# Week 8 – Digital Logic, Micro-Computer Systems & Digital Music

**Lab Report**

|  |  |  |
| --- | --- | --- |
| Name: Jacqueline Radding |  | Date: |
|  |  |  |
| EE 151 | Section: Monday Online | Lab Bench: |

**LEARNING OBJECTIVES**

1. Understand the function of the tone( ) Arduino command and how to use it.
2. Connect waveform fundamental frequency with musical pitch.
3. Learn to make use of a distance sensor.
4. Know how to utilize several more advanced programming instructions and methods, including functions returning values, compound Boolean conditional statements, and Boolean functions.
5. Understand basic digital (Boolean) logic functions like AND (&&) and OR (||)
6. Show how to convert a binary number value to a decimal value..
7. Define the terms *bit* and *byte*.
8. Describe what ASCII character codes are used for.
9. Describe the difference between an “*assembly language*” and a *high-level programming language*.
10. Describe the functions of a “*compiler”* and an “*assembler*”.

**Arduino Project 1 – Making Your Arduino Sing with Constants (not constantly)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Your Song Note Sequence to Program:**   |  |  | | --- | --- | | C4 | 2 (half note) | | G4 | 4 (whole note) | | F4 | 1 (quarter note) | | E4 | ½ (eighth note) | | D4 | ½ (eighth note) | | C5 | 2 (half note) | | G4 | 4 (whole note) | |

**Name that Tune: Star Wars intro**

**Your Arduino sketch:**

// Star wars intro

//by Jacqueline Radding

//plays the first few notes of star wars intro song

const int Beeper = 9 ; // Digital Pin 9

const int quarterNote = 500 ; // msec for quarter note

const int C4freq = 262 ; // Hz for c4 note

const int G4freq = 392 ; // Hz for d4 note

const int F4freq = 349 ; // Hz for c4 note

const int E4freq = 330 ; // Hz for d4 note

const int D4freq = 294 ; // Hz for c4 note

const int C5freq = 523 ; // Hz for c5 note

void setup() {

// put your setup code here, to run once:

//plays all of the notes

tone(Beeper, C4freq, 2\*quarterNote); // Output C4 for a half note

delay(2\*quarterNote); // Wait for a half note to finish

tone(Beeper, G4freq, 4\*quarterNote); // Output G4 for a half no

delay(4\*quarterNote); // Wait for a note to finish

tone(Beeper, F4freq, quarterNote); // Output F4 for a half note

delay(quarterNote); // Wait for a .25 note to finish

tone(Beeper, E4freq, .5\*quarterNote); // Output E4 for a half note

delay(0.5\*quarterNote); // Wait for a 1/8 note to finish

tone(Beeper, D4freq, .5\*quarterNote); // Output D4 for a half note

delay(0.5\*quarterNote); // Wait for a 1/8 note to finish

tone(Beeper, C5freq, 2\*quarterNote); // Output C5 for a half note

delay(2\*quarterNote); // Wait for a half note to finish

tone(Beeper, G4freq, 4\*quarterNote); // Output G4 for a half note

delay(4\*quarterNote); // Wait for a note to finish

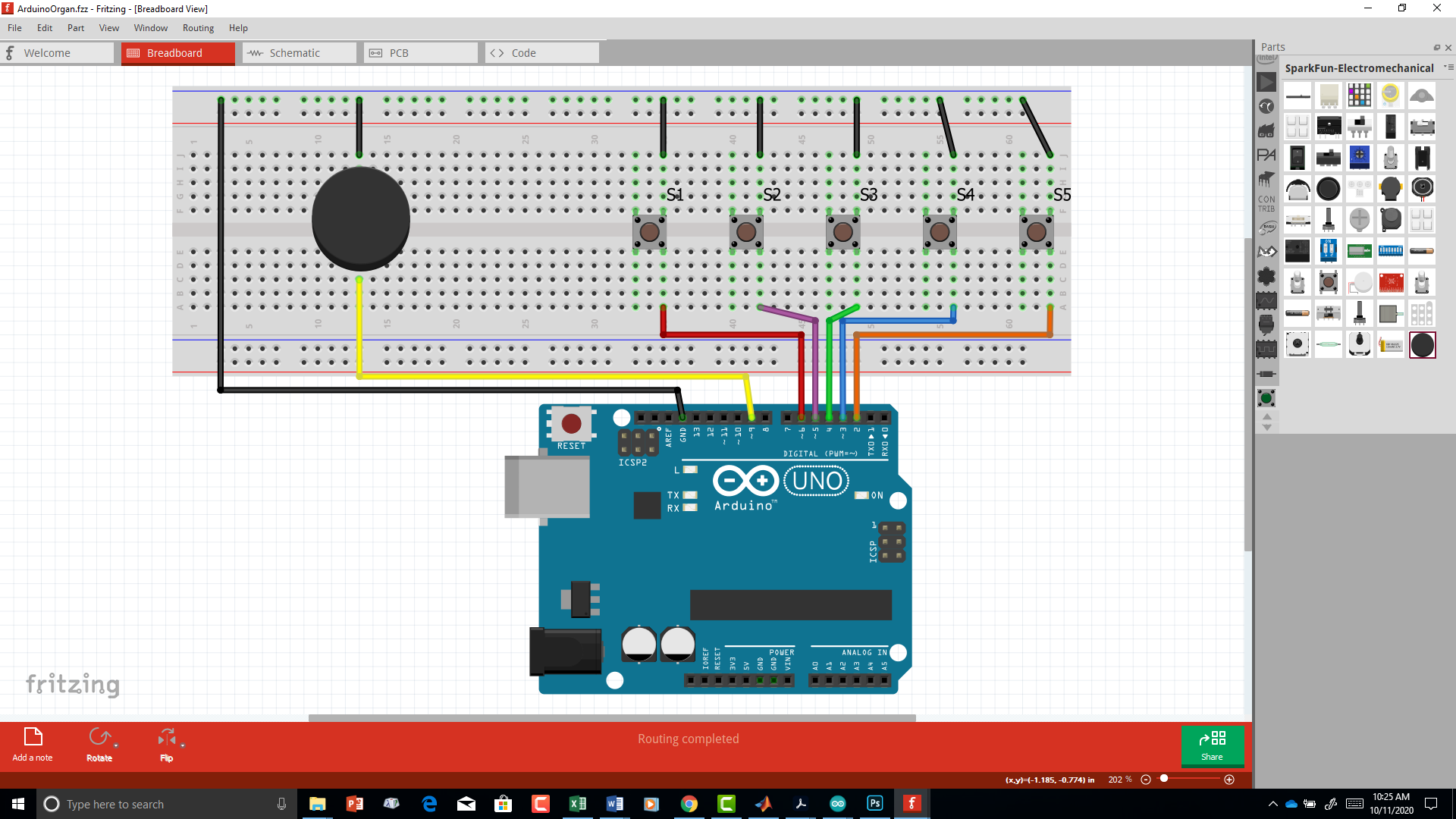
}

void loop() {

// put your main code here, to run repeatedly:

}

**Arduino Project 2 – Making An Arduino Keyboard Instrument (*An IFfy Organ*)**



**Your Arduino sketch:**

// Arduino Keyboard

//by Jacqueline Radding

//plays a note when a button is pressed

const int Ckey = 6 ; // Pin for S1 acting as C4 key

const int Dkey = 5 ; // Pin for S2 acting as C4 key

const int Ekey = 4 ; // Pin for S3 acting as C4 key

const int Fkey = 3 ; // Pin for S4 acting as C4 key

const int Gkey = 2 ; // Pin for S5 acting as C4 key

int reading;

//bool f; // this is the true or false value

const int Beeper = 9 ; // Digital Pin 9 for buzzer

const int quarterNote = 500 ; // msec for quarter note

const int C4freq = 262 ; // Hz for c4 note

const int G4freq = 392 ; // Hz for d4 note

const int F4freq = 349 ; // Hz for c4 note

const int E4freq = 330 ; // Hz for d4 note

const int D4freq = 294 ; // Hz for c4 note

const int A4freq = 523 ; // Hz for c5 note

const int B4freq = 523 ; // Hz for c5 note

void setup() {

// put your setup code here, to run once:

pinMode(Beeper, OUTPUT); // connected to buzzer +

pinMode(Ckey, INPUT\_PULLUP); // s1 button

pinMode(Dkey, INPUT\_PULLUP); // s2 button

pinMode(Ekey, INPUT\_PULLUP); // s3 button

pinMode(Fkey, INPUT\_PULLUP); // s4 button

pinMode(Gkey, INPUT\_PULLUP); // s5 button

}

void loop() {

reading = digitalRead(Ckey); // gets an accurate reading of the Ckey pin/ button status

if (KeyPressed(Ckey)){

tone(Beeper, C4freq);

delay(quarterNote); // delay so can user play a quarter note at least

}

reading = digitalRead(Dkey); // reads pin too see if button pressed

if (KeyPressed(Dkey)){ // sees if button pressed

tone(Beeper, D4freq, quarterNote); // plays sound

delay(quarterNote); // delay so can user play a quarter note at least

}

reading = digitalRead(Ekey); // reads pin

if (KeyPressed(Ekey)){ // sees if button pressed

tone(Beeper, E4freq);// plays sound

delay(quarterNote); // delay so can user play a quarter note at least

}

reading = digitalRead(Fkey); // reads pin

if (KeyPressed(Fkey)){ // is button pressed?

tone(Beeper, F4freq);// note sound

delay(quarterNote); // delay so can user play a quarter note at least

}

reading = digitalRead(Gkey); // read the button

if (KeyPressed(Gkey)){ // if button pressed

tone(Beeper, G4freq, quarterNote);

delay(quarterNote); // so button can put a note in

}

else

{

noTone(Beeper); // turns off tone if button not pressed

}

}

bool KeyPressed(int key){ // this displays true if button pressed

reading == digitalRead(key); // to double check digital reading

if (reading == 1){ // if not pressed

return false;

}

else { // if pressed

return true; // if button pressed

}

}

**Arduino Project 3 – Making A Full Octave Keyboard Instrument (*A Better Organ*)**

**Logic Design Table:**

**(1 = button pressed, 0 = button not pressed, X = don’t care)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Button**  **S1** | **Button**  **S2** | **Button**  **S3** | **Button**  **S4** | **Button**  **S5** | **Music Note** |
| 0 | 0 | 0 | 0 | 0 | (None) |
| 1 | 0 | 0 | 0 | 0 | C4 |
| 0 | 1 | 0 | 0 | 0 | D4 |
| 0 | 0 | 1 | 0 | 0 | E4 |
| 0 | 0 | 0 | 1 | 0 | F4 |
| 1 | 0 | 0 | 0 | 1 | G4 |
| 0 | 1 | 0 | 0 | 1 | A4 |
| 0 | 0 | 1 | 0 | 1 | B4 |
| 0 | 0 | 0 | 1 | 1 | C5 |

**Your Arduino sketch:**

// Arduino Keyboard

//by Jacqueline Radding

//plays a few notes a or a combination of notes when a button(s) is pressed

const int Ckey = 6 ; // Pin for S1 acting as C4 key

const int Dkey = 5 ; // Pin for S2 acting as C4 key

const int Ekey = 4 ; // Pin for S3 acting as C4 key

const int Fkey = 3 ; // Pin for S4 acting as C4 key

const int Gkey = 2 ; // Pin for S5 acting as C4 key

int reading;

//bool f; // this is the true or false value

const int Beeper = 9 ; // Digital Pin 9 for buzzer

const int quarterNote = 500 ; // msec for quarter note

const int C4freq = 262 ; // Hz for c4 note

const int G4freq = 392 ; // Hz for d4 note

const int F4freq = 349 ; // Hz for c4 note

const int E4freq = 330 ; // Hz for d4 note

const int D4freq = 294 ; // Hz for c4 note

const int A4freq = 440 ; // Hz for c5 note

const int B4freq = 494 ; // Hz for c5 note

const int C5freq = 523 ; // Hz for c5 note

void setup() {

// put your setup code here, to run once:

pinMode(Beeper, OUTPUT); // connected to buzzer +

//digitalWrite(Beeper, HIGH);

pinMode(Ckey, INPUT\_PULLUP); // s1 button

pinMode(Dkey, INPUT\_PULLUP); // s2 button

pinMode(Ekey, INPUT\_PULLUP); // s3 button

pinMode(Fkey, INPUT\_PULLUP); // s4 button

pinMode(Gkey, INPUT\_PULLUP); // s5 button

}

void loop() {

//reading = digitalRead(Ckey);

//reading = digitalRead(Dkey);

//reading = digitalRead(Ekey);

//reading = digitalRead(Fkey);

//reading = digitalRead(Gkey);

reading = digitalRead(Gkey); // reads G and E keys

reading = digitalRead(Ekey); // reads Ekey pin

if (KeyPressed(Gkey) && KeyPressed(Ekey)){ // if 2 buttons: G and Ekey pressed

tone(Beeper, B4freq, quarterNote);// plays sound

delay(quarterNote); // delay to keep sound going

}

reading = digitalRead(Gkey); // reads G pin

reading = digitalRead(Fkey); // reads F pin

if (KeyPressed(Gkey) && KeyPressed(Fkey)){ // if button G and Fkey pressed

tone(Beeper, C5freq, quarterNote);// plays sound

delay(quarterNote); // delay to keep sound going

}

reading = digitalRead(Ckey);

if (KeyPressed(Ckey)){

tone(Beeper, C4freq);

delay(quarterNote); // delay so can user play a quarter note at least

}

reading = digitalRead(Dkey); // reads pin too see if button pressed

if (KeyPressed(Dkey)){ // sees if button pressed

tone(Beeper, D4freq, quarterNote); // plays sound

delay(quarterNote); // delay so can user play a quarter note at least

}

reading = digitalRead(Ekey); // reads pin

if (KeyPressed(Ekey)){ // sees if button pressed

tone(Beeper, E4freq);// plays sound

delay(quarterNote); // delay so can user play a quarter note at least

}

reading = digitalRead(Fkey); // reads pin

if (KeyPressed(Fkey)){ // is button pressed?

tone(Beeper, F4freq);// note sound

delay(quarterNote); // delay so can user play a quarter note at least

}

reading = digitalRead(Gkey); // read the button

if (KeyPressed(Gkey)){ // if button pressed

tone(Beeper, G4freq, quarterNote);// plays sound

delay(quarterNote); // delay to keep sound going

}

else

{

noTone(Beeper); // turns off tone if button not pressed

}

}

bool KeyPressed(int key){ // this displays true if button pressed

reading == digitalRead(key); // to double check digital reading

if (reading == 1){ // if not pressed

return false;

}

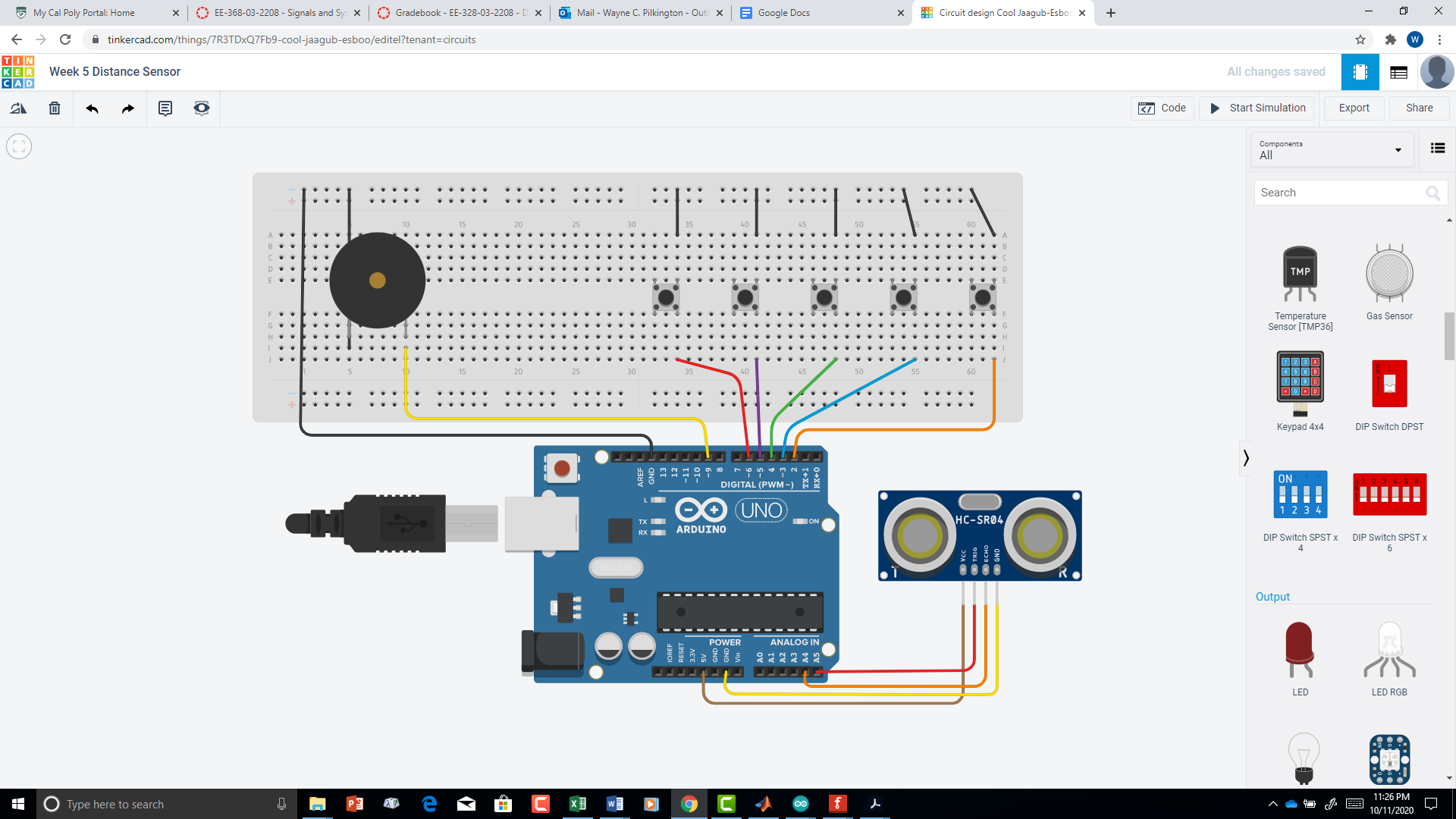
else { // if pressed

return true; // if button pressed

}

}

**Arduino Project 4 – Connecting and Characterizing the Distance Sensor**



Prepare an Arduino sketch to characterize the Obstacle Sensor that includes the following features:

1. Trigger a range measurement on the ultrasonic range sensor
2. Measure the echo time.
3. Convert the echo time into the distance the object is away from the robot.
4. Display the echo time and the computed distance on the Serial Monitor.

**Your Arduino sketch:**

//Sonar distance measuring

// by Jacqueline Radding

//This program measures the distance the sonar is from an object in inches

// trig pin is A5

int echoTime = 0; // time it takes to reach device

float distance = 0.0; // distance from device

void setup(){

pinMode(A5, OUTPUT);// trig pin

pinMode(A4, INPUT); // echopin

// Check Range Sensor by pulsing the trigger to create a PING

Serial.begin(9600); // sets serial up for data

}

void loop(){

// Check Range Sensor by pulsing the trigger to create a PING

digitalWrite(A5, LOW); // pulses the trigPin

delayMicroseconds(2); // delay

digitalWrite(A5, HIGH);//pulses trigPin

delayMicroseconds(10); //delay

digitalWrite(A5, LOW);//Pulses trigPin

MeasureDistance();

if (echoTime > 300 && echoTime < 4000){ // this makes sure there is an accurate reading because arduino is not accurate below 300 miliseconds and above 4 seconds

Serial.println("Distance");

Serial.println(distance);

delay(1000);

}

if(echoTime < 300 || echoTime > 4000){ // 300 miliseconds or less means there is little accuracy

Serial.println("Distance");

Serial.println("0"); // prints zero to be accurate

delay(10);

}

}

float MeasureDistance(){ // produces distance float and calculatons

echoTime = pulseIn(A4, HIGH);//recieves sound and its time

distance = (echoTime / 74.0) / 2.0; // Convert time to inches one-way// Speed of sound is 0.0135 inches/ microsecond// or 74 microseconds per inch.

}

**Arduino Project 5 – Ultrasonic Theremin**



**Specification:**

Purpose: Play tones with pitches proportional to the distance the player’s hand is from an ultrasonic distance sensor, when their hand is within the range of 2 – 22 inches away (for reliable readings). Otherwise, play no tones.

Inputs: Ultrasonic Sensor Echo Pulse – Arduino Analog Pin A4

Outputs:

Ultrasonic Sensor Trigger – Arduino Analog Pin A5

Passive Buzzer – Arduino Digital Pin 9

Features:

* No tone output if hand is not in proper range
* The same tone frequency should not be repeatedly commanded if it is not changing, as this makes the sound “choppy”. Only output a new tone( ) command if the frequency needs to change or stop.
* Use the MeasureDistance() function from the previous project.
* Use the map( ) command to make the tone frequency higher as the distance gets smaller. Map the distance range of 2 inches to 22 inches to frequency range 1047 Hz (C6) to 175 Hz (F3). WARNING: map() uses integer math and so it rounds all numbers. Therefore, if you map the distance values in inches they will only give you 20 different frequencies (since they will round to 2, 3, 4, 5… inches before mapping). To get more frequencies out of the mapping, map(100\*Distance,…) from a range of 200 (instead of 2 inches as the lower bound) and 2200 (instead of 22 inches as the upper bound for distances) to the frequency range 1047-175 Hz.

Pseudocode:

In setup():

Initialize pins as needed

In loop():

Measure distance to hand*. (using your function)*

*If* Distance > 0

*Then* Map the distance to a proportional frequency*.*

*If* the new frequency != old frequency

*Then* Output a new tone at the new frequency

*Else* Do Nothing (leave the old tone playing)

*Else (if Distance = 0 )*

*Turn off the tone*

Wait briefly before checking again

*// End of Program*

**Your Arduino sketch:**

**/// Theremin**

**//by Jacqueline Radding**

**//plays notes based on how far your hand is away from the sonar**

**int beeper = 9; // pin buzzer is on**

**int distance; // distance from sonar**

**int echoTime; // times it takes to reach sonar**

**int sound; // this will be the distance to tone hz variable**

**int newsound; // newsound calculation**

**void setup() {**

**// put your setup code here, to run once:**

**pinMode(beeper, OUTPUT); // connected to buzzer +**

**//digitalWrite(Beeper, HIGH);**

**Serial.begin(9600);**

**}**

**void loop() {**

**digitalWrite(A5, LOW); // pulses the trigPin**

**delayMicroseconds(2); // delay**

**digitalWrite(A5, HIGH);//pulses trigPin**

**delayMicroseconds(10); //delay**

**digitalWrite(A5, LOW);//Pulses trigPin**

**MeasureDistance(); // calculates the distance**

**Serial.println(distance);**

**if (distance > 2 && distance < 22){ // this makes sure there is an accurate reading because arduino is not accurate below 300 miliseconds and above 4 seconds**

**sound = map(distance\*100,200,2200,1047, 175); //maps the range and distance**

**Serial.println(sound);**

**tone(beeper,sound);// play sound,**

**newsound = map(distance\*100,200,2200,1047, 175); // new distance is taking another calculation**

**if (sound == newsound){ // if origional sound is same as new sound**

**tone(beeper,sound); // play origional sound**

**}**

**else{**

**tone(beeper,newsound);// if there is a new distance, play new sound**

**}**

**}**

**else{**

**noTone(beeper); // turns off beeper when there is no hand infront of sonar**

**}**

**delay(50);**

**}**

**float MeasureDistance(){ // measures distance from sonar and echotime**

**echoTime = pulseIn(A4, HIGH);//recieves sound and its time**

**distance = (echoTime / 74.0) / 2.0; // Convert time to inches one-way// Speed of sound is 0.0135 inches/ microsecond// or 74 microseconds per inch.**

**}**

**Arduino Project 6 – Chromatic Ultrasonic Theremin**



**Your Arduino sketch:**

/// Theremin

//by Jacqueline Radding

//plays notes based on how far your hand is from the sonar in the C major

const int C4freq = 262 ; // Hz for c4 note

const int G4freq = 392 ; // Hz for d4 note

const int F4freq = 349 ; // Hz for c4 note

const int E4freq = 330 ; // Hz for d4 note

const int D4freq = 294 ; // Hz for c4 note

const int A4freq = 523 ; // Hz for c5 note

const int B4freq = 523 ; // Hz for c5 note

const int C7freq = 2093 ; // Hz for c7 note

// Initialize needed local variables.

int Frequency = 0;

int FreqMultiplier = 1;

int newNoteFrequency;

int beeper = 9; // pin buzzer is on

int distance; // distance from sonar

int echoTime; // times it takes to reach sonar

int ToneNum; // this will be the distance to tone hz variable

void setup() {

// put your setup code here, to run once:

pinMode(beeper, OUTPUT); // connected to buzzer +

//digitalWrite(Beeper, HIGH);

Serial.begin(9600);

}

void loop() {

digitalWrite(A5, LOW); // pulses the trigPin

delayMicroseconds(2); // delay

digitalWrite(A5, HIGH);//pulses trigPin

delayMicroseconds(10); //delay

digitalWrite(A5, LOW);//Pulses trigPin

MeasureDistance(); // calculates the distance

Serial.println(distance);

float handDistance = MeasureDistance( ); // creates a new variable for distance

int NoteFrequency = ConvertDistanceToFrequency(handDistance); // creates frequency

if (distance > 2 && distance < 22){ // this makes sure there is an accurate reading because arduino is not accurate below 300 miliseconds and above 4 seconds

tone(beeper, NoteFrequency);

newNoteFrequency = NoteFrequency;

if (NoteFrequency == newNoteFrequency){ // if origional sound is same as new sound

tone(beeper,NoteFrequency); // play origional sound

}

else{

tone(beeper,newNoteFrequency);// if there is a new distance, play new sound

}

}

else{

noTone(beeper); // turns off beeper when there is no hand in front of sonar

}

delay(50); // small delay

}

float MeasureDistance(){ // measures distance from sonar and echotime

echoTime = pulseIn(A4, HIGH);//recieves sound and its time

distance = (echoTime / 74.0) / 2.0; // Convert time to inches one-way// Speed of sound is 0.0135 inches/ microsecond// or 74 microseconds per inch.

}

int ConvertDistanceToFrequency(float Distance\_inches) {

/\* This function converts a distance value between 2.0 – 22.0 to the frequency

of a note between C4 (at 22.0 inches) and C7 (at 2.0 inches), with the note chosen

proportional to the distance. Only notes in the C major scale are provided. It

assumes global constants are available for the note frequencies for C4 through C5.

\*/

// Initialize needed local variables.

int Frequency = 0;

int FreqMultiplier = 1;

// Map the distance value to a Tone number.

// Spanning 3 octaves (7 notes per octave) gives Tone numbers 1 - 22

int ToneNum = map(100 \* distance, 200, 2200, 22, 1);

/\*This algorithm works by using only the frequencies from the C4-B4 range.

These would be ToneNum 1-7. The frequencies for C5-B5 (ToneNum 8-14) are 2 X the

C4-B4 frequencies for the same musical notes (E5freq = 2 x E4freq). If you

subtract 7 from ToneNum 10 (E5), you get ToneNum 3 (E4). So ToneNum is reduced to

always be within 1-7, and a frequency multiplier scales up the note’s frequency to

the correct octave for the desired note (what it was before reducing ToneNum)\*/

if (ToneNum > 21) { // If note is in the 4th octave

ToneNum = ToneNum - 21; // Drop three octaves

FreqMultiplier = 8; // Double-Double-Double the frequency

}

if (ToneNum > 14) { // If note is in the 3rd octave

ToneNum = ToneNum - 14; // Drop two octaves

FreqMultiplier = 4; // Double-Double the frequency

}

if (ToneNum > 7) { // If note is in the 2nd octave

ToneNum = ToneNum - 7; // Drop 1 octave

FreqMultiplier = 2; // Double-Double the frequency

}

// Map the ToneNum to the correct 1st octave base frequency

if (ToneNum == 1) Frequency = C4freq; // Need constants for frequencies

if (ToneNum == 2) Frequency = D4freq;

if (ToneNum == 3) Frequency = E4freq;

if (ToneNum == 4) Frequency = F4freq;

if (ToneNum == 5) Frequency = G4freq;

if (ToneNum == 6) Frequency = A4freq;

if (ToneNum == 7) Frequency = B4freq;

// Scale up the note frequency to the correct octave

Frequency = Frequency \* FreqMultiplier;

return Frequency;

}

/\* This function converts a distance value between 2.0 – 22.0 to the frequency

of a note between C4 (at 22.0 inches) and C7 (at 2.0 inches), with the note chosen

proportional to the distance. Only notes in the C major scale are provided. It

assumes global constants are available for the note frequencies for C4 through C5.

\*/

**Conclusions:**

*Identify three useful lessons or learnings that you acquired while completing this lab:*

1. *I learned how to use and apply the Arduino sonar*
2. *I learned how musical notes and octaves worked and their relationship to frequency*
3. *My understanding of functions is clearer*

*Identify at least one way to improve the effectiveness of this lab in the future:*

1. *I think more pseudo code examples would make the lab run smoother*