Ultrasonic Range Finder Alarm

Introduction:

The necessity of keeping physical items secure and private has been prevalent in human nature as long as people have wanted to protect what is rightfully theirs, a naturally territorial instinct. The earliest alarming systems for these individuals could have been composed of just a trip wire and a hatchet that comes barreling down a string line shaft towards the unwelcome intruders. This is an example of a purely mechanical device. Fast forward to 20th century and mechanics has given way to electronics. The trip wire has been replaced with a digital sensor and the primitive and barbaric hatchet going into the face of an attempted burglar has been replaced by a more humane (and legal) electronic sound alarm system. The benefits of this sound alarm system are clear, as anyone who’s even witnessed a car alarm in action can testify. These alarms deter (or more specifically, highly annoy and make uncomfortable) any would intruder to someone’s physical property. This property can consist of a home, car, safe deposit box, or anything the user wishes to protect and be alerted in case of any unwanted meddling. With improvements to digital electronics, these alarm systems have been made smaller, more efficient, and more difficult to disable by trespassers.

Objective:

I wish to show that with an introductory level knowledge of electrical engineering, I can
build a small and efficient alarm system. My alarm will shoot out ultrasonic frequencies and measure the distance of the nearest object in front sensor. When something gets too close to the sensor (an intruder nearing your home or car) or when the sensor reads a distance that is too far (someone opens a box or disturbs the location placement of the sensor), an audible alarm noise and blinking lights will go off. This will warn the intruder and alert the user (its necessary that the user is near the vicinity of the alarm).

Bill of Materials:

- 1 Arduino Duemilanove micro controller
- 1 AC/DC adapter for the Arduino (or any external power source)
- 1 Parallax “PING” Ultrasonic Range Finder
- 1 Speaker (5V)
- 2 Led Lights
- 1 Breadboard
- 5 Copper Wires

Hardware Design:
Interfacing the hardware for the alarm is rather simple. First, connect the sensor to the breadboard. The breadboard is necessary for allowing the user enough ports to ground when connecting all the parts to the Arduino. Next, connect the Ground port on the sensor to the 5V port on the Arduino with a wire. Do this for the 5V port on the sensor to Ground port on the Arduino and Signal port on the sensor to the Digital 7 port on the Arduino port as well. Your sensor hardware is now completely interfaced with the Arduino (software interfacing is done in the next section). Now, connect the + port on your Speaker to a Ground port on the Arduino and – port to the Digital 9 port. This takes care of your Speaker hardware. Finally, connect LED 1 to the Digital 13 port and Ground port which should be side to side. The flat edge of the LED should go in the Ground port. Additionally, connect LED 2 to the breadboard with wires.
connecting the LED to Digital port 11 and Ground on the Arduino.

**Software Design:**

I needed a program to do exactly what I wanted accomplished in my objectives section; a sensor trips a combination of sound and lights when set off. This is what I came up with:

```cpp
int ultraSoundSignal = 7;
int val = 0;
int ultrasoundValue = 0;
int timecount = 0;
int ledPin = 13;
int speakerPin = 9;
int led2Pin = 11;

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(speakerPin, OUTPUT);
}

void loop() {
{
    timecount = 0;
    val = 0;
    pinMode(ultraSoundSignal, OUTPUT);

digitalWrite(ultraSoundSignal, LOW); // Send low pulse
delayMicroseconds(2); // Wait for 2 microseconds
digitalWrite(ultraSoundSignal, HIGH); // Send high pulse
delayMicroseconds(5); // Wait for 5 microseconds
digitalWrite(ultraSoundSignal, LOW); // Holdoff

    pinMode(ultraSoundSignal, INPUT); // Switch signal pin to input
    val = digitalRead(ultraSoundSignal); // Append signal value to val
    while(val == LOW) { // Loop until pin reads a high value
        val = digitalRead(ultraSoundSignal);
    }
}
```
while(val == HIGH) {
    val = digitalRead(ultraSoundSignal);
    timecount = timecount +1;
}

if(timecount > 0){
    digitalWrite(ledPin, HIGH);
    delay(250); //LED on for 250 microseconds
    digitalWrite(ledPin, LOW);
}

delay(100);

{ ultrasonicValue = timecount;
}

if (ultrasoundValue > 800)
{
    digitalWrite(led2Pin,HIGH);
    digitalWrite(speakerPin,HIGH);
    delayMicroseconds(500);
    digitalWrite(speakerPin,LOW);
    delayMicroseconds(500);
}
else
{
    digitalWrite(speakerPin,LOW);
    digitalWrite(led2Pin,LOW);
}
}

The program allows the Sensor to send out ultrasonic pulses, record the time the pulse takes to hit an object and come back, and then compares this value in order to determine whether or not the Speaker the LED 2 should turn on. The purpose of LED 1 is to ensure that the Sensor is active, LED 1 only blinks when Sensor is recording values. This program is specifically designed to go off when someone opens a box that the sensor has been placed in (more generally, if the
environment of the Sensor has been disturbed). If the sensor reads a value more than 800 microseconds (divide this value by 79 and then 2 to convert this time into a distance according to the speed of sound), or an object that is more than about 5 inches away (the size of my box), then the alarm turns on. The alarm stays on until the box is closed again.

**Design Results:**

This design of the alarm works best when the user is in the nearby vicinity. Although this seems either unpractical or undesirable, an example of this alarm safely warning the user of an unwanted intrusion is for those users’s who own guns and have children. Every year, tragic deaths result for families with gun owners and children that, through their own resourceful means, manage to stumble upon the families firearms. The alarm would be useful in this case by warning an adult within the nearby vicinity that someone had opened the container that alarm is placed in and allowing the adult to act accordingly.

The alarm proves also useful in alarming users when someone has come close to private piece of property, like a house or car. This change simply requires a flip of the inequality sign in the software code and a change in the desired distance for the alarm to activate. The alarm will turn off when the unwanted visitor has reestablished a safe distance away from the private property, highly useful for those users wishing to avoid false alarms.

Although the design is very basic and does the objective of what I want the alarm to do, there are still many areas available for improvement. Chiefly, I would like to improve the sound portion of the alarm by writing a code that produces a more audible and recognizable sound alarm. My lack of understanding of advance programming techniques (such as creating a loop within another loop) disables me from currently writing this code. Additionally, I would also like
to develop an external switch mechanism to either enable or disable the alarm, much like how car alarms are set. This would decrease the need for direct user contact with the alarm when things such as false alarms occur. Sustainability is also an area of concern, but I have little knowledge of conservative power systems at this moment.

**Conclusion:**

I consider the project a great success considering that I met all my desired objectives. Further knowledge in developing software code would help in improving my design and is an area I wish to further review in the future.