Touchless Door Sensor | Version 2.0

Russell MacGregor

```ino
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

//****Pin initializations****
int buzzer = 4;
int led = 9;
int button = 7;

int trigPin = 6;
int echoPin = 5;

//****Variables****
//Ultrasonic Sensor
long duration;
int distance;

//Program
int mode = 0;
bool distanceSet = false;
bool boundsExceeded = false;
int closedDoorDistance;
int upperBound;
int lowerBound;
int timeRemaining;
int index = 0;

//****Settings****
int tolerance = 20;
int countdownTime = 10; //time in seconds it takes for the alert to get set off after distance exceeds acceptable bounds

void setup() {
  //LCD Initialization
  lcd.init();
  lcd.backlight();

  //Serial monitor initialization
  Serial.begin(9600);

  //pinMode Initialziation
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(led, OUTPUT);
  pinMode(button, INPUT_PULLUP);

  //Code to run once:
  mode = 1;
  timeRemaining = countdownTime;
}

void loop() {
  getDistance();

  if (mode == 1) {
    runConfig();
  }

```

```
  if (mode == 2) {
    runSurveillance();
  }

  if (mode == 3) {
    runCountdown();
  }

  if (mode == 4) {
    runAlert();
  }
}

void returnToConfig(){
  mode = 1;
  ledOff();
  buzzerOff();
  lcd.clear();
  timeRemaining = countdownTime;
  distanceSet = false;
  boundsExceeded = false;
  delay(100);
}

void returnToSurveillance() {
  mode = 2;
  ledOff();
  buzzerOff();
  lcd.clear();
  timeRemaining = countdownTime;
  delay(10);
}

void displayDistance(){
  lcd.home();
  lcd.print("Distance: ");
  lcd.print(distance);
  lcd.print("   ");
  delay(500);
}

void displayButtonNotPushed(){
  lcd.setCursor(0,1);
  lcd.print("distance not set");
}

void displaySurveillanceScreen(){
  lcd.setCursor(0,1);
  lcd.print("Set, monitoring ");
}

void displayCountdownScreen(){
  lcd.setCursor(11,1);
  lcd.print("    ");
  delay(50);
  lcd.setCursor(0,1);
  lcd.print("Countdown: ");
  lcd.print(timeRemaining);
  Serial.print("Countdown: ");
  Serial.print(timeRemaining);
  Serial.print("\n");
}

void displayAlertScreen(){
  lcd.setCursor(0,1);
```

```
    lcd.print("Close door");
  }

  //Configuration mode
  void runConfig(){
    displayDistance();
    displayButtonNotPushed();

    if (digitalRead(button) == LOW){
      closedDoorDistance = distance;
      upperBound = closedDoorDistance + tolerance;
      lowerBound = closedDoorDistance - tolerance;
      mode = 2;
      lcd.clear();
      delay(10);
    }
  }

  //Surveillance
  void runSurveillance() {

    displayDistance();
    displaySurveillanceScreen();
    checkDistanceBounds();

    if (boundsExceeded == true) {
      mode = 3;
      lcd.clear();
      delay(10);
    }

    if (digitalRead(button) == LOW){
      returnToConfig();
    }
  }

  //Countdown
  void runCountdown() {

    ledOn();
    displayDistance();
    displayCountdownScreen();
    checkDistanceBounds();

    if (boundsExceeded == true) {
      if (timeRemaining == 0) {
        mode = 4;
        lcd.clear();
        delay(10);
      }

      if (timeRemaining != 0) {
        timeRemaining = timeRemaining - 1;
        delay(1000);
      }
    }

    if (boundsExceeded == false) {
      returnToSurveillance();
    }

    if (digitalRead(button) == LOW){
      returnToConfig();
    }
  }
```

```
188
189  //Alert
190  void runAlert() {
191    buzzerOn();
192    displayDistance();
193    displayAlertScreen();
194    checkDistanceBounds();
195
196    if (boundsExceeded == false) {
197      returnToSurveillance();
198    }
199
200    if (digitalRead(button) == LOW) {
201      returnToConfig();
202    }
203  }
204
205  void ledOn() {
206    digitalWrite(led, HIGH);
207  }
208
209  void ledOff() {
210    digitalWrite(led, LOW);
211  }
212
213  void buzzerOn() {
214    tone(buzzer, 1000);
215    delay(50);
216
217    noTone(buzzer);
218    delay(50);
219
220  }
221
222  void buzzerOff() {
223    noTone(buzzer);
224  }
225
226  void getDistance() {
227    // Clears the trigPin
228    digitalWrite(trigPin, LOW);
229    delayMicroseconds(2);
230
231    digitalWrite(trigPin, HIGH);
232    delayMicroseconds(10);
233    digitalWrite(trigPin, LOW);
234
235    duration = pulseIn(echoPin, HIGH);
236
237    distance = duration * 0.034 / 2;
238  }
239
240  void checkDistanceBounds(){
241    if ( (distance < lowerBound) || (distance > upperBound) ){
242      boundsExceeded = true;
243    }
244    else{
245      boundsExceeded = false;
246    }
247  }
```