Co-Music Room

Sheng-Kai Tang
Tung-Hsien Wang
Yuchang Hu

Computational Design Lab
School of Architecture,
Carnegie Mellon University
Pittsburgh, PA, 15213

Conceptual Design

Co-Music room is a music space designed for children to explore the music through collaborating with each other. The original idea is to allow children to play the music and experience the corporations though the process. This space, an 8x8 cube, consists of two fundamental components that are circular tile sensors on the floor and ball-shape sensors hanging on the ceiling. Both these two parts are utilized to activate the sound in this space, including music pitches and short tunes. Basically, each circular tile sensor in this demonstration is set up for one single music pitch and we have 7 different dimensional tile sensors in this stage. The other five ceiling ball sensors are regarded as the switches for background music tunes (Figure 1).

Figure 1: The Conceptual Model of Co-Music Room

Detail Mechanisms

During this demonstration, the pre-recorded music will be used to set up the game and then the user will need to follow the indications from both the tile and ball sensors to play the music. In more details, each tile sensor with four sets of LEDs will give the hint for the users by creating the rotating pattern while the status is being activated. Because of the distance between tile sensors and the configuration of them, more than two users will be easier to play the music and thus the collaborative relationship can be made. For the ball sensors, we use the different color for the indication of the next step. The methods used to activate these two sensors are slightly different. On the one hand, as the user step on the tile sensor, the switch is turned on and the corresponding music pitch is played. For the other hand, background music will be played while the ball detects the vibration and this happens when the twinkling ball is hit. Although these two switches are quite simple, they are very intuitive and attracting particularly through their diffusing lighting effects.

Implementation of Tile Sensors

This system has seven round tile sensors which represent seven music pitches. In order to make players easy to recognize and remember what each tile represents, these tiles are designed in different diameters and heights (Figure2).

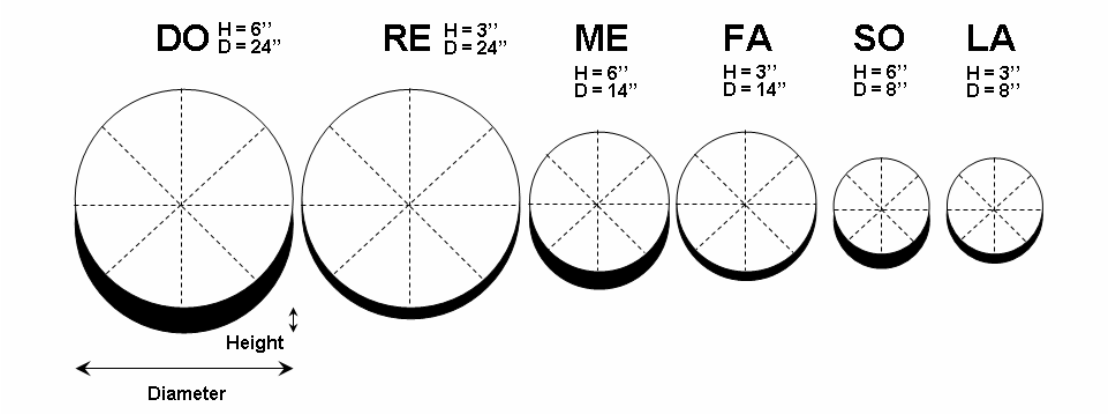


Figure 2: Tiles in different sizes

When starting to implement these tiles, we divide the task of making a tile into two main parts to work on. One is making the physical figure; the other is building the electronic circuit. We have detail records and discussions below:

Figure Design of Tiles

There are some constraints when designing tile sensors. First of all, these tiles should be strong enough to be stepped on, so that we choose wood as the material and design radiative structure to hold weight (Figure 4). Second, these tiles should emit light as indication, so that the upper cover should be transparent or translucent. A 1/4 inch of acrylic board is a good choice. Third, a player is going to trigger a sensor under the tile when stepping on it, so that we mount some springs under the tile to create the clearance for the sensor (Figure 3).

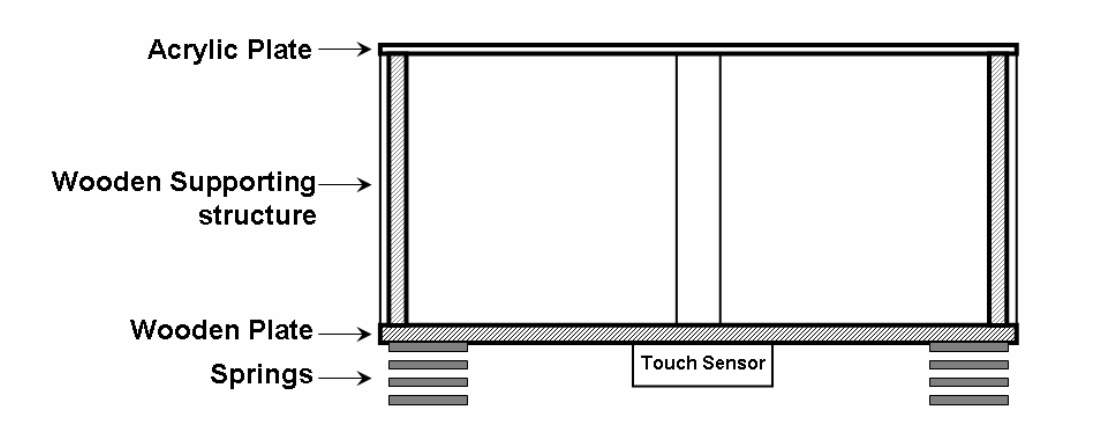


Figure 3: The Tile Sensor Design Scheme

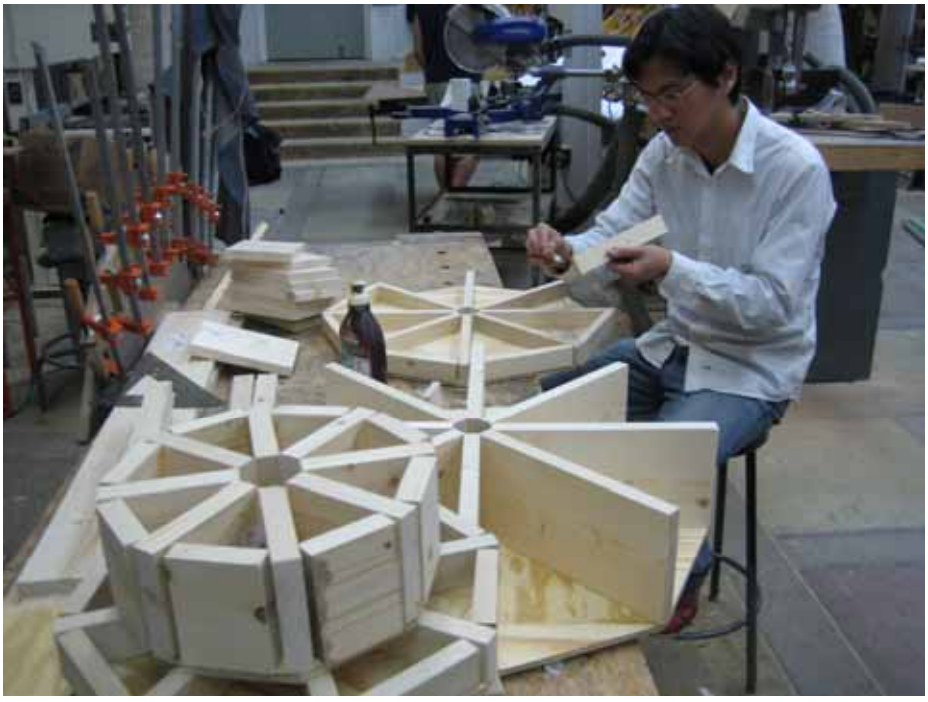


Figure 4: Real conditions of making tiles

Circuit Design of Tiles

The circuit design is the most important part of making a tile. With a circuit consists of LEDs and a pressure sensor, we can turn a none-interactive wooden tile into an interactive one. There are eight LEDs in each tile. Two of them opposites of each other are parallel connected (Figure 5). By so doing, we can create rotation-like effect by trigger these four pairs of LEDs one by one (Figure 6).

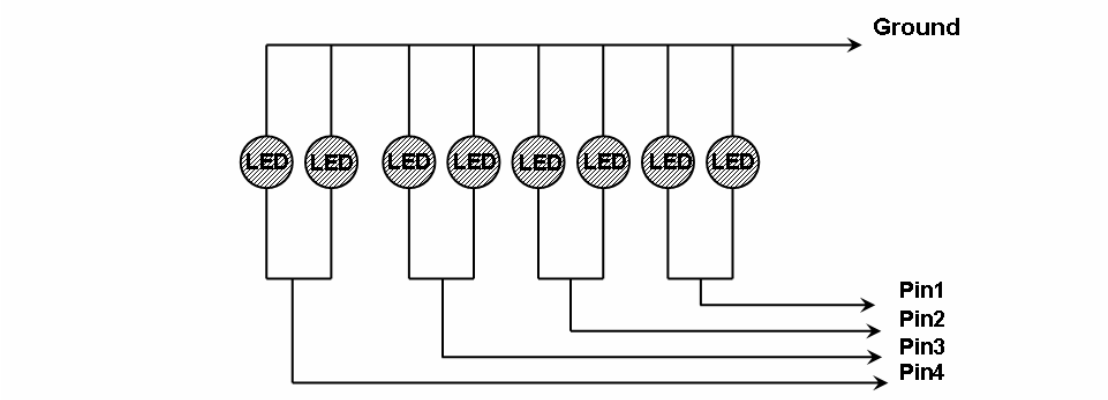


Figure 5: The Circuit Design of LEDs

Figure 6: The Rotation-like Effect

The pressure sensor we made for the tile consists of two copper boards (Figure 7). This sensor is mounted in the clearance, which is the height of the spring, between the tile and the ground. When the spring is compressed, these two copper boards will touch each other and the circuit will be open (Figure 8).

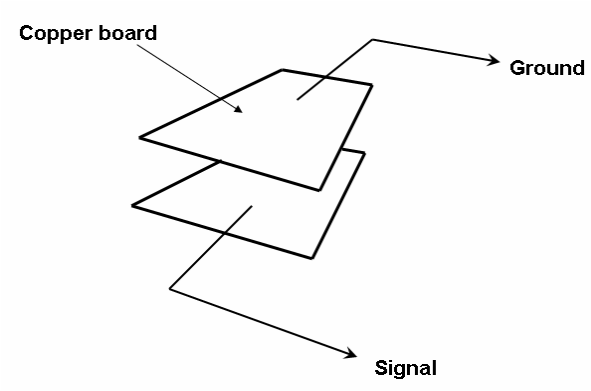


Figure 7: The diagram of sensor design

Figure 8: The pressure sensor

Test

After implementing it, we test its performance. First of all, we test a single tile. The tile is strong enough to step on and the pressure sensor is sensitive enough to trigger. These four pairs of LEDs also create good rotation-like effect. In the test of all set of tiles, we figure out that the spring create good experience for users to jump on and jump to another tiles. When jumping between tiles, it is really fun and also a kind of exercise.

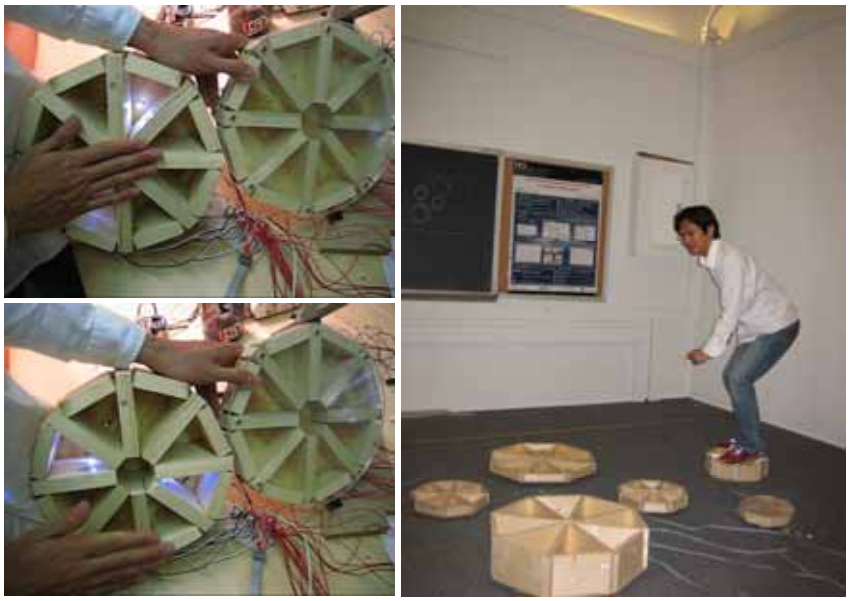


Figure 8: The records of test

Implementation of Ball Sensors

Music balls are designed to be hanged on the ceiling. They allow users to hit them to play short tunes such as drums. Like those tiles, the music balls also provide indications. In the free mode (which we have not implemented it yet), users can hit whichever balls they want to play. When a ball is hit the RGB LED inside the ball lights up and plays music simultaneously. In the learning mode, users have to hit balls according to indications (which are rotating colors). When a lighting ball is hit, it triggers music and lights up the next ball to indicate next step.

Figure and Circuit Design of Balls

Each music ball is made of a plastic lampshade (figure 9) and a set of vibration sensor inside. The schematic is shown as figure 10, which consists of a piezo, an operational amplifier, a diode and several resistors. Whenever the sensor senses vibrate the piezo produces voltage. Because it is too low to be detected directly by a microcontroller, the Op-Amp responses to amplify the signal according to the combination of resistors. In this case, an inverting amplifier is used with the combination of a 1 M and 1K resistors to multiple the original voltage 1000 times.

Figure 9: The plastic lampshade

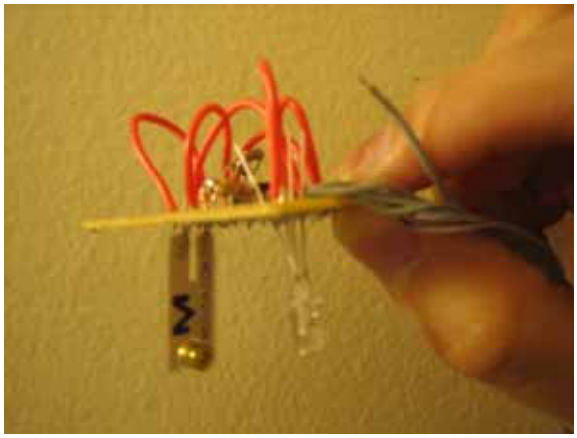


Figure 10: The vibration sensor

Combination

The heart of the circuit is an Atmel[1] Atmega16 microcontroller which has four eight pins ports[2] to be inputs and outputs. It lights up LEDs in tiles and balls as indications according to a predefined music table. It also listens to responses coming from sensors (the switches in the tiles and the vibration sensors in the balls). The detail of the mechanism is that port A as analog inputs listen to tiles and balls at the same time. For example, the first pin of the port A listens to the first tile and first ball, due to different voltage levels. The switches produce digital signal[3], while the vibration sensors produce around 2.5V[4]. Therefore, the microcontroller is able to distinguish where a signal is from. Port B controls turning on/off tiles[5]. The first two pins of port D are UASRT receiving and transmitting pins which response to communicate with computers. The rest six pins of port D control turning on/of balls[6]. The lower four pins of port C controls rotating colors of balls. All balls are hooked up parallely. The higher four pins of port C controls rotating LEDs of tiles. The same as balls, all tiles are hooked up parallely (Figure 11).

In the computer program is a simple Python program which listens to serial signals sent from the microcontroller and plays corresponding music (pitches or tunes).

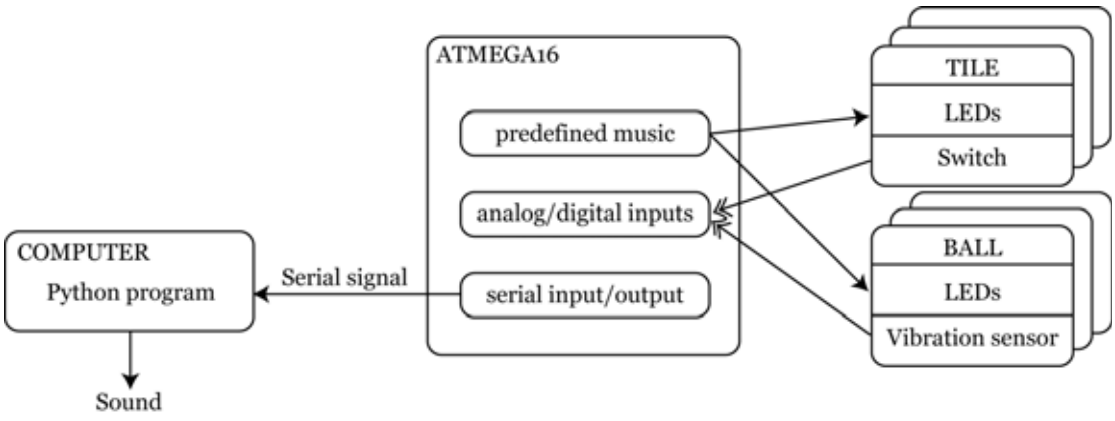


Figure 11: The system scheme

[1]
[2] Marked as A, B, C, D ports
[3] 5V or 255 in a 8-bits ADC
[4] When a ball is hit.
[5] Pull a pin low to turn on a tile or pull it up to turn off a tile.
[6] Pull a pin high to turn on a ball or pull it low to turn off a ball

Results

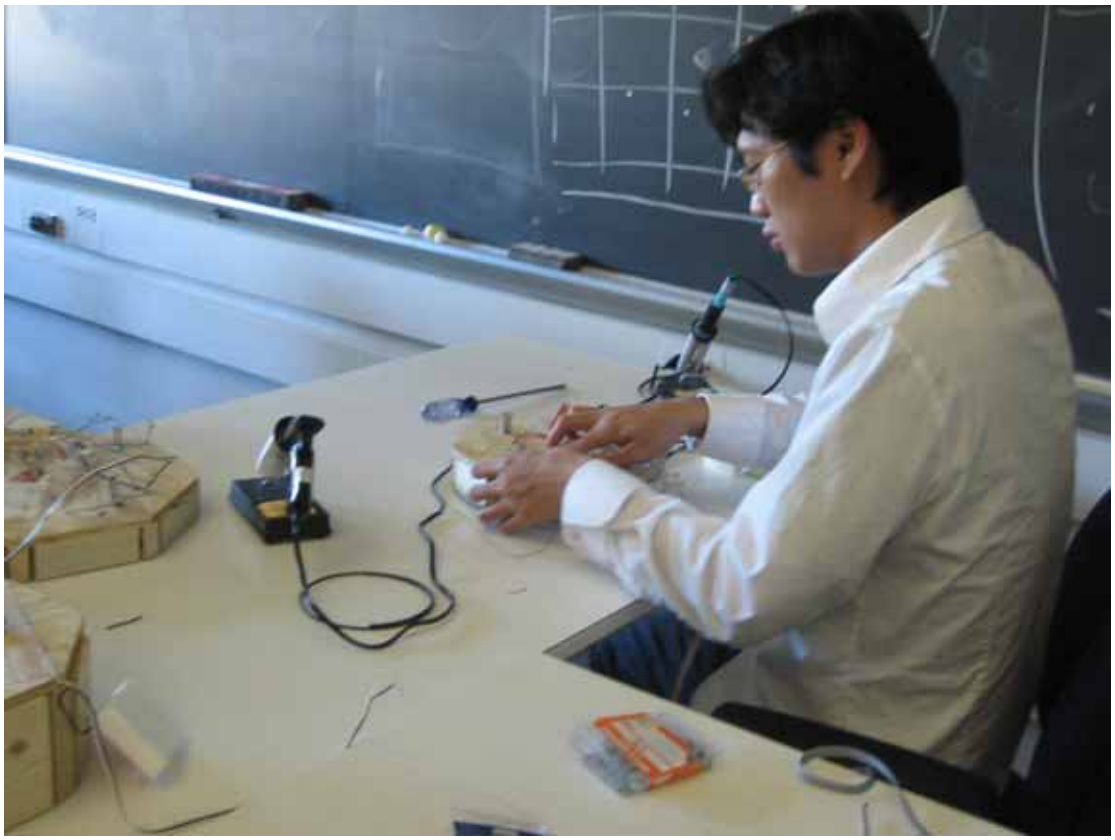
Basically, most of our sensors work quite well during the demonstration despite that is a little bit slower than our expectation. The issue we met in this stage is that we cannot really synchronize all the music together while we have two different kinds of music. One is the simple music pitch and the other is the background music. Besides this, it is interesting to play the music with these sensor inputs. Moreover, the circular tile with the rotating pattern seems to attract more attention than simply providing led as the activating signal.

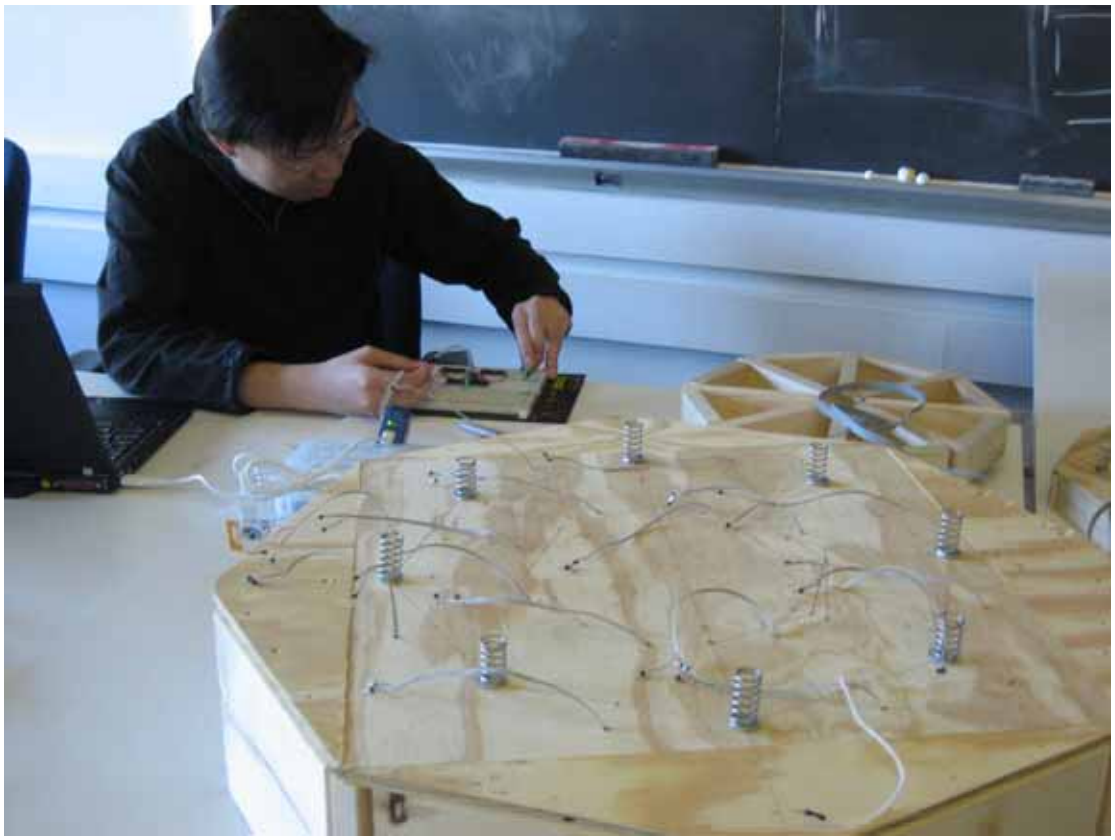
Future Study

In the future, we would like to deal with synchronizing the different type of music first. In this way, the music room will become more interesting with flexible combinations of the different music. More, without regarding pre-recorded music as the destination, we are also delighted to provide users with more possibilities in exploring music composition by means of these sensors.

Appendix

Working Records





Code

```
#include <avr/io.h>
#include <stdlib.h>
#include <stdio.h>

#define F_CPU 4000000UL
#define BAUD 9600
#include <util/delay.h>

#define THRESHOLD 50

int tileTable[] =
{5,3,3,4,2,2,1,2,3,4,5,5,5,5,3,3,4,2,2,1,3,5,5,1,1,1,2,2,2,2,2,3,4,3,3,3,3,3,4,5,5,3,3,4,2,2,1,3,5,5,1,1,1,-1
};

void USART_Init(int baudRate){
	unsigned int ubrr;
	ubrr = F_CPU/(16l*baudRate)-1;
	UBRRH = (unsigned char)(ubrr>>8);
	UBRRL = (unsigned char)ubrr;

	UCSRB = (1<<RXEN)|(1<<TXEN);//|(1 << RXCIE);
	UCSRC = (1<<URSEL)|(3<<UCSZ0);//|(1<<USBS);
}

void USART_putc(unsigned char c) {
    // wait until UDR ready
	while(!(UCSRA & (1 << UDRE)));
	UDR = c;	// send character
}

void USART_puts (char *s) {
	//	loop until *s != NULL
	while (*s) {
		USART_putc(*s);
		s++;
	}
}
```

```
unsigned char read_adc (int PIN) {

        unsigned char value_read;
        int i;

        //start conversion

        ADMUX = (1<<ADLAR) | (PIN);
        ADCSRA = (1<<ADEN) | (1<<ADSC) | (1 << ADPS1);//(1<<ADPS2) | (1<<ADPS0);
        //delay to allow conversion to finish
        for(i = 0; i < 7000; i++){}
        //read value
        value_read = ADCH;
       ADCSRA = 0x00;                          // disable ADC
        return value_read;
}

void rotateLights( void ){
        int i;
        for(i = 4; i < 8; i++){
                PORTC = 0x0F | (1 << i); // lower four bits > ball
                                                // higher four bits > tile
                //PORTC = 0x00 | ~(1 << ball);
                _delay_ms(5000);
                _delay_ms(5000);
                //_delay_ms(5000);
               PORTC = 0x00 | (1 << (i - 4));
               _delay_ms(5000);
               _delay_ms(5000);
               //_delay_ms(5000);
        }
}

void setIndex(int tile, int ball){
        PORTB = 0x00 | ~(1 << tile);
        PORTD = 0x00 | (1 << (ball+2));
}
```

```
void playBall(int ball){
        USART_putc('b');
        //_delay_ms(500);
        USART_putc((char)ball);
        USART_putc('\n');
}

void playTile(int tile){
        USART_putc('t');
        //_delay_ms(500);
        USART_putc((char)tile);
        USART_putc('\n');

}

int checkBall( int Ball ){
      unsigned char adc_value;
      int i;
      for(i = 0; i < 6; i++){
            adc_value = read_adc(Ball);
            //USART_putc(adc_value);

            if(adc_value < 200){
            }
            if(Ball == 5){
                  if(adc_value > 99){
                        if(adc_value < 200){
                              return 1;
                        }
                  }
            }
            if(Ball == 0){
                   if(adc_value < 10){
                        //USART_putc((char)99);
                           return 1; // check if any ball is touched
                   }
            }
```

```
            if(Ball == 1 || Ball == 3){
                if(adc_value < 50){
                    //USART_putc((char)99);
                      return 1; // check if any ball is touched
                }
            }
        }
        return 0;
}

int checkTile(int tile){
        unsigned char adc_value;
        adc_value = read_adc(tile);
        //USART_putc((char)99);
        //USART_putc(adc_value);
        if(adc_value >= 200){
            return 1;
        }
        return 0;
}

int main(void){
        DDRB = 0xFF; //outputs for tile grounds;
        DDRD = 0xFF; //outputs for ball grounds
        DDRC = 0xFF; //outputs for rotating lights

        PORTB = 0xFF; // turn off all tiles
        PORTC = 0x00; // turn off all lights
        PORTD = 0x00; // turn off all balls
        int i;
        int tileIsSet;
        int ballColor;
        int whichTile;
        int whichBall;

        whichTile = 0;
        whichBall = 0;
        ballColor = 0;
```

```
	tileIsSet = 0;
	i = 0;

	USART_Init(BAUD); // initilize USART port

	while(1){
		rotateLights();
		whichBall = whichTile;
		if(checkBall(whichBall)){
			playBall(whichBall);
		}
		if(tileIsSet == 0){
			if(tileTable[i] != -1){
				whichTile = tileTable[i] - 1;
				i++;
			}
			if(tileTable[i] == -1){
				i = 0;
			}
			tileIsSet = 1;
		}
		if(tileIsSet == 1){
			if(checkTile(whichTile) == 1){
				playTile(whichTile);
				//USART_putc((char)whichTile);
				tileIsSet = 0;
			}
		}
		setIndex(whichTile, whichBall);
		_delay_ms(1000);
	}
	return 1;
}
```