EEL 4924C Design II

Preliminary Design Report

Ocarina Home Automation

Oracle of Seniors

**Team:**

Adrian Alvarez (EE)

Ishmael Contreras (EE)

# Project Abstract

Home automation or *domotics* is currently experiencing a large growth in the market

share for creative IoT applications. Our idea is to have a central control unit that will be listening for successive notes i.e. (a known tune acting as a passcode) that will trigger certain peripherals e.g. a motor to turn a fan, a water dispenser, or a Bluetooth enabled device. NOTE there are many possibilities with a configuration such as this one. The main processing will be to parse the instrument for the notes and be able to filter, window, and analyze the fundamental frequency of that tone. There will be an LCD display to monitor the current status of the appliances that can be controlled.

# Project Features/Objectives

The primary objective of this project is to provide a creative way of changing and configuring settings on household appliances without physical intervention that can assist handicapped individuals or the lethargic folk. The main method of control will be through pitch detection of musical notes using a pre-packaged ocarina/flute.

The Home Automation system will have the following components:

* Microphone Audio Capture and Signal Conditioning
* ADC Sampling of Audio Signal
* Audio Processing using TI TMS320F28335 DSP
* Motor Driver and Control
* Solenoid Driver and Control (Door Lock)
* Solenoid Valve or Water Pump Driver and Control
* LED Control
* Environment Sensing (Light, Temperature, Pressure)
* LCD Display
* Power Design for all Digital and Analog Circuits

## Software Objectives

To accomplish the task of audio processing and task triggering, we will base our code around the TI-RTOS kernel. This will make scheduling of tasks simpler and allow for real time control of hardware peripherals. The main unit will connect to the available equipment and be in charge of manipulating the captured audio from an omnidirectional dynamic microphone that will receive a continuous stream of audio.

Inside the SRAM, there will be pre-programmed songs that upon replication from the instrument, will generate the appropriate control signal. For specific applications, the intensity can be adjusted by playing certain notes i.e. for a fan, the PWM duty cycle can be changed by playing a C to decrease and a C’ to increase. The songs inside the SRAM could be reprogrammed to other tunes by the user.

For the main PDA algorithm, we will need to use proprietary TI drivers for FFT and convolution implementations, additionally we also must incorporate the proper library settings for out project and DSP.

Separate sensors that are locked from the user will be utilized to ensure that certain modules do not operate erroneously, for example, a water pump will not dispense water if a pressure sensor does not detect a cup.

## Analog Objectives

Multiple motors and pumps (solenoid) will require analog circuitry,

## Audio Processing Objectives

Analog:

It is imperative to amplify the incoming signal from the microphone with an appropriate preamp circuit and then pass it through an active anti-aliasing filter (low-pass).

Digital:

IIR bandpass filter to get rid of possible harmonic contamination and then we will ensure that the TI DSP will be processing a conditioned audio signal in real time using FFTs to determine the maximum frequency in the spectrum. It will then use pattern detection to determine if a specific set of frequencies were being played contiguously (i.e. A song). If a specific “song” is detected, it will trigger a designated home automation task. Depending on the timing requirements, we may implement a block based approach with a DMA based double-buffering to ensure that the incoming signal does not experience too much variance.

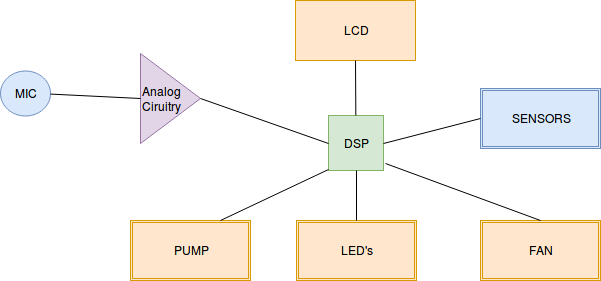
# Concept/Technology Selection

‘Ocarina home automation’ makes for a simple yet charismatic design that can be applied to a myriad of home appliances. Current ideas on the market are the Amazon Echo, Google Home, and Apple’s Siri, however it is important to mention that they require proprietary hardware and the speech recognition is not optimized for all languages, our idea simply requires that you play a flute.

A dedicated DSP will be used for audio processing and control of all peripherals. The TI TMS320F28335 DSP is a suitable processor for this application. It contains a C2000 150MHz core and a dedicated Floating-Point Unit. It has plenty of peripherals such as ADC, McBSP, SPI, I2C, PWM, and many GPIO pins. These extra peripherals allow us to dedicate specific pins to the various home automation tasks.

Real Time Operating Systems (RTOS) allow users to take advantage of a scheduler and basic thread synchronization techniques such as HWI, SWI, and Semaphores to implement their applications. This makes it simple for the user to schedule various tasks and functions based on priority and maintain real time characteristics. TI offers the TI-RTOS kernel as a software solution for real time operating systems on their devices, such as the TMS320F28335 DSP. This software package contains libraries and APIs which the user can use to develop their code for RTOS on the target hardware. It allows the user to maintain control of all hardware peripherals. We will use TI-RTOS in our project because it will allow us to build our system on top of a RTOS and aid us in function scheduling during the software development part of our project. This solution is preferred because otherwise having multiple interrupts could cause severe data contention because the interrupt vectors in the TI DSP are not ordered by request.

# Flowcharts & Diagrams



# Proof of concept and MATLAB simulations

function freq = welch(x,N,fs)

% Choose FFT size and calculate spectrum

[Pxx,f] = pwelch(x,gausswin(N),N/2,N,fs);

% Plot frequency spectrum

plot(f,Pxx);

ylabel('PSD'); xlabel('Frequency (Hz)');

grid on;

% Get frequency estimate (spectral peak)

[~,loc] = max(Pxx);

freq = f(loc)

title(['Frequency estimate = ',num2str(freq),' Hz']);

>> [x,Fs] = audioread('ocarina\_low\_frequency.mp3');

>> Nsamps = 226303;

>> fsamp = 44100;

>> Tsamp = 1/fsamp;

>> t = (0:Nsamps-1)\*Tsamp;

>> Nfft = 1024;

>> plot(t,x);

>> X = fft(x);

>> X = X(1:length(x)/2+1);

>> freq = 0:fsamp/length(x):fsamp/2;

>> plot(fsamp,abs(X))

% I ran welch function to verify findings as it is MATLAB’s best audio detection algorithm

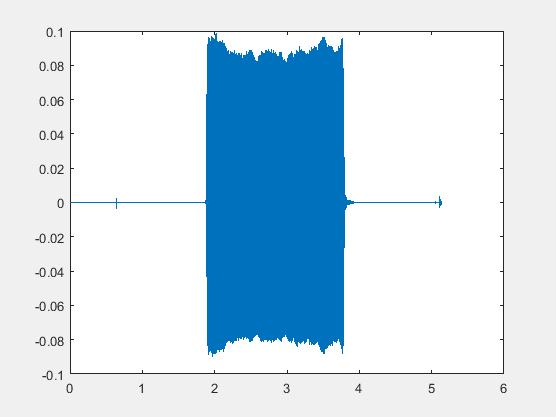


Figure 1: lowest reproducible note (C)

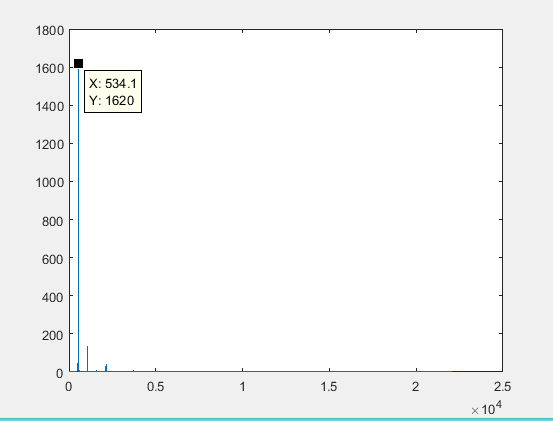


Figure 2: 512 point FFT frequency approximation of waveform

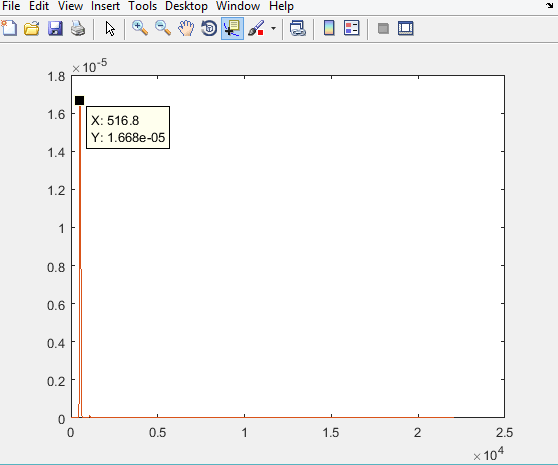


Figure 3: Welch’s algorithm of pitch detection of high accuracy overall 3.24% off

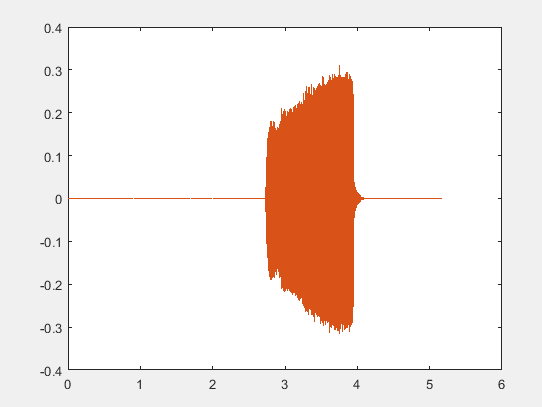


Figure 4: Highest reproducible note (F’)

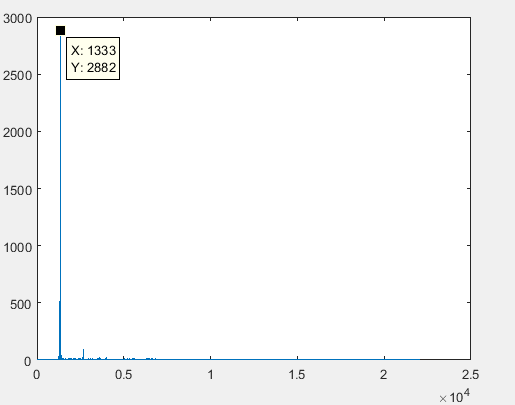


Figure 5: 512 point FFT of second waveform

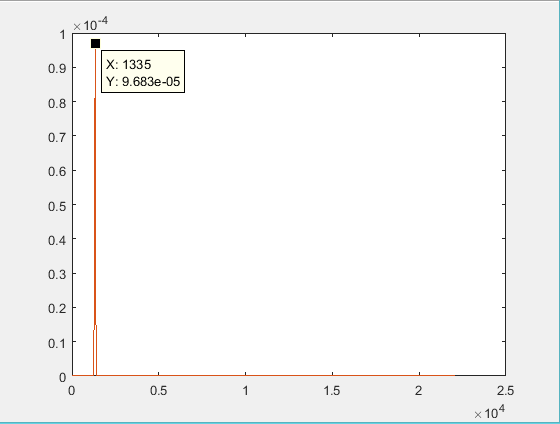


Figure 6: Welch’s method for second waveform with near exact precision

NOTE: im recording on a low sensitivity HyperX cloud 2 microphone so lower f’s aren’t that great