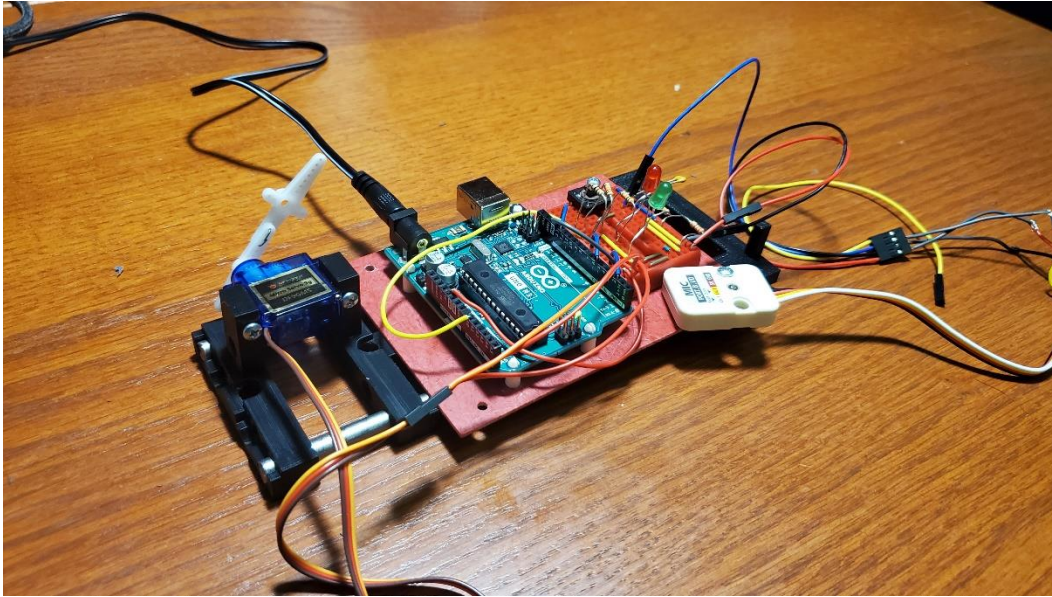


Overview



The purpose of this device is to record an alarm sound and, upon hearing that alarm sound again, actuate a positional servo motor to click a key on a laptop keyboard, thereby awaking the laptop from a sleep state.

This actuation can also be manually accomplished via a short press of the included button.

The actual recording of the alarm sound can be accomplished via a long press of the included button. A green LED turns on to indicate recording has begun, and then is turned off and replaced with a red LED to indicate the recording is complete. These LEDs are both grounded through the same resistor for current control. Though ordinarily this would limit the brightness, since both are never on at the same time, their brightness is not limited. Furthermore, the roughly 2V voltage drop across each LED in operation means that the reverse bias on one LED when the other is active is roughly 3V, less than the reverse breakdown voltage for LEDs ($\sim 5V$). When all LEDs are off, the device has returned to normal operation, listening for the alarm and waiting for a button press.

The recognition of the alarm sound is accomplished via counting the number of zero-crossings of an analog signal from the included microphone within the runtime of a certain loop. In order to produce a reliable value for zero, a high-pass filter removes the DC signal from the microphone on top of which the AC sound sinusoid is overlaid, and then a voltage divider between +5V and 0V sets that sinusoid's "zero" to roughly 2.5V, depending on the exact resistances of the two resistors in the divider. This characteristic number is correlated with the frequency of the sound. This number has been found to be highly reproducible, and as such is used as a recognition characteristic

for certain sounds when the number as calculated for the heard sound is within a certain tolerance of the number as calculated for the recorded alarm.

Design Considerations

Mechanically, this project is relatively simple, and all assembly is designed with the maximum possible degrees of freedom in order to accommodate minor errors in the dimensions of purchased components. Nevertheless, measurements should be taken carefully of all purchased components in order to ensure that no interferences exist and to minimize the number of post-3D printing modifications that need to be made.

If I were to redo this project with the same constraints, the first change I would make is a more accurate measurement of the hole spacing on the board to which the Arduino is glued. Although the spacing was good enough to assemble, a better measurement would have made assembly easier. I would actually, however, recommend offsetting one hole in the 3D printed support (Figure C.3) to which the board mounts, in order to produce, in effect, a distorted thread mechanism in the nuts that holds the nuts to the bolts and the nuts to the 3D printed support without the use of glue. I would also research the microphone I purchased more carefully. Ultimately, after buying a \$4 microphone, I had nearly \$6 left over. A higher-quality, higher-gain microphone could likely have improved the abilities of this device. I could also have included reverse-bias protection diodes on the LEDs or simply connected them to separate resistors. Another possible improvement, which I only learned about the day before the project was due, is using interrupts to recognize the button press instead of the several instances of polling I had to ensure adequate response time.

If I had more time and more money, I would replace the inset nuts in the 3D printed support (Figure C.3) with threaded inserts. Although threaded inserts are not individually more expensive, they were not provided at a discount as part of the craft supplies. I would again purchase a nicer microphone – the quality of the microphone signal is essential to the purpose of this device. I would also develop a better frequency-recognition algorithm. Using a Fast Fourier Transform or Fast Hartley Transform on the audio signal could likely produce a much more reliable frequency characteristic for the sound, but the necessary testing to validate that would have required too much time. I might also build the entire structure from a single 3D printed piece of plastic (Figure C.3, C.4, C.5, C.6 joined into one). The use of steel rods reduced cost at the expense of weight and assembly.

Assembly Instructions

Tasks to be completed before these instructions apply are as follows: Program the Arduino, connect the circuit as per Figure B.1, manufacture 3D Printed Support 1, 2, and 3 as well as the steel rods as per Figures C.3, C.4, C.5, C.6. Each step corresponds to a cell in Table C.1, reading left to right and top to bottom:

1. With the 2 steel rods (Figure C.6) parallel on a table, place 3D Printed Support 1 (Figure C.3) on top of them such that the semicylindrical channels in 3D Printed

Support 1 lock onto the rods and the side of the 3D Printed Support closest to the vertical protrusion is flush to the ends of the rods.

2. Insert one 4-40 nut into each of the hexagonal insets in 3D Printed Support 1 (Figure C.3)
3. Slide the 2 copies of 3D Printed Support 2 (Figure C.4) onto the opposite end of the steel rods. These should be oriented such that the step down is facing upward and the steps on each one of the 3D Printed Supports are not facing one another. This will be a friction fit. The spacing between them is arbitrary right now, but the 3D Printed Support 2 (Figure C.4) farthest from 3D Printed Support 1 (Figure C.3) should be flush to the end of the steel rods.
4. Glue one copy of 3D Printed Support 3 (Figure C.5) to each of 3D Printed Support 2 (Figure C.4). The side glued down should be the smallest face, closest to the larger of the two holes in 3D Printed Support 3 (Figure C.5). The small hole is offset on the face through which it passes. The side toward which it is closer should be the side oriented toward the 3D Printed Support 2 (Figure C.4) to which it is not attached. For positioning along the long axis of 3D Printed Support 2 (Figure C.4), the faces of 3D Printed Support 3 (Figure C.5) and 2 (Figure C.4) should be flush.
5. Place the board containing the Arduino and the breadboard on top of the assembly such that the holes in the board closest to the breadboard are aligned with the two nuts inserted into 3D Printed Support 1 (Figure C.3) and such that the board extends from there toward the 3D Printed Support 2s (Figure C.4). Place the microphone unit such that one of its holes is concentric with one of the nuts inset into 3D Printed Support 1 (Figure C.3).
6. Insert screws through the board and through the microphone unit hole into the nuts inset in 3D Printed Support 1 (Figure C.3).
7. Place the servo motor such that the output faces out away from 3D Printed Support 2 (Figure C.4) and the two mounting holes are aligned with the small holes in 3D Printed Support 3 (Figure C.5). Move the 3D Printed Support 2 (Figure C.4) as necessary to accomplish this. Screw one 4-40 bolt through each of the small holes in the 3D Printed support 3s (Figure C.5) into the mounting holes in the servo motor. The holes in 3D Printed Support 3 should be small enough that the bolt self-taps through the plastic.
8. Assembly complete.

Operation Instructions

To manually actuate the keyboard key click, short press the button.

To record an alarm sound, begin playing the alarm sound and long press the button. When the green LED turns on, recording has begun. Wait until the green LED is off and the red LED has turned on, and then stop playing the alarm sound. Once the red LED has turned off, normal operation has resumed and the device is ready to either receive a short button press, record a new alarm, or recognize the existing alarm.

When the device hears the recorded alarm, it will actuate the keyboard key click.

Appendix A: Bill of Materials

| Item | Part Number | Vendor | Cost per Unit | Qty | Total Cost |
|-----------------------------------|--------------|------------------------|---------------|-----|------------|
| <i>Purchased/ Scavenged Items</i> | | | | | |
| Microphone U096 | 221-U096-ND | Digikey | \$ 3.95 | 1 | \$ 3.95 |
| Steel Round Rod 0.25"x2' | 8920K114 | McMaster-Carr | \$ 2.83 | 1 | \$ 2.83 |
| 3D Printing Job PLA | N/a | Cornell RPL | \$ 1.00 | 1 | \$ 1.00 |
| 3D Printing Plastic PLA | N/a | Cornell RPL | \$ 0.40 | 5.5 | \$ 2.20 |
| 4-40 Hex Screw, 0.5" | N/a | Cornell Craft Supplies | \$ 0.01 | 4 | \$ 0.04 |
| 4-40 Hex Nut | N/a | Cornell Craft Supplies | \$ 0.02 | 2 | \$ 0.04 |
| Superglue | 1739050 | Amazon: Loctite | \$ 4.16 | 1 | \$ 4.16 |
| <i>Kit Items</i> | | | | | |
| Arduino Uno R3 | 1050-1024-ND | Digikey | \$ 20.90 | 1 | \$ 20.90 |
| 4-wire harness | 1568-1931-ND | RobotShop.com | \$ 1.35 | 1 | \$ 1.35 |
| Micro Servo Positional | SER0006 | DFRobot | \$ 3.30 | 1 | \$ 3.30 |
| Mini Breadboard | 98AC7296 | Newark | \$ 1.05 | 1 | \$ 1.05 |
| Tactile Switch Push Button | 155380 | Jameco | \$ 0.35 | 1 | \$ 0.35 |
| Wire Kit | B07PQKNQ22 | Amazon: Austor | \$ 2.17 | 1 | \$ 2.17 |
| Resistor 220 Ohm | 220QBK-ND | Digikey | \$ 0.01 | 1 | \$ 0.01 |
| Resistor 1 kOhm | 1.0kQBK-ND | Digikey | \$ 0.01 | 1 | \$ 0.01 |
| Resistor 10 kOhm | 10kQBK-ND | Digikey | \$ 0.01 | 3 | \$ 0.03 |
| Capacitor 1 uF | BC1162CT-ND | Digikey | \$ 0.18 | 1 | \$ 0.18 |
| Red LED | 697602 | Jameco | \$ 0.05 | 1 | \$ 0.05 |
| Green LED | 334086 | Jameco | \$ 0.08 | 1 | \$ 0.08 |
| Total Purchased/Scavenged | | | | | \$ 14.22 |
| Total | | | | | \$ 43.70 |

Table A.1: Bill of Materials

Appendix C: CAD Images, Drawings, and Assembly Drawings

Section C.1: CAD Images

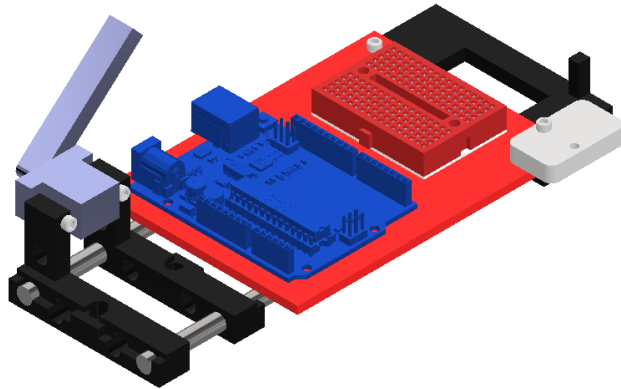


Figure C.1: Full Assembly, Top Isometric View
Arduino board in blue, breadboard and supporting board in red
Microphone in white, servo in periwinkle
3D Printed Support 1 at back in black
3D Printed Support 2 (2x) at front bottom in black
3D Printed Support 3 (2x) attached to servo in black
4-40 screws in white
Steel Rods in gray

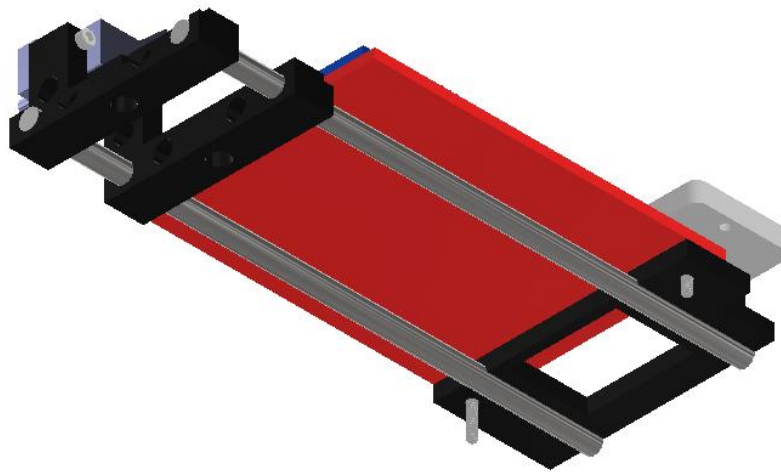


Figure C.2: Full Assembly, Bottom Isometric View

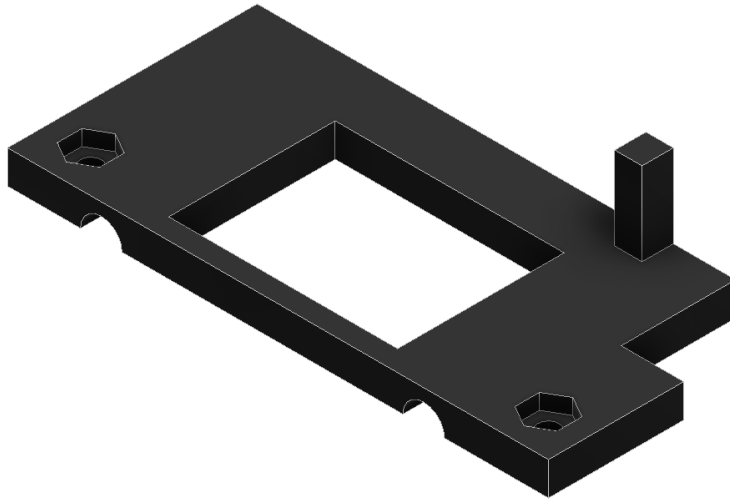


Figure C.3: 3D Printed Support 1

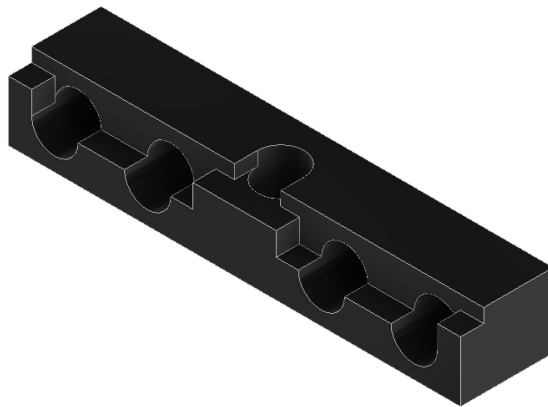


Figure C.4: 3D Printed Support 2

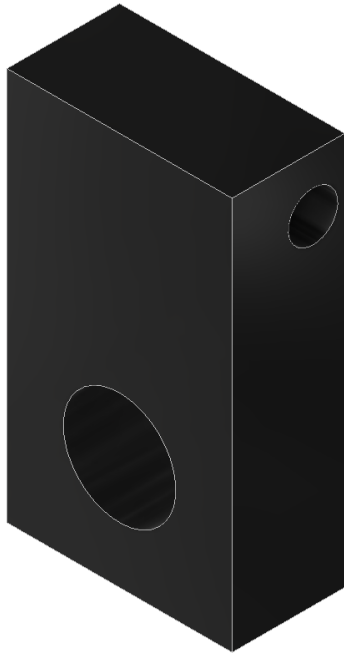


Figure C.5: 3D Printed Support 3

Section C.2: Manufacturing Drawings

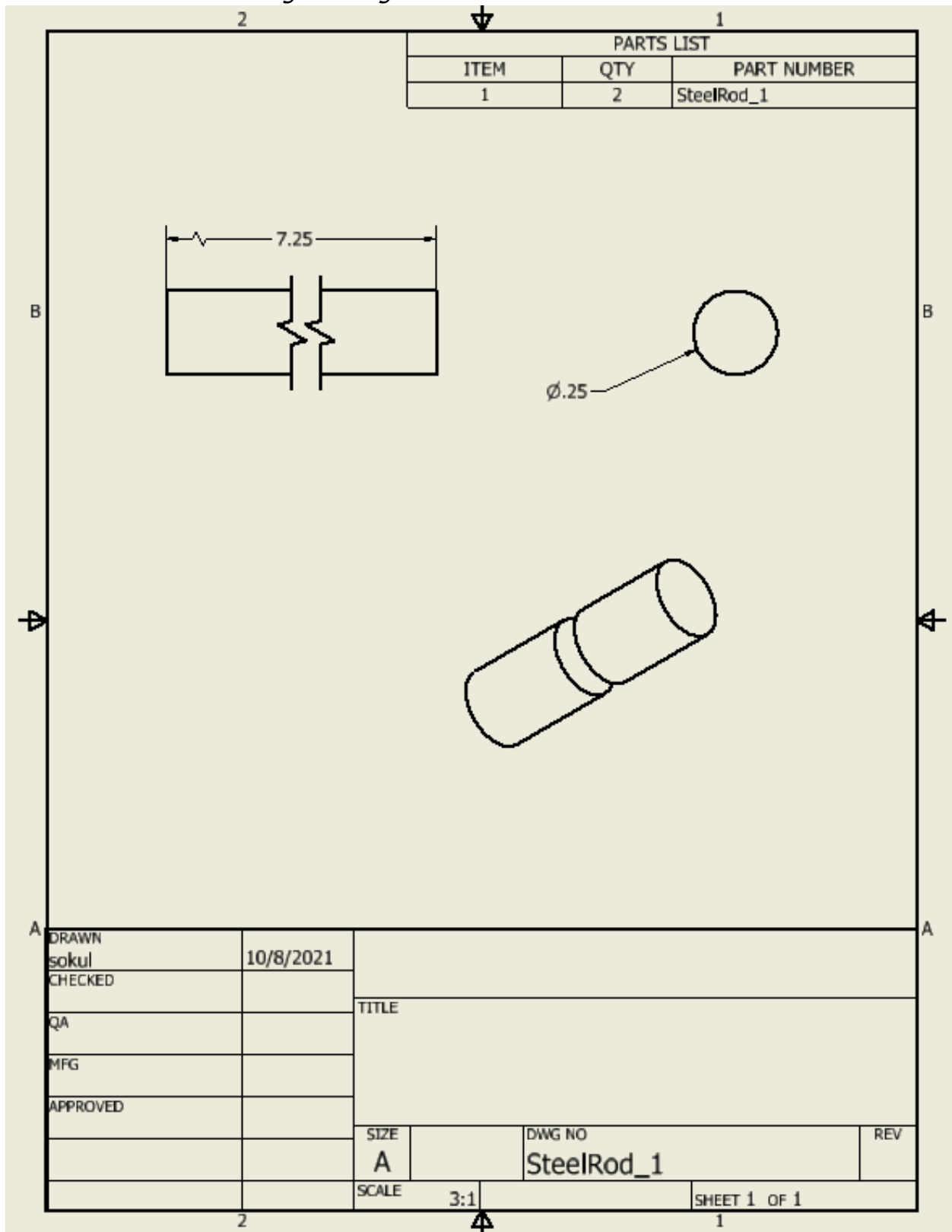


Figure C.6: Steel Rod Drawing

Section C.3: Assembly Drawings

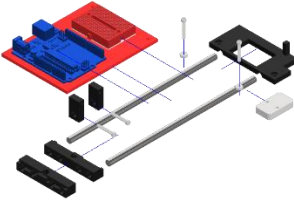
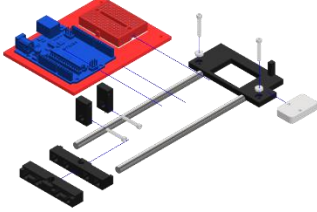
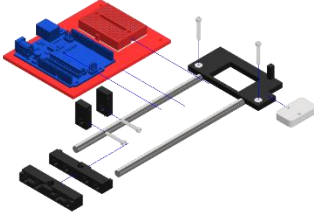
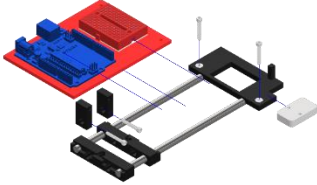
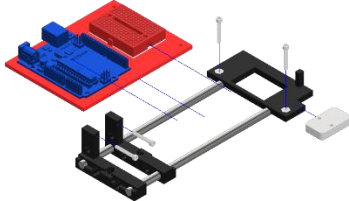
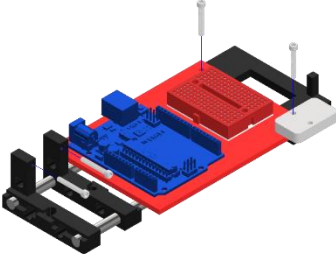
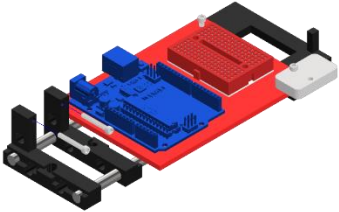
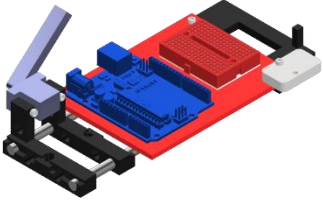
| | |
|---|--|
|  |  |
|  |  |
|  |  |
|  |  |

Table C.1: Assembly Drawings

```

/*
 * This program operates a device that can detect an alarm signal and depress a lever when that signal is detected.
 * The alarm signal can be custom recorded to the device. The function for that is record().
 * The function butPressed() handles button presses.
 * The function getSound() converts raw data from the microphone to a frequency characteristic of the recorded sound.
 * This frequency characteristic is the parameter used to compare sounds and recognize the alarm.
 */

#include <Servo.h>

int curVol; //Stores retrieved value from microphone
int pinBut = 12; //pin input from button
int pinG = 2; //pin output for green led
int pinR = 6; //pin output for red led
int pinVol = A0; //pin input from microphone
int pinServo = 3; //pin output for servo
int alarmSound; //stores frequency characteristic of alarm
int curSound; //stores current measured frequency characteristic
int fht_input[256]; //stores list of measured microphone inputs
Servo servo; //Servo motor initialization

void setup() {
  pinMode(pinVol,INPUT); //Set input for microphone
  pinMode(pinBut,INPUT); //Set input for button
  pinMode(pinR,OUTPUT); //Set output for red led
  pinMode(pinG,OUTPUT); //Set output for green led
  servo.attach(pinServo); //attach servo object to servo pin
  servo.write(180); //default to servo lever up
}

void loop() {
  if(digitalRead(pinBut)==HIGH){ //If button is pressed, run butPressed()
    butPressed();
  }
  curSound = getSound(); //retrieve frequency characteristic for currently heard sound
  if(curSound-alarmSound<50 && curSound-alarmSound>-50){ //if current freq. characteristic is similar to alarm
    servo.write(10); //servo lever to down
    delay(500);
    servo.write(180); //servo lever back up
    delay(500);
  }
}

void record() { //This function records and saves the frequency characteristic of the alarm
  delay(1000);
  digitalWrite(pinG,HIGH); //Indicate recording beginning with green led
  delay(500);
  alarmSound = getSound(); //Record alarm
  delay(2000);
  digitalWrite(pinG,LOW); //Indicate recording ending with green led off
  digitalWrite(pinR,HIGH); //Indicate recording complete with red led
  delay(200);
  digitalWrite(pinR,LOW); //Indicate return to normal device operation with red led off
  delay(100);
}

int getSound() { //This function retrieves a frequency characteristic for the current sound
  int q[10]; //Set of frequency characteristics
  int sum; //multiuse variable in summation & averaging processes
  for(int j=0;j<10;j++){ //iterating through q[]
    if(digitalRead(pinBut)==HIGH){ //jump to butPressed() if button is pressed - intended to improve response time
      butPressed();
    }
    q[j]=0; //clear any previous value of freq. characteristic
  }
  for(int i=0;i<256;i++){ //fill list of microphone inputs
    int k = analogRead(pinVol);
    fht_input[i]=k;
  }
  for(int i=1;i<256;i++){ //knowing zero point of microphone signal at 534, find number of times mic signal crosses zero
    if((fht_input[i]-534)/(fht_input[i-1]-534)<0){ //check if two consecutive inputs have opposite sign
      q[j]=q[j]+1; //store number of zero crossings in q[j] - this is a frequency characteristic
    }
  }
  sum=0; //clear any previous value of sum
}

```

```
for(int i=0;i<10;i++){//iterate through [q]
  sum=sum+q[i]; //sum all frequency characteristics from q[]
}
int avg_h=sum/10; //average freq. characteristics from q[]
return avg_h; //return freq. characteristic
}

void butPressed() { //what to do if the button is pressed
  delay(500); //wait to check if long press or short press
  if(digitalRead(pinBut)==HIGH){ //if long press
    record(); //record alarm signal
  }
  else{ //if short press
    servo.write(10); //servo lever down
    delay(500);
    servo.write(180); //servo lever up
    delay(500);
  }
}
```