Patrick Austin

CPE 301 – 1104

Assignment # 10

11/27/2016

Assignment description:

In this lab we recreated the circuit and program from lab 7 to create a digital keyboard that produced a tone corresponding to a given note when a matching serial input was given from the PC keyboard. Though this program serves the same purpose as the one from lab 7, this time the operation of toggling the output pin to generate the tone and restarting the timer is handled via a timer overflow interrupt, and not just in loop via the timer overflow flag as happened previously.

Problems encountered:

We were already acquainted with the circuit, the idea of the program, and some of the pitfalls of the project from lab 7, so things moved fairly smoothly. We still had a few difficulties owing to bugs that occurred when porting our lab 7 code into this project, some of which had even the TAs scratching their heads. One of these involved using the volatile keyword with the global variables used by the ISR, one involved a convoluted logic error in the else-if block used to generate the timer preload value (since excised), and one involved what may have been a PC-specific UART issue involving duplicate/ghost inputs. With a little persistence we were able to generate code that satisfied the objective and functioned more or less identically, as far as we could determine, to the code produced in lab 7 on the end-user side.

Lessons learned:

Aside from experience gained running down the bugs described above, this lab gave a lot of review of UART basics, and dealing with UART temperamental-ness. It also gave some review of the timer and generating square waves with specific parameters, this time with the extra modifier of using the timer overflow interrupt for the toggle and the timer restart. Using the ISR for this purpose was itself a review, and a good chance to iterate on the homework concerning interrupts.

Description of completed lab:

We were able to verify generation of specified notes. Regular notes are implemented with lower case inputs form the keyboard and sharps with upper case (ie ‘a’ for the note A, ‘A’ on the keyboard for the note A sharp).

Here is the specified code (extra credit implemented):

//Patrick Austin

//CPE 301 Lab 10

//Revision Number 1

//Revision date: 11/21/2016

//hardware pointer declarations

//GPIO

volatile unsigned char\* myPortDDRB = (unsigned char\*) 0x24;

volatile unsigned char\* myPortB = (unsigned char\*) 0x25;

//timer

volatile unsigned char\* myTIFR1 = (unsigned char\*) 0x36;

volatile unsigned char\* myTCC1A = (unsigned char\*) 0x80;

volatile unsigned char\* myTCC1B = (unsigned char\*) 0x81;

volatile unsigned char\* myTCC1C = (unsigned char\*) 0x82;

volatile unsigned int\* myTCNT1 = (unsigned int\*) 0x84;

volatile unsigned char\* myTIMSK1 = (unsigned char\*) 0x6F;

//UART

volatile unsigned char \*myUCSR0A = (unsigned char \*)0x00C0;

volatile unsigned char \*myUCSR0B = (unsigned char \*)0x00C1;

volatile unsigned char \*myUCSR0C = (unsigned char \*)0x00C2;

volatile unsigned int \*myUBRR0 = (unsigned int \*) 0x00C4;

volatile unsigned char \*myUDR0 = (unsigned char \*)0x00C6;

//interrupt

volatile unsigned char\* mySREG = (unsigned char\*) 0x5F;

//global variables for storing serial input data/ISR interaction

volatile byte byteRead; //will store keyboard input received via UART

volatile unsigned int timerCount = 0; //will store timer preload value for desired note. set in setup, updated in loop

volatile unsigned int timerMode = 0x01; //stores 0x01 for 1 prescaler/normal mode

//ISR

ISR(TIMER1\_OVF\_vect)

{

\*myTCC1B = 0; //turn off the timer

\*myPortB = \*myPortB ^ 0x40; //toggle output pin

\*myTCNT1 = timerCount; //reset the preload value

\*myTCC1B = timerMode; //enable timer with 1 prescaler

}

//setup & loop

void setup()

{

\*myPortDDRB = \*myPortDDRB | 0x40; //enable output on specified pin, pin 6

\*myUCSR0A = 0x22; //intitialize UART at 112500 baud

\*myUCSR0B = 0x18;

\*myUCSR0C = 0x06;

\*myUBRR0 = 16;

\*myTCC1A = 0; //zero out timer controls to initialize for normal mode

\*myTCC1B = 0;

\*myTCC1C = 0;

\*myTIMSK1 = 0x01; //enable TOV1 interrupt

\*myTCNT1 = 300000; //arbitrarily initialize timer to start

timerCount = 300000;

\*myTCC1B = timerMode; //turn on the timer

}

void loop()

{

if (U0kbhit()) //if serial data has been sent...

{

byteRead = U0getchar(); //store the data

U0putchar(byteRead); //echo the data to console

timerCount = 65536 - getDelay(byteRead); //update the time between interrupts

}

}

//function implementations

unsigned int getDelay ( byte input ) //for a char corresponding to a note (convention: regular notes lower case,

{ //sharps upper case), return an appropriate timer count value for 1/2 the period

if ( input == 'a' ) //to generate a 440hz square wave, go high 1/2 period, low 1/2 period.

//For example 1/440 = 2.27272, half that is 1.1363, so want approx 1136 microsec delay.

//So 1136 / .0625 = 18176 should be loaded for use with the 1 prescaler

{

return 18176;

}

else if (input == 'A' ) //etc.

return 17152;

else if (input == 'b' )

return 16192;

else if (input == 'c' )

return 15296;

else if (input == 'C' )

return 14448;

else if (input == 'd' )

return 13616;

else if (input == 'D' )

return 12816;

else if (input == 'e' )

return 12144;

else if (input == 'f' )

return 11456;

else if (input == 'F' )

return 10816;

else if (input == 'g' )

return 10688;

else if (input == 'G' )

return 9632;

}

unsigned char U0kbhit() //kbhit, getchar, putchar as used in previous labs & hw

{

return (\*myUCSR0A & 0x80);

}

unsigned char U0getchar()

{

return (\*myUDR0);

}

void U0putchar(unsigned char U0pdata)

{

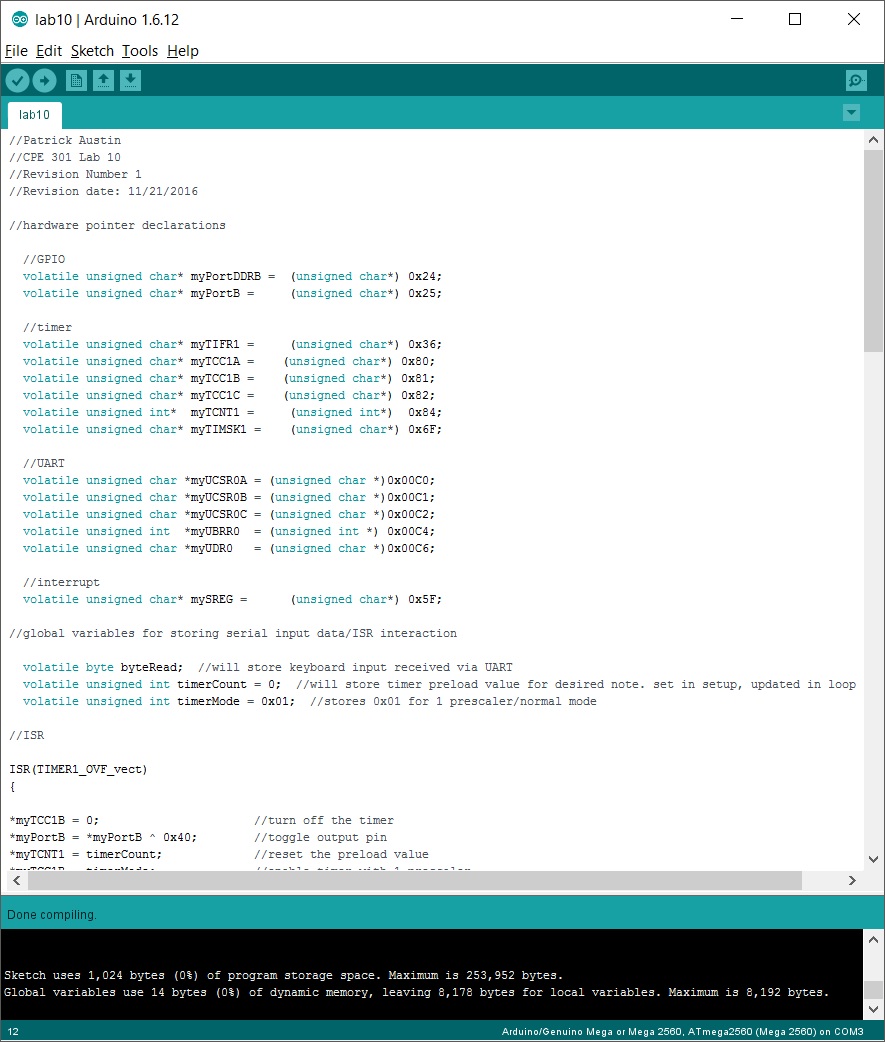
while ( (\*myUCSR0A & 0x20) == 0 )

{}

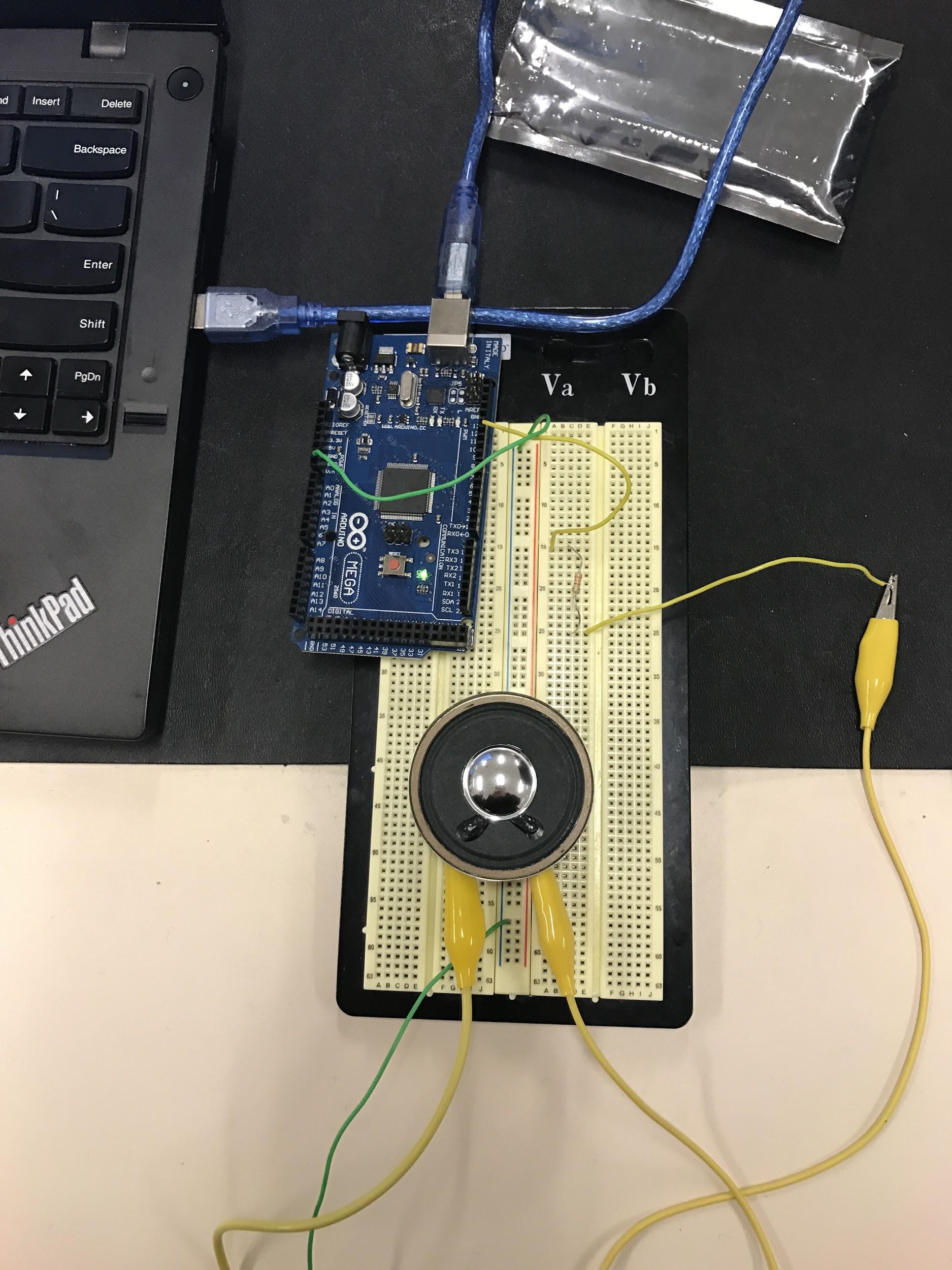
\*myUDR0 = U0pdata;

}

Here is the specified code compiling:



Here is the completed circuit:



Problem 3 question: Using the signal generator to replicate a note, we observed that corresponding sounds were different from what the speaker connected to the board produced. And moreover the sound changed when the generated wave was switched between square, sine, and jagged waveforms. This result made sense. A sine wave is a very different shape than a square wave, producing many more voltage levels. Moving smoothly between a voltage max and min will sensibly produce a different sound than a harsh square wave fluctuation between only the minimum and maximum voltage.