Assignment: Random walk

1) Using the robot's sensors for input, write a function that returns a random integer in the range 0 to 9,999,999. This function should not merely return one of the inputs, but must do some transformation of the input data, such that results are approximately uniformly distributed over the desired range—that is, that any particular number has about as much likelihood of being returned as any other. (Note: Achieving truly uniform, truly random results are well beyond this course. However, even a fairly simple transformation based on rapidly-changing input is often good enough.)
2) Using your random-number generator, write a function that takes a duration (in seconds) as input and has the robot do a random walk with each step taking 0.5 seconds—that is, randomly select whether to move forward, back, turn left, or turn right. Keep the speed at 0.2 or 0.1.
3) Turn in your source code, and 3 sheets of paper showing your robot's 10-second random walk on 3 separate runs.
4) bonus: Play a short random tone each time the robot turns.
5) Now do a really random walk: After randomly selecting what your robot will do, randomly select a duration from 0.1 to 5 seconds to execute that maneuver. You may increase the speed if you wish, or allow the speed to be randomly chosen as well. (To do this step, you may need to take your randomly-generated integer and convert it to a random floating-point number in a specified range. There are other ways to solve this step as well.)
6) Now, using the IR sensors, have your robot detect if it's about to back into something if it's moving backwards; before colliding, it should stop and play a series of 2 distinct tones (to distinguish between a 'normal' end of step beep and a collision-avoidance stop).
7) The obstacle detectors on the front of the fluke card return data in a much wider range than the IR detectors in the back. We will add the ability to detect a front-end collision after completing the next assignment.

Assignment: signal processing

The obstacle detector on the front of the Fluke dongle is a fairly 'noisy' signal; that is,there is a large amount of variation from second to second, independent of what the sensor is actually detecting. Mathematically, we model each reading as a signal of some unknown strength, plus a random amount of noise. We must process what we receive to reduce the effect of the noise so we can detect the underlying signal. For example, if the noise has a normal (bell-shaped) distribution, then the errors will tend to make the signal appear too high as often as too low. This means that if we add them together they will tend to cancel each other out. This isn't perfect—there may be some left-over (residual) error—but it at least reduces the noise's effect. Therefore, the sum (or average) of several signals can be more reliable than an individual reading.

Write a function that keeps a running average of the 5 most recent readings from the center obstacle detector.

Write a simple loop that prints 20 consecutive readings from the center sensor; then write a function that prints 20 consecutive 'average-of-5' estimates. What do you observe?

Write a function that causes your robot to move forward until the average of the 5 most recent readings exceeds a threshold (which will vary by robot) when an obstacle is about 2 feet away. (NOTE: This is approximate. Results may vary depending on robot speed, ambient lighting, battery strength, and other factors.) A signal input of 1200 is a reasonable starting value, but you will have to experiment to find the right value for your robot, and you may find that there is still fairly significant variation in its detection ability. Add this functionality to your “Really Random Walk” function from the previous assignment so the robot can also detect a front collision. Add the following functionality:

If the robot completes a step normally, play a random tone in the range 400-1200 Hz for 0.5 seconds. If the robot stops because it detects an obstacle behind it, play a 2-tone sequence (using tones you specify) to indicate the condition; and a different 2-tone sequence if the front obstacle-detection caused the robot to stop.