LAB REPORT

Course Title	**:**	Robotics and Automation Lab
Course Code	**:**	CSE - 4102
Submitted To	**:**	Rahat Hossain Faisal Associate Professor Computer Science and Engineering Department University Of Barishal
Submitted By	**:**	
Name	:	Md. Moonaz Rahman
Class Roll	:	19CSE023
Exam Roll	:	CSE014/7
Session	:	2018-19
Year	:	4th year, 1st semester
Department	:	Computer Science and Engineering University Of Barishal
Date of Submission	**:**	24-03-2024

Name of the Project: Arduino Line Follower Car.

Objective:

The objective of this project is to design and construct a Line Follower Robot (LFR) using Arduino Uno and infrared sensors to autonomously follow black lines on contrasting surfaces. The goal is to create a versatile LFR capable of adapting to various line-following scenarios with ease by utilizing precise sensor feedback and robust Arduino-based control mechanisms. Through this endeavor, we aim to develop a reliable and efficient system for line tracking that showcases the potential of Arduino-based robotics in real-world applications.

Description of Problem:

The Line Follower car must detect black lines on a contrasting surface to navigate autonomously. This is achieved using infrared sensors placed on the underside of the robot. The sensors detect variations in surface reflectivity, allowing the robot to follow the line's path accurately.

Components:

The components required to implement the robotic car are given below:

1. Arduino UNO
2. Motor Driver Shield
3. IR Sensor (2x)
4. TT Gear Motor (4x)
5. 18650 Li-on Battery (2x)
6. 18650 Battery Holder
7. Jumper Wires
8. Acrylic Sheet
9. DC Power Switch
10. Wheels (4x)

Description of Components:

•Arduino UNO**:** The Arduino UNO is a widely-used microcontroller board. It provides the brainpower for the line follower car, executing programmed instructions to control various components.

•Motor Driver Shield: An expansion board that interfaces with the Arduino UNO to control motor movement, typically containing H-bridge circuits for driving motors in both directions.

•IR Sensor (2x): Infrared sensors used for detecting the contrast between the black line and the surrounding surface. They provide feedback to the Arduino UNO, allowing the robot to follow the desired path autonomously.

•TT Gear Motors: DC motors with gearboxes, offering high torque and low-speed rotation, used for propulsion in the car.

•18650 Li-on Batteries: Rechargeable lithium-ion batteries known for high energy density, providing power for driving motors and electronics.

•18650 Battery Holder: A casing designed to securely hold and connect 18650 batteries, ensuring proper battery management and easy installation.

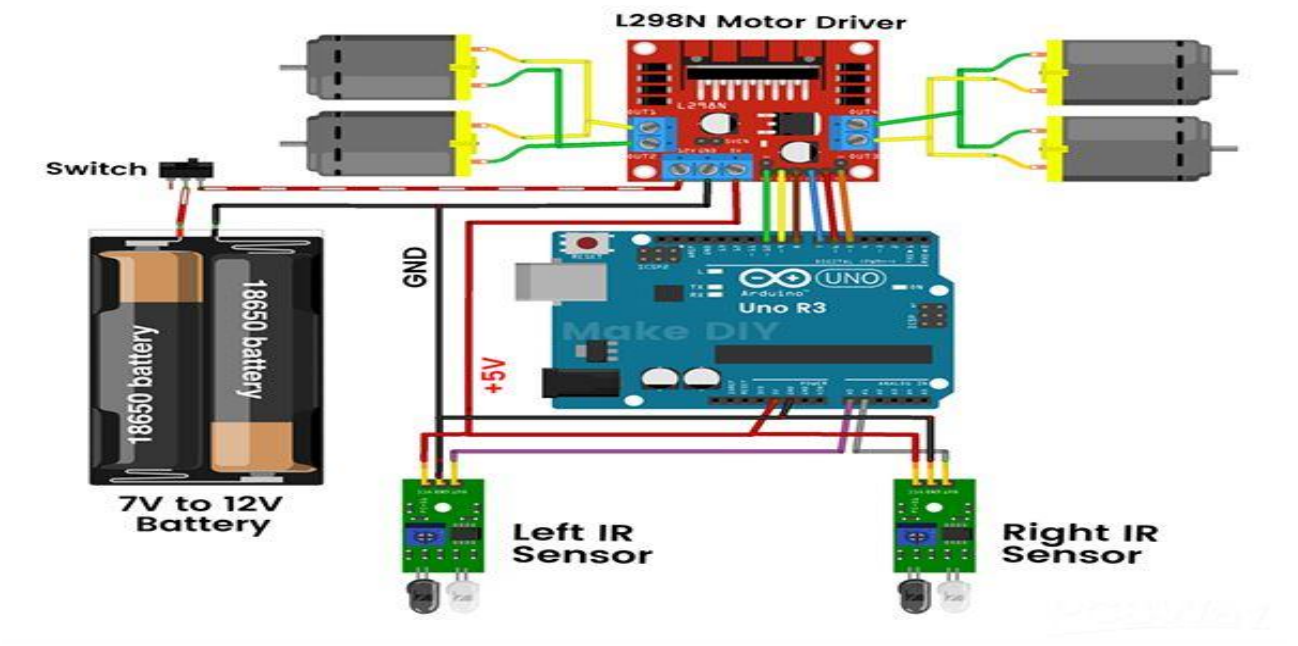
•Jumper Wires: Flexible wires with connectors at each end, used to establish electrical connections between components, facilitating communication with the Arduino UNO.

•Acrylic Sheet: Material for constructing the chassis, providing a sturdy and lightweight structure to hold components together.

•DC Power Switch: A mechanical switch controlling electrical power flow to the car, allowing convenient on/off operation for safety and energy conservation.

•Wheels: Providing traction and enabling movement for the robot. They are typically attached to the shafts of the TT Gear Motors to facilitate motion along the surface

Circuit Design:



Implementation:

<u>Theoritical description:</u>

Experimental Setup:

1. Assemble the chassis using acrylic sheet, wheels, and TT Gear Motors.
2. Attach Motor Driver Shield to Arduino UNO and also the IR sensors.
3. Connect DC Power Switch to power supply circuit.
4. Insert 18650 Li-on Batteries into Battery Holder.

Procedure:

To implement the LFR, assemble the robot chassis and attach the motors, wheels, and infrared sensors. Connect the infrared sensors to digital pins on the Arduino Uno. Define the motor control pins and sensor input pins in the Arduino code. Program the Arduino to read sensor inputs and control motor movements accordingly. The LFR moves forward when both sensors detect white, turns left when the left sensor detects black, turns right when the right sensor detects black, and stops when both sensors detect black.

Operation:

- Sensor Feedback Acquisition: Infrared sensors detect surface reflectivity changes.
- Data Processing: Arduino UNO processes sensor inputs for position determination.
- Motor Control: Arduino sends commands to Motor Driver Shield for motor adjustments.
- Course Correction: Real-time adjustments ensure car stays on the black line.

Code:

```
/including the
libraries
#include
<AFMotor.h>

//defining pins
and variables
#define lefts A4
#define rights A5
#define
frontLeftMotor 1
#define
frontRightMotor 2
#define
rearLeftMotor 3
#define
rearRightMotor 4

//defining motors
AF_DCMotor
motor1(frontLeftM
otor,
MOTOR12_8KHZ)
;
AF_DCMotor
motor2(frontRight
Motor,
MOTOR12_8KHZ)
;
AF_DCMotor
motor3(rearLeftMotor,
MOTOR12_8KHZ);
AF_DCMotor
motor4(rearRightMotor,
MOTOR12_8KHZ);

void setup() {
  //setting the speed of motors
  motor1.setSpeed(200);
  motor2.setSpeed(200);
  motor3.setSpeed(200);
  motor4.setSpeed(200);
  //declaring pin types
  pinMode(lefts,INPUT);
  pinMode(rights,INPUT);
  //begin serial communication
  Serial.begin(9600);
}
}
```

```
void loop(){
  //printing values of the sensors to the serial
monitor
  Serial.print("Left Sensor: ");
  Serial.println(analogRead(lefts));
  Serial.print("Right Sensor: ");
  Serial.println(analogRead(rights));


  //line detected by both sensors
  if(analogRead(lefts) <= 400 &&
analogRead(rights) <= 400){
    //stop all motors
    motor1.run(RELEASE);
    motor2.run(RELEASE);
    motor3.run(RELEASE);
    motor4.run(RELEASE);
  }
//line detected by left sensor
  else if(analogRead(lefts) <= 400 &&
analogRead(rights) > 400){
    //turn left
    motor1.run(BACKWARD);
    motor2.run(FORWARD);
    motor3.run(BACKWARD);
    motor4.run(FORWARD);
  }
  //line detected by right sensor
  else if(analogRead(lefts) > 400 &&
analogRead(rights) <= 400){
    //turn right
    motor1.run(FORWARD);
    motor2.run(BACKWARD);
    motor3.run(FORWARD);
    motor4.run(BACKWARD);
  }
  //no line detected by both sensors
  else {
    //move forward
    motor1.run(FORWARD);
    motor2.run(FORWARD);
    motor3.run(FORWARD);
    motor4.run(FORWARD);
  }
}
```

Results:

Input:

Infrared sensors detect variations in surface reflectivity.

Output:

The Arduino processes sensor inputs and controls motor movements, allowing the robot to autonomously follow black lines on contrasting surfaces.

Conclusion:

Upon successful assembly and programming, the Arduino Line Follower Car is ready to autonomously navigate along black lines on contrasting surfaces, showcasing the effectiveness of Arduino-based robotics in real-world applications.

Name of the Project: RFID Door Lock

Objective:

This experiment's main goal is to design and build an RFID-based door lock system with an Arduino UNO, an RC522 RFID sensor, a Micro Servo, LEDs, and other parts. The objective is to develop a safe and effective access control system that locks and unlocks doors using RFID card data.

Description of Problem:

Based on the principles of RFID technology, the RFID door lock system gives each authorized user a unique ID that is recorded on an RFID card. After these identities are read by the RC522 RFID sensor and verified, the Micro Servo unlocks the door. LEDs improve user feedback by offering visible signs of the door's condition.

Components:

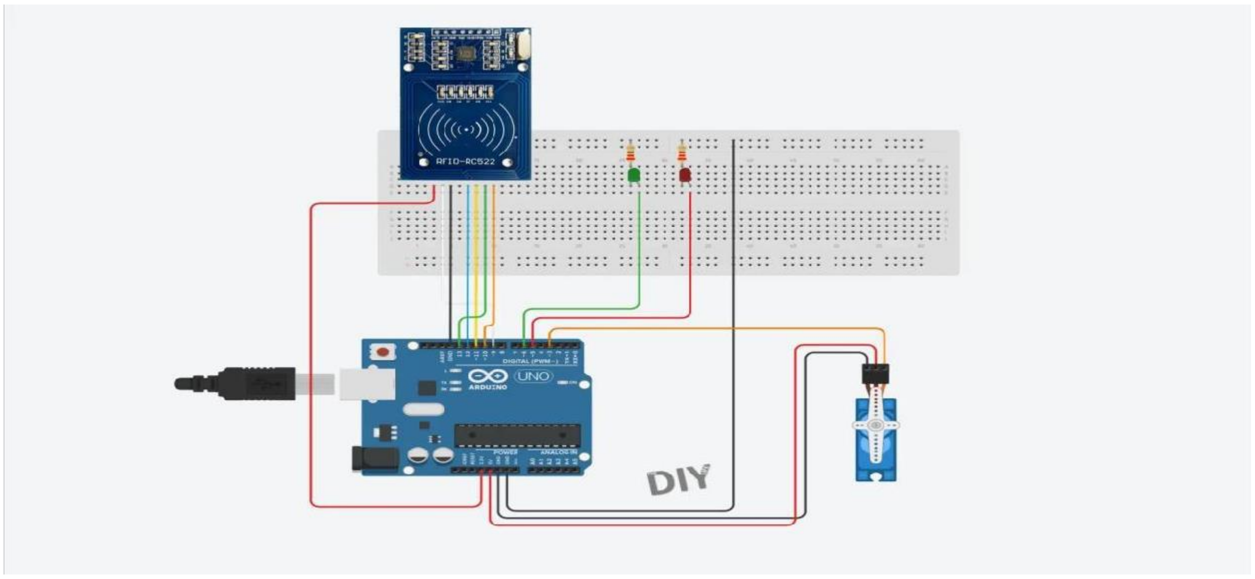
The required components for the project “RFID Door Lock” are:

1. Arduino UNO
2. Arduino Power Supply
3. Breadboard & Jumpers
4. RC522 RFID Sensor
5. Micro Servo
6. LEDs
7. Resistors
8. 3D Printer & Filament

Description of the components:

1. Arduino UNO: Microcontroller platform for controlling LEDs and servos, as well as RFID data management.
2. Arduino Power Supply: Gives the Arduino UNO consistent power.
3. Breadboard and Jumpers: Tools for circuit prototyping and connection creation.
4. RC522 RFID Sensor: Provides access authentication by reading RFID tags.
5. Micro Servo: The door's physical locking and unlocking mechanism.
6. LEDs: Visible indicators that show the lock status or provide feedback from the system.
7. Resistors: Used to safeguard components and restrict the passage of current.
8. 3D Printer and Filament: Used to create unique housing or component mounting options.

Circuit Design:



Implementation:

<u>Experimental Setup:</u>

1. Connect the RFID sensor, LEDs, and Micro Servo to the Arduino UNO using jumpers and the breadboard.
2. Power the Arduino UNO using the Arduino Power Supply.
3. Program the Arduino to read RFID card information and control the Micro Servo.
4. Optional: Use a 3D printer to create a custom lock mechanism.

<u>Procedure:</u>

1. RFID Card Registration:
   - Register RFID cards by reading their unique identifiers and associating them with authorized users.
2. Locking and Unlocking Mechanism:
   - Implement a locking mechanism using the Micro Servo. When a valid RFID card is presented, the servo should rotate to unlock the door.
3. LED Indicators:
   - Utilize LEDs to indicate the status of the door lock (e.g., red for locked, green for unlocked).
4. Optional 3D-Printed Lock:
   - If using a 3D printer, create and install a custom lock mechanism. Ensure it interfaces seamlessly with the Micro Servo.
5. Testing:
   - Test the system by presenting registered and unregistered RFID cards.
   - Observe the LED indicators and servo movement to confirm the lock's status.

Operation:

1. RFID Card Recognition:
   - When an RFID card is presented, the RC522 sensor reads its unique identifier.
2. Verification:
   - The Arduino processes the RFID data and verifies the card against the registered users.
3. Servo Activation:
   - Upon successful verification, the Arduino sends a signal to the Micro Servo to unlock the door.
4. LED Indication:
   - LEDs provide visual feedback, indicating whether the door is locked or unlocked.

Code:

```
#include <SPI.h> #include
<RFID.h> #include <Servo.h>

RFID rfid(10, 9);      //D10:pin of tag reader
SDA. D9:pin of tag reader RST unsigned char
status;    unsigned    char    str[MAX_LEN];
//MAX_LEN is
16: size of the array

String accessGranted [2] =

int accessGrantedSize = 2;
{"310988016", "19612012715"};  //RFID
serial numbers to grant access to
//The number of serial numbers

Servo lockServo;                          //Servo
for locking mechanism int lockPos = 15;
//Locked position limit int unlockPos = 75;
                      //Unlocked position limit
boolean locked = true;
 int redLEDPin = 5; int
greenLEDPin = 6; void
setup()
{
  Serial.begin(9600);    //Serial monitor is only
required  to  get  tag  ID  numbers  and  for
troubleshooting SPI.begin();          //Start    SPI
communication with reader  rfid.init();
//initialization    pinMode(redLEDPin,
```

```
{
  if (rfid.findCard(PICC_REQIDL, str) == MI_OK)
             //Wait for a tag to be placed near the reader
  {
    Serial.println("Card found"); String temp =
    "";
//Temporary variable to store the read RFID number        if
(rfid.anticoll(str) == MI_OK)
//Anti-collision detection, read tag serial number            {


               Serial.print("The card's ID number is


OUTPUT);                  //LED  startup  sequence
pinMode(greenLEDPin,                      OUTPUT);
digitalWrite(redLEDPin, HIGH);            delay(200);
digitalWrite(greenLEDPin,                 HIGH);
delay(200);          digitalWrite(redLEDPin, LOW);
delay(200);        digitalWrite(greenLEDPin, LOW);
lockServo.attach(3);  lockServo.write(lockPos);    //Move
servo into locked position  Serial.println("Place card/tag near
reader...");
}     void loop()
: "); for (int i = 0;
i < 4; i++)
//Record and display the tag serial number
         {              temp = temp + (0x0F &
(str[i] >>
4));                    temp = temp + (0x0F &
str[i]);
          }
          Serial.println (temp); checkAccess (temp);
                                        //Check if the
identified tag is an allowed to open tag
}          rfid.selectTag(str); //Lock card to prevent a
redundant read, removing the line will make the sketch
read cards continually
   }
rfid.halt();
}   void checkAccess (String temp)
//Function to check if an identified tag is registered to
allow access
{      boolean granted = false;                  for (int
i=0; i <= (accessGrantedSize-1); i++)
//Runs through all tag ID numbers registered in
the array
   {          if(accessGranted[i] == temp)
//If a tag is found then open/close the lock
      {
          Serial.println ("Access Granted"); granted =
true;                    if (locked == true)//If the lock is
closed then open it
          {
lockServo.write(unlockPos);
locked = false;                    }              else if
(locked == false)              //If the lock is open
then close it                              {
lockServo.write(lockPos);
locked = true;                    }
digitalWrite(greenLEDPin, HIGH);
//Green LED sequence                    delay(200);
digitalWrite(greenLEDPin, LOW); delay(200);
digitalWrite(greenLEDPin, HIGH);
delay(200); digitalWrite(greenLEDPin,
LOW); delay(200);


       }      }      if (granted == false)                  //If the tag
is not found
   {
       Serial.println ("Access Denied"); digitalWrite(redLEDPin,
HIGH);
//Red LED sequence  delay(200);
digitalWrite(redLEDPin, LOW); delay(200);
                    digitalWrite(redLEDPin, HIGH);
                    delay(200);
digitalWrite(redLEDPin, LOW); delay(200);
   }
}
```

Results:

By doing so the effective registration and recognition of RFID cards, the Arduino-based RFID door lock allowed the door to be locked and unlocked using the Micro Servo. The door's physical mechanism was made unique and secure with the optional 3D-printed lock.

Input:
Users present RFID cards to the RC522 RFID sensor.
Processing: Arduino verifies the card against registered users.
Output: Arduino signals Micro Servo to unlock the door; LEDs indicate the door's status.

Conclusion:

An effective and safe door access system combining RFID technology, servo control, and LED indications was effectively shown by the Arduino-based RFID door lock project. Later releases may concentrate on improving the user interface and security aspects for broad uses.

Name of the Project: Obstacle Avoiding Car

Objective:

The main objective of this project is to design and build an obstacle-avoiding car powered by Arduino. In order to develop an autonomous car that can avoid obstacles in its path, the project aims to demonstrate the integration of numerous components, such as an Arduino UNO, a Motor Driver Shield, TT Gear Motors, a Servo Motor, an Ultrasonic Sensor, and extra accessories.

Description of The Problem:

The ultrasonic sensing and motor control principle underlies the operation of the obstacle- avoiding car. The Ultrasonic Sensor generates ultrasonic waves and timed their return to determine how long it takes. The distance from obstacles is calculated using this information. The TT Gear Motors are controlled by the Arduino UNO for navigation, and the Servo Motor assists with obstacle avoidance and detection.

Components:

The components to implement the project "Obstacle Avoiding Car" includes:

1. Arduino UNO
2. Motor Driver Shield
3. Wheels (4x)
4. TT Gear Motor (4x)
5. Servo Motor
6. Ultrasonic Sensor
7. 18650 Li-on Battery (2x)
8. 18650 Battery Holder
9. Male and Female Jumper Wire
10. Acrylic Sheet
11. DC Power Switch

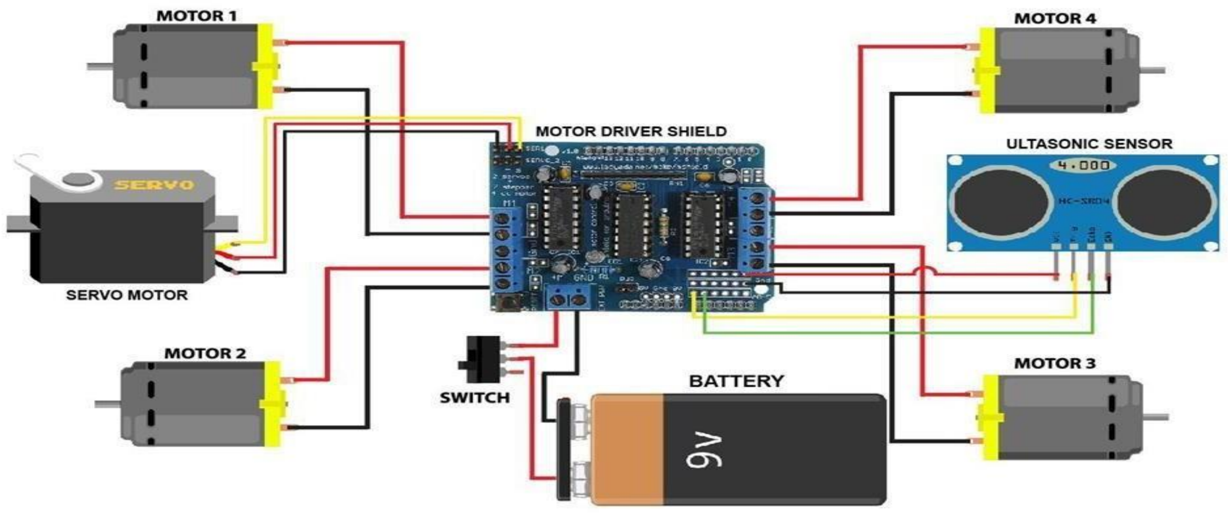
Description of the components:

1. Arduino UNO: A central controller for motor control and sensor data processing.
2. Motor Driver Shield: Controls motor direction and speed.
3. Wheels (4x): Help the car move and have traction.
4. TT Gear Motor (4x): This motor generates torque to move the wheels.
5. Servo Motor: Provides accurate steering and other movement control.

Sixth, the ultrasonic sensor: It uses obstacle detection to activate the avoidance system.

7. 18650 Li-on Battery (2x): The vehicle's power supply.
8. 18650 Battery Holder: Holds the batteries within the chassis securely and arranged.
9. Male and Female Jumper Wire: Enables component electrical connections.
10. Acrylic Sheet: Provides the framework and foundation for mounting parts.
11. DC Power Switch: Provides simple control over the vehicle's power source.

Circuit Design:



Implementation:

Experimental Setup:

1. Assemble the chassis using the acrylic sheet, wheels, and TT Gear Motors.
2. Connect the Ultrasonic Sensor, Servo Motor, and TT Gear Motors to the Arduino UNO using Male and Female Jumper Wires.
3. Attach the Motor Driver Shield to the Arduino UNO.
4. Connect the DC Power Switch to the power supply circuit.
5. Insert the 18650 Li-on Batteries into the Battery Holder.

Procedure:

1. Ultrasonic Sensor Setup:
   - Position the Ultrasonic Sensor on the servo motor to enable a 180-degree scanning range.
2. Motor Control Programming:
   - Program the Arduino to read data from the Ultrasonic Sensor and control the TT Gear Motors for obstacle avoidance.
3. Servo Motor Control:
   - Implement code to control the Servo Motor for scanning and detecting obstacles.
4. Power and Switch Integration:

   - Connect the power supply components, including the 18650 Li-on Batteries and DC Power Switch, to ensure a stable power source.

Operation:

1. Ultrasonic Scanning:
   - The Servo Motor rotates, allowing the Ultrasonic Sensor to scan the surroundings for obstacles.
2. Obstacle Detection:
   - If an obstacle is detected, the Arduino processes the data and sends commands to the TT Gear Motors to navigate and avoid the obstacle.
3. Continuous Navigation:
   - The car continuously scans and navigates, demonstrating its ability to avoid obstacles in real-time.

Code:

```
//OBSTACLE AVOIDING CAR// // Before uploading the
code you have to install the necessary library// //AFMotor
Library https://learn.adafruit.com/adafruit- motorshield/library-
install // //NewPing Library
https://github.com/livetronic/Arduino- NewPing//
//Servo Library https://github.com/arduinolibraries/Servo.git //
// To Install the libraries, go to sketch
>> Include Library >> Add .ZIP File >> Select the
Downloaded ZIP files From the Above links //



#include       <AFMotor.h>
#include       <NewPing.h>
#include <Servo.h>

#define TRIG_PIN A0 #define
ECHO_PIN A1
#define MAX_DISTANCE 200   #define MAX_SPEED
190 // sets speed of DC motors #define
MAX_SPEED_OFFSET 20

NewPing sonar(TRIG_PIN, ECHO_PIN,
MAX_DISTANCE);

AF_DCMotor   motor1(1,   MOTOR12_1KHZ);
AF_DCMotor   motor2(2,   MOTOR12_1KHZ);
AF_DCMotor   motor3(3,   MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);
Servo myservo; boolean
goesForward=false; int
distance = 100; int
speedSet
= 0;   void setup()
{
myservo.attach(10)
;
myservo.write(115)
;         delay(2000);
distance = readPing();
delay(100); distance =
readPing(); delay(100);
distance = readPing();
delay(100); distance =
readPing(); delay(100);
}
void loop() {   int distanceR = 0;
int distanceL = 0; delay(40);
if(distance<=15)
{       moveStop();
delay(100);
moveBackward();
delay(300);
moveStop();
delay(200);
distanceR =
lookRight();
delay(200);
distanceL = lookLeft(); delay(200);
if(distanceR>=distanceL)                    {
turnRight();                    moveStop();
}else          {            turnLeft();
moveStop();            }
  }else  {            moveForward();  }
distance = readPing();
}  int lookRight() { myservo.write(50);
                                delay(500); int distance =
readPing(); delay(100);
                                myservo.write(115);
return distance;
}  int lookLeft() { myservo.write(170);
                                delay(500); int distance =
readPing(); delay(100);         myservo.write(115);
return distance;           delay(100);
}  int readPing() { delay(70);
                          int cm =
sonar.ping_cm();  if(cm==0)
{           cm = 250;          }       return
cm;
}     void     moveStop()     {
motor1.run(RELEASE);
motor2.run(RELEASE);
motor3.run(RELEASE);
motor4.run(RELEASE);
    }           void moveForward() {               if(!goesForward)
{           goesForward=true;
motor1.run(FORWARD);
motor2.run(FORWARD);              motor3.run(FORWARD);
motor4.run(FORWARD);                        for (speedSet = 0;
speedSet < MAX_SPEED; speedSet +=2) // slowly bring the
speed up to avoid loading down the batteries too quickly  {
motor1.setSpeed(speedSet);
motor2.setSpeed(speedSet);
motor3.setSpeed(speedSet);
motor4.setSpeed(speedSet);                      delay(5);               }
    }
}


  void moveBackward() {                   goesForward=false;
motor1.run(BACKWARD);
motor2.run(BACKWARD);            motor3.run(BACKWARD);
motor4.run(BACKWARD);            for (speedSet = 0;
speedSet < MAX_SPEED; speedSet +=2)
// slowly bring the speed up to avoid
loading down the batteries too quickly
                             {
motor1.setSpeed(speedSet);
motor2.setSpeed(speedSet);
motor3.setSpeed(speedSet);
motor4.setSpeed(speedSet); delay(5);  }
}         void
turnRight() {
motor1.run(FORWA
RD);
motor2.run(FORWA
RD);
motor3.run(BACKW
ARD);
motor4.run(BACKW
ARD);
delay(500);
motor1.run(FORWA
RD);
motor2.run(FORWA
RD);
motor3.run(FORWA
RD);
motor4.run(FORWA
RD);
}         void turnLeft()
{
motor1.run(BACKW
ARD);
motor2.run(BACKW
ARD);
motor3.run(FORWA
RD);
motor4.run(FORWA
RD);
delay(500);
motor1.run(FORWA
RD);
motor2.run(FORWA
RD);
motor3.run(FORWA
RD);
motor4.run(FORWA
RD);
}
```

Results:

With the help of an integrated ultrasonic sensor, servo motor, and TT gear motor, the Arduino Obstacle Avoiding Car effectively displayed its capacity to recognize and avoid obstacles. For independent operation, the 18650 Li-on Batteries offered a dependable power source.

Input:

Ultrasonic Scanning: Servo Motor rotates, enabling Ultrasonic Sensor to scan for obstacles.

Processing:

Obstacle Detection: Arduino processes data, commands TT Gear Motors for navigation.

Output:

Continuous Navigation: Car demonstrates real-time obstacle avoidance by continuously scanning and navigating.

Conclusion:

The integration of motor navigation, servo control, and ultrasonic sensing for autonomous obstacle avoidance was successfully demonstrated by the Arduino Obstacle Avoiding Car project. Enhancements in the future can concentrate on enhancing sensor accuracy and looking into new features for advanced navigation.

Name of the Project: Bluetooth Control Car

Objective:

This experiment's main goal is to use an Arduino UNO to design and build a robot car that can be controlled via Bluetooth. In order to produce a wirelessly operated robot car, the project intends to illustrate the integration of Arduino UNO, Bluetooth Module HC-05, DC Geared Motors, L298N H-Bridge Motor Driver, and other components.

Description of The Problem:

The Arduino UNO and a Bluetooth-enabled device, like a smartphone or tablet, communicate wirelessly in order for the Bluetooth Robot Car to function. Through the L298N Motor Driver, the Arduino UNO regulates the DC Geared Motors after processing the signals received from the Bluetooth Module HC-05. The robot car can move more easily according to its wheels, which are driven by the motors.

Components:

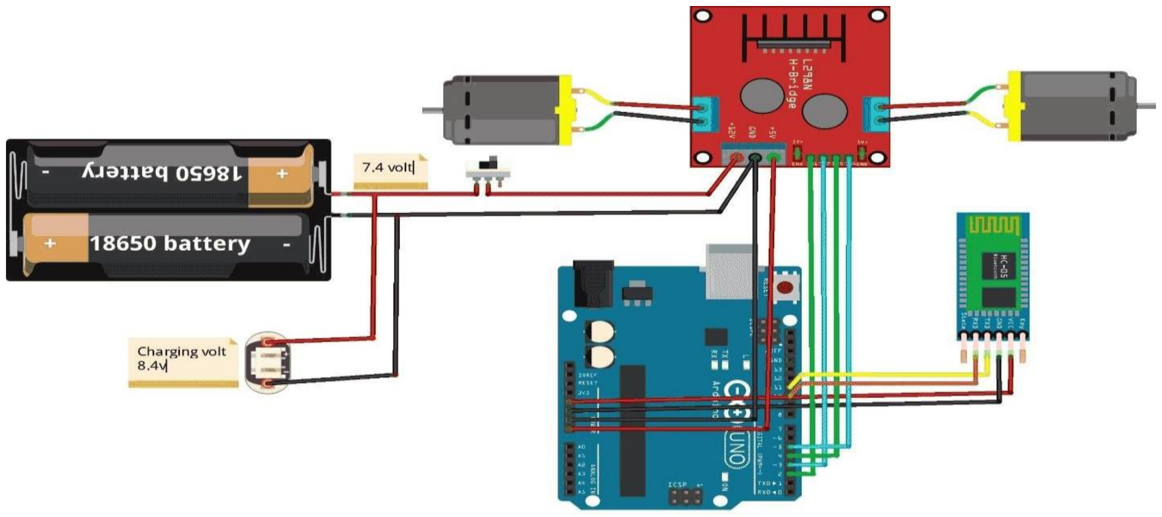
The components for the experiment “Bluetooth Control Car” includes:

1. Arduino UNO
2. Arduino Bluetooth Module HC-05
3. 6V DC Geared Motor 180 RPM Dual Shaft (4 pcs)
4. L298N H-Bridge Dual Motor Driver, Stepper Motor Driver
5. Yellow Plastic Mag Wheel Soft Tire (4 pcs)
6. 18650 Li-ion Rechargeable Battery (2 pcs)
7. Battery Holder 2s for 18650 Battery
8. Male and Female Jumper Wires (40 pcs)
9. 6 Inch X 6 Inch Acrylic Sheet
10. On / Off Switch (Mini)

Description of the components:

1. Arduino UNO: Primary control unit for handling commands and managing motors.
2. Arduino Bluetooth Module HC-05: Enables hands-free automobile connectivity.
3. 6V DC Geared Motor 180 RPM Dual Shaft (4 pieces): Provides movement to the wheels.
4. The L298N H-Bridge Dual Motor Driver, Stepper Motor Driver: Regulates the motors' direction and speed.
5. The automobile is supported and has traction due to the four yellow plastic mag wheel soft tires.
6. 18650 Li-ion Rechargeable Battery (2 pieces): The vehicle's power supply.
7. 18650 Battery Holder 2s: Safeguards and arranges the batteries within the vehicle.
8. Male and female jumper wires (40 pieces): Allows components to be electrically connected.
9. Six-by-six-inch acrylic sheet: Creates the foundation and chassis for mounting parts.
10. Mini On/Off Switch: This switch makes it simple to regulate the car's power supply.

Circuit Diagram:



Implementation:

Experimental Setup:

1. Assemble the chassis using the acrylic sheet, DC Geared Motors, and Yellow Plastic Mag Wheels.
2. Connect the DC Geared Motors to the L298N Motor Driver.
3. Wire the Arduino UNO to the L298N Motor Driver and the Bluetooth Module HC-05.
4. Connect the 18650 Li-ion Batteries to the Battery Holder and integrate them into the power supply circuit.
5. Include an On/Off Switch for convenient power control.

Procedure:

1. Bluetooth Module Configuration:
   a. Pair the Bluetooth Module HC-05 with a Bluetooth-enabled device.
2. Motor Driver and Arduino Wiring:
   a. Connect the DC Geared Motors to the L298N Motor Driver and link it to the Arduino UNO.
3. Power Supply Integration:
   a. Connect the 18650 Li-ion Batteries to the Battery Holder and the power supply circuit.
4. Bluetooth Communication Programming:
   a. Program the Arduino UNO to interpret signals from the Bluetooth Module and control the movement of the DC Geared Motors accordingly.
5. Chassis Assembly:
   a. Securely assemble the chassis components, ensuring stability and balance.

Operation:

1. Bluetooth Connection:
 - Pair the Bluetooth-enabled device with the Bluetooth Module HC-05 on the robot car.
2. Wireless Control:
 - Use the paired device to send signals to the Arduino UNO via Bluetooth, controlling the movement and direction of the robot car.
3. Obstacle Avoidance (Optional):
 - Implement obstacle avoidance mechanisms using additional sensors if desired.

Code:

```
char t;   void setup() { pinMode(13,OUTPUT);
//left motors   forward pinMode(12,OUTPUT);
//left  motors  reverse  pinMode(11,OUTPUT);
//right  motors forward pinMode(10,OUTPUT);
//right  motors  reverse  pinMode(9,OUTPUT);
//Led Serial.begin(9600);

}       void loop() {
if(Serial.available())
{       t = Serial.read();
Serial.println(t);
}       if(t == 'F'){                          //move
forward(all motors rotate in forward direction)
digitalWrite(13,HIGH); digitalWrite(11,HIGH);
}       else if(t == 'B'){                      //move reverse (all
motors rotate in reverse direction)
digitalWrite(12,HIGH); digitalWrite(10,HIGH);


}         else if(t == 'L'){                        //turn right (left
side motors rotate in forward direction,   right side
motors doesn't rotate)
          digitalWrite(11,HIGH);
}       else  if(t == 'R'){                       //turn left (right side
motors rotate in forward direction, left   side motors doesn't
rotate) digitalWrite(13,HIGH);
}  else if(t ==  'W'){                       //turn led on or
off)              digitalWrite(9,HIGH);
} else if(t == 'w'){ digitalWrite(9,LOW);
}       else if(t == 'S'){                       //STOP (all
motors stop)           digitalWrite(13,LOW);
digitalWrite(12,LOW); digitalWrite(11,LOW);
digitalWrite(10,LOW);
} delay(100);
```

Results:

Input: Arduino UNO receives signals from Bluetooth Module HC-05.

Processing: Arduino interprets signals to determine car movement commands.

Output: Arduino controls DC Geared Motors via L298N Motor Driver for car movement; LED control and power regulation implemented.

Conclusion:

The Bluetooth Control Car project effectively demonstrated how wireless communication, chassis design, and motor control can be integrated to produce a robotic system that can be operated remotely. Subsequent versions may investigate supplementary functionalities and improvements for expanded use.