**System Requirements:**

1. **Microcontroller:**
   * Use an Arduino controller to manage the system.
2. **User Interface:**
   * **Pushbuttons:**
     + **Button 1 (ON/OFF):** Toggle the system power.
     + **Button 2 (Scroll):** Scroll through the list of vegetables.
     + **Button 3 (Select):** Select a vegetable from the list.
   * **Display:**
     + Use a large LCD display to show:
       - List of vegetables (e.g., Potato, Cauliflower, Spinach, Tomato, Carrot).
       - System messages and alerts.
3. **Vegetable Data Structure:**
   * For each vegetable, store:
     + **Name of Vegetable**
     + **Growth Period (in days)**
     + **Pesticide Spraying Frequency (in days)**
     + **Spare Variable 1**
     + **Spare Variable 2**
     + **Spare Variable 3**
4. **Watering System:**
   * **Soil Moisture Sensor:**
     + Monitor soil moisture levels.
   * **Water Pump Control:**
     + Activate the pump when watering is needed.
     + Deactivate the pump when adequate moisture is detected.
5. **EEPROM Storage:**
   * Record the selected vegetable and start a growth period timer.
6. **Alarms and Notifications:**
   * **Buzzer:**
     + Sound an alarm when it's time to spray insecticides.
     + Sound an alarm when the water drum level is low.
   * **Display Messages:**
     + Notify the user to spray insecticides or refill the water drum.
7. **Water Drum Level Monitoring:**
   * **Water Level Sensor:**
     + Detect when the water drum level is low.
     + Trigger buzzer and display an alert.
8. **Outputs:**
   * **Water Pump Control Output**
   * **Buzzer Control Output**
9. **System Functions:**
   * Daily check of soil moisture and automatic watering.
   * Timely alerts for pesticide spraying based on the vegetable's schedule.
   * System should be able to handle at least 5 different vegetables.

**Component List:**

1. **Arduino Uno or Mega** - 1 unit
2. **16x2 or 20x4 LCD Display** (with I2C interface recommended) - 1 unit
3. **Pushbuttons (Momentary Switches)** - 3 units
4. **Soil Moisture Sensor** - 1 unit
5. **Water Level Sensor for Drum** (e.g., float switch) - 1 unit
6. **Water Pump** (suitable for irrigation) - 1 unit
7. **Relay Module** (to control the water pump) - 1 unit
8. **Buzzer Module** - 1 unit
9. **10kΩ Resistors** (for pull-down or pull-up configuration with buttons) - 3 units
10. **Breadboard and Jumper Wires** - As needed
11. **Power Supply** (e.g., 12V adapter for pump, 5V for Arduino) - As required
12. **EEPROM Library** (built into Arduino IDE) - Software component

**Arduino Code:**

Below is the Arduino code that implements the described system:

cpp

Copy code

#include <LiquidCrystal\_I2C.h>

#include <EEPROM.h>

// Initialize the LCD (adjust the address to 0x27 or 0x3F based on your module)

LiquidCrystal\_I2C lcd(0x27, 16, 2);

// Button pins

const int buttonOnOffPin = 2;

const int buttonScrollPin = 3;

const int buttonSelectPin = 4;

// Sensor pins

const int soilMoisturePin = A0;

const int waterDrumLevelPin = A1;

// Output pins

const int pumpRelayPin = 8;

const int buzzerPin = 9;

// Variables

bool systemOn = false;

int selectedVegetableIndex = 0;

unsigned long startTime = 0;

unsigned long lastPesticideTime = 0;

// Structures

struct Vegetable {

String name;

int growthPeriod; // in days

int pesticideFrequency; // in days

int spare1;

int spare2;

int spare3;

};

Vegetable vegetables[5] = {

{"Potato", 90, 15, 0, 0, 0},

{"Cauliflower", 75, 12, 0, 0, 0},

{"Spinach", 45, 10, 0, 0, 0},

{"Tomato", 80, 14, 0, 0, 0},

{"Carrot", 70, 13, 0, 0, 0}

};

void setup() {

// Initialize serial communication

Serial.begin(9600);

// Initialize LCD

lcd.init();

lcd.backlight();

// Initialize pins

pinMode(buttonOnOffPin, INPUT\_PULLUP);

pinMode(buttonScrollPin, INPUT\_PULLUP);

pinMode(buttonSelectPin, INPUT\_PULLUP);

pinMode(pumpRelayPin, OUTPUT);

pinMode(buzzerPin, OUTPUT);

pinMode(soilMoisturePin, INPUT);

pinMode(waterDrumLevelPin, INPUT);

// Read saved vegetable selection and start time from EEPROM

EEPROM.get(0, selectedVegetableIndex);

EEPROM.get(4, startTime);

lastPesticideTime = startTime;

// Display initial message

lcd.clear();

lcd.print("Press ON to Start");

}

void loop() {

// Read button states

bool onOffButton = !digitalRead(buttonOnOffPin);

bool scrollButton = !digitalRead(buttonScrollPin);

bool selectButton = !digitalRead(buttonSelectPin);

// Handle ON/OFF button

if (onOffButton) {

systemOn = !systemOn;

delay(300); // Debounce delay

if (systemOn) {

lcd.clear();

lcd.print("System ON");

delay(1000);

displayVegetable(vegetables[selectedVegetableIndex].name);

} else {

lcd.clear();

lcd.print("System OFF");

digitalWrite(pumpRelayPin, LOW);

digitalWrite(buzzerPin, LOW);

}

}

if (systemOn) {

// Handle Scroll button

if (scrollButton) {

selectedVegetableIndex = (selectedVegetableIndex + 1) % 5;

displayVegetable(vegetables[selectedVegetableIndex].name);

delay(300); // Debounce delay

}

// Handle Select button

if (selectButton) {

EEPROM.put(0, selectedVegetableIndex);

startTime = millis();

EEPROM.put(4, startTime);

lastPesticideTime = startTime;

lcd.clear();

lcd.print("Selected:");

lcd.setCursor(0, 1);

lcd.print(vegetables[selectedVegetableIndex].name);

delay(2000);

lcd.clear();

displayVegetable(vegetables[selectedVegetableIndex].name);

delay(300); // Debounce delay

}

// Daily check for watering

checkSoilMoisture();

// Check for pesticide spraying time

checkPesticideSchedule();

// Check water drum level

checkWaterDrumLevel();

}

}

void displayVegetable(String name) {

lcd.clear();

lcd.print("Select Veg:");

lcd.setCursor(0, 1);

lcd.print(name);

}

void checkSoilMoisture() {

int moistureValue = analogRead(soilMoisturePin);

// Adjust threshold as per sensor calibration

if (moistureValue < 500) {

// Soil is dry, start pump

digitalWrite(pumpRelayPin, HIGH);

lcd.setCursor(0, 1);

lcd.print("Watering Plants ");

} else {

// Soil is moist, stop pump

digitalWrite(pumpRelayPin, LOW);

lcd.setCursor(0, 1);

lcd.print("Soil Moist OK ");

}

}

void checkPesticideSchedule() {

unsigned long currentTime = millis();

unsigned long elapsedDays = (currentTime - lastPesticideTime) / 86400000UL; // Convert ms to days

if (elapsedDays >= vegetables[selectedVegetableIndex].pesticideFrequency) {

// Time to spray pesticide

lcd.clear();

lcd.print("Spray Pesticide!");

digitalWrite(buzzerPin, HIGH);

delay(5000); // Alert duration

digitalWrite(buzzerPin, LOW);

lastPesticideTime = currentTime;

}

}

void checkWaterDrumLevel() {

int drumLevel = analogRead(waterDrumLevelPin);

// Adjust threshold based on sensor calibration

if (drumLevel < 500) {

// Water level is low

lcd.clear();

lcd.print("Fill Water Drum!");

digitalWrite(buzzerPin, HIGH);

delay(5000); // Alert duration

digitalWrite(buzzerPin, LOW);

// Wait until water drum is refilled

while (analogRead(waterDrumLevelPin) < 500) {

// Keep checking

}

lcd.clear();

displayVegetable(vegetables[selectedVegetableIndex].name);

}

}

**Notes:**

* Adjust the I2C address (0x27 or 0x3F) in the LiquidCrystal\_I2C initialization based on your LCD module.
* The analogRead thresholds (500 in this case) for soil moisture and water drum level sensors need to be calibrated based on your specific sensors.
* The 86400000UL is the number of milliseconds in a day, used to convert milliseconds to days.
* Debounce delays are added to prevent multiple detections from a single button press.
* Ensure that the sensors and pump are compatible with the Arduino's voltage levels, or use appropriate interfacing circuits.

**Block Diagram Description:**

1. **User Interface:**
   * **Pushbuttons:**
     + Connected to digital input pins with pull-up resistors (using INPUT\_PULLUP mode).
   * **LCD Display:**
     + Connected via I2C interface to the Arduino.
2. **Sensors:**
   * **Soil Moisture Sensor:**
     + Analog output connected to an analog input pin on the Arduino.
   * **Water Drum Level Sensor:**
     + Analog or digital output connected to the Arduino (depending on sensor type).
3. **Outputs:**
   * **Water Pump:**
     + Controlled via a relay module connected to a digital output pin.
     + The relay switches the higher voltage required by the pump.
   * **Buzzer:**
     + Connected to a digital output pin, possibly with a driver transistor if needed.
4. **Microcontroller:**
   * **Arduino Board:**
     + Central controller that processes inputs, runs the program logic, and controls outputs.
5. **Power Supply:**
   * **Arduino Power:**
     + Powered via USB or external 5V supply.
   * **Pump Power:**
     + Separate power supply matching the pump's voltage and current requirements.
     + Ground connections common with the Arduino for proper operation.

**Block Diagram:**

scss

Copy code

[ Pushbuttons ] ---> [ Arduino ] <--- [ LCD Display ]

^ ^

| |

[ Soil Moisture Sensor ] [ Water Drum Level Sensor ]

|

[ Relay Module ] <--- [ Water Pump ] --- (Water Source)

|

[ Buzzer ]

**Explanation:**

* **Pushbuttons** allow user interaction for system control and vegetable selection.
* **LCD Display** shows the current vegetable selection, system messages, and alerts.
* **Soil Moisture Sensor** monitors the moisture level in the soil to control watering.
* **Water Drum Level Sensor** checks if there is enough water in the drum for watering.
* **Relay Module** interfaces the Arduino with the water pump, allowing control of higher voltages.
* **Water Pump** irrigates the plants when activated.
* **Buzzer** provides audible alerts for pesticide spraying and low water drum levels.
* **Arduino** processes all inputs and controls outputs based on the programmed logic.

**Additional Tips:**

* **Safety Precautions:**
  + When working with water and electricity, ensure all connections are secure and protected against moisture.
  + Use enclosures for the Arduino and other electronics to prevent water damage.
* **Sensor Calibration:**
  + Test your soil moisture sensor and water level sensor to determine the exact threshold values for your specific setup.
* **Expansion:**
  + The spare variables in the Vegetable structure can be used for additional features like fertilizer schedules, specific watering needs, etc.
* **Code Customization:**
  + Feel free to adjust the code to better suit your specific requirements or to add more features.

**Purchasing Components:**

* Components can be purchased from electronics suppliers like SparkFun, Adafruit, Mouser, or Amazon.
* Ensure compatibility between components, especially voltage levels and communication interfaces.