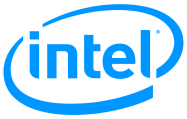
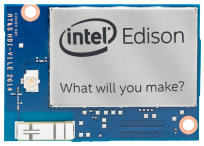




Intel® Edison Tutorial: TCP Socket Communications

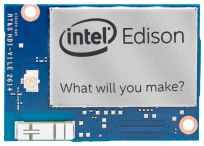
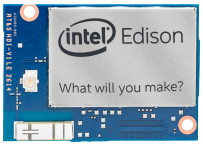


Table of Contents

Revision history		
Version	**Date**	**Comment**
1.0	mm/dd/yy	Initial release



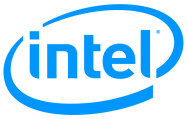
Introduction

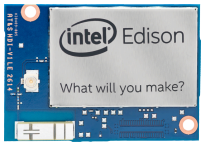
In this tutorial, you will:

1. Learn about TCP connections and how they are different from UDP.
2. Try out code that implements a simple TCP server-client communication link.

List of Required Materials and Equipment

- 2x Intel Edison Kit.
- 4x USB 2.0 A-Male to Micro B Cable (micro USB cable).
- 2x powered USB hub OR an 2x external power supply.
- 1x Grove – Starter Kit for Arduino.
- 1x Personal Computer.
- 1x Wi-Fi network configured to access the internet.





Introduction

For a comprehensive explanation on networking protocols such as TCP and UDP, please refer to the below links:

- http://www.howtogeek.com/190014/htg-explains-what-is-the-difference-between-tcp-and-udp/
- http://www.cs.rutgers.edu/~pxk/rutgers/notes/sockets/

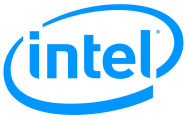
To summarize, UDP stands for **U**ser **D**atagram **P**rotocol, and TCP stands for **T**ransmission **C**ontrol **P**rotocol. The key differences and tradeoffs are as follows:

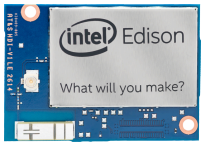
TCP	UDP

• Tracks packets sent over the internet
• Is slower, but more reliable
• Used in most situations
	• Sends the packets and does not care if they are received or not
	• Is faster, but less reliable
	• Used in live-stream type situations such as
	  ○ Live video stream
	  ○ Gaming

There are a few steps to communicate over a network:

1. Create a socket
2. Identify the socket
3. Use the socket
    a. Server: wait for incoming connection
    b. Client: connect to server's socket
4. Send and receive messages
    a. Perhaps do something based on the messages received/sent
5. Close the socket





Simple Client-Server Architecture

For server-client communication, it is best to start with a very simple implementation. To do this, the server will set up a socket, and wait for the client to connect. Once the client has connected, the client code will ask the user for a string to input via standard input.

Once the input is received, it will send the message to the server. The server will then display the string received from the client, and send a response to the client stating "I got your message". After checking that the message was successfully received by the client, the server code will exit. The client will wait for wait for the server's acknowledgement, after which, the client will exit. The code to achieve this can be found in the folder labelled "FILES/non-threaded" provided with this PDF. Follow the below instructions to learn about this system.

1. Download the code from the folder labelled "FILES/**non-**threaded".
2. Upload the source code file labelled "client.c" to your Intel Edison. This Intel Edison will be referred to as **the client** for the rest of this tutorial.

   For more information on how to transfer files between a personal computer and a remote computing device, please refer to the ***SFTP*** section in the document labelled ***Intel Edison Tutorial – Introduction, Shell Access and SFTP***.

3. Upload the source code file labelled **server.c** to the other Intel Edison. This Intel Edison will be referred to as **the server** for the rest of this tutorial.

4. Access the shell on each Intel Edison through SSH.

   Make sure to use SSH as this will verify if your board has internet connectivity throughout the tutorial.

   If you are accessing the shell of each Intel Edison on the same personal computer, make sure to have a separate window for each. Ensure you do not close a window that is accessing the shell of an Intel Edison while a process is running. If a process is running when the shell session is quit, the execution of said process will likely be interrupted and halted. For more information, refer to the document labelled ***Intel Edison Tutorial – Introduction, Shell Access and SFTP***.

5. On the shell of **the server**, issue the following commands:

   $ configure_edison --showWiFiIP
   Record the output from this command. You will need the IP address of **the server** Intel Edison soon.

   $ gcc -o server server.c

```
root@ucla_iot_dev:~/SUSP/TCP# gcc -o server server.c
root@ucla_iot_dev:~/SUSP/TCP#
```

Figure 1: Successful compilation of source the file server.c

6. On the shell of **the client**, issue the following command:

   $ gcc -o client client.c

```
root@ucla_iot:~/SUSP/TCP# gcc -o client client.c
root@ucla_iot:~/SUSP/TCP#
```

Figure 2: Successful compilation of the source code file client.c

7. To start the server, issue the following command at the shell prompt for **the server**

   $ ./server 5000

```
root@ucla_iot_dev:~/SUSP/TCP# ./server 5000
```

Figure 3: Successfully running the server

If you get an error stating "ERROR on binding: Address already in use", try a different port number such as 8000:

$ ./server 8000

```
root@ucla_iot_dev:~/SUSP/TCP# ./server 5000
ERROR on binding: Address already in use
root@ucla_iot_dev:~/SUSP/TCP# ./server 8000
```

Figure 4: Restarting the server with a new port number

The server is functioning correctly if the cursor is on a blank line on your terminal session

```
root@ucla_iot_dev:~/SUSP/TCP# ./server 5000
```

Figure 5: Correct server functionality

8. Access **the client** Intel Edison's shell and issue the following command:

   $ ./client <IP_ADDR> <PORT>

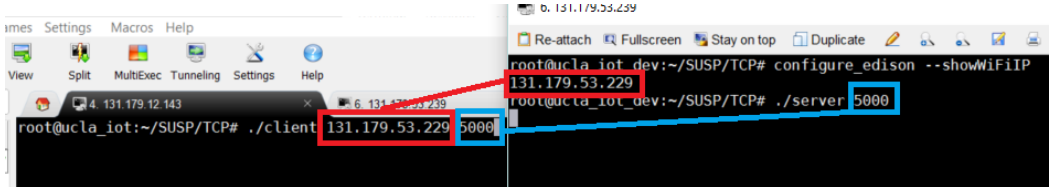


Figure 6: Example of running the client application after running the server application

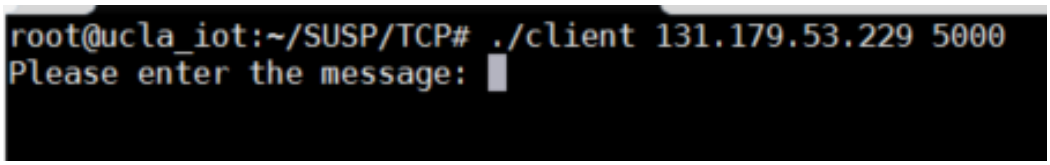


Figure 7: The client Intel Edison after running the client application successfully

Type a message and press enter.

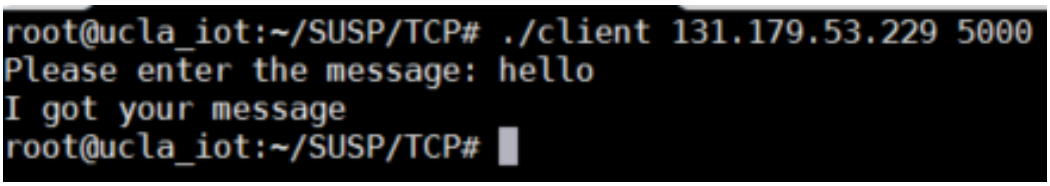


Figure 8: Sending a string from a client Intel Edison to a server Intel Edison using TCP

9. Examine the output on the shell of **the server** Intel Edison.

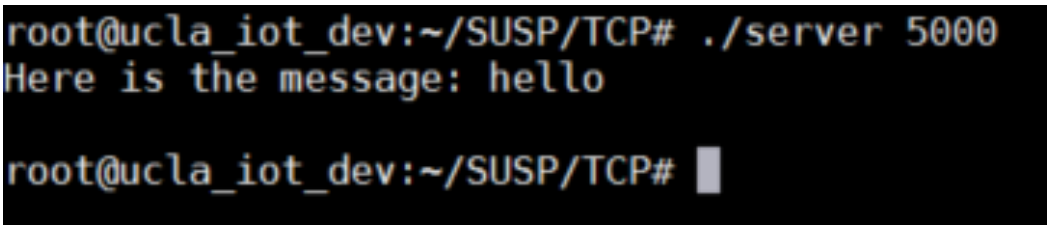
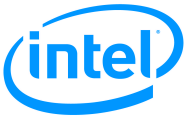
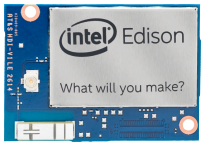


Figure 9: Output of shell on the server Intel Edison on receipt of message passed through TCP

Notice how the server exits as soon as one message has been received.





Tasks

1. Examine the 5 steps required for communication over a TCP socket.

   Write down which lines of code (in server.c and client.c) achieve each these steps.

   HINT: The comments in the code and the following links may help:
   a. http://www.linuxhowtos.org/C_C++/socket.htm
   b. http://www.cs.rutgers.edu/~pxk/rutgers/notes/sockets/

2. Modify the server.c file such that the server sends the following message to the client:

   I got the following message from you: <MESSAGE>

   where **<MESSAGE>** is the message the client sent to the server

```
root@ucla_iot:~/SUSP/TCP# ./client 131.179.53.229 8000
Please enter the message: Hello World!
I got the following message from you: Hello World!

root@ucla_iot:~/SUSP/TCP#
```

   Figure 10: Output of the client's shell after modification to the server code from task 2

3. Using the source code provided and the online resources below, discuss why the server can only handle servicing one client.

   http://www.linuxhowtos.org/C_C++/socket.htm

4. Discuss what modifications you would need to make to the source file server.c in order to service multiple, simultaneous client connections.

5. **EXTENSION**

   Implement the modifications discussed in task 4 and have multiple client Intel Edison's connect simultaneously to one server Intel Edison.