OLD DOMINION
UNIVERSITY

ODU BLAST:
Hands-on Sensing, Measurement, and Programming

- Dr. Murat Kuzlu
- Department of Engineering Technology

Let's take a look to a very useful device, the CrowPi

We need this device for hands-on activities

What is the CrowPi?

Module List:

1. ABS Plastic Case
2. Camera
3. 7inch HDMI Touch Screen
4. Power Circuit
5. LCD Module(MCP23008)
6. Segment LED (HT16K33)
7. Vibration Motor
8. Matrix LED(MAX7219)
9. Light sensor(BH1750)
10. Buzzer
11. Sound Sensor
12. PIR Motion Sensor(LH1778)
13. Ultrasonic Sensor
14. Servo Inteface
15. UART
16. Step Motor Interface
17. Tilt Sensor(SW-200D)
18. IR Sensor
19. Touch Sensor(TTP223)
20. Temperature and Humidity Sensor (DH11)
21. Relay
22. Matrix Buttons
23. Independent Buttons
24. RFID Module(MFRC522)
25. Switches
26. Bread board
27. Raspberry Pi
28. GPIO LED Indicator
29. Acrylic board
30. Microphone

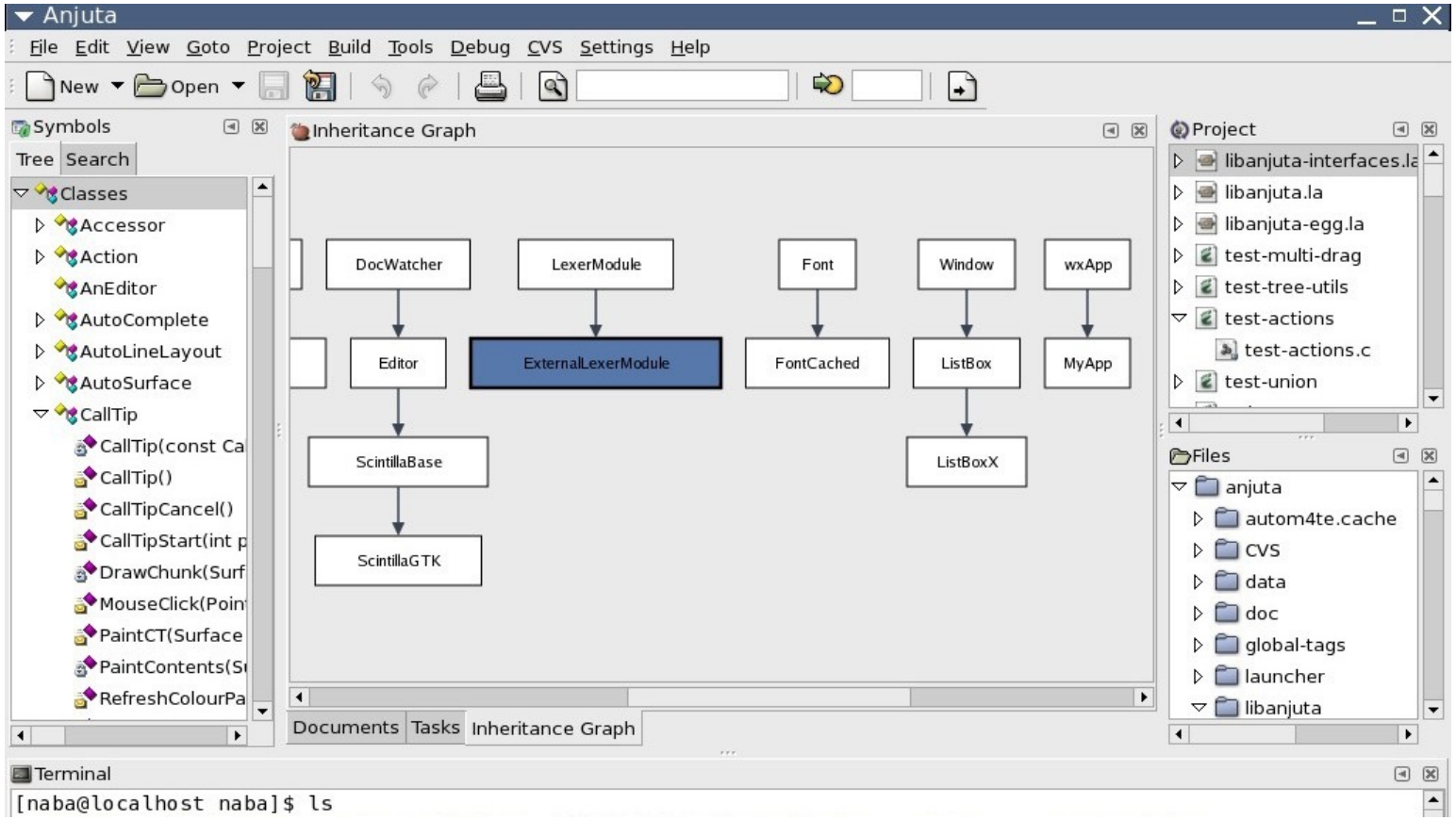
- It is an **educational kit**, that is a one-of-a-kind Raspberry Pi Development board
- The kit can be used as a laptop, gaming console, IoT project, for studying device, prototype, and much more.
- The kit has many features, and it was specially crafted for those who are passionate about **STEM education**.

Why the CrowPi over a regular Raspberry Pi?

- The CrowPi is easy to set up and it's comes with a Raspberry Pi inside of both kits.
- There are 32 extra devices that come in this kit
- Durable, well-design case
- This kit eliminates the need for an individual's need to learn about programming and electronics at the same time because everything that you need is included.
- CrowPi is easy to power using a battery pack

Why the CrowPi over an Arduino?

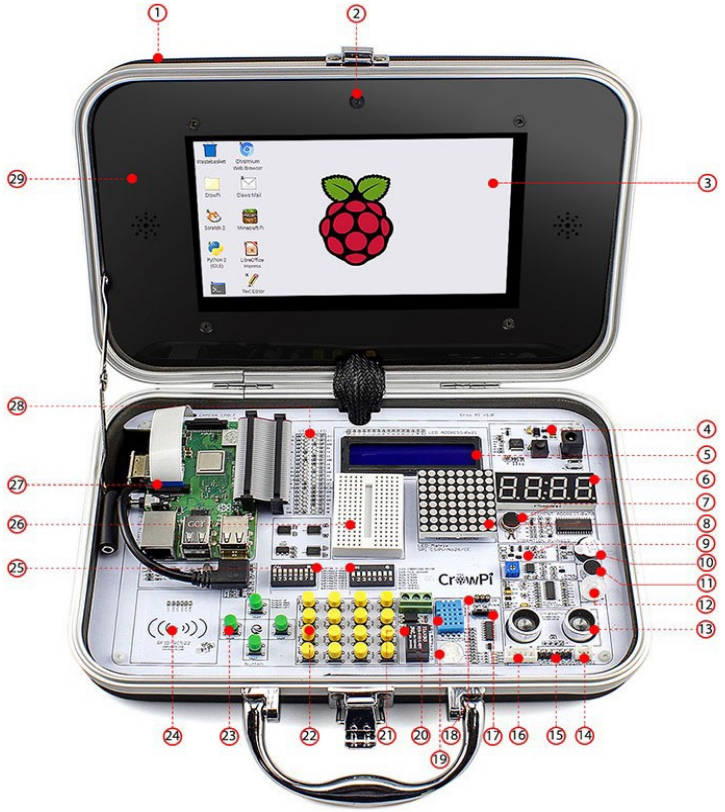
- Arduino is **microcontroller** board while raspberry pi is a **minicomputer**
- Raspberry Pi is good at software applications, while Arduino makes hardware projects simple
- Arduino uses Arduino, C/C++
- Raspberry Pi uses **Python** but C, C++ are pre-installed



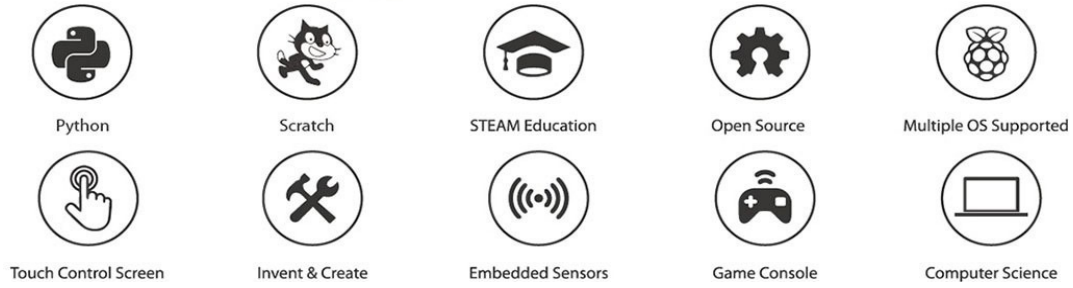
IDE's for the CrowPi

- **Integrated development environment (IDE)**
- BlueJ - java programming language
- Geany - C, C++, C#, Java, HTML, PHP, Python, Perl, Ruby, Erlang and even LaTeX
- Adafruit WebIDE - Python, Ruby, JavaScript
- AlgoIDE - C, C++, Python, Java, Smalltalk, Objective C, ActionScript
- Ninja IDE/Greenfoot IDE/Codeblock IDE
- Node-red

CrowPi



1. ABS Plastic Case
2. Camera
3. 7inch HDMI Touch Screen
4. Power Circuit
5. LCD Module(MCP23008)
6. Segment LED (HT16K33)
7. Vibration Motor
8. Matrix LED(MAX7219)
9. Light sensor(BH1750)
10. Buzzer
11. Sound Sensor
12. PIR Motion Sensor(LH1778)
13. Ultrasonic Sensor
14. Servo Inteface
15. UART
16. Step Motor Interface
17. Tilt Sensor(SW-200D)
18. IR Sensor
19. Touch Sensor(TTP223)
20. Temperature and Humidity Sensor (DH11)
21. Relay
22. Matrix Buttons
23. Independent Buttons
24. RFID Module(MFRC522)
25. Switches
26. Bread board
27. Raspberry Pi
28. GPIO LED Indicator
29. Acrylic board

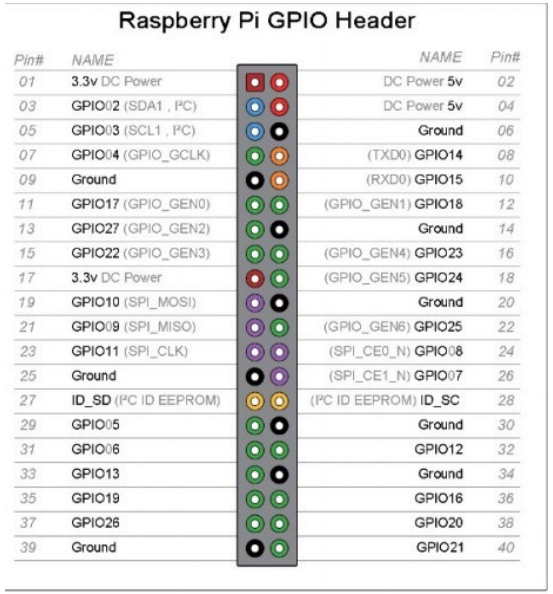


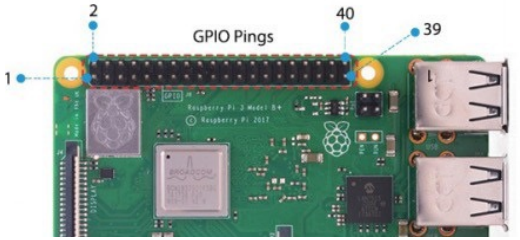
https://www.gadgetking.com/2018/10/24/diy-raspberry-pi-tech-kit/

Inputs/Outputs

Sensor Name	Control Pin	Control Method	Remarks
BUZZER	pin12/GPIO18	GPIO OUTPUT	
RELAY	pin40/GPIO21	GPIO OUTPUT	
SOUND SENSOR	pin18/GPIO24	GPIO INPUT	
TILT SWITCH	pin15/GPIO22	GPIO INPUT	UX5-2 ON
VIBRATION MOTOR	pin13/GPIO27	GPIO OUTPUT	UX5-1 ON
MOTION SENSOR	pin16/GPIO23	GPIO OUTPUT	
TOUCH SENSOR	pin11/GPIO17	GPIO INPUT	
STEPPER MOTOR	pin29/GPIO5 pin31/GPIO6 pin33/GPIO13 pin35/GPIO19	GPIO OUTPUT	UX5-3 ON UX5-4 ON UX5-5 ON UX5-6 ON
SERVO	pin22/GPIO25	GPIO OUTPUT	UX5-8 ON
IR RECEIVER	pin38/GPIO20	GPIO BOTH	
TEMPERATURE&HUMIDITY	pin7/GPIO4	GPIO INPUT	
ULTRASONIC SENSOR	ECHO:Pin32/GPIO12 TRIG:Pin36/GPIO16	GPIO INPUT	
LIGHT SENSOR	I2C	I2C	Address:0x5c
I2C LCD	I2C	I2C	Address:0x21
4 BITE SEGMENT	I2C	I2C	Address:0x70
LED MATRIX	SPI	SPI	CS: pin26/GPIO8
NFC MOUDLE	SPI	SPI	CS: pin24/GPIO7

INDEPENDENT BUTTON	UP: pin37/GPIO26 DOWN: pin33/GPIO13 RIGHT: pin35/GPIO19 LEFT:pin22/GPIO25	GPIO INPUT	UX1-5 ON UX1-6 ON UX1-7 ON
BUTTON ARRY	Row1: pin13/GPIO27 Row2: pin15/GPIO22 Row3: pin29/GPIO5 Row4: pin31/GPIO6 Col1: pin22/GPIO25 Col2: pin37/GPIO26 Col3: pin35/GPIO19 Col4: pin33/GPIO13	GPIO INPUT	UX1-1 ON UX1-2 ON UX1-3 ON UX1-4 ON UX1-5 ON UX1-6 ON UX1-7 ON UX1-8 ON





Basic Python and Linux usage

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.

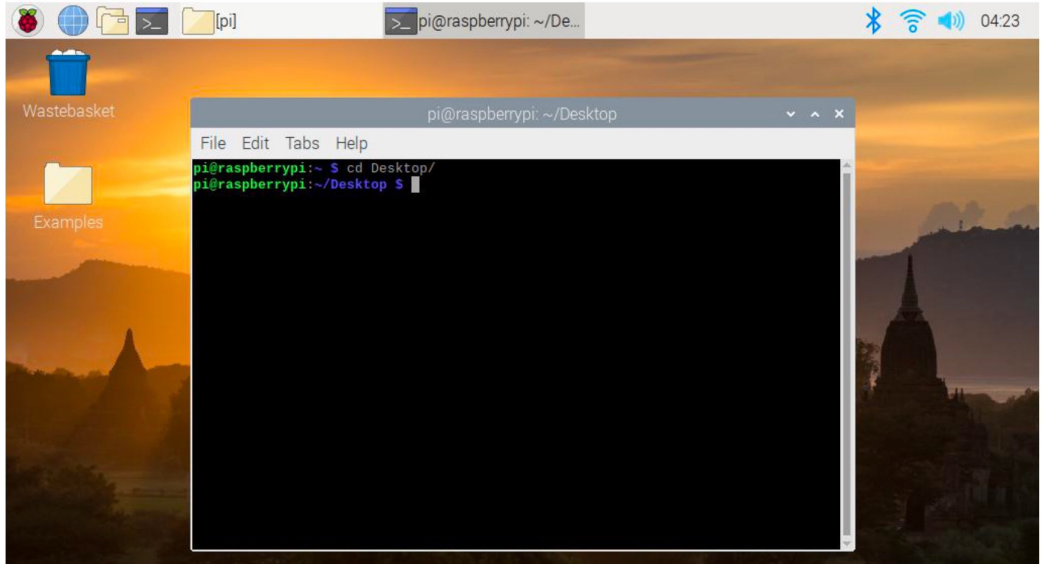
This incredibly useful way called "git cloning".

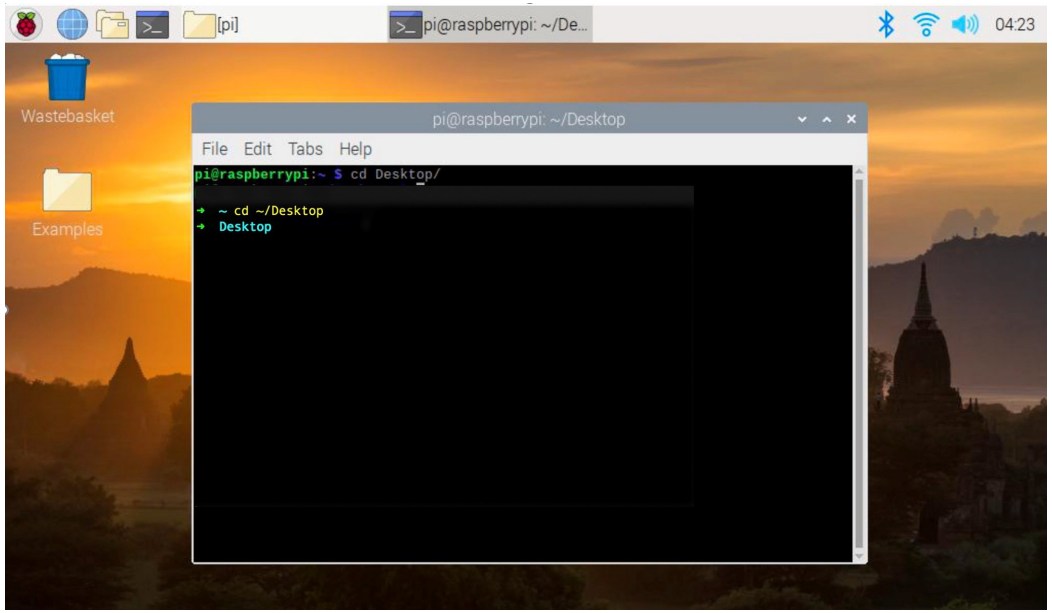
In simple terms: we will clone the GitHub directory where all the example scripts are located into our desktop environment, so we won't need to download every single script every time. And how are we going do that? Using the git clone command:

Cloning the git repository

1. First step

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. We need to change directory to desktop in terminal

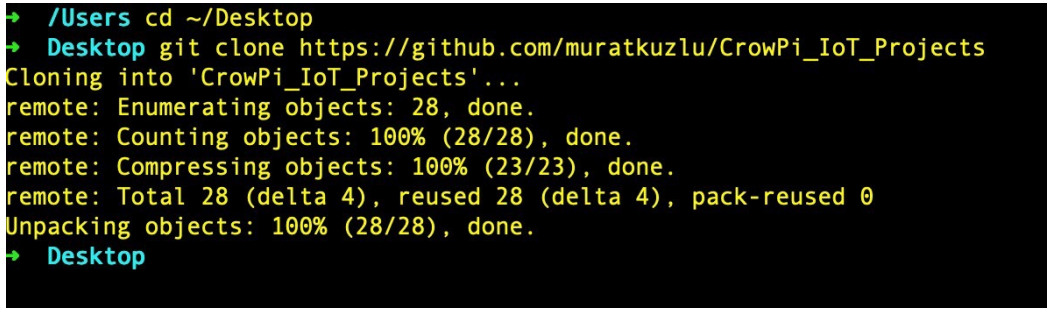
To do so you need to type in the terminal: *cd ~/Desktop*

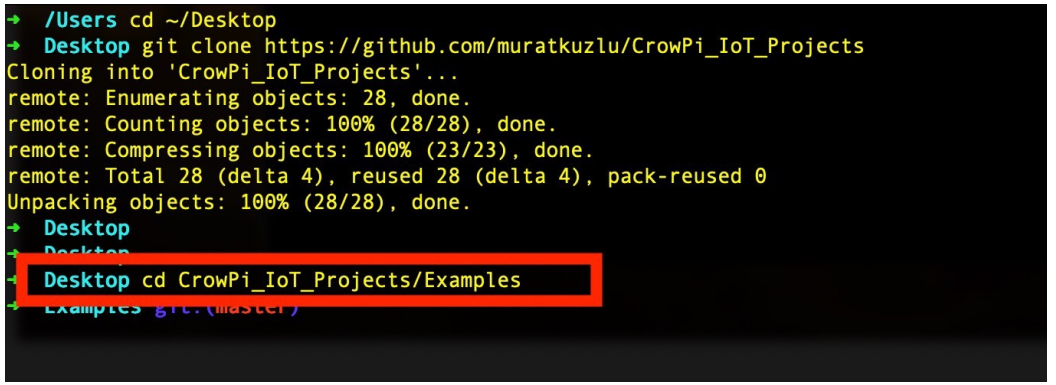
3. Press "Enter" on your keyboard. Now you should be in your desktop folder which is your desktop that you see.

4. Inside the terminal type the following command:

git clone
https://github.com/muratkuzlu/CrowPi_Python_Projects

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





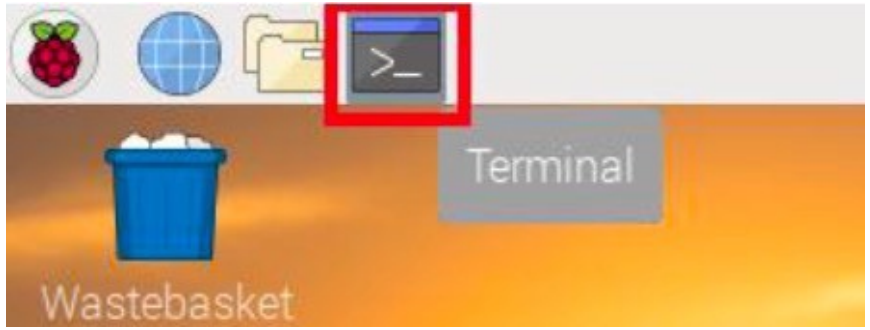
6. After cloning the git repository, it should show up on our desktop as a “CrowPi_Python_Projects” folder. Let’s go into that folder by “cd” command so we can use the scripts that are inside

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!**

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:

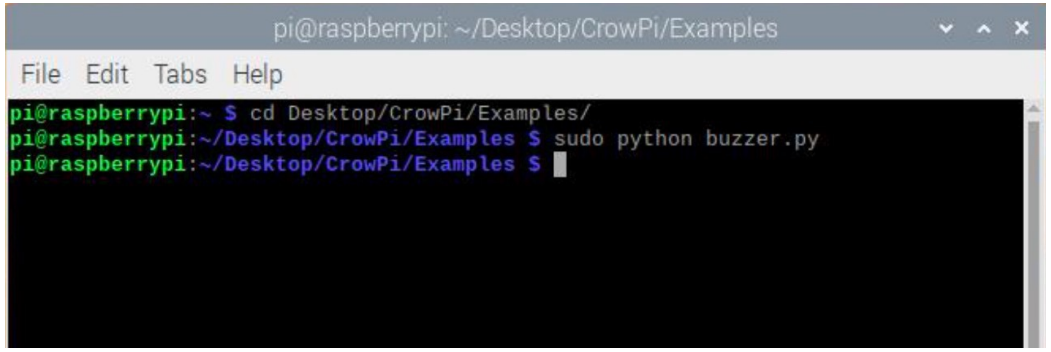
cd ~/Desktop/CrowPi_Python_Projects/Examples

Or

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”***



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!

Hands-on Activities

1-Using the buzzer as an alert notification

2- Controlling the LCD Display

3- Detect motion using the motion sensor

4- Detect room temperature and humidity using the **DHT11** sensor

5- Using the CrowPi Camera

1- Using the buzzer as an alert notification

Using the buzzer as an alert notification

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.

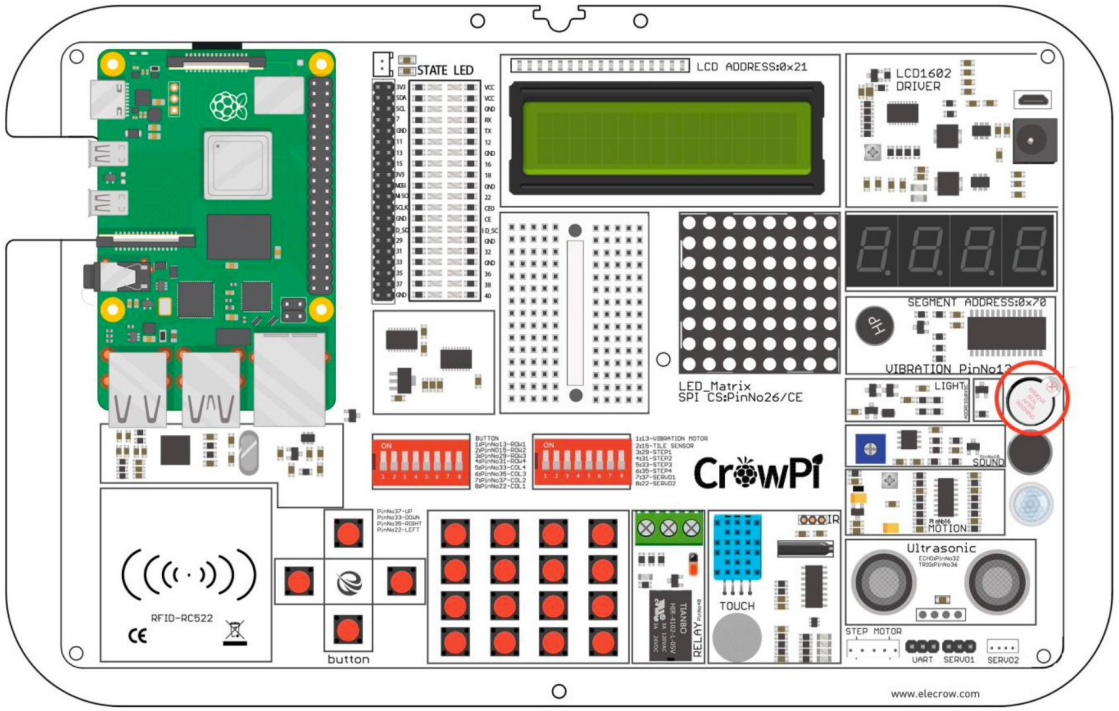
What will you learn

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

* Be able to control the buzzer module using GPIO output

Location of the buzzer on the CrowPi

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.

Implementation

Script

```
#Import RPi.GPIO library
import RPi.GPIO as GPIO
#Import time library
import time

#set buzzer pin
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()
```

Walk through the Python script shown on the left side

At first, we **import RPi.GPIO** library and **the time library for sleeping**. Then we **configure the buzzer at pin 18**, 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.

Execute the following commands and try it by yourself:

1- //Download the code from repository: ***(Skip this part if you already downloaded)***
https://github.com/muratkuzlu/CrowPi_Python_Projects/blob/master/Examples

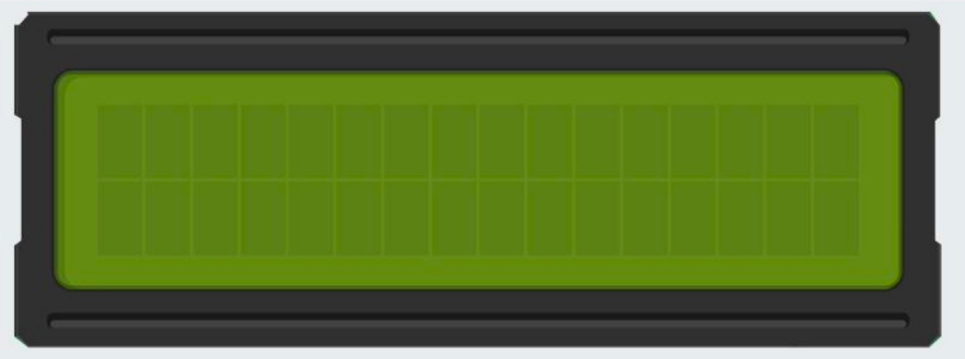
2- //open a terminal and change the directory with following command
cd ~/Desktop/CrowPi_Python_Projects-master/Examples
or
cd ~/Desktop/CrowPi/Examples

3- //type following command to run Python script **sudo python3 buzzer.py**

2- Controlling the LCD Display

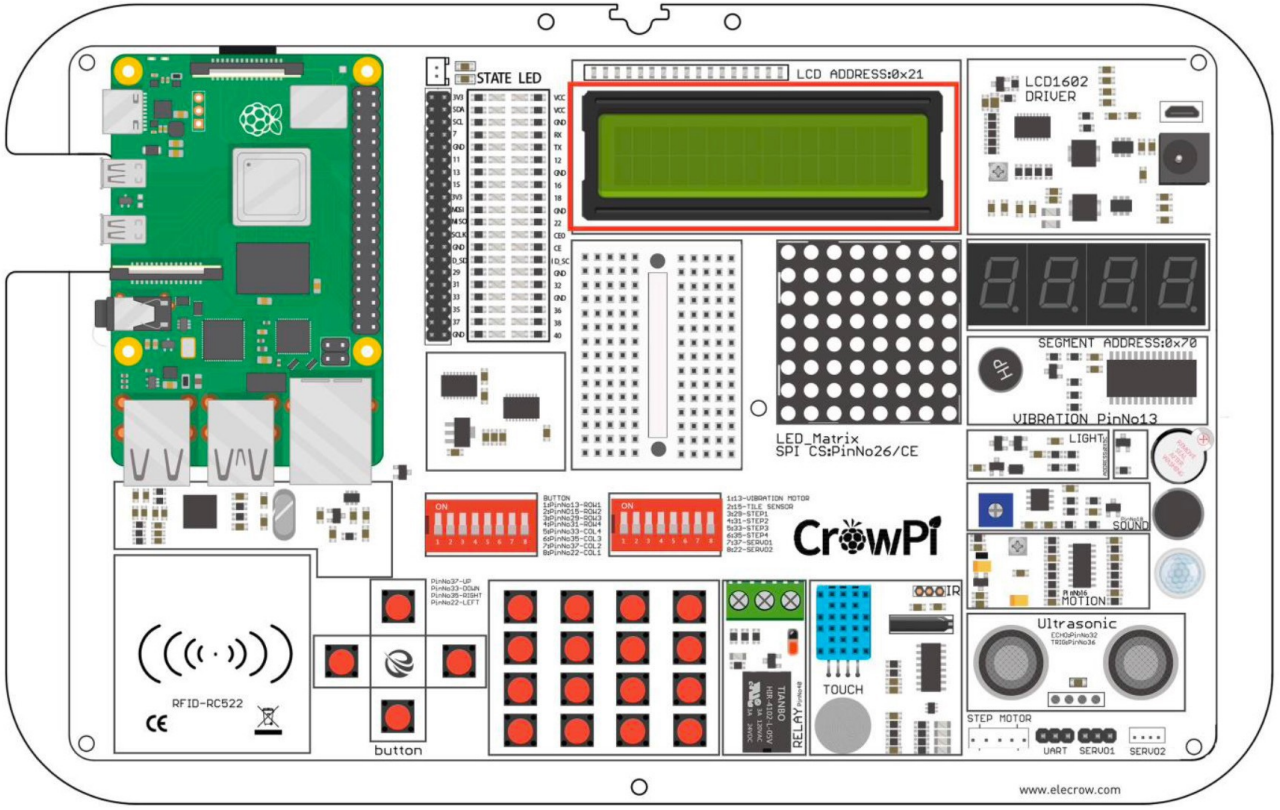
Controlling the LCD Display

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!



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



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.

Implementation

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!

Execute the following commands and try it by yourself:

1- //Download the code from repository: (Skip this part if you already downloaded)
https://github.com/muratkuzlu/CrowPi_Python_Projects/blob/master/Examples

2- //open a terminal and change the directory with following command
cd ~/Desktop/CrowPi_Python_Projects-master/Examples
or
cd ~/Desktop/CrowPi/Examples

3- //type following command to run Python script
sudo python3 lcd.py

Code

#Script start

```
#!/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!')

    # Wait 5 seconds
    time.sleep(5.0)

    # Demo showing the cursor.
    lcd.clear()
    lcd.show_cursor(True)
    lcd.message('Show cursor')

    time.sleep(5.0)

    # Demo showing the blinking cursor.
    lcd.clear()
    lcd.blink(True)
    lcd.message('Blink cursor')

    time.sleep(5.0)
```

#Script continue

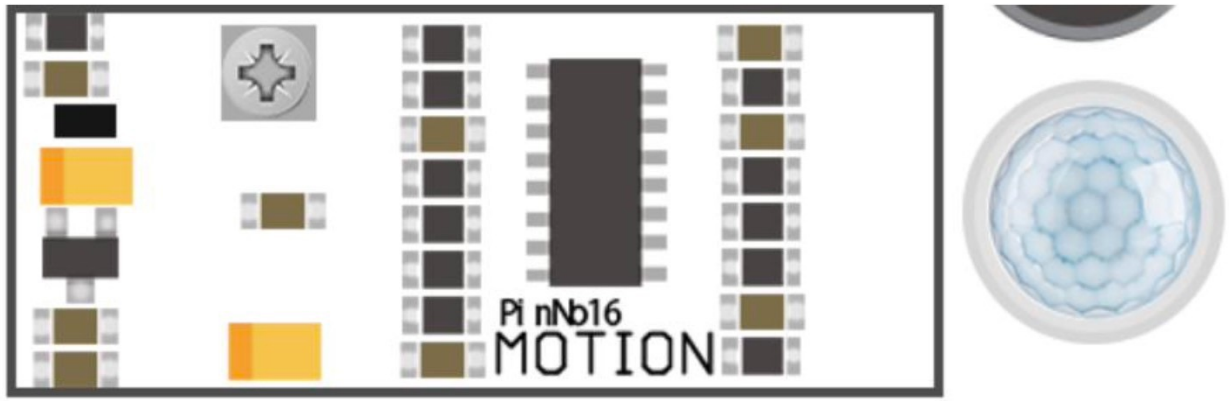
```
    # Stop blinking and showing cursor.
    lcd.show_cursor(False)
    lcd.blink(False)

    # Demo scrolling message right/left.
    lcd.clear()
    message = 'Scroll'
    lcd.message(message)
    for i in range(lcd_columns-len(message)):
        time.sleep(0.5)
        lcd.move_right()
    for i in range(lcd_columns-len(message)):
        time.sleep(0.5)
        lcd.move_left()

    # Demo turning backlight off and on.
    lcd.clear()
    lcd.message('Flash backlight\nin 5 seconds...')
    time.sleep(5.0)
    # Turn backlight off.
    lcd.set_backlight(1)
    time.sleep(2.0)
    # Change message.
    lcd.clear()
    lcd.message('Goodbye!')
    # Turn backlight on.
    lcd.set_backlight(0)
    # Turn backlight off.
    time.sleep(2.0)
    lcd.clear()
    lcd.set_backlight(1)
except KeyboardInterrupt:
    # Turn the screen off
    lcd.clear()
    lcd.set_backlight(1)
```

3- Detect motion using the motion sensor

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 use 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

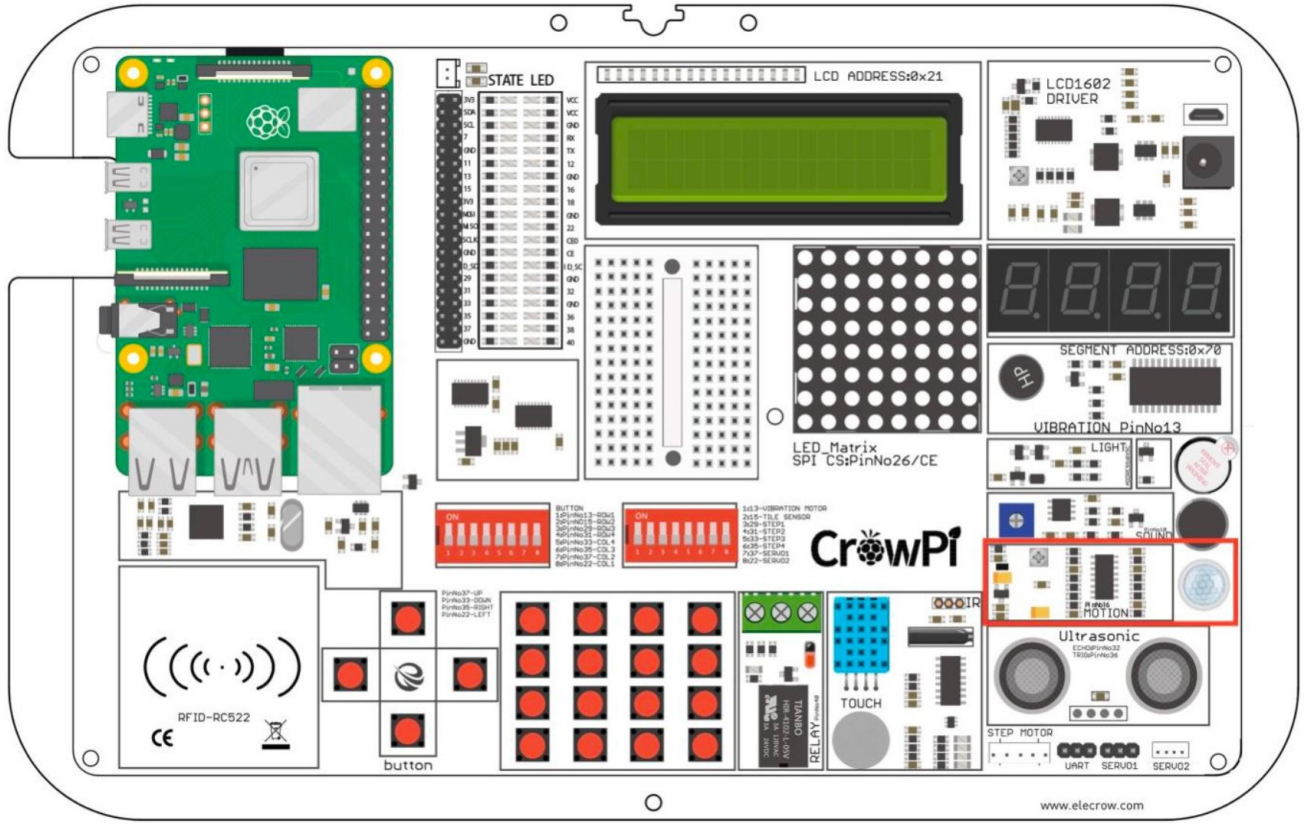
At the end of this lesson, you'll be able to:

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

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!



Implementation

Script

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

import RPi.GPIO as GPIO
import time

# define motion pin
motion_pin = 23

# set GPIO as GPIO.BOARD
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)
except KeyboardInterrupt:
    GPIO.cleanup()
```

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 ...

- **Execute the following commands and try it by yourself:**

1- //Download the code from repository: ***(Skip this part if you already downloaded)***

https://github.com/muratkuzlu/CrowPi_Python_Projects/blob/master/Examples

2- //open a terminal and change the directory with following command

cd ~/Desktop/CrowPi_Python_Projects-master/Examples

or

cd ~/Desktop/CrowPi/Examples

3- //type following command to run Python script

- **sudo python3 motion.py**

Configuring the sensitivity - Optional

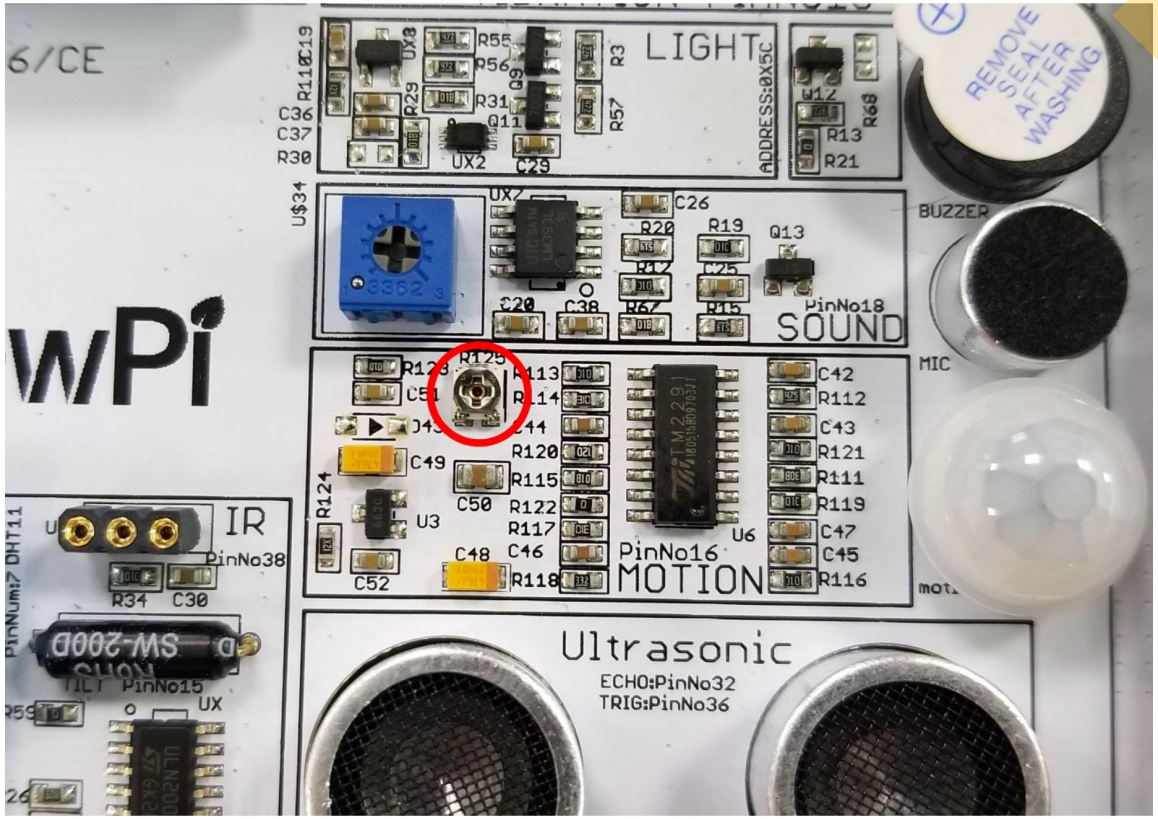
The motion sensor includes a tiny screw right under the sound sensor potentiometer (the blue thing to configure the sound sensitivity)

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

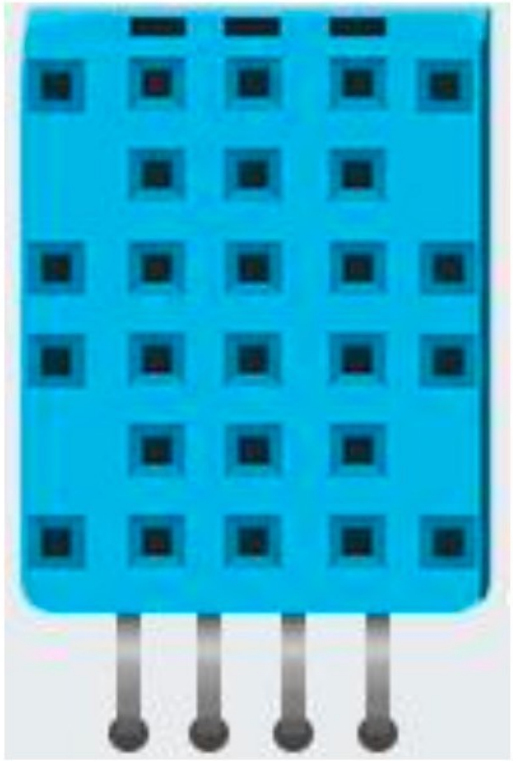
By adjusting the sensitivity of the motion sensor we'll be able to let the motion sensor know from what

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.



4- Detect room temperature and humidity using the **DHT11** sensor



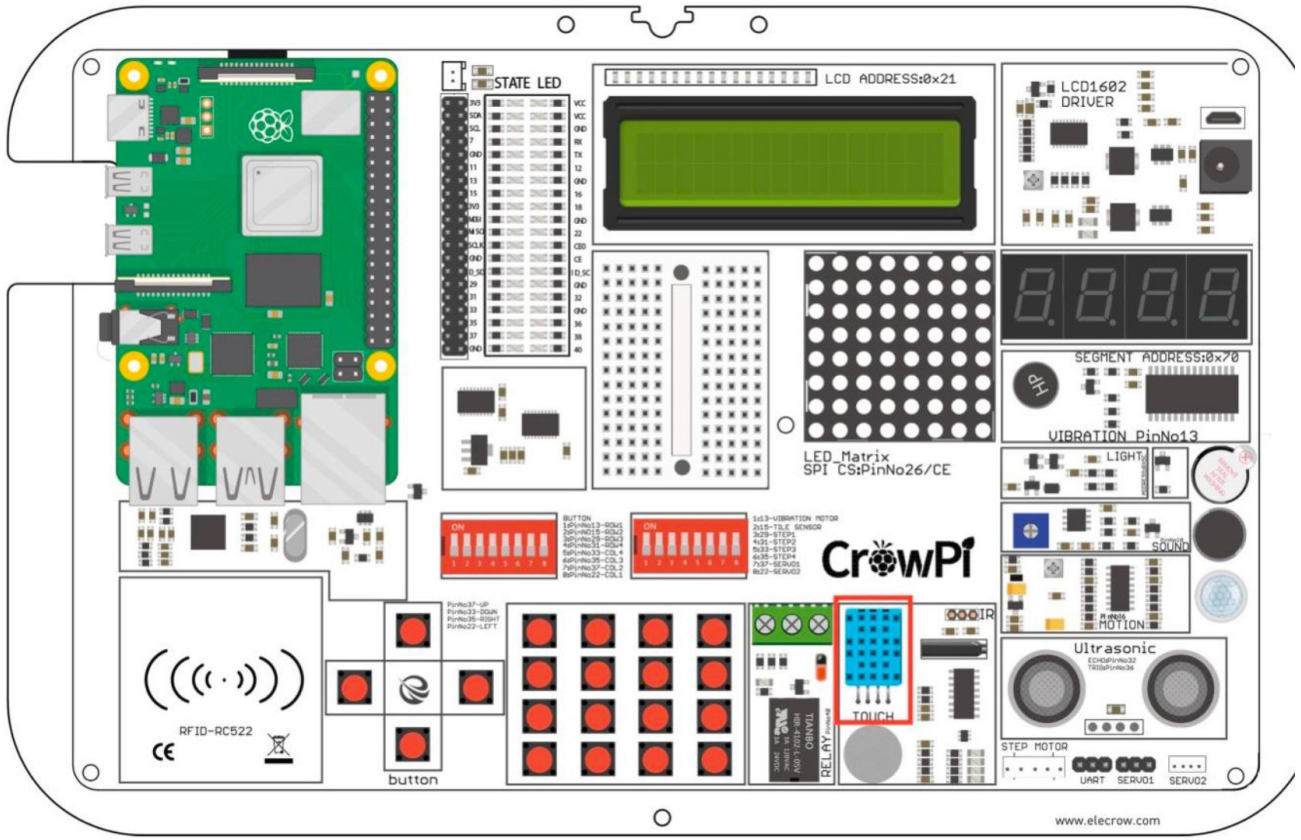
Detect room temperature and humidity

The DHT11 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 DHT11 sensor



DHT11 sensor location on the CrowPi

The DHT11 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!

Implementation

Working with the DHT11 sensor

Working with DHT11 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!

Execute the following commands and try it by yourself:

1- //Download the code from repository: (Skip this part if you already downloaded)

https://github.com/muratkuzlu/CrowPi_Python_Projects/blob/master/Examples

2- //open a terminal and change the directory with following command

cd ~/Desktop/CrowPi_Python_Projects-master/Examples

or

cd ~/Desktop/CrowPi/Examples

3-//type following command to run Python script

sudo python3 dh11.py

Example Code

Script

```
#!/usr/bin/python
# Copyright (c) 2014 Adafruit Industries
# Author: Tony DiCola

import sys
import Adafruit_DHT

# set type of the sensor
sensor = 11
# set pin number
pin = 4

# Try to grab a sensor reading.  Use the read_retry method which will retry up
# to 15 times to get a sensor reading (waiting 2 seconds between each retry).
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

# Un-comment the line below to convert the temperature to Fahrenheit.
# temperature = temperature * 9/5.0 + 32

# Note that sometimes you won't get a reading and
# the results will be null (because Linux can't
# guarantee the timing of calls to read the sensor).
# If this happens try again!
if humidity is not None and temperature is not None:
    print('Temp={0:0.1f}*  Humidity={1:0.1f}%'.format(temperature, humidity))
else:
    print('Failed to get reading. Try again!')
```

License agreement

5- Using the CrowPi Camera

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

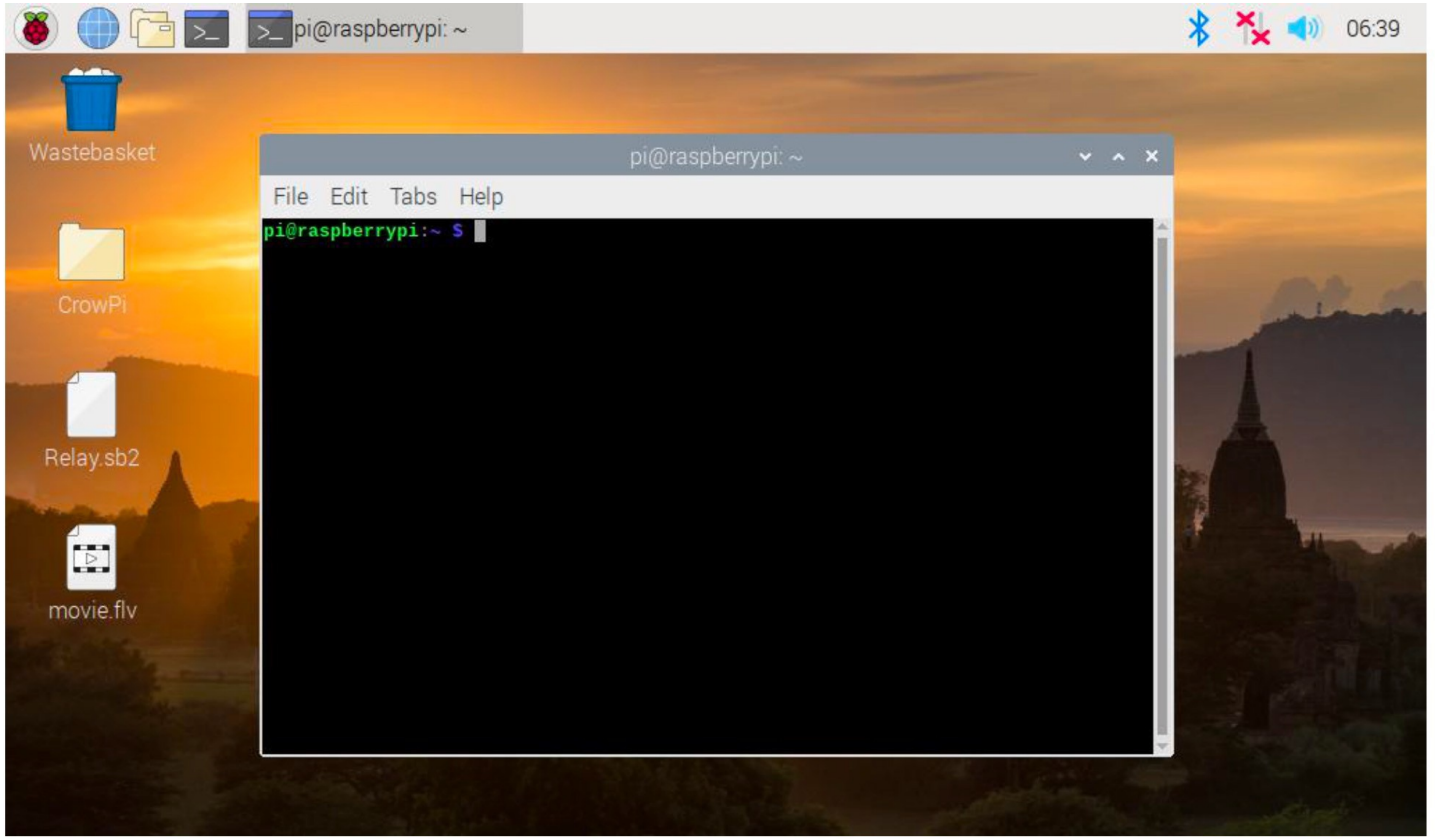
At the end of this lesson, you'll be able to:
* Take picture using the CrowPi Raspberry Pi camera

What will you need

* CrowPi Board after initial installation

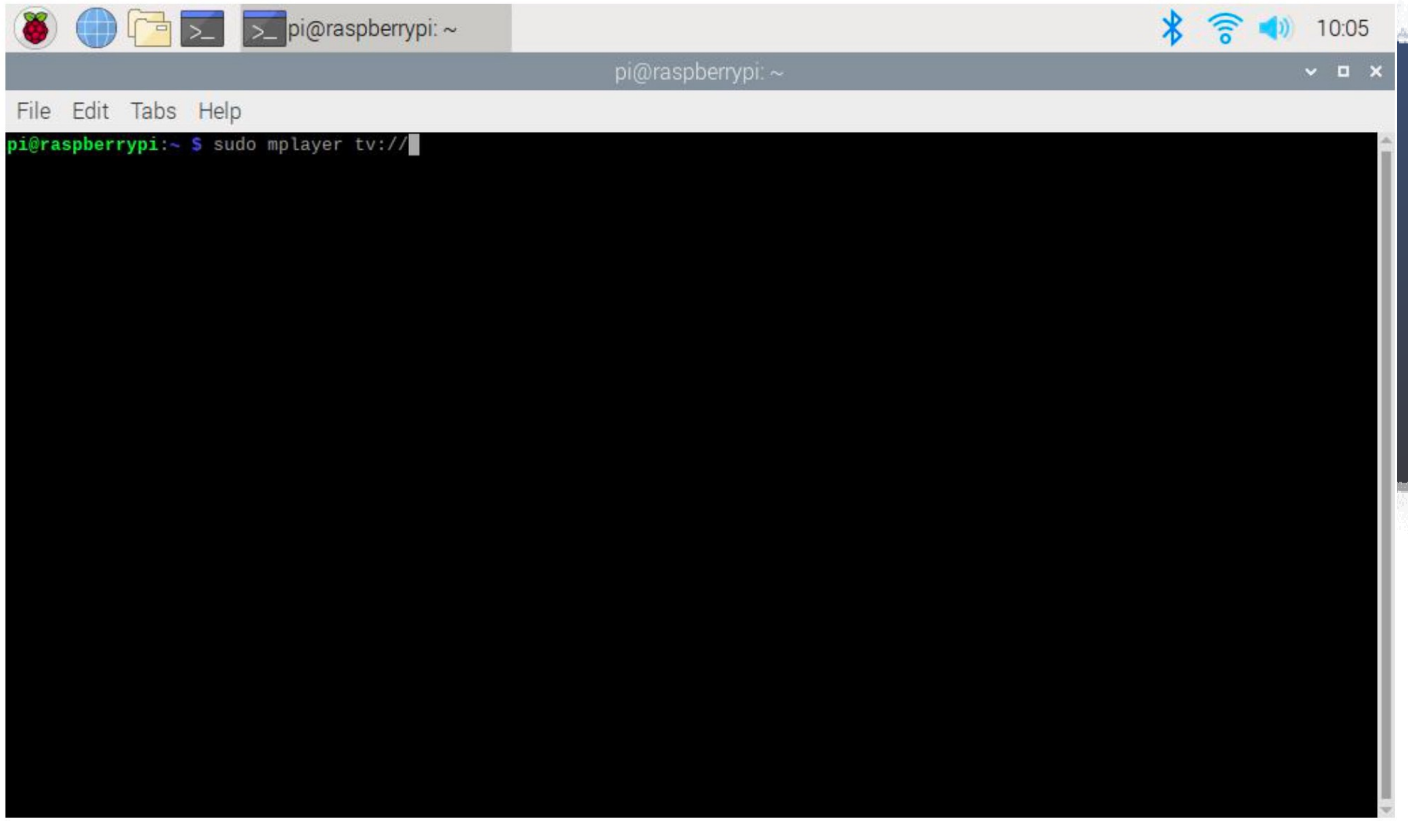
STEPS

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

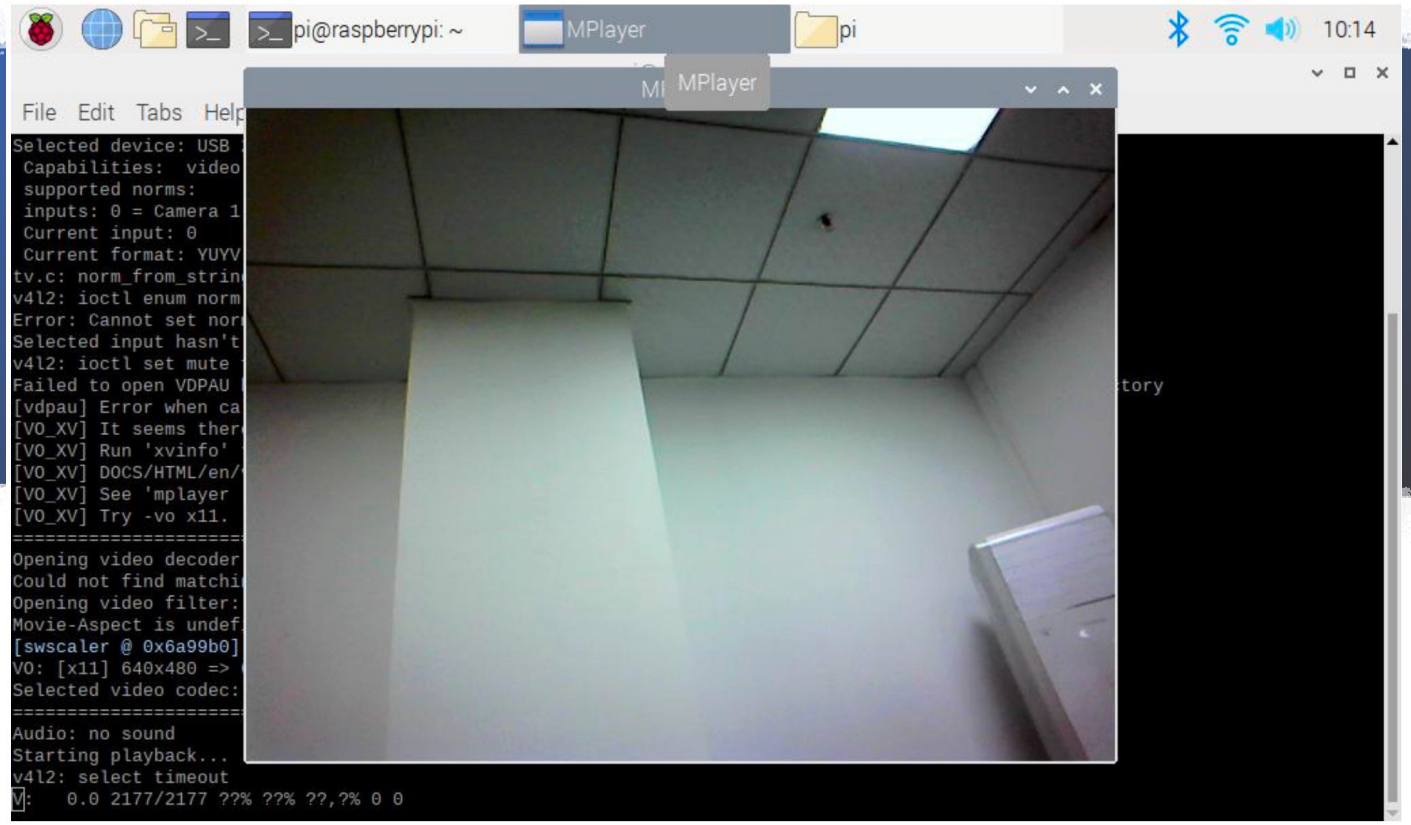


1. Open the camera

sudo mplayer tv://



Then you will see the camera window

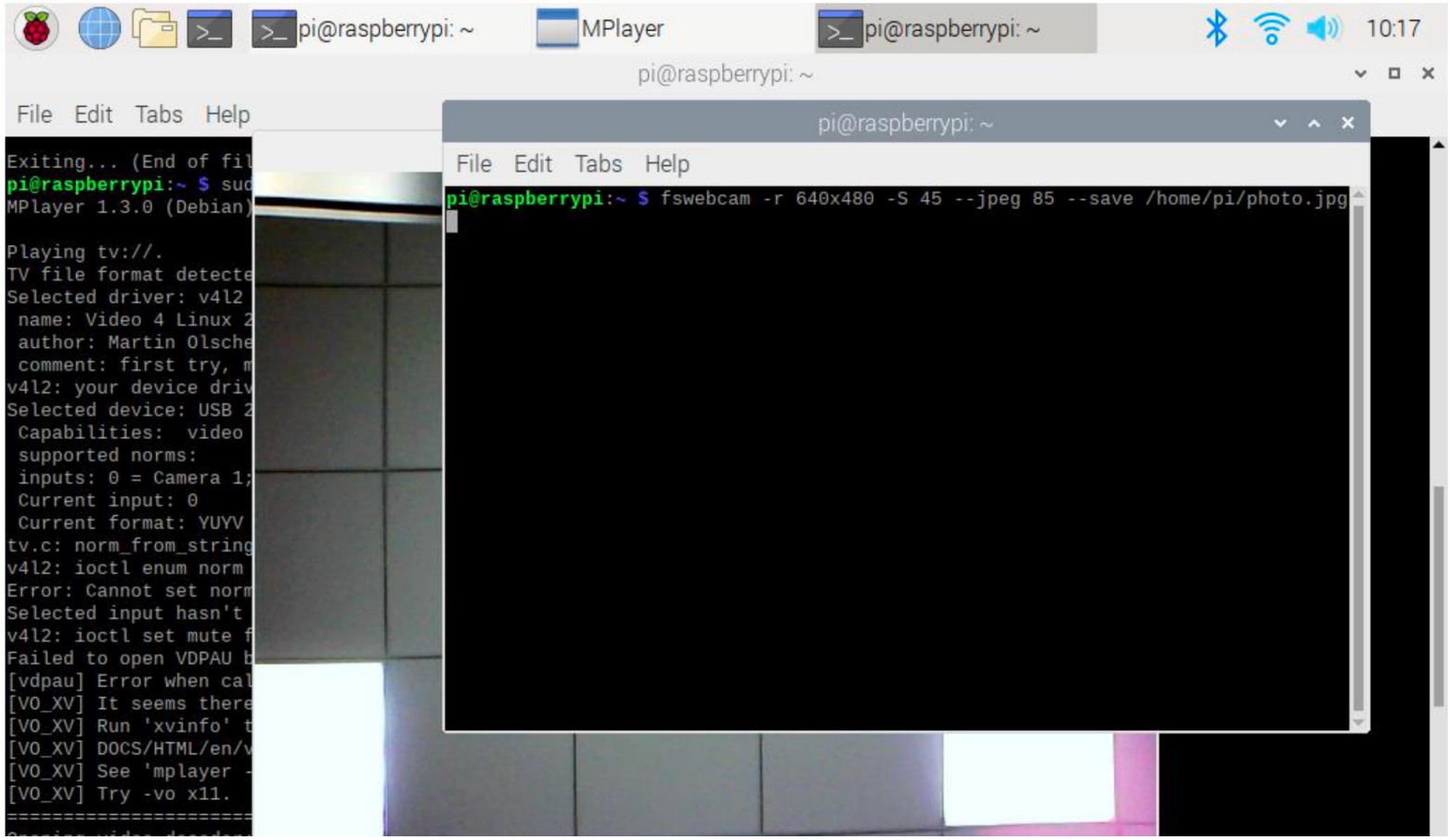


2. Take pictures

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.



More

Get input from a button to control the Buzzer

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.

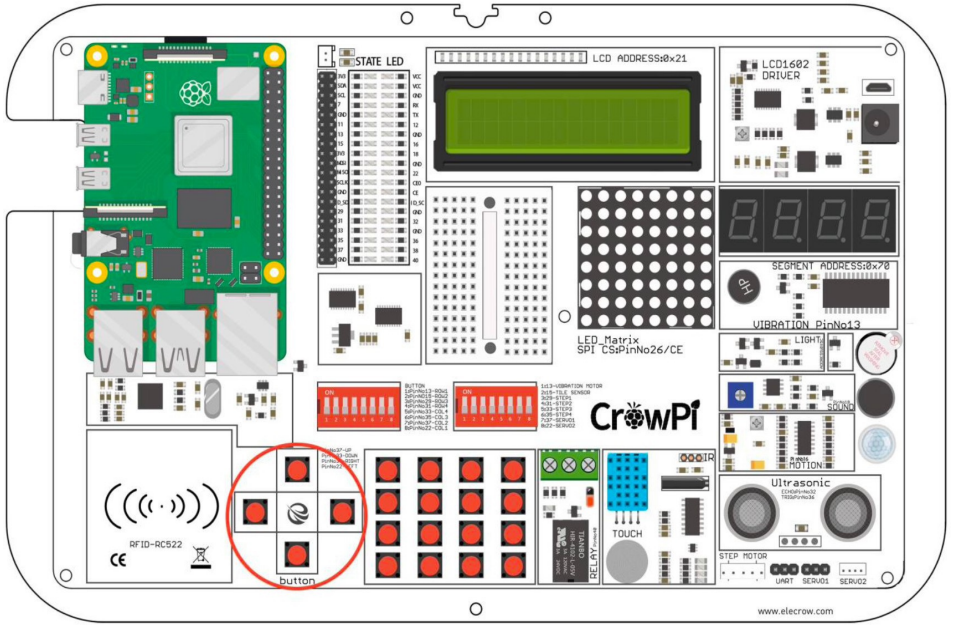
One will be the GPIO.INPUT which will take charge of the button as an input way to get the “press”, another one will be the GPIO.OUTPUT which will send a signal to the buzzer to make some noise.

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

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!

Implementation

Script

```
# !/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.BOARD
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
        if(GPIO.input(button_pin) == 0):
            # set buzzer on
            GPIO.output(buzzer_pin, GPIO.HIGH)
        else:
            # it's not pressed, set button off
            GPIO.output(buzzer_pin, GPIO.LOW)
except KeyboardInterrupt:
    GPIO.cleanup()
```

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.

Execute the following commands and try it by yourself:

//Download the code from repository: ***(Skip this part if you already downloaded)***
https://github.com/muratkuzlu/CrowPi_Python_Projects/blob/master/Examples

//open a terminal and change the directory with following command
cd ~/Desktop/CrowPi_Python_Projects-master/Examples

//type following command to run Python script
sudo python3 button_buzzer.py

Detect sound using the sound sensor

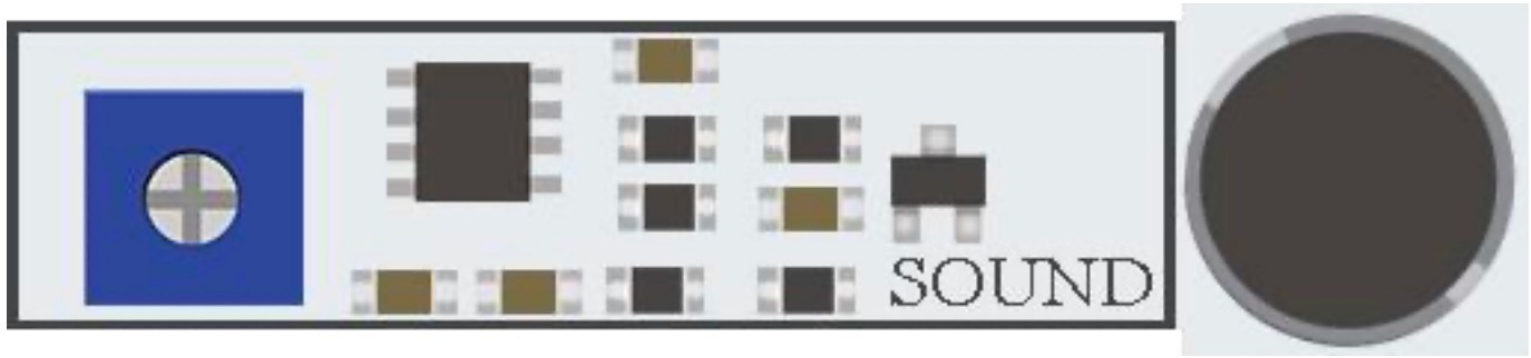
Detect sound using the sound sensor

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

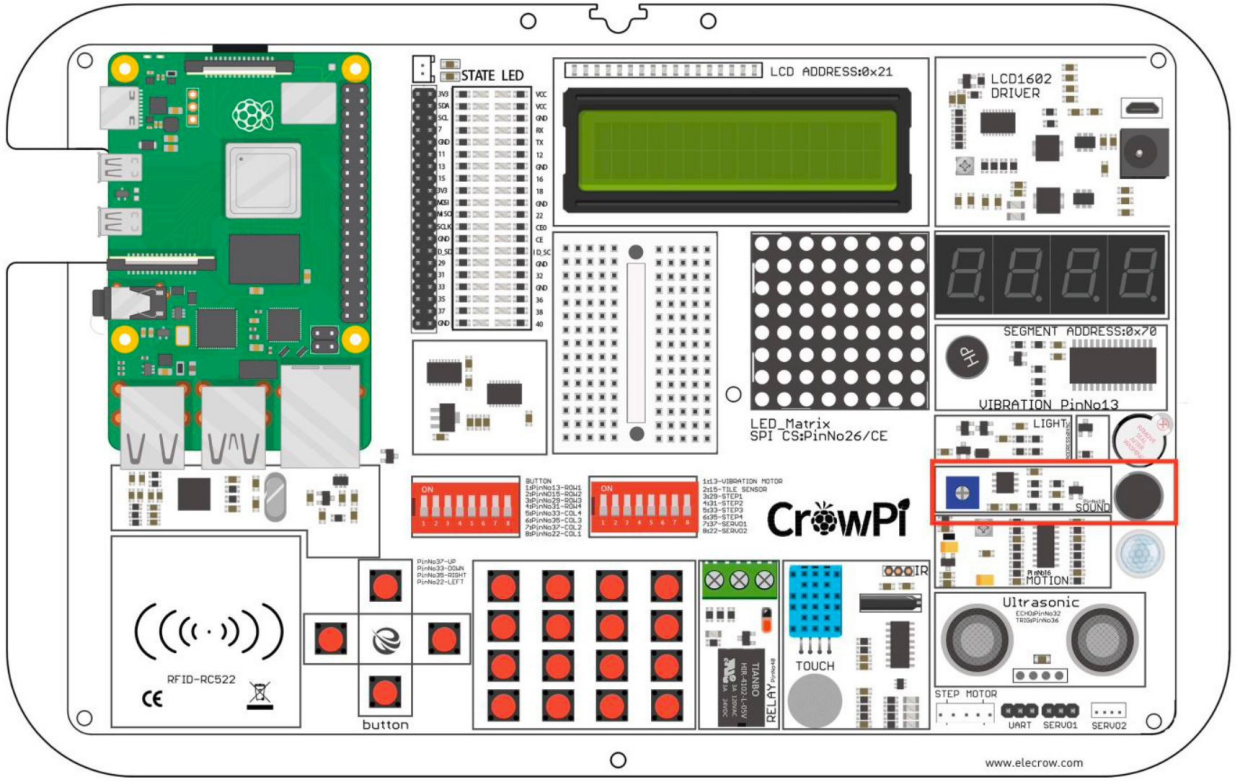
At the end of this lesson, you'll be able to:
* You will learn how to detect sound using the sound sensor module



Sound sensor location on the CrowPi

The sound sensor can be seen 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

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 screwdriver.

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.

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 below:

Implementation

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:

Script

```
#!/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.BOARD
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:
    # CTRL+C detected, cleaning and quitting the script
    GPIO.cleanup()
```

* 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:

//Download the code from repository: ***(Skip this part if you already downloaded)***
https://github.com/muratkuzlu/CrowPi_Python_Projects/blob/master/Examples

//open a terminal and change the directory with following command
cd ~/Desktop/CrowPi_Python_Projects-master/Examples

//type following command to run Python script
sudo python3 sound.py

Detect low or bright light using the Light sensor

Detect low or bright light using the Light sensor.

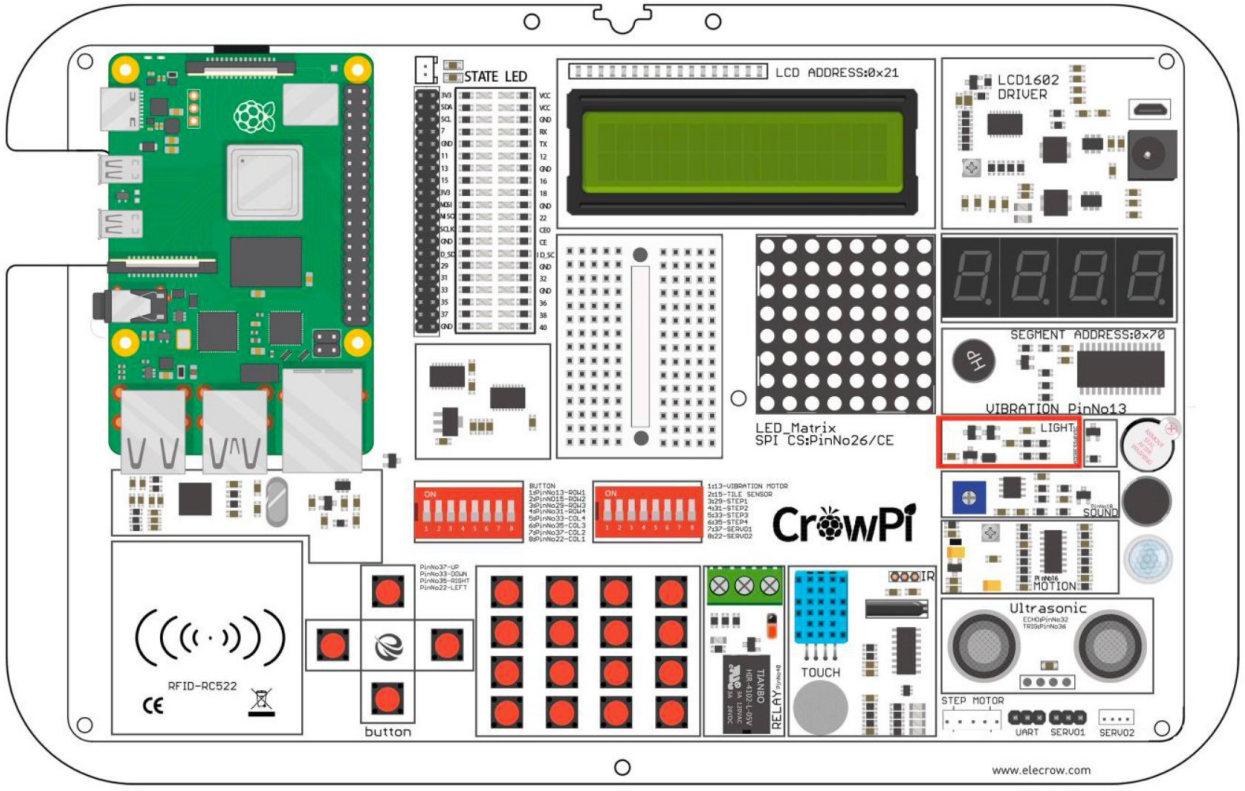
The Light sensor is one of our favorites. It’s extremely useful in many projects and situations for example if you’d like to control something by detecting if there is bright light around, the light sensor is a great module for that exact case.

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



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 ...

Implementation

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).

Execute the following commands and try it by yourself:

//Download the code from repository: ***(Skip this part if you already downloaded)***
https://github.com/muratkuzlu/CrowPi_Python_Projects/blob/master/Examples

//open a terminal and change the directory with following command
cd ~/Desktop/CrowPi_Python_Projects-master/Examples

//type following command to run Python script
sudo python3 light_sensor.py

Script

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author: Matt Hawkins
# Author's Git: https://bitbucket.org/MattHawkinsUK/
# Author's website: https://www.raspberrypi-spy.co.uk
# http://elecrow.com/

import RPi.GPIO as GPIO
import smbus
import time

# Find the right revision for bus driver
if(GPIO.RPI_REVISION == 1):
    bus = smbus.SMBus(0)
else:
    bus = smbus.SMBus(1)

class LightSensor():

    def __init__(self):

        # Define some constants from the datasheet

        self.DEVICE = 0x5c # Default device I2C address

        self.POWER_DOWN = 0x00 # No active state
        self.POWER_ON = 0x01 # Power on
        self.RESET = 0x07 # Reset data register value

…….
```

```
        ……..
        self.CONTINUOUS_HIGH_RES_MODE_2 = 0x11
        # Start measurement at 1lx resolution. Time typically 120ms
        # Device is automatically set to Power Down after measurement.
        self.ONE_TIME_HIGH_RES_MODE_1 = 0x20
        # Start measurement at 0.5lx resolution. Time typically 120ms
        # Device is automatically set to Power Down after measurement.
        self.ONE_TIME_HIGH_RES_MODE_2 = 0x21
        # Start measurement at 1lx resolution. Time typically 120ms
        # Device is automatically set to Power Down after measurement.
        self.ONE_TIME_LOW_RES_MODE = 0x23

    def convertToNumber(self, data):

        # Simple function to convert 2 bytes of data
        # into a decimal number
        return ((data[1] + (256 * data[0])) / 1.2)

    def readLight(self):

        data =
bus.read_i2c_block_data(self.DEVICE,self.ONE_TIME_HIGH_RES_MODE_1)
        return self.convertToNumber(data)

def main():

    sensor = LightSensor()
    try:
        while True:
            print("Light Level : " + str(sensor.readLight()) + " lx")
            time.sleep(0.5)
    except KeyboardInterrupt:
        pass

if __name__ == "__main__":
    main()
```

Getting distance information using the Ultrasonic sensor

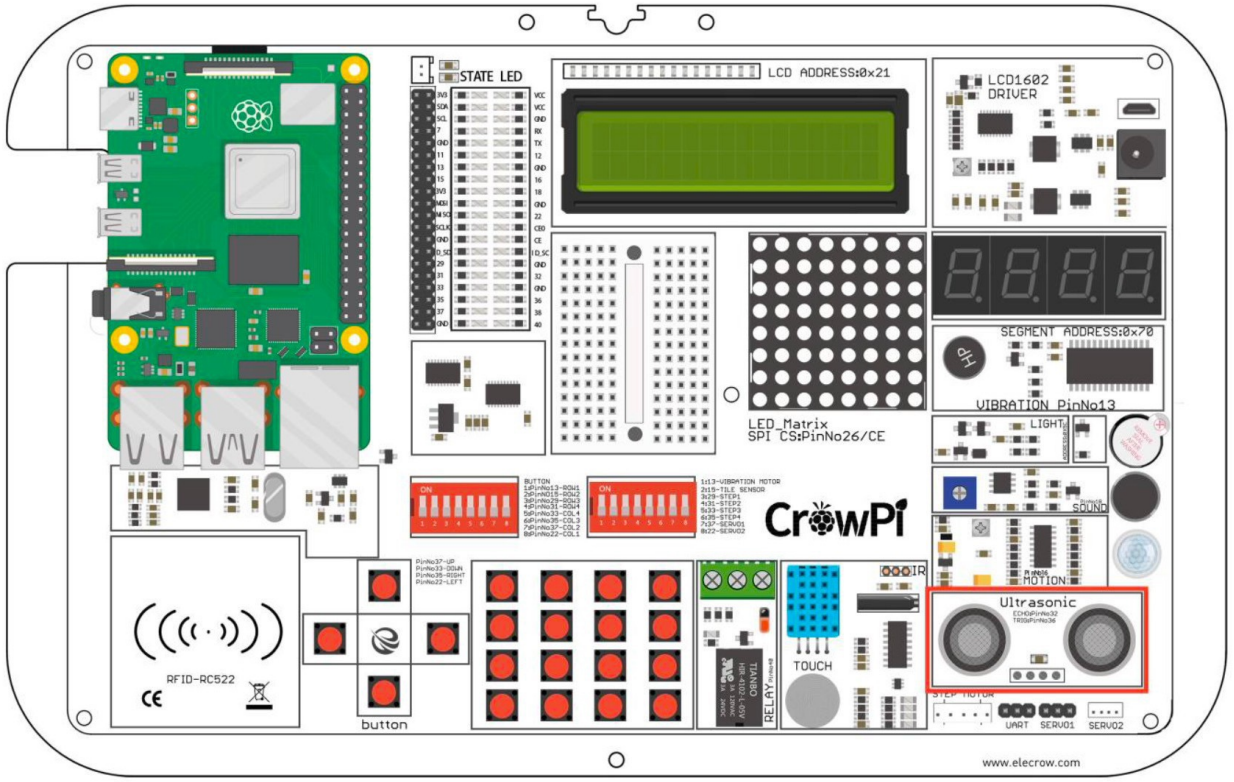
Getting distance information using the Ultrasonic sensor

- 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



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!

Implementation

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author : www.modmypi.com
#  Link:  https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

TRIG = 16
ECHO = 12

print("Distance Measurement In Progress")

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)

GPIO.output(TRIG, False)
print("Waiting For Sensor To Settle")
time.sleep(2)

GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

while GPIO.input(ECHO)==0:
  pulse_start = time.time()

while GPIO.input(ECHO)==1:
  pulse_end = time.time()

pulse_duration = pulse_end - pulse_start

distance = pulse_duration * 17150

distance = round(distance, 2)

print("Distance: %scm" % distance)

GPIO.cleanup()
```

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.

Execute the following commands and try it by yourself:

//Download the code from repository: ***(Skip this part if you already downloaded)***
https://github.com/muratkuzlu/CrowPi_Python_Projects/blob/master/Examples

//open a terminal and change the directory with following command
cd ~/Desktop/CrowPi_Python_Projects-master/Examples

//type following command to run Python script
sudo python3 distance.py

Detecting touch using the Touch Sensor

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.

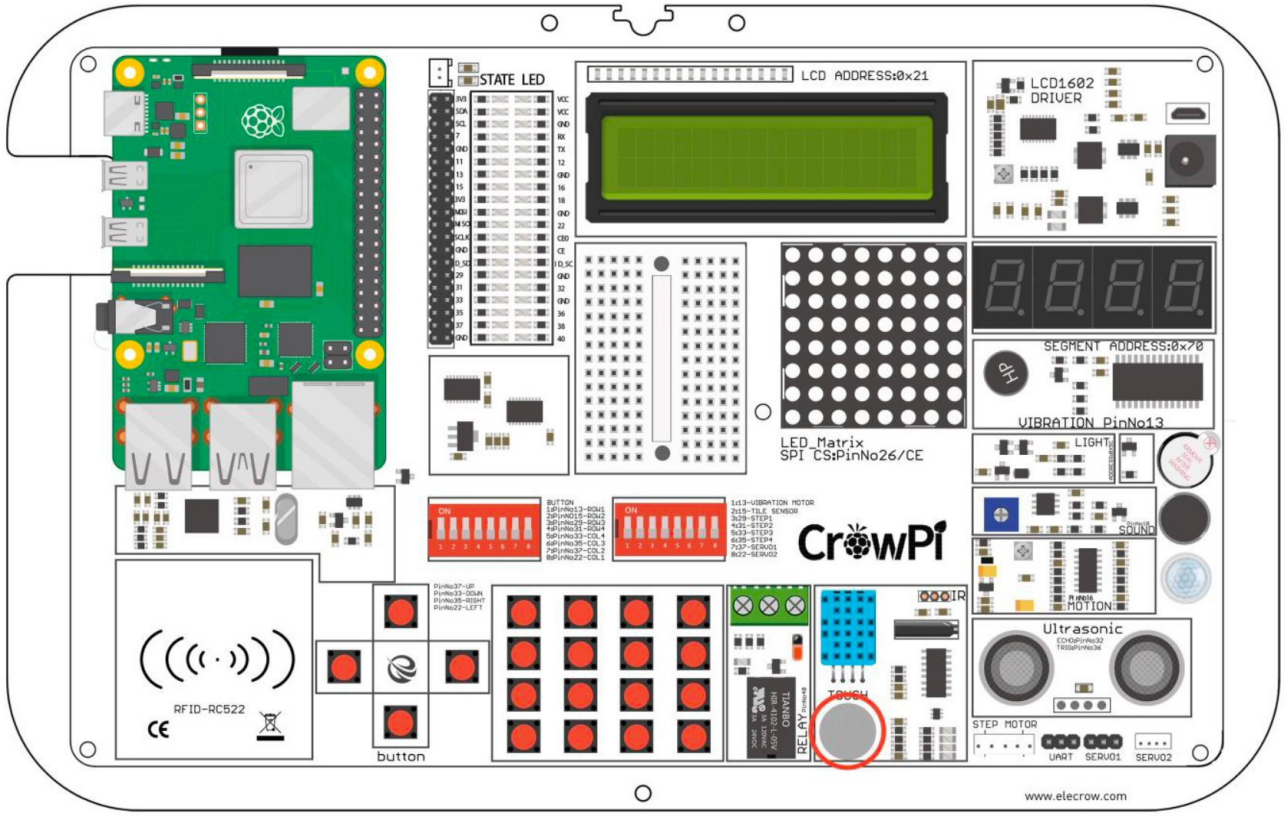
The touch sensor comes in handy in games like who can touch the pad first, or when you want to activate something by touch and not by pressing a button.

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



Touch Sensor location on the CrowPi

The touch sensor is located right under the DHT11 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!

Implementation

Script

```
#!/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.BOARD
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')
        time.sleep(0.1)
except KeyboardInterrupt:
    # CTRL+C detected, cleaning and quitting the script
    GPIO.cleanup()
```

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.

Execute the following commands and try it by yourself:

//Download the code from repository: ***(Skip this part if you already downloaded)***
https://github.com/muratkuzlu/CrowPi_Python_Projects/blob/master/Examples

//open a terminal and change the directory with following command
cd ~/Desktop/CrowPi_Python_Projects-master/Examples

//type following command to run Python script
sudo python3 touch.py

Detecting tilt using the Tilt Sensor

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.

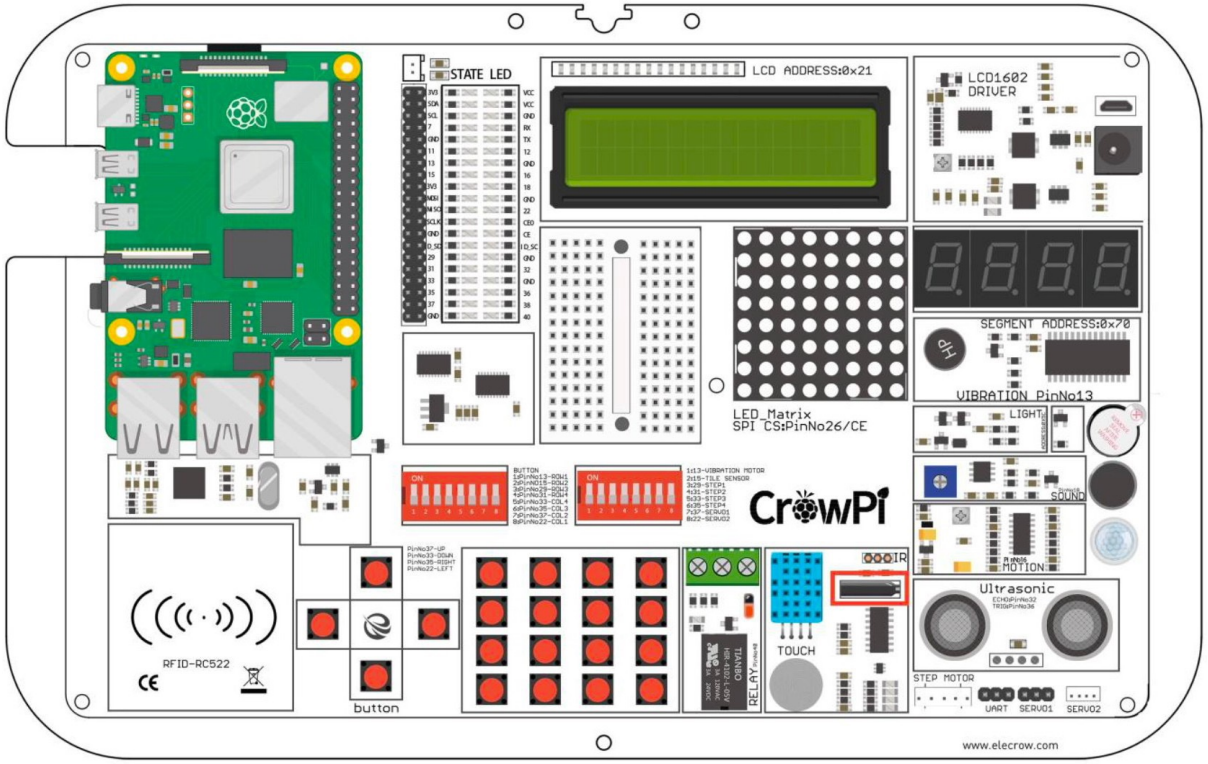
Tilt Sensor location on the CrowPi

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.



Implementation

Script

```
#!/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.BOARD
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:
            print("[-] Right Tilt")
        time.sleep(1)
except KeyboardInterrupt:
    # CTRL+C detected, cleaning and quitting the script
    GPIO.cleanup()
```

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!

Execute the following commands and try it by yourself:

//Download the code from repository: ***(Skip this part if you already downloaded)***
https://github.com/muratkuzlu/CrowPi_Python_Projects/blob/master/Examples

//open a terminal and change the directory with following command
cd ~/Desktop/CrowPi_Python_Projects-master/Examples

//type following command to run Python script
sudo python3 tilt.py

References

- CrowPi- compact Raspberry Pi educational kit. (n.d.). Retrieved February 25, 2021, from https://www.elecrow.com/crowpi-compact-raspberry-pi-educational-kit.html#:~:text=The%20only%20difference%20is%20that,kit%20is%20good%20for%20you.
- Dr Anand NayyarThe author works in a Graduate School, Nayyar, D., & The author works in a Graduate School. (2020, April 03). Top 8 IDEs for Raspberry Pi: Top Raspberry Pi IDEs. Retrieved February 25, 2021, from https://www.opensourceforu.com/2017/06/top-ides-raspberry-pi/
- J., Jony, Says, M., Mark, Says, R., . . . Bubba. (2017, December 25). What are the differences between Raspberry Pi and ARDUINO? Retrieved February 25, 2021, from https://www.electronicshub.org/raspberry-pi-vs-arduino/#:~:text=The%20main%20difference%20between%20them,Arduino%20makes%20hardware%20projects%20simple.

- Github - https://github.com/muratkuzlu/CrowPi_Python_Projects