REPUBLIC OF THE PHILIPPINES

SURIGAO DEL NORTE STATE UNIVERSITY

Narcisso Street, Surigao City 8400, Philippines

Caspe,Mark Angelo J

BSECE-3A

Laboratory Activity no.6

Music:Piezo Elements

**Introduction:**

To this point we have controlled light, motion, and electrons, Lets tackle sound next. But sound is ananalog phenomena, how will our digital Arduino cope? We will once again rely on its incredible speed which will let it mimic analog behavior. To do this, we will attach a piezo element to one of the Arduino's digital pins. A piezo element makes a clicking sound each time it is pulsed with current.If we pulse it at the right frequency (for example 440 times a second to make the note middle A) these clicks will run together to produce notes. Lets get to experimenting with it and get your Arduino playing 'Twinkle Twinkle Little Star'.

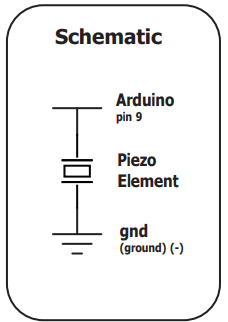
**Objectives:**

The goal is to make an Arduino create sounds like a musical instrument by using its fast processing abilities. We'll connect a piezo element to the Arduino and control it to produce different notes by pulsing it at specific frequencies. The objective is to program the Arduino to play the melody of "Twinkle Twinkle Little Star" and test this setup using Proteus simulation software to make sure everything works correctly before trying it out in real life.

**Materials:**

1.) Atmega328P 2.) Crystal 3.) Piezo Element x1 4.) 22pf capacitor 5.)Push Button

**Schematic Diagram:**



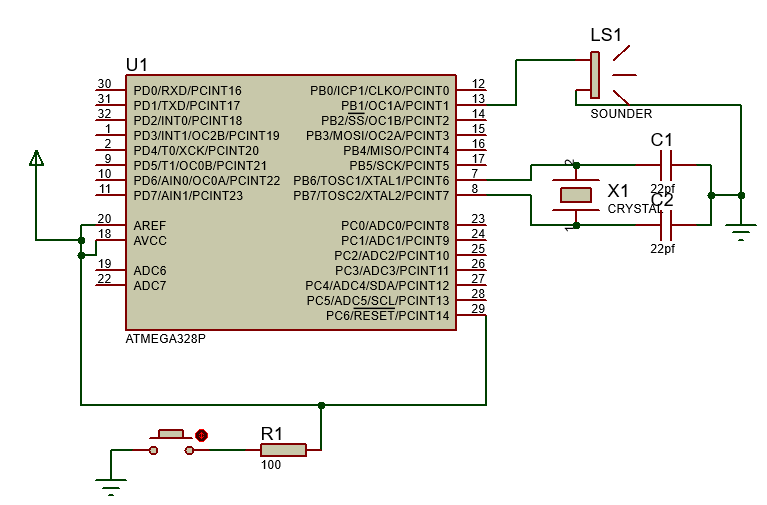
**Procedure:**

* Open Proteus and Create a New Project: Launch Proteus and start a new project to begin setting up the simulation environment.
* Find Components: Within Proteus, locate and add the necessary components for your project, including the Arduino board and the piezo buzzer.
* You can refer to the schematic diagram for guidance on how to connect these components correctly.
* Open Arduino IDE and Write Code: Open the Arduino IDE and write the code for your project. This code will control the behavior of the Arduino, including how it interacts with the piezo buzzer to produce sound.
* Once the code is written, save it for later use.
* Link Code to Proteus: Return to Proteus and click on the ATmega328P microcontroller component. Then, locate the file you saved in step 3, which is the .hex file containing the compiled Arduino code.
* Choose this file to link your code to the microcontroller in the simulation.
* Simulate: With the code linked to the microcontroller, you can now simulate or run the simulation in Proteus. This will allow you to see your project in action. The piezo buzzer will produce sound according to the instructions provided in the code.

**Results and Discussion:**

In the simulation using Proteus, the piezo buzzer played the tune of "Happy Birthday." We could change how fast or slow the song played by adjusting certain parts of the code. This shows how flexible and controllable the Arduino is in creating musical sounds.

**Circuit:**



**Program:**

int speakerPin = 9;

int length = 13;

char notes[] = "ccdcfeccdcgf ";

int beats[] ={1,1,1,1,1,2,1,1,1,1,1,2,4};

int tempo = 250;

void playTone(int tone, int duration) {

  for (long i = 0; i < duration \* 1000L; i += tone \* 2) {

    digitalWrite(speakerPin, HIGH);

    delayMicroseconds(tone);

    digitalWrite(speakerPin, LOW);

    delayMicroseconds(tone);

  }

}

void playNote(char note, int duration) {

  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };

  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };

  // play the tone corresponding to the note name

  for (int i = 0; i < 8; i++) {

    if (names[i] == note) {

      playTone(tones[i], duration);

    }

  }

}

void setup() {

  pinMode(speakerPin, OUTPUT);

}

void loop() {

  for (int i = 0; i < length; i++) {

    if (notes[i] == ' ') {

      delay(beats[i] \* tempo); // rest

    } else {

      playNote(notes[i], beats[i] \* tempo);

    }

    // pause between notes

    delay(tempo / 5);

  }

}

**Findings:**

The finding from the simulation in Proteus is that the Arduino, when programmed correctly, effectively controls the piezo buzzer to play the melody of "Happy Birthday." By adjusting specific parts of the code, such as note durations or tempo, the speed of the song can be manipulated. This highlights the versatility and precision of Arduino-based systems in generating musical sounds.

**Recommendations:**

Based on the simulation results, it's recommended to further explore the capabilities of the Arduino in generating musical sounds. Experiment with different melodies, rhythms, and tempo adjustments to understand the full potential of the Arduino's musical applications. Additionally, consider incorporating other components or sensors to create interactive musical projects, fostering creativity and innovation in Arduino-based music projects.

**Conclusions:**

In conclusion, the simulation demonstrated that Arduino can effectively control a piezo buzzer to play music like "Happy Birthday." By adjusting the code, we could change the song's speed. This shows Arduino's potential for creating musical projects and encourages further exploration of its capabilities in music and beyond.