Project report on

Remote Temperature and Distance Measurement using ESP-NOW protocol

Submitted by:

Group Name: Team iπ

Group Members:

1. Mohit Talwar BT22ECI036
2. Plawang Shishu BT22ECI039
3. Ashwani Baghel BT22ECI048

A report submitted for the partial fulfilment of the requirements of the course

ECL-108 IoT Workshop - 1

Submission Date: 12/06/2023

Task Number # 4

Under the guidance of:

Dr. Mayank B. Thacker

Department of Electronics and Communication Engineering

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर

Indian Institute of Information Technology, Nagpur

Table of Contents:

Chapter 1: Introduction

The **Internet of things** (IoT) describes physical objects (or groups of such objects) with sensors, processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks.[1]

The use of IoT devices for **monitoring** and controlling environmental parameters such as temperature, humidity, and air quality has become crucial in many applications.

In this project, we aim to develop an IoT-based system for monitoring attendance and calculating average temperature with multiple **ultrasonic** sensors ,**DHT11** sensors and the **ESP32** microcontroller. The reading of Attendance and average temperature is shown on the **LCD** and the for the average temperature , if the range of temperature goes beyond a certain set level , an indicator like **LED** starts blinking , we can also set it up as a notification based system by using different appropriate libraries.

HC-SR04 is a popular ultrasonic sensor module that is widely used in electronics projects for measuring distances. It uses ultrasonic waves to determine the distance between the sensor and an object by calculating the time taken for the sound wave to bounce back to the sensor. The HC-SR04 sensor is easy to use and low cost, making it a popular choice.

The **DHT11** sensor is a popular sensor that is widely used for monitoring temperature and humidity. It provides accurate readings and is cost-effective.

The **ESP32** is a series of low-cost and low-power System on a Chip (SoC) microcontrollers developed by Espressif that include Wi-Fi and Bluetooth wireless capabilities and dual-core processor. [2]

THE ESP-NOW PROTOCOL

Espressif, the makers of the ESP8266 and ESP32, have developed a protocol that allows all these devices to create a private, wireless network using the **2.5GHz** transceivers. This is a separate network from the WiFi network and can only be used by ESP-type microcontrollers. The protocol is called ESP-NOW.

In ESP-NOW, application data is encapsulated in a vendor-specific action frame and then transmitted from one Wi-Fi device to another without connection. CTR with CBC-MAC Protocol(CCMP) is used to protect the action frame for security. ESP-NOW is widely used in smart light, remote controlling, sensor, etc.

In this arrangement, the Initiator ESP32 transmits data to the Responder ESP32. The Initiator can tell if the Responder received the message successfully.

Chapter 2: Methodology and Design

Methodology

The first step was to analyze the requirements of the system, which included the sensors needed to measure the student count/attendance and the temperature. The next step involved selecting the appropriate hardware components, programming the microcontroller to process the data, and designing the user interface. The last step was to test and validate the system, this involved testing the system to ensure that it met the requirements and specifications.

Design

The design comprises of four main hardware components, namely the ESP32 microcontroller, the DHT11 sensors, and the Ultrasonic Sensors.

DIAGRAMS OF HARDWARE USED IN THE PROJECT

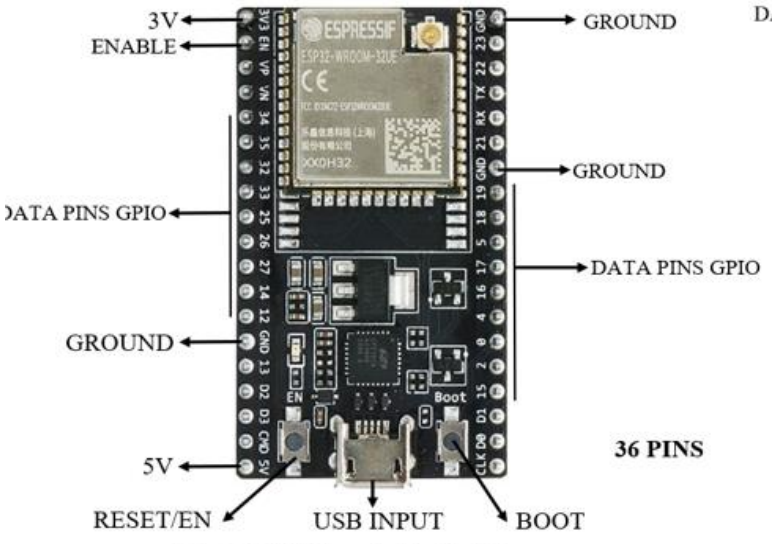


Fig. 1: ESP32 board: labelled diagram

Fig. 2: DHT11: labelled diagram

Fig. 3: HC-SR04 Diagram

LCD DIAGRAM (SIMILAR LOGIC FOR ESP32)

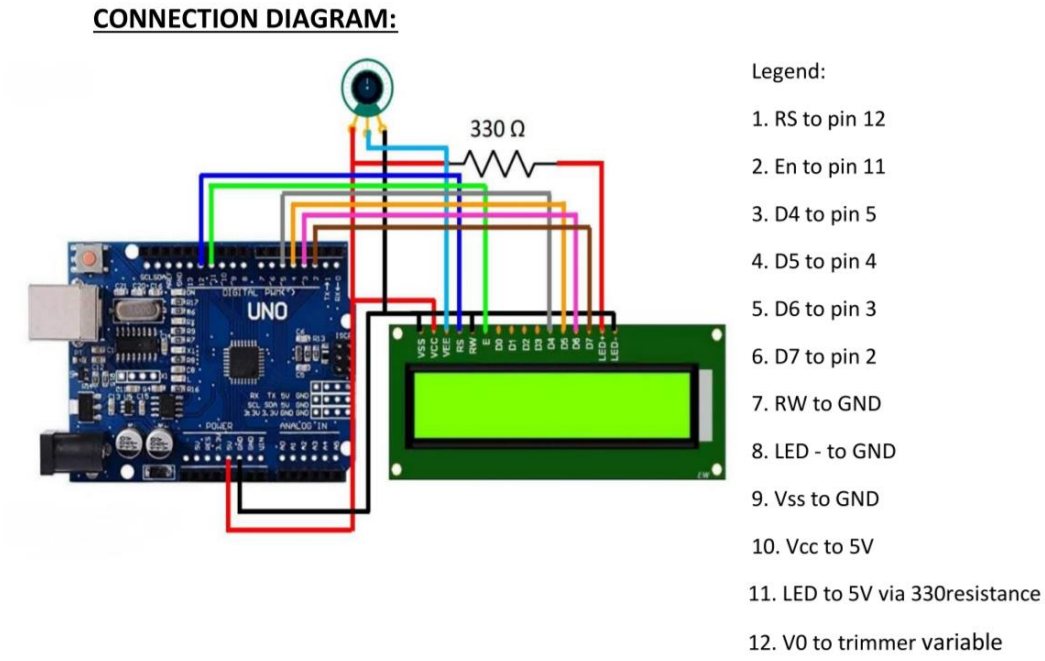
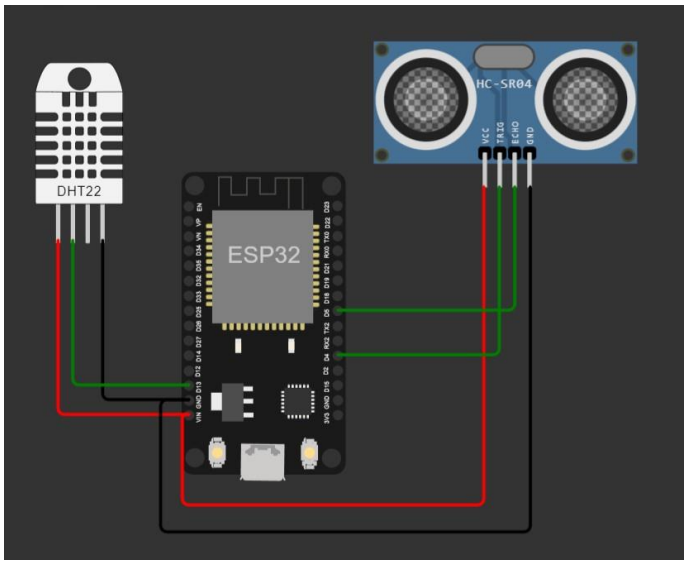


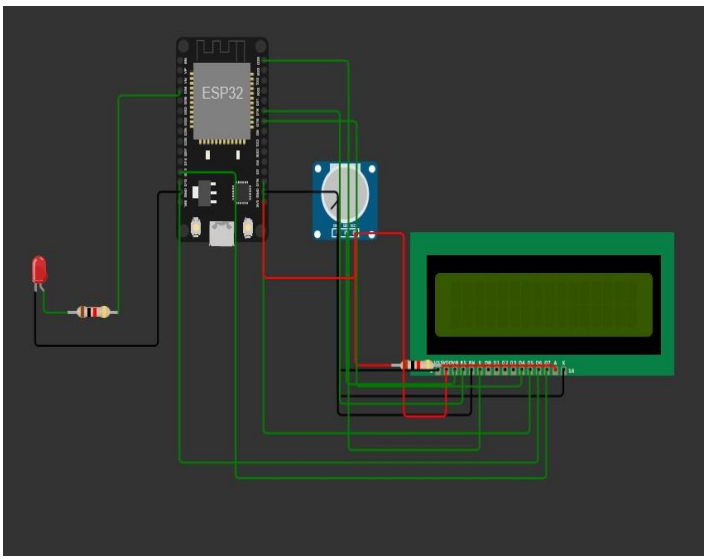
Figure 4 : LCD

Figure 5 : Simulation

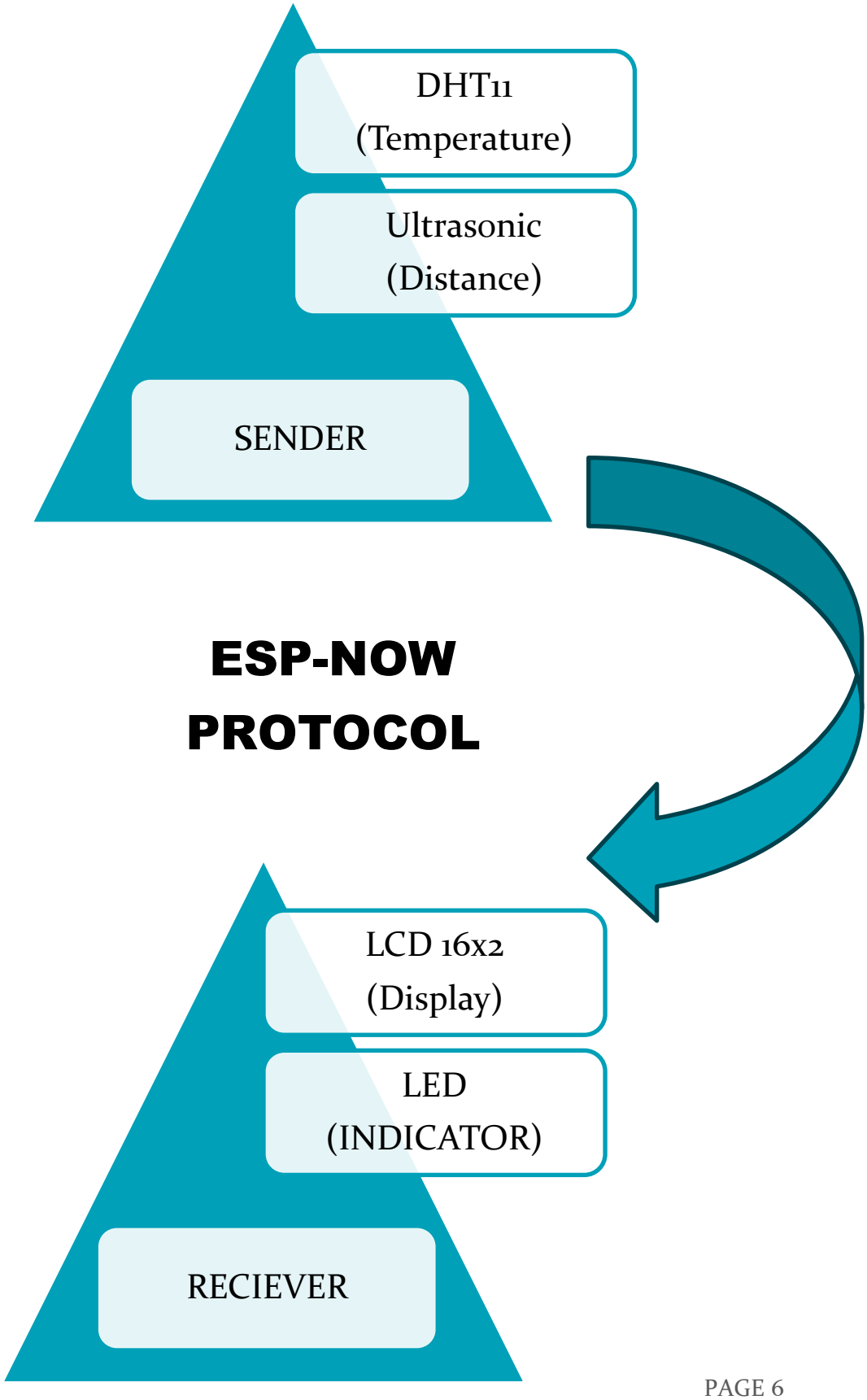
Sender



Receiver



Chapter 3: Flowchart



Chapter 4: Code

ESP32: Getting Board MAC Address

```
#include "WiFi.h"
void setup(){
  Serial.begin(115200);
  WiFi.mode(WIFI_MODE_STA);
  Serial.println(WiFi.macAddress());
}

void loop(){ }
```

SENDER ESP32

```
#include <WiFi.h>
#include <esp_now.h>
#include <DHT.h>

// Define DHT11 parameters
#define DHTPin 27
#define DHTType DHT11

// Create DHT Object
```

```
DHT dht(DHTPin, DHTType);

int trigPin = 33;    // TRIG pin
int echoPin = 32;    // ECHO pin

float duration_us, distance;

// Variables for temperature
float temp;

// Responder MAC Address (Replace with your responders MAC Address)
uint8_t broadcastAddress[] = {0xB8, 0xD6, 0x1A, 0xB3, 0x09, 0x7C};

// Define data structure
typedef struct struct_message {
  float a;
  float b;

} struct_message;

struct_message myData; // Create structured data object
```

```
// Register peer
esp_now_peer_info_t peerInfo;

// Sent data callback function
void OnDataSent(const uint8_t *macAddr,
esp_now_send_status_t status)
{
  Serial.print("Last Packet Send Status: ");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ?
"Delivery Success" : "Delivery Fail");
}

void setup() {
  Serial.begin(115200);
  delay(100);
            // configure the trigger pin to output mode
  pinMode(trigPin, OUTPUT);
            // configure the echo pin to input mode
  pinMode(echoPin, INPUT);
             // Initiate DHT11
  dht.begin();
```

```
                // Set ESP32 WiFi mode to Station temporarly
  WiFi.mode(WIFI_STA);
                 // Initialize ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
                // Define callback
  esp_now_register_send_cb(OnDataSent);
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }}

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

```
digitalWrite(trigPin, LOW);

// measure duration of pulse from ECHO pin
duration_us = pulseIn(echoPin, HIGH);

// calculate the distance
distance = 0.017 * duration_us;

// print the value to Serial Monitor
Serial.print("distance: ");
Serial.print(distance);
Serial.println(" cm");

delay(500);

// Read DHT22 module values
temp = dht.readTemperature();
delay(10);


Serial.print("Temp: ");
Serial.println(temp);
```

```
  Serial.print("Distance: ");
  Serial.println(distance);

  // Add to structured data object
  myData.a = temp;
  myData.b = distance;

  // Send data
  esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));

  delay(2000);
}
```

RECIEVER ESP32

```
#include <WiFi.h>
#include <esp_now.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(19, 23, 18, 14, 27, 15);

// Define data structure
```

```
typedef struct struct_message {
    float a;
    float b;
} struct_message;

struct_message myData;  // Create structured data object

// Callback function
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len)
{
  memcpy(&myData, incomingData, sizeof(myData));   // Get incoming data

  lcd.setCursor(10,1);
  lcd.print(myData.a);
  lcd.setCursor(14,0);
  lcd.print(myData.b);

  if (myData.a > 25){
    digitalWrite(9, LOW);
    delay(500);
    digitalWrite(9, HIGH);
```

```
    delay(500);  }
}

void setup() {
  pinMode(26, OUTPUT);  //FOR LED
  lcd.begin(16,2);
  lcd.print("Distance:-");
  lcd.setCursor(0,1);
  lcd.print("Temp:-");

  WiFi.mode(WIFI_STA);   // Start ESP32 in Station mode

  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  // Register callback function
  esp_now_register_recv_cb(OnDataRecv);
}
```

Chapter 5: Limitations and Future Scope of this Project

The IoT setup of measuring temperature and distance by DHT11 sensors and HCS04 ultrasonic sensors remotely and wirelessly has some limitations and challenges that need to be considered. Some of these limitations include:

1. **Sensor Accuracy**: The DHT11 sensor is known for its lower accuracy as compared to other sensors, which could lead to inaccurate readings. The HCS04 ultrasonic sensor can also give inaccurate readings in noisy environments or with the presence of obstacles.

2. **Placement of sensors**: The placement of the sensors is critical for accurate readings. The placement of the sensors needs to be carefully planned to avoid sources of interference.

3. **Power Consumption**: The ESP32 microcontroller consumes a considerable amount of power. This could be a challenge in battery-powered applications where power consumption needs to be minimized.

4. **Privacy**: The use of ultrasonic sensors for counting people could raise privacy concerns in some applications, such as in residential or public spaces.

Here are some potential future scopes and applications of this project:

1. **Smart Homes**: This project can be used in smart homes to monitor and control the temperature and occupancy levels in different rooms. The system can provide real-time alerts and notifications to the homeowner to optimize energy consumption, improve indoor air quality, and ensure comfort.

2. **Healthcare:** The project can be used in hospitals and clinics to monitor the temperature and occupancy levels in different rooms, such as operating rooms, patient rooms, and waiting areas. The system can provide real-time alerts and notifications to healthcare providers to ensure the safety and comfort of patients and staff.

3. **Commercial Buildings**: The project can be used in commercial buildings, such as offices and shopping malls, to monitor and control the temperature and occupancy levels in different areas. The system can optimize energy consumption and improve indoor air quality while ensuring the comfort of occupants.

4. **Public Spaces**: The project can be used in public spaces, such as airports and train stations, to monitor and control the temperature and occupancy levels in different areas. The system can provide real-time information to travelers to ensure their safety and comfort.

Chapter 6: Conclusion

In conclusion, the IoT setup of measuring temperature by DHT11 and distance by HCS04 ultrasonic sensors remotely wirelessly via ESPNOW protocol is a valuable project that offers real-time monitoring and tracking of temperature and occupancy levels. However, the project has its limitations and challenges, including sensor accuracy, placement of sensors, connectivity, power consumption, and privacy concerns.

To address these limitations and challenges, the project requires careful planning and implementation of potential solutions such as calibration of sensors to improve accuracy, optimization of sensor placement, backup internet connection to improve connectivity, optimization of the code to reduce power consumption, and measures to ensure data anonymity to address privacy concerns.

References:

1. https://en.wikipedia.org/wiki/ESP32
2. https://randomnerdtutorials.com/getting-started-with-esp32/
3. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html
4. https://dronebotworkshop.com/esp-now/
5. https://circuits4you.com/
6. https://www.google.com/
7. https://www.google.co.in/imghp?hl=en&ogbl
8. https://forum.arduino.cc/
9. https://github.com/adafruit/DHT-sensor-library
10. https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
11. https://dl.espressif.com/dl/package_esp32_dev_index.json
12. https://github.com/
13. https://arduino.stackexchange.com/