Sensor Data Integrity Verification for Autonomous Vehicles with Infrared and Ultrasonic Sensors

Sunday Oyijewu Ochedi

A20503423

10/18/2023

Outline

- Introduction
- Project Overview
- Literature Review
- Methodology
- Expected Results
- Timeline
- Resources and Budget
- Conclusion

Introduction

- Autonomous vehicles, integrate AI, sensors, and cameras, for navigation and driving without human intervention.

- With the growing reliance on AVs, securing them through safeguarding sensors and developing resilient algorithms is increasingly vital to prevent accidents from attacks.

- One way of doing this is by verifying the integrity of data collected from sensors.

Project Overview

Importance of Data Integrity

Data integrity is essential for accurate, trustworthy data, critical in decision-making, compliance, trust, error prevention, efficiency, research, security, and long-term reliability

Brief Overview of the Project (Scope)

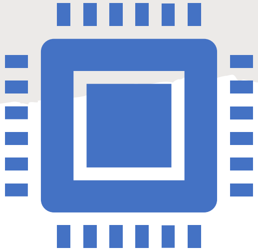
Integration of sensors with autonomous vehicle

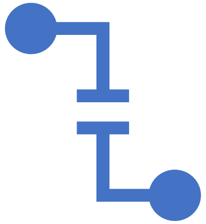
Develop mechanisms for data integrity verification

Focus on infrared and ultrasonic sensors

Potential integration of camera and LiDAR sensors

Literature Review





Challenges:

Noise and Interference: Environmental interference can distort sensor data.

Sensor Calibration: Ensuring precise sensor calibration is time-consuming and challenging.

Sensor Degradation: Sensors degrade over time, affecting accuracy.

Multi-Sensor Fusion: Integrating multiple sensors can be complex and error-prone.

Data Anomalies: Detecting and managing sensor anomalies is an ongoing challenge.

Solutions:

Redundancy: Employing multiple sensors to verify data and enhance reliability.

Sensor Fusion: Combining data from diverse sensors for accuracy and redundancy.

Error Correction Algorithms: Using algorithms for real-time sensor error detection and correction.

Sensor Calibration: Regularly calibrating sensors to maintain accuracy.

Methodology

Autonomous Vehicle:
Integration of sensors with autonomous vehicle

Data Collection:
Gather data from infrared and ultrasonic sensors.

Data Validation Algorithms:
Develop algorithms for anomaly detection.

Real-time Data Integrity Checks

Potential Sensor Integration:
Explore the integration of camera and LiDAR sensors (if time allows)

Expected Results

IMPROVED DATA RELIABILITY

ANOMALY DETECTION

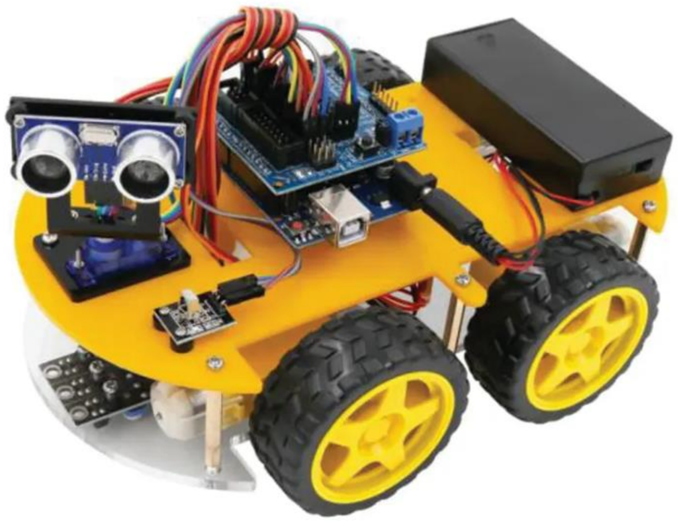
POTENTIAL FOR CAMERA AND LIDAR INTEGRATION (IF TIME PERMITS)

Timeline

Task	Start Date	End Date
Autonomous vehicle	10/19/2023	10/30/2023
Data Collection	10/30/2023	11/9/2023
Algorithm Development	11/9/2023	11/16/2023
Testing and Validation	11/16/2023	11/22/2023
Integration	11/22/2023	11/28/2023

Resources and Budget

- Sensors: $25
- Software and Hardware: $65
- Time



Conclusion

- Expected Impact of the project:
  - Safer and more reliable autonomous driving.
  - Reduced risk of accidents due to sensor errors.
  - Enhance trust and acceptance of autonomous technology.

Thank You

Thank you for your time and attention

Sensor Data Integrity Verification for Autonomous Vehicles with Infrared and Ultrasonic Sensors

Presentation 2: Progress report

Sunday Oyijewu Ochedi

A20503423

11/8/2023

Outline

- Project recap
- Project timeline
- Autonomous vehicle setup
- Data collection
- Next steps
- Conclusion

Project timeline

Task partially completed

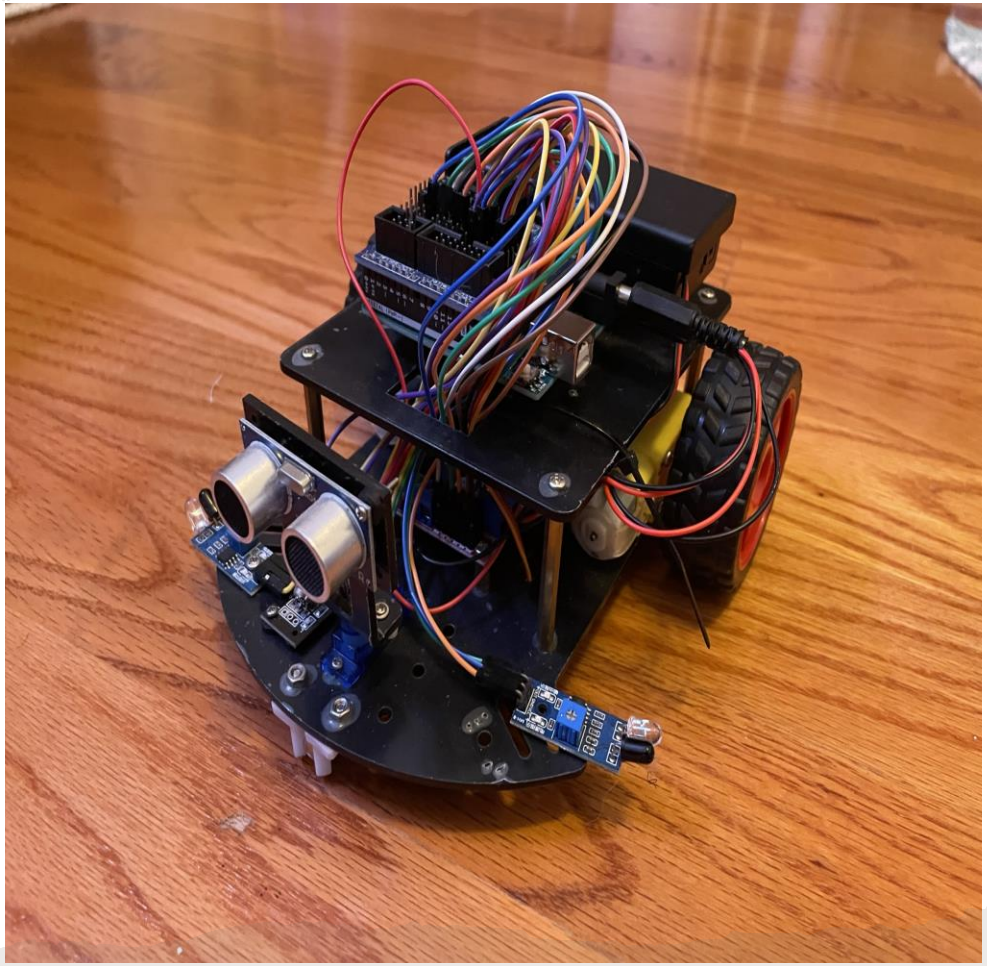
Task	Start Date	End Date
Autonomous vehicle	10/19/2023	10/30/2023
Data Collection	10/30/2023	11/9/2023

Task to be completed

Algorithm Development	11/9/2023	11/16/2023
Testing and Validation	11/16/2023	11/22/2023
Integration	11/22/2023	11/28/2023

Autonomous vehicle – initial setup

- The initial setup had an ultrasonic sensor and two infrared sensors.
- This setup was used to program the robot using only the ultrasonic sensor initially.
- The potential field algorithm is used to control robot movement.



Autonomous vehicle – potential field algorithm

- The potential field algorithm is like a magnet that helps a robot move. It pulls the robot toward its destination and pushes it away from things it should avoid. This way, the robot can find its way without bumping into obstacles.

Potential field algorithm

- Attractive Force:

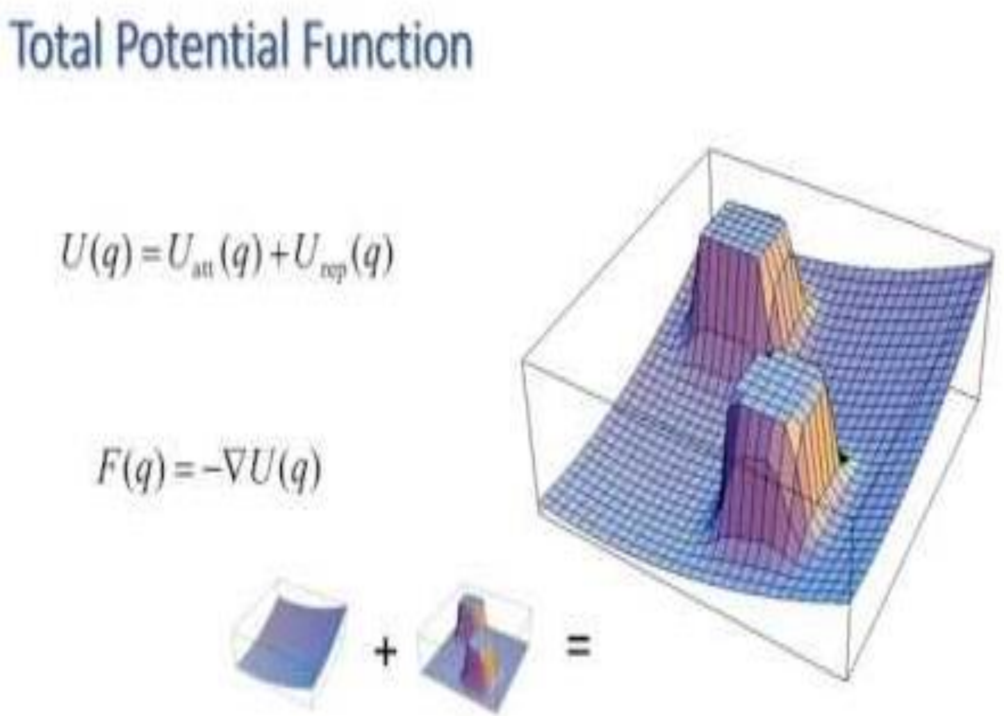
  U_attractive = k_att * (goal_position - current_position)

- Repulsive Force (for each obstacle):

  U_repulsive = k_rep / D

  D= distance from obstacle

- The robot moves based on a combination of attraction to the goal and repulsion from obstacles.



Autonomous vehicle – potential field algorithm

- Arduino code to control robot movement using potential field algorithm.

Attractive force

```
// Calculate attractive force
void attractiveforce()
{
  if (d_qq_goal <= d_goalstar && d_qq_goal >= 0.5)
  {
    Fa = (1.0 / 2.0) * Ka * (pow((d_qq_goal),2));
  }
  else
  {
    Fa = (d_goalstar*Ka*d_qq_goal) - ((1.0 / 2.0) * Ka*pow(d_goalstar,2)).
  }
}
// Calculate attractive force gradient
void attractiveforcegradient() {
  if (d_qq_goal <= d_goalstar && d_qq_goal >= 0.5)
  {
     Faxgrad = -Ka * (x_goal - x);
     Faygrad = -Ka * (y_goal - y);
  }
  else
  {
    Faxgrad = -(d_goalstar*Ka * (x_goal - x))/d_qq_goal;
    Faygrad = -(d_goalstar*Ka * (y_goal - y))/d_qq_goal;
  }
}
```

```
// Calculate repulsive force
void repulsiveforce() {
  if (d_sensormiddle <= r_safe)
  {
  Fr = (1.0 / 2.0) * Kr * (pow((1/d_sensormiddle) - (1/r_safe), 2));
  }
  else
  {
    Fr=0;
  }
}

// Calculate repulsive force gradient
void repulsiveforcegradient() {
  if (d_sensormiddle <= r_safe)
  {
    float r_diff = (1.0 / d_sensormiddle) - (1.0 / r_safe);
    Frxgrad = (Kr * r_diff / (d_sensormiddle * d_sensormiddle)) * (x - (d_sensormiddle*cos(w)));
    Frygrad = (Kr * r_diff / (d_sensormiddle * d_sensormiddle)) * (y - (d_sensormiddle*sin(w)));
  }
  else
  {
    Frxgrad = 0;
    Frygrad = 0;
  }
}
```

repulsive force

Total force

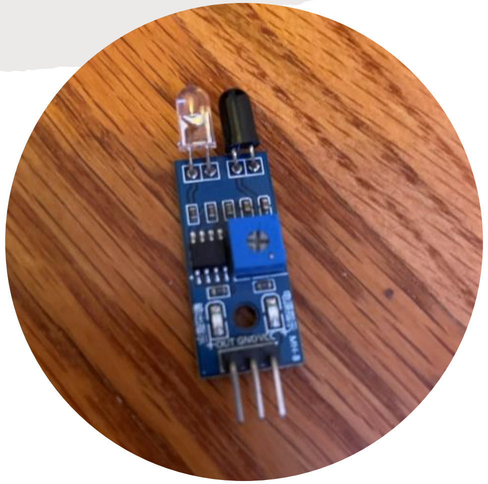
```
// Calculate the total force components
void totalforcegradient()
{
  Fx = Faxgrad + Frxgrad;
  Fy = Faygrad + Frygrad;
}
 // Update the robot's position based on the forces (you may need to integrate this)
 void robotpositionupdate()
 {
 x += -1*Fx * dt; // dt is the time step
 y += -1*Fy * dt;
 }


// Function to calculate angular velocity w
void calculateAngularVelocity() {
 // Calculate angle between robot orientation and attractive force direction
 theta_robot = atan2(Faygrad, Faxgrad);

 // Calculate angle between robot orientation and obstacle
 theta_obstacle = atan2(Frygrad, Frxgrad);

 // Calculate angle between robot orientation and goal direction
 theta_goal = atan2(Fy, Fx);

 // Calculate angle difference
 if (d_sensormiddle <= r_safe)
 {
 //theta_diff = theta_goal - theta_robot;
 theta_diff = -1*(theta_goal - theta_obstacle);
 }
 else
 {
 theta_diff = theta_goal - theta_robot;
 }

 //Ensure theta_diff is within the range of -pi to pi
 if (theta_diff > M_PI) {
   theta_diff -= 2 * M_PI;
```

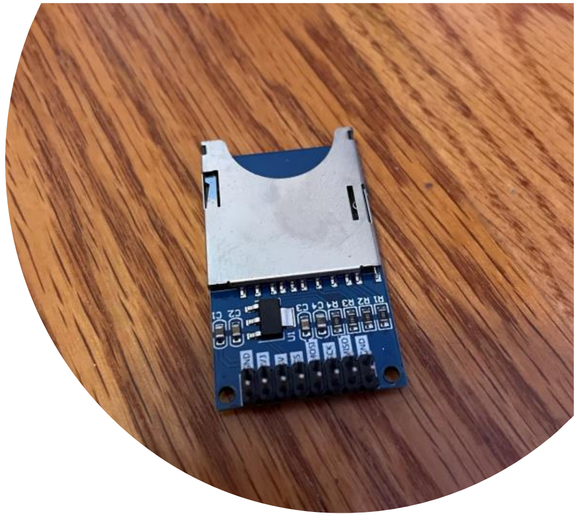
Autonomous vehicle – initial test run

- Initial test of robot movement using potential field algorithm

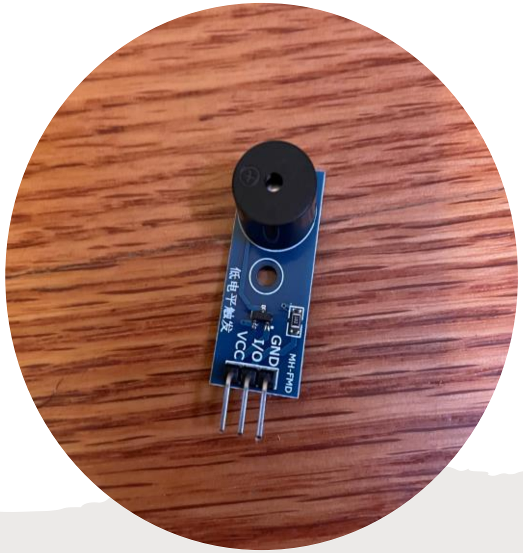
- Problem:
  Due to the lower value of K_att compared to K_rep, the robot gives higher priority to avoiding obstacles over reaching its ultimate destination.
- Solution:
  Fine-tuning these values is necessary to achieve optimal functionality.

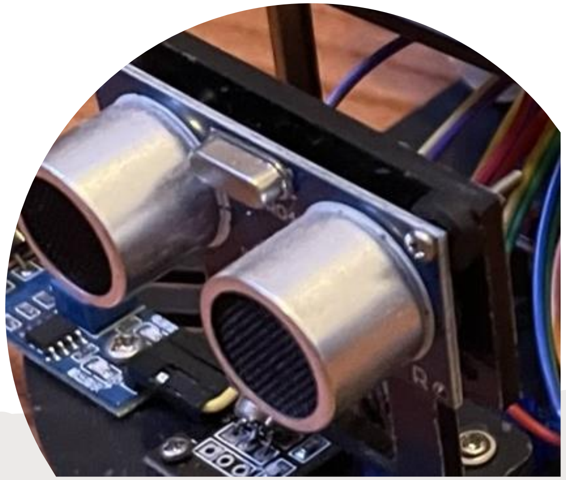
Autonomous vehicle – new setup





- For the new setup, I used 3 ultrasonic sensors, 4 infrared sensors, an alarm, and an SD card module.





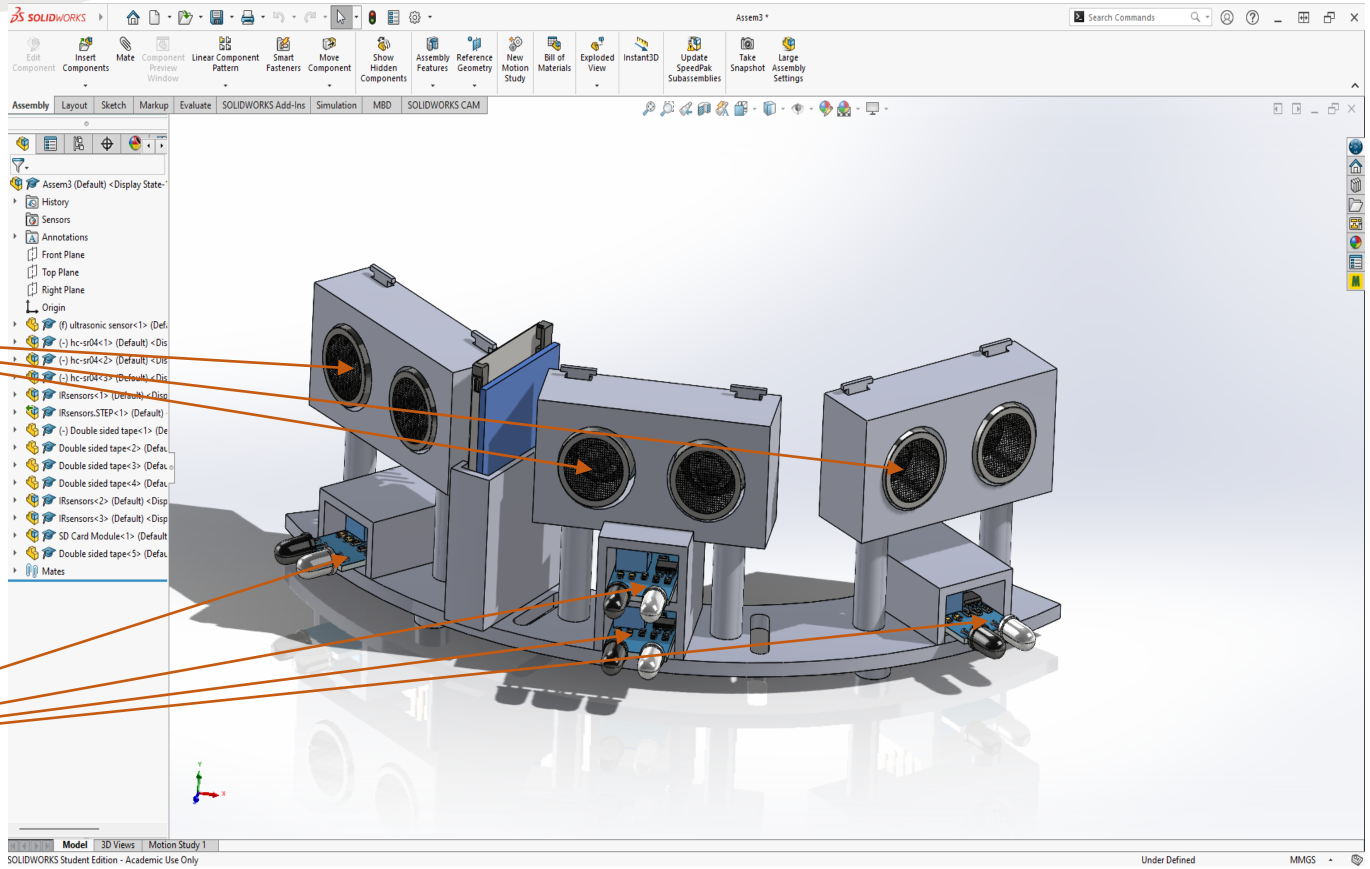
- The 3 ultrasonic and 4 infrared sensors are used to sense obstacles.
- The SD card reader is used to store data for analysis.
- The alarm will be used to alert users when there is an anomaly.

Autonomous vehicle – ultrasonic and infrared sensor setup

- I designed a platform using Solidworks to securely mount the ultrasonic and infrared sensors onto the robot.

Ultrasonic sensors

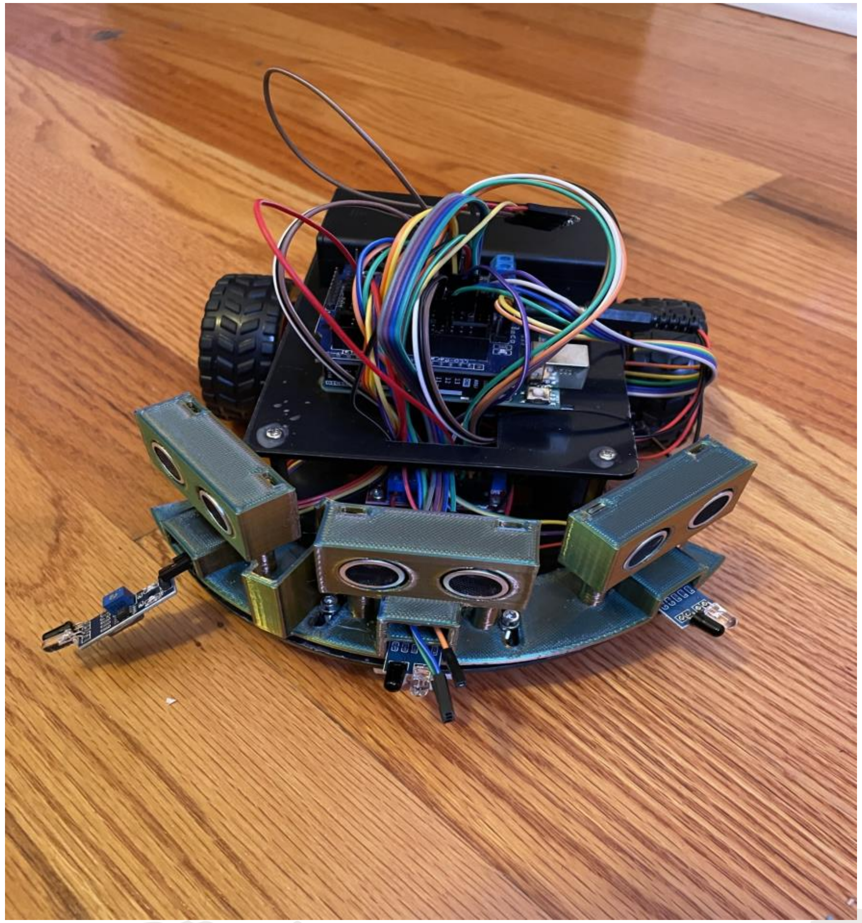
Infrared sensors



Autonomous vehicle – ultrasonic and infrared sensor setup

- The 3D-printed platform securely attaches the ultrasonic and infrared sensors to the mobile robot.

- The left, right, and bottom middle infrared sensors perform flawlessly, whereas I plan to intentionally calibrate the top middle infrared sensors to function inadequately. This calibration will serve as a test to observe the effects when data from the ultrasonic and infrared sensors do not align (anomaly check).



Data collection – SD card module setup

- The SD card module is positioned at the back of the mobile robot and linked to the Arduino Uno to store data while the robot is in operation.

Data collection – Writing data to SD card

- Arduino code to store data on an SD card for later analysis.

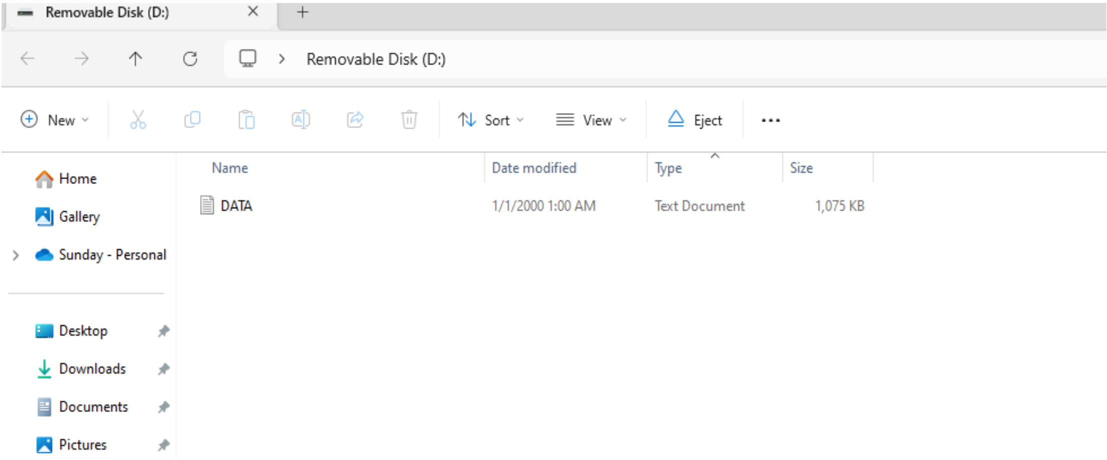
```
void setup() {
  // Initialize serial communication for debugging
  Serial.begin(9600);


  // Set pin modes
  // SD card
  pinMode(chipSelect, OUTPUT);
  // Initialize SD card
 Serial.print("Initializing SD card...");
  if (!SD.begin(chipSelect)) {
    Serial.println("SD card initialization failed.");
    return;
  }
  Serial.println("SD card initialized.");

```

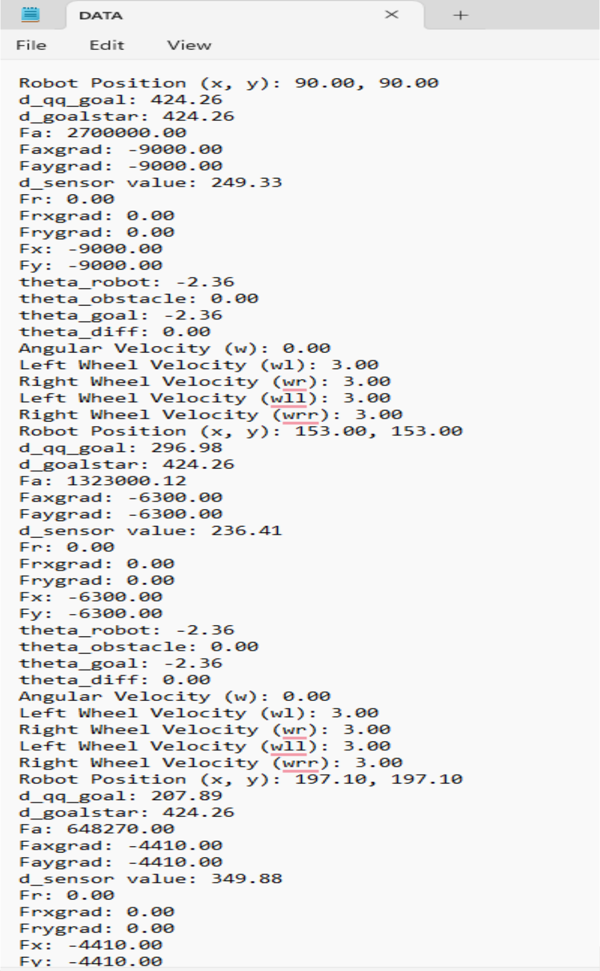
```
  // Open a file on the SD card for writing
  dataFile = SD.open("data.txt", FILE_WRITE);

  // Check if the file opened successfully
  if (dataFile) {
    Serial.println("File opened for writing.");
  } else {
    Serial.println("Error opening data.txt for writing.");
  }
```

```

// Write data to the file on the SD card
dataFile.print("Robot Position (x, y): ");
dataFile.print(x);
dataFile.print(", ");
dataFile.println(y);
dataFile.print("d_qq_goal: ") ;
dataFile.println(d_qq_goal);
dataFile.print("d_goalstar: ") ;
dataFile.println(d_goalstar);
dataFile.print("Fa: ") ;
dataFile.println(Fa);
dataFile.print("Faxgrad: ") ;
dataFile.println(Faxgrad);
dataFile.print("Faygrad: ") ;
dataFile.println(Faygrad);
dataFile.print("d_sensormiddle value: ");
dataFile.println(d_sensormiddle);
dataFile.print("d_sensorleft value: ");
dataFile.println(d_sensorleft);
dataFile.print("d_sensorright value: ");
dataFile.println(d_sensorright);
dataFile.print("left_light_value = ");
dataFile.println(leftinfraredSensor);
dataFile.print("right_light_value = ");
dataFile.println(rightinfraredSensor);
dataFile.print("middle_light_value = ");
dataFile.println(middleinfraredSensor);
dataFile.print("Fr: ") ;
dataFile.println(Fr);
dataFile.print("Frxgrad: ") ;
dataFile.println(Frxgrad);
dataFile.print("Frygrad: ") ;
dataFile.println(Frygrad);
dataFile.print("Fx: ") ;
```

Data collection – saved data

- Data saved to SD card for future analysis



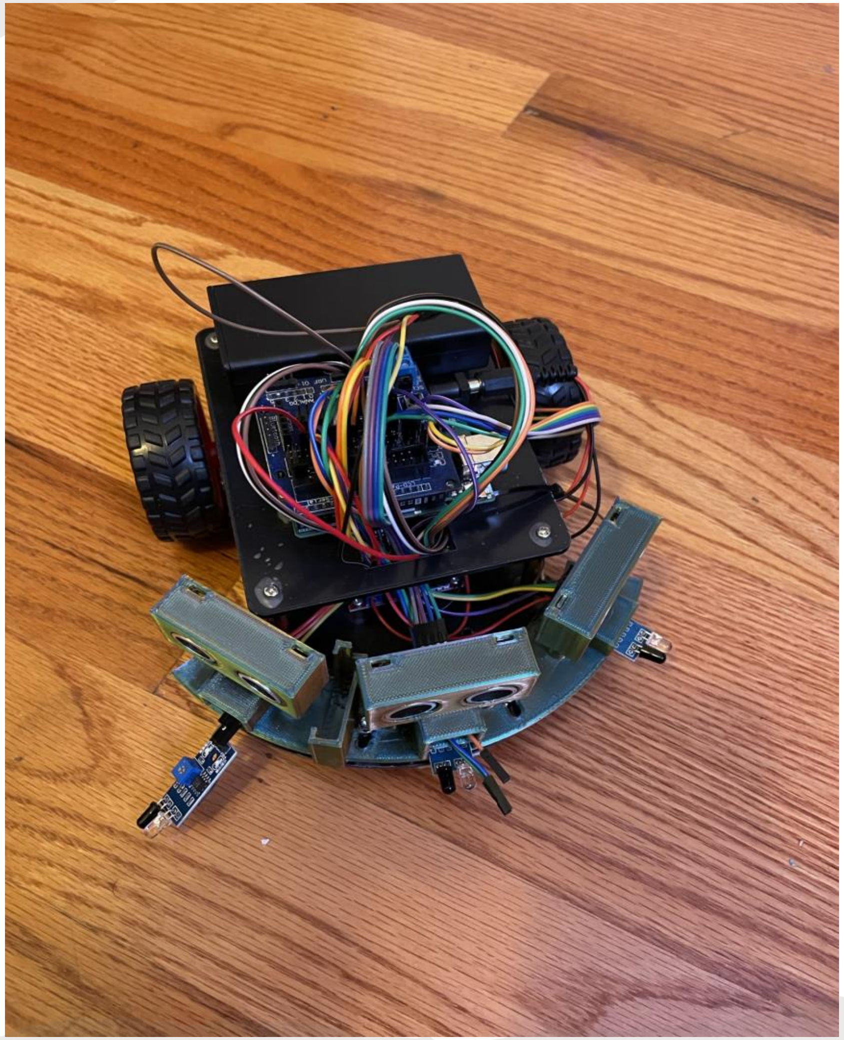


Next steps

- Modify the Arduino code to incorporate all the onboard sensors, including the three ultrasonic sensors, four infrared sensors, and the alarm, for the robot's motion control.

- Develop an Anomaly detection algorithm.

- Test run.



Conclusion

The mobile robot is functioning adequately, but I need to fine-tune the code for optimal performance before proceeding to develop the anomaly detection algorithm in the next stage.

Thank You

Thank you for your time and attention

Sensor Data Integrity Verification for Autonomous Vehicles with Infrared and Ultrasonic Sensors

Presentation 3: Final update report

Sunday Oyijewu Ochedi
A20503423
11/29/2023

Outline

- Project recap
- Project timeline
- Autonomous vehicle setup
    - Data integrity verification
    - Control algorithm
    - Anomaly detection
- Data collection
- Testing and validation
- Conclusion

Project recap - methodology

Autonomous Vehicle:

Integration of sensors with autonomous vehicle

→

Data Collection:

Gather data from infrared and ultrasonic sensors.

→

Data Validation Algorithms:

Develop algorithms for anomaly detection.

→

Real-time Data Integrity Checks

Project recap

Initial setup (First presentation):

- One ultrasonic and two infrared sensors.
- Use potential field algorithm for movement.
- Problem: Fine-tuning the potential field algorithm takes up a lot of time and consumes the available memory of the Arduino uno.

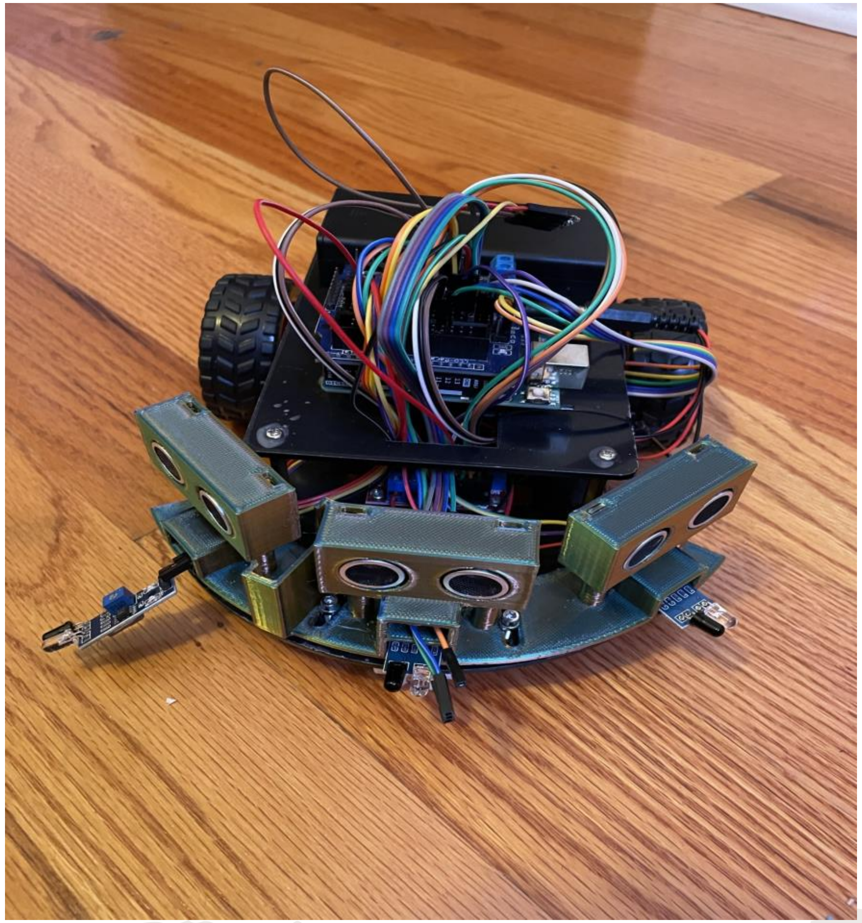
Second setup (Second presentation):

- 3 ultrasonic sensors, 4 infrared sensors, an alarm, and an SD card module.
- Use potential field algorithm for movement.
- Problem: Fine-tuning the potential field algorithm takes up a lot of time and consumes the available memory of the Arduino uno.

Timeline

Task	Start Date	End Date
Autonomous vehicle	10/19/2023	10/30/2023
Data Collection	10/30/2023	11/9/2023
Algorithm Development	11/9/2023	11/16/2023
Testing and Validation	11/16/2023	11/22/2023
Integration	11/22/2023	11/28/2023

Autonomous vehicle – final setup and changes made



- 3 ultrasonic sensors, 3 infrared sensors, an alarm, and an SD card module.

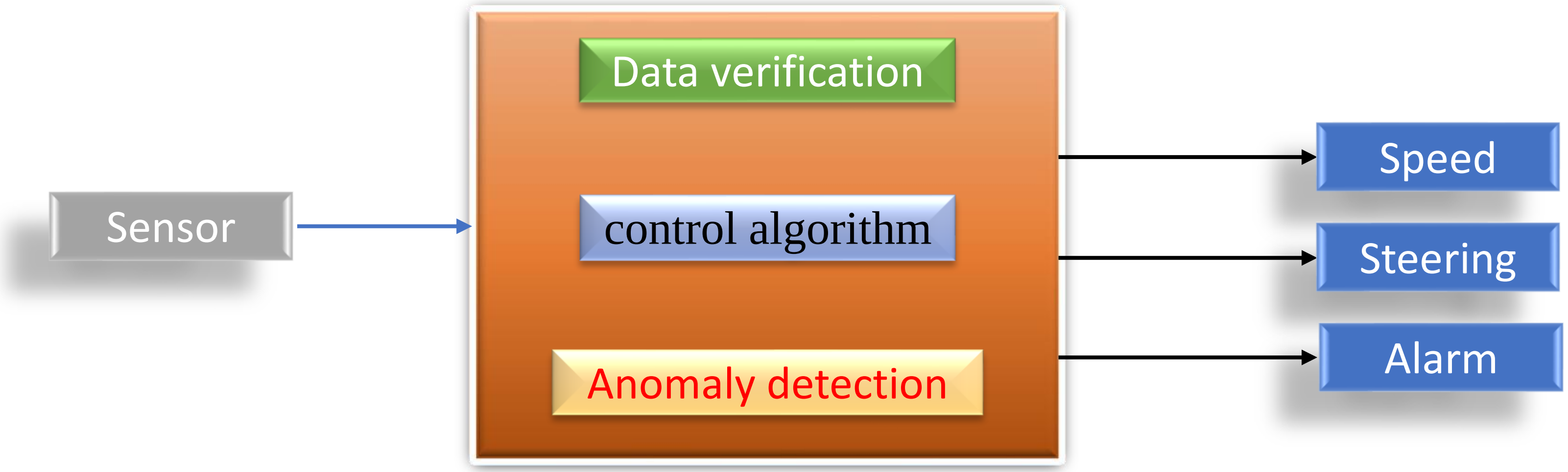
Changes made:

- One infrared sensor was removed leaving one sensor each on the left, right, and middle of the robot.

- Because of the complexity of using a potential field algorithm to control the robot, a basic control algorithm was developed.

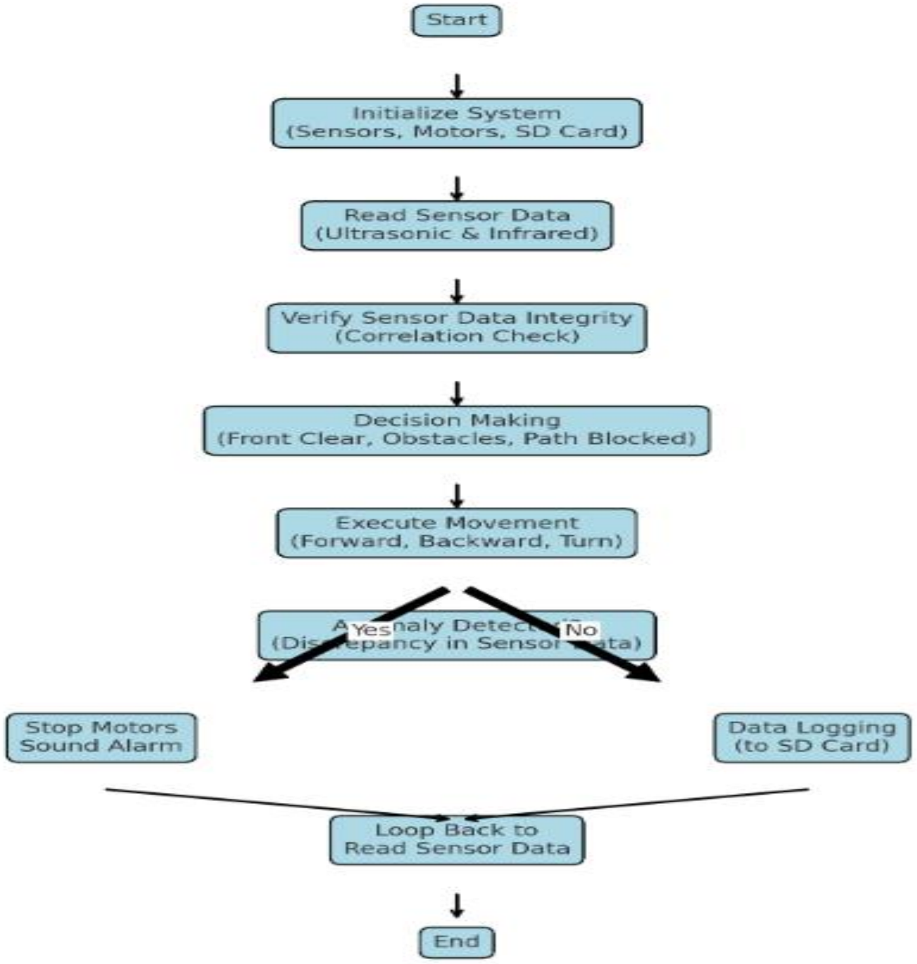
Autonomous vehicle – robot control

A simple control algorithm is used to control the robot's behavior in an environment.



Control unit

Autonomous vehicle – control unit flow chart diagram



Autonomous vehicle - Data integrity verification

Validating ultrasonic sensor data:
A validation check on the data received from the ultrasonic sensor is implemented to guard against erroneous readings of zero or infinity, which can occur in the absence of a detectable object.

```
• float getDistance(int echoPin) {
•   digitalWrite(triggerPin, LOW);
•   delayMicroseconds(2);
•   digitalWrite(triggerPin, HIGH);
•   delayMicroseconds(10);
•   digitalWrite(triggerPin, LOW);
•   float duration = pulseIn(echoPin, HIGH);
•   float distance = duration * 0.034 / 2; // Speed
  of sound at 20°C is approximately 343 m/s
•     if(distance == 0 || distance >  safeDistance) {
•     Serial.println("Sensor: Invalid distance");
•      d_sensor = 200; // Default or safe value
•   } else {
•      d_sensor= distance;
•   }
•   return d_sensor;
• }
```

Autonomous vehicle - Data integrity verification

- Opens the SD card file for data logging.
- Measures distances using ultrasonic sensors.
- Reads infrared sensor values.
- The system checks if the readings from the ultrasonic sensors correlate with the infrared sensors.
- Move forward if no obstacles are detected within the safe distance.

```
void loop(){
  SDcardopenfile(); // opens SDcard to store data
  distanceFront = getDistance(echoPinFront);
  distanceLeft  =  getDistance(echoPinLeft);
  distanceRight = getDistance(echoPinRight);
  infraredsensor();
  if (distanceFront > safeDistance) {
    // Check side distances when front is clear
    if (distanceLeft <= safeDistance && leftSensorvalue == 0 ) {
      // Turn right slightly if an obstacle is close on the left
      moveForward();
      turnRightslightly();
      Serial.println(" Turn right slightly ");
      dataFile.println(" Turn right slightly ");
}else if (distanceRight <= safeDistance && rightSensorvalue == 0) {
      // Turn left slightly if an obstacle is close on the right
      moveForward();
      turnLeftslightly();
      Serial.println(" Turn left slightly ");
      dataFile.println(" Turn left slightly ");
```

Autonomous vehicle – control algorithm

Decision-making logic based on sensor readings:

- ➢ Move forward: If no obstacles are detected within the safe distance.
- ➢ Turn slightly: Adjusts direction if an obstacle is detected on one side.
- ➢ Move backward: If blocked from all directions.
- ➢ Additional decision logic for turning or stopping based on sensor readings.

```
void loop(){
  SDcardopenfile(); // opens SDcard to store data
  distanceFront = getDistance(echoPinFront);
  distanceLeft  =  getDistance(echoPinLeft);
  distanceRight = getDistance(echoPinRight);
  infraredsensor();
  if (distanceFront > safeDistance) {
    // Check side distances when front is clear
    if (distanceLeft <= safeDistance && leftSensorvalue == 0 ) {
      // Turn right slightly if an obstacle is close on the left
      moveForward();
      turnRightslightly();
      Serial.println(" Turn right slightly ");
      dataFile.println(" Turn right slightly ");
    } if (distanceLeft <= criticaldistance && leftSensorvalue == 1 ) {
      //anomaly detected stop robot and sound alarm
      stopMotors();
      alarmtone();
      Serial.println(" Anomaly detected: left distance data not verified by left infraredsensor data ");
      dataFile.println(" Anomaly detected: left distance data not verified by left infraredsensor data ");
    }else if (distanceRight <= safeDistance && rightSensorvalue == 0) {
      // Turn left slightly if an obstacle is close on the right
      moveForward();
      turnLeftslightly();
      Serial.println(" Turn left slightly ");
      dataFile.println(" Turn left slightly ");
    } if (distanceRight <= criticaldistance && rightSensorvalue == 1 ) {
      //anomaly detected stop robot and sound alarm
      stopMotors();
      alarmtone();
      Serial.println(" Anomaly detected: Right distance data not verified by right infraredsensor data ");
      dataFile.println(" Anomaly detected: Right distance data not verified by right infraredsensorr data ");
    }else {
      // Move forward if both sides are safe
      moveForward();
      Serial.println(" Move forward ");
      dataFile.println(" Move forward ");
    }
```

Autonomous vehicle – anomaly algorithm

Anomaly Detection:

Stops the vehicle and sounds an alarm if there's a discrepancy between ultrasonic and infrared sensor readings, indicating a possible sensor failure or data integrity issue.

Where:
Criticaldistance: The minimum safe space needed for the robot to make a safe choice.
leftSensorvalue: left infrared sensor value.

```
if (distanceFront <= criticaldistance && leftSensorvalue == 1 ) {
    //anomaly detected stop robot and sound alarm
    stopMotors();
    alarmtone();
    Serial.println(" Anomaly detected: left distance data not verified by left infraredsensor data ");
    dataFile.println(" Anomaly detected: left distance data not verified by left infraredsensor data ");
  }
```

Data collection – Writing data to SD card

Arduino code to store data on an SD card for later analysis.

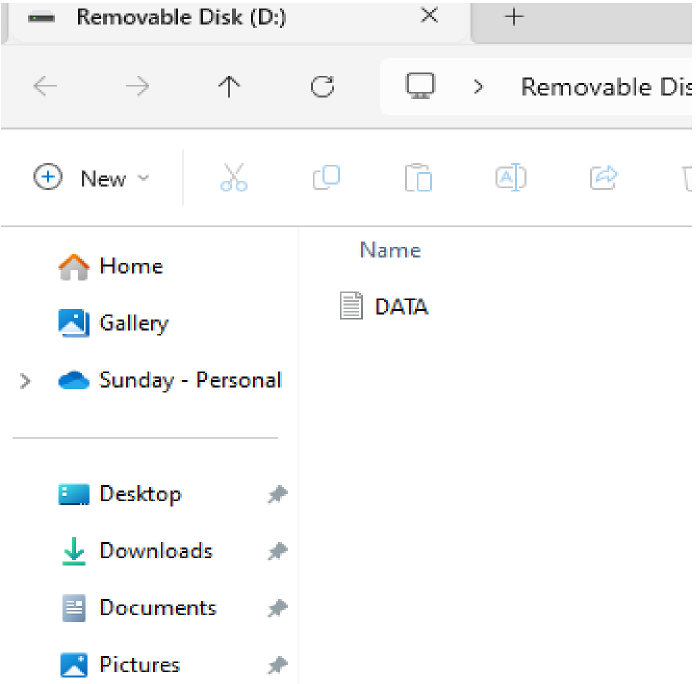
```
void setup() {
  // Initialize serial communication for debugging
  Serial.begin(9600);


  // Set pin modes
  // SD card
  pinMode(chipSelect, OUTPUT);
  // Initialize SD card
 Serial.print("Initializing SD card...");
  if (!SD.begin(chipSelect)) {
    Serial.println("SD card initialization failed.");
    return;
  }
  Serial.println("SD card initialized.");
```

```
    // Open a file on the SD card for writing
    dataFile = SD.open("data.txt", FILE_WRITE);

    // Check if the file opened successfully
    if (dataFile) {
      Serial.println("File opened for writing.");
    } else {
      Serial.println("Error opening data.txt for writing.");
    }
```

```
// Write data to the file on the SD card

dataFile.println("     ");
dataFile.print("Front distance value: ");
dataFile.print(distanceFront);
dataFile.print("|| Left distance value: ");
dataFile.print(distanceLeft);
dataFile.print("|| Right distance value: ");
dataFile.println(distanceRight);
dataFile.print("middle_light_value = ");
dataFile.print(middleSensorvalue);
dataFile.print("|| left_light_value = ");
dataFile.print(leftSensorvalue);
dataFile.print("|| right_light_value = ");
dataFile.print(rightSensorvalue);
dataFile.println("");
dataFile.println("");
dataFile.println("");
dataFile.close();
}
```

Data collection – saved data

Data saved to SD card for future analysis




```
 Stop motors
 Turn left

Front distance value: 33.37|| Left distance value: 200.00|| Right distance value: 12.05
middle_light_value = 1|| left_light_value = 1|| right_light_value = 0


 Stop motors
 Anomaly detected: Right distance data not verified by right infraredsensor data

Front distance value: 16.85|| Left distance value: 200.00|| Right distance value: 200.00
middle_light_value = 1|| left_light_value = 1|| right_light_value = 0


 Move forward

Front distance value: 200.00|| Left distance value: 200.00|| Right distance value: 200.00
middle_light_value = 1|| left_light_value = 1|| right_light_value = 0


 Move forward

Front distance value: 200.00|| Left distance value: 200.00|| Right distance value: 200.00
middle_light_value = 1|| left_light_value = 1|| right_light_value = 0
```

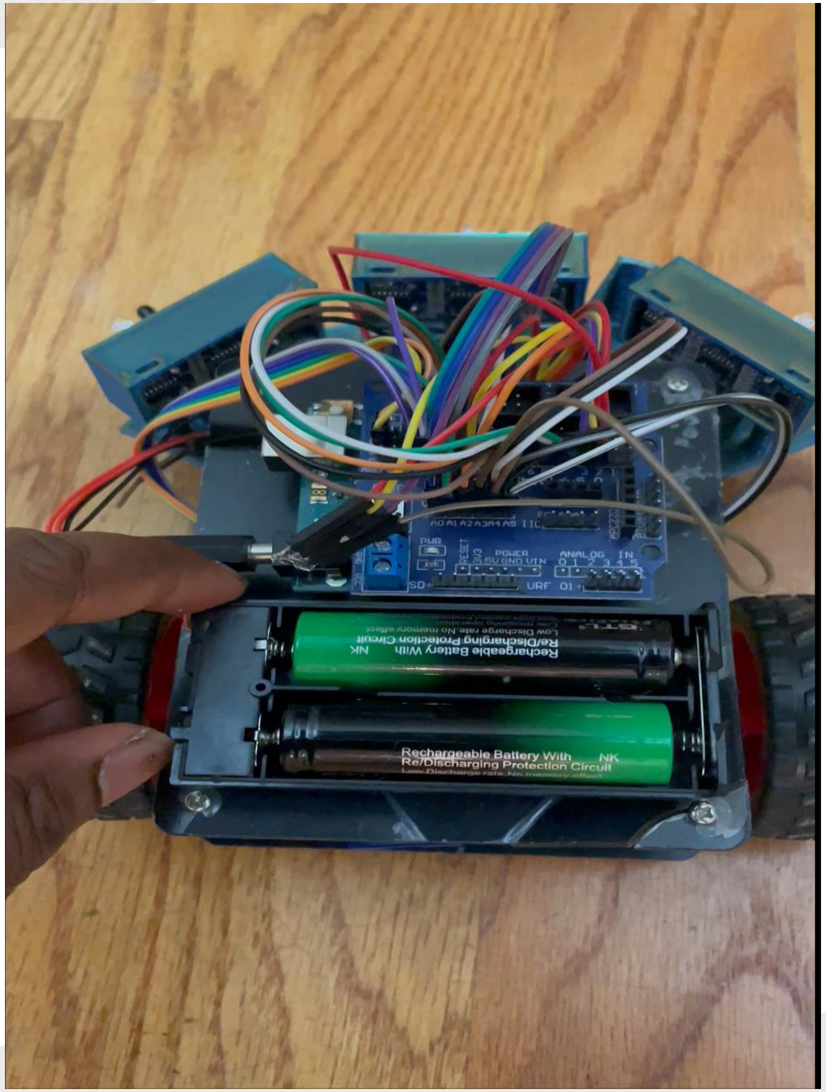
Testing and validation – original code

Criticaldistance = 11
safeDistance = 40

Where:
Criticaldistance: The minimum safe space needed for the robot to make a safe choice.

safeDistance: The sensor's threshold distance for the robot to start taking evasive action.

```
else {
   // If the front distance is not safe
   stopMotors();
   Serial.println(" Stop motors ");
   dataFile.println(" Stop motors ");
   if (distanceFront <= criticaldistance && leftSensorvalue == 1 ) {
     //anomaly detected stop robot and sound alarm
     stopMotors();
     alarmtone();
     Serial.println(" Anomaly detected: left distance data not verified by left infraredsensor data ");
     dataFile.println(" Anomaly detected: left distance data not verified by left infraredsensor data ");
   }
```

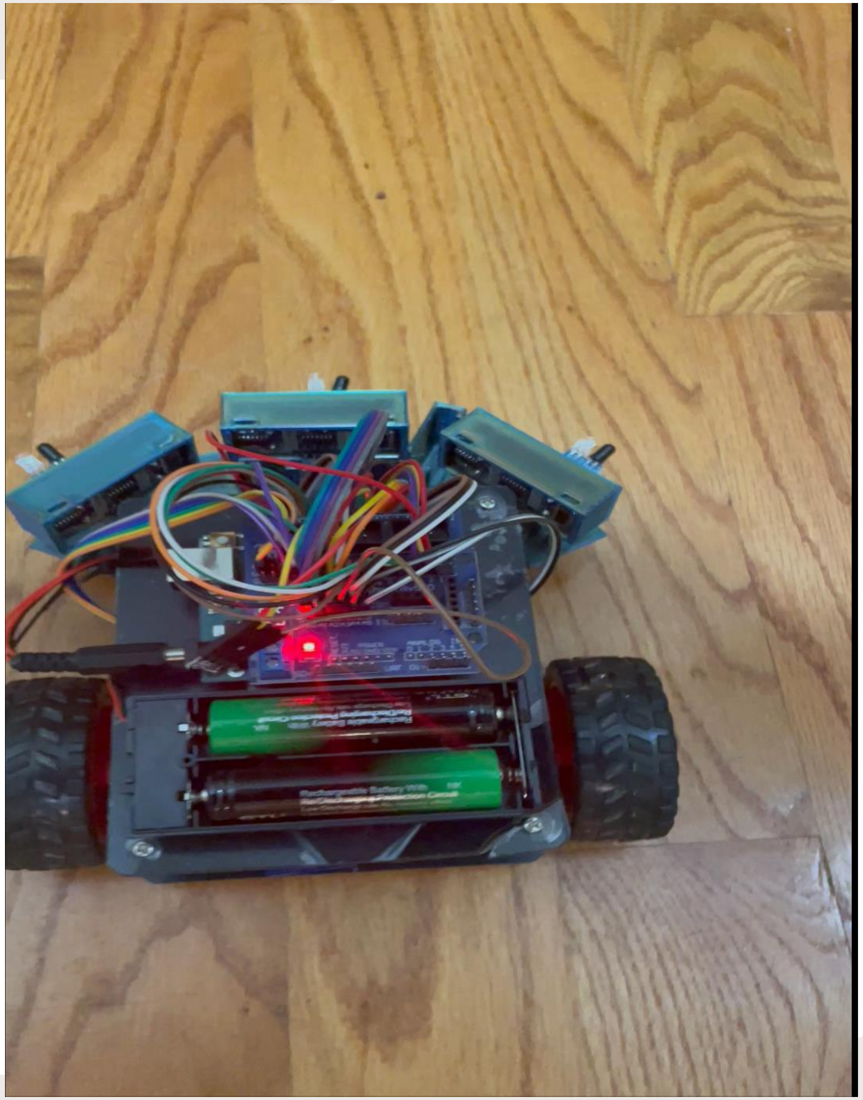


Testing and validation – failure test

Criticaldistance = 30
safeDistance = 40

Where:
Criticaldistance: The minimum safe space needed for the robot to make a safe choice.

safeDistance: The sensor's threshold distance for the robot to start taking evasive action.



Conclusion

This project is a comprehensive example of integrating multiple sensors for autonomous navigation with a focus on safety and data integrity.

It ensures that the vehicle operates safely by cross-verifying sensor data and taking precautionary actions in case of discrepancies, highlighting the importance of sensor data integrity in autonomous systems.

Thank You

Thank you for your time and attention