Embedded System Lab 3

Sensor node with wifi and socket client/server

b07901095 吳宜庭

Github project link: https://github.com/yitingwu31/eslab2021/tree/master/lab3

I. Introduction

This lab focuses on the application of built-in sensors of STM32 and the Wi-Fi and socket data transmission. There are many kinds of sensors in STM32, including 3D accelerometer, 3D gyroscope, magnetometer, and so on. In my application, I chose the 3D accelerometer as my main sensor. As for data transmission, the STM32 board acts as the client whereas the python code on PC as the server. A socket connection was created via Wi-Fi between the board and the PC to successfully send and receive data in both directions.

II. Implementation

1. Wi-Fi Socket

a. Client

On the STM32 board, we first connect the board to a specified wifi address by

```
printf("\nConnecting to %s...\n", MBED_CONF_APP_WIFI_SSID);
int ret = wifi.connect(MBED_CONF_APP_WIFI_SSID, MBED_CONF_APP_WIFI_PASSWORD, NSAPI_SECURITY_WPA_WPA2);
if (ret != 0) {
    printf("\nConnection error\n");
    return -1;
}
```

Once the board is successfully connected to wifi, a TCP socket is created. We set the IP address and the port to the specified number that matches the socket opened on the PC server.

```
const char* HOST = "192.168.50.101";
const int PORT = 65432;
```

```
TCPSocket socket;
nsapi_error_t response;
// Open a socket on the network interface, and create a TCP connection to www.arm.com
SocketAddress a;
net->get_ip_address(&a);
printf("IP address: %s\n", a.get_ip_address() ? a.get_ip_address() : "None");
printf("Sending HTTP request to PC server...\n");
// Open a socket on the network interface, and create a TCP connection to //www.arm.com
socket.open(net);
a.set_ip_address(HOST);
a.set_port(PORT);
response = socket.connect(a);
printf("Socket Address is %s\n", a.get_ip_address());
```

After the TCP socket connection is set up, we can start sending data. In addition to sending data, a buffer is created to catch the incoming response, which is an echo of the data sent from the server. This checks whether the entire data content is sent and received successfully.

```
            response = socket.send(xyz+response, size);
            if (response < 0) {
                printf("Error sending data: %d\n", response);
                socket.close();
                return;
            } else {
                size -= response;
                // Check if entire message was sent or not
                printf("sent %d [%.*s]\n", response, strstr(xyz, "\r\n")-xyz, xyz);
            }
```

```
char rbuffer[64];
response = socket.recv(rbuffer, sizeof rbuffer);
if (response < 0) {
    printf("Error receiving data: %d\n", response);
} else {
    printf("recv %d [%.*s]\n", response, strstr(rbuffer, "\r\n")-rbuffer, rbuffer);
}
```

b. Server

Server is implemented on the PC with python codes (named echo_server.py). It first opens a socket and bind the socket to the specified host and port. It is important to set the socket on listen() mode so it will hang listening to incoming data.

```
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
```

A while True loop keeps the socket on accepting data. The server will also send a copy of the received data back to the client for double-checking.

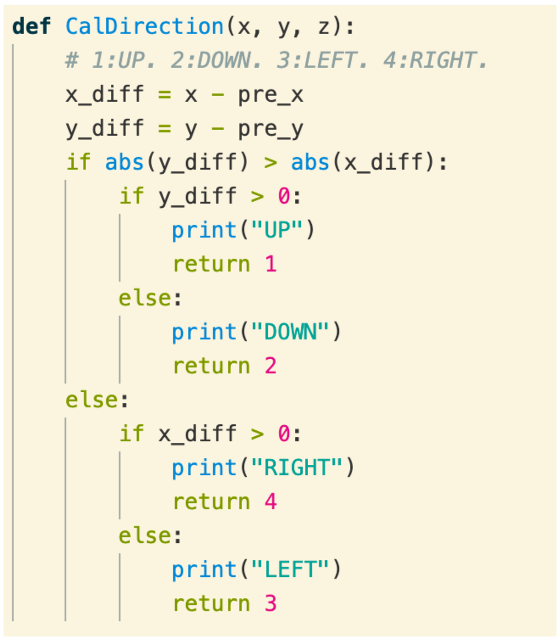
```
while True:
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        gameLoop()
data = conn.recv(1024)
if not data:
    print('\n')
    break
conn.sendall(data)
x, y, z = GetCoor(data)
```

Since the data received is 'bytes', we have to convert it to 'string' and chop out the information we need, namely, the xyz coordinates. These values will be used later in the snake game.

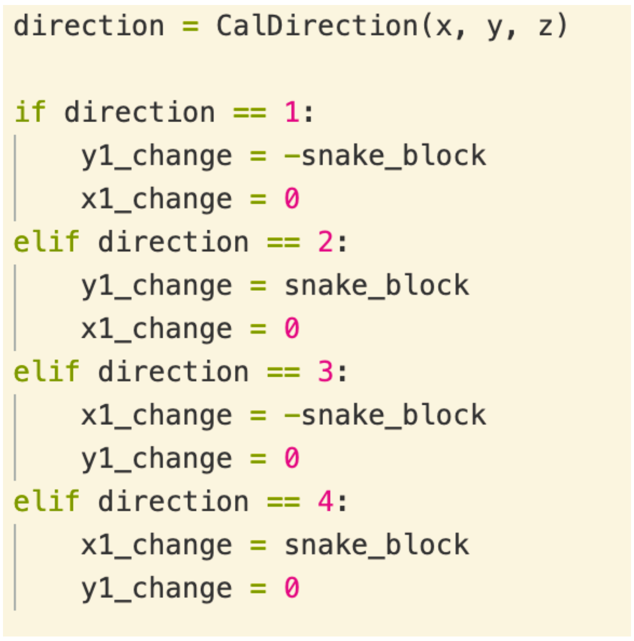
```
def GetCoor(ins):
    print("get s: ", ins)
    words = ins.decode("utf-8")
    idx1 = words.find(':', 0)
    idx2 = words.find(',', idx1)
    xx = int(words[idx1+1 : idx2])
    idx1 = words.find(':', idx2+1)
    idx2 = words.find(',', idx1)
    yy = int(words[idx1+1 : idx2])
    idx1 = words.find(':', idx2+1)
    idx2 = words.find('}', idx1)
    zz = int(words[idx1+1 : idx2])
    return xx, yy, zz
```

2. Pygame

The snake game is implemented with pygame. Most of the codes are referenced from Edureka (link is in the last section). I changed the keydown events to the xyz coordinates received through socket from the STM32 board. I simply calculated the change in the x and y axis and chose the one with the most change to be the snake's moving direction.



```
def CalDirection(x, y, z):
    # 1:UP. 2:DOWN. 3:LEFT. 4:RIGHT.
    x_diff = x - pre_x
    y_diff = y - pre_y
    if abs(y_diff) > abs(x_diff):
        if y_diff > 0:
            print("UP")
            return 1
        else:
            print("DOWN")
            return 2
    else:
        if x_diff > 0:
            print("RIGHT")
            return 4
        else:
            print("LEFT")
            return 3
```



```
direction = CalDirection(x, y, z)

if direction == 1:
    y1_change = -snake_block
    x1_change = 0
elif direction == 2:
    y1_change = snake_block
    x1_change = 0
elif direction == 3:
    x1_change = -snake_block
    y1_change = 0
elif direction == 4:
    x1_change = snake_block
    y1_change = 0
```

III. Results

This application is a classic snake game controlled by tilting the STM32 board.
See demo video here: https://www.youtube.com/watch?v=0sybx2__hVo

IV. Discussion

This is my first time playing with Wi-Fi and socket connection so I spent a lot of time studying the concepts behind socket connections at first. Once I successfully differentiated Wi-Fi address from socket host address and set up the first socket connection correctly (with lots of help from my teammates), the rest wasn't that difficult. All that was left was to decode the incoming bytes into useful data.

One problem regarding the Mbed compiler was that since the only sensor header file included in main.cpp was "stm32l475e_iot-1_accelero.h", I tried removing other header files (eg. gyro.h). However, errors arose once the seemingly unreferenced files were removed. I haven't found out why but my guesses are that these header files are secretly included somewhere other than main.cpp.

V. References

1. Python Snake Game: https://www.edureka.co/blog/snake-game-with-pygame/
2. Mbed TCP Socket: https://os.mbed.com/docs/mbed-os/v6.8/apis/tcpsocket.html
3. Python Socket: https://realpython.com/python-sockets/
4. Shout out to my teammate 陳映樵 for answering tons of my silly questions!