KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

Cyber Physics in Industrial IOT Group Project

Under the guidance of Dr. Subhra Debdas

School of Electrical Engineering, KIIT University

Topic: Distance Measurement Using HC-SR04 Via NodeMCU (ESP8266)

Submitted By:

1. **Ajit Tripathy (2230142)**
2. **Arya Vats (2230155)**
3. **Atul Rajput (2230158)**
4. **Deep Habiswashi (2230167)**
5. **Manaswita Dey (2330610)**

Title: Distance Measurement Using HC-SR04 Via NodeMCU (ESP8266)

Introduction:

Distance measurement is a fundamental aspect in various applications, from robotics to home automation. This experiment focuses on integrating the HC-SR04 ultrasonic sensor with the NodeMCU (ESP8266) microcontroller to create a wireless and versatile distance measurement system. The project aims to explore the accuracy, adaptability, and potential applications of this IoT-based solution.

Objective:

The primary objective is to design, implement, and evaluate a distance measurement system using the HC-SR04 sensor and NodeMCU. Specific goals include assessing accuracy, enabling wireless connectivity, showcasing versatility in different applications, and providing a foundation for future enhancements in the realm of IoT and automation.

Methodology:

Requirements:

1. NodeMCU (ESP8266)
2. Breadboard
3. HC-SR04 Ultrasonic Sensor
4. USB Cable
5. Jumper Wires
6. Arduino IDE Software

NodeMCU:
NodeMCU is an open-source IoT platform built around the ESP8266 Wi-Fi module. It combines a 32-bit microcontroller with integrated Wi-Fi capabilities on a single board. With support for both Lua scripting and Arduino programming, it caters to a wide range of developers, enabling the creation of IoT applications and projects. The NodeMCU board offers easy access to GPIO pins for connecting sensors and peripherals. Known for its affordability, compact size, and active community, NodeMCU is widely used for prototyping and developing IoT solutions.

Breadboard:
A breadboard is a versatile tool used in electronics for prototyping and testing circuits without soldering. It features a grid layout of interconnected holes, allowing components to be easily

inserted and connected by jumper wires. The power rails along the edges provide convenient access to power (VCC) and ground (GND). Breadboards are reusable, enabling rapid iteration and modification of circuit designs. They are essential for beginners and experienced engineers alike, facilitating the creation and testing of electronic circuits in a temporary and flexible manner.

HC-SR04 Ultrasonic Sensor:
The HC-SR04 Ultrasonic Sensor is a popular distance measurement device commonly used in robotics, automation, and various electronic projects. The HC-SR04 uses ultrasonic sound waves to measure distance. It emits ultrasonic pulses and calculates the time it takes for the pulses to bounce back after hitting an object, providing an accurate distance measurement.

Specifications:
Operating Voltage: Typically operates at 5V DC.
Current Consumption: Around 15mA.
Operating Frequency: 40 kHz.
Measurement Range: Commonly has a range of **2 cm to 400 cm** (0.79 inches to 157.48 inches).
Accuracy: Generally provides a high level of accuracy, often with a resolution of 3 mm.
Trigger Pulse: Requires a short 10-microsecond trigger pulse to initiate measurements.

Components of HC-SR04 Ultrasonic Sensor:

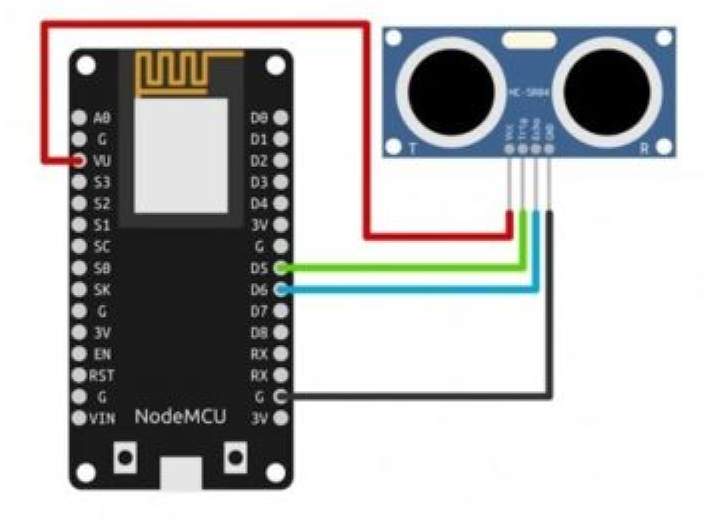
Transducer: Emits ultrasonic pulses and receives echoes.
Control Circuit: Manages the timing of the ultrasonic pulses and calculates distance.
Echo and Trigger Pins: Connects to a microcontroller or other control circuit. The **trigger pin** initiates the measurement, and the **echo pin** outputs the pulse width proportional to the distance.

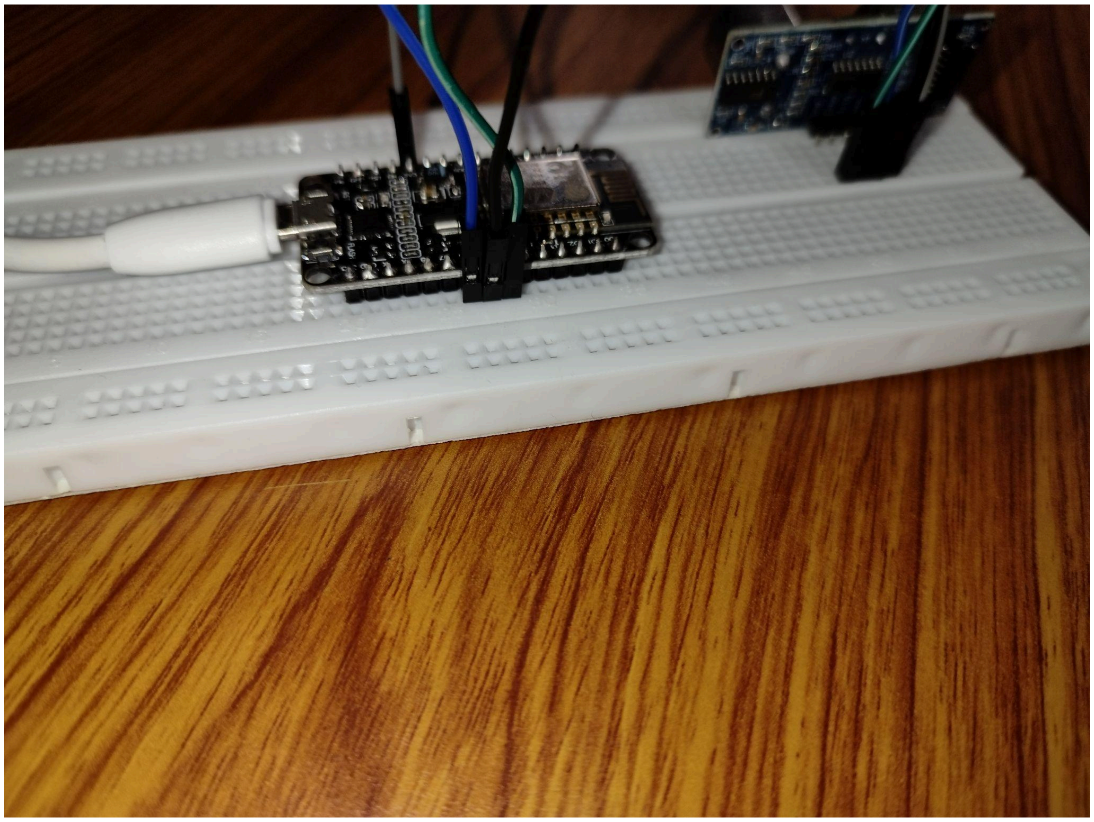
Arduino IDE:
The Arduino IDE is an open-source software platform designed for programming and developing applications for Arduino microcontroller boards. It features a user-friendly code editor with syntax highlighting and includes a library manager for easy integration of pre-written code modules. The IDE supports a variety of Arduino-compatible boards, and its simplicity makes it accessible to beginners. The built-in Serial Monitor assists in debugging and communication between the Arduino board and a computer. Regular updates and a thriving community contribute to its widespread use for a range of electronic projects.

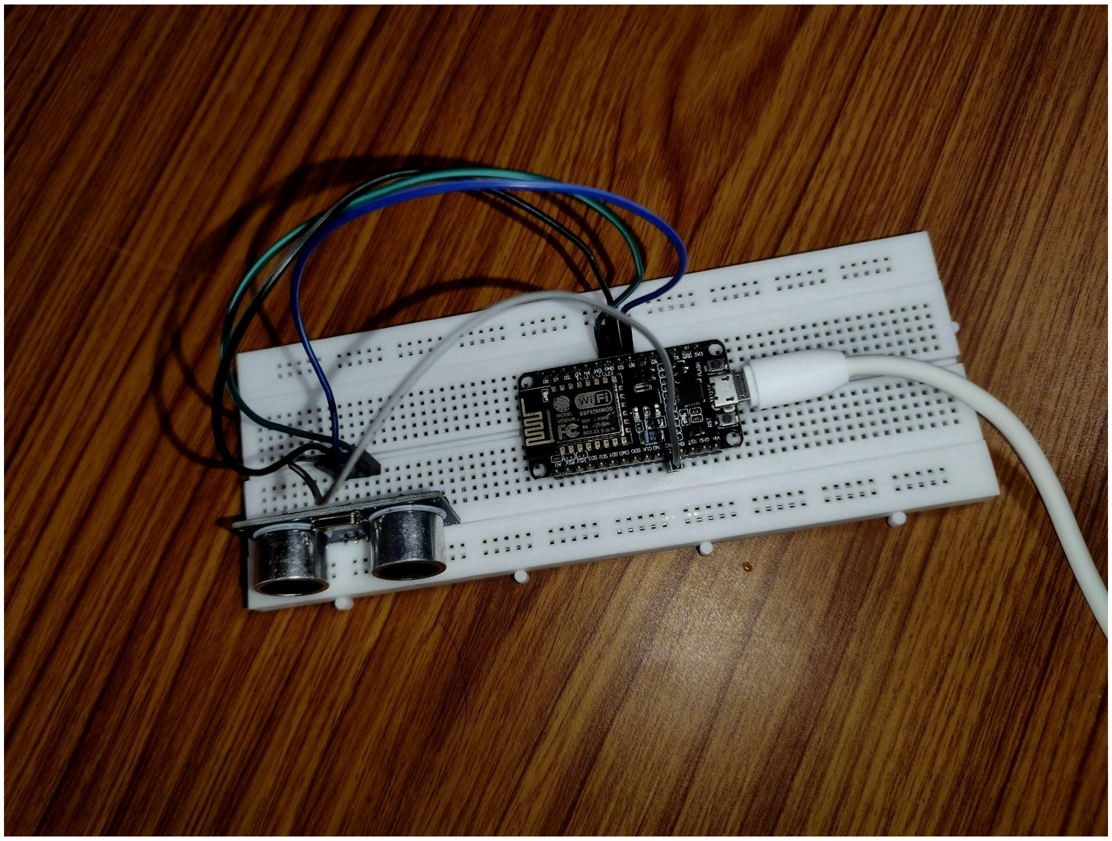
Setup and Hardware Configuration:
Assemble the hardware components, including the HC-SR04 sensor and NodeMCU microcontroller.

Connect components according to specified pin configurations:

- The VCC pin of the Ultrasonic sensor is connected to the VU pin of the NodeMCU.
- The GND pin of the Ultrasonic sensor is connected to the GND pin of the NodeMCU.
- The TRIG pin of the Ultrasonic sensor is connected to the D5 pin of the NodeMCU.
- The ECHO pin of the Ultrasonic sensor is connected to the D6 pin of the NodeMCU.







Programming the NodeMCU using Arduino IDE:

- Develop the Arduino program to read data from the HC-SR04 sensor and calculate distance.
- Create an algorithm to trigger the ultrasonic sensor, measure echo signal time, and calculate distance.

Code:
Upload the following code into the NodeMCU:

```
// Define constants for the ultrasonic sensor pins
const int trigPin = D5;
const int echoPin = D6;

// Variables to store ultrasonic sensor measurements
long duration;
int distance;
```

```
// Setup function runs once at the beginning
void setup()
{
  // Set the trigPin as an output
  pinMode(trigPin, OUTPUT);

  // Set the echoPin as an input
  pinMode(echoPin, INPUT);

  // Start serial communication at a baud rate of 9600
  Serial.begin(9600);
}

// Loop function runs repeatedly
void loop()
{
  // Clear the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Set the trigPin on HIGH state for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echoPin, which returns the sound wave travel time in
microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance using the formula: distance = speed * time
/ 2
```

```
  // Speed of sound is approximately 34 cm per millisecond (or 0.034
cm/microsecond)
  distance = duration * 0.034 / 2;

  // Print the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println("cm");

  // Wait for 2 seconds before taking the next measurement
  delay(2000);
}
```

Code Explanation:

Firstly the D5 and D6 pins of the NodeMCU are defined as trigPin and echoPin respectively, as constants with integer data type.

Then a variable “duration” is defined with a float data type because it will hold the value of the sound wave travel time (microseconds) which will have decimal points.

The variable distance is in the integer data type because we require it to be in solid integers rather than floating point values with decimal points.

In the <void setup()> function the pin modes for trigPin and echoPin are set, moreover the serial communication between the PC and the NodeMCU is initialised at a baud rate of 9600 bits per second.

In the <void loop()> function, firstly the trigPin is cleared by setting it LOW and waiting for 2 milliseconds, and then it is set HIGH for 10 millisecond as to transmit a 10 ms sound wave trigger pulse, again setting it LOW afterwards.

The time taken by the pulse to hit an object and returning back is calculated by the pulseIn function which measures that after how long the echoPin received a HIGH after the sound wave trigger pulse was transmitted, this time is saved in the duration variable.

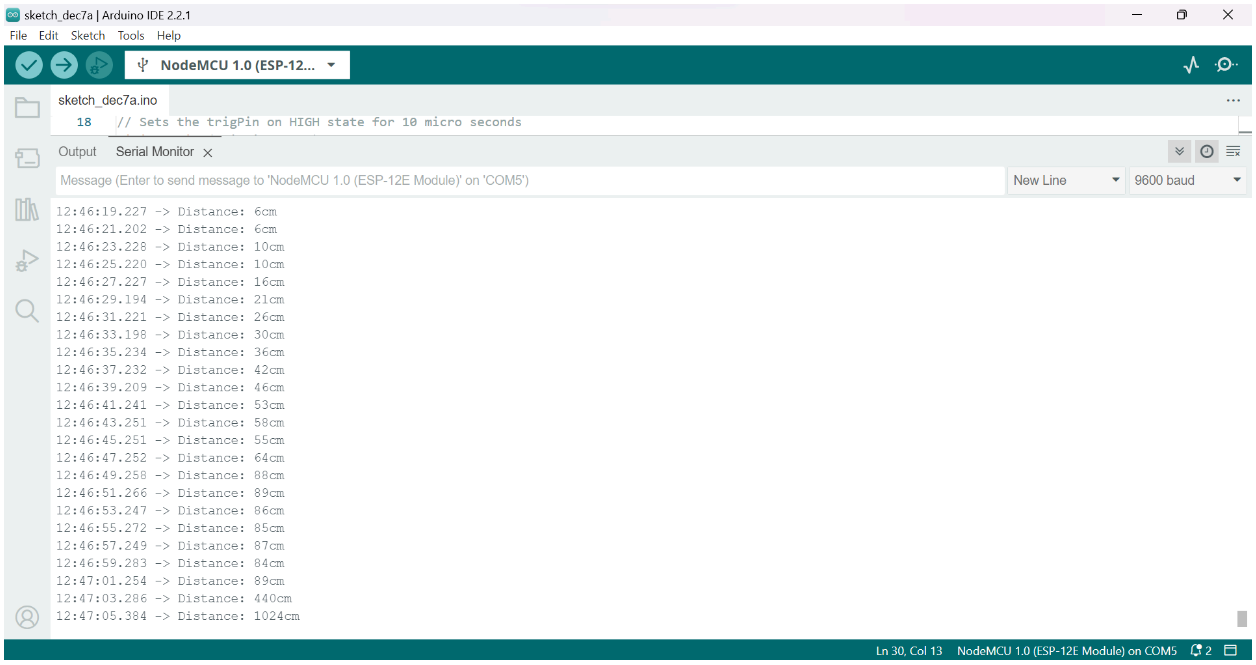
Finally the distance is calculated by multiplying the duration with the velocity of sound (0.034 centimetres/microseconds) as to employ the formula S=Vt, this distance is is divided by two because it is calculated on the two-way time (pulse going to the object and returning back after hitting the object) and only one-way of it is sufficient.

Testing and Calibration:

- Conduct initial tests in a controlled environment.
- Calibrate the system if necessary for improved accuracy.

Data Logging and Output:

After the connections are made and the code is uploaded in the NodeMCU, the Arduino Serial Monitor would be showing the distance between the ultrasonic sensor and the object placed in front of it in centimetre.


sketch_dec7a | Arduino IDE 2.2.1
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12...
sketch_dec7a.ino
Output
Serial Monitor
New Line
9600 baud
Ln 30, Col 13
NodeMCU 1.0 (ESP-12E Module) on COM5

```
// Sets the trigPin on HIGH state for 10 micro seconds
```

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM5')

```
12:46:19.227 -> Distance: 6cm
12:46:21.202 -> Distance: 6cm
12:46:23.228 -> Distance: 10cm
12:46:25.220 -> Distance: 10cm
12:46:27.227 -> Distance: 16cm
12:46:29.194 -> Distance: 21cm
12:46:31.221 -> Distance: 26cm
12:46:33.198 -> Distance: 30cm
12:46:35.234 -> Distance: 36cm
12:46:37.232 -> Distance: 42cm
12:46:39.209 -> Distance: 46cm
12:46:41.241 -> Distance: 53cm
12:46:43.251 -> Distance: 58cm
12:46:45.251 -> Distance: 55cm
12:46:47.252 -> Distance: 64cm
12:46:49.258 -> Distance: 88cm
12:46:51.266 -> Distance: 89cm
12:46:53.247 -> Distance: 86cm
12:46:55.272 -> Distance: 85cm
12:46:57.249 -> Distance: 87cm
12:46:59.283 -> Distance: 84cm
12:47:01.254 -> Distance: 89cm
12:47:03.286 -> Distance: 440cm
12:47:05.384 -> Distance: 1024cm
```

Challenges and Solutions:

During the implementation of the Distance Measurement Using HC-SR04 Via NodeMCU (ESP8266) project, several challenges were faced:

1. Making correct circuit connections was not easy.
2. Error in Arduino Code
3. Not providing accurate distance

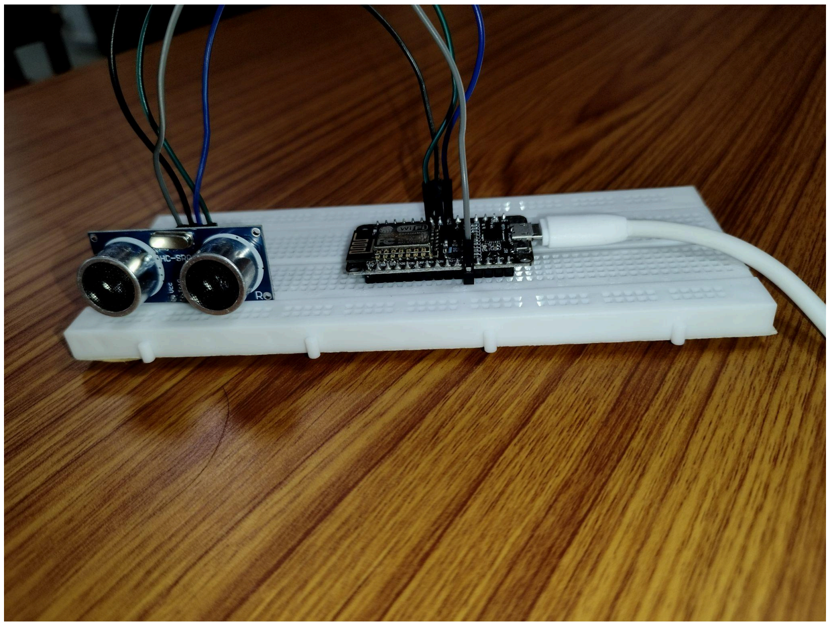
Solutions are as follows:

1. Carefully review and verify the circuit connections, ensuring they align with the specifications. Utilise clear diagrams and documentation to aid in the correct assembly of the hardware components.
2. Debug the code systematically, checking for syntax errors, logical issues, and compatibility with the NodeMCU platform. Utilise serial debugging to identify and address code-related issues effectively.

3. Investigate potential causes for inaccurate readings, such as sensor calibration, environmental factors, or interference.

Lesson Learned:

1. Comprehensive testing of both hardware connections and software code is crucial for identifying and resolving challenges.
2. Clear and detailed documentation is essential for troubleshooting and future reference.
3. Ongoing calibration is necessary to maintain accuracy, especially in dynamic environments.



Future Direction:

- Explore the integration of more advanced distance sensors, such as LIDAR or infrared sensors, to enhance accuracy and expand the range of applications.
- Investigate the incorporation of machine learning algorithms to enable the system to adapt and learn from various environments, improving accuracy over time.
- Implement communication protocols that enable seamless integration with other IoT devices, allowing for collaborative data collection and analysis.

Conclusion:

In conclusion, the experiment on "Distance Measurement Using HC-SR04 Via NodeMCU (ESP8266)" has provided valuable insights into the integration of ultrasonic sensors with IoT platforms for accurate and wireless distance measurements. The challenges encountered during the project, including circuit connections, code errors, and accuracy issues, have been addressed through systematic testing, debugging, and calibration efforts.

References:

1. "Getting Started with Arduino IDE 2 | Arduino Documentation." https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2
2. F. Rahmah, F. Hidayanti, and M. Innah, "Penerapan Smart Sensor untuk Kendali pH dan Level Larutan Nutrisi pada Sistem Hidroponik Tanaman Pakcoy," *Jurnal Teknologi Informasi Dan Ilmu Komputer*, vol. 6, no. 5, pp. 527–534, Oct. 2019, doi: 10.25126/jtiik.2019651738.
3. "Parking space detection using ultrasonic sensor in parking assistance system," *IEEE Conference Publication | IEEE Xplore*, Jun. 01, 2008. https://ieeexplore.ieee.org/abstract/document/4621296