SMART IRRIGATION SYSTEM USING LoRaWAN

A PROJECT REPORT

Submitted by

ALYANA ROJA RANI (220101130031)

in partial fulfillment for the award

of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SCHOOL OF ENGINEERING AND TECHNOLOGY

PARALAKHEMUNDI CAMPUS

CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT

ODISHA APRIL 2025

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
CENTURION UNIVERSITY OF TECHNOLOGY&MANAGEMENT,
PARALAKHEMUNDI-761211

BONAFIDE CERTIFICATE

Certified that this project report “***SMART IRRIGATION SYSTEM USING LoRaWAN ”*** is the Bonafide work of **“**ALYANA ROJA RANI” who carried out the project work under my supervision. This is tofurther certify to the best of my knowledge, that this project has not been carried out earlier in this institute and the university.

SIGNATURE	SIGNATURE
(N.Jeevaratnam)	**(External)**

Assistant Professor
Of
School of Engineering and Technology

Certified that the above mentioned project has been duly carried out as per the norms of the college and statutes of the university

SIGNATURE
(Prof. Prabhat Kumar Patnaik)
HEAD OF THE DEPARTMENT
Electronics And Communication Engineering

DEPARTMENT SEAL

DECLARATION

I hereby declare that the project entitled "**SMART IRRIGATION SYSTEM USING LoRaWAN**" submitted for the "EMBEDDED SYSTEM DOMAIN" of 6th semester B.Tech in Electronics and Communication Engineering is my original work and the project has not formed the basis for the award of any Degree / Diploma or any other similar titles in any other University / Institute.

Name of the Student:

Signature of the Student:

Registration No:

Place:

Date:

ACKNOWLEDGEMENTS

We wish to express my profound and sincere gratitude to Prof. N Jeevaratnam Department of Electronics And Communication Engineering, SoET, Parlakhemundi Campus, who guided me into the intricacies of this project nonchalantly with matchless magnanimity.

I thank Prof. Prabhat Kumar Patnaik, Head of the Dept. of Department of Electronics And Communication, SoET, Parlakhemundi Campus and Prof. Prafulla Kumar Panda, Dean, School of Engineering and Technology, Parlakhemundi Campus for extending their support during Course of this investigation.

I would be failing in my duty if I don't acknowledge the cooperation rendered during various stages of image interpretation by Prof.N Jeevaratnam

I am highly grateful to Prof.Rajesh Kumar Mishra who evinced keen interest and invaluable support in the progress and successful completion of my project work.

I am indebted to Assistant Prof.Rajesh Acharya for their constant encouragement, co-operation and help. Words of gratitude are not enough to describe the accommodation and fortitude which they have shown throughout my endeavor.

Name of the Student:

Signature of the Student:

Registration No:

Place:

Date:

ABSTRACT

This paper proposes a Smart Irrigation System utilizing LoRaWAN (Long Range Wide Area Network) technology for efficient water management in agriculture. Traditional irrigation methods often lead to wastage of water and energy due to imprecise scheduling and manual operation. In contrast, the proposed system integrates sensors, actuators, and LoRaWAN communication to enable real-time monitoring and control of irrigation processes. Soil moisture sensors collect data from the field, which is transmitted wirelessly to a central control unit via LoRaWAN. The control unit processes this data and determines the optimal irrigation schedule based on predefined parameters such as soil type, crop type, and weather conditions. Actuators then adjust the irrigation system accordingly, ensuring that crops receive the appropriate amount of water at the right time. By leveraging LoRaWAN's long-range and low-power capabilities, the system offers scalability and cost-effectiveness, making it suitable for deployment in large agricultural fields. Experimental results demonstrate the effectiveness of the proposed system in conserving water resources while improving crop yield and quality. Overall, the Smart Irrigation System presented in this paper represents asignificant advancement towards sustainable and efficient agriculture practices.

TABLE OF CONTENTS

LIST OF FIGURES

Fig.No **Page.No**

CHAPTER 1
INTRODUCTION

1.1 Background and Motivation

The background and motivation behind LoRa-based smart irrigation systems stem from the need for efficient and autonomous water management in agriculture. LoRa technology offers long-range, low- power wireless communication, ideal for remote areas where traditional maintenance-intensive systems fall short By utilizing sensors like DHT11 and soil moisture sensors connected to Arduino and ESP32, these systems enable real-time monitoring of soil conditions, temperature, and humidity, allowing for precise irrigation control The goal is to create a self-sustaining irrigation solution that optimizes water usage and crop health while reducing manual intervention, enhancing agricultural productivity in smart cities and remote locations.

Background:

- Traditional irrigation systems often rely on manual control or basic timers, leading to inefficient water usage and potential crop damage.
- The rise of the Internet of Things (IoT) and advancements in low-power wireless technologies like LoRa have enabled the development of smart, automated irrigation solutions.
- LoRa provides long-range, low-power communication capabilities, making it well-suited for agricultural applications in remote or hard-to-reach areas.

Motivation:

- Improve water use efficiency and conservation by automating irrigation based on real-time soil moisture and environmental data.
- Reduce labor costs and manual oversight required for traditional irrigation systems.
- Enhance crop health and yields by providing the right amount of water at the optimal times.
- Enable remote monitoring and control of irrigation systems, especially in areas with limited infrastructure.
- Contribute to the development of smart agriculture and precision farming techniques using IoT technologies.
- Address growing concerns about water scarcity and the need for sustainable water management practice.

1.2 Objectives:

1. Improve water use efficiency and conservation by automating irrigation based on real-time soil moisture and environmental data

2. Reduce labor costs and manual oversight required for traditional irrigation systems.

2. Enhance crop health and yields by providing the right amount of water at the optimal times

3. Enable remote monitoring and control of irrigation systems, especially in areas with limited infrastructure

4. Contribute to the development of smart agriculture and precision farming techniques

5. Address growing concerns about water scarcity and the need for sustainable water

Key Components:

1. Sensor Nodes:
 - Arduino Uno or ESP32 microcontroller
 - LoRa transceiver module (SX1278)
 - Soil moisture sensor
 - Temperature and humidity sensor (DHT11/DHT22)
 - Water pump or solenoid valve
2. Central Control Unit:
 - ESP32 (NodeMCU) microcontroller
 - LoRa transceiver module (SX1278)
 - LCD display
 - Cloud connectivity (WiFi)
3. Software:
 - Arduino IDE for programming sensor nodes
 - Blynk IoT platform for cloud integration and mobile app
 - Machine learning algorithms (e.g., Random Forest) for soil moisture prediction

1.2.1 ARDUINO BASE BOARD DETAIL

16*2 Lcd Display	=01
Dc Female Socket	=01
1000/25v Capacitors	=01
100/25v Capacitors	=01
7805 Regulators	=01
5mm Red Led	=01
2pin Male Connectors	=As Per Our ReQ
Uno Base Board	=01
1 K Resistor	=01
Male Bucks	=01
Female Bucks	=01
Arduino Micro Controller	=01

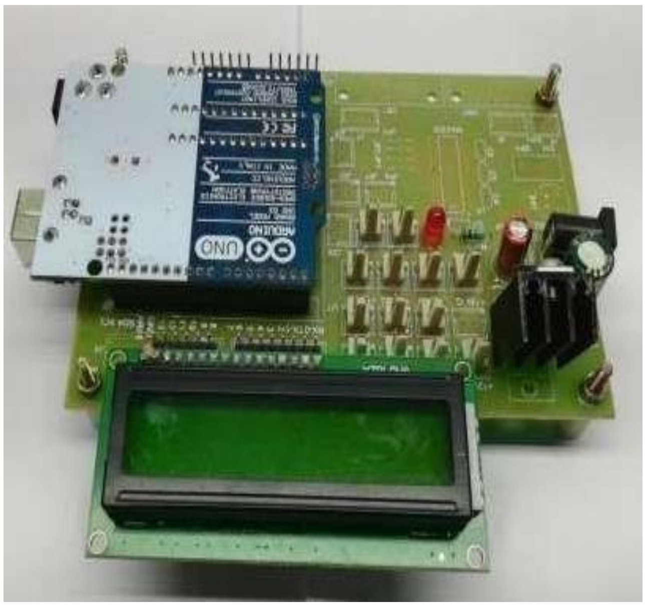


Fig 1.1

An Arduino base board is a microcontroller development platform designed for ease of use in prototyping and electronics projects. It typically features an Atmel AVR microcontroller, such as the ATmega328 on the popular Arduino Uno, along with a set of digital and analog input/output (I/O) pins that allow users to interface with sensors, actuators, and other devices. The board includes a USB interface for programming and serial communication, a voltage regulator for power management, and often an onboard LED for basic testing.
Arduino boards are supported by an open-source IDE and extensive community libraries, making them ideal for hobbyists, educators, and engineers to develop embedded systems without deep knowledge of low-level electronics or programming. In addition to its core components, an Arduino base board often includes features that enhance its versatility and ease of integration. Most boards support both 5V and 3.3V logic, allowing compatibility with a wide range of external modules and sensors. The board is typically powered through a USB cable or an external DC power source via a barrel jack, with built-in protection circuits to prevent damage from incorrect voltage input. Expansion is made simple through standardized pin layouts and stackable “shields”—add-on boards that provide extra functionalities like motor control, wireless communication (e.g., Wi-Fi, Bluetooth, LoRa), or GPS. Arduino boards also support serial communication protocols such as UART, SPI, and I2C, making them suitable for more complex embedded systems. Because of its open-source nature, there are many variants of Arduino boards tailored for specific applications, such as the Arduino Mega for projects requiring more I/O pins, or the Arduino Nano for compact installations.

1.2.1.1 Arduino Uno

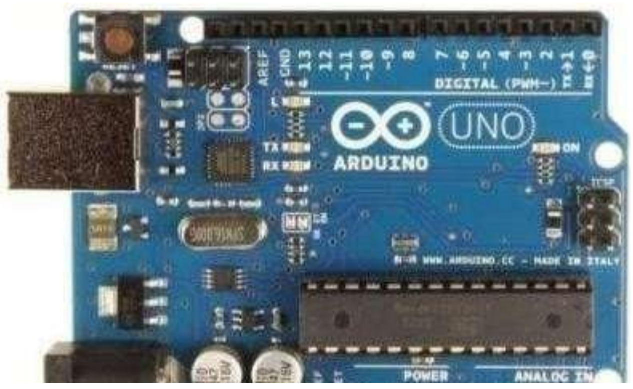


Fig 1.2

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everythingneeded to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs fromall preceding boards in that it does not use the FTDI USB- to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serialconverter.of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to putinto DFU mode.of the board has the following new features:

- ✓ pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage providedfrom the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- ✓ Stronger RESET circuit.
- ✓ Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a seriesof USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions.

1.2.1.2 Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows,a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data tobe sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).
A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.. For SPI communication,use the SPI library.

1.2.1.3 Programming

- ✓ The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno fromthe **Tools** > **Board** menu (according to the microcontroller on your board).
- ✓ The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload newcode to it without the use of an external hardware programmer. It communicates using the original STK500 protocol
- ✓ You can also bypass the bootloader and program the microcontroller through the ICSP (In-CircuitSerial Programming) .
- ✓ The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . TheATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- ✓ On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy)and then resetting the 8U2.
- ✓ On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground,making it easier to put into DFU mode.

1.2.1.4 Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line dropslong enough to reset the chip. The Arduino software uses this capability to allow you to upload code bysimply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well- coordinated with the start of the upload.
This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half- second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration orother data when it first starts, make sure that the software with which it communicates waits a secondafter opening the connection and before sending this data.
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the tracecan be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

1.2.1.5 USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extralayer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

1.2.1.6 Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board tobe attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

1.2.2 ESP32 Dev board Pinout, Specifications, datasheet & Schematic

ESP32 is a series of powerful, power-efficient, cheap microcontrollers that comes with integrated Wi-Fi and dual-mode Bluetooth. In this post, you will find ESP32 Dev board Pinout, Specifications, datasheet, and Schematic in detail.
The ESP32 series consist of a Tensilica Xtensa LX6 32-bit, dual-core microprocessor(has two processors) running at 240MHz and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. ESP32 is developed by a semiconductor-based company Espressif Systems in China, and it is manufactured by TSMC (Taiwan Semiconductor Manufacturing Company). It is a successor to their previous ESP8266 microcontroller module.

1.2.2.1 Introduction to ESP32 Development board

The details given here holds true for all development boards: NodeMCU ESP32-S board by AI Thinker, ESP32 DOIT board by Espressif, Sparkfun ESP32 Thing board, Adafruit ESP32 board, etc.
ESP WROOM-32 Chip: The ESP32 board comes with a powerful WROOM-32 module having capabilities like 802.11b/g/n Wi-Fi |BT 4.0|BLE. The chip consists of a dual-core processor and both can be controlled individually, operating at 240Mhz, 520KB of SRAM. It is compatible with Arduino IDE, LUA, MicroPython, etc.
Not all development boards expose all the pins on the chip. The ESP32 board(breakout board) shown here has all the pins of the ESP-WROOM-32 chip exposed. Whereas some ESP32 boards have only 30 or 36 pins exposed(available for external connection), although they have the same WROOM-32 chip installed in them.
Boot Button: This button is used to upload the new sketch/programs to the ESP32 microcontroller. During the upload, we have to HOLD down the Boot Button.

Micro USB: The board is powered using an onboard Micro USB port. This port can also be used to connect the board to the computer to upload programs.
RESET Button: This button is simply used to reset the board.
Touch Sensor: The ESP32 comes with 10 capacitive touch sensors connected to GPIO pins-

- Touch0 (GPIO4)
- Touch1 (GPIO0)
- Touch2 (GPIO2)
- Touch3 (GPIO15)
- Touch4 (GPIO13)
- Touch5 (GPIO12)
- Touch6 (GPIO14)
- Touch7 (GPIO27)
- Touch8 (GPIO33)
- Touch9 (GPIO32)

These GPIO pins can sense variations if it touches anything that conducts electricity like the human skin. So, when we touch the GPIO pin with our finger it generates this variation that is read by the sensor. We can use the touchRead(GPIO**)** function to read the value from the touch sensor.
Hall Effect Sensor: The board comes with another built-in sensor, a Hall effect sensor. Hall effect sensor is used to measure the change in the magnetic field in the surrounding.

1.2.2.2 ESP32 Dev board PINOUT

ESP32 Dev Board pin out: The ESP 32 board consists of a set of 38 pins that can be used to connect it with the external sensors and components. Out of these 38 pins, 25 pins are GPIO pins that can be used for a number of different functions

Power Pins: The board comes with two power pins – a 5V pin & a 3.3V pin. The 5V pin can be used to directly supply the ESP32 and its peripherals if you have a regulated 5V voltage source. While the 3.3V pin is the output of a voltage regulator (CP2102), it can be used to power up the external components.

GND: The ground pin of ESP32 is used to complete the circuit.

ADC (Analog to digital) Channels:

The board has 18, 12-bit SAR ADCs and supports measurements on 15 channels (analog enabled pins. ADC1_CH0 (GPIO 36)

- ADC1_CH1 (GPIO 37)
- ADC1_CH2 (GPIO 38)
- ADC1_CH3 (GPIO 39)
- ADC1_CH4 (GPIO 32)
- ADC1_CH5 (GPIO 33)
- ADC1_CH6 (GPIO 34)
- ADC1_CH7 (GPIO 35)
- ADC2_CH0 (GPIO 4)
- ADC2_CH1 (GPIO 0)
- ADC2_CH2 (GPIO 2)
- ADC2_CH3 (GPIO 15)
- ADC2_CH4 (GPIO 13)
- ADC2_CH5 (GPIO 12)
- ADC2_CH6 (GPIO 14)
- ADC2_CH7 (GPIO 27)
- ADC2_CH8 (GPIO 25)
- ADC2_CH9 (GPIO 26)

DAC (Digital to Analog) Channels: The board comes with two 8-bit DAC channels that can be used to convert digital signals into an analog voltage.

- DAC_1 (GPIO25)
- DAC_2 (GPIO26)

UART (Universal Asynchronous Receiver-Transmitter) Pins:ESP32 dev board has three UART interfaces, – UART0, UART1 and, UART2**,** which provide asynchronous communication between the UART-enabled devices up to a speed of 5 Mbps. UART interface in the dev board also consists of two extra pins that allow receiver and sender to alert each other of their current state – CTS and RTS signals pins.

UART0 Pins:

- U0 TXD (GPIO1)
- U0 RXD (GPIO3)
- U0 CTS (GPIO19)
- U0 RTS (GPIO22)

UART1 Pins:

- U1 TXD (GPIO10)
- U1 RXD (GPIO9)
- U1 CTS (GPIO6)

UART2 Pins:

- U2 TXD (GPIO17)
- U2 RXD (GPIO16)
- U2 CTS (GPIO8)
- U2 RTS (GPIO7)

SPI Pins on board:

- SPI_D (GPIO8)
- SPI_WP (GPIO10)
- SPI_HD (GPIO9)
- SPI_Q (GPIO7)
- SPI_CLK (GPIO6)
- SPI_CS0 (GPIO11)

HSPI Pins:

- V_SPI_ID (GPIO23)
- V_SPI_WP (GPIO22)
- V_SPI_HD (GPIO21)
- V_SPI_Q (GPIO19)
- V_SPI_CLK (GPIO18)
- V_SPI_CS0 (GPIO5)

VSPI Pins:

- HSPI_ID (GPIO13)
- HSPI_WP (GPIO2)
- HSPI_HD (GPIO4)
- HSPI_Q (GPIO12)
- HSPI_CLK (GPIO14)
- HSPI_CS0 (GPIO15)

FIG 1.3

PWM Pins:

The board comes with 25 PWM enabled pins (Nearly All GPIO pins) The PWM output can be used for driving digital motors and LEDs.

EN or Enable Pin:

EN stands for Enable, this pin is a 3.3 V regulator enable pin. When this is pulled LOW it resets the micro-controller.

1.3 LORA Transceiver ModuleRFM6601W Datasheet

1.3.1 Overview

The RFM6601W module includes a general LoRa wireless communication SoC, integrated witha RF transceiver and a 32-bit RISC MCU. The MCU uses ARM Cortex M4, with 48 MHz operation frequency. The RF Transceiver has continuous frequency coverage from 150 MHz to 960 MHz.The RFM6601W Module provides ultra-long range range and ultra-low power communication for LoRa application. RFM6601W can achieve a high sensitivity to -138 dBm and the maximum transmit power is up to +22 dBm. This makes the module suitable to be used in long range LoRa andhave high efficiency.

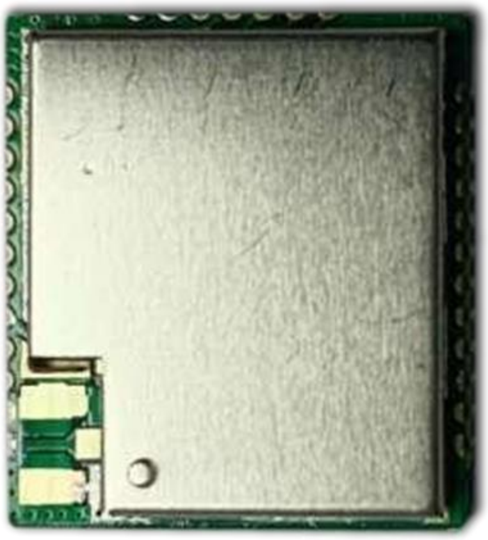


Fig 1.4

The RFM6601W is a high-performance, low-power LoRa transceiver module developed by HopeRF, specifically designed for long-range wireless communication in IoT applications. It operates across a wide frequency range (150 MHz to 960 MHz), supporting common LoRa bands like 433 MHz, 868 MHz, and 915 MHz. Powered by a 32-bit ARM Cortex-M4 MCU running at 48 MHz, it offers advanced features such as high receiver sensitivity (up to -138 dBm) and a transmission power of up to +22 dBm. The module communicates via SPI and supports various modulation schemes including LoRa, (G)FSK, BPSK, and (G)MSK. Despite its powerful capabilities, it is optimized for low energy consumption with a sleep current as low as 1.5 µA, making it suitable for battery-powered applications. Its compact SMD form factor (approximately 18 × 16 × 2.8 mm) allows easy integration into space-constrained devices. Common use cases include smart metering, environmental monitoring, smart agriculture, and asset tracking. With its robust design and versatile functionality, the RFM6601W is an ideal solution for developers building long-range, energy-efficient wireless systems.

Features

- Working Voltage: 2.4V - 3.7V
- Working Frequency: 433.92 MHz, 470 MHz, 868 MHz, 915 MHz
- Receiving Sensitivity: -138 dBm @SF=12, BW=125KHz
- Tx Current: 108mA @+22dbm, 433.92MHz
- Rx Current: 10mA @433.92MHz

Applications

- Smart meters
- Supply chain and logistics
- Building automation
- Agricultural sensors
- Smart cities
- Retail store sensors

1.3.2 Pin Diagram

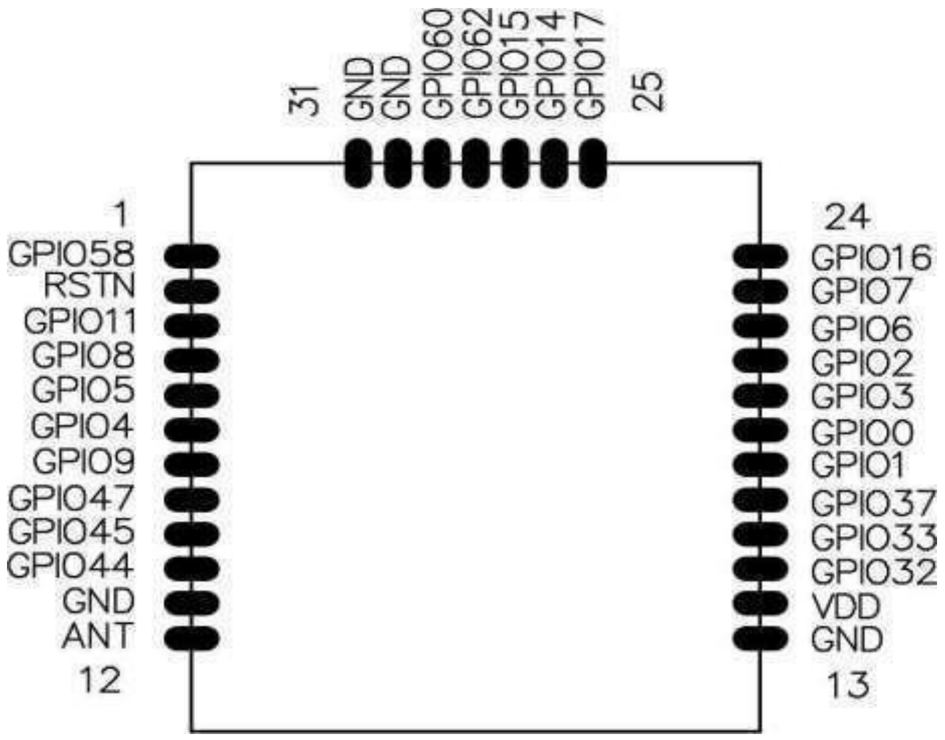


Fig 1.5

1.3.3 Pin Description

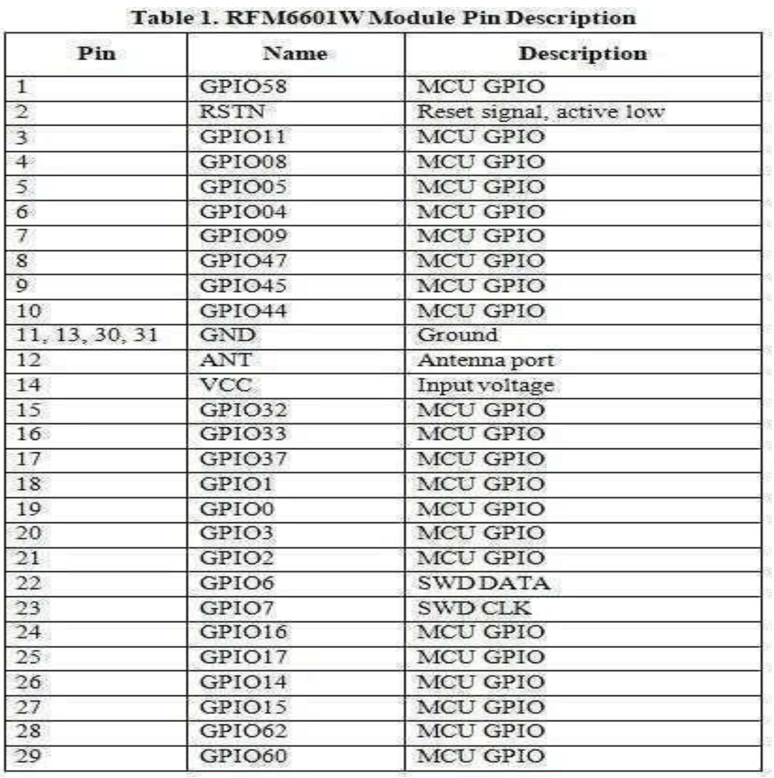


Table 1. RFM6601W Module Pin Description

Pin	Name	Description
1	GPIO58	MCU GPIO
2	RSTN	Reset signal, active low
3	GPIO11	MCU GPIO
4	GPIO08	MCU GPIO
5	GPIO05	MCU GPIO
6	GPIO04	MCU GPIO
7	GPIO09	MCU GPIO
8	GPIO47	MCU GPIO
9	GPIO45	MCU GPIO
10	GPIO44	MCU GPIO
11, 13, 30, 31	GND	Ground
12	ANT	Antenna port
14	VCC	Input voltage
15	GPIO32	MCU GPIO
16	GPIO33	MCU GPIO
17	GPIO37	MCU GPIO
18	GPIO1	MCU GPIO
19	GPIO0	MCU GPIO
20	GPIO3	MCU GPIO
21	GPIO2	MCU GPIO
22	GPIO6	SWD DATA
23	GPIO7	SWD CLK
24	GPIO16	MCU GPIO
25	GPIO17	MCU GPIO
26	GPIO14	MCU GPIO
27	GPIO15	MCU GPIO
28	GPIO62	MCU GPIO
29	GPIO60	MCU GPIO

Fig 1.6

1.3.4 Electrical Characteristics of LoRaWAN

Parameters	Symbol	Conditions	Min.	Typ.	Max.	Unit
Frequency	Fc	RFM6601W-433S2		433.92		MHz
		RFM6601W-470S2		470		MHz
		RFM6601W-868S2		868		MHz
		RFM6601W-915S2		915		MHz
Receiving Sensitivity	S	LORA Mode SF=12, BW=125KHz		-138		dBm
Workin g Voltage	V_{DD}		1.7	3.3	3.7	V
Rx Current	I_{RX}	433.92 MHz		10	11	mA
		470 MHz		10	11	mA
		868 MHz		10	11	mA
		915 MHz		10	11	mA
Tx Current	I_{TX}	433.92 MHz @+22dbm		108	120	mA
		470 MHz @+22dbm		108	120	mA
		868 MHz @+22dbm		120	135	mA
		915 MHz @+22dbm		120	135	mA
Sleep Current	I_{sleep}	Without RF and RTC		1.3	2	uA
Operating Temperature	T_{OP}		-40		+85	℃

Fig 1.7

1.3.5 Specification of LCD Module

An LCD (Liquid Crystal Display) module commonly used in embedded systems, such as the 16x2 or 20x4 character LCD, is a widely adopted display solution for showing alphanumeric characters and simple symbols. These modules are typically based on the Hitachi HD44780 controller or its compatible variants, which make them easy to interface with microcontrollers like the Arduino, ESP32, or STM32. The "16x2" model displays 16 characters per line over two rows, while the "20x4" version offers four rows of 20 characters each. Operating at 5V (or sometimes 3.3V), these modules have parallel data interfaces (4-bit or 8-bit mode) and use a standard 16-pin configuration that includes power, contrast adjustment (via a potentiometer), control lines (RS, RW, E), and data lines (D0–D7). Many versions also include a backlight, which improves visibility in low-light conditions and can be powered separately or controlled via a transistor. Their simple command set allows for cursor movement, display clearing, character shifting, and custom character creation. Due to their affordability, reliability, and ease of use, LCD modules are popular in projects requiring real-time visual feedback such as smart irrigation systems, DIY meters, and industrial controls.

1.3.5.1 Precaution for using LCD/LCM

General Precautions:

- ✓ LCD panel is made of glass. Avoid excessive mechanical shock or applying strong pressure onto the surface of display area.
- ✓ The polarizer used on the display surface is easily scratched and damaged. Extreme care should be taken when handling. To clean dust or dirt off the display surface, wipe gently with cotton, or other soft material soaked with isoproply alcohol, ethyl alcohol or trichlorotriflorothane, do not use water, ketone or aromatics and never scrub hard.
- ✓ Do not tamper in any way with the tabs on the metal frame.
- ✓ Do not make any modification on the PCB without consulting AMOTEC
- ✓ When mounting a LCM, make sure that the PCB is not under any stress such as bendingor twisting. Elastomer contacts are very delicate and missing pixels could result from slight dislocation of any of the elements.
- ✓ Avoid pressing on the metal bezel, otherwise the elastomer connector could be deformedand lose contact, resulting in missing pixels and also cause rainbow on the display.
- ✓ Be careful not to touch or swallow liquid crystal that might leak from a damaged cell. Any liquid crystal adheres to skin or clothes, wash it off immediately with soap and water.
- ✓ Soldering should be performed only on the I/O terminals.
- ✓ Use soldering irons with proper grounding and no leakage.
- ✓ Soldering temperature: 280°C±10°C
- ✓ Soldering time: 3 to 4 second.
- ✓ Use eutectic solder withresin flux filling.
- ✓ If flux is used, the LCD surface should be protected to avoid spattering flux.
- ✓ Flux residue should be removed.
- ✓ The viewing angle can be adjustedby varyingthe LCD driving voltage Vo.
- ✓ Since applied DC voltage causes electro-chemical reactions, which deteriorate the display, the applied pulse waveform should be a symmetric waveform such that no DC component remains. Be sure to use the specified operating voltage.
- ✓ Drivingvoltage should be kept within specified range; excess voltage will shorten display life.
- ✓ Response time increases with decrease in temperature.
- ✓ Displaycolor may be affected at temperatures above its operational range.
- ✓ Keep the temperature within the specified range usage and storage. Excessive temperature and humidity could cause polarization degradation, polarizer peel-off or generate bubbles.
- ✓ For long-term storage over 40 C is required, the relative humidity should be kept below 60%,and avoiddirect sunlight.

1.3.6 SERVER SPECIFICATIONS

- A modern multi-core CPU.
- 4 GB dedicated RAM with an extra 1.5 GB RAM for anyadditional language servers.
- Additional 4 GB of RAM for running the Analytics server.
- GB of free hard disk space in the installation directory.
- Development environment: HTML, CSS, JavaScript
- OS: Windows10 pro
- Processor speed : 2.9 GHz muclti core
- Type : Server client
- Packet request type : HTTP secured
- Data based : mysql
- Account security : Password protected
- Data Visulatlizaton: table and graphs
- Data Stored : 10Gb

1.3.7 Submersible pump and how it works

A submersible pump, also called an electric submersible pump, is **a pump that can be fully submerged in water**. The motor is hermetically sealed and close-coupled to the body of the pump. A submersible pump pushes water to the surface by converting rotary energy into kinetic energy into pressure energy. A submersible pump, also called an electric submersible pump, is a pump that can be fully submerged in water. The motor is hermetically sealed and close-coupled to the body of the pump.

A submersible pump pushes water to the surface by converting rotary energy into kinetic energy into pressure energy. This is done by the water being pulled into the pump: first in the intake, where the rotation of the impeller pushes the water through the diffuser. From there, it goes to the surface. The major advantage to a submersible pump is that it never has to be primed, because it is already submerged in the fluid. Submersible pumps are also very efficient because they don’t really have to spend a lot of energy moving water into the pump. Water pressure pushes the water into a submersible pump, thus “saving” a lot of the pump’s energy.

Fig 1.8

Also, while the pumps themselves aren't versatile, the selection certainly is. Some submersible pumps can easily handle solids, while some are better for liquids only. Submersible pumps are quiet, because they are under water, and cavitations is never an issue, because there is no "spike" in pressure as the water flows through the pump.

There are a few disadvantages with submersible pumps, and two have to do with the seal. The seals can become corroded with time. When that happens, water seeps into the motor, rendering it useless until it is repaired. Also, that seal makes the submersible pump a bit difficult to get into for repairs.

The other main disadvantage is that one pump does not fit all uses. Single stage pumps are used for most home and light industrial pumping. This includes aquarium filters, sewage pumping, or sump pumps for drainage. Multiple stage pumps are used for anything underground, such as water wells or oil wells.

Also, pumps are made to work with thin liquids like water, or thick ones like sewage.

Caution must be used with submersible pumps; they must be fully submerged. The water around asubmersible pump actually helps to cool the motor. If it is used out of water, it can overheat.

1.3.8 DHT 11 Humidity & Temperature Sensor

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital- signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability. and cost-effectiveness.DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration.The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connecand special packages can be provided according to users' request.

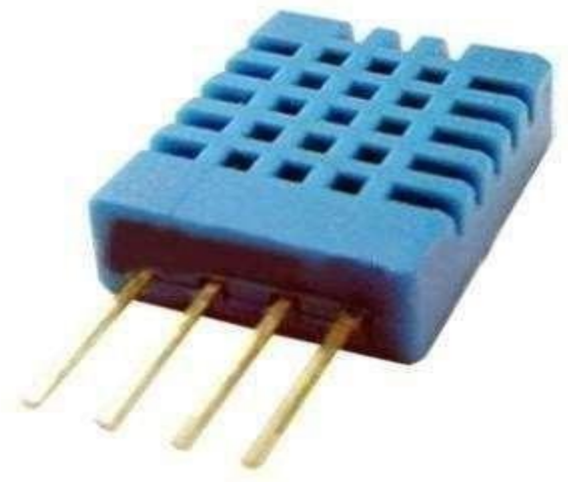


Fig 1.9

1.3.8.1 Detailed Specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃ ， 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
ResponseTime (Seconds)	1/e(63%)	6 S		30 S

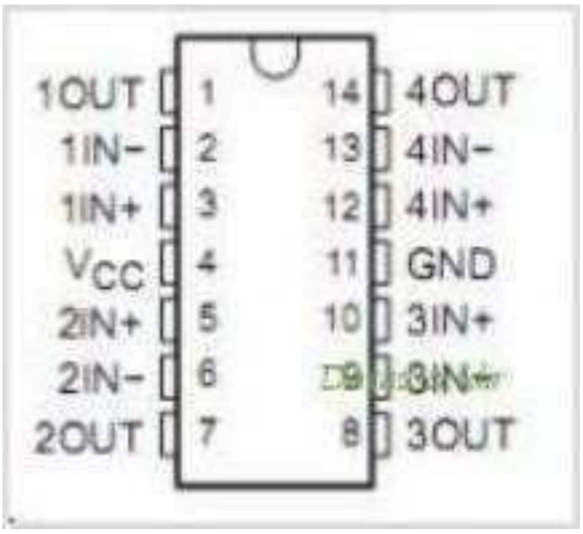
1.3.9 Soil Moisture Sensor LM324

The LM324 series consists of four independent, high gains; internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage.

Application areas include transducer amplifiers, DC gain blocks and all the conventional op amp circuits which now can be more easily implemented in single power supply systems. For example, the LM124 series can be directly operated off of the standard +5V power supply voltage which is used in digital systems and will easily provide the required interface electronics without requiring the additional ±15V power supplies.

1.3.9.1 PIN Diagram of LM324:

In the linear mode the input common-mode voltage range includes ground and the output voltage can also swing to ground, even though operated from only a single power supply voltage. The unity gain cross frequency is temperature compensated. The input bias current is also temperature compensated.



The LM124LM124/LM224/LM324/LM2902 Low Power Quad Operational Amplifiers series are op amps which operate with only a single power supply voltage, have true-differential inputs, and remain in the linear mode with an input common-mode voltage of 0 VDC.

Fig 1.10

The pinouts of the package have been designed to simplify PC board layouts. Inverting inputs are adjacent to outputs for all of the amplifiers and the outputs have also been placed at the corners of the package (pins 1, 7, 8, and 14). Precautions should be taken to insure that the power supply for the integrated circuit never becomes reversed in polarity or that the unit is not inadvertently installed backwards in a test socket as an unlimited current surge through the resulting forward diode within the IC could cause fusing of the internal conductors and result in a destroyed voltages can be easily accommodated and, as input differential voltage protection diodes are not needed, no large input currents result from large differential input voltages.

The differential input voltage may be larger than V+ without damaging the device. Protection should be provided to prevent the input voltages from going negative more than −0.3 VDC (at 25°C). An input clamp diode with a resistor to the IC input terminal can be used.

To reduce the power supply drain, the amplifiers have a class an output stage for small signal levels which converts to class B in a large signal mode. This allows the amplifiers to both source and sinks large output currents. Therefore both NPN and PNP external current boost transistors can be used to extend the power capability of the basic amplifiers. The output voltage needs to raise approximately 1 diode drop above ground to bias the on-chip vertical PNP transistor for output current sinking applications.

For ac applications, where the load is capacitive coupled to the output of the amplifier, a resistor should be used, from the output of the amplifier to ground to increase the class a bias current and prevent crossover distortion. Where the load is directly coupled, as in dc applications, there is no crossover distortion.

Soil Moisture Sensor

Features:

1. Internally frequency compensated for unity gain
2. Large DC voltage gain 100 dB
3. Wide bandwidth (unity gain) 1 MHz (temperature compensated)
4. Wide power supplyrange: Single supply 3V to 32V or dual supplies ±1.5V to ±16V
5. Very low supply current drain (700 μA)—essentially independent of supply voltage

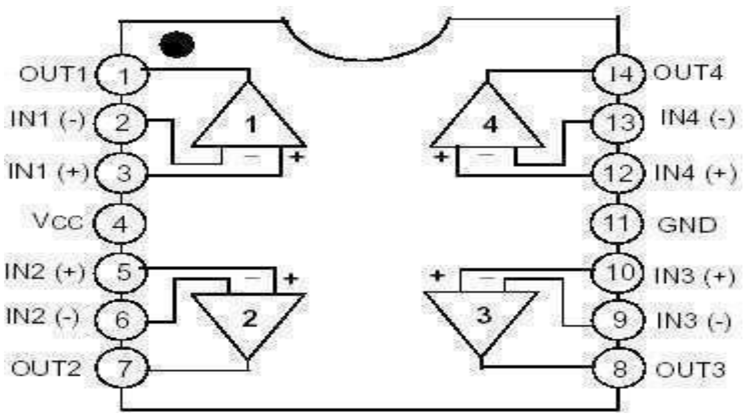


Fig 1.11

6. Low input biasing current 45 nA (temperature compensated)
7. Low input offset voltage 2 mV and offset current: 5 nA
8. Input common-mode voltage range includes ground
9. Differential input voltage range equal to the power supply voltage
10. Large output voltage swing 0V to V+ − 1.5V

Advantages:

1. Eliminates need for dual supplies
2. Four internally compensated op amps in a single package
3. Allows directly sensing near GND and VOUT also goes to GND
4. Compatible with all forms of logic
5. Power drain suitable for battery operation

1.3.10 Step Down Transformer

A Transformer is a static apparatus, with no moving parts, which transforms electrical power from one circuit to another with changes in voltage and current and no change in frequency. There are two types of transformers classified by their function: Step up Transformer and Step down Transformer.

A Step up Transformer is a device which converts the low primary voltage to a high secondary voltage i.e. it steps up the input voltage. A Step down Transformer on the other hand, steps down the input voltage i.e. the secondary voltage is less than the primary voltage.

The following images shows a simple demonstration of the use of Transformers (both Step up and Step down Transformers) in a typical Transmission System.

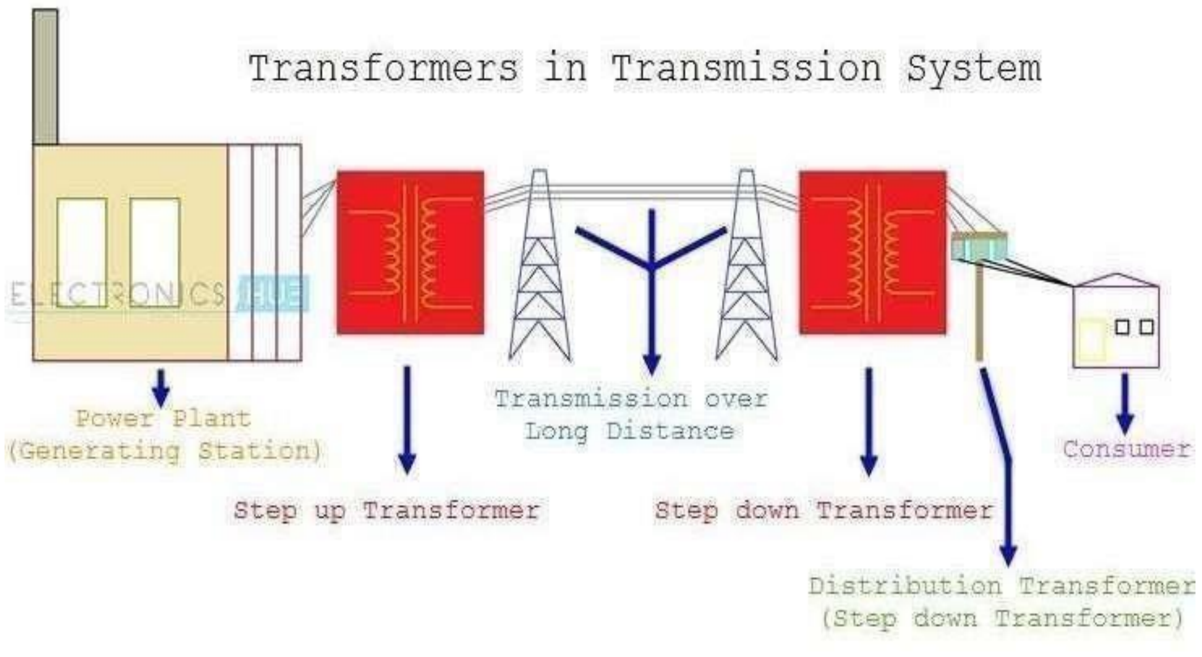


Fig 1.12

1.3.10.1 Real Time Application of Step Down Transformer

The voltage from the Power Plant or Generation Station is around 20kV. In order to transmit this voltage over long distances, it is stepped up to 440kV using a Step up Transformer. This voltage with increasedlevels is then transmitted to a distribution station.

At the distribution station, the 440kV is reduced to 11kV using a Step down Transformer. The voltage with decreased level is then made ready for consumer use.

Before going in to the details of the Step down Transformer, we will first see the working principle of a transformer in general.

Transformer Works?

- ✓ The first set of the coil, which is called as the Primary Coil or Primary Winding, is connected to an alternating voltage source called Primary Voltage.
- ✓ The other coil, which is called as Secondary Coil or Secondary Winding, is connected to the load and the load draws the resulting alternating voltage (stepped up or stepped down voltage)
- ✓ The alternating voltage at the input excites the Primary Winding, an alternating current circulates the winding. The alternating current will result in an alternating magnetic flux, which passes through the iron magnetic core and completes its path.
- ✓ Since the secondary winding is also linked to the alternating magnetic flux, according to Faraday's Law, anE.M.F is induced in the secondary winding. The strength of the voltage at the

  secondary winding is dependent on the number of windings through which the flux gets passed through.
- ✓ Thus, without making an electrical contact, the alternating voltage in the primary winding is transferred to the secondary winding.

CHAPTER-2
HW/SW REQUIREMENTS AND TECHNOOLGY

SMART IRRIGATION SYSTEM USING LORA ARCHITECTURE

2.1. LoRa Communication Network:

The Smart Irrigation System utilizes LoRa (Long Range) technology for its communication network. LoRa offers long-range communication with low power consumption, making it ideal for large-scale agricultural applications. LoRaWAN (LoRa Wide Area Network) protocols facilitate the communication between sensor nodes and the central control unit.

2.2. Sensor Nodes:

Sensor nodes are deployed throughout the agricultural field to collect relevant data for irrigation management.

2.2.1. Soil Moisture Sensors:

Soil moisture sensors are embedded in the soil at various locations within the field. These sensors measure the moisture content of the soil, providing real-time data on soil moisture levels.

2.2.2. Weather Sensors:

Weather sensors are installed to monitor environmental conditions such as temperature, humidity, and rainfall. This data is crucial for determining irrigation requirements based on weather patterns.

2.3. Central Control Unit:

The central control unit serves as the brain of the irrigation system, processing sensor data and making decisions regarding irrigation scheduling.

2.3.1. LoRa Gateway:

The LoRa Gateway serves as the interface between the sensor nodes in the field and the central control unit. It receives data from the sensor nodes via LoRa communication and forwards it to the central control unit for processing.

2.3.2. Data Processing and Decision-Making Requirements:

The central control unit processes the incoming sensor data to determine the optimal irrigation schedule. This process involves analyzing soil moisture levels, weather conditions, and other relevant factors to make informed decisions about when and how much water to irrigate.

2.4. Technologies Used:

- LoRa (Long Range) Communication Technology: Enables long-range communication between sensor nodes and the central control unit with low power consumption.
- LoRaWAN (LoRa Wide Area Network) Protocols: Facilitate communication and data transmission in the LoRa network.
- Soil Moisture Sensors: Measure soil moisture levels to determine irrigation requirements.
- Weather Sensors: Monitor environmental conditions such as temperature, humidity, and rainfall.
- Data Processing Algorithms: Analyze sensor data and make decisions regarding irrigation scheduling.
- Actuators: Controlthe irrigation system based on2t6he/ 2d1ecisions made bythe central control unit.

CHAPTER 3
SYSTEM DESIGN AND IMPLEMENTETION

3.1. Sensor Node Design:

3.1.1. Microcontroller:
The sensor node is equipped with a microcontroller responsible for interfacing with sensors, managing data collection, and controlling the LoRa transceiver. Popular microcontrollers such as Arduino or ESP32 may be utilized for this purpose.

3.1.2. LoRa Transceiver:
A LoRa transceiver module is integrated into the sensor node to facilitate communication with the central control unit. This module allows the sensor node to transmit sensor data wirelessly over long distances with minimal power consumption.

3.1.3. Sensor Integration:
Various sensors are integrated into the sensor node, including soil moisture sensors and weather sensors. These sensors measure relevant parameters such as soil moisture levels, temperature, humidity, and rainfall, providing essential data for irrigation management.

3.2. Central Control Unit Design:

3.2.1. LoRa Gateway:
The central control unit is equipped with a LoRa gateway, which serves as the interface between the sensor nodes in the field and the central processing system. The LoRa gateway receives data from the sensor nodes via LoRa communication and forwards it to the central processing unit for analysis.

3.2.2. Data Storage and Management:
Data storage and management systems are implemented within the central control unit to store and organize the incoming sensor data. This data may include historical sensor readings, weather forecasts, and irrigation schedules, facilitating efficient data analysis and decision-making.

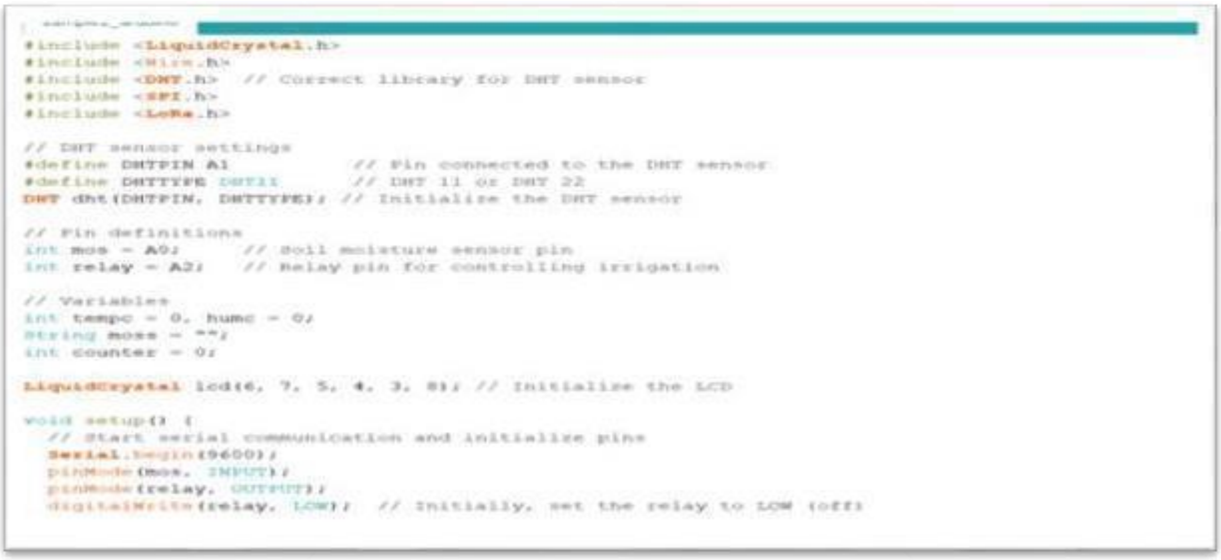
3.2.3. Irrigation Control Algorithm:
An irrigation control algorithm is implemented within the central control unit to determine the optimal irrigation schedule based on the sensor data and predefined parameters. This algorithm takes into account factors such as soil moisture levels, weather conditions, crop type, and water requirements to schedule irrigation events effectively.

By integrating these components into the system design and implementing robust algorithms, the Smart Irrigation System can effectively monitor soil conditions, analyze weather patterns, and optimize irrigation scheduling to conserve water resources and maximize crop yield.

CHAPTER 4
CODING IMPLEMENTATION

4.1 IMPLEMENTION PHASE:

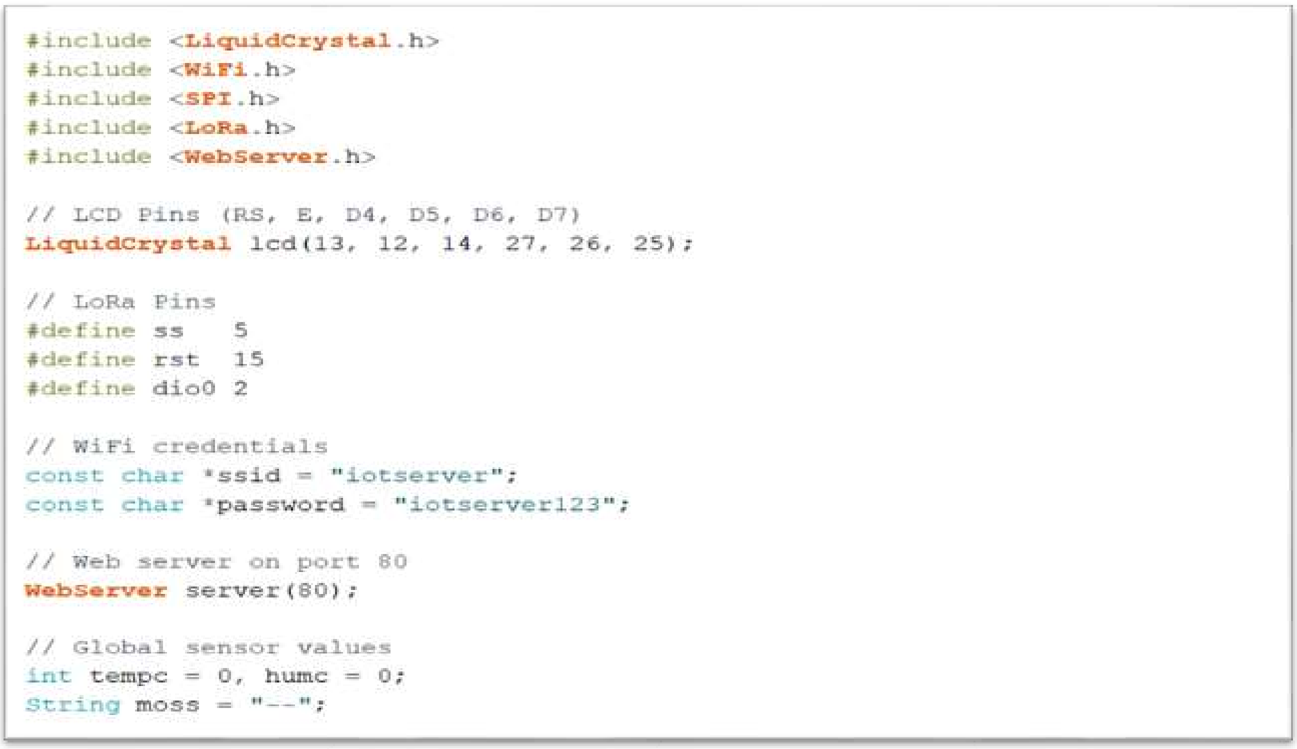
4.1.1 Transmitter Code



```
#include <LiquidCrystal.h>
#include <Wire.h>
#include <DHT.h>  // Correct library for DHT sensor
#include <SPI.h>
#include <LoRa.h>

// DHT sensor settings
#define DHTPIN A1             // Pin connected to the DHT sensor
#define DHTTYPE DHT11         // DHT 11 or DHT 22
DHT dht(DHTPIN, DHTTYPE); // Initialize the DHT sensor

// Pin definitions
int mos = A0;      // Soil moisture sensor pin
int relay = A2;    // Relay pin for controlling irrigation

// Variables
int tempc = 0, humc = 0;
String moss = "";
int counter = 0;

LiquidCrystal lcd(6, 7, 5, 4, 3, 8); // Initialize the LCD

void setup() {
  // Start serial communication and initialize pins
  Serial.begin(9600);
  pinMode(mos, INPUT);
  pinMode(relay, OUTPUT);
  digitalWrite(relay, LOW);  // Initially, set the relay to LOW (off)
```

Fig 4.13

4.1.2 Receiver Code



```
#include <LiquidCrystal.h>
#include <WiFi.h>
#include <SPI.h>
#include <LoRa.h>
#include <WebServer.h>

// LCD Pins (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd(13, 12, 14, 27, 26, 25);

// LoRa Pins
#define ss   5
#define rst  15
#define dio0 2

// WiFi credentials
const char *ssid = "iotserver";
const char *password = "iotserver123";

// Web server on port 80
WebServer server(80);

// Global sensor values
int tempc = 0, humc = 0;
String moss = "--";
```

Fig 4.14

4.2 Circuit Diagram

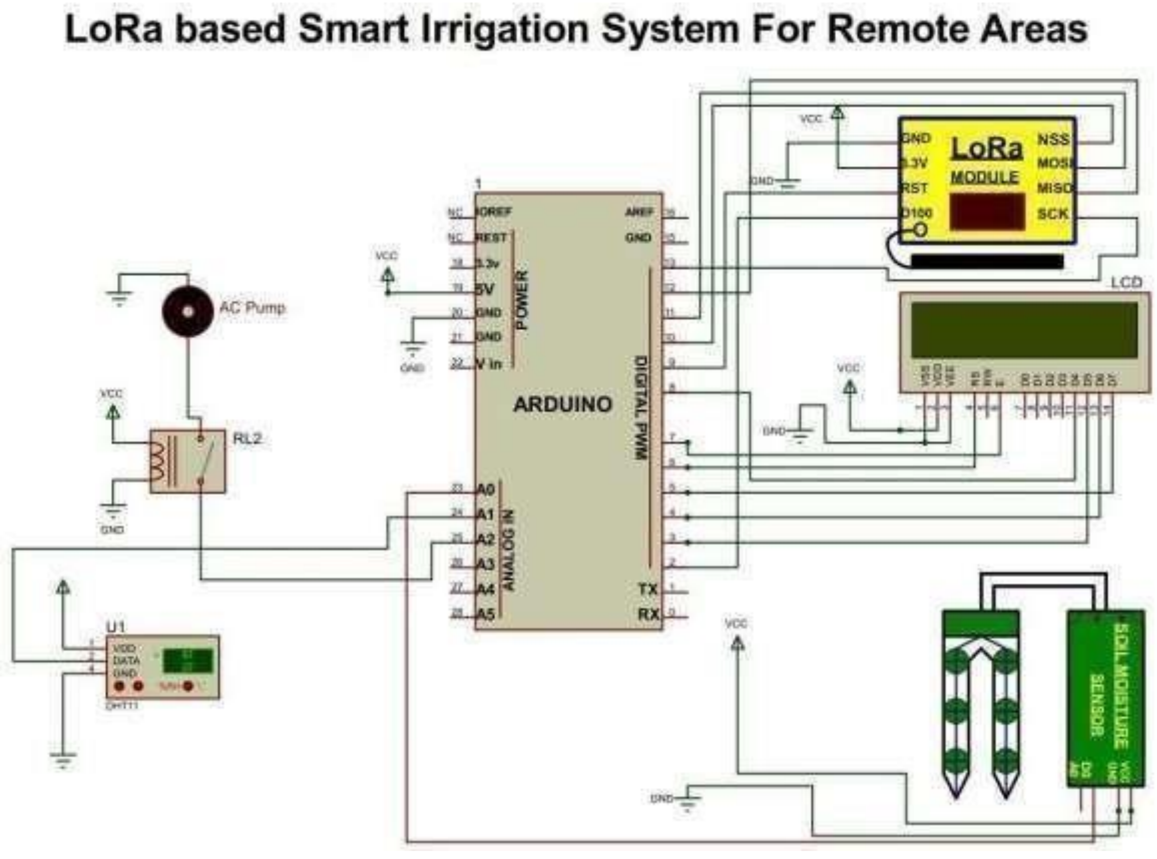


Fig 4.15

The image depicts a LoRa-based Smart Irrigation System for Remote Areas using an Arduino microcontroller. The system integrates key components including a soil moisture sensor, DHT11 temperature and humidity sensor, a relay-controlled AC water pump, a LoRa module for wireless data transmission, and an LCD display for local monitoring. The soil moisture sensor feeds analog data to the Arduino to determine the soil's water content, while the DHT11 sensor provides environmental data. When the moisture level drops below a defined threshold, the Arduino activates a relay (RL2) to switch on the AC pump and irrigate the soil. The LoRa module, connected via SPI, transmits real-time sensor data to a remote receiver for monitoring, such as on an ESP32-based dashboard. Additionally, the 16x2 LCD connected in 4-bit mode displays live sensor readings and system status locally. This setup is ideal for automating irrigation in agriculture, especially in remote areas where manual monitoring is difficult and long-range wireless communication is essential.

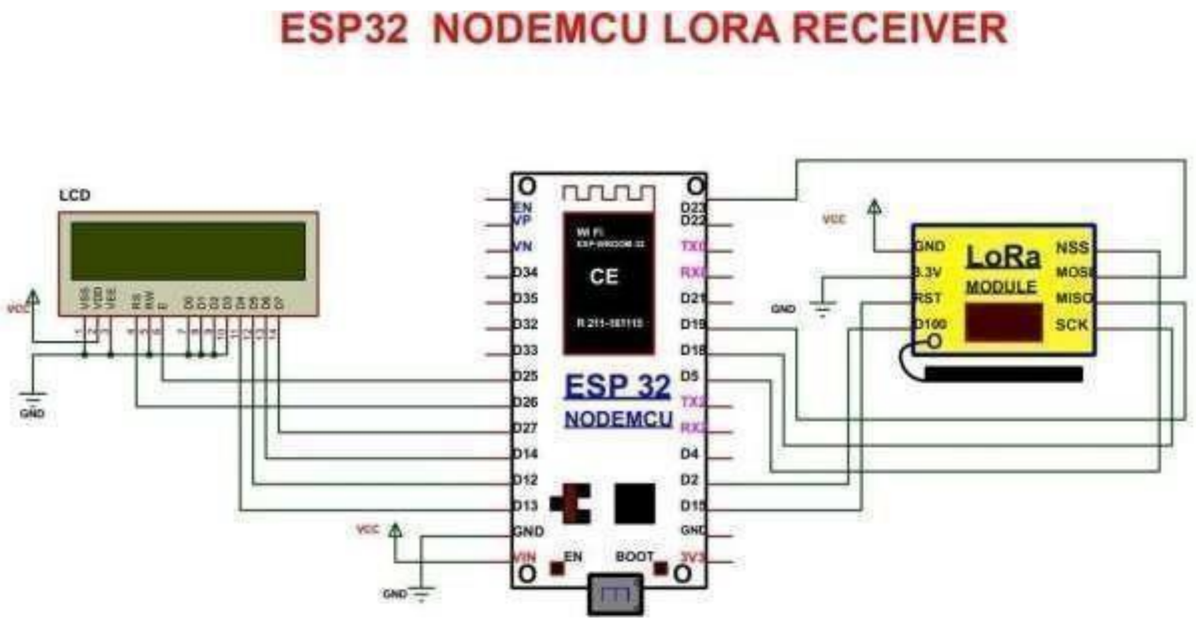


Fig 4.16

The image illustrates the circuit diagram for an ESP32 NodeMCU LoRa Receiver module designed to receive data wirelessly via LoRa communication and display it on an LCD. The central component is the ESP32 microcontroller, known for its built-in Wi-Fi and Bluetooth capabilities, which interfaces with both a LoRa module and a 16x2 LCD display. The LoRa module communicates with the ESP32 via the SPI protocol, using connections such as MOSI, MISO, SCK, NSS (CS), RST, and DIO0, powered by the 3.3V and GND pins of the ESP32. On the left, the LCD is connected in 4-bit mode, with RS, EN, and data pins (D4–D7) wired to GPIOs on the ESP32, and powered via Vcc and GND. This setup is typically used in remote sensing applications where data—like soil moisture, temperature, or humidity—sent from a distant Arduino-based transmitter (as in the earlier diagram) is received by this ESP32 LoRa receiver, then displayed locally on the LCD for real-time monitoring. This system is ideal for smart farming, allowing centralized, wireless updates from remote sensor nodes.

TX ARDUINO UNO

```
#include <LiquidCrystal.h>

#include <Wire.h>

#include <DHT.h> // Correct

library for DHT sensor #include

<SPI.h>

#include <LoRa.h>
// DHT sensor settings
#define DHTPIN A1        // Pin

connected to the DHT sensor #define

DHTTYPE DHT11 // DHT 11 or DHT

22

DHT dht(DHTPIN, DHTTYPE); // Initialize the DHT sensor
// Pin definitions
int mos = A0;        // Soil moisture sensor pin
int relay = A2; // Relay pin for controlling irrigation
// Variables
int tempc = 0, humc = 0;
```

```
Stringmoss = "";
    int counter = 0;
  LiquidCrystal lcd(6, 7, 5, 4, 3, 8); //
  Initialize the LCD  void setup() {
   // Start serial communication and
   initialize pins Serial.begin(9600);
   pinMode(mos,
   INPUT);
   pinMode(relay,
   OUTPUT);
   digitalWrite(relay, LOW); // Initially, set the relay to LOW (off)
   // Initialize LCD
   lcd.begin(16, 2);
   lcd.print("Lora
   Based Smart");
   lcd.setCursor(0, 1);
   lcd.print("Irrigation
   System");
   delay(3000);
   lcd.clear();
   lcd.print("For
   Remote Areas");
   delay(3000);
   // Initialize LoRa
   communication
   Serial.println("LoRa
   Sender");  if
   (!LoRa.begin(433E6
   )) {
```

```
 Serial.println("Starting  LoRa

 failed!");  lcd.clear();

 lcd.print("LoRa Failed");
 while (1); // Stay here if LoRa initialization fails
}


lcd.clear();
lcd.print("LoRa

Initialised");

lcd.setCursor(0, 1);

lcd.print("Successfu

lly");  delay(3000);

// Display labels

on the LCD

lcd.clear();

lcd.setCursor(0, 0);
lcd.print("T:"); // Temperature

label lcd.setCursor(8, 0);

lcd.print("H:"); //

Humidity label

lcd.setCursor(0, 1);

lcd.print("M:"); //

Moisture label

lcd.setCursor(8, 1);
```

```
 ld.print("P:"); // Pump
 label  dht.begin(); //
 Initialize DHT sensor
}
void loop() {
 // Read temperature and humidity
 from DHT sensor  humc =
 dht.readHumidity();
 tempc = dht.readTemperature();
 // Display temperature and
 humidity on the LCD
 lcd.setCursor(2, 0);
 convertl(tempc); // Display
 temperature lcd.setCursor(10, 0);
 convertl(humc); // Display humidity
 // Check soil
 moisture moss
 = "";
 if (digitalRead(mos)
  == LOW) {
  lcd.setCursor(2, 1);
  lcd.print("Wet ");
  digitalWrite(relay, LOW); //
  Turn off irrigation
  lcd.setCursor(10, 1);
  lcd.print("
  OFF ");
moss = "Wet";
```

```
}
if (digitalRead(mos)
 == HIGH) {
 lcd.setCursor(2, 1);
 lcd.print("Dry ");
 digitalWrite(relay, HIGH); //
 Turn on irrigation
 lcd.setCursor(10, 1);
 lcd.print("
 ON ");
 moss =
 "Dry"; }
// Send data via LoRa
every 200 cycles  if
(counter >= 200) {
 counter = 0;
 lcd.setCursor(14,  1);
 lcd.print("LS"); // Indicate data sending
 // Send packet via
 LoRa
 LoRa.beginPack
 et();
 LoRa.print(int(tempc)); // Send temperature data
 LoRa.print(",");
 LoRa.print(int(humc)); // Send
 humidity data  LoRa.print(",");
 LoRa.print(moss);           // Send moisture status
```

```
  LoRa.endPacket();
  delay(5000); // Wait for a while before sending the
  next packet  lcd.setCursor(14, 1);
  lcd.print(" ");  // Clear
 "LS"  }  counter++;  //
 Increment the counter
 delay(100); // Delay for
 LCD update}
// Function to convert values
to LCD format  void
convertl(unsigned int value)
{  unsigned int a, b, c, d, e,
f, g, h;
 a = value / 10000;
 b = value % 10000;
 c = b / 1000;
 d = b % 1000;
 e = d / 100;
 f = d % 100;
 g = f / 10;
 h = f % 10;
 a  =  a  |
 0x30;    c
 =     c    |
 0x30;    e
 =     e    |
 0x30;
 g = g | 0x30;
 h = h | 0x30;
 lcd.write(c); lcd.write(e); lcd.write(g); lcd.write(h);  // Display value on LCD
}
```

RX ESP32 NODEMCU 30 PIN

```
#include<LiquidCryst

al.h> #include

<WiFi.h> #include

<SPI.h>

#include  <LoRa.h>

#include<WebServe

r.h>


// LCD Pins (RS, E, D4, D5, D6, D7)

LiquidCrystal lcd(13, 12, 14, 27, 26, 25);

// LoRa Pins

#define ss  5

#define rst  15

#define dio0 2

// WiFi credentials
```

```
const char *ssid = "iotserver";
const char *password = "iotserver123";
// Web server on
port 80 WebServer
server(80);
// Global sensor
values int tempc =
0, humc = 0; String
moss = "--";  void
wifiinit() {
 WiFi.begin(ssid, password);
 while (WiFi.status() !=
 WL_CONNECTED) {
  Serial.print(".");
  delay(500);
 }
 Serial.println("\nWiFi Connected. IP: " + WiFi.localIP().toString());
}
void setup() {
 Serial.begin(1152
 00); lcd.begin(16,
 2);

 lcd.print("Connecting
 WiFi"); wifiinit();
```

```
lcd.clear(); lcd.print("WiFi
Connected"); delay(1000);
// LoRa setup
LoRa.setPins(ss, rst,
dio0); if
(!LoRa.begin(433E6))
{
  Serial.println("LoRa Init
  Failed!"); lcd.clear();
  lcd.print("LoRa Failed");
  while (1);
}
lcd.clear();
lcd.print("LoRa
Ready!");
delay(1000);

//
Displaylayou
t lcd.clear();
lcd.setCursor(0, 0); lcd.print("T:   H:");
lcd.setCursor(0, 1); lcd.print("M:  P:");

// Web server route
server.on("/", HTTP_GET,
[]() { String html =
R"rawliteral(
   <!DOCTYPE html><html><head>
   <meta name="viewport" content="width=device-width,  initial-scale=1.0">
   <meta http-equiv="refresh" content="5">
   <title>Smart Irrigation</title>
   <style>
```

```
 body { font-family: Arial; background: #f3f6f5; text-
 align: center; } h2 { background: #2e7d32; color: white;
 padding: 10px 0; margin: 0; }
 .box { margin: 20px auto; padding: 20px; background: #e0f7fa;
     width: 80%%; box-shadow: 0 4px 8px rgba(0,0,0,0.1);
     border-radius: 10px; }
 .label { font-size: 1.5em; margin:
 10px 0; } canvas { max-width:
 100%%; }
</style>
</head><body>
<h2>Smart Irrigation Dashboard</h2>
<div class="box">
 <div class="label"><strong>Temperature:</strong> %TEMPC% °C</div>
 <div class="label"><strong>Humidity:</strong> %HUMC% %%</div>
 <div class="label"><strong>Moisture:</strong> %MOSS%</div>
 <canvas id="chart" width="300" height="150"></canvas>
</div>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
 const ctx =
 document.getElementById('chart').getContext('2d');
 const chart = new Chart(ctx, {
  type:
   'line',                labels:
   Array(10).fill(''),          //
   dummylabels datasets: [{
   label: 'Temperature (°C)',
   borderColor: 'red',
    fill: false,
    data: Array(10).fill(%TEMPC%)
   }
```

```
   , {
```

```
 <script>
  const ctx =
  document.getElementById('chart').getContext('2d');
  const chart = new Chart(ctx, {
   type:
   'line',
   data: {
    labels: Array(10).fill(''), //
    dummylabels datasets: [{
    label: 'Temperature (°C)',
    borderColor: 'red',
     fill: false,
     data: Array(10).fill(%TEMPC%)
    }, {
     label: 'Humidity
     (%)', borderColor:
     'blue', fill: false,
     data: Array(10).fill(%HUMC%)
    }]
 </body></html>
)rawliteral";

html.replace("%TEMPC%",
String(tempc));
html.replace("%HUMC%",
String(humc));
html.replace("%MOSS%", moss);
```

```
r.send(200, "text/html", html);
    });

    server.begin();
    Serial.println("Web server started");
   }

   void loop() {
    server.handleClie
    nt();

    int packetSize=
    LoRa.parsePacket(); if
    (packetSize) {
     String data = LoRa.readString();
     Serial.println("LoRa Received: "
     + data);

     int i1 = data.indexOf(',');
     int i2 = data.indexOf(',', i1 + 1);
     if (i1 != -1 && i2 != -1) {
      tempc = data.substring(0, i1).toInt();
      humc = data.substring(i1 + 1,
      i2).toInt(); moss = data.substring(i2
      + 1);

      // LCD update
      lcd.setCursor(2
      , 0);
      lcd.print(tempc < 100 ? " " : ""); //
      alignment lcd.print(tempc);
      lcd.print(" ")
```

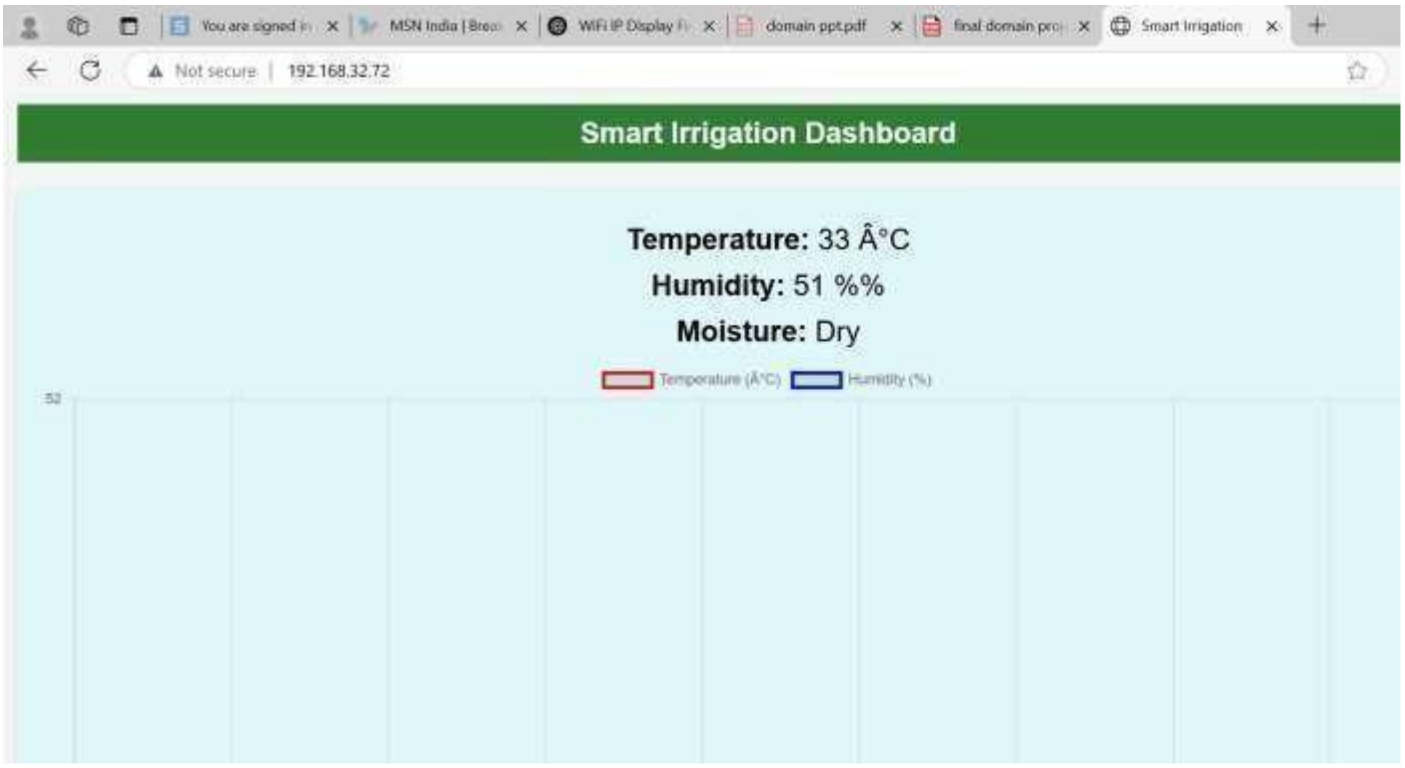
```
        server.send(200, "text/html", html);

            lcd.setCursor(8

            , 0);

            lcd.print(humc < 100 ? " "

            : ""); lcd.print(humc);

            lcd.print(" ");

            lcd.setCursor(2, 1);

            lcd.print(moss + " ");

            lcd.setCursor(8, 1);

            lcd.print((moss == "Dry") ? "ON " :
              "OFF");
  WiFi.begin(ssid,  password);

      delay(500);
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());

}

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

//LoRa based smart irrigation system for remote areas
  lcd.begin(16, 2);
  /*
  lcd.print("Lora Based
  Smart"); lcd.setCursor(0,1);
  lcd.print("Irrigation
  System"); delay(2500);
  lcd.clear();
  lcd.print("For Remote Areas");
  delay(1500);
  */
  Serial.print("Lora Receiver");

  wifiinit();
  lcd.clear();lcd.print("WIFI
    Connected"); delay(1000);
        //setup LoRa transceiver module LoRa.setPins(ss, rst,)
```

```
       delay(500);
      }

  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());

}

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

//LoRa based smart irrigation system for remote areas
  lcd.begin(16, 2);
  /*
  lcd.print("Lora Based
  Smart"); lcd.setCursor(0,1);
  lcd.print("Irrigation
  System"); delay(2500);
  lcd.clear();
  lcd.print("For Remote Areas");
  delay(1500);
  */
  Serial.print("Lora Receiver");

  wifiinit();
  lcd.clear();lcd.print("WIFI
    Connected"); delay(1000);

  //setup LoRa transceiver
  module LoRa.setPins(ss, rst,
  dio0);

  if(!LoRa.begin(433E6))
    {
     Serial.println("Starting LoRa
     failed!"); lcd.clear();lcd.print("LoRa
     Failed"); while(1);
    }

  lcd.clear();     lcd.print("LoRa     Initialised");
  lcd.setCursor(0,1);  lcd.print("Successfully");
  delay(1000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("T:");     //2,0
  lcd.setCursor(8, 0);
  lcd.print("H:");
  //10,0 lcd.setCursor(0,
  1); lcd.print("M:");
                         //2,1
```

```
 lcd.print("P:");    //10,1
}

char chrt;
unsigned long int
cntlmk=0; int cntk=0;

void loop()
{
  // try to parse packet
  int packetSize = LoRa.parsePacket();

  if(packetSize)
    {
      Serial.print("Received packet  '");
      // read packet
      //lcd.clear()
      ;
      loradata=""
      ; cntlmk=0;

  /*
      while(LoRa.available())
          {
           chrt = (char)LoRa.read();
           loradata += chrt;
           //Serial.print(chrt)
           ; lcd.write(chrt);
           cntlmk++;
           if(cntlmk == 16){lcd.setCursor(0,1);}
           if(cntlmk                                ==
           32){cntlmk=0;lcd.clear();} cntlmk=0;
          }
  */
      while(LoRa.available())
          {
           loradata = LoRa.readString();
           //lcd.print(loradata);

           //int
           tempc=0,humc=0;
           temps="";
           hums="";
           moss="";

           int i1 = loradata.indexOf(',');
           int i2 = loradata.indexOf(',',i1+1);

            temps = loradata.substring(0, i1);
             hums = loradata.substring(i1+1,
                         i2); moss  =
                  loradata.substring(i2+1);
     lcd.setCursor(2, 0);
           convertl(tempc);
           lcd.setCursor(10,
           0); convertl(humc);
           lcd.setCursor(2, 1);
           lcd.print(moss);
           if(moss == "Wet")
             {
             lcd.setCursor(10, 1);
             lcd.print("OFF");
             }
           if(moss == "Dry")
             {
             lcd.setCursor(10, 1);
             lcd.print("ON ");
             }
          }

       rssivalue = LoRa.packetRssi();
       Serial.print("' with RSSI ");
       Serial.println(rssivalue);

       LoRa.begin(433E6);delay(5000

       ); String pf_data = "";
       pf_data = servername + accountname + field1 + String(tempc) + field2 + String(humc) + field3 +
moss;
       Serial.println(pf_data

       ); http.begin(pf_data);

      // http.begin(servername + accountname + field1 + String(tempc) + field2 + String(humc) + field3
+ moss);

       httpResponseCode = http.GET();

    if(httpResponseCode>0)
      {
        payload="";
        Serial.print("HTTP Response code:
        "); Serial.println(httpResponseCode);
        payload = http.getString();
        Serial.println(payload);
      }
    else
      {
        Serial.print("Error code: ");
        Serial.println(httpResponseCode
        );
      }
       //Serial.println(servername + accountname + field1 + loradata + field2 + String(rssivalue));
     }
```

```
  delay(10);
  cntlmk++;
  if(cntlmk >
  12000)
    {cntlmk=0;
        ESP.restart()
        ;
    }

}


void convertl(unsigned int value)
{
  unsigned int a,b,c,d,e,f,g,h;

      a=value/10000;
      b=value%10000
      ; c=b/1000;
      d=b%1000;
      e=d/100;
      f=d%100
      ;
      g=f/10;
      h=f%10
      ;


      a=a|0x30;
      c=c|0x30;
      e=e|0x30;
      g=g|0x30
      ;
      h=h|0x30
      ;


   //lcd.write(a)
   ; lcd.write(c);
   lcd.write(e);
   lcd.write(g);
   lcd.write(h);
}
```

4.3 Web DashBoard



Smart Irrigation Dashboard

Temperature: 33 Â°C

Humidity: 51 %%

Moisture: Dry

Fig 4.17

The image shows a **web-based Smart Irrigation Dashboard** designed to monitor environmental parameters essential for efficient irrigation. It is hosted locally at the IP address 192.168.32.72, as seen in the browser's address bar. The interface presents real-time data including **temperature (33 °C)**, **humidity (51%)**, and **moisture status (Dry)**. The dashboard has a green header indicating its function and a light blue background for readability. Below the readings, a legend with red and blue boxes represents the **Temperature (°C)** and **Humidity (%)** respectively, suggesting that the system likely includes graphical data visualization, though the chart itself appears empty or not yet populated. Notably, there are minor display issues like the temperature unit rendered incorrectly as "Â°C" and the percentage symbol duplicated ("%%"), likely due to character encoding errors in the HTML or JavaScript code. This dashboard is part of an IoT-based smart irrigation system that helps monitor field conditions remotely, aiming to optimize water usage based on sensor inputs.

CHAPTER 5
CONCLUSION

The conclusion of a smart irrigation system using LoRaWAN technology would likely emphasize the benefits and effectiveness of such a system in optimizing water usage and enhancing agricultural practices. Here's a sample conclusion:

"In conclusion, the implementation of a smart irrigation system utilizing LoRaWAN technology presents a significant advancement in agricultural management. Through real-time monitoring and data- driven decision-making, this system enables precise and efficient water distribution, resulting in improved crop yields and resource conservation. By leveraging LoRaWAN's long-range, low-power capabilities, farmers can remotely control irrigation schedules, receive alerts for anomalies, and adapt strategies according to weather forecasts and soil moisture levels. Moreover, the scalability and cost- effectiveness of LoRaWAN networks make this solution accessible to a wide range of agricultural operations, from small-scale farms to large plantations.

The smart irrigation system using LoRaWAN technology provides an efficient and sustainable solution for modern agriculture by enabling long-range, low-power wireless communication between sensor nodes and a central receiver. The system continuously monitors soil moisture, temperature, and humidity using dedicated sensors, and displays real-time data on an LCD screen for local observation. Based on the soil moisture levels, the Arduino automatically controls a water pump via a relay, turning it on when the soil is dry and off when the moisture level is sufficient, thus ensuring optimal water usage and avoiding both under- and over-irrigation. The integration of LoRa modules allows data to be transmitted wirelessly over several kilometers to an ESP32-based receiver, which can be connected to the internet for cloud-based data logging and remote monitoring. This enables farmers to track environmental conditions and irrigation events from any location using a mobile or web application. The use of open-source hardware like Arduino and ESP32 makes the system cost-effective and easy to develop, while its modular design allows for scalability and future expansion. By conserving water, reducing manual labor, and supporting data-driven decisions, the system enhances crop productivity and promotes sustainable farming practices, especially in remote and drought-proneregions.

References

1. **M. Mehmood** – *Smart irrigation system using IoT and LoRa*
2. **H. T. Nguyen**– *Design and Implementation of a LoRa Based Smart Irrigation System*
3. **S. A. Shaikh**– *A LoRa-based Smart Irrigation System for Precision Agriculture*
4. **P. Rajalakshmi** – *Internet of Things Based Smart Irrigation Using LoRa*
5. **H. Sharma** – *Development of a LoRa-based Smart Irrigation System for Efficient Water Management in Agriculture*
6. **M. A. Shah et al –** LoRa Based Irrigation
7. **M. H. Le –** IOT Based on LoRaWAN
8. **A. M. Khedkar –** Smart Irrigation Using LoRa Module
9. **S. Sharma -** Developmemt of LoRa-based for efficient Irrigation

ASSESSMENT

Internal:

SL. NO	**RUBRICS**	**FULL MARK**	**MARKS OBTAINED**	**REMARKS**
1	Understanding the relevance, scope anddimension of the project	10		
2	Methodology	10		
3	Quality of Analysis and Results	10		
4	Interpretations and Conclusions	10		
5	Report	10		
	Total	**50**		

Date: **Sig/natureoftheFaculty :**

COURSE OUTCOME (COs) ATTAINMENT

➢ **Expected Course Outcomes (COs):**
(Refer to COs Statement in the Syllabus)

Gain and apply Knowledge about the architectural features and instructions of 32-bit ARM microcontrollers to develop the embedded system. CO2 Identify, analyze, formulate, develop and design various product-based applications based on Embedded Systems. CO3 A diversified team will learn, configure and build a customized Linux Kernel and also be able to set up and use the Cross Development platform, which will help them in Life long learning.

➢ **Course Outcome Attained:**
How would you rate your learning of the subject based on the specified COs?

☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**

LOW **HIGH**

➢ **Learning Gap (if any):**

NO GAP

➢ **Books / Manuals Referred:**

1.M.Mehmood – Smart irrigation System using IOT and LoRa

Date: **Signature of the Student**

➢ **Suggestions / Recommendations:**
(By the Course Faculty)

Date: **Signature of the Faculty**