ECE471: Embedded Systems – Fall 2026

Homework 7: SPI, A/D and Temperature Probe

Due: Friday, 30 October 2026, 5:00pm

1. Use your Raspberry Pi for this homework.
   You will need an MCP3008 SPI A/D converter as well as a TMP36 temperature sensor (looks like a transistor) that I handed out in class. If you missed class, you can stop by my office to pick these up.

   You can view the datasheet for the MCP3008:
   https://web.eece.maine.edu/~vweaver/classes/ece471_2014f/datasheets/MCP3008.pdf
   You can view the datasheet for the TMP36:
   https://web.eece.maine.edu/~vweaver/classes/ece471_2014f/datasheets/TMP35_36_37.pdf

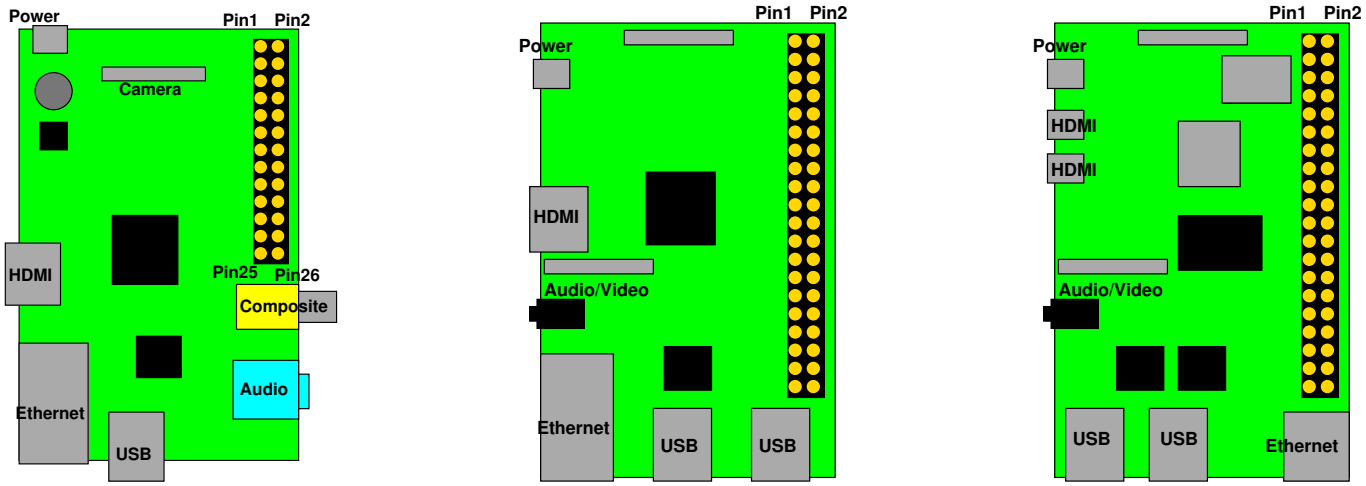


Figure 1: Location of GPIO header on Raspberry Pi Model B, B+/2/3, and 4B/5B



Figure 2: TMP36 and MCP3008 Pinouts

2. **Get the MCP3008 providing values over SPI** (3 points)

   First wire up the SPI device to the Raspberry Pi. You can use Figure 1, Figure 2 and Table 1 for guidance.

Table 1: Raspberry Pi Header Pinout

3.3V	1	2	5V
GPIO2 (SDA)	3	4	5V
GPIO3 (SCL)	5	6	GND
GPIO4 (1-wire)	7	8	GPIO14 (UART_TXD)
GND	9	10	GPIO15 (UART_RXD)
GPIO17	11	12	GPIO18 (PCM_CLK)
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10 (MOSI)	19	20	GND
GPIO9 (MISO)	21	22	GPIO25
GPIO11 (SCLK)	23	24	GPIO8 (CE0)
GND	25	26	GPIO7 (CE1)
ID_SD (EEPROM)	27	28	ID_SC (EEPROM)
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

(a) Put the MCP3008 on a breadboard and connect to your Pi:

i. Connect `3.3V` on the Pi to V_{DD} (pin16) on the MCP3008.
Also connect 3.3V to V_{REF} (pin15) on the MCP3008.

ii. Connect `GND` on the Pi to `AGND` (pin14) on the MCP3008.
Also connect GND to `DGND` (pin9) on the MCP3008.

iii. Connect `SCLK` on the Pi to `CLK` (pin13) on the MCP3008.

iv. Connect `MOSI` on the Pi to D_{IN} (pin11) on the MCP3008.

v. Connect `MISO` on the Pi to D_{OUT} (pin12) on the MCP3008.

vi. Connect `CE0` on the Pi to $\overline{CS}$ (pin10) on the MCP3008.

(b) For this first part, we will measure 0V on CH0 (pin1) and 3.3V on input CH1 (pin2). Hook up power and ground to those pins on the MCP3008.

(c) Enable Linux SPI support on your Pi. This is a bit confusing as they have changed the directions for this at least four times over the years.

- The current way to do this is to run the `sudo raspi-config` command, select "interface options", select "SPI" and then say "yes".
- If you get permissions errors you may need to do a `sudo addgroup YOURUSER spi` to allow accessing the device as a normal user.

(d) Modify the `test_spi.c` file to read values from the MCP3008 and print them to the screen. See the classnotes for more details.

- Open the `/dev/spidev0.0` file for read/write access.

- Use `ioctl` to set the mode to `SPI_MODE_0`
- In an infinite loop, read the value of CH0 and CH1 once a second and print the voltages to the screen.
  - As described in class use `ioctl` to transmit 3 bytes:
    * Be sure to clear out your transmit buffer with zeroes first (using memset or similar) or you may get errors!
    * You'll have to set the frequency. 100kHz will work.
    * Transmit three bytes:
      i. the first byte contains just the start bit,
      ii. the second byte has the single-ended mode bit, followed by 3 bits indicating the channel to read, followed by zeros
      iii. the third byte should be all zeros.
    * Receive 3 bytes. The first byte can be ignored, the bottom 2 bits of the second byte are bits 9 and 8, and the third byte is the bottom 8 bits of the result.
  - Put the 10 bits you receive into an integer, then use $V_{IN} = \frac{value \times V_{REF}}{1024}$ to convert to a floating-point voltage.
  - CH0 should be roughly 0V and CH1 should be 3.3V.

(e) Be sure to comment your code and check for errors!

3. **Hook up the TMP36 to the SPI device and measure Temperature** (3 points)

(a) Copy your test_spi.c file to display_temp.c

(b) Connect the TMP36 temperature probe to CH2.
Connect pin1 (3.3V) of the TMP36 to 3.3V
Connect pin2 (Vout) of the TMP36 to CH2 on the MCP3008
Connect pin3 (GND) of the TMP36 to ground
WARNING! the datasheet shows the pins from the *bottom* not the top. If you reverse the power/ground settings on the chip it will quickly heat up to 100+ degrees and will possibly be ruined! Follow the diagram in Figure 2 and you will be OK.

(c) Modify the code to print the current temperature as read by the probe, once a second. You can print degrees C or F as per your preference, but make sure the units are displayed.

(d) The temperature can be determined with the following equation:
$deg_C = (100 \times voltage) - 50$

(e) Also the following might be useful:
$deg_F = (deg_C \times \frac{9}{5}) + 32$
Beware of C floating point/integer conversion rules!

(f) Be sure to comment your code!

4. **Something Cool** (1 point)

   Copy your code to `temp_cool.c` and modify it to do something cool. Describe what you did in the README file.

   - Monitor the temperature, and after a while print the high/low temperatures recorded.
   - Monitor the temperature and print a message if a temp is exceeded (for example, print a message if someone touches the probe long enough to raise the temperature).
   - Hook up the analog discovery board and plot one SPI transaction on the MOSI/MISO/CS0/CLK lines. Send the plot in with your assignment.
   - Note: Homework 9 involves displaying the temperature on the LED display so while it would be cool to do that, you might as well wait a few weeks.

5. **Questions** (2 points)

   Answer the following in the README file:

   (a) What is one disadvantage of SPI compared to i2c?

   (b) What is one advantage SPI has over i2c?

   (c) You want to connect a second temperature probe to the MCP3008 from this homework, but with a 150-foot cable between the MCP3008 and the TMP36. (**NOTE**: the cable is between the MCP3008 and the TMP36, not between the Pi and the MCP3008). Would this work? Why or why not?

   (d) The MCP3008 datasheet says:
   "... the time between the end of the sample period and the time that all 10 data bits have been clocked out must not exceed 1.2 ms (effective clock frequency of 10 kHz). Failure to meet this criterion may introduce linearity errors into the conversion outside the rated specifications."
   Your boss asks you to talk an MCP3008 by bitbanging SPI via GPIO from regular Linux. How might this be a problem? Can you guarantee this timing if controlling the pins from userspace?

6. **Linux Fun** (1 point)

   Linux has very useful devices under /dev besides the i2c and spi nodes.

   (a) The first is `/dev/null`. What happens if you pipe a command to it?
   For example `ls > /dev/null`?

   (b) The next is `/dev/full`. What happens if you pipe a command to it?
   For example `ls > /dev/full`?

   (c) `/dev/zero` contains nothing but zeros. Why might that be useful?

   (d) What do you think `/dev/random` contains?

7. **Submitting your work**

   - Run `make submit` which will create a `hw7_submit.tar.gz` file containing `Makefile`, `README`, `test_temp.c`, `temp_cool.c` and `display_temp.c`.
   You can verify the contents with `tar -tzvf hw7_submit.tar.gz`
   - e-mail the `hw7_submit.tar.gz` file to me by the homework deadline. Be sure to send the proper file!