MUSIC WAND

(Real-Time Optical Scanning of Music Notes)

OBJECTIVE:

Our aim is to make a device capable of assisting a novice in reading musical notes. To implement this idea we made a real time optical scanner capable with hardware capability to play the notes.

PROJECT OVERVIEW:

The Music Wand is a device that optically reads printed sheet music in real-time and synthesizes the notes which are read from the page.

The device uses a linear image sensor mounted on the end of a handheld wand to scan printed sheet music and identify the note pitches. For each note detected, a synthesized piano note is played at the detected pitch.

We chose this project in order to explore image processing and sound synthesis on the microcontroller in a creative and practical context. The concept of a music-reading wand appealed to us because it would allow a novice musician to easily learn sheet music without the help of a musical instrument.

CONCEPTS INVOLVED:

Image sensing and processing:

- **Image sensor specifications and device operation:**

We have used TSL1402R LINEAR Image sensor from Texas Advanced Optoelectric Solutions.

The photodiode data is integrated (by an opamp-capacitor integrator), and output as an analog value between 0 (black) and 255 (saturated white). The pixels are output sequentially on each clock cycle after a start pulse (SI pulse). Thus, the output of the array is a series of analog values which represent each pixel value.

SI stands for "start integration". The entire time between the SI pulse on one cycle and the SI pulse on the next cycle, the photodiode data is being integrated.

- **Usage of two clock speeds:**

We wanted a sensitivity that gave us good distinction between the black and white on the page, but did not saturate the sensor due to ambient light.

After extensive testing, we determined that there was no single clock rate that was both slow enough to run the ADC and fast enough that the pixels were not saturated by ambient light.

This forced us to run the clock at two rates. The clock rate alternates between running quickly for at least one full pixel line to set the integration time, during which the pixel data is not read, and then running much more slowly for one pixel line as the pixel data is read out. Thus, the integration time can be kept low while the data can be read at a reasonable rate.

Ideally, we want the image sensor to be focused exactly on the music. Unfortunately, what we found is that the data which we got from the image sensor was not proper due to some problems. We were not able to get correct hex values between 0-255. This made the image processing very difficult.

Amplification of signals:

We had to amplify the analog outputs which we got using an Op-amp so as to get proper sound of the musical notes detected.

Digitizing signals:

To make the sine wave, the bit values from 0-255 are required. So for each frequency of notes we had detected different bit values from the sine wave at regular intervals and stored it in array and used it as an input to the atmega which had to be finally converted to the analog values for which we needed a DAC.

Synthesising :

On finally producing the sine wave, we just had to give that output to the headphone jack, which had inbuilt noise filters, and listen to the synthesized notes.

Output devices:

Head-phone jack was the device which we had used to listen to the output of the sound signals.

LCD was also used to display the values corresponding to the analog outputs of the image sensor after ADC conversion.

HARDWARE :

1.Image sensor:

The image sensor we choose is a 256x1 pixel linear array made up of a line of 256 photodiodes. It has a 400 DPI resolution, with each pixel measuring 63.5 micrometers by 55.5 micrometers with an 8 micrometer spacing between pixels.

It requires a 5V power supply, a ground supply, and a clock at any speed between 5KHz and 8MHz.

We had to position the sensor on pcb in such a way that we would easily move it through the staff lines printed on the sheets.

Its connections were made serially and then an atmega was used to connect with the sensor which would read its analog outputs through the ADC pin.

Power Supply Considerations

For optimum device performance, power-supply lines should be decoupled by a 0.01-mF to 0.1-mF capacitor with short leads mounted close to the device package.

Direct digital synthesizer:

We needed a DAC0808 to be connected with the atmega16 to read the bit values so as to produce the sine wave. And then amplifying its analog outputs to such a level that it could be heard from the headphone.

To generate the bit values, we had written a code in C and the stored the bit values in an array to be used in the cvavr code.

The formula used to calculate the bit values was:

Bit value=127 +127 * sine (2*pi*w*1/t)

Where w=omega i.e 2*PI/T Where t is calculated by the frequency.

And this value was calculated after every delta t=0.0005 Hertz.

SOFTWARE:

Image sensor control:

All of the timing for the image sensor is controlled by timer 1. Timer 1 is set to toggle on compare match, and OCR1A changes between a low (fast) value and a higher (slow) value depending on whether an integrate cycle or a read cycle is running. The toggling output, which is the clock for the image sensor, comes out on pin D.5, and an interrupt is thrown at every compare match.

The number of integrate lines between read lines is variable and defined by `integrateCycles`. The number of successive read lines is never more than 1.

The value of the sensor clock signal is recorded at the beginning of every interrupt so that we can be sure whether we are on sending a clock high or low. To define a full line of pixels, the **variable `clockCounter` counts the number of high sensor clock values (full periods of the sensor clock) up to 257**. **On the 257th cycle, a new SI pulse can be sent** and a new line of pixel output can begin. The **SI pulse is sent simply by setting pin D.7 high for 1.5 sensor clock periods**, ensuring that at least one high clock edge coincides with the SI pulse. This happens regardless of whether we are integrating or reading, because a new line is always initiated when the previous line finishes.

<u>During integrate lines</u>:

No pixel data is read, so the code simply counts sensor clock periods until 257, sends an SI pulse, and restarts.

During the read cycle:

The pixels are read in by the ADC. The pixels are output from the sensor sequentially on each rising edge of the sensor clock.

To give the output time to settle, the ADC is set to start a conversion on the falling edge of the sensor clock. This conversion is not read until the falling edge of the next sensor clock, where it is stored in an array of pixel data to be processed (`imLine`).

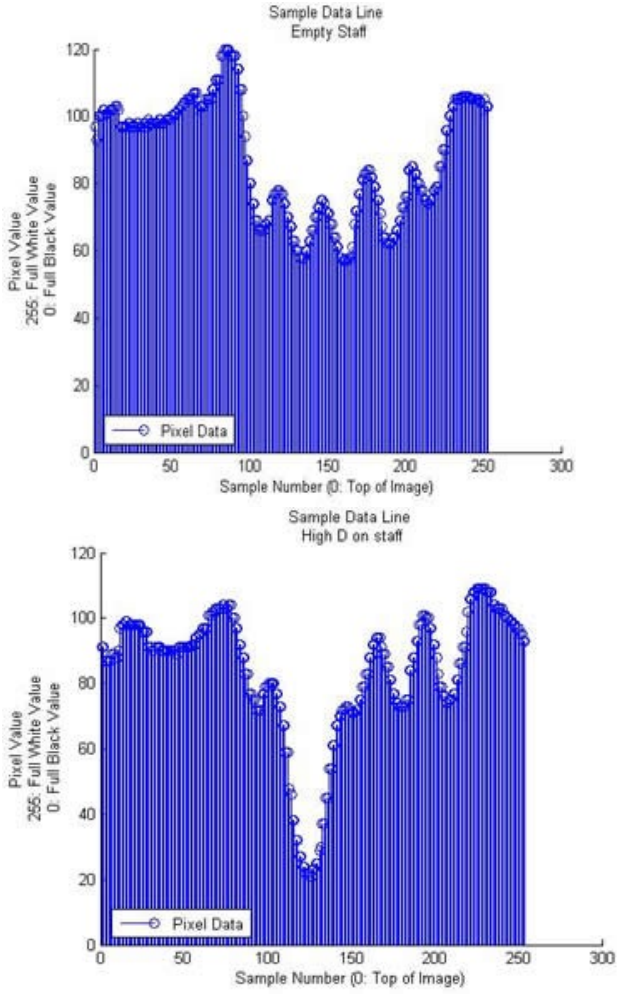
We chose to use 200 for the value of OCR1A during the integrate lines and 800 during the read line. Using 3 integration lines before reading, this gives a total time of around 45ms per read.

Image processing:

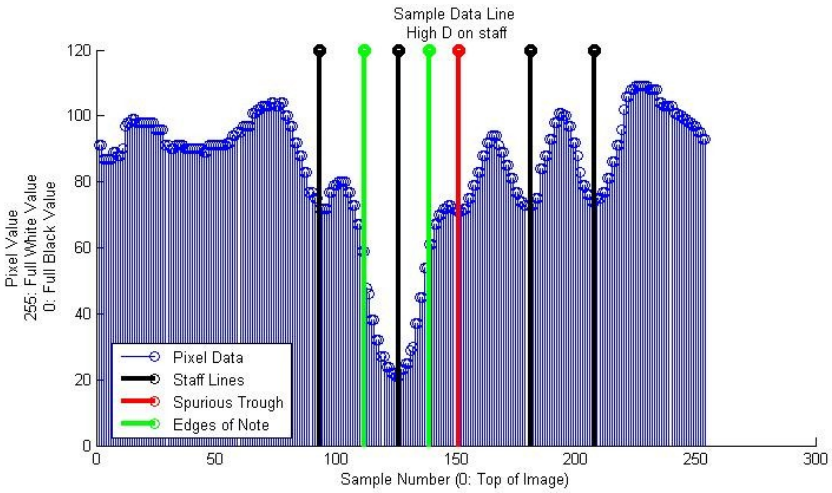
Since our "image sensor control "part was not working properly so we could not proceed but the idea which we had in our mind was to form the image data received from the image sensor in the form of a 256-element array containing integer values between 0 and 255. Analysis of a line of image data takes place in the following steps:

1. Identification of Peaks and Troughs (Peak-Picking)
2. Note Detection
3. Dynamic Quantization
4. Note Identification

1. Identification of Peaks and Troughs



For this part, basically we wrote an algorithm to find the maxima and minima in the graph formed by the values obtained from sensor. The large troughs in the image data correspond to black features in the image (such as notes and staff lines) and hence easy to distinguish between our music notes and white paper.



2.Note Detection

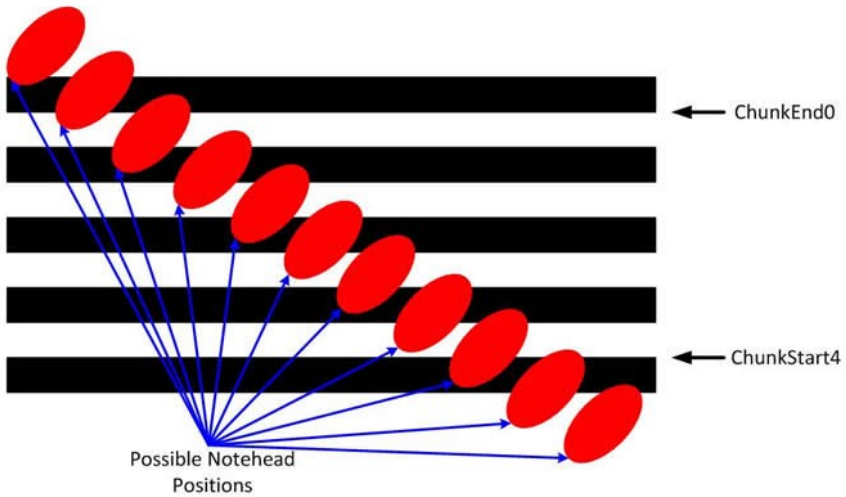
As music line and note both will correspond to full black colour we can distinguished the two by checking the area over which black is detected. Music line will be a thin black line so can be easily detected and hence we can find our note.

3. Dynamic Quantization

The next step in the algorithm is to determine the widths of the sections of the image lines which are seen as "black" by the sensor. Width of the section is meant to help in the detection of a particular note.

4. Note Identification

We confined our problem to notes which are placed on the staff, leaving us with the following set of eleven possible notehead placements:



Furthermore, we know that either `chunkStart4` contains the index of the top of the bottom staff line or `chunkEnd0` contains the index of the bottom of the top staff line. Assuming that the center of a note trough corresponds to the center of a notehead on the staff (an assumption which was confirmed by examining a large number of data sets), we can identify the note by calculating its distance from the known staff position (in terms of the width of staff lines and spaces).This marks the completion of the image processing algorithm.

Synthesizer:

Firstly our main task was to obtain the bit values for different notes which was done using a C code:

Here t1 is frequency of notes.

```
T1=1/f;
   i=0;
   delta=0.00005;
   while (t<=T1)
   {
       bit[i] = (127 + (127*sin((2*3.14*t)/T1)));
       t=t+delta;
       printf("%d,",bit[i]);
       i++;
   }
```

here is the general idea of what we had done in cvavr:

We have used Timer 0 output compare interrupt service routine.

And the OCR value was calculated as :

Ocr=(256*clock value*n*frequency of note)

Here n is the number of elements stored in the array of the bit values created for different frequencies.

Thus we had to calculate different OCR values for different notes.

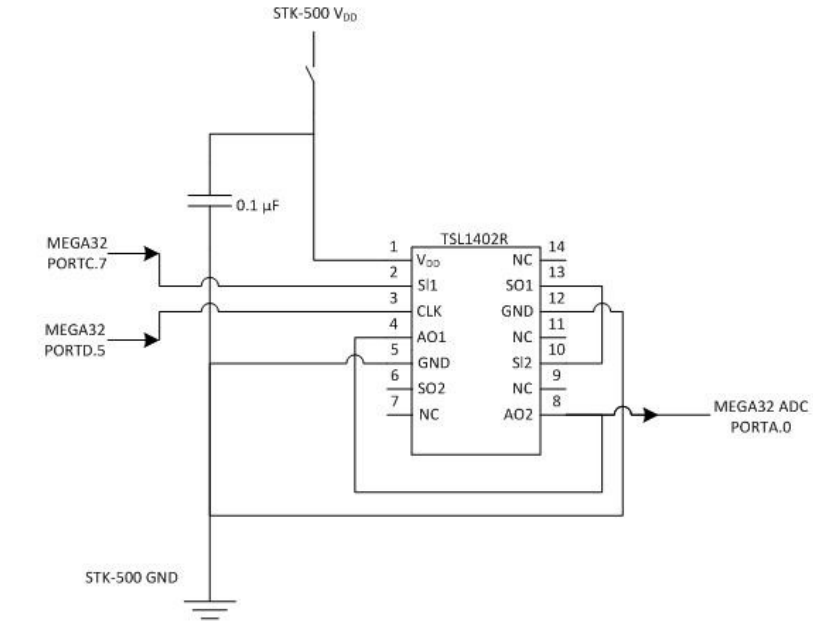
```
interrupt [TIM0_COMP] void timer0_comp_isr(void)
{
// Place your code here
    if(flag==1)
      {PORTD=bit14[i];if(bit14[i+1]==255) {i=0;k++;} else i++;}
      else
            {PORTD=bit14[i];if(bit14[i+1]==255) {i=0;k++;} else i++;}
     if(k==1000)
     {flag++;k=0;}
  TCNT0=0;
}
```

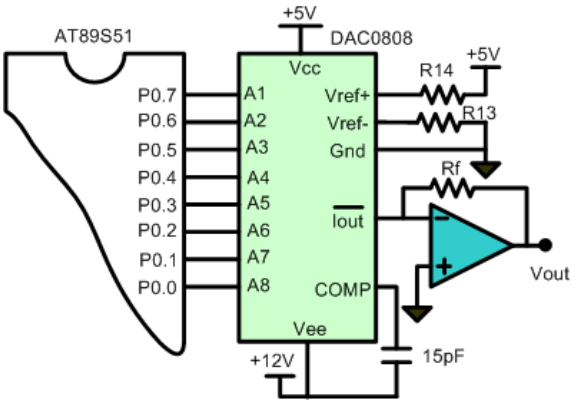
IMPLEMENTATION AND PROBLEM FACED:

We first started with the DAC part.We used port D of atmega as outputs fot 8 bit values and through DAC ,the output was sent to opamp and then to the headphone jack.We checked the output from the oscilloscope and we could get a perfect sine wave of the frequency of different octave notes.And we could get distinguishing sounds of the notes continuosly. Our synthesizer part was completely working.But there was some problem to make the DAC part work properly. Initially we were getting the signals of mega hertz frequency but then on changing the ocr values we were able to get the sounds in a very proper manner.

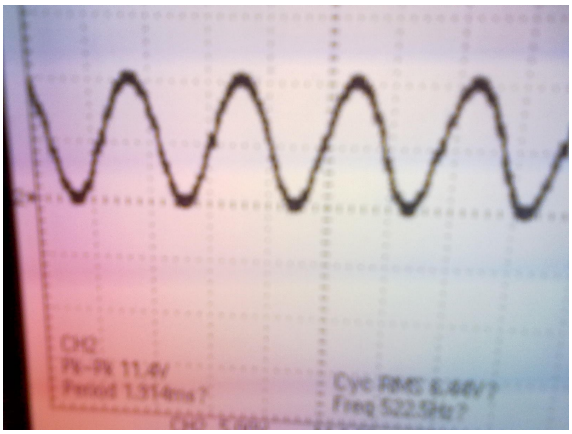
And after 30 days of our project our “Image sensor” arrived and then we started working with the image sensor part.We made the connection as given below and also used an LCD to check the outputs of the sensor using which we had to perform the image analysis. With few days left for the completion of the project we couldn’t make the sensor work properly.We are still not able to rectify the problem.

When I was printing the hex values in main() ,then I was getting only the value ‘255’ in lcd.Even on completely covering the sensor,I was getting 255 instead of a value closer to zero.We tried to debug in many ways but couldn’t find the error.But on printing the values in the interrupts(although it is not a proper way),we are getting random values between 0-255 but for a particular clock counter we are getting multiple bit values.This is where we got stuck in the project.





This is the sine wave which we have generated through the oscilloscope for 522.5 hertz frequency for a C5 musical note.



RESULTS:

Our image sensing part is not producing correct outputs but the synthesizer part to play the notes corresponding to the different frequencies of the notes is working fine. The accuracy of the music synthesis is nearly perfect.

CONCLUSIONS:

When we first envisioned the idea of an optical music-scanning sensor, we knew that the problems of image processing on the microcontroller would be difficult to solve. Furthermore, we knew that we would have to make some sacrifices with regards to the optics of the system, considering our lack of knowledge of optics and our somewhat imprecise construction. With this in mind, we set out to see if we could build a system that would perform optical note recognition and synthesis to a reasonable degree of accuracy, and we achieved the current level of performance of a synthesizer.

ACKNOWLEDGEMENT:

We are thankful to our mentor, Shadab Alam and our eclub co-ordinators :Ganesh Pitchiah, Rajat Arora and Rishabh Maheshwari for their support and for giving us time from their busy schedule and for providing us the required materials.We are thankful to Texas and the Bangalore company for helping us in providing us the image sensor.

REFERENCES:

1.TSL1402R : Linear Sensor Array | TAOS
www.taosinc.com/ProductDetails.aspx?id=4

2.http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2008/tjc42_nah35/tjc42_nah35/index.html

3. www.datasheetcatalog.com/datasheet/D/DAC0808.shtml

4 .Frequencies of the musical notes (http://en.wikipedia.org/wiki/Key_signature)

***Team Name:* INFRARED**

Members:

Anu khandwal

K.Swarna

Harsha Mulchandani

Soniya Parmar