Jynx Boyne BME 301

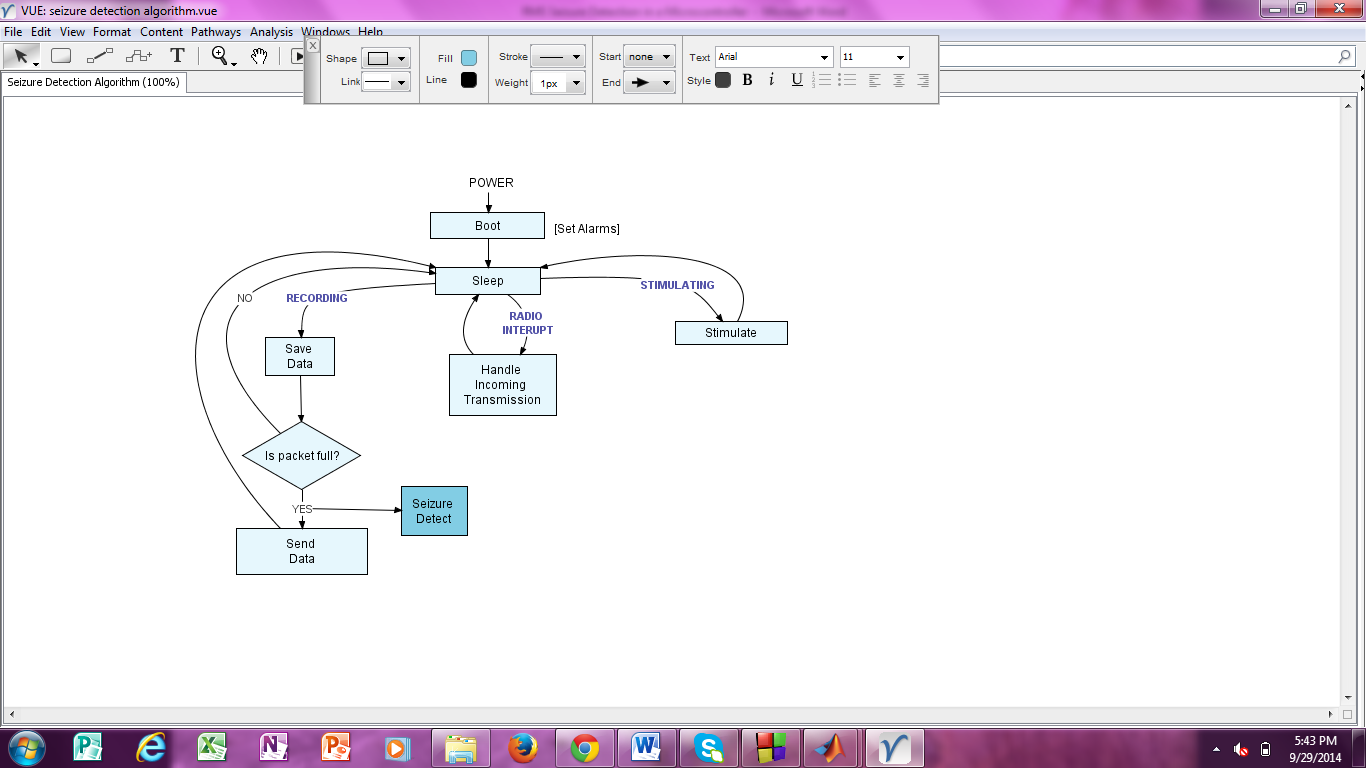
Amanda Goetze 10/3/14

Kaitlyn Jarry

Shadman Jubaer

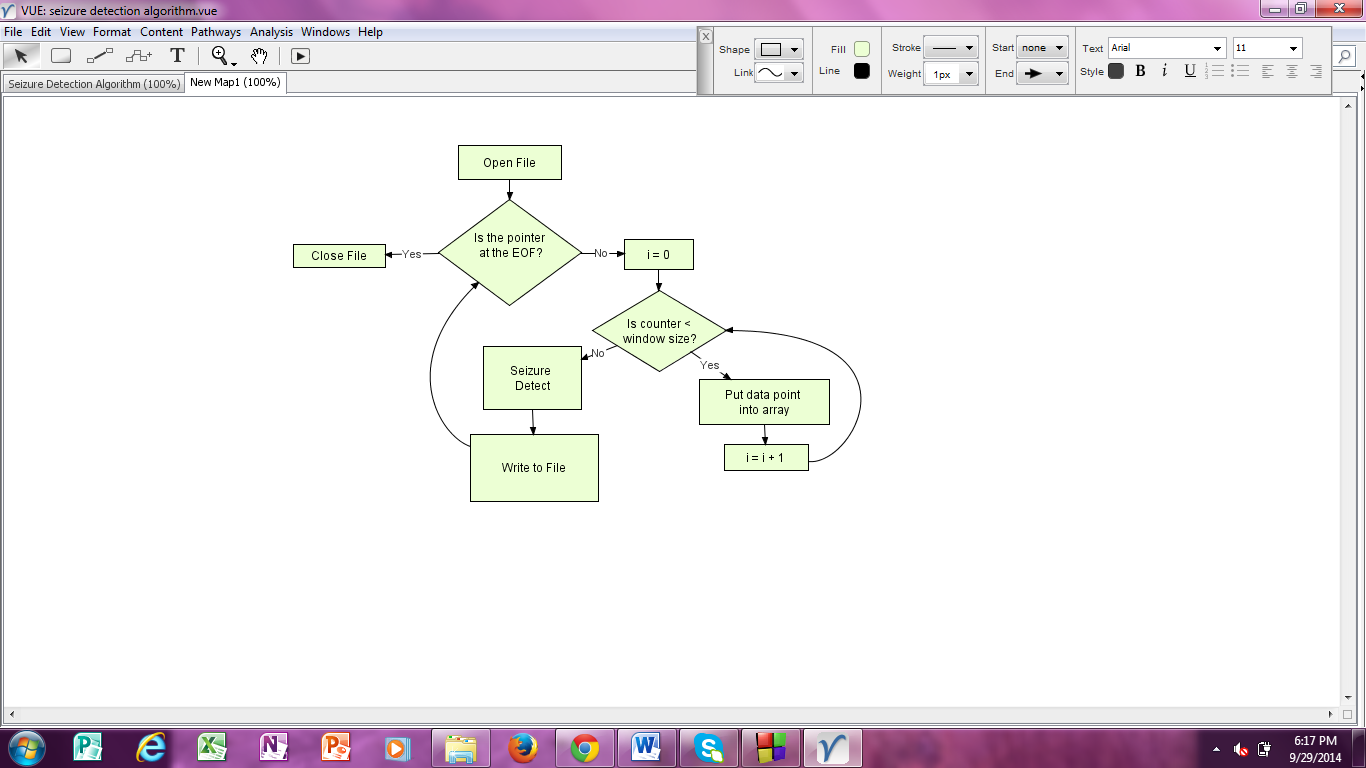
Peter Russel

RMS Seizure Detection in a Microcontroller

 Detecting seizures in MATLAB without limitations on the length or complexity of code is relatively easy, however when detecting seizures in a microcontroller there are other design constraints that need to be considered. Physical constraints on the microcontroller translate to limitations in the amount of memory that can be stored in the microcontroller at a given time. For this particular application, the memory used needs to be less than one kilobyte of raw data. This means that the microcontroller can only store a few hundred data points at a time before it needs to send a packet of information wirelessly to a more powerful computer that can store all of the data. Another important aspect to consider that will affect how the algorithm works is at what point the data will be analyzed in the overall sequence of steps that the microcontroller iterates through. There are two ways of accomplishing seizure detection. The data can be checked for seizures after every data point is collected, or the data can be analyzed after a full packet is stored in the microcontroller. There are advantages and disadvantages to both methods. In this algorithm, the method of analyzing the data after a full packet is stored is used. This happens at a specific point in the processes carried out by the microcontroller and can be seen in Figure 1.

*Figure 1. Algorithm of processes of microcontroller*

* Advantage to placing the seizure detection function at this point in the flow diagram

 The algorithm first builds an array by getting one data point at a time until the window size is reached. The window size used was 512. This number was chosen because it is a power of base 2, (29), which makes it easier to bit shift. Bit shifting is an efficient method to divide numbers by simply moving the decimal point in binary numbers. Once the window size is reached, the seizure detection function is ran and the output is written to a csv file. When the pointer reaches the end of the file, all of the data points have been analyzed and the file is closed. A diagram of this process is shown in Figure 2.

*Figure 2. File input and output algorithm*

* How Average was calculated

After an appropriate number of data points are collected, they are passed into the seizure detect function. In this function the average of values in the window are calculated. The sum is calculated using a while loop to run through the array and sum the next value in the array with a previous sum. This is the most intensive part of the function, and most optimization efforts focused on reducing the number of data points to use in calculating the sum while maintaining a high level of specificity and sensitivity. Since the window size is a power of two, dividing to calculate the average is easily done with a bitshift operator and costs minimal processing power. The average value is then compared to a threshold value. If the value is within an accepted range above the threshold, but is still below a second noise threshold, the function determines a seizure has been detected and returns a one. If the value exceeds the noise threshold, or is below the seizure threshold the function determines no seizure was detected and returns a zero.

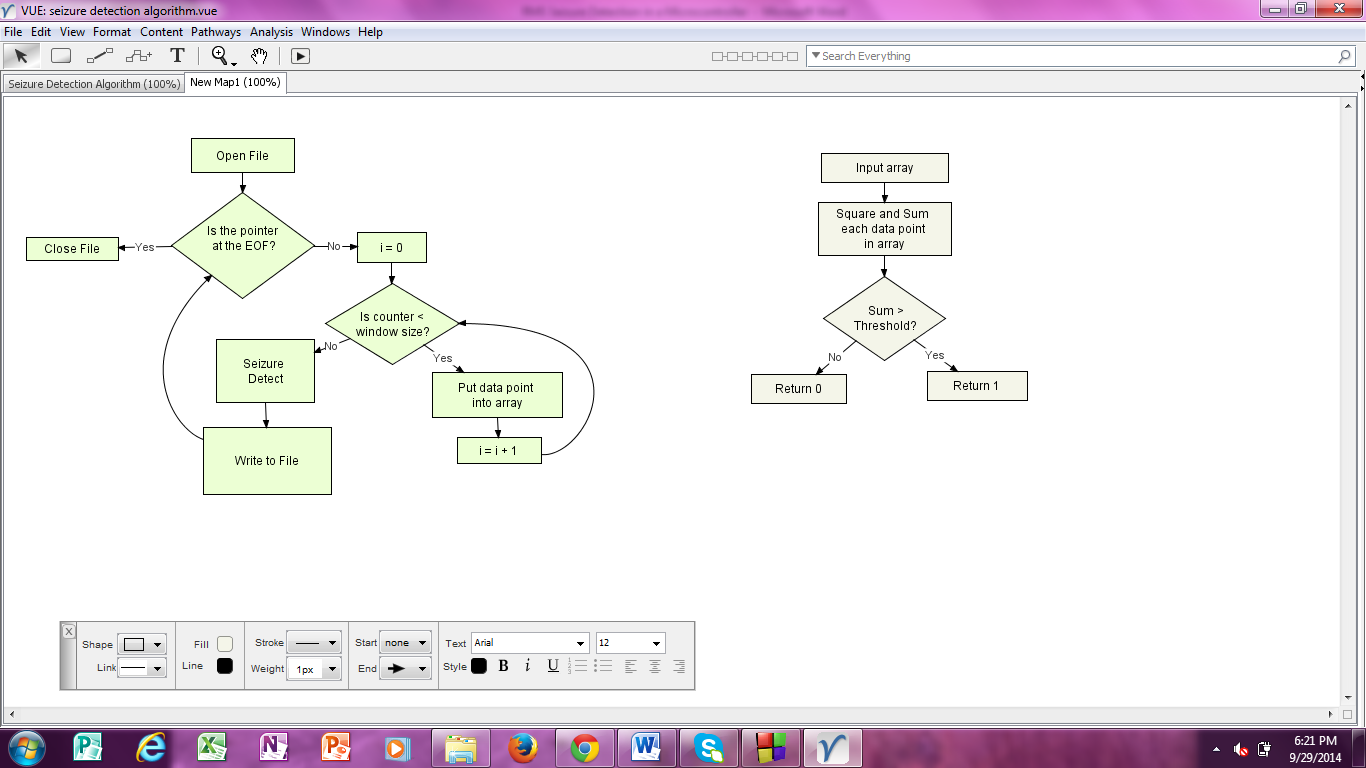
* How the threshold value was determined/ output plot -Jynx
* Sensitivity and Specificity

One of the main constraints while designing the code revolved around creating the most efficient code possible. Efficiency called for a high degree of specificity and sensitivity while reducing the intensiveness of the code required to detect a seizure. Efficiency was defined as:

Sensitivity is calculated using the following equation:

Specificity is calculated using the following equation:

The minimum requirement the team strove towards while developing the code was to eliminate any false negatives and detect every seizure. An input array size of 512 data points was chosen to calculate the average value of the EEG data. This achieved a total of zero false positives, and a total of zero false negatives. The efficiency of the program was calculated to be . The team did try switching the window size from 512 data points to 2 data points. However, such a small window size introduced a lot of noise into the system, and it proved difficult to find a threshold value that would be able to detect seizures while ignoring the noise. For this reason, the team chose to use a window size of 512, which despite having a relatively high RAM and clock usage, seems to provide the most accurate data.



*Figure 3. Seizure Detection Algorithm*