Abschlussprojekt Piano

Systemnahe Programmierung

Project goals:

The goal of the project based on the conditions for the project. The conditions were rather loose and didn't expect anything than the functional implementation of our choosing. At first, we decided to implement a thermometer as we weren't experienced at all with creating projects for microcontrollers. However, when we first started discussing the project during brainstorming, we quickly realized that the piano might be a much more creative approach. We had a lot of ideas and got excited to accept the challenge. The goal we set included playing 16 different tones with 16 pins by buttons (Button settings not included) and 1 pin connected with a speaker. With the ability to play single tones, we also wanted the microcontroller to play a given melody. The melody should be chosen randomly.

Description of the solution:

The solution of the piano project works with the PIC18F4520 microcontroller, using 17 pins all together. The pin settings were initialized by using the PINTRIS command, a better short form saving more code expressions as a result.
In the beginning we redefined several names, making work with the Pins and tones easier. Therefore, we added two definition lists.
We used 16 pins as input signals, connected to the buttons which are controlled by the user. The pins we have been using, were RB0-RB7 and RD0-RD7. The buttons were implemented with polling. By pressing one of the buttons the change in charge is noticed and transfers the charge to the connected pin, setting it to 1.
The code of the microcontroller reacts to the change in the active pin and plays the function playTone().

The playTone() function plays the chosen tone by sending an output signal. To create a tone, we first start calculating the time needed the pin to be active, by dividing 1 by tone frequency, divided by 2 and multiplying the microseconds by 1.000.000 into milliseconds. The result equals the amount of time the pin needs to be active to create a tone in the speaker. After activating the pin for the given time, the pin will be turned inactive for the same amount of time, to avoid accumulating different tones.
The output is sent by RC0, being connected to the speaker. The speaker works with an active Power source and starts playing once the output pin becomes active.
The melody function were realized by creating a given array melody[]. The melody should then be played by iterating over the array, calling the playTone() function each time. However, several attempts using different strategies or functions failed playing a fix tone pattern by the speaker. The two melodies were Happy Birthday and Boogie Woogie.

Results (Screenshots, Diagrams, Prints):

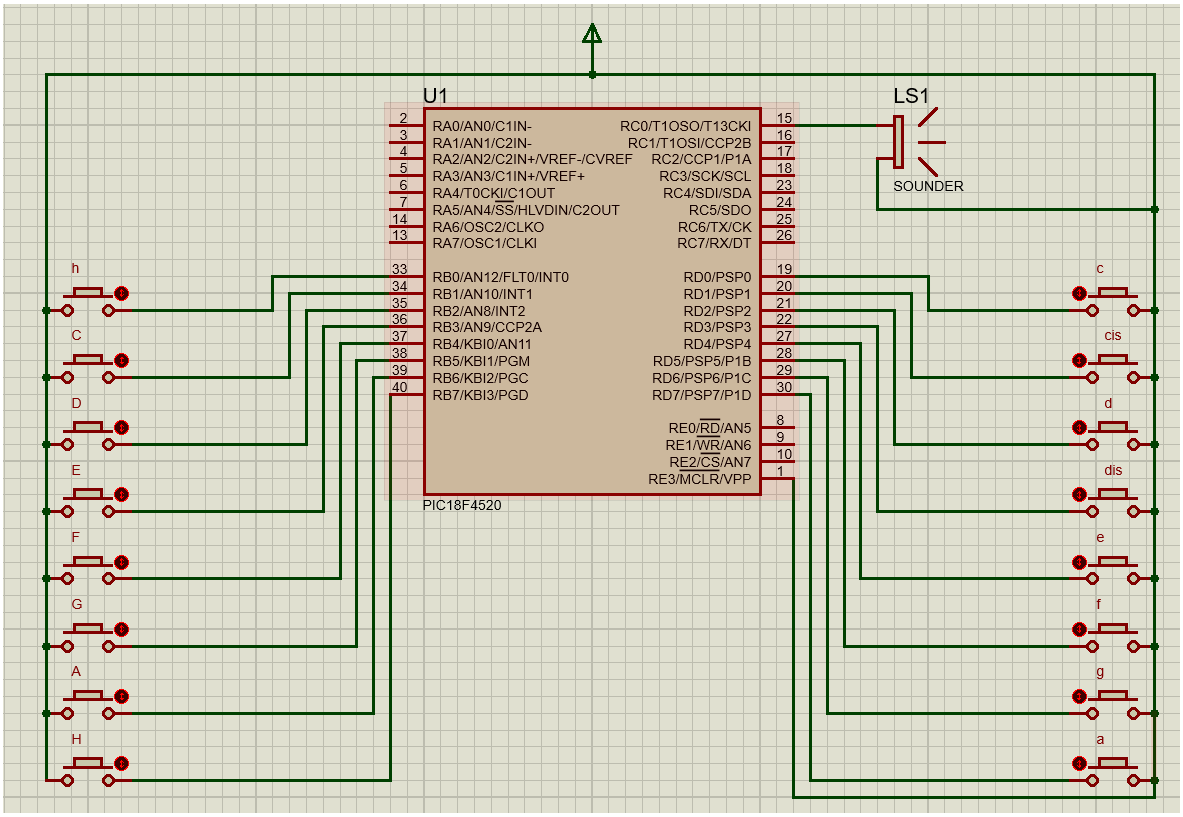
```
// Set DPins, BPins and CPin.
// Inputs.
TRISD = 0b11111111;
TRISB = 0b11111111;

//Same as above but longer
TRISDbits.RD0 = 1;
TRISDbits.RD1 = 1;
TRISDbits.RD2 = 1;
TRISDbits.RD3 = 1;
TRISDbits.RD4 = 1;
TRISDbits.RD5 = 1;
TRISDbits.RD6 = 1;
TRISDbits.RD7 = 1;
TRISBbits.RB0 = 1;
TRISBbits.RB0 = 1;
TRISBbits.RB1 = 1;
TRISBbits.RB2 = 1;
TRISBbits.RB3 = 1;
TRISBbits.RB4 = 1;
TRISBbits.RB5 = 1;
TRISBbits.RB6 = 1;
TRISBbits.RB7 = 1;

// Ouput
TRISCbits.RC0 = 0;
```

The results were mostly enjoyable: Playing single tones by user input were successful. We were able to test our solution with Proteus to make sure, that every note is played correctly. Pressing the button shortly allows playing short tones, while keeping the button pressed keeps playing the tone longer.
Most of the implemented code worked on the first hand, where playing the tones with the simulation showed that changing some pins were necessary, as most of buttons originally connected to RA0-RA7 had major issues getting used by the controller. We still don’t know what caused the issue, but changing to RD0-RD7 fixed the problem magically.

We also decided to implement given melodies by their notes, but unfortunately, we had trouble realizing playing the chosen melodies. The tones weren't played correctly and even after several attempts using Nop(), Delay_ms(), changing the playTone() function, it still didn't work as it was supposed to do.
Short side note: Interestingly, some C# expressions didn't work out and weren't accepted by the MPLab.



Conclusion:

Because we have never worked with microcontrollers in our lives and were not familiar with the subject, we were a little skeptical at the beginning whether we can implement the project as we imagined it. Some C# methods did not work as expected, so sometimes we had to improvise and find other solutions.
But after some research on the Internet, we were able to implement the project as we imagined it. Almost all of our goals were achieved and in general we are proud of what we were able to achieve.

References / Literature:

Microchip PIC18F4520 Data Sheet
06b_Button_Interrupts.PDF
https://www.wolframalpha.com
https://www.intmath.com/trigonometric-graphs/music.php
http://picguides.com/beginner/digital.php