

```
In [22]: from pyng.overlays.base import BaseOverlay
import time
from datetime import datetime
base = BaseOverlay("base.bit")
import threading
btns = base.btns_gpio
import random
import socket
```

```
In [23]: %%microblaze base.PMODB

#include "gpio.h"
#include "pyprintf.h"

//Function to turn on/off a selected pin of PMODB
void write_gpio(unsigned int pin, unsigned int val){
    if (val > 1){
        pyprintf("pin value must be 0 or 1");
    }

    gpio pin_out = gpio_open(pin);
    gpio_set_direction(pin_out, GPIO_OUT);
    gpio_write(pin_out, val);
}

void reset_gpio() {
    write_gpio(1,0);
    write_gpio(2,0);
    write_gpio(3,0);
    write_gpio(0,0);
}
```

```
In [24]: def buzz(tone, num_beeps):
    for i in range(num_beeps):
        write_gpio(0,1)
        time.sleep(1/(2*tone))
        write_gpio(0,0)
        time.sleep(1/(2*tone))
```

```
In [ ]:
```

In [25]:

```
def server_side():
    #original_sigint = signal.getsignal(signal.SIGINT)
    #signal.signal(signal.SIGINT, exit)
    sock_l = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock_l.bind(('192.168.2.99', 1237))
    sock_l.listen()
    print('Waiting for connection')
    conn, addr = sock_l.accept()
    print('Connected')
    with conn:
        while True:
            data = conn.recv(1024)
            #print(data.decode())
            #print(type(data.decode()))

            if data.decode() == '1':
                print(data)
                buzz(100,50)

            elif data.decode() == '2':
                print(data)
                buzz(300,150)

            elif data.decode() == '4':
                print(data)
                buzz(500,250)

            elif data.decode() == '8':
                print(data)
                print("Client left!")
                break
```

In []:

In [26]:

```
def client():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # TODO:
    print("Client side running")
    # 1: Connect the socket (sock) to the <SERVER-IP> and choosen port <LISTENING
    sock.connect(('192.168.2.99', 1237))
    # 2: Send the message "Hello world!\n"
    while True:
        sock.send(bytes(str(btns.read()), 'utf-8'))
        if btns.read() == 8:
            print("Closing socket from client side")
            break
    # 3: Close the socket
    sock.close()
```

In [27]:

```
from multiprocessing import Process

if __name__ == '__main__':
    p = Process(target=server_side)
    f = Process(target=client)
    p.start()
    f.start()
    f.join()
    p.join()
```

Waiting for connection

Client side running

Connected

b'1'

b'2'

b'4'

b'2'

b'1'

b'2'

Closing socket from client side b'8'

Client left!

In []: