ASSIGNMENT FINAL REPORT

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	**Unit 45: Internet of Things**		
Submission date	2/8/2024	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Hoang Van Quyet	**Student ID**	Bh00711
Class	IT0603	**Assessor name**	Nguyen Manh Tuan

Student's signature	Quyet

Grading grid

P1	P2	P3	P4	P5	P6	P7	P8	M1	M2	M3	M4	M5	M6	D1	D2	D3

❐ Summative Feedback:	❐ Resubmission Feedback:

Grade:	Assessor Signature:	Date:

Internal Verifier's Comments:
Signature & Date:

Table of content

Table of figure

I. Introduction

The Internet of Things (IoT) represents a transformative technological advancement that integrates physical devices with digital systems to enable intelligent interaction and automation. This report explores the IoT ecosystem, focusing on its broad applications and essential components, and then delves into a specific application: smart remote-controlled cars.

In Activity 1: IoT Overview, we define IoT and examine its impact across various fields, including smart homes, smart cities, healthcare, transportation, and agriculture. By understanding the role of IoT in these areas, we gain insights into how interconnected devices enhance functionality and efficiency in diverse sectors.

Activity 2: Smart Remote-Controlled Cars focuses on a practical implementation of IoT concepts. We explore the use cases and components of smart remote-controlled cars, detailing how each part contributes to the overall functionality of the system. The section on electronic control circuits and algorithm flowcharts provides a technical perspective on the integration of hardware and software in creating a functional and responsive smart vehicle.

Activity 3: Evaluate shifts to evaluating the practical aspects of our smart remote-controlled car project. We review the actual components used, assess the advantages and disadvantages of the system, and discuss potential directions for future development. This evaluation aims to provide a comprehensive understanding of the project's effectiveness and areas for improvement.

II. Body

Activity1: IoT Overview

1. Definition of IoT

The Internet of Things (IoT) refers to a distributed network connecting physical objects that are capable of sensing or acting on their environment and able to communicate with each other, other machines or computers. The data these devices report can be collected and analysed in order to reveal insights and suggest actions that will produce cost savings, increase efficiency or improve products and services.

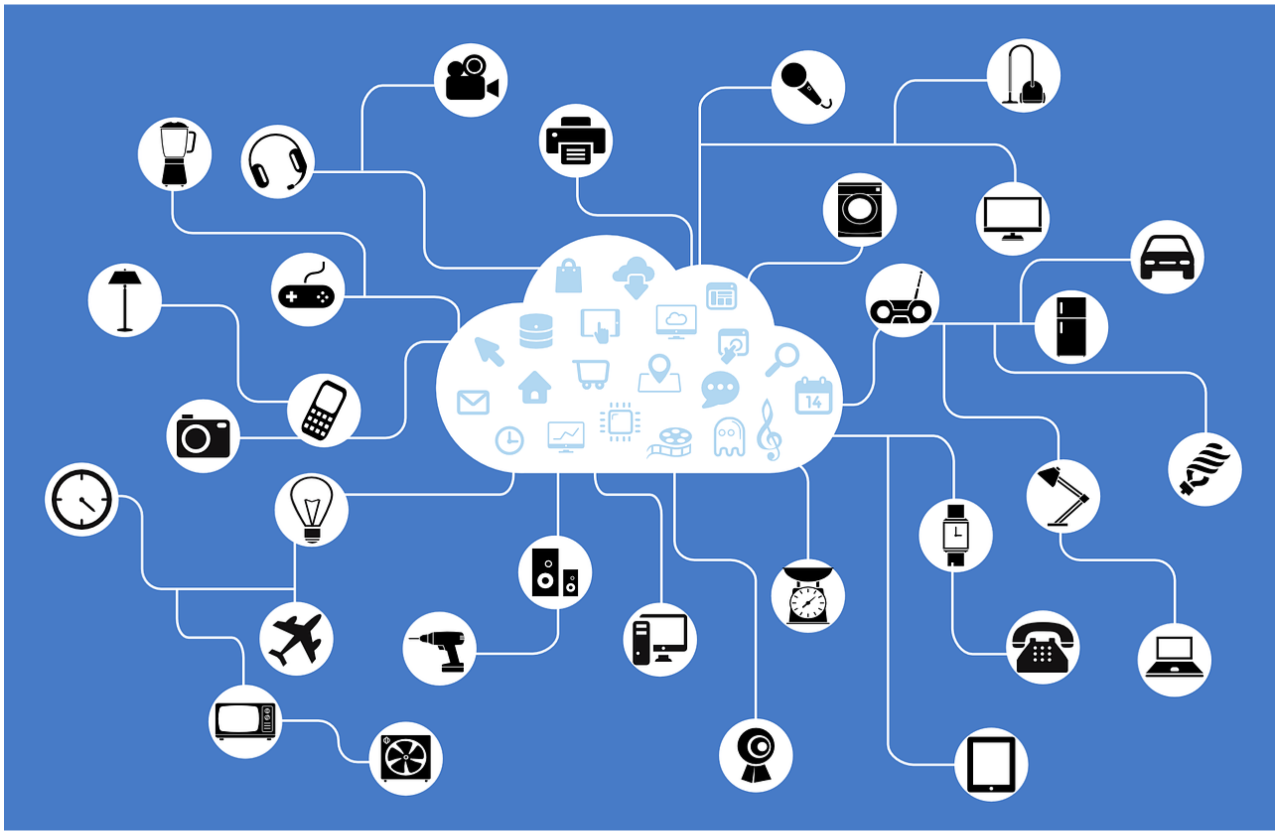

Figure 1: Definition of IOT

2. Role of IoT in Various Fields

2.1. IoT in Smart Homes

The Internet of Things (IoT) has transformed the concept of smart homes by enabling interconnected devices to communicate and automate various household tasks. Smart home technologies encompass systems for lighting, heating, security, and entertainment, all of which can be controlled remotely via smartphones or voice-activated assistants. IoT devices such as smart thermostats, security cameras, smart locks, and intelligent lighting systems enhance energy efficiency, convenience, and security for homeowners. Additionally, they provide real-time data and analytics, facilitating better management of household resources and improving the overall quality of life.

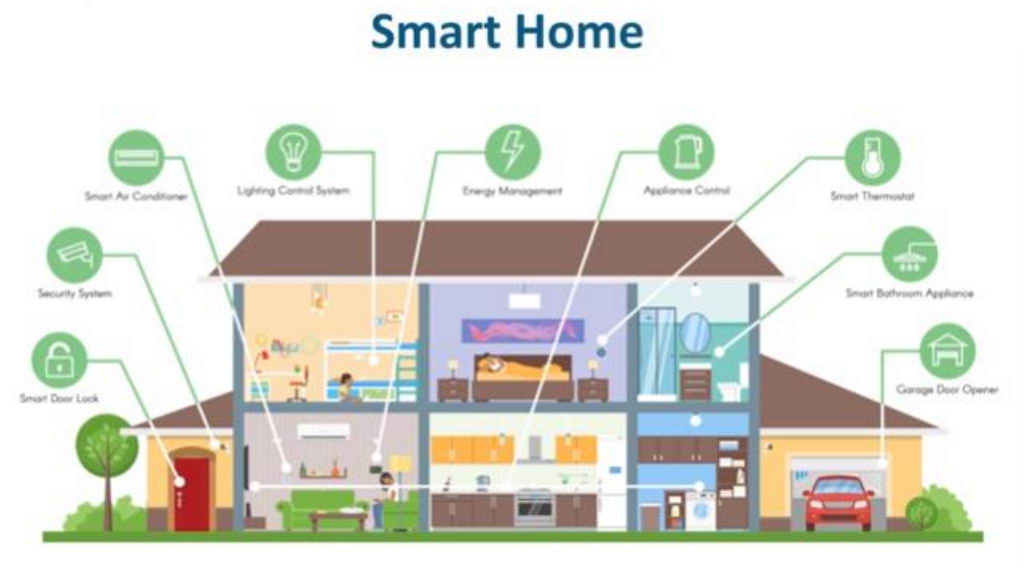

Figure 2: smart home

2.2. IoT in Smart Cities

In smart cities, IoT is pivotal for managing urban infrastructure and services efficiently. IoT technologies are employed to monitor and control various aspects of city life, such as traffic flow, public transportation, waste management, and energy consumption. For instance, smart traffic lights can adjust to real-time traffic conditions, alleviating congestion and enhancing road safety. Smart parking systems assist drivers in locating available parking spaces quickly, while smart meters and sensors optimize energy usage in public buildings and street lighting. The integration of IoT in smart cities results in improved urban living, decreased environmental impact, and more effective resource management.

Figure 3: Smart city

2.3. IoT in Healthcare

In healthcare, IoT has the potential to greatly enhance patient care and streamline medical processes. IoT devices enable remote monitoring of patients' health metrics, such as heart rate, blood pressure, and glucose levels, through wearable devices and smart medical equipment. These devices transmit real-time data to healthcare providers, allowing for timely interventions and personalized treatment plans. Additionally, IoT facilitates the automation of routine tasks, such as medication management with smart dispensers and inventory tracking in hospitals. This connectivity improves patient outcomes, enhances the efficiency of healthcare delivery, and provides valuable insights through data analytics for better health management and preventive care.

Figure 4: IoT in healthcar

2.4. IoT in Smart Transportation

IoT is a fundamental technology in advancing smart transportation systems, significantly improving the efficiency, safety, and sustainability of transportation networks. Connected vehicles and smart infrastructure utilize IoT to enhance traffic management and real-time data analysis. IoT-enabled vehicles can communicate with one another and with traffic management systems to optimize routes, reduce congestion, and decrease emissions. Smart public transportation systems leverage IoT to offer real-time updates on schedules and routes, enhancing the convenience and reliability of transit services. Furthermore, IoT supports the development of autonomous vehicles, which depend on a network of sensors, cameras, and communication devices to navigate safely and efficiently.

Figure 5: IoT in transportation

2.5. IoT in Agriculture (Smart Farming)

IoT technologies have revolutionized agriculture by introducing the concept of smart farming, making the industry more efficient and sustainable. Through the deployment of IoT devices like soil sensors, weather stations, and GPS-enabled equipment, farmers gain access to comprehensive data on soil conditions, crop health, and environmental variables. This wealth of information enables precision agriculture, where irrigation, fertilization, and pest control are tailored to the specific needs of crops. As a result, resource usage is optimized, leading to higher crop yields and more efficient use of water and nutrients.

Figure 6: iot in agriculture

Moreover, IoT technologies extend beyond crop management. They play a crucial role in monitoring livestock health by tracking vital signs and behaviors, which helps in early detection of diseases and improves animal welfare. Additionally, IoT facilitates the management of agricultural supply chains by providing real-time insights into inventory levels, transportation conditions, and storage environments. This ensures the safety and quality of food products as they move from the farm to the consumer. Overall, smart farming driven by IoT enhances productivity, reduces waste, and supports sustainable agricultural practices, contributing to a more resilient and efficient food system.

3. Components of IoT

3.1. Hardware

Hardware constitutes the physical foundation of IoT systems, incorporating a range of devices, sensors, actuators, and other essential components that work together to collect, process, and transmit data. Here’s a more detailed overview of the key hardware elements in IoT:

- **Sensors**: These crucial components are responsible for capturing data from the environment. Sensors can measure a variety of parameters, such as temperature, humidity, light intensity, motion, and pressure. By converting these physical parameters into digital signals, sensors provide the raw data necessary for further processing and analysis. For instance, temperature sensors might be used in smart thermostats to regulate home heating and cooling, while motion sensors could be employed in security systems to detect intrusions.

- **Actuators**: Actuators are devices that perform physical actions based on commands received from the IoT system. They act on the data processed by sensors to effectuate changes in the environment. Examples include turning on a light, opening a valve, adjusting a thermostat, or controlling the movement of machinery. Actuators are vital for implementing automated responses and achieving desired outcomes in various applications.

- **Microcontrollers and Microprocessors**: These serve as the central processing units within IoT devices. Microcontrollers are used in simpler, low-power applications and handle tasks such as reading sensor data and executing basic control functions. Microprocessors, on the other hand, are employed in more complex systems that require higher computational power, such as smart home hubs or industrial control systems. They process large amounts of data and execute more sophisticated operations and algorithms.

- **Communication Modules**: To facilitate interaction between IoT devices and external networks, communication modules are essential. These modules enable data exchange through various wireless technologies, including Wi-Fi, Bluetooth, Zigbee, and cellular networks. By providing connectivity, communication modules allow IoT devices to send and receive information, integrate with other systems, and participate in broader networks of interconnected devices.

- **Power Sources**: Power management is critical for IoT devices, which often operate in environments where frequent maintenance or access to power outlets is not feasible. Many IoT devices rely on batteries, which need to be efficient and long-lasting to ensure continuous operation. Alternatively, some devices might use energy harvesting techniques or other power sources to maintain functionality. Effective power management ensures that devices can operate reliably over extended periods, minimizing downtime and maintenance needs.

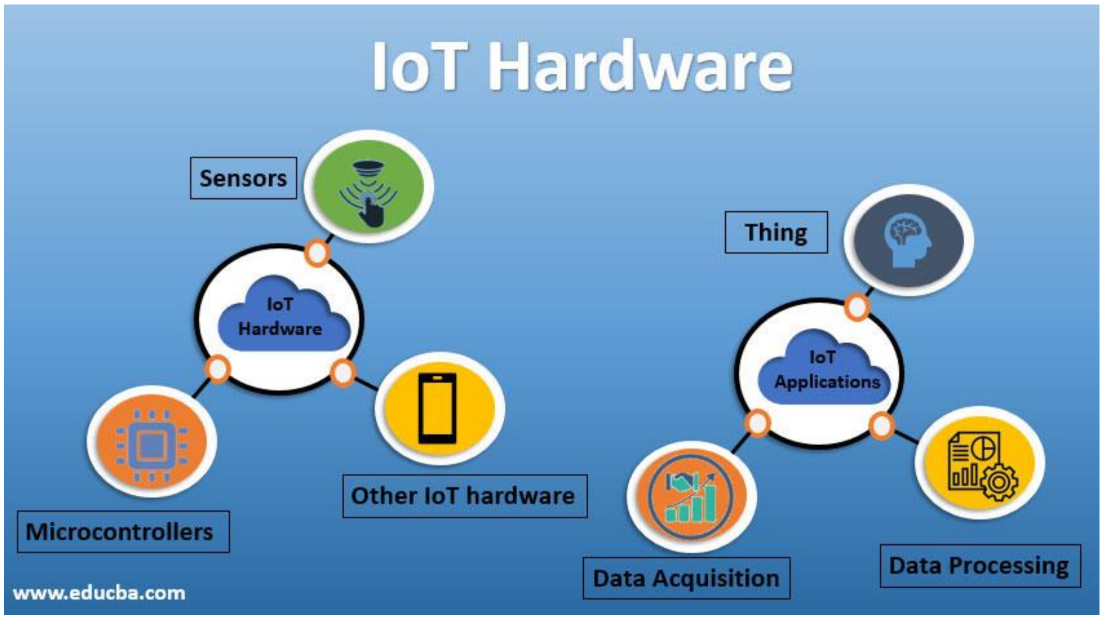

Figure 7: IoT hardware

Together, these hardware components enable IoT systems to collect, process, and act on data, driving advancements across various applications such as smart homes, industrial automation, healthcare, and transportation.

3.2. Software

Software plays a crucial role in the IoT ecosystem by processing, analyzing, and managing the data collected from IoT hardware. The software components involved in IoT systems include:

- **Firmware**: Firmware is a specialized type of low-level software embedded directly into IoT hardware. It provides essential control over the device's fundamental functions, such as handling data acquisition from sensors and enabling communication with other devices. Firmware operates at a close level to the hardware, ensuring that the device performs its basic operations efficiently and reliably. It often includes routines for managing sensor inputs, executing control commands, and facilitating initial data processing.

- **Middleware**: Middleware serves as an intermediary layer between the IoT hardware and higher-level applications. It is designed to manage the complexities of data integration and device communication, providing essential services that enable seamless operation within the IoT ecosystem. Middleware handles tasks such as data aggregation, filtering, and preliminary analysis, ensuring that data from various sources is organized and made available for further processing. It also facilitates interoperability between different hardware devices and software systems, enabling a cohesive and functional IoT network.

- **Cloud Computing**: Cloud platforms are integral to IoT systems, providing the infrastructure needed to store, process, and analyze large volumes of data generated by IoT devices. Cloud computing offers scalability, allowing IoT systems to handle growing amounts of data without the need for extensive on-premises hardware. It also provides flexibility in accessing and utilizing advanced analytics tools and machine learning algorithms, which can be employed to derive insights from the data. Cloud platforms enable centralized data management and support various applications, from simple storage to complex data processing and analysis.

- **Data Analytics Software**: Data analytics software is used to interpret the data collected from IoT devices, transforming raw data into actionable insights. This software includes a range of analytical tools, such as real-time analytics for immediate insights, historical data analysis for trend identification, predictive analytics for forecasting future events, and visualization tools for presenting data in an understandable format. By processing and analyzing the data, this software helps users make informed decisions, optimize operations, and enhance overall system performance.

- **User Applications**: User applications provide the interfaces through which individuals interact with the IoT system. These applications can take various forms, including mobile apps, web-based interfaces, or desktop software. They offer functionalities such as device control, real-time data monitoring, and notification services. User applications are designed to be intuitive and user-friendly, allowing users to manage their IoT devices, view data, and receive alerts or updates as needed. They play a crucial role in making the complex capabilities of IoT systems accessible and manageable for end-users.

Figure 8: IoT software

Together, these software components enable IoT systems to function effectively, turning raw data into valuable insights, managing device interactions, and providing users with intuitive tools to control and monitor their IoT-enabled environments.

3.3. Connectivity

Connectivity is crucial for IoT systems, enabling communication between devices and networks. Key aspects include:

- **Communication Protocols**: These protocols, such as MQTT, HTTP/HTTPS, CoAP, and AMQP, standardize data exchange between IoT devices and systems, ensuring effective messaging and data management.

- **Wireless Communication Technologies**: Technologies like Wi-Fi, Bluetooth, Zigbee, LoRa, and NB-IoT provide different ranges and power efficiencies for wireless data transmission, suitable for various IoT applications from smart homes to industrial monitoring.

- **Cellular Networks**: 4G LTE and 5G offer wide-area connectivity, supporting high-speed data transfer and low latency for mobile and remote IoT applications, enhancing performance in areas like autonomous vehicles and smart cities.

- **Network Topologies**: Common topologies, such as star and mesh, determine how IoT devices are connected and communicate, affecting network management and coverage.

- **Gateway Devices**: Gateways aggregate data from IoT devices, manage communication protocols, and facilitate data transfer to cloud services or other systems, playing a key role in bridging devices and networks.

Figure 9: IoT connectivity

These connectivity elements ensure IoT devices can effectively exchange data and integrate with other systems, crucial for the overall functionality and efficiency of IoT solutions.

3.4. Security

Security is a critical aspect of IoT systems, ensuring that data and devices are protected from unauthorized access, breaches, and other threats. Key security measures for IoT include:

- Authentication and Authorization: Ensures that only authorized users and devices can access or control IoT systems. This involves using credentials, such as passwords, biometrics, or digital certificates, and establishing permissions to limit access to sensitive data and functions.

- Encryption: Protects data during transmission and storage by converting it into a secure format that can only be read by authorized parties. Common encryption methods include SSL/TLS for data in transit and AES for data at rest.

- Network Security: Safeguards IoT networks from attacks and unauthorized access. This includes using firewalls, intrusion detection systems (IDS), and secure communication protocols to protect data exchanges between devices and networks.

- Firmware and Software Updates: Regular updates and patches are essential to address vulnerabilities and enhance security. IoT devices should support secure update mechanisms to protect against known threats and exploits.

- Data Integrity: Ensures that data remains accurate and unaltered during transmission and storage. Techniques such as hashing and digital signatures are used to verify that data has not been tampered with.

- Device Management: Involves monitoring and managing IoT devices to detect and respond to security threats. This includes implementing secure boot processes, managing device configurations, and performing regular security audits.

- Privacy Protection: Ensures that personal and sensitive information collected by IoT devices is protected and used in compliance with privacy regulations. This involves data anonymization, secure data storage, and clear privacy policies.

- Incident Response and Recovery: Establishes procedures for responding to security breaches and recovering from attacks. This includes having an incident response plan, conducting regular security drills, and maintaining backups to restore operations if needed.

Figure 10: IoT security

Implementing these security measures helps protect IoT systems from potential threats, ensuring data confidentiality, integrity, and availability while maintaining user trust.

Activity 2: Smart Remote-Controlled Cars

1. Use Cases of Smart Remote-Controlled Cars

   Smart remote-controlled (RC) cars, equipped with components like the Arduino WiFi ESP8266 WeMos D1 R2 and controlled via the Blynk IoT platform, offer advanced control and monitoring features. Key use cases include:

   - **Remote Control**: Users can operate the car wirelessly from a smartphone using the Blynk app, providing convenience and flexibility in controlling the vehicle.
   - **Real-Time Feedback**: The Blynk app delivers real-time updates on the car's status, including speed and battery level, allowing users to make informed decisions during operation.
   - **Educational Tool**: The system serves as an excellent educational resource for learning about microcontrollers, motor control, and IoT applications.

   Usage Scenarios and Operational Modes:

   ➢ **Manual Driving Mode:**
   - **Operational Mode**: Real-Time Driving
   - **User Interaction**: Users control the car's movements through the Blynk app, adjusting speed and direction.
   - **System Response**: The Arduino processes Wi-Fi signals and controls the motors, providing immediate feedback to user inputs.

   ➢ **Autonomous Navigation Mode:**
   - **Operational Mode**: Pre-Defined Routes or Obstacle Detection
   - **User Interaction**: Users set a route or activate obstacle detection via the Blynk app.
   - **System Response**: The car uses onboard sensors and the Arduino for autonomous navigation, avoiding obstacles or following a set path.

   ➢ **Telemetry Monitoring:**
   - **Operational Mode**: Data Collection and Display
   - **User Interaction:** The Blynk app shows data such as speed, distance, and battery status.
   - **System Response**: The Arduino gathers this data and sends it to the Blynk server for visualization.

2. Components of Smart Remote-Controlled Cars

   The smart RC car system is built from several key components, each playing a crucial role in its operation. Detailed descriptions of these components are as follows:

   2.1. Programming Cable
   - **Function**: Used to upload code and configurations to the Arduino board.
   - **Detail**: Typically a USB cable, it connects the Arduino to a computer for programming. It transfers the firmware and control algorithms that dictate how the Arduino interacts with sensors, motors, and other peripherals.

   2.2.Arduino WiFi ESP8266 WeMos D1 R2
   - **Function**: Acts as the central microcontroller, managing all aspects of control, processing, and communication.

- **Detail**: This board combines the functionality of a microcontroller with built-in Wi-Fi capabilities, allowing it to connect to the Blynk IoT platform. The ESP8266 chip handles network communication, while the WeMos D1 R2 provides the necessary I/O pins for interacting with other components.

2.3.Jumper Wires (Male-to-Male and Male-to-Female)

- **Function**: Facilitate electrical connections between different components of the RC car.
- **Detail**:
- ➢ **Male-to-Male Jumper Wires:** Used for connecting the Arduino's output pins to input pins on modules or sensors.
- ➢ **Male-to-Female Jumper Wires**: Connect the Arduino's pins to other components like the motor driver or sensors, making it easier to interface different parts of the system.

2.4. 9V DC Jack and 9V Battery

- **Function**: Provide power to the Arduino and other electronic components.
- **Detail**:
- ➢ **9V DC Jack**: Connects to an external power source to supply the Arduino with a stable voltage, ensuring reliable operation.
- ➢ **9V Battery**: Offers a portable power solution for the Arduino and associated electronics, making the RC car operable without being tethered to a fixed power source.

2.5. Smart Robot Chassis with Three Wheels and Two 3-9V DC Motors

- **Function**: Provides the structural base and movement mechanism for the RC car.
- **Detail**:
- ➢ **Chassis**: The frame of the car, which supports all other components and allows for assembly and stability.
- ➢ **Wheels and Motors**:
  - – **Wheels**: Typically include two driven wheels and one free-spinning wheel for balance.
  - – **Motors**: Two DC motors (rated 3-9V) drive the wheels, allowing the car to move forward, backward, and turn. These motors receive power and control signals from the L298 motor driver module.

2.6. L298 Motor Driver Module

- **Function**: Controls the speed and direction of the DC motors.
- **Detail**:
- ➢ **Motor Control**: The L298 module translates the digital control signals from the Arduino into the appropriate voltage and current to drive the motors. It allows for precise control over the motor speed and direction by using pulse-width modulation (PWM) signals from the Arduino.
- ➢ **Connections**: It interfaces with the Arduino through input pins for direction and speed control and outputs to the motors. It also requires a separate power supply for the motors, which is different from the Arduino's power source.

System Integration and Communication:

- **Arduino Programming**: The Arduino is programmed using a computer, with code uploaded through the programming cable. This code defines how the Arduino reads sensor data, processes commands, and controls the motors.
- **Blynk App Integration**: The Blynk app serves as the user interface, allowing remote control and monitoring of the RC car. It communicates with the Arduino over Wi-Fi, sending commands and receiving data.
- **Wi-Fi Communication**: The ESP8266 module in the Arduino enables Wi-Fi connectivity, allowing the RC car to connect to the internet and interact with the Blynk server.

Data Communication Standards:

- **Wi-Fi**: Facilitates the remote control and monitoring of the RC car via the Blynk app, leveraging the ESP8266's wireless capabilities for internet connectivity.
- **UART (Universal Asynchronous Receiver-Transmitter):** Handles serial communication between the Arduino and the Wi-Fi module, enabling data exchange and control signals.
- **I2C/SPI**: Protocols used for communication between the Arduino and various sensors and peripherals, ensuring efficient data transfer and system integration. I2C (Inter-Integrated Circuit) and SPI (Serial Peripheral Interface) are used based on the specific requirements of the sensors and modules in the system.

This detailed breakdown of components and their roles highlights the complexity and functionality of the smart RC car system, demonstrating how various elements work together to enable advanced remote control and monitoring capabilities.

3. Electronic control circuit

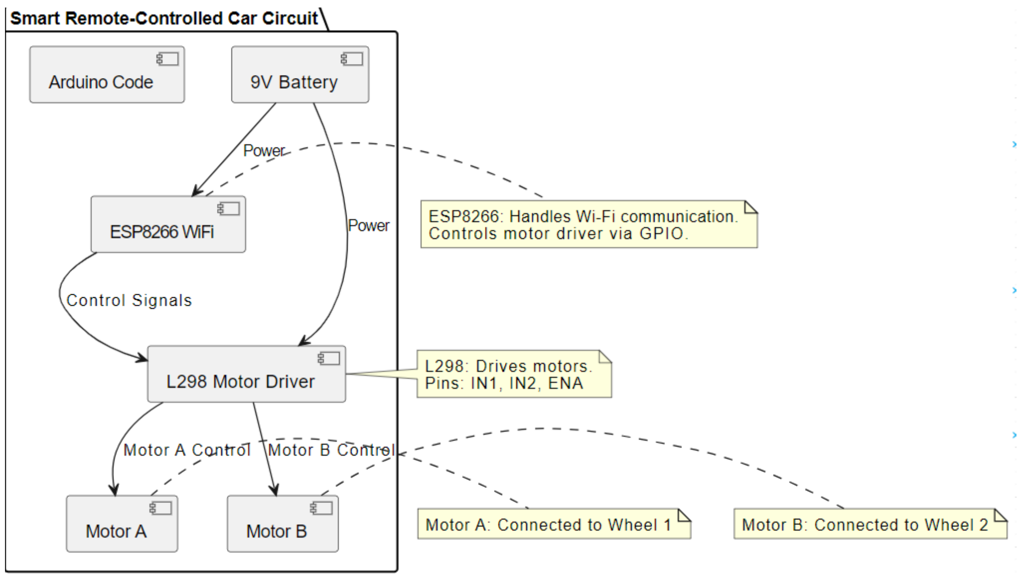
Smart Remote-Controlled Car Circuit
Arduino Code
9V Battery
Power
ESP8266 WiFi
Power
ESP8266: Handles Wi-Fi communication.
Controls motor driver via GPIO.
Control Signals
L298 Motor Driver
L298: Drives motors.
Pins: IN1, IN2, ENA
Motor A Control
Motor B Control
Motor A
Motor B
Motor A: Connected to Wheel 1
Motor B: Connected to Wheel 2

Figure 11: Smart remote-controlled car circuit

Here’s the explanation of the components and their functions in the schematic diagram for the smart remote-controlled car system:

ESP8266 WiFi

- **Function**: The ESP8266 is a microcontroller with built-in Wi-Fi capabilities, used for communicating with the Blynk app and controlling the L298 motor driver.
- **Purpose:**

➢ Receives commands from the Blynk app via the Wi-Fi connection.
➢ Processes these commands and converts them into control signals for the L298 motor driver.
➢ Provides remote control capabilities for the car over the internet.

L298 Motor Driver

- **Function**: The L298 is a dual H-bridge motor driver IC used to control the speed and direction of DC motors.
- **Purpose:**

➢ Receives control signals from the ESP8266 and manages the operation of motors A and B.
➢ Provides control signals (IN1, IN2, ENA) to adjust the direction and speed of the motors.
➢ Controls the motor’s direction (forward, reverse, left, right) and speed via its control pins.

9V Battery

- **Function**: The 9V battery provides power to both the ESP8266 and the L298 motor driver.
- **Purpose**:

➢ Supplies a stable power source for the ESP8266 and L298 motor driver to operate.
➢ Ensures continuous operation of the system without requiring an external power source.

Motor A and Motor B

- **Function**: DC motors attached to the car, which drive the wheels of the robot car.
- **Purpose**:

➢ Provide movement to the car, allowing it to drive in the directions specified by the control system.
➢ Controlled by the L298 motor driver to adjust speed and direction.

Connections Between Components

- **Connection Between 9V Battery and ESP8266:**

➢ Supplies power to the ESP8266, enabling it to perform control functions and communicate with the Blynk app.

- **Connection Between 9V Battery and L298 Motor Driver:**

➢ Provides power to the L298 motor driver to control the motors.

- **Connection Between ESP8266 and L298 Motor Driver:**

➢ Sends control signals from the ESP8266 to the L298, including direction and speed commands for the motors.

- **Connection Between L298 Motor Driver and Motor A/B:**

➢ Transmits control signals from the L298 to Motor A and Motor B to manage the car’s movement.

4. Algorithm flowchart

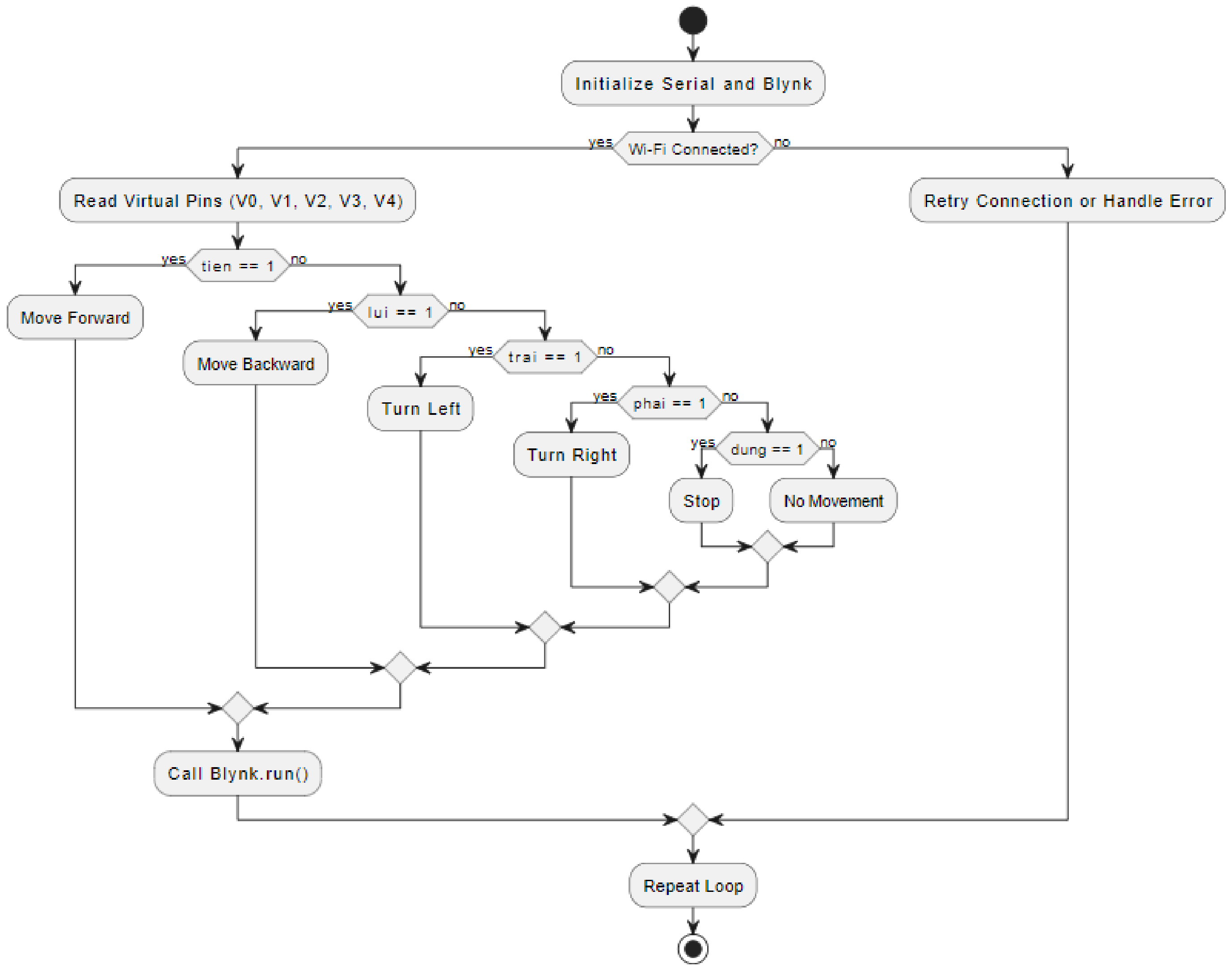

Figure 12: Flowchart

Start

- **Action**: Begin the execution of the code.

Initialize Serial and Blynk

- **Action**: Set up the serial communication and initialize the Blynk library for IoT connectivity. This prepares the system to communicate with the Blynk app and handle incoming data.

Check Wi-Fi Connection

- **Decision**: Verify if the ESP8266 has successfully connected to the Wi-Fi network.
- ➢ **Yes**: Proceed to the next steps.
- ➢ **No**: Handle connection errors or retry connecting.

Read Virtual Pins (V0, V1, V2, V3, V4)

- **Action**: Read the values from the virtual pins defined in the Blynk app. Each pin corresponds to a specific control command.
- **V0**: Move Forward
- **V1**: Turn Left
- **V2**: Move Backward
- **V3**: Turn Right
- **V4**: Stop

Decision Making Based on Pin Values

- **Decision**: Determine the action based on the values read from the virtual pins.
- **If tien == 1 (Move Forward):**
  - Action: Set the motor control pins to move the car forward.
- **Else If lui == 1 (Move Backward):**
  - Action: Set the motor control pins to move the car backward.
- **Else If trai == 1 (Turn Left):**
  - Action: Set the motor control pins to turn the car left.
- **Else If phai == 1 (Turn Right):**
  - Action: Set the motor control pins to turn the car right.
- **Else If dung == 1 (Stop):**
  - Action: Stop all movement by turning off the motors.
- **Else:**
  - Action: No movement; all motors are off.

Call Blynk.run()

- **Action**: Continuously run the Blynk library to handle communication with the Blynk app and process any incoming commands.

Repeat Loop

- **Action**: Continuously repeat the loop to keep processing new commands and updating the motor actions based on user input from the Blynk app.

End

- **Action**: The process will continue indefinitely, as the loop repeats and the ESP8266 keeps listening for new commands and updates.

5. Coding

```
/*************************************************************

  This sketch shows how to read values from Virtual Pins

  App dashboard setup:
    Slider widget (0...100) on Virtual Pin V1
 *************************************************************/

/* Fill-in information from Blynk Device Info here */
```

```
#define BLYNK_TEMPLATE_ID "TMPL69YvHXeiH"
#define BLYNK_TEMPLATE_NAME "Vehicle controlled"
#define BLYNK_AUTH_TOKEN            "NKktGaHkhq91fIvHF-5U6CeXL4tir2dk"

/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "phone";
char pass[] = "88888888";
int tien = 0;
int lui = 0;
int trai = 0;
int phai = 0;
int dung = 0;
// This function will be called every time Slider Widget
// in Blynk app writes values to the Virtual Pin V1
BLYNK_WRITE(V0)
{
  tien = param.asInt(); // assigning incoming value from pin V1 to a variable

  // process received value
}
BLYNK_WRITE(V1)
{
  trai = param.asInt(); // assigning incoming value from pin V1 to a variable

  // process received value
}
BLYNK_WRITE(V2)
{
  lui = param.asInt(); // assigning incoming value from pin V1 to a variable

  // process received value
}
BLYNK_WRITE(V3)
{
  phai = param.asInt(); // assigning incoming value from pin V1 to a variable

  // process received value
}
```

```
BLYNK_WRITE(V4)
{
  dung = param.asInt(); // assigning incoming value from pin V1 to a variable

  // process received value
}


void setup()
{

  pinMode(D4,OUTPUT);
  pinMode(D5,OUTPUT);
  pinMode(D6,OUTPUT);
  pinMode(D7,OUTPUT);

  // Debug console
  Serial.begin(115200);

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  // You can also specify server:
  //Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass, "blynk.cloud", 80);
  //Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass, IPAddress(192,168,1,100), 8080);
}

void loop()
{
  if(tien == 1 ){
    digitalWrite(D4,1);
    digitalWrite(D5,0);
    digitalWrite(D6,1);
    digitalWrite(D7,0);


  }else if(lui == 1){
    digitalWrite(D4,0);
    digitalWrite(D5,1);
    digitalWrite(D6,0);
    digitalWrite(D7,1);

  }else if( trai == 1){

    digitalWrite(D4,1);
    digitalWrite(D5,0);
    digitalWrite(D6,0);
    digitalWrite(D7,0);
  }else if( phai == 1){
    digitalWrite(D4,0);
    digitalWrite(D5,0);
    digitalWrite(D6,1);
    digitalWrite(D7,0);
    }else if (dung ==1){
      digitalWrite(D4,0);
    digitalWrite(D5,0);
    digitalWrite(D6,0);
    digitalWrite(D7,0);

    }


  Blynk.run();
}
```

This code is designed to control a vehicle remotely using the Blynk app. The vehicle can move in different directions (forward, backward, left, right, stop) based on commands sent from the Blynk app to the ESP8266 WiFi module, which then controls the motors connected to the vehicle. Here's a detailed explanation:

Libraries and Definitions:

- ESP8266WiFi.h: Allows the ESP8266 to connect to a WiFi network.
- BlynkSimpleEsp8266.h: Provides functionality to connect the ESP8266 to the Blynk platform, enabling remote control via the Blynk app.

Blynk Authentication and WiFi Credentials:

- BLYNK_TEMPLATE_ID, BLYNK_TEMPLATE_NAME, and BLYNK_AUTH_TOKEN are identifiers and tokens for authenticating your device with Blynk.
- ssid and pass are the WiFi network name and password that the ESP8266 will connect to.

Global Variables:

- The variables tien, lui, trai, phai, dung represent the state of different vehicle controls: forward (tien), backward (lui), left (trai), right (phai), and stop (dung).

Blynk Virtual Pin Handlers:

- The functions BLYNK_WRITE(V0), BLYNK_WRITE(V1), BLYNK_WRITE(V2), BLYNK_WRITE(V3), BLYNK_WRITE(V4) are called whenever the value of the corresponding Virtual Pin (V0 to V4) in the Blynk app changes. The received value is assigned to the respective variables.

Setup Function:

- The pins D4, D5, D6, D7 are configured as output pins to control the motors.
- Serial communication is initialized for debugging.
- The ESP8266 is connected to the Blynk platform using the authentication token and WiFi credentials.

Main Loop Function:

- The loop checks the state of the control variables and sets the corresponding output pins high (1) or low (0) to control the motors:
- Forward: Pins D4 and D6 are set high, and D5 and D7 are set low.
- Backward: Pins D5 and D7 are set high, and D4 and D6 are set low.
- Left: Pin D4 is high, and the others are low.
- Right: Pin D6 is high, and the others are low.
- Stop: All pins are set low.
- The Blynk.run() function keeps the Blynk connection alive and processes incoming commands.

Activity3 : Evaluate

1. Actual Product

1.1. Code upload wire

Fiure 13: Code upload wire

1.2. Arduino WiFi ESP8266 WeMos D1 R2

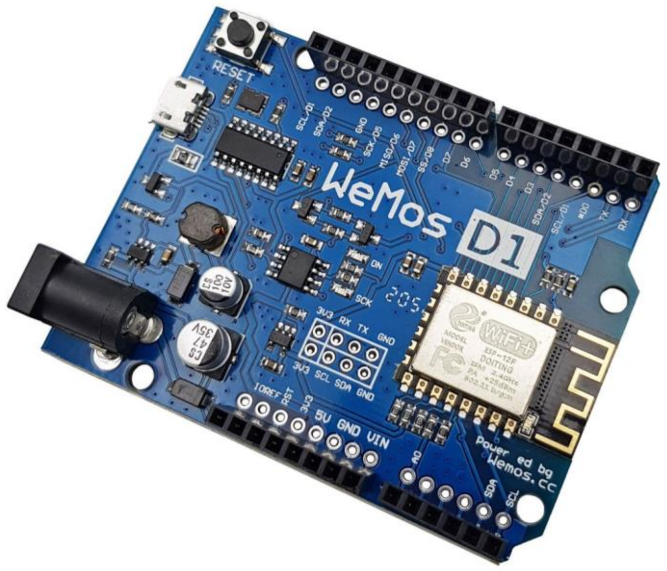

Figure 14: Arduino WiFi ESP8266 WeMos D1 R2

1.3. Male-male connecting wire

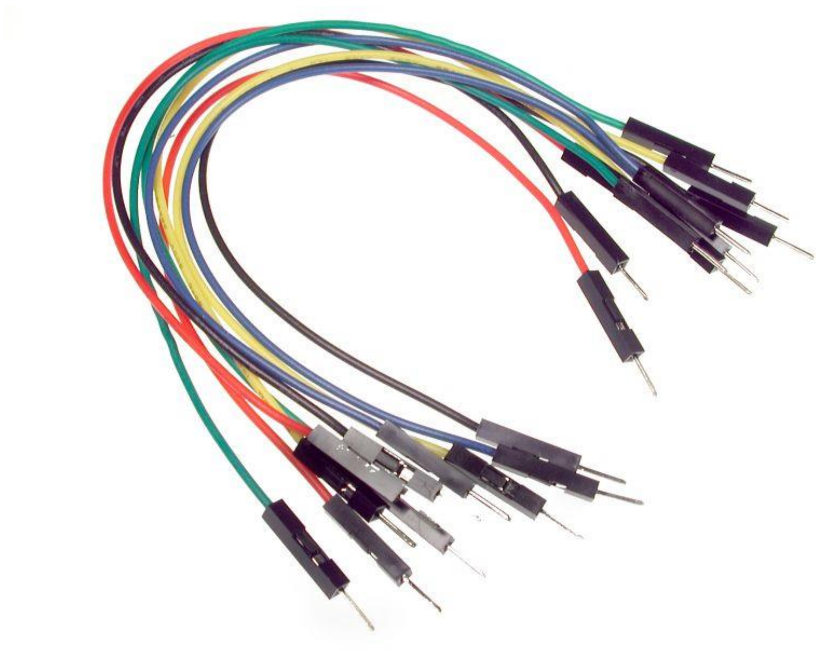

Figure 15: Male-male connecting wire

1.4. Male-female connecting wire

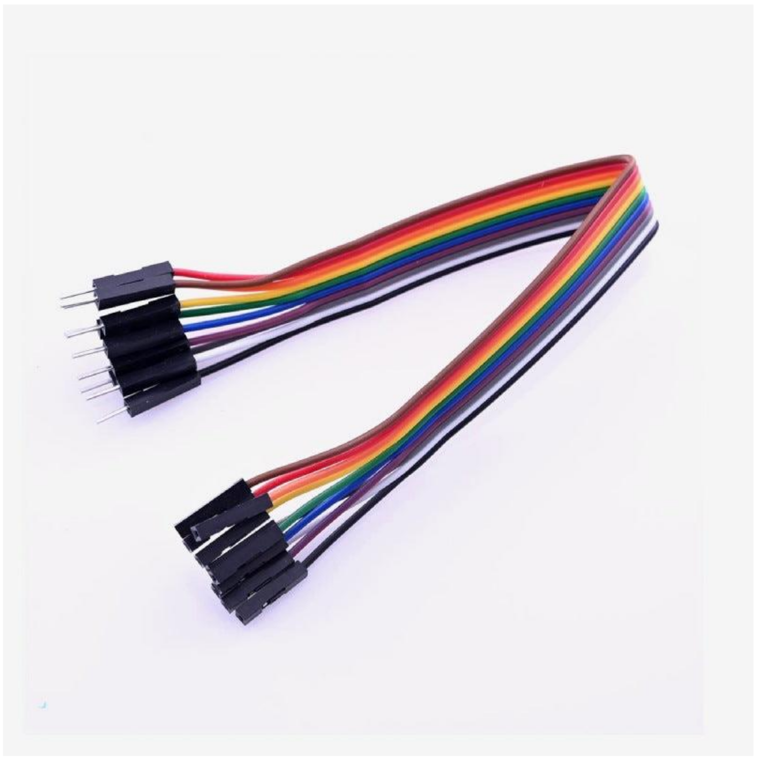

Figure 16: Male-female connecting wire

1.5. DC 9V jack for Arduino

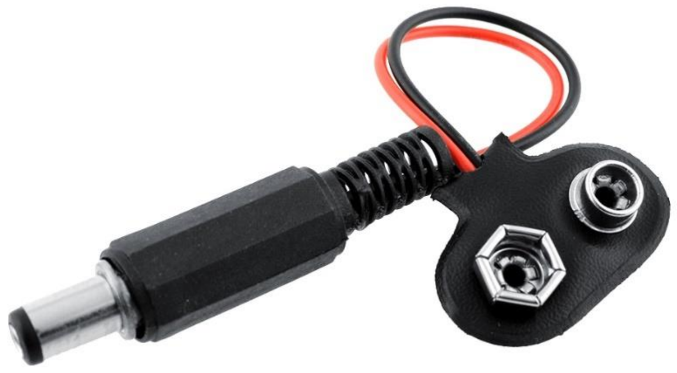

Figure 17: DC 9V jack for Arduino

1.6. Smart 3-wheel Robot chassis and 2 3-9V gear motors

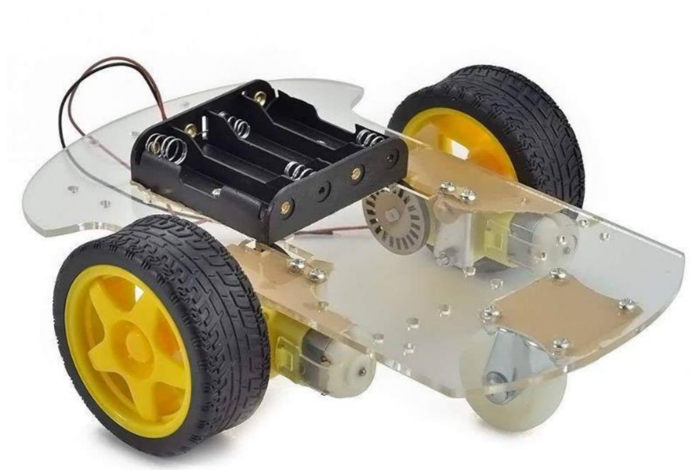

Figure 18: Smart 3-wheel Robot chassis and 2 3-9V gear motors

1.7. L298 circuit

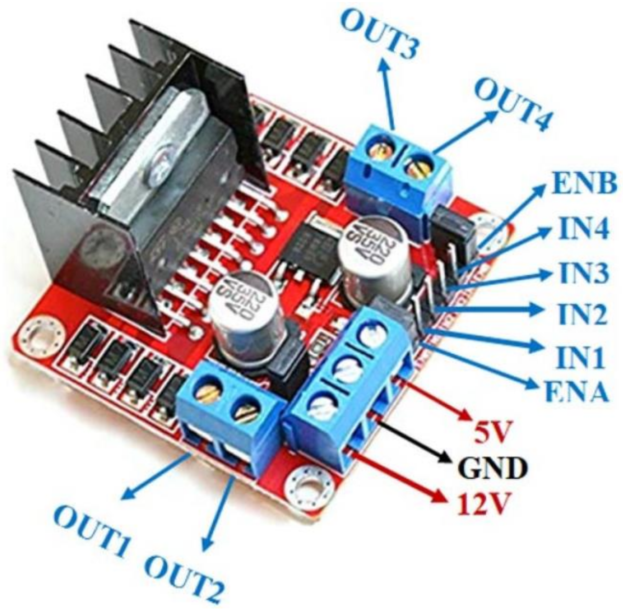

Figure 19: L298 circuit

1.8. 9V battery for Arduino

Figure 20: 9V battery for Arduino

1.9. Finished product

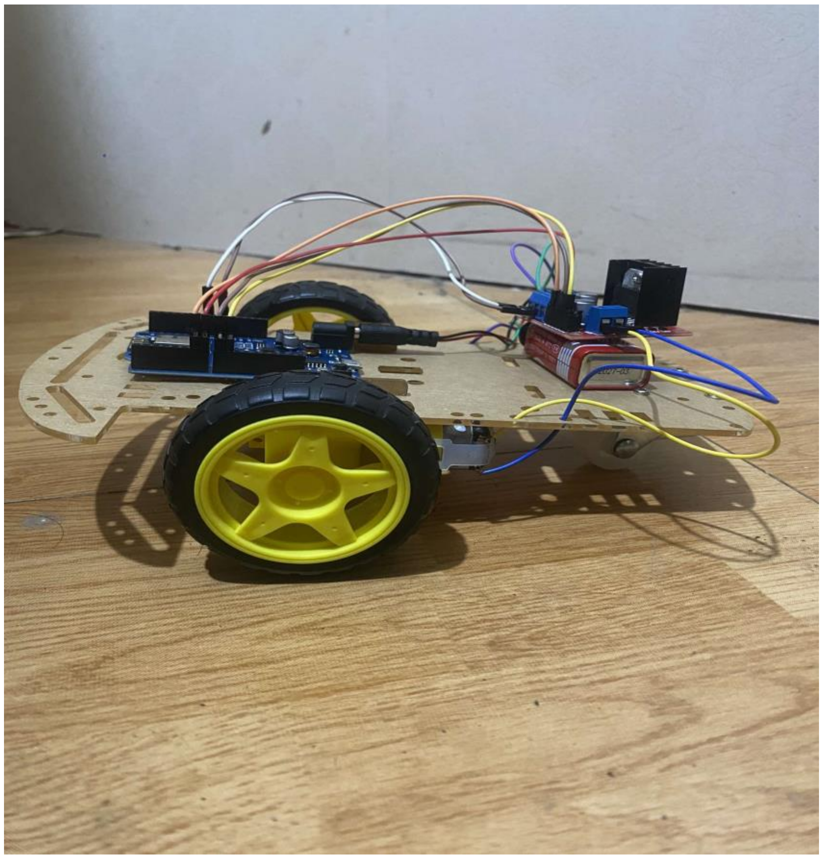
Figure 21: Finished product

2. Advantages and Disadvantages of the System

   Advantages:

   - **Remote Control via Network:**
     - ➢ Description: Utilizing the Blynk app and a Wi-Fi connection allows users to control the RC car remotely.
     - ➢ Benefit: Offers significant convenience and flexibility, enabling operation from anywhere within the Wi-Fi range.
   - **Real-Time Feedback:**
     - ➢ **Description**: The system provides live updates on important metrics like speed, battery level, and car status.
     - ➢ **Benefit**: Facilitates immediate monitoring and adjustments, enhancing user control and decision-making.
   - **Mobility**:
     - ➢ **Description**: The compact design, powered by a 9V battery, ensures the car's portability.
     - ➢ **Benefit**: Allows the car to operate independently of external power sources, enhancing mobility and ease of use.

Disadvantages:

- **Dependence on Network Connection:**
  - ➢ **Description**: The car’s control is reliant on a stable Wi-Fi network.
  - ➢ **Drawback**: Limits usability in areas with poor or no network coverage, potentially restricting the car's functionality.
- **Power Supply Requirement:**
  - ➢ **Description**: The system needs a continuous power supply from the 9V battery or an external source.
  - ➢ **Drawback**: Limits operation time and requires regular battery replacement or recharging to maintain functionality.
- **Limited Movement:**
  - ➢ **Description**: The car’s movement capabilities are restricted to basic actions such as turning left or right.
  - ➢ **Drawback**: May constrain its use in more complex scenarios that require advanced maneuverability.

3. Future Product Development

   To further improve the smart RC car and enhance its user experience, several key development areas can be explored:

- **Offline Functionality:**
  - ➢ **Proposal**: Introduce Bluetooth connectivity alongside or instead of Wi-Fi.
  - ➢ **Advantage**: This would enable control of the car in environments where Wi-Fi is unavailable, increasing its versatility and range of use.

- **Extended Battery Life:**
  - ➢ **Proposal**: Develop advanced power management solutions or use alternative energy sources.
  - ➢ **Advantage**: Enhances the car’s operational longevity, allowing for longer playtime and reducing the need for frequent battery changes.

- **Enhanced Navigation Capabilities:**

  - ➢ **Proposal**: Integrate advanced sensors and more complex algorithms for obstacle avoidance and autonomous driving.
  - ➢ **Advantage**: Improves the car’s ability to navigate complex environments and perform tasks with greater autonomy and precision.

- **Expanded Movement Abilities:**
  - ➢ **Proposal**: Redesign the chassis and refine the control mechanisms to support a wider range of movements.
  - ➢ **Advantage**: Provides more dynamic movement options, such as moving forward and backward, and offers better steering control for diverse driving scenarios.

- **Upgraded User Interface:**
- **Proposal**: Enhance the Blynk app with additional features and customization options.
- **Advantage**: Improves user interaction with the car, making the app more intuitive and offering a more personalized and enjoyable control experience.

III. Conclusion

This report provides a comprehensive overview of IoT technology and its applications, illustrating how it revolutionizes various domains. From enhancing home automation to optimizing urban infrastructure and improving agricultural practices, IoT demonstrates its versatility and potential to transform industries.

The detailed exploration of smart remote-controlled cars in Activity 2 showcases a practical example of IoT application, highlighting the interplay between hardware components, software control, and connectivity. Through the examination of electronic control circuits and algorithm flowcharts, we gained valuable insights into the technical workings of smart vehicles.

Activity 3 provides a critical evaluation of the smart remote-controlled car project, assessing its performance, identifying strengths and weaknesses, and suggesting future improvements. This evaluation underscores the importance of continuous development and innovation in harnessing the full potential of IoT technology.

Overall, the integration of IoT in smart devices, such as remote-controlled cars, exemplifies the broader impact of IoT on technology and daily life. As IoT continues to evolve, it promises to bring even greater advancements and opportunities across various fields.

IV. References

Al-Furaydi, A. (2020). The Internet of Things (IoT) Overview. [online] IoT For All. Available at: https://www.iotforall.com/internet-of-things-iot-overview/.

Smith, J. (2018). Smart Home Technology: Revolutionizing Modern Living. [online] TechCrunch. Available at: https://techcrunch.com/2018/02/10/smart-home-technology/.

Jones, M. (2019). The Role of IoT in Smart Cities. [online] Smart Cities World. Available at: https://www.smartcitiesworld.net/news/news/the-role-of-iot-in-smart-cities-2652.

Brown, L. (2021). Healthcare Innovations: IoT's Impact on Medical Devices. [online] HealthTech Magazine. Available at: https://healthtechmagazine.net/article/2021/03/healthcare-innovations-iots-impact-medical-devices.

Davis, R. (2022). Autonomous Vehicles and the Future of Smart Transportation. [online] Wired. Available at: https://www.wired.com/story/autonomous-vehicles-future-smart-transportation/.

Clark, K. (2020). Smart Farming: IoT Applications in Agriculture. [online] AgFunder Network Partners. Available at: https://agfundernetwork.com/smart-farming-iot-applications/.

Kumar, P. (2019). Introduction to IoT Hardware and Software. [online] Techopedia. Available at: https://www.techopedia.com/definition/31230/internet-of-things-iot-hardware-and-software.

Green, A. (2018). Connectivity in IoT: Standards and Protocols. [online] Network World. Available at: https://www.networkworld.com/article/328632/connectivity-in-iot-standards-and-protocols.html.

Williams, S. (2021). Ensuring IoT Security: Challenges and Solutions. [online] Security Magazine. Available at: https://www.securitymagazine.com/article/9206-ensuring-iot-security-challenges-and-solutions.

Taylor, H. (2022). Components of Smart Remote-Controlled Cars. [online] Electronics Weekly. Available at: https://www.electronicsweekly.com/components/smart-remote-controlled-cars/.