An-Najah National University Faculty of Engineering

Electrical Engineering Department

Graduation Project

In Partial Fulfillment of the Requirements for the

Bachelor's Degree in Electrical Engineering

Project Title:

Smart weather station that supports wireless & power line communication technology

Submitted by:

- **Ali Mohammad Ali Al-Khalili- ID :12239151**
- **Fathi Fadi Rashid -ID :12239190**
- **Kamel Sharif Kamel Qaraan- ID :12239189**
- **Mohammad Imad Jawad Zaid - ID :12239177**

Supervisor:

Dr. Omar Tamimi

Academic Year: 2025 / 2026

Table of Contents

List of Figures

Dedication

To those whose prayers have lit my path and sustained my spirit...

To my dear parents, who instilled in me a love of learning, instilled confidence in me, and have always been loyal supporters every step of the way.

To my esteemed professors, beacons of knowledge and learning, who have inspired me and shaped my academic and professional thinking.

To my colleagues on this project, partners in success and challenge, whose determination and collective spirit have played a role in the completion of this work.

To everyone who believed in me and supported me with a kind word or a sincere prayer...

I dedicate this work to all of you with love and gratitud

Acknowledgments

By the grace and blessing of God, this modest project has been completed. It represents the fruit of continuous collective effort and fruitful cooperation from many individuals who greatly supported and encouraged me throughout the project's implementation phases.

I extend my sincere thanks and gratitude to **Dr. Omar Al-Tamimi**, the academic supervisor of this project, whose valuable guidance, scientific advice, and precise observations played a major role in the success of this project.

I also extend my deep gratitude to my **esteemed professors** in the Electrical Engineering Department, who contributed to building my scientific and practical knowledge throughout my years of study and set an example of commitment and excellence.

I must also express my deep gratitude to my dear parents, who have generously provided me with moral and material support and have supported me every step of the way.

Finally, I would like to extend my special thanks to my colleagues on the project team:

Ali Muhammad Ali Al-Khalili, Fathi Fadi Rashid,

Kamel Sharif Kamel Qaraan, and Muhammad Imad Jawad Zaid

for the spirit of cooperation, perseverance, and teamwork that brought us together throughout the project's implementation.

Abstract (In English)

Abstract (in English) This project aims to design and implement a smart weather station capable of accurately measuring and analyzing weather conditions in real time. The station supports two communication methods: Wi-Fi and power line communication (PLC). This enhances its ease of use in diverse environments, whether with advanced or limited infrastructure. The station relies on a set of sophisticated sensors to monitor variables such as temperature, humidity, atmospheric pressure, wind speed, rainfall, and gases. A central control unit collects and analyzes this data, then transmits it to an electronic display interface, facilitating real-time weather monitoring. This station is an effective tool that can be used in various fields, such as smart homes, smart agriculture, energy management, and environmental planning, especially in remote areas or regions where access to reliable climate data is difficult. The project has the potential to be developed for integration with smart forecasting systems or automated environmental systems in the future.

Chapter One: Introduction

1.1 The Importance of Smart Weather Stations

Smart weather stations represent an advanced solution in environmental research, offering accurate climate data across various sectors such as agriculture, renewable energy, transportation, and urban safety. By collecting and processing atmospheric data directly on-site, these systems provide real-time metrics to planners and emergency responders, improving preparedness in the face of growing challenges posed by global warming and accelerating climate change.

1.2 Advances in Meteorological Technology

After the initial observations, which involved sitting on two heat-sensitive glass panes and using wind speed sensors, today digital devices, interactive microcontrollers, and miniature fire gauges are employed. Wi-Fi technology and data analysis lines have paved the way for inexpensive, self-recovering meteorological trails covering vast areas, allowing users to transmit their readings for analysis. However, the problem is that many northern regions and some industrial areas remain beyond the range of reliable wireless connectivity. This lack of internet access is the focus of our project, which utilizes power lines for data transmission — and this is where its value lies.

Chapter Two: Project Background and Objectives

1. Project Background

The main idea of the project is to design a smart weather station that can collect real-time climate data such as temperature, humidity, wind speed, and atmospheric pressure, then transmit it to display and analysis interfaces using two methods:

• **The first method**: wireless communication (Wi-Fi or LoRa), which is the traditional method.

• **The second and unique method**: data transmission via power lines (Power Line Communication (PLC)).

1. What makes a PLC an innovative idea?

- Relying on the existing electrical grid without the need to establish new communication networks.
- High flexibility in areas lacking Wi-Fi or LoRa coverage.
- • Low cost because the infrastructure is already in place.
- Accessibility to remote locations within agricultural or industrial facilities without the need to dig or extend networks.

2. Relying on the existing electrical grid without the need to establish new communication networks.

• Explanation:

In traditional systems, providing data communication requires laying special cables (such as Ethernet or fiber optic cables) or relying on wireless networks such as Wi-Fi. In a PLC, the existing electrical grid (the wires in walls used to distribute electricity) is exploited to transmit data over the electrical current itself.

• **Advantage:**

This eliminates the need for any new installation costs or architectural modifications, making the system quick to deploy and inexpensive.

3. High flexibility in areas lacking Wi-Fi coverage

- **Explanation:**

In areas where wireless signals such as Wi-Fi technologies are difficult to find (such as basements, metal factories, or mountainous areas), wireless communications are inefficient or unstable .

- **Benefit:**

PLC technology is not significantly affected by environmental conditions because it relies on fixed electrical cables, giving it high operational flexibility in places where other methods fail.

4. Low cost because the infrastructure is already in place

• **Explanation:**

Establishing a traditional communications network requires:

- Network cables
- Signal boosters
- Distribution devices

In the case of PLC, the basic infrastructure—the electrical grid—is already in place, requiring only PLC converters (small devices that plug into electrical outlets and send and receive data).

• **Benefit:**

This significantly reduces installation and maintenance costs, making the

technology suitable for projects with limited budgets or large-scale use.

Summary :

PLC technology combines the benefits of existing technologies, reducing costs, and providing connectivity in challenging environments. This makes it a smart and innovative choice for Internet of Things (IoT) projects, monitoring systems, industrial control, and even home networks.

PLC technology is not significantly affected by environmental conditions because it relies on fixed electrical cables, giving it high operational flexibility in places where other methods fail.

- In many industrial or agricultural facilities, environmental conditions may be unsuitable for the use of wireless communication technologies such as Wi-Fi, whether due to metal insulators, large spaces, or electromagnetic interference.
- What distinguishes Power Line Communication (PLC) technology is its reliance on existing power lines, which are fixed and extend to various locations within the facility.
- This makes it possible to transfer data between different devices easily and stably, without the need to install new cables or use radio waves, which are affected by environmental factors such as thick walls or long distances.

2.2 Project Objectives

- To build a weather station capable of operating autonomously without direct human intervention.
- To integrate Wi-Fi and PLC technologies for data transmission.
- To intelligently analyze data to produce weather reports and alerts.
- To design a simple user interface for visualizing data.
- To demonstrate the feasibility of using PLCs as an alternative or complement to wireless communication.

2.3 Project Significance

- The project provides an effective technical solution to the problem of weak infrastructure in non-urban areas.
- To demonstrate how traditional electrical grids can be repurposed for modern digital purposes without major modifications.
- To serve the smart agriculture, energy, and environment sectors.
- To open the way for the use of PLCs in other future applications, such as smart irrigation or industrial monitoring.

2.4 Project Scope

The project includes:

1. Using sensors to measure (temperature, pressure, wind speed and direction, rain sensor, light sensor, gas sensor)

• A set of sensors is installed in a specific location, such as a greenhouse or an open area.

• These sensors periodically measure weather conditions.

2. Designing an integrated electronic circuit that includes an Arduino Mega 2560 microcontroller.

• All sensors are assembled and connected to an Arduino Mega 2560 control board.

- This module is responsible for reading data from the sensors, processing it, and preparing it for transmission.
- It is powered either by a direct electrical source or by solar energy in the event of a power outage**.**

3. Sending data via Wi-Fi. If Wi-Fi is unavailable, it is sent via power lines (PLC).

• Normal mode: Data is sent directly to the server or user interface via Wi-Fi.

4. Receive and analyze data on an electronic platform (web interface or simple application)

• A simple interface (dashboard) is programmed to display the values received from the sensors live.

• The values can be displayed on a screen, a web page, or a local mobile application**.**

2.5 Project Feasibility

1. Cost Reduction

- No need to extend network cables or install Wi-Fi towers.
- Utilizing the existing electrical grid means virtually free infrastructure.

2. Operational Capability in Complex Environments

• Works in greenhouses or metal structures that impede Wi-Fi.

• Suitable for farms, warehouses, or factories.

3. Enhanced Reliability

- The hybrid system (Wi-Fi + PLC) ensures continuous connectivity even if one of the two paths fails.

4. Scalability

- The system can later be integrated with automated irrigation or cooling systems.
- Or linked to other smart platforms such as Firebase or Home Assistant.

5. Proof of Concept

- The project does not aim to market an off-the-shelf product, but rather demonstrates the feasibility of operating a smart system at a low cost and with high efficiency, an important step in research and development projects.

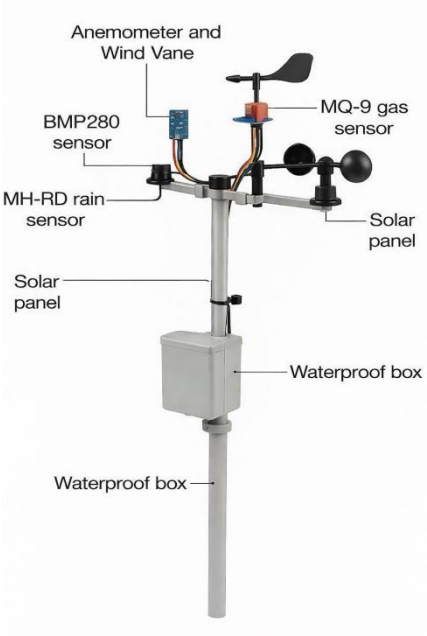


Figure (2.a): Schematic diagram of the components of a smart weather station

Chapter Three: Traditional Weather Stations and Previous Studies

3.1 Types of Traditional Weather Stations

First: Manual Weather Stations

Rely on readings from traditional measuring devices such as thermometers, mercury gauges, and handheld barometers. They require direct human intervention to record data, leading to:

- Delayed processing.
- Potential human error.
- Limited real-time data updates.

Second: Wired Automatic Stations

Use digital sensors connected to a storage unit or central processor via wires. These stations are more accurate than manual stations, but suffer from:

- Difficulty in installation in open environments.
- Restricted movement due to wires.
- High installation cost.

Third: Wireless Weather Stations

Use transmission modules (such as Wi-Fi, Zigbee) to transmit data from sensors to a server or central system. Their advantages:

- Easy to install and expand.
- Low power .
- Suitable for farms and open spaces. However, it:
- Is affected by signal quality.
- May be ineffective in areas with poor coverage or crowded frequencies

3.2 Similar Projects

1. Wi-Fi-only Stations

Several smart weather projects have been developed that rely on ESP8266 or ESP32 modules to send data to cloud servers via Wi-Fi. However, they fail to work in areas without stable internet coverage.

2. LoRa (Long Range) Stations

LoRa (Long Range) has proven its efficiency in transmitting environmental data over long distances using very low power, but they:

- Require a central receiving unit (gateway).
- Susceptible to interference in urban environments.
- Require a frequency license in some countries.

3. GSM/3G/4G Models

Rely on mobile networks, but their cost is higher due to data subscriptions and they consume more power.

Similar Projects from Universities and Companies

1. Project from TU Delft University – Netherlands

Title: "Hybrid Communication System for Smart Greenhouses Using PLC and Wireless Sensor Networks"

- Concept: Develop a smart monitoring system within a greenhouse using sensors to measure temperature and humidity. Data is initially transmitted via Wi-Fi, and upon failure, it is automatically switched to PLC technology.
- Goal: Provide continuous and stable connectivity in greenhouses that often block wireless signals.
- Result: The system achieved a success rate of over 92% in terms of uptime compared to systems that rely solely on Wi-Fi.

2. Siemens – PLC Solution for Industrial Systems

Project: "SIMATIC Powerline Communication Module"

Goal: Use PLCs in industrial environments to control machines in areas that are difficult to access via Wi-Fi or Ethernet.

Result: The company reduced the need for new communication cables by 70% and increased operational efficiency in large factories.

3. Research Project – University of Malaya, Malaysia Title: "IoT-based Smart Farming using Arduino and PLC Communication"

- Objective: To design a smart agricultural system that uses sensors connected to an Arduino, sending data via a PLC to a central server.
- Results: The system successfully reduced water consumption by 25% by monitoring soil and weather.
- Advantages of the study: The use of inexpensive modules such as the Arduino, making the system suitable for small and medium-sized farms.

4. Project from Devolo – Germany

Project: "Smart Home over Powerline"

- Technology : Smart home systems that rely entirely on a PLC to transmit the internet and control lighting and appliances using only electricity.
- Objective : To overcome the problems of thick walls or weak Wi-Fi signals in homes.
- Advantages : Easy installation, with no additional wiring required.

5. Research Project – Cairo University (Faculty of Engineering)

Title: "Design of a Smart Environmental Monitoring System using ESP32 and Hybrid Communication (Wi-Fi & Powerline)"

- Objective: A system for monitoring air quality and temperature in crowded industrial areas.
- Technologies used: ESP32 + sensors + PLC modules.
- Results: The project succeeded in providing real-time monitoring and warnings when pollutants exceed permissible limits.

3.3 The Gap Addressed by the Project

Despite the advancement of weather systems, there remains a clear gap in areas with weak communication network infrastructure. This gap includes:

• The inability to transmit data in real-time.

• The lack of wireless communications in some areas.

Here comes the unique solution: Power Line Communication (PLC).

Power Line Communication (PLC) technology is not common in weather stations, but it represents an innovative solution to address this gap, allowing:

• Using the existing electrical grid as a digital communication channel.

• Delivering data from remote points to processing units without the need for a

• ny additional communications infrastructure.

• Expanding the station without digging or extending additional networks.

CHAPTER FOUR: SYSTEM COMPONENTS AND PROJECT IMPLEMENTATION METHODOLOGY

4.1 SYSTEM OVERVIEW

The system designed in this project consists of a set of functionally interconnected modules that work together to collect, process, and transmit climate data using two technologies:

- Wireless communication (Wi-Fi) .
- Data transmission over power lines (PLC), the technology that distinguishes this project.

This combination represents a flexible and smart solution to address communication problems in areas with weak or unstable infrastructure.

The power source used in the project

" (220V main power source, **Adapter** SMPS)"

TYPES OF POWER SUPPLIES USED:

Power supply components in a smart weather station

3. The basic components of a dual energy system

Power System Components and Their Functions

Source	Details
Main Power Supply 220V	An electrical power source used as the primary energy supply, typically connected through a transformer to a voltage regulator.
Adapter SMPS	An electrical adapter that converts voltage from 220V to or 3.3V or 12V or 5V or

Power Sources Used in the System

1. Main Power Supply (220V AC)

Description:
The **220V AC main power supply** is considered the primary energy source for operating the smart weather station when electricity is available. This source is typically connected through a **transformer or power supply unit**, where the voltage is stepped down and regulated using a **voltage regulator** to provide a stable and safe DC voltage suitable for electronic components such as microcontrollers and sensors.

Advantages:

- Provides a stable and reliable power source for continuous operation.
- Suitable for long-term use without the need for frequent recharging.
- Reduces dependence on batteries when grid electricity is available.
- Ideal for urban areas or locations close to electrical infrastructure.

---

2. SMPS Adapter (Switch Mode Power Supply)

Description:
An **SMPS (Switch Mode Power Supply) adapter** is an advanced electronic power converter that transforms **220V AC** into low - voltage **DC outputs such as 12V, 5V, or 3.3V**, which are required to power sensitive electronic devices like **Arduino boards, Raspberry Pi, communication modules, and sensors**.
This type of power supply operates using high-frequency switching techniques, resulting in higher efficiency and lower power losses compared to traditional linear power supplies.

Advantages:

- High energy conversion efficiency with minimal heat loss.
- Compact size and lightweight design.
- Provides stable and regulated output voltage, protecting sensitive electronics.

- Supports multiple voltage levels (3.3V, 5V, 12V) depending on system requirements.
- Well-suited for smart systems and embedded electronic projects.

Importance of Using Both Power Sources in the Project

Combining the **220V main power supply** with an **SMPS adapter** ensures:

- Reliable and stable system operation.
- Protection of electronic components from voltage fluctuations.
- Flexibility in supplying different system modules with appropriate voltage levels.

This integrated power approach enhances the overall reliability of the smart weather station, especially when combined with **solar power and battery backup systems** to maintain operation during power outages.

Figure (4.a): Model of a smart weather station

4.2 Hardware Components

Components used in the Smart Weather Station project

Type	Component Used
Controller Unit	Arduino Mega 2560 – Responsible for reading sensors, processing, and sending to communication units.
Input Units	Sensor set includes: • Temperature Sensor • Pressure Sensor • Humidity Sensor (BME280) • Rain Sensor (MH-RD) • Gas Sensor (MQ-9) • Wind and Direction Sensor • Light Meter Sensor
Display Interface	LCD Screen (1280×400) to display data + Raspberry Pi Zero W for a local graphical interface.
Communication Units	• Wi-Fi Module (e.g., NRF24L01) • PLC Module for data transmission over power lines.
Voltage Regulator	• LM2596-5V Voltage Regulator • XL4016DC –DC Buck converter
Power Supply	An electrical adapter that converts voltage from 220V to or 3.3V or 12V or 5V
"4-20mA to Voltage Converter Module XY-ITOV	XY-ITOV-4 CConverting the 4–20mA current signal coming out of the wind speed sensor into a voltage signal (e.g., 0–5V) that can be easily read by an Arduino or any other microcontroller.onverting the 4–20mA current signal coming out of the wind speed sensor into a voltage signal (0–5V) that can be

4.3 System Components and Their Functions

System components and their functions – Smart Weather Station

Component	Function
Arduino Mega 2560	Central control unit that reads sensor signals and sends them to the communication unit.
Raspberry Pi Zero W	Supports advanced processing and runs a graphical interface to display data locally.
Light Meter Sensor	Measures light intensity (Lux), used to evaluate sunlight or agricultural lighting.
Wind Sensor	Measures wind speed and direction via analog signals.
BME680 Sensor	Integrates high-precision atmospheric pressure, temperature, and humidity measurement.
Rain Sensor	Detects the presence of rain and is used for alerts or automatic control.
Air Quality Sensor	Monitors pollutants like CO or PM2.5 to assess ambient air quality.
Wi-Fi Module (NRF24L01)	Transmits data wirelessly to the server or user interface.
PLC Module (Power Line Communication)	Transmits data over power lines when wireless coverage is unavailable.
LCD Screen	Used to display weather data locally in real time.
5V LM2596 Voltage Regulator	It reduces the voltage from 12 volts to 10 volts to operate the solar radiation sensor.
XL4016DC –DC Buck converter	Voltage regulating units use high voltage (e.g., 24V or 12V) to reduce voltage (e.g., 9V or 5V) to power electronic project components such as Arduinos, PLCs, or communication modules. They use 12V to 10V to power PLC.
Power Supply	Used to step down voltage from 220V to 12V and convert from AC to DC .

First: A detailed explanation of the hardware components used.

1.Arduino Mega 2560

• Function: Main control unit for sensors.

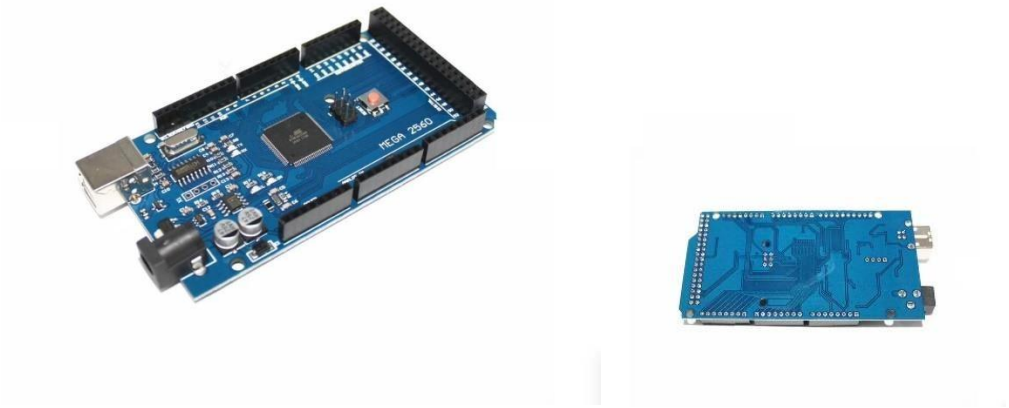
• Features: Large number of I/O pins and more memory than the Arduino Uno.

• Programming: Arduino IDE (C++) to collect data from sensors and transmit it wirelessly or over power lines.

• Operating voltage:

VIN input = 7–12V

Powered by a voltage regulator or 9V/12V source



Figuer(4.B): Arduino Mega 2560

2. Arduino Uno

- **Function:**
  The **Arduino Uno** is used as a **secondary control unit** in the system. It receives data from the main controller (Arduino Mega) or from communication modules, processes this data, and forwards it through **wireless communication modules** or **Power Line Communication (PLC)** units according to the system design.
- **Features:**
  - Provides a moderate number of **digital and analog I/O pins**, suitable for controlling peripheral devices.
  - Easy to program and widely supported, making it ideal for distributed or modular system architectures.
  - Lower power consumption compared to the Arduino Mega.
  - Well suited for handling dedicated tasks such as communication management or local control operations.
- **Programming:**
  The Arduino Uno is programmed using the **Arduino IDE** with **C++**, where the firmware is developed to receive data from the Arduino Mega or attached modules and transmit it via **wireless** or **PLC communication technologies**.
- **Operating Voltage:**
  - Standard operating voltage: **5V**.
  - **VIN input: 7–12V**.
  - Can be powered via:
    - An external voltage regulator,
    - A **9V or 12V** power source, or
    - The **USB port (5V)**.

Figuer(4.B2): Arduino Mega 2560

3.**Raspberry Pi Zero W:**

- Function: Central processing unit, used to display and analyze data on a graphical interface.
- Features: Wi-Fi support, small size, and Linux operating system .
- Programming: Python – used to display data on a website or graphical dashboard.
- Typical operating voltage: 5V via Micro-USB port

Figuer(4.C): Raspberry Pi Zero W

4.BME680 Sensor (Pressure & Altitude)

Function:

• Measures atmospheric pressure and altitude.

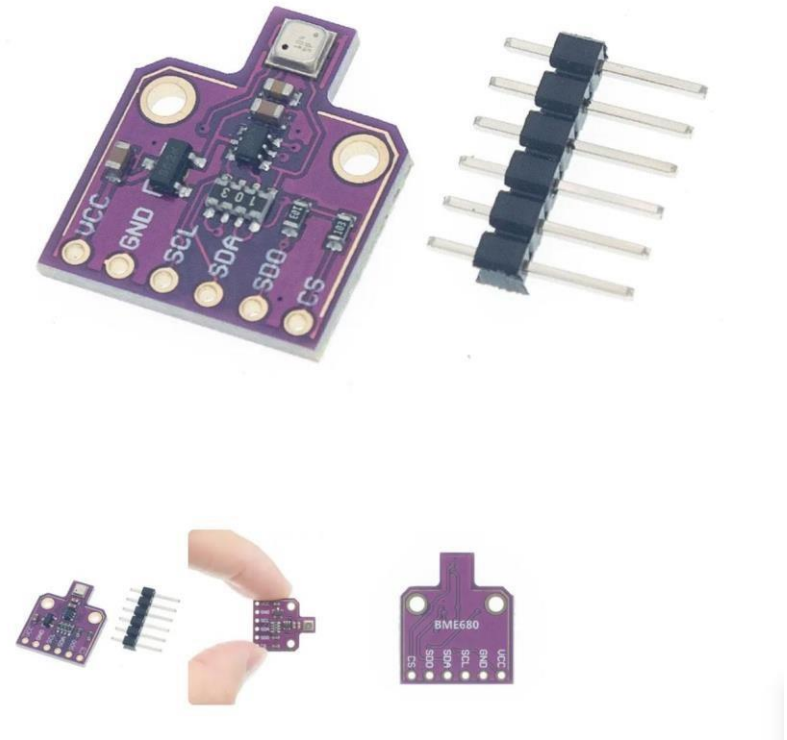
• Measures temperature and humidity.

• Accuracy: More accurate than the DHT11.

• Programming: DHT library for Arduino.

• Operating voltage: 5V.

• Communication: I2C or SPI.



Figuer(4.D): BME680 Sensor (Pressure & Altitude)

5. MH-RD Rain drop Sensor Module

- Function: Rain sensor.
- Programming: Reading the analog value and converting it to a rain/no rain state.
- Operating voltage: 5V

Figuer(4.E): MH-RD Rain drop Sensor Module

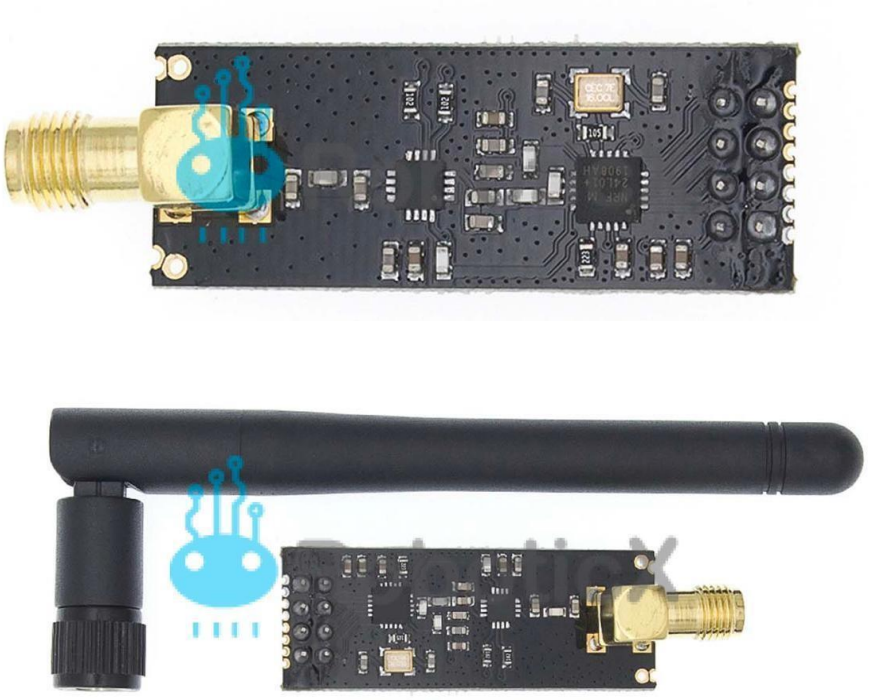
6.**Wind Speed and Direction Sensors\ Anemometer and WindVane.**

- Function: Measuring wind speed and direction.
- Type: Typically analog or digital sensors based on magnetic pulses .
- Programming: Calculating the time between pulses or reading the voltage.
- Operating voltage: 5V

Figuer(4.F): Wind Speed and Direction Sensors/ Anemometer and WindVan

7.WIRELESS COMMUNICATION MODULE - 2.4GHz NRF24L01+PA+LNA SMA Wireless Transceiver Antenna

- Function: Send data over long distances using low-power radio waves.
- Communication: SPI.
- Programming: Using the LoRa library in Arduino.
- Operating voltage: 5V



Figuer(4.G): WIRELESS COMMUNICATION MODULE - 2.4GHz NRF24L01+PA+LNA
SMA Wireless Transceiver Antenna

8.Power Line Communication (PLC) Module

- Function: Transfers data over power lines.
- Connectivity: Integrates with Arduino or Raspberry Pi and uses protocols such as HomePlug or X10.
- Programming: Depends on the type of PLC used, and data is often exchanged via UART or SPI.
- Operating voltage: 12V, often operated directly from a 12V supply.

Figuer(4.H): Power Line Communication (PLC) Module

9. LCD screen

- Function: Displays data directly to the user.
- Programming: Using the Adafruit or LiquidCrystal libraries
- Operating voltage: 5V

Figuer(4.I): LCD screen

10.MQ-9 – Gas Sensor

Module Its primary function:

Detecting flammable and toxic gases, especially:

- Carbon monoxide (CO)
- Methane (CH_4)
- Butane and other flammable gases

How does it work?

The sensor contains an internal heating element and chemicals that react with gases.

When gas is present, the element's resistance changes, producing a voltage difference that is sent to the Arduino.

The programming language for the MQ-9 sensor when used with Arduino is:

C++ via the Arduino IDE

The sensor is programmed using the Arduino IDE, which is based on C++.

- Operating voltage: 5V

Figuer(4.J): MQ-9 – Gas Sensor Module

11.Light Meter Sensor

Light Meter Sensor

Main Function:

- Measures luminous intensity or ambient brightness.
- Provides a digital reading expressing the amount of light reaching the sensor, which can be correlated with solar radiation levels.
- Operating Voltage: 5V

Figuer(4.K): Light Meter Sensor

12.Power Regulation Module Description

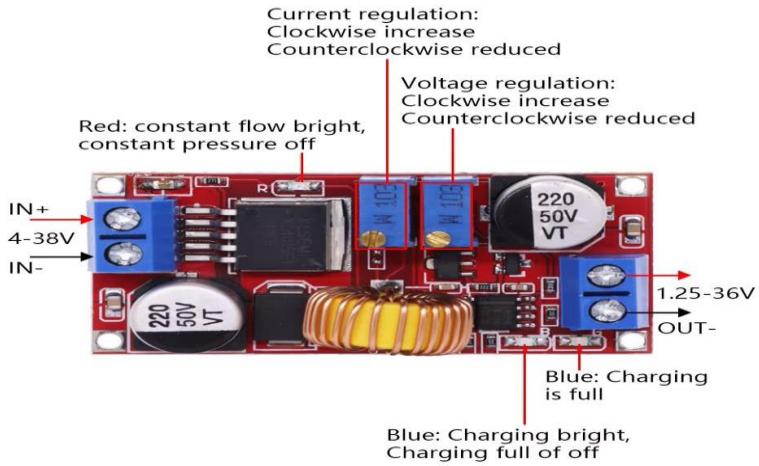
To ensure safe and stable power delivery to the **PLC Power Line Communication module**, a **DC-DC Buck Converter (XL4016)** is used in the system. This module steps down the input voltage from **12V DC (from the main power source)** to a regulated **10V DC**, which matches the operating requirements of the PLC communication circuit.

Module Details:

- **Type:** XL4016 DC-DC Step-Down Buck Converter
- **Input Voltage:** 12V DC
- **Output Voltage:** Adjusted to 10V DC
- **Max Output Current:** Up to 5A (with proper heat dissipation)
- **Control:**
  - **Voltage Regulation Potentiometer:** Used to precisely set the output to 10V.
  - **Current Limiting Potentiometer:** Optional, used to protect the PLC in case of overload.

Function in the System:

This buck converter is placed **between the 12V power supply and the PLC module**, acting as a protective and stabilizing intermediary. It ensures that any voltage fluctuations from the input source do not damage the PLC or cause data transmission errors.



- Figuer(4.M): XL4016 DC-DC Step-Down Buck Converter

13.Voltage Regulation Module for Solar Radiation Sensor

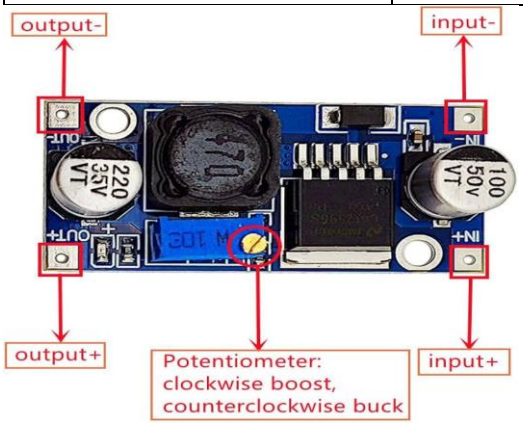
To provide a stable and safe power supply to the **Solar Radiation Sensor**, a **LM2596 Buck Converter** was used in the system. This is a reliable and widely used DC-DC step-down converter, capable of efficiently reducing voltage from 12V to the required 10V for the sensor.

Tchnical Specifications:

Parameter	Value
Type	LM2596 Buck Converter
Input Voltage (IN+ / IN−)	12V DC
Output Voltage (OUT+ / OUT−)	Adjusted to 10V DC
Maximum Output Current	Up to 2A continuous
Voltage Adjustment	Via onboard potentiometer (blue knob)

Wiring Configuration:

LM2596 Pin	Connected To
IN+	+12V Power Supply
IN−	GND (Ground)
OUT+	VCC of Solar Radiation Sensor
OUT−	GND of Solar Radiation Sensor



Figuer(4.N): LM2596 Buck Converter

15.SMPS Power Supply Unit (ATX)

A **Switched-Mode Power Supply (SMPS)** computer unit—commonly known as an **ATX Power Supply**—was integrated into the project as a **high-performance and reliable DC power source** to support various components in the system.

Key Features:

- **Type:** ATX SMPS (from a desktop computer)
- **Input:** AC 230V / 50–60Hz
- **Output:** Multiple regulated DC voltages:
  - **+12V** (for PLCs, relays, LCD backlights)
  - **+5V** (for Arduino boards and digital sensors)
  - **+3.3V** (for low-voltage logic devices)
- **Total Output Power:** ~250W to 500W (depending on model)
- **Built-in Cooling Fan** for continuous operation

Purpose in the Project:

Provides specific power output for components such as:

• Sensors
• Wireless (Wi-Fi) modules
• Arduino Mega/Uno boards
• PLC power line communication modules
• Supports multiple simultaneous voltage levels from a single module.

• Enables safe, self-powered module operation (with settings 3.3, 5, and 12).

Figuer(4.Nn): SMPS Power Supply Unit (ATX)

15.Voltage Converter XY-ITOV-4

Converting the 4–20mA current signal coming out of the wind speed sensor into a voltage signal (e.g., 0–5V) that can be easily read by an Arduino or any other microcontroller.

Purpose of Using the XY-ITOV Module

Main Function:

To **convert a 4–20mA current signal** (from a wind speed sensor) into a **voltage signal (0–5V or 0–3.3V)** that can be read by an analog input on a microcontroller like **Arduino, Raspberry Pi, ESP32**, etc.

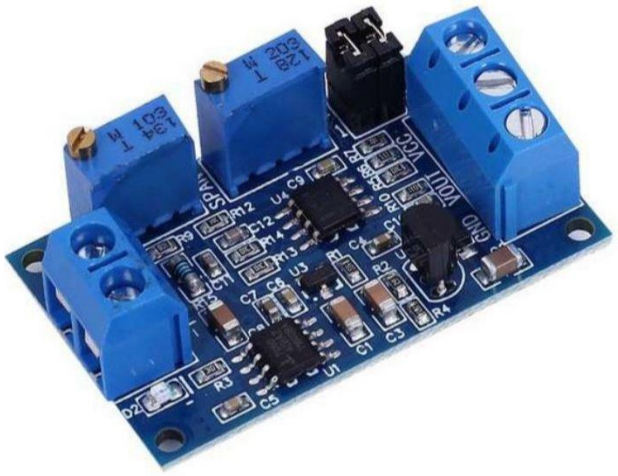
Current-to-Voltage Conversion

Current from Sensor	Voltage Output	Approx. Wind Speed
4 mA	0 V	0 m/s
12 mA	2.5 V	~15 m/s
20 mA	5 V	~30 m/s

The actual mapping from voltage to wind speed depends on the sensor's datasheet.

Wiring Guide

Pin on XY-ITOV	Connect To
IN+ / IN–	4–20mA signal from the sensor
VOUT	Analog pin on Arduino (e.g., A0)
VCC	Power supply (e.g., 12V or 24V depending on the board)
GND	Shared ground with Arduino



Figuer(5.Ns): XY-ITOV-4

Figuer(6.P) :LCD

4.4 Software Components

- Programming the Arduino module using C/C++ and the Arduino IDE
  . • Using Python or Node-RED on a Raspberry Pi to process and display data.
- A dashboard displays real-time data.
- Custom PLC software to convert digital signals into signals transmitted over power lines.

1. Programming an Arduino module using C/C++ and the Arduino IDE Objective:

Programming the microcontroller (Arduino Mega for example) to read and process sensor data, then sending it via LoRa or PLC

Tools:

- Programming language: C/C++ Development environment: Arduino IDE Required libraries:
- DHT.h for reading the DHT22
- Adafruit_BMP280.h for compression
- RTClib.h for time
- SD.h for storage
- LoRa.h for transmission

Tasks performed by the code:

- (Collecting data from sensors (temperature, humidity, pressure).
- Calculating values or formatting them as text (String or CSV).)
- Sending them to the receiving unit.
- (Storing them on an SD card (optional)

2. Using Python or Node-RED on Raspberry Pi to process and display data

Objective:

Analyze data coming from an Arduino (via LoRa or PLC) and display it on a graphical display or website.

Option 1: Python

Tasks:

- Receive data via a Serial or USB port.
- Save it to a CSV file or SQLite/MySQL database.
- Display it using Flask (a simple web interface) or Matplotlib to plot graphs**.**

Option 2: Node-RED

Features:

A visual drag-and-drop environment.

Easy to connect inputs and outputs, such as:

- Reading from Serial or MQTT. Display on a dashboard.
- Sending notifications.

Tasks:

- Direct data processing.
- Displaying it on a built-in dashboard interface.
- Exporting data as a file or to a web service.

3. Dashboards displaying real-time data

Objective:

Visually display the values coming from sensors to the end user in realtime.

Implementation tools:

- A simple and fast Node-RED Dashboard
- A Python Flask interface for greater customization, or even a platform like ThingSpeak or Blynk.

Displayed Content:

- Graphs of temperature, humidity, and pressure...
- Time-lapse log.
- LED indicators when an alert occurs (such as rain or strong winds).
- Local map/radar, if available.

4. PLC Software Dedicated to Converting Digital Signals into Power Line Signals Purpose:

Using Power Line Communication (PLC) technologies to transmit data from the Arduino to a Raspberry Pi or other system over power lines instead of a
wireless network.

How it Works:

- A PLC module, such as a modem or PLC module, is used.
- Digital data is converted into signals that are transmitted over power lines.
- A receiver, such as a Raspberry Pi or another PLC module, captures and decodes these signals.

PLC Software:

- Relies on a protocol such as HomePlug, X10, or Power Line UART.
- Often, libraries or firmware are used on a dedicated microcontroller or external module (PLC TX/RX).
- The Arduino interfaces with the PLC via Serial or SPI.

4.4Smart Weather Station Components Software Programming

Smart Weather Station – Arduino Mega Sensor Interface Code

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME680.h>

Adafruit_BME680 bme;

// ===== Pins =====
#define WIND_SPEED_PIN A0
#define SOLAR_PIN      A1
#define WIND_DIR_PIN   A2
#define MQ9_PIN        A3
#define RAIN_PIN       A4

void setup() {
  Serial.begin(9600);
  delay(1000);

  if (!bme.begin()) {
    Serial.println("ERROR=BME680");
    while (1);
  }

  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setGasHeater(320, 150);

  Serial.println("MEGA_READY");
}
```

```
void loop() {

 if (!bme.performReading()) {
  Serial.println("ERROR=READ_BME680");
  delay(1000);
  return;
 }

 // ===== Read Sensors =====
 int windSpeedRaw = analogRead(WIND_SPEED_PIN);
 int solarRaw     = analogRead(SOLAR_PIN);
 int windDirRaw   = analogRead(WIND_DIR_PIN);
 int mq9          = analogRead(MQ9_PIN);
 int rainRaw      = analogRead(RAIN_PIN);

 // ===== Processing =====
 float T = bme.temperature;
 float H = bme.humidity;
 float P = bme.pressure / 100.0;

 // Rain: 0 = لا مطر ، 1 = مطر
 int R = (rainRaw > 500) ? 1 : 0;

 // Wind Direction 0–360°
 float W = map(windDirRaw, 0, 1023, 0, 360);

 // Wind Speed (m/s) – تحويل تقريبي
 float UV = (windSpeedRaw * 5.0 / 1023.0) * 2.0;

 // Solar Radiation
 int L = solarRaw;

 // ===== Serial Monitor =====
 Serial.println("----- Sensor Readings -----");
 Serial.print("Temp: "); Serial.print(T); Serial.println(" C");
```

```
  Serial.print("Hum : "); Serial.print(H); Serial.println(" %");
  Serial.print("Pres: "); Serial.print(P); Serial.println(" hPa");
  Serial.print("Wind Dir: "); Serial.print(W); Serial.println(" deg");
  Serial.print("Wind Speed: "); Serial.print(UV); Serial.println(" m/s");
  Serial.print("Solar: "); Serial.println(L);
  Serial.print("Gas MQ9: "); Serial.println(mq9);
  Serial.print("Rain: "); Serial.println(R);
  Serial.println("--------------------------");

  // ===== Send to Raspberry Pi =====
  Serial.print("T=");   Serial.print(T, 1);   Serial.print(",");
  Serial.print("H=");   Serial.print(H, 1);   Serial.print(",");
  Serial.print("P=");   Serial.print(P, 0);   Serial.print(",");
  Serial.print("W=");   Serial.print(W);      Serial.print(",");
  Serial.print("R=");   Serial.print(R);      Serial.print(",");
  Serial.print("L=");   Serial.print(L);      Serial.print(",");
  Serial.print("CO2="); Serial.print(mq9);    Serial.print(",");
  Serial.print("UV=");  Serial.println(UV, 2);

  delay(1000);
}
```

#	Code Line / Section	Function / Description
1	#include <Wire.h>	Includes I2C communication library (used by BME680).
2	#include <Adafruit_Sensor.h>	Adafruit's base sensor library.
3	#include <Adafruit_BME680.h>	Library specific for the BME680 environmental sensor.
4	Adafruit_BME680 bme;	Creates a sensor object named bme.
5	#define WIND_SPEED_PIN A0	Assigns analog pin A0 for wind speed sensor.
6	#define SOLAR_PIN A1	Assigns analog pin A1 for solar radiation sensor.
7	#define WIND_DIR_PIN A2	Assigns analog pin A2 for wind direction sensor.
8	#define MQ9_PIN A3	Assigns analog pin A3 for gas sensor (MQ-9).
9	#define RAIN_PIN A4	Assigns analog pin A4 for rain sensor.
10	Serial.begin(9600);	Starts USB serial communication at 9600 baud.
11	if (!bme.begin()) { ... }	Checks if BME680 is connected; if not, prints error and halts.
12–15	bme.setTemperatureOversampling(...) etc.	Sets the oversampling settings for temperature, humidity, pressure, and gas heater config.
16	analogRead(...)	Reads analog values (range 0–1023) from the sensors connected to A0–A4.
17	map(..., 0, 1023, 0, 360)	Converts wind direction from analog value to degrees (0°–360°).
18	float UV = ...	Approximates wind speed in m/s from analog voltage.
19	int R = (rainRaw > 500) ? 1 : 0;	Sets rain status: 1 = no rain, 0 = rain detected (based on analog value).
20	Serial.print(...)	Prints readable sensor data to Serial Monitor.
21	Serial.print("T=...")	Sends compact sensor readings in CSV format for transmission to Uno/Raspberry Pi.
22	delay(1000);	Waits 1 second before next loop cycle .

Arduino UNO code #1 (Receiver from Mega → Sender nRF24

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8);
const byte address[6] = "NODE1";

void setup() {
  Serial.begin(9600);   // من Mega
  radio.begin();
  radio.setPALevel(RF24_PA_LOW);
  radio.setChannel(100);
  radio.openWritingPipe(address);
  radio.stopListening();
}

void loop() {
  if (Serial.available()) {
    String data = Serial.readStringUntil('\n');
    data.trim();

    radio.write(data.c_str(), data.length() + 1);
  }
}
```

nRF24L01 Transmitter Code – Explanation Table

Line	Code / Section	Purpose
1	#include <SPI.h>	Includes SPI library (needed for nRF24L01 communication).
2	#include <nRF24L01.h>	Includes low-level nRF24L01 definitions.
3	#include <RF24.h>	Includes high-level RF24 library for easy control.
4	RF24 radio(7, 8);	Initializes radio module with CE pin on 7 and CSN pin on 8.
5	const byte address[6] = "NODE1";	Sets a unique 5-character pipe address for wireless communication.
6	Serial.begin(9600);	Starts serial connection to receive sensor data (from Mega).
7	radio.begin();	Initializes the radio module.
8	radio.setPALevel(RF24_PA_LOW);	Sets Power Amplifier level to LOW (to reduce power usage/interference).
9	radio.setChannel(100);	Sets the communication channel (0–125). Must match receiver's channel.
10	radio.openWritingPipe(address);	Opens a sending pipe with the given address.
11	radio.stopListening();	Puts radio in transmit (TX) mode.
12–17	if (Serial.available()) { ... }	When data arrives via USB (e.g., from Arduino Mega), read and send it.
13	String data = Serial.readStringUntil('\n');	Reads a line of incoming data until newline.
14	data.trim();	Removes any whitespace (e.g., \r, extra spaces).
15	radio.write(data.c_str(), data.length() + 1);	Sends the string via RF to the receiver Arduino.

What This Code Does

- Acts as RF transmitter for sensor data sent over USB Serial.
- When data like:
  T=24.5,H=40.2,P=1005,...
  arrives via Serial, it sends it via nRF24L01 module over wireless.
- You can use this on Arduino Mega connected to sensors.
- The receiver Arduino Uno (with nRF24L01) will have a
  complementary receiver code to receive and display the data.

nRF24L01 Wireless Receiver Code

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8);
const byte address[6] = "NODE1";

char buffer[32];

void setup() {
  Serial.begin(9600);   // USB → Raspberry Pi
  radio.begin();
  radio.setPALevel(RF24_PA_LOW);
  radio.setChannel(100);
  radio.openReadingPipe(1, address);
  radio.startListening();
}

void loop() {
  if (radio.available()) {
    radio.read(buffer, sizeof(buffer));
    Serial.println(buffer);   // يذهب مباشرة للـ Raspberry Pi
  }
}
```

nRF24L01 Wireless Receiver Code – Explanation Table

Line of Code	Description / Function
#include <SPI.h>	Includes the SPI communication library needed for nRF24L01 module.
#include <nRF24L01.h>	Includes definitions specific to the nRF24L01 radio module.
#include <RF24.h>	Loads the RF24 library for managing the radio module easily.
RF24 radio(7, 8);	Creates a radio object using CE pin = 7 and CSN pin = 8.
const byte address[6] = "NODE1";	Sets the wireless communication address (must match the sender's address).
char buffer[32];	Creates a character array (buffer) to store incoming data (max 32 bytes for nRF24L01).
Serial.begin(9600);	Initializes serial communication with the Raspberry Pi or computer.
radio.begin();	Initializes the nRF24L01 module.
radio.setPALevel(RF24_PA_LOW);	Sets Power Amplifier level to low (to reduce power consumption or avoid interference).
radio.setChannel(100);	Sets communication channel (0–125); both sender and receiver must use the same channel.
radio.openReadingPipe(1, address);	Opens pipe #1 to listen for messages at NODE1 address.
radio.startListening();	Puts the module into receiving mode.
if (radio.available())	Checks if any data has arrived wirelessly.
radio.read(buffer, sizeof(buffer));	Reads the incoming data into the buffer array.
Serial.println(buffer);	Prints received data to Serial Monitor (to be seen on Raspberry Pi, for example).

Purpose of This Code

- This code **receives sensor data wirelessly** from another Arduino using the **nRF24L01+ module**.
- Once received, the data is **sent via USB serial to the Raspberry Pi** (or PC) for logging, display, or further processing.

UNO #1) – Relay Code Between MEGA and UNO #2

```
#include <SoftwareSerial.h>
// من MEGA
SoftwareSerial fromMega(2, 3);
// إلى UNO #2
SoftwareSerial toUNO2(4, 5);
void setup() {
  Serial.begin(9600);
  fromMega.begin(4800);
  toUNO2.begin(4800);
  Serial.println("UNO #1 READY (FORWARD)");
}
void loop() {
  if (fromMega.available()) {
    String data = fromMega.readStringUntil('\n');
    data.trim();
    if (data.length() > 0) {
      Serial.println("From MEGA: " + data);
      toUNO2.println(data);
    }
  }
}
```

UNO #1 – Relay Code Between MEGA and UNO #2

Line of Code	Description / Function
#include <SoftwareSerial.h>	Includes the SoftwareSerial library to allow serial communication on custom digital pins.
SoftwareSerial fromMega(2, 3);	Defines a SoftwareSerial port to receive data from Arduino MEGA: RX = pin 2, TX = pin 3.
SoftwareSerial toUNO2(4, 5);	Defines a second SoftwareSerial port to send data to UNO #2: TX = pin 5, RX = pin 4.
Serial.begin(9600);	Starts the default serial monitor for debugging.
fromMega.begin(4800);	Initializes communication with MEGA at 4800 baud.
toUNO2.begin(4800);	Initializes communication with UNO #2 at 4800 baud.
Serial.println("UNO #1 READY...");	Confirms UNO #1 is ready in the Serial Monitor.
if (fromMega.available())	Checks if data was received from Arduino MEGA.
String data = fromMega.readStringUntil('\n');	Reads the incoming string from MEGA until a newline.
data.trim();	Removes any trailing or leading whitespace from the string.
if (data.length() > 0)	Ensures the string is not empty.
Serial.println("From MEGA: " + data);	Prints received data for debugging.
toUNO2.println(data);	Forwards the same data to UNO #2 through the second SoftwareSerial.

How This Works

- Arduino MEGA sends data via SoftwareSerial on pins 2/3 of UNO #1.
- UNO #1 **receives the data**, then **forwards it** to UNO #2 via another SoftwareSerial port on pins 4/5.
- This forms a **middle communication bridge** between MEGA and UNO #2 — ideal for relaying data over **PLC modules** or similar communication links.

UNO #2 – Final Receiver

```
#include <SoftwareSerial.h>

// من UNO #1
SoftwareSerial link(2, 3);

void setup() {
  Serial.begin(9600);
  link.begin(4800);

  Serial.println("UNO #2 READY (DISPLAY)");
}

void loop() {
  if (link.available()) {
    String msg = link.readStringUntil('\n');
    msg.trim();

    if (msg.length() > 0) {
      Serial.println("DATA: " + msg);
      // هنا لاحقًا بنضيف شاشة LCD
    }
  }
}
```

UNO #2 – Final Receiver (Display Node)

Line of Code	Description / Function
#include <SoftwareSerial.h>	Includes the SoftwareSerial library to allow serial communication on custom digital pins.
SoftwareSerial link(2, 3);	Defines SoftwareSerial port for receiving data from UNO #1 (RX = pin 2, TX = pin 3).
Serial.begin(9600);	Starts communication with Serial Monitor (USB) for debugging or data display.
link.begin(4800);	Initializes SoftwareSerial communication with UNO #1 at 4800 baud.
Serial.println("UNO #2 READY...");	Prints startup message to confirm system is ready.
if (link.available())	Checks if any data was received from UNO #1.
String msg = link.readStringUntil('\n');	Reads full message string until newline (\n).
msg.trim();	Removes any whitespace or extra characters from the beginning/end of the string.
if (msg.length() > 0)	Ensures non-empty message before printing.
Serial.println("DATA: " + msg);	Displays the received message in the Serial Monitor (e.g., from MEGA → UNO1 → UNO2 chain).
LCD	Comment: A reminder that the same message can be shown on an LCD screen later.

Purpose & Flow

- This Arduino UNO #2 acts as the **final receiver**.
- It gets the weather sensor data (or any message) **forwarded by UNO #1**, which was originally sent from **MEGA**.
- This code currently **prints the data to Serial Monitor**, but you can expand it later to:
  - Display on an **LCD (e.g., 16x2 or I2C)**.
  - Send to **Raspberry Pi**.
  - Save to **SD card**, etc.

Operating System of Raspberry Pi in Smart Weather Station

The Raspberry Pi used in the Smart Weather Station operates on a **Linux-based operating system**, specifically **Raspberry Pi OS** (formerly known as Raspbian). This OS is optimized for the Raspberry Pi's hardware and supports Python programming, serial communication, and GUI interfaces like Pygame, making it ideal for receiving and displaying sensor data transmitted from Arduino units via USB, wireless (NRF24L01), or power line communication (PLC).

```
import os

import time

import re

import serial

import pygame

PORT = "/dev/ttyUSB0"

BAUD = 9600

PROJECT_TITLE = "Weather Station Project"

DOCTOR_NAME = "Dr. Omar Tamimi"

STUDENTS_LINE = "Fathi, Ali, Mohammad, Kamel"
```

```
BASE_DIR = os.path.dirname(os.path.abspath(__file__))

ICON_DIR = os.path.join(BASE_DIR, "icons")

SENSORS = [

   {"key": "T",   "name": "Tempreture",              "unit": "C",   "icon": os.path.join(ICON_DIR, "temp.png"),     "default": "25"},

   {"key": "H",   "name": "Humidity",                "unit": "%",   "icon": os.path.join(ICON_DIR, "humidity.png"), "default": "60"},

   {"key": "P",   "name": "Pressure",                "unit": "hPa", "icon": os.path.join(ICON_DIR, "pressure.png"), "default": "1012"},

   {"key": "W",   "name": "Wind Direction",          "unit": "",
  "icon": os.path.join(ICON_DIR, "wind.png"),     "default": "30"},

   {"key": "R",   "name": "Rain",                    "unit": "mm",  "icon": os.path.join(ICON_DIR, "rain.png"),     "default": "0"},

   {"key": "L",   "name": "Solar Radiation Intensity", "unit": "",
"icon": os.path.join(ICON_DIR, "light.png"),    "default": "320"},

   {"key": "CO2", "name": "Gas Sensor",              "unit": "ppm",
"icon": os.path.join(ICON_DIR, "co2.png"),      "default": "450"},

   {"key": "UV",  "name": "Wind Speed",              "unit": "m/s",
"icon": os.path.join(ICON_DIR, "uv.png"),       "default": "0.3"},

]

STALE_SECONDS = 4.0

FPS_DELAY = 0.05

KEYS = sorted([s["key"] for s in SENSORS], key=lambda x: -len(x))


KEY_ALT = "|".join(re.escape(k) for k in KEYS)
```

```
PAIR_RE = re.compile(rf"(?<![A-Za-z0-
  9])({KEY_ALT})\s*[:=]\s*([-+]?\d+(?:\.\d+)?)", re.IGNORECASE)


def safe_load_icon(path, size):
    if not os.path.exists(path):
        return None
    try:
        img = pygame.image.load(path).convert_alpha()
        return pygame.transform.smoothscale(img, size)
    except Exception:
        return None
def fit_font(text, max_w, max_h, bold=False):
    size = max(12, int(max_h))
    while size >= 12:
        f = pygame.font.Font(None, size)
        f.set_bold(bold)
        w, h = f.size(text)
        if w <= max_w and h <= max_h:
            return f
        size -= 1
    f = pygame.font.Font(None, 12)
    f.set_bold(bold)
```

```
    return f
def draw_text_center(screen, text, rect, color, bold=False):
    if not text:
        return
    font = fit_font(text, rect.width - 12, rect.height - 8, bold=bold)
    img = font.render(text, True, color)
    x = rect.x + (rect.width - img.get_width()) // 2
    y = rect.y + (rect.height - img.get_height()) // 2
    screen.blit(img, (x, y))
def draw_multiline_center(screen, lines, rect, color):
    if not lines:
        return
    total_h = rect.height
    line_h = max(16, int(total_h / max(3, len(lines) + 1)))
    fonts = [fit_font(line, rect.width - 16, line_h, bold=True) for line in
  lines]
    rendered = [fonts[i].render(lines[i], True, color) for i in
  range(len(lines))]
    used_h = sum(img.get_height() for img in rendered) +
(len(rendered) - 1) * 6
      y = rect.y + (rect.height - used_h) // 2
       for img in rendered:
          x = rect.x + (rect.width - img.get_width()) // 2
```

```
            screen.blit(img, (x, y))
        y += img.get_height() + 6
def parse_pairs_from_buffer(buf_text):
    matches = list(PAIR_RE.finditer(buf_text))
    data = {}
    for m in matches:
        k = m.group(1).upper()
        v = m.group(2)
        data[k] = v
    last_end = matches[-1].end() if matches else 0
    keep = buf_text[last_end:]
    if len(keep) > 4096:
        keep = keep[-4096:]
    return data, keep
def open_serial():
    try:
        return serial.Serial(PORT, BAUD, timeout=0)
    except Exception:
        return None
def main():
    os.environ.setdefault("SDL_VIDEODRIVER", "x11")
```

```
pygame.init()
screen = pygame.display.set_mode((0, 0), pygame.FULLSCREEN)
pygame.mouse.set_visible(False)
w, h = screen.get_size()
margin = max(10, int(min(w, h) * 0.02))
gap = max(8, int(min(w, h) * 0.015))
header_h = max(100, int(h * 0.19))
card_h = int((h - 2 * margin - header_h - 8 * gap) / 8)
card_h = max(70, card_h)
iconsize = int(min(w * 0.16, card_h * 0.55))
iconsize = max(36, iconsize)
icons = {}
for s in SENSORS:
    icons[s["key"]] = safe_load_icon(s["icon"], (iconsize, iconsize))

defaults = {s["key"]: s["default"] for s in SENSORS}
values = dict(defaults)
last_seen = {k: 0.0 for k in defaults}
ser = open_serial()
buf = ""
running = True
while running:
```

```
        now = time.time()
        for event in pygame.event.get():
            if event.type == pygame.KEYDOWN and event.key ==
pygame.K_ESCAPE:
                running = False
        if ser is None:
            ser = open_serial()
            time.sleep(0.2)
        else:
            try:
                n = ser.in_waiting
                if n and n > 0:
                    chunk = ser.read(n).decode(errors="ignore")
                    if chunk:
                        buf += chunk
                        parsed, buf = parse_pairs_from_buffer(buf)
                        for k, v in parsed.items():
                            if k in values:
                                values[k] = v
                                last_seen[k] = now
            except Exception:
                try:
```

```
                ser.close()
            except Exception:
                pass
            ser = None

    for k in list(values.keys()):
        if last_seen[k] == 0.0:
            values[k] = defaults[k]
        elif now - last_seen[k] > STALE_SECONDS:
            values[k] = defaults[k]
    screen.fill((0, 0, 0))
    header_rect = pygame.Rect(margin, margin, w - 2 * margin,
header_h)
    pygame.draw.rect(screen, (10, 30, 55), header_rect,
border_radius=18)
    pygame.draw.rect(screen, (90, 120, 160), header_rect, 2,
border_radius=18)

    header_lines = [PROJECT_TITLE, DOCTOR_NAME,
STUDENTS_LINE]
    draw_multiline_center(screen, header_lines, header_rect, (255,
255, 255))
    y = margin + header_h + gap
    for s in SENSORS:
```

```
        r = pygame.Rect(margin, y, w - 2 * margin, card_h)
        pygame.draw.rect(screen, (20, 20, 20), r, border_radius=18)
        pygame.draw.rect(screen, (70, 70, 70), r, 2, border_radius=18)

        left_w = int(r.width * 0.22)
        icon_rect = pygame.Rect(r.x + 10, r.y + 10, left_w - 20,
r.height - 20)
        icon = icons.get(s["key"])
        if icon is not None:
            ix = icon_rect.x + (icon_rect.width - icon.get_width()) // 2
            iy = icon_rect.y + (icon_rect.height - icon.get_height()) // 2

            screen.blit(icon, (ix, iy))
        else:
            box = min(icon_rect.width, icon_rect.height)
            bx = icon_rect.x + (icon_rect.width - box) // 2
            by = icon_rect.y + (icon_rect.height - box) // 2
            pygame.draw.rect(screen, (120, 120, 120), (bx, by, box,
box), 2, border_radius=10)
        right = pygame.Rect(r.x + left_w, r.y, r.width - left_w,
r.height)

        name_h = int(right.height * 0.40)
        value_h = right.height - name_h
```

```
            name_rect = pygame.Rect(right.x + 10, right.y + 8, right.width
  - 20, name_h - 10)

            value_rect = pygame.Rect(right.x + 10, right.y + name_h,
right.width - 20, value_h - 10)

            draw_text_center(screen, s["name"], name_rect, (255, 255,
255), bold=True)

            v = values.get(s["key"], defaults[s["key"]])

            unit = s["unit"]

            txt = f"{v} {unit}".strip()

            draw_text_center(screen, txt, value_rect, (255, 255, 255),
  bold=False)

            y += card_h + gap

        pygame.display.flip()

        time.sleep(FPS_DELAY)

    if ser is not None:

        try:

            ser.close()

        except Exception:

            pass

    pygame.quit()

if __name__ == "__main__":

    main()
```

Smart Weather Station Dashboard – Code Breakdown

Section	Purpose	Explanation
1. Imports & Constants	Load libraries and set constants	Uses pygame for GUI, serial for data input, and re for parsing sensor key-value pairs. The COM port is set to /dev/ttyUSB0.
2. Project Metadata	Display header info	Variables like PROJECT_TITLE, DOCTOR_NAME, and STUDENTS_LINE are shown at the top of the screen.
3. Sensor Definitions	Define sensors & icons	Each sensor (Temp, Humidity, Pressure, etc.) has a key, name, unit, icon image, and default value.
4. Icon & Font Functions	GUI helpers	Functions like safe_load_icon, fit_font, and draw_text_center manage image loading and text display scaling.
5. Serial Port Handling	Open serial port	The open_serial() function connects to the Arduino port using pyserial.
6. Data Parsing	Extract data	Uses regular expressions to parse incoming strings like T:25 H:60 and extract sensor data.
7. Pygame Setup	Start full screen GUI	Initializes a fullscreen pygame window, hides the mouse cursor, and defines layout metrics.
8. Main Loop	Core display logic	Continuously reads serial input, parses sensor data, updates values, and redraws the GUI with sensor cards.
9. Display Cards	Show each sensor	Each sensor gets a colored card with icon, name, and latest value (with unit), arranged vertically.
10. Escape Key	Exit program	Pressing ESC will exit the dashboard cleanly and close the serial port.

Notes

- **Icons** must be placed in the icons/ folder and named as in the sensor definitions (e.g., temp.png, rain.png, etc.).
- This code is optimized for a **Raspberry Pi** running **Raspberry Pi OS (Linux)** with a fullscreen HDMI display.
- Sensor data must be sent over serial in format:
  T:25.3 H:60.2 P:1011.9 W:NE R:1.2 L:310 CO2:480 UV:0.5

Arduino Mega Sensor Code: Functional Breakdown

Section	Component / Function	Details
Libraries	Wire.h, Adafruit_Sensor.h, Adafruit_BME680.h	For I2C communication and BME680 sensor support
Object	Adafruit_BME680 bme	Initializes BME680 sensor object
Analog Pins Defined	A0–A4	Used for analog sensor readings: • A0 = Wind Speed • A1 = Solar Radiation • A2 = Wind Direction • A3 = MQ-9 Gas • A4 = Rain Sensor
Serial Begin	Serial.begin(9600)	Communicates sensor data to Raspberry Pi via USB
BME680 Setup	bme.begin()	Initializes sensor; if fails, prints error
	Oversampling settings	Temp = 8x, Humidity = 2x, Pressure = 4x
	Gas Heater	Heats sensor for 150 ms at 320°C
Loop Function	bme.performReading()	Reads temperature, humidity, pressure, gas
Analog Reads	analogRead(pin)	Reads raw values from 5 analog sensors
Data Processing	- Wind Speed: approx. m/s = (raw * 5 / 1023) * 2 - Wind Dir: mapped 0–360° - Rain: 1 = No Rain, 0 = Rain	Converts sensor signals in to physical values
Serial Monitor Output	Serial.print()	Human-readable sensor values for debugging
Data Output Format	Serial.print("T= ...")	Raspberry Pi receives: T=23.5,H=45.2,P=1013,W=180,R=1,L=679,CO2=580,UV=3.62

Sensor Pin Mapping

Sensor	Pin on Arduino Mega	Unit / Format
Wind Speed Sensor	A0	m/s (approx.)
Solar Radiation	A1	Raw ADC (0–1023)
Wind Direction	A2	Degrees (0–360°)
MQ-9 Gas Sensor	A3	Raw analog (0–1023)
Rain Sensor	A4	Digital logic (0 = Rain)
BME680 (I2C)	SDA/SCL (20/21)	Temp, Humidity, Pressure

Code Overview

RF24 radio(7, 8); // CE = D7, CSN = D8

This line in the code defines the two control pins used to connect the **nRF24L01**:

- **CE (Chip Enable)** → Pin **7**
- **CSN (Chip Select Not)** → Pin **8**

The rest of the communication happens through the **SPI hardware pins** of the Arduino Mega.

Wiring Table: nRF24L01 to Arduino

nRF24L01 Pin	Connects to Arduino Mega	Description
VCC	3.3V	⚠ Use only 3.3V
GND	GND	Ground
CE	D7	As defined in the code
CSN	D8	As defined in the code
SCK	D52	SPI Clock
MOSI	D51	SPI Data (Mega to nRF)
MISO	D50	SPI Data (nRF to Mega)
IRQ	Not connected (optional)	Not used in your code

Code Summary (Receiver Side)

RF24 radio(7, 8); // CE = D7, CSN = D8

This means:

- **CE pin** of the nRF24L01 is connected to Arduino **digital pin 7**
- **CSN pin** is connected to Arduino **digital pin 8**

The SPI communication uses the Arduino's hardware SPI pins.

What the Code Does:

- Initializes the **nRF24L01** module
- Listens for wireless messages on **channel 100**
- When data is received, it prints it through the USB serial port (9600 baud) — usually read by a **Raspberry Pi**

nRF24L01 to Arduino Wiring Table

nRF24L01 Pin	Arduino Pin (Uno / Mega)	Purpose
VCC	3.3V	Power Not 5V
GND	GND	Ground
CE	D7	Defined in code
CSN	D8	Defined in code
SCK	Uno: D13 / Mega: D52	SPI Clock
MOSI	Uno: D11 / Mega: D51	SPI Data to nRF24L01
MISO	Uno: D12 / Mega: D50	SPI Data from nRF24L01
IRQ	Not connected (optional)	Interrupt (unused in this code)

Important Notes

Important Notes

- **We used a 100µF capacitor between VCC and GND near the nRF24L01 module. This helps prevent connection instability.**

- **The values for radio.setChannel() and address[] must match on both the transmitter and receiver.**

Code Summary: UNO #1 Forwarding Unit

This Arduino Uno acts as a **middle relay** between the Arduino Mega and Arduino UNO #2 using two software serial ports.

Code Functionality

- Receives serial data from **Arduino Mega**
- Forwards that data to **UNO #2**
- Prints what it receives to the Serial Monitor for debugging

---

SoftwareSerial Configuration

```
SoftwareSerial fromMega(2, 3); // RX, TX from MEGA
SoftwareSerial toUNO2(4, 5);   // RX, TX to UNO #2
```

Name	Arduino Pin	Direction	Purpose
fromMega	D2 (RX), D3 (TX)	Reads from MEGA	Data IN from Arduino Mega
toUNO2	D4 (RX), D5 (TX)	Sends to UNO #2	Data OUT to Arduino UNO 2

Wiring Guide: UNO #1

Connection	UNO #1 Pin	To Device
Receive from Mega (TX from Mega)	D2	Mega TX pin (e.g., TX1)
Transmit to Mega (RX from Mega)	D3	Mega RX pin (optional)
Transmit to UNO #2 (TX to next)	D5	UNO #2 RX pin
Receive from UNO #2 (RX from next)	D4	UNO #2 TX pin (optional)
GND	GND	Shared Ground

Only TX from Mega → RX of UNO #1 and TX from UNO #1 → RX of UNO #2 are required for one-way communication.

Baud Rate Setup

```
fromMega.begin(4800);
toUNO2.begin(4800);
```

- Both serial links use **4800 baud**, which must match the sending and receiving Arduinos.

Output to Monitor

```
Serial.println("From MEGA: " + data);
```

- Displays the forwarded data on the PC's Serial Monitor for verification.

Wiring Details for Arduino UNO #2:

Purpose	UNO #2 Pin	Connected To
RX (Receive) from UNO #1	**D2 (Pin 2)**	TX of UNO #1 (Pin D4)
TX (Transmit) to UNO #1	**D3 (Pin 3)**	RX of UNO #1 (Pin D5)
USB Serial to PC/Raspberry Pi	**USB Port**	For monitoring/logging
LCD (future use)	*Not connected yet*	Will be connected later

Explanation:

- SoftwareSerial link(2, 3); defines:
  - **Pin 2** as RX → receives data from UNO #1
  - **Pin 3** as TX → not actively used in the code
- The code reads messages from the software serial link and prints them to the main serial port (Serial) for display on Serial Monitor or Raspberry Pi.
- There is **no PLC** included in this code. The communication between UNO #1 and UNO #2 is direct through wires (UART).

4.4 PLC (Power Line Communication)

PLC technology is the innovative cornerstone of this project, setting it apart from other similar projects. Instead of using only wireless networks, climate data is transmitted via conventional electrical wiring already present in the area.

Advantages of using PLC:

- Eliminates the need for Wi-Fi coverage or LoRa network extension.
- Lower cost compared to long-range wireless solutions.
- Can be operated inside buildings or agricultural facilities without establishing a separate network.
- Transmission stability in closed industrial or rural environments.

Challenges of PLC technology:

1. Interference from other electrical devices.
2. The need for compatibility between the transmitter and receiver.
3. The need to improve signal quality to avoid data loss.

4.5Project implementation stages :

1. Design a pilot circuit on a breadboard.
2. Program the sensors and control unit and connect them to the communication unit.
3. Test each unit individually to ensure accurate measurements and transmission.
4. Integrate the wireless communication and PLC units and test the conversion between them.
5. Build a dashboard to visually display the data.
6. Complete environmental testing of the system in a realistic agricultural or outdoor environment.

Chapter Five: Data Analysis and Results

5.1 Reading and analyzing climate data:

The smart station in this project was developed to collect and analyze multiple climate data in real time, with a focus on providing accurate data to support the following areas:

- Smart agriculture
- Solar energy management
- Weather forecasting
- Environment and local climate
- Components used in measurement and analysis:

Sensor	Function
Atmospheric Pressure Sensor	Measures atmospheric pressure in **hPa** units
Temperature Sensor	Measures air temperature
Humidity Sensor	Calculates the ratio of water vapor in the air
Wind Speed and Direction Sensor	Monitors wind intensity and direction
Light/UV Radiation Sensor	Measures the intensity of sunlight or ultraviolet

Mechanism of action:

- The Arduino Mega module collects sensor readings in real time.
- The data is processed internally and converted into a transmittable format .
- The data is sent either via:
    1. Wi-Fi to a Raspberry Pi or a cloud server.
    2. Or via the PLC via electrical wiring to the processing unit, without the need for wireless networks.

5.2 Results based on technical analysis of the system

1. **Accuracy of atmospheric measurements**:

The performance of the selected sensors was analyzed based on technical data(Datasheet), and the results were summarized that the level of accuracy ranges as follows:

Table:

Accuracy rate of climate variables:

Variable	Accuracy Rate
Temperature	±0.5 °C
Relative Humidity	±3% RH
Atmospheric Pressure	±1 hPa
Wind Speed	±0.5 m/s

MQ-9 Gas Sensor Accuracy:

Detected Gas	Typical Detection Range (ppm)	Approximate Accuracy
Carbon Monoxide (CO)	10 – 1000 ppm	±10% – ±20%
Methane (CH_4)	200 – 10000 ppm	±10% – ±20%

Sensor Results by Season – Yearly Analysis:

This section presents the results obtained from the smart weather station sensors in Palestine over a full year, categorized by the four seasons. Key climate data was recorded monthly and analyzed to extract average values per season

Parameter	Average / Status
Temperature	23.2 °C
Relative Humidity	55%
Air Pressure	1012 hPa
Wind Speed	2.8 m/s
Solar Radiation	Moderate
Rainfall	Light Rain

Summer (June - August):

Parameter	Average / Status
Temperature	32.7 °C
Relative Humidity	41%
Air Pressure	1005 hPa
Wind Speed	2.0 m/s
Solar Radiation	High
Rainfall	Rare

Autumn (September - November):

Parameter	Average / Status
Temperature	24.1 °C
Relative Humidity	58%
Air Pressure	1011 hPa
Wind Speed	3.1 m/s
Solar Radiation	Moderate
Rainfall	Intermittent

Winter (December - February):

Parameter	Average / Status
Temperature	11.2 °C
Relative Humidity	72%
Air Pressure	1019 hPa
Wind Speed	4.2 m/s
Solar Radiation	Low
Rainfall	Heavy and Regular

These values are highly suitable for environmental and agricultural applications, and are considered highly accurate compared to the requirements of non-industrial use. 2. Wireless connection efficiency (Wi-Fi): When using a module such as the ESP32 or Raspberry Pi:

- Transmission speed up to 150 Kbps.
- Transmission interval: approximately every 60 seconds.
- Rely on stable Wi-Fi coverage.

This solution is suitable for environments where wireless internet is available, such as cities or advanced facilities.

3. Power Line Communication (PLC) efficiency:

- Transmission speeds up to 200 Kbps.
- Coverage range: up to 100 meters indoors.
- Can transmit data stably over electrical wires, even in the absence of wireless networks.

A very important feature that makes the project suitable for remote agricultural or industrial environments, and reduces the need for additional infrastructure.

4. Response time and data display:

- Reading time for all sensors: less than 1 second.
- Data processing and display on the LCD or Raspberry Pi is almost instantaneous.
- The graphical interface is updated periodically every minute or on demand.

5.3 Technical conclusions:

1. **Flexibility of communication:**

The combination of Wi-Fi and PLC technologies allows the system to be used in different locations depending on the infrastructure availability.

2. Reliability in challenging environments:

The system can operate efficiently in rural or agricultural areas where WiFi coverage is not available, thanks to Power Line Communication (PLC) technology.

3. Ease of use:

Through an LCD screen or a graphical interface on the Raspberry Pi, data can be displayed directly and clearly even to non-expert users, such as:

- Farmers
- Emergency teams
- Field workers

4.Next stage:

Real-world field tests will be conducted during Graduation Project 2 (Phase 2) to confirm:

- Measurement accuracy
- Communication efficiency in different environments
- The system's ability to operate continuously and for a long time

Chapter Six: Types of Smart Weather Stations

Smart weather stations have evolved into various forms, varying in components, accuracy, cost, and intended use . These stations can be classified into three main types:

6.1 Personal Weather Stations

Description:

They are miniature stations designed for individual or home use, often intended for hobbyists or regular users to monitor the weather conditions in their surroundings.

Features:

- Sold as ready-made units or manually installed.
- Typically measures temperature, humidity, wind speed, and rainfall.
- Relies on wireless connectivity (Wi-Fi or Bluetooth) to send data to phone or computer applications.
- Data is stored locally or on simple cloud services.

Limitations:

- Limited coverage.
- Lower accuracy compared to professional stations.
- Typically not compatible with other smart systems such as irrigation or cooling.

Uses:

- Personal and home use.
- Educational projects.
- Weather monitoring in gardens or villas.

6.2 Professional Weather Stations

Description:

These are high-precision stations used by institutions, research centers, or government agencies to monitor weather reliably and professionally.

Features:

- Contains precise, pre-calibrated sensors.
- Includes advanced data such as UV, evaporation, and environmental noise.
- Uses powerful transmission modules (such as LoRa, GSM, or PLC).
- Equipped with advanced central processing units and digital displays.
- Supports connectivity to cloud servers and advanced data analysis.

Limitations:

- High cost.
- Requires periodic maintenance and continuous calibration.

Uses:

- National meteorological stations.
- Research projects.
- Airports, ports, and major industrial areas.

6.3 Agricultural and Industrial Weather Stations

Description:

They are weather stations specifically designed for use in agricultural or industrial environments, and are often part of an integrated smart ecosystem.

Features:

Contains specialized sensors such as:

- Soil moisture
- Light intensity
- Carbon dioxide in the air
- Can be integrated with irrigation, cooling, and environmental control systems
- Uses flexible communication modules depending on the environment: Wi-Fi, LoRa, or PLC
- Low energy consumption and can be operated using solar energy.

Limitations:

- May be limited to functions outside the agricultural or industrial sector.
- Requires preparation and training for proper use.

Uses:

- Smart farms.
- Greenhouses.

Factories and plants sensitive to heat or humidity

Types of irrigation stations:

Type	Accuracy Level	Connectivity Type	Cost	Ease of Installation	Applications
Personal	Medium	Wi-Fi / Bluetooth	Low	High	Home and Educational
Professional	Very High	LoRa / GSM / PLC	High	Medium	National and Research-based
Agricultural/Industrial	High and Specialized	Wi-Fi / PLC / LoRa	Medium to High	Medium	Farms, Factories

Our project falls within these categories:

Our project falls under the third category (agricultural and industrial), as it:

- Supports sensors dedicated to the agricultural environment

(soil moisture, rain, light).

- Relies on flexible communication technologies, most notably data transmission over power lines (PLC).
- Can be expanded and integrated with smart irrigation or monitoring systems in the future.

Chapter Seven: How to Choose a Smart Weather Station

Choosing a smart weather station is an important technical decision that directly affects the quality of the collected climate data and its suitability for the intended use—whether domestic, agricultural, industrial, or research-related. This choice depends on several key criteria that must be carefully evaluated to ensure the desired performance, accuracy, and efficiency.

7.1 Basic Technical Criteria for Choosing a Weather Station

First: Nature of Use

The type of use determines the shape and design of the required station. Home use differs from agricultural or industrial use. From this, the following questions arise:

- Is the purpose of the station personal monitoring of weather conditions?
- Or to feed an intelligent irrigation system with environmental data?
- Or to collect precise data for scientific analysis or disaster prediction?

Second: Type of Sensors and Required Data

It is important to accurately determine the types of data the station should provide. The basic options include:

- Ambient Temperature
- Relative Humidity
- Barometric Pressure
- Wind Speed and Direction
- Rainfall Rate
- Light Intensity or Solar Radiation Air Quality (e.g., CO_2, PM2.5)

The greater the variety of data types, the more critical the accuracy of the sensors used becomes, and the more complex the system grows.

Third: Measurement Accuracy and Sensor Calibration

Accuracy is a decisive factor in professional or research applications.

The following should be considered:

- Permissible margin of error (±%)
- Type of output: analog or digital
- Support for manual or automatic calibration
- Certification of industrial or environmental standards (ISO or CE)

Fourth: Means of communication and data transfer

The connection type determines how and where data is sent. The most
prominent options are:

- Wi-Fi: Suitable for urban environments but requires coverage.
- LoRa: Excellent for rural or remote areas, offering wide range and low power consumption.
- Power Line Communication (PLC): The focus of our project. It allows data transmission via the power grid without the need for a separate network infrastructure, making it ideal for enclosed areas or regions with weak wireless coverage.

Fifth: Expandability and Integration

Can new sensors be added?
Is the station capable of integration with other smart systems like irrigation, cooling, or forecasting systems?
Future expandability is one of the most important indicators of the station's "sustainability" and long-term technical value.

Sixth: Data display and storage method

Options include:

- Built-in LCD display.
- Custom mobile app.
- Web dashboard.
- Local or cloud server.

It is important to ensure that the data can be easily accessed and analyzed, with reference to recorded climate history.

7.2 Cost-Performance Analysis

Higher price does not necessarily mean better performance. A careful comparison is required between:

- Cost of parts and components.
- Performance efficiency and data accuracy.
- Maintenance, upgrades, and technical support.

In our current project, we achieved an excellent balance between cost and performance by using reliable measurement units and dual communication modules (Wi-Fi + PLC). This enables the project to compete with expensive commercial stations without compromising quality.

7.3 Smart Weather Station Components Cost (USD)

(Component)	(Quantity)	(USD)	(USD)
Light Meter Sensor	1	34.79	34.79
Raspberry Pi Zero	1	21.43	21.43
Wind Speed & Direction Sensor	1	18.00	18.00
PLC Communication Module	2	26.48	52.96
LCD Display 400×1280	1	40.00	40.00
Arduino Mega	1	40.00	40.00
BME680 Sensor	1	21.33	21.33
Rain Drop Sensor	1	5.13	5.13
GAS Sensor	1	6.21	6.21
Wireless Communication Module	2	9.45	18.90
XL4016 DC-DC Buck Converte	1	10.00	10.00
SMPS Power Supply Unit (ATX)	1	60.00	60.00
Arduino Uno	4	10.00	40.00
Voltage Regulator /LM2596	1	10.00	10 .00
wires	20	10.00	10.00
.Voltage Converter XY-ITOV-4	1	40.00	40.00
(Control Cabinet)	1	80.00	80.00

Total price: $508.75

7.4 Technical summary for selecting the optimal station

Criterion	Low-Cost Station	Professional Smart Station	Project Station
Measurement Accuracy	Low to Medium	Very High	High
Number of Sensors	Limited	Extended	Flexible and Customizable
Communication Type	Wi-Fi / Bluetooth	GSM / LoRa	Wi-Fi + PLC
Power Supply	Electric Only	Solar + Battery	Electricity + Expandable
Display Interface	Simple App	Web + Built-in Screens	LCD Screen + Web Interface
Smart Integration	No	yas	Yes (Open Design)
Price	Low	High	Moderate – Smart

Chapter Eight: Installation and Maintenance of the Smart Weather Station

Installing the smart weather station is a pivotal stage in the success of the project. This process requires precise mounting, engineering planning for sensor distribution, and proper programming and communication settings. Additionally, regular maintenance is essential to ensure continuous operation and data quality. In this chapter, we will present in detail the installation steps, operational setup, and principles of optimal maintenance.

8.1 Installation and Connection Steps

First: Site Selection

The accurate performance of sensors depends on selecting the appropriate location:

- **The wind sensor** should be installed in an open area at a height of no less than 3–5 meters to avoid obstructions.
- **The rain sensor** needs an open, unshaded surface.
- **The light sensor** should be oriented toward the sun without any obstructions.
- **The temperature and humidity sensor** should be placed in a shaded, well-ventilated area to avoid the effects of direct radiation.
- **The soil sensor** should be installed below the surface at a suitable depth depending on the type of plant or study.

Second: Electrical Connections

- Connect the sensors to the control unit (Arduino or a similar board) through analog or digital input pins.

- Ensure stable power sources are used (5V or 3.3V depending on the sensor).
- Connect the PLC unit to the main power line through safe and well-planned points.

Third: Installing the Processing and Communication Unit

- Mount the Raspberry Pi or Arduino inside a sealed enclosure to protect it from dust and humidity.
- Connect the Wi-Fi module via the local network credentials or a hotspot.
- Link the PLC unit to the appropriate port in the internal power network, ensuring proper grounding and signal isolation.

8.2 Configuring Communication and Integration Settings

Wireless Communication (Wi-Fi or LoRa):

- Set the SSID and password for network access.
- Assign a static IP address (optional) to simplify access to the display interface.
- Test data transmission to a local server or open cloud platform

(Thing Speak or a dedicated platform).

Power Line Communication (PLC):

- Ensure the transmitter and receiver units are on the same electrical circuit.
- Test signal quality using monitoring software (PLC Configurator).
- Assign a unique identifier to each unit to avoid data overlap.

Integration with the Display Interface:

- Program the dashboard using Python or Node-RED.
- Configure a graphical interface to display real-time values.
- Include alerts in case of sudden changes (e.g., rainfall, temperature spike...).

8.3 Routine Maintenance and Troubleshooting

First: Preventive Maintenance

- Clean the sensors monthly from dust or contaminated rain. Check calibration every 3 to 6 months using reference values.

- Inspect electrical connections to prevent corrosion or poor contact.
- Update firmware or the display interface when necessary.

Second: Corrective Maintenance

- If data transmission stops, check the power source first.
- Inspect the control unit's input ports for wire damage or disconnection.
- Test the wireless signal or PLC unit to verify data transmission.

Third: Maintenance Log All maintenance and update activities should be documented in a regular log including:

- Date of the procedure
- Type of modification or repair
- Affected sensor
- Specific notes on performance before and after repair

Chapter Summary:

The success of a smart weather station relies not only on design accuracy or sensor efficiency, but also requires:

- Precise installation in a well-considered environment
- Professional-level software configuration
- Regular maintenance and proactive monitoring
- Smart management of PLC technology to avoid interference and ensure stable data transmission.

Chapter Nine: Building a Smart Weather Station (Practical Application)

9.1 Introduction

This chapter presents the practical implementation of a **Smart Weather Station** designed to measure various environmental parameters in real-time and transmit the data through **two parallel communication technologies**:

1. **Wireless Communication** using NRF24L01+ modules
2. **Power Line Communication (PLC)** using dedicated PLC modules

This dual-path transmission design increases system **reliability**, ensuring uninterrupted data delivery even in remote or infrastructure-limited areas.

9.2 Components Table of the Smart Weather Station Project

Smart Weather Station – Components List

Component	Quantity	Function / Use
Light Meter Sensor	1	Measures light intensity / solar radiation
Raspberry Pi Zero	1	Displays data and runs the graphical interface
Wind Speed & Direction Sensor	1	Measures wind speed and direction
PLC Communication Module	2	Transmits data over power lines (PLC)
LCD Display (400×1280)	1	Displays environmental data locally
Arduino Mega	1	Main control unit for reading sensor signals
BME680 Sensor	1	Measures pressure, temperature, humidity, and gases
Rain Drop Sensor	1	Detects rainfall
Gas Sensor (CO / CH_4)	1	Detects gas concentrations
Wireless Communication Module	2	Transmits data via Wi-Fi or LoRa
XL4016 DC-DC Buck Converter	1	Steps down voltage for system components
SMPS Power Supply Unit (ATX)	1	Provides stable power to the system
Arduino Uno	4	Assistive microcontrollers for PLC or relay

Voltage Regulator (LM2596)	1	Provides stable voltage for modules
Voltage Converter (XY-ITOV-4)	1	Converts voltage between levels
Control Cabinet	1	Houses and protects system components
Wires	20	Connect sensors, controllers, and power units

9.3 Smart Weather Station: Wiring and Practical Application

1. Wiring and Data Flow Structure

The smart weather station consists of several interconnected components responsible for sensing, processing, and transmitting climate data. The sensors are connected to an Arduino Mega, which collects readings and transmits them through intermediate units until reaching the Raspberry Pi for display.

System Flow:

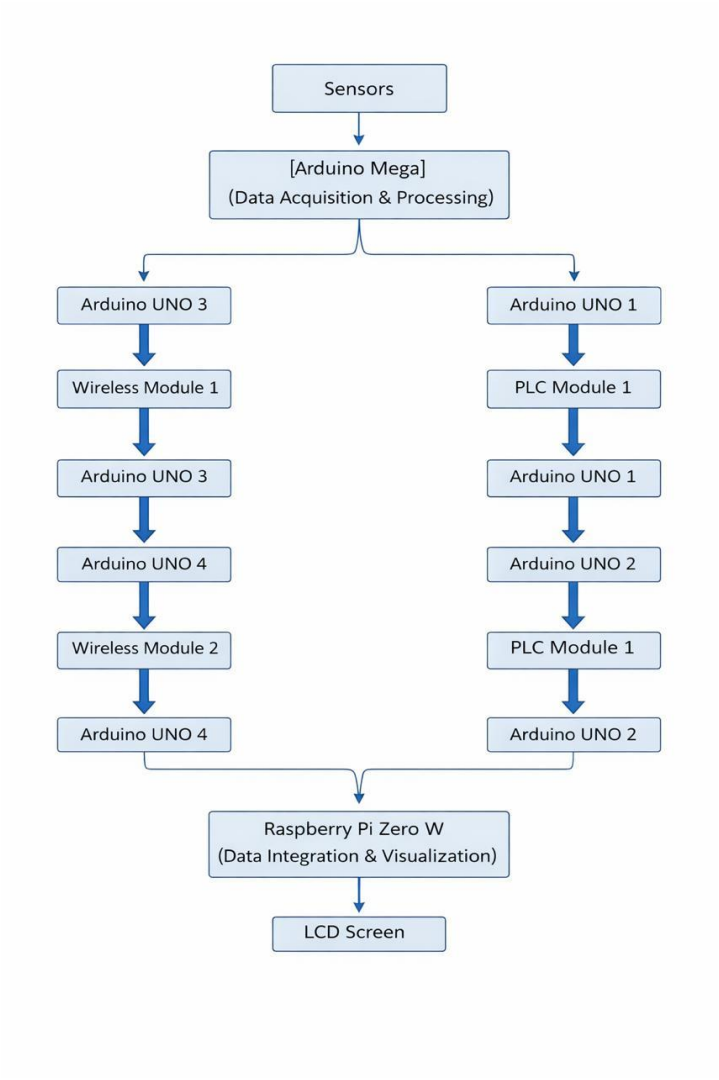


Figure 4.Q :Sustem flow

9.4 Project Objectives

- Build a system capable of real-time weather data acquisition and monitoring.
- Implement both **Wireless** and **PLC** data communication channels.
- Ensure continuous, redundant data transmission through multiple pathways.
- Power the system using a **single 12V source**, with regulated voltage levels for different modules.
- Enable future integration with **cloud services**, forecasting APIs, or smart irrigation systems.

9.5 System Overview

The system is structured into three functional levels:

Level 1 – Sensing Unit:

- A variety of sensors measure temperature, humidity, pressure, light, wind, rain, gas, and soil moisture.
- All sensors are connected to **Arduino Mega**, which serves as the main data aggregator.

Level 2 – Communication Network:

- Data from Arduino Mega is forwarded to:
  - **Arduino UNO 1** → communicates via **PLC**
  - **Arduino UNO 3** → communicates via **Wireless**

Each path transmits the data to:

- **UNO 2** (PLC receiver)
- **UNO 4** (Wireless receiver)

Level 3 – Data Display and Processing:

- The final data is received by **Raspberry Pi Zero W**
- Displayed on a panoramic **LCD screen**
- Optional: store or forward data to a server/cloud

9.6 Sensors Overview

Sensor Name	Measurement
BME680	Temperature, Humidity, Pressure *(no gas)*
MQ-9	Gas detection (CO, CH4, LPG)
MH-RD	Rain detection (presence & intensity)
Wind Direction Sensor	Wind direction
Anemometer	Wind speed
Soil Moisture Sensor	Soil moisture level
Light Sensor (Pyranometer)	Solar radiation / sunlight intensity

All sensors are connected directly to **Arduino Mega**, using a mix of analog, digital, and I2C communication protocols.

9.7 Communication Paths

Wireless Path:

1. Arduino Mega → Arduino UNO 3
2. UNO 3 → **Wireless Module 1** → **Wireless Module 2** → UNO 4
3. UNO 4 → Raspberry Pi Zero W

PLC Path:

1. Arduino Mega → Arduino UNO 1
2. UNO 1 → **PLC Module 1** → **PLC Module 2** → UNO 2
3. UNO 2 → Raspberry Pi Zero W

Both paths operate simultaneously. If one fails, the other can ensure continued operation.

9.8 Power Supply and Voltage Regulation System

The entire system operates from a **single 12V DC power source**. Different modules require various voltages (3.3V, 5V, 10V, 24V), so multiple **buck and boost converters** are used.

Voltage Distribution Table

Component	Required Voltage	Regulation Module Used	Notes
BME680 Sensor	5V	Direct	Measures Temp/Pressure/Humidity
MQ-9 Gas Sensor	5V	Direct	Analog output
MH-RD Rain Sensor	5V	Direct	Digital + Analog outputs
Wind Speed & Direction Sensors	24V	**LM2596 Boost + XY-ITOV Converter**	Output converted to voltage
Solar Radiation / Light Sensor	24V	**LM2596 Boost Converter**	Boosted from 12V
PLC Communication Module	10V	**XL4016 Buck Converter**	Stepped down from 12V
Wireless Module (NRF24L01+PA+LNA)	3.3V / 5V	Built-in regulator / external LDO	SPI connection
Raspberry Pi Zero W	5V	USB or dedicated 5V rail	Powers GUI and processing
LCD Panoramic Display	5V	Direct	Powered via 5V rail
Arduino Mega and UNO Boards	5V	Onboard regulator or 5V rail	Powered via Vin or USB

Using regulated converters like **LM2596** and **XL4016**, the system ensures clean and stable power delivery, preventing noise or voltage drops that could affect sensor readings or communication reliability.

9.9 Software Logic and Programming

- **Arduino Mega**:
  - Collects data from all sensors.
  - Formats and sends data via UART.
- **UNO Modules (1 to 4)**:
  - Act as transmitters and receivers for their respective paths.
  - Relay data without modifying the payload.
- **Raspberry Pi Zero W**:
  - Reads data from UNO 2 and UNO 4 (via Serial USB).
  - Displays sensor values using Python (Tkinter or Pygame).
  - Optional logging to CSV or cloud.

9.10 Challenges and Solutions

Challenge	Solution
Voltage mismatch between modules	Used step-up/down converters (LM2596, XL4016)
Unstable wireless signal	Used antennas + added CRC/retry logic
Sensor calibration (MQ-9, Soil)	Manual calibration with reference environments
Power line noise (PLC)	Added capacitors and signal filters
High voltage needs (24V sensors)	Boosted safely using regulated converters

---

9.11 Challenges and Solutions

Challenge	Cause	Solution
PLC signal fluctuation	Interference from electrical devices	Use signal filters and connect directly through a clean power outlet
Weak sensor readings	Humidity or poor wiring	Insulate wires and use well-secured sensor housings
Weak Wi-Fi coverage	Distance or wall barriers	Use PLC as an alternative channel or add a Wi-Fi extender

9.12 Scalability and Future Expansion

- Add advanced sensors (e.g., UV sensor, gas sensor).
- Integrate the system with smart irrigation or cooling systems.
- Support cloud storage and AI-powered data analysis.
- Enable a smartphone application to display real-time data.

9.13 Summary

This chapter presented the complete practical implementation of a robust and scalable **Smart Weather Station**, capable of sensing multiple environmental parameters and transferring the data through **dual redundant communication paths**.

The system is ideal for:

- Smart agriculture
- Environmental research
- Remote climate monitoring

With modular components and structured voltage planning, it is ready for future upgrades such as cloud integration, solar power, and mobile access dashboards.

Closing Word

At the conclusion of this work, we pause for a moment of reflection and gratitude.

This experience has been more than just an academic project; it has been a cognitive and human journey that we embarked on with passion. During this journey, we faced intellectual and technical challenges, and strove with all our effort and willpower to present work worthy of the trust of our professors, the support of our families, and the cooperation of our colleagues.

We have learned that achievement is not born from an idea alone, but from belief in it, the continuous pursuit of its realization, and the ability to transform theory into reality and plans into tangible steps.

We thank everyone who has had an impact along the way, even with a word of encouragement or a sincere prayer. We promise this project will be the first step in an academic and professional journey that we aspire to be filled with giving, innovation, and responsibility.

To everyone who reads these pages...

Believe in your ideas and work for them, for they may be the nucleus of real change.

Reference:

1. **An IoT based Low-cost Weather Monitoring System for Smart Farming**
   https://www.researchgate.net/profile/Narmilan-Amarasingam/publication/355183495_An_IoT_based_Lowcost_Weather_Monitoring_System_for_Smart_Farming/links/621ec71919ab0c3b4d2ca1de/An-IoT-based-Low-cost-Weather-Monitoring-System-for-Smart-Farming.pdf?utm_source=chatgpt.com

2. **This research paper presents a weather tracking system developed for agricultural purposes. The system utilizes various sensors and communication technologies to collect and analyze climate data in real time, providing farmers with accurate and timely information to help them make informed decisions about their crops.**
   https://www.irjmets.com/uploadedfiles/paper//issue_2_february_2023/33726/final/fin_irjmets1677164450.pdf?utm_source=.com

3. **These challenges serve as driving forces for the agricultural sector to shift toward smart farming by implementing IoT and big data solutions to enhance operational efficiency and productivity.**
   https://cdn.fbsbx.com/v/t59.2708-21/491049725_717136587646714_3211874799064736835_n.pdf/Smart_remote_sensing_weather_station.pdf?_nc_cat=106&_nc_cb=47395efc-74c935b2&ccb=17&_nc_sid=2b0e22&_nc_ohc=Xj8fc0zLTbgQ7kNvwHCf8Oe&_nc_oc=AdlVVl kqPUET1l8kXi0Nq0j3N-SoKo2mY_6TGU9gyZLQDeV5k6o9XZ8MNxu2A7j47ow&_nc_zt=7&_nc_ht=c dn.fbsbx.com&_nc_gid=a-

48ZREIAEb6P1vPBri_Pw&oh=03_Q7cD2gEMFL_Te24T0TbwS8SMdEHOtDL
-qMmMEjDdi68ZZxlN0Q&oe=6859322A&dl=1

4. **Weather Monitoring System using IoT for Smart Farming**
https://www.researchgate.net/publication/362429748_Weather_Monitoring_Syste m_using_IoT_for_Smart_Farming?utm_sourcet.com

5. IoT Applications in Smart Agriculture: Issues and Challenges
https://www.researchgate.net/publication/347833981_IoT_Applications_in_Smart _Agriculture_Issues_and_Challenges

6. https://cdn.fbsbx.com/v/t59.2708-21/491049725_717136587646714_3211874799064736835_n.pdf/Smart_remote_sensing_weather_station.pdf?_nc_cat=106&_nc_cb=47395efc-74c935b2&ccb=17&_nc_sid=2b0e22&_nc_ohc=lOes0HDfyYEQ7kNvwHvomAr&_nc_oc=AdlH5BoCq_u8G17i_WcxHFljcRbyYrRSIFg323uYpWC4_OO2nVnC4dnVbdiUK8MJTks&_nc_zt=7&_nc_ht=cdn.fbsbx.com&_nc_gid=PMItrENrwlLTWiwRzqzQZw&oh=03_Q7cD2gFxTZxbmv8Ll7TDCTpxVuZeeTbBrzXkL2Ptp3EwYDwNA&oe=685C0D6A&dl=1

7. https://cdn.fbsbx.com/v/t59.2708-21/468576400_1989961344835068_8029665850213132434_n.pdf/IoTenabled_Smart_Weather_Stations_Innovations_Cha.pdf?_nc_cat=104&_nc_cb=47395efc-74c935b2&ccb=1-7&_nc_sid=2b0e22&_nc_ohc=dhjsu51cTE4Q7kNvwEXvGdu&_nc_oc=AdmUAkl

U3MEGO8kDplHcK3EDkLHQljEnuZkpa_3qm1MTm8KjL4yOkm0Z85OPXAoKhnI&_nc_zt=7&_nc_ht=cdn.fbsbx.com&_nc_gid=VPjN4L5Vj-jrSXacFZ05g&oh=03_Q7cD2gHMZV4S8M9r3AhUJ6Ws-6OhY1GPEi-Iw06WOXVuUEuEVg&oe=685BE71F&dl=1

8. https://www.electronicsforu.com/electronics-projects/weather-station-using-stm32

9. https://www.instructables.com/Building-a-Weather-Station-WithSTM32F103C8T6/

10. https://www.shutterstock.com/image-photo/weather-station-green-tea-field-5g2284312609