Virtual Copernicus NG

A tool for simulating physical electronic devices through graphical user interface.

Copernicus is an implement used during IoT classes at AGH University of Science and Technology. You can learn more about the project here.

This package was created by Jonatan Kłosko, Mateusz Benecki, Dominik Kawalec and Tomasz Rosiek as a form of crediting the IoT classes. It is based on tkgpio by @wallysalami.

Installation

```
# You can optionally create and enable a virtual environment
# to avoid installing the packages globally
python -m venv venv
source venv/bin/activate

# Install Virtual Copernicus NG
pip install git+https://github.com/dkawalecc/virtual_copernicus_ng.git
```

Usage

```
from virtual_copernicus_ng import TkCircuit

# Step 1: Define a virtual circuit by listing its elements and their parameters

configuration = {
    "name": "CopernicusNG SmartHouse",
    "sheet": "sheet_smarthouse.png",
    "width": 332,
    "height": 300,
    "leds": [
        {"x": 112, "y": 70, "name": "LED 1", "pin": 21}
    ],
    "buttons": [
        {"x": 242, "y": 146, "name": "Button 1", "pin": 11}
    ]
}

# Step 2: Create the virutal circuit based on the above definition.
# It builds a GUI representing the real circuit.

circuit = TkCircuit(configuration)

@circuit.run
def main():
    # Step 3: Write a program the same way you would do it
```

```
    # on a Raspberry Pi board.

    from gpiozero import LED, Button
    from time import sleep

    led1 = LED(21)

    def handle_button1_press():
        print("button 1 pressed!")
        led1.toggle()

    button1 = Button(11)
    button1.when_pressed = handle_button1_press

    while True:
        sleep(0.1)
```

Examples

Complete examples can be found in the `examples` directory.

Available virtual elements

LED

Circuit configuration

```
configuration = {
    # ...
    "leds": [
        {"x": 112, "y": 70, "name": "LED 1", "pin": 21}
    ],
    # ...
}
```

Usage

```
from gpiozero import LED

led1 = LED(21)
led1.toggle()
```

Button

Circuit configuration

```
configuration = {
    # ...
    "buttons": [
        {"x": 242, "y": 146, "name": "Button 1", "pin": 11},
    ],
    # ...
}
```

Usage

```
from gpiozero import Button

def handle_button1_press():
    print("button 1 pressed!")

button1 = Button(11)
button1.when_pressed = handle_button1_press
```

Buzzer

Circuit configuration

```
configuration = {
    # ...
    "buzzers": [
        {"x": 277, "y": 9, "name": "Buzzer", "pin": 16, "frequency": 440},
    ],
    # ...
}
```

Usage

```
from gpiozero import Buzzer

buzzer = Buzzer(16)
buzzer.on()
```

Servo

Circuit configuration

```
configuration = {
    # ...
    "servos": [
```

```
        {"x": 170, "y": 150, "length": 90, "name": "Servo 1", "pin": 17,
"min_angle": 0, "max_angle": 180}
    ],
    # ...
}
```

Usage

```
from gpiozero import AngularServo

servo = AngularServo(17, min_angle=0, max_angle=180)
servo.angle = 90
```

Analog-to-digital converter (MCP3002)

Circuit configuration

```
configuration = {
    # ...
    "mcp3002s": [
        {"x": 34, "y": 160, "name": "ADC 1", "clock_pin": 11, "mosi_pin": 10,
"miso_pin": 9, "select_pin": 8, "max_voltage": 5},
    ],
    # ...
}
```

Usage

```
from gpiozero import MCP3002

pot = MCP3002(1, max_voltage=5, clock_pin=11, mosi_pin=10, miso_pin=9,
select_pin=8)

while True:
    print(f"pot: value = {pot.value:0.2f}\tvoltage = {pot.voltage:0.2f}\traw value
= {pot.raw_value:0.2f}")

    sleep(0.1)
```

Virtual MCP3002

The primary project objective was to extend the existing Virtual Copernicus NG with a virtual analog-to-digital converter, namely MCP3002.

Such device takes an analog signal on its input (a voltage level from a continuous range) and produces a digital signal on its output (several bits representing the voltage level converted into a discrete range). A possible use case is introducing a potentiometer to a digital circuit. A potentiometer is a source of an analog signal in a fixed range, so we would connect it to an ADC's input and then connect the ADC's output (via pins) to our digital circuit.

MCP3002 is a concrete ADC device, which converts the input analog signal to a 10 bit number. The device communicates with the microprocessor system via a serial interface - SPI. This interface uses 4 lines (i.e. pins) for communication.

Implementation-wise, reading the current MCP3002 value involves sending a request over the SPI interface and receiving a series of bits. The virtual MCP3002 mimics the physical behavior by using virtual pins to effectively intercept the SPI requests and reply with a value taken from the UI (specifically a slider representing a potentiometer).