In Lee and Seshia, 2nd Edition 22/22

2. The dynamic range of human hearing is approximately 100 decibels. Assume that the smallest difference in sound levels that humans can effectively discern is a sound pressure of about 20 μPa (micropascals).

(a) Assuming a dynamic range of 100 decibels, what is the sound pressure of the loudest sound that humans can effectively discriminate?

Let 20 µPa represent the reference sound level, 0 dB_{SPL}.
Notice that a Pascal is a pressure level, not a power level. Thus the formula for converting pressure levels to decibels is:

$$(Pressure\ in\ dB_{SPL}) = 20 \log_{10}\left(\frac{Pressure\ in\ \text{Pa}}{Refernce\ Pressure\ in\ \text{Pa}}\right)$$

Dividing both sides of the above by 20 and then raising 10 to the power of each side shows that 100 dB_{SPL} can be converted to an absolute sound level as follows:

4/4

$$10^{\frac{100\ \text{db}_{SPL}}{20}} = \frac{Pressure\ in\ \text{Pa}}{20\ \mu\text{Pa}}$$

$$10^5 (20\ \mu\text{Pa}) = Pressure\ in\ \text{Pa}$$

$$\boxed{Pressure = 2\ Pa}$$

(b) Assume a perfect microphone with a range that matches the human hearing range. What is the minimum number of bits that an ADC should have to match the dynamic range of human hearing?

The needed dynamic range is $(max - min)/(min)$

$$\frac{2\ Pa - 20\ \mu PA}{20\ \mu Pa} \simeq 10^5 \text{ or } 100\ \text{dB}$$

To convert this to bits, take the base 2 logarithm

$$\log_2(10^5) = 16.61$$

The microphone will need 17 bits, minimum.

In Lee and Seshia, 2nd Edition

3. The following questions are about how to determine the function

$$f: (L, H) \to \{0, \dots, 2^B - 1\}$$

for an accelerometer, which given a proper acceleration x yields a digital number $f(x)$. We will assume that x has units of "g's," where 1g is the acceleration of gravity, approximately $g = 9.8$ meters/second^2.

(a) Let the bias $b \in [0, \dots, 2^B - 1\}$ be the output of the ADC when the accelerometer measures no proper acceleration. How can you measure b?

Put the sensor (the accelerometer) into a free fall. Then the proper acceleration (the sum of gravitation plus actual acceleration) will be zero. The digital output will then be the bias. $f(\text{free fall}) = b$

(b) Let $a \in \{0, \dots, 2^B - 1\}$ be the difference in output of the ADC when the accelerometer measures 0g and 1 g of acceleration. This is the ADC conversion of the sensitivity of the accelerometer. How can you measure a?

Put the sensor (the accelerometer) in a stationary position facing upward. Then the proper acceleration (the sum of gravitation plus actual acceleration) will be 1 g. The digital output will then be the bias plus a. The sensitivity will be $f(\text{stationary and vertical}) - f(\text{free fall}) = a$

(c) Suppose you have measurements of a and b from parts (3b) and (3a). Give an affine function model for the accelerometer, assuming the proper acceleration is x in units of g's. Discuss how accurate this model is.

Given an actual acceleration, x, the model should predict the digital output of the acc.

$$f(x) = ax + b$$

(d) Given a measurement $f(x)$ (under the affine model), find x, the proper acceleration in g's.

$$x = \frac{f(x) - b}{a}$$

(e) The process of determining a and b by measurement is called calibration of the sensor. Discuss why it might be useful to individually calibrate each particular accelerometer, rather than assume fixed calibration parameters a and b for a collection of accelerometers.

Some manufacturing variability can be expected from sensor to sensor. By calibrating each sensor individually this manufacturing variability can be accounted for and removed from the uncertainty of the measurement. On the other hand, other variations such as environmental temperature and the age of the sensor can also contribute to measurement uncertainty. If these are greater than the manufacturing variability, then in makes sense to assume fixed calibration parameters and accept the slightly larger measurement uncertainty that results.

(f) Suppose you have an ideal 8-bit digital accelerometer that produces the value $f(x) = 128$ when the proper acceleration is 0 g, value $f(x) = 1$ when the proper acceleration is 3 g to the right, and value $f(x) = 255$ when the proper acceleration is 3 g to the left. Find the sensitivity a and bias b. What is the dynamic range (in decibels) of this accelerometer? Assume the accelerometer never yields $f(x) = 0$.

Given in the problem statement: $f(0) = 128$ and from the model $f(0) = a(0) + b$
Set these equal and solve for b.

$$b = 128$$

Assume that acceleration to the left is given a positive sign and acceleration to the right is negative.
Given in the problem statement: $f(3\text{ g}) = 255$ and from the model, $f(3\text{ g}) = a(3\text{ g}) + 255$

$$a = \frac{255}{3} = 42.\bar{3}/\text{g}$$

Dynamic range is $20 \log_{10}\left(\frac{255-1}{1}\right) = 20 \log_{10}((3\text{ g} - (-3\text{ g}))(42.\bar{3}\ /\ \text{g})) = 48.1$ dB ←

Draw a schematic to show to connect a hex-pad to an Arduino and be able to read the key-presses. Assume that the `loop()` program will repeat rapidly (at least 100 times a second). Specify how to connect the keypad to specific GPIO pins of the Arduino, how to set up each GPIO pin, and how to collect the key-presses in the loop procedure. Here is a typical hex keypad:

The keypad requires eight connections to digital I/O pins on the Arduino Uno. I have chosen D13 down to D6, but any eight pins will do.

The rows of the keypad, which are connected to Arduino Uno pins D13 down to D10, should be configured as outputs. All of them should be driving HIGH except one. The one that is driven LOW should be considered the *active row*.

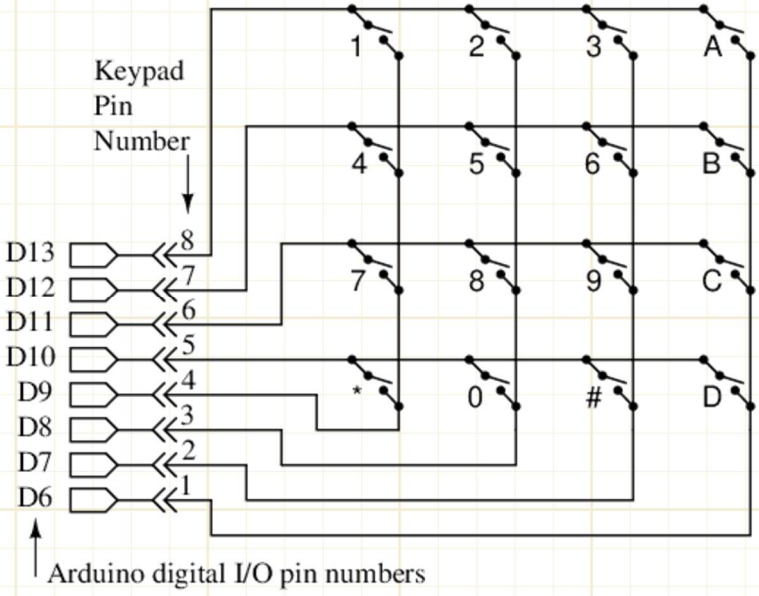
The columns of the keypad, which are connected to Arduino Uno pins D9 down to D6 should be configured as inputs with pull-up resistors enabled. (`pinMode(pin, INPUT_PULLUP);`) A digital read of these four inputs will reveal any depressed button on the active row of the keypad. Here is how that happens. First of all, assume only one button on the keypad is pressed at any one time. If more than one button happens to be pressed, allow that an incorrect result may be returned.

6/6

Start by reading pin D9. The only way that it can return a LOW state is if it is connected to the active row, which would signal that either button 1, 4, 7, or * was depressed, depending on which row was active.
If it was indeed low, return that keystroke to the calling program but also lock out any further reading of inputs for a short time, say 20 ms to let switch bouncing settle out. The read the same input again. If it is still LOW, the switch is still down. Wait for it to Return to HIGH before reporting another keystroke.

However, if reading pin D9 resulted in HIGH, move on and read bit D8, then D7, then D6, then switch the active row and continue scanning D9 down to D6 until a LOW is detected.

Whenever a low is detected return the correct key symbol to the calling program and go thought the debounce and key-release procedure as described two paragraphs above.



The Arduino IDE provides a library of functions to do these tasks. The library is sensibly titled, "Keypad." The library is written in C++. It provides an object called Keypad with methods to configure the pins and read the keypad. See the link below for details.

https://playground.arduino.cc/Code/Keypad/