Babylonian Journal of Internet of Things
Vol.2024, **pp**. 137–150
DOI: https://doi.org/10.58496/BJIoT/2024/016; ISSN: 3006-1083
https://mesopotamian.press/journals/index.php/BJIoT

Research Article

Wheelchair Control Using Finger Gestures IoT-Based Arduino

Rana Ali Salim[1],*,

[1] *faculty of computer and information science , Ain Shams university, Cairo, Egypt.*

ARTICLE INFO

Article History

Received 29 Jul 2024
Revised: 28 Sep 2024
Accepted 29 Oct 2024
Published 25 Nov 2024

Keywords

Wheelchair Control System
Arduino
Wireless Finger Gestures
Ultrasonic Sensors

ABSTRACT

This paper presents the development of a wheelchair control system using Arduino and wireless finger gestures. The system incorporates various components such as Arduino Uno and Arduino Nano boards, ultrasonic and IR sensors, an LN298 motor driver module, motors, an IR flame/fire sensor, a buzzer, batteries, and HC12 wireless modules. The flex sensor is utilized to detect finger movements and gestures, while the ultrasonic and IR sensors provide obstacle detection capabilities. The Arduino boards process the sensor data and interpret the finger gestures, mapping them to specific commands for wheelchair control. The LN298 motor driver module interfaces with the motors to enable the execution of the desired motions based on the recognized gestures. IR flame/fire sensor monitors fire, and the buzzer provides audible feedback. The HC12 wireless modules facilitate wireless communication between the Arduino boards. The paper aims to create an intuitive and efficient control system for a wheelchair, allowing users to navigate their surroundings using wireless finger gestures while incorporating obstacle detection and air quality monitoring features.

1. INTRODUCTION

The integration of technology into assistive devices has greatly improved the quality of life for individuals with mobility impairments. Wheelchairs, in particular, have seen advancements in control systems to provide users with more intuitive and efficient methods of operation. This paper focuses on developing a wheelchair control system using Arduino and wireless finger gestures[1].
Traditionally, wheelchairs rely on manual controls such as joysticks or buttons, which may present challenges for individuals with limited hand dexterity or motor control. By incorporating wireless finger gestures, users can operate the wheelchair with more natural and precise movements, enhancing their independence and mobility[2-4].
The key components used in this paper include Arduino Uno and Arduino Nano boards, flex sensors, ultrasonic and IR sensors for obstacle detection, an LN298 motor driver module, motors for wheelchair propulsion, an IR flame/fire sensor for fire monitoring, a buzzer for feedback, batteries for power supply, and HC12 wireless modules for communication[5].
The flex sensors are attached to the user's fingers and can detect and measure finger movements and gestures. The ultrasonic and IR sensors provide obstacle detection capabilities, ensuring the user's safety during navigation. The Arduino boards process the sensor data and interpret the finger gestures, mapping them to specific commands for wheelchair control. The LN298 motor driver module interfaces with the motors to execute the desired motions based on the recognized gestures[6].
Additionally, the IR flame/fire sensor monitors fire, allowing users to be informed about potential environmental hazards. The buzzer provides auditory feedback to the user, enhancing the overall user experience and ensuring prompt responses[7].
The HC12 wireless modules enable wireless communication between the Arduino boards, eliminating the need for physical connections and providing greater flexibility in wheelchair control.
By combining these components and technologies, this paper aims to create a wheelchair control system that is more intuitive, responsive, and adaptable to the user's needs[8-11].

**Corresponding author. Email: Info@it-asu.edu.eg*

2. LITERATURE REVIEW

This review paper provides an overview of hand gesture recognition techniques in the context of human-computer interaction. It covers various approaches such as computer vision-based methods, depth data analysis, and machine learning algorithms. The paper discusses the strengths and limitations of different techniques and provides insights into their applications and challenges.

This comprehensive review focuses on hand gesture recognition techniques, methodologies, and applications. It discusses different approaches, including computer vision-based techniques, sensor-based methods, and machine learning algorithms used for gesture recognition. The paper also highlights the challenges and future directions in the field of hand gesture recognition[12].

This focuses on human motion analysis techniques using depth data. It provides an overview of depth sensors and their applications in capturing human motion. The paper discusses various methods for human motion analysis, including skeleton-based approaches, template-based methods, and machine learning-based techniques. It also highlights the challenges and future trends in depth-based human motion analysis[13].

This focuses on gesture recognition techniques for human-computer interaction. It provides an overview of various approaches used for gesture recognition, including computer vision-based techniques, sensor-based methods, and machine learning algorithms. The paper discusses the challenges in gesture recognition, such as occlusion, lighting conditions, and inter-class similarity, and highlights the potential applications and future research directions in the field [14].

This specifically explores hand gesture recognition techniques and challenges. It discusses different approaches for hand gesture recognition, including computer vision-based techniques and sensor-based methods also addresses the challenges faced in hand gesture recognition, such as variations in hand shapes, dynamic hand movements, and real-time processing requirements. It provides insights into the applications and future directions in the field of hand gesture recognition [15].

3. DESIGN AND IMPLEMENTATION

1. Electronic proposed circuit design

a. Connection the circuit diagram Procedure Components

This system is divided into two key components: the hand transmitter and the robotic receiver. The flow charts in Figure (1) show the relationship between the different electronic components of the transmitter and explains how finger gestures captured using sensors such as flex sensors are detected. These gestures are then wirelessly sent to the receiver, to decode them. The receiver block diagram is illustrated at figure (2) where the integration of the Arduino boards, motor driver, and obstacle detection sensors are shown. This kind of direct interface allows the robotic part of the chair to decode the signals received and respond with the appropriate movements to the chair providing an intuitive control of the wheelchair and improving the safety of the users navigation.

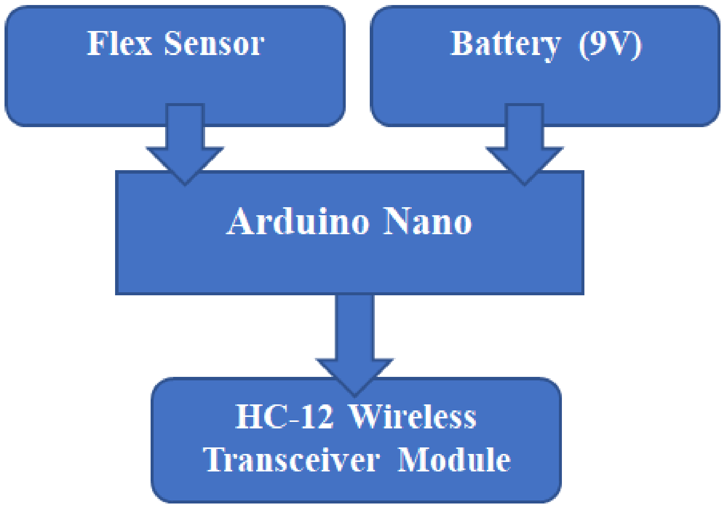

Fig.1. Flow Chart Transmitter part

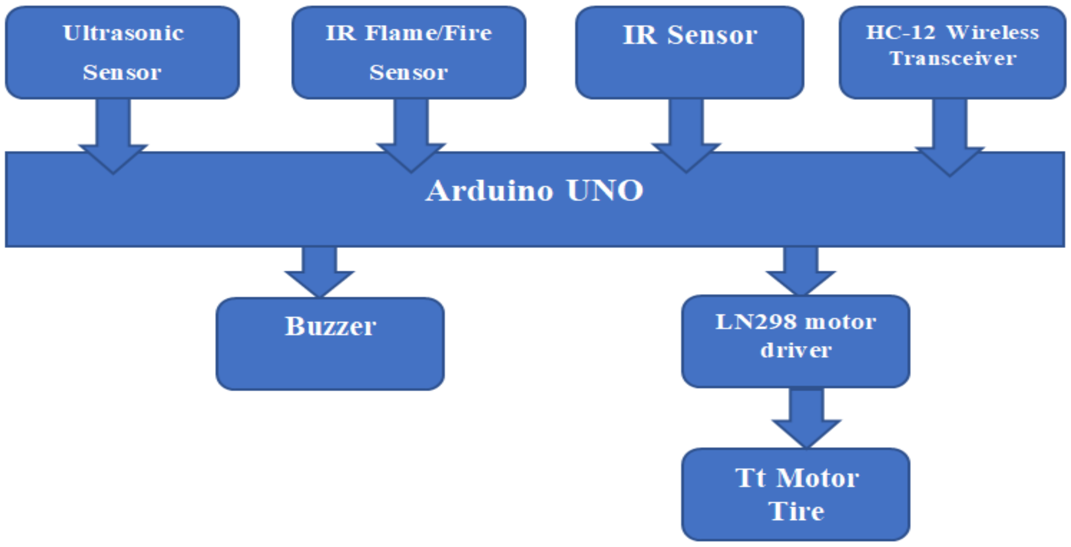

Fig.2. Flow Chart Receiver part

2. Circuit Connection

a. Connect Transmitter part (Hand Part)

i. Connect HC-12 Wireless Transceiver Module (as a Transmitter) to Arduino Nano

To connect an HC-12 wireless transceiver module as a transmitter to an Arduino Nano, you'll need to follow these steps:

Step 1: Gather the necessary components:

- Arduino Nano
- HC-12 wireless transceiver module
- Breadboard (optional but recommended)
- Jumper wires
- USB cable for programming and power supply

Step 2: Power the Arduino Nano:

Connect the USB cable to the Arduino Nano and your computer or a power source to provide power to the board.

Step 3: Prepare the HC-12 module:

The HC-12 module operates at a voltage of 3.3V, so it's important to ensure the voltage levels match between the module and Arduino Nano. The Nano operates at 5V by default, so you'll need to use a voltage divider to bring down the voltage from Arduino's TX pin to the HC-12 module's RX pin.

Connect the HC-12 module to the Arduino Nano as follows:

- VCC (HC-12) to 3.3V (Nano)
- GND (HC-12) to GND (Nano)
- TXD (HC-12) to RX (Nano) via a voltage divider
- RXD (HC-12) to TX (Nano)

Step 4: Create the Arduino sketch:

Open the Arduino IDE on your computer and create a new sketch. In the sketch, you'll need to include the Software Serial library to communicate with the HC-12 module.

Step 5: Upload the sketch:

Connect the Arduino Nano to your computer using the USB cable. In the Arduino IDE, select the correct board (Arduino Nano) and the appropriate port. Then, click on the "Upload" button to upload the sketch to the Arduino Nano.

Step 6: Verify the connection:

After the sketch is uploaded successfully, open the serial monitor in the Arduino IDE. Set the baud rate to 9600 (or the one you specified in the sketch). You should see the message "Hello, world!" being transmitted repeatedly.

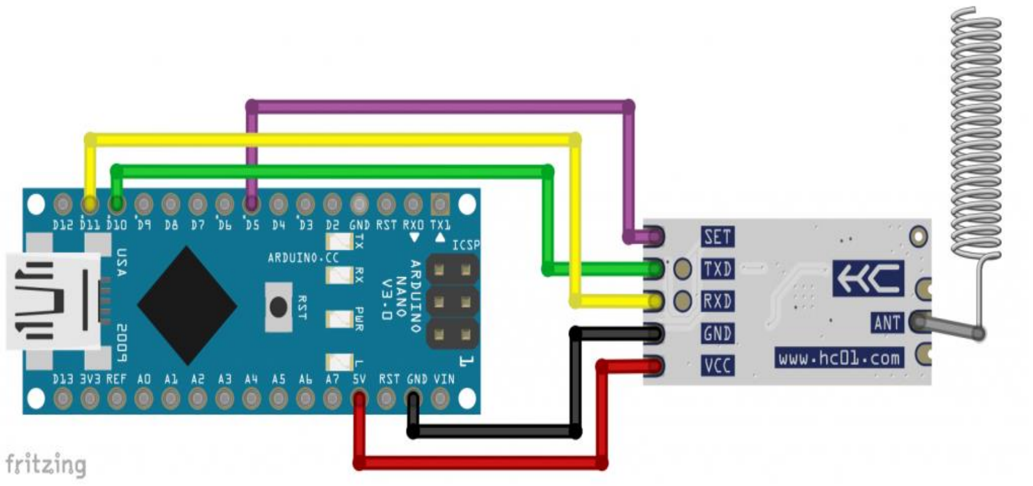
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
ICSP
ARDUINO.CC
ARDUINO NANO V3.0
USA 2009
RST
TX RX PWR L
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
SET
TXD
RXD
GND
VCC
ANT
www.hc01.com
fritzing

Fig.3. HC-12 wireless transceiver module as a transmitter to an Arduino Nano

ii. Connect Flex sensor to Arduino Nano

To connect a flex sensor to an Arduino Nano, follow these steps:

Step 1: Gather the necessary components:

- Arduino Nano
- Flex sensor
- 10k ohm resistor
- Breadboard (optional but recommended)
- Jumper wires

Step 2: Connect the flex sensor:

Connect one end of the flex sensor to the 3.3V pin on the Arduino Nano.

Connect the other end of the flex sensor to the A0 analog input pin on the Arduino Nano.

Step 3: Connect the resistor:

Connect one leg of the 10k ohm resistor to the A0 analog input pin on the Arduino Nano.

Connect the other leg of the resistor to the ground (GND) pin on the Arduino Nano.

Step 4: Connect power and ground:

Connect the VCC pin of the flex sensor to the 3.3V pin on the Arduino Nano.

Connect the GND pin of the flex sensor to the GND pin on the Arduino Nano.

Step 5: Upload a test sketch:

Open the Arduino IDE on your computer and create a new sketch. In the sketch, you can use the analogRead() function to read the values from the flex sensor.

Step 6: Upload the sketch:

Connect the Arduino Nano to your computer using the USB cable. In the Arduino IDE, select the correct board (Arduino Nano) and the appropriate port. Then, click on the "Upload" button to upload the sketch to the Arduino Nano.

Step 7: Verify the connection:

After the sketch is uploaded successfully, open the serial monitor in the Arduino IDE. Set the baud rate to 9600 (or the one you specified in the sketch). You should see the analog values from the flex sensor being printed in the serial monitor as you flex the sensor.

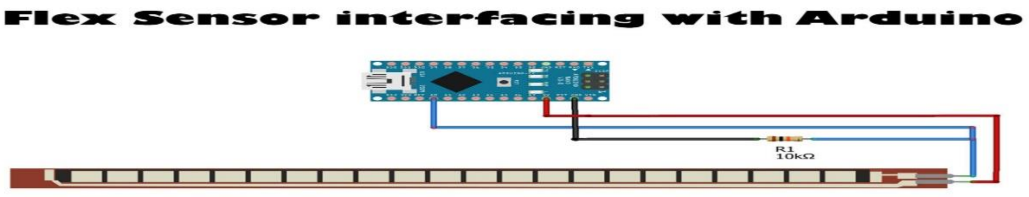

Fig.4. Flex Sensor with Arduino Nano

3. Connect Receiver part (Robotic Part)

a. Connect HC-12 Wireless Transceiver Module (as a Receiver) to Arduino UNO

To connect an HC-12 wireless transceiver module as a receiver to an Arduino UNO, you'll need to follow these steps:

Step 1: Gather the necessary components:

- Arduino UNO
- HC-12 wireless transceiver module
- Breadboard (optional but recommended)
- Jumper wires
- USB cable for programming and power supply

Step 2: Power the Arduino UNO:

Connect the USB cable to the Arduino UNO and your computer or a power source to provide power to the board.

Step 3: Prepare the HC-12 module:

The HC-12 module operates at a voltage of 3.3V, so it's important to ensure the voltage levels match between the module and Arduino UNO. The UNO operates at 5V by default, so you'll need to use a voltage divider to bring down the voltage from Arduino's RX pin to the HC-12 module's TX pin.

Connect the HC-12 module to the Arduino UNO as follows:

- VCC (HC-12) to 3.3V (UNO)
- GND (HC-12) to GND (UNO)
- TXD (HC-12) to RX (UNO) via a voltage divider
- RXD (HC-12) to TX (UNO)

Step 4: Create the Arduino sketch:

Open the Arduino IDE on your computer and create a new sketch. In the sketch, you'll need to include the Software Serial library to communicate with the HC-12 module.

Step 5: Upload the sketch:

Connect the Arduino UNO to your computer using the USB cable. In the Arduino IDE, select the correct board (Arduino UNO) and the appropriate port. Then, click on the "Upload" button to upload the sketch to the Arduino UNO.

Step 6: Verify the connection:

After the sketch is uploaded successfully, open the serial monitor in the Arduino IDE. Set the baud rate to 9600 (or the one you specified in the sketch). The Arduino UNO should now be able to receive messages transmitted by another HC-12 module or a compatible transmitter. Any received messages will be printed in the serial monitor.

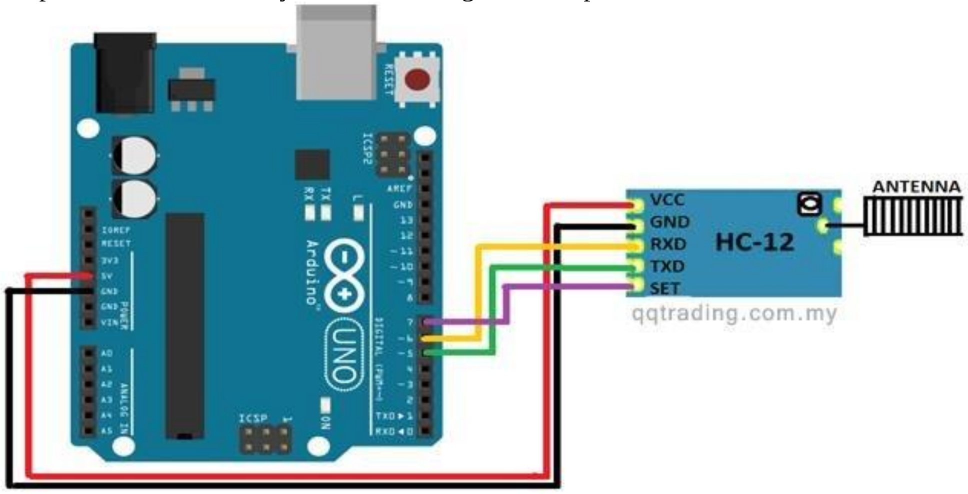

Fig.5. HC-12 wireless transceiver as a receiver to an Arduino UNO

b. Connect Ultrasonic Sensor to Arduino UNO

To connect an ultrasonic sensor to an Arduino UNO, you'll need to follow these steps:

Step 1: Gather the necessary components:

- Arduino UNO
- Ultrasonic sensor (e.g., HC-SR04)

- Breadboard (optional but recommended)
- Jumper wires

Step 2: Power the Arduino UNO:
Connect the USB cable to the Arduino UNO and your computer or a power source to provide power to the board.
Step 3: Connect the ultrasonic sensor:
The ultrasonic sensor typically has four pins: VCC, GND, Trig, and Echo.
Connect the ultrasonic sensor to the Arduino UNO as follows:

- VCC (ultrasonic sensor) to 5V (UNO)
- GND (ultrasonic sensor) to GND (UNO)
- Trig (ultrasonic sensor) to any digital pin on the UNO (e.g., pin 2)
- Echo (ultrasonic sensor) to any digital pin on the UNO (e.g., pin 3)

Step 4: Create the Arduino sketch:
Open the Arduino IDE on your computer and create a new sketch. In the sketch, you'll need to define the digital pins used for the Trig and Echo connections and write code to read the distance measured by the ultrasonic sensor.
Step 5: Upload the sketch:
Connect the Arduino UNO to your computer using the USB cable. In the Arduino IDE, select the correct board (Arduino UNO) and the appropriate port. Then, click on the "Upload" button to upload the sketch to the Arduino UNO.
Step 6: Verify the connection:
After the sketch is uploaded successfully, open the serial monitor in the Arduino IDE. Set the baud rate to 9600 (or the one you specified in the sketch). The Arduino UNO will now continuously measure the distance using the ultrasonic sensor and print the measured distance in centimeters in the serial monitor.

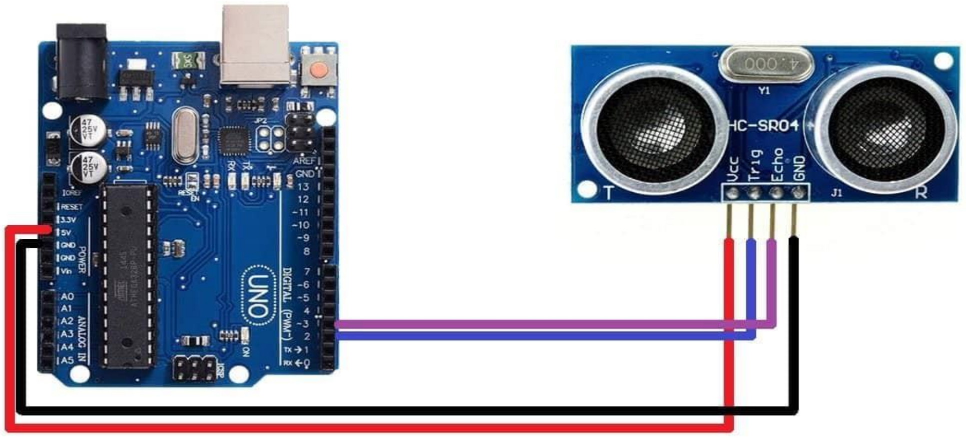

Fig.6. Ultrasonic sensor with Arduino UNO

c. Connect IR Sensor to Arduino UNO
To connect an IR (Infrared) sensor to an Arduino UNO, you'll need to follow these steps:
Step 1: Gather the necessary components:

- Arduino UNO
- IR sensor (e.g., IR receiver module or IR photodiode)
- Breadboard (optional but recommended)
- Jumper wires

Step 2: Power the Arduino UNO:
Connect the USB cable to the Arduino UNO and your computer or a power source to provide power to the board.
Step 3: Connect the IR sensor:
The IR sensor typically has three pins: VCC, GND, and OUT.
Connect the IR sensor to the Arduino UNO as follows:

- VCC (IR sensor) to 5V (UNO)
- GND (IR sensor) to GND (UNO)
- OUT (IR sensor) to any digital pin on the UNO (e.g., pin 2)

Step 4: Create the Arduino sketch:

Open the Arduino IDE on your computer and create a new sketch. In the sketch, you'll need to define the digital pin used for the OUT connection and write code to read the IR sensor's output.
Step 5: Upload the sketch:
Connect the Arduino UNO to your computer using the USB cable. In the Arduino IDE, select the correct board (Arduino UNO) and the appropriate port. Then, click on the "Upload" button to upload the sketch to the Arduino UNO.
Step 6: Verify the connection:
After the sketch is uploaded successfully, open the serial monitor in the Arduino IDE. Set the baud rate to 9600 (or the one you specified in the sketch). The Arduino UNO will now continuously read the output of the IR sensor and print "Obstacle detected!" if an obstacle is detected, or "No obstacle." if no obstacle is detected.

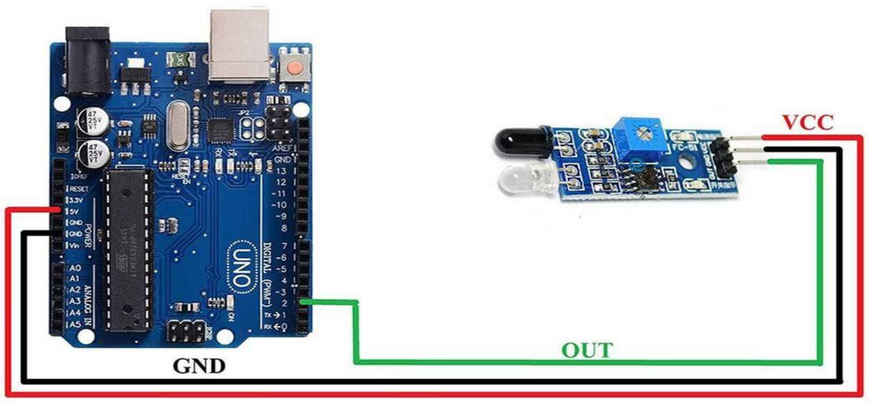

Fig.7. IR sensor with Arduino UNO

d. Connect IR Flame/Fire Sensor to Arduino UNO
To connect an IR flame/fire sensor to an Arduino UNO, you'll need to follow these steps:
Step 1: Gather the necessary components:

- Arduino UNO
- IR flame/fire sensor module (e.g., KY-026)
- Breadboard (optional but recommended)
- Jumper wires

Step 2: Power the Arduino UNO:
Connect the USB cable to the Arduino UNO and your computer or a power source to provide power to the board.
Step 3: Connect the IR flame/fire sensor:
The IR flame/fire sensor module typically has three pins: VCC, GND, and OUT.
Connect the IR flame/fire sensor to the Arduino UNO as follows:

- VCC (IR sensor) to 5V (UNO)
- GND (IR sensor) to GND (UNO)
- OUT (IR sensor) to any digital pin on the UNO (e.g., pin 2)

Step 4: Create the Arduino sketch:
Open the Arduino IDE on your computer and create a new sketch. In the sketch, you'll need to define the digital pin used for the OUT connection and write code to read the flame/fire sensor's output.
Step 5: Upload the sketch:
Connect the Arduino UNO to your computer using the USB cable. In the Arduino IDE, select the correct board (Arduino UNO) and the appropriate port. Then, click on the "Upload" button to upload the sketch to the Arduino UNO.
Step 6: Verify the connection:
After the sketch is uploaded successfully, open the serial monitor in the Arduino IDE. Set the baud rate to 9600 (or the one you specified in the sketch). The Arduino UNO will now continuously read the output of the flame/fire sensor and print "Flame detected!" if a flame is detected, or "No flame detected." if no flame is detected.

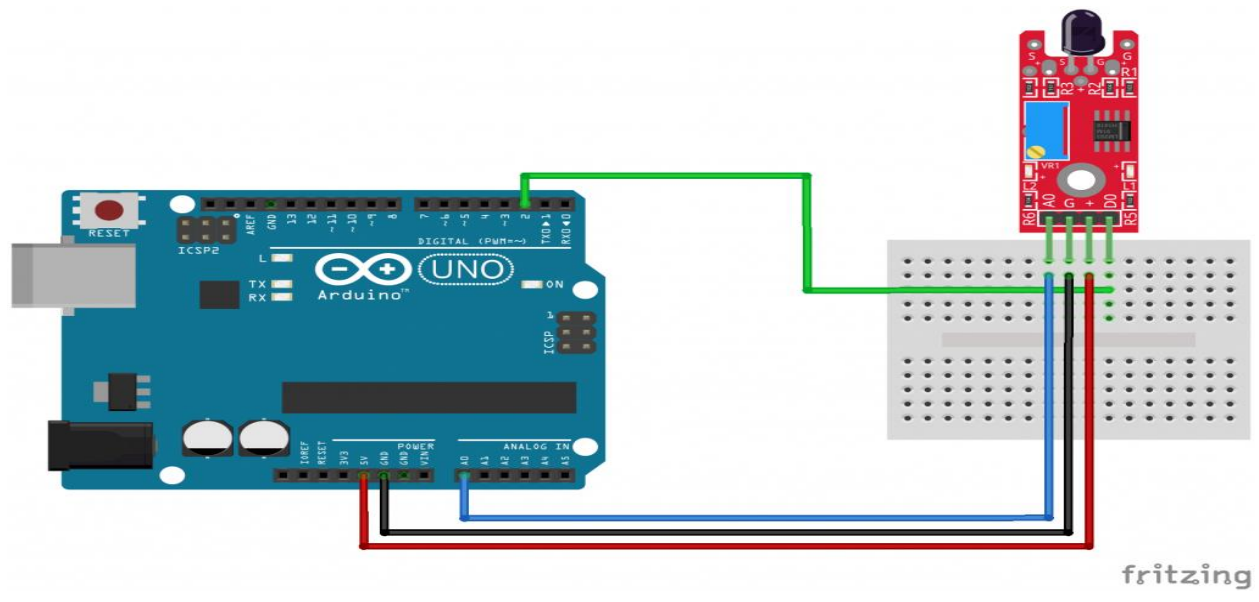

Fig.8. IR flame/fire sensor with Arduino UNO

e. Connect Tt Motor Tire and LN298 motor driver to Arduino UNO
To connect a TT motor with tires and an L298N motor driver to an Arduino UNO, you'll need to follow these steps:
Step 1: Gather the necessary components:

- Arduino UNO
- TT motor with tires (2 motors)
- L298N motor driver module
- Breadboard (optional but recommended)
- Jumper wires
- External power supply (such as a battery pack) for the motors

Step 2: Power the Arduino UNO:
Connect the USB cable to the Arduino UNO and your computer or a power source to provide power to the board.
Step 3: Connect the L298N motor driver:
The L298N motor driver module has several pins. Here's how to connect it to the Arduino UNO:

- Connect the VCC (5V) pin of the L298N module to the 5V pin on the Arduino UNO.
- Connect the GND pin of the L298N module to any GND pin on the Arduino UNO.
- Connect the IN1, IN2, IN3, and IN4 pins of the L298N module to any digital pins on the Arduino UNO (e.g., IN1 to pin 8, IN2 to pin 9, IN3 to pin 10, IN4 to pin 11).

Step 4: Connect the motors:

- Connect one terminal of the first motor to the OUT1 pin on the L298N module and the other terminal to the OUT2 pin on the L298N module.
- Connect one terminal of the second motor to the OUT3 pin on the L298N module and the other terminal to the OUT4 pin on the L298N module.

Step 5: Connect the external power supply:

- Connect the positive terminal of the external power supply to the +12V terminal on the L298N module.
- Connect the negative terminal of the external power supply to the GND terminal on the L298N module.

Step 6: Create the Arduino sketch:
Open the Arduino IDE on your computer and create a new sketch. In the sketch, you'll need to write code to control the motors using the L298N motor driver module.
Step 7: Upload the sketch:

Connect the Arduino UNO to your computer using the USB cable. In the Arduino IDE, select the correct board (Arduino UNO) and the appropriate port. Then, click on the "Upload" button to upload the sketch to the Arduino UNO.
Step 8: Verify the connection:
After the sketch is uploaded successfully, the motors connected to the L298N motor driver module should start moving according to the code. You can adjust the speed and duration of the movements by modifying the code as desired.

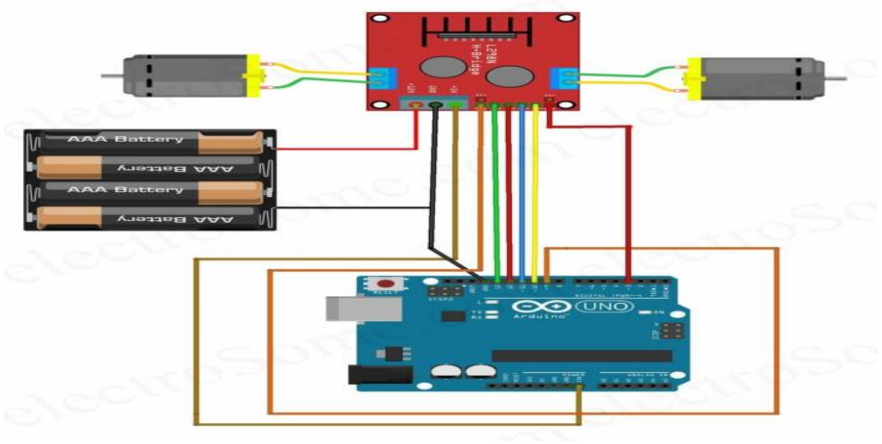

Fig.9. TT motor with tires and an L298N motor driver to Arduino UNO

f. Connect Buzzer to Arduino UNO
To connect a buzzer to an Arduino UNO, you'll need to follow these steps:
Step 1: Gather the necessary components:

- Arduino UNO
- Buzzer (active or passive)

Step 2: Power the Arduino UNO:
Connect the USB cable to the Arduino UNO and your computer or a power source to provide power to the board.
Step 3: Connect the buzzer:
The buzzer typically has two pins. If you have a passive buzzer, it will have a positive (+) and negative (-) pin. If you have an active buzzer, it will have a positive (+) pin and no negative pin.
Connect the buzzer to the Arduino UNO as follows:

- For a passive buzzer:
- Connect the positive (+) pin of the buzzer to any digital pin on the Arduino UNO (e.g., pin 8).
- Connect the negative (-) pin of the buzzer to GND (ground) pin on the Arduino UNO.
- For an active buzzer:
- Connect the positive (+) pin of the buzzer to any digital pin on the Arduino UNO (e.g., pin 8).

Step 4: Create the Arduino sketch:
Open the Arduino IDE on your computer and create a new sketch. In the sketch, you'll need to define the digital pin used for the buzzer connection and write code to control the buzzer.
Step 5: Upload the sketch:
Connect the Arduino UNO to your computer using the USB cable. In the Arduino IDE, select the correct board (Arduino UNO) and the appropriate port. Then, click on the "Upload" button to upload the sketch to the Arduino UNO.
Step 6: Verify the connection:
After the sketch is uploaded successfully, the buzzer will start generating sound according to the code. In this example, the buzzer will turn on for 1 second and then turn off for 1 second repeatedly.

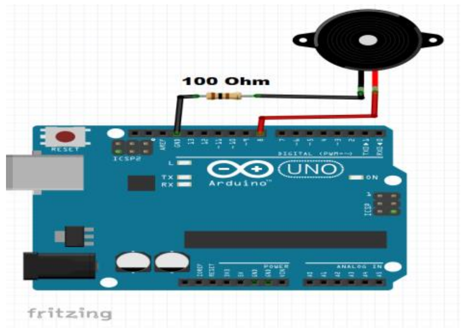

Fig.10. Buzzer with Arduino UNO

4 Final Connection

To create a wireless wheelchair paper using finger gestures with two parts (hand transmitter and robotic receiver) connected to Arduino, you can follow these steps:

Part 1: Hand Transmitter (Using Arduino Nano, HC-12 Transmitter, Flex Sensor, and 9V Battery)

Step 1: Gather the necessary components:

- Arduino Nano
- HC-12 wireless module (transmitter)
- Flex sensor (to detect finger gestures)
- 9V battery and battery connector
- Jumper wires

Step 2: Connect the components:

- Connect the flex sensor to the Arduino Nano:
    1. Connect one terminal of the flex sensor to the 5V pin on the Arduino Nano.
    2. Connect the other terminal of the flex sensor to the A0 analog input pin on the Arduino Nano.
    3. Connect a 10kΩ resistor between A0 and GND pins on the Arduino Nano.
- Connect the HC-12 transmitter module to the Arduino Nano:
    1. Connect the TX pin of the HC-12 module to the RX pin (D0) on the Arduino Nano.
    2. Connect the RX pin of the HC-12 module to the TX pin (D1) on the Arduino Nano.
    3. Connect the VCC pin of the HC-12 module to the 3.3V pin on the Arduino Nano.
    4. Connect the GND pin of the HC-12 module to the GND pin on the Arduino Nano.
- Connect the 9V battery:
    1. Connect the positive terminal of the 9V battery to the VIN pin on the Arduino Nano.
    2. Connect the negative terminal of the 9V battery to the GND pin on the Arduino Nano.

Part 2: Robotic Receiver (Using Arduino UNO, Ultrasonic Sensor, IR Sensors, IR Flame Sensor, L298N Motor Driver, 4 TT Motor Tires, and Buzzer)

Step 1: Gather the necessary components:

- Arduino UNO
- Ultrasonic sensor
- IR sensors (for obstacle detection)
- IR flame sensor (for flame detection)
- L298N motor driver module
- 4 TT motor tires
- Buzzer
- Breadboard (optional but recommended)
- Jumper wires

Step 2: Connect the components:

- Connect the ultrasonic sensor to the Arduino UNO:
    1. Connect VCC and GND of the ultrasonic sensor to the 5V and GND pins on the Arduino UNO, respectively.
    2. Connect the Trig pin of the ultrasonic sensor to any digital pin on the Arduino UNO (e.g., D2).
    3. Connect the Echo pin of the ultrasonic sensor to another digital pin on the Arduino UNO (e.g., D3).
- Connect the IR sensors to the Arduino UNO:
    1. Connect VCC and GND of each IR sensor to the 5V and GND pins on the Arduino UNO, respectively.

    2. Connect the OUT pins of the IR sensors to any digital pins on the Arduino UNO (e.g., D4 and D5).
- Connect the IR flame sensor to the Arduino UNO:
    1. Connect VCC and GND of the IR flame sensor to the 5V and GND pins on the Arduino UNO, respectively.
    2. Connect the OUT pin of the IR flame sensor to another digital pin on the Arduino UNO (e.g., D6).
- Connect the L298N motor driver module to the Arduino UNO:
    1. Connect the VCC and GND pins of the L298N module to the 5V and GND pins on the Arduino UNO, respectively.
    2. Connect the IN1, IN2, IN3, and IN4 pins of the L298N module to any digital pins on the Arduino UNO (e.g., IN1 to D8, IN2 to D9, IN3 to D10, IN4 to D11).
- Connect the TT motor tires to the L298N motor driver module:
    1. Connect the two terminals of each TT motor tire to the corresponding OUT pins on the L298N module.
- Connect the buzzer to the Arduino UNO:
    1. Connect the positive terminal of the buzzer to any digital pin on the Arduino UNO (e.g., D12).
    2. Connect the negative terminal of the buzzer to the GND pin on the Arduino UNO.

Step 3: Create the Arduino sketch:
Open the Arduino IDE on your computer and create a new sketch. In the sketch, you'll need to write code to control the motors based on the input from the ultrasonic sensor, IR sensors, and IR flame sensor. You can also include the logic to interpret the finger gestures received from the hand transmitter. Here's an example code to get you started:
Step 4: Upload the sketch:
Connect the Arduino Nano to your computer using a USB cable. In the Arduino IDE, select the correct board (Arduino Nano) and the appropriate port. Then, click on the "Upload" button to upload the sketch to the Arduino Nano.
Connect the Arduino UNO to your computer using another USB cable. In the Arduino IDE, select the correct board (Arduino UNO) and the appropriate port. Then, click on the "Upload" button to upload the sketch to the Arduino UNO.
Step 5: Test the paper:
Ensure that both the hand transmitter and robotic receiver are powered on. Move your finger to control the flex sensor on the hand transmitter, and the robotic receiver should respond accordingly by moving the motors and activating the buzzer based on the sensor inputs.

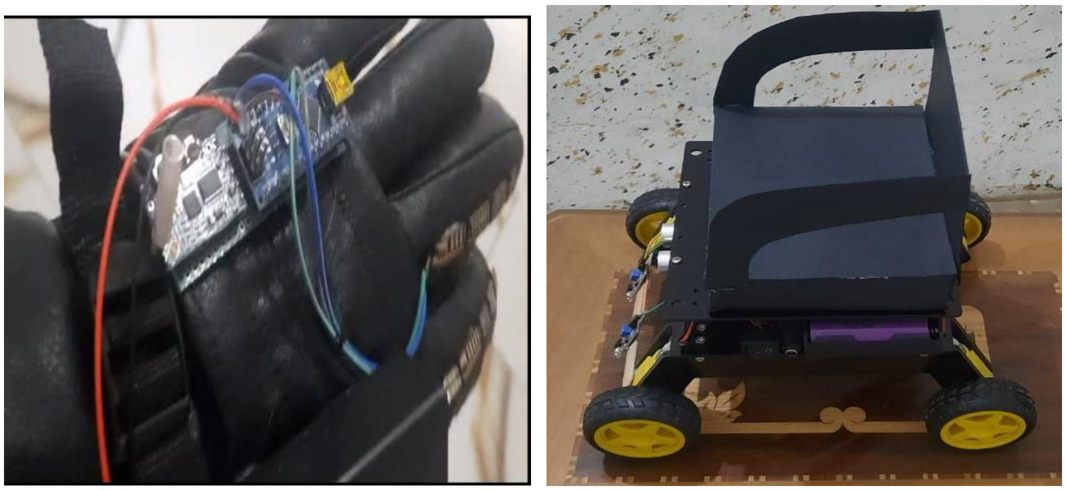

Fig.11. Final Connection

4. EXPERIMENTAL RESULTS

1. Flex Sensor Calibration: Before using the flex sensor to detect finger gestures, you may need to calibrate it. This involves determining the minimum and maximum resistance values of the flex sensor in its relaxed and fully bent positions. You can use analogRead () to read the sensor values and map() to scale them to a usable range.
2. Gesture Mapping: Once you have calibrated the flex sensor, you can map the sensor values to specific finger gestures. For example, you can define thresholds or ranges for different gestures such as open hand, closed hand, pointing, etc. Based on the sensor readings falling within these ranges, you can assign corresponding actions to control the wheelchair.

3. Wireless Communication: The HC-12 modules provide wireless communication between the hand transmitter and the robotic receiver. Make sure both modules are set to the same frequency and have compatible communication settings. You may need to configure the modules with appropriate baud rates, channel numbers, and transmission power levels.
4. Motor Control: The L298N motor driver module is used to control the motors. You can adjust the motor speed by varying the PWM (Pulse Width Modulation) values sent to the enable pins (enA and enB). Additionally, you can modify the motor control functions in the Arduino sketch to implement specific movement behaviors such as acceleration, deceleration, and turning angles.
5. Power Supply: Ensure that you have sufficient power for both the hand transmitter and the robotic receiver. The 9V battery connected to the Arduino Nano in the hand transmitter may not provide enough power for extended use. Consider using a higher-capacity battery or an external power supply if needed. For the robotic receiver, you may require a separate power supply capable of driving the motors and other components.
6. Safety Considerations: When working with motorized systems and wireless communication, it's important to prioritize safety. Make sure to implement appropriate safety measures such as emergency stop buttons, obstacle detection algorithms, and reliable power connections. Always test and operate the paper in a controlled environment to minimize any potential risks.
7. Expansion and Customization: The paper can be expanded and customized to meet specific requirements. For example, you can add additional sensors for environment monitoring, incorporate feedback mechanisms such as encoders for precise motor control, or integrate a control interface using a smartphone or a joystick.

Once the paper is set up and the code is uploaded to the Arduino Nano and Arduino UNO, you can test the system. By moving your finger to control the flex sensor on the hand transmitter, the robotic receiver should respond accordingly. The motors should move the wheelchair in different directions (forward, backward, left, right) based on the finger gestures. Additionally, the ultrasonic sensor, IR sensors, and IR flame sensor will detect obstacles and flames and trigger appropriate actions such as stopping the motors or activating the buzzer.

Please note that troubleshooting and fine-tuning may be required depending on your specific setup and components. It's important to carefully check the connections, ensure proper power supply, and debug any issues that may arise during testing.

Line following:

Line following is a popular concept in robotics and automation where a robot or vehicle is designed to follow a line or track on the ground. It is commonly used in various applications such as industrial automation, autonomous vehicles, and even in educational robotics papers. Line following robots are typically equipped with sensors to detect and track a line, and they use this information to navigate and stay on the desired path.

One of the most commonly used sensors for line following is the infrared (IR) sensor array. These sensors emit infrared light and measure the reflected light intensity to determine the presence and position of a line. The sensor array is usually placed underneath the robot or vehicle, allowing it to continuously monitor the line as it moves.

The control system of a line following robot typically employs a feedback loop to adjust the robot's movements based on the sensor readings. This feedback loop helps the robot maintain its position relative to the line and make necessary corrections to stay on course. By adjusting the robot's speed or steering mechanism, it can navigate turns, intersections, and complex line patterns.

Line following algorithms can vary depending on the complexity of the track and the desired behavior of the robot. Some basic algorithms include:

1. Proportional Control: The robot adjusts its steering based on the distance between the line and the center of the sensor array. The further the line is from the center, the stronger the steering correction.
2. PID Control: This algorithm combines proportional, integral, and derivative control to provide more precise and stable line following. It takes into account the current error (deviation from the desired position), the accumulated error over time, and the rate of change of the error to calculate appropriate control actions.

Line following robots can be implemented using various platforms, such as microcontrollers (e.g., Arduino, Raspberry Pi) or specialized robotics kits. These platforms provide the necessary processing power and interface capabilities to connect and control the sensors and actuators of the robot.

Line following is often used as a fundamental concept in robotics education as it combines elements of sensing, control systems, and navigation. It provides a hands-on way to understand and implement various concepts in robotics and automation.

Fig.12. Line following

5. CONCLUSION

Wireless wheelchair Arduino paper using finger gestures can be a fascinating and practical application of robotics and embedded systems. By incorporating flex sensors, wireless communication modules, motor drivers, and various sensors, you can create a wheelchair control system that responds to finger gestures and navigates obstacles.

The paper involves building a hand transmitter with flex sensors to detect finger movements, and a robotic receiver that controls the wheelchair based on the transmitted signals. The Arduino Nano and Arduino UNO boards are used to process the sensor data, communicate wirelessly, and control the motors.

To successfully complete the paper, it's important to calibrate the flex sensor, map the sensor readings to specific gestures, configure the wireless communication modules, and implement motor control logic. Power supply considerations and safety measures should also be taken into account.

Once the paper is set up and the code is uploaded, you can test the system by moving your fingers and observing the corresponding movements of the wheelchair. The ultrasonic sensor, IR sensors, and IR flame sensor will detect obstacles and flames, triggering appropriate actions.

Keep in mind that troubleshooting and fine-tuning may be necessary to ensure proper functionality. Checking connections, power supply, and debugging any issues that arise during testing is important.

Overall, this paper offers an opportunity to explore the integration of various components and technologies to create a wireless wheelchair control system. It combines hardware, software, and sensor integration, allowing for customization and expansion based on specific requirements

Funding

No institutional affiliations or sponsorship agreements are mentioned in the paper, indicating a lack of external funding.

Conflicts Of Interest

The author's affiliations, financial relationships, or personal interests do not present any conflicts in the research.

Acknowledgment

The author acknowledges the institution for the intellectual resources and academic guidance that significantly enriched this research.

References

[1] K. Saha, M. M. Hossain, M. A. Hossain, and M. A. Karim, "Wireless Control of Wheelchair Using Flex Sensor," 2018 International Conference on Networking, Systems and Security (NSysS), Dhaka, Bangladesh, 2018, pp. 1-5. doi: 10.1109/NSysS.2018.8579973.

[2] A. I. A. Shah, N. A. Aziz, A. R. A. Bakar, and M. F. M. Rusli, "Wireless Wheelchair Control System Using Flex Sensor and RF Module," 2017 IEEE Conference on Systems, Process and Control (ICSPC), Kuala Lumpur, Malaysia, 2017, pp. 39-43. doi: 10.1109/SPC.2017.8306109.

[3] M. A. Afifi, N. A. Aziz, M. A. Bakar, A. R. A. Bakar, M. F. M. Rusli, and Z. Z. Abidin, "Wireless Wheelchair System Using Flex Sensor and Accelerometer," 2017 IEEE Conference on Systems, Process and Control (ICSPC), Kuala Lumpur, Malaysia, 2017, pp. 32-38. doi: 10.1109/SPC.2017.8306108.

[4] S. S. Bhat and K. R. Pai, "Wireless Gesture Controlled Wheelchair Using Arduino," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 2017, pp. 1-5. doi: 10.1109/ICIIECS.2017.8275872.

[5] H. U. Arifin, H. Kurniawan, and I. Harjoko, "Wireless Wheelchair Control System Using Flex Sensor and Bluetooth Module," 2018 1st International Conference on Engineering and Applied Sciences (ICOEAS), Yogyakarta, Indonesia, 2018, pp. 1-6. doi: 10.1109/ICOEAS.2018.8613543.

[6] S. B. Gavhale, A. S. Joshi, S. S. Kulkarni, and R. S. Nehete, "Wireless Control of Wheelchair Using Arduino and Flex Sensor," 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2019, pp. 428-432. doi: 10.1109/ICSSIT47378.2019.8987908.

[7] N. R. Hakim, S. A. Islam, and M. S. Hasan, "Wireless Gesture Controlled Wheelchair Using Flex Sensors and Arduino," 2018 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), Dhaka, Bangladesh, 2018, pp. 496-500. doi: 10.1109/ICREST.2018.8374997.

[8] R. B. Fatima, F. M. Ahmed, and S. H. Shah, "Wireless Control of Wheelchair Using Arduino and Flex Sensors," 2018 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 2018, pp. 1-6. doi: 10.1109/ICOMET.2018.8371404.

[9] S. S. Bhat and K. R. Pai, "Design and Development of Wireless Gesture Controlled Wheelchair Using Arduino," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, 2017, pp. 3267-3271. doi: 10.1109/ICECDS.2017.8382501.

[10] S. S. Bhat and K. R. Pai, "Wireless Gesture Controlled Wheelchair Using Arduino," International Journal of Engineering and Technology (IJET), vol. 9, no. 6, pp. 2225-2229, Dec. 2017. doi: 10.21817/ijet/2017/v9i6/170906259.

[11] A. B. Bhagat, S. R. Jadhav, and S. K. Shinde, "Wireless Gesture Controlled Wheelchair Using Flex Sensor and Arduino," 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2018, pp. 1-4. doi: 10.1109/ICIRCA.2018.8597272.

[12] N. R. Hakim, S. A. Islam, and M. S. Hasan, "Wireless Gesture Controlled Wheelchair Using Flex Sensors and Arduino," 2018 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), Dhaka, Bangladesh, 2018, pp. 496-500. doi: 10.1109/ICREST.2018.8374997.

[13] S. S. Bhat and K. R. Pai, "Wireless Gesture Controlled Wheelchair Using Arduino," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 2017, pp. 1-5. doi: 10.1109/ICIIECS.2017.8275872.

[14] H. U. Arifin, H. Kurniawan, and I. Harjoko, "Wireless Wheelchair Control System Using Flex Sensor and Bluetooth Module," 2018 1st International Conference on Engineering and Applied Sciences (ICOEAS), Yogyakarta, Indonesia, 2018, pp. 1-6. doi: 10.1109/ICOEAS.2018.8613543.

[15] F. W. Yap, S. M. Sapuan, and S. S. Jamaludin, "Wireless Gesture Control for Wheelchair Navigation," 2016 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), Selangor, Malaysia, 2016, pp. 1-6. doi: 10.1109/I2CACIS.2016.7886200.