NAVODAYA INSTITUTE OF TECHNOLOGY, RAICHUR

DEPARMENT OF COMPUTER SCIENCE & ENGINEERING

# IOT Lab

# Program - 03

---

## 03 Develop a program to deploy an intrusion detection system using Ultrasonic and sound sensors

🔧 What you'll need

- Arduino Uno/Nano (any 5V Arduino)
- HC-SR04 ultrasonic sensor (TRIG/ECHO)
- Sound sensor module
  - Option A: **Analog** mic module (A0 output)
  - Option B: **Digital** sound sensor (DO output, pot threshold)
- 1 × Active buzzer or LED (+ 220 Ω resistor) or Relay module (to drive a siren/light)
- Breadboard + jumper wires
- (Optional) 10–47 µF electrolytic across sensor 5V/GND for noise smoothing

---

🔌 Wiring

**Ultrasonic (HC-SR04)**

- VCC → 5V
- GND → GND
- TRIG → D9
- ECHO → D10

**Sound sensor (choose one)**

- Analog mic **AO → A0**, VCC → 5V, GND → GND
- Digital mic **DO → D4**, VCC → 5V, GND → GND

**Alarm**

- Buzzer/LED/Relay IN → D7
- Relay VCC → 5V, GND → GND (or LED to GND via 220 Ω)

If using a **relay module**, use its **COM/NO** to switch your external siren/light. Keep mains wiring isolated and handled safely.

---

🧠 Detection logic

- **Ultrasonic trip** when distance < *arm zone* (e.g., **≤ 60 cm**) for a short, consistent time.
- **Sound trip** when noise exceeds threshold (analog) or DO goes HIGH/LOW (digital depends on module).
- Alarm latches ON for a hold time (e.g., 10 s) and can auto-reset if no new events.

---

🍼 Experiment steps

1. **Build the circuit** as above.
2. **Upload the sketch** (below). Open **Serial Monitor @ 115200**.
3. **Calibrate sound threshold** (if using analog):
   - Watch "Sound:" values at quiet vs clap/speech.
   - Set `SOUND_THRESHOLD` ~ midway between quiet and event spikes.
   - If using **digital** module, adjust its trimmer until DO toggles on loud clap.
4. **Set distance threshold**: place an object/person at intended boundary and note "Dist: X cm". Set `DIST_THRESHOLD_CM` a bit larger than that.
5. **Test**:
   - Wave a hand/human in front (inside threshold) → alarm.
   - Clap or make a sharp noise → alarm.
6. **(Optional) Relay test**: replace buzzer/LED on D7 with relay IN; switch a low-voltage lamp to visualize.
7. **(Optional) Reduce false triggers**: increase `SAMPLES` for averaging, tweak `QUIET_TIME_BEFORE_RESET_MS`, or add felt/foam to mic to desensitize.

---

💻 Arduino code (handles both analog or digital sound sensors)

```
/* Intrusion Detection with Ultrasonic + Sound Sensor
   - HC-SR04 on TRIG D9, ECHO D10
   - Sound sensor: Analog (A0) or Digital (D4) -> enable one path
below
   - Alarm output (buzzer/LED/relay): D7
*/

const uint8_t PIN_TRIG = 9;
const uint8_t PIN_ECHO = 10;
const uint8_t PIN_ALARM = 7;

// ---- Choose ONE sound input path ----
#define USE_ANALOG_SOUND 1  // set to 0 if using a digital DO sound
sensor
const uint8_t PIN_SOUND_A = A0;  // analog mic AO
const uint8_t PIN_SOUND_D = 4;   // digital mic DO

// ---- Tunables ----
int  DIST_THRESHOLD_CM = 60;   // trip if distance <= this
int  SOUND_THRESHOLD  = 520;  // analog 0..1023; adjust after reading
Serial
```

```
bool DIGITAL_DO_ACTIVE_LOW = true; // many DO boards pull LOW on sound
uint16_t SAMPLES = 5;        // ultrasonic averaging samples
unsigned long ALARM_HOLD_MS = 10000;  // keep alarm on after a trigger
unsigned long QUIET_TIME_BEFORE_RESET_MS = 3000;

unsigned long alarmUntil = 0;
unsigned long lastEventMs = 0;

long measureDistanceCm() {
 // Take multiple samples and return average (basic noise filtering)
 long sum = 0;
 for (uint16_t i = 0; i < SAMPLES; i++) {
  // trigger pulse: 10us HIGH
  digitalWrite(PIN_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(PIN_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);

  // pulseIn with timeout (38ms ~ >6.5m)
  unsigned long duration = pulseIn(PIN_ECHO, HIGH, 38000UL);
  if (duration == 0) {
   // timeout -> treat as far away; cap at 400 cm
   sum += 400;
  } else {
   // distance cm = (duration us * speed of sound 0.0343 cm/us) / 2
   long cm = (long)(duration * 0.0343 / 2.0);
   sum += cm;
  }
  delay(5);
 }
 return sum / (long)SAMPLES;
}

bool ultrasonicTripped(long cm) {
 return (cm <= DIST_THRESHOLD_CM);
}

bool soundTripped() {
#if USE_ANALOG_SOUND
 int val = analogRead(PIN_SOUND_A); // 0..1023
 // Simple high-pass-ish by comparing to threshold
 return (val >= SOUND_THRESHOLD);
#else
 int s = digitalRead(PIN_SOUND_D);
 return DIGITAL_DO_ACTIVE_LOW ? (s == LOW) : (s == HIGH);
#endif
}

void setAlarm(bool on) {
 digitalWrite(PIN_ALARM, on ? HIGH : LOW);
}

void setup() {
```

```
  pinMode(PIN_TRIG, OUTPUT);
  pinMode(PIN_ECHO, INPUT);
  pinMode(PIN_ALARM, OUTPUT);

#if USE_ANALOG_SOUND
  pinMode(PIN_SOUND_A, INPUT);
#else
  pinMode(PIN_SOUND_D, INPUT);
#endif

  setAlarm(false);
  Serial.begin(115200);
  delay(500);
  Serial.println(F("Intrusion Detection: Ultrasonic + Sound"));
  Serial.println(F("Calibrate thresholds using the live readout..."));
}

void loop() {
  long cm = measureDistanceCm();
  bool uTrip = ultrasonicTripped(cm);
  bool sTrip = soundTripped();

#if USE_ANALOG_SOUND
  int soundVal = analogRead(PIN_SOUND_A);
  Serial.print(F("Dist: ")); Serial.print(cm); Serial.print(F(" cm  |  "));
  Serial.print(F("Sound: ")); Serial.print(soundVal);
#else
  int s = digitalRead(PIN_SOUND_D);
  Serial.print(F("Dist: ")); Serial.print(cm); Serial.print(F(" cm  |  "));
  Serial.print(F("SoundDO: ")); Serial.print(s);
#endif
  Serial.print(F("  -> U: "));
  Serial.print(uTrip ? F("TRIP") : F("ok"));
  Serial.print(F("  S: "));
  Serial.println(sTrip ? F("TRIP") : F("ok"));

  unsigned long now = millis();

  if (uTrip || sTrip) {
    lastEventMs = now;
    alarmUntil = now + ALARM_HOLD_MS;  // latch
  }

  // Auto-reset if quiet for a while and hold time elapsed
  bool holdActive = (now < alarmUntil);
  bool quietLongEnough = (now - lastEventMs) > QUIET_TIME_BEFORE_RESET_MS;
  setAlarm(holdActive && !quietLongEnough ? true : (now < alarmUntil));

  // Small loop delay
  delay(50);
}
```