**PROBLEM DEFINITION:**

**Project Overview:** This project aims to revolutionize water management by implementing IoT sensors to monitor water consumption in public places such as parks and gardens.

**Objectives:**

1. To improve water efficiency by reducing water wastage, optimizing distribution, and promoting responsible water consumption.
2. To develop real-time monitoring systems to ensure proper management of water in public spaces like parks and gardens
3. To collect and analyze data to make informed decisions regarding water usage and promote water conservation by making the data available to the public.

**Design Thinking:**

1. **Project Objectives:**
   1. To develop a system that allows real-time monitoring of water consumption in public places like parks, etc. and to enable data-driven decisions for proper water management and infrastructure maintenance.
   2. Also, to create strategies to educate the public about responsible water use and encourage water-saving behaviors among consumers.
2. **IoT Sensor Design:** 
   1. Sensors such as flow meters or pressure sensors are selected based on the specific monitoring needs in public places.
   2. Relevant data is gathered by sensor installation in key public locations such as parks etc.
   3. Power options such as battery or solar power is selected for uninterrupted IoT sensor working.
3. **Real-time Transit Information Platform:**
   1. To employ user-centered design principles to create an intuitive and visually appealing mobile app interface which provides users with up-to-the-minute information about water consumption, quality and availability in their area.
   2. This includes features that allow users to set water usage goals, receive alerts for anomalies, and access water conservation tips.
4. **Integration Approach:** 
   1. To define data protocols for IoT sensors to transmit data to the data-sharing platform and implement data validation and error-checking mechanisms to ensure the accuracy and integrity of incoming data.
   2. An alert system is created to notify relevant personnel when sensor data indicates anomalies or potential issues and user access to the platform needs to be secured to ensure only authorized people and view and edit the data as needed.

**IMPLEMENTATION:**

Implementing a linear regression model on an Arduino for a smart water system involves collecting data, training the model, and making predictions. A simplified step-by-step guide to demonstrate linear regression on an Arduino is given:

**1. Collect Data:**

Historical water consumption data and relevant features that may affect water usage, such as temperature, time of day, or occupancy are collected.

**2. Data Preprocessing:**

Collected data is cleaned and preprocessed. A suitable format for training the linear regression model on the Arduino is ensured.

**3. Train the Linear Regression Model:**

The linear regression model is trained on a more powerful machine (e.g., a PC) since Arduino's computational resources are limited. Python libraries like sci-kit-learn are used to do this. The model should predict water consumption based on input features.

**4. Export the Model:**

After training the model, it is exported in a format that can be loaded onto the Arduino. Some libraries like TensorFlow Lite for Arduino or Edge Impulse may help with model conversion.

**5. Set Up the Arduino:**

The necessary hardware components, including sensors, collect the input features (e.g., temperature sensors, occupancy sensors), and a display is used to show the predictions.

**6. Load the Model onto the Arduino:**

An appropriate library or framework is used to load the linear regression model onto the Arduino.

**7. Collect Real-time Data:**

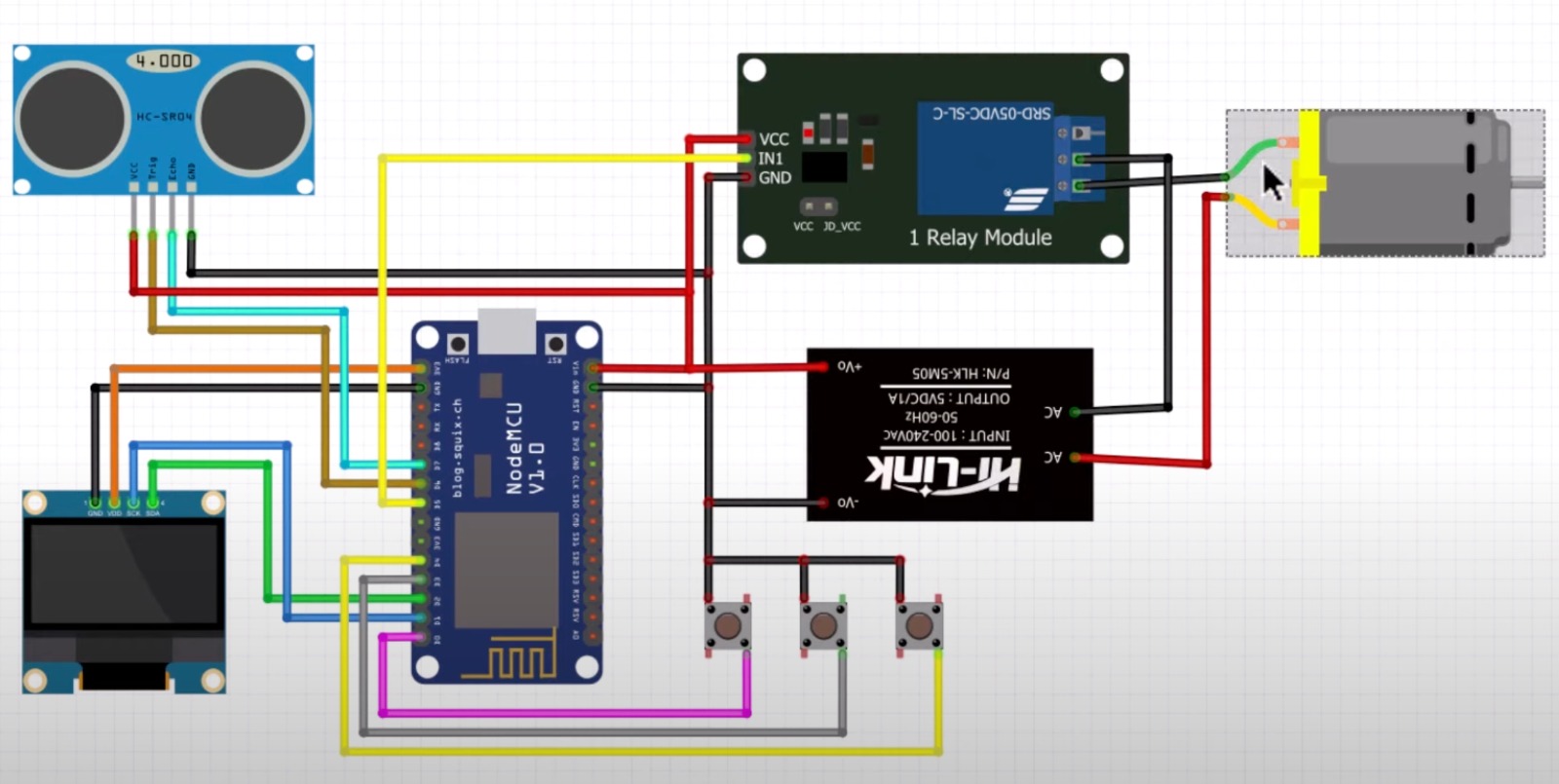
The Arduino is set up to collect real-time data from sensors (e.g., temperature, time of day) that the model needs as input for making predictions.

**8. Make Predictions:**

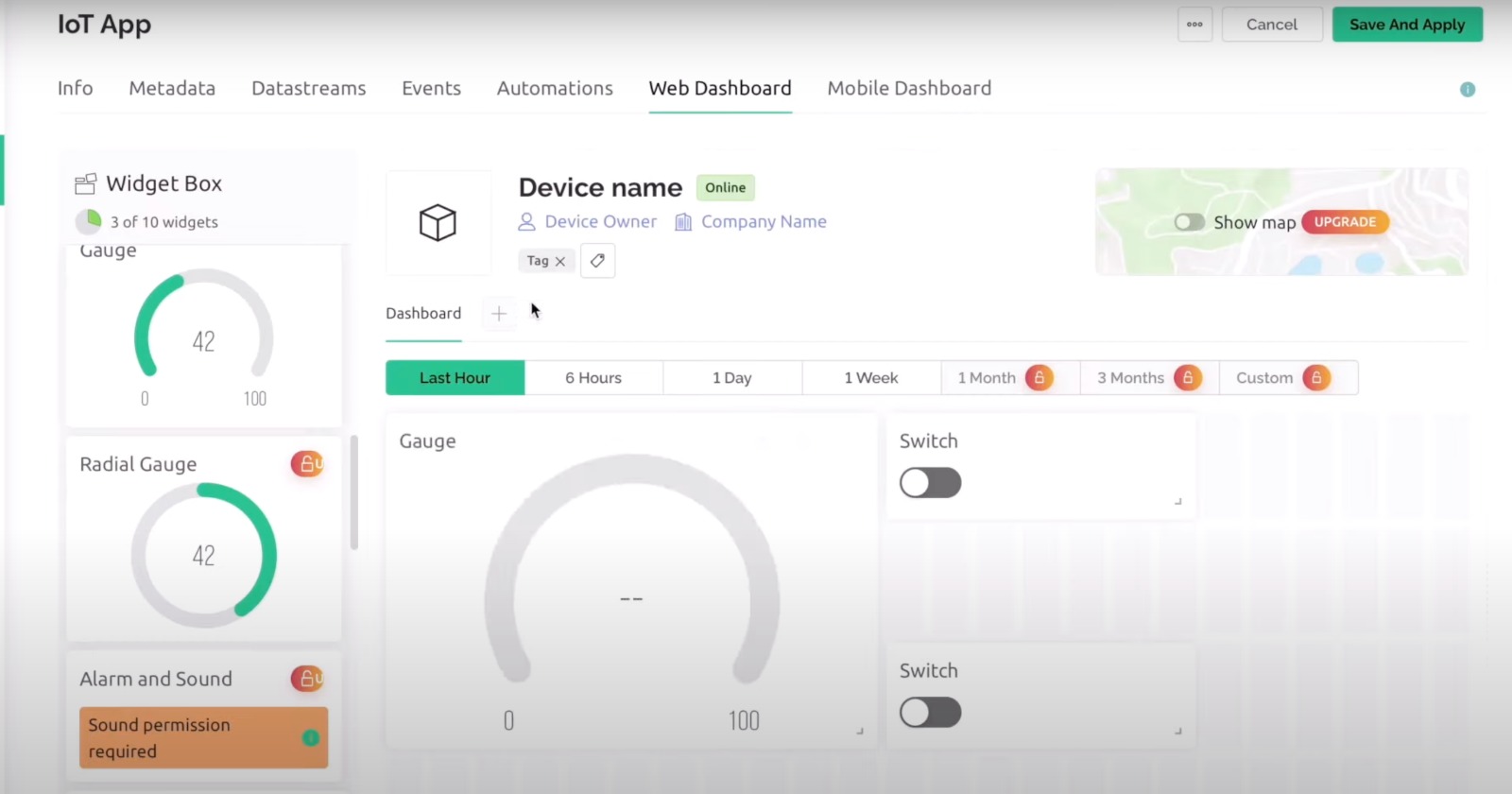
The loaded linear regression model is used to make predictions based on the real-time sensor data. The Arduino will use the model to calculate the expected water consumption.

**9. Display Results:**

The predictions or results are displayed on the Blynk application.



**Figure 1 HARDWARE SETUP OF SMART WATER SYSTEM**



**Figure 2 BLYNK INTERFACE WITH SMART WATER SYSTEM**

**WATER CONSERVATION STRATEGIES**

Implementing water conservation strategies with the help of a smart water system can greatly enhance the efficiency and effectiveness of water resource management. Here are some water conservation strategies using a smart water system:

1. **Real-Time Monitoring:**

Smart meters and sensors can detect leaks, abnormal usage patterns, and inefficiencies immediately, allowing for quick intervention.

1. **Leak Detection and Alerts:**

Automated leak detection systems that can detect even small leaks and send instant alerts to property owners or maintenance personnel, are implemented. This helps prevent water wastage and costly damage.

1. **Water Usage Analytics:**

Collect and analyze historical water consumption data to identify trends and areas where water conservation efforts can be improved. Machine learning algorithms can provide insights and predictions.

1. **Mobile Apps and User Engagement:**

Mobile apps or web interfaces that allow users to monitor their water consumption and receive personalized conservation tips are developed. Gamification and incentives can encourage users to reduce water usage.

1. **Automated Shutoff Systems:**

Automated shutoff valves that can be controlled remotely or set to turn off the water supply during non-peak usage hours or when specific conditions are met (e.g., low occupancy), are installed.

1. **Greywater Recycling:**

Smart systems can manage the collection and treatment of greywater for reuse in non-potable applications like irrigation and toilet flushing.

1. **Water Quality Monitoring:**

Sensors are included to monitor water quality in the smart system. Detecting contaminants early can help prevent pollution and improve overall water quality.

1. **Remote Control and Scheduling:**

It enables users to remotely control water-consuming appliances and allows them to schedule tasks like dishwashing, laundry, and irrigation during off-peak hours.

1. **Automated Appliance Efficiency:**

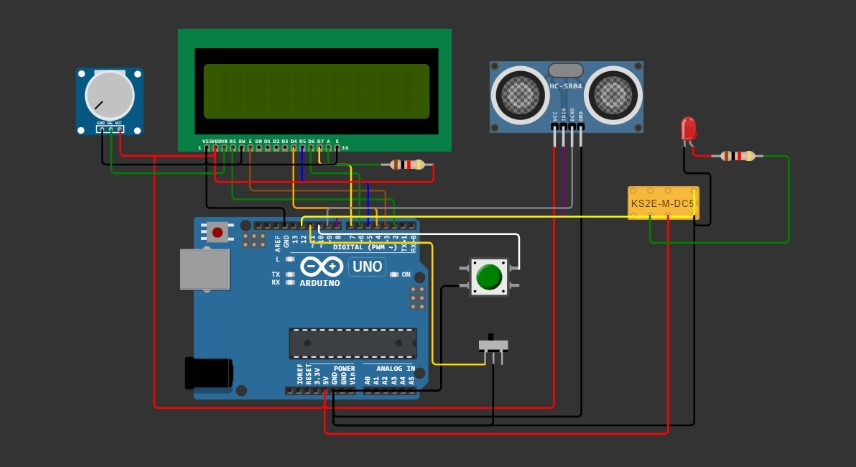
Smart appliances like washing machines and dishwashers are connected to the smart system and their water usage is optimized based on load size and water availability.

1. **Rainwater Harvesting Integration:**

Rainwater harvesting systems are integrated with the smart water system to optimize the collection and use of rainwater for various purposes.

Smart water systems have the potential to transform water conservation efforts by providing data-driven insights, automation, and user engagement. These strategies can help communities and individuals reduce water usage, minimize wastage, and contribute to sustainable water management.

**CIRCUIT SIMULATION**



**CODE**

The code is written for an Arduino platform to control a water level monitoring system.

#include <EEPROM.h>

#include <LiquidCrystal.h>

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

long duration, inches;

int set\_val,percentage;

bool state,pump;

void setup() {

  lcd.begin(16, 2);

  lcd.print("WATER LEVEL:");

  lcd.setCursor(0, 1);

  lcd.print("PUMP:OFF MANUAL");

  pinMode(8, OUTPUT);

  pinMode(9, INPUT);

  pinMode(10, INPUT\_PULLUP);

  pinMode(11, INPUT\_PULLUP);

  pinMode(12, OUTPUT);

   set\_val=**EEPROM**.read(0);

   if(set\_val>150)set\_val=150;

}

void loop() {

   digitalWrite(3, LOW);

   delayMicroseconds(2);

   digitalWrite(8, HIGH);

   delayMicroseconds(10);

   digitalWrite(8, LOW);

   duration = pulseIn(9, HIGH);

   inches = microsecondsToInches(duration);

   percentage=(set\_val-inches)\*100/set\_val;

   lcd.setCursor(12, 0);

   if(percentage<0)percentage=0;

   lcd.print(percentage);

   lcd.print("%   ");

   if(percentage<30&digitalRead(11))pump=1;

   if(percentage>99)pump=0;

   digitalWrite(12,!pump);

   lcd.setCursor(5, 1);

   if(pump==1)lcd.print("ON ");

   else if(pump==0) lcd.print("OFF");

    lcd.setCursor(9, 1);

    if(!digitalRead(11))lcd.print("MANUAL");

    else lcd.print("AUTO   ");

    if(!digitalRead(10)&!state&digitalRead(11)){

      state=1;

      set\_val=inches;

**EEPROM**.write(0, set\_val);

      }

     if(!digitalRead(10)&!state&!digitalRead(11)){

        state=1;

        pump=!pump;

      }

    if(digitalRead(10))state=0;

    delay(500);

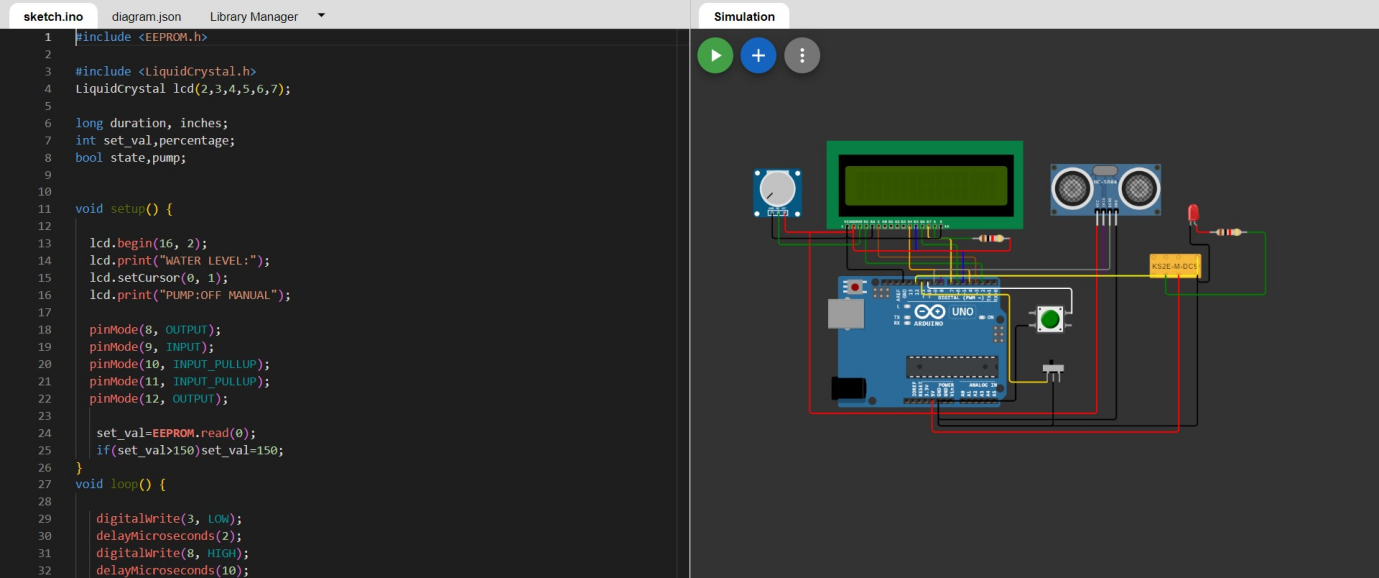
}

long microsecondsToInches(long microseconds) {

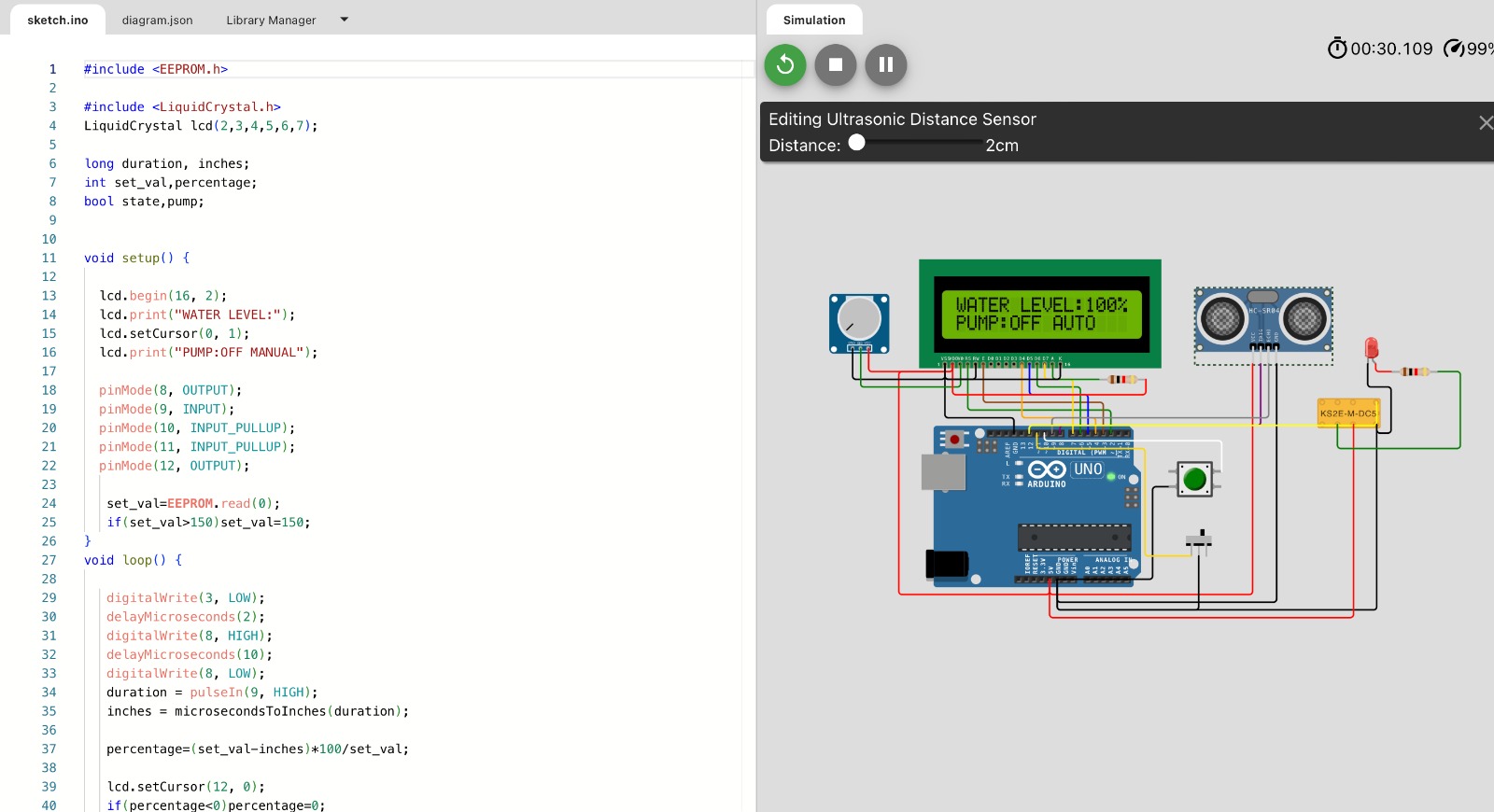
   return microseconds / 74 / 2;

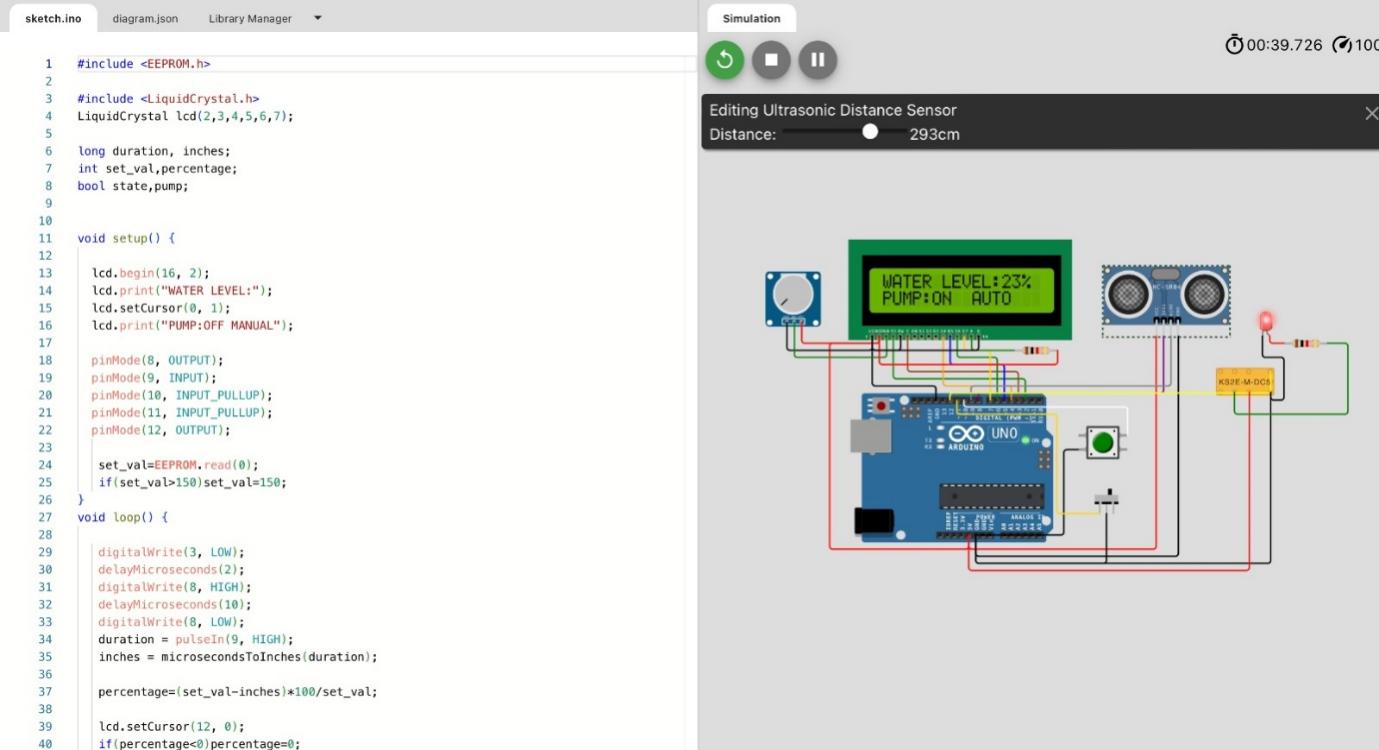
}

**WOKWI SIMULATION INTERFACE**



**SIMULATION OUTPUT**





**CODE EXPLANATION**

**Including Libraries**

**#include <LiquidCrystal.h>**

* The code begins by including the **LiquidCrystal** library, which is used to control a character LCD display.

**Initializing LCD and Variables**

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

   long duration, inches;

   int set\_val, percentage;

   bool state, pump;

* An instance of the **LiquidCrystal** class is created, indicating the pins used to connect to the LCD.
* **duration** and **inches** are used to store the duration of a pulse and the corresponding distance in inches.
* **set\_val** stores a set water level value (a reference point for the desired water level).
* **percentage** is used to calculate the percentage of the current water level compared to the set value.
* **state** and **pump** are Boolean flags for system state and the pump's status.

**Setup Function**

void setup() {

       // ... (initialize LCD and pins)

   }

* In the **setup()** function, the code initializes the LCD, sets up input and output pins and reads a stored value from EEPROM, which is the reference water level.

**Loop Function**

 void loop() {

       // ... water level measurement, calculations, and control

   }

* The main logic of the code is in the **loop()** function.

**Water Level Measurement**

* It triggers an ultrasonic distance sensor connected to **pin 9** to measure the distance to the water surface.
* The duration of the pulse is measured and converted to inches using the **microsecondsToInches** function.

**Calculating Water Level Percentage**

* The code calculates the water level percentage using the difference between the set value and the actual measurement.

**Displaying Water Level and Pump Status on LCD**

* The calculated percentage is displayed on the LCD.
* The code determines whether the pump should be on based on the water level and whether it's in manual or automatic mode.
* The pump's status is displayed on the LCD.

**Controlling the Pump**

* The pump is controlled based on the water level. If the water level is below 30% and in automatic mode (as determined by pin 11), the pump is turned on.
* If the water level is above 99%, the pump is turned off.

**User Interaction**

* The code checks for user interaction via a button connected to pin 10. If the button is pressed and released, it stores the current water level as the set value in EEPROM.
* If the button is pressed and released in manual mode (determined by pin 11), it toggles the pump status.

**Delay**

* There is a delay of 500 milliseconds before the loop repeats.

**Summary**

This code controls a water level monitoring and pump control system. It uses an ultrasonic sensor to measure the water level, displays the water level percentage and pump status on an LCD, and allows user interaction to set the desired water level and control the pump manually. The pump operates automatically to maintain the water level within a specified range.