SRI DHARMASTHALA MANJUNATHESHWARA COLLEGE (AUTONOMOUS), UJIRE – 574240

DEPARTMENT OF B.VOC

IN

SOFTWARE AND APPLICATION DEVELOPMENT

2022-2023

PROJECT REPORT ON

"ARDUINO POWERED AUTONOMOUS VEHICLE"

SUBMITTED BY:

THIMMAIAH C V 201344

UNDER THE GUIDENCE OF

MS. YASHODA S PATIL

DEPARTMENT OF B.VOC

IN

SOFTWARE AND APP DEVELOPMENT

SHRI DHARMASTHALA MANJUNATHESHWARA COLLEGE

(AUTONOMOUS), UJIRE – 574240

DEPARTMENT OF B.Voc

IN

SOFTWARE AND APPLICATION DEVELOPMENT

2022-2023

PROJECT REPORT ON:
"ARDUINO-POWERED AUTONOMOUS VEHICLE"

SUBMITTED BY:

THIMMAIAH C V 201344

UNDER THE GUIDANCE OF

Ms. YASHODA.S. PATIL

DEPARTMENT OF B.Voc

IN

SOFTWARE AND APPLICATION DEVELOPMENT

SHRI DHARMASTHALA MANJUNATHESHWARA COLLEGE

(AUTONOMOUS), UJIRE - 574240

DEPARTMENT OF B.Voc

IN

SOFTWARE AND APPLICATION DEVELOPMENT

2022-2023

Certificate for the approval of project report

This is to certify that Thimmaiah C V, 201344 of 3rd B.Voc has satisfactorily completed the project on **"ARDUINO-POWERED AUTONOMOUS VEHICLE"** for the Bachelor of Vocational Course Prescribed by the college during academic year 2022-2023.

(Mr. Sammed Jain)	**(Ms. Yashoda.S.Patil)**	**(Dr B A Kumar Hegde)**
Head of the Department	**Project Guide**	**Principal**

Examiners

1

2

DECLARATION

I hereby declare that this project work entitled **"ARDUINO-POWERED AUTONOMOUS VEHICLE"** has been prepared by us during the year 2022-23 under the guidance of **Ms. Yashoda.S.Patil** Department of **Software and Application Development**, S.D.M College in the partial fulfilment of B.Voc degree prescribed by the college.

I also declare that this project is the outcome of our own efforts, that it has not been submitted to any other university for the award of any degree.

Thimmaiah C V

201344

ACKNOLWEDGEMENT

I would like to thank our principal **Dr. B A Kumar Hegde,** for his support. I also thank HOD, Dept. of B.Voc in Software and Application Development, SDM College Ujire, for his valuable suggestions.

I would like to thank Assistant Professor **Ms. Yashoda S Patil** Department of B.Voc in Software and Application Development, SDM College Ujire, for their help and for providing guidance in developing the project.

Thimmaiah C V

201344

INDEX

Chapter 1

Project Synopsis

1.1 Title of the project:

“Arduino-Powered Autonomous Vehicle”

1.2 Description about the project:

This project involves the development of an Arduino-powered autonomous vehicle that incorporates lane following using infrared sensors and obstacle detection using ultrasonic sensors. The vehicle utilizes infrared sensors to track and follow the lanes, ensuring accurate navigation within the designated paths. Furthermore, ultrasonic sensors are employed to detect obstacles in the vehicle's surroundings, allowing it to respond and avoid collisions in real-time. By combining these technologies, the project aims to enhance road safety and improve traffic management by minimizing human errors and providing an autonomous solution for efficient and secure transportation. This project represents a significant step towards the advancement of autonomous vehicle technology and its potential applications in various industries. The project seeks to gain practical experience in robotics, programming, and share its findings with the wider community.

1.3 Project Category:

IOT with sensors interfacing.

1.4 Languages to be used:

- C++

Chapter 2

Introduction

Internet of Things (IOT) is a concept where each device is assigned to an IP address and through that IP address anyone makes that device identifiable on internet. The mechanical and digital machines are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

The Arduino-Powered Autonomous Vehicle represents an exciting endeavor in the realm of intelligent transportation systems. By harnessing the capabilities of Arduino technology, this project aims to develop a car that can navigate autonomously while incorporating advanced features such as lane detection, tracking, and obstacle avoidance. The project utilizes an Arduino board as the central processing unit, which enables real-time data processing and decision-making capabilities. The integration of infrared and ultrasonic sensors with Arduino enhances the vehicle's ability to interpret its surroundings and make informed decisions for safe navigation. One crucial aspect of the project is the implementation of an infrared sensor for lane detection and tracking. This sensor allows the car to precisely identify and stay within designated lanes on the road. By continuously monitoring the infrared reflections from the road surface, the vehicle can accurately determine its position and make necessary adjustments to ensure it remains on the correct path. To further enhance safety and autonomous capabilities, an ultrasonic sensor is utilized to detect obstacles in the car's path. By emitting sound waves and measuring their reflections, the car can detect and analyze potential obstacles, enabling it to autonomously navigate and avoid collisions. By combining Arduino technology with these advanced functionalities, the project seeks to showcase the potential of intelligent transportation systems. The Arduino-Powered Autonomous Vehicle aims to demonstrate how Arduino's real-time data processing capabilities, along with infrared and ultrasonic sensors, can be effectively integrated to enhance the safety and autonomy of a vehicle. Through this project, we hope to contribute to the advancement of autonomous navigation technologies, laying the foundation for future developments in Self-Driving cars and intelligent transportation systems. By pushing the boundaries of what is possible with Arduino, infrared, and ultrasonic technologies, we aim to inspire innovation and drive progress in the field of intelligent transportation.

2.1 Background

In recent years, there has been a growing interest in autonomous vehicles and intelligent transportation systems. The development of Self-Driving cars has the potential to revolutionize transportation by improving safety, efficiency, and accessibility. Arduino, a popular open-source electronics platform, has emerged as a versatile tool for creating innovative projects and prototypes. The Arduino-Powered Autonomous Vehicle project utilizes Arduino technology to develop a car with autonomous capabilities. Arduino is an open-source electronics platform known for its versatility in creating interactive projects. Lane detection and tracking are crucial for autonomous navigation, often accomplished using infrared sensors to identify road markings. Obstacle avoidance is another essential feature for autonomous vehicles. Ultrasonic sensors are commonly used to detect objects in the vehicle's path by emitting sound waves and measuring their reflections. By integrating Arduino with infrared and ultrasonic sensors, the project aims to enhance the vehicle's autonomy and safety. Arduino's real-time data processing capabilities enable the vehicle to interpret sensor inputs and make intelligent navigation decisions. The Arduino-Powered Autonomous Vehicle project showcases the potential of combining Arduino, infrared, and ultrasonic technologies in developing intelligent transportation systems. It contributes to the advancements in autonomous vehicles, demonstrating the feasibility of creating efficient and reliable autonomous navigation systems.

2.2 Objective

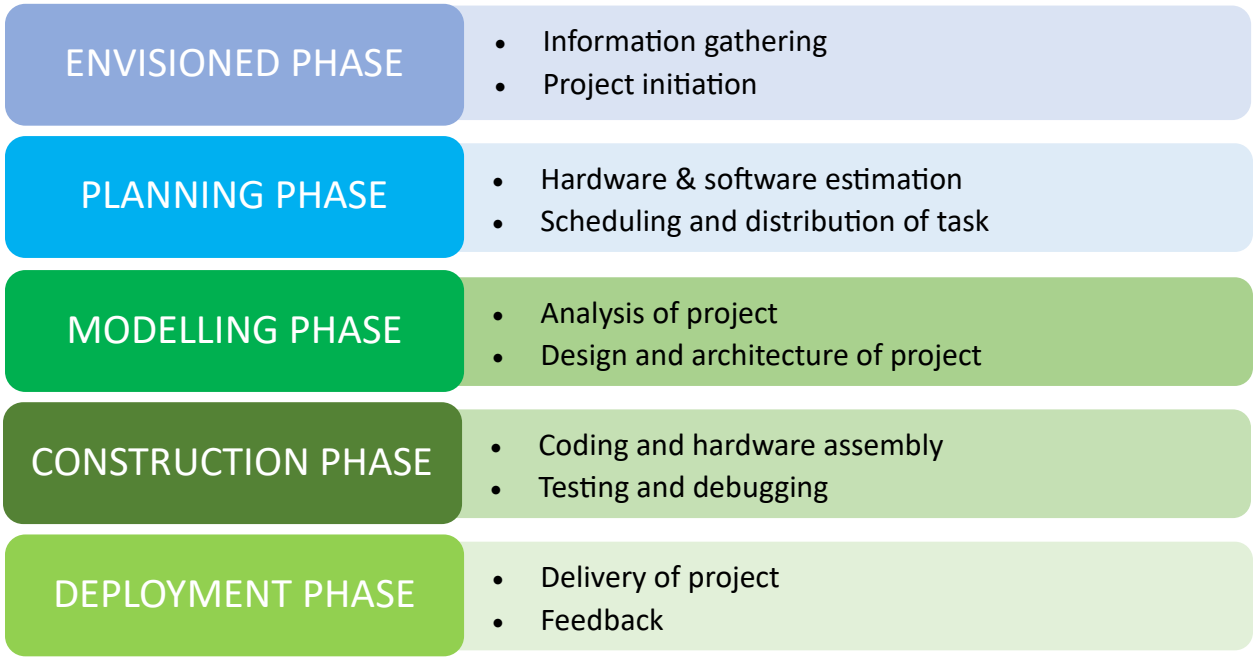
The objective of the Arduino-Powered Autonomous Vehicle project is to develop a car that utilizes Arduino technology, infrared sensors, and ultrasonic sensors to achieve autonomous navigation. The project aims to implement lane detection and tracking, obstacle avoidance, and real-time data processing to enable safe and efficient autonomous operation. The ultimate goal is to showcase the potential of combining these technologies in creating an intelligent transportation system that enhances vehicle autonomy and safety.

2.3 Scope

The development of an Arduino-powered autonomous vehicle with lane following using infrared sensors and obstacle detection using ultrasonic sensors has significant potential impact. This project offers the opportunity to enhance road safety by minimizing human errors and improving traffic management. The implementation of lane following technology allows the vehicle to navigate and stay within the designated lanes, reducing the risk of accidents due to lane drifting. Additionally, obstacle detection using ultrasonic sensors enables the vehicle to detect and respond to obstacles in real-time, preventing collisions and ensuring safer journeys. This project contributes to the advancement of autonomous vehicle technology, paving the way for future applications in transportation and robotics.

2.4 Project management

Effective project management is crucial for the successful execution of the Arduino-Powered Autonomous Vehicle project. Our project has been decomposed into the following phases:



2.4.1 Experimentation

Experimentation in the Arduino-Powered Autonomous Vehicle project involves systematic testing and evaluation to validate the performance and reliability of the autonomous functionalities. It includes designing test scenarios, collecting relevant data, analyzing results, and making iterative improvements based on findings. Through experimentation, the project team gains insights into the vehicle's capabilities, identifies areas for improvement, and ensures that the autonomous features meet the project's objectives.

2.4.2 Design

The design of the Arduino-Powered Autonomous Vehicle involves the integration of hardware components and software algorithms for autonomous navigation. It includes the selection of a suitable chassis, integration of sensors for lane detection and obstacle avoidance, and the development of software modules for data processing and control. The design also encompasses considerations for user interface, safety features, and iterative improvements based on testing and feedback. The overall goal is to create a well-designed system that enables the vehicle to autonomously navigate, detect lanes, detect obstacles, and ensure reliable and efficient operation.

2.4.3 Development and Testing

The development phase involves building and integrating the hardware components, implementing the software algorithms, and calibrating the sensors. Testing involves evaluating the functionality and performance of the autonomous vehicle through various scenarios, making iterative improvements based on test results and feedback. The goal is to ensure the vehicle meets the project's objectives, such as accurate lane detection, obstacle avoidance, and reliable decision-making. Documentation is maintained to capture the development process, including hardware configurations, software code, and modifications, facilitating replication and future development.

2.4.4 Real-World Testing

Real-world testing is a crucial phase in the development of the Arduino-Powered Autonomous Vehicle project. It involves evaluating the performance and reliability of the vehicle in real-world scenarios, simulating actual driving conditions.

2.5 Overview and Benefits

The Arduino-Powered Autonomous Vehicle project focuses on developing a car equipped with advanced functionalities of lane detection and tracking, as well as obstacle avoidance. The project integrates Arduino, infrared sensors, and ultrasonic sensors to enable real-time data processing and decision-making capabilities for autonomous navigation.

Benefits:

2.5.1 Enhanced Safety

The Arduino-Powered Autonomous Vehicle aims to enhance safety by incorporating lane detection and tracking functionalities. It helps the vehicle stay within designated lanes, reducing the risk of accidents caused by drifting out of lanes.

2.5.2 Improved Efficiency

The autonomous vehicle utilizes real-time data processing and decision-making capabilities to navigate efficiently. It can make informed decisions to optimize its path and speed, potentially improving fuel efficiency and reducing travel time.

2.5.3 Autonomous Navigation

The integration of Arduino with infrared and ultrasonic sensors enables the vehicle to autonomously navigate and avoid obstacles. It reduces the need for manual intervention and provides a hands-free driving experience.

2.5.4 Real-World Adaptability

The project focuses on real-world testing to ensure the vehicle's performance and reliability under various road and traffic conditions. This adaptability allows the vehicle to operate in practical driving environments and handle diverse scenarios.

2.5.5 Technology Showcase

The Arduino-Powered Autonomous Vehicle project demonstrates the potential of combining Arduino, infrared, and ultrasonic technologies in intelligent transportation systems. It showcases the capabilities of these technologies in enhancing vehicle autonomy and safety.

2.5.6 Learning and Innovation

The project provides an opportunity for learning and innovation in the fields of robotics, autonomous systems, and software development. It encourages the exploration of new algorithms, sensor integration techniques, and hardware-software integration.

Chapter 3

Literature Review

“Obstacle Avoiding Car”, by Prof. N.A. Kumbhar, Shubham Pingale, Purvesh Khairnar, Mayuresh Devadkar, Mahesh Yejare.

This research discusses an ultrasonic sensor-controlled robot vehicle that can avoid obstacles. The robot is constructed with an ultrasonic sensor, and an Arduino microcontroller is used to operate it. Ultrasonic sensor mounted on the robot vehicle\'s front end. Through sensors that are attached on the robot, the sensor receives data from its surroundings. The sensor detects the obstruction and changes its course to choose a path devoid of obstacles. The sensor will be send the data to the controller is compared with controller to decide the movement of the robot Wheel. The robot wheel\'s movement and direction will depend on the wheel encoder and an ultrasonic sensor for sensing. This vehicle is employed for obstacle detection and collision avoidance. We have programmed the controller to be used with ARDUINO SOFTWARE.

“Line following with Obstacle avoidance Autonomous nursing assistant robot”, by Mohd Tanzeem Abdul Rahim, Rahil Ahmed Khan, Mohammad Ashraf, Neeraj Kumar, Utsav Kumar, Dr. Arun Kumar Singh.

The goal of this project is to create a line-following robot with the ability to avoid obstacles. The robot will be equipped with an infrared sensor array to detect the black line on the ground, and an ultrasonic sensor to detect obstacles. A microcontroller will be used to process the sensor data and generate control signals for the robot's motors. The robot's behavior will be determined by a set of algorithms that combine line following and obstacle avoidance techniques. This research will demonstrate the feasibility of designing an autonomous robot that can navigate through a track with a black line while avoiding obstacles. In this study, a line-following, obstacle-avoidance robot that serves as a nursing assistant is developed. The robot can safely move food, medicine, blood samples, and other items around a hospital on its own initiative and in accordance with user instructions. A nurse can call the robot from any location using wireless connectivity. The robot will automatically travel a path to the caller nurse in response to the call. After that, the caller placed any further materials.

“Design of autonomous line follower robot with obstacle avoidance”, by Kumar Rishab.

This paper shows design and implementation of the Line Follower Robot and its ability to select the desired line among black and white line. This can be combined with different colors. Since each color has its own distinct property, robot can therefore easily differentiate among different colors and possess the ability to detect the presence of an obstacle and choose the other path to find its target. It is programmed in such a way that instructions are given to the robot which senses a line and attempts to move towards the target. The robot can easily move along very congested curves as it continuously data from the sensors. This robot avoids collision and it can detect collision with an obstacle sensor and hence reaching the target. The proposed system can be implemented in any commercial, industrial, medical and also in educational labs.

Chapter 4

System Requirement Specification

Requirements specification is a specification of software requirements and hardware requirements required to do the project. Requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements.

4.1 Hardware Requirements

SL No	Hardware	Specification
1	4WD Car Chassis Kit	A kit includes a chassis, wheels, and a frame to assemble the vehicle structure.
2	Arduino UNO	An Arduino microcontroller board that acts as the brain of the vehicle, controlling the motors and processing sensor data.
3	L298N Motor Driver	A motor driver module used to control and drive the DC gear motors of the vehicle.
4	DC Gear Motors	Motors that provide the necessary propulsion and steering for the vehicle.
5	Breadboard	A board used for prototyping and connecting various components and circuits.
6	Infrared Sensor	A sensor that detects infrared light to aid in lane detection and tracking.
7	Ultrasonic Sensor	A sensor that uses sound waves to detect obstacles and measure distances.
8	Jumper Wires	Jumper Wires that are used to establish connections between components and the Arduino board.
9	Castor Wheel	Castor wheels are commonly used in Arduino projects for easy movement and navigation.
10	Power Supply	A power source that provides the required voltage and current to operate the Arduino board, motor driver, and other components.

4.2 Software Requirements

SL No	Software	Specification
1	Operating System	Windows 8 and above
2	IDE	Arduino IDE
3	Programming Language	C++

4.3 Hardware Components

4.3.1 4WD Car Chassis Kit

The 4WD Car Chassis Kit is a hardware component that serves as the foundation for the Arduino-Powered Autonomous Vehicle project. It includes a chassis, wheels, motors, and mounting hardware. The chassis provides a sturdy structure for assembling the vehicle, while the wheels and motors enable mobility and control. The kit is designed for a four-wheel drive configuration, allowing the vehicle to navigate various terrains. By utilizing the 4WD Car Chassis Kit, you can easily build and customize your autonomous vehicle, providing a reliable platform for integrating the other components and sensors required for autonomous operation.

Figure 4.3.1 4WD Car Chassis Kit

4.3.2 Arduino UNO

Arduino Uno is an open-source microcontroller board popular for DIY projects. It features an ATmega328P microcontroller running at 16 MHz. With 14 digital I/O pins, 6 analog inputs, and various communication interfaces, it enables easy interaction with sensors, motors, and other electronic components. Arduino Uno can be programmed using the Arduino Software (IDE) and supports shields for expanded capabilities. It operates on a wide voltage range of 7-20 volts and provides a regulated 5-volt output. Beginner-friendly and extensively documented, the Uno is widely used for prototyping, robotics, home automation, and more, making it a versatile and accessible choice for electronics enthusiasts.

Figure 4.3.2 Arduino UNO

4.3.3 L298N Motor Driver

The L298N Motor Driver is an integrated circuit module used to control and drive DC motors. It features dual H-bridge circuits for bidirectional motor control, handles significant current and voltage ranges, and includes built-in protection features. With easy interface options, it is compatible with various microcontrollers and is commonly used in robotics and electronics projects for controlling motor movement and speed.

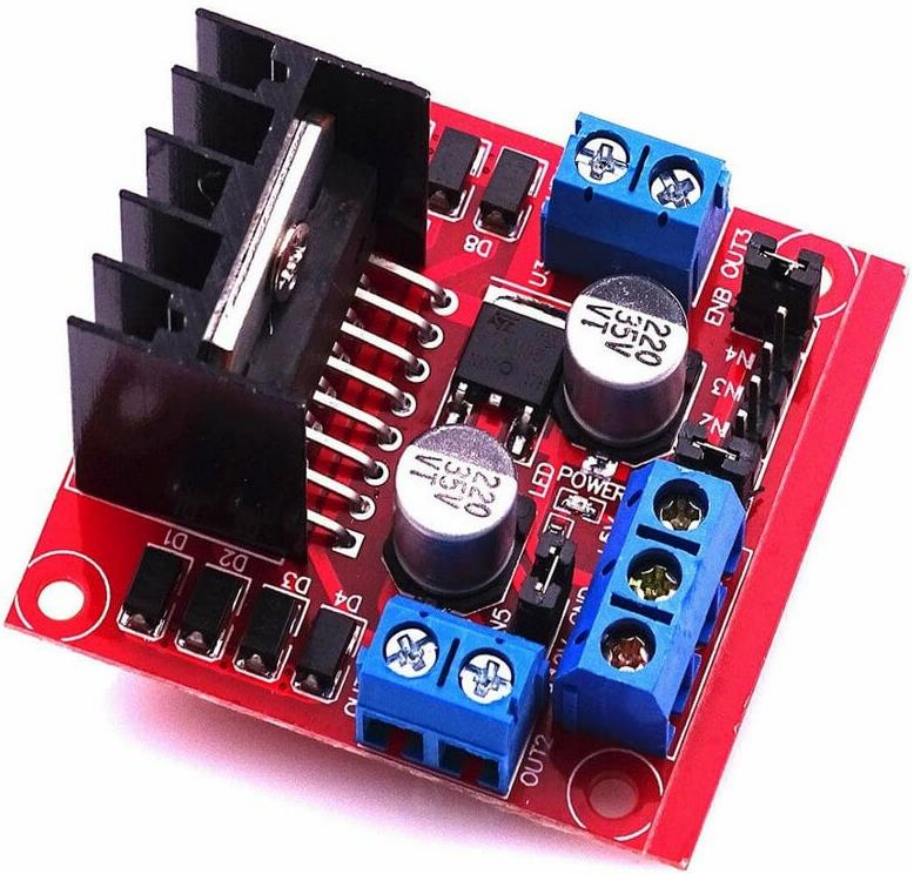


Figure 4.3.3 L298N Motor Driver

4.3.4 DC Gear Motors

DC gear motors are electric motors that incorporate a gearbox for increased torque and reduced speed. They are commonly used in robotics, automation, and other projects that require precise control over motor speed and torque. With their compact size and controllable characteristics, DC gear motors are well-suited for applications such as the Arduino-Powered Autonomous Vehicle project, where they can be used to drive the vehicle's wheels and enable precise maneuverability.

Figure 4.3.4 DC Gear Motors

4.3.5 Breadboard

A breadboard is a prototyping tool used in electronics projects for creating temporary connections and testing circuit designs. It consists of a grid of interconnected holes arranged in rows and columns, allowing components to be easily inserted and connected without the need for soldering. Breadboards provide a flexible and reusable platform for quick circuit assembly and experimentation, making them ideal for testing and prototyping circuits in projects like the Arduino-Powered Autonomous Vehicle.

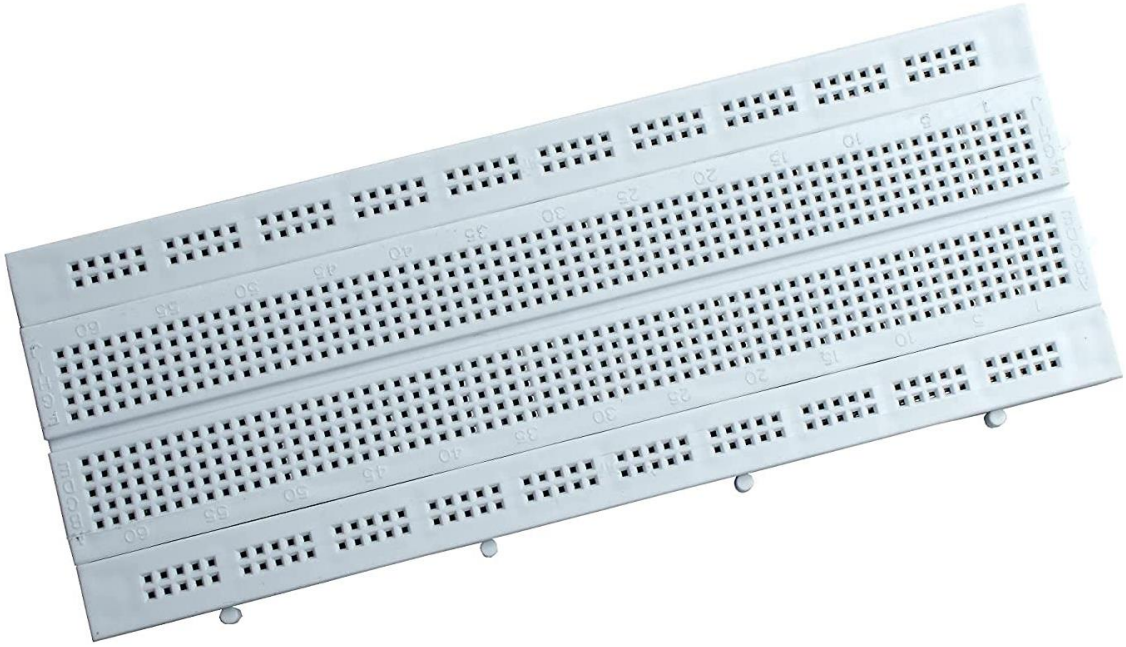


Figure 4.3.5 Breadboard

4.3.6 Infrared Sensor

An infrared (IR) sensor is a device that detects infrared radiation emitted or reflected by objects. It converts this radiation into an electrical signal for further processing. In the Arduino-Powered Autonomous Vehicle project, an infrared sensor can be used for lane detection and tracking. By detecting the infrared radiation from the lane markings, the vehicle can accurately determine its position and stay within the designated lanes. The sensor can be interfaced with the Arduino for real-time data processing and decision-making, enhancing the vehicle's autonomy and safety.

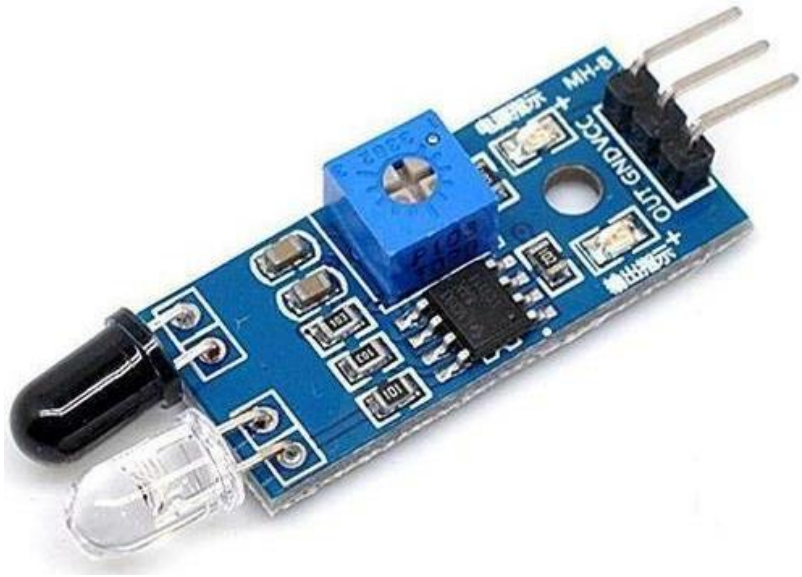


Figure 4.3.6 Infrared Sensor

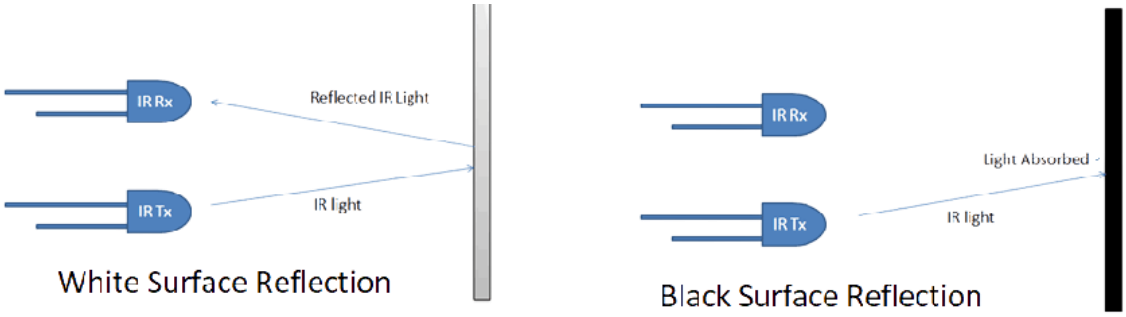


Figure 4.3.7 Working of Infrared Sensor on different Surface

4.3.7 Ultrasonic Sensor

An ultrasonic sensor is a device that uses sound waves to measure distance and detect objects. It emits high-frequency sound waves and measures the time it takes for the waves to bounce back after hitting an object. This allows the sensor to calculate the distance to the object. In the context of the Arduino-Powered Autonomous Vehicle project, an ultrasonic sensor can be used to detect obstacles in the vehicle's path. By interfacing the sensor with the Arduino, the vehicle can autonomously navigate and avoid collisions. Ultrasonic sensors provide a reliable and accurate means of detecting objects, making them valuable for enhancing the vehicle's safety and obstacle avoidance capabilities.

Figure 4.3.7 Ultrasonic Sensor

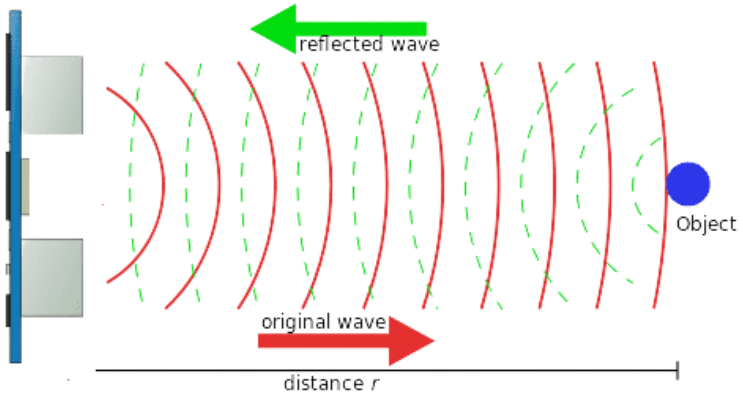


Figure 4.3.8 Working of Ultrasonic Sensor for obstacle detection

4.3.8 Jumper wire

Jumper wires are versatile electrical wires used in electronics projects to create temporary connections between components. They are made of insulated copper wire with connectors on each end, allowing for easy insertion into breadboards or other prototyping platforms. Jumper wires provide flexibility in circuit design, can be easily adjusted and reused, and are compatible with various components and prototyping platforms. They are essential for quick and flexible prototyping and testing in projects like the Arduino-Powered Autonomous Vehicle.

Figure 4.3.8 Jumper Wires

4.3.9 Castor Wheel

Castor wheels play a crucial role in Arduino projects, providing mobility and flexibility to various robotic creations. These small, rotating wheels are designed to be attached to the bottom of a robot or any moving mechanism, allowing it to move smoothly and change directions effortlessly. Castor wheels typically have a swivel mechanism, enabling omnidirectional movement. Arduino enthusiasts often use castor wheels in conjunction with motorized wheels to create agile and versatile robots. By incorporating these wheels into their projects, makers can enhance the maneuverability and overall functionality of their Arduino-based creations, opening up a world of possibilities for robotic exploration and experimentation.

Figure 4.3.9 Castor Wheels

4.3.10 Power Supply

A power supply is a device or system that converts input power into a suitable form to supply electrical energy to electronic components. It ensures a steady and regulated power output to ensure proper functioning of the components. Power supplies can be of different types, such as linear or switch-mode, and provide specific voltage and current ratings. They often include features like voltage regulation and safety protections. In the Arduino-Powered Autonomous Vehicle project, a power supply is needed to provide stable power to the Arduino board, motors, sensors, and other components, ensuring reliable operation of the vehicle system.

Figure 4.3.10 Power Supply

4.2 Software Requirement

4.4.1 Operating System - Windows 8 and above

The choice of the operating system for the Arduino-Powered Autonomous Vehicle project is Windows 8 and above. Windows 8 and above provide a reliable and widely adopted operating system environment that supports the development and execution of the project. The decision to specify Windows 8 and above ensures compatibility with the Arduino IDE and its associated tools, enabling a seamless programming experience for the Arduino boards. By selecting Windows 8 and above, the project benefits from the stability, user-friendly interface, and driver support offered by these operating systems. Furthermore, Windows 8 and above have a large user base, which translates into a vast community of Arduino enthusiasts and developers who can provide support, share knowledge, and contribute to the project's success. The availability of Windows 8 and above as the recommended operating system ensures that the Arduino-Powered Autonomous Vehicle project can be executed efficiently, leveraging the power of the Windows platform to develop and control the autonomous vehicle's functionalities.

4.4.2 Arduino IDE

The Arduino IDE (Integrated Development Environment) is a powerful software application designed specifically for programming Arduino microcontrollers. It offers a comprehensive set of tools and features that simplify the development process and enable users to write and upload code to Arduino boards.

The Arduino IDE provides an intuitive and user-friendly interface for writing, editing, and organizing code. It offers a code editor with syntax highlighting and auto-completion features, making it easier to write Arduino sketches in the C/C++ programming language. The IDE supports various Arduino boards, including the Arduino Uno, Nano, Mega, and others, ensuring compatibility with a wide range of projects. It allows you to select the appropriate board and port for uploading the code. Compiling and uploading code to the Arduino board is seamless with the IDE. It automatically compiles the code, generates the necessary binary files, and uploads them to the board via a USB connection. The IDE includes a serial monitor that allows you to communicate with the Arduino board and monitor the output or receive input during runtime. This is particularly useful for debugging and troubleshooting your code.

The library manager in the Arduino IDE simplifies the process of adding additional functionality to your projects. It provides a vast collection of pre-written libraries that you can easily install and use in your code, saving you time and effort.

The IDE supports third-party libraries and allows you to import custom libraries, expanding the capabilities of your projects. This enables you to leverage existing code and easily integrate complex functionalities, such as sensor data processing or motor control. The Arduino IDE is cross-platform, available for Windows, macOS, and Linux, ensuring compatibility with different operating systems. This allows users to work with their preferred OS without any limitations. The IDE provides an active and supportive community where users can share their projects, ask questions, and find resources. This fosters collaboration and learning, making it easier to overcome challenges and explore new possibilities. The Arduino IDE is constantly updated and improved with new features, bug fixes, and performance enhancements. Regular updates ensure that users can benefit from the latest developments in Arduino programming.

4.4.3 Programming Language – C++

The Arduino-Powered Autonomous Vehicle project utilizes the C++ programming language. C++ is a powerful and efficient language that supports object-oriented programming and offers a vast ecosystem of libraries. It is compatible with Arduino and provides access to low-level hardware resources. The use of C++ ensures efficient code execution and facilitates the development of modular and reusable code structures. Additionally, C++ benefits from a large community of developers and extensive online resources, making it easier to find support and solutions. Overall, C++ is well-suited for the project, enabling efficient and effective programming of the Arduino board for autonomous vehicle functionalities.

Chapter 5

Implementation

5.1 IOT (INTERNET OF THINGS)

IoT, or the Internet of Things, is a network of physical devices, vehicles, appliances, and other objects embedded with sensors, software, and connectivity that enables them to connect and exchange data over the internet. It allows these objects to collect and transmit data, interact with their environment, and be remotely monitored and controlled. IoT technology has the potential to transform various industries by enabling automation, remote access, and data-driven decision-making. It offers benefits such as improved efficiency, enhanced productivity, cost savings, and the ability to create intelligent and connected systems. By leveraging IoT in the Arduino-Powered Autonomous Vehicle project, it can enhance the vehicle's capabilities, enable remote monitoring and control, facilitate data analysis and insights, and contribute to creating a smarter and more connected transportation system.

Figure 5.1 Internet of Things

5.2 Features of IOT

5.2.1 Intelligence

IoT becomes a lot more useful when combined with artificial intelligence. For instance, if you are out of groceries, your smart refrigerator can notify you to bring some on your way back home. Things like these have been made possible by the application of artificial intelligence. IoT devices collect raw data from their surroundings and convert them into something useful and insightful. The IoT devices and systems are also trained with various machine learning models so that they can better understand the changes in their surroundings and perform better.

5.2.2 Connectivity

Connectivity refers to the ability of devices and systems to connect and communicate with each other over the internet. It enables seamless data exchange, collaboration, and coordination among IoT devices. Through connectivity, IoT devices can transmit and receive data, access remote resources, and leverage cloud services for storage and processing. This connectivity empowers real-time monitoring, remote control, and intelligent decision-making in various IoT applications, leading to enhanced efficiency, automation, and improved user experiences.

5.2.3 Dynamic nature

IoT systems should be dynamic in to change according to the changes in their environment to be of any business use. Let us understand this with an example. A smart air conditioner should be able to set the temperature of the room according to the prevalent weather conditions using the data gathered by the temperature sensor. It must also be able to set the perfect humidity level inside the room according to the changes in the humidity level of the surroundings.

5.2.4 Scaling

IoT systems are designed in such a way that the number of devices, sensors, or computers can be scaled up and down according to the need. An IoT system should be elastic enough so that it can handle workload during peak demand hours and can resort back to the normal state when the demand is low.

5.2.5 Sensing

Humans can naturally understand and analyze our circumstances easily based on past experiences with various things or situations. In the case of IoT, we need to read the analog signal and convert it to derive meaningful insights from it to get the best of it. We use Electrochemical, gyroscope, pressure, light sensors, GPS, Electrochemical, pressure, RFID, etc., to gather data based on a particular problem. For example, we use Light detection and pressure, velocity, and imagery sensors for automotive use cases. We must choose the proper sensing paradigm to make a use case successful.

5.2.6 Active Engagement

IoT connects its devices and products with cross-domain technologies like cloud computing, artificial intelligence, blockchain technology, etc. An active engagement between these products and technologies is essential to gather and manipulate data to make it of business use. Raw data has huge potential and can improve business decisions considerably. Therefore, an active engagement between various IoT products and these technologies is the need of the hour.

5.2.7 Security

Security is one of the major concerns among the users of IoT. IoT systems carry and store a lot of sensitive information, so the security of the devices and the data flowing between them should be given foremost priority. Proper security and safety measures are implemented while designing an IoT system to prevent a possible breach of security. Resources and investment required to ensure a safe and conductive IoT system are huge, but its safety and security must be ensured. Failing to do so can lead to mistrust among its users and businesses and can reduce its demand.

5.3 Advantages of IOT

5.3.1 Monitoring

IoT allows remote monitoring of devices and personal assets. The owner could be in any part of the world and by the means of his/her smartphone they can check up and monitor their assets without spending revenue on manual labour. Further IOT helps in understanding the internal details of your devices. It notifies the user when the product is damaged or out of stock. IoT keeps life going without much interference from the user.

5.3.2 Accessibility

IoT allows users to access real-time information about their devices from any part of the world. Users can connect to the application and gather information about their personal devices. For example: a person switches on the AC when they are close to home so the room can be cool before they enter.

5.3.3 Automation and control

The physical objects are connected to each other with wireless forms of technology and hence automation plays a huge role in the internet of things. It allows the working of these devices without human intervention. These devices communicate with each other to send and receive information at all times.

5.3.4 Human effort minimization

Human effort minimization refers to the reduction of manual work through the use of automation, technology, and efficient processes. By automating repetitive tasks, streamlining workflows, integrating technology solutions, and optimizing processes, organizations can minimize the amount of manual labor required. This leads to increased productivity, improved efficiency, and the ability to allocate human resources to more valuable and strategic activities. Human effort minimization aims to optimize operations and maximize output while reducing the physical and mental strain on individuals.

5.3.5 Time management

Time management is the process of effectively organizing and prioritizing tasks to optimize productivity. It involves setting goals, creating schedules, and allocating time to specific activities. By avoiding procrastination, setting priorities, and eliminating time-wasting activities, individuals can make the most efficient use of their time and achieve their desired outcomes. Effective time management improves productivity, reduces stress, and helps individuals meet their goals in a timely manner.

5.3.6 Security

IOT devices with string security steam are almost impossible to hack into. These devices have tighter secret protocols that alert the user instantly in case of any destructive attempt to physical devices. Securing IoT devices and systems is crucial to prevent data breaches, unauthorized access, and malicious activities. By implementing robust security measures, organizations can protect sensitive data, maintain user privacy, and ensure the trustworthiness of IoT deployments.

5.3.7 Resource utilization efficiency

Efficient resource utilization helps organizations optimize productivity, reduce costs, and enhance overall performance. By effectively managing and allocating resources, organizations can achieve their objectives while minimizing waste and maximizing efficiency. Adopting practices that reduce waste and promote recycling can improve resource utilization efficiency. By implementing waste management strategies, organizations can minimize material waste, energy consumption, and environmental impact.

5.3.8 Reduction in usage of technical equipment

By reducing the usage of technical equipment, organizations can minimize costs, energy consumption, and environmental impact. It promotes efficiency, sustainability, and the optimal utilization of available resources. Regularly maintain and optimize existing equipment to ensure its efficiency and extend its lifespan. Implement preventive maintenance practices, calibrate equipment as needed, and upgrade or replace outdated components to improve efficiency.

5.3.9 Cost efficiency

Cost efficiency refers to achieving desired outcomes or results while minimizing costs. It involves analyzing costs, optimizing resource utilization, streamlining processes, and leveraging technology to reduce expenses. By implementing cost-saving measures, organizations can enhance profitability, maintain financial stability, and achieve their objectives effectively and efficiently. Regularly monitor and evaluate costs to identify trends, anomalies, and areas for improvement. Implement performance metrics and reporting systems to track cost efficiency and identify opportunities for optimization.

5.3.10 Data Collection

Data collection is the process of gathering information or data from various sources for analysis and decision-making. It involves identifying the purpose of data collection, selecting appropriate methods and tools, ensuring data quality, and documenting the process. Effective data collection enables organizations to obtain reliable and relevant data to derive insights and make informed decisions. It clearly defines the purpose and objectives of data collection. Determine what information is needed, why it is important, and how it will be used to support the desired outcomes.

5.3.11 Information

Information is processed and organized data that provides meaningful insights, understanding, and knowledge. It adds context, interpretation, and relevance to raw data, enabling informed decision-making and effective communication. Information is accurate, reliable, and communicated in a timely manner, supporting action and driving outcomes. It is a valuable resource for addressing specific needs, solving problems, and making informed choices. Information is typically communicated through various formats such as reports, charts, graphs, or presentations. It is presented in a clear, concise, and easily understandable manner to facilitate effective communication and decision-making.

5.4 Disadvantages of IOT

5.4.1 Security Risks

IoT devices can be vulnerable to cybersecurity threats, including hacking, data breaches, and unauthorized access. The interconnected nature of IoT systems increases the potential attack surface, making it crucial to implement robust security measures.

5.4.2 Privacy Concerns

IoT devices collect and transmit vast amounts of data, raising concerns about privacy. Personal information and sensitive data may be at risk if not properly protected or if used without individuals' consent.

5.4.3 Complexity

The complexity of IoT systems, with their diverse devices, protocols, and interfaces, can make their management and maintenance challenging. Integration and interoperability issues may arise, requiring specialized expertise to ensure smooth operation.

5.4.4 Reliability and Connectivity

IoT relies heavily on network connectivity. Any disruption in connectivity can affect the functionality and performance of IoT devices, leading to unreliable or intermittent operation.

5.4.5 Data Overload

The sheer volume of data generated by IoT devices can overwhelm systems and users. Extracting meaningful insights from massive datasets requires efficient data processing and analytics capabilities.

5.4.6 Cost and Infrastructure

The implementation of IoT systems often involves significant costs, including device procurement, infrastructure development, and maintenance. These costs can be a barrier to entry for smaller businesses or organizations with limited resources.

5.4.7 Standardization and Compatibility

Lack of uniform standards and compatibility between different IoT devices and platforms can lead to fragmentation and interoperability challenges. This can limit the seamless integration and scalability of IoT solutions.

5.4.8 Power Consumption

Many IoT devices rely on batteries or power sources with limited capacity. Power consumption can be a concern, especially for devices deployed in remote locations or inaccessible environments, necessitating frequent battery changes or recharging.

5.4.9 Ethical and Legal Considerations

The use of IoT raises ethical and legal questions, such as data ownership, consent, and accountability. Clear frameworks and regulations need to be established to address these concerns and protect individuals' rights.

5.4.10 Overreliance and Dependency

Increased reliance on IoT devices and automation can lead to dependency. System failures, malfunctions, or disruptions can have significant consequences if there are no backup systems or alternative solutions in place.

5.5 Application areas for the Internet of Things

5.5.1 Smart Home

Smart home in the Internet of Things (IoT) refers to a network of interconnected devices and sensors within a residence that can be remotely controlled and monitored. It involves integrating devices such as lighting, thermostats, security systems, and appliances, enabling automation, energy efficiency, and convenience. IoT-enabled smart homes enhance the overall living experience by providing seamless connectivity and intelligent control over various aspects of the home environment.

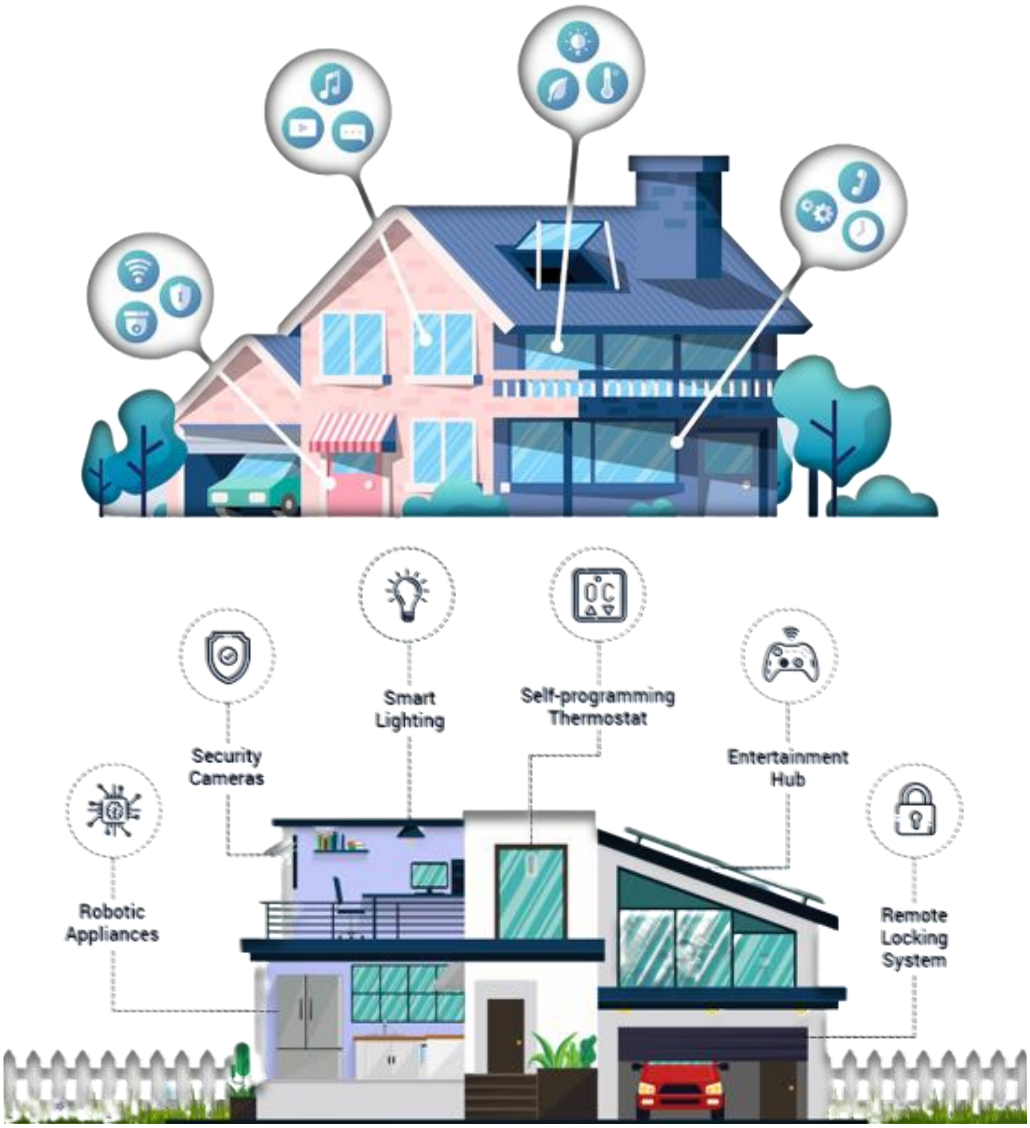


Figure 5.5.1 Smart Home

5.5.2 Smart City

Smart surveillance, automated transportation, smarter energy management systems, water distribution, urban security and environmental monitoring are examples of internet of thins applications for smart cities. IOT will solve major problems faced by the people living in cities like pollution, traffic congestion and shortage of energy supplies etc. By installing sensors and using web applications, citizens can find free available parking slots across the city. Also, the sensors can detect meter tempering issues, general malfunctions and any installation issues in the electricity system.

Figure 5.5.2 Smart City

5.5.3 Wearables

Wearable devices are installed with sensors and software's which collect data and information about the users. This data is later preprocessed to extract essential insights about user. These devices broadly cover fitness, health and entertainment requirements. The pre requisite from internet of things technology for wearable applications is to be highly energy efficient or ultra low power and small sized

Figure 5.5.3 Wearables

5.5.4 Healthcare

IoT in healthcare is aimed at empowering people to live healthier life and regular checkup by wearing connected devices. The collected data will help in personalized analysis of an individual's health and provide tailor made strategies to combat illness.

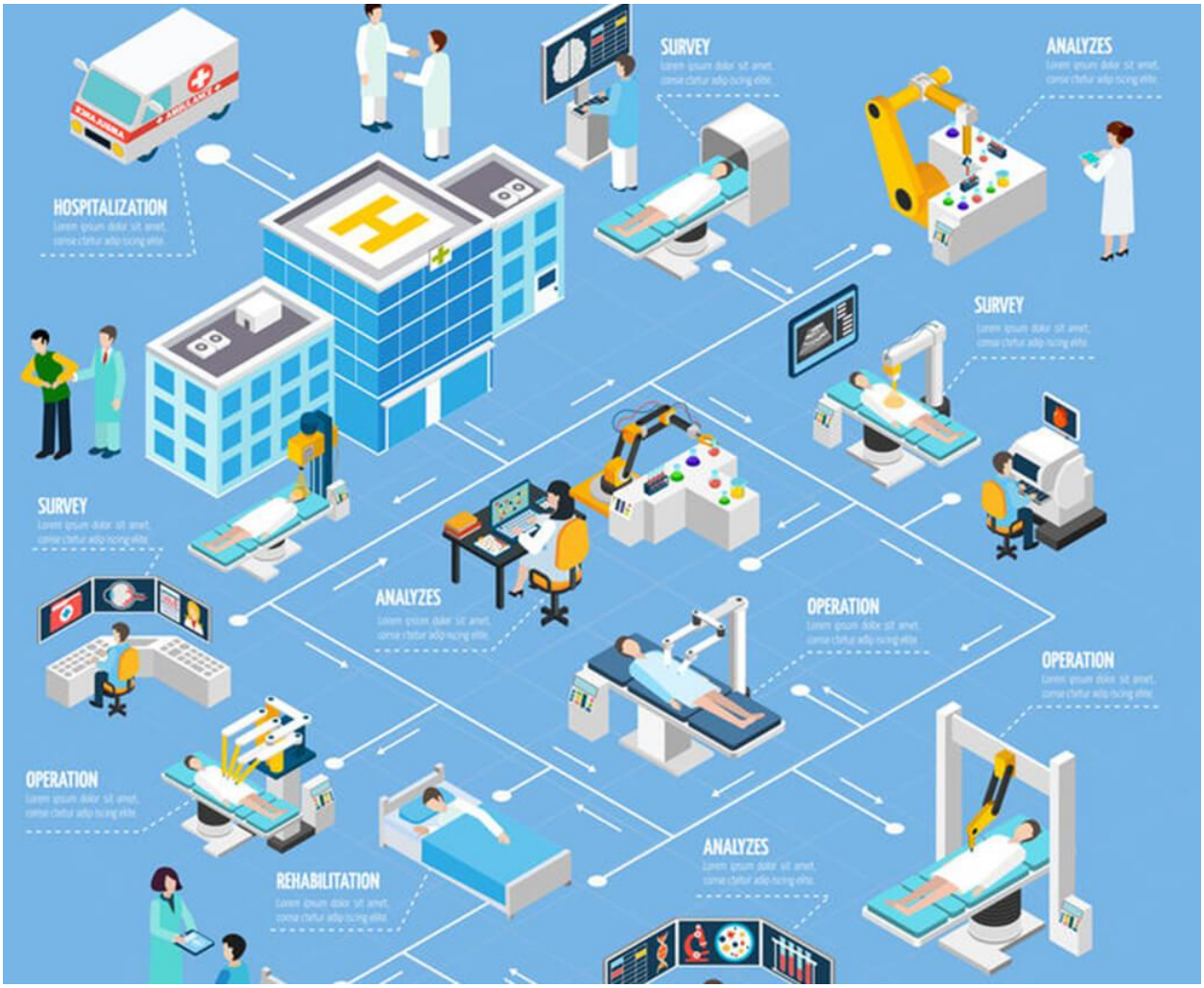


Figure 5.5.4 Smart Healthcare

5.5.5 Smart farming

Smart farming is an often-overlooked business-case for the internet of Things because it does not really fit into the well-known categories such as health, mobility, or industrial. However, due to the remoteness of farming operations and the large number of livestock that could be monitored the Internet of Things could revolutionize the way farmers work. But this idea has not yet reached large-scale attention. Nevertheless, one of the Internet of Things applications that should not be underestimated. Smart farming will become the important application field in the predominantly agricultural-product exporting countries

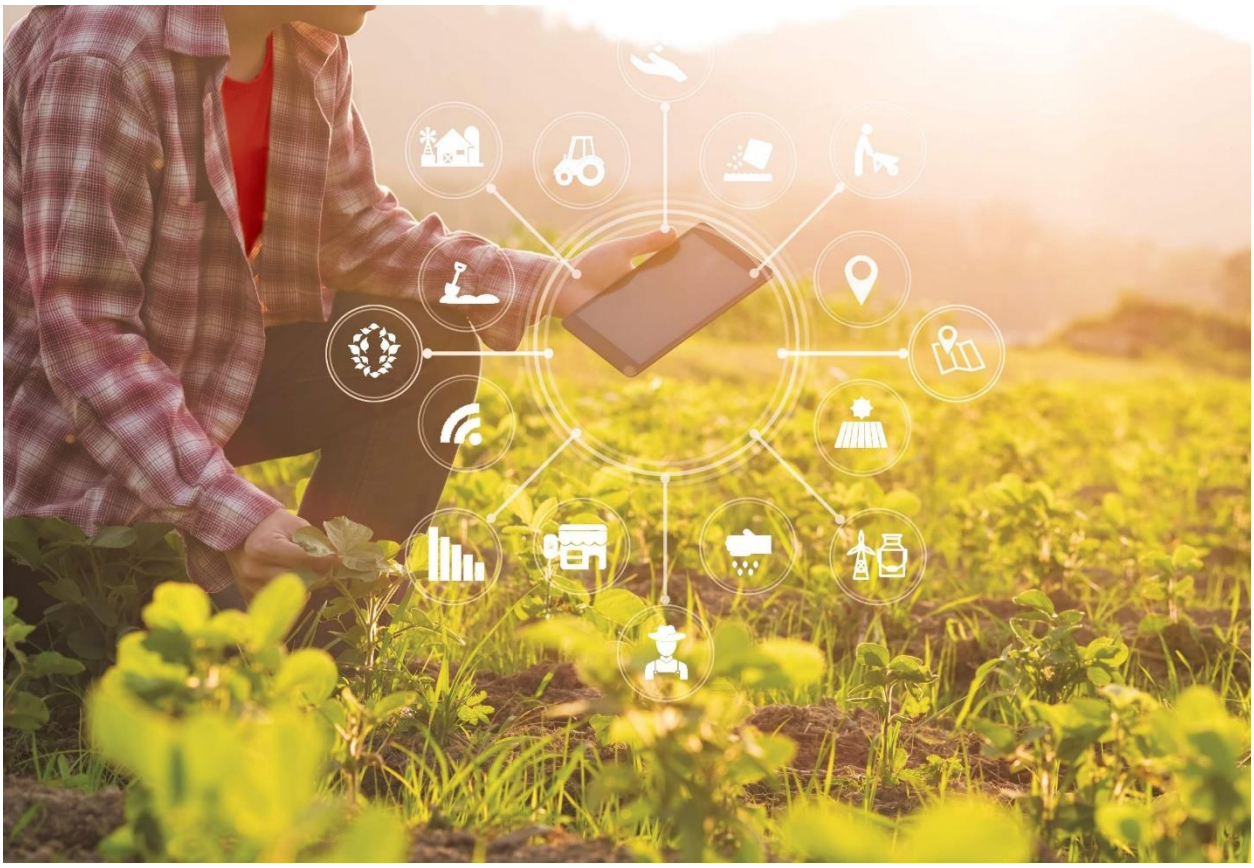


Figure 5.5.5 Smart farming

5.5.6 Smart supply chain

Supply chains have been getting smarter for some years already. Solutions for tracking goods while they are on the road, or getting suppliers to exchange inventory information have been on the market for years. So while it is perfectly logic that the topic will get a new push with the Internet of Things, it seems that so far its popularity remains limited.

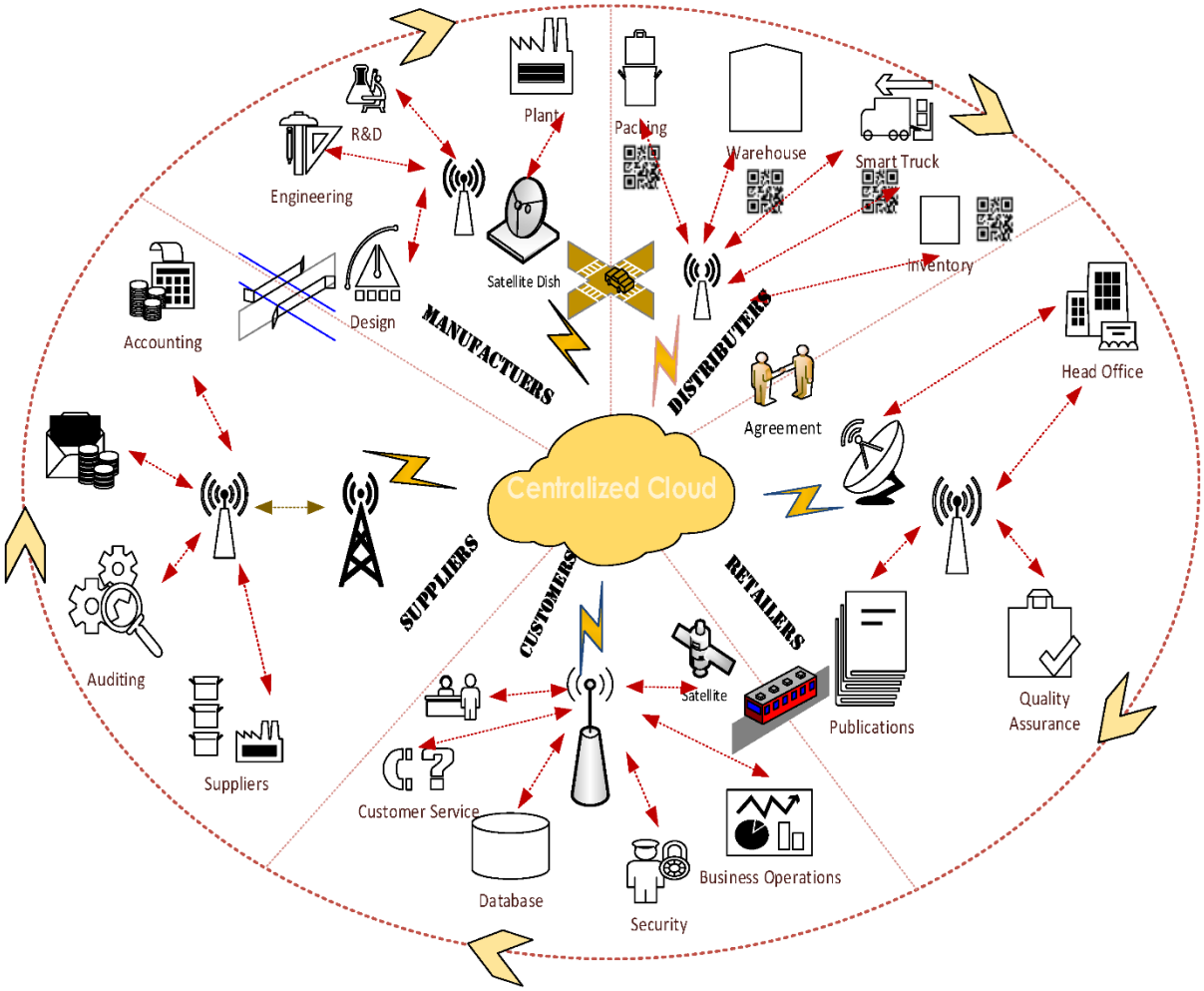


Figure 5.5.6 Smart supply chain

5.5.7 Connected car

The connected car is coming up slowly. Owing to the fact that the development cycles in the automotive industry typically take 2-4 years, we haven't seen much buzz around the connected car yet. But it seems we are getting there. Most large auto makers as well as some brave startups are working on connected car solutions. And if the BMWs and Fords of this world don't present the next generation internet connected car soon, other well-known giants will: Google, Microsoft, and Apple have all announced connected car platforms.

Figure 5.5.7 Connected Car

5.5.8 Industrial Automation

For a higher return of investment this field requires both fast developments and quality of products. This vitality thus coined the term IIOT. This whole schematic is re-engineered by 10T applications. Following are the domains of IOT applications in industrial automation

- Factory Digitalization
- Product flow Monitoring
- Inventory Management
- Safety and Security
- Quality Control
- Packaging optimization
- Logistics and Supply Chain Optimization

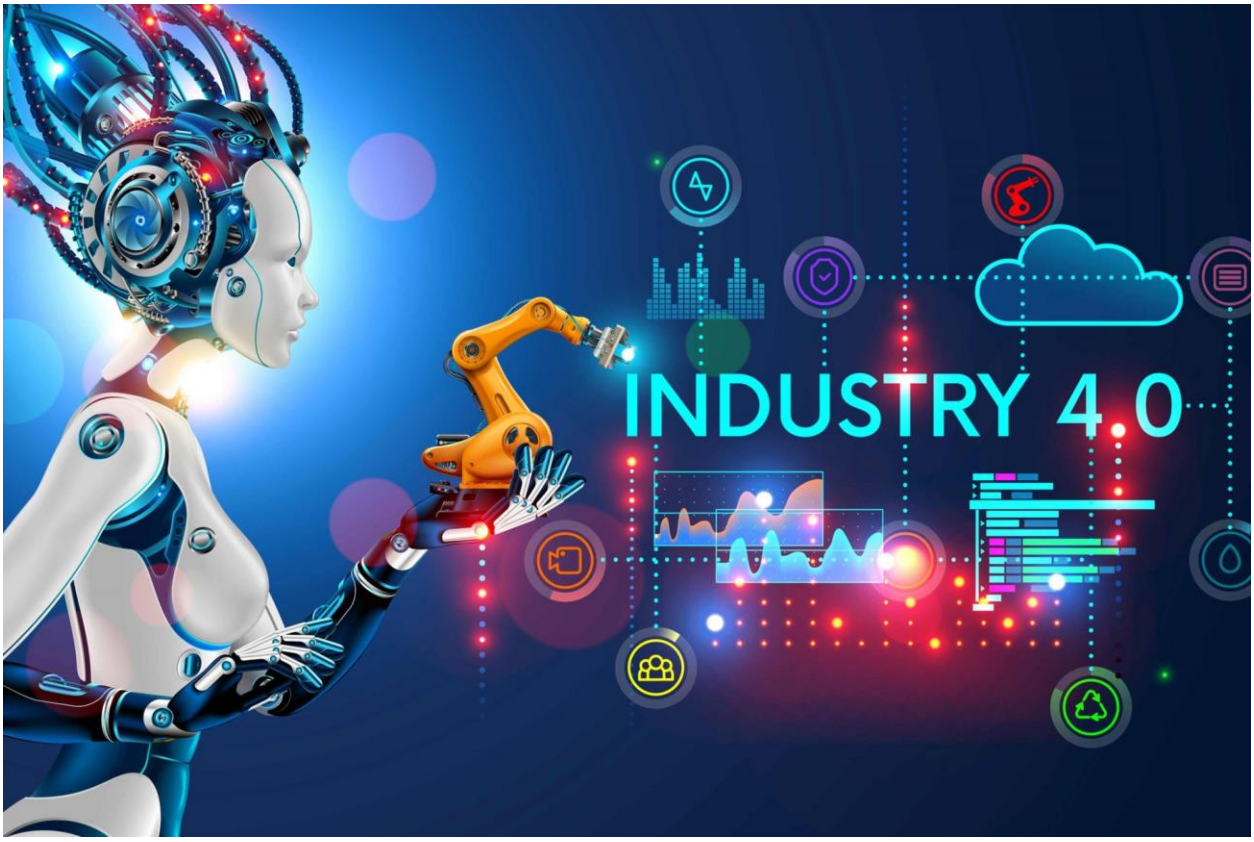


Figure 5.5.8 Industrial Automation

5.5.9 Government and safety

IOT applied to government and safety allows improved law enforcement, defense, city planning, and economic management. The technology fills in the current gaps, corrects many current flaws, and expands the reach of these efforts. For example, IOT can help city planners have a clearer view of the impact of their design, and governments have a better idea of the local economy.

Figure 5.5.9 Government and safety

5.6 Architecture of IoT

There are four-stage architecture of an IoT system

Stage 1 of an IoT architecture consists of your networked things, typically wireless sensors and actuators. Stage 2 includes sensor data aggregation systems and analog-to-digital data conversion. In Stage 3, edge IT systems perform preprocessing of the data before it moves on to the data center or cloud. Finally, in Stage 4, the data is analyzed, managed, and stored on traditional back-end data center systems. Clearly, the sensor/actuator state is the province of operations technology (OT) professionals. So is Stage 2. Stages 3 and 4 are typically controlled by IT, although the location of edge IT processing may be at a remote site or nearer to the data center. The dashed vertical line labeled "the edge" is the traditional demarcation between OT and IT responsibilities, although this is blurring. Here's a look at each in detail.

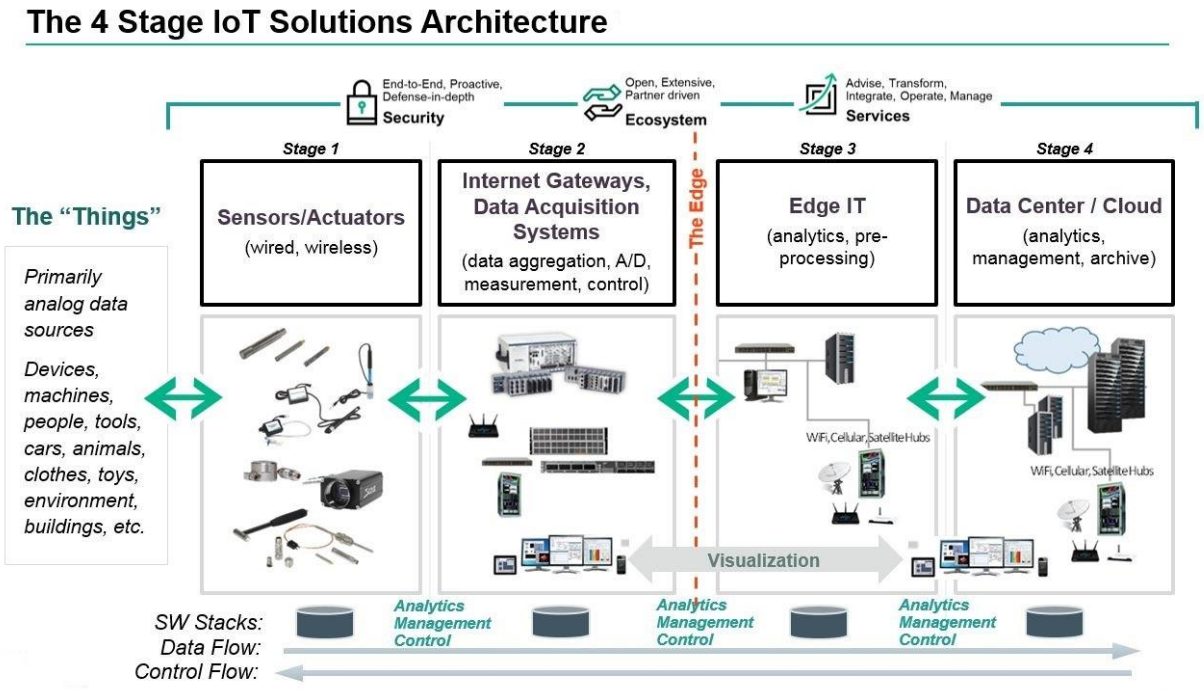


Figure 5.6 Four Stages of IOT Architecture

Stage 1. Sensors/Actuators

Sensors collect data from the environment or object under measurement and turn it into useful data. Think of the specialized structures in your cell phone that detect the directional pull of gravity and the phone's relative position to the "thing" we call the earth and convert it into data that your phone can use to orient the device. Actuators can also intervene to change the physical conditions that generate the data. An actuator might, for example, shut off a power supply, adjust an air flow valve, or move a robotic gripper in an assembly process.

The sensing/actuating stage covers everything from legacy industrial devices to robotic camera systems, water-level detectors, air quality sensors, accelerometers, and heart rate monitors. And the scope of the IoT is expanding rapidly, thanks in part to low-power wireless sensor network technologies and Power over Ethernet, which enable devices on a wired LAN to operate without the need for an A/C power source.

In an IoT architecture, some data processing can occur in each of the four stages. However, while you can process data at the sensor, what you can do is limited by the processing power available on each IoT device. Data is at the heart of an IoT architecture, and you need to choose between immediacy and depth of insight when processing that data. The more immediate the need for information, the closer to the end devices your processing needs to be.

For deeper insights that require more extensive processing, you'll need to move the data into a cloud- or data center-based system that can bring several sources of data together. But some decisions simply can't wait for deep processing. Did the robotic arm performing the surgery cut an artery? Will the car crash? Is the aircraft approaching the threat detection system a friend or a foe? You don't have time to send that data to your core IT assets. You must process the data right at the sensor- at the very edge of the edge network-for the fastest response.

Stage 2. The Internet gateway

The data from the sensors starts in analog form. That data needs to be aggregated and converted into digital streams for further processing downstream. Data acquisition systems (DAS) perform these data aggregation and conversion functions. The DAS connects to the sensor network, aggregates outputs, and performs the analog-to-digital conversion.

The Internet gateway receives the aggregated and digitized data and routes it over Wi-Fi, wired LANS, or the Internet, to Stage 3 systems for further processing.

Stage 2 systems often sit in close proximity to the sensors and actuators. For example, a pump might contain a half-dozen sensors and actuators that feed data into a data aggregation device that also digitizes the data. This device might be physically attached to the pump. An adjacent gateway device or server would then process the data and forward it to the Stage 3 or Stage 4 systems.

Why preprocess the data? The analog data streams that come from sensors create large volumes of data quickly. The measurable qualities of the physical world in which your business may be interested motion, voltage, vibration, and so on-can create voluminous amounts of constantly changing data. Think how much sensor data a complex machine like an aircraft engine might generate in one day, and there's no theoretical limit to the number of sensors that could be feeding data into an IoT system. What's more, an IoT system is always on, providing continuous connectivity and data feeds. IoT data flows can be immense I've seen as much as 40 TB/second in one case. That's a lot of data to transport into the data center. It's best to preprocess it.

Another reason not to pass the data on to the data center in this form is that analog data has specific timing and structural characteristics that require specialized software to process. It's best to convert the data into digital form first, and that's what happens in Stage 2.

Intelligent gateways can build on additional, basic gateway functionality by adding such capabilities as analytics, malware protection, and data management services. These systems enable the analysis of data streams in real time. Although delivering business insights from the data is a little less immediate at the gateway than it would be when sent directly from the sensor/actuator zone, the gateway has the compute power to render the information in a form that is more understandable to business stakeholders.

Gateways are still edge devices they're external to the data center so geography and location matter. In the pump example, if you have 100 pump units and want to process data on- premises, you might have instant data at the pump level, aggregate the information to create a plantwide view for the facility, and pass the data on to the data center for companywide view. DAS and gateway devices may end up in a wide variety of environments, from the factory floor to mobile field stations, so these systems are usually designed to be portable, easy to deploy, and rugged enough to withstand variations in temperature, humidity, dust, and vibration.

Stage 3. Edge IT

Once IoT data has been digitized and aggregated, it's ready to cross into the realm of IT.

However, the data may require further processing before it enters the data center. This is where edge IT systems, which perform more analysis, come into play. Edge IT processing systems may be located in remote offices or other edge locations, but generally these sits in the facility or location where the sensors reside closer to the sensors, such as in a wiring closet. Because IoT data can easily eat up network bandwidth and swamp your data center resources, it's best to have systems at the edge capable of performing analytics as a way to lessen the burden on core IT infrastructure. If you just had one large data pipe going to the data center, you'd need enormous capacity. You'd also face security concerns, storage issues, and delays processing the data. With a staged approach, you can preprocess the data, generate meaningful results, and pass only those on. For example, rather than passing on raw vibration data for the pumps, you could aggregate and convert the data, analyze it, and send only projections as to when each device will fail or need service.

Here's another example: You might use machine learning at the edge to scan for anomalies that identify impending maintenance problems that require immediate attention. Then you could use visualization technology to present that information using easy-to-understand dashboards, maps, or graphs. Highly integrated compute systems, such as hyper-converged infrastructure, are ideally suited to these tasks because they're relatively fast, and easy to deploy and manage remotely.

Stage 4. The data center and cloud

Data that needs more in-depth processing, and where feedback doesn't have to be immediate, gets forwarded to physical data center or cloud-based systems, where more powerful IT systems can analyze, manage, and securely store the data. It takes longer to get results when you wait until data reaches Stage 4, but you can execute a more in-depth analysis, as well as combine your sensor data with data from other sources for deeper insights. Stage 4 processing may take place on-premises, in the cloud, or in a hybrid cloud system, but the type of processing executed in this stage remains the same, regardless of the platform.

5.7 Communication protocol used for IOT

One of the essential factors of the IoT are communication protocols. One of the key aspects of the IoT is the communication between devices and it will be provided with communication protocols. The following paragraphs will provide a brief overview about each of the Internet of Things communication techniques.

5.7.1 6LoWPAN

6LoWPAN that is an IP based communication protocol is an acronym of IPv6 over Low Power Wireless Personal Area Network. 6LoWPAN, therefore, allows for the smallest devices with limited processing ability to transmit information wirelessly using an internet protocol. The most important detail of 6LOWPAN is IPv6. IPv6 (Internet Protocol version 6) is a 128-bit internet protocol developed due to inability of 32-bit IPv4 in addressing. While IPv6 developed, rapid increase of the number of connected devices has been considered.

5.7.2 MQTT (Message Queue Telemetry Transport)

It was created about 15 years back for monitoring remote sensor nodes and is designed to conserve both power and memory. It is based on the Publish Subscribe' communication model. Using MQTT, a connected device can subscribe to any number of topics hosted by an MQTT broker. It is overall a lightweight protocol that runs on embedded devices and mobile platforms. The good performance and reliability of MQTT is demonstrated by Facebook Messenger, Amazon IoT (AWS-IoT), IBM Node-Red, etc.-organizations that are using it to serve millions of people daily.

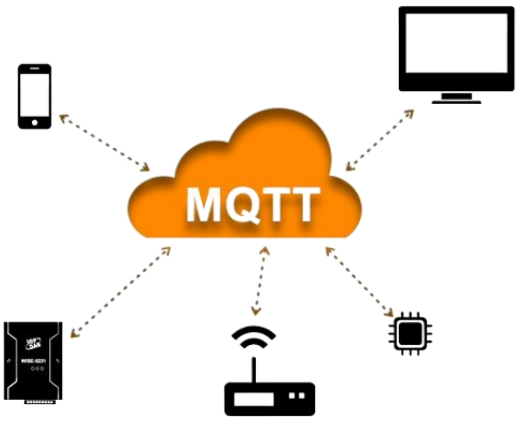


Figure 5.7.2 MQTT (Message Queue Telemetry Transport)

5.7.3 CoAP (Constrained Application Protocol)

Constrained Application Protocol (CoAP) is an Internet application protocol for constrained devices (defined in RFC 7228). It enables constrained devices to communicate with the wider Internet using similar protocols. COAP is designed for use between devices on the same constrained network, between devices and general nodes on the Internet, and between devices on different constrained networks joined by the Internet. COAP is a client/server protocol and provides a one-to-one "request/report" interaction model with accommodations for multicast. COAP is designed to interoperate with HTTP and the Restful web through simple proxies, making it natively compatible with the Internet. The two messaging protocols MQTT and CoAP are emerging as leading lightweight messaging protocols for the booming IoT market.

5.7.4 Bluetooth and Bluetooth Low Energy

While MQTT and COAP are infrastructure-independent, which means that it doesn't matter whether you're connected to a wired or a wireless network, Bluetooth provides only wireless communication over radio frequency. Bluetooth, generally, is divided into three categories. Bluetooth Classic: This is meant for high data rate applications like streaming audio wirelessly. Bluetooth Smart or Low Energy/BLE: This is meant for low powered battery-operated devices that stream low packets of data.

Bluetooth Smart Ready: These are essentially the 'hub' devices such as computers, smartphones, etc. They support both the 'classic' and 'smart' devices. Bluetooth technology is being used in the beacon technology that expected to revolutionize real-time marketing in the coming years. The new features of Bluetooth 5.0 version that is introduced as Bluetooth 5 have been developed completely for the Internet of Things. Generally, Bluetooth's range is 10 meters, but its range is up to 100 meters. Its data rate is 1mbps.

5.7.5 Mobile Network

The mobile network or in other words the cellular network, means wireless communication protocols such as 2G, 3G and 4G. It's easy to send and receive data in high quantity through especially 4G. High cost and high-power requirement are the disadvantages of this technology, but

having a high range of 200 km will be helpful for mobile applications in the IoT area. If it is based on LTE Advanced which is a 4G technology, the maximum data rate of this communication protocol is 1gbps.

5.7.6 Wi-Fi

Wi-Fi is highly suited for IoT applications where high-volume data transfer is made, however, it requires high power consumption. Generally, Wi-Fi technology's range is 10 meters, but Wi-Fi connectivity can be provided for up to 30 kilometers by using private antennas.

5.7.7 Z-Wave

Z-Wave is a wireless communication protocol developed specifically for home automation and has a low power requirement. Because it operates at 900 MHz, it is not affected by Wi-Fi and other wireless communication protocols running at 2.4 GHz, such as Bluetooth and Zigbee. Z- Wave is a simpler protocol than other communication protocols. This makes it possible to develop faster and simpler. Generally, Z-Wave's range is 30 meters, but this range can be up to 100 meters. Its maximum data rate is only 100 kbps.

5.7.8 Zigbee

Zigbee is a short-range wireless communication protocol based on the IEEE 802.15.4 protocol, which is widely used in home automation and the industry. It is preferred in applications where low power is required and data exchange is infrequent at low data rates. Low power consumption, high scalability, security, and durability makes Zigbee suitable for M2M and IoT applications. Generally, Zigbee's range is 10 meters, but this range can be up to 100 meters in certain situations. Its maximum data rate is 250 kbps.

5.7.9 RFID

Radio frequency identification (RFID) is the wireless use of electromagnetic fields to identify objects. Short-range RFID is about 10cm, but long-range can go up to 200m. This protocol was designed specifically so devices without batteries could send a signal. In most systems, one side

of an RFID system is powered, creating a magnetic field, which induces an electric current in the chip. This creates a system with enough power to send data wirelessly over and over again. Because of this, RFID tags are used for shipping and tracking purposes.

5.7.10 SigFox

SigFox is a global loT network operator. It uses differential binary phase-shift keying (DBPSK) in one direction and Gaussian frequency shift keying (GFSK) in the other direction. SigFox and their partners set up antennas on towers (like a cell phone company) and receive data transmissions from devices such as parking sensors or water meters.

There are many different protocols and industry standards that are specially designed for IoT or can be used for it, such as the few mentioned above and others like Wi-Fi WebSockets, Zigbee, LORA, Simple RF, XMPP, RFID, NFC, etc. Yet, one's choice should be based on the project requirements and the constraints of the application you are thinking of developing. The possibilities in the lot space are endless.

5.8 IOT software

IOT software addresses its key areas of networking and action through platforms, embedded systems, partner systems, and middleware. These individual and master applications are responsible for data collection, device integration, real-time analytics, and application and process extension within the IOT network. They exploit integration with critical business systems (e.g., ordering systems, robotics, scheduling, and more) in the execution of related tasks.

5.8.1 Data Collection

This software manages sensing, measurements, light data filtering, light data security, and aggregation of data. It uses certain protocols to aid sensors in connecting with real-time, machine-to-machine networks. Then it collects data from multiple devices and distributes it in accordance with settings. It also works in reverse by distributing data over devices. The system eventually transmits all collected data to a central server.

5.8.2 Device Integration

Software supporting integration binds (dependent relationships) all system devices to create the body of the IOT system. It ensures the necessary cooperation and stable networking between devices. These applications are the defining software technology of the IOT network because without them, it is not an IOT system. They manage the various applications, protocols, and limitations of each device to allow communication.

5.8.3 Real-Time Analytics

These applications take data or input from various devices and convert it into feasible actions or clear patterns for human analysis. They analyze information based on various settings and designs in order to perform automation-related tasks or provide the data required by industry.

5.8.4 Application and Process Extension

These applications extend the reach of existing systems and software to allow a wider, more effective system. They integrate predefined devices for specific purposes such as allowing certain mobile devices or engineering instruments access. It supports improved productivity and more accurate data collection.

Chapter 6

System Design Specification

6.1 Design Approach

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system.

6.2 Overview

This chapter overall explains about the system design for Arduino powered Autonomous vehicle with lane tracking and object detection capabilities. This chapter also includes the Context flow diagram and Data Flow Diagram that follow the user requirement.

6.3 Context Flow Diagram

Context flow diagram is a top-level data flow diagram. It only contains one process node that generalize the function of the entire system in relationship to external entities. In context diagram the entire system is treated as a single process and all its inputs, outputs, sinks and sources are identified and shown.

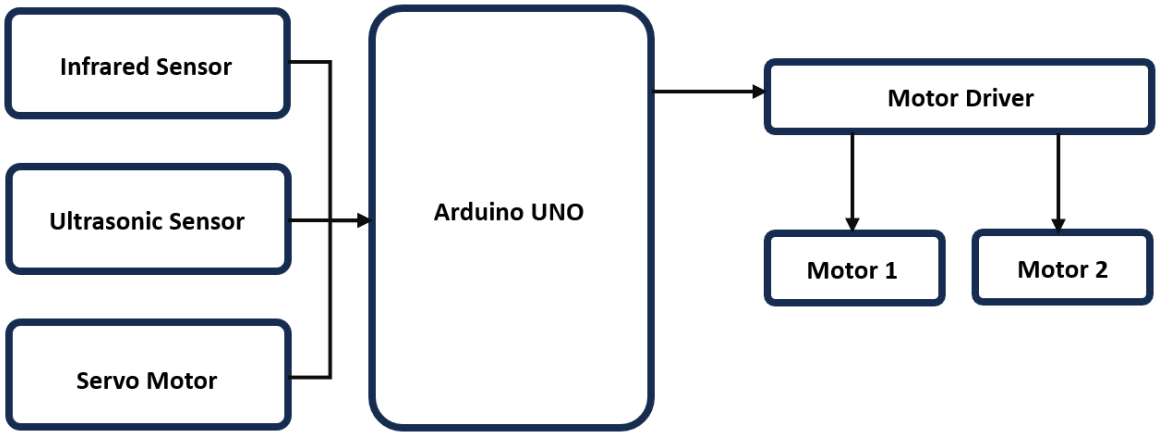


Figure 6.3 Context flow diagram of Arduino Powered Autonomous Vehicle

6.3.1 DFD (Data Flow Diagram)

Data Flow Diagram is a graphical representation of a system or a portion of the system. It consists of data flows, process, sources and sink and stores all the description through the use of easily understandable symbols.

DFD is one of the most important modelling tools. It is used to model the system, components that interact with the system, uses the data and information flows in the system.

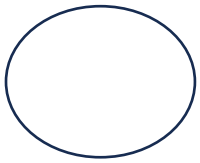
DFD shows the information moves through the and how it is modified by a series of transformations. It is a graphical technique that depicts information moves from input or output.

DFD is also knows as bubble chart or Data Flow Graphs. DFD may be used to represent the system at any level of abstraction. DFD's may partition into a level that represents increasing information flows and functional details.

Rules Regarding DFD Construction:

- A process cannot have only outputs.
- A process cannot have only inputs.
- The inputs to a process must be sufficient to produce the outputs from the process.
- All data stores must be connected to at least one process.
- All data stores must be connected to a source or sink.
- A data flow can have only one direction of flow. Multiple data flows to and/or from the same process and data store must be shown by separate arrows.
- If the exact same data flows to two separate arrows, it should be represented by a forked arrow.
- Data cannot flow directly back into the process it has just left. All data flows must be named using a noun phrase.

6.3.2 DFD Symbols

Name	Notation	Description
Process		A process transforms incoming data flow into outgoing data flow. The processes are shown bynamed circles.
Datastore		Data stores are repositories of data in the system. They are sometimes also referred to as files.
Dataflows		Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.
External Entity		External entities are objects outside the system with which the system communicates. External Entities are sources and destinations of the system's inputs and outputs.

6.3.3 Data Flow Diagram

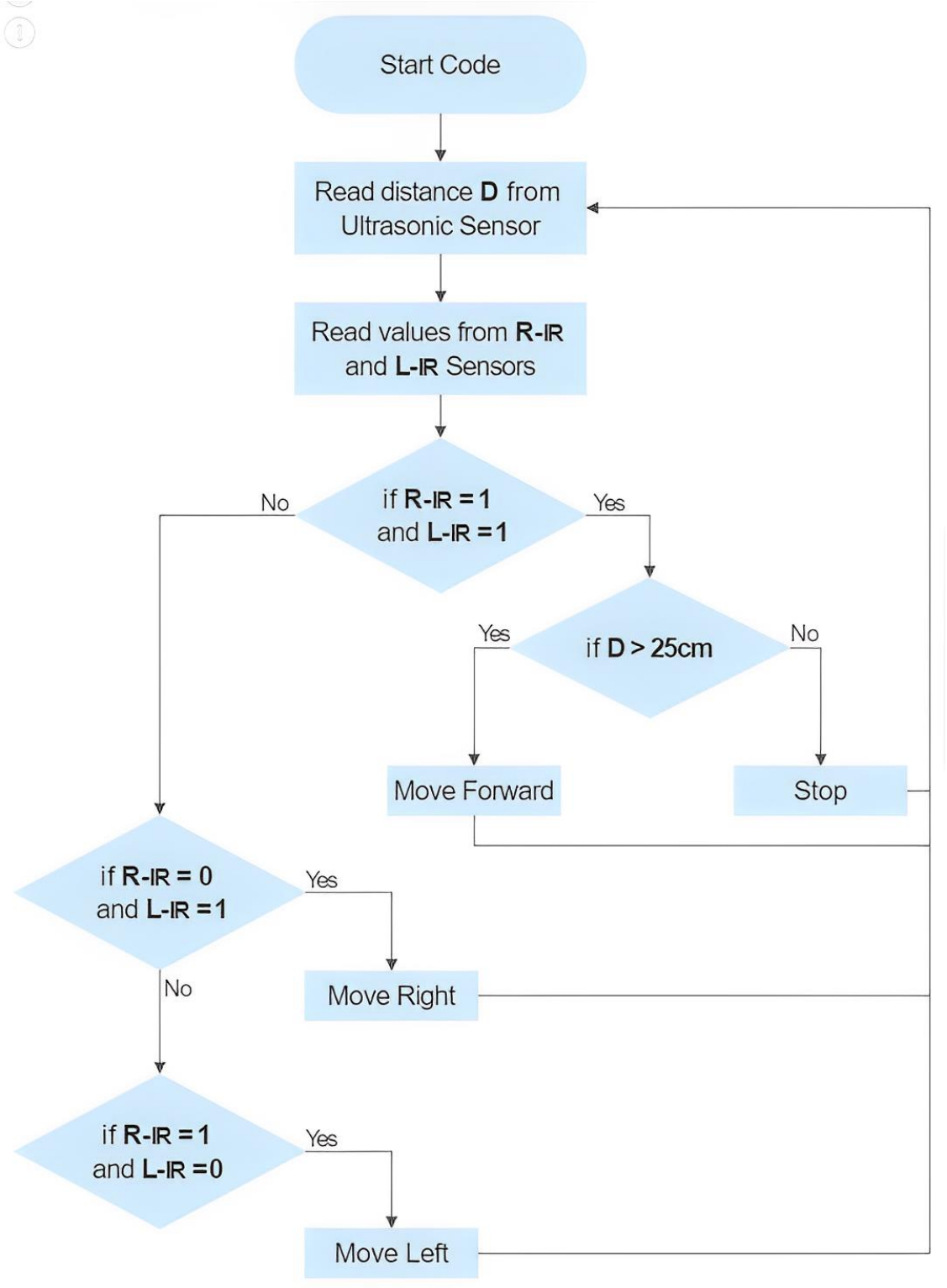
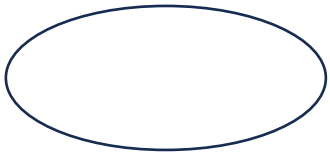
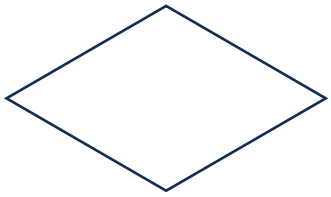


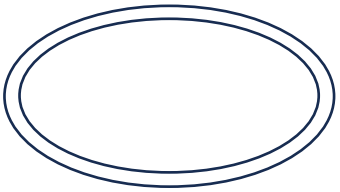
Figure 6.3.3 Data Flow Diagram of Arduino Powered Autonomous Vehicle

6.4 Entity-Relationship Diagram

The basic objective of the ER model representation is an entity which is a "thing" in a real world with an independent existence. Entities are physical items or aggregations of data items that are important to the business we analyze or to the system; we intend to build. An entity represents an object defined within the information system about which you want to store information. Entities are named as singular nouns and are shown in rectangles in an ER-Diagram. Each entity is described by several attributes; individual instances of an entity will have different attribute values.

6.4.1 ER-Diagram Symbols

Name	Notation	Description
Entity		It may be an object with the physical existence or conceptual existence. It is represented by a Rectangle.
Attribute		The properties of the entity can be a attribute. It is represented by a Ellipse.
Relationship		Whenever an attribute of one entity refers to another entity, some relationship exists. It is represented by a Diamond.
Link		Lines link attributes to entity sets and entity sets to relation
Derived Attribute		Dashed ellipse denote derived attributes.

Key Attribute	<u>_____</u>	An entity type usually has an attribute whose values are distinct for each individual entry in the entity set. It is represented by a Underlined word in ellipse.
Multivalued Attribute		Attributes that have different numbers of values for a particular attribute. It is represented by a Double ellipse represents multi-valued attributes.
Cardinality Ratio	**1 : 1** **1 : M** **M : 1** **M : M**	It specifies the maximum number of relationships instances that an entity can participate in. There are four cardinality ratios.

6.5 Circuit Diagram

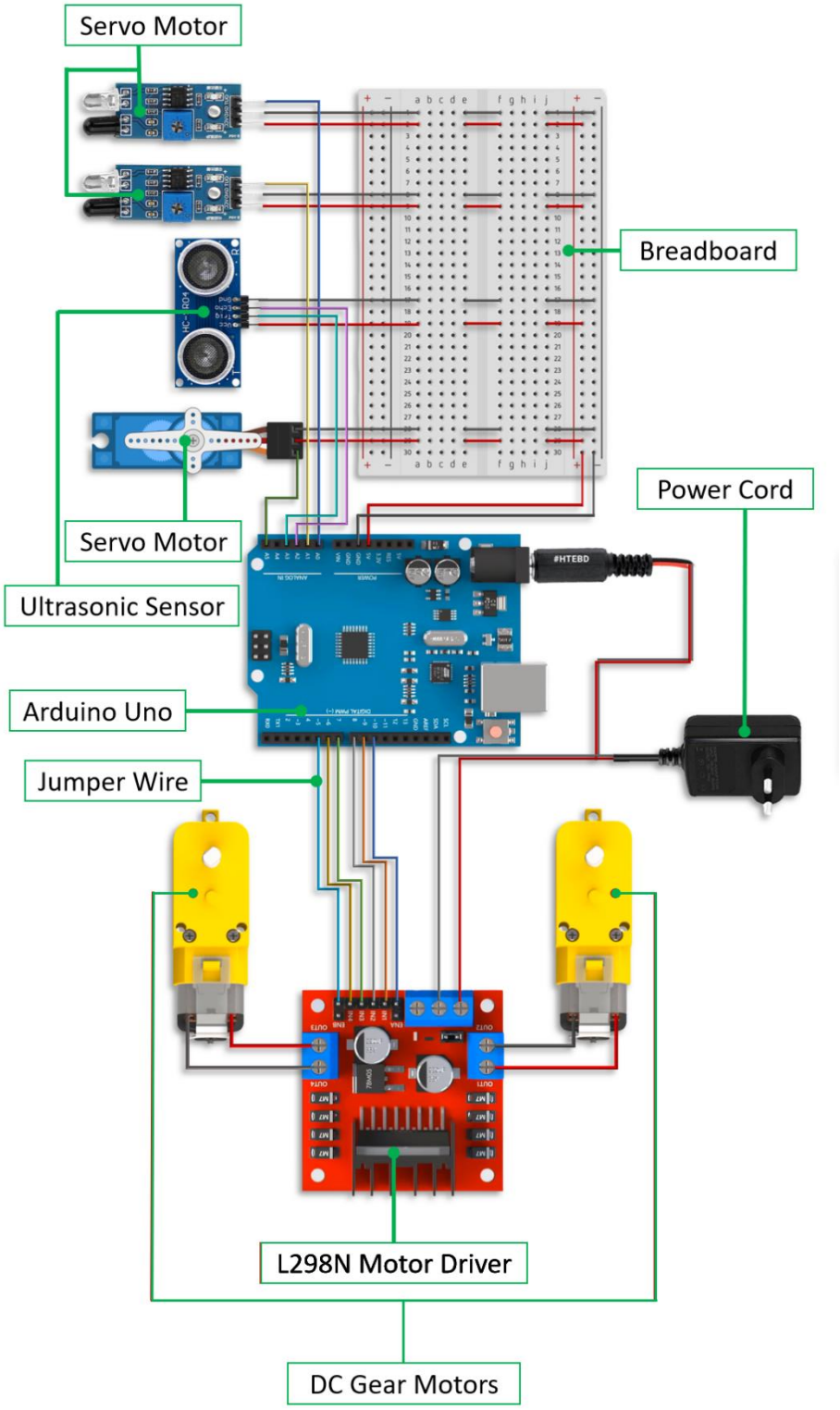


Figure 6.5 Connection diagram of Arduino Powered Autonomous Vehicle

Chapter 7

Source Code

```
#define EnableR 10
#define LowR 9
#define HighR 8
#define LowL 7
#define HighL 6
#define EnableL 5
#define Left_IR A0
#define Right_IR A1
#define echo A2
#define trigger A3

int Obj_Distance_Limit = 25;
int distance_Left;
int distance_Front;
int distance_Right;

void setup()
{
 Serial.begin(9600);
pinMode(Right_IR, INPUT);
pinMode(Left_IR, INPUT);
pinMode(echo, INPUT);
pinMode(trigger, OUTPUT);
pinMode(EnableR, OUTPUT);
pinMode(LowR, OUTPUT);
pinMode(HighR, OUTPUT);
pinMode(LowL, OUTPUT);
```

```
pinMode(HighL, OUTPUT);
pinMode(EnableL, OUTPUT);
analogWrite(EnableR, 60);
analogWrite(EnableL, 60);
pinMode(servo, OUTPUT);

distance_Front = Ultrasonic_read();
delay(500);
}

void loop()
{
distance_Front = Ultrasonic_read();
Serial.print("D F=");
Serial.print(distance_Front);
Serial.println("cm");
if ((digitalRead(Right_IR) == 1) && (digitalRead(Left_IR) == 1))
{
if (distance_Front > Obj_Distance_Limit)
{
forward();
}
else
{
Stop();
}
}
else if ((digitalRead(Right_IR) == 0) && (digitalRead(Left_IR) == 1))
{
turnRight();
}
```

```
else if ((digitalRead(Right_IR) == 1) && (digitalRead(Left_IR) == 0))
{
  turnLeft();
}
delay(10);
}

long Ultrasonic_read()
{
digitalWrite(trigger, LOW);
delayMicroseconds(2);
digitalWrite(trigger, HIGH);
delayMicroseconds(10);
long time = pulseIn(echo, HIGH);
return time / 29 / 2;
}

void forward()
{
digitalWrite(LowR, LOW);
digitalWrite(HighR, HIGH);
digitalWrite(LowL, HIGH);
digitalWrite(HighL, LOW);
}

void backward()
{
digitalWrite(LowR, LOW);
digitalWrite(HighR, HIGH);
digitalWrite(LowL, HIGH);
digitalWrite(HighL, LOW);
```

```
}

void turnRight()
{
digitalWrite(LowR, LOW);
digitalWrite(HighR, LOW);
digitalWrite(LowL, HIGH);
digitalWrite(HighL, LOW);
}

void turnLeft()
{
digitalWrite(LowR, LOW);
digitalWrite(HighR, HIGH);
digitalWrite(LowL, LOW);
digitalWrite(HighL, LOW);
}

void Stop()
{
digitalWrite(LowR, LOW);
digitalWrite(HighR, LOW);
digitalWrite(LowL, LOW);
digitalWrite(HighL, LOW);
}
```

Chapter 8

Testing

The project involves testing an Arduino-powered autonomous vehicle equipped with two IR sensors for lane following and an ultrasonic sensor for object detection. The testing phase focuses on assessing the accuracy and reliability of the sensors, as well as the vehicle's ability to autonomously navigate and avoid obstacles.

8.1 Test Cases

The test cases for the Arduino-powered autonomous vehicle with lane following using 2 IR sensors and object detection with an ultrasonic sensor include verifying accurate lane detection, successful lane following, obstacle detection and avoidance, reliable distance measurement using the ultrasonic sensor, and ensuring proper functioning of the vehicle's autonomous navigation system.

8.1.1 Vehicle on a straight lane

The vehicle is tested on a straight lane to assess its ability to maintain lane position, detect obstacles accurately, and respond accordingly.

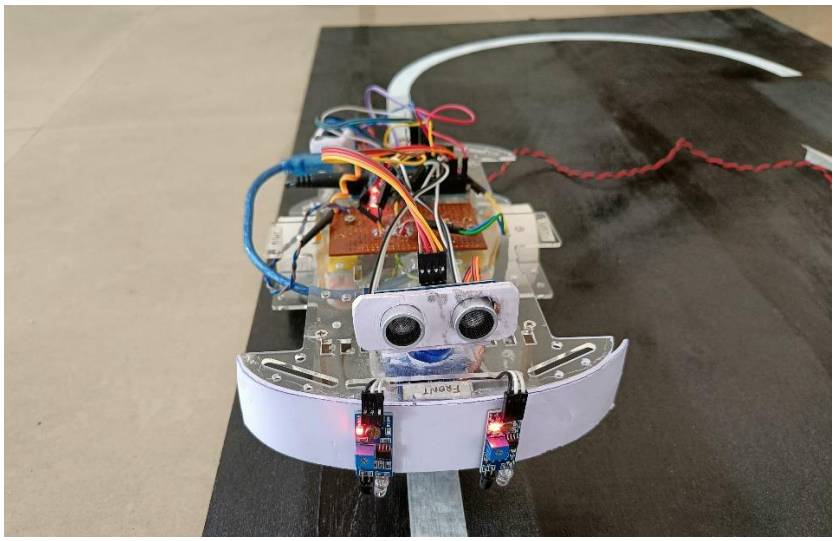


Figure 8.1.1 Vehicle on a straight lane

8.1.2 Vehicle on a curve

The test involves assessing the vehicle's ability to accurately navigate a curved path while maintaining its position within the designated lane and detecting obstacles along the way.

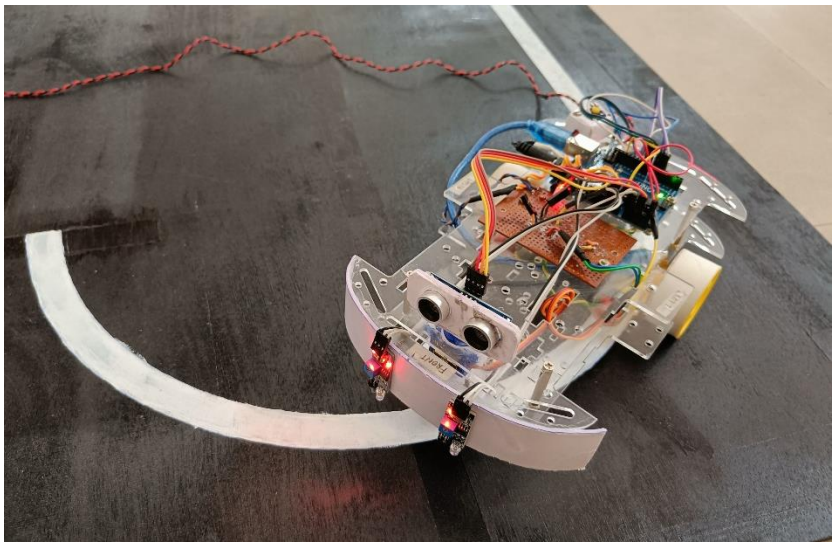


Figure 8.1.2 Vehicle on a curve

8.1.3 Vehicle Steering right

The objective is to verify the vehicle's ability to steer right accurately and safely by interpreting data from the sensors and adjusting the steering mechanism accordingly.

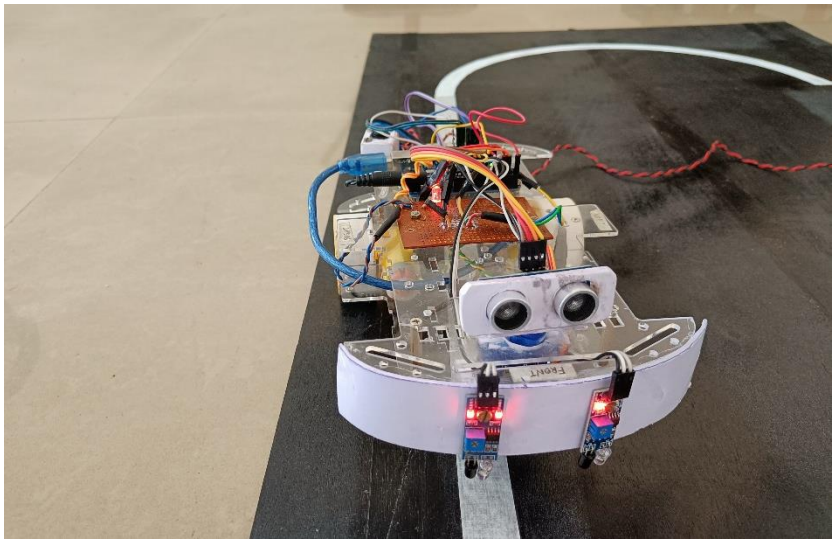


Figure 8.1.3 Vehicle Steering right

8.1.4 Vehicle Steering left

The objective is to verify the vehicle's ability to steer left accurately and safely by interpreting data from the sensors and adjusting the steering mechanism accordingly.

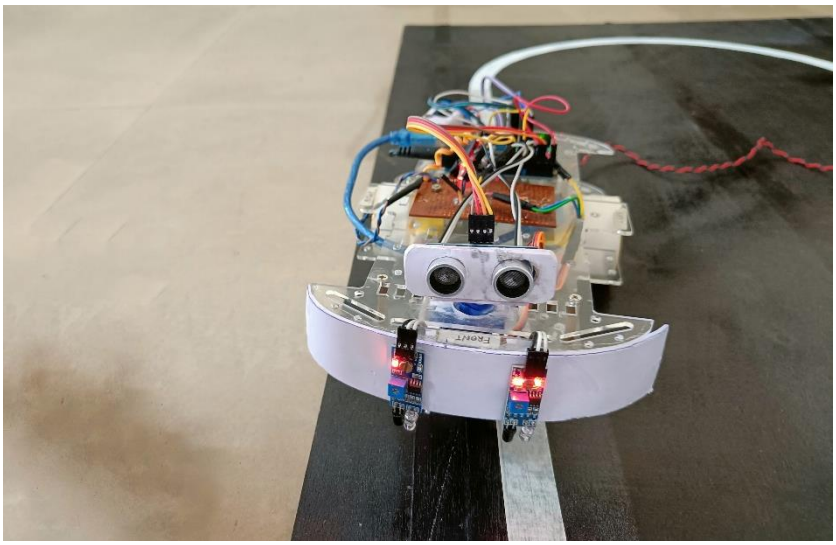


Figure 8.1.4 Vehicle Steering left

8.1.5 Obstacle Detection

The test verifies the vehicle's ability to detect obstacles accurately and take appropriate actions, ensuring safe navigation in real-world scenarios.

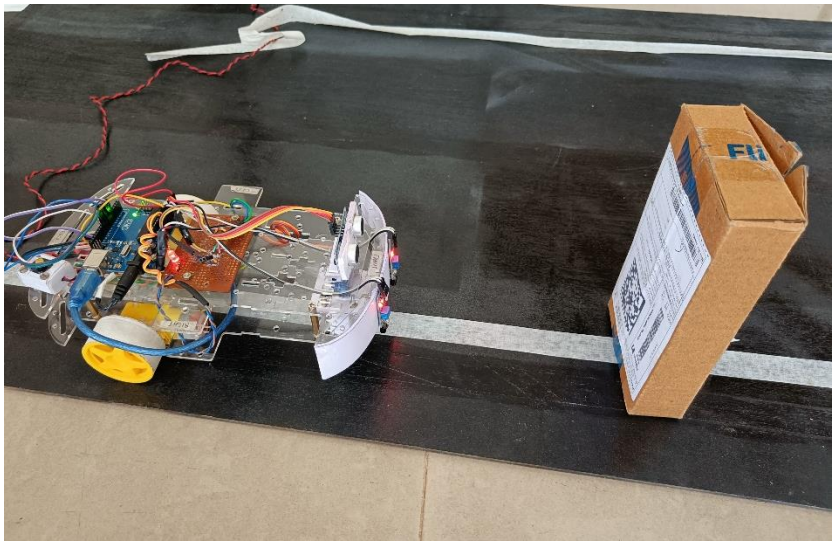


Figure 8.1.5 Obstacle Detection

Chapter 9

Conclusion

In conclusion, this project successfully demonstrates the potential of integrating Arduino, infrared sensors, and ultrasonic sensors in intelligent transportation systems. The implemented functionalities of lane detection, tracking, and obstacle avoidance contribute to improved safety and efficiency in real-world applications. The project provides a basis for further research and development in autonomous navigation and obstacle detection, benefiting the automotive industry. While there are some limitations and challenges to address, such as environmental adaptability and sensor accuracy, future enhancements like machine learning integration and advanced sensor fusion hold promise for even greater advancements. Overall, this project showcases the possibilities of leveraging technology to create smarter and safer vehicles for the future.

Chapter 10

Future Enhancement

The future of autonomous vehicles is an exciting field that holds tremendous potential for innovation and advancement. With the integration of Arduino-powered systems, lane tracking using IR sensors, and obstacle detection with ultrasonic sensors, we can further enhance the capabilities of autonomous vehicles. These enhancements pave the way for safer, more efficient, and intelligent Self-Driving vehicles.

- Self-Driving capability
- Traffic Signs identification and detection
- Integration of Machine Learning
- Advanced Sensor fusion like LiDAR
- Vehicle-to-Vehicle Communication
- Energy Efficiency Optimization

Chapter 11

Bibliography

- https://www.arduino.cc/en/software
- https://forum.arduino.cc/
- https://www.youtube.com/watch?v=4PQgjjOqJa4
- https://www.youtube.com/watch?v=zq51oZMzyP0&t=235s
- https://www.youtube.com/watch?v=vf2lW4LkmMQ
- https://www.tinkercad.com/