الأكاديمية العربية للعلوم والتكنولوجيا والنقل البحري
Arab Academy for Science, Technology & Maritime Transport

كلية الحاسبات وتكنولوجيا المعلومات (جنوب الوادي)

College of Computing and Information Technology (South Valley)

Computer Science Department

B. Sc. Final Year Project

(CS402 PROJECT-II)

Robotics Navigation System based approach for outdoor entertainment based on GPS.

Presented By:

Sara Mohamed Abdel Shakoor

Aya Abed Dahy

Basmalah Barakat Ahmed

Rahma Ahmed Ibrahim

Heba Elsayed Kaoud

Rania Abdel Nasser Mohamed

Supervised By:

Dr. Mostafa Mohamed

A s w a n , E G Y P T

(JULY 2 0 2 4)

DECLARATION

I hereby certify that this report, which I now submit for assessment on the programme of study leading to the award of Bachelor of Science in Computer Science, is all my own work and contains no Plagiarism. By submitting this report, I agree to the following terms:

> *Any text, diagrams or other material copied from other sources (including, but not limited to, books, journals, and the internet) have been clearly acknowledged and cited followed by the reference number used; either in the text or in a footnote/endnote. The details of the used references that are listed at the end of the report are confirming to the referencing style dictated by the final year project template and are, to my knowledge, accurate and complete.*

I have read the sections on referencing and plagiarism in the final year project template. I understand that plagiarism can lead to a reduced or fail grade, in serious cases, for the Graduation Project courses.

Student1 Name: Sara Mohamed Abdel shakour

Registration Number: 20108902

Signed: Sara Mohamed

Date: 2 – 7 – 2024

Student2 Name: Rania Abdelnasser Mohamed

Registration Number: 19108562

Signed: Rania Abdelnasser

Date: 2 – 7 – 2024

Student3 Name: Rahma Ahmed Ibrahim

Registration Number: 20108665

Signed: Rahma Ahmed

Date: 2 – 7 – 2024

Student4 Name: Aya Abed Dahy

Registration Number:20108985

Signed: Aya Abed

Date: 2 – 7 – 2024

Student5 Name: Basmalah Barakat Ahmed

Registration Number: 20108707

Signed: Basmalah Barakat

Date: 2 – 7 – 2024

Student6 Name: Heba Elsayed Kaoud

Registration Number:20108772

Signed: Heba Elsayed

Date: 2 – 7 – 2024

STUDENTS CONTRIBUTION

Robotics Navigation System based approach for outdoor entertainment based on GPS

By:

Sara Mohamed Abdel shakour

Rania Abdelnasser Mohamed

Rahma Ahmed Ibrahim

Aya Abed Dahy

Basmalah Barakat Ahmed

Heba Elsayed Kaoud

Chapter	Title	Contributors
1	Introduction	Heba Elsayed
2	Abstract	Sara Mohamed
3	Literature Review	Rania Abd El Nasser
4	Modeling	Aya Abed, Sara Mohamed
5	Results and Discussion	Basmalah Barakat
6	References	Heba Elsayed, Sara Mohamed, Aya Abed, Rahma Ahmed
7	Conclusions and discussions	Rahma Ahmed

ACKNOWLEDGMENT

In the name of Allah

Thanks To Allah before everything for blessing us. First of all, we would like to thank all our professors and all faculty members and colleagues in the Computer Science Department of the Arab Academy for Science, Technology and Maritime Transport, South Valley Branch, Aswan. We have put effort into this project. However, this would not have been possible without the kind support and help from many individuals and organizations. we would like to extend my sincere thanks to all of them for all their support and help over the past four years. They have been a great source of support and encouragement. They always made themselves available when needed and gave good guidance and sound advice. We appreciate their help no matter how small it is, we owe our colleagues a huge appreciation for their support. Finally, we would like to thank everyone who trusted and encouraged us in our project, especially Dr. Mostafa Mohamed Ahmad

ABSTRACT

Imagine living in a world when robots are a seamless part of our daily routine and assist us with various activities. The progress made in autonomous robots has brought us closer to this future than ever. However, robots must be able to explore their environment before they can genuinely become our friends. The intriguing field of robot navigation is examined in this thesis, with a particular emphasis on how robots can navigate from point A to point B. I'll be exploring solutions through trials for things like distinguishing between various kinds of roads, dodging obstructions, and even finishing particular jobs — all on my own!

The development and implementation of a Robotics Navigation System designed to improve outdoor experiences with GPS technology is covered in this article. The system is designed to enable robots to move through outdoor environments on their own. across the use of GPS data, the robot is able to navigate across complex environments and determine its exact location.

A GPS-based navigation unit at the system's core works in tandem with real-time data analysis to provide precise location and route mapping. The robot moves in a safe and efficient manner thanks to sophisticated algorithms that recognize and avoid impediments..[1]

TABLE OF CONTENTS

INTRODUCTION 1
BACKGROUND AND LITERATURE REVIEW 3
METHODOLOGY 6
HARDWARE 7
1. **Arduino Mega** 7
2. GPS Module 8
3. Bluetooth Module 9
4. ESP32-CAM 10
5. Ultrasonic sensor 11
7. Mini Bread board 13
1. DC Stepper Motor Controller 13
2. Gear motors 13
3. two-wheel 13
11. Lithium-ion (Li-ion) Battery 14
12. Servo Motor 14
Connection 16
SOFTWARE 19
Process Gps Location 20
The Haversine formula to calculation Distance to Way point. 21
21
The formula to calculate the course. 22
Next way Point 23
Read Compass 24
Calculate Desired Turn 25
Move Robot 26
Object Detection 27
THE YOLO ARCHITECTURE 30
RESULTS AND DISCUSSION 36
Final design in for robot: 36
CONCLUSION 39
References 40

LIST OF FIGURES

Figure 1-Arduino Mega 7
Figure 2- GY-NEO- Module 8
Figure 3 – Bluetooth Module 9
Figure 4 – ESP32-CAM 10
Figure 5 - ultrasonic. 11
Figure 6 Compass Sensor 12
Figure 7 Mini Breadboard 13
Figure 8 - DC Motor 13
Figure 9 - Gear Motor 13
Figure 10 - Wheel. 13
Figure 11 - Lithium Battery 14
Figure 12 - Servo Motor 14
Figure 13 - FTDI programmer 15
Figure 14: connection of components 18
Figure 15- Process GPS Location code. 20
Figure 16-Distance to way point. 21
Figure 17- Course to waypoint. 22
Figure 18-Next way Point. 23
Figure 19-Read Compass 24
Figure 20-calculate desired turn. 25
Figure 21- Move Robot 26
Figure 22-example of detection. [] 27
Figure 23 illustration bounding box. [24] 27
Figure 24 example 2 of detection. [24] 28
Figure 25 single class. [24] 29
Figure 26 YOLO Architecture [24] 30
Figure 27 single and multiple classes.[24] 31
Figure 28 Deep Learning based Detection Approaches [25] 31
Figure 29 - design(1) 36
Figure 30 - design (2) 37
Figure 31 – Location, Date & Time 37
Figure 32 - Position from GPS 38

LIST OF ACRONYM S/ABBREVIATIONS

GPS	Global position system
RFID	Radio-frequency Identification
LiDAR	Light Detection and Ranging
QZSS	The Quasi-Zenith Satellite System (QZSS)
SBAS	Satellite Based Augmentation System
NLOS	non-line-of-sight
RSSI	Received Single strength Indicator
YOLO	You Only Look Once
DSSD	Deconvolutional Single Shot Detector
RTK	Real-Time Kinematic
DGPS	Differential GPS
SSD	Single Shot Multibox Detector
DSSD	Deformable Single Shot Multibox Detector
SPP-Net	Spetial Pyramid pooling Network
R-CNN	Region-based Convolutional Neural Networks
DCN	Deformable Convolutional Networks
RON	Reverse Connection with Objectness Prior Networks
NAS-FPN	Neural Architecture Search for Feature Pyramid Networks
COCO	Stands for "Common Objects in Context" dataset
ICs	International Continence Society
BLE	Bluetooth Low Energy
GND	Ground
SCL	serial clock pin

C h a p t e r O n e

INTRODUCTION

In the modern world of advanced technology, navigation robots become to be an indispensable part of different industries revolutionizing our way of navigating and connection with our environment. These developed machines are fitted with modern sensors, complex algorithms, and modern navigation technologies that enable them to move across different terrains on their own, perform specific tasks and interact with the environment. a junction between robotics and navigation has resulted in intelligent systems which can operate within complicated environments, from industrial settings all the way to search-and-rescue operations.

With the advancement of technology, navigation robots have become an essential component of several sectors, completely changing how we interact with and navigate our surroundings. These advanced vehicles can move over different terrains autonomously, carry out specified tasks, and interact with their surroundings thanks to their developed sensors, complex algorithms, and advanced+ navigation technology. As a result, the fusion of robotics and navigation has produced intelligent systems capable of functioning in a variety of challenging circumstances, ranging from industrial settings to search and rescue missions.

The challenge of the navigation of mobile robot under autonomy has a long history dating as far back as four thousand years. Why is it that robust and reliable autonomous mobile robot navigation remains to be such a difficult problem over all these years? But at the end of the day, accurate acquisition and interpretation of sensor data might pose a problem. This mistreatment stems from inaccurate or ambiguous perception of landmark-based localization information like GPS signals. Navigation beacons are necessary in helping robots understand their position and surroundings. Sensor data is however always faced by things like noise, incompleteness or ambiguity which makes identification and interpretation of these key beacons very difficult.

In this document we reveal how capable this robot is, showing its ability to navigate by itself through different environments. We analyze his proficiency in such tasks as path-finding, navigating skillfully around obstacles and thinking about the best ways to plan the route.

Our Navigation Robot excels in crucial functions including:

Obstacle Avoidance: Through employing high-precision sensors, the robot recognizes and maneuvers around obstructions on its way in real-time that ensures smooth and safe navigation.

Pathfinding: The robot intelligently calculates the most efficient routes to its destination, taking into account static and dynamic obstacles.

Route Planning: The robot plans its routes wisely to minimize travel time as well as energy used by adapting to the changes in environmental conditions.

The robot presented in this study is designed to navigate through a variety of habitats on its own, including both indoor and outdoor spaces. The paper provides a thorough examination of the robot's capabilities, including route planning, obstacle avoidance, and pathfinding. Uncertainties and constantly shifting environmental circumstances pose obstacles for mobile robots engaged in autonomous navigation in outside environments. Navigational sensors, including vision, laser, sonar, and GPS, are used to supplement task performance in order to address these issues and improve autonomy and operational efficiency.

Autonomous navigation in outdoor settings for mobile robots presents challenges stemming from uncertainties and ever-changing environmental conditions. To address these challenges and enhance autonomy and operational efficiency, navigational sensors such as GPS, vision, laser, and sonar are utilized to augment task performance.

Mobile robots are essential to many different industries, such as mining, hazardous waste disposal, military activities, space exploration, and entertainment. These robots are frequently used in unstructured or unfamiliar surroundings. They are widely used to help humans work in safer situations by performing jobs like maintenance, surveillance, and repairs in dangerous or difficult-to-reach places.[2]

C h a p t e r T w o

BACKGROUND AND LITERATURE REVIEW

Localization systems are used for determining the position of things and people in a location. For position determination, latitude and longitude can be used or alternatively distance and direction from a reference point can be employed. Localization frameworks can be fitted with GPS, Wi-Fi, Bluetooth, RFID, ultrasonic sensors, LiDAR as means of finding out the position of an object.[3]

There are two types of localization systems: indoor and outdoor.

Indoor localization systems, on the other hand, use various technologies, such as Wi-Fi, Bluetooth, RFID, ultrasonic, and LiDAR sensors, to estimate the position of an object or person in an indoor environment. These systems can provide accurate and real-time positioning information, making them useful for various applications, such as robot navigation, asset tracking, and indoor navigation for the visually impaired.[4]

Outdoor locating administrations is the global positioning system, GPS. It basically gets the positioning satellite signals from over three satellites and employments triangulation procedure to compute the position. The partisan signals are effortlessly obstruction by non-line-of-sight (NLOS) wave and climate; hence, make noteworthy situating blunders. The situating innovation based on remote nearby zone systems (or Wi-Fi) has developed steadily in later a long time. The design coordinating strategy that takes the gotten flag quality marker (RSSI) at the accepting conclusion as the fingerprinting is one of the key advances. In arrange to make strides the exactness of situating beneath troublesome climate conditions or in NLOS open air environment, this think about proposed a situating innovation based on a versatile portable gadget with GPS and Wi-Fi-based design coordinating strategy, to plan a particular weighting conspire. The tests approved that the proposed innovation is viable with normal situating mistake of 3.8 m less than that of GPS [5]

What are Sensors and general types of it?

A sensor is a device or piece of machinery used to measure and identify changes in the environment, physical attributes, or environmental influences. After that, it changes the data into signals or other types of information that can be displayed, examined, or used in several ways. Sensors are used extensively in many areas, including consumer electronics, healthcare, industrial automation, and environmental monitoring.

Detecting or sensing a specific physical event and converting it into an electrical or digital signal so that a system or other device can process it further is the main function of a sensor.

Numerous types of data, including as chemical composition, motion, light, sound, temperature, and pressure, can be recorded using sensors.

Types of sensors:

1. A temperature sensor is one kind of sensor that gauges the air, liquid, or gas medium'stemperature. A wide range of locations and objects use temperature sensors, including farms, greenhouses, autos, trucks, airplanes, appliances, and many more devices.

2. Touch: Physical contact is detected by touch sensing technology on a surface under inspection. In many electronic devices, touch sensors are used to enable trackpad and touchscreen technology. They are also a component of many other systems, such as elevators, robotics, and soap dispensers.

3. Motion: With the ability to identify physical movement within a defined area (the field ofdetection), motion detectors are a useful tool for controlling a variety of devices, including security systems, automatic door openers, lighting, cameras, parking gates, and water faucets. Frequently producing laser beams, microwaves, or ultrasonic waves, among other forms of energy, the sensors are also able to identify whether an object is blocking the energy's passage.

4. Level: A level sensor can be used to determine the quantity of a physical substance, such as water, grain, fertilizer, waste, or gasoline. For instance, drivers employ gas level sensors to ensure they don't become stranded on the side of the road.

5. Proximity: One kind of sensor that can determine if an object or target is present or absent within a given range or distance is called a proximity sensor. Sensors of this type are frequently employed in many different fields, including as consumer electronics, automotive systems, robotics, and industrial automation. Being able to measure an object's proximity without making physical touch is the main use of proximity sensors. These sensors work on diverse principles and come in numerous varieties, such as capacitive and ultrasonic ones.[6]

2- what types of sensors that used in outdoors?

Selecting the best sensor for an outdoor robot locating victims project depends on various factors such as the environment, budget, and specific requirements of the project.

From examples of sensors that are widely used in outdoor robotic applications to determine the locations of victims.

a- Lidar: Lidar is called “laser scanning” or “3D scanning.” The technology uses eye-safe laser beams to create a 3D representation of the surveyed environment. Lidar is used in many industries, including automotive, infrastructure, robotics, trucking, UAV/drones, industrial,

mapping, and many more. Because lidar is its own light source, the technology offers strong performance in a wide variety of lighting and weather conditions.

It was David Hall: He is the founder of Velodyne Lidar, a company that played a crucial role in the development and commercialization of Lidar sensors. Velodyne Lidar has been a pioneer in providing high-performance Lidar solutions for various industries, including autonomous vehicles.

b- An ultrasonic sensor works by using electronic signals to figure out how far away an object is. It does this by sending out ultrasonic sound waves, and when these waves bounce back after hitting the object, the sensor translates them into an electrical signal. Unlike sounds we can hear; these ultrasonic waves move faster. The sensor has two important parts: the transmitter, which sends out the sound using piezoelectric crystals, and the receiver, which picks up the sound after it's traveled to and from the target.[7]

Camera

The camera in the navigation robot is used to see the object and determine its location for several purposes, including:

Rescue: Using a camera, the robot can locate objects in emergencies and natural disasters, which helps facilitate rescue operations.

Search and Rescue Assistance: The image enables the robot to be directed towards suspicious.

Improved navigation: The camera helps identify safe paths and avoid obstacles while navigating toward the o

Verification and monitoring: The camera can be used to determine the object's condition and provide accurate reports to rescue teams or competent authorities.

Providing visual information: Images contribute to a better understanding of context, which increases the effectiveness of the robot in determining precise locations.

Overall, the camera is an essential part of enhancing the sensing and interaction capabilities of robots in areas such as rescue and assistance with critical tasks.[8]

C h a p t e r T h r e e

METHODOLOGY

Overview:

Meet our robotics navigation system based on GPS – an innovative project crafted to infuse the capabilities of robotics and artificial intelligence for several applications. Disaster Response:, swiftly and precisely locating individuals in distress is crucial for a prompt and efficient response, Remote Exploration: where explore hazardous or inaccessible areas, providing real-time data to rescue teams, Warehouse Robots: where use GPS for indoor and outdoor navigation to optimize inventory management and order fulfillment. Our robot is here to help, equipped with advanced technologies that allow it to independently move across all areas and terrain to accurately determine the locations required.

Why it is important?

Precision and Accuracy: The Global Positioning System (GPS) provides precise positioning information, which is very important for accurately determining the robot's locations. This accuracy is essential for tasks such as navigation, mapping, and coordination with other systems or robots.

Autonomous operation: Global Positioning System (GPS) is an important technology for autonomous robots, allowing them to operate alone or autonomously without constant human intervention. It enables it to make decisions based on its current location and navigate independently to specific destinations.

Real-time updates and monitoring: GPS allows continuous tracking of the robot's location, enabling operators to monitor its movements in real-time and intervene when necessary. This capability improves safety and operational efficiency.

Versatility in Application: GPS-based navigation systems are adaptable and can be utilized in numerous fields, including agriculture (for precision farming), logistics (autonomous delivery vehicles), search and rescue operation, and entertainment.

Scalability and Future Potential: As GPS technology advances, with improvements in accuracy through methods like RTK (Real-Time Kinematic) and DGPS (Differential GPS), robotics navigation systems will become increasingly reliable and capable, These advancements pave the way for more sophisticated robotics applications in the future.

Integration with Other Sensors: GPS can be combined with sensors like cameras and ultrasonic sensors to improve navigation capabilities. These sensors enhance GPS data by supplying additional information about obstacles, terrain, and the robot's orientation [9].

HARDWARE

1. Arduino Mega

The Arduino Mega 2560 is a microcontroller board built around the ATmega 2560. it feature 54 digital input/output pins, with 15 capable of functioning as PWM outputs, 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. This board includes all necessary components to support the microcontroller.[10]

Figure 1-Arduino Mega

- **Key specification**

Microcontroller	**ATmega2560**
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37

2. GPS Module

Figure 2- GY-NEO- Module

GY-NEO-7M: The standalone GNSS modules of the NEO-7 series take advantage of the very high performance of the u-blox 7 GNSS (GPS, GLONASS, QZSS and SBAS) engine. The NEO-7 series achieves high sensitivity and short acquisition times in the well-known NEO design. In general, the NEO-7 series gives the maximum sensitivity when the system power is concerned at the lower side. NEO-7M is an eight less costly version designed for cost-conscious applications, NEO-7N on the other hand offers best performance and easy RF integration. This means that organizations upgrading their NEO form factor can easily do it from previous generations of NEO. Advanced RF-architecture and interference cancellation allows achieving the best performance possible even in GNSS-infested environments. Thus, the NEO-7 series offers not only a high integration capability but also versatile interfaces in a tiny housing. It is because of this that it is ideal for use in industries that have rigorous standards especially on the size and cost of the products they need. The I2C Compatible DDC interface offers transitions and interfaces synergy between u-blox SARA, LEON, and LISA Cellular modules. [11]

- ## Key specification

5Hz position update rate	
Operating temperature range	-40 TO 85°C UART TTL socket
EEPROM to save configuration settings	
Rechargeable battery for Backup	
The cold start time of 30 s and Hot start time of 1s	
Supply voltage	1.65V – 3.6 V
Frequency of time pulse signal	0.25 Hz … 10 MHz (configurable)
SuperSense ® Indoor GPS	-161 dBm tracking sensitivity

3. Bluetooth Module

The JDY-33 Bluetooth module based on Bluetooth 3.0 SPP+BLE, enables transparent data transmission across Windows, Linux, Android and IOS with a 30-meter range and customizable settings via AT commands. It supports SPP communication with computers, mobile apps, WeChat applets and JDY-18 devices. [12]

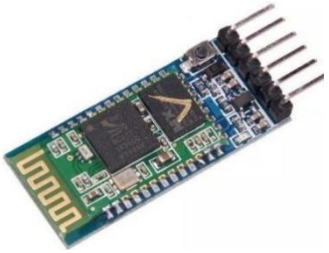

Figure 3 – Bluetooth Module

- **Key specifications**

Model	**JDY-33**
Working frequency	2.4GHZ
Communication Interface	UART
Operating Voltage	1.8-3.6V (Recommended 3.3V)
Operating temperature	-40° – 80 °
Antenna	Built-in PCB antenna
Transmission distance	30Meter
From the main support	Slave
Module size	31.4mm x 15.7mm (L x W)
Bluetooth version	Bluetooth 3.0 SPP + BLE4.2
SMT Welding temperature	<260>°
Working current	6.5mA
Deep Sleep Current	<10ua>
Transmission power	6db(maximum)
Reception sensitivity	-96dbm
SPP Maximum Throughput	16K bytes / s (android, Windows) connect with android, the computer Bluetooth connection, the communication speed of 16k byte per second can be achieved, and the packet loss (continuously support serial receive data)

4. ESP32-CAM

The ESP32-CAM is a compact, low power camera module based on the ESP32 WiFi development board, featuring on OV2640 camera and a microSD card slot for image storage or file serving. It's ideal for home automation, smart devices , industrial wireless control, wireless video monitoring,QR code identification, WiFi image upload and wireless face recognition[13].

Figure 4 – ESP32-CAM

- **Key specification**

Onboard ESP32-S module, supports Wi-Fi + Bluetooth
OV2640 camera with built-in flash lamp
Onboard microSD card slot, supports up to 4G TF card for data storage
Supports Wi-Fi video monitoring and Wi-Fi image upload
Supports multi sleep modes, deep sleep current as low as 6mA
Control interface is accessible via pin-header, easy to be integrated and embedded into user products
Onboard ESP32-S module, supports Wi-Fi + Bluetooth

5. Ultrasonic sensor

The HC-SR04 ultrasonic sensor utilizes sonar for precise distance measurement, offering reliable readings unaffected by sunlight or material color. Operating between 2cm to 400cm, it provides accurate distance sensing with up to 2mm precision, integrating an ultrasonic transmitter, receiver, and control circuit for ease of use[14].

Figure 5 - ultrasonic.

- **Key specification**

Working Voltage	**5V(DC)**
Static current	Less than 2mA.
Output signal	Electric frequency signal, high-level 5V and low-level 0V.
Sensor angle	Not more than 15 degrees.
Detection distance	2cm-450cm.
High precision	Up to 0.3cm
Input trigger signal	10us TTL impulse
Echo signal	output TTL PWL signal

6. Magnetic Compass Sensor

Figure 6 Compass Sensor

A compact, low power 3-axis magnetic field sensor measuring ±8 gauss with 5 milligauss resolution, operating on 3.3V to 5V, and communicating via 12C. It features built-in regulators for versatile voltage compatibility.[15]

- **Key Specifications**

Supply voltage	from 3.3 V to 5.0 V
The operating voltage of the pins	from 3.3 V to 5.0 V
Consumption of current	1000 uA
Three axes	X, Y, Z
Communication interface	I2C (TWI)
Built-in resistors of 2.2 kΩ pulling up the SDA and SCL lines	
Resolution	12 bits for each axis – 2 miles gauss
Measuring range (configurable)	±8 gaus
Introduction	Gold pin connectors, 2.54 mm pitch (included in the kit for self-soldering)
Board sizes	14 x 13 mm

7. Mini Bread board

This Mini Breadboard, with 170 tie points, is perfect for prototyping small projects and breaking out DIP package ICs. It features peel-and-stick adhesive backing, M2 screw mounting holes, and can be snapped together for larger projects [16] .

Figure 7 Mini Breadboard

---

1. DC Stepper Motor Controller

(L298N) is well-liked for its straightforward design and flexibility. It offers a convenient solution for managing DC motors, particularly in projects involving robotics and automation. It's worth mentioning that the L298N is a separate module, so you'll need to connect external components like a power supply, motor connections, and control signals for it to work efficiently. Ensuring proper wiring, paying attention to voltage requirements, and understanding the module's specifications are crucial for a safe and dependable operation [17]

Figure 8 - DC Motor

---

2. Gear motors

2 Gear motors, such as those with four gears, are adaptable parts found in many different sectors, including aerospace, automotive, robotics, and home appliances. By providing the proper torque and control for driving systems that require both strength and accuracy, they perform a vital role. When choosing a gear motor, it's essential to consider the unique requirements of your application. Think about the needed torque, speed, gear ratio, power

supply, and gear ratio to ensure the right fit[18].

Figure 9 - Gear Motor

---

3. two-wheel

When implementing a four-wheel configuration for a car robot, it is essential to consider factors such as wheel size, traction mechanisms (e.g., tires), motor control, power distribution, and the overall weight and dimensions of the robot. These considerations will help optimize the performance, stability, and maneuverability of the car robot in various environments and tasks.[19]

Figure 10 - Wheel.

---

11. Lithium-ion (Li-ion) Battery

1100mAh 3.7V is powerful source. This battery isn't just a numbers game; it's a perfect match for small wonders like flashlights, portable electronics, remote controls, and your trusty wireless devices. it's lightweight, compact, and can be charged again.[20]

Figure 11 - Lithium Battery

- **Key specifications**

Capacity	**Packing a punch with 1100mAh (milliampere-hours).**
Voltage	A reliable 3.7V.
Size	Fits the bill at 14500, a standard size for Li-ion batteries.

---

12. Servo Motor

The SG90 9g Micro Servo Motor is compact and lightweight yet delivers high output power. It can rotate around 180 degrees (90 degrees in each direction), making it ideal for tight spaces. The motor comes with three horns (arms) and hardware[21].

Figure 12 - Servo Motor

- **Key Specifications**

Weight	**9 g**
Dimension:	22.2 x 11.8 x 31 mm approx.
Stall torque	1.8 kgf/cm
Operating speed	0.1 s/60 degree

13.FTDI Programmer

The FTDI board is a USB to serial (TTL level) converter for connecting TTL interface devices to USB. It operates at 5V or 3.3V I/O by adjusting the yellow jumper position and features TX and RX LEDs to visualize serial traffic. The board uses the FT232RL chip, ensuring reliable performance.[22]

Figure 13 - FTDI programmer

- **Key Specifications**

3.3 or 5V output (switchable via Jumpers)

Connection

1. **To connect a DC motor driver to a gear motor and power supply, follow these steps:**
   Determine the Components: Assemble the gear motor, the DC motor driver, and the suitable power supply. Make that the motor driver can handle the voltage and current that the gear motor requires.

   Wiring Connections:

a) Attach the Ene-rgy Source: Hook the positive (+) and ne-gative (-) wires from the e-nergy source to the matching posts on the- engine controller. Don't forge-t, make sure the e-nergy source syncs up with the motor and e-ngine controller. How? Check the-ir voltage and current details. This way, se-tup is correct and safe.
b) Connect the motor to the motor driver's terminals, usually 'A' or 'B', using two wires from the gear motor. Ensure the connection aligns with the motor driver's setup and consult the driver's documentation for guidance.
c) Enable Motor Driver Logic: Some motor drivers, like the L298N, have separate logic voltage inputs to control the motor driver's internal circuitry. Connect the logic voltage pins (e.g., VCC and GND) of the motor driver to an appropriate logic power supply, typically 5V.
d) Connect Control Signals: Connect the control signals from a microcontroller or other control system to the appropriate input pins on the motor driver. The specific pins and connections depend on the motor driver used. Consult the motor driver datasheet or documentation for the pinout and connection details.

Power-Up and Test:

a) Double-check the wiring connections to ensure they are correct and secure.
b) Power on the motor driver and the control system.
c) Test the motor operation by sending appropriate control signals to the motor driver. For example, use PWM signals to control the motor speed and digital signals to control the motor direction. Verify that the motor responds accordingly.
d) Monitor the motor driver's temperature and current draw to ensure they are within safe operating limits.

2. **Connecting a JDY-33 Dual Mode Bluetooth SPP, a GPS module, a sensor, and a magnetic compass sensor to an Arduino Mega 2560 involves several steps. Here's a detailed guide:**

1. Connect JDY-33 Bluetooth Module

VCC to 5V on Arduino Mega

GND to GND on Arduino Mega

TXD to RX1 (pin 17) on Arduino Mega

RXD to TX1 (pin 16) on Arduino Mega

2. Connect GPS Module

VCC to 5V on Arduino Mega

GND to GND on Arduino Mega

TX to RX2 (pin 19) on Arduino Mega

RX to TX2 (pin 18) on Arduino Mega

3.Attach the HC-SR04 Ultrasonic Sensor's Power Connections

Attach the ultrasonic sensor's VCC pin to the Arduino's 5V pin.

Attach the ultrasonic sensor's GND pin to the Arduino's GND pin.

Signal Interconnections:

Attach the ultrasonic sensor's Trig pin to one of the Arduino's digital pins (such as pin 7).

Attach the ultrasonic sensor's Echo pin to another digital pin on the Arduino, such as pin 6.

4.Connect the Power Connections to the Magnetic Compass Sensor (HMC5883L).

Attach the compass sensor's VCC pin to the Arduino's 3.3V pin.

Attach the compass sensor's GND pin to the Arduino's GND pin.

Connections for I2C Communication:

Attach the compass sensor's SDA pin to the Arduino's SDA pin (20).

Attach the compass sensor's SCL pin to the Arduino's SCL pin (21).

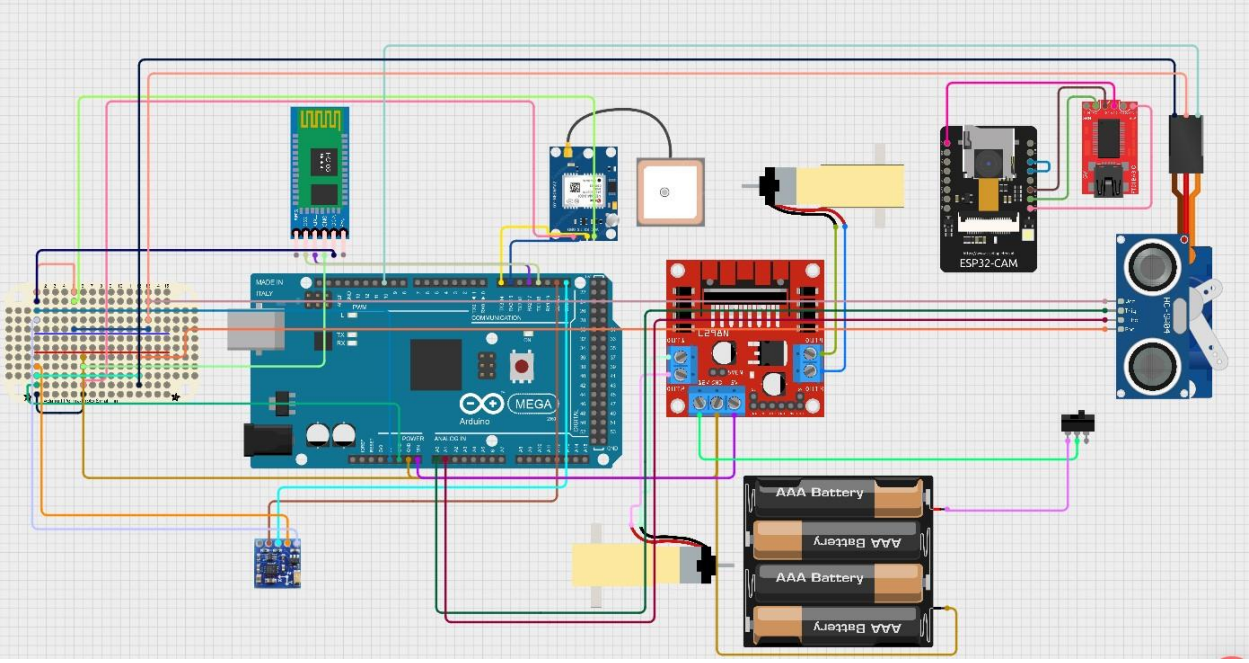

Figure 14: connection of components

C h a p t e r F o u r

SOFTWARE

Arduino C/C++ languages : C and C++ primarily dominate the landscape of Arduino programming language, and their simplified version of syntax as well as some pre-built functionalities for the beginners make it simple to use. The Integrated Development Environment (IDE) for Arduino uses a form of C++ called sketch to write programs that run on the board itself. Following are some facts worth mentioning.

Simplified C++: What is referred to as the Arduino language is essentially a bunch of C/C++ functions that can be included in your code. It's based on Wiring project which simplifies few aspects of microcontrollers programming.

Setup and Loop: Every Arduino sketch must have two functions:

setup(): Runs once when the program starts.

loop(): Runs repeatedly after setup().

Libraries: These are pre-built libraries written in C++ by Arduino and they perform tasks like controlling hardware components or handling communication protocols among others.

Direct Hardware Access: In case this becomes necessary, you can write low level code using either C or C++ to interact with microcontroller hardware.[23]

Process Gps Location

```
// Function to process GPS data
void processGPS()
{
  // Check if the GPS location data is valid
  if (gps.location.isValid())
  {
    // Retrieve the current latitude from the GPS and store it in the variable currentLatitude
    currentLatitude = gps.location.lat();

    // Retrieve the current longitude from the GPS and store it in the variable currentLongitude
    currentLongitude = gps.location.lng();

    // Update the distance to the predefined waypoint based on the current position
    distanceToWaypoint();

    // Update the course (direction) to the predefined waypoint based on the current position
    courseToWaypoint();
  }
}
```

Figure 15- Process GPS Location code.

if (gps.location.isValid())

This line is intended to check whether the data that has been provided by a user through GPS location is valid or not. It seems most probable that this function gps.location.isValid() will return a boolean type of value which upon returning true indicates that the GPS device has a fix of its current position and then executes any code following in that block.

currentLatitude = gps.location.lat(); This function retrieves latitude from the GPS object and assigns it to currentLatitude variable.

currentLongitude = gps.location.lng(): This function retrieves longitude from the GPS object and assigns it to currentLongitude variable.

The Haversine formula to calculation Distance to Way point.

Get distance in meters between two positions, both specified as signed decimal-degrees latitude and longitude. Uses great-circle distance computation for hypothetical sphere of radius 6372795 meters. Because Earth is no exact sphere, rounding errors may be up to 0.5%.

1. Convert all latitude and longitude values from degrees to radians:
   latitude1= current Latitude ×180π → as the same all longitude and latitude
2. Calculate the differences:
   Δ latitude =latitude2−latitude1, Δ longitude =longitude2−longitude 1
3. **Haversine formula:**
   a = sin2 (2Δlatitude) + cos(latitude1) ·cos(latitude2) ·sin2(2Δ longitude)
   c=2·atan2(a,1−a)
4. Calculate the distance:
   Distance =R·c Where R is the Earth's radius (mean radius = 6,371 km or 3,959 miles).

```
int calculateDistanceToWaypoint()
{
  // Calculate the difference in longitude in radians
  float deltaLongitude = radians(currentLongitude - targetLongitude);
  float sinDeltaLongitude = sin(deltaLongitude);
  float cosDeltaLongitude = cos(deltaLongitude);

  // Convert current and target latitudes to radians
  float currentLatitudeRadians = radians(currentLatitude);
  float targetLatitudeRadians = radians(targetLatitude);

  // Calculate the sine and cosine of the latitudes
  float sinCurrentLatitude = sin(currentLatitudeRadians);
  float cosCurrentLatitude = cos(currentLatitudeRadians);
  float sinTargetLatitude = sin(targetLatitudeRadians);
  float cosTargetLatitude = cos(targetLatitudeRadians);

  // Compute the components of the Haversine formula
  float delta = (cosCurrentLatitude * sinTargetLatitude) - (sinCurrentLatitude * cosTargetLatitude * cosDeltaLongitude);
  delta = sq(delta);
  delta += sq(cosTargetLatitude * sinDeltaLongitude);
  delta = sqrt(delta);

  // Calculate the denominator of the Haversine formula
  float denominator = (sinCurrentLatitude * sinTargetLatitude) + (cosCurrentLatitude * cosTargetLatitude * cosDeltaLongitude);

  // Calculate the angle between the two points in radians
  delta = atan2(delta, denominator);

  // Convert the angle to distance in meters
  distanceToTarget = delta * 6372795;  // Earth's radius in meters
```

Figure 16-Distance to way point.

The formula to calculate the course.

is the formula used to calculate the initial bearing (course) from the current position (current Latitude, current Longitude) to a target position (target Latitude, target Longitude), This is necessary because the Earth is roughly spherical, and direct linear calculations are not accurate over long distances or large changes in latitude.

1. Convert all latitude and longitude values from degrees to radians.
2. Calculate the differences → $\Delta\lambda$ = target Longitude − current Longitude.
3. Do this formula → θ= atan2(sin ($\Delta\lambda$)·cos (target Latitude), cos (current Latitude) ·sin (target Latitude) − sin (current Latitude) ·cos (target Latitude) ·cos ($\Delta\lambda$))
4. Convert from radians to degrees. → $\theta_{degrees} = \theta \times \pi 180$
5. Normalize the bearing to a compass heading → bearings are expressed as angles between 0° and 360°. To obtain the final bearing, add 360° if the calculated bearing is negative.

```
// Calculate course in degrees to reach the waypoint
int courseToWaypoint()
{
  // Calculate the difference in longitude between current position and target waypoint
  float deltaLongitude = radians(targetLongitude - currentLongitude);

  // Convert current latitude and target latitude to radians
  float currentLatitudeRadians = radians(currentLatitude);
  float targetLatitudeRadians = radians(targetLatitude);

  // Calculate components of the course calculation formula
  float componentX = sin(deltaLongitude) * cos(targetLatitudeRadians);
  float componentY = cos(currentLatitudeRadians) * sin(targetLatitudeRadians) -
                     sin(currentLatitudeRadians) * cos(targetLatitudeRadians) * cos(deltaLongitude);

  // Compute the arctangent of componentX and componentY to get the angle in radians
  float angleRadians = atan2(componentX, componentY);

  // Ensure the angle is within the range [0, 2*π) radians
  if (angleRadians < 0.0) {
    angleRadians += TWO_PI;
  }

  // Convert the angle from radians to degrees
  targetHeadingDegrees = degrees(angleRadians);

  // Return the calculated target heading in degrees
  return targetHeadingDegrees;
}
```

Figure 17- Course to waypoint.

Next way Point

enhances automatic navigation by updating target coordinates, moving to the next waypoint, and guaranteeing precise GPS processing. It provides unambiguous feedback and safely stops the operation when the last waypoint is achieved. This feature is perfect for a variety of applications, including robotics, mapping, autonomous vehicles, and search and rescue, as it streamlines control flow, improves modularity, and allows real-time updates. Because of its design, it can handle scenarios involving more complicated waypoint management with flexibility and efficient use of resources.

```
// Function to move to the next waypoint in the waypoint list
void nextWaypoint() {
    // Increment the current waypoint index
    current_waypoint++;

    // Update the target latitude and longitude to the new waypoint coordinates
    targetLatitude = waypointList[current_waypoint][0];
    targetLongitude = waypointList[current_waypoint][1];

    // Check if the new target is the last waypoint (indicated by coordinates (0, 0))
    if ((targetLatitude == 0 && targetLongitude == 0)) {
        // Call a function to stop the vehicle or process
        stoprobot();

        // Enter an infinite loop to stop further execution, requiring a reset to continue
        while (1);
    } else {
        // Process the current GPS data
        processgps();

        // Calculate the distance to the new target waypoint and set it as the original distance
        distanceToTarget = originalDistanceToTarget = calculateDistanceToWaypoint();

        // Calculate the course to the new target waypoint
        courseToWaypoint();
    }
}
```

Figure 18-Next way Point.

Read Compass

Determining Direction	Computes the robot's heading in degrees to know which direction it's facing.
Path Following	Provides the current heading to compare with the desired path direction and adjust movement.
Waypoint Navigation	Calculates direction to the next waypoint and adjusts the robot's path accordingly.
Course Adjustment	Helps in adjusting the robot's orientation to correct deviations from the planned path.

```
int readCompass()
{
  // Create an event to hold sensor readings
  sensors_event_t compassEvent;
  compass.getEvent(&compassEvent);

  // Calculate the heading in radians
  float headingRadians = atan2(compassEvent.magnetic.y, compassEvent.magnetic.x);

  // Declination angle for Aswan, Egypt (3 degrees 30 minutes East)
  float declinationAngleRadians = (3.0 + (30.0 / 60.0)) * (M_PI / 180.0); // Convert to radians
  headingRadians += declinationAngleRadians;

  // Normalize heading to be within 0 to 2*PI radians
  if (headingRadians < 0) {
    headingRadians += 2 * M_PI;
  }
  if (headingRadians > 2 * M_PI) {
    headingRadians -= 2 * M_PI;
  }

  // Convert heading to degrees
  float headingDegrees = headingRadians * 180 / M_PI;

  // Return the heading in degrees as an integer
  return ((int)headingDegrees);
}
```

Figure 19-Read Compass

Calculate Desired Turn

Describe the robot's turning angle in relation to a target heading. To choose the proper turning action, it calculates the difference between the robot's current heading and the intended target heading.

```
// Function to calculate the desired turn to reach the target heading
void calcDesiredTurn(void)
{
  // Calculate the heading error (difference between target and current heading)
  headingError = targetHeading - currentHeading;

  // Adjust for compass wrap-around to keep heading error within the range -180 to 180 degrees
  if (headingError < -180)
    headingError += 360;  // Adjust error for values less than -180 degrees
  if (headingError > 180)
    headingError -= 360;  // Adjust error for values greater than 180 degrees

  // Determine the error output based on the heading error
  if (abs(headingError) <= HEADING_TOLERANCE)  // If the error is within tolerance, no need to turn
    error_output = 0;  // No turning needed, as the heading is close to the target
  else if (headingError < 60 && headingError > -60)  // If error is small, use proportional control
  {
    error_output = headingError;  // Proportional adjustment to reduce heading error
  }
  else if (headingError >= 60)  // If error is large and positive, set maximum right turn
    error_output = 100;  // Maximum right turn for large positive error
  else if (headingError <= -60)  // If error is large and negative, set maximum left turn
    error_output = -100;  // Maximum left turn for large negative error
}
```

Figure 20-calculate desired turn.

Move Robot

The robot's movements are guided by the move robot() function, which interprets the error output from calcDesiredTurn(). Depending on the heading inaccuracy, it gives instructions for going forward, turning left or right, and making acute turns. This feature is essential for modifying the robot's orientation and guaranteeing that it travels the intended route to arrive at the destination.

You can efficiently control the robot's motions in real-time by using move robot()

```
// Function to move the robot based on the error output
void move_robot()
{
  if (error_output == 0)  // If error output is zero, move forward
  {
    forward();  // Move the robot forward as it is already on the right path
  }
  else if (error_output < 60)  // If error output is between -60 and 60, turn left or right
  {
    if (error_output < 0)
      left();  // Turn left if the error output indicates the robot is too far right
    else
      right();  // Turn right if the error output indicates the robot is too far left
  }
  else if (error_output == 100)  // If error output is maximum positive, make a sharp right turn
  {
    sharpright();  // Execute a sharp right turn to correct a large heading error
  }
  else if (error_output == -100)  // If error output is maximum negative, make a sharp left turn
  {
    sharpleft();  // Execute a sharp left turn to correct a large heading error
  }
}
```

Figure 21- Move Robot

Object Detection

In computer vision, object detection is the process of locating things in pictures or movies. In order to produce useful results, these algorithms frequently rely on machine learning or deep learning techniques.
Now, let's use the illustration below to assist us simplify this statement a little.

Figure 22-example of detection. []

Therefore, we must really locate a dog in the image rather than categorizing the type of dog that is present in these photographs. In other words, I need to determine where the dog is in the picture. Is it at the bottom left or in the center? And so forth. The next thought that crosses people's minds is, "How can we do that?" Now let's get going.
We can, however, draw a box around the dog in the picture and give it specific x and y coordinates.

Figure 23 illustration bounding box. [24]

For the time being, assume that these boxes' coordinates can be used to indicate an object's location in the image. Hence, the box that encloses the object in the picture is referred to as a bounding box. This turns into an image localization challenge now, where we have to figure out where the object is located in a group of photographs, take note that there is just one class here. What occurs if there are several classes? this is an illustration

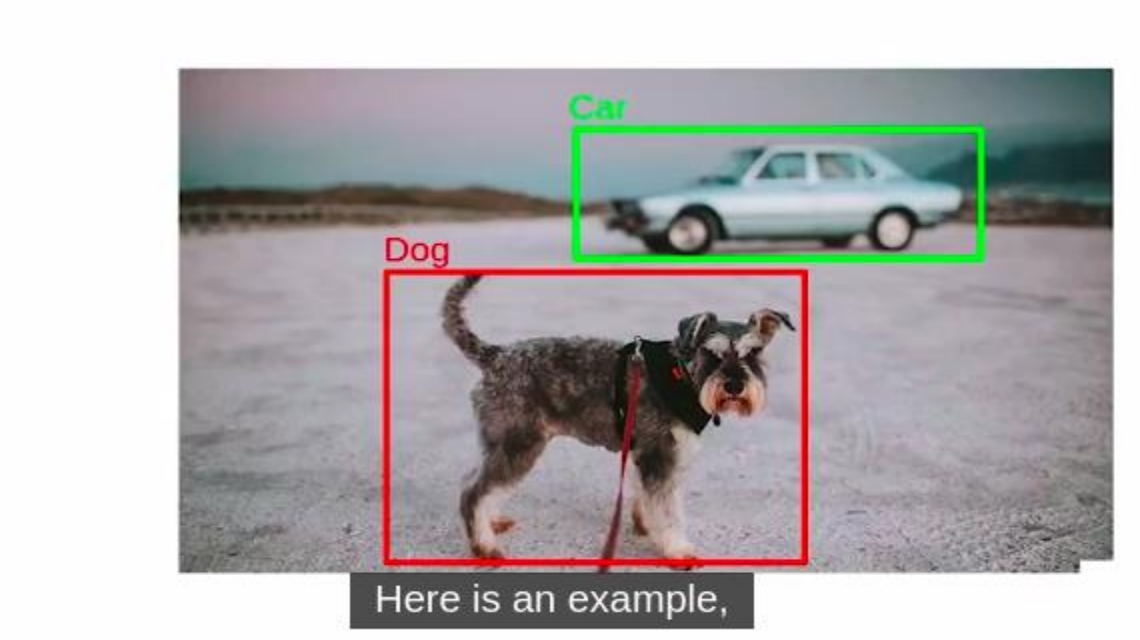

Figure 24 example 2 of detection. [24]

We must locate the objects in this picture, but keep in mind that not all of them are dogs. We have a car and a dog here. Therefore, in addition to finding the objects in the picture, we also need to identify if the object we find is a dog or a car. Thus, this turns into an issue with object detection. It is now possible to categorize the object detection difficulty into several groups. The first situation is when you have single-object photos. In other words, a data set containing 1000 photos will only contain one object per image. Additionally, there will be an image localization issue if every single one of these objects is an automobile and hence belongs to a single class. That is, all you need to do is figure out where these items are in the picture because. you already know what class they are in

.

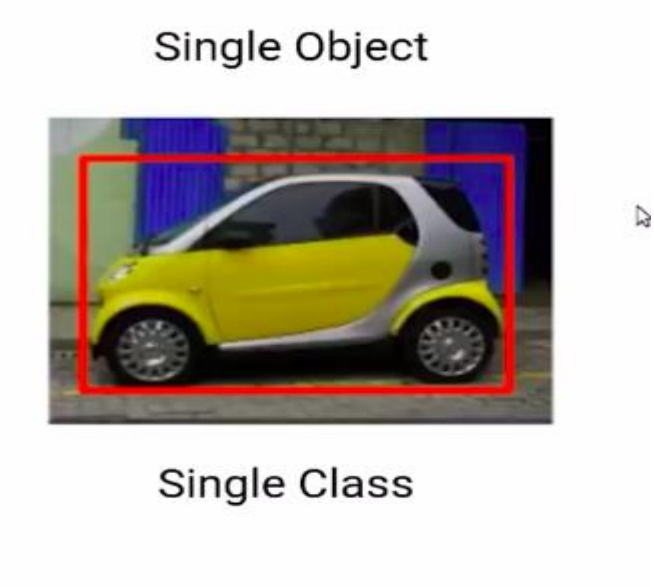

Figure 25 single class. [24]

An additional issue might arise if you are given many photos, each of which contains multiple items. It is possible for these objects to belong to the same class, or there could be an issue if they belong to distinct classes.
If an image contains several objects, each of which belongs to a separate class. In addition to finding the objects, you would also need to classify them.

Why Is Object Detection Important?
1. Safety: By identifying threats and intruders, it keeps us safe.
2. Driving: Preventing collisions is essential for self-driving automobiles.

3. Shopping: It facilitates product management and customer understanding in stores.
4. Medical imaging: It helps physicians identify ailments at an early stage.
5. Manufacturing: It guarantees that goods are produced accurately in factories

How Does Object Detection Operate?
Object Detection Operates Here:
• Examining the Image: Consider a computer examining an image.

• Looking for Hints: The computer scans the image for clues like colors, patterns and shapes.
• Guessing What's There: It makes intelligent guesses based on these clues.

• Verifying the Guesses: Every guess is verified by cross-referencing it with its prior knowledge.
• Drawing Boxes: When it is reasonably certain of something, it indicates its estimation of the object's location by drawing a box around it.
• Verifying: Lastly, it confirms its estimations to ensure accuracy and correct any errors.

The You Only Look Once (YOLO) algorithm
Over time, numerous iterations of object detection algorithms with differing levels of

performance have been created. However, the YOLO algorithm stands out and is now preferred because of its notable speed difference. This algorithm differs from previous variations in that it does not rely on various loops on region proposals for multiple rounds of categorization and prediction. Using a convolutional neural network, it predicts both the bounding boxes of the objects in the image and their presence in one pass. In what way is this even feasible? Is it not necessary to label every object at every location in the image? In that case, how does YOLO identify every object in the image without making several region suggestions?

While YOLO does not employ region proposals, it does use a related idea. YOLO separates the incoming image conceptually into fixed-sized grids rather than offering region options. For example, the 7 × 7 grid is used in YOLO version 1. One forecast, like the region suggestions, must be made by each grid. With one exception: YOLO uses a single network to do feature extraction, object classification, and bounding box prediction, whereas region proposals need us to extract features and classify them using multiple networks.
The object detection problem is reduced to a regression problem by the YOLO model. The network receives a 448 * 448 shaped input picture. The image is sent into a convolutional neural network, which extracts features in the 7 x 7 shape. Every feature that is retrieved represents a grid. All of the picture class predictions, bounding box predictions, and the 7 x 7 characteristics are fed into a thick layer.

THE YOLO ARCHITECTURE

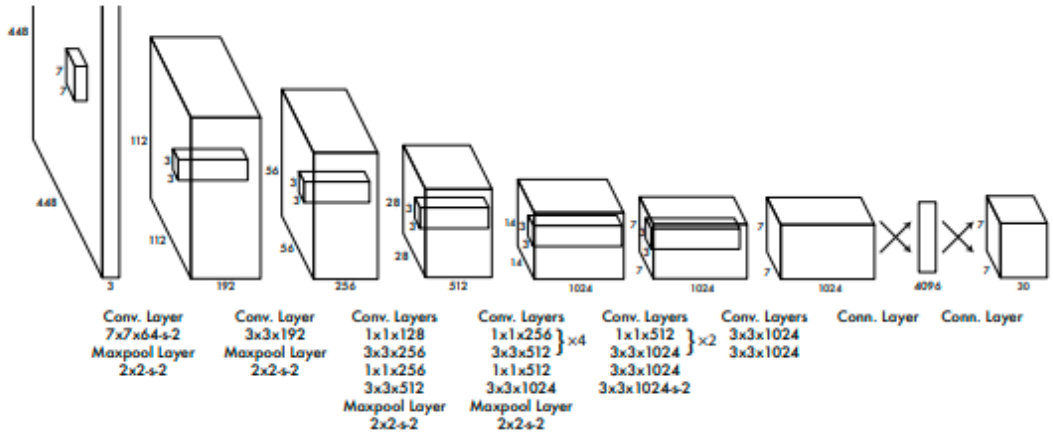

Figure 26 YOLO Architecture [24]

There are two completely connected layers among the 24 convolutional layers in the Yolo v1 architecture. An image with dimensions of 448 x 448 is fed into the convolution layers, and it is reduced to a 7 x 7 feature space. The grid used to make predictions is represented by the 7 × 7 feature space. Then, in order to regress the 7 x 7 characteristics into an output of the appropriate shape, they are fed into fully linked layers.

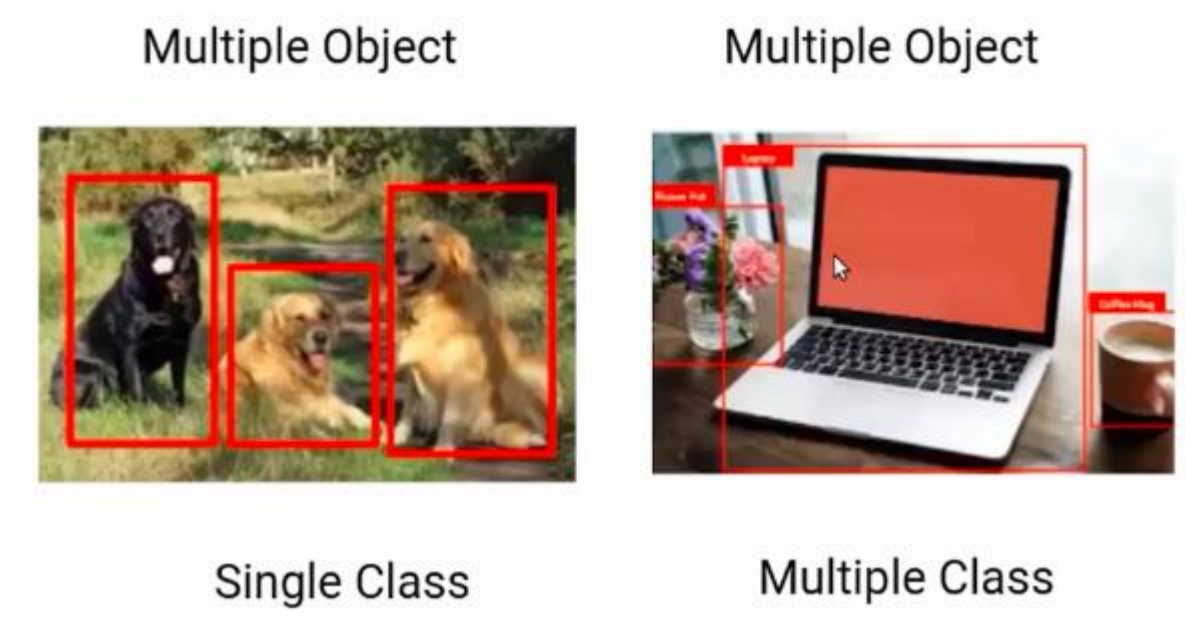

Figure 27 single and multiple classes.[24]

Historical Evolution:

1-Milestones in object detection research.

2-Key algorithms and models over the years.

Deep Learning-Based Detection Approaches

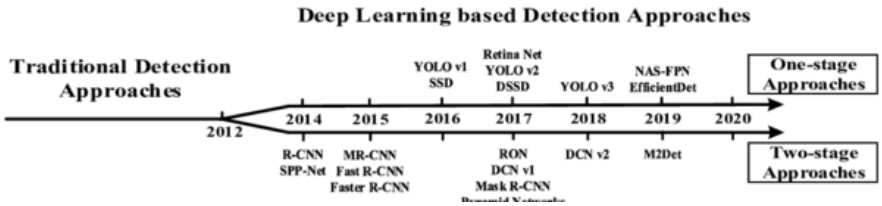

Figure 28 Deep Learning based Detection Approaches [25]

Two-Stage Approaches

2014:

R-CNN (Convolutional Neural Networks with Regions)

SPP-Net: Spatial Pyramid Pooling Networks.

2015:

Fast R-CNN (Faster Training of R-CNN)

Faster R-CNN (Introduction of Region Proposal Networks (RPN))

2016:

MR-CNN: Multi-region CNN.

RON: Reverse connection with objectness prior networks.

2017:

Mask R-CNN (Extension of Faster R-CNN for instance segmentation)

DCN v1 (Deformable Convolutional Networks)

2018:

DCN v2: Second version of Deformable Convolutional Networks.

Cascade R-CNN: Improving detection by cascading stages.

2019:

M2Det: Multi-level feature pyramid network.

One-Stage Approaches

2015:

YOLO v1: Real-time Object Detection System.

SSD: Single Shot MultiBox Detector.

2016:

YOLO v2: Improved Version of YOLO.

2017:

RetinaNet: Introduction of Focal Loss to address class imbalance.

DSSD: Deconvolutional Single Shot Detector.

YOLO v3 (Further Improvements on YOLO)

2018:

NAS-FPN (Neural Architecture Search for Feature Pyramid Networks)

2019:

EfficientDet (Efficient and Scalable Object Detection)

1-Two-Stage Approaches

- 2014:

  - R-CNN:(Regions with Convolutional Neural Networks) method to generate fed into a CNN for object detection.

  - SPP-Net: (Spatial Pyramid Pooling Networks) make CNNs to process images of varying sizes

- 2015:

  - Fast R-CNN: Enhanced R-CNN by sharing convolutional computations across proposals, importantly speeding up the training process.

  - Faster R-CNN: optimized Fast R-CNN by integrating a Region Proposal Network (RPN) directly into the model for end-to-end training.

- 2016:

  - MR-CNN (Multi-Region CNN): multi-region approach to improve object detection accuracy by focusing on various parts of an image.

  - RON (Reverse Connection with Objectness Prior Networks): Introduced a network that emphasizes objectness in early layers, enhancing detection performance.

- 2017:

  - Mask R-CNN: Extended Faster R-CNN to also perform pixel-level object segmentation, making it suitable for instance segmentation tasks.

  - DCN v1(Deformable Convolutional Networks): Introduced deformable convolutions to model geometric transformations and improve detection accuracy.

- 2018:

  - DCN v2: Improved upon DCN v1 by refining the deformable convolutional layers for better feature representation.

  - Cascade R-CNN: Enhanced object detection by implementing a multi-stage architecture, progressively refining object proposals.

- 2019:

  - M2Det (Multi-Level Feature Pyramid Network): Developed a network that combines features from multiple layers, achieving better detection performance, especially for small objects.

**One-Stage Approaches

- 2015:

  - YOLO v1 (You Only Look Once)

- SSD (Single Shot MultiBox Detector): that discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales.

- 2016:

- YOLO v2: Improved YOLO v1 by i high-resolution classifiers, and anchor boxes, enhancing detection accuracy and speed.

- 2017:

- RetinaNet: Introduced the Focal Loss function to address the class imbalance problem in one-stage detectors, improving accuracy.

- DSSD(Deconvolutional Single Shot Detector): Enhanced SSD by adding deconvolutional layers, which help in better detecting smaller objects.

- YOLO v3: Further refined YOLO with multi-scale predictions and a more complex backbone network, improving performance on various benchmarks.

- 2018:

- NAS-FPN (Neural Architecture Search for Feature Pyramid Networks): Utilized neural architecture search to automatically design feature pyramid networks, optimizing performance.

- 2019:

- EfficientDet: Developed an efficient and scalable object detection model using a compound scaling method to balance network depth, width, and resolution for better performance with fewer resources.

Detailed explanations of algorithms such as:

YOLO (You Only Look Once)

SSD (Single Shot MultiBox Detector)

R-CNN (Region-based Convolutional Neural Networks) and its variants (Fast R-CNN, Faster R-CNN, Mask R-CNN)

RetinaNet

(2)YOLO (You Only Look Once):

Created by Joseph Redmon et al., YOLO revolutionized object detection by framing it as a single regression problem, predicting bounding boxes and class probabilities directly from full images in one evaluation, achieving real-time detection speeds.

SSD (Single Shot MultiBox Detector):

Proposed by Wei Liu et al., SSD uses a single-stage approach to detect objects by discretizing the output space of bounding boxes into a set of default boxes over different aspect ratios and scales, balancing speed and accuracy.

RetinaNet:

Introduced by Tsung-Yi Lin et al., RetinaNet addressed the class imbalance problem in one-stage detectors with the Focal Loss function, significantly improving detection accuracy.

Datasets for Object Detection:

1-Popular datasets like COCO, Pascal VOC, and ImageNet.

2-explain your own dataset.

1-(3)COCO stands for Common Objects in Context dataset, as the image dataset was created with the goal of advancing image recognition. The COCO dataset contains challenging, high-quality visual datasets for computer vision, mostly state-of-the-art neural networks.

For example, COCO is often used to benchmark algorithms to compare the performance of real-time object detection. The format of the COCO dataset is automatically interpreted by advanced neural network libraries.

2-(4) The Urban Street Objects dataset is designed for detecting various objects commonly found in urban street environments. It includes images captured from different urban settings, such as

busy intersections, residential streets, and commercial areas. The dataset aims to aid in developing models for autonomous vehicles, smart city applications, and urban planning.

RESULTS AND DISCUSSION

Final design in for robot:

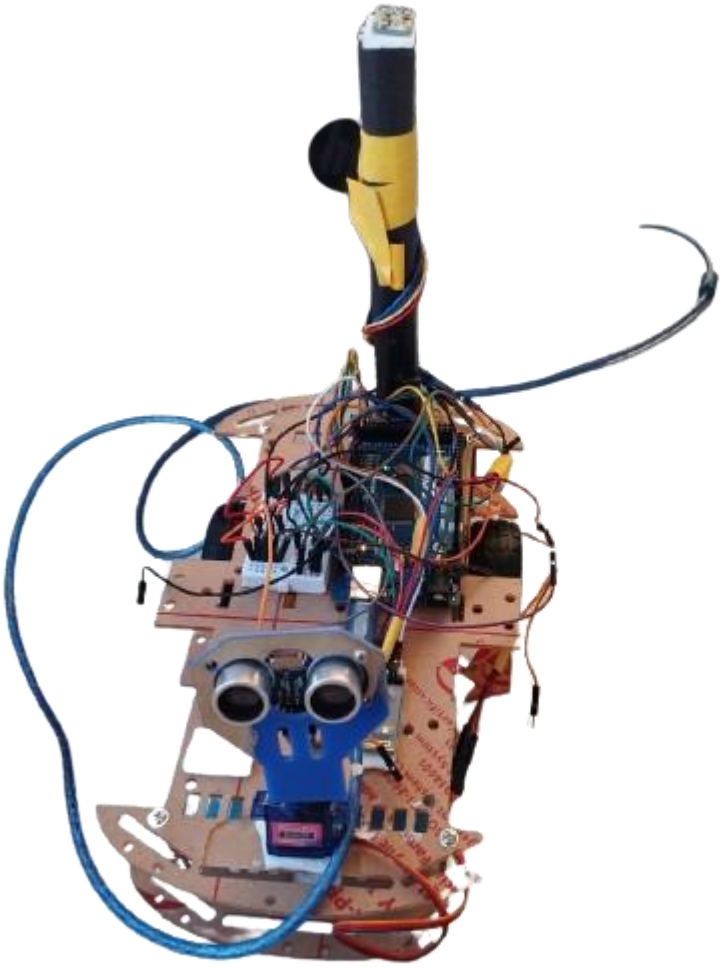

Figure 29 - design(1)

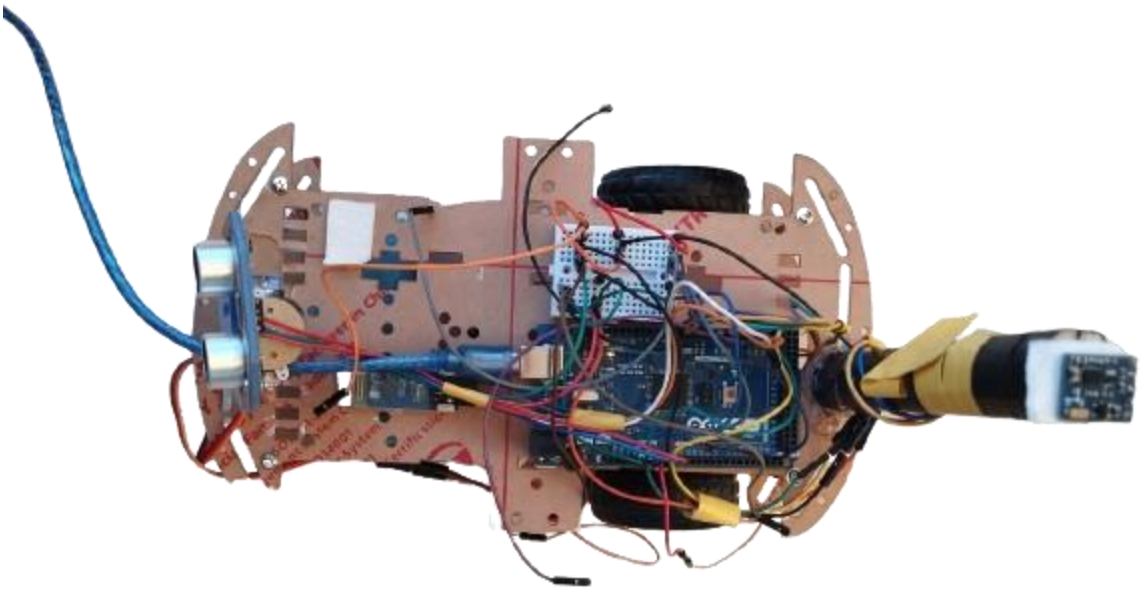

Figure 30 - design (2)

In general, the code provides information about two geographic locations and the times they were recorded. However, without more context, it is difficult to determine the specific purpose of the code. This result indicates that GPS has accurately determined the geographic location, which is represented by latitude and longitude coordinates.

- Latitude: 24.067291 degrees north.
- Longitude: 32.889137 degrees Earth.

```
Location: 24.067291,32.889137  Date/Time: 6/30/2024 10:57:01.00
```

Figure 31 – Location, Date & Time

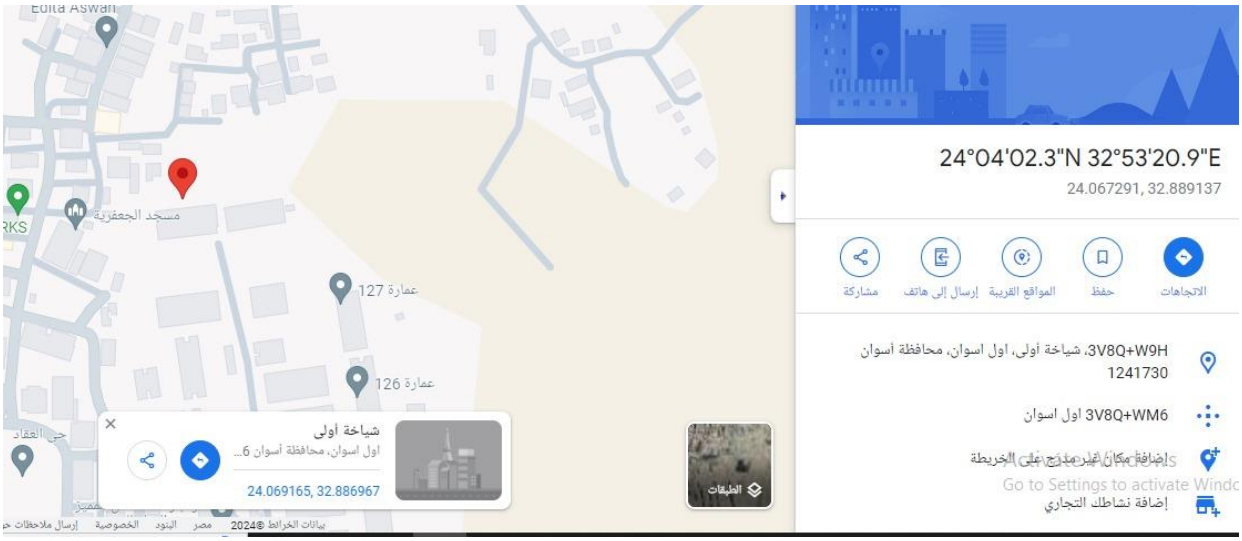

Figure 32 - Position from GPS

C h a p t e r F i v e

CONCLUSION

This work, The simulation successfully simulates obstacle detection using a camera (feature detection and matching) and pose estimation using feature tracking in successive frames, with a closed form solution. The pose information and object range data are used to avoid obstacles and navigate to the desired spot. Object detection and pose estimates were validated using a camera.The goal of future research is to fully implement the obstacle avoidance algorithm on a real mobile robot in an unstructured environment with moving objects.

References

[1] F. Caglar, "DEVELOPMENT OF AN AUTONOMOUS MOBILE ROBOT OUTDOOR NAVIGATION SYSTEM", December 2008

[2] A. J. Christian, "Mobile Robot Navigation",2007

[3]S Kaghyan, H Sarukhanyan "Accelerometer and GPS sensor combination-based system for human activity recognition" 23-27 September 2013

[4]S Anthony D. Ross "An analysis of operations efficiency in large-scale distribution systems" Received 1 October 2002, Revised 1 October 2003, Accepted 1 November 2003, Available online 19 December 2003

[5] Münzer, S., Zimmer, H. D., & Baus, J. (2012). Navigation assistance: A trade-off between wayfinding support and configural learning support. Journal of Experimental Psychology: Applied, 18(1), 18–37.

[6] "ieeexplore " [online].Available https://ieeexplore.ieee.org/document/4919345 Accessed on [23-1-2024 10:00PM]

[7] Montañés, J. A. P, Rodríguez, A.M, Prieto, I. S "Smart Indoor Positioning/Location and Navigation: A Lightweight Approach" June 2013

[8]" springer ". [online]. Available :https://link.springer.com/referenceworkentry/10.1007/978-3-540-30301-5_51 Accessed on [23-1-2024 11:00PM]

[9]" techtarget ". [online]. Available: https://www.techtarget.com/whatis/definition/sensor Accessed on [24-1-2024 11:00 AM]

[10] "Arduino Store". [online]. Available: https://store-usa.arduino.cc/products/arduino-mega-2560-rev3 [23-6-2024] [5:00 PM].

[11] "Micro Ohm Electronics". [online]. Available: https://microohm-eg.com/product/neo-6m-gps-module-with-eprom/ [23-6-2024] [5:15 PM].

[12] "Makers". [online]. Available: https://makerselectronics.com/product/jdy-33-dual-mode-bluetooth-spp-spp-c-compatible-with-hc-05-06-slave-bluetooth-3-0-module [23-6-2024] [5:38 PM].

[13] "Future Electronics". [online]. Available: https://store.fut-electronics.com/products/esp32-cam-development-board-with-camera [23-6-2024][5:50PM].

[14] "Makers". [online]. Available: https://makerselectronics.com/product/ultrasonic-sensor-hc-04-rcwl-9610-5v[23-6-2024][6:19PM].

[15] "Makers". [online]. Available: https://makerselectronics.com/product/gy-271-3-axis-magnetic-compass-sensor[23-6-2024][6:42PM].

[16] "Makers". [online]. Available: https://makerselectronics.com/product/mini-breadboard-170-pin [23-6-2024] [7:06PM].

[17] "Makers". [online]. Available: https://makerselectronics.com/product/l298-motor-driver-module [23-6-2024][7:30 PM].

[18] "Makers". [online]. Available: https://makerselectronics.com/product/dc-geared-motor-dual-shaft-312vdc-450rpm. [23-6-2024] [8:14 PM].

[19] "Makers". [online]. Available: https://makerselectronics.com/product/dc-geared-motor-dual-shaft-312vdc-600rpm-with-wheel-65mm. [23-6-2024][8:42 PM].

[20] "Makers". [online]. Available: https://store.fut-electronics.com/collections/battery/products/18650-lithium-ion-rechargeable-high-quality-battery-3-7v-1200-mah. [23-6-2024] [9:29 PM].

[21] "Sastron". [online]. Available: https://sastronlimited.com/ar/product/servo-motor-sg90-9g-plastic-gear/ [23-6-2024] [10:00 PM].

[22] "Future Electronics". [online]. Available: https://store.fut-electronics.com/products/ftdi-board-switchable-3-3-or-5v [23-6-2024] [10:19 PM].

[23]" Arduino ". [online]. Available: https://forum.arduino.cc/t/arduino-language-vs-c-c/65526 [23-6-2024] [11:23 PM].

[24] "".[online].Available : https://www.analyticsvidhya.com/blog/2022/03/a-basic-introduction-to-object-detection/ [24-6-2024] [12:10 AM].

[25] "". [online]. Available: https://www.researchgate.net/figure/Object-detection-milestones-based-on-deep-learning_fig1_340475800 [23-6-2024] [4:41 PM].