TABLE OF CONTENTS

Internet of Things (IoT):

The Internet of Things (IoT) describes the network of physical objects “things” that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet.

Key Component of IoT:

1. Sensors
2. Connectivity
3. Data Processing
4. Actuators

1. Sensors:

A sensor is an appliance that detects changes in physical or electrical or other quantities. So, it produces an electrical or optical signal output as an acknowledgement of the change in that specific quantity. So, a Sensor is a module or chip that observes the changes happening in the physical world and sends feedback to the microcontroller or microprocessor.

1.1 **Analog Sensors:** Analog Sensors produce continuous analog output signals, proportional to its measurement. It provides reading through environment (0-1024 Units). A few examples of analog sensors are: accelerometers, pressure sensors, light, and sound sensors.

1.2 **Digital Sensors:** Digital Sensors convert the data transmission, digitally. It works in a binary form set of 0_s and 1_s raw form. Examples include digital accelerometers, pressure, and temperature sensors.

2. Connectivity:

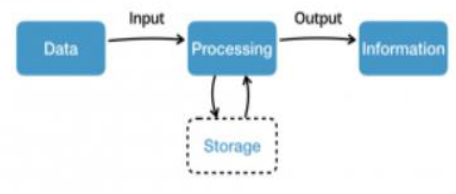
IoT connectivity is the technology and processes that allow Internet of Things (IoT) devices to send and receive data. IoT devices can include sensors, robots, streetlights, and self-driving vehicles. IoT connectivity is essential because it allows devices to become an intra-net, which is a series of interconnected devices that can communicate and exchange data. This allows organizations to make data-driven decisions.

1.1 Wired connectivity: Wired communication is a technology that uses a tangible medium such as optical fibers and metal wires to transmit information. Currently, wired communication is very popular, such as telephone lines, internet cables, TV lines, etc.

1.2 Wireless connectivity: Wireless communication is a communication technology that uses electromagnetic wave signals that travel through space to exchange information. Common wireless communication technologies include cellular wireless connections, wi-fi connections, Bluetooth connections, visible light communication and quantum communication.

3. Data Processing:

Data processing in IoT is the process of collecting and manipulating data to produce useful information. The data collection process uses sensors to monitor the functionality of IoT devices. These devices are equipped with sensors and connectivity, continuously gathering information from the surrounding environment. This data may include temperature, humidity, pressure, location, and more.



Classification: Data is classified into different groups.
Sorting: Data is arranged in some kind of an order
Calculation: Arithmetic and logical operations are performed on numeric data.

1. On-device: For simple operations, simple devices may process data locally. That is on the sensor itself.
2. On the cloud: Data is sent to remote servers for analysis and storage. This makes it possible to perform more complex calculations, do historical analysis, and handle data from several devices centrally.

4. Actuators:

An actuator is a device that converts energy into motion. It does this by taking an electrical signal and combining it with an energy source. In an IoT system, the actuator can act on data collected by sensors to create an outcome as determined by the chosen settings of the user.

Levels of the IoT-

1. Basic: on the basic level of IoT we have the concept of the device that we aim to develop and this is the initial level where we do trial and error. On this level we do not have internet connectivity.

2. Prototype: Making a functional model of our IoT solution is the first step towards moving it to the prototype Level. This could be a simulation that shows the fundamental features or a real gadget integrating sensors and actuators. The objective is to verify and test your idea's viability. But still no connection.

3. Advanced Prototype: An improved prototype expands on the original idea by adding new features and improving the design. Here we finally provide connection to the device (internet/cloud).

4. Prototype: A prototype becomes a solution that is ready for the market at the product Level. It entails mass production as well as cost-effective, scalable, and reliable optimization. The product is now prepared for usage by a larger user base and commercial distribution.

5. Use-Based Product: The integration of the product into actual user contexts and scenarios is highlighted at this level. It entails modifying features in accordance with user requirements, tailoring the IoT solution to particular use cases or industries, and making sure that it integrates seamlessly with current workflows or systems.

Machine-to-Machine(M2M):

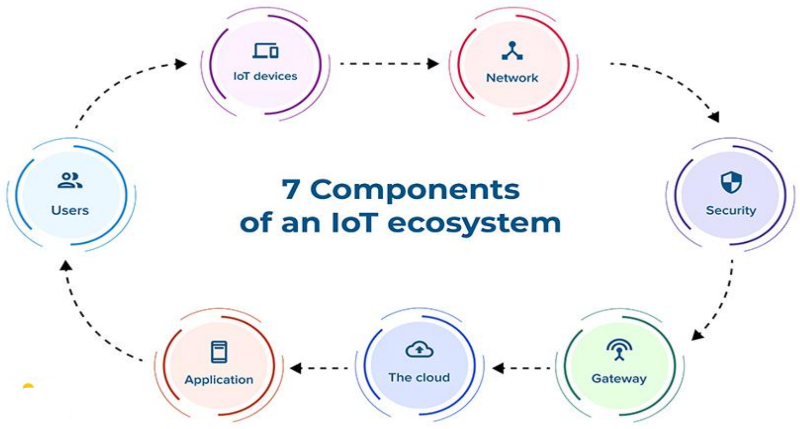
M2M allows machines to communicate with each other without human intervention. M2M integrates the network used by connected objects, servers and applications developed to control them. M2M technology is direct communication between one or more devices using wired or wireless communication channels. These devices capture data, share it, creating a network of smart objects. M2M is considered a subset of the Internet of Things world.

Ex- Controlling the heating of your home using your smartphone is an example of an M2M application. The thermostat and your phone are two interacting machines.

Feature	IoT	M2M
Intelligence	Devices have objects that are responsible for decision making	Some degree of intelligence is observed in this.
Connection type used	The connection is via Network and using various communication types.	The connection is a point to point
Communication protocol used	Internet protocols are used such as HTTP, FTP, and Telnet.	Traditional protocols and communication technology techniques are used

Data Sharing	Data is shared between other applications that are used to improve the end-user experience.	Data is shared with only the communicating parties.
Internet	Internet connection is required for communication	Devices are not dependent on the Internet.
Type of Communication	It supports cloud communication	It supports point-to-point communication.
Computer System	Involves the usage of both Hardware and Software.	Mostly hardware-based technology
Scope	A large number of devices yet scope is large.	Limited Scope for devices.
Business Type used	Business 2 Business(B2B) and Business 2 Consumer(B2C)	Business 2 Business (B2B)
Open API support	Supports Open API integrations.	There is no support for Open APIs
It requires	Generic commodity devices.	Specialized device solutions.
Centric	Information and service centric	Communication and device centric.
Approach used	Horizontal enabler approach	Vertical system solution approach. .
Components	Devices/sensors, connectivity, data processing, user interface	Device, area networks, gateway, Application server.
Examples	Smart wearables, Big Data and Cloud, etc.	Sensors, Data and Information, etc.

Components of IoT:



1. Users:
At the front of the IoT ecosystem are its users. They engage with gadgets, user interfaces, and applications, impacting how well IoT deployments go. For widespread adoption, it is essential to comprehend user wants and preferences and to make sure they have a great experience.

2. IoT Devices: These are the actual physical items or gadgets that have connection, actuators, and sensors built in to gather and share data. Wearables, industrial machinery, sensors, and smart appliances are a few examples.

3. Connectivity: The IoT ecosystem is based on connectivity. It deals with the several communication technologies that make it possible for gadgets to connect to the internet and exchange messages with one another. This can include wireless technologies like Bluetooth, Wi-Fi, and cellular networks, as well as wired technologies like Ethernet and low-power wide-area networks (LPWAN).

4. Data Processing: After devices gather data, it must be processed in order to yield valuable insights. Analyzing and interpreting data is part of data processing, which frequently makes use of edge or cloud computing. This part makes sure the information becomes useful and applicable.

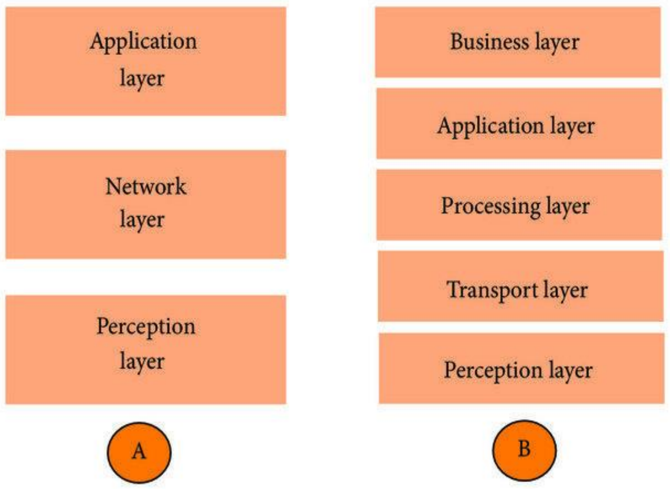
5. Gateways: These serve as a conduit for data between devices and the cloud, gathering information, screening it for security issues, and carrying out preliminary processing before forwarding it. They can also communicate instructions to devices from the cloud.

6. The Cloud: The virtual centers of the Internet of Things, the cloud can store enormous volumes of data, has strong processing capabilities, and offers platforms for service delivery, application development, and data analysis.

7. Applications & Platforms: These are the user interfaces that let people communicate with their Internet of Things (IoT) devices and get the insights that are obtained from the information gathered. This can include specialized industrial control systems, web dashboards, and mobile apps.

IoT Architecture:

IoT architecture refers to the tangle of components such as sensors, actuators, cloud services, Protocols, and layers that make up IoT networking systems. In general, IoT architecture is divided into layers that allow administrators to evaluate, monitor, and maintain the integrity of the system.



The IoT (Internet of Things) architecture outlines how multiple components of an IoT system interact and collaborate to provide seamless communication, data processing, and functionality. Usually, the architecture consists of multiple levels, each with distinct functions

1. Perception: The sensors themselves are on this layer. This is where the data comes from. The data could be gathered from any number of sensors on the connected device. Actuators, which act on their environment, are also at this layer of the architecture.

2. Network: The network layer describes how large amounts of data are moving throughout the application. This layer connects the various devices and sends the data to the appropriate back-end services.

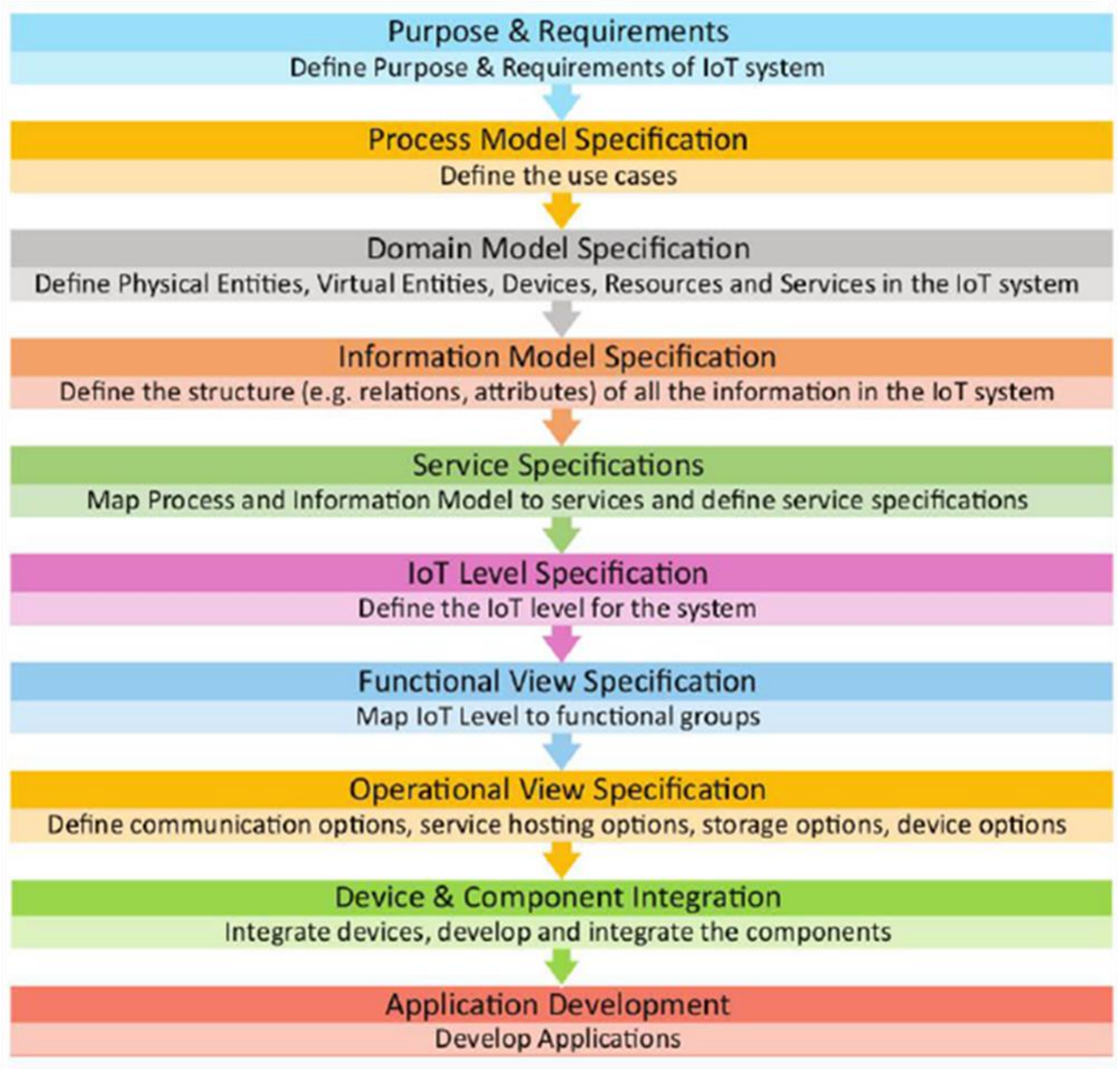
3. Application: The application layer is what the users see. This could be an application to control a device in a smart-home ecosystem, or a dashboard showing the status of the devices which are part of a system.

4. Transport: This layer describes the transfer of data between the sensors in the Perception layer and the Processing layer through various networks.

5. Processing: Sometimes referred to as the Middleware layer, this one stores, analyzes, and pre-processes the data coming from the Transport layer. In modern software applications, this is often located on the edge of the cloud for low latency communications.

6. Business: This layer is often referred to as the Business Intelligence layer. Located at a higher level than the Application layer, the Business layer describes everything that has to do with the stakeholders. Decision-making will be done here based on the data found and consumed at the Application layer.

IoT Design Methodology:


Purpose & Requirements
Define Purpose & Requirements of IoT system
Process Model Specification
Define the use cases
Domain Model Specification
Define Physical Entities, Virtual Entities, Devices, Resources and Services in the IoT system
Information Model Specification
Define the structure (e.g. relations, attributes) of all the information in the IoT system
Service Specifications
Map Process and Information Model to services and define service specifications
IoT Level Specification
Define the IoT level for the system
Functional View Specification
Map IoT Level to functional groups
Operational View Specification
Define communication options, service hosting options, storage options, device options
Device & Component Integration
Integrate devices, develop and integrate the components
Application Development
Develop Applications

Level 1: Purpose & Requirements: Define Purpose & Requirements:
Your IoT system's aims and objectives should be clearly defined. Determine the expectations and needs of the various stakeholders, including users and businesses. Ascertain the limitations and technical requirements.

Level 2: Process Model: Define the Use Cases:
Indicate which scenarios your IoT system will be utilized for. Determine who the users, devices, and systems are in each use case. Explain the procedures in each use scenario.

Level 3: Domain Model: Define Physical Entities, Virtual Entities, Devices, Resources and Services in the IoT system:
Determine which physical systems and items (machines, actuators, sensors, etc.) will be a component of your Internet of Things system. Describe any virtual objects (data models, simulations) that are representations of real-world concepts or objects. Name the equipment that will be used to gather and send data. Ascertain the resources (processing power, data storage) required by the devices and services. Determine the services (data collecting, analysis, and control) that the system will offer.

Level 4: Information Model: Define the structure (e.g., relations, attributes) of all the information in the IoT system:
Specify the information that the sensors and other devices will gather. Indicate the data's structure and format (temperature, timestamp, location, etc.).Identify the connections between the various data items.

Level 5: Service Specifications: Map Process and Information Model to services and define service specifications:
Ascertain the manner in which the services will manage the identified processes and data. Describe each service's functionality and interfaces. Indicate the ways in which the services will communicate with gadgets and one another.

Level 6: IoT Level Specification: Define the IoT level for the system:
Consider aspects like cost, scalability, and security when determining the right degree of complexity for your IoT system. Device-to-device, device-to-cloud, and edge-based processing are examples of common tiers.

Level 7: Functional View Specification: Map IoT Level to functional groups:
Based on the selected IoT level, group the logical functions of the system. Describe the roles that each functional group plays and how they interact.

Level 8: Operational View Specification: Define communication options, service hosting options, storage options, device options:
Choose the technologies and communication protocols (cellular, Bluetooth, Wi-Fi) for data transmission. Choose the hosting location for the services (on-device, cloud, edge gateway). Select the local or cloud storage option for your data. Choose the hardware and parts that the system will employ.

Level 9: Device & Component Integration: Integrate devices, develop and integrate the components:
Attach the components and devices to the system. Create or include the software components required for every feature.

Level 10: Application Development: Develop Applications:
Develop applications and user interfaces that engage the IoT system and benefit users.

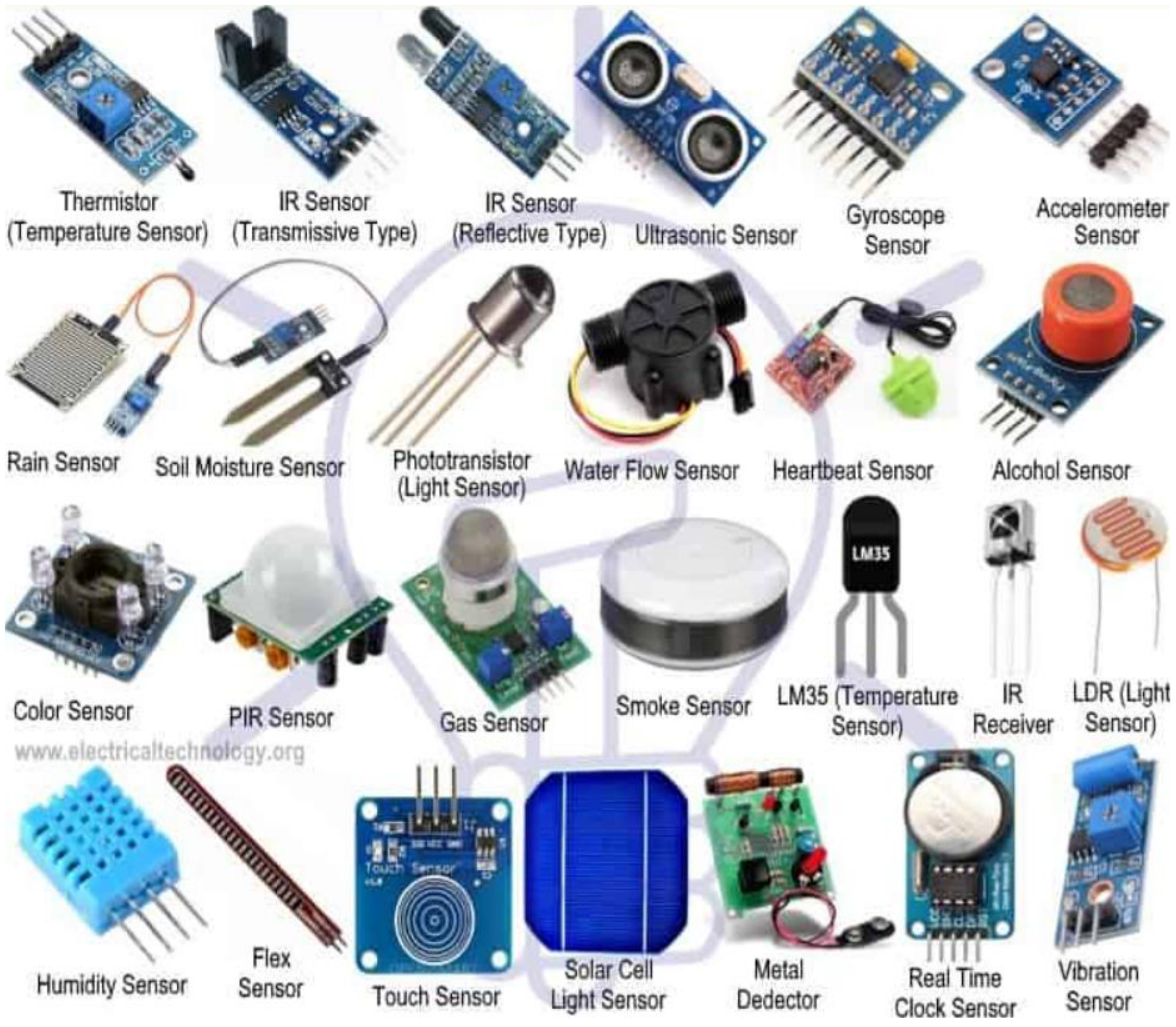
System-in-a-Chip (SOC):

A System-on-a-Chip is a microchip that contains all the components of a computer or other electronic systems on a single integrated circuit (IC). It combines various components such as processor, memory, input/output interfaces, and peripherals on a single chip. This integration provides a cost-effective and power-efficient solution for many electronic devices, especially for IoT and embedded systems.

Printed Circuit Board (PCBs):

PCBs are an essential component of IoT devices. IoT devices require PCBs to connect sensors, actuators, and other components to the internet. PCBs are used in various IoT applications, including smart home devices, industrial automation, healthcare monitoring, and agricultural monitoring.

Types of Sensors:



1. **Infrared Sensor (IR Sensor)**
   The IR sensor is an electronic device that measures and detects infrared radiation in its surrounding environment. Or in simple words, it is a light-emitting diode that can detect the change in color, heat, and IR radiation.

2. **Temperature Sensor**
   Temperature Sensors measure the amount of heat energy or even coldness that is generated by an object or system, allowing us to “sense” or detect any physical change to that temperature producing either an analog or digital output.

3. **Proximity Sensor**
   A proximity sensor is a sensor able to detect the presence of nearby objects without any physical contact. A proximity sensor often emits an electromagnetic field or a beam of electromagnetic radiation (infrared, for instance), and looks for changes in the field or return signal. The object being sensed is often referred to as the proximity sensor's target. Different proximity sensor targets demand different sensors.

4. **Ultrasonic Sensor**
   An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves and converts the reflected sound into an electrical signal.

5. **Accelerometers**
   An accelerometer is a tool that measures proper acceleration. Proper acceleration is the acceleration (the rate of change of velocity) of a body in its own instantaneous rest frame; this is different from coordinate acceleration, which is acceleration in a fixed coordinate system.

6. **Gyroscope Sensor**
   A gyroscope sensor is a device that can measure and maintain the orientation and angular velocity of an object. These are more advanced than accelerometers. These can measure the tilt and lateral orientation of the object

7. **Pressure Sensor**
   A pressure sensor is a device for pressure measurement of gases or liquids. The pressure is an expression of the force required to stop a fluid from expanding and is usually stated in terms of force per unit area. A pressure sensor usually acts as a transducer; it generates a signal as a function of the pressure imposed

8. **Hall Effect Sensor**
   A Hall effect sensor is a device that is used to measure the magnitude of a magnetic field. Its output voltage is directly proportional to the magnetic field strength through it. Hall effect sensors are used for proximity sensing, positioning, speed detection, and current sensing applications.

9. **Load cell**
   A load cell is a type of transducer, specifically a force transducer. It converts a force such as tension, compression, pressure, or torque into an electrical signal that can be measured and standardized. As the force applied to the load cell increases, the electrical signal changes proportionally.

10. **Light Sensor**
    The light sensor is a passive device that converts this “light energy” whether visible or in the infra-red parts of the spectrum into an electrical signal output. Light sensors are more commonly known as “Photoelectric Devices” or “Photo Sensors” because the convert light energy (photons) into electricity (electrons).

11. **Color Sensor**
    A color sensor detects the color of the material. This sensor usually detects color in the RBG scale. This sensor can categorize the color as red, blue, or green. These sensors are also equipped with filters to reject the unwanted IR light and UV light.

12. **Touch Sensor**
    Touch Sensors are the electronic sensors that can detect touch. They operate as a switch when touched. These sensors are used in lamps, touch screens of the mobile, etc… Touch sensors offer an intuitive user interface.

13. **Tilt Sensor**
    A tilt sensor is an instrument that is used for measuring the change in tilt and monitoring of inclination and vertical rotation in vertical structures. The tilt sensor produces an electrical signal which is proportional to the degree of tilt in multiple axes (Uniaxial & Biaxial).

14. **Vibration Sensor**
    The vibration sensor is also called a piezoelectric sensor. These sensors are flexible devices that are used for measuring various processes.

15. **Water Flow Sensor**
    A water flow sensor measures the rate of flow of water and calculates the amount of water flowed through the pipe. The rate of flow of water is measured as liters per hour or cubic meters.

16. **Heartbeat Sensor**
    Heartbeat Sensor is an electronic device that is used to measure the heart rate i.e. speed of the heartbeat.

17. **Level Sensor**
    Level sensors detect the level of liquids and other fluids and fluidized solids, including slurries, granular materials, and powders that exhibit an upper free surface.

18. **Gas Sensor**
    Gas sensors (also known as gas detectors) are electronic devices that detect and identify different types of gasses. They are commonly used to detect toxic or explosive gasses and measure gas concentration

19. **Soil Moisture Sensor**
    The soil moisture sensor is a kind of sensor used to measure the volumetric content of water within the soil.

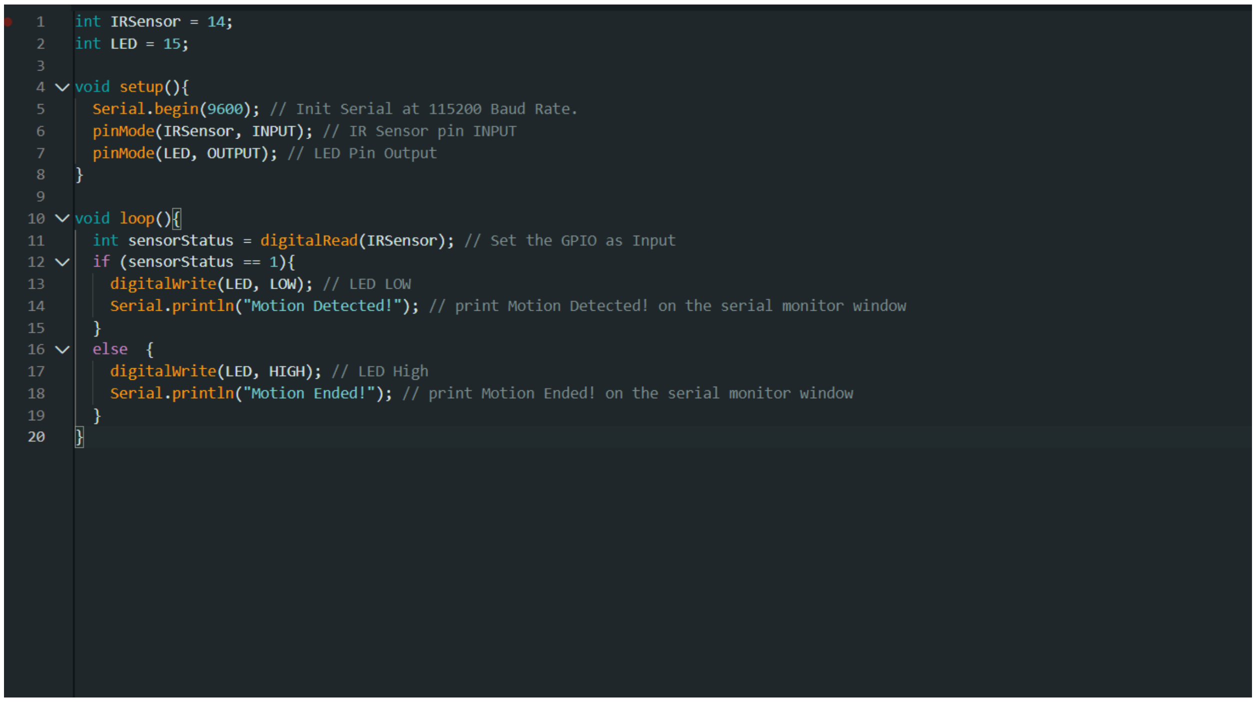
20. **Rotary Encoder**
    A rotary encoder, also called a shaft encoder, is an electro-mechanical device that converts the angular position or motion of a shaft or axle to an analog or digital output signals.

21. **Tachometer**
    A tachometer (revolution-counter, tach, rev-counter, RPM gauge) is an instrument measuring the rotation speed of a shaft or disk, as in a motor or other machine. The device usually displays the revolutions per minute (RPM) on a calibrated analog dial, but digital displays are increasingly common.

Sensor Code:

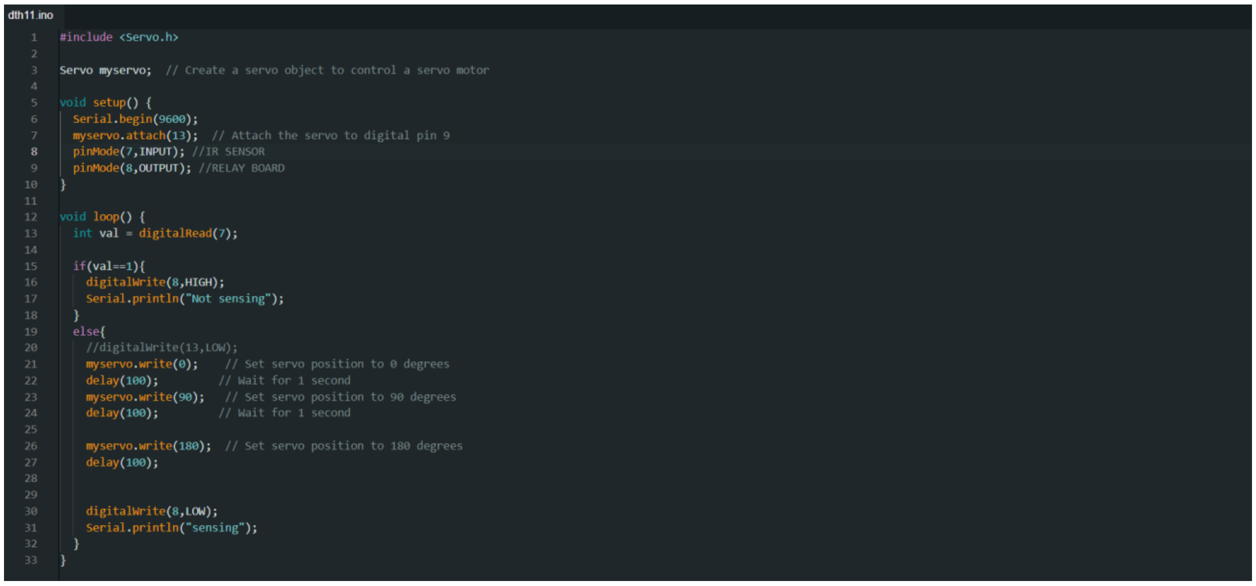
```
const int trigPin = 9;
const int echoPin = 10;
float duration, distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = (duration*.0343)/2;
  Serial.print("Distance: ");
  Serial.println(distance);
  delay(100);
}
```

Code for IR Sensor and LED:



```
int IRSensor = 14;
int LED = 15;

void setup(){
  Serial.begin(9600); // Init Serial at 115200 Baud Rate.
  pinMode(IRSensor, INPUT); // IR Sensor pin INPUT
  pinMode(LED, OUTPUT); // LED Pin Output
}

void loop(){
  int sensorStatus = digitalRead(IRSensor); // Set the GPIO as Input
  if (sensorStatus == 1){
    digitalWrite(LED, LOW); // LED LOW
    Serial.println("Motion Detected!"); // print Motion Detected! on the serial monitor window
  }
  else  {
    digitalWrite(LED, HIGH); // LED High
    Serial.println("Motion Ended!"); // print Motion Ended! on the serial monitor window
  }
}
```

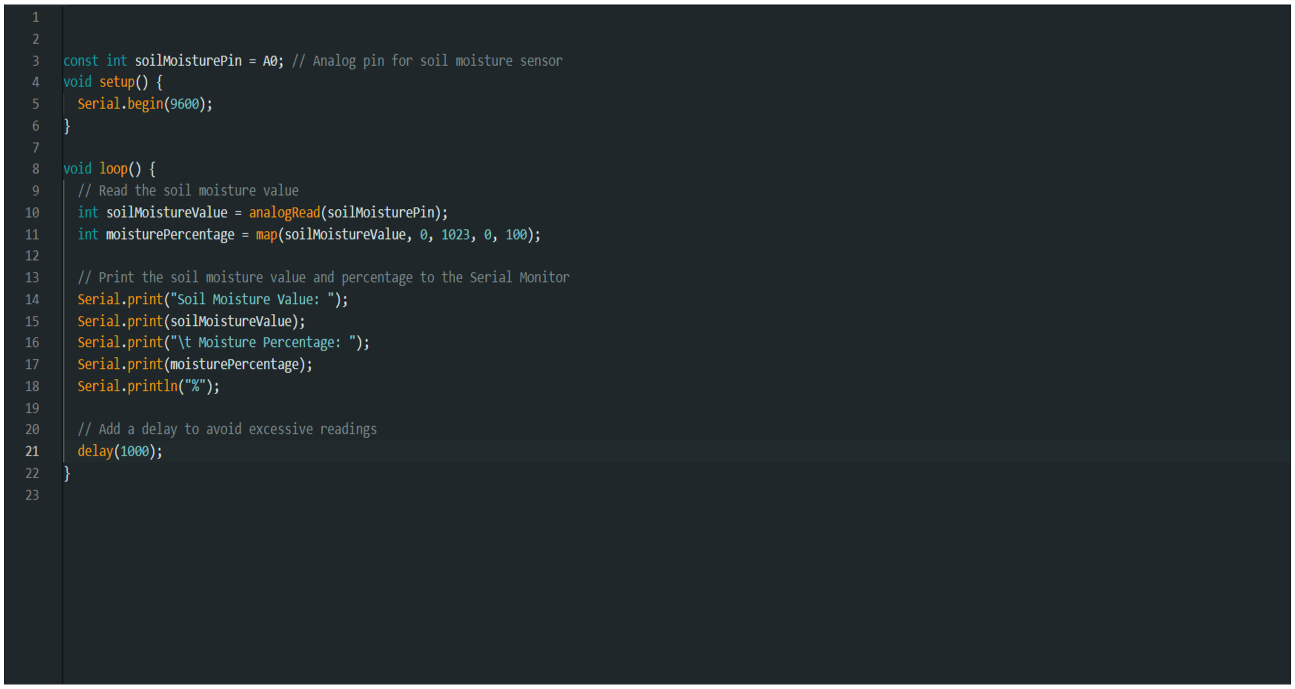
Code for control a servo motor based on the input from an infrared (IR):



dth11.ino

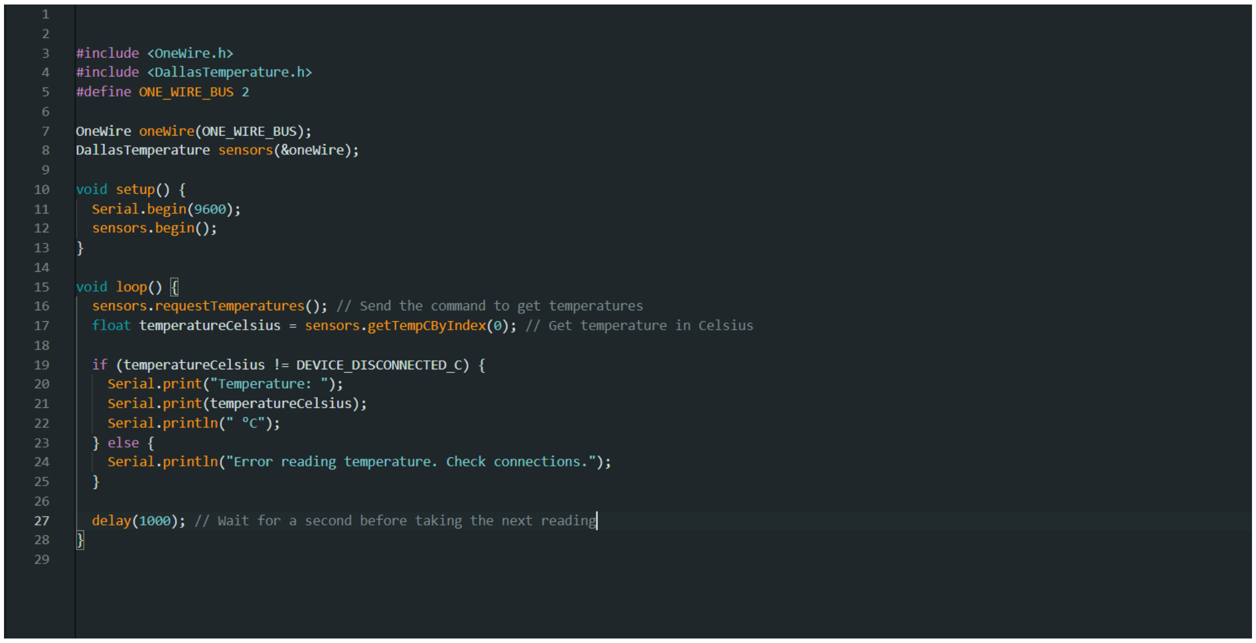
```
#include <Servo.h>

Servo myservo;  // Create a servo object to control a servo motor

void setup() {
  Serial.begin(9600);
  myservo.attach(13);  // Attach the servo to digital pin 9
  pinMode(7,INPUT); //IR SENSOR
  pinMode(8,OUTPUT); //RELAY BOARD
}

void loop() {
  int val = digitalRead(7);

  if(val==1){
    digitalWrite(8,HIGH);
    Serial.println("Not sensing");
  }
  else{
    //digitalWrite(13,LOW);
    myservo.write(0);    // Set servo position to 0 degrees
    delay(100);          // Wait for 1 second
    myservo.write(90);   // Set servo position to 90 degrees
    delay(100);          // Wait for 1 second

    myservo.write(180);  // Set servo position to 180 degrees
    delay(100);


    digitalWrite(8,LOW);
    Serial.println("sensing");
  }
}
```

Code for Soil Moisture Senser:



```


const int soilMoisturePin = A0; // Analog pin for soil moisture sensor
void setup() {
  Serial.begin(9600);
}

void loop() {
  // Read the soil moisture value
  int soilMoistureValue = analogRead(soilMoisturePin);
  int moisturePercentage = map(soilMoistureValue, 0, 1023, 0, 100);

  // Print the soil moisture value and percentage to the Serial Monitor
  Serial.print("Soil Moisture Value: ");
  Serial.print(soilMoistureValue);
  Serial.print("\t Moisture Percentage: ");
  Serial.print(moisturePercentage);
  Serial.println("%");

  // Add a delay to avoid excessive readings
  delay(1000);
}

```

Code for Temperature Detection:



```


#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(9600);
  sensors.begin();
}

void loop() {
  sensors.requestTemperatures(); // Send the command to get temperatures
  float temperatureCelsius = sensors.getTempCByIndex(0); // Get temperature in Celsius

  if (temperatureCelsius != DEVICE_DISCONNECTED_C) {
    Serial.print("Temperature: ");
    Serial.print(temperatureCelsius);
    Serial.println(" °C");
  } else {
    Serial.println("Error reading temperature. Check connections.");
  }

  delay(1000); // Wait for a second before taking the next reading
}

```

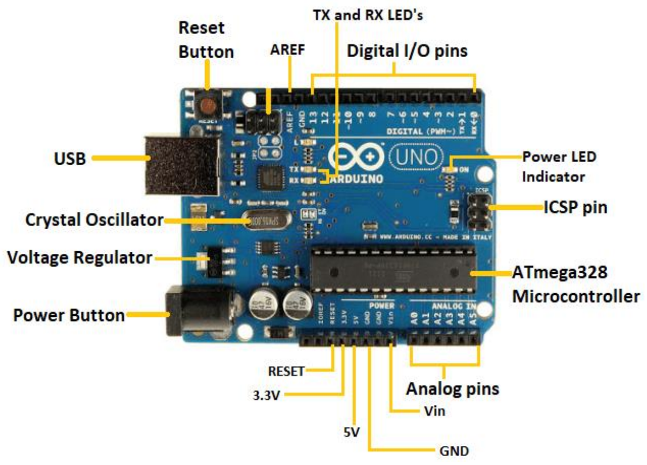
Arduino

A well-liked resource for electronics education and experimentation is Arduino. It combines user-friendly software and hardware to let you control a variety of electronic parts and make interactive projects.

Arduino Setup:

1. Download Arduino IDE:

- Go to the official Arduino website at https://www.arduino.cc/en/software.
- Click on the "Download" button to download the latest version of the Arduino IDE for Windows.



2. Run the Installer:

- When the download is complete, locate the downloaded file named something like `arduino-1.x.x-windows.exe`.
- Double-click the downloaded file to run the installer.

3. Install Arduino IDE:

- Follow the installation wizard prompts:
- Select your language and click "OK".
- Choose the installation directory (the default is usually fine) and click "Next".
- Select the components you want to install (usually, you can leave all options checked) and click "Next".
- Choose whether to create desktop shortcuts and associate file extensions (recommended) and click "Next".
- Click "Install" to begin the installation process.

4. Driver Installation (if needed):

- During the installation process, you may be prompted to install drivers for Arduino-compatible boards (such as Arduino Uno). Follow the on-screen instructions to install these drivers.

5. Finish Installation:

- Once the installation is complete, click "Close" to exit the installer.

6. Launch Arduino IDE:

- After installation, you can find the Arduino IDE shortcut on your desktop or in the Start menu.
- Double-click the Arduino IDE shortcut to launch the application.

7. Verify Installation:

- Upon launching the Arduino IDE, you should see the main window open with the Arduino logo.
- You can now start using the Arduino IDE to program and upload code to your Arduino boards.

Commands:

Defining variables, reading and writing digital and analog signals, controlling timing, and interacting with linked components are all done through Arduino programming commands. These are a few typical commands:

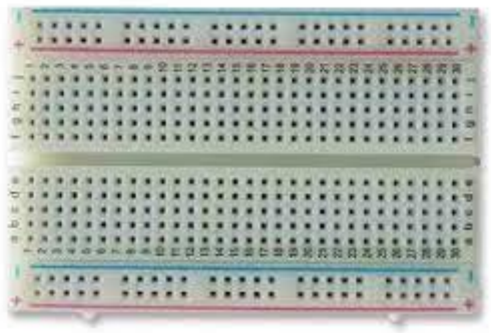
- pinMode(pin, mode): Set a pin as input or output
- digitalWrite(pin, value): Write a HIGH (1) or LOW (0) signal to a digital pin
- digitalRead(pin): Read the value (HIGH/LOW) from a digital pin
- analogRead(pin): Read the analog voltage from an analog pin (0-1023)
- delay(milliseconds): Pause the program for a specified time

Actuators

The key elements that convert digital commands into tangible actions are actuators. They serve as the "muscles" of the system, reacting to controller commands and sensor data to produce actual changes.

1. Breadboard:

A breadboard can be used as a makeshift testing ground for solderless circuit prototyping.
With the use of jumper wires, you may quickly connect electronic components thanks to its rows of metal strips connected underneath.
Breadboards are frequently used for quick prototyping, hobby projects, and electronics education.



2 Servo motor

A servo motor is a rotary actuator that allows for precise control of angular position. It consists of a motor coupled to a sensor for position feedback. It also requires a servo drive to complete the system. The drive uses the feedback sensor to precisely control the rotary position of the motor.

3. Relay boards:

Relay boards are modules that let you use low-power signals to control high-power devices. Relays, electromechanical switches triggered by an electromagnet, are used in it. This enables you to use signals from microcontrollers or other low-power sources to control devices such as lights, motors, and solenoids.

4 LED:

LEDs, or light-emitting diodes, have transformed the electronics industry and are still a shining star in the Internet of Things (IoT) space. These compact, adaptable parts are a desirable option for many applications because they have many advantages over conventional incandescent bulbs.

5. Buzzer:

Electronics and Internet of Things: The Humble Buzzer, Buzzing with Potential Even though they are frequently buried by more visually appealing parts, buzzers are an essential part of the Internet of Things and electronics symphony. These straightforward audios signaling devices bring crucial functionality and user input to a variety of applications by producing audible alerts and tones.

6. Motors:

The IoT's muscles are its motors, which permit automation, movement, and interaction. They enable IoT devices to function in the actual world, from controlling robots to opening blinds. Variations in kinds provide flexibility: DC for easy jobs, stepper for fine control, and servo for both.

MQTT (Message Queuing Telemetry Transport)

The lightweight publish-subscribe messaging protocol known as MQTT, or Message Queuing Telemetry Transport, is extensively utilized in machine-to-machine (M2M) and Internet of Things (IoT) communication. It is perfect for applications where efficiency and low power consumption are essential because it is made for devices and networks with limited resources.

Important elements:
Clients are programs or devices that can post messages or subscribe to them. Cloud platforms, cellphones, actuators, and sensors are a few examples.
1. Broker: A central server that distributes messages to the right subscribers after receiving them from publishers.
2. Topics: Message classification using hierarchical channels. Customers subscribe to particular subjects in order to get pertinent information.
3. QoS (Quality of Service): Levels (0, 1, 2) that balance assurance and speed to define how consistently communications are delivered.

How it functions:
Publish: A client notifies the broker of a message's content and subject.
Route: The broker routes the message by finding subscribers that are interested in the topic.
Receive: The published message is received by subscribing clients, who may then take appropriate action in response to its contents.

Advantages of MQTT

Lightweight: Perfect for devices with limited resources, it has a small footprint and uses little bandwidth.
Efficient and uncomplicated: This minimizes development complexity by being simple to comprehend and apply.
Scalable: Suitable for big installations, the broker can manage millions of connected devices.
Flexible: Dynamic communication patterns are possible using the publish-subscribe approach.
Reliable: Depending on your requirements, QoS settings guarantee message delivery.

Applications of MQTT:

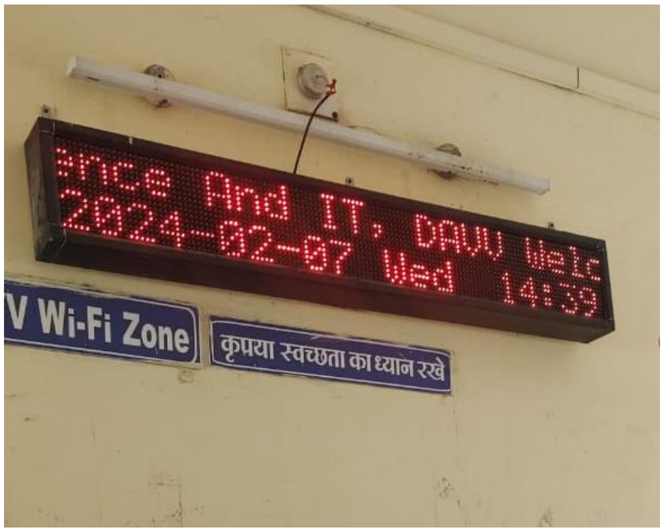
1. Sensor data collection: Provide cloud platforms with sensor readings from devices so they may be watched over and analyzed. Automate your house by allowing gadgets to communicate with one another so that actions can be coordinated.
2. Industrial automation: Establish connections between machinery and equipment to facilitate control and data transmission.
3. Vehicle-to-everything (V2X) communication: Facilitate communication for traffic control and safety between automobiles and infrastructure. MQTT applications:
4. Sensor data collection: Provide cloud platforms with sensor readings from devices so they may be watched over and analyzed.
5. Remote control: Use broadcast signals to remotely operate robots, appliances, or other objects.
6. Automate your house by allowing gadgets to communicate with one another so that actions can be coordinated.
7. Industrial automation: Establish connections between machinery and equipment to facilitate control and data transmission.
8. Vehicle-to-everything (V2X) communication: Facilitate communication for traffic control and safety between automobiles and infrastructure.
9. Remote control: Use broadcast signals to remotely operate robots, appliances, or other objects.

Visit for Project:

We visited SCSIT DAVV, we had an in-depth discussion about the components and workings of a digital LED display. We explored the P4 RGB and NodeMCU + PS266 components, and the functioning of P10 modules. We also discussed the data transfer rate, which is more than 400kbps.

We delved into the Network Time Protocol (NTP) server and MQTT (Message Queuing Telemetry Transport), a publish-subscribe-based messaging protocol. We learned how these technologies enable us to display time on the digital LED display.

We also discussed various functions such as getting formatted time, hours, minutes, and day. The MQTT protocol's publish and subscribe mechanism was another key topic of our discussion.



Finally, we learned about the DmD library, which supports different languages. Overall, it was an enlightening session that provided us with a deeper understanding of the intricacies of a digital LED display.

Our first mini-project was a collaborative effort between Team Beta and Team Alpha. We were provided with a variety of IoT sensors, including the DHT11, jumper wires, Arduino, ultrasonic sensor, breadboard, LED light, and buzzer. Tasked with developing a useful application for these components, we decided to create a car accident emergency alert system.

This small prototype uses the sensors to detect a potential car accident and triggers an emergency alert. The Arduino microcontroller serves as the brain of the system, processing sensor data and controlling the output devices. The ultrasonic sensor detects sudden changes in distance, the LED light provides a visual alert, and the buzzer produces an audible alarm.

The successful development and functioning of this system, powered by Arduino code, placed our group at the forefront of the competition. This project not only showcased our technical skills but also highlighted the potential of IoT in enhancing safety measures.

IFTTT (If This Then That)

"If This Then That" Automation - One "If This Then That" at a Time
If This Then That, or IFTTT (pronounced "if-ttt"), is a platform that enables you to automate actions between different web services and devices. IFTTT may make your dreams come true, such as having your smart lamps turn on automatically at dusk or receiving a signal when your favorite musician drops a new song.

Advantages of IFTTT usage:

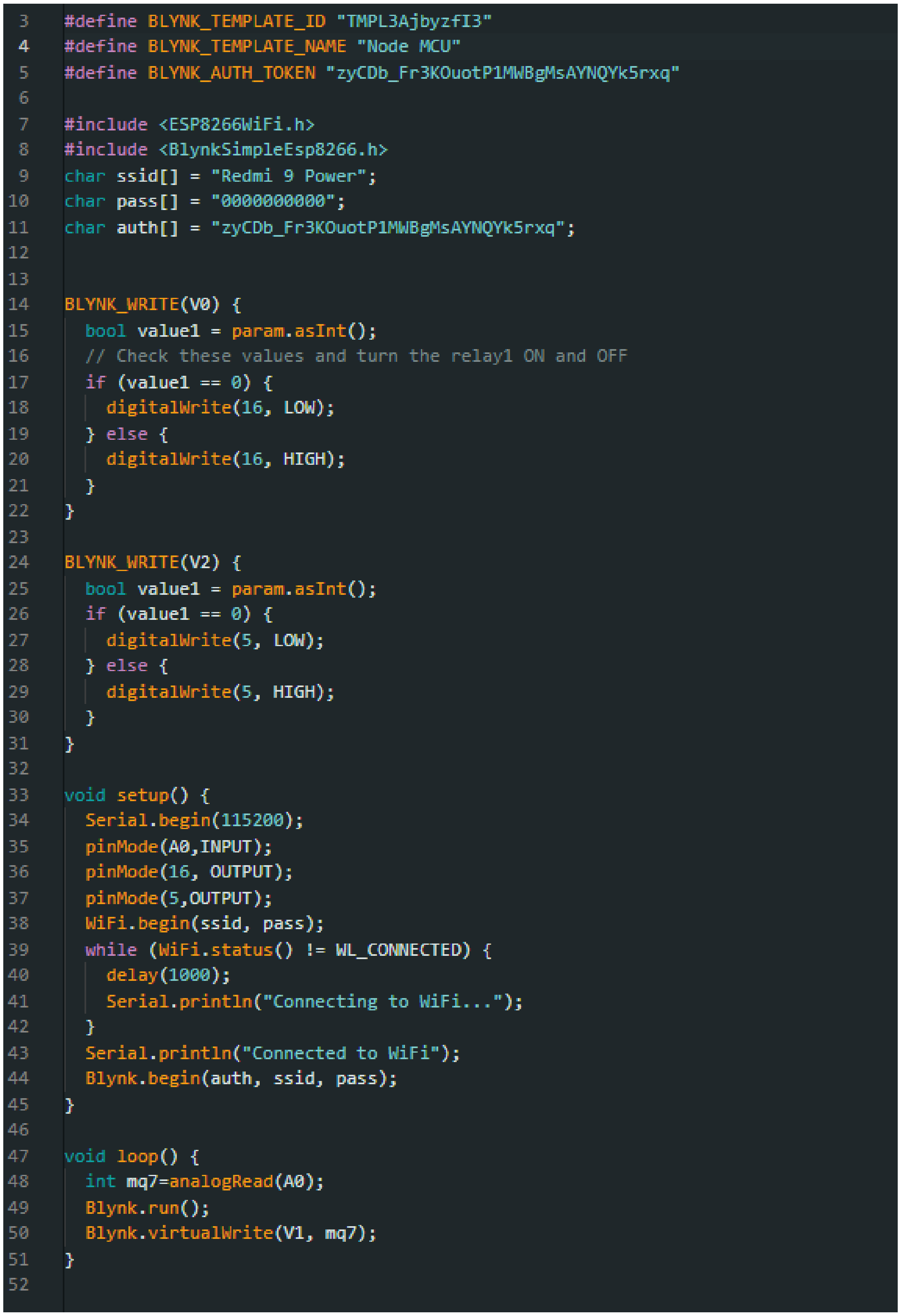
- Easy automation: It requires no coding, so anyone may use it.
- Numerous integrations: Hundreds of channels provide a variety of options.
- Time-saving: Automate monotonous jobs to boost productivity.
- Enhanced convenience: Establish guidelines to streamline your everyday schedule.
- Flexibility: Make personalized applets to meet your unique requirements.

INTRODUCTION TO BLYNK IOT

Blynk is an IoT platform that enables users to control and monitor connected devices remotely via a smartphone app. It provides a user-friendly interface for creating custom dashboards and controlling hardware with minimal coding. Blynk supports a wide range of microcontrollers and connectivity options, making it accessible for hobbyists and professionals alike to build IoT projects easily.

To log in to the Blynk IoT platform and set up with NodeMCU, follow these steps:
1. Download the Blynk app from the App Store or Google Play Store.
2. Create a Blynk account or log in if you already have one.
3. Once logged in, create a new project by tapping the "+" button.
4. Choose the device you'll be using (NodeMCU) and select the connection type (Wi-Fi).
5. You'll receive an authentication token via email. Copy this token.
6. In your Arduino IDE, install the Blynk library.
7. Open the "NodeMCU.ino" example sketch from the Blynk library.
8. Paste your authentication token into the sketch.
9. Set up your NodeMCU board and connect it to your computer via USB.
10. Select the correct board and port in the Arduino IDE.
11. Upload the sketch to your NodeMCU.
12. Once uploaded, power up your NodeMCU.
13. In the Blynk app, tap the play button to start the project.
14. Your NodeMCU should now be connected to the Blynk app.
15. Customize the app interface and add widgets to control your NodeMCU remotely.

Code:



```
#define BLYNK_TEMPLATE_ID "TMPL3AjbyzfI3"
#define BLYNK_TEMPLATE_NAME "Node MCU"
#define BLYNK_AUTH_TOKEN "zyCDb_Fr3KOuotP1MWBgMsAYNQYk5rxq"

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
char ssid[] = "Redmi 9 Power";
char pass[] = "0000000000";
char auth[] = "zyCDb_Fr3KOuotP1MWBgMsAYNQYk5rxq";


BLYNK_WRITE(V0) {
  bool value1 = param.asInt();
  // Check these values and turn the relay1 ON and OFF
  if (value1 == 0) {
    digitalWrite(16, LOW);
  } else {
    digitalWrite(16, HIGH);
  }
}

BLYNK_WRITE(V2) {
  bool value1 = param.asInt();
  if (value1 == 0) {
    digitalWrite(5, LOW);
  } else {
    digitalWrite(5, HIGH);
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(A0,INPUT);
  pinMode(16, OUTPUT);
  pinMode(5,OUTPUT);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  Blynk.begin(auth, ssid, pass);
}

void loop() {
  int mq7=analogRead(A0);
  Blynk.run();
  Blynk.virtualWrite(V1, mq7);
}

```

BLYNK IOT BASED PRACTICE PROJECT

Project : Smart Godown Monitoring System

The Smart Godown Monitoring System is an IoT-based project designed to enhance the efficiency and management of a warehouse or storage facility, specifically targeting the storage of vegetables to prevent spoilage. The system utilizes various sensors and the Blynk platform to monitor environmental conditions and detect potential issues in real-time.

Key Features:

Temperature and Humidity Monitoring:

1. Utilizes a DHT11 sensor to continuously monitor the temperature and humidity levels inside the godown.
2. Alerts warehouse managers if the temperature exceeds a set threshold, ensuring optimal conditions for vegetable storage.

Light Intensity Sensing:

1. Incorporates an LDR (Light-Dependent Resistor) to monitor ambient light levels.
2. Helps determine whether the godown is exposed to adequate light conditions, which can impact the quality and shelf life of stored vegetables.

Gas Detection:

1. Integrates a gas sensor to detect the presence of harmful gases.
2. Raises alarms if gas levels exceed predefined safety thresholds, ensuring a secure storage environment.

Remote Control:

1. Enables remote control of specific functions such as lighting and ventilation through the Blynk mobile application.
2. Offers the ability to turn on or off devices based on real-time sensor data.

Blynk Platform Integration:

1. Utilizes the Blynk platform for real-time monitoring and control of the godown.
2. Provides a user-friendly interface for visualizing sensor data and receiving alerts.

The Smart Godown Monitoring System aims to improve the quality of stored vegetables, reduce spoilage, and enhance operational efficiency by providing timely information and control capabilities to warehouse managers. This project aligns with the growing trend of implementing IoT solutions for smart and sustainable agricultural practices.

Logic: The underlying idea is to prevent the vegetables stored in the warehouse from spoiling.

CODE:

```
#define BLYNK_TEMPLATE_ID "TMPL3WUnT2fgo"
#define BLYNK_TEMPLATE_NAME "godown"
#define BLYNK_AUTH_TOKEN "FXm5QoExF0JFHkAaLp3pnIgJPlQ06o2D"
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT11.h>
char ssid[] = "realme";
char pass[] = "12345678";
char auth[] = "FXm5QoExF0JFHkAaLp3pnIgJPlQ06o2D";
const int ledPin = 0;   //the number of the LED pin
const int  ldrPin = 5;  //the number of the LDR pin
DHT11 dht11(16);//digital
int sensorPin = A0; // Analog pin connected to the sensor gas
int threshold = 80; // Adjust this threshold value based on your sensor calibration

BLYNK_WRITE(V4) {
  bool value1 = param.asInt();
  if (value1 == 0) {
    digitalWrite(12, LOW);
   } else {
    digitalWrite(12, HIGH);
  }
}

void setup() {
 Serial.begin(9600);
  pinMode(ledPin, OUTPUT);  //initialize the LED pin as an output
  pinMode(ldrPin,  INPUT);   //initialize the LDR pin as an input
  pinMode(15,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
 WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
      delay(1000);
      Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  Blynk.begin(auth, ssid, pass);
}
```

```
void loop() {
  int  ldrStatus = digitalRead(ldrPin);
Serial.println(ldrStatus);
 if (ldrStatus== 0) {
      digitalWrite(ledPin,  HIGH);
     digitalWrite(13,  HIGH);          //turn LED on
}
else {
   digitalWrite(ledPin,  LOW);
    digitalWrite(13,  LOW);
  }
  int temperature = 0;
  int humidity = 0;
  int result = dht11.readTemperatureHumidity(temperature, humidity);

    if (result == 0) {
       Serial.print("Temperature: ");
       Serial.print(temperature);
       Serial.print(" °C\tHumidity: ");
       Serial.print(humidity);
       Serial.println(" %");
       }
 else {
       // Print error message based on the error code.
       Serial.println(DHT11::getErrorString(result));
      }
    if(temperature>=35 && humidity<=60 ){
      digitalWrite(4,HIGH);
   }
    else{
      digitalWrite(4,LOW);
   }
 int sensorValue = analogRead(sensorPin);

  Serial.print("Sensor Value: ");
  Serial.println(sensorValue);

  if (sensorValue > threshold) {
    Serial.println("Gas detected!");
    digitalWrite(15,HIGH);
  } else {
    Serial.println("No gas detected.");
    digitalWrite(15,LOW);
  }

  Blynk.run();
  Blynk.virtualWrite(V0, temperature);
  Blynk.virtualWrite(V1, ldrStatus);
  Blynk.virtualWrite(V2,sensorValue);
  Blynk.virtualWrite(V3, humidity);
 delay(1000); // Delay for readability, adjust as needed
}
```

Final Project:

NeXdrive: A Multi-Mode, Voice-Controlled, Obstacle-Avoiding Automatic Car

Team THETA Presents
NeXdrive
A Voice-Guided, Auto Obstacle-Aware car
REMOTLY CONTROLLED
INTUITIVE VOICE CONTROL
LANE DEPARTURE WARNING
ADVANCED OBJECT DETECTION
AUTOMATIC DRIVING CAPABILITY

NeXdrive is an innovative and versatile remote-controlled car, offering a range of features to enhance user experience and functionality. NeXdrive has two modes in Automatic mode it needs No Human interaction and in Mannual mode it is controlled through app or voice commands.

NeXdrive features an automatic mode activated by a dedicated button. In this mode, the car autonomously navigates its path, detecting obstacles using the IR sensor and adjusting its course accordingly. This autonomous behavior adds convenience for users who wish to set NeXdrive on a predetermined path without constant manual control.

Additionally, The car is controlled through a mobile app, providing manual control over eight directions, adjustable speed, light control, and a built-in horn. Notably, the app incorporates voice-to-text capabilities, allowing users to control NeXdrive through voice commands.

The car is equipped with an Infrared (IR) sensor for object detection, ensuring it does not move forward if obstacles are detected in its path. This safety feature prevents collisions and adds an extra layer of control during manual operation.

Funtions:

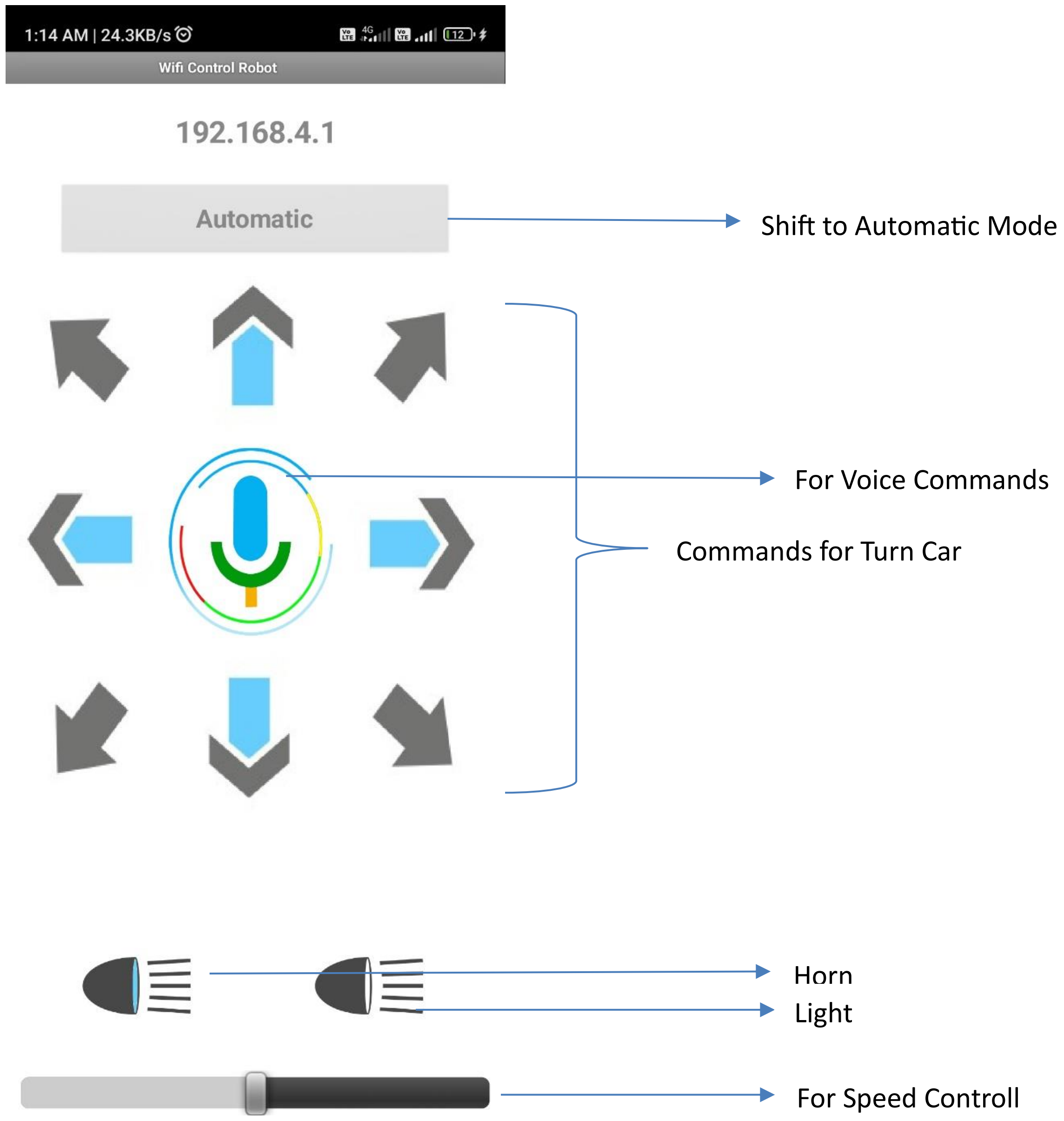
- **Remote control:** Operate it from your mobile app (MIT app) in 8 directions, control speed, turn on lights and sound the horn.
- **Voice control:** Use voice commands through the app's text-to-speech feature for hands-free control.
- **Manual mode:** Take full command and navigate freely.
  Automatic mode: Relax while NexDrive autonomously navigates, detecting obstacles with IR sensors and changing direction as needed.
- **Night vision:** Light sensor automatically activates headlights for safe maneuvering.
- **Additional features:** Buzzer for horn, LED for visual indications.

Benefits:

- **Versatile control:** Choose between manual and automatic modes for different situations.
- **Safety features:** Obstacle detection and automated avoidance minimize collisions.
- **Easy to use:** Intuitive app interface and voice control for smooth operation.
- **Night vision:** Ensures safe navigation even in low-light conditions.
- **Fun and engaging:** Perfect for remote-controlled car enthusiasts and tech-savvy individuals.

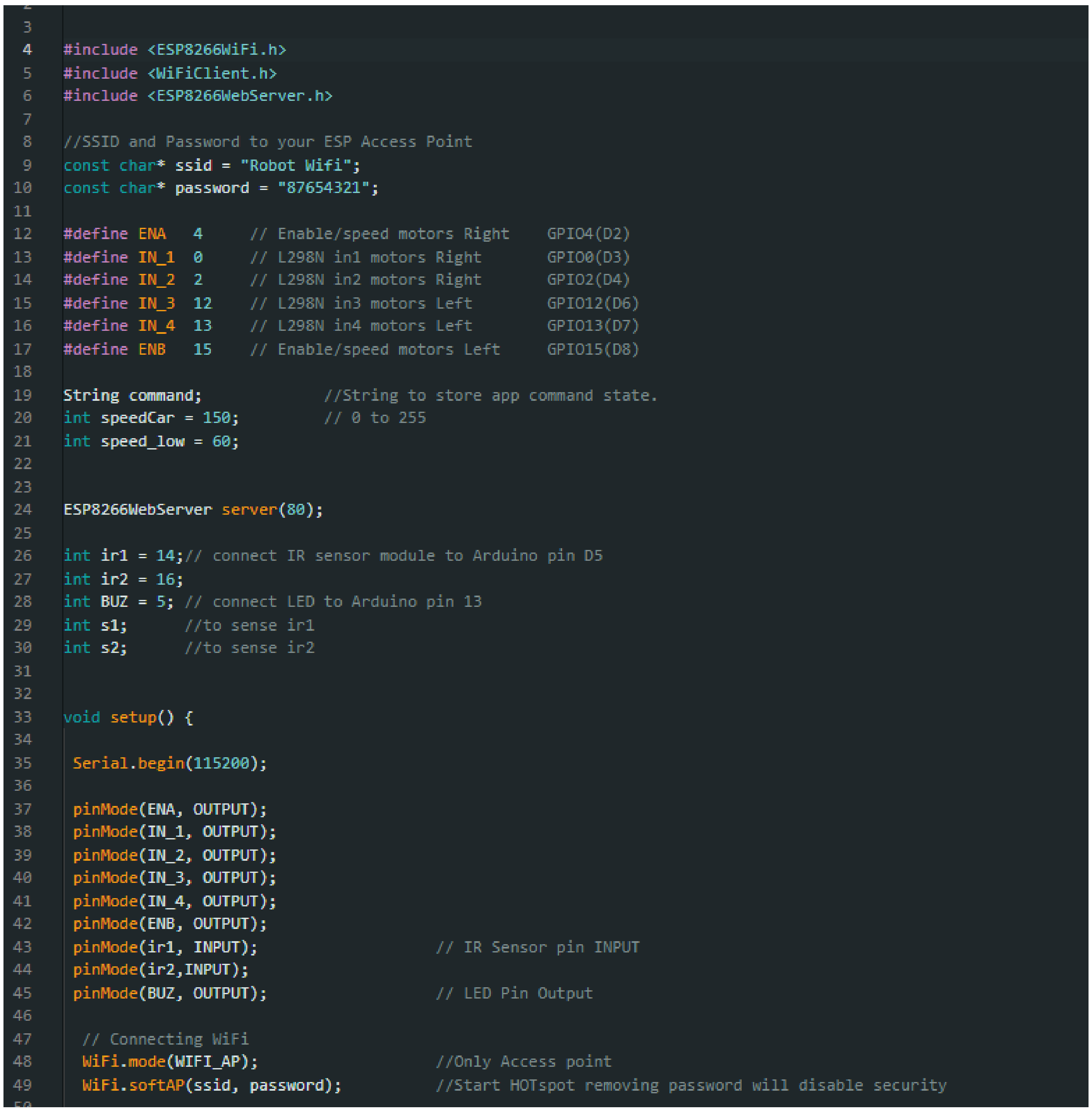
Components of an IoT Car:

- DC Motor: The heart of the car that drives it forward.
- Sensors: Essential for making obstacle avoidance, includes Ultrasonic, LDR & IR Sesnors.
- Microprocessor: Node MCU ESP8266, The brain of the car that controls its actions.
- Battery & Wires: Provides power to the car and transmits signals to the motor and other components.
- Holder: Keeps everything in place and makes the car stable.

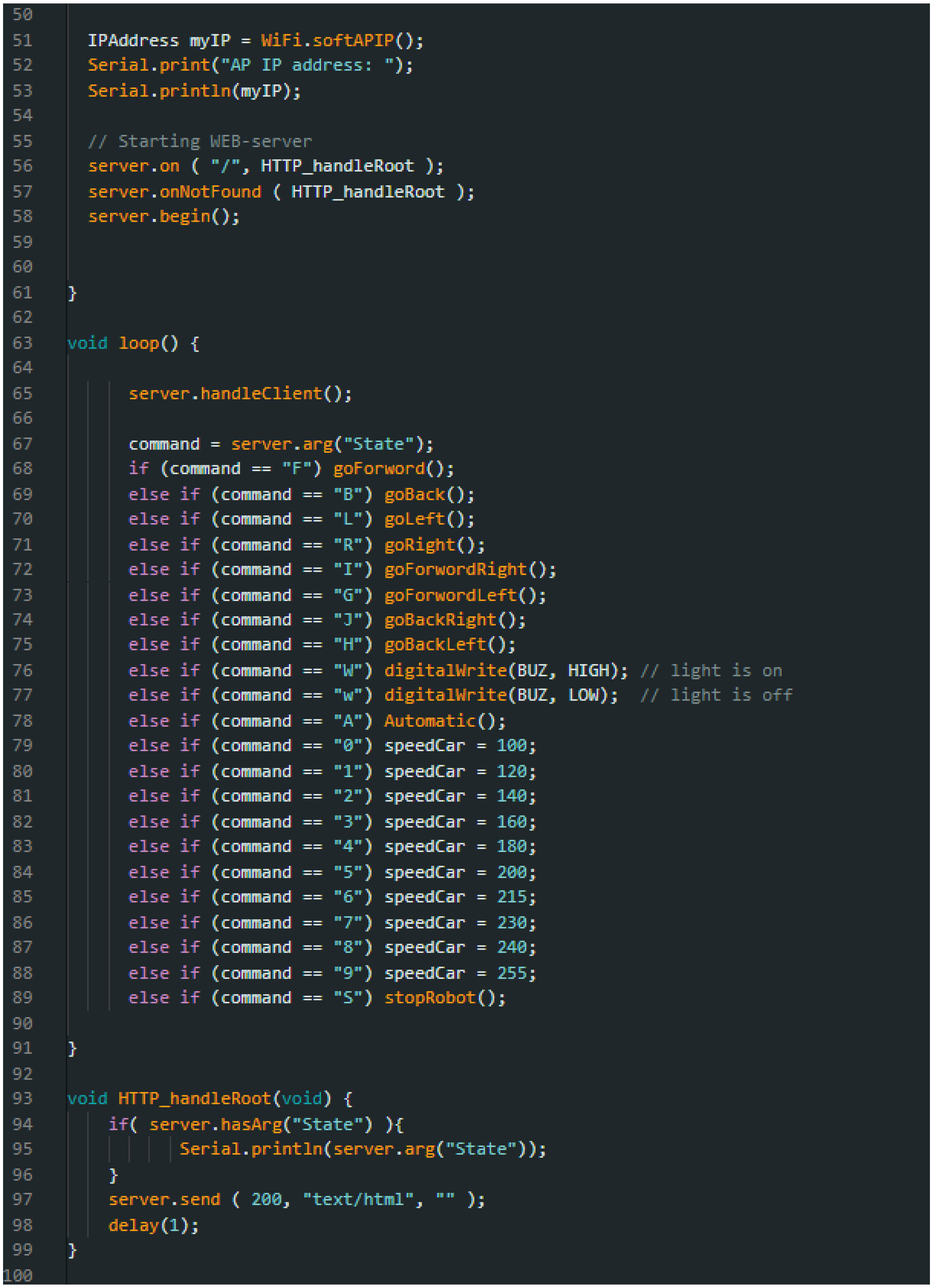


Interface of "NeXdrive" App for Control Car

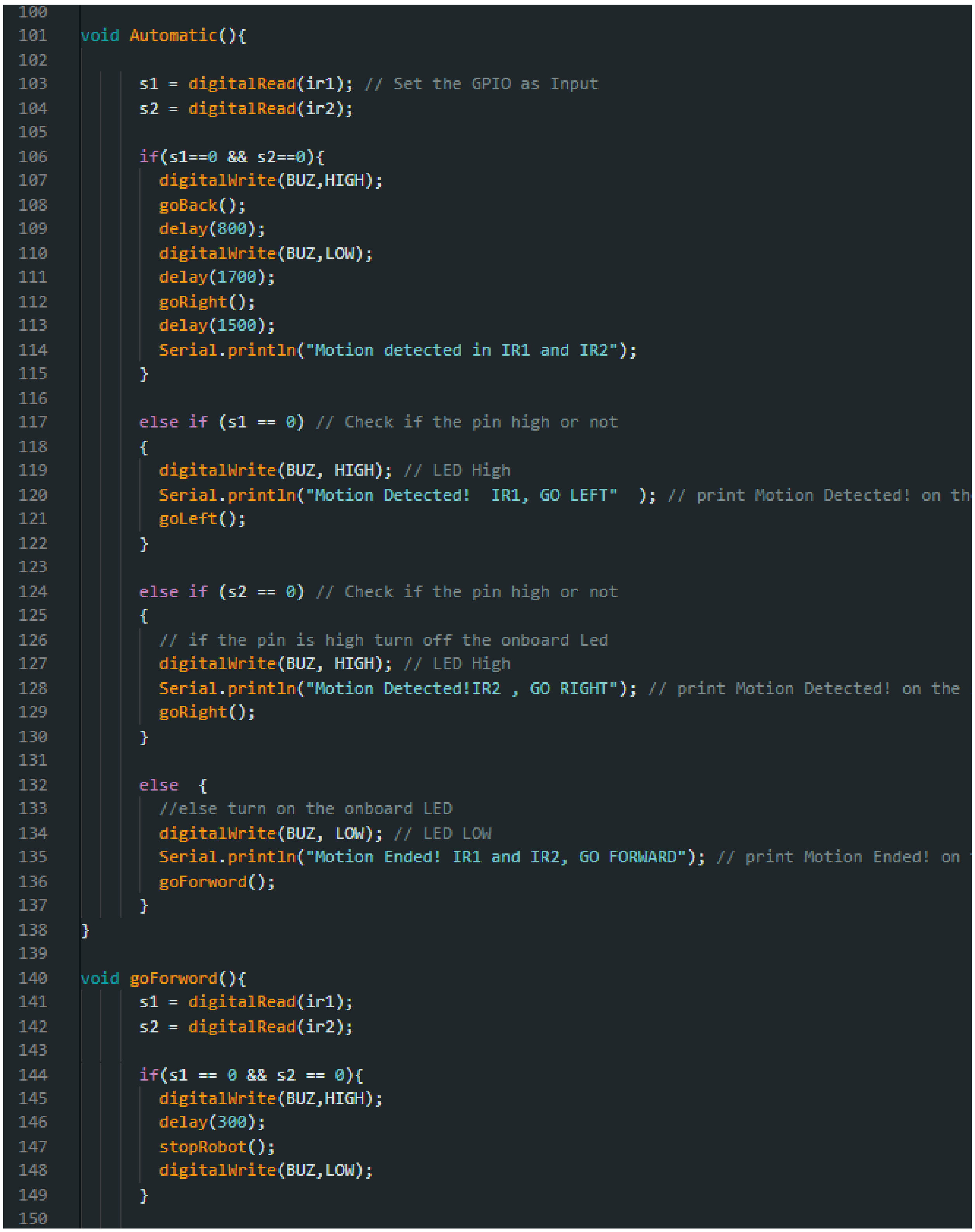
Code:



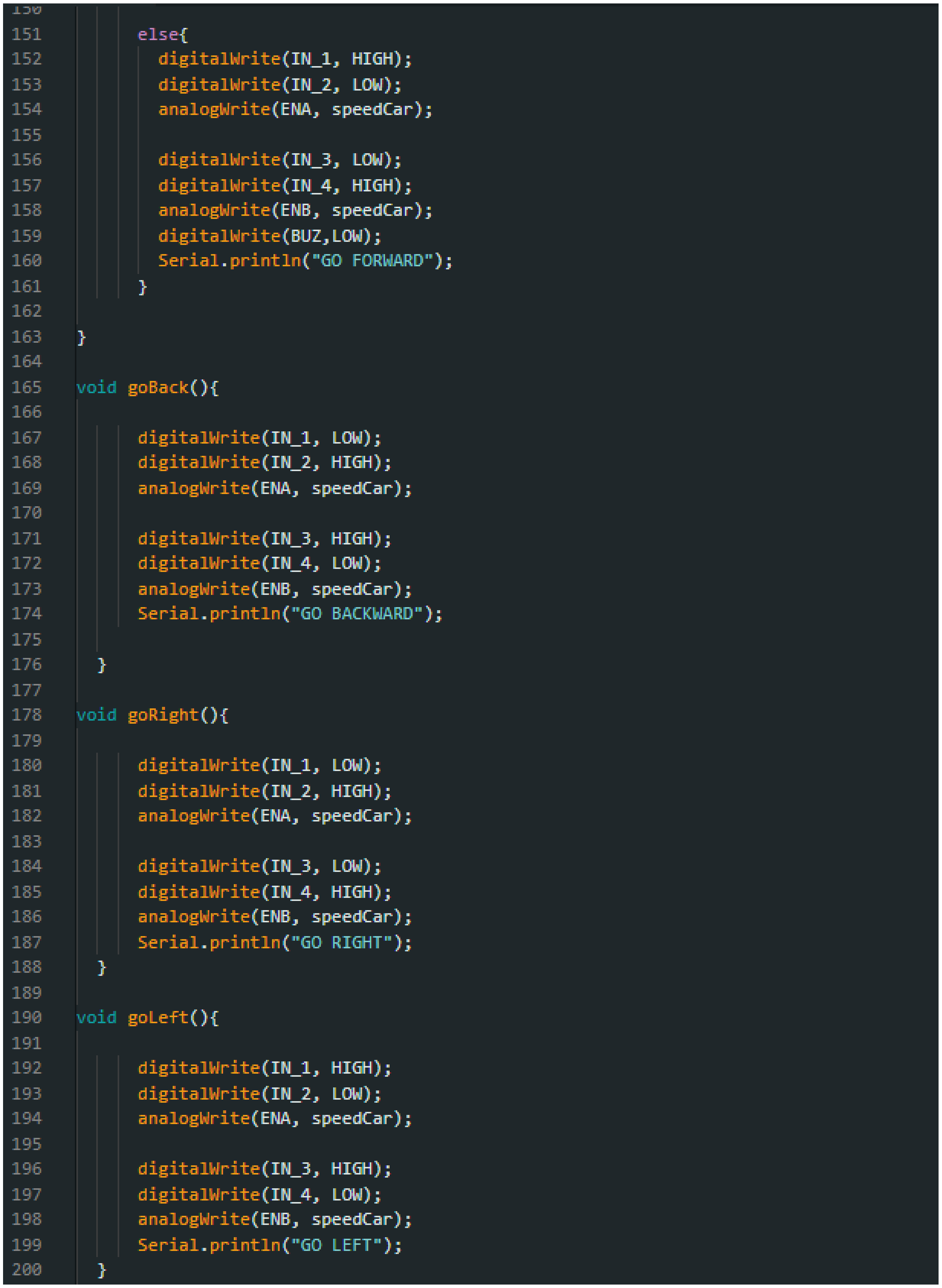
```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

//SSID and Password to your ESP Access Point
const char* ssid = "Robot Wifi";
const char* password = "87654321";

#define ENA   4     // Enable/speed motors Right    GPIO4(D2)
#define IN_1  0     // L298N in1 motors Right       GPIO0(D3)
#define IN_2  2     // L298N in2 motors Right       GPIO2(D4)
#define IN_3  12    // L298N in3 motors Left        GPIO12(D6)
#define IN_4  13    // L298N in4 motors Left        GPIO13(D7)
#define ENB   15    // Enable/speed motors Left     GPIO15(D8)

String command;             //String to store app command state.
int speedCar = 150;         // 0 to 255
int speed_low = 60;


ESP8266WebServer server(80);

int ir1 = 14;// connect IR sensor module to Arduino pin D5
int ir2 = 16;
int BUZ = 5; // connect LED to Arduino pin 13
int s1;      //to sense ir1
int s2;      //to sense ir2


void setup() {

 Serial.begin(115200);

 pinMode(ENA, OUTPUT);
 pinMode(IN_1, OUTPUT);
 pinMode(IN_2, OUTPUT);
 pinMode(IN_3, OUTPUT);
 pinMode(IN_4, OUTPUT);
 pinMode(ENB, OUTPUT);
 pinMode(ir1, INPUT);                   // IR Sensor pin INPUT
 pinMode(ir2,INPUT);
 pinMode(BUZ, OUTPUT);                  // LED Pin Output

  // Connecting WiFi
  WiFi.mode(WIFI_AP);                   //Only Access point
  WiFi.softAP(ssid, password);          //Start HOTspot removing password will disable security
```



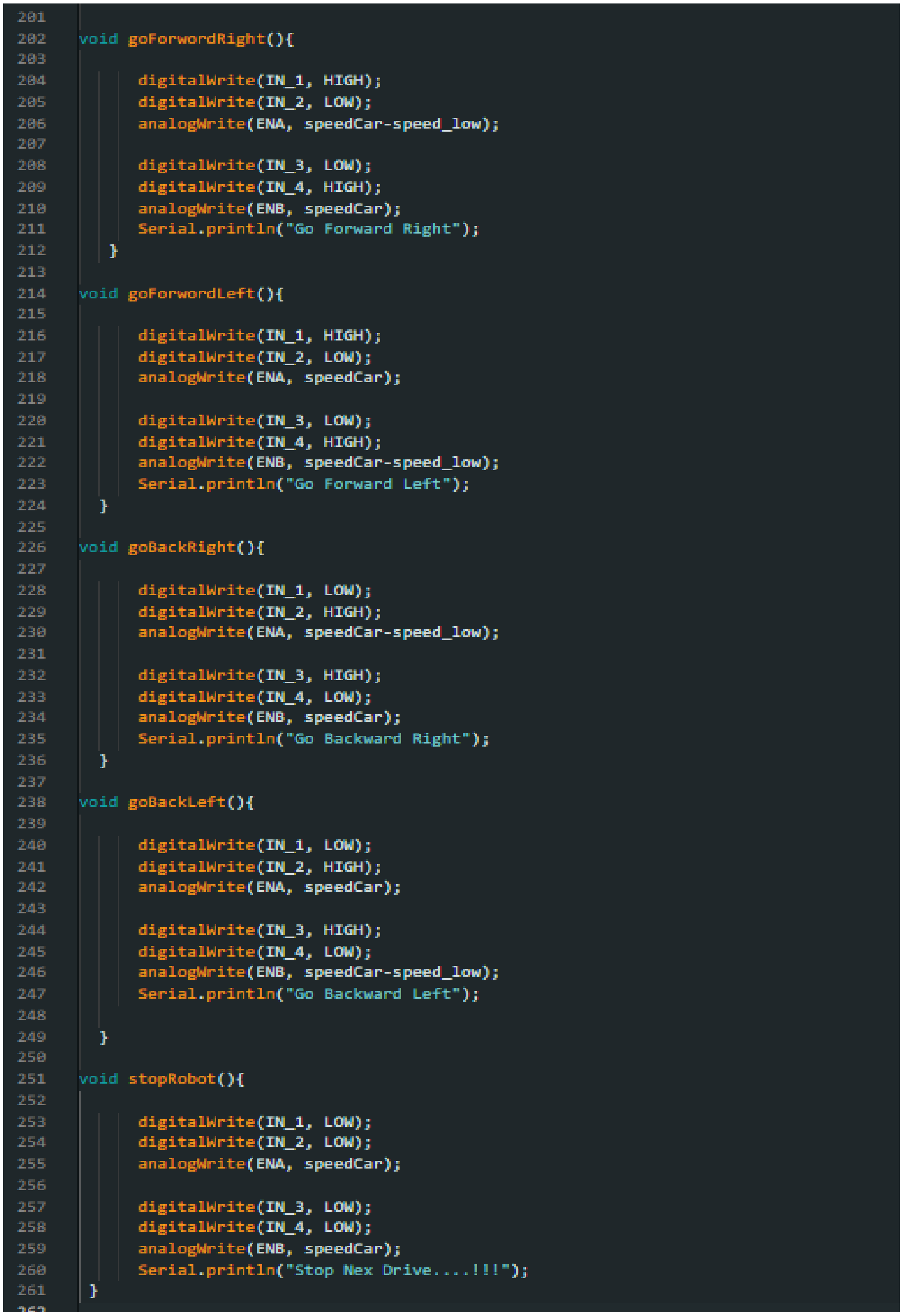
```
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);

  // Starting WEB-server
  server.on ( "/", HTTP_handleRoot );
  server.onNotFound ( HTTP_handleRoot );
  server.begin();


}

void loop() {

      server.handleClient();

      command = server.arg("State");
      if (command == "F") goForword();
      else if (command == "B") goBack();
      else if (command == "L") goLeft();
      else if (command == "R") goRight();
      else if (command == "I") goForwordRight();
      else if (command == "G") goForwordLeft();
      else if (command == "J") goBackRight();
      else if (command == "H") goBackLeft();
      else if (command == "W") digitalWrite(BUZ, HIGH); // light is on
      else if (command == "w") digitalWrite(BUZ, LOW);  // light is off
      else if (command == "A") Automatic();
      else if (command == "0") speedCar = 100;
      else if (command == "1") speedCar = 120;
      else if (command == "2") speedCar = 140;
      else if (command == "3") speedCar = 160;
      else if (command == "4") speedCar = 180;
      else if (command == "5") speedCar = 200;
      else if (command == "6") speedCar = 215;
      else if (command == "7") speedCar = 230;
      else if (command == "8") speedCar = 240;
      else if (command == "9") speedCar = 255;
      else if (command == "S") stopRobot();

}

void HTTP_handleRoot(void) {
    if( server.hasArg("State") ){
           Serial.println(server.arg("State"));
    }
    server.send ( 200, "text/html", "" );
    delay(1);
}
```



```
void Automatic(){

      s1 = digitalRead(ir1); // Set the GPIO as Input
      s2 = digitalRead(ir2);

      if(s1==0 && s2==0){
        digitalWrite(BUZ,HIGH);
        goBack();
        delay(800);
        digitalWrite(BUZ,LOW);
        delay(1700);
        goRight();
        delay(1500);
        Serial.println("Motion detected in IR1 and IR2");
      }

      else if (s1 == 0) // Check if the pin high or not
      {
        digitalWrite(BUZ, HIGH); // LED High
        Serial.println("Motion Detected!  IR1, GO LEFT"  ); // print Motion Detected! on th
        goLeft();
      }

      else if (s2 == 0) // Check if the pin high or not
      {
        // if the pin is high turn off the onboard Led
        digitalWrite(BUZ, HIGH); // LED High
        Serial.println("Motion Detected!IR2 , GO RIGHT"); // print Motion Detected! on the
        goRight();
      }

      else  {
        //else turn on the onboard LED
        digitalWrite(BUZ, LOW); // LED LOW
        Serial.println("Motion Ended! IR1 and IR2, GO FORWARD"); // print Motion Ended! on
        goForword();
      }
}

void goForword(){
      s1 = digitalRead(ir1);
      s2 = digitalRead(ir2);

      if(s1 == 0 && s2 == 0){
        digitalWrite(BUZ,HIGH);
        delay(300);
        stopRobot();
        digitalWrite(BUZ,LOW);
      }

```



```
        else{
          digitalWrite(IN_1, HIGH);
          digitalWrite(IN_2, LOW);
          analogWrite(ENA, speedCar);

          digitalWrite(IN_3, LOW);
          digitalWrite(IN_4, HIGH);
          analogWrite(ENB, speedCar);
          digitalWrite(BUZ,LOW);
          Serial.println("GO FORWARD");
        }

}

void goBack(){

      digitalWrite(IN_1, LOW);
      digitalWrite(IN_2, HIGH);
      analogWrite(ENA, speedCar);

      digitalWrite(IN_3, HIGH);
      digitalWrite(IN_4, LOW);
      analogWrite(ENB, speedCar);
      Serial.println("GO BACKWARD");

  }

void goRight(){

      digitalWrite(IN_1, LOW);
      digitalWrite(IN_2, HIGH);
      analogWrite(ENA, speedCar);

      digitalWrite(IN_3, LOW);
      digitalWrite(IN_4, HIGH);
      analogWrite(ENB, speedCar);
      Serial.println("GO RIGHT");
  }

void goLeft(){

      digitalWrite(IN_1, HIGH);
      digitalWrite(IN_2, LOW);
      analogWrite(ENA, speedCar);

      digitalWrite(IN_3, HIGH);
      digitalWrite(IN_4, LOW);
      analogWrite(ENB, speedCar);
      Serial.println("GO LEFT");
  }
```



```
void goForwordRight(){

    digitalWrite(IN_1, HIGH);
    digitalWrite(IN_2, LOW);
    analogWrite(ENA, speedCar-speed_low);

    digitalWrite(IN_3, LOW);
    digitalWrite(IN_4, HIGH);
    analogWrite(ENB, speedCar);
    Serial.println("Go Forward Right");
  }

void goForwordLeft(){

    digitalWrite(IN_1, HIGH);
    digitalWrite(IN_2, LOW);
    analogWrite(ENA, speedCar);

    digitalWrite(IN_3, LOW);
    digitalWrite(IN_4, HIGH);
    analogWrite(ENB, speedCar-speed_low);
    Serial.println("Go Forward Left");
  }

void goBackRight(){

    digitalWrite(IN_1, LOW);
    digitalWrite(IN_2, HIGH);
    analogWrite(ENA, speedCar-speed_low);

    digitalWrite(IN_3, HIGH);
    digitalWrite(IN_4, LOW);
    analogWrite(ENB, speedCar);
    Serial.println("Go Backward Right");
  }

void goBackLeft(){

    digitalWrite(IN_1, LOW);
    digitalWrite(IN_2, HIGH);
    analogWrite(ENA, speedCar);

    digitalWrite(IN_3, HIGH);
    digitalWrite(IN_4, LOW);
    analogWrite(ENB, speedCar-speed_low);
    Serial.println("Go Backward Left");

  }

void stopRobot(){

    digitalWrite(IN_1, LOW);
    digitalWrite(IN_2, LOW);
    analogWrite(ENA, speedCar);

    digitalWrite(IN_3, LOW);
    digitalWrite(IN_4, LOW);
    analogWrite(ENB, speedCar);
    Serial.println("Stop Nex Drive....!!!");
 }
```

Team THETA Presents
NeXdrive
A Voice-Guided, Auto Obstacle-Aware car
REMOTLY CONTROLLED
INTUITIVE VOICE CONTROL
LANE DEPARTURE WARNING
ADVANCED OBJECT DETECTION
AUTOMATIC DRIVING CAPABILITY