International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 08 | Aug - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Arduino Based Digital Fuel Level Indicator

Mr.Patil Gopal Bhaskar, Mr.Dandge Ashish Gajendra

Mechanical Engineering Department, K.E.Society's Rajarambapu Institute of Technology(Polytechnic),Lohagaon

Mechanical Engineering Department ,Marathwada Mitra Mandals Polytechnic,Thergaon

---

Abstract - The project "Arduino-based Digital Level Indicator" aims to design a precise and efficient system for measuring and displaying the liquid level in a container. Utilizing an Arduino microcontroller, the system is integrated with sensors. The digital output is then displayed on an LCD screen for easy reading. The device can be employed in various applications, including industrial tanks, water reservoirs, or fuel storage, offering an accurate, automated alternative to traditional mechanical gauges. The project demonstrates the use of simple, cost-effective components and provides real- time monitoring of fluid levels with high accuracy and minimal user intervention. Through the implementation of this system, users benefit from enhanced efficiency, early warning features, and a clear digital representation of the fluid levels. This project highlights the potential of Arduino-based solutions for practical, everyday automation tasks

Key Words: fuel indicator,ardunio

1.INTRODUCTION

The aim of our project is to monitor the level of the fuel in the vehicle fuel tank and to automatically indicate the level information digitally, numerical value through LCD.

We are already aware that modern vehicles display the amount of fuel in the fuel tank by the means of analog indicators, which oscillates between E (empty) and F (full) at its extreme ends or by digital bars running through E (empty) and F (full) indicators.

To the contrary every one of us might have experienced the problem with improper estimations of the current fuel indicating system. Thus, digital (numeric) fuel indicator system will help us exterminate common problems like-

1. Misinterpretation of the amount of fuel left by the drivers
2. Petrol pumps theft cases.

REQUIREMENT ANALYSIS

The system must accurately measure and display real-time fuel levels in a motorcycle. It should be compact, cost-effective, easy to install, and resistant to fuel and environmental conditions. The fuel sensor should provide stable readings, avoiding fluctuations.

3.2 COMPONENTS REQUIRED

Hardware –

- Arduino Uno – Microcontroller for data processing
- Fuel Level Sensor – Float sensor, water flow sensor to detect fuel levels
- 16x2 LCD with I2C Module – Displays fuel level digitally
- Battery/Voltage Regulator – Powers the system (12V to 5V)
- Resistors, Capacitors, Wires, and Breadboard – For circuit assembly

Software –

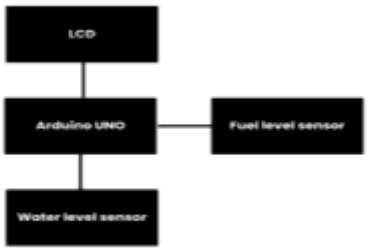
- Arduino IDE – For writing and uploading code
- Serial Monitor – Used for debugging and calibration

VARIOUS TYPES OF MODULES

- Display Module (16x2 LCD with I2C) – Shows real-time fuel level
- Fuel Sensor Module
- Float Sensor – Common in vehicles, resistance-based
- Water Flow Sensor – Uses a magnetic rotor for fuel detection
- Power Module: Battery / Voltage Regulator – Converts 12V to 5V

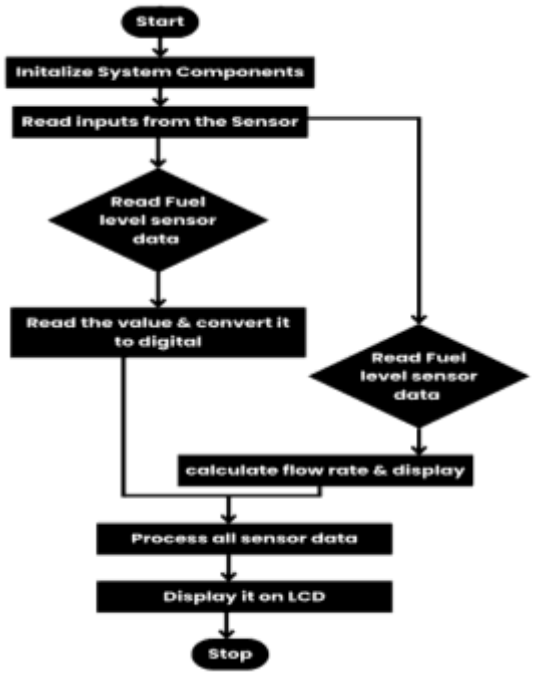
SYSTEM MODELING

BLOCK DIAGRAM



FLOWCHART

SOFTWARE AND PROGRAMMING



PROGRAM STRUCTURE

Initialization:

Include necessary libraries for LCD and sensor operation

Define pin connections for the fuel sensor and display module

Sensor Data Reading:

Read the fuel level sensor and water level sensor output (analog/digital value)

Convert the sensor reading into a meaningful fuel level percentage and water level in l/s

Processing & Calibration:

Map raw sensor values to actual fuel levels

Apply calibration if needed for improved accuracy

Display Output:

Show the fuel level on the 16x2 LCD screen

Update the display continuously for real-time monitoring

```
#include<Wire.h>
#include <LiquidCrystal_I2C.h>

// Initialize the I2C LCD with its I2C address (usually 0x27 or 0x3F)
LiquidCrystal_I2C lcd(0x27, 16, 2);

const int flowPin = 2;   // Pin connected to the sensor signal wire
volatile int pulseCount = 0;  // Counter for pulses

// Variables for calculating flow rate
float flowRate = 0.0;
unsigned long oldTime = 0;

// Pulses per liter (consult sensor datasheet)
const float calibrationFactor = 4.5;  // Example value, adjust based on your sensor



//--------------------FUEL SENSOR--------------------
const int fuelSensorPin = A0;
const int sensorMinValue = 1018;   // Empty tank (calibrate this)
const int sensorMaxValue = 1020; // Full tank (calibrate this)
    flowRate = ((float)pulseCount / calibrationFactor);  // Liters per second
    float liters = flowRate * (elapsedTime / 1000.0);  // Liters passed in elapsed time

    // Print values to Serial Monitor
    Serial.print("Flow Rate: ");
    Serial.print(flowRate, 2);
    Serial.print(" L/s | Liters: ");
    Serial.println(liters, 2);

    // Display values on LCD
    lcd.setCursor(0, 1);
    lcd.print("Flow: ");
    lcd.print(flowRate, 2);
    lcd.print(" L/s");




    pulseCount = 0;  // Reset pulse count
    oldTime = currentTime;  // Update time
  }

//--------------------------FUEL-------------------------
  // Read sensor value
  int sensorValue = analogRead(fuelSensorPin);

  // Add to smoothing array
  smoothingReadings[smoothingIndex] = sensorValue;
  smoothingIndex = (smoothingIndex + 1) % 10;

  // Calculate average
  int smoothedValue = 0;
  for (int i = 0; i < 10; i++) {
    smoothedValue += smoothingReadings[i];
  }
  smoothedValue /= 10;

  // Map sensor value to percentage
  int fuelPercentage = map(smoothedValue, sensorMinValue, sensorMaxValue, 0, 100);
  fuelPercentage = constrain(fuelPercentage, 0, 100);

  // Display on LCD
  lcd.setCursor(0, 0);
  lcd.print("Fuel Level:");
  lcd.print(fuelPercentage);
  lcd.print("%  ");

  delay(500);

//-------------------------FUEL-------------------------
}
```

```
void setup() {
  pinMode(flowPin, INPUT_PULLUP);  // Configure pin as input with pull-up
  attachInterrupt(digitalPinToInterrupt(flowPin), countPulse, RISING);  // Interrupt on rising pulse
  Serial.begin(9600);  // Start Serial Monitor
  Serial.println("Water Flow Sensor Test");


  // Initialize the LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("System Install");
  delay(2000);
  lcd.clear();

  //-------------------------FUEL-------------------------
  // Initialize smoothing array
  for (int i = 0; i < 10; i++) {
    smoothingReadings[i] = 0;
  }
  //------------------------------------------------------
}

void loop() {
unsigned long currentTime = millis();
  unsigned long elapsedTime = currentTime - oldTime;
  // Calculate flow rate every second
  if (elapsedTime >= 1000) {


  }


// Interrupt function to count pulses
void countPulse() {
  pulseCount++;
}
```

SYSTEM DEVELOPMENT

The development of the Arduino-based Digital Fuel Level Indicator involves the integration of hardware and software components to ensure accurate fuel measurement and real-time display.

Development Steps:

1. Component Selection: Choose the appropriate fuel sensor, microcontroller (Arduino Uno), display unit (16x2 LCD with I2C), and power supply
2. Circuit Design: Connect the sensor to the Arduino and integrate the LCD display using an I2C module
3. Programming: Write an Arduino program to read sensor data, process it, and display the fuel level
4. Testing & Calibration: Verify sensor accuracy by comparing measured values with actual fuel levels
5. Final Assembly: Secure components in a protective casing for durability

SYSTEM WORKING

The system works by continuously monitoring fuel levels and displaying the data in a digital format

Working Principle:

The fuel level sensor detects the amount of fuel in the tank and converts it into an electrical signal.The Arduino Uno processes the signal and maps it to a fuel percentage (0-100%).The 16x2 LCD display shows the fuel level in real-time

The system updates the reading continuously, ensuring accurate monitoring while riding

CIRCUIT DIAGRAM USING TINKER CAD

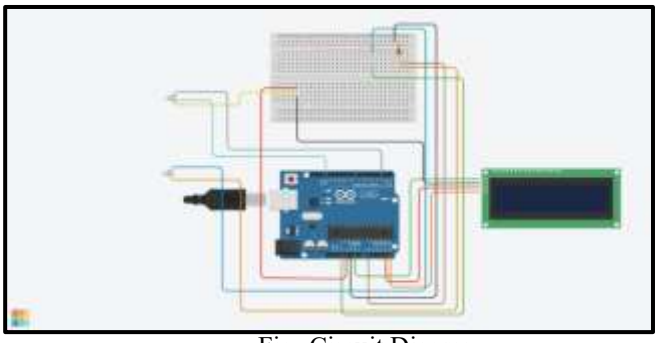
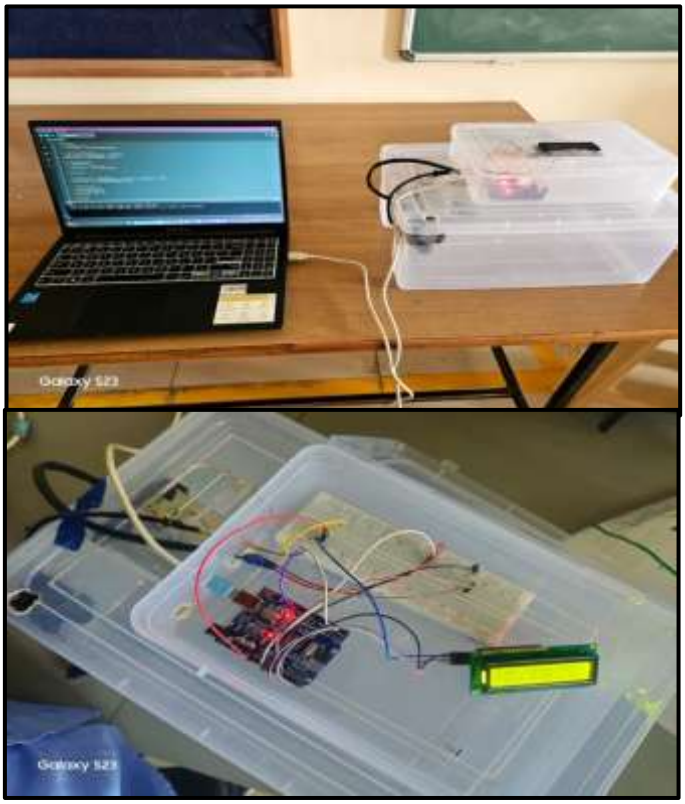


Fig. Circuit Diagram



ADVANTAGES AND LIMITATION

Advantages:

- Provides accurate and real-time fuel level readings
- Displays readings in percentage format for better clarity
- Ensures stable and consistent performance with minimal fluctuations
- Uses cost-effective components, making it affordable
- Offers easy installation with minimal vehicle modifications
- Can be upgraded with IoT features for remote monitoring

Limitation:

- Requires proper calibration for accurate readings
- May face issues with sensor stability in highly vibrating conditions
- Dependence on the Arduino may limit scalability without upgrades
- Power supply fluctuations can affect performance if not properly regulated

SCOPE OF FUTURE WORK

- IoT Integration: Adding Bluetooth or Wi-Fi connectivity for remote monitoring via smartphone apps
- Mobile App Development: Enabling real-time notifications and alerts for low fuel warnings
- Enhanced Display: Replacing the 16x2 LCD with an OLED screen for improved visibility
- Fuel Consumption Analysis: Integrating data logging to track fuel usage patterns
- Multi-Vehicle Adaptability: Expanding the system for use in cars, trucks, and commercial vehicles

APPLICATION

1. Automotive Industry
   - Used in motorcycles, cars, trucks, and buses for real-time fuel level monitoring
   - Provides precise digital readings, reducing fuel-related breakdowns
   - Helps fleet managers track fuel consumption and prevent fuel theft
2. Smart Vehicles & IoT Integration
   - Can be integrated with IoT for remote fuel tracking via mobile apps
   - Sends low-fuel alerts to drivers or fleet owners
   - Helps in fuel efficiency analysis for smart vehicle systems
3. Industrial & Generator Fuel Monitoring
   - Used in diesel generators, heavy machinery, and industrial vehicles
   - Ensures optimal fuel usage, preventing unexpected shutdowns
   - Helps businesses manage fuel inventory and costs effectively
4. Marine and Aviation
   - Applied in boats, ships, and aircraft for accurate fuel tracking
   - Prevents fuel shortages during long trips
   - Enhances safety and efficiency in transport and logistics

3. CONCLUSIONS

The Arduino-Based Digital Fuel Level Indicator successfully addresses the drawbacks of conventional analog systems by providing a precise, digital display of fuel levels. The system effectively improves user convenience by displaying accurate readings, reducing the risk of unexpected fuel depletion. With future enhancements like IoT integration and data analysis, this project has strong potential to contribute to smart vehicle monitoring solutions.

REFERENCES

1.R. S. Khurmi & J. K. Gupta, A Textbook of Automobile Engineering, S. Chand Publishing, 2013

2.C. Ray, Automotive Fuel Systems: Theory & Function, McGraw-Hill Education, 2017

3.M. J. Palumbo & D. C. Rice, Embedded Systems & Microcontroller Applications, Pearson, 2019

4. J. Patel et al., IoT-Based Fuel Monitoring System for Smart Vehicles, IEEE Conference on Embedded Systems, 2022

5. S. Sharma et al., Digital Fuel Level Indicators: A Comparative Study, International Conference on Mechatronics, 20.