Cairo University
Faculty of Engineering

Credit Hours system
Mechanical Department

Course Name (MDPN470 – Mechatronics Lab)

Water Level Control System Lab

Name: Mohamed Abd El Twab Newir

ID: 4210215

Group: 3

Submitted to:

Dr. Mostafa Gamal

Dr. Moataz El Sisi

Submission date: 5- 12- 2024

Table of Contents

Table of Figures

1. Introduction

Water level control systems are essential in a wide range of industrial and domestic applications, such as irrigation, water tanks, and wastewater management. This lab project aimed to design and implement a simple water level control system using Arduino, sensors, and actuators to automatically manage water levels.

2. Objectives

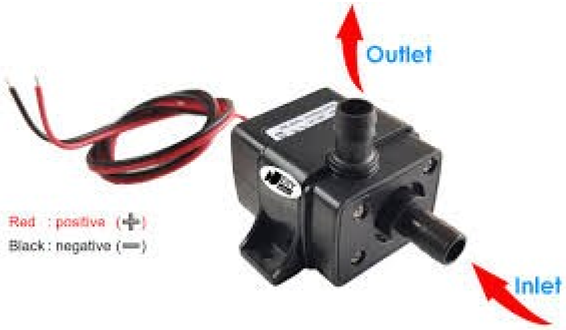
The objectives of this project were:

- Design a water level control system using readily available components.
- Implement and test the functionality of water level detection.
- Control water flow via a pump and valve based on sensor readings.
- Troubleshoot and optimize the system's performance.

3. Components Used

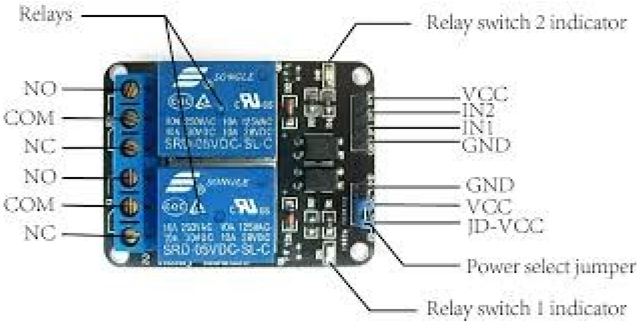
- Arduino Board: Acts as the control unit for processing sensor input and controlling actuators.
- XKC Y25 T12V Sensors: Used for water level detection.

- Water Pump: Transfers water to maintain the desired level.



- Solenoid Valve: Regulates the flow of water when needed.

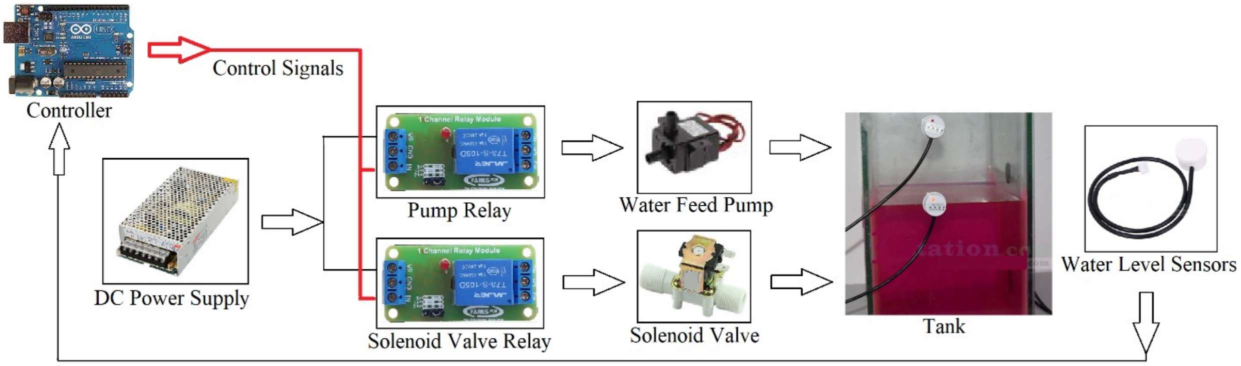
- Relay Module: Provides electrical isolation and control for the pump and the solenoid valve.



- 12V Power Supply: Powers the pump and sensors.

- Connecting Wires and Breadboard: For circuit connections.

4. Circuit Diagram



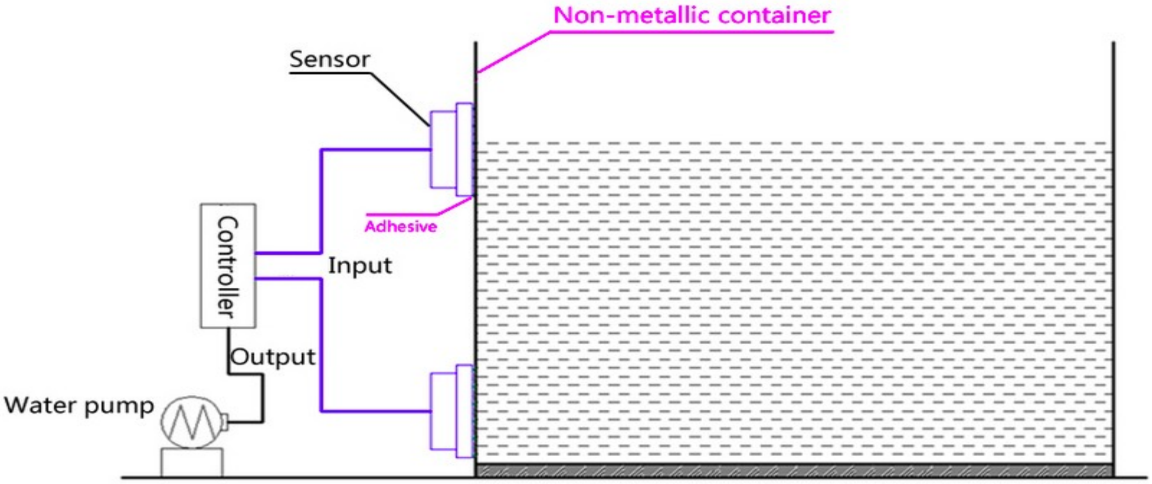
5. Analogy of Work

1. System Setup:

- The XKC Y25 T12V sensors are positioned at predefined high and low water levels.
- The water pump fills the tank, and the valve regulates water discharge.
- The relay module controls both the pump and valve based on signals from the Arduino.

2. Working Logic:

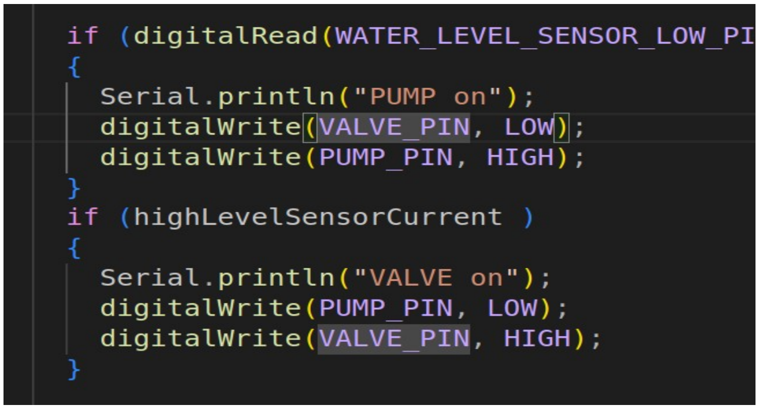
- If the water level drops below the low sensor, the pump turns ON to refill the tank.
- If the water level reaches the high sensor, the pump turns OFF, and the valve can open if necessary to release excess water.



6. Trials and Challenges

- Initial Code Attempts:

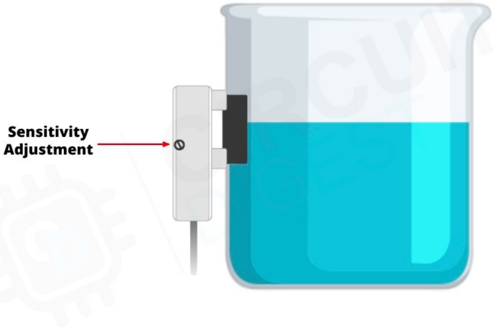
  - Tried multiple code combinations to interface with the sensors.
  - Struggled with inconsistent readings from the sensors.



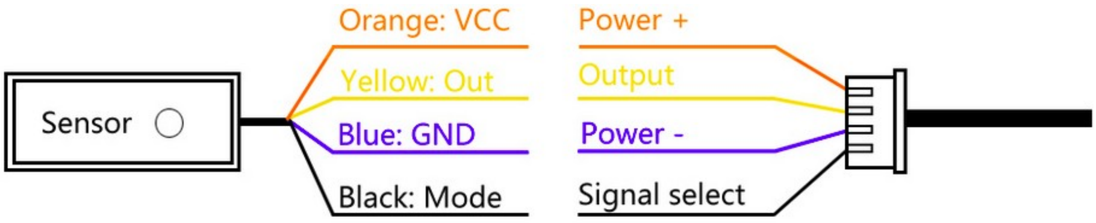
```
if (digitalRead(WATER_LEVEL_SENSOR_LOW_PI
{
  Serial.println("PUMP on");
  digitalWrite(VALVE_PIN, LOW);
  digitalWrite(PUMP_PIN, HIGH);
}
if (highLevelSensorCurrent )
{
  Serial.println("VALVE on");
  digitalWrite(PUMP_PIN, LOW);
  digitalWrite(VALVE_PIN, HIGH);
}
```

- Sensor Sensitivity Adjustments:

  - Adjusted sensor placement and parameters to improve detection accuracy.



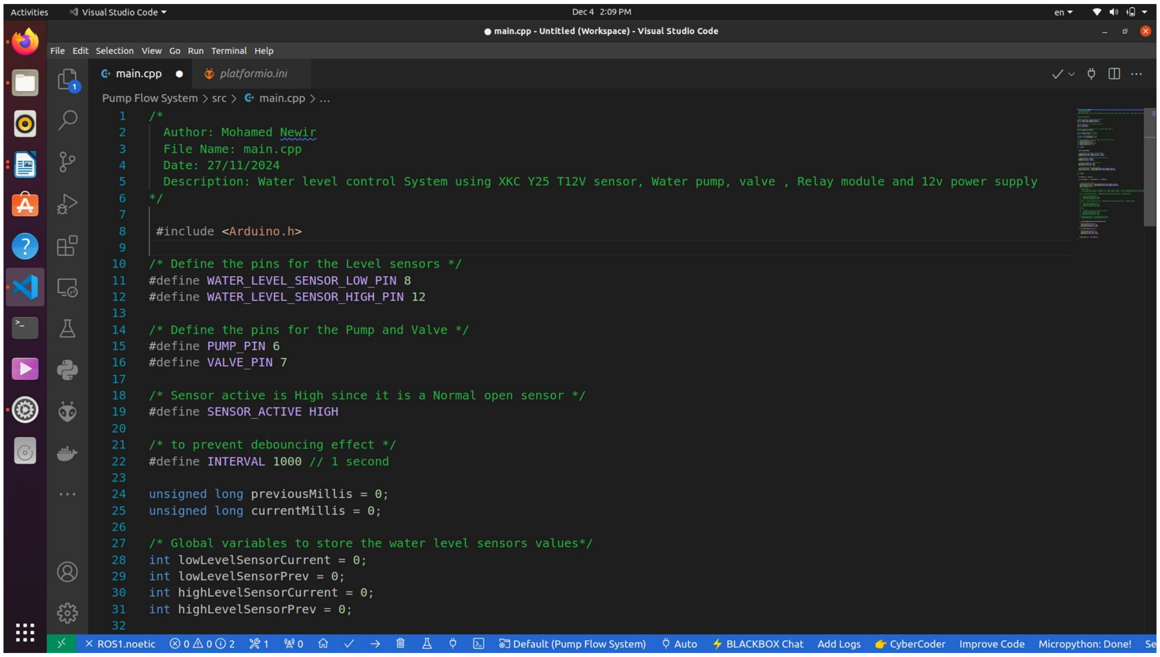
- Trying both Normal open and Normal close terminology



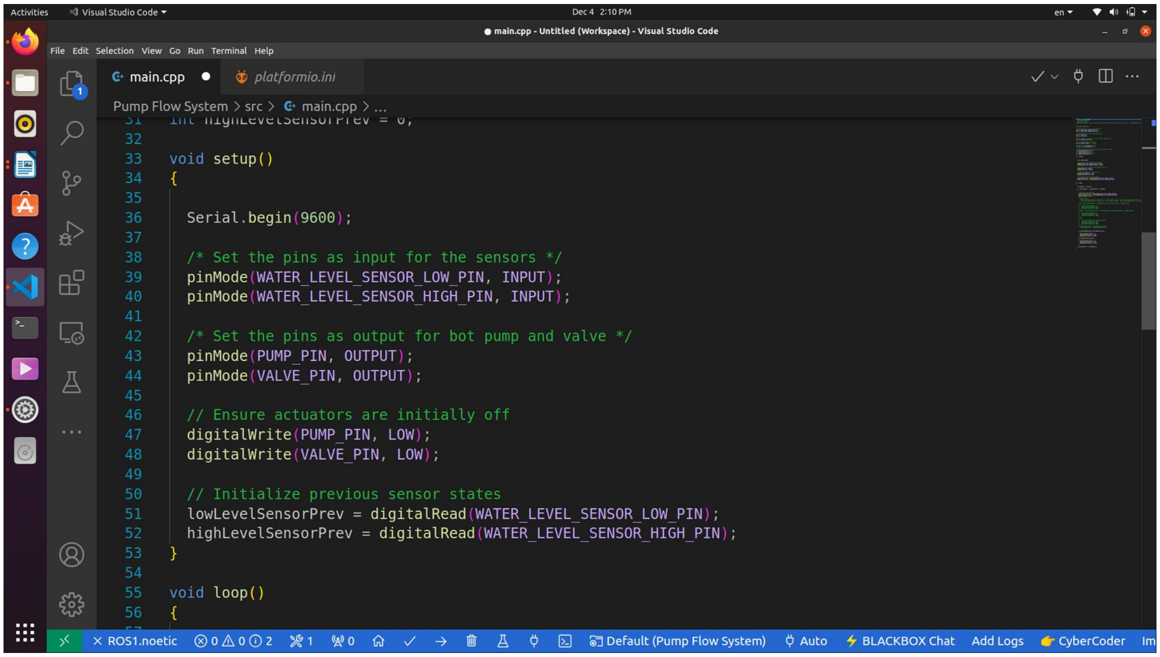
- Sensor Malfunction:

    - Found that the sensors output a HIGH signal regardless of the presence of water.
    - Concluded that the sensors were either defective or incompatible with the setup.

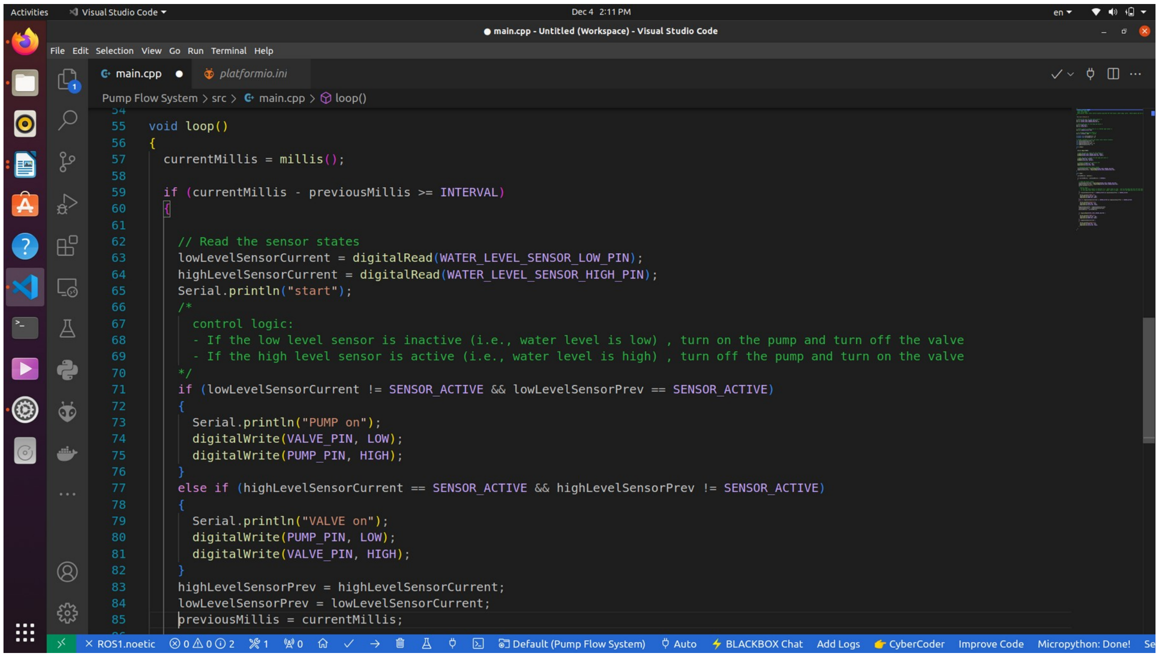
7. Code Section



```cpp
/*
  Author: Mohamed Newir
  File Name: main.cpp
  Date: 27/11/2024
  Description: Water level control System using XKC Y25 T12V sensor, Water pump, valve , Relay module and 12v power supply
*/

 #include <Arduino.h>

/* Define the pins for the Level sensors */
#define WATER_LEVEL_SENSOR_LOW_PIN 8
#define WATER_LEVEL_SENSOR_HIGH_PIN 12

/* Define the pins for the Pump and Valve */
#define PUMP_PIN 6
#define VALVE_PIN 7

/* Sensor active is High since it is a Normal open sensor */
#define SENSOR_ACTIVE HIGH

/* to prevent debouncing effect */
#define INTERVAL 1000 // 1 second

unsigned long previousMillis = 0;
unsigned long currentMillis = 0;

/* Global variables to store the water level sensors values*/
int lowLevelSensorCurrent = 0;
int lowLevelSensorPrev = 0;
int highLevelSensorCurrent = 0;
int highLevelSensorPrev = 0;

```



```cpp
int highLevelSensorPrev = 0;

void setup()
{

  Serial.begin(9600);

  /* Set the pins as input for the sensors */
  pinMode(WATER_LEVEL_SENSOR_LOW_PIN, INPUT);
  pinMode(WATER_LEVEL_SENSOR_HIGH_PIN, INPUT);

  /* Set the pins as output for bot pump and valve */
  pinMode(PUMP_PIN, OUTPUT);
  pinMode(VALVE_PIN, OUTPUT);

  // Ensure actuators are initially off
  digitalWrite(PUMP_PIN, LOW);
  digitalWrite(VALVE_PIN, LOW);

  // Initialize previous sensor states
  lowLevelSensorPrev = digitalRead(WATER_LEVEL_SENSOR_LOW_PIN);
  highLevelSensorPrev = digitalRead(WATER_LEVEL_SENSOR_HIGH_PIN);
}

void loop()
{
```


Activities
Visual Studio Code
Dec 4 2:11 PM
main.cpp - Untitled (Workspace) - Visual Studio Code
File Edit Selection View Go Run Terminal Help
main.cpp
platformio.ini
Pump Flow System > src > main.cpp > loop()
ROS1.noetic
Default (Pump Flow System)
Auto
BLACKBOX Chat
Add Logs
CyberCoder
Improve Code
Micropython: Done!

```cpp
void loop()
{
  currentMillis = millis();

  if (currentMillis - previousMillis >= INTERVAL)
  {

    // Read the sensor states
    lowLevelSensorCurrent = digitalRead(WATER_LEVEL_SENSOR_LOW_PIN);
    highLevelSensorCurrent = digitalRead(WATER_LEVEL_SENSOR_HIGH_PIN);
    Serial.println("start");
    /*
      control logic:
      - If the low level sensor is inactive (i.e., water level is low) , turn on the pump and turn off the valve
      - If the high level sensor is active (i.e., water level is high) , turn off the pump and turn on the valve
    */
    if (lowLevelSensorCurrent != SENSOR_ACTIVE && lowLevelSensorPrev == SENSOR_ACTIVE)
    {
      Serial.println("PUMP on");
      digitalWrite(VALVE_PIN, LOW);
      digitalWrite(PUMP_PIN, HIGH);
    }
    else if (highLevelSensorCurrent == SENSOR_ACTIVE && highLevelSensorPrev != SENSOR_ACTIVE)
    {
      Serial.println("VALVE on");
      digitalWrite(PUMP_PIN, LOW);
      digitalWrite(VALVE_PIN, HIGH);
    }
    highLevelSensorPrev = highLevelSensorCurrent;
    lowLevelSensorPrev = lowLevelSensorCurrent;
    previousMillis = currentMillis;
```

8. Discussion

- Sensor Reliability:
  - The XKC Y25 T12V sensors displayed unreliable performance in detecting water levels.
  - Alternatives such as float switches or ultrasonic sensors may provide more consistent results.

- Code and Control Logic:
  - The control logic effectively toggles the pump and valve.
  - Debugging tools like serial output could further enhance troubleshooting.

9. Conclusion

The lab project successfully implemented a basic water level control system. However, hardware limitations, particularly sensor malfunctions, hindered achieving optimal functionality. The project highlights the importance of sensor selection and troubleshooting skills in system design.

10. Future Recommendations

- Use higher-quality or alternative water level sensors for better accuracy.
- Incorporate a user interface to monitor water levels and control operations manually.
- Test the system in real-world scenarios to validate performance.