ENGINEERING
W Booth School of Engineering
Practice and Technology

McMaster University

SEP 780 -Advance Robotics and Automation

Project Report

on

Group 4 - Autonomous Swarm Robots

Winter 2023

Name	Student ID
1) Meet Patel	(400486035)
2) Jay Patel	(400175525)
3) Vichal Daliya	(400485779)
4) Tushar Bhuva	(400486950)
5) Nikulkumar Dabhi	(400490758)
6) Het Shukla	(400490742)

Table of Contents

Introduction

The term "swarm robotics" refers to a field of robotics that involves coordinating multiple robots to work together to achieve a common goal. In this case, the goal is to perform slow and risky tasks more efficiently than humans. The robots communicate and coordinate among themselves through their artificial swarm intelligence, which is inspired by the behavior of social insects such as ants, bees, and termites.

The use of multiple robots allows for greater efficiency and flexibility compared to a single robot, as the robots can work together to cover more ground and complete tasks faster. Additionally, using robots can be safer than using humans for certain tasks that may be dangerous or difficult to perform.

In this specific project, the group plans to build two mobile robots using Arduino Unos and IR sensors. The use of IR sensors allows the robots to detect their surroundings and navigate autonomously. One robot will act as the master robot, which means that it will have the ability to make decisions and plan a path to complete the end task. The other robot will act as the slave robot, which means that it will follow the master robot's lead and assist in completing the task.

The communication between the master and slave robot is crucial for the success of the project. The master robot will send a signal to the slave robot, instructing it to follow and assist in completing the task. This is known as master and slave communication, and it is a common method used in swarm robotics to coordinate the behavior of multiple robots.

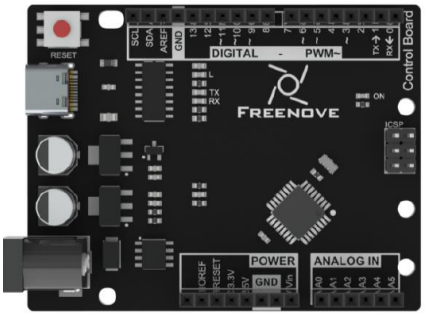
Overall, the goal of building this autonomous multi-robot system is to improve the efficiency and safety of performing certain tasks that may be slow or risky for humans to perform. The use of swarm robotics allows for greater flexibility and coordination between multiple robots, which can lead to faster completion of tasks and a safer working environment.

Problem Statement

“An automated system consisting of multiple robots designed to efficiently and safely complete large, slow, and hazardous tasks”.

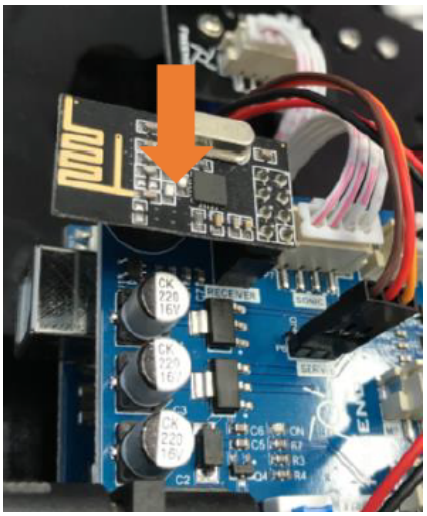
Components

- Arduino - 2 Nos.



The Arduino Uno is a popular microcontroller board that is widely used in the field of electronics and embedded systems. It is based on the ATmega328P microcontroller, which provides a versatile and powerful platform for prototyping and developing various projects. With its user-friendly design and extensive community support, the Arduino Uno offers a simple yet effective way to learn and experiment with programming, electronics, and hardware interfacing. Its compact size, extensive I/O pins, and easy-to-use programming environment make it a go-to choice for hobbyists, students, and professionals alike. Whether you're a beginner or an experienced maker, the Arduino Uno is a versatile and reliable tool that opens up endless possibilities for creative projects.

- RF Transmitter and Encoder Module - 1 No.

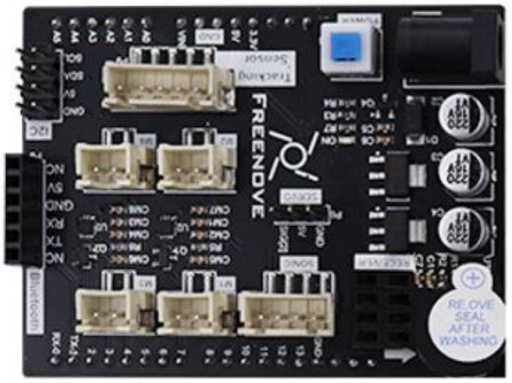


The RF Transmitter and Encoder Module is a key component in wireless communication systems that enables the transmission of data over radio frequency. It is equipped with an encoder that converts digital or analog signals into a format suitable for transmission, and a transmitter that broadcasts the encoded signals into the airwaves. This module is commonly used in remote control applications, wireless sensor networks, IoT devices, and other wireless communication setups. It provides a reliable and efficient means of wirelessly transmitting data, making it an essential component in many modern communication systems.

- Line Tracking Infrared Sensor - 1 No.

IR (Infrared) sensors are electronic devices that detect the presence or proximity of objects based on their emission or reflection of infrared radiation. These sensors work by emitting an infrared beam and measuring the intensity of the reflected or emitted infrared light. They are commonly used for various applications such as obstacle detection, distance measurement, motion detection, and ambient light sensing. IR sensors are widely used in robotics, automation, security systems, and consumer electronics, providing a non-contact and reliable way to detect objects or events based on changes in infrared radiation. They are compact, affordable, and easy to integrate into different systems, making them a popular choice for a wide range of applications.

- L239D Motor Driver Circuit - 2 Nos.



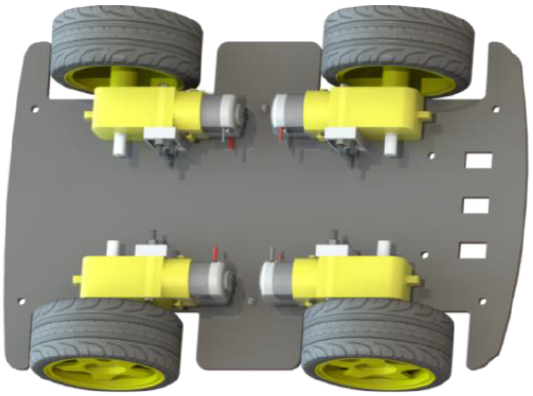
The L239D motor driver circuit is a popular choice for driving DC motors and stepper motors in various robotics and automation projects. It is a dual H-bridge motor driver IC that allows for bidirectional control of two motors simultaneously. The L239D provides high current and voltage capabilities, making it suitable for driving motors with higher power requirements. The circuit typically consists of the L239D IC, capacitors for decoupling, diodes for protecting the circuit from back EMF, and external components for controlling motor speed and direction. It is commonly used in applications such as robotics, automation, and automotive systems, providing a reliable and efficient solution for controlling motors in both forward and reverse directions with precise speed and direction control. The L239D motor driver circuit is widely used by hobbyists, students, and professionals due to its ease of use, versatility, and affordability.

- DC motors - 4 Nos.

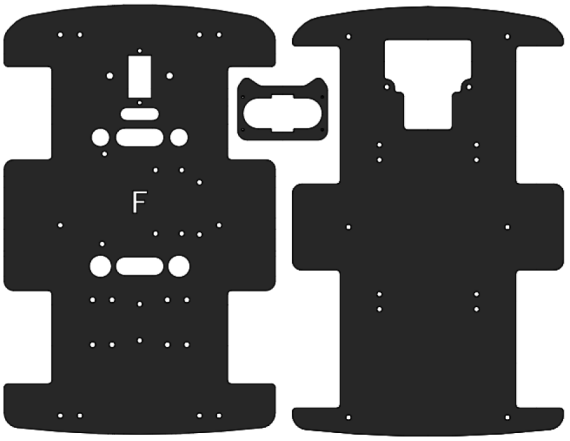
DC (Direct Current) motors are electrical devices that convert electrical energy into mechanical motion. They are commonly used in a wide range of applications, from small hobby projects to large-scale industrial systems. DC motors operate using the principles of electromagnetic induction, where a magnetic field is generated by passing current through a coil of wire, interacting with a permanent magnet or another coil of wire to produce motion. DC motors come in various types, such as brushed and brushless motors, and offer advantages such as simplicity, reliability, and precise speed and torque control. They are widely used in robotics, automotive, aerospace, and consumer electronics, providing a cost-effective and efficient solution for converting electrical energy into mechanical work. DC motors are versatile and can be easily

integrated into different systems, making them a popular choice for a wide range of applications.

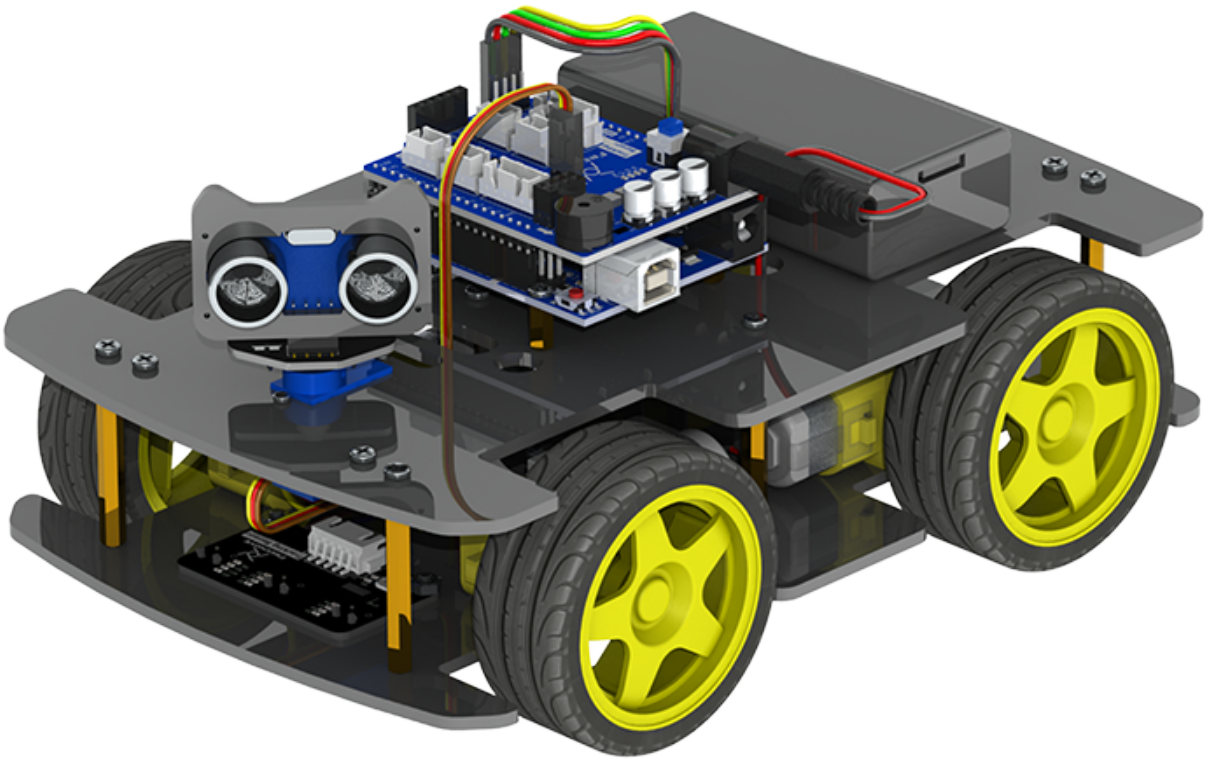
- Wheels for Motors - 4 Nos.



- Chassis - 2 Nos.



- U Clamps - 4 Nos.
- Wire stripper
- Screwdriver
- Connecting wires
- Battery



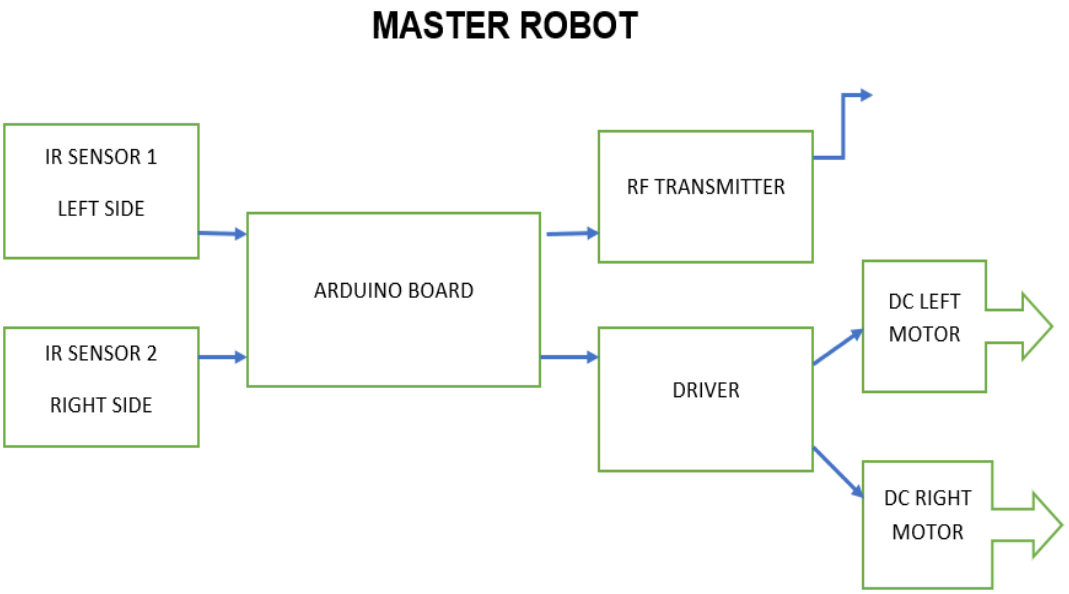
Sensor logic

Left	Middle	Right	Value(binary)	Value(decimal)	Action
0	0	0	000	0	move forward
0	0	1	001	1	Turn Right
0	1	0	010	2	Move Forward
0	1	1	011	3	Turn Right
1	0	0	100	4	Turn Left
1	0	1	101	5	Move Forward
1	1	0	110	6	Turn Left
1	1	1	111	7	Stop

The sensor logic for our robot's movement is based on three sensors: Left, Middle, and Right. These sensors provide binary values (0 or 1) based on their readings. The binary values are then converted into decimal values for ease of processing. Depending on the combination of these sensor values, the robot takes different actions. For example, when all sensors read 0, the robot moves forward. When the Right sensor reads 1 and the other sensors read 0, the robot turns right. When the Middle sensor reads 1 and the other sensors read 0, the robot moves forward. This logic allows the robot to make decisions and perform different actions based on the sensor readings, enabling it to navigate its environment effectively.

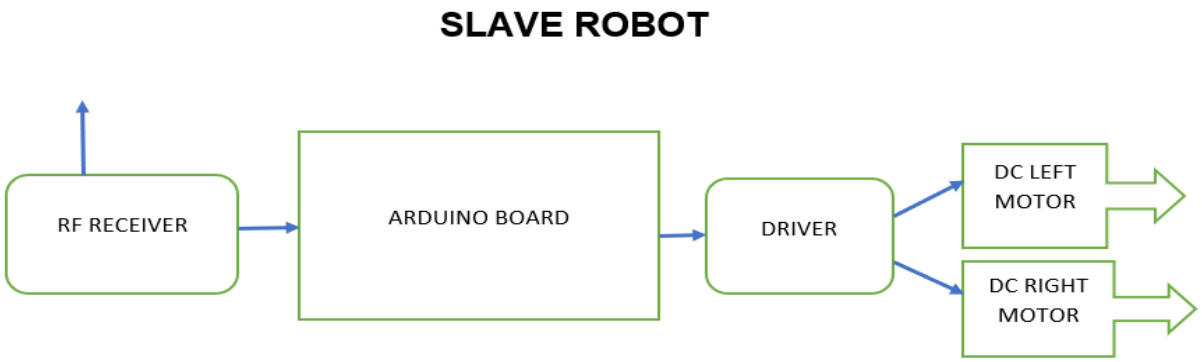
Flow chart

I. Master Robot



The above flowchart gives the idea about how the master robot actually works, the 3 Infrared sensors sense the colour of the line and based on that will create a signal of either 0 or 1. So basically 3 sensor creates 3 combinations of value like 000, 010, 100, etc based on this there is a total of 8 permutation value. These values are stored in the memory and also the robot starts taking action as per the coded task in the sensor logic table like turning right, turning left or moving straight. Here,4 different motors will start to rotate with a decided speed and generate a motion. Simultaneously the code is transferred to slave robots using Radio FrequencyTransmitter so that slave robots receive the signal and work according to the master. The master robot has all the intelligence if the master fails to work all slaves will misbehave.

II. Slave Robot



The above flowchart gives a description of the working of the slave robot, The first function of a slave is that it receives a signal using Radio Frequency Receiver then the signal passes to Arduino Board and based on the programmed logic the robot gives the command to driver to rotate the motors and generate some king of motion output. The main difference between a slave and a master is that a slave does not contain all the sensors and intelligence it has minimalist parts just it follows the command of the master and does the task.

Code

All the code is done using Arduino IDE and Arduino UNO framework.

I. Code for Master Robot

```
#include "Freenove_4WD_Car_for_Arduino.h"
#include <SPI.h>
#include "RF24.h"

#define TK_STOP_SPEED          0
#define TK_FORWARD_SPEED        (90 + tk_VoltageCompensationToSpeed    )

//define different speed levels
#define TK_TURN_SPEED_LV4       (180 + tk_VoltageCompensationToSpeed   )
#define TK_TURN_SPEED_LV3       (150 + tk_VoltageCompensationToSpeed   )
#define TK_TURN_SPEED_LV2       (-140 + tk_VoltageCompensationToSpeed  )
#define TK_TURN_SPEED_LV1       (-160 + tk_VoltageCompensationToSpeed  )

RF24 radio(9, 10);                // define the object to control
NRF24L01
const byte addresses[6] = "Free1";// define communication address which
should correspond to remote control
// wireless communication
int dataWrite[8];

int tk_VoltageCompensationToSpeed;  //define Voltage Speed Compensation

void setup() {
  pinsSetup(); //set up pins
  getTrackingSensorVal();//Calculate Voltage speed Compensation
  radio.begin();                        // initialize RF24
  radio.setPALevel(RF24_PA_MAX);        // set power amplifier (PA) level
  radio.setDataRate(RF24_1MBPS);        // set data rate through the air
  radio.setRetries(0, 15);              // set the number and delay of
retries
```

```
  radio.openWritingPipe(addresses);    // open a pipe for writing
  radio.openReadingPipe(1, addresses);// open a pipe for reading
  radio.stopListening();               // stop listening for incoming
messages
}

void loop() {
  u8 trackingSensorVal = 0;
  trackingSensorVal = getTrackingSensorVal(); //get sensor value

  dataWrite[0] = digitalRead(PIN_TRACKING_LEFT); // save data of
Potentiometer 1
  dataWrite[1] = digitalRead(PIN_TRACKING_CENTER); // save data of
Potentiometer 2
  dataWrite[2] = digitalRead(PIN_TRACKING_RIGHT);  // save data of
direction X of joystick


    if (radio.writeFast(&dataWrite, sizeof(dataWrite)))
  {
    //digitalWrite(led3Pin, HIGH);
  }
  else {
    //digitalWrite(led3Pin, LOW);
  }
  delay(5);

  switch (trackingSensorVal)
  {
    case 0:   //000
      motorRun(TK_FORWARD_SPEED, TK_FORWARD_SPEED); //car move forward
      break;
    case 7:   //111
      motorRun(TK_STOP_SPEED, TK_STOP_SPEED); //car stop
      break;
    case 1:   //001
      motorRun(TK_TURN_SPEED_LV4, TK_TURN_SPEED_LV1); //car turn
      break;
    case 3:   //011
      motorRun(TK_TURN_SPEED_LV3, TK_TURN_SPEED_LV2); //car turn right
      break;
    case 2:   //010
    case 5:   //101
```

```
      motorRun(TK_FORWARD_SPEED, TK_FORWARD_SPEED);  //car move forward
      break;
    case 6:   //110
      motorRun(TK_TURN_SPEED_LV2, TK_TURN_SPEED_LV3); //car turn left
      break;
    case 4:   //100
      motorRun(TK_TURN_SPEED_LV1, TK_TURN_SPEED_LV4); //car turn right
      break;
    default:
      break;
  }
}

void tk_CalculateVoltageCompensation() {
  getBatteryVoltage();
  float voltageOffset = 7 - batteryVoltage;
  tk_VoltageCompensationToSpeed = 30 * voltageOffset;
}

//when black line on one side is detected, the value of the side will be
0, or the value is 1
u8 getTrackingSensorVal() {
  u8 trackingSensorVal = 0;
  trackingSensorVal = (digitalRead(PIN_TRACKING_LEFT) == 1 ? 1 : 0) << 2
| (digitalRead(PIN_TRACKING_CENTER) == 1 ? 1 : 0) << 1 |
(digitalRead(PIN_TRACKING_RIGHT) == 1 ? 1 : 0) << 0;
  return trackingSensorVal;
}
```

Here is the detailed explanation of the above given code:

This is a code written in C++ for controlling the Freenove 4WD Car for Arduino using wireless communication with NRF24L01 modules. The code also incorporates a tracking sensor for detecting black lines on the ground and making the car follow them.The first lines of the code include the necessary header files, which are "Freenove_4WD_Car_for_Arduino.h", which is a library that provides the functions to control the car, "SPI.h", which is the library for communicating with SPI devices, and "RF24.h", which is a library for using NRF24L01 modules. Then, some constants are defined, which represent different speeds for the car's motors when it moves forward or turns, as well as the pin numbers for the tracking sensor.

The next lines define an object of the "RF24" class, which is used to control the NRF24L01 module. The communication address is also defined as "Free1", which is the address that should correspond to the remote control. Then, the necessary setup function is defined, which initializes the pins, calculates the voltage compensation for the speed, and sets up the NRF24L01 module.

The loop function is where the main actions occur. First, the sensor value is obtained by calling the "getTrackingSensorVal()" function. Then, the data from the tracking sensor and the joystick's X-axis direction are saved in an array called "dataWrite", which is sent through the NRF24L01 module using the "radio.writeFast()" function. The delay of 5 milliseconds is added to avoid sending data too frequently.

The switch statement in the loop function checks the value of the tracking sensor and determines the action that the car should take. If the tracking sensor detects a black line in the middle, the car moves forward with a speed defined by "TK_FORWARD_SPEED". If the tracking sensor detects a black line on both sides, the car stops. If the tracking sensor detects a black line only on one side, the car turns in the direction of the black line. The speed of the turn depends on the level of turn defined by the constants. The voltage compensation is added to the speed of the motor to account for changes in battery voltage.

The "tk_CalculateVoltageCompensation()" function calculates the voltage compensation based on the battery voltage, which is obtained by calling the "getBatteryVoltage()" function. The "getTrackingSensorVal()" function returns a byte that represents the value of the tracking sensor. It checks the value of each pin of the tracking sensor and converts it to 0 or 1, which are then combined to form the byte.

Overall, this code is designed to control the Freenove 4WD Car for Arduino using wireless communication and a tracking sensor. The car can move forward, stop, and turn in response to black lines on the ground. The voltage compensation is also included to maintain consistent motor speed despite changes in battery voltage.

II. Code for slave robot

```
#include "Freenove_4WD_Car_for_Arduino.h"
#include <SPI.h>
#include "RF24.h"

#define PIN_SPI_CE      9
#define PIN_SPI_CSN     10

#define TK_STOP_SPEED          0
#define TK_FORWARD_SPEED        (90 + tk_VoltageCompensationToSpeed    )

//define different speed levels
#define TK_TURN_SPEED_LV4       (180 + tk_VoltageCompensationToSpeed   )
#define TK_TURN_SPEED_LV3       (150 + tk_VoltageCompensationToSpeed   )
#define TK_TURN_SPEED_LV2       (-140 + tk_VoltageCompensationToSpeed  )
#define TK_TURN_SPEED_LV1       (-160 + tk_VoltageCompensationToSpeed  )

int tk_VoltageCompensationToSpeed;  //define Voltage Speed Compensation

   RF24  radio(PIN_SPI_CE,  PIN_SPI_CSN); // define an object to control
NRF24L01
  const byte addresses[6] = "Free1";    //set commucation address, same to
remote controller
   int  nrfDataRead[8];               //define an array to save data from
remote controller


void setup() {
  Serial.begin(9600);
  pinsSetup(); //set up pins
  //getTrackingSensorVal();//Calculate Voltage speed Compensation
   if (radio.begin()) {                  // initialize RF24
        radio.setPALevel(RF24_PA_MAX);        // set power amplifier (PA)
level
       radio.setDataRate(RF24_1MBPS);        // set data rate through the
air
```

```
      radio.setRetries(0, 15);             // set the number and delay of
retries
      radio.openWritingPipe(addresses);    // open a pipe for writing
      radio.openReadingPipe(1, addresses);// open a pipe for reading
         radio.startListening();                  // start monitoringtart
listening on the pipes opened
      Serial.println("Start listening remote data ... ");
    }
    else {
      Serial.println("Not found the nrf chip!");
    }
}

void loop() {

delayMicroseconds(8);
    if (radio.available()) {              // if receive the data
      while (radio.available()) {
        //Serial.print("while");// read all the data
        radio.read(nrfDataRead, sizeof(nrfDataRead));
         u8 trackingSensorVal = 0;
         trackingSensorVal = getTrackingSensorVal();

  switch (trackingSensorVal)
  {
    case 0:   //000
      motorRun(TK_FORWARD_SPEED, TK_FORWARD_SPEED); //car move forward
      break;
    case 7:   //111
      motorRun(TK_STOP_SPEED, TK_STOP_SPEED); //car stop
      break;
    case 1:   //001
      motorRun(TK_TURN_SPEED_LV4, TK_TURN_SPEED_LV1); //car turn
      break;
    case 3:   //011
      motorRun(TK_TURN_SPEED_LV3, TK_TURN_SPEED_LV2); //car turn right
```

```
      break;
    case 2:   //010
    case 5:   //101
      motorRun(TK_FORWARD_SPEED, TK_FORWARD_SPEED);  //car move forward
      break;
    case 6:   //110
      motorRun(TK_TURN_SPEED_LV2, TK_TURN_SPEED_LV3); //car turn left
      break;
    case 4:   //100
      motorRun(TK_TURN_SPEED_LV1, TK_TURN_SPEED_LV4); //car turn right
      break;
    default:
      break;
  }
}
    }
}

  void tk_CalculateVoltageCompensation() {
    getBatteryVoltage();
    float voltageOffset = 7 - batteryVoltage;
    tk_VoltageCompensationToSpeed = 30 * voltageOffset;
  }


  u8 getTrackingSensorVal() {
    u8 trackingSensorVal = 0;
        trackingSensorVal = (nrfDataRead[0] == 1 ? 1 : 0) << 2 |
(nrfDataRead[1] == 1 ? 1 : 0) << 1 | (nrfDataRead[2] == 1 ? 1 : 0) << 0;
    return trackingSensorVal;
  }
```

Here is the detailed explanation of the above given code:

The first lines of the code include the necessary header files, which are "Freenove_4WD_Car_for_Arduino.h", which is a library that provides the functions to control the car, "SPI.h", which is the library for communicating with SPI devices, and "RF24.h", which is a library for using NRF24L01 modules. Then, some constants are defined, which represent different speeds for the car's motors when it moves forward or turns, as well as the pin numbers for the SPI pins.

The "tk_VoltageCompensationToSpeed" variable is defined to compensate for changes in battery voltage when controlling the speed of the motors.

The "RF24" object is defined, which is used to control the NRF24L01 module. The communication address is also defined as "Free1", which is the address that should correspond to the remote control. The "nrfDataRead" array is defined to save data from the remote controller.

The setup function initializes the pins, starts the NRF24L01 module, and sets it up to listen for incoming messages. If the module is not found, an error message is printed.

The loop function checks if there is any data available from the remote controller using the "radio.available()" function. If there is data available, the "radio.read()" function reads the data into the "nrfDataRead" array. The value of the tracking sensor is obtained by calling the "getTrackingSensorVal()" function, which converts the data from the remote controller into the value of the tracking sensor. The switch statement in the loop function checks the value of the tracking sensor and determines the action that the car should take. If the tracking sensor detects a black line in the middle, the car moves forward with a speed defined by "TK_FORWARD_SPEED". If the tracking sensor detects a black line on both sides, the car stops. If the tracking sensor detects a black line only on one side, the car turns in the direction of the black line. The speed of the turn depends on the level of turn defined by the constants. The voltage compensation is added to the speed of the motor to account for changes in battery voltage.

The "tk_CalculateVoltageCompensation()" function calculates the voltage compensation based on the battery voltage, which is obtained by calling the "getBatteryVoltage()" function. The "getTrackingSensorVal()" function returns a byte that represents the value of the tracking sensor. It checks the value of each pin of the tracking sensor and converts it to 0 or 1, which are then combined to form the byte.

Working of Swarm Robots

The autonomous multi-robot system operates with a master robot that autonomously determines the path to complete a given task. The master robot then communicates with a slave robot, directing it to follow and cooperatively complete the task. This communication method between the two robots is referred to as master-slave communication.

The basic working principle of swarm robotics is based on the concept of individual agents working together to accomplish a shared goal. Each robot in the swarm is designed to be relatively simple, but when combined with other robots, they can form a complex and robust system. The robots in the swarm rely on local sensing and communication to make decisions and coordinate their actions. This means that each robot has a limited view of its environment and only communicates with its immediate neighbors.

The swarm operates on a set of simple rules that each robot follows to achieve the goal. These rules are programmed into the robots' algorithms and can include behaviours such as obstacle avoidance, path planning, and task allocation. The robots constantly communicate with each other to ensure that they are all following the same set of rules and working towards the same goal.

The use of swarm robotics offers several advantages over traditional robotics approaches. One major advantage is increased flexibility, as the robots in the swarm can adapt to changes in the environment or task requirements without requiring reprogramming or human intervention. Another advantage is scalability, as the number of robots in the swarm can be easily increased or decreased to suit the task at hand.

Swarm robotics has numerous applications, including search and rescue operations, environmental monitoring, and manufacturing processes. For example, a swarm of robots could be deployed to search for survivors in a disaster-stricken area, with each robot using its local sensors and communication to share information about its environment with the other robots in the swarm. Similarly, a swarm of robots could be used in a manufacturing process to work collaboratively on a task, such as assembling a complex product.

Results and Discussion

Swarm robotics is a field of robotics that involves coordinating multiple robots to work together to achieve a common goal. The aim is to perform tasks more efficiently than humans, especially those that are slow and risky. The robots communicate and coordinate with one another through their artificial swarm intelligence, which is inspired by the behavior of social insects like ants, bees, and termites.

This project involves building two mobile robots using a Freenove microcontroller and IR sensors. The IR sensors allow the robots to detect their surroundings and navigate autonomously. One robot acts as the master robot, which makes decisions and plans the path to complete the end task. The other robot is the slave robot, which follows the master robot's lead and assists in completing the task.

Effective communication between the master and slave robot is crucial for the success of the project. The master robot sends a signal to the slave robot, instructing it to follow and assist in completing the task. This is a common method used in swarm robotics to coordinate the behavior of multiple robots.

The primary goal of this autonomous multi-robot system is to improve the efficiency and safety of performing slow or hazardous tasks that may be challenging or risky for humans. The use of swarm robotics allows for greater flexibility and coordination between multiple robots, which can lead to faster completion of tasks and a safer working environment.

In terms of components, this project requires Arduino boards, RF transmitter and encoder modules, IR sensors, L239D motor driver circuits, DC motors, wheels, chassis, U clamps, wire strippers, screwdrivers, connecting wires, and batteries. The IR sensors accurately detect the surroundings and allow the robots to navigate autonomously. The L239D motor driver circuits and DC motors provide sufficient power and control to the robots, allowing them to move smoothly and accurately. The RF transmitter and encoder module provides reliable communication between the master and slave robot. The U clamps and chassis provide a stable and secure platform for the robots to operate on.

The project demonstrated the effectiveness of swarm robotics in completing slow and hazardous tasks. The use of two robots allowed for greater flexibility and coverage, as the robots could divide the work between them and complete the task faster. However, there were also some limitations observed in the project. For instance, the accuracy of the IR sensors was affected by changes in lighting conditions and the background color of the surface. Additionally, the communication range between the master and slave robot was limited, which restricted the coverage area.

In conclusion, the project successfully demonstrated the potential of swarm robotics in completing slow and hazardous tasks. With further research and development, swarm robotics could have a significant impact on a wide range of industries and applications.

Team Members and Duties

The group has distributed tasks among members based on their expertise and interest. Meet Patel is responsible for building the master robot, Jay Patel for building the slave robot, Vichal Daliya for coding the master robot, Tushar Bhuva for coding the slave robot, Nikulkumar Dabhi for integrating gas sensors, and Het Shukla for testing and debugging the code.

Task Number	Task Name	Team Member Responsible
1	Requirement Analysis	Meet , Vichal , Nikul
2	Selection of Algorithm	Jay, Tushar, Het
3	Paper based simulating a blueprint of a project	Vichal, Het , Nikul
4	Sketch the general structure of a solution	All Members
5	Hardware selection integration a Synchronization	Meet, Jay, Tushar
6	Selection of Software Resources	Meet , Nikul
7	Testing of algorithms	Vichal, Tushar
8	Designing and Building GUI	Jay, Het
9	Coding	Meet , Jay
10	Implementation	Vichal , Nikul ,Het
11.	Testing	All Members
12.	Final Deliverables	All members

Challenges & Lessons Learned

The following table summarizes the challenges we have faced during the reporting period and the lessons learned / solutions for each challenge.

Challenge	Lessons learned / solutions
To find a battery which is best suited for the robot	Have to order on time to start working on the robot.
To write a code	Tried various logic and libraries to make the simpler code
communication failure	mitigated this risk by testing and debugging the code thoroughly before demonstrating the robots' capabilities.

Conclusion

In conclusion, a Swarm of Robots is able to avoid obstacles and find their paths autonomously by themselves. They are able to communicate and coordinate with each other and are able to move faster and complete tasks more efficiently than humans. They are more flexible than other robotic systems. Since Swarm robotics can adapt to any environment, it can be used with very few or no limitations. In future using Artificial Intelligence and Machine Learning this type of system's efficiency will be improved as AI can help in decision-making and Machine Learning can take action by adaptive monitoring the surroundings and performing the action.

References

1. http://data.conferenceworld.in/BHIMA/P419-423.pdf
2. https://www-ncbi-nlm-nih-gov.libaccess.lib.mcmaster.ca/pmc/articles/PMC7805972/#:~:text=They%20traditionally%20cooperate%20without%20any,adaptability%2C%20robustness%2C%20and%20scalability.
3. https://ts2.space/en/swarm-robotics-vs-traditional-robotics-a-comparison