21EEO305T

SURVEILLANCE TECHNOLOGY

DEVELOPMENT LIDAR SENSOR AND MAPPING ENVIRONMENT

Anugrah Samuel Frank (RA221101801042)
Vijai Krishna Prasanna (RA2211018010007)
Nabeel Ahmed MD (RA2211018010038)
Maitry Parmar (RA2211018010015)

DEPARTMENT OF MECHATRONICS ENGINEERING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

NOVEMBER 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

This is to certify that the project report titled "Development of a LiDAR System and Mapping Environment" is the genuine work of Anugrah Samuel Frank, Vijai Krishna Prasanna, Nabeel Ahmed MD, and Maitry Parmar, who have carried out the project under the supervision of Dr. K Subha as part of the course 21EEO305T Surveillance Technology during the Odd Semester of the Academic Year 2023-24.

Signature of Course Instructor **Signature of Head of Department**

TABLE OF CONTENT

1.1 INTRODUCTION

INTRODUCTION

This project explores the development of a LiDAR (Light Detection and Ranging) system using the VL53L0X laser sensor, MPU6050 sensor, a servo motor, and the Arduino Uno microcontroller. LiDAR technology is a key tool in remote sensing, utilizing laser pulses to measure distances with high accuracy and create detailed 3D environmental maps. This system was designed to capture precise spatial data by emitting laser pulses, measuring the time taken for reflection, and computing distances based on these measurements. The MPU6050 sensor adds an extra dimension by providing orientation data, enhancing the system's capability to interpret its spatial positioning in real-time.

For this project, the Arduino IDE was used to program the Arduino Uno microcontroller, enabling it to control the sensors and servo motor. Additionally, VS Code was employed to capture data values, allowing for graphical representation and analysis of the recorded distances and orientations. Such a setup demonstrates how low-cost components can be integrated to develop a LiDAR system with applications in autonomous vehicles, robotics, environmental monitoring, and obstacle detection.

OBJECTIVE

The primary objective of this project is to design and implement a LiDAR system to measure distances and orientation accurately. By integrating the VL53L0X laser sensor for distance measurement and the MPU6050 sensor for orientation detection, the project aims to create a responsive, real-time system for spatial mapping. This system is intended to provide accurate data visualization and environmental awareness, with potential applications in obstacle detection, robotic navigation, and autonomous systems.

MECHATRONIC PERSPECTIVE

From a mechatronic perspective, this project integrates mechanical, electronic, and computational elements to achieve precise environmental mapping. The VL53L0X laser sensor serves as the primary distance measurement component, using time-of-flight principles to capture accurate spatial data. The MPU6050 sensor provides real-time orientation and motion data, which are crucial for understanding and adjusting the sensor's position in 3D space.

The servo motor rotates the sensors, allowing for a 180-degree field of view, thus enhancing the system's ability to capture a comprehensive environmental profile. The Arduino Uno microcontroller serves as the central control unit, managing sensor inputs, motor control, and data processing. This setup is programmed using the Arduino IDE, while VS Code handles data visualization and plotting, creating an interactive representation of the distance and orientation data. Through this integration, the project showcases a compact, functional LiDAR system with versatile applications in autonomous navigation, robotics, and smart systems.

HARDWARE

1) Arduino Nano

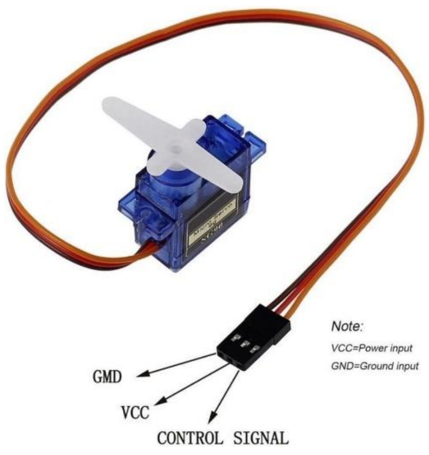
The Arduino Nano is a microcontroller board based on the ATmega328P processor, widely used in electronics projects for its simplicity, flexibility, and extensive community support. It features 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, USB connection, power jack, and reset button, allowing it to interface with various sensors, motors, and other electronic components. The board can be programmed using the Arduino IDE, which provides an easy-to-use programming environment for writing, uploading, and testing code.
The Arduino Uno is chosen for its reliability, ease of integration with sensors, and compatibility with various programming libraries, making it an ideal choice for prototyping sensor-based systems like LiDAR.

2) VL53L0X Lazer Sensor

The VL53L0X sensor is a time-of-flight (ToF) distance sensor manufactured by STMicroelectronics. It uses a laser (VCSEL - Vertical-Cavity Surface-Emitting Laser) to measure distances with high precision and reliability, making it suitable for various applications in robotics, drones, and automation. It can measure distances from around 30 mm up to 2 meters in optimal conditions. Operating Voltage is 2.8V to 5V, often compatible with 3.3V and 5V systems.It supports I²C communication.

3) SG90 Servo Motor

A servomotor is a closed-loop servomechanism that uses position feedback (either linear or rotational position) to control its motion and final position. The input to its control is a signal (either analog or digital) representing the desired position of the output shaft. It uses an internal potentiometer to continuously measure the current position of the shaft, allowing it to adjust and reach the desired position accurately. The SG90 motor can rotate to a specific angle between 0 and 180 degrees, making it ideal for applications needing limited, precise control rather than continuous rotation. Controlled via PWM (Pulse Width Modulation), it adjusts its angle based on the duration of pulses received. It's Operating Voltage is 4.8V to 6V (usually compatible with 5V logic, Torque is Around 1.8 kg/cm at 4.8V



4) BREAD BOARD:

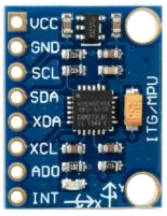
A breadboard is an essential tool in prototyping, providing a convenient way to connect components like pressure sensors and microcontrollers (such as the ESP32) without the need for soldering. It’s particularly valuable for testing and iterating on circuit designs, as connections can be easily modified or removed as needed.

5) MPU6050

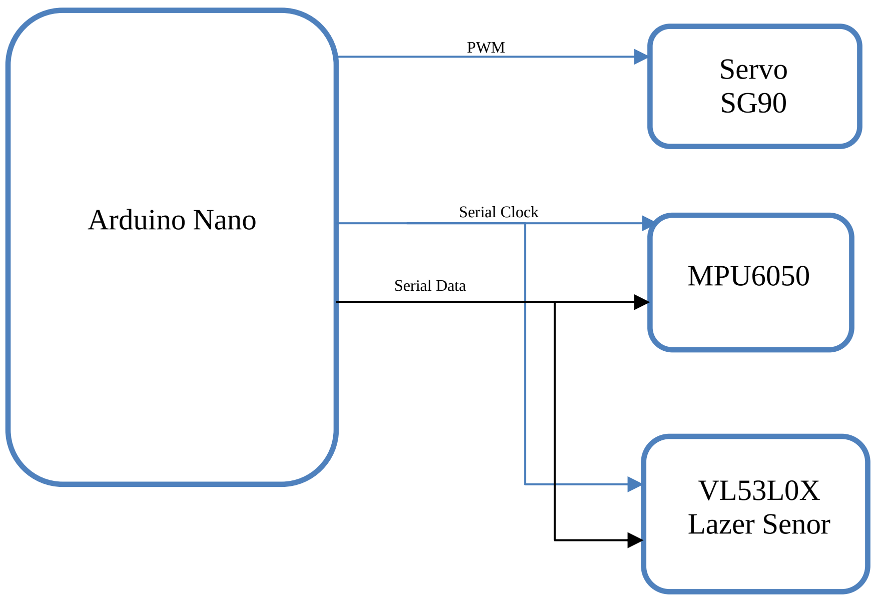
The MPU6050 is a popular 6-axis motion tracking device that integrates a 3-axis accelerometer and a 3-axis gyroscope in a single chip. It’s widely used in applications requiring motion tracking and orientation detection, including robotics, drones, and wearable devices. The MPU6050 communicates through the I²C protocol, making it compatible with microcontrollers such as the Arduino, ESP32, and Raspberry Pi.

3-Axis Accelerometer and 3-Axis Gyroscope:

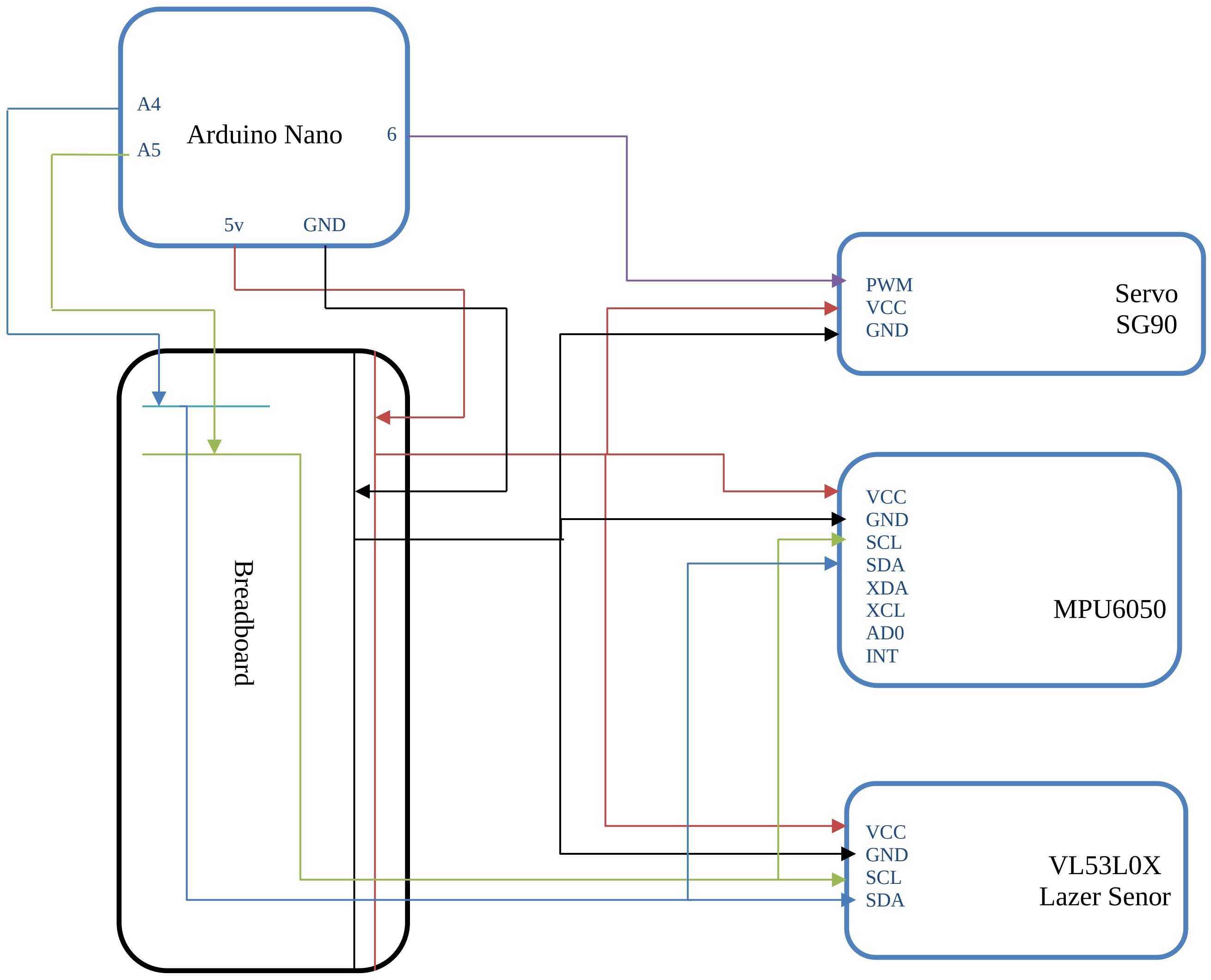
- Measures acceleration (X, Y, Z) and angular velocity (pitch, roll, yaw), providing comprehensive motion and orientation data.
- Range selection for both accelerometer (±2g to ±16g) and gyroscope (±250°/s to ±2000°/s), allowing for customization based on the application’s requirements



BLOCK DIAGRAM



CIRCUIT DIAGRAM



EXPLANATION OF THE WORKING MODEL

In this setup, the VL53L0X laser sensor is mounted on an SG90 servo motor, allowing the sensor to capture distance measurements as the servo rotates. Additionally, the MPU6050 sensor provides orientation and acceleration data in 3D space. The Arduino IDE handles the programming of these components to operate synchronously, enabling efficient data capture. The data from the Arduino can then be accessed and visualized in Visual Studio Code (VS Code) using Python for plotting.

Components and Libraries

1. **Servo Motor (SG90)**:
   - The **SG90 servo motor** rotates the VL53L0X sensor at specified angles. As it rotates, the laser sensor takes distance measurements at each angle, creating a polar coordinate data set for plotting.
   - **Library**: #include <Servo.h>
2. **VL53L0X Laser Sensor**:
   - Measures the distance to objects as the servo rotates. This setup enables creating a **polar plot** of distance versus angle, capturing the spatial layout of the surroundings within the sensor's range.
   - **Library**: #include <Adafruit_VL53L0X.h>
3. **MPU6050 (Accelerometer and Gyroscope)**:
   - Captures linear acceleration (along X, Y, and Z axes) and angular velocity, providing information on the device's orientation in 3D space.
   - With this data, you can create a **3D Cartesian plot** to visualize the orientation and movement of the sensor setup.
   - **Library**: #include <MPU6050.h>
4. **Communication and I2C Protocol**:
   - The **Wire** library enables I²C communication, used for both the VL53L0X and MPU6050 sensors to communicate with the microcontroller (e.g., Arduino).
   - **Library**: #include <Wire.h>

Process Overview

1. **Arduino Code**:
   - **Servo Control**: The SG90 is programmed to rotate in small increments (e.g., 1°), pausing briefly to allow the VL53L0X to capture a distance measurement at each angle.
   - **Distance Measurement**: The VL53L0X captures the distance at each angle, storing it as a pair of polar coordinates (angle, distance).
   - **Orientation Data**: The MPU6050 continuously provides acceleration and angular velocity data for the X, Y, and Z axes, capturing orientation and movement.
2. **Data Transfer**:
   - The Arduino IDE program reads and processes data from the sensors, then sends it to the serial port.
   - Visual Studio Code (VS Code) can access these values from the serial port, making it possible to read and visualize the data in real-time.
3. **Visualization in VS Code**:
   - **Polar Plot (Angle vs. Distance)**: A polar plot can be created in Python (e.g., using matplotlib) to visualize the surrounding layout captured by the VL53L0X.
   - **3D Cartesian Plot for MPU6050**: Using the MPU6050's accelerometer and gyroscope data, a 3D Cartesian plot can represent the orientation of the sensor setup, showing linear acceleration and angular velocity in 3D space.

Appendix

Program Outcome

S.No	Program Outcomes (Pos)	Relation to Project
PO01	**Engineering Knowledge** ***In this project:*** Highlight the knowledge of science and engineering fundamentals used	This LiDAR project demonstrates the application of engineering fundamentals by integrating key mechatronics principles—sensor technology, control systems, and real-time data processing. The VL53L0X sensor uses time-of-flight technology to measure distances accurately, while the MPU6050 provides orientation data via its accelerometer and gyroscope, enabling 3D spatial awareness. The Arduino Uno microcontroller coordinates sensor input and servo motor control through PWM signals and I²C communication, executing precise motion control and data acquisition cycles. The project's code leverages serial communication and visualization in VS Code for real-time data plotting, merging hardware and software into a cohesive mechatronic system.
PO02	**Problem solving and analysis** ***In this project:*** Highlight how did you identify, formulate, analyse the problem undertaken.	The problem addressed in this project was the need for a low-cost, real-time LiDAR system for spatial mapping and obstacle detection. The team identified the challenge of achieving accurate distance and orientation measurement using affordable components. The problem was formulated as designing a LiDAR model that could integrate distance and orientation sensors with a microcontroller to capture and visualize 3D spatial data. Analysis involved selecting appropriate sensors (VL53L0X for distance and MPU6050 for orientation) and developing algorithms for sensor integration, data processing, and visualization to ensure reliable and precise environmental mapping.
PO03	**Design & Developing solutions** ***In this project:*** Highlight the design (mechanical, electronics, mechatronics, algorithmic, control etc.) and techniques used.	**Mechanical Design** The mechanical design focused on mounting the VL53L0X LiDAR sensor onto an SG90 servo motor to allow rotation and extend its field of view. By mounting the sensor on a servo motor capable of rotating 180 degrees, the system could cover a wider angle, enabling comprehensive environmental scanning. The servo's rotation was carefully calibrated to allow incremental movement and precise positioning, with each angle offering a unique distance measurement. This setup was essential for capturing a polar dataset that could be later processed for spatial visualization. **Electronics Design** The electronic design incorporated an Arduino Uno microcontroller, which served as the central control unit, managing inputs from the sensors and controlling the servo motor. The VL53L0X sensor and MPU6050 module were interfaced with the Arduino via I²C

communication, minimizing the number of required pins and simplifying the circuit. A breadboard was used for prototyping connections between components, which allowed for easy adjustments during development. Power was supplied through the Arduino’s 5V and GND pins, making the design efficient and compact.

Mechatronics Integration

This project demonstrated a mechatronic approach by seamlessly integrating mechanical components (servo motor), electronics (sensors, microcontroller), and software for control and data processing. The Arduino Uno, programmed through the Arduino IDE, handled both servo control and sensor data acquisition. By synchronizing the servo motor rotation with data collection from the VL53L0X and MPU6050, the system achieved real-time spatial mapping. This integration of mechanical and electronic components highlights core mechatronic design principles, where the interaction between hardware and software is optimized for system functionality.

Algorithmic Design

The algorithm was designed to rotate the servo incrementally, collect distance and orientation data at each angle, and store these values in arrays for later processing. The program included functions to measure distances using the VL53L0X sensor and to capture accelerometer and gyroscope readings from the MPU6050. Data was formatted into polar coordinates (angle, distance) for the LiDAR readings and Cartesian coordinates for the orientation data, allowing for both 2D and 3D plotting. The algorithm ensured that each sensor reading was synchronized with the servo position, producing accurate and organized datasets for visualization.

Control Techniques

Control of the servo motor was managed using PWM (Pulse Width Modulation) signals, allowing the servo to achieve specific angular positions for precise data acquisition. The servo rotated in small increments, pausing at each position to capture distance measurements with the VL53L0X and orientation data with the MPU6050. This control technique enabled efficient and synchronized data collection without overlapping or missing points. Additionally, I^{2}C communication was used to handle data transfer between the sensors and Arduino, allowing for reliable and fast data exchange essential for real-time operation.

Data Processing and Visualization Techniques

Data collected by the Arduino was transmitted to Visual Studio Code (VS Code) via serial communication, where Python scripts processed and visualized the data. The polar plot, generated using matplotlib, displayed distance versus angle, providing a 2D spatial map of the surrounding environment. The 3D orientation data from the MPU6050 was visualized in a Cartesian plot, showing the device’s orientation in real-time. These visualization techniques not only validated the system’s accuracy but also demonstrated its potential for applications in spatial awareness and obstacle detection.

PO04	**Research based investigation of systems:** ***In this project:*** Highlight the literature review that was conducted and how was research-based knowledge gained.	The literature review for this project focused on existing technologies and methodologies related to LiDAR sensors, servo motor integration, and spatial mapping techniques. Key sources included academic papers on sensor fusion, servo control algorithms, and applications of LiDAR in robotics and automation. This research provided insights into best practices for sensor calibration, data collection protocols, and the effective use of I²C communication. Additionally, studies on the VL53L0X sensor's capabilities and the MPU6050's data interpretation were examined to understand their operational limits and optimal usage conditions. By synthesizing this knowledge, the project was informed by proven techniques and innovative approaches, ensuring that the mechanical and electronic designs were grounded in established research, thus enhancing the overall system's effectiveness and reliability.
PO05	**Modern tool usage** ***In this project:*** Highlight the engineering tools or software for design, analysis, analysis, that was/were used to create, select and apply appropriate techniques.	In this project, **Arduino IDE** was utilized for programming the Arduino Uno microcontroller. This platform provided a user-friendly interface for writing and uploading code, facilitating the integration of the VL53L0X sensor and MPU6050 module. The IDE's built-in libraries and example codes accelerated development, enabling efficient control of the servo motor and data acquisition from the sensors. For data analysis and visualization, **Visual Studio Code (VS Code)** was employed. This versatile code editor allowed for the execution of Python scripts that processed the collected data. Using libraries like matplotlib, VS Code enabled the generation of both 2D polar plots and 3D Cartesian plots, providing clear visual representations of the spatial data. This combination of tools ensured a robust workflow for both programming and data analysis, supporting the project's objectives effectively
PO06	**Engineers and society** ***In this project:*** Highlight whether yourproject take into consider public health & safety, cultural, legal, ethical and global consequences and their relationsto the responsibilities of a professional engineer	This project took into account public health and safety by ensuring that the LiDAR system could be applied in environments such as robotics and automation, where safety is paramount. Ethical considerations were also addressed by prioritizing the accurate and responsible use of sensor data, particularly in applications that may impact public spaces. Furthermore, the project acknowledged the importance of adhering to legal standards regarding data collection and privacy. As professional engineers, it is crucial to understand and mitigate the potential global consequences of technology deployment, ensuring that solutions are both beneficial and socially responsible.

PO07	**Environment and sustainability** ***In this project:*** Highlight the components of your project demonstrate knowledge of and need for sustainable development and understand the social and environmental impacts of engineering solutions	This project took into account public health and safety by ensuring that the LiDAR system could be applied in environments such as robotics and automation, where safety is paramount. Ethical considerations were also addressed by prioritizing the accurate and responsible use of sensor data, particularly in applications that may impact public spaces. Furthermore, the project acknowledged the importance of adhering to legal standards regarding data collection and privacy. As professional engineers, it is crucial to understand and mitigate the potential global consequences of technology deployment, ensuring that solutions are both beneficial and socially responsible.
PO08	***In this project:*** Highlight how did you apply, follow and practice commit to professional ethics and responsibilities and norms of engineering practice.	In this project, I adhered to industry standards and ensured transparency with stakeholders by obtaining informed consent. I maintained data integrity, accurately reporting all findings without manipulation. Considerations for societal impact and environmental sustainability were integral to the design, promoting public welfare. Collaboration was fostered through respect for diverse viewpoints, while accountability was emphasized within the team. I engaged in continuous learning to stay updated on engineering ethics and addressed any potential conflicts of interest responsibly.
PO09	**Communication** ***In this project:*** Highlight how did you manage and/or coordinate communication: (between project members, subject-coordinator, outside community, etc. and also with the community at large	In this project, I facilitated regular team meetings to discuss progress and address challenges, ensuring all members were aligned. I coordinated with the subject coordinator for guidance and feedback, utilizing emails and collaborative tools for effective information sharing. Additionally, I engaged with the outside community through surveys and feedback sessions to incorporate their perspectives, fostering a two-way communication flow that enhanced project relevance and impact.
P10	**Individual and team work** ***In this project:*** Highlight how did you function effectively as an individual,and in multi-disciplinary and multi-cultural teams	In this project, I took initiative as an individual by managing specific tasks and ensuring deadlines were met. I collaborated effectively in a multi-disciplinary team, leveraging diverse skill sets to enhance project outcomes. Embracing a multi-cultural environment, I respected different perspectives and fostered inclusive discussions, which facilitated creative problem-solving and strengthened team cohesion. This approach enabled us to achieve common goals while appreciating the unique contributions of each member.
	Lifelong learning	

PO11	***In this project:*** Highlight the opportunity that you gained or to begained for the need for independent and lifelong learning, and possess the capacity to do so.	**In this project**, I recognized the importance of independent learning, especially when integrating new technologies and methodologies. Engaging with various sensors and programming environments enhanced my technical skills and encouraged me to seek additional knowledge beyond the project requirements. This experience underscored the necessity of continuous education in engineering, fostering a mindset geared towards lifelong learning and adaptability to emerging trends and innovations in the field.
PO12	**Project management and finance**	
	In this project: Highlight how was the engineering project-management was done and along with financial and resources and supply constraints.	In this project, effective engineering project management was achieved by establishing a clear timeline, defining milestones, and allocating tasks among team members. Regular meetings facilitated coordination and tracking of progress. Financial constraints were managed by budgeting for components and tools, prioritizing essential purchases, and sourcing materials within a limited budget. Resource management involved careful planning to ensure the availability of equipment and timely access to necessary supplies, helping to avoid delays and optimize project execution.

CODE

ARDUINO IDE:

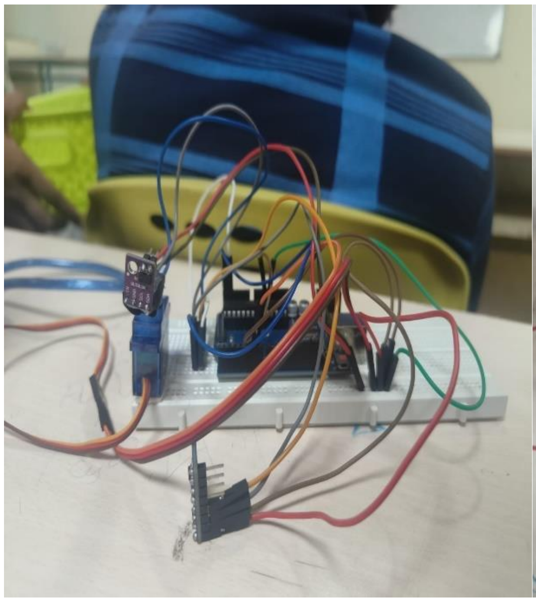
```
#include <Servo.h>
#include <Wire.h>
#include <Adafruit_VL53L0X.h>
#include <MPU6050.h>

Servo myservo;
Adafruit_VL53L0X lox = Adafruit_VL53L0X();
MPU6050 mpu;
int pos_=0;

int distanceArray[180];
int servopin = 6;
int count = 1;

void setup() {
  myservo.attach(servopin);
  Serial.begin(9600);
  Serial.println("\nStarting setup...");

  Wire.begin();
  Serial.println("I2C Initialized");

  // Initialize the VL53L0X sensor
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while (1);
  } else {
    Serial.println("VL53L0X connected successfully!");
  }
  lox.setMeasurementTimingBudgetMicroSeconds(200000);

  // Initialize MPU6050
  mpu.initialize();
  if (!mpu.testConnection()) {
    Serial.println("MPU6050 connection failed.");
    while (1);
  } else {
    Serial.println("MPU6050 connected successfully!");
  }
}

int measure_dist() {
  VL53L0X_RangingMeasurementData_t measure;
  lox.rangingTest(&measure, false);  // Take a distance measurement

  if (measure.RangeStatus != 4) {  // If measurement is valid
    int smoothedDistance = measure.RangeMilliMeter;
    delay(500);
    Serial.print(F("   Distance:  "));
    Serial.print(smoothedDistance);
    Serial.print(F(" mm   "));
    return smoothedDistance;
  } else {
    Serial.println(F("Distance: Out of range"));
    return -1;
  }
}

void measure_mpu() {
  int16_t ax, ay, az;
  int16_t gx, gy, gz;

  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  Serial.print("   Accel X: "); Serial.print(ax);
  Serial.print("   Accel Y: "); Serial.print(ay);
  Serial.print("   Accel Z: "); Serial.print(ay);

  Serial.print("   Gyro X: "); Serial.print(gx);
  Serial.print("   Gyro Y: "); Serial.print(gy);
  Serial.print("   Gyro Z: "); Serial.println(gz);
}

void loop() {
  if (count == 1) {
    for (int pos = 0; pos <= 180; pos++) {
      myservo.write(pos);
      Serial.print("Angle: ");
      Serial.print(pos);  // Debug statement

      int dist = measure_dist();
      pos_=pos;
      distanceArray[pos] = dist;

      measure_mpu();

      //Serial.print(F("  Angle: "));
      //Serial.println(pos);

      delay(100);  // Increased delay to allow sensor to read and process data
    }
    count++;
  }
}
```

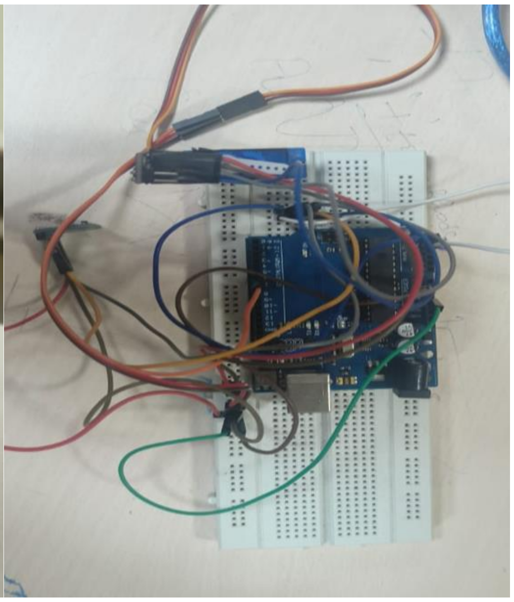
VS CODE:

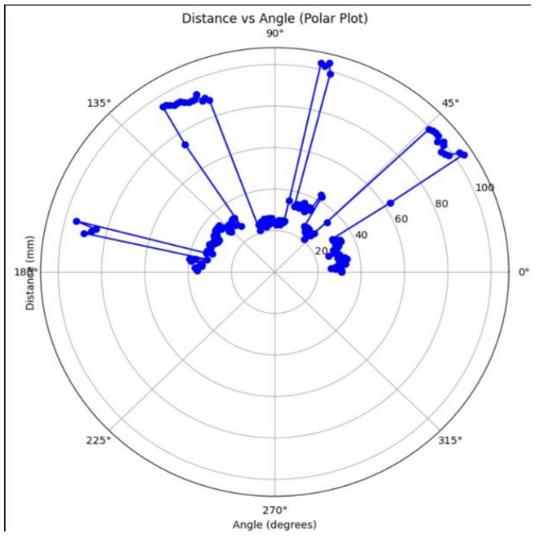
```
import serial
import time
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d.art3d import Poly3DCollection

# Configure the serial port (adjust 'COM3' and baud rate as needed)
ser = serial.Serial('COM5', 9600, timeout=1)
time.sleep(2)  # Wait for the serial connection to initialize

# Initialize lists to hold data
angles = []
distances = []
accel_x = []
accel_y = []
accel_z = []
gyro_x = []
gyro_y = []
gyro_z = []

def parse_value(line, label):
    """Extracts and returns the integer value from a line based on the label."""
    try:
        value_str = line.split(label + ":")[1].strip()
        return int(value_str.split()[0])  # Take only the first part in case of extra text
    except (ValueError, IndexError) as e:
        print(f"Error parsing {label}: {line} - {e}")
        return None

try:
    print("Starting data collection...")
    while True:
        line = ser.readline().decode('utf-8').strip()  # Read the line from Arduino
        print(line)  # Debug statement to check the output from Arduino

        if "Angle:" in line:
            angle = parse_value(line, "Angle")
            if angle is not None:
                angles.append(angle)

        if "Distance:" in line:
            distance = parse_value(line, "Distance")
            if distance is not None:
                distances.append(distance)

        if "Accel X:" in line:
            ax = parse_value(line, "Accel X")
            if ax is not None:
                accel_x.append(ax)

        if "Accel Y:" in line:
            ay = parse_value(line, "Accel Y")
            if ay is not None:
                accel_y.append(ay)

        if "Accel Z:" in line:
            az = parse_value(line, "Accel Z")
            if az is not None:
                accel_z.append(az)

        if "Gyro X:" in line:
            gx = parse_value(line, "Gyro X")
            if gx is not None:
                gyro_x.append(gx)

        if "Gyro Y:" in line:
            gy = parse_value(line, "Gyro Y")
            if gy is not None:
                gyro_y.append(gy)

        if "Gyro Z:" in line:
            gz = parse_value(line, "Gyro Z")
            if gz is not None:
                gyro_z.append(gz)

        # Break the loop after collecting data for 180 angles
        if len(angles) >= 180:
            break

finally:
    ser.close()  # Close the serial port

# Check if any data was collected
if not angles or not distances:
    print("No valid data collected.")
else:
    # Polar Plotting of Distance vs Angle
    plt.figure(figsize=(10, 8))
    ax = plt.subplot(111, projection='polar')
    angles_rad = np.radians(angles)
    distances_np = np.array(distances)
    ax.plot(angles_rad, distances_np, marker='o', linestyle='-', color='b')
    ax.set_title("Distance vs Angle (Polar Plot)")
    ax.set_xlabel("Angle (degrees)")
    ax.set_ylabel("Distance (mm)")
    plt.show()
```

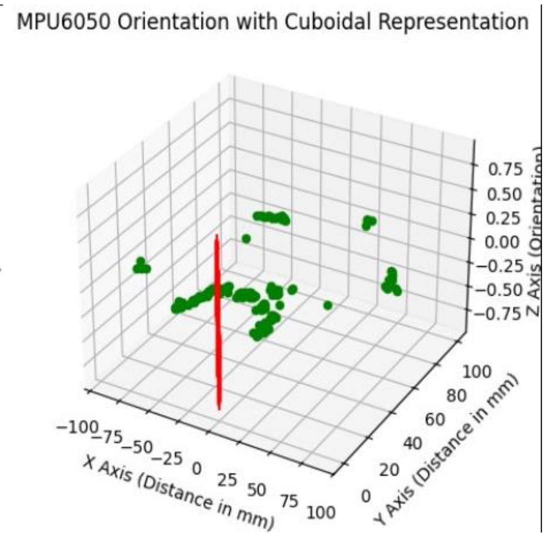
```
# 3D Visualization of MPU6050 Orientation with a Cuboidal Box
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Define the cuboid dimensions
cuboid_dimensions = np.array([[0.5, 0.5, 0.5],
                    [-0.5, 0.5, 0.5],
                    [-0.5, -0.5, 0.5],
                    [0.5, -0.5, 0.5],
                    [0.5, 0.5, -0.5],
                    [-0.5, 0.5, -0.5],
                    [-0.5, -0.5, -0.5],
                    [0.5, -0.5, -0.5]])

for i in range(len(angles)):
    # Calculate the orientation of the cuboid based on MPU6050 data
    # Assuming ax, ay, az are accelerations (for cuboid orientation)
    # Using mean values for demonstration; you can customize as needed
    roll = np.radians(gyro_x[i])  # Convert to radians
    pitch = np.radians(gyro_y[i])
    yaw = np.radians(gyro_z[i])

    # Rotation matrices for roll, pitch, yaw
    R_x = np.array([[1, 0, 0],
            [0, np.cos(roll), -np.sin(roll)],
            [0, np.sin(roll), np.cos(roll)]])

    R_y = np.array([[np.cos(pitch), 0, np.sin(pitch)],
            [0, 1, 0],
            [-np.sin(pitch), 0, np.cos(pitch)]])

    R_z = np.array([[np.cos(yaw), -np.sin(yaw), 0],
            [np.sin(yaw), np.cos(yaw), 0],
            [0, 0, 1]])
    # Combined rotation matrix
    R = np.dot(R_z, np.dot(R_y, R_x))
    transformed_cuboid = np.dot(cuboid_dimensions, R.T)

    # Translate the cuboid to the position based on distance measurement
    ax.scatter(distances[i] * np.cos(angles_rad[i]),
               distances[i] * np.sin(angles_rad[i]),
               0, color='g')  # Position of IMU

    # Plot the cuboid
    ax.add_collection3d(Poly3DCollection([transformed_cuboid[[0, 1, 2, 3]],
                                          transformed_cuboid[[4, 5, 6, 7]],
                                          transformed_cuboid[[0, 1, 5, 4]],
                                          transformed_cuboid[[2, 3, 7, 6]],
                                          transformed_cuboid[[1, 2, 6, 5]],
                                          transformed_cuboid[[4, 7, 3, 0]]],
                                         facecolors='cyan', linewidths=1, edgecolors='r', alpha=.25))

ax.set_xlabel('X Axis (Distance in mm)')
ax.set_ylabel('Y Axis (Distance in mm)')
ax.set_zlabel('Z Axis (Orientation)')
ax.set_title('MPU6050 Orientation with Cuboidal Representation')
plt.show()
```

1.2 PHOTO OF THE PROJECT









CONCLUSION:

The LiDAR-based monitoring system represents a revolutionary advancement in fleet management and vehicle safety. By leveraging cutting-edge technology, this system provides an efficient solution for real-time obstacle detection and navigation, addressing critical needs in various industries, including mining, construction, and logistics.

Working Principle:

The system operates using a combination of LiDAR sensors, a servo motor, and data processing algorithms. The LiDAR sensor emits laser beams that measure the distance to surrounding objects by calculating the time it takes for the light to return. Mounted on a servo motor, the sensor can rotate, allowing it to scan a wide area and gather comprehensive data about the environment. This data is processed in real time to detect obstacles, assess terrain conditions, and provide actionable insights for vehicle operators. The information is then transmitted to a control unit, which assists in making navigation decisions and enhances overall safety.

Conclusion:

In summary, the LiDAR-based monitoring system offers significant benefits that enhance both operational efficiency and safety in fleet management. Its real-time detection capabilities are crucial for preventing collisions and accidents, particularly in hazardous environments like mining sites. By enabling proactive planning and better route management, the system minimizes unexpected delays and enhances productivity.

Cost savings are another critical advantage, as safer navigation leads to reduced equipment wear and tear, ultimately lowering repair costs and extending the lifespan of vehicles and machinery. Furthermore, the technology contributes to lower environmental impacts by optimizing routes, reducing fuel consumption and emissions, thereby promoting sustainability.

The safety of workers is enhanced through real-time hazard identification, creating a safer working environment. Companies utilizing advanced safety measures can build a positive reputation, leading to increased customer trust and loyalty. Additionally, the long-term return on investment (ROI) becomes apparent as organizations benefit from fewer accidents, reduced operational costs, and improved safety metrics.

Applications:

The applications of this monitoring system are vast, spanning multiple sectors such as construction, mining, agriculture, and logistics. It can be utilized in autonomous vehicles for navigation, in heavy machinery for terrain monitoring, and in fleet management for tracking vehicle movements. Integrating the system with existing management software can provide comprehensive data analysis, facilitating informed decision-making.

Technical Aspects:

From a technical standpoint, the system utilizes a servo motor to control the orientation of the LiDAR sensor, allowing it to scan effectively. However, this mechanism presents mechanical constraints, limiting the LiDAR's scanning capabilities to a 180-degree field of view. Additionally, the system employs the VL53L0X laser sensor, which provides distance measurements within a range of 30 cm to 2 meters. While effective, this sensor has inherent blind spots that may result in missed obstacles either too close or beyond its maximum detection range.

Disadvantages:

1. Mechanical Constraints of the Servo Motor: The reliance on a servo motor constrains the LiDAR's scanning capability to a 180-degree field of view, potentially limiting the detection of obstacles outside this range.
2. Blind Spots of the VL53L0X Laser Sensor: The VL53L0X sensor operates effectively within a range of 30 cm to 2 meters, but its blind spots may lead to undetected obstacles, especially those situated too close to the sensor or beyond its operational range.

Despite these limitations, the advantages of the LiDAR-based monitoring system far outweigh its disadvantages, making it a vital investment for enhancing safety, efficiency, and sustainability in fleet management and vehicle operations.

REFERNCES

- https://www.instructables.com/Project-Lighthouse-360-Mini-Arduino-LiDAR/
- https://dzone.com/articles/make-your-own-lidar-sensor
- https://en.wikipedia.org/wiki/Lidar#:~:text=Lidar%20sensors%20mounted%20on%20mobile,inertial%20measurement%20unit%20(IMU).