LAB #1: INTRODUCTION TO RASPBERRY PI & SENSORIAN AND SYSTEM SETUP

Design and Analysis of IoT Software

Abstract

The proceedings of this lab report document our group learning the fundamentals of Raspberry Pi, Sensorian hardware, VMware, and Linux systems. Additionally, the process of setting up a development environment.

Youssef & Devante

100715637 & 100554361

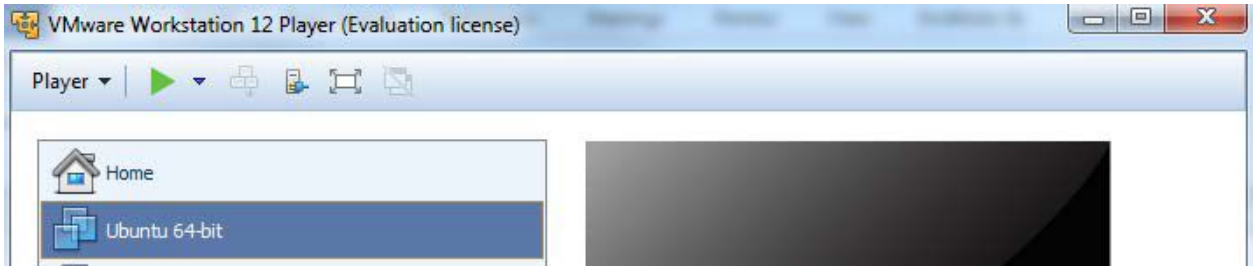
Introduction

The purpose of this lab session was for our group to learn the fundamentals of Raspberry Pi, Sensorian hardware, VMware, and Linux systems. Additionally, our group learned the process of setting up a development environment.

Lab Session Content

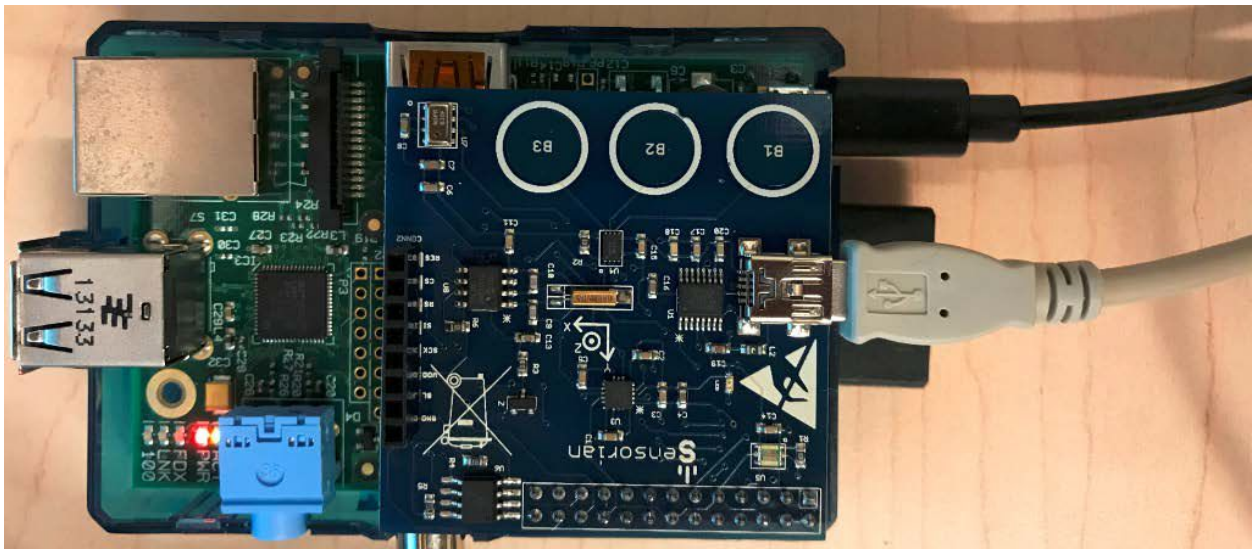
VMware Setup

The process began with setting up VMware on the host computer (running Windows 10). The 64-bit Ubuntu image file was downloaded to run on the virtual machine.



Hardware Setup

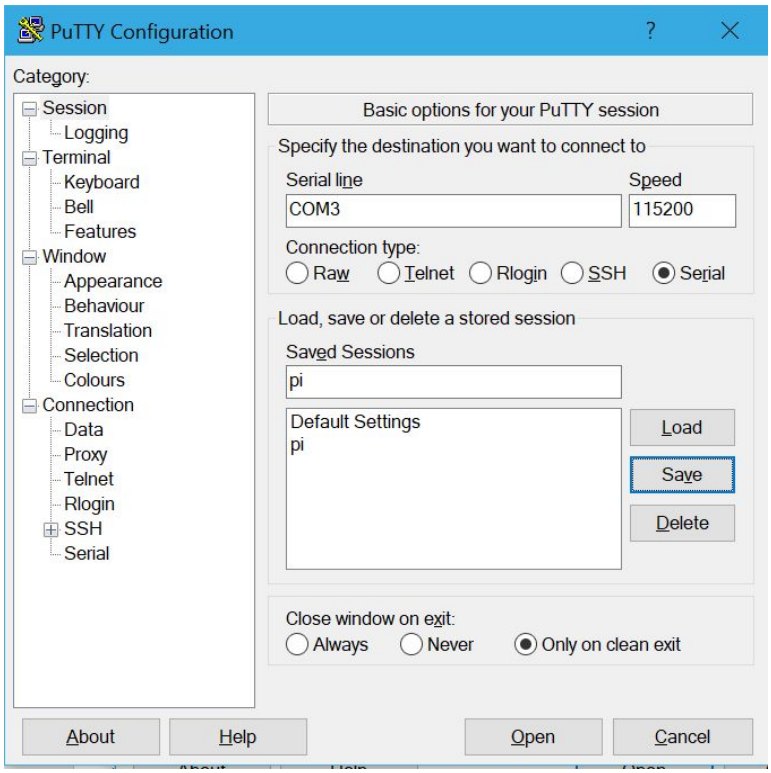
Two pieces of hardware were used during the lab session. Namely, a Raspberry Pi (single board computer), and a Sensorian Shield (sensor packed shield).

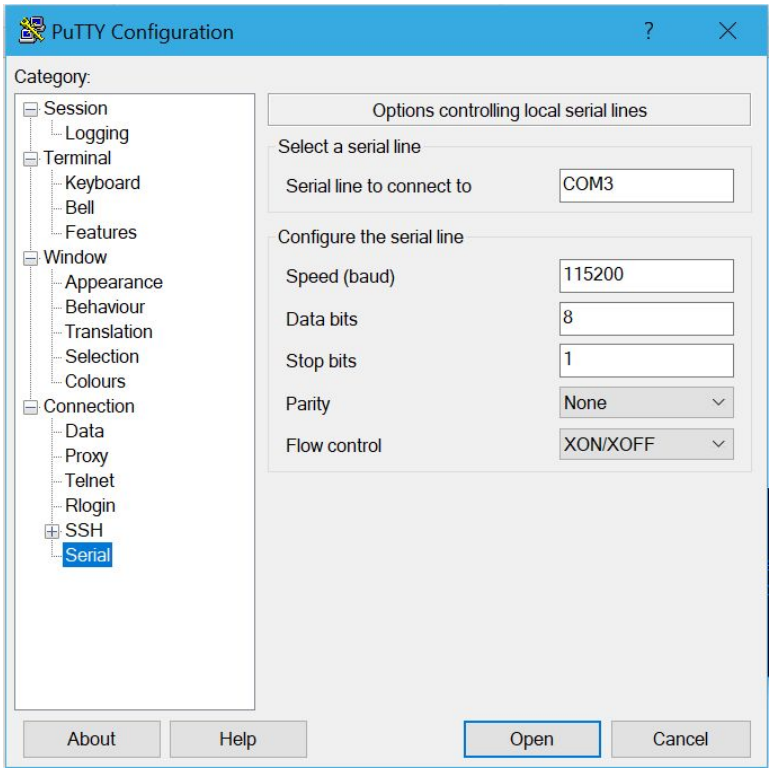


Network Setup

In order to communicate with the Raspberry Pi and Sensorian, a software tool called PuTTY was used as a terminal emulator, serial console and network file transfer application; it has both a Windows and a Linux version.

The images below show the serial connection configuration.





The next step after establishing a serial connection via the PuTTY client was to log into the Raspberry Pi using the following credentials:

Username: pi

Password: raspberry

```
pi@169.254.254.2's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr 10 11:05:58 2017 from desktop-sqd1fpn.local

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $
```

To display device drivers, the command “sudo dmesg” was used.

```
lwu@ubuntu:~$ sudo dmesg
[sudo] password for lwu:
     0.000000] Linux version 4.10.0-28-generic (buildd@lgw01-12) (gcc version 5.
4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4) ) #32~16.04.2-Ubuntu SMP Thu Jul 20
 10:19:48 UTC 2017 (Ubuntu 4.10.0-28.32~16.04.2-generic 4.10.17)
     0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.10.0-28-generic root=UUI
D=961983ec-8eb0-4a29-a856-67fa26b9f810 ro find_preseed=/preseed.cfg auto nopromp
t priority=critical locale=en_US quiet
     0.000000] KERNEL supported cpus:
     0.000000]   Intel GenuineIntel
     0.000000]   AMD AuthenticAMD
     0.000000]   Centaur CentaurHauls
     0.000000] Disabled fast string operations
     0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point regi
sters'
     0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
     0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
     0.000000] x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256
     0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832 bytes,
using 'standard' format.
     0.000000] e820: BIOS-provided physical RAM map:
```

To update the package cache, the command “**sudo apt-get update**” was used.

To install PuTTY in Linux via the terminal, “**sudo apt-get install putty**” was used.

A username needed to be added to the dialout group, thus “**sudo usermod -aG dialout USERNAME**” was used.

To obtain the IP address from the Pi, with the serial Mini B USB connection, the command "**ifconfig eth0**" was used.

```
pi@raspberrypi:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr b8:27:eb:6d:73:fd
          inet addr:169.254.254.2  Bcast:169.254.255.255  Mask:255.255.0.0
          inet6 addr: fe80::9fce:5175:16af:c3ab/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1700 errors:0 dropped:0 overruns:0 frame:0
          TX packets:942 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:264815 (258.6 KiB)  TX bytes:154645 (151.0 KiB)

pi@raspberrypi:~$
```

A route to the Raspberry Pi must be made in the Ubuntu OS. Within the Ubuntu VM, the following command was run: **route**

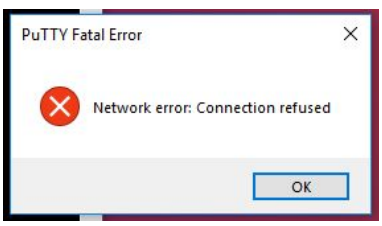
To create the route to the Raspberry Pi: "**sudo route add IP_OF_PI gw GATEWAY_IP**"

Replacing IP_OF_PI with the IP address of the Pi's Ethernet interface and GATEWAY_IP with the IP address obtained in the previous step.

To ping the Raspberry Pi from the virtual machine: **ping IP_OF_PI**

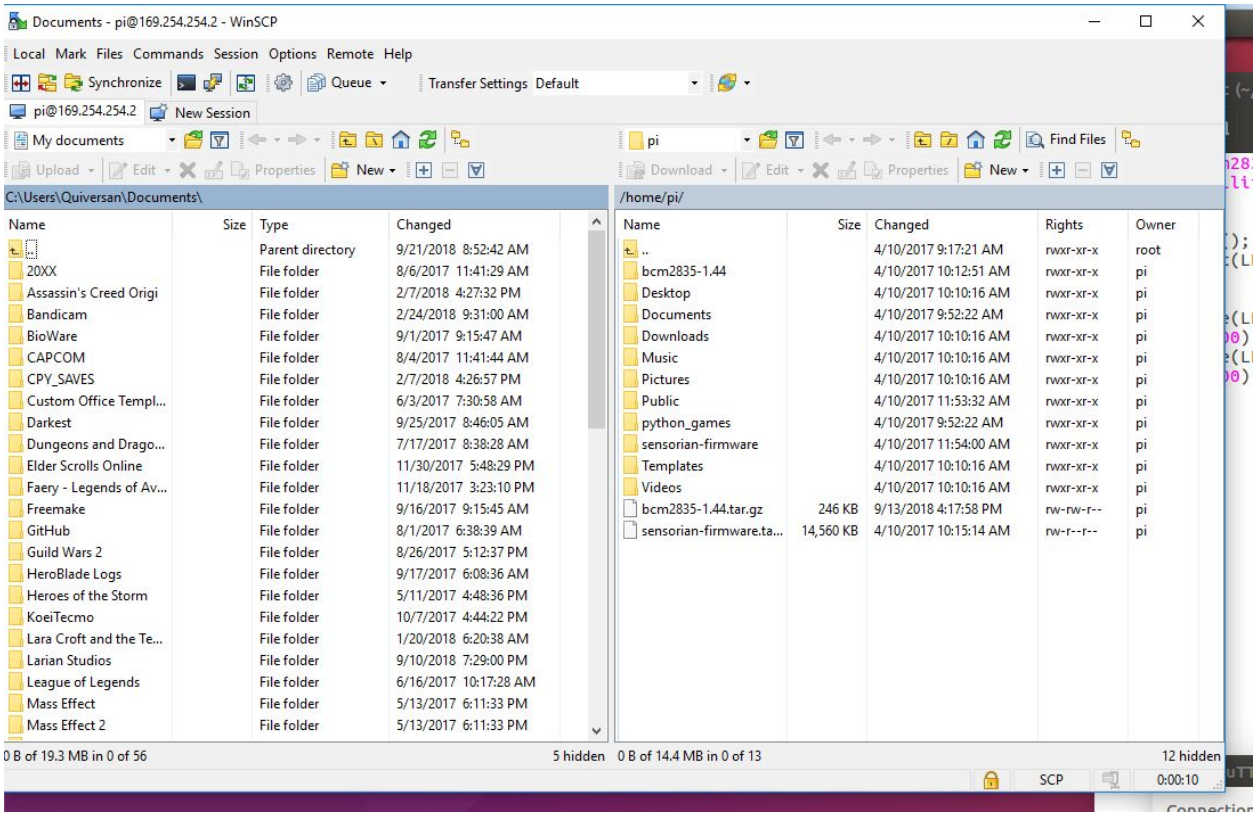
```
youssef@youssef-virtual-machine: ~
64 bytes from 169.254.254.2: icmp_seq=6 ttl=128 time=2.40 ms
64 bytes from 169.254.254.2: icmp_seq=7 ttl=128 time=3.42 ms
64 bytes from 169.254.254.2: icmp_seq=8 ttl=128 time=3.51 ms
64 bytes from 169.254.254.2: icmp_seq=9 ttl=128 time=2.54 ms
64 bytes from 169.254.254.2: icmp_seq=10 ttl=128 time=1.97 ms
64 bytes from 169.254.254.2: icmp_seq=11 ttl=128 time=2.15 ms
64 bytes from 169.254.254.2: icmp_seq=12 ttl=128 time=2.79 ms
64 bytes from 169.254.254.2: icmp_seq=13 ttl=128 time=2.03 ms
64 bytes from 169.254.254.2: icmp_seq=14 ttl=128 time=2.71 ms
64 bytes from 169.254.254.2: icmp_seq=15 ttl=128 time=2.73 ms
64 bytes from 169.254.254.2: icmp_seq=16 ttl=128 time=1.63 ms
64 bytes from 169.254.254.2: icmp_seq=17 ttl=128 time=2.18 ms
64 bytes from 169.254.254.2: icmp_seq=18 ttl=128 time=4.01 ms
64 bytes from 169.254.254.2: icmp_seq=19 ttl=128 time=1.69 ms
64 bytes from 169.254.254.2: icmp_seq=20 ttl=128 time=1.68 ms
64 bytes from 169.254.254.2: icmp_seq=21 ttl=128 time=2.58 ms
64 bytes from 169.254.254.2: icmp_seq=22 ttl=128 time=1.49 ms
64 bytes from 169.254.254.2: icmp_seq=23 ttl=128 time=3.32 ms
64 bytes from 169.254.254.2: icmp_seq=24 ttl=128 time=2.69 ms
^C
--- 169.254.254.2 ping statistics ---
24 packets transmitted, 24 received, 0% packet loss, time 23045ms
rtt min/avg/max/mdev = 1.498/2.545/4.010/0.639 ms
youssef@youssef-virtual-machine:~$
```

The next step was to connect via SSH using the IP address acquired in the previous step, and the default port setting 22. A network error occurred because SSH needed to be enabled first.



SSH was enabled using the command “**sudo raspi-config**”.

Winscp was downloaded to facilitate a secure file transfer between the host computer and the Raspberry Pi. The driver files and required libraries were transferred.



The final step was to compile the code and run it on the Raspberry Pi/sensorian shield.

The terminal commands below demonstrate the compiling and running of the required source files.

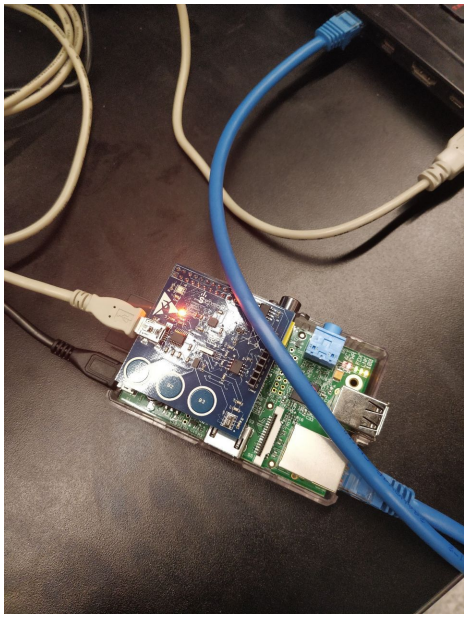
gcc bcm2835.c Utilities.c Lab1.c -lbcm2835 -o lab1

The “gcc” command invokes the compiler we are using in Linux. Along with it, the “-l” and “-o” options are used. “-l” tells the compiler where to find an external header file, and “-o” names the executable (lab1 in this case).

After compiling the files, an executable is created. Thus, the only step left to do is to run the executable. This was done by typing the command: **sudo ./lab1**

```
pi@raspberrypi:~/Desktop/libs(1) (1) $ gcc bcm2835.c Utilities.c Lab1.c -lbcm283
5 -o lab1
pi@raspberrypi:~/Desktop/libs(1) (1) $ ls
bcm2835.c  bcm2835.h  lab1  Lab1.c  Utilities.c  Utilities.h
pi@raspberrypi:~/Desktop/libs(1) (1) $ sudo ./lab1
```

The correct functionality occured on the sensorian shield - the LED was turned off/on every 3 seconds.



```
#include "bcm2835.h"
#include "Utilities.h"

void main()
{
bcm2835_init();
pinModeOutput(LED_PIN);

while (1)
 {
     digitalWrite(LED_PIN, HIGH);
     delay_ms(3000);
     digitalWrite(LED_PIN, LOW);
     delay_ms(3000);
 }
}
```

Conclusion

At the end of the lab session, our group learned many useful fundamentals. Namely, how to securely transfer files from a host computer to the Raspberry Pi/sensorian shield using SSH and SCP. We also gained experience setting up the development environment in Linux via a virtual machine. Lastly, our team gained hands-on experience with setting up network configurations between a host computer and a remote computer. All of this knowledge will lead to developing IoT software and utilizing IoT hardware functionalities.