1 Practical 1 - Git, Bash, GPIO

1.1 Overview

This practical sets out to familiarise you with the Pi and complete a simple programming task. If you have not yet done so, complete Prac 0. Ensure that you can SSH into the pi, as all the tasks for this prac require it.

To be completed individually.

Due date: See Vula

1.2 Pre-Prac Tasks

- Complete Prac 0 to have your Pi set up and configured
- Read the sections "Inputs" and "Outputs" on the RPi.GPIO documentation, available here
- Update your pracsource git repository, as there has been an update to the Python template

1.3 Pre-prac Requirements

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

- Have your Raspberry Pi Set up as per the requirements of Prac 0.
- Have an understanding of how you can edit text files on the Pi using an editor such as nano, or using VNC and a GUI-based editor.
- Have a basic understanding of Git

1.4 Hardware Required

- Raspberry Pi with configured SD Card
- RPi Power Supply
- Ethernet Cable
- A breadboard
- 2 x push buttons
- 3 x LEDs
- 3 x Resistors
- Dupont Wires

1.5 Outcomes of this Practical

You will learn about the following topics:

- Basic GPIO Usage
- Interrupts
- Debouncing
- Git
- Bash
- SSH

1.6 Deliverables

At the end of this practical, you must:

- Demonstrate your working implementation of the binary counter to a tutor
- Single PDF with your screenshots from the Terminal task to Vula in a pdf document. This PDF document should also contain a link to your GitHub repository.

1.7 Walkthrough

This practical consists of two parts. The terminal task can be completed at home and does not require demonstration to a tutor.

1.7.1 Terminal Task

While there are many Graphic User Interfaces (GUI) 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 basics to you, but you should consult the manual for more information.

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 (eth0 must be visible)
- $ lscpu (number of cores must be visible)

Your result should look something like this:

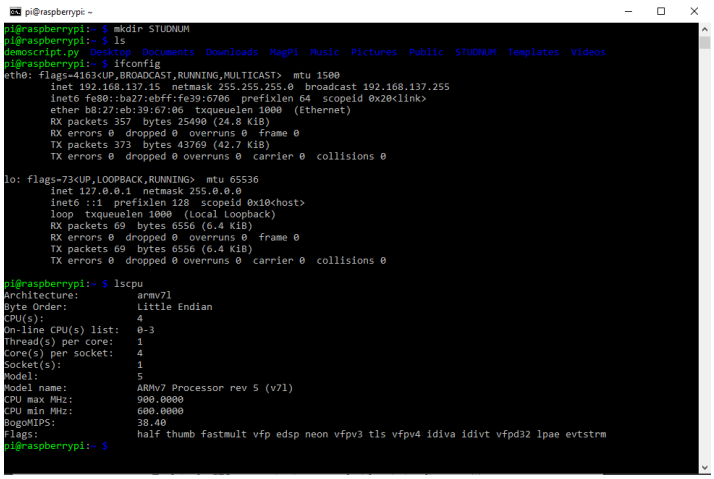

Figure 1: Example output after running the above commands

1.7.2 Programming Task

In this task you will develop a simple 3-bit binary counter, the values of which will be placed on LEDs connected to the Pi. The value on the counter should change depending on a button press. Be sure to use git to keep track of changes to your code. For instructions on using Git, refer to the lab handbook.

- Start by connecting to your Pi through VNC or SSH (see lab handbook)
- If not installed, install the Python GPIO libraries, as explained here. (They should be installed by default.)
- Connect 3 LEDs and 2 Push buttons to the GPIO pins of the Pi - taking care of which pins to use. Look at pinout.xyz and be sure to not use special purpose pins
- Read the Python Tips and tricks in the lab handbook to gain understanding of the commands and why they are used.
- Create a copy of the template in the mypracs folder.
- Write your code. It's suggested you work incrementally, committing your code using git each time you accomplish something.
  - Take a look at the RPI.GPIO examples, available here
  - Start by turning on and off a single LED in the main() method. Delay between toggling the GPIO by using something like `time.sleep()`
  - Now, use a button to toggle the state of the LED. Look at the lab handbook for help with debouncing and interrupts in Python.
  - Finally, implement a system that displays a binary value on 3 LEDs. Use one button to increase the value, and another button to decrease the value. The values should wrap around (i.e. increasing "111" should then display "000" and decreasing "000" should display "111"

- Hints
  - Look at *itertools.product* to generate a list of binary values.
  - Use a single integer counter as an index to the Python array that you've created. Make use of Python's `global` construct to do so
- Demonstrate this implementation to a tutor to get signed off, and push your code to GitHub.
- Submit a PDF as per the deliverables section

1.8 Mark Allocations

Marks will be given for:

- Correctly completing the terminal task
- Well structured and commented code
- Meaningful git commit messages
- A correct demonstration

Marks will be deducted for:

- Not using interrupts and debouncing on your button presses
- Not submitting files in the correct format
- Not linking to the practical on GitHub
- Copying another student's code
- Late submission