SAN FRANCISCO STATE UNIVERSITY

SCHOOL OF ENGINEERING

Final Project Report: Quake Detector

ENGR 844

Fall 2021

By

Amir Modan, #918234621

amodan1@mail.sfsu.edu

Date report submitted: Dec. 18, 2020

Table of Contents

Abstract

Throughout this project, we will design an Embedded System capable of detecting Seismic Activity using an Accelerometer. The system will display on an LCD screen how much Seismic Activity is currently being experienced and, if that amount were to surpass a set threshold, the system will classify this as a Quake and warn the user. An overheating prevention mechanism will also be implemented which will reduce power consumption of the system by sending the microcontroller into its Hibernation mode whenever the sampled internal temperature is past the safe threshold. Both the threshold for Seismic Activity and Temperature can be modified by the user via a Rotary Encoder.

Introduction: For this project, we will design a real-time system capable of detecting quakes. Quakes are characterized by shaking or trembling which, when severe, can result in destruction of property or even loss of life. While our Quake Detector will be designed to simply warn the user whenever such an event is occurring, it can easily be embedded into a larger system, such as a power grid, and be used to deactivate dangerous components during a catastrophic event.

Hardware Design:

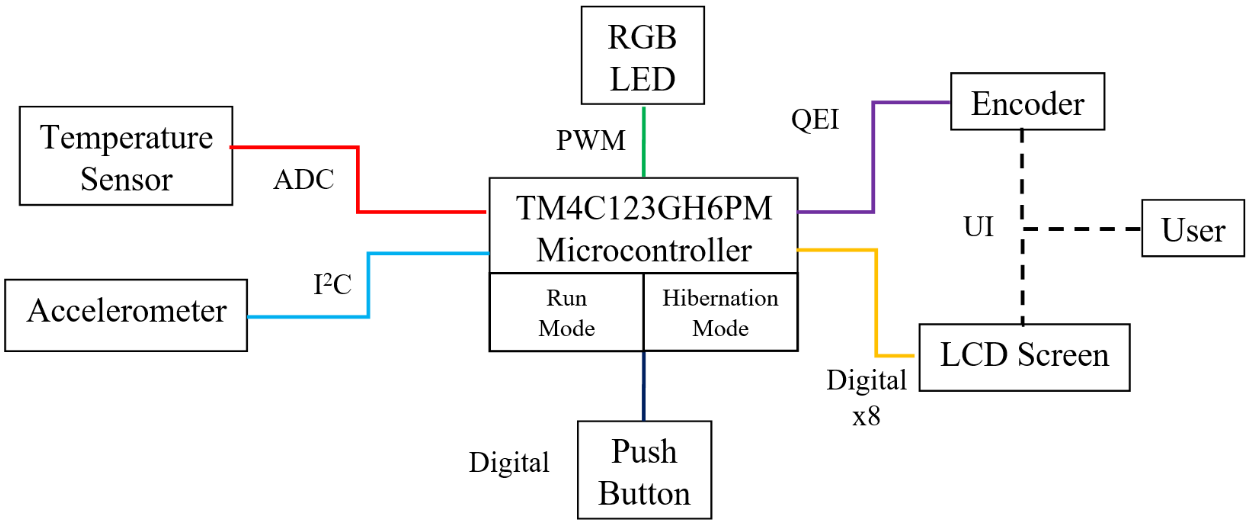

Figure 1 – System Architecture

Several different components will be used to perform this task while interfacing effectively with the user as shown in Figure 1. One major component will be the ADXL345 Accelerometer which is capable of detecting force in the x, y, and z axis. Because earthquakes tend to cause sudden changes in force for a period of time, the Accelerometer will be responsible for detecting the quakes. The ADXL345 Accelerometer uses the Inter-Integrated Circuit (I^2C) Protocol to send data, so we will take full advantage of the Tiva Board's capabilities to deal with this communication protocol with no support from external libraries.

We will include a module that will determine whether the system is overheating using the Tiva Board's on-chip temperature sensor. To read the device's temperature, we will have to sample analog readings from a Temperature Sensor and use the Tiva Board's ADC capabilities to convert the readings into digital data that the microcontroller can understand. For this project, we will convert the digital data into Celsius. If an extremely high temperature is detected, we will reduce power consumption by entering the microcontroller's hibernation state to ensure that the system continues to function correctly.

We will also include an interface that the user can interact with to make changes to some of the system's attributes, such as adjusting the sensitivities of the two features mentioned above. This will give the user flexibility over which application they need to use the system for. We will use an LCD1602 screen and a Rotary Encoder to provide output and input, respectively. For the LCD screen, we used pre-defined libraries to print out strings such as the warning that is displayed when a threshold is passed. The LCD screen will also display the current temperature reading or quake intensity, depending on which attribute the user wishes to view.

The Encoder is used to adjust the threshold of either the seismic activity that will define a quake, or the temperature at which the microcontroller is overheating. We implemented the Encoder by using the Tiva's Quadrature Encoder Interface (QEI). A button on the Encoder will be used to switch the view between the Seismic Activity and Temperature modes using an Edge-Triggered Interrupt. In Seismic Activity mode, the current value of Seismic Activity will be displayed on the LCD screen as well as how it compares to the set threshold for Seismic Activity. By rotating the Encoder shaft, the quake threshold will increment or decrement by 10, depending on whether the shaft is rotated clockwise or counterclockwise. Similarly if Temperature mode is selected, the current temperature will be displayed as well as how it compares to its threshold. In this mode, rotating the Encoder shaft will increment/decrement the temperature threshold by 1^{o}C. Regardless of which mode is chosen, the LCD screen will always display a warning if its corresponding threshold is passed.

While it is possible to simply display both the current Seismic Activity and the current Temperature on the 2x16 LCD screen, doing so may confuse the user as to which threshold they are currently adjusting with the Encoder. For this reason, only the attribute which is being adjusted will be displayed by the LCD screen while the other will be indicated by an RGB LED mounted on the Tiva board. The RGB LED actually consists of three separate LEDs each controlled by their own GPIO pin on the microcontroller. A simple approach would be to assign "sub-thresholds" to each mode, where a low value is indicated by a blue LED, a medium value by a green LED, and a high value by a red LED. However, this does not give specific information to the user as there are only 7 possible colors that can be produced. Instead, we can utilize Pulse Width Modulation to produce hundreds of different colors.

When operating for long periods of time, the microcontroller may be prone to overheating. Damage incurred from overheating is typically irreversible, so we as Embedded Systems Engineers should take preventative precautions to detect when it is happening. For this reason, we have built in a mechanism to read the internal temperature of the microcontroller and transition the microcontroller into Hibernation mode if the temperature is past the safe threshold. In Hibernation mode, the microcontroller will significantly reduce power consumption by halting all clocks except for that which drives the hibernation module. In doing so, the heat generated by the microcontroller will decrease, and the microcontroller would have a chance to cool down. Note that in this mode, changes in motion, temperature, or encoder position will no longer have an effect on the system as these peripherals are now disabled to reduce power consumption. To transition the microcontroller back into its normal mode of operation, the user simply has to press and hold SW2 on the Tiva board.

Software Design:

We have decided to tackle this project module-by-module. This means that we have written and tested separate C programs for each hardware module, then integrated them all into one software system that behaves according to our design plan.

The main component of our system, the ADXL345 Accelerometer, was programmed using the I^2C (Inter-Integrated Circuit) serial interface. I^2C allows a master to send data to and from a slave using two bi-directional lines, Serial Data Line and Serial Clock Line, each assigned to their own pin on the microcontroller. Three functions have been written that allows a master to read from a slave address, write to a slave address, and conjoin two 8-bit values into one 16-bit value. This conjunction is necessary as motion values for x, y, and z vectors are stored in two separate addresses on the ADXL345. According to the data sheet for the ADXL345, we will have to first configure the ADXL345 by writing the value 0x2D to the slave, which will prompt it to begin sampling data. We can then use our read and conjoin functions as well as a timer interrupt to sample the motion vectors of the ADXL345 at a frequency of 20Hz.

Note that the data sampled from the ADXL345 is the absolute acceleration experienced by the accelerometer. When determining how much Seismic Activity is occurring, we would need to compare each motion vector to a relative average. For this reason, we use the *Mean Absolute Deviation* function, given below, to quantitate how much each motion vector deviates from the average. The Mean Absolute Deviations of the x, y, and z vectors will be added together and compared to the quake threshold. A motion vector that does not deviate from the average will result in very low Seismic Activity, whereas a motion vector that deviates greatly will yield more Seismic Activity and possible be interpreted as a quake. The Mean Absolute Deviation will be calculated over a window of recently sampled vectors. This will be calculated using a sliding window approach, meaning that, every time a new vector is sampled, the oldest vector in the window will be replaced by the vector being sampled. This will allow the Seismic Activity to be updated every period rather than every time the window is flushed and refilled as with the standard approach.

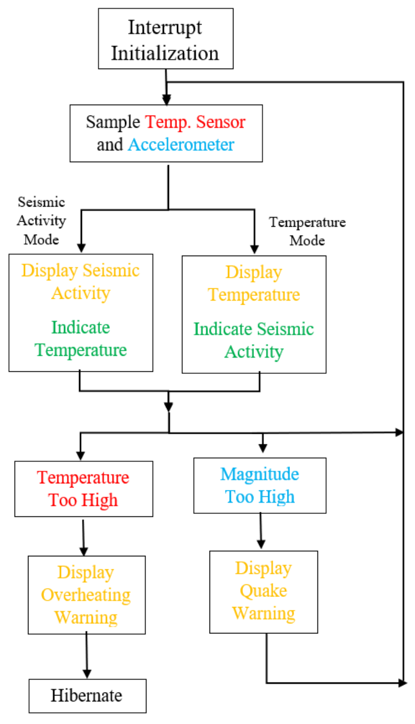

Figure 2 - Periodic Interrupt Flow @20Hz

$$\frac{1}{n}\sum_{i=1}^{n}|x_i - m(X)|$$

Equation 1 - Mean Absolute Deviation

Temperature must also be sampled alongside the Seismic Activity, so the microcontroller can determine whether it is overheating. Temperature is an Analog measurement that has an infinite number of possible values, whereas our microcontroller can only process digital information. Therefore, we must make use of the microcontroller's 12-bit Analog-to-Digital Converter (ADC) to determine the temperature of the microcontroller. We will sample from the built-in Temperature Sensor using ADC Sequence SS3. By using Sequence SS3, an interrupt will occur every time one sample is collected. This is different from, say, Sequence SS1 which will trigger an interrupt when every 4 samples are collected. We will also convert the raw 10-bit ADC value to Celsius, so it is more easily understood by the user. Like the Seismic Activity, sampling of the Internal Temperature will be triggered inside the same Periodic Interrupt occurring at 20Hz. Figure 2 shows the function of the Periodic Interrupt which will sample both the Seismic Activity and the Temperature.

The User Interface is largely dependent upon a rotary encoder which we have programmed using QEI (Quadrature Encoder Interface). Two GPIO pins will be used to detect whether the shaft of the encoder is rotating clockwise or counterclockwise. The TivaWare Library contains functions that determine the position and direction of rotation based on the phase difference between these two pins. If a change in position is detected, the selected threshold will increment or decrement once based on the direction of rotation.

The third pin of the encoder acts as a button which will be used to switch between Seismic Activity mode and Temperature mode. When pressed, an interrupt will occur and will be handled by a GPIOPort D_Handler function. Here we will clear the trigger flag for the Encoder pin and toggle a flag we instantiated to keep track of which mode we are currently in.

An LCD1602 screen will be used to display the current Seismic Activity/Temperature value being read and whether that value is less than, greater than, or equal to the threshold. Seven GPIO Pins are required to program the LCD screen correctly. Four of these pins will be used to supply the LCD screen with data including several possible characters. The other three pins will be used as a register select pin, a read/write pin, and an enable pin. We have used several helper functions that will automatically configure these pins whenever the functions are called. Functions for printing a character or printing a string will be used extensively in our main function.

In addition to the LCD screen, the user will be able to monitor the system via the on-board RGB LED. Instead of using the digital on/off state of each LED, we will utilize Pulse Width Modulation (PWM) to dim each LED to a degree, allowing us to explore considerably more possible colors. This can be done by switching the digital signal between its on and off state at a consistent rate. How dim or bright an LED will depend on the Duty cycle, or the percentage of time over a period that a signal is on for. We can adjust the Duty cycle using the Tivaware Library functions in which we specify 512 possible Duty cycles. To represent the relative severity of each measure value to its respective threshold, each LED will be represented by an expression. The Blue LED will be at its brightest when the value is at 0 and will steadily dim until the value reaches half of the respective threshold. On the contrary, the Red LED will be off when the value is less than half of the respective threshold and will brighten until the threshold is reached at which point the Red LED is completely on. The Green LED will be at its brightest at half the threshold and will dim when the value deviates from the halfpoint.

Hibernation mode will be entered whenever the internal temperature of the microcontroller surpasses the threshold set by the user. The Tivaware Library grants us access to functions that allow us to control when the microcontroller transitions into Hibernation mode as well as how it will transition back into the normal mode of operation. The microcontroller can be awakened by using the Tiva Board's *WAKE_PIN*, the pin controlled by SW2. Additionally, we can specify the GPIO pins to retain their digital value while the microcontroller is in hibernation. This can be useful as the user can view the status of the User Interface right before it enters the hibernation state. Because most of the power consumed by the microcontroller is due to the switching of the clocks, retaining the GPIO levels will not counter the effects of entering hibernation.

Experimentation:

One problem that we faced was the encoder sending multiple rotation signals as once for one rotation. This is due to the bouncing of input signals as the shaft is turned or pressed. The mechanical contacts inside the encoder do not allow for clean transitions between an on and off state. However because the signal bounces typically have a much higher frequency than the user input, we can filter out these bounces using a low-pass filter. In circuit design, this would mean using an RC filter much like that which is shown in Figure 3 in which the voltage across the capacitor is the filtered signal. The values of R and C will determine the cutoff frequency at which point the frequencies will begin to be blocked by the filter. When designing the filter, the cutoff frequency must be configured such that signal bounces are blocked while still passing even the user's fastest inputs. As we will be debouncing both the encoder's shaft as well as its button, we have to acknowledge that the two components require a different degree of filtering, i.e. shaft can be rotated at a higher rate than a button can be pressed. For this reason, we will design two different filters, each with their own cutoff frequency. The button will be filtered with a cut-off frequency of 1.6Hz to ensure that as many signal bounces are filtered out. Meanwhile, the encoder will be filtered with a cut-off frequency of 16Hz to account for the fact that the shaft can be rotated many times in one second.

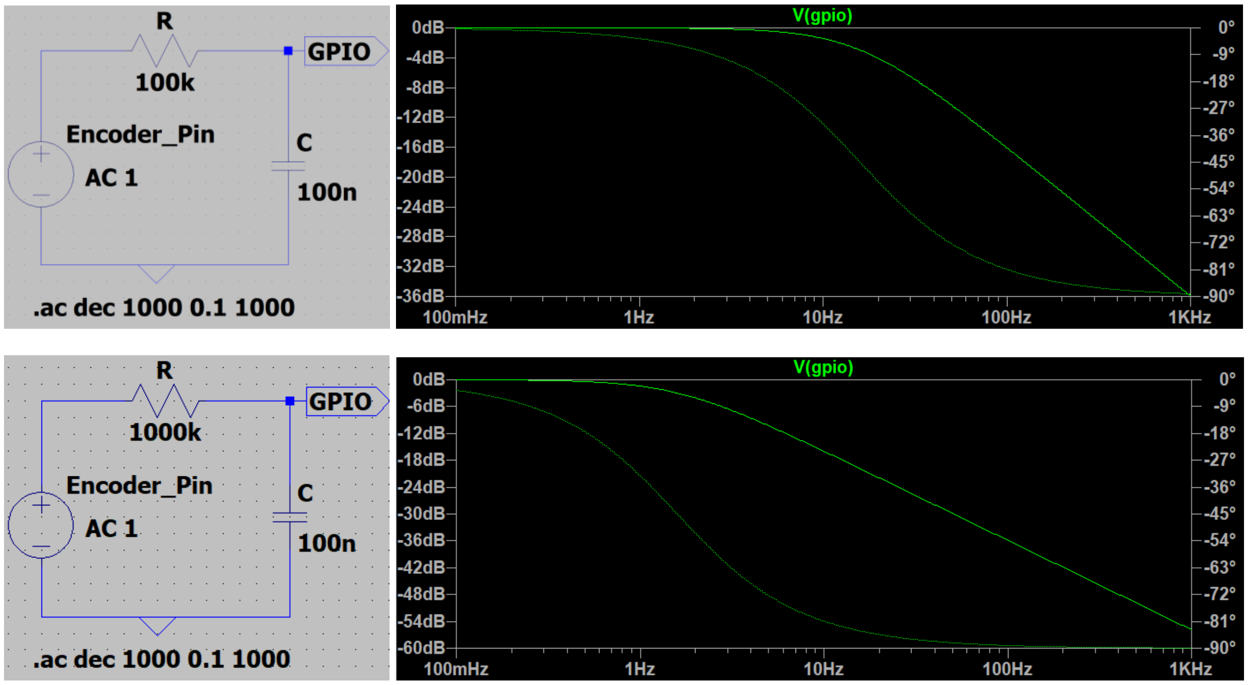

Figure 3 – Lowpass RC Filter for Encoder (Top) and Button (Bottom)

One could also use a simple delay when an action is detected to ensure that multiple inputs are not taken instantaneously. While this approach would be simpler without the use of additional hardware, this makes the peripherals within the system harder to synchronize as they are working in parallel using interrupts. Therefore, we decided to use the hardware approach in our system instead.

Additionally, one may notice the presence of a Raspberry Pi device in our setup. While the Tiva Board can power the LCD1602 Screen using its own 3.3V source, a higher contrast can be achieved by using a 5V source. Therefore, an external source, the Raspberry Pi in this case, is used solely to supply power the LCD screen, resulting in better visibility.

Validation: While observing the behavior of the LCD screen during execution of the system does tell us whether the system is behaving correctly, it is not ideal for debugging as only 32 characters be displayed at a time. In order to facilitate the testing stage of our system, we set up a UART interface solely for monitoring the values of any I/O Pin at any time. Unlike the LCD screen, we can display as many characters as we want on a terminal window to thoroughly monitor any register in the system.

In order to do so, we have to establish UART communication between the terminal and the TM4C using two Port A pins for transmission and receival of characters. “Print” functions from the UART library are used to print the value of whatever variable we provide it with.

Results: After validating the Quake Detector at each stage of the design flow, we are able to say with confidence that our Quake Detector performs the tasks specified in our design plan. When the system is at rest, we observe the behavior shown on the left in Figure 4. The LCD display correctly indicates to us that the system that the system is at rest with the Seismic Activity well below the threshold. Meanwhile, the RGB LED indicates with its Red-Orange color that the system is close to overheating, though it has not done so yet. Whereas a system detecting a large enough motion vector would exhibit results shown on the right. The system now recognizes the increase in Seismic Activity and displays to the user that a Quake has been detected.

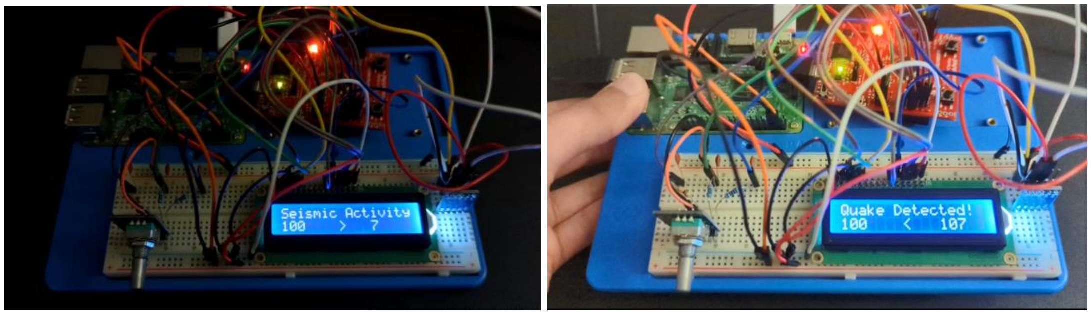

Figure 4 – Quake Detector with little Seismic Activity (Left) and with Seismic Activity (Right)

The Quake Detector displays the status of its Internal Temperature in Figure 5. On the left, the temperature has not yet surpassed the threshold, so the system can still be considered to be operating safely. However when an increase of temperature is introduced (using a heater), the threshold is exceeded and the System Overheat message is displayed. To reduce power consumption, the microcontroller will enter its Hibernation mode while retaining the voltage of each GPIO pin. Now, the UI will no longer be updated as the clocks running those peripherals are inactive due to the microcontroller entering the Hibernation State. To reenable the system, the user will simply have to press SW2 which will wake the microcontroller from its Hibernation mode. Also note the Yellow/Green color of the RGB LED which indicates that the system is at roughly half of what is considered to be a quake, likely caused by the user when interacting with the encoder which caused slight movement to the system.

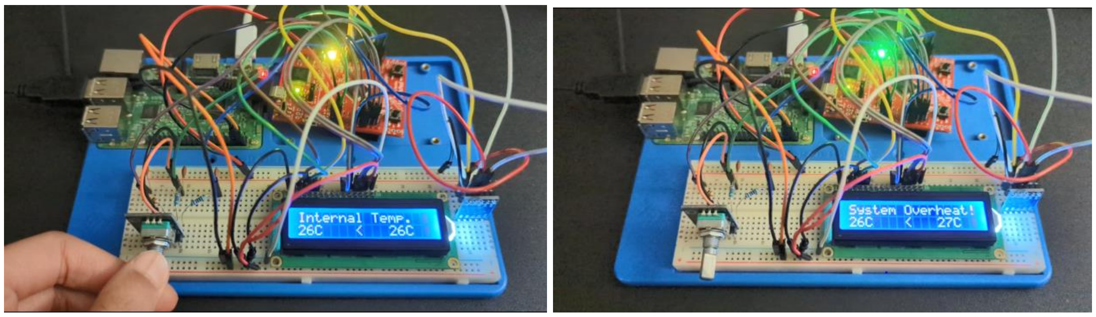

Figure 5 - Quake Detector Before (Left) and After (Right) Overheating

Discussion: While our project already possesses the functionality we aimed to achieve; some improvements can be made to increase the robustness of the system. For instance, we can add additional power modes to the system, so that the overheating can be divided into tiers based on severity. For example, we can just disable the main clock when a lower threshold is reached, whereas all peripherals can be disabled upon surpassing the higher threshold.

We could also configure the wake condition of the hibernation module with the interrupt pin on the ADXL345 accelerator so that the microcontroller automatically wakes from hibernation when a quake is detected. In this situation, the Accelerometer will be able to remain active even without an active clock driving its operation.

One function we wish to implement in the future is the ability to interact with the system remotely via IoT. The reason for using the Raspberry Pi to supply power to the LCD screen rather than, say a level shifter, is because the Raspberry Pi possesses abilities that the Tiva Board does not. As a microcomputer, the Raspberry Pi can create its own HTTP web server which can monitor and control the Quake Detector. This will allow a user to interact with the system anywhere and on any device which can connect to the internet. Of course, this was not one of the core topics of the class, so we did not pursue this.

Conclusion: Throughout this project, we designed a system capable of detecting quakes using the ADXL345 Accelerometer. In doing so, we familiarized ourselves with the I^2C Protocol, useful for sending multiple bytes of data serially. We have gained experience in designing low-power embedded systems by making use of the Tiva Board's Hibernation module. By sending the microcontroller into a low-power state whenever the internal temperature exceeds the safe threshold, we can ensure that no damage will be done to the microcontroller due to overheating. Additionally, we have built an intuitive UI that displays relevant information via LCD and LED and even allows the user to tune parameters of the system using an Encoder. As an Embedded System, the Quake Detector can be built into a larger system such as a Home Monitoring System, or it can be used on its own.

References:

1. TivaWare Peripheral Driver Library
2. TM4C Data Sheet
3. ADXL345 Data Sheet
4. LCD1602 Data Sheet
5. Interfacing LCD1602 with Arduino
6. Jain, Rishabh; How to Interface a LCD Display with TIVA C Series TM4C123G LaunchPad from Texas Instruments; https://circuitdigest.com/microcontroller-projects/how-to-interface-16x2-lcd-with-tiva-c-series-tm4c123g-launchpad