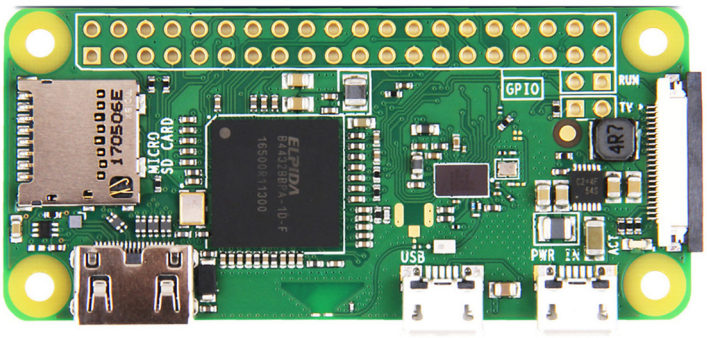
University of Cape Town



EEE3096S/EEE3095S

Embedded Systems II

Work Packages 2021

---

6 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

1 Work Package 1 **3**

1.1 Practical - Using a Real Time Clock 3

1.1.1 Overview 3

1.1.2 Outcomes and Knowledge Areas 4

1.1.3 Deliverables 4

1.1.4 Hardware Required 4

1.1.5 Walkthrough 4

1.1.6 Some Hints 5

1.1.7 Submission 5

Chapter 1

Work Package 1

1.1 Practical - Using a Real Time Clock

1.1.1 Overview

Before connecting to the internet for the first time, you may have noticed that the time on your Pi is a little strange. This is because the Pi doesn't have what is called an RTC (real time clock). Instead, it relies NTPD (Network Time Protocol Daemon) to fetch, set and store the date and time. However, this might be problematic as you may not always have an internet connection, and if your Pi doesn't have the correct localisation options, you may end up with the wrong time due to timezone settings. It's possible to add an RTC to the Raspberry Pi to hold the system time correctly, but for this practical we're simply going to interface with the RTC using I2C, and set the time using buttons and interrupts.

Design overview

In this prac, you will be using C to write to time to an RTC module. You will also have an LED that will flicker on and off every second, as well as have two buttons to change the time.

The buttons should be connected using interrupts and debouncing, which do the following:

1. Button 1 - Fetches the hours value from the RTC, increases it by 1, and writes it back to the RTC.
2. Button 2 - Fetches the minutes value from the RTC, increases it by 1, and writes it back to the RTC.

You cannot use any time libraries in your script, i.e. make sure you are using the I2C communication protocol between the RTC and Pi in your script.

1.1.2 Outcomes and Knowledge Areas

You will learn about the following aspects:

- I2C
- Real Time Clocks
- You should acquaint yourself with the Wiring Pi documentation, available here.

1.1.3 Deliverables

For this practical, you must:

- Demonstrate your working implementation to a tutor
- Submit a single PDF with your answers to the questions to "Practical 1" on Vula. It should also contain your code. (Do not paste a screenshot of your code)

1.1.4 Hardware Required

- Configured Raspberry Pi
- A breadboard
- 1 LED
- 1 Resistor
- 2 x push buttons
- Dupont Wires
- DS3231 RTC Module

1.1.5 Walkthrough

1. Start by updating and upgrading your pi. You can do this by running `$ sudo apt-get update && sudo apt-get upgrade`
2. If you haven't already, enable I2C in raspi-config. You can do this by running `$ sudo raspi-config` and navigate through the menu.
3. Do a git clone of the prac source files from: https://github.com/UCT-EE-OCW/EEE3096S-Pracs-2021. The prac we're concerned with is Prac 1! Copy the 'WorkPackage1' directory into your own Prac directory to avoid trying to push to the EE Repository
4. Build the circuit by connecting the LED and the buttons. You can see the pinout of the pi from pinout.xyz.
5. Connect the RTC Module by aligning pin 1 of the Module with pin 1 on the pi.
6. Run `$ gpio i2cdetect` to see if you can see the RTC. Update `BinClock.h` with the address.

7. Open `BinClock.c` and write the code required. Some function templates are made available to give you a guide, but you may be required to write more functions as you see fit. Function definitions are placed in `BinClock.h`
    - Be sure to take care with regards to which pin numbering is used. You can run `$ gpio readall` to check pinouts and current assignments
8. To build and run the script, run `$ make` and `$ make run`. To use GDB, run `$ make run_gdb`. To delete the generated build files, you can run `$ make clean`
9. This prac uses a makefile to build the source code. There also exists cmake that further simplifies the build process. If you want, you can look at this tutorial and try to replace the Makefile with cmake.

1.1.6 Some Hints

1. Read the Docs. This includes the datasheet for the RTC (which you will absolutely have to do), as well as the documentation of Wiring Pi.
2. You will need to debounce your button presses. You can use hardware debouncing, but this circuit is a bit complex, so to save space we recommend software debouncing.
3. The source code provided to you has a lot of implementation in it already. Ensure you read and understand it before embarking out on the practical.
4. You can read more tips and guidelines on programming the pi in C on the Wiki page

1.1.7 Submission

Please Answer the following questions

1. What is I2C?
2. What is an interrupt and why is it important in the world of Embedded Systems?
3. Embedded Systems Good Practices
    (a) Why do we use pull up and pull down resistors?
    (b) Explain the difference between hardware debouncing and software debouncing, with an example of how you might implement each (i.e draw a circuit diagram and write a code snippet).
    (c) What is "polling" in this context, and why is it better to use an interrupt over polling?
4. The circuit diagram of your implementation.

5. The Core Functions of your code

6. A Github link to your shared repository.