The “Stokowski” EZ-Conduct

By Eatone Cheng

Background

In the 1949 animated film *Bugs Bunny - Long Haired Hare*, Bugs Bunny impersonates and emulates the famous conductor Leopold Stokowski by directing a band using Stokowski’s style of using his hands rather than a baton.

For this project, I will create a gadget that will allow one to also control the pitch of a musical noisemaker by simply moving your hand up and down, just like Bugs Bunny did.

Project Description

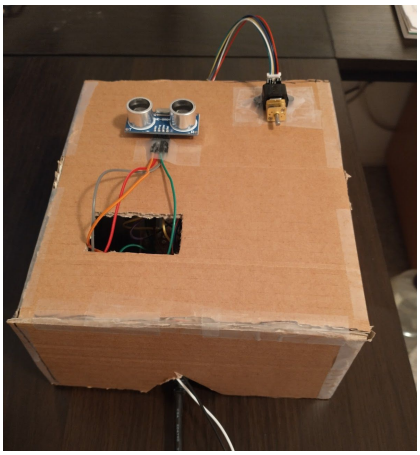
Inspired by the viral video of an F1 race car playing *God Save the Queen*[1] by throttling its engine, I sought out to use the stepper motor provided in our microkit as a noise-making apparatus. To do this, I first had to run several tests of my motor at various speeds to see which particular speed corresponded to the pitches that I wanted. Once I had these desired motor speeds, I needed a way to control them on the fly.

The microkit provided this year also includes a HC-SR04 Ultrasonic Sonar Distance Sensor, which allows you to accurately measure distances of detected objects from around 10 to 250 centimeters.

The HC-SR04 uses two ultrasonic transducers , with one that emits ultrasonic sound pulses, and one that listens for the returning pulses after they bounce off a distant object. Using this sensor, I can wave my hands, or any object with a relatively large, flat area facing the transducers to accurately send a distance measurement to my microcontroller.

1. https://www.youtube.com/watch?v=XRXwWbo_mX0

Electromechanical Details



Due to the lack of special prototyping equipment or material available to me at home, I elected to make a basic cardboard enclosure to hide the majority of the electronics. The breadboard, microcontroller, and most of the wiring is obscured by this box, partially for aesthetics, and partially to prevent any wiring from interfering from the sensor's readings. Only two objects will be visible - the motor, which makes the sound, and the sensor, to sense the waving hands/object.

There is an extra cut in the enclosure for wiring to connect my microcontroller to my computer, and the 5V barrel connector to my wall socket.

Circuit

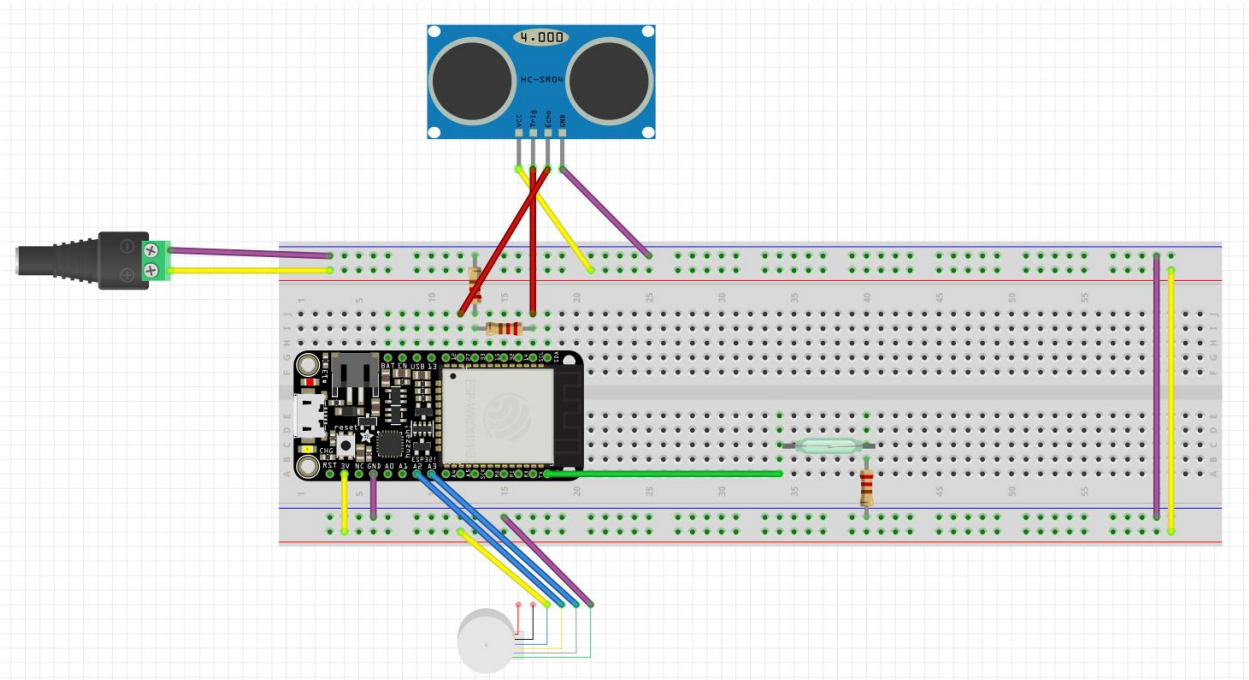
Parts:

1. Microcontroller
2. Breadboard
3. 5V Power Supply
4. Resistors (2x 10Ω, 1x 4.7kΩ)
5. DC Gearmotor (With Motor Driver)
6. Ultrasonic Sensor
7. Button Switch

The microcontroller used for this project was the Adafruit HUZZAH32 ($19.95). The two primary parts attached to this microcontroller are the HC-SR04 Ultrasonic Sensor ($3.95) and the Polulu 75:1 6V Micro Metal Gearmotor ($16.95). Both of these parts are relatively inexpensive and can be easily acquired.

The stepper motor is controlled by PWM signals from our microcontroller, and is powered by our 5V power supply barrel connector. PWM, or Pulse-Width Modulation, controls the speed of the motor by applying DC current in square-wave pulses. The wider these individual pulses, the faster the motor will spin. For example, at 75%, the wave is HIGH 75% of the time, and LOW 25% of the time. The width of individual pulses is known as the "duty cycle." In our code, we can send a 0 to 255 signal to the motor, which corresponds to 0% to 100% duty cycle.

As mentioned previously, the ultrasonic sensor sends a pulse using one of the transducers, and listens for the return with the other. In the code, it will count the amount of time passed between sending the pulse and hearing the pulse, and use that duration to calculate the distance in centimeters. I then used this distance measurement to control what PWM signal I would send to the motor.



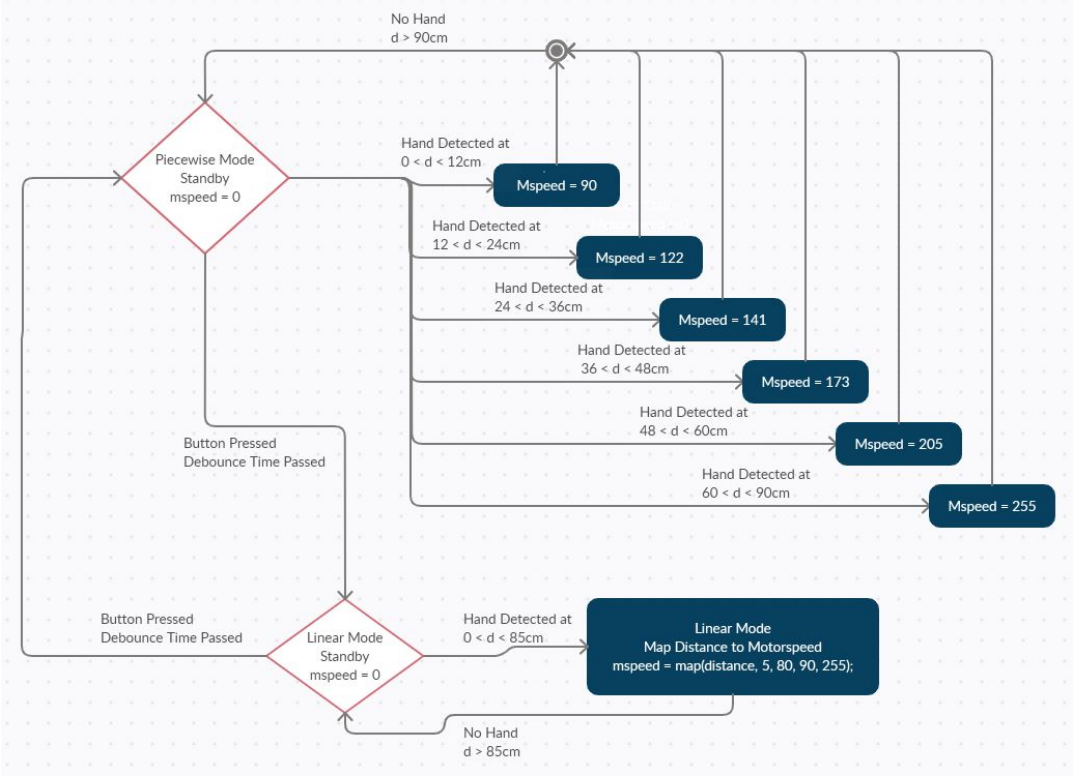
Wiring Legend:

Color	Purpose
Red	Wiring for Ultrasonic Transducer Pins
Yellow	Vcc Pins, + Terminal on Power Supply
Green	Wiring for the Switch
Blue	Wiring for the Motor Pins
Purple	GND Pins, - Terminal on Power Supply

Also, 2 sets of resistors were used. First, a 4.7kΩ resistor was used with the button as a pullup resistor. This is to limit the current to 1mA when the button is pressed, through the equation Vcc = 3.3V = 1mA / 4.7kΩ. By doing this, it will ensure that our microcontroller will detect a well-defined, controlled voltage when the switch is pressed.

The second set of resistors, 2 10Ω resistors, is used so that the sensor will send a save amount of voltage to the microcontroller. This is because the sensor is powered at 5V and returns 5V logic from its “Trig” pin, while we’re using a 3V controller. Hence, using these two resistors, we are able to reduce the 5V logic to a safe 2.5V logic for our MCU to use.

Finite State Diagram



The EZ-Conduct has two modes, a Piecewise mode and a Linear mode. In the Piecewise mode, there are 6 distance “regions” which each correspond to a particular pitch and motor command. If an object is detected in that region, the motor will be commanded to that speed. If no object is detected in any of the regions, the motor will stop.

In the Linear mode, if an object is detected in the prescribed 0 ~ 85cm region, it will linearly map that distance to a motor command of 90 to 255. Once again, if an object is not detected in this region, the motor will stop.

There is a basic ISR set up where you can switch between the two modes. When you press the button, the code will check if the designated debounce time has elapsed. If it has elapsed, it will switch modes, otherwise it will stay in its current mode.

Demonstration

I hired world-renowned conductor and musician Catone Eheng to play a rendition of Beethoven’s *Ode to Joy* using my project to demonstrate the ‘marvelous’ tunes that it can create. I hope you enjoy it!

View Here!: https://youtu.be/PjVnlYpvR0Y

Appendix

Arduino Code:

```
// Declare Motor pins
const int m1pin = 12;
const int m2pin = 13;

// Set up PWM
const int freq = 5000;
const int pwmchannel1 = 0;
const int pwmchannel2 = 1;
const int resolution = 8;
int mspeed;

// Declare Ultrasonic Sensor Pins
const int trigPin = 27;
const int echoPin = 33;
float duration, distance;

// Setup for Switch and Debounce Settings
const int switchPin = 21;
int buttstate;
int lastdeb = 0;
const int debtime = 150;
int onoff = 1;

// Switch Interrupt to Change Modes
void IRAM_ATTR pin_ISR() {
  if (millis() - lastdeb > debtime) {
    buttstate = digitalRead(switchPin);
    onoff = !onoff;
  }
  lastdeb = millis();
}

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
```

```
  pinMode(switchPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(21), pin_ISR, RISING);

  pinMode(switchPin, INPUT_PULLUP);
  pinMode(m1pin, OUTPUT);
  pinMode(m2pin, OUTPUT);

  ledcSetup(pwmchannel1, freq, resolution);
  ledcSetup(pwmchannel2, freq, resolution);
  ledcAttachPin(m1pin, pwmchannel1);
  ledcAttachPin(m2pin, pwmchannel2);
}

void loop() {
  // Detect Distance using Ultrasonic Sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration*.0343)/2;

  // Print Distance and Motor Command for Debugging
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.print(", mspeed: ");
  Serial.println(mspeed);

  // Using Distance to Select Motor Command
  if (onoff = 1) { // Piecewise Mode
    if (distance < 12) {
      mspeed = 90;
    } else if (distance > 12 && distance < 24) {
      mspeed = 122;
    } else if (distance > 24 && distance < 36) {
      mspeed = 141;
    } else if (distance > 36 && distance < 48) {
      mspeed = 173;
```

```
    } else if (distance > 48 && distance < 60) {
      mspeed = 205;
    } else if (distance > 60 && distance < 90) {
      mspeed = 255;
    } else {
      mspeed = 0;
    }
  } else if (onoff = 0) { // Linear Mode
    if (distance > 5 && distance <85) {
      mspeed = map(distance, 5, 80, 90, 255);
    } else {
      mspeed = 0;
    }
  }
  // Send Motor Command to Motors
  ledcWrite(pwmchannel1, LOW);
  ledcWrite(pwmchannel2, mspeed);
  delay(50);
}
```

Misc Images:

The costuming department spared no expense for this performance.
Included:

- Suit + Dress Shirt
- Post-it Bowtie
- Panda-head Beanie (Backwards)
- Balaclava

String I taped to the ceiling with distance markers, used to help me practice where to position my hands for the final video.