# University of Cape Coast

## College of Agriculture and Natural Sciences

## School of Physical Sciences

### Department of Computer Science and Information Technology

### End of First Semester Examination

Duration: 3 hours

# Examination exercises: The at24c32 EEPROM

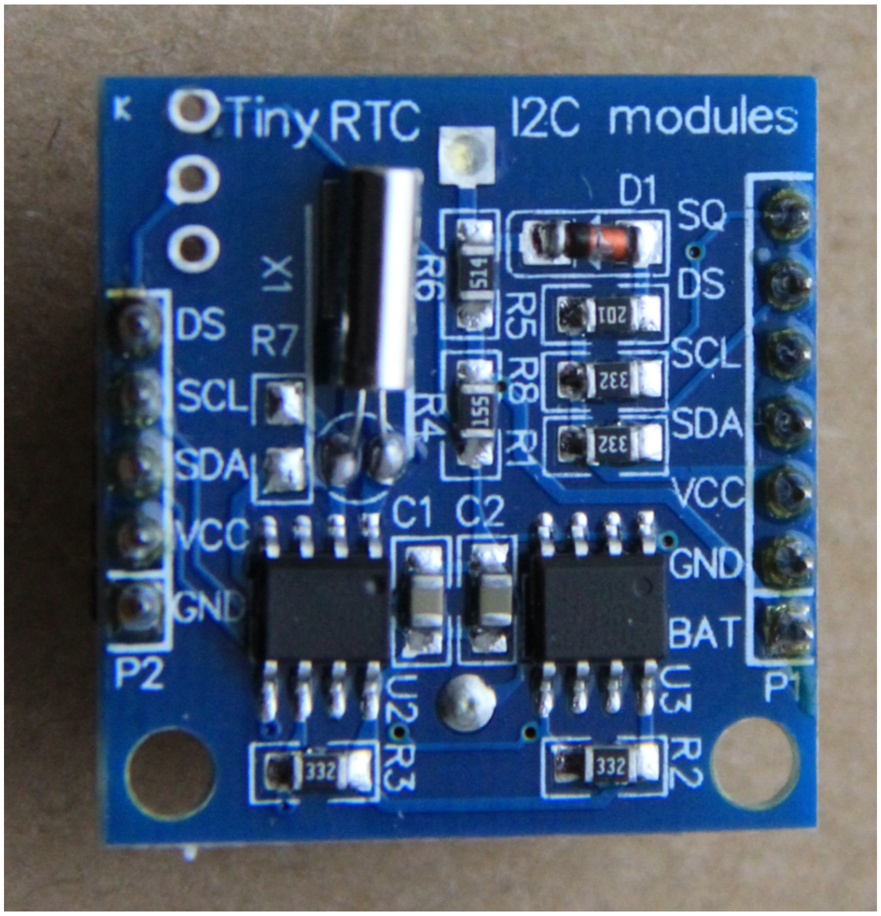
## Goal:

The goal of this exercise is twofold: First we will learn about an EEPROM, a **E**lectrically **E**rasable **R**ead **O**nly **M**emory. These devices are often used in instruments to store calibration values or configuration parameters. EEPROMs can usually be read byte by byte but can be erased and written only in blocks.

The at24c32 EEPROM is mounted together with the DS1307 RTC on the Tiny RTC module (chip on the right). The Tiny RTC module therefore responds to 2 I2C addresses. You find these 2 addresses with the i2cdetect command and you can figure out from the data sheets which of the addresses corresponds to which chip.

The data sheet for the at24c32 can be found here:

http://www.atmel.com/Images/doc0336.pdf



**For the examination please solve exercise 1-3 of this exercise list.**

**For all three exercises, write**

- the C code
- a Makefile
- a shell script that sets up the LD_LIBRARY_PATH to include /opt/ucc/lib, such that the at24c32 library can be found (not necessary of course if you do not use the at24c32 library)
- a short description on how you designed the program and what the individual steps are that the program takes to solve the problem.

## Exercise 1: Read the EEPROM

The at24c32 is a 32 kBit device and can therefore store a total of 4096 bytes (4096 * 8 = 32 kBit). Like the DAC it is an I2C device. To make things easier for you, again a library is supplied.

The documentation of the library is again generated from the source code and can be found at the URL:

https://dcsit.twiki.ucc.edu.gh/html/libDoc/at24c32

This one is an exercise for detectives: There is a secret message hidden in the EEPROM and your job is to uncover the mystery. So… Sherlock Holmes and Dr. Watson: show your talents and find out the contents of the message. Write a program called *readEnigma* reading the EEPROM and printing its content in hex (%x) and as characters (%c).

## Exercise 2: Re-write the program without using the library

Since there are only very few library calls it is also possible to access the EEPROM through pigpio calls avoiding my library all together. Make sure you get the same result as in the first exercise.

## Exercise 3: Write the EEPROM

Prepare a buffer with some data to be written to the EEPROM. Use your previous program to read these data back. Power down the Raspberry Pi, wait a minute and then switch it back on again. Verify that you can still read back the data.