Individual Report

Tim Crawford

My role in this project was creating the two songs that are in the song portion of the project. This included making functions for most of the notes and then using the notes to create each song. I also made functions to light up the LEDs when each note is played. Since my focus was creating the songs that were used in the project, I decided that my individual portions of the project would be adding more songs. There are a total of five songs that I added: Cave Story Main Theme, Mimiga Town, On to Grasstown, Meltdown 2, and Balrog's Theme. See figure 5 for all songs. Each song can be played by pressing switches 1-5 respectively when in mode 1.

All of the songs consist of a series of notes that are played for certain lengths. Each note is played by calling the note function which uses Timer 2 to play a note for a certain amount of time based on the bpm of the song and the length of the note that is being played. See figure 1. The timer in this function runs for about 250,000 machine cycles. Using the noteLength and bpm, the timer is looped the necessary number of times to play notes such as quarter or eighth. Since the bpm in the code is not equivalent to the actual bpm of the song, the bpm will need to be converted. A whole note in 120 bpm lasts for 2 seconds. The Atmega324pb executes an instruction in 0.0625 µs. This is calculated with t_{MC} = (1cc/1MC)(1s/(16x106cc)) = 0.0000000625. Taking 2/0.0625µs results in 32,000,000 machine cycles to play a whole note in 120 bpm. Dividing by the 250,000 machine cycles for the timer results in 128 for the bpm value of a 120 bpm song. This method was used to convert the bpm of each song.

```
int bpm = 128; //Note: not actual bpm of song (120bpm)

//Measure 1
Note(bpm, 0.25, NoteDSharp, "D#, ");
Note(bpm, 0.25, NoteHighGSharp, "HG#, ");
Note(bpm, 0.25, NoteDSharp, "D#, ");
Note(bpm, 0.25, NoteHighGSharp, "HG#, ");

//Play a specified note for the length of the note
void Note(int bpm, float noteLength, void (*NoteToPlay)(), char* note)
{
        USART_TxString(note);
        for(int i = 0; i < bpm*noteLength; i++)
        {
        TCNT2 = -244; //Initialize timer to ~250,000MCs (1024 prescaling)
        while (!(TIFR2 & (1<<TOV2)))
        {
                NoteToPlay();
        }
        TIFR2 = 1<<TOV2;
        }
}
```

Figure 1: The Note function along with the first 4 notes of Cave Story Main Theme

As Timer 2 is counting the length of the note in the Note function, another function is called based on the given parameter that will play the frequency of the desired note using the speaker and light up the corresponding LED. Every note has its own function that will send the frequency of the note to the speaker and light up the note’s LED(s). See figure 2. Sharp notes will light up the LED of the corresponding note as well as the note above it in the scale because the sharps are between notes. Note G will light up LED 1 when played, A lights up LED2, B lights up LED6, C lights up LED5, D lights up LED7, E lights up LED 8, and F lights up LED 9. The function will then call either the NoteTimer or NoteTimer256 function which generates the frequency required to play the note.

```
void NoteA()
{
	LEDA();
	NoteTimer(-35);
	LEDA();
}

void NoteASharp()
{
	LEDA();
	LEDB();
	NoteTimer256(-134);
	LEDA();
	LEDB();
}
```

```
void LEDA()
{
	PORTD ^= 0b00000010;
}

void LEDB()
{
	PORTD ^= 0b00010000;
}
```

Figure 2: The functions for an A and an A# note (left) along with the LED functions for both (right).

The timer functions are given a wait value by every note’s function that determines how long the timer will wait before toggling PE4 which is connected to the speaker to generate the desired frequency. Both of these functions use Timer 0. NoteTimer has 1024 prescaling while NoteTimer256 has 256 prescaling. The reason for having two different functions to generate sound is because the higher frequencies required a more precise wait which could not be accomplished with 1024 prescaling. Likewise, 256 prescaling can’t be given a large enough wait to generate low frequencies.

```
// Timer with 1024 prescaling
void NoteTimer(int wait)
```

```
{
	TCNT0 = wait;
	TCCR0A = 0x00;
	TCCR0B = 0b00000101;
	while (!(TIFR0 & (1<<TOV0)));
	PORTE ^= 0b00010000;
	TIFR0 = 1<<TOV0;
}

// Timer with 256 prescaling
void NoteTimer256(int wait)
{
	TCNT0 = wait;
	TCCR0A = 0x00;
	TCCR0B = 0b00000100;
	while (!(TIFR0 & (1<<TOV0)));
	PORTE ^= 0b00010000;
	TIFR0 = 1<<TOV0;
}
```

Figure 3: The timer functions that generate the frequency of each note.

Since so many notes were used in this project, I wrote a python script to generate the wait values for both timers given a frequency. See figure 4. A calculation was included for 64 prescaling, but not notes ever had a wait of more than -255 so it was unused. This script would wait for a frequency to be input. The number of machine cycles required to produce the sound was then found to be MCs = ((1/frequency)/2)/0.0000000625. (1/frequency)/2) is used to get the half period of the desired sound wave. 0.0000000625 is the number of seconds it takes the Atmega324pb to execute an instruction. MCs is then divided by the three different prescale values. The frequency of an A note is 220Hz. Entering this in the script results in the following values: 35.51136363636363 for 1024, 142.04545454545453 for 256, and 568.1818181818181 for 64. Since 256 prescaling is the closest to an integer, -142 would be used as the timer wait for the A note along with 256 prescaler for Timer 1. This is not the case is the project because I did the calculation for the A note by hand using 1024 as the prescale value before I made the script. Most of the other wait values were found using the script and the above calculations.

```
while(1):
    frequency = float(input("Enter Number: "))
    MCs = ((1/frequency)/2)/0.0000000625
    print("1024: " + str(MCs/1024))
    print("256: " + str(MCs/256))
    print("64: " + str(MCs/64))
```

Figure 4: Python script to convert frequency to the wait value for Timer 1.

```
//Cave Story Main Theme
if(~PINA & 0b00000001)
{
                int bpm = 128; //Note: not actual bpm of song (120bpm)
                Display_Menu(mode, "Cave Story Main Theme");

                //Measure 1
                Note(bpm, 0.25, NoteDSharp, "D#, ");
                Note(bpm, 0.25, NoteHighGSharp, "HG#, ");
                Note(bpm, 0.25, NoteDSharp, "D#, ");
                Note(bpm, 0.25, NoteHighGSharp, "HG#, ");

                //Measure 2
                Note(bpm, 0.25, NoteD, "D, ");
                Note(bpm, 0.25, NoteHighGSharp, "HG#, ");
                Note(bpm, 0.25, NoteD, "D, ");
                Note(bpm, 0.25, NoteHighGSharp, "HG#, ");

                //Measure 3
                Note(bpm, 0.25, NoteCSharp, "C#, ");
                Note(bpm, 0.25, NoteHighGSharp, "HG#, ");
                Note(bpm, 0.25, NoteCSharp, "C#, ");
                Note(bpm, 0.25, NoteHighGSharp, "HG#, ");

                //Measure 4
                Note(bpm, 0.25, NoteC, "C, ");
                Note(bpm, 0.25, NoteHighGSharp, "HG#, ");
                Note(bpm, 0.25, NoteC, "C, ");
                Note(bpm, 0.125, NoteCSharp, "C#, ");
                Note(bpm, 0.125, NoteD, "D, ");
}

//Mimiga Town
if(~PINA & 0b00000010)
{
                int bpm = 140; //110 bpm
                Display_Menu(mode, "Mimiga Town");

                //Measure 1
                Note(bpm, 0.25, NoteLowF, "LF, ");
                Note(bpm, 0.25, NoteA, "A, ");
                Note(bpm, 0.125, NoteG, "G, ");
                Note(bpm, 0.25, NoteLowF, "LF, ");
                Note(bpm, 0.125, NoteLowF, "LF, ");

                //Measure 2
                Note(bpm, 0.25, NoteC, "C, ");
                Note(bpm, 0.125, NoteA, "A, ");
                Note(bpm, 0.375, NoteC, "C, ");
                Note(bpm, 0.25, NoteC, "C, ");

                //Measure 3
                Note(bpm, 0.25, NoteD, "D, ");
                Note(bpm, 0.25, NoteC, "C, ");
                Note(bpm, 0.25, NoteF, "F, ");
```

```
		Note(bpm, 0.25, NoteC, "C, ");

		//Measure 4
		Note(bpm, 0.75, NoteASharp, "A#, ");
}

//On to Grasstown
if(~PINA & 0b00000100)
{
		int bpm = 96; //160 bpm
		Display_Menu(mode, "Grasstown");

		//Measure 1
		Note(bpm, 0.0625, NoteG, "G, ");
		Note(bpm, 0.0625, Rest, "");
		Note(bpm, 0.125, NoteLowC, "LC, ");
		Note(bpm, 0.125, Rest, "");
		Note(bpm, 0.125, NoteG, "G, ");
		Note(bpm, 0.0625, NoteA, "A, ");
		Note(bpm, 0.0625, Rest, "");
		Note(bpm, 0.125, NoteG, "G, ");
		Note(bpm, 0.125, Rest, "");
		Note(bpm, 0.125, NoteLowC, "LC, ");

		//Measure 2
		Note(bpm, 0.1875, NoteG, "G, ");
		Note(bpm, 0.1875, NoteLowF, "LF, ");
		Note(bpm, 0.125, NoteLowE, "LE, ");
		Note(bpm, 0.125, Rest, "");
		Note(bpm, 0.125, NoteLowE, "LE, ");
		Note(bpm, 0.125, NoteLowD, "LD, ");
		Note(bpm, 0.125, NoteLowE, "LE, ");

		//Measure 3
		Note(bpm, 0.375, NoteLowF, "LF, ");
		Note(bpm, 0.0625, NoteLowF, "LF, ");
		Note(bpm, 0.0625, NoteLowE, "LE, ");
		Note(bpm, 0.375, NoteLowD, "LD, ");
		Note(bpm, 0.0625, NoteLowF, "LF, ");
		Note(bpm, 0.0625, NoteLowE, "LE, ");

		//Measure 4
		Note(bpm, 1, NoteLowD, "LD, ");
}

//Meltdown 2
if(~PINA & 0b00001000)
{
		int bpm = 110; //140 bpm
		Display_Menu(mode, "Meltdown 2");

		//Measure 21
		Note(bpm, 0.1875, NoteHighG, "HG, ");
		Note(bpm, 0.0625, NoteFSharp, "F#, ");
		Note(bpm, 0.1875, NoteHighG, "HG, ");
		Note(bpm, 0.0625, NoteFSharp, "F#, ");
		Note(bpm, 0.125, NoteHighG, "HG, ");
		Note(bpm, 0.0625, NoteE, "E, ");
		Note(bpm, 0.0625, NoteFSharp, "F#, ");
		Note(bpm, 0.125, NoteHighG, "HG, ");
		Note(bpm, 0.125, NoteHighA, "HA, ");
```

```
                    //Measure 22
                    Note(bpm, 0.125, NoteHighB, "HB, ");
                    Note(bpm, 0.125, NoteHighC, "HC, ");
                    Note(bpm, 0.375, NoteHighD, "HD, ");
                    Note(bpm, 0.0625, NoteHighC, "HC, ");
                    Note(bpm, 0.0625, NoteHighB, "HB, ");
                    Note(bpm, 0.125, NoteHighA, "HA, ");
                    Note(bpm, 0.125, NoteFSharp, "F#, ");

                    //Measure 23
                    Note(bpm, 0.1875, NoteHighA, "HA, ");
                    Note(bpm, 0.0625, NoteHighG, "HG, ");
                    Note(bpm, 0.1875, NoteHighA, "HA, ");
                    Note(bpm, 0.0625, NoteHighG, "HG, ");
                    Note(bpm, 0.125, NoteFSharp, "F#, ");
                    Note(bpm, 0.125, NoteE, "E, ");
                    Note(bpm, 0.125, NoteD, "D, ");
                    Note(bpm, 0.125, NoteE, "E, ");

                    //Measure 24
                    Note(bpm, 0.0625, NoteD, "D, ");
                    Note(bpm, 0.0625, NoteE, "E, ");
                    Note(bpm, 0.875, NoteD, "D, ");
}

//Balrog's Theme
if (~PINE & 0b01000000)
{
                    int bpm = 106; //145 bpm
                    Display_Menu(mode, "Balrog's Theme");

                    //Measure 1
                    Note(bpm, 0.0625, NoteLowC, "LC, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowC, "LC, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowFSharp, "LF#, ");
                    Note(bpm, 0.1875, Rest, "");
                    Note(bpm, 0.0625, NoteLowF, "LF, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowF, "LF, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowFSharp, "LF#, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteG, "G, ");
                    Note(bpm, 0.0625, Rest, "");

                    //Measure 2
                    Note(bpm, 0.0625, NoteG, "G, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteG, "G, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowFSharp, "LF#, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowF, "LF, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowDSharp, "LD#, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowDSharp, "LD#, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.1875, NoteLowC, "LC, ");
                    Note(bpm, 0.0625, NoteLowB, "LB, ");

                    //Measure 3
                    Note(bpm, 0.0625, NoteLowASharp, "LA#, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowASharp, "LA#, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowF, "LF, ");
                    Note(bpm, 0.1875, Rest, "");
                    Note(bpm, 0.0625, NoteLowE, "LE, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowE, "LE, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowF, "LF, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowFSharp, "LF#, ");
                    Note(bpm, 0.0625, Rest, "");

                    //Measure 4
                    Note(bpm, 0.0625, NoteG, "G, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowFSharp, "LF#, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowF, "LF, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowF, "LF, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowDSharp, "LD#, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.0625, NoteLowDSharp, "LD#, ");
                    Note(bpm, 0.0625, Rest, "");
                    Note(bpm, 0.125, NoteLowC, "LC, ");
                    Note(bpm, 0.0625, Rest, "");
}
```

Figure 5: The five songs that are in the project.