python Guide

Guide for the teacher

```
import sys, atexit , msvcrt
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(

channel = 7
moway.usbinit_moway()
ret = moway.i
if ret == 0:
        pri
else:
    print 'Mo
    exit(-1)

while True:
    if msvcrt.kbhit():
        ch = msvcrt.getch()
        if ch=='w':
            moway.command_moway(CMD_GO_SIMPLE,0)
            moway.command_moway(CMD_LEDSOFF,0)
            moway.command_moway(CMD_FRONTLED
        if ch=='z':
            moway.command_moway(CMD_BA
            moway.command_moway(CMD
            moway.command_moway(
        if ch=='a':
            moway.command
            moway.co
            mowa
        if
```

mOway®

education

INDEX

1. INTRODUCTION

mOway is a small, programmable autonomous robot designed mainly for practical applications of mobile robotics. It is a perfect educational tool for those who wish to take their first steps in the world of robotics and for those who have worked with robots and wish to develop more complex applications.

The mOway robot is fitted with a number of sensors that help us to move around a real environment. It is also equipped with a motor that enables it to travel on the floor. All these peripheral devices are connected to a microcontroller, which is responsible for controlling the robot.

mOway develops personal skills such as creativity, a desire to continue learning and team work. Its great advantage is its rapid learning curve: students get results right from the very first class and this generates a great deal of motivation.

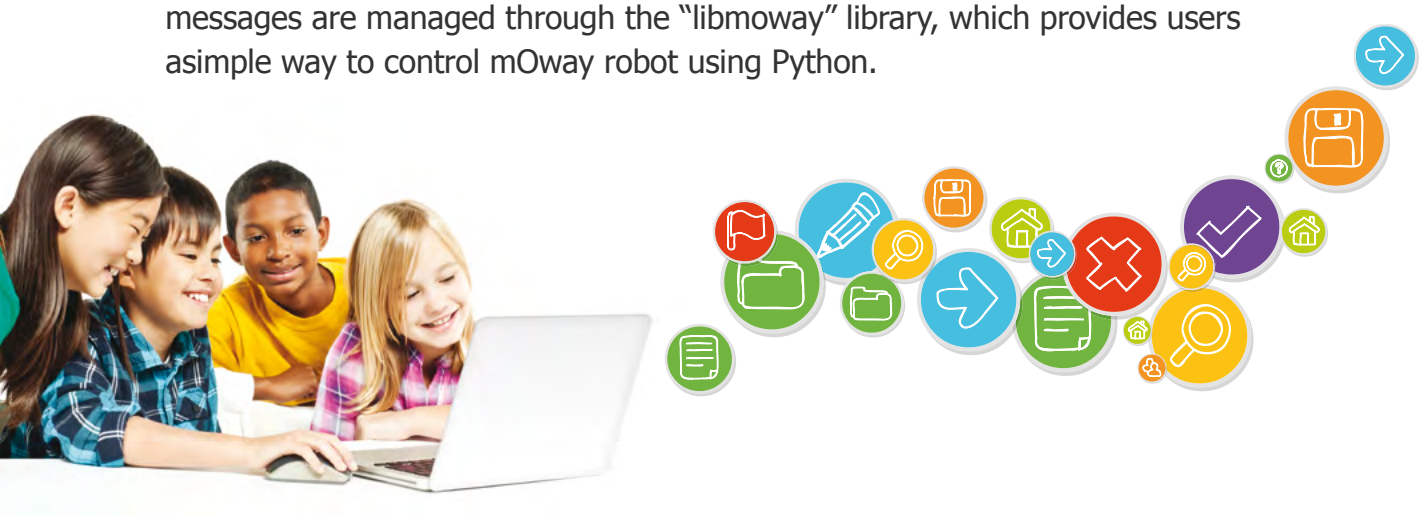
2. MOWAY AND PYTHON COMMUNICATION

Introduction

The mOway robot is controlled from Python environment using a number of different commands. A command is an order coded and sent to the robot. When the robot receives this order, it performs the action it has been ordered to carry out. For example, if we want the robot to advance, we must send the "CMD_GO" command in python. When we execute this command, the robot will receive this order and its wheels will activate.

On the other hand, the mOway robot is equipped with a number of different sensors. A sensor is an electronic device used to measure the surrounding conditions. For example, mOway is fitted with obstacle sensors to detect objects in front of the robot, a light sensor to detect whether it is daytime or night time and a temperature sensor, etc. The robot sends the values detected by its sensors to the PC environment on a continuous basis.

Both the commands and the values detected by the sensors are sent by radiofrequency (RF). The mOway robot and the Python environment exchange messages wirelessly. These messages are managed through the "libmoway" library, which provides users asimple way to control mOway robot using Python.



Radiofrequency

To enable wireless communications between the Python environment and the mOway robot, the following items are required:

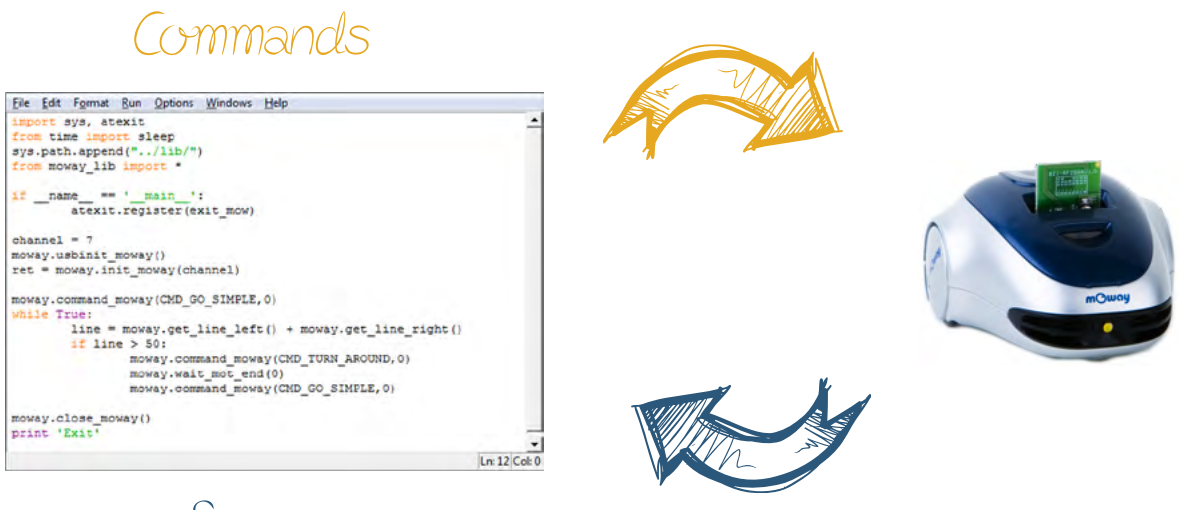
	RFUSB	- This is connected to the computer. - It is used to send commands from Python and to receive the values detected in the robot sensors.
	RF Module	- This is connected to the mOway robot. - It is used to receive commands from Phtyon and to send values detected by the sensors.

Commands

In the Python environment, commands are sent with the:

```
moway. command_moway(COMMAND_TO_SEND,TIMEOUT)
```

function. This block sends the command in brackets to the mOway Robot. The second parameter of the function called TIMEOUT is the amount of time the system waits for mOway to receive the command. As the communication between Python and mOway is based on RF it is possible that commands cannot get to its destination correctly. This function also has a return value. The value returned is "0" if the command is received by mOway correctly. It is "1" if the command sent is the same as the last command sent to the mOway and the value is "2" if the timeout counter ends and the command is not received by mOway. The TIMEOUT parameter is expressed in ms.



Sensors

The robot sends the value detected in its sensors via radiofrequency on a continuous basis. These values are received by libmoway and can be read in Python using `moway.get__XXXXX()` command. We have different coding for each sensor.

For example `moway.get_obs_center_left()`, gets the value of center-left obstacle sensor. In ANNEX I you can learn all about the different sensors and how to read their values.

3. EXERCISE: MOVEMENT

Introduction

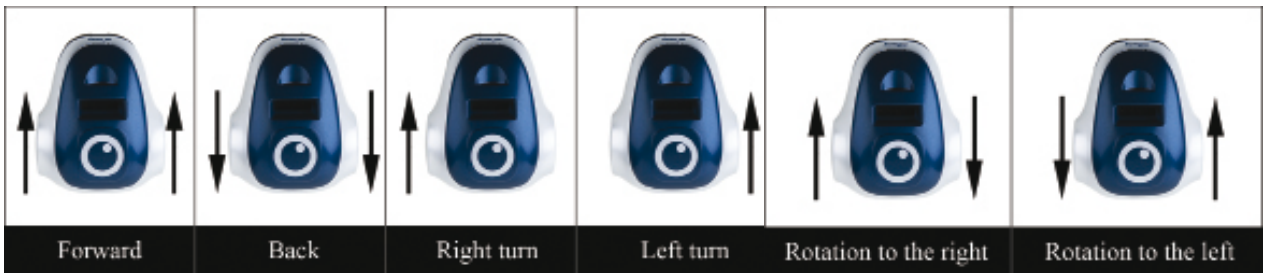
In this exercise, we will learn how to control the movements of the robot using Python. In addition, we will see how we can make a RC car using mOway.

mOway wheels

The robot can move as it is equipped with two wheels which allow it to move forwards, backwards and turn. In this way we can make mOway move in any direction to explore different areas with its sensors, for example.

The two robot wheels are independent of each other, which means that each of these can turn at a different speed and in a different direction. This provides a number of different opportunities:

- If the wheels turn forwards at the same speed, the robot moves forward.
- If the wheels turn backwards at the same speed, the robot moves backwards.
- If the wheels turn at different speeds, the robot moves in a curve.
- If the wheels turn in different directions, the robot turns on its axis.
- If one wheel turns and the other doesn't, the robot turns on the wheel that is not turning.



The movements of the mOway robot robot can be controlled according to the following parameters:

- **Speed**: The speed of the wheels is higher or lower.
- **Time:** We can chose the time during which the robot is to move.
- **Distance:** We can choose the distance to be travelled by the robot.

In order to determine the distance travelled by the robot, the wheels are fitted with an **"encoder"**. An "encoder" is similar to a bicycle odometer: this counts the number of times a wheel turns and, knowing the diameter of the wheel, it is possible to calculate the distance it has travelled.

In order to introduce mOway movement we start using the simple movement commands:

Command	Description
CMD_GO_SIMPLE	mOway goes forward indefinitely
CMD_BACK_SIMPLE	mOway goes backwards indefinitely
CMD_STOP	mOway stops
CMD_LEFT_SIMPLE	mOway turns 90 degree left
CMD_RIGHT_SIMPLE	mOway turns 90 degree right

Exercise 3.1. Square Simple

Before starting check that you have correctly install the software for your operating system. You can read Annex III or IV (Windows or Linux) before programming mOway.

In this exercise we are going to make the mOway robot draw a square on the ground. Starting for a basic implementation of a square we will end creating a regular polygon drawing using more complex functions.

The first part of mOway and python programs should be always the same. We import moway_lib.py file with is small python library in charge of loading mOway functions depending on the OS and it also contains the constants declarations. You can review the source code of moway_lib.py in the lib folder of the mOway_python_pack.

```
import sys, atexit
from time import sleep
sys.path.append("../lib/")
from moway_lib import *
```

This part of the code closes the RF connection once the program is exited.

```
if __name__ == '__main__':
        atexit.register(exit_mow)
```

If you wish to use several robots in different computers, it is necessary to assign a different channel to each robot to avoid any interferences between them. in this example we have programmed mOway for working in channel 7. If the connection cannot be established with mowayRFUSB the "init_moway" function will return a value different from "0".

```
channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
        print 'Moway RFUSB Connected'
else:
        print 'Moway RFUSB not connected. Exit'
        exit(-1)
```

Once the libraries are loaded and the connection is established it is time to program mOway. We will repeat the sequence 4 times to draw a rectangle: go forward for two seconds, turn right and wait one second.

```
for i in range(4):
        moway.command_moway(CMD_GO_SIMPLE,0)
        sleep(2)
        moway.command_moway(CMD_RIGHT_SIMPLE,0)
        sleep(1)
```

Exercise 3.2. Square

In this square exercise we introduce a new command for controlling mOway movements. This command is `wait_mot_end(TIMEOUT)`. This command is used when we send a movement command to mOway with a limited time, distance or angle. In the first example we used the python "sleep" function to move mOway robot for two seconds. But in this new example we will draw a 20 cm side square using `set_distance()` and `wait_mot end()`. Instead of using `CMD_RIGHT_SIMPLE`, we use the command `CMD_ROTATERIGHT` which requires variable settings before calling the function.

```
import sys, atexit
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
         print 'Moway RFUSB Connected'
else:
    print 'Moway RFUSB not connected. Exit'
    exit(-1)

for i in range(4):
    #Set variables for go command
    moway.set_distance(200)
    moway.set_speed(100)
    moway.command_moway(CMD_GO,0)
    #wait for movement end
    moway.wait_mot_end(0)
    #Set variables for rotate command
    moway.set_rotation(90)
    moway.set_rotation_axis(CENTER)
    moway.command_moway(CMD_ROTATERIGHT,0)
    #wait for movement end
    moway.wait_mot_end(0)
```

Exercise 3.3. Regular Polygons

In the next exercise we draw regular polygons using moway. Number and length of sides is entered by user and mOway then calculates the movements needed.

```
import sys, atexit
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
         print 'Moway RFUSB Connected'
else:
    print 'Moway RFUSB not connected. Exit'
    exit(-1)
```

Sides and length is entered by user with `raw_input()function`. This function works in python 2.7. If you are using python 3.x you must modify this part of the code:

```
sides = int(raw_input("Enter sides of the regular polygon (3
-10): "));
length = int(raw_input("Enter length of the side (100 - 255 mm):
"));

if sides < 3 and sides > 10:
        print "Sides number not correct. Exit"
        exit(-1)
if length < 100 and length > 255:
        print "Sides number not correct. Exit"
        exit(-1)
```

Angle is calculated using the formula:

$$\text{interior angle} = (n-2) \times 180° / n$$

As mOway is drawing the external angle:

angle = 180 - interior angle

```
angle = 180 * (1 - float(sides-2)/float(sides))
print angle

for i in range(sides):
    #Set variables for go command
    moway.set_distance(length)
    moway.set_speed(100)
    moway.command_moway(CMD_GO,0)
    #wait for movement end
    moway.wait_mot_end(0)
    #Set variables for rotate command
    moway.set_rotation(int(angle))
    moway.set_rotation_axis(CENTER)
    moway.command_moway(CMD_ROTATERIGHT,0)
    #wait for movement end
    moway.wait_mot_end(0)
```

Exercise 3.4. RC CAR

In this exercise we use mOway as an RC car controlled by the keyboard of the PC. We have two different versions of the program depending on the OS you are using. Here we will explain the Windows version. Note that this example must run in a terminal and not in IDLE shell to work as msvcrt functions used to detect the keys pressed work in a terminal. You can double click the python script to run it on Windows or write `$python 3_4_mowayRC_linux.py` in a terminal in Linux.

```
import sys, atexit , msvcrt
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
```

```
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
        print 'Moway RFUSB Connected'
else:
    print 'Moway RFUSB not connected. Exit'
    exit(-1)
```

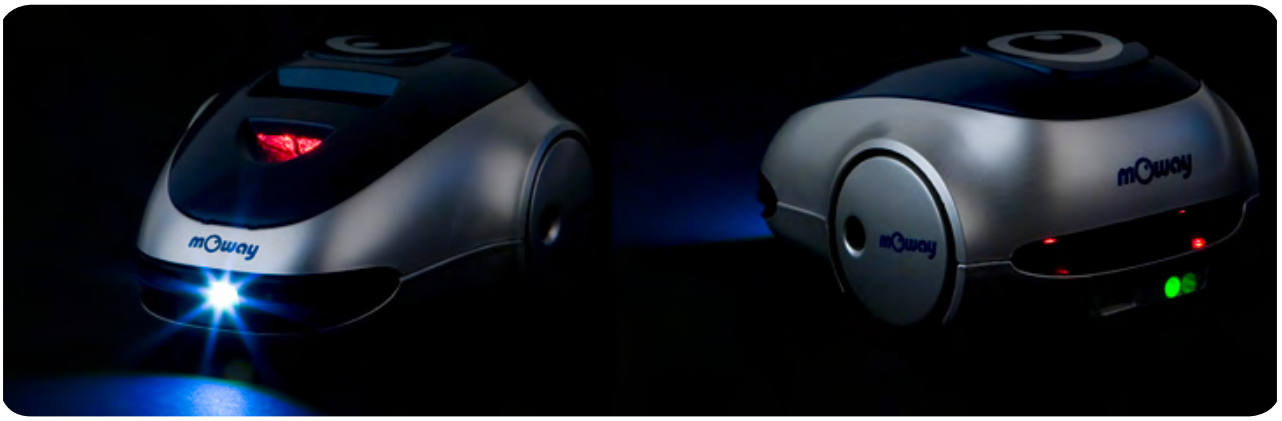
If a key is pressed (`kbhit`), we get the key (`getch`) and depending on its value we send different commands to mOway.

```
while True:
    if msvcrt.kbhit():
         ch = msvcrt.getch()
         if ch=='w':
              moway.command_moway(CMD_GO_SIMPLE,0)
         if ch=='z':
              moway.command_moway(CMD_BACK_SIMPLE,0)
         if ch=='a':
              moway.command_moway(CMD_LEFT_SIMPLE,0)
         if ch=='d':
              moway.command_moway(CMD_RIGHT_SIMPLE,0)
         if ch=='s':
              moway.command_moway(CMD_STOP,0)
```

4. EXERCISE: LEDS AND LOUDSPEAKER

Introduction

In this exercise we are going to explain what an LED is and how we can activate the LEDs of the mOway robot. We will also learn how to emit sounds with the robot using its internal loudspeaker.

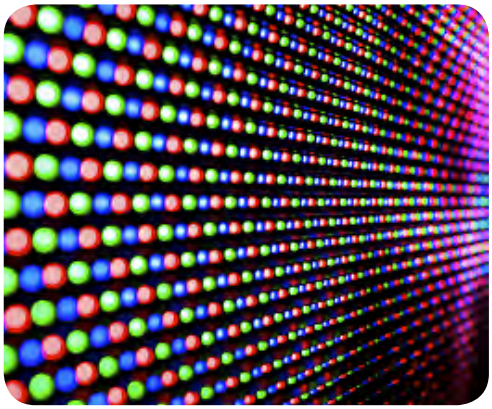


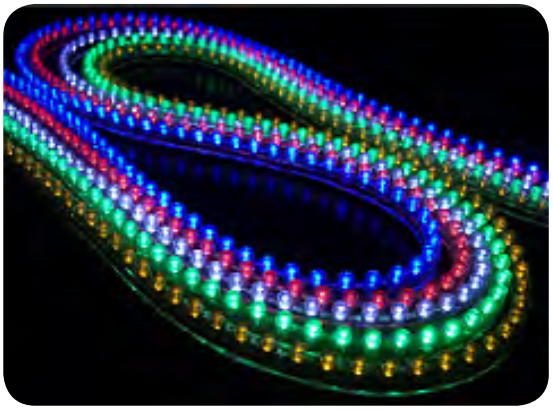
LED lights

An LED is an electronic device similar to a light bulb: when an electric current is passed through an LED, it lights up. The great difference between an LED and a normal light bulb is that LEDs consume much less energy. Moreover, LEDs last longer and withstand vibrations better.

There are LEDs of different colours: white, blue, red, green, etc.

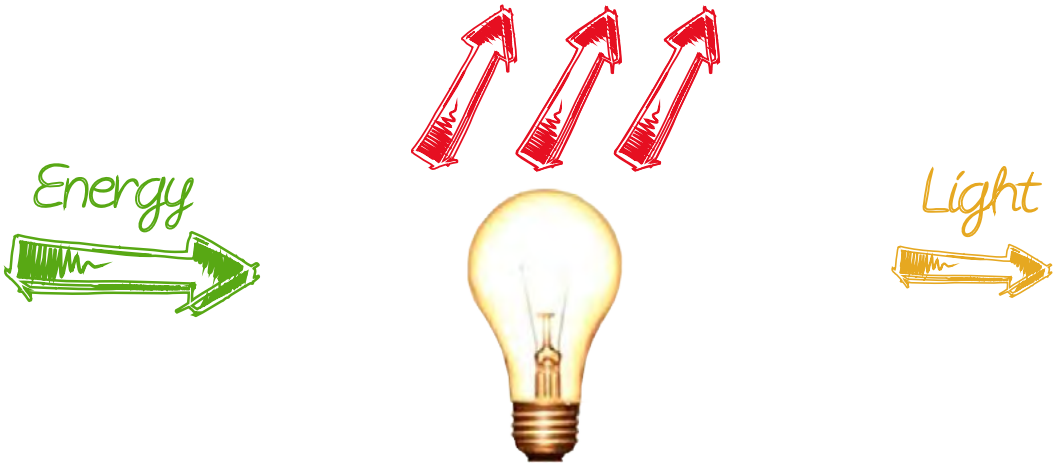
Nowadays, we can see them everywhere: light bulbs, lamps, torches, TVs, car headlights, etc.



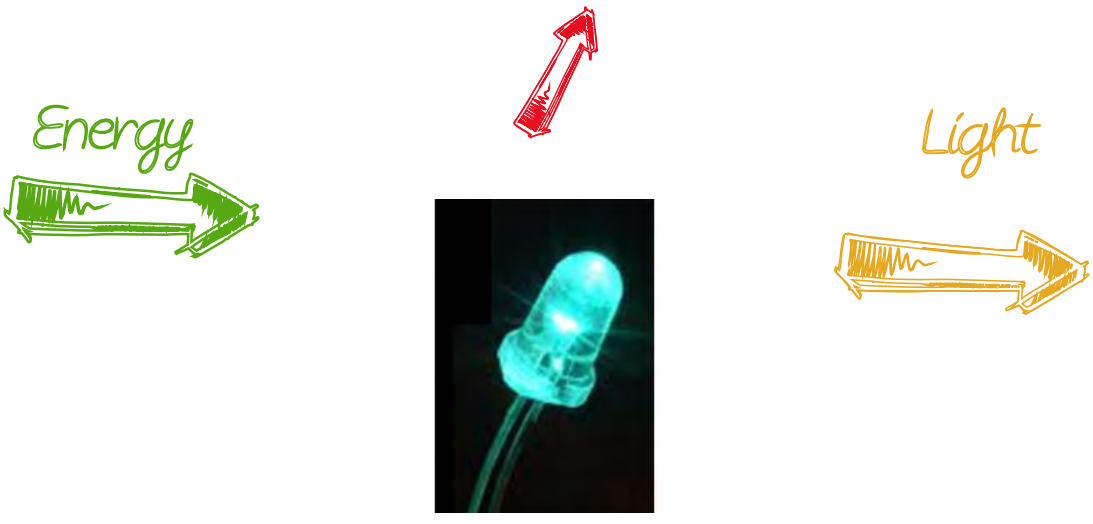




A normal light bulb is a little different due to the fact that most of the electrical energy used to supply it (90%) is transformed into heat. Only 10% of the energy is transformed into light.



In contrast, a LED does not heat up, taking greater advantage of the energy used to supply it, transforming this into light.



The mOway robot can control 4 LEDs independently. Their locations are indicated in the figure:

Exercise 4.1. RC Car with leds

The program consists of activating the LEDs every time the corresponding key is pressed. The "up" and "down" arrows control the front and brake LEDs. The "left" and "right" arrows control the green and red LEDs. The new commands we are going to use are as follows:

Command	Description
CMD_FRONTLEDON	Turns ON the front LED
CMD_FRONTLEDOFF	Turns OFF the front LED
CMD_GREENLEDON	Turns ON the top green LED
CMD_GREENLEDOFF	Turns OFF the top green LED
CMD_BRAKELEDON	Turns ON the brake LED
CMD_BRAKELEDOFF	Turns OFF the brake LED
CMD_REDLEDON	Turns ON the top red LED
CMD_REDLEDOFF	Turns OFF the top red LED
CMD_LEDSON	Turns ON all LEDs
CMD_LEDSOFF	Turns OFF all LEDs

```
import sys, atexit , msvcrt
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
         print 'Moway RFUSB Connected'
else:
    print 'Moway RFUSB not connected. Exit'
    exit(-1)
```

Each time a key is pressed LEDs turn off and then the proper LED is turned on: front for going forward, brake for stop and red and green for right and left.

```
while True:
    if msvcrt.kbhit():
         ch = msvcrt.getch()
         if ch=='w':
              moway.command_moway(CMD_GO_SIMPLE,0)
              moway.command_moway(CMD_LEDSOFF,0)
              moway.command_moway(CMD_FRONTLEDON,0)
         if ch=='z':
              moway.command_moway(CMD_BACK_SIMPLE,0)
              moway.command_moway(CMD_LEDSOFF,0)
              moway.command_moway(CMD_BRAKELEDON,0)
         if ch=='a':
              moway.command_moway(CMD_LEFT_SIMPLE,0)
              moway.command_moway(CMD_LEDSOFF,0)
              moway.command_moway(CMD_GREENLEDON,0)
         if ch=='d':
              moway.command_moway(CMD_RIGHT_SIMPLE,0)
              moway.command_moway(CMD_LEDSOFF,0)
              moway.command_moway(CMD_REDLEDON,0)
```

```
if ch=='s':
      moway.command_moway(CMD_STOP,0)
      moway.command_moway(CMD_LEDSOFF,0)
      moway.command_moway(CMD_BRAKELEDON,0)
```

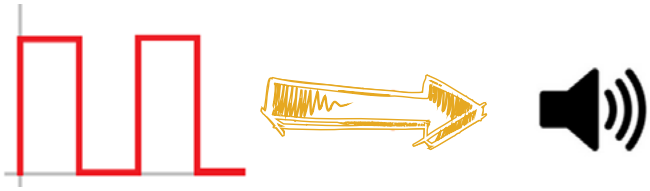
Exercise 4.2. RC Car with LEDs and sound

In the last exercise of this section we are going to make the mOway robot emit a sound when it reverses.

The mOway robot is equipped with a "loudspeaker" or "buzzer". A **buzzer** is a device that emits a sound when it is connected to a variable voltage signal.

The sound is produced when an object vibrates. These vibrations are transmitted through the air and reach our ear, where the eardrum is located. On reaching the eardrum, the vibrations make this move, and in this way we can detect the sounds.

For this reason, when a variable voltage signal is connected to the mOway loudspeaker, the loudspeaker vibrates and transmits this vibration to the air.



These are the commands and variables used for controlling the loudspeaker:

Command	Description	Description
CMD_BUZZERON	Turn ON the buzzer	Frequency
CMD_BUZZEROFF	Turn OFF the buzzer	-

```
import sys, atexit , msvcrt
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
         print 'Moway RFUSB Connected'
else:
    print 'Moway RFUSB not connected. Exit'
    exit(-1)
```

Set the frequency to 440 Hz.

```
moway.set_frequency (440)

while True:
    if msvcrt.kbhit():
         ch = msvcrt.getch()
         if ch=='w':
              moway.command_moway(CMD_GO_SIMPLE,0)
              moway.command_moway(CMD_LEDSOFF,0)
              moway.command_moway(CMD_FRONTLEDON,0)
```

Before going backwards the buzzer turns on and off as some trucks or industrial machines do.

```
         if ch=='z':
                   moway.command_moway(CMD_STOP,0)
              moway.command_moway(CMD_LEDSOFF,0)
              moway.command_moway(CMD_BUZZERON,0)
              sleep(0.5)
              moway.command_moway(CMD_BUZZEROFF,0)
              sleep(0.5)
              moway.command_moway(CMD_BUZZERON,0)
```

```
        sleep(0.5)
        moway.command_moway(CMD_BUZZEROFF,0)
        sleep(0.5)
        moway.command_moway(CMD_BACK_SIMPLE,0)
    if ch=='a':
        moway.command_moway(CMD_LEFT_SIMPLE,0)
        moway.command_moway(CMD_LEDSOFF,0)
        moway.command_moway(CMD_GREENLEDON,0)
    if ch=='d':
        moway.command_moway(CMD_RIGHT_SIMPLE,0)
        moway.command_moway(CMD_LEDSOFF,0)
        moway.command_moway(CMD_REDLEDON,0)
    if ch=='s':
        moway.command_moway(CMD_STOP,0)
        moway.command_moway(CMD_LEDSOFF,0)
        moway.command_moway(CMD_BRAKELEDON,0)
```

5. EXERCISE: SENSORS - SOUND

Now we are going to introduce the "sensor" concept and concentrate on the application and use of one of these: the microphone.

A sensor is an item that allows a robot to discover the world around it. It is somewhat similar to our senses. Thanks to the sensors, the mOway robot can "see", "hear" and "feel". This allows it to stop when it nears an object, move forwards when it detects a sound, turn on a light when going through a tunnel, etc.

A **sensor** is a device capable of detecting physical or chemical magnitudes and transforming these into electrical signals. These electrical signals are read by the mOway microprocessor and are transformed and sent to Scratch in the form of numerical magnitudes for use in our programmes.

In order to measure the amount of sound received, mOway uses an electret type microphone like the one in the picture. This microphone uses a plastic electrode which, when polarised, provokes electrical variations when the sound comes into contact with the microphone.

Electret microphones are used in multiple applications:

- Clip-on microphones (like the ones used on TV)
- Microphone in portable recorders
- Mobile phones

This is the function used to read the value of the microphone:

Function	Returns	Parameters	Description
int get_mic()	0 -100 %	-	Returns the value of the microphone

Exercise 5.1. Clap and Move

When we generate a first loud sound (it may be a shout or clap) mOway will move forward for two seconds using time variable and wait for a new sound. With the second sound, mOway will return to the starting position.

```
import sys, atexit
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
         print 'Moway RFUSB Connected'
else:
    print 'Moway RFUSB not connected. Exit'
    exit(-1)

#set time varibale to 2 secs
moway.set_time(20)
moway.command_moway(CMD_GREENLEDON,0)

while True:
```

While the value readed is less than 40% mOway will remain in its place. If a sound is detected by mOway, it will move forward for two seconds and turn around waiting for another sound to return to its initial position.

```
    while moway.get_mic() < 40 :
         print moway.get_mic()
         sleep(0.1)
    print moway.get_mic()
    moway.command_moway(CMD_GREENLEDOFF,0)
    moway.command_moway(CMD_GO)
    moway.wait_mot_end(0)
    moway.command_moway(CMD_TURN_AROUND,0)
    moway.wait_mot_end(0)
    moway.command_moway(CMD_GREENLEDON,0)
    sleep(0.5)
```

6. EXERCISE: SENSORS. LINE AND OBSTACLES

In this new chapter we are going to concentrate on line and obstacle sensors. Both sensors share this space because they are **based on the same technology**. They are infrared sensors consisting of an LED emitter (such as the ones we saw earlier) and a receiver. Unlike the previous ones, these LED emitters emit a non-visible light, infrared light.

Infrared light emitters and receivers have multiple applications in the world we live in. A very common use is in TV remote controls. They are also used for short distance communications between computers and peripheral devices.

Infrared is also used in night vision equipment, surveillance video or, for example, in rescue equipment when there is an absence of light.

Another of the common uses of these sensors is to detect obstacles. For example, these sensors are used in the doors of garages and lifts to detect when a person or car is passing through to prevent the doors from closing on them.

Line sensors

mOway's two line sensors are made up of an infrared LED transmitter and an infrared receiver. These use **reflected infrared light** to detect the colour (on a grey scale) of the floor where the robot is standing.

As we have stated before, sensors convert physical magnitudes into electrical signals. In this case, the amount of infrared light received, which depends on the colour of the floor, is **converted into an electrical signal** which is read by the mOway microprocessor and sent to the PC.

In this case, mOway translates the amount of light received into numbers. When the surface under mOway is white, a greater amount of light will be reflected than when the surface is black. **mOway interprets and translates those signals**, indicating by a value of 0 to 100 the amount of colour detected, 0 being white and 100 black.

These are the functions used for reading the line sensors:

Function	Returns	Parameters	Description
int get_line_left()	0 -100 %	-	Returns value of the left line sensor
int get_line_right()	0 -100 %	-	Returns value of the right line sensor

In order to facilitate the development and use of the robot in python, mOway incorporates a number of higher-level functions with respect to its line sensors. mOway has functions in which the robot itself is responsible for its movements, using the line sensors to follow a black line on a white background.

If we send mOway a command `"CMD_LINE_FOLLOW_R"`, mOway will automatically begin to read its line sensors and adjust its movement to follow a black line on a white background. The strategy mOway follows to develop this function is to "walk" along the edge, keeping one of its line sensors on the white part and another on the black part. The second part of the command `(_R)`, indicates that mOway will follow the right-hand edge of the black line. If we write `"CMD_LINE_FOLLOW_L"`, mOway will follow the left-hand edge of the line. In the following exercises we will see how this command works.

Command	Description	Variables used
CMD_LINE_FOLLOW_L	mOway follows the a black line in the left border	Speed
CMD_LINE_FOLLOW_R	mOway follows the a black line in the right border	Speed

Obstacle sensors

mOway has **four obstacle** sensors made up of two infrared LED emitters and four infrared receivers. It uses the reflection of infrared light to detect obstacles in its path.

The light emitted by the LEDs strikes against the objects in front of mOway **and is reflected towards the receivers**. If the receivers detect infrared return beams, this will mean that mOway has an obstacle in front of it. In this case, the infrared light received is converted into an electrical signal which is read by the mOway microprocessor and sent to Scratch in accordance with the nearness and size of the object, representing with a value of 0 the absence of any obstacle and an increasing number of up to 100 in the presence of obstacles.

These are the functions used for reading the line sensors:

Function	Returns	Parameters	Description
int get_obs_side_left()	0 -100 %	-	Returns value of the side left obstacle sensor
int get_obs_center_left()	0 -100 %	-	Returns value of the center left obstacle sensor
int get_obs_side_right()	0 -100 %	-	Returns value of the side right obstacle sensor
int get_obs_center_right()	0 -100 %	-	Returns value of the center right obstacle sensor

Exercise 6.1. Enclosed

In this practice denominated "Enclosed", mOway must remain within an enclosed area outlined by black lines drawn on the ground. We will learn how to use line sensors, required to detect the black line.

```
import sys, atexit
from time import sleep
sys.path.append("../lib/")
from moway_lib import *
```

```
if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
        print 'Moway RFUSB Connected'
else:
    print 'Moway RFUSB not connected. Exit'
    exit(-1)
```

The values of both line sensors are added and compared to 50 in order to detect a black line (value near 100) in any sensor. Once the black line is detected mOway turns around. once the turn is finished (`wait_mot_end`) mOway continue moving forward.

```
moway.command_moway(CMD_GO_SIMPLE,0)
while True:
    line = moway.get_line_left() + moway.get_line_right()
    if line > 50:
        moway.command_moway(CMD_TURN_AROUND,0)
        moway.wait_mot_end(0)
        moway.command_moway(CMD_GO_SIMPLE,0)
```

You can add some commands to use the LEDs in this practice: front led while moving forward and brake led while turning around.

```
moway.command_moway(CMD_GO_SIMPLE,0)
moway.command_moway(CMD_FRONTLEDON,0)
while True:
    line = moway.get_line_left() + moway.get_line_right()
    if line > 50:
        moway.command_moway(CMD_LEDSOFF,0)
        moway.command_moway(CMD_BRAKELEDON,0)
        moway.command_moway(CMD_TURN_AROUND,0)
        moway.wait_mot_end(2)
        moway.command_moway(CMD_GO_SIMPLE,0)
        moway.command_moway(CMD_LEDSOFF,0)
        moway.command_moway(CMD_FRONTLEDON,0)
```

Exercise 6.2 Line Follow

In this practice mOway robot follows a black line using only one line sensor. The strategy will be to follow the left border of the line. When the right sensor detects black mOway turns right and when white is detected mOway turns left. Notice that in this case we set the rotation axis to "WHEEL". If the axis is set to "CENTER" mOway will remain in its place rotating, without going forward as in the center rotation one motor goes forward and the other one goes backward.

Note: As the control of mOway robot using python is not as fast as working with the microcontroller itself the black line must be wide enough to give time to mOway to detect its variation.

```
import sys, atexit
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
        print 'Moway RFUSB Connected'
else:
    print 'Moway RFUSB not connected. Exit'
    exit(-1)

moway.set_rotation_axis(WHEEL)
while True:
    line_r = moway.get_line_right()
    if line_r < 50:
        moway.command_moway(CMD_ROTATERIGHT,0)
    else:
        moway.command_moway(CMD_ROTATELEFT,0)
```

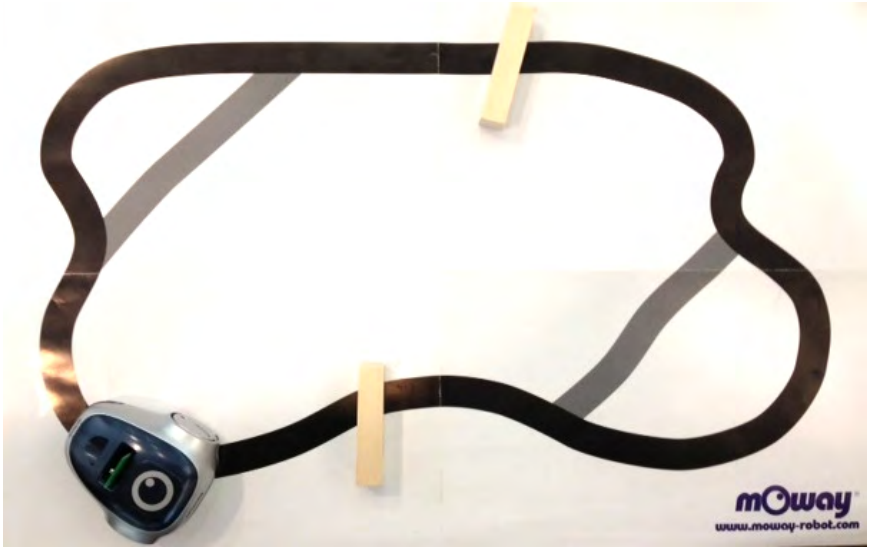
Exercise 6.3. Obstacle detection

This exercise is very similar to the first one in this unit. We change the obstacle sensors instead of line sensor. In this case mOway will turn around whenever it detects an obstacle.

```
import sys, atexit
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
        print 'Moway RFUSB Connected'
else:
    print 'Moway RFUSB not connected. Exit'
    exit(-1)

moway.command_moway(CMD_GO_SIMPLE,0)
while True:
    obstacle = moway.get_obs_center_left() + moway.get_obs_center_
right()
    if obstacle > 0:
        moway.command_moway(CMD_TURN_AROUND,0)
        moway.wait_mot_end(0)
        moway.command_moway(CMD_GO_SIMPLE,0
```

Exercise 6.4. Obstacle detection

In the next exercise we are going to make use of both sensors: line and obstacles. On the same track with a black line from the previous exercise, we will introduce a number of obstacles as shown in the illustration. The aim of this exercise is for mOway to follow the line and, when it finds an obstacle, to a turn around and continue following the line in the opposite direction. If we execute this program in the environment shown in the picture below, mOway will bounce, following the line between one obstacle to the other.



In this exercise we are going to need a turn of a little more than 180 degrees so that when mOway sees an obstacle and has to turn, it can visualize the black line with its sensors. The first time the program is attempted, the "turnaround" command can be used to see how mOway sometimes does not find the line on its return.

import sys, atexit

```
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
         print 'Moway RFUSB Connected'
else:
    print 'Moway RFUSB not connected. Exit'
    exit(-1)

moway.set_rotation(210)
while True:
    moway.command_moway(CMD_LINE_FOLLOW_L,0)
    obstacle = moway.get_obs_center_left() + moway.get_obs_center_
```

```
right() + moway.get_obs_side_left() + moway.get_obs_side_right()
     if obstacle > 0:
          moway.command_moway(CMD_ROTATELEFT,0)
          moway.command_moway(CMD_BRAKELEDON,0)
          moway.wait_mot_end(0)
          moway.command_moway(CMD_BRAKELEDOFF,0)
```

Exercise 6.5. Defender

The final exercise in this chapter is highly entertaining and interesting for students. In this exercise we will also make use of the line and obstacle sensors, but for a different purpose. For our physical scenario, we can use the track utilized in previous exercises as long as this line is closed.

mOway will remain within its area (inside the black line) and will try to push any foreign objects it finds in its area. In this exercise, we will introduce the use of a new command "CMD_PUSH". When mOway moves forward, looking for objects and the green LED on, it will do this in a straight line until it reaches the end of its area or finds an object. If it reaches the end of the line, it will move back a little in order not to go out of its area, turn to the right and continue to move forward in search of new "invaders". Once it finds an object, it will push this until it reaches the end of its area and the object is left outside.

```
import sys, atexit
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
        print 'Moway RFUSB Connected'
else:
        print 'Moway RFUSB not connected. Exit'
        exit(-1)

moway.command_moway(CMD_GO_SIMPLE,0)
moway.command_moway(CMD_GREENLEDON,0)
```

```
moway.set_rotation(144)
```

If mOway finds a black line will go back a bit and turn around.

```
while True:
        if moway.get_line_left() + moway.get_line_right() > 50:
                moway.set_time(3)
                moway.command_moway(CMD_BACK,0)
                moway.wait_mot_end(0)
                moway.set_time(0)
                moway.command_moway(CMD_ROTATERIGHT,0)
                moway.wait_mot_end(0)
                moway.command_moway(CMD_GO_SIMPLE,0)
```

If mOway detects and obstacle in any of the sensors, mOway turns on front and red LEDs and pushes the "intruder" until the black line is detected.

```
       else:
                    obstacle = moway.get_obs_center_left() + moway.
get_obs_center_right()  +  moway.get_obs_side_left()  +  moway.get_
obs_side_right()
                if obstacle > 0:
                        moway.command_moway(CMD_PUSH,0)
                        moway.command_moway(CMD_LEDSOFF,0)
                        moway.command_moway(CMD_FRONTLEDON,0)
                        moway.command_moway(CMD_REDLEDON,0)
                        while moway.get_line_left() + moway.get_
line_right() < 50:
                                print 'Pushing'
                        moway.command_moway(CMD_LEDSOFF,0)
                        moway.command_moway(CMD_GREENLEDON,0)
```

7. EXERCISE: LIGHT SENSOR

A light sensor is a device that measures the amount of light in a location. For example, in cars, it can be used to turn on the lights automatically in tunnels or at dusk.

Light sensors are generally based on an element called a photodiode. This electronic component allows more or less current to pass through it in accordance with the amount of light received.

Light sensors have many applications in our environment. For example, have you noticed that mobile phone screens dim when we turn off the lights? In this way it saves batteries thanks to the light sensor similar to the one fitted in the mOway robot.

Exercise 7.1. Car Lights

In this exercise we will make mOway follow the black line while it reads the light sensor. When it goes through a tunnel (low light condition), the front LED will turn on. We will print light sensor value in the screen each second.

```
import sys, atexit
from time import sleep
sys.path.append("../lib/")
from moway_lib import *

if __name__ == '__main__':
        atexit.register(exit_mow)

channel = 7
moway.usbinit_moway()
ret = moway.init_moway(channel)
if ret == 0:
         print 'Moway RFUSB Connected'
else:
```

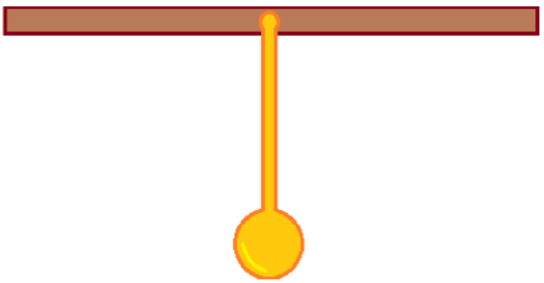
```
    print 'Moway RFUSB not connected. Exit'
    exit(-1)

while True:
    moway.command_moway(CMD_LINE_FOLLOW_L,0)
    light = moway.get_light()
    print light
    if light < 40:
         moway.command_moway(CMD_FRONTLEDON,0)
    else:
         moway.command_moway(CMD_FRONTLEDOFF,0)
    sleep(1)
```

8. EXERCISE: ACCELEROMETER

An accelerometer is a device that measures the inclination of an object. It can be found in mobile phones, video console controls, and so on,...

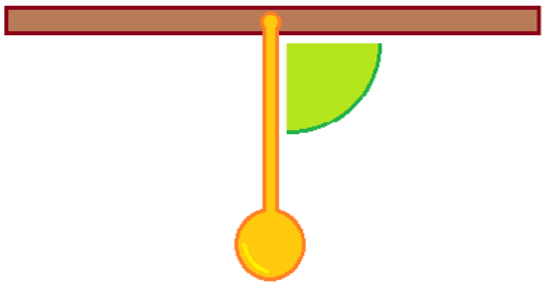
When you turn a mobile phone, the image on the screen switches to the correct position. In other words, a mobile phone "knows" that when it is turned it has to change the image so that it can be read correctly. How does the mobile phone "know" what position it is in? In other words, how can we measure the inclination have an object?

Imagine we have a board from which a pendulum is hung. Due to the force of gravity, the pendulum is always going to be in a vertical position. What happens if we place the board in different positions?

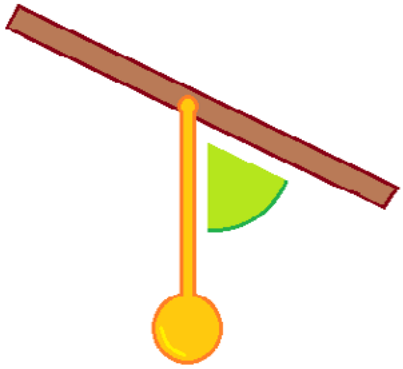


A number of different positions are shown below. The angle formed by the board and the pendulum is shown in green.

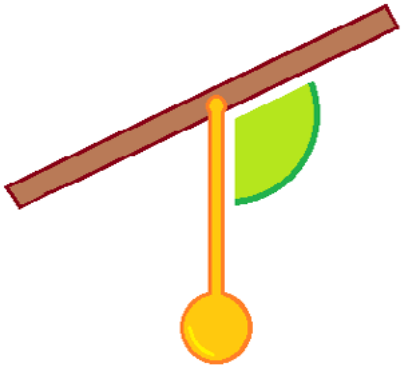
If the board is in a horizontal position, the board and the pendulum form a right angle. In the following drawing, this angle is indicated in green:



If the board is tilted towards the right, the angle is less than 90°. In the following drawing, the angle is approximately 45°:



If the board is inclined towards the left, the angle formed is greater than 90°. In this drawing, the angle is approximately 120°:



In reality, an accelerometer measures accelerations. Any force is an acceleration, including the force of gravity. The accelerometer inside mOway is a kind of "pendulum" which allows the robot to determine whether it is tilted and on what side it is tilted. This "pendulum" is capable of detecting the inclination in 3 axes.

In other words, if we tilt the robot in the following ways:

Inclination	Forwards and backwards	Right to left	Up and down
Axis	Y axis	X axis	Z axis
Positions			

In this exercise we are going to use the mOway robot as a videogames remote control. By tilting the robot towards the left or the right, the figure in the video game will move in that direction.

To do this, it is necessary to determine the accelerometer value. We are going to control the figure by tilting the mOway robot towards the right and left.



The programmes in this exercise feature several figures or objects. Each of these figures and objects have their own programme.

These are the functions for retrieving the accelometer data from mOway:

Function	Returns	Parameters	Description
float get_accel_X()	+/- 2 g	-	Returns the value of the accelerometer X-axis
float get_accel_Y()	+/- 2 g	-	Returns the value of the accelerometer Y-axis
float get_accel_Z()	+/- 2 g	-	Returns the value of the accelerometer Z-axis

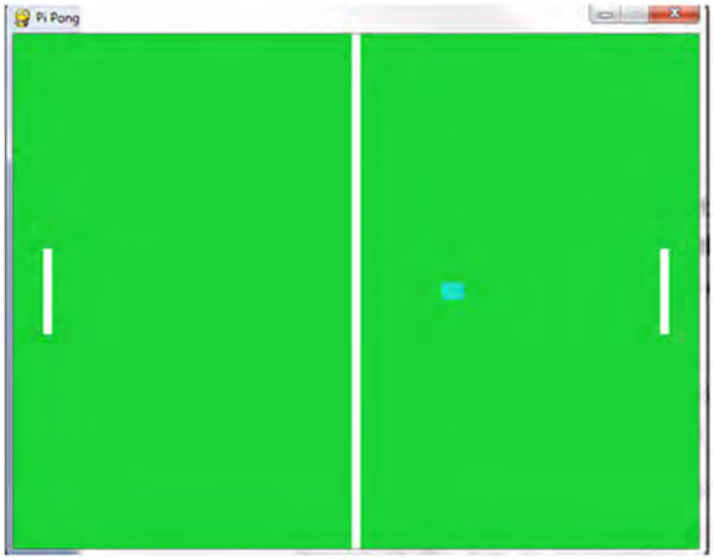
Exercise 8.1. Plot accelerometer

In this exercise we are going to use the mOway accelerometer like a remote sensor and we will paint the magnitude of the acceloremeter data received in the screen.

This example is programmed in Linux using pygtk, matplotlib, and numpy libraries.

Exercise 8.2. Pong game

The next exercise is a recreation of the classic game "Pong", with one player using moway as a controller and the other one using the keyboard (up and down keys). The game consists of a ball that bounces across the screen and players have to prevent the ball from touching the wall behind it.



In the projects folder the complete source code of this exercise is available. The part of code involving mOway controller is:

```
#Set bat 1 position using accelerometer
move_x = moway.get_accel_X()
print move_x
#speed of the bat is proportional to the inclination of x-axis
self.player1Bat.speed = abs(int(move_x*26))
#if mOway robot is nearly plain the bat does not move
if move_x > 0.1:
    self.player1Bat.startMove("up")
elif move_x < -0.1:
    self.player1Bat.startMove("down")
else:
    self.player1Bat.stopMove()
```

9. ANNEX I: Libmoway functions

Function	Returns	Parameters	Description
usbinit_moway()	-	-	Initialization of libmoway communication library
exit_moway()	-	-	Exit of libmoway library
int init_moway(uint8_t channel)	0: OK -1: error	1-100	RFUSB connection and channel selection
void close_moway()	-	-	Closes RFUSB connection
int command_ moway(uint8_t command, int timeout);	0: Received 1: Repeated 2: Timeout	Command: See list of commands Timeout: in ms	Sends a command to mOway Robot
void set_speed (int speed)	-	Speed: 0-100%	Sets distance variable
void set_rotation (int rotation)	-	rotation: 0 -360	Sets rotation variable
void set_distance (uint8_t distance)	-	distance: 0 - 255 mm	Sets distance variable
void set_radius (uint8_t radius)	-	radius: 0 -100	Sets radius variable
void set_rotation_axis (uint8_t axis)	-	axis: WHEEL - CENTER	Sets rotation axis to WHEEL or CENTER
void set_time (uint8_t time)	-	time: 0 - 255 ms	Sets time variable
void set_frequency (int frequency)	-	frequency: 0 -20.000 Hz	Sets frequency of the buzzer
int wait_mot_end(int timeout)	0: Finished -1: Not started -2: Timeout	timeout - secs	Waits for motor movement to complete
int get_obs_side_left()	0 -100 %	-	Returns value of the side left obstacle sensor

Function	Returns	Parameters	Description
int get_obs_center_left()	0 -100 %	-	Returns value of the center left obstacle sensor
int get_obs_side_right()	0 -100 %	-	Returns value of the side right obstacle sensor
int get_obs_center_right()	0 -100 %	-	Returns value of the center right obstacle sensor
int get_line_left()	0 -100 %	-	Returns value of the left line sensor
int get_line_right()	0 -100 %	-	Returns value of the right line sensor
int get_mic()	0 -100 %	-	Returns the value of the microphone
int get_light()	0 -100 %	-	Returns the value of the light sensor
int get_distance()	0 -65535 (mm)	-	Returns the distance counter of the motors
float get_accel_X()	+/- 2 g	-	Returns the value of the accelerometer X-axis
float get_accel_Y()	+/- 2 g	-	Returns the value of the accelerometer Y-axis
float get_accel_Z()	+/- 2 g	-	Returns the value of the accelerometer Z-axis
int moway_active()	0: Inactive > 0: Active	-	Returns mOway state
int init_prog_moway()	0: OK -1: Error	-	Initialization of libmoway programming library
int program_moway (char * file)	0: OK -1: Error -2: File not found	-	Download an hex file into mOway robot
int program_moway_ channel (char * file, int channel)	0: OK -1: Error -2: File not found	-	Download python programming hexfile into mOway robot with channel selection
int read_moway_batt()	0-100%	-	Returns mOway battery when mOway is connected via USB

10. ANNEX II: Libmoway commands

Simple movement Commands

Command	Description	Variables used
CMD_GO_SIMPLE	mOway goes forward indefinitely	-
CMD_BACK_SIMPLE	mOway goes backwards indefinitely	-
CMD_STOP	mOway stops	-
CMD_LEFT_SIMPLE	mOway turns 90 degree left	axis
CMD_RIGHT_SIMPLE	mOway turns 90 degree right	axis
CMD_TURN_AROUND	mOway turns around	axis

Action Commands

Command	Description	Variables used
CMD_RESET_DIST	Reset the distance counter of the motor	-
CMD_FRONTLEDON	Turns ON the front LED	-
CMD_FRONTLEDOFF	Turns OFF the front LED	-
CMD_GREENLEDON	Turns ON the top green LED	-
CMD_GREENLEDOFF	Turns OFF the top green LED	-
CMD_BRAKELEDON	Turns ON the brake LED	-
CMD_BRAKELEDOFF	Turns OFF the brake LED	-
CMD_REDLEDON	Turns ON the top red LED	-
CMD_REDLEDOFF	Turns OFF the top red LED	-
CMD_FRONTBLINK	Blinks front LED	-
CMD_BRAKEBLINK	Blinks brake LED	-
CMD_GREENBLINK	Blinks top green LED	-
CMD_REDBLINK	Blinks top red LED	-
CMD_LEDSON	Turns ON all LEDs	-

Command	Description	Variables used
CMD_LEDSOFF	Turns OFF all LEDs	-
CMD_LEDSBLINK	Blinks all LEDs	-
CMD_BUZZERON	Turn ON the buzzer	frequency
CMD_BUZZEROFF	Turn OFF the buzzer	-
CMD_MELODYCHARGE	Plays "charge" music	-
CMD_MELODYFAIL	Plays "fail" music	-
CMD_LINE_FOLLOW_L	mOway follows the a black line in the left border	speed
CMD_LINE_FOLLOW_R	mOway follows the a black line in the right border	speed
CMD_PUSH	mOway pushes an obstacle in front	-

Full movement commands

This movement commands can use a limited time or distance. If time and distance are set to zero the movements continue indefinitely. In case both parameters are different from zero the distance or angle parameter is predominant.

Command	Description	Variables used
CMD_GO	mOway goes forward for a set time or distance	speed, time, distance,
CMD_BACK	mOway goes backwards for a set time or distance	speed, time, distance,
CMD_GOLEFT	mOway goes forward and left for a set time or distance	speed, time, distance, radius
CMD_GORIGHT	mOway goes forward and right for a set time or distance	speed, time, distance, radius
CMD_BACKLEFT	mOway goes backwards and left for a set time or	speed, time, distance, radius
CMD_BACKRIGHT	mOway goes backwards and right for a set time or	speed, time, distance, radius
CMD_ROTATELEFT	mOway turns left for a set time or angle	speed, time, rotation, axis
CMD_ROTATERIGHT	mOway turns left for a set time or angle	speed, time, rotation, axis

11. ANNEX III: Install Moway python Libraries in Windows systems

1. Install python v2.7 for your operating system from http://www.python.org/download/. NOTE: Python v3 could also be used but it will require little changes in the source files.

2. Download moway_python_pack.zip from www.moway-robot.com to your home folder and extract it.

3. Install vcredist_x86 or vcredist_x64 (depending on you OS 32 or 64 bits) located in the "install_windows" folder in moway_python_pack.

4. If it is your first mOway installation, you have to install the drivers of mOway RFUSB located in "RFUsb Driver" once you plug the RFUSB.

5. If you create new files with python and mOway create them in the projects folder. If you create them in another folder you may have to change moway_lib and the header of the file with the locations of the libraries.

6. In order to start working with mOway robot and python, firmware must be downloaded first into mOway. To download software you have to run "firm_download.py" program located in projects folder. You can edit channel variable in this program. The channel selected in the download is the channel that we will use in the future in our python programs.

12. ANNEX IV: Install Moway python Libraries in Linux

1. Download moway_python_pack.zip from www.moway-robot.com to your home folder and extract it.

2. If you already have some mOway software working on your computer you can avoid this step. if not you must copy the udev rules locates in install_linux/rules files to the /etc/udev/rules.d/ folder. You must do this as root.

3. If you create new files with python and mOway create them in the projects folder. If you create them in another folder you may have to change moway_lib and the header of the file with the locations of the libraries.

4. In order to start working with mOway robot and python, firmware must be downloaded first into mOway. To download software you have to run "firm_download.py" program located in projects folder. You can edit channel variable in this program. The channel selected in the download is the channel that we will use in the future in our python programs.