TDDD63 Project: API

Martin Söderén

September 13, 2014

Contents

1 Using the API

You will build programs for the robot to run in the main_program.py where all the neccesary modules have been imported for you. It should look something like this:

```
from system.sensors import touch,motor,speak,IR,color

def main(brick):
    #TODO make a program for the ev3
```

The structure of the API is the following.

Function:
Displays the function.

Argument:
Shows you what the parameters are and what they stand for.

Ports:
The description of the constructors will have this heading that will tell which ports you can connect that sensor to because not every sensor can be connected to every sensorport.

Return:
Tells you what value the function will return when calling it.

Description:
Gives you additional information about the function that may be good to know.

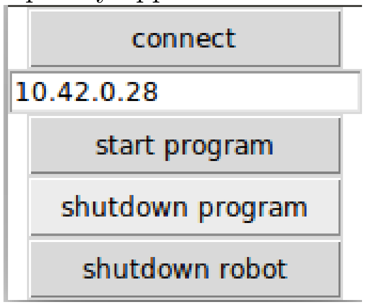
Example:
Example code showing you how to call the function.

2 Connection and Status

2.1 how to connect through USB

Make shure the ev3 is connected your computer with an USB-cabel. If the ev3 has batteries in it it should only be a matter of pressing any button for it to start. Wait for it to complete the bootseqence which will be indicated by the two leds on the front both being continously green. Go into the code skeleton which have been given to you by your labassistent. Start the python script named "start.py" with whatever application you prefer.

A small window will hopefully appear which should look something like:

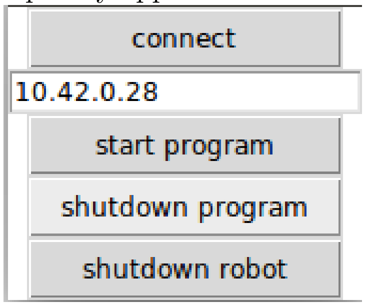


The first time you run this script the adress in the entry will be 0.0.0.0 but after this a config file will be created and the latest ip-adress used will be loaded in automatically. The ip-adress when you connect through USB should be 10.42.0.28. Enter the adress into the entry and press connect. If the connection is succesfull the ”connect” button will turn green. After this you can press the ”start program” button and the program which you have put in the ”main_program.py” file will be executed by the robot. When the program is finished you can press the ”start program” button again to run it again. Or you can edit your program and press the button again without having to run the ”start.py” script again because when you press the button the programs automatically reloads the main_program module.

2.2 how to connect through wlan

Make shure the ev3 is connected your computer with an USB-cabel. If the ev3 has batteries in it it should only be a matter of pressing any button for it to start. Wait for it to complete the bootseqence which will be indicated by the two leds on the front both being continously green. Go into the code skeleton which have been given to you by your labassistent. Start the python script named ”start.py” with whatever application you prefer.

A small window will hopefully appear which should look something like:



The first time you run this script the adress in the entry will be 0.0.0.0 but after this a config file will be created and the latest ip-adress used will be loaded in automatically. The ip-adress when you connect through USB should be 10.42.0.28. Enter the adress into the entry and press connect. If the connection is succesfull the ”connect” button will turn green. After this you can press the ”start program” button and the program which you have put in the ”main_program.py” file will be executed by the robot. When the program is finished you can press the ”start program” button again to run it again. Or you can edit your program and press the button again without having to run the ”start.py” script again because when you press the button the programs automatically reloads the main_program module.

2.3 Shutdown program

The program can be shutdown either by pressing the ”shutdown program” button in the ”robot commands” windows or by simple pressing alt+F4 but the ”shutdown program” button is recommended. It will throw some exceptions from the of type tkinter.*. Thats nothing to worry about.

2.4 Shutdown the robot

When you are done for the day. Simple press the ”shutdown robot” in the ”robot commands” window and wait until the robots is completely shutdown. Try to avoid turning the robot off by taking the batteries out because there is a risk of memory corruption.

3 Sensors

3.1 touch sensor

3.1.1 touch

Function:

```
touch(brick,port)
```

Argument:
Brick: an threadobject that is used to access the robot.
Port: an integer from 1 to 4 to tell which port the sensor is connected to

Ports:
2-4

Return:
A sensorobject of class touch

Description:
Create sensorobject

Example:

```
from touch import touch
def main(brick):
    touch_sensor=touch(brick,3)
```

3.1.2 is_pressed

Function:

```
is_pressed()
```

Argument:
None

Return:
Boolean. True if the sensor is pressed, else False

Description:
A method of the touch-class to get the value from the sensor

Example:

```
from touch import touch
def main(brick):
    touch_sensor=touch(brick,3)
    print(touch_sensor.is_pressed())
```

3.2 color sensor

3.2.1 color

Function:

```
color(brick,port)
```

Argument:
Brick: an threadobject that is used to access the robot.
Port: an integer from 1 to 4 to tell which port the sensor is connected to

Ports:
2

Return:
A sensorobject of class color

Description:
Creates sensorobject that is default in ambient-mode

Example:

```
from color import color
def main(brick):
    color_sensor=color(brick,2)
```

3.2.2 set_reflect_mode

Function:

```
set_reflect_mode()
```

Argument:
None

Return:
None

Description:
A method for the color-class. Set the mode of the color-object to reflect

Example:

```
from color import color
def main(brick):
    color_sensor=color(brick,2)
    color_sensor.set_reflect_mode()
```

3.2.3 set_ambient_mode

Function:

```
set_ambient_mode()
```

Argument:
None

Return:
None

Description:
A method for the color-class. Set the mode of the color-object to ambient

Example:

```
from color import color
def main(brick):
    color_sensor=color(brick,2)
    color_sensor.set_ambient_mode()
```

3.2.4 set_color_mode

Function:

```
set_color_mode()
```

Argument:
None

Return:
None

Description:
A method for the color-class. Set the mode of the color-object to color

Example:

```
from color import color
def main(brick):
    color_sensor=color(brick,2)
    color_sensor.set_color_mode()
```

3.2.5 set_rgb_raw_mode

Function:

```
set_rgb_raw_mode()
```

Argument:
None

Return:
None

Description:
A method for the color-class. Set the mode of the color-object to rgb-raw

Example:

```
from color import color
def main(brick):
    color_sensor=color(brick,2)
    color_sensor.set_rgb_raw_mode()
```

3.2.6 get_value

Function:

```
get_value()
```

Argument:
None

Return:
Mode=reflect:int
The value represents how much of the light from a led on the sensors that is reflected. If you hold a mirror in front of the sensor you will get a high value)

Mode=ambient:int
The value represents how bright it is in front of the sensor. If you shine a lamp right into the sensor you will get a high value

Mode=color:int
Returns a int from 0 to 7 depending which color the object in front of the sensor has. The color can be extracted from the following list:

```
["NONE", "BLACK", "BLUE", "GREEN", "YELLOW", "RED", "WHITE", "BROWN"]
```

If the value is the index of the list.

Mode=rgb_raw:[int,int,int]
The first int represent how red the object in front of the sensor is. The second how green it is and the third how blue it is, aka RGB values. Each ranges from 0 to 255.

Description:
A method of the color-class to get the value from the sensor

Example:

```
from color import color
def main(brick):
    color_sensor=color(brick,2)
    print(color_sensor.get_value())
```

More advanced example:

```
import time
from color import color
def main(brick):
    color_sensor=color(brick,2)
    print("ambient")
    for i in range(10):
        print(color_sensor.get_value())
    color_sensor.set_color_mode()
    time.sleep(1)
    print("color")
    for i in range(10):
        print(color_sensor.get_value())
    color_sensor.set_reflect_mode()
    time.sleep(1)
    print("reflect")
    for i in range(10):
        print(color_sensor.get_value())
    color_sensor.set_rgb_raw_mode()
    time.sleep(1)
    print("rgb")
    for i in range(10):
        print(color_sensor.get_value())
```

3.2.7 get_color_string

Function:

```
get_color_string()
```

Argument:
None

Return:
String

Description:
A method of the color-class that returns the color in front of the color sensor in form of a string, for example "BLACK".

Example:

```
from color import color
def main(brick):
    color_sensor=color(brick,2)
    print(color_sensor.get_color_string())
```

3.3 IR sensor

3.3.1 IR

Function:

```
IR(brick,port)
```

Argument:

Brick: an threadobject that is used to access the robot.
Port: an integer from 1 to 4 to tell which port the sensor is connected to

Ports:

2-4

Return:

A sensorobject of class IR

Description:

Creates sensorobject which default mode is prox

Example:

```
from IR import IR
def main(brick):
    ir_sensor=IR(brick,4)
```

3.3.2 set_proximity_mode

Function:

```
set_proximity_mode()
```

Argument:

None

Return:

None

Description:

A method for the IR-class. Set the mode for the IR-object to proximity-mode

Example:

```
from IR import IR
def main(brick):
    ir_sensor=IR(brick,4)
    ir_sensor.set_proximty_mode()
```

3.3.3 set_seek_mode

Function:

```
set_seek_mode()
```

Argument:
None

Return:
None

Description:
A method for the IR-class. Set the mode for the IR-object to seek-mode

Example:

```
from IR import IR
def main(brick):
    ir_sensor=IR(brick,4)
    ir_sensor.set_seek_mode()
```

3.3.4 set_remote_control_mode

Function:

```
set_remote_control_mode()
```

Argument:
None

Return:
None

Description:
A method for the IR-class. Set the mode for the IR-object to remote_control-mode

Example:

```
from IR import IR
def main(brick):
    ir_sensor=IR(brick,4)
    ir_sensor.set_remote_control_mode()
```

3.3.5 get_value

Function:

```
get_value()
```

Argument:

None

Return:

Mode=prox:int

The value represent how far away the object in front of the sensor is. Higher values means the object is further away.

Mode=seek:[(int, int), (int, int), (int, int), (int, int)]

This gives the distance and the degree to the remote control if its have been put in seek mode which you do by pushing the largest button which makes a small green led shine. You can also choose between 4 different channels on the remote and the distance and the degree to a remote on channel 1 will be put in the first tuple in the list. Channel 2 in the seconds and so on so you can track 4 remotes at the same time.

Mode=remote_control:(int, int, int, int)

The integers show if a button is pressed on a remote. The first int is for remotes on channel 1 and the second for remotes on channel 2 and so on.
Large button:9
Red up:1
Red down:2
Blue up:3
blue down:4

Description:

A method of the IR-class to get the value from the sensor

Example:

```
from IR import IR
def main(brick):
    ir_sensor=IR(brick,4)
    print(ir_sensor.get_value())
```

More advanced example:

```
from IR import IR
def main(brick):
    ir_sensor=IR(brick,4)
    print("prox")
    for i in range(10):
        print(ir_sensor.get_value())
    ir_sensor.set_seek_mode()
    time.sleep(1)
    print("seek")
    for i in range(10):
        print(ir_sensor.get_value())
    ir_sensor.set_remote_control_mode()
    time.sleep(1)
    print("remote")
```

```
          for i in range(10):
              print(ir_sensor.get_value())
```

3.3.6 speak

Function:

```
speak(brick)
```

Argument:
Brick: an threadobject that is used to access the robot.

Return:
A sensorobject of class speak

Description:
Create sensorobject

Example:

```
from speak import speak
def main(brick):
    speak_sensor=speak(brick)
```

3.3.7 talk

Function:

```
talk(speech,lang="en")
```

Argument:
Speech: an string that contains what the robot should say
Lang: a volontary string that is used for choosing accent of the voice. The choices are languages that Linux eSpeak supports as standard.

Return:
Nothing

Description:
A method of the speak-class to make the robot talk

Example:

```
from speak import speak
def main(brick):
    speak_sensor=speak(brick)
    speak_sensor.talk("hello world",lang="en")
```

4 Motors

4.1 motor

Function:

```
motor(brick,port)
```

Argument:
brick: an threadobject that is used to access the robot.
port: a string with values "A","B","C" or "D" to tell which port the sensor is connected to

Ports:
A-D

Return:
Object of class motor

Description:
Creates a object of class motor to be used for controlling the ev3s motors. Don't send commands to the motors to often beacuse it will handle every commands and it takes some time which might cause a delay. Send no more than 4 commands per second.

Example:

```
from motor import motor
def main(brick):
    motor_A=motor(brick,"A")
```

4.2 run_forever

Function:

```
run_forever(speed)
```

Argument:
speed: an int from -100 to 100 that tells the robot how fast and which direction to rotate the motor

Return:
Nothing

Description:
A method for the motor class that makes the motor run forever

Example:

```
from motor import motor
def main(brick):
    motor_A=motor(brick,"A")
    motor_A.run_forever(100)
```

4.3 run_time_limited

Function:

```
run_time_limited(speed, time)
```

Argument:
speed: an int from -100 to 100 that tells the robot how fast and which direction to rotate the motor
time: an int larger than 0 that tells the motor how long time to run (ms)

Return:
Nothing

Description:
A method for the motor class that makes the motor run at a certain time and speed

Example:

```
from motor import motor
def main(brick):
    motor_A=motor(brick,"A")
    motor_A.run_time_limited(100,5000)
```

4.4 run_position_limited

Function:

```
run_position_limited(speed, position)
```

Argument:
speed: an int from -100 to 100 that tells the robot how fast and which direction to rotate the motor
position: an int that tells the motor how many degress to rotate

Return:
Nothing

Description:
A method for the motor class that makes the motor run rotate a certain amount of degrees at a certain speed. The slower you run the motor the more accurate it will be. If it is run at 100 it will most certainly overshoot the target degree. If you need accurancy do not go above 50.

Example:

```
from motor import motor
def main(brick):
    motor_A=motor(brick,"A")
    motor_A.run_position_limited(100,360)
```