Arduino uno project

```
// Pin connections
const int soilSensorPin = A0;    // Analog pin for soil sensor
const int relayPin = 2;          // Digital pin for relay control

// Threshold values for moisture levels
const int dryThreshold = 700;    // Adjust this value based on your soil sensor readings
const int wetThreshold = 300;    // Adjust this value based on your soil sensor readings

void setup() {
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);   // Initialize relay pin to LOW (off)
  Serial.begin(9600);            // Initialize serial communication
}

void loop() {
  int moistureLevel = analogRead(soilSensorPin);  // Read the moisture level from the soil sensor

  if (moistureLevel >= dryThreshold) {
    // Soil is dry, turn on the pump
    digitalWrite(relayPin, HIGH);
    Serial.println("Pump ON");
  } else if (moistureLevel <= wetThreshold) {
    // Soil is wet enough, turn off the pump
    digitalWrite(relayPin, LOW);
    Serial.println("Pump OFF");
  }

  delay(1000);  // Delay for stability, you can adjust this value if needed
}
```

> Now let's go through the connections:
>
> 1. Connect the VCC (power) and GND (ground) pins of the soil sensor to the 5V and GND pins of the Arduino Uno, respectively.
> 2. Connect the analog output pin of the soil sensor to the A0 pin of the Arduino Uno.
> 3. Connect one terminal of the relay to a digital pin of your choice (in this example, we're using pin 2). Connect the other terminal of the relay to the positive terminal of the DC motor pump.

4. Connect the negative terminal of the DC motor pump to the GND pin of the Arduino Uno.

With these connections and the uploaded sketch, the Arduino will continuously read the moisture level from the soil sensor. If the moisture level goes above the dry threshold, it will activate the relay and turn on the pump, and if it goes below the wet threshold, it will turn off the pump.

Remember to adjust the dryThreshold and wetThreshold values based on the readings you obtain from your soil sensor. Also, ensure that you have a separate power supply for the DC motor pump if it requires more current than the Arduino can provide.

Alternative code

```
// Smart Irrigation System

int sensor_pin = A0;
int output_value;

void setup() {
  pinMode(3, OUTPUT);
  Serial.begin(9600);
  Serial.println("Reading from the Moisture sensor...");
  delay(2000);
}

void loop() {
  output_value = analogRead(sensor_pin);
  output_value = map(output_value, 10, 550, 0, 100);
  Serial.print("Moisture: ");
  Serial.print(output_value);
  Serial.println("%");

  if (output_value < 30) { // Adjust the threshold value according to your needs
    digitalWrite(3, HIGH); // Turn on the pump
    delay(1000);
  } else {
    digitalWrite(3, LOW);  // Turn off the pump
    delay(1000);
  }

  delay(1000);
}
```

Moisture: 10%

10 <30

In case if your using LCD display 16 x 2

To add an LCD display to your Smart Irrigation System, you'll need to make the necessary connections and modify the code to include the LCD library functions. Here's an example code that incorporates a 16x2 LCD display:

```
#include <LiquidCrystal.h>

// Pin connections for LCD
const int rs = 12;
const int en = 11;
const int d4 = 5;
const int d5 = 4;
const int d6 = 3;
const int d7 = 2;

// Pin connections for moisture sensor and pump
const int sensor_pin = A0;
const int pump_pin = 7;

// Threshold value for moisture level
const int moisture_threshold = 30; // Adjust the threshold value according to your needs

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  pinMode(pump_pin, OUTPUT);
  digitalWrite(pump_pin, LOW); // Turn off the pump initially

  lcd.begin(16, 2); // Initialize the LCD display
  lcd.print("Smart Irrigation");
  lcd.setCursor(0, 1);
  lcd.print("System");

  Serial.begin(9600);
  Serial.println("Reading from the Moisture sensor...");
  delay(2000);
}

void loop() {
  int output_value = analogRead(sensor_pin);
  output_value = map(output_value, 10, 550, 0, 100);
  Serial.print("Moisture: ");
  Serial.print(output_value);
  Serial.println("%");

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Moisture: ");
  lcd.print(output_value);
```

```
  lcd.print("%");

  if (output_value < moisture_threshold) {
    digitalWrite(pump_pin, HIGH); // Turn on the pump
    lcd.setCursor(0, 1);
    lcd.print("Pump: ON");
    delay(1000);
  } else {
    digitalWrite(pump_pin, LOW); // Turn off the pump
    lcd.setCursor(0, 1);
    lcd.print("Pump: OFF");
    delay(1000);
  }

  delay(1000);
}
```

In this code, the LiquidCrystal library is included, and the pin connections for the LCD display (rs, en, d4, d5, d6, d7) are defined. Additionally, the pin for the pump is defined as `pump_pin` (pin 7 in this example).

Make sure to connect the appropriate LCD pins to the corresponding Arduino pins. Modify the pin numbers in the code if necessary.

The LCD display is initialized in the `setup()` function, and in the `loop()` function, the moisture value and pump status are displayed on the LCD using the `lcd.print()` and `lcd.setCursor()` functions.

Note: If you're using a different type of LCD display, make sure to refer to the manufacturer's documentation to adjust the pin connections and LCD initialization accordingly.

▼ Take Care

When implementing the code with an LCD display, here are a few things to keep in mind:

1. **Correct wiring**: Ensure that you have properly connected the LCD display to the Arduino. Double-check the pin connections and follow the datasheet or documentation for your specific LCD module.
2. **Adjust contrast**: Some LCD displays may require adjusting the contrast using a potentiometer or trimmer. Make sure to adjust it so that the text is clear and readable on the display.
3. **Power supply**: LCD displays usually require an external power supply in addition to the Arduino's power. Ensure that you have provided the necessary power to the LCD display. Refer to the datasheet or documentation for the voltage and current requirements of your specific LCD module.
4. **Avoid conflicting pin usage**: Check that the pin assignments in the code do not conflict with other components or features you may have connected to your Arduino. Make sure the LCD

pins and other pins used by the soil sensor, pump, or any other components are not conflicting with each other.

5. **Test and debug**: After connecting the LCD and uploading the code, test it thoroughly. Check if the displayed information is correct and if the pump turns on and off based on the moisture level. If you encounter any issues, double-check the wiring, pin connections, and ensure that the code is properly uploaded.

By keeping these considerations in mind, you should be able to successfully implement the code with the LCD display for your smart irrigation system.