Digital and analogue project

(Device controller with using head movement)

Author : Jisoo Goo- gjs10133@gmail.com

Teacher : Bertil Lindvall

Table of contents

1. Abstract

It is a music player, which can be controlled by head movement or buttons. First, Display will show which song is now playing. And you can change songs just by moving your head right or left. If you don't want to use your head movement you can use 2 buttons instead. Additionally, LED will let you know the state of the player. When LED is on, it means the controller works well and display will change.

2. Construction

Before making the player, I had to decide components, the part of the player.

- Microcontroller
- display to show the song
- accelerometer to recognize head movements.
- 2 buttons which have same function as head movements
- LED to show the state of the player

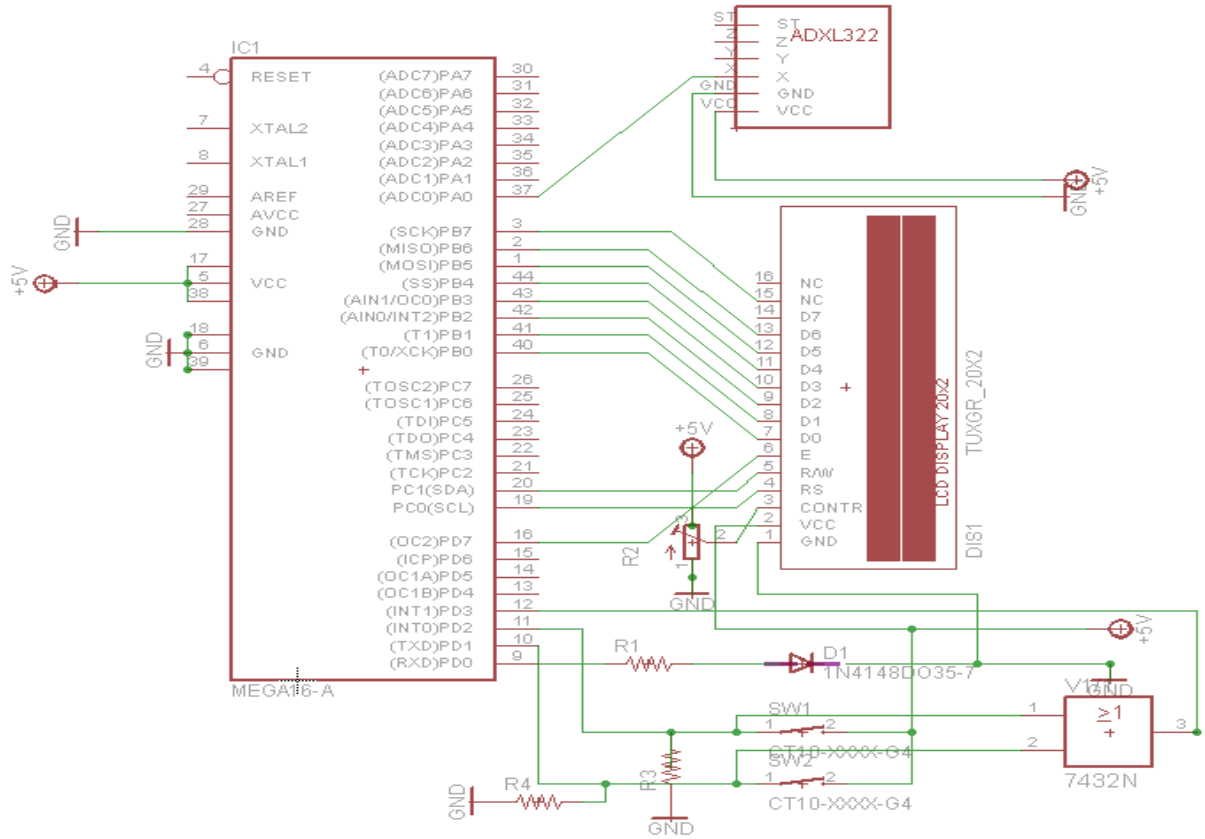
After this, I built the schematics to connect each port.

- Port A : connected the output of accelerometer. In this port, analogue signal will be converted into the digital signal.
- Port B : connected with the Data bit of display.
- Port C : C0 and C1 is connected to the 'Rs' and 'Read Write' of the display. from C2 to C7 are connected to JTAG.
- Port D : D0 is connected to the LED and D1,D2 are connected to the switches. To use interrupt, I put OR gate with D1 and D2. After that I connected the output of the OR gate to the D3. D7 is connected to the 'enable' of the display.

And then I started to build the hardware of the player.

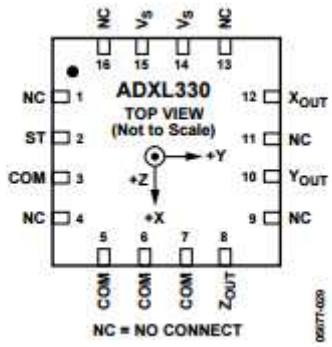
3. Hardware

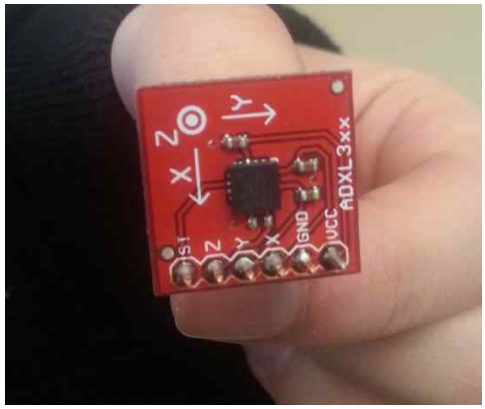
*schematics



3-1. Accelerometer

Accelerometer is a sensor that is typically used to measure the acceleration forces. These forces may be static, like constant force of gravity pulling at your feet, or they could be dynamic-caused by moving or vibrating the accelerometer. Accelerometer which I used has 3-axis: X-axis, Y-axis, Z-axis. But I used only X-axis. It has a sensitivity of ± 3g. Sensitivity is a ratio of change in acceleration to change in the output signal. It also has a sleep mode to reduce power consumption when not in use. So this accelerometer recognizes your head movement which controls the music player.





Pin No.	Mnemonic	Description
1	NC	No Connect
2	ST	Self-Test
3	COM	Common
4	NC	No Connect
5	COM	Common
6	COM	Common
7	COM	Common
8	Z_{OUT}	Z Channel Output
9	NC	No Connect
10	Y_{OUT}	Y Channel Output
11	NC	No Connect
12	X_{OUT}	X Channel Output
13	NC	No Connect
14	V_S	Supply Voltage (2.0 V to 3.6 V)
15	V_S	Supply Voltage (2.0 V to 3.6 V)
16	NC	No Connect

<accelerometer pin function>

In general, the signal (or data) acquisition process has 3 steps.

-In the Real World, a sensor senses any physical parameter and converts into an equivalent analog electrical signal.
-For efficient and ease of signal processing, this analog signal is converted into a digital signal using an Analog to Digital Converter (ADC).
-This digital signal is then fed to the Microcontroller (MCU) and is processed accordingly.

And the port A contains the ADC pins. That's why I connected accelerometer to A0.

*ADC prescaler

The ADC of the AVR converts analog signal into digital signal at some regular interval. This interval is determined by the clock frequency. In general, the ADC operates within a frequency range of 50kHz to 200kHz. But the CPU clock frequency is much higher (in the order of MHz). So to achieve it, frequency division must take place. The prescaler acts as this division factor. It produces desired frequency from the external higher frequency. There are some predefined division factors – 2, 4, 8, 16, 32, 64, and 128. And I chose 128, you can see at the software.

*ADC register

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

The ADC needs a reference voltage to work upon. For this we have a three pins AREF, AVCC and GND.

3-2 Microcontroller

ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

3-3 Display

The LCD unit receives character codes (8 bits per character) from a microcontroller, latches the codes to its display data RAM (80-byte DDRAM for storing 80 characters), transforms each character code into a 5X7 dot matrix character on its LCD screen

Pin No.	Symbol	Level	Description
1	Vss	0V	Ground
2	Vdd	5.0V	Supply Voltage for logic **(option +3V)**
3	Vo	(Variable)	Operating voltage for LCD
4	RS	H/L	H:DATA, L:Instruction code
5	R/W	H/L	H:Read(MPU→Module)L:Write(MPU→Module)
6	E	H,H→L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	A/Vee	—	Power supply for LED backlight (+) / Negative voltage output (optional)
16	K	—	Power supply for LED backlight (-)

<display interface pin function>

4. Software

4-1 display

toggle	PORTD&=0b01111111; _delay_ms(10); PORTD|=0b10000000; _delay_ms(10); PORTD&=0b01111111; _delay_ms(10);	To toggle the Enable signal of the display
turn_on	PORTC&=0b11111100; PORTB&=0b00110011; PORTB|=0b00110000;toggle(); PORTB&=0b00000000; PORTB|=0b00001110; toggle();	Function set and to turn on display

reset	int i=0;
	PORTC&=0b00000000;
	PORTB&=0b00000000;
	PORTB|=0b00000010; toggle();
	
	for(i=0;i<16;i++){
	display_char(0b00100000);
	toggle(); }
	PORTC&=0b00000000;
	PORTB&=0b00000000;
	PORTB|=0b00000010; toggle();
		Clear the display
		
		take cursor to home

display_char	PORTC&=0b11111100; PORTC|=0b00000001; PORTB=character; toggle();	C0 : RS to 1, C1 : RW to 0
display_menu	Using display_char function	Indicate “welcome!”
display_song	Using display_char function	Indicate the number of song

4-2 interface

buttons	uint8_t oct = 0b00000000;
	
	while(oct==0b00000000){ oct=0b00000110&PIND ; } switch(oct){ case 0b00000010: nb--; break; case 0b00000100: nb++; break; }
	if (nb==0){ nb+=10; }
	else if (nb==11){ nb-=10; }
	return nb; }
		D1=1:
		previous song
		
		D2=1:
		Next song

input_choice	int choice =0; int ok=0; int i=0;
	uint8_t oct = 0b00000000; blink();
	while(!ok) { oct=0b00000110&PIND; i++;
	if (oct==0b00000100){ ok++; choice++; }
	else if(oct==0b00000010){ ok++; }
		D1=1:using buttons
		D2=1:using accelerometer

InitADC	ADMUX|=(1<<REFS0); ADCSRA|=(1<<ADEN)|(1<<ADPS0)|(1<<ADPS1)|(1<<ADPS2);	ENABLE ADC, PRESCALER 128

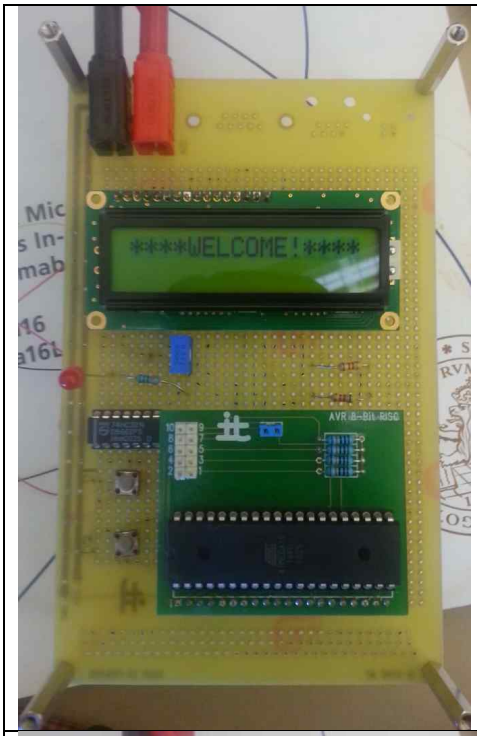
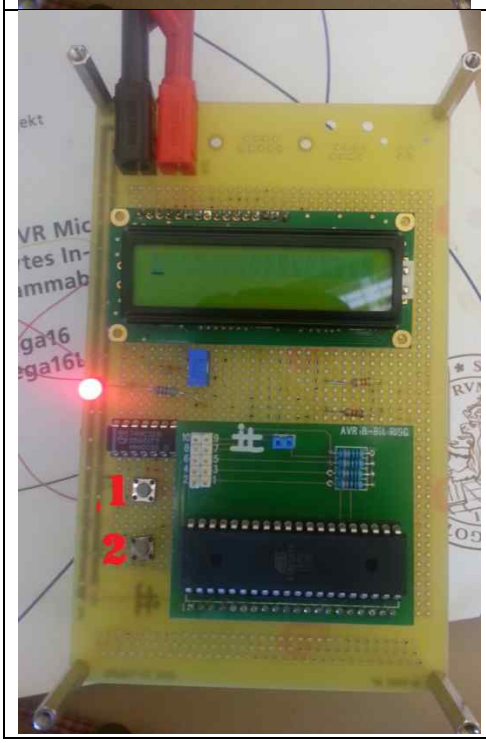
readadc	ch&=0b00000111; ADMUX = (ADMUX & 0xf8)|ch;
	ADCSRA|=(1<<ADSC);
	while((ADCSRA)&(1<<ADSC));
	CONVERSION IS COMPLETE return(ADC);
		START CONVERSION
		And wait

blink	PORTD|=0b00000001; _delay_ms(500); PORTD&=0b11111110;	toggles led for 500 msec

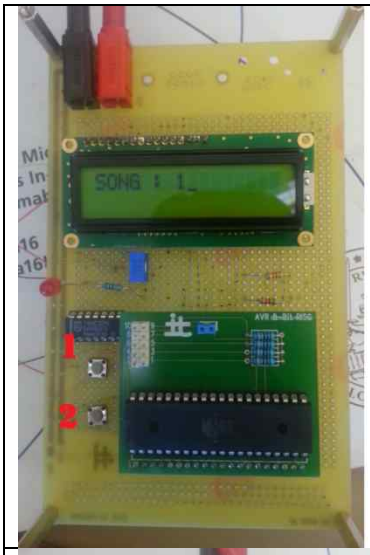
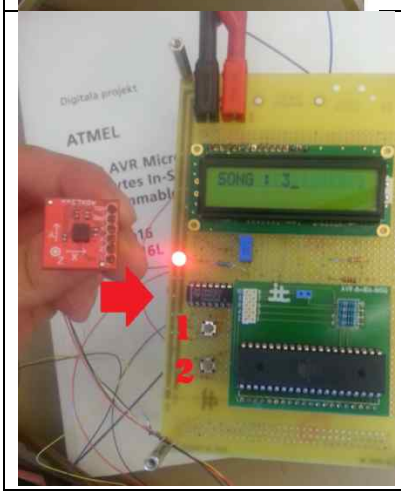
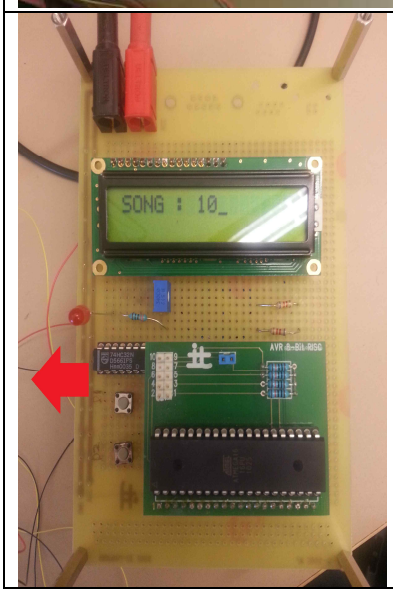
4-3 main

```
int main(void)
{
    int track=1;
    int x=0;
    int choice=0;
     ports_init();
     turn_on();
     InitADC();

     display_menu();
     _delay_ms(4000);
     reset();

     choice=input_choice();


     while(1){

        // only if buttons selected


        reset();
        display_song(track);
        if(choice){
            blink();
            track=buttons(track);
        }else if(!choice){

                x=readadc(0);

                while((x>400) & (x<500)){
                    x=readadc(0);
                }

                if (x<400){
                    blink();
                    track--;
                    if (track==0){
                        track+=10;
                        }else if (track==11){
                        track-=10;
                    }
                }else if (x>500){
                    blink();
                    track++;
                    if (track==0){
                        track+=10;
                        }else if (track==11){
                        track-=10;
                    }
```

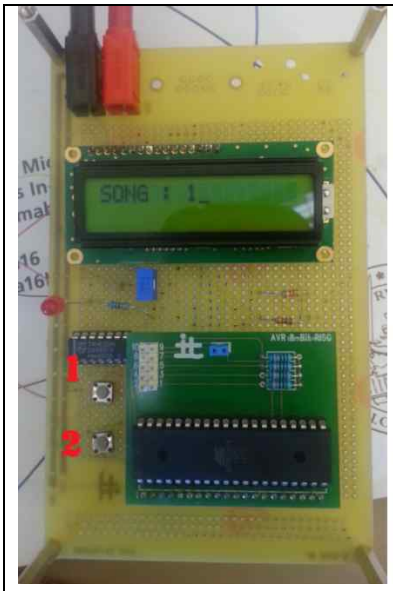
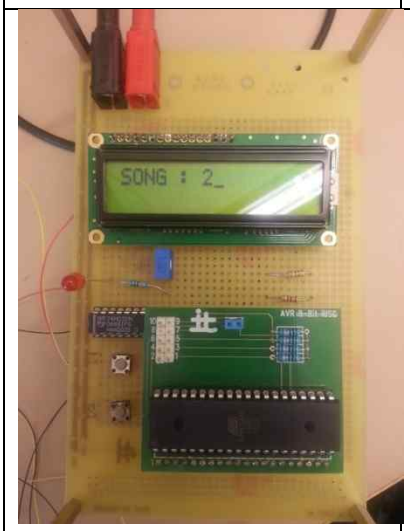
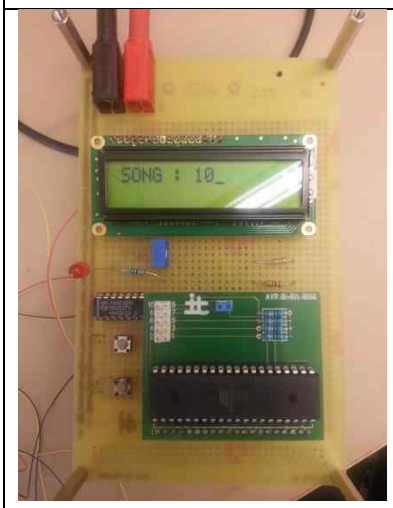
5. How it works

 ***WELCOME!****	1. Music player starts !
 1 2	In this screen, you need to select the external signal to use. 1: accelerometer 2: buttons

If you choose number 1

	"Song 1" appears
	If you shake the accelerometer to that side, song number will increase. If it works well, LED will turn on.
	If you shake the accelerometer to that side, the song number will decrease. But when the song is the first one, song number will go to the last one.

If you choose number 2

	"Song 1" appears
	If you press the button 1, the song number will increase. If it works well, LED will turn on.
	If you press the button 2, the song will decrease. But there is a special case, when the song is in first, it will go to the last song. If it works well, LED will turn on.

6. Conclusion

As I thought at first, it was possible to control the display and LED by accelerometer and buttons. And I could see that display changes well following external signals. But It will be better that using more elaborate accelerometer. And using "interrupt" will make this controller better because you can use buttons whenever you want.

I think this prototype music player can be used in several ways:

-combined with Bluetooth

-can be applied not only for music but also for picture or E-book.