

ECE471: Embedded Systems – Homework 9
Temperature Display

Due: Wednesday, 26 November 2014, 12pm (noon) EST

1. Use your Raspberry Pi for this homework.

You will need the i2c display from Homework 5 as well as *either* the MCP3008/TMP36 from Homework 7 *or* the DS18B20 sensor from Homework 8 (your choice).

2. **Part 2: Writing to the Display**

Take one of your temperature reading homeworks as a basis for this project. Copy your code over `display_temp.c` and get it displaying the temperature to the screen.

Now hook up the i2c LED display, and make it display the temperature (in F or C, your choice), updated once per second. Feel free to re-use code from earlier homeworks.

Your code should handle four cases:

- (a) Temperatures between 0 and 99.9 degrees, inclusive. These should be displayed as two digits, a decimal point, another digit, and then a degree symbol (which is just a crude circle made of the top 4 segments on the display).
- (b) Temperatures between -99 and 0 degrees. These should display a minus sign and then two digits of temperature, then the degree symbol.
- (c) Temperatures between 100 and 999 degrees should print three digits of temperature, then the degree symbol.
- (d) Invalid temperatures that won't fit the display (and errors reading the thermometer) should be reported in a method that isn't a valid temperature. It is your choice how to indicate this.

To test the above you can first write the display code (maybe as a separate function) and hard-code the value to display. Then once it works on all of the possibilities, then hook it up to your temperature reading code.

3. **Something Cool**

No something cool for this homework. Put any coolness to use in your final project.

4. **Questions**

Edit the README file to have your name and answer the following questions.

- (a) Name one example of how poorly written embedded code can have disastrous effects in a product.
- (b) Why might it be good to always try to write correct, documented, well tested code even if you think it's not going to ever be used in anything important?

5. **Linux Fun**

When a file is created or modified on Linux various timestamps are updated. `atime` (last access time) `mtime` (last modified time) and `ctime` (creation time).

The `ls -lt` (that's a lowercase l) will show all files and their last modified time.

The Linux `touch` command will update the timestamps on a file to the current time (and create the file if it doesn't exist). You can also specify the time. You can do things like

```
touch --date "1983-10-16 14:40" blah
```

which will update the timestamp on the file `blah` to the specified date.
You can also do fun things like

```
touch --date "next Thursday" blah
```

- (a) Use `touch` to change the file creation time of the "fakedate" file (included with the test code) with a date from some other year (not 2014).
- (b) What happens if you try to create a date in the year 2044? Why?
- (c) I have had students turn in homework late, but telling me "check the file timestamp, it shows I finished it before the deadline". Why might I not take this as a good argument?

6. Submitting your work

- Run `make submit` which will create a `hw9_submit.tar.gz` file containing `Makefile`, `README`, `display_temp.c`, and `fakedate`. You can verify the contents with `tar -tzvf hw9_submit.tar.gz`
- e-mail the `hw9_submit.tar.gz` file to me by the homework deadline. Be sure to send the proper file!