CODE OF AUTOMATIC WATER LEVEL CONTROLLER AND TEMPERATURE DETECTION SYSTEM

Libraries Used:

LiquidCrystal_I2C: Used to interface with the I2C LCD display.

One Wire: A library for communicating with the DS18B20 temperature sensor.

Dallas Temperature: A library that simplifies the interaction with the DS18B20 temperature sensor.

Ultrasonic: A library for interfacing with the ultrasonic distance sensor.

Wire: An essential library for I2C communication.

Arduino Board: An Arduino Uno or any other compatible board can be used for this project. The board reads inputs from the ultrasonic sensor and temperature sensor, processes the data, and sends commands to control the relay (and, ultimately, the water pump).

Hardware Components:

Arduino Uno

16x2 I2C LCD

Ultrasonic Sensor (HC-SR04)

DS18B20 Temperature Sensor

5V Relay Module

220V Water Pump

Flowchart for Program Logic

```
+-----------------------+
| Start System          |
+-----------------------+
           |
           v
```

```
+-------------------------+
| Initialize Sensors & LCD|
+-------------------------+
           |
           v
+-------------------------+
| Read Water Level & Temp|
+-------------------------+
           |
           v
+-------------------------+
| Display Data on LCD    |
+-------------------------+
           |
           v
+-------------------------+
| Check Water Level      |
+-------------------------+
      |         |
      v         v
  Below 30%    Above 97%
      |         |
      v         v
+-------------------------+
| Turn ON Pump (Relay)   |
+-------------------------+
      |         |
      v         v
  Turn OFF Pump (Relay) |
           |
           v
+-------------------------+
| Repeat Process         |
+-------------------------+
```

Below is an example of the key sections of the code used to implement the Automatic Water Level Sensor & Controller System. This code includes the initialization of the sensors, the logic for detecting the water level, controlling the pump, and displaying data on the LCD.

1. INCLUDE LIBRARIES

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <OneWire.h>
#include <DallasTemperature.h>
```

2. DEFINE PIN CONNECTIONS

```
const int TRIG_PIN = 9;     // Ultrasonic Trigger Pin
const int ECHO_PIN = 10;    // Ultrasonic Echo Pin
const int RELAY_PIN = 7;    // Relay Control Pin
const int TEMP_PIN = 2;     // DS18B20 Temperature Sensor Pin
```

3. INITIALIZE OBJECTS

```
// Sensor Setup
OneWire oneWire(TEMP_PIN);
DallasTemperature tempSensor(&oneWire);

// Variables
long duration;
int distance;
bool pumpStatus = false;

// Display timing
unsigned long previousDisplayTime = 0;
const unsigned long waterLevelDisplayTime = 5000;  // 5 seconds
const unsigned long tempDisplayTime = 4000;        // 4 seconds
bool isWaterLevelDisplay = true;
```

1. SETUP FUNCTION

```
void setup() {
  // Initialize Serial Communication (for debugging)
  Serial.begin(9600);

  // Initialize LCD
  lcd.init();
  lcd.backlight();

  // Pin Mode Setup
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);

  // Initialize Temperature Sensor
  tempSensor.begin();

  // Initial Display
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Water Level");
  lcd.setCursor(0,1);
  lcd.print("Monitoring");
  delay(2000);
  lcd.clear();
}
```

5. LOOP FUNCTION (MAIN LOGIC)

```
void loop() {
  // Current time
  unsigned long currentTime = millis();

  // Measure Distance
  distance = measureDistance();

  // Map distance to percentage (27 cm = 0%, 1 cm = 100%)
  int waterLevel = map(distance, 1, 27, 100, 0);

  // Constrain the water level between 0-100%
  waterLevel = constrain(waterLevel, 0, 100);

  // Control Pump Logic
  controlPump(waterLevel);

  // Display Management
  if (isWaterLevelDisplay) {
    // Check if it's time to switch to temperature display
    if (currentTime - previousDisplayTime >= waterLevelDisplayTime) {
      displayTemperature();
      isWaterLevelDisplay = false;
      previousDisplayTime = currentTime;
    } else {
      // Display Water Level
      displayWaterLevel(waterLevel);
    }
  } else {
    // Check if it's time to switch back to water level display
    if (currentTime - previousDisplayTime >= tempDisplayTime) {
      displayWaterLevel(waterLevel);
      isWaterLevelDisplay = true;
      previousDisplayTime = currentTime;
    } else {
      // Display Temperature
      displayTemperature();
    }
  }

  // Small delay to prevent rapid updates
  delay(500);
}

// Function to measure distance
int measureDistance() {
  // Clear the trigger pin
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  // Send ultrasonic pulse
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Measure echo duration
  duration = pulseIn(ECHO_PIN, HIGH);

  // Calculate distance in centimeters
  int calculatedDistance = duration * 0.034 / 2;

  // Debug print
  Serial.print("Distance: ");
  Serial.print(calculatedDistance);
  Serial.println(" cm");

  return calculatedDistance;
}

// Function to display water level and pump status
void displayWaterLevel(int level) {
  // Clear previous display
  lcd.clear();

  // Display Water Level
  lcd.setCursor(0, 0);
  lcd.print("Level:");
  lcd.print(level);
  lcd.print("%   "); // Extra spaces to clear previous characters

  // Display opposite Pump Status
  lcd.setCursor(0, 1);
  lcd.print("Pump:");
  lcd.print(pumpStatus ? "OFF " : "ON "); // Display "OFF" if pump is ON and
"ON" if pump is OFF
}

// Function to display temperature
void displayTemperature() {
  // Request temperature
  tempSensor.requestTemperatures();

  // Read temperature in Celsius
  float tempC = tempSensor.getTempCByIndex(0);

  // Convert to Fahrenheit
  float tempF = tempC * 9.0 / 5.0 + 32.0;

  // Clear previous display
  lcd.clear();

  // Display Temperature
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(tempC, 1);
  lcd.print("C");

  lcd.setCursor(0, 1);
  lcd.print("       ");
  lcd.print(tempF, 1);
  lcd.print("F");
}

// Function to control pump
void controlPump(int waterLevel) {
  // Pump control logic
  if (waterLevel < 30) {
    // Turn pump OFF if water level is below 30%
    digitalWrite(RELAY_PIN, LOW);
    pumpStatus = false;
  }
  else if (waterLevel >= 97) {
    // Turn pump ON if water level is at 95% or above
    digitalWrite(RELAY_PIN, HIGH);
    pumpStatus = true;
  }
}
```