Arduino ‘Simon Says’ Using LEDs and Buzzer

E. Kando
E.T. Pop

Technical University of Cluj-Napoca
Computer Science Department
Str. G. Baritiu nr 26-28, 400027 Cluj-Napoca, Romania

E-mail: gebadaiafan@hotmail.com
edinakando@gmail.com

Abstract. This article presents the implementation of a “Simon Says” game using LEDs, buttons and a Buzzer.

1. Introduction

These guidelines discuss how to proceed in implementing an Arduino “Simon Says” game using LEDs, buttons and a Buzzer. It is a very simple game: 4 LEDs will lit in a random order, then you are supposed to click on the button below the LED in the correct order. If the order is incorrect, 4 LEDs will light in a consecutive order. Otherwise, it keeps adding a light every level until you lose.

2. Piezo Buzzer

Piezo Buzzers are usually used for making beeps alarms and tones. We used separate frequencies of the buzzer for each colour of the LEDs, and an error sound when the order introduced is incorrect.

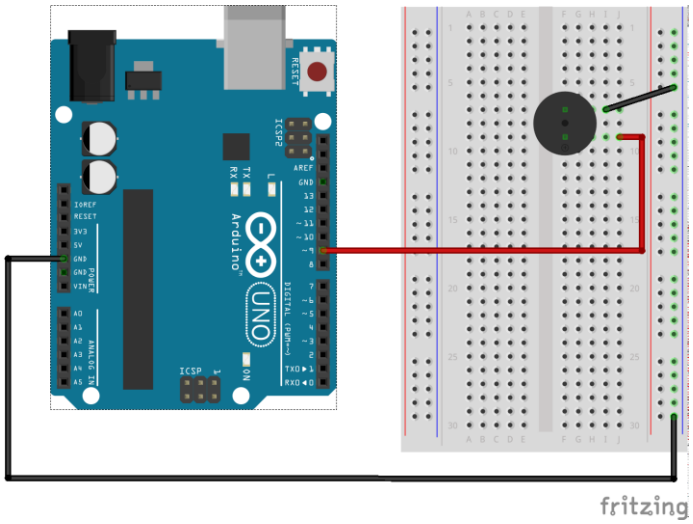


Figure 1: Piezo Buzzer connected to Arduino UNO

3. Buttons

Pushbuttons connect two points in a circuit when you press them. When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that we read a HIGH.
In the circuit, we used 5 buttons. 4 of them were used for the LEDs, and the other one for the reset function.

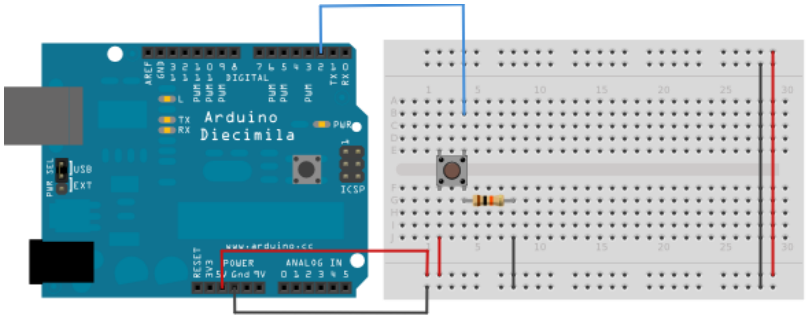


Figure 2: Button connected to Arduino UNO

4. Arduino UNO

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller.

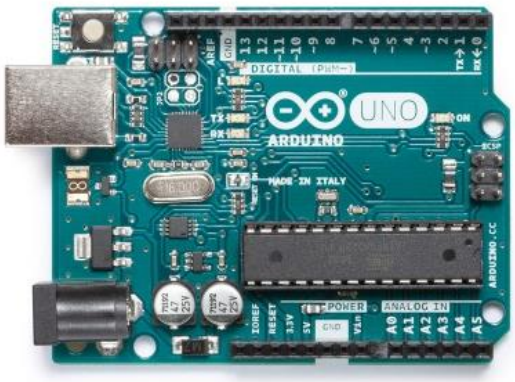


Figure 3: Arduino UNO

5. Implementation

We used:

- 4 different coloured LEDs (blue, green, red, yellow)
- A buzzer
- 5 buttons
- One resistor of 100 Ω for the Buzzer
- 4 resistors of 10k Ω for the buttons

- 4 resistors of 330 Ω for the LEDs
- Arduino UNO

Each LED is connected to GND and to an even digital pin (2,4,6,8), whilst the push buttons that are used for playing the game are connected to VCC and to an odd digital pin (3,5,7,9) and the reset button to GND and RESET pin. The piezo buzzer is connected to pin 13 and to GND.

6. Code – Algorithm

- First of all we introduce in the vector colors a randomly generated sequence of colours that we will use in the current game.
- Secondly we have a subprogram called printSequence that will turn on the lights from the generated sequence one by one (depending on the level reached)

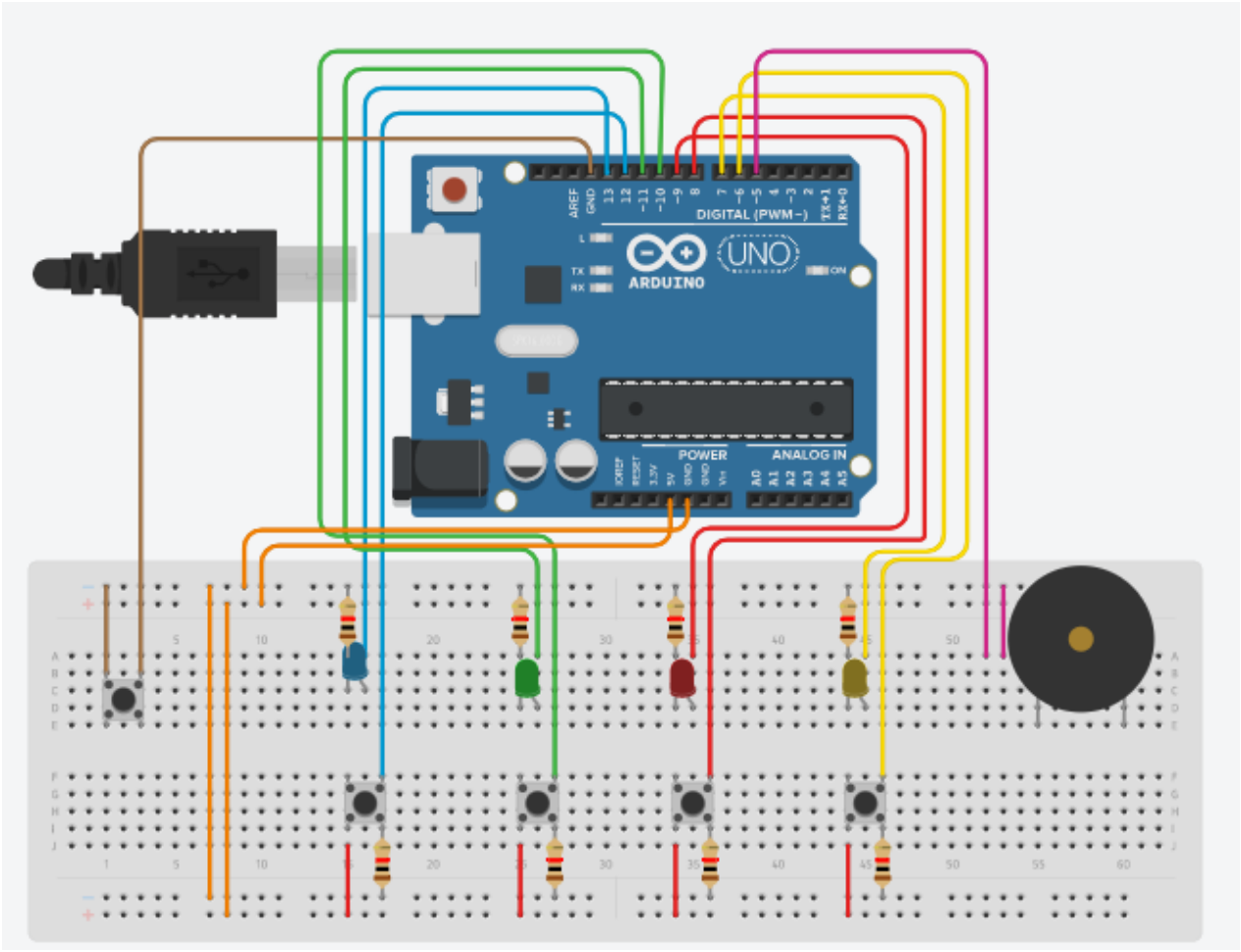
```
void printSequence(int level){
  for(int i = 0; i <= level; i++){
    digitalWrite(L1 + colors[i] * 2, HIGH);
    tone(buzzer,200+(L1 + colors[i] * 2)*100,500);
    delay(300);
    digitalWrite(L1 + colors[i] * 2, LOW);
    delay(300);
  }
}
```

- digitalWrite is used to turn on the selected light. We used the variable L1 + colors[i] * 2 because we start from LED 1 (having the value 4) and add colors[i] *2. Finally, the whole number varies like: 4 – 6 – 8 – 10 and activates the corresponding LEDs (since they are connected to even-numbered digital pins).
- tone will create a specific sound for each LED activated.

- Then we have the readAnswer function that finds the inputted button corresponding to the desired LED. After finding it, it uses a switch statement in order to find out whether or not the button pressed is the correct one. If it isn't the correct one, we use another function called flicker to create the error message and reset the sequence.
- The function flicker will create an audio and visual error message. The audio error message is a specific sequence of sounds the buzzer will create, while the audio error message will light the LEDs in consecutive order 4 times. After the error message is finished, the whole sequence will start from level one, regardless of the level you reached before.

```
void flicker()
{


  for(int j=1;j<=3;j++)
  {  tone(buzzer, 300-j*23);
    for(int i=0;i<=6 ;i+=2)
    {
      digitalWrite(L1+i, HIGH);
      delay(50);
      digitalWrite(L1+i, LOW);
      delay(50);

    }
    noTone(buzzer);
    delay(50);
  }

  delay(200);

}
```

7. Schematic



8. Conclusions

In conclusion, this project provides the quite fun and addictive game of “Simon Says”. The game is accessible not only for kids, due to its simplicity, but also for adults. It challenges the player to use their visual/audio memory in order to get the sequence right.

This game is widely known in many countries, and it is originally meant to be a child's game for 3 or more players where 1 player takes the role of "Simon" and issues instructions to the other players, which should only be followed if prefaced with the phrase "Simon says". Players are eliminated from the game by either following instructions that are not immediately preceded by the phrase, or by failing to follow an instruction which does include the phrase "Simon says".

The object for the player acting as Simon is to get all the other players out as quickly as possible; the winner of the game is usually the last player who has successfully followed all of the given commands. Occasionally however, 2 or more of the last players may all be eliminated by following a command without "Simon Says", thus resulting in Simon winning the game.

9. References

[1] https://www.arduino.cc/en
[2] https://www.youtube.com/
[3] http://www.instructables.com/
[4] https://programmingelectronics.com/home/