Industrial Monitoring of Humidity Temperature Senso and Water Level Using Raspberry Pi

:

Component Pin Connections

BME280 (I2C)

VCC	3.3V (Pin 1)
GND	GND (Pin 6)
SDA	GPIO 2 (Pin 3)
SCL	GPIO 3 (Pin 5)

HC-SR04 VCC

5V (Pin 2)

GND	GND (Pin 14)
TRIG	GPIO 23 (Pin 16)
ECHO	GPIO 24 (Pin 18) (with voltage divider)

YF-S201

VCC (Red)	5V (Pin 4)
GND (Black)	GND (Pin 9)
OUT (Yellow)	GPIO 17 (Pin 11)

Relay Module

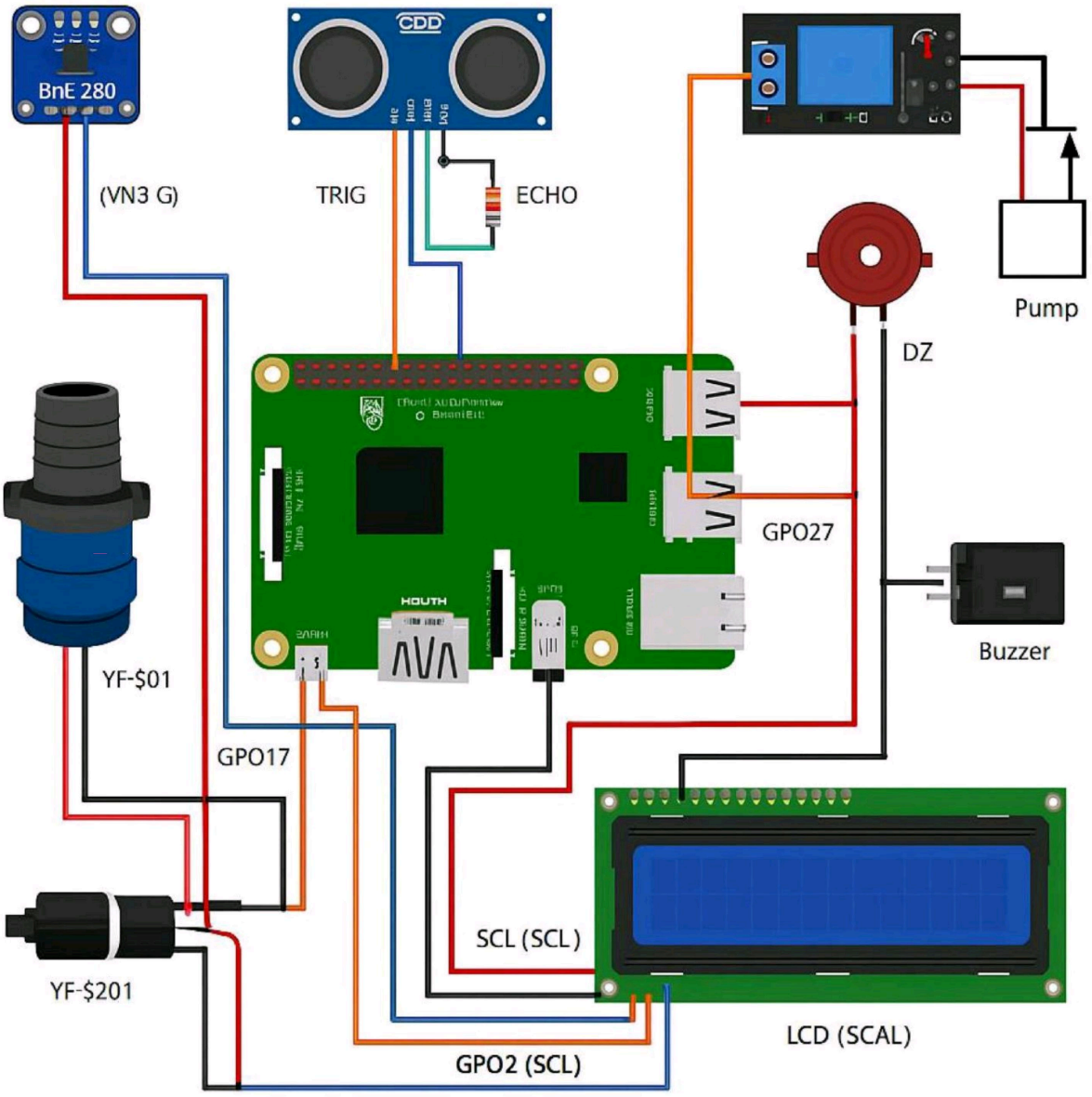
VCC	5V (Pin 2)
GND	GND (Pin 6)
IN	GPIO 27 (Pin 13)

Buzzer

Signal (+)	GPIO 22 (Pin 15)
GND (-)	GND (Pin 9)

16x2 LCD / OLED (I2C)

VCC	5V (Pin 2)
GND	GND (Pin 6)
SDA	GPIO 2 (Pin 3) (shared)
SCL	GPIO 3 (Pin 5) (shared)


BnE 280
(VN3 G)
TRIG
ECHO
Pump
DZ
GPO27
Buzzer
YF-$01
GPO17
YF-$201
SCL (SCL)
LCD (SCAL)
GPO2 (SCL)

PROGRAMING

```
import RPi.GPIO as GPIO
import smbus2
import bme280
import time
import csv
import datetime
import pyttsx3
import requests
import sqlite3
from RPLCD.i2c import CharLCD  # Added for LCD

# Initialize text-to-speech engine
tts = pyttsx3.init()

# GPIO pin setup
TRIG = 23
ECHO = 24
FLOW_SENSOR = 17
RELAY = 27
BUZZER = 22
TANK_HEIGHT_CM = 100
I2C_PORT = 1
BME280_ADDRESS = 0x76

# GPIO setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.setup(FLOW_SENSOR, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(RELAY, GPIO.OUT)
GPIO.setup(BUZZER, GPIO.OUT)
GPIO.output(RELAY, GPIO.HIGH)  # Pump initially off
GPIO.output(BUZZER, GPIO.LOW)

pulse_count = 0

def count_pulse(channel):
    global pulse_count
    pulse_count += 1

GPIO.add_event_detect(FLOW_SENSOR, GPIO.FALLING, callback=count_pulse)

# BME280 setup
bus = smbus2.SMBus(I2C_PORT)
calibration_params = bme280.load_calibration_params(bus, BME280_ADDRESS)

# LCD setup
lcd = CharLCD('PCF8574', 0x27)  # Adjust I2C address if needed
```

```
lcd.clear()
lcd.write_string("Smart Monitor Ready")
time.sleep(2)

def read_distance_cm(timeout=0.03):
    GPIO.output(TRIG, False)
    time.sleep(0.05)
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    pulse_start = time.time()
    timeout_start = pulse_start

    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()
        if pulse_start - timeout_start > timeout:
            return None

    pulse_end = time.time()
    timeout_start = pulse_end

    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()
        if pulse_end - timeout_start > timeout:
            return None

    duration = pulse_end - pulse_start
    return round(duration * 17150, 2)

def read_bme280():
    data = bme280.sample(bus, BME280_ADDRESS, calibration_params)
    return data.temperature, data.humidity, data.pressure

def send_telegram_alert(message):
    bot_token = 'YOUR_BOT_TOKEN'
    chat_id = 'YOUR_CHAT_ID'
    url = f'https://api.telegram.org/bot{bot_token}/sendMessage'
    payload = {'chat_id': chat_id, 'text': message}
    requests.post(url, data=payload)

def send_email_alert(subject, message):
    pass

def send_whatsapp_alert(message):
    pass

def log_to_db(flow_rate, water_level, temp, humidity, pressure):
    conn = sqlite3.connect("sensordata.db")
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS logs (
        timestamp DATETIME,
        flow_rate REAL,
        water_level REAL,
```

```
        temperature REAL,
        humidity REAL,
        pressure REAL
    )''')
    cursor.execute("INSERT INTO logs VALUES (datetime('now'), ?, ?, ?, ?, ?)",
                   (flow_rate, water_level, temp, humidity, pressure))
    conn.commit()
    conn.close()

try:
    print("Smart Monitoring System Started...")
    while True:
        pulse_count = 0
        time.sleep(1)
        flow_rate = pulse_count / 7.5

        temp, humidity, pressure = read_bme280()
        distance = read_distance_cm()
        if distance is None:
            print("Ultrasonic timeout")
            continue

        water_level = max(0, min(100, 100 - (distance / TANK_HEIGHT_CM * 100)))

        print(f"Flow Rate: {flow_rate:.2f} L/min")
        print(f"Water Level: {water_level:.1f} %")
        print(f"Temp: {temp:.2f} °C | Humidity: {humidity:.2f}% | Pressure: {pressure:.2f}
hPa")

        lcd.clear()
        lcd.write_string(f"Temp:{temp:.1f}C\nHumidity:{humidity:.1f}%")
        time.sleep(2)
        lcd.clear()
        lcd.write_string(f"Water:{water_level:.1f}%\nFlow:{flow_rate:.2f}L/m")

        if flow_rate > 0 and water_level < 20:
            msg = "Leakage Alert: Water is flowing but tank level is low."
            print(f"[!] {msg}")
            tts.say(msg)
            tts.runAndWait()
            GPIO.output(BUZZER, GPIO.HIGH)
            send_telegram_alert(msg)
            send_email_alert("Leak Detected", msg)
            send_whatsapp_alert(msg)
        elif flow_rate == 0 and water_level < 10:
            msg = "Dry Alert: Tank is empty and no water flow."
            print(f"[!] {msg}")
            tts.say(msg)
            tts.runAndWait()
            GPIO.output(BUZZER, GPIO.HIGH)
            send_telegram_alert(msg)
            send_email_alert("Dry Tank Alert", msg)
            send_whatsapp_alert(msg)
        else:
```

```
            GPIO.output(BUZZER, GPIO.LOW)

        if water_level > 90:
            GPIO.output(RELAY, GPIO.HIGH)  # turn OFF pump
        elif water_level < 20 and flow_rate > 0:
            GPIO.output(RELAY, GPIO.LOW)   # turn ON pump

        log_to_db(flow_rate, water_level, temp, humidity, pressure)

except KeyboardInterrupt:
    print("Exiting...")
    GPIO.cleanup()
```

SAMPLE OUT[PUT

Smart Monitoring System Started...

Flow Rate: 1.65 L/min
Water Level: 76.8 %

Temp: 28.40 ºc I Humidity: 61.80% | Pressure: 1013.05 hPa

LCD:

Temp:28.4C

Humidity:61.8% —Y (waits

2 seconds)

Water:76.8%

Flow:1.65L/m

Flow Rate: 1.80 L/min

Water Level: 18.3 %

Temp: 28.42 ºc I Humidity: 62.10% | Pressure:

1013.18 hPa > Leakage Alert: Water is flowing but

tank level is low.

> Voice Alert Triggered >

Buzzer ON > Telegram Alert

Sent > Email Sent (simulated)

> WhatsApp Alert Triggered

LCD:

Temp:28.4C

Humidity:62.1%

Water:18.3%

Flow:1.80L/m
Flow Rate: 0.00 L/min

Water Level: 9.5 %

Temp: 28.38 °c I Humidity: 61.70% | Pressure: 1013.00 hPa
[!] Dry Alert: Tank is empty and no water flow.

> Voice Alert Triggered >

Buzzer ON > Telegram Alert

Sent > Email Sent

> WhatsApp Alert Triggered

LCD:

Temp:28.4C

Humidity:61.7%

Water:9.5%

Flow:O.OOL/m

Flow Rate: 2.05 L/min

Water Level: 92.0 % Pump turned OFF

via Relay (GPIO 27)

Flow Rate: 1.35 L/min

Water Level: 14.2 %

Pump turned ON via Relay (GPIO 27)

DATABASE

TimestampFlow Rate Water Level Temperature Humidity Pressure

2025-06-23 18..45..	01 1.65	76.8	28.40	61.80	1013.05
2025-06-23 18..45..	06 1.80	18.3	28.42	62.10	1013.18
2025-06-23 18..45..	11 0.00	9.5	28.38	61.70	1013.00
2025-06-23 18..45..	16 2.05	92.0	28.45	61.90	1013.25
2025-06-23 18..45..	21 1.35	14.2	28.39	61.75	1013.10

Smart Monitoring and Control of Pressure and Power in Toyota Airjet Loom using Arduino

Connection details

Ardiuno UNO Configuration

Component	Function	Arduino Pin
MPX5700AP Pressure Sensor	Analog Pressure Input	A0
PZEM-004T	Power Monitoring (Serial UART)	D2 (RX), D3 (TX)
16x2 I2C LCD	Display	A4 (SDA), A5 (SCL)
Relay Module	Loom Motor ON/OFF Control	D5
Buzzer	Audio Alert	D6
LED	Visual Alert	D7
Bluetooth Module (HC-05)	Wireless Monitoring	D10 (TX), D11 (RX)*
Wi-Fi Module (ESP8266)	Cloud Storage (optional)	TX → D8, RX → D9 (via voltage divider)
Power Supply	12V Adapter or USB	VIN or USB

MPX5700AP

Pin on MPX5700AP	Connects to
VCC	5V on Arduino
GND	GND on Arduino
Output (Vo)	A0 on Arduino

PZEM pin configuratio

PZEM Pin	Arduino Pin
TX	D2 (RX)

RX	D3 (TX)
VCC	External 5V
GND	Arduino GND
Current Clamp	AC line
Voltage Line	AC Line

LCD pin configuration

LCD Pin	Arduino Pin
SDA	A4
SCL	A5
VCC	5V
GND	GND

Relay pin Configuration

Relay Pin	Arduino Pin	Description
IN	D5	Control signal to toggle relay
VCC	5V	Power for relay
GND	GND	Common ground
NO/NC/COM	Motor Line	Connect motor AC line through it

Buzzer Configuration

Device	Arduino Pin	Notes
Buzzer	D6	5V active buzzer
LED	D7	220Ω resistor in series

Bluethooth module

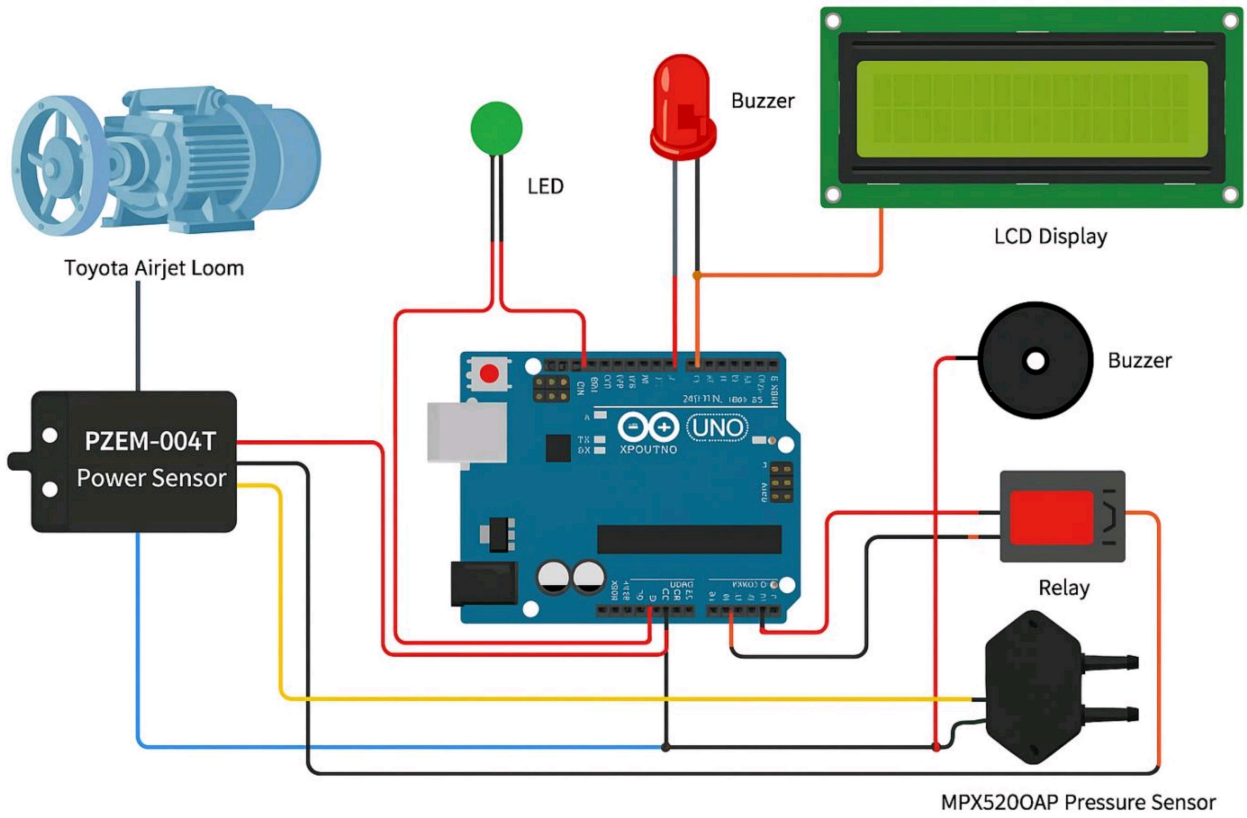
HC-05 Pin	Arduino Pin	Notes
VCC	5V	Power
GND	GND	Common Ground
TXD	D10 (Arduino RX)	Can go directly
RXD	D11 via divider	Use voltage divider (5V → 3.3V)

HC-05 Configuration

HC-05 Pin	Arduino Pin	Notes
VCC	5V	Power
GND	GND	Common Ground
TXD	D10 (Arduino RX)	Can go directly
RXD	D11 via divider	Use voltage divider (5V → 3.3V)

ESP8266 Configuration

ESP8266 Pin	Arduino Pin	Notes
VCC	3.3V	Do not connect to 5V!
GND	GND	Common ground
TX	D8 (Arduino RX)	
RX	D9 (Arduino TX) via divider	5V → 3.3V via divider
CH_PD/EN	3.3V	Pull-up resistor may be needed



Program

```
// Smart Monitoring and Control of Pressure and Power in Toyota Airjet Loom
// Using Arduino UNO, MPX5700AP, PZEM-004T, LCD, Relay, Buzzer, Bluetooth, and Cloud

#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include <PZEM004Tv30.h>

// LCD setup
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Pressure Sensor (MPX5700AP)
const int pressurePin = A0;
float voltage, pressure_kPa;
float lowerLimit = 400;  // Minimum safe pressure (kPa)
float upperLimit = 600;  // Maximum safe pressure (kPa)

// Actuators
const int relayPin = 5;
const int buzzerPin = 6;
const int ledPin = 7;
```

```
// Bluetooth
SoftwareSerial bluetooth(10, 11); // TX, RX

// PZEM Module (Power Monitoring)
SoftwareSerial pzemSerial(2, 3); // RX, TX
PZEM004Tv30 pzem(pzemSerial);

void setup() {
  pinMode(relayPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);

  lcd.begin();
  lcd.backlight();
  Serial.begin(9600);
  bluetooth.begin(9600);
}

void loop() {
  // Pressure calculation
  voltage = analogRead(pressurePin) * (5.0 / 1023.0);
  pressure_kPa = ((voltage - 0.2) / 4.5) * 700;

  // Power parameters
  float voltage_p = pzem.voltage();
  float current_p = pzem.current();
  float power = pzem.power();
  float energy = pzem.energy();
  float frequency = pzem.frequency();
  float pf = pzem.pf();

  // Display on LCD
  lcd.setCursor(0, 0);
  lcd.print("P:"); lcd.print(pressure_kPa); lcd.print("kPa");
  lcd.setCursor(0, 1);
  lcd.print("V:"); lcd.print(voltage_p); lcd.print(" I:"); lcd.print(current_p);

  // Bluetooth Output
  bluetooth.print("Pressure: "); bluetooth.print(pressure_kPa); bluetooth.println(" kPa");
  bluetooth.print("Voltage: "); bluetooth.println(voltage_p);
  bluetooth.print("Current: "); bluetooth.println(current_p);
  bluetooth.print("Power: "); bluetooth.println(power);
  bluetooth.print("Energy: "); bluetooth.println(energy);
  bluetooth.print("Frequency: "); bluetooth.println(frequency);
  bluetooth.print("Power Factor: "); bluetooth.println(pf);

  // Alert and control logic
  if (pressure_kPa < lowerLimit || pressure_kPa > upperLimit || power > 1000) {
    digitalWrite(buzzerPin, HIGH);
    digitalWrite(ledPin, HIGH);
    digitalWrite(relayPin, LOW); // Shut down loom
  } else {
    digitalWrite(buzzerPin, LOW);
    digitalWrite(ledPin, LOW);
```

```
    digitalWrite(relayPin, HIGH); // Keep loom running
  }

  delay(2000);
}
```

Output

1. LCD Display Output (16x2 I2C LCD)

- **Normal Condition (Safe Range):**
- P:520.3kPa
  V:229.4 I:4.2

- **Pressure Drop Detected (< 400 kPa):**
- P:378.6kPa
  ⚠ Low Pressure

Overpressure Detected (> 600 kPa):

P:615.7kPa
⚠ High Pressure

2. Bluetooth Serial Output (via HC-05)

- **Normal Monitoring Output:**
- Pressure: 520.3 kPa
- Voltage: 229.4
- Current: 4.2
- Power: 960.5
- Energy: 3.1
- Frequency: 50.0
  Power Factor: 0.95

- **On Fault (e.g., low pressure or high power):**
- Pressure: 378.6 kPa
- Voltage: 230.1
- Current: 5.2
- Power: 1021.0
  Status: FAULT – Loom Shutdown Triggered

3. Buzzer, LED & Relay Status Table

Condition	Buzzer	LED	Relay (Motor Control)
Normal Operation	OFF	OFF	ON (Running)
Pressure Drop (<400kPa)	ON	ON	OFF (Shutdown)

Overpressure (>600kPa)	ON	ON	OFF (Shutdown)
Power > 1000W	ON	ON	OFF (Shutdown)

4. Serial Monitor Output (Arduino IDE)

- **Normal State:**
- Pressure: 520.3 kPa
- Voltage: 229.4
- Current: 4.2
- Power: 960.5
- Energy: 3.1
- Frequency: 50.0
- Power Factor: 0.95
  System Status: NORMAL

- **Fault State:**
- Pressure: 378.6 kPa
- ⚠ ALERT: Pressure Drop Detected!
  System Status: SHUTDOWN