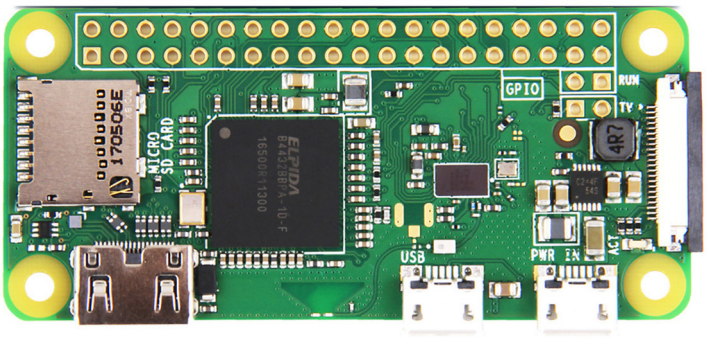
University of Cape Town



EEE3096S/EEE3095S

Embedded Systems II

Work Packages 2021

---

13 September 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!
  - If working in a prac group, do only one submission to Vula for the team (i.e. by one of the team members). Ensure all files submitted to Vula are in the format: pracnum_studnum1_studnum2.fileformat (although if you are submitting a zip file, you do not need to have all the sub-files named according to the student numbers, but do please have the report filename containing student numbers to ensure the marker is aware that it is a team submission).
  - 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 10% deduction per day, up to a maximum of 50%. After this, you will receive 0% and have the submission marked as incomplete.

Contents

1 Practical - ADC on the Pi **3**

1.1 Circuit 3

1.2 Walkthrough 4

1.3 Requirements for submission 5

Work Package 4

1 Practical - ADC on the Pi

This prac is about using an ADC to digitize an analogue signal, the analogue signal being a voltage corresponding to the current sensed temperature as well as a light reading. It will require you to configure some basic interfacing aspects before integrating them all into the final program to digitize the temperature signal. Part of the deliverables for this prac is a short demo video which you could record on a smartphone and upload as part of the assignment, see Section 1.3.

In this prac you're going to generate temperature and light reading data every 10 seconds, by sampling a temperature sensor and Light dependent resistor connected to the ADC. We're going to be using the Adafruit MCP3008 Library for Python.

1.1 Circuit

You need to connect the following:

- MCP3008 CLK to Pi SCLK
- MCP3008 DOUT to Pi MISO
- MCP3008 DIN to Pi MOSI
- MCP3008 CS/SHDN to Pi CE0
- MCP3008 VDD to Pi 3.3V
- MCP3008 VREF to Pi 3.3V
- MCP3008 AGND to Pi GND
- MCP3008 DGND to Pi GND
- Read the data sheet for the MCP9700 and connect it correctly to channel 1 (pin 2) of the ADC.
- Connect the LDR to channel 2 of the ADC
- Add a button on a suitable GPIO pin

Figure 1.1: The pinout of the MCP3008

1.2 Walkthrough

1. Enable SPI using the raspi-config tool
2. Install the MCP3008 Adafruit library

```
$ sudo apt-get update
$ sudo apt install build-essential python3-dev python3-smbus python3-pip
$ sudo pip3 install adafruit-circuitpython-mcp3xxx
```

3. Build the circuit described above.
4. Create a new python file
5. Place the following code in it:

```
import busio
import digitalio
import board
import adafruit_mcp3xxx.mcp3008 as MCP
from adafruit_mcp3xxx.analog_in import AnalogIn

# create the spi bus
spi = busio.SPI(clock=board.SCK, MISO=board.MISO, MOSI=board.MOSI)

# create the cs (chip select)
cs = digitalio.DigitalInOut(board.D5)

```

```
# create the mcp object
mcp = MCP.MCP3008(spi, cs)

# create an analog input channel on pin 0
chan = AnalogIn(mcp, MCP.P0)

print('Raw ADC Value: ', chan.value)
print('ADC Voltage: ' + str(chan.voltage) + 'V')
```

If you run this code you will see the reading from the LDR. Cover the sensor and run the code again. The reading should change.

6. Adapt your code to read from both sensors and print to terminal every ten seconds. You **must** use a thread. You can read about simple threading in Python here. The output of your application should look as follows:

```
Runtime     Temp Reading    Temp                    Light Reading
0s          <ADC value>     <converted temp>  C     <ADC value>
10s         <ADC value>     <converted temp>  C     <ADC value>
20s         <ADC value>     <converted temp>  C     <ADC value>
30s         <ADC value>     <converted temp>  C     <ADC value>
40s         <ADC value>     <converted temp>  C     <ADC value>
50s         <ADC value>     <converted temp>  C     <ADC value>
60s         <ADC value>     <converted temp>  C     <ADC value>
```

7. Runtime must be calculated, and not be implemented as a counter
8. Add a button to toggle the rate of sampling between 10s, 5s and 1s

1.3 Requirements for submission

There are two items you need to uploaded to Vula for submission of this practical assignment:

1. Your prac report provided as a single PDF file, named `Prac4_<studnum1>_<studnum2>.pdf`
2. A short demo video in which you present your design and implementation (see demo marking guide in Table I below). You can upload the demo video where you please, Just include a link to it in the report.

Report Requirements

The report should contain the following:

1. The circuit diagram you used

2. A paragraph on your validation and testing for the values you got from the ADC for both light and temperature
3. Your Python code (screenshots of code will not be marked)

Marks will be awarded for:

- Neatness (circuit diagram and code)
- Correctness
- Using threads correctly
- Correct implementation of the temperature reading algorithm

Demo Marking Guide

Marks will be deducted for:

- Not following instructions
- Plagiarism/Copying

You can prepare a short video, about 5 minutes will probably be sufficient. There is no minimum duration, but you need to respond to the aspects as listed in the marksheet below; but we do *not* want videos beyond 10 minutes in length. The demo should respond to the following aspects (the marking allocations are shown on the right):

Table I: The Mark sheet for the demo aspect of Prac5

Category	**Item**	**Description**	Max Mark	Attempt 1
Intro	Intro and division of workload	Start off the demo by indicating your prac team members. Indicate how you divided up the work	6	
Design	Design and connections	Indicate the way you hooked up the ADC (using a circuit diagram or alternate clear visualization) and present the software design e.g block diagram or flow chart of your program	5	
		Point to significant parts of your code that concerns the hardware and any timing aspects (use of comments, and reminders in the code, e.g. of things to draw attention to in the demo are recommended)	5	
ADC Test	Testing Implementation	Run through the operation of your prototype (e.g. increase light decrease light, etc)	10	
Conclude	Conclude the presentation	Report on the extent the test was successful provide some suggestions for how you might improve things or add features if time allowed	4	
Mark			30	