Julia Frothingham

Lab 18 - Projects 2 A B and C

What does digitalRead do?

`digitalRead` takes in a signal on a specified digital input pin and determines whether it is one of two possible values, LOW or HIGH.

Why did we need to use 2 equal signs with the if statements in the code for 2B?

In most programming languages, a single equal sign is used to set the value of a variable. In order to compare two things, the double equal signs are used instead.

What are FOR loops useful for?

FOR loops are useful for operations where you need to perform the same action a number of times in a row - in this case, iterating through an array and checking an input against each array value.

What's the difference between Delay(xxx) and Millis()?

`millis` returns the time since the program started and can be used to determine exact times relative to each other. `delay` pauses for a set amount of time.

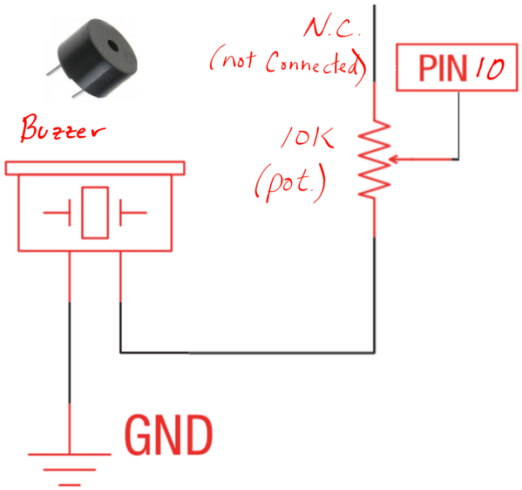
What's an array? How can you tell if a variable is an array or not?

An array is a data structure that contains a number of values. In Arduino, you can tell if an array is a variable if it is declared with the datatype of the variables it contains followed by square brackets (e.g., `int[]`), or if it is set equal to a list of comma-separated values contained in curly braces.

Why might an array be more useful than an equal number of independently defined variables?

If you have more than a few variables, it will quickly become nearly impossible to read or write code that defines and uses them individually - for instance, if you're storing data, an array is a MUCH better option than a variable for each value. Also, arrays can be used to group variables together - you can have an array of timestamps, an array of frequencies, an array of different settings, etc. all in the same program. Finally, with an array, you can use an iterative loop (like a FOR loop) to efficiently access and operate on each variable.

Figure 1. Project 2 Circuit 2A diagram and code



Circuit_2A_Buzzer

```
/*
 * CIrcuit 2A-Buzzer
 * Play notes using a buzzer (i.e., a very cheap speaker
 */

int speakerPin = 10;    // buzzer connects to pin 10

void setup() {
  pinMode(speakerPin, OUTPUT);    // the arduino will output a signal to drive the speaker
}

void loop() {
  // play specific notes for specific beats
  play('g', 2);
  play('g', 1);
  play('a', 4);
  play('g', 4);
  play('C', 4);
  play('b', 4);
  play(' ', 2);   //a pause before playing next set of notes

  play('g', 2);
  play('g', 1);
  play('a', 4);
  play('g', 4);
  play('D', 4);
  play('C', 4);
  play(' ', 2);

  play('g', 2);
  play('g', 1);
  play('G', 4);
  play('E', 4);
  play('C', 4);
  play('b', 4);
  play('a', 6);
  play(' ', 2);
```

Figure 1 continued.

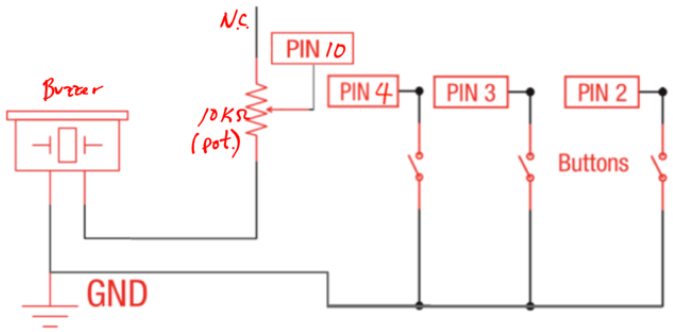
```
  play('F', 2);
  play('F', 1);
  play('E', 4);
  play('C', 4);
  play('D', 4);
  play('C', 6);

  while(true){}   // this line prevents the song from being played again by essentially
                  // "traps" the code in an always-true condition that makes it do nothing
}

void play(char note, int beats) {
  // number of available notes
  int numNotes = 14;
  // available notes (C major scale in two octaves)
  char notes[] = {'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C', 'D', 'E', 'F', 'G', 'A', 'B', ' '};
  // frequencies associated with each note, in same order as notes are named in array notes[]
  int frequencies[] = {131, 147, 165, 175, 196, 220, 247, 262, 294, 330, 349, 392, 440, 494, 0};
  // variable to store the frequency of the note to play
  int currentFrequency = 0;
  // one beat = 150 milliseconds
  int beatLength = 150;

  // iterate through the list of notes until you hit the note passed as function's input parameter
  // use the array index of that note to find the associated frequency
  for(int i=0; i<numNotes; i++){
    if(notes[i]==note){
      currentFrequency = frequencies[i];
    }
  }

  // output a soundwave at given frequency through given pin for a given amount of time
  tone(speakerPin, currentFrequency, beats*beatLength);
  delay(beats*beatLength);  // wait for note to finish playing
  delay(50);  // wait 50ms before continuing code

}
```

The circuit plays the song Happy Birthday once through the buzzer. A video of the circuit in action can be viewed here: https://youtu.be/eLfOhwztBQo

When viewed through an oscilloscope, the arduino output is an approximation of a square wave. Its FFT shows a number of distinct frequency components, approximately evenly spaced over the range measured. This makes sense, as the output of the arduino is digital, so it will be a square wave, and a square wave can be written as a sum of sinusoids of different frequencies with harmonics.

Figure 2. FFT and v(t) of Arduino output while playing music through the buzzer.

Figure 3. Project 2 Circuit 2B diagram and code.



Circuit_2B_DigitalTrumpet

```
/*
 * Circuit 2B-ButtonTrumpet
 * Use 3 buttons to play musical notes on a buzzer
 */

// the arduino pins each component is connected to
int firstKeyPin = 2;
int secondKeyPin = 3;
int thirdKeyPin = 4;
int buzzerPin = 10;

void setup() {
  // set the connected to the buttons as input pins
  // that use the internal arduino pullup resistors to force output between 0V and Vmax
  pinMode(firstKeyPin, INPUT_PULLUP);
  pinMode(secondKeyPin, INPUT_PULLUP);
  pinMode(thirdKeyPin, INPUT_PULLUP);
  // set the pin connected to the buzzer as output pin
  // so that the arduino will drive the buzzer
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  // if the first button is pressed, buzzer will play note at 262Hz
  if(digitalRead(firstKeyPin)==LOW){
    tone(buzzerPin, 262);
  }
  // if the second button is pressed, buzzer will play note at 330Hz
  else if(digitalRead(secondKeyPin)==LOW){
    tone(buzzerPin, 330);
  }
  // if the third button is pressed, buzzer will play note at 392Hz
  else if(digitalRead(thirdKeyPin)==LOW){
    tone(buzzerPin, 392);
  }
  // if no buttons pressed, buzzer will not play anything
  else{
    noTone(buzzerPin);
  }
}
```

The circuit plays three different notes through the buzzer, depending on which button is pressed. Adjusting the code slightly to use binary, the circuit can produce seven notes from the same three buttons. Pressing the buttons in different combinations produces different pitches. A video of the circuit in action can be viewed here: https://youtu.be/gwGFlsZBoAk

Figure 4. Project 2 Circuit 2B modified code

Circuit_2B_DigitalTrumpet_MoreNotes

```
// the arduino pins each component is connected to
int firstKeyPin = 2;
int secondKeyPin = 3;
int thirdKeyPin = 4;
int buzzerPin = 10;

// variable to store state
int inputSum = 0;

// array of frequencies for each pitch
int freq[] = {0, 262, 294, 330, 349, 392, 440, 494};

void setup() {
  // set the connected to the buttons as input pins
  // that use the internal arduino pullup resistors to force output between 0V and Vmax
  pinMode(firstKeyPin, INPUT_PULLUP);
  pinMode(secondKeyPin, INPUT_PULLUP);
  pinMode(thirdKeyPin, INPUT_PULLUP);

  // set the pin connected to the buzzer as output pin
  // so that the arduino will drive the buzzer
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  //determine the combination of buttons pressed
  inputSum = readInputs(firstKeyPin, secondKeyPin, thirdKeyPin);

  if(inputSum > 0){
    // if any buttons are pressed, play tone
    // frequency depends on combination of buttons pressed
    tone(buzzerPin, freq[inputSum]);

    // reset state variable
    inputSum = 0;
  } else {
    // if no buttons are pressed, don't play a tone
    noTone(buzzerPin);
  }
}

//define custom function to read button combination and determine state
int readInputs(int one, int two, int three){
  int sum = 0;
  if(digitalRead(one)==LOW){sum+=1;} // equivalent to 001 in binary
  if(digitalRead(two)==LOW){sum+=2;} // equivalent to 010 in binary
  if(digitalRead(three)==LOW){sum+=4;} // equivalent to 100 in binary

  return sum;
}
```