Lab 2

Interfacing Analog Sensors

Authors:

Alexandre Mesot, Michelle Mattille, Naveen Shamsudhin, Jonas Lussi and Prof. Bradley J. Nelson

Multiscale Robotics Lab, Institute of Robotics and Intelligent Systems

Date: 2022

Version: 2.4

Summary: This week we will interface an analog Hall-effect sensor to the Featherboard. A Hall-effect sensor is a transducer that varies its output voltage (V) in response to a magnetic field (milli-Tesla, mT). We will use it to measure the magnetic field generated by a stationary magnet. The sensor response itself is linear, i.e. the voltage readout is linearly related to the input magnetic field, but the magnetic field changes nonlinearly with the distance. In this 1-week lab module, we will learn:

- How to interface a magnetic field sensor with analog output to the microcontroller
- How to fit a parametric model to the experimentally obtained distance - magnetic field data in MATLAB

2.1 Background

2.1.1 Featherboard Input/Output

As stated before, the microprocessor on your computer accepts only digital inputs and digital outputs while the microcontroller on the Featherboard has 13 analog input channels (A0-A12).

2.1.1.1 Analog Inputs: Analog to Digital Converters (ADCs)

The analog input channels (Pins A0 to A12 - see Featherboard pinout diagram) can read in voltages in the range of 0V to a maximum of 3.3V. Any voltage higher than 3.3V will damage the internal circuitry. The analog-to-digital converter (ADC) on the Featherboard has a maximum resolution of 12-bits. The voltage measured is then assigned to a value between 0 and 4095 (= 2^{12} - 1), in which 0 V corresponds to 0, and 3.3 V corresponds to 4095. Any voltage between 0 V and 3.3 V will be given the corresponding value in between.

```
A digital value of:  4095 = 0xFFF represents  3.3 volts.

For any other digital reading, the equivalent voltage
V = 3.3*(digital reading)/4095 when the voltage range is 0 to 3.3V.

For example,
Digital value   Digital value in hex       Analog output voltage

3000              0xBB8                     2.418 V
1500              0x5DC                     1.209 V
```

Warning: The Feaetherboard I/O pins are 3.3V compliant. Higher voltages (like 5V) will damage the board. Please check the voltage ratings on the datasheet of the sensor, actuator or integrated circuit that you wish to connect to the Featherboard.

2.1.1.2 Using ADCs with different resolution and input range

Other embedded systems incorporate ADCs with different input voltage ranges (for example $\pm$ 5 V or $\pm$ 10 V) and different resolutions. Different sensors on the market have different output voltage ranges, and when designing your embedded system, its quite important to check whether the internal ADCs can tolerate the sensor output range. Consider an ADC which accepts $\pm$10 V and has 16 bit resolution. The maximum number that could be represented with a 16-bit word is 65535 (= 2^16 - 1). Since the 15th bit is reserved for sign representation, we can represent numbers from -32768 to +32767. The following illustrates how voltages are represented by integers.

```
A digital value of:  32767 = 0x7FFF represents  10 volts.
A digital value of: -32768 = 0x8000 represents -10 volts.

For any other digital reading, the equivalent voltage
V = 10.0*(digital reading)/32767 when the voltage range is +/- 10V.
For any other digital reading, the equivalent voltage
V = 5.0*(digital reading)/32767 when the voltage range is +/- 5V.

For example,
Digital value   Digital value in hex.       Analog output voltage

20000              0x4E20                      6.10 V
-10000              0xD8F0                     -3.06 V
```

It is important to use the appropriate data type here. Consider the following code section.

```
int16_t hex1 = 0xD8F0;
int32_t hex2 = 0xD8F0;
printf("Short integer 0xD8F0 in decimal is:%d\n",hex1);
printf("Integer 0xD8F0 in decimal is:%d\n",hex2);
```

The output is:

```
Short integer 0xD8F0 in decimal is: -10000
Integer 0xD8F0 in decimal is: 55536
```

The second output is not -10,000 as you would expect. Instead we get

```
13*16^3 + 8*16^2 + 15*16 + 0 = 55536.
```

You must realize that the data type `int16_t` represents a 16 bit integer where the 15th bit is the sign bit. A zero at the 15th bit indicates a positive number and a "1" at the 15th bit indicates a negative number. In the above case, 0xD8F0 had a "1" at the 15th bit, and is therefore a negative number. However, `int32_t` is a 32-bit integer where the 31st bit is the sign bit. Thus, an `int32_t` variable stores 0xD8F0 as 0x0000D8F0. This number has a zero in the sign bit and is not recognized as a negative number. So the implication of this is that we need to read in voltage data from the ADC into a `int16_t` variable, so that the presence of a sign bit at the 15th bit is properly recognized by the compiler.

Note: In this lab, we will only work on the Featherboard within a positive voltage range (0 to 3.3V) and should therefore not encounter this problem. We can use a data type with bit size larger than 12 bits and leave some bits empty.

Resolution:

The voltage resolution of the ADC is dependent on both the input voltage range and the bit size. For the ADC on the Featherboard,

```
Voltage range = 0 to 3.3 -> Resolution = (3.3 - 0)/4095 = 0.806 mV.
```

2.1.2 Magnetism

We will use a hall-effect sensor to measure the decay of the magnetic field of a longitudinally magnitized cylindrical magnet along its axis of symmetry. The following statements that can be considered true for a magnet

- the magnetic field B decreases rapidly with the distance z to the magnetic field source
- the magnetic field B is dependent on the magnetization of a magnet

There are different approaches to model the magnetic field B around one or several magnetic bodies and their results vary depending on their assumptions.

Attention: The forces between the large magnet and other large ferromagnetic objects (or other large magnets) are very strong when getting closer than a few centimeters.

The magnetic field B decay along the symmetry axis of an axially magnetised cylinder magnet can be modeled by

$$B(z) = \frac{B_r}{2}\left(\frac{D+z}{\sqrt{R^2+(D+z)^2}} - \frac{z}{\sqrt{R^2+z^2}}\right) \tag{2.1}$$

where B_r is the remanence field at the surface of the magnet $B(z=0)$, z is the distance from a pole face on the symmetry axis, D is the thickness of the magnet, and R is its radius, as illustrated in Fig. 2.1. (See Reference) The magnet used in this lab is disc-shaped with dimensions $D = 10$ mm and $R = 20$ mm.

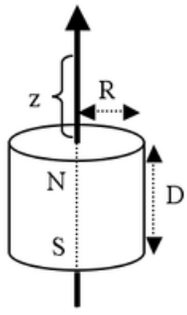

Figure 2.1: Magnetic field on the symmetry axis of an axially magnetised cylinder magnet.

Alternatively, a power-law function of the form,

$$B = f(\frac{1}{z^n}) \tag{2.2}$$

may be used to model the magnetic field decay.

We will find estimate the best fit parameters of the model curve for the distance - magnetic field relationship. The model can then be used to determine the field at any given distance from the magnet. The relationship between the voltage and the field on the other hand, is already known because the hall sensors have a linear sensitivity.

2.1.3 The Linearizing Function

Figure 2.2 shows the experimental setup and Figure 2.3 (a) shows a typical distance - magnetic field measurement. As expected the relationship is not linear. This curve will vary slightly from setup to setup.

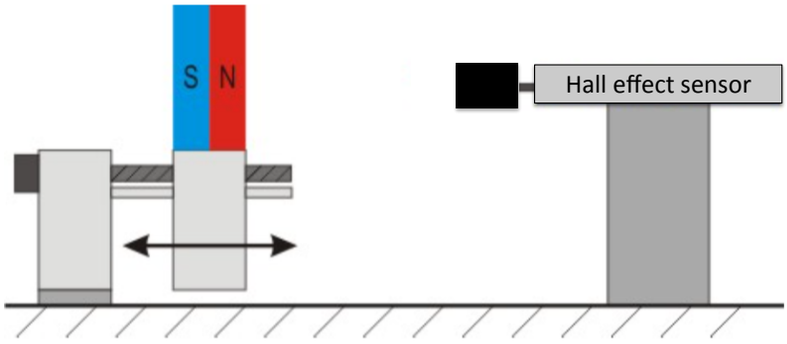

Figure 2.2: Experimental Setup for Lab02. A large magnet is placed on a movable stage and a Hall sensor is placed at a distance.

We want to find a conversion function $f(z)$ from distance from magnet to the magnetic field. Before we can start to determine a conversion function, it is useful to examine the original data. In a first step, we look at the data and assume that the field drops similar to a power law function $1/z^n$ with z being the distance. So lets take a look at a plot of the magnetic field B versus $1/z^n$, in this case with $n = 0.1$, as shown in Figure 2.3 (b). Note that this plot is almost linear, so we are able to use a linear approximation of the form: $y = mx + b$ with y in units of μT and x is units of mm^{-1}:

$$y = B \tag{2.3}$$
$$x = 1/z^n \tag{2.4}$$

Determining m and b is a fairly simple operation and can be performed by a `Least Squares Fitting` technique and the result is shown in Figure 2.3 (c) (fitted with the parameters $m = 176.9$ and $b = -118.9$).

$$B = 176.9x - 118.9 = 176.9(1/z^n) - 118.9 \tag{2.5}$$

Now we substitute x back to $1/z^n$ and plot the nonlinear function. Figure 2.3 (d) shows the fitted curve plotted against z together with the original graph and this method of approximation matches the original data quite well.

We can also use the other magnetic model to estimate the magnetic remanence B_r of the cylindrical magnet. To do this, we can make use of a `Non-linear Least Squares` solver in MATLAB. First we plot the magnetic field B against a function

$$f(D,R,z) = \frac{D+z}{\sqrt{R^2+(D+z)^2}} - \frac{z}{\sqrt{R^2+z^2}}. \tag{2.6}$$

This gives us a relationship in the form of

$$B = \frac{B_r}{2} f(D,R,z) \tag{2.7}$$

We can use the MATLAB function `lsqcurvefit` to find a value for B_r. Figure 2.4 shows the measured data for the magnetic field and the fitted data according to the model, in this case with $B_r = 175.6$.

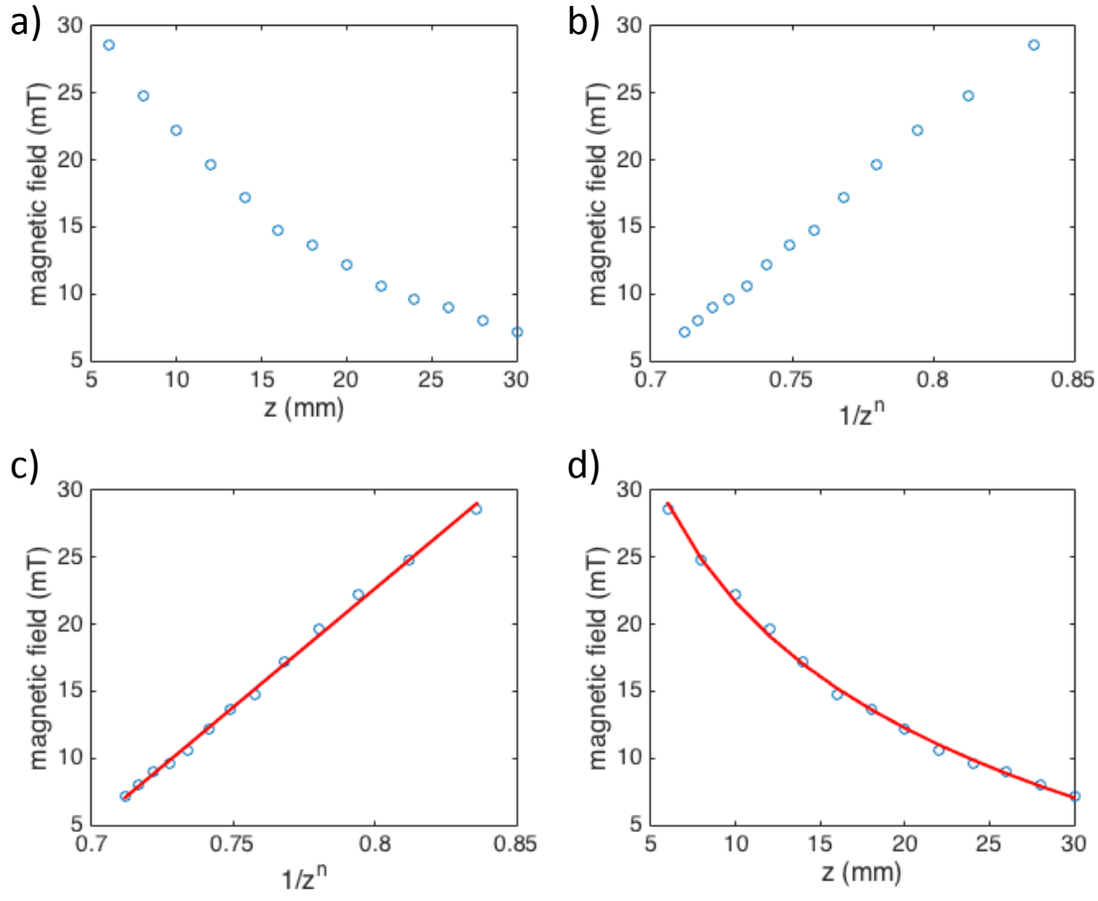

Figure 2.3: Hall-Effect Sensor Sample Data and Power-law Modeling

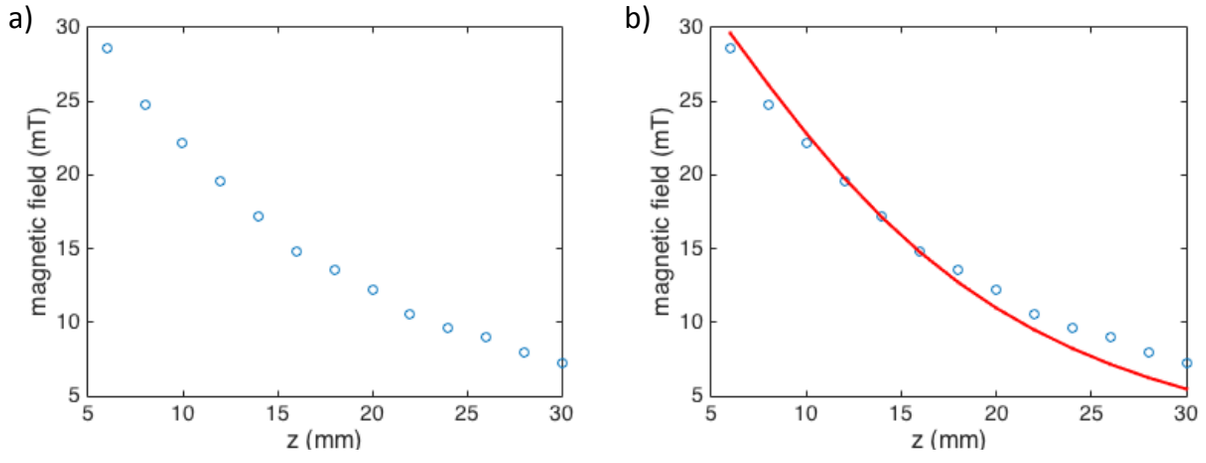

Figure 2.4: Nonlinear Least Square fitting of Sample Data.

2.2 Prelab Procedure

Note: The Prelab quiz on Moodle must be done before reporting to the lab. The submission deadlines are on Moodle. Late submissions will not be corrected. Each group member has to complete the quiz on their

own, however you are allowed to work with students in your group to solve the quiz (and we absolutely recommend you to do so).

2.3 Lab procedure

2.3.1 Open project files

1. Create a folder labeled `lab02` under the `/home/ubuntu/` folder for the skeleton files that you can access in Moodle.
2. Go to the Arduino IDE and verify and upload the code you prepared in the prelab to the Arduino. (Optional: To know more about the build process, see https://arduino.github.io/arduino-cli/0.21/sketch-build-process/)
3. Connect the cables of the hall sensor to the Featherboard in the order described next. When connecting any electronic device, it is recommended to first connect the electrical ground (GND). You will find three cables attached to the Hall Sensor PCB. First we power the hall sensor by connecting the black cable to ground (GND) followed by the red cable to the voltage supply (3.3V). The output voltage of the hall sensor is available on the blue cable which should be connected to ADC A0 (GPIO Pin 26) of the microcontroller.
4. To compile and run the program, you can type

```
irm@ubuntu:~/lab02$ make all
irm@ubuntu:~/lab02$ ./hall_daq
```

Note 1: Your program is empty, so nothing should happen right now. By pressing `Ctrl` + `C` in the console you can abort a running program. You will write the code in the next tasks.

Note 2: To clean your directory type:

```
irm@ubuntu:~/lab02$ make clean
```

2.3.2 Reading analog voltage inputs (Postab Q1)

In this part, you will implement your own functions to read magnetic field data from the Hall sensor.

1. You have already written the arduino code in the prelab assigment. Now we need to write a main function that runs on the microprocessor and communicates with the microcontroller. For this, have a look at the skeleton given to you in `hall_daq.c`.
2. First you need to initialize the serial port. Use a baud rate of 115200. The baud rate is the bit rate per second at which we transfer information. For example, a baud rate of 9600 means that we are transferring information through the serial connection at a maximum of 9600 bits per second.
3. Initialize a char buffer, in which the string value can be written. The buffer should hold 4 bytes. (A buffer is an array, which is used to temporally store values.)
4. Send a message (for example a character) to the featherboard, to let it know that it needs to send a value. (Hint: use the function `serialport_write()`). The featherboard will receive this character that you send through the serial port. According to the code implemented in the Prelab the featherboard will then send a response through the serial port if the character you are sending matches the one you defined in the `lab02.ino` file.
5. Read the value that was sent from the featherboard through the serial port (using `serialport_read()`) and convert the value in your buffer to an integer value. (Hint: use the C library function `int atoi(const char *str)` to convert the char to an integer value).

6. Now you can convert your sensor value to a voltage (make sure to consider the quiescent voltage output). The sensor value will be between 0 and 4095, the maximum voltage output is 3.3V. (**Hint:** Consider the sensitivity)

7. In a last step print the measured voltage value to the terminal. Finally close the serial port.

8. You can now test your function by typing:

```
irm@ubuntu:~/lab02$ make
irm@ubuntu:~/lab02$ ./hall_daq
```

2.3.3 Sensor noise analysis - (Postlab Q2)

1. Until now we have sampled one voltage value from the sensor. Run your code in a loop for 50 times and sample 50 voltage values. You can implement a for-loop or a while-loop to do this. Instead of printing the values to the console, print the output into a file, such as `voltage_data.txt` . The simplest way is to redirect the console output to file by running the following command in the console:

```
irm@ubuntu:~/lab02$ ./hall_daq > voltage_data.txt
```

2. Open MATLAB for data analysis. In MATLAB, load the data from the file `voltage_data.txt` into a `Numeric Matrix` and plot it. You can use the MATLAB command `importdata` and `plot` to plot the data. You can visualize now that the sensor output is quite noisy. Calculate and write down the average and standard deviation of this dataset.

3. Now write a microcontroller (Arduino) code snippet to read in 20 consecutive values of the sensor voltage instead of a sigle one like before, average them into a single value and send it through the serial port. Output 50 such averaged values to a file as you did before. Import them into MATLAB, and plot the 50 datapoints. Calculate and write down the average and standard deviation. Finally, read in 200 consecutive voltage values, average them on the microcontroller (Arduino), and save 50 such values into a data file. Again, plot your results in MATLAB and compare the average and standard deviation. Explain your observations and conclusions.

2.3.4 Characterizing magnetic field decay of permanent magnet - (Postlab Q3)

We have successfully read the voltage value from the Hall sensor. Now, we shall convert the voltage into a magnetic field measurement. Figure 2.2 shows the setup. A large magnet is placed on a movable stage and the Hall sensor is placed at a distance. Until now, we have taken measurements with the Hall sensor far away from the magnet.

1. Take the calculations you did to compute the voltage in Section 2.3.2.6 (**Postlab Q1**) and implement it as part of the function `hall_sensor_get_field()` in the file `hall_sensor.c` according to the description in the header file and call it in the main function in the file `hall_daq.c`. This way you have a cleaner Main file without the calculations. Do not forget to print the measured field.

2. Run the program and move the magnet towards the Hall sensor and print the field values to the terminal - you should see an increase in magnetic field as you move the magnet closer to the Hall sensor.

3. The function `hall_sensor_get_field()` exists to get field measurements at several concrete distances from the magnet. In the main program, comment the for-loop as we don't need it anymore for this part of the task. We only need to find the magnetic field at specific distances. For this, take field measurements (take an average of 50 values) at distances starting from approximately 40 mm to 15 mm from the magnet's surface (use a reasonable step size, *e.g.* 2 mm). When you run the test program, redirect the results (distance - field) to a file `field_distance.txt`. If you want to display information on the console as well as writing the output to a file, you can use `stderr` to output to the console and `stdout` for the content that you want to be redirected to the `field_distance.txt` file. Use tabs between the distance and force value to store in the .txt-file such that MATLAB can recognize the 2 rows of data.

```
/* the following text will appear on the console */
fprintf(stderr,"Please move the magnet to position %d and press ENTER \n", pos)
/* (get sensor signal and convert to field) */
/* the following text can be redirected to field_distance.txt */
fprintf(stdout,"%d\t%.3f", pos, field)
```

4. Plot the field vs. the distance in MATLAB. It should look similar to the Figure 2.4. Use the MATLAB function `lsqcurvefit` to fit a line to the measured values to estimate the magnet remanence B_r, as outlined in Equation 2.6 and illustrated in Fig. 2.4. Again, find the optimal B_r to fit your data, that minimizes the total error. Take into account that the hall sensor saturates at a certain field. This can lead to a flat curve close to the magnet. If that is the case with your data make sure to disregard the values in the saturated region.

   (Hint: If you don't know how to use the function `lsqcurvefit`, type `doc lsqcurvefit` into the Command Window in MATLAB, read the explanation of the function and take a look at the example.)

2.4 Postlab and lab report

1. Record a video of your working system (**Postlab Q3**) and how the values increase on the terminal as you move the magnet towards the Hall sensor. Share it with your assistant on slack.
2. Upload a zip file and a pdf with your solutions to Moodle. The zip file should contain the following files:
   - The `hall_sensor.c`, `hall_daq.c` and `lab02.ino` files from **Postlab Q1 - Q3** and all the files necessary to compile and run them.
   - The MATLAB code you wrote to create the plots mentioned below.
3. Also hand in the answers to the Matlab Questions **Q2 & Q3** as a PDF-file. It should contain the following plots:
   - The three MATLAB plots of the noise analysis measured in **Postlab Q2** and the mean and standard deviation calculated for these signals.
   - The MATLAB plot of the magnetic field decay measured in **Postlab Q3** with the nonlinear least square approximation.
4. Please upload your solution in time (before next lab session), late submissions will not be corrected.
5. Come prepared with the Prelab Procedure for Lab 03 - Analog Filtering.

$$H(s) = \frac{\omega_c}{s + \omega_c} = \frac{1}{1 + s/\omega_c} = \frac{1}{1 + \frac{1}{200\pi} s}$$

$$\omega_c = 2\pi \cdot 100$$