleep(0.00006)

        idx = 0
        cnt = 0
        data = [0,0,0,0]
        for i in range(0,32):
            count = 0
            while GPIO.input(PIN) == 0 and count < 15:
                count += 1
                time.sleep(0.00006)

            count = 0
            while GPIO.input(PIN) == 1 and count < 40:
                count += 1
                time.sleep(0.00006)

            if count > 8:
                data[idx] |= 1<<cnt
            if cnt == 7:
                cnt = 0
                idx += 1
            else:
                cnt += 1
        if data[0]+data[1] == 0xFF and data[2]+data[3] == 0xFF:
                print("Get the key: 0x%02x" %data[2])
                exec_cmd(data[2])
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  IR.py
```

Lesson 20

Making your own custom circuit using the Bread Board



Breadboard is extremely useful part of the CrowPi, which will allow us to make our own custom circuits and functions.

After we've learned how to use all those sensors, it's time for us to learn how to make our own! In this lesson you'll create your first custom circuit using a blinking LED example!

What will you learn

At the end of this lesson you'll be able to:

* Create your custom circuit on top of the breadboard, and make blinking LED.

What will you need

* CrowPi Board after initial installation
* Jumpers
* 3mm / 5mmLED's
* Resistors

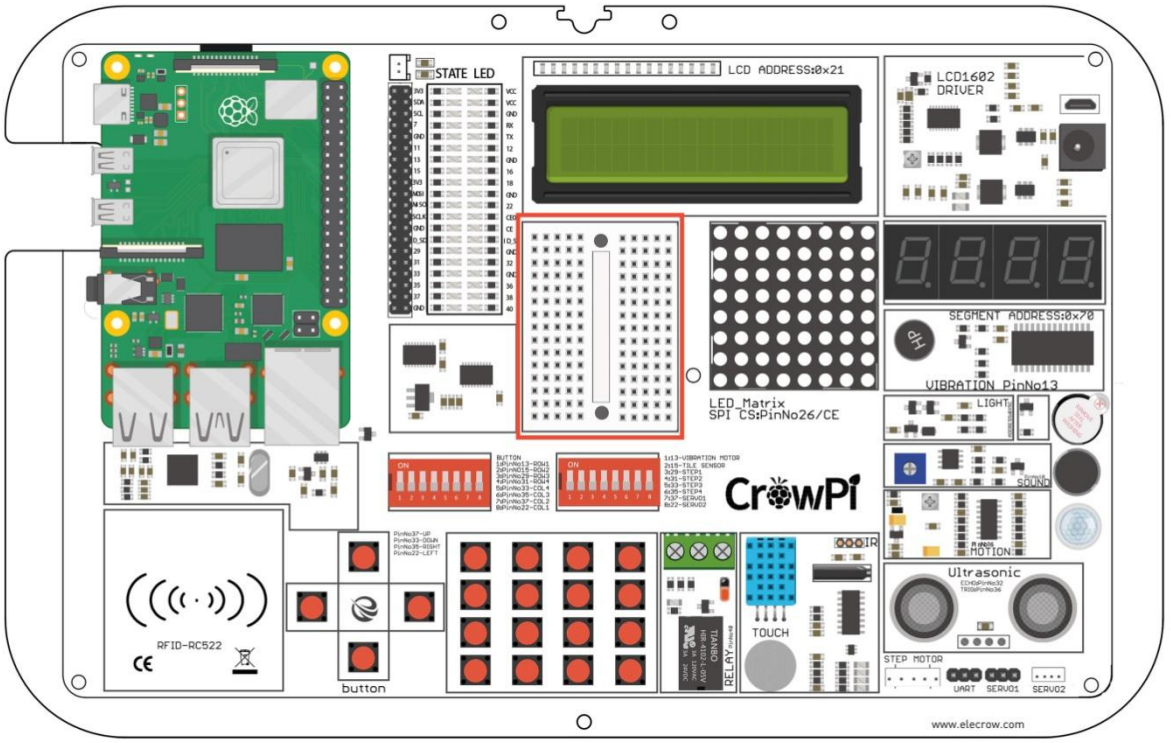
Requires switching modules using the switch

* **Yes**, We'll use the servo pins to make custom circuits, pins numbers 7,9 turn them ON by switching it UP (refer to page number 5 if you forgot how to switch the sensors)

Bread Board location on the CrowPi

The breadboard is located right in the middle of the CrowPi. It looks like small white board with many holes inside which are used to create custom circuits.

On the other hand, we'll also use the servo's interfaces which we showed before in our tutorials - specifically SERVO1 Interface for making our custom circuit.



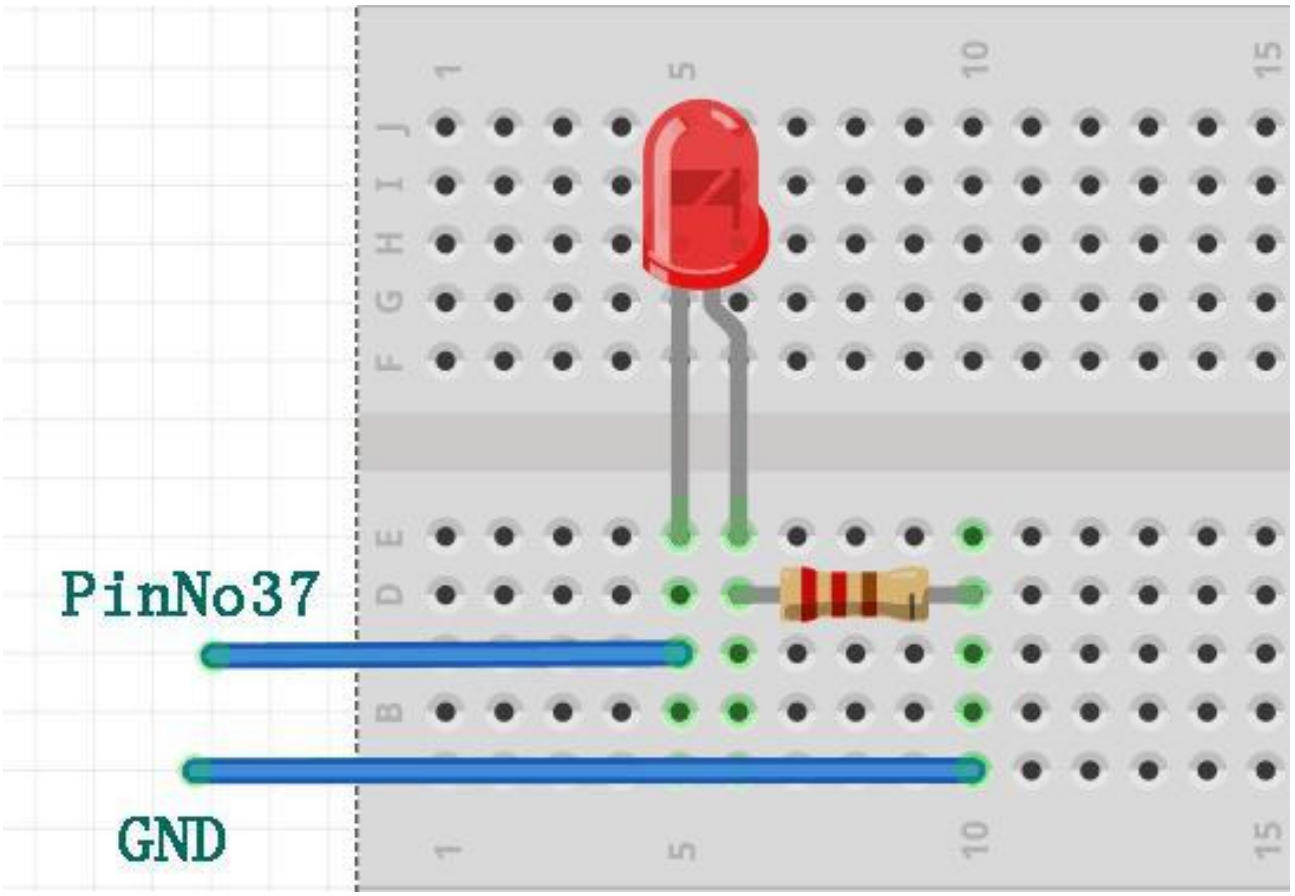
The “Blinking LED” circuit

We’ll be making a custom circuit that's function will literally be a blinking LED.

Before we start make sure to switch the right switch properly!

In order to do that we’ll need to use GPIO as output and GND as we did before in our previous lessons. We’ll use the servo interface (specifically SERVO1 Interface) on GPIO 37.

The first thing will be to create the custom circuit. Take a look at the following picture:



You can refer to this picture to make your circuit on the

breadboard. Don’t forget that PinNo37 is located on the SERVO1

Interface GPIO port. The GND is located on the SERVO1

Interface as well on the GND port.

We’ll need to take one resistor that comes with the CrowPi package and wire it to the negative side of the LED **(the negative side of the LED is the shorter leg between the 2 LED legs).**

From the other side of the resistor we’ll wire it directly with jumper to the GND pin on the SERVO1 interface.

The other LED leg would be the positive pin which we’ll wire directly into the GPIO37 pin on SERVO1 to be able to control it and make it blink!

The final result should look similar to this:

Working with the blinking LED

After we've successfully made the blinking LED circuit it's time to make the code that will control it.

In our code, we'll send GPIO.HIGH to our SERVO GPIO PIN which has the LED connected to it, wait 0.2 seconds, and turn it off with GPIO.LOW.

We'll repeat that infinity times which will make the LED turn on and off and basically blink!

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# http://elecrow.com/

import time
import RPi.GPIO as GPIO

# define LED pin
led_pin = 26

# set GPIO mode to GPIO.BCM
GPIO.setmode(GPIO.BCM)
# set puin as input
GPIO.setup(led_pin, GPIO.OUT)

try:
    while True:
        # turn on LED
        GPIO.output(led_pin, GPIO.HIGH)
        # Wait half a second
        time.sleep(0.2)
```

Execute the following commands and try it by yourself:

```
cd  Desktop/CrowPi/Examples/
sudo  python3  blinking_led.py
```

Taking a picture using the Raspberry Pi camera

How can we use CrowPi camera? Don’t worry, it will be easy if you follow the following step.

What will you learn

At the end of this lesson you'll be able to:

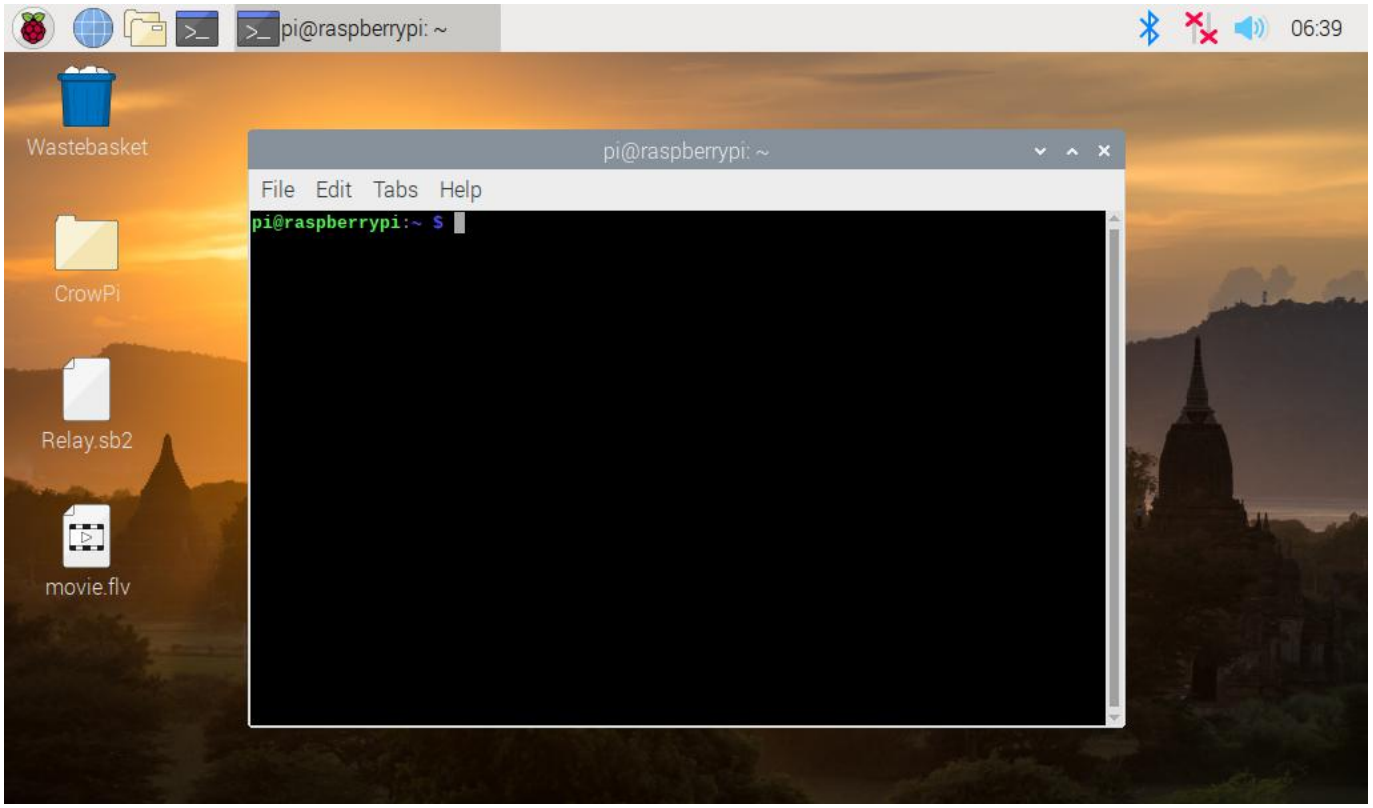
* Take picture using the CrowPi Raspberry Pi camera

What will you need

* CrowPi Board after initial installation

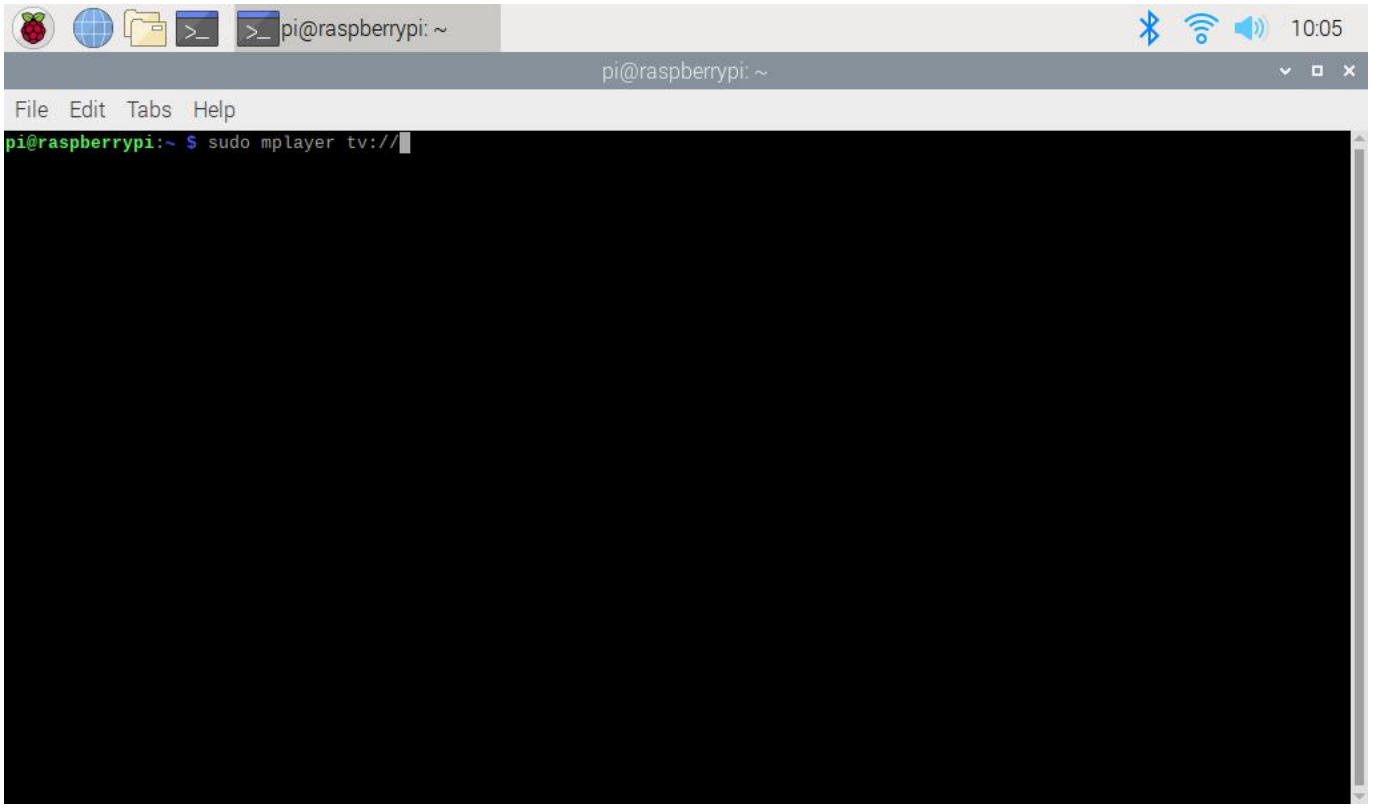
STEPS

All the software required for camera photography is already installed when we ship CrowPi. So you just need to open the CrowPi’s terminal and type the command below.

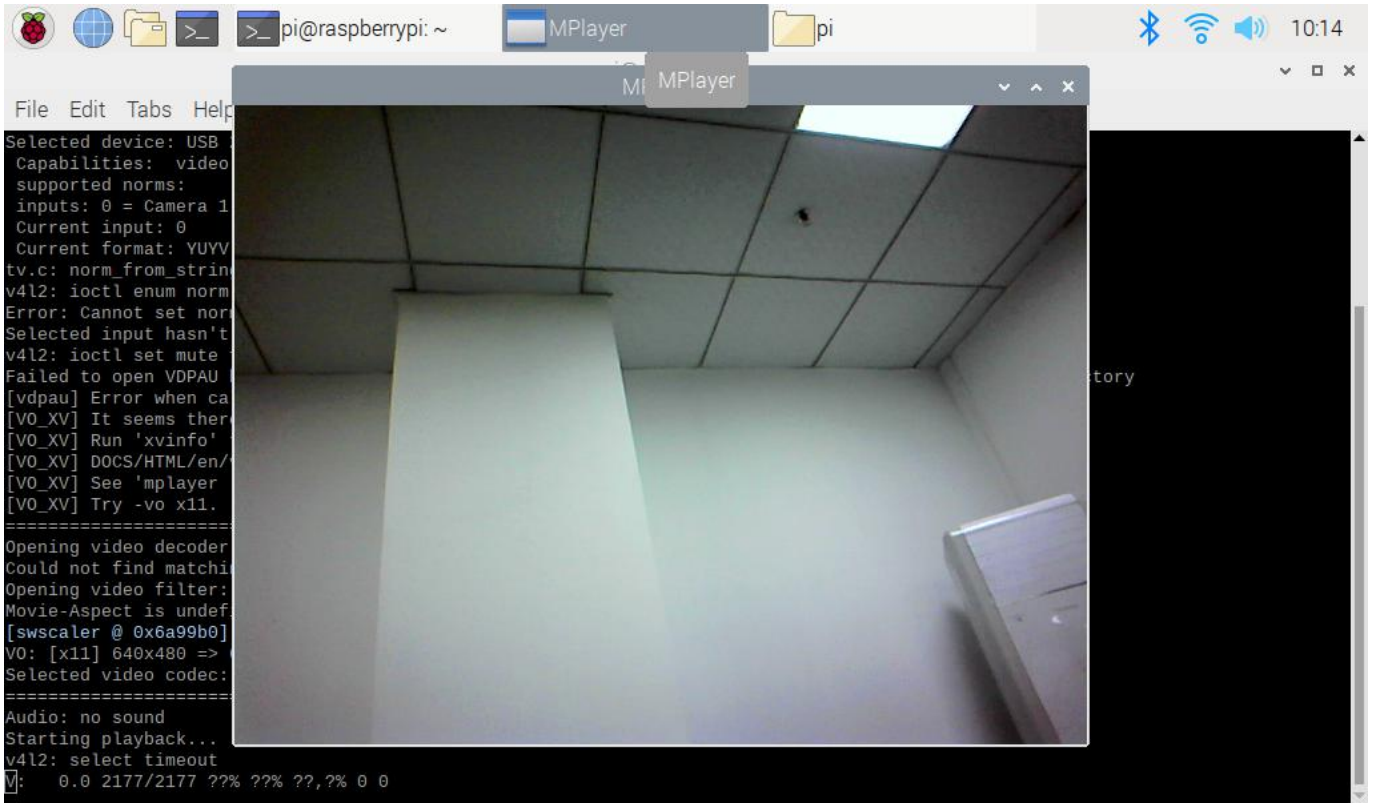


1. Open the camera

   sudo mplayer tv://

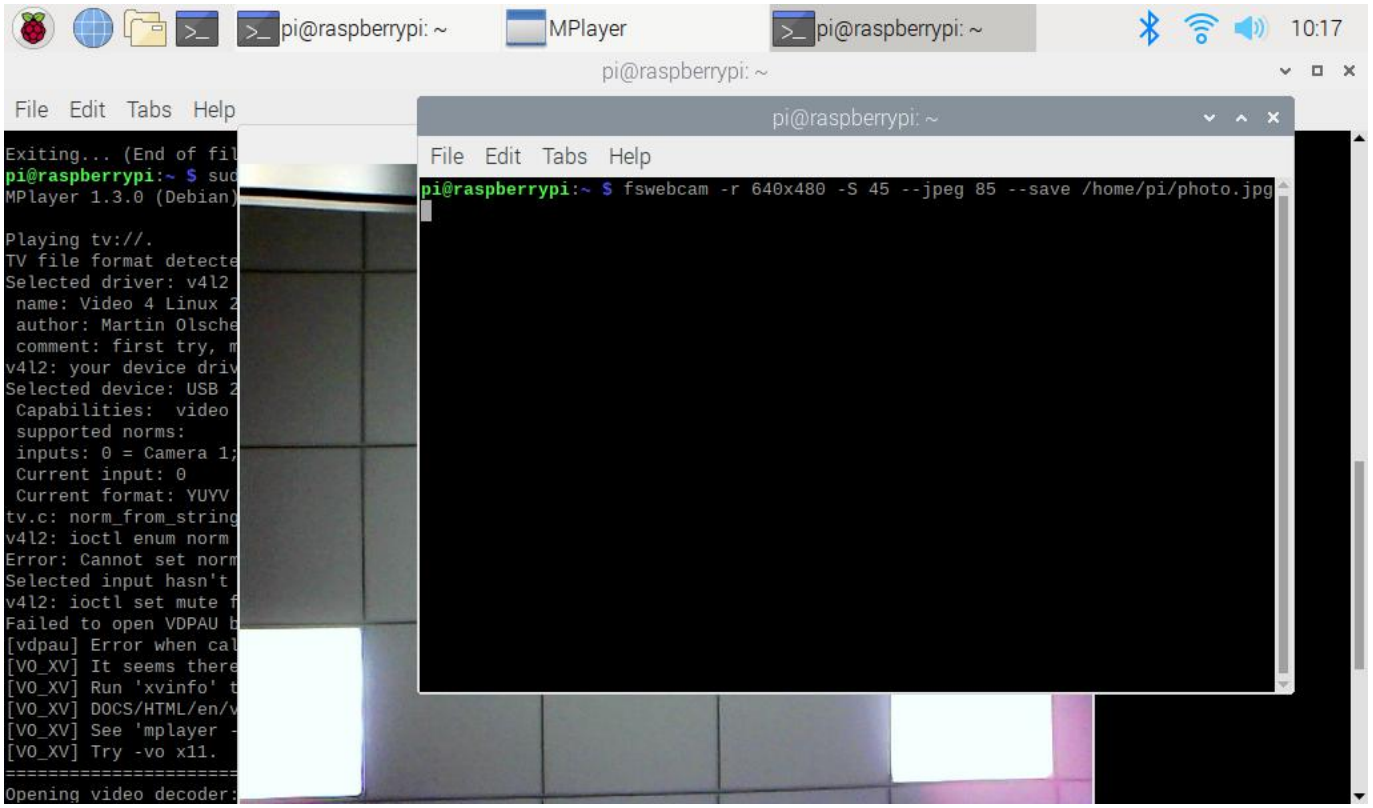


Then you will see the camera window



2. Take pictures

fswebcam -r 640x480 -S 45 --jpeg 85 --save /home/pi/photo.jpg



The name of the picture is photo.jpg and the picture will be saved in the /home/pi directory.

Press enter and then you can find a picture named photo.jpg in the /home/pi directory.