Christopher Considine

Johan Ospina

EC 450 Final Project

4/29/2016

The MicroP-an Flute ™

**Goal:**

The goal of our project was to implement an interesting, low-cost, and power-efficient instrument that can be played by the MSP430. Our goal was to make the instrument use some unorthodox method of input, in our case, light, instead of the traditional use of wind or vibration/percussion. Additionally, we avoided purchasing expensive peripherals for our MSP430 in an attempt to keep the cost of the project low and affordable. We utilized many of the features inside the MSP430 chip to implement our pan flute, rather than rely on external peripheral gadgets. Finally we wanted to keep the power consumption of our project low, and have been able to run our project for multiple hours on just two AA batteries.

**Design:**

In designing our pan-flute style instrument we designed our approach with the goal of being able to play the entire C Major scale. This provides us with the ability to play any song that falls into this single key. Additionally, our code is highly modular and can be expanded with more MSP430 chips to implement additional scales.

Determining when to play a note was done by reading analog input from a series of photoresistors which are sensitive to light. Upon reaching a certain light threshold, the note would be triggered sending a value to one of the onboard timers, which was used to output a square wave at the desired frequency. Additionally, an LED would be lit in order to easily distinguish which notes are being played.

We implemented pulse-width modulation control in order to set the volume of our speaker’s output. The idea behind this was to output a sound at the level corresponding to how intense the light was. For instance, two notes being played at the same time, with one having twice the light intensity over its photoresistor, would also have double the output volume. This feature was set to output all notes at max volume (50% duty cycle) for our demo, but can easily be reimplemented, as the functions are still a part of the existing code.

One of our initial design goals was to link together multiple MSP430s in a Master/Slave chain using which would have the master chips reading analog input, and the slave chips in charge of playing the correct notes. That way, we could have 1 master doing multiple ADC conversions and then have the slaves solely dedicated Our final design was a simplified version of this using one chip to play 4 notes each. Using our current and final design, it would be possible to add up to two more notes per chip, but as we only implemented one scale, it made the most sense to implement 4 notes per chip, using two chips.

**Description of Implementation:**

Analog Input Conversion:

In order to determine which note to play, we utilized the onboard ADC10 module to convert the input from four photoresistors hooked up through 4 GPIO pins. These were converted when the WatchDog Timer interrupt was triggered, and were saved to an array. We read the array in reverse order, in a for loop using custom structs determined the index and magnitude of the two most intense tones. We then sent those two tones to be output to the speakers, if they met the threshold we set for light intensity.

Tone Output & Pulse-Width Modulation:

To output the desired notes, we utilized the two onboard TimerA modules to output square waves at the desired frequency. By using TA0 and TA1, we would set the value of the CCR0 registers in both to the half period and then set TA0CCR1 and TA1CCR2 to the desired duty cycle and connected those registers to their respective Port 1 and Port 2 pins. For some reason, the output on the MSP for timer A1’s CCR1 register outputted extremely low and we were forced to use A1’s CCR2’s multiplexed output pin. Finally, we made helper functions to set the duty cycle of our tones that are called inside of the logic that controls the note that is playing.

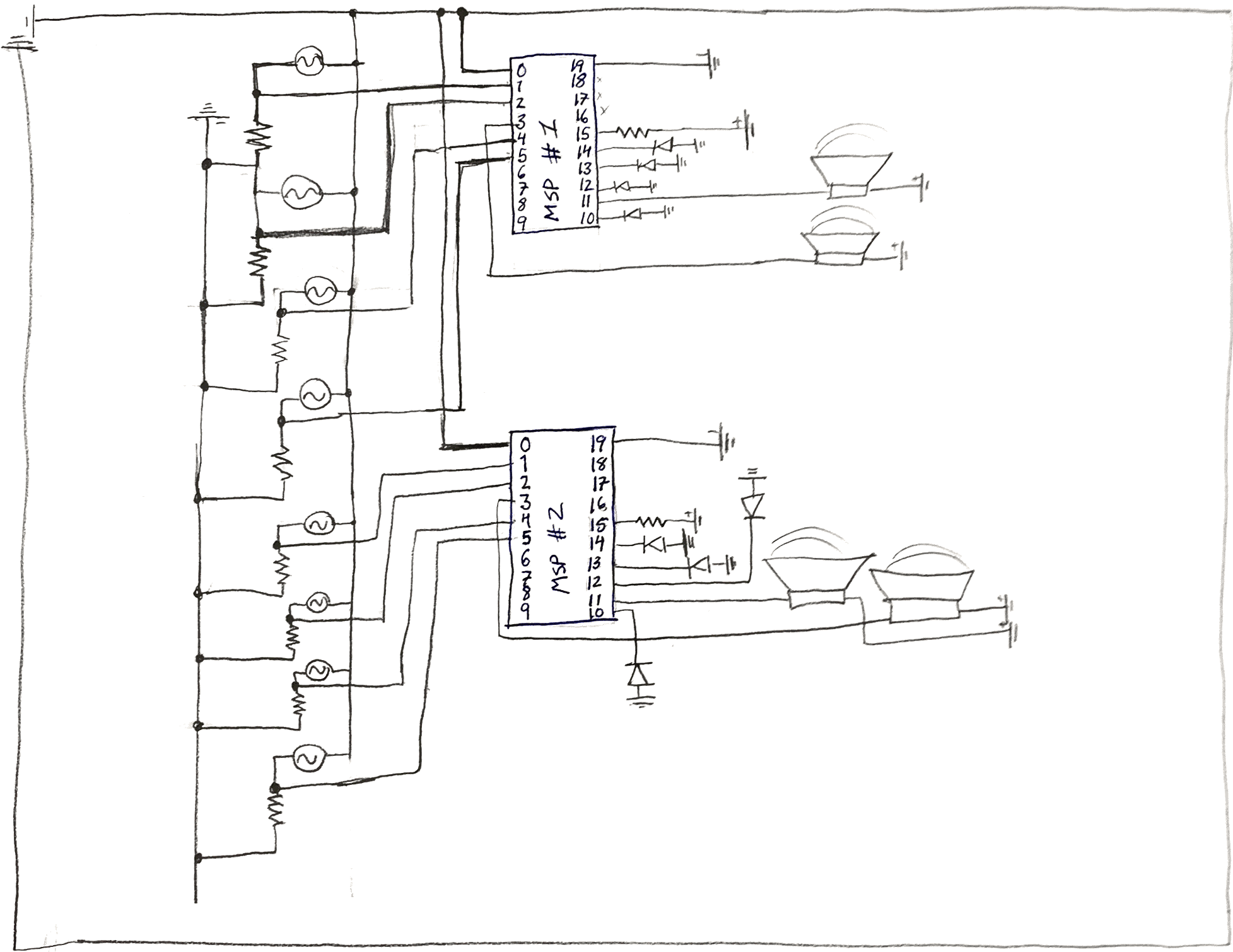
LED Output:

Because of the limit of two timer modules on each chip, our project only outputs a maximum of two notes at a time per chip, four total. In order to demonstrate not only which not was being played, but also to demonstrate that more than four notes at a time can be considered over the threshold to play, we have set the LEDs for each note to flash when its corresponding photoresistor reaches the threshold. Therein demonstrating that it is possible for the chips to realize that all notes should be playing, but only is capable of outputting the two notes with the most light intensity that are sampled by their respective microcontroller

Attempted Implementation:

We tried implementing but were unable to do so. Understanding the theory behind was not difficult, but realizing it with the hardware on the MSP430 proved to be the real challenge. We understood how to set the slave and master addresses for both transmission and reception and how the interrupt handlers were combined for transmission and reception. What we could not figure out was how to effectively drive transmission from a master to slaves. It proved daunting to do so, as we had to synchronise clock signals to mark the start of a transmission. After fidgeting around with it for two days and getting nowhere with it, we decided to scrap the protocol from the final design of our project.

**Schematic:**



**Assessment of the Success of the Project:**

This project was very successful even though we were unable to fully implement the original idea. We were able to adapt the peripherals of the MSP430 to arrive at a similar end user experience. The biggest example of this was scrapping the I2C component from the project and opting for a simpler one chip mechanism. We were able to play the notes we wanted to with ease and learned how to play joy to the world.

**Next steps you might take to make it more successful:**

To make our pan flute more effective, I recommend using photoresistors with more surface area. We ran into the issue that the light chimneys on the first 3D printed model we made, coupled with the low power gloves we bought to play our flute, ended up being too short. Something else that could be done is implement some sort of communication protocol between two MSP’s as per our original plan. This way, another way to play the notes could be devised using the port 1 pins as outputs on one MSP and then another MSP could be in charge of figuring out what notes are being selected with light. Something else we could do in the future to make this project better is add some sort of op amp circuit to amplify the amplitude of the sound we are outputting. That we we have something that can be heard by a large crowd.

**Code and Listing for the Project:**

The code for this project is attached within the zip file submitted. We broke up initializing the different peripherals that we were using into different header files. One initializes the Timer A modules on the MSP430 to output pulse width modulated waves from P1.2 and P2.4 whose frequencies are decided in the notes.h header file. Finally, the last file initializes the ADC module into the multiple conversion mode that we used to sample our photo resistors. We include all of these headers inside our two main files to call the correct initializers and enable interrupts and begin playing music.

**Summary of Team Members' Contributions to the Project:**

Johan worked on Designing and Printing the base for the flute, he also worked to try and get working but did not get anywhere for two days and thus scrapped that component. He also worked on getting the pulse width modulation working on both P1.2 and P2.4. Christopher worked on the ADC conversion, lighting up the LED’s and setting the threshold for the light intensity. He also worked on modularizing the code. Both Christopher and Johan worked on the overall design of the source code. Additionally we worked together on taking the Microprocessor off of the development board and placing it onto a breadboard and all of the circuitry of the overall system design.

**Additional Project Photos on the Following Pages:**

