IOT A1

```
// Q1: Adaptive LED Blink with DHT11
// Blink fast if temp > 30°C, slow if temp < 20°C, medium otherwise.
#include <DHT.h>
#define DHTPIN 2      // DHT11 data pin
#define DHTTYPE DHT11 // Sensor type
#define LEDPIN 8      // LED pin
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  pinMode(LEDPIN, OUTPUT);
  Serial.begin(9600);
  dht.begin();
  Serial.println("Adaptive LED Blink - DHT11");
}
void loop() {
  // Read temperature (in Celsius)
  float t = dht.readTemperature();
  // If reading failed, try again next loop
  if (isnan(t)) {
    Serial.println("Failed to read from DHT11!");
    delay(1000);
    return;
  }
  // Decide blink speed
  int onOffDelay; // milliseconds for ON and for OFF
  if (t > 30.0) {
    onOffDelay = 100;  // fast
  } else if (t < 20.0) {
    onOffDelay = 700;  // slow
  } else {
    onOffDelay = 300;  // medium
  }
  // Debug print
  Serial.print("Temp: ");
  Serial.print(t);
  Serial.print(" °C -> delay: ");
  Serial.print(onOffDelay);
  Serial.println(" ms");
  // Blink once
  digitalWrite(LEDPIN, HIGH);
  delay(onOffDelay);
  digitalWrite(LEDPIN, LOW);
  delay(onOffDelay);
}
// Q2: Secure Button Lock (Easy Version - just press 3 times to unlock)
#define BUTTON_PIN 2
#define LED_PIN 8
int pressCount = 0;
void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP); // Button input with internal pull-up
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  Serial.println("Press button 3 times to unlock LED");
```

```
}

void loop() {

  // Check button press

  if (digitalRead(BUTTON_PIN) == LOW) {

    pressCount++;

    Serial.print("Press Count: ");

    Serial.println(pressCount);

    // Small delay to avoid multiple counts for one press

    delay(400);

    // Check if password entered (3 presses)

    if (pressCount == 3) {

      digitalWrite(LED_PIN, HIGH);   // Unlock = LED ON

      Serial.println("✅ Correct! LED Unlocked.");

    }

  }

}

// Q3: Emergency Siren (Easy Version)

// Buzzer beeps fast if hot, slow if cold

#include <DHT.h>

#define DHTPIN 2

#define DHTTYPE DHT11

#define BUZZER 8

DHT dht(DHTPIN, DHTTYPE);

void setup() {

  pinMode(BUZZER, OUTPUT);

  Serial.begin(9600);
  dht.begin();
}
void loop() {
  float t = dht.readTemperature();
  if (isnan(t)) {
    Serial.println("Failed to read from DHT11!");
    delay(1000);
    return;
  }
  int beepDelay;
  if (t > 30) {
    beepDelay = 100;   // Fast beep if too hot
  } else if (t < 20) {
    beepDelay = 700;   // Slow beep if cold
  } else {
    beepDelay = 300;   // Medium beep
  }
  Serial.print("Temp: ");
  Serial.print(t);
  Serial.print(" °C -> Beep Delay: ");
  Serial.println(beepDelay);
  digitalWrite(BUZZER, HIGH);
  delay(beepDelay);
  digitalWrite(BUZZER, LOW);
  delay(beepDelay);
}
```

```
//Q4

int sensorPin = 2;     // IR sensor output pin

int count = 0;

int segPins[] = {3,4,5,6,7,8,9}; // 7-segment pins (a-g)

// Digit patterns for 0-9 on 7-segment (common cathode)

byte digits[10][7] = {

  {1,1,1,1,1,1,0}, //0

  {0,1,1,0,0,0,0}, //1

  {1,1,0,1,1,0,1}, //2

  {1,1,1,1,0,0,1}, //3

  {0,1,1,0,0,1,1}, //4

  {1,0,1,1,0,1,1}, //5

  {1,0,1,1,1,1,1}, //6

  {1,1,1,0,0,0,0}, //7

  {1,1,1,1,1,1,1}, //8

  {1,1,1,1,0,1,1}  //9

};

void setup() {

  pinMode(sensorPin, INPUT);

  for(int i=0; i<7; i++) {

    pinMode(segPins[i], OUTPUT);

  }

}

void loop() {

  if(digitalRead(sensorPin) == LOW) {   // Object detected

    count++;
    if(count > 9) count = 0; // Reset after 9
    displayDigit(count);
    delay(500); // Debounce delay
  }
}
void displayDigit(int num) {
  for(int i=0; i<7; i++) {
    digitalWrite(segPins[i], digits[num][i]);
  }
}
//Q5
#include <DHT.h>
#define DHTPIN 3      // DHT11 data pin
#define DHTTYPE DHT11
#define PIRPIN 2      // PIR sensor output pin
#define LEDPIN 4      // LED pin
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  pinMode(PIRPIN, INPUT);
  pinMode(LEDPIN, OUTPUT);
  dht.begin();
  Serial.begin(9600);
}
void loop() {
  float temp = dht.readTemperature();   // read temperature in °C
  int pirValue = digitalRead(PIRPIN);   // motion detected = HIGH
  Serial.print("Temp: ");
  Serial.print(temp);
  Serial.print(" °C, PIR: ");
  Serial.println(pirValue);
  if (pirValue == HIGH && temp < 30) {
    digitalWrite(LEDPIN, HIGH);  // Turn ON LED
  } else {
    digitalWrite(LEDPIN, LOW);   // Turn OFF LED
  }
  delay(500); // small delay
}
//Q6
#include <Adafruit_LiquidCrystal.h>
#define TRIG_PIN 9
#define ECHO_PIN 10
// RS=2, EN=3, D4=4, D5=5, D6=6, D7=7
Adafruit_LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
long duration;
float distance, prevDistance = 0, speed;
void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  lcd.begin(16, 2);   // 16x2 LCD
  Serial.begin(9600);
  lcd.print("Ultrasonic SPD");
  delay(1000);
  lcd.clear();
}
void loop() {
  // Trigger ultrasonic pulse
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  // Measure echo
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.0343 / 2; // cm
  // Calculate speed (difference in distance per 0.5 sec)
  speed = (distance - prevDistance) / 0.5; // cm/s
  prevDistance = distance;
  // Show on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("D:");
  lcd.print(distance, 1);
  lcd.print("cm");
  lcd.setCursor(0, 1);
  lcd.print("V:");
  lcd.print(speed, 1);
  lcd.print("cm/s");
  // Also print on Serial
  Serial.print("Distance: ");
  Serial.print(distance, 1);
  Serial.print(" cm   Speed: ");
  Serial.print(speed, 1);
  Serial.println(" cm/s");
  delay(500); // update every 0.5 sec
}
//Q7
// Pins for 7-segment (a, b, c, d, e, f, g)
int segPins[] = {2, 3, 4, 5, 6, 7, 8};
// Button pin
int buttonPin = 9;
bool running = false;   // Stopwatch state
int seconds = 0;
// Digit patterns for 0–9 (a,b,c,d,e,f,g)
byte digits[10][7] = {
  {1,1,1,1,1,1,0},  // 0
  {0,1,1,0,0,0,0},  // 1
  {1,1,0,1,1,0,1},  // 2
  {1,1,1,1,0,0,1},  // 3
  {0,1,1,0,0,1,1},  // 4
  {1,0,1,1,0,1,1},  // 5
  {1,0,1,1,1,1,1},  // 6
  {1,1,1,0,0,0,0},  // 7
  {1,1,1,1,1,1,1},  // 8
  {1,1,1,1,0,1,1}   // 9
```

```
};
void setup() {
  for (int i = 0; i < 7; i++) {
    pinMode(segPins[i], OUTPUT);
  }
  pinMode(buttonPin, INPUT_PULLUP); // Button with pull-up
}
void loop() {
  // Check button (toggle stopwatch state)
  if (digitalRead(buttonPin) == LOW) {
    delay(200);   // debounce
    running = !running;
  }
  if (running) {
    seconds++;
    if (seconds > 9) seconds = 0; // Reset after 9
    showDigit(seconds);
    delay(1000); // wait 1 second
  } else {
    showDigit(seconds); // keep showing last value
  }
}
void showDigit(int num) {
  for (int i = 0; i < 7; i++) {
    digitalWrite(segPins[i], digits[num][i]);
  }
```

```
}

//Q8

#include <Adafruit_LiquidCrystal.h>

#define TRIG_PIN 9

#define ECHO_PIN 10

// RS=2, EN=3, D4=4, D5=5, D6=6, D7=7

Adafruit_LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

long duration;

float distance;

int screen = 0;    // to switch screens

int seconds = 0;   // simple time counter

void setup() {

  pinMode(TRIG_PIN, OUTPUT);

  pinMode(ECHO_PIN, INPUT);

  lcd.begin(16, 2);

  lcd.print("Dynamic LCD Menu");

  delay(1500);

  lcd.clear();

}

void loop() {

  // --- Ultrasonic distance measurement ---

  digitalWrite(TRIG_PIN, LOW);

  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);

  delayMicroseconds(10);

  digitalWrite(TRIG_PIN, LOW);
```

```
duration = pulseIn(ECHO_PIN, HIGH);
distance = duration * 0.0343 / 2; // cm
// --- Cycle screens automatically ---
screen = (screen + 1) % 3;  // total 3 screens
if (screen == 0) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Screen 1: TIME");
  lcd.setCursor(0, 1);
  lcd.print("Sec: ");
  lcd.print(seconds);
  seconds++;
}
else if (screen == 1) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Screen 2: DIST");
  lcd.setCursor(0, 1);
  lcd.print("D: ");
  lcd.print(distance, 1);
  lcd.print(" cm");
}
else if (screen == 2) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Screen 3: SYS");
```

```
    lcd.setCursor(0, 1);
    lcd.print("System OK :)");
  }
  delay(3000);  // hold each screen for 3 seconds
}
//Q9
#include <Adafruit_LiquidCrystal.h>
#define BUTTON_PIN 7    // Doorbell button
#define PIR_PIN 8       // PIR motion sensor
#define BUZZER_PIN 12   // Buzzer
// RS=2, EN=3, D4=4, D5=5, D6=6, D7=7
Adafruit_LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
int pressCount = 0;   // log number of button presses
void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(PIR_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("Smart Door System");
  delay(1500);
  lcd.clear();
}
void loop() {
  int buttonState = digitalRead(BUTTON_PIN);
  int pirState = digitalRead(PIR_PIN);
  if (buttonState == LOW) {   // button pressed (active LOW)
    pressCount++;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Visitor Detected");
    lcd.setCursor(0, 1);
    lcd.print("Count: ");
    lcd.print(pressCount);
    if (pirState == HIGH) {   // Motion detected too
      digitalWrite(BUZZER_PIN, HIGH);
      delay(500);
      digitalWrite(BUZZER_PIN, LOW);
    }
    delay(1000); // debounce delay
  }
}
//Q10
#include <Adafruit_LiquidCrystal.h>

#define TRIG_PIN 9
#define ECHO_PIN 10
#define IR_PIN 2

// LCD wiring: RS=2, EN=3, D4=4, D5=5, D6=6, D7=7
Adafruit_LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
```

```
long duration;
float distance;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(IR_PIN, INPUT);

  lcd.begin(16, 2);  // 16x2 LCD
  lcd.print("Obstacle Detect");
  delay(1000);
  lcd.clear();

  Serial.begin(9600);
}

void loop() {
  int irStatus = digitalRead(IR_PIN);

  if (irStatus == LOW) {  // Object detected by IR
    // Trigger ultrasonic
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    // Measure echo
    duration = pulseIn(ECHO_PIN, HIGH);
    distance = (duration * 0.0343) / 2;  // in cm

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Obj Dist: ");
    lcd.print(distance);
    lcd.print("cm");

    lcd.setCursor(0, 1);
    if (distance < 10) {
      lcd.print("Too Close!");
    } else {
      lcd.print("Safe Distance");
    }

    delay(500);
  } else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No Obstacle");
    delay(500);
  }
}
```

```
//Q11
#include <Adafruit_LiquidCrystal.h>

// Ultrasonic pins
#define TRIG_PIN 9
#define ECHO_PIN 10

// LCD: RS=2, EN=3, D4=4, D5=5, D6=6, D7=7
Adafruit_LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

long duration;
float distanceCM, distanceInch, distanceFeet;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  lcd.begin(16, 2);   // 16x2 LCD
  lcd.print("Digital Ruler");
  delay(1000);
  lcd.clear();

  Serial.begin(9600);
}

void loop() {
  // 1. Send ultrasonic pulse
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // 2. Read echo
  duration = pulseIn(ECHO_PIN, HIGH);

  // 3. Convert to distance
  distanceCM = (duration * 0.0343) / 2;     // cm
  distanceInch = distanceCM / 2.54;         // inch
  distanceFeet = distanceInch / 12.0;       // feet

  // 4. Show on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(distanceCM, 1); lcd.print(" cm");

  lcd.setCursor(0, 1);
  lcd.print(distanceInch, 1); lcd.print(" in ");
  lcd.print(distanceFeet, 2); lcd.print(" ft");

  delay(500);
}
```

```
//Q12
#include <Adafruit_LiquidCrystal.h>
#include <DHT.h>

// DHT sensor setup
#define DHTPIN 8
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// LCD setup (RS=2, EN=3, D4=4, D5=5, D6=6, D7=7)
Adafruit_LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

// Clock values
int hours = 0, minutes = 0, seconds = 0;
unsigned long lastSecond = 0;

void setup() {
  lcd.begin(16, 2);
  dht.begin();

  lcd.print("Weather Clock");
  delay(1000);
  lcd.clear();
}

void loop() {
  unsigned long now = millis();

  // Update time every 1 second
  if (now - lastSecond >= 1000) {
    lastSecond = now;
    seconds++;

    if (seconds == 60) { seconds = 0; minutes++; }
    if (minutes == 60) { minutes = 0; hours++; }
    if (hours == 24)   { hours = 0; }

    showTime();
    showTemp();
  }
}

void showTime() {
  lcd.setCursor(0, 0);
  lcd.print("Time ");
  if (hours < 10) lcd.print("0");
  lcd.print(hours);
  lcd.print(":");
  if (minutes < 10) lcd.print("0");
  lcd.print(minutes);
  lcd.print(":");
  if (seconds < 10) lcd.print("0");
  lcd.print(seconds);
}

void showTemp() {
  float t = dht.readTemperature();
  lcd.setCursor(0, 1);
  if (isnan(t)) {
    lcd.print("Temp --.- C  ");
  } else {
    lcd.print("Temp ");
    lcd.print(t, 1);
    lcd.print((char)223); // degree symbol
    lcd.print("C   ");
  }
}

//Q13
#define PIR_PIN 2
#define LDR_PIN A0
#define LED_PIN 9
#define BUZZER 10

unsigned long motionStart = 0;
bool motionDetected = false;

void setup() {
  pinMode(PIR_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  int pirState = digitalRead(PIR_PIN);
  int ldrValue = analogRead(LDR_PIN); // 0-1023

  bool isDark = (ldrValue < 500); // Adjust threshold

  if (pirState == HIGH && isDark) {
    digitalWrite(LED_PIN, HIGH);  // Turn on streetlight

    if (!motionDetected) {
      motionDetected = true;
      motionStart = millis();     // Start counting time
    } else {
      // Check if > 10 seconds of continuous motion
      if (millis() - motionStart >= 10000) {
        tone(BUZZER, 2000);
      }
    }
  } else {
    digitalWrite(LED_PIN, LOW);   // Light off

    noTone(BUZZER);               // Stop buzzer

    motionDetected = false;       // Reset motion tracking

  }



  delay(200); // small delay for stability

}



//Q14

#define TRIG_PIN 9
#define ECHO_PIN 10
#define GREEN_LED 11
#define YELLOW_LED 12
#define RED_LED 13
#define BUZZER 8

long duration;
int distance;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(BUZZER, OUTPUT);
}

int getDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  return duration * 0.034 / 2; // cm
}

void loop() {
  distance = getDistance();
  if (distance > 40) {
    digitalWrite(GREEN_LED, HIGH);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(RED_LED, LOW);
    noTone(BUZZER);
  }
  else if (distance > 20 && distance <= 40) {
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(YELLOW_LED, HIGH);
    digitalWrite(RED_LED, LOW);
    tone(BUZZER, 1000); delay(300);
    noTone(BUZZER); delay(300);
  }
  else if (distance > 8 && distance <= 20) {
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(RED_LED, HIGH);
    tone(BUZZER, 1500); delay(150);
    noTone(BUZZER); delay(150);
  }
  else if (distance <= 8) {
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(RED_LED, HIGH);
    tone(BUZZER, 2000); // continuous alarm
  }
}

//Q15
#include <Adafruit_LiquidCrystal.h>

#define IR_PIN 8
#define BUZZER 9
#define LED 10
#define RESET_BTN 11

Adafruit_LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {
  pinMode(IR_PIN, INPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(RESET_BTN, INPUT_PULLUP);

  lcd.begin(16, 2);
  lcd.print("Laser Security");
  delay(1000);
  lcd.clear();
}

void loop() {
  int irState = digitalRead(IR_PIN);
  int resetBtn = digitalRead(RESET_BTN);

  if (irState == LOW) { // Beam broken
    if (resetBtn == HIGH) { // alarm active until reset
      lcd.setCursor(0, 0);
      lcd.print(" *** ALARM *** ");
      tone(BUZZER, 2000);
      digitalWrite(LED, HIGH);
      delay(200);
      digitalWrite(LED, LOW);
      delay(200);
    }
  } else {
    noTone(BUZZER);
    digitalWrite(LED, LOW);
    lcd.setCursor(0, 0);
    lcd.print("System Armed   ");
  }
}



//Q16
#include <Adafruit_LiquidCrystal.h>
#include <DHT.h>

// --- Pins ---
#define DHTPIN 8
#define DHTTYPE DHT11
#define BUZZER 9
#define FAN_LED 10

// Objects
DHT dht(DHTPIN, DHTTYPE);
Adafruit_LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {
  pinMode(BUZZER, OUTPUT);
  pinMode(FAN_LED, OUTPUT);

  lcd.begin(16, 2);
  dht.begin();
  lcd.print("Thermal Alert");
  delay(1000);
  lcd.clear();
}

void loop() {
  float temp = dht.readTemperature(); // Celsius

  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  if (isnan(temp)) {
    lcd.print("--.- C");
    lcd.setCursor(0, 1);
    lcd.print("Sensor Error   ");
    digitalWrite(FAN_LED, LOW);
    noTone(BUZZER);
  } else {
    lcd.print(temp, 1);
    lcd.print((char)223); // degree symbol
    lcd.print("C   ");

    if (temp > 35.0) {
      digitalWrite(FAN_LED, HIGH);  // Fan ON
      tone(BUZZER, 2000);           // Buzzer ON
      lcd.setCursor(0, 1);
      lcd.print("Cooling ON     ");
    } else {
      digitalWrite(FAN_LED, LOW);   // Fan OFF
      noTone(BUZZER);               // Buzzer OFF
      lcd.setCursor(0, 1);
      lcd.print("Normal Temp    ");
    }
  }

  delay(1000); // update every second
}

//Q17
#include <Adafruit_LiquidCrystal.h>

// --- Pins ---
#define PIR_PIN 4
#define IR_PIN 5
#define TRIG_PIN 9
#define ECHO_PIN 10
#define BUZZER 11
#define ALARM_LED 12
```

```
// LCD: RS=2, EN=3, D4=6, D5=7, D6=8, D7=13
Adafruit_LiquidCrystal lcd(2, 3, 6, 7, 8, 13);

long duration;
float distance;

void setup() {
  pinMode(PIR_PIN, INPUT);
  pinMode(IR_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(ALARM_LED, OUTPUT);

  lcd.begin(16, 2);
  lcd.print("Home Intrusion");
  delay(1000);
  lcd.clear();
}

float getDistanceCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH, 20000UL); // timeout ~3.4m
  if (duration == 0) return -1;
  return (duration * 0.0343) / 2.0;
}

void loop() {
  int pir = digitalRead(PIR_PIN);
  int ir = digitalRead(IR_PIN);
  distance = getDistanceCM();

  if (pir == HIGH && ir == LOW && distance > 0 && distance < 100) {
    // Intruder detected
    lcd.setCursor(0, 0);
    lcd.print("INTRUSION ALERT");
    lcd.setCursor(0, 1);
    lcd.print("Object <1m     ");

    tone(BUZZER, 2000);
    digitalWrite(ALARM_LED, HIGH);
    delay(200);
    digitalWrite(ALARM_LED, LOW);
    noTone(BUZZER);
    delay(200);
  } else {
    // Safe
    noTone(BUZZER);
    digitalWrite(ALARM_LED, LOW);
    lcd.setCursor(0, 0);
    lcd.print("System Armed    ");
    lcd.setCursor(0, 1);
    lcd.print("No Motion       ");
    delay(500);
  }
}

//Q18
#include <Adafruit_LiquidCrystal.h>

// Pins for ultrasonic sensors
#define TRIG_HAND 4
#define ECHO_HAND 5
#define TRIG_TRASH 6
#define ECHO_TRASH 7

#define LID_LED 9   // LED simulating lid

// LCD object
Adafruit_LiquidCrystal lcd(2, 3, 10, 11, 12, 13);

long duration;
float distanceHand, distanceTrash;
bool lidOpen = false;

// Function to measure distance
float getDistanceCM(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long dur = pulseIn(echoPin, HIGH, 20000UL);
  if (dur == 0) return -1;
  return (dur * 0.0343) / 2.0; // cm
}

void setup() {
  pinMode(TRIG_HAND, OUTPUT);
  pinMode(ECHO_HAND, INPUT);
  pinMode(TRIG_TRASH, OUTPUT);
  pinMode(ECHO_TRASH, INPUT);

  pinMode(LID_LED, OUTPUT);

  lcd.begin(16, 2);
  lcd.print("Smart Dustbin");
  delay(1000);
  lcd.clear();
}

void loop() {
  distanceHand = getDistanceCM(TRIG_HAND, ECHO_HAND);
  distanceTrash = getDistanceCM(TRIG_TRASH, ECHO_TRASH);

  // --- Hand detection for lid (LED) ---
  if (distanceHand > 0 && distanceHand < 20) {
    digitalWrite(LID_LED, HIGH);  // Lid open
    lidOpen = true;
  } else {
    digitalWrite(LID_LED, LOW);   // Lid closed
    lidOpen = false;
  }

  // --- LCD Display ---
  lcd.setCursor(0, 0);
  if (lidOpen) {
    lcd.print("Lid: OPEN      ");
  } else {
    lcd.print("Lid: CLOSED    ");
  }

  lcd.setCursor(0, 1);
  if (distanceTrash > 0 && distanceTrash < 10) {
    lcd.print("Bin: FULL       ");
  } else {
    lcd.print("Bin: EMPTY      ");
  }

  delay(500);
}



//Q19
#include <Adafruit_LiquidCrystal.h>
#include <DHT.h>
#include <EEPROM.h>
```

```
// Pins
#define DHTPIN 8
#define DHTTYPE DHT11
#define BUZZER 9
#define LEDPIN 10

// LCD: RS=2, EN=3, D4=4, D5=5, D6=6, D7=7
Adafruit_LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
DHT dht(DHTPIN, DHTTYPE);

// EEPROM address (we’ll just keep overwriting, very simple)
int eepromAddr = 0;

unsigned long lastLog = 0;

void setup() {
  pinMode(BUZZER, OUTPUT);
  pinMode(LEDPIN, OUTPUT);

  lcd.begin(16, 2);
  dht.begin();

  lcd.print("Weather Station");
  delay(1000);
  lcd.clear();
}

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // --- Display ---
  lcd.setCursor(0, 0);
  lcd.print("T:");
  if (isnan(t)) lcd.print("--");
  else lcd.print(t, 1);
  lcd.print((char)223); lcd.print("C ");

  lcd.print("H:");
  if (isnan(h)) lcd.print("--");
  else lcd.print(h, 0);
  lcd.print("% ");

  // --- Alarm ---
  if (!isnan(h) && h > 70) {
    digitalWrite(LEDPIN, HIGH);
    tone(BUZZER, 2000);
    lcd.setCursor(0, 1);
    lcd.print("ALERT: HUMIDITY ");
  } else {
    digitalWrite(LEDPIN, LOW);
    noTone(BUZZER);
    lcd.setCursor(0, 1);
    lcd.print("Normal          ");
  }

  // --- Log every 1 minute ---
  if (millis() - lastLog >= 60000UL) {
    lastLog = millis();
    if (!isnan(h) && !isnan(t)) {
      EEPROM.update(eepromAddr, (int)h);   // store humidity (0–100)
      EEPROM.update(eepromAddr + 1, (int)t); // store temp (0–50 typically)
      eepromAddr += 2;
      if (eepromAddr >= EEPROM.length() - 2) eepromAddr = 0; // wrap around
    }
  }

  delay(1000); // update every second
}


//Q20
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>

// WiFi
const char* ssid = "YOUR_WIFI";
const char* pass = "YOUR_PASS";

// ThingSpeak
unsigned long channelID = 123456;       // your channel ID
const char* apiKey = "YOUR_API_KEY";    // write API key

WiFiClient client;

// Sensors
#define DHTPIN 2   // D4
#define PIR_PIN 14 // D5
#define TRIG 12    // D6
#define ECHO 13    // D7
DHT dht(DHTPIN, DHT11);

unsigned long lastUpdate = 0;
```

```
float getDistance() {
  digitalWrite(TRIG, LOW); delayMicroseconds(2);
  digitalWrite(TRIG, HIGH); delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
  long dur = pulseIn(ECHO, HIGH, 30000);
  return dur * 0.034 / 2;
}

void setup() {
  Serial.begin(115200);
  pinMode(PIR_PIN, INPUT);
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);

  dht.begin();
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) { delay(500); }
  ThingSpeak.begin(client);
}

void loop() {
  if (millis() - lastUpdate >= 20000) {
    lastUpdate = millis();

    float t = dht.readTemperature();
    float h = dht.readHumidity();
    int motion = digitalRead(PIR_PIN);
    float dist = getDistance();

    int alert = (motion == HIGH && t > 30) ? 1 : 0;

    ThingSpeak.setField(1, t);
    ThingSpeak.setField(2, h);
    ThingSpeak.setField(3, dist);
    ThingSpeak.setField(4, motion);
    ThingSpeak.setField(5, alert);

    if (alert) ThingSpeak.setStatus("ALERT: Hot & Motion Detected");
    else ThingSpeak.setStatus("OK");

    int code = ThingSpeak.writeFields(channelID, apiKey);
    Serial.println("Update code: " + String(code));
  }
}
```