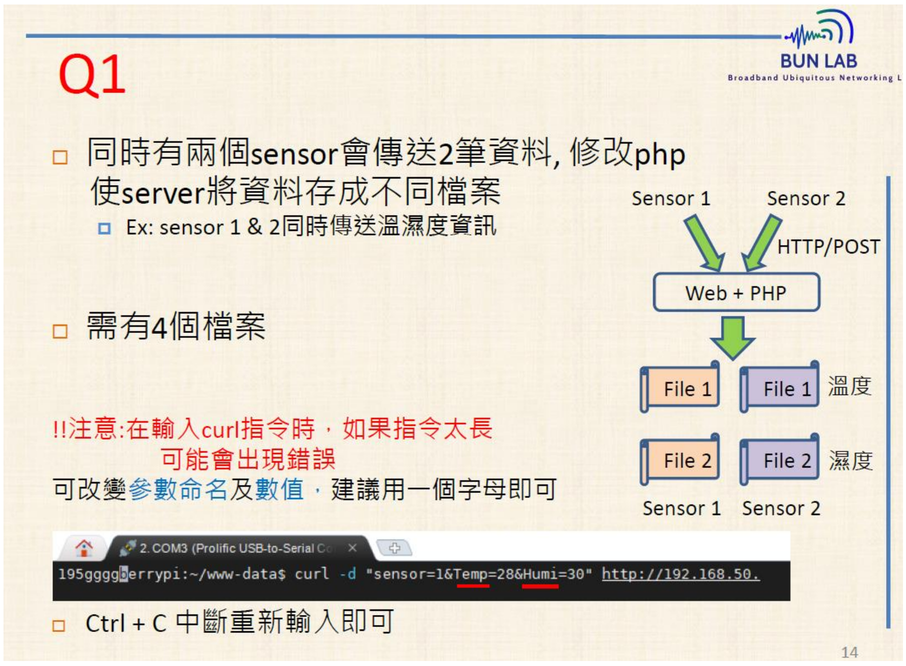
Communication Networks Lab
Topic IOT-Lab2

Q1:



Q1

BUN LAB
Broadband Ubiquitous Networking L

- 同時有兩個sensor會傳送2筆資料, 修改php使server將資料存成不同檔案
  - Ex: sensor 1 & 2同時傳送溫濕度資訊
- 需有4個檔案

!!注意:在輸入curl指令時，如果指令太長
可能會出現錯誤
可改變參數命名及數值，建議用一個字母即可

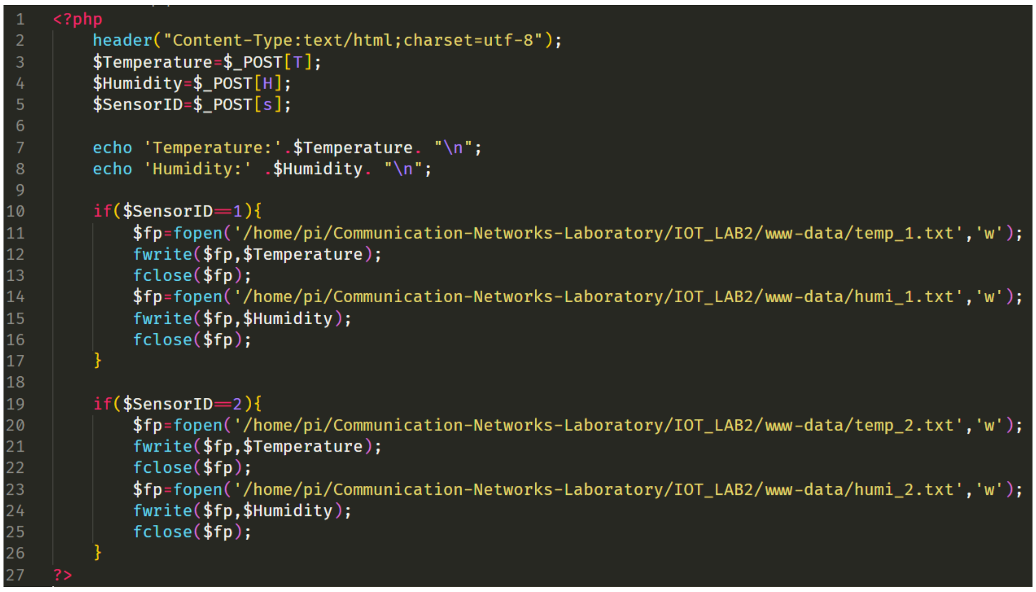
```
195ggggberrypi:~/www-data$ curl -d "sensor=1&Temp=28&Humi=30" http://192.168.50.
```

- Ctrl + C 中斷重新輸入即可

Sensor 1
Sensor 2
HTTP/POST
Web + PHP
File 1
File 1
溫度
File 2
File 2
濕度
Sensor 1
Sensor 2
2. COM3 (Prolific USB-to-Serial Co

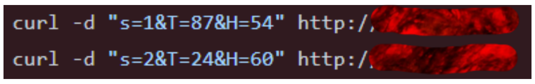
14

Code description:



```php
<?php
    header("Content-Type:text/html;charset=utf-8");
    $Temperature=$_POST[T];
    $Humidity=$_POST[H];
    $SensorID=$_POST[s];

    echo 'Temperature:'.$Temperature. "\n";
    echo 'Humidity:' .$Humidity. "\n";

    if($SensorID==1){
        $fp=fopen('/home/pi/Communication-Networks-Laboratory/IOT_LAB2/www-data/temp_1.txt','w');
        fwrite($fp,$Temperature);
        fclose($fp);
        $fp=fopen('/home/pi/Communication-Networks-Laboratory/IOT_LAB2/www-data/humi_1.txt','w');
        fwrite($fp,$Humidity);
        fclose($fp);
    }

    if($SensorID==2){
        $fp=fopen('/home/pi/Communication-Networks-Laboratory/IOT_LAB2/www-data/temp_2.txt','w');
        fwrite($fp,$Temperature);
        fclose($fp);
        $fp=fopen('/home/pi/Communication-Networks-Laboratory/IOT_LAB2/www-data/humi_2.txt','w');
        fwrite($fp,$Humidity);
        fclose($fp);
    }
?>
```

Execute by:

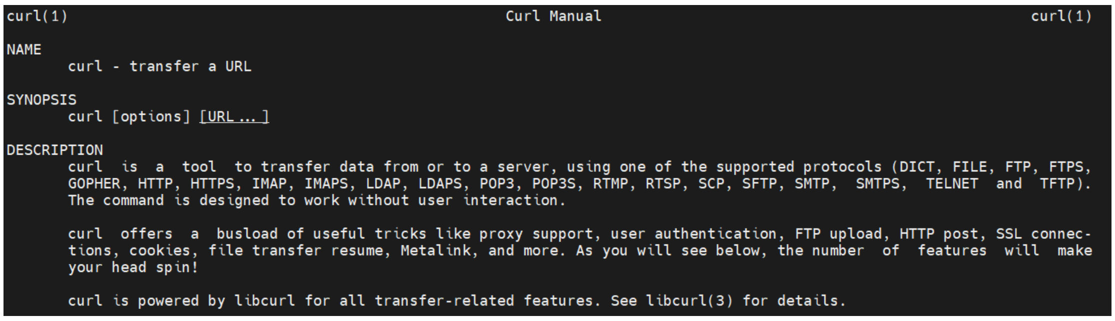


```
curl -d "s=1&T=87&H=54" http:/[redacted]
curl -d "s=2&T=24&H=60" http:/[redacted]
```

First: curl

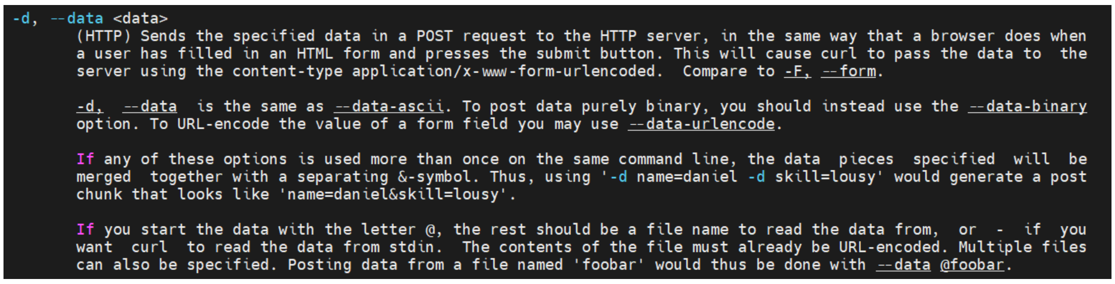
To better know the principle of the code, let's first dive into the man-page of curl, and try to figure what does curl means and what's is the option flag -d means.

$ man curl



```
curl(1)                                   Curl Manual                                   curl(1)

NAME
       curl - transfer a URL

SYNOPSIS
       curl [options] [URL ...]

DESCRIPTION
       curl  is  a  tool  to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP, FTPS,
       GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP,  SMTPS,  TELNET  and  TFTP).
       The command is designed to work without user interaction.

       curl  offers  a  busload of useful tricks like proxy support, user authentication, FTP upload, HTTP post, SSL connec-
       tions, cookies, file transfer resume, Metalink, and more. As you will see below, the number  of  features  will  make
       your head spin!

       curl is powered by libcurl for all transfer-related features. See libcurl(3) for details.
```

-d option description:



```
 -d, --data <data>
        (HTTP) Sends the specified data in a POST request to the HTTP server, in the same way that a browser does when
        a user has filled in an HTML form and presses the submit button. This will cause curl to pass the data to  the
        server using the content-type application/x-www-form-urlencoded.  Compare to -F, --form.

        -d,  --data  is the same as --data-ascii. To post data purely binary, you should instead use the --data-binary
        option. To URL-encode the value of a form field you may use --data-urlencode.

        If any of these options is used more than once on the same command line, the data  pieces  specified  will  be
        merged  together with a separating &-symbol. Thus, using '-d name=daniel -d skill=lousy' would generate a post
        chunk that looks like 'name=daniel&skill=lousy'.

        If you start the data with the letter @, the rest should be a file name to read the data from,  or  -  if  you
        want  curl  to read the data from stdin.  The contents of the file must already be URL-encoded. Multiple files
        can also be specified. Posting data from a file named 'foobar' would thus be done with --data @foobar.
```

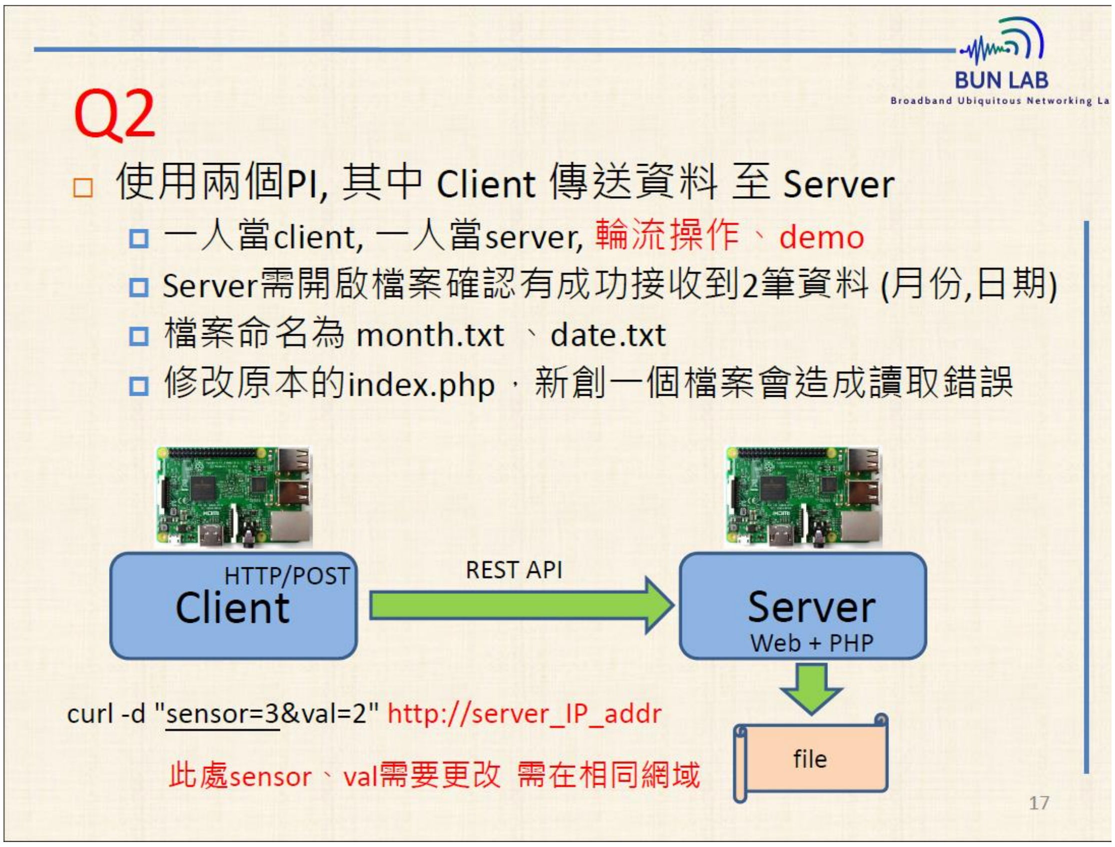
Through the description, we can obviously see that curl can use http protocol to upload or download the file or data. With option -d we specify that we are going to transfer data in "Post" request.

Second: index.php

Simply store the data transfer by the curl into the variable, and using C-like syntax to create file and write the data into it. Nothing fancy here.

Result: create the file with data transfer by curl into www-data folder.

Q2:



BUN LAB
Broadband Ubiquitous Networking La

Q2

- 使用兩個PI, 其中 Client 傳送資料 至 Server
  - 一人當client, 一人當server, 輪流操作、demo
  - Server需開啟檔案確認有成功接收到2筆資料 (月份,日期)
  - 檔案命名為 month.txt 、date.txt
  - 修改原本的index.php，新創一個檔案會造成讀取錯誤

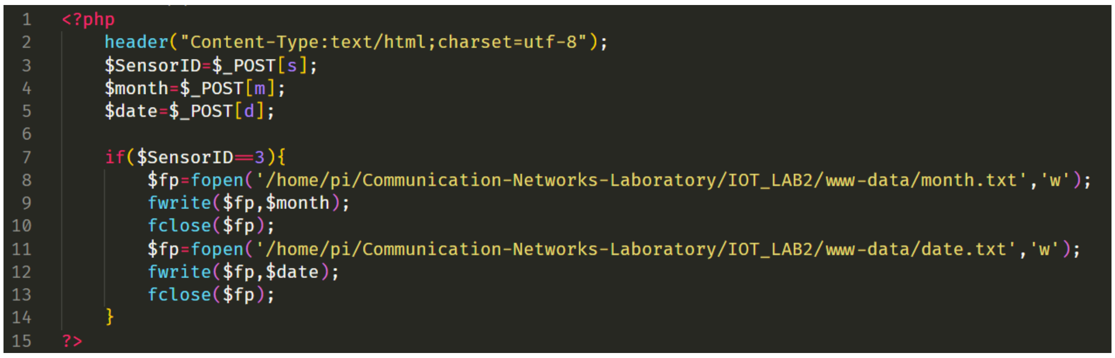
HTTP/POST
Client
REST API
Server
Web + PHP
file

curl -d "sensor=3&val=2" http://server_IP_addr

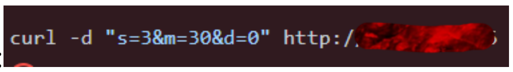
此處sensor、val需要更改 需在相同網域

17

Code description:



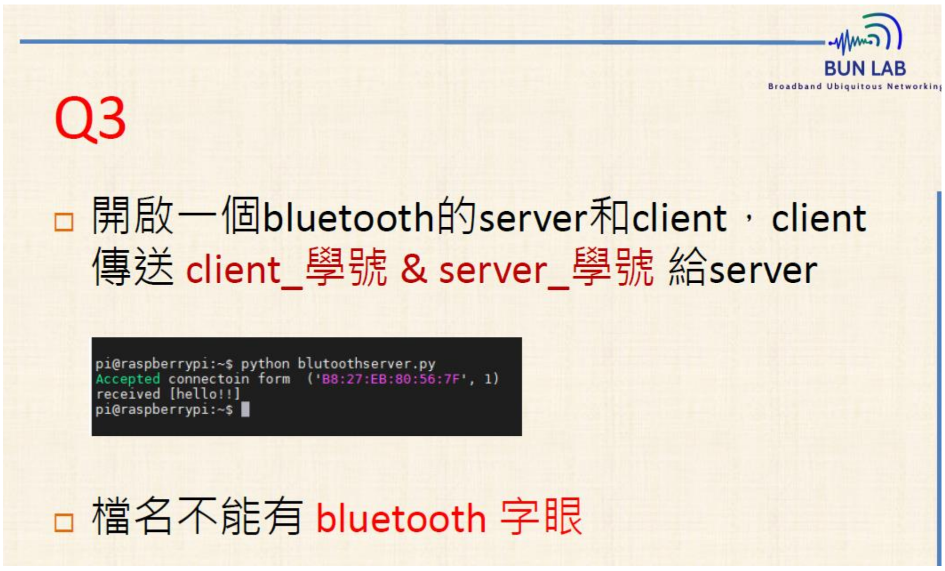
```php
<?php
    header("Content-Type:text/html;charset=utf-8");
    $SensorID=$_POST[s];
    $month=$_POST[m];
    $date=$_POST[d];

    if($SensorID==3){
        $fp=fopen('/home/pi/Communication-Networks-Laboratory/IOT_LAB2/www-data/month.txt','w');
        fwrite($fp,$month);
        fclose($fp);
        $fp=fopen('/home/pi/Communication-Networks-Laboratory/IOT_LAB2/www-data/date.txt','w');
        fwrite($fp,$date);
        fclose($fp);
    }
?>
```

Executed by my friend: 

```
curl -d "s=3&m=30&d=0" http:/[redacted]
```

Result: create the file with data transfer by my friend with curl into my www-data folder.

Q3:



Q3

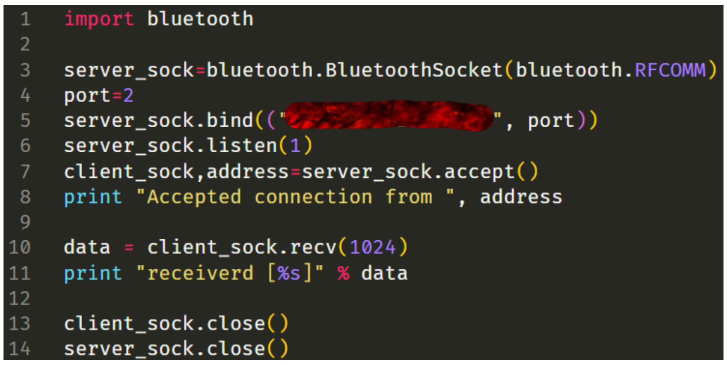
- 開啟一個bluetooth的server和client，client 傳送 client_學號 & server_學號 給server

```
pi@raspberrypi:~$ python blutoothserver.py
Accepted connectoin form  ('B8:27:EB:80:56:7F', 1)
received [hello!!]
pi@raspberrypi:~$
```

- 檔名不能有 bluetooth 字眼

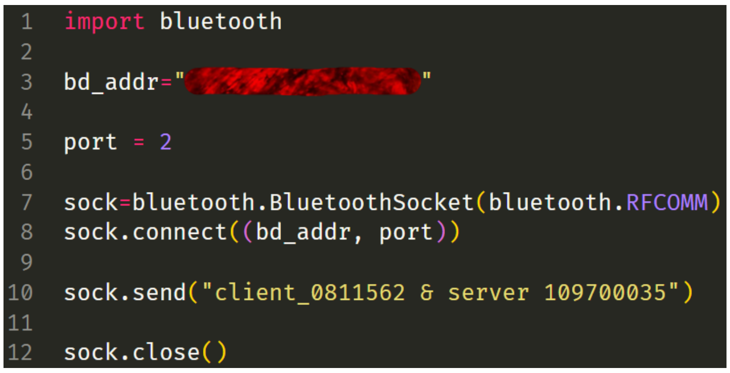
Code description:

server.py:



```python
import bluetooth

server_sock=bluetooth.BluetoothSocket(bluetooth.RFCOMM)
port=2
server_sock.bind(("[redacted]", port))
server_sock.listen(1)
client_sock,address=server_sock.accept()
print "Accepted connection from ", address

data = client_sock.recv(1024)
print "receiverd [%s]" % data

client_sock.close()
server_sock.close()
```

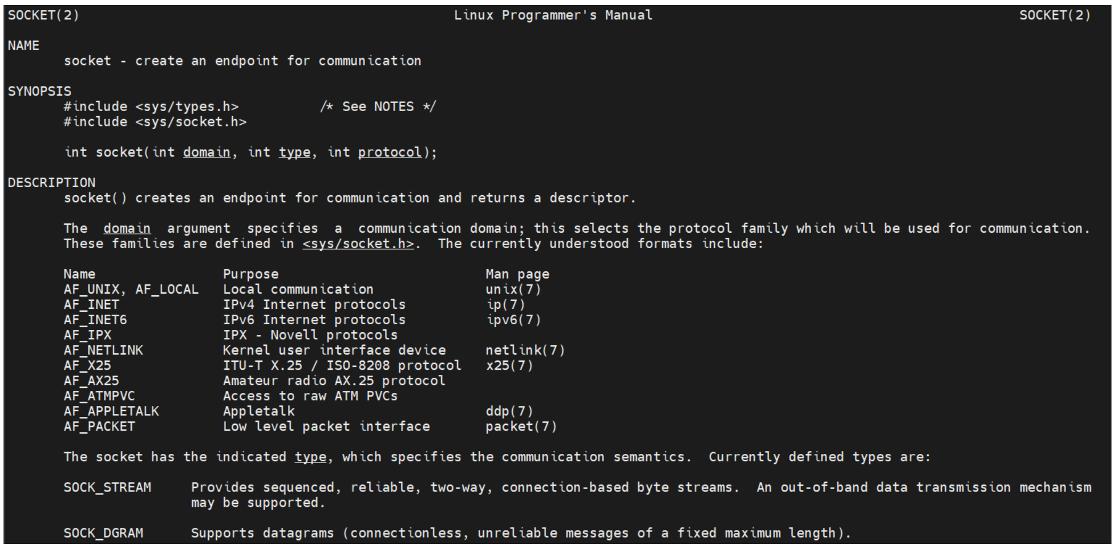
client.py:



```python
import bluetooth

bd_addr="[redacted]"

port = 2

sock=bluetooth.BluetoothSocket(bluetooth.RFCOMM)
sock.connect((bd_addr, port))

sock.send("client_0811562 & server 109700035")

sock.close()
```

First: socket

To better know the principle of the code, let's first dive into the man-page of socket, and try to figure out what does socket means.

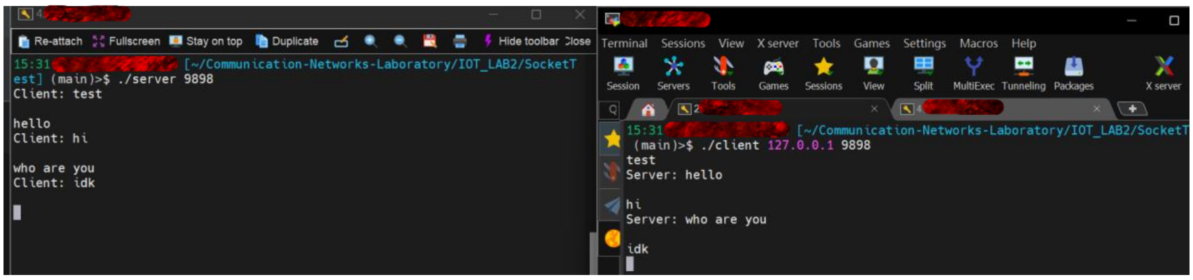


```
SOCKET(2)                                Linux Programmer's Manual                                SOCKET(2)

NAME
       socket - create an endpoint for communication

SYNOPSIS
       #include <sys/types.h>          /* See NOTES */
       #include <sys/socket.h>

       int socket(int domain, int type, int protocol);

DESCRIPTION
       socket() creates an endpoint for communication and returns a descriptor.

       The  domain  argument  specifies  a  communication domain; this selects the protocol family which will be used for communication.
       These families are defined in <sys/socket.h>.  The currently understood formats include:

       Name                Purpose                          Man page
       AF_UNIX, AF_LOCAL   Local communication              unix(7)
       AF_INET             IPv4 Internet protocols          ip(7)
       AF_INET6            IPv6 Internet protocols          ipv6(7)
       AF_IPX              IPX - Novell protocols
       AF_NETLINK          Kernel user interface device     netlink(7)
       AF_X25              ITU-T X.25 / ISO-8208 protocol   x25(7)
       AF_AX25             Amateur radio AX.25 protocol
       AF_ATMPVC           Access to raw ATM PVCs
       AF_APPLETALK        Appletalk                        ddp(7)
       AF_PACKET           Low level packet interface       packet(7)

       The socket has the indicated type, which specifies the communication semantics.  Currently defined types are:

       SOCK_STREAM     Provides sequenced, reliable, two-way, connection-based byte streams.  An out-of-band data transmission mechanism
                       may be supported.

       SOCK_DGRAM      Supports datagrams (connectionless, unreliable messages of a fixed maximum length).
```

But, I think it's not helpful for beginner to know the what does socket means. Let's try some basic socket programming in C with TCP protocol. Since the code is a little bit longer, I just put the implementation code on my Github repository:

https://github.com/coherent17/Socket-Programming/tree/main/SocketTest

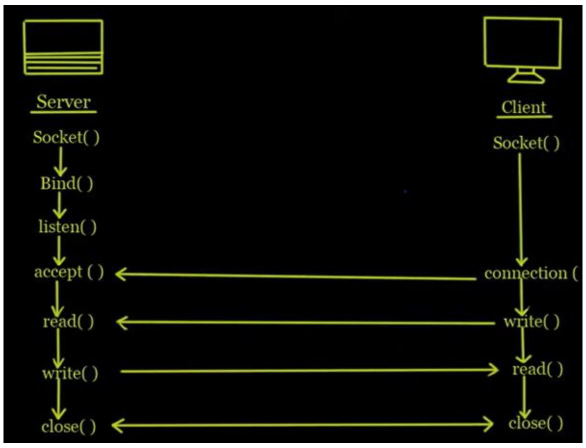
Socket test Result: a simple chat room implementation



```
15:31 [redacted] [~/Communication-Networks-Laboratory/IOT_LAB2/SocketT
est] (main)>$ ./server 9898
Client: test

hello
Client: hi

who are you
Client: idk
```

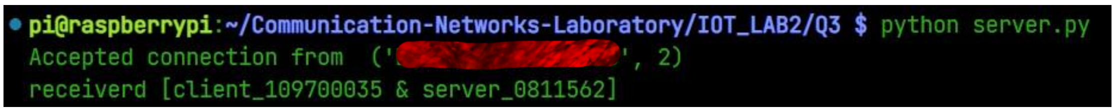
```
15:31 [redacted] [~/Communication-Networks-Laboratory/IOT_LAB2/SocketT
 (main)>$ ./client 127.0.0.1 9898
test
Server: hello

hi
Server: who are you

idk
```

Second: structure of how socket work



Third: transfer data using Bluetooth implement by socket.
With the unique BD address, we can transfer our data (string in this lab) using Bluetooth (UART protocol).

Result:



```
• pi@raspberrypi:~/Communication-Networks-Laboratory/IOT_LAB2/Q3 $ python server.py
  Accepted connection from  ('[redacted]', 2)
  receiverd [client_109700035 & server_0811562]
```

Feedback:

It is quite hard for us to know all of the details about communication protocol in this short time, but I think I try my best to figure out most of the knowledge. It's quite fun in this lab. Thanks for TA's help and my friend. I hope my friend will not be late next time, so that we can finish the lab more quickly.