LAB #2: CROSS COMPILE AND PYTHON GPIO INTRODUCTION

SOFE 4610U: Internet of Things

Devante Wilson - 100554361

Youssef Osman - 100715637

Pre Lab Questions

What microprocessor does the RPi use?

The raspberry Pi 3 uses the ARM Cortex-A53 processor.

What is the kernel version of the Ubuntu computer you are using.

The kernel version of the Ubuntu image is 14.0.3 generic. The command **uname -r** was used to determine.

Find the same information about the ARM Linux board you'll be using.

The same processor listed above.

Lab Activity

Introduction

The proceedings of this lab report detail our group getting an introduction to cross compiling on different platforms and Python GPIO interaction. We also gain experience and understanding of system environment variables. Lastly, we become familiar with compiler techniques.

Lab Tasks

The first step of the lab tasks was to clone the github repository by running the command:

git clone https://github.com/raspberrypi/tools/

Next, the tools folder was unzipped and the path needed to be updated via environment variables. To update the path, the command: **export PATH** was used (as shown in the screenshot below). We used the 64-bit version as the Ubuntu host architecture utilizes this.

Lab 2: Cross Compile and Python GPIO Introduction

Our next task was to write a HelloWorld program in C and then cross-compiling it. To compile the program, the command used was: **gcc -o hello Hello.c** and **./hello**

```
youssef@youssef-virtual-machine:~$ cd Desktop/
youssef@youssef-virtual-machine:~/Desktop$ gcc -o hello Hello.c
youssef@youssef-virtual-machine:~/Desktop$ ./he
bash: ./he: No such file or directory
youssef@youssef-virtual-machine:~/Desktop$ ./hello
Hello world!
youssef@youssef-virtual-machine:~/Desktop$
```

After the program was compiled and run, we then tried to create a Makefile in order to automate the compiling process for the program.

```
youssef@youssef-virtual-machine:~/Desktop/pi/tools/arm-bcm2708/gcc-linaro-arm-li
nux-gnueabihf-raspbian-x64/bin$ arm-linux-gnueabihf-gcc -o hello /home/youssef/D
esktop/Hello.c
```

```
youssef@youssef-virtual-machine:~/Desktop$ cd /home/youssef/Desktop/pi/tools/arm
-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin
youssef@youssef-virtual-machine:~/Desktop/pi/tools/arm-bcm2708/gcc-linaro-arm-li
nux-gnueabihf-raspbian-x64/bin$ ./hello
bash: ./hello: cannot execute binary file: Exec format error
youssef@youssef-virtual-machine:~/Desktop/pi/tools/arm-bcm2708/gcc-linaro-arm-li
nux-gnueabihf-raspbian-x64/bin$
```

Once the compile attempt gave an error on our Linux system, our team transferred the executable file to the Raspberry Pi. To accomplish this, Winscp; it is used as a method of secure file transfer.

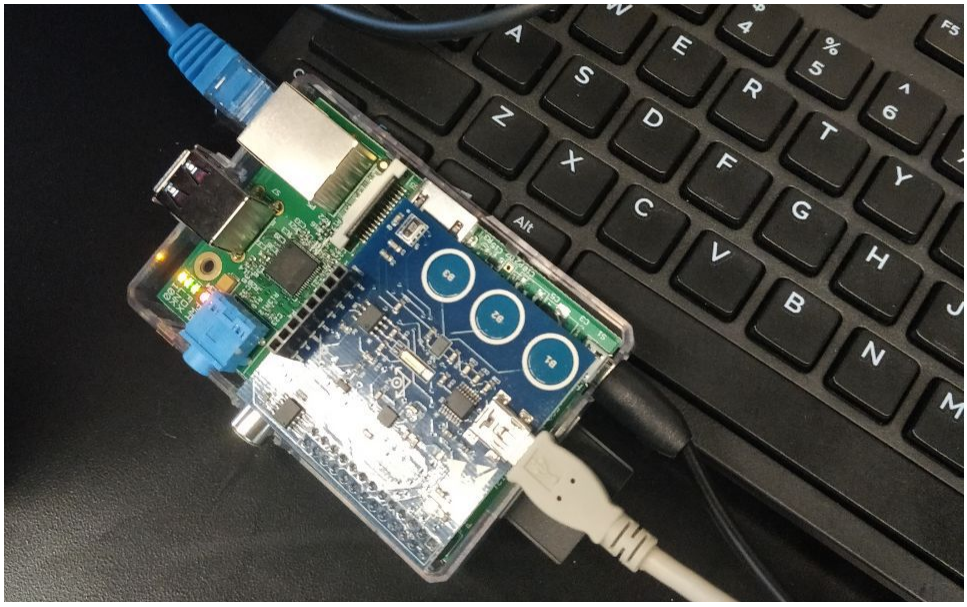
We were able to run the c program successfully after it was loaded onto the Raspberry Pi (screenshot shown below). We had to make the “hello” file into an executable by using the command **chmod +x hello**

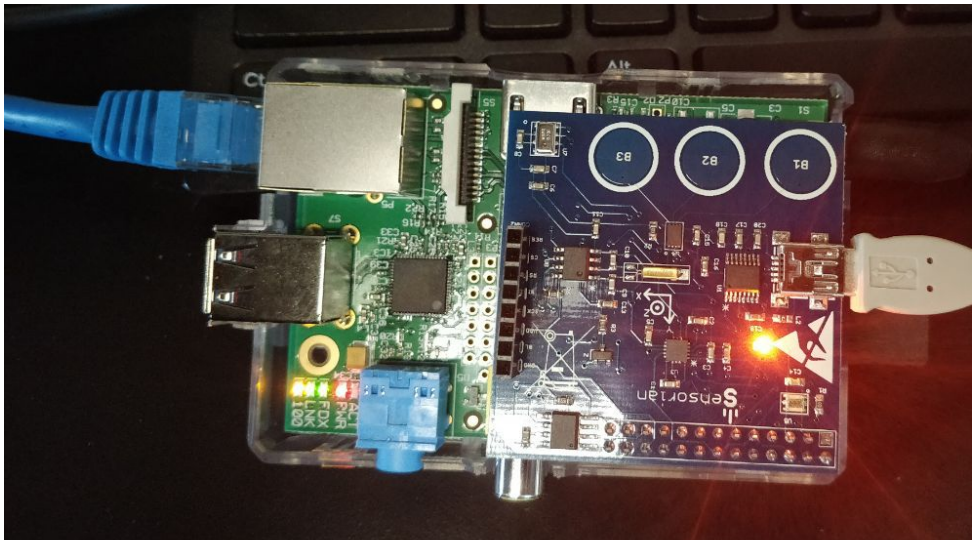
```
pi@raspberrypi:~/Pictures $ chmod +x hello
pi@raspberrypi:~/Pictures $ ls
hello
pi@raspberrypi:~/Pictures $ ls ltr
ls: cannot access ltr: No such file or directory
pi@raspberrypi:~/Pictures $ ls -ltr
total 8
-rwxr-xr-x 1 pi pi 5977 Oct  5  2018 hello
pi@raspberrypi:~/Pictures $ ./hello
Hello world!
pi@raspberrypi:~/Pictures $
```

Lastly, a python script was modified to turn on/off the LED on the Sensorian Shield for 1 second. **Note: the script was run as root (superuser/sudo).**

To run the python script, the command **sudo python example.py** was used.

RESULT





PYTHON code:

```
#!/usr/bin/env python

# -*- coding: utf-8 -*-


# import required libraries

import RPi.GPIO as GPIO

import time


# perform setup

GPIO.setmode(GPIO.BOARD)

GPIO.setup( 12 , GPIO.OUT)


# perform ON/OFF LED switching

for i in range ( 10 ):

  # set LED ON through GPIO pin

  GPIO.output( 12 , GPIO.HIGH)

  # set sleep delay for a second

  time.sleep( 1 )

  # set LED OFF through GPIO pin

  GPIO.output( 12 , GPIO.LOW)

  # set sleep delay for a second

  time.sleep( 1 )
```

Conclusion

By the end of the lab session, our team successfully learned the concepts of cross compiling on different platforms. Additionally, we learned about Python GPIO interaction and gain experience with setting system environment variables in Linux. Lastly, we become familiar with compiler techniques (using GNU compiler, Makefiles, and the ARM compiler).