Department of Computer Science and Engineering

CSE 406: Internet of Things

Spring 2025 Semester

Lab 01 Assignment: Real-time Data Acquisition and Visualization with a Water Level Sensor

Dr. Raihan Ul Islam

September 29, 2025

Objective

This assignment is designed to provide hands-on experience with analog sensors and real-time data visualization. You will interface a water level sensor with an Arduino Uno, read its analog output, process the data to derive meaningful insights, and visualize the results using the Arduino IDE's Serial Plotter. This lab directly contributes to Course Outcome 4 (CO4), focusing on designing and implementing end-to-end IoT solutions.

Hardware & Software Requirements

- Arduino Uno Board
- Water Level Measurement Sensor
- Breadboard & Jumper Wires
- A container with water
- Arduino IDE

---

1 Part 1: Sensor Interfacing and Initial Reading

The water level sensor acts as a variable resistor. As more of the sensor is submerged in water, its conductivity increases, which we can measure as a change in voltage. The Arduino's analog pins can read this voltage and convert it into a numerical value between 0 and 1023.

1.1 Wiring Instructions

Connect the sensor to the Arduino Uno as follows:

- **S (Signal) Pin** on Sensor → **A0 Pin** on Arduino
- **+ (VCC) Pin** on Sensor → **5V Pin** on Arduino
- **- (GND) Pin** on Sensor → **GND Pin** on Arduino

1.2 Basic Reading Code

Upload the following code to your Arduino. This will read the raw analog value from the sensor and print it to the Serial Monitor.

```
/*
  CSE406 - Lab Assignment 02: Basic Water Level Reading
  Reads the raw analog value from the water level sensor on pin A0.
*/

const int sensorPin = A0; // Sensor signal pin connected to Analog pin A0

void setup() {
  // Initialize serial communication at 9600 bits per second
  Serial.begin(9600);
}

void loop() {
  // Read the analog value from the sensor
  int sensorValue = analogRead(sensorPin);

  // Print the value to the Serial Monitor
  Serial.print("Raw Sensor Value: ");
  Serial.println(sensorValue);

  // Wait for 100 milliseconds before the next reading
  delay(100);
}
```

1.3 Using the Serial Plotter

To visualize the data, use the Arduino's built-in Serial Plotter.

1. After uploading the code, go to `Tools -> Serial Plotter`.
2. Slowly dip the sensor into a cup of water and pull it out. You should see a live graph of the raw sensor value changing.
3. **Note:** The Serial Plotter works best when you only print the numerical data. Modify the `loop()` function to print just the value: `Serial.println(sensorValue);`

2 Part 2: Programming and Data Processing Tasks

Modify the basic code to implement the following three features in a single sketch.

Task 1: Data Mapping and Visualization

The raw sensor value (0-1023) is not intuitive. Map this value to a percentage scale (0-100%).

- Use the `map()` function to convert the `sensorValue` to a percentage. You may need to determine the minimum (dry) and maximum (fully submerged) raw values for your sensor to calibrate the map function effectively.
- Modify your code to print both the raw value and the calculated percentage, separated by a space or comma.
- Use the Serial Plotter to visualize both data streams simultaneously.

Task 2: Water Level Alarms

Implement a system that triggers alarms at different water levels.

- Define three thresholds: `LOW_LEVEL` (e.g., 25%), `MEDIUM_LEVEL` (e.g., 75%), and `HIGH_LEVEL` (e.g., 95%).
- Use `if-else if-else` statements to check the current water level percentage.
- Print a specific alert message to the Serial Monitor (not the plotter) when the water level crosses these thresholds.

Task 3: Calculate and Plot Rate of Change

Determine how quickly the water level is rising or falling.

- In your `loop()`, store the previous water level reading in a variable.
- Calculate the difference between the current reading and the previous reading. This is the "rate of change."
- Print the current water level percentage and the calculated rate of change to the Serial Plotter.

3 Submission Guidelines

1. **Code Comments:** Your final `.ino` file must be thoroughly commented. Explain the purpose of variables, key functions, and the logic behind each of the three tasks.
2. **GitHub Repository:** Create a new public repository on GitHub named `CSE406-Lab02`. Commit and push your final, commented `.ino` file to this repository.
3. **Submission:** Submit the URL of your public GitHub repository.

4 Grading Rubric (Total: 10 Marks)

Criteria	**Unsatisfactory (0-0.5)**	**Developing (1)**	**Excellent (2)**	**Marks**
Circuit & Basic Functionality	Circuit is incorrect or code does not read any sensor data.	Code reads sensor data but it is unstable or incorrect.	The sensor is correctly wired and the code successfully reads and plots the raw analog data.	/ 2
Task 1: Data Mapping	The `map()` function is not implemented or works incorrectly.	Data is mapped, but the scale is incorrect or only one value is plotted.	The raw sensor value is correctly mapped to a 0-100% scale, and both raw and mapped values are plotted.	/ 2
Task 2: Threshold Alarms	No alarm logic is implemented.	Alarm logic is partially implemented or contains significant errors.	`if-else` statements correctly identify and print alerts for at least three distinct water levels.	/ 2
Task 3: Rate of Change	Rate of change is not calculated.	The calculation is attempted but the logic is flawed (e.g., doesn't account for time).	The rate of change is correctly calculated and plotted, accurately reflecting the speed of water level changes.	/ 2
Code Quality & Submission	No comments and no GitHub submission.	Code has minimal comments OR is submitted without using GitHub.	Code is well-commented, easy to understand, and submitted correctly to a public GitHub repository.	/ 2
			Total	**/ 10**