# My project

Here is the **well-formatted content** with a structured GPIO pin list you can copy directly into your .docx document, following your project's format:

**ESP32 LoRa RA-02 Master-Slave Communication System Analysis**

This project implements a wireless monitoring and control system using ESP32 microcontrollers with LoRa RA-02 modules. The system consists of one Master device and two Slave devices that communicate via LoRa radio.

**System Overview**

**Master Device**

* Acts as a web server (STA mode) with a user interface
* Alternately polls Slave devices for sensor data (DHT11 temperature/humidity)
* Controls LEDs on Slave devices through the web interface
* Displays all data on a web page

**Slave Devices (1 & 2)**

* Each has a DHT11 sensor and two LEDs
* Respond to Master's requests with sensor data
* Accept LED control commands from Master
* Similar code with different addresses (0x02 for Slave 1, 0x03 for Slave 2)

**Key Features**

1. **Dual Communication Modes**:
   * get\_Data\_Mode (Mode 1): Requests sensor data from Slaves
   * led\_Control\_Mode (Mode 2): Sends LED control commands
2. **Web Interface**:
   * Displays temperature/humidity from both Slaves
   * Provides toggle switches to control LEDs remotely
   * Shows status information and timestamps
3. **LoRa Communication**:
   * Operates at 433MHz
   * Includes automatic reset mechanism to prevent freezing
   * Message format: destination|sender|length|mode|content
4. **Error Handling**:
   * DHT11 read failure detection
   * LoRa connection monitoring
   * Message length verification

**Code Structure**

**Master (ESP32\_Lora\_Ra-02\_Master\_Web\_Server\_STA\_MODE.ino)**

* Sets up WiFi access point and web server
* Implements alternating polling of Slaves
* Processes incoming data from Slaves
* Provides web interface using AsyncWebServer

**Slave (ESP32\_Lora\_Ra-02\_Slave\_1.ino)**

* Reads DHT11 sensor periodically
* Responds to Master's data requests
* Implements LED control commands
* Similar structure for Slave 2 (different address)

**Web Page (PageIndex.h)**

* HTML/CSS/JavaScript for the user interface
* Real-time data display using EventSource
* Interactive controls for LEDs
* Visual feedback for system status

**ESP32-WROOM-32 GPIO Pinout Summary**

| **GPIO** | **Functionality** | **Notes** |
| --- | --- | --- |
| 0 | GPIO, strapping pin | Used to enter flash mode (boot pin) |
| 1 | UART TX0 | Debug/Programming (avoid use) |
| 2 | GPIO, strapping pin, built-in LED | Often safe for output |
| 3 | UART RX0 | Debug/Programming (avoid use) |
| 4 | GPIO, PWM, ADC2, HSPI (CS) | Safe for general use |
| 5 | GPIO, PWM, ADC2, HSPI (CLK) | Safe for general use |
| 12 | GPIO, strapping pin | Be cautious, affects boot |
| 13 | GPIO, PWM, ADC2, HSPI (MOSI) | Safe for general use |
| 14 | GPIO, PWM, ADC2, HSPI (MISO) | Safe for general use |
| 15 | GPIO, PWM, ADC2, HSPI (CS), strapping | Be cautious with pull-down on boot |
| 16 | GPIO, UART2 TX | Safe for general use |
| 17 | GPIO, UART2 RX | Safe for general use |
| 18 | GPIO, PWM, SPI (CLK) | Often used for SPI |
| 19 | GPIO, PWM, SPI (MISO) | Often used for SPI |
| 21 | GPIO, PWM, I2C (SDA) | Safe for general use |
| 22 | GPIO, PWM, I2C (SCL) | Safe for general use |
| 23 | GPIO, PWM, SPI (MOSI) | Often used for SPI |
| 25 | GPIO, DAC1, ADC2 | Safe for output/input |
| 26 | GPIO, DAC2, ADC2 | Safe for output/input |
| 27 | GPIO, ADC2 | Safe for general use |
| 32 | GPIO, ADC1, Touch sensor | Input/output supported |
| 33 | GPIO, ADC1, Touch sensor | Input/output supported |
| 34 | GPIO, ADC1 (input only) | Input only |
| 35 | GPIO, ADC1 (input only) | Input only |
| 36 | GPIO, ADC1 (input only, VP sensor) | Input only |
| 39 | GPIO, ADC1 (input only, VN sensor) | Input only |

**⚠️ Avoid Using:**

* GPIO **6–11**: Reserved for SPI Flash
* GPIO **0, 12, 15**: Bootstrapping pins — use carefully
* GPIO **1, 3**: UART debugging (avoid unless reconfigured)

**Potential Improvements**

1. **Security**:
   * Add authentication for web interface
   * Encrypt LoRa communications
2. **Robustness**:
   * Implement message acknowledgment
   * Add retry mechanism for failed transmissions
3. **Scalability**:
   * Support for more Slave devices
   * Dynamic address assignment
4. **Power Management**:
   * Low-power modes for battery operation
   * Adaptive polling intervals
5. **User Experience**:
   * Data logging and visualization
   * Alert/notification system
   * Mobile-friendly interface

Let me know if you want this as a .docx file or need help formatting it into a printable report.

# Communication

Certainly! Below is the content **exactly as you requested**, well-formatted for easy copy-pasting into your .docx document — no changes made:

In this LoRa communication system, 0x02 is the **hexadecimal address** assigned to **Slave 1**. Here's how the communication works when Slave 1 sends data to the Master (0x01):

**1. Message Structure (LoRa Packet Format)**

All messages follow this format:

Destination\_Address | Sender\_Address | Message\_Length | Mode (optional) | Message\_Content

**Example: Slave 1 sending sensor data to Master**

0x01 | 0x02 | 14 | s,80,30.50,0,1

* **0x01**: Destination = Master's address
* **0x02**: Sender = Slave 1's address
* **14**: Message length (14 characters in s,80,30.50,0,1)
* **s,80,30.50,0,1**: Actual data payload (explained below)

**2. How Slave 1 Prepares the Data**

When the Master (in get\_Data\_Mode) requests data, Slave 1:

1. **Reads sensors** (DHT11) and LED states:
   * s = Sensor status (s = success, f = failed)
   * 80 = Humidity (%)
   * 30.50 = Temperature (°C)
   * 0 = LED 1 state (0 = OFF, 1 = ON)
   * 1 = LED 2 state (0 = OFF, 1 = ON)
2. **Constructs the message**:

String Message = "s,80,30.50,0,1"; // Example data

1. **Sends via LoRa** using sendMessage():

LoRa.beginPacket(); // Start packet

LoRa.write(Destination\_Master); // 0x01 (Master)

LoRa.write(LocalAddress); // 0x02 (Slave 1)

LoRa.write(Message.length()); // Length (e.g., 14)

LoRa.print(Message); // Payload ("s,80,30.50,0,1")

LoRa.endPacket(); // Transmit

**3. Master Processes the Data**

When the Master receives the message:

1. **Checks sender address** (0x02 = Slave 1).
2. **Parses the payload** (s,80,30.50,0,1):
   * Splits by commas using GetValue() (e.g., GetValue(Incoming, ',', 2) → 30.50).
3. **Updates the web interface**:
   * Displays temperature (30.50°C), humidity (80%), and LED states.
   * Shows the last received time.

**Key Notes**

* **Hex Addresses**:
  + Master: 0x01
  + Slave 1: 0x02
  + Slave 2: 0x03 (if uncommented in Slave code)
* **Error Handling**:
  + If the DHT11 fails, Slave 1 sends f,0,0.00,0,1 (failed status, zeros for data).
* **LED Control**:
  + When the Master sends 1,t (turn LED 1 ON), Slave 1 sets digitalWrite(LED\_1\_Pin, HIGH).

**Visual Flow**

Slave 1 (0x02) Master (0x01)

| |

| -- 0x01 | 0x02 | 14 | s,80,30.50,0,1 -->|

| | (Updates Web UI)

| <-- 0x02 | 0x01 | 3 | 2 | 1,t ----------| (LED Control Command)

| (Turns ON LED 1) |

This ensures **bidirectional communication** where the Master polls sensors and controls Slaves using LoRa packets with clear addressing.