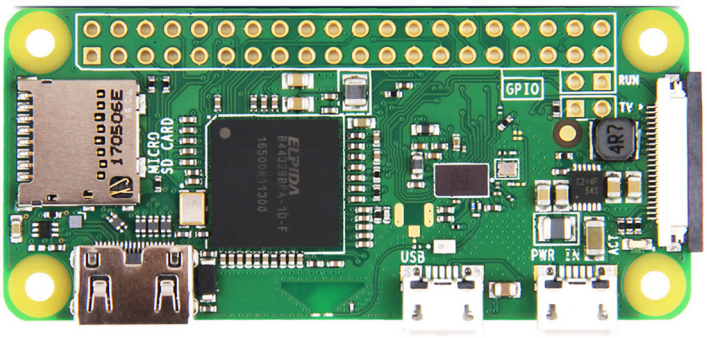
University of Cape Town



EEE3096S/EEE3095S

Embedded Systems II

Work Packages 2021

---

2 August 2021

Abstract

Welcome to the practicals for EEE3096S. These instruction are applicable to all practicals so please take note! It is critical to do the pre-practical/tutorial work. Tutors will not help you with questions with answers that would have been known had you done the pre-practical work. The UCT EE Wiki (wiki.ee.uct.ac.za) is a very useful point to find any additional learning resources you might need. Use the search functionality on the top right of the page.

Practical Instructions

- **Naming**
  All files submitted to Vula need to be in the format. Marks will be deducted if not.

```
pracnum_studnum1_studnum2.fileformat
```

- **Submission**
  - Work packages consist of a tutorial and a practical. Be sure you submit to the correct submission on Vula!
  - All text assignments must be submitted as pdf, and pdf only.
  - Code within PDFs should not be as a screenshot, but as text.
  - Where appropriate, each pdf should contain a link to your GitHub repository.
- **Groups**
  All practicals are to be completed in groups of 2. If a group of 2 can't be formed, permission will be needed to form a group of 3. You are to collaborate online with your partners. See more here.
- **Mark Deductions**
  Occasionally, marks will be deducted from practicals. These will be conveyed to you in the practical, but many will be consistent across practicals, such as incorrectly names submissions or including code as a screenshot instead of formatted text.
- **Late Penalties**
  Late penalties will be applied for all assignments submitted after the due date without a valid reason. They will work as 20% deduction per day, up to a maximum of 60%. After this, you will receive 0% and have the submission marked as incomplete.

Contents

Chapter 1

Work Package 1

1.1 Tutorial - Pi Setup and Linux Basics

1.1.1 Overview

This tutorial sets out to familiarise you with the Pi and to complete a simple programming task. If you have not yet done so, setup your Pi.

There is tons of Raspberry Pi information on the EE Wiki, with an overview page accessible here. It's suggested you take a look at what information is available to you.

Due date: See Vula

1.1.2 Pre-tut Tasks

- Set up your Pi and ensure it's configured correctly! See the Wiki for details http://wiki.ee.uct.ac.za/RaspberryPi:Installation
  There is also a video here. While it was created on Windows, the steps for setting them up are universal.

- Add internet access to your Pi. The easiest way is through WiFi, which you can read about here. If you do not have access to WiFi, you can read about Ethernet passthrough here.
- Get a basic understanding of git. For details regarding git and it's usage, read through http://wiki.ee.uct.ac.za/Git

1.1.3 Pre-tut Requirements

This section covers what you will need to know before starting the practical.

- Have your Raspberry Pi Set up correctly.
- Have an understanding of how you can edit text files on the Pi using an editor such as vim or nano.
- Have a basic understanding of Git.

1.1.4 Hardware Required

- Raspberry Pi with configured SD Card

1.1.5 Outcomes of this Tutorial

You will learn about the following topics:

- GDB Debugging
- Linux and Bash Basics

1.1.6 Deliverables

At the end of this tutorial, you must:

- Submit a single PDF with your screenshots and answers to the questions from the Terminal task to "Tutorial 1" on Vula. A link to your shared repository with your partner should also be included.

1.1.7 Walkthrough

This tutorial consists of two parts - a terminal task and a programming task.

Terminal Task

While there are many Graphic User Interfaces (GUIs) available for various distributions of Linux including Raspbian, it is often necessary to operate an embedded system without a GUI given the limited processing and memory resources of the device. Consequently it is important to learn to use Linux through the command line shell. A very common shell on many Linux systems is Bash. Learning about and being comfortable with the command line will help you greatly in working with embedded systems. The tasks below will introduce the basics to you, but you should consult the wiki for more information. Some basic terminal commands can be found here.

Start by SSH'ing into your Pi and create a folder called <your_student_number>.
Run the following commands, and take a screenshot after each command:

- $ ls (must show the folder you just created)
- $ ifconfig
- $ lscpu
- $ vcgencmd measure_temp

Your result should look something like this:

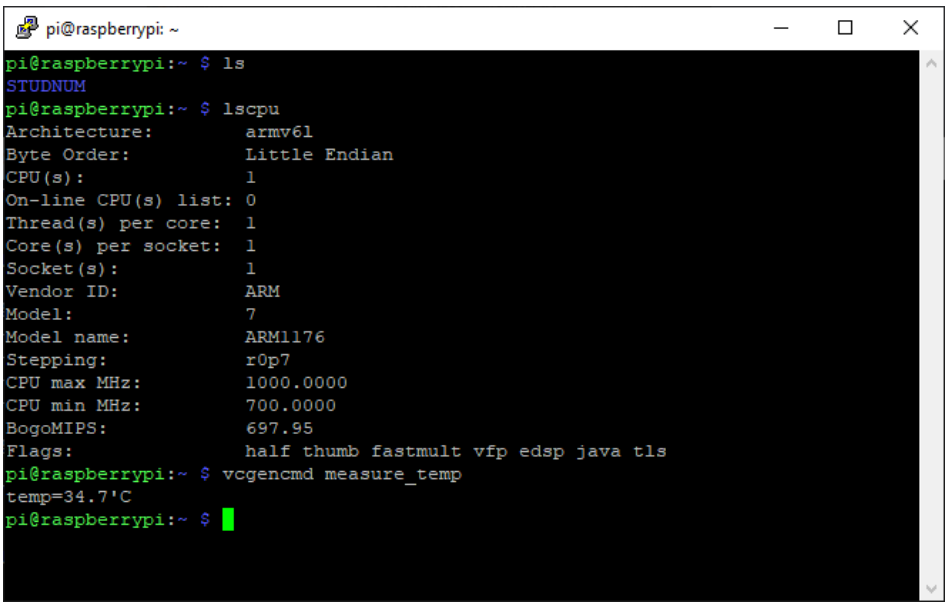


Figure 1.1.1: Example output after running the **ls**, **lscpu** and **vcgencmd measure_temp** command

Answer the following questions related to Git:

1. What is the purpose of using Git?
2. List the four commands you would use to commit the file 'changes.txt' (assuming the file has been changed since the last commit) to Git and push it to the GitHub repository https://github.com/fake/link.git
3. What does it mean for a file to be:
   (a) untracked
   (b) staged
   (c) committed

Programming Task

For this task we'll be working with C/C++. In order to do so, you need to have a suitable compiler. Ubuntu and most linux Distros (including Raspbian on the raspberry Pi) have this by default. On Windows however, you'll need MinGW. For a guide on how to install and configure that, visit this page on the EE Wiki. The solution of this task should pushed to a shared repository with your partner and a link to the repository should be included in the report.
By the end of this task, we expect you to:

- Be comfortable working in terminal/on the command line
- Understand real-time debugging
- Be familiar with at least 1 development and debugging tool-chain

Compilers will be covered in more detail in Prac 2, however here is a useful link on GDB on the EE Wiki.

Simple GDB Walkthrough

To learn about gdb, let's create a very simple[1] calculator.

1. Create a file called "main.c" on the command line:
   Ubuntu: `$touch main.c`
   Windows: `type nul > main.c`
2. Open the file in your favourite editor.:
   Ubuntu: `$nano main.c`
   Windows: `notepad main.c`
   However it is recommended you use notepad++ which can be downloaded here.
   To start notepad++ from the command line, use `start notepad++ main.c`
3. Inside it, place the following code:

```
1  # include <stdio.h>
2  int main(){
3          int a, b, sum;
4
5          printf("Enter a value for a: ");
6          scanf("%d", &a);
7
8          printf("Enter a value for b: ");
9          scanf("%d", &a);
10
```

[1] Incredibly simple.

```
11          sum = a + b;
12
13          printf("The sum of a and b is %d \n.",sum);
14  }
```

4. Compile it with the debugging flag:
   Ubuntu: `$ gcc -g main.c`
   Windows: `gcc -g main.c`
   Note that this assumes you've correctly installed and configured MinGW

5. Run it!
   Ubuntu: `$ ./a.out`
   Windows: `a.exe`
   If you try add two numbers - you'll see something really odd happen! Let's use `gdb` to figure it out

6. Open it in gdb:
   Ubuntu: `$ gdb a.out`
   Windows: `gdb.exe a.exe`

7. We need to add breakpoints. Let's add them at a points where we need to validate data:

   (a) Line 7 - once we have a value for a
   `break 7`

   (b) Line 10 - once we have a value for for b
   `break 10`

   (c) Line 12 - once we've added them together
   `break 12`

8. Run the application inside gdb!
   `run`

   (a) You'll be prompted to enter in a value for a. Enter in something simple, such as 3

   (b) Once you hit the enter key, gdb will step in. Let's print the value of a to ensure it is 3
   `print a`

   (c) If we're happy that it's the value we entered, we can continue execution by entering in `c` and pressing enter.

   (d) We're prompted for a value for b. Let's do another simple number, 5
   (Please note Windows may not prompt you for the next number, but you can enter

it anyway)

(e) gdb now steps in again. Let's validate that the value for b is indeed 5.
`print b`
It's not! Let's double check the value of a with `print a`
The variable a was assigned the value of 5!

(f) We now know that when we enter out value for b, it's being assigned to a. We also know to look between lines 9 and 11 for our error. Close gdb by typing "q" and pressing enter. You will be prompted to kill the debugging session, do so by typing "y" and then pressing enter.

9. Look for the error.
Do you see it? Here's a hint: It was caused by a copy-paste that went unedited.
That's right! Line 10 assigned the value entered for b to the variable a. As a result, a gets updated with the intended value for b, and, because b is never initialised or assigned, it holds a random value from whatever might be in memory.

10. Fix up the error, and run the calculator again. It should now work as expected.

Push your corrected code into your shared repository and add a link to it in your submission.