МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ, ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА АЕРОКОСМІЧНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Юрій МЕЛЬНИК
«_____» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
«БАКАЛАВР»

Тема: «**Адаптивна система автоматичного управління наземної безпілотної платформи**»

Виконавець: студент групи СУ-404 _____ Даніл ГЛУХИХ

Керівник: д.т.н., професор _____ Юрій МЕЛЬНИК

Нормоконтролер: _____ Микола ДИВНИЧ

Київ 2024

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY
FACULTY OF AIR NAVIGATION, ELECTRONICS AND TELECOMMUNICATIONS
AEROSPACE CONTROL SYSTEMS DEPARTMENT

APPROVED FOR DEFENCE
Head of the Department
_____ Yurii MELNYK
"____" _____ 2024

QUALIFICATION PAPER

(EXPLANATORY NOTE)

FOR THE ACADEMIC DEGREE OF BACHELOR

Title: "**Adaptive System of Automatic Control of Ground Unmanned Platform**"

Submitted by: student of group CS-404 _____ Danil HLUKHYKH

Supervisor: Doctor of Science, Professor _____ Yurii MELNYK

Standards inspector: _____ Mykola DYVNYCH

Kyiv 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра аерокосмічних систем управління

Спеціальність: 151 «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Юрій МЕЛЬНИК

«_____»_____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Глухих Даніла Олександровича

1. Тема кваліфікаційної роботи «Адаптивна система автоматичного управління наземної безпілотної платформи» затверджена наказом ректора від «01» квітня 2024 р. № 511/ст.
2. Термін виконання роботи: з 13.05.2024 по 16.06.2024.
3. Вихідні дані роботи: уявна наземна безпілотна платформа, симуляція моделі поведінки уникнення перешкод та слідування за маршрутом

4. Зміст пояснювальної записки: Огляд та аналіз систем автоматичного керування наземними безпілотними платформами, Принципи роботи та логічна схема адаптивної роботизованої наземної системи колісного типу, Дослідження та випробування адаптивної системи

5. Перелік обов’язкового ілюстративного матеріалу: Ілюстрація коду результатів моделювання алгоритмів A* та D* в інтегрованому середовищі розробки Python для уявного робота, презентація PowerPoint

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Уточнення постановки задачі	13.05.2024-15.05.2024	
2	Огляд літературних джерел	16.05.2024-20.05.2024	
3	Ознайомлення з адаптивними системами наземних платформ	21.05.2024-23.05.2024	
4	Дослідження моделів поведінки платформ	23.05.2024-25.05.2024	
5	Розробка експерементального коду моделі поведінки та їх апробація	26.05.2024-01.06.2024	
6	Оформлення пояснювальної записки	02.06.2024-10.06.2024	
7	Підготовка доповіді та презентації	11.05.2024-16.05.2024	

7. Дата видачі завдання: «13» травня 2024 р.

Керівник кваліфікаційної роботи _____ Юрій МЕЛЬНИК
(підпис керівника)

Завдання прийняв до виконання _____ Даніл ГЛУХИХ
(підпис випускника)

NATIONAL AVIATION UNIVERSITY

Faculty of Air Navigation, Electronics and Telecommunications

Aerospace Control Systems Department

Specialty: 151 “Automation and Computer-integrated Technologies”

APPROVED BY

Head of the Department

_____ Yurii MELNYK

“_____”_____2024

Qualification Paper Assignment for Graduate Student

Hlukhykh Danil Oleksandrovych

1. The qualification paper title “ Adaptive system of automatic control of ground unmanned platform” was approved by the Rector’s order of “ 01 ” April 2024 № 511/ст.

2. The paper to be completed between: 13.05.2024 and 16.06.2024

3. Initial data for the paper: imaginary ground-based unmanned platform, Model "Avoiding obstacles", Model "Route following"

4. The content of the explanatory note: Review and analysis of automatic control systems for ground-based unmanned platforms, Principles of operation and logical unit of adaptive robotic ground system of wheeled type, Research and testing of the adaptive system

5. The list of mandatory illustrations: Illustarion of the code of the results of modeling the A* and D* algorithms in an integrated Python development environment for an imaginary robot, PowerPoint presentation

6. Timetable

№	Assignment	Dates of completion	Completion mark
1	Clarification of the problem statement	13.05.2024-15.05.2024	
2	Review of literature sources	16.05.2024-20.05.2024	
3	Familiarization with adaptive systems of ground platforms	21.05.2024-23.05.2024	
4	Study of platform behavior models	23.05.2024-25.05.2024	
5	Development of experimental code for behavioral models and their testing	26.05.2024-01.06.2024	
6	Preparation of an explanatory note	02.06.2024-10.06.2024	
7	Preparation of the report and presentation	11.05.2024-16.05.2024	

7. Assignment issue date: “13” May 2024

Qualification paper supervisor _____ Yurii MELNYK
(the supervisor’s signature)

Issued task accepted _____ Danil HLUKHYKH
(the graduate student’s signature)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи "Адаптивна система автоматичного управління наземної безпілотної платформи" містить 83 сторінки, 34 ілюстрацій, 32 джерела.

Актуальність теми полягає в необхідності підвищення ефективності та безпеки наземних безпілотних платформ через використання адаптивних алгоритмів управління в умовах динамічного середовища.

Об'єктом дослідження є адаптивна система автоматичного управління наземною безпілотною платформою.

Предметом дослідження є алгоритми планування маршруту та уникнення перешкод, зокрема алгоритми A* та D*.

Метою роботи є розробка та моделювання адаптивної системи автоматичного управління для наземної безпілотної платформи, що використовує ефективні алгоритми для уникнення перешкод та слідування маршруту.

Методи дослідження включають алгоритмічне моделювання, симуляцію та аналіз ефективності алгоритмів A* та D* у процесі планування маршруту та уникнення перешкод для наземної безпілотної платформи.

Ключові слова: АДАПТИВНА СИСТЕМА, НАЗЕМНА БЕЗПІЛОТНА ПЛАТФОРМА, АЛГОРИТМ A*, АЛГОРИТМ D*, УНИКНЕННЯ ПЕРЕШКОД, ПЛАНУВАННЯ МАРШРУТУ, АВТОМАТИЧНЕ УПРАВЛІННЯ

ABSTRACT

Explanatory note for the qualification paper "Adaptive System of Automatic Control of Ground Unmanned Platform" includes: 83 pages, 34 figures, 32 sources.

The relevance of the topic is the need to improve the efficiency and safety of ground unmanned platforms through the use of adaptive control algorithms in a dynamic environment.

The object of research is an adaptive automatic control system for a ground unmanned platform.

The subject of the study is route planning and obstacle avoidance algorithms, in particular, A* and D* algorithms.

The purpose of the study is to develop and model an adaptive automatic control system for a ground-based unmanned platform that uses effective algorithms for obstacle avoidance and route following.

The research methods include algorithmic modeling, simulation, and analysis of the effectiveness of the A* and D* algorithms in the process of route planning and obstacle avoidance for a ground-based unmanned platform.

Keywords: ADAPTIVE SYSTEM, GROUND UNMANNED PLATFORM, A* ALGORITHM, D* ALGORITHM, OBSTACLE AVOIDANCE, ROUTE PLANNING, AUTOMATIC CONTROL

CONTENT

INTRODUCTION

In today's world, unmanned ground platforms are increasingly used in various industries, from military and security to agriculture and logistics. The relevance of research in the field of automatic control of such platforms is due to the need to ensure their high maneuverability, reliability, and adaptability to changing environmental conditions.

The aim of the study is to create an effective system capable of autonomous movement in complex and dynamically changing environments, taking into account the presence of various obstacles and route changes.

To achieve this goal, several algorithms for finding paths and modeling the behavior of a ground platform were considered and implemented. In particular, the A* and D* algorithms were studied in detail, which are noted for their effectiveness in finding optimal routes in discretized space. These algorithms were integrated into the developed control system and tested in various scenarios.

The implementation of the developed adaptive automatic control system for a ground unmanned platform will increase the efficiency and reliability of platforms in various industries, ensuring their safe and autonomous operation in difficult conditions.

Thus, this thesis makes a contribution to the development of automatic control technologies for ground unmanned platforms and opens up new prospects for their further improvement and practical application.

SECTION 1
REVIEW AND ANALYSIS OF AUTOMATIC CONTROL SYSTEMS FOR GROUND-BASED UNMANNED PLATFORMS

1.1. Robotic platforms: variations, features and applications

Ground-based moving objects play an increasingly important role in various spheres of life. They are capable of performing a wide range of tasks, which facilitates human work and increases the efficiency of many processes. The main types of ground-based moving objects include autonomous mobile robots, teleoperated robots, and semi-autonomous robots.

Robotic platforms are playing an increasingly important role in a variety of fields, including industry, logistics, research, and military operations. With the development of automatic control technologies and sensor systems, these platforms are becoming more autonomous, reliable and efficient. One of the key areas of development is the creation of adaptive control systems that can respond to changing environmental conditions and tasks.

This section discusses the main variations of robotic platforms, their features and applications. Particular attention is paid to adaptive automatic control systems that ensure a high level of autonomy and efficiency of ground-based unmanned platforms. The principles of functioning of such systems and remote-control technologies, including ultrasonic, infrared and radio frequency sensors, are also considered.

The analysis of modern solutions and technologies makes it possible to identify the main challenges and prospects for development in the field of automatic control of ground unmanned platforms.

The main types of ground moving objects are:

1. **Autonomous mobile robots**. Autonomous mobile robots (AMRs) (see in Fig. 1.1.1) are a type of robot that can independently understand and move around in the environment without the need for control or constant supervision by an operator. This makes them more advanced than autonomous guided vehicles (AGVs), which rely on predefined paths or tracks and often require operator control [1].

Key features of autonomous mobile robots

Integration of sensors and cameras: AMRs use a variety of sensors and cameras to collect information about their environment. These sensors help robots to "see" and "hear" what is happening around them.

Artificial intelligence and machine learning: Due to artificial intelligence and machine learning algorithms, AMRs can interpret the data received from sensors and make decisions based on this information. This allows robots to efficiently plan routes and adapt to changes in the environment [2].

Autonomy: AMRs do not require wired power, which gives them the ability to move freely and perform tasks without route restrictions. This allows them to operate in large and complex environments, such as warehouses, hospitals, or manufacturing plants.

Fig. 1.1.1. Autonomous mobile robots “Dematic Pallet Mover AMR D900”

2. Teleoperated robots. These robots (see Fig. 1.1.2) are controlled remotely by a human and are typically used in situations where human safety may be at risk, such as in chemical contamination zones or mine clearance.

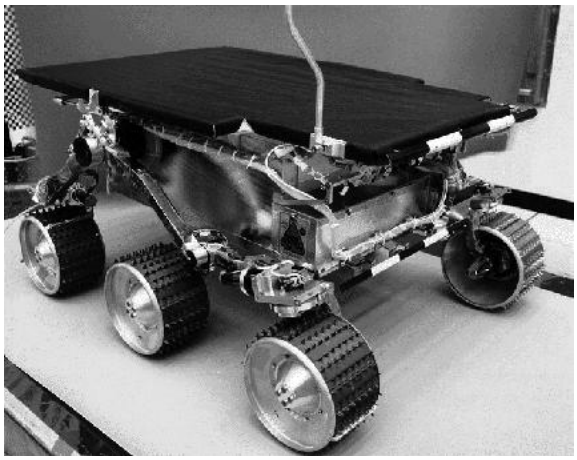


Fig. 1.1.2. The mobile robot Sojourner was used during the Pathfinder mission to explore Mars in summer 1997. It was almost completely teleoperated from Earth. However, some on-board sensors allowed for obstacle detection

3. Semi-autonomous robots. They (see Fig. 1.1.3) combine autonomous and teleoperation capabilities, allowing them to perform tasks with minimal human intervention. They are used in complex environments where real-time decision-making is required [3].

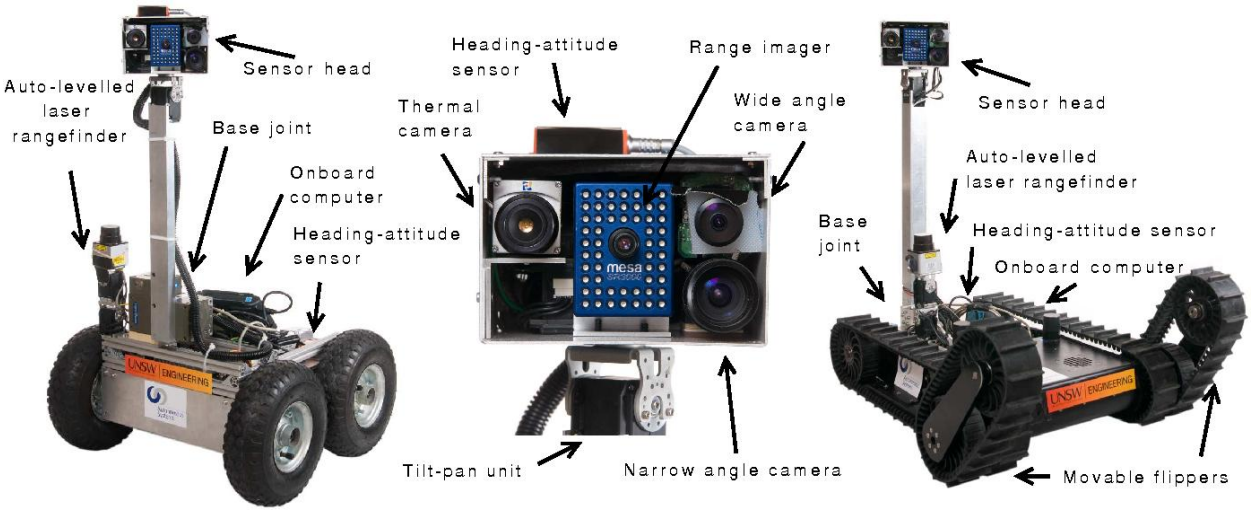


Fig. 1.1.3. Emu (left) and Negotiator (right), the two semi-autonomous robots forming our 2009 RoboCup Rescue Robot League entry, plus an enlargement of the sensor head

Robotic platforms that move on the ground are used in many scientific and commercial fields due to the continuous development of computer science, engineering and related disciplines. Their applications include different types of environments, from controlled to disordered outdoor environments, which creates different challenges and requirements for robots.

Variations and features of robotic platforms

Wheeled robots. Wheeled robots (Fig. 1.1.4) are the most common type of ground robot. They have high maneuverability on smooth surfaces, but may have difficulty on uneven terrain. Examples include logistics and warehousing robots, such as the Amazon Robotics robots [3].

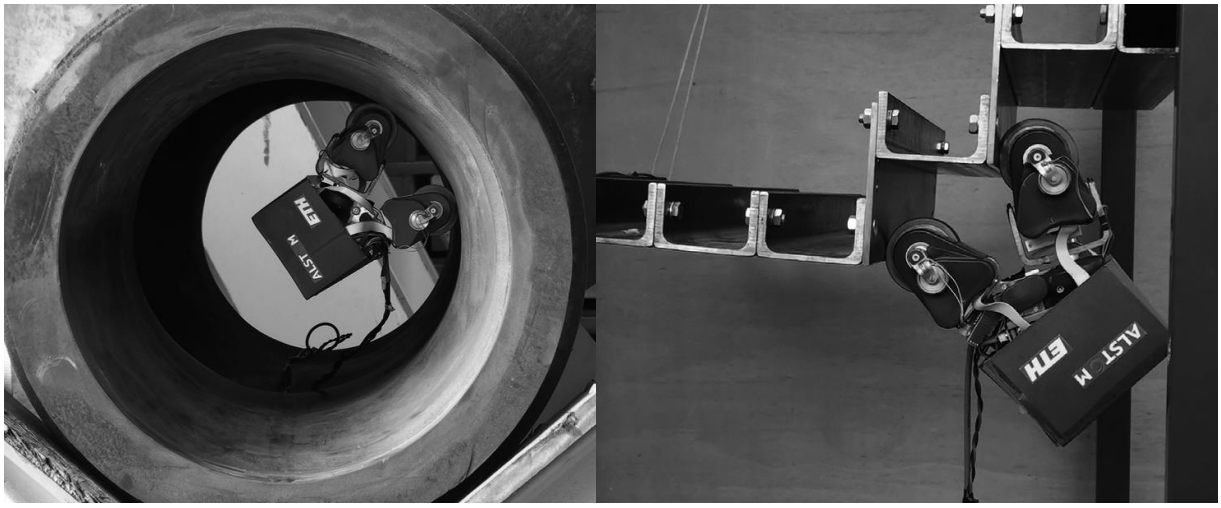


Fig. 1.1.4. The MagneBike robot developed by ASL (ETH Zurich) and ALSTOM. MagneBike is a magnetic wheeled robot with high mobility for inspecting complex shaped structures such as ferromagnetic pipes and turbines (http://www.asl.ethz.ch/). © ALSTOM / ETH Zurich.

Tracked robots. Tracked robots (see Fig. 1.1.5) are better suited to work on uneven and difficult terrain due to their increased stability and cross-country ability. They are often used in military, rescue operations, and heavy industry.

Fig. 1.1.5. Black Gladiator - Tracked Robot Chassis

Leg robots. Leg robots (see Fig. 1.1.6) have limbs that allow them to move over very difficult terrain where other types of robots cannot operate. They are used in volcano exploration, rescue operations in hard-to-reach places, and space exploration.

Fig. 1.1.6. ANYmal quadrupedal robot

Artificial intelligence (AI) robots. AI-enabled robots (see Fig. 1.1.7) can adapt to changing environments in real time, using machine learning algorithms to improve navigation and task performance. They are used in autonomous vehicles, agriculture, and healthcare [4].

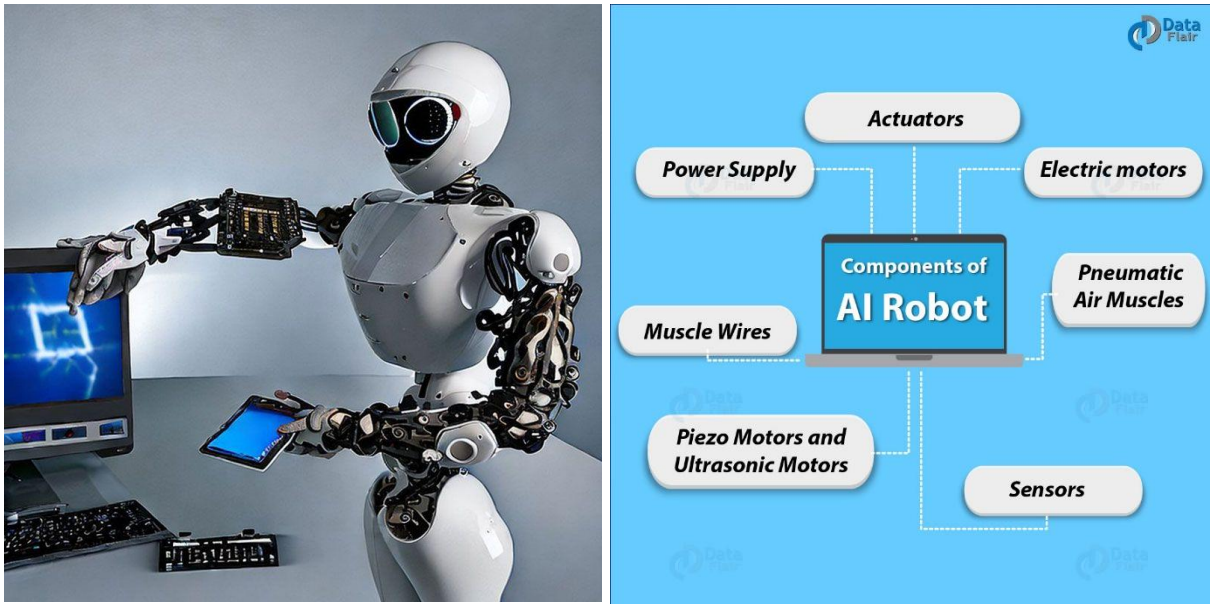


Fig. 1.1.7. AI-powered robotics and components of AI robot

Areas of application of ground moving objects

Modern robotics technologies are opening up new horizons for the use of ground moving objects in various industries. Let us look at the main areas where these technologies have a significant impact and bring numerous benefits.

In industry and logistics, robots are used to automate production processes, transport materials, and manage warehouse stocks. This helps to increase production efficiency and reduce costs.

In agriculture, robotic systems provide precision farming by performing tasks such as planting, spraying, harvesting, and monitoring field conditions. This increases productivity and reduces the need for manual labor.

In healthcare, robots perform complex surgical procedures, deliver medical supplies, and disinfect rooms. They can also provide assistance to patients with disabilities, which significantly improves the quality of medical services.

Rescue operations greatly benefit from the use of robots that can penetrate places that are dangerous to humans, such as natural disaster zones. They help in searching and rescuing victims, as well as in performing mine clearance tasks.

In space exploration, ground-based robots such as rovers explore the surface of other planets, conduct scientific experiments, and collect samples. This allows us to expand our knowledge of the Universe [3].

Such widespread use of moving objects demonstrates their versatility and potential for transforming various spheres of life.

1.2. Adaptive automatic control systems of a ground-based unmanned platform and its functioning principles

Unmanned ground platforms (UGPs) are an important component of modern technologies that are actively developing. Adaptive automatic control systems (AACS) for UGVs use the principles of machine learning and adaptation of control parameters in real time, which allows for optimized movement and avoidance of obstacles.

1. Using machine learning to analyze sensor data

Machine learning (ML) is a central element of modern AUVs, as it allows processing and analyzing large amounts of data from a variety of sensors. Sensors that can be used on UAS include:

- Lidars
- Cameras
- Radars
- Ultrasonic sensors
- Infrared sensors
- Radio frequency sensors

Machine learning helps to identify correlations, patterns, and insights in data that are not available to traditional analysis methods. The main benefits of ML for analyzing sensor data include:

- Automatic anomaly detection: ML models can detect unusual or dangerous conditions that may be missed by other methods.
- Prediction: Based on historical data, future events such as changes in the environment or possible obstacles can be predicted.
- Object classification and recognition: Deep learning algorithms can classify and recognize objects in real time, which is important for navigation and obstacle avoidance.

2. Adapts steering parameters in real time to optimize driving and avoid obstacles

Adaptive control algorithms are a key element of modern automatic control systems for unmanned ground vehicle. They allow for dynamic adjustment of control parameters in real time, which optimizes platform movement and ensures obstacle avoidance. This process includes the use of advanced path planning algorithms, Bayesian optimization, and adaptive filters.

Path planning algorithms

1. A* (A-Star): A is an efficient path-finding algorithm that finds the shortest path between two points, taking into account obstacles. It uses a heuristic function to estimate the cost of the path to the goal, which allows you to quickly and accurately find the best routes.

2. DWA (Dynamic Window Approach): DWA is a dynamic obstacle avoidance algorithm that takes into account the robot's kinematics and the location of nearby objects. It calculates the optimal trajectory in real time, ensuring safe and efficient maneuvering in difficult conditions.

Bayesian optimization

Bayesian optimization is used to tune controller parameters such as speed, acceleration, and steering angles to ensure optimal performance in changing conditions. This approach allows the system to adapt to new data and changing conditions, ensuring high efficiency and reliability.

Adaptive filters

1. Kalman filter: The Kalman filter is used to estimate the state of the system in real time, allowing control parameters to be adjusted based on the data. This filter works effectively in linear systems and provides accurate estimates even in the presence of noise.

2. Particle filter: The particle filter is a more flexible approach that can be used in nonlinear systems. It allows to take into account complex dynamic changes and provides high accuracy of system state estimation.

Using adaptive control algorithms

Adaptive control algorithms allow systems to automatically adjust control parameters in real time to optimize platform movement and avoid obstacles. Key approaches include:

Automatic parameter tuning: Systems can automatically adjust speed, acceleration, and steering angles to ensure optimal performance in different conditions. Processing data from sensors: The use of various types of sensors (lidars, cameras, radars, ultrasonic sensors, and infrared sensors) allows you to collect detailed information about the environment.

Implementation of adaptive control strategies: Integration of data and analysis results in the control process allows for the creation of adaptive strategies that ensure reliable and efficient navigation.

Adaptive automatic control systems for ground-based unmanned platforms use advanced path planning algorithms, Bayesian optimization, and adaptive filters to dynamically adjust control parameters. This allows for optimized platform movement, obstacle avoidance, and overall system efficiency in changing and challenging environments. Such systems are key to the development of robotic ground platforms capable of operating autonomously with a high level of reliability and safety.

Advantages of adaptive automatic control systems.

Improved navigation and obstacle avoidance: By continuously adapting and utilizing ML, UGVs can navigate effectively in complex environments.

Increased safety: Automatic anomaly detection and prediction of possible threats increase the safety of using BNPs.

Efficiency and autonomy: The systems can operate without constant operator intervention, which reduces the need for human resources and increases the autonomy of the platform.

1.3. Remote control technologies

Remote control technologies for unmanned ground vehicles (UGVs) have evolved significantly due to the development of sensor systems and data processing algorithms. These platforms require highly accurate and reliable navigation to perform

complex tasks in a variety of environments. The use of modern sensors allows UGVs to effectively navigate the environment, avoid obstacles and perform their functions with maximum efficiency.

The main types of sensors used in remote control technologies include ultrasonic, infrared, and radio frequency sensors. Each of these sensors has its own unique properties and benefits that make them indispensable for specific applications in ground-based unmanned platforms. Ultrasonic sensors provide precise distance measurement and object detection, infrared sensors are used to detect heat and motion, and radio frequency sensors such as RFID provide identification and tracking of objects over long distances.

These sensor technologies work in conjunction with adaptive control systems and machine learning algorithms to enable the BNP to quickly adapt to changing environmental conditions and perform tasks efficiently. The interaction of sensor data with processing algorithms allows platforms to make real-time decisions, which is critical for safe and reliable operation in dynamic and unpredictable environments.

The following sub-sections will provide a detailed analysis of each of the remote-control technologies, including how they operate and their specific applications in ground unmanned platforms.

1.3.1. Ultrasonic sensors

Ultrasonic motion sensors are an innovative technology that significantly improves the way robots interact with their environment. They use high-frequency sound waves to detect objects and measure distances, allowing robots to maneuver smoothly even in the most challenging environments. The main components of an ultrasonic sensor include a transmitter that emits sound waves and a receiver that waits for them to echo. By measuring the time between sending and receiving the signal, the sensor determines the distance to the object, allowing robots to make informed decisions and act accordingly.

Key advantages of ultrasonic sensors:

1. Obstacle detection: Ultrasonic sensors effectively detect obstacles such as walls, furniture, or other objects that may interfere with the platform's movement. This allows you to avoid collisions and navigate in tight spaces.

2. Automatic control: Using data from ultrasonic sensors, automatic control systems can be developed that allow the platform to respond to changes in the environment. For example, the platform can change its speed or direction to avoid obstacles and maintain safety.

3. Precise positioning: Ultrasonic sensors help the platform pinpoint its location in space, which is critical for a variety of tasks such as delivering goods or working in hard-to-reach areas.

4. Avoidance of collisions: Thanks to ultrasonic sensors, the platform can detect and avoid collisions with other objects or platforms in its path. This ensures the safety of both the platform and the surrounding objects.

Ultrasonic sensors are an integral part of modern robotic systems, opening up many opportunities for automation and innovation. They provide reliable object detection and high accuracy distance measurement, which are critical for the safety and efficiency of robotic systems. Due to their properties, ultrasonic sensors significantly improve the efficiency and safety of remote-control technologies for ground-based unmanned platforms, making them more autonomous and able to respond to changes in the environment [5].

1.3.2. Infrared Sensors

Infrared (IR) sensors play a crucial role in modern remote-control systems for ground-based unmanned platforms, offering a range of functionalities beyond simple object detection. Here's a deeper dive into the technology and its applications:

1. Heat Detectors:

Thermal Cameras: These devices use infrared radiation emitted by objects to create images based on temperature variations. They are invaluable in scenarios where visibility is limited, such as during nighttime operations or in environments with heavy fog or smoke. Thermal cameras can detect not only the presence of objects but also their thermal signatures, allowing for enhanced situational awareness.

Passive Infrared Sensors (PIR): PIR sensors are commonly used for motion detection in security systems, automatic lighting controls, and energy-efficient applications. They work by detecting changes in infrared radiation patterns caused by the movement of warm objects, such as humans or animals. PIR sensors are highly sensitive and can distinguish between background radiation and the radiation emitted by moving objects, making them ideal for detecting intruders or occupants in a designated area.

Pyroelectric Sensors: These sensors utilize the pyroelectric effect to detect changes in temperature and convert them into electrical signals. They are widely used in occupancy sensors, alarm systems, and industrial automation applications. Pyroelectric sensors offer fast response times and can detect even subtle temperature changes, making them suitable for various security and surveillance tasks [6].

2. Motion Detectors:

Active Infrared Sensors: Unlike passive sensors, active IR sensors emit infrared radiation and measure the intensity of the reflected signal. They are commonly used in proximity sensors, object detection systems, and automated door openers. Active IR sensors can operate effectively in both indoor and outdoor environments and are capable of detecting objects at relatively long ranges.

Infrared sensors offer numerous advantages for ground-based unmanned platforms, including:

Enhanced Perception: By detecting heat signatures and motion patterns, infrared sensors provide unmanned platforms with valuable information about their surroundings, enabling them to navigate safely and avoid obstacles.i

Improved Security: Infrared sensors are widely used in security systems for intrusion detection, perimeter monitoring, and asset protection. They offer reliable detection capabilities even in low-light conditions, making them indispensable tools for maintaining the security of critical infrastructure and facilities.

Energy Efficiency: In applications such as automatic lighting control and PIR sensors help conserve energy by activating or deactivating devices based on occupancy detection. By reducing unnecessary energy consumption, infrared sensors contribute to overall energy efficiency and cost savings.

Overall, infrared sensors play a vital role in enhancing the functionality, safety, and efficiency of ground-based unmanned platforms across various industries and applications.

1.3.3. Radio Frequency Sensors

Radio frequency (RF) technology for remote control, including radio frequency sensors, is a crucial component of modern ground-based unmanned platforms. This technology provides reliable communication and control over unmanned vehicles, enabling them to perform various tasks with high efficiency and precision. RF sensors, also known as RFID sensors, are used for object identification and tracking. They can transmit and receive data over long distances, making them ideal for applications such as inventory management, cargo tracking, and more [7].

Analysis of Recent Advances in Radio Frequency Sensor Construction

1. **Radio Frequency Identification (RFID) Sensors**:

Identification and Tracking: RFID sensors are utilized for identifying and tracking objects. They can transmit and receive data over long distances, making them ideal for applications such as inventory management, cargo tracking, and monitoring the status and location of unmanned platforms.

Communication: RFID sensors facilitate communication between unmanned platforms and control centers, allowing real-time exchange of data regarding the platform's status, location, and other important parameters.

2. **Radio Frequency Transmitters and Receivers**:

Command Transmission: RF transmitters are used to send commands from operators to unmanned platforms. This may include commands for movement, stopping, changing direction, and executing special maneuvers.

Data Reception: RF receivers receive data from the platform, including information about its current status, speed, route, and other parameters.

3. **Modulated Waveform**:

Radio frequency technology operates on a modulated waveform in the radio frequency spectrum. This ensures a stable and reliable connection that is not affected by obstacles in the signal path, such as walls or other physical barriers.

Advantages of RF Technology for Ground-based Unmanned Platforms

- Wide Operating Range: RF sensors and remote-control devices have a broad operating range, allowing for platform control over significant distances.
- Obstacle Penetration: Radio signals can penetrate through walls and other physical barriers, ensuring reliable communication even in complex environments.
- High Precision and Reliability: The use of RF sensors ensures high accuracy in object identification and tracking, which is critical for the effective execution of tasks.
- Frequency Flexibility: RF sensors can operate across a wide range of frequencies, making them suitable for use in various communication systems.
- Signal Amplification: RF sensors can amplify signals, which is important when dealing with weak signals or transmitting signals over long distances.

- Signal Filtering: RF sensors can filter out unwanted signals, improving signal quality and reducing interference from other sources.
- Modulation and Demodulation: RF sensors can modulate and demodulate signals, which is essential in communication systems where information needs to be transmitted using a carrier wave.
- Compatibility: RF sensors can work with a wide range of devices, such as antennas, amplifiers, transmitters, receivers, and other radio frequency components.

In conclusion, radio frequency technology for remote control is critical for modern ground-based unmanned platforms, providing reliable communication, precise control, and efficient execution of various tasks.

1.4. Research Problem Statement

In our modern era, we are witnessing remarkable strides in the field of robotic systems designed for terrestrial operations. These advancements are made possible through cutting-edge technologies in mechanics, sensor technology, control algorithms, and artificial intelligence. As a result, these robotic platforms are becoming increasingly versatile, dependable, and efficient.

The landscape of robotic platforms today is diverse, catering to a wide range of tasks and industries. For instance, Boston Dynamics has introduced highly agile and stable robots like Spot and Atlas, renowned for their ability to navigate various terrains. Similarly, Clearpath Robotics has developed the Husky platform, serving applications in logistics and scientific research. Additionally, robots such as ANYmal are demonstrating remarkable effectiveness in challenging environments, including rugged terrains and hazardous zones.

While these advancements are significant, they come with their set of challenges. One crucial aspect to address is enhancing the mobility of these robots across diverse

surfaces to ensure operational efficiency in complex environments. Furthermore, continual advancements in artificial intelligence algorithms and reinforcement learning are essential to achieving full autonomy and ensuring reliable performance.

Looking ahead, there is great promise for further advancements in ground-based robotic systems. Through the integration of state-of-the-art technologies, these systems are poised to become even more efficient and autonomous, opening up new possibilities for applications across various sectors. This progression holds the potential to significantly enhance the quality of life and contribute to societal development.

It's noteworthy that robotics has transitioned from being a mere concept in science fiction to a tangible reality applied in numerous domains. Beyond just Boston Dynamics and Clearpath Robotics, companies worldwide are actively engaged in developing unique robotic platforms with diverse functionalities.

A particularly promising application of ground-based robots lies in the field of medicine. These robots can play a vital role in tasks such as delivering medical supplies or small equipment within hospital premises, thereby reducing the time required for essential operations. Moreover, in scenarios involving remote or hazardous locations, robots can serve in transporting injured individuals or providing preliminary medical assistance.

Looking forward, it is imperative to not only advance the technological capabilities of robots but also consider ethical and societal implications. This includes addressing concerns related to the impact of automation on employment and devising strategies for social welfare. Additionally, ensuring the security and privacy of data in the deployment of robots across various sectors is paramount.

In essence, the development of ground-based robotic systems represents just the beginning of a journey towards a future where robots will play a pivotal role in enhancing human life and driving societal progress.

1.5. Conclusions

The first section provided an overview and analysis of contemporary automatic control systems for ground-based unmanned platforms. Various robotic platforms, their variations, features, and application areas were examined. The functioning principles and adaptability of automatic control systems for ground-based unmanned platforms were discussed, highlighting their crucial role in ensuring effective and reliable operation in dynamic environments.

Special attention was given to remote control technologies, including ultrasound, infrared, and radio frequency sensors. It was determined that these technologies are essential components for ensuring precise and efficient control of unmanned platforms.

Based on the analysis of literature and existing solutions, the main research goal was formulated, and the tasks necessary to achieve this goal were identified. It was noted that the implementation of adaptive control systems would enhance the productivity and reliability of ground-based unmanned platforms.

Thus, the information presented in this section serves as a foundation for further research and development in the field of automatic control of ground-based unmanned platforms. Subsequent sections will delve into more detailed investigation and testing of the developed adaptive control system, as well as its modeling and optimization.

SECTION 2
PRINCIPLES OF OPERATION AND LOGICAL UNIT OF ADAPTIVE ROBOTIC GROUND SYSTEM OF WHEELED TYPE

2.1. Robotic wheeled systems

Adaptive control systems (see in Fig. 2.1.1) are an important advancement in robotics and automation, allowing machines to perform tasks with a high degree of autonomy and reliability. They adjust their behavior in response to changes in the environment or internal state without human intervention. The basic concept of adaptive control involves monitoring, analyzing, and adjusting system parameters in real time to maintain optimal performance in different conditions.

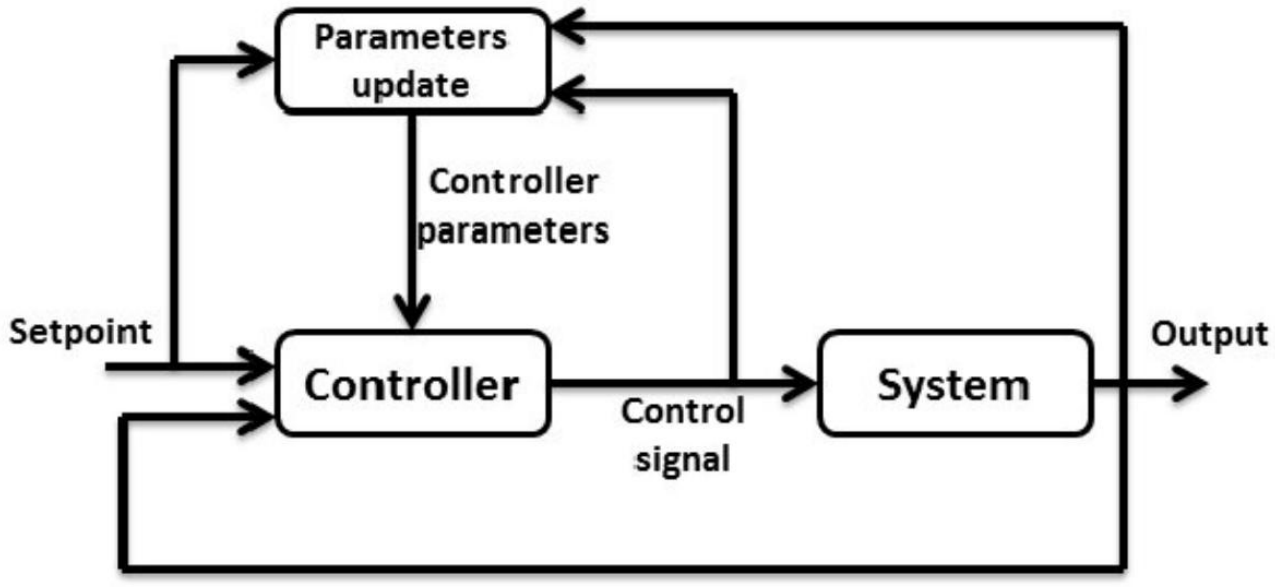


Fig. 2.1.1. Block Diagram of adaptive controller

Key components of adaptive control systems

Sensors and data collection: Adaptive control systems rely on a variety of sensors, such as cameras, lidar, radar, GPS, accelerometers, and gyroscopes, to collect data about the environment and the system's own state.

Data processing and analysis: The collected data is processed using algorithms that can detect patterns, changes, and predict future states. Machine learning methods, neural networks, and statistical analysis are used for this purpose.

Control algorithms: Based on the processed data, control algorithms determine the necessary adjustments to the system. These algorithms provide navigation, obstacle avoidance, speed control, and stability control.

Actuators and execution: The control commands generated by the algorithms are executed by actuators such as motors, brakes, steering mechanisms, or other devices that affect the physical state of the system.

Feedback loops: Adaptive control systems operate on the principle of feedback loops, where the output is constantly monitored and compared to the desired performance. Deviations trigger adjustments to ensure optimal performance [8].

Basic concepts of adaptive algorithms for wheel systems control

Adaptive algorithms for controlling wheeled systems are the basis of their ability to self-regulate.

Sensor data processing algorithms: Collect and process information from various sensors such as lidars, cameras, inertial measurement units (IMUs), ultrasonic sensors, and GPS. This data is needed to understand the environment and the internal state of the vehicle.

Algorithms for analyzing and predicting road surface conditions: Use sensor data to determine road surface characteristics such as friction, bumps, water, ice, or snow. Predict changes in conditions based on weather data and historical records.

Adaptive driving algorithms: Adjust the vehicle's driving parameters after processing data and analyzing road conditions, including speed changes, braking, wheel steering, and steering angles [8-9].

Application of adaptive control in unmanned ground platforms

Unmanned ground platforms (UGPs) or autonomous ground vehicles are the main area of application for adaptive control systems. They are designed to operate without direct human control, making them suitable for a variety of tasks in challenging environments.

Main areas of application:

- Autonomous vehicles: Used to ensure safe and efficient transportation without the need for a driver.
- Military and defense: Performing intelligence, logistics, and support tasks for military operations.
- Agriculture: Automate the processes of tillage, sowing, and harvesting.
- Disaster response and search and rescue: Perform tasks in hazardous environments.
- Mining and construction: Used to automate heavy and dangerous work.

Technological aspects

Combining sensor data: Integrates data from multiple sensors to create a complete view of the environment, increasing the reliability and accuracy of the control system.

Machine learning and artificial intelligence: Uses algorithms to improve decision-making, predict potential problems, and optimize control strategies in real time.

Reliable control algorithms: Designed to operate in conditions of uncertainty and variability, ensuring the stability and reliability of the UGP.

Communication systems: Provide reliable communication between the UGP and control centers for remote monitoring, command updates, and real-time data.

Adaptive control systems are at the forefront of unmanned ground platforms, providing them with the autonomy and reliability to perform complex tasks in a variety of environments. By integrating advanced sensors, data processing methods, and robust control algorithms, these systems allow UGPs to adapt to dynamic conditions, ensuring optimal performance and expanding their application in various industries [10-11].

2.2. Robotic ground system as an object of research

An adaptive robotic ground system (ARGS) is a complex integrated platform that includes hardware and software components designed to automatically control and perform a variety of tasks in different environments. This system consists of a main chassis equipped with various sensors, processors, communication modules, and actuators. ARNS is the subject of research because of its ability to adapt to environmental changes and variable operating conditions.

Adaptive capabilities

The automated decision support system has high adaptive capabilities due to the use of modern machine learning and artificial intelligence algorithms. This allows the system to:

- Automatically adjust its control parameters depending on environmental conditions.
- Detect and avoid obstacles in real time.
- Predict changes in the environment and adapt to them.
- Detect and respond to anomalies, ensuring reliable and safe operation.

This thesis will focus on a robotic ground platform.

Let us design an imaginary robotic platform and give it some specifications and description.

Components of an adaptive robotic platform

Wheels. Our robotic platform uses four wheels to provide movement and maneuverability. The wheels are equipped with electric motors to independently control each wheel.

Frame. The robot's frame is made up of aluminum alloy, which ensures that the platform is lightweight, stable, and durable. It has a design that allows easy access to internal components for maintenance.

Movement mechanism. The platform has an electromechanical movement mechanism that allows you to control the angular velocity of the wheels and provides forward, reverse, and turning movements in place. The use of a differential drive improves the robot's maneuverability [12].

Environment perception system. The robot is equipped with several types of sensors:

- **A laser scanner (lidar)** for three-dimensional scanning of the environment and obstacle detection.
- **High-resolution visual cameras** for perceiving the environment, recognizing objects and processing images.
- **Ultrasonic sensors** to accurately measure the distance to objects and avoid collisions.
- **Infrared sensors** for detecting objects and measuring their temperature.

Control. The robot is equipped with a central processor that is responsible for processing data from sensors and controlling movement. The CPU is integrated with a microcontroller that controls motor drivers and sensors.

The control system also includes specialized processor modules for performing machine learning algorithms and adaptive control in real time.

Communication system. The platform has wireless communication modules (Wi-Fi, Bluetooth, LTE) for remote control and data transmission. The GPS module provides accurate navigation and orientation in space.

Adaptive software. The robot uses machine learning algorithms to analyze sensor data, identify correlations, patterns, and insights.

Adaptive algorithms allow changing control parameters in real time, optimizing the robot's movement and avoiding obstacles.

Prediction systems can predict changes in the environment and identify potential obstacles.

Technical characteristics of the adaptive robotic platform

Weight: 40 kg

Dimensions: 90 cm x 70 cm x 50 cm

Speed: up to 1.5 m/s

Load capacity: 100 kg

Number of degrees of freedom: 4

Environment perception system: laser scanner, visual cameras, ultrasonic and infrared sensors

Description of the adaptive robotic platform

Our adaptive robotic platform is a four-wheeled wheel system. It has a robust aluminum alloy construction and a compact size that allows it to operate in a variety of environments and on different surfaces. The robot is equipped with a laser scanner and visual cameras for environment perception and navigation. It has a payload capacity of up to 100 kg and can move at a speed of up to 1.5 meters per second.

The system uses machine learning algorithms to analyze sensor data and adaptive control, allowing it to effectively avoid obstacles and optimize its movement in real time.

2.3. Logic block

The logic unit of our imaginary robotic platform is responsible for processing signals, making decisions, and coordinating the actions of all system components. It is the central element of the automated control system, ensuring the integration of all functions and ensuring the efficient operation of the platform. The main components of the logic unit and their interaction are described below.

Main components of the logic unit

1. Microprocessor controller

The central computing module that receives and processes data from all sensors and modules, and makes decisions about platform management. It coordinates the work of all system components, executes data processing algorithms, and makes decisions based on the data received.

2. Motor drivers

Interface between the microprocessor controller and the motors, provides control over the speed and direction of rotation of the wheels. Execute commands from the controller to precisely control the movement of each wheel.

3. Temperature and humidity sensors

Collect data on the climatic conditions of the environment. Provides important information for the controller that can affect the platform's algorithms in certain conditions.

4. Distance sensor

Measures the distance to obstacles and objects in the environment. Ensures the safety of the platform by providing information for collision avoidance and route planning.

5. Laser scanner (lidar)

Three-dimensional scanning of the environment for obstacle mapping and navigation. Provides detailed information about the environment for improved navigation and obstacle detection.

6. Visual cameras

Capture and analyze images of the environment. Object recognition, obstacle detection, support for visual navigation algorithms.

7. Wi-Fi module

Ensures wireless data transmission between the platform and a remote operator. Remote control of the platform, monitoring its status and receiving commands [13].

Interaction of logic block components

The microprocessor controller receives data from temperature and humidity sensors, a distance sensor, a laser scanner, and visual cameras. Based on this data, the controller executes information processing algorithms to make decisions about the movement of the platform.

The motor drivers receive commands from the microprocessor controller to control the speed and direction of rotation of the wheels. This allows the platform to move to a given point, adjust speed, and perform maneuvers.

Temperature and humidity sensors continuously transmit information about climate conditions to the microprocessor controller. If the conditions change, the controller can adapt the movement algorithms to ensure stable operation of the platform.

A distance sensor and laser scanner provide the controller with information about the location of obstacles. The controller uses this data to create a map of the environment and plan a route to avoid collisions.

Visual cameras provide additional information about the environment, allowing the controller to recognize objects and refine the platform's route.

The Wi-Fi module provides two-way communication between the platform and the remote operator. The operator can send commands to the platform, receive data on its status, and coordinate its actions in real time.

Block diagram of a logic block:

The microprocessor controller is in the center. Connections to the drivers for the motors, temperature and humidity sensor, distance sensor, laser scanner, visual cameras, and Wi-Fi module extend from it. Interaction between the components is indicated by arrows that indicate the direction of data and command transmission.

This structure ensures high efficiency and allows the system to be adapted to different conditions, providing reliable and safe control of the moving platform.

2.3.1. WiFi module

WiFi, or wireless fidelity (see in Fig. 2.3.1), is a crucial element in modern technology, enabling devices to connect to local networks and the Internet wirelessly through radio waves. The WiFi standard is developed and maintained by the WiFi Alliance, a consortium of electronics and technology manufacturers.

Fig. 2.3.1. WiFi Module

In adaptable robotic ground unmanned systems, the WiFi module is essential for real-time data transmission. This capability is fundamental for remote control, monitoring, and autonomous decision-making processes.

Key Functions and Applications

1. **Real-Time Data Transmission:**
   - Sensor Data: The WiFi module transmits data from onboard sensors like cameras, lidar, radar, and environmental sensors. This data is vital for monitoring the robot's surroundings, detecting obstacles, and navigating complex environments.
   - Telemetry Data: It sends telemetry data, including the robot’s position, speed, battery status, and system health metrics, to remote operators or control centers. This information is crucial for maintaining operational oversight and ensuring optimal performance.
2. **Remote Control and Command:**
   - Operator Commands: Operators can send real-time commands to the robotic system via the WiFi module, such as navigation directives, task-specific instructions, or emergency stop signals.
3. **Data Processing and Analysis:**
   - Edge Computing: Some data processing is performed locally on the robot to reduce latency. The processed data is then transmitted via WiFi to control centers or cloud servers for further analysis, aiding decision-making and long-term data storage.
   - Cloud Computing: For complex data analysis, the WiFi module facilitates data transfer to cloud-based systems where advanced analytics and machine learning algorithms can be applied. This is particularly useful for tasks requiring intensive computation, such as image and pattern recognition or predictive maintenance.

4. **Collaborative Operations:**

   - Swarm Robotics: In multi-robot systems, the WiFi module enables inter-robot communication, crucial for coordinated tasks like area coverage, search and rescue operations, and collective transport of objects.

Technical Specifications and Considerations

1. **Bandwidth and Latency:**

   - High Bandwidth: The WiFi module must support high bandwidth to handle large volumes of data, especially from high-resolution cameras and lidar systems.
   - Low Latency: Low latency is essential for real-time control and decision-making, ensuring prompt system responses to commands and environmental changes.

2. **Range and Coverage:**

   - Extended Range: Effective WiFi-modules require extended range and robust signal strength for operations in large or obstructed areas.
   - Enhanced Coverage: Use of repeaters or mesh networks can enhance coverage, especially in challenging environments like urban canyons or dense forests.

3. **Security:**

   - Data Security: Data transmitted over WiFi must be secured to prevent unauthorized access and tampering. Encryption protocols such as WPA3 and secure authentication mechanisms are implemented to safeguard communication.
   - Regular Updates: Frequent updates and security patches are essential to protect against emerging threats and vulnerabilities.

4. **Power Consumption:**

- Efficient Power Management: WiFi modules can be power-intensive. Optimizing their power consumption helps extend the operational time of the robotic system, especially when running on batteries.

The WiFi-module is a critical component in adaptable robotic ground unmanned systems, enabling reliable and real-time data transmission. Its integration supports essential functions like remote control, real-time monitoring, data processing, and collaborative operations. Addressing technical considerations such as bandwidth, latency, range, security, and power consumption allows these systems to effectively leverage WiFi communication to enhance performance and operational capabilities [14].

2.3.2. Motor Shield

Motor shields are integral components in the design and operation of robotic systems, providing the necessary interface for motor control and actuation. In the context of adaptive ground unmanned robotic systems, motor shields play a crucial role in implementing adaptive control strategies to optimize movement. These strategies are essential for enabling robots to navigate effectively across various terrains and respond to dynamic environmental conditions.

Components and Functionality

1. **Motor Drivers:**

H-Bridge Circuits: These circuits control the direction of motor rotation by changing the current flow through the motor. The H-bridge can make the motor rotate forward or backward by reversing the current direction.

Pulse Width Modulation (PWM): Motor drivers use PWM signals to control motor speed. By adjusting the duty cycle of the PWM signal, the motor shield can precisely modulate motor speed.

2. **Power Management:**

Voltage Regulation: Ensures stable power supply to the motors, protecting them from voltage spikes and fluctuations.

Current Limiting: Prevents damage to the motors and shield by limiting the current that can flow through the motors, especially during stalls.

3. **Protection Circuits:**

Overcurrent Protection: Shuts down the motor if excessive current is detected, preventing damage from short circuits or motor stalls.

Thermal Protection: Monitors motor temperature and shuts down the motor if overheating is detected, protecting both the motors and drivers.

Adaptive Control Strategies

Adaptive control involves real-time monitoring and adjustment of motor parameters to maintain optimal performance. For motor shields, this means dynamically adjusting speed and torque based on feedback from the robot's sensors and environmental conditions.

1. **Feedback Mechanisms:**

Encoders: Provide real-time data on the position, speed, and direction of motor shafts. This information is critical for precise control and adjustment of motor operations.

Current Sensors: Monitor the current consumed by the motors, providing data to prevent overloading and to adjust power supply for improved efficiency.

2. **Control Algorithms:**

PID Control: Proportional-Integral-Derivative (PID) controllers are commonly used to maintain desired motor speed and position. By continuously adjusting motor

inputs based on error values (the difference between desired and actual performance), PID controllers achieve smooth and accurate movement.

Adaptive Algorithms: These algorithms adjust control parameters in real-time to adapt to changing conditions. For example, on rough terrain, the control system might increase torque to maintain speed or decrease it to conserve energy in easier conditions.

3. **Optimization Methods:**

Energy Efficiency: Adaptive control can optimize motor energy consumption by adjusting speed and torque according to task demands and battery status. This is crucial for battery-powered unmanned systems to extend operational time.

Performance Enhancement: By dynamically adjusting motor parameters, the system can improve overall performance, achieving faster response times, smoother acceleration, and more precise movements.

Applications

1. **Terrain Adaptation:**

Rough Terrain: In challenging conditions, adaptive control can enhance traction and stability by adjusting motor power to navigate uneven surfaces and obstacles.

2. **Load Management:**

Variable Loads: The system can adapt to different loads, such as carrying varying weights. By monitoring the load and adjusting motor parameters, the robot ensures stable and efficient movement.

3. **Obstacle Avoidance:**

Real-Time Adjustment: When sensors detect obstacles, the motor shield can quickly adjust motor power to maneuver around them, ensuring continuous and safe operation.

Motor shields are pivotal in enabling adaptive control in robotic systems, providing the necessary interface and control capabilities for optimizing movement.

Through advanced feedback mechanisms, control algorithms, and optimization methods, motor shields enable robots to navigate complex environments effectively and efficiently. This adaptability is crucial for the performance and reliability of unmanned ground platforms across various applications, from industrial automation to autonomous exploration [15].

2.3.3. Arduino UNO

The Arduino UNO is a highly versatile and popular microcontroller platform utilized by developers, hobbyists, and researchers worldwide. Based on the ATmega328P microcontroller from ATMEL, it operates at 16MHz with an 8-bit core, 32KB of flash memory for program storage, and 2KB of SRAM for data storage. The board features a 6-channel 10-bit analog-to-digital converter (ADC) that maps input voltages (0 to 5 volts) into integer values (0 to 1023), providing a resolution of approximately 4.9 mV per unit [16].

Key Features of Arduino UNO for Adaptive Algorithms

One of the key features of the Arduino UNO is its 10-bit ADC, which enables precise analog signal capture from various sensors. This capability is essential for adaptive control systems that rely on real-time sensor feedback to dynamically adjust behavior. Despite its modest processing power, the ATmega328P microcontroller is sufficient for running adaptive algorithms that do not require intensive computation. The 16MHz clock speed supports quick execution of control loops and real-time decision-making.

The board's numerous digital and analog input/output (I/O) pins allow it to interface with a wide range of sensors and actuators. This versatility is crucial for implementing adaptive systems that interact with multiple external devices. Additionally, the Arduino IDE provides a user-friendly environment for programming in a simplified version of C/C++. This ease of use makes it accessible for both beginners and experienced developers to implement and test adaptive algorithms. Furthermore,

Arduino shields, which are stackable add-on boards, can extend the functionality of the Arduino UNO. For adaptive algorithms, motor shields, sensor shields, and communication shields (such as Wi-Fi and Bluetooth) enhance the capabilities of the base board.

Implementing Adaptive Algorithms with Arduino UNO

Adaptive algorithms enable systems to adjust their behavior based on changing inputs and conditions. In robotics, these algorithms can optimize motor control, sensor data processing, and overall system performance. The ADC of the Arduino UNO reads data from analog sensors, such as temperature, light, and proximity sensors. Digital sensors interface through the digital I/O pins, providing the necessary inputs for adaptive algorithms. The Arduino UNO's microcontroller can execute these algorithms within the timing constraints necessary for real-time applications, such as adjusting motor speeds or changing navigation paths based on sensor inputs.

Motor shields connected to the Arduino UNO drive DC motors, stepper motors, or servos. By using PWM (Pulse Width Modulation) signals generated by the Arduino, the speed and direction of the motors can be controlled. Wi-Fi or Bluetooth shields enable the Arduino UNO to send and receive data to and from a central control system or other devices, facilitating the implementation of adaptive algorithms that depend on external data sources [17-18].

Implementation of Control Algorithms

Adaptive filters, such as the Least Mean Squares (LMS) algorithm, adjust their parameters based on input data to optimize performance. This is used for noise cancellation, echo cancellation, and equalization. The Arduino UNO can update filter coefficients in real-time based on input signals, minimizing the mean squared error between the desired output and the actual output. The LMS algorithm can be represented as:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e(n)\boldsymbol{x}(n),$$

where $\mathbf{w}(n)$ is the filter coefficient vector;

μ is the step size (learning rate);

$e(n)$ is the error signal;

$\mathbf{x}(n)$ is the input vector [19].

The Proportional-Integral-Derivative (PID) controller is widely used for feedback control in various systems. By adding adaptive elements, such as the Recursive Least Squares (RLS) algorithm, the PID gains can be adapted based on system behavior, ensuring optimal control performance even when system dynamics change. Implementing simple neural networks on Arduino UNO is challenging due to memory constraints. However, basic feedforward neural networks for tasks like pattern recognition or function approximation can be experimented with. The network should be small and use lightweight activation functions (e.g., ReLU) to fit within memory limits.

2.3.4. The HC-06 Bluetooth Module

The HC-06 Bluetooth module (seen in Fig. 2.3.2) is a widely used component for providing wireless communication between unmanned ground platforms (UGPs) and other devices such as smartphones or computers. The module communicates via UART (serial communication protocol) and typically operates at a default baud rate of 9600 bps. Due to its ease of integration and use, this module enables efficient implementation of serial communication, which is important for real-time control, monitoring and data exchange [20].

Fig. 2.3.2. Bluetooth module HC-06

Wiring the HC-06 to Arduino UNO

To connect the HC-06 module to an Arduino UNO, follow these steps:

- Connect the HC-06 GND to Arduino GND.
- Connect the HC-06 VCC to Arduino 5V.
- Connect the HC-06 TX (Transmit) to Arduino digital pin 4 (RX).
- Connect the HC-06 RX (Receive) to Arduino digital pin 2 (TX).

Note: Use a level shifter or a voltage divider to protect the HC-06 RX pin (which operates at 3.3V) from the 5V output of the Arduino TX pin [21].

Configuring the HC-06

By default, the HC-06 module operates at a baud rate of 9600 bps. To change its settings, use AT commands. Upload the following Arduino sketch to configure the HC-06 via serial communication:

C++ listing:

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(4, 2); // RX, TX
```

```
void setup() {
  Serial.begin(9600);
  Serial.println("Type AT commands!");
  BTSerial.begin(9600);
}
void loop() {
  if (BTSerial.available()) {
    while (BTSerial.available()) {
      Serial.write(BTSerial.read());
    }
  }
  if (Serial.available()) {
    BTSerial.write(Serial.read());
  }
}
```

Open the Serial Monitor in the Arduino IDE, set the baud rate to 9600, and type AT commands (e.g., **AT+BAUD8** to change the baud rate to 115200).

Using the HC-06 in UGPs

Adaptive Filtering

Adaptive filters adjust their parameters based on input data to optimize performance. For example, the Least Mean Squares (LMS) algorithm can be used to reduce noise or cancel echoes. Here is an example code snippet for a simple adaptive filter using LMS:

```
float w = 0.0; // Filter weight
```

```
float mu = 0.01; // Learning rate

void updateFilter(float input, float desired) {
   float error = desired - (input * w);
   w += mu * error * input;
}
```

PID Control with Bluetooth

PID (Proportional-Integral-Derivative) control can be enhanced with adaptive elements to adjust the PID gains based on system feedback. Using Bluetooth, you can wirelessly send new PID parameters to the Arduino, which will adjust the control in real-time.

Neural Networks

Although implementing neural networks on an Arduino UNO is challenging due to memory constraints, simple feedforward networks can be used for tasks such as pattern recognition or function approximation. Keep the network small and use efficient activation functions to fit within the Arduino's memory limits.

Example Application: Adaptive Control System for Ground Platform

Using the Arduino UNO with an HC-06 Bluetooth module, you can create a ground unmanned platform that can be controlled and adapted via a smartphone or PC. The smartphone can send commands to adjust the speed or change the direction of the platform, while the Arduino uses feedback from sensors (such as position or speed sensors) to adaptively control the movement based on environmental conditions.

Using Bluetooth for feedback and adaptive control in ground unmanned platforms based on Arduino UNO opens up numerous possibilities for creating efficient, wireless, and responsive systems. Whether for automation, robotics, or other applications, the combination of Arduino and Bluetooth technology provides a flexible and powerful platform for innovation.

2.4. Sensors for remote control of the robotic system

As already described in Section 1.3, sensors play a key role in providing efficient and reliable remote control of robotic systems. The use of modern sensor technologies allows unmanned ground vehicles (UGVs) to perform complex tasks in various environments, ensuring accurate navigation and obstacle avoidance. The main types of sensors used for these purposes are ultrasonic, infrared, and radio frequency sensors.

In the following subsections, we will discuss in detail the principle of operation and application of two main types of sensors that are often used in remote-controlled robotic systems: the E18-D80NK infrared sensor and the HC-SR04 ultrasonic rangefinder.

2.4.1. Infrared Sensor E18-D80NK

The infrared sensor E18-D80NK (see in Fig. 2.4.1) is an affordable proximity sensor that operates using an infrared emitter and receiver. It is an optical sensor designed for non-contact detection of objects at distances up to 80 cm. This sensor features an integrated infrared emitter and receiver, allowing it to detect objects by emitting modulated infrared light and sensing reflections from nearby objects. The E18-D80NK sensor is equipped with three pins: VCC (connected to a power source, either 5V or 12V depending on the model), GND (ground), and OUT (output signal that changes state when an object is detected) [22].

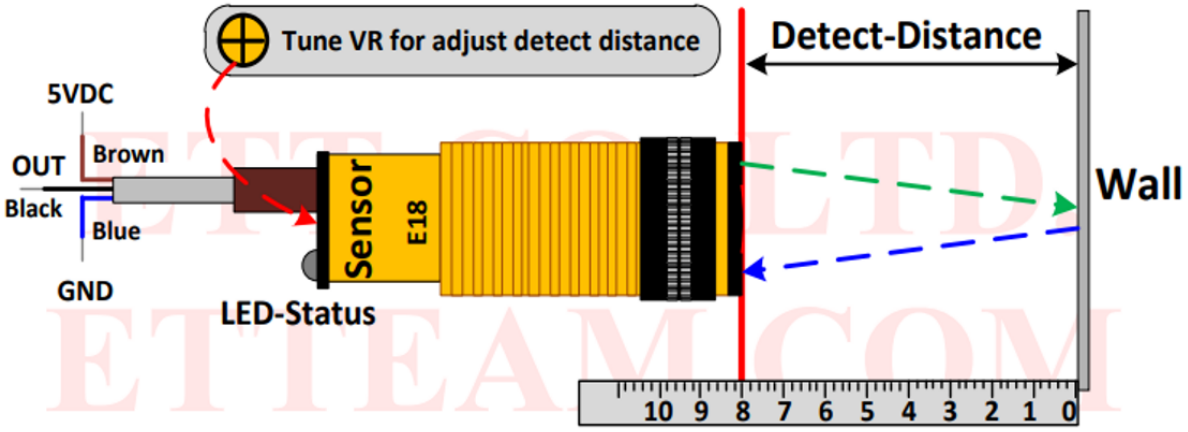


Fig. 2.4.1. The infrared sensor E18-D80NK

When the infrared emitter generates light, it reflects off an object and returns to the receiver. The receiver detects the reflected light and generates a signal on the OUT pin. If an object is within the sensor's detection range, the output signal changes state, typically switching from high to low.

This sensor is particularly valuable in adaptive systems for UGPs due to its precise object detection capabilities, enhancing both safety and efficiency. The E18-D80NK can be mounted on the front or around the perimeter of a GUP to continuously scan the area ahead, identifying obstacles and enabling the controller to decide whether to stop or change direction. This automatic obstacle detection is crucial for navigation in complex environments, preventing collisions and ensuring smooth operation.

One practical application is automatic braking. When the sensor detects an object at a dangerous distance, the controller can immediately stop the platform, avoiding potential collisions. Additionally, in narrow spaces, data from multiple E18-D80NK sensors can be used to identify tight passages and select the optimal path, enhancing the platform's navigational capabilities. The sensor also contributes to environmental mapping by measuring distances to objects, which can be used to create detailed maps for route planning.

The E18-D80NK integrates seamlessly with other sensors such as ultrasonic rangefinders, laser scanners, and cameras, providing a comprehensive understanding of the surroundings. This integration ensures that the GUP can make well-informed decisions based on a variety of sensor inputs, leading to better performance in obstacle avoidance and path planning.

The E18-D80NK offers several advantages for adaptive systems. Its quick response time allows for instant object detection, which is crucial for high-speed platforms. Its low cost makes it an economically viable solution for large-scale systems. Furthermore, the simplicity of integration with microcontrollers like Arduino and Raspberry Pi makes it versatile across different platforms. The lack of moving parts in the sensor increases its reliability and reduces maintenance needs.

However, there are some limitations to consider. The sensor's effectiveness can be impacted by external lighting conditions, being less efficient in bright sunlight or complete darkness. Additionally, its maximum detection range of 80 cm might necessitate the use of additional sensors to ensure comprehensive coverage.

In conclusion, incorporating the infrared sensor E18-D80NK into adaptive systems for ground unmanned platforms significantly enhances the platform's ability to move safely and efficiently. Its rapid response, ease of integration, and cost-effectiveness make it an optimal choice for a wide range of applications, from obstacle avoidance to detailed navigation in challenging environments. This sensor is an essential component for developing advanced, adaptive, and reliable unmanned ground platforms [23].

2.4.2. Ultrasonic Rangefinder HC-SR04

1. Description of the Ultrasonic Rangefinder HC-SR04

The HC-SR04 (see in Fig. 2.4.2) is a sensor used to measure distances to objects using ultrasonic waves. It has four pins: VCC, GND, Trig, and Echo.

- **VCC**: Connects to the power supply (usually 5V).
- **GND**: Ground connection.
- **Trig**: Input signal that activates the ultrasonic pulse.
- **Echo**: Output signal that generates a pulse proportional to the time taken for the ultrasound to return.

Fig. 2.4.2. Ultrasonic Rangefinder HC-SR04

Principle of Operation

1. Triggering the Pulse: The controller sends a short (10-microsecond) high signal to the Trig pin.
2. Generating the Ultrasonic Pulse: The sensor emits an ultrasonic pulse at a frequency of 40 kHz.
3. Propagation and Reflection: The ultrasonic pulse travels through the air, reflects off an obstacle, and returns to the sensor.
4. Receiving the Echo: The sensor detects the reflected signal and sends a pulse to the Echo pin. The duration of this pulse is proportional to the distance to the object.

5. Calculating the Distance: The controller measures the duration of the Echo pulse and calculates the distance using the formula:

$$\text{Distance (cm)} = \frac{\text{Time (μs)} \times 0.03432}{2}.$$

Application of HC-SR04 in Adaptive Systems for Ground Unmanned Platforms (UGP)

Adaptive systems for ground unmanned platforms require accurate and reliable distance measurement to ensure safe navigation, obstacle avoidance, and efficient movement in various environments. The HC-SR04 ultrasonic rangefinder effectively fulfills these requirements due to its specific characteristics.

Key Functions of HC-SR04 in Ground Unmanned Platforms:

1. **Obstacle Detection**:

Installation: The HC-SR04 can be mounted on the front or around the perimeter of the platform to continuously scan the area ahead.

Function: The controller processes data from multiple rangefinders to create a map of the surroundings and avoid collisions.

2. **Automatic Braking**:

Detection: When an obstacle is detected at a dangerous distance, the controller can automatically stop the platform or change its trajectory to prevent collisions.

3. **Navigation in Confined Spaces**:

Data Utilization: By using data from several HC-SR04 sensors, the platform can identify narrow passages and choose the optimal route for movement.

4. **Environmental Data Collection**:

Distance Measurement: Ultrasonic rangefinders can measure distances to objects along the platform's path, enabling the creation of detailed maps for route planning.

5. **Integration with Other Systems**:

Comprehensive Perception: The HC-SR04 can work alongside other sensors (e.g., laser scanners, cameras) to provide a comprehensive perception of the environment.

Advantages of HC-SR04 for Adaptive Systems:

- High Accuracy: Measures distances from 2 cm to 400 cm with an accuracy of ±3 mm.
- Low Cost: Economically viable for large-scale implementation.
- Ease of Integration: Easily connects to microcontrollers and works with various platforms like Arduino and Raspberry Pi.

Limitations. Accuracy may decrease in humid or dusty conditions. The narrow field of view may require multiple sensors to cover the entire perimeter.

Using the HC-SR04 ultrasonic rangefinder in adaptive systems for ground unmanned platforms significantly enhances the platform's ability to navigate safely and efficiently. Its accuracy, ease of integration, and cost-effectiveness make the HC-SR04 an optimal choice for various tasks, from obstacle avoidance to detailed navigation in complex environments. This sensor is essential for developing advanced, adaptive, and reliable unmanned ground platforms [24].

2.5. Adaptive PID Controllers for Ground Unmanned Platforms (UGP)

A PID controller (Proportional-Integral-Derivative controller) is a crucial component in automated control systems, used to maintain desired system parameters by adjusting the control signal. Adaptive PID controllers differ from standard ones as they can modify their parameters (proportional, integral, and derivative coefficients)

based on environmental conditions, enhancing control efficiency in varying circumstances.

Main Components of a PID (Fig. 2.5.1) Controller:

- **Proportional Component (P)**: Determines the reaction of the controller proportional to the current error.
- **Integral Component (I)**: Accounts for the sum of past errors, reducing steady-state error.
- **Derivative Component (D)**: Responds to the rate of change of the error, helping to predict future error and mitigate it.

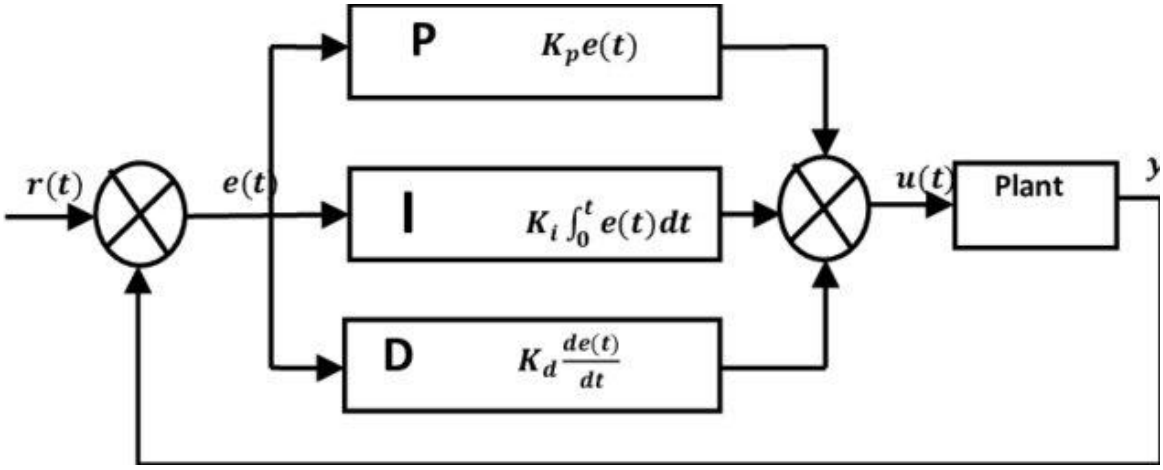


Fig. 2.5.1. The model of a PID controller configuration.

Adaptiveness of the PID Controller: Adaptive PID controllers can automatically adjust their parameters in response to changing environmental conditions, such as load variations, friction coefficients, and temperature changes. This is achieved through adaptive tuning algorithms that analyze the current system state and make appropriate adjustments to the controller parameters [25].

Tuning Parameters of Adaptive PID Controllers

Various methods and algorithms are used to tune the parameters of adaptive PID controllers:

1. **Ziegler-Nichols Method**:
    - Identify the critical proportional coefficient (Ku) and the oscillation period (Tu). Calculate the parameters Kp, Ti, and Td using Ziegler-Nichols formulas for different types of PID controllers.
2. **Self-Tuning Method**:
    - Utilize algorithms that automatically adjust controller parameters based on current system data and feedback. Implement self-learning algorithms to improve controller parameters using historical data and predictions.
3. **Model-Based Adaptive Algorithms**:
    - Develop a mathematical model of the system describing its behavior. Use identification algorithms to continuously update the model based on real data. Adjust controller parameters based on the current system model.
4. **Neural Networks and Artificial Intelligence**:
    - Employ neural networks to predict system behavior and adjust PID controller parameters accordingly. Use machine learning algorithms to optimize controller parameters in real-time.
5. **Fuzzy Logic**:
    - Apply fuzzy logic to create an adaptive controller that considers fuzzy input data and makes decisions based on them. Adjust controller parameters according to fuzzy logic rules, determining the system's response to various situations.

Examples of Tuning Parameters of an Adaptive PID Controller

1. **Adapting to Load Changes**:

- When the load on the platform increases, the system can increase the proportional coefficient (Kp) for quicker error response and raise the integral coefficient (Ki) to reduce steady-state error.

2. **Adapting to Friction Changes**:
    - If the friction coefficient on the surface changes, the adaptive controller can adjust the derivative coefficient (Kd) to avoid oscillations and enhance stability.
3. **Adapting to Temperature Variations**:
    - Temperature changes can affect the characteristics of motors and other system components. The adaptive controller can modify its parameters based on temperature sensors to ensure stable platform operation.

Adaptive PID controllers are vital components of modern automated control systems, providing high precision, reliability, and efficiency. By using various parameter tuning methods, such as the Ziegler-Nichols method, self-tuning algorithms, model-based adaptive algorithms, neural networks, and fuzzy logic, optimal platform control can be achieved under diverse environmental conditions [26].

2.6. Conclusions

In this section, we have examined in detail the principles of operation and the logical block of a wheeled robotic ground system. Based on the analysis of information from various sources, we can draw the following conclusions:

Types of wheeled robotic systems: There are different types of wheeled platforms, such as differential steering systems and all-wheel drive platforms. Each of them has its own characteristics and purpose.

Robotic ground system as an object of study: the selected wheeled ground system has its own technical characteristics, functionality, and main components that should be taken into account when working with it.

Logic unit: The structure of the logic unit includes the main components such as the WiFi module, Motor Shield, Arduino UNO, and Bluetooth module, which interact with each other to control the system.

Remote control sensors: Different types of sensors, such as infrared sensors and ultrasonic rangefinders, are used to measure the distance to obstacles and ensure the safety of the robotic system.

1. **PID controller:** The principle of the PID controller is to control the movement of the platform by adjusting its parameters. This component is important to ensure stable and accurate movement of the system.

Key findings and achievements

We have successfully investigated and described the principles of operation and the logical block of a wheeled robotic ground system. The information obtained will allow us to understand the principles of operation of such systems and develop their further improvement.

Areas for further improvement of the system

Optimization of control: Improving control algorithms to ensure greater accuracy and speed of the system's response to external influences.

Expanding functionality: Adding new modules and sensors to extend the system's functionality, such as visual surveillance systems or autonomous navigation.

Improving reliability: Improving the safety and resilience of the system to prevent emergencies and increase the reliability of operation in various conditions.

The overall goal of future work will be to create efficient and reliable robotic ground systems that can be successfully used in a variety of applications, from industry to research and rescue operations.

SECTION 3
RESEARCH AND TESTING OF THE ADAPTIVE SYSTEM

In this section, we focus on creating an adaptive control system for our imaginary robotic ground platform. Modern robotics requires effective control solutions that allow robots to adapt to changes in the operating environment and perform tasks efficiently.

In the context of ground vehicles, adaptive control ensures optimal and safe movement over a variety of surfaces and conditions, as well as adaptation to the environment. This includes detecting and avoiding obstacles, selecting the optimal route, and adjusting control parameters to suit the current circumstances.

The goal of this chapter is to develop an adaptive control system that includes algorithms for environment perception, route planning, obstacle avoidance, and optimal driving. To achieve this goal, we will use modern methods and technologies such as machine learning, distributed systems, image processing, etc.

During the development process, we will use advanced methods and technologies to create an efficient and flexible control system. We will also consider the possibility of using sensor data to interact with the environment and take into account safety constraints.

This chapter is aimed at expanding the understanding of the principles of developing adaptive control systems and their implementation in ground vehicles. The results obtained can serve as a basis for further research and improvement of control systems, as well as for practical applications in various fields, including industry, autonomous navigation, and rescue operations.

Thus, this work is an important step in the development of an effective adaptive control system for a robotic ground platform that can adapt to changing conditions and perform a variety of tasks in different environments.

3.1. Modeling the behavior of the ground platform

Behavior models define the principles of how a robot interacts with its environment and performs various tasks. They include algorithms that regulate the behavior of the robot in different conditions.

The code by MATLAB (see in Fig. 3.1.1 (a-b)) below implements a simple model of a robot with four wheels, taking into account their size, weight, and type, which were defined in subsection 2.2.

```
% Physical characteristics of the platform
mass = 40; % mass in kg
width = 0.7; % width in meters
length = 0.9; % length in meters
height = 0.5; % height in meters
wheel_radius = 0.1; % wheel radius in meters

% Creating the robot object
robot = struct();

% Setting the physical characteristics
robot.mass = mass;
robot.dimensions = [length, width, height];
robot.wheel_radius = wheel_radius;

% Setting the initial state of the robot
robot.state = [0; 0; 0]; % [x; y; theta]

% Setting the initial wheel speeds
robot.wheel_speeds = [0; 0; 0; 0]; % [v_left_front; v_right_front; v_left_rear;
v_right_rear]

% Function to update the state of the robot
function update_state(robot, dt)
    % Calculate the average speed of the robot
    v = mean(robot.wheel_speeds) * robot.wheel_radius;

    % Update the position of the robot
    robot.state(1) = robot.state(1) + v * cos(robot.state(3)) * dt;
    robot.state(2) = robot.state(2) + v * sin(robot.state(3)) * dt;
end
```

Fig. 3.1.1. (a) Code for implementing a simple model

```
% Parameters of the robot
robot_radius = 0.45; % radius of the robot in meters
max_speed = 1.5; % maximum speed of the robot in m/s

% Parameters of the sensor
sensor_range = 5.0; % range of the sensor in meters
sensor_angle = 60; % angle of view of the sensor in degrees

% Initialization of the robot and the sensor
robot = DifferentialDriveRobot(robot_radius, max_speed);
sensor = RangeSensor(sensor_range, sensor_angle);

% Control loop of the robot
for i = 1:10
    % Getting data from the sensor
    [ranges, angles] = sensor.getRangesAndAngles();

    % Determining the closest obstacle
    [min_range, idx] = min(ranges);

    % If the obstacle is close, the robot turns
    if min_range < robot.radius
        robot.turn(pi/2);
    else
        % If the obstacle is far, the robot moves forward
        robot.moveForward(max_speed);
    end

    % Pause between iterations of the loop
    pause(0.1);
end
```

Fig. 3.1.1. (b) Code for implementing a simple model

In the nexts subsections, three main behavioral scenarios will be discussed:

Obstacle avoidance model: This model includes algorithms that allow the robot to detect and avoid obstacles in its environment. This may include the use of sensors to detect obstacles and navigation algorithms to avoid obstacles. Additionally, integrating a predictive maintenance model could significantly enhance the robot's performance and longevity. Predictive maintenance algorithms can monitor the condition of various components such as motors, sensors, and batteries in real-time. By analyzing data from these components, the model can predict potential failures before they occur, allowing for proactive maintenance and minimizing downtime [27].

Path following model: This model implements algorithms to ensure that the robot follows a given route. GPS or other navigation technologies are used to determine the robot's location, and path planning algorithms provide the optimal route to the target. Alongside path following, integrating predictive maintenance can further enhance the robot's reliability by continuously monitoring the health of its components and predicting potential failures [28].

The "stop before obstacle" model: This model includes algorithms that stop the robot from moving when it encounters an obstacle. Sensors are used to detect obstacles, and control algorithms ensure that the robot stops. In addition to stopping before obstacles, incorporating predictive maintenance algorithms can help prevent unexpected breakdowns and ensure smooth operation in various environments [29].

Each of these models is discussed in detail in the following subsections. They will describe the algorithms used and their implementation in the chosen software environment. This will allow us to evaluate the effectiveness of each model and its impact on the overall performance and efficiency of our robot platform.

3.2. Model "Avoiding obstacles"

To solve the problem of obstacle avoidance in robotics and intelligent vehicles, the A* algorithm is used for global route planning. It works by dividing the area into a grid based on the information about the environment detected by the sensors. To solve the problem of vehicle trajectory planning in a dynamic environment while avoiding obstacles by combining decision-making and motion control modules, this paper proposes an autonomous obstacle avoidance method.

The algorithm assumes that the vehicle can accurately perceive external information about the environment, transforming route planning into a search for obstacle-free zones in a known model of the environment. The general architecture of the proposed automated system is shown in Fig. 3.2.1.

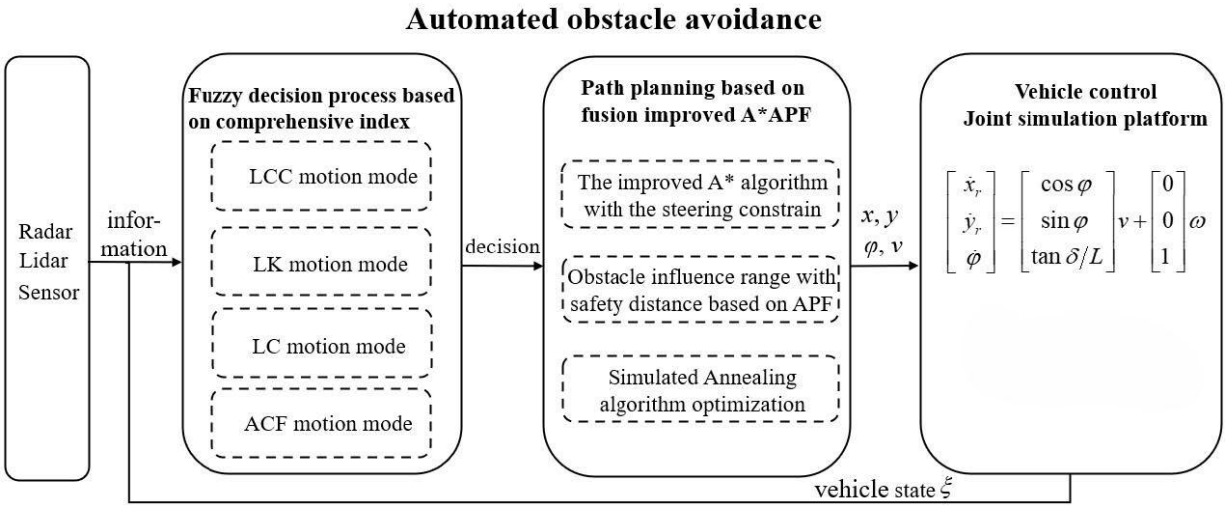


Fig. 3.2.1. Architecture of an automated adaptive system

The A* algorithm combines the advantages of the Dijkstra algorithm and breadth-first search (BFS), making it very effective for heuristic searches in static environments. The algorithm starts by creating an OPEN list and storing eight neighboring meshes as nodes to be computed, also known as current nodes.

These nodes in the OPEN list are analyzed through a cost function, and the path point with the lowest moving cost is selected and removed from the OPEN list. At the same time, a CLOSE list is created to store all the waypoints and obstacle points.

The cost function, which is usually written as

$$f(n) = g(n) + h(n),$$

is a key element of the A* algorithm. Here, g(n) represents the actual cost of moving from the initial position to the intermediate state n, while h(n) represents the estimated cost of moving along the best path from the intermediate state n to the target position. In a conventional algorithm, g(n) is defined by the Euclidean distance and h(n) by the Manhattan distance [30].

```
def a_star(grid, start, goal):
    def heuristic(a, b):
        return abs(a - b) + abs(a - b)

    def reconstruct_path(came_from, current):
        total_path = [current]
        while current in came_from.keys():
            current = came_from[current]
            total_path.append(current)
        return total_path

    open_set = [start]
    came_from = {}
    g_score = {start: 0}
    f_score = {start: heuristic(start, goal)}

    while open_set:
        current = min(open_set, key=lambda x: f_score[x])

        if current == goal:
            return reconstruct_path(came_from, goal)

        open_set.remove(current)

        for neighbor in get_neighbors(current):
            tentative_g_score = g_score[current] + 1

            if neighbor not in g_score or tentative_g_score < g_score[neighbor]:
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score[neighbor] = g_score[neighbor] + heuristic(neighbor, goal)

                if neighbor not in open_set:
                    open_set.append(neighbor)

    return None
```

Fig. 3.2.2. Overall A* algorithm code

3.3. Model "Route following"

The Dynamic A* algorithm is an extension of the A* algorithm specifically designed for adaptive route planning in environments where the cost of travel can change dynamically. This algorithm is especially useful for robots and autonomous systems operating in variable or unknown environments, as it is able to efficiently recalculate the optimal path when conditions change [31].

The main idea of the D* algorithm is to be able to efficiently recalculate paths in real time as the environment changes. To achieve this goal, the following key concepts are used:

1. States and arcs: The graph represents the state space of the robot, where each state is connected by arcs with a transition cost.
2. OPEN list: Contains states that need to be processed to update the path cost.
3. State types: The state can be NEW, OPEN, or CLOSED.

Algorithm of work:

1. PROCESS-STATE (Fig. 3.3.1):

- Selects the state X from the OPEN list with the lowest key value $k(X)$.
- If X is a LOWER state (i.e., $k(X)=h(X)$), its path cost is optimal. Processes neighbors to update their path cost.
- If X is a RAISE state, it first checks the optimal neighbors to possibly reduce the path cost of X. Then it updates the neighbors.

```
def process_state():
    X = min_state()
    k_old = k(X)
    delete(X)
    if X == None:
        return "NO-VAL"
    if k_old < h(X):
        for each neighbor Y of X:
            if r(Y) != NEW and h(Y) <= k_old and h(X) > h(Y) + c(Y, X):
                b(X) = Y
                h(X) = h(Y) + c(Y, X)
    else:
        if (b(X) == X and h(Y) == h(X) + c(X, Y)) or (b(Y) == X and h(Y) > h(X) + c(X, Y)):
            pass
        else:
            insert(X, h(X))
    if b(Y) == X and h(X) > h(Y) + c(Y, X) and k_old != NX:
        for each neighbor Y of X:
            if r(Y) == NEW or b(Y) == X:
                b(Y) = X
                insert(Y, h(X) + c(X, Y))
    if b(N) != X and h(Y) > h(X) + c(X, Y) and t(X) == CLOSED:
        r(Y) = CLOSED
        if h(Y) > k_old:
            insert(Y, h(Y))
    return min_val()
```

Fig. 3.3.1. PROCESS-STATE

2. MODIFY-COST (Fig. 3.3.2):

- Updates the cost of transition between states *X* and *Y*.
- Adds state *X* to the OPEN list if its state was CLOSED.

```
def modify_cost(X, Y, cval):
    c(X, Y) = cval
    return "WIN-VAL"
```

Fig. 3.3.2. MODIFY-COST

3. MOVE-ROBOT (Fig. 3.3.3):

- Initializes all states as NEW and sets the cost of the path to the goal to zero.
- Calls PROCESS-STATE to process the states until the initial path to the goal is found or the path is found to be impossible.
- The robot follows the path to the goal, updating the cost of paths when obstacles are detected.

```
def move_robot(S, G):
    for each state X in the graph:
        r(X) = NEW
    insert(G, 0)
    val = 0
    process_state()
    while r(S) != CLOSED and val != NO-VAL:
        R = S
        process_state()
        R = b(R)
        if t(S) == NEW:
            return "NO-PATH"
        while R != G:
            for each (X, n) such that s(X, Y) != c(X, n):
                val = modify_cost(X, Y, s(X, Y))
                while less(val, COST(.)) and val != NO-VAL:
                    pass
    return "GOAL-REACHED"
```

Fig. 3.3.3. MOVE-ROBOT

Main functions:

PROCESS-STATE: Processes the state from the OPEN list with the lowest key value $k(X)$ and updates the path cost for its neighbors.

MODIFY-COST: Modifies the cost of transition between two states and adds the corresponding states to the OPEN list for further processing.

3.4. Model " Stopping before an obstacle"

The VFH algorithm consists of the following parts:

Cartesian histogram grid. This is a grid (see in Fig. 3.4.1) that is superimposed on the robot's environment, with each cell representing a space. This grid is constructed using data from range sensors such as sonars or laser rangefinders, providing a detailed snapshot of the robot's environment.

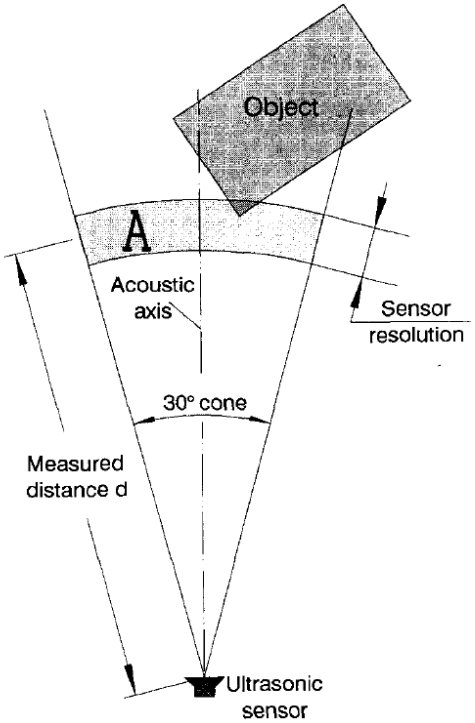


Fig. 3.4.1. Two-dimensional projection of the conical field of view of an ultrasonic sensor. A range reading d indicates the existence of an object somewhere within the shaded region A

Polar histogram. This complex mesh (Fig. 3.4.2) is now simplified into a one-dimensional representation. By condensing the Cartesian histogram around the robot's current position, a polar histogram is created. This condensed representation helps us understand the overall density and distribution of obstacles around the robot.

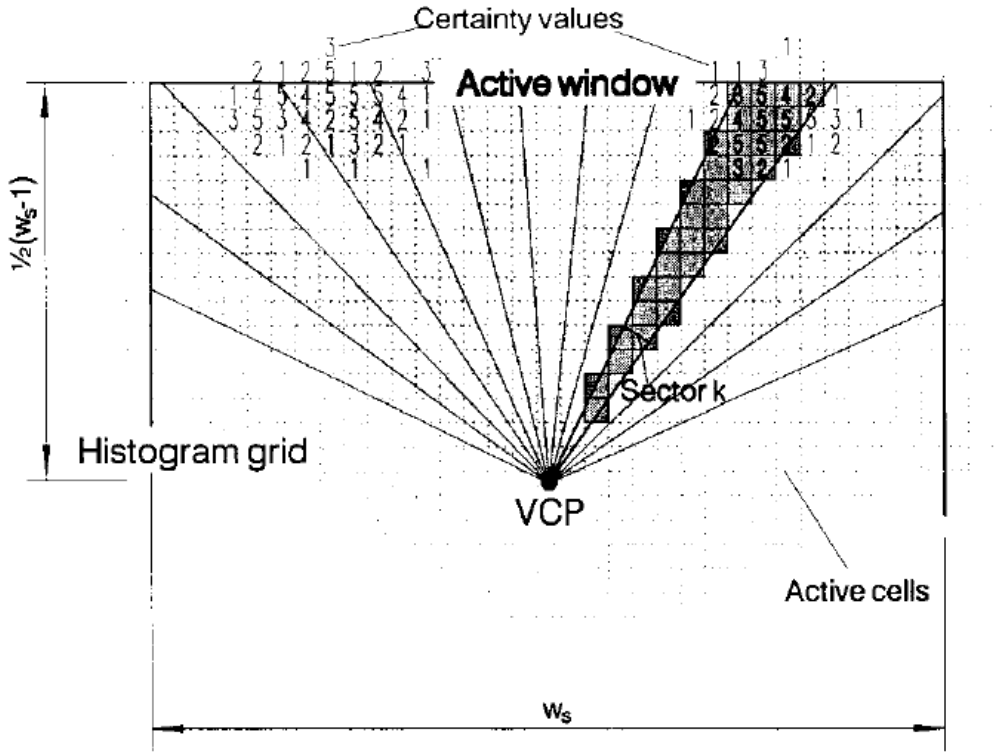


Fig. 3.4.2. Mapping of active cells onto the polar histogram

Candidate valley: While navigating, the robot looks for valleys in the polar histogram where the obstacle density falls below a predefined threshold. These valleys, called candidate valleys, are clusters of consecutive sectors that indicate potential paths with fewer obstacles. The selection of these valleys is influenced by their proximity to the intended direction of travel.

In essence, VFH provides robots with a strategy for perceiving the environment, analyzing potential paths, and navigating efficiently, even in the face of uncertainty and sensor errors.

The VFH algorithm can be implemented using the controller VFH object in MATLAB (Fig, 3.4.3). A brief example of the implementation of this algorithm is given:

```
% Creating the VFH object
vfh = controllerVFH;

% Setting the properties of the object
vfh.RobotRadius = 0.1;
vfh.SafetyDistance = 0.1;
vfh.MinTurningRadius = 0.2;
vfh.DistanceLimits = [0.05 2];

% Function to get data from the laser scanner
function [laserScan] = getLaserScan()
    % Getting data from the laser scanner
end

% Function to get the target direction
function [targetDir] = getTargetDirection()
    % Getting the target direction
end

% Using the object to calculate the obstacle-free steering direction
laserScan = getLaserScan();
targetDir = getTargetDirection();
steeringDir = vfh(laserScan, targetDir);
```

Fig. 3.4.3. Overall VFH algorithm code

Note that the `getLaserScan' and `getTargetDirection' functions need to be replaced with special functions to get the data from the laser scanner and the target direction, respectively [32].

3.5. Conducting experimental research

To perform experimental studies on the behavior of Obstacle Avoidance, an experimental model implemented in MATLAB was used. The experiment process can be divided into the following stages:

1. Determining the map and location of the robot (see in Fig. 3.5.1 (a)): This code loads the map from the exampleMaps.mat file and determines the start and end locations of the robot.

```
% Load the map from the exampleMaps.mat file
filePath = fullfile(fileparts(which('PathPlanningExample')),'data','exampleMaps.mat');
load(filePath);

% Define the map and the robot's start and end locations
map = robotics.BinaryOccupancyGrid(simpleMap, 2);
startLocation = [2 2];
endLocation = [12 2];
```

Fig. 3.5.1 (a). Obstacle Avoidance code

2. Path planning (see in Fig. 3.5.1 (b)): Using the Probabilistic Roadmap (PRM) algorithm to plan a path from the start location to the end location.

```
% Initialize PRM and set the map and the number of nodes
prm = robotics.PRM;
prm.Map = map;
prm.NumNodes = 100;

% Find a path from the start to the end location
path1 = findpath(prm, startLocation, endLocation);
```

Fig. 3.5.1 (b). Obstacle Avoidance code

3. Path visualization (see in Fig. 3.5.1 (c)): Visualizing the found path on the map.

```
% Show the found path on the map
show(prm, 'Map', 'on', 'Roadmap', 'off');
```

Fig. 3.5.1 (c). Obstacle Avoidance code

4. Calculate path length (see in Fig. 3.5.1 (d)): Calculate the length of the found path.

```
% Initialize the length of the path
m1=length(path1(:,1));
s1=0;

% Compute the length of the path
for i=1:m1-1
s1=s1+sqrt((path1(i,1)-path1(i+1,1))^2+(path1(i,2)-path1(i+1,2))^2);
end
```

Fig. 3.5.1 (d). Obstacle Avoidance code

The result of the experiment is that the Roadmap (Fig. 3.5.2) shows that the robot's path is shorter than the one set by the user.

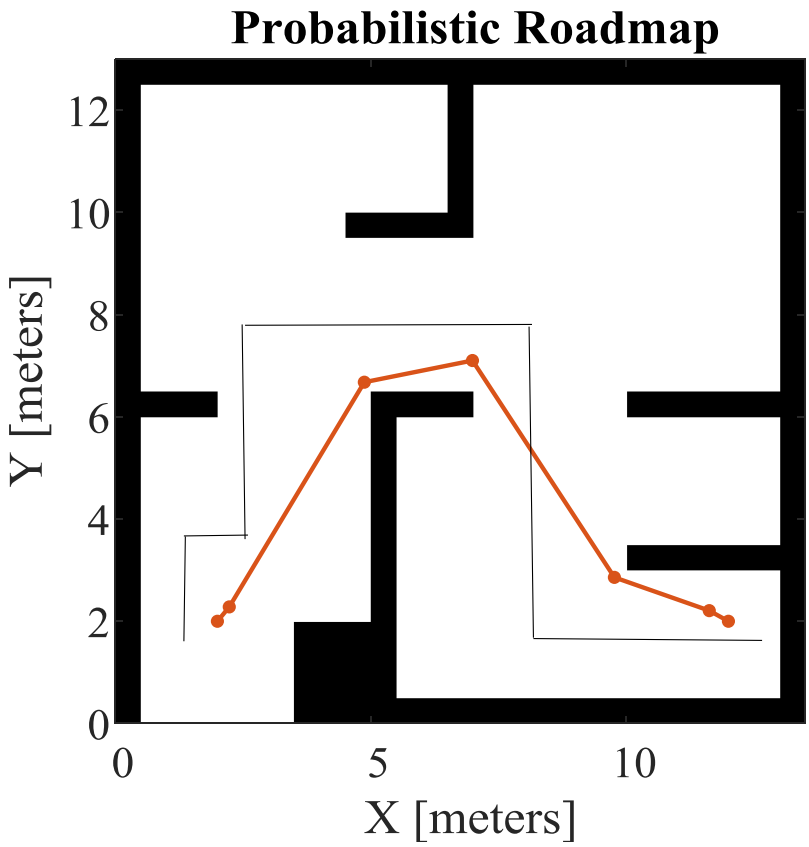


Fig. 3.5.2. Probabilistic Roadmap

The results of the experiment showed that the length of the trajectory is:

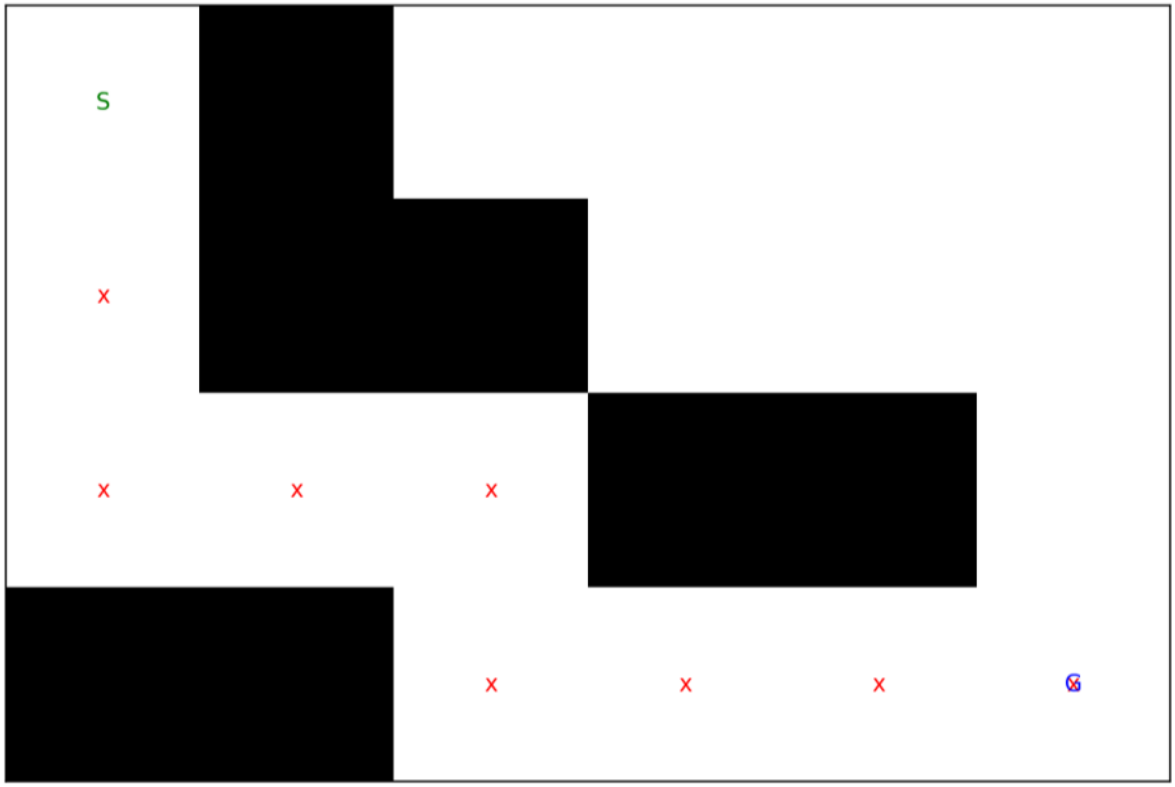
s1 is equal 15.1202 m – optimized path to avoid obstacles, s2 is equal 22 m – the standard path that I set myself for the experiment. This indicates the successful adaptation of the system to route changes and successful obstacle avoidance.

Then make the simulation (Fig. 3.5.3) of the A* algorithm for obstacle avoidance in Python.

```python
import heapq

def heuristic(a, b):
    return abs(a[0] - b[0]) + abs(a[1] - b[1])


def astar(input_grid, start_point, end_point):
    neighbors = [(0, 1), (0, -1), (1, 0), (-1, 0)]
    close_set = set()
    came_from = {}
    g_score = {start_point: 0}
    f_score = {start_point: heuristic(start_point, end_point)}
    open_heap = []

    heapq.heappush(open_heap, (f_score[start_point], start_point))

    while open_heap:
        current = heapq.heappop(open_heap)[1]

        if current == end_point:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start_point)
            path.reverse()  # Reversing the path to get it from start to end
            return path

        close_set.add(current)
        for i, j in neighbors:
            neighbor = current[0] + i, current[1] + j
            tentative_g_score = g_score[current] + 1  # Assuming constant cost for each step

            if 0 <= neighbor[0] < len(input_grid):
                if 0 <= neighbor[1] < len(input_grid[0]):
                    if input_grid[neighbor[0]][neighbor[1]] == 1:  # Assuming 1 represents an obstacle
                        continue
                else:
                    continue
            else:
                continue

            if neighbor in close_set and tentative_g_score >= g_score.get(neighbor, float('inf')):
                continue

            if tentative_g_score < g_score.get(neighbor, float('inf')) or neighbor not in [i[1] for i in open_heap]:
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score[neighbor] = tentative_g_score + heuristic(neighbor, end_point)
                heapq.heappush(open_heap, (f_score[neighbor], neighbor))

    return False

grid_map = [[0, 1, 0, 0, 0, 0],
            [0, 1, 0, 0, 0, 0],
            [0, 1, 0, 1, 0, 0],
            [0, 0, 0, 0, 1, 0]]

start_pos = (0, 0)
end_pos = (len(grid_map) - 1, len(grid_map[0]) - 1)

path_found = astar(grid_map, start_pos, end_pos)

if path_found:
    print("Path found:", path_found)
else:
    print("No path found.")
```

Fig. 3.5.3. Experimental code of A* algorithm

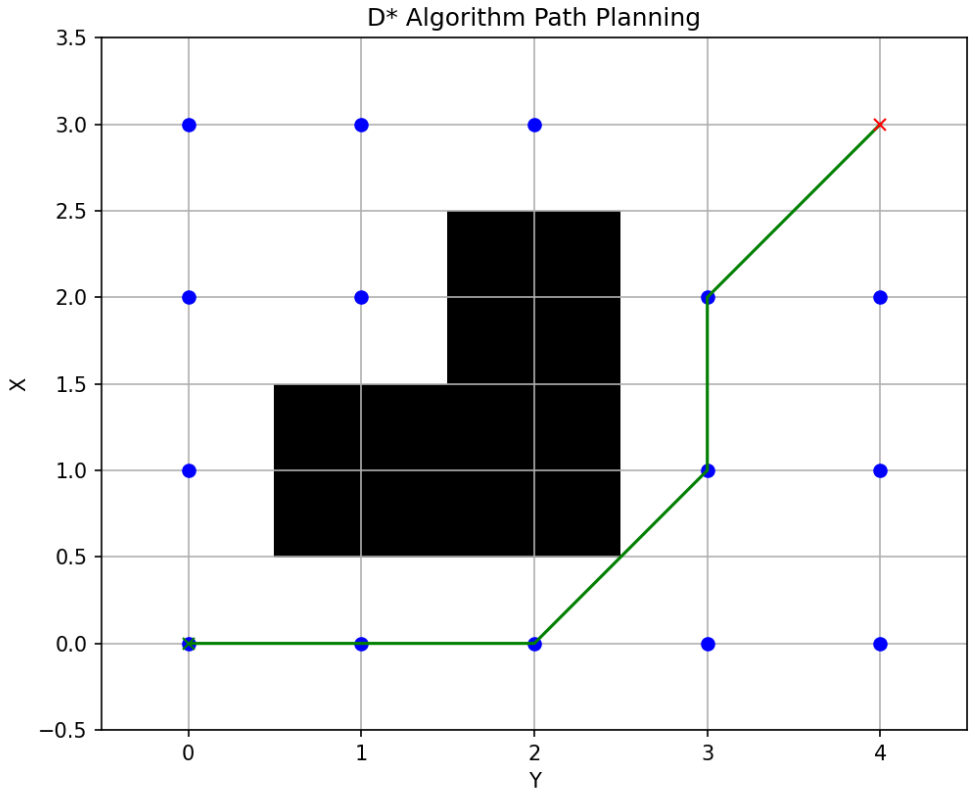
In Fig. 3.5.4, we can see how this algorithm helped the robot to get around the obstacle, and it also gave the path on the map along which it would pass this obstacle. The experiment was successful, the code was implemented correctly.




```
>>> %Run -c $EDITOR_CONTENT
Path found: [(0, 0), (1, 0), (2, 0), (3, 0), (3, 1), (3, 2),
(2, 2), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5)]
```

Fig. 3.5.4. Result of simulation of A* algorithm for my robot

Now we can implement the D* algorithm. Since the code turned out to be cumbersome, the code will be moved to Appendix A. However, the result of the code can be seen in Fig. 3.5.5.




```
Path: [(0, 0), (0, 1), (0, 2), (1, 3), (2, 3), (3, 4)]
```

Fig. 3.5.5. Result of simulation of the D* algorithm for my robot

This code implements the D* algorithm. The D* algorithm works by moving from the start point to the end point, for finding a path on a map updating the cost of movement and taking into account the heuristic distance to the goal. To efficiently select the next point to visit, it uses a priority queue (heapq). On the map, this algorithm displays the path (in blue) that it has found from the start point to the end point. The cost of each step is determined by the distance between the points and is estimated by a heuristic that helps the algorithm find the optimal path, avoid obstacles, and reach the goal.

3.6. Analysis of experimental results

After conducting experiments with the A* and D* algorithms, we came to the following conclusions:

1. Algorithm A*:

- Execution time: For small map scales, the algorithm works quite fast, but for large maps or complex areas, the runtime may increase.
- Path accuracy: The algorithm usually finds the optimal path, taking into account obstacles and minimizing costs.
- Resource intensity: A large number of computations may be required to find a path in complex environments, which leads to an increase in execution time.

2. Algorithm D*:

- Adaptability: This algorithm demonstrates flexibility in changing environments, allowing for real-time path optimization.
- Resource consumption: Compared to A*, D* may have a lower resource consumption due to its adaptability and local optimization capability.

3.7. Conclusion

In this chapter, we have researched and tested an adaptive control system for a ground-based unmanned platform, focusing on key aspects of behavior and control through the lens of adaptive algorithms. The main achievements of this research can be summarized as follows:

1. Implementation of behavioral algorithms:

We have successfully implemented the Obstacle Avoidance, Route Following, and Obstacle Stopping algorithms, which enable the platform to effectively avoid obstacles, follow a given route, and respond to dangerous situations.

2. Testing on maps:

We conducted experiments with these algorithms on maps, which allowed us to visualize their work and evaluate their effectiveness in different conditions.

3. Exploring pathfinding algorithms:

We reviewed and tested the A* and D* algorithms, which allow for efficient wayfinding in complex environments.

4. Conclusions about the results:

We drew conclusions about the performance and efficiency of the different algorithms, including comparing A* and D* algorithms on criteria such as execution time, path accuracy, and resource consumption.

Overall, our research has allowed us to better understand how adaptive algorithms can be used to control a ground-based unmanned platform and how their choice can affect the performance and efficiency of the system.

The overall value of these algorithms to our adaptive system is to ensure the reliability, safety, and efficiency of the ground-based unmanned platform control. They make our system flexible and adaptive to changing conditions, which are key properties in modern robotic systems. Thus, the results of our research emphasize the importance of using adaptive algorithms to improve the functionality and reliability of robotic platform control systems.

CONCLUSION

As part of the thesis, the research and development of an adaptive automatic control system for a ground-based unmanned platform was carried out. The main goal of the work was to create a system capable of autonomous movement in a changing environment, taking into account the presence of various obstacles and the need to adjust the route in real time.

One of the key parts of the work was the study and implementation of algorithms for modeling the behavior of the ground platform. Three main behavioral models were considered: Obstacle avoidance behavior model. Behavioral model "Following a route", Behavioral model "Stopping before an obstacle".

Due to the implementation and testing of these models, high efficiency and reliability of the automatic control system was achieved. In the course of the work, we analyzed the results of the experiments, which made it possible to evaluate the effectiveness of the developed algorithms.

Evaluation of the effectiveness of our adaptive system showed that the A* and D* algorithms allow for efficient route planning, while the VFH algorithm guarantees safety during platform movement due to the possibility of timely stopping in front of obstacles. The implementation of such algorithms allows the platform to adapt to changing environmental conditions, which is a key requirement for modern autonomous systems.

The developed and implemented algorithms are important for the topic of an adaptive automatic control system for a ground unmanned platform. They demonstrate the capabilities of modern technologies in ensuring the autonomy and safety of ground-based unmanned platforms, opening up new prospects for their further development and application.

REFERENCES

1. https://www.dematic.com/en-us/products/amr/ [Electronic resource]
2. https://www.intel.com/content/www/us/en/robotics/autonomous-mobile-robots/overview.html [Electronic resource]
3. Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. Introduction to Autonomous Mobile Robots second edition
4. https://www.linkedin.com/pulse/ai-robot-robotics-artificial-intelligence-malini-shukla [Electronic resource]
5. https://vayuyaan.com/blog/ultrasonic-sensor-in-robotics/ [Electronic resource]
6. Aden, Samuel & Bialas, James & Champion, Zachary & Levin, Eugene & McCarty, Jessica. (2014). Low cost infrared and near infrared sensors for UAVS. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XL-1. 10.5194/isprsarchives-XL-1-1-2014.
7. https://www.rfpage.com/radio-frequency-sensors-types-applications/ [Electronic resource]
8. Dr Lutz Elba & Dr Halit Eren. Process Control. Mineral Processing Design and Operations, 763–816. 2016. doi:10.1016/b978-0-444-63589-1.00020-4
9. https://studfile.net/preview/9400084/page:23/ [Electronic resource]
10. https://www.researchgate.net/publication/278658653_Introduction_to_Adaptive_Control?enrichId=rgreq-74a8408583eb98a08a543581d8bbe76a-XXX&enrichSource=Y292ZXJQYWdlOzI3ODY1ODY1MztBUzoxMTQzMTI4MTE3NDk2MjAzNkAxNjg5NDM5ODE5Mzc5&el=1_x_2 [Electronic resource]
11. https://www.sciencedirect.com/topics/chemical-engineering/adaptive-control-systems [Electronic resource]
12. Z. M. Bi, Y. Lin, and W. J. Zhang. 2010. The general architecture of adaptive robotic systems for manufacturing applications. Robot. Comput.-Integr. Manuf. 26, 5 (October, 2010), 461–470. https://doi.org/10.1016/j.rcim.2010.03.002
13. M. Arbib, A. Hanson. – Cambridge MA. – MIT Press. 1987.

14. Vázquez Leiva, J. H., Vargas Fernández , J. E., Vélez Rodríguez, Y., & Núñez Blanco, W. R. (2024). WIRELESS ADAPTIVE CONTROL OF HEAVY ROTATING PLATFORMS BASED ON ARDUINO AND TCP/IP PROTOCOL. Telemática, 21(4), 59–68. Retrieved from https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/610
15. Г.В.Альошин, С.В.Панченко, С.І.Приходько «Радіоавтоматика в системах зв'язку»
16. https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/ [Electronic resource]
17. Кривонос О.М., Кузьменко Є.В., Кузьменко С.В. Огляд та перспективи використання платформи Arduino Nano 3.0 у вищій школі. Інформаційні технології і засоби навчання. 2016. Т. 56. № 6. С. 77–87.
18. https://www.instructables.com/Add-bluetooth-to-your-Arduino-project-ArduinoHC-06/ [Electronic resource]
19. Amrita Rai, Amit Kumar Kohli; Convergence Analysis of LMS based Adaptive filter. AIP Conf. Proc. 6 November 2010; 1324 (1): 349–351. https://doi.org/10.1063/1.3526230
20. Marhoon, Hamzah. (2021). Implementation of rover tank firefighting robot for closed areas based on arduino microcontroller. Indonesian Journal of Electrical Engineering and Computer Science. 21. 56-63.
21. https://www3.ntu.edu.sg/home/ehchua/programming/arduino/Arduino.html [Electronic resource]
22. https://www.alldatasheet.com/datasheet-pdf/pdf/1221863/ETC1/E18-D80NK.html?gad_source=1&gclid=CjwKCAjw9cCyBhBzEiwAJTUWNcA2CBmleLQWkSSWE75KKe8qwEteHQV8TMRqJChC5iybiW4kFTH0uxoCO9IQAvD_BwE [Electronic resource]
23. https://electropeak.com/learn/interfacing-e18-d80nk-infrared-photoelectric-switch-obstacle-avoidance-distance-ranging-sensor-with-arduino/ [Electronic resource]
24. https://dronebotworkshop.com/hc-sr04-ultrasonic-distance-sensor-arduino/ [Electronic resource]

25. W. Burgard, L. Kavraki, S. Thrun. Cambridge MA – MIT Press: 2005. 68– 603 p.
26. Kushwah, Manoj & Patra, Ashis. (2014). PID Controller Tuning using Ziegler-Nichols Method for Speed Control of DC Motor.
27. Jincong, Yi & Xiuping, Zhang & Zhengyuan, Ning & Quanzhen, Huang. (2010). Intelligent Robot Obstacle Avoidance System Based on Fuzzy Control. 3812 - 3815. 10.1109/ICISE.2009.688.
28. Yang, Liwei, Ping Li, Song Qian, He Quan, Jinchao Miao, Mengqi Liu, Yanpei Hu, and Erexidin Memetimin. 2023. "Path Planning Technique for Mobile Robots: A Review" Machines 11, no. 10: 980. https://doi.org/10.3390/machines11100980
29. Wei, Zhen, "Modular Design of an Educational Robotics Platform" (2016). Graduate Theses - Electrical and Computer Engineering. Paper 7
30. Chu, Liang, Yilin Wang, Shibo Li, Zhiqi Guo, Weiming Du, Jinwei Li, and Zewei Jiang. 2024. "Intelligent Vehicle Path Planning Based on Optimized A* Algorithm" Sensors 24, no. 10: 3149. https://doi.org/10.3390/s24103149 Fig. 3.2.2 shown the code implemented in Python.
31. Stentz, Anthony. (2011). The D* Algorithm for Real-Time Planning of Optimal Traverses.
32. Ulrich, Iwan & Borenstein, Johann. (2000). VFH*: Local Obstacle Avoidance with Look-Ahead Verification. Proceedings - IEEE International Conference on Robotics and Automation. 3. 10.1109/ROBOT.2000.846405.

APPENDIX A

Python code for implementing the D* algorithm

```
import heapq
import math
import matplotlib.pyplot as plt

class Node:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.cost = 0
        self.parent = None

    def __lt__(self, other):
        # Define how Nodes should be compared
        return self.cost < other.cost


class DStar:
    def __init__(self, start, goal, grid):
        self.start = Node(start[0], start[1])
        self.goal = Node(goal[0], goal[1])
        self.grid = grid
        self.open_set = []
        self.closed_set = []

    def heuristic(self, node):
        dx = self.goal.x - node.x
        dy = self.goal.y - node.y
        return math.sqrt(dx**2 + dy**2)

    def get_motion_model(self):
        # 8-connected grid
        return [(1, 0, 1), (0, 1, 1), (-1, 0, 1), (0, -1, 1),
                (-1, -1, math.sqrt(2)), (-1, 1, math.sqrt(2)), (1, -1, math.sqrt(2)), (1, 1, math.sqrt(2))]

    def plan(self):
        heapq.heappush(self.open_set, (0, self.start))

        while True:
            if len(self.open_set) == 0:
                print("Failed to find a path")
                return []

            _, current = heapq.heappop(self.open_set)

            if current.x == self.goal.x and current.y == self.goal.y:
                break

            for dx, dy, cost in self.get_motion_model():
                next_node = Node(current.x + dx, current.y + dy)
                next_node.cost = current.cost + cost

                if not self.verify_node(next_node):
                    continue

                if self.node_in_closed_set(next_node):
                    continue

                if self.node_in_open_set(next_node):
                    continue

                next_node.parent = current
                heapq.heappush(self.open_set, (next_node.cost + self.heuristic(next_node), next_node))

        path = []
        while current is not None:
            path.append((current.x, current.y))
            current = current.parent
```

```
        return path[::-1]

    def verify_node(self, node):
        if node.x < 0 or node.y < 0 or node.x >= len(self.grid) or node.y >= len(self.grid[0]):
            return False

        if self.grid[node.x][node.y]:
            return False

        return True

    def node_in_closed_set(self, node):
        for n in self.closed_set:
            if n.x == node.x and n.y == node.y:
                return True
        return False

    def node_in_open_set(self, node):
        for _, n in self.open_set:
            if n.x == node.x and n.y == node.y:
                return True
        return False

    def draw_map(self, path=[]):
        plt.imshow(self.grid, cmap='Greys', origin='lower')

        for node in self.closed_set:
            plt.plot(node.y, node.x, 'ro')

        for node in self.open_set:
            plt.plot(node[1].y, node[1].x, 'bo')

        if len(path) > 0:
            path_x = [p[0] for p in path]
            path_y = [p[1] for p in path]
            plt.plot(path_y, path_x, 'g-')

        plt.plot(self.start.y, self.start.x, 'gx')
        plt.plot(self.goal.y, self.goal.x, 'rx')

        plt.xlabel('Y')
        plt.ylabel('X')
        plt.title('D* Algorithm Path Planning')
        plt.grid(True)
        plt.show()

# Приклад використання алгоритму D* з малюванням мапи
grid = [[0, 0, 0, 0, 0],
        [0, 1, 1, 0, 0],
        [0, 0, 1, 0, 0],
        [0, 0, 0, 0, 0]]
start = (0, 0)
goal = (len(grid) - 1, len(grid[0]) - 1)

dstar = DStar(start, goal, grid)
path = dstar.plan()
print("Path:", path)

# Малюємо мапу з шляхом
dstar.draw_map(path)
```