Wesley Gardner
RuBW
CapStone Fall 2013

Design and Requirements

Project RuBW

Acronyms

EEG - Electroencephalography
ADC - Analog-to-Digital Converter
FFT - Fast-Fourier-Transform
IB - Interface Board
IC - Integrated Circuit
GPIO - General Purpose Input/Output
SPI - Serial Peripheral Interface
RuBW - Realtime BrainWave analyzer (pronounced “Ruby”)

Fundamental Requirements

1) Build an EEG device that uses electrodes on the scalp to measure brain-activity.
2) The scalp electrode(s) will be sampled at a rate of at least 100kHz.
3) The scalp electrode(s) will be sampled with an ADC that’s at least 12-bits wide and samples with true differential input.
4) All signal processing (less an ADC conversion) will be done in software.
5) The device must perform a brain-activity dependent task(s).

Design Decisions

6) The microprocessor will perform a 1-D FFT to convert the sampled time domain signal to the frequency domain. This will facilitate further analysis.
7) The Raspberry Pi microprocessor has been chosen for the processing platform.

Requirement Flowdown

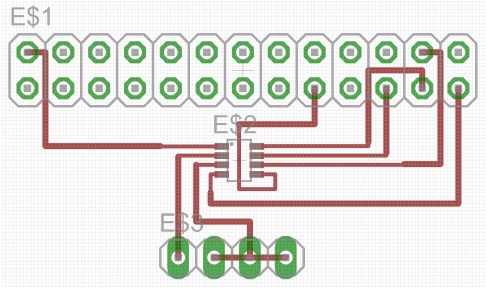
1.A) One (or more) electrodes will be placed on the scalp, two at various grounding points, and a final 60Hz ground antenna to help eliminate background noise.
2.A) The microprocessor must complete the sampling and processing steps in no more than 10μs.
2.A.1) The sampling time must be significantly less than 10μs to allow time for processing.
2.A.2) The microprocessor must compute all signal analysis within the time remaining in it’s 10μs cycle after signal acquisition.
3.A) EEG electrodes must interface to the ADC, which inturn must interface to the Raspberry Pi microprocessor.
3.A.1) An interface board (IB) must be designed and built which houses the ADC.
3.A.1.A) The IB must interface with the Raspberry Pi’s GPIO Header.

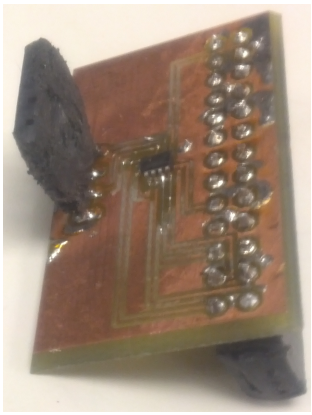
3.A.1.B) The IB must provide an interface for signals from the electrodes.
4.A) Software must provide an interface for controlling and sampling the electrodes.
5.A) The microcontroller must make a decision(s) based on recent brainwave activity within it's 10µs cycle.
6.A) Determine effective resolution of FFT.

Requirement Satisfaction by Design

The MAX1286 IC has been chosen for the ADC. This IC samples true differential input at speeds up to 100kHz, and is 12-bits wide, satisfying 2,3. This IC operates with a clock speed up to 8MHz, and with total signal acquisition time from start to finish of 5.1µs. This allows one sample to be collected in as little as 6.6µs, satisfying 2.A.1.

The IB printed circuit board design Satisfies 3.A and below. The ADC is wired directly to the GPIO bus, satisfying 4, since no signal processing is implemented outside of software.
On the right on is the interface to the Raspberry PI GPIO bus. On the left is the interface for plugging in external wires, coming from the electrodes. In the middle lies the ADC. See Pictures:





The communication between the ADC and the microprocessor can be idealized as SPI. The SPI Control software was hand written to ensure optimization and to meet the timing requirements, satisfying 4.A.

7: The Raspberry Pi microprocessor has the requisite sampling ability, is able to receive data at >150KB/s, and has sufficient memory and processing ability to store the data, perform a 1-D FFT over time slices of data, and still have processing time remaining for further data analysis.

Requirements Pending Satisfaction

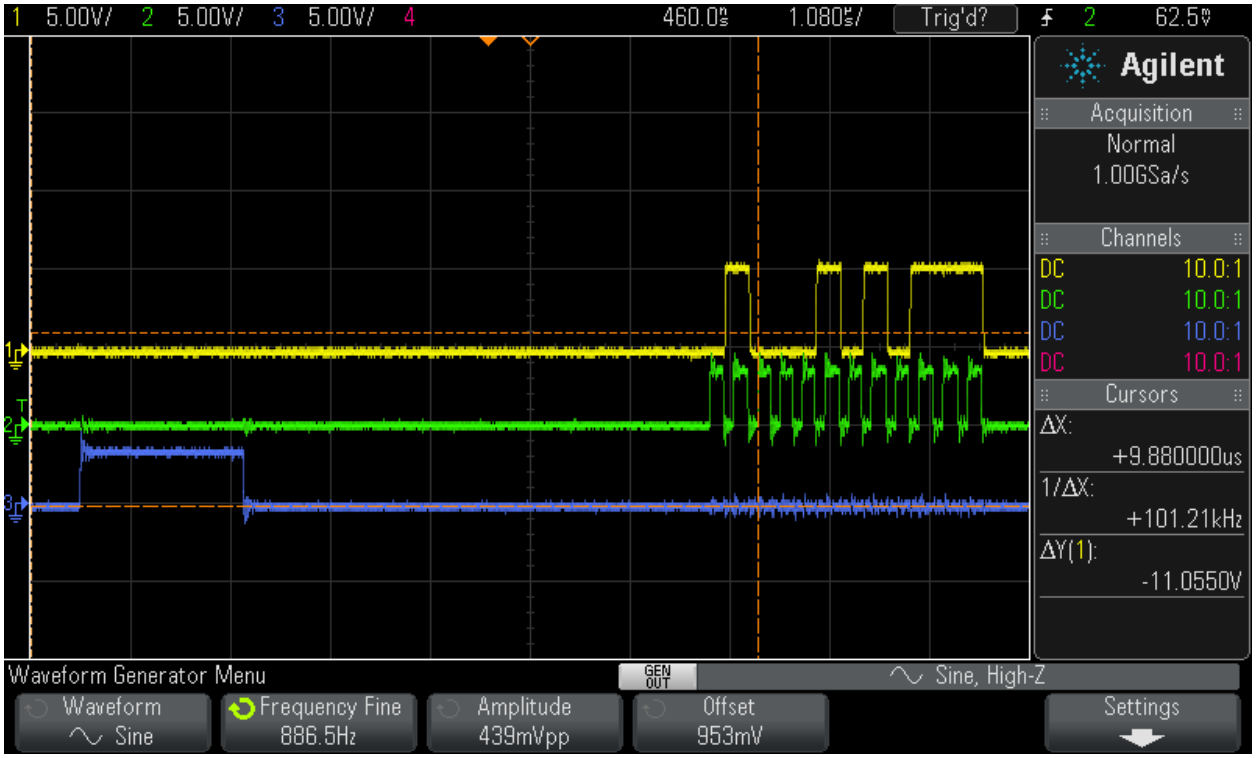
The following requirements are not yet satisfied: 1, 1.A, 2.A, 2.A.2, 4, 5, 5.A, 6, 6.A
1: Electrodes are not in place. Brain Activity is not yet being measured.

1.A: Electrodes are on order.
2.A, 2.A.2: Signal Analysis is not implemented.
4: Signal analysis is not implemented.
5: No decisions are made based upon brain activity, since 1 is not yet satisfied.
5.A: No decisions are made based upon brain activity, since 1 is not yet satisfied.
6: Signal analysis is not implemented.
6.A: Signal analysis is not implemented.

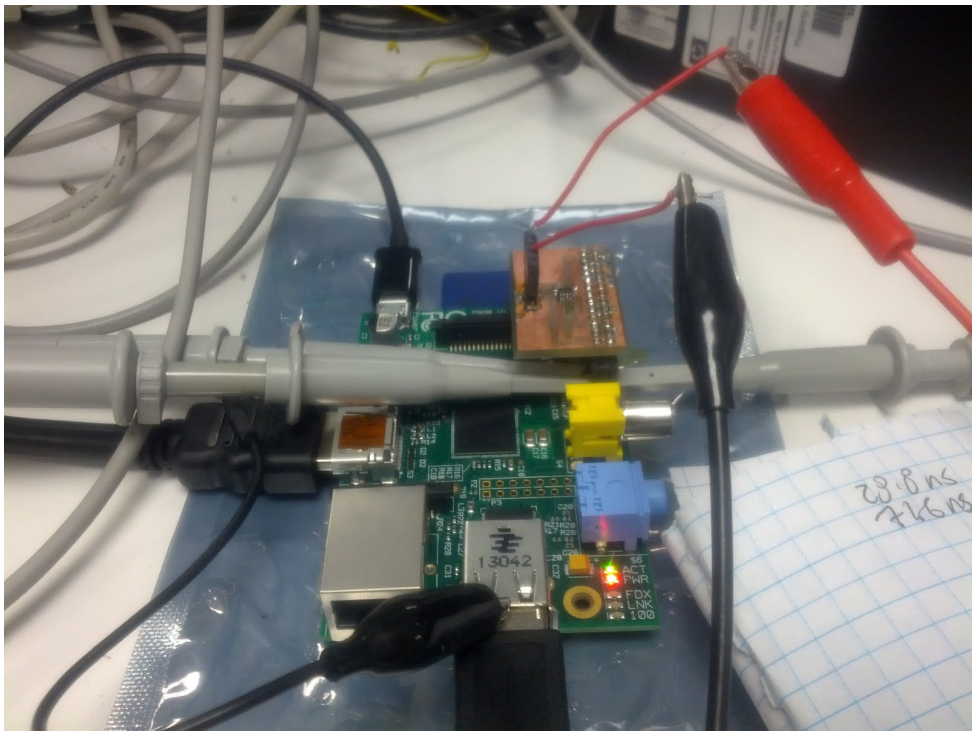
Project Status

All physical components are chosen and integrated into the system, with the exception of the electrodes, which are on order.Typically, integration and testing are the most difficult and time consuming part of a project. That stage is nearly finished, lacking only the integration of the electrodes themselves, which is expected to be smooth and quick.

Specifically, the final IB design in implemented and built.
The IB has been integrated onto the Raspberry Pi.
The Raspberry Pi implements a program that samples the ADC in approximately 6.6μs, and at a rate of 100kHz. However, there is no further signal processing implemented. The ADC Value is merely read and stored at this point. See picture below for a typical sample acquisition. The green line is the clock signal, the yellow line is the serial data line, and the blue signal corresponds to the CNVST pin on the ADC (See Reference Section for more information). The time delay between the blue signal activation and the data acquisition is required by the ADC.



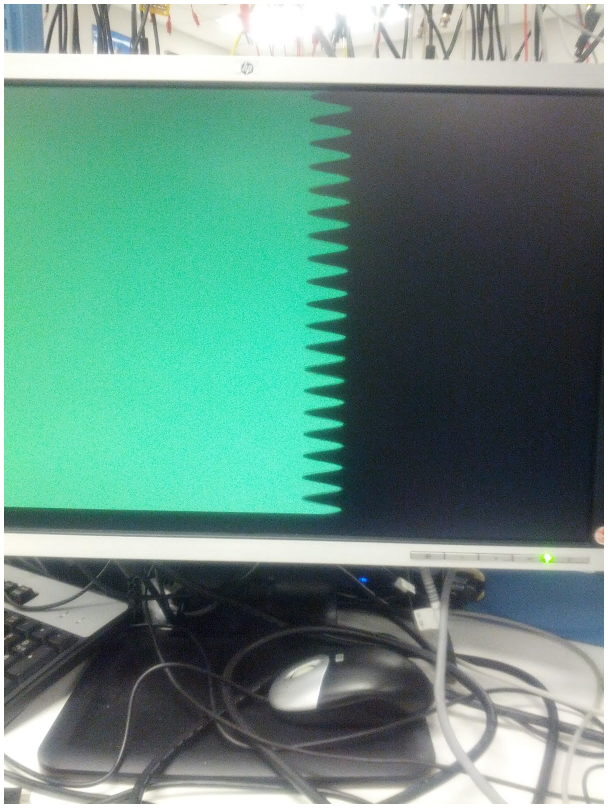
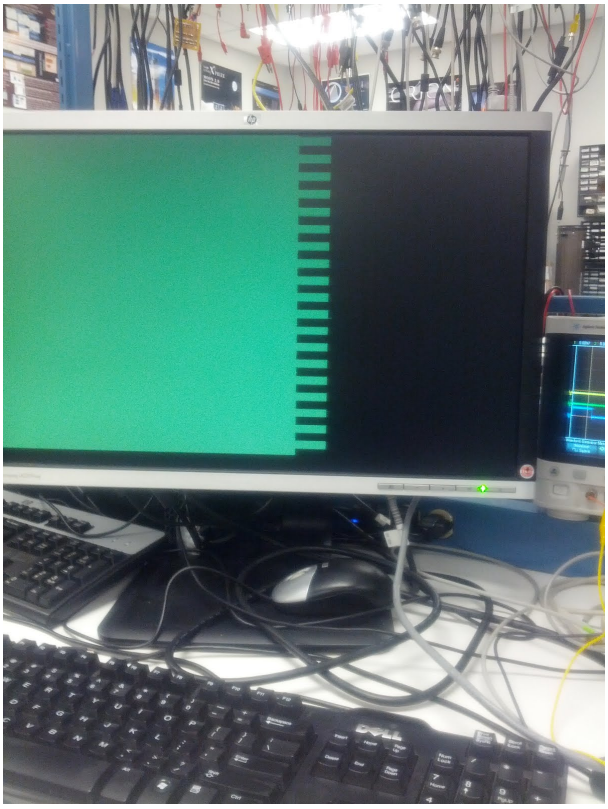
The following picture shows the integrated hardware with the Raspberry Pi board, the IB, and all peripheral cables needed to operate the hardware as well as an oscilloscope that was being used for functionality testing at the time.



Currently the software is able to sample from the ADC at a rate of 100kHz with an estimated 240 atomic operations available between samples. This can be optimized by making use of the timing delays inherent in the sampling routine, but that will only be done if necessary.

The current operational software is available for download from the reference section. This software includes a GUI interface for testing. The GUI turns the screen into a horizontal bargraph of voltage vs time, updating one row of the screen per sample, and cycling back through when it reaches the end of the screen. The implementation of the GUI drops the sample rate to ~54kHz, and with 1000 lines on the screen the software is updating the screen at ~54Hz. Since the GUI slows the process significantly, it will probably not be present in the final design without significant optimization, but for testing/demonstration it is pure gold.
The software maps a section of physical memory from /dev/mem to program memory. This memory portion controls GPIO bus activity. The BCM2835 datasheet included in the reference section details how to manipulate these registers. Specifically of interest are pages 90-96. Here address 0x7E2... is mapped to physical memory address 0x202... within the operating system.
The general gist is that there are registers controlling the input/output state, the pull-up/pull-down state, and the current level associated with each pin. These registers combine to yield full control over the pins. Through this the SPI bus is implemented as a precise timing sequence of register manipulation. For performance reasons, the SPI data acquisition is implemented as a pre-compiler macro rather than a function call.
The software directly manipulates the frame buffer in /dev/fb0 to display the GUI. As stated above, this is just a horizontal bar graph of scaled voltage vs time. Since the frame buffer is written in row major order, it is constant time to write one horizontal line of pixels to the buffer, making the processing for each sample constant time. This type of time-wise constancy is necessary to ensure the device eventually satisfies 2.A for each sample.
This characterizes the biggest challenge for this project, doing as much computational analysis as possible within the required timing constraints. We need to distinguish several states or characteristics of brain activity and be able to make a decision(s) within the allotted time, before moving on to the next data state. This project integrates both digital signal processing, and high performance computing on an embedded system.

With the current GUI implementation, the system acts like a basic oscilloscope. Insomuch as that it is fun to play with an oscilloscope, below are two examples demonstrating this functionality by feeding various signals into the hardware and processing them with the given software.

The remaining time has been spent with signal analysis algorithms. I detail one here that has a significant chance of being used in the final design. Other algorithms that are used will be detailed in later documentation.
The following is a derivation of an original algorithm that might be implemented in the final design. Although I call this original, that only goes so far as to say that I thought of it, and not that it hasn't been thought of before. An algorithm this simple has almost definitely been used before, while I just do not know what it might be called to try to find it.
Given 4 samples from a sinusoidal signal (values a,b,c,d in order), sampled at a frequency f, the angular frequency, w, of the signal can be approximated by:

$$w = f\sqrt{\frac{(a-d)+3\,(c-b)}{c-b}}$$

The derivation is as follows. Let the time dependent sinusoid be $f(t) = A\sin(B\,t + C) + D$, where A,B,C,D are real (unknown) numbers. Then

$f'(t) = A\,B\cos(B\,t + C)$, and $f^{(3)}(t) = -A\left(B^3\cos(B\,t + C)\right)$. Therefore, $\frac{f^{(3)}(t)}{f'(t)} = -B^2$, and B is the angular frequency of the signal.
Now assume we have sampled the sinusoid at 4 points (0,a), (T,b), (2T,c), and (3T,d), where the first element of the point is the time sampled, the second element is the value sampled, and 1/T is the sampling frequency. The first derivative in the middle of the sampling range (at time=3/2*T) can be approximated by

$f'\left(\frac{3}{2}T\right) = \frac{c-b}{T}$. Similarly the derivatives at time=1/2*T, 5/2*T can be approximated. From these numbers, the second derivative can be approximated in a similar fashion at points time=T, 2T. The approximation of the third derivative at time=3/2*T follows simply. The resulting equation for the angular frequency is given above in terms of the original dataset.

Remaining Schedule

A) Integrate and test electrodes.
B) Implement signal analysis.
C) Make brain-wave dependent decisions via signal analysis.

The electrodes are on order, and are expected to arrive sometime the week of the 20th. I will proceed to A at that time. I expect to finish A within that week (including the corresponding weekend).
B and C blend together as the work I can do with C depends entirely on the processing I can accomplish in B. Thus B and C will be developed together until the end of the project. I could if rushed implement the above algorithm, and have it make a binary decision based on the current estimated instantaneous frequency of the brainwaves, in less than a week; but I can develop much more interesting and useful functionality given the remaining timeline of the project.
Ultimately, a personal goal of mine is to have this system generate sound in realtime based on brainwave activity. This would allow one to actively synthesise arbitrary waveform sound in realtime. This may or may not get implemented before the capstone project nears completion depending on the complexity of the algorithms. However, my device will do something interesting and useful.
I know this is rather ambiguous, but that's how I roll. I like to see where the analysis takes me, and I ensure you that I will find something very interesting. I always do.

While I am waiting for the electrodes, I will be working on the following immediate subgoals:
-Find an optimization independent nop C routine.
    -Apply -O3 optimization with this nop routine to see how fast I can get it to run. This would also give me more processing time between sampling.
-Research efficient analysis techniques in addition to FFT's

Reference Material

The following reference materials are too long to concatenate with this document, but deserve due reference.

The datasheet for the microcontroller chipset, the BCM2835, may be found for the duration of the project at:

https://docs.google.com/file/d/0Byd6ngUnOQEeTXdxMkE0Mkx6Rms/edit?usp=sharing

The datasheet for the ADC IC, the MAX1286, may be found for the duration of the project at:
https://docs.google.com/file/d/0Byd6ngUnOQEeMnloTnZiQVVqWnc/edit?usp=sharing

The current version of the software is available for download at the following url. I considered copying the text at the end of this document, but it was syntactically destroyed when pasted in here, so I offer a download of the cpp file instead. Go word processors!
https://docs.google.com/file/d/0Byd6ngUnOQEeRjQ3Uno4V2gwQW8/edit?usp=sharing