# REAL-TIME DAM SAFETY MONITORING WITH SENSOR-BASED DISPLAY AND ALERT SYSTEM

**A PROJECT REPORT**

**21ES602 – Embedded System Design Using ARM**

***submitted by***

**Anish Sandaye CB.EN.P2EBS24003**
**Katta Sreenivas CB.EN.P2EBS24023**

***in partial fulfilment for the award of the degree***

***of***

**MASTER OF TECHNOLOGY**

**IN**

**EMBEDDED SYSTEMS**


**AMRITA SCHOOL OF ENGINEERING, COIMBATORE**

**AMRITA VISHWA VIDYAPEETHAM**

**COIMBATORE – 641 112**

**Submitted on - November, 2024**

## DECLARATION

I Anish Sandaye (CB.EN.P2EBS24003) & Kata Sreenivas (CB.EN.P2EBS24023) hereby declare that this project report entitled **Real-Time Dam Safety Monitoring with Sensor-Based Display and Alert System** is a record of the original work done by me under the guidance of **Shanmugha Sundaram R.** Assistant Professor, Department of Electrical and Electronics Engineering, Amrita School of Engineering, Coimbatore and this work has not formed the basis for the award of any degree / diploma / associateship / fellowship or a similar award, to any candidate in any University, to the best of my knowledge.

**Place: Coimbatore**

**Date:**

**Signature of the Student**

**Shanmugha Sundaram R.**

**Assistant Professor**

**Department of Electrical and Electronics Engineering**

**Amrita School of Engineering, Coimbatore – 641112**

## ABSTRACT

This project presents an embedded system for monitoring dam water levels and structural force using an LPC214x microcontroller. The system integrates an ultrasonic sensor for real-time water level measurement and a force sensor connected through an ADC channel to detect applied force. Key thresholds for water level and force are defined, and when these limits are exceeded, a buzzer and LED are activated to alert operators to potential hazards. The system also displays current sensor readings on an LCD, providing operators with immediate, on-site data. This automated monitoring and alert system enhances safety by ensuring timely response to changes in dam conditions. The project's modular design allows for easy adaptation to other monitoring and control environments requiring real-time data processing and alerting.

## CONTENTS

## Chapter 1

# INTRODUCTION

This embedded system project uses an LPC214x microcontroller to monitor force and water levels in a dam environment. In order to determine the current water level in the dam, the system uses an ultrasonic sensor to measure the distance between the water's surface and a predetermined height. It also has a force sensor to measure applied force that is connected via an ADC channel. The system alerts users with an internal LED and a buzzer if the force or water level readings surpass predetermined safety thresholds.

An LCD attached to the microcontroller shows force and water level data in real time. The application has features to set up the LCD in 4-bit mode for effective updates, control the ultrasonic sensor's echo and trigger for accurate distance measurement, and initialize the ADC for sensor data acquisition. In order to achieve the microsecond accuracy required for ultrasonic measurements, a timer setup is also utilized.

The system is divided into distinct functions for ultrasonic distance measurement, LCD control, threshold-based alerting, and ADC setup by the modular design. This methodical approach facilitates easy modification and improves readability, making it suitable for a range of applications where real-time response to shifting environmental conditions—like dam monitoring is crucial.

## Chapter 2

# LITERATURE SURVEY

**Paper 1:**
**Title:** IoT Based Water Level Monitoring System for Dams.
**Author:** V.M.V.S Aditya, Ch.T.S. Tanishq, V. Chaitanya Badri Sai, Sateesh krishna Dhuli
**Methodology:**
The literature survey in "IoT Based Water Level Monitoring System for Dams" examines IoT applications in dam safety, flood prevention, and water resource management. Advanced approaches integrated machine learning and neural networks for flood prediction based on rainfall data, though with accuracy challenges, and blockchain-based IoT solutions were also applied to ensure data integrity and secure dam surveillance. Research on pre-alarm systems for tailings dams further highlighted the use of IoT for structural safety. While these solutions address monitoring and alerts, there remains a gap in establishing a complete routing system between dams and canals. The paper addresses this by proposing an IoT-based system that routes water efficiently, aiming to improve water distribution and flood management across connected reservoirs.

**Paper 2:**
**Title:** Dam Management and Disaster Monitoring System using IoT.
**Author:** D. Dhinakaran, S. M. Udhaya Sankar
**Methodology:**
The system utilizes an Arduino Uno microcontroller to distribute power to connected sensors, which monitor environmental conditions and send analog data to the Arduino. This data is converted into a digital signal and displayed on an LED screen. In disaster scenarios, pre-determined alarm triggers alert management teams to potential hazards, such as temperature, pressure, or seismic thresholds. Sensors continuously monitor key parameters like water levels and stress, transmitting real-time data to a cloud server for analysis and decision-making. Reliability is ensured through redundancy, fault tolerance, testing, and machine learning algorithms that detect anomalies, improving early warning and disaster preparedness.

In recent years, there has been a growing interest in using IoT technology for dam management and disaster monitoring. These systems aim to offer real-time monitoring and early disaster warnings, helping to prevent catastrophic events. Additionally, the integration of data analytics and machine learning algorithms holds promise for identifying patterns and anomalies, enhancing the accuracy of early warning systems.

## Chapter 3

# COMPONENT SPECIFICATION

### 3.1 Components Required

i. LPC 2148 Microcontroller
ii. Ultrasonic Sensor (HC-SR04)
iii. Force Sensor (FSR)
iv. 16x2 LCD Display
v. Buzzer
vi. LED
vii. DC Motor

### 3.2 Specifications

**LPC 2148 Microcontroller:**
**Architecture**: ARM7TDMI-S, 32-bit RISC architecture
**Clock Speed**: Up to 60 MHz
**Flash Memory**: 512 KB
**RAM**: 40 KB
**I/O Pins**: 45 (GPIO)
**Peripherals**: 2 UART, 2 I2C, 2 SPI, ADC, DAC, PWM, Timers, USB 2.0 full-speed support
**Operating Voltage**: 3.3V

**Ultrasonic Sensor (HC-SR04):**
**Operating Voltage:** 5V DC
**Measuring Distance:** 2 cm to 400 cm
**Accuracy**: ±3 mm
**Frequency:** 40 kHz ultrasonic pulse
**Operating Current:** 15 mA
**Working Principle:** Uses ultrasonic waves to measure the distance between the sensor and an object.

**Force Sensor (FSR):**
**Type**: Force-sensitive resistor (FSR)
**Operating Voltage**: 0-5V
**Force Range**: 0.2 N to 20 N (can vary based on model)
**Size**: Typically, 0.5 inch in diameter (FSR 402)
**Sensitivity**: Resistance decreases as the force increases.

**16x2 LCD Display:**
**Display Size:** 16 characters x 2 lines
**Operating Voltage:** 4.7V - 5.3V
**Current:** 1mA (with backlight off), up to 60mA (with backlight on)
**Interface:** Parallel, 4-bit or 8-bit data interface
**Character Size:** 5x8 dot matrix per character
**Backlight:** LED backlight

**Buzzer:**
**Operating Voltage**: 3V - 12V
**Current Consumption**: 10 mA - 30 mA (depending on voltage)
**Sound Frequency**: 1.5 kHz to 3 kHz (typical)
**Type**: Active (self-driving) or passive (requires external signal)
**Sound Level**: Typically, 85 dB at 12V

**LED:**
**Operating Voltage**: Typically, 2V to 3.5V (varies by colour)
**Current**: 10mA - 20mA
**Brightness**: Typically, 1000-5000 mcd (depends on the type and colour)
**Colours**: Available in multiple colours (Red, Green, Blue, etc.)
**Lifespan**: Up to 50,000 hours

**DC Motor:**
**Operating Voltage**: Typically, 6V to 12V (can vary)
**Current**: 100 mA to 1 A (depending on motor size)
**Speed**: 1000 RPM to 6000 RPM (varies by model)
**Torque**: Typically, 0.5 N·m to 1.5 N·m (depends on motor size)
**Type**: Brushed or brushless DC motor
**Applications**: Widely used for controlling physical movements like gate control, wheels, etc.

Chapter 4

# WORK DONE

## 4.1 Methodology:

### 1. Objective and Setup

The primary goal of this project is to monitor the water level in a dam and measure the force applied to a force sensor, triggering alerts if certain thresholds are exceeded. The system is built using an LPC214x microcontroller, interfaced with an ultrasonic sensor, force sensor, LCD display, buzzer, and internal LED.

### 2. Measuring Water Level with an Ultrasonic Sensor

The ultrasonic sensor is positioned to measure the distance from its mounting point to the water surface. Key steps in this process include:
Triggering the Sensor: The system sends a 10µs pulse to the sensor's TRIG_PIN, initiating a distance measurement.
Reading Echo Response: The ECHO_PIN receives the return signal from the ultrasonic sensor. The time between sending the trigger and receiving the echo is used to calculate the distance.
Calculating Water Level: The system calculates water level by subtracting the measured distance from the predefined dam height (TANK_HEIGHT_CM). This calculated water level is then displayed on the LCD.

### 3. Measuring Force with an ADC-Connected Force Sensor

The force sensor is connected to an ADC channel on the microcontroller:
Analog-to-Digital Conversion: The ADC reads the analog signal from the force sensor and converts it to a digital value. This value represents the force applied on the sensor.
Threshold Comparison: If the force reading exceeds the defined FORCE_THRESHOLD (20), it will trigger an alert by activating the buzzer and internal LED. This provides an immediate response to critical force levels on the dam structure.

### 4. Displaying Sensor Values on LCD

The LCD is used to display real-time sensor readings for water level and force:
LCD Initialization: The system initializes the LCD in 4-bit mode, enabling efficient data transmission.
Updating Readings: Each reading cycle, the system updates the LCD with the current water level and force values, providing clear, real-time monitoring for dam operators.

### 5. Alert System with Buzzer and LED

The project includes an alert mechanism to notify users when safety thresholds are breached:
Activating Alerts: If either the water level exceeds WATER_LEVEL_THRESHOLD (25 cm) or the force reading exceeds the set threshold, the buzzer and internal LED are activated.
Visual and Audible Alerts: The internal LED flashes, and the buzzer sounds, making it clear that action is needed.
Resetting Alerts: If the readings fall below the thresholds, the buzzer and LED are turned off.

### 6. Adding Delays for Stability

Small delays are strategically placed throughout the code:
Stabilizing Signals: Delays help stabilize signals, especially for components like the ultrasonic sensor and LCD, ensuring that each component has enough time to process data before moving on to the next reading or command.
User Readability: The LED's blinking pattern includes delays to make it more noticeable, enhancing the alerting function.

**7. Timer Setup for Ultrasonic Sensor Timing**
To ensure accurate measurement of the echo timing from the ultrasonic sensor, a timer is configured:
Microsecond Resolution: The timer is set to 1µs resolution, which allows precise calculation of the travel time of the ultrasonic pulse.
Distance Calculation: The travel time measured by the timer is used to compute the distance to the water, which directly informs the water level calculation.

**8. Continuous Monitoring Loop**
The system operates in a continuous loop:
Regular Sensor Readings: In each iteration of the loop, the system reads both water level and force data.
Real-Time Threshold Checking: Thresholds are continuously checked, ensuring immediate response if any readings exceed safe limits.
Display and Alert Update: The LCD display and alert system are updated every loop iteration, providing real-time monitoring and alerts.

### 4.2 Block Diagram

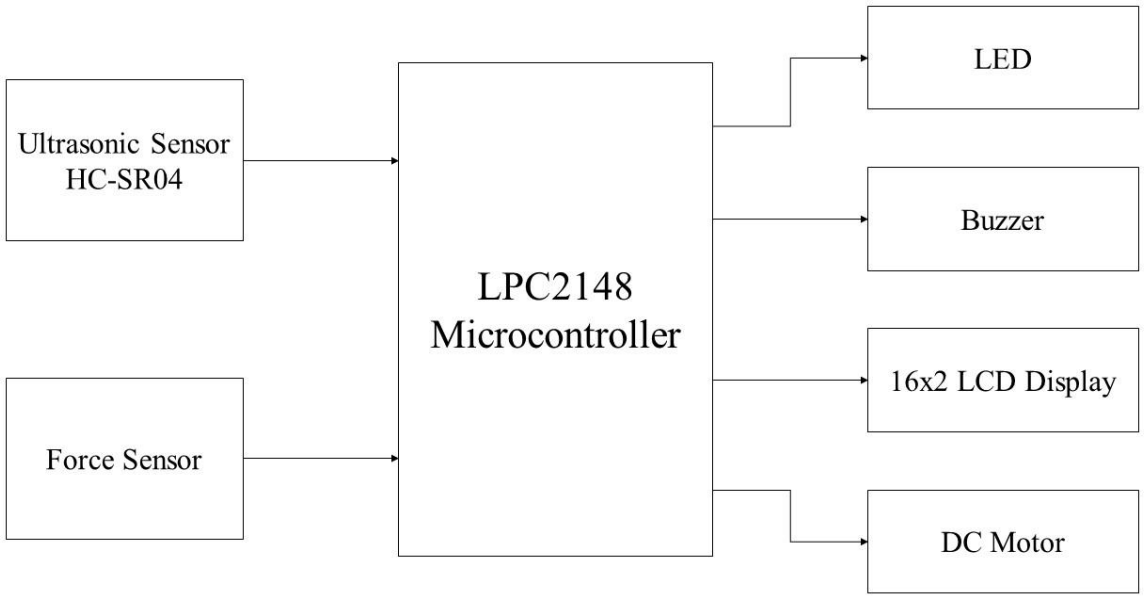


Fig 4.1 Block Diagram

## 4.2 Flowchart

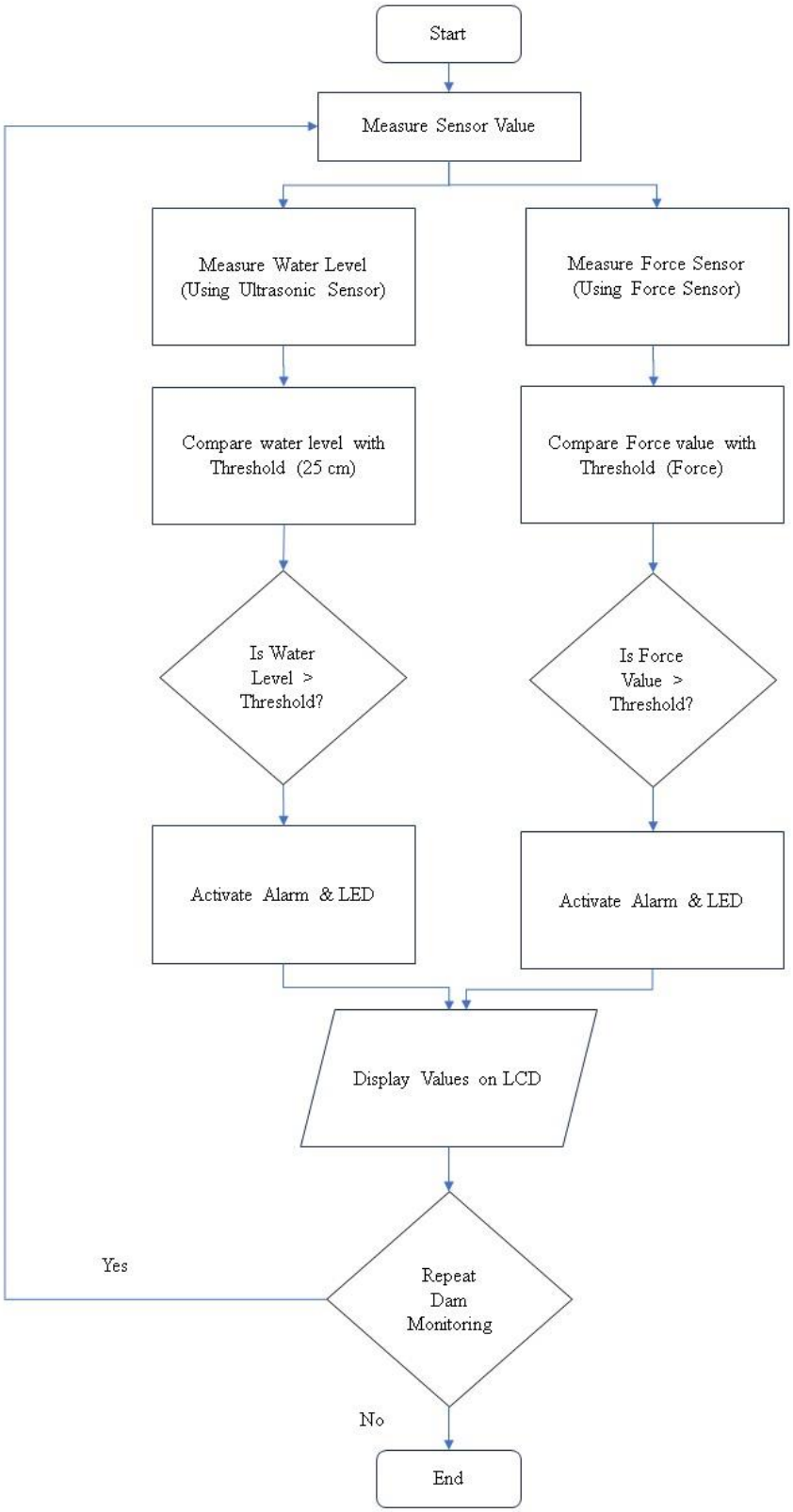


Fig 4.2 Flow Chart

## Chapter 5

# RESULT

### 5.1 Hardware implementations:

Water Level Detection: The ultrasonic sensor accurately measures the water level, with readings displayed on the LCD. The system successfully triggers an alarm and controls the motor when the water level exceeds the set threshold.

Force Sensor Detection: The force sensor provides real-time feedback on any abnormal force applied to the dam structure. The system detects and activates the alarm if the force exceeds the threshold, ensuring timely intervention.

LCD Display: The LCD shows the force sensor value and water level in real-time. The display is clear and easy to read, with updates happening at regular intervals.

Alarm and LED: Both the buzzer and the LED are triggered when the system detects any threshold breach (high water level or excessive force).

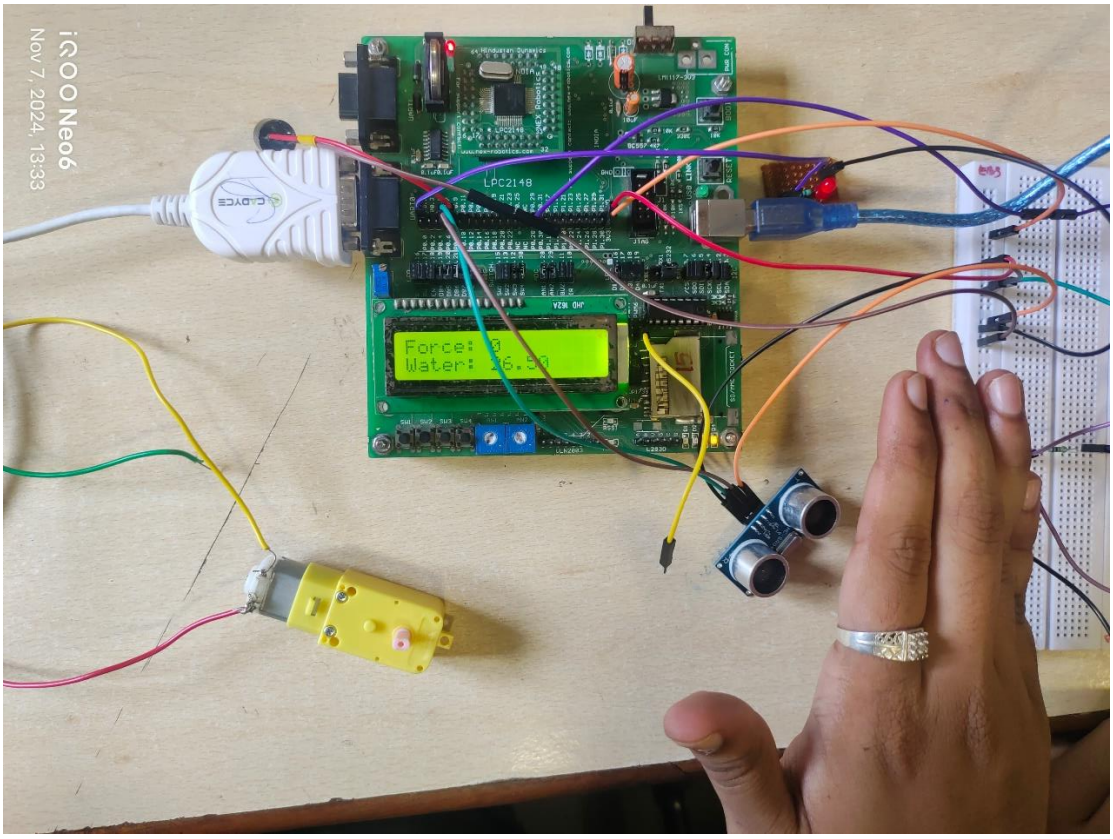


Fig 5.1 Water level measurement using Ultrasonic Sensor

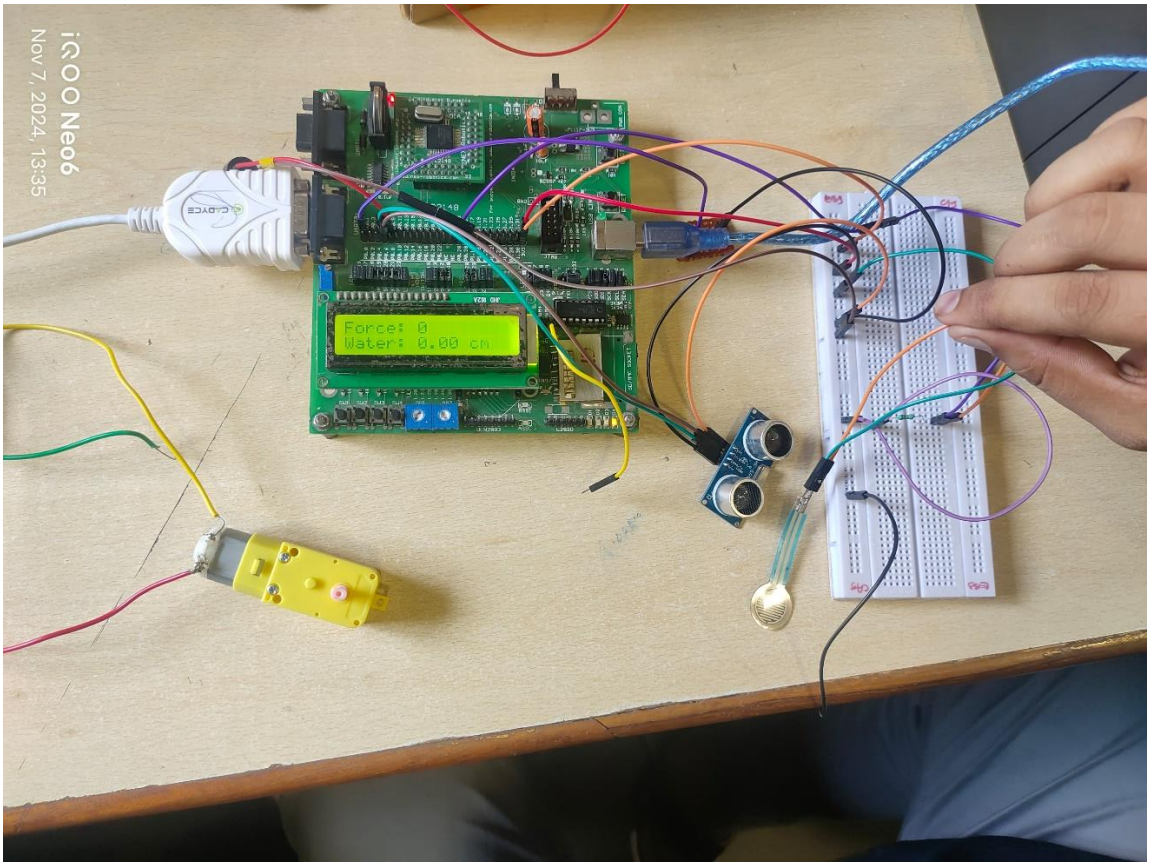


Fig 5.2 Force measurement using Force Sensor (Without giving force)

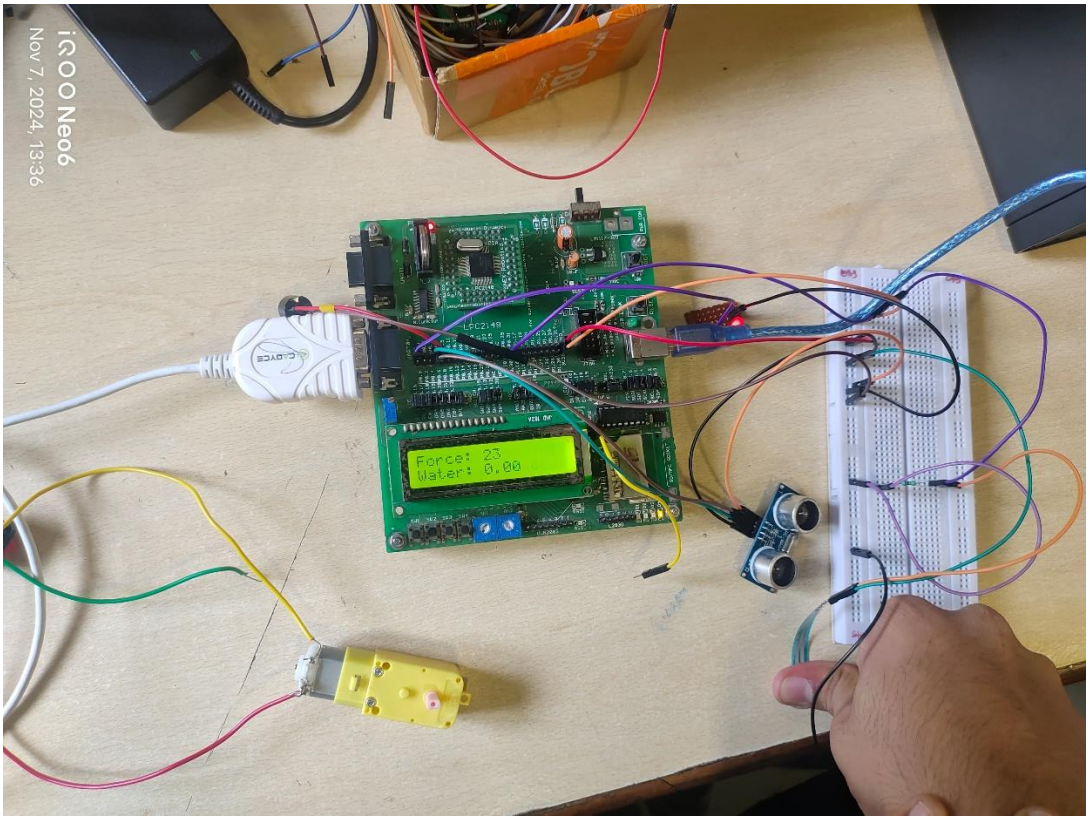


Fig 5.2 Force measurement using Force Sensor (With force)

**5.2 Code:**

```
#include <lpc214x.h>
#include <stdio.h>

#define bit(x) (1 << x)
#define delay for(i = 0; i < 30000; i++);  // Increased delay for stable operation

unsigned int i;
#define TANK_HEIGHT_CM 30.0 // Maximum water level height
#define FORCE_THRESHOLD 20  // Threshold for force sensor to activate buzzer
#define WATER_LEVEL_THRESHOLD 25.0 // Threshold water level to activate buzzer
#define TRIG_PIN (1 << 6) // P0.6 for Trigger
#define ECHO_PIN (1 << 7) // P0.7 for Echo
#define BUZZER_PIN (1 << 9) // P0.9 for the buzzer
#define INTERNAL_LED (1 << 1) // P0.1 for internal LED

void lcd_init(void);
void dat(unsigned char);
void cmd(unsigned char);
void string(char *);
void adc_init(void);
unsigned int read_adc(void);
float read_ultrasonic_distance(void);
void timer_init(void);
void display_sensor_values(float water_level, unsigned int force_value);

int main() {
   float distance, water_level;
   unsigned int force_value;

   IO0DIR |= (0x7F << 16); // Set P0.16 to P0.22 as output for LCD
   IO0DIR |= TRIG_PIN;     // Set TRIG_PIN as output
   IO0DIR &= ~ECHO_PIN;    // Set ECHO_PIN as input
   IO0DIR |= BUZZER_PIN;   // Set buzzer pin as output

   // Set internal LED pin (P0.1) as output
   IO0DIR |= INTERNAL_LED; // Set internal LED pin as output (P0.1)

   lcd_init();           // Initialize LCD
   adc_init();           // Initialize ADC
   timer_init();         // Initialize Timer

   while (1) {
      force_value = read_adc();              // Read ADC value from FSR
      distance = read_ultrasonic_distance();  // Measure distance to water surface
      water_level = TANK_HEIGHT_CM - distance; // Calculate water level height
```

```
        // Limit the water level to be between 0 and 30 cm
        if (water_level < 0) {
            water_level = 0;
        } else if (water_level > TANK_HEIGHT_CM) {
            water_level = TANK_HEIGHT_CM;
        }

        display_sensor_values(water_level, force_value); // Display sensor values on LCD

        // Check if either the force sensor or water level exceeds thresholds
        if (force_value > FORCE_THRESHOLD || water_level > WATER_LEVEL_THRESHOLD) {
            IO0SET |= BUZZER_PIN;  // Turn on buzzer
            IO0SET |= INTERNAL_LED; // Turn on internal LED
            delay;                 // Wait for a short time
            IO0CLR |= INTERNAL_LED;  // Turn off internal LED
            delay;                 // Wait for a short time
        } else {
            IO0CLR |= BUZZER_PIN;  // Turn off buzzer
        }

        delay;  // Small delay for readability
    }
}

void lcd_init(void) {
    cmd(0x02);  // Initialize in 4-bit mode
    cmd(0x28);  // 4-bit mode, 2 lines, 5x7 dots
    cmd(0x0C);  // Display ON, Cursor OFF
    cmd(0x06);  // Auto-increment cursor
    cmd(0x01);  // Clear display
    delay;      // Delay to allow LCD to process the commands
}

void cmd(unsigned char a) {
    IO0PIN &= ~(0x7F << 16);    // Clear P0.16 to P0.22
    IO0PIN |= ((a >> 4) << 19); // Send higher nibble to P0.19 to P0.22
    IO0CLR |= bit(16);          // RS = 0 for command
    IO0CLR |= bit(17);          // RW = 0 for write
    IO0SET |= bit(18);          // EN = 1
    delay;
    IO0CLR |= bit(18);          // EN = 0

    IO0PIN &= ~(0x7F << 16);    // Clear P0.16 to P0.22
    IO0PIN |= ((a & 0x0F) << 19); // Send lower nibble to P0.19 to P0.22
```

```
    IO0CLR |= bit(16);          // RS = 0 for command
    IO0CLR |= bit(17);          // RW = 0 for write
    IO0SET |= bit(18);          // EN = 1
    delay;
    IO0CLR |= bit(18);          // EN = 0
}

void dat(unsigned char b) {
    IO0PIN &= ~(0x7F << 16);    // Clear P0.16 to P0.22
    IO0PIN |= ((b >> 4) << 19); // Send higher nibble to P0.19 to P0.22
    IO0SET |= bit(16);          // RS = 1 for data
    IO0CLR |= bit(17);          // RW = 0 for write
    IO0SET |= bit(18);          // EN = 1
    delay;
    IO0CLR |= bit(18);          // EN = 0

    IO0PIN &= ~(0x7F << 16);    // Clear P0.16 to P0.22
    IO0PIN |= ((b & 0x0F) << 19); // Send lower nibble to P0.19 to P0.22
    IO0SET |= bit(16);          // RS = 1 for data
    IO0CLR |= bit(17);          // RW = 0 for write
    IO0SET |= bit(18);          // EN = 1
    delay;
    IO0CLR |= bit(18);          // EN = 0
}

void string(char *p) {
    while (*p != '\0') {
        dat(*p++);
    }
}

void adc_init(void) {
    // Enable power for ADC0
    PCONP |= (1 << 12);

    // Set AD0.1 pin (P0.28) for ADC function
    PINSEL1 &= ~(3 << 24);  // Clear bits
    PINSEL1 |= (1 << 24);   // Set to 01 for AD0.1

    // Set ADC0 control register
    AD0CR = (1 << 1) |      // Select AD0.1
            (4 << 8) |      // ADC clock: PCLK / 5
            (1 << 21);      // ADC enabled
}

unsigned int read_adc(void) {
```

```
    unsigned int result;

    AD0CR |= (1 << 24);              // Start conversion

    // Wait for the conversion to complete
    while (!(AD0DR1 & (1 << 31)));

    result = (AD0DR1 >> 6) & 0x3FF;  // Extract 10-bit result (bits 6-15)

    AD0CR &= ~(1 << 24);             // Stop conversion

    return result;
}

float read_ultrasonic_distance(void) {
    float distance;
    unsigned long start_time, travel_time;

    // Trigger the sensor
    IO0SET = TRIG_PIN; // Send 10us pulse
    delay;             // Keep high for 10us
    IO0CLR = TRIG_PIN; // Pull it low

    // Wait for echo
    while (!(IO0PIN & ECHO_PIN)); // Wait for echo start
    start_time = T0TC; // Record start time

    while (IO0PIN & ECHO_PIN); // Wait for echo end
    travel_time = T0TC - start_time; // Calculate travel time

    // Calculate distance in cm (speed of sound is 34300 cm/s)
    distance = (travel_time * 0.0343) / 2; // Divide by 2 for one-way distance
    return distance;
}

void timer_init(void) {
    T0TCR = 0x02; // Reset timer
    T0PR = 15;    // Set prescaler for 1us resolution
    T0TCR = 0x01; // Enable timer
}

void display_sensor_values(float water_level, unsigned int force_value) {
    char buffer[16];
    sprintf(buffer, "Force: %u", force_value);
    cmd(0x01);      // Clear display
    delay;          // Delay to allow LCD to process the command
```

```
    string(buffer); // Display force value on LCD

    sprintf(buffer, "Water: %.2f cm", water_level);
    cmd(0xC0);      // Move to second line
    string(buffer); // Display water level on LCD
}
```

## Chapter 8

# CONCLUSION

The project successfully implements an IoT-based water level monitoring system for dams using the LPC2148 microcontroller. This system integrates an ultrasonic sensor to continuously measure water levels, while an ADC processes data from a force sensor to detect pressure changes. When preset thresholds for water level or force are exceeded, a buzzer and LED are activated, signalling potential risks. The real-time display on an LCD provides both force sensor and water level readings, enhancing situational awareness for dam operators. Through efficient sensor integration and the use of IoT, this system promotes proactive dam management, helping to prevent overflows and maintain safety in surrounding areas.