How to cite this paper: Kyi Phyu Soe | Win Zaw Hein "Design and Construction of Musical Lighting Water Fountain" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-3 | Issue-5, August 2019, pp.161-165, **https://doi.org/10.31142/ijtsrd25254**

IJTSRD25254

ABSTRACT

This paper presents the design and construction of a musical water fountain system driven by Arduino using Fourier Transform technology. Audio signals were taken from sources such as MP3 player, smart phone, laptop, etc and these signals were first amplified using LM386 IC and then transferred into an analog input of Arduino for FFT processing. To give out signals for DC pump motors, digital output pins was used. With various frequency band, turning on and off the DC motors and LED systems was created the lighting and water spray effects which should go well along with showed music. The response of this foundation is analysed for different audio frequency range and pump motor control was emphasised and, fountain model is designed for domestic usages.

KEYWORDS: *Musical water fountain, Fast Fourier Transform, Audio frequency, Pump, Signal Amplifier, Arduino*

I. INTRODUCTION

Nowadays, fountains that just make patterns with water jets have developed into multimedia shows with music, light, and other effects. They are used in parks, buildings and offices for entertainment and attraction. A musical fountain is especially achieved by synchronizing light and water jets to the frequency of music to produce a harmonic spectacle.

Moreover, dancing fountains have been implemented using FFT (Fast Fourier Transform) and FHT (Fast Hartley Transform), to turn on valves of jets corresponding to different frequency range in music. Each different range will correspond to a different valve or jet which comprises a musical water fountain operating to the beats and rhythms of the songs.

This paper presents a system design which can play any song without the need of the prior arrangements. The control system for MFS samples audio frequencies through FFT algorithm [1, 4]. This algorithm is set up on an Arduino, which receives audio signals in the time domain and transforms them into the frequency domain via FFT algorithm. The signals in the frequency domain will be standardized and used to control pumps, and LEDs, creating simultaneous lighting and water spray effects as well.

II. Musical Water Fountain

As the input of the system, audio signal is achieved from devices such as smart phone, MP3 player or Laptop etc. Then, this signal is amplified into 12V range speaker for listening song. For ADC (Analog to Digital Converter) pin of Arduino, 12V range signal should be reduced. Voltage divider circuit is apply for voltage matching. Arduino will perform FFT algorithm and it will control pumps and LEDs of the fountain. The overall block diagram of the system is described in Figure1.

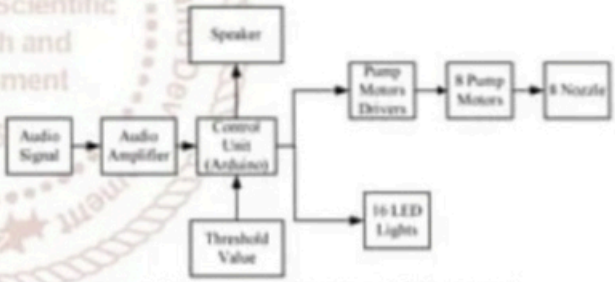

Figure1. The Block Diagram of the System

III. Design Consideration

This system can be divided into four parts and they are audio signal amplification, FFT application, hardware control for fountain, software control for fountain and overall circuit diagram.

A. Audio Signal Amplification

For amplifying smart phone audio signal to 12V speaker range, LM286 audio amplifier is used. Its gain is internally set to 20 to keep external part count low, but the addition of an external resistor and capacitor between pins1 and8 will increase the gain to any value from 20 to 200. [6] The inputs are ground referenced while the output automatically biases to one-half the supply voltage. In this system 12V supply range makes 6V DC bias level. Decoupling capacitor of 250 uF makes 0V to 12V signal to +6V to -6V ac swing wave for sound signal.

12V output is too larger and voltage divider is needed. One third of maximum output voltage is feed to analog input of

Arduino as shown in Figure2. 1kΩ and 2kΩ resistors will reduce LM386 amplifier output to 4V maximum so that this signal cannot damage Arduino because the acceptable range of Arduino analog channel is 0V to 5V range.

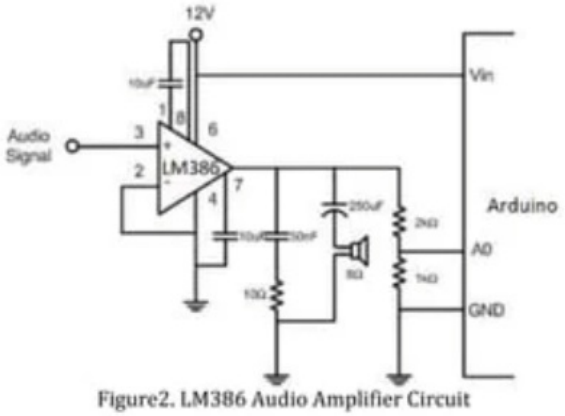

Figure2. LM386 Audio Amplifier Circuit

B. FFT application

Fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

The DFT is obtained by decomposing a sequence of values into components of different frequencies. Fast Fourier transforms are widely used for applications in engineering, science, and mathematics. The basic ideas were popularized in 1965, but some algorithms had been derived as early as 1805. In 1994, Gilbert Strang described the FFT as "the most important numerical algorithm of our lifetime" [7].

An FFT computes the DFT and produces exactly the same result as evaluating the DFT definition directly; the most important difference is that an FFT is much faster [7]. Let x(0), x(N−1) be complex numbers. The DFT is defined by the formula.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} = \sum_{n=0}^{N-1} x_n w^{-kn} \qquad (1)$$

Where k = 0, 1, 2, ..., N-1

Evaluating this definition directly requires O(N2) operations: there are N outputs X_k, and each output requires a sum of N terms. An FFT is any method to compute the same results in O (N log N) operations. All known FFT algorithms require Θ (N log N) operations, although there is no known proof that a lower complexity score is impossible. To illustrate the savings of an FFT, consider the count of complex multiplications and additions. Evaluating the DFT's sums directly involves N2 complex multiplications and N(N−1) complex additions, of which O(N) operations can be saved by eliminating trivial operations such as multiplications by 1. The radix-2 Cooley–Tukey algorithm, for N a power of 2, can compute the same result with only (N/2)log2(N) complex multiplications (again, ignoring simplifications of multiplications by 1 and similar) and N log2 (N) complex additions.

FFT can be expressed as the following procedure as shown in Figure3.

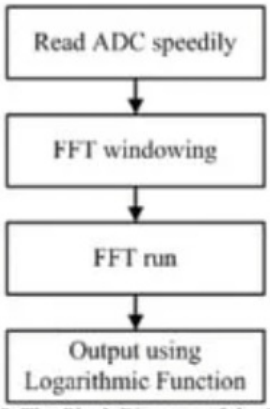

Figure3. The Block Diagram of the System

To read ADC speedily, default analog reading should not be applied. Default Prescaler setting is 128.

$$\text{ADC clock frequency} = \frac{\text{clock crystal}}{\text{Prescaler}} \qquad (2)$$
$$= \frac{16\,MHz}{128}$$
$$= 125\,kH$$

For 13 Multiplexer in ADC, one channel of ADC reading speed will be

ADC speed per one channel = ADC clock frequency /13

= 9.6 kHz

$$\text{ADC speed per one channel} = \frac{\text{ADC clock frequency}}{13} \qquad (3)$$
$$= \frac{125\,kHz}{13}$$
$$= 9.6\,kHz$$

$$\text{Sampling time} = \frac{1}{\text{ADC speed per one channel}} \qquad (4)$$
$$= 104\,microseconds$$

After changing Prescaler into 32 for equation 2, 3 and 4, ADC clock becomes 500 kHz, and ADC speed per one channel becomes 38.4 kHz. New sampling time becomes 26 microseconds, which is quite fast for minimum audio signal sampling rate 38 kHz.

Another preparation for ADC and putting even bins to FFT algorithm are implemented in Arduino IDE. Arduino ADC longs for 10 bits. ADC value can be get by combining 8 bit left shit of upper 2 bits ADC upper register and 8 bits of ADC lower register. Incoming signal range is DC value or zero to five voltage range. For converting signed value (plus and minus value), these values are needed to be subtracted by 512 which is one half of 1024 (10 bit resolution). For log value calculation, above signed value is converted into 16 bit signed value. Above setting can be expressed as follows in Arduino IDE.

```
byte m = ADCL;          // contain lower 8 bit;
byte j = ADCH;          // contain upper 2 bit
int k = (j << 8) | m;   // form into an int
k -= 0x0200;            // form into a signed int
k <<= 6;                // form into a 16b signed int
```

For combining sine waves, Hann windowing is applied as FFT windowing. Windowing reduces the amplitude of the discontinuities at the boundaries of each finite sequence acquired by the digitizer. Windowing consists of multiplying the time record by a finite-length window with an amplitude that varies smoothly and gradually toward zero at the edges. This makes the endpoints of the waveform meet and, therefore, results in a continuous waveform without sharp transitions. This technique is also referred to as applying a window.

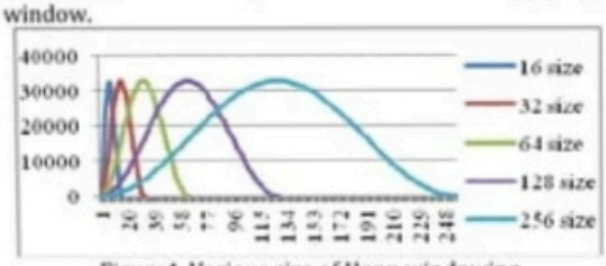

Figure4. Various size of Hann windowing

For this research work, 512 sample size is applied and 256 Hann windowing is applied.

In the context of fast Fourier transform algorithms, a butterfly is a portion of the computation that combines the results of smaller discrete Fourier transforms (DFTs) into a larger DFT, or vice versa (breaking a larger DFT up into sub transforms) Figure5 and Table 1 show 16 order of butterfly bit reversal pattern.

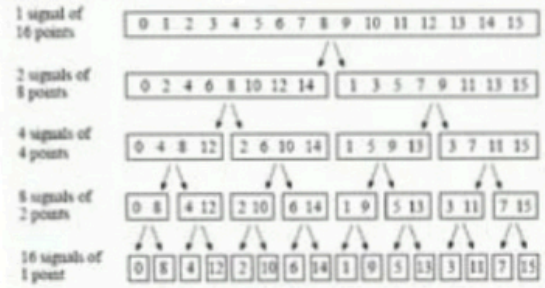

Figure5. Sample of 16 bit butterfly bit reversal pattern

Table 1. 16 reorder of butterfly pattern			
12	1100	3	0011
14	1110	7	0111
13	1101	11	1011
8	1000	1	0111
10	1010	5	0101
4	0100	2	0010

To perform Sine and Cosine calculation, Arduino like low cost microcontroller will make delay problem. Lookup table for 2(pi)k/N value is needed for faster response. Figure6 shows pre-calculated 16 parts of Sine and Cosine calculation. These lookup tables are cascaded in library of FFT program.

Figure6. Lookup table calculation for Sin and Cosine values within 32767 to -32767 16 bit signed values

For the output of FFT algorithm, output value should be reduced as one register side where Arduino Mega2560 is 8-bit size register or 255 value maximum. Entire conversion can be expressed as Figure7.

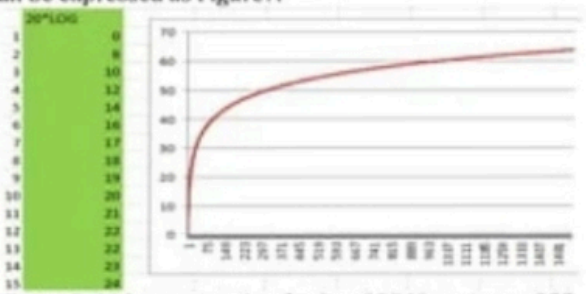

Figure7. 20 log conversion for 0 to 32768 output to 255 values

C. Hardware Control for Fountain

Each different range of frequencies will correspond to a different pump which comprises a musical water fountain. 8 car wind screen washer pumps (in Figure8) are controlled via Arduino Mega2560.

Figure8. 12V 35W car windscreen washer pump

Eight pumps are applied for this system, each pump can draw up to 3A maximum. But FFT drive the output in the form of pulse. The current draw of each motor is reduced to 1A. For 8 pumps, 8A current draw is considered and 10A maximum current draw is expected because Arduino and LED will not draw more than 2A maximum.

Figure9. 240W 12V supply

In Figure9, 240W 12V supply is expressed for enough current draw of whole system.

D. Software Control for Fountain

For overall fountain control, ADC reading is performed firstly. Then other stages of FFT such as FFT windowing, butterfly bit reversal and logarithmic function are performed accordingly. The threshold value is defined as 90. The logarithmic outputs which are exceeded 90 will activate appropriate pumps and LEDs. These chart is expressed in Figure10.

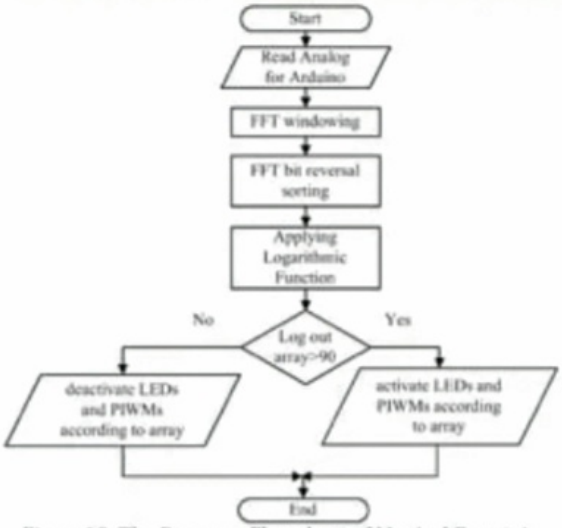

Figure10. The Program Flowchart of Musical Fountain

E. Overall Circuit Diagram

For each motor IRFZ44N N-channel MOSFET is applied because it can operate up to 49 A and 55V maximum. Digital pins 2 to 17 of Arduino Mega2560 are used for LEDs and 2,4,6,8,10,12,14 and 16 pins are connected to pump drivers. Analog pin A0 is applied for Analog input. Figure11 represents overall circuit diagram of the system.

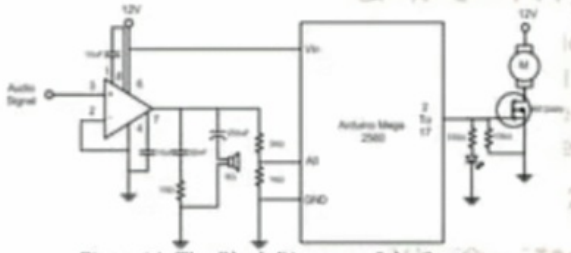

Figure11. The Block Diagram of the System

IV. Simulation and Experimental Results

Before construction, the sample program is tested using Proteus Simulation Software. Figure12 show simulation test of the system. Firstly sample audio wave file is feed to the LM386 based audio amplifier circuit and the output signal can be listened via laptop (PC) speaker. The output wave from digital pin 5 and 9 of Arduino Mega is watched using oscilloscope. Moreover, the input and output of LM386 based circuit is watched there.

Figure(13) shows audio input in magenta colour and its magnitude is about +0.01 to -0.01 V peak to peak. That ac signal is changed to DC offset 2.3V as shown in yellow colour using LM286 and voltage divider for DC input of Arduino. Blue colour represents nearly equal to 3000 Hz frequency range and green colour represents 5000 Hz. Due to threshold level comparison, significant sound signal is detected and classified frequency range as shown on that figure.

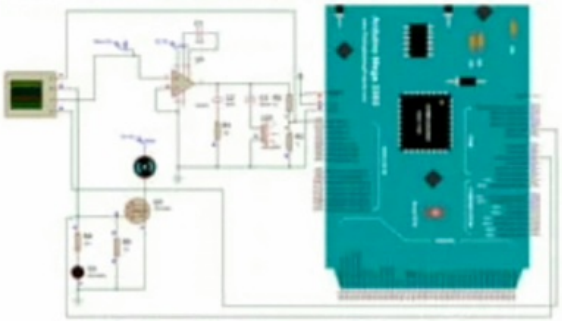

Figure12. Simulation Test using Proteus Software

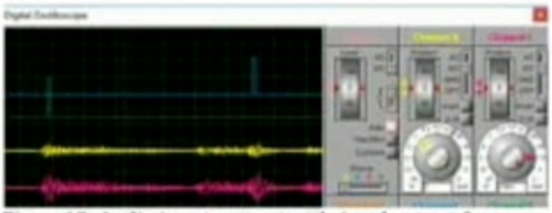

Figure13. Audio input, output and signal output for pump

Figure14. shows FFT outputs for 3.1 kHz and 7.1k kHz sine wave inputs from smart phone sound generator. The first harmonic of these signal closed to zero and it can be neglected as noise level. These noise wide as 300 Hz and it means that frequencies lower than 300 Hz cannot be specified by this system. Actually 300 Hz is generally noise level for audio signal.

But, the second harmonic goes up to 150 magnitude values and show nearly at 3 kHz and 7 kHz frequencies range. The third harmonic goes up to 80 magnitudes. So, threshold level around 90 magnitude can distinguish clearly for second harmonics.

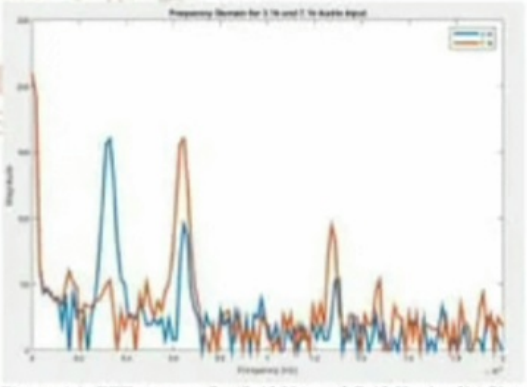

Figure14. FFT output for 3.1kHz and 7.1kHz Audio Sine Waves

Although audio signal varies 20 Hz to 20 kHz, signal over 10 kHz cannot be generated because it cannot be generated by smart phone speaker. 10 kHz is even too sharp to hear. Figure15 shows various outputs for 100 Hz to 9100 Hz sound signal. All outputs pass over 90 magnitude level.

1.3 *Physical Design*:

A pictorial representation of your project that puts your solution in context. Not necessarily restricted to your design. Include other external systems relevant to your project (e.g. if your solution connects to a phone via Bluetooth, draw a dotted line between your device and the phone). Note that this is not a block diagram and should explain how the solution is used, not a breakdown of inner components.

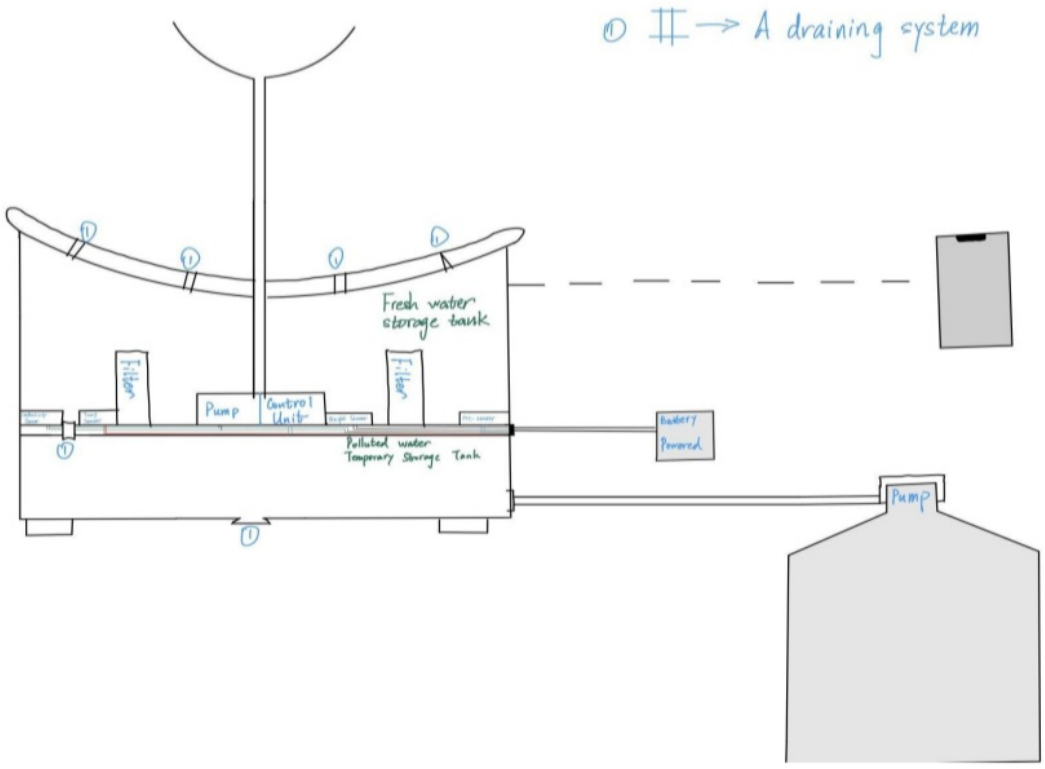

Figure 1 Smart Fountain Physical Diagram

2. Design

The block diagram below is a general design of our solution. We divide our design into four modules, including Power Supply, Control Unit, External Control, and Mechanical Unit. Details of each unit is presented in the diagram and described in the next section.

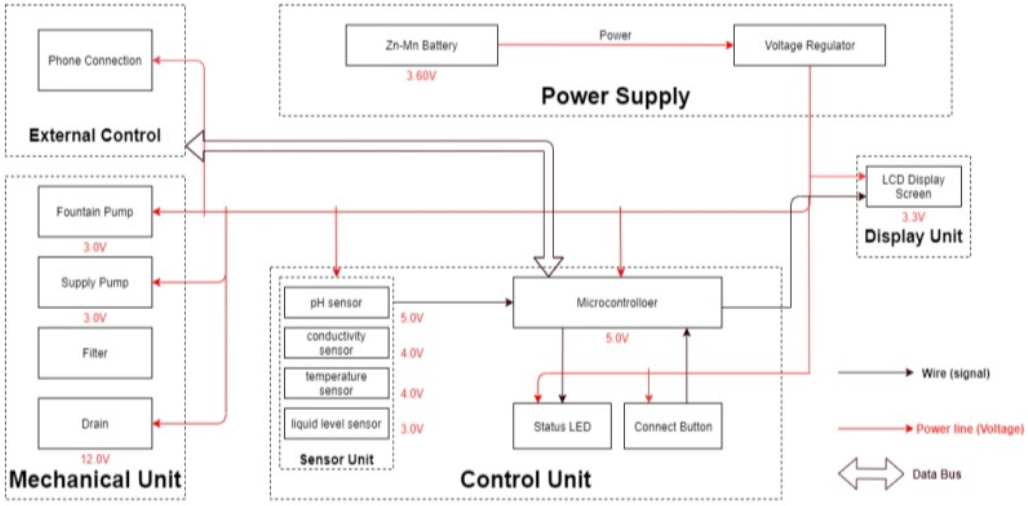

Figure 2 Block Diagram of Smart Water Fountain