Sonar Alarm Security System

By Michael Ingerman

Description of Project:

I think that sonar sensors are really cool. In fact I have always had a deep appreciation for animals that use their biological sonar sensors to acquire food and communicate with one another. Whether it be bats or narwhals, sonar appears to have really effective applications in all kinds of medium with the correct system. As such, I decided to create a sonar based alarm system. Essentially, this system uses sonar to evaluate if a person has walked through a particular range. My system was designed to be innocently placed near a wall on the ground. As a person tries to walk past the security system, a motor with bells attached to it will begin to rotate alerting the entire house. With the pandemic at hand, this project provided good insight into the difficulties of integrating a sonar sensor into a housing without requiring a lot of external hardware.

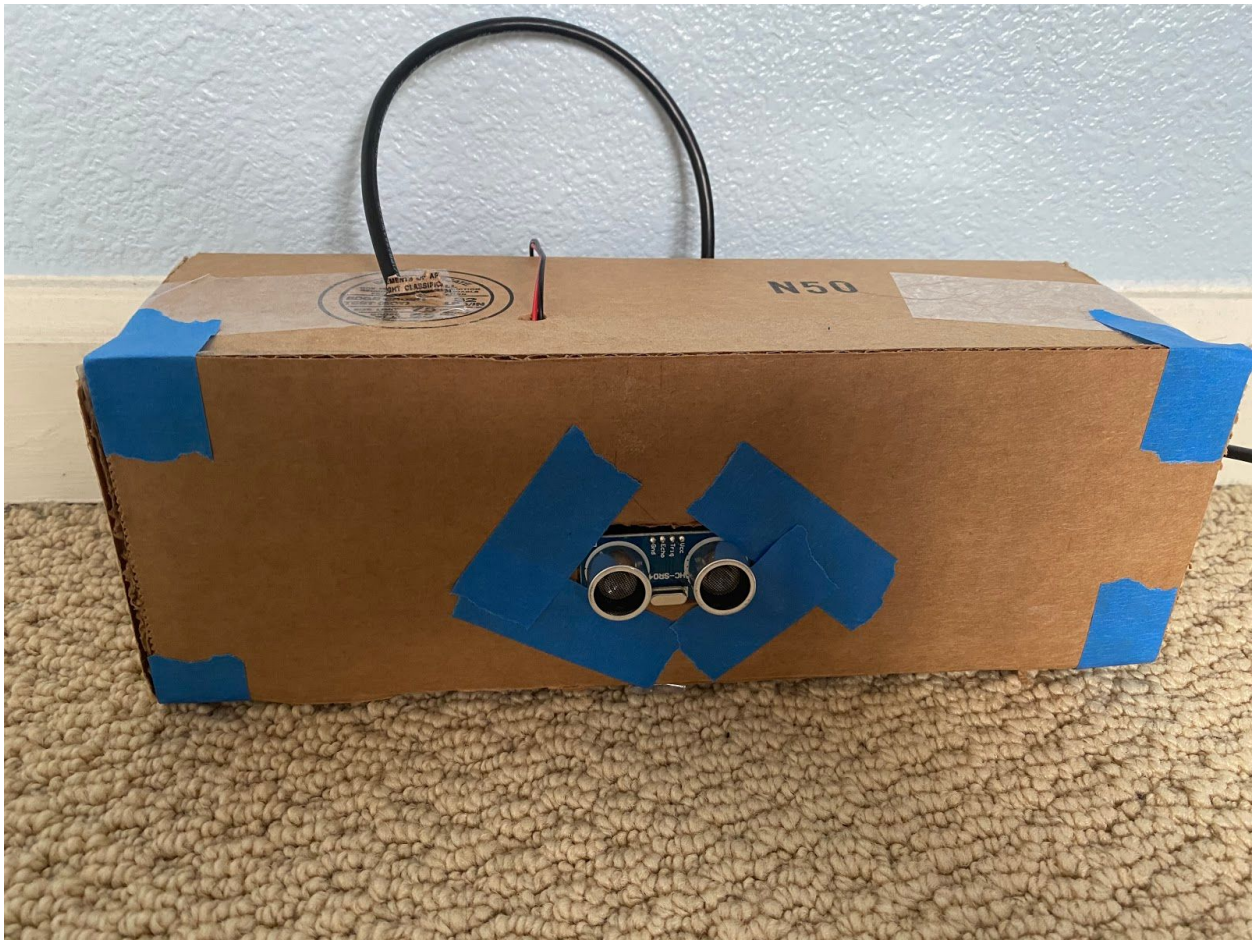

Fig 1: The sonar alarm security system has a sonar system attached to the faceplate. There are wires protruding out of the box that are responsible for providing 5V power for the sonar sensor and motor along with a USB responsible for powering the microcontroller.

Design Details:

The objective for this project was to create a completely enclosed system with one body to move. As such, all the parts of the system are consolidated to the body. Furthermore, the sonar sensor, the only external component, is mounted to the faceplate of the box. The motor with the bells along with the circuit are located inside the box. This design choice was made to protect the motor and bells from accidentally hitting anything the device is in contact with such as a wall.

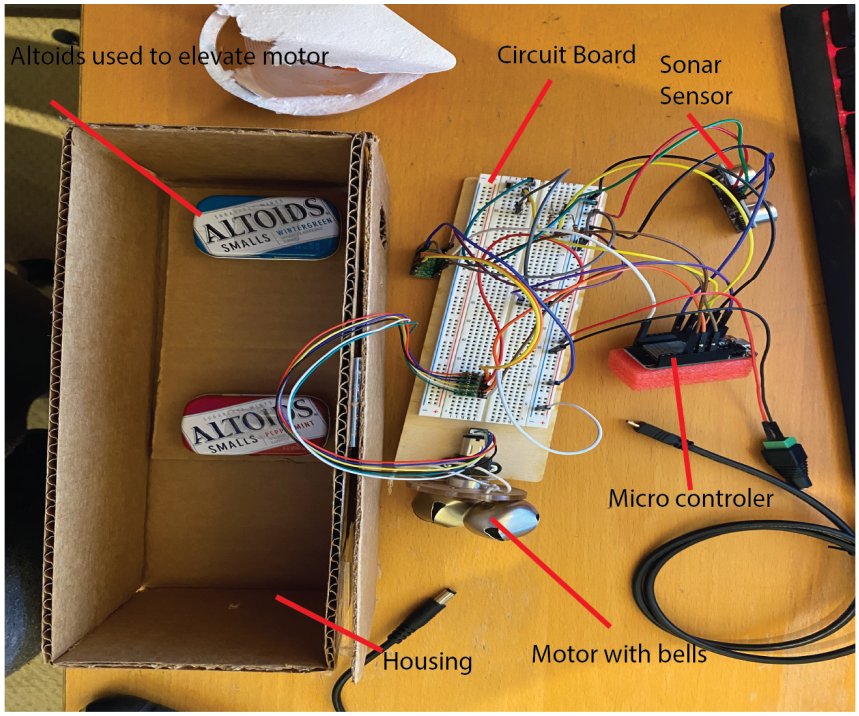

Figure 2: A dissection of the project design. The housing with altoid supports, the motor with bells attached to the motor coupler, the microcontroller, the general circuit, and the sonar sensor are all labeled above.

The housing for the device was repurposed from cardboard packaging boxes. Additionally, empty altoid containers were glued on to create adequate distance/support for the motor along with motor faceplate to rotate without interference from the housing. Holes were punched in through the side to create access points for the 5V power line for the components and for the usb responsible for powering the microcontroller.

Circuit:

The main task for this project was to understand how to integrate a sonar sensor into a design. As such, the main circuit components used for this project included: (1) Sonar Sensor, (2) a DC brushless motor and (3) motor controller, both used in previous assignments.

(1) Sonar: I found an ultrasonic sensor (~$4) inside our lab kit. The sensor has 4 inputs: ground, trigger (output), echo (input) , ground. The sensor requires 5V power to function, as such two 10k ohm resistors were used to convert the output signal from 5V logic to 2.5V logic for the microcontroller.

(2) Motor: I used the Brushed DC motor(~$16) from our lab kit. The motor has a 75:1 transmission ratio. Attached the motor is a motor shaft hub(~$6) used to attach a circular plate from which the bells are attached.

(3) Motor controller: I used an H-bridge (~$5), from our lab kit. This motor controller can switch 2.7V - 10.8V which far exceeds the input from the 5V power supply.

In addition to these main components, I repurposed cardboard packaging to create the housing. I attached bells ($7) to the motor plate to create noise. I also added a temporary reset button for debugging purposes, however this component was not used in the final product. I also used a 5V 2A power supply (~$7) to power my main components.

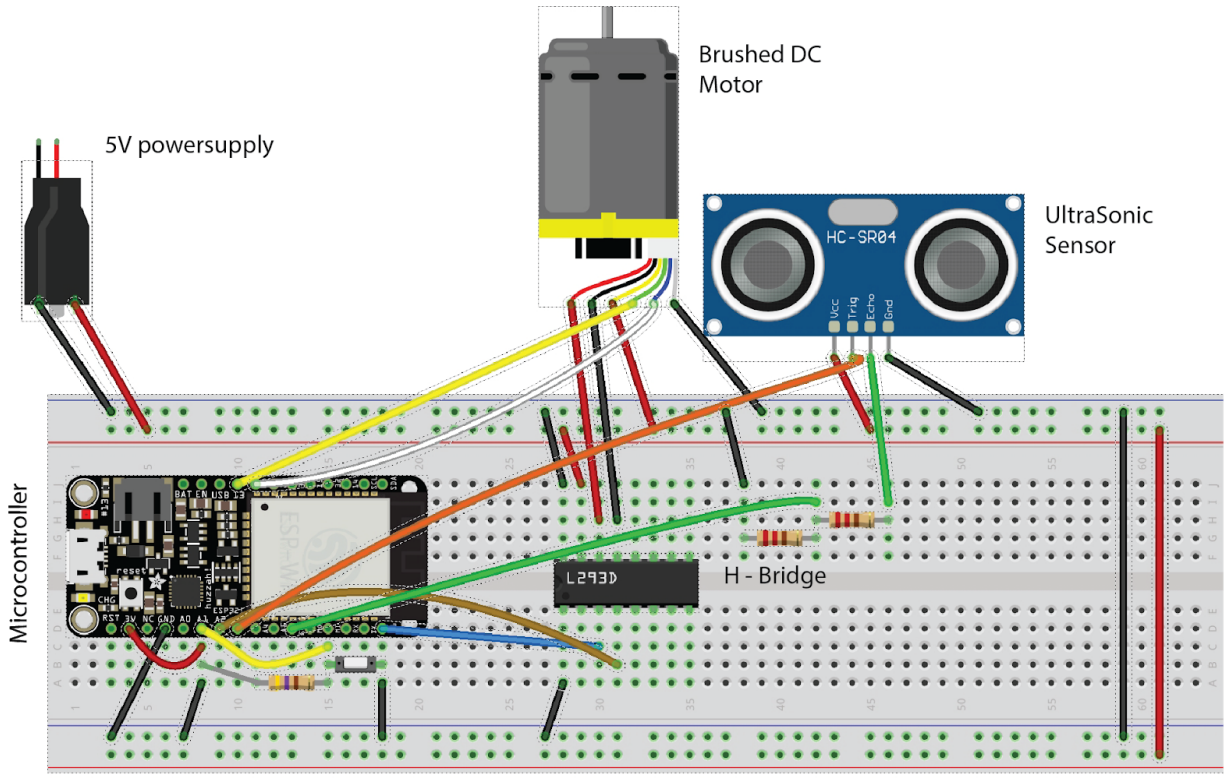

Fig 3: Circuit diagram with components labeled.

Finite State Diagram:

The device behavior is simple, if the sonar sensor detects a distance measurement that is less than the distance of the hallway the motor turns on.

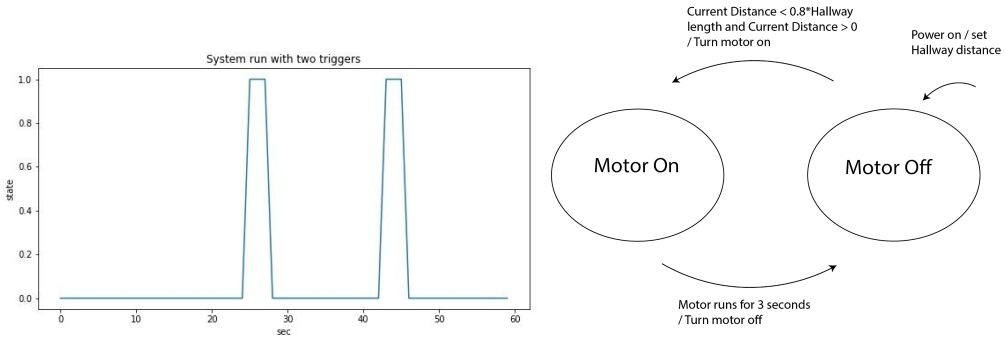

Fig 4: Testing of the FS behavior. On the left is a test run for the device with two triggers from objects moving through the sonar sensors range. The right is a FSD, describing the high level processing of the device logic.

The complete arduino IDE code can be found in Appendix I. I included a 3 second delay in the main loop when the device triggers the motors to move. In general, embedded systems do not use delay features as freezing the system is not ideal. However, since this device is responsible for just responding to triggers from a single sonar sensor, using delay doesn't affect the performance of the device and serves as a means of making sure that triggers don't stack up from an object constantly sitting in front of the device.

Appendix I: Code

```
/*
 * Sonar Alarm Security System  code by Michael Ingerman
 */

#include <NewPing.h>
#include <ESP32Encoder.h>


//PWM channel
#define freq 28000
#define resolution 8
#define channel1 0
#define channel2 1


// Motor Driver
#define A2in 21
#define A1in 15

// Encoder Setup
ESP32Encoder encoder;
#define encoder_1 13 //first pin to micro from encoder
#define encoder_2 27 // second pin to micro from encoder
int encoder_count;// encoder count
unsigned long encoderlastToggled;
double rot_coeff = 0.06666666666;
long encoder_last_time;// last time step at encoder count
double rpm; // rad per sec
static uint32_t last_time_pressed = millis();
uint32_t rollover = 0;

//POT and Button Variables
int MODE = 1; //Initializes MODE to setting 1, meaning the I/O starts
in a mode where the POT changes brightness.
int POTVALUE = 0; // Initializes POTVALUE to 0.
float desiredSpeed = 0;
int dutyCycle = 100;
int actual_dutyCycle = 0;
```

```
//Sonar setup
NewPing sonar(A5, A3, 250);
int initialDistance;
int currentDistance;

//Timer to update current distance
hw_timer_t * timer = NULL;


void IRAM_ATTR updateDist() {
  currentDistance = sonar.ping_cm();
}


void IRAM_ATTR buttonISR() {

  if(initialDistance == 0) {
    initialDistance = sonar.ping_cm();
  }
  else {
    initialDistance = 0;

  }
  Serial.print("hallway distance changed: ");
  Serial.println(initialDistance);
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(2000);


  //Motor Driver Inputs
  ledcSetup(channel1, freq, resolution);
  ledcSetup(channel2, freq, resolution);
```

```
  ledcAttachPin(A1in, channel1);
  ledcAttachPin(A2in, channel2);



  //Timer for distance check
  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &updateDist, true);
  timerAlarmWrite(timer, 1000000, true);
  timerAlarmEnable(timer);

  //Button setup
  pinMode(A1, INPUT);
  attachInterrupt(A1, buttonISR, HIGH);


   //Sonar Setup
  initialDistance = sonar.ping_cm();
//  Serial.println("");
//  Serial.print("The distance of hallway!: ");
//  Serial.println(initialDistance);


}

void loop() {
  // put your main code here, to run repeatedly:
  if (currentDistance < initialDistance*0.8 & currentDistance > 0.
20*initialDistance) {
    Serial.print("Intruder Alert at: ");
    Serial.print(currentDistance);
    Serial.print(" which is less than ");
    Serial.println(initialDistance);

    alarmTriggered();
  }
  else {

//    Serial.println("No intruder spotted {-_-} ");
```

```
  }

}

void alarmTriggered() {
  Serial.println(" ಠ_ಠ SOUNDING THE ALARM!!!! ಠ_ಠ");
  ledcWrite(channel2,   200);
  ledcWrite(channel1, 0);
  delay(3000);
  ledcWrite(channel2, 0);
  ledcWrite(channel1, 0);
  delay(1000);
}
```