Assignment 4 Write Up

When it came to designing this assignment, my lab partner (Faisal) and I started out by referring to the previous lab assignments with regards to using I/O and multiprocessing. We split up different aspects of this assignment; Faisal focused on writing code to activate the buzzer as well as encoding of messages to activate the buzzer. I focused on splitting the server and client processes to handle intercommunication between the PYNQ boards.

For the main implementation of the assignment, I started off by copying over the socket_example notebook from Lab 4, since it already has the server/client code already implemented. I've also referred to the multiprocessing notebook from Lab 2 with regards to running the server and client processes simultaneously. Faisal referred to Lab/Assignment 1 to write the code for activating the buzzer with one of the onboard buttons. A big problem that we encountered was because our laptops were interfacing with our respective PYNQs wirelessly over the local router, our laptops were not connected to the internet. As a result, we could not easily share code with each other to ensure consistent functionality, so we ended up mostly writing the code independently from each other. This caused a few hiccups with development; for instance my PYNQ would see messages from Faisal's PYNQ, but his PYNQ could not see mine due to slight differences in our server code. Another issue is that the buzzer in my own IoT kit did not seem to work as well as his, so my buzzer ended up being much quieter. We ended up having to arrange a quiet space for the demo. As for testing, we made sure to write debug print lines to make sure the correct if-else branches were being taken.

The Github repository can be found here.

In [1]:
```
from pynq.overlays.base import BaseOverlay
import multiprocessing
import time
import socket
import os
base = BaseOverlay("base.bit")
base = BaseOverlay("base.bit")
btns = base.btns_gpio
leds = base.leds
```

In [2]:
```
%%microblaze base.PMODB

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

//Function to turn off all GPIO pins on a PMOD
void reset_gpio(){
    for (unsigned int i = 0; i < 8; i++)
    {
        gpio pin_out = gpio_open(i);
        gpio_set_direction(pin_out, GPIO_OUT);
        gpio_write(pin_out, 0);
    }
}

//Function to turn on/off a selected pin of PMODB
unsigned int 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);
    return 0;
}

//Function to read the value of a selected pin of PMODB
unsigned int read_gpio(unsigned int pin){
    gpio pin_in = gpio_open(pin);
    gpio_set_direction(pin_in, GPIO_IN);
    return gpio_read(pin_in);
}
```

In [3]:
```
reset_gpio()
read_gpio(1)
```

Out[3]:
```
0
```

In [4]:
```
freq = 2000
slp = 1/(2*freq)
count = 250
def beep(ct, s):
    i = 0
    while i < ct:
        write_gpio(3,1)
        time.sleep(slp)
        write_gpio(3,0)
        time.sleep(slp)
        i = i + 1
```

In [6]:
```
def serverProc():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.bind(('192.168.8.174',12345))
    sock.listen()
    print("Waiting for connection...")
    (clientSock,addr) = sock.accept()
    print("Connection received!")
    flag = True
    while (flag):
        rec = clientSock.recv(1024)
        dec = rec.decode()
        if dec == '1':
            print("Beep!")
            beep(count,slp)
        elif dec == '2':
            print("Terminating connection...")
            flag = False
        #else:
        #    print("wtf is that")
    sock.close()
    print("Server closed. See ya!")

def clientProc():
    print("Press 1 to Connect to Faisal's Server!")
    while (btns[0].read() == 0):
        time.sleep(0.01)
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print("Attempting to Connect to Server...")
    sock.connect(('192.168.8.212',12345))
    print("Connected! Press 2 to buzz Faisal's PYNQ. Press 3 to end.")
    while (btns[2].read() == 0):
        while (btns[1].read() == 0 and btns[2].read() == 0):
            time.sleep(0.01)
        if (btns[1].read() == 1 and btns[2].read() == 0):
            msg = '1'
            b_en = msg.encode(encoding= "utf-8")
            sock.send(b_en)
            print("Signal sent! Did they get it?")
            print("Cooldown...")
            time.sleep(1)
            print("1 sec cooldown done. Try another input.")
    print("Client closing...")
    end_msg = '2'
    b_end = end_msg.encode(encoding= "utf-8")
    sock.send(b_end)
    sock.close()
    print("Client closed!")
```

In [11]:
```
procs = [] # a future list of all our processes

# Launch process1
p1_start = time.time()
p1 = multiprocessing.Process(target=serverProc, args=())
p1.start() # start the process
print("Server started!")
procs.append(p1)

# Launch process2
p2_start = time.time()
p2 = multiprocessing.Process(target=clientProc, args=())
p2.start() # start the process
procs.append(p2)
```

```
Server started!
Waiting for connection...
Press 1 to Connect to Faisal's Server!
Connection received!
Attempting to Connect to Server...
Connected! Press 2 to buzz Faisal's PYNQ. Press 3 to end.
Beep!
Beep!
Beep!
Beep!
Beep!
Beep!
Signal sent! Did they get it?
Cooldown...
1 sec cooldown done. Try another input.
Signal sent! Did they get it?
Cooldown...
1 sec cooldown done. Try another input.
Signal sent! Did they get it?
Cooldown...
1 sec cooldown done. Try another input.
Signal sent! Did they get it?
Cooldown...
1 sec cooldown done. Try another input.
Client closing...
Client closed!
Terminating connection...
Server closed. See ya!
```