UNIVERSITY OF CALIFORNIA SAN DIEGO

Course # WES 237A

Course Title: Intro to Embed Sys Des

Assignment #4.1

Professor Nadir Weibelt
TA Chen Chen

Author	Student ID
Abdullah Ajlan	A69028719

https://github.com/ajlan-UCSD/Assignment4.1

https://youtu.be/IkFrkdF0Zwk

Assignment 4: Alarm System

The learning objectives for this assignment are

- Run a server and client on the same machine on separate processes
- Connect to and disconnect from a remote server
- Communicate button presses between PYNQ boards.

Part A4.1

In this part, each board will be using the **Buzzer** sensor module.

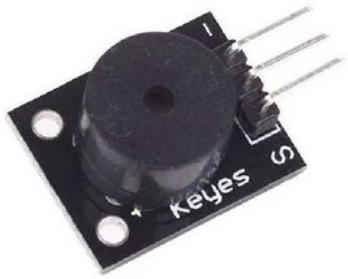

Figure 1: Buzzer module

The buzzer module works as follows

- The '-' pin connects to gnd.
- The '+' pin is the signal you want to write.
- The middle pin is the power (3.3V)
- Writing a square wave (1, 0, 1, 0, 1, 0, etc) alternating high/low, will generate a tone at the given frequency. Psuedocode looks like this
  - while we want a tone
    * write gpio value high
    * sleep for 1/(2 * tone_freq)
    * write gpio_value low
    * sleep for 1/(2 * tone_freq)

The communication part requires the following

- Using multiprocessing library, create two processes: one process for server and one process for client.
- Each board will need to have the following happening.
  - The server process should always be running in listening mode.
  - By pushing one of the buttons on the PYNQ board, the client has to start and connect to the server board.
  - After client connects, pressing a different button should emit a tone on the other PYNQ board.
    * Pushing the button should emit a ~0.5 second tone each time it's pressed.
  - By pushing a third button, the client board disconnects from the server. This will end the communication and both the server and client will terminate

- These steps should be completed for each version of PYNQ1 -> PYNQ2. This means pushing a button on PYNQ1 will emit a tone on PYNQ2 as well as pushing a button on PYNQ2 will emit a tone on PYNQ1.
  - Both directions of communication should be able to operate at the same time.

We used PYZQ Z2 as client , and run server in windows, the next code reveal the server code:

Deliverables

server labtop.py - C:\Users\abdullah.ajlan\Downloads\server labtop.py (3.12.2)

File Edit Format Run Options Window Help

```
import socket

# Define server IP address and port
SERVER_IP = '0.0.0.0'  # Listen on all available interfaces
SERVER_PORT = 12345

# Function for the server process
def server_process():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('0.0.0.0', 12345))
    server_socket.listen(1)
    print("Server is listening on", SERVER_IP, "port", SERVER_PORT)
    conn, addr = server_socket.accept()
    print("Connected to client:", addr)
    while True:
        data = conn.recv(1024)
        if data:
            print("Received message:", data.decode())
    conn.close()

if __name__ == '__main__':
    server_process()
```

As you can see we assign '0.0.0.0' ip server which means that server listen to all available ports, to avoid the case the user define the port my blocked or not wok. The next code shows the client process running in ZYNQ z2 board.

In [5]:

```
import socket
import time
from pynq import Overlay
from pynq.lib import Pmod_IO
from pynq.overlays.base import BaseOverlay

# Initialize the base overlay
base = BaseOverlay("base.bit")

pmod_pin3 = Pmod_IO(base.PMODA, 3, 'out')

# Correctly initialize buttons
button0 = base.buttons[0]
button1 = base.buttons[1]

# Define server IP address and port
SERVER_IP = '192.168.2.1'
SERVER_PORT = 12345

# Function to generate a simple tone
def generate_tone(duration=0.5, frequency=440):
    period = 1.0 / frequency
    cycles = int(duration * frequency)
    for _ in range(cycles):
        pmod_pin3.write(1)
        time.sleep(period / 2)
        pmod_pin3.write(0)
        time.sleep(period / 2)

# Function for the client process
def client_process():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((SERVER_IP, SERVER_PORT))
    print("Connected to server.")
    while True:
        if button0.read():
            generate_tone()  # Generate a tone for 0.5 seconds at 440 Hz
            print("Sending message from client")
            client_socket.sendall(b'Hello world\n')
            time.sleep(1)  # Short delay before next action
        if button1.read():  # Corrected to use button1
            print("Disconnected from server.")
            client_socket.close()
            break

if __name__ == '__main__':
    client_process()
```

```
Connected to server.
Sending message from client
Sending message from client
Sending message from client
Sending message from client
Sending message from client
Sending message from client
Sending message from client
Sending message from client
Sending message from client
Sending message from client
```

```
Sending message from client
Disconnected from server.
```

In [ ]:

The client process work in main, and define to event , when button0 zero pressed that run connection to server ,send hello world and run generate_tone function to run the buzzer that connecting to PMODA 3 for 0.5 second as algorithm defined in puesdo code :

Writing a square wave (1, 0, 1, 0, 1, 0, etc) alternating high/low, will generate a tone at the given frequency. Psuedocode looks like this – while we want a tone

* write gpio value high
* sleep for 1/(2 * tone_freq)
* write gpio_value low
* sleep for 1/(2 * tone_freq)

The out put of Jupiter note book shows that the client send Hello word and run buzzer when buttun0 pressed , and disconnected from server when button 1 pressed. However, the output of server shows the server connected in socket 48120 , and receiver sending message from client then disconnection when button1 pressed .